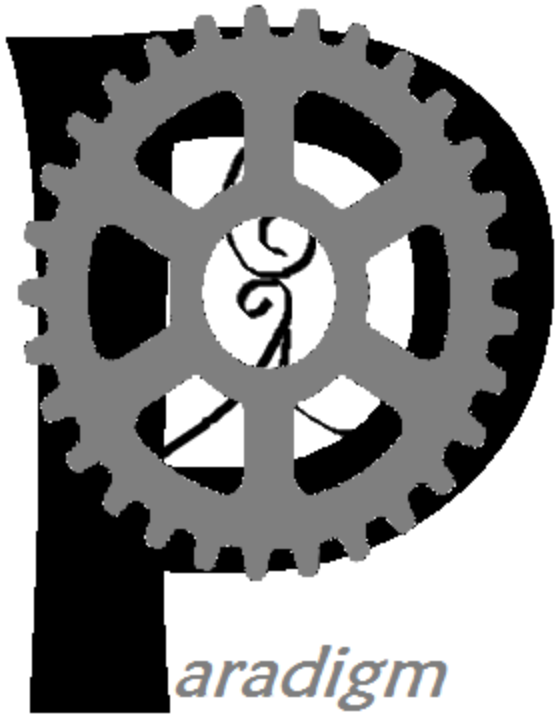
Test Suite User Manual

v.2.0



Paradigm
Damien Moeller
Aimee Phillips
Cavaughn Browne
Christian Norfleet

Table of Contents

Test Suite Overview

The purpose of the Test Suite is to create appropriate test data for early undergraduate computer science student's programs. The test suite is to be run concurrently with the program being tested [the test suite does not run the program].

The Test Suite allows for a variety of test settings that can be saved for later use. After running the test suite can then save a report which can be reviewed by the student's professor to help validate the program. During testing, if the student hand-writes expected results, they may compare them to the results, which is given by the program, and analyzed within the Test Suite.

The user manual is designed to explain the different functions and features of the Test Suite.

System Requirements

Windows OS, .NET Framework 4.5 and above, 128 MB RAM

Installation Instructions

To install the Test Suite, insert your installation medium and run the TestSuitSetup.msi.

Opening Source Code

To open the .cpp and/or .h files you wish to test, either click File → Open File as shown in figure 1.

Figure 1

Tabs

Multiple files can be open at a time in separate tabs. Once a file is opened, concurrent files will be opened in a new tab. Switch between tabs by clicking on them as shown in figure 2.

