ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Báo cáo môn học:

Kiểm thử và đảm bảo chất lượng phần mềm

Chủ đề: Kiểm thử dòng dữ liệu

Giảng viên: ThS. Nguyễn Thu Trang

Mã lớp môn học: INT3117 7

Sinh viên: Nguyễn Hữu Đồng

Mã sinh viên: 21020760

Lớp: K66I-IT1

Hà Nội – 2024

Mục lục

Chương 1

Bài tập

1.1 Bài 1

1.1.1 Đề bài

Trình bày các bước trong quy trình kiểm thử dòng dữ liệu động.

1.1.2 Bài làm

Các bước trong quy trình kiểm thử dòng dữ liệu:

- Bước 1: Phân tích từ mã nguồn, xây dựng đồ thị luồng điều khiển.
- Bước 2: Xác định độ đo kiểm thử luồng dữ liệu.
- Bước 3: Xác định các đường đi thoả mãn.
- Bước 4: Sinh các ca kiểm thử tương ứng và thực hiện kiểm thử.

1.2 Bài 2

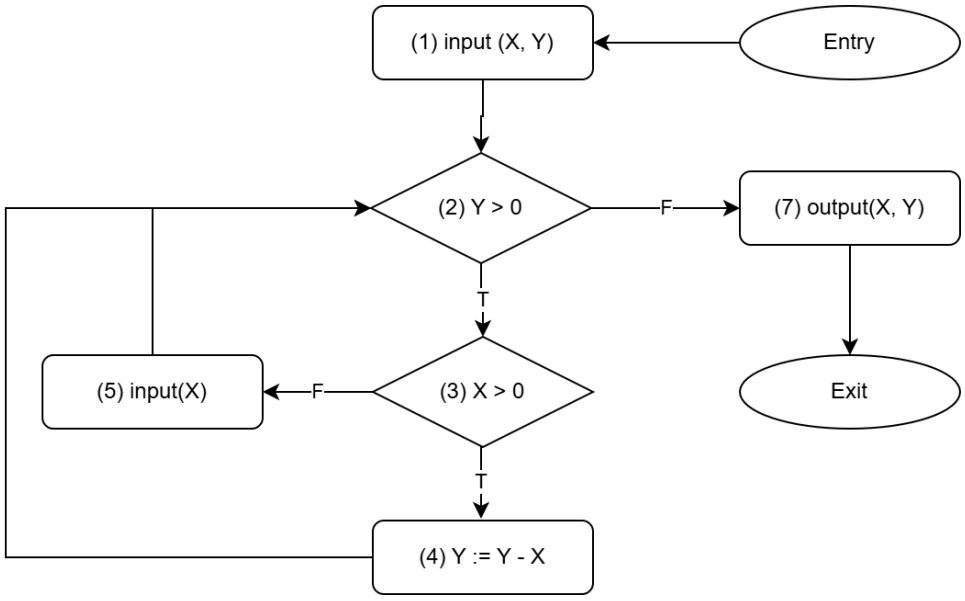
1.2.1 Đề bài

```
1. input(X,Y)
2. while (Y>0) {
3.    if (X>0)
4.        Y := Y-X
      else
5.        input(X)
6.   }
7. output(X,Y)
```

Cho đoạn mã nguồn sau, hãy:
1. Vẽ đồ thị dòng điều khiển (CFG)
2. Xác định các du-pairs cho biến X và Y
3. Sinh đường đi và các ca kiểm thử với độ đo all-use

1.2.2 Bài làm

- Vẽ đồ thị dòng điều khiển (CFG)



