**EXERCISE 1**

**NAME: Nguyễn Công Tuyển STUDENT CODE: 3122411239**

1. **VERIFICATION AND VALIDATION**

**Description:** The purpose is to help users to solve a 2-degree equation (a*x*2+b*x*+c).

**Spec:** Given input of *a*, *b*, and *c*; the system returns the outputs of *x*1 and *x*2 (extreme cases are temporarily not considered)

Two systems are developed as follows.

|  |  |  |  |
| --- | --- | --- | --- |
| SYSTEM 1 | SYSTEM 2 | | |
| a b c    Press here to get *x1*  solutions *x2* |  | Step 1: DELTA calculation  a b c    Press here to get DELTA *delta*  (discriminant) |  |
|  | | |
|  | Step 2: Solutions resolving  DELTA *x1*  Press here to get  solutions *x2* |  |
| Code:  x1 = (-b +sqrt(DELTA))/2a x2 = -b -sqrt(DELTA/2a) | Code:  DELTA = (b\*b-4\*a\*c)  x1 = (-b +sqrt(DELTA))/2a x2 = (-b -sqrt(DELTA))/2a | | |

What are the problems of those two systems? Write down your answer here.

Vấn đề của System 1:

* Không tính DELTA trước khi tính nghiệm, hệ thống trên giả định DELTA có sẵn nhưng không có các bước tính toán liên quan dẫn đến hệ thống không hoàn chỉnh
* Công thức tính x2 sai : ghi "-sqrt(DELTA/2a)" thay vì "(-b - sqrt(DELTA)) / 2a" lúc này dẫn đến kết quả sai cho x2 trong mọi trường hợp DELTA > 0
* Công thức tính sai: Ghi 2a thay vì 2\*a
* Không tính các trường hợp DELTA < 0 hoặc DELTA = 0

Vấn đề của System 2:

* Giao diện tách bước có thể sai nếu người dùng đổi a, b, c sau bước 1 mà không tính lại DELTA
* Công thức tính sai: Ghi 2a thay vì 2\*a
* Không tính các trường hợp DELTA < 0 hoặc DELTA = 0

1. **TEST-CASES**

**Description:** Some input values

* 1. How many test-cases we need for the following function *f*1. What are they?

int f1(int x) { if (x > 10)

return 2 \* x; else

return -x;

}

Dựa trên điều kiện x > 10 suy ra có 2 nhánh:

x > 10: return 2 \* x.

x <= 10: return -x

Số test-case cần cho function *f*1: 2 test (x > 10) và (x ≤ 10).

|  |  |  |
| --- | --- | --- |
| **TC** | **Input x** | **Output** |
| TC1 | 11 | 22 |
| TC2 | 10 | -10 |

* 1. Check if your test-cases can detect error if *f*1 is implemented as follows

int f1(int x) { if (x > 10)

return 2 \* x; else if (x > 0)

return -x; else

return 2 \* x;

}

In this case, how many test-cases we need to test this function? What are they?

Nếu f1 được triển khai như sau test-cases có thể phát hiện lỗi

Test-case từ a):

x=11: Trả 22 (không phát hiện lỗi)

x=10: Trả -10 (không phát hiện lỗi)

x= -1: theo hàm gốc -(-1)=1 nhưng trả 2\*(-1)= -2 => Phát hiện lỗi

Số test-case càn cho funtion trên: 3 test (x > 10), (0 < x ≤ 10), (x ≤ 0)

|  |  |  |
| --- | --- | --- |
| **TC** | **Input x** | **Output** |
| TC1 | 11 | 22 |
| TC2 | 10 | -10 |
| TC3 | -1 | 1 |

* 1. How many test-cases we need to test this function? What are they?

int f2(int x) { if (x < 10)

return 2 \* x; else if (x < 2)

return -x; else

return 2 \* x;

}

In this case, how many test-cases we need to test this function? What are they?