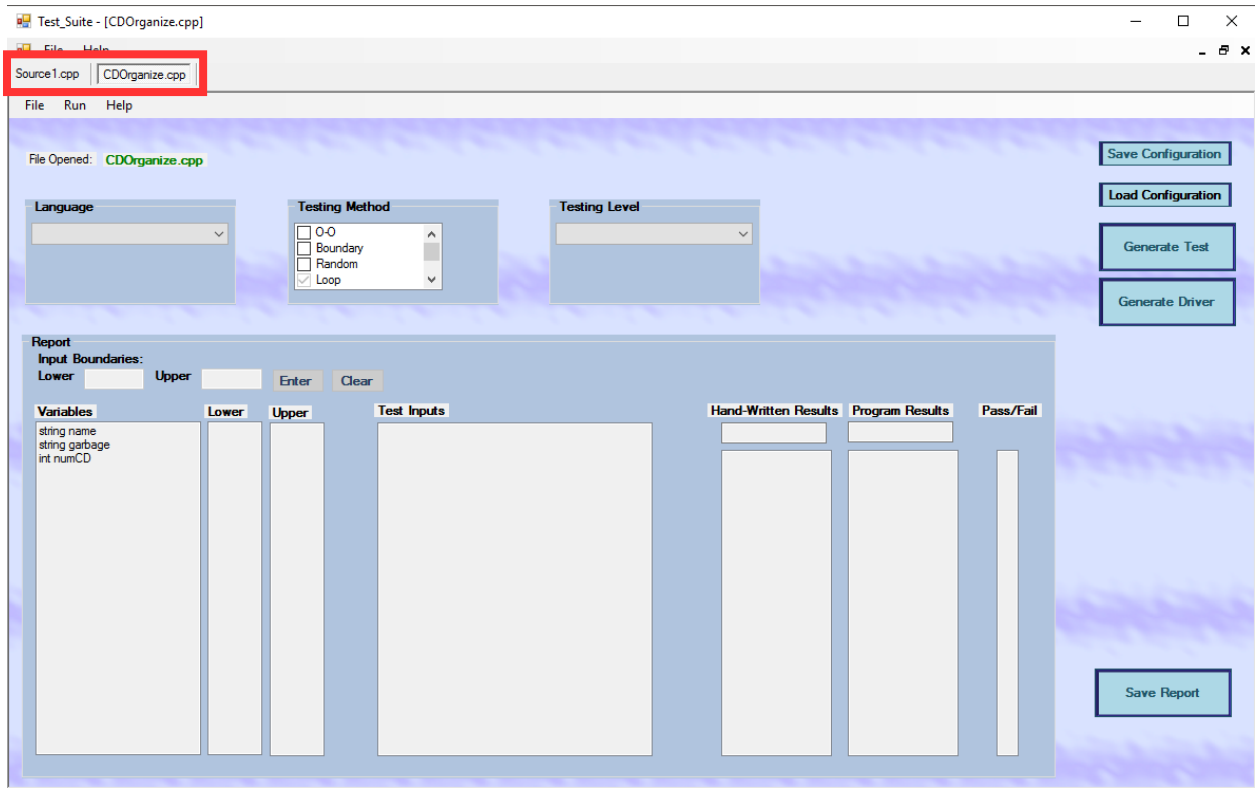


Figure 2

Setting Test Configuration

The configurations of a test must be specified by the user before a test can be generated from source code. Some of these configurations include the language being read, the testing method, and expected exception handling (or test level).

Language

You can set the language through the language drop-down list as shown in Figure 3. (This release only supports C++).

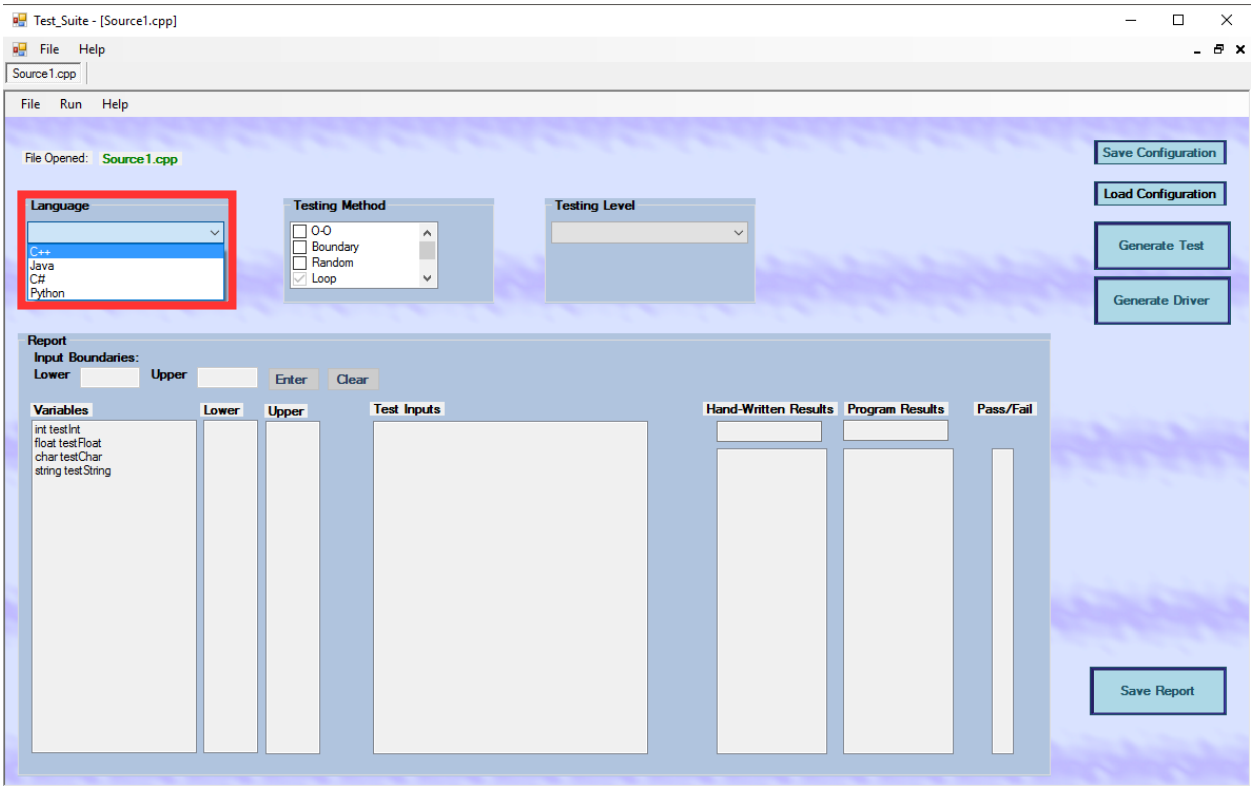


Figure 3

Testing Method

The Test Suite supports a number of different testing methods which can be selected from the Testing Method Checklist shown in figure 4.

