A Three-level Problem Corpus for Introductory Computer Programming Courses

Weijun Chen, University of Tsinghua, China

The Asian Conference on Education 2017
Official Conference Proceedings

Abstract
Solving problems is an important exercising activity for students in an introductory computer programming course. The paper introduces the construction of a three-level problem corpus for training the students' programming skills. The first level consists of the programming language syntax problems, including data types, control flow, functions, pointers and arrays, structures, input and output, etc. For each syntax point, there are several problems of different difficulties and each problem is in the form of single-choice question. The second level consists of the programming problems. For each problem, a text description is given and the students are required to write the corresponding source code. The purpose of this part is to train the skills of problem analysis, algorithm design and coding. Besides, each problem is provided with a couple of reference source codes which were written by experienced programmers, the students can learn how to program by studying these good examples. The third level consists of many programming projects that are collected from the Internet and our previous courses. These projects are relatively bigger programming problems such as the Tetris game. For each project, the full source codes and detailed documents are given. The students can learn how to implement a practical software by reviewing these projects, they can also modify the projects and add some new features. The above corpus is already used in our programming course at Tsinghua University. Initial observations show that students improved their programming skills after they have completed those problems in the corpus.

Keywords: programming, problem corpus, syntax, algorithm

1. Introduction

Introduction to Computer Programming is an introductory programming course for non-computer science undergraduate students at Tsinghua University, China. It is an optional course for anyone that is interested in studying computer programming. The course focuses on common computational problem solving techniques and no prior programming experience is assumed. The course is given in one semester of sixteen weeks, and in each week, there is one two-hour lecture and one three-hour lab session. Generally there are about 150 students in the class and the author of this paper is one of the lectures of the course.

After teaching the course for several years, we find that many students are not able to study the course well. They may be good at studying the textbook or learning the syntax of the C language. But they are not good at solving programming problems and writing solid codes. A common situation is that a student is quite familiar with the details of the syntax such as the different usage methods of ++ operator and the order of evaluation of operators. However, when he is given a practical programming problem, he will have no idea how to solve it. In fact, many students find programming to be difficult and disheartening.

Previous research has shown that the skills required for computer programming are problem solving and analytical skills (Riley, 1981; Henderson, 1986; Maheshwari, 1997; Linn & Clancy, 1992). To make the students to obtain those skills, the course should provide them with an effective mechanism to practice programming beside the traditional classroom lectures.

Currently, the students will have an assignment of four or five programming problems for each week. They will complete the assignment in the three-hour lab session and several teacher assistants will be there to help them. The students' source codes are submitted to a online judge website whose core facility is automatic assessment of programming assignments, which means that students can receive the feedbacks immediately after they have submitted their assignments. Therefore, the students can interact with the system for a couple of rounds if he fails to submit a correct solution at the first time. By this way, the students can finish their assignments efficiently, they don't have to wait a couple of days or a week for the result.

However, there are still several drawbacks exist:

Firstly, although the online judge system is a powerful tool, it may be still inconvenient for some students, especially those who have few (if not zero) prior programming experience before taking this course. They will face a lot of difficulties when they use the system in the beginning. For example, since many students are not familiar with the syntax of the programming language, when they submit their solutions, the compiler even fails to compile these source files. And sometimes, these files can get compiled successfully, however, there are still exist some bugs which make the program run incorrectly. In that case, the system will provide no help and the teacher assistants are required to help the students to spot and correct the errors in the source code.

Secondly, if one want to learn computer programming, he should spend enough time on doing exercises after the classes. Currently, there are only four or five programming assignment problems for each week, and it generally takes the students two or three hours to finish the assignment. This can make the students review the content of the classroom lectures and understand the basic principle of programming. However, it is far from enough if one want to become a professional programmer. Therefore, the course should provide the students with more resources if they are willing to learn more about programming.

Thirdly, the online judge system employed currently is just an evaluation software, it can compile and run the source code submitted by students automatically and suggest a score according to the result. However, it is just a kind of software tool instead of course resources. The students are not able to learn how to program by using the software only.

Finally, the online system can only judge whether a program is correct or wrong, however it can not judge whether the program is good or bad. When the students are doing their homework, they are usually doing by themselves. However, since they have few prior programming experience, the corresponding result is that many source code they submit may be correct in functionality but are written poorly. These programs will occupy too much memory and computational time, which are useless in practical software projects. As mentioned earlier, the current online system is not able to solve this problem but it lacks enough information and intelligence to judge whether a program is good or bad and provides suggestions and help the students to improve their source code. If the students ask for help from TAs, they will get very good suggestions and solve the problem. However the TAs are not available 24 hours a day, you can't find a TA whenever you encounter a programming problem.

