# Kiểm thự hộp trắng

- Structural/Clear box/Glass box testing
- Thiết kế các trường hợp kiểm thử dựa vào cấu trúc của thủ tục để suy dẫn các trường hợp cần kiểm thử
- Nguyên tắc
  - Thực hiện mọi đường dẫn độc lập ít nhất một lần
  - Thực hiện mọi điều kiện logic trên True/False
  - Thực hiện mọi vòng lặp tại các biên và trong phạm vi hoạt động
  - Thực hiện mọi cấu trúc dữ liệu bên trong để đảm bảo tính hợp lệ

# Kiểm thự hộp trắng

- 2 hướng tiếp cận
  - Kiểm thử đường dẫn cơ sở (Basic path testing)
  - Kiểm thử cấu trúc điều kiện (Control structure testing)

# Kiểm thử đường dẫn cơ sở

- Đảm bảo tất cả đường dẫn độc lập (independent path) điều được kiểm thử
- Đường dẫn độc lập là đường dẫn đi từ đầu đến cuối chương trình mà không chứa đường dẫn độc lập khác
- Tập đường dẫn độc lập → tập cơ sở (basic set)

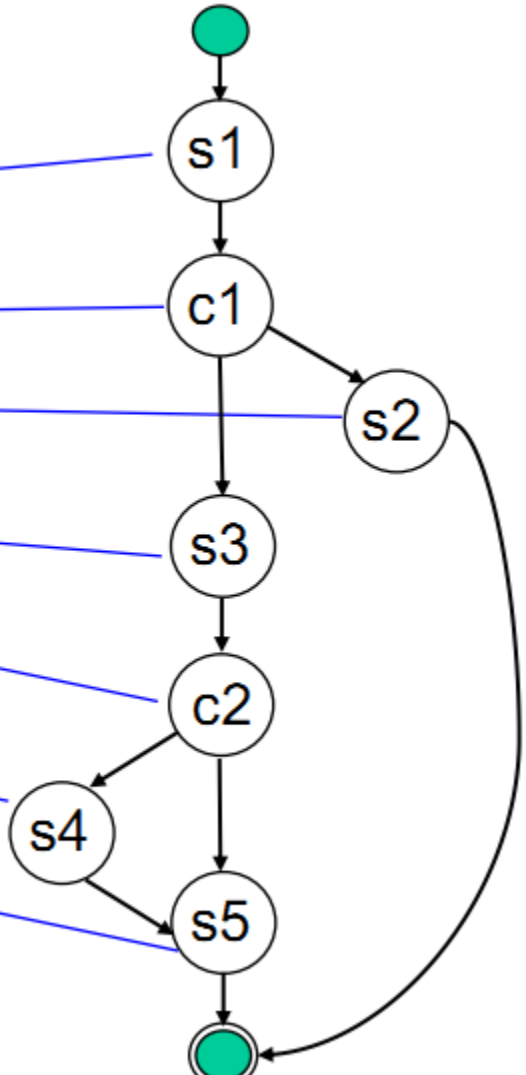
# Kiểm thử đường dẫn cơ sở

- Các bước thực hiện
  - Bước 1: Vẽ đồ thị lưu trình (flowgraph)
  - Bước 2: Xác định độ phức tạp Cyclomat của đồ thị lưu trình
  - Bước 3: Xác định tập cơ sở các đường dẫn độc lập
  - Bước 4: Thiết kế test case cho mỗi đường dẫn độc lập

# B1: Vẽ đồ thị lưu trình

- Thí dụ :

```
1   float foo(int a, int b, int c, int d) {
2     float e;
3     if (a==0)
4       return 0;
5     int x = 0;
6     if ((a==b) || ((c==d) && bug(a)))
7       x = 1;
8     e = 1/x;
9     return e;
10  }
```



