Test System and Software for Evaluation of Students' Knowledge of Programming

Nikolay Kirov

New Bulgarian University, Sofia, Bulgaria

***Abstract**: The article describes a test system for assessing students' knowledge of programming. It combines the evaluation with elements of the training process and inquiry-based approach in education. Software for the generation, verification and analysis of tests is also presented.*

***Keywords:** Multiple-choice test, software.*

1. INTRODUCTION

The idea of making a test system for assessing students' knowledge in programming arose in 1998, when I started to teach programming in Pascal to students from South-West University "Neofit Rilski" [4]. Then I wrote the first version of test generator. Since then I use this system in my courses in programming, object-oriented programming and data structures. The test system is also applied to courses in operating systems and advanced programming.

In terms of students, the test consists of three stages: preparation for the test, making the test, and verifying the results. The teacher must create a test, generate the required number of copies for the students, check the completed tests and analyze the results of the students.

2. SPECIAL MULTIPLE CHOICE TEST

The test presented here is of a multiple-choice type with some peculiarities with respect to the classic multiple-choice. It is on paper and is held within 1.5 astronomical hours.

2.1. Terminology

The test consists of items. Stem is the introductory question or statement at the beginning of each item. The stem is followed by the options. The options are: answers – the correct options, and distractors – the incorrect options. The items are stored in an item bank. Individual test consists of fixed number of items with fixed number of options.

2.2. Description

The individual tests are generated from the test bank randomly. The number of options in any individual test is fixed on 4 for all items. In any individual test the number of correct options (answers) of an item may be any number in the interval [0,4]. The student has to identify each answer and each distractor. For any option he/she has 3 choices of response:

- yes, i.e. I know the option is an answer;
- no, i.e. I know the option is a distractor;
- nothing, i.e. I do not know whether the option is an answer or a distractor.

Any correct response (yes or no) adds one point in the total score but any incorrect (opposite) response subtracts one point (a penalty point) from the total score.

We estimate an individual test of M items with x points total score as follows:

```
int p = ceil(100.0*x/(M*4)), e = 2;
     if (p >= 90) e = 6;
else if (p >= 76) e = 5;
else if (p >= 60) e = 4;
else if (p >= 50) e = 3;
```

At least one week before the date of the test all original stems and two example options per item (an answer and a distractor) are published online on the course website. A few days before the test day, a seminar is conducted where students discuss the published items and options, as well as other possible proposals for options.

During the completion of the test the students can use lectures, textbooks and any other printed materials. They are allowed also to use computer as a book, or also compiler, or even Internet. Anyone can ask questions about ambiguities in the test.

After the verification of the tests the individual tests and responses are returned to the students. Each student should carefully check his/her individual test in order to determine whether he/she agrees with the noted errors. If something is not clear she/he can discuss the case. It is normal to increase the total score of the student if his/her arguments about the case are reasonable.

Arguments for this way of doing the test can be found at [1] and [2].

2.3. Inquiry-Based Approach

According to a definition by Linn, Davis and Bell [3], inquiry is the intentional process of diagnosing problems, critiquing experiments, distinguishing alternatives, planning investigations, researching conjectures,

searching for information, constructing models, debating with peers, and forming coherent arguments. The elements of this definition can be found in the described test system. The table shows the presence of the corresponding element of the definition in the three phases of the test: preparation (before), conducting (during) and verification (after).

Tab. 1: Elements of inquiry-based education in the test

Inquiry	Test	Before	During	After
diagnosing problems	understanding the item	+	+	+
critiquing experiments	using computer	+	+	-
distinguishing alternatives	comparing options	-	+	+
planning investigations	how to search	+	+	-
researching conjectures	yes, no, nothing	+	+	-
searching for information	searching	+	+	-
constructing models	programming	+	-	-
debating with peers	discussing	+	-	+
forming coherent arguments	understanding	-	-	+

The second column of the table gives the meaning of the elements of the definition in our case. A plus sign in the other columns means that this feature significantly present at the corresponding stage of the test.

3. SOFTWARE

The developed software allows the teacher easily to prepare a test bank, to generate the required number of individual tests, to upload the students' responses and to make analysis of the results for each item of the test bank.

3.1. Test generator

Preparation of the test begins with selecting items – stems and options – and put them into an item bank. At least 10 items each having at least 5-6 possible answers should be completed and stored as a text file in a particular format. This file is the input to test generator. The test generator generates individual tests using random distribution of both items and their options. Each individual test consists of 10-20 items with 4 options (a, b, c, d). The output plain text file (out.tex) contains all individual tests. The second output file (tab.tex) is a table for checking the individual tests. The third file (data.tex) is a copy of the input file with additional data for the generated individual tests. All output files are in LATEX format and have fixed as the above names.