For the reasons mentioned above, the author of this paper think it is necessary to reform the homework assignments of programming courses and construct a three-level problem corpus for training the students' programming skills.

2. Basic Idea

The basic idea is to construct a problem corpus for programming courses, it consists of necessary software tools as well as a large amount of course resources. The construction of the system is based on the characteristics and learning patterns of programming courses and it is designed for new students. They can begin to study programming from zero background knowledge with the help of the system and gradually improve their skills of problem solving and coding.

After teaching relative courses for several years, we summarize the characteristics and learning patterns of programming courses as follows:

Firstly, for computer programming courses, their homework should be completed in an online training system. Therefore, the students can study programming 24 hours a day, 7 days a week. They can study at any time and any place, they can also receive the feedbacks from the system soon after they submit their source code.

Secondly, there are should be enough number of problems in the corpus. There is an ancient Chinese saying: if you read aloud more than three hundreds of poems of Tang Dynasty again and again, you will learn to write your own poems. This ancient saying tells us that we need to solve enough number of problems if we want to learn how to program. In fact, the skills of computer programming are developed from experienced practices.

Thirdly, the students can improve their programming skills by doing homework by themselves, however, that is not enough. We should provide the students with a positive feedback learning mechanism which means that they can study good examples of source code written by experienced programmers. As we know, the poems of Tang Dynasty are the best ancient poems in Chinese history, if we study and read aloud these good examples again and again, we will get familiar with poetry and literature very soon.

Finally, there are several different stages for a beginner to become a trained programmer. At the first stage, they will study the programming language syntax and learn how to write solid source codes. At the second stage, they will learn the skills of algorithm design and problem solving. At the third stage, they will learn how to accomplish a practical software project. These stages need different type of knowledge and skills, start from easy and end to hard. Therefore, we need to design a different type of training scheme for each stage.

3. Design and Implementation

In this paper, we proposed a complete and systematic afterclass training schema for improving the students' programming skills. It consists of a three-level problem corpus and corresponding software tools.

3.1 Syntax Level

The syntax level indicates the syntax knowledge of a programming language such as C or Java. Most of the students that take this course have very few prior programming experience, they will meet many different kinds of difficulties when entering this completely new area. For example, some students are even lack of the necessary skills to use common software applications because they used to have few time to use computers under the pressure of the university entrance exams. Therefore, the purpose of the syntax level is to help the students review and consolidate the syntax knowledge they have learned. We need to build a corresponding syntax problem corpus that should cover different aspects of a programming language. For each syntax point of the language, several different types of problems are required. They are chosen from different angles and from easy to hard. The corpus is not just a problem corpus, it is also a knowledge base. Most of the problems are in the form of selection questions or fill-in-the-blank questions.

The problems in the corpus are organized according to the structures of the course which are summarized as follows:

- Data types and expressions: integers, float type, double type, char type, type conversion, variables, constants, expressions and operators.
- Simple control structures: sequencing, input and output, the relation operators, the logical operators, the if statement, the switch statement.
- Iteration: the for loop, the while loop, the break and continue statements.
- Arrays: one-dimensional arrays, two-dimensional arrays, character strings
- Functional abstraction: definition and declaration of a function, parameter passing, return values, scope of variables, stack frames
- Pointers: pointer variables, & and * operators, pointers as parameters, pointer and array, dynamic arrays, pointer arrays, pointer to pointer.
- Basic data structures: structs, struct arrays, -> operator, structs as parameters, linear list.

For each syntax point, there will be several different problems. These problems had been collected one by one for several years. They are mainly from the exercises and exams of our course or other similar courses. For each problem, there will be problem specification, reference answer, several keywords (used for further search) and the related syntax points. Each problem is also labeled with a degree of difficulty (1-5) where 1 means the simplest and 5 means the hardest.

For example, in the Chapter 1 “Data types and expressions”, there is a syntax point “data overflow”. The follow question will be arranged when reviewing that syntax point.

```
#include  <stdio.h>
void main( )
{
    short  x = 32767;

    x = x + 1;
    printf("%d", x);
}
```
The output is: _____

Some students may think that it is very easy and the answer should be 32768. However this is not the truth because the variable x will overflow.

