Proceedings IFTE-2019, 163-169
doi:10.3897/ap.1.e0152

V International Forum on Teacher Education

Current Trends in the Study of Object-Oriented Programming in Higher Education

Tatyana Yu. Gainutdinova*(a), Marina Yu. Denisova (b), Olga A. Shirokova (c)

(a), (b), (c) Kazan Federal University, 420008, Kazan (Russia), 18 Kremlyovskaya street

Abstract

The article examines the current methodological and technological problems of software development, based on an object-oriented approach in the field of programming. The relevance of the topic is determined by the general need to develop a methodology for teaching object-oriented programming based on object-oriented design.
In this paper the problems of teaching object-oriented programming to students studying at the Faculty of Mathematics majoring in teacher education are formulated. The methodology for development of object-oriented projects related to the implementation of mathematical abstractions and structure classes is proposed.
The object-oriented programming enables one to take advantage of the object-oriented approach not only at the design stage, but also at the stage of their implementation, testing and maintenance. The creation of object-oriented modeling projects for mathematical systems and structures facilitates the development of skills for problem formulation and highlighting abstractions and objects of a given subject area and the relations between them.

Keywords: object-oriented programming; object-oriented design; visual components; classes; objects.

* Corresponding author. Tel.: +79063275844; *e-mail:* tgainut@mail.ru

Introduction

Object-oriented programming (OOP) is a discipline included in the basic part of the curriculum for students of the Faculty of Mathematics with a teacher education major (Gainutdinova & Shirokova, 2016a; Gainutdinova & Shirokova, 2016b; Shirokova, 2015). That is why it is necessary to pay serious attention to the methodology of teaching object-oriented programming with the help of information technology. The introduction of information and communication tools is difficult from the viewpoint of methodological and pedagogical aspects of teaching programming in universities.

The use of the object-oriented approach develops students' object thinking skills. However, students face a number of problems. These problems are associated with the complexity of the studied subject area, inability to identify necessary classes and objects, their connections and structures (Booch, 2007; Graham, 2001; Meyer, 2007).

Purpose and objectives of the study

The research is aimed at developing object-oriented programming skills due to projects for implementing the class of a mathematical object “straight line in space” in the object-oriented programming system C ++.

Literature review

There are also the following methodological problems:

Teaching of software design in schools and in the first year of the university is based on the principles of structural programming, which uses functional decomposition of tasks and does not include an object-oriented approach. This type of training develops a stereotype of procedural thinking. Object-oriented programming (OOP) is the next step in the development of a structural approach to programming, which makes use of a person's reasoning rather than structure. OOP provides a user-friendlier interface for programming. Therefore, mastering of object-oriented programming skills is an important aspect in the students' training. Object-oriented decomposition of tasks requires structured programming knowledge.

The course in object-oriented programming has a great practical application, which makes it difficult to balance between theory and practice, considering the limited time frame and programming level of students. This discipline is characterized by a high degree of abstraction and theoretical complexity. The study of OOP reflects the fundamentals of the scientific nature of education, which requires of students broad theoretical knowledge.

Currently, object-oriented programming is focused on the design of complex programs. Therefore, the developers of software tools, as a rule, publish their program implementations in fragments. These fragments comprise the general logic of the program, but lack a detailed description of the stages of object-oriented analysis, design, modeling, and programming. Therefore, there are no good examples of implementing object-oriented software with explanations on the choice of solutions.

In relation to the above mentioned, methodological work on examples of implementing object-oriented projects in the C++ programming language (Pavlovskaya, 2003; Pobegailo, 2006; Schildt, 2008) by teachers is necessary and important, as it forms the basis of experience transferred from teachers to students (Barkov, 2009; Ivanova, 2011; Petrov, 2008).

Methodology

The main building blocks of the object-oriented methodology of program analysis and design are classes and objects. Classes are abstractions of reality.

When running the object-oriented programming course at the Faculty of Mathematics, practical classes are aimed at inculcating the basic skills of developing object-oriented projects. It should be noted that the basis of these projects should be classes and objects, prototypes of which are mathematical structures. Thus, students are expected to create projects related, for example, to the implementation of classes of mathematical abstractions and structures.

The article discusses the development of projects for implementing the class of a mathematical object “line in space” in an object-oriented environment C ++.

