Đỗ Đức Thắng – 23020158 – Tuần 4

Link github: https://github.com/V1ctorious3010/KiemThuPhanMem

Câu hỏi: Trình bày các bước nhằm kiểm thử một đơn vị chương trình theo phương pháp kiểm thử dòng điều khiển với một độ đo kiểm thử cho trước.

Trả lời:

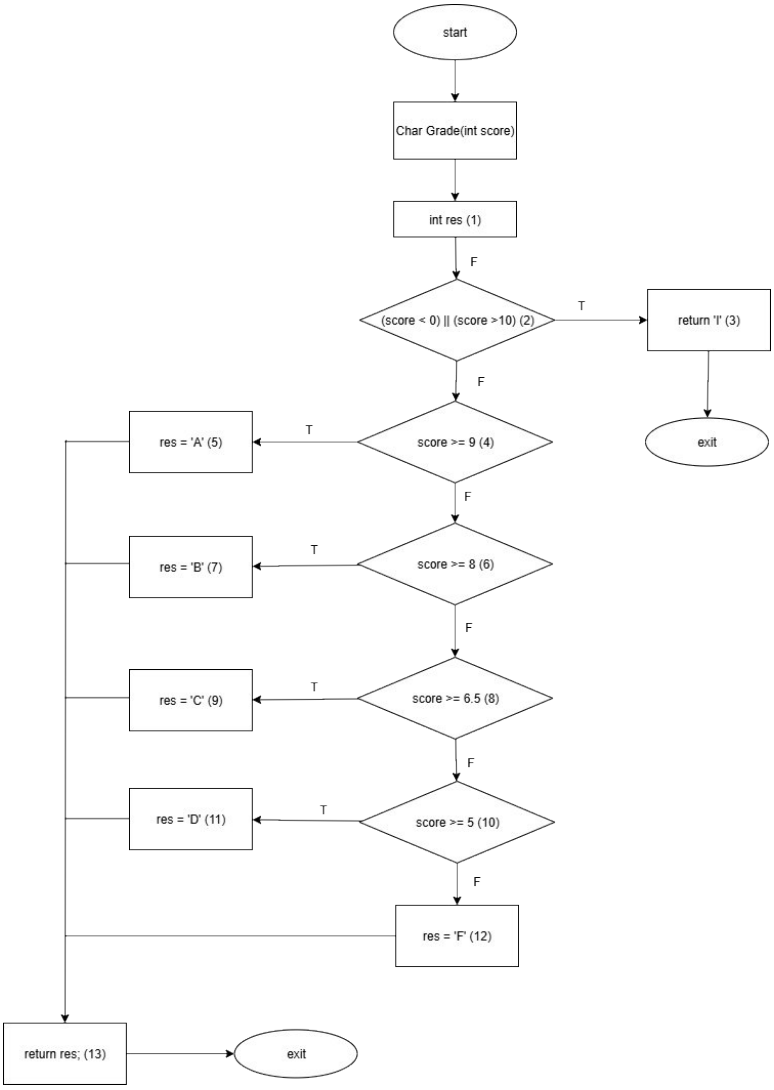
B1: Vẽ đồ thị dòng điều khiển dựa vào mã nguồn

