Methods and tools for teaching programming at university

Victor N. Kasyanov, Elena V. Kasyanova
A.P. Ershov Institute of Informatics Systems SB RAS, Novosibirsk, 630090, Russia
kev@iis.nsk.su

Abstract. The paper is devoted to the methods and tools for teaching programming developed in the A.P. Ershov Institute of Informatics Systems. Practical work on programming and an introductory course of programming on the basis of the Zonnon language are considered. The paper also describes the development of an adaptive system for distance learning and a visual cloud system that supports parallel and functional programming.

1. Introduction

The rapid development of information technologies and their penetration in all aspects of society and all spheres of production activity leads to the fact that, in order to become successful in his future activities, a university graduate should not only master the existing user technologies and gain skills in finding ready-made solutions, but also learn solve new problems with programming. However, mastering the programming skills is still a very difficult problem for many students.

Currently, distance learning systems are actively investigated and developed. The benefits of online learning are clear: classroom and platform independence. Networked training software, once installed and maintained in one place, can be used at any time and around the world by thousands of students with any computer connected to the Internet. Thousands of online training programs and other educational applications have become available on the Internet in recent years. The problem is that most of them are nothing more than static hypertext pages and poorly support the problematic approach to learning. Adaptive hypermedia systems that have appeared recently have significantly increased the capabilities of learning systems [1], [2]. The purpose of these systems is the personalization of the hypermedia system, its adjustment to the characteristics of individual users.

Cloud services, which provide various opportunities, including those for training, are gaining more and more popularity. Cloud hosting services such as Amazon and Cloud9 provide computing resources without requiring additional installation of runtime support libraries and allowing program code to be executed inside the browser, and it is safe for the user. The growing number of services on the Internet that provide various services directly or indirectly related to computing indicates the popularity of this approach. It is very attractive, without having a compiler for a particular programming language, to be able at any time to go to the corresponding page on the network and execute the code of interest in that language. Such services range from completely ascetic, designed for extremely short programs, to providing a rich development environment with grouping by projects, syntax highlighting, etc.

The purpose of the research presented in this paper is to develop the methods and tools for teaching programming at a university that make this process more individual, accessible and effective. Practical work on programming and an introductory course of programming on the basis of the Zonnon
language are considered. Research on the development of an adaptive system WAPE for distance learning and a visual cloud system CPPS for supporting parallel and functional programming is described.

2. Practical work on programming
When teaching programming, the most important, in our opinion, is the initial stage at which a student masters his/her skills of designing the algorithms in a high-level language, which cannot be done by reading only a few manuals or listening to a course of lectures in programming. The practice of algorithm constructing is necessary, and it is impossible without an appropriate set of examples and tasks.

Therefore, it is not a coincidence that for many years the first-year students of the Department of Mechanics and Mathematics of Novosibirsk State University (NSU) have practical studies on programming. It is considered as an important part of the main programming course and currently it focuses on the use of C / C++ languages and Visual Studio system. This does not mean, of course, that other implementations of C / C++ or other programming languages are not suitable for solving these individual tasks by students.

The main objective of practical studies is not only and not so much teaching students the actual recording (coding) of a well-known algorithm in a high-level programming language, but practical consolidation of the knowledge gained in the programming course and mastery of general methods, techniques and skills of solving problems on a computer. Each individual task performed by a student is a combinatorial or logical task with a short and clear wording that does not contain a description of the algorithm. Thematic tasks are divided into five sections (graphs and road systems; grammars, languages and automata; formulas and programs; geometry; games and models) and labeled as having regular, high or low complexity. During the studies, each student solves five individual problems, one from each section. The tasks are rated using the following three metrics: model complexity, algorithm complexity, and “ingenuity”. The metric “ingenuity” is based on the properties of the task, such as complexity of understanding application area from which the task is drawn, complexity of designing the model and the algorithm for the task, as well as complexity of interconnections between control and data structures which should be represented by the program.

Each individual task includes the following: (1) analysis of the problem statement and development of an approach to its solution; (2) step-by-step development (based on the chosen approach) of the solution algorithm and its description; (3) justification of the algorithm; (4) selection and justification of the representation for input, output and intermediate data; (5) coding of the algorithm, that is, its writing in the C language; (6) selection and justification of the set of tests for the program; (7) program debugging and demonstration of its correct running on the selected set of tests.

Practical recommendations for the implementation of all these steps are given in the manual and make up most of it [7].

3. Zonnon-based introductory programming course
Zonnon is a new universal programming language in the Pascal, Modula-2 and Oberon family of languages, work on which is carried out at the Zürich Institute of Informatics [4]. Since the Zonnon language is conceived as a further evolution of the Oberon language, the authors strive to preserve important features of the Oberon language and its successors, such as the compactness of the language, clarity, unambiguity and orthogonality of its basic concepts. However, in order to create a modern alternative to the Oberon language, the authors made a number of changes to the language, the main of which are: a more developed modular structure of the language; an advanced and at the same time simple and clear object model; the concept of active objects.

