frontiers | Frontiers in Computer Science

TYPE Systematic Review
PUBLISHED 19 October 2022
DOI 10.3389/fcomp.2022.997222

OPEN ACCESS

EDITED BY
Carmen Llorente Cejudo,
University Seville, Spain

REVIEWED BY
Cesar Collazos,
University of Cauca, Colombia
Bert Zwaneveld,
Open University of the
Netherlands, Netherlands

*CORRESPONDENCE
Héctor Belmar
hector.belmar@hotmail.com

SPECIALTY SECTION
This article was submitted to
Digital Education,
a section of the journal
Frontiers in Computer Science

RECEIVED 18 July 2022
ACCEPTED 26 September 2022
PUBLISHED 19 October 2022

CITATION
Belmar H (2022) Review on the
teaching of programming and
computational thinking in the world.
Front. Comput. Sci. 4:997222.
doi: 10.3389/fcomp.2022.997222

Review on the teaching of programming and computational thinking in the world

Héctor Belmar[1,2]*

[1]School of Computer Engineering, National Training Institute (INACAP), Santiago, Chile,
[2]Universidad Metropolitana de Ciencias de la Educación, Santiago, Chile

Recent studies suggest that computational thinking, composed of the skills of abstraction, decomposition, algorithmization, debugging, and problem-solving, is the fundamental skill for scientific, technological, and economic development for the twenty-first century. However, this diagnosis that is unveiled in rich countries remains nebulous for poor countries. The problem is that education in computational thinking is fundamental for countries to insert themselves in the international arena in an advantageous way and thus achieve the welfare goals for the population of each country. The objective of this research was to make a bibliographic review that shows the state of the art in the teaching of computer programming and computational thinking in the 5 continents. In the review, the advances in the countries of Europe, North America, Oceania, and Asia were observed, whereas in Latin America and Africa, the advances are still basic in some countries and non-existent in others. This review is based on Preferred Reporting Items for Systematic reviews and Meta-Analyses (PRISMA). The main search terms were "Computational thinking" and "Teaching computer programming." The search was performed in the ACM, Conference on Computational Thinking Education (Hong-Kong), Google Scholar, WOS, and SCOPUS databases, from October until December 2020, whose publication year was from 2016 onward. One of the main results found is that the teaching of computational thinking in England was implemented in schools in 2014; in Germany, it has been implemented since 2016 at a transversal level in universities; in South Korea, China, and Taiwan, it has been implemented since 2016. However, in Latin America and Africa governments, the subject is still not considered.

KEYWORDS

computational thinking, computer programming, algorithm, programming, computer science

Introduction

Contextualization

In the last decade, teaching of computational thinking skills in compulsory education is being developed in various countries around the world, which is a strategic decision

for technological development and for the acquisition of twenty-first century skills. In this rapidly advancing world, education professionals are subjected to new demands during the exercise of teaching, which has caused training institutions to permanently analyze their academic work, to make the necessary adjustments to their teacher training programs to respond to the new demands (Coppelli, 2018).

In this context, questions arise; Will it be necessary to include the teaching of computational thinking as a compulsory subject? Second, from which course should it be incorporated, how many hours per week would be necessary? Or would it be better to incorporate it as a development axis in existing subjects, such as technology education? Even the question arises whether it will be necessary to incorporate it in a transversal way, in all subjects? Then comes a second group of questions: Regarding teachers, are there teachers prepared to teach computational thinking, are there didactic strategies aimed at it, and also have evaluation instruments been developed to measure achievements at each educational level? All these questions and many others may arise when dealing with the issue of teaching computational thinking and the teaching of these skills in the educational system.

As for computational thinking, it encompasses a range of thinking skills specific to problem-solving, including abstraction, decomposition, debugging, pattern recognition, logic, and algorithm design, among other skills. But it is not only this because problem-solving skills also go beyond rote learning and procedural skills. Logical thinking involves analyzing situations to decide about an event. Algorithms are step-by-step procedures for solving problems, which are then codified in a programming language. It should be noted that computational programming is the natural scaffold for the acquisition of computational thinking skills, so its teaching from school is essential for future professionals to fully integrate it as mathematics, science, and language (Grover and Pea, 2018).

The implementation of computational thinking in school is already advancing in several countries, as in USA in December 2015 was signed the Federal Law entitled "Every Student Succeeds," which is responsible for public policies in this country. This law places computing on an equal footing with other academic disciplines, such as Mathematics, Geography, History, and Science (Brackmann et al., 2016). In January 2018, the Spanish Ministry of Education, Culture, and Sport published the report "Programming, robotics and computational thinking in the classroom," the document describes the current situation of programming, robotics, and computational thinking in the basic curriculum and different autonomous communities and several unofficial initiatives, led from companies, universities, or civil society. There are some questions to answer about it, such as integrating it, within the current disciplines or whether it is preferable for it to be an extracurricular activity (Adell-Segura et al., 2019).

Teaching of computer programming

Despite it is true that this review aims to study the teaching-learning of computer programming, it is also true that the concept of "Computational Thinking" has taken some prominence whenever the subject of computer programming is addressed. Nowadays, the world is incorporating the teaching of computer programming for elementary and high school students, to develop computational thinking skills, which go far beyond learning to program. Thus, the teaching of computational programming serves as a scaffold for the development of computational thinking skills, which are transferable to other areas of knowledge, and it means that they not only remain in computer science scholars, but also radiate into the whole area of science and technology (STEM), also including art (STEAM) and beyond (Rojas and García, 2020).

According to Wing (2006), computer science is the study of computation and asks himself, what can be computed and how to compute it? To have enough elements of judgment and to appropriate a definition, we will show several research that address the studies based on computational sciences, which orient their meanings based on their applicability. We can number some applications of computing such as internet of things, social networks, big data, artificial intelligence, robotics, video games, communications, smart phones, augmented reality, virtual reality, etc. The reality is that computer science is ubiquitous, so it is something to know, or at least a part of its applicability (Psycharis et al., 2020).

The contributions of computer science in education are very broad. Thus, universities around the world are revising their undergraduate computer science curricula, because of which they are changing their first course in computer science to cover fundamental concepts, not just programming. In addition, interest in computational thinking has grown beyond undergraduate education, with many focusing on incorporating computational thinking into education from kindergarten through K-12. As for sponsors, they include professional organizations, government, academia, and industry. Computer scientists know the value of thinking abstractly, thinking at multiple levels of abstraction, abstracting ideas to manage complexity, abstracting to scale, iteration, debugging, and software testing, among others (Wing, 2011).

For García (2018), we cannot abstract from the teaching of programming, but schools must take steps with our young people to operate in a virtual world, for which they must prepare in the language of this century, without which they will become digital illiterates. Therefore, the school should train the youth with the skills of computational thinking. So far, the energy has been directed at training users of existing computational tools.

Of course, this is insufficient, since what the present century demands is to acquire the skills of computational thinking, to live a new way of thinking and problem-solving. Therefore, instead of teaching students to be the users of a changing technology, they should be trained in the new paradigm of computational thinking, to be creators of new technologies (García, 2018).

According to Wing's (2006) definition, computational thinking has been supplemented as a generalized problem-solving approach applicable to a broad matrix of STEM and non-STEM fields. A formal definition is still an open topic of discussion in the literature, but in general, scholars agree that computational thinking skills include algorithmic thinking, navigating multiple levels of abstraction, decomposing problems into manageable pieces, and representing data. Computational thinking can be taught with or without the use of computers, but it is often operationalized through computer programming, as this makes the abstraction at the heart of computational thinking easier (Grover and Pea, 2018).

It should be noted that, with the research shown and speaking from experience, it could be safely said that computational thinking is the set of skills such as abstraction, decomposition, pattern recognition, algorithmization, debugging, and problem-solving, which could be extended to the skill of critical thinking. For the author, computational thinking is a new paradigm that expands the cognitive skills of students, in terms of making visible issues that until now are typical of computer technicians and professionals, but that undoubtedly will be a great contribution to the learning of science, mathematics, and many other areas of knowledge, where abstraction and the decomposition of problems, parallelism, and thinking at multiple levels provide the necessary contributions for students to learn from a different approach to the traditional one, especially modeling skills that allow the integration of all skills into a broad knowledge capable of integrating diverse cognitive skills.

General objective and research questions

The general objective was to make a bibliographic review that shows the state of the art in the teaching of computational programming and computational thinking in the 5 continents (see Figure 1).

It also suggests the possible lines of action that should be followed by the countries that are at the stage of evaluating the implementation of computational thinking.

Research questions:

What is computational thinking and how is it linked to computational programming?

Are there didactic strategies in the teaching of computational programming at school? And if so, what didactic strategies are used in the teaching of computer programming at school?

What is the international context in which the process of teaching computer programming is designed and implemented at different educational levels; primary, secondary, and tertiary?

Which are the pioneer countries in implementing the teaching of computer programming, what are they doing?

Which computational tools/programming languages are most used to provide the scaffolding toward the acquisition of computational thinking skills?

Method

Search terms and databases consulted

In the first instance, the trend of the investigated concepts (Computational Thinking and Computational Programming) was reviewed in Google trends (trends.google.es/trends), which showed a growing interest in investigating computational thinking from 2014 onward. This review is based on The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. International Journal of Surgery, 88, 105906 (Page et al., 2021).

The search terms were as follows:

- Primary search terms: "Computational thinking" and "Teaching computer programming."
- Secondary search terms: Teaching computational programming in elementary school, secondary school, tertiary/higher education, educational robotics, gamification, and didactics in computational programming.

A detailed search was conducted in the following databases since October until December 2020, whose year of publication was from 2016 onward, due to the accelerated obsolescence of the investigated subjects and associated technologies:

* ACM (Association for Computing Machinery) Digital Library.
* Google Scholar.
* WOS AND SCOPUS.
* Conference on Computational Thinking Education (Hong-Kong).

Selection of articles

The flow chart summarizing the searches in the various databases is below. It shows the articles eliminated and the reasons why (see Figure 2).