# B2: Xác định độ phức tạp cyclomat

- → cho biết số lượng đường dẫn độc lập
  - V(G) = R(số vùng) = 3
  - V(G) = P(số đỉnh điều kiện)+1 = 2+1 = 3
  - V(G) = E(số cạnh)–N(số đỉnh)+2 = 10-9+2 = 3

# B3: Tìm tập cơ sở các đường dẫn độc lập

- Tìm 1 đường dẫn từ đầu đến cuối chương trình
- Tìm đường dẫn mới có đi qua một cạnh mới mà không trùng với các đường dẫn trước đó
- Làm cho đến khi đủ số lượng đường dẫn
- Ví dụ:
  - Đường dẫn 1: S1→C1→S3→C2→S5
  - Đường dẫn 2: S1→C1→S2
  - Đường dẫn 3: S1→C1→S3→C2→S4→S5

# B4: Thiết kế test case cho từng đường dẫn độc lập

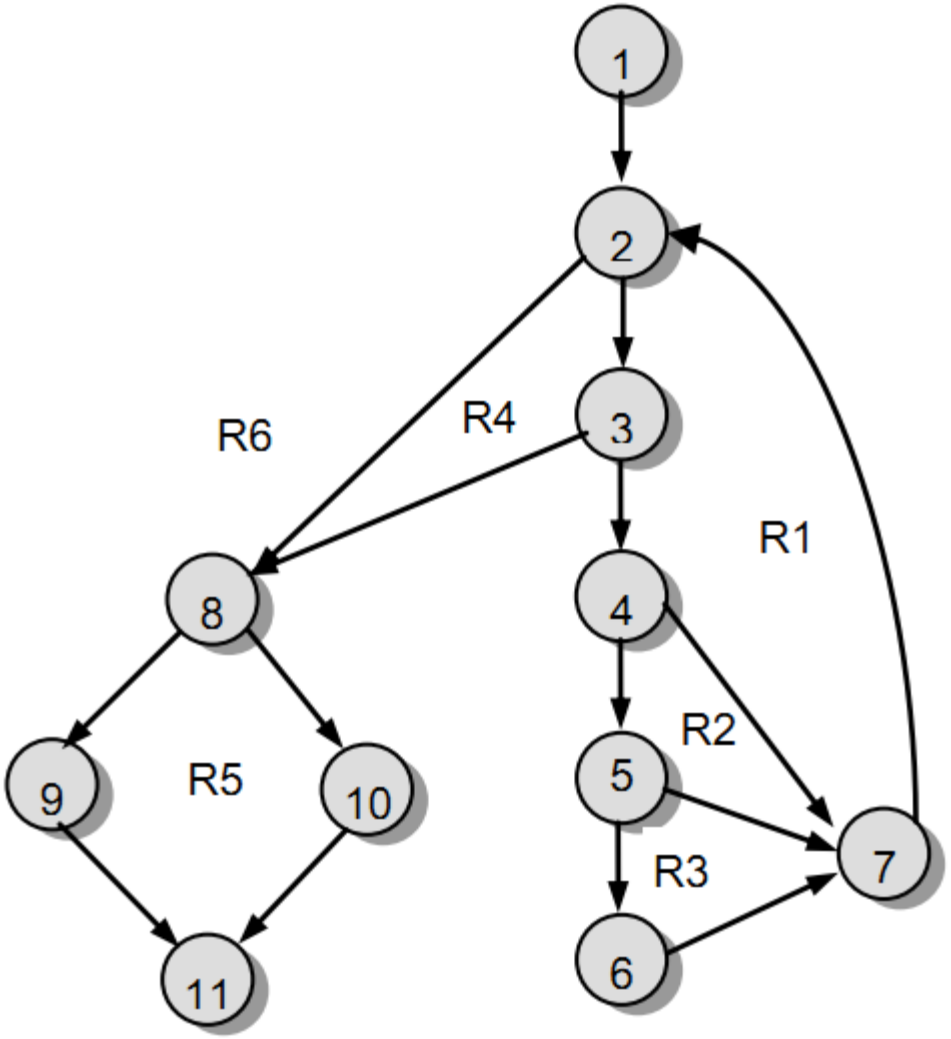
- Test case
  - Đầu vào: …
  - Đầu ra mong muốn: …
  - Mục đích: …
- Ví dụ test case cho đường dẫn 1
  - Đường dẫn 1: S1→C1→S3→C2→S5
  - Đầu vào: a=1, b=0, c=1, d=0
  - Đầu ra mong muốn:
    - x=0, e=1/x => Divide by zero error

