



CURRENT STATE OF PROGRAMMING IN HIGH SCHOOLS AND UNIVERSITIES - VERTICAL ANALYSIS IN CROATIA

Davor Fodrek

Lidija Kozina

Zlatko Stapić

Ivanec, August 2024



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

Authors	Davor Fodrek, High School Ivanec Lidija Kozina, High School Ivanec Zlatko Stapić, Faculty of Organization and Informatics Varaždin
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	August 2024

Table of contents

1. Analysis.....	7
1.1. Legislative framework	8
1.1.1. Law on the Croatian qualification framework.....	8
1.1.2. National qualification and occupation standards	9
1.1.3. National curriculum for preschool, elementary and high school education.....	10
1.1.4. Curriculum for informatics subject in elementary and grammar schools.....	10
1.1.5. Rulebook on taking the state matura exam	11
1.1.6. Curricula for programming related subjects in university studies	12
1.1.7. Conclusion on legislative analysis.....	13
1.2. Gap analysis.....	13
1.2.1. Freshmen students' prior competencies and knowledge analysis.....	14
1.2.1.1. Introduction.....	14
1.2.1.2. Data analysis.....	15
1.2.2. Teachers' expectations – semi-structured interview	29
1.2.2.1. Introduction.....	29
1.2.2.2. Interview design	30
1.2.2.3. Conducting the interview	33
1.2.2.4. Deciding on analysis method.....	34
1.2.2.5. Case study 1 – experienced teacher.....	35
1.2.2.6. Case study 2 – young teacher.....	36
1.2.2.7. Case study 3 – teaching assistant / student demonstrator.....	38
1.2.2.8. Comparing cases.....	39
2. Conclusion on gap analysis.....	42

List of tables

Table 1 – Review of teachers’ experience.....	40
---	----

List of charts

Chart 1 - Distribution of respondents by subjects.....	16
Chart 2 - Distribution of respondents by type of high school	16
Chart 3 - Number of years taking informatics and related subjects in high school	17
Chart 4 - Number of years with programming contents within informatics subjects in high schools.....	18
Chart 5 - The way students acquired their programming knowledge	19
Chart 6 - Programming concepts that high school students are familiar with and have practical experience	19
Chart 7 – Usage of programming languages by students in high schools.....	20
Chart 8 – Teamwork experience among students in high schools.....	21
Chart 9 – High school students' self assessment of programming knowledge.....	22
Chart 10 – How the students recognized the definition of an algorithm	23
Chart 11 – Recognition of basic algorithmic structures	23
Chart 12 – Results of three tasks with code analyses	24
Chart 13 – Distribution of students who are/are not familiar with basic OOP concepts	25
Chart 14 – Comparison of distributions of respondents by schools (all respondents) and respondents by schools familiar with OOP concepts	26
Chart 15 - How the students recognized the definition of a class	27
Chart 16 - How the students recognized the definition of an object.....	27
Chart 17 – Understanding definitions of basic OOP concepts	28
Chart 18 – Characteristic of an abstract class	28
Chart 19 – Class/object relationship statements	29

List of pictures

Picture 1 – Introduction to the questionnaire in Croatian language	14
Picture 2 – Example of code section with iteration	24
Picture 3 - Responses – Semi-structured interview with teachers	30

1. Analysis

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.

1.1. 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

1.1.1. 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>

1.1.2. 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.

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

¹ source: Methodology for creating occupational standards and sets of competencies, retrieved on July 30, 2024 <http://www.kvalifikacije.hr/sites/default/files/documents-publications/2021-12/Metodologija%20za%20izradu%20standarda%20zanimanja%20i%20skupova%20kompetencija.pdf>

² source: Croatian qualification framework, retrieved on July 30, 2024, <https://hko.srce.hr/registar/standardi>

1.1.3. 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>

1.1.4. 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:

- description of informatics subject
- educational goals of learning and teaching the curriculum in informatics
- domains in the organization of the informatics subject curriculum

³ source: National curriculum, Ministry of Science and Education, retrieved on July 30, 2024, <https://mzo.gov.hr/istaknute-teme/odgoj-i-obrazovanje/nacionalni-kurikulum/125>

- 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>

1.1.5. 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>

1.1.6. 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.

1.1.7. 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.

1.2. 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

1.2.1. Freshmen students' prior competencies and knowledge analysis

1.2.1.1. 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 1.



Picture 1 – 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](#)

1.2.1.2. 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 1.

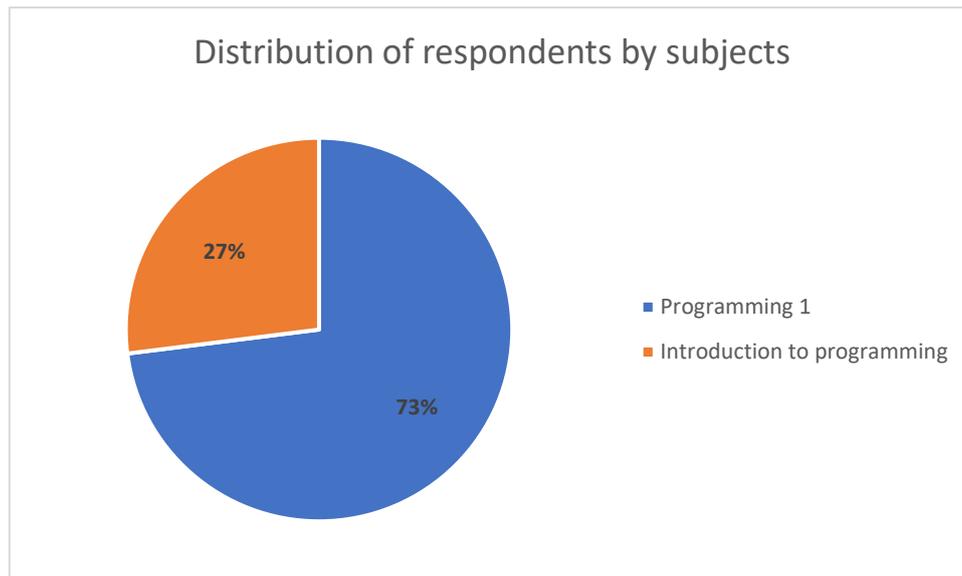


Chart 1 - Distribution of respondents by subjects

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 2.

