软件工程导论实验四

详细设计说明书主要包含如下内容：

- 系统各功能模块的英文标识。
- 详细说明模块所选用的算法（如果该模块有算法时，此项可选），具体的计算公式和计算步骤。
- 各功能模块的程序流程图。
- N-S图(也叫盒图)。
- PAD （ Problem Analysis Diagram,问题分析图）

系统各功能模块的英文标识：

可以按照：子系统名__模块名__子模块的英文名来命名。

结构化方法

结构化方法程序设计(SP)

- 基本思想是：自顶向下，采用模块化技术，分而治之，逐步求精地将系统按功能分解为若干模块进行分析与设计。
- 应用子程序实现模块化，模块内部由顺序结构、选择结构、循环结构等三大基本控制结构组成。

结构化方法

结构化方法程序设计(SP)

- 从代表目标系统整体功能的单个处理着手，自顶向下不断地把复杂的处理分解为子处理，这样一层一层地分解下去，直到仅剩下若干个容易实现的子处理为止，并写出各个最低层处理的描述。

结构化方法

- 结构化方法程序设计(SP)的基本思想就是把大的程序划分为若干个相对独立、功能简单的程序模块。它以过程为中心，强调的是过程，强调功能和模块化，通过一系列过程的调用和处理完成相应的任务。

结构化方法

- **结构化方法程序设计(SP)**的基本思想就是把大的程序划分为若干个相对独立、功能简单的程序模块。它以**过程**为中心，强调的是过程，强调功能和模块化，通过一系列过程的调用和处理完成相应的任务。

结构化语言

- **模块**是结构化语言编程的**基本单位**，**计算方法**（简称为算法）是程序的**核心**。

结构化语言设计的原则

1、使用语言中的顺序、选择、循环等有限的基本控制结构表示程序。

2、选用的控制结构只准许有一个入口和一个出口 。

3、程序语句组成容易识别的块（Block），每块只有一个入口和一个出口。

4、复杂结构应该用基本控制结构进行组合嵌套来实现 。

5、严格控制GOTO语句。

结构化语言

结构化语言编程是一种设计程序的技术，它采用自顶向下逐步细化的设计方法和单入口(Single entry)、单出口（Single exit）的控制结构。这种控制结构包括有：顺序、选择和循环。