# B4: Thiết kế test case cho từng đường dẫn độc lập

| #TC | a | b | c | d | Expected Output |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | Divide by zero error |
| 2 | 0 | 0 | 0 | 0 | 0 |
| 3 | 1 | 1 | 1 | 0 | 1 |

# Kiểm thử đường dẫn cơ sở

- Bước 1: đồ thị lưu trình
  - Đỉnh
  - Cung
  - Đỉnh điều kiện
  - Vùng



# Kiểm thử đường dẫn cơ sở

- Bước 2: Xác định độ phức tạp cyclomat → cho biết số lượng đường dẫn độc lập
  - V(G) = R(số vùng) = 6
  - V(G) = P(số đỉnh điều kiện)+1 = 5+1 = 6
  - V(G) = E(số cạnh)–N(số đỉnh)+2 = 17-13+2 = 6

# Kiểm thử đường dẫn cơ sở

- Bước 3: tìm tập cơ sở các đường dẫn độc lập
  - Tìm 1 đường dẫn từ đầu đến cuối chương trình
  - Tìm đường dẫn mới có đi qua một cạnh mới mà không trùng với các đường dẫn trước đó
  - Làm cho đến khi đủ số lượng đường dẫn
- Ví dụ:
  - Đường dẫn 1: 1→2→8→9→11
  - Đường dẫn 2: 1→2→8→10→11
  - Đường dẫn 3: 1→2→3→8→9→11
  - Đường dẫn 4: 1→2→3→4→7→2→…
  - Đường dẫn 5: 1→2→3→4→5→7→2→…
  - Đường dẫn 6: 1→2→3→4→5→6→7→2→…

# Kiểm thử đường dẫn cơ sở

- Bước 4: thiết kế test case cho từng đường dẫn độc lập
- Ví dụ:
  - Test case cho đường dẫn 1
    - Đầu vào: …
    - Đầu ra mong muốn: …
    - Mục đích: …

# Kiểm thử cấu trúc điều kiện

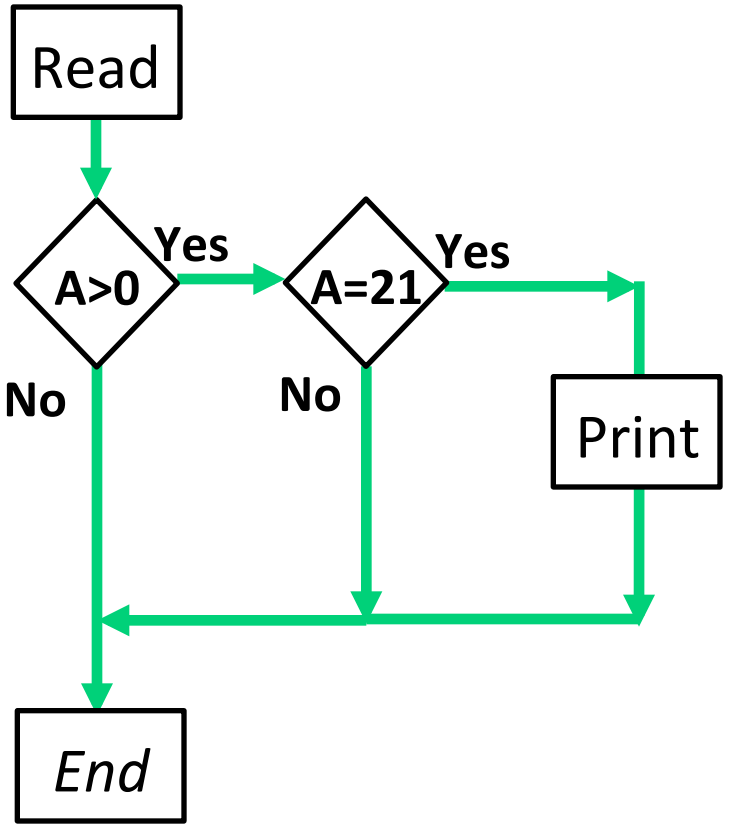
- Kiểm thử dòng điều khiển (Control-flow/ Coverage testing)
- Kiểm thử dòng dữ liệu (Data flow testing)
- Kiểm thử vòng lặp (loop testing)

