# 白盒测试

- 逻辑覆盖与路径测试
- 基本路径测试（参考 书第9章）
- 主路径测试
- 数据流测试（书第10章）

# 基于控制流图的路径测试

- 将测试路径压缩在一定的限度内（基本路径），以保证每一条基本路径最少被执行一次。
  - 其中，每个循环体最多只执行一次。

- 在结构化测试中，将程序模型化为一个控制流图Control-flow Graph（简称 流图Flowgraph）
  - 从入口到出口的路径代表了控制的流向。

# 基本路径（独立路径）测试

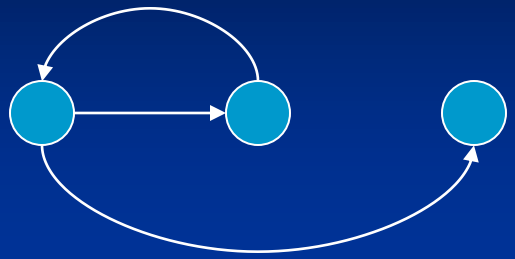
- 由Tom McCabe 提出，将每个测试输入映射到流图中的一个独立的（unique）路径。

- 路径假设： 执行路径一次就可以找出与这条路径相关的缺陷
  - 假设执行同一条路径的所有输入 要么都能找出路径上的缺陷，要么都不能找出。

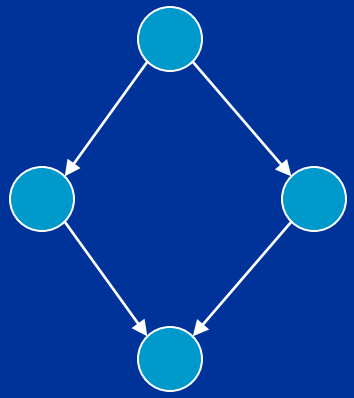
- 方法：在流图（Flowgraph）的基础上，分析环路复杂性，导出基本可执行路径，进而设计测试用例。

# 控制流图的符号

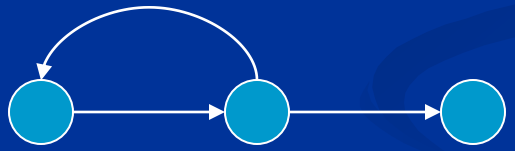
顺序语句



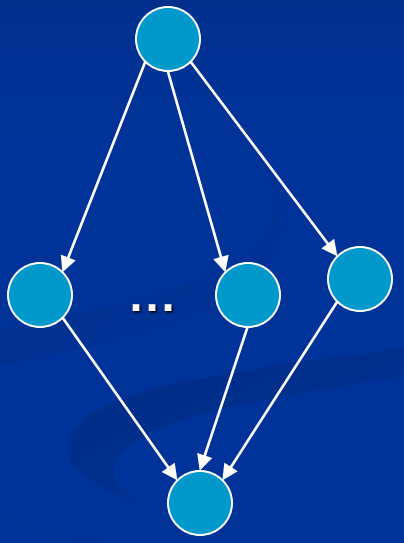
While 语句



If 语句



Until 语句




Case 语句

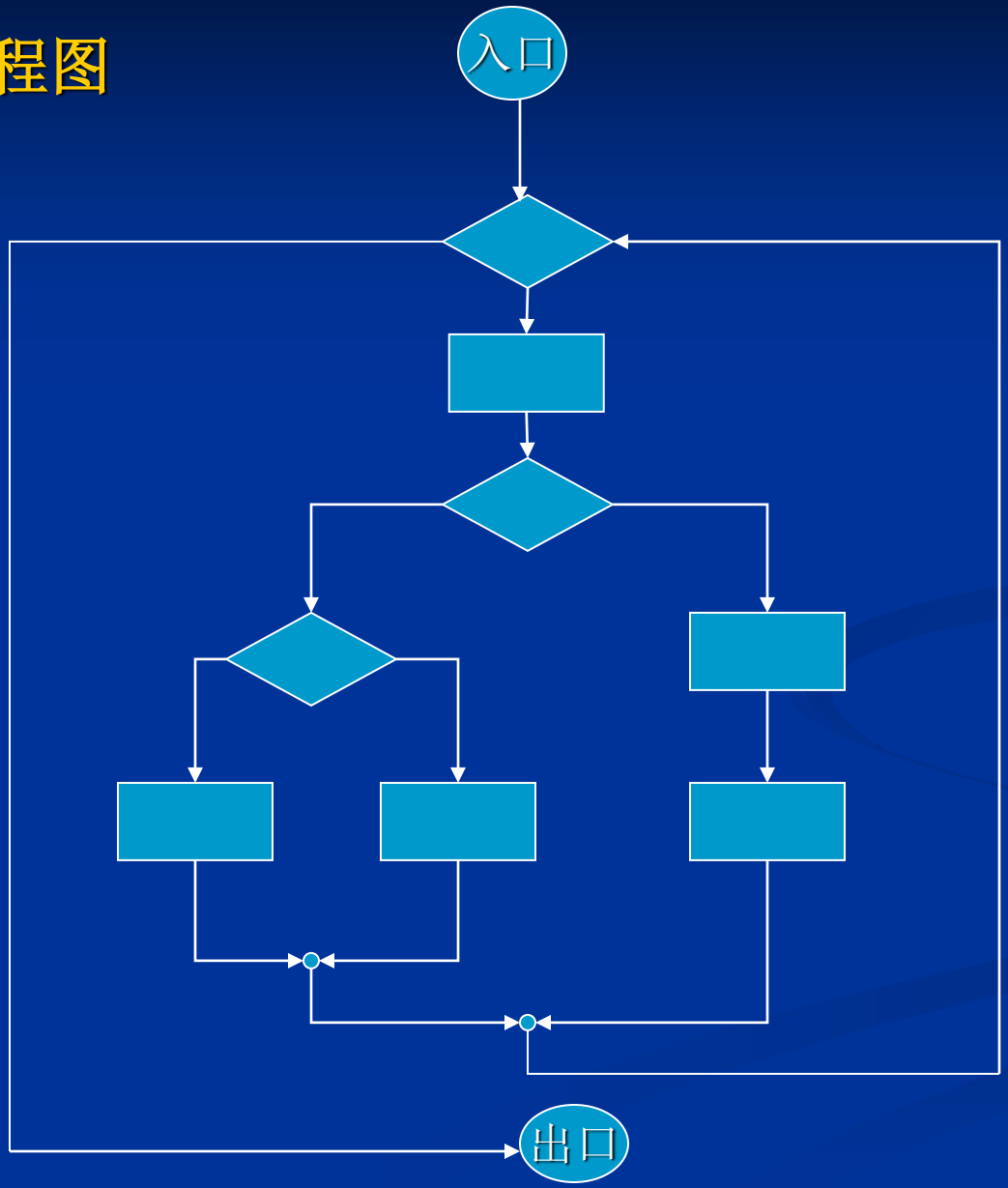
- ○为结点: 表示一个或多个 无分支的源程序语句。
- 箭头为边: 表示控制流的方向。