B2: Xác định các đường đi theo độ đo kiểm thử cho trước

B3: Sinh các ca kiểm thử

B4: Thực hiện các ca kiểm thử

C1: Grade

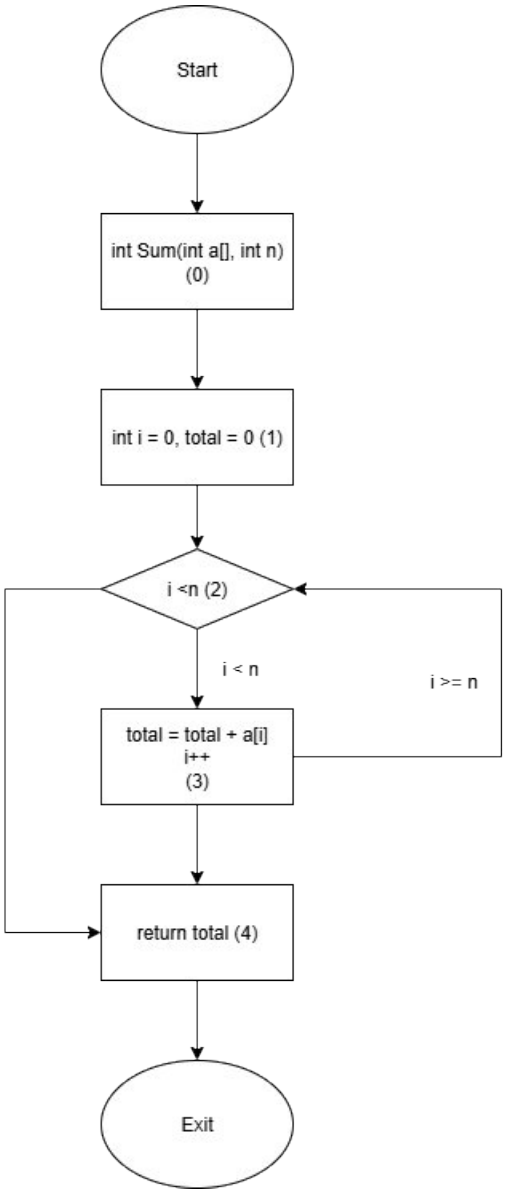


Path	Biểu diễn đường đi	Test case (đầu vào)
Path 1	0 → 1 → 2 (T) → 3	**Grade(-1)**
Path 2	0 → 1 → 2 (F) → 4 (T) → 5 → 13	**Grade(9.2)**
Path 3	0 → 1 → 2 (F) → 4 (F) → 6 (T) → 7 → 13	**Grade(8.6)**
Path 4	0 → 1 → 2 (F) → 4 (F) → 6 (F) → 8 (T) → 9 → 13	**Grade(7.5)**
Path 5	0 → 1 → 2 (F) → 4 (F) → 6 (F) → 8 (F) → 10 (T) → 11 → 13	**Grade(5.5)**
Path 6	0 → 1 → 2 (F) → 4 (F) → 6 (F) → 8 (F) → 10 (F) → 12 → 13	**Grade(4.5)**

C2: 6 paths

Path	Biểu diễn đường đi	Test case (đầu vào)
Path 1	0 → 1 → 2 (T) → 3	**Grade(-1)**
Path 2	0 → 1 → 2 (F) → 4 (T) → 5 → 13	**Grade(9.2)**
Path 3	0 → 1 → 2 (F) → 4 (F) → 6 (T) → 7 → 13	**Grade(8.6)**
Path 4	0 → 1 → 2 (F) → 4 (F) → 6 (F) → 8 (T) → 9 → 13	**Grade(7.5)**
Path 5	0 → 1 → 2 (F) → 4 (F) → 6 (F) → 8 (F) → 10 (T) → 11 → 13	**Grade(5.5)**
Path 6	0 → 1 → 2 (F) → 4 (F) → 6 (F) → 8 (F) → 10 (F) → 12 → 13	**Grade(4.5)**

