O1/A4 – PROGRAMMING SKILL BUILDING REQUIREMENTS FOR SECONDARY EDUCATION

INTELLECTUAL OUTPUT: O1 – METHODOLOGICAL LEARNING FRAMEWORK

Elaborated by SWU (Bulgaria) and all project partners

Document Data

Deliverable: O1/A4 – Programming skill building requirements for secondary education

Intellectual Output No - Title: O1 – Methodological Learning Framework

Elaborated by: SWU (Bulgaria)

Partners involved: All partners

Disclaimer

This project has been funded by the Erasmus+ Programme of the European Union.

TABLE OF CONTENTS

1. SUMMARY OF O1/A2 CURRENT STATUS IN PROGRAMMING SKILLS DEVELOPMENT

Courses related to Coding4Girls project in the age group of 10 to 16 are mandatory/compulsory in Greece (in each grade), Bulgaria (in 3rd - 4th Grade, 8th grade in specialised schools), Slovenia (in 1st grade of High School (70 hours)), Turkey (5th and 6th grades in Anatolian High Schools/General High Schools - 14, 15, 16 and 17 year-olds), Croatia (according to the new curriculum the programming content is integrated in all grades). In Portugal, for the ages 10 to 16, programming is learned in extracurricular activities. In Italy and Slovenia programming is also part of extracurricular activities

In all Coding4Girls project partner countries programming is additionally offered as extracurricular activities.

Diversity in learning outcomes regarding grades in different partner countries.

2. PROGRAMMING SKILL-BUILDING REQUIREMENTS:

The programming skill-building requirements for Coding4Girls project in the age group of 10 to 16 are based on the learning outcomes, defined in the partner countries' curricula.

The programming skill-building could be divided into the following modules: Algorithms; Programming; Problem Solving and Projects Development; Sharing of projects.

Algorithms

The student:

- Describes the concept of the algorithm and its properties.
- Describes everyday problems as a sequence of steps.
- Gives/recognises appropriate examples of algorithms from everyday practice, stories/ fairy tales, other school subjects (mathematics, languages, science, music, fine art, PE etc.).
- Uses an algorithm to present simple task.
- Presents the algorithm in different ways - symbolically (in a flowchart), through instructions in simple language or pseudo-code.
- Reads and follows an algorithm prepared by someone else.
- Reorders steps in an algorithm that is designed in a wrong way.

- Includes branching (if) and repetition (loops) in the algorithm;
- Decomposes a complex algorithm in sub-algorithms.
- Connects multiple algorithms to create a new algorithm that solves a problem.

Programming

The student can:

- Defines a concept programming language and explains its functions and characteristics.
- Describes functionalities of visual-block programming environment.
- Describes the main groups of blocks in a visual programming environment and explains blocks functionalities with appropriate examples.
- Reads and interprets already existing programmes, understanding the functioning of the commands involved and verbalizing the purpose of the programme;
- Uses: blocks for event-driven programming; movement, loops, branching, mathematical and logical operators, input and output of data, variables, sub-programs, some data structures (list, array), parallel executions and synchronisations of characters to solve problems with a different level of complexity.
- Creates a computer program for a given algorithm.
- Finds errors in a foreign code, optimizes the code and algorithms in given solutions of problem solving.
- Enlarges algorithms and codes in “half-backed” problems.
- Uses a loop construction - initialization, body of loop and loop’s condition.
- Evaluates the need for the use of algorithms with loop structures with counter, pre-condition and post-condition, applies cyclic algorithm constructs for verifying input data and managing a graphical user interface.
- Uses arrays; identifies array elements; recognizes index and array element value; defines an array with the means of a programming language; is able to process, input, and output the array’s elements; uses a list field to display the array element values; calculates the sum and product of the values the elements of a one-dimensional array; searches for an element of the array of maximum and minimum values, as well as elements of the array meeting a condition.

