Informatics in Education, 2016, Vol. 15, No. 2, 163–182

DOI: 10.15388/infedu.2016.09

Introductory Programming Subject in European Higher Education

Veljko ALEKSIĆ[1], Mirjana IVANOVIĆ[2]
[1]*Faculty of Technical Sciences, University of Kragujevac*
Svetog Save 65, Čačak, Serbia
[2]*Faculty of Sciences, University of Novi Sad*
Trg Dositeja Obradovića 4, Novi Sad, Serbia
e-mail: veljko.aleksic@ftn.kg.ac.rs, mira@dmi.uns.ac.rs

Received: January 2016

Abstract. Programming is one of the basic subjects in most informatics, computer science mathematics and technical faculties' curricula. Integrated overview of the models for teaching programming, problems in teaching and suggested solutions were presented in this paper. Research covered current state of 1019 programming subjects in 715 study programmes at total of 218 faculties and 143 universities in 35 European countries that were analyzed. It was concluded that while most of the programmes highly support object-oriented paradigm of programming, introductory programming subjects are mainly based on imperative paradigm.

Keywords: programming, teaching, higher education.

1. Introduction

Framework for teaching specific programming language and/or appropriate tools at higher education is primarily defined by the competencies that students need to master and teaching methodology and learning model that are defined by the particular institution and teacher. While programming is seen as a single activity for specific domain, in reality it presents different sequence of activities directed towards wide specter of problem solving. Term Programming in informatics/computer education is widely accepted as gaining and application of problem solving skills, so teachers feel that development of those should be the foundation in curricula creation. Tremendous constant development of computer science and ICT creates permanent problem of continuously changing programming languages, techniques and tools within programming courses. However, there are important concepts that stay as essential/fundamental over the time, such as control structures, procedures, pointers, dynamic variables, parameter passing techniques, etc.

Programming in the narrow sense represents the activity of problem understanding and analysis, application of adequate algorithms and its implementation in the form understandable to the computer. Most common method of implementation constitutes a set of instructions using programming language such as C, C++, Java, Python, PHP etc. Depending on the problem, in real environments these activities can include one or more programmers with different knowledge and specializations.

Variety of programming languages are available for study in the introductory programming, so unique choice of language would be at least controversial. Different computer techniques and programming languages and tools have been developing rapidly over the last two decades, so programmers in their career have to use more different programming languages than ever before. Although the adoption of practical programming skills is the base for teaching certain programming languages, it is more important to highlight that essential understanding of basic theoretical concepts and critical thinking are more important to novice programmers so that they can prepare for later studying of new languages and tools. Fast technology changes more or less influence teaching of introductory programming courses due to the need for teaching content to be up-to-date. Programming courses are often perceived as one of the "most difficult" classes in higher education. As Comenius (1957) defined didactical triangle of factors in teaching process, problem of teaching programming can actually be observed from the angle of content, teachers and students.

According to Rahmat *et al.* (2012) content that is being studied is new for most of students, which makes the subject difficult. Programming is a hierarchical skill in which mastery students gradually acquire new knowledge by increasing complexity. For example, student must first master the basic syntax, and then semantics, structures and programming styles. Curricula's are often very demanding and extensive, so students do not have the time to master the content before a new one is introduced. The programming process begins by translating the algorithm into the code. Before that, the hardest part is to analyze and understand the problem, and then to translate the required specification into algorithm. The correct algorithm is the basis of the correct program code, so students have to be able to design and translate algorithms and know the proper program syntax. Most programming languages are designed for the industry – not for teaching, hence the choice of the "proper" educational programming language is difficult.

Students' interest and motivation are important factors as well. While certain elements may be affected by the teacher, motivation and learning styles are completely individual. To ensure efficient knowledge transfer, teachers must be adapted to conditions such as group size, timing and duration of the lectures.

The rest of the paper is organized as follows. Section 2 presents a closer examination of different perspectives on adequate paradigm and programming language selection in introductory programming courses. Further, we emphasized some of the main problems and possible solutions in programming teaching practice. An empirical study presents the variety of introductory programming languages taught in European tertiary education institutions, and its structure were further discussed in Section 4. The last section gives final remarks on the topic.

2. Selecting the First Programming Language

During the long history of teaching programming skills and languages there has been developed different criteria for the selection of the first programming language that should be respectable, appropriate and accepted by teachers and students (Kaplan, 2010; Mclver and Conway, 1996; Parker *et al.*, 2006; Thomas *et al.*, 2002). Meanwhile, three clearly distinguished paradigms are recognized and separated: imperative (or procedural), object-oriented and functional.

2.1. *Programming Paradigms*

Although different programming languages can successfully implement imperative (procedural) paradigm, typical representatives are C, Pascal, Basic etc. Top-down design methodology for program development is the basis of this approach. Weisert (1997) stated that programming building blocks are procedures, while functions are created to realize smaller tasks and communicate with other functions through parameters, arguments and return values. Main advantage of procedural approach lies in its simplicity. According to Deitel and Deitel (2011) it is not necessary to introduce students to multitude terms in the beginning, so teachers may begin with syntax and logic right away. However, procedural approach has clear drawbacks. Modern trends in methodology of teaching programming languages are more and more oriented to object-oriented paradigm (Böszörmenyi, 2005). According to Ivanović and Budimac (2013) "it is extremely important to introduce essential/key concepts of programming like: procedures, records, pointers, as a good base for better understanding of concepts in object-oriented, Parallel, WEB programming which will be taught later in subsequent courses".

With object-oriented paradigm, program design is focused on objects, their attributes and relations and interactions between them. Typical object-oriented programming languages are C++, C#, Java, Python etc. Main advantage of teaching object-oriented programming approach for introductory programming course is that students learn consistent methods of program development that are widely used in modern software industry (Chen, 2004). Other advantages are simple program modification, extensibility through inheritance and the fact that same objects can be reused in different programming solutions (Wang, 2003). Due to its relative high complexity, this approach can be difficult to be realized in introductory programming courses. Wide range of new diverse concepts and terms must first be introduced, often with rather complex examples, so the syntax is a little bit postponed to be taught later in the course. However, there is an increasing number of faculties that apply object-oriented as a first approach in introductory programming courses by using Java programming language. Even though this choice of programming language has many advantages, it should be bared in mind that basic programming concepts are applied mostly indirectly. This can be the reason that significant number of Universities today still implement "natural" sequence of learning the essential programming concepts first (with C), which are later upgraded through object-oriented approach and appropriate programming languages (C++ or Java).