# Kiểm thử dòng điều khiển

- Coverage dùng để đánh giá tính phủ của tập test case
  - Statement coverage
  - Decision/branch coverage
  - Condition coverage
  - Path coverage

# Ví dụ

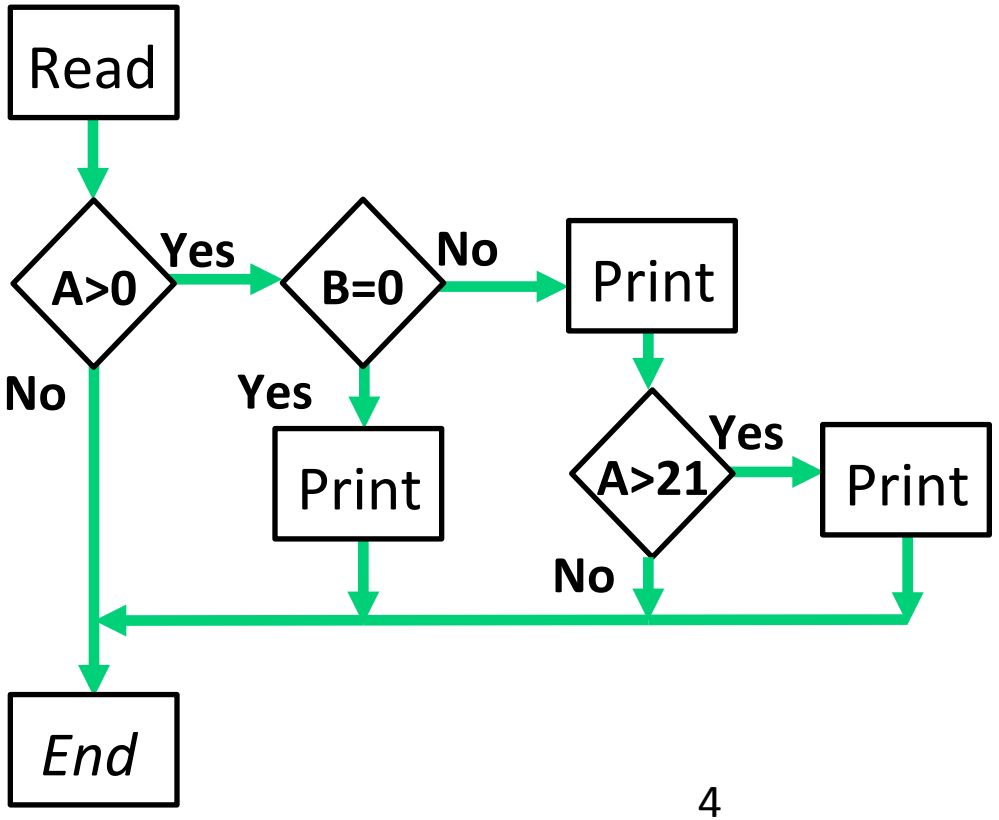
```
Read A
IF A > 0 THEN
    IF A  = 21 THEN
     Print “Key”
    ENDIF
ENDIF
```




- Cyclomatic complexity: 3
- Minimum tests to achieve:
  - Statement coverage: 1
  - Branch coverage: 3

