Students' perspective on the first programming language: C-like or Pascal-like languages?

Stelios Xinogalos[1], Tomáš Pitner[2], Mirjana Ivanović[3] & Miloš Savić[3]

[1] Department of Applied Informatics, School of Information Sciences, University of Macedonia, 156 Egnatia Street, GR-54636 Thessaloniki, Greece
stelios@uom.edu.gr

[2] Department of Computer Systems and Communications, Faculty of Informatics, Masaryk University, Šumavská 416/15, Ponava, Brno, Czech Republic
tomp@fi.muni.cz

[3] Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad, Trg Dositeja Obradovića 4, Novi Sad 21000, Serbia
{mira, svc}@dmi.uns.ac.rs

Abstract The choice of the first programming language (FPL) has been a controversial issue for several decades. Nearly everyone agrees that the FPL is important and affects students' subsequent education on programming. The study presented in this article investigates the suitability of various C-like and Pascal-like programming languages as a FPL. Students from three Universities express their perceptions on the FPL through a specially designed questionnaire. The students had been introduced to programming using different FPLs and having experience on more than one language and formulated two distinct groups: a group introduced to programming with a C-like and another with a Pascal-like language. The statistical analysis of the data collected showed that the choice of the FPL does not have a deep impact on difficulties that novices may experience when learning the basics of programming. However, this result was recorded in the context of three distinct programming courses that were positively evaluated by students in relation to their content, organization and quality. Consequently, this study indicates that the overall quality of the course and the instructors have a great impact on a smooth introduction to programming no matter what the FPL is.

Keywords: First programming language, student difficulties in programming, course evaluation, course organization, course quality

1 Introduction

The course with the title *Introduction to Programming* or similar exists almost in all bachelor studies of computer science (CS) and information and communications technology (ICT) (Moritz 2005). A programming language provides a notion in which to express algorithms, techniques, and data structures and focuses on teaching programming (Goosen 2007). Computer science is a theoretical, as well as a practical discipline and usually deals with science of computation, art, and craft. So, it is extremely important to learn an appropriate programming language (PL) in the first course in computer science.

Different factors and concerns (Ali and Smith 2014) influence the choice of the first programming language (FPL) for CS and ICT studies and it is extremely important to differentiate key features of a language in the educational context.

These factors are driven by didactical and pedagogical issues but also by business and technology development, in general, but regional as well. Nowadays, for ICT teachers, it may seem that the choice of the FPL is a long-resolved matter. Even non-ICT teachers believe that it is obvious which language to use for teaching programming at bachelor studies. Therefore still there are a lot of papers that seriously consider this problem. Still the majority of teachers of programming topics are in dilemmas which programming paradigm and language to choose as first

(Aleksic 2016), as this decision has a strong impact on later courses and finally for the development of a programmer.

During the history of developing numerous programming languages some important programming paradigms have been recognized: imperative, object-oriented, functional, and logic. At the majority of universities in ICT curricula the first two paradigms are predominant. Some teachers like to teach the most popular PLs but there are numerous possible measures of language popularity like: Most widely used? Most jobs? Most lines of code? Another problem connected to popularity of programming language appears: Is popularity of a language a guarantee of its suitability for teaching programming concepts? Based on the above mentioned it is clear that there is no best language for all situations.

In the last two decades with the appearance and intensification of use of some particular languages we are again witnesses of a lot of debates focusing on the choice of the FPL. Should we teach C, Java, C#, Python, Perl, Scala, JavaScript, or something else? Some authors plead that if the choice of a language is a key point then teachers are on the road to failure already. More and more institutions recently adopted Java in their introductory programming courses. Nevertheless there is no universal satisfaction with this decision. As the choice of the FPL for ICT education is not an easy task teachers have to be aware of this and continuously re-evaluate language choice.

As a part of our Universities' policies, in the last several years the students are regularly posed questionnaires in order to provide feedback and give suggestions for improvements at each particular course. The students are asked to judge different aspects of the course like course content, quality of methodological approach, different aspects of teachers and assistants and available educational material.

Motivated by this good policy and useful feedback from students, the authors of the paper decided to go a step ahead and try to obtain constructive students' opinion about a wide range of aspects and features of FPL (C-like and Pascal-like) they were taught in their institutions. Students were asked to judge elements like: learning language syntax and semantics, installation and usage of programming environments, selection of language constructs and implementation of algorithms in the language, correcting syntax and semantic errors.

The rest of the paper is organized as follows. In section 2 a brief state-of-the -art regarding the choice of FPL is presented. In section 3 the questionnaire used is presented and the data collected are analyzed giving emphasis on discussing main findings. Finally, the most important conclusions are summarized.

2 Choice of first programming language: state-of-the-art

The choice of the first programming language has been a controversial issue for several decades. Nearly everyone agrees that the FPL is important and affects students' subsequent education on programming. The importance of the FPL has resulted in extended research regarding the effectiveness of various programming languages and paradigms when used for an introduction to programming (Farooq et al. 2014).

