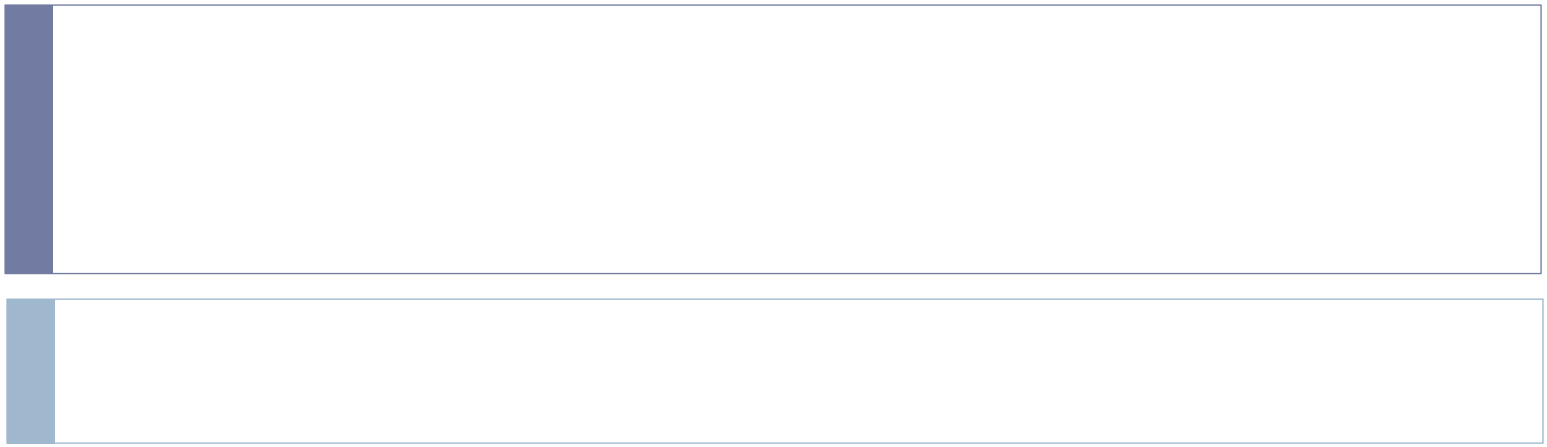
软件工程导论 –

第6章 详细设计



目录

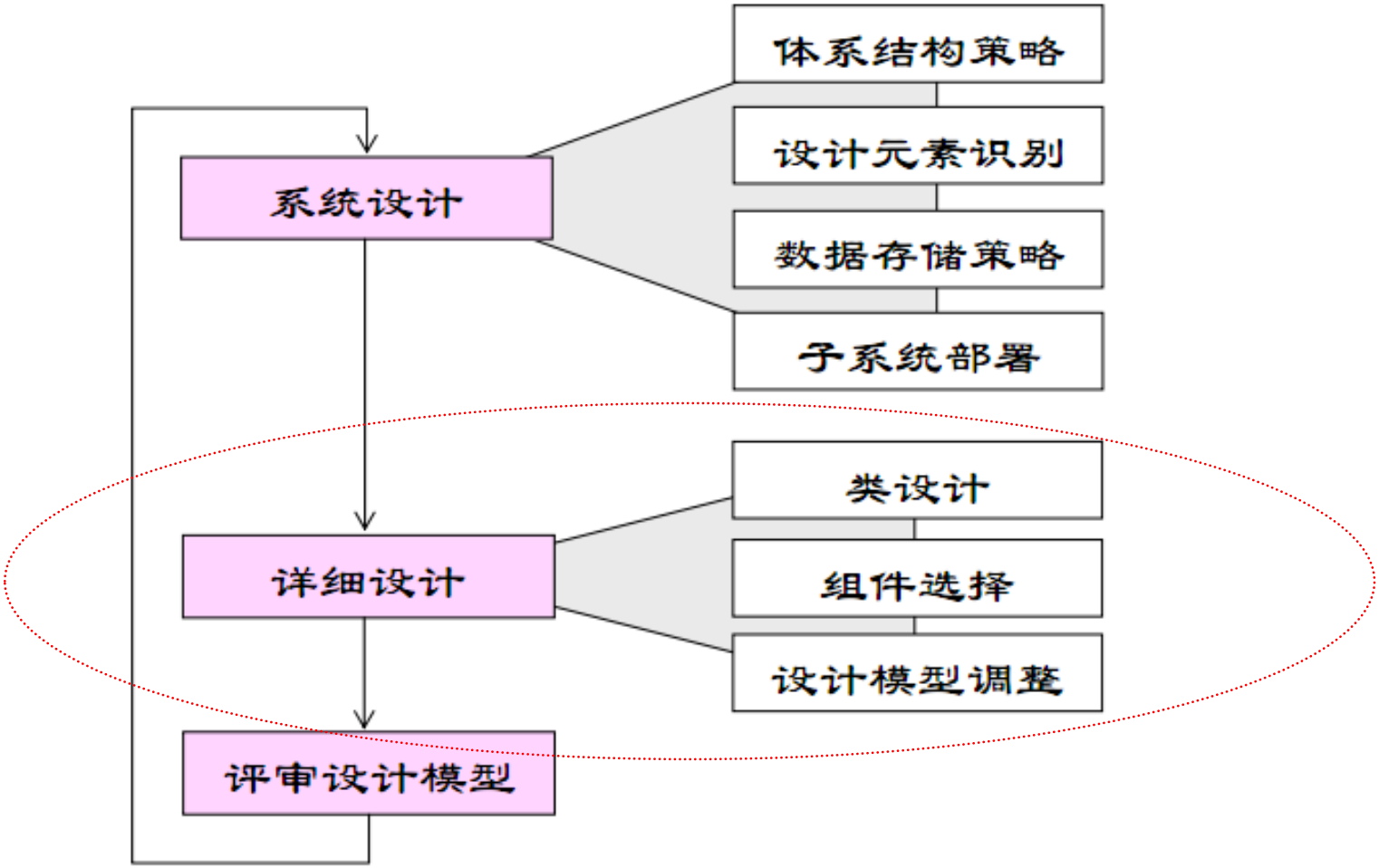
第6章　详细设计

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。

面向对象的设计过程


体系结构策略
设计元素识别
数据存储策略
子系统部署
系统设计
类设计
详细设计
组件选择
设计模型调整
评审设计模型

6.1 详细设计的任务和原则

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。

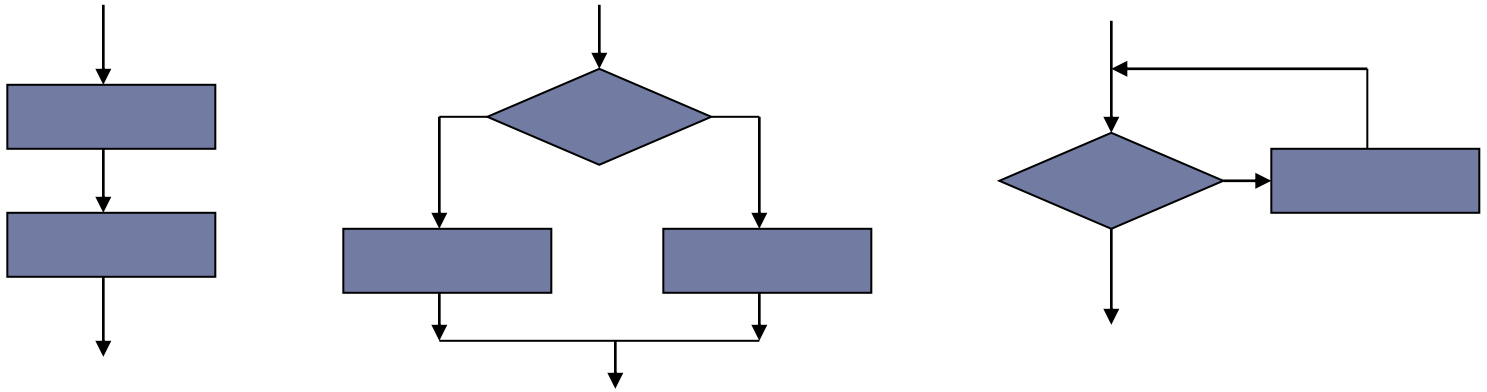
（2）采用结构化程序设计方法，改善控制结构，降低程序复杂度，提高程序的可读性、可测试性和可维护性。

6.2 结构程序设计