Problem solving and project development

- Formulates problems from daily life situations;
- Describes the phases of solving a computational problem;
- Analyses a problem, defines input and output values and recognizes the steps for solving the problem;
- Conducts research and analyses solutions for a group project, designs and develops a model to solve the problem set in the project assignment;
- Creates programmes that involve:
- Character and Scene animations;
- Storytelling;
- Interactive games;

- Modifies or creates his/her own characters and backstage appropriate to the topic of the project or problem;
- Explains the need to create the appropriate documentation;

Sharing of projects

- Explores other projects online;
- Presents and shares group work with his/her classmates and in an online community;
- Writes instructions to the product that was developed;
- Appreciates ethical issues when working in an online community;

3. APPENDIX – CONTENT AND LEARNING OUTCOMES ACCORDING TO O1/A2 - THE CURRENT STATUS IN PROGRAMMING SKILLS DEVELOPMENT.

3.1. Slovenia

Course name	Computer Science
Grade	5^{th} – 6^{th} grade of Primary School
Year	10 - 11
Compulsory / Elective	elective

Algorithms

- understand the concept of the algorithm,
- know how to describe everyday problem as a sequence of steps,
- know how to use an algorithm to present simple task,
- present the algorithm symbolically (with a flowchart) or with instructions in a simple language,
- follow an algorithm prepared by someone else,
- know how to include branching (if) and repetition (loops) in the algorithm,
- know how to divide a program (into functions?) into subprograms,
- know how to connect multiple algorithms to create a new algorithm that solves a problem;
- understand the role of testing the algorithm and know that testing is a tool for finding errors and not for validating the correctness;
- compare multiple algorithms to solve the problem and are able to find the most appropriate algorithm according to the given criteria;
- know how to use some of the basic search and sort algorithms;
- know some of the basic algorithms for data searching.
- know how to follow the implementation of a foreign program
- know how to write a program with the help of an algorithm,
- know how to integrate constants and variables into the program;
- understand different data types and can use them in the program;
- know how to change the value of the variable with the assignment statement;
- can read the input data in the program and include them in the program;

- know how to get the value of the variable while executing the program and get the final result as an output;
- include logical operators in the program;
- know how to use and execute a conditional sentence;
- understand the concept of a loop and can use it to solve the problem;
- understand complex data types (strings, lists / tables) and can use them in the program;
- recognize and correct errors in their program;
- correct an error in a foreign program;
- know how to change the program to reach a new mode of operation of the program;
- they can record the results of the task in a file;
- know event driven programming;
- are capable of graphic presentations of the scene (object size, background, positioning);
- are able to synchronize dialogs / sounds;
- are able to understand and realize the interactions between characters and objects;
- are able to create animations.

Data

- distinguish between data and information;
- understand the binary system for recording various data;
- understand data encoding;
- understand that there are data in various pop-ups (text, sound, images, video);
- know how to present certain data and relationships between them (binary trees and graphs);
- know data compression and understand lossy and lossless compression- explain the difference between the constants and the variables in the program;
- know how to write structured data in tables with rows and columns;
- describe the need for data editing;
- are familiar with basic algorithms for searching data;
- are aware of the importance of protecting personal data.

Problem solving

- know how to use different strategies to solve the problem;
- know the stages of the problem solving process;
- know how to ask questions and find out what data is known;
- know how to get key points of the problem, depending on the assignment;
- know how to find the right tool to solve the problem;
- know how to divide the problem into several minor problems;
- know how to plan and realize a solution;
- know how to assess the consequences and impact on the "environment";
- know how to use a known strategy in new circumstances;
- know how to create a new algorithm for more complex problems;

- can effectively participate in the group and solve the problem using information communication technology;
- can appreciate unsuccessful attempts to solve the problem as part of the way to solution;
- know how to critically evaluate the solution and find out if the solution successfully solves the given problem;
- know how to critically evaluate the problem-solving strategy;
- are aware of the limitation of information and communication technology in problem solving.

Course name	Editing Texts
Grade	7th grade of Primary School
Year	12
Compulsory / Elective	elective

Programming (*additional content)

Pupils:

- - can write an algorithm that solves a simple everyday problem,
- - can create and modify a simple computer program.

Course name	Computer Networks
Grade	8th grade of Primary School
Year	13
Compulsory / Elective	elective

- - can write an algorithm that solves a simple everyday problem,
- - can create and modify a simple computer program with decision-making

Course name	Multimedia
Grade	9th grade of Primary School
Year	14
Compulsory / Elective	elective

- can write an algorithm to solve a more demanding, but familiar and understandable problem,
- can create and modify a computer program with loops and a branching.

Course name	Informatics
Grade	1st – 2nd year of Gymnasium
Year	15 - 16
Compulsory / Elective	1st year compulsory, 2nd year elective

Students:

Computer processing data:

- define computerized data processing,
- explain the purpose of computerized data processing and define the characteristics that determine the quality of computerized data processing,
- know the meaning and the use of the program and explain the importance of programming

Algorithm:

- define the algorithm and know the basic requirements for the working algorithm,
- know the basic parts of the algorithm, can develop an algorithm to solve the problem with branching and loops, use a flowchart and justify the solution used,
- analyze an algorithm for solving a more complex problem and evaluate it.