Several studies focus on presenting the results of making a transition from one programming language to another in terms of a more suitable FPL. Next, we report on relevant studies in ascending chronological order with the aim of studying how the landscape on FPL changed as years passed.

Radesnky et al. (1988) report on their positive experience of using Ada as a FPL. The authors consider Ada to be more appropriate than Pascal and Modula-2 as a FPL and state that students are able to learn Ada easily and quickly. Moreover, students appreciate the advanced features of Ada and write more sophisticated programs. However, these are all based on subjective evidence.

Skublics and White (1991) report on their experience of moving from Pascal to Smalltalk. The main reasons for this decision were the simpler syntax of Smalltalk and the fact that it supports more the notions of data and procedural abstraction, encapsulation and modular programming, as well as the fact that it supports the teaching of concepts prior to their implementation. Students' grades in the revised course were compared with those of previous years when the course was based on Pascal and no significant difference was recorded. A questionnaire for recording students' opinion for the course was also used; however the small number of respondents (14) limits its importance. Students with prior knowledge of programming stated that had difficulty in their transition to the object-oriented paradigm. Another problem reported was the extended size of the class library.

Hitz and Hudec (1995) report on their unexpected positive experience on moving from an educational language to a conventional language used in industry. Specifically, the authors based exclusively on didactical considerations used - starting back in 1974- Fortran, PL/1, Pascal and Modula-2. Although they considered both Pascal and Modula-2 successful for the purposes of the course, they decided in 1992 to move to C++ in order to satisfy students' willingness to learn a language they would afterwards use in their studies and the market. Object-

orientation and inheritance were not taught. No significant difference in students' performance was recorded, while students' feedback was positive. However, no formal instrument was used for investigating students' feedback.
As shown from the last two studies, at some point the research on FPL started to focus in the programming paradigm supported by the FPL. Brilliant and Wiseman (1996) used a survey regarding "the first programming paradigm and language dilemma" that was sent to Computer Science Departments at 145 Universities and received 45 replies. This survey investigated: the procedural paradigm and the languages Pascal, Modula-2, Ada and C; the object-oriented parading and C++; and the functional paradigm and Scheme. Although, Ada seemed a good choice at that time, there did not seem to be a paradigm that offered significant advantages for the first programming course, while students seemed able to move to another paradigm with reasonable ease. However, other studies conclude that students face more difficulties during their transition from procedural programming to object-oriented programming and not vice versa (Decker and Hirshfield 1994; Hadjerrouit 1998a; Tempte 1991; Wick 1995).

As expected, this controversy between paradigms and languages was intensified with the advent of Java. Several papers were published reporting the advantages and disadvantages of Java as a FPL, as well as experience on using it in introductory programming courses. Hadjerrouit (1998b) reports on the experience of moving from Simula and C++ to Java as a FPL in order to take advantage of object-orientation, concurrent programming and its combination with the World Wide Web. It turned out that Java is not as easy as expected as a FPL, but it is also clear that it is necessary to incorporate it in the CSE curriculum. Several years later, Jabłonowski (2007) also reported on the great advantages of Java and at the same time its inadequacy as a FPL in comparison to Pascal or Smalltalk for instance. In a contemporary study, Farag et al. (2013) compared the effectiveness of an online introductory course based on C++ (control group) and Java (experimental group) respectively. No statistically significant differences were recorded for the two groups neither in terms of students' achievements nor in terms of satisfaction. Duke et al. (2000) stress out that object-orientation and Java are important for attracting students at universities. However, adopting Java as a FPL is just the first step that offers great opportunities for changing the way of teaching: adopting a problem-based approach; allowing students to work independently on their own pace; applying an inverse curriculum adopting object-orientation from the very beginning.

On the other hand, Ivanović et al. (2015) concluded that the choice of the first programming language does not matter if students' performance at the course is used as the only criterion of suitability of the language. They presented their recent experience in changing the first programming language from Modula-2 to Java. The authors wanted to discover if there are any significant differences in students' success between the most recent generation taught in Java, and several previous generations taught in Modula-2. Since there were not many other significant factors distinguishing the generations they concluded that statistical analysis of collected scores and grades could provide insight into the effects of the change of language. Comparisons involving two non-parametric statistical tests showed that there are no statistically significant differences between the considered generations with respect to success in passing the exam.

For several years now, Python has come to surface as a FPL (Dierbach 2014). Sanders and Langford (2008) studied students' opinions of Python as a FPL using a survey. Twenty-eight students with no prior programming experience and twenty-seven students with some prior experience completed the survey. Both groups consider Python suitable as a FPL, however some students with prior programming knowledge are reluctant to move to new languages. Lepping et al. (2009) analyze the reasons for deciding to move from Java to Python, after a small-scale experiment on using Python as a FPL with 25% of the students. Python is a multi-paradigm language with a simple and clear syntax, while its popularity in the private sector is growing. Yadin (2011) carried out an action research with the aim of reducing the high dropout rates of students after introductory programming courses. Using Python as a FPL was an important factor, since its simple syntax helped students concentrate on problem solving and algorithms. Other contributing factors were using a visualization environment (microworld) and individual assignments. These factors in combination reduced the failing students by 77%.