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。

6.2　结构程序设计

- 任何程序逻辑都可用顺序、选择、和循环等三种基本结构来表示。



6.3 详细设计的工具

一、程序流程图

二、N-S图

三、PAD图

四、PDL语言

五、详细设计工具的选择

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。

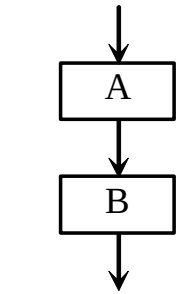
- 它独立于任何一种程序设计语言，比较直观和清晰地描述过程的控制流程，易于学习掌握。因此，至今仍是软件开发者最普遍采用的一种工具。

流程图也存在一些严重的不足：

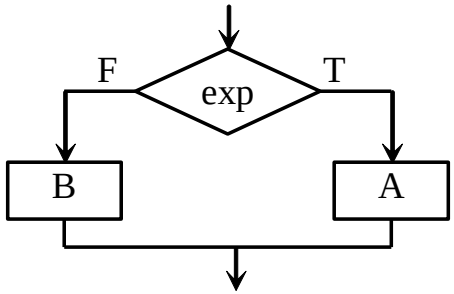
程序流程图虽然比较直观，灵活，并且比较容易掌握，但是它的随意性和灵活性却使它不可避免地存在着一些缺点：

