mathematics

Article

Teaching and Learning to Program: Umbrella Review of Introductory Programming in Higher Education

Sónia Rolland Sobral

REMIT (Research on Economics, Management, and Information Technologies), Universidade Portucalense, 4200 Porto, Portugal; sonia@upt.pt

Abstract: Introductory programming courses in science, technology, engineering, and mathematics (STEM) degrees are critical to student success throughout the students' academic and professional route but have traditional failure and dropout rates. The number of publications and reviews on the subject is growing, so we realize the need to prepare an umbrella review, or review of reviews, to group previous studies and extract more comprehensive and ambitious results. Based on the databases ACM, Google Scholar, IEEE, SCOPUS and Web of Science libraries, a formal search was created that resulted in 21 reviews on programming fundamentals in higher education context. Results include bibliometric information on the CS1 reviews, in the context of higher education (namely annual evolution, number and information on authors, types and sources of information, countries of affiliation, languages, keywords and most cited articles), the purpose of the reviews and research questions, methods (namely search strategy, databases used, eligibility criteria) and results (number of records and tables or divisions were made to catalog the articles). We present a taxonomy with four different types of purpose: general, specific, student group and teacher directed. We found very interesting catalogs that can serve as a tool for future work, whether by researchers in the field or by authors who intend to carry out reviews related to introduction to programming, especially in the context of higher education.

Keywords: introduction to programming; review; umbrella review; higher education; CS1; STEM

Citation: Sobral, S.R. Teaching and Learning to Program: Umbrella Review of Introductory Programming in Higher Education. *Mathematics* **2021**, *9*, 1737. https://doi.org/10.3390/math9151737

Academic Editor: David Pugalee

Received: 26 June 2021
Accepted: 21 July 2021
Published: 23 July 2021

1. Introduction

Introduction to programming is the curricular unit that promotes the initial contact of science, technology, engineering and mathematics (STEM) undergraduate students with computational thinking and programming languages. Programming fundamentals are more than coding, more than translating an algorithm into a language that computer can understand. It is necessary to think and solve the problem to create an algorithm [1]. Much research has been done to explain the traditional failure and dropout rate of students in these early programming units, often responsible for the dropout of students in undergraduate courses [2,3] and also to improve retention in the major [4,5]. This problem concerns researchers and teachers who try in various ways to combat poor results and contribute to the success of the teaching-learning process, sometimes adapting the design of the units and learning objectives [6–8]. The Association for Computing Machinery (ACM) and Institute of Electrical and Electronics Engineers (IEEE) curriculum recommendations are a powerful tool for those who must design courses and more precisely those curricular units. In one of the first curricular recommendation documents [9], this curricular unit was named CS1, computer science 1, a name that has already been changed to introduction to programming or programming fundamentals [10,11]. There have been made distinctions between types of courses [12], departments that teach courses or even by continents [13]. While in many countries computer programming is included in early years, from kindergarten to 12th grade (K-12) [14], in other countries students enter university without having any knowledge related to computers [13].

Many of the studies are concerned with the paradigm to adopt [15] and the programming language [1,16], an old discussion [17]. Various techniques such as educational data mining have been used to predict failures in advance [18–20], also concerned with initial motivation [21] or promoting increased motivation [22]. Several experience reports have been made to improve the teaching—learning process and the respective success in introducing programming [23,24]. Many of them are related to pedagogical strategies with encouraging results, such as pair programming [25–27], flipped classrooms [28–30], peer instruction [31,32], cooperative learning [33] and collaborative learning [34]. The attempt to involve students by promoting "learning by doing" has been widely used with active learning approaches [35,36], like Project Based-Learning [37] and Problem-Based Learning [38]. There have been many approaches to the use of gamification [39,40], robots [41,42] and augmented reality [43]. To improve online learning [44,45], use of analogies [46], automatically providing feedback [47,48] or incorporate social media [49]. There are studies that direct to the instruments like Mobile Phones [50] and bring-your-own-device (BYOD) to the exam [51]. There is concern about not leaving anyone out like students with disabilities [52] and having gender concerns, especially regarding how to attract women to informatics [53,54].

As described, there are different perspectives and approaches. There has been an exponential growth of work in this field, as well as an exponential growth of literature reviews, to rigorously bring together several studies for informed decision-making. The following figure (Figure 1) shows this exponential growth from 2001 to 2020, considering groups of 5 years and two axis scales (left for publications and right for revisions), using a search in the SCOPUS database carried out in April 2021 with the words "introductory programming", "introduction to programming", "novice programming", "novice programmers", "learn programming", "Learning to program", "teach programming", "teach* programming", "learn* programming", "programming education", "Learning to program" or "teach to program" to publications and also "review*" for reviews. We know that these numbers are only indicative, as publications appear in the results that are not exactly the intended ones, such as "code review" instead of the literature review perspective.

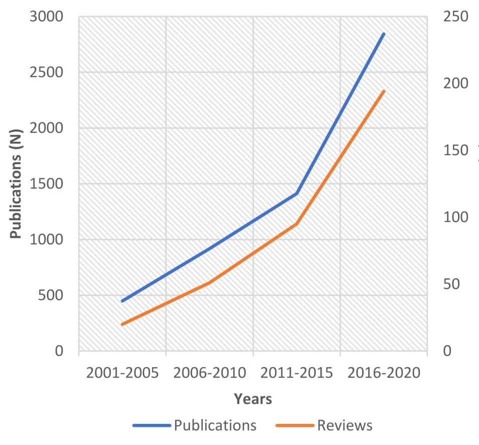

Figure 1. Publication and review grow, CS1, SCOPUS search (2001–2020).

For these reasons, we think that there is a need for an innovative study that groups these literature reviews, a kind of review of the reviews, so that they can serve as an instrument for work in the area. The research questions to be addressed by this study are:

RQ1: What is the bibliometric information on the CS1 reviews, in the context of higher education (namely annual evolution, number and information on authors, types and sources of information, countries of affiliation, languages, keywords and most cited articles)?

RQ2: What is the purpose of the reviews and what research questions do they set?

RQ3: What were the methods used to find the studies each review was based on (namely search strategy, databases used, eligibility criteria)?

RQ4: What were the results found (number of records and which tables or divisions were made to catalog the articles)?

This document is divided into methods, results and discussion as suggested by PRISMA checklist [55].

2. Methods

2.1. Design

Thus, we identify reviews of introductory programming in the context of higher education, commonly known as CS1. This overall review was conducted following the Guidelines for performing Systematic Literature Reviews in Software Engineering, namely the Protocol for a Tertiary study of Systematic Literature Reviews and Evidence-based Guidelines in IT and Software Engineering [56]. This umbrella study is also in accordance with the Cochrane Manual principles for systematic reviews of interventions [57] and the preferred reporting items for systematic reviews and meta-analyses (PRISMA) [55]. In review of reviews, the unit of search, inclusion and analysis of data is the systematic review, not the primary study [57]. There are several different denominations for systematic reviews of reviews such as umbrella reviews, overviews of reviews, reviews of reviews, a summary of systematic reviews and synthesis of reviews [58]. Considering individual studies (or primary research) as the lowest level of the synthesis pyramid, the intermediate levels will be the systematic review, followed by the meta-analysis and finally, the top level will be the umbrella reviews [59]. A tertiary review uses almost the same methodology as a standard systematic literature review.

2.2. Search Strategy

A search string was constructed and the logical operators used to complement the execution results. For each database, it was necessary to build a specific query, as each one has a different syntax; and an example of a resulting query is shown below.

TITLE ((“introductory programming” OR “introduction to programming” OR “noviceprogramming” OR “novice programmers” OR “learn programming” OR “learning to program” OR “teach programming” OR “teach* programming” OR “learn* programming” OR “programming education” OR “Learning to program” OR “teach to program”) AND (revie*)).

2.3. Information Sources

We considered five different databases to execute the search strings, shown in Table 1. The choice of the ACM, IEEE, SCOPUS and WoS databases was made because they are the most used databases in computer science with high quality and peer reviewed, among other reasons [60,61], while the GoogleS database seems to be more complete [62,63].

Table 1. Databases used in the search (access date 19 May 2021).

Name	Acronym	URL
ACM Digital Library	ACM	https://dl.acm.org/
IEEE Xplore	IEEE	https://ieeexplore.ieee.org/
Google Scholar	GoogleS	https://scholar.google.com/
Scopus	SCOPUS	https://www.scopus.com/
Web of Science	WoS	https://apps.webofknowledge.com/

2.4. Eligibility Criteria

We did not consider language as an inclusion or exclusion criterion because we have access to a greater number of articles. The type of articles accepted were journals, conference proceedings and reviews, so we excluded editorials, for example. The size of the articles to be included must be at least three pages long. The time horizon was not used, considering all publications until May 2021. The study design must be of the review type to be included. Study settings is being in a higher education context, introductory programming, or similar focus. The identification of inclusion and exclusion criteria are listed in the following table (Table 2).

Table 2. Eligibility criteria.

Criteria	Inclusion	Exclusion
Country	All	.
Date	All (until May 2020)	-
Design	Review	Nor review
Language	All	-
Length	More than three pages	Three pages or less
Publication Type	Article, Review and Conference	Not Article, Review or Conference
Settings 1	introductory programming or similar	Not introductory programming or similar
Settings 2	higher education context	Not higher education context

2.5. Selection Process

Research result records were found that were not related to the present study, such as related with "peer review" or "program review". First, the databases were searched, then the records were joined and the duplicates eliminated. Subsequently, the records that were not eligible for the inclusion criteria were deleted, first through the title, then through the abstract and then through the reading of the full article. There were only doubts in three records, so the opinion of two experts was asked to get the most correct results.