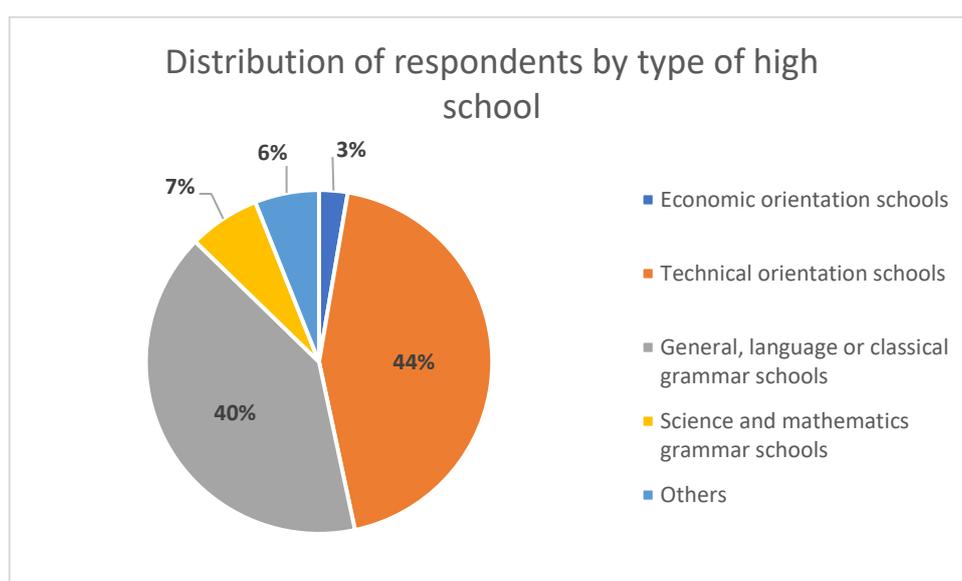


Chart 2 - Distribution of respondents by type of high school

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 3.

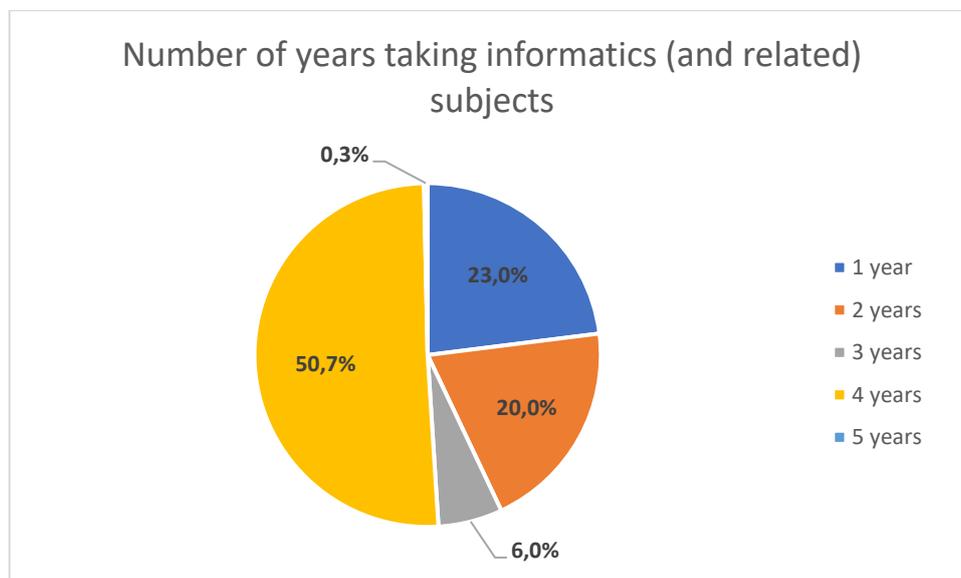


Chart 3 - Number of years taking informatics and related subjects in high school

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 4. 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.

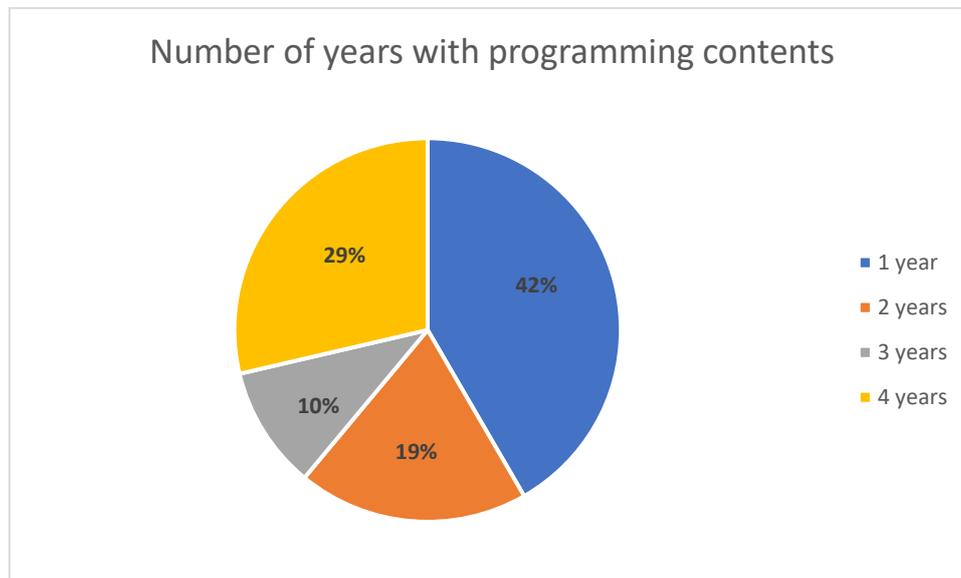


Chart 4 - Number of years with programming contents within informatics subjects in high schools