3 Students' opinion about first programming language

Our institutions have adopted the "imperative first" approach and C-like and Pascal-like languages for teaching the first programming language.

The University of Macedonia, Department of Technology Management (further referred to as UOM-TMD) where the questionnaire analysed in the next paragraphs was distributed, used since its establishment in 2004 the imperative programming paradigm and the language of C for the introductory programming course taking into account various didactical and pedagogical issues recorded in the literature (Xinogalos 2016). Starting from the academic year 2014 UOM-TMD was merged as a new direction of studies with the Department of Applied

Informatics. The Department of Applied Informatics uses C as the FPL for several years now, while its predecessor was Pascal.

Masaryk University, Faculty of Informatics (further referred to as MUNI-FI) is a research and higher educational institution established in 1994 as the first educational institution in CZ primarily focused at computer science and IT. Since its foundation, it used Pascal as the first imperative programming language together with Haskell as the first functional paradigm language for teaching purposes. Since 2011, the imperative paradigm introductory course has been changed from Pascal to C with an alternative Python-based one which recently (2016) entirely replaced the C-based course as the introductory one.

The University of Novi Sad, Faculty of Sciences (further referred to as UNS-PMF) till last school year used Modula-2 as a typical language designed for teaching purposes. However, in the last years we were under constant pressure from different sources (like local companies, competition with other faculties with similar study programs, students, their parents, or other colleagues) to change the language used in the introductory course. So in the last year we decided to teach basic programming concepts relying on Java but keep the imperative approach first.

In order to obtain students' opinion on different aspects of teaching, characteristics and influences of first programming language we conducted a specially designed questionnaire. Results of this questionnaire are presented and discussed in more details later in this section.

3.1 First programming courses in involved institutions: current situation

At UOM-TMD the Computer Programming course covered fundamental imperative programming concepts (primitive data types, control structures, functions, arrays, strings, pointers and text files). Since the inclusion of UOM-TMD as a direction at the Department of Applied Informatics (DAI) the corresponding Procedural Programming course that is common for all the students (irrespectively of the direction of studies) covers in addition structures and arrays of structures. The course design both at UOM-TMD as an autonomous department and as a direction of studies at DAI consists of 2 hours of lectures and 2 hours of labs per week. The assessment includes weekly assignments (20 points at UOM-TMD and 15 at DAI), mid-term exams (20 points at UOM-TMD and 30 at DAI) and final written exams (60 points at UOM-TMD and 55 at DAI). Students have to collect at least 50 out of 100 points in order to pass the course.

At MUNI-FI the introductory programming course (for computer science studies – there are no other studies) through either C or Python have in principle the same didactical goals but slightly different techniques to attain them.

The C-based introductory course follows more traditional approach derived from the older Pascal-based introduction to programming. The main aim of the course Introduction to Programming using C is to introduce students to basic principles of computer problem-solving. At the end of this course, students should be able to: design an algorithm to solve a given problem; code an algorithm in the C programming language; debug a created program. The topics include basics of I/O functions, expressions, variables, basic statements and control structures, decomposition into functions, data types, and basic algorithms such as numerical and text processing algorithms together with some theory of algorithm analysis: correctness, efficiency. At the end, the pointer type are dynamic data structures are mentioned.

The Introduction to Programming through Python obtains an introduction to programming skills and development of algorithmic style of thinking. At the end of the course students should be able to understand and apply basic constructs of programming languages, such as conditions, loops, functions, basic data types, and will master several basic algorithms and techniques (sorting, searching, recursion, some fundamental use of the turtle graphics).

In general, the C-based course follows rather a "bottom-up" approach building the solution from basic elements upwards while the Python one is more "top-down", progressing from motivation and problem to the coded solution.

At UNS-PMF the introductory programming course for computer science studies covers the basic concepts of the imperative programming style (primitive data types, statements and control structures, procedures, abstract data types and recursion). Our opinion, based on experience spanning multiple decades, is that good educational programming languages (like those designed by Niklaus Wirth), are the most suitable to explain the aforementioned concepts to first year students. Modula-2 has been taught up to last year at our faculty as it satisfied most of the desirable educational premises: most of students have pre-knowledge of Pascal so Modula-2 is convenient for them; It offers clean and well-structured features; It is good for teaching other classical techniques and algorithms. Two years ago we finally decided to teach basic programming concepts relying on Java. Our intention is to keep imperative approach first and objects are introduced at the end of the course.

