软件工程导论

主 讲 人：刘文洁
工作单位：西北工业大学计算机学院二系
电　　话：13572173962
Email　　：liuwenjie@nwpu.edu.cn

- 详细设计的概念
- 结构程序设计
- 详细设计的工具
- 程序复杂程度的定量度量

本章主要内容

以总体设计阶段的工作为基础；

- 结构程序设计；
- 详细设计的工具。

概念

详细设计以总体设计阶段的工作为基础的，但又不同于总体设计，主要表现为以下两个方面：

（1）在总体设计阶段，**数据项和数据结构**以比较抽象的方式描述，而详细设计阶段则应在此基础上给出**足够详细描述**。

（2）详细设计要提供关于**算法的更多的细节。**

例如：总体设计可以声明一个模块的作用是对一个表进行排序，详细设计则要确定使用哪种排序算法。在详细设计阶段为每个模块增加了足够的细节后，程序员才能够以相当直接的方式进行下一阶段的编码工作。

任务和原则

一、详细设计的任务

（1）确定每个模块的算法。

（2）确定每一个模块的数据组织。

（3）为每个模块设计一组测试用例。

（4）编写详细设计说明书。

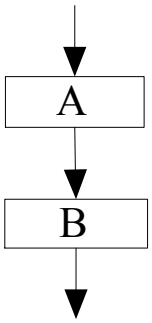
二、详细设计的原则

（1）模块的逻辑描述正确可靠、清晰易读。

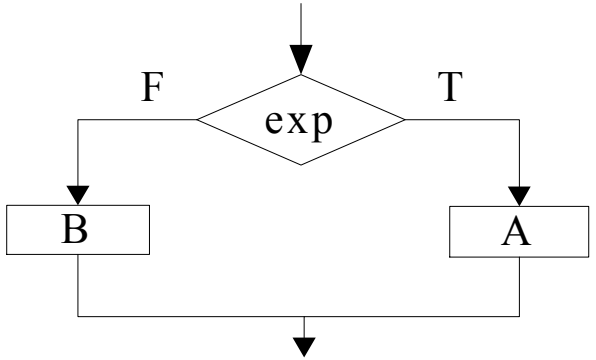
（2）采用结构化程序设计方法，改善控制结构，降低程序复杂度，提高程序的可读性、可测试性和可维护性。

- 详细设计的概念
- **结构程序设计**
- 详细设计的工具
- 程序复杂程度的定量度量

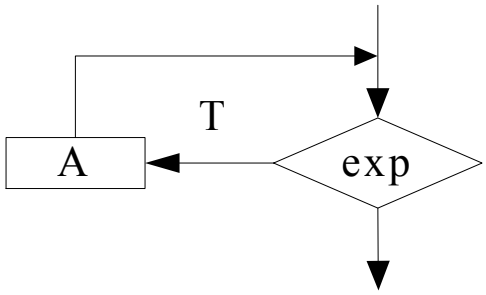
结构程序设计是一种设计程序的技术，它采用自顶向下逐步求精的设计方法和单入口单出口的控制结构。



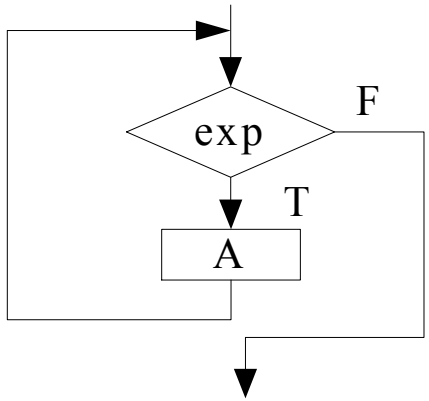
(a)



(b)



或



(c)

使用**结构程序设计技术**的好处：

（1）自顶向下逐步求精的方法符合人类解决复杂问题的普遍规律，可以显著**提高**软件开发的**成功率**和**生产率**。

（2）先全局后局部、先整体后细节、先抽象后具体的逐步求精过程开发出的程序有**清晰的层次结构**。

（3）使用**单入口单出口**的控制结构而不使用GOTO语句，使得程序的静态结构和它的动态执行情况比较一致。

（4）控制结构有**确定的逻辑模式**，编写程序代码只限于使用很少几种直截了当的方式。

（5）程序清晰和模块化使得在修改和重新设计一个软件时**可以重用**的代码量最大。

（6）程序的**逻辑结构清晰**，有利于程序正确性证明。

- 详细设计的概念
- 结构程序设计
- **详细设计的工具**
- 程序复杂程度的定量度量

详细设计的工具

一、程序流程图

二、N-S图

三、PAD图

四、PDL语言

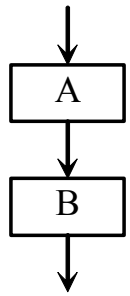
程序流程图

- **程序流程图**又称之为程序框图，它是软件开发者最熟悉的一种算法表达工具。

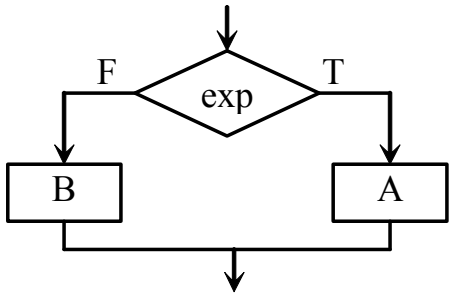
- 它独立于任何一种程序设计语言，比较直观和清晰地描述过程的控制流程，易于学习掌握。因此，至今仍是软件开发者最普遍采用的一种工具。

程序流程图

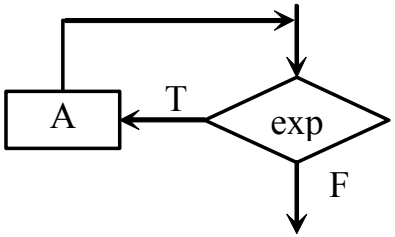
流程图的五种基本控制结构:



