EDUCATIONAL SOFTWARE APPLICATION FOR THE STUDY OF THE C PROGRAMMING LANGUAGE

Alexandru ENE
FECC, University of Pitesti, Romania
alexandru.ene@upit.ro

Keywords: educational software, adaptive testing, modular programming

Abstract: *This paper presents a C++ application for the study of the way of defining function header and calling functions, in the C language. The study is based on grid tests, that are random generated. The advantage of the automatic generation for the tests is that there is no longer need of a teacher to manually edit these tests. Also, each time we use this program, we shall experiment new tests. So, no one could learn by heart the correct results of tests (this could happen for the manually edited tests). The program allows also the possibility to generate new tests, taking into account the current mistakes made by the user, thus improving the efficacy of the educational process, so the C programming language could be better and faster learned by the user.*

1. INTRODUCTION

The use of software in education and training dates from the early 1940's. Some major types of educational software are: courseware, classroom aids, assessment software, reference software [1].

The application that is described in this paper is an educational software to be used as a classroom aid. The main objective of this software application is to help the user to learn better and faster the way of defining function header and the way of calling functions in the C language. The typical user of this application is the beginner student who studies the C programming language. That is why this software can be used in the software learning laboratory, when the students study the topic of modularization in the C language ([2],[3]). It is very important for a beginner who studies a programming language to understand how to write modular programs.

Modular programming is a must, because:

-the general program built from many programmer written functions is easier to debug and easier to be understood.

-each module can be tested independent of the other modules

-a completed module can be reused in other software projects (thus increasing work productivity)

-a modular written program can be developed in a team.

-when there are some activities that are repeated in the program, if we use a function that is repeatedly called, the program becomes shorter (less lines of code).

The software application written for the study of C language is a typical e-learning application.

The study is based on grid tests, that are random generated. The advantage of the automatic generation for the tests is that there is no longer need of a teacher to manually edit these tests. Also, each time we use this program, we shall experiment new tests. So, no one could learn by heart the correct results of tests (this could happen for the manually edited tests). The program allows also the possibility to generate new tests taking into account the current mistakes made by the user, thus improving the efficiency of the educational process. At the moment, three classes of mistakes are used for the generation

of a new test, as explained in the next paragraph.

The program is written in the C++ programming language.

2. DESCRIPTION OF THE PROGRAM

The program has four modules (menus). The first module is for the testing of function header definition. This menu has two submenus.

In **the first submenu**, the answer of the user is by selecting the correct header definition from a total of four alternatives. Only one alternative can be selected.

The whole application uses the generic name of **f** for the required function. The random generated list of parameters for the function f could have 0, 1 or 2 parameters, each having one type from the basic data types in the C language.

In addition to the work presented in [4], each parameter could be an input parameter or an output parameter, passed by reference to the function. Here are two examples of random generated questions:

Example 1:

Which is the header definition of a function f, which:

-returns no value

-has two parameters:

a input parameter of int type

b input parameter of int type

And the alternatives:

1. *int f(int a, int b)*
2. *void f(int a, int b)*
3. *void f(double a, double b)*
4. *void f(int& a, int & b)*

Example 2:

Which is the header definition of a function f, which:

-returns an int value

-has two parameters:

a input parameter of int type

b output parameter of double type

And the alternatives:

1. *int f(int a, double& b)*
2. *void f(int a, double& b)*
3. *int f(int& a, double b)*
4. *int f(double a, int& b)*

With the exception of the correct answer, the other three wrong alternatives are automatically generated. They put into evidence three classes of mistakes:

-mistake for the returned type.

Example:

int f(…) instead of *void f(…)*

This is the case for the first alternative of example 1.

Or:

void f(…) instead of *int f(…)*

This is the case for the second alternative of example 2.

-mistake for the type of input parameters

Example:

void f(double a, double b) instead of

void f(int a, int b)

This is the case for the third alternative from example 1.

Or:

int f(double a, int& b) instead of

int f(int a, double& b)

This is the case for the fourth alternative of example 2.

-mistake about input parameters vs output parameters.

Example:

void f(int& a, int& b) instead of

void f(int a, int b)

This is the case for the fourth alternative from example 1.

Or:

int f(int& a, double b) instead of

int f(int a, double& b)

This is the case for the third alternative of example 2.

All this three types of mistakes (through wrong alternatives selected by the user) are used for the automatic generation of new mistake based tests. But this is explained for the third module of the application.