Nevertheless, after changing the first programming language we decided to keep the same teaching design, the grading and assessment methodology:

- The course consisted of 2+2+1 hours per week of lectures, theoretical exercises and laboratory exercises.
- A student has to achieve at least 30 (out of maximal 60) points at 4 practical (lab) tests in order to fulfill the prerequisites to approach the final oral examination. Also they attend two theoretical tests and achieve additional points, maximally 20. The total grade accumulates the scores at practical assignments, theoretical tests and the oral examination (maximally100 points).

Concluding, two important remarks regarding the most prominent factors that have been taken into account for selecting the FPL and the main learning goals of the underlying courses at the three institutions have to be made:

- Educators and researchers agree that different factors and concerns (Ali and Smith 2014) influence the choice of the FPL, as well as that the selection of the FPL has to be reconsidered from time to time based on both didactical/pedagogical issues and technological trends. Factors that have influenced the choice of the FPL, more or less, at each one of the three institutions include: the popularity of the PL in industry; feedback from industry partners; programming languages used in competitive study programs; students' and their parents' concerns with respect to employability; the needs of the research labs at the Faculty; availability of good quality development tools and didactical material for the PL; and community support. However, the most important factor for selecting or changing the current FPL are its educational aspects or at least the ability to support students in learning the FPL with specially designed educational programming environments and tools, as well as with applying teaching approaches that have proven to be effective for teaching and learning programming. In any case, the selection of the FPL is usually based on the proposals from the instructors of the corresponding courses and has to be carefully justified in order to be approved by the academic council that is responsible in each institution (such as Faculty assembly, Chair of the Faculty and so on).
- The main learning goal of the FPL course is to support students in: developing an algorithmic way of thinking and reasoning; acquiring problem solving skills; learning and applying the main programming concepts that exist in nearly every programming language; learning to design simple algorithms; learning to use contemporary programming environments for implementing and debugging programs; acquiring good style programming habits. Emphasis is not given on the details of the specific FPL, but on concepts and skills considered necessary for learning other programming languages, either in the context of courses or studying alone. Moreover, the aforementioned skills are the first skills that have to be developed in terms of employability and research in the field.

3.2 Questionnaires and results

In order to investigate students' opinions about introductory programming courses and introductory programming languages used at our institutions we designed a questionnaire that consists of 27 questions separated in two parts (see Table 1). The first part of the questionnaire contains questions related to the content, difficulty, quality, organization, and contribution of the introductory programming course. The second part of the questionnaire asks students to evaluate the difficulty of learning programming language used in the introductory programming course, installing and using related programming environments, and solving concrete programming problems in that language. This part of the questionnaire also contains questions related to the students' abilities to theoretically understand programming structures and conceptualize algorithms for solving programming problems. To each question in the second part of the questionnaire, as well as questions C1-C6 and C9-C11 from the first part, respondents answer by choosing exactly one of the responses arranged in the following five-point Likert scale: not at all (1) / slightly (2) / averagely (3) / much (4) / very much (5). Other questions, except C7 and C8 that are "yes-no" questions, also have responses arranged in five-point Likert scales that are explicitly given and explained as the part of the question.

Table 1 Questionnaire used to collect students' opinions about the introductory programming course (Part I) and the introductory programming language (Part II).

	Part I
C1	Are the topics appropriate for an introductory course?
C2	How would you describe the difficulty of the whole course?
C3	Was the quality of the course high?

C4 Is the relationship between theoretical and practical aspects of the course adequate?

C5 Was the organization of the course flawless?

C6 How important was the contribution of the course?

C7 Would you change something in the course?

C8 Would you change something in the style of teaching?

C9 Was the lecturer a good teacher?

C10 Was the assistant good?

C11 Do you think that the relationship between theory and practical work in the course is adequate?

C12 Do you attend the lectures/labs regularly?
(1=not at all, 2=rarely, 3=quite often, > 4 =very often, 5=always)

C13 Based on your experience from other courses, this course is considered to be:
(1=too easy, 2=easy, 3=of average difficulty, 4=difficult, 5=very difficult)

C14 Besides the lectures and labs, how many hours do you devote to this course each week?
(1=<2 hours, 2=2-4 hours, 3=4-6 hours, 4=6-8 hours, 5=> 8 hours)

C15 Did you learn from the other students?
(1=not at all, 2=slightly, 3=averagely, 4=much, 5=very much)

Part II

Which programming language was used in the introductory programming course?

Which programming environment was used in the introductory programming course?

