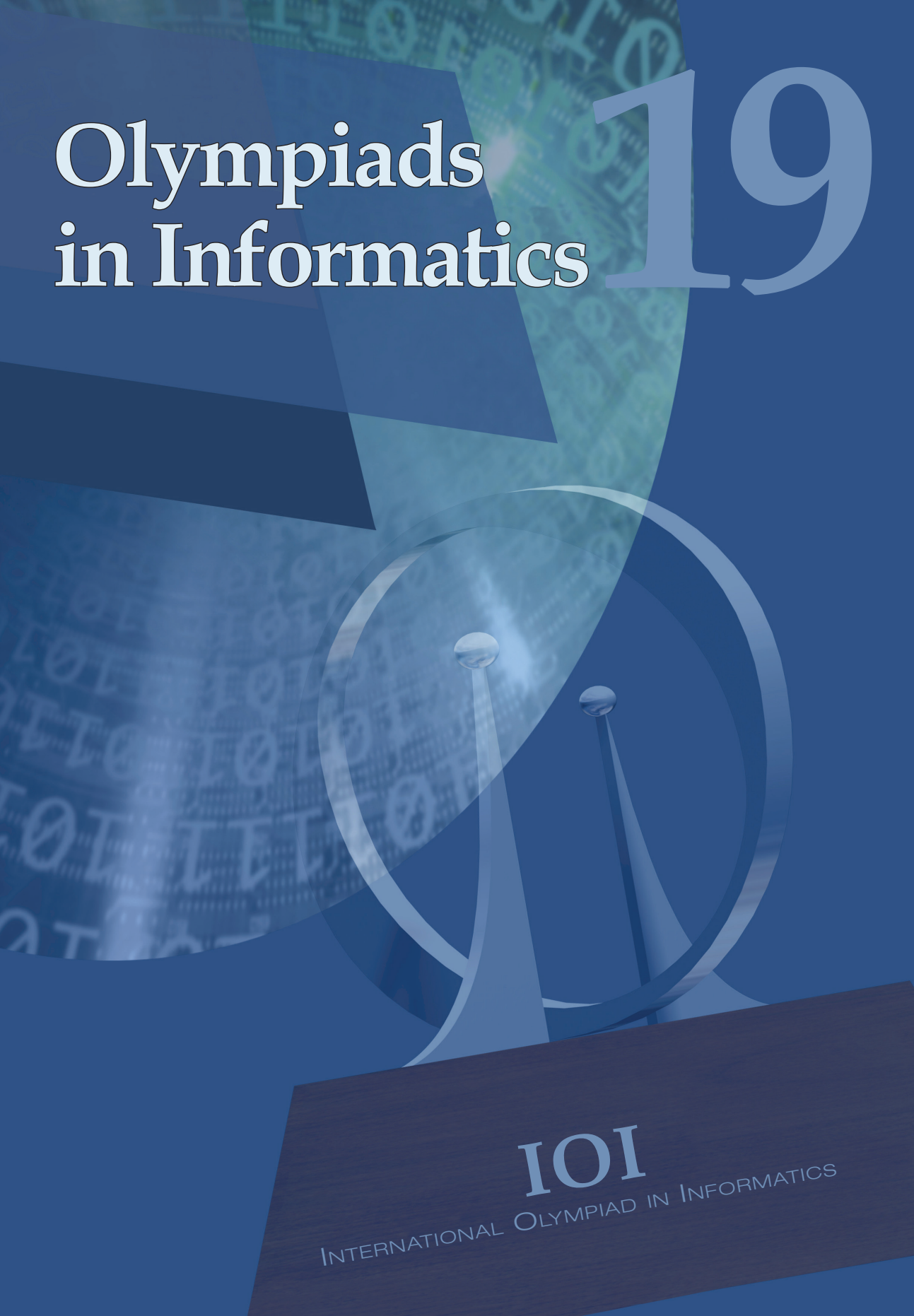
Olympiads in Informatics 19



IOI

International Olympiad in Informatics

ISSN 1822-7732

INTERNATIONAL OLYMPIAD IN INFORMATICS
VILNIUS UNIVERSITY

OLYMPIADS IN INFORMATICS

Volume 19 2025

Selected papers of
the International Conference joint with
the XXXVII International Olympiad in Informatics
Sucre, Bolivia, July 27 to August 3, 2025

OLYMPIADS IN INFORMATICS

The journal Olympiads in Informatics is an international open access journal devoted to publishing original research of the highest quality in all aspects of learning and teaching informatics through olympiads and other competitions.

`https://ioinformatics.org/page/ioi-journal`

ISSN 1822-7732 (Print)
2335-8955 (Online)

Olympiads in Informatics, 2025, Vol. 19, 1–25

DOI: 10.15388/ioi.2025.01

Interactive Problem Solving in the Classroom: Experiences with Turing Arena Light in Competitive Programming Education

Giorgio AUDRITO[1], Luigi LAURA[2], Alessio ORLANDI[3], Dario OSTUNI[4], Romeo RIZZI[5], Luca VERSARI[3]

[1]*University of Turin, Italy*
[2]*International Telematic University Uninettuno, Italy*
[3]*Google*
[4]*Università degli Studi di Milano, Italy*
[5]*Università di Verona, Italy*
e-mail: giorgio.audrito@unito.it, luigi.laura@uninettunouniversity.net, oalessio@google.com, dario.ostuni@unimi.it, romeo.rizzi@univr.it, veluca@google.com

Abstract. Turing Arena Light (TAlight) is a contest management system designed having in mind the needs of classroom teaching, rather than competitive programming contests. In TALight all problems are interactive by default. This means the contestant's solution for a problem will always interact with the problem, in real time. In this paper, we discuss our experience of using Turing Arena light in the course of Competitive Programming, an optional course offered to students of various LTs and LMs at the Department of Computer Science at the University of Verona. The course is meant to teach the students how to solve algorithmic problems, by teaching them the most common algorithmic techniques and data structures, and how to actively use them to solve problems.

Keywords: competitive programming, programming contest, competitive programming education.

1. Introduction

Programming contest management systems are the backbone of competitive programming events, handling everything from problem distribution and solution submission to automated judging and live scoreboarding (Revilla *et al.*, 2008; Maggiolo and Mascellani, 2012; Maggiolo *et al.*, 2014). Over the years, these systems have evolved from ad-hoc scripts and manual procedures into sophisticated platforms that emphasize security, scalability, and fairness (Leal and Silva, 2003). Traditional contest systems like the **Programming Contest Control System (PC^2)**, used in ACM ICPC since the 1990s,

enabled basic contest operations (login, submissions, judging interface) and were reliable for on-site contests. However, many early systems required judges to manually run solutions or provided limited automation.

Turing Arena Light (TALight) is a new contest management system that distinguishes itself by focusing on simplicity*, interactivity, and flexibility*. It was conceived as a lightweight platform geared toward educational use and practice environments rather than large-scale contests (`https://github.com/turingarena/turingarena`). TALight's design philosophy is to keep the core system minimal and conceptually simple, delegating most functionality to problem-specific modules. Uniquely, all problems in TALight are treated as interactive by default, meaning a contestant's solution gets connected to and interacts in with a problem manager program that provides inputs and checks outputs. The interaction takes place in real-time, while nothing prevents the constant to monitor the interaction in full while also print-debugging his code or logging everything in local. With this, it is quite handy to design interactive problems. And, even with standard input/output problems, the real commitment of the problem maker is to provide as much feedback as possible so that all of the students get their chance to actively learn and no one is cut out.

2. Related Work

Competitive programming systems can be classified into three main categories, each serving distinct purposes while sharing some overlapping features.

Contest Management Systems (CMS) are sophisticated platforms specifically designed for formal competitions like the International Olympiad in Informatics (IOI), ACM-ICPC, or national olympiads. Systems like DOMjudge (Pham and Nguyen, 2019), CMS (Maggiolo and Mascellani, 2012; Maggiolo *et al.*, 2014), and Kattis (Enstrom *et al.*, 2011) provide robust infrastructure for high-stakes, in-person events. They focus on security, reliability, and scalability to handle numerous concurrent submissions while maintaining fair evaluation conditions. These systems typically include features like real-time scoreboards, detailed analytics for judges, and stringent sandboxing mechanisms to ensure solution integrity. CMS platforms prioritize standardized evaluation environments where all participants compete under identical conditions with controlled resource limitations.

Online Judges serve a broader educational purpose by providing continuous access to problem-solving opportunities outside formal competitions. Platforms like Codeforces, LeetCode, and SPOJ host extensive problem libraries that users can attempt at their own pace. Unlike Contest Management Systems, they emphasize learning progression through difficulty-ranked challenges, detailed performance statistics, and community engagement via discussion forums and editorials. While they can host virtual contests, their primary value lies in self-directed practice. Many online judges incorporate gamification elements like ratings, badges, and streaks to motivate continued participation. Furthermore, since these systems have, in some cases, order of thousands of differ-

ent tasks, there is a vast literature related to the development of recommender systems able to suggest a suitable task depending on the learner's abilities (Audrito *et al.*, 2020; Fantozzi and Laura, 2020, 2021a, 2021b); also the problem of plagiarism is addressed (Iffath *et al.*, 2021). We refer the interested reader to the surveys of Wasik *et al.* (2019) and Watanobe *et al.* (2022).

Classroom Ad-hoc Tools like Turing Arena Light are specifically tailored for educational settings where pedagogical considerations outweigh competitive rigor. These systems prioritize ease of use, interactive problem types, and flexibility to accommodate diverse learning objectives. Unlike the standardized environments of CMS platforms, classroom tools often allow students to work in familiar development environments on their own machines. They typically feature simplified interfaces, immediate feedback mechanisms, and support for interactive problems that engage students through real-time interactions. While less suited for large-scale competitions, these tools excel at reinforcing classroom concepts and providing instructors with meaningful insights into student progress.

3. Turing Arena Light

In this chapter we introduce *Turing Arena light*, the spiritual successor of *Turing Arena* (`https://github.com/turingarena/turingarena`). *Turing Arena light* is a contest management system that is designed to be more geared towards the needs of classroom teaching, rather than competitive programming contests. It strives to be as simple[1] as possible, while being very flexible and extensible.

While we will discuss each point in more detail later, as an overview the design of *Turing Arena light* focuses on the following aspects:

- **Simplicity:** the design of *Turing Arena light* tries to keep things as simple as possible, while achieving the desired functionalities. While a meaningful objective metric for simplicity is hard to define, the current implementation of *Turing Arena light* consists of only 2197 lines of code (`https://github.com/romeorizzi/TALight`).
- **Interactivity:** in *Turing Arena light* all problems are interactive by default. This means the contestant's solution for a problem always interacts in real-time with the problem. In particular, a problem in *Turing Arena light* is defined by the problem *manager*, which is a program that interacts with the contestant's solution and gives a verdict at the end of the interaction. By being interactive by default, *Turing Arena light* allows a wider range of problems to be implemented with less effort, while not causing much overhead for non-interactive problems.
- **Flexibility:** *Turing Arena light* is designed to be able to run on all major operating systems, and allow solutions and problem managers to be written in any programming language, while still being able to guarantee a certain level of security. To

[1] Simple might mean very different things, in this context it is conceptual simplicity.

achieve this, *Turing Arena light* only consists of a small core written in Rust (Matsakis and Klock, 2014), whose main purpose is to spawn the process of the problem manager on the server, to spawn the process of the contestant's solution on its own machine, and to connect the standard input and output of the two processes. Thus, the contestants' code is never run on the server, and the problem manager can run without a sandbox, being trusted code written by the problem setter.

- **Extensibility:** as stated in the previous point, *Turing Arena light* only consists of a small core that has the fundamental role of spawning to processes and connecting their standard input and output. All the other functionalities are implemented by the problem manager itself, possibly using a common library of utilities. This allows the problem setter to implement any kind of problem, while still being able to use the same contest management system.

4. Architecture and Design

The fundamental idea behind *Turing Arena light* is to have two programs that talk to each other through the standard input and output channels. One of the two programs is the problem *manager*, which is a program that interacts with a solution to give it the input and evaluate its output, and eventually give a verdict. The other program is the *solution*, which is the program written by the contestant that is meant to solve the problem.

While this is not too far off from what other contest management systems do, the two main differences are that in *Turing Arena light* these two programs run on different machines, and the interaction between them is done in real-time. This is unlike mainstream contest management systems, where the two programs run on the same machine (like in DOMjudge (Eldering *et al*., 2020), CMS (Maggiolo and Mascellani, 2012) and Codeforces (`https://codeforces.com/`)), or where the interaction is not done in real-time (like in the old Google Code Jam (`https://codingcompetitionsonair.withgoogle.com/#codejam`) and Meta Hacker Cup (`https://www.facebook.com/codingcompetitions/hacker-cup`)).

In the following subsections we will discuss the components of *Turing Arena light* and how they interact with each other. We will start from the problem manager, going through the server and the client, and finally discussing the user interface.

4.1. *Problem Manager*

A problem in *Turing Arena light* is defined as a set of *services* and a set of *attachments*. A service is a program that can be spawned with a set of well-defined parameters, and that will ultimately interact with the solution. An attachment is a generic file that can be attached to the problem and downloaded by the contestant, such as the statement of the problem, or a library that the contestant can use in their solution.

A service defines which parameters it accepts, and the accepted values for each parameter. Parameters can be either strings or files. Each string parameter has a regular

```
%YAML 1.2
---
public_folder: public
services:
  free_sum:
    evaluator: [python, free_sum_manager.py]
    args:
      numbers:
        regex: ^(onedigit|twodigits|big)$
        default: twodigits
      obj:
        regex: ^(any|max_product)$
        default: any
      num_questions:
        regex: ^([1-9]|[1-2][0-9]|30)$
        default: 10
      lang:
        regex: ^(hardcoded|hardcoded_ext|en|it)$
        default: it
  help:
    evaluator: [python, help.py]
    args:
      page:
        regex: ^(free_sum|help)$
        default: help
      lang:
        regex: ^(en|it)$
        default: it
```

Fig. 1. Description file for a problem in *Turing Arena light.*

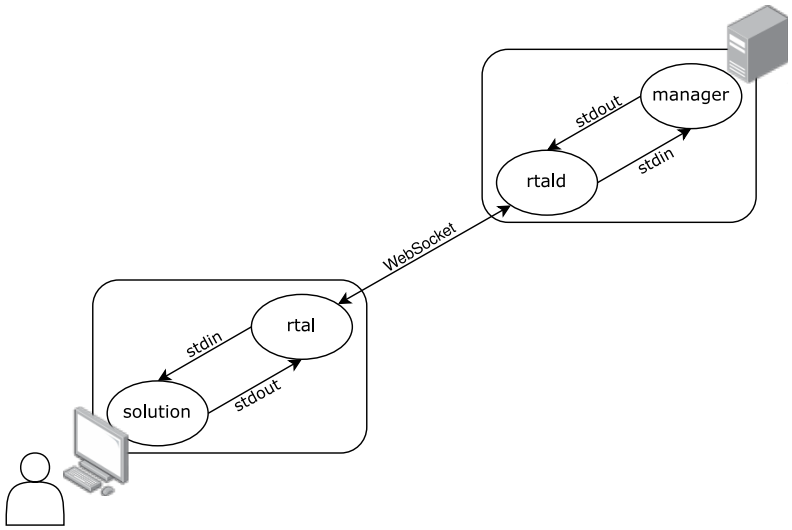


Fig. 2. Architecture of *Turing Arena light.*

expression that defines the set of accepted values and a default value. Furthermore, a service defines which program will be invoked with the given parameters: the problem *manager* (also called the *evaluator*). The attachments are just regular files in a folder on the file system.

The description of a problem is contained in a file called `meta.yaml`, which is a YAML (Ben-Kiki *et al*., 2009) file. The file contains the description of all the services, and their parameters, and the directory of the attachments. An example of a `meta.yaml` file is shown in Fig. 1. Thus, a problem in *Turing Arena light* is represented by a folder containing a `meta.yaml` file, and all the files and subdirectories needed for services and attachments.

4.2. *Server*

After the problem manager, there is the server. The server is the beating heart of *Turing Arena light*: its role is to accept incoming connections from the clients, spawn the problem manager corresponding to the client requested problem and service, passing to it the parameters specified by the client, and finally connect the standard input and output of the problem manager to the client.

Note that up to this point, *Turing Arena light* is merely a specification of how the problem is defined and how the interaction between the problem manager and the solution should happen. This opens up the possibility of having multiple implementations of the *Turing Arena light* framework, since the specification is very simple and does not require any particular technology, such as sandboxing.

Currently, there is only one implementation of the *Turing Arena light* framework, which is `rtal` (*Rust Turing Arena light*). It is written in Rust (Matsakis and Klock, 2014), and it is the reference implementation of *Turing Arena light*. The server component, `rtald`, is a small program that, given a folder containing problems, listens for incoming connections from the clients, and spawns the correct problem manager, and relays the standard input and output of the problem manager to the client via a protocol based on WebSockets (Fette and Melnikov, 2011).

4.3. *Client*

On the other side of the network[2] there is the client. The client is the program that the contestant runs on their machine to connect to the server and interact with the problem manager. Its role is to connect to the server, send the request for a problem and a service, send the string and file parameters for the service, and finally spawn and attach itself to the standard input and output of the solution running on the local machine of the contestant.

[2] Which might even be on the same machine, if both the server and the client are running on the same machine.

Once everything is up and running, the client will send the standard output of the solution to the server, which will relay it to the problem manager, and forward on the standard input of the solution all the incoming data from the server. Basically, the client is a proxy that connects the standard input and output of the solution to the server.

Like the server, there is also a `rtal` component for the client, also called `rtal`. This client component is a command line program that takes as parameters the address of the server, the problem and the service, and the parameters for the service. It also takes the command to run the solution. The client will then connect to the server, send the request for the problem and service, and spawn the solution with the given command, proxying the data between the solution and the server.

4.4. *User Interface*

As far as the contestant is concerned, what they must do is to write a solution to the problem in their favourite programming language. The only requirement is that it reads from the standard input and writes to the standard output. To read the problem statement, the contestant can download the attachments of the problem using the client. The client will download the attachments and save them on the local machine of the contestant.

Once the solution is ready, the contestant can run the client passing the right parameters, including the command to run their solution. The client will then connect to the server, send the request for the problem and service, and spawn the solution with the given command. Note that the solution is spawned and run on the local machine of the contestant, which means that the contestant has full freedom on which files it can read and write, which resources it can use, and so on. This is unlike other contest management systems that support real-time interaction, where the solution is run on a sandboxed environment on the server.

The ability to run the solution on the local machine opens to many possibilities. For example, the contestant can precompute some large set of data, save it on their machine, and then use it during the interaction with the problem manager to speed up the computation. Another example is the potential to use external libraries, multithreading, or even GPU computation. All of this is possible because the solution is run on the local machine of the contestant, where they have full control, and not on the server.

5. Implementation Details

As mentioned in the previous section, *Turing Arena light* currently has only one full implementation, which is *Rust Turing Arena light* (`rtal`). Like the name suggests, it is written in Rust (Matsakis and Klock, 2014). The choice of language was motivated by the fact that Rust is a systems programming language, and thus it is well suited for writing low-level programs that need to interact with the operating system and other pro-

grams. Furthermore, one key factor is portability: Rust is a compiled language whose compiled binaries require only minimal external dependencies to run, which makes it ideal to produce distributable binaries. This is important because *Turing Arena light* is meant to be used by students, which might not have the technical knowledge to install and configure a complex system. Having a single binary that can be downloaded and run without any configuration is a big advantage.

The implementation of *Turing Arena light* is split into three components: the server (`rtald`), the client (`rtal`), and the checker (`rtalc`). All three components share some common parts. The main one is the problem description definition, also known as the `meta.yaml` file. The definition can be found in Fig. 3. The definition is written using Rust structures which are then serialized to and deserialized from YAML using *serde* (`https://serde.rs/`), a serialization framework for Rust. `rtalc` is a small independent command-line program that takes as input a directory containing the problem description, and checks that the description is valid and matches the content of the directory. This is useful to check that the problem description is correct before uploading it to the server.

```rust
pub const META: &str = "meta.yaml";

#[derive(Debug, Clone, Serialize, Deserialize)]
pub struct Problem {
    pub name: String,
    pub root: PathBuf,
    pub meta: Meta,
}

#[derive(Debug, Default, Serialize, Deserialize, Clone)]
pub struct Meta {
    pub public_folder: PathBuf,
    pub services: HashMap<String, Service>,
}

#[derive(Debug, Default, Serialize, Deserialize, Clone)]
pub struct Service {
    pub evaluator: Vec<String>,
    pub args: Option<HashMap<String, Arg>>,
    pub files: Option<Vec<String>>,
}

#[derive(Debug, Serialize, Deserialize, Clone)]
pub struct Arg {
    #[serde(with = "serde_regex")]
    pub regex: Regex,
    pub default: Option<String>,
}
```

Fig. 3. Problem description definition in *Rust Turing Arena light*.

The two main jobs of the client and the server are process spawning and networking. For both of these tasks, `rtal` and `rtald` use the *tokio* (`https://tokio.rs/`) library, which is a framework for writing asynchronous programs in Rust. For the process spawning part, there is nothing particularly interesting: the server spawns the problem manager, and the client spawns the solution. They then, through *tokio*, manage the channels of the standard input and output of the spawned processes. All the internal communication within the server and the client is done using the actor threading model (Hewitt *et al.*, 1973; Hoare, 1978).

For the networking part, the communication protocol between the server and the client is based on WebSockets (Fette and Melnikov, 2011). The protocol definition is shown in Fig. 4. The protocol is based on JSON (Crockford, 2006) messages, which are serialized and deserialized using *serde*. These messages are then exchanged between the server and the client using WebSockets. The interaction between the server and the

```rust
pub const MAGIC: &str = "rtal";
pub const VERSION: u64 = 4;

#[derive(Serialize, Deserialize, Debug)]
pub enum Request {
    Handshake {
        magic: String,
        version: u64,
    },
    MetaList {},
    Attachment {
        problem: String,
    },
    ConnectBegin {
        problem: String,
        service: String,
        args: HashMap<String, String>,
        tty: bool,
        token: Option<String>,
        files: Vec<String>,
    },
    ConnectStop {},
}

#[derive(Serialize, Deserialize, Debug)]
pub enum Reply {
    Handshake { magic: String, version: u64 },
    MetaList { meta: HashMap<String, Meta> },
    Attachment { status: Result<(), String> },
    ConnectBegin { status: Result<Vec<String>, String> },
    ConnectStart { status: Result<(), String> },
    ConnectStop { status: Result<Vec<String>, String> },
}
```

Fig. 4. Network protocol definition in *Rust Turing Arena light*.

client is shown in Fig. 2. Using WebSockets enables a client of *Turing Arena light* to be implemented as a web application.

Both `rtal` and `rtald` run their spawned processes in an unsandboxed environment. This is done to avoid the complexity of sandboxing, but we argue that it does not pose a major security risk. The reason is that, for the client, the program being run is the contestant's own written solution, which is run on their local machine. Thus, the contestant has full control over the program, and can do whatever they want with it. For the server, the program being run is the problem manager, which is written by the problem setter. Thus, as long as the problem setter is trusted, there is no need to sandbox the problem manager. This is usually the case, as the problem setter is the one who also is responsible for the server where the `rtald` program is running. If this is not the case, then `rtald` can be run in a virtualized environment, such as a Docker container (Merkel *et al.*, 2014), to mitigate the risk of a bug in the problem manager that could cause unauthorized access to the server.

5.1. *Problem Manager Libraries*

So far we have discussed the architecture, the design and the implementation of Turing *Arena light*. However, we have not yet discussed how the problem manager is implemented. As mentioned in the previous sections, the problem manager is a program that interacts with the solution, and gives a verdict at the end of the interaction. The problem manager, just like the solution, has to communicate with its counterpart, which is the solution, using the standard input and output channels. Thus, the problem manager has full freedom on how to interact with the solution, as long as it does so using the aforementioned channels.

While this grants the problem maker a great deal of freedom, it also means that the problem maker has to potentially write a lot of boilerplate code each time they want to implement a new problem. To mitigate this problem, a problem maker can create a library of utilities that can be used to implement the problem manager. This library can be based on a particular style of problems, so that the problem maker can offer a consistent experience to the contestants.

In our case, we wrote a library called `tc.py`. A snippet of the library is shown in Fig. 5. This library allows to write a *old-Google-Code-Jam* like problem by only writing the code essential to the problem, and leaving all the boilerplate code to the library. What the manager has to implement is a function that generates a test case, and a function that evaluates the solution given by the contestant on a test case. The library will then take care of the rest, including enforcing the time limit, generating the right number of test cases, and assigning and storing the score for the solution. Note that with the *Turing Arena light* there is no way to enforce the memory limit, as the solution is run on the local machine of the contestant. However, the time limit can be enforced by measuring how much time passes between the sending of the input and the receiving of the output. While this is not a very precise measurement, it is good enough for distinguishing between solutions that have very different computational complexities.

```
class TC:
  def __init__(self, data, time_limit=1):
    self.data = data
    self.tl = time_limit

  def run(self, gen_tc, check_tc):
    output = open(join(environ["TAL_META_OUTPUT_FILES"], "result.txt"), "
        w")
    total_tc = sum(map(lambda x: x[0], self.data))
    print(total_tc, flush=True)
    tc_ok = 0
    tcn = 1
    for subtask in range(len(self.data)):
      for tc in range(self.data[subtask][0]):
        tc_data = gen_tc(*self.data[subtask][1])
        stdout.flush()
        start = time()
        try:
          ret = check_tc(*tc_data)
          msg = None
          if isinstance(ret, tuple):
            result = ret[0]
            msg = ret[1]
          else:
            result = ret
          if time() - start > self.tl:
            print(f"Case #{tcn:03}: TLE", file=output)
          elif result:
            print(f"Case #{tcn:03}: AC", file=output)
            tc_ok += 1
          else:
            print(f"Case #{tcn:03}: WA", file=output)
          if msg is not None:
            print(file=output)
            print(msg, file=output)
            print(file=output)
        except Exception as e:
          print(f"Case #{tcn:03}: RE", file=output)
          print(file=stderr)
          print("".join(traceback.format_tb(e.__traceback__)), e, file=
              stderr)
        tcn += 1
    print(file=output)
    print(f"Score: {tc_ok}/{total_tc}", file=output)
    output.close()
```

Fig. 5. Snippet of the python version of the competitive-programming like problem manager library for *Turing Arena light*.

As the name suggests, the `tc.py` library is written in Python (Van Rossum *et al.*, 2007), and it is meant to be used with problem managers written in Python. This works great for problems where the optimal solution plays well with Python, however in problems where the performance of the solution is critical, having the problem manager written in Python may make the evaluation of the contestant's output too slow. To mitigate this problem, we ported the `tc.py` library to Rust, thus creating the `tc.rs` library (`https://github.com/dariost/tal-utils-rs`). By using Rust as the program-

```
CREATE TABLE users (
    id TEXT PRIMARY KEY,
    name TEXT NOT NULL,
    other TEXT
);

CREATE TABLE problems (
    name TEXT PRIMARY KEY
);

CREATE TABLE submissions (
    id INTEGER PRIMARY KEY,
    user_id TEXT NOT NULL,
    problem TEXT NOT NULL,
    score INTEGER NOT NULL,
    source BLOB NOT NULL,
    address TEXT,
    FOREIGN KEY (user_id) REFERENCES users(id),
    FOREIGN KEY (problem) REFERENCES problems(name)
);
```

Fig. 6. SQLite schema for database used by `tc.py` and `tc.rs`.

ming language for the problem manager, the whole execution of the problem manager is much faster. The functionality of the two libraries is the same, and they are interoperable with each other. This means that in a single contest, the problem maker can use both Python and Rust problem managers.

Turing Arena light has no built-in support for saving the results of the contest, as this job is left to the problem manager. This is done to allow the problem maker to have full control over how the results are saved. In `tc.py` and `tc.rs` we implemented a simple database that saves the results of the contest in a SQLite (Owens, 2006) database. The schema of the database is shown in Fig. 6. The database provides a way to save the results of the contest, and it enables contestants to see their position in the ranking during the contest, using a service defined in *Turing Arena light*.

6. Graphical User Interface

The Rust implementation of *Turing Arena light* only comes with a command line interface for the client. While this is enough to run the contest, it is not very user friendly. Contestants have to remember the right parameters to pass to the client, and the less experienced ones might have trouble working with a terminal. To mitigate this problem, a graphical user interface for the client was developed.

A web application was developed as a new client for *Turing Arena light* (`https://talco-team.github.io/TALightDesktop/`). A screenshot of the application is shown in Fig. 7. It was developed using the *Angular* framework (Green and Seshadri, 2013), and it is written in TypeScript (`https://www.typescriptlang.org/`). The

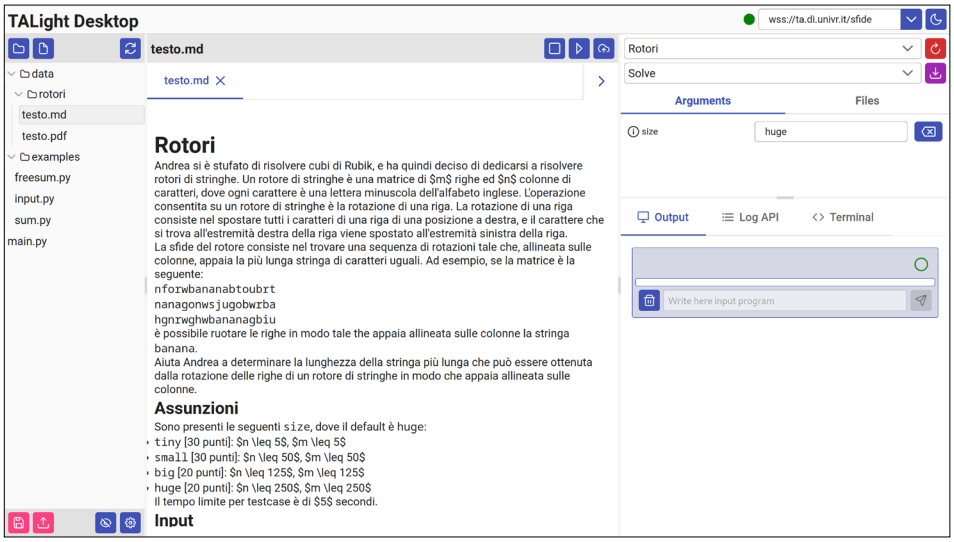


Fig. 7. Graphical user interface of T*uring Arena light.*

peculiar thing about this application is that aside from offering all the functionalities of the command line client, it also offers a way to write the solution directly in the browser. Not only that, but the solution is run directly in the browser, without the need to install any additional software. This functionality is currently only available for Python solutions, but it could be extended to other languages as well. To do this, the Python interpreter has been compiled to JavaScript, using *Pyodide* (`https://pyodide.org/`). This allows to run Python code directly in the browser. Thus, the contestant can do everything from an integrated environment in its browser.

Aside from running the solution in the browser, the web application also implements an emulated file system within the browser. This allows the contestant to send file parameters and receive file attachments and file outputs, all from the browser. Another useful feature that derives from having a file system is the ability to save and restore the working environment. This is useful for example when the contestant is working on a problem, and they want to save their progress and continue working on it later. Another scenario is when a template is provided to the contestant, and they can start working directly on it. The file system can be exported as a tar archive, or can be stored in the cloud using either GitHub (`https://github.com/`), Google Drive (`https://drive.google.com/`), or OneDrive (`https://onedrive.live.com/`). They can be later imported back from a tar archive or from the cloud, specifically from GitHub.

7. Experience in the Classroom

Turing Arena light has seen a good amount of use in some of the courses of the Computer Science department at the University of Verona. In particular, it has been used in

the courses of *Algorithms and Data Structures*, *Operations Research* and *Competitive Programming*. In this section we will discuss the experience of using *Turing Arena light* in the course of *Competitive Programming*.

The course of *Competitive Programming* is a course that is offered to the students of the department of Computer Science at the University of Verona. The course is meant to teach the students how to solve algorithmic problems, by teaching them the most common algorithmic techniques and data structures, and how to use them to solve problems. The course is structured in two parts: the first part is a series of lectures where the theory is explained, and the second part is a series of practical lessons where the students are given problems to solve, and they have to write a solution to the problem.

The practical lessons are done in a computer lab, where the students have access to a computer with the *Turing Arena light* client (`rtal`) installed. We prepared a body of problems that the students can solve, and we give them a problem to solve during each lesson. The problems are themed around the topic of the lecture, so that the students can practice the theory they learned during the lecture. The students have access to the problems both during the lesson, and at home, so that they can practice on their own. To achieve this, we have a server running `rtald` that is accessible from the Internet, and the students can connect to it from the client. The whole implementation of *Rust Turing Arena light* is released under the `MPL-2.0` license, which allows the students to download and use the client and the server for free.

Particular emphasis has been put on interactive problems. Since *Turing Arena light* allows to implement interactive problems with little effort, and they are kind of scarce in other contest management systems, we decided to focus on them. In particular, focusing on interactive problems allows us to give the students problems that do not focus on the time to compute the solution, but rather on the ability to interact with the problem manager or how many queries they can make. The kind of problems that are best suited for this are problems that involve some kind of game, where the contestant has to play against the problem manager. Another format is where the contestant has to guess some hidden information, and the problem manager gives hints to the contestant. In both cases the problem gives the contestant a sense of *playing a game*, rather than *solving a problem*, which is a good way to keep the students engaged.

Retaining the students' attention is more difficult in a classroom setting rather than in a competitive programming contest. In a contest, the contestants are motivated by the fact that they are competing against other contestants, and they want to win. Thus, they challenge themselves, they can be motivated to solve problems and learn on their own. In a classroom setting, the students are usually only motivated by the fact that they have to pass the exam, and they are not motivated to learn anything more than what is needed for that. Thus, it is important to keep the students engaged and make them interested about the topic, in order for them to then be motivated to learn more on their own. Interactive problems are a way to move towards this goal, as they can be more engaging than other kinds of problems.

As an example of such a problem, following this paragraph there is a problem that was given in the first laboratory lesson of the course.

Anna and Barbara's game Anna and Barbara discovered a new game: it is played on a V vector of n natural numbers. The first player picks a number from one end and takes it, then the second player does the same, and the game continues like this until the vector becomes empty. The player whose sum of the numbers taken is greater.

Anna always likes to play as the first player, while Barbara always wants to always go second. You want to play a game, but you have already seen the vector that will be used. Use this information to choose who to play against in so that you are sure not to lose and win the game!

Assumptions The following `size` are present, where the default is `big`:

- `small`: $n <= 8$, $max(V) <= 20$.
- `big`: $n <= 50$, $max(V) <= 10^6$

The sum of the values of V is always odd.

The time limit for testcase is 5 seconds.

Interaction The first line contains T, the number of games that will be played. In each game you are given on the first line n, and on the second line the vector V of natural separated by space. At this point it is your turn, write 0 if you want to play first, or 1 if you want to play second. The game starts with the first player and alternates players until all numbers have been taken. The player whose turn it is must write `L` or `R` followed by a carriage return depending on whether he or she wants to choose the number furthest left or the one furthest to the right.

To get `AC` you must win the game. It is always possible to win the game by some choice of which player to be and the moves to make, regardless of what the opponent will do.

Example Lines beginning with < are those sent by the server, those that begin with > are those sent by the client.

```
< 2
< 4
< 0 8 5 4
> 0
> R
< R
> R
< L
< 4
< 7 4 5 3
> 1
< R
> L
< R
> L
```

7.1. *Exams*

Aside from the laboratory lessons, Turing *Arena light ha*s also been used for the exams of the course. The exam is structured in a fashion similar to a competitive programming contest: the students are given three problems to solve, each worth 100 points, and they have four hours to solve them. The exam is taken in a computer lab, where the students have access to a computer with the Turing *Arena light clien*t (`rtal`) installed, other than the usual tools for programming, like an editor, a C++ compiler and a Python interpreter. The students are allowed to use any programming language they want, and they can use any piece of documentation they want, as long as they do not communicate with other students. However, they do not have internet access, so they cannot look up solutions online, or use other fancy tools, like GitHub Copilot (`https://copilot.github.com/`).

In the one academic year the course was offered, we administered five exam sessions. At the end of each session, the results were published, having a randomly generated identifier for each student[3]. The results of the exams are shown in Figures 8, 9, 10, 11 and 12. As shown in the figures, the total number of students that took the exam across all sessions is 31. The participation to the exam started low, with only 4 students taking the first exam, but it increased over time, with 12 students taking the last exam. Students were allowed to take the exam multiple times, and some of them did, since they could improve their score by taking the exam again, and keeping the best score. The results of the exams become better over time, as the students got more opportunities to practice and become accustomed with the kind of problems that were given.

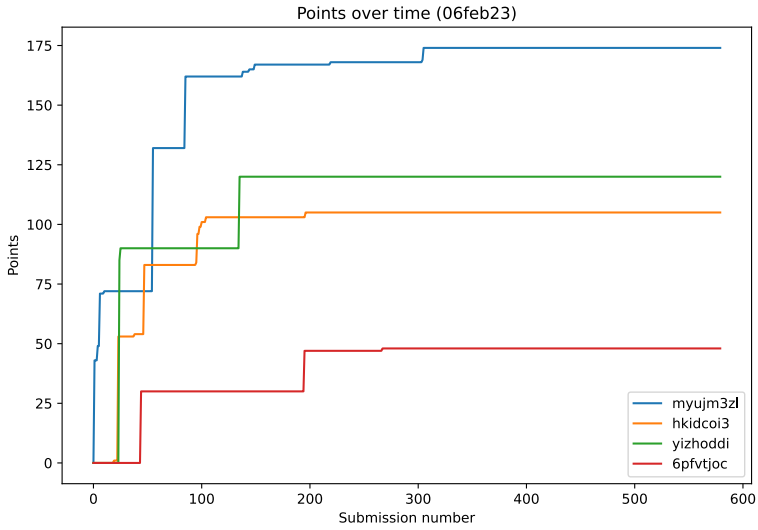


Fig. 8. Results of the 6th of February 2023 exam.

[3] If a student took the exam multiple times, they would have a different identifier each time.

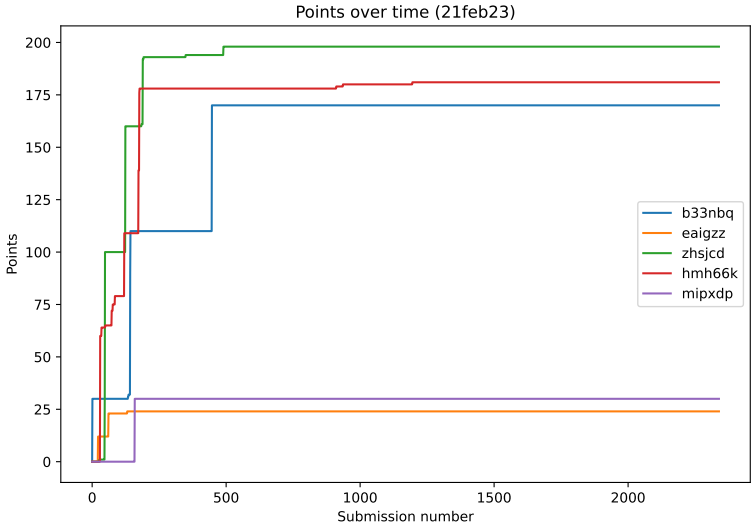


Fig. 9. Results of the 21st of February 2023 exam.

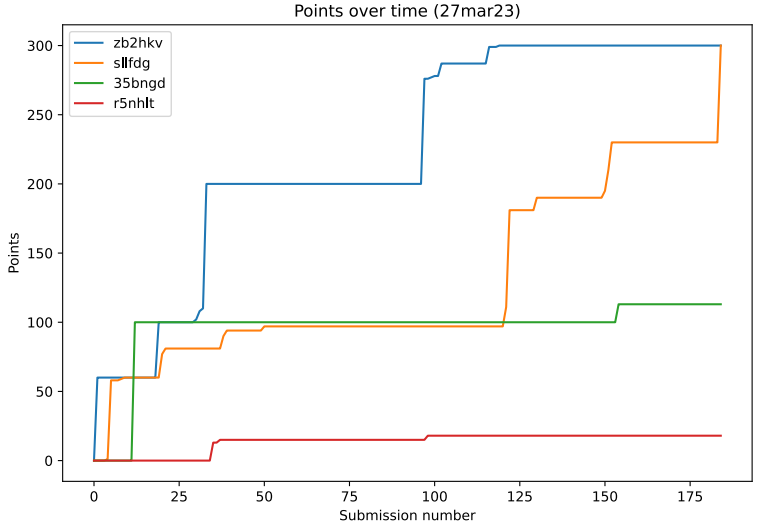


Fig. 10. Results of the 27th of March 2023 exam.

As an example of the problems given in the exam, following this paragraph there is a problem that was given in the exam session of the 22nd of June 2023.

Plumbing Luigi has begun his new adventure as a plumber, and now he is faced with his first job. In an old building there is a piping system that connects by joints the various apartments. Specifically, in this system there are n joints connected by pipes, and each pipe is connected to two joints. Between each pair of joints there is a piping path connecting them, and the number of pipes is $n - 1$. Each pipe has some length L_i.

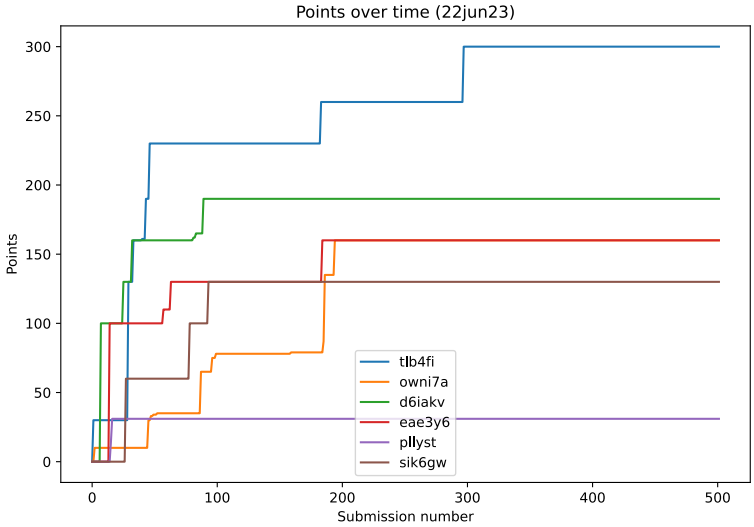


Fig. 11. Results of the 22nd of June 2023 exam.

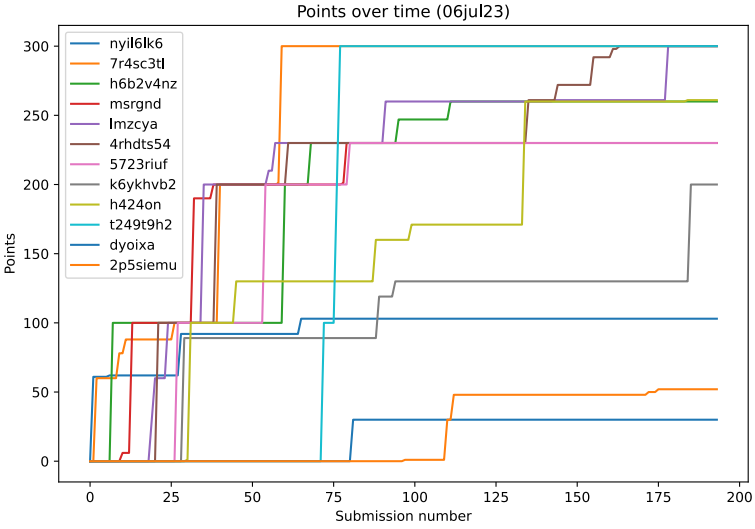


Fig. 12. Results of the 6th of July 2023 exam.

Luigi was called to calculate the total length of the piping system. However, Luigi does not have a diagram of the system, but he can measure the total length between two joints by running water between the two joints and measuring the time it takes to travel the path, thus measuring its length.

Luigi wants to finish the job as soon as possible so that he can move on to the next one, so he wants to calculate the total length of the piping system with as few measure-

ments as possible. Help him take the minimum number of measurements needed to calculate the total length of the piping system.

Assumptions The following `size` are present, where the default is `big`:

- `tiny` [30 points]: $n <= 45$, each joint is connected to at most 2 pipes
- `small` [30 points]: $n <= 45$
- `big` [40 points]: $n <= 50$

The maximum number of measurements that Luigi can make is 1000.

For each pipe i, L_i is an integer between 1 and 10000.

Interaction The first line contains T, the number of testcases to be solved. This is followed by T instances of the problem.

In each instance, initially the server sends n, the number of joints. Next, the client can make two kinds of requests:

- `? u v`: the client asks for the path length between joints u and v.
- `! l`: the client communicates the total length of the pipeline system, which is l.

Whenever the client makes a request of type `? u v`, the server responds with an integer, which is the length of the path between joints u and v.

The client can make at most 1000 requests of type `? u v`, after which it must make a request of type `! l` to terminate the interaction of the current instance.

Technical details While this problem has no time limit, sending 1000 queries and receiving as many responses can take a non-negligible amount of time.

However, it is possible to send queries in batches: if you do not need to know the result of the current query to send the next one, you can send all queries, and only after sending them do an explicit flush of the standard output.

In this way, all queries will be sent as a single packet, and all responses will be received as a single packet, greatly reducing communication time.

Example Lines beginning with < are those sent by the server, those that begin with > are those sent by the client.

```
< 1
< 3
> ? 0 1
< 4
> ? 0 2
< 2
> ? 1 2
< 6
> ! 6
```

Technical details While this problem has no time limit, sending 1000 queries and receiving as many responses can take a non-negligible amount of time.

However, it is possible to send queries in batches: if you do not need to know the result of the current query to send the next one, you can send all queries, and only after sending them do an explicit flush of the standard output.

In this way, all queries will be sent as a single packet, and all responses will be received as a single packet, greatly reducing communication time.

Example Lines beginning with < are those sent by the server, those that begin with > are those sent by the client.

```
< 1
< 3
> ? 0 1
< 4
> ? 0 2
< 2
> ? 1 2
< 6
> ! 6
```

7.2. *Survey*

After all the exams were administered, we asked the students to fill in a survey about their experience with *Turing Arena light*. The survey was anonymous, and it was done using Google Forms (`https://www.google.com/forms/about/`). The survey consisted of five questions:

- How much did you like the problems available in Turing Arena light (rtal)?
- How difficult did you find the problems proposed with Turing Arena light (rtal)?
- Did you find the interactive problems more interesting than the regular ones?
- How hard was to use Turing Arena light (rtal)?
- How strongly would you like for Turing Arena light to have a graphical user interface?

These questions were chosen to get a general idea of how the students felt about *Turing Arena light*, and to check if initial goals of *Turing Arena light* were being met. Note that this survey was conducted before the graphical user interface was available, so the last question was meant to check if the work being done on the graphical user interface was worth it.

- The responses for the first question are shown in Fig. 13. As can be seen from the results, the students liked the problems available in *Turing Arena light*. The average score is 4.07, which means that the problems were liked by the students.
- The responses for the second question are shown in Fig. 14. As can be seen from the results, the students found the problems proposed with *Turing Arena light* to be of slightlyabove-medium difficulty. The average score is 3.67, which means

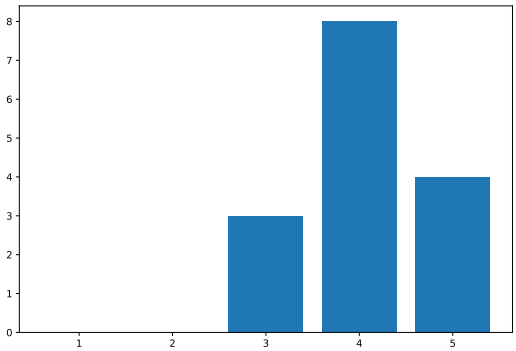


Fig. 13. Answers of question 1: *How much did you like the problems available in Turing Arena light (rtal)?*

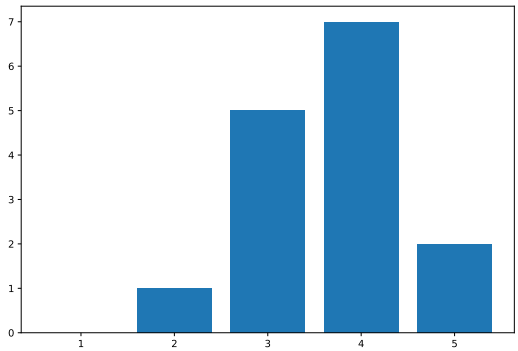


Fig. 14. Answers of question 2: *How difficult did you find the problems proposed with Turing Arena light (rtal)?*

that the problems were not too difficult, but they were not too easy either, although they were slightly on the hard side.

- The responses for the third question are shown in Fig. 15. As can be seen from the results, the students found the interactive problems to be more interesting than the regular ones. The average score is 3.80, which reinforces the idea that interactive problems are more engaging than regular ones.
- The responses for the fourth question are shown in Fig. 16. As can be seen from the results, the students found *Turing Arena light* to be easy to use. The average score is 2.20, which means that for the sample of students that took the survey, *Turing Arena light* did not pose any particular difficulty.
- The responses for the fifth and final question are shown in Fig. 17. As can be seen from the results, the students are kind of split whether they would like *Turing Arena light* to have a graphical user interface. The average score is 2.93, which means that the students are indifferent on average about wanting a graphical user interface, although there are some students that would strongly like it.

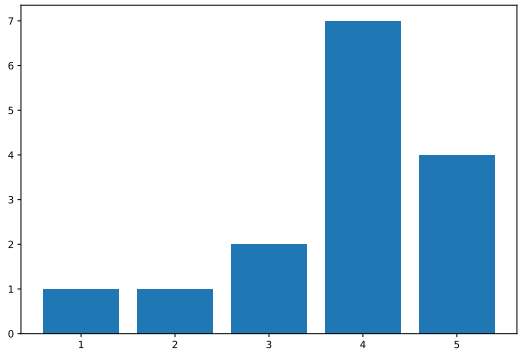


Fig. 15. Answers of question 3: *Did you find the interactive problems more interesting than the regular ones?*

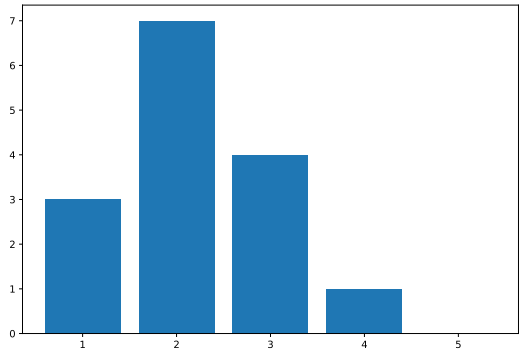


Fig. 16. Answers of question 4: *How hard was to use Turing Arena light (rtal)?*

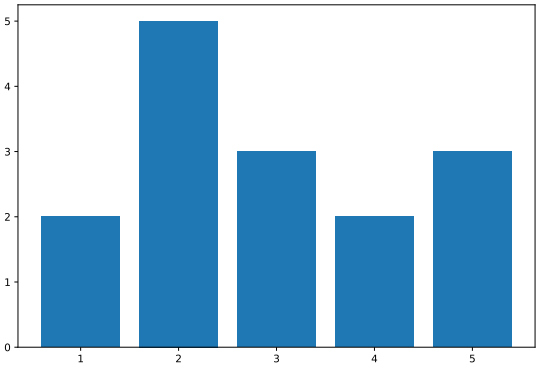


Fig. 17. Answers of question 5 of the post-exams survey: *How strongly would you like for Turing Arena light to have a graphical user interface?*

8. Future Directions

Turing Arena light has been developed enough to be used in a real-world classroom setting, and it has been used in the course of *Competitive Programming* at the University of Verona. It has been used for both the laboratory lessons and the exams, and it has been well received by the students. However, there is still a debate to be had in which direction *Turing Arena light* should move forward.

While the extreme flexibility of *Turing Arena light* made it possible to experiment a lot with different kinds of problems, it also made it difficult to find a common ground on which to standardize some common features, without having all of the problem manager libraries reimplement them. One such feature is the ability to save the results of the contest. While *Turing Arena light* does not have any built-in support for saving the results of the contest, it is possible to implement it in the problem manager. However, this means that each problem manager has to reimplement the same functionality, which is not ideal.

Moreover, some feature are implementable only by standardizing them at the core of *Turing Arena light*. One such feature is the ability of accurately measuring the time consumed by the solution. Right now, the time used by the solution is measured by measuring the time between the sending of the input and the receiving of the output. However, this is not a very accurate measurement, as it does not take into account the time spent sending and receiving the packets over the network. This is not a problem when the server and the client are on the same local network, as it happened in the course of *Competitive Programming*, but it becomes a problem when the server and the client are on different networks, such as when the server is on the Internet.

There is a solution to mitigate this problem, which is to encrypt the data, send it, then start the clock and send the decryption key. Doing it this way, one can eliminate the time spent sending the data, which can be a significant amount of time when the input is big. However, to implement such a solution, it would require to have some mechanism to make the problem manager and the core communicate on a meta-level to require this functionality from the core. However, such mechanism could cause a narrowing of the flexibility of *Turing Arena light*.

While the command-line interface has worked great for the course of *Competitive Programming*, it is not very probable that it would be fine for other courses with less *hardcore* students. Thus, the development of the graphical user interface continues, and it is planned to be tested in the next iteration of the course of *Competitive Programming*, and possibly in other courses with more *general* students.

References

Audrito, G., Di Mascio, T., Fantozzi, P., Laura, L., Martini, G., Nanni, U., Temperini, M. (2020). Recommending tasks in online judges. In: *Advances in Intelligent Systems and Computing*. Cham: Springer International Publishing, pp. 129–136.

Ben-Kiki, O., Evans, C., Ingerson, B. (2009). Yaml ain't markup language (yaml™) version 1.1. In: *Working Draft 2008-05* 11.

Crockford, D. (2006). *The Application/JSON Media Type for Javascript Object Notation (JSON)*. Tech. rep.

Eldering, J., Kinkhorst, T., van de Warken, P. (2020). *DOM Judge–Programming Contest Jury System. 2020*.

Enstrom, E., Kreitz, G., Niemela, F., Soderman, P., Kann, V. (2011). Five years with kattis – Using an automated assessment system in teaching. In: *2011 Frontiers in Education Conference (FIE)*. IEEE, Oct. 2011, T3J–1–T3J–6.

Fantozzi, P., Laura, L. (2020). Recommending tasks in Online Judges using Autoencoder neural networks. In: *Olymp. Inform.* (Dec. 2020).

Fantozzi, P., Laura, L. (2021a). A dynamic recommender system for online judges based on autoencoder neural networks. In: *Methodologies and Intelligent Systems for Technology Enhanced Learning, 10th International Conference. Workshops*. Advances in intelligent systems and computing. Cham: Springer International Publishing, pp. 197–205.

Fantozzi, P., Laura, L. (2021b). "Collaborative recommendations in online judges using autoencoder neural networks". In: *Advances in Intelligent Systems and Computing*. Advances in intelligent systems and computing. Cham: Springer International Publishing, pp. 113–123.

Fette, I., Melnikov, A. (2011). The websocket protocol. In:

Green, B., Seshadri, S. (2013). *AngularJS*. " O'Reilly Media, Inc."

Hewitt, C., Bishop, P., Steiger, R. (1973). A universal modular actor formalism for artificial intelligence. In: *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, pp. 235–245.

Hoare, C.A.R. (1978). Communicating sequential processes. In: *Communications of the ACM* 21.8, pp. 666–677.

Iffath, F., Kayes, A., Tahsin Rahman, Md., Ferdows, J., Arefin, M., Sabir Hossain, Md. (2021). Online judging platform utilizing dynamic plagiarism detection facilities. In: *Comput. Rev. Esp. Hist. Contab.* 10, p. 47.

Leal, J., Silva, F. (2003). Mooshak: a Web-based Multi-site Programming Contest System. *Software: Practice and Experience.* 33, 567–581.

Maggiolo, S., Mascellani, G. (2012). Introducing CMS: A Contest Management System. In: *Olympiads in Informatics*, 6 (2012).

Maggiolo, S., Mascellani, G., Wehrstedt, L. (2014). CMS: a Growing Grading System. In: *Olympiads in Informatics*, 123.

Matsakis, N.D., Klock, F.S. (2014). The rust language. *ACM SIGAda Ada Letters*, 34(3), 103–104.

Merkel, D. *et al.* (2014). Docker: lightweight linux containers for consistent development and deployment. *Linux j.* 239(2), 2.

Owens, M. (2006). *The Definitive Guide to SQLite*. Springer.

Pham, M.T., Nguyen, T.B. (2019). The DOMJudge based online judge system with plagiarism detection. In: *2019 IEEE-RIVF International Conference on Computing and Communication Technologies (RIVF)*. IEEE, pp. 1–6.

Revilla, M.A., Manzoor, S., Liu, R. (2008). Competitive learning in informatics: The UVa online judge experience. *Olympiads in Informatics*, 2, 131–148.

Van Rossum, G. *et al.* (2007). Python Programming Language. In: *USENIX annual technical conference*. Vol. 41.1. Santa Clara, CA, pp. 1–36.

Wasik, S., Antczak, M., Badura, J., Laskowski, A., Sternal, T. (2019). A survey on online judge systems and their applications. en. *ACM Comput. Surv.* 51(1), pp. 1–34.