# 控制流图

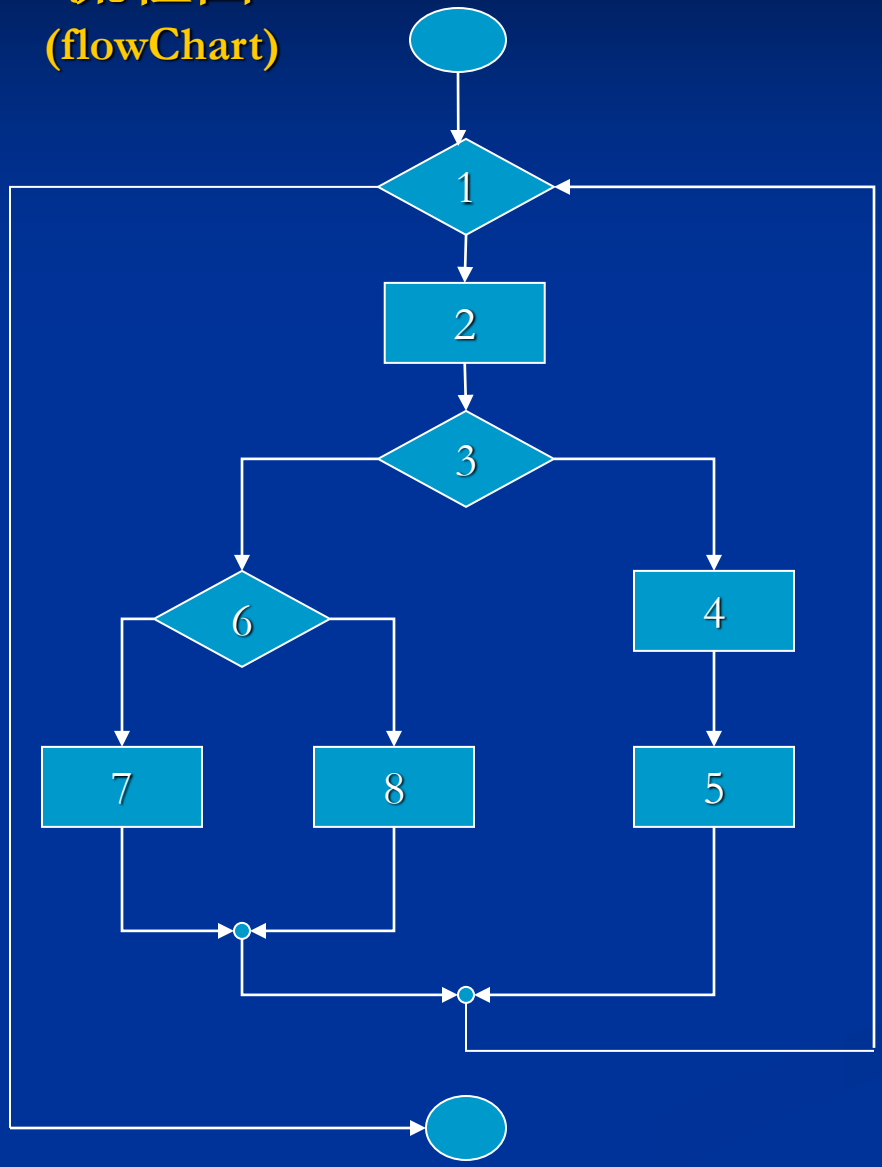
- 在选择或多分支结构中，分支的汇聚处应有一个汇聚结点。
- 边和结点圈定的区域叫做区域，当对区域计数时，图形外的区域也应记为一个区域。
- 如果判断中的条件表达式是由一个或多个逻辑运算符 (OR, AND, ...) 连接的复合条件表达式，则需改为一系列只有单个条件的嵌套的判断。