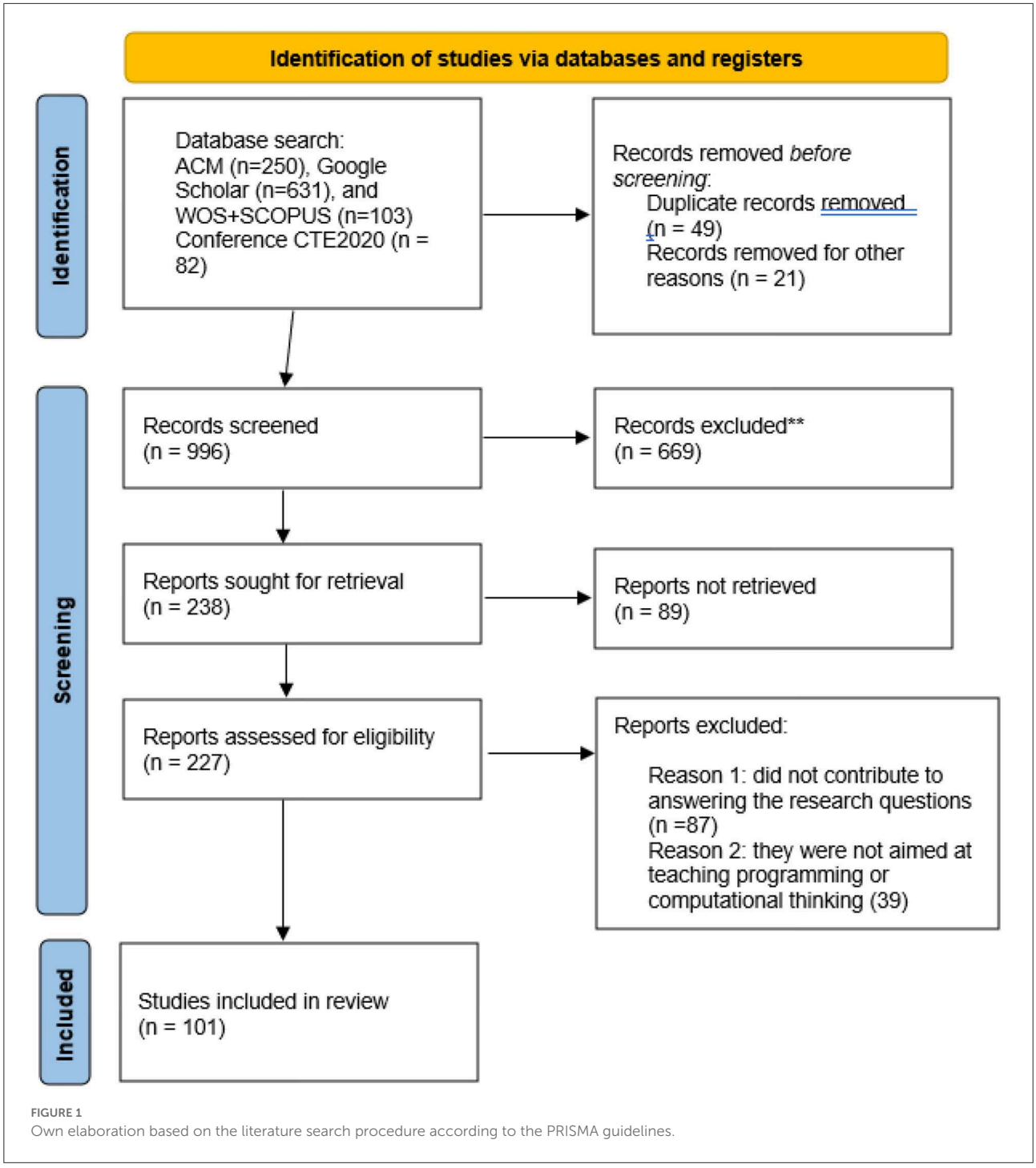

FIGURE 1
Own elaboration based on the literature search procedure according to the PRISMA guidelines.

In the ACM Digital Library, the use of the main terms together with all secondary search terms resulted in 251 articles. After eliminating by title, 60 publications were retained.

A Google Scholar search (combining all secondary search terms with computational thinking) yielded 631 articles. After filtering by title and eliminating unrelated articles, 96 publications were retained. In addition, 21 articles were found in WOS and SCOPUS.

In the publications of the Conference on Computational Thinking Education (Hong-Kong, 2020), which was held for the first time in 2017, many papers were found and an own categorization of computational thinking, which guide the

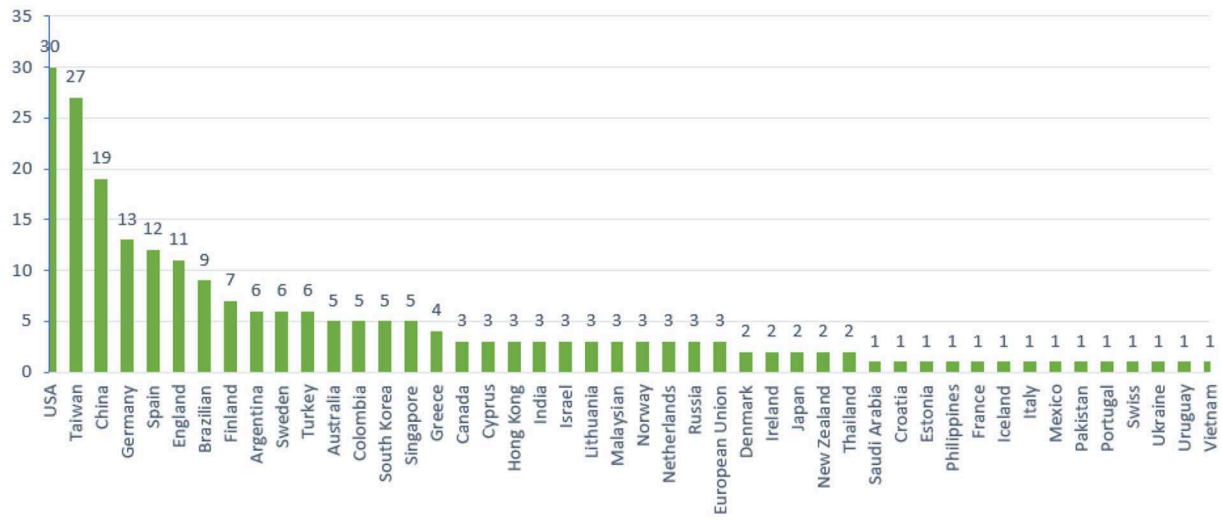

FIGURE 2
Number of paper per country.

search toward new horizons, from where in addition to the published papers were the references of each of them, so we proceeded to review most recent publications and those of greater interest, and thus, 82 publications were completed.

The inclusion and exclusion criteria were developed as recommended in the systematic review guidelines.

The texts considered have the following characteristics:

* Directly answered one or more research questions.
* Were related to the teaching of computer programming in educational institutions.

Studies were excluded if they:

* Were in a book format or gray literature (opinion articles, technical reports, blogs, presentations, etc.).
* Did not answer any research question.
* Other exclusion criteria used, was to ask the following questions:
- How well does the evaluation address its original objectives and purposes?
- How well was the data collection conducted?
- How clear and coherent is the report?

After these steps, which consisted of reading the abstracts and conclusions, and eliminating articles if they did not comply with the above criteria, 101 articles were selected for review. Regarding the year of publication of the articles reviewed, it is highlighted that 19% are from 2018, 40% correspond to the publications from 2019, 22% are the publications from 2020, and 5% are the publications from 2021.

Results: The main research

Contextualization

Undoubtedly, the teaching of computer programming lacks a didactic that guides it and that doses the contents in smaller units, disaggregating the complexity, to make learning more fluid for the students. In the computing field, it is like another side of teaching, which does not include didactic elements from other STEM areas, and runs on its own track, where those who teach do not have pedagogical training and teach using the same method as they learned, which generates a vicious circle contrary to didactics as it happens in mathematics, science, and other areas of learning. The teaching of computer programming is fundamental to acquire the skills of computational thinking, which opens new opportunities to learning with the skills of abstraction, problem disaggregation, algorithmization, parallel task processing, debugging, and pattern recognition, among others.

Thus, in this review, we will seek to answer the research questions, to guide what is being done in the field, or to define what is intended to be done in the various countries. It should be noted that not all the publications were written in English, so the translation from Chinese, Russian, Korean, and German, among others, was needed. However, the common denominator found was that the teaching of computational thinking is the key skill for the twenty-first century, that it should be taught from an early period of time, that its skills are transferable to other areas of knowledge, and that computer programming is the natural scaffolding to achieve this goal.

The research questions to which answers were sought were the following: What is computational thinking and how is it linked to programming? Are there didactic strategies in the teaching of computer programming at school? And if they exist, what didactic strategies are used in teaching computer programming at school? What is the international context in which the process of teaching computer programming is designed and implemented at different educational levels; primary, secondary, and tertiary? What are the pioneer countries in the implementation of teaching computer programming, what are they doing? What computational tools or programming languages are the most used to provide the scaffolding toward the acquisition of the skills of computational thinking?

By obtaining the results, several ways of facing the problem of the lack of didactic strategies in the teaching of computer programming were found. Some of them resort to the use of games, others to educational robotics, metaphors, etc., but none of them proposes a properly developed didactic, which is applicable to all contexts and levels of teaching, rather they are trials and errors. It stands out to the general rule, the 10 didactic principles of Brown and Wilson (2018), which create a kind of algorithm for teaching programming, which is perhaps the path that should be followed. In this sense, it is left to the reader to acquire the necessary elements of judgment, such as validating one or another attempt to shape didactics of computational programming, just as didactics of mathematics and science have.

Computational thinking—Research and concepts

Computational thinking is unquestionably the skill that young people of the twenty-first century must acquire, and computational programming seems the logical way to achieve it, since making a computer program confronts the student with problem-solving, abstraction, task sequencing or task algorithmization, parallelism, and code debugging. In 2018, Ching et al. published the research, in which he begins by positing that computer programming should be taught for children of all ages. Here, in the research conducted, computational thinking is considered as a broad problem-solving framework involving problem-solving skills, processes, and approaches, and "programming" as a key practice to support and cultivate the cognitive tasks involved in computational thinking (Ching et al., 2018).

