详细设计工具

- 详细设计概览
- 伪码
- 程序流程图
- PAD图
- N-S图
- 判定表和判定树

二、 详细设计层

详细设计的任务：定义每一模块

--主要引入了关于三种动作控制结构的术语/符号

三种控制结构:顺序,选择和循环

这三种结构在表达系统行为方面是完备的

- 结构化程序设计的概念:

设计具有如下结构的程序：一个程序的代码块仅仅通过顺

、选择和循环这3种基本控制结构进行连接，并且每个代码块只有一个入口和一个出口。

- 结构化程序设计的提出:
  - 1966年，C.Bohm和G.Jacopini证明只用三种基本控制结构就能实现单入口和单出口的程序
  - 1968年，Dijkstra的短文“Goto Statement considered harmful”引发了程序结构设计的讨论，明确了结构化程序设计思想.

详细设计工具

- 详细设计概览
- 伪码
- 程序流程图
- PAD图
- N-S图
- 判定表和判定树

第一种表达-伪码（类程序设计语言PDL,Program Design language）

顺序

begin s1;s2;…sn end;

选择

if 条件表达式 then s1
else s2;

循环

while 条件表达式 do s ;

伪码是一种混合语言。外部采用形式语言定义控制结构和数据结构，内部使用自然语言。

例如:

```
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.
```

详细设计工具

- 详细设计概览
- 伪码
- 程序流程图
- PAD图
- N-S图
- 判定表和判定树

优点：PDL不仅可以作为设计工具，而且可作为注释工具，直接插在源程序中间，以保持文档和程序的一致性，提高了文档的质量.

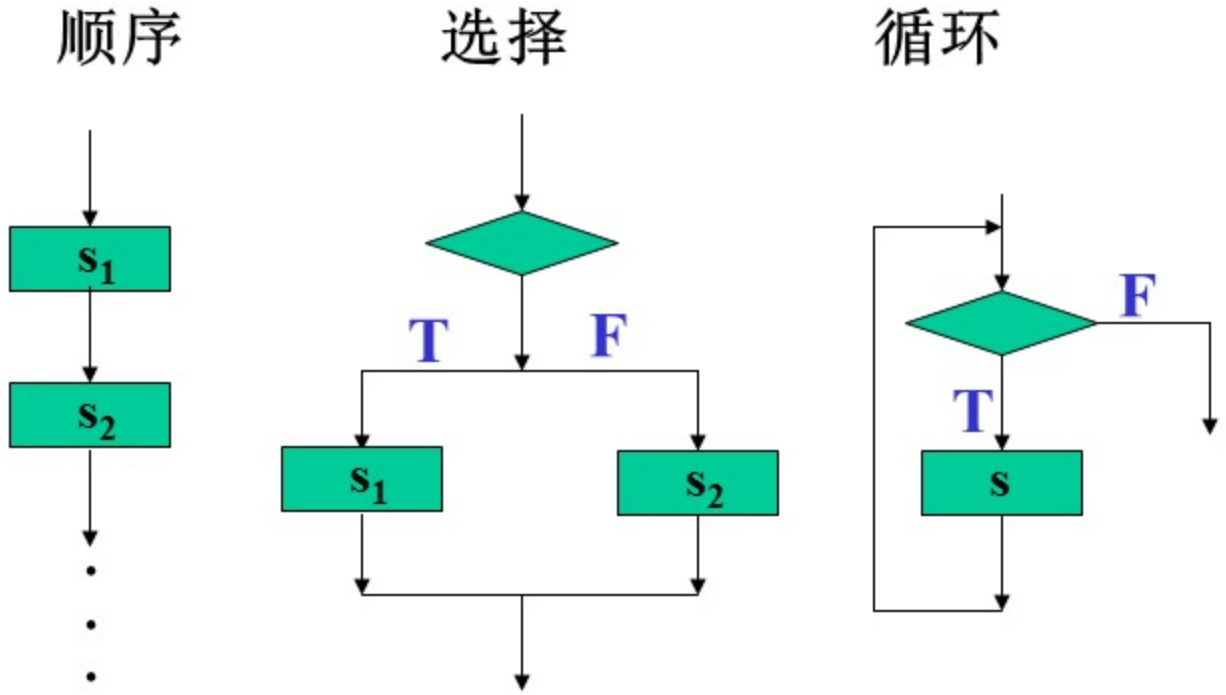
缺点：1．不如图形工具那样形象直观.

