СЕКЦИЯ «ТЕХНИЧЕСКИЕ НАУКИ»

УДК 004.43

КЛАССИФИКАЦИЯ ТВОРЧЕСКИХ ЗАДАЧ ПО ПРОГРАММИРОВАНИЮ

Вакалюк Татьяна Анатольевна,
доктор педагогических наук, доцент,
профессор кафедры компьютерной инженерии и кибербезопасности,
Государственный университет «Житомирская политехника»,
г. Житомир, Украина
Коротун Ольга Владимировна,
кандидат педагогических наук, доцент кафедры компьютерных наук,
Житомирский государственный технологический университет,
Государственный университет «Житомирская политехника»,
г. Житомир, Украина,
Антонюк Дмитрий Сергеевич,
кандидат педагогических наук,
доцент кафедры инженерии программного обеспечения,
Государственный университет «Житомирская политехника»,
г. Житомир, Украина,
Новицкая Инесса, *кандидат педагогических наук,*
доцент кафедры педагогики,
Житомирский государственный университет имени Ивана Франко»,
г. Житомир, Украина

АННОТАЦИЯ

В статье приводится авторское толкование термина "творческая задача по программированию", представлена авторская классификация творческих задач по программированию, а также рассматриваются некоторые типы решения таких задач. Авторская классификация творческих задач по программированию предусматривает разделение задач на три блока: по разделам изучения (предполагается решение задач по основным темам изучения дисциплины "Программирование"), за необходимыми знаниями в других областях (сделан акцент на задачах, при решении которых должны использоваться базовые знания по соответствующим предметным областям), по уровню сложности (акцентируется внимание на задачах различных уровней сложности). Установлено, что при решении творческих задач по программированию у студентов формируются умения и навыки решения творческих задач, что способствует их умственному развитию, а также развиваются такие мыслительные операции, как классификация и систематизация, при решении такого плана задач обеспечивается творчество и дискуссии в поиске решения, необходимость объяснять свой ответ и рассуждать. При проведении исследования использовались следующие методы: анализ теоретических источников по вопросам решения творческих задач по программированию, обобщение и систематизация полученных результатов, обобщение педагогического опыта.

Ключевые слова: задача, творческая задача, задача по программированию, творческая задача по программированию, классификация творческих задач по программированию.

CLASSIFICATION OF CREATIVE TASKS FROM PROGRAMMING

Vakaliuk Tetiana,
Doctor of Pedagogical Sciences, Associate Professor,
Professor of the Department of Computer Engineering and Cyber Security
Zhytomyr Polytechnic State University, Zhytomyr, Ukraine
tetianavakaliuk@gmail.com
Korotun Olha,
PhD in Pedagogics, associate professor of the Department of computer science,
Zhytomyr Polytechnic State University, Zhytomyr, Ukraine,
olgavl.korotun@gmail.com,
Antoniuk Dmytro
PhD in Pedagogics, associate professor
of the Department of software engineering,
Zhytomyr Polytechnic State University, Zhytomyr, Ukraine,
dmitry_antonyuk@yahoo.com

Novitska Inesa,
PhD in Pedagogics, Associate Professor of the Department of Pedagogy,
Zhytomyr Ivan Franko State University, Zhytomyr, Ukraine,
inesanovicka@gmail.com

ABSTRACT

The article gives an author's interpretation of the term "creative task in programming", the author's classification of creative programming tasks is given, and some types of solving such tasks are considered. The author's classification of creative tasks in programming involves the division of tasks into three blocks: according to the sections of study (it is planned to solve problems according to the main subjects of studying the discipline "Programming"), with the necessary knowledge in other fields (the emphasis is placed on the tasks in solving which use basic knowledge of the relevant subject areas), in terms of complexity (emphasis is placed on tasks at different levels of complexity). It is asserted that when solving creative tasks in programming students develop the skills and skills of solving creative tasks that promote their mental development, as well as developing such thoughtful operations as classification and systematization, when solving such a plan of tasks creativity is provided. And discussions in finding a solution, the need to explain your own answer and reason. During the research, the following methods were used: analysis of theoretical sources on solving creative tasks in programming, generalization and systematization of the obtained results, generalization of pedagogical experience.

Key words: task, creative task, task on programming, creative task on programming, classification of creative tasks in programming.

Traditional methods of securing and testing knowledge in the process of computer science education, including programming (laboratory work, tests, classroom and non-classroom modular control works) are good to show how accurately learned the studied material is. However, these methods do not reveal how the student learned to use, use, and manipulate various knowledge.

The use of creative tasks in programming classes greatly enhances the effectiveness of the learning process. From the conditions of tasks of this type, it is not directly possible to say what kind of knowledge students will need to solve for them, and therefore the students' activity is aimed at identifying ways of solving and selecting the necessary data, information and regularities. Solving creative tasks may require a variety of knowledge on other subjects, for example, on algebra, geometry, probability theory, etc. The main sign that students are in the creative process is the rejection of traditional approaches to the interpretation of existing information.