- （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。

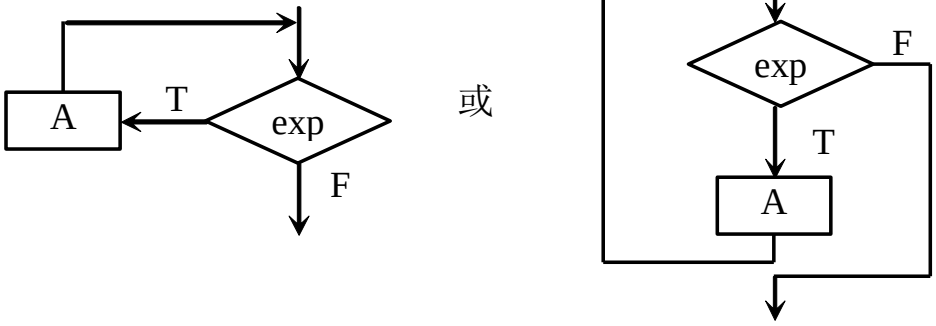
流程图的五种基本控制结构：



(a) 顺序结构



(b) 选择结构



(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。

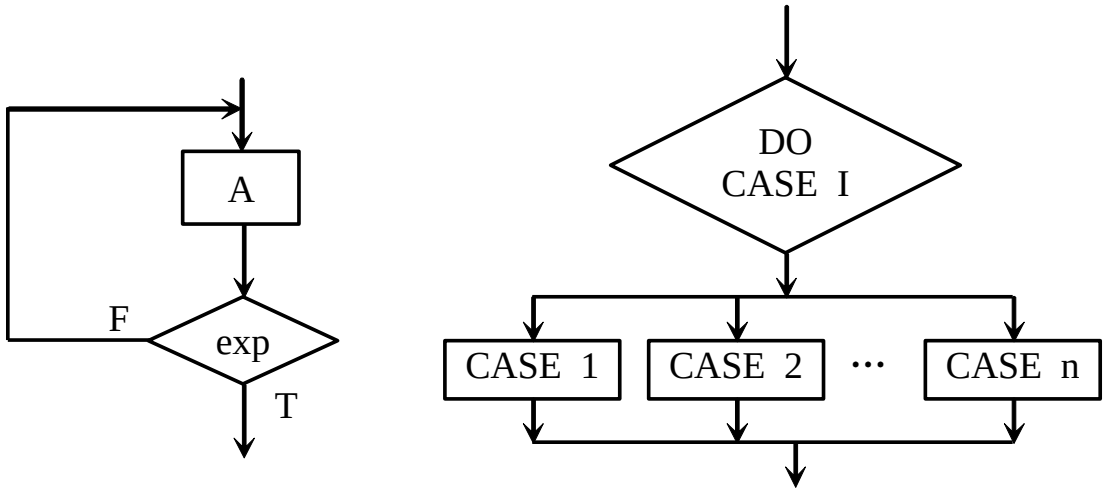
2、选择型

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。

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

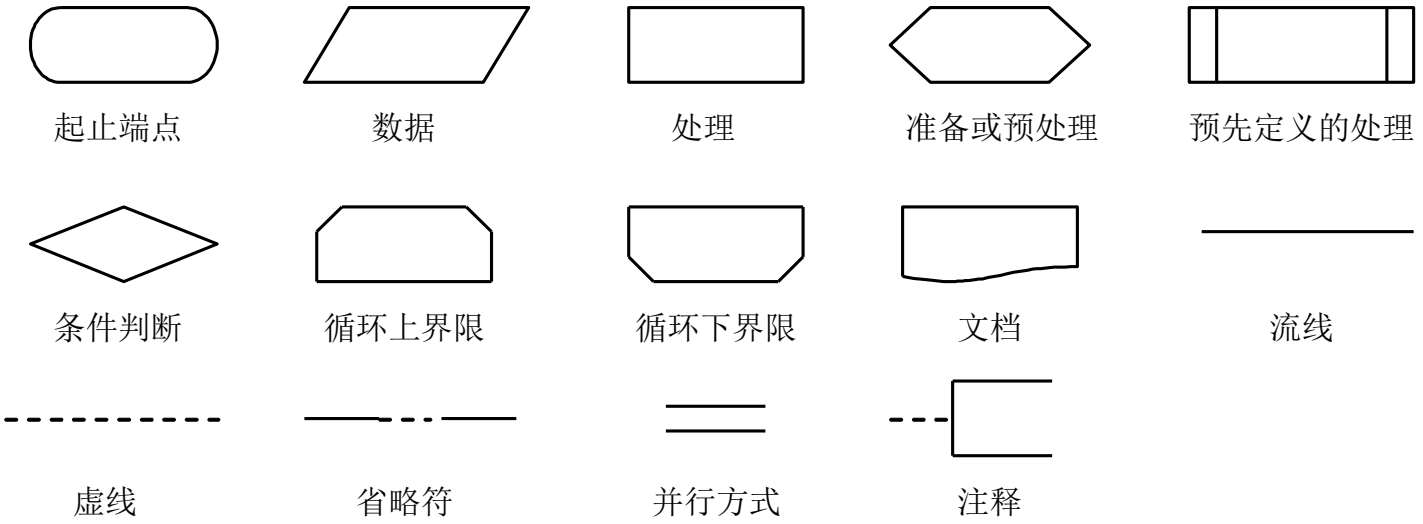
4、until型循环

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。

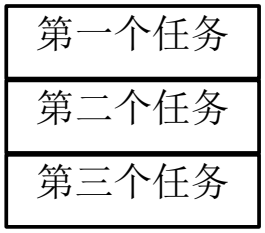
5、多情况型选择

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。