# 从流程图导出控制流图

流程图



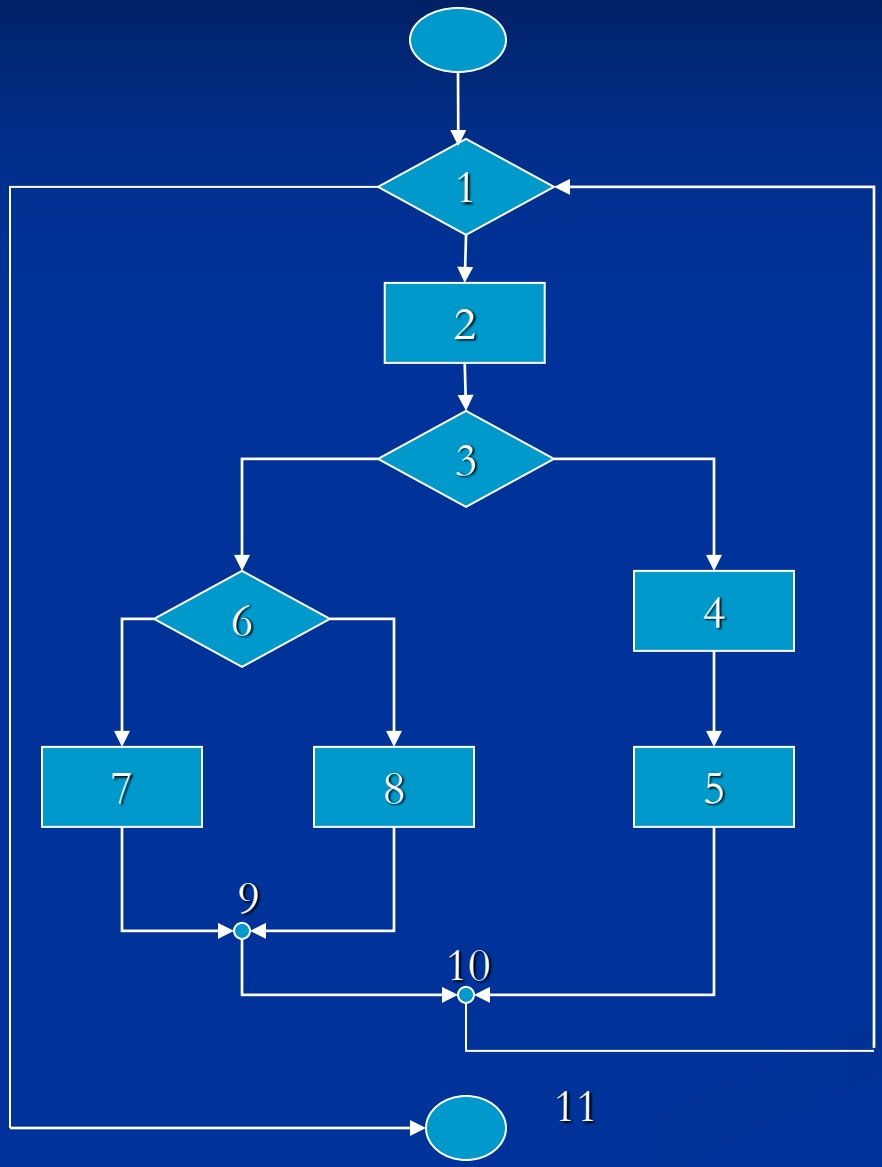
流程图
(flowChart)



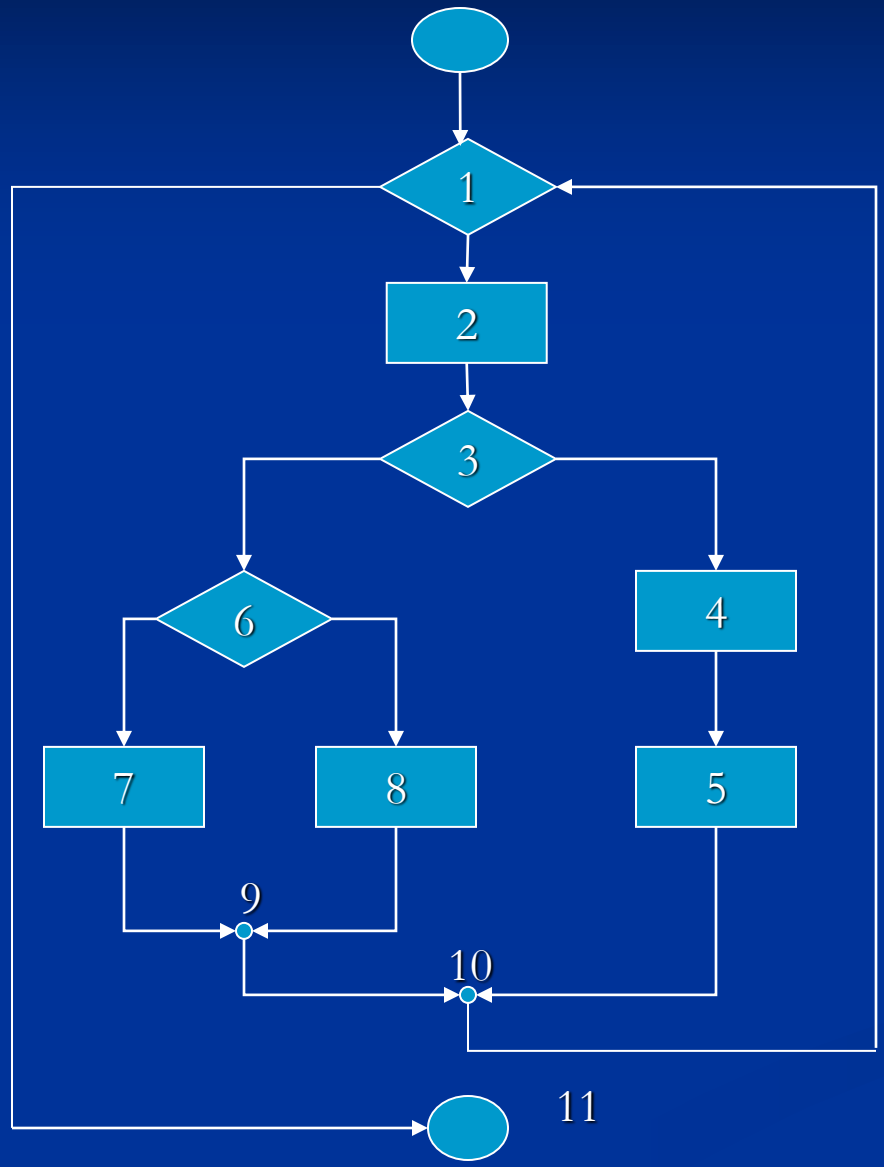

控制流图
(flowGraph)

