|  |
| --- |
| Topic: Programs and the Programming Languages  **1**. Transcribe and read these terms relating to the computers and computer manufacturing industry. Translate them. Make use of a specialized dictionary.  Deal; source; several; a brief description; means; artificial intelligence; general purpose language; support; consequently; specific.  **2**. Translate these words without using a dictionary.  Compile; compiler, interpret; convert; algorithmic; machine code, a system program, instruction, introduce, type, routines, manufacturer of the machines.   1. Vocabulary:   a high-level language – язык высокого уровня  a source program – исходнаяпрограмма  to compile (interpret) into machine code - преобразовыватьвмашинныйкод  a portable language – переносимыйязык  an application program – прикладнаяпрограмма  an object program –объектнаяпрограмма  a system program – системнаяпрограмма  a load module –загрузочныймодуль  secondary memory –вторичнаяпамять.  **4**.Read and translate the text. |
| **Text A. Programs and Programming Languages** |
| Computers can deal with different kinds of problems if they are given the right instructions for what to do. Instructions are first written in one of the high-level languages, e.g. FORTRAN, COBOL. ALGOL, PL/I, PASCAL, BASIC, or C, depending on the type of problem to be solved. A program written in one of these languages is often called a source program, and it cannot be directly processed by the computer until it has been compiled, which means interpreted into machine code. Usually a single instruction written in a high-level language, when transformed into machine code, results in several instructions. Here is a brief description of some of the many high-level languages:  FORTRAN acronym for FORmulaTRANslation. This language is used for solving scientific and mathematical problems. It consists of algebraic formulae and English phrases. It was first introduced in the United States in 1954.  COBOL acronym for COmmon Business-Oriented Language. This language is used for commercial purposes. COBOL, which is written using English statements, deals with problems that do not involve a lot of mathematical calculations. It was first introduced in 1959.  ALGOL acronym for ALGOrithmic Language. Originally called IAL, which means International Algebraic Language. It is used for mathematical and scientific purposes. ALGOL was first introduced in Europe in 1960.  PL/I Programming Language I. Developed in 1964 to combine features of COBOL and ALGOL. Consequently, it is used for data processing as well as scientific applications.  BASIC acronym for Beginner's All-purpose Symbolic Instruction Code. Developed in 1965 at Dartmouth College in the United States for use by students who require a simple language to begin programming.  С developed in the 1970s to support the UNIX operating system. С is a highly portable general-purpose language.  Other such languages are APL (developed in 1962). PASCAL ( named after Blaise Pascal and developed in 1971), and LISP and PROLOG, both of which are used for work in artificial intelligence. LOGO is a development of LISP (LISt Processing) which has been used to develop computer-based training (CBT) packages.  When a program written in one of these high-level languages is designed to do a specific type of work such as calculate a company's payroll or calculate the stress factor on a roof, it is called an applications program. Institutions either purchase these programs as packages or commission their own programmers to write them to meet the specifications of the users.  The program produced after the source program has been converted into machine code is referred to as an object program or object module. This is done by a computer program called the compiler, which is unique for each computer. Consequently, a computer needs its own compiler for the various high-level languages if it is expected to accept programs written in those languages. For example, in order that an IBM RS/6000 may process a program in FORTRAN, it needs to have a compiler that would understand that particular model and the FORTRAN language as well.  The compiler is a systems program which may be written in any language, but the computer's operating system is a true systems program which controls the central processing unit (CPU), the input, the output, and the secondary memory devices. Another systems program is the linkage editor, which fetches required systems routines and links them to the object module (the source program in machine code). The resulting program is then called the load module, which is the program directly executable by the computer. Although systems programs are part of the software, they are usually provided by the manufacturer of the machine.  