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 5.

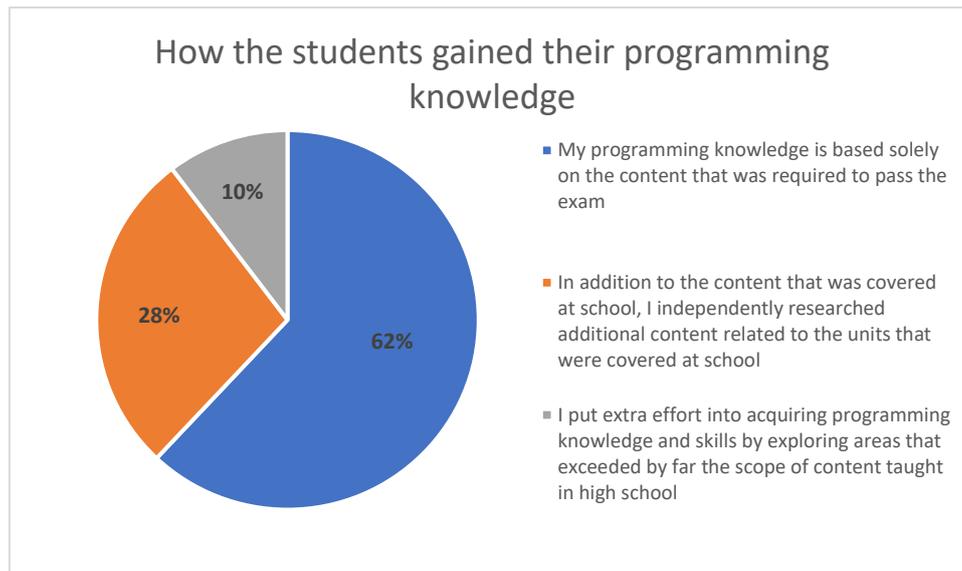


Chart 5 - The way students acquired their programming knowledge

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 6.

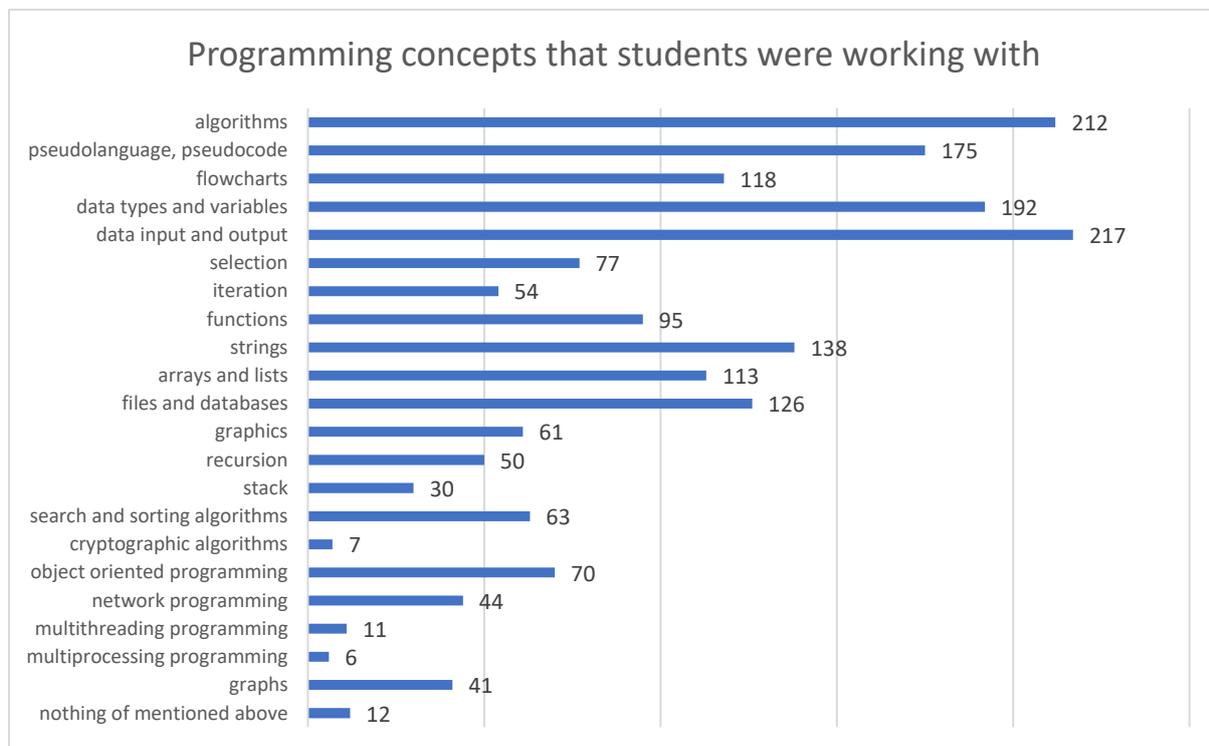


Chart 6 - Programming concepts that high school students are familiar with and have practical experience

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 7.