# Ví dụ

```
Read A
Read B
IF A > 0 THEN
    IF B  = 0 THEN
     Print “No values”
    ELSE
     Print B
     IF A > 21 THEN
        Print A
     ENDIF
    ENDIF
ENDIF
```




- Cyclomatic complexity: 4
- Minimum tests to achieve:
  - Statement coverage: 2
  - Branch coverage: 4

# Kiểm thử dòng dữ liệu

- Một biến (variable)
  - Được xác định (define): được gán hay thay đổi giá trị
  - Được sử dụng (use): tính toán (c-use) hay điều kiện (p-use)
- Def-use path: đường dẫn từ def đến use của một biến
- Dữ liệu test được tạo ra để phủ tất cả các def-use

# Kiểm thử dòng dữ liệu

- Ví dụ

| | | |
|---|---|---|
| 1 | sum = 0 | *sum, def* |
| 2 | read (n), | *n, def* |
| 3 | i = 1 | *i, def* |
| 4 | while (i <= n) | *i, n p-sue* |
| 5 | read (number) | *number, def* |
| 6. | sum = sum + number | *sum, def, sum, number, c-use* |
| 7 | i = i + 1 | *i, def, c-use* |
| 8 | end while | |
| 9 | print (sum) | *sum, c-use* |

# Kiểm thử dòng dữ liệu

- Ví dụ

Table for sum

| pair id | def | use |
|---|---|---|
| 1 | 1 | 6 |
| 2 | 1 | 9 |
| 3 | 6 | 6 |
| 4 | 6 | 9 |