- Xác định các du-pairs cho biến X và Y

Var	du-pairs	Def-clear path
X	(1, 3T)	1, 2T, 3T
	(1, 3F)	1, 2T, 3F
	(1, 4)	1, 2T, 3T, 4
	(1, 7)	1, 2F, 7
	(5, 3T)	5, 2T, 3T
	(5, 3F)	5, 2T, 3F
	(5, 4)	5, 2T, 3T, 4
	(5, 7)	5, 2T, 3T, 4, 2F, 7
Y	(1, 2T)	1, 2T
	(1, 2F)	1, 2F
	(1, 4)	1, 2T, 3T, 4
	(1, 7)	1, 2F, 7
	(4, 2T)	4, 2T
	(4, 2F)	4, 2F
	(4, 4)	4, 2T, 3T, 4
	(4, 7)	4, 2F, 7

- Sinh các đường đi các ca kiểm thử với độ đo all-use

Var	**du-pairs**	**Complete paths**	**Input (X, Y)**	**Input(X)** (Optional)	**Input(X)** (Optional)	**Output** (X, Y)
X	(1, 3T)	1, 2T, 3T, 4, 2F, 7	(4, 2)			(4,-2)
	(1, 3F)	1, 2T, 3F, 5, 2T, 3T, 4, 2F, 7	(-2, 2)	4		(4, -2)
	(1, 4)	1, 2T, 3T, 4, 2F, 7	(4, 2)			(4,-2)
	(1, 7)	1, 2F, 7	(4, -2)			(4, -2)
	(5, 3T)	1, 2T, 3F, 5, 2T, 3T, 4, 2F, 7	(-2, 2)	4		(4, -2)
	(5, 3F)	1, 2T, 3F, 5, 2T, 3F, 5, 2T, 3T, 2F, 7	(-2, 2)	-4	4	(4, -2)
	(5, 4)	1, 2T, 3F, 5, 2T, 3T, 4, 2F, 7	(-2, 2)	4		(4, -2)
	(5, 7)	1, 2T, 3F, 5, 2T, 3T, 4, 2F, 7	(-2, 2)	4		(4, -2)
Y	(1, 2T)	1, 2T, 3T, 4, 2F, 7	(4, 2)			(4,-2)
	(1, 2F)	1, 2F, 7	(4, -2)			(4, -2)
	(1, 4)	1, 2T, 3T, 4, 2F, 7	(4, 2)			(4,-2)
	(1, 7)	1, 2F, 7	(4, -2)			(4, -2)
	(4, 2T)	1, 2T, 3T, 4, 2T, 3T, 4, 2F, 7	(-2, 2)	4		(4, -2)
	(4, 2F)	1, 2T, 3T, 4, 2F, 7	(4, 2)			(4,-2)
	(4, 4)	1, 2T, 3T, 4, 2T, 3T, 4, 2F, 7	(-2, 2)	4		(4, -2)
	(4, 7)	1, 2T, 3T, 4, 2F, 7	(4, 2)			(4,-2)

1.3 Bài 3

1.3.1 Đề bài

7. Cho hàm `calFactorial` viết bằng ngôn ngữ C như Đoạn mã 7.7.

   - Hãy liệt kê các câu lệnh ứng với các khái niệm def, $c-use$, và $p-use$ ứng với các biến được sử dụng trong hàm này.
   - Hãy vẽ đồ thị dòng dữ liệu của hàm này.

