UDC 004:37

THE CODE EDITOR IN THE ARCHITECTURAL AND TECHNICAL DESIGN STRATEGY OF THE INTERACTING COMPONENTS OF THE PROGRAM DEMONSTRATION ENVIRONMENT WHILE CONDUCTING COMPUTATIONAL EXPERIMENTS

Alferov E.

Kherson State University

The paper presents an integrated environment for studying the course «Basics of algorithmization and programming» (http://weboap.ksu.ks.ua), which solves the problem of improving the efficiency of studying this important discipline. We describe the design and development of new version of the program demonstration environment, which allows execution of computational experiments to study the complexity and majorizability of sorting algorithms. Much attention is paid to the development component of the code editor. An architectural design strategy was formed. It determines the solution components and their interactions. The paper describes the technologies and products which were selected as a means for implementing solutions.

Keywords: *basics of algorithmization and programming, program demonstration environment, code editor, computational experiment.*

1 Introduction

One of the most important steps towards improving the effectiveness study of the course «Basics of algorithmization and programming» in higher educational establishments is the development and implementation of software, using the same methodology, and the interaction of all electronic tutorials.

In the integrated environment for the course «Basics of algorithmization and programming» not just learn lexical structures of the programming language, but also ways of algorithmization and their wide use while solving tasks. It is also proposed together with the study of theoretical material to carry out a computational experiment to study the complexity and efficiency of algorithms. This kind of approach to content enhances students' research activities, the fundamental subjective training of future professionals through formal and logical display of cause and effect relationships and, as a result, impacts on the motivation of students [1]. A computational experiment to study the efficiency of algorithms is performed by using a special module «Program demonstration environment» from the integrated environment for course study of the «Basics of algorithmization and programming».

2 Program demonstration environment

Today's version of the module program demonstration environment is integrated by the modified program interpreter Pascal (see figure 1). This version was created with the help of the parser of languages ANTRL – Another Tool Recognition Language, and is made as a separate static library which is performed and compiled in a module demonstration environment [2, 3].

The demonstration environment for performing algorithms is created as an ActiveX component, which is why Internet Explorer browser and Windows platform must be used.

The system of the new version of program demonstration environment includes 6 main components (figure 2):

- Code Editor;
- Input Data Generator;
- Syntax Analyzer (Parser);
- Walker;
- Visualizer;
- Component of Results and Statistics.

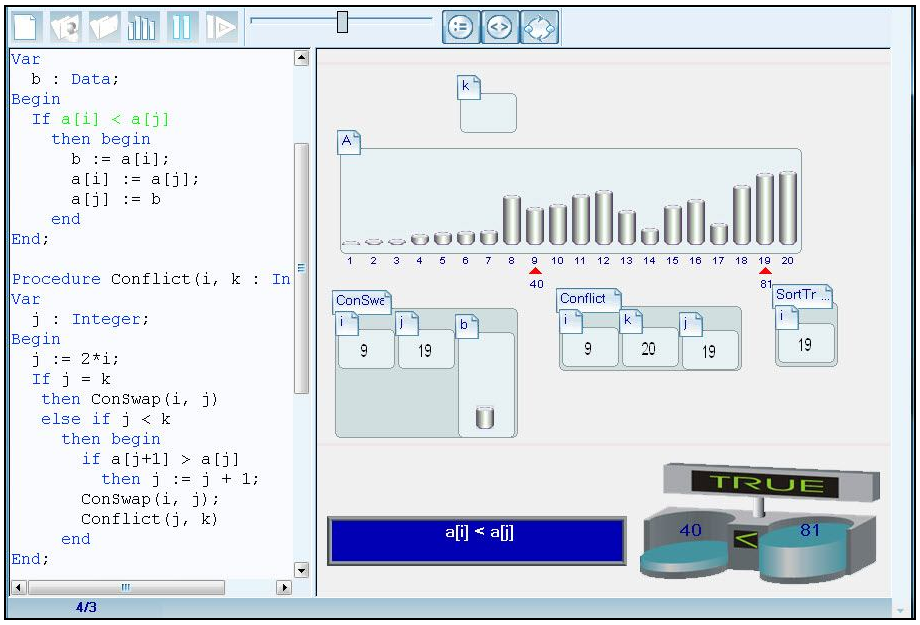

Fig. 1. Today's version of program demonstration environment screenshot.