Watanobe, Y., Rahman, Md.M., Matsumoto, T., Rage, U.K., Ravikumar, P. (2022). Online Judge System: Requirements, architecture, and experiences. en. *Int. J. Softw. Eng. Knowl. Eng.*, 32(06), 917–946.

G. Audrito is involved in the training of the Italian team for the IOI since 2006, and since 2013 is the team leader of the Italian team. Since 2014, he has been coordinating the scientific preparation of the OIS and of the first edition of the IIOT. He got a Ph.D. in Mathematics in the University of Turin, and currently works as a Junior Lecturer in the University of Turin.

L. Laura is currently the president of the organizing committee of the Italian Olympiads in Informatics that he joined in 2012; previously, since 2007, he was involved in the training of the Italian team for the IOI. He is Associate Professor of Theoretical Computer Science in Uninettuno university.

A. Orlandi helped train, tutor and coach potential candidates for, and members of, the Italian IOI delegation. He is currently a Principal Engineer at Google Switzerland.

D. Ostuni is currently the president of *Associazione Nazionale Programmazione Competitiva*, the largest Italian association of competitive programmers. He has been in the organization of the Italian Olympiad in Informatics since 2015. He got a Ph.D. in Computer Science from the University of Verona, and is currently a postdoctoral researcher at the University of Milan.

R. Rizzi begun training high school students for their participation toregional and national competitions of the Italian Olympiads in Informatics in 2001. Since 2004 he also begun training and coaching the Italian team to the International Olympiads in Informatics. He is full professor of Operations Research in the University of Verona.

L. Versari has been training Italian competitors in Informatics Olympiads since 2012 and Swiss competitors since 2023. He is a member of the IOI ITC, as well as part of the technical committees of multiple international competitions. He is currently a Software Engineer at Google.

Olympiads in Informatics, 2025, Vol. 19, 27–33

DOI: 10.15388/ioi.2025.02

Virtual Time Measurement in Programming Contests

Paweł DIETRICH[1], Bartosz KOSTKA[2]
[1]*Google Poland,*
[2]*Google Canada*
e-mail: p.dietrich@fri.edu.pl, kostka@oij.edu.pl

Abstract: We propose a novel method for measuring time in programming competitions. This method utilizes the instruction count, using existing hardware capabilities, to ensure a fair and deterministic evaluation. By basing the measurement on the count of executed instructions, the method becomes less dependent on the underlying machine's specifications. Furthermore, it facilitates scalability by enabling parallel program execution on multiple threads. We also discuss the adoption of this method in various Polish competitions.

Keywords: virtualized time measurement, hardware performance counters, deterministic program evaluation, programming contest judging systems.

1. Introduction

The goal of this paper is to share a novel approach to measure execution time of participants' submissions developed for Polish Olympiad in Informatics (POI). The authors also share their experience with using this approach in a production environment.

Traditionally, competitive programming problems challenge participants to write a program that performs computations and that execution is restricted by certain time and memory limits. The memory footprint of the program is straightforward to measure, as the approach to handling memory has not changed in years. On the other hand, there are multiple CPU architectures, and within a single architecture, there are different revisions of processors. Participant submissions can run faster or slower depending on the clock speed, cache sizes, CPU die layout, memory controller latency, and many other factors.

Organizers wanting to provide consistent judging verdicts using standard time measurement techniques need to maintain a uniform fleet of judging machines and carefully control their software. To achieve time measurement accuracy within an error margin of a few percent, only one program can be judged on each judging machine. This approach also disallows the use of shared hardware (e.g., cloud machines) due to "noisy neighbor" problems. The requirement to use physical machines is usually costly, and this

approach often wastes significant resources, since today's machines have multiple CPU cores. However, to maintain the time measurement accuracy, only one submission can be judged at a time.

2. Implementations

There were multiple iterations of the mechanism that we use today for time measurement in Polish competitions. The first version of the tool called oitimetool Acedański (2009) used Intel's library PinLuk *et al.* (2005). The current version used in POI is called sio2jailDubiel (2018) was based on perf API.

The first tool was limited to running only on Intel CPUs, and since there are other popular vendors in the CPU space, we decided to switch to sio2jail exclusively and abandon oitimetool. This way, we depend on an open-source software component (the Linux kernel with perf enabled), rather than on a proprietary CPU interface. Both solutions were intentionally implemented to calculate the same time for the same programs. The idea was to make the switch a small technical nuance.

2.1. *Using a Linux perf*

Modern computer systems are equipped with various ways of collecting and analyzing performance data. One of the tools that uses these build-in systems is perf – a powerful performance analysis tool available on Linux systems. perf was introduced as a tool for using the Performance Monitoring Units (PMUs) hardware in modern processors to collect data on hardware events such as instruction counting or cache monitoring.

The perf tool provides a very simple command line interface. For example:

```
$ perf stat -B dd if=/dev/zero of=/dev/null count=1000000

1000000+0 records in
1000000+0 records out
512000000 bytes (512 MB) copied, 0.956217 s, 535 MB/s

 Performance counter stats for 'dd if=/dev/zero of=/dev/null count=1000000':

          5,099 cache-misses             #      0.005 M/sec (scaled from 66.58%)
        235,384 cache-references         #      0.246 M/sec (scaled from 66.56%)
      9,281,660 branch-misses            #      3.858 %     (scaled from 33.50%)
    240,609,766 branches                 #    251.559 M/sec (scaled from 33.66%)
  1,403,561,257 instructions             #      0.679 IPC   (scaled from 50.23%)
  2,066,201,729 cycles                   #   2160.227 M/sec (scaled from 66.67%)
            217 page-faults              #      0.000 M/sec
              3 CPU-migrations           #      0.000 M/sec
             83 context-switches         #      0.000 M/sec
    956.474238 task-clock-msecs          #      0.999 CPUs

      0.957617512  seconds time elapsed
```

There are two main metrics, which can be used to model execution time of the program itself. They are `cycles` and `instructions`. Essentially different instructions can take different amount of cycles. From our experiments the ratio of cycles per particular instruction can vary from CPU to CPU, even within a particular vendor. For more accurate results for a particular hardware this can be a good metric. However our original goal was to be more hardware agnostic, that is why we choose the instructions counter, which counts the literal instructions written in a program's binary file as they are executed.

In sio2jail, we use the perf interface using its C++ API (`linux/perf_event.h`). We set up performance monitoring using `perf_event_open`[1] system call. We set it up to count only instructions from user namespace and only after the execve call (the call that executes the program to be judged) is executed. Additionally, we want to implement the instruction limit. We use the `sample_period` and `wakeup_events` options, which allow us to receive a notification when the counter values exceed certain values, to do so.

2.2. *sio2jail Usage*

The implementation of sio2jail is available in the GitHub repository SIO2Project (2018). The implementation offers process sandboxing using seccomp The Linux Kernel Documentation (2025) and instruction counting using above mentiond perf, memory limit verification and more. There is also a test program available, that was running exactly 1 second on a reference machine and was used to properly scale instructions to seconds ratio.

Unfortunately we did not provide any default options for the sio2jail binary. The users are responsible for setting all the options by themselves. We have released an extra script called `oiejq`, which sets the sio2jail command line options to print `time(1)` like output, which is handy for the participants to use.

3. Pros and Cons

Using this special virtual environment for time measurement can be beneficial for programming competitions in a number of ways.

First, the described method, leveraging hardware event counters and potentially running in a virtualized environment with namespaces, eliminates the dependency on the specific physical configuration of the machine evaluating the program. This means the same program, when run on different machines with varying factors like clock speed, cache size, or even the number of cores (assuming they don't impact the algo-

[1] `https://man7.org/linux/man-pages/man2/perf_event_open.2.html`

rithm's execution path), will result in highly consistent instruction counts. This has several advantages:

- Flexibility in judging machines. The judging machines do not need to have identical configurations, which is often impractical and costly to achieve, especially for larger competitions. This also allows for the judging machines to differ from the contestants' machines. As long as the virtual environment with hardware event counters is implemented, the measurement conditions remain consistent.
- If the environment is published, the contestants also have an ability to run their programs in the exact same environment. This is extremely helpful for situations where we cannot guarantee that each contestant has access to the exactly same machine (for instance in online competitions where contestants participate from their homes, using their personal machines). The contestants do not have to experiment how much smaller/faster are the judging machines compared to the machine they use for the competition, as they can run their programs in the exact same environment.
- Problem setters can set the time limits for their problems much easier. They can now execute their model solutions on their own machines and confidently set appropriate time limits for the problems based on the measured execution time during the development stage. This eliminates the need to worry about hardware discrepancies between their machines and the judging environment.
- A cornerstone of the proposed method is its guarantee of deterministic and repeatable execution. This ensures that the measured execution time for a given solution remains invariant across multiple runs. This characteristic proves particularly advantageous in scenarios where a solution requires re-evaluation. Under the conventional time measurement approach, such re-judgement can lead to discrepancies in the verdict, especially when the solution is fairly close to the time limit. Traditionally, this necessitates executing the solution multiple times and verifying if any of those runs fall within the allotted time. The virtual time measurement method effectively eliminates this issue. Furthermore, the inherent consistency of virtual time measurement facilitates the seamless portability of problems across different platforms, provided they share the same virtual environment. This eliminates the need for time limit re-calibration for each platform, As we use distinct platforms for competitions and public training purposes, this feature streamlines problem reuse across these platforms.
- Simplified maintenance and upgrades. If machines need to be replaced in the judging environment, the consistency of time measurement ensures that time limits for past problems don't need to be recalculated or reevaluated.

Furthermore, we want to highlight that this virtual environment allows for a high degree of isolation for program execution. This isolation ensures that the measured execution time remains independent of extraneous processes running concurrently on the judging machine. Consequently, the time measurement becomes a more accurate reflection of the program's intrinsic performance, as factors like CPU or memory utilization

by other processes are effectively eliminated from the equation. This isolation offers a significant advantage for both contestant and judging machines. For judging machines, this isolation permits the parallel execution and evaluation of the programs. By leveraging multi-core processors, the judging process can achieve significant efficiency gains. Additionally, the isolation allows us to utilize virtualization techniques without concerns regarding the impact of other processes running within the same virtual machine. This allows for greater flexibility in resource allocation and management within the judging environment.

While the proposed method offers substantial advantages, especially for the contest organizers, we have to acknowledge that it represents an abstraction of real-world time measurement. While the discrepancies between the standard method and the virtual processor approach are minimal, we recognize that transitioning to a new time measurement method necessitates a comprehensive preparation and education effort. Our experience, encompassing over a decade of utilizing virtual time measurement, instills confidence that this method will have a negligible impact on the competition experience for the vast majority of participants, particularly for new contestants. This is contingent upon providing them with appropriate tools to gauge execution time within the virtual environment.

The main differences to traditional approach, which can be viewed as cons are as follows.

- Uniform memory access cost. Cache size does not affect the program performance as measured by the number of instructions. It means that programs do not get any penalty due to cache misses and the whole memory basically has a uniform access cost, which is far from the reality.
- A separate realtime limit is still needed. This is because a program, can still hang on a syscall or even just sleep. In such cases no instructions are being executed and a instruction limit might not be reached at all. However in competitive programming usage of sleep and complex syscall instructions is very rare.

4. Adoption in Polish Olympiad in Informatics

Recognizing the significance and potential ramifications of adopting a new time measurement system, we prioritized a well-defined implementation process. This ensured thorough preparation and understanding for both the scientific committee and the contestants.

During the initial phase (2008), all first-stage submissions for the Polish Olympiad in Informatics (POI) were re-evaluated using the oitimetool after the competition concluded. This analysis compared results generated by oitimetool with those obtained through conventional time measurements. The results exhibited a high degree of statistical similarity. Notably, correlation coefficients remained consistently above 0.998. Detailed findings are available in Acedański (2009).

The oitimetool system's first official integration into the competition occurred during the final stage of the POI in 2011. This stage was chosen strategically due to the smaller number of advanced participants (under 100). During this contest, contestant submissions were judged in two distinct environments:

- The traditional environment measuring actual execution time.
- A virtual environment utilizing the Pin library for instruction counting.

The final score for each submission was determined by the higher value obtained from these two executions.

Following the successful pilot run, oitimetool was adopted as the sole judging environment for the first time during the subsequent school year (2011/2012). All three stages of the POI utilized oitimetool, resulting in [number] submissions being judged.

Over the following years, advancements in technology necessitated adaptation. This included the introduction of new hardware-based instruction counting and mechanisms for program privilege isolation and restriction. To address these changes, a modernized judging environment named sio2jail was implemented in 2018. Built upon state-of-the-art technologies of the era, sio2jail addressed some of the limitations present in oitimetool.

The virtual environment was not incorporated into other Polish programming competitions. Notably, both the Polish Collegiate Programming Competition (Akademickie Mistrzostwa Polski w Programowaniu Zespołowym) and the country's largest open competition, Algorithmic Engagements (Potyczki Algorytmiczne), opted out. This decision was based on the belief that both contestants (students and professionals) and problem-setters in these competitions possess sufficient experience with traditional real-time measurement approaches, and therefore, the virtual environment wouldn't offer them significant benefits.

We believe that this implementation allowed us to use full potential of multithreaded judging machines, minimizing impact of the hardware to the scoring and running costs. On the downsides, it slightly changed the characteristics of a CPU and treated memory accesses as uniform.

References

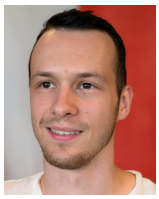
Acedański, S. (2009). *Wykorzystanie sprzętowych liczników zdarzeń do oceny wydajności algorytmów* (Unpublished master's thesis). University of Warsaw.

Dubiel, M. W. T. D. P. J. K. W. (2018). *Sio2jail, narzędzie do nadzorowania wykonania programów zgłaszanych w ramach konkursów algorytmicznych.* (University of Warsaw, 2018)

Luk, C.-K., Cohn, R., Muth, R., Patil, H., Klauser, A., Lowney, G., ... Hazel-wood, K. (2005). Pin: building customized program analysis tools with dynamic instrumentation. *Acm Sigplan Notices*, 40(6), 190–200.

SIO2Project. (2018). *A tool for supervising execution of programs submitted in algorithmic competitions.* Retrieved from `https://github.com/sio2project/sio2jail`

The Linux Kernel Documentation. (2025). *seccomp: secure computing.* Retrieved from `https://www.kernel.org/doc/html/latest/userspace-api/seccomp_filter.html`



P. Dietrich is an engineer at Google Poland working on platforms and infrastructure. He is serving as a member of Polish Olympiad in Informatics Committee focusing on technical details. Paweł is leading a technical team that organizes most competitive programming events in Poland. This team was responsible for running contests with time measurement approach described in this document.

B. Kostka is a software developer at Google Canada. He is actively involved in the competitive programming community, serving as a member of the IOI Scientific Committee, Vice President of the Polish Junior Olympiad in Informatics, and a member of the Universal Cup organising committee. Previously, he worked with many talented students who won multiple IOI medals and also organised and set problems for Google Kick Start.

Olympiads in Informatics, 2025, Vol. 19, 35–44

DOI: 10.15388/ioi.2025.03

Strategy and Tactics for Introducing Generative Artificial Intelligence into the Instrumental Distance Learning System DL.GSU.BY

Michael DOLINSKY

Faculty of Mathematics and Technologies of Programming, F. Skorina
Gomel State University, Sovetskaya str., 104, Gomel. 246019. Republic of Belarus
e-mail: dolinsky@gsu.by

Abstract. This paper provides the strategy and tactics for introducing generative artificial intelligence (GenAI) into the instrumental **distance learning** system DL.GSU.BY (hereinafter **DL**). The strategy consists of sequential implementation of the following stages of development: creating opportunities for convenient work with GenAI in the DL system; launching electronic GenAI students to automatically complete training courses in the DL system and comparative analysis of the achievements of various GenAIs among themselves and with real students; accumulation and dissemination of students' experience working with GenAI; improving the quality of training using GenAI by developing a system of preprompts for tasks and subjects; further personalization of training through the implementation of advanced techniques for using GenAI (active GenAI, Learning By Teaching). GenAI implementation tactics consistently and in detail describe the practical steps to implement the strategy.

Keywords: generative artificial intelligence, instrumental system for distance learning, DL.GSU.BY.

1. Introduction

The goal of initial programming training at the university is to develop Computational Thinking skills (Kaleem, 2024), which includes the following components: abstraction, decomposition, pattern recognition, algorithmization, debugging.

A significant increase in learning efficiency comes from the use of online platforms for teaching programming, such as EduCoder (Zhang, 2023; Li, 2023), HTProgramming (Figueiredo, 2021), Code4brownies (Phan, 2018).

A new stage in increasing the efficiency of initial programming training is associated with the introduction of generative artificial intelligence (GenAI). Text-based generative systems have a chatbot as the front end to interact with the user and are based on LLMs – generative models that can produce new content based on the data they are trained on.

The most famous LLMs: GPT (OpenAI), Gemini (Google), Llama (Meta), Claude3 (Anthropic). User interaction with a chat bot is implemented by requests. Herden (2024) formulated tasks for such chat bots during the initial programming learning process: explaining basic knowledge, constructing code, explaining code, refactoring code, formatting code, checking coding style, commenting on code.

The most effective seems to be the integration of GenAI into online systems for training and testing programs. TutorBot+ (Martinez-Araneda, 2023, 2024) is integrated into a WEB training system and an online program testing platform: it allows you to take the problems, obtain solutions in dialogue with GenAI, and check solutions. KOGI (Kuramitsu, 2023) is a learning support system that integrates ChatGPT and the Jupyter framework. KOGI helps the student get advice from ChatGPT in response to errors and questions. KOGI is implemented in two introductory courses: Algorithms, Data Science. As a result, there is a significant reduction in the number of unresolved student errors and teachers receive information about questions and errors.

IPSSC is an intelligent programming scaffolding system based on ChatGPT (Liao, 2024), where instead of direct interaction, students work through three structured modules: Solution Assessment, Code Assessment, and Free Interaction. In the Solution Assessment module, students decompose a complex problem into several simple ones and abstract them so that the solution is suitable for similar problems (thus, this module focuses on decomposition and abstraction). In the Code Assessment module, students design algorithms, write code, and continuously debug code, using the module to identify errors in the code, then fix them. ChatGPT helps improve algorithms and code. In the Free Interaction module, students can interact with ChatGPT directly, discussing topics that are not covered in other modules.

Coding Step (Sarshartehrani, 2024) is a web application designed to teach the basics of Python programming.

LearningProgrammingWithGPT (Abolnejadian, 2024) – an environment that is used with students in the CS1 course while studying Python. The course integrates ChatGPT as a means to support students and instructors in the classroom. The platform acts as a medium between students and their GenAI instructor, providing personalized educational material based on preprompts built into student requests.

The developers of CodeHelp (Liffiton, 2023; Denny,2024) claim that programming teachers have huge problems being able to devote enough time to each student. CodeHelp provides students with on-demand help without offering immediate solutions, an example of a "Socratic" tutor who avoids revealing solutions directly to the user.

CodeAid (Kazemitabaar, 2024) also avoids direct answers with code, trying to guide the student to solve the problem.

The developers of NotebookGPT (George, 2024) believe that direct use of ChatGPT can interfere with student learning, so NotebookGPT gives access to GPT, but does not return complete solutions, providing: feedback on programming style, explanations of how pieces of code work, help with debugging code, the ability to see alternative solutions to problems.

AI TA (Lee, 2023) helps students perform decomposition – that is, partitioning tasks into subtasks. AI TA supports the Subgoal learning, an effective learning strategy that helps students break down complex problems into subtasks.

StAP-tutor (Roest, 2024) provides recommendation of the next step to solve the problem. TeachYou+AlgoBo (Jin, 2023) offers an LBT (Learning By Teaching – teaching yourself by teaching someone). Here TeachYou is an LBT environment for teaching algorithms, and AlgoBo is a learning chatbot for problem solving.

2. Instrumental System of Distance Learning DL.GSU.BY

Since September 1999 at Gomel State University named after F. Skorina, under the leadership of the author, the instrumental distance learning system DL.GSU.BY (hereinafter DL) has been introduced and developed (Dolinsky, 2022a). The author has been using it all this time to teach programming to schoolchildren (starting from the first grade) (Dolinsky, 2016) and first-year students of the Faculty of Mathematics and Programming Technologies (Dolinsky, 2022b)., as well as to teach first- and second-year students the basics of digital electronics within the subjects Machine-oriented programming, and Computer architecture (Dolinsky, 2022c).

DL allows you to take problems and send solutions to these problems for testing in various programming languages, including Pascal, C++, Python, Java, C#. And when studying the basics of digital electronics, programs are sent in assembly languages and C microprogramming, as well as circuit diagrams of digital devices developed in the high-level design system HLCCAD (Dolinsky, 2022d). To support personalized learning for students with different levels of training, a tree-based learning system has been introduced, in which a student who cannot solve a problem has the opportunity to press a "I don't know" button and receive a tree of auxiliary tasks that will help him ultimately solve the original problem. However, on the one hand, this is a static, predetermined system. On the other hand, there are many test tasks for which there is no such built-in training system. The introduction of generative artificial intelligence can help in learning to solve these problems as well. In addition, for courses related to teaching the basics of digital electronics, there is no such learning tree at all. Further development the system of learning capabilities of the DL is associated primarily with its integration with GenAI.

3. Strategy for Introducing GenAI into the Instrumental Distance Learning System DL.GSU.BY

3.1. *Creating Opportunities for Convenient Work with GenAI in the DL System*

The basis for introducing GenAI into the DL system is a chatbot (#1), which receives student requests and sends them through the API provided by GenAI and receives responses, and then displays the responses to students. The presence of such a chatbot allows the use of special additional training preprompts for tasks, programming languages, topics and subjects, including those taking into account the individual characteristics

of a particular student. To explore the capabilities of GenAI from various companies, they are permanently added to our chatbot, and the student is given the opportunity to select a GenAI and then assess satisfaction with the result of working with GenAI with a score from 0 to 5. Accordingly, at the student's request, he is provided with a list of available GenAI in alphabetical order or in descending order of the average GenAI grade by students. GenAIs, which are worked interacting with our chatbot #1, are called built into the DL system.

Alternatively, students are given the opportunity to work with the same GenAI directly, without using our chatbot, to be able to develop the latter based on student suggestions. Such GenAI are called non-embedded.

3.2. *Launching e-Learners in DL Courses*

A special bot is being developed (#2), the parameter of which is name of the GenAI website. After manual registration on DL, this bot permanently performs the following work:

- Goes to the first available unsolved task.
- Takes the problem text.
- Sends it to GenAI with a request to return the solution to the problem.
- Receives a solution from GenAI.
- Sends it for testing on DL.
- If the solution passes, moves on to the next task.
- If the solution does not pass – takes from the DL website, a test on which the solution did not pass.
- Sends a test to GenAI with a request to correct the solution.
- Receives a corrected solution in response.
- Sends the corrected solution to the site DL.

This process continues until the solution passes or until the limit on the number of submissions of a solution to the same problem.

There are educational and control tasks. Educational tasks are open all the time, except for those classes when control tasks are open. The results of all e-students appear in the special result tables: only e-students – to compare GenAI sites with each other based on the success of solving problems; together with students – to compare the results of e-learners with the results of students.

For each problem, the following marks are saved: not solved, solved at what attempt, by which GenAI.

3.3. *Accumulation of Student Work Experience*

Insert call icons to our chatbot, which supports working with built-in and non-built-in GenAI, into the DL in the menu of the site, each course (subject), each task. As students

work with the GenAI chatbot, a protocol is kept that records the date and time of work, the student, the task, and the student's assessment of the quality of the assistance received. In parallel, for the built-in GenAI, the entire dialogue is saved in the file system. For non-embedded GenAI, the student is asked to save the dialogue in the cloud and enter a link to it in a special field. DL will automatically copy the dialog file to its file system. Links to each such dialogue are provided directly on the page of the task for which this dialogue was conducted. In addition, pages are created with links to all available dialogues, both on tasks and on general questions. Search by keywords is provided. In this way, students gain experience working with GenAI. And a student can receive help without directly contacting GenAI, if such help has already been provided to another student before and turned out to be effective.

3.4. *Improving the Quality of Teaching by Developing Preprompts*

A special direction of development is the permanent development and improvement of the system preprompts by tasks, courses, programming languages. Such preprompts are additional information that is sent before the student's request to improve the quality of the response to the student. Such preprompts are developed primarily by the students themselves. In addition, students are provided with direct access to all preprompt information so that they can use it interactively when working with non-embedded GenAI.

3.5. *Personalization of Learning*

Currently, the author sees two directions for the development of personalization:

(1) The "active GenAI" mode, when, having received the condition of the problem, it does not immediately give a solution, but tries to consistently lead the student to a solution, asking clarifying questions that help GenAI find out what exactly the student does not know and, accordingly, eliminate this ignorance. At the same time, the student, under the guidance of GenAI, consistently performs abstraction, decomposition, pattern recognition, algorithm development, coding and debugging for the task.
(2) Mode LBT (Learning By Teaching) – when a student is asked to teach a specially tuned GenAI to solve problems of a given type. The same mode can be used to research and develop pre-prompts on tasks, topics and programming languages.

3.6. *Entry from the other End*

Since 1999, the DL system has been accumulating problems, their solutions by students, and the corresponding entries in the protocol: pass or fail, and if not pass, then on what test. These solutions are of high-level programming languages, Intel 8086 assembler,

C-MPA microprogramming language, digital device diagrams – represented by projects – in graphical or text form. This data can be used for direct LLM training.

4. Tactics for Introducing GenAI into the Instrumental Distance Learning System DL.GSU.BY

1. Create opportunities for convenient work with GenAI in the DL system:

- Launch the page `http://dl.gsu.by/ai/chat` – this task has already been completed.
- Create and expand the list of sites accessed via API (currently, there are four such sites).
- Create and expand the list of sites where students can work interactively after logging in through `http://dl.gsu.by/ai/chat`.
- Develop a page for adding new sites to the second list (by entering a URL).
- Allow students to rate their experience with sites from both lists (using a 0–5 star rating system).
- Display site lists both alphabetically and sorted by descending ratings.

2. Accumulate of student work experience:
GenAI call icons are built into the menu of the DL website, DL course, and tasks in the DL course. Students are provided with the opportunity to:

- Select GenAI for dialogue from the list provided (ordered alphabetically, the current rating of students, by the number of tasks passed using this GenAI).
- Replenish the AI list with new URLs with starting comments (the system administrator has the ability to edit and delete information entered by students).
- Ask questions about tasks.
- Ask any questions (for clarification).
- Leave your comments/evaluations of the work – search and view dialogues on tasks and answers to questions.

The system automatically accumulates links to task dialogs, providing ordering and search by task number. CC also accumulates lists of dialogues on general issues, sorted alphabetically and searched by keywords. Dialogs are stored as doc files in a special folder Dialogs.

GenAI is called by following the link `http://dl.gsu.by/ai/chat` with the transmission of any combination of the following parameters, depending on the already available information:

AI_ID = GenAI identifier
Course_ID = course (subject) identifier
Task_ID = task identifier
Language_ID = programming language identifier (the default is taken from the task page, the student is also given the opportunity to select another programming language from the proposed list).

To support the operation of the chatbot, information about courses, programming languages and users is copied from the database (DB) of the DL system:

Courses: <Number> <Name>
Languages: <Number> <Name>
Users : <Number> <personal information as in DL>

In addition, special database tables are created

AI_ID <Number> <URL><Indicator built-in/non-embedded)><Name>
Themes <Number> <Subject of the request to ID>
(for subsequent search and/or display
in the list of topics by alphabetical order/relevance>
References <Number> <file name in the Dialogues folder>
AI_LOG <Date><Time><AI_ID> <Refs_ID (dialogue link)> <(0–5)
User Rating> [User_ID]
[Course_ID] [Task_ID] [Theme_ID]

During work in the database, a log of work with sites is kept in the format

<Date><Time><AI_ID> <Refs_ID (link to dialogue)> <(0–5)
User Rating> [User_ID] [Course_ID]
[Task_ID] [Theme_ID]

[] mean that this information may absent

All dialogues are saved in the file system in the Dialogues folder, in subfolders

<Task_ID> <Language_ID>

The dialog file names are as follows:

<UserID> <AI_ID> <Date><Time><User Rating>.doc

When working with built-in GenAI, these files are created automatically by the system.

When working with non-embedded GenAI, the student is asked to manually create a dialogue doc file in the cloud, and then enter a link to this file, as well as select dialogue metadata (Task_ID, Language_ID) or enter the topic of the dialogue.

If a student requests a task with previously saved dialogues, a list of them is first shown for viewing. (together with user rating in descending order of ratings). In addition, it is a supporting list of saved dialogs by task / topic / language.

3. Improve the quality of teaching by developing preprompts

A folder Preprompts is created in the file system with files Course_ID.txt, Task_ID_.txt, Language_ID.txt.

When working with the built-in GenAI, suitable preprompts are automatically sent to GenAI, anticipating student requests. When working with unbuilt GenAI, students have access to all preprompts to manually send preprompts.

Resources are available for editing preprompt files, developing and accumulating preprompt files for tasks, languages, courses by students.

In order to provide the chatbot with access to task texts, folders containing them are copied from the DL to the chatbot:

Tasks – html task texts
Images\original – pdf and doc – task texts

Make and add to the folder

Tasks-ext – txt-texts of tasks.
Replenish task txt texts by extracting from pdf and doc, if not found in Tasks and Tasks-ext.

Automatically include the task texts into the student's request for the task.

Preprompts are also automatically inserted depending on the task (language, course).

4. Launch e-learners in DL courses
The goal is to support the continuous operation of GenAI on DL: solving training problems, tests, team olympiads, possible with a limit on the number of attempts, followed by comparison of the GenAI results with each other and with the results of students. To ensure the principle of "do no harm" to the operation of the DL system, tests for tasks (Archives folder) and the DL database are copied to the chatbot. At first there will be a routine copying of information (for example, once a day at night), and in the future it will be incremental (that is, as the information changes).

Next development of DL API is needed:

- Log in with your account.
- Go to the course.
- Get a list of tasks to be solved (during training, control, by options).
- Get the problem text.
- Send the solution to the problem.
- Find out the verdict on the sent solution.
- Take the first failed test for the solution.
- Skip task.

E-learners are launched by universal bot #2, which receives AI_ID as a parameter.

5. Handmade
This work is done until it is automated in order to accumulate relevant experience:

- Create – expand the list of AI sites where you can ask for solutions to problems.
- Register AI sites on DL (such as AI URL instead of last name and first name).
- Comparison of the effectiveness of different GenAI (by manually sending solutions to the same tasks).
- Accumulation of dialogues and links to them – for solving problems.
- Accumulation of preliminary notes on tasks, languages, courses.
- Accumulation of tasks and GenAI most suitable for LBT (Learning By Teaching).
- Solving problems using these sites (every problem on each site).
- Write on the forum about students impressions.
- Posting a link to the task on the DL, the URL of the AI site, a link to the entire dialogue with the AI site.
- Accumulate a helper file for preprompts before sending the task conditions.

- Systematic work on pre-prompts for the most problematic tasks for students/e-students
- Training AI sites to "consistently teach students how to solve problems" – by individual types.

5. Conclusion

This paper presents the strategy and tactics for introducing GenAI into the instrumental distance learning system DL.GSU.BY. As a result, it is expected to create opportunities for convenient work with GenAI in the DL system; launching electronic GenAI students to automatically complete training courses in the DL system and comparative analysis of the achievements of various GenAIs among themselves and with real students; accumulation and dissemination of students' experience working with GenAI; improving the quality of training using GenAI by developing a system of preprompts for tasks and subjects; further personalization of training through the implementation of advanced techniques for using GenAI (active GenAI, Learning By Teaching).

References

Abolnejadian, M , Alipour, S., Taeb K. (2024). Leveraging ChatGPT for Adaptive Learning through Personalized Prompt-based Instruction: A CS1 Education Case Study. In: *Extended Abstracts of the 2024 CHI Conference on Human Factors in Computing Systems (CHI EA '24)*. ACM, New York, NY, USA, Article 521, 1–8.

Denny, P., MacNeil, S., Savelka J., Porter, L., Luxton-Reilly, A. (2024). Desirable Characteristics for AI Teaching Assistants in Programming Education. `https://arxiv.org/pdf/2405.14178v1`

Dolinsky, M. (2016) Gomel Training School for Olympiads in Informatics. *Olympiads in Informatics*, 10, 237–247.

Dolinsky, M. (2022a) Instrumental system of distance learning DL.GSU.BY and examples of its application. *Global Journal of Computer Science and Technology Interdisciplinary*, 22(1), 45–53.

Dolinsky, M. (2022b). Teaching Algorithms and Programming First Year University Students on Base of Distance Learning System DL.GSU.BY. *WSEAS Transactions on Advances in Engineering Education*, 19, 52–57.

Dolinsky, M. (2022c). Experience of Blended Learning the Fundamentals of Digital Electronics for First/Second Year University students On Base of Distance Learning System DL.GSU.BY. *International Journal of Education and Learning Systems*, 7, 59–64.

Dolinsky, M. (2022d). Tool HLCCAD for Blended Learning the Fundamentals of Digital Electronics. *International Journal of Circuits and Electronics*, 7, 47–55.

Figueiredo ,J., Garcia-Penalvo, F.J. (2021). Teaching and Learning Tools for Introductory Programming in University Courses. In: A. Balderas, A. J. Mendes and J. M. Dodero, Eds., *Proceedings of the 2021 International Symposium on Computers in Education (SIIE) (23–24 September 2021, Malaga, Spain)*. USA: IEEE.

George, S.D., Dewan P. (2024). NotebookGPT – Facilitating and Monitoring Explicit Lightweight Student GPT Help Requests During Programming Exercises. In: *Companion Proceedings of the 29th International Conference on Intelligent User Interfaces (IUI '24 Companion)*. ACM, NY, USA, 62–65.

Herden, O. (2024). Integration of Chatbots for Generating Code Into Introductory Programming Courses International Conference on Future of education.

Jin, H., Lee, S., Shin, H., Kim, J. (2023). Teach AI How to Code: Using Large Language Models as Teachable Agents for Programming Education. `https://arxiv.org/pdf/2309.14534`

Jin, H. , Lee, S., Shin, H., Kim, J. (2023). Teach AI How to Code: Using Large Language Models as Teachable Agents for Programming Education. `https://arxiv.org/pdf/2309.14534v2`

Kaleem, M., Hassan, M.A., Khurshid, S.K. (2024). A Machine Learning-Based Adaptive Feedback System to Enhance Programming Skill Using Computational Thinking. IEEE Access https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10506466

Kazemitabaar, M., Ye, R., Wang, X., Henley, A.Z., Denny, P., Craig, M., Grossman T. (2024). CodeAid: Evaluating a Classroom Deployment of an LLM-based Programming Assistant that Balances Student and Educator Needs. https://arxiv.org/pdf/2401.11314

Kuramitsu, K. , Obara, Y., Sato, M.., Obara, M. (2023). KOGI: A Seamless Integration of ChatGPT into Jupyter Environments for Programming Education. In: *Proceedings of the 2023 ACM SIGPLAN International Symposium on SPLASH-E (SPLASH-E 2023)*. ACM, NY, USA, 50–59.

Lee, C. , Myung, J., Han, J., Jin, J., Oh, A. Learning from Teaching Assistants to Program with Subgoals: Exploring the Potential for AI Teaching Assistants. https://arxiv.org/pdf/2309.10419

Li, Z., Zhang, S., Sang, X. (2023). Exploration of Machine Learning Teaching based on the EduCoder Platform. *Journal of Education and Educational Research*, 4(3), 130.

Liao, J., Zhong, L., Zhe, L., Xu, H. , Liu, M., Xie T. (2024). Scaffolding Computational Thinking With ChatGPT. *IEEE Transactions on Learning Technologies*, 17.

Liffiton, M., Sheese B., Savelka, J., Denny P. (2023). CodeHelp: Using Large Language Models with Guardrails for Scalable Support in Programming Classes. https://arxiv.org/pdf/2308.06921

Martinez-Araneda C., Gutierrez, M., Maldonado, D., Gomez, P., Segura, A., Vidal-Castro, C. (2024). Designing a Chabot to support problem-solving in a programming course. In: *INTED2024 Proceedings*, pp. 966–975.

Martinez-Araneda C., Valenzuela, M.G., Meneses, P.G., Montiel, D.M., Navarrete, A.S. Vidal-Castro C. (2023). How Useful TutorBot+ is for Teaching and Learning in Programming Courses: a Preliminary Study. In: *42nd IEEE International Conference of the Chilean Computer Science Society (SCCC)*. Concepcion, Chile, pp. 1–7.

Phan, V., Hicks, E. (2018). Code4brownies: an active learning solution for teaching programming and problem solving in the classroom. In: *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, pp. 153–158.

Roest, L., Keuning, H., Jeuring, J. (2024). Next-Step Hint Generation for Introductory Programming Using Large Language Models. In: *Proceedings of the 26th Australasian Computing Education Conference (ACE '24)*. Association for Computing Machinery, New York, NY, USA, 144–153.

Sarshartehrani, F., Mohammadrezaei, E., Behravan, M., Gracanin, D. (2024). Enhancing E-Learning Experience Through Embodied AI Tutors in Immersive Virtual Environments: A Multifaceted Approach for Personalized Educational Adaptation. In: Sottilare, R.A., Schwarz, J. (eds), *Adaptive Instructional Systems. HCII 2024 LNCS, 14727*. Springer, Cham.

Zhang, S., Yang, J., Sang X. (2023). Exploring the Applications of EduCoder Platform in Blended Teaching for Computer Major. *Journal of Education and Educational Research*, 4(2), 100.

M. Dolinsky is a lecturer in Gomel State University "Fr. Skoryna" from 1993. Since 1999 he is a leading developer of the educational site of the University (dl.gsu.by). Since 1997 he is heading preparation of the scholars in Gomel to participate in programming contests and Olympiad in informatics. He was a deputy leader of the team of Belarus for IOI'2006, IOI'2007, IOI'2008 and IOI'2009. His PhD is devoted to the tools for digital system design. His current research is in teaching Computer Science and Mathematics from early age.

Olympiads in Informatics, 2025, Vol. 19, 45–62

DOI: 10.15388/ioi.2025.04

International Science Olympiads: The Israeli Teams

Judith GAL-EZER, Doron ZOHAR, Anat ROLNIK
The Open University of Israel, Mathematics and Computer Science Dpt. Ra'anana, Israel
e-mail: galezer@openu.ac.il, doron.zohar@openu.ac.il, anatrolnik@gmail.com

Abstract. The International Olympiads (IOs) caters to exceptional students worldwide. Israel, for example, in 2023, competed in nine out of 14 IOs, including five in the sciences. Students participating in the Olympiads must possess a high level of knowledge and a broad understanding of their subject area, along with the ability to tackle complex problems, synthesize knowledge, and develop creative solutions.

Ministries of Education as well as academic institutions and policy makers see the Olympiads as a vehicle of developing the social conditions and economics of their country, consequently, they encourage talented students by providing financial support for training programs, travel, and accommodation.

In this paper, we provide an overview of the science IOs: the International Mathematical Olympiad (IMO), the International Physics Olympiad (IPhO), the International Chemistry Olympiad (IChO), and the International Biology Olympiad (IBO), with a particular focus on the International Olympiad in Informatics (IOI). We describe the organizational structure, student training process, and ranking system, while highlighting the participation and achievements of Israeli delegations.

Keywords: International Olympiads; International Science Olympiads; Israeli National Teams.

1. The International Olympiads

The International Olympiads (IOs) are social-educational events that bring together outstanding students from around the world who share a common interest in science, and a team of expert coaches with aligned goals and interests. The official aims of each Olympiad are specified in the competition regulations (for example (IBO, 2025; IChO, 2023; IMO, 2023; IOI, 2023; IPhO, 2023)). In addition, unofficial goals arising from the nature of the event itself include:

- Encouraging scientific excellence among teens.
- Increasing interest in the sciences.
- Encouraging students to choose scientific subjects in schools and in the future.
- Enabling international relations between educators, scientists, and future scientists.

Lim *et al.* (Lim *et al.*, 2014) highlight the impact of Science Olympiad programs on students' education in science, technology, engineering, and mathematics (STEM), enhancing their skills and shaping their career aspirations. These programs provide students with a perspective on science beyond their high school experience, offering close contact with university faculty and researchers, and insight into potential science careers: "Science Olympiads are important vehicles for science communication, especially between scientists, educators, and students. When students participate in these Olympiads, they gain exposure to a different view of science from that which they experience at the high school level. Close contact with university faculty and science researchers will provide students with a glimpse of what careers in science can offer. For those who will not pursue science any further than high school, at least they will hopefully have a higher level of scientific literacy". In a later research, Jovanov and Stankov (Jovanov and Stankov, 2020) mention an additional aspect of the Olympiads: "Science Olympiads are not just a science competition but a means to care for talent in the particular scientific field. International Olympiad in Informatics (IOI) is one of the first five Olympiads that arose, after Mathematics, Physics and Chemistry, and before Biology Olympiad. Being the "summit" of the brightest students, at all Olympiads contestants are generously awarded with recognitions in the form of gold, silver and bronze medals, and additionally, the so-called 'Honorable Mention' award".

STEM competitions significantly contribute to student development, fostering creativity and critical thinking among emerging professionals as presented in (Campbell *et al.*, 2017). Campbell and Welberg observed that many students who participated in these programs later chose to pursue science-related academic programs and technical careers that contribute to national development (Campbell and Welberg, 2010). Years later, Smith *et al.* also reached the conclusion that participation in such rigorous competitions, along with the preparatory training involved, bolsters students' motivation and promotes sustained academic growth. Science Olympiad experiences influenced both academic and career decisions, especially in college and major selection (Smith *et al.*, 2021).

There are many international competitions designed for talented youth in the fields of science; any competition may be called an "Olympiad". Some examples are: The Asian Physics Olympiad (APhO) (APhO, 2024), the Romanian Master in Mathematics (RMM) (RMM, 2024), the International Mendeleev Chemistry Olympiad (IMChO) (IMChO, 2024), computer science competitions, such as the Asia-Pacific Informatics Olympiad (APIO) (APIO, 2024), Central European Olympiad in Informatics (CEOI) (CEOI, 2024), the European Girls' Mathematical Olympiad (EGMO) (EGMO, 2024), the European Girls' Olympiad in Informatics (EGOI) (EGOI, 2024), and the Baltic Olympiad in Informatics (BOI) (BOI, 2025). These competitions are similar in nature and structure to the IOs, but participation is conditional and often based on factors such as continent (or region), gender, or the country's ranking in the IOs.

In contrast to these and many other competitions, IOs are accessible to any country interested in participating, without restrictions based on location, gender, or prior achievements. IOs are regarded as the world's largest and most challenging competitions for youth in the sciences and humanities.

Table 1
The International Olympiads

Foundation	Scientific Field	Name	Acronyms
1959	Math	International Mathematical Olympiad (IMO, 2023)*	IMO
1967	Physics	International Physics Olympiad (IPhO, 2023)*	IPhO
1968	Chemistry	International Chemistry Olympiad (IChO, 2023)*	IChO
1989	Computer Science/Informatics	International Olympiad in Informatics (IOI, 2023)*	IOI
1990	Biology	International Biology Olympiad (IBO, 2025)*	IBO
1993	Philosophy	International Philosophy Olympiad (IPO, 2023)*	IPO
1996	Astronomy	International Astronomy Olympiad (IAO, 2024)	IAO
1996	Geography	International Geography Olympiad (iGeo, 2024)	iGeo
2003	Linguistics	International Linguistics Olympiad (IOL, 2023)*	IOL
2004	Sciences (Chemistry, Biology, and Physics)	International Junior Science Olympiad (IJSO, 2023)*	IJSO
2007	Earth Sciences	International Earth Science Olympiad (IESO, 2024)*	IESO
2007	Astronomy	International Olympiad on Astronomy and Astrophysics (IOAA, 2024)	IOAA
2015	History	International History Olympiad (IHO, 2024)	IHO
2018	Economy	International Economics Olympiad (IEO, 2024)	IEO
2024	Artificial Intelligence	International Olympiad in AI (IOAI, 2024)	IOAI

*Israeli participation

Fifteen Olympiads (listed in Table 1) fall under the umbrella of the IOs, which include participants from dozens of countries worldwide.

In Section 2, we will describe the International Science Olympiads IMO, IPhO, IChO, and IBO; with more details on IOI; In Section 3, we highlight the case of Israel, including the efforts made to ensure inclusiveness by providing opportunities to all populations, and we explain the crucial role of teachers, and Section 4 presents a brief summary.

2. International Science Olympiads

In the following we provide detailed descriptions of the organizational structure, regulations, delegations, procedure and finally, ranking and medals of the International Science Olympiads (ISOs).

2.1. *Organizational Structure and Regulations*

The organizational structure of the ISOs is generally as follows: The General Assembly (GA) is the main authority. It elects or appoints the Executive Committee. The Executive Committee (EC sometimes called Executive Board) oversees overall operations and helps the next Host Country Organizing Committee. The Host Country Organizing Com-

mittee sets up the Scientific Committees for the competition. The Scientific/Academic committees are composed of experts from various disciplines. Some Olympiads have a small Permanent Secretariat for long-term support. Each year, a new country organizes the Olympiad, but the basic structure remains.

The GA composed of official country representatives. The EC is a smaller elected group managing operations between Olympiads. The Host Country Organizing Committee is a temporary committee responsible for running a specific year's Olympiad.

The rules and regulations for ISOs are generally established and decided upon by the GA. The committees are responsible for coordinating the event, setting the rules and standards, and ensuring that the competition runs smoothly. They also work to maintain consistency and fairness across different countries and regions. In some cases, an Olympiad will be organized by the president of one of the committees, to whom the other committees and bodies report. The international committee of each ISO includes representatives from the participating countries. These committees have several key roles, for example, approving competition regulations, electing a president, and determining future decisions, such as the order of host countries or whether to hold the competition online (for instance, during the COVID-19 pandemic). In addition to their general duties, committees are responsible for overseeing the competition itself. This includes maintaining the quality of competition questions and tasks, managing the scoring process, addressing any issues that arise, and establishing the score cutoffs for medal allocation and criteria for special awards. They also play a vital role in upholding the professional standards and integrity of the ISOs.

Each ISO can establish additional committees, based on the regulations and specific needs of the competition. The mission of these committees is always to support the effective organization, management, and execution of each Olympiad.

The regulations of each ISO may change from time to time, as responses to requests from the representatives of the participating countries or the international committee of the Olympiad. These changes aim to produce an efficient, fair, and ethical competition. The representatives from the participating countries vote on any proposed changes; majority consent is required for their implementation.

Although each Olympiad has its own specific regulations, there are some basic rules common to all. For example, the host country is obliged to invite all the countries that participated in the previous year's Olympiad, without discrimination on political or religious grounds. Regardless of nationality, gender identity, physical ability, religion, or sexual preference, the host country must treat all delegations equally.

The official language of competitions is English. Olympiad questions are translated by coaches and trainers into the native language of participating students. The students' answers are typically written in English although, in some cases, they may also provide answers in their native language as a backup, in the event of an appeal. In natural sciences and exact sciences Olympiads, some of the students' answers may be in the form of formulas or computer code. However, English language proficiency is still important, as some answers require an explanation of the approach and essence of the solution.

2.2. *The Delegations*

Each Olympiad delegation includes the student team and the coaches, trainers, and observers, all led by the chair of the delegation. The students chosen to represent their country are selected through a national competition open to all eligible youth in that country. The trainers and observers are experts in the specific field of knowledge and must be fluent in English. Each year, a different participating country hosts the Olympiad.

Each delegation consists of between four and six students (IMO – up to six students, IPhO – up to five students, IChO, IBO, and IOI – up to four students). Middle school and upper school students up to the age of twenty may participate. Some ISOs, such as the IMO and the IOI, allow home-schooled students to take part, as long as they have not yet completed high school.

Coaches, who are responsible for training the students and accompanying them to the competitions, are integral members of each delegation. The coaches are subject-matter experts with professional backgrounds, some of whom may have even participated in a previous ISO. At the head of the delegation are the leaders, one of whom is designated Head of the delegation, and next in line are the observers who assist the leaders. The coaches serve as mentors to the students in the training process and during the competition itself. This professional team is mandated by the rules and regulations of the ISOs.

Unlike sports competitions, in which coaches have no influence on the judges' evaluation, during ISOs, these mentors play a decisive role and have a significant influence on the final results, in the following realms:

1. **Translation:** The mentors are responsible for translating the exam questions from English into the competitors' native languages. Precise translation in scientific fields is critical for students' understanding of the questions which, in turn, can impact their performance.
2. **Evaluation:** After the competition, judges (appointed by the host country) check and grade the solutions. Mentors evaluate their students' solutions and compare their assessment with the judges' grades.
3. **Appeal process:** If there is a discrepancy between the mentors' and judges' evaluations, a structured appeal process takes place. The mentors conduct scientific-professional negotiations with the competition judges on a dedicated day during the program. This appeal process requires negotiation skills and extensive academic knowledge, as it can affect the final grades and overall results.

The mentors' involvement in translation, evaluation of solutions, and the appeal process (if necessary) can significantly impact the team's achievements in the ISOs.

2.3. *Competition Program*

Each ISO features an organized program that is announced to all participating countries in advance, both directly and through the competition website. The syllabus is published

several months before the event so that all participants may organize accordingly. The Olympiad website, published by the host country, includes comprehensive information about the competition, including the registration process and fees. The site is updated before and after the competition with Olympiad questions, solutions, various statistics, such as the participants' scores, the scores of the leading countries, and the distribution of medals.

Usually, ISOs take place for a week to ten days; a period of time that allows for both the exam days and social activities that reflect the dual goals of the Olympiad: forging friendships between students and professionals from around the world.

Usually, each ISO begins with a solemn opening ceremony, during which all participating countries and delegation members are presented. Some also feature a group ceremony in which students pledge to uphold exam integrity and human dignity. Each event includes two exam days, social activities, and cultural tours. In the IOI, the two exam days are experimental. In the other ISOs the first day is usually devoted to a theoretical and the second day to an experimental. The Olympiad concludes with a closing ceremony during which medals and prizes are awarded.

To preserve exam integrity, competitors are typically isolated from accompanying coaching staff who have had prior access to competition questions. In some ISOs, students are also separated from devices with Internet connections in order to maintain this goal.

2.4. *Medals and Ranking of Countries*

In sport competitions, gold, silver, and bronze medals are awarded to the three athletes with the highest achievements. At ISOs, the distribution of medals is done in order to encourage outstanding youth to engage in, and deepen their study of scientific subjects. Therefore, at each ISO, a large number of medals of every kind, as well as certificates of appreciation and marks of honor are given. Each country's ranking is determined by weighing the results of its delegation.

The distribution of medals is carried out in accordance with the rules established by the International Olympiad Committee and stipulated in the regulations. For example, IMO regulations state that the number of medal winners will not exceed half the number of participants and the ratio between them will be approximately 1 : 2 : 3. For example, if 100 students participate, 50 may receive medals: eight students will receive a gold medal (one-sixth of fifty), 16 will receive a silver medal (two-sixths of 50) and the rest will receive a bronze medal. According to IPhO regulations, gold medals are to be awarded to the 8% with the highest results, silver medals to the following 25% in the order of the results, and bronze medals to 50% of the remaining students. A commendation is awarded to 67% of the participants with the highest scores. For example, if 100 students participated, eight will receive gold, 17 silver, 25 bronze, and 17 students will receive a commendation.

In addition to the distribution of medals, some ISOs add additional categories and special prizes. For example, at the International Biology Olympiad, a prize is given to a

"Social Delegation" (IBO, 2025) for promoting friendships between young people from around the world, and for sharing and exchanging ideas for studying biology education. In the International Physics Olympiad, the award "Absolute Winner" (IPhO, 2023) is given to the student who achieved the highest combined score in both the theoretical and experimental sections of the competition. In the International Math Olympiad, "The Most Outstanding and Creative Solution" (IMO, 2023) prize was awarded.

In addition, some regulations include a section that allows the host country to determine categories for additional prizes as long as they preserve the goals of the competition and the rights of the participants. Some of these categories have become traditional and are included annually. For this reason, during the awards distribution portion of the closing ceremonies, different categories are added. These include "Honorable Mention", "Perfect Score", and "The Best Solution" in the International Math Olympiad; and the "Best Theoretical Score", "Best Experimental Score", and "The Most Creative Solution" in the International Physics Olympiad. The "Absolute Gold" prize is awarded to the student with the heights score in all ISOs.

Despite the Olympiads are individual competitions, with medals and prizes awarded to students who achieve the best results, reference is also made to the ranking of participating countries. Each country's ranking is calculated by weighing the results of the delegation's members. In the ranking process, the points earned by all delegation members are added up, and based on this sum, the country's ranking is determined. According to this system, the countries that stand out for their excellence are China, Russia, and the United States (IMO, 2023; IOI, 2023; IPhO, 2023).

As an example of a specific Olympiad we elaborate in the next section, on the International Informatics Olympiad.

2.5. *The International Informatics Olympiads (IOI)*

The official Website of the IOI (IOI, 2023) provides useful information about the goals, the organization, the participating countries and statistics.

According to the history of the IOI by Mārtiņš Opmanis (IOI, 2023) the concept of organizing an IOI for school students was first proposed at the 24th General Conference of the United Nations Educational, Scientific and Cultural Organization (UNESCO) in Paris by Bulgarian delegate Professor Blagovest Sendov in October 1987. In May 1989, UNESCO launched and sponsored the first IOI, which took place in Bulgaria that same year.

As described above, each year, a different country hosts the event. The competition is supervised by the International Committee and follows the UNESCO-endorsed structure. Teams are officially selected by their national organizations through national contests. Each country can send up to 4 participants, along with team leaders.

The contest consists of two days of algorithmic programming challenges, where students solve problems using programming languages like C++ first, for a few years Java was also supported. The solutions are submitted as files of C++ code which are then compiled and executed on a set of test cases for grading.

Many former IOI participants have gone on to become startup founders and tech leaders: Adam D'Angelo co-founder of Quora received a silver medal in the 2002 competition (IOI, 2023). Nikolai Durov, a four-time medalist at the IOI (1995–1998), co-founded VKontakte (VK, 2023), Russia's largest social network, and later Telegram Messenger, focusing on secure communication (IOI, 2023; Maréchal, 2018). These individuals exemplify how the skills and experiences gained from participating in the IOI can serve as a strong foundation for entrepreneurial success in the technology sector.

In 2007 the IOI community in cooperation with the Institute of Mathematics and Informatics (now Vilnius University Institute of Data Science and Digital Technologies) initiated the journal Olympiads in informatics, an international open access journal (IOI, 2023).

In the following section, we explore Israel's participation in the International Science Olympiads, highlighting its involvement and achievements.

3. The Case of Israel

Israel began competing at ISOs in 1994. In its first year, Israel sent only one delegation to the IPhO (IPhO, 2023). Since then, Israel has expanded its involvement. In 1997, two delegations participated in two additional ISOs: the IMO (IMO, 2023) and the IOI (IOI, 2023). Israel joined the IChO (IChO, 2023) in 2005. A year later, a delegation participated in the IPO (IPO, 2023) for the first time, and in 2012, the first delegation took part in the IESO (IESO, 2024). Over the next ten years, delegations continued to represent Israel at the Olympiads. In 2022, the Israeli delegation took part for the first time in the IBO (IBO, 2025) and in 2023, Israel participated for the first time in the IJSO (IJSO, 2023). In 2012, Israel also participated in the IOL (IOL, 2023). As of 2023, Israel has taken part in nine of the 14 existing IOs.

Israel's participation in ISOs serves several objectives, including enhancing the country's standing in the global science education community, fostering role models for Israeli youth, particularly young women, pursue science studies in middle and high school and encouraging minority groups, such as the Arab population. The Israeli education system has embraced this approach as a means of effectively addressing the needs of gifted students with a strong interest in science. Engaging in these programs allows students to build connections with peers and exceptional students from around the world.

To achieve a high rank in the Olympiads, the most outstanding students must be selected and properly prepared according to the competition's syllabus. Five high school Israeli Science National Teams (ISNTs) have been established for this purpose. Each focuses on a specific field: mathematics (The Israeli Ministry of Education, 2024d), chemistry (The Israeli Ministry of Education, 2024b), physics (The Israeli Ministry of Education, 2024e), biology (The Israeli Ministry of Education, 2024a), and computer science (The Israeli Ministry of Education, 2024c). An additional team, the Israeli Science Young Team (ISYT), serves middle school students and acts as a feeder for the high school ISNTs (IYST, 2024). The teams train throughout the school year and the outstanding students compete at the end of the year to join the delegation representing Israel.

3.1. *Paving the Way to the Olympiads*

In this section we describe the process leading to participation in five science Olympiads of the ISNTs and the ISYT.

3.1.1. *The Israeli National Science Teams (ISNTs)*

Candidates for the teams undergo a structured recruitment and selection process consisting of three stages. In Phase A, students take a 90-minute online exam via the Israel National Teams Website (The Israeli Ministry of Education, 2024f). Any student in the education system is eligible to participate, with the exam available in both Hebrew and Arabic. Some disciplines emphasize creative and scientific thinking without requiring prior knowledge in the sciences, while others demand extensive background knowledge in a specific field and advanced familiarity with the science curriculum. The exam is offered in two sittings, and students may attempt both. The highest score is considered for advancing candidates to the next phase.

Phase B takes place at a number of nationwide centers. The participants chosen in Phase A (with the exception of ISYT) must attend in person. Students who pass this stage will continue to the final phase.

