



CURRENT STATE OF PROGRAMMING IN
PROJECT PARTNERS' HIGH SCHOOLS AND
UNIVERSITIES AT COUNTRY LEVEL -
VERTICAL ANALYSIS



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

Project	Object Oriented Programming for Fun
Project acronym	OOP4FUN
Agreement number	2021-1-SK01-KA220-SCH-00027903
Project coordinator	Žilinska univerzita v Žiline (Slovakia)
Project partners	Sveučilište u Zagrebu (Croatia) Srednja škola Ivanec (Croatia) Univerzita Pardubice (Czech Republic) Gymnazium Pardubice (Czech Republic) Obchodna akademia Povazska Bystrica (Slovakia) Hochschule fuer Technik und Wirtschaft Dresden (Germany) Gymnasium Dresden-Plauen (Germany) Univerzitet u Beogradu (Serbia) Gimnazija Ivanjica (Serbia)
Year of publication	2023

Disclaimer:
Funded by the European Union. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or Slovak Academic Association for International Cooperation. Neither the European Union nor the granting authority can be held responsible for them.

Table of contents

1. Czech Republic.....	8
1.1. Introduction.....	8
1.2. Legislative framework	9
1.3. Computer science education at the gymnasium.....	9
1.4. Courses at University of Pardubice, Faculty of Electrical Engineering and informatics (computer science study program)	10
1.5. Comparison and gap analysis	11
1.6. Conclusion	12
2. Slovakia.....	13
2.1. Introduction.....	13
2.2. Legislative framework	14
2.3. Computer science education at the Business Academy in Považská Bystrica	15
2.4. Courses at University (computer science study programme).....	16
2.5. Comparison and gap analysis	17
2.6. Conclusion	18
3. Germany.....	19
3.1. Introduction.....	19
3.2. Legislative framework	19
3.3. Computer science education at the gymnasium.....	19
3.4. Courses at University (computer science study programme).....	20
3.5. Comparison and gap analysis	20
3.6. Conclusion	21
4. Serbia.....	22
4.1. Introduction.....	22
4.2. Legislative and institutional framework of the education system in the Republic of Serbia....	22
4.3. Secondary education in the Republic of Serbia.....	24
4.4. Higher education in the Republic of Serbia.....	26
4.5. Competence analysis and knowledge analysis of first-year students.....	27
5. Croatia	39
5.1. Introduction.....	39
5.2. Legislative framework	40
5.2.1. Law on the Croatian qualification framework.....	40

5.2.2.	National qualification and occupation standards	41
5.2.3.	National curriculum for preschool, elementary and high school education.....	42
5.2.4.	Curriculum for informatics subject in elementary and grammar schools.....	42
5.2.5.	Rulebook on taking the state matura exam	43
5.2.6.	Curricula for programming related subjects in university studies	44
5.2.7.	Conclusion on legislative analysis.....	45
5.3.	Gap analysis.....	46
5.3.1.	Freshmen students' prior competencies and knowledge analysis.....	46
5.3.1.1.	Introduction	46
5.3.1.2.	Data analysis	48
5.3.2.	Teachers' expectations – semi-structured interview	62
5.3.2.1.	Introduction	62
5.3.2.2.	Interview design	62
5.3.2.3.	Conducting the interview	66
5.3.2.4.	Deciding on analysis method.....	66
5.3.2.5.	Case study 1 – experienced teacher	67
5.3.2.6.	Case study 2 – young teacher	69
5.3.2.7.	Case study 3 – teaching assistant / student demonstrator.....	70
5.3.2.8.	Comparing cases.....	72
5.4.	Conclusion on gap analysis.....	74
6.	Conclusion	76

List of tables

Table 1 - Number of years in high school in which students attended subjects with IT or related content, i.e. with programming content	28
Table 2 - The distribution of the points for Programming basics questions grouped by the types of high schools.....	33
Table 3 - Total number of right answers in Programming Basics.....	34
Table 4 - Students' knowledge of sorting algorithms.....	35
Table 5 - Average number of correct answers in OOP	36
Table 6 - Total number of correct answers in OOP	36
Table 7 - Students' estimations and average number of points	37
Table 8 – Review of teachers' experience.....	72

List of charts

Chart 1 - The structure of respondents according to the type of secondary school completed	27
Chart 2 - The current knowledge of programming and its relation with the contents that were covered in school or were covered by the respondents themselves	29
Chart 3 - Programming languages in which respondents can independently apply basic concepts	30
Chart 4 - The structure of the respondent's response to teamwork during classes in secondary schools	31
Chart 5 - The structure of the responses related to projects or project activities during your high school education.....	32
Chart 6 - The average number of points by the types of high schools.....	33
Chart 7 - Distribution of knowledge for OOP	35
Chart 8 - Total number of points in both groups of questions.....	37
Chart 9 - The students' knowledge estimation	38
Chart 10 - Distribution of respondents by subjects	48
Chart 11 - Distribution of respondents by type of high school	49
Chart 12 - Number of years taking informatics and related subjects in high school	49
Chart 13 - Number of years with programming contents within informatics subjects in high schools...	50
Chart 14 - The way students acquired their programming knowledge	51
Chart 15 - Programming concepts that high school students are familiar with and have practical experience	52
Chart 16 – Usage of programming languages by students in high schools.....	53
Chart 17 – Teamwork experience among students in high schools.....	54
Chart 18 – High school students' self assessment of programming knowledge.....	54
Chart 19 – How the students recognized the definition of an algorithm	55
Chart 20 – Recognition of basic algorithmic structures	56
Chart 21 – Results of three tasks with code analyses	57
Chart 22 – Distribution of students who are/are not familiar with basic OOP concepts	58
Chart 23 – Comparison of distributions of respondents by schools (all respondents) and respondents by schools familiar with OOP concepts	58
Chart 24 - How the students recognized the definition of a class	59
Chart 25 - How the students recognized the definition of an object.....	60
Chart 26 – Understanding definitions of basic OOP concepts	60
Chart 27 – Characteristic of an abstract class	61
Chart 28 – Class/object relationship statements	62

List of pictures

Picture 1 – Education system in Slovakia	14
Picture 2 - The education system in the Republic of Serbia with qualification levels.....	24
Picture 3 – Introduction to the questionnaire in Croatian language	47
Picture 4 – Example of code section with iteration	56
Picture 5 - Responses – Semi-structured interview with teachers	63

1. Czech Republic

1.1. Introduction

Our task is to analyze details of teaching and learning at the Gymnasium Pardubice and University of Pardubice, Faculty of electrical engineering and informatics.

First, let us start with description of educational system in Czech Republic. Education has following levels. It begins with elementary education at elementary school. Elementary education is mandatory. Pupils start at age 6 or 7 years. The length of education is 9 years (primary stage encompasses grades 1-5, the lower secondary stage is grades 6-9.). At the end of primary stage, there is possibility to enter 8-year grammar schools. There are also some 6-years grammar schools for pupils after 7 years on elementary schools.

Upper secondary education can be either general or vocational. It is generally four years in length (grades 10-13). This education is not mandatory. It is finished with either vocational certificate (in Czech “výuční list”) of leaving exam (in Czech “maturita”). Only students with leaving exam can attend universities. After 2008, leaving exams consist of two parts. A common (state) exam and a profiling (specific for individual schools) exam. This was implemented due to better comparability of the final examinations across different schools and better preparation of students for universities.

Universities offer bachelor’s, master’s, and doctoral degrees. The length of a bachelor’s study is usually three years. After finishing bachelor’s degree, students can either leave their studies or continue to the master’s program (master’s or engineering degree), which usually takes 2 years. Last type of university study is doctoral study. It is for students after successful master’s study. Its length is 3-4 years.

Gymnasium Pardubice offers 4-years general study program (for students after 9 years of elementary schools) and 8-years general study program for students after 5 years of elementary school (primary stage).

University of Pardubice, Faculty of electrical engineering and informatics offers following study programs.

1. Bachelor’ degree
 - a. Applied Electrical Engineering
 - b. Communication Technology
 - c. Automation
 - d. Information Technology
 - e. Web Technologies
2. Master’s degree
 - a. Automatic Control
 - b. Information Technology
 - c. Communication and Radar Systems
3. Doctor’s degree
 - a. Electrotechnics and informatics

To identify gaps in education in the field of OOP between Gymnasium Pardubice and the Faculty of electrical engineering and informatics, we focus on the analysis of informatically oriented subjects that are taught at the Gymnasium Pardubice and on the analysis of programming subjects in the 1st and 2nd grades at Faculty of electrical engineering and informatics

1.2. Legislative framework

The following sources were used for this analysis:

- **Faculty of electrical engineering and informatics**
 - Study programs
<https://studuj.upce.cz/fakulta-elektrotechniky-informatiky>
- **Gymnazium, Pardubice, Dasicka 1083**
 - new study program for 8-years study (from 2022)
<https://www.gypce.cz/download/69042/>
 - old study program for 8-years study (2016-2021)
<https://www.gypce.cz/download/57627/>
 - study program for 4-years study (from 2018)
<https://www.gypce.cz/download/56977/>
 - appendix to study program for 4-years study (from 2018)
<https://www.gypce.cz/download/56981/>

1.3. Computer science education at the gymnasium

The school education includes computer science courses during 1st to 6th grade for all students in 8years study program and during 1st to 2nd grade in 4-years program in the subject Informatics. This subjects Informatics is mandatory and cover very general topics and some topics related to programming. Students have informatics 30 hours each year from 1st to 4th year, 60 hours each year in 5th and 6th year of 8years study program and 60 hours each year in 1st and 2nd year of 4years study program.

In 2022 there is a study program change in Czech Republic. Until now students were taught the basics of working with a computer, making documents, working with spreadsheets, presentation skills, computer graphic, multimedia skills, working specialized software and making web pages with simple web applications. Only small part of informatics was related to programming. It included basics of algorithmization, notation of algorithms with a flowchart and simple programming and making of simple applications in Scratch.

New conception of subject informatics in whole Czech Republic started new study program change at Gymnasium Pardubice. Some topics will be moved from the subject informatic to another subjects.

Making documents and presentation skills will be moved to language subjects, working with spreadsheets will be moved to mathematics and physics. Also some special software will be moved to other subjects for example computer graphics will be moved to art education. Subject informatics will be oriented to digital technologies, data modelling, robotics, algorithmization and programming. Programming should start with Scratch and finish with programming language (Python, Java, ...).

At the 7th and 8th year in 8 years study program and 3rd and 4th year in 4 years study program there is an optional 2 years subject Seminar of programming. It is a subject suitable for students that will attend technical and informatically oriented universities. This subject is based on Java programming language, so principals of object oriented programming are taught there. At first year following topics are taught in 60 teaching hours:

- basic of algorithmization
- basic of object principal (object, class)
- Java language with Netbeans editor
- making objects, constructor, methods
- inheritance
- method overloading
- data types and variables
- conditions, cycles
- array
- recursive algorithms
- sorting algorithms
- classes for working with text

At second year following topics are taught in 45 teaching hours:

- exception dealing
- classes for working with files
- java application with form
- abstract classes, abstract methods
- interface
- basic data structures – queue, stack
- basics of computer networks

1.4. Courses at University of Pardubice, Faculty of Electrical Engineering and informatics (computer science study program)

Bachelor's study program Information technology is focused on teaching OOP. It is also suitable program for Gymnasium Pardubice grades. Students that passed seminar of programming there have basic OOP skills.

At first year of study in the winter semester students have three subjects related to programming. They are:

- Basics of Algorithmization
- Algorithmization and programming practicum
- Basics of Programming Using Java Programming Language

Basics of algorithmization has a supplementary course Algorithmization and programming practicum. The aim of these courses is to familiarize students with the basics, algorithmic way of thinking and prepare students for learning the basics of programming. Subject Basics of Programming Using Java Programming Language is taught with Java programming language. OOP principal is started there. No previous knowledge of programming is required. However, the ability to work with a computer is assumed. At summer semester subject Object oriented programming is taught. Passing the final exam from Basics of Programming Using Java Programming Language is not required but knowledge and skills from the subject are necessary.

At the second year of study subject Data structures subject is taught. It is based on familiarization with basic data structures and associated algorithms. Skills from this subject are necessary when designing and implementing effective software applications. Knowledge of first year subject is necessary. However, successful exam passing not.

From the beginning of summer semester at second year of study to the end of study other programming languages are taught. They are languages C, C++, .NET and C#. Web applications are also taught.

After successful study of bachelor's degree, students can continue to get master's degree in study program Information Technology. Students improve their skills in software design, advanced programming techniques, parallel computing and in theoretical part of informatics - Turing machines, the problem of decidability and complexity of algorithms. After successful master's degree student can continue to doctor's degree in study program Electrotechnics and informatics.

1.5. Comparison and gap analysis

The gymnasium is focused on general education. Since 2022 new concept of informatics has been started. More programming skills will be taught. Last two years students can choose optional subjects. One of them is seminar of programing, which gives good starting position to study technical or informatically oriented university. Such university is also University of Pardubice, Faculty of electrical engineering and informatics with its study program Information Technology.

Problem is that without optional subject seminar of programing there is no OOP topic during the study in Gymnasium Pardubice. Without seminar students have bad start position in informatically oriented study programs in universities.

20% of graduates of gymnasium study technical and informatically oriented universities. Problem in whole Czech Republic is that only small number of students attend technical universities, which have free capacities.

In University of Pardubice, Faculty of electrical engineering and informatics with its study program Information Technology no knowledge of programming and OOP is officially required at the beginning

of study. The result of this fact is a large number of unsuccessful students that do not complete university study.

1.6. Conclusion

After comparing both institutions, we can conclude that the main problems in secondary school education is small number of hours related to teach algorithmization, programming. OOP topics are missing in mandatory subjects.

Due to fact that study in Faculty of electrical engineering and informatics is based on OOP the principal of OOP should be started at secondary education. Students should be also motivated to acquire knowledge in the field of algorithmization and programming. All should start with new concept of informatics since 2022. However new concept does not include OOP principals, which should be added.

For improving OOP skills, the following should be done:

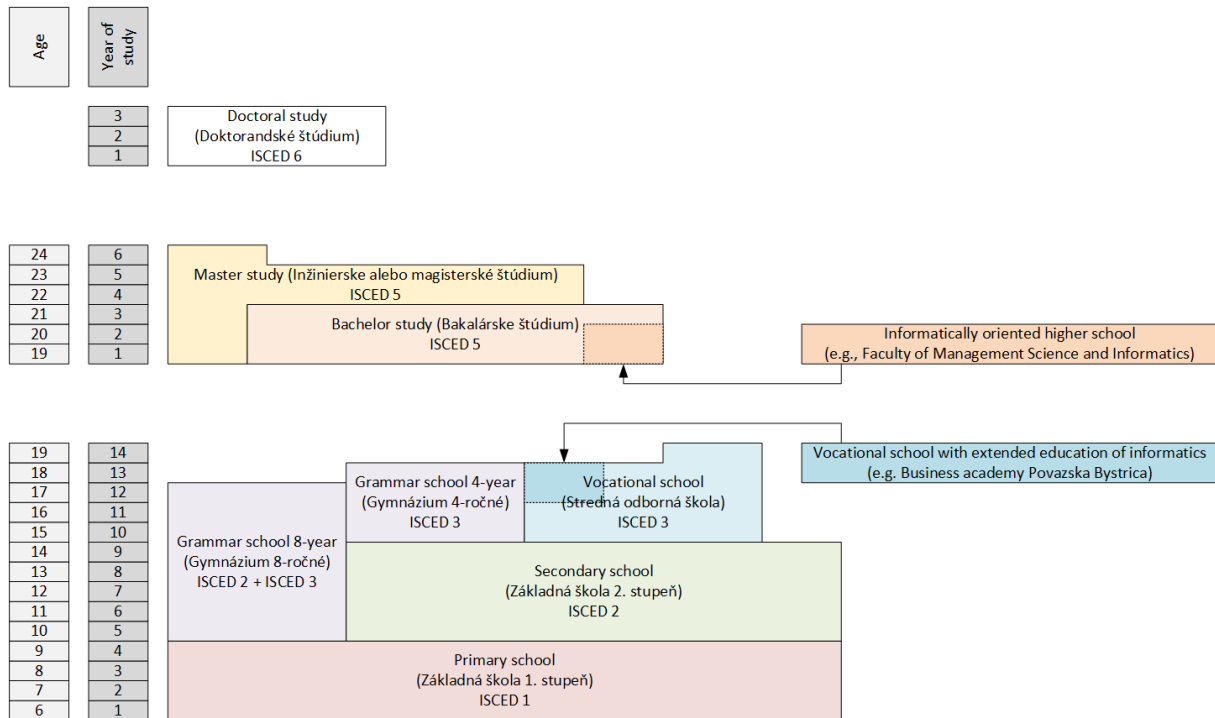
- Improve the attractiveness of teaching OOP – explain it in playful way
- Add OOP principal to teaching of programming at secondary schools
- Improve the motivation of students at gymnasium to study technical and informatically oriented universities
- Prepare of materials with OOP principal for teachers and students in gymnasium.
- Motivate students in gymnasium to improve their programming activities

2. Slovakia

2.1. Introduction

Within the vertical analysis, our task is to analyze the teaching process at the Business Academy Povazska Bystrica (OAPB) and the Faculty of Management Science and Informatics of the University of Zilina (UNIZA). We analyze teaching subjects that focuses on computer science, algorithmizing and programming. Especially, we focus on the subjects that most represent the objectives of the OOP4FUN project in their content and try to identify possible gaps and problematic parts that occur on the part of the OAPB or UNIZA.

Firstly, we would like to briefly describe the education system in Slovakia (visually represented on the Picture 1). Compulsory education begins in primary and secondary school. In primary school, pupils start at the age of 6 or 7 years and length of education is 4 years. In secondary school, pupils start at the age of 10 or 11 years and the normal duration is 5 years. At the end of the primary school, it is possible to enter an 8-year grammar school, which combines the secondary and upper secondary school (starting age is the same as for the secondary school, duration 8 years). After finishing secondary school, pupils aged 14-15 years apply to upper secondary school to continue their education. Within the education system, there are 2 basic types of these schools – 4-year grammar schools and 3-5 years vocational schools. Regarding vocational schools, lot of types exist depending on the focus and objectives of teaching. One such type of secondary school is the business academy (this also includes the OAPB), at which the duration of study is 4 years. During the last year, the graduation exam (in Slovak “maturita”) have to be completed. After successful graduation from upper secondary school, graduates can continue their studies at the university (students enter by default at the age of 18-20 years and the duration of study is 3-6 years, depending on the study program). After completing a master's degree, university graduates can continue their doctoral studies. This takes 3-4 years depending on the study subject. The Faculty of Management Science and Informatics of UNIZA offers education in three levels of university studies (bachelor's, master's and doctoral). Within the faculty, bachelor's degree programs are 3-year, master's 2-year, and doctoral 3-year.



Picture 1 – Education system in Slovakia

In order to identify gaps in education in the field of OOP between OAPB and the Faculty of Management and Informatics of UNIZA, we focus on the analysis of informatics-oriented subjects that are taught in the 3rd and 4th grades at the OAPB (pupils aged 17-19) and in the 1st and 2nd grades at the Faculty of Management Science and Informatics of UNIZA (students aged 19-21).

2.2. Legislative framework

In addition to the documents produced by the project, the following sources were also used for this analysis:

- Information about teaching at the Faculty of Management Science and Informatics of the University of Zilina:
 - <https://www.fri.uniza.sk/stranka/ponukane-bakalarske-studijne-programy>
 - <https://www.fri.uniza.sk/stranka/ponukane-inzinierske-studijne-programy>
 - <https://vzdelavanie.uniza.sk/vzdelavanie/plany.php>
 - <https://www.fri.uniza.sk/stranka/akreditacne-spisy>
- Information about teaching at the Business Academy in Považska Bystrica:
 - <https://oapb.edupage.org/a/ucebne-plany-tried>
 - <https://oapb.edupage.org/a/skvp>
- Mutual meetings between OAPB and UNIZA teachers during which they exchanged their experiences as well as views on the current teaching process.

2.3. Computer science education at the Business Academy in Považská Bystrica

In Slovakia, an upper secondary school is an education provider for secondary school graduates (3rd level in the international ISCED classification). The aim of upper secondary education is to develop the knowledge, skills and abilities acquired in primary and secondary school, while also placing emphasis on the personal development of the pupil. OAPB provides pupils with 4 years of study, which is completed by a graduation exam. As a priority, the school is focused on deepening knowledge and practical skills in the field of economics and accounting. In addition, it also has specialized classes focused on sports training, extended foreign language teaching and extended informatics teaching.

OAPB offers general basics in the field of computer science to pupils of all specializations. This is provided during the 1st and 2nd year within the subject **Informatics** (1 h per week; a total of 32 h during one school year). Here pupils are taught the basics of working with a computer, work with office suites, typing, working with specialized software for the needs of accounting and conducting financial operations.

In addition to the above-mentioned subject Informatics (1 h per week in 1st and 2nd year of study), specialized classes focused on the extended teaching of computer science also offer several other informatically focused subjects, which explain work with a simple IDE, illustrate the basics of algorithmizing using the BBC Micro:bit platform and provide a basic information about the creation of simple web applications. These subjects are **Seminar in Applied Informatics** (2 h per week in 1st and 2nd year; a total of 64 h in school year) and **Web Application Development** (1 h per week in 2nd year; a total of 32 h in school year).

More advanced teaching of algorithmizing and programming is taught in subject **Seminar in Applied Informatics** (2 h per week in the 3rd and 4th year; a total of 64 h in 3rd year and 46 h in 4th year). Pupils in 3rd year work primarily in the Python programming language by using IDE called PyCharm. The main goal is to familiarize pupils with structured programming and with the basics of procedural programming. Thus, the main topics of this subject are:

- data types in general,
- basic commands for working with input and output in the terminal,
- arithmetic and relational operations,
- variables (local, global, function parameters),
- functions with and without return value,
- conditions and cycles,
- the concept of encapsulation, which is also related to the introduction of the basic terminology of OOP (class, instance, attribute, method).

Pupils in 4th year are engaged primarily in OOP. This paradigm is explained with the use of Java programming language and the BlueJ IDE. The main teaching goals are:

- repetition of the basics of structured programming,
- working with strings, enumerated types, basic containers (array list and linked list), random numbers and text files
- more advanced principles of OOP (inheritance, polymorphism, abstract class, interface, exceptions),

- basic concepts of UML (class diagram),
- basics of debugging,
- creating a simple graphics.

During the subject Seminar in Applied Informatics, which has 110 h of teaching in total (64 h in 3rd year and 46 h in 4th year), only about 56 h (10 h in 3rd year and 46 h in 4th year) are devoted to the OOP. Teachers of this subject are directly from the Faculty of Management Science and Informatics of UNIZA.