C2: Sum



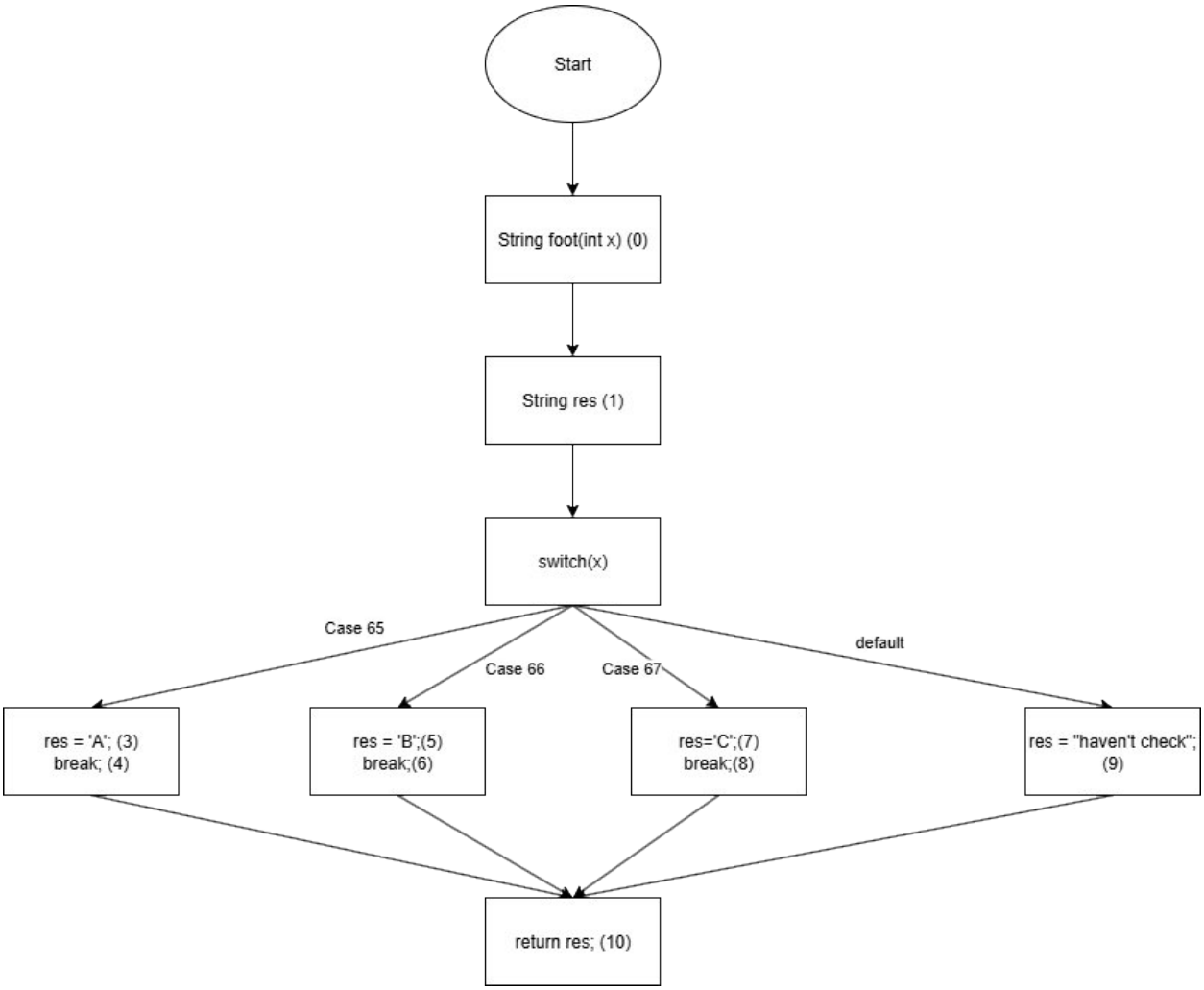
C1: 1 path

Path 1: 0 → 1 → 2 (T) → 3 → 2 (F) → 4
Test case 1: Sum([2], 1)

C2: 1 path

Path 1: 0 → 1 → 2 (T) → 3 → 2 (F) → 4
Test case 1: Sum([9], 1)

