Volodymyr Miroshnychenko
E. V. Grishchenko, research supervisor
S. I. Kostrytska, language adviser
National Mining University, Dnipro, Ukraine

Comparative Characteristics of Programming Languages for Solving Cross-platform Problems

The main purpose of cross-platform programming is to create software which will function correctly on different operation systems and hardware platforms. For writing such type of software we use programming languages that have compilers for various platforms (C, C++, Pascal). Besides, we use interpreted languages (Java, Python).

Despite the extensive set of libraries, languages and programming environments, writing an application which will work on a number of different platforms without any bugs can be problematic for a number of reasons:

1. Difficulty in ensuring stable operation of the program on the required types of operation systems leads to the fact that the developer has to cut back some functionality.
2. Simple rigid positioning of interface elements in the application is not possible as each platform often has different user interface conventions and cross-platform application cannot always be accommodated to them.
3. During testing, the application can interact differently with platforms, which leads to unexpected types of bugs, crashes or incorrect runnig, and this is the main cause of a long code debugging.
4. Each platform requires its own format of installation packages (".msi" for Windows and ".rpm" for Linux).

There are different ways of solving the problems that arise when implementing cross-platform program.

The most straightforward option is to write separate blocks of code for each of the required platforms. It is a longtime approach, but as a result, we will have as much optimized product as possible.

In addition to the method mentioned above, the abstract platform approach is used. It is the case when the program "does not realize" on which platform it is running. An excellent example of this method is Java Virual Machine (JVM), software that runs program on various types of OS.

A mixed approach of cross-platform programming can also be applied. For example, the Mozilla Firefox web-browser uses an abstraction to create some low-level components and its Graphical User Interface (GUI), implemented with the help of XUL.

Besides, Mozilla in question uses separate blocks of code in different scripting languages (C, C ++, JS, Rust) to reach easier portability.

To solve the problem of installing programs on different OS, Flexera Software has developed InstallAnywere, a tool allowing to assemble IP for multiple platforms.
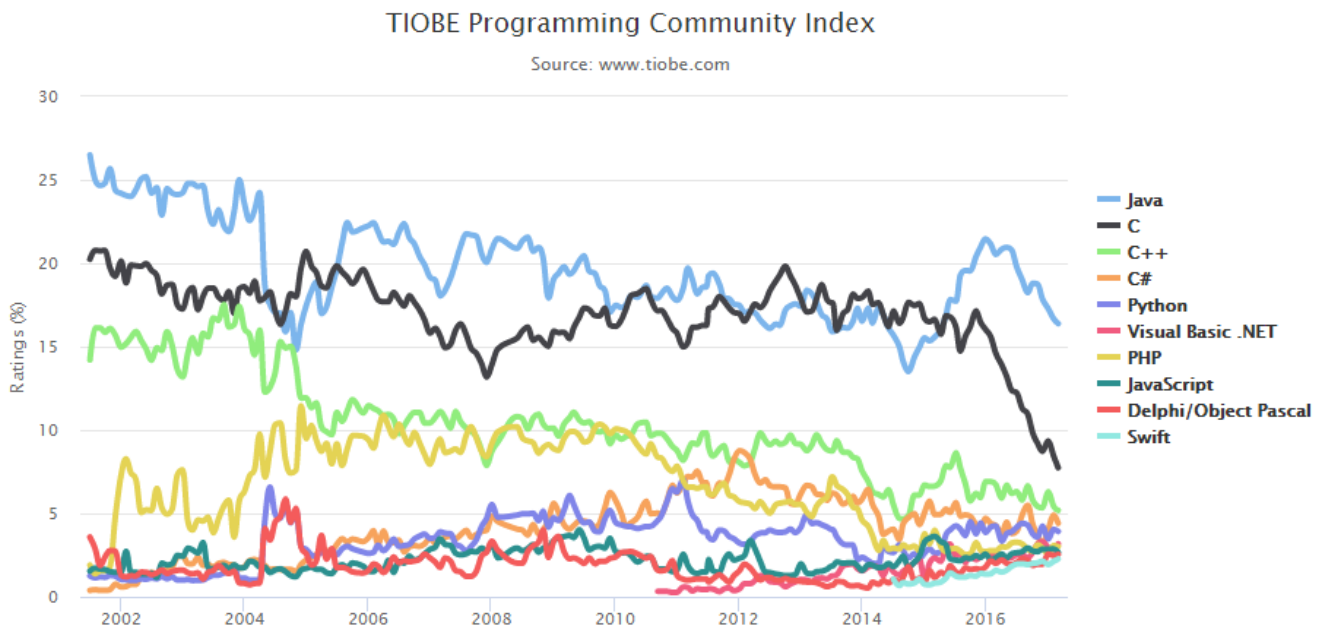

Figure. 1
TIOBE Index for March 2017

If we turn to the world rankings (Fig. 1), we will see that the most popular language is Java (16.38%), which is considered as the main language for cross-platform programming, followed by C (7.74%), C++ (5.18%), C# (4.49%), Python (3.91%) etc.

It is thanks to JVM that Java has gained such popularity (the programmer only needs to write the code once, and JVM has already taken care about how it will work on all other platforms). However, the Java language does not provide the ability to work with memory in the same way that C++ does. On the one hand, such a peculiar "safety belt" does not allow the programmer to screw up with memory, but on the other hand, it deprives the programmer of the opportunity to optimize the product to the maximum.

The choice of language directly depends on the approach by which the cross-platform will be implemented. Interpreted languages such as Python, Ruby, Perl suit small multiplatform projects better, where you will use an abstract model. Java is for the more serious projects of the same type. In turn, such languages as C, C++ will be better for writing programs, where you need to reduce the use of the device memory to the minimum. The bright example of cross-platform application is Sublime Text, proprietary code editor which works on Linux, Mac OS X and Windows. It supports many markup and programming languages. This editor is written in C++ and its API is implemented with the help of Python.