Nhánh else-if (x < 2) không bao giờ được thực thi do đã bị (x < 10) bắt trước đó => dead code

Số test-case càn cho funtion trên: 2 test (x < 10) và (x >= 10)

|  |  |  |
| --- | --- | --- |
| **TC** | **Input x** | **Output** |
| TC1 | 5 | 10 |
| TC2 | 10 | 20 |

* 1. How many test-cases we need to test this function? What are they?

int f3(int x) {

if (log(x \* x \* cos(x)) < 3 \* x) return 2 \* x;

else

return 2 \* x;

}

Số test-case càn cho funtion trên: 1 test

|  |  |  |
| --- | --- | --- |
| **TC** | **Input x** | **Output** |
| TC1 | 1 | 2 |

* 1. Check if your test-cases can detect error if *findMax* is implemented as follows

int findMax(int num1, int num2, int num3) { int max = 0;

if ((num1 > num2) && (num1 > num3)) max = num1;

if ((num2 > num1) && (num2 > num3)) max = num2;

if ((num3 > num1) && (num3 > num2)) max = num3;

return max;

}

In this case, how many test-cases we need to test this function? What are they?

Cần 5 test case để bao phủ và phát hiện lỗi:

|  |  |  |  |
| --- | --- | --- | --- |
| **TC** | **Input (num1,num2,num3)** | **Kỳ vọng** | **Output** |
| TC1 | (3,2,1) | 3 | 3 |
| TC2 | (2,5,1) | 5 | 5 |
| TC3 | (2,1,7) | 7 | 7 |
| TC4 | (-5,-2,-3) | -2 | Lỗi |
| TC5 | (5,5,1) | 5 | Lỗi |

- Lỗi: khởi tạo max = 0 => với toàn số âm, trả về 0 sai

- Lỗi: dùng '>' (không cho phép = ) => trường hợp bằng nhau, không nhánh nào chạy, trả 0

1. **PRATICE 1**

* Mô tả bài toán, các input / output có thể có của bài toán

Mô tả: Giải phương trình bậc 4 dạng trùng phương ax⁴ + bx² + c = 0, đặt y = x² để chuyển thành phương trình bậc hai theo y

Input: a, b, c (số thực)

Output:

* -1 nếu vô số nghiệm.
* 0 nếu vô nghiệm.
* n (1, 2, hoặc 4) là số nghiệm thực, kèm giá trị nghiệm x.
* Xây dựng các test cases kiểm tra tính đúng đắn chương trình

Cần xây dựng 9 test-case

|  |  |  |  |
| --- | --- | --- | --- |
| **TC** | **Input(a,b,c)** | **Trường hợp** | **Output** |
| TC1 | 0,0,0 | a=b=c=0 | **Vô số nghiệm** (trả -1 theo spec) |
| TC2 | 0,0,5 | a=b=0, c≠0 | **Vô nghiệm** (0) |
| TC3 | 0,1,0 | a=0, b≠0, y=0 | x = 0 (1 nghiệm thực) |
| TC4 | 0,1,-4 | a=0, b≠0, y>0 | y = 4 => x = ±2 (2 nghiệm) |
| TC5 | 0,1,4 | a=0, b≠0, y<0 | vô nghiệm (0) |
| TC6 | 1,0,1 | a≠0, Δ < 0 | vô nghiệm (0) |
| TC7 | 1,-2,1 | a≠0, Δ = 0, y>0 | y=1 => x = ±1 (2 nghiệm) |
| TC8 | 1,2,1 | a≠0, Δ = 0, y<0 | y = -1 => vô nghiệm (0) |
| TC9 | 1,-5,4 | a≠0, Δ > 0, y1,y2>0 | x = ±1, ±2 (4 nghiệm) |

* Viết đoạn mã tự động kiểm tra chương trình cho bên dưới đúng hay sai?