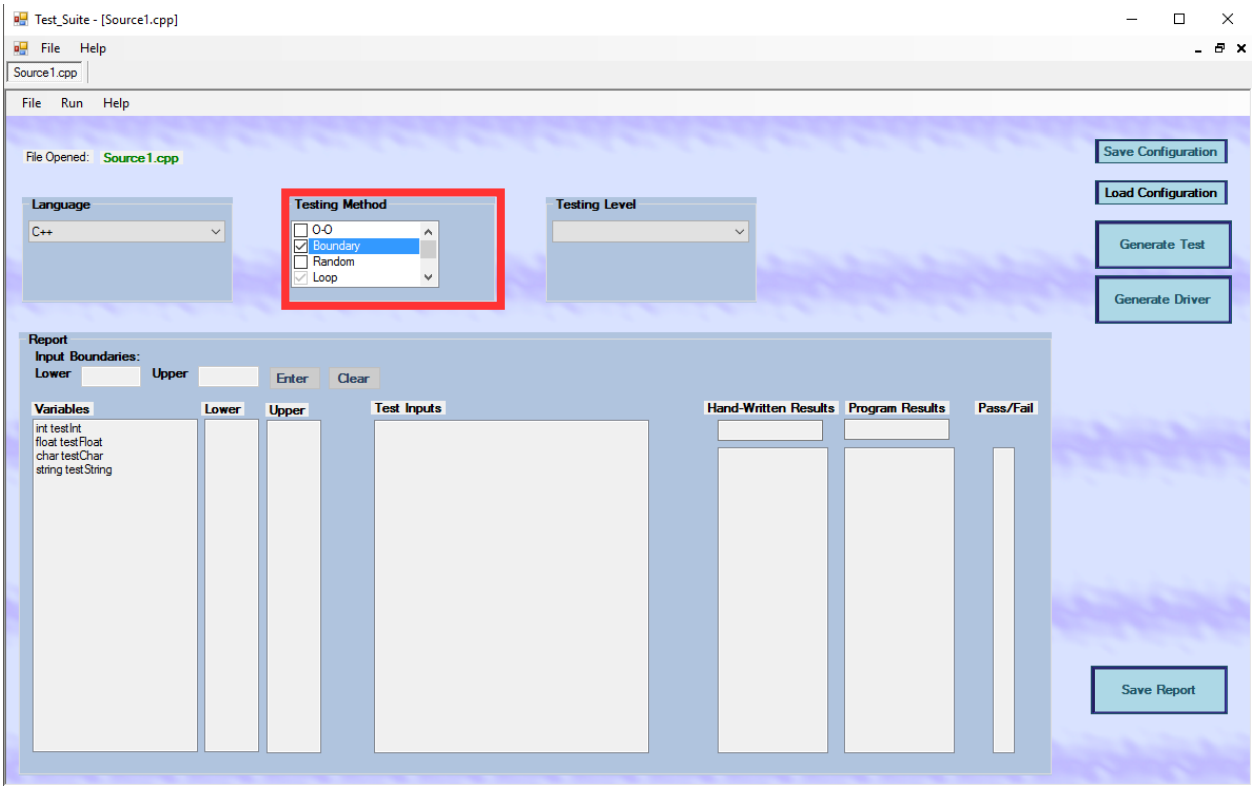


Figure 4

Testing Level

The testing level represents the degree of exception handling you wish to use during testing. The higher the testing level, the more exception handling the Test Suite will consider. The testing level can be changed in the testing level drop list as shown in figure 5.