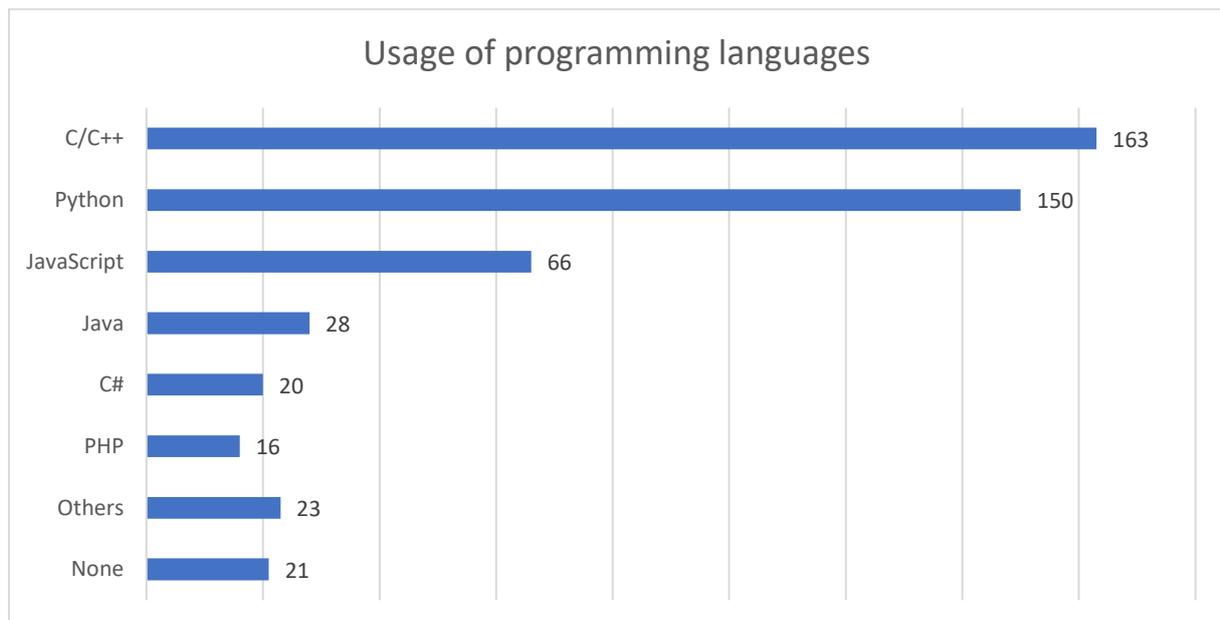


Chart 7 – Usage of programming languages by students in high schools

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 8. 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.

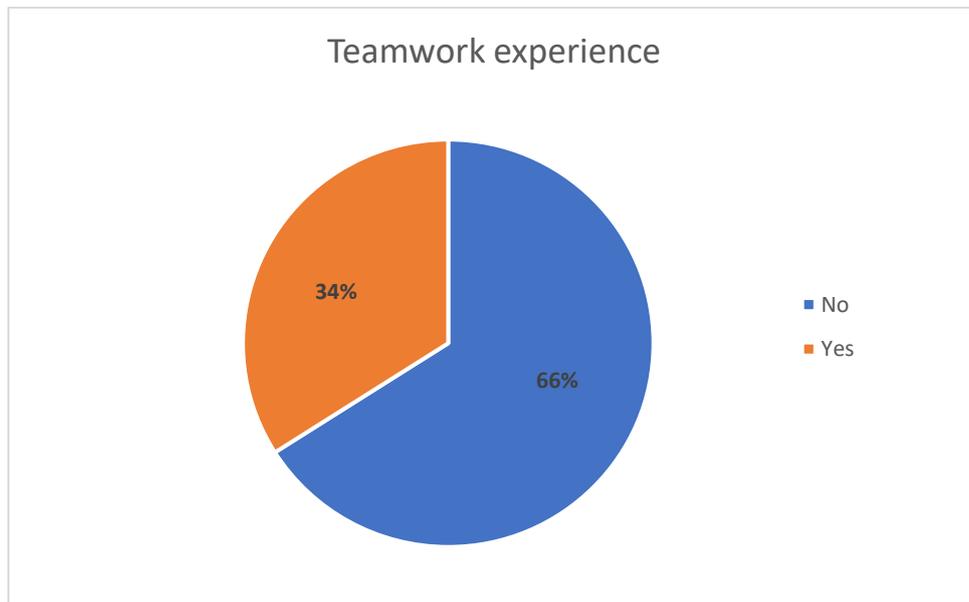


Chart 8 – Teamwork experience among students in high schools

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 9.