Phase C is a one-day program conducted at an academic host institution for the ISNT. It includes a lecture, an exam, a personal interview, and an overview of the preparation process leading up to the Olympiad.

The transition between Phases B and C differs from field to field. The study materials required for preparing for each stage are published on the Israel National Teams Website (The Israeli Ministry of Education, 2024f) and sent directly to students who have passed Phases A and B. Students who successfully complete Phase C are assigned to the ISNTs based on their area of expertise.

After having been accepted to an ISNT, students undergo a training period during which they learn different areas of knowledge related to their field while emphasizing various study skills. These include independent learning and group collaboration, organization of study materials, time management, strategic development for diverse solutions, perseverance, working under pressure, and quick recovery from failure. In addition, students engage in self-reflection on their activities, draw conclusions, and practice critical, scientific, and creative thinking.

During this stage of the training, emphasis is placed on experimental and playful learning, which helps all team members reach higher achievement.

For examples from the Israeli National Informatics Olympiad, see Appendix A.

3.1.2. *The Israeli Science Young Team (ISYT)*

As mentioned above, the ISYT is a feeder team for the ISNTs. The ISYT is comprised of about 200 students from grades 8 and 9 who undergo a two-year training process. During the 7th grade they take exams, during the 8th grade, students study mathematics, physics,

chemistry, and computer science. In the 9th grade, they specialize in two of the four fields, in coordination with the academic staff and based on their preferences. At the end of the 9th grade, the four most outstanding students are chosen to represent Israel in the International Junior Science Olympiad (IJSO) (IJSO, 2023). In this competition, the students are tested together in the fields of physics, chemistry, and biology. As part of the preparations for this Olympiad, students complete the biology studies which were not covered during their ISYT training.

Like the ISNTs, the ISYT also has three selection phases, with the first two available online (IYST, 2024). In Phase A, the exam consists of math questions that require a numerical answer or a multiple-choice response. Phase B requires students to present the solution process. Phase C takes place at the academic institution hosting the team, in the format of a study day that includes a personal interview, lecture, and a number of exams. These exams test abilities such as mathematical thinking, listening comprehension, reading comprehension, and include questions based on the lecture content.

The ISNTs participants gain significant benefits, the most important for all students is the acquisition of a deep understanding of academic subjects, the development of thinking skills, and the tools gained during the training and coaching process. These include independent learning, dealing with heavy workloads, teamwork, scientific creative thinking, strategies for solving complex and challenging problems, study habits, time management, perseverance, and learning from failure. These tools will become integral to the students and will serve them, not only in the Olympiads, but also throughout their lives.

In addition, ISNT members who have completed a full year of training are entitled to exemption from the Ministry of Education matriculation exam which would normally take place at the end of the training year.

Academic institutions offer benefits to the ISNTs, especially to members of the delegations. These benefits are primarily reflected in tuition fees.

Many ISYT graduates successfully advance to the ISNT selection process. Some even integrate into the ISNTs while still members of the ISYT, depending on their skills and level of emotional readiness for academic challenges.

Israel's performance has improved steadily over the years. For instance, at the 2015 IOI, three of the four Israeli contestants earned bronze medals, placing between 116th and 163rd. By 2022, all four contestants received medals – one gold, two silver, and one bronze – placing between 27th and 92nd (IOI, 2023). At the IMO, Israel was ranked 40th in 2015 and 10th in 2022 (IMO, 2023).

3.2. *Inclusion of all Populations*

In order to give every student an equal opportunity to participate in the ISNTs or the ISYT, the Ministry of Education's unit in charge of the ISOI decided to increase the number of students participating in the program. To this end, the data of students who had participated over the years were examined, and their level of success in the exams

leading up to the ISNTs and acceptance to the Israeli delegations were analyzed[1]. The data were examined with reference to area of residence, socioeconomic status, gender, and identification with specific sectors that make up Israel's population. Based on the analysis, it was decided to operate through several different channels. Changes were implemented to encourage participation by underrepresented groups, including those from the geographic periphery, female students, pupils attending religious state education (HEMED) schools (HEMED, 2024), and the Arab sector.

3.2.1. *Peripheral Communities*

To encourage the participation of students from the geographic and/or socioeconomic periphery, meetings were held with educators and decision makers responsible for these areas. During the meetings, which were attended by inspectors from the Ministry of Education, heads of education departments in local authorities, school administrators, and teachers, the selection process was explained and benefits for team members were highlighted. The meetings also emphasized the image benefits for schools and municipalities, as well as for the country as a whole. These meetings led to the development of a preparatory system for the exams, including classes for the placement tests, study groups for students who had already gone through Phase A, monitoring of student needs, and providing emotional support during the next screening stages. One of the challenges was providing financial support for travel costs to exams. The Ministry of Education and the municipalities had to assist with this support. The students who were accepted to the national team were given recognition and appreciation by heads of municipalities, as well as by teachers and school principals.

3.2.2. *Female Students*

In order to encourage female students to participate, emphasis was placed on marketing and advertising directed to a female audience. This was accomplished by pointing out the achievements of female students and by using written and visual cues in marketing (capitalizing on the grammatical difference between addressing males and females in Hebrew) and by using photos of female students.

Since the advanced selection stages of the recruitment process and the teams themselves are overwhelmingly male-dominated, an effort was made to group several female students from the same school or residential area together during the transition between the phases, in order to provide support and encourage them to reach advanced screening stages.

In addition, it was decided that Israel would participate in two additional Olympiads in which only female students compete: the European Girls' Mathematical Olympiad (EGMO) (EGMO, 2024) and the European Girls' Olympiad in Informatics (EGOI) (EGOI, 2024). The goals of these are to promote excellence among women in mathematics and computer science, to increase the number of female students studying these subjects, and to provide them with an international platform in which to excel.

[1] This was carried out by moderator from the Ministry of Education.

For example, in 2022, the number of female students on the computer science team tripled. In EGMO, Israel was ranked ninth, with students winning four medals (one gold and three bronze) (EGMO, 2024). In the IPhO, a female student from the Israeli delegation won the gold medal (IPhO, 2023). In the IMO, a female student from the Israeli delegation won a silver medal **(**IMO, 2023) and was first place champion of the Asian continent.

3.2.3. *The Jewish Religious Community*

In the Israeli education system, an educational stream called HEMED (Religious State Education) (HEMED, 2024) provides a response to the unique academic needs of religiously observant students. HEMED schools provide gender-separate classrooms in upper grades, and the curriculum provides religious as well as secular studies while ensuring that students can observe religious rituals such as praying, keeping kosher, observing Shabbat, and celebrating Jewish holidays.

In the current setup of the ISNTs and the ISYT, certain situations may conflict with modes of religious lifestyles. Issues may arise with gender-mixed learning, social activities, and training camps that include mixed-gender accommodations.

It was found that school stakeholders sometimes refrained from sending their students to participate in the Olympiads due to concerns that the activities might conflict with their students' religious lifestyle. They feared criticism from parents, and as a result, over the years, fewer students were sent from religious schools.

In a dialogue with the management of HEMED, and with the assistance of administrators and teachers, the rules of conduct of the ISNTs and the ISYT were modified in order to allow religiously observant students to participate. Sabbath observance, keeping kosher, etc. were offered and were also made possible for delegations traveling abroad. It was also suggested that Phase A of the selection process be held at their own HEMED schools (HEMED, 2024). In any case, the decision of whether to participate was left up to each individual and their parents, allowing them to decide according to their religious beliefs.

The possibility of observing religious precepts is also given to members of other religions participating on the national teams.

3.2.4. *The Arab Community*

During the team training period, studies are conducted entirely in Hebrew. In the past, this prevented native Arabic speakers who lacked proficiency in Hebrew from participating in the selection tests and later, from joining the ISNTs or the ISYT. Over the years, it has become evident that the Arab community in Israel has improved its command of the Hebrew language, and students whose mother tongue is Arabic are now better equipped to cope with studies in Hebrew.

However, the timed placement tests remain an obstacle for acceptance to the team. In the past, these screening tests were only given in Hebrew. To overcome this barrier, starting in 2020, the test for Phase A has been translated into Arabic for both test dates.

Additionally, during the exams, a telephone support center was opened to provide answers to questions in Arabic.

For the ISYT, the exams for Phases B and C were also translated. Students who passed Phase C but struggled with Hebrew were provided language improvement lessons. During the selection process, an Arabic-Hebrew dictionary can be used during the exam for students who require assistance.

These efforts will hopefully lead to an increase in participation of Arabic-speaking students, ensuring equal opportunities for all students, regardless of their native language.

3.2.5. *Intervention Impact*

Following the implementation of these interventions, the total number of participants from schools countrywide has increased, especially the number of female students in the ISNTs and the ISYT. Today, many students from the religious education stream have joined the national teams and an increasing number of students from the Arab sector are also joining, especially in the ISYT, which paves the way to the ISNTs.

At the same time, to maintain the growth trend and in order for the screening tests to become part of a school's routine, these processes require continued work and constant effort in order to establish awareness among school stakeholders.

3.3. *The Role of Teachers*

Teachers are the cornerstone of every process in the education system (Gal-Ezer *et al.*, 1995). Armoni and Gal-Ezer in their paper (Armoni and Gal-Ezer, 2023) demonstrate, among other findings, that high-school teachers influence students' later decisions to pursue computing disciplines in higher education. Teachers also play a key role in encouraging student participation, their guidance contributes to students passing the exams required to join these teams and support students in staying committed, which helps maintain a steady flow of qualified participants in national teams over time. Teachers who express enthusiasm, interest, involvement, and care for the subject will influence their students, leading them to participate in the national exams, motivating them to continue the process from throughout all the phases of the selection process.

As key educational influencers, teachers, especially science teachers, play a vital role in raising awareness about the ISNTs and ISYT. This includes explaining the challenges and benefits that students may experience from participating, both during high school and beyond.

In order to effectively prepare students for initial challenges, science teachers can direct all students in their classes to questions from the ISNT's question bank and even solve such questions in class, thereby introducing their pupils to different thinking skills, which are often more challenging than what is needed for the official curriculum.

Upon receiving the results of the Phase A exam, which are also sent to the teachers by the Ministry of Education, teachers share the outcome with their students. If the result

is positive, they can turn the good news into a pivotal moment. Conversely, if the result is negative, teachers can offer emotional support and encourage students to continue participating in other excellence programs. Support and encouragement from teachers is valuable at every stage throughout the process and during team training. Instructors play a vital role, both by showing interest and providing motivation to continue, and on an organizational level, by granting permission for absences or helping students catch up on material missed due to team commitments.

The schools, with the assistance of the teachers, can create a supportive social environment for students who are progressing towards, or are already part of, national teams, turning them into role models. This can be done, for instance, by sharing their stories on school websites and on social networks, highlighting their efforts and the significant investment required to secure a place on the Israeli national teams.

The path to success begins with the first phase of training and testing at a student's school, and continues to worthy achievements with the help of supportive teachers all along the way.

4. Summary

The International Olympiads are the highlight of international school students' competitions. They are the main goal that Ministries of Education or their equivalents worldwide aim for, and the driving factor behind national teams training programs. Each national team member aspires to be part of the delegation representing his or her country in these international competitions.

The IOs serve as a quality incubator for youth with high intellectual abilities who have a passion for science, technology, mathematics, and other disciplines. This enables the development of human capital that will be at the forefront of progress in science, industry, technology, security, health, and the economy.

Israel views human capital as its most important resource. The country's academic and economic strength is based on exporting scientific and technological knowledge. Therefore, developing these academic strengths and fostering excellence in these fields is of utmost importance.

The essence of Israel's efforts goes beyond just preparing for the international competitions. It focuses on creating expert human resources in science, Informatics and mathematics through education for scientific distinction, creative thinking, and complex problem solving.

Within the scope of this paper, we focused on the general structure of five science Olympiads in which Israel participates, with a focus on the informatics Olympiad. The paper offers an overview without delving into the structure of each competition. Despite the local point of view, same as (Cohen *et al.*, 2022; Zohar and Gal-Ezer, 2023), it can serve worldwide populations, who will benefit by following the guidelines and practices described, replicating them, and emphasizing inclusiveness in their own countries.

References

HEMED (2024). Administration of Religious Education (HEMED). The Ministry of Education, State of Israel. https://hemed.education.gov.il/ (in Hebrew). Accessed: 2024, July 29.

Armoni, M., and Gal-Ezer, J. (2023). High-School Computer Science – Its Effect on the Choice of Higher Education. *Informatics in Education*, *22*(2), 183–206.

APhO (2024). *Asian Physics Olympiad (APhO)*. http://asianphysicsolympiad.org/. Accessed: 2024, February 16.

APIO (2024). Asia-Pacific Informatics Olympiad (APIO). https://ioimalaysia.org/competition/apio/. Accessed: 2024, February 16.

BOI (2025). Baltic Olympiad in Informatics (BOI), https://boi2019.eio.ee/. Accessed: 2024, February 16.

Campbell, J. R., and Walberg, H. J. (2010). Olympiad Studies: Competitions Provide Alternatives to Developing Talents That Serve National Interests. Roeper Review, 33(1), 8–17. https://doi.org/10.1080/02783193.2011.530202.

Campbell, J.R., Cho, S., and Tirri, K. (2017). Mathematics and Science Olympiad Studies: The Outcomes of Olympiads and Contributing Factors to Talent Development of Olympians. *International Journal for Talent Development and Creativity*, 5(1–2), 49–60.

CEOI (2024). Central European Olympiad in Informatics (CEOI). http://www.ceoi.inf.elte.hu/. Accessed: 2024, February 16.

Cohen, A., Dolev, S. and Gal-Ezer, J. (2022). The journey of computer science and software engineering in Israeli schools. *ACM Inroads*, 13(3), 29–37. https://doi.org/10.1145/3556879

EGMO (2024). European Girls' Mathematical Olympiad (EGMO). https://www.egmo.org/, EGMO: Israel (ISR): https://www.egmo.org/countries/country47/. Accessed: 2024, October 21.

EGOI (2024). European Girls' Olympiad in Informatics (EGOI). https://egoi.org/about-egoi/. Accessed: 2024, February 16.

Gal-Ezer, J., Beeri, C., Harel, D. and Yehudai, A. (1995). A High-School Program in Computer Science, *IEEE Computer*, 28, 10, 73–80.

Ginat, D. (2015). Intersecting lines. Colorful Challenges Column, ACM Inroads, 6(3), pp. 47–48.

Ginat, D. (2019), Fence levelling. Colorful Challenges Column, *ACM Inroads*, *10*(1), pp. 28–29. https://doi.org/10.15388/infedu.2023.14

IAO (2024). International Astronomy Olympiad (IAO), http://www.issp.ac.ru/iao/. Accessed: 2024, February 16.

IBO (2025). *International Biology Olympiad (IBO)*. https://www.ibo-info.org/en/. Accessed: 2025, April 18.

IChO (2023). *International Chemistry Olympiad (IChO)*. https://www.ichosc.org/. Accessed: 2023, December 18.

IESO (2024). *International Earth Science Olympiad (IESO)*. https://www.ieso2022.com/. Accessed: 2024, February 16.

IEO (2024). *International Economics Olympiad (IEO)*. https://ecolymp.org/. Accessed: 2024, February 16.

iGeo (2024). *International Geography Olympiad (iGeo)*. http://www.geoolympiad.org/. Accessed: 2024, February 16.

IHO (2024). *International History Olympiad (IHO)*. https://www.historyolympiad.com/. Accessed: 2024, February 16.

IJSO (2023). *International Junior Science Olympiad (IJSO)*. https://www.ijsoweb.org/. Accessed: 2023, December 18.

IOL (2023). *International Linguistics Olympiad (IOL)*. https://ioling.org/index.html. Accessed: 2023, December 18.

IMO (2023). *International Mathematical Olympiad (IMO)*. https://www.imo-official.org/. Accessed: 2023, November 25.

IMChO (2024). *International Mendeleev Chemistry Olympiad (IMChO)*. https://mendeleevolympiad.kz/en#about. Accessed: 2024, February 16.

IOAI (2024). *International Olympiad in Artificial Intelligence*. https://ioai-official.org/, Accessed: 2024, April 29.

IOI (2023). *International Olympiad in Informatics (IOI)*. https://ioinformatics.org/. Accessed: 2023, December 18.

IOAA (2024). *International Olympiad on Astronomy and Astrophysics (IOAA)*. https://www.ioaastrophysics.org/. Accessed: 2024, February 16.

IPO (2023). *International Philosophy Olympiad (IPO)*. https://www.philosophy-olympiad.org/. Accessed: 2023, December 18.

IPhO (2023). International Physics Olympiad (IPhO), https://ipho-unofficial.org/. Accessed: 2023, December 18.

The Israeli Ministry of Education (2024a). Israel National Biology Team, Students and Alumni Portal. *The Israeli Ministry of Education*. https://students.education.gov.il/social-and-enrichment-activities/contests/bio (in Hebrew). Accessed: 2024, April 26.

The Israeli Ministry of Education (2024b). Israel National Chemistry Team, Students and Alumni Portal. *The Israeli Ministry of Education*. https://students.education.gov.il/social-and-enrichment-activities/contests/chem (in Hebrew). Accessed: 2024, April 26.

The Israeli Ministry of Education (2024c). Israel National Computer Science Team, Students and Alumni Portal. *The Israeli Ministry of Education*, https://students.education.gov.il/social-and-enrichment-activities/contests/comp (in Hebrew). Accessed: 2024, April 26.

The Israeli Ministry of Education (2024d). Israel National Mathematics Team, Students and Alumni Portal. *The Israeli Ministry of Education*. https://students.education.gov.il/social-and-enrichment-activities/contests/math (in Hebrew). Accessed: 2024, April 26.

The Israeli Ministry of Education (2024e). Israel National Physics Team, Students and Alumni Portal. *The Israeli Ministry of Education*. https://students.education.gov.il/social-and-enrichment-activities/contests/phy (in Hebrew). Accessed: 2024, April 26.

The Israeli Ministry of Education (2024f). Israel's Science Teams – The International Olympiads, Student and Graduate Portal. *Israeli Ministry of Education*. https://students.education.gov.il/social-and-enrichment-activities/contests/international-science-olympiads (in Hebrew). Accessed: 2024, October 21.

IYST (2024). Israel's Young Science Team, Pedagogical space, Teaching Portal, state of Israel, Ministry of Education, https://pop.education.gov.il/olimpiadot-madaim/nivheret-israel-tzeira/ (in Hebrew). Accessed: 2024, October 21.

Jovanov, M., and Stankov, E. (2020). Introduction of "Honorable Mention" award at the International Olympiad in Informatics. *Olympiads in Informatics*, 14, 87–104.

Lim, S. S. L., Cheah, HM., and Hor, T. S. A. (2014). *Science Olympiads as Vehicles for Identifying Talent in the Sciences: The Singapore Experience*. In: Tan Wee Hin, L., Subramaniam, R. (eds) Communicating Science to the Public. Springer, Dordrecht. pp. 195–211. https://doi.org/10.1007/978-94-017-9097-0_12

Maréchal, N. (2018). From Russia with crypto: A political history of Telegram. The *8th USENIX Workshop on Free and Open Communications on the Internet* (*FOCI'18*). USENIX Association.

RMM (2024). *Romanian Master in Mathematics (RMM)*. https://rmms.lbi.ro/rmm2023/index.php?id=home. Accessed: 2024, February 16.

Smith, K. N., Jaeger, A. J., and Thomas, D. (2021). Science Olympiad Is Why I'm Here: the Influence of an Early STEM Program on College and Major Choice. Res Sci Educ 51 (Suppl 1), 443–459. https://doi.org/10.1007/s11165-019-09897-7

Zohar, D., and Gal-Ezer, J. (2023). Navigating to Tomorrow's High-Tech Landscape: Outlining a Path Based on the Israeli Case, *ACM Inroads*, 14(4), 51–56. https://doi.org/10.1145/3630606

J. Gal-Ezer is Professor Emerita of Computer Science (CS) at the Open University of Israel (OUI), where she played a pivotal role in developing Math and CS courses, and designing the undergraduate and master's programs in CS. She has held significant positions, including Head of the Math and CS Department and Vice President for Academic Affairs. Her research focuses on CS education. She holds key roles in various prestigious committees, including the ACM Education Board, the I4All Coalition, chairs the ACM Europe Education Committee and is member of the EU expert group for teaching Informatics in school. She is ACM Fellow. And the recipient of the ACM SIGCSE Award, the IEEE Taylor L. Booth Education Award, and the ACM Karl V. Karlstrom Outstanding Educator Award.

D. Zohar is an educator specializing in science and technology, with over 30 years of experience in higher education, educational policy, and secondary-school instruction in Israel. At the OUI, he teaches computer science courses, coordinates curriculum development, and develops multimedia instructional materials. He also supports pedagogical initiatives aimed at encouraging STEM careers across the national education system. From 2018 to 2019, Dr. Zohar served as Acting Chief Inspector for Computer Science at Israel's Ministry of Education, overseeing the K7-12 curriculum and managing the 5-unit matriculation exams. Since 1996, Dr. Zohar has taught computer science at Ohel Shem High School, preparing students for advanced-level matriculation exams in CS. He coordinates the "Academia in High School" initiative, a collaborative effort between the Ministry of Education and the Open University of Israel. Throughout his career, Dr. Zohar has focused on enhancing excellence, equity, and accessibility in computer science education.

A. Rolnik, an educator and jurist, has been a prominent figure in science and technology education in Israel for four decades. In recent years, her work has focused on advancing scientific and technological excellence within the national education system. Among her notable achievements, Anat was one of the founders of the Scientific-Technological Cadet Program, which transformed the national approach to science and technology education and led to a significant increase in the number of students earning high-quality matriculation certificates. She also led Israel's national science teams for six years, founded both the Israeli Biology Team and the Young Israeli Team, and helped expand opportunities for excellence among students nationwide. Most recently, Anat directed the Israeli High-Tech Program at the Ministry of Education, working in close cooperation with the public sector, high-tech companies, NGOs, and philanthropic foundations to promote excellence and strengthen the connection between education and the high-tech industry.

Appendix A: Examples from the Informatics Olympiads

The core of the IOI competitions is algorithmic problem solving. Today online preparation sessions are offered, these sessions help students develop skills in writing pseudocode (between Phases A and B) and basic programming (between Phases B and C).

Such algorithmic problems can be found in David Ginat's column in the *ACM InRoads* Magazine, for example: "Given the points 1 .. N in the X-axis and the points 1 .. N in Y-axis of a Cartesian coordinates system, where N is even; each point i on each axis is connected, with two lines, to two points on the other axis – the points i and N-i + 1; devise a recursive solution for computing the number of line intersections. For example, for N = 2, the answer is 1; for N = 4 the answer is 10. (If two lines meet on one of the axes, we do not consider it intersection.) The challenge may be solved directly, with suitable counting. It may also be solved recursively in some cumbersome ways. We are asking for an elegant, illuminating, recursive solution." (Ginat, 13) The emphasis on simple and elegant solutions makes these problems even more challenging.

In (Ginat, 14) the problem presented involves optimizations task, which was designed to exercise induction and abstraction. Through these and other algorithmic challenges, students refine their abstraction and creativity, deepening their understanding of algorithmic problem-solving – the core of computer science.

Olympiads in Informatics, 2025, Vol. 19, 63–86

DOI: 10.15388/ioi.2025.05

Evaluating Interactive Tasks through the Lens of Computational and Algebraic Thinking, Interactivity Types, and Multimedia Design Principles

Yasemin GULBAHAR[1], Tugba ÖZTÜRK[2], Valentina DAGIENĖ[3], Marika PARVIAINEN[4], Ismail GÜVEN[2], Javier BILBAO[5]

[1]*Teachers College, Columbia University, USA*
[2]*Ankara University, Türkiye*
[3]*Vilnius University, Lithuania*
[4]*University of Turku, Finland*
[5]*University of the Basque Country (UPV/EHU), Spain*
e-mail: gulbahar@tc.columbia.edu, tozturk@ankara.edu.tr, valentina.dagiene@mif.vu.lt, mhparv@utu.fi, guveni@ankara.edu.tr, javier.bilbao@ehu.eus

Abstract. This study investigates student engagement within an online assessment environment, focusing on the interplay between interactivity types, and multimedia design. We analyzed (1) descriptive patterns of student engagement and performance, (2) time-on-task in relation to interactivity type of tasks, (3) the frequency and distribution of multimedia design principles (e.g., contiguity, modality) across tasks designed to elicit computational and algebraic thinking, and (4) the correlation between the application of these principles and students' average scores. Findings reveal distinct engagement patterns and performance levels across task types. Time spent varied significantly by interactivity, with certain types eliciting more engagement. Specific multimedia principles were more prevalent in particular tasks, and correlational analysis indicated relationships between the application of certain design principles and student performance outcomes. These insights contribute to a more nuanced understanding of how interactive design and multimedia integration can influence learning in computationally and algebraically rich online environments.

Keywords: interactive tasks, multimedia principles, computational thinking, algebraic thinking.

1. Introduction

The modern era in which we live differs significantly from the past, and hence, there are substantial differences in the skills that individuals are expected to possess. In today's competitive world, international non-governmental organizations such as the World Economic Forum (WEF) regularly update the 21st-century skills which have been frequently identified and featured in the WEF's Future of Jobs reports (World Economic

Forum, 2020). Some of these skills have attracted scholars' attention, with Computational Thinking (CT) as one of these crucial key skills (Agbo *et al.*, 2023; Curzon *et al.*, 2009; Dede *et al.*, 2013; Papadakis, 2022; Voogt *et al.*, 2015; Yang *et al.*, 2023). CT is a term used to define the ability to perform logical reasoning and algorithmic thinking (Denning, 2009). Not just computer scientists, but individuals from various disciplines should possess CT skills (Wing, 2006). CT is said to encompass a set of mental processes, algorithms, and solutions utilized in formulating problems (Grover & Pea, 2013). From a wider perspective, the International Society for Technology in Education (ISTE) and the Computer Science Teachers Association (2011) formulated an operational definition for CT, where they included core skills such as problem solving, decomposition, abstraction, algorithm design and pattern recognition.

While the definitions of CT have varied over time, the consensus is that it is a particularly significant skill for today's world. This disposition leads to the question: "What makes CT one of the most important skills of the 21st century?". Promoting CT skills is essential in educational settings since these skills may serve as a catalyst for learners to understand effective use of technology for solving real-life problems. For example, the ISTE and CSTA (2011) suggested that CT can enhance students' academic performance and better prepare them to join today's competitive workforce. Additionally, Gülbahar *et al.* (2019) argued that CT can expand students' thinking horizons, whilst Tripon (2022) opined that CT can enhance students' lifelong learning skills. Furthermore, Korkmaz and Bai (2019) emphasised that CT is a critical factor in shaping students' attitudes toward science and technology. Numerous studies in the literature have emphasised the significance of CT skills and their implications. The ISTE and CSTA (2011) underlined the existence of a close link between CT and algorithmic and arithmetic thinking. Additionally, Shang *et al.* (2023) revealed a positive relationship between CT and students' self-efficacy, while Polat and Yilmaz (2022) highlighted the correlation between CT and academic achievement.

All of these research studies provided evidence that CT is an important skill that every child has to learn. Based on this fact, numerous research studies have revealed the effect of different plugged and unplugged pedagogical approaches in teaching-learning processes. Although a number of tools exist that could be utilized in accordance with this purpose, the researchers agreed on the use of tasks from Bebras Challenge as the subject of inquiry in the current study, due to its international impact and role on society (`https://www.bebras.org/`).

Computational thinking (CT) is a problem-solving process that involves formulating problems in a way that enables us to use computer-based tools and techniques to help solve them. It's not just about programming, but rather a set of cognitive skills and approaches applicable across many disciplines. The most commonly cited elements of computational thinking include: algorithm design (developing step-by-step procedures or rules to solve a problem), decomposition (breaking down a complex problem into smaller, more manageable parts), pattern recognition (identifying similarities and recurring structures within problems or data), abstraction (focusing on the essential information while ignoring irrelevant details), and evaluation (assessing the effectiveness and efficiency of a solution).

On the other hand, Algebraic Thinking (AT) is a way of thinking about mathematical situations that focuses on generalizing relationships, identifying patterns, and representing these relationships using symbols. It is about moving beyond just calculating with specific numbers to understanding the underlying structure and properties of mathematical operations and systems. AT comprises four thinking strands (Pitta-Panzazi *et al.*, 2020): generalized arithmetic, functional thinking, modeling languages, and algebraic proof (i.e., abstract algebra). AT includes a shift of attention from the result to the process in problem-solving situations (Malara & Navarra, 2018) where some ideas merge and overlap with CT. So, we can talk about some shared features in CT and AT, as follows:

- Algorithmic thinking algorithm design.
- Problem-solving process, abstraction skills.
- Concepts of equality, variable, algorithm and function.
- Defining patterning, generalization and parametrization.

The assessment of computational and algebraic thinking within interactive tasks should consider several key dimensions. Firstly, it should evaluate students' mastery of algorithms, specifically their capacity to comprehend, execute, and create algorithmic procedures, encompassing skills such as decomposition into smaller steps and generalization to broader contexts. Secondly, it needs to gauge their proficiency in problem solving and working with data, including their abilities to analyze, represent in meaningful ways, and derive insights through visualization and abstraction. Furthermore, the evaluation should focus on students' skills in dealing with concepts like equality, variables and function, as well as structures in the problems. Finally, the assessment should probe their ability to identify patterns and, importantly, to generalize from observed examples to formulate broader rules or principles. These interconnected dimensions provide a comprehensive view of how interactive tasks can foster and reveal students' computational thinking abilities.

2. Interactive Tasks

In today's skills-based world, the promotion of digital tools for teaching and learning has become increasingly important. The substantial support provided by technology-assisted approaches to education has had a significant impact on educational outcomes. These solutions also provide interaction that is crucial in the acquisition of critical skills, as they enable the transformation of learners' cognitive structures and facilitate effective learning (Johnson *et al.*, 2014; Moore, 1989). Since interaction emerged as a key concept to the cultivation of valuable understanding and practicing in learning contexts, technology-assisted solutions are used to help enhance interaction in educational settings through a wide range of means and methods. As such, computer games and digital simulations are utilized to teach essential skills such as CT (Basawapatna *et al.*, 2011; 2014) with context-appropriate elements of interaction.

Gaining importance in recent years, the literature illustrates a multitude of works that have addressed interactivity in teaching and learning processes, and the utiliza-

tion of Bebras tasks to impact upon learners' CT skills. For example, Lutz *et al.* (2018) used Bebras tasks to promote students' CT skills in programming training, and stated that a positive relation exists between CT and task completion. Similarly, Chiazzese *et al.* (2018) performed project-based robotics training with the help of Bebras tasks to improve learners' CT skills, and revealed that a positive effect was seen in supporting learners' CT skills. These examples provide insight into how CT skills can be promoted and supported by using Bebras task-oriented interactive products, and how inspiration can be drawn from such attempts to enhance learners' CT skills through effective design with the use of multimedia design principles.

Interactive tasks employ a variety of interaction types to engage learners. 'Click-on-object' interactions allow users to manipulate the state of elements, such as changes in color, shape, or orientation, often revealing underlying concepts or triggering events. 'Drag and drop' functionalities enable users to physically manipulate digital objects, facilitating tasks like categorization, sequencing, or spatial reasoning. Tasks that involve 'Writing text or integers' provide opportunities for students to input answers, define variables, or express their understanding in a textual or numerical format. Finally, 'Click & draw' interactions, such as creating arrows or freehand markings, can support learners in annotating diagrams, illustrating relationships, or visualizing their thinking directly on the interface. These diverse interactivity types offer different affordances for engaging with the learning content.

Incorporating CTML principles into the design of instructional materials has been recommended to enhance learners' achievement, and also to decrease their cognitive load. Various investigations in the existing literature have examined the effectiveness of CTML principles (Basawapatna *et al.*, 2013; Schnotz & Bannert, 2003; Selby, 2015; Torcasio & Sweller, 2010) in promoting CT whereas some studies on CT have identified significant gaps. For example, Tang *et al.* (2020) conducted a systematic review which revealed that CT assessments heavily prioritize programming skills. The researchers also proposed that additional assessments, particularly those focused on quality, are greatly needed in order to achieve better results. Similarly, Hsu *et al.* (2018) reported that CT is primarily used in activities based on computer science. Given that cognitive abilities vary among learners of different ages, it logically sounds that the most appropriate methods and ways should be adopted to promote CT skills, and that this may involve the use of varied approaches so as to effectively foster CT skills at different stages of development.

3. Cognitive Theory of Multimedia Learning

Richard Mayer's cognitive theory of multimedia learning centers on the idea that our working memory has a limited capacity, which can easily be overwhelmed by too much simultaneous information. Mayer proposes that learning is an active process where we build our own understanding from what we are presented with. He also suggests that using both visuals and audio in learning materials can improve understanding by engaging multiple senses and linking different pieces of information. Rooted in cognitive psy-

chology, Mayer's theory aims to optimize multimedia design to match how our minds process information. A key aspect is the **limited capacity of working memory**, meaning instructional materials should be well-structured to avoid overload. Furthermore, Mayer emphasizes **active learning**, where learners actively select, organize, and integrate information, and his principles guide the creation of materials that support these processes for better learning.

Mayer's (2009) cognitive theory of multimedia learning is based on cognitive load theory, which states that our working memory has a limited capacity for processing information at once. Mayer's theory suggests that the information we encounter during learning leads to three kinds of cognitive processing: selective processing, where learners focus on only one part of the information, typically when it's simple; coherent processing, where learners connect new information to what they already know, leading to deeper understanding; and generative processing, where learners actively engage with the material to create new mental representations, which is considered the most effective for learning.

Mayer's theory proposes that multimedia presentations can be structured to encourage these various types of cognitive processing, ultimately leading to improved learning results. For instance, when multimedia is designed to promote generative processing, learners are more likely to remember and use the information in practical contexts.

Extraneous load, or wasted mental effort, arises from non-essential elements that do not aid learning. To reduce this, instructors should stick to the core content and avoid distractions like unnecessary animations or irrelevant details.

Intrinsic load, or the essential mental effort of processing the material itself, depends on its complexity. Instructors can manage this by breaking down content into smaller parts and explaining technical terms beforehand.

Germane load, or the effort learners put into truly understanding the material, is heavily influenced by their motivation. Instructors can boost this by providing learning support and pacing the content well.

Essentially, cognitive load theory expands on the idea of our limited processing capacity. It suggests that to maximize learning and memory, instructors should create multimedia that appropriately manage intrinsic load, optimize germane load, and minimize extraneous load. Consequently, Mayer's principles for effective multimedia design can also be understood as practical strategies for managing cognitive load.

Briefly, the cognitive theory of multimedia learning posits that our minds have separate visual and auditory channels with limited processing power, and that learning is an active process. Therefore, those creating multimedia should design their presentations to carefully manage the demands on these cognitive resources. Mayer views multimedia as tools that help learners build their own understanding, not just as ways to transmit information.

In order to promote effective learning, interactive learning environments that incorporate information and communication technologies are being developed in response to the rapid evolution seen in the use of digital technologies (Liu & Szabo, 2009). Owing to

the fact that interaction within learning environments is often facilitated through the use of multimedia elements, Mayer (2013) developed the Cognitive Theory of Multimedia Learning (CTML) to provide a theoretical foundation for the development of multimedia learning environments. Since then, CTML has been identified as a critical theory for instructional technology designers, as it aims to facilitate efficient learning for learners (Cavanagh & Kiersch, 2022).

4. Principles of Multimedia Learning

Drawing on his theories of learning, Richard Mayer outlined twelve principles to guide the creation of effective multimedia learning tools. These guidelines, listed below, aim to help designers produce engaging materials that lead to better understanding and memory.

1. **Multimedia Principle:** Combining words and pictures is more effective for learning than using words alone.
2. **Modality Principle:** Presenting related information visually and audibly is better than using just one of these methods.
3. **Redundancy Principle:** Avoid presenting the same information both as on-screen text and as spoken narration.
4. **Coherence Principle:** Learning is improved by excluding any unnecessary content.
5. **Signaling Principle:** Using cues to highlight important information and the structure of the material enhances learning.
6. **Spatial Contiguity Principle:** Placing related text and visuals close together improves understanding.
7. **Temporal Contiguity Principle:** Presenting spoken words and corresponding graphics simultaneously is more effective.
8. **Segmenting Principle:** Breaking down lessons into smaller, manageable parts helps learners.
9. **Pre-training Principle:** Providing necessary background information before the lesson improves learning.
10. **Image Principle:** Use relevant graphics that help explain the content and support learning objectives.
11. **Personalization Principle:** Avoid formal language and instead use a conversational tone to engage learners.
12. **Voice Principle:** A friendly human voice is better for narration than a computer-generated one.

Ultimately, Mayer's principles offer practical advice for developing effective multimedia learning experiences. By applying these guidelines, creators can design materials that actively involve learners and lead to more profound understanding and better retention of the information.

Among these principles, seven of them, namely coherence, signaling, spatial contiguity, temporal contiguity, segmenting, pre-training, and image principles, which are

applicable for text and visuals as well as use of interactivity, are further investigated according to their potential for minimizing extraneous, intrinsic and germane load.

4.1. *Principles That Minimize Extraneous Load*

4.1.1. *The Coherence Principle*

According to the Coherence Principle, people learn better when multimedia messages exclude extraneous material. This means instructors should avoid adding information that will not be tested, is just for show, or distracts from the learning goals. Mayer also points out the danger of "seductive details" – engaging but irrelevant content that can be better remembered than the main points, thus interfering with learners' construction of understanding. To adhere to this, multimedia should only contain learning-relevant graphics, text, and narration, should not include background music, and should use simple visuals.

In the 'Writing text or integers' type interactive task presented in Fig. 1, the use of several simple visuals aims to just improve learning, in other words, students can understand the change in the pattern by just observing relevant graphics.

4.1.2. *The Signaling Principle*

According to the Signaling Principle, adding cues that emphasize the organization of key material helps people learn better. When multiple elements are visible, learners benefit from knowing what to focus on, their current location in the presentation, and how the information fits together to build their mental models. Thus, instructors should use signals to draw attention to important content, but Mayer cautions against

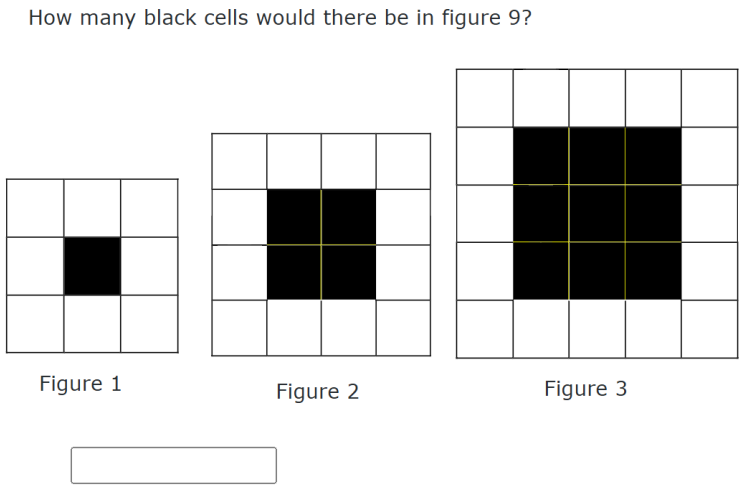


Fig. 1. Example of an interactive task: Writing text or entering integers.

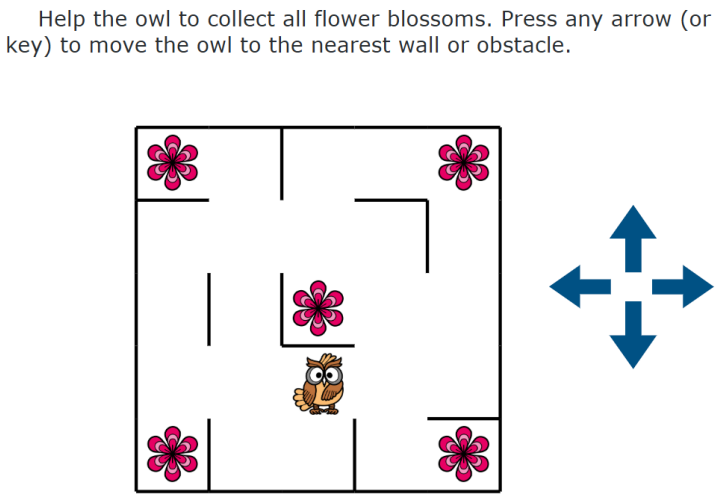


Fig. 2. Example of an interactive task: Click-on-object.

overuse. To apply this principle, use tools for highlighting for emphasis, and provide an advanced organizer that outlines the presentation's structure, referring back to it as you progress.

For example, in the 'Click-on-object' type interactive task seen in Fig. 2, arrows are used for moving the bird to the target. Hence, arrows are the signals in this question which provides interactivity and step by step trial.

4.1.4. *The Spatial Contiguity Principle*

The Spatial Contiguity Principle suggests that learners understand better when related text and visuals are placed close together on the screen or page. This intuitively means keeping labels and captions near the graphics they describe. Doing so reduces the mental effort learners need to connect the text and images, allowing them to focus on understanding and integrating the information instead of scanning the screen to make those links. To apply this, place text near its corresponding graphics, provide feedback close to related questions or answers, present instructions on the same screen as the activity, and have learners read any text before an animation begins.

As it is seen in the 'Writing text or integers' type of interactive question in Fig. 3, although there are too many shapes and words which could lead to a potential cognitive load, combination of verbal and visual elements closer to each other make it easy to understand the question.

4.1.5. *The Temporal Contiguity Principle*

The Temporal Contiguity Principle states that learning is more effective when related words (narration) and pictures (animation) are presented at the same time rather than one after the other. To maximize learning, narration should be synchronized with the

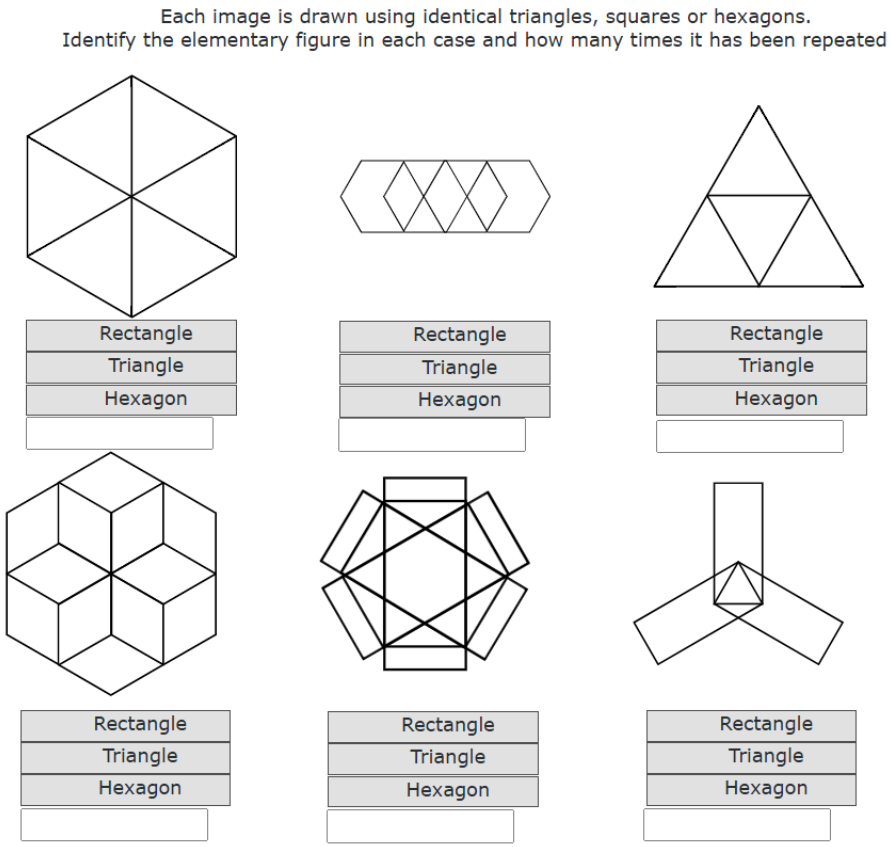


Fig. 3. Example of an interactive task: Writing text (several times).

graphics, rather than having students read an explanation and then see the visuals separately. To apply this, ensure that narration is timed to coincide with the corresponding animations.

For example, in the 'Drag and drop' type interactive task seen in Fig. 4, having all text and graphics altogether makes it easy to understand the problem, and through interactivity students can try and learn even if they make mistakes.

4.2. *Principles That Manage Intrinsic Load*

4.2.1. *The Segmenting Principle*

According to the Segmenting Principle, people learn better when they can proceed through a multimedia message at their own pace, rather than having it presented continuously. Mayer's research with asynchronous lessons on processes showed that allowing students to control the speed improved their recall and transfer abilities. Therefore,

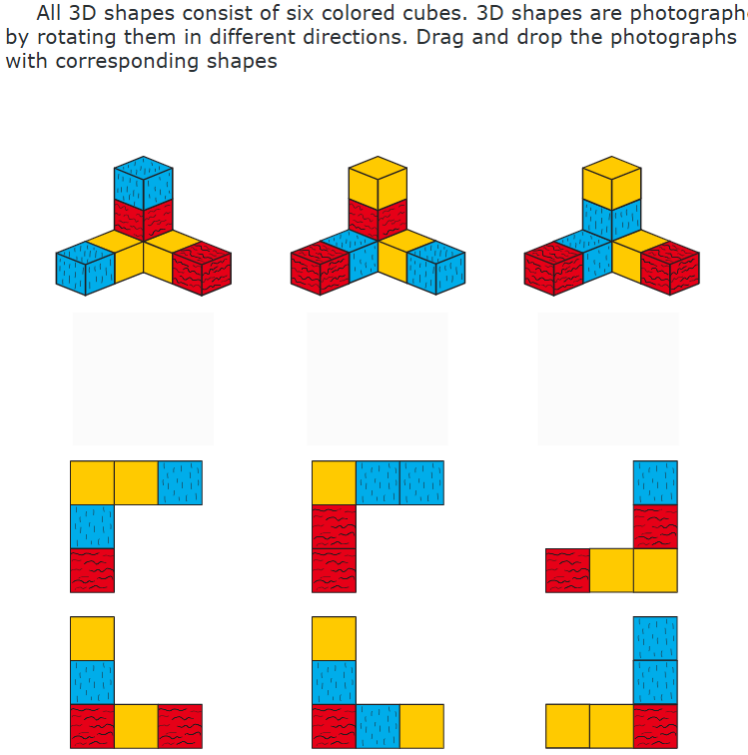


Fig. 4. Example of an interactive task: Drag and drop.

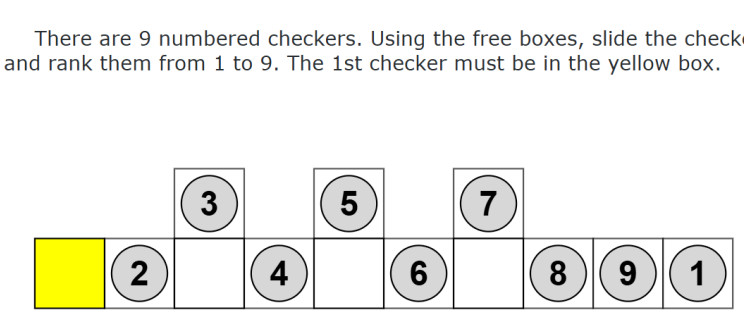


Fig. 5. Example of an interactive task: Drag and drop.

this principle advises giving users control over the lesson's pace (like speed controls or "next" buttons) and breaking down extensive content into smaller segments to allow for sufficient processing at each step.

In the 'Drag and drop' type interactive task seen in Fig. 5, the users can drag numbers and drop empty places having the control over the sorting process, so they can observe each step through interaction, which allows them to work at their own pace.

4.2.2. *The Pre-Training Principle*

The Pre-training Principle suggests that learners understand more deeply when they are familiar with the names and features of the main ideas before the lesson. Because managing the inherent complexity of information can easily overwhelm beginners, instructors should introduce key terms and concepts before explaining processes. Without this pre-training, students might struggle to learn the components of a process while simultaneously trying to understand the process itself, hindering their learning. Essentially, pre-training scaffolds learning by ensuring students have the necessary background knowledge before starting a multimedia lesson.

For example, in the 'Writing text or integers' type of interactive task seen in Fig. 6, the students need some pre-training in order to be able to solve the task. Hence, the

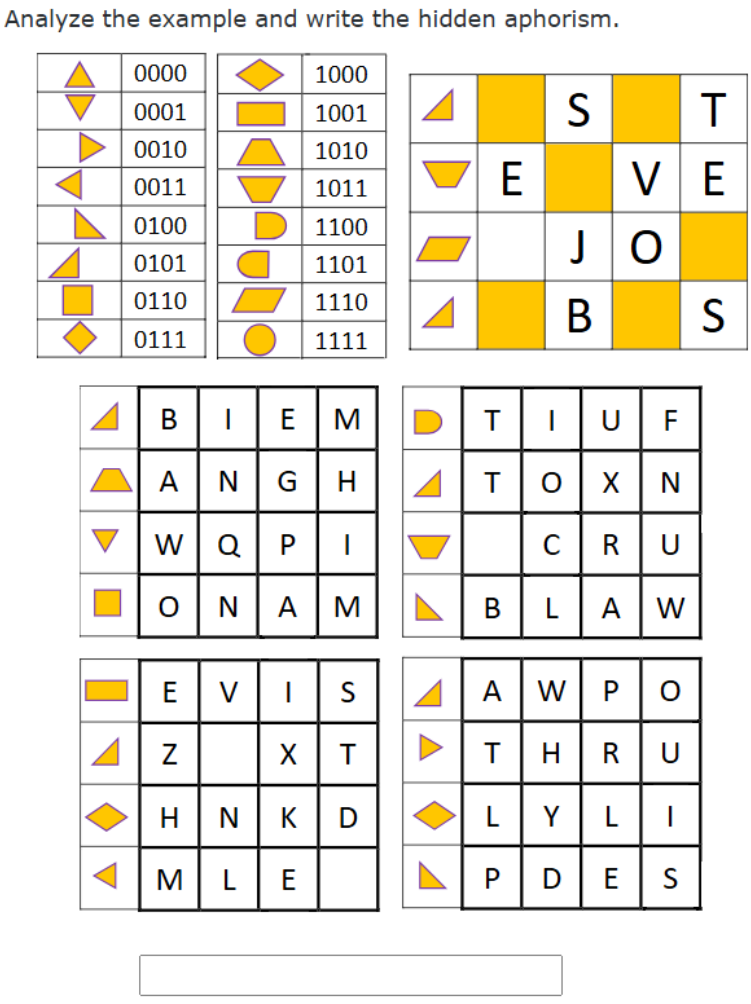


Fig. 6. Example of an interactive task: Writing text or integers.

question provides some clues prior to asking the question. This is where students have to understand the codes related to shapes, so they can apply this information to solve the actual problem.

4.3. *Principle That Optimize Germane Load*

4.3.1. *The Image Principle*

The Image Principle encourages the use of diagrams, animations, or other visual aids that directly relate to the topic being discussed. Using visuals, symbols and images while aligning text closely with these visual cues make information easy to follow or supports easy understanding of concepts and processes.

In the 'Drag and drop' type interactive task seen in Fig. 7, the text explanations appear before and after the visual in order to improve the effect of the visual besides text, where use of images support easy understanding of the explanations provided as text.

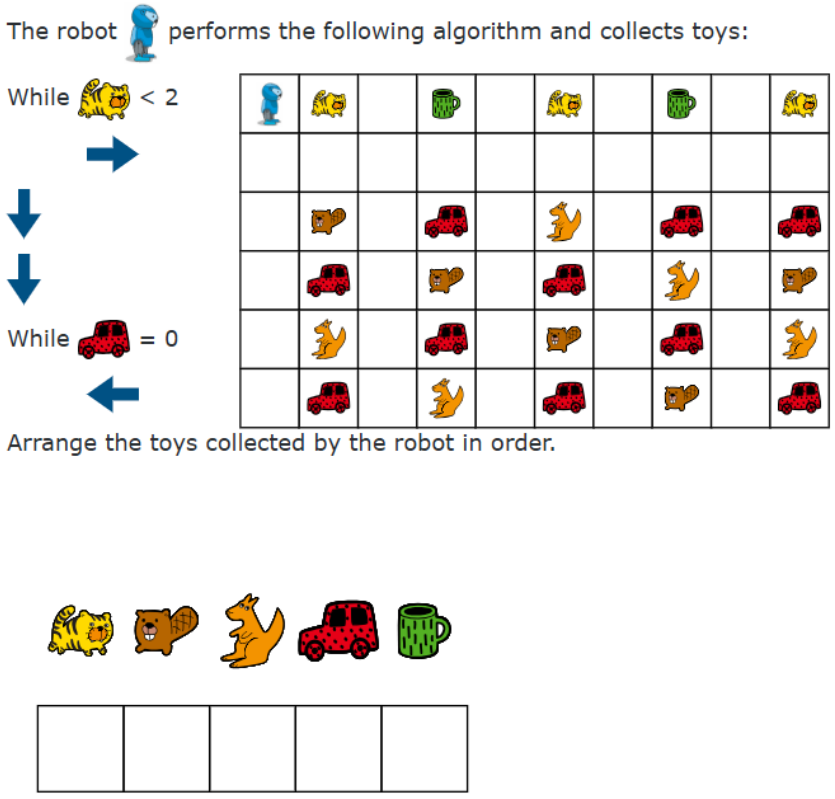


Fig. 7. Example of an interactive task: Drag and drop.

4.4. *Stating the Gap in the Literature*

Research consistently highlights the significant potential of using digital interactive tasks in mathematics education to support thinking processes and foster deeper learning (Mierluş-Mazilu & Yilmaz, 2023; Drijvers & Sinclair, 2024). These tasks, leveraging elements like click-on interactions, drag-and-drop, text input, and drawing tools, can engage students in ways that traditional methods often cannot.

Interactive tasks, especially those incorporating game-based learning or real-world applications, can significantly increase student engagement and motivation in mathematics (Wang, Chang, Hwang, & Chen, 2018; Hwa, 2018; Moon & Ke, 2020). The dynamic and often playful nature of these tasks can also make learning more enjoyable and less intimidating, fostering a more positive attitude towards math (Applebaum, 2025).

Interactivity allows students to visualize and manipulate abstract mathematical concepts, leading to a deeper and more intuitive understanding (Ziatdinov & Valles Jr, 2022). Tasks that require active participation and experimentation help students break down complex ideas and build stronger cognitive connections (Boaler, 2022).

Interactive tasks often require students to actively solve problems and think critically, contributing to the development of these essential skills (Blyznyuk & Kachak, 2024). Studies have shown the positive impact of interactive math apps on early math learning gains, including basic facts, concepts, and higher-level reasoning (Bang, Li & Flynn, 2023; Clements, Lizcano & Sarama, 2023). Besides, digital platforms with features like dynamic representations, simulations, and virtual manipulatives help students engage with math in more meaningful ways (Cirneanu & Moldoveanu, 2024; Bush, 2021).

In conclusion, research strongly supports the use of digital interactive tasks in math teaching as a way to enhance engagement, understanding, and skill development. However, the effectiveness hinges on thoughtful design, appropriate pedagogical integration, and equitable access for all learners. Hence, while the potential is high, the design of effective interactive tasks is crucial to maximize their benefits, since simply digitizing traditional exercises may not yield the desired outcomes.

5. Research Methodology

Based on the literature review and theoretical basis provided, the purpose of this article is to analyze interactive tasks based on the computational and algebraic thinking they promote, the nature of their interactivity, and their adherence to established multimedia principles. To address this issue, we tried to answer the following research questions:

1. What are the typical patterns of student engagement (types of interaction, time spent) within the online learning environment?
2. How does the average time students spend on a task vary across different types of interactive tasks?

3. How frequently are various multimedia design principles (e.g., contiguity, modality, coherence, redundancy) applied across the learning tasks?
4. Is there a statistically significant correlation between the extent to which multimedia design principles are applied in a task and the average scores achieved by students on that task?

By addressing these research questions, the study will provide a comprehensive analysis of the multimedia design effectiveness of the interactive mathematics questions for students. The findings will not only inform the refinement of the platform but also contribute to our understanding of how interactive design and multimedia principles can be optimally applied in educational contexts to foster critical thinking skills.

Hence, this research study is based on quantitative data analysis where descriptive analysis as well as correlational analysis were used.

5.1. *Working Group*

Data from a total of 63 students (Grades 1–2) were used in this study (Table 1). However, some of them did not provide answers to some questions. So, the total number of students answering each question shows variety for each group and task.

5.2. *Interactive Tasks*

The interactive tasks (which are called as COMATH items in the context of the project) used within this research are developed in the scope of CT&MathABLE project (2025) which focuses on benchmarking Computational and Algebraic thinking skills in school systems in the project partner countries. The tasks are provided through a digital online learning platform, ViLLe, which focuses on learning analytics.

Interactive tasks in the Computational Thinking (CT) Dimension of the scale are derived from items developed for the 2022 International Bebras Challenge (Dagienė and Stupurienė, 2016). Algebraic Thinking (AT) items were developed at University of Turku by staff in the Turku Research Centre for Learning Analytics during Autumn of 2023 and Spring of 2024.

Table 1
Demographic Information about Working Group

Grades	Number of Participants	Female	Male
Grades 1–2	63	27	36
Grades 3–4	57	24	33
Grades 5–6	59	26	33
Grades 7–8	28	19	9
Total	**207**	**96**	**111**

A total of 207 students participated in the study; however, the analysis focused on Grade 1 and 2 students, as they represented the largest subgroup and offered the most suitable data for the research objectives.