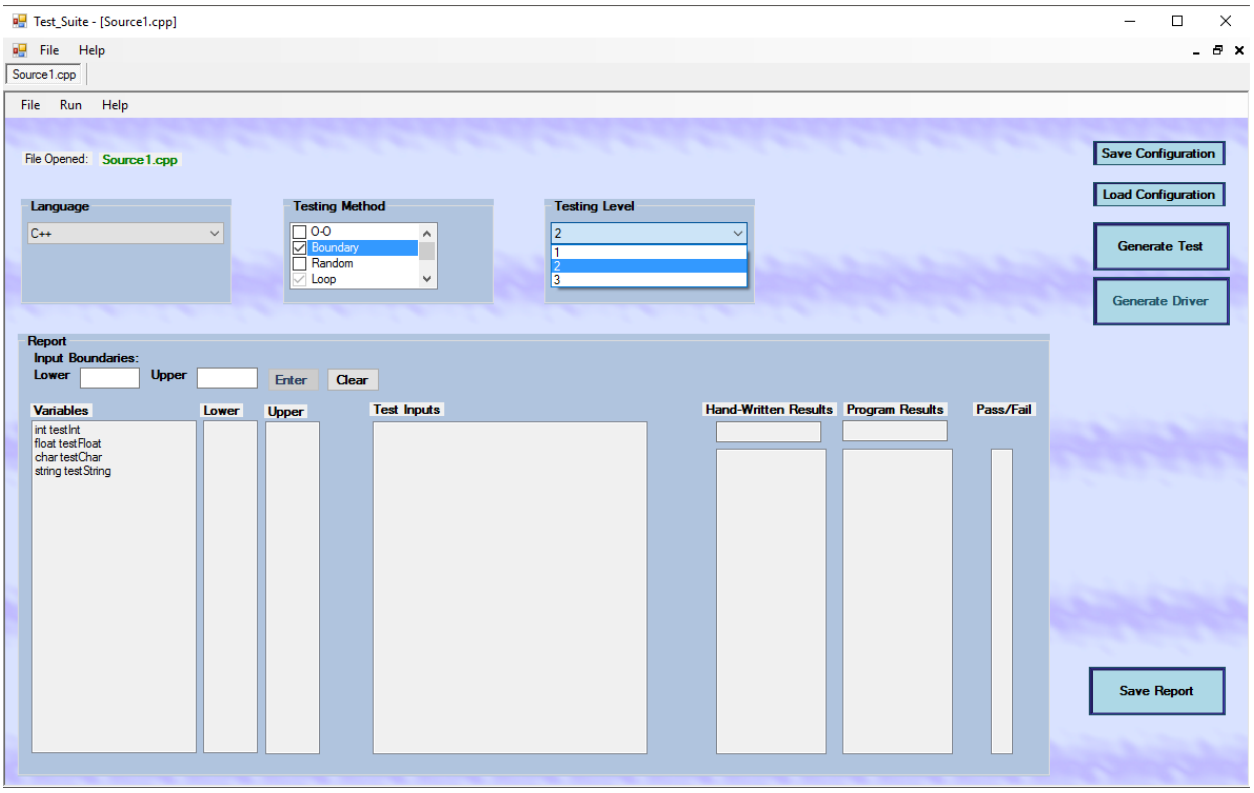


Figure 5

Saving a Configuration

After all of the test settings have been set, you may save the configuration to be used for later test. This is achieved by either clicking File → Save Config or clicking the Save Configuration button as shown in figure 6.

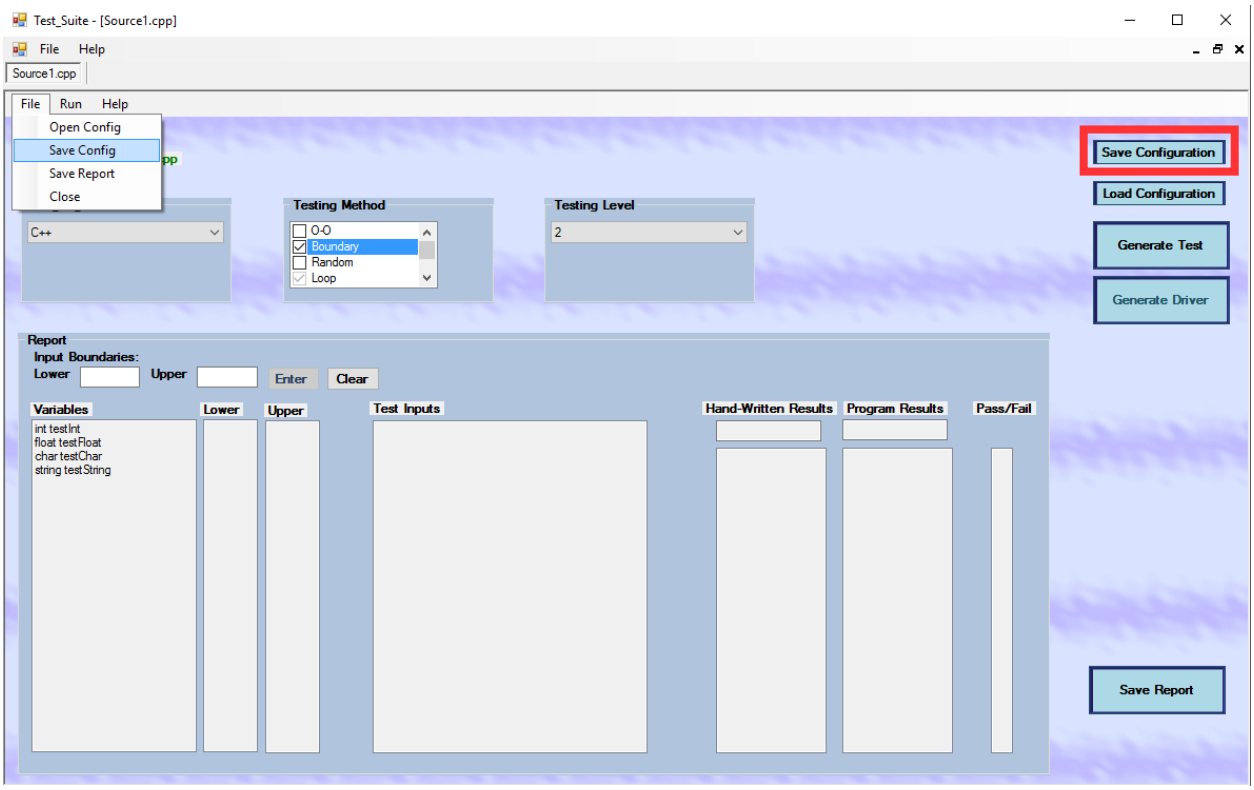


Figure 6

Loading a Configuration

You may load a configuration file if you have one saved. Once loaded, the settings within that file can be used to generate your test. You can load a configuration either by clicking File → Open Config or by clicking the Load Configuration button as shown in figure 7.