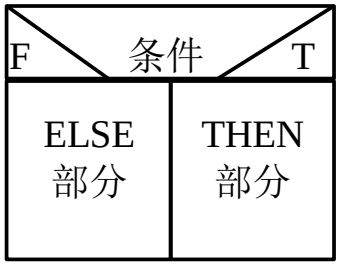
程序流程图中常用的符号：



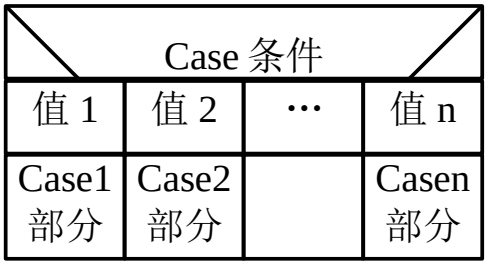
盒图（N-S图）



(a) 顺序结构

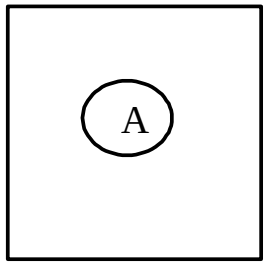


(b) 选择结构



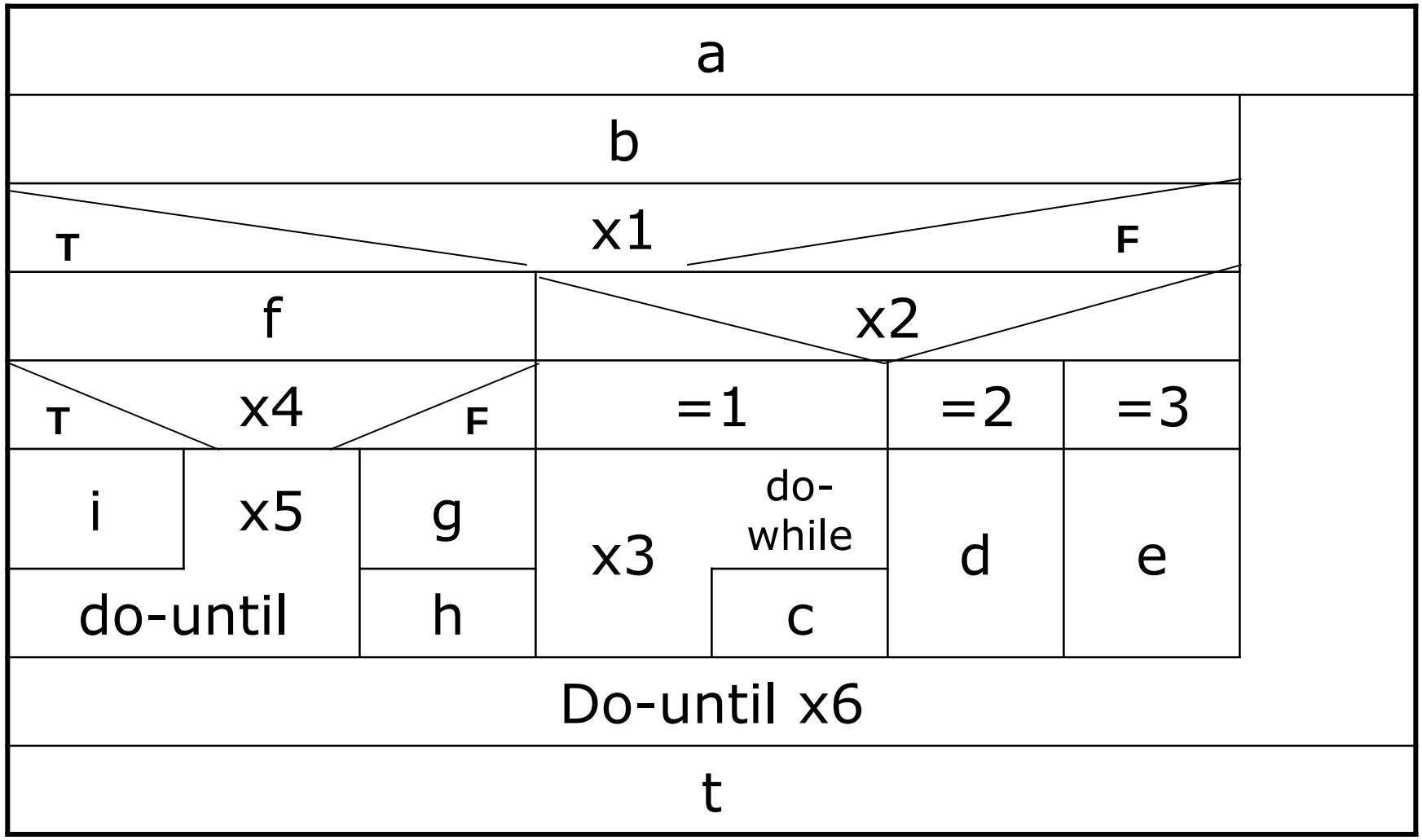
(c) 多分支结构

(d) 循环结构

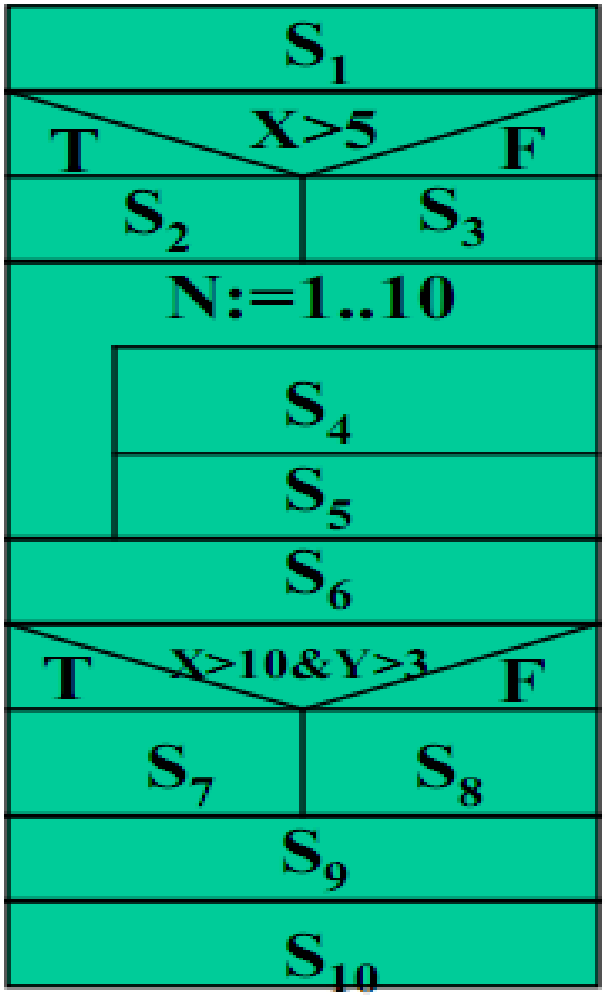


(e) 调用子程序 A

N-S图应用举例


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

N-S图逐步求精举例



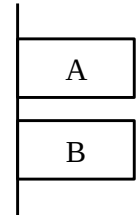
N-S图有以下一些特点：

- （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。

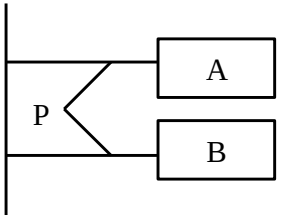
PAD图

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。