How difficult was each one of the following actions in learning programming (in the context of the introductory programming course):

L1 Theoretical understanding of programming structures

L2 Understanding the definition of a problem

L3 Developing an algorithm for solving a problem (in paper and pencil, or conceptually)

L4 Installing a programming environment

L5 Using programming environment(s)

L6 Learning the programming language syntax

L7 Learning the semantics of programming structures/concepts

L8 Selecting appropriate language structures (if, if/else, switch, for, while,...) for solving a problem

L9 Transferring the algorithm to the programming language

L10 Dividing functionalities in functions/procedures

L11 Understanding compilation error messages and correcting the corresponding errors

L12 Finding bugs from my own program

In each one of our institutions the questionnaire survey was conducted online using the institutional learning management systems. A total of 288 students responded to the survey: 90 students from UOM-TMD, 87 students from MUNI-FI and 111 students from UNS-PMF. According to the first programming language used in the introductory programming course all respondents are classified into one of the following two categories: (1) students that learned basics of programming in a C-like programming language and (2) students that learned basics of programming in a Pascal-like programming language. All respondents from UOM-TMD were introduced to programming in C or Java and belong to the first category. To the contrary, all respondents from UNS-PMF are in the second category because all of them had the introductory programming course based on Modula-2. For MUNI-FI respondents there is a mixed situation: a vast majority of MUNI-FI respondents had the introductory programming course based on a C-like language (C or Java), but also there is a small number of respondents (five of them) that belong to the second category (students that had the introductory programming course in Pascal). In total, 172 respondents belong to the first category, while 116 of them are in the second category.

3.2.1 Students' opinions about the introductory programming course

For each Likert-type question from the first part of the questionnaire basic descriptive statistics of students' responses per institutional group were computed: the median (the central tendency of answers), and inter-quartile range (IQR, a measure of the variability of answers expressed as the difference between the third and the first quartile). The results are shown in Table 2.

Appropriateness of content and course difficulty

Firstly, it can be seen that students who responded to the survey regularly attended courses (question C12, median = 4 in all groups, IQR $\leq$ 2). Secondly, it can be observed that for questions C1, C2, C4 and C13 there are no differences between central tendencies of students' responses from different institutional groups. Moreover, the IQR values for those questionnaire items are low (IQR $\leq$ 1), so central tendencies strongly reflect opinions of a vast majority of examined students. Therefore, it can be concluded that in all institutional groups students think that:

1. The *topics covered by the introductory programming course are highly appropriate* (question C1, Median = 4, IQR < 2).
2. The introductory programming *course is of medium difficulty* compared to other courses (questions C2 and C13, Median = 3, IQR < 2).
3. The *balance between the theoretical and practical aspects of the course is highly adequate* (question C4, Median = 4, IQR < 2). The question C11 is also related to the balance between the theoretical and practical aspects of the course but in terms of working hours, and we can observe that students from UOM-TMD and UNS-PMF think that the existing ratio is highly adequate (Median = 4), while students from MUNI-FI consider it adequate (Median = 3).

Table 2 The descriptive statistics of students' responses to the Likert-type questions from the first part of questionnaire.

Question	UOM-TMD		MUNI-FI		UNS-PMF	
	Median	IQR	Median	IQR	Median	IQR
C1	4	0	4	1	4	1
C2	3	1	3	1	3	1
C3	4	0	3	1	4	1
C4	4	1	4	1	4	1
C5	4	0	4	2	3	1
C6	4	1	4	1	4	2
C9	4	1	4	2	4	2
C10	4	0	4	2	4	1
C11	4	1	3	1	4	2
C12	4	1	4	2	4	2
C13	3	1	3	1	3	1
C14	2	1	1	2	2	2
C15	3	1	1	2	3	2

Quality, organization and contribution of the course

Students also positively evaluated the *quality* of our introductory programming courses and *teaching staff* (questions C3, C9 and C10), as well as the *organization* (question C5) and *contribution* of the courses (question C6). Moreover, a majority of students in each institutional group (73% in UOM-TMD, 71% in MUNI-FI, and 74% in UNS-PMF) would not change anything in the content and realization of the introductory programming course (questions C7 and C8).

Studying hours and habits

An interesting difference between institutional groups can be observed with respect to *self-studying habits of students* (questions C14 and C15):

1. The students from UOM-TMD and UNS-PMF devoted more time (2-4 hours per week on average) for extra learning activities compared to the students from MUNI-FI (less than 2 hour per week on average), and
2. The students from UOM-TMD and UNS-PMF tended to learn from other colleagues (possibly by self-studying in groups), while the students from MUNI-FI are strongly individual learners.

3.2.2 Students' opinions about the first programming language

The main aim of the conducted survey was to investigate students' opinions regarding the programming language used in the introductory programming course. Namely, our main aim was to determine whether students learning basic programming principles in two different classes of programming languages (C-like and Pascal-like languages) have the same or different opinions about the difficulty of learning programming. Therefore, students were asked to evaluate the degree of learning difficulty related to learning language syntax and semantics, selecting and using appropriate language constructs when solving programming problems, implementation of the algorithms in the language, usability of existing language environments (compilers and IDEs), and correcting programs written in the language (questionnaire items L4-L12). However, in order to conclude something about introductory programming languages by the comparison of students' opinions, it is necessary to ensure that the opinions of students that have on average the same or highly similar cognitive capabilities are actually compared. Therefore, students were also asked to evaluate the degree of learning difficulty related to the understanding of programming structures and problems, as well as the degree of learning difficulty related to conceiving an algorithm for a given programming problem (questionnaire items L1-L3).

