# COMPUTATIONAL THINKING FOR TEACHERS AND CLASSES

Robert Porzak, Panagiotis Psomos

Innovatio Press Publishing House

# COMPUTATIONAL THINKING FOR TEACHERS AND CLASSES

**Edited by**
Robert Porzak
Panagiotis Psomos

# COMPUTATIONAL THINKING FOR TEACHERS AND CLASSES

**Edited by**
Robert Porzak
Panagiotis Psomos

Lublin 2023

This project has been funded with the support from the European Commission (project no: 2020-1-PL01-KA201-081924). This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

WSEI UNIVERSITY

*Publishing series:*
Monograph of the Faculty of Human Sciences of WSEI University

**Computational Thinking for Teachers and Classes**
First edition

*Editors:*
Robert Porzak
Panagiotis Psomos

*Reviewers:*
dr hab. Wiesław Kowalski, prof. WSEI
Paraskevi Poulogiannopoulou, Phd, European University Cyprus

*Proofreading:*
Beata Machulska

*DTP:*
Marta Krysińska-Kudlak

*Cover design:*
Wiktor Bogusz

*Cover artwork:*
The cover was designed using assets from freepik.com
Cover art: gpointstudio/ freepik.com


This publication has been funded with the support from the European Commission (project no: 2020-1-PL01-KA201-081924). This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Printed in Poland
Innovatio Press Publishing House
WSEI University
20-209 Lublin, Projektowa 4
tel.: +48 81 749 17 77, fax: +48 81 749 32 13
www.wsei.lublin.pl, e-mail: wydawnictwo@wsei.lublin.pl

E-ISBN: 978-83-67550-13-0
OPEN ACCESS

## Spis treści

# Introduction

Computational thinking is a skill that enables us to solve complex problems by breaking them down into smaller and simpler steps, finding patterns and similarities, abstracting away irrelevant details, and designing algorithms that can be executed by computers or humans. It is a skill that is essential for the 21st century, as technology becomes more pervasive and influential in every aspect of our lives. Computational thinking can also enhance our creativity, critical thinking, and collaboration skills, as we learn to apply it to various domains and disciplines such as art, language arts, math, science, and social studies.

This book is aimed at teachers who have not yet made extensive use of tools to support the development of students' computational thinking. The book aims to help teachers understand what computational thinking is, why it is important and how they can integrate it into the existing curricula. It is based on a literature review of the current research and best practice in computational thinking education, as well as on the experience and insights of the authors and partners of the CTApp project. The book provides a number of practical tips and examples for teachers who want to integrate computational thinking into their classrooms, using educational technology tools as well as content-specific methods.

The book consists of seven chapters. The first chapter introduces the concept of computational thinking and its components: decomposition, pattern recognition, abstraction and algorithm design. It also explains the benefits and challenges of teaching computational thinking and the role of educators in supporting it. Chapter Two focuses on teaching decomposition and Chapter Three on pattern recognition, the ability to break down a problem into smaller parts and find similarities between them. They include strategies and exercises for teaching these skills across subjects and grade levels. The fourth chapter covers teaching abstraction and the fifth covers algorithm design, that is, the skill of removing unnecessary detail and creating a sequence of steps to solve a problem. These chapters provide guidance and examples for teaching abstraction and algorithm design skills in a variety of contexts and scenarios. The sixth chapter provides an overview of the CTApp Game, which helps students practise computer thinking skills in a fun and engaging way. The chapter describes the idea, structure and features of the game and how teachers can use it in their classrooms. The seventh chapter summarises some popular strategies and online resources for integrating computer thinking into different subjects. It also provides links to additional resources and identifies opportunities for further training in computational thinking.

The Appendix to Chapter Seven presents three sample scenarios for different topics (one scenario provided by the CTApp project partners Cyprus, Poland and Italy), which illustrate how computational thinking can be applied in different fields.

We hope that this book will inspire you to explore the possibilities of computational thinking in your teaching practice, and help you prepare your students for the future challenges and opportunities that technology will bring.

*Cristina Fregonese*
*INNOVA S.r.l.*

# 1. Computational Thinking at first glance - What, why and how

## 1.1. Definition of Computational Thinking (CT)

Is that a fact? There is still no universally accepted definition of "Computational Thinking".

The concept of computational thinking (CT) was first introduced by an educationalist Seymour Papert in 1967 talking about LOGO, the programming language he developed at MIT (Massachusetts Institute of Technology) to teach programming to children. He was convinced that the use of computers could foster formal thinking in children and, in particular, could allow children to autonomously "construct" their learning and thinking.

The concept of CT was then revitalized in 2006 by a computer scientist Jeannette M. Wing who, in the article "Computational Thinking", argued that it addresses the conundrum of machine intelligence by asking what machines do better than man and what man does better than machines. Wing argues that computational thinking is not simply a procedural coding activity, but is a basic conceptual skill that, along with reading, writing and arithmetic, should be taught to all children. It appears that computational thinking purports to be critical thinking in evaluating situations and an advanced problem-solving ability using computerized tools.

If computer science is the science of what can be computerized and how to computerize it, however, computational thinking is not a skill unique to computer scientists. It allows problems to be solved, a system to be designed and human behaviour to be understood in everyday life, in an alternative way, through the fundamental concepts of information technology.

## 1.2. Characteristics of Computational Thinking

Some studies have linked CT to critical thinking further defining it as a new method of solving a problem using computer science techniques. These authors (Giacalone, 2020) define critical thinking as a "*skill or competence, by which the individual transcends, in a deliberate manner, in order to reach reasonable conclusions that can be corroborated using valid information*". It is a way of thinking that makes it possible to give rise to multiple solutions. In 2016, Prof. Leen-Kiat Soh indicated that CT complements critical thinking as a way of reasoning to solve problems, make decisions and interact with the world. CT therefore involves techniques such as abstraction, decomposition, algorithmic design, generalization, evaluation and interaction with the computer.

*Key Skills for Computational Thinking*

There are four key skills in computational thinking:

1 Decomposition
2 Pattern Recognition
3 Pattern Abstraction
4 Algorithm Design

1) *Decomposition*

Breaking down big and difficult problems into something much simpler. Often big problems are just many little problems put together. Decomposition is an important life skill to be relied on in the future when students and adults need to take on larger tasks. Students will learn ways to delegate group projects and build time management skills

2) *Pattern Recognition*

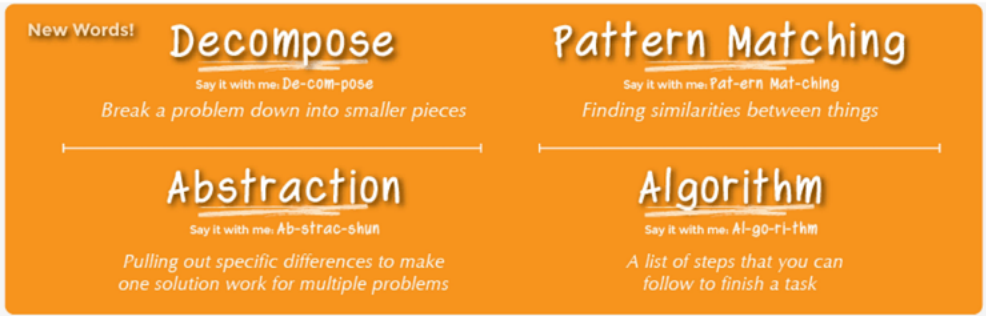
Sometimes when a problem is made up of many small bits, you will notice that these bits have something in common. If they do not, they could, however, have strong similarities with the pieces of another problem that has already been solved. If you are able to find these regularities, it will become a lot easier to identify the individual pieces: pattern recognition is simply looking for patterns in the puzzles and determining if any of the problems or solutions we encountered in the past may apply to a present situation. May, what we learned in the past, help us sort out the actual problem? If you have ever built a piece of IKEA furniture, you will understand the importance of patterns. While assembling an IKEA drawer unit, it is likely to take you much longer to assemble the first drawer than the fourth or fifth. When we repeat steps in our assembling process we learn how to solve the instructions more quickly and learn from our mistakes. The painstaking process of assembling that first part teaches us the skills to perform the process more efficiently in the future.

3) *Pattern Abstraction*

Once you have located a pattern, it is possible to abstract (ignore) the details that differentiate the various things and use general techniques for finding solutions that work for more than one problem. Identifying the crucial information in a problem and disregarding the irrelevant information is one of the hardest parts of computational learning.

4) *Algorithm design*

When the solution is ready, it is possible to write it down so it can be executed step by step. This makes easier to obtain the expected results. Algorithm design is setting out the steps and rules needed to follow in order to achieve the same desired outcome every time.




Sorce: https://code.org/curriculum/course3/1/Teacher.pdf

The essence is that CT creates procedures that allow an agent to meet the given objectives within a predefined context. Therefore, CT is a useful intellectual tool for everyone, no matter what their job is.

## 1.3. Algorithms and Coding - why computational thinking is so valuable

Computational thinking helps us deal with problems by generalizing them.

The problems deriving from the lack of computational thinking occur in irrational behaviors: we are not robots, because we are guided by emotions, but tackling a problem rationally simplifies its solution. In addition, developing this approach makes children more cooperative with their peers.

Following computational thinking, **a solution to a problem is found by formalizing it in a sequence of actions to communicate it to others**: a practical example is a recipe. We need to follow precise steps to make a cake and if we know how to pass the procedure clearly and unambiguously, the cake can also be made by other people.

Algorithms and Coding are Strictly connected with Computational Thinking, but what are they?

**Algorithms** are among the greatest achievements of humanity. An algorithm is a rigorous process for solving a problem or realize an idea. Algorithms are the basis of most of our daily activities. It is thanks to their application that we know how to calculate the sum of two numbers, find a name in a list, decide the way to follow to reach a place. CT it is nothing more than the ability to understand, apply and conceive algorithms. The practice of Coding allows everyone to become familiar with the algorithms.

**Coding** is an English word which corresponds to an Italian word "programming". It obviously means programming computer science, but not in a traditional sense. Coding in school is a recent discovery.

It's an approach that puts programming at the center of a path where learning, starting from the first years of life, is open to new paths and is at the center of a larger project that breaks down IT barriers and stimulates a problem-solving approach. We should say that CT is an unusual approach to problems and their solution. **With**

**coding, children and young people develop CT and the ability to solve more or less complex problems. They learn not only to code, but to program for learning**.

**Coding** can also be practised **without digital devices**, in analogical contexts: in this case we speak of "**coding unplugged**". This discipline is especially **suitable for students from kindergarten to lower secondary school**. (Midoro, 2016)

By now many schools indicate, among the activities they reserve for children, hours of coding and computational thinking. There is such a need because students very often arrive at high school and at university without possessing the ability to manage and deal with problematic situations.

The problem is that young **people struggle to interpret reality**: perhaps this is because they are digital natives and have set aside those skills that were once used to deal with problems. Added to this is the fact that today the logic that dominates society is "***everything and immediately***": if the problem is difficult, one puts it aside and avoidance strategies are used.

The advantage of coding is that of bringing children closer to information technology as a cultural object that is increasingly part of our lives and the world of work. This language must therefore be understood and children cannot ignore it. Like geography, robots must also know: mastery of coding in fact helps to govern the machines, knowing how to write their language, no longer just read it. (Giacalone, 2020)

**CT helps students to develop skills that are attractive for future employment opportunities**. Computer science is the fastest growing job market and students with skills in coding are highly sought after job applicants. While hard tech skills are very important, it's softer skills of reasoning and problem solving that employers really find attractive. These skills are the key to successful understanding why computational thinking is so valuable.

The scientific-cultural side of computer science, also referred to as "computational thinking", helps to develop logical skills and the ability to solve problems in a creative and efficient way - the qualities that are important for future citizens. Through coding, therefore, **transversal skills and attention, concentration, memory, creativity, etc. are enhanced**. (Gabbari, Gagliardi, Gaetano, Sacchi, 2020)

**Problem - solving key skill of a future job**

The present-day job market requires workers capable of solving non-routine problems (see *World Economic Forum 2016*). According to the Report on Adult Skills (PIACC Survey of Adult Skills) people today encounter growing difficulties in their work which give way to situations in which it is increasingly necessary to think before acting.

One possible explanation for the transition from routine to non-routine tasks in the workplace is that, due to the widespread introduction of computerized equipment, workers no longer have to perform manual routines. Instead, they are required to **face unexpected and unfamiliar problems** in governing the machines they manage.

Too often teachers find that while their students excel at routine exercises, they fail to solve problems they have never encountered before. Problem-solving skills are an essential component of the skills required to analytically approach interpersonal and

non-routine tasks. In both cases people have to think about how to deal with the situation, systematically monitor the effect of their actions, and modify them accordingly.

Starting from a primary school, we can affirm that computational thinking teaches children to think in an algorithmic manner, to find a solution and develop it, and this occurs with programming and coding. Actually **coding gives children a mindset, which will allow them to tackle complex problems when they are older.**

*In the same way that we don't teach music in schools to students to become professional violinists, or English to students to get jobs in journalism, we should not teach programming to children to get programming jobs: we should do so to them to* ***acquire a new way of thinking and seeing the world.***

## 1.4. Application of computational thinking in the classroom

The concepts of computational thinking described above (the algorithmic thinking, decomposition, etc.) are associated with several behaviors on the part of students, which can be observed in the classroom (Computing at School – CAS, 2015)

***Algorithmic thinking***

Algorithmic thinking represents the ability to think in terms of sequences and rules, as a way of solving problems. It's about a basic competence that pupils develop as they learn to write their own computer programs. In class it is possible to observe:

- Formulating commands to obtain a desired effect.
- Formulating instructions to be followed in a given order (sequence).
- Formulating commands using arithmetic operations and logics.
- Writing command sequences that memorize, move and manipulate data (variables and assignment).
- Writing commands that choose between several instructions that compose them (selection).
- Writing commands that repeat groups of instructions that make them up (loop / repeat).
- Grouping and naming a set of commands that perform a defined task to create a new instruction. (subroutines, procedures, functions, methods).
- Writing commands, which include subroutines which use replicas of themselves (recursive algorithm).

Sorce: AlgoBlocks - Suzuki, H., & Kato, H. (1993): https://www.edutech.it/education/blog-edu/item/41-coding-tangibile-la-sua-evoluzione-e-i-vantaggi-di-utilizzo.html

- Writing a series of commands that can be followed simultaneously by several agents (computers/people, parallel thinking and processes, competition).
- Writing a set of declarative rules (coding in Prolog or in a database querying language).

This also implies:

- Using appropriate wording to write a code that represents one of the above commands.
- Creating algorithms to test a hypothesis.
- Creating algorithms that provide solutions based on experience (heuristics).
- Creating algorithmic descriptions of real world processes in order to have a better understanding of them. (computational modeling).
- Designing algorithmic solutions that take into account the capabilities, limitations and needs of the people who use and benefit from them.