Programming language:

- define a programming language and explain its function
- know the basic building blocks of the selected programming language, explain their function and illustrate the explanation with examples,
- define structured, object and event driven programming,
- distinguish between the compiler and the interpreter and explain the difference

Programming:

- create a computer program for a given algorithm,
- write the documentation of the program and explain its importance,
- analyze the program and evaluate the results obtained with the software solution.

EXTRACurricular activities

There are also several extracurricular activities, that promote programmig and computer science, e.g. CodeWeek (programming), Summer school of computer science UL FRI (computer science and programming), Summer school of computer science UM FERI (programming, computer science), Zavod za računalniško izobraževanje (programming), Zavod 404 (programming, physical computing). There are also several companies that organize programming courses for students of primary and secondary schools.

3.2. Portugal

Compulsory course

17 years old (12 grade)

The student can:

- Understand the concept of algorithm;
- Elaborate simple algorithms through pseudo-code, flowcharts, and natural language;
- Distinguish and identify natural and formal languages.
- Use an imperative programming language to elaborate simple programs in consoles;
- Identify and use different types of data in programs;
- Recognize different arithmetic, logical and relational operators and respective priority rules;
- Develop programs that include selection control structures and repetitive structures aiming at solving problems with low complexity;
- Use functions in programs;
- Distinguish different techniques of passing parameters to functions;
- Execute basic operations with arrays

Nationwide Extracurricular Initiatives

8-9 (3-4)
Algorithms

In Phase 1:

- Recognise that an algorithm is a group of instructions, with a sequence, that allow to reach an objective;
- Recognize that an algorithm can be represented in a simple way and can describe, for example, the activities we do daily;
- Recognize that computers need instructions more precise than humans do and that an alteration in an algorithm will culminate in a change (perceivable or not) in the result of the programme.

In Phase 2

- Recognise that different algorithms can reach the same results, some being more efficient than others;
- Be capable of decomposing a problem in parts;
- Understand that programmes can be optimized and that two programmes can have the same effect while having different programmes.

Programming

In Phase 1:

- Read and interpret programmes already existing, understanding the functioning of the commands involved and verbalizing the purpose of the programme;
- Being capable of creating programmes that involve:
  - Character animations;
  - Storytelling;
  - Artistic creation;
  - Interactive games.

In Phase 2

- Understand and use fundamental programming concepts, as:
  - sequences
  - parallel executions;
  - events;
  - decision-making structures/conditions
  - cycles;
  - operators;
  - data/variables.
- Create interactive programmes, using keys and/or mouse to input information.

Sharing

In Phase 1:

- Understand the importance of presenting and sharing group work with their classmates and in an online community;
- Get to know and explore other projects online.

In Phase 2

- Present and share their projects with classmates and share them online;
- Understand the need of mentioning instructions to the product that was developed.

Probótica – Programação e Robótica no Ensino Básico

6-14 (1-9)

Computational thinking

- Understand the dimensions involved in computational thinking;
- Identify strategies for handling problems (complexity reduction, decomposition, abstraction, adaption and adoption of known models and algorithms, data collection and analysis, etc.);
- Formulate problems from daily life situations;
- Describe and represent symbolically sequences of activities in different complexity degrees;
- Solve problems by decomposing them into smaller parts.

Algorithms

- Understand what algorithms are, how they work and their applications;
- Describe and represent symbolically sequences of activities in different complexity degrees;
- Recognize the importance of drawing algorithms as a method of solving problems;
- Solving problems by decomposing them;
- Understand that different algorithms can reach the same result and that the same algorithm can be reused for different situations;
- Recognize that some algorithms are more appropriate for a specific context than others;
- Reuse an algorithm in different situations.

Programming

Programming

- Understand and apply fundamental principles and concepts of programming (logics, types of data, variables, conditional and repetitive structures, among others)
- Analyse programmes, identifying their results, mistakes and respective correction;
- Optimize the programming of a solution;
- Draw programmes with different levels of complexity for solving specific problems;
- Create programmes to solve problems, anime stories or games, using a text-based programming language or a visual programming environment.

EduScratch

- Promote the integration of digital technologies in the development of digital literacy;
- Promote the development of programming and problem solving competencies;
- Stimulate memory, attention and logical thinking;
- Promote creativity;
- Promote interdisciplinary; Share learning resources developed by schools.