Interactive tasks for this research study are provided and accessed through ViLLE Platform. ViLLE is a collaborative learning platform[1] developed by the Centre of Learning Analytics of the University of Turku. It offers students and teachers detailed information regarding their learning process in the form of immediate feedback and learning analytics. Teachers can create exercises for their personal use but they can also utilize materials made by others. Most exercises are automatically assessed which allows teachers to spend more of their time supporting students.

In response to the significant findings reported in the literature and Mayer's (2017) call for future research on the use of educational multimedia designed in accordance with CTML principles in educational contexts, the current study explores the integration of these principles in interactive tasks. ViLLE system is not supporting audio and video content, so this research study will focus on the principles which are applicable for text and visuals. The interactive tasks will be evaluated using seven of the multimedia design principles considered appropriate to the characteristics of interactive tasks, which are such as the use of coherence, signaling, spatial contiguity, temporal contiguity, segmenting, pre-training, and multimedia principles.

5.3. *Data Collection*

Data for this study were collected through the implementation of interactive tasks using the ViLLE system. From the system logs, we extracted several key variables: "scores," representing the points students earned for each task; "submissions," indicating the timestamp when a student clicked "Submit" and received immediate feedback; and "time on task," reflecting the total duration a student spent completing each task. Additionally, the dataset includes anonymized student identifiers and gender information. Thus, all data were gathered online via the ViLLE platform to analyze students' performance.

5.4. *Data Analysis*

The data analysis focused on understanding student interaction types, time and performance based on descriptive statistics. The findings can inform improvements in interactive task design and multimedia principles.

[1] `https://en.learninganalytics.fi/ville`

6. Findings

This section presents the results of the study, which aimed to assess interactive tasks in relation to computational and algebraic thinking processes, interactivity types, and the application of multimedia design principles. A combination of descriptive and inferential statistical analyses was conducted to examine patterns in student engagement, task performance, and the instructional quality of the tasks through multimedia design.

The dataset comprises interaction data from 18 tasks completed by students in Grades 1 and 2. Key variables analyzed include the number of students per task, time spent on tasks, average scores, number of submissions, and the frequency and type of multimedia principles applied. Additionally, the study explores the potential relationship between multimedia principles and student performance using non-parametric statistical methods due to the non-normal distribution of score data.

As aligned with the research questions, the results are structured into four main areas: (1) descriptive patterns of student engagement and performance, (2) time on task by interactivity type, (3) frequency and distribution of multimedia design principles across tasks, and (4) correlation between the application of these principles and students' average scores. These findings offer insight into how interactive design and multimedia application may influence students' learning behaviors and outcomes in digital environments.

Descriptive and correlation analyses of the data are presented in Table 2.

The dataset is composed of performance and engagement metrics across 18 tasks completed by students. Below is a summary and interpretation of the key variables:

On average, each task was completed by approximately 32 students. The number of students per task varied, ranging from a minimum of 12 to a maximum of 40. This indicates that while most tasks had similar participation levels, a few were completed by significantly fewer or more students, possibly due to differences in group size or task allocation.

A. Time

In terms of time on task, on average, students spent about 172 seconds (around 2.9 minutes) on each task (Table 3). Some tasks took as little as 48 seconds, while others took

Table 2
Descriptive Results for Grades 1–2

	Mean	Min	Max	Std. Dev.
Time on Task	171.8 s	48.4	502.1	112.9
Time Std Dev	142.8	25.5	411.2	110.8
Average Score	5.24	0.00	10.0	3.24
Score Std Dev	3.32	0.00	4.92	1.36
No. of Submissions	1.69	1.00	2.93	0.52
Submission Std Dev	1.04	0.00	2.30	0.65

Table 3
The average time spent on each interactivity type

Task	Time on task	Interactivity
1	97	Drag and drop
2	93	Drag and drop
3	48	Drag and drop
4	90	Drag and drop
5	197	Write text, integer
6	115	Write text, integer
7	198	Click
8	189	Write text, integer
9	79	Click
10	157	Write text, integer
11	113	Click
12	502	Drag and drop
13	131	Click
14	140	Click
15	119	Write text, integer
16	266	Drag and drop
17	182	Click
18	377	Click

more than 8 minutes. This big difference shows that some tasks were much harder or more detailed than others. While a few tasks were quick and easy to finish, others took a lot more time – probably because they were more difficult, had more content, or needed more interaction. The high variation (112.9 seconds) also shows that the time students spent on each task was quite different from one task to another.

B. Interactivity type

Tasks that involved Click on object and Drag and Drop interactions took the longest average time to complete, suggesting these may require more exploration or user engagement. Write-type tasks, where students input text or numbers, took moderately less time, possibly due to more direct response formats.

C. Performance (Score)

Score variability is relatively high, implying that some students performed very well while others struggled, possibly due to differences in multimedia principles. Number of Submissions as an indicator of performance shows that on average, students submitted each task about 1.7 times. The number of submissions ranged from 1 to almost 3 times.

This shows that many students submitted the same task more than once. This might mean they had a chance to revise their work, made mistakes on the first try, or wanted to improve their scores by trying again. In line with this, the standard deviation for the number of submissions was 1.04, which indicates there was a noticeable difference among students in how often they resubmitted tasks. Some students submitted only once, while others submitted multiple times. The variation in submission frequency further supports the likelihood of repeated attempts or iterative learning processes for certain tasks.

Multimedia principles

An analysis of multimedia principle usage across the interactive tasks is shown in Table 4.

The result of the analysis reveals that the *Temporal Contiguity Principle* was the most frequently applied, being present in 100% of the tasks. A number of other principles were also widely implemented. Specifically, the *Redundancy Principle, the Spatial Contiguity Principle, and the Image Principle* were each applied in 94.4% of the tasks, indicating strong alignment with best practices for presenting visual and verbal information effectively.

In terms of moderate usage, both the *Segmenting Principle* and the *Pre-training Principle* appeared in 83.3% of the tasks. These were followed by the *Coherence Principle*, which was used in 77.8% of tasks, and the Signaling Principle, present in 72.2%.

Regarding multimedia principles and the average scores, the data do not meet the requirements for a parametric test. A Shapiro-Wilk test was conducted to assess the normality of the score data. The results indicated a deviation from normality, $W = 0.89$, $p = .037$. Therefore, the assumption of normality was violated, and non-parametric tests were used in subsequent analyses. Since the p-value is less than 0.05, the null hypothesis of the Shapiro-Wilk test was rejected. This means that the score data is not normally distributed.

The correlational analysis was run based on each principle. The result of the analysis is shown in Table 5.

Table 4
Frequencies of Principles within Questions

Principle	f	%
Temporal Contiguity	18	100
Redundancy Principle	17	94,44
Spatial Contiguity Principle	17	94,44
Image principle	17	94,44
Segmenting Principle	15	83,33
Pre-training Principle	15	83,33
Coherence Principle	14	77,78
Signaling Principle	13	72,22

Table 5
Correlation Between Multimedia Principles and Student Scores

Principle	Point-Biserial *r*	*p*-value
Redundancy Principle	0.19	0.44
Coherence Principle	0.59	0.01
Signaling Principle	0.50	0.03
Spatial Contiguity	0.19	0.44
Segmenting Principle	0.42	0.08
Pre-training Principle	0.52	0.02
Image Principle	0.26	0.30

A non-parametric point-biserial correlation, a specialized form of Pearson's correlation for measuring the relationship between a continuous variable (average score) and binary variables (multimedia principles, coded as 0/1) was used to analyze the data. Temporal Contiguity Principle is excluded because all values are constant (all 1s) and this makes correlation undefined. The analysis revealed statistically significant positive correlations for the Coherence Principle ($r = 0.59$, $p = 0.01$), Signalling Principle ($r = 0.50$, $p = 0.03$), and Pre-training Principle ($r = 0.52$, $p = 0.02$), suggesting that the presence of these principles is associated with higher average scores. Other principles, including Redundancy, Spatial Contiguity, and Segmenting, did not show significant relationships with the average score ($p > 0.05$).

7. Conclusion

This study examined the effectiveness of interactive tasks in promoting computational and algebraic thinking skills among primary and lower secondary students, with a focus on interactivity types and the application of multimedia design principles. The analysis, based on data collected from 63 students across multiple grade levels, provided insight into learner performance in terms of average scores, time on task, and number of submissions, as well as the instructional quality of task design.

Descriptive results revealed that the majority of tasks were completed within approximately three minutes, though there was considerable variation in completion time, suggesting differences in task complexity and cognitive demand. Tasks involving Click-on-object and Drag-and-drop interactions took longer to complete, indicating that these formats may encourage exploration and deeper engagement. Conversely, Write-type tasks resulted in shorter engagement time, possibly due to their more straightforward input structure. The number of submissions also indicated active student engagement, with several students revisiting tasks – suggesting that the platform effectively supported iterative learning.

From a multimedia design perspective, the most frequently applied principles were Temporal Contiguity, Spatial Contiguity, Redundancy, and the Image Principle, each appearing in over 94% of tasks. These align well with Mayer's Cognitive Theory of

Multimedia Learning, which emphasizes minimizing extraneous load and promoting meaningful learning through the integrated presentation of verbal and visual content.

The results demonstrate positive correlations for the Coherence, Signaling and Pretraining Principle with higher average scores. These results reinforce prior literature suggesting that effective multimedia design can positively influence learning outcomes when principles are applied deliberately (Dubois & Vial, 2001; Mayer, 2017).

Importantly, the findings also reflect on the alignment of interactivity types and multimedia principles. Tasks that embedded multimedia design more consistently tended to correlate with higher scores and increased engagement, supporting the notion that cognitive load can be strategically managed through careful instructional design (Schnotz & Bannert, 2003).

In sum, this study supports existing research that emphasizes the pedagogical value of interactive, well-structured multimedia tasks in fostering computational and algebraic thinking. These findings can inform educators and designers in developing more adaptive and cognitively balanced learning environments that enhance both engagement and achievement. Future research may consider longitudinal studies and qualitative feedback to explore how learners perceive and respond to these design elements over time.

Acknowledgements

This work has been co-funded through the European Union. Information about the project is on the CT&MathABLE website `https://www.fsf.vu.lt/en/ct-math-able`. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the National Agency. Neither the European Union nor National Agency can be held responsible for them.

References

Applebaum, M. (2025). Fostering creative and critical thinking through math games: A case study of Bachet's game. *European Journal of Science and Mathematics Education*, 13(1), 16–26.

Bang, H.J., Li, L., & Flynn, K. (2023). Efficacy of an adaptive game-based math learning app to support personalized learning and improve early elementary school students' learning. *Early Childhood Education Journal*, 51(4), 717–732.

Blyznyuk, T., & Kachak, T. (2024). Benefits of interactive learning for students' critical thinking skills improvement. *Journal of Vasyl Stefanyk Precarpathian National University*, 11(1), 94–102.

Boaler, J. (2022). *Mathematical Mindsets: Unleashing Students' Potential through Creative Mathematics, Inspiring Messages and Innovative Teaching*. John Wiley & Sons.

Bush, J.B. (2021). Software based intervention with digital manipulatives to support student conceptual understandings of fractions. *British Journal of Educational Technology*, 52(6), 2299–2318.

Cirneanu, A.L., & Moldoveanu, C.E. (2024). Use of digital technology in integrated mathematics education. *Applied System Innovation*, 7(4), 66.

Clements, D.H., Lizcano, R., & Sarama, J. (2023). Research and pedagogies for early math. *Education Sciences*, 13(8), 839.

CT&MathABLE (2025). Computational Thinking and Mathematical Problem Solving, an Analytics Based Learning Environment (2025). https://www.fsf.vu.lt/ct-math-able#about-the-project

Dagienė, V., Stupurienė, G. (2016). Bebras – A sustainable community building model for the concept based learning of informatics and computational thinking. *Informatics in Education*, 15(1), 25–44.

Dubois, M., & Vial, I. (2001). Multimedia design: The effects of relating multimodal information. *Journal of Computer Assisted Learning*, 17(3), 292–302. https://doi.org/10.1046/j.0266-4909.2001.00186.x

Drijvers, P., & Sinclair, N. (2024). The role of digital technologies in mathematics education: purposes and perspectives. *ZDM – Mathematics Education*, 56(2), 239–248.

Hwa, S. P. (2018). Pedagogical change in mathematics learning: Harnessing the power of digital game-based learning. *Journal of Educational Technology & Society*, 21(4), 259–276.

Mayer, R.E., & Moreno, R. (1998). A cognitive theory of multimedia learning: Implications for design principles. *Journal of Educational Psychology*, 91(2), 358–368.

Mayer, R.E. (2009). *Multimedia Learning* (2nd ed.). Cambridge, England: Cambridge University Press.

Mayer, R.E. (2013). Ten research-based principles of multimedia learning. In: *Web-Based Learning* (pp. 371–390). Routledge.

Mayer, R.E. (2017). Using multimedia for e-learning. *Journal of Computer Assisted Learning*, 33(5), 403–423.

Mierluş-Mazilu, I., & Yilmaz, F. (2023, July). Teaching Mathematics in STEM Education. In: *International Conference on Mathematics and its Applications in Science and Engineering* (pp. 147–170). Cham: Springer Nature Switzerland.

Moon, J., & Ke, F. (2020). In-game actions to promote game-based math learning engagement. Journal of Educational Computing Research, 58(4), 863–885.

Laird, J.E., Gluck, K., Anderson, J., Forbus, K., Jenkins, O., Kunda, M., ... & Rosenbloom, P. (2017). Interactive task learning. *IEEE Intelligent Systems*, 32(4), 6–21. https://doi.org/10.1109/MIS.2017.3121573

Liu, Q., Haiyan, C., & Crabbe, M.J.C. (2021). Interactive study of multimedia and virtual technology in art education. *International Journal of Interactive Mobile Technologies*, 16(1), 93–104. https://doi.org/10.3991/ijim.v14i13.14359

Proske, A., Körndle, H., & Narciss, S. (2012). Interactive learning tasks. In: N. M. Seel (Ed.), *Encyclopedia of the Sciences of Learning* (pp. 1606–1610).Springer

Soleimani, M., Ibrahim, R., Zamzami, M., & Shahbodin, F. (2017). CyberPLAYce: A tangible interaction construction kit to enhance young children's computational thinking. *Procedia Computer Science*, 105, 169–174.

Wang, S.Y., Chang, S.C., Hwang, G.J., & Chen, P.Y. (2018). A microworld-based role-playing game development approach to engaging students in interactive, enjoyable, and effective mathematics learning. *Interactive Learning Environments*, 26(3), 411–423.

Ziatdinov, R., & Valles Jr, J.R. (2022). Synthesis of modeling, visualization, and programming in GeoGebra as an effective approach for teaching and learning STEM topics. *Mathematics*, 10(3), 398.

Y. Gulbahar's career is marked by a strong foundation in mathematics and a dedication to computer science education and educational technology. After obtaining her Bachelor's degree in Mathematics from Middle East Technical University (METU) in 1992, she worked as a programmer. Subsequently, she pursued advanced studies in Computer Education and Instructional Technologies (CEIT) at METU, earning both her Master's and Doctoral degrees. Her professional experience includes more than 30 years in public and private universities, where she chaired academic departments as well as administrative units. After retiring from Ankara University, she moved to the USA and is currently working at Teachers College, Columbia University. Her teaching portfolio is extensive, covering topics such as learning analytics, data management, programming, instructional design, and research methods. She is also a prolific author, contributing numerous publications to her field.

T. Öztürk is a scholar of educational technology and digital learning with deep expertise in computational thinking, instructional design, and multimedia learning environments. She earned her Ph.D. from Lancaster University's Centre for Studies in Advanced Learning Technologies (UK), following her graduate studies in Computer and Instructional Technologies Education in Turkey. She is a co-editor of the IGI Global volume Teaching Computational Thinking in Primary Education and has led multiple international projects on computational thinking curricula. She has supervised several graduate theses on the integration of computational thinking into diverse learning contexts and continues to develop instructional approaches that bridge algorithmic logic with creative problem-solving. She runs post-graduate courses on computational thinking.

V. Dagienė is professor at Vilnius University, Lithuania. She has published over 300 scientific papers, and more than 50 textbooks in the field of informatics for schools. She coordinated over 50 national and international projects on CS education. In 2004 she established an International Challenge on Informatics and Computational Thinking BEBRAS which runs in more than 80 countries (`http://bebras.org`). V. Dagienė is editor of two international journals "Informatics in Education" (since 2002, indexed by Web of Science, Q1) and "Olympiads in Informatics" (since 2007, indexed by Scopus). She is acknowledged by Ada Lovelace Computing Excellence Award by the European Commission´s, the Presidential Award, the Cross of the Knight of the Order of the Lithuanian Grand Duke Gediminas, as well as the Distinguish Award of IOI – International Olympiad in Informatics.

M. Parviainen is a doctoral researcher at the University of Turku, where her thesis focuses on computational thinking and mathematical problem solving. She has a strong academic background in computer science and mathematics, with a specialization in education. With over a decade of experience in digital learning environments, she has contributed to all facets of the field – from teacher training and content creation to leading initiatives in both national and international projects.

İ. Güven has B.A. from the Faculty of Educational Sciences, Department of Curriculum and Instrution at Ankara University, Turkey in 1989. He earned his master's degree in 1992 and doctoral degree (Ph.D.) the Social Sciences Institute of Ankara University, Department of Curriculum and Instruction and, has a second doctoral degree (PhD) in History at Ankara University in 1998. He has been working in the Faculty of Educational Sciences in Ankara University since 1990. Dr. Guven has conducted many studies about teaching methods, computational thinking in social sciences, textbook analysis, educational ideologies and policies, social studies and history teaching method courses, citizenship education courses at every level, and in-service education.. He has carried out many applied research projects on computational thinking, innovative teaching methods and approaches. He has served as head of department of Social Studies and History Teaching, and Director of Graduate School of Educational Sciences at Ankara University and Editor of Journal of Faculty of Educational Sciences. Dr. Guven worked as visiting scholar at College of Education at Arkansas Tech University, in USA in 2018/2019. He is working as full Professor at the Department of Turkish and Social Sciences Education in Faculty of Educational Sciences in Ankara University since 2011. Prof. Guven has been working at Teachers College in Columbia University in the USA since 2024.

J. Bilbao is Industrial Engineer (equivalent to Grade and Master) from University of the Basque Country, Spain. Ph.D. in Applied Mathematics, by the same university. Subdirector of the Applied Mathematics Department, University of the Basque Country, Spain; Member of the Evaluation Committee of the Industrial Engineering Graduate for the National Plan for the Evaluation of Quality of the Universities of Spain. (1998–2000); General Chair of different International Conferences; Member of the International Scientific Committee or Scientific Advisory Board of different Conferences; Member of the Reviewer Board of the international journal Applied Energy, of the international journal Mathematical and Computer Modelling, of the international journal International Journal of Renewable Energy Research – IJRER, etc.; President of the Labour Health and Safety Committee of the Bizkaia Campus from 2019. Research Interests: Optimization of series capacitor batteries in electrical lines, Modelization of a leakage flux transformer, Losses in the electric distribution Networks, Artificial Neural Networks, E-learning, Machine Learning, Computational Thinking. With more than 75 articles in international journals, 30 books, 1 patent, 48 projects, more than 200 conference papers, and scientific member of the Bebras Community, and IOTPE Organization.

Olympiads in Informatics, 2025, Vol. 19, 87–100

DOI: 10.15388/ioi.2025.06

Pisek – a Caching Task Preparation System

Martin MAREŠ, Daniel SKÝPALA
Department of Applied Mathematics Faculty of Mathematics and Physics Charles University
Malostranské nám. 25 118 00 Praha 1 Czech Republic
e-mail: mares@kam.mff.cuni.cz, skipy@kam.mff.cuni.cz

Abstract. We introduce a new tool for developing competition tasks. It helps with creating test data and checking that the tests award the expected scores to a set of reference solutions. It supports batch, interactive, and open-data tasks in a variety of programming languages. Test results are cached, which significantly accelerates task development. Automated checks are utilized to detect common errors, including fuzzing of output checkers. The tool interfaces to CMS for configuring tasks, testing them, and semi-automatically establishing time limits.

Keywords: task preparation tool, automated testing.

1. Introduction

Preparing a task for a programming competition is an elaborate process, which includes developing the task statement, creating test data, and checking that the test data award the expected scores to a set of reference solutions. Experience shows that this process is prone to errors, especially when last-minute changes are introduced in a hurry.

Contest organizers therefore strive to make the task preparation process rigorous. One such process was documented by Diks *et al.* (2008) and its principles are still followed by major contests.

An immediate consequence is the development of task preparation systems that try to automate as much of the process as possible. They take a formal description of the task, its tests, and reference solutions. Then they go through all steps of the process and check for errors. Some steps still require human intervention, for example setting of time limits. But even there, the task preparation system can provide guidance.

There already exist multiple task preparation systems, most notably Polygon[1] (popular at CodeForces), TPS[2] (developed for IOI 2017), sinol-make[3] (originated in the Polish OI), and Taskmaker[4] (originated in the Italian OI).

[1] `https://polygon.codeforces.com/`
[2] `https://github.com/ioi/tps`
[3] `https://github.com/sio2project/sinol-make`
[4] `https://github.com/olimpiadi-informatica/task-maker-rust`

In this paper, we present Pisek[5] – a system we have developed over the past few years. It is powerful and fast, while being very simple with minimal dependencies. In particular, it can be easily used by task authors on their own machines. The current version of Pisek is available at `https://github.com/piskoviste/pisek/`.

We aim for supporting a much wider range of contest types and task formats – in particular, both IOI-type contests where solutions are submitted as source code, and open-data contests where the contestants download test inputs and submit the corresponding outputs. We also support a wide variety of programming languages.

Pisek is based on its own task format, which tries to make common things straightforward and less common things possible. Tasks developed in this format can be later exported to an actual contest system.

Pisek has a simple command-line interface, which can be used manually or invoked as a part of a continuous integration system. Pisek employs a lot of caching behind the scenes to make development cycles short while ensuring correctness.

Inside, Pisek is implemented in as a collection of Python modules that can also serve as building blocks of other tools for handling tasks, or even of contest systems.

This paper presents the features of Pisek and the foundations on which it is built. Section 2 introduces the task format and the components of the task development process. Section 3 describes deeper layers, in particular handling of programming languages and the caching layer. Section 4 discusses integration with contest systems like the CMS.[6]

1.1. *History of Pisek*

The first version of Pisek was developed in 2019 by Jiří Beneš, Richard Hladík, Michal Töpfer, and Václav Volhejn for a Czech open-data contest called Kasiopea,[7] drawing inspiration from the KSP open-data system[8] developed by Martin Mareš. Then it was extended to handle IOI-type tasks for the Czech IOI team selection camp.

Between 2023 and 2025, Pisek was rewritten by Jiří Kalvoda, Daniel Skýpala, and Benjamin Swart, based on experience with the initial version and further ideas by Martin Mareš. This version is described in this paper. It is also used to develop tasks for the Czech national programming olympiad and CEOI 2024.

2. Tasks and their Parts

First of all, we introduce the underlying concepts of tasks and their testing. Then we explain how these concepts are expressed in Pisek.

[5] "písek" is a Czech word for sand, alluding to a playground for children.
[6] `https://github.com/cms-dev/cms`
[7] `https://kasiopea.matfyz.cz/`
[8] `https://ksp.mff.cuni.cz/`

2.1. *Anatomy of a Task*

Pisek supports two types of tasks: *batch tasks* (the solution is a single program that reads an input and then produces the corresponding output) and *interactive tasks* (a program that interacts with the contest system in multiple steps; e.g., a two-player game). By default, all communication is performed via the standard input and output, but the task can define a library that wraps such communication in an arbitrary API provided to the solution.

The goal of the task is specified in a *task statement* given to contestants. Statements are not handled by Pisek.

Solutions are graded using a set of *tests*, each having one or more *testcases*. Each test is worth a certain amount of points, which are awarded for solving all testcases in the test. For IOI-style tasks, tests correspond to subtasks. Sample input and output (given openly to the contestant) is also considered a separate test.

In a batch task, a testcase specifies an input to the solution and the correct output. The input can be a static file, but it is usually created using a *generator*. The correct output can be static, but it is often computed from the input using a correct *primary solution*. A *checker* then decides if the solution's output matches the correct output. It can be a diff-like program, or if there are multiple correct outputs, the task can provide a *judge program* for checking correctness. The judge may also award partial score (e.g., in optimization tasks), the total score per test is then computed as the minimum over all testcases.

In an interactive task, there is always a judge program, which interacts with the solution over a pair of pipes. There is also an input file, but it is consumed by the judge. Again, the judge may award partial score.

A task also comes with several *reference solutions* with expected scores. One of the solution is declared *primary*. A primary solution is expected to solve all testcases correctly and efficiently.

In addition to solutions, a task can define a *validator*. It is a separate program that meticulously verifies that the input files conform to the format set in the task statement. In some cases, validation is integrated in the primary solution instead.

Generators, judges, and validators can have access to a *dataset* – a collection of data files that are either contained in the task package or generated by a separate program.

2.2. *Task Package*

Pisek represents everything related to a single task as a *task package*. The package is stored as a single directory in the file system (possibly with subdirectories). The contents are typically maintained in a Git repository, but Pisek is oblivious to versioning.

Behavior of the task is controlled by a *configuration file* with a simple INI-like syntax (essentially a collection of key-value pairs divided to sections) – see Fig. 1 for an

```
[task]
version=v3           # Config version, v3 is the newest one.
use=ceoi2024         # Use contest-specific default settings.

[tests]
in_gen=gen           # Program used to generate inputs.
validator=validate   # Program used to validate inputs.
out_check=tokens     # Compare outputs using token-based checker.

[solution_model]
primary=yes          # This sol. used for generating correct outputs.
tests=1111           # It should succeed on all tests.

[solution_greedy]
tests=W1WW           # Succeed only on test01, otherwise wrong answer.

[solution_slow]
tests=111T           # Timeout on test03 and succeed otherwise.

[solution_segfault]
tests=X!!!           # Any result on samples, runtime error on the rest.

[test01]
points=2             # This test is for two points.
in_globs=01*.in sample1.in   # Inputs assigned to this test.

[test02]
points=3
in_globs=02*.in sample2.in

[test03]
points=5
in_globs=03*.in
predecessors=1 2     # This test is a superset of both test01 and test02.

[run_solution]
time_limit=1         # Time limit for solutions.

[cms]                # Data for CMS importer.
name=sandcastle
title=Building sandcastles
time_limit=1
```

Fig. 1. An example configuration file.

example. The configuration can refer to a parent configuration file that supplies defaults for non-specified items. Typically, the parent configuration is specific to a contest. The ultimate parent is the set of defaults provided by Pisek itself.

The task package also contains a collection of static input and output files and source code of all programs related to the task (generators, validators, judges, reference solutions etc.).

Finally, there may be extra files not handled by Pisek. This typically includes the task statement.

2.3. *Generators*

In addition to static testcases, task authors can implement a *generator* that produces further testcases in a mechanic way. Pisek supports multiple generator interfaces, but all of them follow the same logic:

- **The generator is deterministic** – the generated input file depends only on the generator itself, its runtime arguments, and possibly on the dataset. If the generator uses pseudo-random numbers, it should fix their *random seed* to one provided in the runtime arguments. This is crucial for reproducibility of testing and Pisek's caching.
- **The generator respects the seed** – for different seeds, the generator should generate different input files. This is especially useful in open-data contests where each attempt to solve the task produces new input data based on a fresh seed, which expires after some time. It is also possible to declare that a particular test does not have a seed.

The mapping of tests to testcases depends on the particular generator interface. In the trivial case, each test has a single testcase named after the test.

With the more advanced interfaces, the generator can be asked to produce a list of testcases it can generate. Each testcase has a file name (e.g., `easy01.in`) and optional attributes: if it is seeded and how many instances of the testcase (with different seeds) should be generated.

The configuration file can then specify a list of filename globs for each test, e.g., `in_globs=01*.in easy*.in`. All testcases (static and generated) matching any of the globs are included in the test.

Moreover, a test can also define one or more *predecessor tests*, whose testcases are automatically included. For example, the contest-specific configuration can specify that the predecessor should be the previous test. Transitively, this makes each test to include its own testcases and testcases of all previous tests.

2.4. *Checkers*

A batch task needs a *checker* to decide if the solution's output is correct. Pisek provides a variety of built-in checkers that compare the solution's output with the correct output at different levels of strictness:

- **Diff** – runs the `diff` utility provided by the operating system, set to ignore differences in whitespace. This is a traditional method, but it suffers from quadratic time complexity in the worst case.
- **Tokens** – compares the two outputs as sequences of whitespace-separated tokens. By default, newline characters are considered separate tokens, but the task can choose to make them equivalent to other whitespace. Additionally, the checker can be configured as case-insensitive and/or to compare numeric tokens with a given precision. This checker is the recommended choice if the correct output is unique up to formatting.
- **Shuffle** – a token-based checker that accepts all permutations of tokens within a line, or all permutations of lines within a file, or both. It is useful if the correct output is unique up to order.

If there are multiple correct outputs (e.g., multiple shortest paths in a graph), the task provides a custom *judge*. Pisek supports several interfaces to judges, including the one used in CMS.

Depending on the interface, the judge can be given the test number, the seed used to generate the input, the input, the correct output (as produced by the primary solution), and the solution's output. The input and the correct output are optional – some judges do not need them, as they can compute everything from the seed. This is useful if Pisek is used within an open-data contest system, which can skip generating the unneeded files and save time.

The main part of the judge's output is the *verdict* (accept or reject, possibly with a message for the contestant). Optionally, the judge can award points (absolute or relative to the number of points per test).

Interactive tasks always require a custom judge, which talks to the solution over a pair of pipes. Pisek currently supports only the manager interface of communication tasks in CMS. The judge gets the input and produces a verdict as with the batch judges.

In the future, we plan to design a more flexible interactive judge interface, because the CMS interface suffers from multiple problems. In particular, use of named pipes leads to deadlocks if they are opened in an unexpected order. Furthermore, it is not possible to report wrong answers differently from protocol errors, which leads to confusing results if the protocol error is caused by a pipe being closed automatically after the solution crashes. This is in need of more research and hopefully also cooperation among maintainers of contest systems.

2.5. *Solutions*

The task specifies a *primary solution* and an arbitrary number of *secondary solutions*. The primary solution should be correct and efficient; it is used to produce the correct output if the checker needs it. The set of secondary solutions usually includes other correct solutions (to ensure that the primary solution is correct) and also incorrect solutions with a wide range of mistakes (to ensure that the scoring strategy works as expected).

Solutions communicate over their standard input and output, although this can be wrapped in a library (see below). Solutions typically have their running time and memory limited.

For each solution, the task configuration specifies the expected outcome. It can be the expected number of points or the expected outcome for each test (e.g., test 1 passes, test 2 produces a wrong answer, test 3 times out). The expected outcomes are preferred, but expected points can be more useful in optimization tasks.

2.6. *Verification*

There are many possible mistakes in competition tasks, but they frequently follow one of a few typical patterns. Pisek provides a battery of checks for such common errors. All of them are optional, defaults are usually provided by the per-contest configuration.

- **Size of inputs and outputs** – sizes are compared with a configured maximum. This can catch a run-away generator. In open-data contests, the limits are usually more strict, because the contestants must be able to download the input, run their program, and upload its output within a short time window.
- **Coverage of tests by solutions** – for each test, there should be a reference solution that succeeds on this test and all its predecessors, but fails on all other tests. This is useful if subtasks of the task are linearly ordered (each is a strict superset of the preceding one) or if their dependencies form a rooted tree.
- **Unused inputs** – every input (static or produced by the generator) should be included in at least one test.
- **Last test uses all inputs** – if the subtasks are linearly ordered, the last test should include all inputs.
- **Generator depends on seed** – the generator produces different input files for the same testcase with different seeds. This can produce false positives in tasks with short inputs, but our experience shows that it is rare in practice.
- **Fuzzing** – if the task has a custom judge, this check tries to run it on many randomly mutated copies of the sample outputs. This often crashes judges with sloppy parsing of the solution's output.

Additionally, a validator supplied with the task is ran on each testcase. Its goal is to check conformance of the input to the task statement. The validator is also given the test where the testcase belongs, so it can verify properties required by specific subtasks.

2.7. *Preprocessors*

Input and output of most tasks is a simple ASCII text. But the simplicity is often deceiving: text files can contain trailing spaces at the end of a line, multiple spaces in a row, or tabulators instead of spaces. Lines can be terminated by different newline characters,

the final newline can be missing, or perhaps there are a few extra empty lines at the very end. Windows programs tend to add the UTF-8 byte-order marker at the beginning of text files, even if the text contains only ASCII characters. Sometimes, they also encode the ASCII text in UTF-16.

Some of these problems are unknown in the C++-centric world of major competitions. But once a competition enables more exotic programming languages, or if the tasks are open-data, all of them become everyday issues.

Handling all these anomalies in checkers and judges is a tedious task prone to errors. Pisek avoids problems with irregular whitespace by using token-based checkers (and we provide a tokenization library to custom judges). To handle the other problems, Pisek runs all text files through a preprocessor that normalizes character encoding and newline characters (including proper termination of the last line).

Preprocessing takes place in three situations:

- **All inputs (both static and generated)** – the inputs are normalized first. If an input contains non-ASCII or non-printable characters, normalization fails and so does testing of the task. If normalization changes the input, depending on the configuration either the normalized input is used instead, or an error is raised.
- **Outputs produced by solutions** – they are normalized before they are checked for correctness. A warning can be also produced if the output was non-normalized. Failed normalization causes the testcase to fail.
- **Outputs produced by contestants** – if Pisek is used as a part of an open-data contest system, outputs uploaded by contestants are also normalized.

Tasks with non-ASCII input/output can set their input/output format to binary and check correctness using a judge. New formats can be added easily. An obvious candidate is Unicode text in UTF-8 or UTF-16, but that would bring a completely new set of normalization issues (see Whistler (2024)).

Preprocessing does not take place for interactive tasks. Their judges must cope with non-normalized text.

3. Building and Running Programs

Development of a task involves running different programs: generators, validators, judges, and reference solutions. They are written in varying programming languages. First, it is good practice to test solutions in all languages available to the contestants, so that time limits can be calibrated accordingly. Second, task authors often prefer to use higher-level languages (e.g., Python) for generators and validators, which need not run quickly.

Let us consider typical use cases first:

- **Simple C++** – The task package contains one source file. We need to run a compiler, which produces an executable file. In some cases, the task author wants to add custom compiler options or to link a well-known library. Most traditional compiled languages also fall into this category.

- **Simple Python** – The task package contains one source file. We can run it directly. This also applies to languages like Perl, Ruby, Raku, and JavaScript.
- **Simple Java** – The task package contains one source file. We need to run a compiler, which produces byte code. To run it, we need to invoke the Java virtual machine. Alternatively, we can set up `binfmt_misc` on Linux to make the kernel recognize the byte code signature and run the JVM automatically. We prefer to avoid this approach, because it needs root privileges and we cannot adjust JVM options per task. A similar case is C#.
- **Multi-file C++ or Java** – Like Simple C++, but we have multiple source files which have to be compiled and linked together to produce a single binary.
- **Multi-file Python** – We have multiple source files, but no compiler. All files have to be present when running the program. An alternative is to use the little-known `zipapp` module from Python's standard library that can pack all files to a single ZIP archive which is then runnable by the Python interpreter. We still need a generic solution for other Python-like languages.
- **Rust with Cargo** – Rust programs are usually built using Cargo from a directory with all source files and a configuration of Cargo. A similar case is Go with its module system.
- **Make** – In rare cases, there is a program with a complex building process. It can be a multi-language program, or perhaps a program whose source code is generated by another program. As we do not want to implement yet another universal build system, we prefer to defer to an existing build system in such cases. For sake of tradition, let us consider Make. The source code is then a directory with a `Makefile`.
- **Task-specific stubs and libraries** – At some contests (e.g., recent IOIs), solutions are expected to implement an API instead of communicating using files. The solution is then linked with a *stub*: a piece of code specific for a combination of a task and a language that serves as the interface between the contest system and the API. Usually, the stub reads the input from the standard input, calls the solution's API, and writes the result to the standard output. Similarly, an interactive task can provide an API called by the solution to interact with the judge.
- **Multi-purpose binaries** – Sometimes, we want to share code among generators, judges, and validators. A single source file can participate in compilation of multiple binaries. Or we can produce a single binary which can play multiple roles, depending on the command-line arguments passed.

Overall, we want to handle the simple cases (e.g., a single C++/Python source file) with as little configuration as possible, while still allowing the complex cases.

This is accomplished by two parts of configuration: *build sections* that describe how programs are compiled from their sources, and *run sections* that specify how the programs should be run. All settings in these sections have defaults such that in the typical case, you can omit the sections completely and just specify the name of the program.

3.1. *Building Programs*

A build takes the source and produces an executable program. The *source* is either a single file or a sub-directory. The *executable program* is either a single file executable by the OS or a sub-directory containing an executable file called `run` that can refer to the rest of the sub-directory (relative to the path it was ran from).

The build is governed by a *build section* in the task configuration. The section is named after the combination of a program name and its purpose, e.g., `[build_solution:good1]`. It specifies the name of the source and a *build strategy* to be used. Available strategies include:

- **A simple C++ program** – compiles a single source file to a single executable file.
- **A simple Python program** – just copies the source file and marks it as executable.
- **A simple Java program** – compiles a single source file to a byte code file, produces a directory, were `run` is a shell script that runs the JVM on the byte code.
- **A multi-file Python program** – takes a directory and a given entry point, produces a directory with `run` symlinked to the file with the entry point.
- **Cargo** – takes a directory and runs Cargo in it to produce a single file.
- **Make** – takes a directory, runs `make` in it; the `Makefile` is supposed to produce output in a sub-directory called `target`, which contains either a single executable file or a collection of files with an executable `run`.

If neither the source nor the strategy is given, Pisek chooses automatically. Most strategies have an auto-detection rule. For example, if we are building the solution `good1` and the task package contains a file `good1.cpp`, the C++ strategy is willing to build it. If multiple strategies match, an error is raised and the user must make an explicit choice. So in the simple cases, the whole build section can be omitted.

Additionally, the build section can set strategy-specific options like compiler options, further files to be made available to the compiler (e.g., header files) and additional source files to be compiled together with the main source file (e.g., task stubs). This is useful in conjunction with inheritance of build section: `[build_solution:good1]` inherits from `[build_solution]` (e.g., task-specific libraries) and `[build]` (e.g., compiler flags provided by contest-specific configuration).

3.2. *Running Programs*

Whenever task configuration specifies a program to be run (e.g., a solution), it actually refers to a *run section* named after the program and its purpose. For example, `[run_solution:good1]`. The run section refers to a build section that produces the program and it specifies the command-line arguments to be passed and resource limits to be applied (e.g., a time and memory limit).

Again, there are defaults that allow omitting the whole section: we build `[build_solution:good1]` and run the program with no arguments. There is an inheritance

hierarchy of `[run_solution]` and `[run]` that typically provides resource limits. For solutions in particular, we also inherit from `[run_primary_solution]` and `[run_secondary_solution]`, which is often used to run secondary solutions with a less strict time limit.

3.3. *Sandboxing*

Programs should be run within a sandbox that imposes resource limits and checks that the programs access only the expected files (this is important to ensure consistent caching).

Pisek currently uses `minibox`, a simple pseudo-sandbox which limits memory using the kernel's `ulimit` for virtual memory and which kills the program when the time limit is exceeded. It is not a proper sandbox as it is easy to escape from it. But it is actually sufficient in most cases as the programs in the task package can be trusted not to be malicious. (However, beware when using somebody else's task packages.)

The advantage of this approach is simplicity and no need for root privileges. Disadvantages include problems with limiting memory in C# and Go (both runtimes allocate enormous amounts virtual address space without actually using it) and the impossibility of controlling programs with multiple processes or threads.

In the future, we plan to switch to Isolate (Blackham and Mareš, 2012)) and/or `systemd-run` (weaker, but available in most Linux distributions by default).

3.4. *Caching*

Testing a task in Pisek can be a time-consuming process. We need to generate all input files, validate them, run all verification checks, run all solutions, and check their output. All this can easily take at least minutes for an IOI-level task. On the other hand, it is good practice to re-test the task after every change, especially in the later stages of contest preparation.

We observe that minor changes in the task often affect only a small subset of Pisek's operations. We can therefore save significant time by caching results of operations and re-computing them only if the relevant parts of the task change.

This is similar to what build systems like `make` do, but they need the user to declare explicit dependencies, which is prone to errors. We prefer a systematic and automated approach that is as close to obviously correct as possible.

Testing of tasks is divided to small pieces called *jobs*. Each job can depend on results of other jobs, called its *prerequisites*. There is a universal mechanism for caching job results. Each *cache entry* contains the following information:

- *Name* – a human-readable description of the job (e.g., "Run solution *name* on input *name*").
- *Result* – the output of the job (e.g., if running the solution succeeded).

- *Signature* – a cryptographic hash of all data on which the job depends. This includes:
  - `__init__` *arguments* – each job is internally a class, whose initialization parameters specify what the job should do (with more details than what is specified in the job's name).
  - *Results of prerequisites.*
  - *Testing context* – values of command-line arguments and all settings in task configuration which have been accessed when the job was run.
  - *Contents of files* – for each file read or created by the job, we record the hash of its contents, which is then added to the collective signature.
  - *Evaluation of globs* – if the job uses filename globbing (e.g., to select testcases for a given test), we need to check that the glob still produces the same set of files. Otherwise, dependencies on file contents would not catch a newly matched file.
- *Signature recipe* – a list of all inputs from which the signature was computed.

The cache can contain multiple entries with the same job name, but different signatures. (This is why the signature covers contents of files produced by the job: Different versions of the job may have the same output file with different contents.)

When Pisek wants to run a job, it looks up all entries with the right name in the cache. For each such entry, it computes the signature according to the entry's recipe. If it matches the entry's signature, the job is considered unchanged and the cached result is re-used. If the job needs recomputing, a new entry is created with the same name and a new signature. If there are too many entries with the same name, we trim the oldest ones.

The jobs are fine-grained, which enables Pisek to recompute only the absolute minimum when the task changes. For example, when we change the judge, we do not re-run the solutions and we only re-judge their outputs. When a new testcase is added, solutions are run only on that testcase etc.

This systematic approach has proven itself efficient and reliable. Over the years we used Pisek, there were very few errors, usually caused by colliding job names or file names. The cache is automatically invalidated when Pisek is upgraded to avoid compatibility errors. (However, this does not apply when using the development version of Pisek from its Git repository as the version number changes only for official releases.)

4. Integration with Contest Systems

When the task is tested in Pisek, we need to export it to the actual contest system. The export should be automated to the greatest extent possible to avoid human errors.

The environment in which the solutions run within the contest system is obviously different from the environment used by task authors. So we need to verify that the behavior of tasks in the contest system matches the expectations.

4.1. *Integration with CMS*

We have implemented an export to CMS, which can set up the task, create a dataset with the test data, set time and memory limits, submit all reference solutions, download test results, and compare them to the expected results.

We also have a semi-automatic tool for choosing the time limit. This requires detailed specification of the expected behavior of the reference solutions on tests. In particular, we need to separate timeouts from the other failure modes. Then we can compute the time interval between the slowest solution that should not time out and the fastest one that should time out. The time limit is then chosen manually from the computed interval. If the interval is empty, we must improve the test data.

Since Pisek supports a much wider variety of tasks, there are some restrictions. The task must use judge interfaces compatible with CMS and it cannot rely on the text preprocessor.

Currently, the CMS lacks a public API for creating tasks and submitting solutions. Our CMS interface therefore relies on CMS internals and calls CMS libraries in ways that can break in the future. We will try to keep up with changes in CMS, but the proper solution is to make CMS offer a well-defined API.

4.2. *KSP Open-data System*

With the KSP open-data contest system, we plan a completely different approach. We are going to re-implement the back-end of the contest system on the top of Pisek. Most of the necessary functionality is already available in Pisek as separate modules: most importantly generating the input data for a given seed and testing if the output is correct.

The only significant difference is that we have to separate the actions performed online during the contest (generating inputs and checking outputs) from those that take place when setting up the task (compiling programs, preparing datasets).

5. Conclusions

Pisek has proven itself useful when developing tasks for multiple contests including CEOI 2024.

Still, there remain several areas which call for further research and development. Most importantly, we would like to extend compatibility between Pisek and CMS: support the full range of Pisek's built-in checkers, judge interfaces, and possibly also the text preprocessor. One possibility is to improve CMS itself, another is auto-generating manager code for CMS.

The task format could be brought closer to the Kattis problem package specification.[9] Both formats would benefit from cross-pollination and automated conversion of tasks between them.

A task statement (and its translations) could be added to the task package format, which would enable automatic inclusion of sample inputs and outputs. Formalizing descriptions of subtasks (at least partially) could enable sharing a single definition of limits among the task statements, the validator, and possibly also the generator. The task-maker already supports similar features.

Further automated checks for common errors should be included, especially a more powerful fuzzer.

Judges and validators of different tasks contain a lot of common code, often implemented with insufficient handling of malformed inputs. We suggest that this common code should be generalized and made available as a library. The library should be independent of the task preparation system used.

References

Blackham, B., Mareš, M. (2012). A New Contest Sandbox. *Olympiads in Informatics*, 6, 100–109.
Diks, K. *et al.* (2008). A Proposal for a Task Preparation Process. *Olympiads in Informatics*, 2, 64–74.
Whistler, K. (ed.) (2024). Unicode Standard Annex #15: Unicode Normalization Forms. Available online at: https://unicode.org/reports/tr15/

M. Mareš is a lector at the Department of Applied Mathematics of Faculty of Mathematics and Physics of the Charles University in Prague, organizer of several Czech programming contests, member of the IOI Technical Committee, and a Linux hacker.

D. Skýpala is a Bachelor student at Faculty of Mathematics and Physics of the Charles University in Prague, IOI 2022 bronze medalist, organizer of several Czech programming contests and a member of CEOI 2024 Scientific Committee.

[9] https://github.com/Kattis/problem-package-format

Olympiads in Informatics, 2025, Vol. 19, 101–114

DOI: 10.15388/ioi.2025.07

Girls in STEM: A Qualitative Analysis of Factors and Actors Impacting on Girls' Engagement in International Computer Science Competitions

Laura MARRONE BERZETTI di BURONZO[1], Noemi GAMBIRASIO[2]
[1]*Università di Modena e Reggio Emilia, Libera Università Internazionale degli Studi Sociali Guido Carli, National Ph.D. Program in Learning Sciences and Digital Technologies, Italy*
[2]*Università Statale degli Studi di Milano, Department of Informatics, Italy*
e-mail: lmarrone@luiss.it, noemi.gambirasio@gmail.com

Abstract. The study aims to explore the key-factors and influential actors impacting on girls' participation in international Computer Science competitions and on their overall aspirations in STEM career-paths. By analyzing societal perceptions, self-efficacy, and other social and subjective factors through qualitative analysis, the research wants to uncover both motivators and barriers that shape young women's interest in scientific fields. The findings will contribute to raise awareness of the social factors fostering the Gender Gap in Computer Science, providing useful insights to shape more inclusive and supportive learning environments for young women in competitive programming.

Keywords: STEM literacy, gender equality, inclusive learning, computer science.

1. Introduction

Gender equality in STEM (Science, Technology, Engineering, and Mathematics) education and careers remains a pressing challenge despite recent global initiatives aimed at fostering inclusivity and diversity. In fact, women continue to be significantly underrepresented in STEM disciplines – particularly in "hard science" fields such as engineering, computer science, and physics (Fry *et al.*, 2021; NCSE, 2021) – facing social barriers that not only limit their potential talent pool and innovation capacity (Cowgill *et al.*, 2021; Van Camp *et al.*, 2019), but also prevent them from fully participating in and benefiting from economic opportunities offered by these crucial industries (Carnevale *et al.*, 2011). The underrepresentation of women in STEM-related education not only represents an issue of social justice, it has indeed strong economic and developmental consequences in today's modern world. Present and future global challenges require in fact a multidisciplinary culture that combines humanistic knowledge with widespread STEM literacy. In the Digital era, STEM fields are essential for economic growth, technologi-

cal innovation, and overall global competitiveness: the Fourth Industrial Revolution has amplified the need for STEM-educated professionals, making gender inclusivity in STEM a critical issue for workforce development (Elazab *et al.*, 2019). In this perspective, investments in techno-scientific education – fostering interdisciplinary learning, preparing students for dynamic and evolving career paths (Bybee, 2013) – contribute to national economic stability, promoting job creation, enhancing problem-solving skills, and driving innovation (Marginson *et al.*, 2013). Countries that invest heavily in STEM education and workforce development experience therefore significant advancements in infrastructure, sustainability, and industrial growth (Xie *et al.*, 2015).

Despite extensive research on gender disparities, a significant gap remains in understanding the specific factors that shape girls' participation in STEM competitive environments. While general barriers to inclusion have been widely studied, little attention has been given to the complex interplay between societal perceptions, self-efficacy, and the influence of key figures such as educators, mentors, and family members. Additionally, while biases and institutional obstacles are acknowledged, there is a lack of empirical data on how these challenges interact with individual psychological factors. Gaining deeper insights into these dynamics is crucial for designing targeted interventions that not only encourage more girls to enter competitive STEM environments but also support their long-term engagement and success in these fields.

This study therefore seeks to explore the underlying social and psychological mechanisms that influence young women's engagement with Computer Science competitions and their broader aspirations in STEM careers. By investigating the roles of societal expectations, self-efficacy, and key influencers such as teachers, mentors, and family, this research aims to uncover both barriers and motivators that shape girls' participation in these competitive environments. The study will provide a nuanced understanding of the complex interactions between individual aspirations and external influences, offering evidence-based insights to support the development of more inclusive STEM learning environments. Through this approach, the research will contribute to face gender disparities in STEM by answering the following main research questions:

- What psychological and socio-cultural factors influence girls' participation and persistence in Computer Science competitions and STEM fields in general?
- How can educational and institutional support improve gender inclusivity in Computer Science competitions and STEM fields in general?

By addressing these questions, this study seeks to contribute to the broader discourse on gender equality in STEM and provide actionable insights for improving diversity in these critical fields. In a world where the mastery of scientific tools is in fact a full-citizenship requirement, it is in fact necessary to make STEM literacy more inclusive, adopting a perspective of educational equity: this means supporting the many potential students held back by gaps in educational opportunities (Lee and Buxton, 2010) by addressing the obstacles to academic access – primarily linked to unfavorable social backgrounds (OECD, 2012) – and the biases associated with them, which are often internalized since childhood.

The findings will be of interest to educators, policymakers, and researchers aiming to develop strategies for improving inclusivity in STEM fields: addressing gender dispari-

ties in STEM will help institutions implement more equitable policies and foster a more inclusive educational ecosystem. By understanding the challenges faced by women in STEM, organizations can create targeted interventions to enhance female participation and retention in these high-demand fields.

2. Related Work

The underrepresentation of women in STEM is the result of a complex interplay of personal, social, and institutional factors.

At an individual level, Self-Efficacy, Motivation, and Confidence play significant roles in determining whether girls pursue STEM fields (Rosenzweig & Wigfield, 2016; van den Hurk *et al.*, 2019; Prieto-Rodriguez *et al.*, 2020): in fact, research suggests that when girls receive encouragement from teachers, parents, and mentors, they are more likely to develop a strong STEM identity and persist in these fields (Prieto-Rodriguez *et al.*, 2020). A sense of belonging has also been identified as a critical factor in women's persistence in STEM education and careers. Students who feel socially and academically integrated into their academic communities are more likely to persist in their chosen fields (Strayhorn, 2018; Murphy *et al.*, 2020). Support networks, mentorship programs, and inclusive institutional cultures play a key role in fostering this sense of belonging for women in STEM.

At a socio-cultural level of examination, the major challenge is represented by the continued presence of gender stereotypes that depict STEM professionals as predominantly male (Thébaud & Charles, 2018; Sáinz *et al.*, 2019). Gender disparities in STEM can be traced back to early childhood and adolescence: as a matter of fact, girls' interest in STEM subjects declines between the ages of 12 and 15 (Tsan *et al.*, 2016), largely due to social and cultural factors, including deeply rooted stereotypes that associate technical and mathematical abilities with men (Fouad & Santana, 2017; Lent & Brown, 2019). In fact, studies indicate that societal beliefs about gender roles contribute to the perception that women are less competent in STEM disciplines: the fact that male students are often perceived as more competent in STEM subjects can affect young girls' confidence and engagement in STEM activities (UNESCO, 2018; Sáinz, 2020), discouraging them from pursuing future careers in these fields (Leaper & Brown, 2014; Sáinz *et al.*, 2020). As a consequence, addressing these disparities requires interventions at various educational levels, starting in primary and secondary education (Wang & Degol, 2017). In particular, creating inclusive classroom environments and implementing gender-sensitive teaching strategies can help foster early interest in STEM careers among young girls (UNESCO, 2018).

Regarding the institutional level, gender disparities have been addressed through different STEM Intervention Programs ("SIPs") implemented to engage young women and provide them with the resources and support needed to succeed (Liben & Coyle, 2014; Rosenzweig & Wigfield, 2016; van den Hurk *et al.*, 2019). Providing institutional resources and inclusive learning environments can in fact help to mitigate the systemic challenges that disproportionately affect women in STEM (Cooper *et al.*, 2019).

3. The Current Study: Insights from Girls participating in International Computer Science Competitions

Despite Computer science being one of the fastest-growing fields globally, with increasing demand for skilled professionals (Xie *et al.*, 2015), the underrepresentation of women in this field remains a critical issue in STEM education and career pathways: lower participation and retention rates are attributable to strong gender disparities, structural barriers that preclude women's technical roles, especially in competitive and high-performance computing environments (Cheryan *et al.*, 2017).

Despite numerous initiatives aimed at fostering inclusivity, there remains a gap in understanding the personal journeys of young women in Computer Science, particularly those engaged in competitive programming. Competitive programming serves as an essential gateway to high-level technical skills, problem-solving expertise, and career opportunities in software engineering, artificial intelligence, and cybersecurity (Li *et al.*, 2019). Given its importance, examining the experiences of young women in this domain provides critical insights into how to better support female participation and persistence in the field.

While quantitative studies highlight disparities in participation and retention, qualitative research provides deeper insights into the lived experiences of young women in this field. Understanding these personal narratives is crucial, as they offer a nuanced perspective on the motivations, challenges, and structural barriers faced by young women entering competitive programming and computer science.

This study therefore employs a qualitative methodology through semi-structured interviews with female participants in international Programming competitions. By exploring their educational background, support systems, role models, challenges, and perspectives on gender diversity, the research aims to shed light on the factors that influence young women's engagement in competitive programming. The findings contribute to the ongoing discourse on gender diversity in STEM and inform future strategies to create more inclusive and equitable learning environments in computer science.

3.1. *Methodology*

To explore the factors influencing girls' engagement in international Computer Science competitions, the study employed a qualitative methodology based on semi-structured interviews. The research involved twelve female participants, aged 18 to 23, from diverse geographic backgrounds, including Sweden, Lithuania, Germany, Luxembourg, Italy, Algeria, and Switzerland. These participants were selected based on their involvement in STEM education and participation in the European Girls' Olympiad in Informatics (EGOI) 2024, held in Veldhoven, the Netherlands.

Data collection focused on participants' educational backgrounds, support systems, role models, challenges, and perspectives on gender diversity in STEM. The interviews,

conducted in English, were transcribed verbatim and analyzed using thematic analysis. This approach enabled the identification of recurring patterns, categorized according to three levels of analysis: individual, socio-cultural, and institutional.

- Individual Level: regarding self-efficacy, motivation, personal aspirations, and the role of intrinsic interest in STEM.
- Socio-cultural Level: regarding family support, peer influence, and societal perceptions about gender roles in STEM.
- Institutional Level: regarding the impact of school curricula, teacher encouragement, mentorship programs, and broader policy initiatives on girls' participation in STEM.

3.2. *Results*

The Individual Level

Participants described diverse motivation drivers, identifying several entry points into Computer Science. Exposure to Mathematics played a foundational role for some participants, as problem-solving skills translated naturally into an interest in coding. Informal learning, such as self-directed study and online courses, also emerged as a crucial factor, particularly in cases where formal education did not provide sufficient programming instruction. In general, self-motivation emerged as a key factor in sustaining interest and overcoming challenges in male-dominated environments.

> *"I have always been good in math, initially I actually used to participate in math competitions; it was from there that I have been introduced to computer science"* (Serena, 18, Italy)

> *"I was fascinated by smartphones and how they work, and from there I started exploring coding"* (Vaiva, 23, Lithuania)

However, some participants struggled with sustaining long-term interest in Computer Science, particularly when motivation drivers – such as competition results – declined. When faced with these challenges, they tended to develop personal strategies to navigate biases and self-doubt: many emphasized resilience, self-motivation, and seeking supportive environments.

> *"I think the biggest challenge is my own self-esteem, sometimes I still feel like this is not really the world I belong in… I try to remind myself that I'm capable."* (Priska, 21, Switzerland)

> *"I faced discrimination from some lecturers, but I made sure to choose those who supported me."* (Vaiva, 23, Lithuania)

> *"I overcame self-doubt by focusing on my achievements and surrounding myself with supportive people."* (Noemi, 20, Italy)

In fact, participants' future aspirations often included working in the STEM field and improving its gender inclusivity.