Đoạn mã 7.7: Mã nguồn C của hàm calFactorial

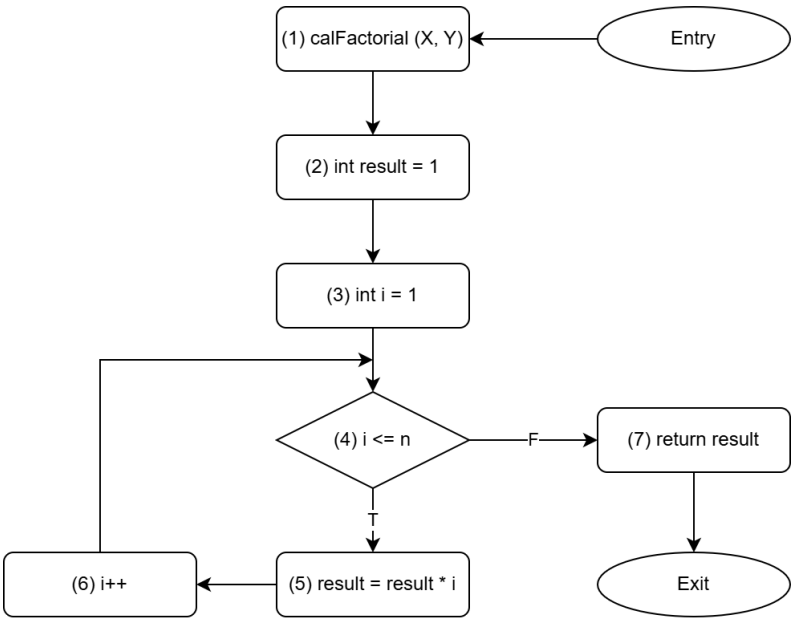
```
1 int calFactorial (int n){
      2 int result = 1;
      3 int i=1;
      4 while (i <= n){
            5 result = result *i;
            6 i++;
        }//end while
      7 return result;
}//the end
```

1.3.2 Bài làm

- Các câu lệnh (đã được đánh số trong hình) tương ứng với các khái niệm def, c-use và p-use ứng với các biến được sử dụng trong hàm.

Biến	def	c-use	p-use
n	1		4
i	3, 6	5, 6	4
result	2, 5	5, 7	

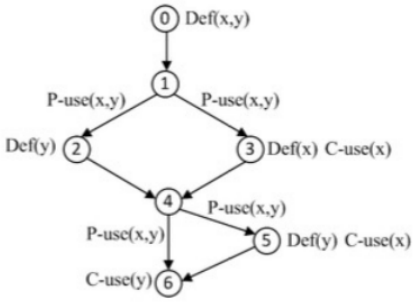
- Vẽ đồ thị dòng điều khiển của hàm



1.4 Bài 4

1.4.1 Đề bài

10. Cho đồ thị dòng dữ liệu như hình 7.11.



Hình 7.11: Một ví dụ về đồ thị dòng dữ liệu và việc sử dụng các biến.

- Hãy xác định tất cả các *Def-clear-path* của các biến x và y.
- Hãy xác định tất cả các du-paths của các biến x và y.
- Hãy xác định tất cả các *All-p-uses/Some-c-uses* và *All-c-uses/Some-p-uses* (dựa vào các chuẩn của kiểm thử dòng dữ liệu).
- Biểu thức của các *p-use*(x, y) tại cạnh (1,3) và (4,5) lần lượt là $x + y = 4$ và $x^2 + y^2 > 17$. Đường đi (0 - 1 - 3 - 4 - 5 - 6) có thực thi được không? Giải thích.
- Tại sao tại đỉnh 3 biến x được định nghĩa và sử dụng nhưng không tồn tại mối quan hệ def-use?

1.4.2 Bài làm

- Tất cả các def-clear-path của biến x và y

Var	Def-clear path
X	0, 1
	0, 1, 2
	0, 1, 2, 4
	0, 1, 2, 4, 5
	0, 1, 2, 4, 5, 6
	0, 1, 2, 4, 6
	3, 4
	3, 4, 5
	3, 4, 5, 6
	3, 4, 6
Y	0, 1
	0, 1, 3
	0, 1, 3, 4
	0, 1, 3, 4, 6
	2, 4
	2, 4, 6
	5, 6

- Tất cả du-paths của các biến x và y

Var	Du-path
X	0, 1
	0, 1, 2, 4
	0, 1, 2, 4, 5
	3, 4
	3, 4, 5
Y	0, 1
	0, 1, 3, 4
	0, 1, 3, 4, 6
	2, 4
	2, 4, 6
	5, 6

- Tất cả các All-p-uses/Some-c-uses và All-c-uses/Some-p-uses

Var	Du-path	All-p-uses/Some-c-uses	All-c-uses/Some-p-uses
X	0, 1	x	x
	0, 1, 2, 4	x	
	0, 1, 2, 4, 5		
	3, 4	x	x
	3, 4, 5		x
Y	0, 1	x	
	0, 1, 3, 4	x	
	0, 1, 3, 4, 6		x
	2, 4	x	
	2, 4, 6		x
	5, 6	x	x