3. Results

The literature search identified a total of 42 unique records after removing the duplicate records retrieved by ACM (n = 11), IEEE (n = 10), GoogleS (n = 7), SCOPUS (n = 34) and WoS (n = 26). A total of 14 records were excluded by title and/or abstract, one record because we cannot access the entire text and six for other reasons (length three pages or less, not formal computer science 1 or similar and not higher education context). The result is a sample resulting in 21 reviews on programming fundamentals in the context of higher education. Figure 2 shows Study selection flow diagram (PRISMA diagram adapted) [55]. It was very interesting to see that all papers retrieved from the ACM and IEEE libraries were at SCOPUS too. It was found that only one GoogleS record was unique, but it was discarded as it was not a review in the literature sense but a sequence of words "self-review". In the end, it turns out that there are only two articles that are not included in the SCOPUS database, but in WoS: they are two articles from the journal Tecnológicas and written in Spanish.

3.1. Bibliometrics

Bibliometrics is the statistical analysis of published works, a term used since 1969 by Pritchard [64] and has been used to analyze scientific literature in various fields [65]. Bibliometrics refers to the research methodology employed in library and information sciences, which utilizes quantitative analysis and statistics to describe articles' distribution patterns within a given topic, field, institution or country [66]. Bibliometric studies can build solid foundations for advancing a field, allowing scholars to obtain a complete overview and identify gaps in knowledge, as well as allowing for the derivation of new ideas for investigation [67], a way to provide a valuable reference for future research [68]. This process was originated in medical science due to the increasing amount of research in each area [69]. Consequently, it was necessary to identify and guide the research towards an uninvestigated subject [70]. The scientific community has proposed steps to apply these protocols, more specifically in the software engineering area. In this section we will make use of bibliometric analysis to obtain the information we propose, namely the annual evolution of the research, type of publications, information about authors and their affiliations, publication sources and keyword analysis.

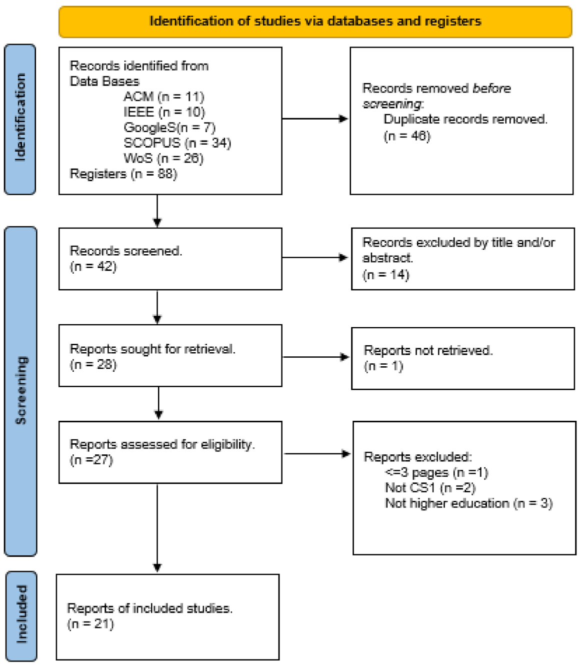
Identification of studies via databases and registers
Identification
Records identified from Data Bases
ACM (n = 11)
IEEE (n = 10)
GoogleS(n = 7)
SCOPUS (n = 34)
WoS (n = 26)
Registers (n = 88)
Records removed *before screening*: Duplicate records removed. (n = 46)
Screening
Records screened. (n = 42)
Records excluded by title and/or abstract. (n = 14)
Reports sought for retrieval. (n = 28)
Reports not retrieved. (n = 1)
Reports assessed for eligibility. (n =27)
Reports excluded: <=3 pages (n =1) Not CS1 (n =2) Not higher education (n = 3)
Included
Reports of included studies. (n = 21)

Figure 2. Study selection flow diagram (PRISMA diagram adapted).

The figure below (Figure 3) shows the number of articles by year of publication. It appears that although there is no reference to the variable year in the inclusion/exclusion criteria, the first article in the database is dated 2011. The year with the most publications is 2019, with 6 publications (29%; n = 6).

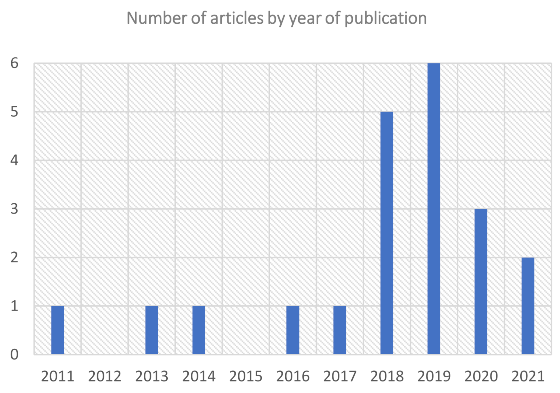

Figure 3. Number of articles by year of publication (May 2021).

The next figure (Figure 4) shows the article percentages by type. 72% of the articles are conference proceedings (72%; n = 15).

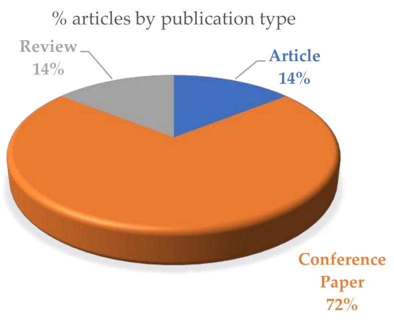

Figure 4. % Articles by publication type.

There are 16 different publication sources: ACM International Conference Proceeding Series publishes three of the articles, ACM Technical Symposium on Computer Science Education, ACM Transactions on Computing Education and journal Tecnológicas publish two articles each. Seven articles are linked to the ACM and three to the IEEE.

Three articles were written in Spanish and the rest in English (86%; n = 18).

Figure 5 shows the number of authors by article. There are articles with 1 to 5 co-authors and one article with 11 co-authors. The average is 3.29 authors per article, with a standard deviation of 1.93 and the mode is 3 (48%; n = 10).

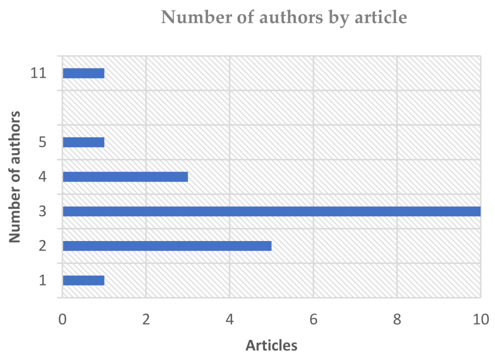

Figure 5. Number of authors by article.

There are 61 different authors, eight of them having two articles. The following table (Table 3) presents these authors name with the indication of their affiliation and ORCID id.

Table 3. Authors with two articles, respective affiliation and ORCID.

Author	Affiliation	ORCID
Anabela Gomes	Coimbra Polytechnic—ISEC, Coimbra, Portugal	0000-0001-8418-8095
Andrew Luxton-Reilly	The University of Auckland, Auckland, New Zealand	0000-0001-8269-2909
António José Mendes	Universidade de Coimbra, Coimbra, Portugal	0000-0001-6659-660X
Brett A. Becker	University College Dublin, Dublin, Ireland	0000-0003-1446-647X
Cesar A. Collazos	Universidad del Cauca, Popayan, Colombia	0000-0002-7099-8131
Javier Alejandro Jiménez-Toledo	Institución Universitaria CESMAG, Pasto, Colombia	0000-0003-3489-3663
Leonardo S. Silva	Universidade de Coimbra, Coimbra, Portugal	
Oscar Revelo-Sánchez	Universidad de Nariño, Pasto, Colombia	0000-0003-2882-5779

The articles have authors with affiliation in 15 countries. The authors with affiliations in Brazil are responsible for 14% of the articles and from Finland for 12% of the articles, as we can see in the following figure (Figure 6).

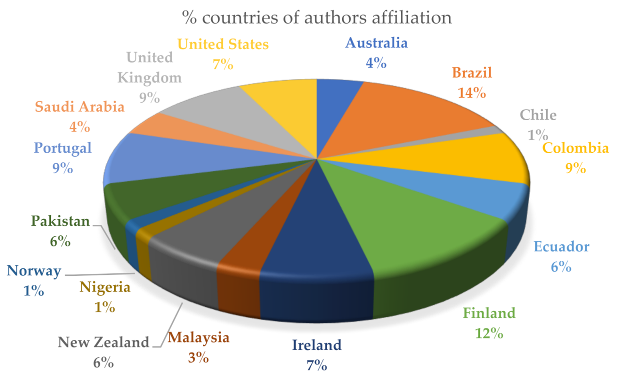

Figure 6. % Countries of authors affiliation.

There are 72 different keywords, the most frequent being systematic review (seven times), introductory programming and programming (six each) and cs1 and programming education (five each). The following figure (Figure 7) represents these keywords in the cloud words format created from the website WordClouds.com [71].

Figure 7. Cloud keywords.

Considering the number of citations presented in SCOPUS, which excludes the two articles published in the journal Tecnológicas, we verified that there are three articles that, at the time of the research, have about 100 citations (one with 96 [72] and two with 95 citations [73,74]).

3.2. Purpose

We have divided the articles into four different types of purpose: general, specific, student group and teacher directed. In the specific type, we find nine different goals: Blended learning [75], collaborative work [76], computational thinking [77], Collaborative learning with computer support [78], Intelligent programming tutors [79], MIT App Inventor [80], Robotics [81], Serious programming games [82] and Self-regulation of learning [83]. The papers aimed at students relate to a specific audience: blind [84] and novice [85,86], for the generality of students [72] or directed to the anxiety of students [87]. There are five articles that are general in scope [74,88–91] and two that are aimed at teachers and teaching methodologies [73,92].

In PRISMA checklist [93], it is suggested that authors include a systematic review, meta-analysis, or both in the title of the report. Analyzing the title of the articles, we found that the most frequent ($n = 8$) is the use of the designations "A systematic literature review" and "A systematic review" ($n = 5$). Other authors use the longer "A Systematic review of the literature", "A literature review", "A Mapping Review" or the more synthetic "A review". Titles can be descriptive (containing the essential elements, namely patients/subjects, interventions, comparisons/control, outcome, and study design—PICOS—but not the conclusion), declarative (shows the main finding of the study) or interrogative (search question in the title) [94]. None of the articles have an interrogative or declarative title. Eight of the articles use the designation "Introductory Programming" [72–75,85,88–90], five use "Programming Education" [72,77,78,86,91], two just the word "Programming" [81,86] while the rest use teaching programming [80,84,92], learning programming [82,87] or both [76]. The articles referred to that have a special purpose, have the designation of the purpose in the title. Others use words like "evolution" [88], "analysis" [85] or even "innovative approaches" [92].