We have developed for MSDN Academic Alliance an introductory course [12], which is designed to teach the basic methods of constructing correct, efficient and reliable programs based on the Zonnon language and the Microsoft.NET platform. The course is intended for a wide range of people learning programming methods, primarily the students of NSU, other universities and secondary
schools, as well as schoolchildren who want to deepen their programming knowledge. In particular, it can be useful for those Russian secondary schools that currently use Pascal as the language of primary instruction in programming and have a desire to move on to a more modern programming course, covering the concepts of new generation programming languages such as Java and C#, but to make this transition smoothly, without a sharp change in the prevailing style of teaching programming.

The course is based on the principle of concentric presentation of the material. It is assumed that, from the first lessons, students begin to practice creating programs that can actually be run on accessible computers and gradually master the skills of developing linear, branching and iterative algorithms, algorithms with arrays and procedures in the Zonnon language. Also, gradually, along with the expansion of the class of tasks to be solved, students deepen their knowledge of the Zonnon language. Other basic methodological and technological principles on which the developed course is based are as follows.

1. The principle of training in the construction of programs on detailed clarified samples of the solutions of carefully selected problems. The purpose of the examples is not only to give samples and describe the basic schemes of the algorithms, but also, on a comparative analysis of different solutions of the same problem, to introduce the concepts such as efficiency, self-documenting and reliability of the program solution to students.

2. The principle of evidence-based programming, when a program is built along with a proof of its correctness. To do this, the concepts of intermediate statements and invariants are introduced in the course, and such statements are written in the form of program comments in the developed problem solving algorithms.

3. The principle of step-by-step program development, when a program is built from the formal specification of a task using small formally verifiable conversion steps.

4. The principle of modular programming, which allows you to design, develop and assemble a program in parts using libraries of ready-made parts.

5. The principle of object-oriented programming, allowing program developers to easily create more and more complex applications using encapsulation, inheritance, and polymorphism.

4. The WAPE system for distance learning

The class of adaptive hypermedia systems consists of all such hypertext and hypermedia systems that reflect some features of the user in his model and use this model to adapt various aspects of the system that are visible to the user [1], [2]. Thus, each user has his own interface and individual navigation capabilities for working with an adaptive hypermedia system.

Adaptability has become particularly important for distance learning Web systems since trainees began to learn mostly on their own (usually from home). The intellectual and personal assistance that a teacher or fellow student can give with conventional (classroom) training is not easily achievable with distance learning. Adaptability is important for distance learning software also because it should be used by a much more diverse set of students than any single-user educational application. Network software designed for a class of users (with a certain mindset and with one level of knowledge) may not be suitable for other trainees at all. The following user characteristics of the training system that are important for its adaptation are distinguished: the goal (or task) of the user, his level of knowledge, his level of training, the user's experience with this hypermedia system, the set of user preferences, personal characteristics of the user, and characteristics of the user environment. Adaptation in adaptive hypermedia may consist in adjusting the content of the next page (adaptation at the content level) or in changing links from the current page, index pages and map pages (adaptation of navigation).

We have developed a project of a system WAPE [6] that supports active individual distance learning of programming via solving problems and combines some capabilities of adaptive hypermedia systems and intelligent learning systems. The system supports not only students but also instructors, lecturers, and administrators. All of them access the system through a standard Web browser, but they differ in their capabilities. The system supports the following three levels of the learning process: when a student is studying the theoretical material, when the system is testing the
knowledge of a student and when a student is solving a set of individual problems (tasks and exercises).

As a rule, adaptive systems track the user's movement on a hyperbook for updating his/her knowledge model. The main disadvantage of this approach is the difficulty in measuring the knowledge acquired by the user while reading a Web page. Instead, in the WAPE system we use only the students' successes and failures shown by them in solving tests, tasks and exercises to update their knowledge models. Another important difference between our approach and the usual one is that we are not trying to evaluate the student’s success in studying the course based on the level of knowledge in his/her model. In the WAPE system, achieving a certain level of knowledge by a student does not allow him/her to complete the study of the course with a certain grade, but only is used for adaptive navigation to give the student the opportunity to start solving a particular problem from his/her individual set of problems.

5. The cloud parallel programming system

Parallel computing is one of the main paradigms of modern programming, but the existing curricula of most universities do not properly address the major transition from single-core to multi-core systems and from sequential to parallel programming. They focus on application program interface (API) libraries and open multiprocessing (OpenMP), message passing interface (MPI), and compute unified device architecture (CUDA) / graphics processing unit (GPU) techniques. This approach misses the goal of developing the students' long-term ability to solve real-life problems by “thinking in parallel”.

Functional programming is a programming paradigm, which is entirely different from the conventional model: a functional program can be recursively defined as a composition of functions where each function can itself be another composition of functions or a primitive operator (such as arithmetic operators, etc.). Using traditional languages and methods, it is very difficult to develop high-quality, portable software for parallel computers. In particular, parallel software cannot be developed with low costs on serial computers and then transferred to parallel computing systems without significant rewriting and debugging. Therefore, high-quality parallel software can be developed only by a small group of specialists with direct access to expensive equipment. However, using the programming languages with implicit parallelism, such as the functional language Sisal [3], it is possible to overcome this barrier and provide a wide range of application programmers who do not have sufficient access to parallel computing systems, but who are specialists in their application fields, the opportunity to quickly develop high-quality portable parallel algorithms in their workplace.