3.3. GREECE

5-6 grade (10-12 years old)

Check and code	Use a simple programming language (Logo like) to code.

- Understand that the computer is getting instructions from human in an encoded form.
- Use simple commands to create shapes or solve simple problems.
- Activities:
- Students can create simple geometric shapes by giving the right commands to move or turn the turtle.
- Through selected examples, with on purpose-mistakes, students understand that the computer is only getting and doing the instruction a person gives.

Middle school

3rd 14-15 y	Acquaintance with the computer as a system	Programming languages. Basic steps of problem solving using computer. Create and execute the program.

Goals: Students should:

- Recognize the concept of the programming language and the necessity of using it.
- Find the solution of a simple problem and implement it in a programming environment.
- Programming languages.
- Basic stages of problem solving using the computer.
- Describing and understanding the problem.
- Designing a solution to the problem.
- Describing an algorithm.
- Create and run a program.

High School

2nd grade (16-17)

Goals: Students should:

- Be able to classify a problem in the category it belongs to.
- Be able to tell the difference between a computational and a non-computational problem.
- Describe the phases of solving a computational problem.
- Describe the concept of the algorithm and its features.

- Report and relate these concepts.
- Justify the existence of different types of algorithms and explain how they work.
- Recognize the different forms of representation of the algorithm and choose the most suitable for a specific problem.
- Report the basic types and structures of data used in algorithms
- Understand the basic command and structures used in an algorithm.
- Determine how data structures work.
- Identify and correct the logical errors of an algorithm.
- Explain the need to create the appropriate documentation.

3.4. Croatia

Primary school – 70 hours per year in all grades

Computational Thinking and Programming

1st grade

After the first year of studying Computer Science in the domain of Computational Thinking and Programming, the student:

- B. 1. 1 solves a simple logical task
- B. 1. 2. follows and presents a sequence of steps required for solving a simple task

2nd grade

After the second year of studying Computer Science in the domain of Computational Thinking and Programming, the student:

- B. 2. 1 analyses a set of instructions that perform a simple task and corrects the wrong order if needed
- B. 2. 2 creates a set of instructions where he uses repetition.

3rd grade

After the third year of studying Computer Science in the domain of Computational Thinking and Programming, the student:

- B. 3. 1 creates a program by using a visual environment where he uses a sequence of steps, repetition and decision making and evaluates his solution with teacher's help
- B. 3. 2 arranges data in a useful way.

4th grade

After the fourth year of studying Computer Science in the domain of Computational Thinking and Programming, the student:

- B. 4. 1 creates a program by using a visual environment in which he uses sequencing, repetition, decision making and input values
- B. 4. 2 solves more complex logical tasks by using a computer or not

5th grade

After the fifth year of studying Computer Science in the domain of Computational Thinking and Programming, the student:

- B. 5. 1 uses program tools to create a program in which he uses input and output values and repetition

- B. 5. 2 creates an algorithm for solving a simple task, checks if the algorithm is correct, discovers and fixes errors.

6st Grade

After the sixth year of studying Computer Science in the domain of Computational Thinking and Programming, the student:

- B. 6. 1 creates, monitors and readjusts programs that contain branching and conditional looping structures and anticipates behaviour of simple algorithms that can be presented with a diagram, spoken language or programming language
- B. 6. 2 considers and solves more complex problems by separating them into a string of subproblems.

7th Grade

After the seventh year of studying Computer Science in the domain of Computational Thinking and Programming, the student:

- B. 7. 1 develops algorithms for solving various problems using a programming language while employing the appropriate structures and types of data
- B. 7. 2 applies a (sequential) search algorithm while solving problems
- B. 7. 3 designs and creates modular programs that contain subprograms in a programming language
- B. 7. 4 uses simulation while solving some, not necessarily computer, problem.

8th Grade

After the eighth year of studying Computer Science in the domain of Computational Thinking and Programming, the student:

- B. 8. 1 identifies a problem from the real world, creates a program for its solving, documents the program operation and presents how the program works to others
- B. 8. 2 recognizes and describes the sorting algorithm, applies one sorting algorithm to solving a given problem in a programming language
- B. 8. 3 recognizes and describes the possibility of applying recursive procedures to solve given problems and explores further possibilities of applying recursion.