Two of the articles do not explicitly have research questions; however, we have adapted the text to potential research questions. Almost all research questions start by identifying topics and measuring the methods—the most efficient or the most used instruments. There are articles that intend to know the programming languages used and others the methodology designs. We chose to list all research questions for each article. In Appendix A

(Table A1) we present the reference (author and year), the title, purpose and research questions of each article, including the two adapted by us and mentioned above.

3.3. Methods

A total of 25 different sources of information were used. The mean is 4.57, the standard deviation is 2.79, the minimum is one and the maximum is 12 sources of information used by each review. ACM Library is the most used data base (n = 19), followed by the IEEE Digital library (n = 18), ScienceDirect (n = 9), ERIC and SpringerLink (n = 8), Google Scholar (n = 5), Web Of Science (n = 4), SCOPUS (n = 3) and the remaining 17 sources of information were used only in one or two studies.

We can consider that the keywords (search strategy) are divided into two parts: the first one related to CS1 and the second related to the specific objective of the literature review. The latter does not exist for reviews that have a general purpose. There are 45 different keywords to search for CS1 related articles, with CS1 (9 times), programming (7 times) and intruductory programming (6 times) being the most frequent keywords. In the second type, there are several words used, for example (robots OR robotics) [81], "App Inventor" [80], Anxiety [87] or cscl [78]. Interestingly, we found that only one of the records [74] uses asterisks for complementary search keywords.

Inclusion criteria are varied, for example language criteria (only articles in english [82,89,91], or in english or spanish [76]),: there are cases where it is limited to one or more publications soureces [73,88], a certain period of time [85,86], domain of computer science [85], directly answered one or more research question [87], beyond the obvious obligations of being related to higher education [75], for example. The exclusion criteria, in addition to those that contradict the inclusion criteria, are related to the minimum size of the article [85,88,89], work in progress [74] or did not address the research questions [89].

In Appendix B (Table A2) we present the reference (author and year), data bases, the search strategy (first part and second), as well the inclusion and exclusion criteria.

3.4. Reviews Results

The average of articles contained in each review is 166, with the minimum number being 10 [72] and the maximum number 2189 [74].

The way each systematic review structures its results is one of the most important points in this umbrella review. As described above, we found four different types of purpose: general, specific, student group and teacher directed. We found very interesting divisions. We will then list these divisions by category and article. We tried to make a model, but it seems to us that a model would lose the spectacularity of the different divisions found in each of the articles.

(a) Five articles that are general in scope [74,88–91]

Ref. [88] First Languages & Paradigms (General languages & paradigms, Specific paradigms, Specific text-based languages, Other), CS1 Design, Structure & Approach (General design & structure & approach, How CS1 relates to CS0 or CS2, Dealing with ACM model curricula, Physical Settings > [Online, remote or MOOC delivery, Labs, Other physical settings], Other), CS1 Content (Upper-level topics in CS1, Software engineering approaches, other), Tools (Editors & APIs, Libraries, Visualization, Debugging & testing, Other), Collaborative Approaches (Pair programming, Peer instruction, Other), Teaching (General teaching, Model problems & exercises, Specific topics (arrays, recursion, etc.), Games, Hardware (robots, etc.), Aids, examples & tricks, Flipped approaches, Video, Other), Learning & Assessment (General learning, Conceptual or cognitive issues, Learning styles, Reading, writing, tracing & debugging, Errors, Other learning, General assessment, Automatic tutoring & assessment systems, Authentic assessment, Exams, Other assessment), Students (Non-majors. Retention, Gender, diversity, inclusion & accessibility, Prior knowledge, Concept inventories, geek genes, misconceptions, Predicting & measuring success, Programming process data, Other).

Ref. [89] Overview of methods (Pedagogical interventions evaluated through analysis of student performance and/or feedback, survey, questionnaires, interviews and/or focus groups, observation of students' learning and strategies, analysis of students' learning styles and strategies, analysis of student performance with regard to type of assessment, pedagogical approach or contextual factors, analysis of student errors, comparison between teachers« predictions/perceptions and students' actual performance, theoretical reflection based on teaching experience, literature review), student skills (category + skill) (Programming-related skills [problem solving, mathematic ability, previous knowledge in programming, abstraction], General education skills [basic knowledge in English, critical thinking and discussion skills, creativity, time management]), student difficulties (category + difficulty) (problem formulation [problem solving, abstract nature of programming, algorithmic and logical reasoning], solution express [syntax of programming language, control structures, data structures, structure of the code, Others], solution execution and evaluation [debugging, tracing the execution], behavior [motivation and engagement, time management, study skills, confidence, perception of programming as a complete discipline], challenges [methods and tools for teaching programming, scale problem, keeping students' motivation, engagement and persistence, teacher-student communication and feedback, programming language, curriculum and instructional sequences, addressing students' inadequate mathematical background]).

Ref. [74] The student (student learning, underrepresented groups, student attitudes, student behavior, student engagement, student ability, the student experience, code reading, tracing, writing and debugging), Teaching (teaching tools, pedagogical approaches, theories of learning, infrastructure), Curriculum (competencies, programming languages, paradigm), Assessment (assessment tools, approaches to assessment, feedback on assessment, academic integrity).

Ref. [90] Teaching/learning problems of computer programming (Difficulties in learning the computer programming, Disadvantages in teaching the computer programming), Tools used in the teaching-learning programming of computers (Visualization or algorithm simulators, Automatic evaluation tools, educational games focused on teaching a specific learning unit, Educational games focused on the teaching of multiple learning units, collaborative environments), Strategies used in teaching-learning of Computer programming (Intervention strategies, Pair programming, Pair code evaluation, Intelligent collaborative virtual environment, Collaborative environments, MOOC Visualization systems, Multimedia tools, Intelligent tutoring systems, Visual learning tools, Serious games, Educational games, Pseudo-languages, Video game creation, Analogies, Metaphors, Robots, Problem Based Learning (PBL), Augmented Reality, Mixed Reality, Frameworks, Programming paradigm and programming language, Procedural programming, Current programming language, Peer tutoring, Languages with a simple syntax, Pre-programming course, Support tools, Projects, Environments, Development of mental models, Spiral approach, Finite state machines, Lectures and laboratory classes, Automatic qualification tools), Methodological considerations in a first programming course Computers (Semiotic ladder, Cognitive taxonomy objectives, Problem solving, Own didactics, Constructivist, Constructionist, Based on the methodology of the didactic process, Based on teaching, Based on the curriculum, Instructional development based on video games, Instructional development oriented towards paradigm, Instructional development focused on particular VARK topics, Based on coding and debugging, Implementation of basic algorithms, Focused on algorithmic complexity).

Ref. [91] year; publication source; number of pages; 25 Most Cited Publications; Research Methods (simple, complex, or mixed); Educational Levels and Participants (k12, high school . . .), Course Contexts and Themes (CS1 . . .), Pedagogical Approaches and Observations, Tools and Frameworks (Programming languages, Graphical User Interfaces/Computer Graphics, Robotics), Programming languages that were used in more than two publications.

(b) Two articles focus on teachers and teaching methodologies [73,92].

Ref. [73] Pass rates and study sizes before and after teaching intervention (Descriptive, min, max, median, mean, sd) (pass rate pre, pass rate, students pre, students post), intervention tag (Descriptive, min, max, median, mean, sd) (collaboration, content change, contextualization, game-theme, grading schema, group work, media computation, peer support, support), Primary Intervention Effect (Collaboration and Peer Support, Bootstrapping, Relatable Content and Contextualization, Course Setup, Assessment, Resourcing, Hybrid Approaches, Comparing Primary Interventions).

Ref. [92] Year, Teaching Context, Proposal of Studies, Main Results, Main Challenges.

(c) Five that are focus on students: blind [84] and novice [85,86], for the generality of students [72] or directed to the anxiety of students [87].

Ref. [84] Quantitative Details, Publication Topics, Programming Languages, Assistive Devices.

Ref. [85] year, publication title, Country, theoretical framework (Self-Determination Theory or Four-Phase Model of Interest Development), Research Goals: (Tasks [Assignments 1b. Projects], Use of technology [Robots, Augmented reality, Games/gamification], Pedagogical approach [Active learning, Collaborative learning, Feedback, Problem-Based, Flipped classroom, Peer instruction, Competition]), Instrument Occurrence (Grades, Surveys, Semi-structured interviews, Self-reports), Qualitative analysis of source-code, Observation, Diaries, Focal groups.

Ref. [86] Development Phase for CS1 (Problem abstraction and solution proposal, Formalizing the solution, Implementing the formalized solution through tools support, Testing the implemented solution and Documentation and submission of work), Support for Instructor (Active learning through tools, Active learning without using tools, Assessment tools, Curriculum and Threshold Concepts, Collaborative learning and other pedagogy ideas). Novice Programming Environment and Intelligent Programming Tutors.

Ref. [72] Knowledge (syntactic, conceptual, strategic), exemplars of difficulties, Major Related Factors, Potential Strategies and Tools.

Ref. [87] Reference, Keywords, Publication Source, Type (Conference or Journal), Type of study (Questionnaire, Literature Review, Interview, Experimental) and Number of participants.

(d) Nine with specific focus type: Blended learning [75], collaborative work [76], computational thinking [77], Collaborative learning with computer support [78], Intelligent programming tutors [79], MIT App Inventor [80], Robotics [81], Serious programming games [82] and Self-regulation of learning [83].

Ref. [75] Publication year and geographic distribution (continent); Quality assessment (N Met criterion and N Did not meet) (Outcome measures; Background/literature review; Sample; Study design or methodology; Conclusions); Blended learning components (Online self-paced, face-to-face instructor-led, online collaboration, face-to-face collaboration, online instructor-led); Approaches applied by the reviewed studies.