# 流程图

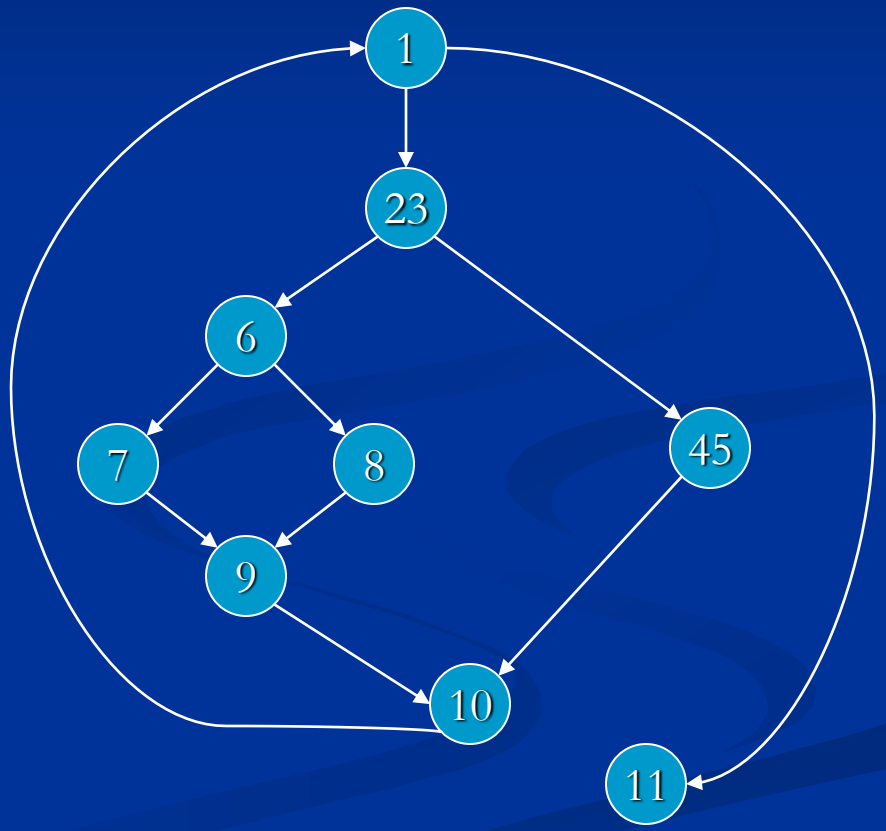
# 控制流图



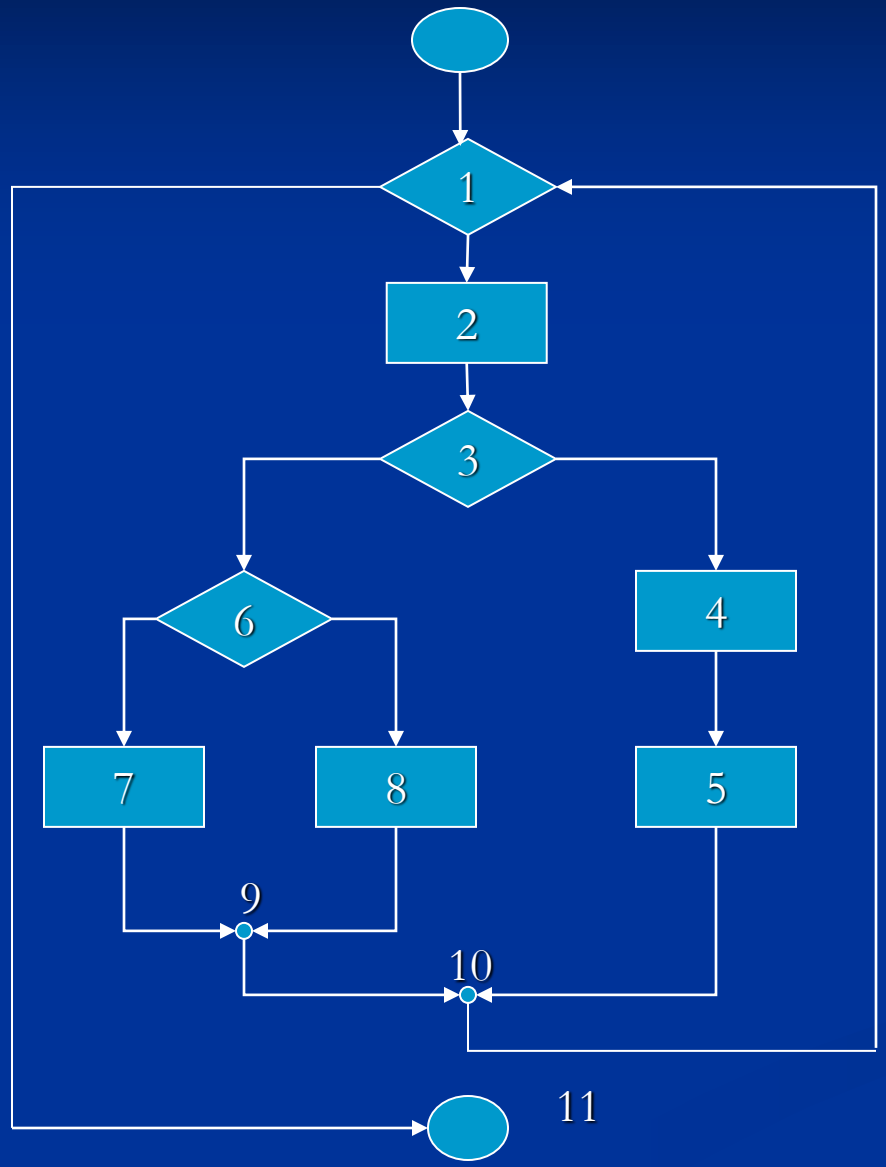
# 流程图



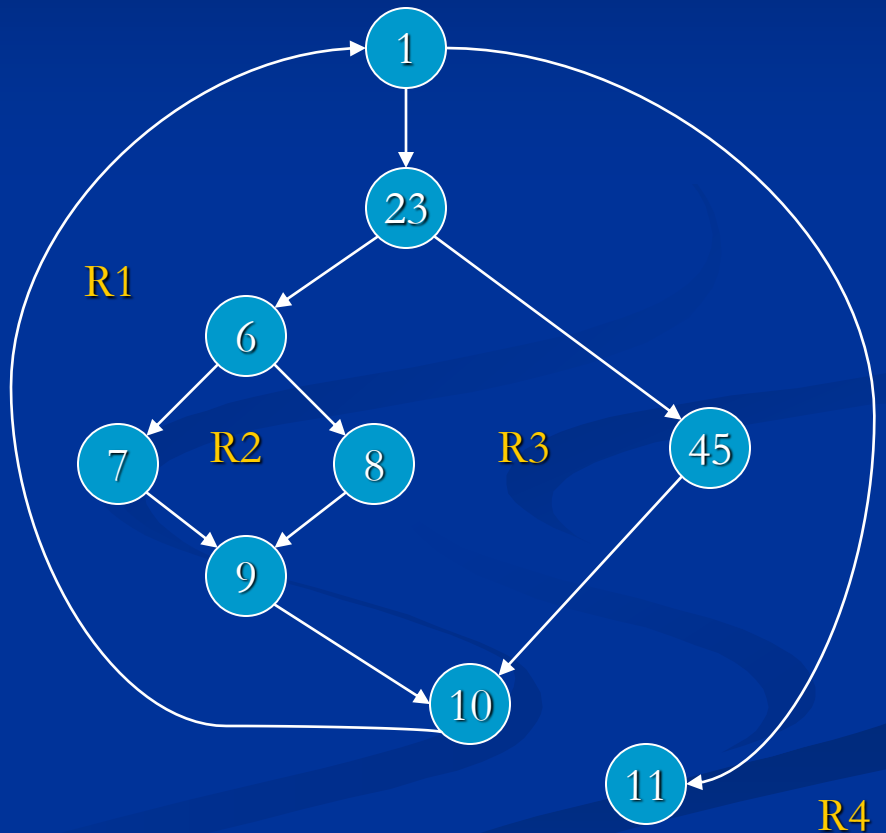

# 控制流图



# 流程图




# 控制流图



# 从程序导出控制流图

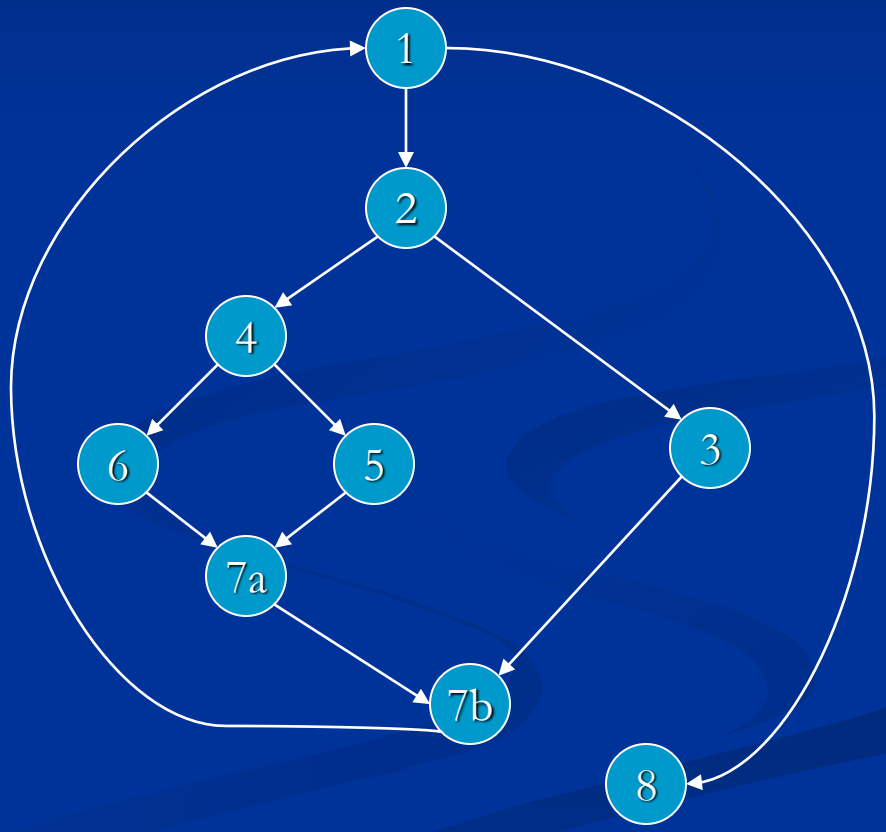
```
Procedure :sort
 while records_remain
       Read record;
   if record_field 1=0
   then process record;
            Store in buffer;
            Increment counter;
   else if record_field 2=0
          then reset counter;
          else process record;
                    Store in field;
        endif
  endif
     endwhile
end
```

# 从程序导出控制流图

```
Procedure :sort
1: while records_remain
       Read record;
2:   if record_field 1=0
3:   then process record;
           Store in buffer;
           Increment counter;
