BASIC PRINCIPLES OF CONSTRUCTION OF INTERPRETER SYSTEMS FOR PROBLEM-ORIENTED LANGUAGES IN SCIENCE AND ENGINEERING

V. P. Klimenko

We describe the main properties of problem-oriented programming languages for scientific and engineering applications and the principles of hardware interpretation of these languages in small computers.

1. The aim of this study is to generalize the experience with construction, implementation, and use of language tools in computer systems intended for solving scientific and engineering problems. These problems, as a rule, involve calculations and prediction of the behavior of technical objects and processes in design stages and also analysis of experimental data. The critical nature and complexity of these problems place special requirements on their solution and emphasize the importance of programming deadlines and programmer productivity.

The performance of programming systems for engineering applications is determined by the following factors:

1) the elapsed time for solving the problem;
2) the time spent by the user solving the problem;
3) the number of job submissions to the computer in order to debug the program;
4) the machine time spent solving the problem.

High efficiency of programming systems can be attained by increasing the level of machine intelligence through development of interactive tools, higher computing precision, and the use of the apparatus of transformation of analytical models. The inclusion of the apparatus of analytical transformations in the language makes it possible to derive general solutions dependent on symbolic parameters for subsequent qualitative analysis and exact transformations (differentiation, integration, etc.).

The algorithmic languages of the ANALITIK family designed for generating such solutions [1-9] were hardware-implemented in some small Soviet computers, such as MIR-1, MIR-2, MIR-3, SM-1410, and ES-2680 (similar American systems MACSYMA, REDUCE, and FORMAC were software-implemented on mainframe computers). Further development of hardware- and software-implementations of different versions of the ANALITIK language is currently under way at the V. M. Glushkov Institute of Cybernetics of the Ukrainian Academy of Sciences.

The description language for mathematical models in engineering and scientific applications is the language of mathematical analysis. Subsequent transformation of the mathematical models into adequate numerical models and selection of efficient numerical methods are very time-consuming operations requiring highly qualified programmers. Therefore, an important property of the language should be its ability to describe numerical-analytical problem-solving methods. Automated description requires an advanced apparatus for recognizing the functional properties of analytical expressions. Moreover, because of recognition and formalization difficulties, highly developed interactive systems are needed for the formulation of scientific and engineering problems.

Analysis of problems and solution methodologies suggests the following functional features of software systems:

1) the source language of the programming system should be similar to the language of the mathematical disciplines invoked for the description of the mathematical models;
2) the language implementation system should ensure open access to the program both during its development and in execution time;
3) the operating system and the programming system should support special interactive tools designed to simplify the debugging process.
2. Efficient problem orientation is achieved by observing the following principles of source language construction [10-12].

**Similarity of the Source Language to the Language of the Relevant Application Domain**

For engineering problems, the application language is the language of mathematical analysis. This principle has been implemented in the family of languages for the MIR family of computers:
- the alphabet includes conventional symbols of mathematical operations and operators (+, -, *, /, f, d, etc.);
- natural-language words are used as keywords (COMPUTE, IF, LENGTH, END, etc.);
- the syntax used for the description of objects and constructs in the language is highly similar to the description of corresponding constructs in the language of mathematical analysis (for instance, \(\sum_{i=1}^{N} a_i\) is written as \(\Sigma(I = 1, N, A[I])\), etc.).

**Choice of Mathematical Expression as the Basic Information Object of the Language Family**

The expression is used in all advanced algorithmic languages as the basic language construct defining the algorithm that computes a numerical value. In the ANALITIK language family, expressions play a dual role: on the one hand, an expression is an algorithm to compute a value (not necessarily numerical), and on the other hand, it is an information object (a data element) which can be subjected to transformations by the transformation system defined in the languages.

The ANALITIK language essentially differs from the traditional programming languages by the variety of numbers that it allows. These include:
- decimal numbers with arbitrary mantissa dynamically defined or redefined in the program, without any restriction of the range (the order) of representation of the number;
- integers without length restriction;
- rational fractions \(a/b\), where \(a\) is an integer and \(b\) an unsigned integer.

Data aggregates in ANALITIK, in addition to traditional structures (such as an array of arbitrary dimension), include sets of expressions with hierarchical functional dependences. This data structure plays a central role in the ANALITIK languages.

**Inclusion of a Recognition System for the Properties of Information Objects**

Human intellectual activity involves manipulating a complex hierarchical system of concepts and classes reflecting various levels of abstraction. This system is supported by corresponding natural-language constructs. One of the determining functions of intelligence is the capability of recognition or classification.

The level of "artificial" mathematical intelligence of languages for engineering applications is also determined by the efficiency of the recognition system for the properties of information objects. A recognition system with the following components has been developed for the ANALITIK language family [10].

**Recognition of the Properties of Numbers.** This recognition is exhausted by the two-place predicates testing the relation of a pair of numbers:

\[ a = b, \quad a \neq b, \quad a \geq b, \quad a \leq b, \quad a > b, \quad a < b. \]

**Recognition of Equivalence of Expressions.** The relation of equivalence of two expressions is established if both expressions can be reduced to coinciding expressions by identical transformations. For the realization of the equivalence relation in the ANALITIK languages, preliminary goal-directed simplification of the expressions is performed by applying groups of identical transformations. In this way, the number of identical transformations is reduced and the resolution of the equivalence recognizer is increased. Several such groups are formed according to the frequency of joint use of the transformations. Assume that the application of each group of identical transformations produces an expression in corresponding canonical form. If the expressions \(Q_1\) and \(Q_2\) belong to some subalgebra \(\Omega\) in which the canonical form \(F\) is defined, and \(F(Q_1)\) and \(F(Q_2)\) completely coincide, then \(Q_1\) and \(Q_2\) are equivalent.

The following groups of transformations and corresponding canonical forms are defined in the ANALITIK languages.

**Group I** (first canonical form). This includes a) transformations that involve arithmetic operations on numbers, b) transformations that allow for the properties of 0 and 1, i.e., \(P + 0 = P, P \cdot 0 = 0, P \cdot 1 = P\), etc., c) transformations that use identities associated with superpositions of transcendental functions.