JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 15 (2020) 202-205
Received: 27 April 2020
Accepted: 6 May 2020

Analysis of typical programming mistakes made by first and second year IT students

Analiza typowych błędów programistycznych popełnianych przez studentów pierwszego i drugiego roku Informatyki

Monika Kaczorowska*

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

The research paper contains a review and analysis of common programming mistakes made by first and second year students of Computer Science. The data were collected during the courses entitled "Algorithmics and Data Structures" and "Numerical Methods", where students have to write programs in the C++ language. The article includes examples of three selected mistake types. A comparison of mistakes made by first and second year students is presented. The analysis carried out shows that the percentage of mistakes made decreases when the students are in the second year, but three types of mistakes demonstrate the opposite trend. It can be concluded that those three types of mistakes are related to the course of Numerical Methods, where students have to deal with a significant amount of mathematical expressions. The results show that the students have the most significant problems with memory management.

Keywords: programming mistakes; C++; education

Streszczenie

Artykuł zawiera przegląd i analizę typowych błędów programistycznych popełnianych przez studentów pierwszego idrugiego roku Informatyki. Dane wykorzystane podczas analizy zostały zgromadzone w trakcie zajęć z przedmiotów: Algorytmy i Struktury Danych oraz Metody Numeryczne. Podczas zajęć studenci piszą programy w języku C++. Artykuł zawiera przykłady trzech wybranych typów błędów. W artykule przedstawione zostało porównanie błędów popełnianych przez studentów pierwszego oraz drugiego roku. Przeprowadzona analiza wykazała, że liczba popełnianych błędów jest mniejsza dla studentów drugiego roku ale przy trzech rodzajach błędów można było zaobserwować odwrotną tendencję. Błędy te powiązane są ze specyfiką przedmiotu Metody Numeryczne. Podczas tego przedmiotu studenci mają do czynienia w większym stopniu z wykonywaniem obliczeń matematycznych. Wyniki pokazują, że studenci mają największe problemy z zarządzaniem pamięcią.

Słowa kluczowe: błędy programistyczne; C++, edukacja

*Corresponding author

Email address: m.kaczorowska@pollub.pl(M. Kaczorowska)

1. Introduction

Programming is an essential part of any Computer Science course. IT students study several programming languages such as: C, C++, Python and Java. They write programs not only during classes dedicated to programming, but also use their programming skills during other classes, where they have to write some lines of code. It is said that the more programs are written, the fewer mistakes will appear, as it is obvious that every learner makes mistakes. It can be said that making mistakes in programming is a part of the learning process.

One of the first topics concerning programming mistakes appeared during a workshop on Empirical Studies of Programming [1]. In [2] the authors focused on the mistakes which students made in the Java language and how they managed them. There are papers where the authors analyse how long it takes students to solve a problem, for example [3]. In [4] the authors identified the most common errors made in Java. M. Hristova at al. made a list of common mistakes in Java and created the software which allowed to interpret the mistakes [5]. In [6] the authors presented the software which allowed to interpret the mistakes and enabled students to learn Java programming as well. Neil C. C. Brown at al. in their paper [7] presented the BlackBox project which has been developed since 2013. The data have been collected from users of BuleJT IDE, which is dedicated for the Java programming language. Papers [8-9] present the mistakes based on the data collected from Blackbox. The authors analyse the frequency, time to fix and the spread of errors among users and their development during a year. In [10] and [11] the authors compared the ranking of Java programming mistakes made by educators and the ranking of programming mistakes made on the basis of data collected from BlackBox. S. Hubalovsky and J. Sediy wrote about mistakes in object oriented programming [12]. The authors mention that mistakes which do not cause syntax errors are not recognised by programmers, because programs containing them seem to work properly. In [13] the authors present common mistakes in OpenMP in C/C++. Interestingly, they also point out the good practices of using OpenMP. In paper [14] S. Júnior at al. analyse a class of mistakes which may prevent students from obtaining a proper solution. The authors write that students do not often inform the teacher about their difficulties, and if they do it – they have a difficulty in defining their problem. The problem, it is argued, might be caused by the fact that students often do not understand the programming vocabulary. A. Stefik and S. Siebert found that the languages, that are closer to English may be more intuitive to beginner programmers [15]. In [16] the authors draw attention to the fact that the feedback focuses more on identifying mistakes and less on fixing the problems. To the best of the author's knowledge, research papers dedicated to mistakes in the C/C++ language are not numerous. Owing to this fact, the present analysis of the issue was conducted.