Functional paradigm is rarely used in introductory programming courses, as it lacks control over the exact way in which the computation is carried out, and is subsequently more suitable for advanced programming. These languages are usually chosen for theoretically/mathematically oriented study programmes. Typical modern representative of functional programming paradigm is Haskell programming language.

Visual approach in teaching introductory programming is also one of the methodical challenges. It is based on usage of graphical user interface from the beginning (Bradley *et al.*, 2002). Every programing example is explained and program input/output is shown using graphical objects (panels). Even though this approach could be preferred as it relies on modern operating system interfaces, its realization demands extensive time to explain the user interface instead of teaching essential programming concepts.

According to Bordini *et al.* (2006) Integrated Development Environments (IDEs) focus on the programming language level and intend to enhance the productivity by automating tedious coding tasks. Looking at current IDEs for the object-oriented domain it can be seen that such IDEs tend to provide functionalities that can be classified into five categories:

- Project management (organizing the project structure according to developers' needs).
- Creating and editing source files (providing structure views for quick and easy navigation, online error detection, auto-completion, and so on).
- Refactoring to enable fast and reliable code restructuring operations.
- Build and run process allowing the execution of applications from within the IDE.
- Testing developed programs (supported by unit testing with test cases).

Despite the benefits, which are reflected in the multitude of resources that facilitate course realization, the main challenge for teachers who teach introductory programming courses is how to teach problem solving and develop critical thinking by using algorithms. Polya (1957) observed problem-solving process through four stages: understanding the problem, solution planning, its execution and reflection on the results. Introductory programming should assist students in acquisition of problem solving skills parallel to studying principles of particular programming languages.

Freund *et al.* (2010) stated that the availability of so many languages and models means that students are supposed to make proper choices about which programming language to use for solving specific programming tasks. Even computer-integrated manufacturing systems are now rarely built entirely in only one language. Instead, they are the composition of various components, each written in a language chosen for its strengths in that component's particular problem domain. For example, a web application may include database queries written in SQL, server application logic written in Java, data transformers written in XSLT, and client-side code written in JavaScript.

Key finding of Meyerovich and Rabkin (2013) research was that in order to improve developer versatility, students should be exposed to diverse language families while they are still at school. Past school experience moderately influenced the languages that respondents knew later on. The vast majority of respondents know C or Java, regardless of study programme or curriculum. Education leads to increased likelihood of knowing a language, but only within a 10% increase. Less-popular language families (assembler,

functional, and mathematical languages) are much more sensitive to the form of prior education. Developers usually learn new languages throughout their career, but are less likely to learn new paradigms.

Jablonowski's case study (2007) illustrated clear drawbacks of using Java (and indirectly C++) as introductory programming language, even though it is a great language with lots of applications and powerful IDE's. It was concluded that "the language of choice" for introductory programming course could/should be Pascal.

The authors of this paper support this conclusion, in fact the need to teach introductory programming using some of the educational programming languages.

2.2. *Teaching Programming*

According to Gomes and Mendes (2007) there are various types of problems in teaching programming, such as teaching methods, learning styles, students' skills and attitudes and psychological effects.

Traditional teaching methods do not seem to be adequate when it comes to meeting the needs of students. Individualized instruction model represents one of the ideal ways in teaching programming, but in reality it is not feasible due to time and financial constraints. One solution could be the usage of a computer tutor and specially developed personalized tutoring system (Butz *et al.*, 2004; Vesin *et al.*, 2012, 2013). Although it could not completely replace the teacher, it could have a positive effect on students' motivation. Teaching strategies often do not consider all the learning styles of students. In the traditional face-to-face teaching all students must advance at the same pace and in line with teachers' pedagogical strategies that are based on teachers' learning style. This problem can be overcome by creating various presentation formats for each activity and by adapting interaction to students' characteristics. At the beginning of the course, students should complete one of the variety of questionnaires based on the Kolb's (2005) or Felder-Silverman's (1988) learning style model. The survey results can later be used in the LMS environment implementation in order to try to prepare teaching material more or less according to well-known learning styles. The purpose of the introductory programming course should be to improve and enhance students' programming skills. However, students and teachers are often more focused on pure syntax and technical thinking. On the other hand, usual opinion is that programming language should be used only as a tool to realize ideas and algorithms. Hence, the choice of "first" programming language should be based on "pedagogical benefits" rather than on its popularity.

Many students have learned to solve problems in other disciplines by memorizing formulas and procedures. They often do this without a complete understanding of the essence of concepts and problems. Since the programming is oriented to critical thinking, problem solving, but also gaining practical skills, students often mistakenly believe that it can be mastered mainly by studying textbooks without intensive practicing. This can be prevented by providing frequent opportunities during regular classes as well as in order to solve practical and real problems.

Problem solving requires a variety of skills that students often do not ad hoc possess sufficiently: understanding the problem, knowledge transfer, reflective thinking and perseverance. The teacher should ask the students to predict the results of certain activities, set new tasks to address some of the procedures used in previous solutions etc. Learning how to program requires abstraction and generalization abilities, and critical thinking. Accordingly, it is important to provide an environment that facilitates the development of these students' capacities. It is important to recognize patterns in various problems in order to develop generalizations. Programming languages are mostly developed for professional use, which result in notably complex structures and syntax. This fact can be prevented by using high quality programming environments and tools that allow students to be focused on solutions, not on syntax.

Many secondary school students lack motivation to enroll CS studies due to the negative general opinion about programming that is spread among them. Jenkins (2002) emphasizes that there is a public image of the programmer as socially inadequate "nerd". If students take the predisposition that course will be difficult, it is hard to imagine that they will be motivated to work, and students that are not motivated are often not successful (Ng and Bereiter, 1991). In order to increase the motivation, it is useful to implement a multimedia environment and learning through playing and gaming. It is important to show students how to use programming and general ICT knowledge in order to facilitate the society's quality of life (Ulrich and Karvonen, 2011). As the introductory programming subject is mandatory for many other related subjects, it needs to be included in the curriculum as early as possible, ideally in the first or second semester.