- Nếu biểu thức của các $p - use(x, y)$ tại cạnh (1, 3) và (4, 5) lần lượt là $x + y = 4$ và $x^2 + y^2 > 17$ thì đường đi (0 - 1 - 3 - 4 - 5 - 6) có thể thực thi được.
  Giải thích: Với đầu vào (x, y) = (2, 2), chương trình sẽ chạy đoạn lệnh 0, 1. Đến đoạn lệnh thứ 3, x được define = 4, khi đó p-use $x^2 + y^2 > 17$ được thoả mãn,

chương trình tiếp tục chạy dòng lệnh thứ 5 và 6. Ta có đường đi đầy đủ là (0 - 1 - 3 - 4 - 5 - 6).

- Tại đỉnh 3, biến x được sử dụng trước rồi mới định nghĩa. Vì vậy không tồn tại mối quan hệ def-use tại đỉnh 3 đối với biến x.

1.5 Bài 5

1.5.1 Đề bài

Cho đoạn mã nguồn như hình bên,

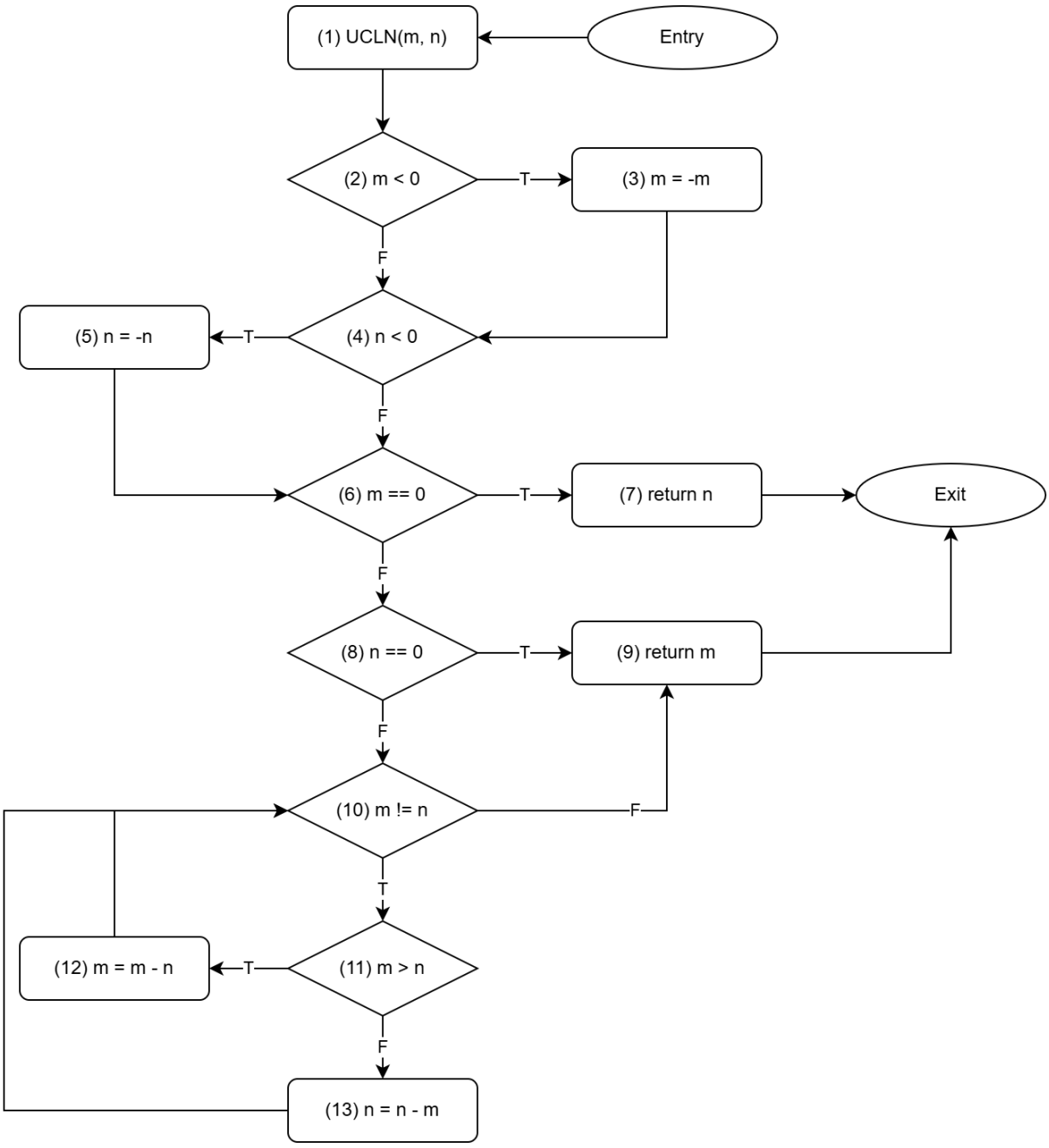
1. Xây dựng CFG cho hàm UCLN với đồ thị C2
2. Sinh đường đi và các ca kiểm thử với độ đo C2
3. Sinh đường đi và các ca kiểm thử với độ đo all-def coverage