> *"I want to explore both IT and design in my future career."* (Vaiva, 23, Lithuania)

> *"I want to create a strong IT community in Algeria for future generations."* (Maya, 20, Algeria)

> *"I hope to continue supporting young girls in Olympiads and competitions."* (Noemi, 20, Italy)

The Socio-cultural Level

In the family context, support varied across participants. Some reported highly encouraging environments, with parents who actively promoted STEM engagement by enrolling their children in STEM-related programs (many reported that they have been introduced to programming through family members who worked in related fields), while others described a more neutral stance, with parents providing only passive encouragement and approval by allowing them to explore their interests independently.

> *"My father is a PhD in Physics and now he is working in IT: that probably had some influence. And my mother is a doctor, so she also has some STEM background. I guess mathematical skills were always high valued"* (Priska, 21, Switzerland)

> *"My mom signed me up for course about programming and I got obsessed with it"* (Isabel, 19, Germany)

> *"My family has been fundamental. Programming competitions involve a lot of stress and also self-esteem levels are not regular: there are peaks of it after victories, but of course you also experience defeats… having someone who still sees you as capable, even when you think you cannot do it anymore, is key"* (Noemi, 20, Italy)

> *"My father has a degree in economics, he doesn't even know what computer science is; my mother too, she studied accounting... But in the end I think they are quite happy that I've found my way"* (Serena, 18, Italy)

In some cases, friends who were already active in the field also played an important role in introducing participants to the programming world.

> *"I saw a friend taking an online coding course, and I decided to try it too. That's how I got started."* (Zelma, 18, Germany)

> *"A girl in my school helped me enter the math-advanced group, and that made all the difference."* (Matilde, 18, Italy)

> *"I used to go to math camps, where I met friends who weren't really interested in math, they were there just because it was the closest thing to programming, and they introduced me to it… so yeah, it was because of my friends"* (Maya, 20, Algeria)

Opinions on overall gender diversity in STEM varied among participants, even though a recurring theme was an overall gender imbalance in STEM contexts, with many participants describing male-dominated environments (particularly in competitive programming). Social isolation was in fact a recurring theme, with many participants reporting being among the few girls in their programming classes or competitions: they felt socially excluded or found it difficult to integrate into male-dominated spaces, struggling to find a community of like-minded peers, particularly within their local environments.

> *"When I went to math competitions, there were almost only guys. I felt out of place."* (Anita, 22, Sweden)

> *"It was struggling to find a community of other people that lived in Luxembourg that I knew that were also interested in IT"* (Laura, 18, Luxembourg)

While some of them did not experience explicit discrimination, stating that their gender did not significantly impact their experiences, others faced actual gender biases from teachers, peers, or competition organizers, acknowledging subtle differences in treatment and representation. For instance, some of them described situations where especially male peers questioned their competence despite superior performance in competitions. The European Girls' Olympiad in Informatics itself was reported to be perceived as less prestigious than mixed-gender programming contests, reinforcing gendered perceptions of ability.

> *"A teacher literally told me: 'This is not for you.'"* (Maya, 20, Algeria)

> *Some lecturers didn't like girls in STEM, they treated us differently."* (Vaiva, 23, Lithuania)

> *"I was told several times that I was there (i.e. at female programming competitions) only because I am woman... there is still a lot of discrimination"* (Serena, 18, Italy)

Despite these challenges, many participants emphasized that gender biases appeared to diminish at more advanced levels of expertise and within more specialized communities, where peers and mentors were more open-minded: over time, those who engaged in external competitions or international programs were in fact able to establish a network of supportive peers, reinforcing the importance of community-building initiatives.

Institutional Level

Institutional support for girls in competitive programming varied significantly, as schools were found to play an inconsistent role in fostering engagement. As a matter of fact, some participants encountered supportive educators and had access to competitions and preparatory resources, whereas others faced a lack of of institutional encouragement and had to rely only on external initiatives.

> *My school supported me by organizing competitions, but most teachers didn't really know about coding."* (Anita, 22, Sweden)

> *"My teacher was amazing. She encouraged me to participate in Olympiads and supported me throughout."* (Priska, 21, Switzerland)

> *"Some teachers were great, but others were dismissive. I had to choose carefully who to work with"* (Vaiva, 23, Lithuania)

Mentorship programs were also identified as a crucial element in fostering participation, with several participants emphasizing the positive impact of female mentors and role models. In particular, several participants noted that growing up mentors or visible figures in the field would have helped them to maintain intrinsic motivation.

> *"If I had a role model, I would have been more motivated."* (Anita, 22, Sweden)

> *"One of my mentors created a space for women in Olympiads, which really helped."* (Priska, 21, Switzerland)

Synthetizing Findings

The study highlights key factors influencing young women's experiences in competitive Computer Science. While some participants reported positive experiences, challenges such as gender biases, social isolation, and limited access to specialized training persist. To make STEM subjects more gender-inclusive in the future, three focal needs emerged:

- Early exposure to STEM subjects: family support, school-based initiatives and competitions can spark interest and build confidence in young girls.
- Supportive environments: creating inclusive learning spaces and combating gender biases can improve retention and engagement.
- Mentorship and community-building: expanding mentorship programs and fostering networks for women in STEM can provide essential guidance and support.

Addressing these areas can contribute to a more equitable and supportive landscape for women in STEM fields, ensuring that more young girls feel empowered to pursue these careers.

4. Noemi's Testimony

My name is Noemi Gambirasio, and I am a woman who works, studies, and lives within the field of a STEM discipline: computer science. I consider myself lucky because, in my country, Italy, no one has ever actively prevented me from pursuing my passions. Don't get me wrong, there are still people here who believe that certain jobs or fields of study are meant for men and others for women, but I am happy to say that in Italy, they are just an annoying minority. The real issue here is all those social concepts, prejudices, or beliefs that still today make it seem like a woman in my field is an exception or something unusual.

To be honest, if someone asked me, "Why don't most women choose the same path as you?" I wouldn't be able to give a definite answer. Many in Italy would respond that it's simply a field that tends to interest men more, and that could be a factor, after all, there are no laws or rules here that prevent a woman from choosing to work or study in these areas. However, throughout the journey that led me to choose computer science not only as a job but also as a passion, I have experienced situations that make me believe that this is not the only reason for the low female participation in STEM disciplines.

I started my journey thanks to a school competition in competitive programming. To be honest, my choice wasn't entirely conscious, I signed up because the school notice described it as a "logical-mathematical problem-solving contest," and I had always enjoyed those kinds of challenges. It was only during the competition that I encountered an algorithm written in pseudocode for the first time. Needless to say, I didn't pass the competition that year, but those problems definitely sparked my interest. So, I began studying C and C++ and got into competitive programming. The following year, I passed the school-level round, then the regional round, reached the national competition, and from there, I was selected to attend training camps for the selection and preparation for international competitive programming contests. That was where, for the first time, I found myself in a predominantly male environment focused on a STEM discipline.

I won't deny it, the first impact was fantastic. I was surrounded by people my age who shared my passion. I could express a part of myself that I wasn't used to sharing with classmates or friends. I didn't want to lose that feeling. I wanted to be part of the group. Unfortunately, the first problems arose precisely because of this, and the prejudices I mentioned earlier weren't imposed on me by others, I imposed them on myself. In my mind, simply finding people with whom I could share a part of myself wasn't enough. I convinced myself that to truly belong, I had to eliminate everything about me that didn't interest them, which meant I decided it shouldn't interest me either. The unfortunate part is that this included many of my feminine traits, which I mistakenly began to classify as weaknesses or, worse, as a waste of time.

The point is that when people say STEM fields have a male-dominated environment, they are not only referring to the fact that there are more men than women. It also means that common interests, priorities, humor, and pastimes statistically align more

often with those of men than with those of women. In my case, this led me to emphasize certain aspects of myself over others, but for different people, whether women or men, it could lead to distancing themselves from the environment and, consequently, from the field itself.

And so, we arrive at the core of what we must fight against in countries like mine. Over time, and with the support of those around me, I realized what I was unconsciously doing. With effort, and, I won't lie, with some pain, I regained balance within myself. The truth, however, is that this struggle is much more likely to be faced by a woman than by a man in STEM disciplines. The goal, therefore, is this: that a woman should not have to choose, whether consciously or not, between fighting against herself and others or abandoning her passions. There is still a long way to go, but thanks to experiences like EGOI, I have met women who were either going through or had already overcome the same inner conflict as mine, and nothing makes you believe you can succeed like seeing someone else who already has.

5. Conclusions

Despite progress in increasing gender diversity in STEM, significant challenges remain that hinder the full participation of women in these fields. Societal stereotypes, institutional barriers, and a lack of representation still contribute to the underrepresentation of women in STEM careers. These disparities are not only detrimental to gender equity but also limit economic growth, technological advancement, and innovation. Addressing these challenges requires a multi-faceted approach that targets the root causes of gender disparity at multiple levels: individual, institutional, and societal.

One key strategy for bridging the gender gap in STEM is the implementation of targeted intervention programs that address both structural and cultural barriers. Research has shown that mentorship programs, inclusive curricula, and initiatives that foster a sense of belonging significantly improve women's participation and retention in STEM fields. Early exposure to STEM education, particularly through hands-on and experiential learning opportunities, can shape young girls' interest and confidence in pursuing these disciplines. Additionally, increasing the visibility of female role models in STEM plays a crucial role in countering stereotypes and inspiring future generations of women to enter and thrive in STEM careers.

Institutional reforms are equally vital in creating inclusive academic and professional environments. Universities and workplaces must actively work to eliminate biases in hiring, promotion, and academic evaluation processes. Creating supportive networks, implementing gender-sensitive policies, and promoting leadership opportunities for women in STEM can lead to greater retention and career advancement. Furthermore, addressing implicit biases in STEM education through teacher training and curriculum design is critical to fostering equitable learning environments.

From a broader societal perspective, cultural and policy changes are necessary to challenge deep-rooted stereotypes that associate STEM fields with masculinity. Media

representation of women in STEM should be enhanced to showcase diverse role models who break conventional gender norms. Public policies that support work-life balance, parental leave, and flexible career pathways can also help retain women in STEM professions, allowing them to sustain long-term careers while managing personal and family responsibilities.

Ultimately, gender equity in STEM is not just a women's issue but a societal imperative that requires collective action from educators, policymakers, industry leaders, and communities. Achieving true gender parity will require sustained efforts to dismantle existing barriers and create environments where women are empowered to succeed in STEM. By fostering a culture of inclusivity, innovation, and equal opportunity, we can ensure that STEM fields benefit from the full diversity of talent and perspectives necessary to address the complex challenges of the future.

The findings of the present study underscore the importance of early exposure, supportive educational environments, and the presence of role models in fostering female participation in STEM. While some participants reported positive experiences, challenges such as gender biases, social isolation, and limited access to specialized training persist. These results align with previous research indicating that women in computer science often face unique structural barriers, including implicit biases, lack of role models, and limited access to peer communities (Cheryan *et al.*, 2017; Wang & Degol, 2017). Moreover, the findings suggest that participation in competitive programming plays a crucial role in skill development and career opportunities, reinforcing the importance of integrating such activities into formal education settings (Li *et al.*, 2019). However, disparities in school support and motivation indicate the need for more targeted interventions to ensure inclusivity in these competitive environments.

Future research may explore a broader sample of participants across different educational and cultural contexts to generalize findings and develop more comprehensive strategies for fostering diversity in computer science. As Xie *et al.* (2015) suggest, strengthening STEM education and reducing gender disparities are crucial for technological advancement and innovation. By addressing structural and cultural barriers, we can work towards a more equitable and inclusive STEM landscape for future generations.

References

Astin, A. (1984). Student involvement: A developmental theory for higher education. *Journal of College Student Development,* 25(4), 297–308.

Berger, J. B., Braxton, J. M. (1998). Revising Tinto's interactionalist theory of student departure through theory elaboration: Examining the role of organizational attributes in persistence. *Research in Higher Education,* 39(2), 103–119.

Blickenstaff, J. C. (2005). Women and science careers: Leaky pipeline or gender filter? *Gender and Education,* 17(4), 369–386.

Bottia, M. C., Stearns, E., Mickelson, R. A., Moller, S., Valentino, L. (2021). Growing the roots of STEM majors: Female math and science high school faculty and the participation of students in STEM. *Economics of Education Review,* 79, 102066.

Braxton, J. M., Sullivan, A. S., Johnson, R. M. (1997). Appraising Tinto's theory of college student departure. *Higher Education: Handbook of Theory and Research,* 12, 107–164.

Bybee, R. W. (2013). *The case for STEM education: Challenges and opportunities.* NSTA Press.

Carnevale, A. P., Smith, N., Melton, M. (2011). STEM: Science, technology, engineering, mathematics. *Georgetown University Center on Education and the Workforce.*

Cheryan, S., Ziegler, S. A., Montoya, A. K., Jiang, L. (2017). Why are some STEM fields more gender-balanced than others? *Psychological Bulletin,* 143(1), 1–35.

Cooper, K. M., Krieg, A., Brownell, S. E. (2019). Who perceives they are smarter? Exploring the role of gender and self-efficacy in STEM. *International Journal of STEM Education,* 6(1), 1–14.

Cowgill, B. O., Dell'Acqua, F., Deng, S., Deng, S. (2021). The gender gap in STEM fields: Evidence from a quasi-experiment. *Labour Economics,* 71, 102027.

Diekman, A. B., Eagly, A. H. (2008). Of men, women, and motivation: A role congruity account. *Advances in Experimental Social Psychology,* 40, 1–59.

Dou, R., Hazari, Z., Dabney, K., Sonnert, G., Sadler, P. (2019). Early informal STEM experiences and STEM identity: The importance of talking science. *Science Education,* 103(3), 623–637.

Eccles, J. S. (1994). Understanding women's educational and occupational choices. *Psychology of Women Quarterly,* 18(4), 585–609.

Eccles, J. S. (2009). Who am I and what am I going to do with my life? Personal and collective identities as motivators of action. *Educational Psychologist,* 44(2), 78–89.

Eccles, J. S. (2005). Subjective task value and the Eccles *et al.* model of achievement-related choices. *Handbook of Competence and Motivation*, 105–121.

Eccles, J. S., Wigfield, A. (1995). In the mind of the actor: The structure of adolescents' achievement task values and expectancy-related beliefs. *Personality and Social Psychology Bulletin,* 21(3), 215–225.

Elazab, A., Al-Azab, M., Utsumi, T. (2019). The role of STEM education in the fourth industrial revolution: A sustainable approach. *Journal of Sustainable Development,* 12(2), 113–124.

Estrada, M., Woodcock, A., Hernandez, P. R., Schultz, P. W. (2011). Toward a model of social influence that explains minority student integration into the scientific community. *Journal of Educational Psychology,* 103(1), 206–222.

Feder, M. A., Malcom, S. M. (2016). *Barriers and opportunities for 2-year and 4-year STEM degrees: Systemic change to support diverse student pathways.* National Academies Press.

Fouad, N. A., Santana, M. C. (2017). SCCT and underrepresented populations in STEM fields: Moving the needle. *Journal of Career Assessment,* 25(1), 24–39.

Fry, R., Kennedy, B., Funk, C. (2021). STEM jobs see uneven progress in increasing gender, racial, and ethnic diversity. *Pew Research Center*.

Funk, C., Parker, K. (2018). Diversity in the STEM workforce varies widely across jobs. *Pew Research Center*.

Good, C., Rattan, A., Dweck, C. S. (2012). Why do women opt out? Sense of belonging and women's representation in mathematics. *Journal of Personality and Social Psychology,* 102(4), 700–717.

Giles, M. (2015). The chilly climate for women in STEM: Examining gender bias in science, technology, engineering, and mathematics. *Gender & Society,* 29(5), 722–743.

Hall, R. M., Sandler, B. R. (1982). *The classroom climate: A chilly one for women?* Project on the Status and Education of Women, Association of American Colleges.

Hurtado, S., Carter, D. F. (1997). Effects of college transition and perceptions of the campus racial climate on Latino college students' sense of belonging. *Sociology of Education,* 70(4), 324–345.

Leaper, C., Brown, C. S. (2014). Perceived experiences with sexism among adolescent girls. *Child Development,* 85(3), 778–795.

Lee, O., Buxton, C. (2010). Diversity and Equity in Science Education: Research, Policy and practice. *International Journal of Multicultural Education*, 13(1).

Lee, J. J., McCabe, J. (2020). Gendered expectations in STEM education: A review of literature. *Review of Educational Research,* 90(3), 401–436.

Li, G., Zhang, J., Tang, X. (2019). Competitive programming and its role in computer science education. *Journal of Computer Science Education,* 27(2), 123–140.

Liben, L. S., Coyle, E. F. (2014). Developmental interventions to address the STEM gender gap: Exploring intended and unintended consequences. *Advances in Child Development and Behavior,* 47, 77–115.

Marginson, S., Tytler, R., Freeman, B., Roberts, K. (2013). *STEM: Country comparisons.* Australian Council for Educational Research.

Marx, D. M., Stapel, D. A., Muller, D. (2005). We can do it: The interplay of construal orientation and social comparisons under threat. *Journal of Personality and Social Psychology,* 88(3), 432–446.

Master, A., Cheryan, S., Meltzoff, A. N. (2016). Gender stereotypes about interests start early and cause gender disparities in STEM fields. *Proceedings of the National Academy of Sciences,* 113(30), 8046–8051.

McGee, E. O. (2020). Interrogating structural racism in STEM higher education. *Educational Researcher,* 49(9), 633–644.

Muro, M., Liu, S., Whiton, J., Kulkarni, S. (2018). *Digitalization and the American workforce*. Brookings Institution.

National Council for Special Education (2021), Annual Report.

Nugent, G., Barker, B., Grandgenett, N., Adamchuk, V. (2015). Impact of robotics and geospatial technology interventions on youth STEM learning and attitudes. *Journal of Research on Technology in Education,* 47(3), 173–191.

OECD (2012). Equity and Quality in Education: Supporting Disadvantaged Students and Schools", OECD Publishing.

OECD (2015). PISA In Focus, n. 49.

Olson, S., Riordan, D. G. (2012). *Engage to excel: Producing one million additional college graduates with degrees in STEM.* Report to the President, Executive Office of the President.

Sadler, P. M., Sonnert, G., Hazari, Z., Tai, R. (2012). Stability and volatility of STEM career interest in high school: A gender study. *Science Education,* 96(3), 411–427.

Sáinz, M., Eccles, J. S., Wigfield, A. (2019). Understanding gender differences in STEM. *Journal of Research in Science Teaching,* 56(6), 654–678.

Sáinz M., Eccles, J. S. (2012). "Self-concept of computer and math ability: Gender implications across time and within ICT studies", *Journal of Vocational Behavior*.

Shin, J. E. L., Levy, S. R., London, B. (2016). Effects of role model exposure on STEM and non-STEM student engagement. *Journal of Applied Social Psychology,* 46(7), 410–427.

Strayhorn, T. L. (2018). College students' sense of belonging: A key to educational success for all students. Routledge.

Tsan, J., Boyer, K. E., Lynch, C. (2016). The impact of stereotype threat on women's STEM interest and identity. *Journal of Computer Science Education,* 26(4), 305–320.

UNESCO. (2018). Cracking the code: Girls' and women's education in science, technology, engineering and mathematics (STEM). United Nations Educational, Scientific and Cultural Organization.

Vogt, C. M. (2008). Faculty as a critical juncture in student retention and performance in engineering programs. *Journal of Engineering Education,* 97(1), 27–36.

Wang, M. T., Degol, J. L. (2017). Gender gap in STEM: Revisiting the role of school context. *Educational Psychology,* 52(4), 222–241.

Williams, W. M., Ceci, S. J. (2012). When scientists choose motherhood: A single factor goes a long way in explaining the dearth of women in math-intensive fields. *American Scientist,* 100(2), 138–145.

Xie, Y., Fang, M., Shauman, K. (2015). STEM education. *Annual Review of Sociology,* 41(1), 331–357.

L. Marrone Berzetti di Buronzo is a Marketing graduate specialized in Analytics and Metrics, and currently a Ph.D. candidate in Learning Sciences and Digital Technologies at Università di Modena e Reggio Emilia. She is deeply interested in building educational community engagement through creativity and inclusive communication, with a strong research focus on Behavioral Economics and Educational Equity. Accordingly, her diverse work experience spans from teaching Digital Economy Marketing and being a Research and Teaching Assistant in Consumer Behavior, to working as an Academic Orientation Adviser and collaborating on several projects related to social and educational inclusion.

N. Gambirasio is a Software Engineer intern specializing in Front-End Android Development at Bending Spoons. She is also an Informatics student at Università degli Studi di Milano and an active collaborator with the Olimpiadi Italiane di Informatica, where she contributes to competition task creation and national internships. With a strong background in algorithmic problem-solving, she has earned multiple medals at the Italian Olympiad in Informatics. She competed in the European Girls' Olympiad in Informatics (EGOI) in 2022 and 2023, earning medals, and was the team leader for the Italian delegation in 2024.

Olympiads in Informatics, 2025, Vol. 19, 115–127

DOI: 10.15388/ioi.2025.08

Olympiad Tasks in Changing Environment

Pavel S. PANKOV[1], Elena S. BUROVA[2], Elzat J. BAYALIEVA[3]
[1]*Institute of Mathematics, Kyrgyzstan*
[2]*AUCA, Kyrgyzstan*
[3]*J.Balasagyn Kyrgyz National University, Kyrgyzstan*
e-mail: pps5050@mail.ru, burova_e@auca.kg, elzat.bayalieva@gmail.com

Abstract. Traditionally, tasks in informatics are formulated within fixed (discrete) environment such as segments (1..N), rectangles (1..N × 1..M), or graphs containing various objects, obstacles, connections, and goals. However, in practice environment often change dynamically, and the information available is incomplete, which makes standard algorithmic solutions inadequate. Likewise, a living agent perceives only nearby elements of its environment. This paper explores a class of tasks that address these limitations. Some of these tasks were generated with the assistance of AI. The concepts of timexels (introduced by the authors in their prior work) and program time are used as foundational tools to describe changing discrete environments.

Keywords: Olympiad, informatics, task design, dynamic environment, motion, timexel, program time.

1. Introduction

In real-world applications, actions often occur with incomplete knowledge of a changing environment. While many problems in informatics focus on optimization and planning paths or motions for one or more agents, few explicitly account for environmental uncertainty and change. The aim of this paper is to introduce theoretical constructs, propose definitions, develop methodology, and create and solve illustrative problems reflecting such conditions.

Remark. There exist classical tasks involving visibility, such as *Seeing the Boundary* (IOI'2003 (along polygons)) and *Hermes* (IOI'2004 (along streets)), which are based on visibility along straight lines. However, these problems do not take distances into account, nor do they use visibility data for active decision-making.

Additionally, several well-known mathematical problems can be reinterpreted under the lens of dynamic environments or limited visibility (discussed in Section 3).

To incorporate a changing environment formally, we have introduced the concept of timexels (Pankov *et al.*, 2021).

General Task 1 (Formal Setup):

Let USt be a set of states St, with initial state Pt0 and desired final subset Ft1 $\subset$ USt. At each step, the state transitions – either deterministically or non-deterministically – according to a computable function W: USt $\rightarrow$ subset of USt. You are given partial or full information I(St) about the current state St. Your allowed action is to transition the system via a computable function Y: U $\rightarrow$ subset of USt. Your objective is to bring Pt0 to some Ft1 in the minimum number of steps.

The program must either:

- reach Ft1,
- output the minimum number of steps to Ft1, or
- compete against a specified (or jury-defined) algorithm W to validate its optimality.

Attempts were made to use AI to generate Olympiad-style tasks involving "to write a program in changing environment" but initial queries were too general. A more focused prompt, "write a program to catch a moving goal," yielded satisfactory tasks (see Section 4).

Remark. Informatics often presents dualities (Pankov, 1990). In virtual motion, one may interpret motion in two ways: either the space moves toward the observer, or the observer moves through space. Similarly, tasks may be described using permissions or prohibitions – passes or obstacles. We use the more convenient formulation.

Section 2 introduces *program time*, applies *timexels*, and proposes a formal definition of what it means to find an optimal computer program under dynamic conditions.

Section 3 defines tasks based on well-known mathematical problems as well as a *limited visibility* domain.

Section 4 contains new tasks on catching moving objects, including under conditions of limited visibility, with some of them generated by AI systems. Section 5 contains dual tasks involving temporal objects: one requires jumping onto objects only, while the other requires avoiding contact with them.

Notation:

- Z_0 := {…, -2, -1, 0, 1, 2, …};
- N_0 := {0, 1, 2, …};
- N_1 := {1, 2, 3, …};
- Directions: E, NE, N, NW, W, SW, S, SE (counterclockwise).

2. Program Time and Timexels

To represent dynamic processes, we extend the Eulerian perspective from continuum mechanics to discrete settings. We define space primitives as follows:

- 1D: dots
- 2D: pixels
- 3D: voxels

- Any similar objects could be called as “spacexels”.

By extending space primitive with a time dimension, we introduced “timexels” (space elements existing during one time step). Each timexel is indexed by its spatial coordinates, temporal index (a natural number indicating a moment of time-step), and optional attributes.

The concept of timexcel offers a framework for the approximate description of various processes.

Example (models red and green points converging into a black one, followed by rapid disappearance):

Represented as a set of 1D-timexcels (unordered):

(2, 0, red), (3, 1, red), (4, 2, black), (6, 0, green), (5, 1, green), (6, 3, black), (8, 4, black)

Same elements presented as spacexels with optional color attribute, ordered in a time dimension:

- time=0: (2, red), (6, green)
- time=1: (3, red), (5, green)
- time=2: (4, black)
- time=3: (6, black)
- time=4: (8, black)

We can identify that such time-indexed representation is limited in flexibility. It cannot, for example, model the dynamic placement of obstacles by an external agent (e.g. jury’s program for evaluation).

Definition 1: Program Time (p_time)

To formalize timexel-based programs, we define discrete *program time* p_time as follows:

- At the beginning, p_time := 0
- The program must include a clock procedure (subprogram or function) named Clo (or multiple procedures, such as Clo1, Clo2, etc.). Each execution of this procedure performs the operation p_time := p_time + 1
- All other operations in a program are instantaneous.

Example Task 2: Sorting Three Numbers (elements)

Define a Boolean function Clo(X, Y) = true if X < Y, false otherwise.

Comparisons of the elements in the program are allowed only via function Clo. Sample program code:

Program 2-1:

```
p_time := 0;
Input(U, V, W);
if (not Clo(U, V)) then Swap(U, V);
if (not Clo(V, W)) then Swap(V, W);
if (not Clo(U, V)) then Swap(U, V);
```

```
Output(U, V, W);
Output(p_time);
end.
```

After executing this algorithm, we will get result with p_time = 3.

Task 3: Optimal sorting problem

Using timexel model, the classical sorting problem becomes formalized in a following way.
Let:

- PP: a permutation of M elements
- P[M]: set of all permutations of M elements
- QQ: a program permutating PP with Clo function
- Q[M]: the (infinite) set of programs sorting via Clo function

Defining a function FF: $Q[M] \times P[M] \to N_1$ with the following conditions:

- If the program QQ with the initial data PP is completed with p_time < M^3 steps, then FF(QQ, PP) := p_time
- otherwise FF(QQ, PP) := M^3.

The goal is:

$$F(M) \coloneqq \min_{QQ \in Q[M]} \left\{ \max_{PP \in P[M]} F\,(QQ, PP) \right\} \tag{1}$$

Remark. Some authors refer to Swap as Clo2 (in our terminology), but distinguishing latent swaps within program code is challenging – particularly when assignment operators are disallowed.

Method 1. For small values of M, such problems are addressed as follows (according to our terminology):

First, it is formally proven that the inequality $F(M) < F_1$ is not possible. Subsequently, a program that executes in exactly F_1 steps is constructed.

In the case of Problem (1), the inequality $2^{F(M)} < M!$ does not hold.

This notation and methodological framework can be consistently applied to a broad range of problem types.

General Task 4: Adaptive Coin Weighing Problem

The study of such task was initiated by H. Steinhaus (H.Steinhaus, 1989, 1999). The task is defined as follows:

Given M indistinguishable coins, either (A) exactly one or (B) at most one of them is counterfeit. Additionally, it is known that (C) the counterfeit coin is either heavier or lighter than the genuine ones, or (D) the counterfeit coin differs in weight, but it is not known in which direction.

The objective is to determine the minimum number of weighings required to: (G1) identify the counterfeit coin, or (G2) in case (D), determine both the identity of the coun-

terfeit and whether it is heavier or lighter. Additional genuine coins, whose authenticity is known in advance, may be used in the process.

To formalize this task, the *Clo-weight function* is employed. This function accepts an even number of arguments, $2k$, representing coins. It outputs one of three possible relations: the first k coins are lighter (L), equal in weight (E), or heavier (H) than the second k coins. The relation H is defined as the inverse of L, i.e., H = −L.

For case (C), it has been shown that the inequality $3^{F(M)} < M$ does not hold. Similarly, in case (D), the inequality $3^{F(M)} < 2M$ is invalid.

Significant contributions to this problem were made by M. Kołodziejczyk (n. d.), who obtained the following results: in case (A)(C), it was shown that F(19) = 3; and in case (B)(D), assuming an unlimited number of additional genuine coins, F(40) = 4.

Task 5: Coin Swapping Scenario

As a novel contribution, we introduce Task 5, which, to the best of our knowledge, has not been previously studied. The task involves three coins, denoted *U*, *V*, and *W*, with the condition (A)(D): at most one coin is counterfeit, and it is known only that it differs in weight (heavier or lighter) from the genuine coins. After the first weighing, two of the coins are swapped. The objective (G2) is to identify the original counterfeit coin.

The following decision procedure is proposed:

Algorithm 5-1.

1. Initialize p_time := 0.
2. Perform the first weighing: Z1 := Clo-weight(U, V).
3. If Z1 = ‘E’, then conclude that coin *W* is counterfeit. Terminate. Set p_time := 1.
4. Otherwise (note: *U*, *V*, *W* refer to positions, not fixed coins), proceed:
    - Swap two coins and perform a second weighing: Z2 := Clo-weight(U, V).
    - If Z2 = Z1 (indicating that the counterfeit coin remained in place, and a genuine coin was swapped, making the new *W* genuine), then:
        - Perform a third weighing: Z3 := Clo-weight(U, W).
        - Analyze the outcome and conclude. Terminate. Set p_time := 3.
    - If Z2 = 'E' (indicating that the counterfeit coin – originally *U* or *V* – has moved to position *W*), then:
        - Perform a third weighing: Z3 := Clo-weight(U, V).
        - Analyze the outcome and conclude. Terminate. Set p_time := 3.

This algorithm demonstrates an efficient strategy to detect the original counterfeit coin under dynamic conditions involving positional changes after the initial measurement.

3. Tasks with a Horizon

We consider a class of problems involving partial information and spatial exploration, where the environment is incrementally revealed to the agent. This includes classical mathematical and algorithmic formulations as well as generalizations suited for practical robotics or agent navigation.

Task 6: Cantor's Spiral and Obstacle Detection

This task is inspired by Cantor's diagonal argument demonstrating the countability of pairs of natural numbers. Let the environment be defined as $\mathrm{Z}_{\geq 0} \times \mathrm{Z}_{\geq 0}$. The agent (denoted as A) begins at the origin (0, 0) and may attempt to move in one of the four cardinal directions: East (E), North (N), West (W), or South (S). Movement is governed by a Boolean function Clo(Z), where $\mathrm{Z} \in \{\mathrm{E}, \mathrm{N}, \mathrm{W}, \mathrm{S}\}$. If the move in direction Z is possible, then the agent is shifted accordingly and Clo(Z) = true; otherwise, the agent remains in place and Clo(Z) = false. An obstacle is present in the environment.

Objective of the task is to design an algorithm to detect the location of the obstacle.

Algorithm 6-1 (Spiral Search).

The agent performs a spiral motion originating from the starting point, thereby systematically exploring the grid and detecting obstacles via failed moves.

Remark. This formulation also implies the countability of rational numbers since every rational number can be represented as a pair of integers.

Task 7: General Graph Search with Local Visibility

Traditional formulations of labyrinth or maze-solving problems assume full knowledge of the environment, including coordinates and obstacle locations. In contrast, we present the problem in a setting that better reflects real-world constraints, where only local information is accessible to the agent.

The environment is an unknown connected undirected graph. The agent starts at an arbitrary vertex and may place markers on visited vertices. The following operations are available:

- Clo1(): Returns the number of adjacent vertices and indicates whether the goal vertex is among them.
- Clo2: Indicates whether a marker is present on the current vertex.
- Go(k): Moves the agent to the kth adjacent vertex (in a list returned by Clo1()).
- Goback(): Returns the agent to the previous vertex.

Design an algorithm to find the goal vertex.

General Task 8: Goal Search with Obstacles and Partial Visibility

We generalize the previous tasks to a scenario involving spatial navigation with a distant goal and partially observable obstacles.

The agent starts at an unknown location. The goal is far away and may or may not be directly visible. Some obstacles are present but can only be detected when in close proximity. Devise a strategy to reach the goal. This task can be formalized as follows:

Task 9

Let the environment be a 2D grid defined over 1..M × 1..M, with the origin in the southwest corner. The agent starts at position A(x_0, y_0), and the goal is located at Ft1(M, M). There is a single obstacle B_1. It is guaranteed that A, $Ft1$, and B_1 are distinct. The agent

may use the function Clo(Z) to attempt a move in direction $Z \in \{E, N, W, S\}$. The function behaves as in Task 6.

Objective: Determine the minimal program time p_time necessary to reach $Ft1$. Let the function F(M, x_0, y_0) denote the worst-case minimal steps required to reach the goal. Then:

$$F(M, x_0, y_0) = \min_{QQ \in Q[M]} \left\{ \max_{B_1} F\,(QQ, x_0, y_0, B_1) \right\} \tag{2}$$

Example. For M = 100, with initial position A(98, 100) and unknown obstacle at (99, 100):

- Clo(E) = false → agent remains at (98, 100)
- Clo(S) = true → (98, 99)
- Clo(E) = true → (99, 99)
- Clo(N) = true → (99, 100)
- Clo(E) = true → (100, 100)

Result: p_time = 5

Remark. In general, for most points (except those adjacent to the goal), the expression $F(M, x_0, y_0) = (M - x_0) + (M - y_0) + 3$ holds. However, calculating the exact value of $F(M)$ is difficult even for small values (e.g., $M = 3$) due to the infeasibility of brute-force methods over the infinite space of potential programs.

Algorithm 9-1: **Dynamic Programming Approach**

We present a backward dynamic programming approach based on the Manhattan distance.

Let:

Dist(x, y) = ($M - x$) + ($M - y$)

Initialize:

- F(99, 100) = 1
- F(100, 99) = 1
- F(98, 100) = 5
- F(99, 99) = F(100, 98) = 5

Then for increasing values of Dist(x, y), update:

$F(x, y)$ = max($F(x + 1, y) + 1$, Dist(x, y) + 3)

(e.g., for x = 97, y = 100)

Task 10: Generalization to Multiple Obstacles

Introduce B_1, B_2, …, B_K as obstacles. The prior guarantee that A, $Ft1$, and obstacles are distinct is no longer sufficient. It must additionally be guaranteed that a valid path from A to $Ft1$ exists.

Remark. Even for $K = 2$ obstacles, the task becomes computationally difficult. A direct extension of Algorithm 9-1 is not possible since multiple blocked directions may occur simultaneously (e.g., Clo(E) = false and Clo(S) = false).

Task 11: Search with Limited Field of View

Let the environment be $\mathbb{Z}_{\geq 0} \times \mathbb{Z}_{\geq 0}$, and the agent starts at (0, 0). The agent can step in cardinal directions and sees only a 2 × 2 square centered at its current position (simulating a weak flashlight in a dark environment). A rectangle with even-length sides is placed somewhere in the environment. The task is to reach the center of the rectangle.

Algorithm 11-1.

1. **Stage I**: Perform spiral motion until the rectangle is detected.
2. **Stage II**: Traverse two adjacent sides of the rectangle while counting steps.
3. **Stage III**: Compute half-lengths of the sides.
4. **Stage IV**: Move to the center of one side, then to the center of the rectangle.

4. Tasks Involving Pursuit of a Moving Adversary

In this section, we present a series of algorithmic problems focused on capturing or immobilizing a moving target (referred to as a "virus" or "target") within a constrained environment. The initial formulations were proposed to AI systems (Gemini and *chat.deepseek.com*), and their responses have been unified and adapted into rigorous task definitions.

Task 12: The Elusive Target (Inspired by Gemini)

You are tasked with writing a program to control a virtual "catcher" that must intercept a moving "target" within a specified arena.

Environment: A toroidal grid shaped arena with dimensions $K \times L$, where coordinates wrap around at the boundaries (i.e., movement beyond one edge reappears at the opposite edge).

Initial Conditions:

- The **target** starts at position (x_t, y_t) and moves at a constant velocity vector (dx_t, dy_t).
- The **catcher** starts at position (x_c, y_c) and may move one step per time unit in any of the eight cardinal or diagonal directions.

Input (per time step):

- Grid dimensions: K, L
- Target's position: (x_t, y_t)
- Target's velocity: (dx_t, dy_t)
- Catcher's position: (x_c, y_c)

Output:

- One of the nine possible catcher moves: {N, NE, E, SE, S, SW, W, NW, STAY}

Goal: The catcher's goal is to reach the same grid cell as the target.

Constraints:

- The target may randomly change its velocity at unspecified intervals.
- The algorithm must operate within specified time and memory limits.

Scoring:

- The number of time steps taken to catch the target (lower is better).
- The algorithm is evaluated over a set of randomly generated target trajectories.

Example:
Input: 10 10 \ 5 5 \ 1 1 \ 2 2
Output: NE

Task 13: Dynamic Target Pursuit (Inspired by chat.deepseek.com)

You are tasked with writing a program to simulate an agent (e.g., a robot or drone) that must catch a moving target in a 2D grid-based environment. The target moves according to a predefined pattern, and the agent must determine the optimal path to intercept the target as quickly as possible.

Environment: A finite 2D grid of size $M \times M$, (1 ≤ M ≤ 100) with no wrap-around behavior.

Movement Rules:

- The **agent** moves in four directions: {N, E, S, W}.
- The **target** follows a repeating predefined movement sequence (e.g., E, N, W, S).
- The agent and target move simultaneously, one step at a time.
- The agent catches the target if they occupy the same cell at the same time.
- Both the agent and target cannot move outside the grid boundaries.

Input:

- Grid size M
- Initial positions of agent (x_a, y_a) and target (x_t, y_t)
- Target's repeating movement sequence

Output:

- Sequence of agent moves to intercept the target
- Total number of steps taken
- If interception is not possible, return "Target unreachable"

Scoring:

- 50%: Correctness
- 30%: Algorithmic efficiency
- 20%: Handling of edge cases (e.g., unreachable targets)

Example 13-1:
Input: $M = 5$; Agent: (0, 0); Target: (2, 2); Target Movement Sequence: (E, N, W, S).
Output: Agent moves (E, E, N, N).
Explanation:

At step 1: Agent moves E to (0, 1); Target moves E to (2, 3).
At step 2: Agent moves E to (0, 2); Target moves N to (1, 3).
At step 3: Agent moves N to (1, 2); Target moves W to (1, 2).
At step 4: Agent moves N to (2, 2); Target moves S to (2, 2).
Agent catches the target at step 4.

Remark. This task challenges participants to think about pathfinding, simulation, and optimization in a dynamic environment. It can be extended with additional complexities, such as obstacles in the grid or multiple targets, to increase the difficulty level for higher-tier Olympiads.

Extending this idea, we define the following generalized task:

General Task 14: Virus Immobilization on a Directed Graph

Environment: A directed graph. The adversary ("virus") starts at a known vertex and may move at each time step.

Variants:

- **(A)**: Virus moves arbitrarily along any outgoing edge.
- **(B)**: Virus follows a deterministic rule for choosing edges.

Objective: At each step, one or more vertices (excluding the virus's current location) may be disabled ("closed"). The goal is to immobilize the virus in the minimum number of steps.

Remark. In case (B), the problem may be interpreted as a pseudo-game due to the partially predictable behavior of the virus.

For example,

Task 15: Virus on a Line with Position Access (One-Dimensional)

Environment: Grid size 1..100, Virus starts at given position P_0 in {1, ..., 97}, and can move -2, -1, 1, 2 (arbitrarily).

Interaction Modes:

- P1 (Full Information): A numerical function Pos1() returns the current virus location.
- P2 (Partial Information): A Boolean function Pos2() returns true if the virus is at P; otherwise, false. This function may be used once per move.

Constraint: At each step, one position may be blocked with Clo-put() (excluding the current virus location).

Example 15-1.
Beginning of solution of the *Task 15-P1* for $P_0 = 50$.

- p_time := 0; Clo-put(52); [Virus moves]
- If Pos1() = 51 then Clo-put(53);
- If Pos1() = 49 then Clo-put(47);
- If Pos1() = 48 [Virus's optimal move] then Clo-put(46) [p_time = 2]

Example 15-2.
Beginning of solution of the *Task 15-P2* for $P_{0=}50$.

- p_time := 0; Clo-put(52); [Virus moves]
- If Pos2(51) then Clo-put(53) else Clo-put(47) [Virus moves]
- If Pos2(48) then Clo-put(46) else Clo-put(53) [Virus moves]

Task 16: Deterministic Virus in 2D Grid (for Task 14-B)

Environment: 100 × 100 grid. The virus attempts to move in priority order: East → North → West → South. If all are blocked, the virus stops. Agent can Clo-put an obstacle at each step.

Task: Given the virus's initial location, determine the minimum number of steps required to immobilize it by placing one obstacle per time step.

Example 16-1.

- p_time := 0; P_0 = (99, 100). Clo-put(100, 99) [Virus moves N];
- Clo-put(99, 100);
- Stop.

Result: Virus is immobilized and p_time = 2.

5. Tasks Involving Temporal Constraints

We now consider "timexcels" – time-constrained positions – as part of the environment.

Task 17: Robot Movement on Timed Platforms (1D)

Environment Setup:

- The robot starts at $P_{0\neq}100$
- It can wait or jump(K) where $|K| < 10$
- M timed platforms (coasters) are given: each specified by a position and time

Objective: Find the shortest command sequence to reach 100, using only valid coaster positions at the correct times.

Example 17-1:

- Input: $P_{0=}97$; Platforms: (98, 1), (95, 2), (91, 3), (100, 4)
- Sequence: Clo-wait; Clo-jump(-2) [R = 95, 2nd coaster]; Clo-jump(-3) [R = 92, 3rd coaster]; Clo-jump(8) [R = 100, goal]
- Output: WAIT, JUMP(-2), JUMP(-3), JUMP(8); p_time = 4

Task 18: Robot Navigation with Temporal Obstacles

Environment Setup:

- 1D grid 1..100
- The robot executes commands E and W
- M(1..1000) time-bound obstacles are specified by position and time (1D-timexcels)
- Guaranteed that there exists a sequence of commands to bring Robot to 100

Objective: Determine the shortest sequence of commands that navigates the robot from $P_0 \neq 100$ to 100 while avoiding obstacles.

Example 18-1:

- Input: $P_0 = 97$; Obstacles: (97, 2), (99, 2) [two obstacles appear at p_time = 2].
- Sequence: The command Clo-E cannot be continued because the sequences EE (to 99) and EW (to 97) cannot be executed.
- Output: Path WEEEE, p_time = 5.

Open Question: Can such tasks be solved more efficiently than via brute-force search?

6. Conclusion

This paper presents a range of computational tasks involving dynamic environments and limited observability. These tasks, often deceptively simple in description, reveal deep algorithmic complexity due to temporal dynamics and evolving state constraints. Many such problems are not solvable by brute force due to combinatorial explosion, highlighting the need for intelligent search strategies and heuristics. We propose that incorporating such "natural" tasks into Olympiads can better prepare participants for real-world algorithmic problem solving, as emphasized in Pankov (2008).

References

Steinhaus, H. (1989). *Kalejdoskop Matematyczny*. Wydawnictwa Szkolne i Pedagogiczne, Warszawa.

Steinhaus, H. (1999). *Mathematical Snapshots*. Dover Publs.

Kołodziejczyk, M. (n. d.). Two-pan balance and generalized counterfeit coin problem `https://www.mimuw.edu.pl/~andkom/Kule-en.pdf`

Pankov, P.S. (1990). Duality of control and observation parameters in obtaining guaranteed estimates. In: *Problems of Theoretical Cybernetics: Abstracts of Reports of the IX All-Union Conference* (September 1990). Volgograd, Part 1(2), p. 57.

Pankov, P.S., Imanaliev, T.M., Kenzhaliev, A.A. (2021). Automatic Makers as a Source for Olympiad Tasks. *Olympiads in Informatics*, 15, 75–82.

Pankov, P.S. (2008). Naturalness in Tasks for Olympiads in Informatics. *Olympiads in Informatics: Country Experiences and Developments*, 2**,** 16–23.

P.S. Pankov (1950), doctor of physics-mathematics sciences, prof., corr. member of Kyrgyzstan National Academy of Sciences (KR NAS), was the chairman of jury of Bishkek City OIs, 1985–2013, of Republican OIs, 1987–2012, participates in National OIs since 2020, was the leader of Kyrgyzstani teams at IOIs, 2002–2013, 2018–2023. Graduated from the Kyrgyz State University in 1969, is a head of laboratory of Institute of mathematics of KR NAS.

E.S. Burova (1986), Assistant Professor, Applied Mathematics and Informatics Program, American University of Central Asia.

E.J. Bayalieva (1984), Senior Lecturer in the Software Engineering program, Institute of Computer Technologies and Artificial Intelligence, J. Balasagyn National University.

Olympiads in Informatics, 2025, Vol. 19, 129–144

DOI: 10.15388/ioi.2025.09

OI-Assistant: A Retrieval Augmented System for Similar Problem Discovery and Interactive Learning in Competitive Programming

Yuhua SU[1,*], Ping NIE[2], Xin MENG[2]
[1]*International School Altdorf, Altdorf, Switzerland*
[2]*Peking University, Beijing, China*
e-mail: suyuhuahz21@gmail.com, ping.nie@pku.edu.cn, 1601214372@pku.edu.cn

Abstract. Competitive programming (CP) often requires quickly identifying relevant problems and solutions, yet current online judge (OJ) platforms offer only limited keyword or tag-based search. This makes it difficult for contestants and coaches to find past problems with similar patterns or concepts, hindering efficient practice and problem-solving.

We introduce OI-assistant, the first intelligent problem search and solution assistant based on Retrieval Augmented Generation (RAG) to bridge this gap. The proposed OI assistant provides insightful and similar problems based on our curated problem database, detailed and structured code explanation, and interactive code validation and follow-up chat.

We first collected over 11,000 programming problems from Luogu, a widely-used Chinese platform. Then we use multiple embeddings and llm-based ranker to retrieve and rank semantically similar CP problems based on user queries or code snippets. An LLM generates context-aware responses, like related problem suggestions or solution summaries, enabling more accurate discovery than traditional keyword-based searches. Additionally, it validates these solutions by automatically generating test cases, validating code in real-time and providing educational improvement feedbacks.

The proposed RAG-based search engine significantly improves the precision and recall of finding relevant problems, as evidenced by enhanced search results in our preliminary tests. When we introduced OI-Assistant to competitive programming students, their feedback was overwhelmingly positive. They rated the similar-problem recommendations highly and particularly appreciated the clear algorithm visualizations and real-time validation and improvement feedbacks. Overall, students found our platform significantly more helpful compared to traditional OJ systems or standalone ChatGPT.

By simplifying the discovery of related practice problems and enhancing real-time interactive learning, OI-Assistant significantly improves the effectiveness of competitive programming training and opens up new possibilities for the community.

Keywords: Competitive programming, Retrieval Augmented Generation, Large Language Model

* Corresponding author

1. Introduction

Competitive programming (CP) has grown remarkably worldwide in recent decades. Back in 2000, the International Olympiad in Informatics in Beijing had 278 participants from 72 countries. Last year's IOI in Egypt drew 362 participants from 91 countries. Similarly, national contests like the USACO Open have seen participation quadruple in just ten years. This surge in interest has led to a boom in problem repositories and online judge (OJ) platforms where competitors hone their skills.

Current OJs have a major weakness: they can't effectively search for problems based on their underlying mathematical concepts (C.R.A.C. Generation, 2024; Sollenberger *et al.*, 2024). Simple keyword searches miss the importance of CP problems, especially CP problems typically are wrapped in stories to hide their core algorithmic challenges. The search options on popular platforms like Codeforces, LeetCode, and Luogu only let you filter by basic methods like difficulty, algorithm tags, or problem sources. These methods often return too many irrelevant results, forcing users to spend time manually filtering through them. The problem gets worse because algorithm tagging is wrong or inconsistent across platforms.

We created OI-Assistant to solve these issues. Our system uses Retrieval Augmented Generation (RAG) specifically designed for competitive programming education (Lewis *et al.*, 2020; Shao *et al.*, 2025). It brings three key innovations. First, we built a rich database with over 11,000 quality problems from Luogu, one of the most popular competitive programming sites. This extensive collection gives our similarity search a strong foundation, helping students find inspiration from problems with related patterns. Second, we developed a multi-pronged search approach using three different strategies: question vector search, concept vector search, and summary vector search (Lewis *et al.*, 2020). We enhance these parallel searches with an LLM-based reranker that significantly boosts result quality. Our tests show this method achieves over 80% recall when finding similar problems – much better than traditional keyword searches. Third, OI-Assistant generates comprehensive solutions that include detailed algorithm explanations, clear flowcharts, and a real-time code validation system (Kumar, 2025; Fakhoury *et al.*, 2024). This validation feature uses GPT to automatically create test cases that cover even edge scenarios (Sollenberger *et al.*, 2024). Users can run their code right in our platform, get immediate feedback, and receive helpful suggestions when errors occur (Zhou *et al.*, 2024; Nicol and Macfarlane-Dick, 2006). The system can also regenerate improved solutions that address specific issues, creating a feedback loop that enhances learning (Zhang *et al.*, 2024).

Our system builds on recent advances in large language models. As our experiments show, modern GPT models like O3-mini can score at bronze medal levels in IOI competitions even without retrieval augmentation (El-Kishky *et al.*, 2025). But OI-Assistant's real value isn't about beating these baseline capabilities – it's about educational impact (Marouf *et al.*, 2024; Alyoshyna, 2024).

By combining strong problem-solving abilities with our innovative retrieval system, OI-Assistant creates a powerful learning tool that helps students find relevant historical problems matching their current needs (Kazemitabaar *et al.*, 2024; Denny

et al., 2023). Our retrieval system's 80%+ recall rate ensures students efficiently find appropriate practice materials, while the real-time validation provides the immediate feedback crucial for learning (Price *et al.*, 2016; Price *et al.*, 2017). User studies confirm that this feature combination significantly improves problem-solving skill development and learning efficiency compared to traditional approaches (Anderson *et al.*, 1995; Marouf *et al.*, 2024).

2. Related Work

2.1. *LLMs in Computer Science Education*

Generative AI is changing how we teach computer science. Researchers are exploring ways to use Large Language Models (LLMs) in educational settings, with promising results for helping students learn programming (Kazemitabaar *et al.*, 2024; Zhang *et al.*, 2024; Alyoshyna, 2024). Recent studies show LLMs can solve programming problems quite well. Denny and colleagues (Denny *et al.*, 2023) tested GitHub Copilot on 166 programming problems. It solved about half of them on the first try. With better prompts, that success rate jumped to 60% for the remaining problems. Even more impressive, OpenAI's ChatGPT-o3 earned a gold medal at the 2024 International Olympiad in Informatics and achieved a rating on Codeforces similar to top human competitors (El-Kishky *et al.*, 2025). These results show that today's best LLMs can perform at high levels in competitive programming.

LLMs can do more than just solve problems – they can create educational content too. Kazemitabaar and team (Kazemitabaar *et al.*, 2024) built CodeAid, a coding assistant based on ChatGPT. It has six functions, including writing code, explaining concepts, and fixing errors. They tested it with 700 students over a full semester. The feedback was mostly positive. This shows how carefully designed prompts can make LLMs much more useful for teaching. These studies lay the groundwork for using LLMs in programming education.

Fakhoury (Fakhoury *et al.*, 2024) and Sollenberger (Sollenberger *et al.*, 2024) shows these models can generate test cases and check if code is correct. This creates interactive learning environments that both generate and validate code. Zhang (Zhang *et al.*, 2024) studied what students want from AI feedback. They found that detailed explanations in context are what students value most when learning to program.

2.2. *Current Online Judge Status*

As shown in Table 1, Online Judge (OJ) platforms have grown from simple grading tools into full-fledged competitive programming communities (El-Kishky *et al.*, 2025; C.R.A.C. Generation, 2024). Today's platforms like Codeforces let users join contests, participate in forums, and study other coders' solutions. Luogu offers similar features.

Table 1
Different OJ Platforms

OJ	Search by difficulty	Algorithm	Source	Title search	Content search	Similarity search
Codeforces	800–3500	37 parallel tags	Yes	No	No	No
LeetCode	3 categories	71 parallel tags	Yes	Yes	No	No
Luogu	7 categories	Tags in 22 categories	Yes	Yes	Yes	No
USACO	7 categories	162 parallel tags	Yes	Yes	No	No
SPOJ	Rating for conceptual and implementational difficulties	119 Tags	Yes	No	No	No
UVa	NO	No	Yes	No	No	No

These platforms have become essential for competitive programmers to learn and connect with others.

Despite these improvements, OJs still struggle with organizing problem databases and providing good search tools, even though users really want these features (C.R.A.C. Generation, 2024; Sollenberger *et al.*, 2024). We surveyed six major OJ platforms and found several patterns in how they handle searching:

- **Search by origin:** Almost all OJs tag problems by where they came from, as shown in Table 1. This lets users filter problems by specific competitions or sources. Only SPOJ lacks this kind of search help.
- **Search by difficulty:** Most OJs group problems by how hard they are, but they do this differently. Codeforces uses numbers from 800–3500, while SPOJ separates difficulty into concept and implementation scores based on user votes.
- **Search by algorithms:** Algorithm tagging varies widely across platforms. Codeforces, LeetCode, and USACO use flat tag structures with no hierarchy. Luogu offers better organization with 22 algorithm categories, each containing related tags. SPOJ has the most sophisticated approach with a tree-structured system. All OJs support tag searches, but the differences between platforms make things confusing for users.
- **Search by content & problem similarity:** The biggest gap is in content-based searching (Lewis *et al.*, 2020; Asai *et al.*, 2024). While some OJs let you search by problem title or text, none offer true similarity search based on the underlying math concepts. This is a serious problem since competitive programming tasks usually come wrapped in stories during contests like IOI, ICPC, and Codeforces, which makes keyword searching pretty useless.

2.3. *Retrieval-Augmented Generation*

LLMs can do amazing things, but they still struggle with making up information and having outdated knowledge, especially in specialized fields (El-Kishky *et al.*, 2025; Tang *et al.*, 2024). Retrieval-Augmented Generation (RAG) solves these problems by

combining external information lookup with LLM generation. This greatly improves accuracy and relevance (Lewis *et al.*, 2020; Asai *et al.*, 2024).

For programming tasks, Wang and team created CODERAG-BENCH (C.R.A.C. Generation, 2024), a benchmark for testing how well RAG works for coding. Their research shows that RAG-enhanced models consistently beat regular LLMs, especially when tasks need external libraries. This matters for competitive programming, where specialized algorithms are often needed.

Shao's team (Shao *et al.*, 2025) looked at the effects of scale in retrieval-based language models with their MassiveDS project, which has a massive 1.4 trillion-token database. They found that bigger datastores consistently improve performance across language modeling and various tasks. Interestingly, their smaller models with large datastores outperformed bigger models without retrieval in knowledge-heavy tasks. This finding applies directly to algorithm-intensive competitive programming.

In scientific applications, Asai and colleagues (Asai *et al.*, 2024) developed OpenScholar, a retrieval-enhanced model for synthesizing scientific literature. By pulling relevant passages from open-access papers, OpenScholar reduced fake citations and beat larger models like GPT-4 in factual accuracy. This shows how RAG can improve precision in technical areas.

Collectively, these studies establish that RAG significantly enhances LLM performance in tasks requiring current, domain-specific knowledge (Lewis *et al.*, 2020; Shao *et al.*, 2025; Asai *et al.*, 2024). This approach proves especially valuable for competitive programming applications, where precise algorithmic understanding and accurate code generation are essential. Implementing RAG frameworks in this context can substantially improve solution retrieval, problem explanation, and code generation capabilities (C.R.A.C. Generation, 2024; Tang *et al.*, 2024; Zhou *et al.*, 2024).

3. Framework

Our OI-Assistant helps students find the right practice problems and understand their solutions (Kazemitabaar *et al.*, 2024; Marouf *et al.*, 2024). Fig. 1 shows how our system works. It has four main parts: Data Construction, Backend Search, Solution Generation, and Frontend Display.

3.1. *Data Construction*

We started by building a large collection of programming problems (Shao *et al.*, 2025; Asai *et al.*, 2024). We created web crawlers to gather problems from Luogu, a popular Chinese programming platform. We collected over 11,000 problems along with their details and more than 10,000 community solutions.

