第六章 基本程序结构

本章内容及要求：

1.了解算法概念及算法的表示，掌握用N—S流程序表示算法

2. 熟练掌握赋值语句、End语句和注释语句等语句及输入/输出消息框函数的使用；

3. 熟练掌握行if语句、块if结构、Select Case情况选择结构有使用，掌握选择的嵌套结构；

4. 熟练掌握实现循环结构的For/Next循环结构及Exit For语句、Do/Loop循环结构的使用，掌握多重循环。

重点： 选择结构及循环结构的实现及其应用

难点： 选择的嵌套及多重循环结构

6.1 算法及算法的表示

6. 1.1 算法概述

什么是算法:

广义地讲: 算法是为完成一项任务所应当遵循的一步一步的规则的、精确的、无歧义的描述，它的总步数是有限的。

狭义地讲: 算法是解决一个问题采取的方法和步骤的描述。

下面通过两个简单的例子加以说明:

例6.1 输入三个数，然后输出其中最大的数。

将三个数依次输入到变量A、B.C中，设变量MAX存放最大数。其算法如下:

1) 输入A、B.C。

2) A与B中大的一个放入MAX中。

3) 把C与MAX中大的一个放入MAX中。

4) 输出MAX，MAX即为最大数。

例6.2 输入10个数，打印输出其中最大的数。

算法设计如下:

（1）输入1个数,存入变量A中,将记录数据个数的变量N赋值为1，即N=1

（2）将A存入表示最大值的变量Max中，即Max=A

（3）再输入一个值给A，如果A>Max 则 Max=A, 否则Max不变

（4）让记录数据个数的变量增加1，即N=N+1

（5）判断N是否小于10，若成立则转到第（3）步执行，否则转到第（6）步。

（6）打印输出max

6.1.2 算法的特性

- 有穷性
- 确定性
- 有0个或多个输入
- 有一个或多个输出
- 有效性

6.1.3 算法的表示

一、自然语言与伪代码表示算法

自然语言: 就是指人们日常使用的语言，可以是汉语、英语或其它语言。

伪代码: 是用介于自然语言和计算机语言之间的文字和符号（包括数学符号）来描述算法。

例如: 例6.1可用如下的伪代码表示

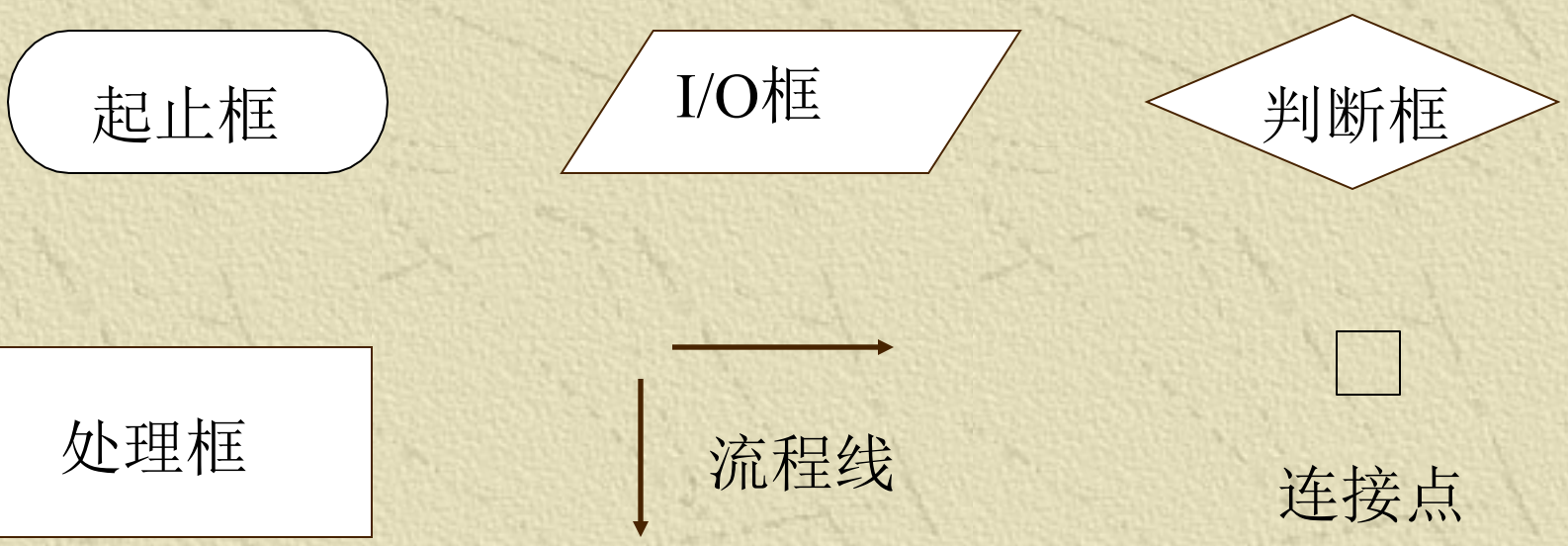
Begin（算法开始）

输入 A，B，C

IF A>B 则

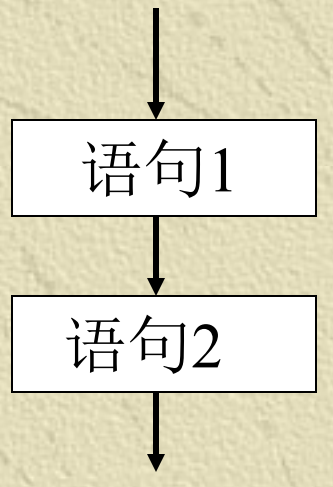
二、用传统流程图表示算法

1.传统流程图中的基本符号

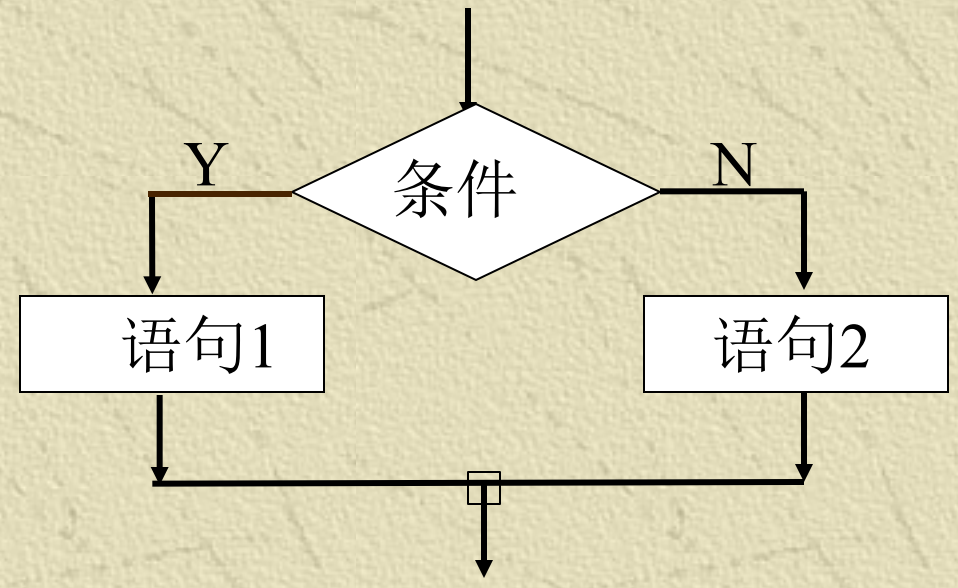


2.三种基本结构的传统表示

（1）顺序结构

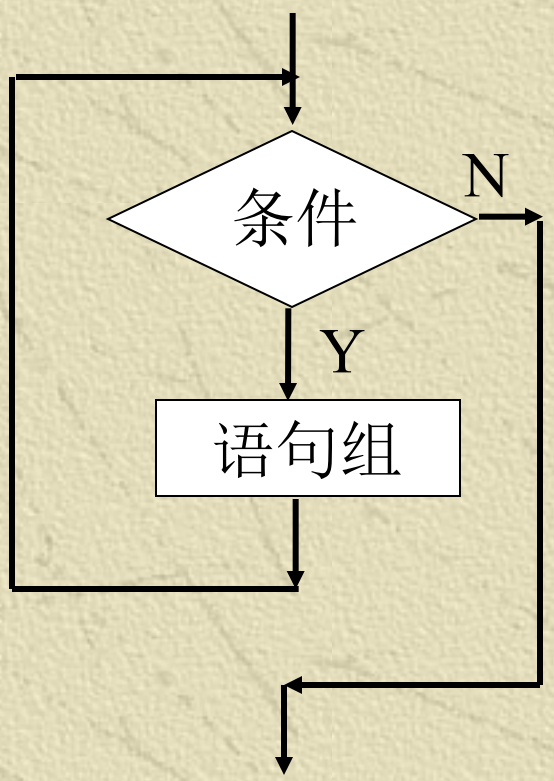


（2）选择结构



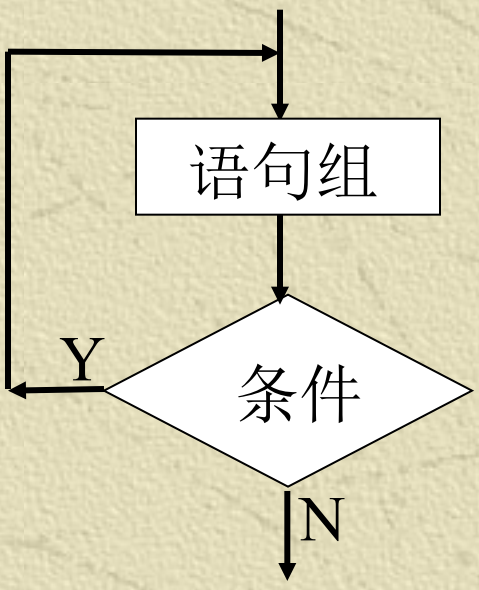
（3）循环结构

a) 当型循环



(a)

b) 直到循环



(b)