In a functional program, the programmer only needs to specify the results of calculations and can transfer most of the work on organizing them to the compiler, which is responsible for mapping the algorithm to a specific architecture of the computer (including scheduling instructions, transmitting data, synchronizing calculations, managing memory, etc.). Compared to other functional languages, Sisal supports data types and operators inherent in scientific computing, such as loops and arrays.

Therefore, it seems very promising to carry out a project to develop a system CPPS that is aimed to be an interactive visual environment of functional and parallel programming for support of computer science teaching and learning [8], [9]. The input language of CPPS is the Cloud Sisal language which carries on the tradition of the previous versions of Sisal [3], [5] while remaining a functional data-flow language focused on writing large scientific programs and expanding their capabilities by supporting cloud computing. The functional semantics of Cloud Sisal guarantees deterministic results for parallel and sequential implementations — something that cannot be guaranteed for traditional imperative languages like Fortran or C. Moreover, the implicit parallelism of the language removes the need to rewrite the source code when transferring it from one computer to another. It is guaranteed that the Cloud Sisal program correctly executed on a personal computer will be correctly executed on any high-speed parallel or distributed computer.

The CPPS system uses an internal graph-theoretic representation of functional and parallel programs focused on their visual processing [10], [11]. It is assumed that using the CPPS system any user will be able to develop, verify and debug a Cloud Sisal program in clouds on his/her low-cost
device in a visual style not taking into account the target supercomputers. Then it is possible to tune the debugged program to a supercomputer available via network in order to achieve high performance execution of the developed parallel program, as well as to transfer this program to the supercomputer to run it and receive its results. The CPPS system can be used also for teaching and learning formal verification, optimizing compilation and high performance computing.

6. Conclusion

In this paper, we considered the methods and tools of teaching programming developed at the Institute of Informatics System SB RAS. The purpose of the research is to make the process of teaching and learning programming at a university more individual, accessible and effective. Some of the methods and tools are already successfully used at the Department of Mechanics and Mathematics of NSU while others are waiting to be applied in order to bring modern technologies of distance learning and cloud computing to the educational process of our University.

At present, the CPPS system consists of experimental versions of Web-interface, interpreter, graphic visualization / debugging subsystem, verification subsystem, profiler, and optimizing cross-compiler. The current target platform for the Cloud Sisal compiler is .NET. The compiler performs conventional optimizing transformations and generates the C# code. It allows the users to perform the experimental execution of Cloud Sisal programs and to examine the efficiency of optimizing transformations applied by the compiler. We plan to continue working on the compiler and other subsystems of CPPS and to begin some experiments on using our system for teaching and learning the functional and parallel programming, as well as optimizing compilation, formal verification of Cloud Sisal programs, and high performance computing.

Acknowledgements

This work was supported by a grant of the Russian Science Foundation (project 18-11-00118).

References

[1] Brusilovsky P 2001 Adaptive hypermedia User Modeling and User-Adapted Interaction 11 87–110
[2] De Bra P 2002 Adaptive educational hypermedia on the Web Comm. of the ACM 45 (5), 60–61
[3] Gaudiot J-L, DeBoni T, Feco J, et all 2001 The Sisal project: real world functional programming Lecture Notices in Computer Science vol 1808 Springer pp 84–72
[4] Gutknecht J, Zueff E 2003 Zonnon for .NET – a language and compiler experiment Lecture Notes in Computer Science vol 2789 Springer pp 132–143
[5] Kas’yanov V N 2013 Sisal 3.2: functional language for scientific parallel programming Enterprise Information Systems 7 (2) 227–236
[6] Kas’yanov V N, Kas’yanova E V 2009 A Web-based system for distance learning of programming Lecture Notes in Electrical Engineering vol 27 Springer pp 453–462
[7] Kas’yanov V N, Kas’yanova E V 2013 Practical work on programming NSU Press Novosibirsk (In Russian)
[8] Kas’yanov V, Kas’yanova E 2018 A system of functional programming for supporting of cloud supercomputing WSEAS Transactions on Information Science and Applications 15 81–90
[9] Kas’yanov V N, Kas’yanova E V 2019 Methods and system for cloud parallel programming Proceedings of the 21st International Conference on Enterprise Information Systems vol 1 SCITEPRESS pp 623–629
[10] Kas’yanov V, Kas’yanova E, Zolotuhin T 2018 Visualization of graph representations of data-flow programs WSEAS Transactions on Information Science and Applications 15 141–146
[11] Kas’yanov V N, Zolotuhin T A, Gordeev D S 2019 Visualization methods and algorithms for graph representation of functional programs Programming and Computer Software 45 (4) 156–162
[12] Zonnon programming course of MSDN Academic Alliance, http://pco.iis.nsk.su/ICP/