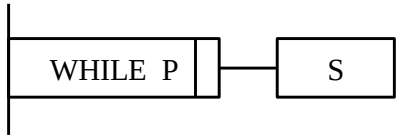
PAD也设置了五种基本控制结构的图示，并允许递归使用。



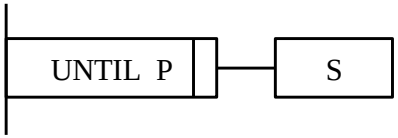
(a) 顺序结构



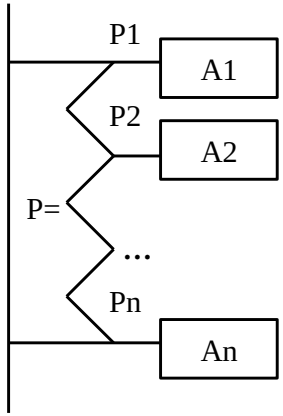
(b) 选择结构



(c) WHILE 型循环结构

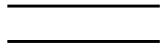


(d) UNTIL 型循环结构



(e) 多分支结构

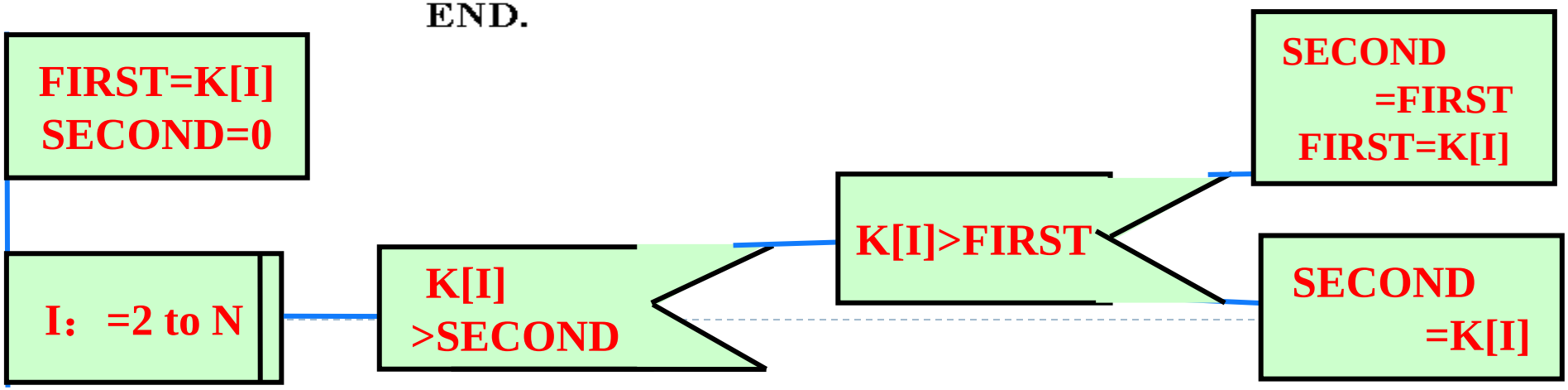
(f) 语句标号



(g) 定义

PAD图应用举例

```
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.
```

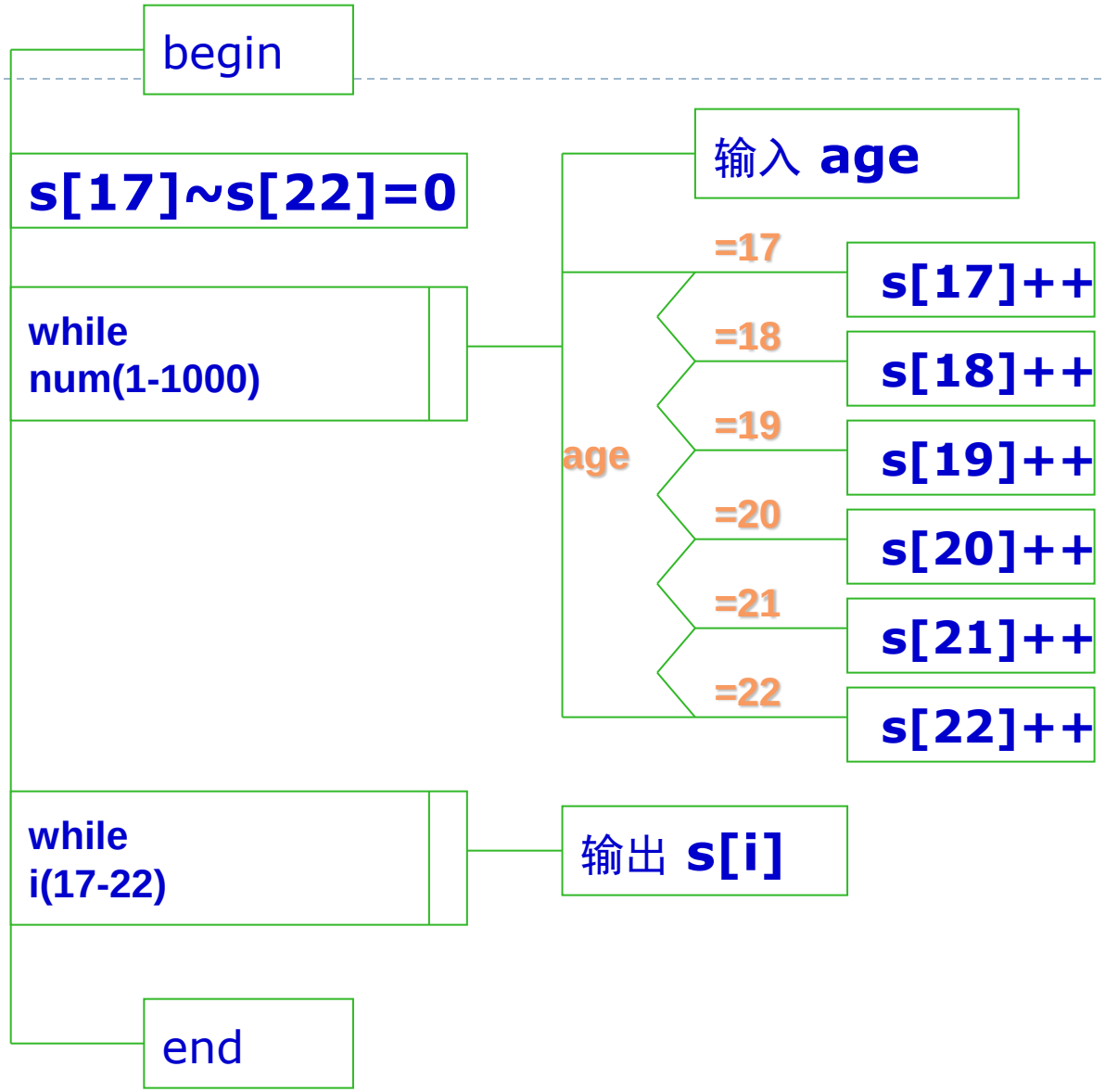


第四次作业

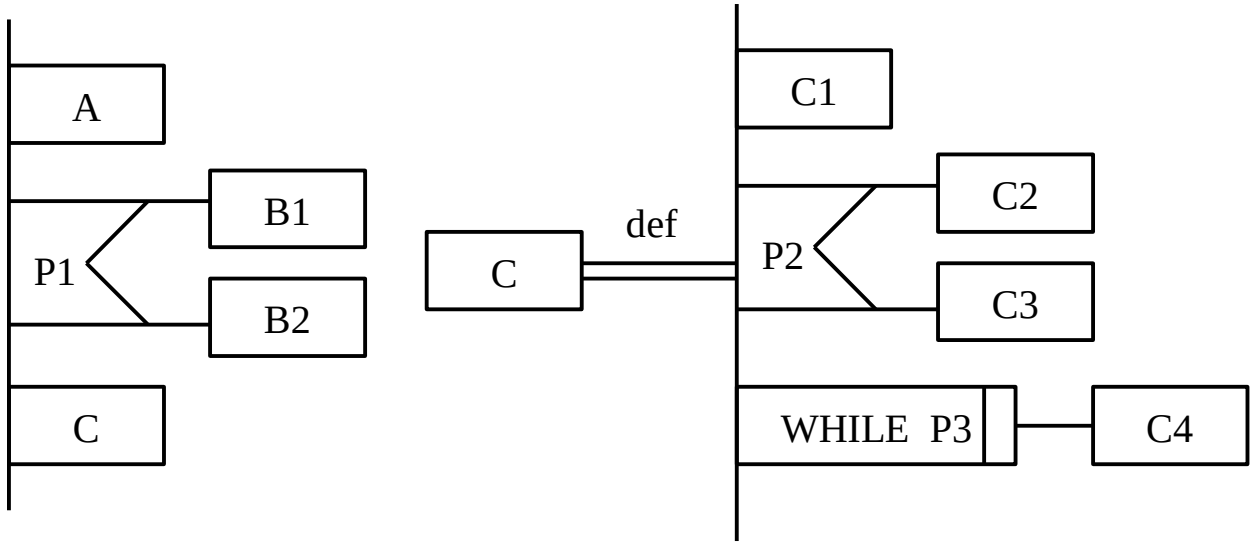
- 画出以上程序的盒图。

```
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.
```

更多例子


begin
s[17]~s[22]=0
while
num(1-1000)
输入 age
age
=17
s[17]++
=18
s[18]++
=19
s[19]++
=20
s[20]++
=21
s[21]++
=22
s[22]++
while
i(17-22)
输出 s[i]
end

PAD图提供的定义功能



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。（利用扩充结构）
6. 既可以用于表示程序逻辑，也可以用于描绘数据结构。

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，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。

过程设计语(PDL)

- 也称为伪码
- 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。

示例:一元二次方程求根

```
Begin
    输入一元二次方程的系数
      a,b,c;
    if b↑2-4ac≥o then 计算两
    实根     else 输出无实根;
end.
```

