УДК 53-3937

Kirill Sherbakov[1], Valentyna Kuzmenko[2]
[1]student of group CST-211SP NU "Zaporizhzhia Polytechnic"

[2]senior teacher NU "Zaporizhzhia Polytechnic"

PROGRAMING LANGUAGES IN PROGRESS

The first programs looked like setting up key switches on the front of a computing device. With the development of computer technology, a machine language appeared, with the help of which a programmer could set commands, operating with memory cells, making full use of the capabilities of the machine. However, using most computers at the machine language level is difficult, especially when it comes to I/O. "Words" in machine language are instructions, each of which represents one elementary action for the central processor, such as reading information from memory location.

In the case when it is necessary to have an effective program, instead of machine languages, machine-oriented languages close to them are used - assemblers. People use mnemonic commands instead of machine commands. But even working with assembler is quite complicated and requires special training.

The next step was taken in 1954 with the creation of the first high-level language, Fortran. High-level languages imitate natural languages using some spoken language words and common mathematical symbols. These languages are more human friendly and can be used to write programs up to several thousand lines long. However, while easily understood in short programs, the language became unreadable and difficult to manage when it came to large programs.

The solution to this problem came after the invention of structured programming languages such as Algol (1958), Pascal (1970), C (1972).Structured programming includes well-defined control structures, program blocks, non-conditional jump (GOTO) instructions, self-contained subroutines, support for recursion and local variables. The essence of this approach lies in the possibility of splitting the program into its constituent elements. Although structured programming produced outstanding results when used, even it failed when the program reached a certain length. To write a more complex (and longer) program, a new approach to programming was needed.

As a result, the principles of object-oriented programming were developed in the late 1970s and early 1980s. OOP combines the best principles of structured programming with powerful new concepts such as encapsulation, polymorphism, and inheritance. Examples of object-oriented languages: Object Pascal, C++. OOP allows you to optimally organize programs by breaking the task into its component parts and working with each separately. A program in an object-oriented language that solves a specific problem, in fact, describes the part of the world associated with this problem.