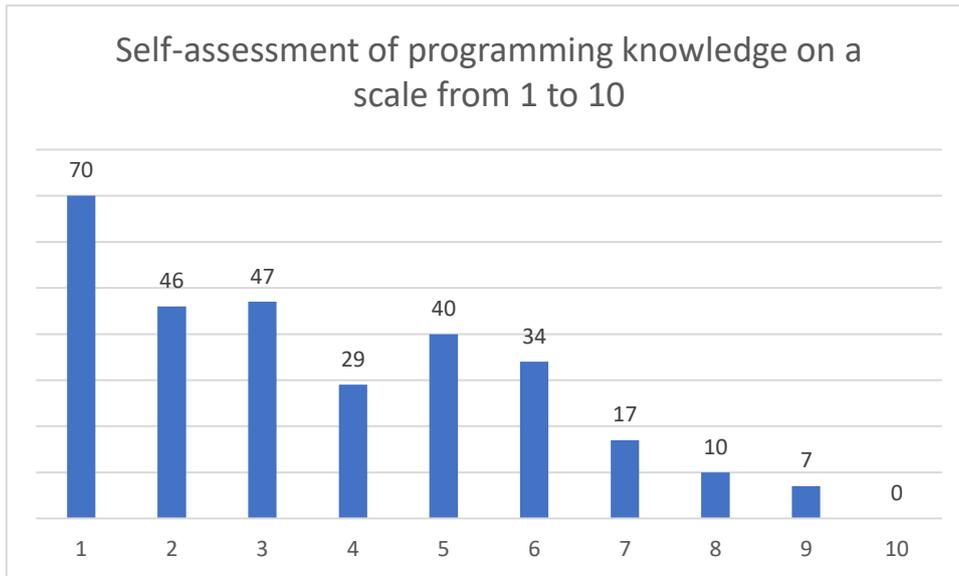


Chart 9 – High school students' self assessment of programming knowledge

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 10.

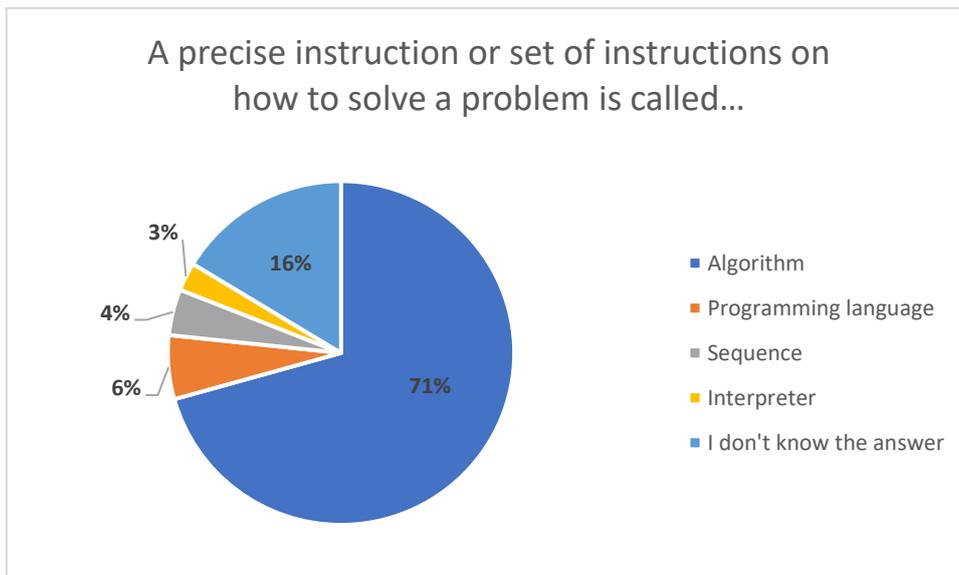


Chart 10 – How the students recognized the definition of an algorithm

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 11.

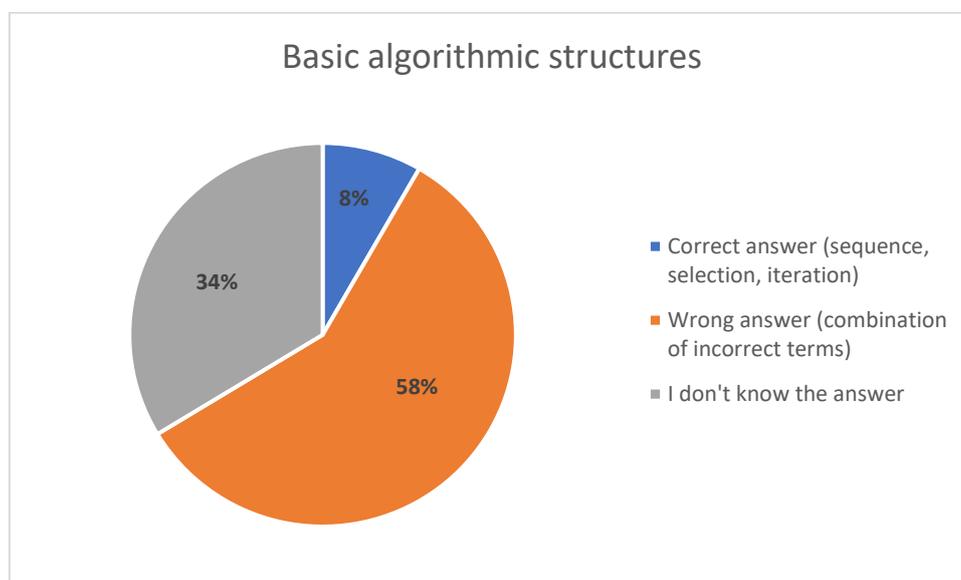


Chart 11 – Recognition of basic algorithmic structures

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 2 and the results are displayed in Chart 12.