To statistically analyze differences between opinions of students that were introduced to programming in C-like languages (further referred to as "C group") and opinions of students that were introduced to programming in Pascal-like languages (further referred to as "Pascal group") two non-parametric statistical tests were employed. Namely, the Mann-Whitney U test (Mann and Whitney 1947) and the two-sample Kolmogorov-Smirnov test (Feller 1948) were used. Both tests do not assume any particular distribution of values (distribution-free methods) and can be applied to groups of different sizes. The Mann-Whitney U (MWU) test checks the null hypothesis that scores in one group do not tend to be systematically larger or smaller (stochastically superior/inferior) than scores in another, independent group. The null hypothesis is rejected if obtained p-value is smaller than 0.05. The MWU test ranks scores from both groups and calculates U statistics which is the number of times a score from the second group precedes a score from the first group in the ranked sequence of all scores. Under the null hypothesis, the distribution of the standardized U value, denoted by Z, asymptotically follows the standard normal distribution. Therefore, a statistically significant positive value of Z indicates that the scores in the first group tend to be greater than the scores in the second group, while a statistically negative value of Z indicates exactly the opposite. The two-sample Kolmogorov-Smirnov test (KS test) checks the null hypothesis that two cumulative distributions are not significantly different. The test relies on the maximal vertical distance between two distributions (D statistics). The null hypothesis is rejected if obtained p-value is smaller than 0.05.

Cognitive capabilities of the C and Pascal groups

The results of non-parametric statistical tests for questionnaire items L1 – L3 are summarized in Table 3. In all cases the null hypotheses of the tests were accepted (p-value > 0.05). This means that there are *no statistically significant differences between cognitive capabilities of students belonging to the C group and students from the Pascal group*. Namely, students from both groups perceive the difficulty to theoretically understand structures of imperative programming style as easy (item L1, Median = 2), while understanding of programming problems and conceiving an algorithm for a particular problem are perceived as activities of medium difficulty (items L2 and L3, Median = 3).

Table 3 The results of statistical comparison between the C group and the Pascal group of students for questionnaire items L1 – L3.

Item	Medians		MWU test			KS test		Null hypotheses
	C	Pascal	U	Z	p-value	D	p-value	
L1	2	2	8984.5	0.96	0.34	0.1	0.43	Accepted
L2	3	3	8881.5	0.84	0.4	0.14	0.16	Accepted
L3	3	3	8771.5	0.5	0.61	0.07	0.81	Accepted

Installing and using programming environments

Questionnaire items L4 and L5 are related to the programming environments used in our introductory programming courses. The two most dominant are Code::Blocks (for the C group of students) and XDS Modula-2 (for the Pascal group of students). The results of the statistical tests for those two questionnaire items are shown in Table 4. It can be seen that there are *no statistically significant differences between students from different groups with respect to the difficulty to install programming environments* that are used in our introductory programming courses – both groups perceive this task as an extremely easy task (item L4, median = 1). On the other hand, there are statistically significant differences related to the difficulty to use the environments in everyday learning practice. Although students from both groups on average perceive the usage of the environments as easy (item L5, median = 2), the application of the MWU test revealed that *students from the C group experience more troubles with the environments for C-like languages compared to students belonging to the Pascal group* ($Z > 0$, $p = 0.02$). 29.8% of students from the C group think that C-like environments are extremely easy to use, 34% perceive them as easy to use, 24% think that they are of medium difficulty, while 12.2% thinks that they are difficult or extremely difficult to use. On the other side, the same percentages for the Pascal group are: 45.9% (extremely easy), 26.5% (easy), 18.4% (medium), and 9.2% (difficult or extremely difficult). Therefore, we can state that the environments for Pascal-like languages are more user-friendly to novice students compared to the environments for C-like languages.

Table 4 The results of statistical comparison between the C group and the Pascal group of students for questionnaire items L4 – L5.

Item	Medians		MWU test			KS test		Null hypotheses
	C	Pascal	U	Z	p-value	D	p-value	
L4	1	1	9096	1.29	0.19	0.1	0.49	Accepted
L5	2	2	9720	2.28	0.02	0.16	0.06	**Rejected**

Difficulties in learning a programming language and using it for problem solving

Table 5 summarizes the results of the statistical tests for questionnaire items L6 - L12. It can be observed that the null hypotheses were accepted for each of those items. Therefore, we can conclude the following:

1. There are no statistically significant differences between the C group and the Pascal group considering the difficulty of learning introductory programming language syntax and semantics. Students from both groups *perceive the difficulty of learning introductory programming language as easy* (items L6 and L7, medians are equal to 2).
2. There are no statistically significant differences between students from the C group and students from the Pascal group considering the difficulty of solving problems in the introductory programming language. *Students from both groups think that transferring an algorithm to the programming language, selecting appropriate language structures for solving problems, and understanding and correcting compilation errors are easy tasks* (items L8, L9 and L11, respectively; medians are equal to 2). On the other hand, *problem decomposition into functions/procedures and finding semantic errors (bugs) in programs are perceived as activities of medium difficulty by both groups*.