We propose a technique that addresses the following sections of object-oriented analysis:

- representation of classes of objects and relations between them when using the means of object-oriented design;
- implementation of object-oriented decomposition in the OOP process;
- use of UML for object-oriented modeling;
- creation of an object-oriented program code using object-oriented design tools;
- implementation of the structure of an object superstructure of the programming system using visual component libraries;
- implementation of the most important classes and their interaction with the operating system.
-

Results

In the course of a project development one encounters the concept of software life cycle which includes processes, actions and tasks that must be performed when projects are created. In the process of object-oriented analysis an enormous amount of attention is given to the definition and description of objects in terms of the subject domain. It is proposed to use the UML system modeling language to develop an object-oriented domain model (Quatrani, 2002). For this purpose analysis and design models are used involving the basic types of diagrams. Throughout the process of object-oriented design logical program objects are defined as future-implemented in a specific programming system. The resulting program objects are classes and these objects include attributes, methods, and properties. An important point in the design is a separate description of the structure and class implementation.

The main task in the development of an object-oriented project is to code using a library of visual components. Moreover, students must know a component hierarchy that describes their interaction; the program environment in which components work; interaction of the program with the operating system.

The development of projects for implementing a class of line in space in an object-oriented environment C ++ gives a teacher an opportunity to solve a number of methodological tasks:

- carry out the object-oriented decomposition of the subject area;
- learn the UML to build analysis and design models applying the main types of diagrams;
- identify logical program objects that will be implemented in the programming environment;
- analyze the programming systems, implement the line in space class and the program structure using the features of the specific environment;
- create object-oriented projects using the features of visual component libraries.

An important methodological aspect is that each subsequent project in the new software system draws on the knowledge gained while developing previous projects. The development of object-oriented models using the UML, the project module codes, the use of basic rules for naming identifiers and other approaches significantly enhance consistency when starting a new project. Students when embarking on a new project are often forced to implement changes in existing programs. Therefore, the awareness of the need to track the program modifiability is quickly built.

Practical sessions of the object-oriented programming course are aimed at implementing the stages of analysis, design, modeling and programming of a given project.

Let us move on to the following project:

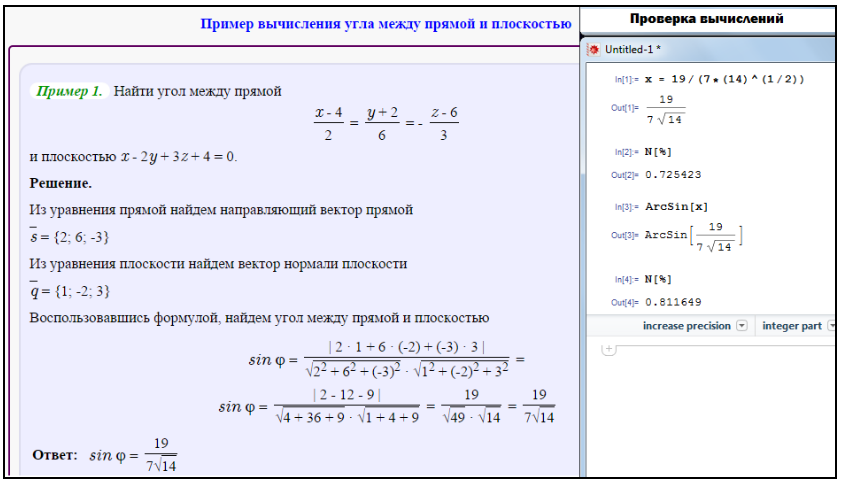
- To create a class of straight lines in space containing, as fields, parameters defining a straight line. The class methods determine the angle of intersection of a straight line and a plane, and methods that define the relationship of a straight line with another straight line.
- To implement the class features for working with specific objects.
- To develop a project in an object-oriented environment C ++.