Unlike systems programs, software packages are sold by various vendors and not necessarily by the computer manufacturer. They are a set of programs designed to perform certain applications which conform to the particular specifications of the user. Payroll is an example of such a package which allows the user to input data – hours worked, pay rates, special deductions, names of employees – and get salary calculations as output. These packages are coded in machine language (0s and 1s) on magnetic tapes or disks which can be purchased, leased, or rented by users who choose the package that most closely corresponds to their needs.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Note: payroll – list of employees and the amount of money to be paid to each of them.  **5**. Find in the text English equivalents to the following Russian ones.  Решать проблему;в зависимости от типа проблемы, которую предстоит решить; представлять; искусственный интеллект; для поддержки системы; следовательно; в научных целях; предназначенный для выполнения конкретного вида работы; покупать эти программы; преобразовать в машинный код; исполняемый; в отличие от; соответствовать потребностям.  **6**. These are the answers to questions about the text. Ask the questions.  1.    No, it is quite wordy so it is used for commercial purposes.  2.    To support the UNIX operating system.  3.    An applications program.  4.    It is done by the compiler.  5.    It fetches required systems routines and links them to the object module.  6.    No, they are also sold by various vendors.  **7**. Translate the sentences paying attention to the Infinitives.  1. Computers can deal with different kinds of problems. 2. PL71 was developed to combine features of COBOL and ALGOL. 3. C language was developed to support the UNIX operating system. 4. LISP has been used to develop computer-based training packages. 5. The compiler is a systems program which may be written in any language. 6. BASIC was developed in 1965 at Dartmouth College in the United States for use by students who require a simple language to begin programming. 7. A source program cannot be directly processed by the computer until it has been compiled. 8. When a program written in one of these high-level languages is designed to do a specific type of work such as calculate a company's payroll or calculate the stress factor on a roof, it is called an applications program. 9. Institutions either purchase these programs as packages or commission their own programmers to write them to meet the specifications of the users. 10. The primary requirement to be met in software development is to perform the function specified for the software. 11. To calculate a company’s payroll is much easier by using an applications program.  **8**. Find the passages in the text where the following ideas are expressed.  1.    Systems programs control the work of the computer system.  2.    Software packages are not always sold by the manufacturer.  3.    Usually, every high-level instruction translates into many more in machine code.  4.    Systems programs are usually provided by the manufacturer.  5.    Programmers may be required to write software for their employers.  **9**. Choose the correct word to complete each sentence. You may have to change some words slightly.  1. *instruction, instruct, instructed, instructor*  a) Our maths \_\_\_\_\_\_\_ explained to us the principles of binary arithmetic.  b) We were \_\_\_\_\_\_\_ to document our programs very carefully.  c) Both \_\_\_\_\_\_ and data have to be changed to machine code before the computer can operate on them.  2. *compilation, compiler, compile, compiled*  a) Our university computer does not have a PASCAL \_\_\_\_\_\_.  b) Usually, a programmer \_\_\_\_\_\_ his program before he puts in the data.  c) A source program cannot be directly processed by the computer until it has been \_\_\_\_\_\_\_.  3. *result, results, resulting*  a) The linkage editor links systems routines to the object module. The program, referred to as the load module, is directly executable by the computer.  b) The \_\_\_\_\_\_ of these mathematical operations were obtained from the university mainframe and not from my micro.  4.*specification, specify, specific, specified, specifically*  a) Our company bought three packages with very \_\_\_\_\_\_\_ applications: payroll, accounts receivable, and accounts payable.  b) An applications program is designed to do a \_\_\_\_\_\_\_\_ type of work, such as calculating the stress factor of a roof.  c) Did the analyst give the new programmer the \_\_\_\_\_\_\_\_ necessary to start on the project? |