Table 5 The results of statistical comparison between the C group and the Pascal group of students for questionnaire items L6 – L12.

Item	Medians		MWU test			KS test		Null hypotheses
	C	Pascal	U	Z	p-value	D	p-value	
L6	2	2	8751	0.77	0.44	0.08	0.69	Accepted
L7	2	2	8409	0.2	0.84	0.09	0.64	Accepted
L8	2	2	9001	0.91	0.36	0.08	0.72	Accepted
L9	2	2	8093	-0.48	0.63	0.05	0.98	Accepted
L10	3	3	8760	1.11	0.27	0.08	0.79	Accepted
L11	2	2	9300	1.4	0.15	0.08	0.77	Accepted
L12	3	3	9060	0.99	0.32	0.11	0.36	Accepted

4 CONCLUSIONS

In this paper a questionnaire survey conducted with the aim of investigating what students think and how they perceive the introductory programming courses and the programming languages used in their context are presented. Students from three different Institutions in three countries participated in the study. All respondents were classified into two categories: students that attended an introductory programming course in some C-like language and students that attended an introductory programming course in some Pascal-like language. Using non-parametric tests the opinions of those two groups of students were compared in relation to the difficulty of learning programming. The ultimate goal was to determine which language class is more appropriate for introductory programming courses from students' perspective.

The main conclusions drawn from the analysis of the collected data regarding students' perceptions for the introductory programming courses they attended can be summarized as follows:

- Students consider the contents and the balance between the theoretical and practical aspects of all three courses as highly appropriate.
- Students evaluated positively the quality, organization and contribution of our courses.
- Students at MUNI-FI study less than two hours per week, while students at UOM-TMD and UNS-PMF study from 2 to 4 hours. An interesting result recorded was that the students from UOM-TMD and MUNI-FI, in contrast with students from MUNI-FI, tend to learn from colleagues and possibly study in groups. This might make studying more fun and might be a possible explanation for the longer time of studying, but it is clear that it needs further investigation.

The main conclusions regarding students' perceptions on first programming language can be summarized as follows:

- The students learning the basics of programming in Pascal-like languages possess the same cognitive characteristics as the students learning the basics of programming in C-like languages, in terms of their abilities to theoretically understand structures of imperative programming style and develop algorithms for solving programming problems.
- There are no statistically significant differences between these two groups of students with respect to the degree of difficulty in learning the syntax and semantics of the introductory programming language, implementing an algorithm in the language, and correcting syntactical and semantic errors in their own programs. The only difference was recorded in terms of using the programming environment that is perceived to be more difficult for the C group.
- Since the compared groups of students have the same cognitive characteristics, it can be finally concluded that the choice of the introductory programming language does not have a deep impact on difficulties that novice students may experience when learning the basics of programming.

Of course it is important to note that the aforementioned results regarding students' perceptions on the FPL were recorded in the context of three different introductory programming courses that were positively evaluated by students in relation to their content, organization and quality. It seems that the FPL is not the most important factor for students' learning of the basics of programming. It is the overall quality of the course and the instructors that make the difference. This might explain the results of the studies reviewed in section 2 that showed no statistical difference in students' performance when some instructors changed the FPL utilized in their courses. Whatever the FPL is, instructors have to work hard in order to make the learning of programming concepts easier for students.

References

Aleksić, V., Ivanović, M., (2016). Introductory Programming Subject in European Higher Education. *Informatics in Education Journal*, Vol. 15, No. 2, 163-182.

Ali, A., & Smith, D. (2014). Teaching an introductory programming language in a general education course. *Journal of Information Technology Education: Innovations in Practice, 13,* 57-67.

Brilliant, S. and Wiseman, T. R. (1996). The first programming paradigm and language dilemma. In *Proceedings of the twenty-seventh SIGCSE technical symposium on Computer science education* (SIGCSE '96), Karl J. Klee (Ed.). ACM, New York, NY, USA, 338-342. DOI=http://dx.doi.org/10.1145/236452.236572

Decker, R., & Hirshfield, S. (1994). The Top 10 Reasons Why Object-Oriented Programming Can't Be Taught In CS1. *ACM SIGCSE Bulletin*, 26(1), 51–55.

Dierbach, C. (2014). Python as a first programming language. *J. Comput. Sci. Coll.* 29, 6 (June 2014), 153-154.

Duke, R., Salzman, E., Burmeister, J., Poon, J. and Murray, L. (2000). Teaching programming to beginners - choosing the language is just the first step. *Proceedings of the Australasian conference on Computing education (ACSE '00)*, 79-86.