结构化程序设计

基本思想：顺序结构

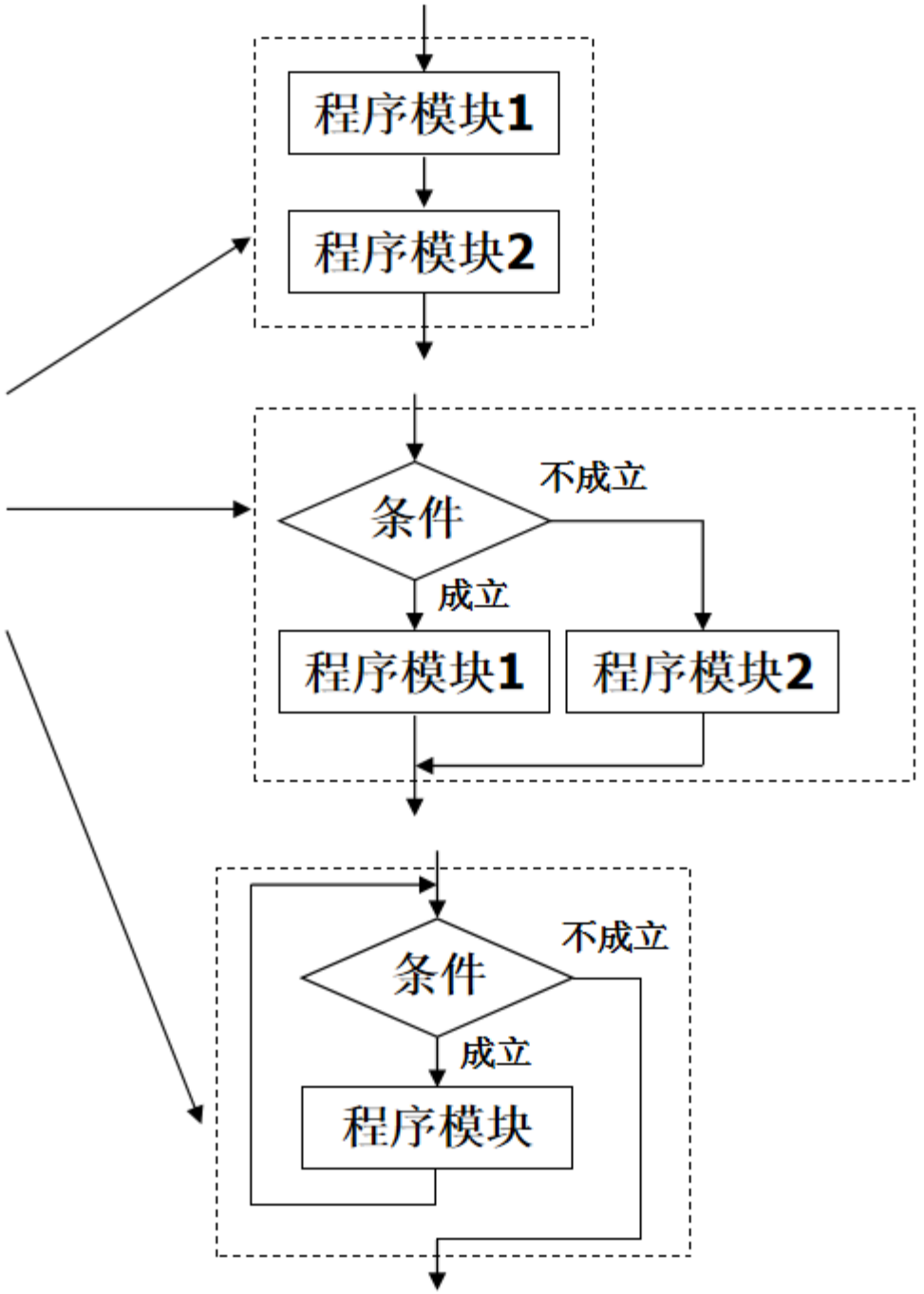
选择结构

循环结构

设计原则：自顶向下

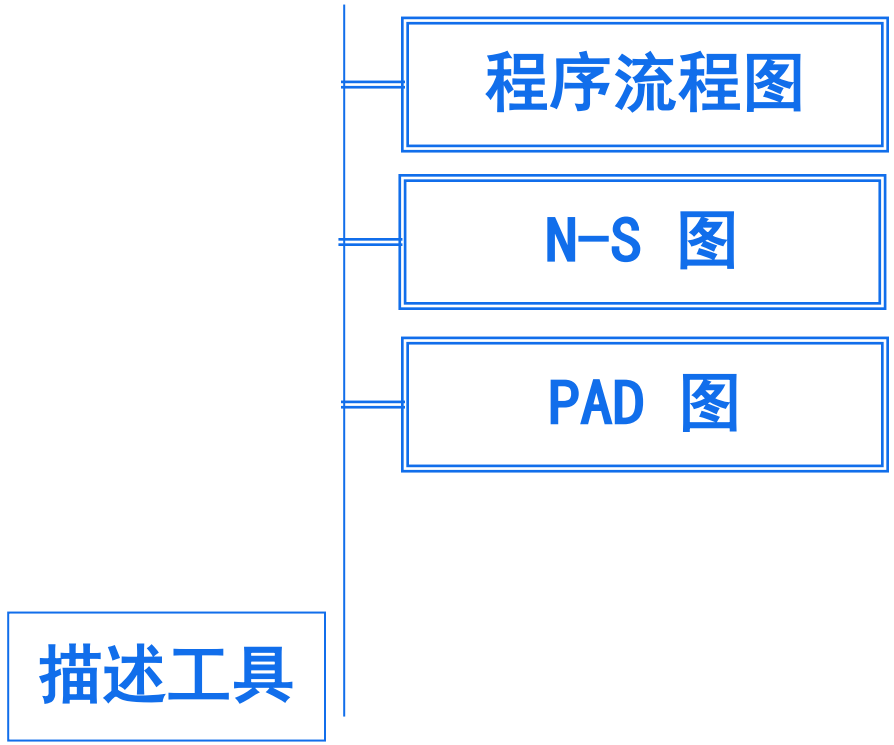
逐步求精

模块化



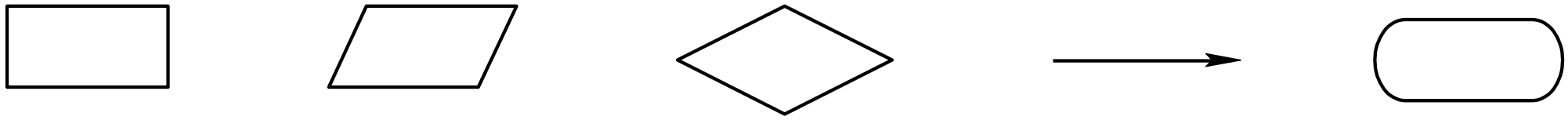