4:   else if record_field 2=0
5:          then reset counter;
6:          else process record;
                  Store in field;
7a:        endif
7b:  endif
     endwhile
8:end
```

# 从程序导出控制流图

```
Procedure :sort
1: while records_remain
       Read record;
2:   if record_field 1=0
3:   then process record;
           Store in buffer;
           Increment counter;
4:   else if record_field 2=0
5:          then reset counter;
6:          else process record;
                   Store in field;
7a:        endif
7b:  endif
     endwhile
8:end
```



# 遇到复合条件OR时

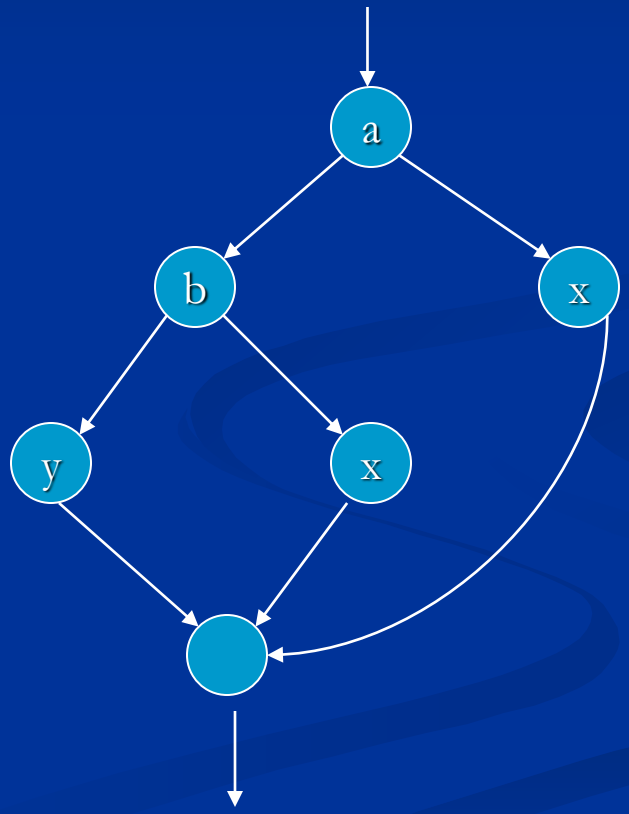
- 需将复合条件表达式改为一系列只有单个条件的嵌套的判断。
  - 为什么？

```
        .
        .
        .
If a OR b
   Then procedure x
   Else procedure y
EndIf
```