As for the overall Informatics preparation of pupils at the OAPB, they can also participate in special events, such as the programming week, which takes place directly at the OAPB or the programming courses organized by the Faculty of Management Science and Informatics of UNIZA for upper secondary school pupils. These activities are not attended by all pupils. Usually pupils, who are interested in continuing their studies in informatics at a university, are involved in them.

2.4. Courses at University (computer science study program)

In Slovakia, the university is an educational and scientific institution for the education of the most qualified specialists (5th degree in the international classification of ISCED). It can be divided into faculties. Only an applicant who graduated from upper secondary school by passing the graduation exam (in Slovak “maturita”) can be admitted to the university. At the Faculty of Management Science and Informatics of UNIZA it is possible to study:

- bachelor's degree programs (typically 3 years or 6 semesters of full-time study):
 - informatics,
 - informatics and management,
 - information and network technologies,
 - computer engineering,
 - management,
- master's degree programs (only for graduates of bachelor's studies, typically 2 years or 4 semesters of full-time study)
- doctoral study programs (only for graduates of master's studies, typically three years or 6 semesters of full-time study).

From these study programs, several are those in which OOP is considered and, at the same time, they are suitable for OAPB graduates. These include the following ones:

- informatics,
- informatics and management,
- information and networking technologies.

The teaching of informatics and programming in 1st year of study in all these study programs focuses on the teaching of OOP. Specifically, the subjects are **Informatics 1** (1st semester, compulsory subject, 60 h of OOP teaching), **Informatics 2** (2nd semester, compulsory subject with the need to successfully complete the subject Informatics 1, 45 h of OOP teaching), **Practice of Programming 1** (1st semester, optional subject, 20 h of OOP teaching) and **Practice of Programming 2** (2nd semester, optional

subject, 20 h of OOP teaching). All these subjects are taught in Java programming language. In case of Informatics 1, no previous knowledge of programming is required (so that pupils who have only marginally encountered programming during their upper secondary studies can enter the faculty). However, the ability to work with a computer is assumed. In case of Informatics 2, passing the final exam from Informatics 1 is required; therefore, the knowledge obtained from this subject is expected in Informatics 2. The subjects Practice of Programming 1 and Practice of Programming 2 are designed to be optional subjects for students of the subjects Informatics 1 and Informatics 2. The teaching of these subjects focuses on the repetition and further explanation of the concepts discussed in the compulsory subjects Informatics 1 and Informatics 2.

In 2nd year of study (3rd and 4th semester), students of the Faculty of Management Science and Informatics of UNIZA have subjects in which the knowledge acquired in 1st and 2nd semester is expanded with more advanced concepts of OOP, in which OOP concepts are taught in other programming languages (typically C++), and in which the object approach is used to illustrate and explain the work with data in operation memory. These topics are covered by subjects **Informatics 3** (3rd semester, compulsory subject with the need to successfully complete the subject Informatics 1, 15 h of OOP teaching and another 40 h of teaching topics related to OOP) and **Algorithms and data structures 1** (4th semester, compulsory subject with the need to complete subjects Informatics 2 and Informatics 3, 13 h of OOP teaching and further 13 h of teaching related to OOP). For these subjects, knowledge from the subjects of 1st and 2nd semester, the basic ability to analyze a problem that must be solved with the use of programming skills, and the ability to critically evaluate the proposed solution are assumed.

2.5. Comparison and gap analysis

The main focus of the OAPB is not to prepare future students for informatically oriented study programs of universities, which means that topics related to informatics are taught in general. The exception is teaching in 3rd and 4th year of specialized classes with extended informatics teaching. However, the number of teaching hours is not sufficient to cover all the problems in the field of OOP in more detailed way.

Another problem is the motivation of the pupils of OAPB, since not all graduates continue their studies further at a university. Furthermore, only several graduates, who continue their studies at a university, select study programs related to informatics.

The problem within the teaching itself at OAPB is in the inconsistency of the used programming languages (Python the 3rd year and Java in 4th year) or the fact that the teaching of the OOP starts at the end of 3rd year.

At the university, no experience with programming is expected in the teaching of compulsory subjects Informatics 1 and Informatics 2. In addition, there are also optional subjects (Practice of Programming 1 and Practice of Programming 2), which give an opportunity to students of 1st year to supplement knowledge and to take a closer look at the problematic parts from the compulsory subjects.

In case of students, the problem is also in the increase in the pace of teaching compared to how it was at upper secondary school. Although, no knowledge of programming is officially required at the

beginning of the study, the basics or even more advanced thematic areas of programming and OOP are a great advantage. These often represent a good basis for successful completion of the 1st year at the Faculty of Management Science and Informatics of UNIZA.

At the university, there is also a presumption of greater independence in the work on assignments (this applies mainly to term papers) and at least basic algorithmic thinking.

2.6. Conclusion

Based on a comparison between the two institutions, it can be concluded that the main problems in upper secondary school relative to higher education are the later focus on teaching OOP programming, the inconsistency in the use of programming languages for teaching, the low motivation of pupils to learn in the field of programming and OOP, and insufficient time for the needs of teaching OOP in more details.

To eliminate these problems, it would be worth considering the following options:

- improving the attractiveness of teaching OOP at the OAPB using materials that explain the concepts of OOP in a playful way (for example, the use of the material “Object approach to problem solving”);
- creating an optional subject at the OAPB which would make a thorough explanation of the concepts used in programming and OOP, make the teaching of OOP more attractive and make it possible to work on practical projects using OOP;
- improving motivation of OAPB pupils by completing special activities at the Faculty of Management Science and Informatics of UNIZA (schools of programming, open day, invited lectures, workshops, etc.), which would make it more attractive for pupils to continue their education in informatics at the Faculty of Management Science and Informatics of UNIZA,
- to unite the programming language used on the OAPB. It would be convenient to synchronize it with the programming language used in the compulsory subjects Informatics 1 and Informatics 2 at the Faculty of Management Science and Informatics of UNIZA.

3. Germany

3.1. Introduction

This section focuses on details of teaching and learning at the Gymnasium Dresden Plauen and Dresden University of Applied Science (HTW Dresden). The description follows the goal to identify how students get prepared in school for an university study program of computer science, in particular whether computer programming and object-oriented programming is concerned in school education. A further aspect are the competences expected from the beginners in the study courses.

3.2. Legislative framework

Our considerations are based on documents as follows:

- module description of university courses that are a supplement of the study program documents, for example of the computer science course at HTW Dresden:
 - <https://apps.htw-dresden.de/app-modulux/frontend/studiengaenge/>
- the gymnasium works according the curriculum in school education issued by the federal state Saxony:
 - www.bildung.sachsen.de/apps/lehrplandb/

Teachers of the gymnasium and the responsible professor of the university exchanged their insights and experiences.

3.3. Computer science education at the gymnasium

The school education includes consecutive computer science courses during 7th to 10th grade for all students. This subject covers very general topics, only partly related to programming.

Two mandatory learning units (German Lernbereich) on "programming concepts" and "data structures" are placed in the 11th and 12th grade and cover 14 and 10 teaching hours respectively. It depends on the teacher and the students how deeply object-oriented programming is addressed. A learning unit is completed with a written exam and credited in a semester grade combined with other learning units.

In addition, the school organizes weekly project sessions after school classes for interested students. These projects reach a sufficient quantitative intensity. With regard to content these projects are comparable to university education.

However, projects are optional, take place in addition to the school classes and are attended solely by a couple of interested students. These projects are not credited for grades.

Statements from an informatics teacher of school:

"OOP plays a minor role to pass the exams for the upper school grades."

"A student may choose informatics for an oral exam part within the final secondary-school examinations. Then the responsible teacher decides about the main focus areas of the exam. OOP plays a minor role."

3.4. Courses at University (computer science study program)

The study program computer science contains five modules in the first two years.

These five modules are strongly related to programming in-depth and cover around 60 hours teaching each. Three modules are mandatory and cover the full range of categories regarding OOP (OOP using, OOP teaching and OOP practicing). In this way, students and professors cannot 'work around' object-oriented programming.

Prior Knowledge

The beginner courses do build on specific prior knowledge in programming. In fact, programming is taught from the beginning, with a "readiness of mind" as the only requirement. However, prior knowledge and competences are a supporting factor for a successful mastering of the study program.

The beginner courses advance quickly and head for a sophisticated level of programming competences at the end of the first year at the university.

3.5. Comparison and gap analysis

With respect to informatics (computer science) the gymnasium is focused on broad knowledge and broad competences, not specifically for programming. The gymnasium delivers an introduction to programming concepts only. This is sufficient for young people who continue their career in different fields and not specifically in the STEM fields.

In-depth programming education and self-studies are supported for those students that are interested and possibly intend to study computer science or another discipline in the STEM field.

In contrast, during the first two years in a university study program (computer science and other related programs) students have to acquire in-depth competences of programming in general and object-oriented programming. These competences are necessary to pass the exams and for further progress in the study program.

A gap (in terms of missing technical knowledge or absence of readiness of mind) may occur when students enter computer science study program without any kind of prior engagement in this field. Such an engagement could be a private hobby, or the aforementioned projects that are supported at the Gymnasium.

From the experience of a university professor, a noticeable group of students exist who do not have any experience in programming. At the other side, a couple of beginners already are capable to write programs, even using object-oriented principles. Similar observations are made in other STEM-related study programs, such as electrical engineering, or mechanical engineering.

3.6. Conclusion

A comparison of the programming education in the Gymnasium Dresden Plauen and the Dresden University of Applied Science reveals differences with regard on quantity and technical depth. As one can expect from a university study program, the workload in the field of programming is much higher and much more focused on technical details. Thus, the university study program is not simply a continuation of learning in the specific area. It is change with a quickly increasing learning curve within the first two years at the university.

A gap between school education and university is present for those students who did not engage in the area of their future subject of study.

From this perspective helpful activities would be

- to provide a variety of interest-catching projects and supporting activities in the school
- to support the computer science courses at school with well-prepared material and examples
- and to motivate the future university students to go ahead with programming activities early, even during upper classes in school.

4. Serbia

4.1. Introduction

Within the vertical analysis, our task is to analyze the teaching process at the Gymnasium Ivanjica and the Faculty of Organizational Science of the University of Belgrade. We analyze teaching subjects that focuses on computer science, algorithmizing and programming. Especially, we focus on the subjects that most represent the objectives of the OOP4FUN project in their content and try to identify possible gaps.

The wave of introducing programming as general education content in primary schools in many countries has been going on for several years. The most famous example is Great Britain, where since the 2014/15 school year computer science is a compulsory subject in schools starting from the youngest age, while in Serbia informatics and computer science become a compulsory subject (starting at 2017/18) in the fifth grade of elementary schools in Serbia.

4.2. Legislative and institutional framework of the education system in the Republic of Serbia

Institutions (state bodies, councils and other organizations) responsible for the education system are:

- The Ministry of Education and the Ministry of Science, Technological Development and Innovation are responsible for planning and ensuring the functioning of the education system, in accordance with the general principles and goals of education and upbringing.
- The National Education Council is an advisory body that has responsibilities regarding the development and improvement of the quality of education and upbringing for preschool, primary and secondary general and artistic education and upbringing, especially in the domain of giving opinions on standards of achievement in general education, programs of preschool and general primary and of secondary education, standards of competence of teachers and principals, standards of quality of work of the institution.
- The Council for Vocational Education and Adult Education is an advisory body that has responsibilities regarding the development and improvement of the quality of secondary vocational education and upbringing as well as adult education.
- The National Council for Higher Education is an advisory body that has the authority to monitor the development of higher education and its compliance with European and international standards.
- The Council for the National Qualifications Framework of the Republic of Serbia is an advisory body whose tasks are focused on giving recommendations on the process of planning and developing human potential in accordance with public policies in the field of lifelong learning, employment, career guidance and counseling.

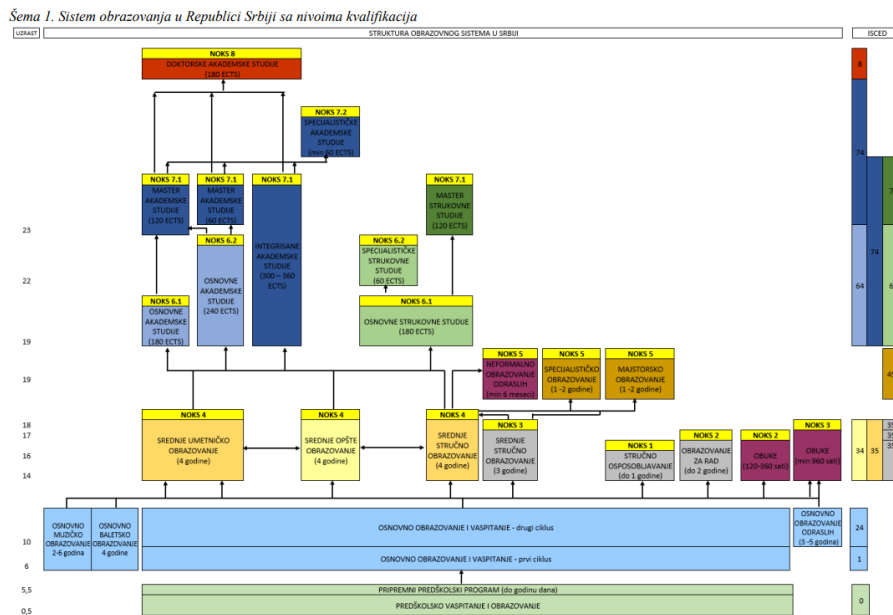
- The Institute for the Improvement of Education and Upbringing is a professional institution that carries out work on the development of teaching and learning programs in primary and secondary education, quality control of textbooks and additional teaching aids, improvement and development of competency standards for the profession of teachers and principals, preparation of programs and exams for the license for teachers and principals, approving programs and other forms of continuous professional development of teachers and principals.
- The Institute for the Evaluation of the Quality of Education and Upbringing is a professional institution that performs tasks on the evaluation of the quality of education, which primarily relate to the development of achievement standards by levels and types of education, the national exam in general education and the evaluation of the quality of the work of institutions.
- The National Body for Accreditation and Quality Control in Higher Education is an independent body that has the authority to ensure the quality of higher education through the implementation of the accreditation procedure and internal quality control of higher education institutions and study programs.
- The Agency for Qualifications has jurisdiction over the development of qualification standards, the recognition of foreign school and higher education documents, and the accreditation of adult education organizations.
- The sector council is an expert and advisory body based on the principle of social partnership, whose main role is to carry out activities on specific qualifications in a certain sector that are acquired in secondary, vocational education, higher education and adult education.

The education system of the Republic of Serbia is governed by the following laws:

- Law on the Fundamentals of Education System ("Official Gazette of RS", no. 88/17, 27/18 - other Law, 27/18 - other Law and 10/19);
- Law on preschool upbringing and education ("Official Gazette of the RS", no. 18/10, 101/17, 113/17 - other laws and 10/19);
- Law on Basic Education ("Official Gazette of RS", no. 55/13, 101/17, 27/18 - other laws and 10/19);
- Law on secondary education and upbringing ("Official Gazette of RS", no. 55/13, 101/17 and 27/18 - other law);
- Law on Dual Education ("Official Gazette of RS", No. 101/17):
- Law on Adult Education ("Official Gazette of RS", no. 55/13, 88/17 - other law and 27/18 - other law);
- Law on Higher Education ("Official Gazette of RS", no. 88/17, 27/18 - other laws, 73/18 and 67/19);
- Law on the Dual Model of Studies in Higher Education ("Official Gazette of RS", No. 66/19);
- Law on the National Framework of Qualifications of the Republic of Serbia ("Official Gazette of the RS", No. 27/18).
- Law on Educational Inspection (Official Gazette of RS, No. 27/18).

Higher education is regulated by the Law on Higher Education, which incorporates the principles of the Bologna Declaration and the Lisbon Convention. The law on the dual study model, which was adopted in September 2019, created a legal basis for the introduction of a dual study model in which part of the study program is realized through learning through work at the employer.

The picture below (Picture 2) shows the education system in the Republic of Serbia with qualification levels.



Picture 2 - The education system in the Republic of Serbia with qualification levels

4.3. Secondary education in the Republic of Serbia

The carriers of secondary education and upbringing in the Republic of Serbia are secondary schools.

Secondary education in the Republic of Serbia consists of:

- general secondary education (gymnasium);
- secondary vocational education;
- secondary music and ballet education

In gymnasiums, the educational process is realized in the social-linguistic and natural-mathematics direction and in general gymnasiums.

There are also specialized gymnasiums for gifted students (philological, mathematical and computing, ie gymnasium for students gifted in physics, biology and chemistry).

In general secondary schools, as well as social secondary secondary schools, there is a Computer Science and Informatics subject that is implemented in the first three years, with a total of 2 hours per week, i.e. 72 hours per year, i.e. a total of 216 hours.

In the Republic of Serbia, there are specialized departments in high schools all over the country, which cover a wide range of different areas of student interest. The students of these departments, in addition to general education that provides various opportunities to continue their education, i.e. to choose higher education, study specific subjects in selected areas, whose teaching and learning plans and programs are made for the needs of this type of education, and subject teachers, with strengthened digital competences, are additionally trained for high-quality implementation of classes

and exercises in specific areas. In the 2021/2022 school year, the total number of students enrolled in these specialized departments is 3130. More than 800 students attend classes in specialized IT departments and thereby increase their competence in computing and informatics.

In 2020/2021 in the Republic of Serbia, a total of 49 schools enrolled students in specialized IT departments. Specialized subjects are studied in different classes. Students have a total of 934 hours of IT subjects in four years. This number of classes, the contents studied, the division of classes into smaller groups, professional teachers and equipped schools represent conditions for high school graduates that will be extremely effective during further studies in IT majors of various faculties. Professional subjects that students have are:

- Programming (in the first three grades, with a fund of 3 hours a week in the first and second grade and an additional 30 hours of teaching in the block and a fund of 2 hours a week in the 3rd grade and an additional 30 hours of teaching in the block)
- Use of computers (in the first three grades, with a fund of 3 hours per week in the first grade and a fund of 2 hours per week in the 2nd and 3rd grades)
- Object-oriented programming (in the third grade, with 3 hours per week, 105 hours per year and an additional 30 hours of teaching in the block)
- Databases (in the third and fourth grades, with a fund of 2 hours a week in the third grade, i.e. 70 in total and 2 hours a week in the 4th grade and an additional 30 hours of teaching in the block)
- Program paradigms (in the fourth grade, with a fund of 2 hours a week and 30 hours of teaching in a block)
- Web programming (in the fourth grade, with a fund of 2 hours per week)
- Computer systems (in the first grade with a fund of 2 hours per week, i.e. 72 hours per year)
- Operating systems and computer networks (in the second year, with a fund of 2 hours per week)

In the subject Programming in the first year, thematic units are covered such as: 1. Concept and examples of algorithms; 2. Basic concepts of programming languages and program development environments; 3. Basic algorithms of linear and conditional structure; 4. Basic algorithms of loop structure; 5. Detailed overview of basic data types (variables, constants, operators and expressions); 6. Arrays, strings and basic algorithms for working with them. In the second year: 1. Multidimensional arrays, matrices and basic algorithms for working with them; 2. User-defined types; 3. Input and program output; 4. Analysis of algorithms; 5. General techniques of algorithm construction; 6. Dynamic data structures and abstract data types. In the third year, topics related to: 1. Graphs and algorithms for working with graphs (20 hours); 2. Text algorithms (10 hours); 3. Geometric algorithms (10 hours); 4. Number theory algorithms (10 hours); 5. Algorithms over bits (4 hours); 6. Overview of selected data structures and algorithms (16 hours).

Object-oriented programming is a subject in the 3rd year. The goal of this course is to acquire basic knowledge about the object-oriented paradigm and its application in solving practical problems. In this course, students learn the basic concepts of object-oriented programming such as: class, object, encapsulation, associations between classes, polymorphism, abstract classes, generic classes and exception handling. Through this subject, students are trained to understand how to solve practical

problems by applying the object-oriented paradigm. In this subject, students solve practical problems using specific object-oriented programming languages.

4.4. Higher education in the Republic of Serbia

The activities of higher education are carried out by the following higher education institutions:

1. University;
2. The Faculty, i.e. the Academy of Arts, within the university;
3. Academy of Vocational Studies;
4. College;
5. College of Vocational Studies

By type, studies are divided into academic and vocational:

- academic studies train students for the development and application of scientific, artistic and professional achievements,
- vocational studies for the application and development of professional knowledge and skills required for inclusion in the work process.

By degree, studies are divided into first, second and third degree studies.

- First degree studies:
 - basic academic studies lasting 3 or 4 years (180-240 ECTS);
 - basic vocational studies lasting 3 years (180 ECTS);
 - specialist vocational studies lasting 1 year (60 ECTS) after vocational or academic studies lasting 3 years (180 ECTS).

Second degree studies:

- integrated academic studies lasting 5 or 6 years (300-360 ECTS);
- master academic studies lasting 1 year (60 ESPB) after basic academic studies lasting 4 years (240 ESPB), i.e. master academic studies lasting 2 years (120 ESPB) after basic academic studies lasting 3 years (180 ECHR);
- master's vocational studies lasting 2 years (120 ECTS) after basic academic or basic vocational studies lasting 3 years (180 ECTS);
- specialist academic studies lasting 1 year (60 ECTS) after master's academic studies.

Third degree studies are doctoral academic studies lasting 3 years (180 ECTS) after integrated academic studies lasting at least 5 years (300 ECTS) or master's academic studies.

4.5. Competence analysis and knowledge analysis of first-year students

In order to determine the knowledge level of first-year students acquired in their previous education, a survey was conducted at the University of Belgrade - Faculty of Organizational Sciences. The survey was conducted using the Google Forms tool and 113 first-year students (i.e. freshmen) participated in it. The survey was not anonymous. In addition to open-ended questions, the questionnaire also contained closed-ended questions related to the verification of acquired knowledge in programming.

The students answered the questions to start with concepts related to programming and OOP, and after that they had tasks based on which it can be determined whether they know the mentioned concepts or not. Now those results need to be cross-referenced.

If the respondents are analyzed according to the type of secondary school completed, the following structure is obtained:

- Gymnasium - classes for students with special abilities for computing and informatics - 12 respondents (11%).
- Gymnasium (general, social or natural-mathematics major) - 78 respondents (69%).
- Vocational school from the technical or IT sector - 19 respondents (17%).
- Vocational school that is not from the technical or IT sector - 4 respondents (3%).

The structure of respondents according to the type of secondary school completed is shown in Chart 1.