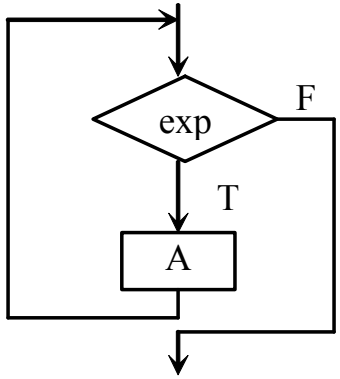
(a) 顺序结构



(b) 选择结构



或



(c) 循环结构

1、顺序型

顺序型由几个连续的处理步骤依次排列构成。

2、选择型

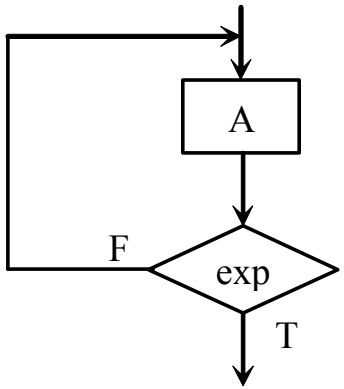
选择型是指由某个逻辑判断式的取值决定选择两个处理中的一个。

3、while型循环

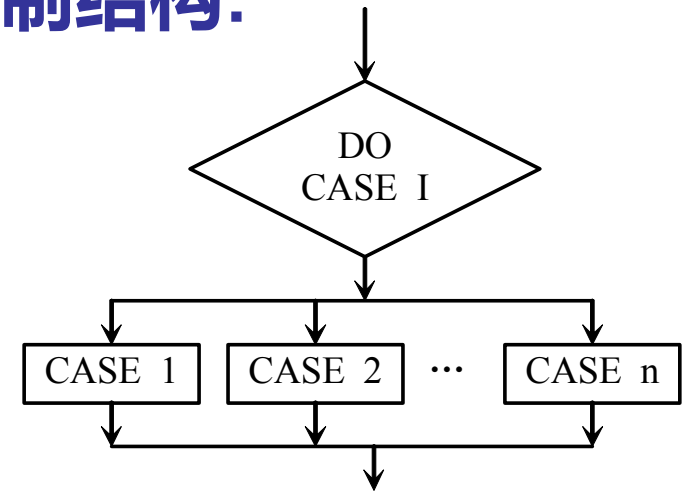
while型循环是先判定型循环，在循环控制条件成立时，重复执行特定的处理。

程序流程图

流程图的五种基本控制结构:



(a) DO_UNTIL 型循环结构



(b) DO_CASE 型多分支结构

4、until型循环

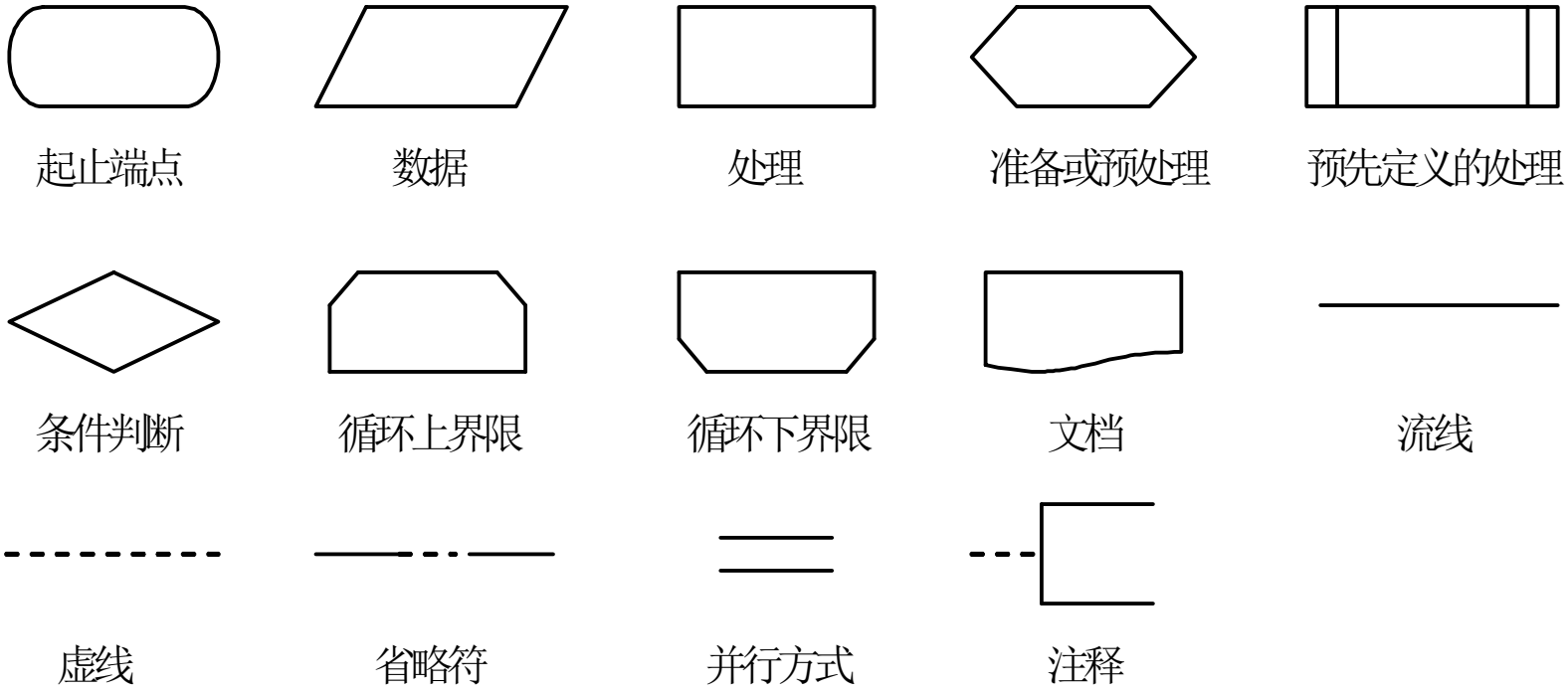
until型循环是后判定型循环，重复执行某些特定的处理，直到控制条件成立为止。

5、多情况型选择

多情况型选择列举多种处理情况，根据控制变量的取值，选择执行其一。

程序流程图

程序流程图中常用的符号:



程序流程图

流程图也存在一些严重的不足：

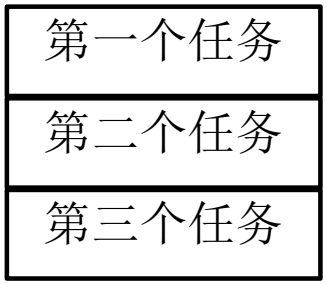
程序流程图虽然比较直观，灵活，并且比较容易掌握，但是它的随意性和灵活性却使它不可避免地存在着一些缺点：

