Lecture №1.11. Programming languages and programming systems.

A programming language is an artificial language designed to communicate instructions to a machine, particularly a COMPUTER. Programming languages can be used to create programs that control the behavior of a machine and/or to express algorithms precisely.

Every COMPUTER has the own programming language is a language of commands or absolute language and can carry out the programs, written only in this language. In an absolute language a certain operation which a machine can execute corresponds every command. However in absolute language, programing is difficult from the excessive working out in detail of the program. Therefore already on COMPUTER of the first and second generation for the increase of the labour of

programmers productivity began to apply languages programming, not consoling with absolute languages. On COMPUTER of the third generation an absolute language practically is not used for programming of tasks, the role of internal language of COMPUTER was saved only after him.

Several hundred different languages are presently counted programming which are classified on different signs. Most general is classification on the degree of dependence of language from COMPUTER. On this sign languages are divided by two large groups:

- Computer-dependent languages,
- Computer-independent languages.

Computer-dependent languages, in turn, divide by machine and machine-oriented.

An absolute language is a programming language, directly perceived by a computer. Every command of absolute language is interpreted by an apparatus, executing the indicated functions. Commands of absolute language in principle are enough primitive. The only corresponding association of these commands in the programs in absolute language enables to describe serious enough algorithms. The sets of commands of absolute language of modern computers include some very effective possibilities frequently.

It is said that an absolute language is computer-dependent: the program, written in absolute language of computer of one type, as a rule, can not be executed on the computer of other type, if its absolute language is not identical to the absolute language of the first computer (or is not expansion in relation to this language). By another sign machine, or by a vehicle, there is character of commands dependence: in the commands of

absolute language the names of concrete registers of computer are specified and processing of data is foreseen in a that physical form in which they exist in this computer. Most first computers programed directly in absolute language, and presently the very small number of the programs is written in absolute language only.

Computer-oriented languages sometimes named ***usercodes***. Distinguish two levels of computer-oriented languages. The languages of the symbol encoding behave to the first level, otherwise called ***mnemocodes***, and to the second are ***macro languages***.

A ***mnemocode*** differs from the absolute language of corresponding COMPUTER replacement of digital kodes of operations alphabetic (mnemonic), and digital addresses of operands - alphabetic or alphanumeric. During translating into the language of COMPUTER every command of mnemocode is replaced by the corresponding command of absolute language (<< one in one >>).

Application of mnemocode allows to automatize work of programmer on storage allocation, more precisely, on appropriating of veritable addresses. It is special useful at programming for machines with the variable format of commands. In addition, a mnemocode substantially facilitates work on drafting of the large programs, when separate program segments (modules) are made different programmers and unite in the single program on the stage of loading.

Language the second level is ***a macro language*** - along with the symbolic analogs of computer instructions, which a mnemocode consists of, assumes the use of macro instructions, not having direct analogs in an absolute language also. At translation every macro instruction is replaced by the group of commands of absolute language (<< one in a few >>). Application of macro instructions abbreviates the program, promotes the productivity of programmer. Programmer, using a computer-oriented language must be well acquainted with the features of device of machine which the program is made for.

Computer-independent languages are also divided by two groups on the degree of working out in detail of the program. Procedure-oriented languages behave to the first group, and problem-oriented languages behave to the second group.

The procedure-oriented languages are intended for description of algorithms (procedures) of decision of tasks, therefore they are also named algorithmic, although the concept of algorithmic language does not coincide with the concept of programming language. If a record in algorithmic language is un mediocre, suitable for an input in COMPUTER and transformations to the prepared executable code, then such language is simultaneously a programming language. Some algorithmic languages, strictly speaking, are not programming languages, if not to add the special tools to them. In particular, the algorithmic language Algol-60 becomes a language programming after plugging in him operators of input and output and specification of methods of implementation of some other operations of management an equipment COMPUTER.

The program in procedure-oriented language does not almost depend on concrete COMPUTER which a task will decide on. A word "almost" it is necessary to understand in that sense, that in most cases the programs of decision of the same task for different COMPUTERS differ in only some unfundamental details of external registration, which in transition from COMPUTER to COMPUTER replaced mechanically.

A structure of the procedure-oriented languages is nearer to the human language, for example Russian or English, what to the language of COMPUTER. Therefore translating from the procedure-oriented language into an absolute language is carried out on principle "a few in a few". In other words, in most cases it is here possible to set accordance only between the group of elementary language constructs and group of commands COMPUTER, like during translating from English into Russian of group of words or even group of suggestions replace the group of words in other language. Word-per-word translation here is not possible.

Appearance of new economic feasibilities set the problem before system managers - to create programmatic tools, providing the operative co-operating of man with COMPUTER they were named dialog languages.

These works were conducted in two directions. The special managing languages were created for providing of the operative affecting passing of tasks which was made on any early unelaborate (not dialog) languages. Languages which except for the aims of management would provide description of algorithms of decision of tasks were developed also.

The necessity of providing of the operative co-operating with an user demanded maintenance in memory of COMPUTER of copy of the initial program even after the receipt of object code in machine codes. At making alteration in the program with the use of dialog language the system of programming by means of the special tables sets intercommunication of structures initial and objective programs. It allows to carry out the required editorial changes in an object code.