Secondary school

General gymnasiums, language gymnasiums, classical gymnasiums and natural science gymnasiums – 2 x 70 hours a year

After the **first year of studying Computer Science** in **secondary school** in the domain of Computational Thinking and Programming, the student:

- B. 1. 1 analyses a problem, defines input and output values and recognizes the steps for solving the problem
- B. 1. 2 applies simple data types and arguments his choice, applies different types of expressions, operations, relations and standard functions for modelling of a simple problem in the chosen programming language
- B. 1. 3 develops an algorithm and creates a program in a chosen programming language while solving the problem by using branching and looping structures.

After the **second year of studying Computer Science in secondary school** in the domain of Computational Thinking and Programming, the student:

- B. 2. 1 analyses basic algorithms with simple data types and basic program structures and applies them while solving new problems
- B. 2. 2 notices smaller units in a given problem, solves them and integrates them in a single problem solution
- B. 2. 3 solves a problem by applying a one-dimensional data structure.
- B. 2. 4 in co-operation with others designs an algorithm, implements it in a chosen programming language, tests the program, documents and presents the capabilities and limitations of the program to others.

Natural sciences and mathematics gymnasium – 4 x 70 hours a year (A and C versions)

After the first year of studying Computer Science in secondary school in the domain of Computational Thinking and Programming, the student:

- B. 1. 1 analyses a problem, defines input and output values and recognizes the steps for solving the problem
- B. 1. 2 applies simple data types and arguments his choice, applies different types of expressions, operations, relations and standard functions for modelling of a simple problem in the chosen programming language
- B. 1. 3 develops an algorithm and creates a program in a chosen programming language while solving the problem by using branching and looping structures.
- B. 1. 4 applies the standard algorithms defined for whole numbers

After the second year of studying Computer Science in secondary school in the domain of Computational Thinking and Programming, the student:

- B. 2. 1 analyses basic algorithms with simple data types and basic program structures and applies them while solving new problems
- B. 2. 2 notices smaller units in a given problem, solves them and integrates them in a single problem solution
- B. 2. 3 solves a problem by applying one-dimensional data structures.
- B. 2. 4 analyses data sorting as an important concept for solving different problems

- B. 2. 5 in co-operation with others designs an algorithm, implements it in a chosen programming language, tests the program, documents and presents the capabilities and limitations of the program to others.

After the third year of studying Computer Science in secondary school in the domain of Computational Thinking and Programming, the student:

- B. 3. 1 visualizes and graphically presents a problem from his environment using a graphics module
- B. 3. 2 solves a problem by applying complex data types defined by a given programming language
- B. 3. 3 solves a problem by applying a recursive function
- B. 3. 4 compares different sorting and data searching algorithms
- B. 3. 5 evaluates algorithms according to their time complexity
- B. 3. 6 analyses traditional cryptographic algorithms and describes the basic idea of modern cryptographic systems
- B. 3. 7 defines a real life problem and creates a software solution by going through all the programming phases, presents the software solution to others and evaluates it.

After the fourth year of studying Computer Science in secondary school in the domain of Computational Thinking and Programming, the student:

- B. 4. 1 designs an object model with the associated complex data structures and implements it in a given programming language
- B. 4. 2 a* solves a problem by using abstract data structures
- B. 4. 2 b* creates an application with a graphical interface for solving a real life problem
- B. 4. 3 uses modelling and simulation to present and understand natural phenomena
- B. 4. 4 defines a real life problem and creates a software solution by going through all the programming phases, presents the software solution to others and evaluates it.

*The teacher picks the outcome B. 4. 2 a or B. 4. 2 b depending on student's interests.

3.5. Turkey

Course 1 Informatics Technology and Software 10-11 years old / 5th and 6th Grade Informatics Technology and Software, Compulsory , 72 hours per year

With Information Technology and Software Course Curriculum, the students;

- Follow and evaluate the process of reasoning";
- To develop cooperative working skills as part of the learning process, to benefit and share what they have learned;
- Developing an understanding of Algorithm design, verbally and visually.
- To be able to apply the appropriate programming approach to solve the problems;
- Create technical accumulation in programming;
- To use at least one of the programming languages;
- Work on product design and management;
- To solve the problems encountered in everyday life by developing innovative and original project (problems faced by elderly and disabled individuals).

Course 2 Informatics Technology and Software 12-13 years old / 7th and 8th grade Informatics and Software , Elective, 72 hours per year

The Program, which is developed for grades 7 and 8, is based on unit-based approach. In the program, there are four basic units for 7th and 8th classes. The title of the first unit is Information Technologies.

The contents of the first unit are : 1. Daily Information Technology 's Importance in life.