The following research works computational thinking from a different perspective, and here, it seeks to teach the same computational thinking skills, but without the need for a computer laboratory. In 2017, Brackmann's et al. research set out to develop computational thinking skills through unplugged activities in elementary school, for which they conducted a quasi-experiment with 5th and 6th graders from two public elementary schools in Madrid, which has 73 students aged between 10 and 12 years. The unplugged approach is important for schools that do not have technological resources, internet connections, or even electricity. Regarding this, there is a lack of research showing the effectiveness of unplugged activities in the development of computational thinking skills, especially in elementary schools (Brackmann et al., 2017).

A key issue in the development of computational thinking is the measurement of such skills, and thus, some design and validation guidelines for computational thinking content have been developed (González, 2015). Developing computational thinking of young children with educational robotics allows an interaction effect between gender and scaffolding strategy (Angeli and Valanides, 2020). In the same sense, there are successful experiences of teaching programming and robotics in elementary and middle school education (Gómez et al., 2019). Developing students' computational thinking with a poly-disciplinary approach becomes fundamental for the student group, due to the diversity and multiple approaches from different professions (Klunnikova et al., 2020). In summary, programming expertise promotes greater STEM motivation among all the students and with a focus on first-year girls and boys (Master et al., 2017).

Other publications show a series of research that deepen the development of computational thinking and programming, of which stand out the methodological proposal based on metaphors to teach programming to children (Pérez et al., 2018), the development of programming skills in engineering education through problem-based game projects with Scratch (Topalli and Cagiltay, 2018), the introduction to computer science as a part of the general education curriculum for the whole university (Khenner, 2019), toward the use of computational models in learning physical computing (hardware) (Seow et al., 2020), the comparison of learning behaviors of third-year elementary students and integrate robots and computational thinking board game in Singapore and Taiwan (Liang and Hsu, 2020), and the alignment of the framework in STEM classrooms infused with computational thinking (Bain et al., 2020).

In the same vein as the previous paragraph, a study about changing the way a generation thinks by teaching computational thinking through programming is also shown that it is a giant challenge, no doubt (Buitrago et al., 2017). Other authors focus on education in computational thinking, the problems, and challenges it holds (Angeli and Giannakos, 2020). In a following study, there is an analysis of response theory to the element of sequencing algorithms and programming concepts (da Cruz Alves et al., 2020). And finally, research about computational thinking skills and their impact on the achievements of the Trends in International Mathematics and Science Study (TIMSS) test, a study that looks in depth at the scope of such measurement, which will set the tone in the educational development of countries (Alyahya and Alotaibi, 2019).

The results will be shown in the following two categories

- Lines of research in applied programming education.
    - * Gamification.
    - * Educational robotics.
- Pedagogical and didactic elements.
    - * Pedagogical and didactic practices.
    - * Didactic methods in Computer Programming.
        - ° Learning programming like learning a second language.
        - ° Metaphors and building blocks for teaching programming.
        - ° Ten principles for teaching programming, by Brown & Wilson.

Now regarding the countries that publish, these investigations are mainly found in the continents, Europe, Asia, and North America. With less participation are some Latin American countries, such as Colombia, Brazil, and Argentina. The following graph shows the number of articles reviewed by country:

Lines of research in programming education applied

The following are some of the research lines associated with this work, such as gamification and educational robotics, which provide guidance on the teaching of computer programming in these areas.

Gamification: Teaching using games. Teaching by games in the field of computational thinking covers a range of possibilities, ranging from the design, construction, and use of games to the production and marketing of video games. Thus, there are experiences of the use of digital games in storytelling, learning mathematics, history, etc. Children learn better history if they play Age of Empires, personal reading books are more attractive if they are developed in animated environments, and mathematics is fun if there is an interaction with numbers and they are contextualized in a real environment, etc. In this regard, de Paula et al. published in 2018 an article entitled "Playing Beowulf," featuring a game produced by two 14-year-olds, "Playing Beowulf," which is a collaboration with teenagers from the British library (de Paula et al., 2018).

In the United States, researchers published an article in 2017, which presents an empirical study that provides evidence that a mathematical educational game can help with superior learning opportunities, as well as being more engaging. The "Decimal Point" game is a single-player game and is based on an amusement park metaphor and is aimed at high school students. The game is called "Decimal Point: the fantastic and fabulous world of fractional fun." In the game, the student travels sequentially to different theme areas (Haunted House, Wild West, Space Adventure, Amusement Park), playing a variety of mini games within each theme area aimed at learning decimals. Student progress is tracked by tracking the park and students are visually cued to the next game they will play (McLaren et al., 2017).

In the same vein, researchers Zhou and Hsu from National Taiwan University published in 2020, a study that aimed to integrate the computational thinking board game with robots, so that students put computational thinking skill into practice when completing the board game tasks by controlling the action of robots. The participants were sixth-grade students in Singapore. In total, two students divided into a team collaborated with each other, competed with the other team composed of two other students. The research used Robots City board games to enable the students to cultivate the concept of computational thinking through game-based learning and use cell phone applications to control robot behavior with programmed logic for analysis (Zhou and Hsu, 2020).

Finally, there are other research such as, Fotaris et al. (2016), with conducting an empirical study of the application of gamification techniques in a computer programming class, where it uses a leaderboard as motivation for players. Daungcharone et al. (2019) present learning the C programming language based on the mobile games to improve students' learning. Also, Pellas and Vosinakis (2018) show the effect of simulation games in learning computer programming on the learning performance of high school students by assessing computational problem-solving strategies. There is a wide range of applications of computer games to motivate and enhance learning, not only in the areas of computer science, but also in different areas (Wing, 2011).

Educational robotics. An educational robotics paper presented at the "International Conference on Computational Thinking Education 2020" in Hong Kong, which explored the learning behaviors of sixth-grade students using educational robots in English oral interaction learning units, in which a smart phone application was provided to control the action of the robots and ask students to orally interact with their peers to put learning sentences into practice. Then, the foreign language interactive behaviors were recorded and observed during the collaborative learning task period. The participants were 18 English foreign language learners, aged between 11 and 12 years (Kong et al., 2020).

It should be noted that multimedia environment can reduce students' anxiety and provide a less stressful classroom environment. In addition, multimedia tools enable English teachers to help the students improve their English performance and reduce language anxiety (Huang and Hwang, 2013 in

Kong et al., 2020). When students begin to feel confident in a foreign language classroom, they will naturally start speaking. Ultimately, all foreign language teachers need to motivate learners, encourage them to speak, and allow them to make mistakes freely (Atas, 2015; in Kong et al., 2020). Experimental results show that, in the learning process, with the help of educational robots, students speak English as a very common action to give commands to the robot. To complete the goal of the class, it is possible to interact with students in spoken English (Kong et al., 2020).

In another research at the Norwegian University of Science and Technology, a paper was conducted that deals with children's collaboration and participation, and their attitudes in game programming and educational robotics. The goal of the work was to investigate how collaboration and engagement moderate children's attitudes about programming activities. For this purpose, a study was designed with 44 children aged between 8 and 17 years, who participated in a full-day computer programming activity. Their participation and collaboration during the activity was measured by recording their gaze and attitudes regarding their learning, enjoyment, teamwork, and intention using post-activity survey instruments. Behavior was found to moderate the relationship among intention to learn, attitude toward teamwork, enjoyment, and observed learning (Sharma et al., 2019).

Also, authors Lee and Low (2020) conducted the implementation of a computational thinking curriculum with robotic programming activities. In addition, Cheng et al. (2018) investigated on the essential applications of educational robot through a requirements analysis from the perspective of experts, researchers, and instructors. Regarding SRA (Sense, Reasoning, Action) programming, authors Fanchamps et al. (2020) investigated the influence of SRA programming on algorithmic thinking using robotics. They also investigated the effects of using cell phone software to control educational robots with third-grade elementary school students (Yi and Ting, 2020, in Kong et al., 2020). In addition, Paucar-Curasma et al. (2022) conducted a study with a group of students aged 6–13 years in Peru, which was developed during 4 weeks, and applied computational thinking assessments applying the concepts of sequences, cycles, parallelism, conditionals, operators, and data manipulation, which allowed an appropriation of computational thinking skills by the participants in an optimal way (Paucar-Curasma et al., 2022). Finally, researchers Souza et al. (2019) analyzed the effect of computational thinking in mathematics through educational robotics.

Pedagogical and didactic elements

From the articles and reports selected and reviewed, all agree that computational thinking is the essential skill for education and industry in the twenty-first century, as it provides complex cognitive skills such as abstraction, decomposition, problem-solving, and algorithmic thinking, skills that are not only necessary for the scientific and technological development of humanity, but also for the preparation for work and the challenges posed by the automation of work and the loss of jobs as a result of robotization and artificial intelligence. With the above in mind, the challenge for governments is to prepare future generations for this complex world ahead; however, countries that have implemented educational reforms have found themselves with a shortage of teachers trained in the areas of computer science (Bocconi et al., 2016).

Pedagogical and didactic practices

Japanese researchers conducted research on pedagogical transformation based on computational design and thinking. Now, that technology has become ubiquitous, and it is more appropriate to discuss transformative pedagogy where technology is no longer considered a tool, but part of who we are. We do not believe that there is a strong basis for claiming that the integration of technology was the game changer, for our students working on extensive and complex robot programming tasks; rather, it was the design of teaching and learning in practice that made the real difference. A key implication for heutagogical (self-determined learning) practice that follows is that technology must build on a solid understanding of key concepts in teaching and learning, not the other way around (Vallance and Towndrow, 2016).

In turn, Chinese researchers from Hainan University investigated on a model of blended learning and the cultivation of innovative talent, whose mechanism is based on computational thinking. Computational thinking includes computational thinking and integration with the social and natural environment, including evolutionary thinking in general computing environments, alternatively promoting and co-evolving problems. In the future, non-computer science professionals can use computer science means for innovation in various disciplines. They can also develop support for various disciplines of research and innovation of new media of digital technology. Computational thinking can effectively help non-computer science professionals to bridge the gap between learning and training with common computing tools for future professionals (Zhang et al., 2019).

Didactic methods in computer programming general. In 2018, the book "Content and Skills of Computer Science" was published in Germany by the university press of the University of Potsdam, which makes a tour on what should be learned in computer science by the students of higher education of all careers, which receives the contributions of different academic teams of the German university world, to contribute collaboratively to lay the foundations of the skills that professionals

of the twenty-first century should possess. On this, some of the papers included in the publication are highlighted (Bergner et al., 2018).