（1）由于程序流程图的特点，它本身并不是逐步求精的好工具。因为它使程序员容易过早地考虑程序的具体控制流程，而忽略了程序的全局结构；

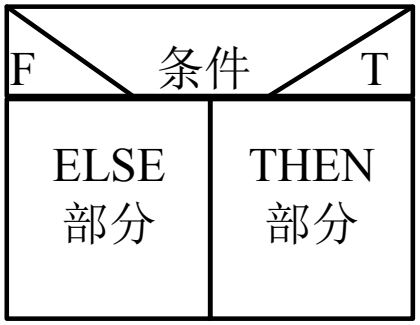
（2）程序流程图中用箭头代表控制流，这样使得程序员不受任何约束，可以完全不顾结构程序设计的精神，随意转移控制；

（3）程序流程图在表示数据结构方面存在不足。

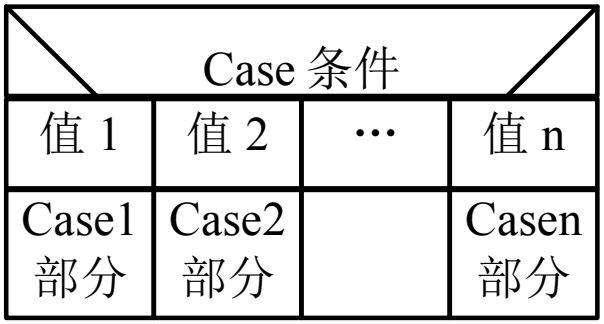
N-S图



(a) 顺序结构

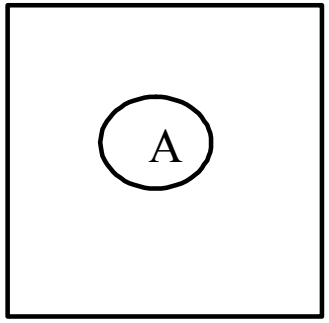


(b) 选择结构



(c) 多分支结构

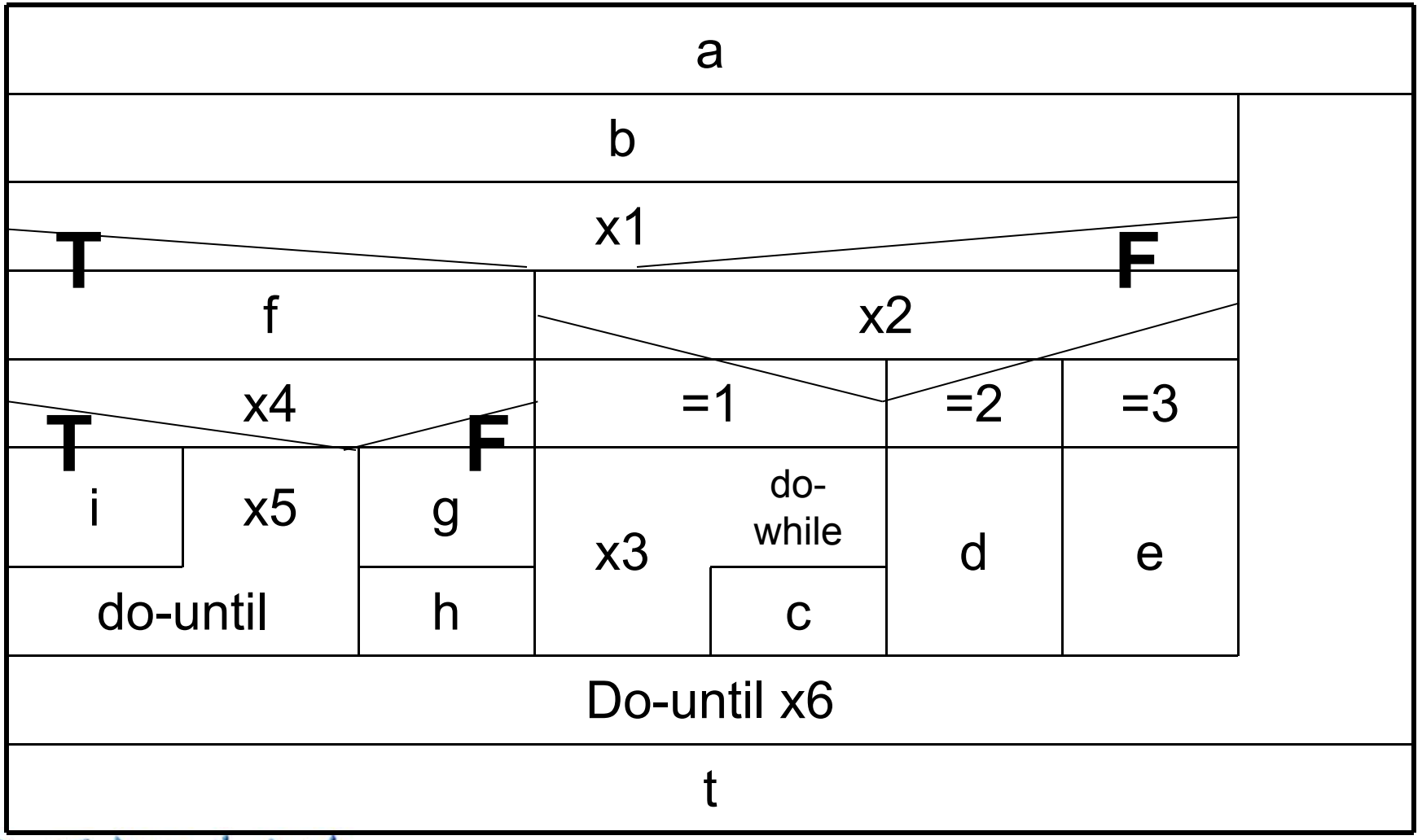
(d) 循环结构



(e) 调用子程序 A

N-S图

N-S图应用举例


a
b
x1
T
F
f
x2
x4
=1
=2
=3
T
F
i
x5
g
x3
do-
while
d
e
do-until
h
c
Do-until x6
t