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

Picture 2 – Example of code section with iteration

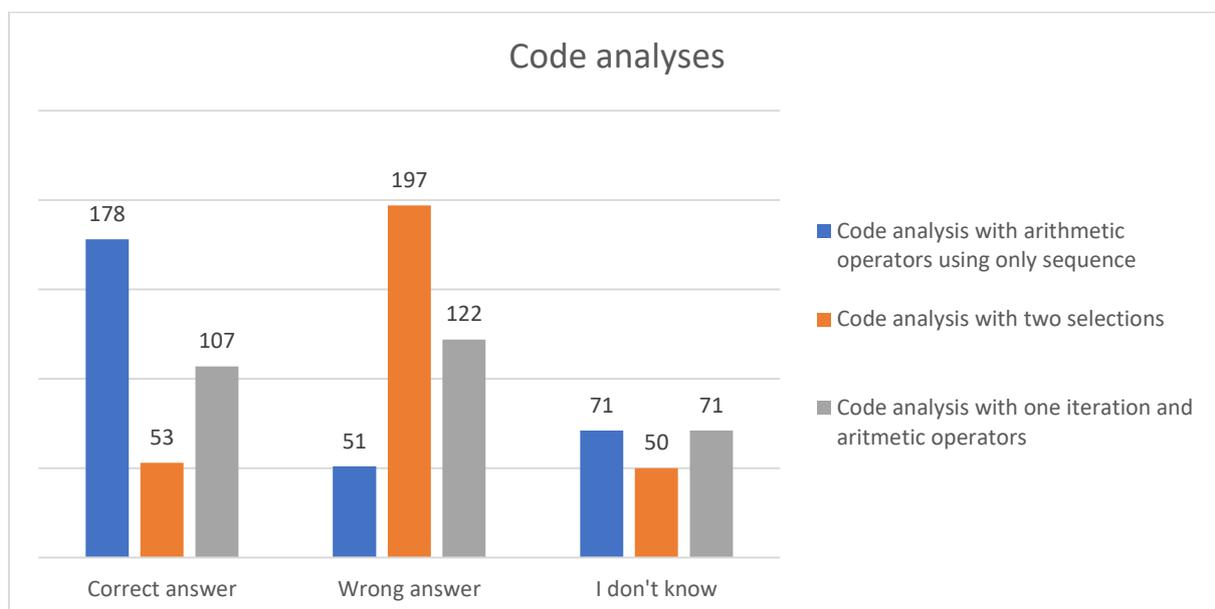


Chart 12 – Results of three tasks with code analyses

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 13.

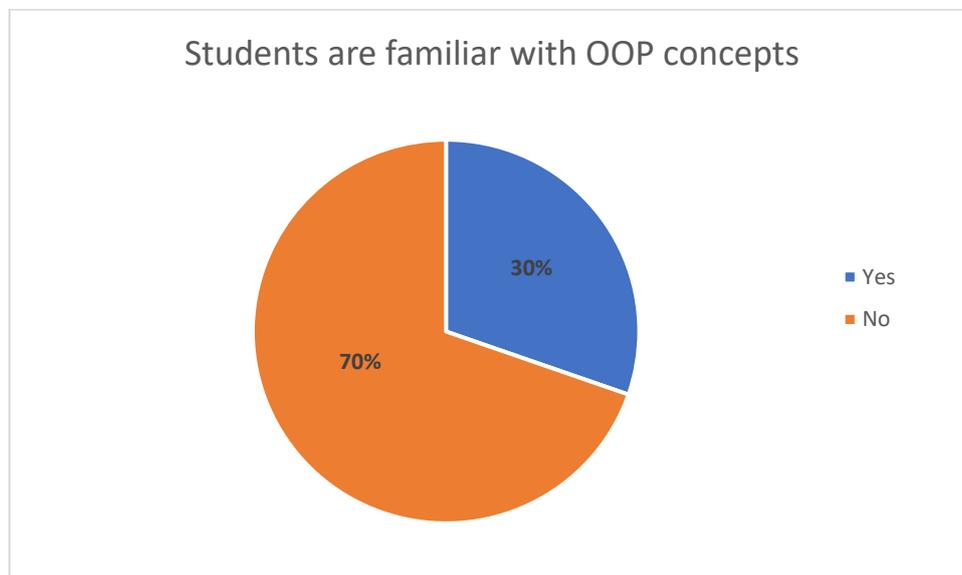
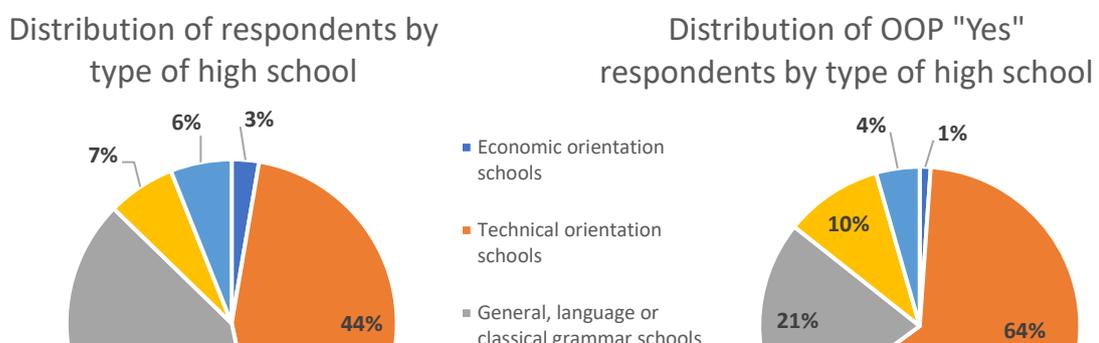


Chart 13 – Distribution of students who are/are not familiar with basic OOP concepts

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 14.



<http://www.kvalifikacije.hr/hr>

<https://hko.srce.hr/registar/>

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

Chart 14 – 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 15 and 16. The students were given the definitions of class and object and they should have recognized which term has been described.

A set of real-world entities that have similar properties and behaviors is called...

