Technical University of Moldova

Faculty of Computers, Informatics and Microelectronics

Software Engineering and Automation Department

Software Testing

Laboratory work #3

Black Box technique

Author:
Vasile Drumea

Supervisor:
Mariana Catruc

Chișinău 2019

1 Purpose of the laboratory work

– Obtaining testing skills for the functionalities of a software;

– Forming abilities of partitioning input data in equivalence classes;

– Using decision tables and state/transition diagram to create test cases;

2 Laboratory Work Requirements

– Elaborate a scenario of testing for 2-3 functionalities of an application;

– Determine organizational criteria of equivalence classes and decision tables;

– Develop a conclusive test according to the criteria including boundary testing;

– Emphasize test cases on which you can get error results;

– Make a report of the laboratory work;

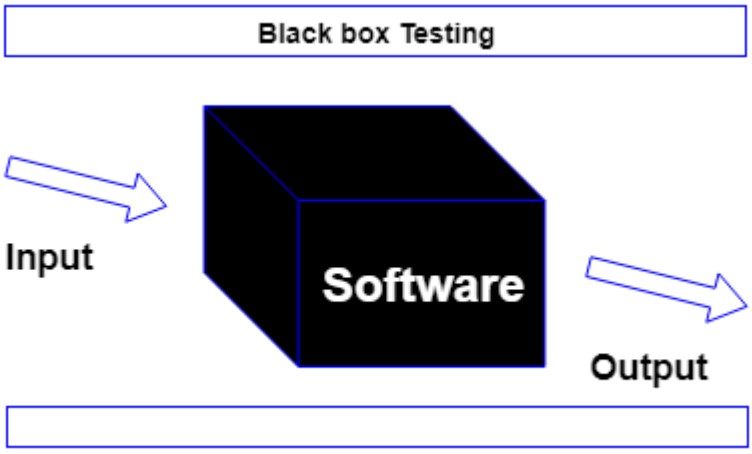
3 Intro to the technique

In black-box testing, the purpose is to the test the different functionalities of an application without looking into the internal structure of the program.

There are different strategies to use in order to test efficiently.

For example equivalence class partitioning (ECP), decision tables and boundary-value analysis (BVA).

A graphical representation is in the following figure :



4 Laboratory work implementation

4.1 Task 1 - Scenario of testing

I've chosen to test the on online application for calculating integrals which you can find at the link IntCalc.

The scenario for testing the 3 functionalities is as follows :

- Verify that the input fields are available to insert data and the submit button works as expected;
- Check if the output gives the solution for an example;
- See if the **Options** menu permits us to change the parameters;

4.2 Task 2 - Organizational criteria of EC and DT

The organizational criteria will be the type of input expression and options we introduce.
This way I've obtained the following partitions :

- Valid mathematical expressions and options;
- Non-valid expression/Non-valid characters but valid options;
- Non-valid expression/options;

The decision table :

Fields\Cases	**Case 1**	**Case 2**	**Case 3**	**Case 4**
Expression	Valid	Valid	Non-valid	Non-valid
Options	Ok	Not Ok	Ok	Not Ok
Output(Res/Err)	Res	Err	Err	Err

4.3 Task 3 - Conclusive tests

Test cases and their description :