三种基本结构的特点：

（1）只有一个入口

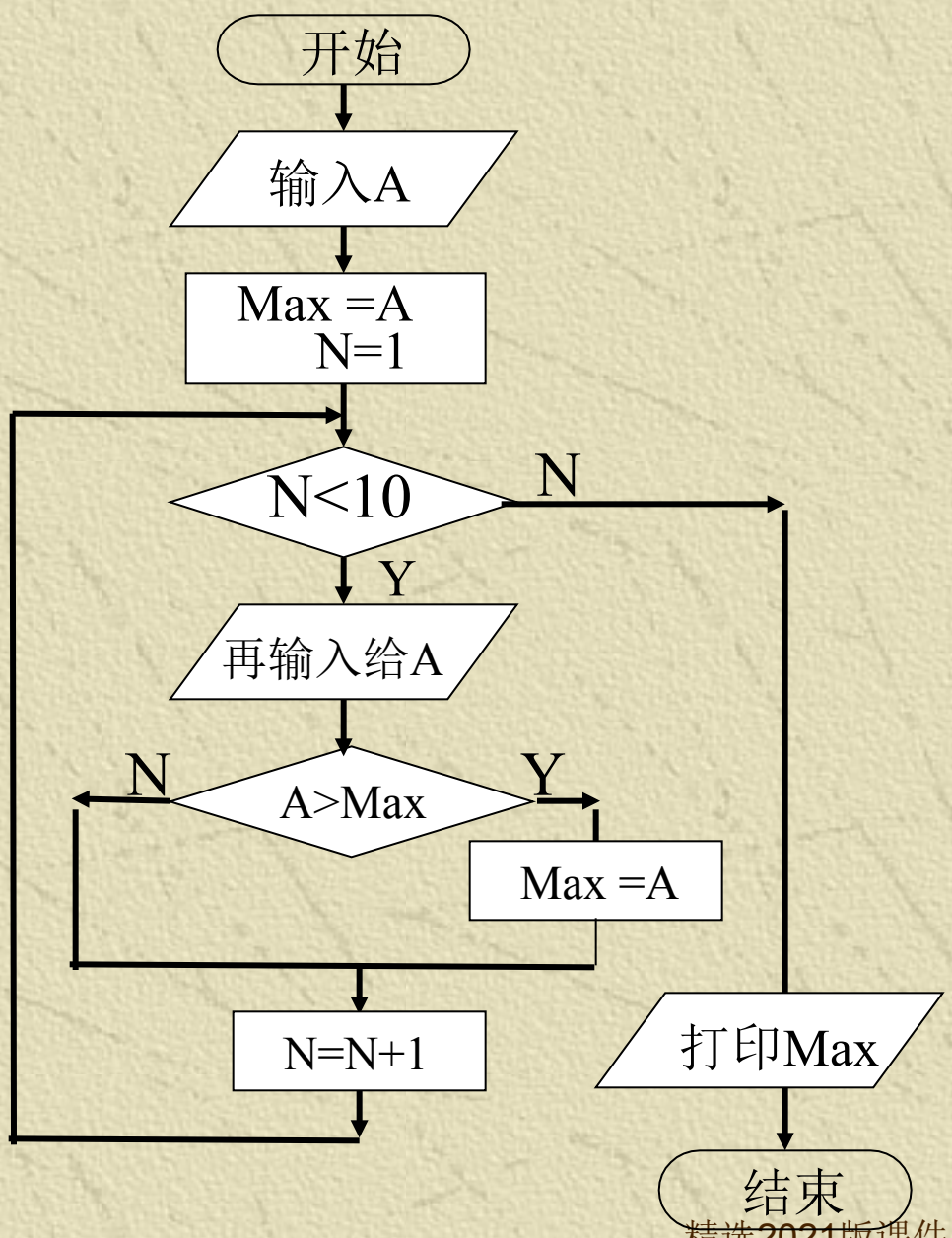
（2）只有一个出口

（3）不存在死语句

（4）不存在死循环

例：例6.2输入10个数，打印输出其中的最大的数的流程图

从10个数中选出最大的数的流程图

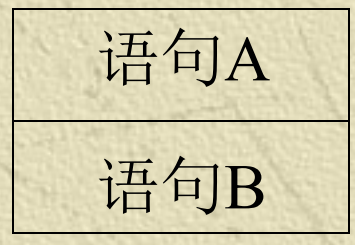


6.1.4 用N—S流程图表示算法

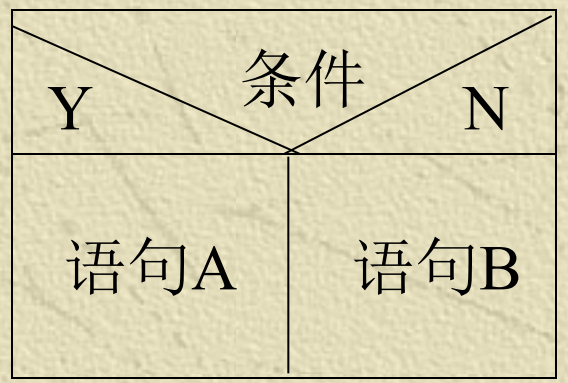
将全部算法写在一个矩形框内, 在矩形内还可包含其它从属于它的框

三种基本结构的N—S图表示:

1.顺序结构

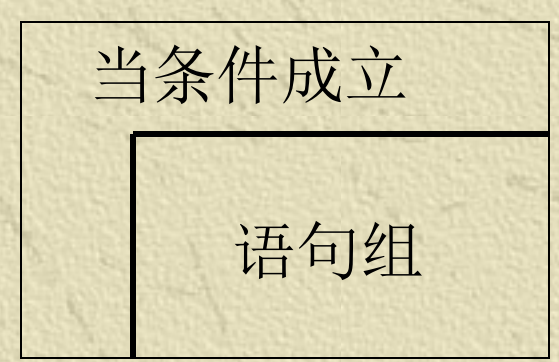


2.选择结构



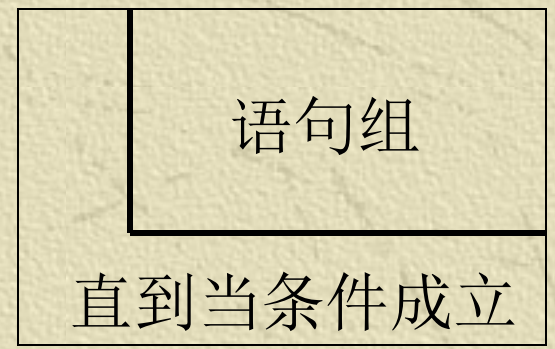
（3）循环结构

a) 当型循环



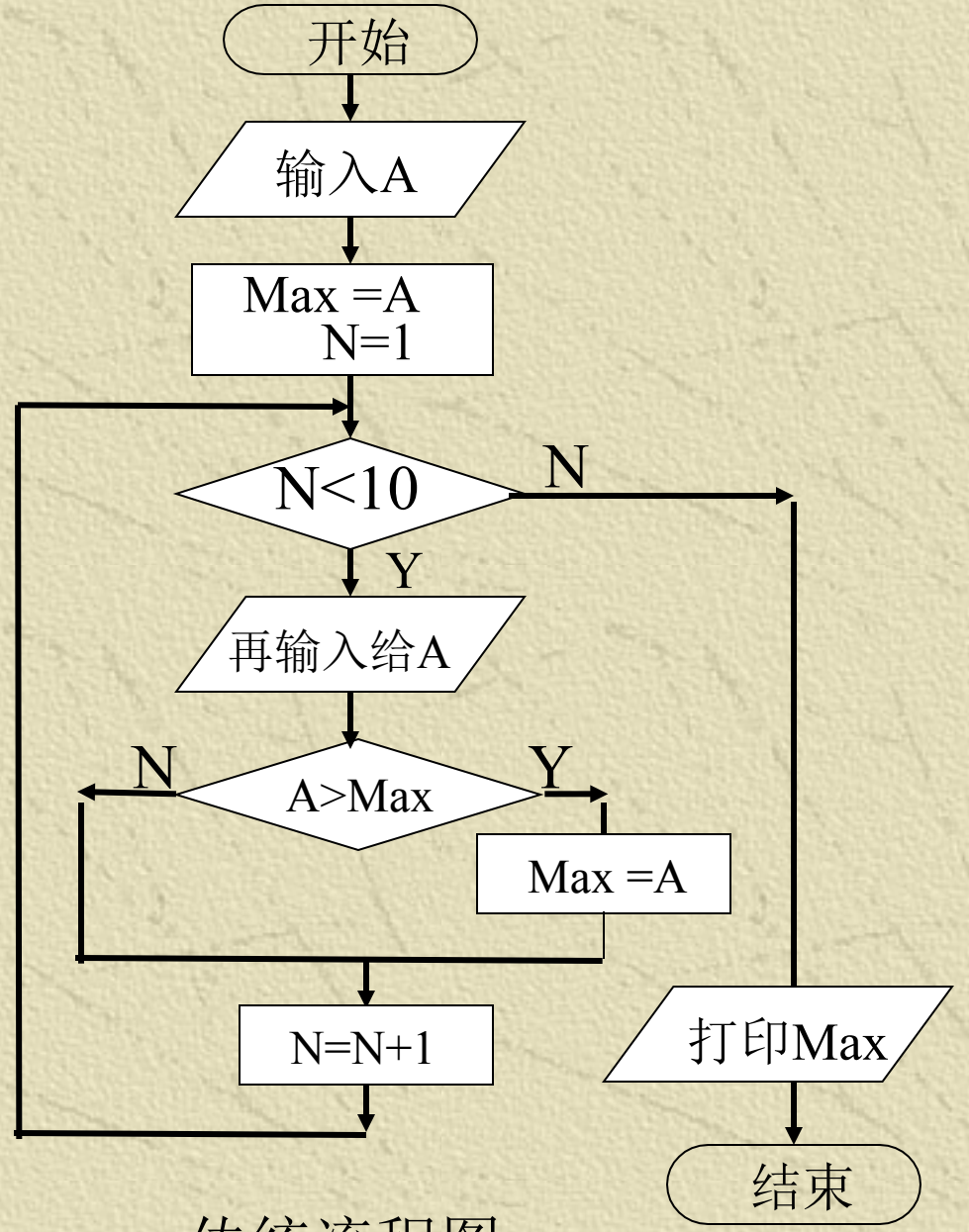
(a)

b) 直到循环

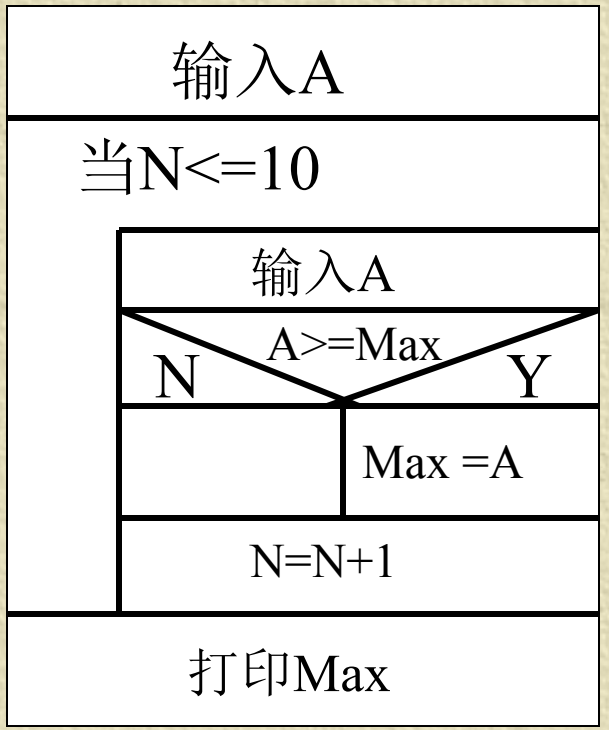


(b)