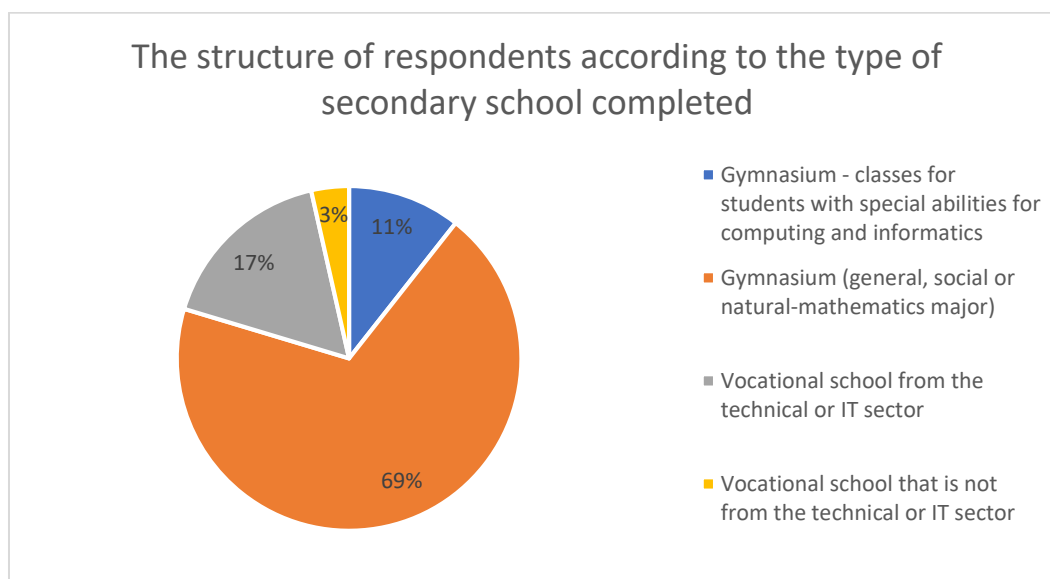


Chart 1 - The structure of respondents according to the type of secondary school completed

Based on the structure of the respondents, it can be noticed that most of them graduated from secondary school of the type "High school (general, social or natural-mathematical major)" - 78 respondents (69%), followed by "Vocational school from the technical or IT sector" - 19 respondents (17 %) and Gymnasium - classes for students with special abilities for computing and informatics - 12

respondents (11%), while the fewest respondents graduated from high school that was not from the technical or IT sector (4 respondents - 3%).

The following observation refers to the number of years in high school in which students attended courses with IT or related content, that is, with programming content. The results are shown in Table 1.

Table 1 - Number of years in high school in which students attended subjects with IT or related content, i.e. with programming content

High school type	Number of years in high school students attended courses with IT or related content	Number of years in which students attended courses with programming content in high school
<i>Gymnasium - classes for students with special abilities for computing and informatics</i>	4 years – 12 respondents (all respondents)	4 years – 12 respondents (all respondents)
<i>Gymnasium (general, social or natural-mathematical course)</i>	1 year – 2 respondents, 2 years – 1 respondent, 4 years – 75 respondents	1 year – 28 respondents, 2 years – 30 respondents, 3 years – 8 respondents, 4 years – 12 respondents
<i>Vocational school in the technical and IT sector</i>	1 year – 1 respondent, 2 years – 2 respondents, 4 years – 15 respondents, 5 years – 1 respondent	1 year – 3 respondents, 2 years – 1 respondent, 3 years – 2 respondents, 4 years – 13 respondents
<i>Vocational school that is not from the technical and IT sector</i>	1 year – 2 respondents, 3 years – 2 respondents	1 year – 4 respondents

All respondents who graduated from secondary school of the type "*Gymnasium - classes for students with special abilities for computing and informatics*" attended informatics subjects for four years, i.e. **entire high school education**. Also, these respondents attended courses related to programming **for all four years**.

On the other hand, the largest number of respondents who graduated from secondary school of the type "*Gymnasium (general, social or natural-mathematical major)*" (75 out of 79 respondents) **attended IT subjects for four years**, while the largest number of them attended programming subjects **for one year** (28 respondents) **or two years** (30 respondents). Only 12 respondents from this group attended programming courses for four years (ie the entire high school education).

When it comes to respondents who graduated from secondary school of the type "*Vocational school from the technical or IT sector*", **the largest number of respondents had four years of IT and programming subjects** (15 respondents and 13 respondents, respectively).

Finally, when it comes to respondents who graduated from secondary school of the type "**Vocational school that is not from the technical or IT sector**", two respondents each attended IT courses for one or three years, while all four respondents attended programming courses in only one year.

The next question considered the current knowledge of programming and its relation with the contents that were covered in school or were covered by the respondents themselves. The results are shown in Chart 2.

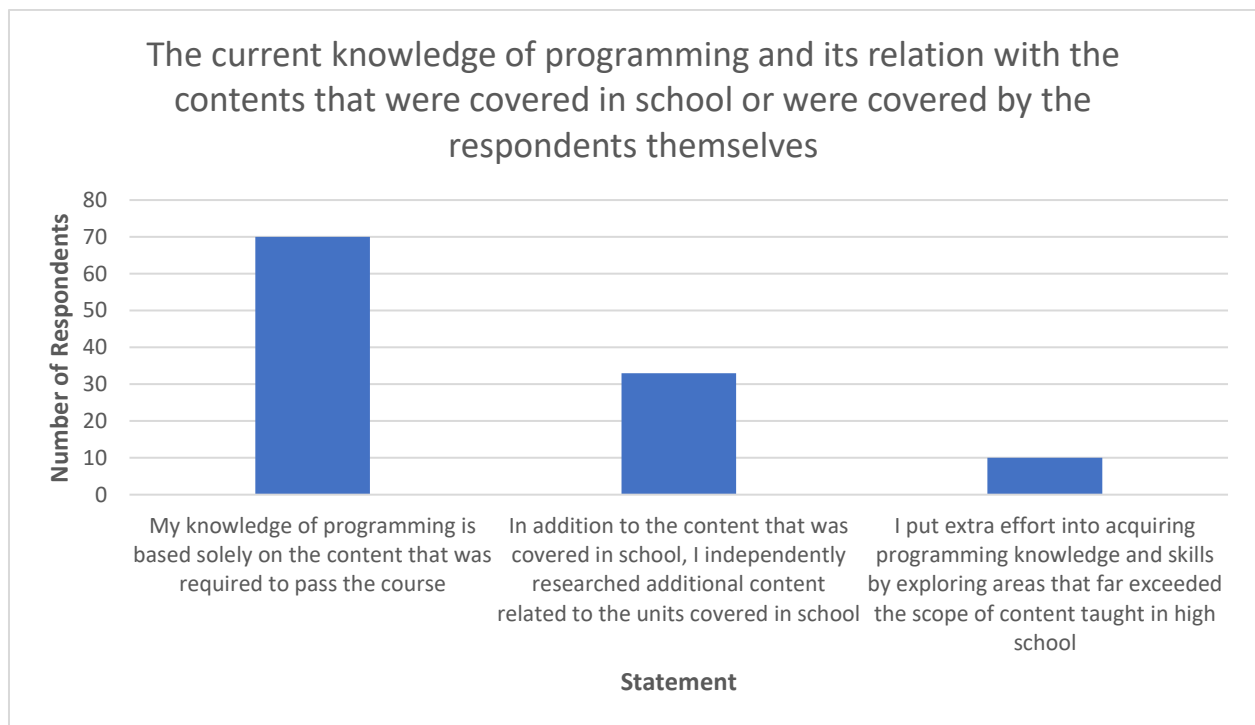


Chart 2 - The current knowledge of programming and its relation with the contents that were covered in school or were covered by the respondents themselves

The largest number of respondents (70 respondents, i.e. 62%) believe that their knowledge of programming is based solely on the content that was required to pass the course. Also, 33 respondents (29%) independently researched additional content in addition to the units covered in school. Only 10 (9%) of respondents invested extra effort in acquiring knowledge and skills in programming by exploring areas that significantly exceeded the scope of content from high school.

When it comes to the areas of programming and problem solving that the respondents know, that is, that they studied during high school education and gained practical experience, the following stand out:

- Algorithmic structures (sequence, selection, iteration),
- Algorithms (for searching, sorting, cryptographic algorithms),
- Programming at multiple levels of abstraction,
- Pseudo-language and pseudocode,
- Data types and variables,
- Functions,
- Recursion,
- Object-oriented programming,
- Concurrent programming,
- Network programming,

- Working with lists,
- Working with strings,
- Working with files (i.e. working with input/output),
- Working with databases,
- Graphical capabilities,
- Flow chart,
- Graphs,
- Stack.

The responses are very different: there are respondents who indicated knowledge of all the previously mentioned concepts, there are respondents who indicated knowledge of only basic concepts (e.g. "Algorithms"), as well as respondents who indicated knowledge of advanced concepts (e.g. "Graphic capabilities", "Working with lists", "Working with files") without mentioning the basic concepts.

The following observation refers to programming languages in which respondents can independently apply basic concepts (e.g. application of basic algorithmic structures, work with variables, use of basic functions, work with input and output). Each respondent could name several programming languages, the results are shown in Chart 3.

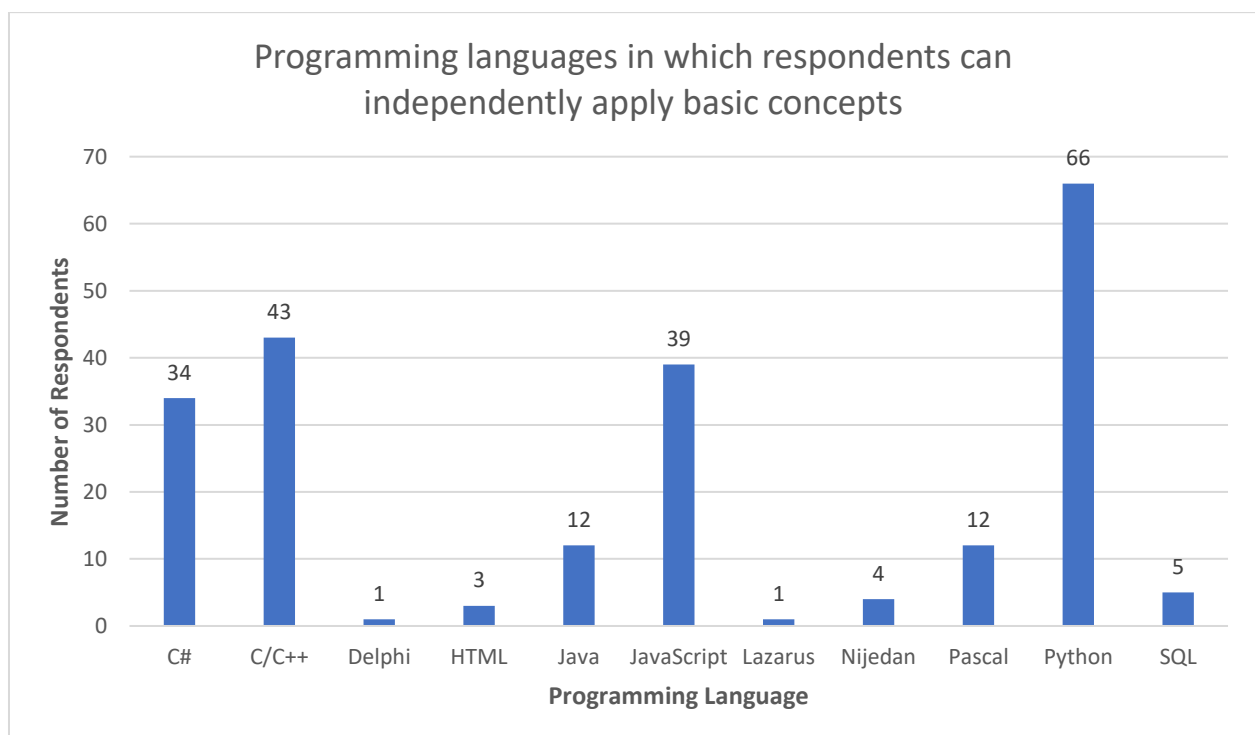


Chart 3 - Programming languages in which respondents can independently apply basic concepts

The largest number of respondents (66 respondents) know the programming language Python, followed by the languages C/C++ (43 respondents), JavaScript (39 respondents) and C# (34 respondents). The programming languages Java and Pascal are known by 12 respondents. Five respondents know the query language SQL, while four respondents do not know any programming language. The HTML language is known by one respondent, while Delphi and Lazarus languages are

known by one respondent each (these languages actually use the syntax of the programming language Pascal).

Given that respondents could name several programming languages, the following observations can be made:

- Knowledge of only one programming language was stated by 44 respondents,
- Knowledge of two programming languages was stated by 32 respondents,
- Knowledge of three programming languages was stated by 17 respondents,
- Knowledge of four or more programming languages was stated by 14 respondents,
- Four respondents do not know any programming language,
- The answers of two respondents are not valid.

The research considered the participation of students in teamwork, which implies a form of coordinated activity of the participants in order to complete the tasks. Chart 4 shows the structure of the respondent's response to teamwork during classes in secondary schools.

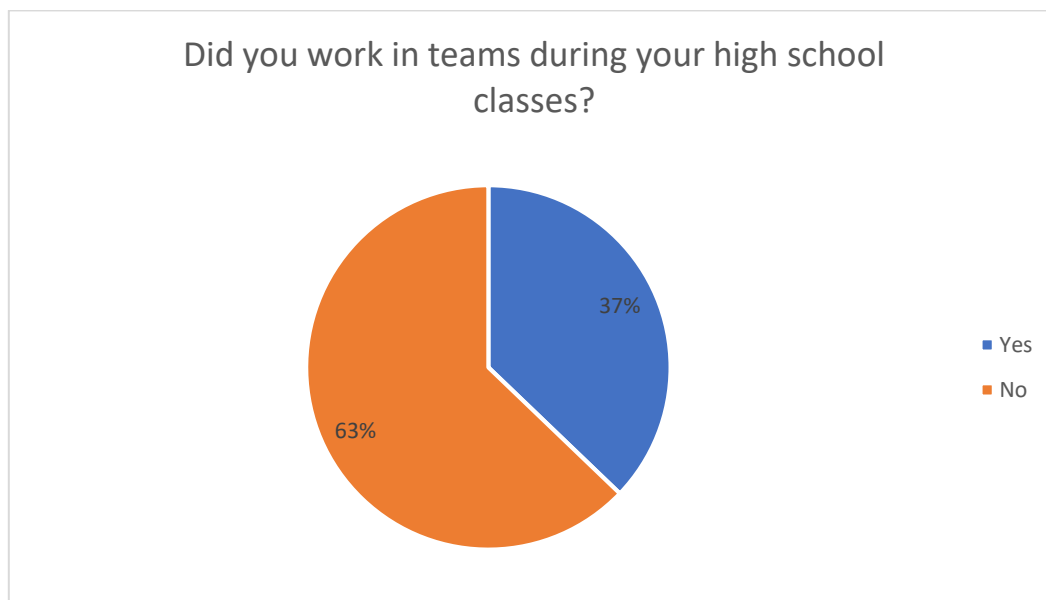


Chart 4 - The structure of the respondent's response to teamwork during classes in secondary schools

It can be concluded that 63% of respondents (i.e. 71 respondents) have no experience related to teamwork, while 37% of respondents (i.e. 42 respondents) have such experiences.

Also, respondents were asked a question related to work on projects or project activities in classes during secondary school education (e.g. independent/team development of a complete and more complex program that has a precisely defined goal and deadline for development and for which documentation is created and presented upon completion of the project). The structure of the responses is shown in Chart 5.

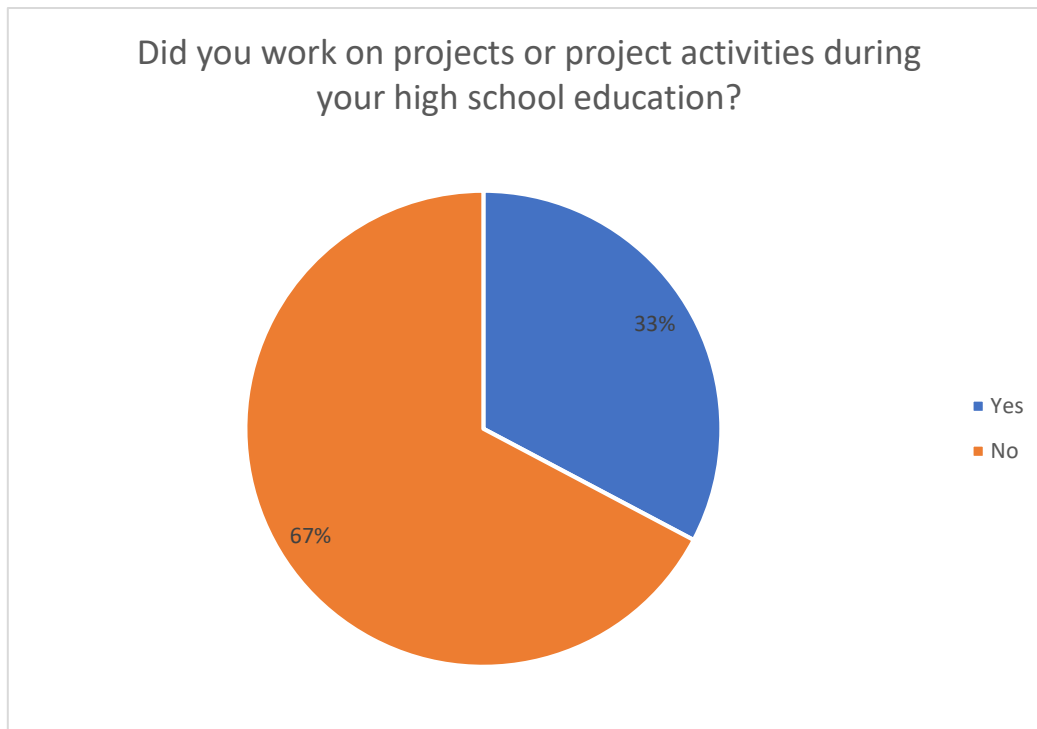


Chart 5 - The structure of the responses related to projects or project activities during your high school education

Similar to the previous question, it can be concluded that 67% of respondents (i.e. 76 respondents) have no experience related to working on projects or project activities, while 33% of respondents (i.e. 37 respondents) have such experiences.

The second part of the questionnaire tried to determine the students' foreknowledge and its relation to the high schools which the students attended. This part of the questionnaire consists of two groups of questions. The first group was related to the basics of programming, while the second group was related to the knowledge of the concepts of object-oriented programming.

First part: Basics of programming

This group consisted of nine questions – five questions were about the knowledge of the basic terms (algorithm, the basic algorithmic procedures, global and local variables, recursion and sorting algorithms), while, in four of them, a snippet of code was given, for which the students had to determine what the outcome will be after the code was executed. Some questions had multiple answers, while one item tried to determine which sorting algorithms are students familiar with and if they are familiar with the concept of sorting algorithms at all.

In order to measure the students' knowledge accordingly, each correct answer within a question had value of one point, regardless of how many correct answers that question had. Maximum number of points (correct answers) for this part was 14.

The distribution of the points is shown at Chart 6.

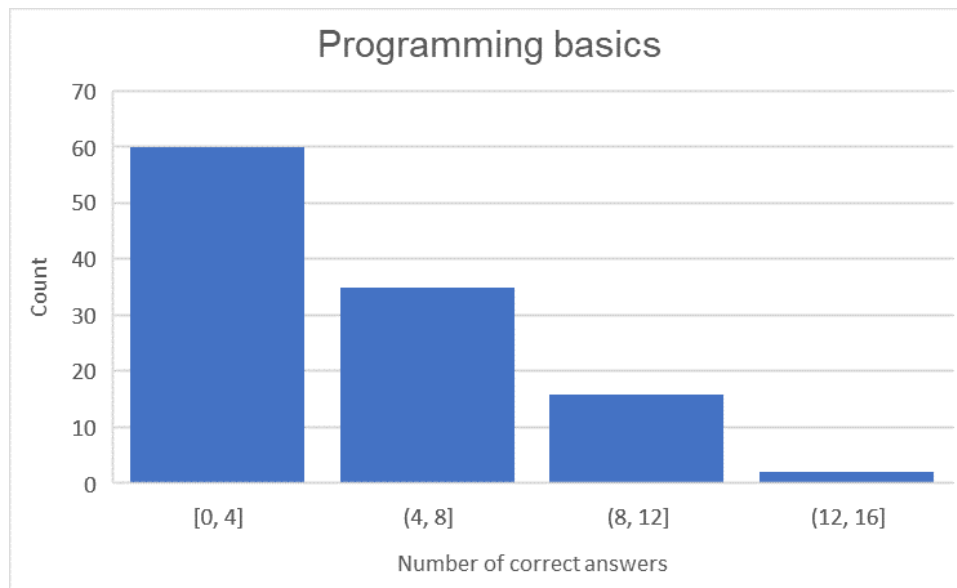


Chart 6 - The average number of points by the types of high schools

The average number of points is 4.78.

The majority of the students (60) had less than 4 points, while 27% (31 student) had more than 50% of the correct answers.

The average number of points grouped by the types of high schools which students attended is shown at Table 2. The group of students who had the highest average number of points are the ones that attended specialized IT class in gymnasium.

Table 2 - The distribution of the points for Programming basics questions grouped by the types of high schools

Row Labels	Average of Programming Basics	StdDev of Total Programming Basics
Gymnasium (IT class)	9,666666667	1,66969422
Gymnasium (general, social studies, and natural sciences and mathematics classes)	3,58974359	2,788826954
Vocational school in the technical or IT sector	7,105263158	3,071277998
Vocational school which are not in the technical or IT sector	2,25	1,258305739
Grand Total	4,778761062	3,445324247

The students which have previously attended vocational studies related to IT and IT gymnasiums had the highest average number of points, compared to the students which have attended high schools and classes which weren't related to IT. Number of correct answers for each question is shown at Table 3.

While the most of the students were familiar with the term of algorithm, the majority were not so familiar with algorithm procedures. Many students were familiar with the terms of global and local variables.

Each row in Table 3 represents a correct answer. For brevity, each answer is presented with the subject of the question. As mentioned before, in four cases, students were given pieces of code for which they should determine the outcome.

Table 3 - Total number of right answers in Programming Basics

1. [Algorithm]	93
2. [Iteration]	19
3. [Sequence]	11
4. [Selection]	17
5. [Global variables]	72
6. [Local variables]	66
7. [Local variable names]	37
8. Code	57
9. Code	37
10. Code	24
11. Code	30
12. [Recursion]	44
13. [Memory and time needed for recursion]	5
14. [Stopping recursion]	28

Students were asked about their knowledge of sorting algorithms and also with which sorting algorithms are they familiar with. Results are shown in Table 4. From the results we can see that the majority of the students are not familiar with any sorting algorithms and are not acquainted with the concept of sorting algorithms. However, those students which are familiar with sorting algorithms are mostly acquainted with the selection sort and the bubble sort algorithms.

