1. Bài 1 : Trình bày các bước trong quy trình kiểm thử dòng dữ liệu động.

Bước 1: Vẽ đồ thị luồng điều khiển (CFG) : từ sources code

* Nắm rõ 1 số khái niệm :
* Def: Một câu lệnh thực hiện việc gán giá trị cho một biến được gọi là câu lệnh

def của biến

vd: y = 10; là câu lệnh def của biến y

* Use: Một câu lệnh sử dụng một biến được gọi là câu lệnh use của biến

vd: x = y; là câu lệnh def của biến x, và là câu lệnh use của biến y

* P-use (predicate-use): câu lệnh sử dụng một biến trong biểu thức điều kiện

vd: if(x == 0) là câu lệnh p-use của biến x

* C-use (computation-use): các câu lệnh use còn lại

vd: x = y; là câu lệnh c-use của biến y

print(y): là câu lệnh c-use của biến y

* Def-clear path: 1 đường đi từ (i, 𝑛!,…,𝑛") được gọi là def-clear path của biến v

từ i đến 𝑛" nếu v được định nghĩa tại i, các đỉnh 𝑛!,…, 𝑛" không chứa câu lệnh

def của v

* Du-pair (Def-use pair): (d,u) là cặp def-use của biến v . trong đó : d là đỉnh tương ứng với câu lệnh def của v, u là đỉnh tương ứng với câu lệnh use của v và tồn tại ít nhất một def-clear path của biến v từ d tới u

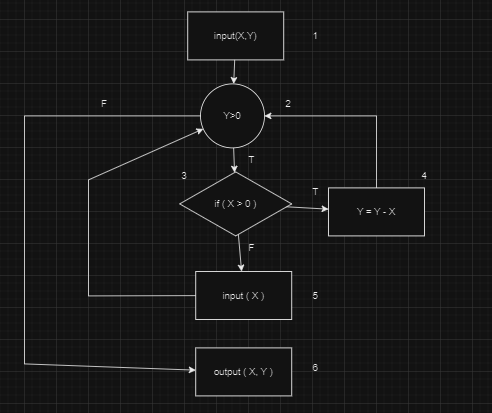
Bước 2: Lựa chọn tiêu chí kiểm thử luồng dữ liệu : Các tiêu chí kiểm thử luồng dữ liệu tập trung vào việc kiểm tra mối quan hệ giữa các biến tại các điểm khai báo (definition) và sử dụng (use) trong chương trình. Sau đây là 1 số tiêu chí kiểm thử dòng dữ liệu phổ biến

* All-definitions: Kiểm thử tất cả các điểm def của các biến trong chương trình.
* All-uses: Kiểm thử tất cả các điểm uses (sử dụng trong phép tính ( c-use) và sử dụng trong điều kiện(p-use)) của biến sau mỗi lần khai báo.
* All-du-paths: Kiểm thử tất cả các đường đi từ điểm khai báo đến điểm sử dụng biến mà không có khai báo mới xen vào.

Bước 3 : Xác định các đường đi trên CFG thoả mãn tiêu chí kiểm thử đã chọn  
Bước 4 : Sinh các ca kiểm thử tương ứng : Dựa trên các đường đi đã xác định, chúng ta sẽ tạo ra các ca kiểm thử với các giá trị đầu vào sao cho đảm bảo kiểm thử được tất cả các đường đi.

1. Bài 2

2.1 : vẽ đồ thị dòng điều khiển ( CFG)



2.2 xác định các du-pair cho biến X và Y

|  |  |  |
| --- | --- | --- |
| **Var** | **def** | **DU-pair** |
| x | 1 | (1,3(T)) |
|  |  | (1,3(F)) |
|  |  | (1,4) |
|  |  | (1,6) |
|  | 5 | (5,3(T)) |
|  |  | (5,3(F)) |
|  |  | (5,4) |
|  |  | (5,6) |
| y | 1 | (1,2(T)) |
|  |  | (1,2(F)) |
|  |  | (1,4) |
|  |  | (1,6) |
|  | 4 | (4,2(T)) |
|  |  | (4,2(F)) |
|  |  | (4,6) |