Many scientists addressed the problems of solving creative tasks in various fields, among them: E. Grigorova, A. Davydenko, S. Danilenko, K. Knop, I. Lerner, Yu. Murashkovsky, S. Pritulyak and others. However, only some scientists, such as I. Lerner, introduced the term «creative task». As for the term "creative task of programming", many scientists turned to him, but no one gave a specific definition and did not indicate their classification.

The purpose of this article is to isolate the concept of "creative task in programming," the author's classification of such tasks, the analysis of the solution of some problems of this type.

In order to proceed to the definition of a creative task in programming, you must first turn to the notions of concepts such as "task", "creative task," "task on programming."

The term "task" in the explanatory dictionary of the Russian language is interpreted in different meanings: 1) in general - that requires execution, decision; 2) in the mathematical exercise, which is performed by means of inference, computation; 3) in the scientific - a complex issue, a problem that requires research and decision [3]. A similar understanding of this concept, we find in the explanatory dictionary Ushakov [4].

Let us assume that the task of programming - this is such a task, which involves finding an algorithm for solving a problem with the means of a certain programming language.

By the term "creative task" in various fields addressed by many scientists, listed above. But this notion was introduced only by some scholars, for example: "Creative," writes I. Lerner, "is considered a task whose actions are not deterministic or incompletely (ambiguously) determined by some prescriptions, that is, if the solver does not know the algorithm of solving" The search is necessary and steps need not be taken beforehand" [2, p. 81].

Summing up the above, we will assume that the creative task of programming is a task that involves finding and constructing an algorithm for its solution using existing methods, followed by implementation of a certain programming language, in which students and students actively acquire new knowledge, mastering abilities and skills, developing abstract and logical thinking, own creative abilities, cognitive interest.

We propose the author's classification of creative programming tasks, which is shown in Fig. 1, according to which such tasks are divided into three blocks: according to the sections of study (it is planned to solve tasks on the main subjects of studying the discipline "Programming"), with the necessary knowledge in other fields (the emphasis is put on the tasks, in solving which basic knowledge in the relevant subject areas), in terms of complexity (emphasis is placed on tasks at different levels of complexity).

Here are examples of the tasks of each classification to some types with explanation for the solution.

In the classification by sections of study, we consider the following types:

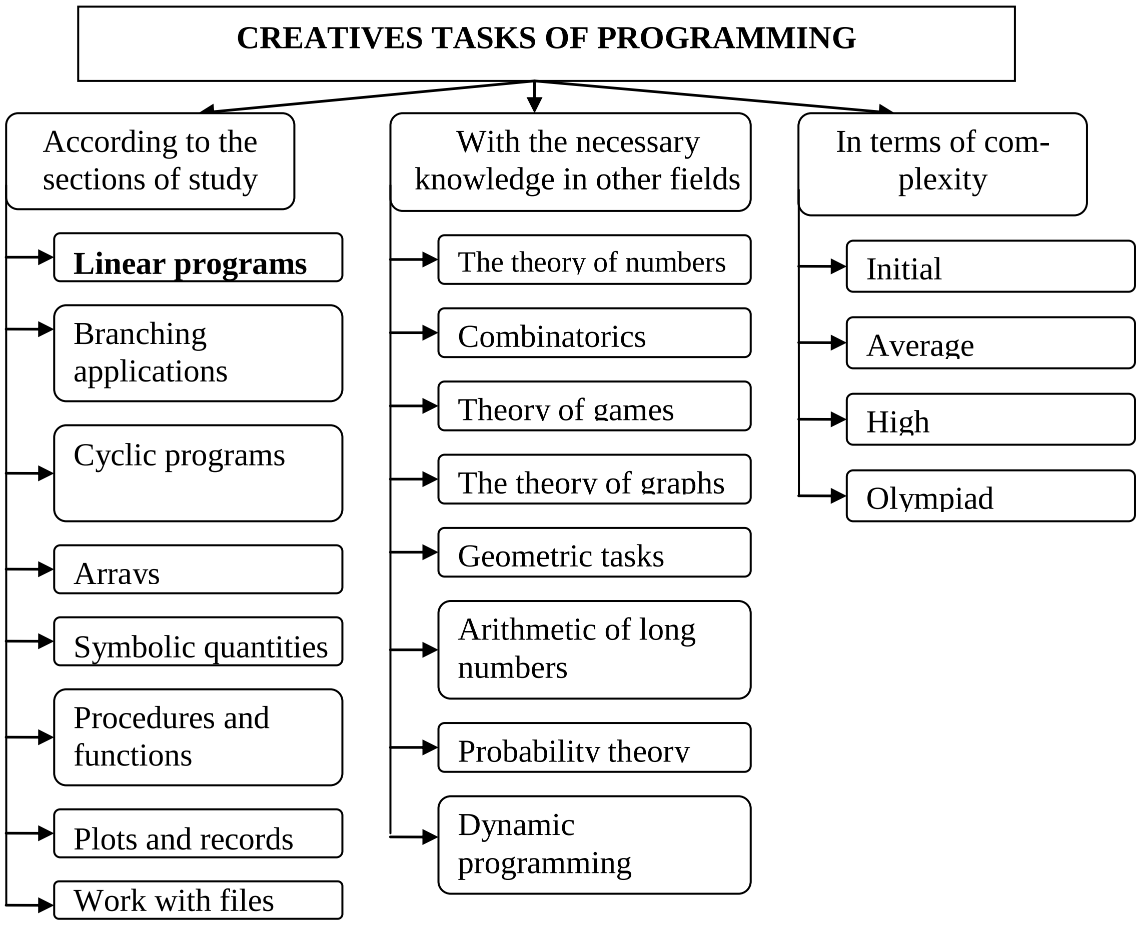

Fig. 1. Classification of creative programming tasks

1. Linear programs

Task 1. Given two integers a and b. If a is divided by b or b is divided by a, then display it on screen 1, otherwise - any other number. Conditional operator and cycle operators do not use.

Solution If a is divisible by b (or b by a) then this means that if the division is still 0. And any number in the product with 0 will give 0, and adding 1 - we get 1. Otherwise, there will be any other number except 1.

Hint: the solution should have an assignment operator, the right part of which looks like: (a mod b) * (b mod a) +1.

As can be seen from this example, when solving such a student plan, the teacher provides creativity and discussion with peers and the teacher in finding a solution to the problem, the need to explain their own response and reason.

2. Cyclic programs.

Task 2. Make a program for finding the area of the figure by a trapezoid method, limited by curves: $y = x^2$ and $y = x^4$.

Solution. Having constructed a graph, we see: the resulting figure is symmetric with respect to the OY axis (see Fig. 2). Therefore, we are looking for the area of the figure, which is in the fourth quarter, and 2 multiply the result. This figure is projected onto the axis of the OH in the segment [0; 1], therefore the limits of integration: a = 0, b = 1. The area of the figure bounded by the lines will be calculated as an integral, where the function $f(x)=x^2-x^4$: $S = 2\int_0^1 \left(x^2 - x^4\right)dx$.

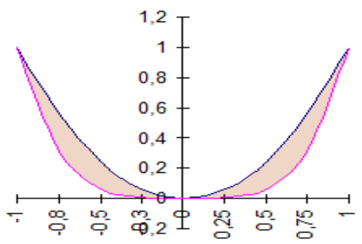

Fig. 2. Graphical representation of task solution 2

The formula for finding the area of the figure by the trapezium has the form: $\int_a^b f(x)dx \approx \frac{b-a}{2n} \cdot \left(f(a) + f(b) + 2\sum_{k=1}^{n-1} x_k \right)$.

Therefore, the program will look like:

```
    #include<iostream.h>
    #include<math.h>
    void main()
    {     float a=0,b=1,x,s=0;
      int i;
      for (i=1; i<=10; i++)
      {
                  x=a+i/10;
                 s=s+(pow(x,2)-pow(x,4));
    }
         s=2*(b-a)/20*(pow(a,2)-pow(a,4)+pow(b,2)-pow(b,4)+2*s);
       cout.precision(2);
      cout<<"s="<<s;
}
```

3. Procedures and functions

Task 3. Define common divisors of two natural numbers.

Solution. We look first at least between the two numbers. Then we set the cycle from 1 to the least among the two given numbers. The shared divisor will be determined by the following criterion: if both numbers are divisible by the variable that passes the loop, it still gives zero, and then this number is a common divisor of two given numbers**.**

```
#include <iostream.h>
#include <math.h>
int s(int, int, int)
void main()
{
  Int c,b,c,min;
  cin>>a>>b;
        if (a<b) min=a; else min=b;
        for (c=1; c<=min; c++)
                if (s(a,b,c)==1) cout<c<<" ";
}
int s(int a1, int b1, int c1)
```

```
{
 Int s1;
       if (a1 %c1==0 && b1 %c1==0) s1=1;
       else s1=0;
 return s1;
}
```

From the classification of the necessary knowledge in other fields, we consider:

4. Geometric Task.

Task 4. The coordinates of its vertices, which are entered in turn clockwise, give the N-angle. Calculate its area using the vector product.

