N8 SWALLEY 3/2/2022

**SE463 – Spring 2022**

**Assignment #2**

Equivalence Class Testing and Decision Table Testing

**Introduction**

This document outlines the analysis for Equivalence Class Testing and Decision Table Testing of a sorting system in C++. The sorting system is menu driven and allows for the analysis of two sorting algorithms: bubble sort and insertion sort.

**Array Class**

*/// Pointer to the actual data.*

*char* \* data\_;

*/// Current size of the array.*

*size\_t* cur\_size\_;

*/// Maximum size of the array.*

*size\_t* max\_size\_;

For testing purposes, the current size of the array has been set to 10 and the maximum size has been set to 100. The array has been filled with the character type ‘W’. This allows us to use the four different techniques of boundary value analysis at the given boundaries.

**Normal Boundary Value Analysis:**

Boundary Value testing uses a “single fault” assumption which implies that faults are due to an incorrect value of a single variable. In this specific technique, each test case represents a variable being tested at its minimum, one value above the minimum, maximum, and one value below the maximum boundaries. In addition, there is one test case using nominal values for each variable. In total, there were 4n+1, or 13 test cases.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cases: | Cur\_size | Max\_size | Data\_ | Outcome |
| 1 | 0 | 50 | [cur\_size] | Valid |
| 2 | 1 | 50 | [cur\_size] | Valid |
| 3 | 5 | 50 | [cur\_size] | Valid |
| 4 | 9 | 50 | [cur\_size] | Valid |
| 5 | 10 | 50 | [cur\_size] | Valid |
| 6 | 5 | 0 | [cur\_size] | Valid |
| 7 | 5 | 1 | [cur\_size] | Valid |
| 8 | 5 | 99 | [cur\_size] | Valid |
| 9 | 5 | 100 | [cur\_size] | Valid |
| 10 | 5 | 50 | [0] | Valid |
| 11 | 5 | 50 | [1] | Valid |
| 12 | 5 | 50 | [cur\_size-1] | Valid |
| 13 | 5 | 50 | [cur\_size] | Valid |

The following snippets demonstrate functionality of each test case highlighted in blue.

A screenshot of a computer

Description automatically generated with medium confidence

Text

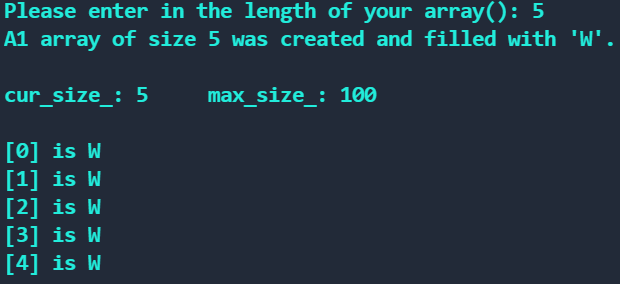
Description automatically generated

Text

Description automatically generated

Text

Description automatically generated



Text

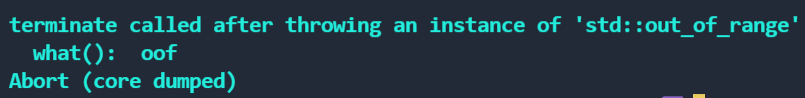
Description automatically generated

All expected outcomes were reported successfully by Normal Boundary Value Testing.