***Decomposition***

Decomposition is a way of thinking about artifacts in terms of parts that compose them. Individual parts can be understood, solved, developed and evaluated separately.

In class it is possible to observe:

- Dividing artifacts into constituent parts to make them easier to work.
- Breaking down a problem into simpler versions of itself, which can be solved in the same way (divide et impera and recursive method).

***Generalization***

Generalization is a way of solving new problems on the basis of the solutions given to the previous ones. It's about identifying and exploiting models. In class it is possible to observe:

- Identification of patterns and common characteristics in artifacts.
- Adjustment of solutions, or parts of them to be applied to the whole class of similar problems.
- Transfer of ideas and solutions from one problem area to another.

***Abstraction***

Abstraction is the process of creating a greater artifact understanding by hiding details. In class it is possible to observe:

- Reducing complexity by eliminating unnecessary details.
- Choosing a way to represent artifacts so that they can be manipulated in a useful way.
- Concealing all artifact complexities (hiding functional complexity).
- Hiding complexity in data such as using data structures.
- Identifying relationships between abstractions.
- Information filtering in solution development.

***Evaluation***

Evaluation is the process of verifying that the solution is correct: that it fits the purpose. During thought-based computational assessment, there is a specific, and often extreme attention to detail. In class it is possible to observe:

- Evaluation that the artifact fits the purpose.
- Process of determining which artifact performs the right function (functional correctness).
- Design and execution of scheduled tests and the interpretation of results (tests).
- Evaluation that the performance of the artifact is good enough (utility: effectiveness and efficiency).
- Comparison of the performance of artifacts running the same function.
- Compromise between conflicting needs.
- Evaluation whether the artefact is easy for people to use (usability).
- Process of determining if the artifact offers an experience adequately positive when used (experience of the user).
- Evaluation of any of the above against the specifications and the criteria set.
- Going through the processes or algorithms/algorithms programming step by step to process what they do (evidence/trace).
- Use of precise argument to justify that algorithm works (test).
- Use of precise argumentation to test usability or the performance of an artifact (analytical evaluation).
- Use of methods that include artifact observation used to evaluate its usability (empirical evaluation).
- Evaluation that the product meets the general criteria of performance (heuristics).

## 1.5. Training approaches

It has been noticed that it is possible to work at school to observe the development of CT, for example by asking pupils to mentally "disassemble" some actions they perform automatically, and to write down the correct, rigorous procedure that would allow a machine (an "alien", if preferred) to reproduce the same action without errors: for example, drawing a rectangle without removing the pen from the paper, or looking up a word in the dictionary. (Gabbari, Gagliardi, Gaetano, Sacchi, 2020)

Then there are an infinite number of unplugged activities that can be practised in class gradually and with a high degree of involvement.

In the *New Scenarios* of 2018 this pervasive function of CT is emphasized: "*It is a creative logical process that, more or less consciously, is put into action in everyday life to face and solve problems. Education is acting consciously: this strategy allows us to learn how to deal with situations in an analytical way, breaking them down into various aspects that characterize them, and planning the most suitable solutions for each* ".

**From computational thinking to coding**

But it is above all in the Coding activities, and therefore in the writing of languages intended for a machine, that CT can find ample space for development. Computers are ideal performers, they are not gifted with intelligence. This is why writing instructions that a machine will have to execute requires, to some extent, a greater degree of formality and rigor than in communication between humans.

Programming makes the concepts of CT concrete and turns it into a learning tool.

There are *textual* programming systems and *visual* programming systems: in the former the instructions (for the machine) must be written in sequence by means of a text editor; in visual systems, on the other hand, the individual instructions are represented by colored blocks that can be dragged into a work area (drag and drop). The blocks can be combined together in order to compose a sequence of instructions, which constitutes the program. (Gabbari, Gagliardi, Gaetano, Sacchi, 2020)

Visual systems are often preferred by teachers because block programming allows them to focus only on the procedure, not taking into account the correctness of the language: the blocks are already linked correctly from the syntactic point of view (at most you can make some semantic mistakes) and focus on reasoning. The teacher's task is to offer the right pretext and the right context.

How to choose a programming language for educational coding? Most of the teachers seem to be oriented towards an educational software language that offers a low floor and a high ceiling, as S. Papert says, that is, a language that facilitates the first steps as much as possible but, at the same time, allows increasingly complex projects to be carried out. (Gabbari, Gagliardi, Gaetano, Sacchi, 2020)

**How serious games support training**

Through serious games, training becomes a powerful and versatile tool. In an era where the large amount of information makes users easily distracted, the training courses created by companies, entities or institutions respond to the fundamental need, namely making training as effective, fast and engaging as possible. In this perspective, the trend has spread to enrich training with videos, cartoons, online games and role plays.

All this multimedia content is adopted by users of all ages to learn and to keep up-to-date in a playful and fun context. Thus, both the Millennials and Generation Z, fond of video games, and those more adult can have fun and learn by playing serious games. After all, learning by playing has always been one of the most deeply rooted mechanisms of people.

Being an interactive type of learning, serious games allow the player to **learn by doing** and to create his or her own content. Thanks to the learning by doing serious games, the learner will find himself in front of multiple decision scenarios, interactions with the didactic objects encountered during the game and feedback received from others. The learner will be assigned a score based on the correctness of the decision to be made. It is also possible to obtain badges or medals based on the objectives achieved**.** (Luciano and Fabio, 2017)

**Coding and computational thinking at primary and nursery school: the tools**

How is coding done at school and what tools are available? They can be fun tools, such as Scratch or Scratch Jr. for the little ones and Kodu (Luciano and Fabio 2017), or the code.org exercises. They look like games more than exercises. And in fact, from a certain point of view they are. The children, while playing and winning every challenge, do solve problems, even small problems like avoiding an obstacle or getting caught by one of the villains in history, just to give a couple of examples. To solve the problem they must work to understand what the possible solution is, and if they reach the goal it means they have learned how to do so. Meanwhile, unknowingly they have written lines of computer code, even if physically they have not written any, not even one, and they only moved rectangular blocks whose function and code correspond to one another.

The strategy, however, which most teachers want to focus on is to use a mathematical problem as an experiential situation, with the aim of stimulating children to get involved in identifying a solution strategy, perhaps through building things together with a group of companions and sharing the work with the whole class. In order to do so, complex activities are required, which are not exercises aimed simply at applying a formula, but proposals aimed at training critical thinking, elaboration, sharing and reflection on one's work. And, at this point, what can really be found useful are problems that have multiple possible solutions, or those "open" ones, where the final result is not unique.

There are many possible proposals, but, as usual, teachers have to deal with the time they want to dedicate to the activity and with the type of class in which they operate. These can be **critical points**, but there are many possible operational proposals to choose from and to adapt to the specific target and environment.

An idea could be to propose to children, divided into groups, that they identify the solution path of a problem, and then give directions for the solution to a robot-companion who does not know the text: the students will certainly have fun and in the meantime they will be motivated to correct themselves in the event of an error. A somewhat more complex activity could be to create a "relay" of problems: one group invents them, another one solves them and the third one corrects them. The final comparison and the stimulus to identify other possible strategies will be useful to focus attention on different resolution steps and on possible alternatives. (Luciano and Fabio 2017).

Another idea could be to have each group of students choose a problem among those proposed by us, asking them to write the steps necessary for the resolution. (Luciano and Fabio 2017).

A useful tool to take advantage of in this case may be the "good old" flow chart, linear or not, which is difficult to find in primary school books and notebooks, but which can be an excellent resource for visualizing the path taken. (Luciano and Fabio 2017). After all, how to define the command strip that is created with the Scratch program, if not by using a digital flow chart?

*Georginia Boyd*
*Aspire-Igen Group Ltd.*

# 2. Decomposition

## 2.1. Decomposition definition

Decomposition is another main core skill in computational thinking, being one of the four key elements in Google's "Computational Thinking for Educators" course. (Yihaun, 2019). If Computational Thinking is "the thought process involved in formulating problems so solutions can be represented as computational steps and algorithms", Decomposition is the vital step within computational thinking of breaking down data, processes, or problems into smaller, more manageable steps or parts until the user is confident with their ability to execute the solution. (Alder et al., 2022; Rich, et al. 2020; Mannila 2014, Aho, 2012; Csizmadia et al., 2015; Sengupta et al., 2013; Yihaun, 2019). Given this definition, you probably recognize that you apply Decomposition into your thought processes and action in everyday life. Every time you find yourself breaking apart tasks into manageable portions, you are successfully exercising decomposition skills.

## 2.2. Benefits of Decomposition in the Academic Setting

As one may have assumed, decomposition skills have many benefits in the academic setting. The three we will cover here are improving the student cognitive functions, enhancing teachers' capacity to convey complex topics, and acting as a way to predict student performance within the lesson, thereby alerting teachers to pay attention to students who may need additional assistance far before traditional retention testing techniques are required, such as exams.

Incorporating Decomposition skills into your lesson plan can help improve your student's cognitive performance by increasing their confidence when tackling larger and more complex theories. (See Weintrop et al. 2016). Just in the same way we chew

food, breaking apart complex tasks and assignments into more manageable portions helps to ensure students engage into the lessons and "digest" or retain, the skills conveyed therein. As a teacher stated in the Alder study: "[Decomposition] is great especially when I am trying to teach a complicated topic and I can break it up into much smaller parts for the students to understand it." (Alder et al., 2022).

Decomposition skills do not just benefit students, they can also benefit teachers. For example, in the Uzumcu (2021) study, the consensus among the participating educators was that decomposition is easily incorporated into existing lessons plans and played a vital step in problem solving and lesson planning exercises. (Uzumcu, 2021). This was, in part, due to the fact that it allowed the educators to break their course into manageable modules. In addition to assisting educators to plan and deliver their lessons, decomposition can act as a litmus test for the effectiveness of their course, as a student's ability to apply decomposition is a good predictor of a student's academic success. (Haddad & Kalaani, 2015 –based on data from 982 students over the course of two years). Similarly, the Uzumcu educators reported an increased capacity to predict and understand the depth of the student's comprehension level with an effective inquiry, built, in part, upon applied decomposition skills. (See Uzumcu, 2021). This in turn gives teachers the time and notice needed to support a class with a wide range of abilities.

## 2.3. Pedagogical Benefits of Decomposition

One of the reasons why decomposition skills are readily incorporated into lessons is that they are largely used in preschool and early childhood education (Calderon, 2015), making subjects and their topics 'digestible'.

## 2.4. Long-term Benefits of Learning Decomposition Skills

It is important to note that decomposition skills are not an independent subject or an independent topic of study (Hsu, 2018). They are used in everyday life, and the benefits of implementing decomposition skills have been widely recognized by researchers and teachers (Hsu, 2018; Lockwood, 2017) For example, participants in the Uzumcu study reported that they were able to use the skills they had learned in their everyday and professional lives, such as how to properly integrate decomposition into their work responsibilities (Uzumcu, 2021). Another example of extracurricular benefits is the ability to recognize gaps in comprehension. Decomposition skills allow students to identify for themselves what they understand and what they need to work on. This increases self-confidence and gives students clues to the proper allocation of time and resources.

## 2.5. How to Integrate Decomposition into Lessons

The primary way to integrate decomposition into lessons is to 'hint' at what the strategy is and then encourage students to use it when solving problems (Rich et al.,

2020). This can be done by asking students to indicate where and how they used the technique and how this contributed to the problem solving (Rich et al., 2020). In an astronomy lesson, for example, a teacher might ask students to identify a planet based on specific parameters rather than general properties. He or she then asks students how they carried out the identification, 'breaking down' the analysis into smaller component activities that are easier to use in other situations (Rich et al., 2020). Essentially, this approach is based on using decomposition practices as general problem strategies that can be applied to different issues (Rich et al., 2020). The key here is to avoid describing to students how a problem has been decomposed and instead provide opportunities for students to explore knowledge and solutions on their own (Alder et al., 2022).

For example, a teacher might incorporate decomposition into a mathematics lesson. He or she first divides the students into groups and then asks them to decompose a large number into prime factors. He or she then asks students to solve the problem by analyzing the equation, one number at a time. After doing this, it is the right time to explain to the students that this was an example of decomposition in practice (Rich et al., 2020).

Another method used to integrate decomposition into the curriculum is to use visual and tactile stimuli such as block diagrams or charts to illustrate decomposition (Krist et al., 2017). Another technique is to 'frame' decomposition and pattern recognition within a lesson. Essentially, this technique, although very similar to prompting, involves informing students about the decomposition technique, giving examples of how to apply it (preparing students to think about the application of decomposition), before students actually engage in the decomposition exercise later in the lesson (Rich et al., 2020). The key difference is that while prompting involves telling students when to use a technique, framing gives students the chance to apply it themselves, without prompting. Framing can also be used as a reflection tool after a lesson.

A class that considers decomposition in a fully integrated way might look like this: We start by exploring the differences between the data we want students to analyze and the information they currently have at hand. We encourage active, verbal reflection on students' criteria for singling out certain aspects of the task, perhaps by asking them to explain this to their partners. For example, in English classes, teachers might ask students to plan an essay by dividing the topic into subheadings and then explaining why they made these distinctions. In art classes, teachers can explain and demonstrate the benefits of approaching the representation of objects through the play of light and shadow.

Whichever way teachers choose to address decomposition, it is important to explain why decomposition is important: step-by-step discovery and learning exercises help with learning and understanding complex topics (Lockwood, 2017). Again, pairing students can be useful here as it will make exploring new skills more comfortable and allow pairs to help, refresh, discuss and test each other's knowledge (Isbell et l., 2010). Similarly, try creating lesson plans designed to show how decomposition skills already exist in students' lives (Burgett 2016).

## 2.6. Integrating decomposition into lessons

Studies show teachers have an easier time incorporating computational thinking strategies into rich, open-ended, inquiry tasks that were intentionally designed to engage students in higher-level thinking. (Rich, et al. 2020.) Unfortunately, many elementary curricula for mathematics and science do not provide teachers with access to such tasks (Rich, et al. 2020; Banilower et al. 2013; van Zanten & van den Heuvel-Panhuizen, 2018). The first step is to therefore ensure any standardized curriculum allows for the exploration of higher-level theoretical thinking, primarily by allowing students the time to reflect upon lessons learnt, rather than focusing on the pure output.

Therefore, you are likely to find that you will have to redesign lessons with the specific intent to introduce opportunities to frame and prompt decomposition. While potentially daunting at first, this will give educators a chance to further polish lessons, by enabling them to think deeply about the instruction and engagement opportunities they are providing. (Rich, et al. 2020). Students, and teachers alike only learn through engagement and application of these techniques. (McCormick, 2022).

*Georginia Boyd*
*Aspire-Igen Group Ltd.*

# 3. Pattern Recognition

## 3.1. Pattern Recognition definition

One of the key critical elements of computational thinking, i.e. pattern recognition, is the identification of data with one or multiple similarities in a sequence. (Howard, W.R. (2007). Other working definitions include looking for similarities between new problems and problems that have already been solved, looking for similarities and patterns between things, or identification of data with one or more similarities in a sequence. (Rich et al., 2020; Howard, W.R. (2007).