The title of the second unit is Ethics and Safety. The contents of the second units are : 1. Ethical Values, 2. Privacy and Security.

The title of the third unit is Problem Solving and Programming.The contents of the third unit are 1. Problem Solving Concepts and Approaches 2. Programming.

The title of the fourth unit is Creating a Product. The contents of the fourth unit are 1. Presentation and Visualization Programs 2. Creating Two-dimensional Animation. In 8th grade there is an additional content for fourth unit as Three-Dimensional Design Programs.

With Information Technology and Software Course Curriculum, the students;

- To acquire and develop problem solving and information-processing skills;

- Follow and evaluate the process of reasoning;
- To develop cooperative working skills as part of the learning process, to benefit and share what they have learned;
- Developing an understanding of Algorithm design, verbally and visually;
- To be able to apply the appropriate programming approach to solve the problems;
- Create technical accumulation in programming;
- To use at least one of the programming languages.

Course 3. Computer Science (for Anatolian High Schools / General High Schools) 14, 15, 16 and 17 years old. 9^{th} , 10^{th}, 11^{th} and 12^{th} degree. Compulsory, 72 hours per year.

3.6. Bulgaria

Core courses related to computer science are divided in subjects: Computer Modeling, Information Technology and Informatics.

Also courses in programming are provided by schools as extracurricular courses, by IT companies, NGO's, Private schools. In 2019 Ministry of education launched new school program for extracurricular learning with focus on STEM education. Most of the schools choose programming and ICT topics for these courses.

In this document only the core courses with focus on programming are described.

Course: Computer modeling (Modeling with computers)

Age/Grade – 9 year-olds (3th grade)

Compulsory subject

The students:

- Describe every day algorithms
- know the working space and the particular visual environment;
- order blocks in linear sequence in visual environment;
- create storytelling according to given plot using blocks in visual environment;
- implement loop algorithm;
- create animated gift card;
- share projects on the Internet.

Course: Computer modeling (Modeling with computers)

Age/Grade – 10 year olds (4th grade)

Compulsory subject

The students:

- know the particular visual environment and create digital content in it.
- Implement linear algorithms in visual environment and manage characters;
- experiment with characters in visual environment and set basic properties;
- choose the characters and their properties according to the particular plot;
- develop codes using blocks for character management in a game;
- implement looping and branching algorithms;
- create projects with animations of more objects, sound and text;
- present the project in a real and virtual environment.

Grades 5-7

As elective or exrtracuricular activities teachers can develop their own sylabi for teaching programming.

In mathematical upper-secondary schools courses are offered as extracurriclar activities for programming in C++ with the aim to prepare students for contests and olympiads in programming.

Grade 8 th (14 year olds) – Secondary schools with specialisation in Mathematics, Computer science, Natural Sciences, Technology and Entrepreneurship.

Module - **Fundamentals of Informatics**.

The module contains the topics: Informatics and Computer Science; Numbers and their Presentations; Algorithms and Programming Languages.

The following concepts are introduced: activities, information, data, discreet presentation of information, non-positional and positional number systems, exponential format, algorithm, programming language, translator.

Learning outcomes:

The student:

- recognizes and describes the basic information activities and the general scheme of information flows;
- provides examples of information processes;
- distinguishes concepts of information and data;
- explains and illustrates with examples the connection between information and data, the essence of the discreet presentation of information and its transformation into data;
- expands and generalises knowledge related to numbers and their representation in decimal, binary and hexadecimal numeral systems and perform the basic arithmetic operations;
- defines the concept of algorithm and describes its main features, describes linear, branched, and cyclic algorithms by various means;
- explains the essence and functional purpose of a programming language, translator.

Module - **Visual Programming Environments**.

The module includes the following topics: Integrated Environment for Visual Programming, Fundamental Stages of Designing and Running a Computer Program, Designing a Graphical User Interface.

Learning outcomes: The student:

- can open, launch, edit and save project in an integrated environment visual programming;
- can recognize the main components of an integrated programming environment: graphic and text editor, panels - with controls, object properties, messaging, viewing the application structure;
- should be able to edit the application design;
- to analyze a simple mathematical problem with a simple mathematical model , to create a non-complex mathematical model for solving a simple task and to compare a program solution;
- to start a pre-prepared (half-backed) computer program with a graphical user Interface; to test a simple pre-prepared (half-backed) project;
- to identify the types of errors in programming; to be aware of the purpose of major containers and controls - form, label, text box, button, dialog box; to identify basic properties of graphical objects - controls - name, state, label, background, font settings, etc.;
- to design a simple form containing graphical objects;
- to set up basic properties of objects and name them according to convention;
- to set/adjust the functionality of a button associated with displaying a static message in a dialog box;
- to design a user interface for an application.