Programming contests often present problems as stories (El-Kishky *et al.*, 2025). While these narratives make problems interesting, they hide the core math concepts,

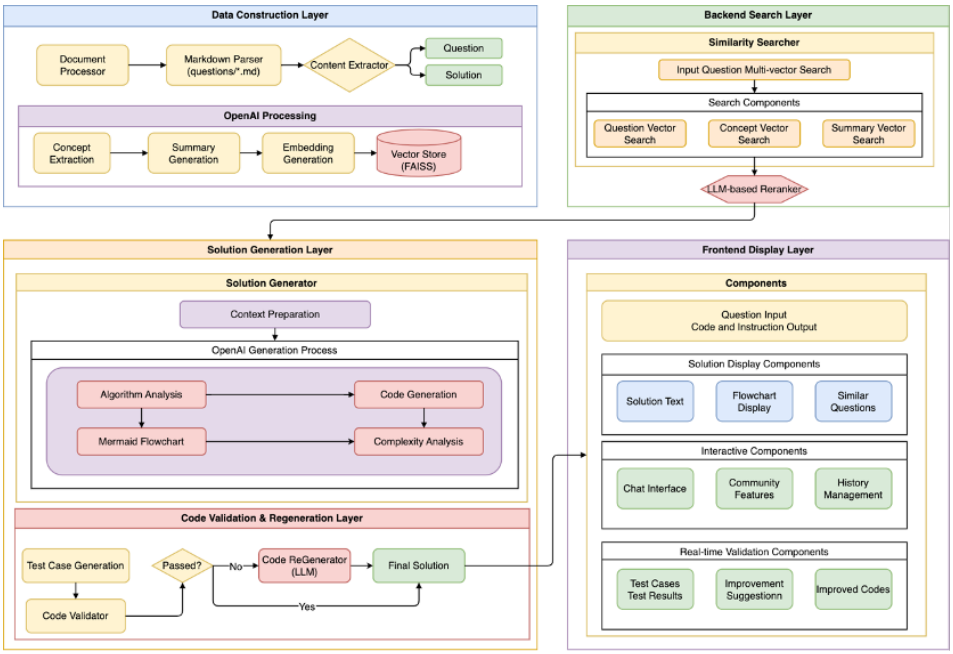


Fig. 1. OI-Assistant Framework.

which makes keyword searches pretty useless. We solved this by using large language models with carefully designed prompts to extract the mathematical essence from each problem, separating the algorithm from the story (Denny *et al.*, 2023; El-Kishky *et al.*, 2025).

We also use Luogu's tagging system, which groups problems by categories like Dynamic Programming, Graph Theory, Math, and Data Structures. Each category has more specific tags – for example, Dynamic Programming breaks down into 1D DP, Interval DP, Tree DP, and so on. Luogu rates problem difficulty on a 7-level scale:

Level 1: Beginner problems teaching basic concepts
Level 2: Easy problems needing simple algorithms
Level 3: Medium problems combining multiple ideas
Level 4: Hard problems requiring complex algorithms
Level 5: Provincial competition level
Level 6: National competition (NOI) level
Level 7: International Olympiad (IOI) level

After cleaning the data, each problem has a standard format with the problem statement, algorithm tags, difficulty level, and solution. For each problem, we create three different vector embeddings using OpenAI's API: Statement embedding that captures the math concepts, Concept embedding that represents the algorithms needed, Solution embedding that encodes how to implement the answer. We store these in FAISS indexes so we can search them quickly.

3.2. *Assistant Response Generation*

Finding similar problems is helpful, but students also need to understand how to solve them (Kazemitabaar *et al.*, 2024; Zhang *et al.*, 2024. As shown in Fig. 1 (bottom left), our Solution Generation Layer transforms a user's problem query into a comprehensive educational solution through several interconnected stages.

3.2.1. *User Input Processing and Retrieval.*

When a user submits a problem, the system first processes it through the Backend Search Layer shown in Fig. 1 (top right). The input question undergoes multi-vector search, which splits into three parallel components:

1. **Question Vector Search** – finds problems with similar statement structures,
2. **Concept Vector Search** – identifies problems using similar algorithmic techniques,
3. **Summary Vector Search** – locates problems with similar high-level approaches

The results from these three search components feed into the LLM-based Reranker (shown in the pink oval in Fig. 1), which evaluates each candidate problem in context and identifies the most top 50 relevant matches.

3.2.2. *Solution Generation Process*

As depicted in Fig. 1 (bottom left), the Solution Generator begins with Context Preparation. This crucial step combines the original problem with the retrieved similar problems and their solutions from our database. This context gives our system concrete examples of approaches that worked for similar challenges. The OpenAI Generation Process (shown in the central box) consists of four key components working in tandem:

1. **Algorithm Analysis:** (shown in the pink box) – The system creates a detailed explanation of the solution approach, key insights, and reasoning steps (Zhang *et al.*, 2024). This forms the conceptual foundation of the solution.
2. **Mermaid Flowchart:** (shown in the pink box) – In parallel, the system generates a visual flowchart using Mermaid syntax. This visualization helps students understand the algorithm's workflow intuitively, making complex concepts easier to grasp.
3. **Code Generation** (shown in the pink box): The system produces implementation code in the student's preferred programming language based on the algorithm analysis.
4. **Complexity Analysis** (shown in the pink box): The system explains the time and space complexity of the solution, helping students understand efficiency considerations (El-Kishky *et al.*, 2025).

3.2.3. *Code Validation and Regeneration*

What makes our system especially valuable is the Code Validation & Regeneration Layer shown at the bottom of Fig. 1 (bottom left). This layer includes: 1. Test Case Genera-

tion: The system automatically creates diverse test cases covering both normal scenarios and edge cases (Sollenberger *et al.*, 2024). 2. Code Validator: These test cases are fed into the Code Validator, which executes the generated code against the tests. 3. Feedback Loop: If the code passes all tests (the "Yes" path in Fig. 1), the system produces the Final Solution. If any tests fail (the "No" path), the Code Regenerator (LLM) analyzes what went wrong and creates an improved solution.

This validation-regeneration cycle continues until a correct solution is found, mimicking how a teacher might guide a student through debugging (Anderson *et al.*, 1995; Price *et al.*, 2017). When tests fail, the system doesn't just flag errors – it analyzes the specific issues and suggests targeted improvements (Zhou *et al.*, 2024; Nicol and Macfarlane-Dick, 2006).

3.3. *Frontend Interface*

As shown on the right side of Fig. 1, the Frontend Display Layer shows the system's functions through nice user interface. Fig. 2 and Fig. 3 illustrate actual screenshots of the UI interface.

3.3.1. *Solution Display Components*

The middle right side of Fig. 1 highlights components of the solution display:

- **Solution:** Clearly shows the algorithm and codes using clean description.
- **Flowchart:** Provides visual diagram that illustrates the algorithm's workflow. This is helping students learn the solution process easier.
- **Similar Questions:** gives related practice problems from our database. This is helping students recognize patterns and connections across multiple problems (Price *et al.*, 2016). Each similar problem also has a reason of its relevance to the current user input.

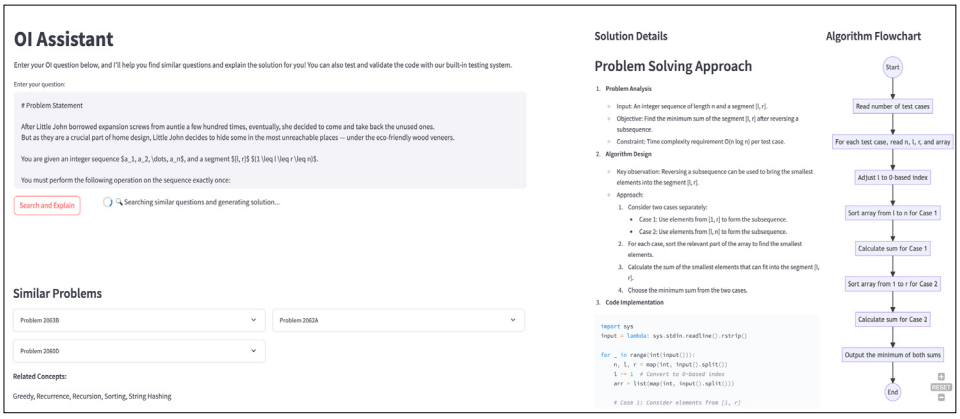


Fig. 2. OI-Assistant Frontend for Response.

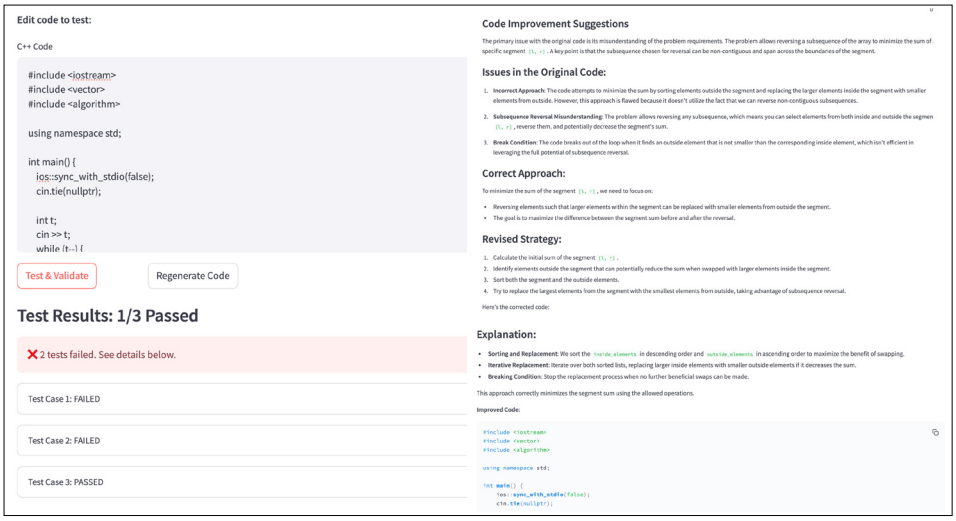


Fig. 3. OI-Assistant Frontend for Real-time Code Validation and Re-generation.

3.3.2. *Interactive Components*

Fig. 2 shows the OI-Assistant's frontend response and Fig. 3 shows the real-time code validation and re-generation feature. As shown in Fig. 2, Student can input the question statement or some code snippts to ask our system, our system will search our database to find similar statements and solutions. Also, the concepts for the user input are also displayed. Then on the right side of Fig. 2, the system will teach the student to understand the problem by text descriptions and a flowchart. After the flowchart, there is a code for the input question. In Fig. 3, there is a code box whose default code is from the system generation. The user can also edit the code box. Then the user can click the button to validate the code in the box. Our system will on the fly generate the test cases for the user input statement and the system will execute the code to check if the code can cover all generated test cases. If some test cases are not passed, the system will again look at the code and failed cases to generate some suggestions for improvement. The system will also output improved codes. The user can copy the improved code or input their own new code back to the code box to validate. This system then provides a super useful interactive process for learning.

4. Experiments and Evaluation

We built a complete evaluation framework to test OI-Assistant. Our tests cover everything from basic model abilities to how users feel about the system (Kazemitabaar *et al.*, 2024; Fakhoury *et al.*, 2024). Each experiment shows how different parts work together to create an effective learning tool.

4.1. *LLM Performance on IOI 2024*

As Fig. 4 shows, all models get better results when they try more times. The O3-mini model starts with okay performance. Yet with enough attempts, it reaches scores similar to human bronze medalists (around 215 points). This matters for real-world use. Even smaller models can do well if you let them try multiple times (El-Kishky *et al.*, 2025; Zhou *et al.*, 2024). We found that performance tends to level off after 10–20 attempts, suggesting this is a practical limit by computation.

Our system needs LLMs that can solve programming problems well (El-Kishky *et al.*, 2025). We tested several GPT models on IOI-2024 problems using methods from the Hugging Face IOI repository. One key question: how does performance improve with multiple solution attempts?

4.2. *Dataset Characteristics and Analysis*

After we know we can get good performance with SOTA LLMs, we next check our own dataset. This knowledge base powers our retrieval system, so it directly affects performance.

Our collected dataset includes 11,000 competitive programming problems from Luogu, one of China's busiest programming platforms. These problems come with 11,000

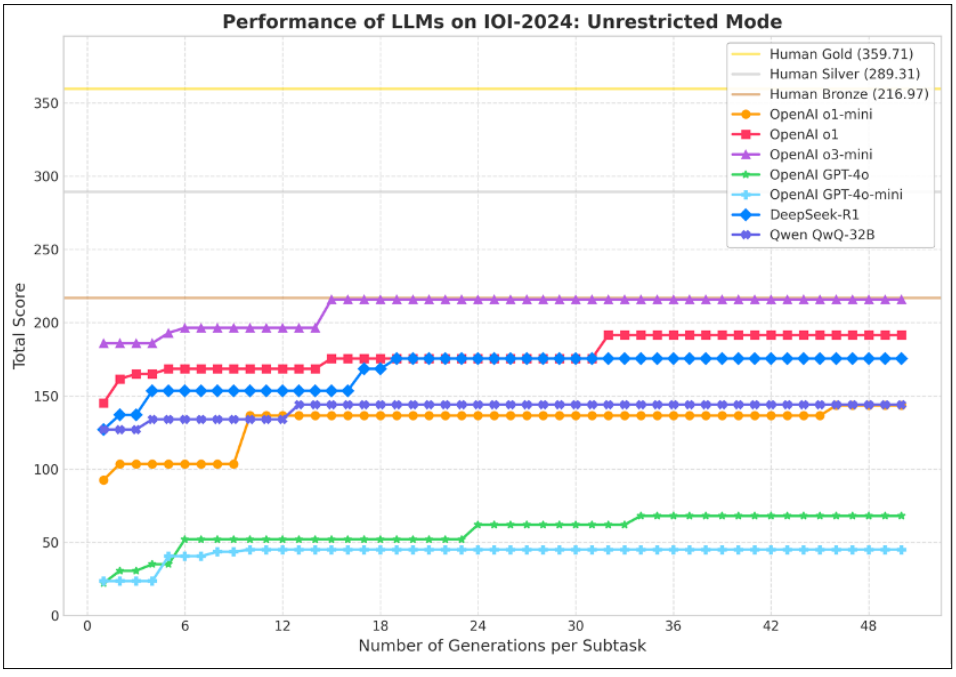


Fig. 4. OpenAI models performance on IOI 2024 with multiple generations.

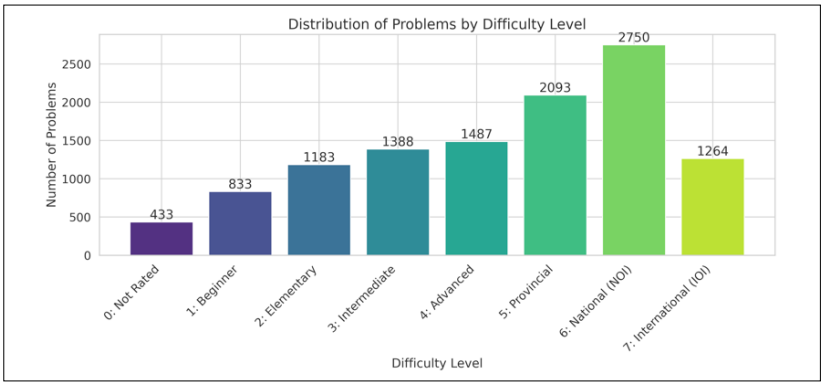


Fig. 5: Number of Problems for Difficulty Level.

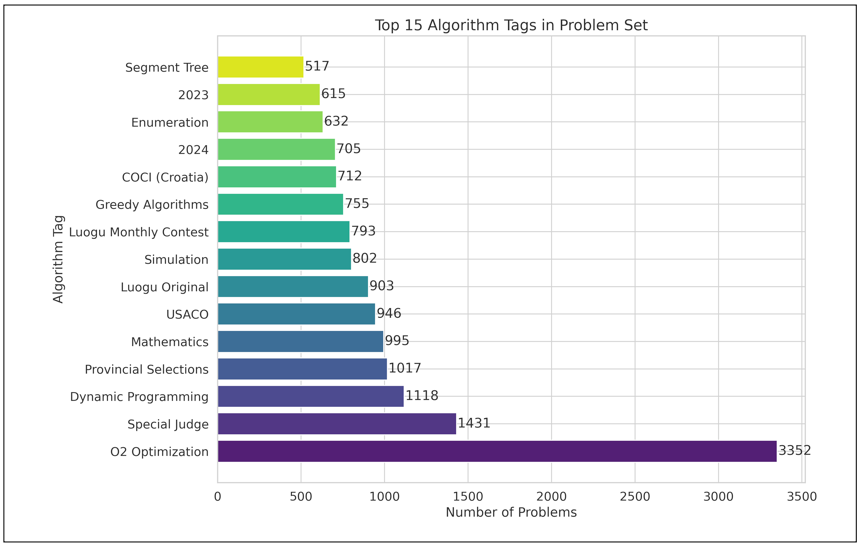


Fig. 6: Popular Tags in the dataset.

high quality solutions that have strong community engagement such as upvotes. We analyzed this dataset from several aspects to understand its educational value.

As shown in Fig. 5, we can see our dataset is balance for different difficulty level. Data difficulty definition can be found in section 3.1. Each difficulty level has about or more than 1000 questions. NOI questions are more than 2500. In Fig. 6, we can check the top 15 popular luogu tags for those questions in our datasets. The tags in Fig. 6 is from luogu to show it's real data features. Dynamic Programming and provincial selections are popular in our datasets. In Fig. 7, we can see the relationship between the acceptance and the difficulty of the luogu data. When the difficulty increases, the acceptance is dropping. And most of those questions are submitted by students for more than 10k times. This shows our dataset's meaningful status of helping those students to learn CP.

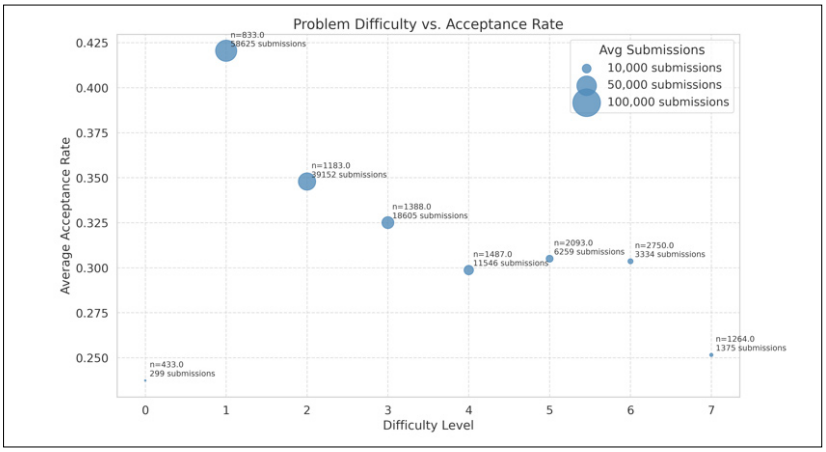


Fig. 7: Relationship between Difficulty and Acceptance Rate.

4.3. *Retrieval System Performance*

Building on our understanding of LLMs and dataset quality, we tested our multi-vector retrieval system. Traditional keyword searches often miss the deeper connections between problems, especially when similar challenges come wrapped in different contexts.

As shown in Fig. 8, we tested our system with three different settings to confirm our system can search and find similar problems given a new coding problem. Substring setting means the input question is a random substring of the existing problems in the datasets. Substring could be the problem definition or the solution to the existing problems. LLM Rewrite setting means we use GPT-4o to rewrite the existing questions with a different story to wrap the question with the same mathematic logics. Code Snippet setting means we use GPT-4o to generate an code answer for each existing problem. So with those three settings, we can comprehensively test our system's retrieval ability for both natural language and code input.

We created a dataset with 300 user input for each setting and 900 user input in total. We evaluated our 3 embedding retrievers and 1 llm ranker's recall performance. As shown in Fig. 8, the LLM ranker always gives the highest recall for three setting for all recall@K, where K = 10, 30, or 50. When we retrieve 50 candidates, the LLM ranker can achieve 80%+ recall for all settings. For the embedding retrieval, the Question representations gives the best score for about 69% at top 59% for the substring setting. For the code snippet settings, the recall drops dramatically to 30% which means it's hard for the embedding vectors to capture the semantic meaning of code. However, the Summary of the input can still keep decent recall at about 34% which shows the multiple vector embedding's advantages. After combining the results from three vector search, the LLM ranker can always get 80%+ recall at any settings. This shows the robustness of the LLM ranker. It also provide reliable similar problems output for our system.

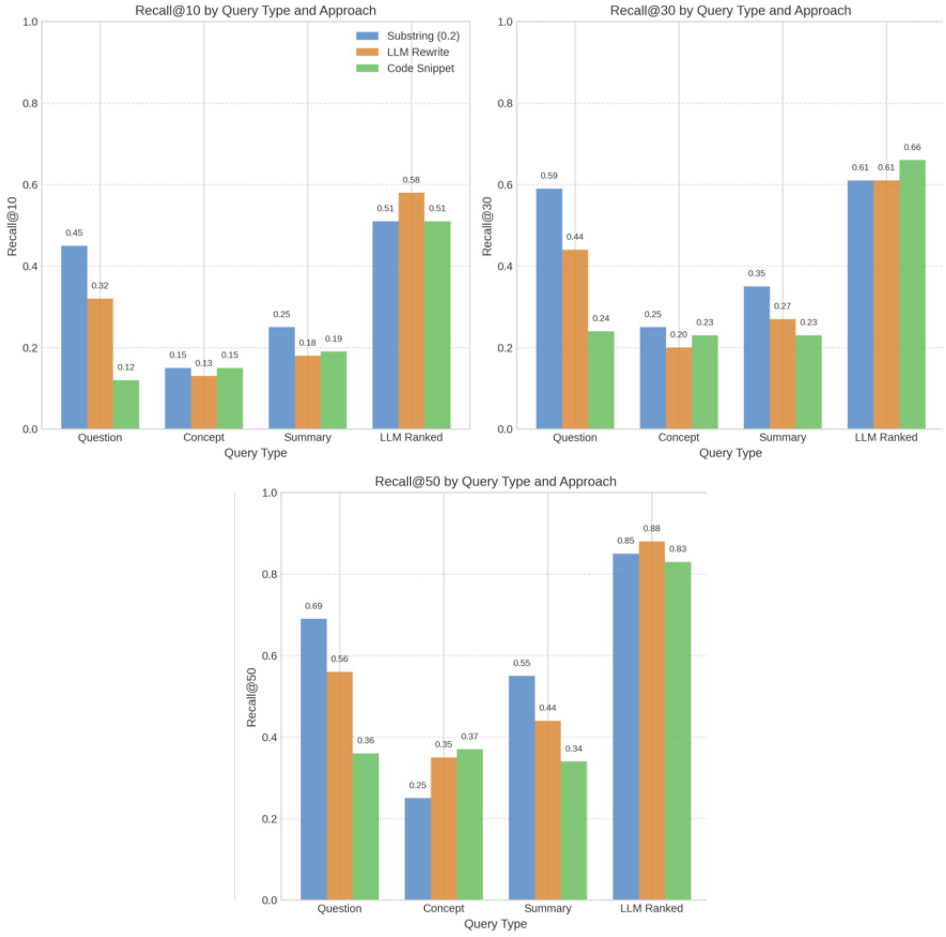


Fig. 8: Recall Performance of different modules on different settings.

4.4. *User Experience Evaluation*

After we confirm our system has good performance with multiple different recall testing. We conducted a user study with 36 competitive programming students who compared OI-Assistant with alternatives (ChatGPT and Luogu).

As shown in Table 2, our system outperformed both ChatGPT and luogu across all dimensions. Students especially give high ratings for our ability of finding similar problems (rated 4.8/5 compared to just 3.1/5 for ChatGPT and 2.8/5 for traditional platforms). This confirms that our focus on finding similar questions makes a real difference.

As shown in Table 3, we also show the system's helpfulness for different modules. students liked the similar problem recommendations (4.8/5), Code validation and feedback (4.9/5). These features create a comprehensive CP learning experience. Students can first understand solution approaches through clear explanations and visuals. Then

Table 2
User Ratings for Different Platforms

System Aspect	OI-Assistant	ChatGPT	Luogu
Solution quality	4.5	3.8	2.6
Finding similar problems	4.8	3.1	2.8
Overall Helpfulness	4.5	3.5	2.1

Table 3
User Ratings for Different Modules

OI-Assistant Feature	Average Rating (1–5)	Standard Deviation
Algorithm analysis quality	4.7	0.4
Flowchart visualization	4.6	0.5
Code validation and feedback	4.9	0.3
Similar problem recommendations	4.8	0.5
Interactive follow-up capability	4.1	0.6
LLM-generated test cases	4.3	0.3

they can improve their learning by using with similar problems found by our retrieval system and the real-time testing system with automatically generated test cases. The Overall helpfulness is also high for our system. Those real user experience feedbacks and ratings confirm our system's educational value.

5. Limitations and Future Work

Our OI-Assistant shows promise, but it has some clear limitations. Let's look at what could be better.

First, the system needs lots of computing power. This makes widespread deployment challenging. The LLM-based reranker that gives us great results also adds delays that users notice. When we generate multiple solutions to find the best one, we need even more computing resources. Not all schools or learning centers can provide this kind of cost. These issues matter most when trying to use the system in places with limited resources or when scaling up to many users.

Our dataset has its own limitations. We only used problems from Luogu. Programming problems often contain cultural references that might confuse users from different backgrounds. Also, different programming communities create problems in their own unique ways.

In the future, we could try model distillation to create smaller, faster versions of our reranker without losing much performance. Better search techniques, like hybrid search and small model ranking. Smart caching for common problem patterns would speed up responses for frequently asked questions.

We also want to expand what our database Adding more competitive programming platforms data would provide a richer knowledge base with diverse problem-solving approaches. Supporting more programming languages would help more students use the system, especially in schools that teach specific languages.

6. Conclusion

We developed OI-Assistant to address a significant challenge faced by competitive programming students and coaches that finding similar problems with similar concepts not just by tags is super useful for student learning. By leveraging Retrieval Augmented Generation (RAG), our system achieves promising results. Experiments demonstrate over 80% recall for retrieving similar questions even for code snippets. Users confirmed that our integrated features including solution generation, algorithm explanations, flowchart, and real-time code validation, greatly help their learning process. Our work shows a great future for competitive programming education. OI-Assistant doesn't just help students find problems; it strengths their understanding by connecting similar code problems and providing real-time feedback for any coding problems with automatic test cases and validation. This approach builds stronger thinking, helping students recognize patterns across diverse problems and figure out the errors in the code by test cases driven way and educational feedback. As language models continue to evolve, combining RAG with them will become increasingly valuable for specific knowledge base, making it easier for students to learn complex concepts and do better in competitive programming.

References

Alyoshyna, Y. (2024). AI in Programming Education: Automated Feedback Systems for Personalized Learning. *University of Twente Student Theses*.

Anderson, J.R., Corbett, A.T., Koedinger, K.R., and Pelletier, R. (1995). Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences*, 4(2), 167–207.

Asai, A., He, J., Shao, R., Shi, W., Singh, A., Chang, J.C. *et al.* (2024). OpenScholar: Synthesizing scientific literature with retrieval-augmented LMs. arXiv preprint arXiv:2411.14199.

C.R.A.C. Generation. (2024). *CODERAG-BENCH: Can Retrieval Augment Code Generation?*

Corbett, A.T. and Anderson, J.R. (1992). The LISP intelligent tutoring system: Research in skill acquisition. In: *Computer Assisted Instruction And Intelligent Tutoring Systems: Establishing Communication and Collaboration*. Lawrence Erlbaum Associates, Inc, 141–194.

Denny, P., Kumar, V., Giacaman, N. (2023). Conversing with Copilot: Exploring prompt engineering for solving CS1 problems using natural language. In: *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. 1136–1142.

El-Kishky, A., Wei, A., Saraiva, A., Minaev, B., Selsam, D., Dohan, D., *et al.* (2025). *Competitive Programming with Large Reasoning Models*. *arXiv preprint arXiv:2502.06807*.

Fakhoury, S., Naik, A., Sakkas, G., Chakraborty, S., and Lahiri, S.K. (2024). LLM-Based Test-Driven Interactive Code Generation: User Study and Empirical Evaluation. *IEEE Transactions on Software Engineering*.

Georgia Department of Education. (2020). *Georgia's ReStart: Embrace, Engage, Expand, and Enhance Learning with Technology (GRE4T) Initiative*.

Kazemitabaar, M., Ye, R., Wang, X., Henley, A.Z., Denny, P., Craig, M., Grossman, T. (2024). CodeAid: Evaluating a classroom deployment of an LLM-based programming assistant that balances student and educator needs. In: *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 1–20.

Kira Learning. (n.d.). *The AI platform for schools*.

Kumar, S. (2025). Teaching LLMs to generate Unit Tests for Automated Debugging of Code. *Medium*.
Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., *et al.* (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33, 9459–9474.
Marouf, A., Al-Dahdooh, R., Abu Ghali, M.J., Mahdi, A.O., Abunasser, B.S., and Abu-Naser, S.S. (2024). Enhancing Education with Artificial Intelligence: The Role of Intelligent Tutoring Systems. *International Journal of Engineering and Information Systems (IJEAIS)*, 8(8), 10–16.
Nicol, D.J. and Macfarlane-Dick, D. (2006). Formative assessment and self-regulated learning: A model and seven principles of good feedback practice. *Studies in Higher Education*, 31(2), 199–218.
Price, T.W., Dong, T., and Barnes, T. (2016). Generating data-driven hints for open-ended programming. In: *International Conference on Educational Data Mining*. 446–451.
Price, T.W., Zhi, R., and Barnes, T. (2017). Evaluation of a data-driven feedback algorithm for open-ended programming. In: *International Conference on Educational Data Mining*. 530–535.
Psotka, J., Massey, L.D., and Mutter, S.A. (Eds.). (1988). *Intelligent Tutoring Systems: Lessons Learned*. Lawrence Erlbaum Associates, Inc.
Shao, R., He, J., Asai, A., Shi, W., Dettmers, T., Min, S. *et al.* (2025). Scaling Retrieval-Based Language Models with a Trillion-Token Datastore. *Advances in Neural Information Processing Systems**, 37, 91260–91299.
Shi, X., Tian, M., and Zhang, J. (2022). A summary of personalized learning research. In: *IET Conference Proceedings*. Vol. 2022, No. 9, 53–58.
Sollenberger, Z., Patel, J., Munley, C., Jarmusch, A., and Chandrasekaran, S. (2024). LLM4VV: Exploring LLM-as-a-Judge for Validation and Verification Testsuites.
Tang, H., Hu, K., Zhou, J.P., Zhong, S.C., Zheng, W.L., Si, X., and Ellis, K. (2024). *REx: An Exploration-Exploitation Framework for LLM-Based Code Refinement*. arXiv preprint arXiv:2411.14199.
VanLehn, K. (1988). Student modeling and mastery learning in a computer-based programming tutor. In: *Intelligent Tutoring Systems*. Springer, Berlin, Heidelberg, 479–506.
Zhang, Z., Cheng, L., and Chen, X. (2024). Students' Perceptions and Preferences of Generative Artificial Intelligence Feedback for Programming. *Journal of Educational Computing Research*, 71(4), 647–673.
Zhou, Y., Peng, X., Zeng, A., Xie, Q., and Luo, T. (2024). *LLMFix: Automatically Fixing Code Generation Errors in Large Language Models*. arXiv preprint arXiv:2409.00676.

Y. Su – a senior high student at the International School Altdorf in Switzerland. With four years of competitive programming experience, he won a gold medal in the Swiss Olympiad in Informatics. His research interest focuses on machine learning and human-computer interaction. His past projects include building a stuttering recognition system and enhancing a micro-expression spotting network.

P. Nie – a Senior Applied Scientist with a Master's degree from Peking University. His research interests lie in Code Large Language Models, Information Retrieval, and Natural Language Processing. He also serves as a program committee member for conferences such as ACL, SIGIR, and NeurIPS.

X. Meng – a Senior Deep Learning Engineer with a Master's degree from Peking University. He has 10 years of experience in AI development, like CUDA acceleration, Parallel Computing. His research interests lie in Computer Vision and Autonomous driving with LLM, Robots, Embodied Intelligence.

Olympiads in Informatics, 2025, Vol. 19, 145–158

DOI: 10.15388/ioi.2025.10

The Olympiad Trap and an Old Trampoline

Tom VERHOEFF
Mathematics and Computer Science, Eindhoven University of Technology
Groene Loper 5, 5612 AE, Eindhoven, Netherlands
e-mail: t.verhoeff@tue.nl

Abstract. After some reminiscing, I describe the Olympiad trap and then delve into a technique to eliminate recursion by trampolining with continuations.

Keywords: programming, recursion, recursion elimination, continuations.

1. Introduction

Since I plan to retire in October 2025, I hope you will permit me to begin with a brief reminiscence, before telling you about the *Olympiad trap*. The bulk of this article, however, concerns a technical topic: how to eliminate recursion using an old technique known as a *trampoline* with *continuations*.

I graduated in 1985 (Applied Mathematics, Eindhoven University of Technology) and started there as a PhD candidate. In that same year, TU Eindhoven somehow was invited to participate in the preliminary regional round of the *ACM International Collegiate Programming Contest* (ICPC). We didn't know that PhD candidates were (at that time) allowed to participate. So, I became the team's coach (rather than a contestant). We went to London with an ad hoc team, and they qualified to participate in the ICPC World Finals in 1987 (Saint Louis, MO). Later that year, we organized a university-wide selection contest for the next regional round of the ICPC.

One thing led to another. In 1988, 1989, and 1990, I organized the preliminary round of the ICPC for Europe, Middle-East, and Africa (EMEA) in Eindhoven. The number of regions increased and they became smaller. In 1997, I organized the North-West European Regional Contest (NWERC), and in 1999, we had the honor of hosting the ICPC Word Finals in Eindhoven (the first time that it took place outside the USA). In 2004, I received the ICPC European Founders Award for my efforts.

It was because of my ICPC experience that Ries Kock of the Netherlands Informatics Olympiad (NIO) approached me in 1994. The Netherlands had been participating in the IOI since 1990, and Ries had taken up the challenge of organizing IOI 1995 in The Netherlands. He wanted me to head the Host Scientific Committee for IOI 1995. I went

to IOI 1994, in Sweden, as an observer, and got hooked. As part of the preparations for IOI 1995, I set up the IOI International Secretariat on the (then still very young) World-Wide Web. In 1999, the IOI International Scientific Committee was established, which I chaired until 2007. In that same year, I received the IOI Distinguished Service Award. Since then, I have followed the IOI on the side.

2. The Olympiad Trap

I already mentioned that I got hooked on the IOI during my first participation as an observer. At the IOI, it feels like you are part of an important mission: discovering, stimulating, and developing young talent (in informatics). You could consider the IOI a trap, because its attractive force keeps you involved. But that is not the trap I want to discuss here. I think that the IOI itself is trapped, viz. in its own format. That is what I mean by the Olympiad trap.

Science olympiads cannot cover the full breadth of their field, certainly not when the main event is a contest. The more prestigious and popular an olympiad becomes, the more the team leaders will want to select and prepare their contestants with a focus on what is relevant for the contest. This leads to training deeply for a narrow field. This in turn makes it harder to change the olympiad's format, because many people have invested in the current format. That is, the Olympiad is trapped in its format.

It is easy to lose sight of the breadth of the field and of social aspects when developing talent. This is particularly worrisome for a field like informatics that still evolves rapidly. Algorithmic problem solving plays a much smaller role nowadays, both in (higher) CS education and in research and industry than when the IOI was established. There is a host of other topics that attract attention. Of these, Data Science and AI are newcomers. Parents are already advising their children not to study informatics, because they fear that AI will affect the job market. That is why I think it should be mandatory to augment training for olympiads with other activities on the side, to help mitigate the Olympiad trap.

Such side activities should be interesting and challenging. In this article, I will explore such a side topic. Since this topic is still related to programming, it might even be useful for IOI contestants.

3. Limits on Recursion

Recursion is a great algorithmic technique, which can lead to more compact and clearer code for various problems. But recursion also has its dangers. One danger it shares with general `while` loops is that it can be hard to reason about such programs, in particular their termination (Verhoeff, 2018, 2023). Another danger is that recursion implicitly uses memory, viz. on the *call stack*, so that deeply nested recursion can run out of memory. In fact, some programming languages, such as Python and Java, impose a (configurable) limit on the recursion depth as built-in protection against infinite recursion. The default

limit for Python is 1000 levels and for Java 256; C++ does not impose a limit other than available memory.

In (Verhoeff, 2018), I discuss various aspects of recursion, in particular, *tail recursion* and how *linear* tail recursion can be mechanically turned into a loop to avoid burdening the stack (which I will recap below). In case of *branching* recursion, it may seem that only one recursive call can be a tail call, and thus the transformation into a loop fails. In (Verhoeff, 2021), we encountered functions that break the recursion by introducing an extra parameter, and then "tying the knot of recursion" on the outside, by making the snake eat its own tail (through a fixed-point construction, which still burdens the stack). It turns out that there is another technique to break recursion and make even branching recursion tail recursive.

I will illustrate this technique through two examples in Python. Source code and visualization of the stack usage during execution is available in (Verhoeff, 2025). The first example is based on function `tri(n)` that computes the n-th triangular number (similar to factorials, but using addition, so that the numbers don't grow so fast):

```
type nat = int  # with assumption >= 0

def tri(n: nat) -> int:
    if n == 0:
        return 0
    else:
        return n + tri(n - 1)
```

The second example is `total(t)` that sums the values in binary leaf tree t:

```
@dataclass
class Leaf:
    value: int

# binary tree type with int in leaves
type Tree = Leaf | tuple[Tree, Tree]

def total(t: Tree) -> int:
    if isinstance(t, Leaf):
        return t.value
    else:  # t is binary fork
        return total(t[0]) + total(t[1])
```

Function `tri` exhibits linear recursion and `total` exhibits branching recursion; neither is tail recursive, since more work is done after the recursive calls return.

The call `tri(1000)` will result in a `RecursionError`. In case of function `tri`, the standard technique of introducing an *accumulation* parameter yields a tail recursive definition:

```
def tri_acc(n: nat, acc: int = 0) -> int:
    if n == 0:
        return acc
    else:
        return tri_acc(n - 1, acc + n)
```

And this in turn is readily transformed into a loop, which avoids the dreaded `RecursionError`:

```
25 def tri_loop(n: nat, acc: int = 0) -> int:
26     while n != 0:
27         n, acc = n - 1, acc + n
28     return acc
```

Exercise: Show how one of the recursive calls in `total` can be transformed into a tail call by introducing an accumulation parameter. See Appendix A.1 for an answer.

4. The Trampoline

Even though the transformation from tail recursion to loop, shown above, is straightforward, it needs to be done for each tail recursive function separately. There is a simple technique that introduces only one loop, which can transform all tail recursive functions, after a small intervention. We don't want nested recursive calls, but we still want to keep the computation the same. This can be accomplished by returning the recursive call itself in *unevaluated form* and let the evaluation be continued from outside the recursive function, after it has returned. Sounds magical?

Python and many other languages (including Java and C++) offer syntax to define anonymous functions, that is, without giving them an explicit name. In Python, the syntax **`lambda`** `x, y: expr` defines a nameless function of two arguments with the result `expr`, where expression `expr` typically involves `x, y`. Similarly, we can use **`lambda:`** `expr` for a function without arguments that evaluates to `expr`.

Here is what `tri_acc` looks like after the intervention to make it return an unevaluated "recursive" call:

```
29 def tri_acc_lazy(n: nat, acc: int = 0) -> Thunk[int]:
30     if n == 0:
31         return acc
32     else:
33         return lambda: tri_acc_lazy(n - 1, acc + n)
```

It is no longer "truly" recursive, because it does not make the recursive call! We say that the call is *suspended*. In order to get the typing correct, we have defined type `Thunk[A]`:

```
34 # Thunk[A]: possibly nested suspended computation of type A
35 # A should not be callable
36 type Thunk[A] = A | Callable[[], Thunk[A]]
```

Note that in Python, `Callable[[], R]` denotes the type of functions *without arguments* returning a value of type `R`. So, a `Thunk[A]` is either a value of type `A` or a function without arguments returning a `Thunk[A]`. For value `thunk` of type `Thunk[A]`,

we can test whether it is actually suspended by `callable(thunk)`. And if it is suspended, it can be resumed by calling it as `thunk()`.

By deseign, `tri_acc_lazy` always immediately returns. How can we get the final result? That is where the *trampoline* gets to the rescue, since it repeatedly resumes a suspended computation until it gets a final (non-suspended) result:

```
37 def trampoline[A](thunk: Thunk[A]) -> A:
38     while callable(thunk):  # thunk is suspended
39         thunk = thunk()  # resume it
40     return thunk
```

Therefore, we have `tri(n) == trampoline(tri_acc_lazy(n))`. Note that `tri_acc_lazy` creates only one stack frame, and `trampoline` repeatedly resumes all suspended tail calls. The control flow bounces between the trampoline and the thunked (lazy) "tail recursive" function, where `lambda:` is placed in front of every tail recursive call to suspend it. Neat, isn't it?

Some notes:

- Upon superficial reading, the definition of `tri_acc_lazy` given above looks recursive, since the body of the function definition contains a call to the function itself.
- However, it does not execute that call itself; that is left to the client code. The "recursive knot" is tied on the outside, by `trampoline`.
- For this to work, the programming language must support function *closures* that capture the current values of variables. In case of `tri_acc_lazy`, the expression `lambda: tri_acc_lazy(n -1, acc + n)` involves two local variables, viz. `n, acc`, which evaporate after the function returns. Python binds their values in the returned lambda object.
- Thunking via `lambda:` resembles the *Command* design pattern from Object-Oriented programming (Gamma *et al.*, 1994).

5. Enforcing Tail Recursion Via Continuations

Before addressing `total`, let's generalize the technique with the accumulation parameter. In general, it may not be easy to find a simple accumulation parameter to make a definition tail recursive. And in case of branching recursion, it would be useless. But there is a way that is guaranteed to work: *Continuation Passing Style*, also known as CPS (Reynolds, 1993). With CPS, you introduce an extra parameter of a *function type*, known as a *continuation*. In Python, we abbreviate that function type to `Func[A, B]`:

```
41 type Func[A, B] = Callable[[A], B]  # functions from A to B
```

We name this continuation parameter `cont`. It represents work that still needs to be done to complete the computation. An example will make this clear. Let's specify `tri_cps(n, cont) == cont(tri(n))`. Then we have in mathematical notation

- $tri\,(n) = id\,(tri\,(n)) = tri\_cps\,(n, id)$, where id is the (polymorphic) identity function defined by $id\,(a) = a$;
- for $n = 0$, we have $tri\_cps\,(0, cont) = cont\,(tri\,(0)) = cont\,(0)$;
- and for $n > 0$, $tri\_cps\,(n, cont) = cont\,(tri\,(n)) = cont\,(n + tri\,(n-1))$.
  The latter expression can be viewed as a new function applied to $tri\,(n-1)$. Which function? The function f defined by $f\,(x) = cont\,(n + x)$. In Python, that function can be expressed as `lambda x: cont(n + x)`. So, we can rewrite further

$$\begin{aligned} cont\,(n + tri\,(n-1)) &= (\lambda\, x : cont\,(n + x))(tri\,(n-1)) \\ &= tri\_cps\,(n-1, \lambda\, x : cont\,(n + x)) \end{aligned}$$

This leads to the following tail recursive (!) definition of `tri_cps`:

```
42 def tri_cps(n: nat,
43              cont: Func[int, int] = id_
44              ) -> int:
45     if n == 0:
46         return cont(0)
47     else:
48         return tri_cps(n - 1, lambda x:
49                cont(n + x)
50                )
```

The identity function serves as default continuation. We have named it `id_`, because **`id`** is already a predefined different function in Python:

```
51 def id_[A](a: A) -> A:
52     return a
```

Function `tri_cps` first accumulates a (possibly big) continuation in n steps, which it then applies to 0. The evaluation of this continuation will also burden the stack. Thus, to make this suitable for trampolining and limiting the stack load, **`lambda:`** is also needed in front of both calls of `cont` (in addition to just the recursive call to `tri_cps_lazy`):

```
53 def tri_cps_lazy(n: nat,
54                  cont: Func[int, Thunk[int]] = id_
55                  ) -> Thunk[int]:
56     if n == 0:
57         return lambda: cont(0)
58     else:
59         return lambda: tri_cps_lazy(n - 1, lambda x:
60                lambda: cont(n + x)
61                )
```

Note that the two calls to `cont` were also tail calls. (This explains the somewhat odd layout of the code.)

6. Making Branching Recursion Tail Recursive

CPS is so powerful that it can even make functions with branching recursion tail recursive. Let's see how that works by revisiting `total` defined in §3. First, we specify `total_cps(t, cont) == cont(total(t))`. Then we have

- $total\,(t) = id\,(total(t)) = total\_cps\,(t, id)$;
- for $t = Leaf\,(v)$, we have $total\_cps\,(t, cont) = cont\,(total\,(t)) = cont\,(v)$;
- for $t = (t_0, t_1)$, we have

$$\begin{aligned}
&total_cps\,(t, cont)\\
&= cont\,(total(t))\\
&= cont\,(total\,(t_0) + total\,(t_1))\\
&= (\lambda\, tt_0 : cont\,(tt_0 + total\,(t_1)))(total\,(t_0))\\
&= total_csp\,(t_0, \lambda\, tt_0 : cont\,(tt_0 + total\,(t_1)))\\
&= total_csp\,(t_0, \lambda\, tt_0 : (\lambda\, tt_1 : cont\,(tt_0 + tt_1))(total\,(t_1)))\\
&= total_csp\,(t_0, \lambda\, tt_0 : total_csp\,(t_1, \lambda\, tt_1 : cont\,(tt_0 + tt_1))).
\end{aligned}$$

Thus, we have derived the following definition for `total_cps`:

```
62 def total_cps(t: Tree,
63                cont: Func[int, int] = id_
64                ) -> int:
65     if isinstance(t, Leaf):
66         return cont(t.value)
67     else:  # t is binary fork
68         return total_cps(t[0], lambda tt0:
69                total_cps(t[1], lambda tt1:
70                cont(tt0 + tt1)
71                ))
```

You may wonder whether this definition is really tail-recursive, because it contains two calls of `total_cps`, only one of which looks like a tail call. It is, since the call with `t[1]` is suspended (but not thunked) by **`lambda`** `tt0`. That call is incorporated into the continuation, and executed when that continuation reaches a leaf. Inside that **`lambda`** `tt0`, it is a tail call.

We can now easily prepare this for the trampoline by adding **`lambda:`** (4×):

```
72 def total_cps_lazy(t: Tree,
73                     cont: Func[int, Thunk[int]] = id_
74                     ) -> Thunk[int]:
75     if isinstance(t, Leaf):
76         return lambda: cont(t.value)
77     else:  # t is binary fork
78         return lambda: total_cps_lazy(t[0], lambda tt0:
79                lambda: total_cps_lazy(t[1], lambda tt1:
80                lambda: cont(tt0 + tt1)
81                ))
```

I hope that this example convinces you that through CPS the transformation of general, possibly branching, recursion to tail recursion can in fact be automated.

However, I need to temper your expectations somewhat. Exercise: Find a recursive function definition that cannot be made tail recursive via CPS. See Appendix A.2 for an answer.

7. Defunctionalization

What did we gain? Well, CPS with trampoline avoids using the stack, and deep recursion is no longer a problem when using a language that limits the recursion depth. Memory usage, however, has shifted from the stack to the continuation. This continuation grows and shrinks (in `total_cps` it can be discarded at leaves, once it has been applied, but that may give rise to a new continuation when hitting another recursive call). The computation is accumulated in the continuation (like meta-programming) and then applied.

That continuation contains all the information needed to complete the computation: both data (operands) and functionality (operations). Since the operations are fairly limited, it is inefficient to copy them multiple times into the continuation. Can't we take that duplicate functionality out and share it? Yes, that is possible through a technique known as *defunctionalization*.

Let's first do this for `tri_cps`. Consider its definition in §5. How are its continuations constructed? In particular, what data is involved and how is it structured? It starts off with `id_`, and "on top" of this, there appear multiple instances of **`lambda`** `x: cont(n + x)`, for varying `n: nat`. Hence, it seems plausible that we can encode a continuation as a **`list`**`[nat]`:

```
82 type tri_cps_data = list[nat]
```

We now define an auxiliary function `tri_cont` to reconstruct the continuation from the data and apply it:

```
83 def tri_cont(data: tri_cps_data, x: int) -> int:
84     if data:  # non-empty
85         n = data.pop()
86         return tri_cont(data, n + x)
87     else:  # empty, act as identity
88         return x
```

So, now the operation (viz. `n+x`) occurs once, viz. in `tri_cont`. The defunctionalized version of `tri_cps` is then given by

```
89 def tri_dcps(n: nat, data: tri_cps_data = None) -> int:
90     if data is None:
91         data = []  # to avoid mutable default argument
92     if n == 0:
```

```
93          return tri_cont(data, 0)
94      else:
95          data.append(n)
96          return tri_dcps(n - 1, data)
```

Of course, this can also be made lazy for the trampoline (see `tri_dcps_lazy` in (Verhoeff, 2025)).

For this particular function, we can go even further and reduce the data to a single natural number, because all continuations basically are compositions of `lambda x: n + x` for various values of n. The simplification hinges on associativity of function composition and this property:

$$(\lambda\, x : n + x) \circ (\lambda\, x : m + x) = (\lambda\, x : (n + m) + x))$$

Note that for $n = 0$, $\lambda\, x : n + x$ is the identity function. This simplification gives us back `tri_acc`, where the whole continuation is compressed into a single integer (`acc`).

It is also instructive to defunctionalize `total_cps` defined in §6. Its continuation grows in three ways:

- `id_`, without capturing any data, is the initial continuation;
- `lambda tt1: cont(tt0 + tt1)`, with `tt0: int` as captured data, extends `cont`;
- `lambda tt0: total_cps(t[1], lambda tt1: cont(tt0 + tt1))`, with `t[1]: Tree` as captured data, also extends `cont`.

Thus, we can encode a continuation as

```
97  type total_cps_data = list[int | Tree]
```

The `int` is the total of a left subtree that needs to be added to the total of a right subtree (which has not been determined yet), and the `Tree` is a right subtree which still must be totaled. Here is how to reconstruct the continuation from the data:

```
98  def total_cont(data: total_cps_data, x: int) -> int:
99      if data:  # non-empty
100         last = data.pop()
101         if isinstance(last, int):  # pending total of left tree
102             return total_cont(data, last + x)  # last is a tt0
103         else:  # pending right tree
104             data.append(x)
105             return total_dcps(last, data)  # last is a t[1]
106     else:  # empty, act as identity
107         return x
```

Notice the call of `total_dcps`. Thus we get the following defunctionalized version of `total_cps`:

```
108 def total_dcps(t: Tree,
109                 data: total_cps_data = None
110                 ) -> int:
111     if data is None:
112         data = []  # to avoid mutable default argument
113     if isinstance(t, Leaf):
114         return total_cont(data, t.value)
115     else:  # t is binary fork
116         data.append(t[1])  # postpone processing of t[1]
117         return total_dcps(t[0], data)
```

We now have two *mutually* tail-recursive definitions. Without knowing how these function definitions were derived, it would not be obvious why they terminate.

By the way, just as with `tri_dcps`, it is possible to merge some continuations for `total_dcps`, and to pass these on as a single integer (`acc`), and the remainder as a `list[Tree]`. We leave it as an exercise to the reader to fill in the missing details (see Appendix A.3 for some hints). The result is a classic tail recursive function, without suspended calls and without the need for an auxiliary function that explictly reconstructs the continuation.

```
118 def total_acc_dcps(t: Tree,
119                     acc: int = 0,
120                     data: list[Tree] = None
121                     ) -> int:
122     if data is None:
123         data = []  # to avoid mutable default argument
124     if isinstance(t, Leaf):
125         acc = acc + t.value
126         if data:  # non-empty
127             t1 = data.pop()
128             return total_acc_dcps(t1, acc, data)
129         else:  # data is empty
130             return acc
131     else:  # t is binary fork
132         data.append(t[1])  # postpone processing of t[1]
133         return total_acc_dcps(t[0], acc, data)
```

In hindsight, it is clear that in all these defunctionalized programs, parameter `data` serves as a custom stack, that stores exactly the information needed to support the (branching) recursion.

8. Conclusion

I hope to have created awareness of what I call the Olympiad trap, where the IOI is locked into its own contest format. One way of mitigating it, is to pay attention to interesting and challenging topics in informatics that fall outside the scope of the IOI. As an

example of such a topic, I explained how tail recursion can be transformed into a loop using thunking and a trampoline. And next, how Continuation Passing Style (CPS) can be used to transform arbitrary recursion into tail recursion. These ideas were discovered a long time ago and have become part of the CS folklore. For a history of continuations see (Reynolds, 1993), which traces it back to 1964, when Adriaan van Wijngaarden (designer of Algol 60 and Algol 68, and my father's promotor) first described it. The trampoline seems to have been introduced by Steele (1977). Gibbons (2022) is a modern exploration of CPS, accumulation, and defunctionalization.

The code for this article is available at (Verhoeff, 2025). It includes code that is instrumented to visualize how the stack is used, together with the output of that code. It also treats the example of flattening a binary leaf tree.

Acknowledgment

I would like to thank Ahto Truu and my colleague Berry Schoenmakers (one of the team members who made it to the ICPC World Finals in 1987) for helping me improve this article.

References

Gamma, E. and Helm, R. and Johnson, R. and Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

Gibbons, J. (2022). Continuation-Passing Style, Defunctionalization, Accumulations, and Associativity. *The Art, Science, and Engineering of Programming*, 6(2):7:1–7:28. https://doi.org/10.22152/programming-journal.org/2022/6/7

Reynolds, J.C. (1993). The Discoveries of Continuations. *LISP Symb. Comput.*, 6(3–4):233–248. https://doi.org/10.1007/BF01019459

Steele, Guy L. (1977). Debunking the "expensive procedure call" myth or, procedure call implementations considered harmful or, LAMBDA: The Ultimate GOTO. In: *ACM'77: Proceedings of the 1977 annual conference*. pp. 153–162. https://doi.org/10.1145/800179.810196

Verhoeff, T. (2018). A master class on recursion. In: *Adventures Between Lower Bounds and Higher Altitudes*. Lecture Notes in Computer Science Vol. 11011, Springer, pp. 610–633. https://doi.org/10.1007/978-3-319-98355-4_35

Verhoeff, T. (2021). Look Ma, backtracking without recursion, *IOI Journal 2021*, 15, 119–132. https://doi.org/10.15388/ioi.2021.10

Verhoeff, T. (2023). "Understanding and designing recursive functions via syntactic rewriting", *IOI Journal*, 17, 99–119. https://doi.org/10.15388/ioi.2023.08

Verhoeff, T. (2025). Git repository with source code for "The Olympiad Trap and an Old Trampoline". (Accessed 30 May 2025) https://gitlab.tue.nl/t-verhoeff-software/code-for-cps-with-trampoline

T. Verhoeff is Assistant Professor in Computer Science at Eindhoven University of Technology, where he works in the group Software Engineering & Technology. His research interests are support tools for verified software development, model driven engineering, and functional programming. He received the IOI Distinguished Service Award at IOI 2007 in Zagreb, Croatia, in particular for his role in setting up and maintaining a web archive of IOI-related material and facilities for communication in the IOI community, and in establishing, developing, chairing, and contributing to the IOI International Scientific Committee from 1999 until 2007.

Appendix A. Answers to Exercises

A.1. Function `total` with Accumulation Parameter

One of the recursive calls in the definition of `total` can be made into a tail call by introducing an accumulation parameter:

```
def total_acc(t: Tree, acc: int = 0) -> int:
    if isinstance(t, Leaf):
        return acc + t.value
    else:
        return total_acc(t[1],  # tail call
               total_acc(t[0], acc))  # not a tail call
```

The other recursive call is not a tail call, since more work is done after it returns.

A.2. Recursive Function where CPS Fails

Continuation Passing Style (CPS) is a powerful technique that can be used to transform any recursive function definition into a tail-recursive definition. But this does not mean that the resulting function is useful. In particular, it may not terminate. One example where this happens is the *fixpoint combinator*.

For given function f, we call x a fixpoint of f when $f(x) = x$. The fixpoint combinator fix returns a fixpoint of a given function f from functions to functions. That is, we have $f(fix(f)) = fix(f)$ as function. It can be defined in Python as `fix`:

```
type Endo[A] = Func[A, A]  # endo-functions on A

def fix[A, B](f: Endo[Func[A, B]]) -> Func[A, B]:
    return f(lambda a: fix(f)(a))
```

Function `fix` is recursive but not tail recursive.

Let's see an application. Function `tri` is a fixpoint of function `pre_tri`:

```
def pre_tri(g: Func[nat, int]) -> Func[nat, int]:
    return lambda n: 0 if n == 0 else n + g(n - 1)
```

Note that `pre_tri` abstracts from the recursive call, by making the function called there a function parameter. Hence, it is itself not recursive. Obviously, we have

```
    pre_tri(tri)(n) == tri(n)
```

That is, `tri` is a fixpoint of `pre_tri`. Hence, we can define `tri` by `fix(pre_tri)`.

Can we make `fix` tail recursive by applying CPS? Let's try:

```
def fix_cps[A, B](g: Endo[Func[A, B]],
                  cont: Endo[Func[A, B]] = id_
                  ) -> Func[A, B]:
    return fix_cps(g, lambda h: cont(g(lambda a: h(a))))
```

This definition is indeed tail recursive. But when you apply it to `pre_tri`, it does not terminate, because the argument `g` remains unchanged and never reaches a base case. The continuation `cont` keeps on growing. So, `fix_cps` is useless and certainly not equivalent to `fix`.