Đoạn mã 6.4: Mã nguồn của hàm UCLN

```
int UCLN(int m, int n){
        if (m < 0) m = -m;
        if (n < 0) n = -n;
        if (m == 0)   return n;
        if (n == 0)   return m;
        while (m != n) {
                if(m > n)
                        m = m - n;
                else
                        n = n - m;
        }//end while
        return m;
}
```

1.5.2 Bài làm

- Xây dựng CFG cho hàm UCLN với đồ thị C2



- Đường đi và các ca kiểm thử đảm bảo độ đo C2 (phủ nhánh)

Test case	Path	Input(m, n)	Expected output
TC1	1, 2F, 4T, 5, 6T, 7	(0, -5)	5
TC2	1, 2T, 3, 4T, 5, 6F, 8F, 10F, 9	(-5, -5)	5
TC3	1, 2F, 4F, 6F, 8T	(5, 0)	5
TC4	1, 2F, 3, 4F, 5, 6F, 8F, 10T, 11T, 12, 10F, 9	(10, 5)	5
TC5	1, 2T, 3, 4T, 5, 6F, 8F, 10T, 11F, 13, 10F, 9	(10, 5)	5

- Đường đi và các ca kiểm thử với độ đo all-def coverage cho cả hai biến

Var	du-pair	Complete path	Input(m, n)	Expected output
m	1, 2	1, 2T, 3, 4T, 5, 6F, 8F, 10T, 11T, 12, 10F, 9	(-10, -5)	5
	3, 6F	1, 2T, 3, 4T, 5, 6F, 8F, 10T, 11T, 12, 10F, 9	(-10, -5)	5
	12, 10F	1, 2T, 3, 4T, 5, 6F, 8F, 10T, 11T, 12, 10F, 9	(-10, -5)	5
n	1, 4T	1, 2F, 4T, 5, 6F, 8F, 10T, 11F, 13, 10F, 9	(5, -10)	5
	5, 8F	1, 2F, 4T, 5, 6F, 8F, 10T, 11F, 13, 10F, 9	(5, -10)	5
	13, 10F	1, 2F, 4T, 5, 6F, 8F, 10T, 11F, 13, 10F, 9	(5, -10)	5

Chương 2

Kiểm thử chương trình

Yêu cầu: Báo cáo phân tích, thiết kế các ca kiểm thử, và kiểm thử chương trình của bạn với độ phủ all-uses.