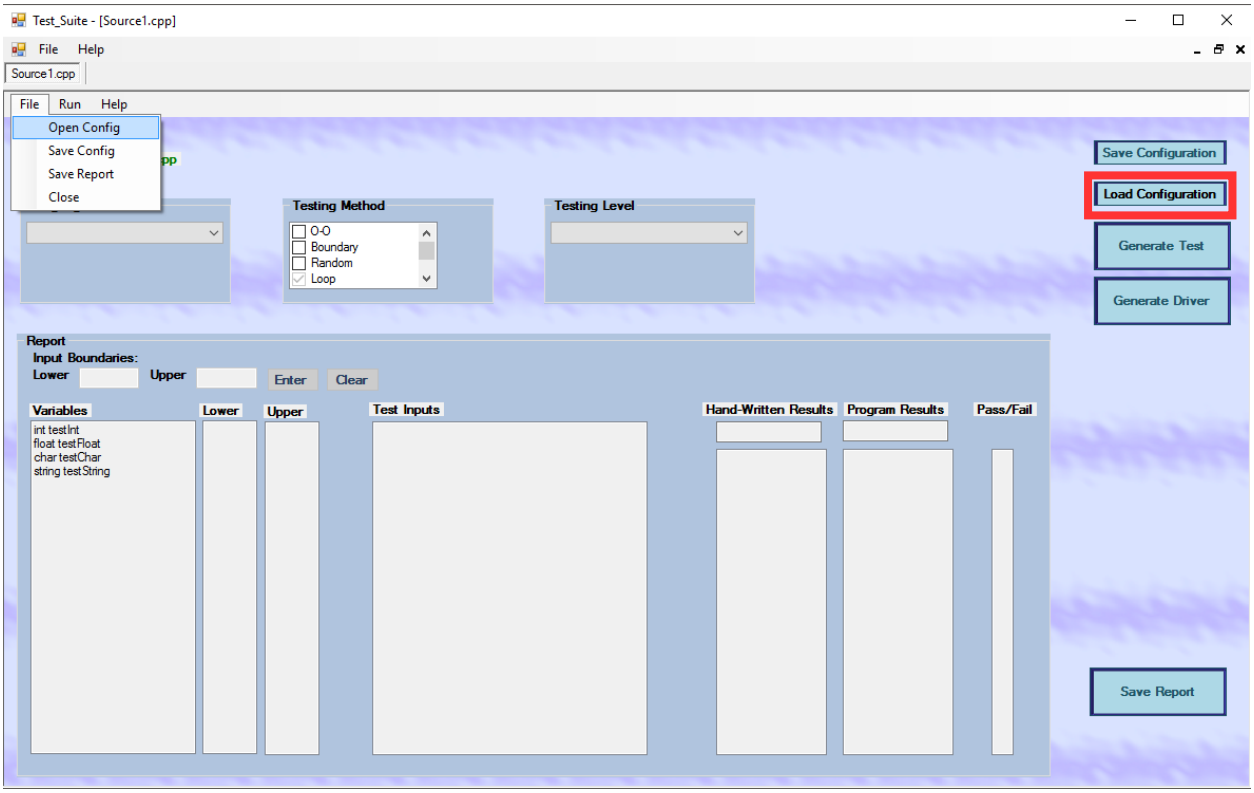


Figure 7

Generating Test Variables

Once you have the configuration set you may generate test data to test your source code. This can be done by clicking Run → Generate Test or clicking the Generate Test button as shown in figure 8. The values to be used for input are displayed in the and Input Values boxes next to the corresponding variable. In the case of boundary testing you must first set upper and lower limits by inputting the values then clicking enter. The values may be cleared by clicking the clear button.