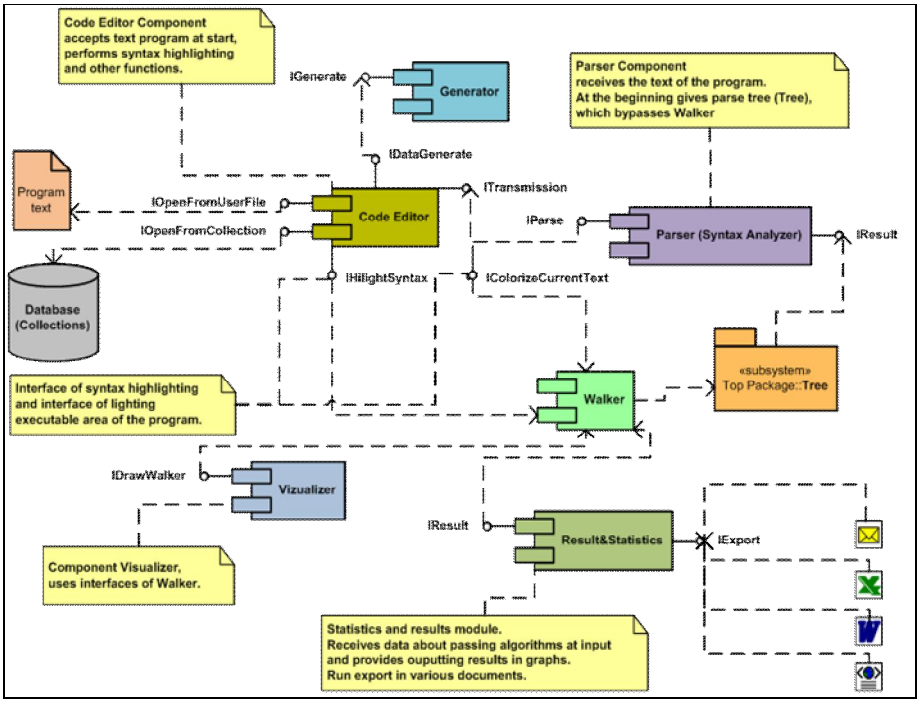

Fig. 2. Diagram of all components of the program demonstration environment and their interaction with each other.

Each of them provides all necessary interfaces that define the above interaction. Relationship of components is represented in the component diagram [4] (see Figure 2).

IOpenFromUserFile interface allows opening the users file with the code.

IOpenFromCollection allows the user to open in the demonstration environment some of the fundamental algorithms contained in the system collection.

IHighlightSyntax – performs syntax highlighting of the program code, gets information about the relevant syntax structure of the Walker component.

ITransmission – is a code editor interface that uses the parser to build an algorithm abstract tree (*Tree*).

IDataGenerate generates input data for the program.

IColorizeCurrentStatement –is an interface that highlights the construction that is currently being performed.

IParse – interface for the parser, which uses *ITransmission* for receiving the text of the program, parses the written code for the necessary structures and builds an abstract tree.

IDrawWalker – visualizer interface that receives data from the parser and performs graphical interpretation of the algorithm.

IResult – the results and statistics component interface which gets information about the algorithm (number of comparisons, assignments, execution time, etc.) and uses this information to show the statistical results.

Currently we are developing a new version of the « demonstration environment» using the advanced technologies Silverlight/Moonlight. The plan is to ensure maximum of cross-platform, interactivity and ease of application software. The future «demonstration environment» will give the ability to view the interpretation of algorithms in such high-level programming languages like Pascal, C and Java.

3 Code editor component in the demonstration environment

In the process of writing code or scripts to implement an algorithm in the selected programming language, one needs, first of all, a practical and comfortable code editor. Coding is a part of programming just like analysis, design, compilation, testing and debugging, accompaniment.