结构化程序设计语言

详细设计阶段的描述工具：



结构化语言设计的常用工具—程序流程图

程序流程图是最早出现且使用较为广泛的算法表达工具之一，能够有效地描述问题求解过程中的程序逻辑结构。程序流程图中经常使用的基本符号如下图所示。



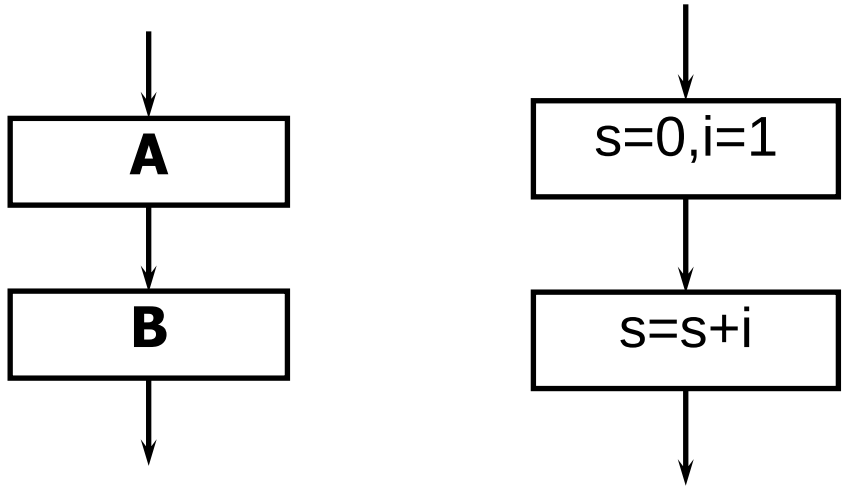
(a) (b) (c) (d) (e)

