PROGRAMMING TECHNOLOGY AND MAIN TRENDS OF THEIR DEVELOPMENT

Djuraeva Dilafruz Raupovna
Doctoral Student of the Navoi State Institute of Education,Uzbekistan

---

ABSTRACT

The methodological approach of this study is to draw a parallel between the evolution of programming (programming paradigms), the development of teaching methods of programming and, finally, the paradigms of education in general.

Key words: Information technology, computer science, programming, algorithm, software languages.

Today in the world there is a rapid informatization of all aspects of society and all spheres of production activity. Informatics is becoming one of the fundamental areas of scientific knowledge, which studies information processes, methods, as well as information processing tools. This discipline is developing rapidly, it is closely related to the use of information technology, and the field of its application in life is expanding every day.

Graduates of pedagogical universities need to have sufficient knowledge and skills to effectively use modern information technologies in their future activities. The main question in the informatics teaching system remains whether it is necessary for a graduate to be able to solve arising problems using programming, or whether it is sufficient to master user technologies and the skills of finding ready-made solutions.

The main argument against teaching programming at a university has recently become its complexity and narrow specialization.

To understand the existing programming technologies and determine the main trends in their development, it is advisable to consider these technologies in a historical context, highlighting the main stages in the development of programming as a science.

The first stage is the period from the moment the first computers appeared until the middle of the 60s of the XX century. During this period, there were practically no formulated technologies, and programming was actually an art. The first programs had the simplest structure. They consisted of the actual machine language program and the data processed by it. The complexity of programs in machine codes was limited by the ability of the programmer to simultaneously mentally track the sequence of operations performed and the location of data during programming.

The second stage is a structured approach to programming (60-70s of the XX century). The structural approach to programming is a set of recommended technological methods that cover the implementation of all stages of software development.

Support for the principles of structured programming was laid in the foundation of the so-called procedural programming languages. As a rule, they included basic "structural" control transfer operators, supported subroutine nesting, localization, and data scoping.

Simultaneously with structured programming, a huge number of languages appeared, based on other concepts, but most of them could not stand the competition. Some languages were simply forgotten, the ideas of others were later used in the next versions of the languages being developed.

Further growth in the complexity and size of the software being developed required the development of data structuring. As a consequence, it becomes possible for languages to define custom data types. At the same time, the desire to delimit access to the global data of the program has intensified in order to reduce the number

of errors that arise when working with global data. As a result, the technology of modular programming appeared and began to develop. This technology is supported by modern versions of Pascal and C (C ++) languages, Ada and Modula languages.

The third stage is an object-oriented approach to programming (from the mid-80s to the end of the 90s of the XX century.) Object-oriented programming is defined as a technology for creating complex software, based on the representation of a program as a set of objects , each of which is an instance of a certain type (class), and the classes form a hierarchy with inheritance of properties [1, 2].

The main advantage of object-oriented programming in comparison with modular programming is the "more natural" software decomposition, which greatly facilitates its development. This leads to a more complete localization of data and their integration with processing routines, which allows practically independent development of individual parts (objects) of the program.

The rapid development of programming technologies based on the object approach has solved many problems. Thus, environments were created that support visual programming, for example, Delphi, C ++ Builder, Visual C ++, etc. When using the visual environment, the programmer has the opportunity to design some part, for example, the interfaces of the future product, using visual means of adding and customizing library components.

The use of the object approach has many advantages, but its concrete implementation in object-oriented programming languages such as Pascal and C ++ has significant drawbacks:

• in fact, there are no standards for linking the binary results of compiling objects into a single whole even within the same programming language: linking objects obtained by different C ++ compilers is problematic at best, which leads to the need to develop software using the tools and capabilities of one high-level programming language and one the compiler, which means that it requires the source codes of the class libraries used;

• a change in the implementation of one of the software objects is at least associated with the recompilation of the corresponding module and the recompilation of all software using this object.

Thus, when using these programming languages, the dependence of software modules on the addresses of exported fields and methods, as well as data structures and formats, is preserved.

The fourth stage is the component approach and CASE-technologies (from the mid-90s of the XX century to our time). The component approach involves building software from separate components - physically separate pieces of software that interact with each other through standardized binary interfaces. Unlike ordinary objects, component objects can be assembled into dynamically called libraries or executable files, distributed in binary form (without source code), and used in any programming language that supports the corresponding technology.

A distinctive feature of the current stage of development of programming technology, in addition to changing the approach, is the creation and implementation of automated technologies for the development and maintenance of software, which were called CASE-technologies (Computer-Aided Software / System Engineering - development of software / software systems). using computer support). Without automation tools, the development of a rather complex software is currently becoming difficult to implement: a person's memory is no longer able to capture all the details that must be taken into account when developing software. Today, there are CASE technologies that support both structural and object (including component) approaches to programming.

The emergence of a new approach does not mean that from now on all software will be created from software components, but an analysis of the existing problems of developing complex software shows that it will be widely used.

LITERATURE

1. Ivanova G.S. Programming technology: Textbook for universities. –M: Publishing house of MSTU named after NE Bauman, 2003. –320 p.
2. Vendrov A.M. CASE - technologies. Modern methods and tools for designing information systems. - M .: Finance and statistics, 1998 .-- 176 p.