Integrated development environments that simplify and support the process of creating and debugging program source code are one of the most demanding types of software on the market. Popular programming languages often are distributed in multifunctional integrated environments that provide easier access to language features for programmer. Complexity of modern editors, compilers and interpreters requires more automated development environments, which avoid more routine programming work.

The Code Editor provides a set of features that help in writing and editing code. The specific features and their arrangement vary depending on the development language and the current settings.

Today there are many code editors with different degrees of functionality and quality of supporting programming languages. One of the central features of code editors is the analysis of the structure of source code and coloring it on the fly (in the process of editing on the part of user). The main index here is universality. Not many editors can boast of a large list of supported programming languages. In many cases, quantitative indices did not correlate with quality of coloring and analysis of source code.

The development of a new editor provides for editing files in one session, code creation by using patterns of algorithmic constructions, hiding blocks of code for easier reading, writing comments to the code, syntax highlighting, validation of the location of brackets, lighting of executed code and some other additional features that simplify the process of writing code.

Figure 4 shows the properties of the code editor in the new version of program demonstration environment. The editor allows the user to open a search or sort system collection algorithms. One can create one's own collection of algorithms, which can be saved on a personal computer. Parallel to this, one may save or open local files (extensions .pas, .c, .java) with the program code in the appropriate previously selected programming language.

In the code editor is presented basic editing features (copy, cut, paste), line numbering, syntax highlighting of the selected language, patterns of algorithmic structures (types, arrays, operations, functions, etc.) and automatic script completion function intellisense, which will simplify coding programs and offer beginner-programmers the possibility to see the theoretical parts of the material associated with the chosen language.

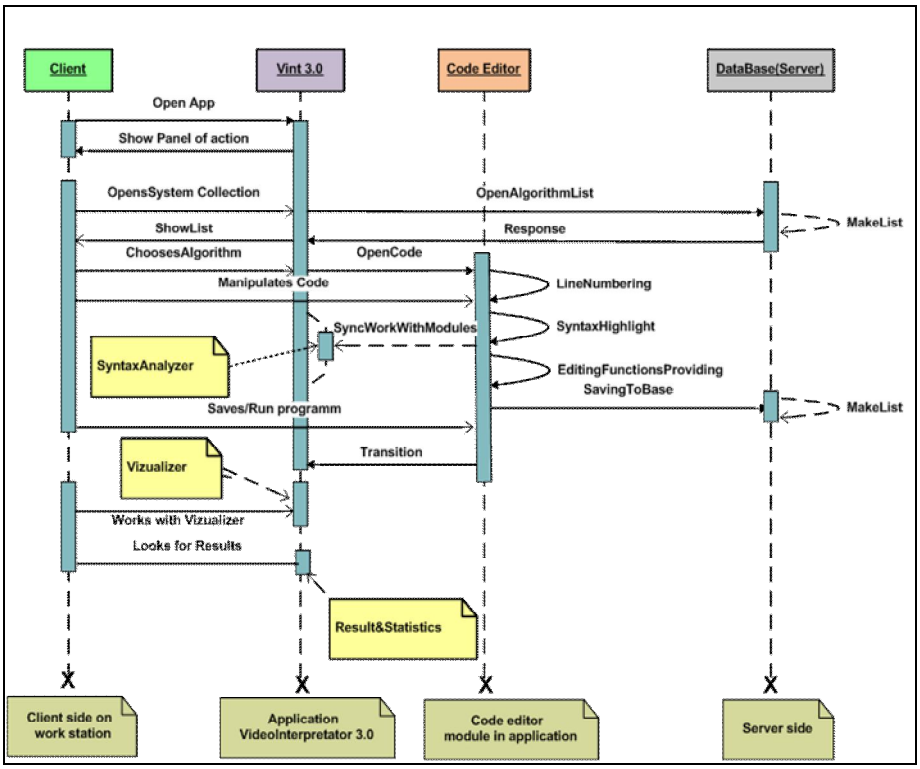

Fig. 3. Sequence diagram, which shows the interaction of objects when working with code editor of environment of programs demonstration, organized by time of their appearance.