2.1 Mô tả bài toán

Bài toán: Tính số tiền cần phải thanh toán cho dịch vụ taxi. Công ty X cung cấp dịch vụ taxi với mức giá cụ thể như sau:

- Phí mở cửa: 30,000 đồng cho quãng đường ban đầu (dưới 1km)
- Từ km thứ 2 đến km thứ 20: 20,000 đồng cho mỗi km tiếp theo
- Từ km thứ 21 trở đi: 10,000 đồng cho mỗi km
- Mỗi người thêm vào (từ người thứ 2) sẽ tăng thêm 20% vào tổng chi phí cơ bản của 1 người.

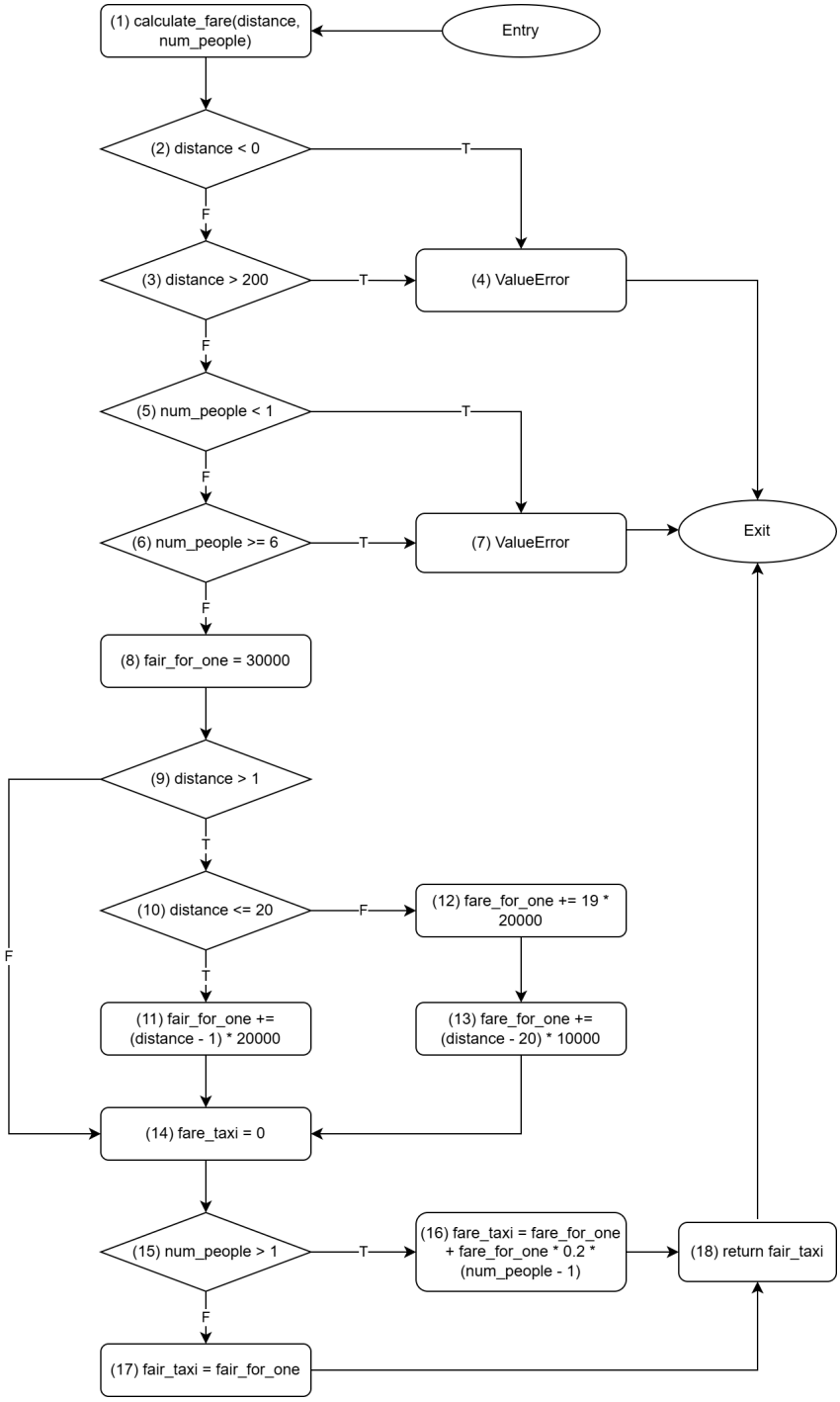
Ngoài ra, do giới hạn của phương tiện và để đảm bảo chất lượng phục vụ, dịch vụ taxi của công ty X chỉ vận chuyển tối đa 6 người trên xe (bao gồm cả tài xế), và giới hạn quãng đường di chuyển là 200km.