Module **Programming**:

The module includes the following topics: Basic Data Types, Creating a Computer Program to Solve a Specific Task, Programming Structures for Branched and Loops Algorithm Implementation, Program Testing and Verification, Component Data Types. One-Dimensional Array, Processing of Rows of Elements;

Learning outcomes:

The student:

- knows and uses the basic data types - string, integer and real data types, Boolean data type, and knows rules for naming the constants and variables, declare, describe, and initialize variables and constants of the different data types, assign value to a variable, input and output data into a text field;
- knows and uses the operations and built-in functions for working with different types of data, construct expressions, keeping the syntax and semantics of the

particular programming language; the student can apply and analyze the results of operations and functions;
- knows the priorities of arithmetic operations, knows and describes an assignment operator;
- knows the basic stages of creating a computer program;
- can analyze and design the solution of a particular task, can create the mathematical model for solving the problem and develop an algorithm for its solution;
- can define the input-output data and their types;
- can structure and develop a graphical interface using objects and declare variables;
- can create, describe and shape the program code, can start, test, and validate the finished project, can remove syntactic and logical programming errors;
- can justify the need for branching of the algorithmic process;
- is able to describe a branched algorithm using a conditional operator (short and full form) and use it to check the correctness of the input data of a program, as well as for processing the properties of a radio button and a field for bookmark;
- knows and uses a loops construction - initialization, body of loop and loop's condition;
- is able to evaluate the need for the use of algorithms with loop's structures with counter, pre-condition and post-condition, to apply cyclic algorithms constructs for verifying input data and managing a graphical user interface, as well as for inputting and outputting data from a file;
- knows and uses algorithms to find a sum, a minimum / maximum element, an average arithmetic, etc. in rows of numbers entered by the user interface / keyboard;
- is able to explain and distinguish the concepts of testing and verification, to define test data;
- understand the need to use arrays, identify array elements, recognize index and array element value, define an array with the means of a programming language, to process, input, and output the array's elements, use a list field to display the array element values, calculate the sum and product of the values the elements of a one-dimensional array, search for an element of the array of maximum and minimum values, and elements of the array meeting a condition.

Module **Development of a Software Project**

The student:

- can describe the stages in the realization of a software project, to conduct research and analyse solutions for a group project;
- designs and develops a model to solve the problem set in the project assignment;
- designs a graphical user interface;
- creates a model implementation code;
- create test examples with input data and expected results;
- prepares documentation for the software project, present and defend it.

3.7. Italy

On 22/02/2018, the updated National Guidelines for the first cycle primary school curriculum, the document "National indications and new scenarios", drawn up by Italian Ministry of Education (MIUR), was presented. The document was edited by the National Scientific Committee (CSN), established by Ministerial Decree n. 254/12 for the National Guidelines implementation and the "lifelong learning in teaching".

It underlines the importance and the relevance of “computational thinking” skills development and presumes that all schools should take care of a mental process development that allows solving problems of various kinds, applying different specific methods and tools with a planning strategy.

Actually, computational thinking is identified as one of the cultural tools for an active citizenship. Schools are called to teach students how to act consciously allowing them to deal with situations analytically, breaking them down into the various characterizing aspects with subsequent analysis and adaptation of the suitable solutions.

One example is described by ABACUS Project implemented in the Industrial Technical Institute, focused on the renovation of Informatics curriculum for the secondary school students (age 16-18). It intends to respond to the needs for the curricular revision proposal. Starting from this base, experiments will have to be activated to allow the evaluation of the choices made and to practice the necessary corrections.

Course: Computer Science in Industrial Technical Institute

Target group: 16 years old Students

Objectives

- Solving problems, regardless of a programming language.
- Setting problems, even from a non-procedural point of view. Verify the correctness of a solution.
- Reading and interpreting syntactic descriptions in multiple notations.
- Reading and interpreting programs in multiple languages.
- Using the command language of an operating system.
- Using imperative language with properties.
- Using at least one non-imperative language.
- Documenting software at the elementary level.

Contents

- Computer science as a science and as a technology (third, fourth, fifth classes).
- Information concept and roles.