The structure of the class of straight lines in space and Line and ConerLine methods in C++ are as follows:

```
ref class Line
{
protected:
        Int32 x1, y1, z1;
        Int32 a, b, c;
        Graphics^ g;

public:
        Line( Int32 x1, Int32 y1, Int32 z1, Int32 a, Int32 b, Int32 c );
        Double ConerLine( Line^ l2 );
        Double ConerPlane( Plane^ Z );
};
……………………………………………
Line::Line( Int32 x1, Int32 y1, Int32 z1, Int32 a, Int32 b, Int32 c  ){
        this->x1 = x1;
        this->y1 = y1;
        this->z1 = z1;
        this->a = a;
        this->b = b;
        this->c = c;
}
……………………………………………
Double Line::ConerLine( Line^ l2 ){
        Double w;
        if( l2->a!=0 && l2->b!=0 && l2->c!=0 && this->a/l2->a == this->b/l2->b && this-
>b/l2->b == this->c/l2->c ) w = 0;
        else{
                Double lengthL1 = sqrt(this->a*this->a + this->b*this->b + this->c*this->c);
                Double lengthL2 = sqrt(l2->a*l2->a + l2->b*l2->b + l2->c*l2->c);
                w = acos(fabs((this->a*l2->a + this->b*l2->b + this->c*l2-
>c)/(lengthL1*lengthL2)));
            }
            return w;
        }
        ………………………………………………
        Double Line::ConerPlane( Plane^ Z ){
            Double w;
            if( Z->getA/this->a == Z->getB/this->b && Z->getB/this->b == Z->getC/this->c ) w =
90;
            else {
            if( this->a*Z->getA + this->b*Z->getB + this->c*Z->getC == 0 ) w = 0;
            else {
    Double lengthL1 = sqrt(this->a*this->a + this->b*this->b + this->c*this->c);
    Double r = sqrtf( Z->getA*Z->getA + Z->getB*Z->getB + Z->getC*Z->getC );
    w = asin(fabs((this->a*Z->getA + this->b*Z->getB +this->c*Z->getC)/ (lengthL1*r)));
                }
            }
            return w;
        }
```

To create a visual project, it is necessary to develop a multi-window application of a modular structure that includes forms (Fig. 1, 2). Figure 2 shows the visual project window in C ++.



Пример вычисления угла между прямой и плоскостью

Пример 1. Найти угол между прямой

$$\frac{x-4}{2} = \frac{y+2}{6} = -\frac{z-6}{3}$$

и плоскостью $x - 2y + 3z + 4 = 0$.

Решение.

Из уравнения прямой найдем направляющий вектор прямой

$\bar{s} = \{2; 6; -3\}$

Из уравнения плоскости найдем вектор нормали плоскости

$\bar{q} = \{1; -2; 3\}$

Воспользовавшись формулой, найдем угол между прямой и плоскостью

$$sin\,\varphi = \frac{|2\cdot 1 + 6\cdot(-2) + (-3)\cdot 3|}{\sqrt{2^2+6^2+(-3)^2}\cdot\sqrt{1^2+(-2)^2+3^2}} =$$

$$sin\,\varphi = \frac{|2-12-9|}{\sqrt{4+36+9}\cdot\sqrt{1+4+9}} = \frac{19}{\sqrt{49}\cdot\sqrt{14}} = \frac{19}{7\sqrt{14}}$$

Ответ: $sin\,\varphi = \frac{19}{7\sqrt{14}}$

Проверка вычислений

Untitled-1 *

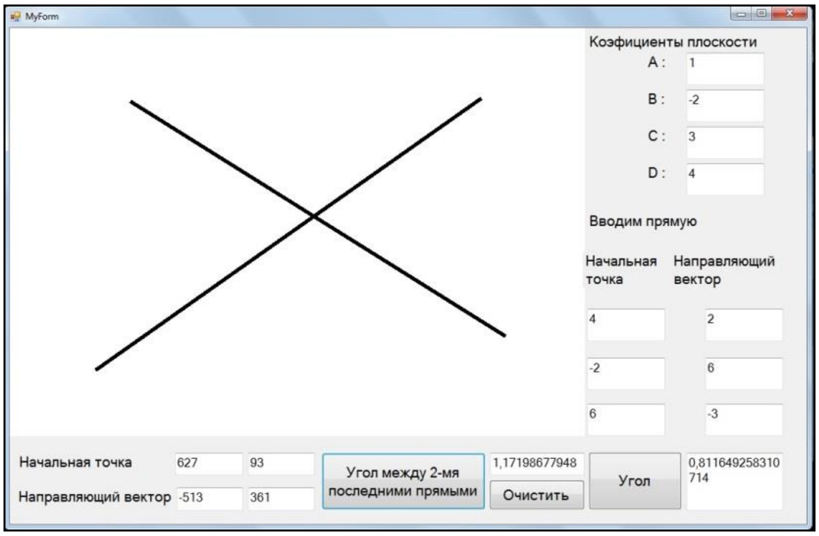
```
In[1]:= x = 19/(7*(14)^(1/2))
Out[1]= 19/(7 √14)

In[2]:= N[%]
Out[2]= 0.725423

In[3]:= ArcSin[x]
Out[3]= ArcSin[19/(7 √14)]

In[4]:= N[%]
Out[4]= 0.811649
```

Figure 1. Calculation Form Window


MyForm

Коэфициенты плоскости

A : 1

B : -2

C : 3

D : 4

Вводим прямую

Начальная точка	Направляющий вектор
4	2
-2	6
6	-3

Начальная точка 627 93

Направляющий вектор -513 361

1,17198677948

0,811649258310714

Figure 2. Visual Project Window

Easy understanding of the program is an important practical principle. Using the naming rules of identifiers incorporated in the programming system leads to the fact that as a result of using standard names (for example, “Button”, “Label”, etc.), the assignment of these components in this program can be determined only after its careful investigation. Therefore, identifiers that denote the names of components, methods, fields, etc., must carry a meaning.