To the problem-oriented languages take the so-called unprocedural languages. Such languages which do not require the detailed record of algorithm of decision of task. An user must only specify problem definition or name the sequence of tasks from a before geared-up set, to specify basic data and required form of delivery of results. This information is used by the special program - generator for generating of executable code. With expansion of application of the computing engineering domains there was a necessity to formalize the presentation of raising and decision of new classes of tasks. It is necessary it was to create such languages programming, which, using denotations and terminology in this area, would allow to describe the required algorithms of decision for the put tasks, they were become problem are the oriented languages. These languages, languages are decision-oriented certain problems, must provide a programmer tools, allowing shortly and clearly to formulate a task and get results in the required form.

In relation to a translator the languages all mentioned higher, except for absolute languages, are an input. In the process of translation the program in input language is translated into some internal language, more comfortable for further work of translator, and then consistently there are a few stages of treatment. On every stage the translated program appears in some intermediate language. And, finally, after treatment a translator the program turns out in object language.

Comparative description of languages. Computer-oriented languages are universal in the same degree the language of machine is universal in which, as in them contained tools of programming and decision on COMPUTER of any tasks, with which COMPUTERS can manage on the economic feasibilities. At programming on these languages it is possible to take into account the features of set of instructions and device of COMPUTER, that allows to create the high-quality programs. However computer-oriented languages are enough difficult for a study, and programing on them is difficult.

Computer-independent languages are effective only for the certain class of tasks. Out of this class of tasks application of most high-level languages ineffectively and in general uselessly. These languages comparatively easily to study. Programming on them is considerably simpler, than on computer-oriented languages.

The procedure-oriented languages are programming languages, where possibility of description of the program is as to the aggregate of procedures (subprograms);

High-level languages are either procedural-oriented or problem-oriented. Procedure-oriented languages high level are universal languages programming which can be used for the decision of the most various tasks. The problem-oriented languages target specially at the decision of tasks of concrete types. Such languages, as Pascal, Cobol, Fortran, Basic are usually considered procedural-oriented, and such languages, as GPSS (simulation language) and SPSS (language for implementation of statistical calculations), - problem-oriented.)

Translators. Any program which translates arbitrary text in some input language in text in other language is named a ***translator***. In particular, a source code can be the input program. A translator transfers it in an object or objective routine.

It is possible to consider in sense of this determination the simplest translator, loader, which transfers the program in conditional addresses, executed as a module of loading, in an object code in absolute addresses. In this case an input language (language of loader) and objective language (COMPUTER language) are the languages of one level. However more frequent input and objective languages behave to the different levels. The level of input language is higher than level of objective language usually.

On the level of input language translators it is accepted to divide by assemblers, macro assemblers, compilers, generators.

The input language of assembler is a mnemocode, macro assembler is a macro language, compiler is the procedure-oriented language, and generator - problem is the oriented language. In this connection an input language is named on the type of translator: assembly language, language of macro assembler et cetera.

The program, got after treatment a translator, is either directly carried out on COMPUTER or exposed to treatment other translator.

Compilers and interpreters. Usually the processes of translation and execution of the program are divided in time. All program is translated at first, and then carried out. Translators, working in such mode, name the translators of compiling type. If the input language of such translator is the procedure-oriented language high level, then a translator is named a ***compiler***.

There are translators in which translation and execution is combined in time, they are named interpreters. The block of analysis, recognizing operators of input language, set of subprograms, corresponding to the different operators, and block, managing all work of interpreter, enters in the complement of ***interpreter***.

On pointing of control block, the block of analysis looks over source statements, recognizes their type and possibility of direct execution determines. Information about possibility of implementation of operator is passed to the control block, which causes corresponding subprogram, carrying out actions, prescribed by an operator.

In such chart a compiler can be done by very simple. An interpreter is some simpler than compiler, as direct execution of the recognized operators of input language does unnecessary actions, related to arrangement of object code, by registration of it in the single module of loading or as a few modules, if it is great.

The lack of interpreter consists in the ineffective use of machine time. For example, at implementation of the cyclic programs, the same operator it is necessary to interpret repeatedly. At the repeated implementation of the program, interpretation has to be executed again, while the translator of compiling type allows to execute translation one time, and then to keep the program in machine codes. On the indicated reason interpreters are used relatively rarely.

Assemblers and macroprocessors

The programming direct requires very much the time and fraught by errors. The languages of assembler type, allowing to promote speed of process of programming and decrease the amount of errors of encoding, were therefore worked out. Instead of numbers, used for writing of the programs on absolute languages, rich in content mnemonic reductions and words of human language are used in the languages of assembler type. However computers can directly perceive the program in assembly language, therefore it must be in the

beginning translated into an absolute language. Such translation is carried out through a program-translator, called an assembler.

Languages of assembler type also are computer-dependent. Their commands straight and simply correspond to program instructions absolute. To accelerate the process of encoding of the program in assembly language, were worked out and plugged in assemblers the so-called ***macroprocessors***. A programmer writes macro instruction as pointing of necessity to execute an action, described by a few commands in assembly language. When a macroprocessor during program translation reads macro instruction, he makes macro expansion - i.e. generates the row of commands of assembly language, corresponding to this macro instruction, Thus, the process of programming is considerably accelerated, as a programmer has to write the less number of commands for determination of the same algorithm.

Thank you for your attention