Ref. [76] Country, Type of publication, (Matrix type of study vs learning approach) Type of study (Solution proposal, evaluation research, opinion articles, Systematic review) and learning approach (Collaborative; Cooperative)), Other significant contributions from the authors, TAC mentioned in documents organized by type cases) (Dialogue [Conversation groups, Reciprocal teaching among peers, Learning cells, Role play, Gamification), Problem solving [Structured problem solving], Graphic information organizers [Virtual worlds, Virtual environment, Specific tool], Writing [Collaborative writing, Peer correction], Other [Integrated collaborative strategy].

Ref. [77] Year, publication type, country, Teaching approach & study technique, Learners Context (university and country), Settings (formal or informal), Study impact/evaluation.

Ref. [78] Description, result, Quality analysis (Y, N, partial N/A) (Q1) research objectives definition; (Q2) sample description; (Q3) description of the experimental context (in this study, it is related to programming education); (Q4) description of the experimental design; (Q5) use of randomized controlled trial; (Q6) use of pre/post testing; (Q7) use of control/experimental group), Collaborative resources used in CSCL (Collaborative programming editor, Scaffolding for problem discussion, Peer review, Chat, Sharing techno logical resource, Pair-programming support, Physical, Discussion forums, Group task assignment, Individual contribution awareness, Automatic group formation, Programming observation, Group goals definition, Gamification, Algorithm sharing, Project-Based, Task recommendation, Help request, Voting-system), Outcome (Programming knowledge, User-acceptance, Specific metrics, Number of defects, Creativity index, Algorithm complexity, Code correctness, Motivation, Logical tests, Self-efficacy, Engagement, Self-regulated learning), Collaboration measurements techniques (Analysis of how students interacted with the platform, Interviews, Dialog analysis and group interaction observation, Peer evaluation, Number of algorithm lines, Group self-efficacy, Expert evaluation, Social Network Analysis, Multimodal data).

Ref. [79] Programming Languages, Primary Part of Tutoring that is Adapted (Adaptive or Intelligent Feedback on Programs, Adaptive Navigation), Supplementary Features of the IPT (Questions, User Generated Plans or Visualizations, Lesson Content and Reference Material Pre-made Plans or Visualizations) and Solutions.

Ref. [80] Generalities, Types of applied research (experimental research design, non-experimental studies, studies that use both approaches), Teaching strategies (Problem-Based Learning, Project-Based Learning, Studio-Based Learning and Mission-Based Learning), programs, methods of evaluation, Objectives, Main findings (by educational levels, according to previous experience in programming, comparison of MIT App Inventor with other environments, Advantages identified in the use of MIT App Inventor).

Ref. [81] Year, Quality Assessment of Included Studies (11 items, one or zero for each, Research, Aim, Context, Research Design, Recruit Strategy, Group Control, Data Collection, Data Analysis, Relationship, Findings and value), SLR Quality Evaluation, SLR Quality Evaluation (After the Removal of Papers that Scored Five or Less), Computer Languages Used, Type of Robot Used, suggest that using robotics to teach introductory programming is effective.

Ref. [82] Classification (empirical, non-empirical studies), Concepts Literature (Problem Solving, Algorithm/Pseudo code Design, Programming Fundamentals, Flow Charts, Syntax and Logic, Data Type, Variable Declaration, Input and Output, Operators and Expression, Sequence Structure, Selection/Conditionals/sequence, Repetition Structure and Functions and Parameters, Arrays & Exception Handling, Recursion, Stings, Link List, Pointers OOP Concepts & Design, Data Structure and Algorithm, Others), Categorization of studies (Languages C/C++, SQL, Java/Java Script, Python, Not Mentioned), Type of Game (Role Play, Puzzle Board/Hybrid * Real Time Strategy, Not specified), Platform (Web Desktop, Mobile, Class room, Not state), Game Elements of MDA Framework (Mechanics, Dynamics, Aesthetics). Evaluation Methods Matrix (Evaluation Methods, Research Study) (Quasi Experimental Design [Pre-Test, Post-Test], Formal Interviews [Structured, Unstructured, Semi Structured, Questionnaire/Survey, Feedbacks [Expert Feedback, Player Feedback, Skill Test/Assessment Tool, Game Play Statistics, Observation, Others).

Ref. [83] Description, Quality Analysis (five items each criterion classified as yes, no and partially: research objectives defined, sample description provided, pedagogical context described, experimental design is presented, use of pre/post-testing), Features used to stimulate SRL (journaling, graphics, questions, worked-examples, social data, feedback, training, pseudocode, prediction), Programming Achievement (Positive or Positive/Neutral), Regulation of learning Development (Positive or Positive/Neutral), SRL Strategies stimulated (Self-assessment; Goal-setting and Planning; Seeking information; Keeping records and monitoring; Reviewing notes, Reviewing tests, Organization and Transformation).

4. Discussion

This umbrella review had as its first objective to review existing reviews on introduction to programming (CS1) in the context of higher education. ACM, IEEE, SCOPUS and

WoS databases and the search strategy presented in Section 2.2 were used. After removing the duplicate records, the inclusion and exclusion criteria were used to discard articles: first, by type, then by abtract and, finally, by the full text. Our final records database consisted of 21 articles, as can be shown in Study selection flow diagram (PRISMA diagram adapted). Four research questions were stipulated and we now answer these questions:

RQ1: What is the bibliometric information on the CS1 reviews, in the context of higher education (namely annual evolution, number and information on authors, types and sources of information, countries of affiliation, languages, keywords and most cited articles)?

The year with the most publications is 2019, with 6 publications, 15 articles are conference proceedings, ACM International Conference Proceeding Series is the most frequent publication source (*n* = 3), 18 articles are written in English, three is the most frequent number of authors per publication, eight out of 61 authors wrote two articles (three with affiliation from Colombia, three from Portugal, one from Ireland and one from New Zealand), Brazil is the most frequent affiliation country, Systematic review is the most frequent keyword. frequent and there are three articles cited in SCOPUS 96 or 95 times.

RQ2: What is the purpose of the reviews and what research questions do they set?

There are four different types of purpose: general, specific, student group and teacher directed. The most frequent propose is specific with nine papers, Blended learning, collaborative work, computational thinking, Collaborative learning with computer support, Intelligent programming tutors, MIT App Inventor, Robotics, Serious programming games and Self-regulation of learning. It was not possible to find a model for the research questions as they are so diverse. For this reason, we chose to list in attachments all research questions for all articles that had research questions and suggesting research questions for two articles that did not provide this methodology.

RQ3: What were the methods used to find the studies each review was based on (namely search strategy, databases used, eligibility criteria)?

ACM Library is the most used data base (*n* = 19), followed by the IEEE Digital library (*n* = 18). The search strategies is divided into two parts: one related to CS1 and another related to the specific objective of the literature review. The latter does not exist for reviews that have a general purpose. There are 45 different keywords to search for CS1 related articles, with CS1 (9 times). Inclusion criteria are varied and listed in Appendix B.

RQ4: What were the results found (number of records and which tables or divisions were made to catalog the articles)?

We chose to list (Section 3.4) by each of the four different types of purpose: general, specific, student group and teacher-directed and by reference, which divisions were elaborated by the review authors.

We found that there is an increase in the number of articles and literature reviews on the fundamentals of programming in higher education. We feel the need to group these studies together—to serve as tools for future studies, as well as a clear identification of what has been published. When analysing the methodology used in the literature—our sample—we found that there are very different perspectives, mainly because half of the studies have a specific purpose. Thus, the authors use a generic perspective (CS1, introduction to programming . . .) accompanied by a specific perspective (people with disabilities or a specific application to be used in the teaching-learning process, for example). Another curious finding was that CS1 is still the most used acronym for this type of curricular unit, although it was used for the first time in 1978 and later many other curricular recommendation documents have been published with other names for these introductory courses. The proposed framework—general, specific, group of students and directed by the teacher—is expected to be a good starting point for those who intend to invest in research in this area. In addition, it is important to investigate in this area—to improve the teaching-learning process and to understand how it can be improved more and more.

There are always, in each study, limitations of the evidence included in the review. Studies are always being published, so a study like this needs constant updating. We think a lot about methodologie adopted and we are not always sure they are the best, namely in relation to research strategies and the use of other keywords, namely in other languages, to find a more comprehensive search.

As future work, it was realized that it would be interesting to create a taxonomy that would serve as a model for literature reviews in order to facilitate the work of authors and those who read these reviews and then use this taxonomy for introductory programming articles. This article can be an excellent tool for researchers in the area, finding the different approaches to the subject and different perspectives, as well as for those who intend to carry out a systematic review because the framework of different methodologies used by other authors in high quality works is here.

Funding: This work was supported by the FCT—Fundação para a Ciência e a Tecnologia, I.P. [Project UIDB/05105/2020.

Data Availability Statement: The data presented in this study, are available on request from the corresponding author.

Conflicts of Interest: The author declares no conflict of interest.

Appendix A

Table A1. Reference, Purpose, Title and Research question.

Reference	Purpose, Title and Research Questions
Becker et al., 2019 [88]	General 50 years of CS1 at SIGCSE: A review of the evolution of introductory programming education esearch RQ1 Identify the important topics in introductory programming education research, including their trends, over the first 50 years of the SIGCSE Technical Symposium RQ2 Situate the introductory programming research presented at the SIGCSE Technical Symposium in the context of the wider literature
Alammary, 2019 [75]	Blended learning Blended learning models for introductory programming courses: A systematic review RQ1 What are the different blended learning models that have been applied in introductory programming courses? RQ2 How effective is blended learning in improving the learning experience of novice programmers? RQ3 Which model is the most appropriate for introductory programming courses?
Medeiros et al., 2019 [89]	General A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education RQ1 What previous skills and background knowledge are key for a novice student to learn programming? RQ2 What challenges do novice students encounter in learning how to program? RQ3 What challenges do teachers encounter in teaching introductory programming?
Scaico et al., 2018 [85]	Novices An Initial Analysis of the Research on Interest and Introductory Programming: A Systematic Review of this Literature RQ1 In this literature, what motivational theories are the studies based on? RQ2 What have researchers studied most? RQ3 How are studies methodologically designed?