Table 4 - Students' knowledge of sorting algorithms

Selection sort	43
Iteration sort	14
Bubble sort	33
Insertion sort	19
Separation sort	8
None of the above	6
I don't know what the sorting algorithms are.	53

Second part: OOP

This group of questions consisted of seven questions regarding basic terms of object-oriented programming, including classes, objects, inheritance, encapsulation, polymorphism, abstraction, modifiers, etc.

The total number of correct answers is 14.

The distribution of points for knowledge of basic terms in object-oriented programming is shown in Chart 7. As it is shown, the majority of the students (83) had 4 or less points. Of 113 students, 22 had 8 or more points in this group of the questions.

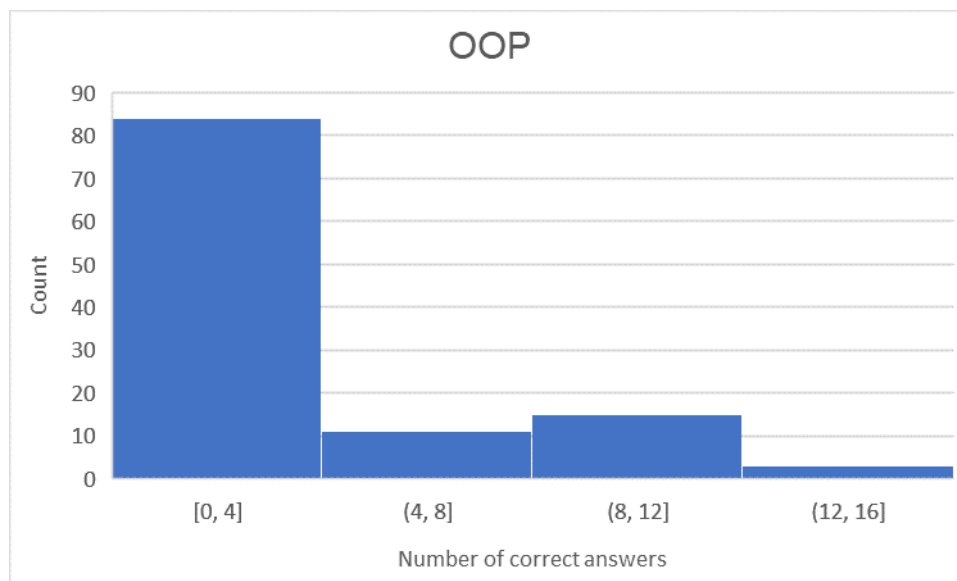


Chart 7 - Distribution of knowledge for OOP

The average number of points for students grouped by their high schools is shown at Table 5.

Table 5 - Average number of correct answers in OOP

High schools	Average of Total OOP	StdDev of Total OOP
Gymnasium (IT class)	8,25	3,519426606
Gymnasium (general, social studies, and natural sciences and mathematics classes)	0,948717949	2,667249354
Vocational school in the technical or IT sector	5,947368421	5,522415778
Vocational school which are not in the technical or IT sector	0	0
Grand Total	2,530973451	4,297156029

Table 6 shows the number of total correct answers given for each correct answer in a question. As in previous case, each answer is presented with the subject of the question to whom it belongs.

Table 6 - Total number of correct answers in OOP

[Objects]	22
[Objects]	23
[Objects]	20
[Access modifiers]	22
[Objects]	10
[Objects]	27
[Constructor]	19
[Object]	19
[Class]	28
[Abstraction]	15
[Inheritance]	26
[Polymorphism]	13
[Encapsulation]	18
[Class]	24

Summary

The total number of points, which consists of correct answers from Programming Basics and OOP groups of questions, are displayed in Chart 8. The majority of the students (92/133) had less than half (16 of 32) of total correct answers.

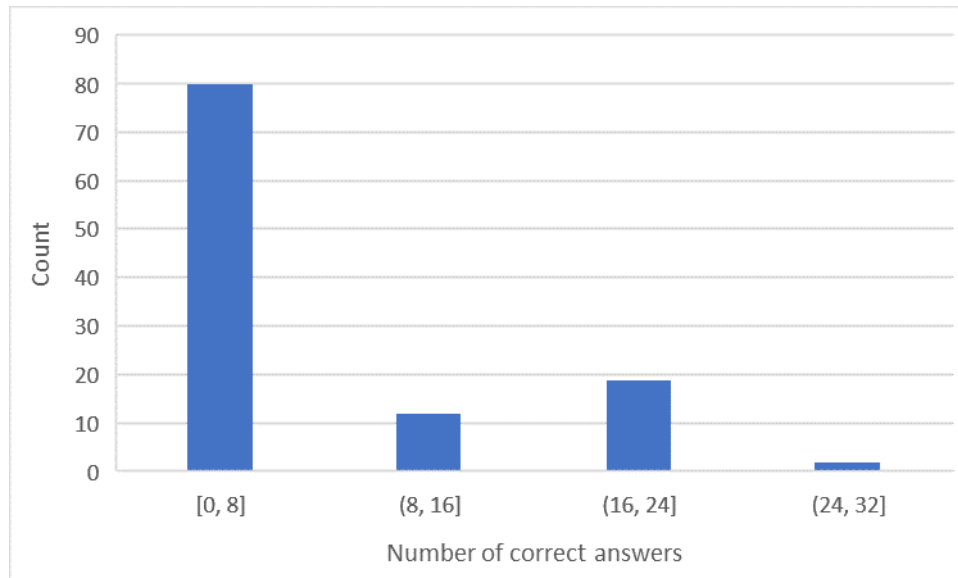


Chart 8 - Total number of points in both groups of questions

The students were asked to estimate their programming knowledge at the scale from 1 to 10, where 1 represents the most basic knowledge, and 10 represents the ability to solve complex programming problems. In Table 7, the average number of points is displayed, alongside with the students' estimation of their own knowledge, shown in Chart 9.

Table 7 - Students' estimations and average number of points

Estimation of knowledge	Average of Total
1	1,807692308
2	3,375
3	3,842105263
4	7,8
5	9,933333333
6	13,5
7	15,44444444
8	17,375
9	19,5
Grand Total	7,309734513

On average, the students have correctly answered 34% of the questions regarding the programming basics and 17% of the questions regarding OOP. On average, the students have correctly answered 25% of all of the questions including both the basics and OOP. As was expected, the students which have attended high schools with IT classes, or technical vocational schools related to IT had more correct answers compared to the students who attended high schools not related to IT.

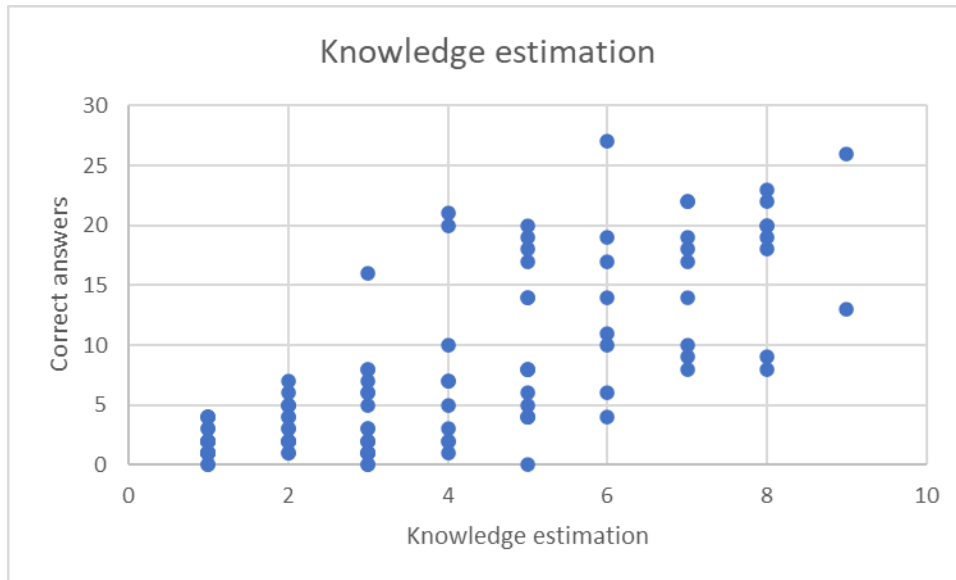


Chart 9 - The students' knowledge estimation

5. Croatia

5.1. Introduction

In the Republic of Croatia, the Croatian Qualifications Framework (HKO) is applied as an instrument for applying the European Qualifications Framework to the entire education system. It also defines all levels of education in Croatia. The levels of education are clearly defined and they range from elementary school to a doctorate. It is also known exactly which qualifications are acquired upon completion of a particular level of education, transitions from a lower to a higher level are facilitated, as well as inclusion in international educational programs.

Levels of education that exist in Croatian education system are:

- elementary education
- vocational training
- one-year and two-year high school vocational education
- three-year vocational education
- grammar school education
- four-year and five-year high school vocational education
- professional studies with the completion of less than 180 ECTS points
- university undergraduate studies, professional undergraduate studies
- university graduate studies, specialist graduate professional studies, postgraduate specialist studies
- postgraduate scientific master's studies
- postgraduate university (doctoral) studies

Unfortunately, there are sometimes inconsistencies between the levels of education in terms of knowledge and skills acquired at a certain level of education, which are a prerequisite for continuing education at a higher level. This can create certain problems in the continuation of the education of individuals, as well as difficulties in the implementation of the teaching process due to the 'gaps' that occur due to the aforementioned inconsistencies.

In this analysis, emphasis will be placed on the differences in education at the high school level and university level, in terms of content related to object-oriented programming. The legislative framework and basic documents within which education is carried out will be considered, the prior knowledge of students in the first year of university courses and the teachers' expectations will be also analyzed, which will finally result in identification of gaps between those two levels, in aspect of object oriented programming topics.

5.2. Legislative framework

There are several fundamental documents that represent the basis of the high school education and university education systems in Croatia. For the purposes of this analysis, the following legal acts will be analyzed in particular:

1. Law on the Croatian qualification framework
2. National qualification and occupation standards
3. National curriculum for preschool, elementary and high school education
4. Curriculum for informatics subject in elementary and grammar schools
5. Rulebook on taking the state matura exam
6. Curricula for programming related subjects in university studies

5.2.1. Law on the Croatian qualification framework

The Croatian qualification framework (HKO) is a reform instrument that regulates the entire system of qualifications at all educational levels in the Republic of Croatia through qualification standards based on learning outcomes and harmonized with the needs of the labor market, the individuals and society as a whole.

The Law on the Croatian Qualification Framework was adopted by the Croatian Parliament at its session on February 8, 2013. The last amendment to the law was made in 2021.

In accordance with the Law, the following principles and goals of HKO are distinguished:

- ensuring the conditions for quality education and learning in accordance with the needs of personal, social and economic development, social inclusion, and the abolition of all forms of discrimination,
- development of personal and social responsibility and application of democratic principles in respect of fundamental freedoms and rights and human dignity,
- strengthening the role of key competencies for lifelong learning,
- developing qualifications based on clearly defined learning outcomes,
- understanding of different qualifications and learning outcomes and their interrelationships,
- ensuring conditions for equal access to education throughout life, for multidirectional horizontal and vertical mobility, acquisition and recognition of qualifications,
- ensuring economic growth based on scientific and technological development,
- strengthening the competitiveness of the Croatian economy, which is based on human resources,
- achieving employability, individual and economic competitiveness and coordinated social development based on education,
- establishment of a coordinated quality assurance system for existing and new qualifications,
- building a system of recognition and evaluation of non-formal and informal learning,
- establishment and sustainable development of partnership between holders and stakeholders of the qualification system,

- ease of recognition and recognition of foreign qualifications in the Republic of Croatia and Croatian qualifications abroad,
- participation in the process of European integration while respecting the guidelines given by EQF and QF-EHEA, European Union guidelines and international regulations,
- preservation of the positive heritage of the Croatian educational tradition,
- improvement and promotion of education in the Republic of Croatia.

The Minister of Science and Education with the consent of the Minister of Labor and the Pension System, the Minister of the Economy, the Minister of Entrepreneurship and Crafts, and the Minister of Regional Development and European Union Funds, issued the Rulebook on the Register of the Croatian Qualification Framework (HKO register), which was published in the Official Gazette, No. 62/2014. , May 22, 2014.

The HKO register is a system in which occupational standards are registered and linked to qualification standards through sets of competences and sets of learning outcomes. All standards from the HKO Register will be publicly available and will serve to develop new educational programs based on learning outcomes, i.e. sets of competencies proven to be needed by the labor market.

The Law on the Croatian qualification framework is available on following link:

<https://www.zakon.hr/z/566/Zakon-o-Hrvatskom-kvalifikacijskom-okviru>

The Croatian qualification framework register is available on following link:

<http://www.kvalifikacije.hr/hr/registar-hko>

5.2.2. National qualification and occupation standards

The occupational standard contains the competencies which are crucial to practicing a certain profession and the qualification standard contains key learning outcomes that must be contained in any program leading to that qualification. The occupational standard is the result of agreement of relevant stakeholders on the labor and education market about the optimal content of a particular profession and about knowledge and skills with the associated independence and responsibility (competencies). They are fortified with key tasks and competencies required for performance of these jobs for a particular occupation. The qualification standard indicates the content and structure of certain qualifications, and includes all data necessary to determine the level, volume and profile qualifications, as well as the information required for ensuring and improving the quality of standards qualifications¹.

¹ source: Methodology for creating occupational standards and sets of competencies, retrieved on October 31, 2022 <http://www.kvalifikacije.hr/sites/default/files/documents-publications/2021-12/Metodologija%20za%20izrada%20standarda%20zanimanja%20i%20skupova%20kompetencija.pdf>

At the time of writing, there are 388 occupation standards and 109 qualification standards defined².

5.2.3. National curriculum for preschool, elementary and high school education

Education in elementary and high schools is based on the National curriculum, subjects' curricula and school curriculum. National curriculum is adopted for individual levels and types of education in accordance with the national curriculum framework document, which determines the elements of the curriculum system for all levels and types of elementary and high school education at the general level. National curriculum and the national curriculum framework document are adopted by the minister responsible for education by decision.

The system of national curriculum documents that make up the complete National Curriculum consists of:

- national curriculum for early and preschool education
- national curriculum for elementary education
- national curriculum for grammar school education
- national curriculum for vocational education
- national curriculum for artistic education
- curriculum areas and curricula of cross-curricular topics
- subject curricula and curricula for obtaining qualifications in the regular system of vocational and artistic education
- a framework for evaluating learning processes and outcomes in the educational system
- a framework for encouraging and adapting learning experiences and valuing the achievements of students with disabilities
- a framework for encouraging learning experiences and evaluating the achievements of talented students³

National curricula are available on following link:

<https://mzo.gov.hr/istaknute-teme/odgoj-i-obrazovanje/nacionalni-kurikulum/nacionalni-kurikulumi/531>

5.2.4. Curriculum for informatics subject in elementary and grammar schools

The last version of The curriculum of the informatics subject was adopted by the Ministry of Science and Education on March 6, 2018 and it consists of the following:

² source: Croatian qualification framework, retrieved on October 31, 2022, <https://hko.srce.hr/registar/standardi>

³ source: National curriculum, Ministry of Science and Education, retrieved on October 31, 2022, <https://mzo.gov.hr/istaknute-teme/odgoj-i-obrazovanje/nacionalni-kurikulum/125>

- description of informatics subject
- educational goals of learning and teaching the curriculum in informatics
- domains in the organization of the informatics subject curriculum
- educational outcomes, elaboration of outcomes, adoption levels and recommendations for the achievement of educational outcomes by classes and domains with a list of literature
- presentation of the annual number of hours and form of implementation of the Informatics subject in elementary schools and high schools
- list of recommended qualifications for Informatics teachers

Curriculum is made for 8 grades of elementary school and informatics is obligatory subject only in 5th and 6th grade while in other grades it is taught as optional subject.

In grammar school, situation is a bit different. Curriculum for grammar school is made for 4 grades and it is divided into three programs:

- general grammar school
- classic and language oriented grammar school
- science and mathematic oriented grammar school

In first two programs (general and classic and language oriented), curriculum for informatics is the same. It means that educational outcomes, elaboration of outcomes, adoption levels and recommendations for the achievement of educational outcomes by grades and domains are the same. The difference is in which grade informatics is taught as obligatory subject. In grammar school, informatics is obligatory in first grade while in other three grades students can choose it as optional subject. That also depends on whether the school even offers informatics as an optional subject in the remaining three grades. In classic and language oriented grammar schools, informatics is taught as obligatory subject in second grade while in others is taught as optional. In all those programs, informatics is taught two school hours a week and 35 weeks in school year, which results in 70 hours a year in total.

In science and mathematic oriented grammar schools, informatics is obligatory subject in all four grades and is taught two hours a week in each school year.

More specific details about learning outcomes and topics will be described later in gap analysis.

Curriculum for informatics subject is available on the following link:

<https://mzo.gov.hr/istaknute-teme/odgoj-i-obrazovanje/nacionalni-kurikulum/predmetni-kurikulumi/informatika/755>

5.2.5. Rulebook on taking the state matura exam

The state matura is a mandatory final written exam that high school students take at the end of their high school education. All grammar high school students are required to take the state matura exam,

while vocational students take the state matura exam only if they plan to continue their education at one of the higher education institutions (universities). The state matura exam can be taken only by vocational students in programs with four-year duration. The state matura examinations are conducted in a standardized manner throughout the state at the same time and under equal conditions and criteria for all students, that is, applicants.

State matura examinations consist of obligatory part exams and elective part exams. Exams of the obligatory part consist of exams from the following subjects:

- Croatian language,
- Mathematics and
- foreign language.

The optional part of the matriculation exam is taken from other subjects that the students attended during their high school education. The number of these subjects is not limited.

As mentioned before, grammar school students must take state matura exams. Successful completion of obligatory subjects also means the successful completion of secondary education. At the moment, all three obligatory subjects can be taken at the basic and advanced level. As part of the reform processes in the education system in Croatia, starting in 2023, the mentioned obligatory subjects will be taken at one common level.

Depending on the faculty, as well as the specific course, the high school graduate must choose the subjects and the level they want to take. If the candidate chooses a level lower than the one required, he will not be able to be listed when enrolling in the desired faculty.

More specific details about topics that are covered by matura exam will be described later in gap analysis.

Rulebook on taking the matura exam is available on the following link:

<https://www.ncvvo.hr/wp-content/uploads/2021/05/Pravilnik-o-polaganju-DM-procisceni-tekst1.pdf>

5.2.6. Curricula for programming related subjects in university studies

The documents on curricula for programming related subject in university studies at Faculty of organization and informatics, University of Zagreb, contain the mandatory information about university courses. This document and information are used for the course and study program approval from accreditation body as well as for students when analyzing study program in terms of the size, content, literature and learning outcomes of the course.

The curricula documents for courses at Faculty of Organization and Informatics of University of Zagreb are available at <https://nastava.foi.hr/>. By choosing the study program visitor would be provided with the options to check the courses and their curricula and for the purpose of this analysis we have been focusing on entry level study programs and particularly on programming courses:

- Programming 1 (Programiranje 1) at university study program (<https://nastava.foi.hr/course/214449/2022-2023>)

- Introduction to programming (Uvod u programiranje) at professional study program (<https://nastava.foi.hr/course/228790/2022-2023/VZ>)

The detailed analysis of these documents would provide us with different sets of data including basic information about the course (such as the course goal and description, study level and year, enrolment status and precondition courses, number of hours for lectures, seminars and laboratory exercises) the information about teachers, the information about course content (such as content of the lectures, content of the seminars and laboratory exercises, learning outcomes of the course, learning outcomes of the study program this course is contributing to, primary and additional literature and similar courses at other universities) and assessment model (assessment elements for full time and part time students, scoring, assessment schedule etc.).

However, although detailed, the curricula for university courses unfortunately does not contain any information on required knowledge which the students should have from their high-school level, but only the list of prerequisite courses that are to be completed prior to enrolling the particular analyzed course. Expectedly, the list on prerequisite courses is empty for the courses which are taught at first semester, which is actually from our interest in this analysis.

Further more, by taking the analysis of the course content, we can conclude that all programming related courses taught at first semester cover both basic (high school level) and advanced (university level) programming concepts.

This all brings us to the conclusion that although containing important information on particular course, the curricula can not provide the information on vertical gap analysis between high-school outcomes and university expectations that we need.

5.2.7. Conclusion on legislative analysis

Given that, at the legislative level, there are no requirements for the mandatory definition of prerequisites for enrollment to universities, which are related to the computer and more specifically programming skills of future university students, teachers of courses from the first year of study do not expect students to have prior computer knowledge. For this reason, teachers at universities include in the curricula of introductory courses also the concepts that should be covered in the computer science and informatics classes in high school. Since there are no defined prerequisites that connect the mentioned two levels of education (learning outcomes at the secondary school level and prior knowledge for enrollment to universities), it was not possible to conduct a gap analysis exclusively through the available prescribed and official documents.

Therefore, we designed our own methodology, which we used to identify the actual knowledge (prior knowledge) of students in the first year of undergraduate studies and compared them with the expectations of teachers teaching first-year courses.

5.3. Gap analysis

The purpose of this analysis was to determine the differences between the output provided by high schools and the input required at the universities, in the aspect of the IT competencies and skills of future university students, with special emphasis on knowledge and skills in the area of object oriented programming. This analysis led to identification of gaps in teaching programming between two mentioned levels of education in Croatia.

