International Journal of
Inclusive and Sustainable Education
ISSN: 2833-5414
Volume 1 | No 6 | Dec-2022

The Importance and Need of C and C++ Programming Language Teaching in Higher Education Institutions

Ibragimova Mohigul Komiljon qizi [1], Xurramov Ruslan Erkin o'g'li [2]

[1,2] Termiz State University Teacher of the Department of Information Technologies

Abstract: This article is a little about why it is still worth choosing C ++ as the first programming language for teaching students and the problems of teaching in universities. Part of the judgment regarding the teaching process is based on personal teaching experience (three years, during postgraduate studies) as well as interaction with teachers and students.

The advantages of the C++ programming language are discussed specifically for learning and in terms of better understanding of key concepts. I emphasize once again: not for the sake of industrial development and not from the point of view of a subjective criterion such as convenience. First of all, I would like to note that the choice of the first programming language for students of non-major specialties is not considered. Algorithm basics can be mastered (if necessary) even in Python, even in C++ (Java, C#, Pascal, etc.). In this case, the simpler the language, the better: people should at least develop a culture of algorithmic thinking and understanding of basic structures. Students of some specialties (legal, economic, humanitarian) do not need to study PL at all

Keywords: C, C++, high-level programming languages, low-level programming languages, compiler, STL, OOP.

This article is a little about why it is still worth choosing C ++ as the first programming language for teaching students and the problems of teaching in universities. Part of the judgment regarding the teaching process is based on personal teaching experience (three years, during postgraduate studies) as well as interaction with teachers and students.

The advantages of the C++ programming language are discussed specifically for learning and in terms of better understanding of key concepts. I emphasize once again: not for the sake of industrial development and not from the point of view of a subjective criterion such as convenience. First of all, I would like to note that the choice of the first programming language for students of non-major specialties is not considered. Algorithm basics can be mastered (if necessary) even in Python, even in C++ (Java, C#, Pascal, etc.). In this case, the simpler the language, the better: people should at least develop a culture of algorithmic thinking and understanding of basic structures. Students of some specialties (legal, economic, humanitarian) do not need to study PL at all.

Despite the obviousness of the above, most first-year economics students are taught to program in Pascal in practical computer science classes. Students who do not yet know how to work with MS Word. The benefits of such activity are very, very doubtful. Similarly, math students can learn to program in C++/C#/Java for several years...but why? It is very useful to apply your knowledge later to learn programs like Mathcad, Simulink, Surfer, etc.

In view of the above, we consider the process of choosing a first language only for students of specialized (e.g. "Software Engineering") and mixed majors (e.g. "Applied Mathematics and Informatics") with an IT inclination. First, the curriculum for such specialties includes a sufficient number of lectures and practices (because the first language is taken into account, only the first course is taken into account): for two interrelated subjects (informatics and programming) about 230 hours. specialty. Secondly, the presence of interest and a certain way of thinking in students. Such students have often tried programming and even written a website/toy. Together, these two reasons create a good foundation for starting to learn and lower the barrier to entry for language learning. In addition, graduates of the considered specialties will have to work in the field of software development in the future. Therefore, the choice of the first language is especially important for them.

Why C++?

In the first year, the basis for additional education is created and the student's approach to further acquisition of knowledge is formed. The programming language plays an important role here. There are four reasons to choose C++ as your first programming language:

a. A compiled language with static typing.

b. A combination of high and low level tools.

c. Implementation of OOP.

d. STL.

e. Support for object-oriented programming (OOP).

f. OOP helps make writing code easier and faster.

g. High speed.

h. Data processing capabilities are at a low level - that is, at a level close to the hardware. With this you can write drivers, microcontrollers in C++.

Why learn C?

C programming skills are very important and universities should teach C instead of C++. Let C++ support OOP, C is the language that 90% of programmers should know. 10% may rely on C++ or any other high-level language. I've taught C#, Java, Visual Basic, ASP.NET, and many other high-level languages in my programming career, and I've always had people complain about their speed. You know what they always say? Learn C or C++ because they compile to machine code, which means they're super fast!

Let's take a closer look at these reasons.

Compiler. This is where C++ is at its best. Many compilers, console commands, program assembly steps... Yes, you need to write the first program in a simple text editor without syntax highlighting and autocompletion, find out what and how to run. This approach forms a person's understanding of how everything works:

Program code is text that does not run on its own.