Table A1. *Cont.*

Reference	Purpose, Title and Research Questions
Luxton-Reilly et al., 2018 [74]	General Introductory programming: A systematic literature review RQ1 What aspects of introductory programming have been the focus of the literature? RQ2 What developments have been reported in introductory programming education between 2003 and 2017? RQ3 What evidence has been reported when addressing different aspects of introductory programming?
Vihavainen et al., 2014 [73]	Teaching approaches A systematic review of approaches for teaching introductory programming and their influence on success. RQ1 How do teaching interventions reported in the literature increase students' success in CS1? RQ2 What practices do the successful teaching interventions comprise of? RQ3 Do so called best practices, or practices that are significantly better than others exist?
Shahid et al., 2019 [82]	Serious programming games A Review of Gamification for Learning Programming Fundamental RQ1 Which programming concepts are usually overlooked by the researchers? RQ2 What Game Elements should be included that ensures active participation of students? RQ3 What evaluation methods are used to measure the effectiveness of gamification?
Agbo et al., 2019 [77]	Computational thinking A systematic review of computational thinking approach for programming education in higher education institutions RQ1 How has the use of CT been explored for teaching programming in HEIs? RQ2 What are the ways CT approach has been used to teach programming education in HEIs? RQ3 How has the use of CT approach for teaching programming impacted students learning experience?
Revelo-Sánchez et al., 2018 [76]	Collaborative Work Collaborative work as a didactic strategy for teaching/learning programming: a systematic literature review RQ1 What are the learning approaches and types of contributions from studies on collaborative work as a didactic strategy for teaching/learning programming? RQ2 What are the most common collaborative techniques that have been defined in the scientific literature? RQ3 What are the purposes of the most common collaborative techniques used in the teaching/learning of the programming? RQ4 What do the authors think about collaborative work as a didactic strategy for teaching/learning programming?
Silva et al., 2020, [78]	Computer supported collaborative learning. Computer-supported collaborative learning in programming education: A systematic literature review RQ1 what collaborative resources were used? RQ2 which resources are most effective? RQ3 what has been measured in those experiments? RQ4 how collaboration is structured? RQ5 how collaborative work was measured?
Jiménez-Toledo et al., 2019 [90]	General Considerations for the Teaching-Learning Processes of Introductory Programming Courses: A Systematic Literature Review RQ1 What learning problems have been reported in programming fundamentals students? RQ2 What drawbacks are reported in teaching programming fundamentals? RQ3 What technological teaching-learning tools are used in a first computer programming course? RQ4 What teaching-learning strategies are applied in a first computer programming course? RQ5 Are there any methodological considerations to face a first course in computer programming? RQ6 What trends does computer programming have that can change current teaching/learning processes?

Table A1. *Cont.*

Reference	Purpose, Title and Research Questions
Lukkarinen et al., 2021 [91]	General Event-driven Programming in Programming Education: A Mapping Review RQ1 Who has written about teaching and learning event-driven programming (EDP), when and where? RQ2 Which research methods have been used to study teaching and learning EDP? RQ3 In which educational contexts has EDP been discussed? RQ4 What kind of pedagogical approaches have been reported to support learning EDP? RQ5 What kind of software tools have been developed to support learning EDP? RQ6 What empirical results of learning EDP have been reported?
Santos et al., 2020 [92]	Teaching methodologies Innovative approaches in teaching programming: A systematic literature review RQ1 What is the teaching context (educational stage)? RQ2 What are the innovative approaches of the studies? RQ3 What are the main results found from these approaches? RQ4 What are the main challenges or difficulties found in general way?
Crow et al., 2018 [79]	Intelligent programming tutors Intelligent Tutoring Systems for Programming Education: A Systematic Review RQ1 What programming languages are being taught with IPTs? RQ2 What types of supplementary features are used in IPTs and how have they been implemented into tutoring tools? RQ3 What parts of the tutoring process are adaptive within IPTs?
Slim et al., 2018 [86]	Novice programmer and online methods Online Tools to Support Novice Programming: A Systematic Review RQ1 What are the challenges faced by a novice programmer. RQ2 What are the online methods that are popular for helping novice students.
Silva et al., 2021 [83]	Self-regulated Learning Regulation of Learning Interventions in Programming Education: A Systematic Literature Review and Guideline Proposition RQ1 What were the features used to foster regulation of learning in introductory programming? RQ2 Which aspects of the regulation of learning were explored in the interventions? RQ3 What results were achieved by the proposed interventions?
Qian et al., 2017 [72]	Students' difficulties Students' misconceptions and other difficulties in introductory programming: A literature review RQ1 What is the relevant literature on general definitions of misconceptions and studies on student and other difficulties in introductory programming? RQ2 What factors contribute to the difficulties? RQ3 What are the best strategies and tools to deal with difficulties?
Major et al., 2011 [81]	Robotics Systematic literature review: Teaching novices programming using robots. RQ1 What computer languages are being taught in introductory programming courses that make use of robots as teaching tools? RQ2 Are the robots that are being used simulated or physical? RQ3 What are the characteristics (i.e., what is the age, level of education etc.) of the novices being taught? RQ4 What types of studies are being performed by researchers that investigate the teaching of introductory programming concepts using robots? RQ5 What is the scale (e.g., number of participants) of studies that are being performed by researchers? RQ6 Do collected studies suggest that using robotics to teach introductory programming is effective?
Al-Ratta et al., 2013 [84]	Blinds Teaching programming for blinds: A review RQ1 Which are the research related to teaching programming to the blind? RQ2 What are the methods applied?

Table A1. *Cont.*

Reference	Purpose, Title and Research Questions
Vinueza-Morales et al., 2020 [80]	MIT app inventor Teaching programming with MIT app inventor: A literature review RQ1 What literature uses MIT App Inventor? RQ2 What is the acceptance of the academic community in MIT App Inventor as an Effective Tool for student motivation and performance?
Nolan et al., 2016 [87]	Anxiety The role of anxiety when learning to program: A Systematic review of the literature. RQ1 Is there a relationship between learning to program (language, syntax, compilation etc.) and anxiety? RQ2 Is there a relationship between mathematical anxiety and learning to program? RQ3 Does computer usage cause anxiety when learning to program? RQ4 Does test anxiety affect programming and more broadly Computer Science students?

Appendix B

Table A2. Reference, Data Bases (DBs), Search (first and second), Inclusion and Exclusion criteria.

Reference	DBs	Search 1	Search 2	Inclusion	Exclusion
Becker et al., 2019 [88]	ACM	CS1 OR “CS 1” OR “introductory programming” OR “introduction to programming” OR “novice programming” OR “novice programmers”		I1. Papers presented at the SIGCSE Technical Symposium	E1. Papers less than three pages in length. E2. Papers not focusing on first-year university-level introductory programming
Alammary, 2019 [75]	ACM, CD, IEEE, ERIC, ProQuest, SAGE, ScienceDirect, SCOPUS and T&F.	“programming”	(“blended learning” OR “blended course” OR “blended classroom” OR “blended class” OR “hybrid learning” OR “hybrid course” OR “hybrid classroom” OR “hybrid class”)	I1. Context of higher education. I2. Introductory programming course. I3. The course presented in the article should have both face-to-face and online components. I4. included studies were limited to the English language.	
Medeiros et al., 2019 [89]	ACM, ERIC, IEEE, ScienceDirect and Springer	(“learning programming” OR “teaching programming”) AND (“novice programmers” OR “CS1”)		I1. written in English, I2. published in conferences or journals, or as book chapters. I3. between 2010 and 2016, I4. teaching and learning introductory programming I5. higher education context.	E1. Did not address the research questions; E2. Were too short, such as workshop papers; E3. Were published in local conferences; E4. Were written by the same research group with the same data (in which case only the most recent was kept).
Scaico et al., 2018 [85]	ACM, ERIC, IEEE, SemanticS and Springer	(“learn” OR “learning” OR “teach” OR “teaching”) AND (“introductory programming” OR “CS0” OR “CS1”) AND (“programming” OR “code” OR “coding”) AND (“undergraduate” OR “higher education” OR “college” OR “university”) AND (“computer” OR “computing”)	(“motivation” OR “interest” OR “engagement”)	I1. published between 2010 and 2017. I2. novices’ interest in learning to code. I3. domain of computer science;	E1. short papers (less than 5 pages). E2. Publications not including a description of its methodological design were excluded; E3. Literature reviews and meta-analysis work; E4. not targeting introductory courses of programming
Luxton-Reilly et al., 2018 [74]	ACM, IEEE, ScienceDirect, SCOPUS and Springer	“introductory programming” OR “introduction to programming” OR “novice programming” OR “novice programmers” OR “CS1” OR “CS 1” OR “learn programming” OR “learning to program” OR “teach programming”			E1. Less than four pages long (such as posters). E2. work in progress