As could be noticed from the shown examples, the text of each question has a fixed part and a dynamic generated part (which changes randomly, from question to question). In the following example, the dynamic generated part in the text of the question is shown with **bold** style:

Which is the header definition of a function f, which:
*-returns **an int** value*
*-has **two** parameters:*
*a **input** parameter of **int** type*
*b **output** parameter of **double** type*

Each variable (dynamic) part of the question is generated automatically using random numbers.
For example, the field ***input*** parameter could have only one other value : ***output*** parameter. So, using the C language instruction: *random(2)*
the two values of this field, are easily computed.
After the user selects his answer, the software program tells him if the answer is correct or not,
and which is the correct answer.

The second submenu of the function header definition testing allows the user to write himself the answer to the question.
Example:
Which is the header definition of a function f, which:
-returns no value
-has one parameter:
a output parameter of char type

And the user has to introduce on the next line, his answer. For example:
void f(char& a)
After entering his answer, the program tells him if his answer is correct or not, and which is the correct answer.

The second menu of the application deals with the tests for the modality of function calling in C language.
Example 1:
How do you call a function f, which:
-returns an int value
-has two parameters:
a input parameter of int type
b output parameter of double type
given the following sequence of a program:
int rezultat;
int a=3;
double b;

Your answer is:

And the user has to answer on the next line of the screen:
rezultat=f(a,b);
or:
rezultat=f(3,b);
Example 2:
How do you call a function f, which:
-returns no value
-has two parameters:
a input parameter of int type
b input parameter of double type
given the following sequence of a program:
int a=5;
double b=1.5;

Your answer is:

And the user has to answer on the next line of the screen:
f(a,b);
or one of the other 3 possible alternatives:
f(5,b);
f(5,1.5);
f(a,1.5);

After entering his answer, it is presented to him the correct answer.
As could be noticed from the shown examples, the text of each question has a fixed part and a dynamic generated part (which changes randomly, from question to question). In the following example, the dynamic generated part in the text of the question is shown with **bold** font:
How do you call a function f, which:
*-returns **no** value*
*-has **two** parameters:*
*a **input** parameter of **int** type*
*b **input** parameter of **double** type*
given the following sequence of a program:
int a=5;
double b=1.5;

Your answer is:

The third menu of the application deals with the generation of new tests based on the mistakes made by the user. For the moment only the mistakes made in the selection of the correct header function

definition are considered. These mistakes are divided for the current application in three classes:

-mistakes for the returned type.

-mistakes for the type of input parameters

-mistakes about input parameters vs output parameters.

For example if a user has more mistakes of the third type, all the questions will be for functions that return no value and have one or two parameters. There will be not generated questions for 0- parameters functions,

If a user has more mistakes of the first type (about the returned value) all the questions will be generated with 0- parameters, in other to be focused on the returned value.

The fourth menu of the application is an online help about the C language syntax of defining and calling functions. It deals with returned type, input parameters and output parameters , their data type and also the way of calling functions. It has examples and there are also some examples with common mistakes that are done when defining or calling C language functions.

3. CONCLUSIONS

In this paper is presented an educational application for the study of C language modularization (how functions written by the programmer are defined or called by other functions).

The typical user of this application is the beginner student who studies the C programming language. That is why this software can be used in the software learning laboratory, when the students study the topic of modularization in the C language.

There are four menus in the application:

-testing of function header definition (the answer of the user is by selecting the correct header definition from a total of four alternatives or by writing manually the requested header definition)

-testing of function calling in C language. In this case the user does not choose the answer from more alternatives. He has to write himself the correct answer.

-testing of the function header definition, based on the mistakes done by the user when he selected a choice of answer, in the first menu of the application.

-a help about the C language syntax of defining and calling functions. This help has also examples with the typical mistakes done by a beginner in C language.

This program can also be used for the automated generation of tests that are developed by a teacher that examines the students ability to call C programming language functions.

4. REFERENCES:

[1] www.wikipedia.org – Educational software,m Accesssed on:20/05.2012

[2] ALEXANDRU ENE, COSMIN STIRBU, Programarea calculatoarelor si limbaje de programare. Teorie si aplicatii, Editura Universitatii din Pitesti, 2005

[3] ALEXANDRU ENE, PETRE ANGHELESCU, Structuri de date si algoritmi. Lucrari de laborator, Editura Universitatii din Pitesti, 2009

[4] ALEXANDRU ENE, IOANA-CRISTINA ROBEA, Client- server application for testing and teaching modular writing programs, Scientific Bulletin of the University of Pitesti, Faculty of Electronics, Communications and Computers,Electronics and Computer Science series,Vol 10/issue 1/2010, ISSN 1453-1119