

Sabuj Kumar Modak

IT Specialist | SQA Trainee | Cybersecurity Specialist

Project Tittle: Boundary Value Analysis

Perform Date: 07 June 2023

Contents

1	Abstract	2
2	Objectives	2
3	Introduction	2
4	Working Procedure	2
4.1	Problem selection for manual testing2
4.2	Output3
5	Black-box Testing for Triangle Problem	5
5.1	Input Variables5
5.2	Designing Test Case Report5
6	Conclusion	5

1 Abstract

Boundary Value Analysis (BVA) is a methodical software testing technique that focuses on particular input values that are close to a system's input domain boundaries in order to increase the efficacy and efficiency of the testing process. This method acknowledges that errors are more likely to happen near the input range's extremities, where system behavior is frequently unpredictable.

2 Objectives

In this project, I will employ Boundary Value Analysis to test the Triangle Problem function. Our objective is to evaluate the precision and effectiveness of the code implementation by constructing test cases that encompass the utmost values and boundary circumstances of the triangle side lengths. The test cases will encompass ordinary values, values slightly exceeding and falling below the minimum and maximum limits, as well as scenarios involving extremely long or short side lengths. Our aim is to comprehensively assess the functionality of the code under various conditions and verify its accuracy and efficiency.

3 Introduction

Boundary value analysis (BVA), a technique for black box testing, leverages boundary values to generate a comprehensive set of test cases. This method operates on the assumption that a single fault often underlies failures. In the specific context of the Triangle Problem, BVA focuses on identifying errors occurring at the boundaries of the input domain. Test cases are meticulously designed to include extreme values for one variable while incorporating nominal values for the remaining variables. Potential problems can be found and fixed by analyzing the function's behavior at boundary circumstances, which increases the Triangle Problem function's dependability and accuracy.

4 Working Procedure

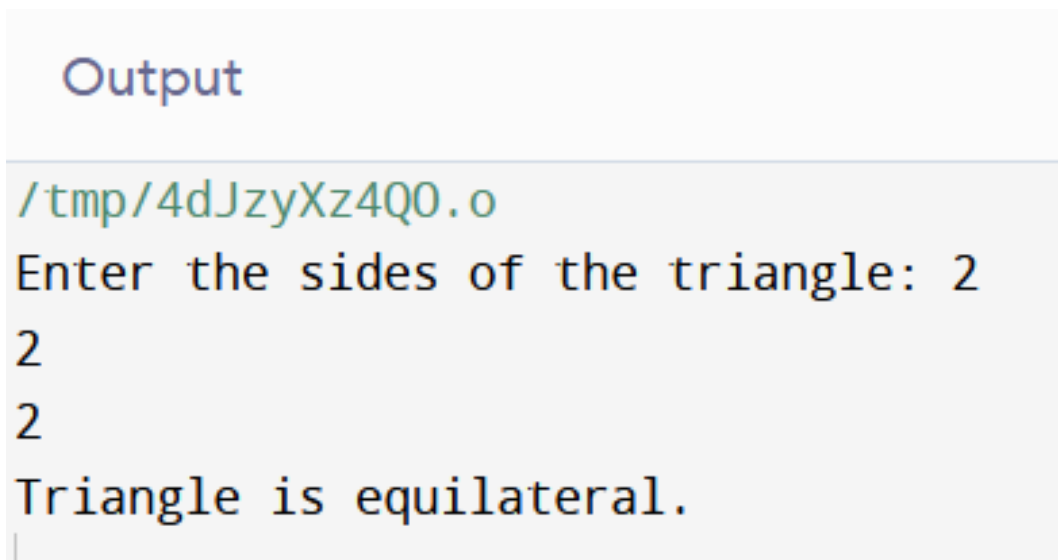
4.1 Problem selection for manual testing