A.3. Defunctionalization of **`total_cps`**

The defunctionalization of `total_cps` can be better understood by analyzing the structure of its continuations. These can always be written as a composition:

- id, the initial continuation, is an empty composition;
- $\lambda\, tt_1 : cont\,(tt_0 + tt_1) = cont \circ (\lambda\, tt_1 : tt_0 + tt_1) = cont \circ (tt_0 +)$, where $(n +)$ abbreviates the function $\lambda\, x : n + x$;
- $\begin{aligned}&\lambda\, tt_0 : total\_cps\,(t_1, cont \circ (tt_0 +))\\ &= \lambda\, tt_0 : (cont \circ (tt_0 +))(total\,(t_1))\\ &= \lambda\, tt_0 : cont(tt_0 + total\,(t_1))\\ &= cont\,(\lambda\, tt_0 : tt_0 + total\,(t_1))\\ &= cont \circ (+ total\,(t_1)).\end{aligned}$

Thus, every continuation is some composition of $(tt_0 +)$ and $(+ total(t_1))$ for varying values of tt_0 and t_1. Compositions of these two kinds of functions commute (due to associativity of addition; see below), and therefore all functions of the form $(tt_0 +)$ can be moved together and merged into one such function, as we have seen before, which can then be defunctionalized into a single integer (`acc`). The composition of the other functions is defunctionalized into a list of `Tree`.

Here is a proof that the composition of $(n+)$ and $(+m)$ commutes because addition is associative. For integer k, we calculate:

$$\begin{aligned}&((n+) \circ (+m))(k)\\ &= (n+)((+m)(k))\\ &= (n+)(k + m)\\ &= n + (k + m)\\ &= (n + k) + m\\ &= (+m)(n + k)\\ &= (+m)((n+)(k))\\ &= ((+m) \circ (n+))(k)\end{aligned}$$

Such calculations with side-effect-free functions lies at the foundation of modern functional programming.

Olympiads in Informatics, 2025, Vol. 19, 159–175

DOI: 10.15388/ioi.2025.11

REPORTS

Informatics Curriculum and Programming Competitions: Azerbaijani Experience

Ismayil SADIGOV
Institute of Information Technology, Ministry of Science and Education of the Republic of Azerbaijan
B.Vahabzade str., 9A, AZ1141 Baku, Azerbaijan
e-mail: ismail.sadigov@gmail.com

Abstract. To what extent does the content of the International Olympiad in Informatics (IOI), as well as other programming competitions in which students participate, correspond to the curriculum of secondary school informatics? In other words, can a student who fully mastered the curriculum of informatics taught in secondary school succeed in programming competitions? This article reviews the history of informatics and Informatics Olympiads in Azerbaijan, how the curriculum of the subject has changed from 1985 to the present, and in particular, whether the space allocated to programming in these updates is sufficient. How to eliminate the existing inconsistency in the new curriculum is explained using the example of specific standards.

Keywords: informatics curriculum, programming competitions, International Olympiads in Informatics, IOI.

1. Introduction

Probably, the team leaders at the International Olympiads in Informatics (IOI), as well as the organizers of such competitions in their countries, are often asked the following question: to what extent does the content of these olympiads correspond to the curriculum of secondary school informatics? In other words, can a student who fully mastered the curriculum of informatics taught in secondary school succeed in programming competitions?

Many teachers still believe that the content of the IOI is not related to the real school curriculum in informatics. In this regard, it is interesting to analyze the extent to which the content of the IOI is included in the curriculum of studying informatics at the profile level. It is important to determine whether there is a real opportunity for students to realize their interest in informatics, including preparation for the olympiad, directly within the framework of the education they receive at school. (Kiryukhin, 2008)

The article will look for answers to these questions, take a brief look at the history of school informatics and Informatics Olympiads in Azerbaijan, and touch on how the integration of these two contents is implemented in the new informatics curriculum.

First, let's turn to official documents. According to the relevant paragraphs of the "Rules for the Organization and Conducting of Schoolchildren's Subject Olympiads", approved by the Order No. 1256 of the Minister of Education of the Republic of Azerbaijan dated December 12, 2014:

3.0.2. Subject Olympiads consist of 2 (two) stages – district (city) and republican stages. The republican stage, in turn, consists of two rounds – semi-finals and finals.

3.1.2. At the district (city) stage, questions are prepared by the jury of the republican subject olympiads in a closed test format in *accordance with the school program*.

3.2.4. In the semi-final round of the republican stage, questions are prepared by the jury of the republican subject olympiads in a closed test format in *accordance with the school program* and in a relatively difficult format.

However, this document does not contain any information about whether the questions for the final round of the republican stage of the subject Olympiads correspond to the school curriculum. (Rules for organizing and holding school subject Olympiads, 2014) Let us note here that in general, it is normal for the questions to deviate from the school curriculum for the final round, because the winners of this round are candidates who will represent the country at the IOI. IOI has an approved syllabus, and this syllabus must be taken into account. However, despite the fact that the Republican Olympiad in Informatics has been held for more than 35 years, it is not normal for these Olympiads to have no approved syllabus to this day.

The Law of the Republic of Azerbaijan on Education, approved on June 19, 2009, included an article (Article 26.5) on the *non-competitive admission* of winners of world subject Olympiads, high-level international competitions and competitions to higher education institutions in relevant specialties. Although this change paved the way for stimulating schoolchildren to participate in Olympiads and developing the teaching of informatics, it naturally did not satisfy specialists, since it covered only international competitions and contests. Under the influence of discussions opened both at the official level and on social networks under the leadership of Ramin Mahmudzadeh, a prominent scientist and educator who led the Azerbaijani team at the IOI (1993–2019), on June 12, 2018, amendments were made to "the Law of the Republic of Azerbaijan on Education" and Article 26.5 was amended as follows: "26.5. *Winners of international subject Olympiads in any specialty, winners of republican subject Olympiads, high-level inter-*

national competitions and contests are admitted to higher education institutions in the relevant specialties without competition".

Prize-winners of the final round of the Republican subject Olympiads are granted the right to admission to relevant specialties in higher education institutions without exams. Therefore, it is important to ensure that not only students from specialized schools but all students have the opportunity to participate in subject Olympiads, including the IOI. In other words, the informatics subject curriculum should correspond to the program of the Republican Olympiad in Informatics to a certain extent. However, of course, the extent of this correspondence will be the subject of serious analysis and discussion from time to time.

2. A Brief Look at the History of School Informatics

Informatics as an independent science began to take shape in the middle of the 20th century, primarily after the invention of electronic computers. In the 1970s, with the invention of microprocessors and the creation of microcomputers and personal computers based on them, the process of informatization of many areas of human activity accelerated. Naturally, this process also had its impact on the education system. There was a mass demand for computer literacy and information literacy among the population. In such conditions, the issue of including the subject of informatics in the curriculum of general education schools became relevant.

Introduction of a new subject to schools did not happen suddenly, this innovation became possible after some preparatory work was carried out for a while. Thus, in the early 1960s, experiments were conducted to teach students the elements of cybernetics. As a result of these experiments, in 1970, the Fundamentals of Cybernetics course was officially included in the list of optional courses of secondary general education schools. This 140-hour course (70 hours in grade 9, 70 hours in grade 10) was taught mainly in physics-mathematics-oriented schools until 1985.

On April 12, 1984, at a joint meeting of the Central Committee of the CPSU (Central Committee of the Communist Party of the Soviet Union) and the USSR Council of Ministers, Resolution No. 313 was adopted. In that resolution, the USSR Ministry of Education, the Academy of Pedagogical Sciences, the State Committee for Technical Vocational Education, and the Ministry of Higher and Secondary Specialized Education were instructed to:

- To organize the study of the basics of computing and electronic technology in the upper grades of general education schools, technical vocational schools, and secondary specialized schools, so that to taught students the skills of using computers and are armed with knowledge of the wide application of this technology in the national economy. For this purpose, special courses should be prepared for students, necessary textbooks, teaching aids, school and inter-school cabinets should be created, and the use of computer technology in basic institutions and other departments should be envisaged for educational purposes.

- To inform the Central Committee of the CPSU and the USSR Council of Ministers in 1986 about the psychological and pedagogical problems associated with the use of computers in the teaching process of general education schools.
- To create cabinets of computing electronics and microprocessor technology in 1986–1990.

In the organizational and methodological document "Main directions of reform in general education and vocational schools", prepared in 1984, one of the main directions of school reform was declared to be the elimination of general computer illiteracy of young people and the inclusion of the basics of informatics and computing technology in the educational process.

At the end of 1984, under the leadership of A.P. Yershov and V.M. Monakhov, the program for the subject "Fundamentals of Informatics and Computing Technology" began to be developed, and in mid-1985, this program was approved by the USSR Ministry of Education. Fundamentals of Informatics and Computing Technology was included in the curriculum of general education schools as a subject from September 1, 1985.

In 1986, the "machine version" of the first program for the Fundamentals of Informatics and Computing Technology course was published. The course, which was intended to be taught in the two upper grades (grades 9 and 10), took 102 hours. Several teaching aids were prepared in accordance with the content provided in the machine version.

Textbooks and teacher's aids prepared under the leadership of A.P. Yershov and V.M. Monakhov were published in Azerbaijani in 1985–1987. This two-part textbook included the following sections:

Part 1

1. Algorithms. Algorithmic language
2. Building algorithms for solving problems

Part 2

1. Computer structure
2. Introduction to programming
   - 2.1. Algorithmic language
   - 2.2. Rapira programming language
   - 2.3. BASIC programming language
3. The role of electronic computing machines in modern society. Development prospects of computing technology

After our country gained independence in 1991, this textbook, like a number of textbooks on other subjects, was replaced by a national textbook (authors: R.A. Aliyev, T.M. Aliyev, M.A. Salahlı).

In 1997, a new program on informatics for general education schools was developed and approved by the Ministry of Education of the Republic of Azerbaijan. On June 15, 1999, the President of the Republic of Azerbaijan signed the Decree No. 168 "On Approval of the Program of Reform in the Field of Education of the Republic of Azerbaijan". By the Order No. 280 of the Ministry of Education of the Republic of Azerbaijan dated April 3, 2000, the new basic curricula for general education schools were approved and this curriculum began to be implemented from the 2000–2001 academic year. Infor-

matics programs (5–11 grades) that are appropriate for the new conditions were developed for general education schools. (Mahmudzadeh, 2015)

According to the Education Sector Development Project implemented under the auspices of the World Bank, in accordance with the implementation plan for the Curriculum Reform sub-component within the Quality of General Education and Compliance with Real Needs component, a working group was established by the order of the Ministry of Education No. 87 dated 08.02.2006 for the purpose of preparing the informatics curriculum and relevant assessment standards, and this group prepared the informatics curriculum. According to this curriculum, teaching of informatics in general education schools was envisaged in all grades of the general education level, that is, from grade 1 to grade 11. In 2007–2017, textbook sets for grades 1–11 were prepared and published in accordance with the new curriculum (Fig. 1). (These textbooks, as well as their Russian and Georgian versions, can be accessed at `https://trims.edu.az/site/search.php?category_id=c-1&courses_id%5B%5D=5&book_type_id=&lang_id=&grif_no=&grif_date=&search=ok`).

Fig. 1. Textbooks of informatics currently in use.

Here, it is necessary to mention two State Programs related to ICT (Information and Communication Technologies) that have had an indirect rather than direct impact on the teaching of computer science in the Azerbaijani education system: State program on the provision of secondary schools with the information and communications technologies in the Azerbaijan Republic (2005–2007) and State Program on Informatization of Education System in the Azerbaijan Republic for 2008–2012. Within the framework of these State Programs, certain works have been carried out in areas such as ICT literacy of teachers and equipping schools with equipment.

The theoretical and applied content of computer science is rapidly developing and updating. One of the reasons why this update is necessary in Azerbaijan is that since 2023, applicants wishing to participate in the competition for computer science-oriented specialties, in addition to mathematics and physics, take an exam *in informatics* at entrance exams to higher and secondary specialized educational institutions (but from 1992 to 2022 they took an exam *in chemistry*?!).

3. Teaching Programming in the Primary and Secondary Education System of Azerbaijan

In this article, since the content line of the computer science curriculum "Algorithms and Programming" is of interest to us, let's look at the changes that have occurred in this direction since 1985.

The initial version of the Fundamentals of Computer Science and Computing course, which was supposed to be introduced in 1985, was not related to any specific programming language. Instead, it was proposed to use the abstract Russian-language algorithmic language (RAYA). This was essentially a symbolic version of flowcharts.

However, A.P. Ershov used the algorithmic language, as well as its machine-implemented adaptation, Rapira, and BASIC in his textbook. Then BASIC became the only school programming language, and this situation continued in Azerbaijan for almost 20 years. Despite the fact that BASIC was removed from the list of programming languages allowed at the International Olympiads in Informatics in 1998, it was certainly not normal for this language to be the only programming language taught in the general education system of Azerbaijan for the next 10 years.

After the start of teaching informatics from the 5th grade in the 2005–2006 academic year, Pascal language was included in the 9th grade textbooks in 2008. However, the "life" of this language in the Azerbaijani general education system was not as long as BASIC. Thus, as already mentioned, according to the curriculum reform that was implemented in the Azerbaijani general education system in 2008, starting the teaching of informatics from the 1st grade also required a new approach to teaching programming. More precisely, it required that the programming language to be taught in the upper grades be one of the languages used in international programming competitions. In the new curriculum, programming was planned to be learned from the 5th grade.

Discussions were held with leading informatics teachers for several years about which programming language to choose. During the discussions, taking into account the age of the students and the level of preparation of the teachers, it was concluded that the most suitable language for grades 5–7 is the Logo language. However, how to solve the problem of the commands in the program code being in Azerbaijani? In 2012, a group consisting of the authors of the informatics textbooks for secondary schools (Ramin Mahmudzade, Ismayil Sadigov, Naida Isayeva) and programmer Jamshid Nakhchivanski developed a programming environment called *ALPLogo* at the Baku publishing house and it is still used with great success in schools of Azerbaijan (Fig. 2). The study of the programming environment was included in the informatics textbooks for grades 5th, 6th and 7th and was distributed to schools of the republic free of charge. The main feature of the program is that the commands can be written in Azerbaijani, English and Russian. Since 2013, republic-wide competitions have been held annually among students in grades 5–7. The ALPLogo program can be accessed and downloaded at `http://www.informatik.az/index/proqram_t_minati/0-13`.

When the question of which programming language to use to continue the programming line from the 8th grade was resolved, Python was given priority. Thus, in the 2015–2016 academic year, the Python programming language was switched from the 8th grade (Table 1).

From the 2018–2019 academic year, the Ministry of Education, with the support of BP-Azerbaijan company, began implementing the project "Organization of Informatics-oriented Classes in Grades 10–11". Within the framework of the project, 50 informatics-oriented classes were completed in selected schools in Baku, covering more than 1,000 students. Within the framework of the project, new content standards and a curriculum were written by experts for informatics-oriented classes. Based on these, 4 new materials for teaching is completed pilot classes – informat-

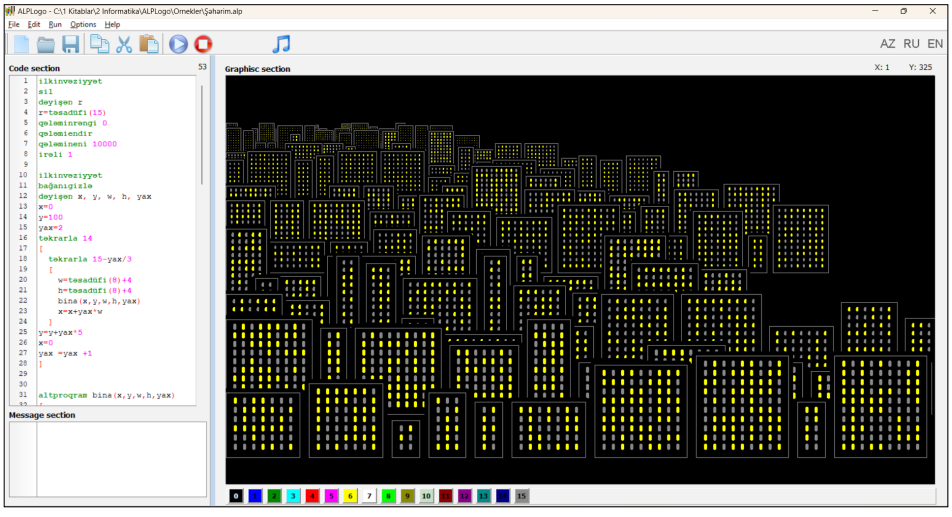


Fig. 2. Screenshot of the ALPLogo programming environment.

Table 1
Programming languages taught in Azerbaijani schools

Academic year	Programming language	Grades
1985/1986 – 1995/1996	BASIC	9-10
1996/1997 – 2008/2009	BASIC	10-11
2008/2009– 2014-2015	Pascal	9
2012-2013 until now	ALPLogo	5-7
2014-2015 untilnow	Python	8-11
Included in the new curriculum.	ALPLogo, Scratch	2-4
	Python	5-9
	C++	10-11

ics textbook and a methodological manual for teachers grades 10th and 11th – were prepared and made available to beneficiary schools. Unfortunately, this project did not produce the expected results. Thus, in 2020, only 30 percent of students graduating from informatics-oriented classes chose the ICT direction in higher education institutions – computer science or engineering, information technologies or security, robotics, aerospace engineering, informatics teaching or other relevant specialties. (`https://test.edugovaztest.cpanel.edu.az/az/news-and-updates/19555`)

The fact that the vast majority (70%) of graduates of informatics-oriented classes choose other specialties rather than ICT indicates either that the selection for those classes is not done correctly, or that the teaching is organized incorrectly (or at a low level).

Here, it is impossible not to mention the Digital Skills project, which has been implemented in the Azerbaijani education system since the 2017–2018 academic year and is gradually expanding its scope. According to the information provided on the official website of the Ministry of Science and Education (`https://edu.gov.az/az/programmes/reqemsal-bacariqlar_16387`), the main goal of this project is to ensure that students acquire in-depth ICT skills, achieve free and purposeful activity in the information space, and form themselves as competitive, logical and non-standard thinking individuals. The project is implemented by the Ministry of Science and Education and the international educational company “Algorithmica”. By improving the teaching of informatics, the project develops algorithmic thinking, logical thinking, project building skills in students, and teaches the basics of programming. The number of schools under the project has been increased to 762 in the 2024–2025 academic year, 230 of which were newly involved in the project. Currently, the project covers 6,000 teachers and 510,000 students in 57 cities and regions of the country. 1,370 students are studying in grades 10–11 with a focus on "Digital Skills".

The project teaches programming languages Scratch (grades 4–6) and Python (grades 7–11). Unfortunately, the project has not lived up to expectations. Although the number of teaching hours in the classes where it is taught is several times higher than the hours allocated to informatics, and sufficient funds have been spent, the project has not had any positive impact on the achievements of Azerbaijani students, neither

in international competitions nor in the latest international assessment (ICILS-2023). In our opinion, among the many reasons for the project's failure, despite the fact that a considerable amount of time has passed since its implementation, this subject, which is taught instead of informatics, still has neither a curriculum nor teaching materials.

4. Participation of Azerbaijani Students in Informatics Olympiads

The 1st All-Union Informatics Olympiad was held from April 13 to 20, 1988. Azerbaijani students did not participate in this competition, because at that time, informatics was not yet included in the list of subject Olympiads held in Azerbaijan among students. This event took place a little later.

In order to organize the Republican subject Olympiads of students in the 1988–1989 academic year, an order No. 329 was signed by the Ministry of Public Education of the Azerbaijan SSR on December 22, 1988. In accordance with the order, it was planned to hold the Olympiads in five subjects – Russian language, physics, chemistry, mathematics and *informatics* in the current academic year in three stages. The 1st Republican Informatics Olympiad was held on March 25–31, 1989. Shortly after this event, an Azerbaijani student also participated in the 2nd All-Union Informatics Olympiad held in Minsk, Belarus, from April 15 to 22 and was awarded 3rd place.

Another important event took place during this period. According to paragraph 2 of Section IV of the "Regulations on Subject Olympiads for Schoolchildren", approved by the order of the Ministry of Public Education of the Azerbaijan SSR dated October 10, 1989 No. 1100, starting from 1990, students who took 1st place in the school Olympiad are admitted to higher and secondary specialized educational institutions of the republic without exams in the established manner. In 1990, for the first time in Azerbaijan, six 10th-grade graduates who received a first-degree diploma in the final round of subject Olympiads were admitted to higher educational institutions in the relevant areas without exams, which had a positive effect on stimulating Olympiad winners in subsequent years. However, unfortunately, after the transition to a new system of admission to higher education institutions (centralized examinations by the State Customs Service), this rule (this concession) was abolished in 1993 (Decree of the President of the Republic of Azerbaijan No. 487). (dated July 27, 1993).

Over the past period, the scale of the Olympiads on informatics in Azerbaijan, although not at the desired level, has expanded from year to year. Now other competitions related to information and communication technologies are also held both among students and among teachers. However, the Republican Olympiad on Informatics for schoolchildren occupies a key place both in terms of importance and state support, because the winners of this Olympiad protect the honor of our republic at the International Olympiads on Informatics.

As for the participation of the Azerbaijani team in the IOI, Azerbaijan was invited to this Olympiad in 1993, but due to visa problems, it was unable to take part in the competitions held in Argentina that year. Our students first took part in the VI International Olympiad in Informatics, which was held in Sweden in 1994, and have since

been represented in all Olympiads held since that year. It should also be noted that the initiator of our country's participation in these Olympiads was Ramin Mahmudzadeh, who led the preparation of the Azerbaijani team for the competition until 2019. (However, during this period he was not listed as a team leader three times – at the 1994, 1995 and 1997 Olympiads.) Despite this, Ramin Mahmudzadeh was the leader of the Azerbaijani team at the International Olympiads in Informatics from 1994 to 2019. (Jalalli I., 2012)

Although about 10 years ago, the only international programming competition in which Azerbaijani schoolchildren participated was IOI, now they also participate in a number of other competitions throughout the year. Among these competitions, the following programming competitions are worth mentioning:

- International Zhautykov Olympiad, IzhO.
- European Junior Olympiad in Informatics, EJOI.
- European Girls' Olympiad in Informatics, EGOI.
- International School & Cup in Informatics "Junior", Cup ISIJ.
- InfO(1)Cup

The Bebras Computing Challenge (bebras.org) also plays a significant role in developing algorithmic thinking in schoolchildren and encouraging them to participate in informatics competitions.

It should be noted that the amendment made to the Law on Education on June 12, 2018 (*non-competitive admission of winners of republican subject Olympiads to higher education institutions in relevant specialties*) is already showing its positive effect. Thus, starting from 2020, Azerbaijani schoolchildren have won at least 1 bronze medal every year at the International Olympiad in Informatics. The results of our country's schoolchildren have also improved in other international programming competitions in informatics. We are confident that the positive effect of including informatics in entrance exams will also be evident in the near future.

At the end of this section, it is worth mentioning two recently published books that will have a great impact on the preparation of Azerbaijani-speaking schoolchildren for the Olympiads. Although the book Basics of Programming in C++ by Ramin Mahmudzadeh and Ismayil Jalali is intended primarily for students studying the basics of programming in higher and secondary specialized schools and who want to work in this field in the future, it can also be useful for teachers and students of computer science-oriented classes in general education schools, as well as anyone who wants to independently learn the basics of programming. The topics of computational geometry, combinatorics, long arithmetic given in a separate section of the book called Mathematics, as well as the Olympiad section, are intended for readers interested in programming competitions. (Mahmudzadeh, 2020).

In the second manual, Preparation for Programming Competitions, co-authored by Ismayil Jalalli (Sadigov) and Mikhail Medvedev, the book describes in detail the mechanism of holding programming olympiads and ways to prepare for them, analyzes the main topics and algorithms (Fig. 3). The manual is intended for students preparing for programming competitions, as well as their teachers. This book will also be useful for high schools with an emphasis on informatics. Students studying in information and

Fig. 3. Mahmudzadeh R., Jalalli I., Basics of Programming in C++. Baku, "Bakıneshr", 2020, 384 p.; Jalalli I., Medvedev M.. Preparation for Programming Competitions, Baku, "Bakıneshr", 2023, 512 p.

communication technologies at higher and secondary specialized educational institutions can also benefit from the textbook.

The materials of the book were selected based on the IOI syllabus. All examples in the book are written in C++. The program codes comply with the C++11 standard, which is allowed in most modern competitions. The Practice section at the end of each section of the book, which consists of 11 sections, provides problems related to the topic and their solution algorithms. In addition, to strengthen the mastery of the topic, additional problems from the Eolymp portal are recommended for independent work. This book, which is the third edition published in connection with the Azerbaijani programming olympiads, is dedicated to the dear memory of Ramin Mahmudzadeh, an outstanding scientist and educator who laid the foundation of competitive programming in Azerbaijan and led the Azerbaijani team in the international computer science olympiads. (Jalalli, 2023)

5. Features of the New Curriculum in informatics for General Education Institutions of the Republic of Azerbaijan

At a time when ICT are developing rapidly all over the world and the information society is being formed, there is a need to update the content of the subject of informatics, which is at the center of the theoretical and applied problems of these processes. This update is required by the recent work carried out in our country towards building an e-state, including the amendments made to the Resolution No. 103 of the Cabinet of Ministers of the Republic of Azerbaijan dated June 3, 2010, "On Approval of the State Standard and

Programs (Curriculums) of the General Education Level" on September 29, 2020. In this document, among the competencies formed in students at the general education level (the set of knowledge, skills, approaches and values that are acquired in the educational process and in life, necessary for any field of activity, as well as personal development, socialization and integration into society, employment, and lifelong education), two are directly related to the subject of informatics.

One of the main reasons for this update, as mentioned above, is the inclusion of informatics in the list of entrance exams to higher education institutions. Thus, according to the Resolution of the Cabinet of Ministers of the Republic of Azerbaijan dated March 12, 2022 "On Amendments to the Rules for Admission of Students to Higher Education Institutions of the Republic of Azerbaijan", approved by Resolution No. 39 of the Cabinet of Ministers of the Republic of Azerbaijan dated February 8, 2017, applicants who wish to participate in the competition for specialties included in the mathematics-informatics (MI) subgroup of the 1st specialty group from 2023 will take exams in physics, mathematics and informatics at the 2nd stage.

Unfortunately, the work on the preparation of the new curriculum for informatics began only in January 2024 and the process continued throughout the year. Currently, the document is awaiting approval. However, it should be noted that, considering that the current curriculum was developed in 2006–2007, textbooks based on the new curriculum will be put into use in the 2026/2027 academic year at best. This is, to put it mildly, not a good situation.

Based on the study and analysis of current world experience, the following content lines have been identified for the implementation of general learning outcomes of informatics:

- Data and information
- Hardware
- Software
- Algorithms and programming
- Information society

The content lines remain the same across all grades, but the content within each of these lines is intended to change from simple to complex, deepen, and expand. It should be noted that any concepts or skills included in the content of a subject may not be limited to the framework of just one content line.

During the preparation of this document, a number of related documents were analyzed, the experience gained during the implementation of the current curriculum (2013) and the Digital Skills project, as well as several international documents were taken as a basis. Among the international documents, the following should be specially noted:

- ICDL Workforce. Computer and Online Essentials; ICDL Workforce. Application Essentials.
- Computer Science Teachers Association (CSTA) K-12 Computer Science Standards.
- International Computer and Information Literacy Study (2023) Assessment Framework.

The **Data and Information** content line is divided into two main standards (content standards) and aims to ensure that students acquire the necessary knowledge and skills in *information processes and data sets*. The "International Computer and Information Literacy Study (2023) Assessment Framework" document was used in the development of the standards for this content line.
(`https://www.iea.nl/publications/icils-2023-assessment-framework`)

The **Hardware** content line is divided into two main standards (content standards) and requires students to acquire the necessary knowledge and skills in *information and communication technology* (computers, computer networks), including knowledge of *technical safety* rules when using these technologies. The "ICDL Workforce. Computer and Online Essentials" document was used in the development of standards for this content line. (`https://icdl.org/wp-content/uploads/2024/01/ICDL-Computer-Online-Essentials-Syllabus-1.0.pdf`)

The **Software** content line is also divided into two main standards (content standards), and it is intended that students acquire the necessary knowledge and skills in *system software* (operating system) and *application software* (text editors, spreadsheets, presentation programs, graphic editors and other programs). The "ICDL Workforce. Application Essentials" document was used in the development of the standards for this content line. (`https://icdl.org/wp-content/uploads/2024/01/ICDL-Application-Essentials-Syllabus-1.0.pdf`)

The **Algorithms and Programming** content line is divided into three main standards (content standards) and aims to provide students with the necessary knowledge and skills in *formalization and modeling*, *algorithmization*, and *programming*. The "Computer Science Teachers Association (CSTA) K-12 Computer Science Standards" document was used in the development of the standards for this content line. (`https://csteachers.org/k12standards/downloads/`)

The **Information Society** content line is divided into two main standards (content standards) and aims to provide students with the necessary knowledge and skills in the *informatization of society* and related *information security*. The "International Computer and Information Literacy Study (2023) Assessment Framework" document was used in the preparation of the standards for this content line. (`https://www.iea.nl/publications/icils-2023-assessment-framework`)

One of the features of the new computer science curriculum is the inclusion of a separate content standard in the form of "Demonstrates knowledge and skills in information security", in other words, information security issues are presented in the form of a separate content standard. This content standard covers all levels of education and is valid from 2nd to 11th grade. Since this content standard is new, let's dwell on it in more detail.

At the primary level, this content covers "the existence of threats to data in emergency situations such as fire, flood, war, earthquake and the importance of storing backup copies of important data elsewhere to prevent losses in such cases", "the existence

of attempts at unauthorized access to devices, programs and data, as well as methods to prevent them, the correct use of passwords and their protection", "potential threats to computers, computer networks and data – malware, cyber attacks, Internet fraud (spam)".

At the general secondary education level, the following issues are addressed: "biometric security techniques used to prevent unauthorized access to computers, computer networks, and data; malicious programs and the consequences they can cause on computers, ways to protect against those programs; the possibility of information posted on social networks being found and used by malicious individuals, not disclosing personal or sensitive information inappropriately when using communication tools; preventing the spread of inappropriate and false content; the purpose of encrypting information, the encryption process, various encryption methods, encrypting a given text with these methods, decrypting the encrypted text; criteria for evaluating the reliability of the website you visit."

At the upper secondary level, information is provided on the following topics: "intentional or unintentional attacks that may damage or otherwise endanger information and the systems that support it can be active or passive, intentional or unintentional, as well as direct or indirect; what tools are used to prevent hacker attacks; software copyright, software piracy and the damage it causes to the software industry".

Another feature of the new curriculum is the consideration of specialization at the secondary education level. Thus, it is envisaged that general issues of informatics will be taught at the primary and secondary education levels, and specific issues at the secondary education level.

As noted, since the winners of the final round of the Republican subject Olympiads gain the right to be admitted to the relevant specialties of higher education institutions without exams, not only students of specialized schools (classes), but all students should have the opportunity to participate in subject Olympiads, including the Informatics Olympiad. In other words, the informatics subject curriculum should correspond to the program of the Republican Olympiad in Informatics to a certain extent (both in terms of the chosen programming language and algorithms).

For the last two reasons, the importance of all content lines is taken into account in the curriculum, but taking into account the current and growing importance of algorithmization and programming, a larger space is allocated to substandards within this content line. Fig. 4 shows how the substandards are distributed across content lines. As can be seen from this table, approximately 44% of the total substandards belong to the algorithms and programming content line.

The Computer Science Teachers Association (CSTA) K-12 Computer Science Standards document was used to develop the standards for Content Line 4 (Algorithms and Programming), especially for the secondary level, and some of the standards were included in the new curriculum without any changes. Some of these standards are shown in Table 2. We believe that the implementation of these standards in the future will have a positive impact on both the teaching of programming in Azerbaijani schools and the achievements of students in programming olympiads.

Grade levels	Content lines					Total substandards
	1. Data, information	2. Hardware	3. Software	4. Algorithms and programming	5. Information society	
	Level 1 – primary education					
1	3	4	4	3	1	**15**
2	3	2	4	5	2	**16**
3	2	2	5	7	3	**19**
4	2	2	5	7	2	**18**
	Level 2 – lower secondary education					
5	3	2	6	7	3	**21**
6	2	2	6	9	3	**22**
7	3	2	5	9	2	**21**
8	3	3	3	10	2	**21**
9	2	2	2	9	3	**18**
	Level 3 – upper secondary education					
10	2	2	1	15	2	**22**
11	2	1	1	11	2	**17**
	27	**24**	**42**	**92**	**25**	**210**

Fig. 4. Distribution of substandards by content lines.

Finally, it should be noted that at a time when work on the curriculum was being finalized, the results of the 2023 international assessment on the International Computer and Information Literacy Study (ICILS) were announced. Since the results of Azerbaijani students were very poor, the standards of the aforementioned assessment were analyzed, and it was determined which standards were either absent from the current informatics curriculum at all or were present in higher grades. Based on this analysis, appropriate changes and additions were made to the new curriculum.

It is also necessary to note one important issue. The new curriculum in informatics requires updating the existing curricula of the specialties "Informatics Teacher" and "Mathematics and Informatics Teacher" of pedagogical universities. According to the information released by the Ministry of Science and Education, work has also begun in this direction (`https://edu.gov.az/az/news-and-updates/21899-1`). Thus, a Commission has been established in the Ministry of Science and Education to develop new educational programs for the bachelor's (basic (base higher) medical education) and master's levels of higher education. At the same time, 9 Working Groups, including the Educational Specialties Group, have begun their activities within the Commission. We hope that the new educational programs to be developed will meet the requirements of the modern era.

Table 2
CSTA Standards

CSTA K-12 Computer Science Standards	Informatics curriculum
3B-AP-10. Use and adapt classic algorithms to solve computational problems. (Examples could include sorting and searching.)	10-4.2.2. Explains sorting algorithms. (Sorting, selection sort algorithm, bubble sort algorithm)
3B-AP-11. Evaluate algorithms in terms of their efficiency, correctness, and clarity. (Examples could include sorting and searching.)	10-4.2.3. Evaluates algorithms. (Algorithm efficiency, algorithm correctness, algorithm clarity)
3B-AP-12. Compare and contrast fundamental data structures and their uses. (Examples could include strings, lists, arrays, stacks, and queues.)	10-4.3.3. Compares basic data structures and their uses. (Strings, lists, arrays, stacks, queues, dictionaries)
3B-AP-13. Illustrate the flow of execution of a recursive algorithm.	10-4.3.8. Uses recursion in the program.
3B-AP-19. Develop programs for multiple computing platforms. (Example platforms could include: computer desktop, web, or mobile.)	10-4.3.9. Develops programs for various computing platforms (desktop, web, mobile).
3B-AP-23 Evaluate key qualities of a program through a process such as a code review. (Examples of qualities could include correctness, usability, readability, efficiency, portability and scalability.)	11-4.3.7. Evaluate key qualities of a program through a process such as a code review. (Correctness, usability, readability, efficiency, portability and scalability)
3B-AP-24. Compare multiple programming languages and discuss how their features make them suitable for solving different types of problems. (Examples of features include blocks versus text, indentation versus curly braces, and high-level versus lowlevel.)	11-4.3.8. Compares several programming languages.

6. Conclusion

1. In an era when information and communication technologies are developing at a very fast pace and the information society is being formed, there is a need to frequently update the content of informatics, which is at the center of theoretical and applied problems of these processes.
2. The inclusion of informatics in the list of entrance exams for relevant specialties of higher education institutions from 2023 makes this update even more necessary.
3. Since the winners of the final round of the Republican Subject Olympiads currently have the right to be admitted to the relevant specialties of higher education institutions without exams, not only students of specialized schools (classes), but all students should have the opportunity to participate in subject Olympiads, including the Informatics Olympiad, and achieve success. For this, the informatics subject curriculum should meet the requirements of the Republican Olympiad in Informatics to a certain extent.
4. The types of questions, their topics and, most importantly, the training program (syllabus) for the final round of the Republican Olympiad in Informatics should be developed and approved soon.

5. Since we are talking about the curriculum for general education institutions, necessary changes should also be made in the direction of teacher training for the successful implementation of the new curriculum. That is, the curricula of the specialties "Informatics Teacher" and "Mathematics and Informatics Teacher" in higher education institutions should be revised.

References

Jalalli, I. (2012), *International Olympiads in Informatics: 1989–2011*. Bakuneshr, Baku (in Azerbaijani, Beynəlxalq İnformatika Olimpiadaları: 1989–2011).

Jalalli, I., Medvedev, M. (2023). *Preparation for Programming Competitions*. Bakuneshr, Baku (in Azerbaijani, Proqramlaşdırma yarışlarına hazırlıq).

Kiryukhin, V. (2008). *Informatics. Russian Olympiads. Issue 1*. Prosveschenie, Moscow (in Russian, Информатика. Всероссийские олимпиады. Вып. 1.)

Mahmudzadeh, R., Jalalli I., Aliyev, A. (2015). *Republican Informatics Olympiads: 1989–2014*. Bakuneshr, Baku (in Azerbaijani, Məktəblilərin İnformatika Olimpiadaları: 1989–2014).

Mahmudzadeh, R., Jalalli, I. (2020). *Basics of Programming in C++*. Bakuneshr, Baku (in Azerbaijani, C++ dilində proqramlaşdırmanın əsasları).

Rules for organizing and holding school subject Olympiads. (2014) (in Azerbaijani: Məktəblilərin fənn olimpiadalarının təşkili və keçirilməsi Qaydaları).

I. Sadigov, PhD of Technical Sciences. Worked in various government agencies in senior positions in the field of ICT (1983–2021). Works in the system of the Ministry of Science and Education of the Republic of Azerbaijan. Engaged in scientific activities at the Institute of Information Technologies. Author and manager of many projects in the field of automation and programming. Author of the "Explanatory Dictionary of Compter Science Terms (English-Russian-Turkish-Azerbaijani)" (2017) and a number of other books on computer science. Deputy leader of the Azerbaijani team at the International Olympiads in Informatics (2007–2018).

Olympiads in Informatics, 2025, Vol. 19, 177–188

DOI: 10.15388/ioi.2025.12

Policy Reforms of Informatics Education of Mongolia

Danzan TSEDEVSUREN
Mongolian National University of Education, Ulaanbaatar, Mongolia
e-mail: tsedevsuren@msue.edu.mn

Abstract. As informatics has been accepted by countries around the world as the basic form of literacy education in the 21st century, it has become universally taught as a compulsory subject in primary education. School informatics is expanding in terms of content since students are needed to obtain not only ICT application knowledge and skills but also digital communication-collaboration, ethics, basic knowledge of computers, and programming. Children are definitely born into and develop in a society where ICT is widely used and their future careers are dependent on technology. Informatics education plays an important role in meeting the needs of young people. As a result, the policy of informatics education should be dramatically changed by considering the current needs and demands. This article introduces research findings with regard to how changes in informatics education have been made in Mongolia and how those changes have integrated with the reform policy of foreign countries. We conducted factual research using the policies, standards, curricula, and textbooks of informatics education as well as research papers of leading researchers. Methods such as comparison of documents on education and reflection have also been incorporated into our research.

Keywords: school informatics education, digital literacy education, education of computer science.

1. Introduction

School informatics education was first addressed at the World Conference on Computer Education which was held in August 1970 in Amsterdam, Netherlands. The conference was organized by the International Federation for Information Processing (Sheepmaker & Zinn, 1970). Subsequently, in 1971, the Conference on Computer Science Education in Secondary Schools was held in Paris, France and hosted by the Centre for Educational Research and Innovation (CERI-OCDE, 1971). (Baron, Drot-Delange, Grandbastien, & Tort, 2014). As a result, the foundation of teaching informatics in secondary education had been laid which led to informatics education being officially included in the system of primary education by the world's leading countries in the 1980s. (Carr & O'Brien, 2010). Informatics was viewed as significantly important in order to learn the methods of solving mathematics and algebra problems as well as to increase the development of new ideas and motivation. (Atchison, 1973).

Although only 40 years have passed since the computer science has been taught in countries around the world, ICT, as the basis of social development, has become the basic requirement for knowledge and skills that young people have to acquire. In terms of the informatics of secondary schools and informatics technology education, the policies and programs which had been accepted in the world hadn't been developed yet. (Dagienė & Stupurienė, 2016).

It is common for countries to implement informatics education using terms such as information communication technology or digital literacy education, (Guerra, 2012). Depending on the term, the subject content is different.

The book entitled "Rethinking Education: Towards a Global Common Good?", published by UNESCO in 2017, emphasized that in the technological era there is a challenge for our time which defines an illiterate person as someone who cannot use ICT, not someone who cannot read, write, and solve math problems. The amendments of two categories, "ICT Social and Ethic Issue" and "Career and ICT," to the application content of the Information and Communication Technology in Secondary Education approved by UNESCO in 2000 demonstrate that the proper use of ICT in all fields of society is significant. (UNESCO, 2000).

Many countries in the world are paying attention to the digital literacy education of young people. The digital literacy education covers a broad content of not only having ICT application knowledge and skills but also having an ability to reliably and responsibly collaborate with the public in social network or on the internet and make a contribution to the development of ICT. In addition to the ICT application, the issues of information ethics, digital citizen, and digital communication occupies an important position to the content of the digital literacy education. However, ICT with literacy is about ethically using the application knowledge and skills of ICT in creative activities, generating innovation, and applying for communication and collaboration. (Literacy with Information, Communication Technology. Across the Curriculum, 2012).

One of the impacts of ICT education is the use of ICT as a tool in the learning process. Although the policy of introducing ICT in training activities has been carried out in many countries, school children lack the knowledge and skills to use it as a learning tool in secondary education. This is due to the fact that the majority of countries have until ICT is taught as a compulsory subject in senior grades.

Regardless of whether school informatics education is implemented choosing one of the three previously mentioned forms, there is a universal principle that considers algorithm and basics of programming as its core concept. This is probably related to the history of informatics education starting with algorithm, programming, and codification. On the other hand, as students study algorithms and programming through contemporary informatics education, they are learning to acquire the basic skills of creating software based on mobile devices and web design. (Benaya, Dagiene & Gal-Ezer, 2015; Csernoch, Biro, Math, & Abari, 2015).

The Informatics Olympiad and competitions on computing and programming have been organized at regional, national, and international levels. It is one of the reasons why the subject of informatics has to be studied in secondary education.

2. Changes for the Policy of ICT Education

The tendency for changes in the policies of ICT education can be seen in international educational programs including "Public Informatics" which was developed by Common European Countries and "Common Computer Science" which was developed by Common Association for Computing Machinery-ACM.

According to the policy of "Public Informatics" developed by the European Union, each of the European countries are required to promote the growth of digital knowledge and skills in young people because of the basic changes of society. One of the ways of implementing the policy is to teach the subject of coding at all secondary schools in the European countries. (Caspersen, Gal-Ezer, McGettrick, & Nardelli, 2018, p.6). The strategy of the policy of "Common Computer Science" defines the objectives of developing informatics education as below.

- In the secondary education system, all students have the right to get continuous education on informatics. The subject of informatics is taught starting with primary education.
- Informatics curricula should reflect the scientific and constructive nature of the discipline, and be seen as fundamental to twenty-first century education by all stakeholders (including educators, pupils and their parents).
- Informatics courses must be compulsory and recognized by each country's educational system as being at least on a par with courses in STEM (Science, Technology, Engineering and Mathematics) disciplines. In particular they must attract equivalent credit, e.g. for the purposes of university entrance. (Caspersen, Gal-Ezer, McGettrick, & Nardelli, 2018, p. 8).

Computer science, the discipline that makes the use of computers possible, has driven innovation in every industry and field of study, from anthropology to zoology. Computer science is also powering approaches to many of our world's toughest challenges. Computer Science has become the foundation of creating new ideas and products in all fields of science as well as every industry to enable our world to conquer the tough challenges we are facing. The following research in the United States shows that the public demand for computer science education is high due to the fact that computing is an integral part of our world. (K-12 Computer Science Framework, 2016, p. 11).

- Most parents want their child's school to offer computer science. (Google & Gallup, 2015).
- Most Americans believe computer science is as important to learn as reading, writing, and math. (Horizon Media, 2015).
- Many of today's students will be using computer science in their future careers, not only in science, technology, engineering, and mathematics (STEM) fields but also in non-STEM fields. (Uddin, S., Imam, T. & Mozumdar, M., 2021).
- Not all young people in the United States have the opportunity to study computer science. The number of schools in the United States which teach computer science or programming effectively is fewer. (Google & Gallup, 2015). Although students in the United States have access to computers, the acquisition of knowl-

edge of computer science is often limited for marginalized students who are facing educational inequities. (Google & Gallup, 2015b)

Although ICT and computer skills are very important for the learning process of young people, in most countries, they have to wait until it is offered in high school. However, computer science has become an important tool for them to create world-renowned innovations. To address this issue, The Association for Computing Machinery (ACM) developed the educational program "Computer Science Education for All" in 2016. The K-12 Computer Science Framework illuminates the big ideas of computer science through a lens of concepts (i.e., what students should know) and practices (i.e., what students should do). The core concepts of the framework represent major content areas in the field of computer science. The core practices represent the behaviors that computationally literate students use to fully engage with the core concepts of computer science. (K-12 Computer Science Framework, 2016, p. 2–3).

K12: The Computer Science Framework is a powerful stimulus of knowledge and skills that creates equality and participation for every student, helps them to solve real-life problems, and discover many areas. The practice of computational thinking, such as summarizing, modeling, and analyzing, intersects with computer science concepts such as algorithm-programming, automation, and data representation. Computer science is more than just coding. It covers comprehensive aspects such as physical systems and networks, data collection, storage and analysis, and the social impact of computation. Students' programming knowledge and skills are valuable and contribute to important intellectual development. (Papert, 2000, p. 728).

Highly developed and developing countries around the world have recognized that basic education of computer science plays an important role in preparing the world's capable citizens to realize the nature of the information society, the fundamentals of social change, and to create changes and innovations in the future. (Fig. 1). Based on this, young people should have a computer literacy at high level.

- Have a computer literacy in primary education.
- Acquire application knowledge and skills of information technology in primary education.
- In secondary education, school informatics education has been planned and implemented to provide learners (or students) with an opportunity to acquire the basic skills of using ICT tools because there is a need for learners (or students)

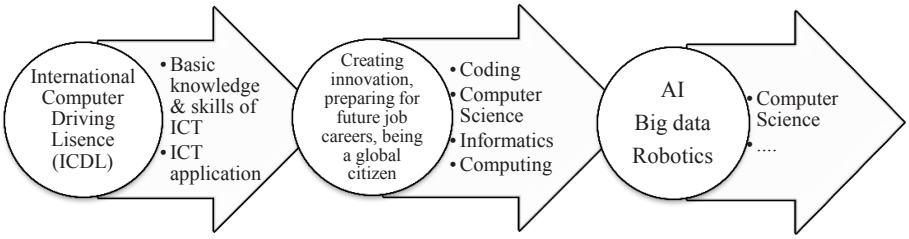


Fig. 1. Mapping on Informatics Education Policy Reforms.

to study computer science as an elective course in integration with their needs, interests, future job choices, and to have access to ongoing education. (Tsedevsuren, 2018, x. 53). How the educational content of computer science changes in the future is not clear in the same way it is impossible to foresee social development.

The questions "*How has computer science been taught in our country?*" and "*How are the global trends in ICT for education and educational principals and approaches to ICT reflected in Informatics Education Policy Reforms?*" have been main concerns. The solutions to these questions have been taken into consideration in this article.

3. Research Methodology

Based on legal documents, we utilized main research methodologies, such as observation, comparison of documents on education, and reflection in this study. The study of Informatics Education Policy Reforms in Mongolia has been conducted using the major documents of informatics education and other sources listed below.

- Curriculum on Informatics and Computing Technique Basis, 1988.
- Standard of Informatics Education (Primary and Secondary Education), 2004.
- National Program on Informatics Education of Primary and Secondary Education, 2004.
- Core Curriculum on Informatics Technology of Primary and Secondary Education, 2014–2015.
- Core Curriculum on Informatics Technology of Primary and Secondary Education, 2015.

4. Research Outcome

Starting in 1988, an informatics course titled "Informatics and Computer Basics" was taught in the 9th and 10th grades at secondary schools. Written by Russian textbook writers led by A.P. Ershov and translated by Ts. Amarsanaa, B. Narankhuu, D. Garmaa, and Ya. Senemeder into Mongolian, this computer science textbook was used in secondary schools. Additionally, in 1988, a methodological guideline of the textbook "Informatics and Computer Basics – 9," translated by B. Jargal, Yu, Namsrai (should this be Yu Namsrai without the comma in between), and L. Choijoovanchig, provided methods for carrying out programs for secondary school teachers. The importance of studying "Informatics and Computer Basics" in the senior grades of secondary schools was defined as "Discovering the actual capacity of the computer for learners is considerably important to extend not only human recognition of the world view but also their intellectual and cognitive skills. The study of informatics is the scientific background for developing students' perceptions of the possibility of automating various human activities based on algorithms."

The objective of the subject was defined by the followings: (Myagmar, 1987, p. 4):

- Systematize distinct areas of study of algorithm within the subject of algebra offered in an eight-year schooling system.
- Have students acquire basic skills of algorithm.
- Give students an imagination on automated performance of algorithm.
- Solve mathematics problems using modern computers.
- Introduce the applications and features algorithm, programming, computer and
- automation to provide an orientation to technical jobs.
- Introduce the basis of modern computing techniques based on examples of general principles of micro-computer.
- Introduce basic concepts and methods of developing programs using a language for programming.
- Give students an understanding of the stages of solving math problems using a computer.
- Introduce the main framework of using computing technique and its role in social development.

The informatics course was very important in order to realize the impact and development of electronic computing machines in future society and give children and young people an understanding of computer development and its use as well as algorithm programming at an early age. (Myagmar, 1987, p. 4)

The objective of the subject "Informatics and Computer Basics" which was defined as "to demonstrate the importance of algorithms, programming, and computers in industries of modern society" (Myagmar, 1987, p. 3) is still viewed as an important concept because it was the first to recognize the nature of the information society. Computer science plays an important role in improving the subject quality of natural sciences including mathematics, algebra, and geometry. Therefore, the curriculum and textbook content provided not only examples of how to solve the problems with those subjects but also examples of the algorithm.

The next major change in information education was made in 2004. The name of the subject was changed to "Informatics" and the content standards were implemented at the primary and secondary education.

The standard defined not only the basis of informatics science but also the role it plays in the development of other sciences and life of modern society. The objective of informatics education was defined in the standard as " … to prepare citizens with competency who can meet the needs of informatics education and who have acquired the skill and ethics using the knowledge and information needed for citizens living and working in a knowledge-based society". The standard demonstrates that informatics education has significance theoretically and practically.

To develop the theoretical and practical skills student need to acquire through the subject of computer science, the content of the standard is designed with following five areas: "Information", "Computer", "Algorithm", "Model", and "Information Technology". (Uyanga, Tsogtbaatar, & Choijoovanchig, 2005, p. 5). Since textbooks for all lev-

els of secondary education had been developed in accordance with the requirements of ICT education standards and used in secondary schools, ICT education has become an integral part of the education system.

To support the standard implementation of informatics in training, the "National Program for Informatics Education" was developed in 2009–2012 (Appendix 2, the Order No. 301 of the Minister of Education, Culture and Science, 2009). The national curriculum is based on the concept of the International Computer Driving License (ICDL), which laid the foundation for making informatics subject usage-based. "National Program for Informatics Education" was developed with the following contents "A1. Basic Concepts of Information and Communication Technology (ICT)", A2. Using a computer and working with files", "A3. Text information processing", "A4. Spreadsheet data processing", "A5. Image processing", "A6. Demonstration and processing of audio and video information", "A7. Working with databases", "A8. Use of website and internet" based on the approach that "Not every student becomes a computer scientist, but every student should become an citizen with information technology education in the future." (Otgonnaran, Tsogtbaatar, Altantuya, & Tsedevsuren, 2009, pp. 14–18). The application-based content, which was implemented with the national program, is fully reflected in the core curriculum developed in 2015. Name of the subject was changed to "Information Technology" because of the changes in core curriculum based on the principles of application knowledge and skills to be used in other ICT subjects and solving life issues (Core Curriculum Handbook for Basic Education, 2015, pp. 48–49). As a result, the content of algorithms, programming, and modeling, which is the basics of computer science, has been dramatically reduced and the content of ICT applications has been increased. (Tsedevsuren, 2016, pp. 72–73). The focus on providing students with the opportunity to learn through practical examples of how ICT can be used to solve other academic and life problems was the important change to increase practical aspects of the subject.

In high school, however, informatics has been included in the compulsory elective section of the curriculum. Consequently, students have had an opportunity to choose one of three areas, such as "Software," "Hardware," and "Multimedia," but local schools, located in rural areas and, especially, in counties, are facing difficulties implementing the program because of the insufficient ICT learning environment. Local schools have common issues such as teacher knowledge and skills, computer laboratory equipment capacity, networking, and Internet availability (Tsedevsuren, 2018a, p. 78). According to the changes in informatics education, the elective subject "Information Technology" consists of two selective areas such as "Information and Communication Technology" and "Programming" to be studied in 10th grade for 1 hour, 11th grade for 3 hours, and 12th grade for 5 hours a week, respectively. (Core Curriculum for Upper Secondary Education. Information Technology, 2016, p. 25). These elective subjects had been offered to a few schools in Khentii, Khuvsgul provinces and the city of Ulaanbaatar due to the lack of teacher skills, learning environment, inadequate laboratories, and lack of interest in elective subjects because informatics is not taken as general entrance examinations (Tsedevsuren, 2018a, pp. 77–78).

Because of the latest curriculum refinement, information technology has been redeveloped into a compulsory subject to study in senior grades of secondary schools and it is being implemented from the 2018–2019 academic year. The revised curriculum was developed on the basis of the principle that "ICT literacy has to be studied in primary education (Primary Education Curriculum, 2019, pp. 49–50), while a basic knowledge of computer science, all types of information processing including school subjects and useful information, problem-solving skills, and consumer culture and ethics in information society has to be acquired in upper secondary education". (Curriculum for Secondary Education, 2019, p. 44).

Table 1 shows the chronology of Informatics Education Policy Reforms of Mongolia.

Table 1

Policy Reforms and Features of Informatics Education (Computer Science)

Policy documents, years published	Principle			Features and explanations
	Content	Methodology	Assessment	
Curriculum on Informatics, Computer Basics, 1988	Based on knowledge and understanding: Information processing, Computer and algorithm, algorithm design, composing algorithm, basics of programming, computer development	The teacher should introduce and explain the main concepts and approaches in detail, recognize the main ideas, and focus on them.	The knowledge and skills to be acquired by students was defined in the curriculum. Students will be assessed by doing exercises, such as solving mathematics and algorithm problems.	Informatics was studied in 9–10th grades for 34 hours of classes per semester respectively. It was available to study 68 hours of classes in 10th grade at a school equipped with computers.
Standard on Informatics Education, 2004	Knowledge and skills necessary for students to acquire were divided into five categories: "Information," "Computer," "Algorithm," "Model," and "Information Technology." The categories were based on the development of students' talents and abilities in addition to features of their ages, thinking, interests, and needs. (Chimedlkham, Uyanga, Tsogtbaatar, & Choijoovanchig, 2005, p. 5).	By considering students' ages, personalities, health situations, and learning abilities, a methodology for supporting their participation and development as well as preparing them to become well-balanced and responsible citizens will be used adhering to flexible, humanitarian, usage based principles.	Students' progress is assessed according to the children's personalities and features of their ages and thinking. *Features:* self and collective assessment. Principle: Assessment should be open and fair. A variety of assessment methods were used.	Informatics will be studied at all levels of education: 35 hours of classes in primary education, 140 hours of classes in secondary education, 175 hours of classes in high school.

Continued on next page

Table 1 – continued from previous page

Policy documents, years published	Principle			Features and explanations
	Content	Methodology	Assessment	
Curriculum for the National Program for Informatics Education, 2009–2012	Usage-based: "A1. Basic Concepts of ICT," "A2. Using a Computer and Working with Files," "A3. Text Information Processing," "A4. Spreadsheet Data Processing," "A5. Image Processing," "A6. Demonstration and Processing of Audio and Video Information," "A7. Working with Databases," "A8. Use of Websites and the Internet."	The methodology is based on learning to obtain information through research, collect the information, process it using technology, solve problems and make decisions as to how to find and collect information, process it using technology, and determine the ways in which information came to be, can be improved, and used. Methodology: Knowledge and understanding, activities of creation of knowledge, communication skills, and creative skills, critical thinking skills, decision making, and problem-solving skills.	Understanding of basic and application knowledge will be equally assessed throughout education. As students advance into senior grades, especially high school, individual and teamwork will be evaluated according to certain criteria. (based on Bloom's taxonomy).	
Core Curriculum on Informatics Education, 2015–2016	Primary: Content based on ICT literacy and its use. Senior: The elective course is based on the principles of applying the application knowledge and skills needed to solve other areas of ICT and life problems. The content includes 5 sections: - Software, - Hardware, - Multimedia, - Information and Communication Technology, - Programming. (Curriculum on Information Technology, 2016, pp. 48–49).	Methodology, such as acquiring application skills and knowledge by applying ICT as a means of communication and using it independently and collaboratively as a learning tool for other subjects to process information and solve the problems, will be used.	Basic knowledge of ICT and computer science. Knowledge and skills of processing information and solving the problems using ICT will be assessed.	Informatics is not studied at an elementary school. The basic content is studied at secondary school for 140 hours of classes while the senior graders of high school study it as an elective course for 105 hours choosing one of the three areas "Software", "Hardware", and "Multimedia". It is available to study 35 hours in 10th grade, 105 hours in 11th grade, 175 hours in 12th grade in the fields of "Information and Communication Technology" and "Programming".

