程序设计基础

算法

问题：

计算长方形的面积

1. 接收用户输入的长方形长度和宽度两个值；
2. 判断长度和宽度的值是否大于零；
3. 如果大于零，将长度和宽度两个值相乘得到面积，
   否则显示输入错误；
4. 显示面积。

算法

算法：解决问题的具体方法和步骤

算法及其表示

程序= 数据结构+ 算法+程序设计方法+语言工具

1. 算法是解决某一问题所采取的一系列步骤，它具有如下特点：

(1) 有输入：算法可以有零个或多个输入。输入是用来在一个算法的执行过程中，向它提供处理对象(数据)或控制算法执行过程的信息。

(2) 有输出：算法必须具有一个或多个输出。它是算法执行结果的输出。没有输出的算法是一个无效的算法。

算法及其表示

(3) 有穷性：任何算法都应该在执行有穷步骤之后结束。

(4) 确定性：算法不能具有二义性。也就是说，算法中每一步的语义都应该是清晰明了的，明确指出应该执行什么操作，如何执行操作。

(5) 高效性：根据算法编写出来的程序应具有较高的执行效率。其主要含义有：一是执行时间短，二是不占用过多内存。

2 算法的表示

- 自然语言表示：易理解和交流，容易产生二义性。
- 伪码表示：用形式化语言描述算法的逻辑过程。
- 程序流程图：用图形符号＋文字说明来表示数据处理过程和步骤。
- N-S流程图：也称方框图。

算法的表示示例

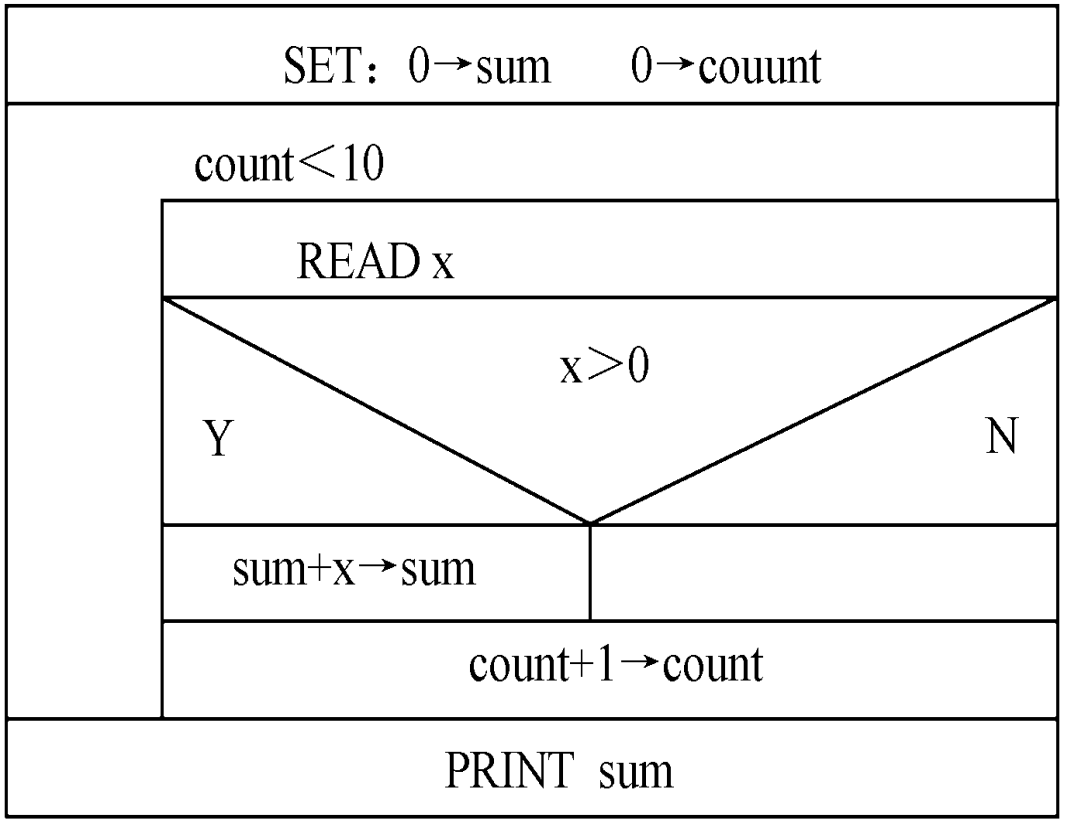
1 .伪码表示：从键盘读入10个正数并输出它们的和。

```
BEGIN
  SET 0→sum
  SET 0→count
  WHILE count < 10
    BEGIN
       READ a integer data to x from keyboard
       IF x>0
          sum + x →sum
       count + 1→count
    END
  END_WHILE
  PRINT sum

END
```