In practical use, pattern recognition allows the user to solve problems quickly, by applying solutions that worked in the past, provided the problem the individual presently faces is similar to those they encountered in the past. This, of course, requires the user to observe and identify patterns, trends, and regularities in data, processes, or problems (Yihaun, 2019). Most teachers and educators are likely to be already with pattern recognition. Practical examples include asking students, 'What did we learn in the past which can help us find this solution?'

Though there are straightforward applications of pattern recognition in the mainstream learning - primarily in mathematics, science, art – there is plenty of relevance within vocational training and further into the world of work. In vocational training, pattern recognition is inherent in allowing students to practise manual work in person to develop their skills. By developing pattern recognition skills, students can relate current lessons to their future careers and gain perspective on what is needed to succeed in their area.

## 3.2. Pattern recognition - benefits for teachers

**Subject-Specific Benefits**

While the relevance of pattern recognition can be seen in topics such as Maths, Computing and IT, and Science, teaching pattern recognition can support students across all subjects. Pattern recognition is simply a method of thinking, and the applications are wide-reaching and comprehensive.

For example, in the Uzumcu study, in which a group of educators applied various computational thinking techniques, including pattern recognition into their lesson plans, all the participants thought that the exercises were helpful in developing problem-solving skills. (Uzumcu, 2021).

To give the context and clear examples, we have highlighted a few subjects which benefit from the pattern recognition. These include mathematics, sciences, art & creativity, and languages – though it should be noted that pattern recognition is largely applicable to all subjects.

In mathematics, pattern recognition is heavily used and effortlessly combined with other computational thinking skills, such as algorithmic thinking. Research shows that young people's ability to recognize patterns forms the basis of an early mathematical theory. When young people are taught to identify similarities, they will ideally begin to apply these lessons across all subjects, therefore extrapolating lessons learned in mathematics, i.e. how to change patterns, adapt, and find irregularities, to other endeavours. (Department of Education, 2014).

In sciences, extensive research has shown that pattern recognition is a successful method for learning scientific skills in areas such as medicine, chemistry, and engineering. (Samarasinghe, 2006). In fact, pattern recognition plays a crucial part in a genetic theory, and helps students conceptualize abstract and challenging concepts. (See Hsu 2018).

In arts, pattern recognition assists in helping convey abstract and developmental theory. For example, pattern recognition helps in the understanding of the development and analysis process. (Shen, 2013). This understanding, identical to that used with the aforementioned nursery rhymes, can also help with the creation of art.

Yet, perhaps the most common application of pattern recognition, apart from music and math, is in linguistics. Much to the average student's dismay, language is almost exclusively taught through the use of patterns, at least in the classroom. The theory is that learning a language using pattern recognition supports a more detailed understanding of the way the language works. For example, a basic understanding of languages can be developed and expanded using pattern identification to define sentence structure, categories of writing, and vocabulary. (Larson 2010). Of course, pattern recognition is not just limited to the above. Later, we will detail specific examples other educators used to incorporate pattern recognition into their subjects.

**Pedagogical Benefits**

Like many skills in computational thinking, pattern recognition is a solid foundational skill for education. In fact, pattern recognition is already stressed in almost every aspect of our lives and early education, with pedagogical practices within early years typically integrating pattern recognition alongside traditional didactic teaching practices such as explorative learning, questioning, scaffolding skills, and acquisition. (Calderon, 2015).

As stated previously, the strength of pattern recognition is that it allows students to tackle problems using universal methods. Therefore, rather than having to teach students answers to every single problem individually, teachers instead must teach pattern recognition and the adequate solution. This helps students to develop the skills to understand complex concepts more quickly by demonstrating similar patterns occurring across subjects. A classic example is teaching multiplication tables,

particularly multiplication by a value of nine. Where a teacher could teach the students every single value in the table, they could instead teach a pattern: the answer can be derived by increasing the first digit by a value of one, while simultaneously subtracting the second digit by a value of one, i.e. nine (09), eighteen (18), twenty--seven (27) etc.

As suggested previously, pattern recognition is particularly useful in linguistics: there are many similarities (an therefore patterns) between Romance languages: Spanish, Italian, and French, and the same similarities in 'Germanic' languages, including German, English and Dutch. If a student is taught pattern recognition, and knows one of the languages in the family, i.e. French, for example, it could be easier for them to learn Spanish than Dutch – as more patterns and similarities (in vocabulary, grammar, or sentence structure) are shared in linguistic families. This knowledge can help students expedite their education, and help teachers more accurately plan lessons.

Pattern recognition could also help educate students on more abstract ideas, such as the development of concepts. While describing a concept as an individual theory can be confusing and may feel too unfamiliar to students, provided a teacher can create a cognitive link for students, the lessons may be more easily understood as simple progressions of familiar concepts thereby expediting uptake via the extrapolation of familiar concepts.

Pattern recognition can also assist teachers. Studies indicate that a student's grasp of the fundamental building blocks are "uniquely predictive of higher level skills above and beyond the effect of lower level skills" and act as a predictor of a student's academic success, subject retention, likelihood of progression, and graduation rates. (Lockwood 2017; Haddad & Kalaani 2015). This knowledge allows teachers to identify students who may need additional support quickly.

Not only does this skill allow teachers to identify these students, it also helps them support them, as teachers can quickly replicate teaching strategies for similar students. Where one student with certain abilities and learning styles may have difficulties with a particular subject, the teacher will be able to recognize similar traits and use past experiences of successful methods with similar students to support them.

## 3.3. Pattern recognition - benefits for students

Alongside improving students' confidence and acting as a force multiplier for subject uptake, the skills learned in computational thinking - including pattern recognition - function as fundamental building blocks for a child's education. Teaching this skill provides the students with computational and critical thinking skills necessary for students in the 21st century, regardless of their ultimate field of study or occupation. (2014 Mannila). In fact, it is widely recognized that the ability to identify and analyze structures for patterns is the foundation of understanding how one processes information. (2014 Mannila). Effectively, teaching pattern recognition is akin to both a teacher and the students on how to learn and study on their own – a vital skill for every aspect of life.

## 3.4. Challenges in teaching pattern recognition

While pattern recognition is found within the existing educational systems, it is not always expressly identified. As with everything else, it may therefore be difficult to expressly incorporate it into lessons, especially if a teacher is not familiar with how to articulate such an innate concept. (Hsu 2018). Given the current demands on teachers, it may be particularly difficult for educators to change teaching materials in a short period of time. (Hsu 2018).

Therefore, it may be of some use to educators to have tested examples of this integration. One such example can be found in France. There, the Chipranov study (2016) integrated computational thinking skills, including pattern recognition, into French elementary education via games and robotics while utilizing a whole-class discussion and demonstrations. These would be followed by collaborative or individual learning, and finally, students would be encouraged to reflect on the solution, skills, and techniques learned. (Chiprianov 2016). Some of the benefits of this multi-tiered approach was increased retention rate, alongside this important capacity for students to apply the skills learned within the classroom to alternative challenges. (See Chiprianov 2016.)

Further refinements were proposed by Burgett (2016), who suggested creating lesson plans which are designed to show how these skills already exist in students' lives and approaches to education that help with the concept uptake. This "hands on approach" is a vital edge if you happen to encounter issues similar to those noted by Mooney, et al. (2014) who found in the study that students found the potentially abstract concepts of computational theory challenging if presented with merely theoretical, and abstract approach. Teachers can also 'prompt' students, encouraging them to use the strategies when tackling the problem. (Rich, et al., 2020). This can be done by asking them to identify where and how they have used them and how they have contributed to solving problems. (Rich, et al, 2020). In essence, this approach hinges on the conceptualization of the pattern recognition practices as general problem strategies that could be applied to a variety of problems. (Rich, et al. 2020).

An example of a more hands on approach can be found in the Mannila study (2014) There, nursery rhymes were used to link middle-school students with the practice of pattern recognition. This was chosen to try and teach pattern recognition to students through a medium they were already familiar with. Pupils were told to collect nursery rhymes from home, analyze them, then identify structural patterns within them (i.e. prologues, repeated refrains, and epilogues). Students were then told to use these pattern recognizing skills in an entirely different manner, by building "toy machines" of cardboard and other cheap materials (effectively flash cards), with the aforementioned pattern components written on them. Then, students were encouraged to "mix-and-match" the nursey rhymes according to the patterns they identified before to create entirely new nursey rhymes. (Mannila 2014). This, in theory, allowed students to begin to apply empirically and consciously what was previously a subconscious unspoken theory.

## 3.5. Integrating pattern recognition into lessons

Despite the aforementioned potential shortfalls of quickly integrating a new theory, pattern recognition can, and has been expressly and successfully integrated into classrooms globally. (Yihaun, 2019). This successful integration is not limited to traditional computer science of STEM course, as even though pattern recognition is vital for computational thinking and programming, it facilitates problem-solving across all subjects. (Yihaun, 2019).

**In particular,** one study of over 116 middle and high school mathematics, science, social studies, and English teachers provides very useful examples of pattern recognition being successfully integrated with the existing resources into the classroom. (Yihaun, 2019).

For instance, high school history & English language arts teachers were able to create a historical puzzle and roleplaying games for their students to explore life in the Middle Ages. The students were primed before these puzzles and games to identify patterns within them, as well as within historical narratives, story beats, and events. (Yihaun 2019). There is a potential to further develop on this model, by encouraging students to use the patterns they recognized in past periods to predict what they would encounter in the next period in history.

In the science sector, students of the sixth to the tenth grade were able to integrate pattern recognition directly into their exam via the Rock & Mineral Quiz. Being exposed to an exploratory lab activity, these students were challenged to apply pattern recognition when measuring different rock & mineral properties. They would then record these results and use them to create educated predictions of future properties in different samples. These predictions would then be tested, further refining student's knowledge of mineral properties. (Yihaun 2019).

Meanwhile, at the University of North Florida, the USA, students were encouraged to use pattern recognition skills to build "concept maps" in English: helping students recognize patterns in writing structures and give them a formulae which could be used to help them "write clearly". (Howell et al. (2011). This pattern recognition skill was also applied to song lyrics, creating storyboards for drama, and identifying recurring symbolism in literature & media. Building on these ideas, the students could also be encouraged to identify repetition in game rules, media characters (i.e. tropes), riddles, poems, graphs, individual & group behaviour (Phycology).

Even in the American Bar Exam, patterns play a vital role, with prospective applicants learning through experience that out of the four potential options in any given question, two tend to outright contain legally inaccurate assertions, leaving the last two options to test the aspirates legal instincts and reasoning skills.

Regardless of the context, exercises for discovery and step-by-step learning have been shown to assist with learning and understanding complex topics. (Lockwood 2017). Teachers may find that pairing up students in collaborative learning environments may help to make the new skills more comfortable, and for the pairs to help, refresh, discuss, and test each other's knowledge. (See Isbell, et al 2010).

*Chrysanthi Konstanti*
*Eftychia Xerou*
*Centre for Advancement of Research*
*and Development in Educational TechnologyLtd.*

## 4. Abstraction

What is abstracting?

Abstraction is the ability to simplify and generalize complex problems or data to find their crucial essence. It is one of the components of computational thinking, which combines the results of the two preceding processes: decomposition and pattern recognition into a qualitatively new whole that shows the essence of the problem being addressed. Problem abstractions that result from abstraction are definitions of this essence and help to organize and simplify the problem. Abstraction is important not only in computer science, but also in other fields of knowledge, such as mathematics, physics, biology or art. An excellent example is abstract painting. Abstraction helps us discover patterns, rules and principles that govern the world. Abstraction is also part of our everyday life when we use symbols, metaphors, diagrams or models to better understand and communicate with others. Single words, such as a house, a horse, or time, for example, are abstractions of large groups of objects or basic concepts. The ability to extract, verbalize and use abstractions is therefore a key skill, important in teaching various school subjects, as it helps explain and clarify difficult concepts in a way that is simple and understandable to students. Abstraction is also a way to develop creativity and imagination, as it encourages us to create new and original ideas. Abstraction is not only a central concept in computer thinking, but also an important part of education and culture.

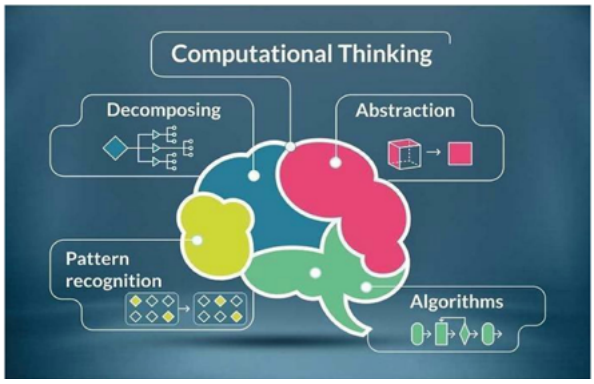



*Figure 1:* Basic elements of computational thinking.
Source: Elements of Computational Thinking. | Download scientific diagram (researchgate.net)

Abstraction is particularly useful in solving complex problems because it allows one to focus on the essential features of an object and ignore irrelevant details that may be hidden, enabling the problem solver to better understand what is going on (Koppelman & Van Dijk, 2010). Gaining experience in problem solving means being able to determine what level of abstraction is appropriate for a given stage of problem analysis, solution design and solution implementation (Haberman & Muller, 2008). Abstraction emphasizes the process of removing details to simplify and focus attention, and emphasizes the process of generalization to identify the common core or essence (Kramer, 2007). According to Koppelman and Van Dijk (2010), abstracting involves the following characteristics: (a) generalization of specific examples; (b) identification, extraction and isolation of relevant components; and (c) ignoring or excluding irrelevant details.

One of the international initiatives aimed at popularizing computational thinking and digital competence is Bebras (https://www.bebras.org), which recognizes abstraction as the most important, though undervalued, competence. Bebras offers the so-called "Bebras challenge," which involves teachers introducing participants to the use of computers or mobile devices. From November 2022 to April 2023, more than 3 million participants from 59 countries took part in the challenge, and it is continuing. Algorithms (66%) and data representation (38%) were the dominant themes of computational thinking in Bebras tasks between 2010 and 2014. Abstracting was identified as an important topic, occurring in 16% of the tasks (Barendsen et al., 2015). While researchers have agreed that abstraction is a central concept in computational thinking, there is no consensus on how to implement it or shape it in education (Cetin & Dubinsky, 2017), although there is no doubt that it is an important part of culture and learning.

## 4.1. The importance of teaching abstraction

Wing (2006) stated that thinking like a computer scientist requires the use of abstraction in many stages of thinking. Abstraction is a key concept and one of the most fundamental ideas underlying computer science and its practice (Armoni, 2013). The concept of abstraction is often used in different ways, depending on the subject being taught. Abstractions in computer science and data analysis are common and obvious (Dorodchi et. al., 2021). However, students should be aware of the different levels of abstraction, i.e. the degree of generalization of information, and recognize the advantages of consciously moving between levels of abstraction when necessary, because of the need to include certain details (Hazzan, 2008).