In the publication, it was found that the central topics were algorithms, computer programming, and data representation, but also elementary technical concepts of computers and the internet. Courses also appeared that touched on traditional topics, on social implications, privacy, and the role of computer science in society. Regarding programming environments, programming systems were used professionally in most of the courses. Among the recommendations, it is indicated that it would be better to design courses focused on the specialties in which they are taught, since this would enhance the students' learning and applicability in subsequent subjects to be taken, thus better achieving the objectives of metacognition.

Computational thinking develops naturally in higher education, as shown by the research seen, which for reasons of space, it is not possible to show them in this section. The research deals with diverse topics, such as augmented reality for STEM learning; a computational thinking curriculum framework for lower sixth with implications for teachers' knowledge; computer science as a core competency for teachers in other disciplines; trainee teachers' views on computational thinking—STEM vs. non-STEM teachers; technology-enhanced learning in higher education; motivations, engagement, and academic performance; a study between Finland, mainland China, Singapore, Taiwan, and South Korea, comparing teachers' perceptions and preparedness to teach programming skills (Wu et al., 2020).

Learning programming using learning a second language methods. The introductory course in computer programming is first and foremost a language course, since teaching a language is already a component of the introductory programming course. The issue has been the pedagogical approach to teaching the linguistic aspects of the course, a teaching structure long abandoned for natural languages modeled after a linguistics approach (based on rules of grammar instruction), rather than incorporating principles of second language acquisition. In the twenty-first century, there are virtually no natural language classrooms using the prescriptive linguistics approach, yet it remains the universal teaching model for learning a programming language (Portnoff, 2018).

The absence of grammatical rules for learning programming causes students has a conceptual pedagogical gap to bridge on their own, even though instructors expect to solve problems using logic mediated by a programming language, in that most struggle to express basic fluency. Even those talented enough to get their programs to run and function correctly still compose inadequately structured programs well into their first year. The specific difficulties in learning a programming language coincide with the difficulties of learning a second natural language. The complications stem largely from the small number of control structures that programming languages employ, even though they are adaptive and are semantically broad (Portnoff, 2018).

Metaphors and blocks for teaching programming. Pérez et al. (2018) proposed using metaphors such as recipe/program, pantry/memory, and boxes/variables. They also illustrate the possibility of applying these metaphors to any resource available to the teacher. In total, four step-by-step scripts are provided on how to use metaphors in class, with the opinions of 62 children (enrolled in fourth, fifth, and sixth grades of Spanish Primary Education, ages 9–11) and their teacher. This proposal has been validated with 62 Spanish children, who found the metaphors useful in more than 65% of the cases. The students were able to understand the metaphors (<30% of the students found the metaphors difficult) and <10% of the students did not want to use the metaphors. The teacher was also asked to evaluate and validate the methodology (Pérez et al., 2020). In this same line in 2021, a research by Jiménez Toledo et al. (2021) published a study entitled "Discovery Model Based on Analogies for Teaching Computer Programming" through which he applied a discovery model that allowed the extraction of patterns, textual and linguistic analysis, in addition to the use of analogies for teaching the fundamental ideas of computer programming, which allowed to achieve better learning in students (Jiménez Toledo et al., 2021).

In another case, 20 years ago, AgentSheets combined four possibilities to create an early form of block scheduling. After initially focusing on syntactic possibilities, AgentSheets in computer science education, new approaches have been experimented with to go beyond syntax to address semantic and pragmatic obstacles. Thus, three approaches are described: (1) contextualized explanations to support understanding, (2) conversational programming to proactively assist and predict the future, and (3) live palettes to make programming more unpredictable. The block scheduling community has been concerned about the syntactic possibilities of the scheduling environment. It is time to shift research agendas toward systematically exploring the semantic and pragmatic possibilities of block programming (Repenning, 2017).

Ten principles for teaching programming, by Brown and Wilson (2018). Neil Brown is a researcher at King's College London University in England, and Greg Wilson belongs to the computer training organization DataCamp in Toronto, Canada, which after arduous bibliometric research and the experience accumulated in the development of their academic functions, offer ten principles for learning to program.

Principle 1: Remember that there is no such thing as the programming knowledge gene.

Computer programming skills are not innate, but rather a learned skill that can be acquired and improved with practice.

Principle 2: Use the help of your peers.

One-on-one tutoring is perhaps the ideal form of teaching, a teacher's full attention can be focused on one student, and they can fully customize their teaching for that person and tailor individual feedback and corrections based on a two-way dialog with them.

Principle 3: Use live coding.

Instructors should create programs in front of their students. This is most effective for several reasons: (1) It allows instructors to better answer "what if?" questions. Live coding allows instructors to follow the interests of their and (2) it facilitates unintended knowledge transfer.

Principle 4: Encourage students make predictions.

When instructors use live coding, they usually run the program several times during its development to show what it does. The key to making demonstrations more effective is to have students predict the outcome of the demonstration before running it.

Principle 5: Use pair programming.

Pair programming is a software development practice in which two programmers share a computer. One individual (called the driver) writes, whereas the other (called the navigator) offers comments and suggestions. The two switch roles 2–3 times per hour.

Principle 6: Use solved examples with labeled objectives.

A good way to guide students in building programs is to use solved examples: step-by-step guides that show how to solve an existing problem.

Principle 7: Stay in one language.

A principle that applies in all areas of education is that transfer only comes with mastery of a programming language.

Principle 8: Use authentic tasks.

Learners find authentic tasks more engaging than abstract examples.

Principle 9: Remember that freshmen are not experts.

Freshmen are taking their first steps in learning programming, so initially confront them with small problems, broken down into chunks.

Principle 10: Do not just code.

Faced with the challenges of learning syntax, semantics, algorithms, and design, examples that seem small to instructors can easily overwhelm beginners.

Discussion and implementation of programming education

For this author, as a computer professional and educator, most of the conclusions are shared; however, with respect to the authors who state that programming could disappear, I consider that it is like saying that the use of the wheel could disappear because of the invention of airplanes. It is clear that advances in artificial intelligence and code generators such as CSS3 Generator, Colorzilla Gradients, Genexus, among others, have been and will continue to be a great contribution in software development, but computer programming is something fundamental in the formation of computational thinking skills, such as addition, subtraction, and multiplication, as it is for elementary school students. I agree that the focus should go beyond programming, since computer programming is a tool to move toward computational thinking, and incidentally allows future generations to have the ability to be creators of technology and not just users of it.

Undoubtedly, the implementation of the teaching of computational thinking in the various countries of the world demands a lot of resources, creativity, knowledge, and will of governments to generate policies that open the way for the training of today's students to become professionals that will make them competitive and capable enough to perform their jobs in the twenty-first century. In the area of the implementation of the teaching of computational thinking, this is where the playing field is most uneven worldwide. There are countries in which all schools are equipped with computers, such as Australia, England, Germany, Japan, and New Zealand, among others, and other countries in which there is a significant percentage of schools that do not have such infrastructure, as in Latin American countries, and other countries that have a percentage of schools that do not even have electricity, as in the case of India and Africa.

Main considerations

This review discusses the concept of computational thinking, a term that finds support in the skills of abstraction, decomposition, sequencing, algorithmization, debugging, and problem-solving skills. In the literature, computational thinking is mentioned as the skill of the twenty-first century, and computational programming as the natural scaffolding to move toward the incorporation of computational thinking as an active problem-solving skill. In addition, it is noted that these types of skills have already been incorporated in international tests such as Trends in International Mathematics and Science Study (TIMSS), which allows measuring the trend of countries in mathematics and science skills. In this sense, a research that talks about computational thinking skills and its impact on TIMSS achievement looks in depth at the scopes of such measurement, which will set the tone in the educational development of countries (Alyahya and Alotaibi, 2019).

For Grgurina (2021) in her work entitled "Getting the Picture: Modeling and Simulation in Secondary Computer Science Education," computational thinking includes the skill of modeling, a skill that is at a higher level of complexity than problem-solving. Thus, computational thinking includes formulating problems, so that it is feasible to solve them with a computer, organizing and analyzing data logically, representing data through models, and automating solutions through algorithmization, which includes abstractions and

parallelism. Thus, looking toward student training, these skills support and improve attitudes such as: confidence in dealing with complexity, persistence in working with difficult problems, tolerance of ambiguity, the ability to deal with open problems, and the ability to work with others to achieve a common goal and communicate it. Thus, for example, in two-schema modeling, students have to construct two schemas and combine them: a schema consisting of the situation to be modeled and the schema of the means (mathematical, computer, scientific, everyday life, etc.) that can be used in the construction of an understandable model representing the situation to be modeled.

Grgurina describes computational thinking in terms of its main concepts, such as: data collection, data analysis, data representation, problem decomposition, abstraction, algorithms and procedures, automation, simulation, and parallelization (CSTA Computational Thinking Task Force, 2011; in Grgurina, 2021). The definition of computational thinking is complemented by the Carnegie Mellon Center for Computational Thinking (CMCCT), which states that it consists of three fundamental aspects: abstraction, modeling, and algorithmic thinking (Carnegie Mellon Center for Computational Thinking, 2010; in Grgurina, 2021).

In the review made by Shute et al. (2017), entitled "demystifying computational thinking," he addresses the concept as the set of skills of decomposition, abstraction, algorithm design, debugging, iteration, and generalization, understood as necessary skills for problem-solving. Regarding computational thinking and programming, they are analyzed at the same level; however, it does not consider that computational programming through teaching allows acquiring computational thinking skills, since it is necessary to abstract and decompose a problem before coding a program, and also, depending on the complexity, it will be necessary to abstract at several levels, and once programmed, debugging and checking that what is done does what it should solve will be necessary, thus incorporating computational thinking step by step. Regarding the bibliography consulted, most of it is observed between 2011 and 2015, so what is said there has been changing, leaving some conclusions, rather obsolete.

