Joint NCC & IIS Bull., Comp. Science, 11 (1999), 39–44

Model of teaching and program-language complex in educational informatics

S. S. Kobilov

Introduction

The history of informatics in the leading countries as well as our twelve-year experience shows that the initial stage of the computerization era is practically connected with the level of the computer competence [1]. Educational informatics provides elementary computer knowledge. This course should be based on a special concept (model) of teaching.

Educational informatics can be considered as an elementary subject (such as mathematics, physics, biology), but it has two peculiarities which should be taken into consideration:

– methodological and technological base and methods of teaching are being rapidly changed;

– it needs a permanent support by special technical, language and programming means. So, the chosen model of teaching and the implementation of its program-language support should take into account these peculiarities.

We present one of the educational informatics models and introduce the ways of developing the program-language complex (PLC) aimed to support this model. The paper consists of two parts and a conclusion. The first part deals with the description of a model of teaching informatics. The peculiarities of the development of PLC components are introduced in the second part.

1. A model of teaching informatics

Different methods are used at present in the conceptual development of teaching informatics. We present a model of teaching informatics with a general strategy consisting in the following five principles:

1. **Subject-tool informatics usage**. Informatics is studied using computer means, and informatics technology is used to support the general ways of educational activities [1];

2. **Learner's skills**. The main accent is on teaching the algorithmic and programming elements;

3. **Knowledge of the basic algorithmical structures and typical programming methods.** The inner relation between algorithmization and programming is demonstrated and a "smooth" transition from one level to another is shown;

4. **The usage of the native language.** Teaching is aimed at the learner's native language with the use of program-language means and technological methods developed on the base of the national interface;

5. **Multinational tools.** The language, program and technological means are developed with the account of their adaptation to various lexicons.

The functional purpose of such a method and PLC development is to provide a model of teaching informatics for senior pupils and junior students.

2. The peculiarities of developing the program-language complex

Let us consider the structure and peculiarities of developing the PLC. This complex should satisfy the needs of the chosen model of teaching. That is why its structure includes the language tools, programming systems and hardware support.

2.1. Language tools

They consist of a special language used for the description of algorithms and a localized programming language of a higher level.

Algorithmic language. To our needs, we have chosen a special algorithmic language (SAL) [2]. This language is close to the natural language, its algorithms can be written and read as usual texts and, which is more, the study of this language will help one to get profound knowledge of any programming language in the future. Another important aspect of SAL is that its structure is close to the algorithmic mentality of the learner since it has no transition (go to) statement, which satisfies the requirements of structural program construction, and there are no details connected with a computer device.

For the sake of usage convenience and realization needs, some changes and additions were made to SAL, such as introduction of two new commands (input, and output) which are used for intermediate data input and output; the specification of dynamic tables; the usage of key words without underlining; and a linear notation of expressions, i.e. operation symbols are placed between the operands and function arguments are located in parenthesis.

Programming language (PL). It is essential to choose a PL for studying the programming elements. We have chosen Pascal/P [3] with a basic

Russian notation of Standard Pascal [4]. Our choice is defined by the following. Pascal was developed as a language of teaching. Its constructions are close to those of SAL and its implementation on the basis of modern educational informatics hardware is rather simple. The Pascal family is being intensively developed and other language families, such as Modula, Oberon, and Delphi are introduced. In general, we can state two main factors influenced our choice.

The first one is the choice of the necessary minimum of language constructions, traditional for many modern languages and sufficient for the initial study of algorithms and programming. Therefore some data types as well as statements and special Pascal functions have been excluded. The label, data type set, variant record, goto statement, cycle statements of the type for $\cdots$ downto and repeat $\cdots$ until have been reduced and the procedures get(f) and put(f) of processing a file f have not been used. For practical needs, data types string and string[n] have been included into the language, as well as two new procedures of file processing: Close(var f) closing the file and Assign(var$f$; FileName: string) setting the correspondence between the logical file f and the physical file of the operating system (OS).

The second factor is connected with the national lexicon of the language facilities. That is why they were related to other national languages (Russian, Uzbek, Tajik) based on Cyrillic and Latin graphics. Thus, algorithmics and programming should be studied on the basis of the learner's native language, and input languages of the systems should have modifiable lexical structure.

2.2. Programming Systems

The programming complex consists of a dedicated system on the basis of SAL and a programming system (PS) with the localized input language Pascal. The first one is used for computer support of the algorithmic course, while PS is aimed at the study of the main programming principles. Each of these systems is developed as an integrated environment with the conventional components. The environment was initially produced to support a definite style of constructing and debugging of algorithms (programs). Its components are the working window with the main menu, editor, compiler, help subsystem and the data base (DB). Let us briefly discuss the peculiarities of the teaching environment and its components.

The Program Constructing and Debugging Style. The environment is aimed at the style of structural construction on the basis of the structural edition of algorithms. It is also supplied with means for interactive debugging and program running [5]. We believe that the use of the structural construction in educational informatics gives the following advantages:

a) This type of construction is close to human operational mentality. It gives the possibility of the most adequate description of typical processes in the application area of a task with the help of the corresponding integral PL constructions;

b) It combines the "strict" requirements to the structural program editing with the usual methods. The first of them puts limitations on the user actions and it is useful in training the beginners;

c) The program becomes an active object at the very beginning and within the process of its construction. Even not being completed, it is partially ready for use. This provides checking the properties of program running, which, step by step, proves the programmer's choice and action correctness.