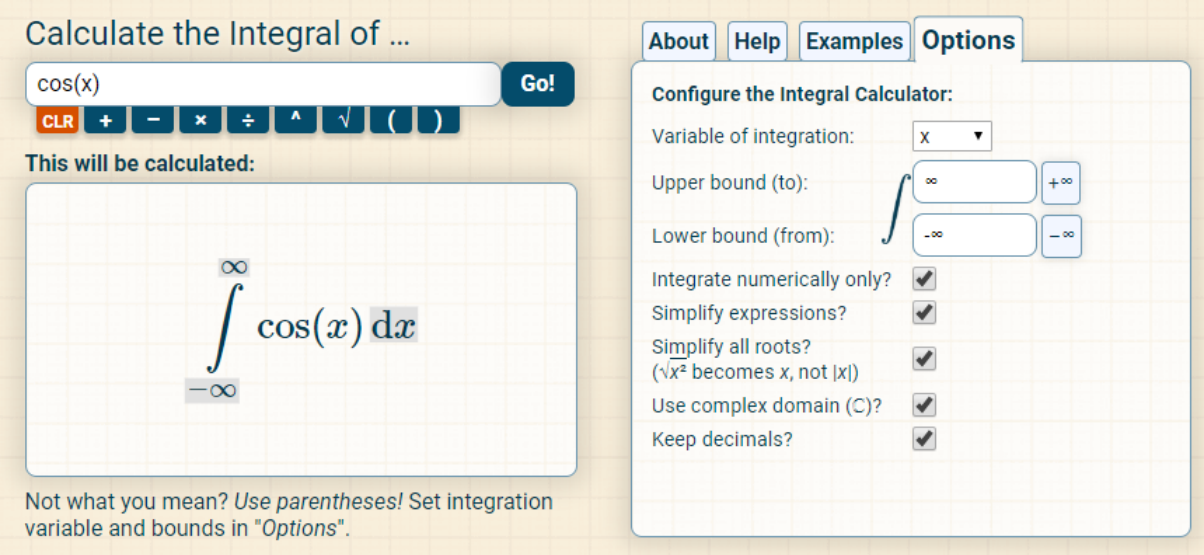
- Case 1 - A valid mathematical expression and options;
  Example : $cos(x)$;
  Expected : $sin(x) + C$;

- Case 2 - A valid expression but non-valid options;
  Example : Upper bound = A russian letter;
  Expected : Invalid character;

– Case 3 - A non-valid mathematical expression, but valid options;
Example : $cos(x)/0$;
Expected : Error;

– Case 4 - A non-valid expression and non-valid options;
Example : Russian chars at the options and input expression;
Expected : Error;

Boundary testing :
I've introduced the max and min values for Upper and Lower bounds:



Results for the test cases :