It is highlighted in Shute Shute et al. (2017), a research developed, in which a scale was created to measure computational thinking (Román-González et al., 2017; in Shute et al., 2017), which includes a 28-item scale and takes about 45 min to complete. It focuses on programming concepts, such as directions and sequences, loops, conditionals, and simple functions. In addition, there is the taxonomy created that categorizes the different levels showing how lesson plans can be designed. The proposed taxonomy proposes the following main categories, from which subcategories are derived, to guide future computational thinking assessment designs:

(a) Data practice; which is composed of data collection, data creation, data manipulation, data analysis, and data visualization,
(b) Modeling and simulation, which is composed of the subcategories; conceptual understanding, testing solutions, model evaluation, model design, and model building,
(c) Computational problem-solving, which is composed of solution preparation, programming, tool selection, solution evaluation, solution development, abstraction, and debugging,
(d) Systems thinking, which is composed of systematic investigation, understanding the relationships between components, multilevel thinking, communication, and system management.

In the author's opinion, this seems to be a significant contribution that goes in the right direction, since it allows addressing in an orderly manner each of the categories necessary to learn in computer science, when it is necessary to acquire the skills of computational thinking.

Moreno-León et al. (2018) worked on a review on computational thinking as a universal skill, in which they conclude several things to highlight:

- That the focus should not be placed exclusively on programming, but on the skills that are developed through learning to code, specifically computational thinking skills,
- The most effective way to train these computational thinking skills is through programming,
- It is possible that in the future new and more efficient mechanisms for developing these skills in students will flourish.
- There are even authors who argue that programming could disappear in a few years due to the advances in artificial intelligence.
- Consequently, we argue that the metaphor for presenting this movement to the educational community should shift toward computational thinking as a universal skill that can foster the learning of subjects and skills across the elementary and middle school curriculum.

When the most advanced countries in this area are investigated, European countries appear in the first line, or rather the European Union, a conglomerate of countries that advance at an even pace in various areas and especially in technology and implementation of educational systems that allow them to be at the forefront in future generations. Thus, the case of Ireland stands out, just to cite one example, the case of England, which have been advancing very fast in the implementation of the teaching of computational thinking in school. However, progress has also been observed in Latin American countries, such is the case of a study conducted by researchers at the Universidad del Cauca Colombia, where

Cruz et al. (2013) and their team published a study entitled "ChildProgramming Process: A Software Development Model for Kids," which was aimed at training children at an early age (8–10 years old) with the concepts and skills of software creation (agile methodologies), which allowed fostering teamwork and introducing computational thinking skills.

In 2017, researchers Lockwood and Mooney from Maynooth University, Ireland, asked a fundamental question about computational thinking: computational thinking in education, where does it fit? It should be noted, that in Ireland, computer science is not yet an assessed subject at state level, such as mathematics, language, or science. Although steps have been taken to include it, so far, all that is available to the students in the curriculum is a short course in programming. While programming is a very useful skill and one that can be beneficial to students in a wide variety of careers, it is not the only part of computer science that is of interest to consider. Although scholars have not been able to agree on a universal definition, according to Wing (2006), she gives two visions that define computational thinking, which are (a) computational thinking will be fundamental to new discoveries in all fields of endeavor, and (b) it will be an integral part of early childhood education (Lockwood and Mooney, 2017).

Furthermore, it should be noted to say the authors, who thinking computationally, is of enormous benefit to all disciplines. In Ireland, researchers in the Department of Computer Science at Maynooth University, designed the PACT program (PACT is an acronym for Programming Algorithms = Computational Thinking). The goal was to introduce computer science to Irish high school students and teachers through programming and algorithms, with the idea of improving computational thinking skills in participating students. Now in its fourth year, the PACT program has been delivered in over 60 schools and to over 1,000 students. With the introduction of programming into the curriculum and the call from administrators and governments to include more computer science content in schools (Lockwood and Mooney, 2017).

In 2016, the European Commission, specifically the Joint Research Center (JRC) for policy reporting, published the paper "Developing computational thinking in compulsory education—Implications for policy and practice," which seeks to shape the implementation of computational thinking in compulsory education. The document points out that, in the recent years, computational thinking, which includes the concepts of coding, programming, algorithmic thinking, among others, has been promoted as skills that are fundamental to everyone like numeracy and literacy. At this point, the questions arise: how can we define computational thinking as a key twenty-first century skill for schoolchildren; what are the central features of computational thinking and its relationship with programming in compulsory education; how can teachers be trained to effectively integrate computational thinking into their teaching practice? (Bocconi et al., 2016).

With respect to the integration of computational thinking in compulsory education, four important areas emerge for policy makers and stakeholders to focus on: consolidated understanding of computational thinking, comprehensive integration, systemic deployment, and policy support. A resurgence in the integration of computational thinking and, more broadly, computer science into compulsory education is evident, as indicated by the recent wave of curriculum reforms. In total, eleven countries in Europe have recently completed a reform process that includes computational thinking and related concepts. Another seven are currently planning to introduce computational thinking in compulsory education. In addition, seven other countries are integrating computational thinking based on their long tradition in computer science education, mainly in upper secondary schools. Some of them are expanding computer science education to include primary and lower secondary levels (Bocconi et al., 2016).

In 2013, the document "Computing programmes of study: key stages 1 and 2 National curriculum in England" was published in England by the Department for Education for its curriculum. The document states that a high-quality computational thinking education prepares students to use computational thinking and creativity to understand and change the world. Computer science has deep links with mathematics, science, design, and technology and provides information about natural and artificial systems. The core of computer science is computer science, in which students are taught the principles of information management and computation, how digital systems work, and how to use this knowledge through programming. Based on this knowledge, students are equipped to use information technology to create programs and a variety of applications (Radin and Hawley, 2013).

In 2017, the education system in England, according to evidence, shows that computer science education in the UK is fragmented and fragile. Its future development and sustainability depends on swift and coordinated action by governments, industry, and non-profit organizations. Neglecting opportunities to act could damage both the education of future generations and economic prosperity as a nation. The broad subject of computer science, which covers the three vital areas of computing, has become compulsory in schools in England for ages 5–16. Students aged 5–14 have computer science lessons 1 h per week, and some schools take the opportunity to teach computer science within other subjects. However, most teachers are teaching an unfamiliar school subject without adequate support. Moreover, they may be the only teacher in their school with this task. Governments need to address a growing shortage of computer science teachers (Calderon et al., 2017).

On the other hand, Asia has also been given the importance that the implementation of teaching computational thinking in

school should have. An example of this is the progress made in South Korea, China, and Japan, among others. The case of Japan, which, although it is a technologized country with 99% robotized factories in the 1980s, encounters the same problems as all countries in terms of teaching computer programming in schools: the shortage of teachers to teach this subject, in addition to the lack of training in science, In other words, there are few teachers and these few are not always well-trained to teach programming, which generates the second floor for students to achieve the skills of computational thinking. In Japan, the teaching of computer programming in schools was implemented in April 2020, and the curriculum requires them to teach it in a playful way, which further complicates Japanese teachers due to the lack of research in the field and the absence of didactics in the teaching of programming (Gougeon and Cross, 2021).

Meanwhile, in South Korea, although computational thinking education is still in its early stages of nationwide implementation, there has been some research evidence supporting the effectiveness of learning programming skills and related computational thinking skills. Computational thinking in K-12 contexts, education has been conducted in the areas of (a) innovating specific pedagogical approaches to computational thinking, (b) developing assessment tools to measure students' computational thinking knowledge, skills, and attitudes, (c) expanding coding education in physical computing and maker education, and (d) training teachers in computational thinking skills. Large-scale research revealed that students, teachers, and parents have positive perceptions about the needs and effectiveness of software education in schools (So et al., 2020).

A very remarkable case is India, a country with many inhabitants, with multiple deficiencies in infrastructure, and education, in addition to a linguistic diversity, within all these difficulties has been proposed to implement computational thinking, not only with the idea of leaving and improve the economic conditions of its inhabitants, but as a tool based on the skills that will allow it to develop in the XXI century with the advantage that its population is knowledgeable about new technologies, and thus provide more job opportunities anywhere in the world.

The National Computer Science Policy for school education in India advocates the development of a curriculum model that would include knowledge enhancement and generic skill development, focusing on digital literacy. The computer science education that has been introduced in urban sectors in India focuses mainly on digital literacy and some computer programming. The implementation of computer science in the curriculum has not been easy and has had to go through several challenges. According to the government reports, India has more than 1.6 million schools offering K-12 education to 300 million students. The problem is compounded by the fact that education in the country is administered by two national boards of education, with each of India's 29 states having its own board of education! While the common language of instruction is English in urban areas, 70% of the population resides in rural areas where education is conducted in their own local language (Shah, 2019).

To prepare the students to creatively engage in the digital age, CSpathshala proposes an activity-based disconnected computational thinking curriculum for primary and secondary schools. The computational thinking curriculum is highlighted: teaching computation without computers, community-created teaching materials: 200 lessons, no cost to schools, subset of teaching materials translated into Gujarati, Hindi, and Marathi (dialects in India), 30,000 schools in Tamil Nadu learn computational thinking as part of the math curriculum. This includes 425 residential welfare schools, conducting 90 awareness workshops and training programs, at no cost to the schools, 5,400 volunteer participants from 2,650 institutions. With this plan, a light is seen in learning computational thinking skills, which though seems big, India is bigger with more than 1.4 billion population, but the growth of CSpathshala (https://cspathshala.org/) is spiraling exponentially (Shah, 2019).

Undoubtedly, other countries are also doing the same, as seen in the publications of Russia, China, Taiwan, Canada, the United States, Spain, Brazil, and Colombia, among others. Even smaller countries such as Chile in Latin America, which more timidly, only implemented elective courses in the third and fourth years of secondary school as a first step in the implementation of the teaching of computational thinking in school. In Chile, although there is no decisive support from the State in this area, the Kodea Foundation of Chile has created a curriculum with didactic material from first grade to fourth grade, which allows schools to have free access to everything necessary to incorporate computational thinking at school (www.ideodigiltal.cl).

Comments on other reviews considered

In 2016, Brackmann et al. conducted a review on computational thinking entitled "Computational thinking: Panorama of the Americas" in which he reflects on the impact of computers in the life of human beings, whose technology advances exponentially, a pace that unfortunately schools cannot follow, or at least cannot follow closely, since the development of the economy around technology encourages its development, and for schools it is more difficult to follow that pace since they depend on public policies, where states are much slower in adapting to changes. The review defines the concept of computational thinking as the skills of abstraction, critical thinking, collaboration, and problem-solving, among other skills. This publication describes an overview of the state of the art of computational thinking in the Americas, to contextualize and guide the incorporation of computational thinking in basic education schools (Brackmann et al., 2016).