The purpose of the present paper is to review and analyse the common programming mistakes made by IT students. The analysis addressed programs of students of the Lublin University of Technology (LUT). Computer Science studies at the LUT include the following courses: "Programming in C" in the first semester, "Algorithms and Data Structures" in the second, and "Numerical Methods" in the third one. Students write programs in C/C++ during the courses "Algorithms and Data Structures" and "Numerical Methods". The methodology applied is discussed in section 2. The section includes a description of the data and a review of common programming mistakes. Section 3 contains the results and Section 4 concludes the paper.

2. Methodology

The programs of first and second year students were considered to analyse the programming mistakes in C++ language. Students wrote programs in C++ in a structural way. Students had the course of Algorithmics and Data Structures during the first year, and they had a course Numerical Methods during the second year. The students had to wrote the programs in C++ during both of courses. They had to send their programs via a dedicated platform or present their programs at the end of the classes. The mistakes which students made during the classes were also collected. The programs of the two courses mentioned of the same 43 students were analysed and their mistakes were collected.

2.1. Data

Eight programs were collected from each student during the Algorithms and Data Structures course and six programs during the Numerical Methods course. The total number of analysed files was 602.

The data which were collected during Algorithms and Data Structures concerned the following topics:

- finding array min/max,
- sorting,
- searching information in an array,
- data structures (implementing stack, queue, list or binary tree).

The data which were collected during Numerical Methods concerned the following topics:

- Newton and Lagrange interpolation,
- mean square approximation,
- searching for zero in nonlinear equations,
- numerical integration,
- solving systems of linear equations.

Students had to write a program where they had to create, for example, an array, implement an algorithm and test it.

2.2. Common programming mistakes

The classification of programming mistakes in structural C++ was defined and the programming mistakes were divided into three groups: syntax errors, memory management errors and logic errors. The first group includes errors which do not allow the program to compile. Memory management errors are related to the situations when the program wants to use a memory area which is not accessible. The third group contains logical errors which can sometimes be difficult to find, and some of them appear when somebody starts to learn programming and is not experienced enough. Each group is discussed in details below. The following list presents the most typical mistakes made in C/C++.

The first group, syntax errors:

1. No semicolon.
2. Invalid number of parameters passed to function.
3. No brackets or odd number of brackets, for example in expressions.
4. Variable names containing the space.
5. No library attached to file.
6. Visibility of variables in switch case section. Students tend to declare the variable in the case section and the program cannot be complied.

The second group contains memory management mistakes:

7. Out of array range. Students use an array and sometimes they do not remember that the first index of it is 0 and the last one is the length of array minus one. They try to refer to the element, which is out of array.
8. Trying to access the memory which was not allocated. Students declare a pointer and, for example, try to read elements and write them into an array. If a student uses the pointer variable to access an array, he/she has to allocate the memory before using it.
9. Memory leak. Students do not delete the allocated memory for example when they use the pointers, define a queue, stack or list. This kind of errors can also appear when students dynamically allocate memory in functions and do not deallocate before end of function.
10. No passing variable by reference. Students create a function where they want to change the variable value and pass this variable into the function without reference.

The third group includes logical mistakes:

11. Shadowing variable. Students create a variable, for example, in the main function and they also create a variable with the same name and type in a loop, causing the first variable to be shadowed by the second one.

12. Incorrect condition in the conditional statement or loop. Students place only one '=' sign in a condition. The incorrect condition in the conditional statement and loop as well causes the wrong operation of a program. Sometimes students also do not know when the algorithm should be ended, and they do not define the correct condition which ends the algorithm.
13. Incorrect initial value of variable. Students do not remember that the neutral element of multiplication is 1 and the neutral element of addition is 0. When searching for a min value of an array they incorrectly assume that it is sufficient to put in the initial value of min equal to some big number. They do not understand that such an idea is not flexible and it is better to assume that the first element is the min and then start comparing it with other elements.
14. Uninitialised variable. For example, when students create an array, they forget that the variable defining the length of the array has first to be initialised and only then used.
15. Dividing two integers. The results of division of two integers is not necessarily an integer.
16. No brackets in condition statement or loop. Students do not remember that if they want to place more than one instruction in a condition statement or a loop, they need to use brackets, otherwise only the first instruction will be assigned to the condition statement or loop.
17. Use of magic numbers. Use of unnamed constants the sense of which is not clear.