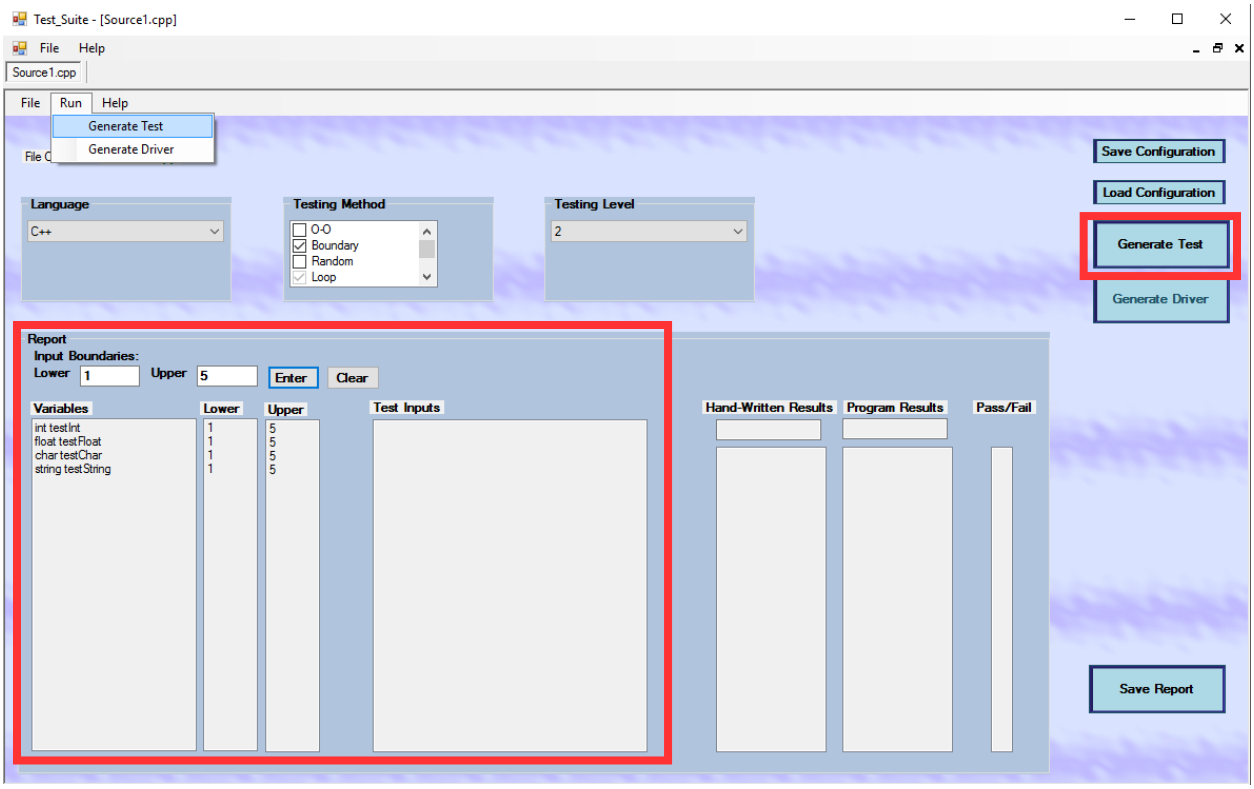


Figure 8

Generating Test Drivers

Once you have the configuration set with the O-O (object oriented) testing method, you may generate test drivers. This can be done by clicking Run → Generate Driver or clicking the Generate Driver button. The test driver code will appear in the Test Inputs box shown in figure 9.

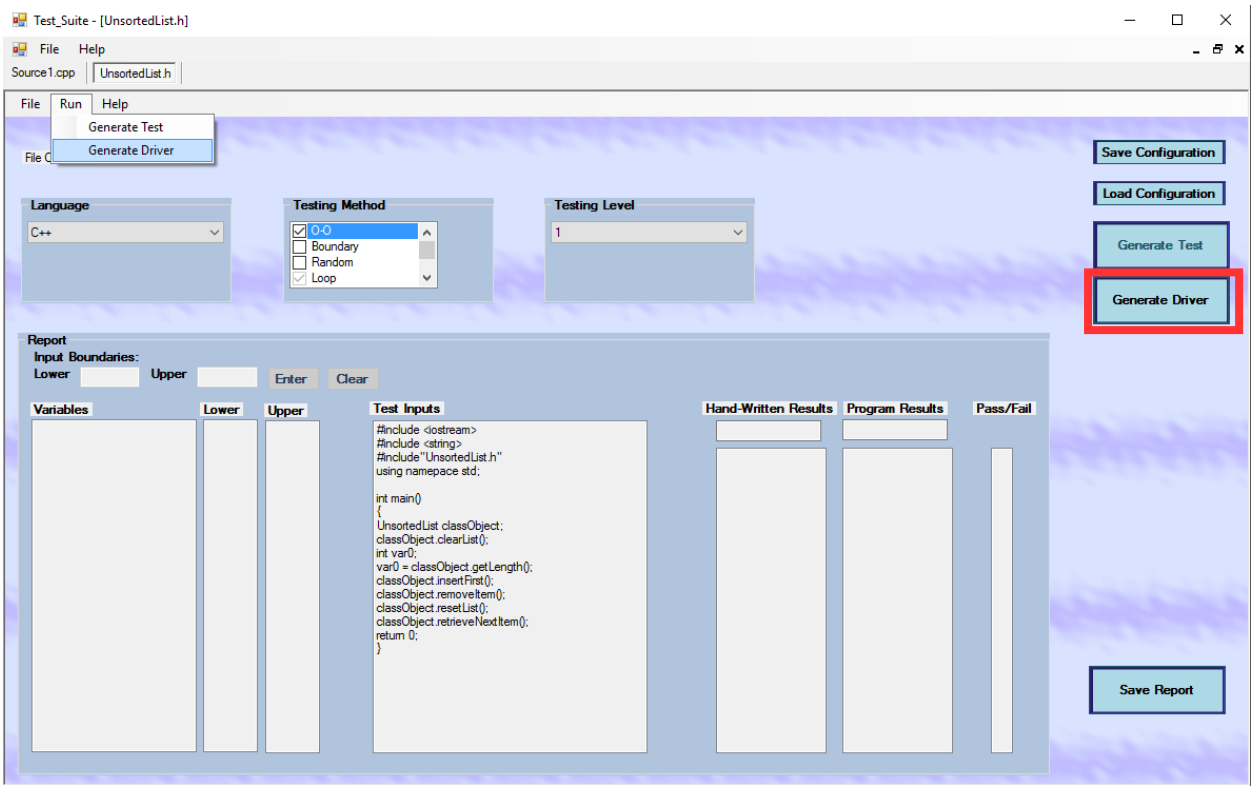


Figure 9

Comparing Results

At any point you may compare the results you expected with the results your program has produced. Type the results into the Handwritten Results Input or the Program Results Input box. The results of the comparison between the two will appear in the Comparison Results box. A √ will appear if the two results are equal and an X if they are unequal as demonstrated in figure 10.