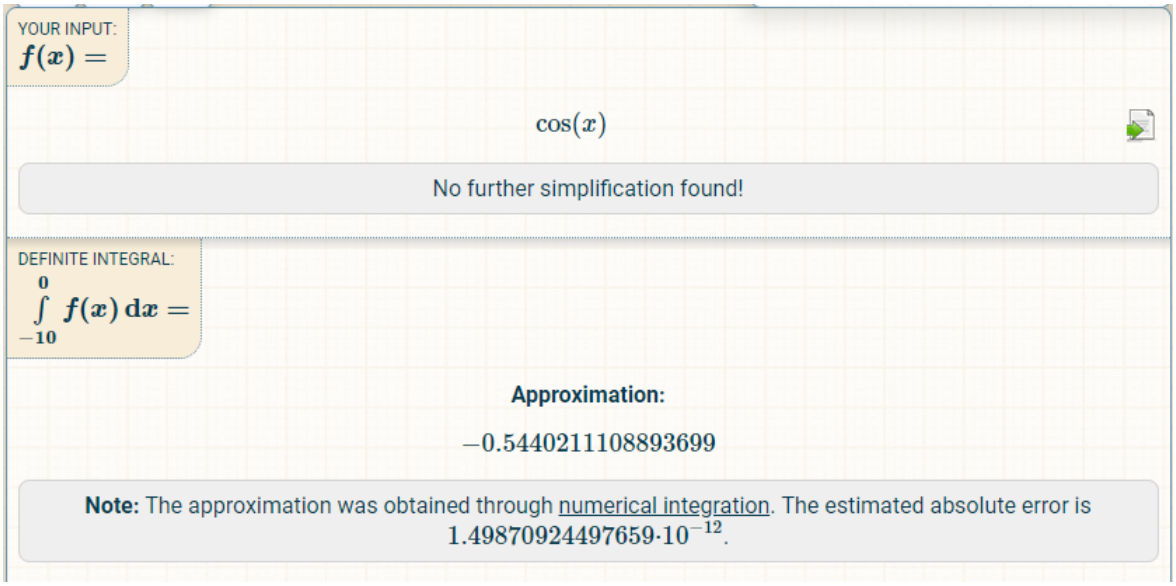


Figure 4.1 – Test Case 1

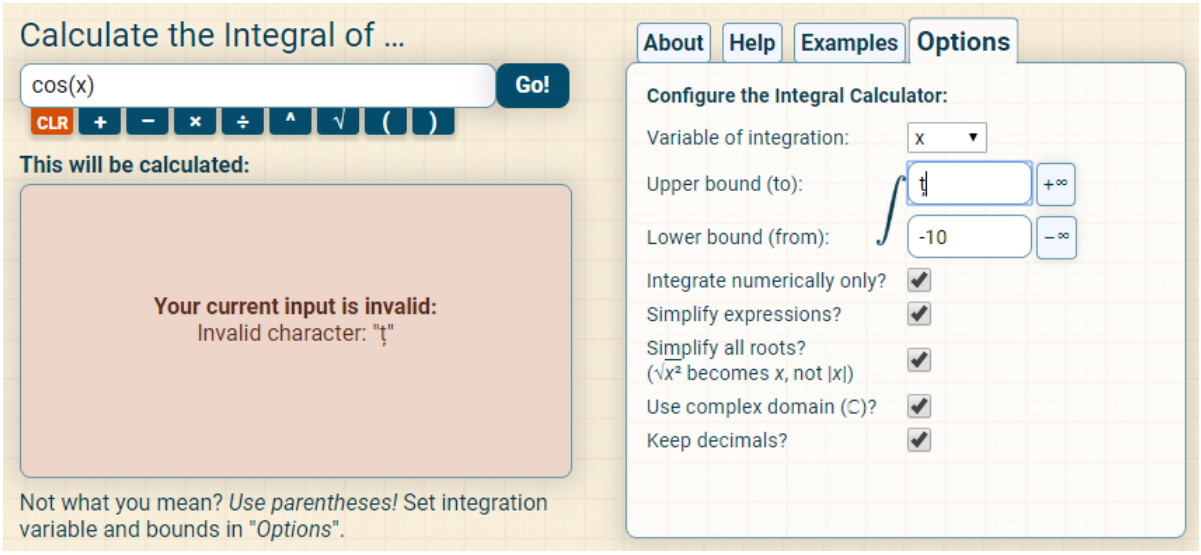


Figure 4.2– Test Case 2

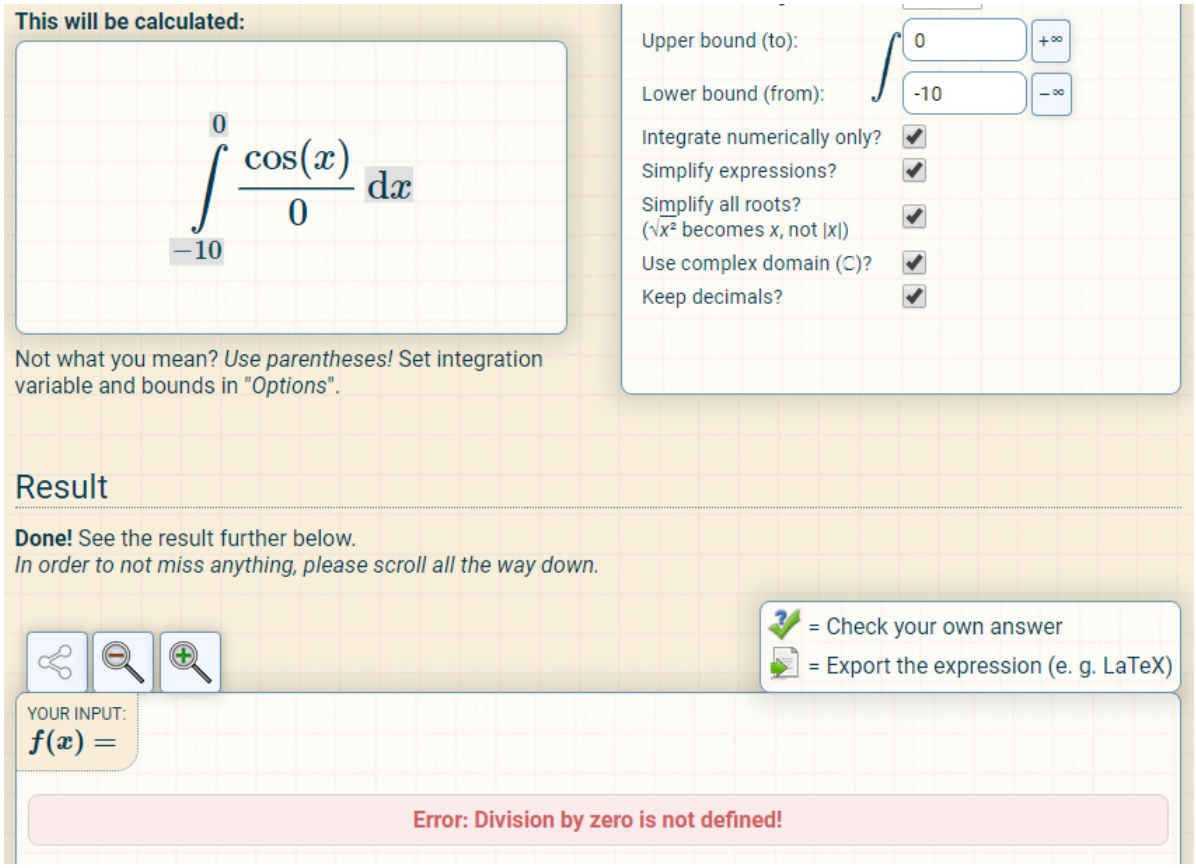


Figure 4.3– Test Case 3

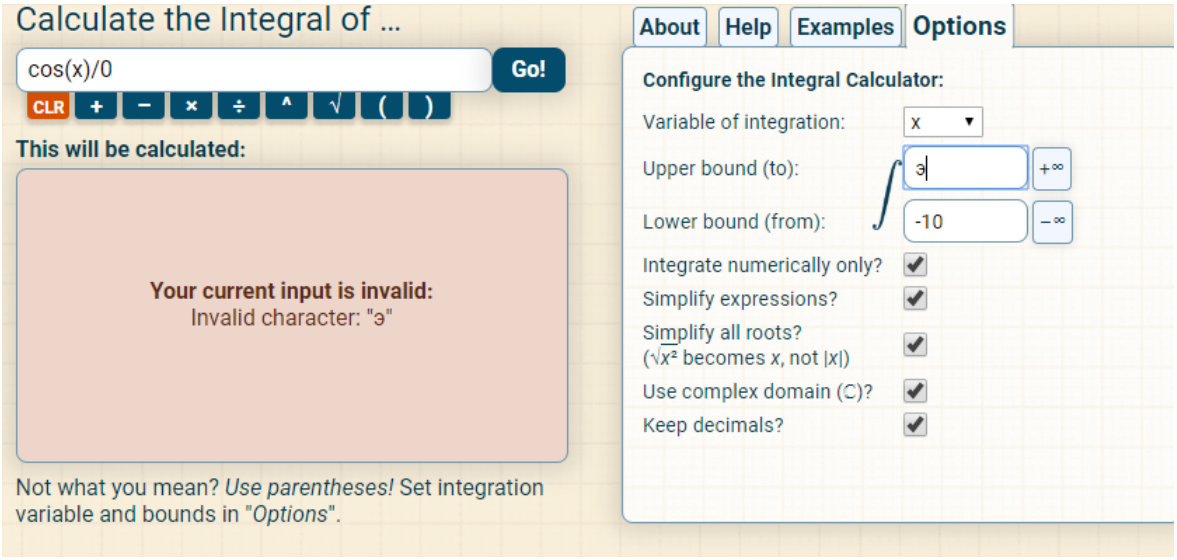


Figure 4.4– Test Case 4

4.4 Task 4 - Emphasis on tests which get errors

The tests that can get errors in the tested system are :

– Non-valid mathematical expressions;

– Semantically non-valid expressions;

– Not supported character sets;

– Options which are beyond the domain of the used functions;

5 Conclusion

– Black box testing is a software testing method in which the internal structure/design of the item being tested is not known to the tester;

– This method is named so because the software program, in the eyes of the tester, is like a black box inside which one cannot see;

– This method has an advantage that tester doesn't need to know programming languages or how the software has been implemented;

– Test cases can be designed as soon as the specifications are complete;

– But the method has its disadvantages, for example without clear specifications, test cases will be difficult to design;

– Also, only a small number of possible inputs can be tested and many program paths will be left untested;