6.5 程序复杂程度的定量度量

一 McCabe方法（环形复杂度）

环形复杂度V(G)又称为圈复杂度，为我们提供了非常有用的软件度量。

计算方法：画出程序图对应的控制流图，是程序流程图的退化。

环形复杂度计算方法：

V（G）= E-N+2
V（G）= P+1
V（G）= 区域数

E:是有图中的边数；
N:是有向图中的节点数；
P:判定节点数

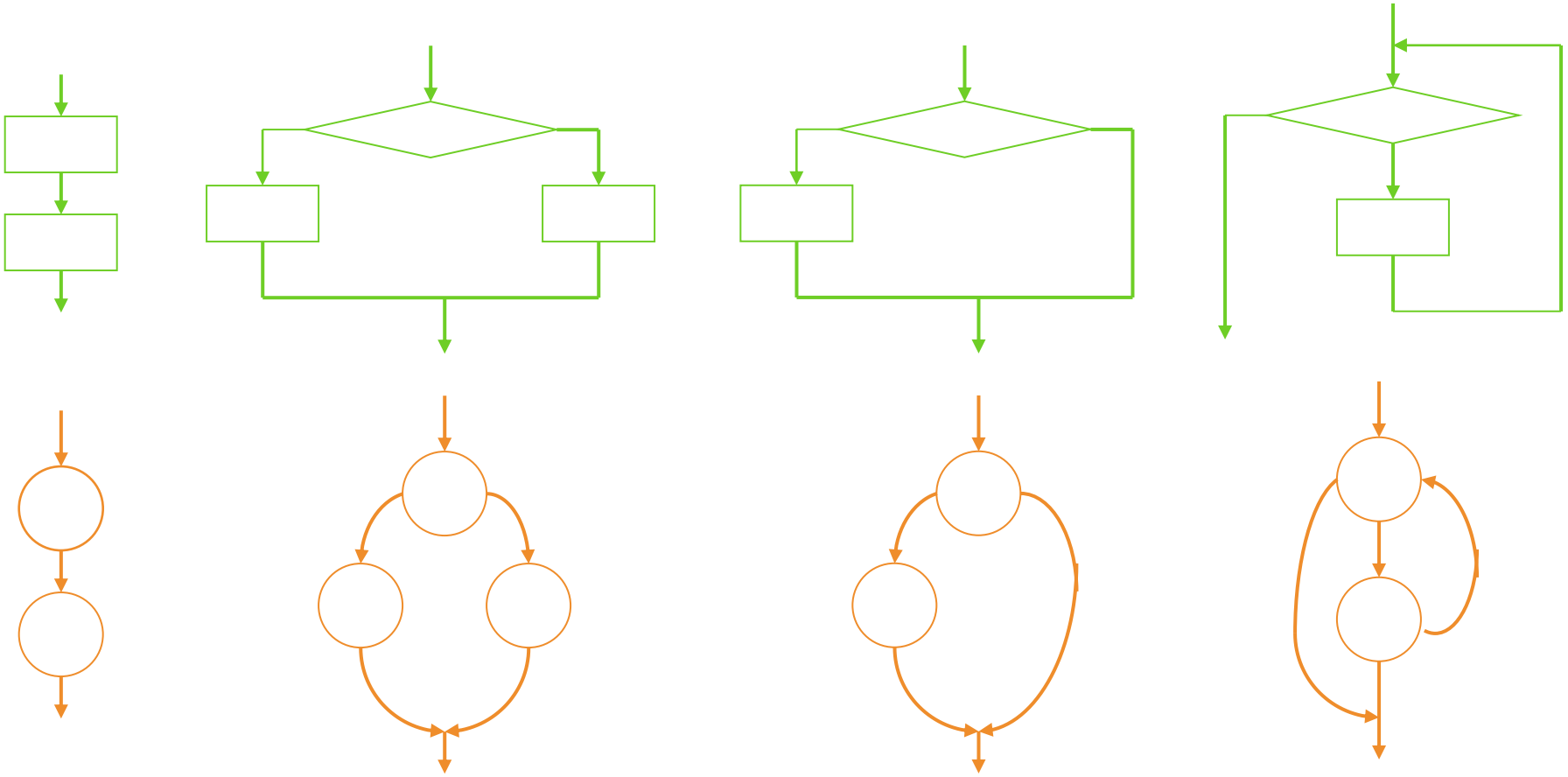
V(G)<= 10的模块规模是比较好的

利用程序的控制流来度量程序的复杂性

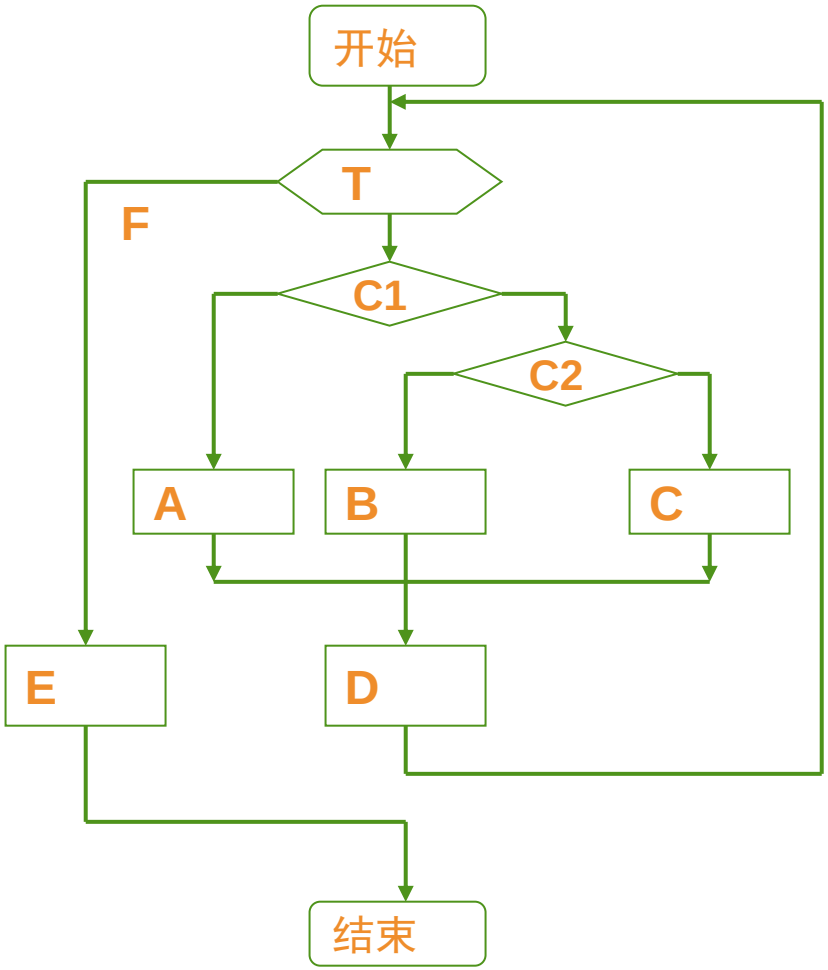
该方法是利用程序模块的**程序图**中环路的个数，来计算程序的复杂性的。为此，该方法也称为环路复杂度计算法。