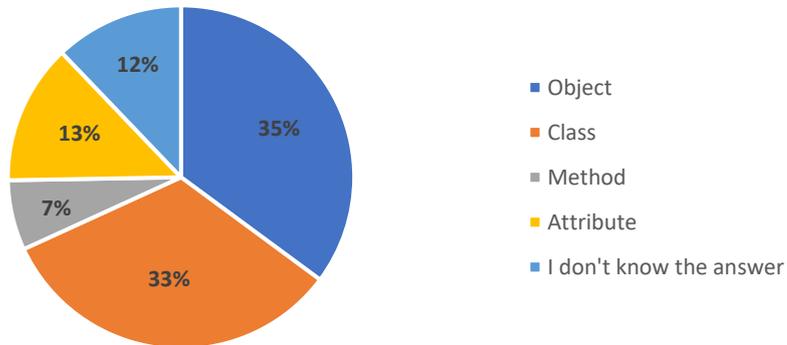


Chart 15 - How the students recognized the definition of a class

A concrete entity from the real world in object oriented programming is called...

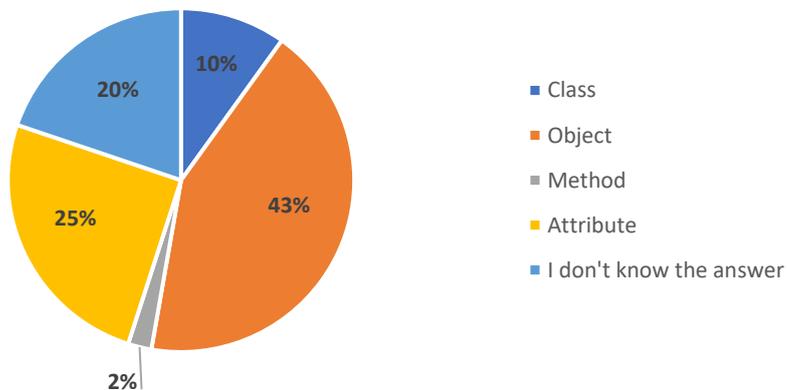


Chart 16 - How the students recognized the definition of an object

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 17.

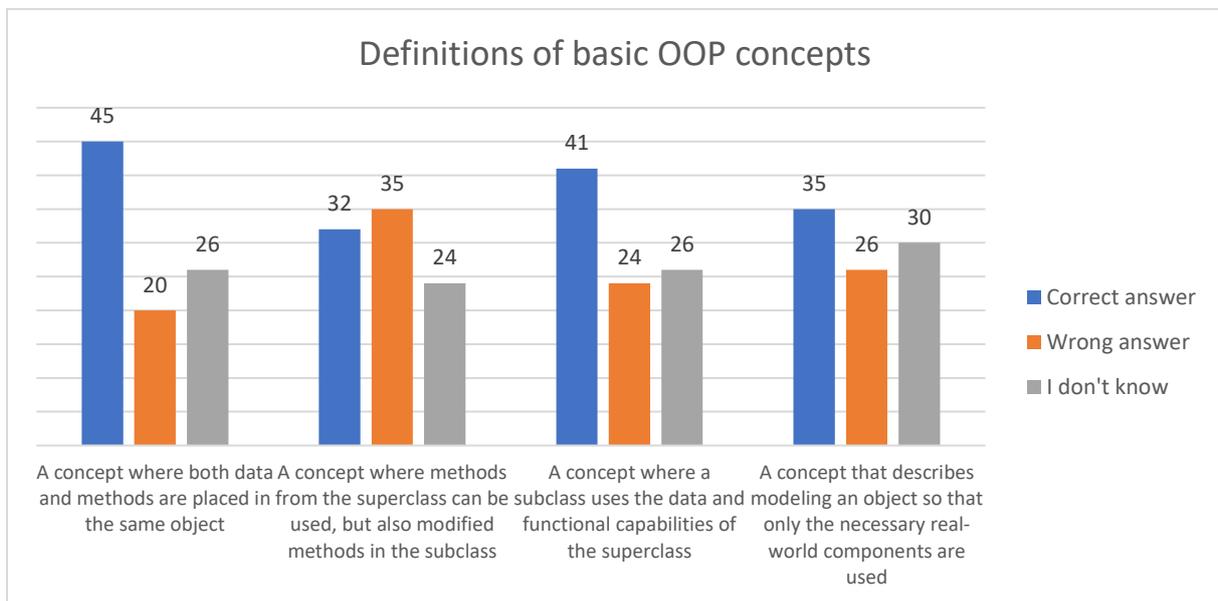


Chart 17 – Understanding definitions of basic OOP concepts

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 18.

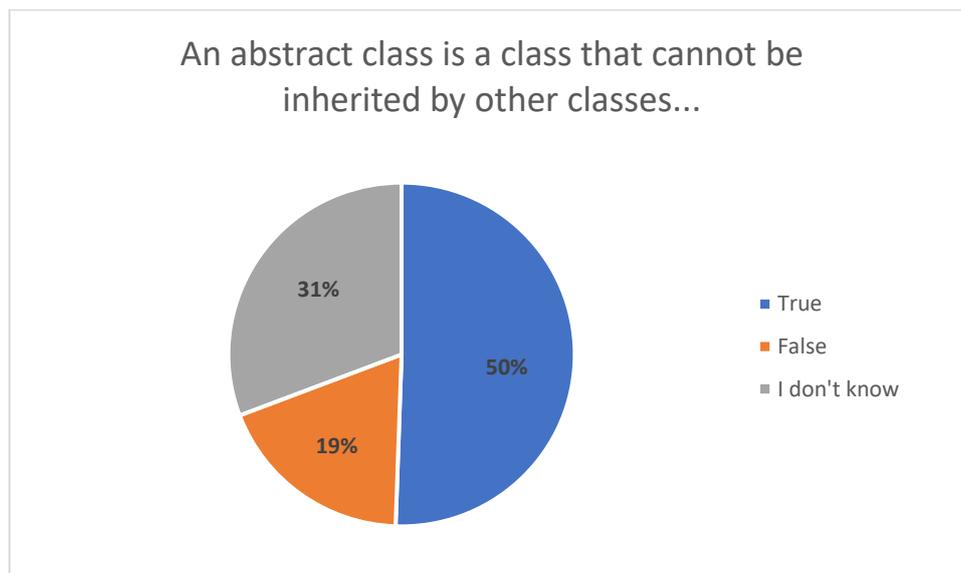


Chart 18 – Characteristic of an abstract class

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 19.