## 4.2. Challenges in teaching abstraction

The age at which children can understand the concept of abstraction and learn to abstract is sometimes determined by Piaget's classic work, often cited in the education literature. Piaget suggests that children cannot learn to abstract until they reach

the fourth stage of cognitive development, the stage of formal operations. This usually happens around the age of twelve (Cetin and Dubinsky, 2017). The hallmark of the formal operations stage is abstract thinking, which allows one to cross the boundary of time and space, with an understanding of the constancy of certain general properties of objects, such as mass or volume, retained despite a change in the shape of the object, e.g. a ball of plasticine rolled out into a roller or a pancake. Teaching abstraction to novices is therefore a very difficult task, according to many experts (Armoni, 2013). Even computer science students tend to lower the level of abstraction, or more accurately, they unconsciously use cognitive mechanisms that allow them to make concrete sense of abstract concepts (Hazzan, 2008).

It is very important to teach general abstract concepts rather than partial abstract concepts, as this may not be enough to understand abstraction mechanisms (White & Mitchelmore, 2010). It is especially important for novices to develop abstraction skills, which poses a challenge to them. Spontaneous use of abstraction is difficult for novices because of the natural tendency to rely on a familiar procedure when we need to solve a new problem (Koppelman & Van Dijk, 2010).

The issue regarding the teaching of abstraction is whether to devote an entire course to the idea of abstraction, or to mention abstraction on various occasions in other courses when appropriate (Hazzan, 2008). Ensuring that abstraction is properly implemented in education lies in its good understanding by teachers. In particular, it is very important for teachers to be able to determine whether a problem-solving algorithm is an abstraction itself, whether it is necessary to abstract at all in order to create a design, or whether the overall program of conduct is at a different level of abstraction than the problem-solving algorithm (Waite et al., 2018).

## 4.3. The need to teach abstraction

Successful mastery of abstraction requires, firstly, knowledge of abstraction itself, and secondly, teaching the process of abstraction and how to consciously move between levels of abstraction (Hazzan, 2008). Students' awareness of the nature of abstraction, the existence of different levels of abstraction, and the skills and mental processes that enable them to abstract and move between levels of abstraction can enhance their educational and, in the future, professional skills (Hazzan, 2008). Research indicates that understanding *Levels Of Abstraction* (*LOA*) and the ability to move between levels is also essential for success in programming (Waite et al., 2018).

## 4.4. Teaching abstraction

According to Hazzan (2008), students must first see and experience abstraction, and only then can they abstract the first general ideas from the set of elements obtained through decomposition and subjected to grouping during pattern recognition. Based on this approach, the authors suggest the following three ways in which general abstraction ideas can be presented to students (Figure 2).

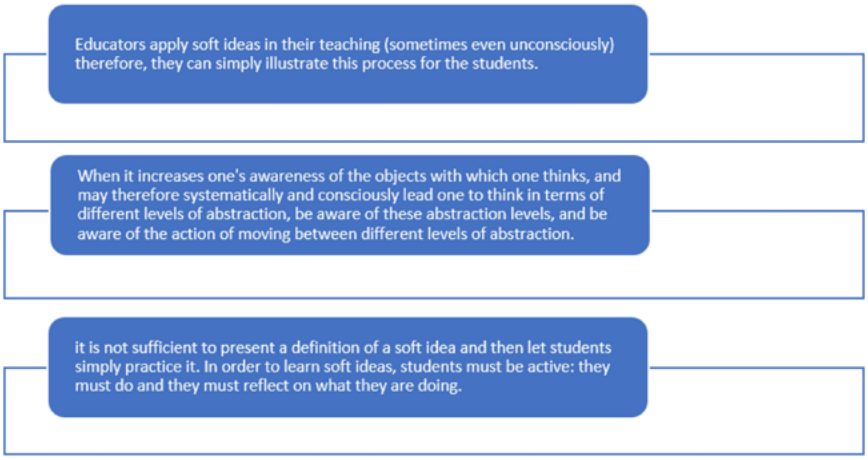



Figure 2: Teaching abstraction.

The results of the research presented in White & Mitchelmore's (2010) article showed that the four-step model for teaching abstraction provides effective mastery of abstract general concepts for students on popular topics in the middle school mathematics curriculum. The research also sheds light on what makes up the four phases shown in Figure 3 and the challenges of putting them into practice.

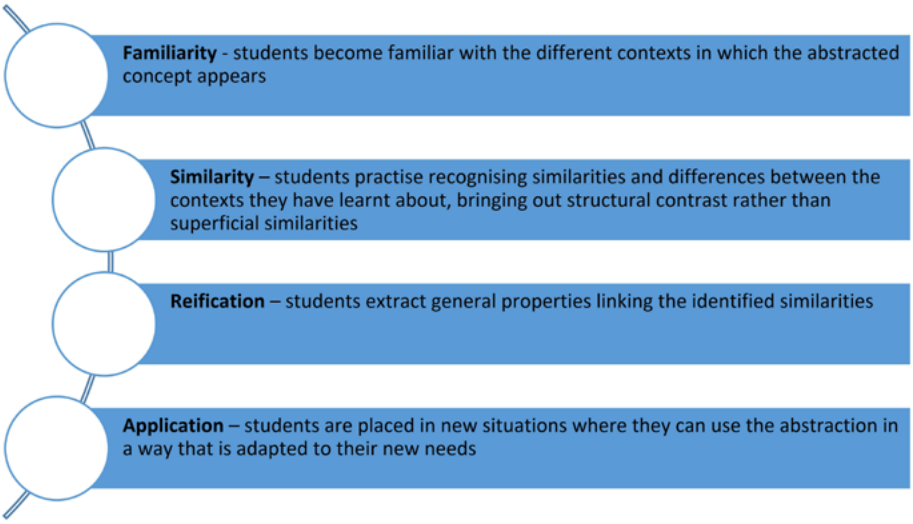



Figure 3: Teaching abstraction.

Haberman and Muller (2008) presented two approaches to teaching abstraction.

1. Pattern-oriented teaching shows abstraction patterns at the beginning of classes. Topics introduced later are organized around the patterns, but their content does not relate to abstracting, only to the subject matter.
2. The use of abstract data types (ADTs) in problem solving and knowledge representation is dominating the curriculum.

It can also be helpful to encourage the creation of a model of the phenomenon under discussion. Preparing a model requires obtaining a comprehensive step-by-step picture of the abstract concept being modeled and requires extracting the essence of the concept beforehand while understanding what needs to be done (Dorodchi et al., 2021).

## 4.5. Abstraction in practice

Abstraction, in computer science terms, is a simplified set of actions or features of a program that explains its essence, leaving out unnecessary details and technical complexities. Abstraction allows an efficient transition from many details to the general. Examples of abstraction:

1. **Baking a cake:** divide the baking process into steps, identify common ingredients and procedures, and create a generalized set of instructions that others can follow. An example abstraction (the essence) of baking a cake is: measuring the ingredients into a bowl, mixing them, and placing the dough-filled baking pan in the oven for a certain amount of time. By using such an abstraction, after adding the details of the recipe, you can successfully bake a cake.
2. **Driving using maps:** Public transportation maps are abstract. Maps only show important information, such as stops and general direction, and leave out the finer details. By omitting the details, you can intuitively determine your preferred route from the many options available, rather than evaluating every turn or potential reroute.
3. **Describing a bicycle**: When we talk about a bicycle, we tend to use selective information to describe it. We usually mention its type, such as a mountain bike or a road bike, and give a basic description of its color. However, if we are talking to someone who is really interested in bicycles, we can go into further detail. We can discuss frame material, tire size, gear ratio, suspension type, and even make and model ("Examples of Abstraction in Everyday Life", 2022).

Abstraction is also worth showing using programming languages as an example. Most programming languages are sets of abstract commands, hiding the complex binary instructions used by computers. Programming languages also offer commands and keywords, such as "print" and "load," that abstract their purpose or expected result from a sequence of complex tasks. In the popular children's programming

language *Scratch*, there are various "bricks" - graphic blocks that are abstractions of background operations. These blocks (Figure 4) visually represent the steps to be performed, allowing users to intuitively understand the flow and purpose of the process (Parry, 2021).


*Figure 4:* *Scratch* blocks showing steps that are hidden from the user in their signatures.

*Chrysanthi Konstanti*
*Eftychia Xerou*
*Centre for Advancement of Research*
*nd Development in Educational Technology Ltd.*

# 5. Algorithmization

## 5.1. Alghorithmic thinking definition

**Algorithmic thinking** is the ability to construct solutions to problems in a way that produces repeatable results in a variety of fields, not just science, mathematics and logic (Mezak & Papak, 2018). An algorithm is like a recipe for preparing a dish, where each step is precisely described and must be done in a specific order (Peel & Friedrichsen, 2018).

Algorithmic thinking is the ability to identify the basic steps to solve a given problem in a way that is appropriate to its specifics and takes into account possible special and typical situations in order to ensure the best performance of the algorithm (Hromkovic et al., 2017).

## 5.2. The role of teaching algorithmization

Algorithmic thinking is both one of the pillars of computer science and a core competency (Malik, et al., 2019). It is a skill that requires the flexible application of knowledge from different disciplines to solve everyday, real-world problems (Sarı, et al., 2022). In the current digital age, algorithmic problem solving is considered an important skill for both society and in any work environment, making it a core competency (Evripidou, et al., 2021).

## 5.3. Challenges in teaching algorithmization

Many students find algorithms a difficult and unattractive subject. Often, traditional courses focus on learning specific algorithms that are considered important in education or in practice (Futschek & Moschitz, 2010). Dugan (2020) emphasizes the importance of training teachers in algorithmic thinking and states that it is very important that teachers, who have an unassailable place in the education system, are well trained, as they are expected to prepare students, who are the other basic element of the system, for the future.

## 5.4. The need to teach algorithmization

There is a need for practical research on how to develop algorithmic thinking and what kinds of activities and educational content can be used in classrooms (Sarı et al., 2022). Algorithmization skills are becoming increasingly central to various aspects of our daily activities, especially after being enhanced by the power of computers and robots (Evripidou et al., 2021).

## 5.5. Teaching algorithmization

Introducing students to algorithmization should take place at a very early stage of teaching computational thinking, using the spiral curriculum of Hromkovic et al. (2017). Students should practice designing algorithms in a structured way from the very beginning, refining detailed competencies in later stages of education. In a spiral curriculum, algorithms are introduced in elementary school, and their improvement continues in secondary education. The authors point out that problems given to students as tasks to be solved should not be too simple, but their description should be easy to understand.

Visualizations of algorithms are also a teaching method that is very useful and effective (Futschek & Moschitz, 2010). According to the authors, learning the principles and concepts of algorithmization with the use of visualizations is much easier for students to understand, and makes activities using visualizations prepared in such a way that they are fun for students. The teacher's task in this method is to formulate problems that are appropriate to the student's level of competence and ask questions that encourage students to think towards creating correctly working algorithms to solve these problems. The teacher also motivates students to improve their algorithms to find even more effective solutions. Futschek (2006) suggests a wide range of computer science topics that can encourage students to use algorithmic thinking, especially with the use of visualizing algorithms in the form of a program-tool or a game made by the students themselves.

It is worth noting that teachers play a very important role as personal role models of algorithmic thinking. Teachers participating in one study (Dugan, 2020) confirmed that with well-developed algorithmic thinking skills, they tend to teach in spirals, follow a process that facilitates learning, encourage students to plan well and develop algorithms neatly, and help them develop/improve their algorithmic thinking skills. They also suggested using techniques such as discovery learning, problem solving, induction, brainstorming, concept mapping, games, discussion, fishbone and case study, which require students to be actively involved in the learning process to improve their algorithmization skills. In teaching algorithmization, one can also use the image of a box of bricks, in which only a few basic elements are available, from which children can create both simple and complex structures (Milkova, 2012).

## 5.6. Resources to support the learning of abstraction and algorithmic skills

Teaching activities on computer thinking, including abstraction and algorithmization, can be conducted with or without computers, software or digital tools. Activities without computers ("unplugged") focus on developing computer thinking skills through tangible and interactive exercises, using paper instructions, visualizations, figures and physical movement as representations of operations performed by computer programs (Sigayret et al., 2022). However, the most common approach used is a combination of "connected" and "unconnected" approaches.

Educational tools that support the formation of competencies related to computational thinking and alliteration discussed in this chapter include "cMinds". The cMinds program is an educational intervention that provides logic challenges to develop problem-solving skills and analytical thinking abilities through game-based learning. The cMinds Learning Suite encourages students to analyze problems, identify the basic components of a solution, critically combine different components, optimize their solutions and reflect on their thinking process (Tsalapatas et al., 2012).

*TeaEdu4CT* is one of the projects combining computer-free and computer-enabled approaches that focuses on innovative educational methods for computer thinking (CT) in the context of transdisciplinary and holistic STEM (science and technology subjects) perspectives in teacher education (https://www.fsf.vu.lt/ct-4teachers#about-the-project). The project provides a set of tools, techniques and approaches that facilitate a seamless transition from computer- and Internet-free activities suitable for young children to advanced computer modeling and simulation for high school and early college students.

*The Tech Interactive*, an organization driven by a mission to promote computational thinking (CT), creates and distributes high-quality STEM educational resources that are attractive to students, allowing them to quickly integrate elements of computational thinking into their teaching. For example, abstraction, which describes natural phenomena with succinct statements, is practiced through a wide range of experiments. Algorithmization, on the other hand, is a set of instructions, or "procedure," aimed at performing one of these laboratory experiments in the classroom. Examples of educational resources, lesson plans and activities for teachers and students can be found on the Tech Interactive website, https://www.thetech.org/educators-students/resources/lessons-activities/computational-thinking/.

## 5.7. Algorithmization in practice

On a daily basis, we encounter different types of algorithms that perform specific steps (linear processes), make decisions (conditional algorithms) or repeat actions a certain number of times (looped algorithms). Algorithmization seems like a complex process, but algorithmizing many everyday activities comes naturally to us. Here are three examples to cite:

1. **Tying shoes:** Step-by-step, often automatic activities that are consistently performed in the same way many times, such as shoe tying, can be considered an algorithm. The process of obtaining a traditional shoelace knot, often referred to as a loop or a bow, can be done with a very simple, limited set of steps: "loop, pull through eyelet, tighten".

2. **Recipe algorithm:** Any simple recipe for cooking or baking can be considered an algorithm. In a recipe there is a sequence of operations, a description of the conditions, a repetition of the steps needed to successfully complete the recipe. Algorithmizing recipes is a useful exercise, promoting algorithmic thinking and demonstrating the ubiquity of algorithmization.