2．当描述复杂的条件组合与动作间的对应关系时，不如判定表和判定树那样清晰简单.

详细设计工具

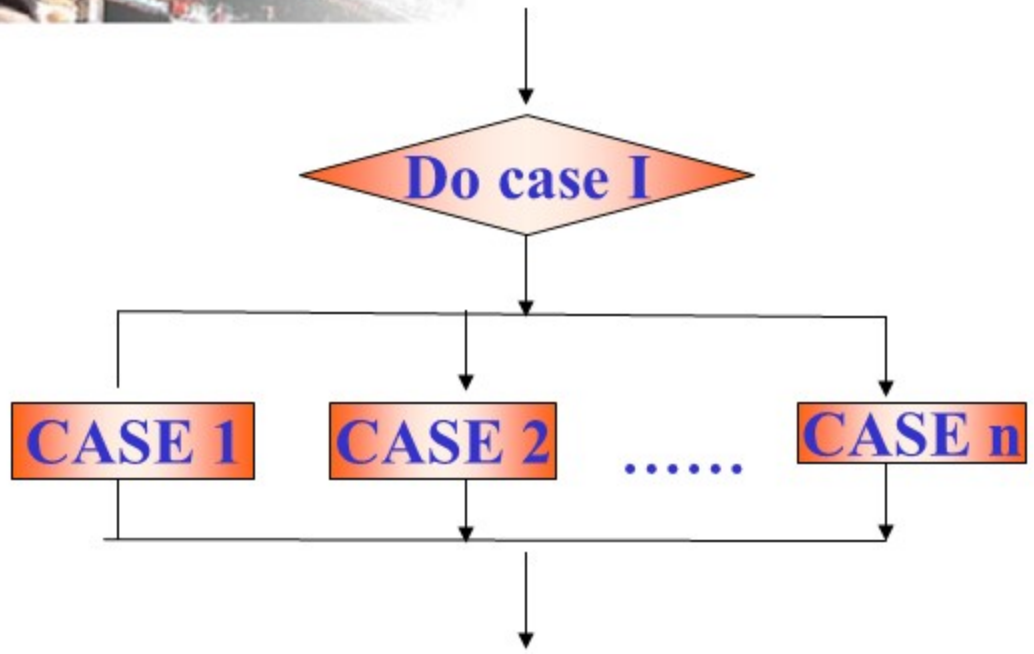
- 详细设计概览
- 伪码
- 程序流程图
- PAD图
- N-S图
- 判定表和判定树

第二种表达-程序流程图（程序框图）

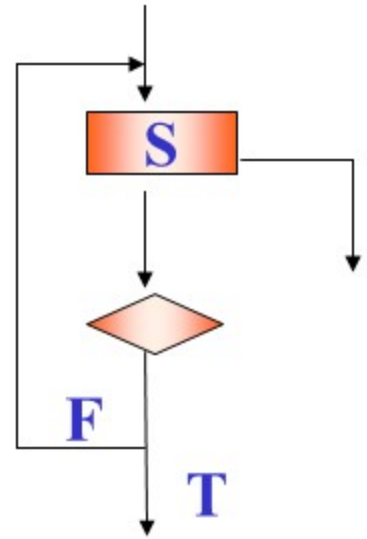


详细设计工具

- 详细设计概览
- 伪码
- 程序流程图
- PAD图
- N-S图
- 判定表和判定树



多分支结构



Repeat-until循环结构

优点：对控制流程的描绘很直观,便于初学者掌握.

缺点：1．不是一种逐步求精的工具，程序员过早地考虑程序的控制流程，而不是全局结构.

2．所表达的控制流，可以不受约束随意转移.

3．不易表示数据结构.