Table A2. *Cont.*

Reference	DBs	Search 1	Search 2	Inclusion	Exclusion
Vihavainen et al., 2014 [73]	ACM and IEEE.	(Programming OR "Programming Course" OR "Introductory Programming" OR CS1)	(Improve OR Increase OR Decrease OR Lower OR Reduce) AND (Retention OR Attrition OR Pass OR Fail OR Success)	I1. Only conference proceedings and selected periodicals: Computer Education Transactions, SIGCSE, ITiCSE, ICER, SIGITE and CALT.	E1. articles that did not describe a replicable teaching intervention. E2. intervention that had overall been attempted at least twice. E3. articles that did not discuss CS1 or introductory programming courses. E4. articles that did not include pass-rates before or after the teaching intervention. E5. articles that did not include the number of students before or after the teaching intervention. E6. articles that described previously reported results
Shahid et al., 2019 [82]	ACM, IEEE, ScienceDirect and WoS.	"computer programming"	"Gamification" AND ("Digital learning" OR "Digital Game") OR ("Serious Games" AND "Effectiveness")	I1. between 2015–2019 I2. content related to Gamification and DGBL. I3. Articles written in English. I4. New games were developed, or new framework proposed. I5. non-empirical papers contain reviews, surveys and experiments.	
Agbo et al., 2019 [77]	ACM, IEEE, ScienceDirect and Springer	(programming OR computer science)	(computational thinking AND problem solving)	I1. Articles that focus on computational thinking, problem-solving and programming education I2. Articles that are published in a peer-reviewed journals or conferences. I3. Articles that either presented a concrete programming artefact, evaluated a solution for programming or design a study to explore CT	E1. Articles that do not focus on the keywords E2. not written in English Language E3. Materials that are not peer reviewed E4. Theoretical and conceptual studies
Revelo-Sánchez et al., 2018 [76]	ACM, IEEE, ScienceDirect and SCOPUS	("computer programming" OR "programming course" OR CS1) AND (teaching OR learning OR education)	AND ("collaborative learning" OR "cooperative learning" OR cscl OR csgf OR "social learning" OR "group learning" OR "team learning")	I1. years 2013 to 2017 I2. magazines or conferences, I3. language English or Spanish, I4. related to computer science and engineering	Five quality criteria E1. relevance of content to answer questions of interest to the review, E2. clarity in the purpose of the investigation, E3. adequate description of the context in which the investigation was carried out, E4. clarity E5. rigor of the research's methodological design
Silva et al., 2020, [78]	GS	("learning programming" OR "programming learning" OR "teaching programming" OR "programming teaching" OR "novice programmer")	cscl	I1. 2015 and above I2. peer-reviewed conferences/journals and Ms.C dissertations/Ph.D. theses I3. pedagogical context must be related to the introductory programming I4. students must have collaborated through technological resources (CSCL) I5. be a quasi-experimental, experimental, or observational study	

Table A2. *Cont.*

Reference	DBs	Search 1	Search 2	Inclusion	Exclusion
Jiménez-Toledo et al., 2019 [90]	IEEE, Redalyc, Springer and WoS	(“Teaching-learning of computer programming” OR “Teaching in computer programming” OR “Learning in computer programming” OR “Programming teaching environments” OR “A first computer programming course” OR “Teaching-learning strategies in computer programming”) AND (education Or Software Or Computer’s science)		I1. year of publication ≥2012 I2. teaching process studies I3. Programming fundamentals to undergraduate students (or synonyms)	
Lukkarinen et al., 2021 [91]	ACM, ERIC, IEEE, GS, ScienceDirect, Springer, T&F and Wiley		(“event-driven programming” OR “event-based programming” OR “event-oriented programming” OR “events-first programming” OR “event programming”)	I1. publication accessible for free I2. in journals or conference proceedings I3. are written in English, I4 not an introduction to some primary content (e.g., editorial to a journal special issue), I5. concerned with teaching and learning event-driven programming, I6. rated high enough in the publication rating system JUFO5 used in Finland.	E1. studies that do not have a corresponding bibliographic citation, E2. document contains only the search terms or synonyms, E3. experiences only with the first courses in computer programming and E4. documents are not available for download
Santos et al., 2020 [92]	ACM, IEEE and ScienceDirect	(“teaching” OR “learning”) AND (“programming” OR “programming languages”)	(“innovative” OR “innovation”) AND (“case study” OR “case studies” OR “lessons learned” OR “experience report”)		E1. Exclude artifacts according to the analysis of titles and abstracts; E2. on-English articles; E3. Articles with paid content; E4. Duplicate, repeated articles; E5. Articles not available for download or viewing quality criteria used: Relevance, Completeness and clarity of contents, study answered the questions.
Crow et al., 2018 [79]	ACM and IEEE	(“computer science education” OR “software engineering education” OR “introductory computer” OR “introductory programming” OR “teach* programming” OR “learn* programming” OR “novice programming” OR “coding education” OR “cs1” OR “introductory computer science”)	(“intelligent tutor*” OR “adaptive tutor*” OR “cognitive tutor*” OR “smart tutor”)	I1. Relates to a completed IPT that has been used by students learning programming. I2. Relates to a programming language that has conditional and iterative control structures. I3. Tutors scripted programming, i.e., the student has to type lines of code as part of the tasks. I4. Meets a working definition of being intelligent by adapting some part of the tutoring process based on multiple metrics and/or an algorithm.	E1. Literature that does not report an implemented IPT. E2. Tools that are only partially prototyped. E3. IPTs that use non-typed/non-scripted programming languages; i.e., block-based languages like scratch. E4. Tools that tutor some other part of computer science like data-structures or SQL.

Table A2. *Cont.*

Reference	DBs	Search 1	Search 2	Inclusion	Exclusion
Slim et al., 2018 [86]	ACM and IEEE	(”novice programming” OR ”CS1” OR ”introductory programming”)	(”threshold concepts”OR ”difficulties” OR ”challenges”)	I1. 1998 to 2018	
Silva et al., 2021 [83]	ACM, ERIC, IEEE and ScienceDirect	(“learning programming” OR” programming learning” OR”teaching programming” OR” programming teaching” OR” novice programmer” OR” programming education”)	(self-regulat* OR SRL OR metacognit* OR co-regulat* OR SSRL OR “regulation of learning”)	I1. pedagogical context is introductory programming I2. studies have mentioned theoretical constructs of SRL I3. must have included students in a quasi-experimental or experimental investigation; I4. there must be a control-group I5. proposed an intervention.	E1. the intervention was described in sufficient detail to allow its replication
Qian et al., 2017 [72]	ACM and ERIC	“programming”	(“misconception” OR “difficulty” OR “error”)		E1. non-introductory-level CS courses.
Major et al., 2011 [81]	ACM, AustralianEI, BritishEI, CiteSeerX, EBSCOhost, IEEE, ERIC, KeeleU and WoS	programming AND (novice OR beginner OR introductory OR teaching OR learning OR CS1 or “first time”).	(robots OR robotics)	I1. Publications were only included that reported on the use of robotics in teaching introductory programming to students who were studying a specific Computing or IT-related course. I2. Papers that involve an empirical study or have a “lessons learned” (experience report) element were included. I3. Where several papers reported the same study only the most recent paper was included. I4. Date of publication did not act as a barrier for inclusion. I5. Grey literature (such as technical papers or government reports) was accepted if relevant.	E1. main focus was not on the use of robotics in teaching E2. Computing or IT students introductory programming but on the use of robots in general education courses, E3. as part of a non-IT or Computing related course syllabus E4. to teach rudimentary programming concepts to very young children. E5. Papers that just propose an approach or describe the use of robots to teach introductory programming (with no “lessons learnt” component) were excluded. E6. Papers and reports were excluded when only the abstract but not the full text was available. E7. Publications were excluded if they are not written in English. E8. Letters, editorials and position papers were all excluded.
Al-Ratta et al., 2013 [84]	ACM, IEEE, GS, Springer and WoS		“Teaching programming for blinds” OR (“Programming” and “visually impaired”) OR “blind programmers”		
Vinueza-Morales et al., 2020 [80]	ACM, BASE, Dialnet, DOAJ, ERIC, IEEE, GS, MicrosoftAcademic, Redalyc, SciELO, ScienceDirect and Springer		“App Inventor”	I1. Year $\geq$ 2010–2020 I2. Articles, Conference paper, thesis programming teaching	E1. did not correspond specifically to programming teaching

Table A2. *Cont.*

Reference	DBs	Search 1	Search 2	Inclusion	Exclusion
Nolan et al., 2016 [87]	ACM, ERIC, GS, IEEE and ScienceDirect	Programming	Anxiety	I1. directly answered one or more research question. I2. focused on anxiety in programming. focused on anxiety I3. related to either mathematics anxiety, computer anxiety or test anxiety.	E1. were in the form of a book or grey literature. E2. related to primary or secondary school learning