# 遇到复合条件OR时

- 将复合条件表达式改为一系列只有单个条件的嵌套的判断。

```
        .
        .
        .
If a OR b
   Then procedure x
   Else procedure y
EndIf
        .
        .
        .
```

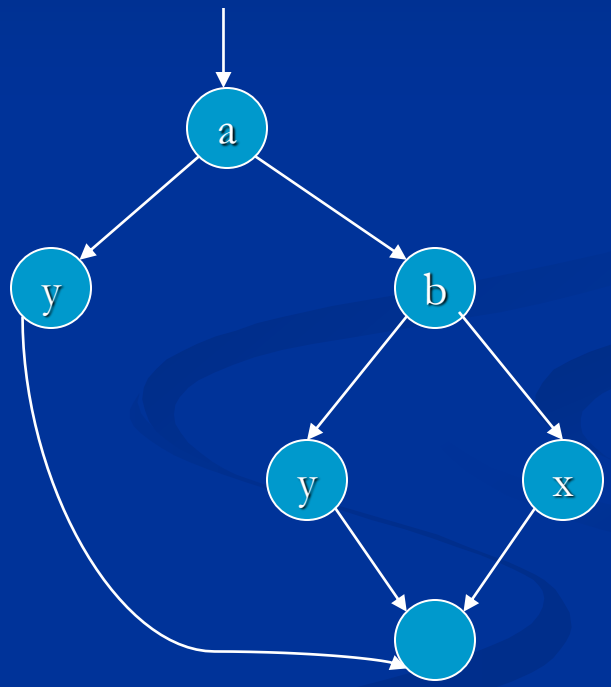


# 遇到复合条件AND时

```
        .
        .
        .
If a AND b
    Then procedure x
    Else procedure y
EndIf
```

# 遇到复合条件AND时

```
        .
        .
        .
If a AND b
    Then procedure x
    Else procedure y
EndIf
```



# 独立路径

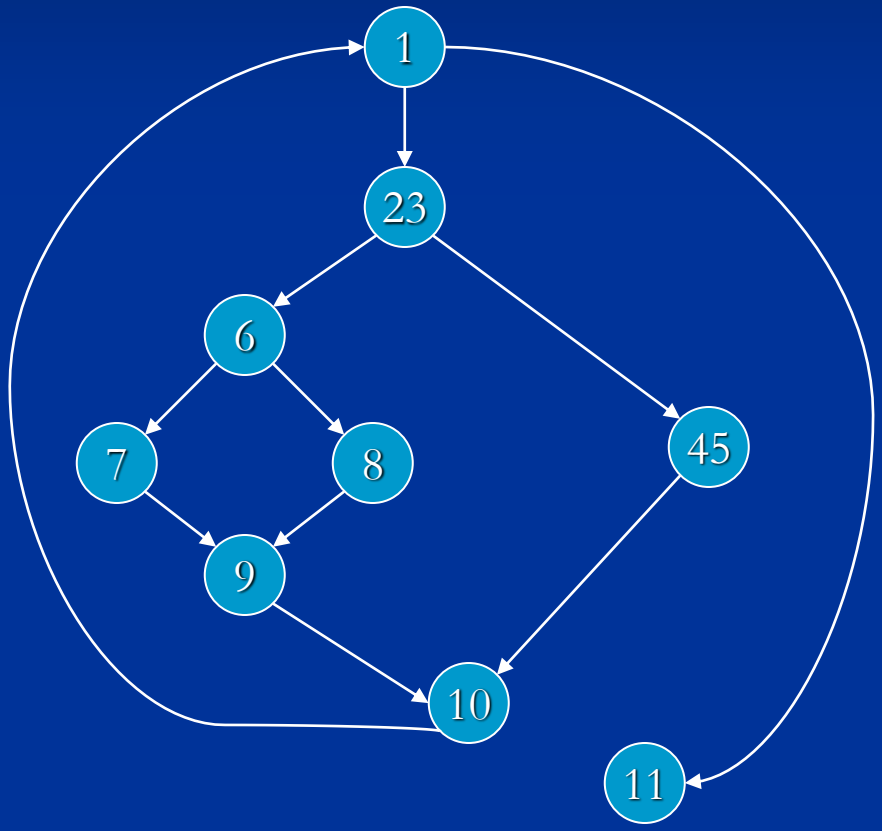
- 不可能穷尽路径测试，只选择基本路径（独立路径）集合。
- McCabe定义的独立路径：是指从被测程序的入口到出口的最短路径开始，包括一组以前没有处理过的语句或条件的一条路径。
  - 从控制流图来看，一条独立路径是至少包含有一条在此前的其它独立路径中从未有过的边的路径。
  - 即：从入口进入控制流图后，至少要经历一条从未走过的边。
- 程序越复杂，路径数越多，基本路径数越多。
  - 程序的复杂度 如何度量？