Another important mechanism of training is visualization of program debugging and running. To display the process of program running, the output facilities should be realized in the system so that the text of the original program could be presented on display in details or shown in a flowchart with underlined and coloured constructions and keywords indicated at the control points.

Working Panel and Main Menu Console. The interactive mode is the base of training systems. It provides a feedback to the learner. At present a lot of standards for the user interface construction are developed. While designing the present complex, we use the main elements of the System Application Architecture (SAA) standard, in particular, a part of this standard — Common User Access (CUA) — is used to construct the user interface. So, when the system starts, the working window appears on display with the main menu at the top. The menu contains the follwing options: File, Editing, Compilation, Lexicon, and each of them can have vertical sub-options. At the bottom of the window, there is a prompting string showing the correspondence between functional keys and operations. Using the option Lexicon [suboption], a learner can choose a notation for writing algorithms on the basis of SAL (Pascal programs) and a lexicon of communication with the system. Further work is being fulfilled in this language environment.

Editor. The embedded system editor has two operation modes, textual and structural. The structural one is main, since in the process of studying PL we pay much attention to its syntactical constructions and rules. The editor supports both conventional and specialized operations with texts in the structural mode, such as a call of language constructions kept in the form of "ready-made" patterns; the transformation of the source program text into an abstract syntactic tree (AST) and vice versa; the recognition of elementary errors and display of error messages; the creation and modification of patterns.

At first the patterns (general structures of algorithms in SAL and Pascal programs, program items, commands, statements, auxiliary algorithms and subroutines) are kept in the data base and called when needed and, after the processing, inserted in a definite place of the abstract tree.

For example, the construction if < condition > then < **statement1** > else < **statement2** > is entirely displayed while the parts in brackets are easily deleted and replaced by real constructions.

Thus, the principles of structural editing open the possibility of program checking at each step of the process of its development, being useful for the learner and preventing him from making errors.

Compiler. Specialized system compilers on the basis of SAL and PS with Pascal as an input language are developed in the form of an interpreter.

This method is used to simplify their implementation and has the following peculiarities. Interpreter allows for easy management of the design process, debugging and visualization of program running [6]. Using the interactive mode, we can write, check and run programs during a single session of interpreter. The errors can be easily corrected. We need not go back to editing environment and then compile the program once more. Structural construction and edition [7] provide the intermediate representation and interpretation of incomplete programs. Programming systems are intended for educational purpose and they are not aimed at solving the tasks which require high operation speed.

Help Subsystem and Data Base System. DB is developed on the basis of the electronic textbook concept. It is supposed to keep the structured information: theoretical, practical and methodological materials (glossary containing the basic notions and terms of input language, a set of typical schemes, demonstration algorithms and programs, as well as a set of tests aimed to check and evaluate the learner's skills).

The data base (DB) is organized and kept in the form a hypertext. The data base files contain texts of the main and intermediate program representations, educational materials and auxiliary information connected with a definite lexicon. The support of constructions and functions of the help subsystem and data base components is implemented by a separate tool system. It contains a set of special operations dealing with the hypertext, as well as with input, processing and output of information in the DB system.

2.3. Hardware Support

Before using a computer in the educational process on the multi-national basis, it must be "taught" the national languages. It is necessary to develop and use special programs — devices-drivers — to provide operation on different national alphabets.

Drivers enhance the possibilities of a definite part of OS and hardware and can be considered as a hardware support while designing the program complexes on the basis of changing lexicons. Therefore, the implementation of a specialized subsystem which gives us the opportunity of creating and using the letters of different national alphabets on PC has been specified in the environment. The subsystem contains the following tools: font designer; ranker used for the distribution of the letters on the keyboard and in the PC code scheme. Besides, there is a managing program for choosing the necessary drivers and creating a working configuration for a definite alphabet.

The PLC is developed for IBM-compatible computers running under Windows environment.

Conclusion

An educatuonal model for studying PC is presented in this paper. The questions of the development of the program-language complex intended for support of this model are discussed. The general strategy of teaching is founded on algorithmic and programming elements based on the native language. The structure of the complex and requirements to its components, including the language and program facilities and hardware support, have been defined.

References

[1] A. P. Ershov, *Computerization of Schools and Mathematical Education*, Proc. of the Sixth Intern. Congress on Mathematical Education, Budapest, 1988, 49–65

[2] A. P. Ershov, *Basic Concepts of Algorithms and Programming to Be Taught in a School Course in Informatics*, Proc. of the Intern. Joint Conf. on Theory and Practice of Software Development (Tapsoft), Berlin, 1985, 14 p

[3] S. S. Kobilov, *Programming system Pascal/R*, Important Problems of Applied Mathematics and Economics, Samarkand: SSU, 1997, 73–79 (in Russian)

[4] K. Jensen, N. Wirth, *Pascal. User Guide and Report*, Springer-Verlag, 1978

[5] T. B. Boltaev, T. V. Kuzminov, I. V. Pottosin *On Structured Construction and Supporting Tools*, Programming Environments: Methods and Tools, Novosibirsk, 1992, 22–37 (in Russian)

[6] D. Gries *Compiler Construction for Digital Computers*, John Wiley&Sons, Inc., No. 4, 1971

[7] T. B. Boltaev *Interpreter of Incomplete Programs*, Programming Environments: Methods and Tools, Novosibirsk, 1992, 38–50 (in Russian)