**10.** Read and translate the text.

## Text B. [Programming Language](http://engtopic.ru/english-language/a-programming-language) Theory

Programming language theory (commonly known as PLT) is a branch of computer science that deals with the design, implementation, analysis, characterization, and classification of programming languages and their individual features. It is a multi-disciplinary field, both depending on and, in some cases, affecting mathematics, software engineering, linguistics, and even the cognitive sciences. It is a well-recognized branch of computer science and an active research area, with results published in numerous journals dedicated to PLT, as well as in general computer science and engineering publications.   
A programming language is a machine-readable artificial language designed to express computations that can be performed by a machine, particularly a computer. Programming languages can be used to create programs that specify the behavior of a machine, to express algorithms precisely, or as a mode of human communication.  
Many programming languages have some form of written specification of their syntax and semantics, since computers require precisely defined instructions. Some are defined by a specification document (for example, an ISO Standard), while others have a dominant implementation (such as Perl ).  
The first programming languages predate the modern computer. The 19th century had "programmable" looms and player piano scrolls which implemented what are today recognized as examples of domain-specific programming languages. By the beginning of the twentieth century, punch cards encoded data and directed mechanical processing. In the 1930s and 1940s, the formalisms of Alonzo Church's lambda calculus and Alan Turing's Turing machines provided mathematical abstractions for expressing algorithms; the lambda calculus remains influential in language design.   
A programming language provides a structured mechanism for defining pieces of data, and the operations or transformations that may be carried out automatically on that data. A programmer uses the abstractions presented in the language to represent the concepts involved in a computation. These concepts are represented as a collection of the simplest elements available (called primitives).   
Programming languages differ from most other forms of human expression in that they require a greater degree of precision and completeness. When using a natural language to communicate with other people, human authors and speakers can be ambiguous and make small errors, and still expect their intent to be understood. However, figuratively speaking, computers "do exactly what they are told to do", and cannot "understand" what code the programmer intended to write. The combination of the language definition, a program, and the program's inputs must fully specify the external behavior that occurs when the program is executed, within the domain of control of that program.  
Programs for a computer might be executed in a batch process without human interaction, or a user might type commands in an interactive session of an interpreter. In this case the "commands" are simply programs, whose execution is chained together. When a language is used to give commands to a software application (such as a shell) it is called a scripting language.  
Many languages have been designed from scratch, altered to meet new needs, combined with other languages, and eventually fallen into disuse. Although there have been attempts to design one "universal" computer language that serves all purposes, all of them have failed to be generally accepted as filling this role.

Comprehension check

**11**. Answer the following questions:

What does programming language theory deal with?

Why can it be called a multi-disciplinary field?

What is a programming language?

What are programming languages used for?

What is known about the history of programming languages?

What does a programming language provide?

How are the concepts involved in a computation represented?

How do programming languages differ from other forms of human expression? Why?

Why hasn`t one universal computer language been produced?

**12**. Render the text in English. Use the following words:

The text is about...; first of all...; Secondly/then...; Next...; the definition of...is given; ...is (are) described; according to ...; In conclusion/ To summarize... .

**13**.Study the words given at the bottom and explain their meanings. Read the text.

# Text C.How do computer languages work?

Every program ultimately has to execute as a stream of bytes that are instructions in your computer's *machine language*. But human beings don't deal with machine language very well; doing so has become a rare, black art even among hackers.

Almost all Unix code except a small amount of direct hardware-interface support in the kernel itself is nowadays written in a *high-level language*. (The ‘high-level’ in this term is a historical relic meant to distinguish these from ‘low-level’ *assembler languages*, which are basically thin wrappers around machine code.)

There are several different kinds of high-level languages. In order to talk about these, you'll find it useful to bear in mind that the *source code* of a program (the human-created, editable version) has to go through some kind of translation into machine code that the machine can actually run.

## 1. Compiled languages

The most conventional kind of language is a *compiled language*. Compiled languages get translated into runnable files of binary machine code by a special program called (logically enough) a *compiler*. Once the binary has been generated, you can run it directly without looking at the source code again. (Most software is delivered as compiled binaries made from code you don't see.)

Compiled languages tend to give excellent performance and have the most complete access to the OS, but also to be difficult to program in.

C, the language in which Unix itself is written, is by far the most important of these (with its variant C++). FORTRAN is another compiled language still used among engineers and scientists but years older and much more primitive. In the Unix world no other compiled languages are in mainstream use. Outside it, COBOL is very widely used for financial and business software.

There used to be many other compiler languages, but most of them have either gone extinct or are strictly research tools. If you are a new Unix developer using a compiled language, it is overwhelmingly likely to be C or C++.

## 2. Interpreted languages

An *interpreted language* depends on an interpreter program that reads the source code and translates it on the fly into computations and system calls. The source has to be re-interpreted (and the interpreter present) each time the code is executed.