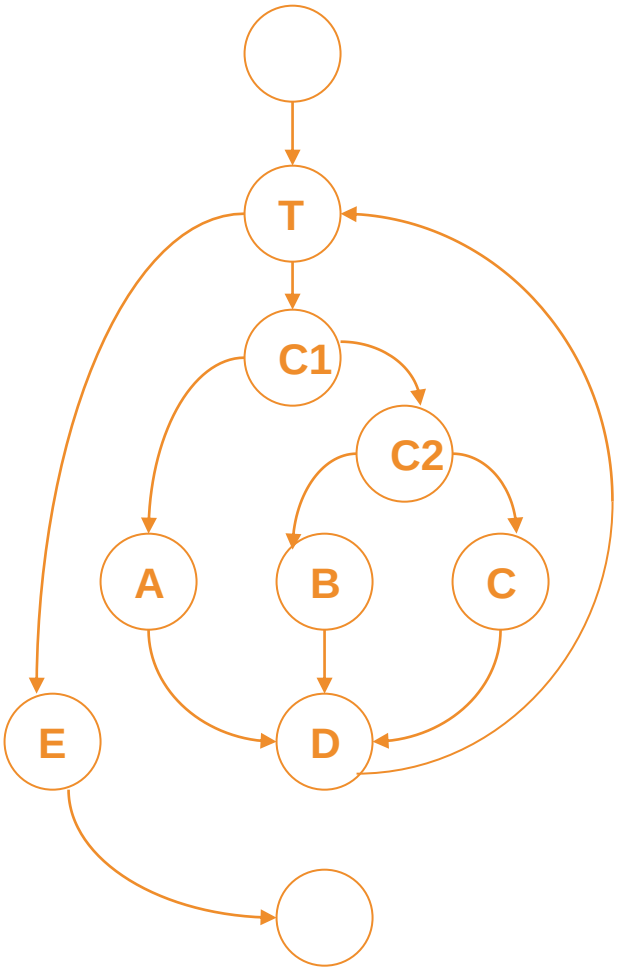
它是一种退化了的程序流程图。即:把程序流程图中每个处理符号都退化成一个**结点**，而原来流程图中的流程线，则变成连接不同结点的**有向弧**。

(1) 程序图符号



(2) 从流程图导出程序图



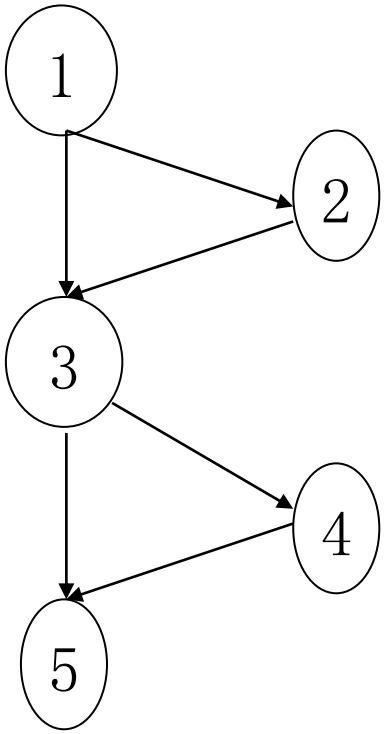


(3)环路复杂性的计算方法

V（G）= E-N+2
V（G）= P+1
V（G）= 区域数

E:是有图中的边数；
N:是有向图中的节点数；
P:判定节点数

程序复杂程度的定量度量



复杂度？

=

(4) 环路复杂度的用途

程序的环路复杂度则取决于程序控制流的复杂度，也就是取决于程序结构的复杂程度。当程序内分支或循环个数增加时，则相应地环域复杂度也随之增加。因此，它是对测试难度的一种定量度量，也能对软件最终的可靠性给出某种预测。

V(G)<=10

掌握要求：能够计算一段代码的圈复杂度

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
- 程序的复杂程度

附录：详细设计的规约 例

附：模块说明表

模块说明表　　制表日期：　年　月　日

模块名：	模块编号：	设计者：
模块所在文件：	模块所在库：	
调用本块的模块名：		
本模块调用的其他模块名：		
功能概述：		
处理描述：		
引用格式：		
返回值：		

END.