In the review made by Hsu et al. (2019), he bases the main part of his findings on defining and redefining the concept of computational thinking, but lacks a detailed description of the skills that compose it, such as those mentioned in the previous paragraph, and does not touch on the relevance of such skills being incorporated in the TIMSS test, and that this will generate an increase in the gap between countries that have incorporated such knowledge vs. those who have not yet done so. He mentions Chile as an example, but in that country, the National Digital Languages Plan is more of an idea that is still not implemented in 2022, as there are 2 elective courses for 3rd year and 4th year of secondary education, which schools are not obliged to teach and which in general are not taught because they do not have teachers. Returning to the review of Hsu et al. (2019), it does not mention or inquire how significant it would be to have one or more didactic strategies for teaching computer programming (scaffolding) and thereby transition to computational thinking skills such as abstraction, decomposition, sequencing, algorithmization, and problem-solving-oriented skills.

It should be noted that the present review is a part of a larger research, whose main objective is the construction and validation of a test to be applied in the first year of higher education with computer science students taking the subject of introduction to programming with the Python programming language. Thus, it seems very pertinent to share the results and conclusions of Tang et al. (2020), which conducted a thorough review of the assessment of computational thinking. Part of his results and conclusions indicates that of the studies carried out, all of them are made for students in elementary and secondary education, and no dedication to higher education is observed, which is shared by this author, since of all the articles reviewed in the total search, which were more than 200, of which 101 articles were reviewed exhaustively, the focus is undoubtedly on school education rather than at the level of vocational training. Another aspect that is important to note is that the studies were conducted in informal education, which undoubtedly affects student's performance, since there is not the incentive of formal education that implies that failing a subject would lead the student to repeat a course, so the pressure on the student to do well on the test would be greater, and thus, the results would be better (Tang et al., 2020).

In the code-centered review conducted by Kite et al. (2021), they highlight the importance of computational programming for transitioning to learning computational thinking and point out that they recognize the historical relationship of computational thinking with computer science. In the review, Kite et al. (2021) point out that there are significant gaps in the conceptualization of computational thinking, highlighting the teaching of computational thinking and the professional development of teachers who will be in charge of teaching computational thinking, which gives rise to the idea on which new research can be based, highlighting the search for code-centered skills and the search for skills from interdisciplinary practices. It is highlighted that of the 80 articles reviewed by Kité et al. 49% of the published research focused on code, highlighting specific issues of computational programming, such as algorithms, abstraction, modularization, debugging, parallelization, loops, and conditionals (Kite et al., 2021).

The importance of teaching computational thinking through computational programming has the characteristic of being a concrete matter, which allows going from the concrete to the abstract that are the skills of computational thinking. It is beautiful to see the students who make their first program that is taught in programming and that only displays by console the message "hello World," they feel satisfaction and feel that they have made their first achievement, and perhaps without knowing it, they have acquired their first practice of abstraction. One way to integrate computational thinking in all students, says Kite et al. (2021), is to incorporate its teaching in the basic and high school curriculum in a mandatory way, which helps to mitigate the digital divide between students, as opposed to making it voluntary, which will generate students who will have advantages over those who for some reasons or another prefer other types of subjects, whether art, history, or craft workshops (Kite et al., 2021).

Finally, as well as the code-centered review by Kite et al. (2021), the review by Ogegbo and Ramnarain (2021), which focuses on the teaching of computational thinking in science classrooms, stands out. Regarding the conceptualization of computational thinking, both the articles reviewed by the author and the systematic reviews all conclude that there is no agreement on the skills of computational thinking, a concept that remains under permanent construction and is redefined as it is applied in different contexts, such as in the arts, language, or science. It should be noted that computational thinking has important consequences for the teaching of science subjects at all levels of education, primary, secondary, and tertiary or higher, especially now that it has been incorporated into the TIMSS tests, a situation that will mark future generations. For Ogegbo, an important result yielded by his review is the concrete concepts of computational thinking skills, which would be composed of: (a) "Decomposition" skill that involves dividing a complex task into smaller parts into manageable components; (b) "Pattern recognition" which involves identifying and defining trends within a problem; (c) "Abstraction," a skill that involves identifying particular similarities and differences between comparable problems to work toward a solution, i.e., understanding the problem in its timeless dimension and not linked to a particular subject; (d) "Algorithm design," which involves the development of step-by-step guidelines to solve a problem; and "Automation," which involves the use of technological tools to mechanize

the solutions to the problems posed (Ogegbo and Ramnarain, 2021).

It should be noted that both the code-based approach and the science classroom approach have certain similarities in terms of applicability, since in the area of science, applications through the use of code are of greater applicability, which has a greater tendency to obtain concrete results. Computational thinking skills are very important for the society of the twenty-first century, which will allow the generations in formation to insert themselves in the global world in an advantageous way and that the new reality of the digital world is a factor that facilitates life and that the complexity is neutralized by the new set of skills that computational thinking delivers. Several aspects in which computational thinking may impact are that it will increase the gap between those who have their skills with those who do not have it, in addition to being voluntary in some countries and mandatory in others, some will be at an advantage over others, and there is also the economic gap between the countries of those who have already implemented it and currently only make improvements, and those who are still only a topic of discussion that is very distant.

Economic and employment impact

A diagnosis made in 2017 points out that the emerging society seen from the technologicalization, is dictated by algorithms, artificial intelligence, automated processes, and robots, which impacts the work and the population, a scenario that does not bring good hopes for those who are not part of the doings in technology, because the technological era will bring workers, low wages, and unemployment. This is the uncertainty of millions of people in the world. The advance of automation threatens at least 14% of current jobs in the world, a figure that has been ratified by the OECD for the last 2 years. In total, 14% of jobs in the 36 richest economies in the world have a high probability of being automated (García, 2018).

A study by McKinsey Global Institute indicates that in Chile, 3.2 million jobs could be replaced by automated systems in the next 20 years. In more developed nations such as the USA, the impact of automation would be 46% of current employment. But in Mexico, the McKinsey report estimates that robots and software could do 52% of the work that exists in that country, while in Peru, the percentage reaches 53%, in Brazil 50% and in Argentina 48%. In the Chilean case, the report estimates that retail and commerce in general would save US$9 billion in wages if 51% of the jobs that have the potential to be automated were replaced; manufacturing industries would save US$6 billion and the administrative and public sector would save US$10 billion in wages. Nationally, the savings in wages would be US$41 billion (Manyika et al., 2017).

The speed of substitution of human labor by automated systems is also affected by demographics. On the one hand, it is expected that by 2050, over one-third of the world's population will be over the age of 50, an indicator that only covered 17.5% of the world's population in 1950. On the other hand, according to published analyses, people over 50 years of age have limitations when performing complex tasks involving technology, being largely surpassed by people of younger ages. This is consistent with the studies that point to the conclusion that the older a country's population is, the greater the acceleration in the adoption of automated systems. The latter would explain why, in relative terms, countries such as the United States and the United Kingdom lag behind Germany, Japan, and South Korea in industrial robotics (Rivera-Taiba, 2019).

In this work, among other things, we seek to link the effect of the advance in the implementation of algorithms as executors of repetitive functions at work, which will leave out of the labor activity a significant number of jobs and those who work in them around the world. Thus, jobs such as accountants, secretaries, cashiers, truck and bus drivers, warehouse workers, and instructors in various areas will be the first to lose their respective jobs. Other professions such as doctors, engineers, architects, and researchers are less feasible to automate; however, there will also be certain types of functions of these professionals that will be automated. In this context, it is important to have adequate training in information technology to be competitive in the world ahead, since in the future, there will be two main categories of jobs: either you are highly specialized or you will have to dedicate yourself to the area of services such as gardening or delivery.

Conclusions

At the conclusion of this review, and after having read and analyzed a multitude of articles, of which dozens of them correspond to other reviews, the author, as a computer science engineer, notes that research runs on two separate threads, on the one hand, the various computer implementations developed by computer engineers and, on the other hand, educators, This has generated that in those countries where the teaching of computational thinking has been implemented or is in the implementation stage, there is a lack of trained teachers in computer science to carry out such a monumental task, and where there are, they are absolutely insufficient. The concept of computational thinking and the skills that compose it are still being discussed in education, whereas computer science publishes multiple applications that educators do not understand or do not have the tools to

transfer this knowledge to the new generations through the educational system.

In the analyzed and compared reviews of Shute et al. (2017), Moreno-León et al. (2018), Hsu et al. (2019), Li et al. (2020), Tang et al. (2020), Kite et al. (2021), and Ogegbo and Ramnarain (2021), among others, coincidences were found in that all the articles take as initial reference (Wing, 2006) from where the concept of computational thinking and its skills were mentioned and defined for the first time. Afterward, there are coincidences that there is no consensus among researchers regarding the skills that define computational thinking, and finally, we all agree that computational thinking is the fundamental skill of the twenty-first century. As for the differences between the reviews, they are rather superficial, some spend more time on qualitative research, and others on analyzing quantitative research, but no significant differences are evident.

From the reviews and articles read, the contribution made by Grgurina (2021) stands out in her work entitled "Getting the Picture: Modeling and Simulation in Secondary Computer Science Education," in whose work she incorporates the concept of modeling and simulation as foundational skills of computational thinking, so that it includes formulating problems, organizing and analyzing data logically, representing data through models, and automating solutions. Thus, looking toward school-based training, these skills enhance attitudes such as: confidence in dealing with complexity, persistence in working with difficult problems, tolerance for ambiguity, the ability to deal with open-ended problems, and the ability to work with others to achieve a common goal and communicate it. Grgurina describes computational thinking in terms of its main concepts, such as: data collection, data analysis, data representation, problem decomposition, abstraction, algorithms and procedures, automation, modeling, and simulation.