The object-oriented style of programming makes it possible to take advantage of the object-oriented approach not only at the stages of designing and constructing software systems, but also at the stages of their implementation, testing and maintenance.

Discussions

The task of teaching students to think objectively is complex. Object-oriented methodology is based on the concepts of classes and objects. Classes form the structure of data and actions and use this information for the construction of objects. One class can form more than one object at the same time; each of them will be independent of the others. The creation of projects related to the implementation of classes of mathematical abstractions shapes object thinking of students at the mathematical faculty.

The development of object-oriented projects includes the environment and a design object (problems of creating software systems; features of a complex system; a project model; a life cycle of the software product), the design of software products (the meaning of design; the importance of model building; elements of software design; object-oriented analysis, design and programming), principles and tools of object-oriented analysis (classification and object-oriented programming; difficulties of classification; object-oriented analysis), principles and tools of object-oriented designing (abstraction; encapsulation; modularity; hierarchy), advantages of the object model.

The presented methodological approach to the implementation of classes of mathematical abstractions and structures allows: to conduct object-oriented decomposition of the subject area (mathematical object “line in space”); to learn UML for building a project model; to define logical program objects that will be implemented in the C ++ system; to realize the class “straight line in space” and the structure of the program, using the features of the concrete environment; create visual projects.

Conclusion

The article focuses on the peculiarities of studying the course of object-oriented programming technology. The object-oriented paradigm of programming contributes to the efficiency of problem solving and is one of the promising trends in the development of programming environments.

This paper presents a methodology for future computer science instructors to teach object-oriented programming and design. The methodology includes the following components of the educational process: a conceptual aspect aimed at setting the goal and objectives of learning; an informative element aimed at selecting the course contents. The information component includes the following sections: the theoretical foundations of OOP and object-oriented design; object model of software development using UML; development of object-oriented software code.

Object-oriented modeling projects for mathematical systems and structures develops skills for problem formulation, highlighting abstractions and objects of a given subject area and the relations between them, creating an object-oriented program code using the modeling language and tools of object-oriented design.

References

Barkov, I. A. (2009). Teaching the discipline “Object-Oriented Programming”. *Educational Technologies and Society, 12*(4), 494-516.

Booch, G. (2007). *Object-Oriented Analysis and Design with Applications in C++*. Boston, MA: Addison-Wesley.

Gainutdinova, T. U., & Shirokova, O. A. (2016a). Features of a professional training teachers of informatics in a programming course. *The European Proceedings of Social & Behavioral Sciences EpSBS, XII*, 30-37.

Gainutdinova, T. Yu., & Shirokova, O. A. (2016b). Features of professional training in programming the future teacher of computer science. *Program and theses of 2nd International Forum on Teacher Education* (pp. 231-232). Kazan: Kazan University.

Graham, I. (2001). *Object-Oriented Methods*. Boston, MA: Addison-Wesley.

Ivanova, G. S. (2011). *Programming technology*. Moscow: Knorus.

Meyer, B. (1997). Object-Oriented Software Construction (2nd ed.). London: Prentice Hall.

Pavlovskaya, T. A., & Shchupak, Yu. A. (2003). *C/C++. Structural programming: Workshop*. St. Petersburg: Piter.

Petrov, A. N. (2008). Features of the methodology for teaching students object-oriented programming and design. *Sovremennyye naukoyemkiye tekhnologii*, 126-128.

Pobegailo, A. P. (2006). *C/C ++ for students.* St. Petersburg: BHV–Petersburg.

Quatrani, T. (2002). *Visual Modeling with Rational Rose 2002 and UML* (3rd Ed.). Boston, MA: Addison-Wesley Professional.

Schildt, H. (2008). *Herb Schildt's C++ Programming Cookbook*. Osborne/McGraw Hill.

Shirokova, O. A. (2015). Object-oriented programming with creation of classes for objects of type “massive” and “matrix”. In P. P. Oleynik (Eds.), *Object systems: Proceedings of XI International Theoretical and Practical Conference* (pp. 15-23). Rostov-on-Don: SI (b) SRSPU (NPI).