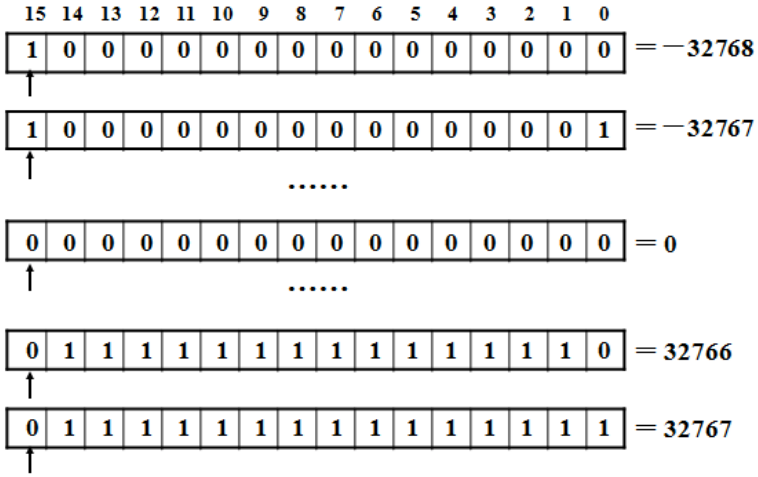

Figure 1: Data overflow

In the corpus we will incorporate Figure 1 to tell the students that why the answer is not 32768. As we know, all numbers are encoded in binaries in a computer and the length of a binary is fixed such as 16 bits. Therefore, 32767 is the maximum number that a 16-bit binary can hold. If you add 1 to 32767, a data overflow will occur and you will get -32768 instead of 32768.

3.2 Programming Level

The programming level indicates problem analysis and solving that are the core skills required for computer programming. When a student is given a programming problem, he should be able to fulfill the problem analysis, design the corresponding algorithm and write the correct source code. These are abstract mental skills that are harder to learn than the language syntax.

At this level, we constructed another problem corpus. All the problems are required to write out the correct source code to accomplish the tasks. The purpose here is not for reviewing the language syntax, but for training and improving the students' programming skills. Each problem is chosen carefully because we want to cover as many of programming patterns as possible.

The problems in the corpus are not organized according to the structure of the course because there are always several different types of syntax points occur in one problem. It is an integrated application of different knowledge sources. For each problem, a number degree of difficulty is labeled, 1 means the simplest and 5 means the hardest. When the students are doing exercises in the corpus, they should do them step by step. We recommend that they begin with problems of degree 1. After they have done all the problems of degree 1, they can move to problems of degree 2, etc.

For each programming problem, there are problem specification, knowledge points involved and the programming patterns that may be used. Since the source codes submitted by the students are judged automatically by the computer system itself, we need to design and prepare a set of test cases for each problem. Generally there will be 10 test cases and each test case is worthy of ten points (the total score of each problem is 100).

As mentioned before, a corpus should contain enough number of problems to be more effective in use. To solve this problem, we tried to collect as many programming problems as possible from different sources. Some are chosen from the homework assignments and exam problems of our own course, some are from the similar courses of other universities and some are from the problems of different kinds of competitions such as ACM ICPC (International Collegiate Programming Contest). Currently we have already collected about 150 problems.

As mentioned before, the students can gradually improve their programming skills by doing homework by themselves, however, that is not enough. We should provide the students with a positive feedback learning mechanism. One will not become an excellent programmer if he does not know what an excellent program is. To solve this problem, we prepared several good examples of source code written by experienced programmers for each problem. The students can study these good examples and learn how to write solid code quickly.

Now we will demonstrate the construction of the problem corpus with an example. It consists of five steps:

- Step 1: Find an appropriate programming problem and write out its problem specification. For example, we find a problem titled “Caesar Cypher”, the corresponding problem specification is “One of the earliest encrypting systems is attributed to Julius Caesar: if the letter to be encrypted is the Nth letter in the alphabet, replace it with the (N+K)th where K is some fixed integer. We usually treat a space as zero and all arithmetic is then done modulo 27. Thus for K = 1 the message 'ATTACK AT DAWN' becomes 'BUUBDLABUAEBXO'. Decrypting such a message is trivial since one only needs to try 26 different values of K. This process is aided by knowledge of the language, since then one can determine when the decrypted text forms recognizable words. If one does not know the language, then a dictionary would be necessary. Write a program that will read in a dictionary and some encrypted text, determine the value of K that was used, and then decrypt the cyphertext to produce the original message. The original message contained only letters and spaces and has been encrypted using the above method. The most suitable value of K will be the one which produces the most matches with the words in the dictionary”.
- Step 2: We will analysis the problem specification and label it with a degree of difficulty. For example, the problem “Caesar Cypher” will be labeled with 4 which means it is very difficult but it is still not the most difficult one.
- Step 3: Design 10 or 20 test cases for this problem. For the “Caesar Cypher” problem, we will use different types of sentences, for example, long sentences, short sentences, sentences with long words, sentences with short words, sentences whose word number is even, sentences whose word number is odd, etc. The value of K also varies each time. Each test case consists of two text files: an input file and an output file.
- Step 4: Write comments on the problem, including the knowledge points and programming patterns it involves. For the “Caesar Cypher” problem, the syntax points include “char type” and “string access”. The programming patterns include “array element searching”, “value addition” and “find maximum/minimum value”.

- Step 5: Prepare good examples of source code for the problem. Some of these examples are written by the TAs and some are chosen carefully from the assignments of previous semesters.

3.3 Software Development Level

Software development is the ability to create a practical software application. It needs strong programming skills, as well as some software engineering techniques such as requirement analysis, system design, coding, testing and management.

At this level, we constructed a programming project corpus. These projects are relatively bigger open-source programming problems such as the Tetris game. For each project, the full source codes and detailed documents are given. The students can learn how to implement a practical software by reviewing these projects, they can also modify the projects and add some new features.

The construction of project corpus is time-consuming because we need to collect them one by one and not every software is suitable for our educational purpose. Currently these projects are mainly collected from the Internet and our previous courses.

The Internet is the largest public resource repository in the world, everybody can find what he want on the Internet. As far as our corpus is concerned, the project resource we find should satisfy the following conditions:

- The project should be free and open to educational use.
- The project should be open-source and the complete source code is downloadable.
- The project should have detailed documents.
- The size of the project is modest, not too big and not too small.
- The programming language of the project is C.

After finding an appropriate project resource, we will examine and improve it. Firstly, we will compile the whole source code to generate the corresponding executable file. If the compilation fail, we will debug the source code and fix it. Secondly, we will organize and improve the project's documents including comments in the source code, design document, specification document, etc. The final purpose is to make it become a runnable, readable and learnable educational resource.

4. Conclusion

In this paper, we constructed a three-level problem corpus for training the students' programming skills in an introductory computer programming course. The corpus is already completed. At the syntax level, we built a syntax knowledge problem corpus that covers different aspects of the C programming language, each problem is in the form of selection or fill-in-the-blank question. Currently there are totally 150 problems in the corpus. At the programming level, we constructed another problem corpus for training and improving the students' programming skills. For each problem, there are problem specification, comments on knowledge points and programming patterns that may be used, 10-20 test cases for online judgment and at least one piece of reference source code. The students are required to write out the correct source code to accomplish the task. Currently there are already 150 carefully chosen

programming problems and in the future, we will add more problems and update the reference source code of each problem. At the software development level, we constructed a programming project corpus. Each project is a relatively bigger open-source programming problem and consists of the full source code and detailed documents. Currently there are totally 45 different types of projects in the corpus, including many famous games such as the Tetris, the Battle City, the Pac-Man, etc. We have already used the above resource in our programming course at Tsinghua University. Initial observations show that students improved their programming skills after they have completed those problems in the corpus.

References

Henderson, P. B.(1986). Anatomy of an introductory computer science course. Proceedings of the 17th SIGCSE ’86: Technical symposium on Computer Science Education, pp. 257-263, New York: ACM Press.

Linn M. C. & Clancy M. J. (1992). The case for case studies of programming problems. Communications of the ACM, 35(3), 121-132.

Maheshwari, P. (1997). Teaching programming paradigms and languages for qualitative learning. Proceedings from ACM International Conference Proceeding Series ’97: Proceedings of the second Australasian Conference on Computer Science Education, pp. 32-39, New York: ACM Press.

Riley, D. (1981). Proceedings from Technical Symposium on Computer Science Education ’81: Proceedings of the twelfth SIGCSE Technical Symposium on Computer Science Education, pp. 244-251, New York: ACM Press.

Contact email: cwj@tsinghua.edu.cn