2.3 : sinh các đường đi và các ca kiểm thử với độ đo all-use

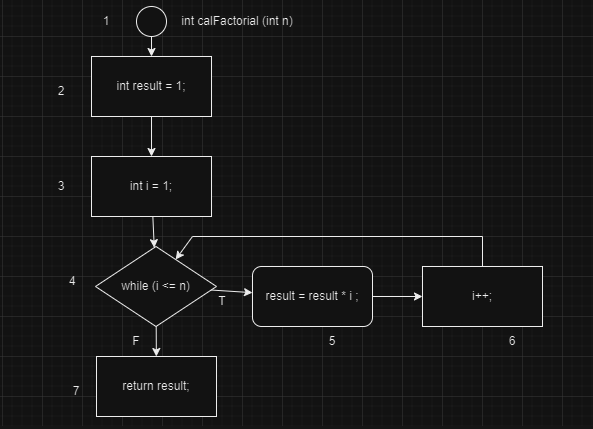
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Var** | **def** | **DU-pair** | **Def-clear path** | **Completed path** | **Input**  **(X,Y)** |
| x | 1 | (1,3(T)) | 1,2(T),3(T) | 1,2(T),3(T),4,2(F),6 | (3,2) |
|  |  | (1,3(F)) | 1,2(T),3(F) | 1,2(T),3(F),5,2(T),3(T),4,2(F),6 | (-1,2)->(3,2) |
|  |  | (1,4) | 1,2(T),3(T),4 | 1,2(T),3(T),4,2(F),6 | (3,2) |
|  |  | (1,6) | 1,2(F),6 | 1,2(F),6 | (3,-1) |
|  | 5 | (5,3(T)) | 5,2(T),3(T) | 1,2(T),3(F),5,2(T),3(T),4,2(F),6 | (-1,2)->(3,2) |
|  |  | (5,3(F)) | 5,2(T),3(F) | 1,2(T),3(F),5,2(T),3(F),5,2(T),3(T),4,2(F),6 | (-1,2)  ->(-2,2)  ->(3,2) |
|  |  | (5,4) | 5,2(T),3(T),4 | 1,2(T),3(F),5,2(T),3(T),4,2(F),6 | (-1,2)->(3,2) |
|  |  | (5,6) | 5,2(F),6 | 1,2(T),3(F),5,2(F),6 | Vô nghiệm |
| y | 1 | (1,2(T)) | 1,2(T) | 1,2(T),3(T),4,2(F),6 | (3,2) |
|  |  | (1,2(F)) | 1,2(F) | 1,2(F),6 | (3,-1) |
|  |  | (1,4) | 1,2(T),3(T),4 | 1,2(T),3(T),4,2(F),6 | (3,2) |
|  |  | (1,6) | 1,2(F),6 | 1,2(F),6 | (3,-1) |
|  | 4 | (4,2(T)) | 4,2(T) | 1,2(T),3(T),4,2(T),3(T),4,2(F),6 | (2,3) |
|  |  | (4,2(F)) | 4,2(F) | 1,2(T),3(T),4,2(F),6 | (3,2) |
|  |  | (4,6) | 4,2(F),6 | 1,2(T),3(T),4,2(F),6 | (3,2) |

1. Bài 3

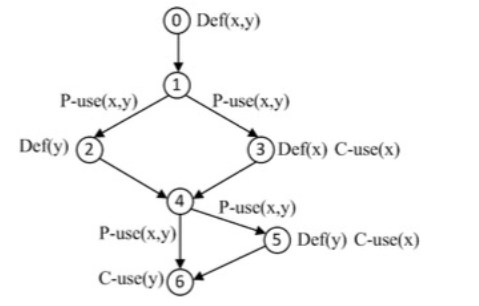
3.1 : liệt kê các câu lệnh def, c-use, p-use ứng với các biến ở trong sources

|  |  |  |  |
| --- | --- | --- | --- |
| **Var** | **def** | **p-use** | **c-use** |
| n | 1 | 4 |  |
| Result | 2,5 |  | 5,7 |
| i | 3,6 | 4 | 6 |