Azad and Kohun (2009) point out many challenges facing teachers in subject realization such as: the different students' background, programming seems difficult and complicated for most students, too much time is spent in the study of language syntax, inability to see results before correcting errors, lack of motivation, most programming environments were designed for professional use and as such are impractical for beginners' usage. It is difficult to propose a simple solution to overcome these challenges.

The first and rather crucial obstacle in teachers' work is a proper choice of first programming language because the teaching of syntax is one of the initial and basic pedagogical goals. According to Kolling (1999), choice of programming language is less important than the question as to what to teach, and finding balance between mastery of syntax, design and problem solving skills.

Programming courses need to be constantly innovated to accommodate the new generations of students who have grown up with computer technology, games and interactive user interfaces (McKenzie, 2009). If students have a positive experience when they first face programming activities and processes they are motivated to acquire good programming habits and develop programming skills through self-learning (Pendergast, 2006).

3. Related Work

In spite of the importance of the topic, relatively few studies empirically analyze usage of programming languages within first programming courses in higher education institutions.

Nowadays, curriculum developers can be divided into four groups concerning the attitude towards selection of first programming paradigm and language. The first group is in favor of object-oriented paradigm through Java, C++ or C#; second group are those who support the imperative style using C, Pascal or Modula-2; the third group advocates functional programming through Haskell; and the fourth group, which is in favor of competitive and script programming languages such as Python (Watt, 2004). Just as there is a disagreement over the choice of programming language, there is a disagreement about the teaching methods, especially among advocates of object-oriented approach. One group of teachers advocate the early introduction to objects, classes and inheritance, while the others argue that it is better to first study procedural aspects of the programming language first and then proceed with object-oriented concepts.

Chalk and Fraser (2006) did a research which included 44 higher education institutions in UK and concluded that 60% of 4800 surveyed students were taught Java as first programming language, while 15% of them studied C++ and C#. Python and Haskell together were represented by 6%.

Mason *et al.* (2012) published results of their survey in 2010 which covered 28 Australian universities and 44 introductory programming curricula. It was stated that 20 various programming languages were taught among which the most common was Java (36%), following Python (19%) and C (12%). Considering the results of the previous 2003 and 2006 surveys, they noticed a trend of slight usage decrease of the Java (from 40% to 36%), and significantly Visual Basic (from 25% to 9%) and C++ (from 14% to 7%). The biggest breakthrough was done by Python, which was statistically insignificant in previous surveys. Part of their study analyzed reasons for the choice of first programming language and concluded that the crucial factor was no longer industry usage, but the pedagogical benefits it provided. As the paradigm, 55% of courses were based on procedural, 25% were based on object-oriented, 2% were based on functional, and the rest 18% applied some combination of paradigms. Most popular IDE's used in first programming courses were Visual Studio (28%) and BlueJ (18%).

4. Results and Discussion

The main objective of our research was to establish a clear empirical answer on the question of which programming languages are mainly taught at European tertiary education institutions, especially in introductory programming courses. The research sample consisted of 143 European representative universities out of 35 countries. In total, 218 faculties with 715 study programmes and 1019 programming subjects were included in the analysis. All the research data were acquired directly from faculties' online accreditation documents, curricula's and study programmes. Data acquisition was done in six-month period of 2015. It should be noted that research considered only state universities with academic level of studies and subjects in the first and second year of study. Table 1 provides a list of universities and faculties from which data were acquired.

Table 1
Overview of the research sample structure

Country	University	Faculty/Department	No. of Study Prog.
1	2	3	4
Austria	University of Vienna	Faculty of Informatics	21
	Vienna University of Technology	Faculty of Electrical Engineering and IT Faculty of Mathematics and Geoinformation Faculty of Computer Science	
	University of Innsbruck	Faculty of Mathematics, Computer Science and Physics	
	Graz University of Technology	Faculty of Electrical and Information Engineering Faculty of Technical Mathematics and Technical Physics Faculty of Computer Science	
Belarus	Belarusian State University	Department of Applied Mathematics and Computer Science	13
	Belarusian National Technical University	Faculty of Information Technology and Robotics	
	University of Informatics and Radioelectronics	Faculty of Information Technology and Management	
	Yanka Kupala State University of Grodno	Faculty of Mathematics and Computer Science	
	Polotsk State University	Faculty of Information Technology	
Belgium	Catholic University of Leuven	Faculty of Sciences	14
	University of Liége	Faculty of Applied Sciences	
	University of Antwerp	Faculty of Sciences	
	Ghent University	Faculty of Engineering and Architecture	
Bosnia and Herzegovina	University of Sarajevo	Faculty of Electrical Engineering Faculty of Natural Sciences and Mathematics	19
	University of Tuzla	Faculty of Electrical Engineering Faculty of Science	
	University of Mostar	Faculty of Science and Education	
Bulgaria	University of Sofia “St. Kliment Ohridski”	Faculty of Mathematics and Informatics	24
	Technical University – Sofia	Faculty of Electrical Engineering Faculty of Applied Mathematics and Informatics Faculty of Computer Systems and Control	
	Ruse University	Faculty of Electrical Engineering and Automation Faculty of Natural Science and Education	
	Plovdiv University	Faculty of Mathematics, Informatics and Information Technology	
	Technical University of Varna	Department of Communication Technics and Technology Department of Computer Science and Technology	
	Burgas Free University	Faculty of Computer Science and Engineering	