Continued on next page

Table 1 – continued from previous page

Policy documents, years published	Principle			Features and explanations
	Content	Methodology	Assessment	
Curriculum on Information Technology, 2019	Intermediate: By studying the application- based content, students will acquire ICT literacy and information processing skills. Senior: The content aims to enable students to have basic knowledge of computer science, process useful information (including information on other subjects), acquire problem-solving skills, and instill in consumers' culture and the ethics to properly use the information in society. (Information Technology Program, 2019, pp. 49–50).	Creative methods will be used to allow students to learn a computer language and how to use ICT as a tool for information processing, problem-solving, and continuing learning.	The following knowledge and skills will be assessed: - process the information using ICT, problem solving; - understand ICT and computer science terminology, use information in an ethical and cultural manner, create, interpret, share and collaborate with others depending on their circumstances and needs.	Informatics is not studied in elementary school, however, the basic content is studied in secondary school for 140 class hours. Informatics is available to 10th graders for 35 hours, 11th graders for 105 hours, and 175 hours for 12th graders in the fields of "Information and Communication Technology" and "Programming." Seniors in high school can also take Informatics as an elective course for 105 hours, choosing one of three areas: "Software," "Hardware," or "Multimedia."

5. Conclusion

Informatics education plays an important role in acquiring basic knowledge and skills for citizens living and working in the information society. Therefore, many countries around the world have included the subject of informatics in secondary education. The content of this subject has been shifted from the use of information technology to the basic content of computer science, which is foundation of technological society since computer science is crucial for preparing future citizens who have realized the nature of social changes, can create innovations, and have an ethical use of communication.

In 1988, Mongolia introduced Informatics into secondary education. It was the right decision at the right time. Informatics has been taught in secondary education for 30 years which is a relatively short period. However, its content and methodology has been changed five times and the subject has been renamed as "Information Technology" based on content application.

The renaming of the subject and the reduction of the basic content of computer science was a step backwards from the reform of the content of global informatics education. Despite the shortcomings, Mongolia is one of the few countries with experience in teaching informatics as an independent subject in secondary education, starting from primary education.

Depending on the changes in the information society and the needs of future citizens who will be living and working in the 21st century, the following policy changes need to be made:

- Change the name "Information Technology" to "Informatics" or "Computer Science.
- Include the basic content of computer science that supports the skills needed in the 21st century, such as understanding the basics of uncertain social development and helping to create innovation in the subject content.
- Increase the teaching hours of informatics.

References

Atchison, W.F. (1973). The Impact of Computer Science Education on the Curriculum. *The Mathematics Teacher*, 66(1), 7–83. http://www.jstor.org/stable/27959160

Baron, G.L., Drot-Delange, B., Grandbastien, M., & Tort, F. (2014). Computer science education in French secondary schools: Historical and didactical perspectives. *ACM Transactions on Computing Education (TOCE)*, 14(2), 1–27. https://doi.org/10.1145/2602486

Benaya, T., Dagiene, V., & Gal-Ezer, J. (2015, July). CS High School Curriculum–A Tale of Two Countries. In: *IFIP TC3 Working Conference "A New Culture of Learning: Computing and next Generations"*, 17–28. Retrieved from http://www.ifip2015.mii.vu.lt/

Carr, J.A., & O'Brien, N.P. (2010). Policy implications of education informatics. *Teachers College Record*, 112(10), 2703–2716. https://doi.org/10.1177/016146811011201006

Caspersen, M.E., Gal-Ezer, J., McGettrick, A., & Nardelli, E. (2018). *Informatics for All the Strategy*. ACM. https://doi.org/10.1145/3185594

Chimedlkham, Ts., Uyanga, S., Tsogtbaatar, D., & Choijoovanchig, L. (2005). *Primary and Secondary Education. Informatics Education Standard MNS 5420-7: 2004*. Ulaanbaatar: National Center for Standardization and Metrology.

College Board. (2016). AP program participation and performance data 2015 [Data file]. Retrieved from https://research.collegeboard.org/programs/ap/data/participation/ap-2015

Core curriculum handout for secondary education (2015). Ulaanbaatar. x. 48–55

Core curriculum on secondary education. Improved second edition (2019). Ulaanbaatar. pp. 48–56

Core curriculum on secondary education. Information technology (2016). Ulaanbaatar.

Csernoch, M., Biró, P., Máth, J., & Abari, K. (2015). Testing algorithmic skills in traditional and non-traditional programming environments. *Informatics in Education*, 14(2), 175–197. DOI: 10.15388/infedu.2015.11

Curriculum on lower secondary education, Second publication. (2019). Улаанбаатар. pp. 48–56

Curriculum on secondary education, improved second edition (2019). Ulaanbaatar. pp. 44–56

Ershov, A., Monakhov, V., & Beshenkov, S. (1987). *Fundamentals of Informatics and Computing*. Translated by Amarsanaa Ts., Narankhuu B. Ulaanbaatar: State Publishing House.

Ershov, A., Monakhov, V., & Kuznetsov, A. (1988). *Recommendation for Teaching the Informatics and Computer Basics*, Translated by B.Jargal *et al.* Ulaanbaatar: State Publishing House.

Google, & Gallup. (2015). Searching for computer science: Access and barriers in US K-12 education. Retrieved from https://services.google.com/fh/files/misc/searching-for-computer-science_report.pdf

Guerra, V., Kuhnt, B., & Blöchliger, I. (2012). Informatics at school-worldwide. An international exploratory study about informatics as a subject at different school levels. https://www.researchgate.net/publication/275031370_Informatics_at_school_-_worldwide_An_international_exploratory_study_about_informatics_as_a_subject_at_different_school_levels

Horizon Media. (2015, October 5). Horizon Media study reveals Americans prioritize STEM subjects over the arts; science is "cool," coding is new literacy. PR Newswire. Retrieved from https://www.prnewswire.com/news-releases/horizon-media-study-reveals-americans-prioritize-stem-subjects-over-the-arts-science-is-cool-coding-is-new-literacy-300154137.html

K-12 Computer Science Framework. (2016). Retrieved from http://www.k12cs.org.

Literacy with Information, Communication Technology. Across the Curriculum. (2012). Northwest Territories, Canada. Retrieved from https://www.ece.gov.nt.ca/en/node/774

Myagmar, Sh. (1987). *Curriculum on Basics of Computer Science and Computing*, ed. D. Shagdar. Ulaanbaatar: State Publishing House.

Otgonnaran, O., Tsogtbaatar, D., Altantuya, Yu., & Tsedevsuren, D. (2009). *Informatics Curriculum*. Ulaanbaatar: Bit Press LLC.

Papert, S. (2000). What's the big idea? Toward a pedagogy of idea power. *IBM Systems Journal*, 39 (3/4), 720–729. DOI: 10.1147/sj.393.0720

Tsedevsuren, D. (2016). Comparative study of secondary education curriculum on informatics. In: *Proceedings of Research Papers of School of Mathematics and Natural Sciences of the MNUE*, No 2, pp. 69–75.

Tsedevsuren, D. (2018). Comparative study on secondary education in information technology. *Lavai Research Journal*, 20, pp. 53–58. https://lavai.msue.edu.mn/index.php?role=no&link=articles715

Tsedevsuren, D. (2018a). The issues of the curriculum implementation of information technology. In: *Proceedings of Research Papers of School of Mathematics and Natural Sciences of the MNUE*, No 4, pp. 69–81.

Tsedevsuren, D. (2019). Results of the ICT readiness survey for secondary school teachers. *Lavai Research Journal*, 22, pp. 130–140. https://lavai.msue.edu.mn/index.php?role=no&link=articles776

Uddin, S., Imam, T. & Mozumdar, M. (2021) Research interdisciplinarity: STEM versus non-STEM. *Scientometrics*, 126, 603–618. https://doi.org/10.1007/s11192-020-03750-9

UNESCO. (2000). *Information & Communication Technology in Secondary Education*.

UNESCO. (2017). *Rethinking Education. Is Education a Public Good?* UNESCO. www.unesco.ch/wp-content/uploads/2017/01/Education-transforms-lives.pdf

D. Tsedevsuren (https://orcid.org/0000-0002-8431-1986) is Professor at Mongolian National University of Education. He is a PhD in ICT and Educational Studies, and he is currently working as a chairman of the Mongolian Informatics Association from 2014. His research interests include CS and Informatics for secondary education, ICT in education, theory and methodology of digital learning and electronic learning content development. He is a deputy leader or team leader of Mongolian teams at IOIs in 2002, 2009, 2013, 2015, 2017, 2018 and 2020–2024. He is the author of over 20 books of the olympiad in informatics and preparations for the olympiads.

Olympiads in Informatics, 2025, Vol. 19, 189–208

DOI: 10.15388/ioi.2025.13

School Startup in Olympiad in Informatics

Marina S. TSVETKOVA, Vladimir M. KIRYUKHIN
Russian Academy of Natural History, Russian Federation, Moscow, 105037, box 47
e-mail: ms-tsv@mail.ru, vkiryukh@gmail.com

Abstract. The article describes the modern content of an advanced school course in Informatics (K 7-9 and K 10-11), corresponding to the new educational standard implemented in Russia since 2023. This content is in full compliance with the IOI Syllabus. This allows us to build effective preparation programs for various Olympiads in informatics, including the International Olympiad in Informatics (IOI).

Keywords: school education, curriculum in Informatics, Olympiad in Informatics, Modern Content of an advanced school course in Informatics (K 7-9 and K 10-11), competencies of Olympiad in Informatics

1. Introduction

As a rule, updating educational standards for schools in any country is aimed at developing the content of education, updating it considering the development of science and technology. This is especially important for the subject of informatics, which is the basis for the digitalization of professions, and reflects the digital transformation of the economy and science in the country.

In Russia, school informatics has already gone through several stages of development: it started in schools in 1985 as a subject based on Algorithmics. At the suggestion of the Academy of Sciences, in the 90s the subject was supplemented with programming and a section on information technology, which was facilitated by the appearance of personal computers in schools. An advanced course in informatics, filled with topics on the theoretical foundations of informatics, was included in schools for grades 10–11 since 2005 (Kiryukhin, 2010), but mainly reflected topics for the Unified State Exam for admission to universities. It is important that this exam is an elective exam for senior schoolchildren in technological or physical and mathematical profiles, in which the course in informatics was studied as an advanced one.

Education in informatics remained within the framework of the basic program for schoolchildren in grades 7/8–9 until 2022. Only electives and additional education courses could compensate for the needs of children interested in informatics. In 2022, as part of the new federal educational standard (Educational Standard, 2022) advanced

courses in informatics were included in the course in informatics in both basic and high school at the federal level, and in them, updated sections of theoretical foundations of informatics, algorithms and programming, which reflect the topics of Olympiad in Informatics.

Based on the new educational standard, since September 2023 federal programs for basic general education (Curriculum Advanced Informatics K7-9, 2023) and for high school (Curriculum Advanced Informatics K10-11, 2023) have been in effect. This allowed schoolchildren involved in Olympiad preparation to choose to study an advanced course in informatics from the 7th grade (14 years old) free of charge using the appropriate textbooks provided by the state.

Currently, the Russian School Olympiad in Informatics is held considering the current advanced standard for the subject, and the winners and prize winners of the Olympiad finals receive the right to enter without exams for free education at any university in the country in the Olympiad profile and successfully participate in international Olympiads. It cannot be denied that this standard and federal programs in informatics for basic general education and high school have played a significant role in this.

Below are sections of the advanced course in informatics for primary and high school, the sections of which fully reflect the content of Olympiad training in informatics and can determine work with schoolchildren in preparation for various levels Olympiads, including international Olympiads and the IOI.

2. Theoretical Foundations of Informatics. Information Structures (Graphs, Trees)

This section covers topics of graph descriptions and graph problems, using the example of a game tree.

K7-9	**K10-11**
Graph. Vertex, edge, path. Directed and undirected graphs. Length (weight) of an edge. Graph weight matrix. Length of a path between graph vertices. Finding the optimal path in a graph. Starting vertex (source) and ending vertex (sink) in a directed graph. Calculating the number of paths in a directed acyclic graph. **Tree.** Root, vertex (node), leaf, edge (arc) of a tree. Height of a tree. Subtree. Examples of using trees. Enumeration of options using a tree.	**Graphs.** Basic concepts. Types of graphs. Description of graphs using adjacency matrices, weight matrices, adjacency lists. Solving algorithmic problems related to graph analysis (constructing an optimal path between graph vertices, determining the number of different paths between vertices of a directed acyclic graph). **Trees.** Binary tree. Search trees. Methods of tree traversal. Representation of arithmetic expressions as a tree. Discrete games of two players with complete information. Construction of a tree of enumeration of variants; description of the game strategy in tabular form. Winning and losing positions. Winning strategies.

3. Theoretical Foundations of Informatics. Data Coding, Data Compression

This section shows the basics of discrete data representation and theoretical approaches to assessing the amount of information. For senior school, the topic of number representation in computer memory is considered, as well as methods of applying coding to data compression, encryption during archiving or transmission over communication channels.

Theoretical foundations of informatics Information is one of the main concepts of modern science. Discreteness of data. The ability to describe continuous objects and processes using discrete data. Information processes are processes associated with storage, transformation and transmission of data. **Symbol. Alphabet. Alphabet power** Diversity of languages and alphabets. Natural and formal languages. Alphabet of texts in Russian. Binary alphabet. Number of different words (code combinations) of fixed length in binary alphabet. Transformation of any alphabet to binary. Number of different words of fixed length in alphabet of certain power. Coding of symbols of one alphabet using code words in another alphabet; code table, decoding. **Binary code** Representation of data in a computer as texts in a binary alphabet. Information volume of data. Bit is the minimum unit of information amount – binary digit. Byte, kilobyte, megabyte, gigabyte	**Theoretical foundations of informatics** Information, data and knowledge. Information processes in nature, technology and society. Continuous and discrete quantities and signals. The need for discretization of information intended for storage, transmission and processing in digital systems. **Binary coding** Uniform and non-uniform codes. Decoding messages written using non-uniform codes. Fano condition. Construction of uniquely decodable codes using a tree. Al. A. Markov's graph. Units of measurement of information quantity. Alphabetical approach to assessing information quantity. Theoretical approaches to assessing the amount of information. Law of additivity of information. Hartley's formula. Information and probability. Shannon's formula.
Positional and non-positional number systems Alphabet. Base. Expanded form of writing a number. Conversion of numbers written in other number systems to the decimal system. Roman numeral system. Binary number system. Conversion of natural numbers to the binary number system. Octal number system. Conversion of numbers from the octal system to the binary and decimal systems and back. Hexadecimal number system. Conversion of numbers from the hexadecimal system to the binary, octal and decimal systems and back.	**Number systems** Expanded notation of integers and fractions in the positional number system. Properties of positional notation of a number: number of digits in the notation, divisibility of the number by the base of the number system. Algorithm for converting an integer from the P-number system to the decimal number system. Algorithm for converting a finite P-number fraction to the decimal number system. Algorithm for converting an integer from the decimal number system to the P-number system. Converting a finite decimal fraction to the P-number system.

Arithmetic operations in the binary number system. Representation of integers in P-number systems. Arithmetic operations in P-number systems.	Binary, octal and hexadecimal number systems, the relationship between them. Arithmetic operations in positional number systems. *Balanced ternary number system. Binary-decimal number system*.
	Representation of integers in computer memory Limited range of numbers with limited number of digits. Overflow of the bit grid. Unsigned and signed data. Signed bit. Binary complement code of negative numbers. Bitwise logical operations. Logical, arithmetic and cyclic shifts. Encryption using the bitwise operation «exclusive OR». **Representation of real numbers in computer memory** Significant part and order of number. Range of values of real numbers. Problems of storing real numbers related to limitation of the number of digits. Carrying out operations with real numbers, accumulation of errors during calculations.
Text encoding Uniform code. Non-uniform code. ASCII encoding. Eight-bit encodings. The concept of UNICODE encodings. Decoding messages using uniform and non-uniform codes. Information volume of text. **Color coding** Color models. RGB, CMYK, HSL models. Coding depth. Palette. Raster and vector representation of images. Pixel. Evaluation of the information volume of graphic data for a raster image. **Sound coding** Bit depth and sampling frequency. Number of recording channels. Evaluation of the information volume of sound files.	**Text encoding** ASCII encoding. Single-byte encodings. UNICODE standard. UTF-8 encoding. Determining the information volume of text messages. **Image encoding** Estimating the information volume of graphic data at a given resolution and color encoding depth. Color models. Vector encoding. Graphic file formats. Three-dimensional graphics. Fractal graphics. **Sound encoding** Estimating the information volume of audio data at a given sampling frequency and encoding bit depth.
Data transfer rate Units of data transfer rate. Data distortion during transmission.	**Data transfer rate** Transfer time dependence on the information volume of data and characteristics of the communication channel. Causes of data transfer errors. Codes that allow detecting and correcting data transfer errors. Hamming's distance. Bit repetition coding. Hamming's co-des.

	Data compression algorithms RLE algorithm. Huffman algorithm. LZW algorithm. Lossy data compression algorithms. Reducing the color coding depth. Basic ideas of JPEG, MP3 compression algorithms. **Data Encryption** Symmetric and asymmetric ciphers. Simple substitution ciphers. Caesar's cipher. Vigenere's cipher. RSA encryption algorithm. *Steganography*.

4. Theoretical Foundations of Computer Science. Logical Algebra

This section, in addition to the theoretical foundations of the logic algebra, which are important for constructing algorithms, shows the logical foundations of a computer and engineering approaches to constructing circuits on logical elements.

Logical statements Logical meanings of statements. Elementary and compound statements. Logical operations: «AND» (conjunction, logical multiplication), «or» (disjunction, logical addition), «NOT» (logical negation), «exclusive or» (addition mo-dulo 2), «implication» (consequence), «equi-valence» (logical equivalence). Priority of logical operations. Determining the truth of a compound statement given the known truth values of the elementary statements it contains. **Logical expressions** Rules for writing logical expressions. Construction of truth tables of logical expressions. Simplification of logical expressions. Laws of Boolean algebra. Construction of logical expressions according to the truth table. Logical elements. Introduction to the logical foundations of a computer. Adder.	**Algebra of logic** Concept of proposition. Propositional forms (predicates). Quantifiers of existence and universality. **Logical operations** Truth tables. Logical expressions. Logical identities. Proof of logical identities using truth tables. Logical operations and operations on sets. **Laws of algebra of logic** Equivalent transformations of logical expressions. Logical equations and systems of equations. **Logical functions** Dependence of the number of possible logical functions on the number of arguments. Complete systems of logical functions. Canonical forms of logical expressions. Perfect disjunctive and conjunctive normal forms, algorithms for constructing them using a truth table. **Logical elements in a computer** Trigger. Adder. Multi-bit adder. Construction of circuits on logical elements using a given logical expression. Writing a logical expression using a logical circuit. *Microcircuits and their production technology.*

5. Mathematical Modeling, Computer Modeling

This is a key section of the advanced course in informatics, which allows you to apply the acquired knowledge on all theoretical topics of informatics and connect mathematical modeling with approaches to computer modeling of information systems.

Control Signal. Feedback. Receiving signals from digital sensors (touch, distance, light, sound, etc.). Examples of using the feedback principle in control systems of technical devices, including robotics. Examples of robotic systems (traffic control system in a transport system, welding line in a car factory, automated control of home heating, autonomous vehicle control system, etc.). **Model** Problems solved with the help of modeling. Classifications of models. Material (natural) and information models. Continuous and discrete models. Simulation models. Game models. Evaluation of the adequacy of the model to the modeled object and the purposes of modeling. **The concept of a mathematical model** Problems solved using mathematical (computer) modeling. The difference between a mathematical model and a natural model and a verbal (literary) description of an object. Stages of computer modeling: problem statement, construction of a mathematical model, software implementation, testing, conducting a computer experiment, analysis of its results, refinement of the model.	**Systems** System components and their interaction. System effect. Management as an information process. Feedback. **Models and modeling** The purpose of modeling. Adequacy of the model to the modeled object or process, the purposes of modeling. Formalization of applied problems. Presentation of modeling results in a form convenient for human perception. Graphical presentation of data (diagrams, tables, graphs). **Stages of computer-mathematical modeling** Problem statement, model development, model testing, computer experiment, analysis of modeling results. Discretization in mathematical modeling of continuous processes. Modeling of motion. Modeling of biological systems. Mathematical models in economics. Computational experiments with models. Computer modeling of control systems. Processing of experimental results. Least squares method. Estimation of numerical parameters of modeled objects and processes. Restoration of dependencies based on experimental results. Probabilistic models. Monte Carlo methods. Simulation modeling. Queueing systems.

6. Algorithms and Programming

The section includes topics of Olympiad informatics. At the choice of schoolchildren, it is envisaged to study two languages from those recorded in the standard for in-depth study (Python, Java, C++, C#).

We will consider in more detail the contents of the section in comparison with K7-9 and K10-11 by years of study.

Algorithms and programming / K7

The concept of an algorithm. Algorithm executors. An algorithm as a plan for managing an executor.

Algorithm properties. Methods of recording an algorithm (verbal, in the form of a flowchart, program).

Algorithmic constructions. The "following" construction. Linear algorithm. Limitations of linear algorithms: the impossibility of foreseeing the dependence of the sequence of actions performed on the initial data.

The "branching" construction: complete and incomplete forms. Fulfillment and non-fulfillment of a condition (truth and falsity of a statement). Simple and compound conditions.

The "repetition" construction: cycles with a given number of repetitions, with a condition of fulfillment, with a cycle variable.

Auxiliary algorithms. Using parameters to change the results of auxiliary algorithms.

Analysis of algorithms for performers.

Execution of algorithms manually and on a computer. Syntax and logical errors. Failures.

The coordinate system in computer graphics. Changing the color of a pixel.

Graphic primitives: segment, rectangle, circle. Contour properties (color, line thickness) and fill. Constructing images from graphic primitives.

Using cycles to construct images. Hatching a closed area of simple shape (rectangle, triangle with base parallel to coordinate axis).

Principles of animation. Using animation to simulate object movement. Controlling animation using the keyboard.

Algorithms and programming / K8

Programming language (Python, C++, Java, C#). Programming system: program text editor, translator, debugger.

Variable: type, name, value. Integer, real and symbolic variables.

Assignment operator. Arithmetic expressions and the order of their calculation. Operations with integers: integer division, remainder from division. Checking the divisibility of one integer by another.

Operations with real numbers. Built-in functions.

Random (pseudo-random) numbers.

Algorithms and programming / K10

Determining the possible results of the simplest executor control algorithms and computational algorithms. Determining the initial data with which the algorithm can give the required result.

Stages of solving problems on a computer. Instrumental tools: translator, debugger, profiler. Compilation and interpretation of programs. Virtual machines.

Integrated development environment. Methods of debugging programs. Using trace tables. Debugging output. Step-by-step execution of a program. Breakpoints. Viewing variable values.

Programming language (Python, Java, C++, C#). Data types: integer, real, symbolic, logical. Branching. Complex conditions. Cycles with a condition. Cycles by variable. Interchangeability of different types of cycles. Cycle invariant. Compiling a cycle using a predetermined cycle invariant.

Documenting programs. Using comments. Preparing a program description and user instructions.

Algorithms for processing natural numbers written in positional number systems: breaking a number into individual digits; finding the sum and product of digits; finding the maximum (minimum) digit.

Finding all prime numbers in a given range. Representing a number as a set of prime factors. Fast exponentiation algorithm.

Processing data stored in files. Text and binary files. File variables (file pointers). Reading from a file. Writing to a file.

Splitting a task into subtasks. Subroutines (procedures and functions). Recursion. Recursive objects (fractals). Recursive procedures and functions. Using a stack to organize recursive calls.

Using the standard library of the programming language. Connecting third-party subroutine libraries. Modular principle of program construction.

Numerical methods. Exact and approximate solutions to problems. Numerical methods for solving equations: enumeration method, bisection method. Approximate calculation of curve lengths. Calculating the areas of figures

Branching. Compound conditions (writing logical expressions in the programming language being studied). Finding the minimum and maximum of two, three, and four numbers. Solving a quadratic equation with real roots. Logical variables. Dialog debugging of programs: step-by-step execution, viewing values, debug output, choosing a breakpoint. Cycle with a condition. Euclidean algorithm for finding the greatest common divisor of two natural numbers. Splitting a natural number in a positional system with a base less than or equal to 10 into separate digits. Decomposition of a natural number into prime factors. Cycle with variable. Algorithm for checking whether a natural number is prime. Analysis of algorithms. Determining possible results of an algorithm for a given set of input data; determining possible input data leading to a given result. Processing data flow: calculating the number, sum, arithmetic mean, minimum and maximum values of sequence elements that satisfy a given condition. Processing symbolic data. Symbolic (string) variables. Character-by-character string processing. Counting the frequency of a symbol in a string. Built-in functions for string processing. Tabular quantities (arrays). One-dimensional arrays. Compiling and debugging programs implementing typical algorithms for processing one-dimensional numerical arrays in one of the programming languages (Python, C++, Java, C#): filling a numerical array with random numbers, according to a formula or by entering numbers; finding the sum of the array elements; linear search for a given value in the array; counting the array elements that satisfy a given condition; finding the minimum (maximum) element of the array. Concept of algorithm complexity.	using numerical methods (rectangle method, trapezoid method). Finding the maximum (minimum) of a function of one variable using the bisection method. Processing symbolic data. Built-in functions of the programming language for processing symbolic strings. Algorithms for processing symbolic strings: counting the number of occurrences of a symbol in a string; splitting a string into words by whitespace characters; searching for a substring within a given string; replacing a found substring with another string. Generating all words in a certain alphabet that satisfy specified restrictions. Converting a number to a symbolic string and back. Arrays and sequences of numbers. Calculation of generalized characteristics of array elements or a numerical sequence (sum, product, arithmetic mean, minimum and maximum elements; number of elements satisfying a given condition). Linear search for a given value in an array. Sorting a one-dimensional array. Simple sorting methods (bubble sort, selection sort, insertion sort). Merge sort. Quick sorting of an array (QuickSort algorithm). Binary search in a sorted array. Two-dimensional arrays (matrices). Algorithms for processing two-dimensional arrays: filling a two-dimensional numerical array according to given rules; searching for an element in a two-dimensional array; calculating the maximum (minimum) and sum of elements of a two-dimensional array; rearranging rows and columns of a two-dimensional array. *Development of programs for solving simple data analysis tasks (data cleaning, classification, deviation analysis).*
Algorithms and programming / K9 Splitting a task into subtasks. Auxiliary algorithms (subroutines, procedures, functions). Parameters as a means of changing the results of a subroutine. Function result. Logical functions.	**Algorithms and programming /K 11** **Formalization of the concept of an algorithm**. Turing machine as a universal model of computation. Church-Turing thesis. Post machine. Normal Markov algorithms. Algorithmically unsolvable problems. Halt

Recursion. Recursive subroutines (procedures, functions). Recursion termination condition (base cases). Using recursion to enumerate options. Sorting arrays. Built-in sorting capabilities of the selected programming language. Sorting by several criteria (levels). Binary search in an ordered array. Two-dimensional arrays (matrices). Basic algorithms for processing two-dimensional arrays (matrices): filling a two-dimensional array with random numbers and using formulas; calculating the sum of elements, the minimum and maximum of a row, column, range; searching for a given value. Dynamic programming. Problems solved using dynamic programming: calculating functions specified by a recurrence formula; counting the number of options, choosing the optimal solution.	problem. Impossibility of automatic debugging programs. Estimation of the complexity of computations. Operating time and amount of memory used, their dependence on the size of the initial data. Estimation of the asymptotic complexity of algorithms. Algorithms of polynomial complexity. Enumeration algorithms. Examples of various algorithms for solving one problem that have different complexity. Search for prime numbers in a given range using the "sieve of Eratosthenes" algorithm. Multi-digit integers, problems of long arithmetic. Dictionaries (associative arrays, mappings). Hash tables. Building an alpha-frequency dictionary for a given text. Natural language text analysis. Extracting sequences by pattern. Regular expressions. Frequency analysis. Stacks.Queues. Graph algorithms. Building a minimum spanning tree of a weighted connected undirected graph. Depth-first graph traversal. Breadth-first graph traversal. The number of different paths between vertices of a directed acyclic graph. Dijkstra's algorithm. Floyd-Warshall algorithm. Trees. Implementation of a tree using reference structures. Binary trees. Construction of a tree for a given arithmetic expression. Recursive algorithms for tree traversal. Using a stack and a queue to traverse a tree. Dynamic programming is a method for solving problems with saving intermediate results. Problems solved using dynamic programming: calculating recursive functions, counting the number of variants, optimization problems. The concept of object-oriented programming. Overview of programming languages. The concept of programming paradigms. *Learning a second programming language*

7. Digital Literacy

It is important that for both basic and advanced study of the subject, the standard has a special section that is mandatory for all schoolchildren to study at both the basic and advanced levels, this is Digital Literacy. The section includes topics that cover computer literacy (computer), communication literacy (network), information literacy (software) and information security.

Digital Literacy K7-9	Digital Literacy K10-11
Safety precautions and rules for working on a computer A computer is a universal computing device that operates according to a program. Types of computers: personal computers, embedded computers, supercomputers. Mobile devices. The main components of a computer and their purpose. Processor. RAM and long-term memory. Input and output devices. Touch input, mobile device sensors, biometric authentication tools. The history of the development of computers and software. Generations of computers. Modern trends in the development of computers. Supercomputers. Parallel computing. Personal computer. Processor and its characteristics (clock frequency, bit depth). RAM. Long-term memory. Input and output devices. The volume of stored data (computer RAM, hard drive and solid-state drive, smartphone ROM) and access speed for different types of media.	**Safety and hygiene requirements when working with computers and other components of the digital environment** Principles of operation of computers and computer systems. Von Neumann architecture. Harvard architecture. Automatic execution of a program by a processor. RAM, read-only and long-term memory. Data exchange using buses. External device controllers. Direct memory access. Main trends in the development of computer technology. Parallel computing. Multiprocessor systems. Supercomputers. Distributed computing systems and big data processing. Mobile digital devices and their role in communications. Embedded computers. Microcontrollers. Robotic production.
Software Computer software. Application software. System software. Programming systems. Files and folders (directories). File types. File properties. Typical file sizes of different types (text page, e-book, photo, song recording, video clip, full-length film). Principles of file system construction. Full file name (folder, directory). Path to file (folder, directory). File manager. Working with files and folders (directories): creating, copying, moving, renaming and deleting files and folders (directories). Searching for files. Data archiving. Using archiving programs. Computer viruses and other malware. Programs for protection against viruses.	**Software** Software for computers and computer systems. Types of software and their purpose. Features of mobile device software. Parallel programming. System software. Operating systems. Utilities. Device drivers. Installation and uninstallation of software. File systems. Principles of placement and naming of files in long-term memory. Templates for describing file groups. Organization of personal information archive. Backup. Password protection of the archive. Antivirus programs.

Network Connecting computers into a network. Internet. Web page, website. Structure of web resource addresses. Browser. Search engines. Searching for information by keywords and images. Reliability of information obtained from the Internet. Modern Internet communications services Global Internet. IP addresses of nodes. Network data storage. Methods of individual and collective placement of new information on the Internet. Big data (Internet data, in particular social network data).	**Network** Principles of construction and hardware components of computer networks. Network protocols. Internet. Addressing on the Internet. TCP/IP stack protocols. Domain Name System. Dividing an IP network into subnets using subnet masks. Network administration. Obtaining data on the computer's network settings. Checking for a connection with a network node. Determining the route of packet movement. Principles of construction and hardware components of computer networks. Network protocols. Internet. Addressing on the Internet. TCP/IP stack protocols. Domain Name System. Dividing an IP network into subnets using subnet masks. Network administration. Obtaining data on the computer's network settings. Checking for a connection with a network node. Determining the route of packet movement.
Network. Activities on the Internet Internet services: communication services (mail service, video conferencing, etc.); reference services (maps, schedules, etc.), search services, software update services. Government services. Cloud data storage. Collaborative document development tools (online offices). Software as a web service: online text and graphic editors, software development environments.	**Network. Types of activities on the Internet** Internet services. Geoinformation systems. Real-time geolocation services (location of mobile phones, determination of highway congestion, etc.); Internet trade; booking tickets and hotels, etc. State electronic services and services. Social networks – organization of collective interaction and data exchange. Network etiquette: rules of conduct in cyberspace. The problem of authenticity of received information. Open educational resources.
Network. Web page development HTML language. Web page structure. Page title and body. Logical markup: headings, paragraphs. Development of pages containing images, lists and hyperlinks.	**Network. Internet applications** The concept of the server and client parts of a site. Client-server technology, its advantages and disadvantages. Basics of HTML and Cascading Style Sheets (CSS). JavaScript scripts. Forms on a web page. Hosting websites. Hosting service. Uploading files to a site.
Information security Legal protection of programs and data. Free and shareware programs. Free software. Network etiquette, basic norms of information ethics and law when working on the Internet. Strategies for safe behavior on the Internet.	**Information security** Russian Federation legislation in the field of software. Licensing of software and digital resources. Proprietary and free software. Commercial and non-commercial use of software and digital resources. Responsibility

The concept of information security. Information security threats when working on the global network and methods of counteracting them. Rules for secure authentication. Protecting personal information on the Internet. Safe strategies for behavior on the Internet. Prevention of involvement in destructive and criminal forms of network activity (cyberbullying, phishing, etc.).	established by Russian legislation for the illegal use of software and digital resources. Technogenic and economic threats are associated with the use of ICT. General problems of information protection and information security. Information security tools in computers, computer networks and automated information systems. Legal support for information security. Electronic digital signature, certified sites and documents. Preventing unauthorized access to personal confidential information stored on a personal computer, mobile devices. *Malicious software and ways to combat it.*

8. Information Technology

This section includes traditional topics about tools for computer processing of various types of data, but it is supplemented with topics on Artificial Intelligence, Data Analysis (Big DATA).

The section is aimed at testing schoolchildren of various competencies based on modern technologies for processing graphics, text, multimedia, spreadsheets and databases – the topics of this section. It is important that artificial intelligence methods have already been implemented in all data processing systems (search services, editors), which must be practically mastered when studying this section. These are voice search, translators, proofreaders, interlinear translators, handwritten-to-printed text converters, automatic settings for templates when working with graphics, texts, in spreadsheets, presentations, video editors and databases.

К 7-9	**К 10-11**
Information technology Text documents and their structural elements (page, paragraph, line, word, symbol). A word processor is a tool for creating, editing and formatting texts. Typing rules. Text editing. Character properties. Font. Font types (chopped, serif, monospaced). Bold and italic. Paragraph properties: borders, paragraph indentation, spacing, alignment. Style formatting. Structuring information using lists and tables. Multilevel lists. Adding tables to text documents. Inserting images into text documents. Wrapping text around images. Including dia-	**Information technology** Word processor. Editing and formatting. Spell and grammar check. Search and autoreplace tools in a word processor. Using styles. Structured text documents. Footnotes, table of contents. Collaborative work with documents. Review tools in word processors. Cloud services. Business correspondence. Abstract. Rules for citing sources and formatting bibliographic references. Formatting a list of references. Standards of bibliographic descriptions. Introduction to computer typesetting of text. Technical means of text input. Specialized means of editing mathematical texts.

grams and formulas in a text document. Page settings, page numbers. Adding headers, footers, and links to a document. Spell checking. Hyphenation. Voice input. Optical character recognition. Computer translation. Using Internet services for text processing.	
Introduction to graphic editors Raster images. Using graphic primitives. Editing operations for graphic objects, including digital photographs: resizing, cropping, rotating, mirroring, working with areas (selecting, copying, filling with color), color, brightness, and contrast correction. Vector graphics. Creating vector images using built-in tools in a word processor or other programs (applications). Adding vector images to documents. Preparing multimedia presentations. Slide. Adding text and images to a slide. Working with multiple slides. Adding audiovisual data to a slide. Animation. Hyperlinks.	**Graphic editor** Input of images using various digital devices (digital cameras and microscopes, video cameras, scanners, etc.). Resolution. Cropping. Perspective correction. Histogram. Levels correction, color correction. Desaturation of color images. Retouching. Working with areas. Filters. Multilayer images. Text layers. Layer mask. Channels. Saving a selection. Preparing illustrations for websites. Animated images. Vector graphics. Primitives. Changing the order of elements. Alignment, distribution. Grouping. Curves. Vector drawing formats. Using contours. Vectorization of raster images. Principles of constructing and editing three-dimensional models. Grid models. Materials. Modeling light sources. Cameras. Additive technologies (3D printers). The concept of virtual reality and augmented reality.
Spreadsheets understanding Data types in spreadsheet cells. Editing and formatting tables. Built-in functions for finding maximum, minimum, sum, and average. Sorting and filtering data in a selected range. Creating charts (histogram, pie chart, scatter chart). Selecting a chart type. Conversion of formulas when copying. Relative, absolute and mixed addressing. Conditional calculations in spreadsheets. Summation and counting of values that meet a given condition. Processing large data sets. Dynamic programming in spreadsheets. Numerical modeling in spreadsheets. Numerical solution of equations using parameter selection. Finding the optimal solution.	**Data analysis** The main tasks of data analysis: forecasting, classification, clustering, deviation analysis. The sequence of solving data analysis problems: collecting primary data, cleaning and assessing the quality of data, selecting and/or building a model, transforming data, visualizing data, interpreting results. Software and Internet services for processing and presenting data. Big data. Machine learning**.** *Intelligent data analysis*. **Data analysis using spreadsheets** Calculating the sum, arithmetic mean, and the largest (smallest) value of a range. Calculating the correlation coefficient of two data series. Plotting bar, line, and pie charts. Plotting function graphs. Selecting a trend line, solving forecasting problems. Numerical solution of equations using parameter selection.

	Optimization as a search for the best solution under given conditions. Objective function, constraints. Local and global minimum of the objective function. Solving optimization problems using spreadsheets.
Tabular models Table as a representation of a relation. Databases. Selecting rows in a table that satisfy a given condition. Developing a single-table database. Compiling database queries using a visual editor.	**Information technology** Tabular (relational) databases. A table is a representation of information about similar objects. Field, record. Table key. Working with a ready-made database. Populating a database. Searching, sorting, and filtering data. Queries for data selection. Queries with parameters. Calculated fields in queries. Multi-table databases. Types of relationships between tables. Foreign key. Database integrity. Queries to multi-table databases. Basic principles of database normalization. SQL data management language. Creating simple queries in SQL to select data from one table. Non-relational databases. Expert systems.
Information technology The role of information technology in the development of the economy of the world, country, region. Open educational resources. Professions related to computer science and information technology: web designer, programmer, mobile application developer, tester, software architect, data analysis specialist, system administrator. Familiarization with promising areas of information technology development (using artificial intelligence and machine learning as an example). Smart city systems (computer vision and big data analysis).	**Information technology. Artificial intelligence tools** Machine translation and speech recognition services. Cognitive services. Image identification and search, face recognition. Self-learning systems. Artificial intelligence in computer games. Using artificial intelligence methods in training systems. Using artificial intelligence methods in robotics. Internet of things. Prospects for the development of computer intelligent systems. Neural networks.

9. Methodological Aspects of Improving Standards for Primary and High School in Countries Involved in the Olympiad Movement in Informatics

Improving the educational standard in informatics in schools requires the emergence of new methods of supporting gifted schoolchildren. Let us consider several important methodological aspects that allow us to do this.

The main task of supporting gifted schoolchildren is the availability of such training for all children in the country. If the educational standard guarantees the availability of

training in new informatics content, then the guarantor of the implementation of such training are the schools of the country. At the same time, schools face various problems: training qualified teachers for new computer science topics, the availability of courses and teaching materials on these topics, including online, and creative events for schoolchildren to demonstrate their giftedness in these new topics of study.

It is important to note that the foundation of all new topics in informatics remains the mathematical foundations of informatics, algorithmization and programming. These sections of the course can be called the fundamental core of the subject of informatics. This fundamental core has been deepened in the school standard in accordance with the IOI syllabus, which is important for the development of programs for teaching gifted children considering these topics (complex algorithms, graphs, strings, expansion of programming languages with languages such as Python, C++ and Java). Gifted children can choose training in these sections, which are focused on the national olympiad in informatics and IOI, starting from the 7th grade of school. At the same time, to solve the problems described, it is important to introduce accessible forms of work with gifted schoolchildren. Traditional forms in many countries are special IT lyceums or centers for preparation for the Olympiad at the country level.

To cover younger schoolchildren, as well as to teach them new specialized modules of informatics, flexible forms of organizing the education of schoolchildren are required. The creation of such forms can be implemented based on the "Hybrid Learning" model, which optimizes the selection of teachers, including the involvement of university students as course curators, combines online, mobile and face-to-face forms and can cover children throughout the country. Such models can be quickly implemented in schools. These include: a school IT-Lab, IT Class at school or IT Club for a group of schools, IT Lyceum, IT Bus as a mobile IT Lab, as well as a partner IT Campus Online for the country, for example, based on a university or IT business park.

All these models create an accessible information educational environment for supporting gifted schoolchildren in the IT sphere. Their comprehensive implementation will allow creating a network of IT platforms for children with a choice of various specific forms for territories in the country. These models of organizing work with children will allow the introduction of new additional IT modules of their standard in computer science, which are flexibly configured according to the choice of students, for example:

- Programming languages (second and third programming languages of choice).
- Programming AI applications (bots, computer vision, voice assistants, translators, biometric recognizers, etc.).
- Algorithms and software tools for big data analysis.
- Control of unmanned robots and machines.
- Software procedures for information security, cryptography.
- Programming of desktop robots and digitally controlled machines.
- Circuitry and chip programming.
- Geoinformatics, cartography systems, geonavigation.
- Additive 3D printing technologies.
- Media education and digital arts.
- Advanced computer technologies and quantum computing.

For mass support of children passionate about computer science, the main role is played by such models as IT laboratories and IT clubs. For example, there are countries where school IT laboratories begin working with children from elementary school to high school in three age groups – elementary, basic, advanced. Small, specialized modules of 12–36 hours of training per year are used in training. In this case, the student can choose up to three training modules per year. This model implements the methodology – «test of the IT profession». It has become widespread in the last 5 years as a form of additional education based on individual schools in all territories. For example, in Russia, this model has become widespread in the form of such digital educational laboratories or clubs as «IT Cube», «Quantorium», and for rural schools – «Growth Point». It is important to note that the implementation of such a model requires modern, but small in number laboratory equipment, including computers, the Internet and sets of applied digital equipment. This allows schoolchildren to choose innovative areas of further professional training.

The IT class model has been widespread since 2010 as a systemic form of support for children interested in informatics. It implements the normative model – specialized training from grades 5 or 7 in schools and operates according to the standard of an in-depth level of informatics study. At the same time, children from IT classes can also choose modules – electives for additional training of their choice. IT classes require a professional – informatics teacher and curators – students at IT universities for specialized IT practices. By choosing an IT class, children receive an early choice of professions in the IT field and further education at an IT university.

The IT lyceum model is a classic model that has become traditional for working with gifted children since the end of the 20th century. Selection to lyceums usually begins in the 7th grade. Such lyceums have become the main training grounds for the national Olympiad in informatics and robotics. This model requires a highly qualified teaching staff with appropriate ICT competencies, equipment for several IT classrooms, and modern and diverse digital equipment in sufficient quantity. Leading universities and IT commercial companies can provide assistance to IT lyceums in this regard, where high school students can do an internship. Often, such IT lyceums are national or open at large universities in the country. In most cases, students leave home to study at an IT lyceum and live on campus. It is obvious that IT lyceums do not solve the problem of mass work with gifted schoolchildren, but prepare the IT elite, which is also important for the country.

During the pandemic, a new mass form of support for gifted children has become widespread – the «Digital School» model based on IT Campus or IT Parks at universities. It has significantly expanded the possibilities of educational courses for applicants at universities, since in addition to preparatory courses for admission to the university, it has become an online platform for popularizing modern IT, preparing and holding open, including international Olympiads in various new tracks of Olympiads in informatics.

A special feature of the «Digital School» model is the implementation of a hybrid form of education. Some events are held online, open to all comers or selected by level of training. Online rounds of Olympiads are also offered, based on the results of which

finals are held in a face-to-face format, when participants are still getting to know the university, being on site at the final. Such digital university schools cover children of all ages, any territories, have a flexible structure of many courses, and also have no staff shortage, as they invite students to work in the relevant departments.

Thus, all the proposed models, close to schoolchildren everywhere in the country, become real methodological resources for the development of children who are passionate about informatics, for mastering new educational IT modules in the informatics standard, technologically and do not lag behind innovations.

10. Discussion of the Prospects for the Development of New Forms of Teaching Gifted Schoolchildren in Informatics

Along with traditional forms of work with gifted schoolchildren in informatics, it is very important to consider IT innovations and help children choose promising development tracks. And here, the expansion of IT training tracks and the coordination of maximum coverage of all children involved in the above-mentioned models of schoolchildren's preparation are of particular importance.

The experience of the last 10 years has proved that the following organizational activities are required:

- Active development of various tracks of Olympiads in informatics, that organically complement the traditional national olympiad in informatics in the country, and which are also based on algorithmization and programming, but are used in an applied IT environment.
- Implementation of small organizational forms of IT training for children in the school infrastructure with the involvement of students from IT universities and IT business partners, and involving children in IT practices.
- Equipping IT classes and IT laboratories based on schools, additional education clubs.
- Development of mobile IT bus laboratories with access to samples of the most modern IT devices (for example, 3D helmets, drones, 3D pen, 3D scanner, devices with AI) and the Internet.
- Development of children's Digital Schools in universities.

If we talk about new tracks of the Olympiads in informatics, these could be the Olympiads in robotics, information security, artificial intelligence, financial literacy (business informatics), IT hackathons (integration of informatics tracks and other school subjects) and the Games of the Future (combining informatics and sports tracks). All of them, one way or another, rely on the traditional sections of the informatics Olympiad, but are supplemented by applied IT areas, showing the penetration of digital into all school subjects, which is now of great interest to children.

The experience of the International School of Informatics for Juniors (ISIJ, 2025), which was organized to support junior participants in the field of informatics, made it possible to identify the potential of IT parks in countries to implement the coordina-

tion function for national Olympiads and their development. In particular, to conduct an open selection stage in an online format. This can be done in an organized manner with children within walking distance for them at the sites of IT laboratories, IT classes in areas throughout the country. Such experience was gained in 2023 in Uzbekistan, where the national IT park has representative offices in each district of the country to work with children. This affects the popularization of informatics and supports new tracks of Olympiads in the IT field directly in partnership with schools, providing the personnel potential of the IT park to work with children. ISIJ allowed countries to begin the formation of a children's Info Park in the country as an integrator of IT education for children, coordinating the work of all models of IT education, including holding various new national IT Olympiads based on informatics school.

It can be argued that the formation of departments for working with children, such as Info Parks (informatics parks), at the IT parks of the country is a very promising area of cooperation between the school and the IT business community. For example, a successfully functioning model of an international digital school is the IT school of Innopolis University (Innopolis, 2025) `https://progmatica.innopolis.university/`, which is open to all children from 12 years of age from all over the world and is an online and face-to-face platform for preparing and holding the Innopolis International Open Olympiad in five tracks: mathematics (for informatics), informatics (programming), robotics, artificial intelligence, information security (OI Olympiad, 2025). (`https://dovuz.innopolis.university/pre-olympiads/innopolis-open/en`) Here children can get their first experience and start in the IT sphere, choosing different ones from the five above-mentioned tracks.

11. Conclusion

Based on the information provided in the article, it can be argued that the introduction of an advanced course in Informatics in the country's schools allows all motivated children to gain free access to Olympiad training in Informatics. But it is also important that in this case, all relevant specialized schools (for example, IT lyceums) receive a single program for Olympiad in Informatics for children, which provides teachers with the opportunity to work in a single standard regardless of the child's place of residence, as well as to receive a specialized textbook for the advanced course.

It is equally important that schoolchildren can master the fundamental principles of theoretical computer science and gain practical experience in applying knowledge in various modern programming environments, as well as try themselves in the profession as part of their first experience working with information systems, artificial intelligence systems and data analysis already at school. At the same time, all schoolchildren must also master digital literacy and information security culture, which is also useful for Olympiad participants.

The development of the federal standard based on the deepening of its scientific content, expansion by innovative technologies is dictated by the needs of any country for personnel in the field of high technologies and the need to modernize science in the

context of the digital transformation of the world. The special value of Olympiad in informatics is that it identifies and prepares human resources among young people for the digital economy.

The authors hope that the content of this article will be useful both to specialists involved in the development of the content of the national educational standard and to teachers working with talented schoolchildren in the field of informatics. Russia's experience shows that the use of the above-described advanced course in informatics starting from the 7th grade will allow talented schoolchildren to achieve high results not only in national, but also in international Olympiads in informatics, including the IOI.

Reference

Kiryukhin, V. (2010). Mutual Influence of the National Educational Standard and Olympiad in Informatics Contents. *Olympiads in Informatics*, 2010. Vol. 4, 15–29. https://ioinformatics.org/journal/INFOL061.pdf

Educational Standard (2022). *Register of sample educational programs, Ministry of Education of Russia*. https://fgosreestr.ru/educational standard

Curriculum of the Base School education (2023). *Register of sample educational programs, Ministry of education of Russia.* https://fgosreestr.ru/poop/federalnaia-obrazovatelnaia-programma-osnovnogo-obshchego-obrazovaniia-utverzhdena-prikazom-minprosveshcheniia-rossii-ot-18-05-2023-pod-370

Curriculum of the Hier school education (2023). *Register of sample educational programs, Ministry of Education of Russia.* https://fgosreestr.ru/poop/federalnaia-obrazovatelnaia-programma-srednego-obshchego-obrazovaniia-utverzhdena-prikazom-minprosveshcheniia-rossii-ot-18-05-2023-pod-371

Curriculum Advanced Informatics K7-9 (2023). *Register of sample educational programs Ministry of Education of Russia.* https://fgosreestr.ru/oop?edl=3&sub=19

Curriculum Advanced Informatics K10-11 (2023). *Register of sample educational programs, Ministry of Education of Russia.* https://fgosreestr.ru/oop?sub=19&edl=2

FLSB (2024). *Federal List of schoolbooks, Ministry of Education of Russia.* https://fpu.edu.ru/

RESH (2024). *Russian Electronic School, Ministry of education of Russia.* https://resh.edu.ru/

Sanitary Standard (2024). *Register of sample educational programs, Ministry of education of Russia.* https://fgosreestr.ru/sanitary_standard

ISIJ (2025). *International School of Informatics for Juniors*. https://isi-junior.ru/uch/infopark/

Innopolis (2025). *Innopolis University*. https://progmatica.innopolis.university/

OI Olympiad (2025). *Innapolis Open International Olympiad.* https://dovuz.innopolis.university/pre-olympiads/innopolis-open/en

M.S. Tsvetkova is professor of the Russian Academy of Natural Sciences, PhD in pedagogic science, prize-winner of competition "The Teacher of Year of Moscow" (1998). From 2002 to 2018 she is a member of the Central methodical commission of the Russian Olympiad in informatics and the pedagogic coach of the Russian team on the IOI. She is the author of many papers and books in Russia on the informatization of education and methods of development of talented students. She is the author of official textbooks and copybooks in Russia for primary school in Informatics. She is author and director of the International school in Informatic ISIJ (since 2017). She is the Russian team leader (2013–2017). She was awarded the President of Russia Gratitude for the success organizing the training of IOI medalists (2016). She was the Expert of Committee on Education and Science State Duma of the Russian Federation (2017–2021), and she has the Committee on Education and Science State Duma Gratitude (2021).

V.M. Kiryukhin is professor of the Russian Academy of Natural Sciences, PhD. He is the author of many papers and books in Russia on development of Olympiad movements in informatics and preparations for the Olympiads in informatics. He is the exclusive representative who took part at all IOI from 1989 to 2017 as a member of the IOI International Committee (1989–1992, 1999–2002, 2013–2017) and as the Russian team leader (1989, 1993–1998, 2003–2012). He received the IOI Distinguished Service Award at IOI 2003, the IOI Distinguished Service Award at IOI 2008 as one of the founders of the IOI making his long term distinguished service to the IOI from 1989 to 2008 and the medal "20 Years since the First International Olympiad in Informatics" at the IOI 2009. He was the chairman of the IOI 2016 in Russia and has the award medal of the President of Russia (2016) for organizing the Olympiad in Informatics in Russia and training IOI medalists since 1989. He is now the President of the International Organizing Committee of the ISIJ.

About Journal and Instructions to Authors

OLYMPIADS IN INFORMATICS is a peer-reviewed scholarly journal that provides an international forum for presenting research and developments in the specific scope of teaching and learning informatics through olympiads and other competitions. The journal is focused on the research and practice of professionals who are working in the field of teaching informatics to talented student. OLYMPIADS IN INFORMATICS is published annually (in the summer).

The format for the journal follows the tracks:

- the primary section of the journal focuses on research
- the second report section is devoted to sharing experiences of countries in informatics olympiads
- the last smallest section presents books reviews or other information

The journal is closely connected to the scientific conference annually organized during the International Olympiad in Informatics (IOI).

Abstracting/Indexing

OLYMPIADS IN INFORMATICS is abstracted/indexed by:

- Cabell Publishing
- Central and Eastern European Online Library (CEEOL)
- EBSCO
- Educational Research Abstracts (ERA)
- ERIC
- InfoBase Index
- INSPEC
- SCOPUS – Elsevier Bibliographic Databases

Submission of Manuscripts

All research papers submitted for publication in this journal must contain original unpublished work and must not have been submitted for publication elsewhere. Any manuscript which does not conform to the requirements will be returned.

The journal language is English. No formal limit is placed on the length of a paper, but the editors may recommend the shortening of a long paper.

Each paper submitted for the journal should be prepared according to the following structure:

- concise and informative title
- full names and affiliations of all authors, including e-mail addresses

- informative abstract of 70–150 words
- list of relevant keywords
- full text of the paper
- list of references
- biographic information about the author(s) including photography

All illustrations should be numbered consecutively and supplied with captions. They must fit on a 124 × 194 mm sheet of paper, including the title.

The references cited in the text should be indicated in brackets:

- for one author – (Johnson, 1999)
- for two authors – (Johnson and Peterson, 2002)
- for three or more authors – (Johnson *et al.*, 2002)
- the page number can be indicated as (Hubwieser, 2001, p. 25)

The list of references should be presented at the end of the paper in alphabetic order. Papers by the same author(s) in the same year should be distinguished by the letters a, b, etc. Only Latin characters should be used in references.

Please adhere closely to the following format in the list of references:

For books:

Hubwieser, P. (2001). *Didaktik der Informatik*. Springer-Verlag, Berlin.

Schwartz, J.E., Beichner, R.J. (1999). *Essentials of Educational Technology*. Allyn and Bacon, Boston.

For contribution to collective works:

Batissta, M.T., Clements, D.H. (2000). Mathematics curriculum development as a scientific endeavor. In: Kelly, A.E., Lesh, R.A. (Eds.), *Handbook of Research Design in Mathematics and Science Education*. Lawrence Erlbaum Associates Pub., London, 737–760.

Plomp, T., Reinen, I.J. (1996). Computer literacy. In: Plomp, T., Ely, A.D. (Eds.), *International Encyclopedia for Educational Technology*. Pergamon Press, London, 626–630.

For journal papers:

McCormick, R. (1992). Curriculum development and new information technology. *Journal of Information Technology for Teacher Education*, 1(1), 23–49. `http://rice.edn.deakin.edu.au/archives/JITTE/j113.htm`

Burton, B.A. (2010). Encouraging algorithmic thinking without a computer. *Olympiads in Informatics*, 4, 3–14.

For documents on Internet:

IOI (2008). *International Olympiads in Informatics.* `http://www.IOInformatics.org/`

Hassinen, P., Elomaa, J., Ronkko, J., Halme, J., Hodju, P. (1999). *Neural Networks Tool – Nenet (Version 1.1)*. `http://koti.mbnet.fi/~phodju/nenet/Nenet/General.html`

Authors must submit electronic versions of manuscripts in PDF to the editors. The manuscripts should conform all the requirements above.

If a paper is accepted for publication, the authors will be asked for a computerprocessed text of the final version of the paper, supplemented with illustrations and tables, prepared as a Microsoft Word or LaTeX document. The illustrations are to be presented in TIF, WMF, BMP, PCX or PNG formats (the resolution of point graphics pictures is 300 dots per inch).

Contacts for communication

Valentina Dagienė
Vilnius University
Akademijos str. 4, LT-08663 Vilnius, Lithuania
Phone: +370 5 2109 732
Fax: +370 52 729 209
E-mail: valentina.dagiene@mif.vu.lt

Internet Address

All the information about the journal can be found at:

`https://ioinformatics.org/page/ioi-journal`

Olympiads in Informatics

Volume 19, 2025

Publisher office: Vilnius University
Akademijos str. 4, LT-08663 Vilnius, Lithuania
June, 2025

Olympiads in Informatics

Volume 19, 2025

ISSN 1822-7732