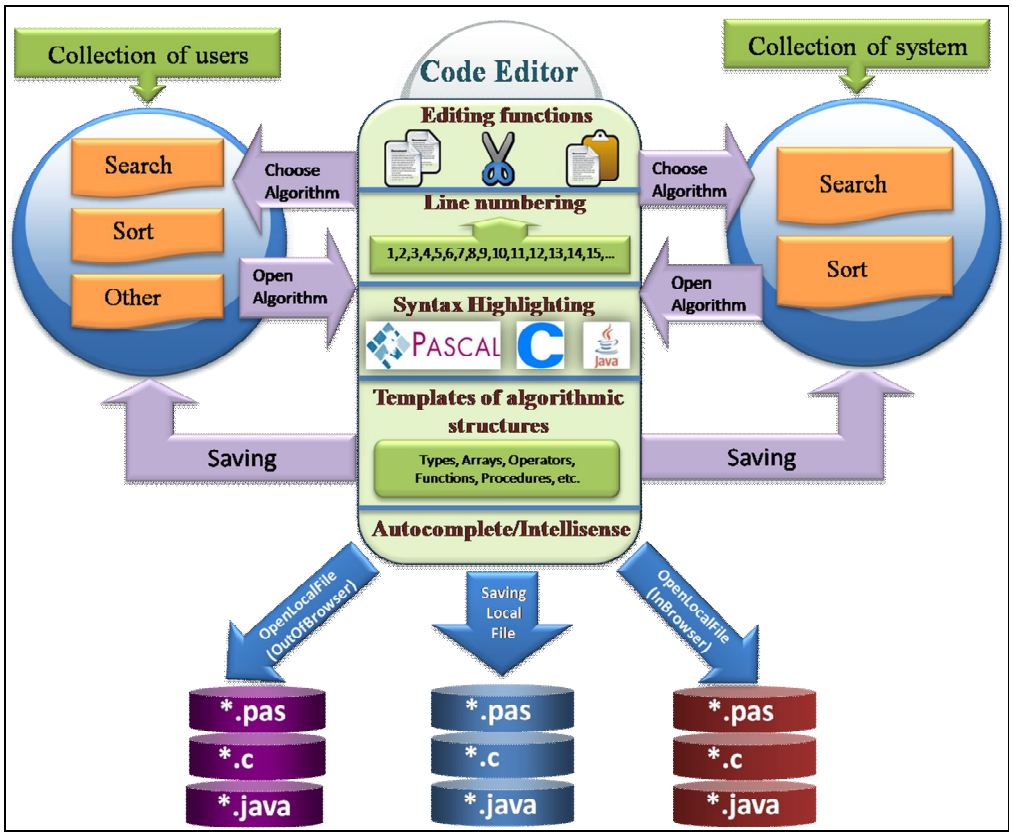

Fig. 4. Above is shown the code editor possibilities of the program demonstration environment.

Creating or designing algorithms is performed in 3 phases:

- Creation of the algorithm code in one of the supported programming languages.
- Generation of data for global algorithm arrays and adding them to the collection.
- The assignment of generated data to the global arrays from the collection

After the user has finished creating the text of the algorithm he/she should press «Build». Then the syntax analyzer will parse the program and create an abstract syntax tree or in case of failure – an error message. In the case of syntactic correctness of the algorithm text a data generation toolbar will be displayed on the right side of the window. The data generation toolbar is divided into three blocks:

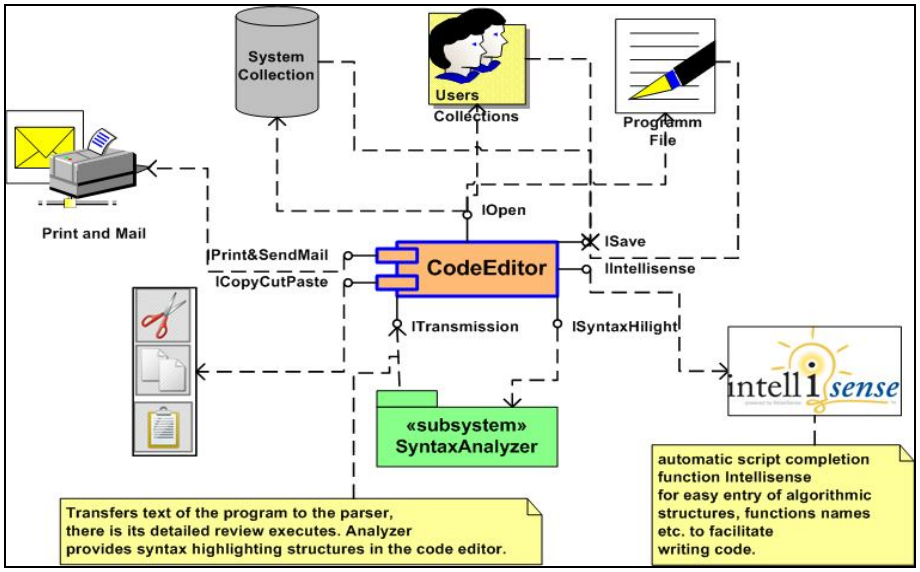

Fig. 5. Above is shown how the code editor component interacts with other components of the programs demonstration environment.

1. An array generation block that includes options for generating the following patterns or templates:
   - An ascending array.
   - A descending array.
   - An array with one extreme point.
   - An array with two extreme points.
   - An array of identical elements.
   - An array of random variables.
2. A block displaying program global variables. All types except arrays can be initialized in the text box next to the name of variable:
   - Arrays of type Data.
   - Logical variables.
   - Integer variables.
   - Real variables.
3. A block of data collection necessary for displaying data sets for global arrays:
   - Adding specific variables in the list.
   - Deleting generated data sets from the collection.
   - Displaying the filtered data set, which includes display of those data sets that match with the size of the selected variable in a block of global variables.

Only after the full generation of input data the visualization of the algorithm is possible. It is implemented by the module Visualizer of program demonstration environment. After the

visualization, the user can see the results in the section, which is responsible for the computational experiment.

With the help of computational experiments students have the opportunity to understand the features of algorithms and understand the dependencies that explain their complexity [1].

4 Conclusions

The programs demonstration environment of the integrated environment for studying of the course «Basics of algorithmization and programming» gives many opportunities for efficient study of this fundamental discipline, including [4]:

- A demonstration of how algorithms work;
- The possibility to perform computational experiments to study the complexity and majorizability of sorting algorithms;
- The opportunity to summarize the results of the algorithm analysis in the comparison of different methods of solving the problem.

Much more attention is given to carrying out computational experiments for studying complexity and efficiency of algorithms using the integrated environment WebBAP (weboap.ksu.ks.ua), developed at the Institute of IT Kherson State University.

In developing the new version of the « demonstration environment» we aim to create a software application that extends the range of opportunities for effective learning of the basics of algorithmization and programming.

REFERENCES

1. Основи алгоритмізації та програмування. Обчислювальний експеримент. Розв'язання проблем ефективності в алгоритмах пошуку та сортування: Навчальний посібник/А.В. Співаковський, Н.В. Осипова, М.С. Львов, К.В. Бакуменко. – Херсон: Айлант, 2010. – 100 с.: іл..
2. Педагогічні технології та педагогічно орієнтовані програмні системи: предметно-орієнтований підхід /О.В. Співаковський, М.С. Львов, Г.М. Кравцов [та ін.] //Комп'ютер у школі та сім'ї. – №4(22), 2002 – С. 24-28
3. Співаковський О.В. Відеоінтерпретатор алгоритмів інтегрованого середовища вивчення курсу «Основи алгоритмізації та програмування» / Співаковський О.В., Колеснікова Н.В. // Нові інформаційні технології в освіті для всіх: система електронної освіти. – 2008. – № 3. – С. 399-404.
4. Алфьоров Є.А. Інструментальні засоби розробки програмного коду, написаного мовою програмування високого рівня / Євген Андрійович Алфьоров// Інформаційні технології в освіті: Збірник наукових праць. Випуск 8. – Херсон: Видавництво ХДУ, 2010. – С.91 – 97.