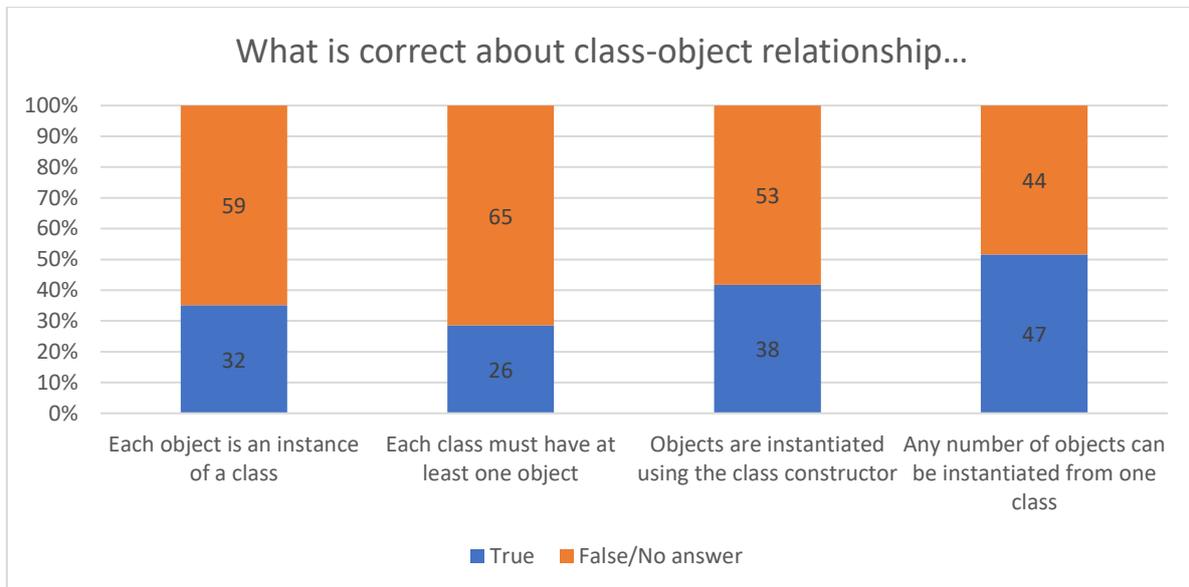


Chart 19 – Class/object relationship statements

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.

1.2.2. Teachers' expectations – semi-structured interview

1.2.2.1. 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

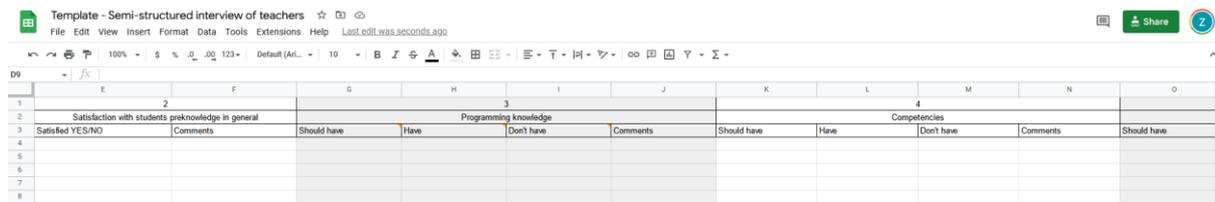
prepared a semi-structured interview to be carried out with these teachers. In this chapter we will bring you the information on the results obtained.

1.2.2.2. 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 3 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 3 - 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.

1.2.2.3. 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 (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.

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.

1.2.2.4. 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.

1.2.2.5. 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.

1.2.2.6. 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.

1.2.2.7. 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 dissatisfaction 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.

1.2.2.8. *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 1.

Table 1 – 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.	<p>Current curriculum of (introductory) programming is built with a prejudice that students do not have any prior-knowledge.</p> <p>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.</p>	
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.	
Prior knowledge in related subjects	<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>	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.	Mainly students who finished technical high schools actually have some kind of programming knowledge whatsoever.
Prior knowledge in programming	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.	When enrolling the university, student should be familiar with variables, data types, selections, iterations and functions.	<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</p>

			use of variables, selection, iteration and functions.
Prior competences and skills	Only up to 10% of enrolling students can find largest element in array with the high school knowledge.	Freshmen students in general don't possess any notable competences or skills related to programming	Student-to-student experience shows that most of the students struggle with the very basics of programming concepts. Students with better prior knowledge are usually overconfident and stop actively attending classes.
How to overcome shortcomings in knowledge	Students in high school should learn basic programming skills and habits, such as loops, selections, process and basic data structures. Good programming practices which include syntax checking, testing requirements, and going through all code blocks should be acquired in high school.	Students are capable of self-learning algorithmic problem solving if provided with good materials.	
	Utilizing a tool like Alice can help those who have not been exposed to structural programming learn OOP.	Education of high school teachers is very important.	

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.

2. 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.