#include <iostream>

#include <cmath>

#include <vector>

#include <array>

using namespace std;

int solveQuartic(double a, double b, double c, double x[]) {

if (a == 0 && b == 0 && c == 0) {

return -1;

}

if (a == 0 && b == 0) {

return 0;

}

if (a == 0) {

double y = -c / b;

if (y < 0) return 0;

x[0] = sqrt(y);

x[1] = -sqrt(y);

return 2;

}

double delta = b \* b - 4 \* a \* c;

if (delta < 0) return 0;

double y1 = (-b + sqrt(delta)) / (2 \* a);

double y2 = (-b - sqrt(delta)) / (2 \* a);

int count = 0;

if (y1 >= 0) {

x[count++] = sqrt(y1);

x[count++] = -sqrt(y1);

}

if (y2 >= 0 && y2 != y1) {

x[count++] = sqrt(y2);

x[count++] = -sqrt(y2);

}

return count;

}

int main() {

double x[4];

// Danh sách test-case: {a, b, c}

vector<array<double, 3>> tests = {

{0,0,0}, // TC1: vô số nghiệm

{0,0,5}, // TC2: vô nghiệm

{0,1,0}, // TC3: nghiệm thực 0

{0,1,-4}, // TC4: ±2

{0,1,4}, // TC5: vô nghiệm

{1,0,1}, // TC6: Δ < 0

{1,-2,1}, // TC7: Δ = 0, y=1

{1,2,1}, // TC8: Δ = 0, y < 0

{1,-5,4} // TC9: 4 nghiệm ±1, ±2

};

for (size\_t t = 0; t < tests.size(); t++) {

double a = tests[t][0], b = tests[t][1], c = tests[t][2];

cout << "Test case " << (t + 1)

<< " (a=" << a << ", b=" << b << ", c=" << c << "): ";

int n = solveQuartic(a, b, c, x);

if (n == -1) {

cout << "Infinite solutions." << endl;

}

else if (n == 0) {

cout << "No solution." << endl;

}

else {

cout << "The equation has " << n << " real solution(s): ";

for (int i = 0; i < n; i++) {

cout << x[i] << " ";

}

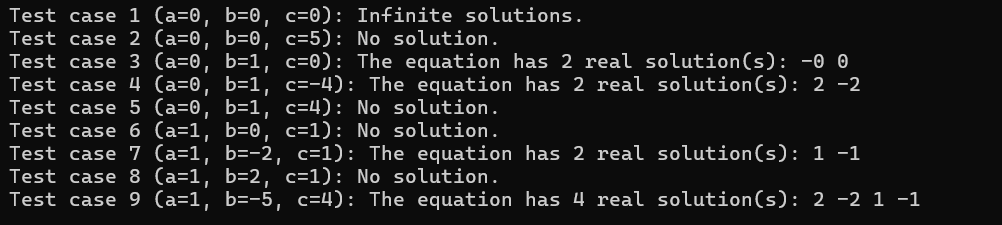
cout << endl;

}

}

return 0;

}



#include <iostream> #include <cmath>

using namespace std;

int solveQuartic(double a, double b, double c, double x[]) { if (a == 0 && b == 0 && c == 0) {

return -1;

}

if (a == 0 && b == 0) { return 0;

}

if (a == 0) {

double y = -c / b; if (y < 0) return 0; x[0] = sqrt(y);

x[1] = -sqrt(y); return 2;

}

double delta = b \* b - 4 \* a \* c; if (delta < 0) return 0;

double y1 = (-b + sqrt(delta)) / (2 \* a); double y2 = (-b - sqrt(delta)) / (2 \* a);

int count = 0; if (y1 >= 0) {

x[count++] = sqrt(y1); x[count++] = -sqrt(y1);

}

if (y2 >= 0 && y2 != y1) { x[count++] = sqrt(y2); x[count++] = -sqrt(y2);

}

return count;

}

int main() {

double a, b, c;

cin >> a >> b >> c;

double x[4];

int n = solveQuartic(a, b, c, x);

if (n == -1) {

cout << " Infinite solutions." << endl;

} else if (n == 0) {

cout << "No solution." << endl;

} else {

cout << " The equation has " << n << " real solution(s): "; for (int i = 0; i < n; i++) {

cout << x[i] << " ";

}

cout << endl;

}

return 0;

}

**---o0o---**

**(End)**