1	2	3	4
Croatia	University of Zagreb	Faculty of Science Faculty of Electrical Engineering and Computing	25
	University of Rijeka	Faculty of Engineering Department of Informatics	
	JJ Strossmayer University of Osijek	Faculty of Electrical Engineering Department of Mathematics	
Czech Republic	Czech Technical University in Prague	Faculty of Information Technology	18
	Masaryk University	Faculty of Informatics	
	Brno University of Technology	Faculty of Information Technology	
	University of Ostrava	Faculty of Science	
Denmark	University of Copenhagen	Department of Computer Science Department of Mathematics	19
	Aarhus University	Department of Computer Science	
	Aalborg University	Department of Computer Science	
	Technical University of Denmark	Department of Mathematics and Computer Science	
Estonia	University of Tartu	Faculty of Science and Technology	15
	Tallin University of Technology	Faculty of Information Technology	
	Tallin University	Institute of Informatics	
Finland	University of Helsinki	Faculty of Science	18
	Tampere University of Technology	Faculty of Computing and Electrical Engineering	
	University of Jyvasyla	Faculty of Information Technology	
	University of Turku	Faculty of Mathematics and Natural Sciences	
	University of Tampere	School of Information Sciences	
Former Yugoslav Republic of Macedonia (FYROM)	Ss. Cyril and Methodius University of Skopje	Faculty of Computer Science and Engineering Faculty of Electrical Engineering and Information Technologies Faculty of Mechanical Engineering Faculty of Natural Sciences and Mathematics Faculty of Civil Engineering	31
	Goce Delcev University of tip	Faculty of Computer Science Faculty of Electrical Engineering Faculty of Natural and Technical Sciences	
France	Pierre and Marie Curie University	Faculty of Engineering Faculty of Mathematics	27
	Claud Bernard University Lyon I	Faculty of Science and Technology	
	University of Nice – Sophia Antipolis	Faculty of Sciences	
	University of Caen Lower Normandy	Faculty of Science	
	University of Paris 1 Pantheon – Sorbonne	Department of Mathematics and Computer Science	

1	2	3	4
Germany	Technical University of Berlin	Faculty of Electrical Engineering and Computer Science	31
	Free University of Berlin	Department of Mathematics and Computer Science	
	Humboldt University of Berlin	Faculty of Mathematics and Natural Sciences	
	University of Munich	Department of Mathematics, Computer Science and Statistics	
	University of Heidelberg	Faculty of Natural Sciences, Mathematics and Computer Science	
	University of Hamburg	Faculty of Mathematics, Computer Science and Natural Sciences	
	University of Leipzig	Faculty of Mathematics and Computer Science	
Greece	Aristotle University of Thessaloniki	Faculty of Sciences	21
	University of Athens	Faculty of Informatics and Telecommunications	
	Technical University of Athens	School of Electrical and Computer Engineering	
	University of Crete	School of Sciences and Engineering	
Hungary	Eötvös Loránd University	Faculty of Informatics Faculty of Science	16
	Budapest University of Technology and Economics	Faculty of Electrical Engineering and Informatics Faculty of Science	
	University of Miskolc	Faculty of Mechanical Engineering and IT	
	University of Pécs	Faculty of Engineering and Information Tech. Faculty of Science	
Ireland	Trinity College Dublin	Faculty of Engineering, Mathematics and Science	23
	University College Dublin	School of Computer Science and Informatics	
	University College Cork	Department of Computer Science	
	University of Limerick	Department of Computer Science and Information Systems	
	Dublin City University	Faculty of Engineering and Computing School of Computing	
	National University of Ireland, Galway	College of Engineering and Informatics	
Italy	University of Bologna	School of Engineering and Architecture	22
	Sapienza University of Rome	Faculty of Inform. engineering, Computer Science and Statistics	
	University of Pisa	Department of Computer Science Department of Information Engineering	
	University of Padua	School of Engineering School of Science	
	Polytechnic University of Milan	Department of Electronics, Informatics and Bioengineering	

1	2	3	4
Latvia	University of Latvia	Faculty of Computing	15
	Riga Technical University	Faculty of Computer Science and Information Technology	
	Vidzeme University of Applied Sciences	Faculty of Engineering	
	Daugavpils University	Faculty of Natural Sciences and Mathematics	
Lithuania	Vilnius University	Faculty of Mathematics and Informatics	16
	Kaunas University of Technology	Faculty of Informatics	
	Vytautas Magnus University	Faculty of Informatics	
	Vilnius Gediminas Technical University	Faculty of Electronics Faculty of Fundamental Sciences	
Netherlands	University of Twente	Department of Computer Science	17
	Ultrecht University	Faculty of Science	
	Leiden University	Institute of Advanced Computer Science	
	Eindhoven University of Technology	Department of Mathematics and Computer Science	
	Delft University of Technology	Faculty of Electrical Eng., Mathematics and Computer Sciences	
	University of Amsterdam	Faculty of Science	
Norway	Norwegian University of Science and Technology	Faculty of IT, Mathematics and Electrical Engineering Faculty of Science and Technology	20
	University of Oslo	Faculty of Mathematics and Natural Sciences	
	University of Bergen	Faculty of Mathematics and Natural Sciences	
	University of Tromsø	Faculty of Science and Technology	
Moldova	Moldova State University	Faculty of Mathematics and Informatics	8
	Moldova Technical University	Faculty of Computers, Informatics and Microelectronics	
Montenegro	University of Montenegro	Faculty of Electrical Engineering Faculty of Mechanical Engineering Faculty of Natural Sciences and Mathematics	9
Poland	AGH University of Science and Technology	Department of Computer Science	18
	Jagiellonian University	Department of Mathematics and Computer Science	
	Warsaw University of Technology	Faculty of Electronics and Information Technology Faculty of Mathematics and Information Science	
	Gdansk University of Technology	Faculty of Electronics, Telecommunications and Informatics	

1	2	3	4
Portugal	University of Porto	Faculty of Sciences Faculty of Engineering	15
	University of Coimbra	College of Science and Technology	
	University of Minho	School of Engineering School of Sciences	
	Technical University of Lisbon	Faculty of Sciences Institute of Superior Technics	
Romania	Universitatea Alexandru Ioan Cuza	Faculty of Computer Science Faculty of Mathematics Faculty of Physics	27
	Babes-Bolyai University Cluj-Napoca	Faculty of Mathematics and Informatics	
	Technical University of Cluj-Napoca	Faculty of Automation and Computer Science Faculty of Electronics, Telecommunications and IT	
	West University of Timisoara	Faculty of Mathematics and Informatics	
	Technical University of Iasi	Faculty of Automation and Computers Faculty of Electronics, Telecommunications and IT	
	Politehnica University of Timisoara	Faculty of Computer Science	
Russia	Moscow State University	Faculty of Computational Mathematics and Cybernetics	33
	Novosibirsk State University	Department of Information Technologies	
	St. Petersburg State Polytech. University	Institute of Computing and Control	
	Tomsk State University	Faculty of Informatics	
	Southern Federal University	Faculty of Automation and Computer Engineering Faculty of Mathematics, Computer Science and Physics Faculty of Mathematics and Computer Science	
Serbia	University of Belgrade	Faculty of Electrical Engineering Faculty of Transport and Traffic Engineering Faculty of Organizational Sciences Faculty of Mathematics Faculty of Physics	36
	University of Novi Sad	Faculty of Technical Sciences Faculty of Sciences Faculty of Education	
	University of Niš	Faculty of Electronic Engineering Faculty of Sciences and Mathematics	
	University of Kragujevac	Faculty of Natural Sciences and Mathematics Faculty of Engineering Faculty of Technical Sciences	
Slovakia	Comenius University in Bratislava	Faculty of Mathematics, Physics and Informatics	16
	Slovak University of Technology Bratislava	Faculty of Electrical Engineering and Informatics Faculty of Informatics and Information Technologies	
	University of Žilina	Faculty of Management Science and Informatics	
	Constantine the Philospher University	Faculty of Natural Sciences	