From the comparisons with the aforementioned reviews, it stands out that the present review emphasizes the lack of evaluation instruments at the three educational levels to measure the computational thinking and the inexistence of didactic strategies to implement the teaching of computational thinking, in this sense, it will propose in two future publications a didactic methodology for the teaching of computational programming and a test to measure its results. It should be borne in mind that something that is not mentioned in any of the reviews is that computational programming is the necessary scaffolding for teaching computational thinking, so it is necessary to differentiate from those who see both concepts as synonymous, when in fact, they are two different rungs of the same ladder. The above implies that it will necessarily be necessary to implement the teaching of computational programming as a previous step for students to acquire the skills of computational thinking.

Undoubtedly, the implementation of computational thinking in the educational system: primary, secondary, and tertiary, which will prepare future professionals with the skills of the twenty-first century, is of utmost importance in terms of its progressive impact on the future of humanity. It should be noted that in countries where computational thinking is being implemented in schools, it has become a fundamental policy, so that these countries are the ones who will dominate the economy, technology and manage advanced knowledge. Moreover, these countries will train the professionals who will occupy the best jobs, those with STEM characteristics, essentially engineers, doctors, and scientists, among others.

In addition, it should be noted that these countries are mainly located in Europe, Asia, Oceania, and North America. Unfortunately, the countries of the continents of Africa and Latin America are not in this group of advanced countries, with the exception of some, such as Brazil, Colombia, Peru, Chile, and Argentina, among others, whose universities are working on projects aimed at implementing computational thinking in schools. This leads to the conclusion that in the coming decades, the socioeconomic differences between rich and poor countries will become even more extreme. Thus, the poorest countries will see their development possibilities diminish and their economies will continue to be based on mineral extraction and tourism.

When comparing the progress of countries in the teaching of computational thinking in schools, it is necessary to see the progress of the states in public policies that establish this subject in the compulsory curriculum, as is the case of England in 2013 and the countries of Europe in general since 2016, or other Asian countries such as Japan, South Korea, and China where computational thinking is defined as the skills of the twenty-first century and will be the engine of technological and economic development. However, in developing countries, such as Latin America and Africa, although there are several initiatives of universities in doing research on the teaching of computational thinking, the states have other priorities, so the issue is not in the discussion of public policy, but rather, the development of the teaching of computational thinking is done by particular initiatives of universities and some other foundation, which leaves them behind in the race to create and develop new technologies to deliver better employment alternatives and thus greater welfare to the population.

However, there is still hope for those countries where large investments in technology are arriving and leading to the hiring of many professionals and technicians in the technological area, which has prompted the updating of the educational system, at least in the interest of meeting the growing demand for advanced professionals, which begins to turn the machinery of the state, which slowly show proposals for the implementation of computational thinking in schools. An example of a European-rich country is England, which implemented the teaching of computational thinking in schools since 2013.

It is also diagnosed that in the coming years, there will be a high number of jobs that will be automated, jobs such as supermarket cashiers, bank tellers, truck and bus drivers,

warehouse operators, and accountants, and all those that perform repetitive tasks and that constitute activities that can be performed by an algorithm. This will inevitably lead to massive job losses of 3 to 4 million in the case of Chile in the next 10–20 years, and in developed countries where, although the impact will be less, it will also cause greater social and economic instability.

A guideline proposed to countries that have not yet begun the process of implementing the teaching of computational thinking in schools should be the following:

- Put the subject in the public discussion.
- Governments should seek the support of experts in technology and education to make a diagnosis of the situation.
- Implement an educational reform that incorporates the teaching of computer programming in compulsory education.
- That teacher-training universities update their curricula to incorporate the teaching of programming languages such as Scratch, Alice, and Python, so that they can successfully implement the new challenges.
- That industry becomes a part of the educational process by investing in software development and educational robotics in order to push the teaching process in an accelerated manner.

Finally, in the language subject, Scratch should be learned to generate animations of the stories they have to read, to make the subject entertaining and responsive to the children's concerns. In natural sciences, some aspects of nanotechnology, its elements, and how infinitely small things are part of our natural world should be shown. In history, different levels should incorporate all the continents, and their history can be recreated with Scratch or Alice applications, building applications such as Age of Empires. In the arts, designs should be created with apps, not working with paintings, which besides being messy, limit creativity, and move away from the central focus, which is technology training. Educational robotics and gamification should be the main branches that collaborate in the training of future professionals.

The limitations of this review and recommendations to be made

A study that allows us to know the number of teachers trained by country, to teach computer programming and thereby transition to the knowledge of computational thinking.

A gender study that shows the participation of women in the world of computational sciences.

A study that makes an inventory of the instruments duly validated to measure computational thinking at the three educational levels.

A study that quantifies and shows the didactic strategies that exist to teach computational programming.

A study of the universities in the world that have incorporated the teaching of computational thinking in teacher training curricula.

Data availability statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

Author contributions

The author confirms being the sole contributor of this work and has approved it for publication.

Funding

This research was carried out entirely with our own resources.

Conflict of interest

The author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

Adell-Segura, J., Llopis-Nebot, M. Á., Esteve-Mon, F. M., and Valdeolivas-Novella, M. G. (2019). *The Debate on Computational Thinking in Education*. Castello: RIED; Iberoamerican Journal of Distance Education.

Alyahya, D., and Alotaibi, A. (2019). Computational thinking skills and its impact on TIMSS achievement: an instructional design approach. *Issues Trends Learn. Technol.* 7, 3–19. doi: 10.2458/azu_itet_v7i1_alyahya

Angeli, C., and Giannakos, M. (2020). Computational thinking education: Issues and challenges. *Comput. Hum. Behav*. 105, 106185.

Angeli, C., and Valanides, N. (2020). Developing young children's computational thinking with educational robotics: an interaction effect between gender and scaffolding strategy. *Comp. Hum. Behav.* 105, 105954. doi: 10.1016/j.chb.2019.03.018

Atas, M. (2015). The reduction of speaking anxiety in EFL learners through drama techniques. *Procedia Soc. Behav. Sci.* 176, 961–969.

Bain, C., Dabholkar, S., and Wilensky, U. (2020). Confronting frame alignment in CT infused STEM classrooms. *CoolThink@ JC* 91: 91–94.

Bergner, N., Röpke, R., Schroeder, U., and Krömker, D. (2018). *Hochschuldidaktik der Informatik HDI.* Berlin: University Press.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., and Engelhardt, K. (2016). *Developing Computational Thinking in Compulsory Education – Implications for Policy and Practice; EUR 28295 en* JRC: European Union.

Brackmann, C., Barone, D., Casali, A., Boucinha, R., and Muñoz-Hernandez, S. (2016). "Computational thinking: panorama of the Americas," in *2016 International Symposium on Computers in Education (SIIE)* (Salamanca: IEEE), 1–6. doi: 10.1109/SIIE.2016.7751839

Brackmann, C., Román, M., Robles, G., Moreno, J., Casali, A., and Barone, D. (2017). "Development of computational thinking skills through unplugged activities in primary school," in *Proceedings of the 12th Workshop on Primary and Secondary Computing Education* (Madrid), 65–72. doi: 10.1145/3137065.3137069

Brown, N., and Wilson, G. (2018). Ten quick tips for teaching programming. *PLoS Comput. Biol.* 14, e1006023. doi: 10.1371/journal.pcbi.1006023

Buitrago, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., and Danies, G. (2017). Changing a generation's way of thinking: teaching computational thinking through programming. *Rev. Educ. Res.* 87, 834–860. doi: 10.3102/0034654317710096

Calderon, A., Catherine, T., and Crick, T. (2017). *An Investigation into Susceptibility to Learn Computational Thinking in Post-Compulsory Education*. doi: 10.1007/978-3-319-93566-9_14

Carnegie Mellon Center for Computational Thinking. (2010). Available online at: https://www.cs.cmu.edu/~CompThink/

Cheng, Y., Sun, P., and Chen, N. (2018). The essential applications of educational robot: requirement analysis from the perspectives of experts, researchers and instructors. *Comput. Educ.* 126, 399–416. doi: 10.1016/j.compedu.2018.07.020

Ching, Y., Hsu, Y., and Baldwin, S. (2018). Developing computational thinking with educational technologies for young learners. *TechTrends* 62, 563–573. doi: 10.1007/s11528-018-0292-7

Coppelli, G. (2018). Economic globalization in the 21st century. Between globalization and de-globalization. *Estudios Int.* 50, 57–80. doi: 10.5354/0719-3769.2018.52048

Cruz, S. S. T., Rojas, O. E., Hurtado, J. A., and Collazos, C. A. (2013). "ChildProgramming process: a software development model for kids," in *2013 8th Computing Colombian Conference (8CCC)* (Armenia: IEEE), 1–6. doi: 10.1109/ColombianCC.2013.6637535

CSTA Computational Thinking Task Force. (2011). *Operational Definition of Computational Thinking for K-12 Education (Vol. 2013)*.

da Cruz Alves, N., Von Wangenheim, C., Hauck, J., Borgatto, A., and de Andrade, D. (2020). An item response theory analysis of the sequencing of algorithms & programming concepts. *CoolThink@ JC* 9, 1–11. doi: 10.5753/educomp.2021.14466

Daungcharone, K., Panjaburee, P., and Thongkoo, K. (2019). A mobile game-based C programming language learning: results of university students' achievement and motivations. *Int. J. Mob. Learn. Organ.* 13, 171–192. doi: 10.1504/IJMLO.2019.098184

de Paula, B., Burn, A., Noss, R., and Valente, J. (2018). Playing beowulf: bridging computational thinking, arts and literature through game-making. *Int. J. Child Comp. Interact.* 16, 39–46. doi: 10.1016/j.ijcci.2017.11.003

Fanchamps, N., Specht, M., Hennissen, P., and Slangen, L. (2020). *The Effect of Teacher Interventions and SRA Robot Programming on the Development of Computational Thinking.* Hong Kong: The Education University of Hong Kong.

Fotaris, P., Mastoras, T., Leinfellner, R., and Rosunally, Y. (2016). Climbing up the leaderboard: an empirical study of applying gamification techniques to a computer programming class. *Electro. J. e-learn.* 14, 94–110.

