The importance of learning computer programming for engineers

Carlos Oliveira
Department of computing
Rio de Janeiro Federal Institute of Education Science and Technology
Rio de Janeiro, Brazil
carlos.roberto@ifrj.edu.br

Abstract— Programming is important because it helps people to do their work better, faster and in a more efficient manner. Learning computer programming can help people to develop skills like critical reading, analytical thinking and creative synthesis. This paper aims to show the importance of learning computer programming and presents a case study of an introductory programming course.

Keywords—case study; computer programming; education.

I. INTRODUCTION

Computing is a vital part of what everybody does nowadays, we all interact with computers. Thus, it is important to know how to properly operate computers and to take advantage of them. Knowing a computer programming language allows the person to be not only a technology user but to create their own computer programs. Thus, programming computers is a concern not only for IT professionals but also for children parents that believe, in the future, not knowing the language of computers will be as challenging as being illiterate or innumerate are today [1]. This is not primarily about equipping the next generation to work as computer programmers; it is about promoting computational thinking. Computational thinking is a skill that everyone should learn; it is how computer programmers solve problems. It combines mathematics, logic and algorithms, and teaches the person a new way to think about the world. Computational thinking teaches how to tackle large problems by breaking them down into a sequence of smaller, more manageable problems. It allows you to tackle complex problems in efficient ways that operate at huge scale. It involves creating models of the real world with a suitable level of abstraction, and focus on the most pertinent aspects. Even if the person never becomes a computer programmer, she or he will benefit from knowing how to think this way. It helps to understand and master technology of all sorts and solve problems in almost any discipline.

Thus, learning a computer programming language can also be useful for engineers, as it allows the creation of computer programs to solve many mathematical problems present on engineering. Programming is important because it helps people to do their work better, faster and in a more efficient manner. It also helps to increase the competitive edge of a company by providing innovative technologies to their customers. Learning to program is important because it develops analytical and problem solving abilities. It is a creative activity and provides us a mean to express abstract ideas. By designing programs, we learn many skills that are important for all professions. These skills can be summarized as critical reading, analytical thinking and creative synthesis [2].

The rest of this paper is organized as follows. Section II shows a case study of an introductory programming course and its challenges. Section III presents different programming languages and aim to explain our choice for a programming language. Section IV concludes the paper.

II. CASE STUDY

We teach an introductory programming course for computer science and engineering students. Our goal is to develop logical and computational thinking, and also introduce the first programming language the student will learn.

In this course, the student learns thinks like (1) understanding the conditional statement, where part of the sentence states a condition or requirement. The other part of the sentence states the result of that condition. If the condition is fulfilled, the result will occur; (2) Understanding premise and conclusion shorthand: the two parts of a conditional statement have specific terms with respect to logic. The first part is called a premise, and the second part is called a conclusion. Within a conditional statement, if a premise is true, the conclusion will be too, because it follows, or results from, the truth of the premise. Once we understand the structure of an original conditional statement in terms of premise and conclusion, we can understand three other statements related to it. They are the converse, the inverse, and the contrapositive. Knowing these three is important to avoid faulty reasoning and to detect faulty reasoning by others; (3) the converse of the original conditional statement simply reverses the premise and the conclusion. In shorthand terms, therefore, the converse is "conclusion then premise"; (4) the inverse of the
original statement keeps the original premise and original conclusion but negates each one; (5) the contrapositive is either the converse of the inverse or the inverse of the converse. That is, it involves a negation of both the premise and the conclusion, along with their reversal; (6) closely related to the conditional and related statements are the ideas of necessary conditions and sufficient conditions. A necessary condition is one that must be met for a certain result to be achieved; (7) A sufficient condition is one that, if met, absolutely guarantees the occurrence of a certain result; (8) A condition can be necessary but not sufficient; (9) A condition can be sufficient but not necessary; (10) A condition can be neither necessary nor sufficient with respect to a result.

Despite of the benefits someone can have by learning computer programming, there are potential sources of difficulty that must be mastered. Du Boulay [3] summarizes them as: (1) general orientation, what programs are for and what can be done with them; (2) the notional machine, a model of the computer as it relates to executing programs; (3) notation, the syntax and semantics of a particular programming language; (4) structures, that is, schemas/plans; (5) pragmatics, that is, the skills of planning, developing, testing, etc.

As the experience in this course showed, computer programming needs to be a hands-on course. Mathematical logic is the subject of introductory classes. Later, the student learns the first words used by the programming language. Then, the student is able to create a simple program, as presented in Figure 1 that shows the message “Hello world!” on the screen. Figure 2 shows the output, for the source code in figure 1, shown on the computer screen. The next step is to introduce more complex structures on the source codes.

```
Program Code Example;
Begin
  textbackground( red );
  textcolor( lightcyan );
  write('Hello, world!');
  readkey;
End.
```

Fig. 1. Example of source code

One problem faced by teachers is that several students that are very well on classes do not achieve good grades. It happens because of the lack of practical lessons that needs to be taken by the student. Computer programming is an abstracted subject and its learning is facilitated by practical lessons [4]. It is recommended that for each hour of class the student spends one hour more in front of the computer creating program codes. Doing this, the student will succeed in the course.

```
Hello, world!
```

Fig. 2. Figure 1 program output
III. PROGRAMMING LANGUAGE TYPES

A programming language needs to be chosen in a programming course. There are several languages, e.g., Java, C, Python, etc. These programming languages can be divided in different types. In this section we describe the two main programming language types, procedural and object-oriented. Pascal is a procedural programming language used in the course described on section II. Nevertheless, we describe the object-oriented languages in this section in order to show the differences and to evidence that procedural languages are better for introductory courses.