3.2 vẽ đồ thị dòng điều khiển



1. Bài 4



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Var** | **def** | **Def-clear path** | **DU-pair** | **All p-use**  **/some**  **c-use** | **All c-use /some**  **p-use** |
| x | 0 | 0,1 | (0,1) |  |  |
|  |  | 0,1,2 |  |  |  |
|  |  | 0,1,3 |  |  |  |
|  |  | 0,1,2,4 | (0,4) |  |  |
|  |  | 0,1,2,4,6 |  |  |  |
|  |  | 0,1,2,4,5 | (0,5) |  |  |
|  |  | 0,1,2,4,5,6 |  |  |  |
|  | 3 | 3,4 | (3,4) |  |  |
|  |  | 3,4,5 | (3,5) |  |  |
|  |  | 3,4,6 |  |  |  |
|  |  | 3,4,5,6 |  |  |  |
| y | 0 | 0,1 | (0,1) |  |  |
|  |  | 0,1,3 |  |  |  |
|  |  | 0,1,3,4 | (0,4) |  |  |
|  |  | 0,1,3,4,6 | (0,6) |  |  |
|  | 2 | 2,4 | (2,4) |  |  |
|  |  | 2,4,6 | (2,6) |  |  |
|  | 5 | 5,6 | (5,6) |  |  |

4.3 Biểu thức của các p-use(x,y) tại cạnh (1,3) và (4,5) lần lượt là x + y = 4 và x2 + y2 > 17. Đường đi (0 - 1 - 3- 4 - 5 - 6) có thực thi được không? Giải thích.

Đường đi (0 - 1 - 3- 4 - 5 – 6) được thực thi. Vì:

Sau khi đi qua lệnh p-use tại 1 (x + y = 4 ) ta có max(x2 + y2) = 16 , tới câu lệnh 3 : def(x),c-use(x) => x ở câu lệnh số 3 đã được redef giá trị ( có thể câu lệnh 3 có dạng x = x …) . tới cạnh (4,5) điều kiện x2 + y2 > 17 hoàn toàn có khả năng được thỏa mãn . nên đường đi (0 - 1 - 3- 4 - 5 – 6) được thực thi.

4.4 Tại sao tại đỉnh 3 biến x được định nghĩa và sử dụng nhưng không tồn tại mối quan hệ def-use?

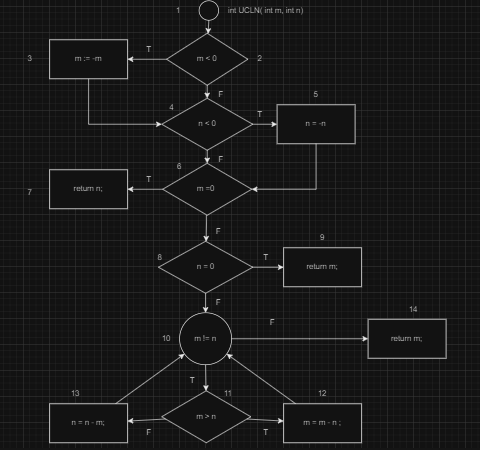
Vì trong 1 câu lệnh vừa định nghĩa vừa sử dụng với biến, cụ thể tại đỉnh 3 có def(x),c-use(x) thì c-use(x) sẽ được thực thi trước def(x).

Def-use pair : sẽ thực thi def trước rồi mới đến đỉnh c-use.

=> tại đỉnh 3 sẽ không tồn tại mối quan hệ def-use.

1. Bài 5

5.1 xây dựng CFG với độ đo C2



5.2 sinh đường đi và các ca kiểm thử với độ đo C2