例：画出从10个数中选出最大的数的N—S 流程图


开始
输入A
Max =A
N=1
N<10
N
Y
再输入给A
N
A>Max
Y
Max =A
N=N+1
打印Max
结束

传统流程图


输入A
当N<=10
输入A
A>=Max
N
Y
Max =A
N=N+1
打印Max

N—S流程图

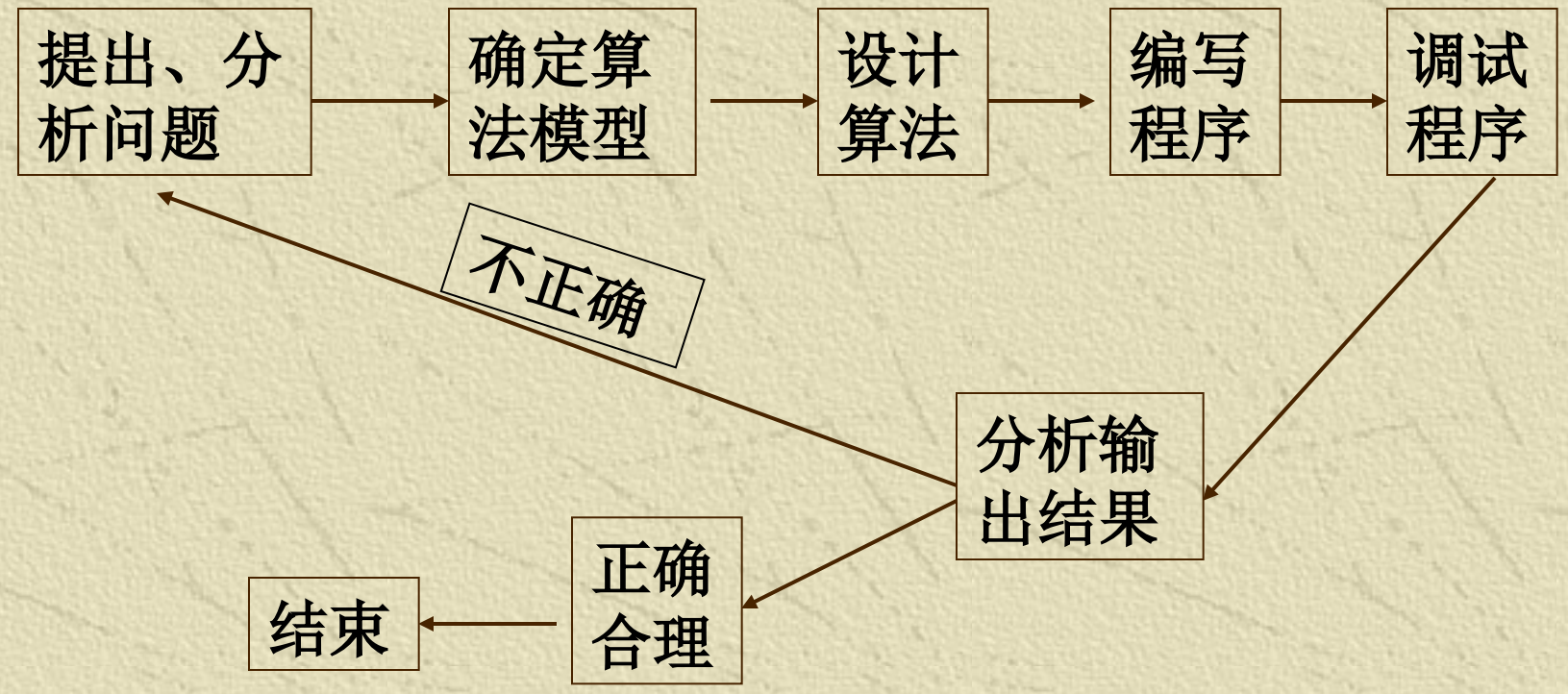
最后需要说明的是: 上面介绍的算法表示是给人看的，即是为帮助程序开发人员阅读、编写程序而设计的一种辅助工具，程序则必须符一计算机语言的语法规则。

下面是例6.2的计算机 程序，即为用计算机语言表示算法:

```
Private Sub Form_Click()
    Dim a%, max%, i%
    max = a
    For i = 1 To 10
        a = Val(InputBox("A=?"))
        If a > max Then max = a
     Next i
     Print "Max="; max
End Sub
```

6.1.5 结构化程序设计方法（补充）

（一）用计算机解决问题的过程



（二）结构化程序设计思想

自顶向下、逐步细化、模块化

自顶向下: 先从全局、整体设计

逐步细化: 将一个问题分解成几个较小的问题解决

模块化:　将一个大任务分解成若干个较小的部分，每个部分承担一定功能，称为“功能模块”