N-S图

N-S图有以下一些特点：

（1）**功能域**（即某一个特定控制结构的作用域）有明确的规定，并且可以很直观地从N-S图上看出来；

（2）它的**控制转移不能任意规定**，必须遵守结构化程序设计的要求；

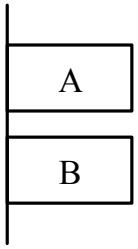
（3）很容易确定局部数据和全局数据的**作用域**；

（4）很容易表现嵌套关系，也可以表示模块的**层次结构**。

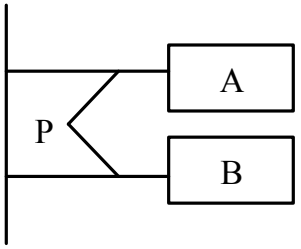
PAD图

PAD是用结构化程序设计思想表现程序逻辑结构的图形工具。

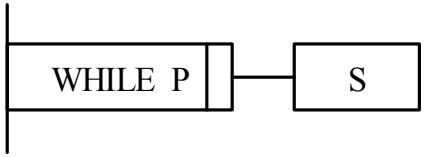
PAD也设置了五种基本控制结构的图示，并允许递归使用。



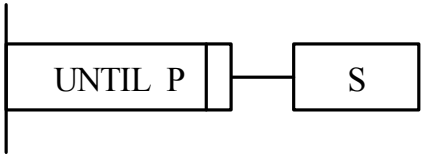
(a) 顺序结构



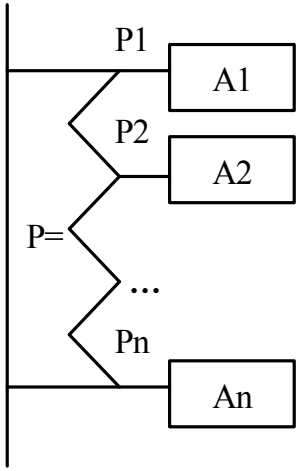
(b) 选择结构



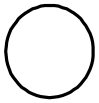
(c) WHILE 型循环结构



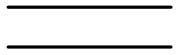
(d) UNTIL 型循环结构



(e) 多分支结构



(f) 语句标号



(g) 定义

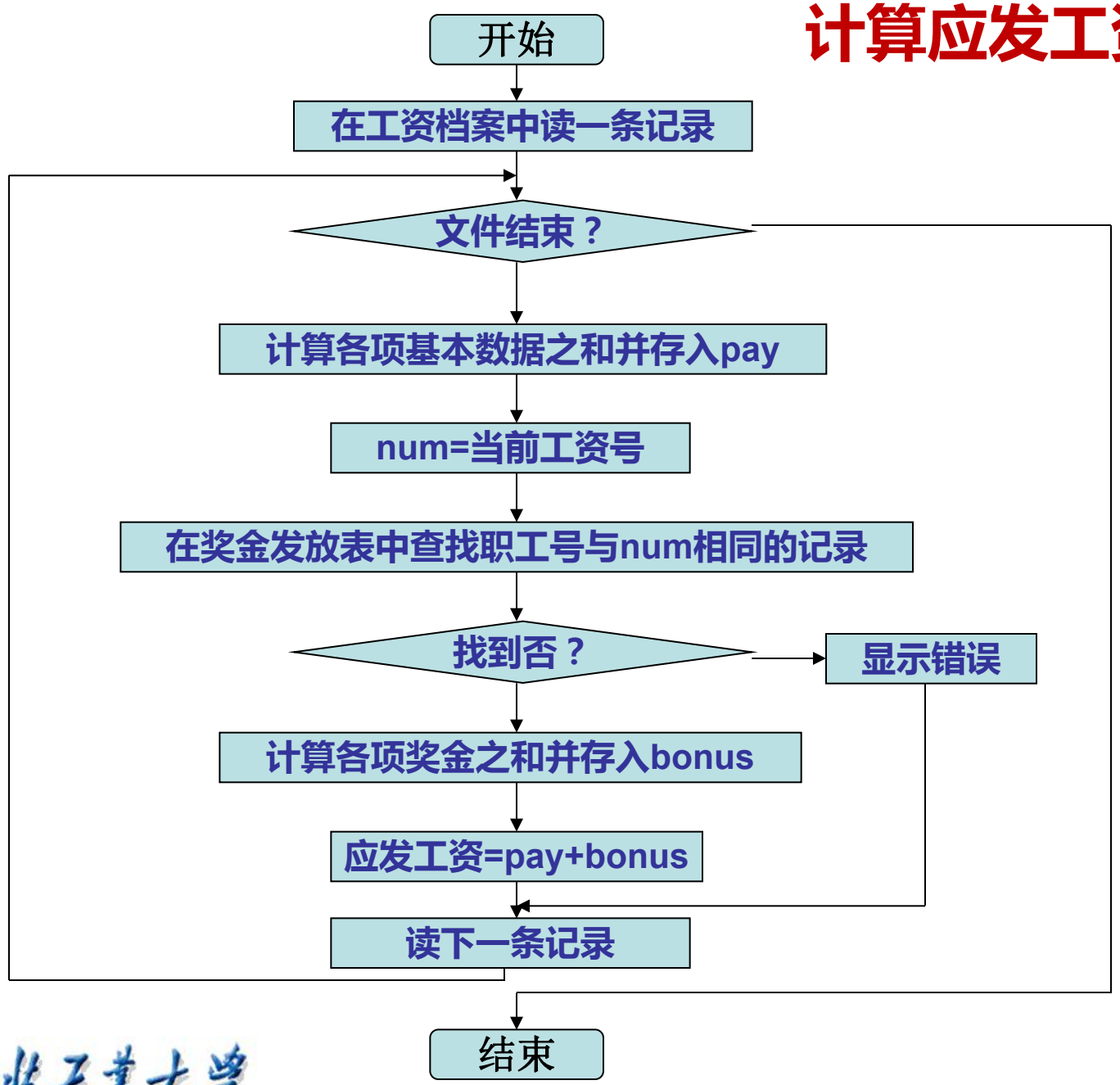
PAD图的特点

1. 清晰度和结构化程度高。
2. PAD图中的最左面的线是程序的主干线,即程序的第一层结构。随着程序层次的增加，PAD图逐渐向右延伸。因此，PAD图可读性强。
3. 利用PAD图设计出的程序必定是结构化的程序。
4. 容易将PAD图转化成高级语言源程序。
5. PAD图支持自顶向下逐步求精的方法。