Code in C language for the triangle problem.

```
1  #include<stdio.h>
2  #include<conio.h>
3
4  intmain()
5  {
6      inta, b, c, c1 , c2 , c3;
7      do
8      {
9          printf("Enterthesidesoftriangle\n");
```

```
10     scanf("%d%d%d", &a, &b, &c);
11     c1 = (( a >= 1) && ( a <= 10) );
12     c2 = (( b >= 1) && ( b <= 10) );
13     c3 = (( c >= 1) && ( c <= 10) );
14     if(!c1)
15         printf("Valueofaisoutofrange\n");
16     if(!c2)
17         printf("Valueofbisoutofrange\n");
18     if(!c3)
19         printf("Valueofcisoutofrange\n");
20 }while(!c1 || !c2 || !c3);
21
22 if(( a + b) > c && ( b + c) > a && ( c + a) > b)
23 {
24     if( a == b && b == c)
25         printf("Triangleisequilateral\n");
26     elseif(a != b && b != c && c != a)
27         printf("Triangleisscalene\n");
28     else
29         printf("Triangleisisosceles\n");
30 }
31 else
32     printf("Trianglecannotbeformed\n");
33
34 getch();
35 return0;
36 }
```

4.2 Output



```
Output
/tmp/4dJzyXz4Q0.o
Enter the sides of the triangle: 2
2
Triangle is equilateral.
```

Figure 1: equilateral triangle

Output

```
/tmp/4dJzyXz4Q0.o
```

```
Enter the sides of the triangle: 1 1 2
```

```
Triangle cannot be formed.
```

Figure 2: can't be formed triangle

Output

```
/tmp/4dJzyXz4Q0.o
```

```
Enter the sides of the triangle: 5 5 7
```

```
Triangle is isosceles.
```

Figure 3: Isosceles triangle

Output

```
/tmp/4aSPd9WPVc.o
```

```
Enter the sides of the triangle: 2 3 4
```

```
Triangle is scalene.
```

Figure 4: scalene

5 Black-box Testing for Triangle Problem

5.1 Input Variables

The triangle sides' lengths is taken into account while doing black-box testing for the Triangle Problem. The triangle's properties and kind are determined by these factors.

5.2 Designing Test Case Report

Test Case Number	Input	1st Side	2nd Side	3rd Side	Expected Output	Test Result
1	Every sides are equal	2	2	2	Equilateral triangle	Pass
2	Two sides are same ,last one different	5	5	7	Isosceles triangle	Pass
3	All sides have distinct lengths	3	4	5	Scalene triangle	Pass
4	The sum of Side 1 Side 2 is not greater than the length of Side 3	1	2	10	Triangle cannot be formed	Pass
5	Zero length side	0	4	5	Triangle cannot be formed	Pass
6	Negative side length	3	-4	5	Triangle cannot be formed	Pass
7	Maximum side length	2147483647	2147483647	2147483647	Equilateral triangle	Pass
8	Random side lengths	5	10	7	Isosceles triangle	Pass

Figure 5: Test case Excel Report

6 Conclusion

Black-box testing can benefit from BVA, especially when working with ordered or sequential data. It makes that the program handles boundary values appropriately and can spot any potential problems with these boundary conditions. It also focuses on testing the boundaries between valid and invalid input values. Based on the code of triangle, we can draw the following conclusions regarding the BVA technique:

- The code expects the sides of a triangle as input values (variables a, b, and c).
- The valid range for each side is defined as values between 1 and 10 (inclusive) ($1 \leq a \leq 10$, $1 \leq b \leq 10$, $1 \leq c \leq 10$).
- The code includes input validation checks (c1, c2, and c3) to ensure that the entered values fall within the valid range.
- If any of the input values are outside the valid range, the code outputs a corresponding error message.
- The code uses the entered side values to determine the type of triangle: equilateral, scalene, or isosceles.
- If the sum of any two sides is greater than the third side for all three combinations, it indicates a valid triangle.

- If a valid triangle is formed, the code outputs the type of triangle based on the side values.
- If a valid triangle cannot be formed (invalid side values), the code outputs a message stating that a triangle cannot be formed.