Section 3. Pedagogy

https://doi.org/10.29013/ESR-20-11.12-21-23

Ibrahymova L. A.,
Art. lecturer at the Department of Informatics and Cybernetics
Melitopol State Pedagogical University
named after Bohdan Khmelnytsky, Ukraine
E-mail: ibragimovaludmila19@gmail.com

FORMATION OF ALGORITHMIC COMPETENCIES IN FUTURE SOFTWARE ENGINEERS IN THE PROCESS OF STUDYING PROGRAMMING

Abstract. The article considers the formation of algorithmic competencies in future software engineers in the specialty 122 "Computer Science" in the process of studying the cycle of programming disciplines. An example of the application of the competence approach in the process of studying the discipline "Programming" is given.

Keywords: algorithmic competence, software engineers, programming, competence approach.

The modern specialist must be an expert in his work, have the ability to respond to events and find new ways to solve problems that arise, as well as meet the changing demands of the labor market [1, P. 5]. Therefore, the question of quality training of future software engineers who will be able to adapt to the rapidly changing conditions of professional activity, implementation and operation, maintenance of new software systems that can make adjustments to program codes or based on them to develop fundamentally new software that meets all modern requirements for their development. In other words, the graduate must not only acquire certain knowledge, skills and abilities in the disciplines of the university course, but also be able to apply this knowledge in practical professional activities to solve problems [2, P. 233].

Algorithmic competence is key in the process of training future software engineers in the specialty 122 "Computer Science", it is characterized by a certain level of development of algorithmic thinking, awareness of the general components of algorithmization and, consequently, used in various forms of algorithmic activities that encourage new learning and knowledge.

Thus, algorithmic competence provides the formation of algorithmic culture, and its result is not only the formed knowledge and skills, but also the acquired skills of algorithmic thinking, which are used both in professional activities and in the social environment and in everyday life.

Algorithmic competence of a future software engineer can be defined as an integrated learning outcome, which is formed primarily through mastering the content of algorithmic subjects and gaining experience in using algorithmic knowledge in the process of teaching subjects from the compulsory and elective cycle.

Programming is the first among professionally-oriented disciplines studied by future software engineers in the specialty 122 "Computer Science", the number of credits 41, equal to 1230 hours.

From the content of the course, from the methods of its teaching, from the knowledge, skills and abilities that students will receive in this course, depends on their further education, the quality of learning disciplines based on programming knowledge, success in future professional activities. At Melitopol State Pedagogical University named after Bohdan Khmelnytsky, students study programming from the first to the fourth year of study. In the second year there is a training practice in programming, which is 90 hours, 3 credits. It is also planned to write a term paper in the third and fourth year [4].

Taking into account the TIOBE index and the demand on the labor market, the sequence of studying programming languages for the specialty 122 "Computer Science" of the compulsory component of the educational program, the discipline "Programming", was proposed as follows:

- 1 course programming language C – 1 semester and C ++ – 2 semester;
- 2nd year C # programming language;
- 3rd year java –1 semester and javascript-2nd semester;
- 4th year Python.

Analyzing the special competencies to which the discipline "Programming" corresponds, from the standard of the educational-professional program of the specialty 122 "Computer Science" the following SC_3, SC_4, SC_8 were determined which allow to form algorithmic competence, in particular:

SC_3. Ability to think logically, build logical conclusions, use formal languages and models of algorithmic calculations, design, development and analysis of algorithms, evaluate their effectiveness and complexity, solvability and unsolvability of algorithmic problems for adequate modeling of subject areas and creation of software and information systems.

SC_4. Ability to use modern methods of mathematical modeling of objects, processes and phenomena, to develop models and algorithms for numerical solution of mathematical modeling problems, to take into account the errors of approximate numerical solution of professional problems.

SC_8. Ability to design and develop software using different programming paradigms: generalized, object-oriented, functional, logical, with appropriate models, methods and algorithms of calculations, data structures and control mechanisms [4].

In accordance with the considered special competencies, laboratory works were developed which allow to form in students from the first to the fourth year of study algorithmic competence in the process of studying disciplines from the programming cycle.