References

1. Sobral, S.R. The First Programming Language and Freshman Year in Computer Science: Characterization and Tips for Better Decision Making. In *Advances in Intelligent Systems and Computing*; Springer: Cham, Switzerland, 2020; Volume 1161, pp. 162–174.
2. Kinnunen, P.; Malmi, L. Why students drop out CS1 course? In Proceedings of the ICER 2006—2nd International Workshop on Computing Education Research, Canterbury, UK, 9–10 September 2006.
3. Horton, D.; Craig, M. Drop, fail, pass, continue: Persistence in CS1 and beyond in traditional and inverted delivery. In Proceedings of the 46th ACM Technical Symposium on Computer Science Education, Kansas City, MI, USA, 4–7 March 2015.
4. Porter, L.; Simon, P. Retaining nearly one-third more majors with a trio of instructional best practices in CS1. In Proceedings of the 44th ACM Technical Symposium on Computer Science Education, Denver, CO, USA, 6–9 March 2013.
5. Yadin, A. Reducing the dropout rate in an introductory programming course. *ACM Inroads* **2011**, *2*, 71–76. [CrossRef]
6. Reges, S. Back to basics in CS1 and CS2. *ACM SIGCSE Bull.* **2006**, *38*, 293–297. [CrossRef]
7. Sobral, S. Bloom's taxonomy to improve teaching-learning in introduction to programming. *Int. J. Inf. Educ. Technol.* **2021**, *11*, 148–153.
8. Schulte, C.; Bennedsen, J. What do teachers teach in introductory programming? In Proceedings of the Second International Workshop on Computing Education Research, Canterbury, UK, 9–10 September 2006.
9. Austing, R.H.; Barnes, B.H.; Bonnette, D.T.; Engel, G.L.; Stokes, G. Curriculum '78: Recommendations for the undergraduate program in computer science—A report of the ACM curriculum committee on computer science. *Commun. ACM* **1979**, *22*, 147–166. [CrossRef]
10. Sobral, S.R. CS1 and CS2 Curriculum Recommendations: Learning from the Past to Try not to Rediscover the Wheel Again. In *Advances in Intelligent Systems and Computing*; Springer: Cham, Switzerland, 2020; Volume 1161, pp. 182–191.
11. Sobral, S.R.; Moreira, F. A portrait of adopted programming languages of Portuguese Higher Education Institutions. In Proceedings of the 2021 IEEE Global Engineering Education Conference (EDUCON), Vienna, Austria, 21–23 April 2021; pp. 1189–1194.
12. Joint Task Force on Computing Curricula; Association for Computing Machinery (ACM); IEEE Computer Society. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*; Association for Computing Machinery: New York, NY, USA, 2013.
13. Joint Task Group on Computer Engineering Curricula. *CE2016: Computer Engineering Curricula*; ACM: New York, NY, USA, 2016.
14. National Research Council. *A Framework for K–12 Science Education: Practices, Crosscutting Concepts, and Core Ideas*; The National Academies Press: Washington, DC, USA, 2012.
15. Alphonce, C.; Ventura, P. Object orientation in CS1-CS2 by design. *ACM SIGCSE Bull.* **2002**, *34*, 70–74. [CrossRef]
16. Sobral, S. CS1: C, Java or Python? Tips for a Conscious Choice. In Proceedings of the 12th Annual International Conference of Education, Research and Innovation, Seville, Spain, 11–13 November 2019; pp. 2512–2519.
17. Sobral, S.R. The Old Question: Which Programming Language Should We Choose to Teach to Program? In *Advances in Intelligent Systems and Computing*; Springer: Cham, Switzerland, 2021; Volume 1352, pp. 351–364.
18. Felix, C.; Sobral, S.R. Predicting students' performance using survey data. In Proceedings of the 2020 IEEE Global Engineering Education Conference (EDUCON), Porto, Portugal, 27–30 April 2020; pp. 1017–1023.
19. Ventura, P.R., Jr. Identifying predictors of success for an objects-first CS1. *Comput. Sci. Educ.* **2005**, *15*, 223–243. [CrossRef]
20. Costa, E.; Fonseca, B.; Santana, M.A.; De Araújo, F.F.; Rego, J. Evaluating the effectiveness of educational data mining techniques for early prediction of students' academic failure in introductory programming courses. *Comput. Hum. Behav.* **2017**, *73*, 247–256. [CrossRef]
21. Shell, D.F.; Soh, L.-K.; Flanigan, A.E.; Peteranetz, M. Students' Initial Course Motivation and Their Achievement and Retention in College CS1 Courses. In Proceedings of the 47th ACM Technical Symposium on Computing Science Education, Memphis, TN, USA, 2–5 March 2016; Association for Computing Machinery (ACM): New York, NY, USA, 2016; pp. 639–644.
22. Santana, B.L.; Bittencourt, R.A. Increasing Motivation of CS1 Non-Majors through an Approach Contextualized by Games and Media. In Proceedings of the 2018 IEEE Frontiers in Education Conference (FIE), San Jose, CA, USA, 3–6 October 2018; pp. 1–9.
23. Porter, L.; Guzdial, M.; McDowell, C.; Simon, B. Success in introductory programming: What works?: How pair programming, peer instruction, and media computation have improved computer science education. *Commun. ACM* **2013**, *56*, 34–36. [CrossRef]
24. Sobral, S.R. Strategies on Teaching Introducing to Programming in Higher Education. In *Advances in Intelligent Systems and Computing*; Springer: Cham, Switzerland, 2021; Volume 1367, pp. 133–150.

25. Nagappan, N.; Williams, L.; Ferzli, M.; Wiebe, E.; Yang, K.; Miller, C.; Balik, S. Improving the CS1 experience with pair programming. *ACM SIGCSE Bull.* **2003**, *35*, 359–362. [CrossRef]
26. Sobral, S.R. Is Pair Programing in Higher Education a Good Strategy? *Int. J. Inf. Educ. Technol.* **2020**, *10*, 911–916. [CrossRef]
27. Sobral, S.R. Pair Programming and the Level of Knowledge in the Formation of Pairs. In *Advances in Intelligent Systems and Computing*; Springer: Cham, Switzerland, 2021; Volume 1367, pp. 212–221.
28. Amresh, A.; Carberry, A.; Femiani, J. Evaluating the effectiveness of flipped classrooms for teaching CS1. In Proceedings of the 2013 IEEE Frontiers in Education Conference (FIE), Oklahoma City, OK, USA, 23–26 October 2013; pp. 733–735.
29. Sobral, S.R. Flipped Classrooms for Introductory Computer Programming Courses. *Int. J. Inf. Educ. Technol.* **2021**, *11*, 178–183. [CrossRef]
30. Campbell, J.; Horton, D.; Craig, M.; Gries, P. *Evaluating an Inverted CS1*; Association for Computing Machinery (ACM): Atlanta, GA, USA, 2014; pp. 307–312.
31. Zingaro, D. Peer instruction contributes to self-efficacy in CS1. In Proceedings of the 45th ACM Technical Symposium on Computer Science Education, Atlanta, GA, USA, 5–8 March 2014; Association for Computing Machinery (ACM): New York, NY, USA, 2014; pp. 373–378.
32. Gilbert, M.; Weikle, D.A.B.; Mayfield, C.; Johnson, C. Fourth hour: A CS1 review session led by teaching assistants using peer instruction. *J. Comput. Sci. Coll.* **2021**, *36*, 45–54.
33. Beck, L.; Chizhik, A. Cooperative learning instructional methods for CS1: Design, implementation, and evaluation. *ACM Trans. Comput. Educ.* **2013**, *13*, 1–21. [CrossRef]
34. LeJeune, N. Critical components for successful collaborative learning in CS1. *J. Comput. Sci. Coll.* **2003**, *19*, 275–285.
35. Gonzalez, G. A systematic approach to active and cooperative learning in CS1 and its effects on CS2. In Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education, Houston, TX, USA, 3–5 March 2006.
36. Sobral, S.R. Two different experiments on teaching how to program with active learning methodologies: A critical analysis. In Proceedings of the 2020 15th Iberian Conference on Information Systems and Technologies (CISTI), Seville, Spain, 24–27 June 2020.
37. Sobral, S.R.; Remit, P. Project Based Learning with Peer Assessment in an Introductory Programming Course. *Int. J. Inf. Educ. Technol.* **2021**, *11*, 337–341. [CrossRef]
38. Fassbinder, A.G.D.O.; Botelho, T.G.; Martins, R.J.; Barbosa, E. Applying flipped classroom and problem-based learning in a CS1 course. In Proceedings of the 2015 IEEE Frontiers in Education Conference (FIE), El Paso, TX, USA, 21–24 October 2015; pp. 1–7.
39. Bayliss, J.; Strout, S. Games as a "flavor" of CS1. In Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education, Houston, TX, USA, 3–5 March 2006.
40. Barnes, T.; Richter, H.; Powell, E.; Chaffin, A.; Godwin, A. Game2Learn: Building CS1 learning games for retention. *ACM SIGCSE Bull.* **2007**, *39*, 121–125. [CrossRef]
41. Becker, B.W. Teaching CS1 with karel the robot in Java. *ACM SIGCSE Bull.* **2001**, *33*, 50–54. [CrossRef]
42. Summet, J.; Kumar, D.; O'Hara, K.; Walker, D.; Ni, L.; Blank, D.; Balch, T. Personalizing CS1 with robots. *ACM SIGCSE Bull.* **2009**, *41*, 433–437. [CrossRef]
43. Lin, P.-H.; Chen, S.-Y. Design and Evaluation of a Deep Learning Recommendation Based Augmented Reality System for Teaching Programming and Computational Thinking. *IEEE Access* **2020**, *8*, 45689–45699. [CrossRef]
44. Lee, M.J.; Ko, A.J. Comparing the Effectiveness of Online Learning Approaches on CS1 Learning Outcomes. In Proceedings of the Eleventh Annual International Conference on International Computing Education Research, Omaha, NE, USA, 9–13 July 2015.
45. Kortsarts, Y.; Akhuseyinoglu, K.; Barria-Pineda, J.; Brusilovsky, P. Integrating personalized online practice into an introductory programming course. *J. Comput. Sci. Coll.* **2020**, *35*, 264–266.
46. Toledo, J.J.; Collazos, C.; Ortega, M. Discovery Model Based on Analogies for Teaching Computer Programming. *Mathematics* **2021**, *9*, 1354. [CrossRef]
47. Singh, R.; Gulwani, S.; Solar-Lezama, A. Automated feedback generation for introductory programming assignments. *ACM SIGPLAN Not.* **2013**, *48*, 15–26. [CrossRef]
48. Ott, C.; Robins, A.; Shephard, K. Translating Principles of Effective Feedback for Students into the CS1 Context. *ACM Trans. Comput. Educ.* **2016**, *16*, 1–27. [CrossRef]
49. Özyurt, Ö.; Özyurt, H. Using Facebook to enhance learning experiences of students in computer programming at Introduction to Programming and Algorithm course. *Comput. Appl. Eng. Educ.* **2016**, *24*, 546–554. [CrossRef]
50. Mir, S.; Lluec, G. Introduction to Programming Using Mobile Phones and MIT App Inventor. *IEEE Rev. Iberoam. Tecnol. Aprendiz.* **2020**, *15*, 192–201.
51. Kurniawan, O.; Lee, N.T.S.; Poskitt, C. Securing Bring-Your-Own-Device (BYOD) Programming Exams. In Proceedings of the 51st ACM Technical Symposium on Computer Science Education, Portland, OR, USA, 11–14 March 2020.
52. Smith, A.C.; Francioni, J.M.; Matzek, S.D. A Java programming tool for students with visual disabilities. In Proceedings of the Fourth International ACM Conference on Assistive Technologies, Arlington, VA, USA, 13–15 November 2000.
53. Rich, L.; Perry, H.; Guzdial, M. A CS1 course designed to address interests of women. *ACM SIGCSE Bull.* **2004**, *36*, 190–194. [CrossRef]
54. Rubio, M.A.; Romero-Zaliz, R.; Mañoso, C.; de Madrid, A.P. Closing the gender gap in an introductory programming course. *Comput. Educ.* **2015**, *82*, 409–420. [CrossRef]
55. Page, M.J.; McKenzie, J.E.; Bossuyt, P.M.; Boutron, I.; Hoffmann, T.C.; Mulrow, C.D.; Shamseer, L.; Tetzlaff, J.M.; Akl, E.A.; Brennan, S.E.; et al. The PRISMA 2020 statement: An updated guideline for reporting systematic reviews. *BMJ* **2021**, *372*, n71. [CrossRef] [PubMed]