3. **Driving to or from some place:** The act of driving, for example from home to school, can be an algorithm. Like any other algorithm, this routine can collide with obstacles, such as roadworks or heavy traffic, which can prompt decisions based on the conditions encountered. If there are roadworks on street X, turn right ("7 Examples of Algorithms in Everyday Life for Students," 2023).

*Panagiotis Psomos*
*Innovation Frontiers*

# 6. The CTApp Game. The idea, the structure and its functionalities

The development of a new mobile app to strengthen computer thinking skills can support education and empower students. The CTApp app aims to engage and motivate students, especially those who may be disadvantaged, to acquire basic computer thinking skills. By fostering a love for problem solving based on a computational approach and preparing students for an increasingly technology-driven world, the app aims to make a lasting impact on their lives and their future. What is more, this innovative approach to education aims to improve teachers' professional development and equips them with the tools they need to introduce innovative classroom practices in collaboration with their students.

## *The idea behind the CTApp Game*

Computational thinking skills have become important for students to thrive educationally, professionally and personally in today's rapidly changing world. The ability to grasp and apply the ideas of computer thinking has become a key skill set as the digital world continues to change our civilization.

Computational thinking is a methodical approach to problem solving that is based on the principles of computer science and logic. It allows people to deconstruct difficult issues into manageable components, analyze data and develop new solutions. Students gain competence in dealing with the complex challenges of today's world by developing computer thinking skills.

The ubiquitous integration of technology in many areas is one of the primary reasons why developing and strengthening computer thinking skills is crucial. Technology permeates virtually every business, from artificial intelligence and robots to data analytics and cyber security. Computational thinking skills provide students with a foundation for effectively understanding and using technology, enabling them to adapt to the ever-changing demands of the digital age.

Moreover, computer thinking encourages critical thinking, creativity and logical reasoning. It teaches students to approach issues methodically, see patterns and create efficient algorithms. These cognitive skills go beyond computer science and can be applied to many fields, including mathematics, physics, business and the humanities. Students with computational thinking skills are better prepared to deal with difficult issues, make informed decisions and make important contributions in their chosen fields.

One of the main goals of the application is to improve teachers' professional development. By providing a comprehensive tool to support the teaching of computational

thinking, teachers can gain valuable insights into the subject and develop their own knowledge. The app offers teachers a platform to expand their knowledge, learn new teaching strategies and stay abreast of the latest developments in teaching computational thinking. This professional development not only benefits the teachers themselves, but also has a direct impact on the quality of education they provide to their students.

Continuous professional development is essential for teachers to remain effective and adapt to the changing needs of their students. In this regard, the app serves as an aid to teachers. Teachers can deepen their understanding of computational thinking concepts and improve their teaching practices. The emphasis on ongoing professional development allows teachers to become educational leaders and mentors, which ultimately benefits both themselves and their students.

In addition to simulating professional development, the CTApp app aims to support teachers in undertaking innovative and collaborative practices in the classroom. By facilitating the implementation of innovative teaching methods, the app encourages teachers to think creatively and adapt their teaching methods to meet the diverse needs of their students. This support for collaboration and innovation contributes to the development of a dynamic and engaging learning environment that effectively reinforces computational thinking skills.

The development of a mobile app dedicated to strengthening computational thinking skills is a significant step toward equipping students with the tools they need to succeed in the digital age. By activating teachers' professional development, supporting their continuous learning, and promoting collaboration and innovative practices, the app enables teachers to become catalysts for change in the education system. Ultimately, the initiative aims to close the gap in education in terms of computer thinking, especially for disadvantaged students, and ensure that all students are prepared to thrive in the world dominated by computers and technology. By providing a robust tool to support school curricula and enriching the toolkit for teachers, this application can improve education and shape a better learning environment.

## *The structure of the CTApp Game & its Functionalities*

### *Entering the CTApp Game*

The CTApp game is set in an escape room (puzzle room) and is readily available to both Android and iPhone users via the Play Store and App Store, respectively. When users open the game, they are greeted by the distinctive CTApp project logo, which is complemented by a background depicting a learning area. In addition, a set of logically arranged buttons is visible to help smoothly navigate the app.

Key among the buttons is the Start button, which acts as an entry point into the world of question-based challenges. If users decide to leave the app, an exit button is prominently placed to ensure a smooth exit. The app also has a settings button that allows users to personalize their experience. Users can easily change music and sounds using this button to adjust the sounds to the way they like it. When users

launch the app, music accompanies them as they play, improving the experience and engagement. Users also enter their name and corresponding avatar at the beginning, which can be changed later.

When starting the game, you can also choose to play with another person. To do this, press the "Multiplayer mode" button. Two characters can be selected for gameplay, whose personalization is carried out independently. Players play on one device. After one participant completes the level, the game can be started by the other student. Scores are calculated independently for each participant.

Users can easily use the app thanks to the availability of CTApp Game on Android and iPhone platforms, a user-friendly layout and a convenient set of buttons for starting, exiting and changing settings. The intriguing background of the learning area sets the tone for the experience, while the ambient music adds a touch of calm and positive emotions. CTApp Game ensures that each person can tailor their engagement with the app to their own tastes and preferences, giving users a choice.

CTApp Game is designed for a diverse audience, with versions available in English, Italian, Polish and Greek. Users who speak these languages will be able to access the app and its content in their native language. The app is user-friendly and offers translated versions of the questions, providing a positive experience for a range of diverse users.

### *Navigating through the Menu*

Students are greeted with an intuitive menu interface that appears on the screen when they click the Start button to begin the main game. This menu acts as a portal to four distinct stages, each of which represents one level in the game. The first level consists mainly of theoretical questions about computer thinking and its key stages, which require users to demonstrate their expertise by providing correct answers. As users progress, the next three levels provide a series of practical problems that must be solved to successfully complete the game.

Practice questions are assigned to the three levels according to their varying levels of difficulty. As a result, each successive level provides slightly more difficult questions than the previous one, ensuring gradual learning. To progress through the game, users must successfully complete each level before moving on to the next one. When users begin using the app, they only have access to the first level, which consists of theoretical questions and requires informational preparation of users during introductory computer thinking classes.

Users are given a clear and chronological path to follow. Going through the game in this way allows users to get a sense of success as they complete each level. The progression system ensures that users continually improve their understanding of computer thinking and their ability to use the competencies they acquire, allowing them to tackle increasingly complex problems as they progress through the game.

In summary, when users launch the main game by pressing the Start button, they are greeted by a user-friendly menu screen with four separate levels. The first level contains theory questions, while subsequent levels contain increasingly difficult

practice questions. Users have the option to move to the next level after successfully completing the previous one. This sequential structure promotes skill development and provides a rewarding experience for users who want to improve their knowledge and competence in the game. With its hint system, the game can be a standalone resource for inquisitive users, but it can also be woven into the educational process as a support tool, used as part of homework assignments or even in lessons, thanks to its multiplayer mode.

### *Playing in the CTApp Game*

When users enter the first level, they are greeted by a study area equipped with ordinary items such as desks, chairs, bookcases, globes, lamps and more. Each area in the game has a specific theme and is decorated with various ordinary objects. One of the areas contains mathematical symbols, such as the plus sign and the pi symbol. The varied designs of the areas provide users with a stimulating experience while going through the game.

Navigation assistance is provided by a button in the lower left corner of the screen that allows users to move the avatar right, left, backward or forward. Touching and moving the screen in other places allows users to change their perspective, allowing them to look up or down the room and rotate the view.

Upon entering the study room, the timer starts counting down five minutes. During this time, users must complete answering a set of questions prepared exclusively for that level. Some items in the study room are highlighted because they act as "key items" that contain sets of questions. When an item is marked as a "key item," a green box appears above it, reflecting the number of questions correctly answered in that set in previous attempts. For example, if no questions in the set were answered, a message will appear above the corresponding "key item" with the name of the item, such as: "Projector 0/16". When the name of the item is touched, a series of questions or tasks is displayed.

Some "key items" are immediately visible and highlighted as soon as users enter the area. Other "key items" get marked only when users approach them and then a green box appears representing the number of unanswered questions.

Users can begin answering questions from easily visible "key elements" as soon as they enter the room. After answering a series of questions, users are prompted to explore the survey area further to find the next set of questions, which are stored in other "key elements." A green box above these additional "key elements" identifies them and becomes visible only when users approach them. With the above features of the interactive learning room, users gain control of the game and increase engagement. The game encourages exploration and engagement, motivating users to use the learning space extensively and actively seek out the next set of questions.

In summary, when students arrive at the first room, they are greeted by a learning space containing a variety of items. They are free to roam the room, pressing on-screen buttons to travel in different directions. A countdown begins, signifying the need to complete sets of questions in a certain amount of time. Some items in

the study room are "key items" and contain question sets, as indicated by a green box above them.

Some of the "key items" are immediately visible, while others get the "key item" designation only when users get closer to them. Users can start answering questions from the visible "key items" and then move around the learning space to find further sets of questions placed in additional "key items." This interactive learning room concept emphasizes autonomy, engagement and extensive exploration of the environment.

### *The questions*

When a user clicks on one of the identified 'key elements', the first pop-up question is displayed to them. Each question follows the same structure. The question is clearly displayed at the top of the pop-up and users have a choice of four possible answers. Once the user has made their choice, the pop-up provides quick feedback, showing whether the chosen answer is correct or incorrect. In addition, an explanation box is displayed along with the feedback, showing users the mindset to adopt in order to arrive at the correct answer.

Once all the questions in the set have been answered to a sufficient level of correctness, a pop-up appears with the message 'level completed'. When the user gives too many wrong answers, a pop-up window appears with the phrase 'try again'. This pop-up contains two options: a restart button, which allows visitors to try answering the questions again, and a 'go to home page' button, which displays the main menu to users.

The use of pop-ups provides users with a dynamic and engaging experience. Displaying questions, answer options, feedback and explanation boxes in pop-up windows allow for efficient and accurate delivery of information and engage students in learning new competencies. Whether users answer correctly or incorrectly, the game provides quick feedback and alternatives for next steps, promoting persistence and encouraging learning.

In summary, after selecting a 'key element', visitors are presented with the first question in a pop-up window. The layout is identical across all questions, with the question on the left and four answer options below. Users receive quick feedback on their answer, along with an explanation box showing a suggested thought process. Completion of all questions within a set time period results in a pop-up window commemorating 'completed level'. If a large number of incorrect answers are given, a second pop-up appears, giving the user the choice of restarting the level or returning to the main page. The use of pop-ups throughout the game provides a dynamic experience for users.

Here is an example of a theory question belonging to **level 1** (theory questions):

Which of these phenomena is an example of decomposition?

A. *Writing out the individual ingredients of a dish from a recipe.*
B. *Checking that you have all the utensils you will need to make a dish from a recipe.*

C. *Thinking about different ways to do the cooking according to a recipe.*
D. *None of the above.*

In this example, the correct answer is A. The explanation is: 'Decomposition involves breaking down a problem into smaller parts'.

Here is an example of a question belonging to **Level 2** (questions with a difficulty level of 1).

In order to fly to Mars, follow the steps below in the order given:

A. *build a rocket, determine the flight trajectory, run the engines to full power, upload the flight target data into the computer, calculate the operation cost, develop a detailed plan of operations, gather information on the key flight challenges and risks, assemble the crew.*
B. *develop a detailed plan of operations, gather information on the key flight challenges and risks, assemble the crew, build a rocket, determine the flight trajectory, run the engines to full power, upload the flight target data into the computer, calculate the operation cost.*
C. *gather information on the key flight challenges and risks, develop a detailed plan of operations, calculate the operation cost, assemble the crew, build a rocket, determine the flight trajectory, upload the flight target data into the computer, run the engines to full power.*
D. *assemble the crew, gather information on the key flight challenges and risks, build a rocket, determine the flight trajectory, run the engines to full power, upload the flight target data into the computer, calculate the operation cost, develop a detailed plan of operations.*

In this question, the correct answer is C. Here is an explanation: "Computational thinking involves breaking down complex tasks into smaller, more manageable parts and using a logical approach to solve them. This order of operations can be used to solve real-world problems in learning or in everyday life."

NOTE - the above examples are fictional, such questions do not occur in the game!

The steps of listing all the elements, recognizing the rules that connect the elements, defining the rules to be applied and preparing the step-by-step instructions are based on a computer-based approach to problem solving, with each step following from the previous ones.

### *Completion of the game*

When users have successfully answered the questions in a given level, a pop-up window will appear congratulating them and informing them that the next room has been opened. Users can move on by returning to the main menu and selecting the

next level. There is also a pause button at the top right of the screen, allowing players to stop the game at any time and resume.

With its multi-level structure and dynamic action, CTApp Game is a useful tool for young users who want to improve their computer thinking skills. The software also supports teachers who want to develop computational thinking in the classroom. The game encourages users to think critically, analyze issues and use logical thinking to arrive at answers, offering a range of challenging questions and puzzles. Users are encouraged to improve their problem-solving skills.

*Robert Porzak*
*Lubelska Akademia WSEI*

# 7. Integrating CT with curricula and classes

**Computational Thinking in schools**

The previous chapters described Computational Thinking as a skill that helps students learn the logic and principles behind technology and coding. It is not about memorizing facts or commands, but about thinking critically and creatively. Students can practice computational thinking without using a device, so it can be part of any classroom, even for younger learners. Computational thinking is becoming an essential skill for students, as it prepares them to understand and use the technologies of the future. It is not an additional, time-consuming task for teachers but can be rather integrated into the existing routines and curricula to save time (Yadav et al., 2017).

The reason why computational thinking can save time in the school curriculum is that it can help students learn other skills more efficiently and effectively. For example, computational thinking can help students understand the logic and structure of math, the rules and patterns of languages. By applying computational thinking to these subjects, students can grasp the underlying concepts more easily, avoid common mistakes, and solve problems faster (Flórez et al., 2017).

Computational thinking can help students develop higher-order thinking skills that are essential for the 21st century. These skills include critical thinking, creativity, collaboration, and communication. By learning computational thinking, students can improve their ability to analyze information, generate new ideas, work with others, and express themselves clearly. These skills can help students succeed in various academic and professional fields, as well as in their personal lives (Pérez-Marín et al., 2020).

Computational thinking can help in teaching vocational schools students as well. When applying CT in vocational school, teachers should guide students to think about the intention of the individual step, not make students just follow the steps. "Unlike traditional teaching methods, the teaching method based on computational thinking not only train students' awareness of the computing environment, but also help them to master solving methods of typical computing environment oriented. In a word, if teachers in higher vocational colleges want to teach a course using the teaching method based on computational thinking, they should:

a) decompose the course into knowledge points,
b) determine the emphasis and difficulty of the task, confirm the content the students need to grasp,
c) design the corresponding experimental models, constructing computing environment,

d) optimize the problem solving process,
e) verify the experiment results and check consistency" (Shuiyan He et al., 2014, p. 819).