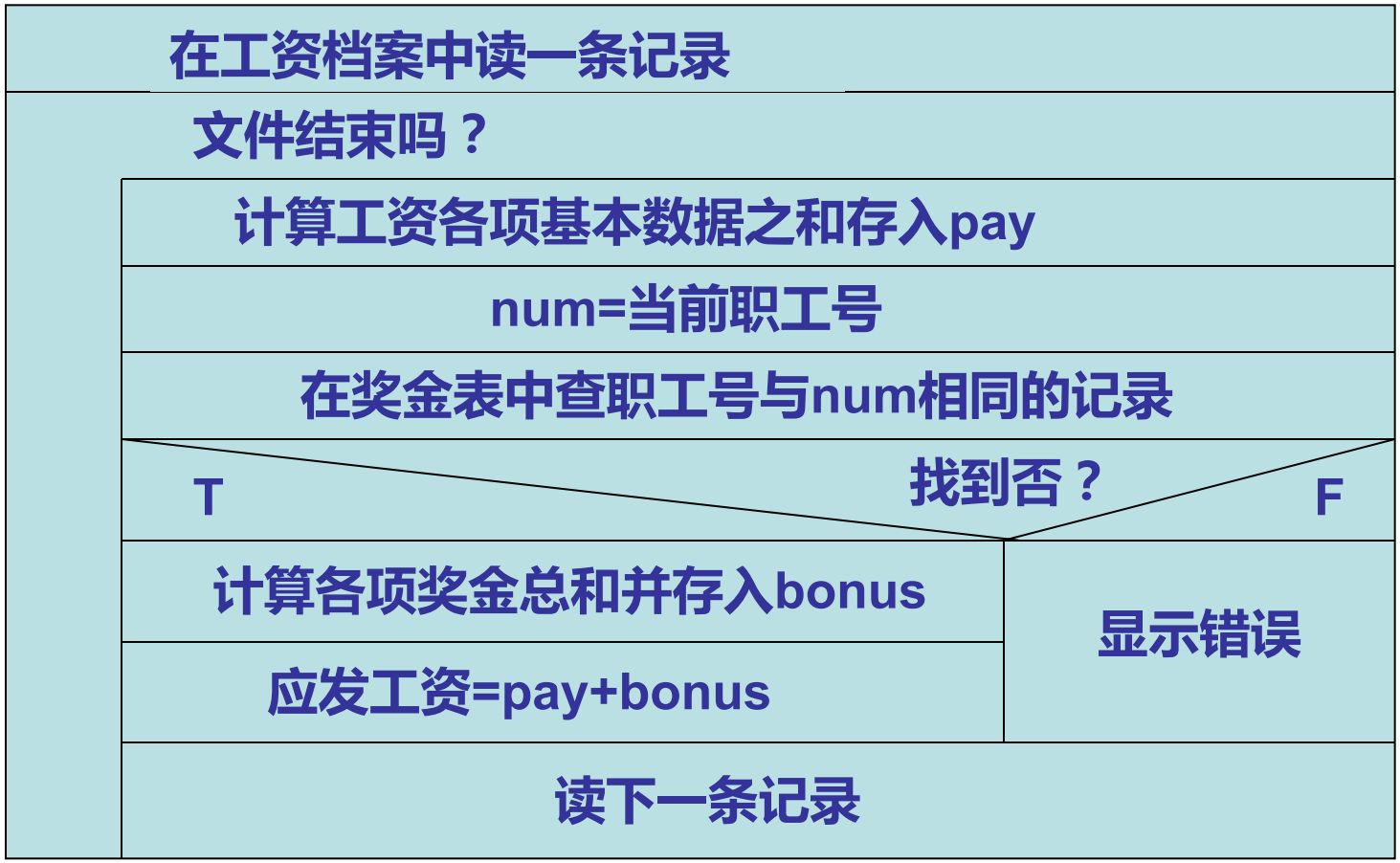
下面以计算应发工资模块为例，

用上述三种图形工具：程序流程图、盒图以及PAD图，分别来设计。

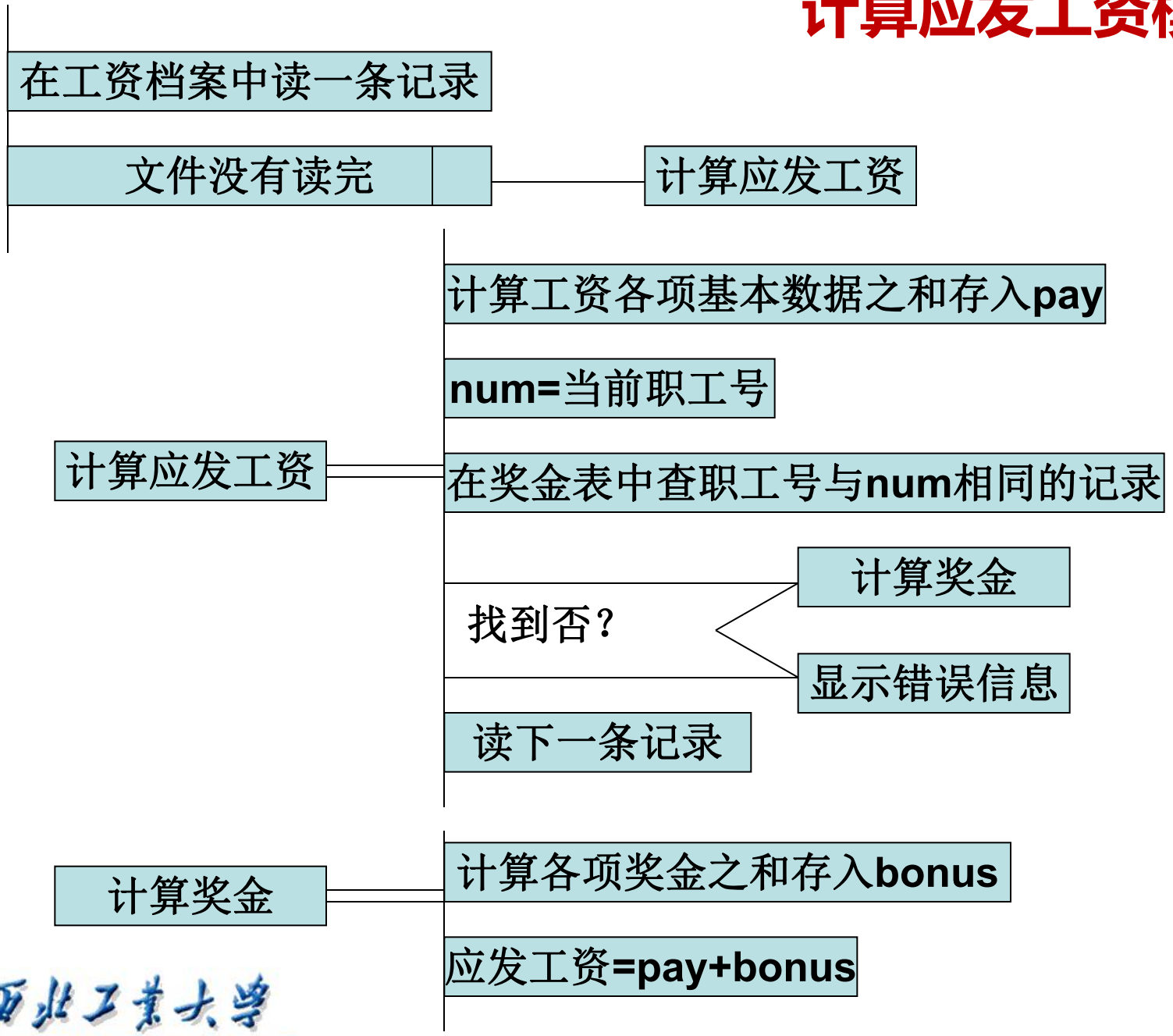
计算应发工资模块



计算应发工资模块


在工资档案中读一条记录
文件结束吗？
计算工资各项基本数据之和存入pay
num=当前职工号
在奖金表中查职工号与num相同的记录
找到否？
T
F
计算各项奖金总和并存入bonus
应发工资=pay+bonus
显示错误
读下一条记录

计算应发工资模块



PDL(过程设计语言)

PDL是所有非正文形式的过程设计工具的统称，也称为伪码。到目前为止已出现多种PDL语言。PDL具有“非纯粹”的编程语言的特点。

PDL(过程设计语言)的特点

- 关键字采用固定语法并支持结构化构件、数据说明机制和模块化；
- 处理部分采用自然语言描述；
- 可以说明简单和复杂的数据结构；
- 子程序的定义与调用规则不受具体接口方式的影响。

PDL(过程设计语言)的特点

- PDL具有严格的关键字外部语法，用于定义控制结构和数据结构；
- 另一方面，PDL表示实际操作和条件的内部语法通常又是灵活自由的，以便可以适应各种工程项目的需要；
- 是一种“混杂”语言，它使用一种语言（某种自然语言）的词汇，同时，却使用另一种语言（某种结构化的程序设计语言）的语法；
- PDL语言的缺点是不如图形工具形象直观

PDL(过程设计语言)

PDL具有严格的关键字外部语法，内部语法灵活。

PDL-----关键字+自然语言

(1)数据说明：其功能是定义数据的类型和作用域

格式: TYPE <变量名> AS <限定词1> <限定词2>

TYPE number AS STRING LENGTH (12)

PDL(过程设计语言)

(2)**程序块**: PDL的过程成分是由块结构构成的，而块将作为一个单个的实体来执行。

```
BEGIN <块名>
        < 一组伪代码语句>
END
```

(3)**子程序结构**: 把 PDL 中的过程称为子程序。

```
PROCEDURE <子程序名> <一组属性>
 INTERFACE  <参数表>
        < 程序块或一组伪代码语句>
END
```

PDL(过程设计语言)

(4)基本控制结构:

```
IF <条件>
  THEN  <程序块/伪代码语句组>；