Listings 1, 2 and 3 show examples of mistakes made by students. Listing 1 presents the mistake of using a dynamic array without memory allocation. Listing 2 shows an incorrect condition in a conditional statement or loop. Listing 3 illustrates a mistake in finding the min or max value in an array.

Listing 1: Using an array without memory allocation

```
int* tab;
for(int i=0;i<10;i++){
 cin>>tab[i];
}
```

Listing 2: Incorrect condition in a conditional statement or loop

```
int i=5;
if(i%2=1)
  cout<<"Odd number";
else
 cout<<"Even number";
```

Listing 3: Finding min value

```
int min=999;
for(int i=0; i<5;i++){
   if(tab[i]<min){
     min=tab[i];
   }
}
```

3. Results

The results of the research are presented below. If the same mistakes appeared several times during one class, it was calculated only one time. The numbers assigned to the specific mistake types in section 2.2 were used for the annotation of the figures presented below. Number 18 on the charts is related to other mistakes which are not specified in the classification. Figure 1 presents the number of mistakes made by first year students during the "Algorithms and Data Structures" course. The maximum possible number of mistakes amounts to 344, due to the fact that 43 programs were analysed during each of the 8 classes. The most common programming mistakes are related to memory management: out of the array range, trying to access the memory which was not allocated, memory leak or no passing variable by reference. The mistake related to initialising a variable with specific value appeared often as well.

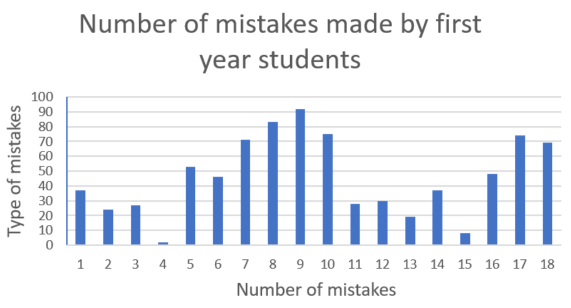

Figure 1: Number of mistakes made by first year students during the "Algorithm and Data Structures" course.

Figure 2. presents the number of mistakes made by second year students during the "Numerical Methods" course. The maximum possible number of mistakes equals 258, due to the fact that 43 programs were analyzed during each of the 6 classes. The most common programming mistakes are related to no passing variable by reference and performing calculations: no brackets or odd number of brackets, dividing two integers.

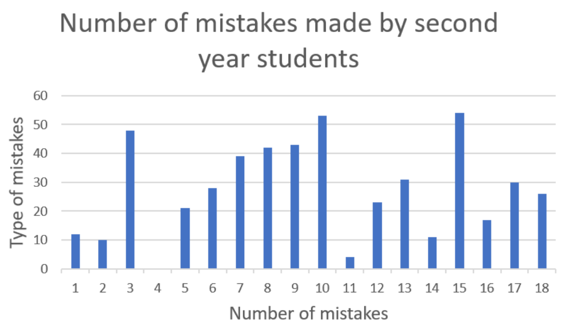

Figure 2: Number of mistakes made by second year students during the "Numerical methods" course.

Figure 3 shows a comparision between the mistakes made by first year students and the second year students during the "Algorithms and Data Structures" and "Numerical Methods" courses. It can be observed that the percent of mistakes decreased in most cases. The opposite situation was observed in the case of the following mistakes: no brackets or odd number of brackets, incorrect initial value of variable and dividing two integers. A similar percentage is noticeable for mistakes related to an incorrect condition in a conditional statement or loop for first and second year students.

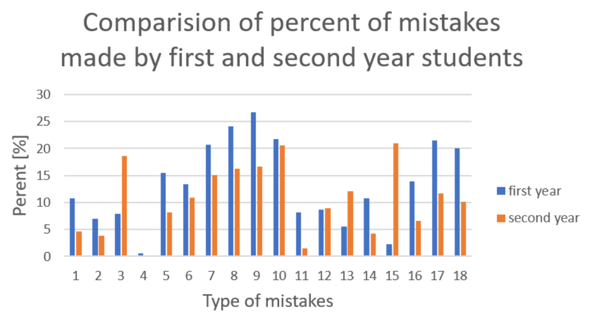

Figure 3: Comparison of percent of mistakes made by first and second year students during the "Algorithms and data structured" and" Numerical Methods" courses.