The methodology used to collect and analyze data consisted of the following:

1. freshmen students' prior competencies and knowledge analysis – questionnaire for students
2. teachers' expectations – semi-structured interview
3. analysis and comparison of collected data

5.3.1. Freshmen students' prior competencies and knowledge analysis

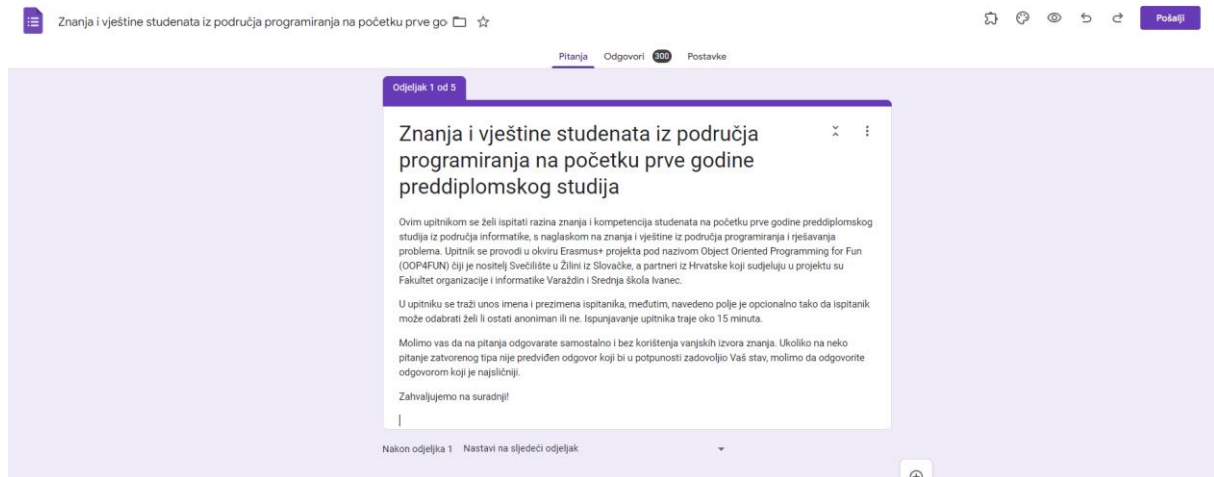
5.3.1.1. Introduction

The target group for this analysis of prior competencies and level of knowledge were freshmen students (first year students) of undergraduate study in the field of computer science at the Faculty of organization and informatics, Varaždin (University of Zagreb). Data were collected through an online questionnaire using Microsoft Forms tool. The students received all the instructions, they were informed about the purpose of conducting the questionnaire and asked to approach the questionnaire as objectively as possible.

The questionnaire was divided into three parts:

1. general information
2. practical programming knowledge and skills
3. object oriented programming knowledge and skills

The questionnaire was anonymous, although students could enter their first and last name if they wanted. It consisted of 30 questions and a total of 300 students filled out the questionnaire. Questionnaire was made in Croatian language. All the respondents were given basic introduction and instructions, which can be seen in Picture 3.



Picture 3 – Introduction to the questionnaire in Croatian language

When translated into English, the introduction to the questionnaire looks like this:

Knowledge and skills of students in the field of programming at the beginning of the first year of undergraduate studies

This questionnaire aims to examine the level of knowledge and competence of students at the beginning of the first year of undergraduate study in the field of computer science, with an emphasis on knowledge and skills in the field of programming and problem solving. The questionnaire is conducted as part of the Erasmus+ project called Object Oriented Programming for Fun (OOP4FUN), led by the University of Žilina from Slovakia, and the Croatian partners participating in the project are Faculty of Organization and Informatics Varaždin and High School Ivanec.

The questionnaire asks for the first and last name of the respondent, however, this field is optional so that the respondent can choose whether he wants to remain anonymous or not. Filling out the questionnaire takes about 15 minutes.

Please answer the questions independently and without using external sources of knowledge. If there is no offered answer to a certain question that would fully satisfy your position, please answer with the most similar answer.

Thank you for your cooperation!

After closing the questionnaire, results were exported in the Excel spreadsheet which is available in the following link:

[Questionnaire-results.xlsx](#)

5.3.1.2. Data analysis

Students filled out a questionnaire within one of the courses they attend, namely Programming 1 and Introduction to programming. The distribution of respondents by subjects can be seen in Chart 10.

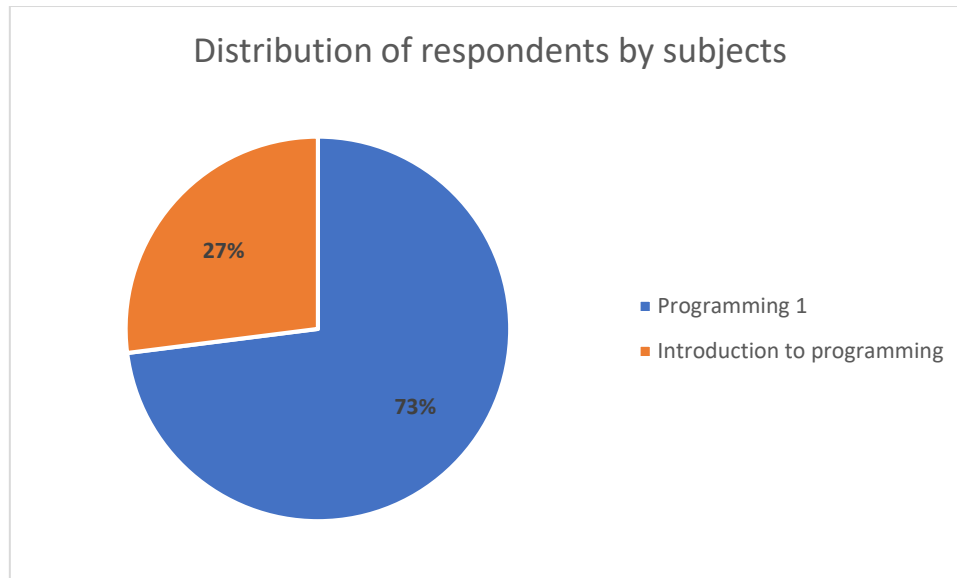


Chart 10 - Distribution of respondents by subjects

As shown on the chart, nearly 3/4 of the students (219 students) filled out the questionnaire within the course Programming 1, while the rest filled it out within the course Introduction to programming. In addition, the analysis shown that almost 2/3 of the students (194 students) wanted to remain anonymous, while around 1/3 of students (106 students) put their names in the questionnaire. This actually has no significance in the analysis, it is only informative, but also could mean that the students, who entered their names, may have answered the questions more precisely and more reliably.

Students who were filling out the questionnaire came from different types of high schools. Some of them are more related to the area of students' future education, some of them less. Distribution of respondents by type of school they are coming from is shown in Chart 11.

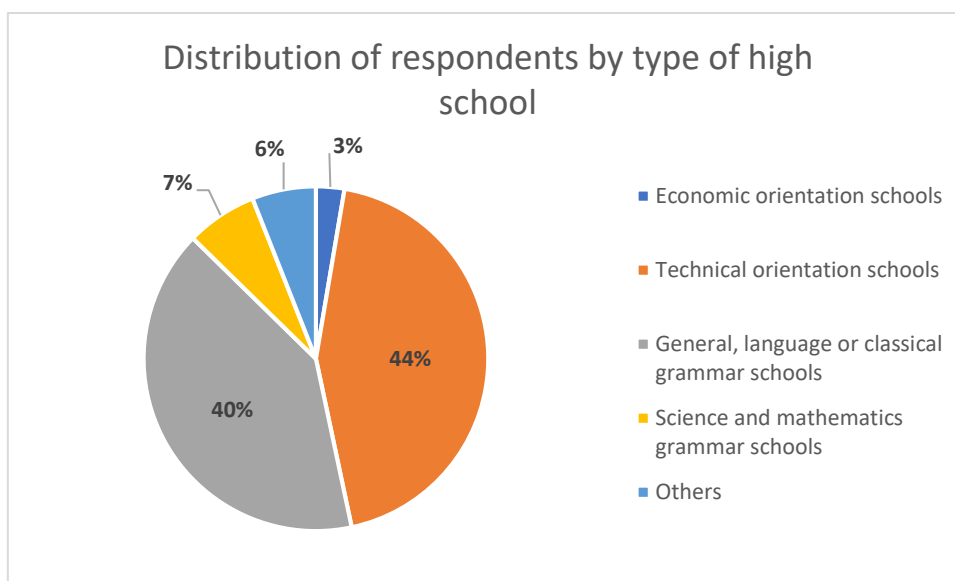


Chart 11 - Distribution of respondents by type of high school

As we can see, the vast majority of respondents came from schools that are technically oriented (44%) and from general, language or classical grammar schools, gymnasiums (40%). Distribution of students between those two types of schools is more or less equal, but we will see later that these two types of schools are very different in terms of object oriented programming knowledge and skills. Then, we had 7% of students coming from science and mathematics grammar schools, 3% from economic schools and 6% of students who filled out the questionnaire came from other types of schools.

After that, students were asked about how many years did they take the Informatics course (or a course with related contents, for example Computing, etc.) in high school. They could choose between 1 and 5 years, because in Croatia there are no high schools, secondary programs or professions whose duration is longer than 5 years. Distribution can be observed in Chart 12.

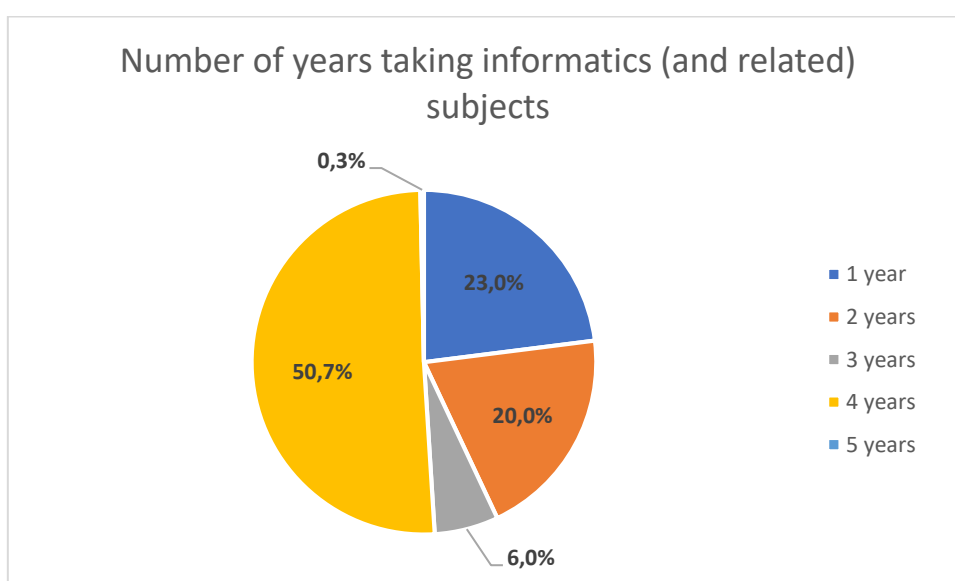


Chart 12 - Number of years taking informatics and related subjects in high school

Almost 51% of the respondents attended informatics subjects for 4 years in their high schools which is quite high number. That means that over half of the students took informatics subjects during their entire high school education, considering that 4-year high school education is most represented (students with occupations which take less than 4 years couldn't take state matura exam and therefore couldn't enroll to the university). On the other hand, almost 1/4 of the students took informatics classes in high school for just 1 year. This is consistent with curricula of different high school programs in which informatics is obligated in just 1 year of education (general, language or classical grammar school and some 4-year vocational programs). That also means that students didn't have the option or didn't want to enroll to informatics as optional subject in higher years of their high school education.

Related to the years of attending informatics subjects, we also asked students to give us information about the years of studying contents with programming topics and results are shown in Chart 13. We can see that almost half of students (42%) encountered programming topics in just 1 year of their high school education. This is in relation with earlier chart where almost 25% of students attended only 1 year of informatics subject in general and the rest of students (who have more than 1 year of informatics) attended programs where programming is implemented in just 1 year of entire informatics curricula). We can see that 19% of students encountered with programming for 2 years, 10% for 3 years and 29% for 4 years of high school education.

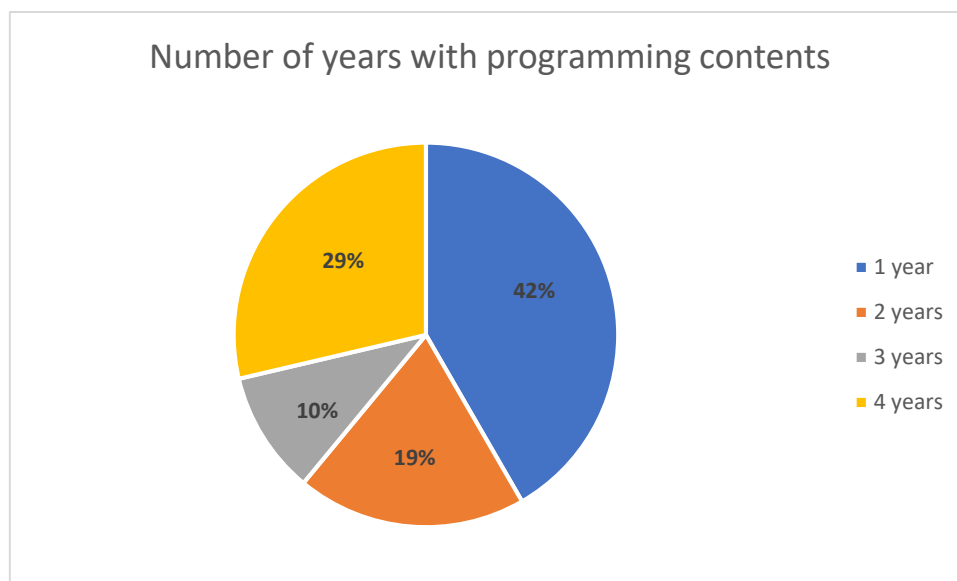


Chart 13 - Number of years with programming contents within informatics subjects in high schools

Besides basic informatics subjects, some students were enrolled to other subjects that were including programming topics in their curricula. Some of those subjects are: Web design, Scripting languages and web programming, Computer networks, CNC technologies, Microprocessors, Microcontrollers, Programming, Databases, Advanced and object oriented programming etc. Despite variety of the other subjects that students were enrolled to, only 28% of them stated that they were attending those subjects, besides informatics.

Regarding the way students were gaining knowledge about programming, results are as follows: 62% of the students stated that their programming knowledge is based solely on the content that was required to pass the exam, 28% of the students stated that, in addition to the content that was covered

at school, they independently researched additional content and only 10% of students put extra effort into acquiring additional programming knowledge and skills by exploring areas that exceeded by far the scope of content taught in high school. This results are displayed in Chart 14.

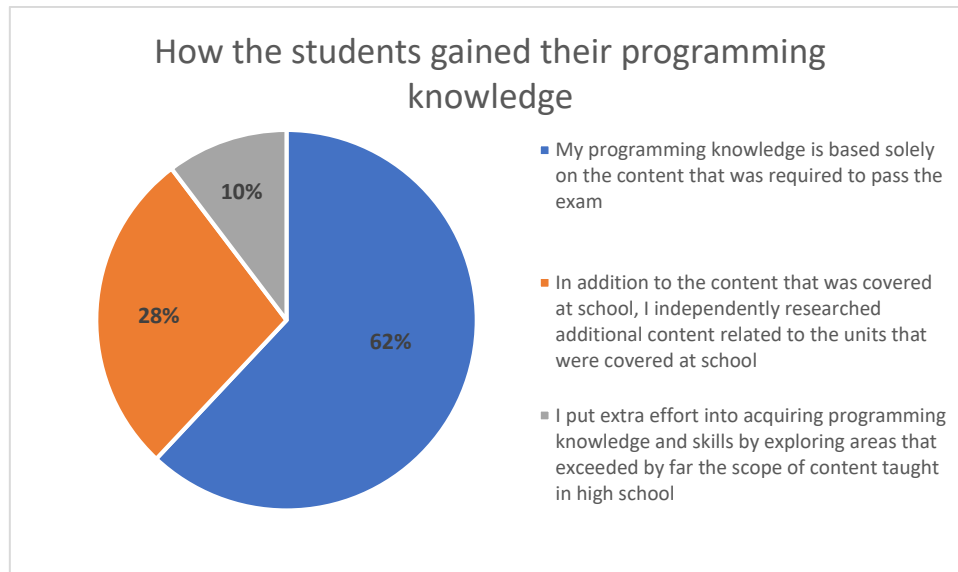


Chart 14 - The way students acquired their programming knowledge

We can conclude that students are not eager for self-learning or very interested in acquiring additional programming skills during their high school education.

After that, students were asked more specific questions about programming, about programming concepts that they recognize and which they have practical experience with. They could choose between 21 different topics and could also add their own answers that were not offered. The question was of the multiple-choice type, so they could choose more than one answer. The results are shown in Chart 15.

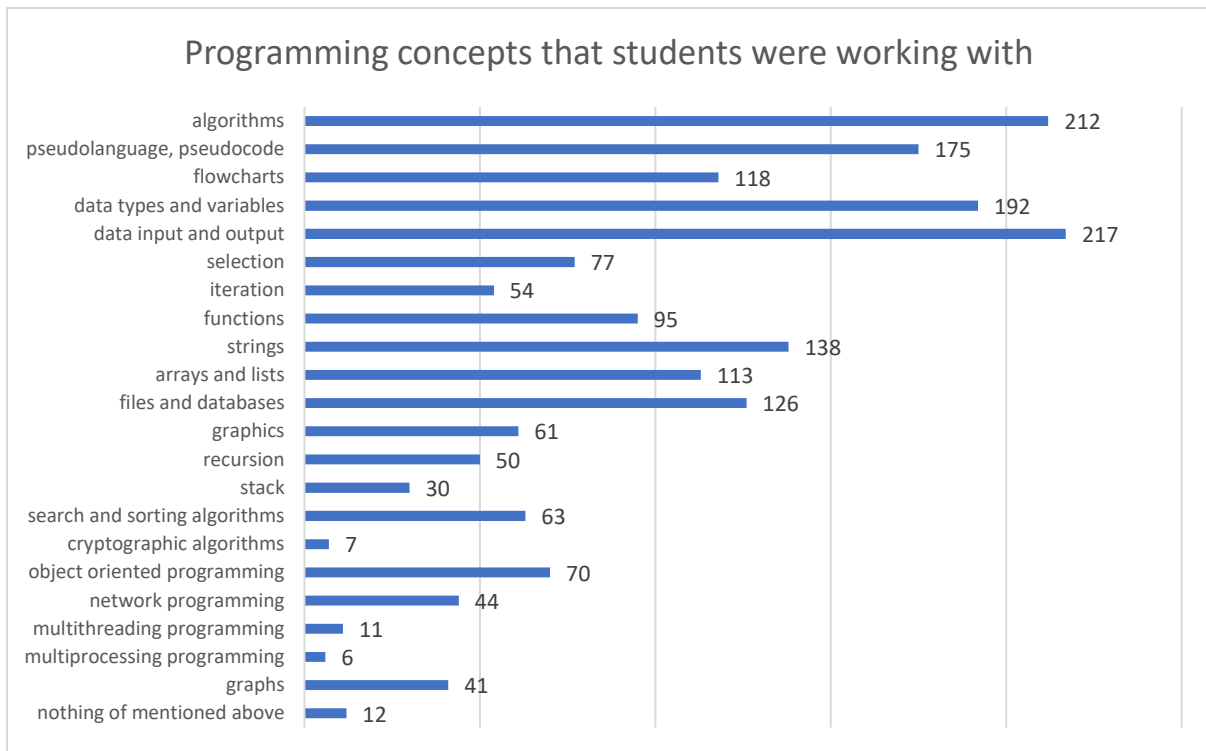


Chart 15 - Programming concepts that high school students are familiar with and have practical experience

We can see that algorithms and basic input and output are concepts that students are most familiar with. This is understandable since these two concepts represent the basics of programming in general. Besides those two, students also have experience with data types and variables, then pseudocode and pseudo language. It is also evident that almost 3/4 of respondents never worked with selections every sixth student worked with iterations. This is a bit unlikely because these two concepts are also representing the basics of programming and they are implemented in informatics curricula of most of high school programs. It is more likely that students didn't recognize those concepts by their names. At the bottom of the list of experiences with programming concepts are concepts like multiprocessing programming, cryptographic algorithms and multithreading programming. That is understandable because those concepts are not concluded into regular curricula of high school subjects, so it is likely that the students who chose these concepts gained experience through independent work. It is also important to mention here that less than 1/4 of the students have experience with object-oriented programming, and also that 12 students (4%) didn't work with any of the following concepts.

It is also interesting to see the results of the usage of different programming languages. These results can be seen in Chart 16.

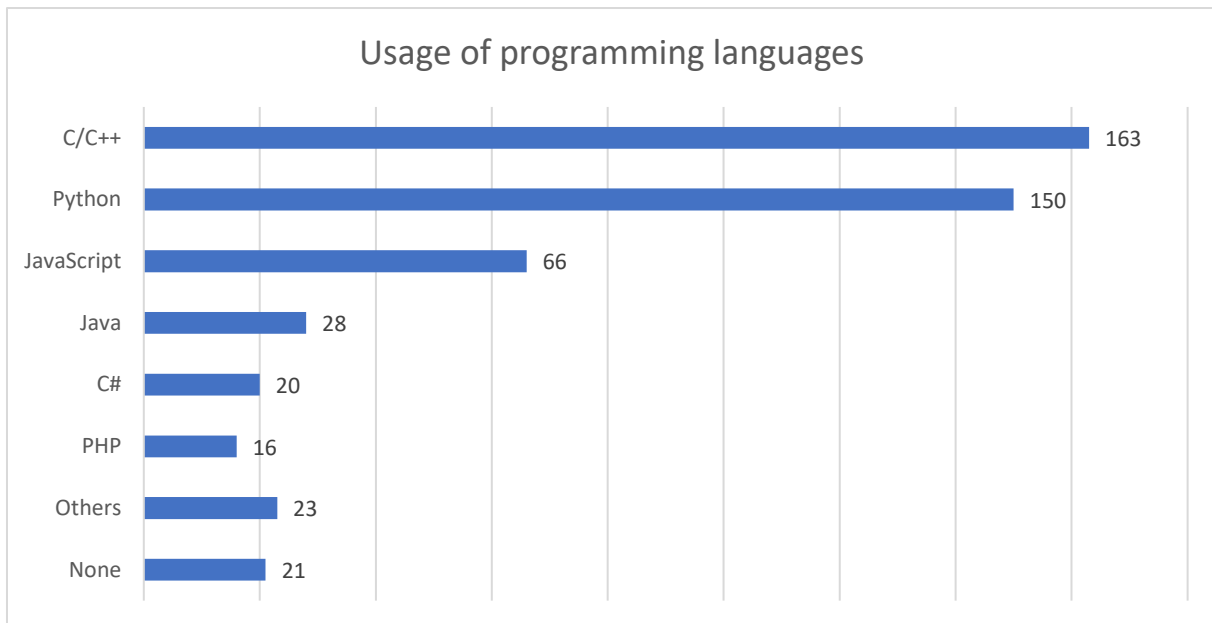


Chart 16 – Usage of programming languages by students in high schools

We can see that more than half of the students worked with C or C++ in high school and also half of the students worked in Python. It is also important to note here that the question was multiple choice, so that one respondent could choose more than one answer. Of the object oriented languages, it is also important to mention Java, in which 28 students worked, and C# with 20 answers. Of the other languages, the most represented are scripting languages and languages for creating web pages (JavaScript in which worked over 1/5 of the respondents, then PHP, HTML etc.). Unfortunately, it is worrying that 21 students (7%) did not work in any programming language at all during their high school education.