García, F. (2018). Editorial computational thinking. *IEEE Rev. Iberoam. Tecnol. Aprendizaje* Salamanca: University Press. 13, 17–19. doi: 10.1109/RITA.2018.2809939

Gómez, M., Palacio, L., Manrique, B., Villada, B., and Arbeláez, S. (2019). "Successful experiences of teaching programming and robotics in elementary and middle school education," in *2019 14th Iberian Conference on Information Systems and Technologies (CISTI)* (IEEE), 1–6.

González, M. (2015). "Computational thinking test: design guidelines and content validation," in *Proceedings of EDULEARN 15 Conference* (Bogota), 2436–2444.

Gougeon, L., and Cross, J. S. (2021). "Japanese elementary schools' playful programming curriculum considerations: readiness, limitations and teacher training," in *2021 IEEE International Conference on Engineering, Technology & Education (TALE)* (Wuhan: IEEE), 23–28. doi: 10.1109/TALE52509.2021.9678771

Grgurina, N. (2021). *Getting the Picture: Modeling and Simulation in Secondary Computer Science Education* Naples: University Press.

Grover, S., and Pea, R. (2018). Computational thinking: a competency whose time has come. *Comp. Sci. Educ. Perspect. Teach. Learn. Sch.* 19:1257–58. doi: 10.5040/9781350057142.ch-003

Hsu, Y. C., Irie, N. R., and Ching, Y. H. (2019). Computational thinking educational policy initiatives (CTEPI) across the globe. *TechTrends* 63, 260–270. doi: 10.1007/s11528-019-00384-4

Huang, P., and Hwang, Y. (2013). An exploration of EFL learners' anxiety and e-learning environments. *J. Lang. Teach. Res*. 4, 27.

Jiménez Toledo, J. A., Collazos, C. A., and Ortega, M. (2021). discovery model based on analogies for teaching computer programming. *Mathematics* 9, 1354. doi: 10.3390/math9121354

Khenner, E. (2019). Introduction to informatics as part of the university-wide general education curriculum. *Международный научный журнал «Современные информационные технологии и ИТ-образование»* 15, 805–814. doi: 10.25559/SITITO.15.201904.805-814

Kite, V., Park, S., and Wiebe, E. (2021). The code-centric nature of computational thinking education: a review of trends and issues in computational thinking education research. *Sage Open* 11, 21582440211016418. doi: 10.1177/21582440211016418

Klunnikova, M., Bazhenova, I., Pak, N., and Kirgizova, E. (2020). Developing students computational thinking with a recursive polydisciplinary approach. *J. Phys.* 1691, 012190. doi: 10.1088/1742-6596/1691/1/012190

Kong, S., Hoppe, H., Hsu, T., Huang, R., Kuo, B., Li, K., et al. (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

Lee, P., and Low, C. (2020). Implementing a computational thinking curriculum with robotic coding activities through non-formal learning. *CoolThink@ JC* 150:150–151.

Li, Y., Schoenfeld, A. H., diSessa, A. A., Graesser, A. C., Benson, L. C., English, L. D., et al. (2020). On computational thinking and STEM education. *J. STEM Educ. Res.* 3, 147–166. doi: 10.1007/s41979-020-00044-w

Liang, Y., and Hsu, T. (2020). Comparison of the learning behaviors of the third grader students integrating robots and the computational thinking board game in Singapore and Taiwan. *CoolThink@ JC* 47, 47–51.

Lockwood, J., and Mooney, A. (2017). Computational thinking in education: where does it fit? A systematic literary review. *arXiv Preprint* arXiv:1703.07659.

Manyika, J., Chui, M., Miremadi, M., Bughin, J., George, K., Willmott, P., et al. (2017). A future that works: automation. *Employ. Product.* 148, 1–135.

Master, A., Cheryan, S., Moscatelli, A., and Meltzoff, A. N. (2017). Programming experience promotes higher STEM motivation among first-grade girls. *J. Exp. Child Psychol.* 160, 92–106. doi: 10.1016/j.jecp.2017.03.013

McLaren, B., Adams, D., Mayer, R., and Forlizzi, J. (2017). A computer-based game that promotes mathematics learning more than a conventional approach. *Int. J. Game Based Learn.* 7, 36–56. doi: 10.4018/IJGBL.2017010103

Moreno-León, J., Román-González, M., and Robles, G. (2018). "On computational thinking as a universal skill: a review of the latest research on

this ability," in *2018 IEEE Global Engineering Education Conference (EDUCON)* (Santa Cruz de Tenerife: IEEE), 1684–1689. doi: 10.1109/EDUCON.2018.8363437

Ogegbo, A. A., and Ramnarain, U. (2021). A systematic review of computational thinking in science classrooms. *Stud. Sci. Educ.* 58, 203–230. doi: 10.1080/03057267.2021.1963580

Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., et al. (2021). The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. *Int. J. Surg.* 88, 105906. doi: 10.1016/j.ijsu.2021.105906

Paucar-Curasma, R., Villalba-Condori, K., Arias-Chavez, D., Le, N. T., Garcia-Tejada, G., and Frango-Silveira, I. (2022). Evaluation of computational thinking using four educational robots with primary school students in Peru. *Educ. Knowl. Soc.* 23, e26161. doi: 10.14201/eks.26161

Pellas, N., and Vosinakis, S. (2018). The effect of simulation games on learning computer programming: a comparative study on high school students' learning performance by assessing computational problem-solving strategies. *Educ. Inform. Technol.* 23, 2423–2452. doi: 10.1007/s10639-018-9724-4

Pérez, D., Hijón, R., Bacelo, A., and Pizarro, C. (2020). Can computational thinking be improved by using a methodology based on metaphors and scratch to teach computer programming to children?. *Comp. Hum. Behav.* 105, 105849. doi: 10.1016/j.chb.2018.12.027

Pérez, D., Hijón, R., and Martín, M. (2018). A methodology proposal based on metaphors to teach programming to children. *IEEE Rev. Iberoam. Tecnol. Aprendizaje* 13, 46–53. doi: 10.1109/RITA.2018.2809944

Portnoff, S. (2018). The introductory computer programming course is first and foremost a language course. *ACM Inroads* 9, 34–52. doi: 10.1145/3152433

Psycharis, S., Kalovrektis, K., and Xenakis, A. (2020). A conceptual framework for computational pedagogy in STEAM education: determinants and perspectives. *Hellenic J. STEM Educ.* 1, 17–32. doi: 10.51724/hjstemed.v1i1.4

Radin, B. A., and Hawley, W. D. (2013). *The Politics of Federal Reorganization: Creating the US Department of Education.* Washington: Elsevier.

Repenning, A. (2017). Moving beyond syntax: lessons from 20 years of blocks programing in AgentSheets. *J. Vis. Lang. Sent. Syst.* 3, 68–91. doi: 10.18293/VLSS2017-010

Rivera-Taiba, T. (2019). Effects of automation on employment in Chile. *J. Econ. Anal.* 34, 3–49. doi: 10.4067/S0718-88702019000100003

Rojas, A., and García, F. J. (2020). Assessment of computational thinking for learning computer programming in higher education. *J. Dist. Educ.* 20, 1–36. doi: 10.6018/red.409991

Román-González, M., Pérez-González, J., and Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Comput. Hum. Behav.* 72, 678–691.

Seow, P., Wadhwa, B., Lim, Z., and Looi, C. (2020). *Towards Using Computational Modeling in Learning of Physical Computing: An Observational Study in Singapore Schools* Singapore: University Press.

Shah, V. (2019). CSpathshala: bringing computational thinking to schools. *Commun. ACM* 62, 54–55. doi: 10.1145/3343445

Sharma, K., Papavlasopoulou, S., and Giannakos, M. (2019). Coding games and robots to enhance computational thinking: how collaboration and engagement moderate children's attitudes? *Int. J. Child Comp. Interact.* 21, 65–76. doi: 10.1016/j.ijcci.2019.04.004

Shute, V. J., Sun, C., and Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educ. Res. Rev.* 22, 142–158. doi: 10.1016/j.edurev.2017.09.003

So, H. J., Jong, M. S. Y., and Liu, C. C. (2020). Computational thinking education in the Asian Pacific region. *Asia Pac. Educ. Res.* 29, 1–8. doi: 10.1007/s40299-019-00494-w

Souza, I., Andrade, W., and Sampaio, M. (2019). "Analyzing the effect of computational thinking on mathematics through educational robotics," in *2019 IEEE Frontiers in Education Conference (FIE)* (Covington, KY: IEEE), 1–7.

Tang, X., Yin, Y., Lin, Q., Hadad, R., and Zhai, X. (2020). Assessing computational thinking: a systematic review of empirical studies. *Comput. Educ.* 148, 103798. doi: 10.1016/j.compedu.2019.103798

Topalli, D., and Cagiltay, N. (2018). Improving programming skills in engineering education through problem-based game projects with scratch. *Comput. Educ.* 120, 64–74. doi: 10.1016/j.compedu.2018.01.011

Vallance, M., and Towndrow, P. (2016). Pedagogic transformation, student-directed design and computational thinking. *Pedagogies Int. J.* 11, 218–234. doi: 10.1080/1554480X.2016.1182437

Wing, J. (2006). Computational thinking. *Commun. ACM* 49, 33–35. doi: 10.1145/1118178.1118215

Wing, J. (2011). Research notebook: computational thinking—what and why. *Link Magaz.* 6.

Wu, L., Looi, C., Multisilta, J., How, M., Choi, H., Hsu, T., et al. (2020). Teacher's perceptions and readiness to teach coding skills: a comparative study between Finland, mainland China, Singapore, Taiwan, and South Korea. *Asia Pac. Educ. Res.* 29, 21–34. doi: 10.1007/s40299-019-00485-x

Yi, L., and Ting, H. (2020). "Effects of using mobile phone programs to control educational robots on the programming self-efficacy of the third grade students," in *Proceedings of International Conference on Computational Thinking Education*. p. 31–35.

Zhang, K., Chen, X., and Wang, H. (2019). "Research on the mixed-learning model and the innovative talent cultivation mechanism based on computational thinking," in *Recent Developments in Intelligent Computing, Communication and Devices* (Singapore: Springer), 59–65. doi: 10.1007/978-981-10-8944-2_8

Zhou, T., and Hsu, T. (2020). Learning behaviors analysis of the six grader students integrating educational robots with the computational thinking board game. *CoolThink@ JC* 144, 144–148.