C3: foo

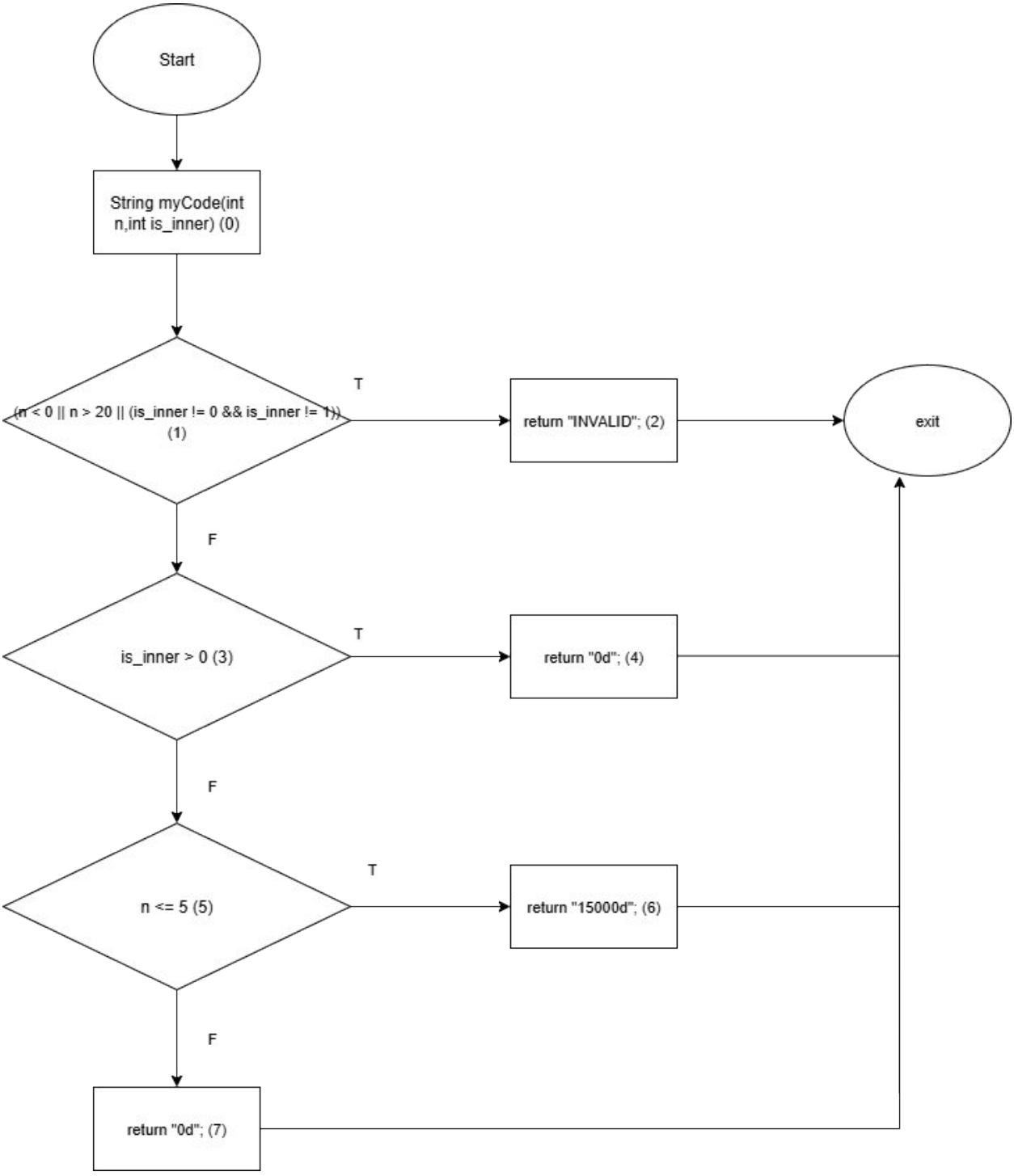


C2: 4 paths

Path	Biểu diễn đường đi	Test case
1	0 → 1 → 2 (case 65) → 3 → 4 → 10	foo(65)
2	0 → 1 → 2 (case 66) → 5 → 6 → 10	foo(66)
3	0 → 1 → 2 (case 67) → 7 → 8 → 10	foo(67)
4	0 → 1 → 2 (case 70) → 9 → 10	foo(69)

Bài làm cá nhân

Đồ thị CFG



Thiết kế ca kiểm thử

Path 1: 0 – 1(T) – 2

Test case: myCode(21, 0)

Path 2: 0 – 1(F) -3(T)- 4

Test case: myCode(20, 1)

Path 3: 0 – 1(F) – 3(F) – 5(T) – 6

Test case: myCode(5, 0)

Path 4: 0 – 1(F) – 3(F) – 5(F) – 7

Test case myCode(15, 0)

ID	n	is_inner	Expected output	Actual output	Result
1	21	0	INVALID	INVALID	✔
2	20	1	0d	0d	✔
3	5	0	15000d	15000d	✔
4	15	0	0d	0d	✔

Test

```
struct TC
{
    int n, is_inner;
    string expect;
};

int main()
{
    vector<TC> tests =
    {
        {21, 0, "INVALID"},
        {20, 1, "0d"},
        {5,  0, "15000d"},
        {15,  0, "0d"},
    };

    for (size_t i = 0; i < tests.size(); ++i)
    {
        const auto& t = tests[i];
        string got = myCode(t.n, t.is_inner);
        assert(got == t.expect);
        cout << "Test " << (i+1) << " passed: fee("
             << t.n << ", " << t.is_inner << ") = " << got << '\n';
    }
    cout << "All tests passed.\n";
    return 0;
}
```