1	2	3	4
Slovenia	University of Ljubljana	Faculty of Electrical Engineering Faculty of Mathematics and Physics Faculty of Computer and Information Science	9
	University of Maribor	Faculty of Electrical Engineering and Computer Science	
Spain	Complutense University of Madrid	Faculty of Information Technology and Computer Science Faculty of Mathematical Sciences	27
	Technical University of Madrid	Faculty of Computer Sciences	
	University of Seville	Faculty of Mathematics Higher Polytechnic School Higher Technical School of Computer Engineering	
	Technical University of Barcelona	School of Mathematics and Statistics Image Processing and Multimedia Technology Centre Manresa School of Engineering Barcelona School of Informatics Vilanova School of Engineering Mataró College of Engineering Barcelona School of Telecommunications Eng. Terrassa School of Engineering	
Sweden	Linkoping University	Institute of Technology	28
	KTH Royal Institute of Technology	School of Computer Science and Communication School of Education and Communication in Engineering Science School of Electrical Engineering School of Information and Communication Tech.	
	Uppsala University	Department of Information Technology	
	University of Gothenburg	Faculty of Science IT Faculty	
Switzerland	Swiss Federal Institute of Technology	Department of Information Technology and Electrical Eng. Department of Computer Science	14
	University of Geneve	Faculty of Science	
	University of Bern	Faculty of Science	
Ukraine	Taras Shevchenko National University	Department of Cybernetics	18
	National Technical University of Ukraine	Faculty of Information Science and Computer Engineering	
	Percarpathian University	Department of Mathematics and Computer Science	
	Lviv Polytechnic National University	Institute of Computer Science and Information Technology	

1	2	3	4
United Kingdom	University of Oxford	Department of Computer Science Department of Engineering Science Mathematical Institute	30
	University of Cambridge	Faculty of Engineering Faculty of Computer Science and Technology	
	University of Edinburgh	College of Science and Engineering School of Informatics	
	University College London	Department of Computer Science	
	University of Glasgow	College of Science and Engineering	
	Imperial College London	Faculty of Engineering	

Table 2 provides an overview of the programming languages used in introductory programming courses within ICT studies in representative European universities for the 2014/2015 school year. Results were obtained based on the analysis of the representation of various programming languages in teaching subjects. Curricula analysis of reviewed 715 study programmes implies that programming courses are primary taught at faculties that belong to the fields of natural sciences, technics and technology. Most curriculums provide the study of introductory programming (also called Introduction to Programming, Programming Languages, Programming, Programming Fundamentals, Principles of Programming, Programming Techniques etc.) in the first semester, after which one or more object-oriented programming languages (similar name cases) are taught. The research did not include 3rd and 4th year of studying, but it anticipates that software-oriented curricula's have developed courses that consider programming in certain specific areas (hardware, web, etc.). Study of programming in European countries is mostly software engineering oriented, with significant participation of object-oriented languages.

Based on performed analysis we come to several conclusions.

Conclusion 1: It was concluded that three programming languages have the leading role: C (30.6%), C++ (21.9%) and Java (20.7%). Weekly average hours for teaching these languages is 2.0 + 1.5 + 1.1 (Lectures + Practice + Laboratory) which is consistent with the presumption that in courses dealing with introductory programming significant time is allocated to study theory due to its characteristics and target objectives.

Conclusion 2: Key finding of the research is that introductory programming language that was most often taught in the 1st semester is C (45.7%), followed by C++ (15%), Java (8.3%), Pascal (7.7%) and Python (5.6%). Although C++ is the language of choice of most (32.8%) 2nd semester curricula's, there still was a significant share of courses that taught C (28.9%) and Java (16.1%). Java is clearly the most common programming language in 3rd and 4th semester, represented in 39% and 35% of the courses, respectively.

Table 2
Summary representation of programming languages at universities in Europe

Programming language	Semester					Weekly Avg. (Lec+Pra+Lab)
	I	II	III	IV	Σ	
C	155	96	39	20	310	2.2 + 1.2 + 1.1
C++	51	110	34	29	224	2.1 + 1.7 + 1.4
Java	28	54	78	52	212	2.2 + 1.7 + 0.7
C#	5	12	12	21	50	1.8 + 2.0 + 1.3
Pascal	26	12	3	–	41	2.3 + 2.1 + 0.8
Python	19	8	1	1	29	1.9 + 1.2 + 1.0
MatLab	4	11	4	–	19	2.0 + 2.2 + 0.2
Visual Basic	11	2	2	1	16	2.0 + 0.6 + 1.5
Haskell	–	2	12	–	14	2.1 + 0.2 + 1.0
Scheme	3	2	2	5	12	2.2 + 0.9 + 1.3
UML	–	2	6	3	11	1.9 + 1.7 + 1.0
Prolog	1	–	5	4	10	1.8 + 1.6 + 1.0
SQL	–	9	–	1	10	2.0 + 0.0 + 2.0
Assembler	4	5	1	–	10	2.7 + 1.7 + 0.4
picoComputer	7	–	–	–	7	3.0 + 2.0 + 0.0
Logo	6	1	–	–	7	2.0 + 2.0 + 3.0
Mathematica	3	–	–	3	6	2.0 + 3.0 + 0.0
XML	3	–	1	2	6	1.9 + 2.3 + 1.3
X	6	–	–	–	6	2.0 + 2.0 + 3.0
Modula-2	5	–	–	–	5	2.0 + 2.0 + 1.0
HTML	–	–	–	4	4	1.6 + 1.8 + 0.3
JavaScript	–	3	–	–	3	2.2 + 1.3 + 0.6
Perl	–	2	–	–	2	1.0 + 0.0 + 2.0
Delphi	1	–	–	1	2	2.0 + 2.0 + 0.0
Fortran	1	–	–	–	1	2.0 + 2.0 + 0.0
Clojure	–	1	–	–	1	2.0 + 2.0 + 0.0
Ada	–	–	–	1	1	2.0 + 2.0 + 0.0

Lec – Lectures, Pra – Practice, Lab – Laboratory

Conclusion 3: Analysis of paradigm representation among programming languages that were taught resulted in following structure: 53.7% object-oriented, 36.8% imperative and 1.8% functional. Results are shown on Fig. 1.