程序流程图中的基本符号

(a) 一般处理框；(b) 输入/输出框；(c) 判断框；(d) 流程线；(e) 起止框

程序流程图的三种基本控制结构

1）顺序结构：几个连续的加工依次序排列，相当于“A、B”。

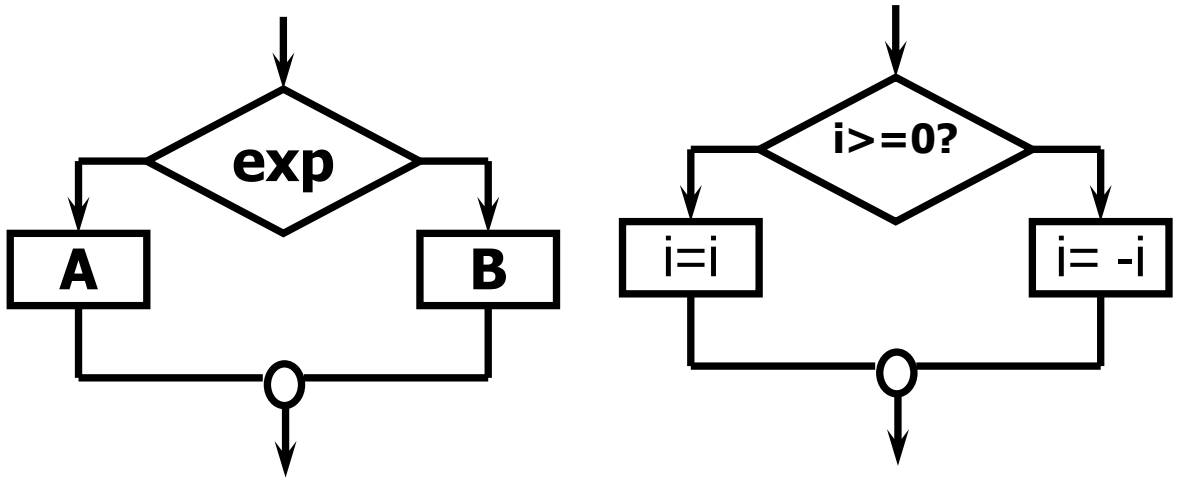


(a) 顺序结构

程序流程图的三种基本控制结构

2）选择结构

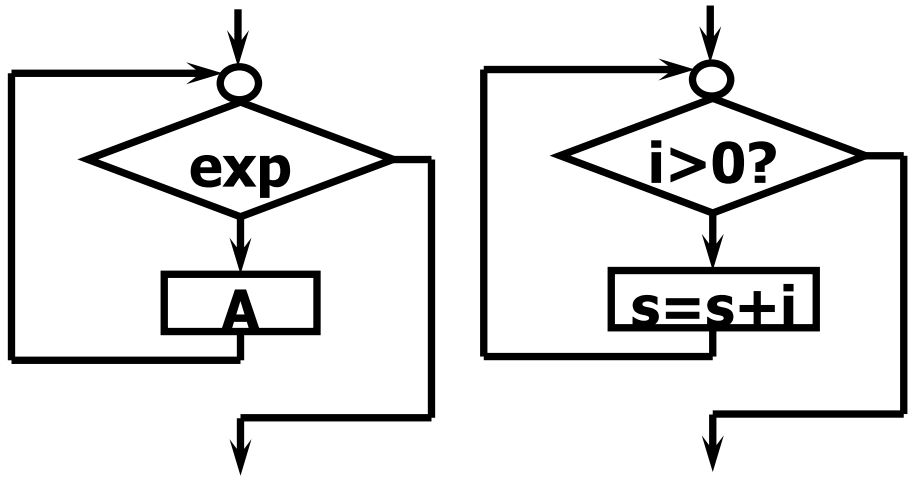
由某个判断式的取值决定选择两个加工中的一个，相当于“If exp then A else B endif”。