Interpreted languages tend to be slower than compiled languages, and often have limited access to the underlying operating system and hardware. On the other hand, they tend to be easier to program and more forgiving of coding errors than compiled languages.

Many Unix utilities, including the shell and bc(1) and sed(1) and awk(1), are effectively small interpreted languages. BASICs are usually interpreted. So is Tcl. Historically, the most important interpretive language has been LISP (a major improvement over most of its successors). Today, Unix shells and the Lisp that lives inside the Emacs editor are probably the most important pure interpreted languages.

## 3. P-code languages

Since 1990 a kind of hybrid language that uses both compilation and interpretation has become increasingly important. P-code languages are like compiled languages in that the source is translated to a compact binary form which is what you actually execute, but that form is not machine code. Instead it's *pseudo code* (or *p-code*), which is usually a lot simpler but more powerful than a real machine language. When you run the program, you interpret the p-code.

P-code can run nearly as fast as a compiled binary (p-code interpreters can be made quite simple, small and speedy). But p-code languages can keep the flexibility and power of a good interpreter.

Important p-code languages include Python, Perl, and Java.

*Ultimately, deal with, rare, black art, Unix, kernel, relic, distinguish, compiled language, source code, OS, interpreted language, utility, shell, BC (binary code), SED (smoke-emitting diode), AWK, BASIC, Tcl, LISP, P-code language (pseudo code), Python, Perl.*

**14**.Describe each kind of high-level languages (advantages, disadvantages, examples). Summarizethetext.

Литература.

С.В. Бобылева, Д.Н.Жаткин « Английский язык для сферы информационных технологий и сервиса» 2009г.

Keith Boeckner P. Charles Brown “Oxford English for Computing” Oxford University Press, 2006.

Great English-Russian Computer dictionary, Moscow, 2010.

Additional exercises.

**1**. Summarize the information on different high-level computer languages by completing the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| Language | Developed | Function | Characteristic |
| FORTRAN |  |  |  |
|  | 1959 |  |  |
|  |  | mathematical and scientific purposes |  |
|  |  |  | combines features of COBOL and ALGOL |
| BASIC |  |  |  |
|  |  | to support UNIX operating system |  |
|  | 1962 |  |  |

**2**. Before reading the text try to fill in the gaps in these sentences. If you can’t, do it after reading the text.

1.    A \_\_\_\_\_\_\_\_\_\_\_\_ is a program written in one of the high-level languages.

2.    A program written in a high-level language must be interpreted into \_\_\_\_\_\_\_\_ before the computer will read and process it.

3.    A program designed to perform a specific task is called an \_\_\_\_\_\_\_\_\_\_\_.

4.    The \_\_\_\_\_\_\_\_\_\_ or \_\_\_\_\_\_\_\_\_\_ is the program produced when the original program has been converted into machine code.

5.    A \_\_\_\_\_\_\_\_\_\_\_is a program that converts a high-level language into machine code.

6.    The systems program which fetches required systems routines and links them to the object module is known as the \_\_\_\_\_\_\_\_\_\_\_\_\_\_.

7.    The \_\_\_\_\_\_\_\_\_\_\_\_\_ is the program directly executable by the computer.

**3**. Answer the following questions.

1.  How do third-generation languages differ from first- and second-generation languages?

2.  What is a natural language?

3.  What are the main characteristics of a machine language?

4.  What are the reasons behind the development of high-level programming languages?

**4**. Choose the correct answer.

1. Which of the following generations of languages will likely include the languages of the future?

a) first-generation

b) second-generation

c) third-generation

d) fourth-generation

e) fifth-generation

2. Which of the following generations does natural language fall into?

a) first-generation

b) second-generation

c) third-generation

d) fourth-generation

e) fifth-generation

3. Which of the following relates to machine language?

a) difficult to learn

b) first-generation language

c) machine-dependent

d) instructions and data are represented by binary digits

e) all of the above

4. The language that does not require the user to learn a specific vocabulary or syntax is:

a) FORTRAN

b) COBOL

c) BASIC

d) none of the above