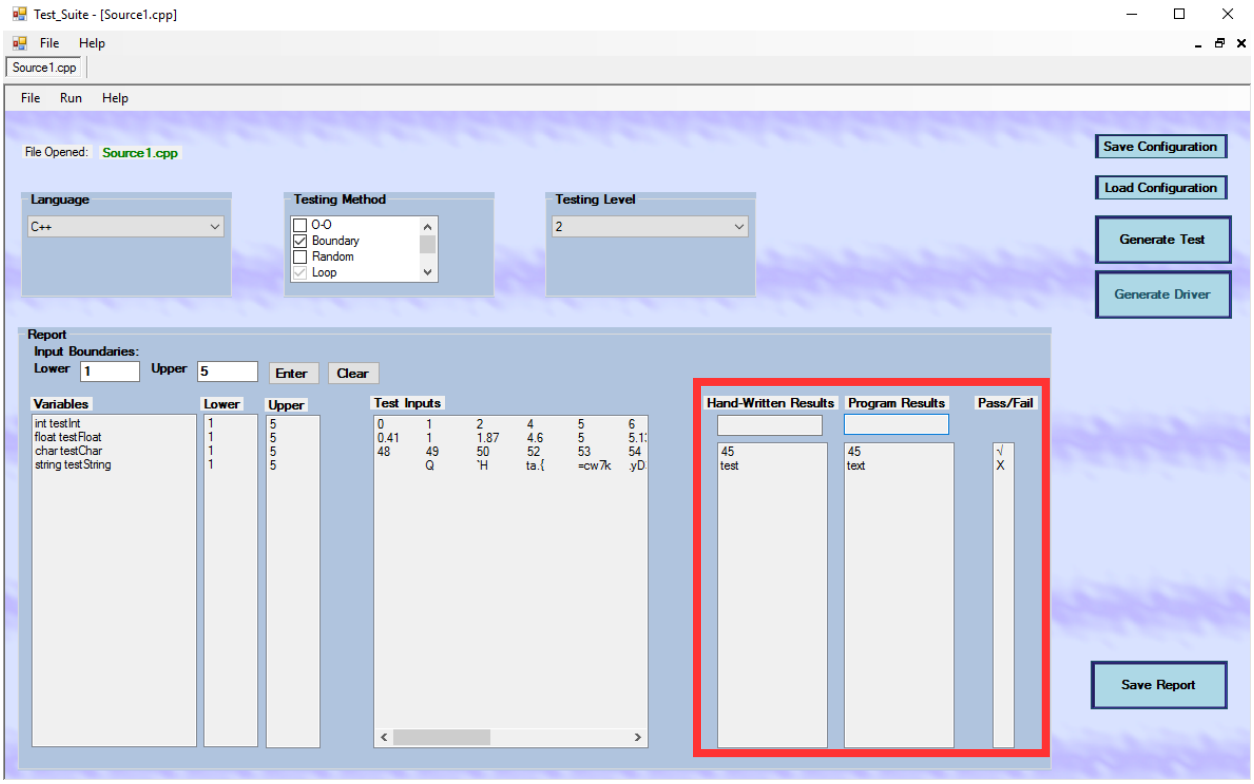


Figure 10

Saving a Report

After you have generated a test, you may save a report which includes the test inputs and any results you have typed into the program. The test and the results will be inserted into a template for readability and consistency. When a test driver is saved, it will create a .cpp file to test your class. To save the report either click File → Save Report or click the Save Report button as shown in figure 11.