Conclusion 4: When the correlation between European regions and participation of programming languages that were taught was analyzed, we came to conclusion that universities from Central and Eastern-European countries mainly based their study programmes on teaching C and C++ programming languages, while programmes of the Scandinavian universities were mainly based on Java. This grouping is probably the consequence of cultural, economic, political, historical, and, indirectly, educational link between these countries. University programmes of Western and South-European countries were heterogeneous but strongly object-oriented, except in France and Bel-

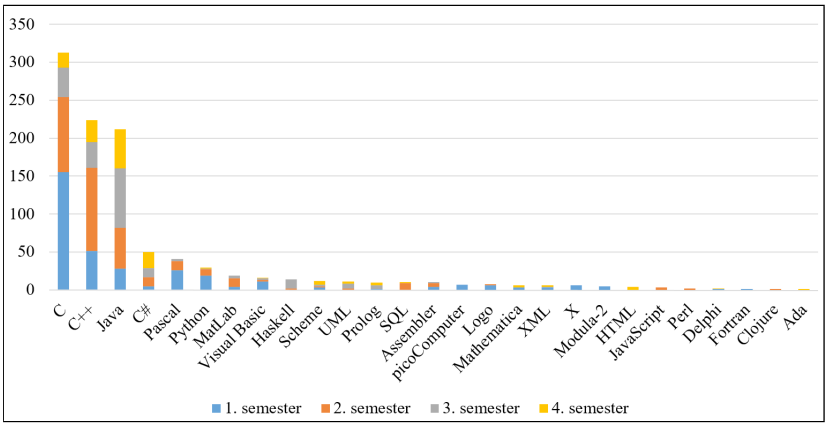

Fig. 1. Programming language structure at European universities.

gium which were mainly based on procedural style. It can be assumed that constantly increa-sing interconnection within European continent shall lead to further homogenization of this distribution. One of the important factors for this projection is an increase in student's mobility through various projects, Erasmus Programme, Erasmus+, etc.

Integrated visual representation of the results achieved through analysis is given in Fig. 2.

5. Final Remarks

Despite a number of studies (Chalk and Fraser, 2006; Gupta, 2004; Ivanović and Budimac, 2013; Jablonowski, 2007; Mason *et al.*, 2012; Mclver and Conway, 1996) in the area of identifying, analyzing and determining the usage of "proper" programming language for the implementation of introduction programming courses, there is a little systematic evidence to support any practical and particular solution. For this reason, there was no attempt to create the canonical form of answers to the question of how to teach the subject.

Our research was designed and directed with a goal to identify and show a variety of approaches that were caused by the need to support the teachers and their decisions in their teaching. It is obvious that there were a number of universities that had outdated curricula with outdated programming languages, so further structural change can be expected.

It is important to notice that some new programming languages that are especially useful for educational purposes (rather than for industrial usage) are emerging and are becoming increasingly popular, such as Python. By using IDE's and various visual and narrative tools teaching process is becoming more relaxed. There is a huge amount of evidence that students have generally positive view on the introduction of supportive programming tools. The introduction of three or more programming languages in the

Country	Avg. Programming Language participation		
Austria	C++ (38%)	C (38%)	Java (11%)
Belarus	C (51%)	C++ (20%)	Java (15%)
Belgium	C (50%)	C++ (31%)	Java (12%)
Bosnia and Herzegovina	C++ (51%)	C (32%)	Java (5%)
Bulgaria	C++ (39%)	Java (18%)	C (18%)
Croatia	C (35%)	Java (22%)	C# (20%)
Czech Republic	C (44%)	C++ (31%)	Java (25%)
Denmark	C (38%)	Java (21%)	C++ (20%)
Estonia	Java (43%)	C (20%)	Python (18%)
Finland	Java (38%)	C (25%)	C++ (12%)
France	C (41%)	C++ (23%)	Java (12%)
Germany	Java (33%)	C (30%)	C++ (25%)
Greece	C (36%)	C++ (28%)	C# (12%)
Hungary	C (33%)	C++ (25%)	C# (21%)
Ireland	Java (51%)	C (18%)	C# (15%)
Italy	Java (40%)	C (33%)	C++ (20%)
Latvia	C++ (35%)	C (32%)	Java (25%)
Lithuania	C (38%)	C++ (33%)	C# (18%)
Netherlands	Java (32%)	C++ (29%)	C# (20%)
Norway	Java (54%)	C (23%)	C# (9%)
Macedonia (FYROM)	C++ (24%)	C (22%)	Java (20%)
Moldova	C (40%)	C++ (23%)	Java (15%)
Montenegro	C++ (33%)	C (33%)	Java (25%)
Poland	C++ (29%)	C (26%)	Java (20%)
Portugal	Java (39%)	Python (27%)	C (18%)
Romania	C (35%)	C++ (26%)	Java (26%)
Russia	C (38%)	Java (26%)	C++ (23%)
Serbia	C (31%)	Java (15%)	C++ (14%)
Slovakia	C++ (30%)	C (30%)	Java (22%)
Slovenia	C (40%)	C++ (25%)	Java (15%)
Spain	C++ (30%)	Java (28%)	C (27%)
Sweden	Java (46%)	C++ (20%)	C (16%)
Switzerland	C++ (48%)	Java (22%)	C (20%)
Ukraine	C (42%)	C++ (35%)	Java (30%)
United Kingdom	C (35%)	Java (26%)	C++ (20%)