# 独立路径数

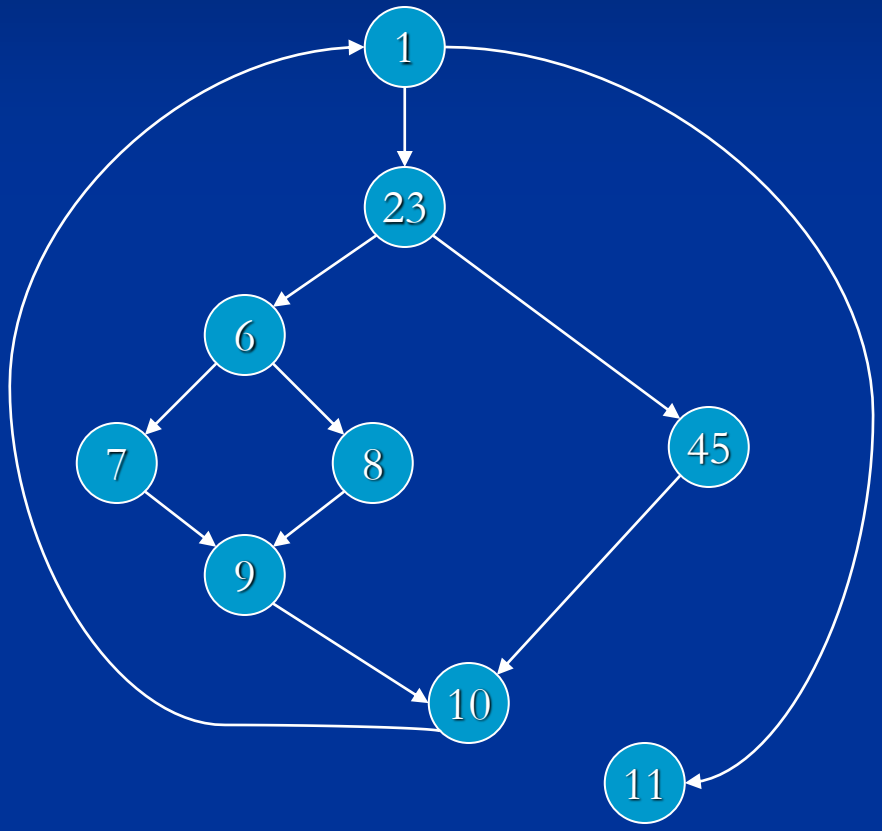
- McCabe定义了程序的流图的环路复杂度
  - V(G)= 边数E -节点数N + 2
- （注：不考虑流图的方向，得到一个无向连通图）可推导出：
  - V(G) = 平面分割的区域的个数
  - = 判断节点的个数P + 1

- McCabe方法：独立路径数＝环路复杂性
  - 这是确保每个语句和判断至少执行一次所需的最少的测试用例数。
  - 只要设计出的测试用例能够保证这些基本路径被执行，就可以使得程序中的每一个可执行语句至少被执行一次，每个条件的取真取假分支也能得到测试。

# 独立路径



# 独立路径




基本路径集包括如下四条独立路径

Path1: 1—11

Path2: 1—23—45—10—1—11

Path3: 1—23—6—7—9—10—1—11

Path4: 1—23—6—8—9—10—1—11

而：

1—23—45—10—1—23—

— 6—8—9—10—1—11

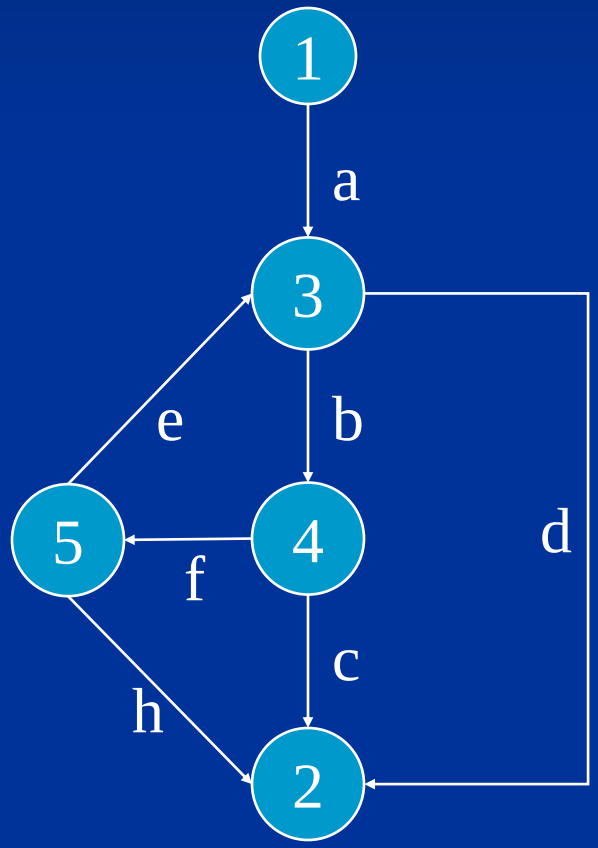
就不是独立路径。

# 图矩阵

- 自动化软件工具：
  - 从程序导出流图
  - 从流图生成基本测试路径
- 开发软件工具需要用矩阵来存储和计算流图.

- 图矩阵：列数和行数均等于流图的结点数，每列和每行均对应图的结点，元素对应结点到结点的边。

# 流图与对应的矩阵




| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | | | a | | |
| 2 | | | | | |
| 3 | | d | | b | |
| 4 | | c | | | f |
| 5 | | h | e | | |

# 例：利用图矩阵求环路复杂性

**最简单的情况：连接权值是1（存在连接）或0（不存在连接）**

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | | | 1 | | |
| 2 | | | | | |
| 3 | | 1 | | 1 | |
| 4 | | 1 | | | 1 |
| 5 | | 1 | 1 | | |

判断结点个数＝出度－入度

1-1 = 0

2-1 = 1

2-1 = 1

2-1 = 1

V(G)=判断结点个数＋1

环路复杂性 ⟶ 3 + 1 = 4

# 流图与对应矩阵的关系

- 图矩阵只是流图的表格表示。
- 若对每个矩阵项加入连接权值，图矩阵就可以用于在测试中评估程序的控制结构。

- 连接权值为控制流提供了另外的信息；
  - 最简单的情况：连接权值是1（存在连接）或0（不存在连接）
  - 为权值赋予更多的属性
    - 执行连接（边）的概率；
    - 穿越连接的处理时间；
    - 穿越连接时所需的内存；
    - 穿越连接时所需的资源；
    - ……