(b) 选择结构

程序流程图的三种基本控制结构

3）循环结构：当循环控制条件成立时，重复执行特定的加工，相当于“While exp do A” 。

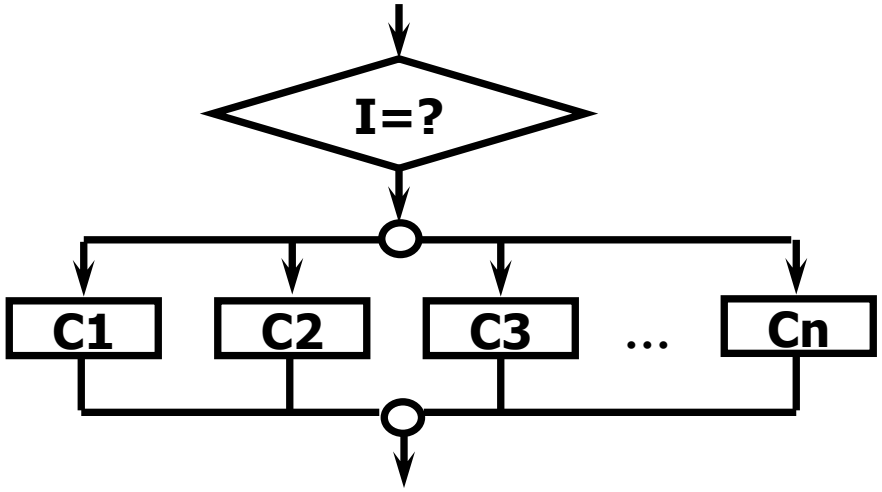


(c)循环结构

程序流程图扩充两种控制结构

1）多分支结构

列出多种加工情况，根据控制变量的取值，选择执行其一。相当于“Case I of I=1:C1; I=2:C2; I=3:C3; … ; I=n:Cn”。

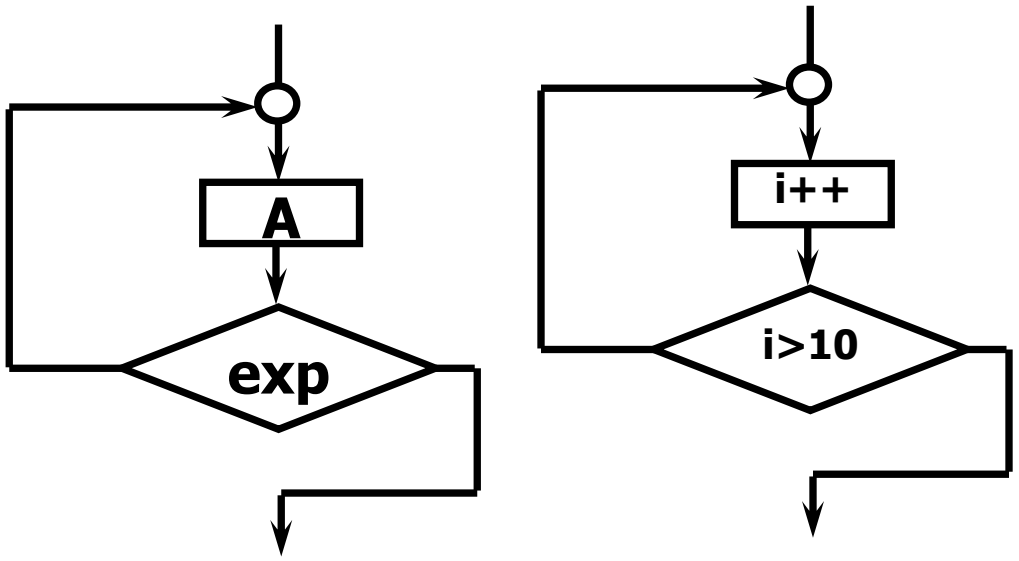


(d) 多分支结构

程序流程图扩充两种控制结构

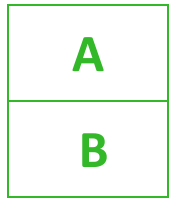
2）UNTIL循环结构

重复执行特定的加工，直到循环控制条件成立时，相当于“Repeat A Until exp”。

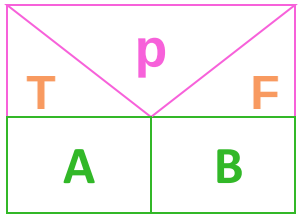


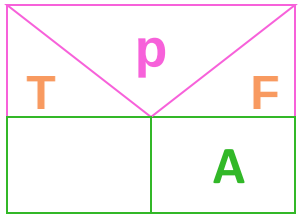
(e)UNTIL循环

N-S 图 ------- Nassi and Shneideman