例: 给100个整数，打印输出其中的素数

输入100个数存入 X_1，x_2,……x_{100}

S1 —— *S1细化* →

i=1

当i<=100

输入x_i

i=i+1

让x_1，x_2,……x_{100}中的非素变为0

S2

打印x_1…..x_{100}中不等于0的数

S3 —— *S3细化* →

i=1

当i<=100

N　$x_i \neq 0$　Y

打印x_i

i=i+1

N—S流程图

输入100个数存入 X_1，x_2,……x_{100}

S1

让x_1，……x_{100}中的非素变为0

S2

S2细化

i=1

当i<=100

判断x_i是否是素数，若不是则将x_i=0

i=i+1

S21

S21细化

j=2

r<=将x_1除j的余

r=0

N　　Y

x_i=0

j=j+1

直到j>x_i/2

打印x_1…..x_{100}中不等于0的数

S3

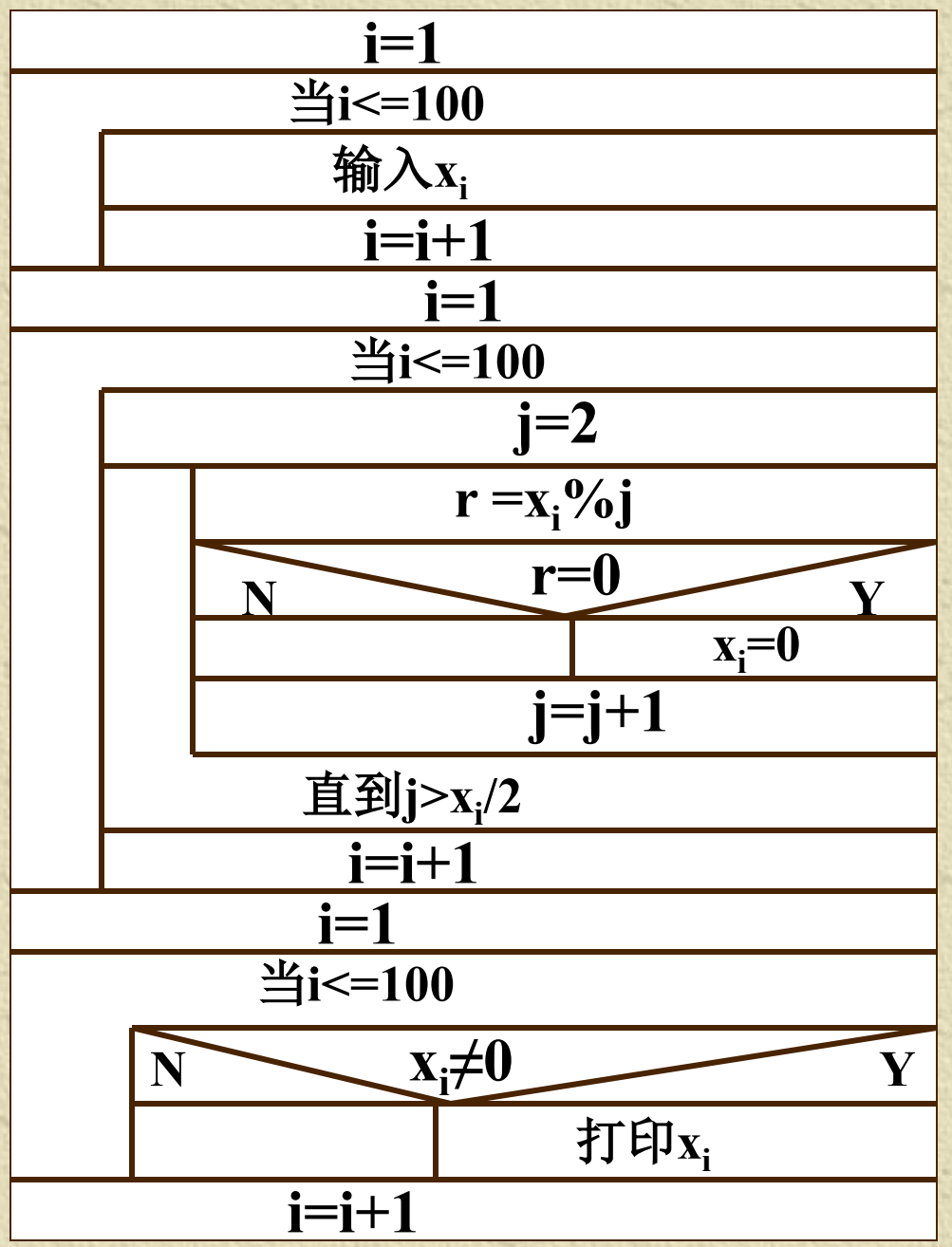
N—S流程图

输入100个数存入
X_1，x_2,……x_{100}

让x_1，……x_{100}中的非素变为0

打印x_1…..x_{100}中不等于0的数

细化后的流程图



6.2 顺序结构

6.2.1 赋值语句

形式：　变量名＝表达式

对象.属性＝表达式

功能：　将表达式的值赋值给变量名或指定对象的属性。

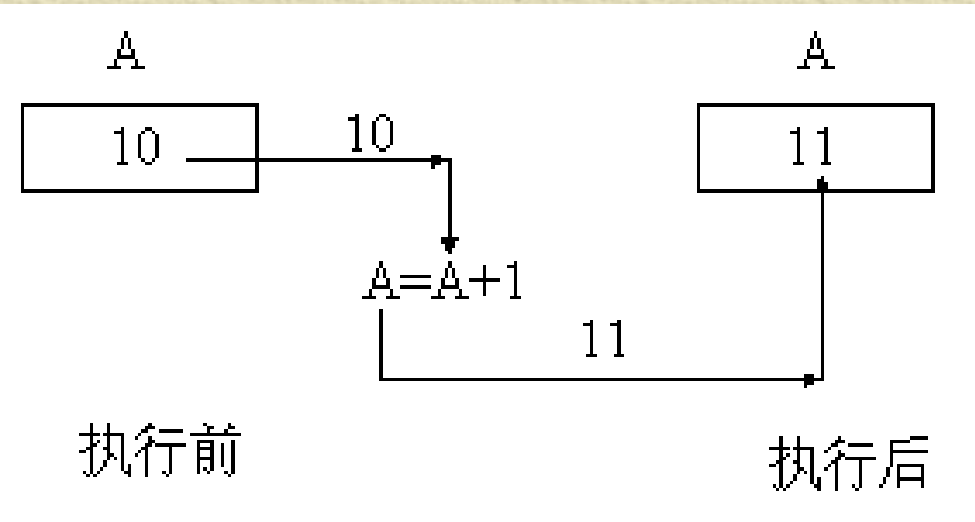
一般用于给变量赋值或对控件设定属性值.