Therefore, computational thinking can be a valuable addition to the school curriculum, as it can help students learn other skills more quickly and effectively, as well as develop higher-order thinking skills that are crucial for the future. Computational thinking can also be fun and engaging for students, as they can use it to create games, animations, stories, art, music, and more. Computational thinking can be integrated into various subjects and activities in the school curriculum, such as math, science, language arts, social studies, art, music, physical education, and extracurricular clubs. By teaching computational thinking to students, educators can prepare them for the challenges and opportunities of the digital age (Grover & Pea, 2018).

You might think that these elements are too advanced for younger students, but they actually fit well into the active learning and thinking that happens in primary or secondary school grades. Children love to play and explore. They are not afraid to try new things. By using their natural curiosity and problem-solving skills, we can help them develop computational thinking. Computational thinking makes learning fun, but also gives it structure so that the skills students learn can be applied to more complex tasks later on. And you might be surprised by how easy it is to invite your young learners to join you as inventors and problem-solvers and dive into the world of computational thinking. It is also good to mix the approach focused on teaching computational thinking with the project-based learning method integrated with the teaching material of robotic visual programs approach. Such a mixed method approach has significantly better effectiveness in improving students' learning achievements than the traditional teaching method integrated with paper practice teaching materials approach (Hsieh et al., 2022).

We live in the world with Smartphones and Smart Homes, and understanding how devices work allows us to use technology as a tool to help us solve problems. Computational thinking allows students to be active, rather than passive users of technology. The way we understand the technology around us, and the way we ask questions about these devices, will be a key factor in the 21st-century workforce. Those who can do it well and efficiently will have more opportunities for both professional and personal success. This preparation can and should start with our young learners (Ung et al., 2022)focusing on empowering digital literacy. Nevertheless, a preliminary investigation revealed an apparent lack of understanding of computational thinking skills in general among teachers. The study explores the feasibility of developing a localized E-learning system to train computational thinking skills among teachers. An E-learning system, termed as myCTGWBL, was developed on the basis of a newly proposed conceptual framework to present computational thinking teaching–learning repertoire to the teachers. The hypothesis is that myCTGWBL would develop teachers' computational thinking and its position in teaching–learning understanding. myCTGWBL relevance was tested through DeLone and McLean's information system and Urbach's collaboration quality construct. To determine the

success factors, partial least squares structural equation modeling was used. A total of 369 teachers participated in a two-stage survey. Participants' understanding of computational thinking and perceptions were recorded at the pre- and post-intervention phases. Open-ended questions of the surveys were analyzed using a simple text analysis technique. The closed-ended questions surveys were analyzed using SPSS Statistics 22.0. A significant improvement in teachers' computational thinking teaching–learning repertoire in a relatively short period has been recorded. Teachers also demonstrated increased confidence in the future delivering computational thinking-based lessons. The E-learning conceptual framework has illustrated the predictive power between user intent, user satisfaction, and Computational thinking (CT.

**Strategies for teaching Computational Thinking in classes**

What are some effective ways to teach computational thinking? Authors suggest to apply some specific computational strategies based on a review of the literature (Hsu et al., 2018). We can identify recommendations coming from classic approaches like the constructivist theory formulated by Dewey Piaget, Vygotsky, Bruner and Glasersfeld (Csizmadia et al., 2019; Oktan & Vural, 2021) and some other most frequently used strategies as problem-based learning, project-based learning, collaborative learning, and game-based learning (Ghani et al., 2022; Yadav & Berthelsen, 2021).

*Problem-based learning*

Problem-based learning is the most popular strategy for enhancing computational skills. In this method, teachers present a real-world problem that students attempt to solve using their prior knowledge and experience. Similar to competency-based education, the activities here focus on developing a set of skills that they can apply to real-life situations.

Computational thinking problem solving can help students to think critically, ask the right questions, and generate multiple solutions on their own. It can also help teachers guide the discussions on the problem at hand (Tekdal, 2021).

*Project-based learning*

In project-based learning, students are also given a problem to solve. The main difference is that they are expected to produce an output that represents their solution.

A supporter of the project-based learning, John Dewey, advocated "learning by doing." As students explore the possible solutions to a problem and create a project, they also develop critical thinking, communication skills, and collaboration skills.

Teachers can use project-based learning to foster computational thinking among students. Thinking computationally can help students come up with a systematic approach to designing a solution (Belmar, 2022).

*Collaborative learning*

Unlike the first two strategies, collaborative learning emphasizes more the idea of working together to solve a problem. Students are challenged to grow not only

intellectually but also socially and emotionally. In the process, they learn skills that they can use in real-world work scenarios, where communication and collaboration skills often give an individual an advantage.

Promoting computational thinking in a collaborative learning environment means that students will be exposed to other viewpoints. They will get a chance to work with their peers to solve more complex problems that may be difficult to solve all by themselves (Neumann et al., 2021).

*Game-based learning*

Now, you may be wondering how to teach computational thinking in a game-based learning. Well, teachers design games with a set of learning objectives in mind. Games can come in various forms, such as board games, card games, role-playing games, and puzzles. One popular option today is digital game-based learning, which some argue, is the future of learning.

Teachers can design or source games specifically designed to enhance computational thinking skills. In a game-based learning setup, students will be more engaged and free to make mistakes without risks or serious academic consequences (Hsu et al., 2018).

**Teaching elements of Computational Thinking in classes**

A method we use to teach computational thinking depends on specific skills we want to focus on. By utilizing the natural inclinations of young children to explore and play, and by encouraging problem-solving skills, we can move students' thinking forward. Practical suggestions for early education are presented by Kristen Thorson and some other authors (Ezeamuzie et al., 2022; Lee et al., 2023; Munn, 2021; Thorson, 2018).

*Teaching Decomposition*

When you teach decomposition to young learners, you ask them to join you in solving problems. You share a complex, multi-step problem and help them to break it down into smaller parts. Even though students at these ages are not always ready for multi-step directions or problems, they are ready to see how adults think. By doing this, you help them to build a framework of strategic, computational thinking.

Ideas to Try: You might use a scenario, such as planning a birthday party, that has many steps. This kind of task can be hard to do without a plan of smaller, more manageable tasks. Students can help you to split up the big task, and you can help them to draw or write a picture of their thinking. This gives them a way of thinking about how to solve similar problems in the future (Munn, 2021; Thorson, 2018)intriguing, and inviting to students. The constructionist philosophy, hands-on application learning, addresses social skills like collaboration, communication, and teamwork to provide an authentic, real-world learning experience. CT brings to the classroom exciting and innovative activities that infuse robotics with hands-on application platforms in the science and mathematics curriculum, but the education system has missed

a core set of young open-minded eager students at the intermediate school level. With today's vision in education focusing on the 21st-Century learner, CT is emerging as a key component in the skill set necessary for the new generation of learners. CT poses a strong ideology based on problem-solving equally conveying an essential position in cross-curricular classroom activities. This session exposes CT through a study relating experiences and interactions by students when engaging in a science lesson utilizing robots. Focusing on how students engage the CT key concepts of: (1.

*Teaching Pattern Recognition*

Pattern recognition is one of the key parts of computational thinking. It starts with the basic ABAB pattern taught in the primary grades and goes on to more complex levels of thinking. Pattern recognition asks students to look at similar things or experiences and find what they have in common. By finding what the things or experiences share, young students can start to understand trends and make guesses about what will happen next.

Ideas to Try: To teach students to recognize patterns, you might start by looking at trees. What do all trees share? They all have a trunk. They all have roots. They all have branches. Even though there are many differences between kinds of trees, these parts are in all trees.

Next, work with your students to make a picture of trees. Notice how they all have trunks, roots, and branches. Then, talk about how the trunks are different from each other. Some are thick, while others are thin. Some are brown, while others are white. Talk about how roots and branches are different. To go further with this thinking, ask your students to draw a picture of a tree, labelling a trunk, roots, and branches. Emphasize that while your class trees might look different from each other, they are similar in their main parts.

Finding patterns makes tasks easier because you can use what you already know. By teaching students to recognize patterns, their awareness of the world around them grows. This helps them to use the patterns they have found to solve future problems and make guesses about the world (Ezeamuzie et al., 2022; Thorson, 2018).

*Teaching Abstraction*

Abstraction is focusing on the information that matters and leaving out the information that does not matter. It involves separating the main information from extra details.

Ideas to Try: In primary classrooms, teachers naturally teach kids the idea of abstraction with books as they find the main idea and key details. To go one step further, teachers can encourage students to search for information, clues, or treasures by giving them a goal as they read a book or even have an experience. As students listen to a speaker during a school presentation about dental hygiene, a kindergarten class might be looking for details about brushing one's teeth. By teaching students abstraction, they are able to sort through all the information available to find the specific information they need. This is a very useful skill as students read bigger texts and are given more and more complex information (Lee et al., 2023; Thorson, 2018).

*Teaching Algorithms*

Algorithmic thinking means providing solutions to a problem. It means making a list of rules to follow in order to solve a problem. In the early grades, kids can learn that the order of how something is done can matter.

Ideas to Try: To show this idea to students, you might ask them to think about making a sandwich. What do we need to do first? Next? What if I put the mayonnaise on my sandwich before I add the cheese and lettuce? Talking about the sequence and order helps students to build the basics of algorithmic thinking.

To get students thinking in algorithms, ask them to plan the path from their classroom to the gym by writing down a series of steps. Then, let them try it out! Also, ask them to think about their morning routine. What steps do they take to get ready for school each morning? How would the order affect the result? Asking students to think about how changing the steps changes the outcome helps them to be thoughtful in their thinking and to make changes to their plan to succeed (Thorson, 2018; Whitney-Smith, 2023).

**Digital tools useful in teaching Computational Thinking**

Teaching computational thinking can be supported by platforms that can help you integrate computational thinking into your curriculum in a fun and engaging way. These platforms are:

- Ozobot: https://ozobot.com/ Ozobot is an innovative platform that allows you to teach computational thinking using robots. Ozobots are smart robots that can follow lines and commands drawn by your students using different colors. Ozobot also provides video lessons on various subjects that you can share with your students in grades 3-5. These lessons help your students learn how to code the robots using visual codes and apply computational thinking to different disciplines.
- Scratch: https://scratch.mit.edu/ Scratch is a free and easy-to-use platform that lets you and your students create interactive stories, games, and animations using colourful blocks of code. Scratch supports more than 70 languages and offers a variety of resources for educators, such as tutorials, coding cards, lesson plans, and strategies. You can also use Google's CS First curriculum or Code Club's project modules to teach Scratch in your classroom. With Scratch, you can inspire your students to express their creativity and collaborate with others while learning computational thinking.
- CTApp: https://ctapp.eu/ CTApp is a new mobile game in an escape room format. The aim of the player is to answer a series of questions and solve problems. The game combines the concepts of Computational Thinking and Serious Game. The game can be used by primary school students aged between 12 and 18 who speak English, Greek, Italian or Polish. Pupils will be able to use the solutions developed in the project in their school subjects.
- Kodable: https://www.kodable.com/ Kodable is a comprehensive coding platform that provides you with a complete curriculum to teach computational thinking. Kodable helps your students learn how to code using JavaScript, Python, HTML,

CSS, Java, and more. Kodable also helps your students develop other skills, such as teamwork, resilience, communication, and design. Kodable is available for iOS devices and desktops, and each lesson includes instruction guidance, vocabulary words, and materials to help you teach effectively.

**Assessment of computational thinking in education**

Computational thinking (CT) is seen as a key competence of the 21st century, but there is no universal model of assessment of CT competencies in education. CT competency can be evaluated by using: questionnaires, tests/tasks, observations, interviews and analysis of products. There are also some tools not related directly to CT assessment, but assessing dimensions connected to CT, like logical thinking / reasoning, data analysis, critical thinking, automation, cognitive planning, coordination, computational concepts and many others. The systematic review of the tools shows that they do not refer to a shared competence model of CT differentiated by age. The different studies assess CT in pupils in different grades, however it is not clear what competences pupils should reach at which age since the same dimensions and tools are used for pupils of different ages (Babazadeh & Negrini, 2022).

A set of sample lesson plans can be a useful aid for implementing computer thinking into the practice of a teacher's daily work with a class. The scenarios included below have been tested by teachers and contain indications for teaching computer science subjects, as well as literature and mother tongue. The suggestions indicate the possibility of spreading the activities over several lesson units or implementing them in a single lesson. We also encourage all teachers interested in computer thinking to share the lesson ideas they have developed.

*Robert Porzak*
*Lubelska Akademia WSEI*

## 7.1. Scenario 1

**Author / Teacher:** Robert Porzak

**Subject:** IT Classes, Computer subjects, etc.

**Level** (ISCED, difficulty)**:** ISCED level 3 – Upper secondary education

**Topic:** Become a spy (basics) – hide a message in a picture

**Prerequisite skills or knowledge** (connected to the prior lesson):

a) Graphic program basic knowledge (GIMP or similar)
b) Understanding what an auto-stereogram is (to find out more, read more about auto-stereograms: https://en.wikipedia.org/wiki/Autostereogram

**Time required for a pre-class activity:** 2 hours
**Time required for an in-class activity:** 1 hour
**Time required for a post-class activity:** 2 hours

Story, canvas, challenges for student (optional, motivational)

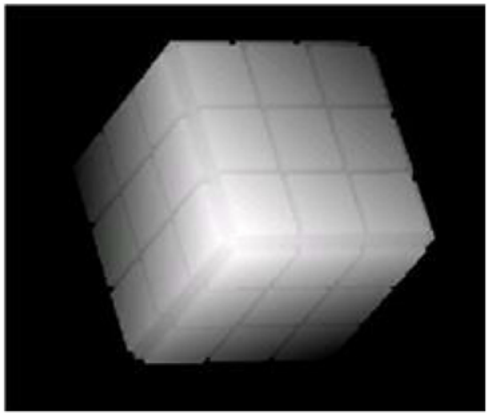
We all know that it is difficult to be a spy. One of the core competencies of a good spy is to make communications secret. One can try to use modern computer codes or special electronic tools to encode communications, but the recipient will then have to rely on a computer or electronics to decipher the message. However, there are visual coding methods that are not easily deciphered by unauthorized people who do not know the mechanism of the process. Your secret message encoded in this way may be visible to everyone, but only informed people can decipher it. Are you ready for a lesson on how to become a spy sending hidden messages in images?