- Mathematical and technological origins of computer science.
- Main ramifications of computer science.
- Software production.
- Software life cycle.
- Software quality factors.
- Testing techniques.
- Top-down and bottom-up development methodologies.
- Software documentation: elementary techniques.
- Intuitive definitions of syntax (in a language) and semantics (in an operating environment) and simple descriptive tools (syntactic graphs, BNF notation, ...).
- Concepts of recognition and language generation.
- Problems and programs.
- The world of problems: classifications and generalizations. Intersections with introductory topics of Artificial Intelligence (planning problems, research spaces, heuristics, ...).
- Distinction and role of languages: natural, design, programming.
- Comparison between procedural and non-procedural programming.
- Recursion as a conceptual scheme.
- Algorithm concept as a solution to a parametric problem.
- Data, results; actions, processes, status of a process; performers.
- Introduction to imperative programming.
- Executive model: the classic model by Von Neumann.
- Role of the operating system.
- Variables, expressions, assignment, elementary types.
- Structures of control and structured programming.
- Sub-programs: - functions and procedures; - recursive programming. - parameter passing techniques, local and non-local environment (visibility rules); Arrays and their classical elaborations (research, ordering, fusion, ...).
- Elements of non-procedural programming General characteristics, origins and motivations.
- Functional paradigm:
    - characteristics and control methods;
    - lists and related operations;
    - representations of expressions (prefix, postfix, list prefix, ...);
    - trace of functional programs;
    - significant applications;
    - relations with other paradigms.
- Logical paradigm:
    - logic elements of the first order;
    - variables, constants and terms;
    - the clauses of Horn;
    - unification;
    - procedural and declarative interpretation;
    - trace of logical programs;
    - significant applications;
    - relations with other paradigms.

Course: Information Processing and Transmission System in Industrial Technical Institute

The teaching of Information Processing and Transmission Systems includes two distinct areas of interest. The first, more technological in nature, is the area of systems for processing (computers) and the transmission (networks) of information, known above all from the point of view of architecture, i.e. at the level of the boundary between the competences of the electronics and those of advanced programming. The second area of interest is that of the applications which represents the knowledge of the systems in the sectors of industry and services, affected by the progressive introduction of IT tools.

EXTRACurricular activities

School Projects

The proposed activities have the objective of the computational thinking spreading among the students of various classes, during the curricular hours of the current school year, according to the provisions of Law 107/2015 - The Good School. The pupils, under the guidance of the chosen teachers, appropriately trained, developed simple Coding experiences exploiting the paths proposed by the "Program the future" project, with the resources available on Code.org, as a part of the initiative promoted by MIUR in collaboration with CINI (National Interuniversity Consortium for Information Technology).

The projects are focused on the introduction, in an intuitive and fun way, of the basic programming concepts to bring the computational thinking into a classroom using Coding in the secondary school students. These concepts are:

- algorithm
- visual block programming
- execution of sequences of elementary instructions
- repeated execution of instructions
- conditional execution of instructions
- definition and use of procedures
- definition and use of variables and parameters
- verification and correction of the code
- reuse of the code

Programma FUTURO

The course proposed is structured into 5 UNITS:

1. Internet – Sending binary messages; Encoding and sending numbers; Encoding and sending text; IP addresses, packets and redundancy; Routing, DNS, Protocols and Abstraction – Practice PT – the internet and society.
2. Digital Data – Text compression; Encoding images; Practice PT – Encode and experience; Interpreting visual data; Communicating with visualization; Cleaning data and making summary tables; Practice PT – Tell a data story.
3. Algorithms and Programming – Designing algorithms; procedural abstraction & top down design; writing functions; loops and documentation; Practice PT – Design a digital scene.
4. Big Data and Privacy – Big data in the real world; Identifying and the cost of "free"; foundations of encryption; Asymmetric and public key encryption; Practice PT: the big data dilemma.
5. Implementation of an App – Designing event-driven apps; user input and variables; Boolean logic and conditionals; Practice PT – digital assistant; while loops; simulations; arrays; functions with return value; processing arrays; Practice PT – improve an app.
6. Final test – Preparation – create performance task (12 hours); preparation – explore performance task (8 hours).

The course, lasting between 150 and 200 hours per students, can be carried out both during a single school year and during several years, depending on the number of weekly hours that can be assigned to secondary school students (II Grade).

Each unit is organized in a number of lessons (between 10 and 20), each of which presupposes the knowledge of the concepts and the acquisition of the skills covered by the previous lessons. Each lesson (lasting about 2 hours) aims to lead the student from a premise or from a need to carry out an activity that makes him acquire knowledge and skills related to one (or more) specific goal of learning.