例：　sRate!=0.1

Text1.Text ＝ "欢迎使用Visual Basic 6.0”

说明：

执行过程如右图



4．赋值符号“=”左边一定只能是变量名或对象的属性引用，不能是常量、符号常量、表达式。

下面的赋值语句都是错的:

5=X　　' 左边是常量。

Abs(X)=20　　' 左边是函数调用，即是表达式。

5.赋值符号“=”两边的数据类型一般要求应一致。

具体规则（P77）

6.2.2 注释语句

其语法格式为:

Rem <注释内容>

或 ' <注释内容>

说明:

1. <注释内容> 指要包括的任何注释文本。在Rem关键字和注释内容之间要加一个空格。可以用一个英文单引号“'”来代替Rem关键字。

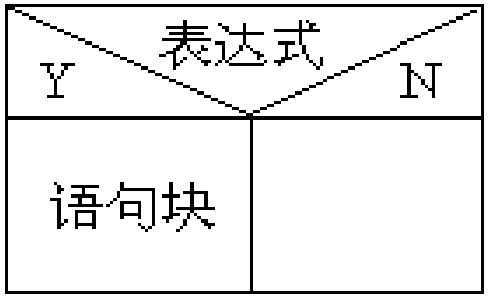
2. 如果在其他语句行后面使用Rem关键字，必需用冒号(:)与语句隔开。若用英文单引号“'”，则在其他语句行后面不必加冒号(:)。

例如:

```
Const PI=3.1415925      ' 符号常量PI
S=PI*r*r                : Rem计算圆的面积
```

6.3 选择结构

1.If…Then语句(单分支结构)



If <表达式> Then

语句块

End If

或 If <表达式> Then <语句>

例:已知两个数x和y,比较它们的大小,

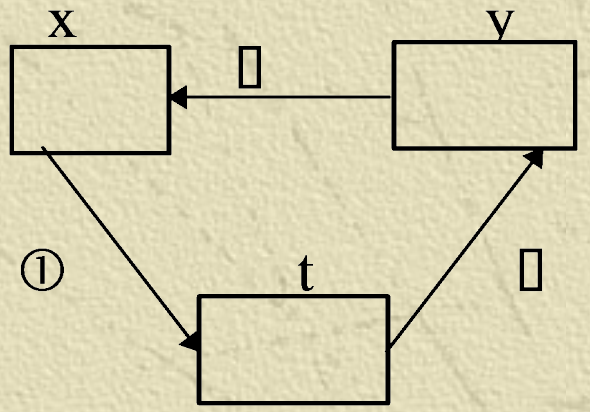
使得x大于y.

If x<y Then

t=x : x=y: y=t

End If

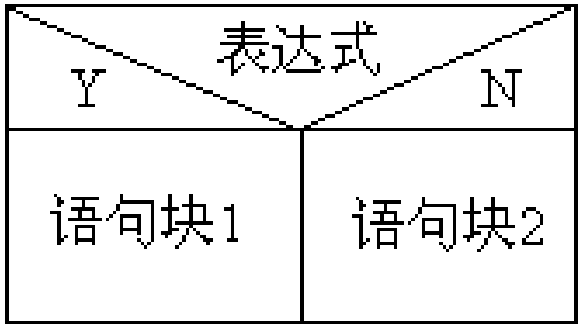
或 If x<y Then t=x: x=y: y=t



2.If…Then…Else语句(双分支结构)

```
If <表达式> Then
   <语句块1>
Else
   <语句块2>
End If
```

If <表达式> Then <语句1> Else <语句2>



双分支选择结构执行过程

例如: 输出x,y两个中值较大的一个值。

```
IF X>Y Then
   Print X
Else
   Print Y
End If
```

也可以写成如下的单行形式:

```
IF X>Y Then  Print  X  Else  Print  Y
```

3.If…Then…ElseIf语句(多分支结构)

形式:

If <表达式1> Then

<语句块1>

Else If <表达式2>Th

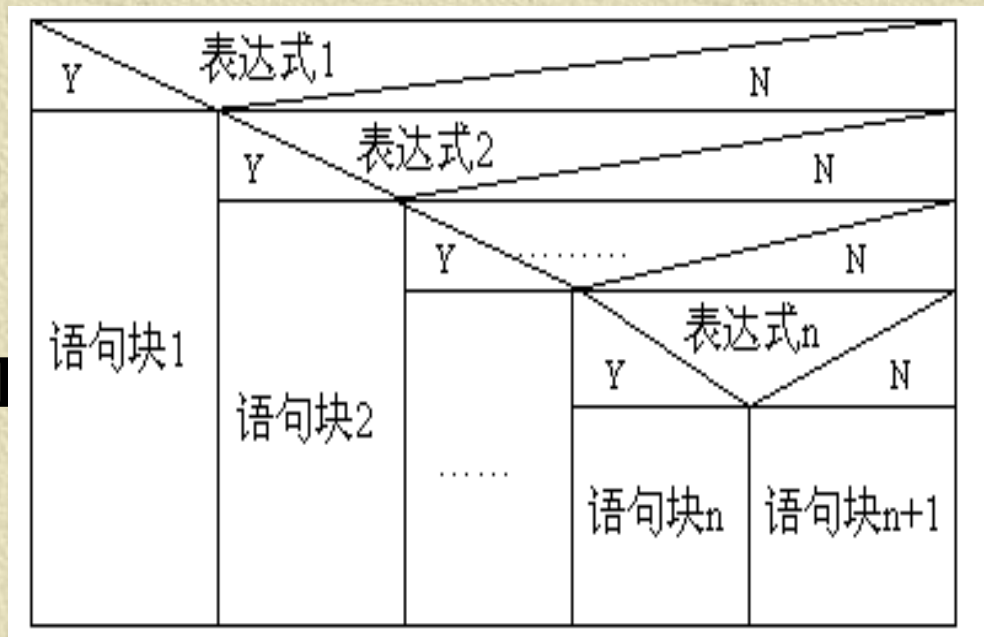
<语句块2>

…

[Else

语句块 n+1]