The next question was about teamwork experience in high schools. Results are shown in Chart 17. We can see that 2/3 of the students didn't work in teams, while 1/3 of them did. These results coincide with the results of a horizontal analysis conducted among high schools in the partner countries of the project.

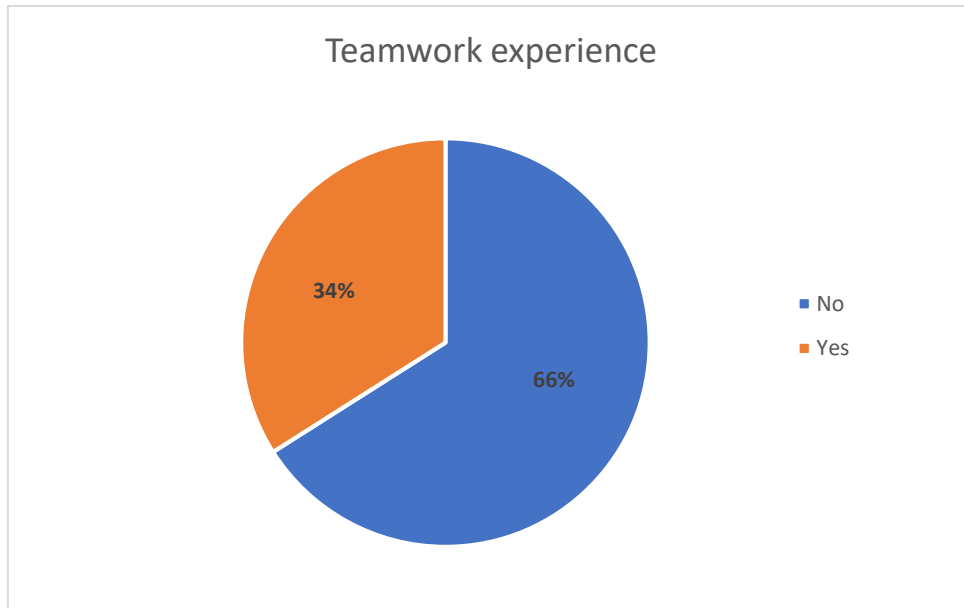


Chart 17 – Teamwork experience among students in high schools

The last question of this part of the questionnaire related to the students' self-assessment of programming knowledge, on a scale from 1 to 10, where 1 represents only a basic level of knowledge, and 10 represents the ability to independently solve very complex problems. Results are shown in Chart 18.

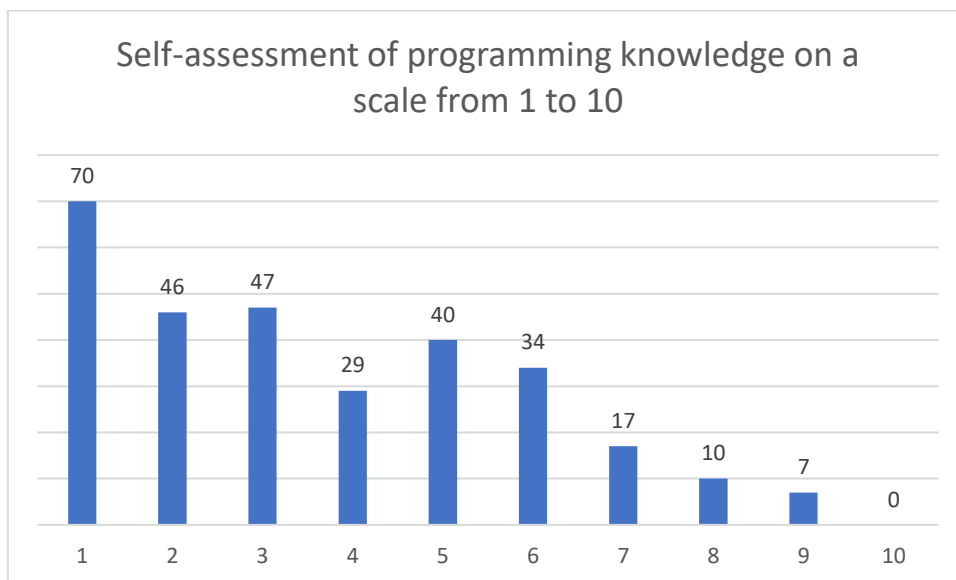


Chart 18 – High school students' self assessment of programming knowledge

We can see that the number of the students is inversely proportional to the degree of problem solving ability. Over 3/4 of the students think that their programming knowledge is below average (scales 1-5) and less than 1/4 of the students think otherwise (scales 6-10). It is interesting that none of the

students gave their knowledge the highest grade. We can conclude that students are aware that they do not possess sufficient competencies and abilities in the field of programming.

Second part of the questionnaire was related to the practical programming knowledge and skills. The students were given several questions from the field of programming (basic terms) and several specific program segments that they had to analyze.

First question was about the definition of the algorithm. The students were given a definition, and they should have recognized that it was a definition of an algorithm. There was also 'I don't know' answer available, just to ensure that students don't guess the correct answer. Results are shown in Chart 19.

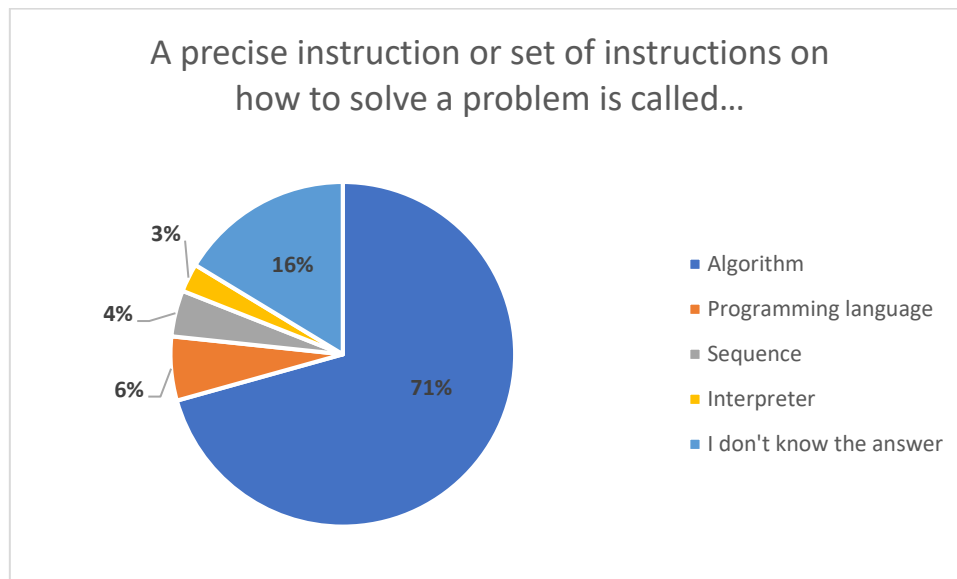


Chart 19 – How the students recognized the definition of an algorithm

We can see that 71% of the students did recognize the definition of an algorithm, which is reasonable number of correct answers. 13% of the students didn't recognize it, they thought it was about some other programming concept, and 16% of the students didn't know the correct answer. On the other hand, if we sum it up, almost 30% of students do not recognize the definition of an algorithm, which is a very high percentage, considering that it is one of the most basic concepts in programming.

The next question was about basic algorithmic structures in programming. Students were offered the names of three basic algorithmic structures (sequence, selection, iteration) along with some additional terms that are not (variable, function). Results can be seen in Chart 20.

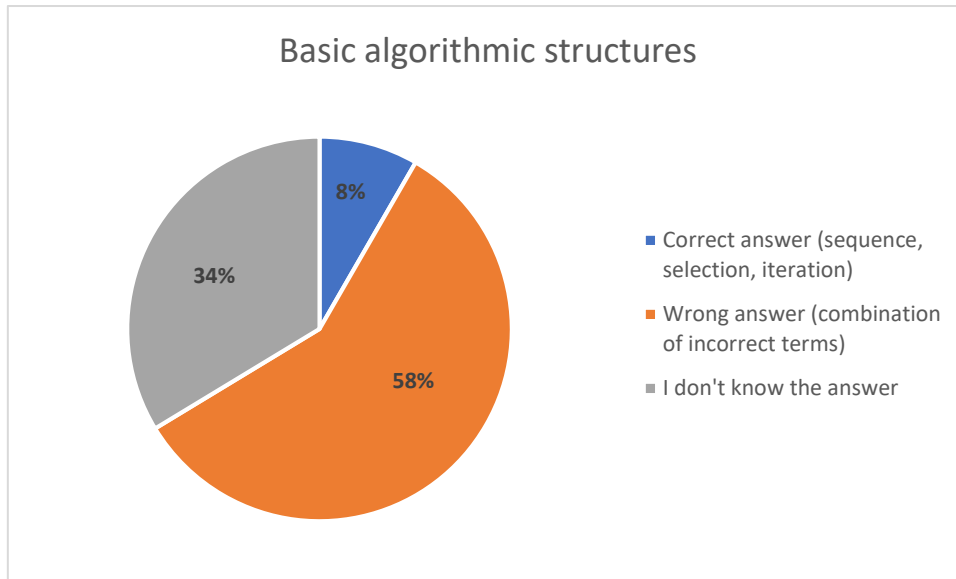


Chart 20 – Recognition of basic algorithmic structures

We can see that only 8% of the students correctly recognized all three algorithmic structures, while 58% of the students chose some other combination of terms. Some of those terms were correct, but in general, their answers were not entirely correct. 34% of the students did not offer an answer to that question.

After those two questions, students were given three program sections where they had to analyze the code sections and offer an answer (what the program section will print as a result). First question was a section containing only sequence, second question was a section of code containing two selections and third one was with iteration. The example of the code (question with iteration) is shown in Picture 4 and the results are displayed in Chart 21.

Python	C
<pre>s = 15 x = 8 for i in range(3): x = x + 6 s = s + x x = x + 6</pre>	<pre>int s, x, i; s = 15; x = 8; for (i=0; i<3; i++){ x = x + 6; s = s + x; x = x + 6; }</pre>

Picture 4 – Example of code section with iteration

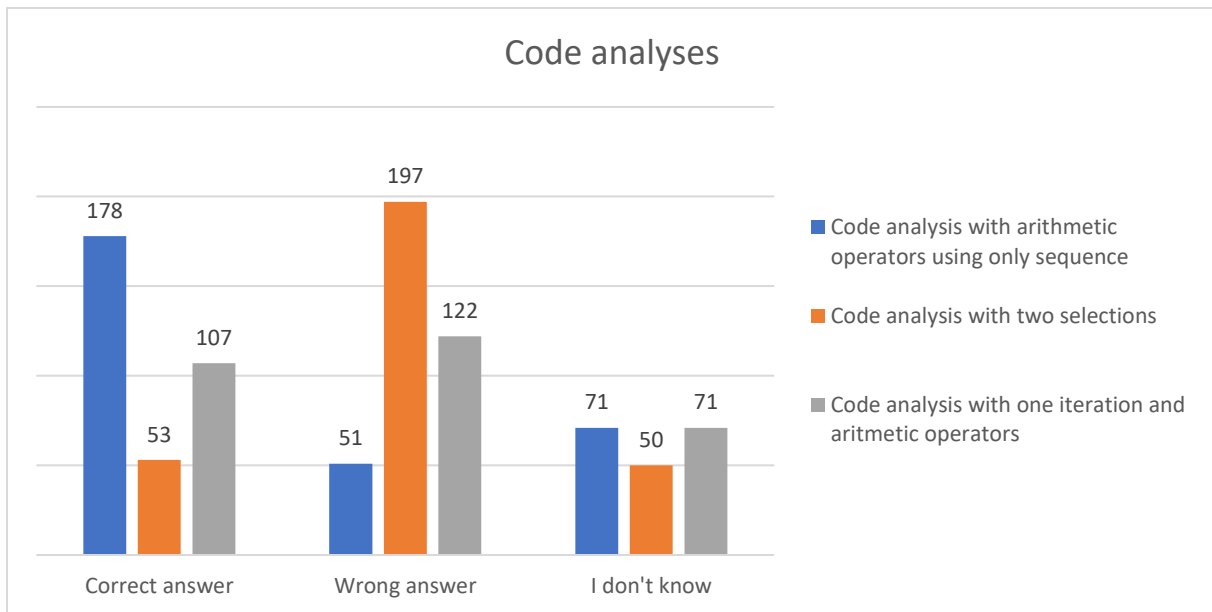


Chart 21 – Results of three tasks with code analyses

We can see that first task (section with code containing simple sequence structure and arithmetic operators) was successfully solved by 178 students (59%), while 41% of them either answered the question wrong or didn't know the answer. Second task with two selections was answered correctly by only 53 students (18%), while majority (82%) didn't offer the answer or offered wrong answer. Third question containing code with one iteration and arithmetic operators was solved by 107 students (36%), while most of them, again, offered no answer or wrong answer (64%). We can conclude that many of the students lack basic skills in analyzing program codes and often do not recognize how a certain program structure works. Also, they miscalculate results using arithmetic operators. It is also interesting to note that only 37 students (12%) offered the correct answer to all 3 tasks.

After those three examples of analyzing code sections, students got two more questions: about recursion and sorting algorithms. 53% of the students wasn't familiar with recursion and couldn't define it, while only 17% of the students could define recursion properly and recognize it's properties. Regarding question with sorting algorithms, 41% of the students couldn't choose correct sorting algorithm's from the list. Only 6% of the students correctly chose the sorting algorithms from the offered list, while 53% chose not to answer that question.

Third part of the questionnaire was dedicated to object oriented programming. Given that the OOP4FUN project is directly related to OOP, we thought it would be a good idea to see how familiar the students are with the mentioned concepts.

The first question in this part was again about student's self-assessment, but this time we asked them if they believe they recognize and can handle basic OOP concepts and have certain knowledge and experience in the field of object oriented programming. Only the students who answered with 'Yes' to that question were allowed to proceed with last set of questions. Results are shown in Chart 22.

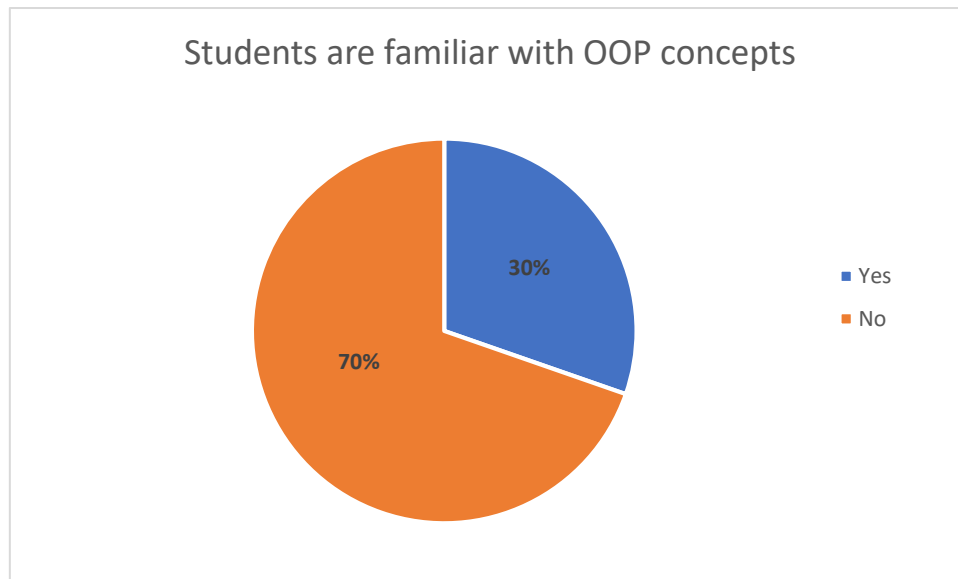


Chart 22 – Distribution of students who are/are not familiar with basic OOP concepts

We can observe that 30% of the students think they are familiar with basic OOP concepts, they can recognize and handle them and have certain knowledge and experience in the field of OOP. 70% of the students have not encountered the mentioned concepts, that is, they consider that they do not have basic knowledge about the mentioned topic.

It is also interesting to see this distribution among different type of high schools. We analyzed how many of those 30% of students attended certain type of school and whether this distribution coincides with the distribution of the total number of respondents by school types. The comparison can be observed in Chart 23.

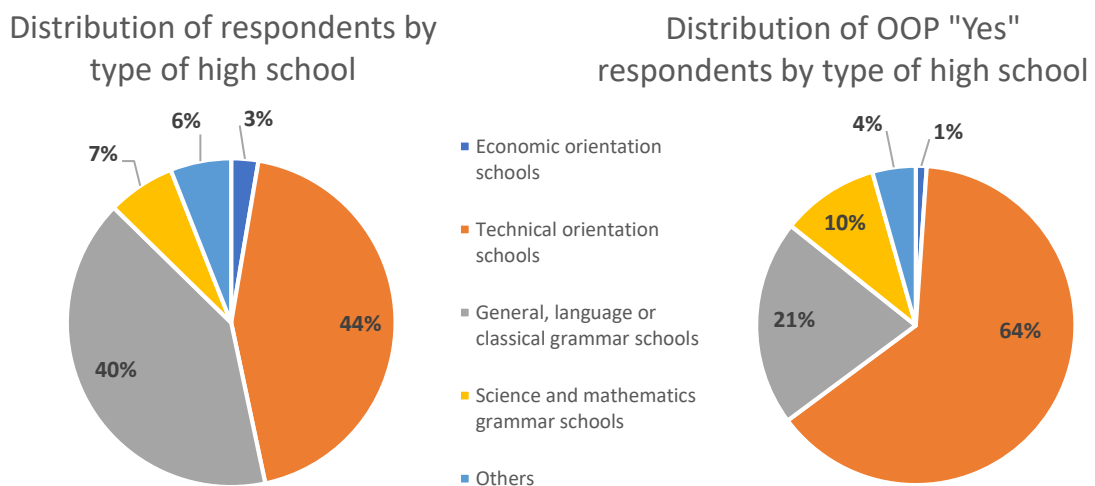


Chart 23 – Comparison of distributions of respondents by schools (all respondents) and respondents by schools familiar with OOP concepts

We can see that, regarding economy oriented schools and science and mathematic grammar schools, there is no big difference in the distribution between the total number of respondents and respondents who think that are familiar with OOP concepts. But, the difference is noticeable in technical schools on the one hand, and gymnasiums on the other. The difference is visible in favor of technical schools, they have 44% of all respondents but 64% of respondents with OOP knowledge, while gymnasiums have 40% of all respondents and only 21% of respondents with OOP knowledge. One of the possible reasons for this difference is that OOP in gymnasiums, according to the subject curricula, is only done within the elective subject of Informatics and only in one year, while in some technical schools, especially the ones that are IT-oriented, the numbers of subjects (and hours dedicated to OOP) is significantly bigger.

As mentioned earlier, only 30% of the respondents were going through the rest of questionnaire and their answers were analyzed. They were asked about basic OOP concepts and the purpose of those questions was to check if those students have understandings of basic theoretical concepts and terms related to OOP.

First, students were asked about classes and objects and understanding of their definitions. The results are shown in Charts 24 and 25. The students were given the definitions of class and object and they should have recognized which term has been described.

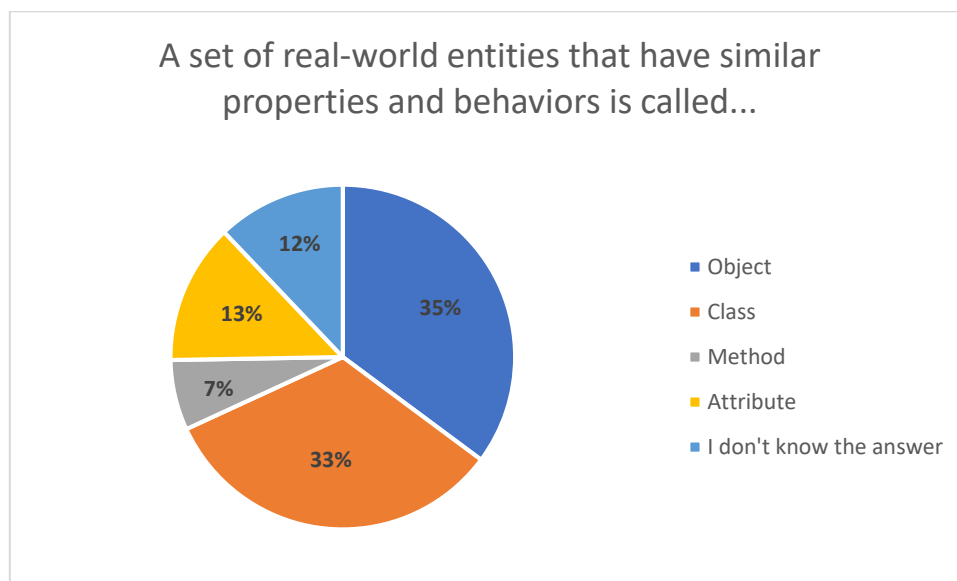


Chart 24 - How the students recognized the definition of a class

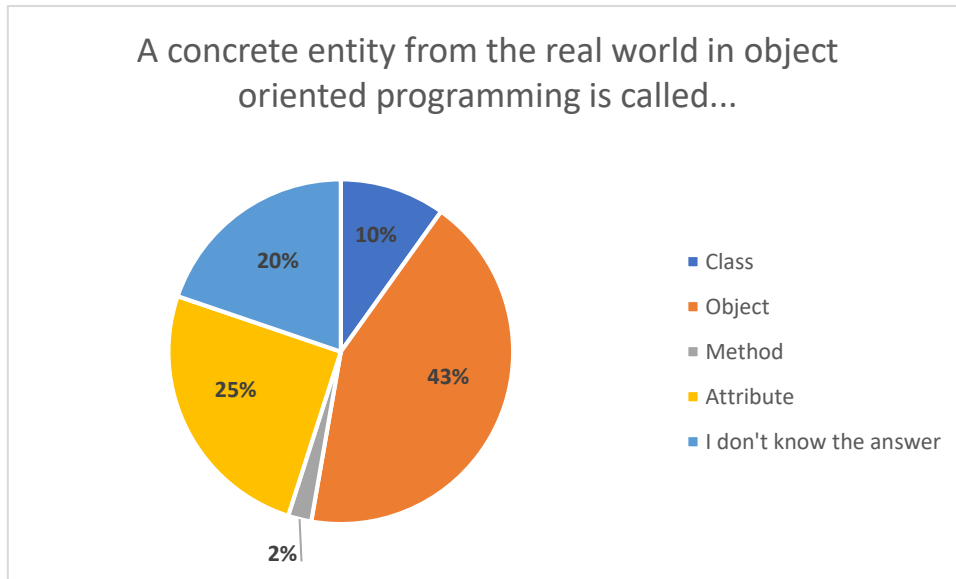


Chart 25 - How the students recognized the definition of an object

We can see that only 33% of the students did recognize the definition of a class, which is very low percentage considering that only a subset of respondents who claimed that are familiar with OOP concepts answered this and other following questions. 55% of the students didn't recognize it, they thought it was about some other programming concept, and 12% of the students didn't know the correct answer. Situation with object question is a bit better, but not satisfying. 43% of the students did know the definition of an object, while 37% offered wrong answer. The answer to that question was not offered by 20% of the students.