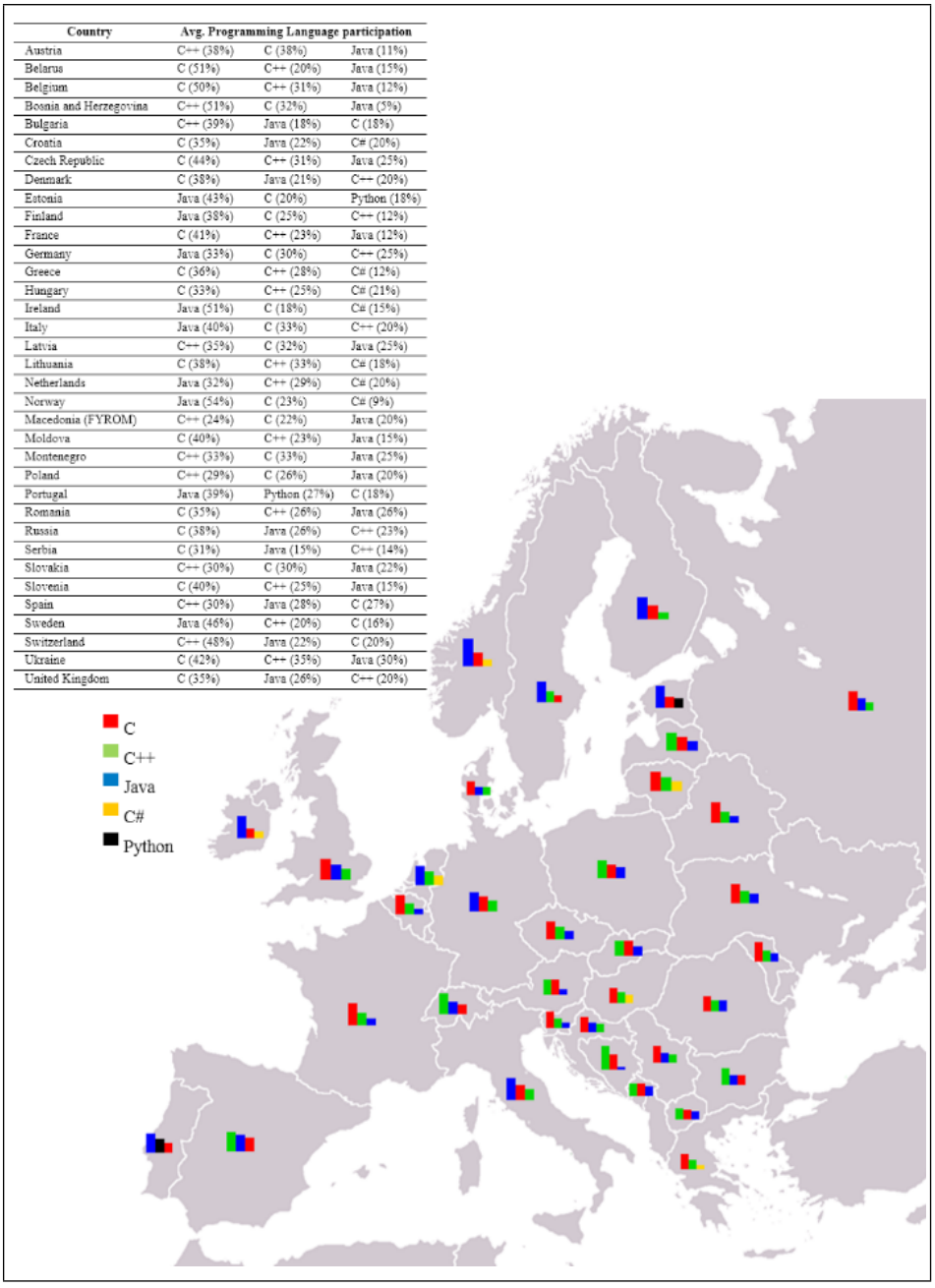

Fig. 2. Avg. participation of the most popular programming languages by European countries.

introductory programming did not prove effective due to cognitive overload and immediately ineffective learning (Pendergast, 2006).

Choice of programming language is influenced by subjective (personal preference) and objective (study programme objectives, market conditions and requirements, etc.) factors (Goosen *et al.*, 2007; Huet *et al.*, 2004). Homogeneity of the programming language selection within university/faculty is a consequence of the fact that ultimately the teacher or several colleagues rarely decide on it independently.

Analysis of study programmes has proven variety in the ways of teaching and types of used programming paradigm and languages. The results largely coincided with a number of research studies in the field of programming languages.

It should be stated that the lack of curricula standardization in this area presents a limitation as the industry cannot rely on teachers' subjective attitude towards programming.

References

Azad, A., Kohun, F. (2009). *Considerations for Selecting a Programming Language to Teach Perspective Teachers*. Alice Symposium, Duke University, Durham.

Bordini, R., Braubach, L., Dastani, M., Segrouchni, A., Gomez-Sanz, J., Laite, J., O'Harre, G., Pokahr, A., Ricci, A. (2006). A survey of programming languages and platforms for multi-agent systems. *Informatica,* 30(1).

Böszörmenyi, L. (2005). *Teaching: People to People–About People A Plea for the Historic and Human View. From Computer Literacy to Informatics Fundamentals*. Springer, Berlin Heidelberg.

Bradley, J., Case, M., Anita, C. (2002). *Programming with Java*. Boston, McGraw-Hill.

Butz, C. J., Hua, S., Maguire, R.B. (2004). A web-based intelligent tutoring system for computer programming. *IEEE/WIC/ACM International Conference on Web Intelligence (WI'04)*.
DOI: 10.1109/wi.2004.10104

Chalk, B., Fraser, K. (2006, February). A Survey on the teaching of introductory programming in Higher Education. In: *Proceedings of the 10th Java & the Internet in the Computing Curriculum Conference (JICC10).* Thomson Publishers, London, United Kingdom, 1–6.

Chen, C. (2004). Comparison of object-oriented and procedural computer languages: case study of C++ programming languages. *Issues in Information Systems,* 4(1).

Comenius, J.A. (1957). *Große Didaktik*. Volk und Wissen, Berlin.

Deitel, P., Deitel, H. (2011). *C++ How to program. 8th Edition*. Prentice Hall.