1. **Student's new material (before the class)**
   a) Learn how to prepare in a graphic program a flat (2-dimensional) pattern map (a picture made of small, coloured dots/pictures/patterns). Sample 2-dimensional patterns should look like those shown below:

Source: https://www.easystereogrambuilder.com/

Learn how to do it in free GIMP https://www.gimp.org/ or any other software. You can see how to do it, for example here: https://www.youtube.com/watch?v=TKhs7F0hAik

b) Learn how to prepare a depth map understood as a simple black and white picture looking like a negative, blurred. A sample depth map can look like this one shown below:



Source: https://www.easystereogrambuilder.com/

Learn how to do it in free Blender https://www.blender.org/ or any other software. You can see how to do it for example here: https://www.youtube.com/watch?v=3P6DVDy6DG0

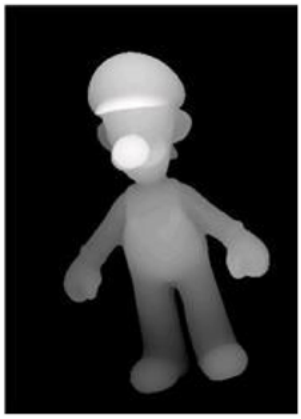
**2. In-class activities**

a) Prepare a flat (2-dimensional) pattern map (a picture made of small, coloured dots/pictures/patterns). A pattern should be an image with colour variations. The most important thing is that it should not have horizontal regions (strips) with the same colour:

Source: https://www.easystereogrambuilder.com/

You can use free GIMP https://www.gimp.org/ or any other software.

b) Prepare a depth map of the corresponding size as a colour map (a simple black and white picture looking like a negative, blurred). The whiter the points, the closer the points are to you. So, black represents the background, and the different shades of white represent points floating above the background:



Source: https://www.easystereogrambuilder.com/

Use free Blender https://www.blender.org/ or any other software.

c) Cover both images with oversampling (adding depth) in the programs mentioned. You can also use GIMP's plugin, if you want to try: https://gimplearn.net/viewtopic.php?t=2974

d) Write a block algorithm showing steps necessary to do a text message hidden in graphics.

**3. Post-class activities**

a) Verify your algorithm preparing two text messages hidden in auto-stereograms. Follow your algorithm carefully, step by step, checking if this is universal enough.

b) Write your conclusions on the possibility of coding algorithms hiding text messages in pictures as a paragraph of about 50 words.

**4. Evaluation and Assessment**

Learning outcome – student develops correct algorithms adequate to the task:

√ E - for simple problems given an algorithmic technique
√ D - for non-trivial problems with a given hint
√ C - for non-trivial problems
√ B - for harder problems using the best method with a given hint
√ A - for harder problems using the best method

*Fahimeh Mousavi*
*INNOVA S.r.l.*

## 7.2. Scenario 2

**Author / Teacher:** Fahimeh Mousavi / Cristina Fregonese

**Subject:** Arts, Language Arts and Media Studies and Character Education

**Level** (ISCED, difficulty)**:** ISCED level 3 – Upper secondary education

**Topic:** Storyboards through video games

**Prerequisite skills or knowledge** (connected to the prior lesson):

a) Understanding what a storyboard is (https://en.wikipedia.org/wiki/Storyboard)
b) Basic knowledge in writing skills

**Time required for a pre-class activity:** 2 hours
**Time required for an in-class activity:** 1 hours
**Time required for a post-class activity:** 2 hours

Story, canvas, challenges for student (optional, motivational)

Because video games are a visual medium, storyboards are a great way to map out the story being told through the video game. Students will apply and share their ideas, personal perspectives, and creative thoughts through the storyboard activity. In particular, students will have the opportunity to map a story out using storyboards and demonstrate how visual cues can help communicate feelings, ideas and understanding when telling a story. Each student will use a storyboard to share a story they create.

**5. Student's new material (before the class)**

Vocabulary to be used:

POV (Point of View): POV, or the point-of-view shot, which allows the audience to see what is going on through a character's eyes.

Master Shot: A term referring to a shot that runs for the length of a scene and shows all of the characters in view.

Zoom: Zooming is a movement of the camera lens as opposed to a movement of the camera itself. Zooming in means adjusting the lens to frame in closer on the subject, while zooming out means the opposite: adjusting the lens to take in more of the scene.

Track: A tracking shot is another way to follow subjects. This type of shot involves moving the entire camera from one place to another, and often follows a moving

subject. Tracking can involve moving the camera with tracks or on a dolly, or it can be done handheld.

**Guiding Questions:**

1. How does the selection of shots contribute to the overall feel of the scene?
2. What kinds of shot (close up vs. wide shot) convey different meanings and emotions?

**Materials:**

a) Scratch app to create projects - Download
b) Adobe AIR (required for Scratch to run) - Download
c) Storyboard templates – Template one, Template two
d) A film or a video game students will use to storyboard their favourite scene

**6. In-class activities**

**Non-Computer Activity**

Part A: View a Scene from your favourite video game or film.

a. Can you find a moment where changing shot types (wide shot, medium shot, close-up) help emphasize a story point?
b. How does the staging and framing heighten the emotional impact of the scene?

Part B: Pick a scene or sequence from your favourite video game or film and create a storyboard for a brief scene (3-5 minutes in length)

a. Sketch the frame for each shot in the scene selected. Indicate if it is a wide shot, medium shot or close-up in the description.

**Computer Activity**

With Scratch, you can program your own interactive stories, games, and animations.

Storyboarding is a great introductory exercise to do with students before using Scratch to help plan and map out the story of your game or interactive story. This activity will guide you through that process.

First, choose some common guidelines for students to follow to create their interactive story or game. These might include a theme each interactive story or game should address (for instance, bullying, community, identity) and also where the story takes place. The following provides a framework for how you may choose to structure your story:

- Act 1 is the "beginning," Have students answer Where, When, Who, What and Why of the story.
- Act 2 is the "middle," where characters attempt to achieve goals and encounter a conflict.
- Act 3 is the "end," where the conflict is resolved and the protagonist's true character is revealed.

**7. Post-class activities**

After students have illustrated their storyboards, cut each cell out and rearrange the sequence of events. Ask students to line up the cells in different orders to consider how this changes the story before deciding on the final outcome. This process introduces the concept of experiential media and demonstrates the basics of story structure by revealing how the story can change depending on how events are sequenced.

Once storyboards are complete each student has a map to start their project in Scratch.

**8. Evaluation and Assessment**

Learning outcome – student develops skills to:

a. Repeat similarities among objects, ideas, and problems.
b. Rely on quantitative or qualitative information (e.g., text, symbols, etc.) collected for analysis.
c. Find patterns among decomposed problems that can help us solve more complex problems.
d. Isolate key details while ignoring the remaining elements.
e. Get familiar with the process of breaking down complex problems into smaller, more manageable parts.
f. Create flowcharts, storyboards or other representational forms to prepare to write.
g. Identify patterns in writing by creating and refining written algorithms, pseudocode or flowcharts.

*Eftychia Xerou*
*Centre for Advancement of Research*
*and Development in Educational Technology Ltd.*

## 7.3. Scenario 3

**Author / Teacher:** Eftychia Xerou

**Subject:** IT Classes, Computer subjects, etc.

**Level** (ISCED, difficulty)**:** Secondary education - gymnasium

**Topic:** Create an interactive story to represent the social phenomenon of the Human Impact on the environment

**Prerequisite skills or knowledge** (connected to the prior lesson):

a. Block based programming (Scratch)
b. Knowledge regarding the human Impact on the environment and more specifically the destruction of the environment (a social phenomenon where the action of man destroys and contaminates different natural resources, such as land, water, air, minerals, and forests).

**Time required for a pre-class activity:** 2 hours
**Time required for an in-class activity:** 1 hour
**Time required for a post-class activity:** 2 hours

Story, canvas, challenges for student (optional, motivational)

As we all can understand the destruction of the natural environment that we live in is an extremely important social phenomenon that affects everyone directly during their life. The environmental degradation is the deterioration of the environment through depletion of resources such as air, water and soil; the destruction of ecosystems and the extinction of wildlife. During this lesson, students will have to present the environmental degradation and one of its many different aspects with the consequences to nature. Are we ready to understand how to conduct a research, learn more, understand the consequences and be proactive for the future?

1. **Student's new material (before the class)**
   a) Conduct a research regarding one specific problem: pollution, air pollution, smog, global warming, species extinction and decide in which area the interactive creation will be focused. Try to underline the representation of

the problem, the causes, the consequences and the solutions. Collect your findings in a word document.

b) Learn how to prepare a prototype storyboard for the scenario in an online program, to represent the 4 areas mentioned above in a story. Write the story and create your storyboard here: Free Storyboard Creator – Easily Make Storyboards Online | Canva. For help for the Canvas storyboard creator: Design Storyboard Activities | Canva for Education – YouTube

**2. In-class activities**

a) After breaking the complex problem into 4 different parts (causes, problem, consequences and solutions) and representing the problem in an interactive story through the creation of the prototype the next step is the creation of the story in Scratch. Scratch -Imagine, Program, Share (mit.edu) Tutorial of the software: How to Make a Story in Scratch | Tutorial – YouTube

b) The interactive story must have at least two main characters (sprites), 4 different backgrounds, sounds and words for each character and the interactive part of the story must represent a choice. The choice may be a choice regarding different consequences (for example if the main character chooses to walk in the city, the number of the cars may decrease and the air pollution level may decrease). This part must be represented as one choice for students and the other choice may be the opposite.

**3. Post-class activities**

a) Verify that the interactive story is working by choosing to start and test both of the choices given.

b) Write your conclusions on the possibility of changing small everyday habits to impact the environment (100 words).

**4. Evaluation and Assessment**

Learning outcomes:

1. Students learnt how to conduct a research
2. Students learnt about the destruction of the natural environment
3. Students represent stories on storyboards and in interactive representations

References: https://archive.unescwa.org/environmental-degradation#:~:text=Environmental%20degradation%20is%20the%20deterioration,and%20the%20extinction%20of%20wildlife.

## Bibliography


7 examples of algorithms in everyday life for students. (2023, February 13). Learning.com. https://www.learning.com/blog/7-examples-of-algorithms-in-everyday-life-for-students/

Ackermann, E., (2001). Piaget's constructivism, Papert's constructionism: What's the difference. *Future of learning group publication, 5*(3), p.438.

Adler, R. F., Hibdon, J., Kim, H., Mayle, S., Pines, B., & Srinivas, S. (2023). Assessing computational thinking across a STEM curriculum for pre-service teachers. *Education and Information Technologies, 28*(7), 8051–8073. https://doi.org/10.1007/s10639-022-11508-4

Ahamed, S.I., Brylow, D., Ge, R., Madiraju, P., Merrill, S.J., Struble, C.A. and Early, J.P. (2010). Computational thinking for the sciences: a three day workshop for high school science teachers. *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 42-46). ACM.

Aho, A. V. (2012). Computation and computational thinking. *Computer Journal, 55*(7), 832–835. https://doi.org/10.1093/comjnl/bxs074

Armoni, M. (2013). On teaching abstraction in CS to novices. *Journal of Computers in Mathematics and Science Teaching, 32*(3), 265-284.

Babazadeh, M., & Negrini, L. (2022). How Is Computational Thinking Assessed in European K-12 Education? A Systematic Review. *International Journal of Computer Science Education in Schools, 5*(4). ERIC. https://search.ebscohost.com/login.aspx?direct=true&db=eric&AN=EJ1366667&lang=pl&site=ehost-live

Banilower, E. R., Smith, P. S., Weiss, I., Malzahn, K. A., Campbell, K. M., & Weis, A. M. (2013). *Report of the 2012 national survey of science and mathematics education*. Chapel Hill: Horizon Research, Inc

Barendsen, E., Mannila, L., Demo, B., Grgurina, N., Izu, C., Mirolo, C., ... & Stupurienė, G. (2015). Concepts in K-9 computer science education. *Proceedings of the 2015 ITiCSE on working group reports* (pp. 85-116).

Belmar, H. (2022). Review on the teaching of programming and computational thinking in the world. *Frontiers in Computer Science, 4*. https://www.frontiersin.org/articles/10.3389/fcomp.2022.997222

Bocconi S., Chioccariello A., Dettori G., Ferrari A., Engelhardt K. (2016) Developing computational Thinking in Compulsory Education. Implication for policy and practice. *JRC Working Papers* JRC104188, Joint Research Centre

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing computational thinking in compulsory education: Implications for policy and practice. Resource document* (EUR 28295 EN). https://doi.org/10.2791/792158.

Burgett, T., Folk, R., Fulton, J., Peel, A., Pontelli, E. and Szczepanski, V. (2015). DISSECT: Analysis of pedagogical techniques to integrate computational thinking into K-12 curricula. *Frontiers in Education Conference (FIE)*, 2015. 32614 2015. IEEE (pp. 1-9). IEEE.

Calderon, A., Crick, T., & Tryfona, C. (2015). *Developing computational thinking through pattern recognition in early years education (Version 1).* Cardiff Metropolitan University. https://hdl.handle.net/10779/cardiffmet.21262746.v1 ([])

Cetin, I., & Dubinsky, E. (2017). Reflective abstraction in computational thinking. *The Journal of Mathematical Behavior, 47*, 70-80.

Chiprianov, V. and Gallon, L. (2016), July. Introducing Computational Thinking to K-5 in a French Context. *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 112-117). ACM.

Costa, E. J. F., Campos, L. M. R. S., & Guerrero, D. D. S. (2017). Computational thinking in mathematics education: A joint approach to encourage problem-solving ability. *2017 IEEE Frontiers in Education Conference (FIE)* (pp. 1–8). IEEE.

Csizmadia A., Curzon P., Dorling M., Humphreys S. (2015). *Computational thinking. A guide for teachers*. Computing At School (CAS).

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). Computational thinking: a guide for teachers.

Csizmadia, A., Standl, B., & Waite, J. (2019). Integrating the Constructionist Learning Theory with Computational Thinking Classroom Activities. Informatics in Education, 18(1), 41–67. ERIC.

Cui, Z., & Ng, O. L. (2021). The interplay between mathematical and computational thinking in primary school students' mathematical problem-solving within a programming environment. *Journal of Educational Computing Research, 59*(5), 988–1012.

Department of Education, UK. (2023). *Early Years Foundation Stage Profile 2024 handbook*. https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/1109972/Early_Years_Foundation_Stage_profile_2023_handbook.pdf

Dogan, A. (2020). Algorithmic Thinking in Primary Education. International *Journal of Progressive Education, 16*(4), 286-301.

Dominici P. (2016) *Il grande equivoco. Ripensare l'educazione (#digitale) per la Società Ipercomplessa.* Nòva.

Dorodchi, M., Dehbozorgi, N., Fallahian, M., & Pouriyeh, S. (2021). Teaching Software Engineering using Abstraction through Modeling. *Informatics in Education, 20*(4).

Evripidou, S., Amanatiadis, A., Christodoulou, K., & Chatzichristofis, S. A. (2021). Introducing algorithmic thinking and sequencing using tangible robots. *IEEE Transactions on Learning Technologies, 14*(1), 93-105.

Examples of Abstraction in Everyday Life. (2022). Learning.com. https://www.learning.com/blog/examples-of-abstraction-in-everyday-life/

Ezeamuzie, N. O., Leung, J. S. C., Garcia, R. C. C., & Ting, F. S. T. (2022). Discovering computational thinking in everyday problem solving: A multiple case study of route planning. *Journal of Computer Assisted Learning, 38*(6), 1779–1796. Academic Search Ultimate.

Flórez, F. B., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a Generation's Way of Thinking: Teaching Computational Thinking Through Programming. *Review of Educational Research, 87*(4), 834–860.