Next, consider an example of laboratory work "Rapid Hoar Sorting" in the first year. The purpose of the work is to consolidate theoretical knowledge and gain practical experience in organizing a set of static and dynamic data structures, fast sorting of Hoare (quicksort). Students are encouraged to write code that implements a quick array sort on my C ++ programming. At the beginning of the work the main idea of the algorithm is discussed, which is implemented using pseudocode.

The main idea of the algorithm.

Let the pointers *L* and *R* be such that all the elements to the left of a [*L*] are smaller than the reference element, and the elements to the right of a [*R*] are smaller than the reference element. Moving the pointer *L* to the right (pointer *R* to the left), find the element not less (not more) basic and to exchange their places. The process continues until the pointer *L* is to the right of the pointer *R*.

Hoare fast sort pseudocode

```
repeat
{while a [L] < x
L = L + 1;
while a [R] > x
R = R – 1;
if L < = R
then {rearrange (a [l], a [r])
L = L + 1;
R = R – 1;}}
docks (L > R)
```

The next step is to write a program in C ++.

In the process of performing this laboratory work, students analyze the presented algorithm, learn to read the pseudocode and step-by-step implementation of the algorithm in the C ++ programming language. This allows students to form algorithmic and logical thinking, learn to apply the acquired skills in programming and algorithms in the process of solving problems.

In the senior course for formation of algorithmic competence laboratory works have other direction. Students have already mastered the basic algorithms, know several programming languages, so the tasks are difficult, for example:

In the fourth year students study the Python programming language, one of the laboratory works has the task of writing a program by the method of fast Hoare sorting, this method was already considered by first-year students, it was implemented in the C ++ programming language. One of the conditions of laboratory work is to write a program without the use of additional memory.

Program code.

```
def Quick Sort (A, l, r):
 if l > = r:
  return
 else:
  q = random.choice (A [l: r + 1])
  i = l
  j = r
  while i <= j:
   while A [i] < q:
    i + = 1
   while A [j] > q:
    j - = 1
   if i <= j:
    A [i], A [j] = A [j], A [i]
    i + = 1
    j - = 1
    Quick Sort (A, l, j)
    Quick Sort (A, i, r)
```

After writing a program in Python, students must determine the complexity class and execution time of the program, and also need to make a comparative analysis of this algorithm, written in programming languages such as C ++, C # and Java. The complexity of the program depends on the size of the input data, so you need to check the array for 100 elements and 1000. The results of comparisons and conclusions must be presented to protect laboratory work.

Conclusion

Thus, the achievement of positive results in the formation of algorithmic competence of future software engineers in the field of training 122 "Computer Science" in the study of the cycle of programming disciplines was achieved by creating conditions for implementing the basic principles of competency approach. The use of laboratory work in the training of future software engineers contributes to the formation of their algorithmic competence and allows you to build a holistic pedagogical process.

References:

1. Keycit 2014: Key Competencies in Informatics and ICT / T. Brinda, N. Reynolds, R. Romeike, A. Schwill (eds.). Potsdam: Universitätsverlag Potsdam, 2015. – 446 p. URL: https: //publishup.uni-potsdam. de / opus4-ubp/frontdoor/deliver index/docId/7032/file/cid07.pdf
2. Lebedeva T. N. Peculiarities of application of the competence approach in training of students of qualification "Engineer-programmer" // Psychology and pedagogics: methods and problems of practical application. 2010.– No. 16–2. -– P. 232–237.
3. Kruglyk V., Osadchyi V. Structure of Professional Competence of Future Software Engineers, Pedagogical Discourse, 2016.– Issue 21.– P. 69–74.
4. Ministry of Education and Science of Ukraine [Electronic resource] – Mode of access to the resource: URL: https://mon.gov.ua/ua/npa/pro-zatverdzhennya-standartu-vishoyi-osviti-za-specialnistyu-122-kompyuterni-nauki -dlya-pershogo-bakalavrskogo-rivnya-vishoyi-osviti.