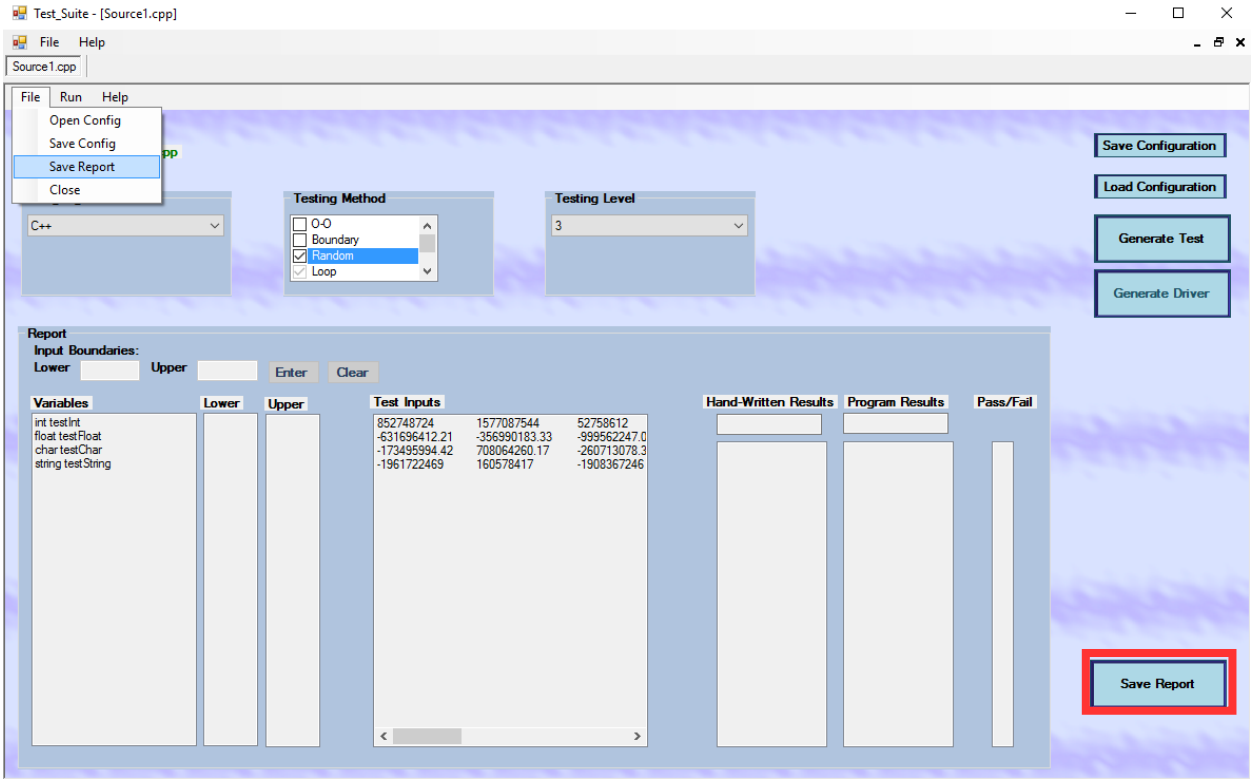


Figure 11

Closing tabs

Tabs may be closed by either clicking File → Close or the x located on the top right corner of the tab as shown in figure 12.

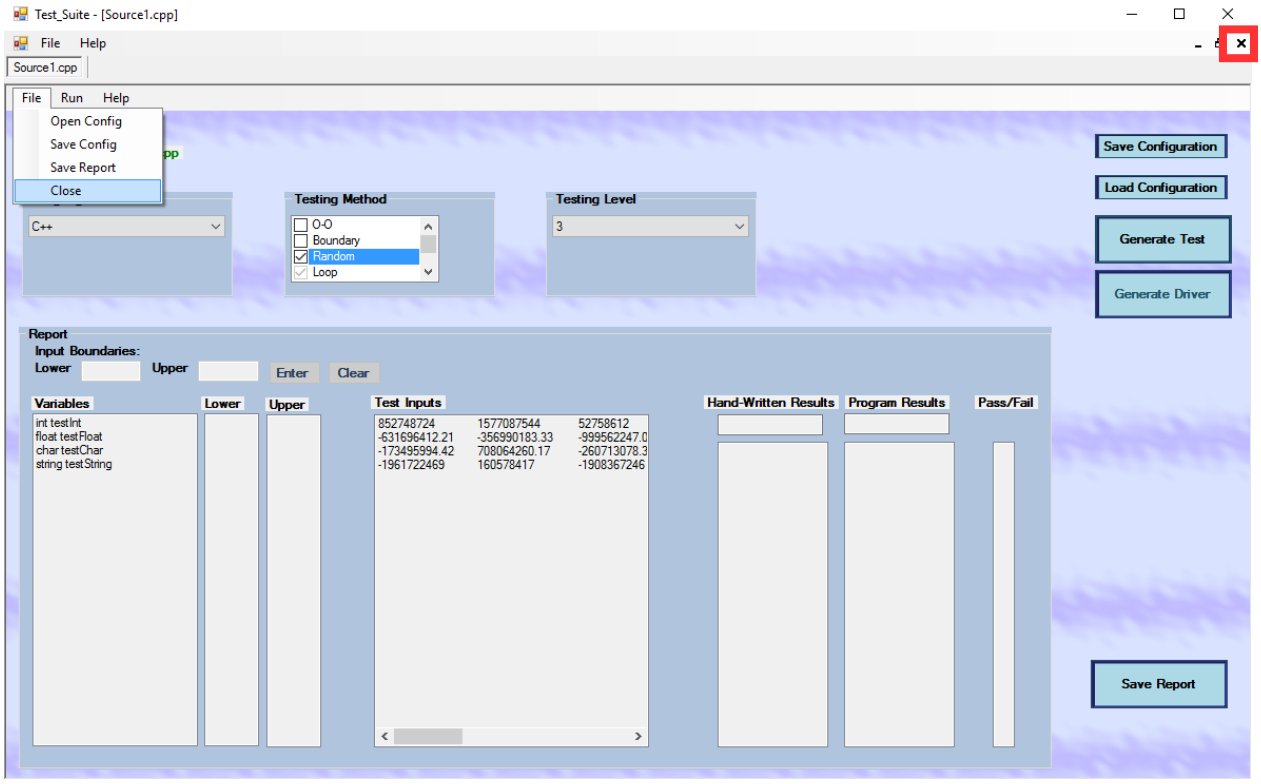


Figure 12

Revision History

Date	Revision History	Comment
4/4/2017	v.1.0	User Manual Draft
4/30/2017	v.2.0	Final User Manual

References

Kung, David Chenho. *Object-oriented Software Engineering: An Agile Unified Methodology*. New York, NY: McGraw-Hill, 2014. Print.

Dr. Catherine Stringfellow (User Manual Sample Documents)