After that, students should have demonstrated an understanding of basic OOP concepts. They were given definitions of concepts and had to associate them with the correct name. The results are displayed in Chart 26.

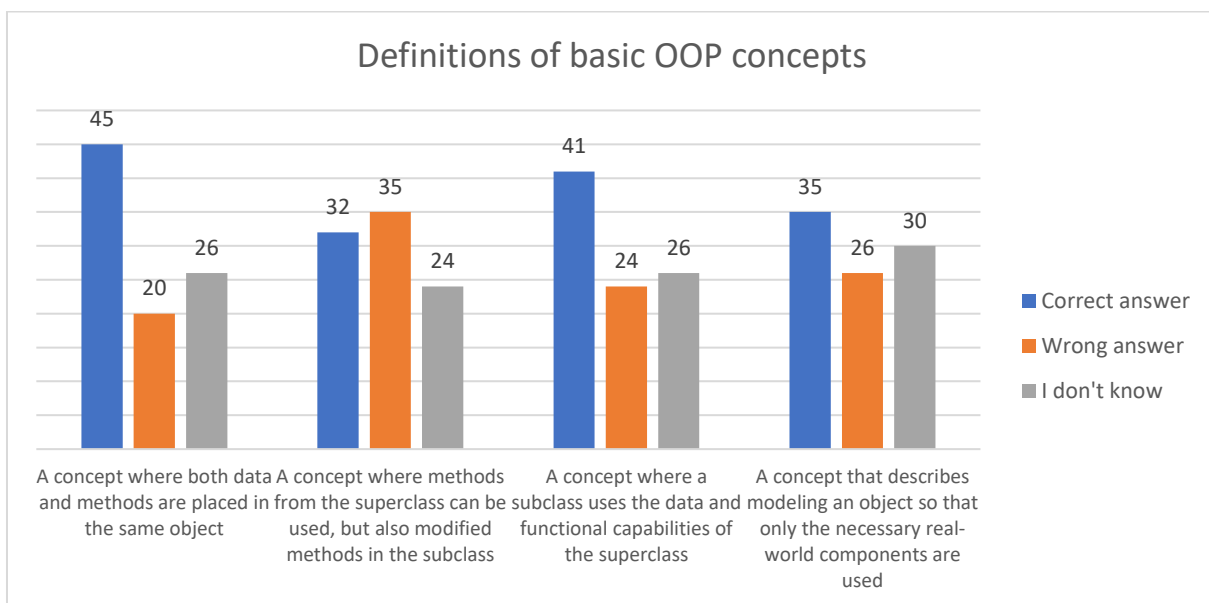


Chart 26 – Understanding definitions of basic OOP concepts

We can see that most of students are not sure about defining basic OOP concepts. 45% of them correctly recognized encapsulation, 32% of them are familiar with polymorphism, 41% correctly associated inheritance and 35% were right about abstraction. Most of the students were not sure about the correct answers or didn't offer the answer. It is also worth to mention that 25 students (out of 91) correctly associated all four concepts.

Then, the students were asked one more questions about classes, specifically about abstract classes and results are shown in Chart 27.

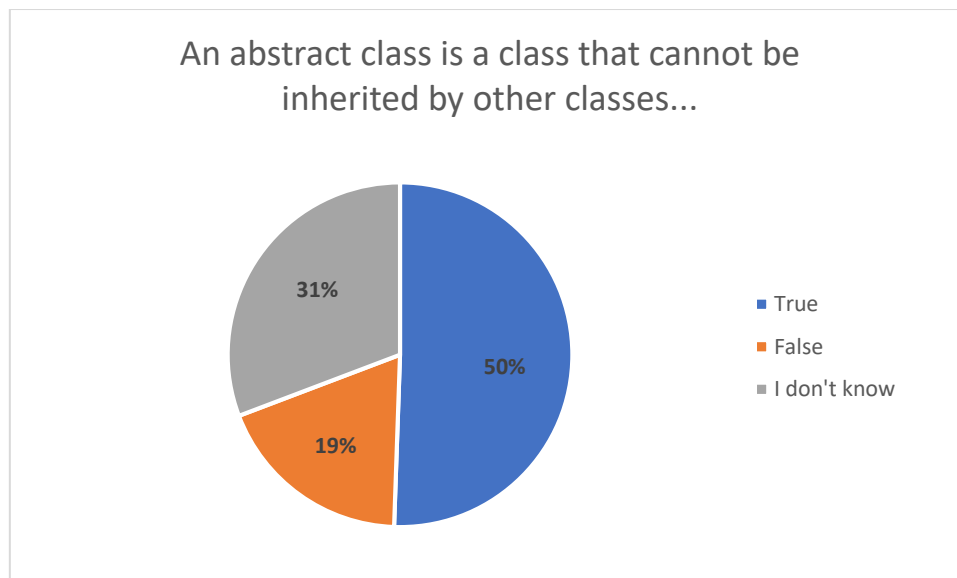


Chart 27 – Characteristic of an abstract class

We can see that most of the students don't understand the concept of abstract classes, only 19% of them offered correct answer.

At the end, students were asked about object-class relationship. They were offered with several statements and they had to choose if a particular statement is correct or wrong. The results are displayed in Chart 28.

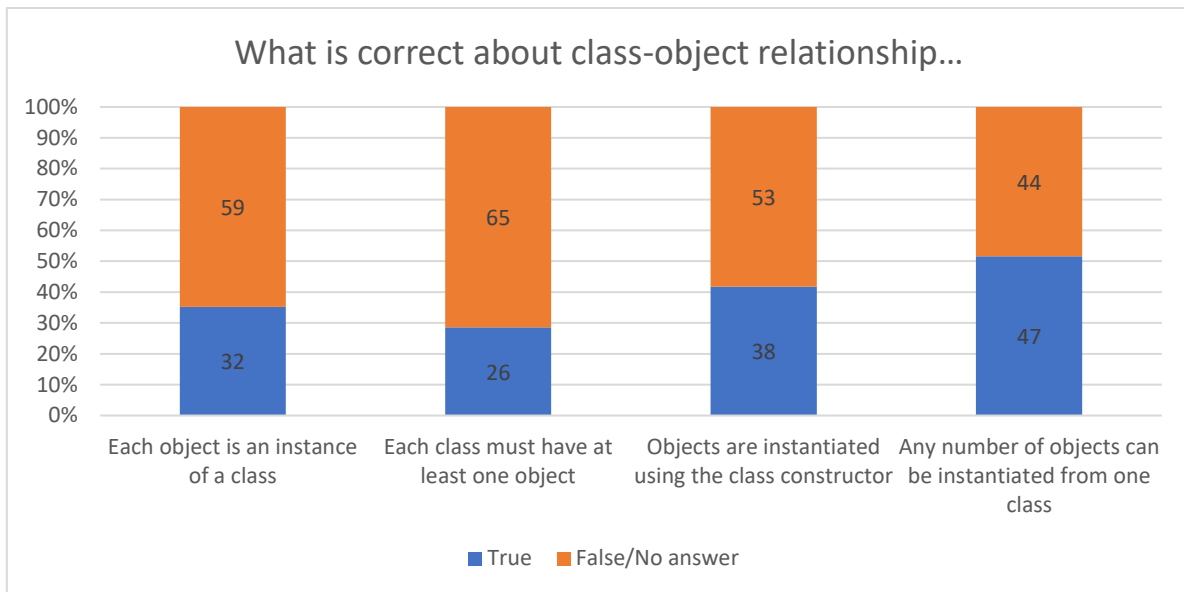


Chart 28 – Class/object relationship statements

Students were again unsure about relationship between classes and objects, answers are mixed, on three statements students offered incorrect answer, while on only one statement more students offered correct answer. Further analysis shown that just 18 students (20%) correctly answered that question entirely.

5.3.2. Teachers' expectations – semi-structured interview

5.3.2.1. Introduction

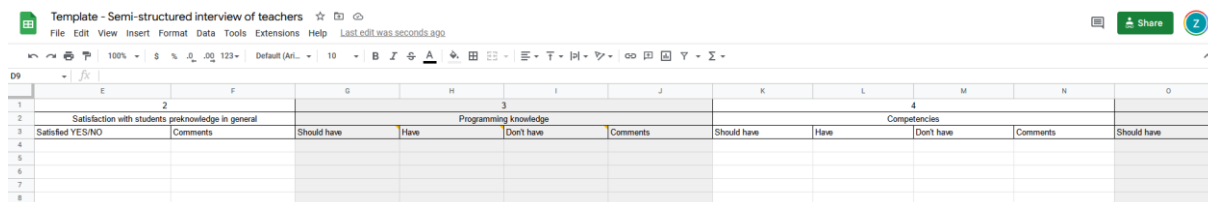
Given the fact that there is no legislative framework which would define the formal connection between the learning outcomes achieved at the high-school level and the pre-knowledge required to enroll a specific course at the university, and as previously defined, we took the approach to conduct limited-size research and compare the real knowledge of the freshmen students (see previous chapter) with the expectations from the teachers. We have identified two entry level courses at Faculty of Organization and Informatics, University of Zagreb (already mentioned in previous chapters) and wanted to include teachers teaching at exactly the same courses into this analysis. Thus, we have prepared a semi-structured interview to be carried out with these teachers. In this chapter we will bring you the information on the results obtained.

5.3.2.2. Interview design

Before conducting the interview, we prepared a set of questions that would be focused during the interview with teachers. The questions were divided into the following groups:

- Questions related to teacher profile and experience
- Satisfaction with entry programming knowledge in general
- Expectations related to programming knowledge
- Expectations related to programming competences
- Expectations related to programming skills

In order to note down the answers the spreadsheet document was prepared. See Picture 5 below.



Satisfaction with students preknowledge in general		Programming knowledge				Competencies				
Satisfied YES/NO	Comments	Should have	Have	Don't have	Comments	Should have	Have	Don't have	Comments	Should have

Picture 5 - Responses – Semi-structured interview with teachers

This set of questions didn't include any personal or sensitive information on teachers or students and thus there was no need to request the permission from the Committee for Ethical Matters (HR: Etičko povjerenstvo) at the Faculty of Organization and informatics prior to conducting this interview. However, the detailed instructions for the interviewers are prepared in advance. Those instructions are attached in the text below.

Also, special care was taken to avoid discussing any particular student or teacher directly or indirectly during the interview. All questions were placed by focusing on programming course in general or on students' population in general.

This interview design along with template to note the answers were shared with partners from Slovakia and Serbia to enable them to use the same/similar methodology in their vertical analysis.

The interviewer was the author of this chapter. To make the interview straight forward, to make sure the important information is presented to the interviewed teachers and to make the same flow of the interview in each instance of it, the detailed guidelines were prepared in advance and are copy-pasted here as follows:

Instructions for the interviewer

Note that the texts in blue are instructions for the interviewer (questions are written in black).

The answers must be written in the provided template (excel file).

All questions are optional.

The interview is to be held informally, in a conversation-like style and in Croatian language.

Introduction

a. Explain the purpose of the interview:

The University of Zagreb, Faculty of organization and informatics along with four other universities from Slovakia, Germany, the Czech Republic, and Serbia, and five high schools from

the same countries is running an Erasmus+ project which aims to identify and eliminate the gaps between high school learning outcomes and university required input skills and knowledge related to object-oriented programming. One of the outputs of the project will be an innovative high school course in games development which will introduce high school students to the basic concepts of object-oriented programming (in order to better prepare students for universities as well as to increase their motivation for enrolling in STEM study programs in general). Thus, we are currently analysing the gap in the high-school outputs and the university inputs requirements.

At the beginning of the semester we gave to the students the short test and we obtained their entry knowledge, and now we want to align that knowledge with the expectations from the teachers.

As a university teacher, with years of experience and by teaching the freshmen students you have the first contact with them, you have the insight into their knowledge which they brought with them to the university, and we hope you could give us insight on the level of alignment of that knowledge with the ideal or expected knowledge the freshmen should have. Thus, with this interview, we would like to ask you about your experiences and expectations which are related to mentioned concepts.

b. Explain that the answers will be treated anonymously.

All obtained answers, opinions, suggestions and other inputs that you will give us during this semi-structured interview will be generalized and treated completely anonymously. Also, all the questions are optional for answering and you can decide if you want to answer any given question or not.

General questions on teachers profile

- 1. Some personal information is needed to contextualize the answers regarding other responders. Name (will be deleted after the analysis), entry study program teacher is teaching, experience in teaching.*

Satisfaction with entry programming knowledge in general

- 2. Are you satisfied with students pre-knowledge in programming in general? The possible answers (YES / NO) are in the template, please write the answers in the provided template – Excel file)*
- 3. Are there any comments you would like to point out related to students programming pre-knowledge in general? Comments, if any should be written in Excel file. The comments are free style comments, thus teachers could comment anything related to their satisfaction or other aspects relevant to pre-knowledge, gaining it etc.*

Expectations related to programming knowledge

Introduction: The knowledge is the possibility to reproduce the facts about the subject of interest. Please ask questions before giving examples, and use examples only if necessary.

4. Which programming knowledge first year (freshmen) students should have when enrolling your course?
5. Which programming knowledge have you noticed that first year (freshmen) students do have when enrolling your course?
6. Which programming knowledge have you noticed that first year (freshmen) students do NOT have when enrolling your course?
7. Are there any comments you would like to point out related to students' programming knowledge expectations we have just discussed about?

Expectations related to programming skills

Introduction: The skill is the ability to perform certain physical tasks or activities in the desired way. Please ask questions before giving examples, and use examples only if necessary.

8. Which programming skills first year (freshmen) students should have when enrolling your course?
9. Which programming skills have you noticed that first year (freshmen) students do have when enrolling your course?
10. Which programming skills have you noticed that first year (freshmen) students do NOT have when enrolling your course?
11. Are there any comments you would like to point out related to students' programming skills expectations we have just discussed about?

Expectations related to programming competences

Introduction: The competencies are the broader term that includes the skills, knowledge and attributes that enable a person to perform effectively in a job or situation. Please ask questions before giving examples, and use examples only if necessary.

12. Which programming competences first year (freshmen) students should have when enrolling your course?
13. Which programming competences have you noticed that first year (freshmen) students do have when enrolling your course?
14. Which programming competences have you noticed that first year (freshmen) students do NOT have when enrolling your course?
15. Are there any comments you would like to point out related to students' programming competences expectations we have just discussed about?

Closing the interview

16. Finally, are there any other aspect or factors you think are important to be considered when analysing the pre-knowledge and the gap between university expectations and what students actually bring? Is there anything you wish to add?

Closing of interview: Thank you very much for your cooperation. It has been exciting to talk with you, and I am sure your help will be of great use in the study we are conducting. I wonder if I can contact you on a future occasion to request further clarification of information or any additional contribution to the project.

5.3.2.3. Conducting the interview

In order to conduct this semi-structured interview, the e-mail message with the request to participate in this semi-structured interview has been sent to all teachers who are teaching programming related courses in first semester of our university study program and professional study program. However, only three (3) teachers have positively responded to the inquiry and the interviews were organized during December 2022.

The interviews were completed in the time frame between 63 minutes (of the shortest) and 86 minutes (of the longest interview). However, due to the fact that the interviewer and the teachers

However, the semi-structured interview with all teachers resulted in unexpected flow of discussion which was determined by the fact that **teachers do not have knowledge expectations from freshmen students**, and the courses are organized in a way that they teach students from scratch. Thus, instead of answering a straight forward questions they gave to the interviewer lots of other, personal, remarks and observations. Consequently, during the interviews some of stated questions were only partially answered while some remained completely unanswered as teachers didn't have any expectations at all.

(interviewees) have long term collaboration in different activities and are working at the same institution, the interviews took place as the informal meetings. Thus, the interviewer tried to utilize the time and to maximally focus on the interview instead of personal chat or discussions.

5.3.2.4. Deciding on analysis method

Although not initially planned, all the mentioned remarks and observations obtained from the teachers are noted down by the interviewer, as they now became the only data we have. Saying it in another words, instead of obtaining structured or semi-structured data on our primary topic, we have received a lots of unstructured data related to different (sometimes only partially connected) topics.

As a result, we had to decide which method or approach of data analysis to use to analyze such unstructured data obtained by the interview. With some previous experience in such analyzes and after the eliminating all the options that are suitable for more structured or quantitative data we narrowed our list of possibilities down to a three possible methods:

- Qualitative content analysis / Coding analysis: Coding involves categorizing and labeling the data. This method would allow us to identify patterns and themes across the data. It can be done manually or with the help of software such as NVivo.
- Thematic analysis: a qualitative research method that uses data from interviews, focus groups, or other sources to identify patterns or themes in the data. We could use thematic analysis to identify the underlying motivations and attitudes behind the responses given in semi-structured interviews.
- Case study analysis: involves collecting and analyzing qualitative data to gain a deeper understanding of the research problem. This method can be used to analyze the data from semi-structured interviews as it provides an opportunity to explore in depth the responses of the interviewees and gain insights into their experiences and beliefs.

Finally, we have decided to use case study analysis as it is advantageous over coding and thematic analysis methods in analyzing text-data obtained from semi-structured interviews because it allows for a deeper exploration of the data. According to the literature, the case study approach allows the researcher to uncover deeper insights into the phenomenon of interest and to provide a more comprehensive understanding of the data. In sum, the case study approach would provide us with a more comprehensive view of the data, which is not possible through coding or thematic analysis methods.

As the three interviewed teachers had quite different positions and experience, we decided to perform three case studies and to seek for the conclusions important for our research of gap analysis from experienced teacher, young teacher and young teaching assistant who has experience as student demonstrator.

5.3.2.5. Case study 1 – experienced teacher

Persona description – This teacher has more than 25 years of experience in teaching mainly on entry level programming courses such as Introduction to programming, Programming 1, Programming 2, Object oriented programming etc.

Satisfaction with students' prior knowledge in general – Due to the fact that there are no formal expectations for students to meet in order to enroll the courses, teacher avoids to explicitly express the satisfaction on students' prior knowledge on programming in general. His experiences are that students have a very diverse range of prior knowledge, from those who know nothing to those who have a lot of theoretical and applied knowledge and skills. His assessment is that about 10% of students have acquired knowledge independently while about 40% of students have some prior knowledge which can be used in subjects related to programming.

Concerning prior knowledge, the teacher states that the problem was when students had experience with Visual Basic, as they were not prepared to delve deeper into the essence of the problem and the algorithms. The bigger issue was the lack of programming practices and habits rather than the prior knowledge itself. Based on teachers' own scientific research, even 13% said they had a fear of programming. Those who had experience with Python said they had less fear of programming but more often said C++ was challenging, as we are talking about a lot of material to be learnt by students in a short timeframe.

Programming knowledge students should/don't have – Teacher thinks that only after mastering the basic programming skills and habits, such as programming thinking, students should learn loops, selections, process and basic data structures in order to be able to independently create complex structures. The programming language itself is not the primary concern. Rather, they should have general IT knowledge, and after the mentioned concepts have been learned, they should familiarize themselves with a specific programming language and afterwards with its syntax. It is beneficial for students to have a general knowledge in IT, especially in a programming environment. Ideally, they should have a knowledge of C and C++, and be able to create algorithms. Being able to code and debug programs in any language, regardless of the syntax, would be an advantage. Although understanding the concepts of object-oriented programming (OOP) is beneficial, teacher thinks that when discussing programming in a school setting, it can be assumed that only a small percentage of students will have encountered Object-Oriented Programming (OOP). Students may have encountered objects in languages like JavaScript or Python, but the concepts of OOP go beyond the concepts of structural programming. Utilizing a tool like Alice, which provides a 3D world to guide students, can help those who have not been exposed to structural programming learn OOP. Possessing basic knowledge of computer science is beneficial in any programming environment. Experience with coding, regardless of the programming language, is beneficial as it allows students to gain insight into the different types of programming cycles. The exact knowledge is not as important as the concepts behind it.

Competences students should/don't have – Experienced teacher thinks that it would be highly beneficial if students' programming skills were developed in the high school to the point where they could implement basic algorithms. However, according to his experiences there are fewer than 10% of students who can traverse an array, find the largest element in an array, or similar with the high school knowledge. Now, the basic computing skills of students entering college are much better than they used to be. The situation is significantly better than it was and we are generally satisfied with it.

Skills students should/don't have – Teacher again emphasizes that programming skills and habits are very important. Thinking in a programming way, in terms of solving problems in a programming way is very important. Programming habits that include activities that make up good programming practices include syntax checking, testing requirements, and that the student must go through all their code blocks, etc.

Several conclusions that could be driven from the first case study, which are related to student's prior knowledge in programming, are:

1. Students enrolling university have a diverse range of prior knowledge, from those who know nothing to those with a lot of theoretical and applied knowledge and skills.
2. As much as 13% of students enrolling university have a fear of programming. Those who programmed in Python have less fear but more challenges in learning C or C++.
3. Only up to 10% of enrolling students can find largest element in array with the high school knowledge.
4. It is beneficial for students to have general IT knowledge and knowledge of C and C++, and be able to create algorithms.
5. Students in high school should learn basic programming skills and habits, such as loops, selections, process and basic data structures.
6. Utilizing a tool like Alice can help those who have not been exposed to structural programming learn OOP.

7. It is important for students to learn in high school to think in a programming way when solving problems.
8. Good programming practices which include syntax checking, testing requirements, and going through all code blocks should be acquired in high school.
9. Basic computing skills of students entering college are much better than they used to be.

5.3.2.6. Case study 2 – young teacher

Persona description – This teacher has about 15+ years of programming experience and about 10 years of teaching experience. He has been teaching entry level courses on both university and professional study programs.

Satisfaction with students' prior knowledge in general – Young teacher also stated that he does not have any particular expectations in terms of prior knowledge of students when enrolling the university. However, he is aware that starting from the zero-knowledge point and educating students from there requires a significant amount of time and the teacher states that the current curriculum of (introductory) programming courses does not plan sufficient teaching hours, which turn out to be necessary in such an ecosystem.