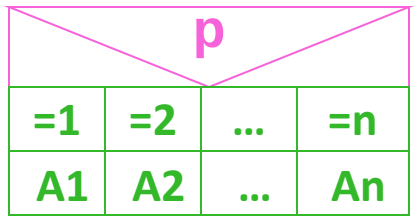


顺序型

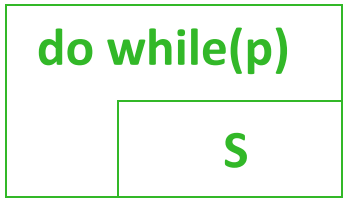




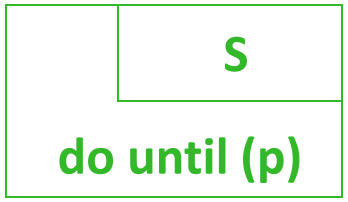
选择型



多分支选择型

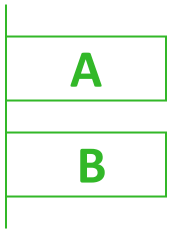


当型循环型

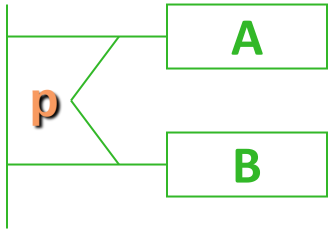


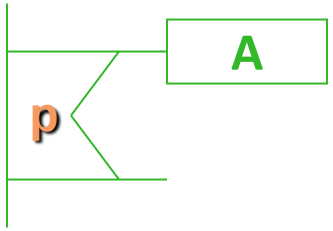
直到型循环型

PAD 图 ------- Problem Analysis Diagram

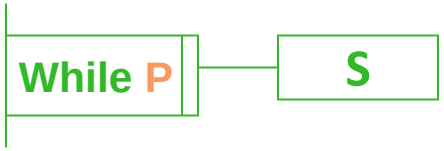


顺序型

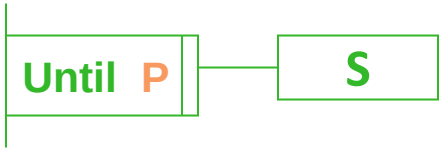




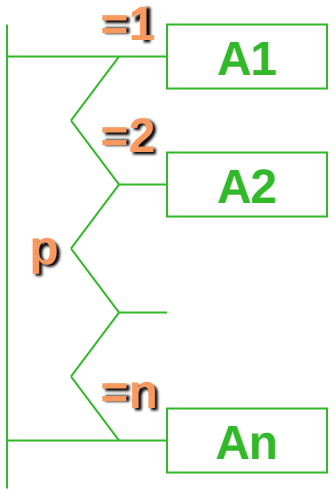
选择型



当型循环型

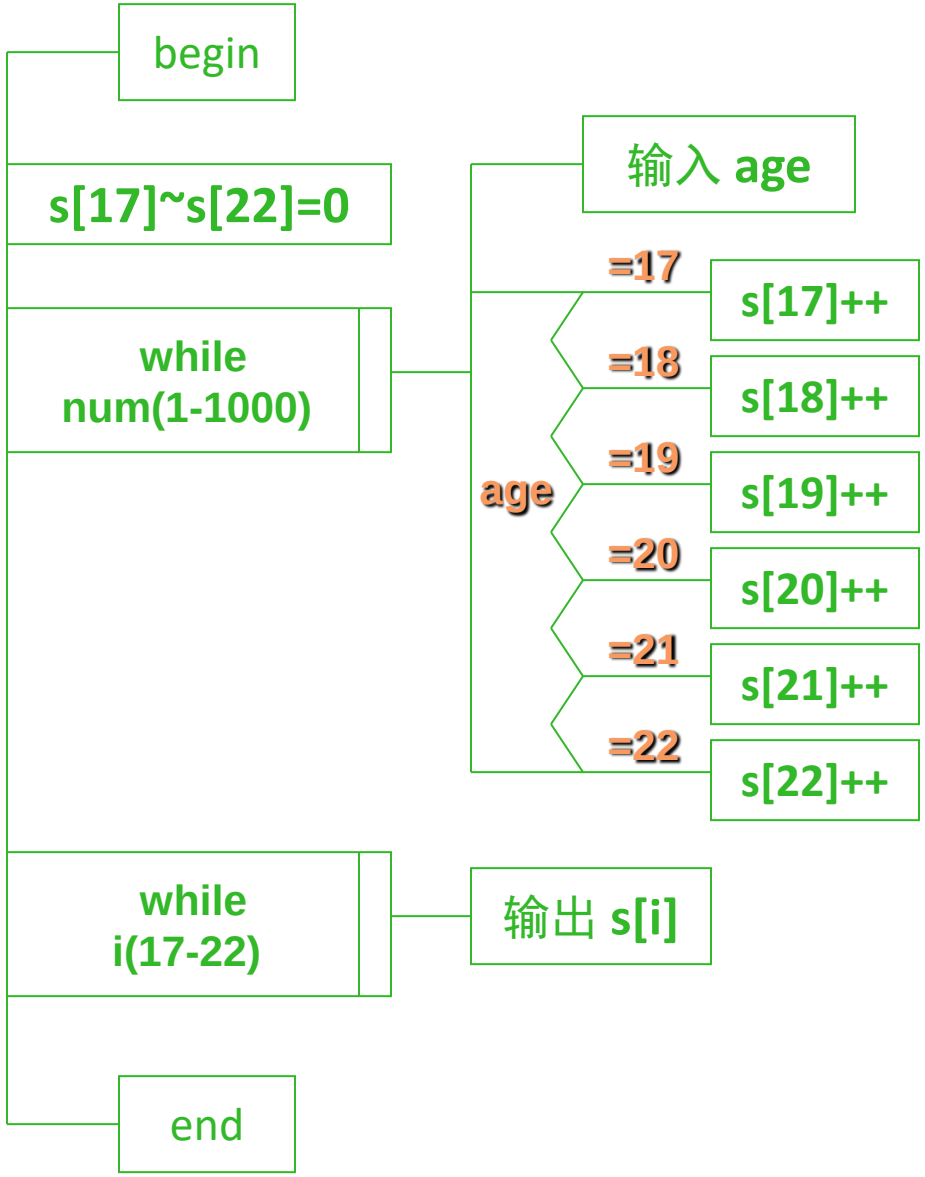


直到型循环型



多分支选择型循环型

软件工程导论实验


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

程序效率

程序效率是指程序的执行速度及程序占用的存储空间。程序编码是最后提高运行速度和节省存储机会，因此在此阶段不能不考虑程序的效率。

算法对效率的影响

源程序的效率与详细设计阶段确定的算法的效率有着直接的关系。当我们把详细设计翻译并转换成源代码之后，那么算法效率就会反映为程序的执行速度和存储容量的要求。

转换过程中的指导原则是：

（1）在编程序前，尽可能化简有关的算术表达式和逻辑表达式。

（2）仔细检查算法中的嵌套的循环，尽可能将某些语句或表达式移到循环外面。

（3）尽量避免使用多维数组。

（4）尽量避免使用指针和复杂的表。

（5）不要混淆数据类型，避免在表达式中出现类型混杂。

（6）尽量采用整数算术表达式和布尔表达式。

（7）选用等效的高效率算法。

许多编译程序都具有优化的功能，它可以自动生成高效率的目标代码，可以剔除重复的表达式计算，采用循环求值法、快速的算术运算，以及采用一些能够提高目标代码运行效率的算法来提高效率。