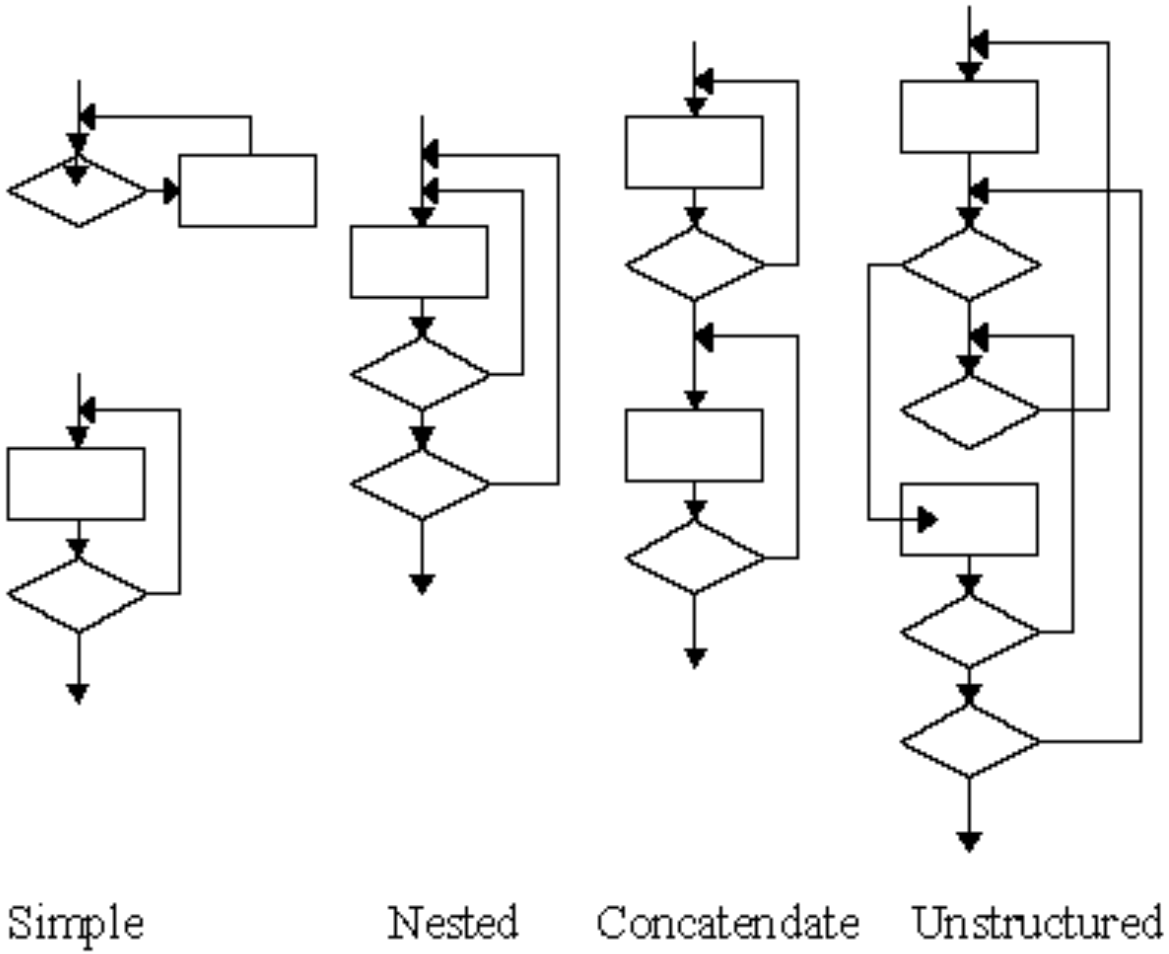
Table for i

| pair id | def | use |
|---|---|---|
| 1 | 3 | 4 |
| 2 | 3 | 7 |
| 3 | 7 | 7 |
| 4 | 7 | 4 |

# Kiểm thử vòng lặp

- Kiểm tra tính hợp hệ của cấu trúc vòng lặp
- Bốn dạng vòng lặp:
  - Lặp đơn (simple loops)
  - Lặp móc nối (concatenated loops)
  - Lặp lồng nhau (nested loops)
  - Lặp không cấu trúc (unstructured loops)

# Kiểm thử vòng lặp



# Kiểm thử vòng lặp

- Kiểm thử vòng lặp đơn (Simple loop)
  - Lặp 0 lần
  - Lặp 1 lần
  - Lặp 2 lần
  - Lặp k lần
  - Lặp n-1, n, n+1 lần với n là số vòng lặp lớn nhất có thể

# Kiểm thử vòng lặp

```
public class loopdemo
{
    private int[] numbers = {5,-3,8,-12,4,1,-20,6,2,10};

    /** Compute total of numItems positive numbers in the array
     *  @param numItems how many items to total, maximum of 10.
     */
    public int findTotal(int numItems)
    {
        int total = 0;
        if (numItems <= 10)
        {
            for (int count=0; count < numItems; count = count + 1)
            {
                if (numbers[count] > 0)
                {
                    total = total + numbers[count];
                }
            }
        }
        return total;
    }
}
```

| numItems |
|---|
| 0 |
| 1 |
| 2 |
| 5 |
| 9 |
| 10 |
| 11 |

# Kiểm thử vòng lặp

- Kiểm thử vòng lặp lồng nhau (nested loops)
  - Xem vòng lặp trong cùng là vòng lặp đơn
  - Tất cả các vòng lặp bên ngoài chọn giá trị lặp nhỏ nhất
  - Tiếp tục với vòng lặp trong kế cuối
    - Các vòng lặp bên trong chọn giá trị lặp k lần
    - Các vòng lặp bên ngoài chọn giá trị lặp nhỏ nhất
    - Lặp lại cho đến khi kiểm đến vòng lặp ngoài cùng

# Kiểm thử vòng lặp

- Kiểm thử vòng lặp móc nối (concatenated loops)
  - Nếu các vòng lặp độc lập với nhau => kiểm như vòng lặp đơn
  - Nếu các vòng lặp phụ thuộc lẫn nhau => kiểm như vòng lặp lồng nhau

# Kiểm thử vòng lặp

- Lặp không cấu trúc (unstructured loops)
  - Không test, re-design