The teacher states that if we were to require any prior-knowledge in programming of freshman students, we would have to amend our university curriculum, which is intentionally built to cover the topics in programming from the beginning. Therefore, the teacher believes that general IT knowledge is more important for students.

If we presume that we would require notable prior-knowledge in programming of freshman students, the young teacher thinks that the legislative framework should also be changed to accommodate such requirements, but also to provide a solid basis for high school reform. This teacher also points out that, due to the fact that IT experts are not eager to teach, especially in high schools, where salaries are rather low, the principals are forced to employ teachers who have only partial knowledge in informatics, and thus they are unable to cover higher demands in teaching students programming. The legislative framework should also address this issue.

Programming knowledge students should/don't have – The teacher said that it is hard to answer this question and that it is relative on the point of view regarding what knowledge should students bring with them. However, previously stated facts are underpinned in this part of the interview as well. The young teacher pointed out again that for our case, the basic IT literacy should be present, such as knowing and understanding how to work in a given operation system when enrolling the faculty. Second underpinned statement from this teacher is that the knowledge in mathematics and logic are the closest what one get learn in high school and that would help in software problem solving and thinking. Thus, basic IT literacy as well as knowledge in logic and mathematics, will help student to go through university curricula.

Ideally, if possible, when enrolling the university, student should be familiar with variables, data types, selections, iterations and functions. Quite contrary to the results of case study 1, the young teacher thinks that students will eventually learn on their own an algorithmic problem solving if possessing above mentioned knowledge. By taking into consideration the issues in high-school education system, that were mentioned in the previous chapter, this teacher puts lots of focus on students' self-learning

capabilities, and thus the teacher gave us the advice to try to have out of this project two results, namely: a portal with self-learning materials for freshmen students and; try to organize the education for teachers as well.

To conclude, having the above mentioned, anybody who enrolls the university with some theoretical or applicative knowledge will have an advantage over other students in obtaining course learning outcomes related to programming and object-oriented-programming.

Competences students should/don't have – Young teacher was quite pessimistic by stating that majority of the students do not possess any competences or skills worth mentioning. This makes them feel fear. The teacher mentioned that having a theoretical knowledge does not guarantee any competences. Only the knowledge underpinned by practical skills means competence. The competences the students should really bring are related to the knowledge and skills in algorithmic problem solving, but as previously stated this is something lacked by vast majority of freshmen.

Skills students should/don't have – Our interview did not yield anything new that would be worth mentioning in the section on skills. We just went again through those requirements mentioned before and discussed about what would be related to knowledge, competences and lectures.

Several conclusions that could be driven from the second case study, which are related to students' prior knowledge in programming, are:

1. Current curriculum of (introductory) programming is built with a prejudice that students do not have any prior-knowledge.
2. Sadly, current curriculum of (introductory) programming courses does not plan sufficient teaching hours as we have to teach students programming from scratch.
3. To change this, new curriculum should be developed, but more importantly a new legislative framework should also be developed to cover some other issues, such as to improve the motivation for teachers with programming knowledge to come and teach in high schools.
4. In current situation, basic IT literacy as well as knowledge in logic and mathematics, will help student to go through university curricula. Actually any theoretical or practical knowledge will help a lot.
5. If possible, when enrolling the university, student should be familiar with variables, data types, selections, iterations and functions.
6. Young teacher thinks that although freshmen students in general don't possess any notable competences or skills related to programming, with adequate materials, they are capable of self-learning algorithmic problem solving.
7. Education of high school teachers is very important.

5.3.2.7. Case study 3 – teaching assistant / student demonstrator

Persona description – This teaching assistant was just employed at the university but has several years of experience of being a student demonstrator and was in contact with freshmen students on different courses, including programming related ones. Contrary to previous teachers, this teaching assistant has experience of working with university study program students but not with professional study program students.

Satisfaction with students' prior knowledge in general – Student demonstrator explicitly states his unsatisfaction with students' prior knowledge in programming. Although, unsatisfied, he also argues that we cannot expect from students a lot and that both answers could be taken in consideration depending on the point of view. He argues that majority of the students after enrolling the first year of college for a few weeks or months even don't know "where they are and what is going on". On the other hand, we can be unsatisfied as they seem not to be very interested in programming. Thus, he concludes that maybe high schools could motivate students to find enjoyment in coding.

Programming knowledge students should/don't have – This teacher notices that those students who finished technical high schools are generally the only ones who actually have some kind of programming knowledge whatsoever. Their knowledge consists of variables, if/for blocks and simple arrays. Rarely anyone knows how to create a linked list. However, in general, students have no idea about OOP, classes etc. Problem is that basically they see no purpose in packing variables into a common structures or objects.

To conclude on this question, this teaching assistant thinks that understanding of variables and variable types, basic syntax of flow and structure, data types and functions would be desirable knowledge from high-school students.

Competences students should/don't have – In order to make their way through introductory classes of object programming with flying colors, the opinion of student demonstrator is that, enrolling students should have the basic idea how to make up an algorithm which includes the use of variables, selection, iteration and functions. It would be also beneficiary if they would know how to automatically pack any data in structures in their minds.

However, the experience of this teaching assistant / student demonstrator is quite opposite. When students arrive at our university, most of them don't have a clue how to write (good) code. Most of them struggle with the very basics of programming. Some even fail to understand the purpose of variables. Usually, students struggle with concepts behind nested for-loops. Those with better knowledge are usually overconfident and stop actively attending classes. Some of them pay the price by failing and attending the course again next year.

Skills students should/don't have – The discussion on skills students should/don't have led us back to repeating or talking about knowledge and competences. Skills related to the programming development environments, programming languages, technology, work in teams, work with end users etc., were not discussed. Due to this fact, it is quite doubtful the opinion that skills should strictly be taught on college by various assignments, not in high schools, and we will neglect it as an outlier in our data.

Several conclusions that could be driven from the third case study, which are related to student's prior knowledge in programming, are:

1. Student demonstrator explicitly states his unsatisfaction with students' prior knowledge in programming.
2. For the first few weeks or months students seem not to know "where they are and what is going on", and what is even worse they seem not to be very interested in programming and they see no purpose in programming knowledge.
3. Mainly students who finished technical high schools actually have some kind of programming knowledge whatsoever.

4. Enrolling students should have the basic idea how to make up an algorithm which includes the use of variables, selection, iteration and functions.
5. Student-to-student experience shows that most of them struggle with the very basics of programming.
6. Students with better prior knowledge are usually overconfident and stop actively attending classes.

5.3.2.8. Comparing cases

In order to better understand the obtained data, we have prepared a table view which tries to summarize and put into the relationship the opinions from three teachers who have different experience in working with freshmen students. The summary is presented in the table below.

Table 8 – Review of teachers' experience

Concern	Experienced teacher	Young teacher	Teaching assistant
Curriculum	Basic computing skills of students entering college are much better than they used to be.	Current curriculum of (introductory) programming is built with a prejudice that students do not have any prior-knowledge. At the same time, current curriculum of (introductory) programming courses does not plan sufficient teaching hours as we have to teach students programming from scratch.	
Curriculum results	As much as 13% of students enrolling university have a fear of programming.		For the first few weeks or months students seem not to know "where they are and what is going on", and what is even worse they seem not to be very interested in programming and they see no purpose in programming knowledge.
New curriculum / legislative framework		New curriculum should be developed, but more importantly a new legislative framework should also be developed to cover some other issues, such as to improve the motivation for teachers with programming knowledge to come and teach in high schools.	

<p>Prior knowledge in related subjects</p>	<p>It is beneficial for students to have general IT knowledge and knowledge of C and C++, and be able to create algorithms, e.g. to think in a programming way when solving problems.</p> <p>Those who programmed in Python have less fear but more challenges in learning C or C++.</p>	<p>Basic IT literacy as well as knowledge in logic and mathematics, will help student to go through university curricula. Actually any theoretical or practical knowledge will help a lot.</p>	<p>Mainly students who finished technical high schools actually have some kind of programming knowledge whatsoever.</p>
<p>Prior knowledge in programming</p>	<p>Students enrolling university have a diverse range of prior knowledge, from those who know nothing to those with a lot of theoretical and applied knowledge and skills.</p>	<p>When enrolling the university, student should be familiar with variables, data types, selections, iterations and functions.</p>	<p>Student demonstrator is unsatisfied with students' prior knowledge in programming.</p> <p>Enrolling students should have the basic idea how to make up an algorithm which includes the use of variables, selection, iteration and functions.</p>
<p>Prior competences and skills</p>	<p>Only up to 10% of enrolling students can find largest element in array with the high school knowledge.</p>	<p>Freshmen students in general don't possess any notable competences or skills related to programming</p>	<p>Student-to-student experience shows that most of the students struggle with the very basics of programming concepts.</p> <p>Students with better prior knowledge are usually overconfident and stop actively attending classes.</p>
<p>How to overcome shortcomings in knowledge</p>	<p>Students in high school should learn basic programming skills and habits, such as loops, selections, process and basic data structures.</p> <p>Good programming practices which include syntax checking, testing requirements, and going through all code blocks should be acquired in high school.</p>	<p>Students are capable of self-learning algorithmic problem solving if provided with good materials.</p>	
	<p>Utilizing a tool like Alice can help those who have not been exposed to structural programming learn OOP.</p>	<p>Education of high school teachers is very important.</p>	

From the table above it can be seen that teachers had a quite diverse views on the topic of the interview. They gave a lots of personal opinions, either from some of their previous researches or from their own experience and observations. Although some of the teachers introduced unplanned topic into the conversation, the table shows that their opinions are very much aligned and that they are

complementary. The opinions on topics that are introduced by one or two teachers are not in contrast to the opinion of other teachers in other topics.

We dare to say that such result which brought out more concerns and opinions that initially planned is welcomed as it finally covered the whole lifecycle of educational concepts, from design to their implementation and evaluation.

5.4. Conclusion on gap analysis

Considering all the aspects of this analysis, it is important to say that the problem with the gap analysis, even on a national level, is deeper, more diverse, and more complex than it seemed at the first glance.

Given that there are no defined prerequisites for enrollment to computer science universities at the legislative level, the level of OOP knowledge required as input to the universities cannot be established and determined, except internally when students are already enrolled into a specific university which might, or might not test such knowledge. Additionally, due to the fact that there are no pre-defined conditions, students come from high schools with different levels of programming knowledge, from those who do not even handle basic concepts to those who are able to solve complex problems. However, the second group of students is much smaller.

The results of the questionnaire given to the first-year students, and presented in previous chapters, gave us a better insight into the actual knowledge of students after finishing high school education from the aspect of programming. The results are not encouraging considering that the students showed a lack of understanding of basic programming concepts. However, they are also aware of their gaps in knowledge. It is also evident that the majority of students are enrolling in universities after only one or two years of studying informatics in high schools, which certainly cannot be enough to acquire a sufficient level of competence and knowledge to successfully (or at least without a certain level of difficulties) continue their education at STEM or IT-related university. What is discouraging is the fact that students during their high school education were not sufficiently interested or motivated to independently research topics and contents in the field of informatics, especially programming, which now results in very poor prior knowledge. According to some of the students' answers, in high school, they only learned the basic of computer architecture and Office programs, without any concepts related to programming. Those students are now in IT university, without any programming prior skills, especially if they didn't study those topics on their own. This is related to the fact that most of the students stated that they are not familiar with some basic programming concepts, such as iterations or selection. This was also seen from their ability to solve some simple tasks in the form of code analysis, where most of them got the wrong results.

As object-oriented programming is the fundamental basis of this entire project, it was important to establish the level of competence of the students in that area as well. The results are devastating considering that only 30% of students are familiar with the concepts of OOP, and according to their answers in the questionnaire, it is evident that the majority of that 30% still do not recognize the basic concepts. From the curricula of IT subjects in high schools, OOP is poorly represented, with an insufficient number of teaching hours. This analysis proved this, given that only a small number of respondents are familiar with the basic concepts of OOP. Regardless of prior knowledge of programming, students also lack experience in teamwork, which is very much used at universities through their work and involvement in various projects. From interviews conducted with teachers and

their research, it was pointed out that 13% of students are afraid of programming and also that students coming from technical high schools have better prior knowledge compared to other schools. Analyzing the correlation between the survey of students and the opinions and experiences of teachers, it can be concluded that everything mentioned above coincides completely.

This ultimately leads to the fact that university teachers have to start from the basics of programming, to teach students some basic concepts so that they can finally reach some level with which they can successfully follow university-level course curricula. However, one of the teacher's observations is that students' basic computer skills are better today than in previous generations.

Another very important aspect of the gap analysis is the prior knowledge and skills that students should have, that is, what teachers expect from students entering universities. Teachers clearly stated that basic IT literacy as well as knowledge of logic and mathematics will help students to go through university curricula. Actually, any theoretical or practical knowledge can help a lot. They also mentioned that students in high school should learn basic programming skills and habits, such as working with variables, data types, loops, selections, iterations, functions, and basic data structures. Good programming practices which include syntax checking, testing requirements, and going through all code blocks should also be acquired in high school. Besides general IT knowledge, it would also be beneficial for students to possess knowledge of C and C++, and be able to create algorithms, e.g. to think in a programming way when solving problems.

Teachers also proposed some recommendations to obtain a higher level of knowledge and skills in programming for high school students. They mentioned that a new curriculum should be developed, but more importantly, a new legislative framework to cover some other issues, such as to improve the motivation for teachers with programming knowledge to come and teach in high schools, as, in their opinion, motivation of high school teacher is very important. IT experts are not eager to teach in high schools, where salaries are low, so the principals are forced to employ teachers who have only partial knowledge of informatics and they are unable to cover higher demands in teaching programming. Utilizing a tool like Alice (or similar) can also help those who have not been exposed to structural programming to learn OOP.

We can conclude that there is a lot of work in front of all stakeholders, legislation enforcers to prepare stable and motivating infrastructure and environment, high-school curriculum designers to take into consideration the growing need for programming knowledge and STEM in general, institutions educating teachers of informatics to enable them to teach programming and related concepts, university curriculum designers and university teachers to build on high-school knowledge and to students to take every opportunity to acquire the skills and competencies required in the future dynamic market.

6. Conclusion

In this part, we first drew some of the most important conclusions we reached for each country, and then highlighted some of the priority activities that should be undertaken in order to raise the quality of teaching and the knowledge of our IT students.

The vertical analysis consists of a brief overview of the education system in the Czech Republic, focusing on secondary schools, vocational schools and universities. A detailed view is taken : (1) on the computer science education at the gymnasium Pardubice that follows a study program of the whole Czech Republic, and (2) on the computer science study program at the University of Pardubice. The project partners in Pardubice found out that an ideal move from Gymnasium to University (also with OOP topics) requires attendance on a seminar in the school that is an optional choice. Students without that seminar have bad start positions in the university. It is also mentioned that a Gymnasium delivers broad education with optional subjects. It depends on the individual case, whether beginners at universities have good starting conditions or not.

Analyzing details of teaching and learning at the Gymnasium Pardubice and University of Pardubice and Faculty of electrical engineering and informatics (**Czech Republic**) with its study program Information Technology we came to the following conclusions:

- In the primary school during 6 years out of 8 and in secondary school in 2 years out of 4 years students concepts related to informatics are covered in the subject Informatics. It is mandatory and cover very general topics and some topics related to programming. Last two years students can choose optional subjects. One of them is seminar of programming, which gives good starting position to study technical or informatically oriented university.
- Problem is that without optional subject seminar of programming there is no OOP topic during the study in Gymnasium Pardubice and students have bad start position in informatically oriented study programs in universities.
- The problem that exists in the Czech Republic is that a very small number of high school students (20% of graduates of gymnasium) study technical and informatically oriented universities.
- Therefore, a large number of unsuccessful students that do not complete university study.

This vertical analysis relates specifically to the computer science study program of HTW Dresden (**Germany**) and the partner secondary school in the project (Gymnasium Dresden Plauen). The analysis was not intended to be representative for Germany in general and also not to be representative for all pairings of secondary school types and university types. Nevertheless, the identified gap between secondary school and university educations in our particular pairing relates close to our experiences with student from other secondary schools and the experiences of colleagues in other Universities. In Germany, secondary schools provide a broad education without an intense programming focus.

Further information on school education in the area of informatics can be found at the following link

- <https://gi.de/meldung/404-informatikunterricht-in-deutschland> and
- <https://informatik-monitor.de/landing-page>.

Analyzing details of teaching and learning at the Gymnasium Dresden Plauen and Dresden University of Applied Science we came to the following conclusions:

- During the high school education concepts related on "programming concepts" and "data structures", extremely little attention is paid (placed in the 11th and 12th grade and cover 14 and 10 teaching hours respectively)
- The quality and scope of the material delivered to the students depends on the enthusiasm, knowledge and skills of the specific lecturer
- The school organizes weekly project sessions after school classes for interested students

Despite these discouraging facts (the first two), it is almost unbelievable that only on the basis of weekly project sessions after school classes for interested students the results of the projects are comparable to university education.

Based on this fact, we believe that it is very important that in the continuation of this project we focus more on the content of these projects, analyze their curriculum and the way in which knowledge related to programming is transferred to students.

The vertical analysis of Republic of Serbia is very detailed, covers prior related work in the analysis of informatics education in secondary schools. In Serbia, the quantity and quality of programming education in schools is higher compared to other countries (as described in the vertical analysis). However, even when freshmen students in Serbia start with more prior knowledge on programming compared to other countries, a survey reveals that only a minority have in-depth knowledge of object-orientated programming, and a majority is not able to answer all questions correctly.

Analyzing details of teaching and learning at the Gymnasium Ivanjica and Faculty of Organizational Sciences of the University of Belgrade (**Serbia**) we came to the following conclusions:

- In general, secondary schools, as well as social secondary schools, there is a Computer Science and Informatics subject that is implemented in the first three years, with a total of 2 hours per week, i.e. 72 hours per year, i.e. a total of 216 hours.
- In specialized IT departments (a total of 49 schools enrolled students in 2020/2021) students have a total of 934 hours of IT subjects in four years.
- This number of classes, the contents studied, the division of classes into smaller groups, professional teachers and equipped schools represent conditions for high school graduates that will be extremely effective during further studies in IT majors of various faculties.
- There are no requirements for the mandatory definition of prerequisites for enrollment to universities, which are related to the computer and more specifically programming skills of future university students
- Teachers of courses from the first year of study do not expect students to have prior computer knowledge
- In general, there are not many professors who are educated, educated to teach students the basics of programming, usually they are teachers who do not have a full fund of classes in subjects such as Mathematics.

Analyzing details of teaching and learning at the OAPB and Faculty of Management Science and Informatics of the University of Zilina (**Slovakia**) we came to the following conclusions:

- The main focus of the OAPB is not to prepare future students for informatically oriented study programs of universities, which means that topics related to informatics are taught in general
- The number of teaching hours is not sufficient to cover all the problems in the field of OOP in more detailed way.

- Not all graduates continue their studies further at a university. Furthermore, only several graduates, who continue their studies at a university, select study programs related to informatics.
- The teaching itself at OAPB is in the inconsistency of the used programming languages (Python the 3rd year and Java in 4th year) or the fact that the teaching of the OOP starts at the end of 3rd year.
- A good preparation for university is provided with special classes and seminars in the vocational school, and with special events that are organized by the university.

Croatia presents a very detailed analysis and first refers to the Croatian qualification framework that defines standards through sets of competences and sets of learning outcomes for schools and universities. A specific in Croatia is that particular exam subjects and levels of the final school exam (matura exam) are required to be able to enroll in a desired study course. While the transition from school to university is regulated more than in other counties there are still shortcomings that are observed regarding the knowledge freshman students.

During the analysis of subjects related to programming in high school Ivanec and Faculty of Organization and Informatics of University of we emphasize the following findings:

- Similar to the Serbia in Croatia, there are no requirements for the mandatory definition of prerequisites for enrollment to universities, which are related to the computer science and more specifically programming skills of future university students.
- Teachers clearly stated that basic IT literacy as well as knowledge of logic and mathematics will help students to go through university curricula.
- Teachers also proposed some recommendations to obtain a higher level of knowledge and skills in programming for high school students. They mentioned that a new curriculum should be developed, but more importantly, a new legislative framework to cover some other issues, such as to improve the motivation for teachers with programming knowledge to come and teach in high schools.
- From the curricula of IT subjects in high schools, OOP is poorly represented, with an insufficient number of teaching hours.

Although analyses are not homogenous and the quantity of information and the generality of the collected information differ, one can draw the following conclusions.

- Most study programs teach programming from the beginning, because of the lack of a homogenous prior education of students.
- A lot of activities are already common practice to prepare school pupils for university courses, particularly for programming. These are intended for those who enroll at universities in STEM-related study programs. How good this works depends strongly on individual interest, choice of non-mandatory subjects in school and the later choice of the study program.
- There are differences among the countries how much programming and OOP content is taught in secondary schools and in vocational schools.

After analyzing the conclusions, we reached for each of the countries, we emphasize a few important facts:

- We noticed similar problems in Croatia and Serbia, as well as Slovakia and the Czech Republic. In Serbia and Croatia there is a significant problem with good staff teaching computer science

and mathematics. In Serbia students that is on the last year of the studding encouraged to apply as teachers in the high schools. In Slovakia and Czech Republic, a very small number of high school students (20% of graduates of gymnasium) study technical and informatically oriented universities, while, on the other hand, in Croatia and Serbia students are very interested in studding IT faculty oriented.

- Despite the fact that there are specialized IT schools, In Serbia, there is no adequate recommended literature for students.
- The number of programming classes in secondary schools is negligible except in specialized IT schools in Serbia that is specialized IT school at the Gymnasium in Ivanjica.
- For each of the analyzed countries the rule is: there are no defined prerequisite, the level of programming as well as OOP knowledge, required as input to the universities cannot be established and determined for enrollment to computer science universities.

Bearing in mind the similarities and differences between the countries we can emphasize that

- Education in the field of information technologies is usually associated with colleges and higher schools, but practice has shown that the training of future IT professionals should be started much earlier. High schools, need to recognize this need and open their doors to a new generation of future computer professionals, the generations that grew up with Facebook, Google and other Internet services.
- We believe that for the effective improvement of programming learning, the starting point and basis should be quality learning materials, which are further supported by teacher trainings, classroom work and independent work of students.

Although analyses are not homogenous and the quantity of information and the generality of the collected information differ, one can draw the following conclusions.

- Most study programs teach programming from the beginning, because of the lack of a homogenous prior edudation of students.
- A lot of activities are already common practice to prepare school pupils for univeristy courses, particularly for programming. These are intended for those who enroll at universities in STEM-related study programs. How good this works depends strongly on individual interest, choice of non-mandatory subjects in school and the later choice of the study program.
- There are differences among the countries how much programming and OOP content is taught in secondary schools and in vocational schools.