A. Procedural Programming Languages

Procedural language is a type of computer programming language that specifies a series of well-structured steps and procedures within its programming context to compose a program. It contains a systematic order of statements, functions and commands to complete a computational task or program [5]. Procedural language, as the name implies, relies on predefined and well-organized procedures, functions or sub-routines in a program’s architecture by specifying all the steps that the computer must take to reach a desired state or output.

Procedural language segregates a program within variables, functions, statements and conditional operators. Procedures or functions are implemented on the data and variables to perform a task. These procedures can be invoked anywhere between the program hierarchy, and by other procedures as well. A program written in procedural language contains one or more procedures.

B. Object-oriented Programming Languages

Object-oriented programming is an approach to problem-solving where all computations are carried out using objects. An object is a component of a program that knows how to perform certain actions and how to interact with other elements of the program. Objects are the basic units of object-oriented programming. A simple example of an object would be a person. A person is expected to have a name. This would be considered a property of the person. A person is also expected to be able to do something, such as walking. This would be considered a method of the person.

A method in object-oriented programming is like a procedure in procedural programming. The key difference here is that the method is part of an object. In object-oriented programming, you organize your code by creating objects, and then you can give those objects properties and you can make them do certain things.

Object-oriented languages help to manage complexity in large programs. Thus, it is recommended to IT professionals not to beginners aiming to write simple programs and develop other skills. Objects package data and the operations on them so that only the operations are publicly accessible and internal details of the data structures are hidden. This information hiding made large-scale programming easier by allowing a programmer to think about each part of the program in isolation. In addition, objects may be derived from more general ones, inheriting their capabilities. Such an object hierarchy made it possible to define specialized objects without repeating all that is in the more general ones [6].

A key aspect of object-oriented programming is the use of classes. A class is a blueprint of an object. A class can be understood as a concept, and the object as the embodiment of that concept. Thus, in a program that uses a person as an example, a class called 'person' would provide a blueprint for what a person looks like and what a person can do [7].

One of the most important characteristics of procedural programming is that it relies on procedures that operate on data - these are two separate concepts. In object-oriented programming, these two concepts are bundled into objects. This makes it possible to create more complicated behavior with less code. The use of objects also makes it possible to reuse code. Once you have created an object with more complex behavior, you can use it anywhere in your code.

C. Pascal programming language

Pascal [8] is the programming language used in this case study. This language was chosen because of its characteristics. It does not present many details and it is easy to learn. Pascal is also called an “education-oriented” language by authors in [6].

About 1970 Niklaus Wirth designed Pascal to teach structured programming, which emphasized the orderly use of conditional and loop control structures without GOTO statements. Although Pascal resembled ALGOL in notation, it provided the ability to define data types with which to organize complex information, a feature beyond the capabilities of ALGOL as well as FORTRAN and COBOL. User-defined data types allowed the programmer to introduce names for complex data, which the language translator could then check for correct usage before running a program [6]. The goal of Niklaus Wirth was to facilitate forming of a good style of algorithmic thinking, and, thus, programming. In particular, he aimed to make Pascal syntax intuitively clear even for beginners.
For years Pascal was one of the most widely used languages for programming instruction. It was available on nearly all computers, and, because of its familiarity, clarity, and security, it was used for production software as well as for education [6].

IV. CONCLUSION

Computational methods and models give us the courage to solve problems and design systems that no one would be capable of tackling alone [9]. Learning computer programming helps to use abstraction and decomposition when attacking a large complex task or designing a large complex system. It also helps to separate concerns and to choose an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable. Thus, it helps to develop skills that are important for all professions. Hence, it shows the importance of learning programming.

REFERENCES


BIOGRAPHY

Carlos Oliveira is an Associate Professor of computer programming in the Rio de Janeiro Federal Institute of Education Science and Technology, Rio de Janeiro, Brazil. He earned B.S. in Computer Science from Rio de Janeiro State University, Brazil and Masters in Computer Science from Fluminense Federal University, Brazil. He has published several conference papers in Brazil, US, Europe and Asia. In the last four years he was designated reviewer for 47 computing and engineering international conferences and journals, most of them in Europe and Asia. Prof. Oliveira has done research projects with Faperj and Cnpq, two brazilian research funding agencies. His research interests include computer programming, educational games, operating systems and distance education. He is a member of the Brazilian Computing Society.