**Robust Boundary Value Analysis:**

In Robust Value Testing, input values are given just outside the set boundaries to test for invalid cases. The total number of test cases is the same as before, only with six extra cases for each variables invalid boundaries.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cases: | Cur\_size | Max\_size | Data\_ | Outcome |
| 1 | 0 | 50 | [cur\_size] | Valid |
| 2 | 1 | 50 | [cur\_size] | Valid |
| 3 | 5 | 50 | [cur\_size] | Valid |
| 4 | 9 | 50 | [cur\_size] | Valid |
| 5 | 10 | 50 | [cur\_size] | Valid |
| 6 | 5 | 0 | [cur\_size] | Valid |
| 7 | 5 | 1 | [cur\_size] | Valid |
| 8 | 5 | 99 | [cur\_size] | Valid |
| 9 | 5 | 100 | [cur\_size] | Valid |
| 10 | 5 | 50 | [0] | Valid |
| 11 | 5 | 50 | [1] | Valid |
| 12 | 5 | 50 | [cur\_size-1] | Valid |
| 13 | 5 | 50 | [cur\_size] | Valid |
| 14 | 11 | 50 | [cur\_size] | Invalid |
| 15 | -1 | 50 | [cur\_size] | Invalid |
| 16 | 5 | -1 | [cur\_size] | Invalid |
| 17 | 5 | 101 | [cur\_size] | Invalid |
| 18 | 5 | 50 | [cur\_size+1] | Invalid |
| 19 | 5 | 50 | [-1] | Invalid |

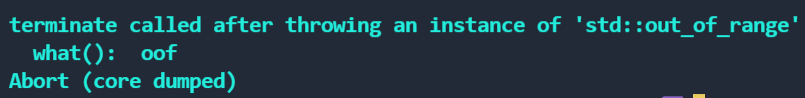


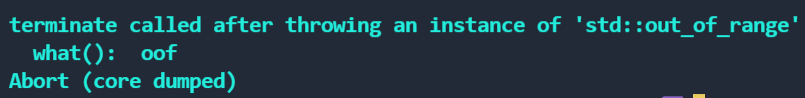
A screenshot of a computer

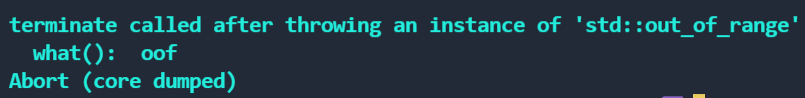
Description automatically generated with medium confidence

Text

Description automatically generated







All expected outcomes were reported successfully by Robust Boundary Value Testing.

**Worst-case Boundary Value Analysis:**

In Worst-case Boundary Value Analysis, the total number of test cases is equal to 5^n, or 125. This is a “black box” testing technique that makes all combinations of each value of one variable with each value of another variable. To demonstrate the boundary tests, I focused on 4 specific cases: When Cur\_size and Max\_size were both near their minimum and maximum boundaries, when Cur\_size was near the minimum boundary while Max\_size was near the maximum boundary, and when Cur\_size was near the maximum boundary while Max\_size was near the minimum boundary.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cases: | Cur\_size | Max\_size | Data\_ | Outcome |
| 1 | 0 | 0 | [cur\_size] | Valid |
| 2 | 10 | 100 | [cur\_size] | Valid |
| 3 | 0 | 100 | [cur\_size] | Valid |
| 4 | 10 | 0 | [cur\_size] | Valid |

Text

Description automatically generated

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

All expected outcomes were reported successfully by Worst-case Boundary Value Testing.

**Robust Worst-case Boundary Value Analysis:**

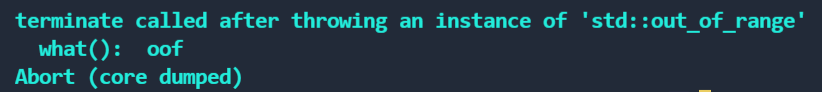
In Robust Worst-case Boundary Value Analysis, the total number of test cases is equal to 7^n, or 343. This technique is essentially combining all other techniques. To demonstrate the boundary tests, I modeled my test cases after the previous example except

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cases: | Cur\_size | Max\_size | Data\_ | Outcome |
| 1 | -1 | -1 | [cur\_size+1] | Invalid |
| 2 | 11 | 101 | [cur\_size+1] | Invalid |
| 3 | -1 | 101 | [cur\_size+1] | Invalid |
| 4 | 11 | -1 | [cur\_size+1] | Invalid |

stretching the inputs beyond the acceptable boundaries.

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidenceText

Description automatically generated

All expected outcomes were reported successfully by Robust Worst-case Boundary Value Testing.