  ELSE   <程序块/伪代码语句组>；
ENDIF
```

--- 选择型结构

PDL(过程设计语言)

```
DO WHILE <条件描述>
    <程序块/伪代码语句组>；
ENDDO
```

```
REPEAT UNTIL <条件描述>
    <程序块/伪代码语句组>；
ENDREP
```

--- 重复型结构

PDL(过程设计语言)

```
CASE OF <case 变量名>；
    WHEN < case 条件1> SELECT <程序块/伪代码语句组>；
    WHEN < case 条件2> SELECT <程序块/伪代码语句组>；
    … …
    DEFAULT: < 缺省或错误case:  <程序块/伪代码语句组>；

ENDCASE
```

----- 多路选择结构

PDL(过程设计语言)

例：数据字典中，使用PDL进行数据处理的说明

```
处理名：核实订票处理
编号：3.2
激活条件：收到取订票信息
处理逻辑：1、读订票旅客信息文件
          2、搜索此文件中是否有与输入信息中
             姓名及身份证号相符的项目
           IF 有
             THEN 判断余项是否与文件中信息相符
               IF 是 THEN 输出已订票信息
                     ELSE 输出未订票信息
               ENDIF
             ELSE  输出未订票信息
           ENDIF
执行频率：实时
```

- 详细设计的概念
- 结构程序设计
- 详细设计的工具
- **程序复杂程度的定量度量**

程序复杂程度的度量

1、软件复杂性

是软件度量的一个重要分支。主要参数有：

- 规模：即总共的指令数，或源程序行数。
- 难度：通常由程序中出现的操作数的数目所决定的量来表示。
- 结构：通常用与程序结构有关的度量来表示。
- 智能度：即算法的难易程度。

McCabe方法（环形复杂度）

根据程序控制流的复杂程度，定量度量程序的复杂度

- 流图
- 计算环形复杂度的方法
- 环形复杂度的用途

流图

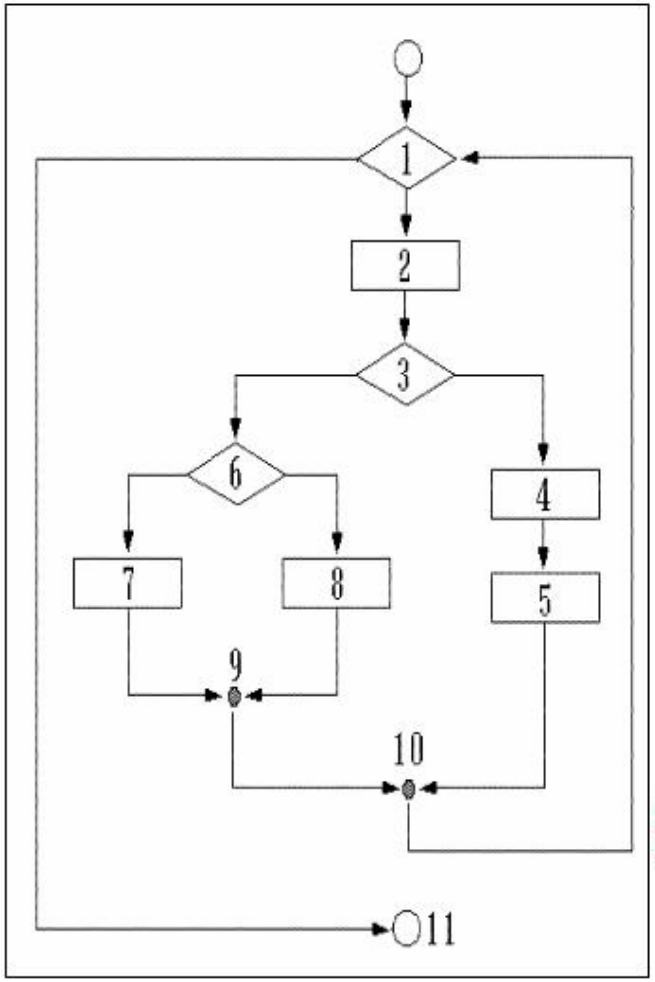
所谓流图实质上是“退化了的”程序流程图，它仅仅描绘程序的控制流程，完全不表现对数据的具体操作以及分支或循环的具体条件。