Farag, W, Ali, S.. and Deb, D. (2013). Does language choice influence the effectiveness of online introductory programming courses?. In *Proceedings of the 14th annual ACM SIGITE conference on Information technology education*. ACM, New York, NY, USA, 165-170. DOI=http://dx.doi.org/10.1145/2512276.2512293.

Farooq MS, Khan SA, Ahmad F, Islam S, Abid A (2014). An Evaluation Framework and Comparative Analysis of the Widely Used First Programming Languages. *PLoS ONE* 9(2): e88941. doi:10.1371/journal.pone.0088941

Feller, W. (1948). On the Kolmogorov-Smirnov limit theorems for empirical distributions. *The Annals of Mathematical Statistics*, 19(2), 177-189.

Goosen G. L., Mentz E., Nieuwoudt E., Choosing the "Best" Programming Language?!, Proceedings of the Computer Science and IT Education Conference, 2007, pp. 269-282.

Hadjerrouit, S. (1998a). A Constructivist Framework for Integrating the Java Paradigm into the Undergraduate Curriculum. *ACM SIGCSE Bulletin*, 30(3), 105–107.

Hadjerrouit, S. (1998b). Java as first programming language: a critical evaluation. *SIGCSE Bull.* 30, 2 (June 1998), 43-47. DOI=http://dx.doi.org/10.1145/292422.292440

Hitz, M. and Hudec, M. (1995). Modula-2 versus C++ as a first programming language—some empirical results. In *Proceedings of the twenty-sixth SIGCSE technical symposium on Computer science education* (SIGCSE '95), Curt M. White, James E. Miller, and Judy Gersting (Eds.). ACM, New York, NY, USA, 317-321. DOI=http://dx.doi.org/10.1145/199688.199838

Ivanović M., Budimac Z., Radovanović M., Savić M. (2015). Does the choice of the first programming language influence students' grades? In *Proceedings of the 16th International Conference on Computer Systems and Technologies, CompSysTech'15*, June 25-26, Dublin, Ireland, ACM International Conference Proceeding Series, Vol. 1008, ACM Inc., N.Y. USA, Pages 305-312, doi>10.1145/2812428.2812448

Jabłonowski, J. (2007). A case study in introductory programming. In *Proceedings of the 2007 international conference on Computer systems and technologies* (CompSysTech '07), Boris Rachev, Angel Smrikarov, and Dimo Dimov (Eds.). ACM, New York, NY, USA, Article 82, 7 pages. DOI=http://dx.doi.org/10.1145/1330598.1330685

Leping, V., Lepp, M., Niitsoo, M., Tõnisson, E., Vene, V. and Villems, A. (2009). Python prevails. In *Proceedings of the International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing* (CompSysTech '09), Boris Rachev and Angel Smrikarov (Eds.). ACM, New York, NY, USA, , Article 87 , 5 pages. DOI=http://dx.doi.org/10.1145/1731740.1731833

Mann, H. B, and Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 18(1), 50-60.

Moritz, S. H., Blank, G. D. (2005). A Design-First Curriculum for Teaching Java in a CS1 Course, *ACM SIGCSE Bulletin*, 37(2), 89 – 93.

Radensky, A., Zivkova, E., Petrova, V., Lesseva, R. and Zascheva, C. (1988). Experience with Ada as a first programming language. *SIGCSE Bull.* 20, 4 (December 1988), 58-61. DOI=http://dx.doi.org/10.1145/54138.54149.

Sanders, I. and Langford, S. (2008). Students' perceptions of python as a first programming language at wits. In *Proceedings of the 13th annual conference on Innovation and technology in computer science education* (ITiCSE '08). ACM, New York, NY, USA, 365-365. DOI=http://dx.doi.org/10.1145/1384271.1384407

Skublics, S. and White, P. (1991). Teaching Smalltalk as a first programming language. In *Proceedings of the twenty-second SIGCSE technical symposium on Computer science education* (SIGCSE '91). ACM, New York, NY, USA, 231-234. DOI=http://dx.doi.org/10.1145/107004.107046

Tempte, M C. (1991), Let's Begin Introducing the Object-Oriented Paradigm, *ACM SIGCSE Bulletin*, Vol. 23, No. I, 338–342.

Wick, M. (1995). On Using C++ and Object-Orientation in CS1: the Message is still more important than the Medium. *ACM SIGCSE Bulletin*, 27(1), 322–326.

Xinogalos, S. (2016). Designing and deploying programming courses: Strategies, tools, difficulties and pedagogy. *Education and Information Technologies*. Springer Sciece+Business Media New York 2016. Vol. 21, Issue 3, 559-588. DOI: 10.1007/s10639-015-9433-1.

Yadin, A. (2011). Reducing the dropout rate in an introductory programming course. *ACM Inroads* 2, 4 (December 2011), 71-76. DOI=http://dx.doi.org/10.1145/2038876.2038894