A compiler is a separate program that needs to be told what to do with the source code and turn it into an executable file. A text editor is also a separate program for writing source code.

There are build options and there are several compilers.

The source code written by the programmer can be preprocessed and modified (for example, by the processor).

The future expert knows that the code itself does not work (in the future, he may be interested in, for example, how the Python interpreter or JIT compilation works). A person asks himself questions: "Why?", "What's the difference?", "How?". There will be no illusion that everything works at the click of two magic buttons or from an interactive command line. The student learns that the process of creating an application can be customized and that the source code can be processed by third-party programs. In the future, when using an IDE, a person will understand that it is a convenient set of programs that perform many simple operations, and if flexibility is not enough, it can be abandoned or expanded.

Static typing. It's easier to understand what a data type is by using an example of a statically typed language,

Programming is the process by which a programmer creates software that runs on a computer and solves a limited number of problems. People have been programming since the invention of computers. At first, this was done using vacuum tubes, and after the invention of transistors, binary codes were used. The first low-level programming languages appeared at the same time as the first programmable computer, Colossus. Since then, many changes have taken place in this field, developers are constantly improving computer systems. Today, we can observe computers that are thousands of times more powerful than those at the beginning of the technological era. OS and Programming Initially, all computers were the same and used the same architecture, but over time, the need for an operating system to control the basic operations of computer hardware arose. So there were several OSes. Dennis Ritchie one of them was the man who rewrote Unix from assembly language to C. To "teach" operating systems, you needed a programming language to tell them what to do. Thus, programming is the process of creating a set of instructions that tell the OS what operations to perform.

C is a low-level programming language that supports direct hardware control. It was developed by Dennis Ritchie in 1973 at AT&T Labs. With this language, you can easily manage memory, CPU, registers and even connected devices! Assembly language was also one of the first low-level programming languages invented after binary code, but it is less widely read than C. C is also known as the father of modern programming. Although there were other languages like BASIC out there, C had many advantages over its competitors. C is a general purpose language. It has no classes, no interfaces. But it has much more useful and powerful tools - for example, pointers that allow direct memory management during program execution, etc. C was also used as a "building material" for other, higher-level languages: Java, D, C#. C++ and Objective-C are compiled to C code and then compiled to native code for faster execution.

Why don't they write good software anymore?

The new generation of developers is busy attracting the attention of the audience, especially girls, their main task is not to build a better digital world. I asked "Why am I getting a null error?" I saw a lot of questions like Are you serious? The person who asks such a question does not understand the essence of programming, does not want to solve the problem himself. He had just heard somewhere that Bill Gates was the richest man on earth and decided that he might as well be. He only needs to create software, and soon he will be a billionaire! This mentality of modern programmers does not allow them to create good programs.

University level problems

In our universities, they only teach C++ and only on paper. To me, teaching programming without practice is like killing the concept of programming. And although this is due to the lack of good staff in universities, new programmers should always be trained in programming skills.

Side effects of not knowing low level PLs:

Many embedded devices and microcontrollers require low-level programming and understanding of its concepts: memory management, pointers, etc. A modern developer (who only writes .NET and Android apps) would never be able to program such devices. And while it can be programmed in .NET, the connected devices themselves must use a low-level language to get the best results.

Lack of support.

There is no simple support for low-level languages such as assembler or C. Visual Studio (the best IDE in my opinion) no longer supports assembler projects (though MASM still does). Because of this misunderstanding, most major colleges do not teach students low-level languages that explain the concept of programming. It kills any student's desire to do better.

Books

1. D.S. Malik. C++ Programming: From problem analysis to program design. Course Technology, 2011.
2. Б. Страуструп. Язик программирования C++. Спетсиалное издание.М.: ООО «Бином-Пресс», 2006.
3. Madraximov Sh.F., Gaynazarov S.M. C++ tilida Dasturlash asoslari//
4. Toshkent, O‘zbekiston Milliy Universiteti, 2009.
5. Stiven Prata. C++ Primer Plus. 2011
6. https://habr.com/ru
7. Energiya samaradorligini nazorat va boshqarishning axborot dasturiy ta’minoti va smart qurilmalar
8. A Abdumalikov, Y Anvar, B Doniyor