Felder, R. M., Silverman, L. K. (1988). Learning and teaching styles in engineering education. *Engineering education*, 78(7), 674–681.

Freund, S., Bruce, K., Hertz, M., Leavens, G., Snyder, L. (2010). *Why Undergraduate Should Learn the Principles of Programming Languages.* ACM SIGPLAN Education Board.

Gomes, A., Mendes, J. (2007). Learning to program – difficulties and solutions. *International Conference on Engineering Education – ICEE*, Portugal.

Goosen, L., Mentz, E., Nieuwoudt, H. (2007). Chosing the "best" programming language?! In: *Computer Science and IT Education Conference CSITEd*, Mauritius.

Gupta, D. (2004). What is a good first programming language? *Crossroads,* 10(4), 7–7.
DOI: 10.1145/1027313.1027320

Huet, I., Pacheco, O.R., Tavares, J., Weir, G. (2004). New challenges in teaching introductory programming courses: a case study. In: *34th Annual Frontiers in Education, 2004. FIE 2004*.
DOI: 10.1109/fie.2004.1408514

Ivanović, M., Budimac, Z. (2013). First programming language – never-ending story. In: *AIP Conference Proceedings 1558*. Rhodes, Greece, 353-356.
DOI: 10.1063/1.4825496

Jabłonowski, J. (2007). A case study in introductory programming. In: *Proceedings of the 2007 International Conference on Computer Systems and Technologies – CompSysTech'07*.
DOI: 10.1145/1330598.1330685
Jenkins, T. (2002, August). On the difficulty of learning to program. In: *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences.* 4, 53–58.
Kaplan, R.M. (2010). Choosing a first programming language. In: *Proceedings of the 2010 ACM Conference on Information Technology Education – SIGITE'10*.
DOI: 10.1145/1867651.1867697
Kolb, A., Kolb, D. (2005). *The Kolb Learning Style Inventory*. Experience Based Learning Systems, Inc.
Kölling, M. (1999). The problem of teaching object-oriented programming. *Journal of Object Oriented Programming*, 11(8), 8–15.
Mason, R., Cooper, G., de Raadt, M. (2012, January). Trends in introductory programming courses in Australian universities: languages, environments and pedagogy. In: *Proceedings of the Fourteenth Australasian Computing Education Conference, Vol. 123*. Australian Computer Society, Inc., 33–42.
McIver, L., Conway, D. (1996). Seven deadly sins of introductory programming language design. In: *Proceedings 1996 International Conference Software Engineering: Education and Practice*.
DOI: 10.1109/seep.1996.534015
McKenzie, B. (2009). Introductory programming with ALICE as a gateway to the computing profession. In: *Proceedings of the 23rd Annual Conference for Information Systems Educators (ISECON 2006)*. 2–5.
Meyerovich, L., Rabkin, A. (2013). *Empirical Analysis of Programming Language Adoption*. UC Berkeley.
Ng, E., Bereiter, C. (1991). Three levels of goal orientation in learning. *Journal of the Learning Sciences*, 1(3), 243–271.
DOI: 10.1207/s15327809jls0103&4_1
Parker, K.R., Ottaway, T.A., Chao, J.T. (2006). Criteria for the selection of a programming language for introductory courses. *IJKL*, 2(1/2), 119.
DOI: 10.1504/ijkl.2006.009683
Pendergast, M. (2006). Teaching introductory programming to IS students: Java problems and pitfalls. *Journal of Information Technology Education: Research*, 5(1), 491–515.
Polya, G. (1957). *How to Solve it*. Princeton University Press, New York.
Rahmat, M., Shahrani, S., Latih, R., Yatim, N.F.M., Zainal, N.F.A., Ab Rahman, R. (2012). Major problems in basic programming that influence student performance. *Procedia-Social and Behavioral Sciences*, 59, 287–296.
DOI: 10.1016/j.sbspro.2012.09.277
Thomas, L., Ratcliffe, M., Woodbury, J., Jarman, E. (2002). Learning styles and performance in the introductory programming sequence. *ACM SIGCSE Bulletin,* 34(1), 33.
DOI: 10.1145/563517.563352
Ulrich, J., Karvonen, M. (2011). Faculty instructional attitudes, interest, and intention: predictors of Web 2.0 use in online courses. *The Internet and Higher Education,* 14(4), 207–216.
DOI: 10.1016/j.iheduc.2011.07.001
Vesin, B., Ivanović, M., Klašnja-Milićević, A., Budimac, Z. (2012). Protus 2.0: ontology-based semantic recommendation in programming tutoring system. *Expert Systems with Applications,* 39(15), 12229–12246.
DOI: 10.1016/j.eswa.2012.04.052
Vesin, B., Ivanovic, M., Klasnja-Milicevic, A., Budimac, Z. (2013). Ontology-based architecture with recommendation strategy in java tutoring system. *Computer Science and Information Systems Journal,* 10(1), 237–261.
DOI: 10.2298/csis111231001v
Wang, P. (2003). *Java with Object-Oriented Programming*. Pacific Grove, Thomson Brooks/Cole.
Watt, A. (2004). *Programming Language Design Concepts*. John Wiley.
Weisert, C. (1997*). Learning to Program: It Starts with Procedural*. Information Disciplines, Chicago.

V. Aleksić received the MSc degree in Teaching technics and informatics from Faculty of Technical Sciences in Čačak, University of Kragujevac, Serbia. He is a Teaching Assistant and LMS Moodle Administrator at the Faculty of Technical Sciences in Čačak. Veljko was a member of several expert teams at the Institute of Educational Quality, Ministry of Education, Serbia. His research interests include educational technology, teacher training, game-based learning, and e-learning. He is the author and co-author of 26 journal and conference papers, and four handbooks and textbooks.

M. Ivanović holds the position of Full Professor at Faculty of Sciences, University of Novi Sad, Serbia since 2002. She is a member of University Council for Informatics. She is author or co-author of 13 textbooks and of more than 320 research papers on multiagent systems, e-learning and web-based learning, software engineering education, intelligent techniques (CBR, data and web mining), most of which are published in international journals and conferences. She is/was a member of Program Committees of more than 160 international conferences and is the Editor-in-Chief of Computer Science and Information Systems Journal. She was member and principal investigator of numerous international projects.