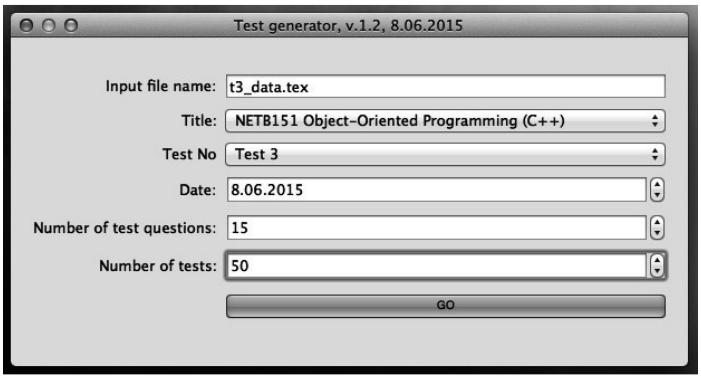

Fig. 1: The user interface of test generator.

3.2. Test checker

Checking the answers of the individual tests can be carried out manually – using the table, generated by test generator, or automatically – by test checker. Uploading data of the students' responses can be done manually – using the user interface of test checker or automatically – using a special template (on paper) for the students' responses and scanner.

The program creates a text file (save.txt), containing verified tests and can upload this file.

After entering the students' responses, test checker gives the results – for each option of each item in the test bank calculates two sets of numbers. The set $X = \{x, x_1, x_2, x_3\}$ represents all the individual tests and the set $Y = \{y, y_1, y_2, y_3\}$ represents the individual tests of students, which pass the test ($e > 2$).

- x, y – the number of individual tests which contain the corresponding item and four of its options;
- x_1, y_1 – the number of tests without response;
- x_2, y_2 – the number of tests with correct response;
- x_3, y_3 – the number of tests with incorrect (opposite) response.

The names of the output files of test checker are data_result.txt and data_result1.txt. The format of the second file is suitable for input in spreadsheet.

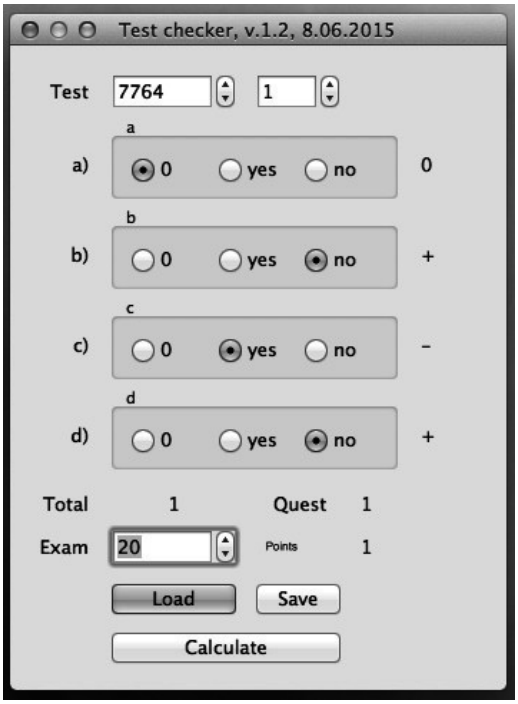

Fig. 1: The user interface of test checker.

4. CONCLUSIONS

The software is written in C++ using Qt – cross-platform application and UI development framework [5]. It is publicly available and open source (see [6] and [7]). The software tools save a lot of time and efforts of the teacher for the preparation and verification the test and evaluation the test results.

Only presented test system is not sufficient for assessing the students' knowledge of programming. However, it is an essential element of a comprehensive evaluation system that includes examination of the ability to write programs as homework and classwork [1].

5. REFERENCES

[1] Kirov, N. (2013) A System for Assessing the Knowledge and Skills of Students in Computer Programming, In: Proceedings of the 9th Annual International Conference on Computer Science and Education in Computer Science, 29-30 June 2013 in Fulda, and 1-2 July 2013 in Wurzburg, Germany, 109-112.

[2] Kirov, N. (2014) A test system for checking and evaluation the students' programming knowledge, In: Proceedings of the 10th Annual International Conference on Computer Science and Education in Computer Science, 4–7 July 2014, Albena, Bulgaria, 125-134.
[3] Linn, M. C., Davis, E. A., & Bell. P. (2004). Inquiry and Technology. In M.C. Linn, E.A. Davis, & P. Bell (Eds.), Internet Environments for Science Education (pp. 3-28). Mahwah, NJ: Lawrence Erlbaum Associates.
[4] Programming and data structures (in Bulgarian). nikolay.kirov.be/swu/i2_bg.html
[5] Qt. www.qt.io
[6] Test checker. github.com/nkirov/tests_checker
[7] Test generator. github.com/nkirov/tests_generator