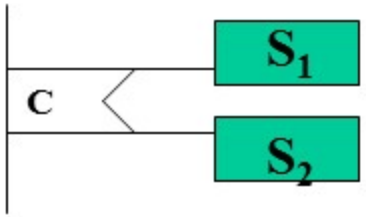
详细设计工具

- 详细设计概览
- 伪码
- 程序流程图
- PAD图
- N-S图
- 判定表和判定树

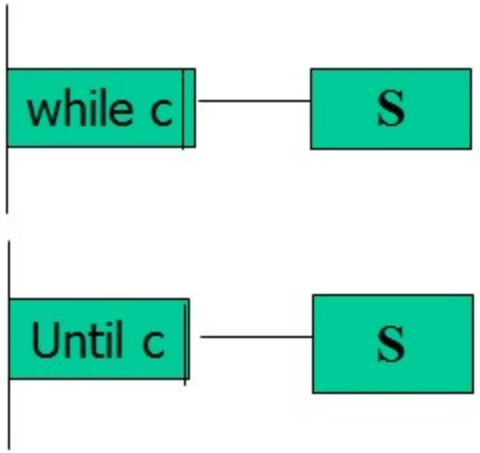
第三种表达—PAD图(Problem Analysis Diagram)

顺序：

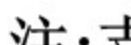
选择：



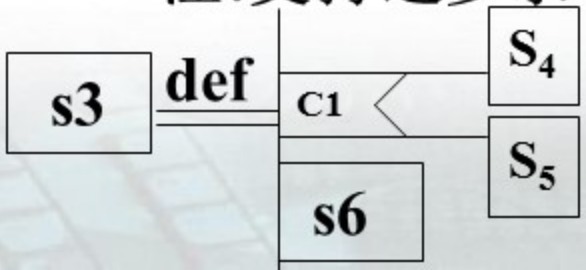
循环：



注:支持逐步求精设计:def



例如：



详细设计工具

- 详细设计概览
- 伪码
- 程序流程图
- PAD图
- N-S图
- 判定表和判定树

优点：

1．支持自顶向下逐步求精的结构化详细设计，可使用“def”符号逐步增加细节.

2．PAD图最左边的竖线是程序的主线，随着程序层次的增加，逐步向右延伸，每增加一个层次，图形向右扩展一条竖线，从而使PAD图所表现的处理逻辑易读、易懂和易记。

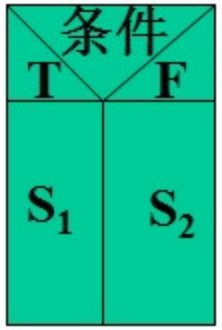
详细设计工具

- 详细设计概览
- 伪码
- 程序流程图
- PAD图
- N-S图
- 判定表和判定树

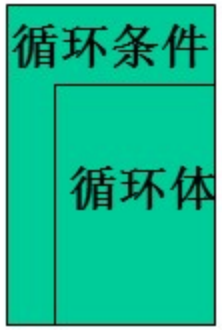
第四种表达-N-S图

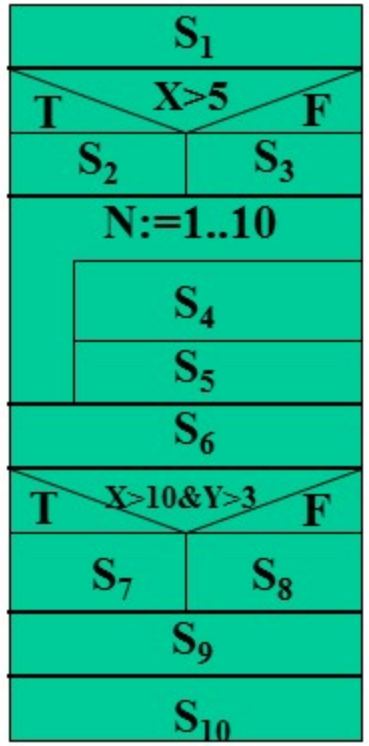
顺序：

选择：



循环：





支持逐步求精设计举例

优点：支持自顶向下逐步求精的结构化详细设计，并且严格限制了控制从一个处理到另一个处理的转移.

北京大学

详细设计工具

- 详细设计概览
- 伪码
- 程序流程图
- PAD图
- N-S图
- 判定表和判定树

第五种判定表和判定树

当算法中包含多重嵌套的条件选择时，用程序流程图、盒图、PAD图、PDL都不易清楚描述，这时可以选择判断表来表达复杂的条件组合与应做的动作之间的对应关系.

判定树是判定表的变种，也能清晰地表达复杂的条件组合与应做的动作之间的对应关系，形式简单，但简洁性不如判定表，数据元素的同一个值往往需要重复写多次，而且越接近树的叶断重复次数越多.