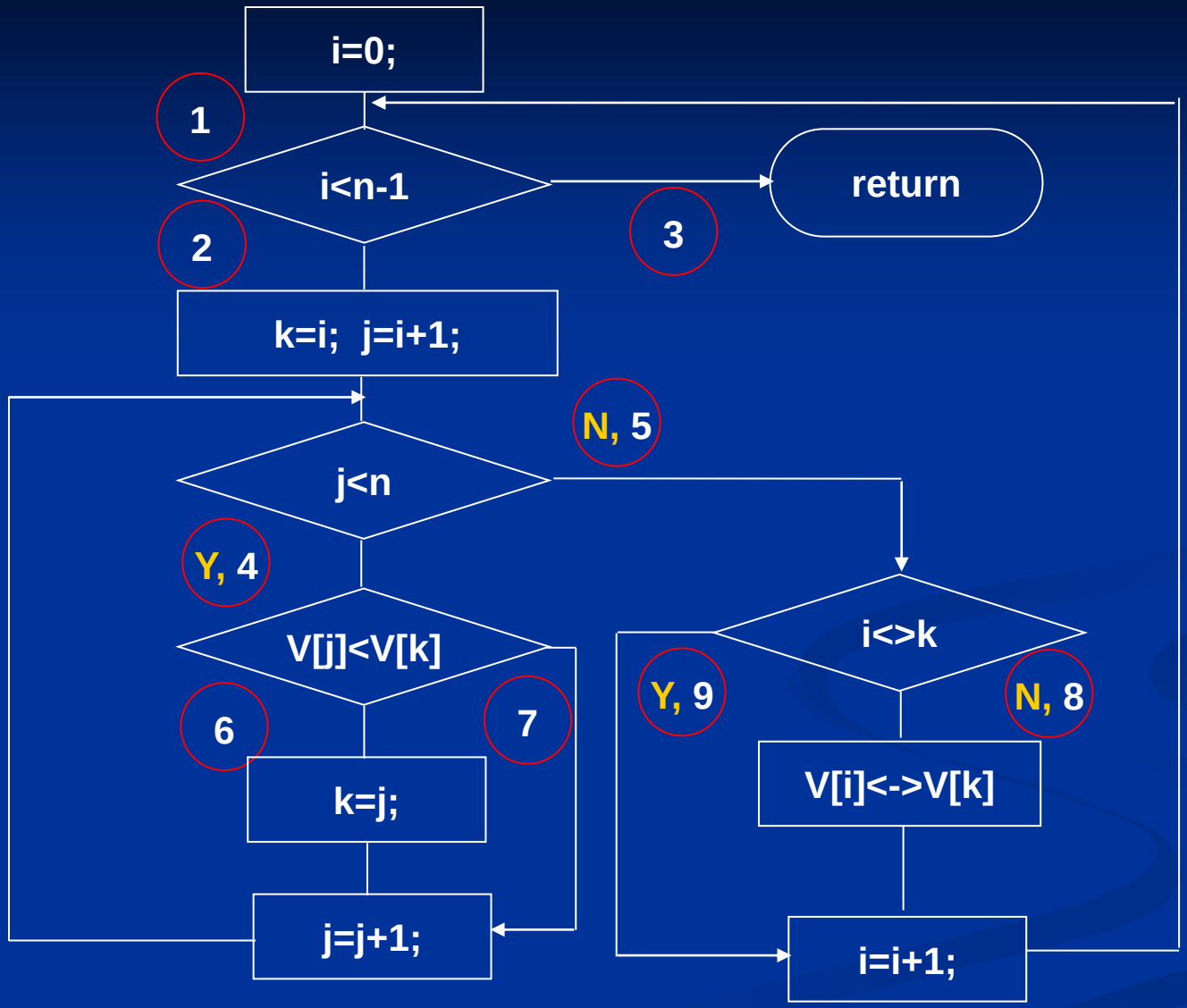
# 基本路径测试的步骤

- 以详细设计或源代码为基础,导出流图;
- 计算出流图G的环路复杂性V(G);
- 确定线性无关的基本路径集;
- 生成测试用例, 确保基本路径集中每条路径的执行。
  - 根据判断结点给出的条件，选择适当的输入数据以保证某一条路径可以被测试到 。
  - 每个输入被执行之后，与预期结果进行比较。
  - 必须注意，一些独立的路径(如例中的Path1)，往往不是完全孤立的，有时它是程序正常的控制流的一部分，这时，这些路径的测试可以是另一条路径测试的一部分。

# 案例：用基本路径测试法 生成测试用例

被测程序模块：

```
void SelectSort( datalist & list ) {
    for (int i=0; i<list.n-1; i++) {
        int k=i;
        for (int j=i+1; j<list.n; j++)
            if (list.V[j].getKey() < list.V[k].getKey()) k=j;
        if (k!=i) Swap(list.V[i], list.V[k]);
    }
}
```




区域数=5

- 基本路径集

  path1: 1-3

  path2: 1-2-5-8 ……

  path3: 1-2-5-9 ……

  path4: 1-2-4-6……

  path5: 1-2-4-7 ……

- ……表示后面剩下的路径是可以选择的

  （注：基本路径集并不是唯一的）

- 测试用例
  - 路径 1-3
    - 取n=1
  - 路径 1-2-5-8 ……
    - 取n=2

      预期结果：路径5-8-3不可到达
  - 路径 1-2-5-9 ……
    - 取n=2

      预期结果：路径5-9-3不可到达

- 路径 1-2-4-6 -5-9-3

  取n=2， v[0]=2, v[1]=1

  预期结果：k=1, 路径9-3不可到达

- 路径 1-2-4-6 -5-8-3

  取n =2， v[0]=2, v[1]=1

  预期结果：k=1, v[0]=1, v[1]=2

- 路径 1-2-4-7-5-8-3

  取n=2， v[0]=2, v[1]=1

  预期结果：k=0, 路径8-3不可到达

- 路径 1-2-4-7-5-9-3

  取n=2， v[0]=2, v[1]=1

  预期结果：k=0, v[0]=1, v[1]=2

# 控制流图的定义

by Frances E. Allen， 1970
"Control Flow Analysis"

A basic block is a linear sequence of program instructions having one entry point (the first instruction executed) and one exit point (the last instruction executed). It may of course have many predes-

Frances E. Allen

sors and many successors and may even be its own successor. Program entry blocks might not have predecessors that are in the program; program terminating blocks never have successors in the program.

A control flow graph is a directed graph in which the nodes repre-sent basic blocks and the edges represent control flow paths. Every-thing that is said about directed graphs in this paper holds for control flow graphs.

# 白盒测试

- 逻辑覆盖与路径测试
- 基本路径测试（参考 书第9章）
- 主路径测试
- 数据流测试（书第10章）

# 主路径测试

- 视频