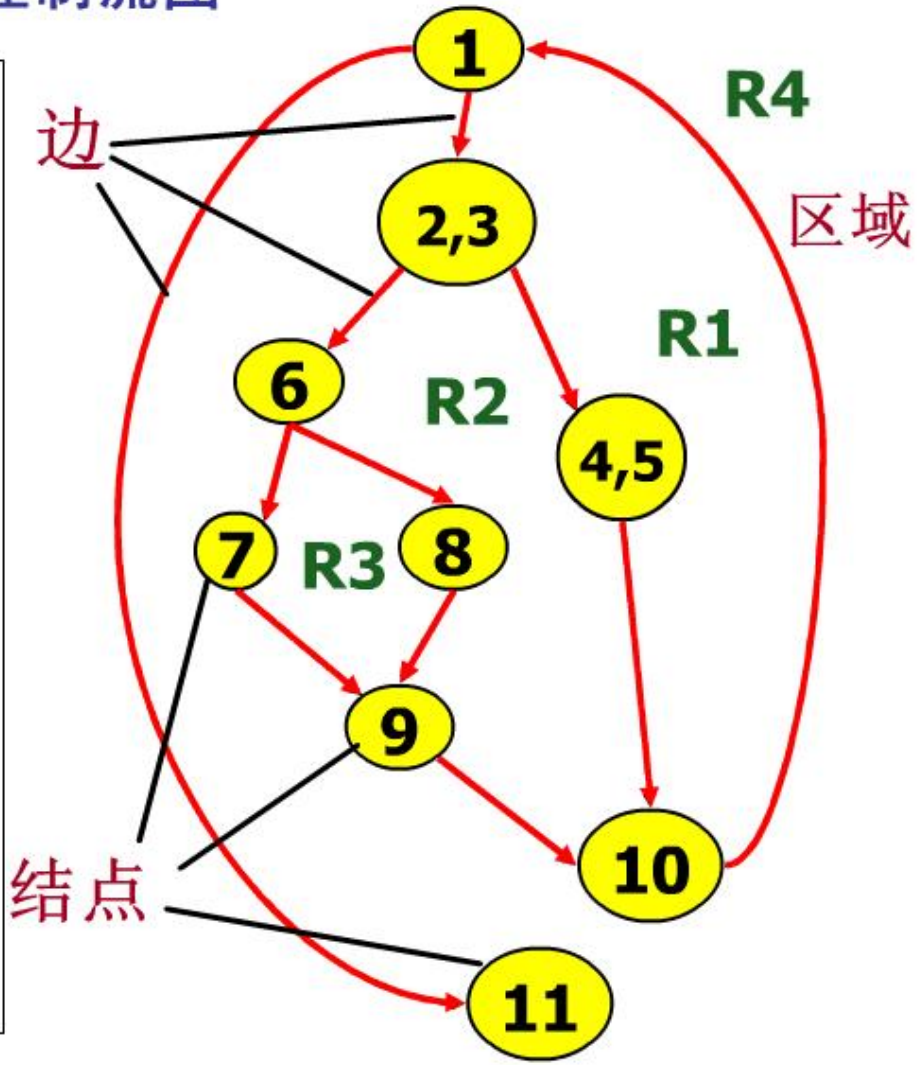
程序流程图中的顺序的处理框序列和菱形判定框，可以映射成流图中的一个结点。

流图中的箭头线称为边，代表控制流。

程序复杂程度的度量

程序流程图与对应的控制流图





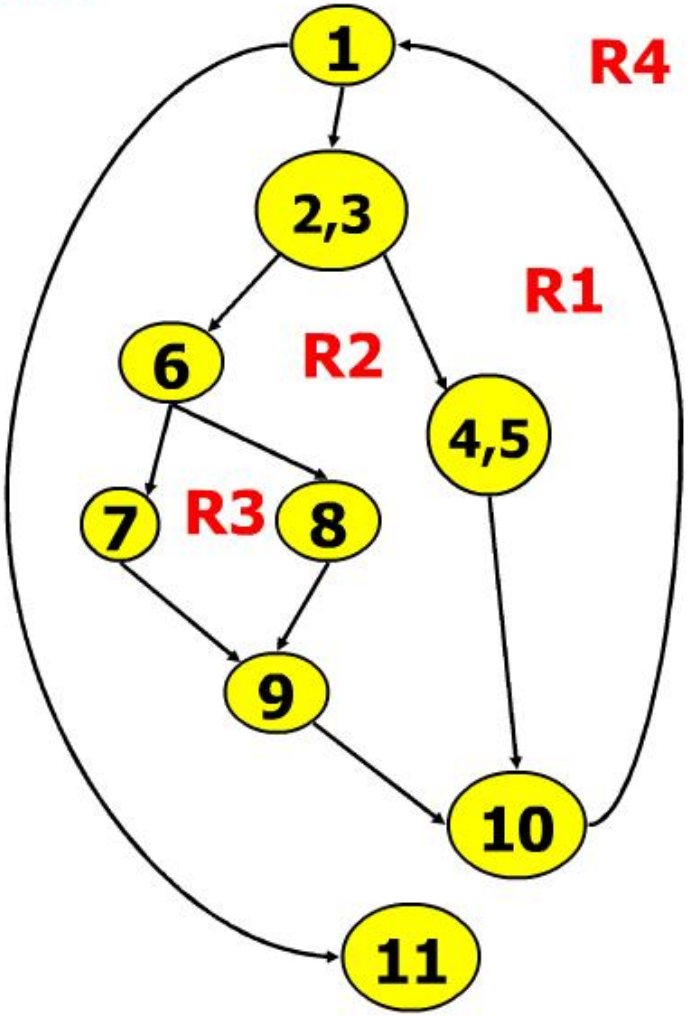
程序复杂程度的度量

2、环路复杂性V(G)的计算方法

方法一：

V（G）＝流图中区域数（包括图外区域）

如右图：V（G）＝4



程序复杂程度的度量

2、环路复杂性V(G)的计算方法

方法二：

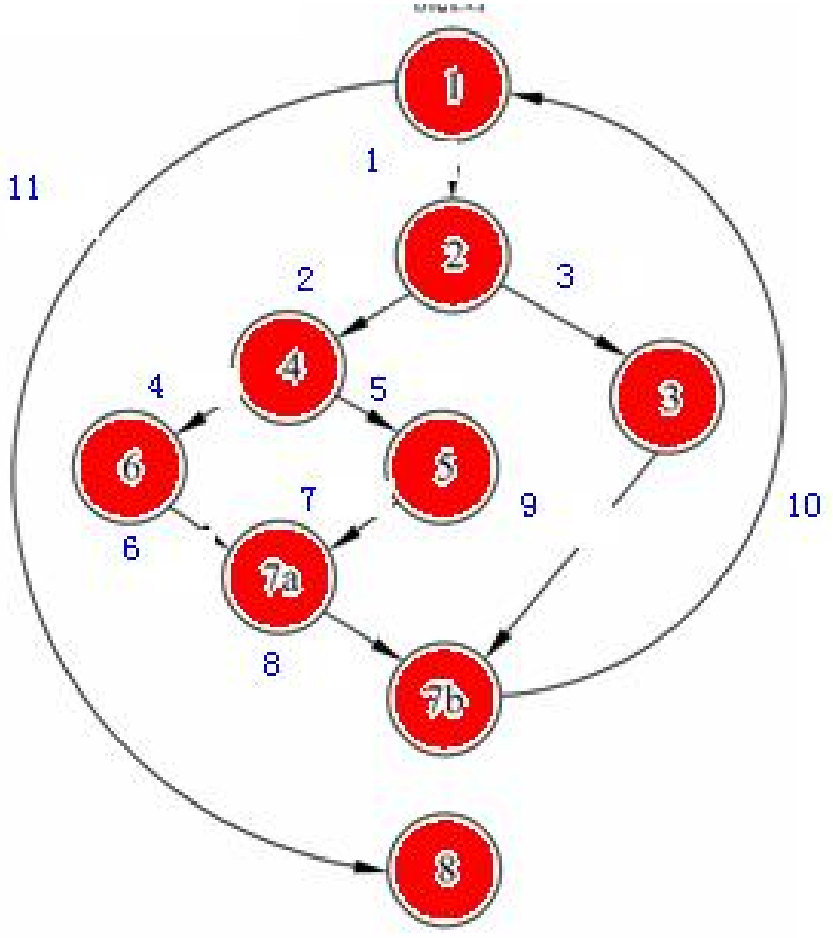
V(G) = E-N+2

= 11-9+2

= 4

E：流图边的条数

N：结点数



程序复杂程度的度量

2、环路复杂性V(G)的计算方法

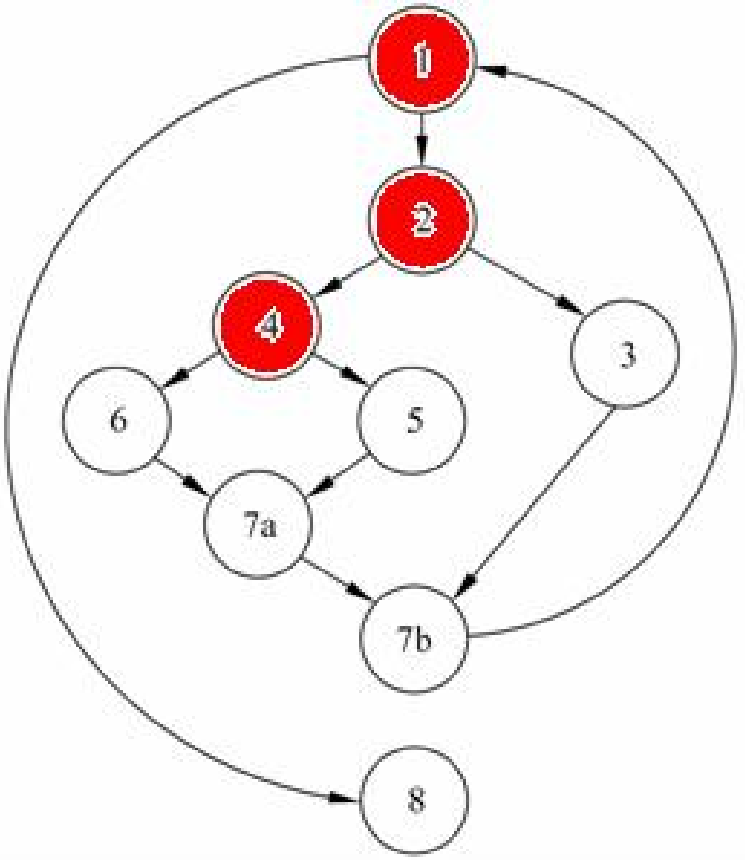
方法三：

V(G) = P+1

= 3+1

= 4

P：判定结点的数目



程序复杂程度的度量

程序的环形复杂度，取决于程序控制流的复杂程度，也即是取决于程序结构的复杂程度。

McCabe研究大量程序后发现，环形复杂度高的程序，往往是最困难、最容易出问题的程序。

模块规模以V(G)≤10为宜，也就是说，V(G)=10是模块规模的一个更科学更精确的上限。

程序复杂程度的定量度量

二 Halstead方法

$N=N1+N2$

N1：运算符出现的总次数；
N2：操作数出现的总次数。

预测程序长度的公式为：

$H=n1\log_2 n1+n2\log_2 n2$

小结

结构化设计方法

程序流程图，盒图，PAD图，PDL语言

程序的复杂程度

作业

画出以下程序的盒图

```
BEGIN
   FIRST:=K[1];
   SECOND:=0;
      FOR I:=2  TO  N  DO
       BEGIN  IF  K[I]>SECOND THEN
            BEGIN   IF  K[I]>FIRST
                 THEN   BEGIN SECOND:=FIRST;
                              FIRST:=K[I]
                        END
                ELSE   SECOND:=K[I]
          END
       END
END.
```

谢 谢！