Futschek, G. (2006). Algorithmic thinking: the key for understanding computer science. *International conference on informatics in secondary schools-evolution and perspectives* (pp. 159-168). Springer, Berlin, Heidelberg.

Futschek, G., & Moschitz, J. (2010). Developing algorithmic thinking by inventing and playing algorithms. *Proceedings of the 2010 constructionist approaches to creative learning, thinking and education: Lessons for the 21st century (constructionism 2010)*, 1-10.

Gabbari M., Gagliardi R., Gaetano A., Sacchi D. (2020). Integrare Coding e Pensiero Computazionale nella didattica. *Bricks Magazine* http://www.rivistabricks.it/2020/03/03/integrare-coding-e-pensiero-computazionale-nella-didattica/

Ghani, A., Griffiths, D., Salha, S., Affouneh, S., Khalili, F., Khlaif, Z. N., & Burgos, D. (2022). Developing Teaching Practice in Computational Thinking in Palestine. *Frontiers in Psychology, 13*. https://www.frontiersin.org/articles/10.3389/fpsyg.2022.870090

Giacalone S. (2020). Che cos'è il pensiero computazionale? Training Course on Computational Thinking, Problem solving e Coding. *Cessaniti, 16/20.*

Grover, S., & Pea, R. (2013). Computational thinking in K–12: a review of the state of the field. *Educational Researcher, 42*(1), 38–43.

Grover, S., & Pea, R. (2018). Computational Thinking: A Competency Whose Time Has Come. In S. Sentance, E. Barendsen, & C. Schulte (Eds.), *Computer Science*

*Education: Perspectives on Teaching and Learning in School* (pp. 19–38). Bloomsbury Academic. https://doi.org/10.5040/9781350057142.ch-003

Haberman, B., & Muller, O. (2008). Teaching abstraction to novices: Pattern-based and ADT-based problem-solving processes. *2008 38th Annual Frontiers in Education Conference* (pp. F1C-7). IEEE.

Haddad, R.J. and Kalaani, Y. (2015). Can computational thinking predict academic performance?. *Integrated STEM Education Conference (ISEC), 2015 IEEE* (pp. 225-229).

Hazzan, O. (2008). Reflections on teaching abstraction and other soft ideas. *ACM SIGCSE Bulletin, 40*(2), 40-43.

Howard, W.R. (2007), Pattern Recognition and Machine Learning, *Kybernetes, Vol. 36* No. 2, pp. 275-275. https://doi.org/10.1108/03684920710743466)

Howell, L., Jamba, L., Kimball, A.S. and Sanchez-Ruiz, A. (2011, March). Computational thinking: modeling applied to the teaching and learning of English. *Proceedings of the 49th Annual Southeast Regional Conference* (pp. 48-53). ACM.

Hromkovic, J., Kohn, T., Komm, D., & Serafini, G. (2017). Algorithmic thinking from the start. *Bulletin of EATCS, 1*(121).

Hsieh, M.-C., Pan, H.-C., Hsieh, S.-W., Hsu, M.-J., & Chou, S.-W. (2022). Teaching the Concept of Computational Thinking: A STEM-Based Program With Tangible Robots on Project-Based Learning Courses. *Frontiers in Psychology, 12*. https://www.frontiersin.org/articles/10.3389/fpsyg.2021.828568

Hsu, T. C., Chang, S. C., & Hung, Y. T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers and Education, 126*, 296-310. https://doi.org/10.1016/j.compedu.2018.07.004

https://archive.unescwa.org/environmental-degradation#:~:text=Environmental%20degradation%20is%20the%20deterioration,and%20the%20extinction%20of%20wildlife.

Isbell, C.L., Stein, L.A., Cutler, R., Forbes, J., Fraser, L., Impagliazzo, J., Proulx, V., Russ, S., Thomas, R. and Xu, Y. (2010). (Re) defining computing curricula by (re) defining computing. *ACM SIGCSE Bulletin, 41*(4), pp.195-207.

Kiss, G., & Arki, Z. (2017). The influence of game-based programming education on the algorithmic thinking. *Procedia-Social and Behavioral Sciences, 237*, 613-617.

Kong, S.-C., Chiu, M. M., & Lai, M. (2018). A study of primary school students' interest, collaboration attitude, and programming empowerment in computational thinking education. *Computers & Education, 127*, 178-189. doi: 10.1016/j.compedu.2018.08.026

Koppelman, H., & Van Dijk, B. (2010). Teaching abstraction in introductory courses. In Proceedings of the fifteenth annual conference on Innovation and technology in computer science education (pp. 174-178).

Kramer, J. (2007). Is abstraction the key to computing?. *Communications of the ACM, 50*(4), 36-42.

Krist, C., Elby, A., Good, J., Gupta, A., Sohr, E. R., & Yadav, A. (2017). Integrating computational thinking strategies that support science inquiry: A case study from a summer PD. *Paper presented at the Annual Meeting of American Educational Research Association*, San Antonio, TX

Larsen, M. (2010). How to recognize word patterns in a foreign language: Autolingual – learn a foreign language by yourself. *AutoLingua*l. https://autolingual.com/word-pattern-recognition/

Lee, J., Joswick, C., & Pole, K. (2023). Classroom Play and Activities to Support Computational Thinking Development in Early Childhood. *Early Childhood Education Journal, 51*(3), 457–468. Academic Search Ultimate.

Lockwood, James & Mooney, Aidan. (2017). Computational Thinking in Education: Where does it Fit? A systematic literary review. *International Journal of Computer Science Education in Schools.2*.

Lv, L., Zhong, B. & Liu, X. (2022). A literature review on the empirical studies of the integration of mathematics and computational thinking. *Educ Inf Technol*. https://doi.org/10.1007/s10639-022-11518-2

Malik, S. I., Mathew, R., Tawafak, R. M., & Khan, I. (2019). Gender difference in perceiving algorithmic thinking in an introductory programming course. *EDULEARN19 Proceedings of the International Conference on Education and New Learning Technologies (Vol. 18*, pp. 8246-8254).

Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., & Settle, A. (2014). Computational Thinking in K-9 Education. *ITiCSE '14 Proceedings of the 2014 conference on Innovation & technology in computer science education* (pp. 1-29). ACM Press Digital Library. https://doi.org/10.1145/2713609.2713610.

McCormick, K. & Hall, J. (2022). Computational thinking learning experiences, outcomes, and research in preschool settings: a scoping review of literature. *Education and Information Technologies. 27*. 1-36. 10.1007/s10639-021-10765-z.

Mezak, J., & Papak, P. P. (2018, May). Learning scenarios and encouraging algorithmic thinking. *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (pp. 0760-0765). IEEE.

Midoro V. (2015). *La Scuola ai tempi del Digitale – Istruzioni per costruire una scuola nuova*. FrancoAngeli.

Milková, E. (2012). Development of algorithmic thinking and imagination: base of programming skills. *Proceedings of the 16th WSEAS International Conference on Computers*.

Mooney, A., Duffin, J., Naughton, T., Monahan, R., Power, J. and Maguire, P. (2014). *PACT: An initiative to introduce computational thinking to second-level education in Ireland*. Conference: International Conference on Engaging Pedagogy.

Munn, C. (2021). A Qualitative Study Exploring Robots as a Potential Classroom Tool for Teaching Computational Thinking within a Sixth-Grade Class. *Journal of Computers in Mathematics and Science Teaching, 40*(3), 229–264. ERIC.

Neumann, M. D., Dion, L., & Snapp, R. (2021). *Teaching Computational Thinking: An Integrative Approach for Middle and High School Learning.* The MIT Press. https://doi.org/10.7551/mitpress/11209.001.0001

Oktan, S., & Vural, S. (2021). A teaching strategies model experiment for computational design thinking. *TECHNE: Journal of Technology for Architecture & Environment*, 154–158. Academic Search Ultimate.

Parry, M. (2021). Abstraction: The important bits. Hello World. https://helloworld.raspberrypi.org/articles/hw16-abstraction-the-important-bits

Peel, A., & Friedrichsen, P. (2018). Algorithms, abstractions, and iterations: Teaching computational thinking using protein synthesis translation. *The American Biology Teacher, 80*(1), 21-28.

*Pensiero Computazionale - Una guida per insegnanti CNR* (2016). Computing At School. https://pensierocomputazionale.itd.cnr.it/pluginfile.php/957/mod_page/content/7/Guida%20al%20Pensiero%20Computazionale.pdf

Pérez-Marín, D., Hijón-Neira, R., Bacelo, A., & Pizarro, C. (2020). Can computational thinking be improved by using a methodology based on metaphors and scratch to teach computer programming to children? *Computers in Human Behavior, 105*, 105849. https://doi.org/10.1016/j.chb.2018.12.027

Rich, K. M., Spaepen, E., Strickland, C., & Moran, C. (2020). Synergies and differences in mathematical and computational thinking: implications for integrated instruction. *Interactive Learning Environments, 28*(3), 272–283.

Rich, K.M., Yadav, A. & Larimore, R.A. Teacher implementation profiles for integrating computational thinking into elementary mathematics and science instruction. *Educ Inf Technol 25*, 3161–3188 (2020). https://doi.org/10.1007/s10639-020-10115-5

Samarasinghe, S. (2006). *Neural Networks for Applied Sciences and Engineering: From Fundamentals to Complex Pattern Recognition* (1st ed.). Auerbach Publications. https://doi.org/10.1201/9780849333750

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: a theoretical framework. *Education and Information Technologies, 18*(2), 351–380. https://doi.org/10.1007/s10639-012-9240-x

Shen, X., Efros, A. A., & Aubry, M. (2019). Discovering visual patterns in art collections with spatially-consistent feature learning. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 9278-9287).

Shuiyan He, Yongmin Hang, & Yi Ding. (2014). Teaching method based on computational thinking a case research. *2014 9th International Conference on Computer Science & Education*, 817–820. https://doi.org/10.1109/ICCSE.2014.6926576

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review, 22*, 142–158.

Sigayret, K., Tricot, A., & Blanc, N. (2022). Unplugged or plugged-in programming learning: A comparative experimental study. *Computers & Education, 184*, 104505.

Sung, W., & Black, J. B. (2020). Factors to consider when designing effective learning: infusing computational thinking in mathematics to support thinking-doing. *Journal of Research on Technology in Education, 53*(4), 404–426.

*Symposium on Computer Science Education (SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, 906–912. https://doi.org/10.1145/3287324.3287431

Tekdal, M. (2021). Trends and development in research on computational thinking. *Education and Information Technologies, 26*(5), 6499–6529. https://doi.org/10.1007/s10639-021-10617-w

The Bowers Institute. (n.d.). *TECH TIP: Computational Thinking*. https://www.thetech.org/media/l0nasrfv/techtip_computationalthinking.pdf

Thorson, K. (2018). *Early Learning Strategies for Developing Computational Thinking Skills. Getting Smart.* https://www.gettingsmart.com/2018/03/18/early-learning-strategies-for-developing-computational-thinking-skills/

Tsalapatas, H., Heidmann, O., Alimisi, R., & Houstis, E. (2012). Game-based programming towards developing algorithmic thinking skills in primary education. *Scientific Bulletin of the Petru Maior University of Targu Mures, 9*(1), 56-63.

Ung, L.-L., Labadin, J., & Mohamad, F. S. (2022). Computational thinking for teachers: Development of a localised E-learning system. *Computers & Education, 177*, 104379. https://doi.org/10.1016/j.compedu.2021.104379

Uzumcu, O., Bay, E. The effect of computational thinking skill program design developed according to interest driven creator theory on prospective teachers. *Educ Inf Technol 26*, 565–583 (2021). https://doi.org/10.1007/s10639-020-10268-3

van Zanten, M., & van den Heuvel-Panhuizen, M. (2018). Opportunity to learn problem solving in dutch primary school mathematics textbooks. *ZDM: The International Journal on Mathematics Education, 50*(5), 827–838.

Waite, J. L., Curzon, P., Marsh, W., Sentance, S., & Hadwen-Bennett, A. (2018). Abstraction in action: K-5 teachers' uses of levels of abstraction, particularly the design level, in teaching programming. *International Journal of Computer Science Education in Schools, 2*(1), 14-40.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*(1), 127–147

White, P., & Mitchelmore, M. C. (2010). Teaching for abstraction: A model. *Mathematical Thinking and Learning, 12*(3), 205-226.

Whitney-Smith, R. M. (2023). The Emergence of Computational Thinking in National Mathematics Curricula: An Australian Example. *Journal of Pedagogical Research, 7*(2), 41–55. ERIC.

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM, 49*(3), 33–35. Business Source Ultimate.

Wing, J. M. (2011). Research Notebook: Computational Thinking—What and Why. *The link Magazine, 6*, 20-23. https://people.cs.vt.edu/~kafura/CS6604/Papers/CT-What-And-Why.pdf Wing, J. M. (2006). Computational Thinking, CACM 49(3), pp. 33-35.

Wing J. M. (2016). *Computational thinking, 10 years later*. https://www.microsoft.com/en-us/research/blog/computational-thinking-10-years-later/

Wray S. (2012). *Not a Tool, but a Philosophy of Knowledge*. http://www.stuartwray.net/philosophy-of-knowledge.pdf

Yadav, A., & Berthelsen, U. (2021). *Computational Thinking in Education: A Pedagogical Perspective*. Routledge. https://doi.org/10.4324/9781003102991

Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM, 60*, 55–62. https://doi.org/10.1145/2994591

Yihuan D., Catete V., Jocius R., Lytle N., Barnes T., Albert J., Joshi D., Robinson R., and Andrews A. (2019). PRADA: A Practical Model for Integrating Computational Thinking in K-12 Education. *Proceedings of the 50th ACM Technical.*

Computational thinking is the ability to understand, analyze and solve problems using IT tools and concepts.
The practical value of the book is undeniable, especially in the context of the present-day educational challenges. The ideas presented are not merely theoretical considerations, but are oriented towards practical application in the daily teacher's work. Specific examples and tips on the implementation of computational thinking in the teaching process provide the reader with specific tools to act effectively.
The book is also distinguished by the inclusion of an educational application – CTApp, which is a valuable addition to theoretical considerations.

**Wiesław Kowalski, PhD**

It is an up to date book covering all the last developments in the field. A key element is CTApp serious game which is an excellent way to learn about Computational Thinking with the latest trend in educational technology, serious games. Serious games can be incredible tools for teaching, learning, and education.

**Paraskevi Poulogiannopoulou, Phd, European University Cyprus**

**Innovatio Press Publishing House**
**WSEI Univeristy**
20-209 Lublin, Projektowa 4
tel.: +48 81 749 17 77, fax: +48 81 749 32 13
www.wsei.lublin.pl
e-mail: wydawnictwo@wsei.lublin.pl

Electronic ISBN: 978-83-67550-13-0

Co-funded by the Erasmus+ Programme of the European Union

This publication has been funded with the support from the European Commission (project no:2020-1-PL01-KA201-081924). This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.