2.2 Mã nguồn

Chi tiết mã nguồn có sẵn tại: `https://github.com/huudong03uet/Testing/tree/main/kiem_thu_luong_du_lieu/`

2.3 Đồ thị dòng điều khiển

Xây dựng đồ thị dòng điều khiển cho hàm $calculate\_fare(distance : float, num\_people : int)$:



2.4 Phân tích, thiết kế cách ca kiểm thử

- Các câu lệnh tương ứng với các khái niệm def, c-use và p-use ứng với các biến được sử dụng trong hàm.

Var	**def**	**c-use**	**p-use**
distance	1	11, 13	2, 3, 9, 10
num_people	1	16	5, 6, 15
fair_for_one	8, 11, 12, 13	11, 12, 13, 16, 17	
fair_taxi	14, 16, 17	18	

- Từ đồ thị dòng điều khiển, ta xác định các đường đi tương ứng để đạt độ phủ all-uses.

No	Var	du-pair	Complete Path
1	distance	(1, 2T)	1, 2T, 4
2		(1, 2F)	1, 2F, 3F, 5F, 6F, 8, 9T, 10F, 12, 13, 14, 15F, 17, 18
3		(1, 3T)	1, 2F, 3T, 4
4		(1, 3F)	1, 2F, 3F, 5F, 6F, 8, 9T, 10F, 12, 13, 14, 15F, 17, 18
5		(1, 9T)	1, 2F, 3F, 5F, 6F, 8, 9T, 10F, 12, 13, 14, 15F, 17, 18
6		(1, 9F)	1, 2F, 3F, 5F, 6F, 8, 9F, 14, 15T, 16, 18
7		(1, 10T)	1, 2F, 3F, 5F, 6F, 8, 9T, 10T, 11, 14, 15T, 16, 18
8		(1, 10F)	1, 2F, 3F, 5F, 6F, 8, 9T, 10F, 12, 13, 14, 15F, 17, 18
9		(1, 11)	1, 2F, 3F, 5F, 6F, 8, 9T, 10T, 11, 14, 15T, 16, 18
10		(1, 13)	1, 2F, 3F, 5F, 6F, 8, 9T, 10F, 12, 13, 14, 15F, 17, 18
11	num_people	(1, 5T)	1, 2F, 3F, 5T, 7
12		(1, 5F)	1, 2F, 3F, 5F, 6F, 8, 9T, 10F, 12, 13, 14, 15F, 17, 18
13		(1, 6T)	1, 2F, 3F, 5F, 6T, 7
14		(1, 6F)	1, 2F, 3F, 5F, 6F, 8, 9T, 10F, 12, 13, 14, 15F, 17, 18
15		(1, 15T)	1, 2F, 3F, 5F, 6F, 8, 9F, 14, 15T, 16, 18
16		(1, 15F)	1, 2F, 3F, 5F, 6F, 8, 9T, 10F, 12, 13, 14, 15F, 17, 18
17		(1, 16)	1, 2F, 3F, 5F, 6F, 8, 9F, 14, 15T, 16, 18
18	fair_for_one	(8, 11)	1, 2F, 3F, 5F, 6F, 8, 9T, 10T, 11, 14, 15T, 16, 18
19		(8, 12)	1, 2F, 3F, 5F, 6F, 8, 9T, 10F, 12, 13, 14, 15F, 17, 18
20		(8, 16)	1, 2F, 3F, 5F, 6F, 8, 9F, 14, 15T, 16, 18
21		(8, 17)	1, 2F, 3F, 5F, 6F, 8, 9F, 14, 15F, 17, 18
22		(11, 16)	1, 2F, 3F, 5F, 6F, 8, 9T, 10T, 11, 14, 15T, 16, 18
23		(11, 17)	1, 2F, 3F, 5F, 6F, 8, 9T, 10T, 11, 14, 15F, 17, 18
24		(12, 13)	1, 2F, 3F, 5F, 6F, 8, 9T, 10F, 12, 13, 14, 15F, 17, 18
25		(13, 16)	1, 2F, 3F, 5F, 6F, 8, 9T, 10F, 12, 13, 14, 15T, 16, 18
26		(13, 17)	1, 2F, 3F, 5F, 6F, 8, 9T, 10F, 12, 13, 14, 15F, 17, 18
27	fair_taxi	(16, 18)	1, 2F, 3F, 5F, 6F, 8, 9T, 10F, 12, 13, 14, 15T, 16, 18
28		(17, 18)	1, 2F, 3F, 5F, 6F, 8, 9T, 10F, 12, 13, 14, 15F, 17, 18