End If



执行过程

例: 输入一学生成绩，评定其等级。方法是: 90~100分为“优秀”，80~89分为“良好”，70~79分为“中等”，60~69分为“及格”，60分以为“不合格”

使用IF语句实现的程序段如下:

```
If  x>=90 then
        Print "优秀"
ElseIf  x>=80 Then
        Print "良好"
ElseIf  x>=70  Then
        Print  "中等"
ElseIf  x>=60 Then
        Print "及格"
Else
        Print "不及格"
End If
```

6.3.2 Select Case语句（条件分支）

形式:

数值型或字符串表达式

```
Select   Case 变量或表达式
    Case 表达式列表1
            语句块1
    Case 表达式列表2
            语句块2
            …
    [Case Else
            语句块n+1]
End Select
```

求表达式的值			
表达式列表1	表达式列表2		Case Else
语句块1	语句块2	……	语句块n+1

<表达式列表>: 与<变量或表达式>同类型的下面四种形式之一:

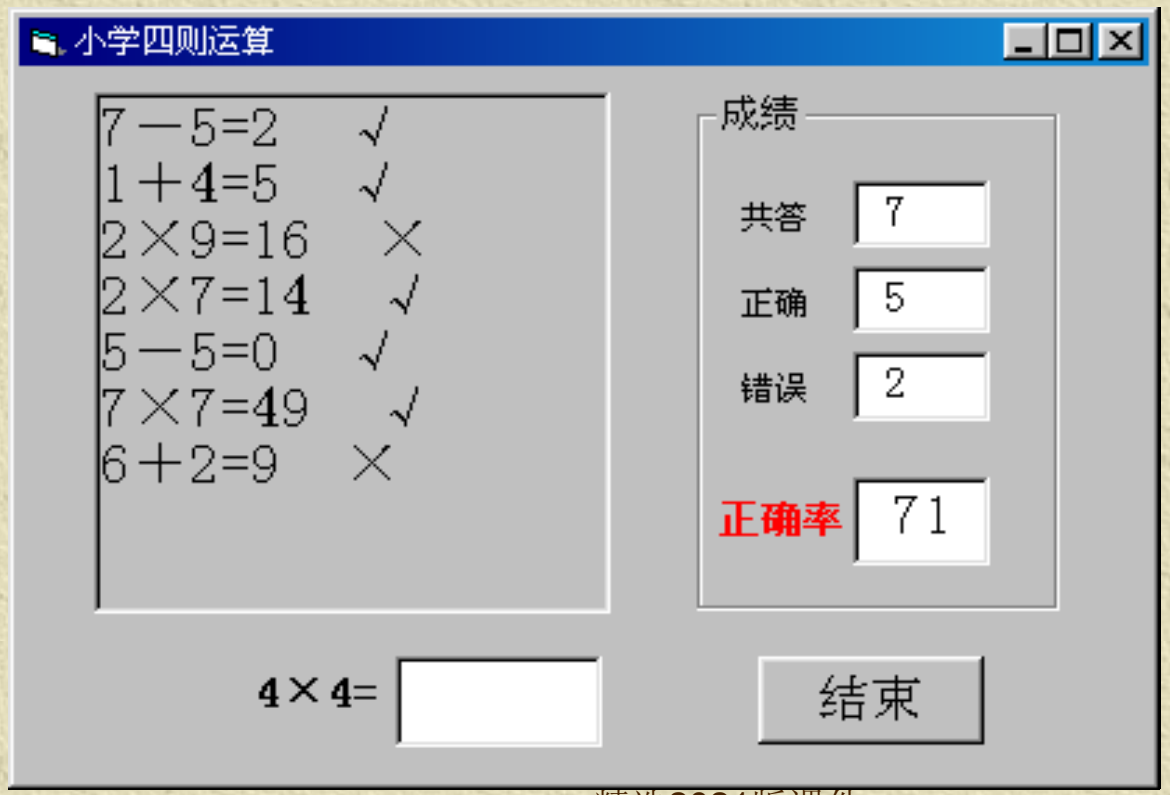
表达式

将例6.3 使用select case…..语句来实现的程序段如下:

```
Select Case x
   Case 90 to 100
      Print "优秀"
   Case 80 to 89
      Print "良好"
   Case 70 to 79
      Print "中等"
   Case 60 to 69
      Print "及格"
   Case Else
      Print "不及格"
End Select
```

补充例题: 设计一个由计算机来当小学低年级学生算术老师的VB应用程序，要求给出一系列的两个1~10数的四则运算的算术题，学生输入该题的答案，计算机根据学生的答案判断正确与否，并给出成绩，单击结束命令按钮，退出应用程序。

分析:产生1～10操作数，可通过Int(10* Rnd + 1) 实现



6.3.3 选择结构的嵌套

在IF语句的Then分支和Else分支中可以完整地嵌套另一IF语句或Select Case语句，同样Select Case语句每一个Case分支中都可嵌套另一IF语句或另一Select Case语句。下面是两种正确的嵌套形式:

（1）
```
IF <条件1> Then
  …..
  if  <条件2> Then
    …….
  Else
    …..
 End If
 ….
Else
 ….
 IF <条件3>  Then
  …..
```

（2）
```
IF <条件1> Then
 …..
Select Case …
   Case  ……
      IF <条件1>  Then
       …….
       Else
        …..
       End If
       ……
    Case….
  …..
 End Select
 ….
End IF
```

注意:

只要在一个分支内嵌套，不出现交叉，满足结构规则，其嵌套的形式将有很多种，嵌套层次也可以任意多。

对于多层IF嵌套结构中，要特别注意IF与Else的配对关系，一个Else必须与IF配结，配对的原则是: 在写含有多层嵌套的程序时，建议使用缩进对齐方式，这样容易阅读和维护。