Solution. Let us divide the polygon into triangles, as shown in Fig. 3. The area of each such triangle will be equal to a vector product divided by 2. Having found all such areas and adding them, we get the area of the polygon. For clarity, consider the area of one such triangle. Let the three vertices of such a triangle (in Fig. 3, respectively, M1, M2 and M3) have pairwise coordinates x1, y1, x2, y2, x3, y3. We put the triangle as created by two vectors, which proceed from one point: $a=(a_x;a_y;a_z)$, $b=(b_x,b_y,b_z)$, де $a_x=x_2–x_1$; $a_y=y_2–y_1$; $a_z=0$; $b_x=x_3–x_1$; $b_y=y_3–y_1$; $b_z=0$. It is known that the area of a triangle is calculated by the formula: $S=\frac{1}{2}\sqrt{\begin{vmatrix} a_y & a_z \\ b_y & b_z \end{vmatrix}^2+\begin{vmatrix} a_x & a_z \\ b_x & b_z \end{vmatrix}^2+\begin{vmatrix} a_x & a_y \\ b_x & b_y \end{vmatrix}^2}$

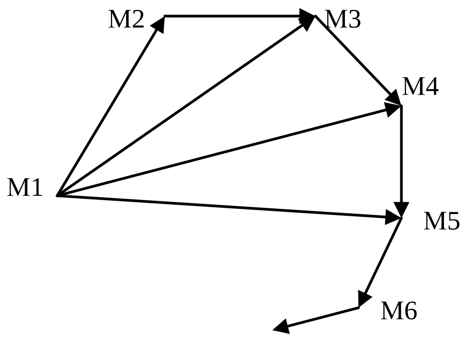

Fig. 3. The polygon is divided into triangles

Substituting the corresponding values, we obtain:

$$S=\frac{1}{2}\sqrt{\begin{vmatrix} a_x & a_y \\ b_x & b_y \end{vmatrix}^2}=\frac{1}{2}\left|a_x b_y - b_x a_y\right|.$$

Then for each next triangle we leave, the first vertex (Fig. 3) unchanged, take the second as the third vertex of the previous triangle, and enter the third vertex of the new triangle again. We calculate the area for the new triangle and so on to the last vertex.

The corresponding implementation program of the described algorithm will look like:

```
#include <iostream.h>
#include <math.h>
void main()
{ int n, i;
  float x1,y1,x2,y2,x3,y3, s=0, ax,ay,bx,by;
  cin>>n;
  cin>>x1>>y1>>x2>>y2>>x3>>y3;
    ax=x2-x1; ay=y2-y1; bx=x3-x1; by=y3-y1; s=s+abs(ax*by-ay*bx)/2;
  if (n>=4)
  for (i=4; i<= n; i++)
  {
          x2=x3; y2=y3;
          cin>>x3>>y3;
          ax=x2-x1; ay=y2-y1; bx=x3-x1; by=y3-y1;
          s=s+abs(ax*by-ay*bx)/2;
  }
  Cout<<"s="<<s;
}
```

5. The theory of numbers.

Task 5. For n given numbers, calculate the NSD by the Euclidean algorithm.

Task 6. For two given numbers, check whether they are interrelated.

There are quite a lot of such tasks, none of which can be counted. Tasks of this type are solved with the help of known properties, definitions of the course of mathematics.

For example, a natural number is called simple if it is greater than 1 and is divided by only 1 and on itself (number 1 is not simple). That is, in order to find out whether the number is simple, it is enough to set a cycle from 1 to the number, and check how many numbers from this interval will be divisors of a given number. If such numbers are 2, then the number is simple, otherwise it is not simple, but is either 1 or composed. Here is a fragment of the algorithm:

```
    ....k=0;
for (i=1; i<=n; i++)
        if (n %i ==0) k++;
if (k==2) cout<<"Число просте"; else cout<<"Число не є просте";...
```

Regarding the last classification, in terms of complexity, a rather large selection of tasks of these types is on the website developed by us www.e-olymp.com, where a special methodological page for the division of tasks was developed according to the complexity levels, entitled "Olimpiad Course". By clicking on the link to the appropriate level of difficulty, a list of tasks offered by the authors will be opened.

Conclusions. Thus, when solving creative tasks in programming, students develop their skills and skills in solving creative tasks that promote their mental development, as well as developing such thoughtful operations as classification and systematization, while solving such a task plan, creativity and discussions in finding a solution, the need to explain your own answer and reason.

References

1. Vakaliuk, T.A. 2013. The Application of Internet Portal e-olimp to Software Development Classes at Universities. Information Technologies and Means of Training, Vol. 4, pp. 84-97.

2. Lerner, I. Ya., 1981. Didactic basics of teaching methods. Moscow, Pedagogics.

3. Ozhegov, S.I., Shvedova, N.Yu. Explanatory Dictionary of the Russian Language (online version). Retrieved from: http://www.classes.ru/all-russian/russian-dictionary-Ozhegov-term-8665.htm.

4. Ushakov's exact dictionary. Retrieved from: http://www.slovopedia.com/3/199/786542.html.