- Từ đồ thị dòng điều khiển, ta xác định các test case tương ứng với complete path ở trên để đạt độ phủ all-uses

No	**Var**	**du-pair**	**Input** distance, num_people	**Expected output**
1	distance	(1, 2T)	(-1, 1)	ValueError
2		(1, 2F)	(100, 1)	1210000
3		(1, 3T)	(300, 1)	ValueError
4		(1, 3F)	(100, 1)	1210000
5		(1, 9T)	(100, 1)	1210000
6		(1, 9F)	(0.5, 2)	36000
7		(1, 10T)	(10, 2)	252000
8		(1, 10F)	(100, 1)	1210000
9		(1, 11)	(10, 2)	252000
10		(1, 13)	(100, 1)	1210000
11	num_people	(1, 5T)	(10, 0)	ValueError
12		(1, 5F)	(100, 1)	1210000
13		(1, 6T)	(10, 6)	ValueError
14		(1, 6F)	(100, 1)	1210000
15		(1, 15T)	(0.5, 2)	36000
16		(1, 15F)	(100, 1)	1210000
17		(1, 16)	(0.5, 2)	36000
18	fair_for_one	(8, 11)	(10, 2)	252000
19		(8, 12)	(100, 1)	1210000
20		(8, 16)	(0.5, 2)	36000
21		(8, 17)	(0.5, 1)	30000
22		(11, 16)	(10, 2)	252000
23		(11, 17)	(10, 1)	210000
24		(12, 13)	(100, 1)	1210000
25		(13, 16)	(100, 2)	1452000
26		(13, 17)	(100, 1)	1210000
27	fair_taxi	(16, 18)	(100, 2)	1452000
28		(17, 18)	(100, 1)	1210000

2.5 Kết quả kiểm thử

- Kết quả kiểm thử được trình bày trong bảng:

Test case	**Input** distance, num_people	**Expected output**	**Output**	**Result**
TC1	(-1, 1)	ValueError	ValueError	Pass
TC2	(300, 1)	ValueError	ValueError	Pass
TC3	(10, 0)	ValueError	ValueError	Pass
TC4	(10, 6)	ValueError	ValueError	Pass
TC5	(10, 1)	210000	210000	Pass
TC6	(10, 2)	252000	252000	Pass
TC7	(0.5, 1)	30000	30000	Pass
TC8	(0.5, 2)	36000	36000	Pass
TC9	(100, 1)	1210000	1210000	Pass
TC10	(100, 2)	1452000	1452000	Pass

- Kết luận
  - Chương trình pass 100% qua tất cả các test.
  - Không tìm thấy lỗi của chương trình với bộ kiểm thử này.