56. Kitchenham, B.; Charters, S. *Guidelines for Performing Systematic Literature Reviews in Software Engineering*; EBSE Technical Report: Keele, UK, 2007.
57. Pollock, M.; Fernandes, R.; Becker, L.; Pieper, D.; Hartling, L. Chapter V: Overviews of Reviews. In *Cochrane Handbook for Systematic Reviews of Interventions Version 6.2*; Cochrane: London, UK, 2021.
58. Aromataris, E.; Fernandez, R.; Godfrey, C.M.; Holly, C.; Khalil, H.; Tungpunkom, P. Summarizing systematic reviews. *Int. J. Evid. Based Healthc.* **2015**, *13*, 132–140. [CrossRef]
59. Fusar-Poli, P.; Radua, J. Ten simple rules for conducting umbrella reviews. *Evid. Based Ment. Health* **2018**, *21*, 95–100. [CrossRef]
60. Zhu, J.; Liu, W. A tale of two databases: The use of Web of Science and Scopus in academic papers. *Scientometrics* **2020**, *123*, 321–335. [CrossRef]
61. Pranckutė, R. Web of Science (WoS) and Scopus: The Titans of Bibliographic Information in Today's Academic World. *Publications* **2021**, *9*, 12. [CrossRef]
62. Sobral, S.; Jesus-Silva, N.; Cardoso, A.; Moreira, F. EU27 Higher Education Institutions and COVID-19, Year 2020. *Int. J. Environ. Res. Public Health* **2021**, *18*, 5963. [CrossRef] [PubMed]
63. Martín-Martín, A.; Orduna-Malea, E.; Thelwall, M.; López-Cózar, E.D. Google Scholar, Web of Science, and Scopus: A systematic comparison of citations in 252 subject categories. *J. Inf.* **2018**, *12*, 1160–1177. [CrossRef]
64. Pritchard, A. Statistical Bibliography or Bibliometrics. *J. Doc.* **1969**, *25*, 348–349.
65. Youngblood, M.; Lahti, D. A bibliometric analysis of the interdisciplinary field of cultural evolution. *Palgrave Commun.* **2018**, *4*, 120. [CrossRef]
66. Li, W.; Zhao, Y. Bibliometric analysis of global environmental assessment research in a 20-year period. *Environ. Impact Assess. Rev.* **2015**, *50*, 158–166. [CrossRef]
67. Donthu, N.; Kumar, S.; Mukherjee, D.; Pandey, N.; Lim, W.M. How to conduct a bibliometric analysis: An overview and guidelines. *J. Bus. Res.* **2021**, *133*, 285–296. [CrossRef]
68. Sobral, S.R.; Sobral, M. Computerized cognitive stimulation for people with dementia or with mild cognitive impairment: A bibliometric review. *Dement. Neuropsychol.* **2021**, *15*, 28–40. [CrossRef] [PubMed]
69. Kitchenham, B.; Pearl Brereton, O.; Budgen, D.; Turner, M.; Bailey, J.; Linkman, S. Systematic literature reviews in software engineering—A systematic literature review. *Inf. Softw. Technol.* **2009**, *51*, 7–15. [CrossRef]
70. Baptista, A.; Martins, J.; Gonçalves, R.; Branco, F.; Rocha, T. Web accessibility challenges and perspectives: A systematic literature review. In Proceedings of the 2016 11th Iberian Conference on Information Systems and Technologies (CISTI), Gran Canaria, Spain, 15–18 June 2016; pp. 1–6.
71. Zygomatic. Word Cloud. Zygomatic, 2021. Available online: https://www.wordclouds.com/ (accessed on 1 May 2021).
72. Qian, Y.; Lehman, J. Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Trans. Comput. Educ.* **2017**, *18*, 1–24. [CrossRef]
73. Vihavainen, A.; Airaksinen, J.; Watson, C. A systematic review of approaches for teaching introductory programming and their influence on success. In Proceedings of the Tenth Annual Conference on International Computing Education Research, Glasgow Scotland, UK, 11–13 August 2014.
74. Luxton-Reilly, A.; Simon; Albluwi, I.; Becker, B.; Giannakos, M.N.; Kumar, A.N.; Ott, L.; Paterson, J.; Scott, M.J.; Sheard, J.; et al. Introductory programming: A systematic literature review. In Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education, Larnaca, Cyprus, 2–4 July 2018; pp. 55–106.
75. AlAmmary, A. Blended learning models for introductory programming courses: A systematic review. *PLoS ONE* **2019**, *14*, e0221765. [CrossRef]
76. Revelo-Sánchez, O.; Collazos-Ordóñez, C.A.; Jiménez-Toledo, J.A. Collaborative work as a didactic strategy for teaching/learning programming: A systematic literature review. *TecnoLógicas* **2018**, *21*, 115–134. [CrossRef]
77. Agbo, F.J.; Oyelere, S.S.; Suhonen, J.; Adewumi, S. A Systematic Review of Computational Thinking Approach for Programming Education in Higher Education Institutions. In Proceedings of the 19th Koli Calling International Conference on Computing Education Research, Koli, Finland, 21–24 November 2019; p. 12.
78. Silva, L.; Mendes, A.; Gomes, A. Computer-supported Collaborative Learning in Programming Education: A Systematic Literature Review. In Proceedings of the 2020 IEEE Global Engineering Education Conference (EDUCON), Porto, Portugal, 27–30 April 2020; pp. 1086–1095.
79. Crow, T.; Luxton-Reilly, A.; Wuensche, B. Intelligent Tutoring Systems for Programming Education: A Systematic Review. In Proceedings of the 20th Australasian Computing Education Conference, Brisbane, Australia, 30 January–2 February 2018.
80. Vinueza-Morales, M.; Rodas-Silva, J.; Chacón-Luna, A.; Mantilla, H. Teaching programming with MIT app inventor: A literature review [Enseñanza de programación mediante MIT App Inventor: Una revisión de literatura]. In Proceedings of the 18th LACCEI International Multi-Conference for Engineering, Education and Technology, Virtual. 27–31 July 2020; Available online: http://dx.doi.org/10.18687/LACCEI2020.1.1.49 (accessed on 19 May 2021).
81. Major, L.; Kyriacou, T.; Brereton, O. Systematic literature review: Teaching novices programming using robots. *IET Softw.* **2012**, *6*, 502. [CrossRef]
82. Shahid, M.; Wajid, A.; Haq, K.U.; Saleem, I.; Shujja, A.H. A Review of Gamification for Learning Programming Fundamental. In Proceedings of the 2019 International Conference on Innovative Computing (ICIC), Lahore, Pakistan, 1–2 November 2019; pp. 1–8.

83. Silva, L.; Mendes, A.; Gomes, A.; Cavalcanti-De-Macedo, G. Regulation of Learning Interventions in Programming Education: A Systematic Literature Review and Guideline Proposition. In Proceedings of the 52nd ACM Technical Symposium on Computer Science Education, Virtual Event, USA, 13–20 March 2021; Available online: https://doi.org/10.1145/3408877.3432363 (accessed on 19 May 2021).
84. Al-Ratta, N.M.; Al-Khalifa, H.S. Teaching programming for blinds: A review. In Proceedings of the Fourth International Conference on Information and Communication Technology and Accessibility (ICTA), Hammamet, Tunisia, 24–26 October 2013; pp. 1–5.
85. Scaico, P.D.; Scaico, A.; De Queiroz, R.J.B. An Initial Analysis of the Research on Interest and Introductory Programming: A Systematic Review of this Literature. In Proceedings of the 2018 IEEE Frontiers in Education Conference (FIE), San Jose, CA, USA, 3–6 October 2018; pp. 1–9.
86. Sim, T.Y.; Lau, S.L. Online Tools to Support Novice Programming: A Systematic Review. In Proceedings of the 2018 IEEE Conference on e-Learning, e-Management and e-Services (IC3e), Langkawi, Malaysia, 21–22 November 2018; pp. 91–96.
87. Nolan, K.; Bergin, S. The role of anxiety when learning to program: A Systematic review of the literature. In Proceedings of the 16th Koli Calling International Conference on Computing Education Research, Koli, Finland, 24–27 November 2016.
88. Becker, B.; Quille, K. 50 years of CS1 at SIGCSE: A review of the evolution of introductory programming education research. In Proceedings of the 50th ACM Technical Symposium on Computer Science Education, Minneapolis, MN, USA, 27 February–2 March 2019.
89. Medeiros, R.P.; Ramalho, G.L.; Falcao, T.P. A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education. *IEEE Trans. Educ.* **2018**, *62*, 77–90. [CrossRef]
90. Jiménez-Toledo, J.A.; Collazos, C.; Revelo-Sánchez, O. Considerations for the Teaching-Learning Processes of Introductory Programming Courses: A Systematic Literature Review. *Tecnológicas* **2019**, *22*, 82–116.
91. Lukkarinen, A.; Malmi, L.; Haaranen, L. Event-driven Programming in Programming Education: A Mapping Review. *ACM Trans. Comput. Educ.* **2021**, *21*, 1–31. [CrossRef]
92. Santos, S.; Tedesco, P.; Borba, M.; Brito, M. Innovative Approaches in Teaching Programming: A Systematic Literature Review. In Proceedings of the 12th International Conference on Computer Supported Education—Volume 1: CSEDU; SciTePress—Science and Technology Publications: Setubal, Portugal, 2020; pp. 205–214.
93. Liberati, A.; Altman, D.G.; Tetzlaff, J.; Mulrow, C.; Ga¸tzsche, P.C.; Ioannidis, J.P.A.; Clarke, M.; Devereaux, P.; Kleijnen, J.; Moher, D. The PRISMA statement for reporting systematic reviews and meta-analyses of studies that evaluate healthcare interventions: Explanation and elaboration. *BMJ* **2009**, *339*, b2700. [CrossRef] [PubMed]
94. Tullu, M.S. Writing the title and abstract for a research paper: Being concise, precise, and meticulous is the key. *Saudi J. Anaesth.* **2019**, *13*, 12. [CrossRef]