4. Conclusions

A classification of common programming mistakes in C++ and analysis of these mistakes were conducted. The obtained results show that a higher number of mistakes is made during the first year. Students have problems with syntactic errors, which are revealed during program compilation. The most problematic for students are mistakes related to memory management. This kind of mistakes could be difficult to detect for beginners. They may not be observed during each execution of a program and may require analysis of the code line by line. Second year students also have a problem with pointers. There are some mistakes which are made by second year students more often. This is caused by the situation that during the "Numerical Methods" course the students write more programs related to mathematical expressions and numbers. The mistakes related to "no brackets or odd number of brackets", "incorrect initial value of variable" or "dividing two integers" were made mostly by students of the second year. It can be observed that only one mistake related to dividing two integers constitutes about 20% of all of students errors.

The obtained results may help to adjust the design of the course and textbooks. The results also show which topics the teacher should focus on more. The article seems to be useful not only for academic teachers and students, but also for school teachers and learners. Further research will focus on a bigger group of students and the object oriented approach to programming in C++.

Literature

[1] E. Soloway, S. Iyengar, editors, Empirical Studies of Programmers: Papers Presented at the First Workshop on Empirical Studies of Programmers, Intellect Books (1986).

[2] M. C. Jadud. Methods and tools for exploring novice compilation behaviour. In Proceedings of the Second International Workshop on Computing Education Research, ICER '06, New York, NY, USA (2006) 73-84.

[3] P. Denny, A. Luxton-Reilly, E. Tempero, All syntax errors are not equal, In Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE'12, New York, NY, USA (2012) 75-80.

[4] J. Jackson, M. Cobb, C. Carver, Identifying top Java errors for novice programmers, In Frontiers in Education, 2005. FIE '05. Proceedings 35th Annual Conference (2005).

[5] M. Hristova, A. Misra, M. Rutter, R. Mercuri, Identifying and correcting Java programming errors for introductory computer science students, In Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education, SIGCSE '03, New York, NY, USA (2003) 153-156.

[6] T. Sirkiä, J. Sorva, Exploring programming misconceptions: an analysis of student mistakes in visual program simulation exercises, In Proceedings of the 12th Koli Calling International Conference on Computing Education Research (2012) 19-28.

[7] N. C. C. Brown, M. Kölling, D. McCall, I. Utting. Blackbox, A large scale repository of novice programmers' activity, In Proceedings of the 45thACM Technical Symposium on Computer Science Education, SIGCSE '14, New York, NY, USA (2014) 223-228.

[8] A. Altadmri, N. C. Brown, 37 million compilations: Investigating novice programming mistakes in large-scale student data, In Proceedings of the 46th ACM Technical Symposium on Computer Science Education (2015) 522-527.

[9] N. C. Brown, A. Altadmri, Investigating novice programming mistakes: educator beliefs vs. student data, In Proceedings of the tenth annual conference on International computing education research (2014) 43-50.

[10] A. Ahadi, R. Lister, S. Lal, A. Hellas, Learning programming, syntax errors and institution-specific factors, In Proceedings of the 20th Australasian computing education conference (2018) 90-96.

[11] N. C. Brown, A. Altadmri, Novice Java programming mistakes: Large-scale data vs. educator beliefs, ACM Transactions on Computing Education (TOCE), 17(2) (2017) 1-21.

[12] Š. Hubálovský, J. Šedivý, Mistakes in object oriented programming, Zeszyty Naukowe Wydziału ETI Politechniki Gdańskiej, Technologie Informacyjne, 18, (2010) 205-210.

[13] M. Süß, C. Leopold, Common mistakes in OpenMP and how to avoid them, In International Workshop on OpenMP, Springer, Berlin, Heidelberg (2005) 312-323.

[14] A. S. Júnior, J. C. A. de Figueiredo, D. Serey, Analyzing the Impact of Programming Mistakes on Students' Programming Abilities, In Brazilian Symposium on Computers in Education, vol. 30, No. 1, (2019) 369.

[15] S. Andreas, S. Susanna, An Empirical Investigation into Programming Language Syntax. Trans. Comput. Educ. 13, 4, Article 19 (2013).

[16] H. Keuning, J. Jeuring, B. Heeren, A systematic literature review of automated feedback generation for programming exercises. ACM Transactions on Computing Education (TOCE), 19(1) (2018) 1-43.