算法的表示示例

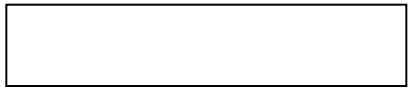
2. N-S流程图表示：从键盘读入10个正数并输出它们的和。



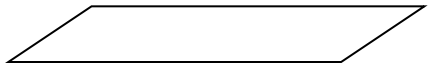
流程图

- 流程图是算法的一种图形化表示方式。
- 流程图直观、清晰，更有利于人们设计与理解算法。
- 它使用一组预定义的符号来说明如何执行特定任务。

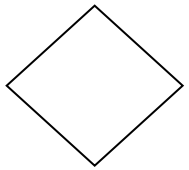
开始/结束



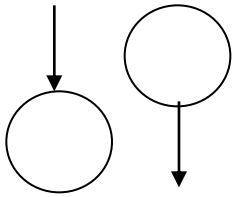
处理



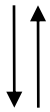
输入/输出



判断/分支

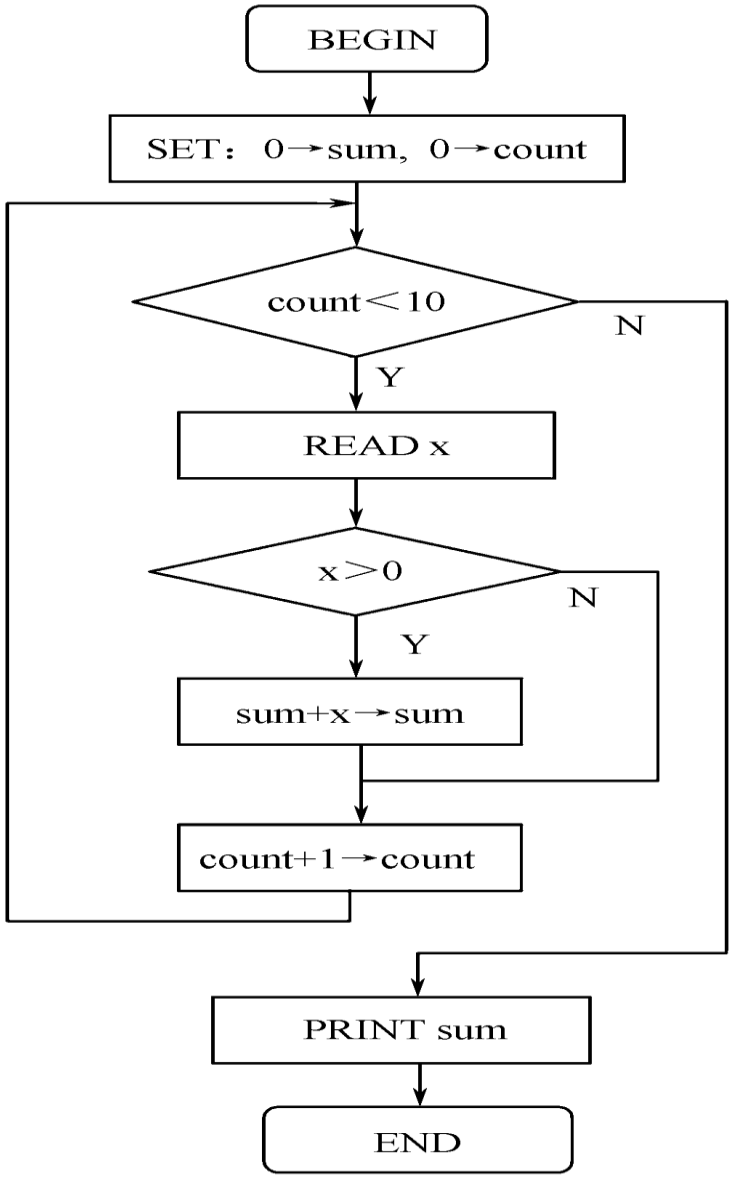


连接符



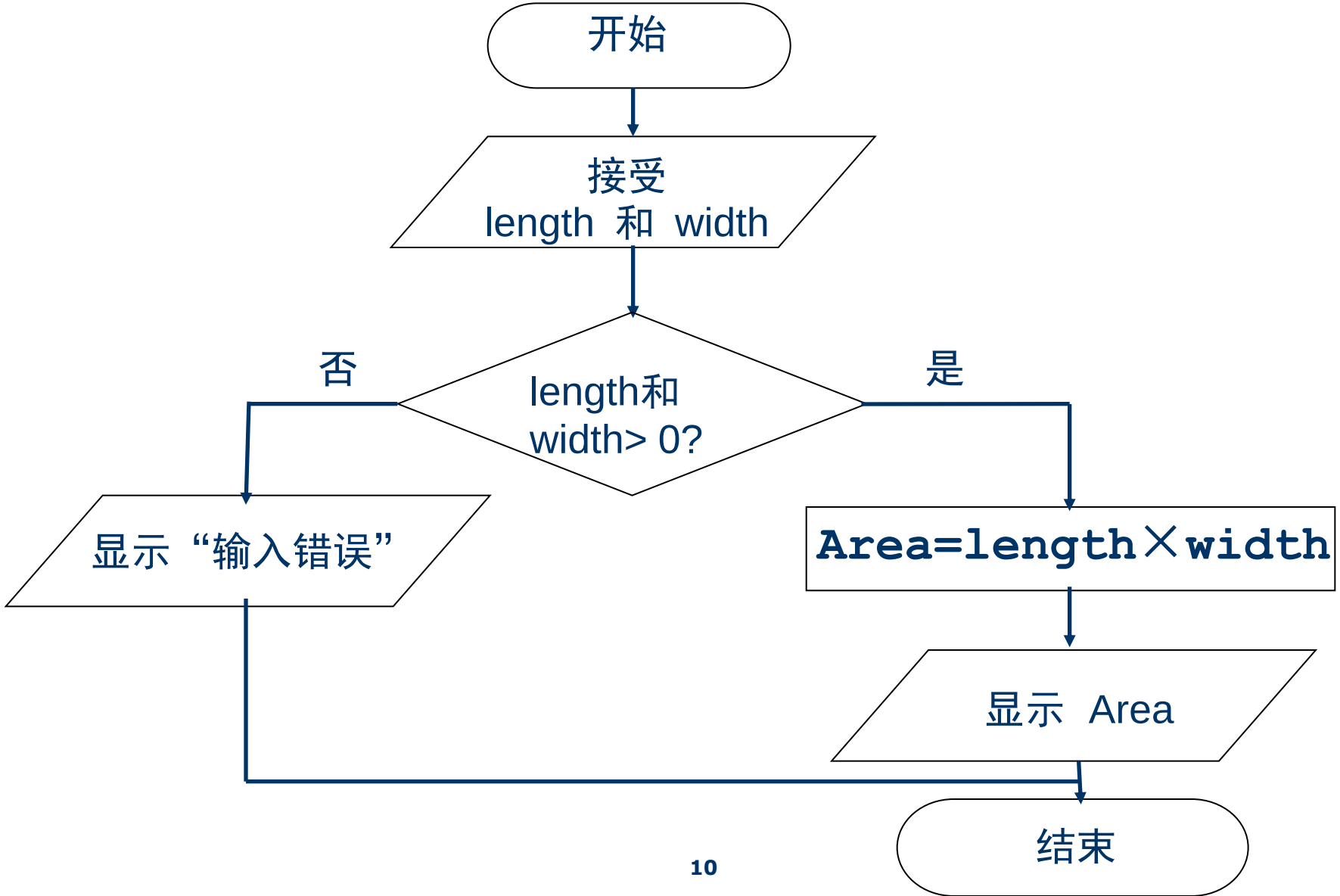
流程线

算法的表示示例



3. 程序流程图描述从键盘读入10个正数并输出它们的和。

流程图---求长方形面积


开始
接受
length 和 width
length和
width> 0?
否
是
显示“输入错误”
Area=length×width
显示 Area
结束

逻辑思维与计算机解题

1 C程序的语句

2 C语言的三种基本结构

3 选择结构程序设计

4 循环结构程序设计

目标

- 将实际问题抽象为逻辑关系
- 枚举法解题思路
- 关系与表达式
- 程序的循环结构与分支结构

问　题

有4位同学中的一位做了好事，不留名，表扬信来了以后，校长问这4位同学是谁做的好事。

- A说：“不是我”
- B说：“是C”
- C说：“是D”
- D说：“他胡说”

已知3人说的是真话，一个人说的是假话。如何找出做好事的同学。

解决问题所需工具

- **如何描述语句关系**
  - 关系运算符与关系表达式
  - 逻辑运算符与逻辑表达式
- **如何用程序设计的方法找到答案**
  - 判定结果是否正确——选择结构
  - 如依次对假定情况进行排解——循环结构

关系运算符和关系表达式

1．关系运算符

关系运算符用于比较两个运算对象的大小。C语言提供的关系运算符，如下所示：

< ，小于运算符
<= ，小于等于运算符
> ，大于运算符
>= ，大于等于运算符
== ，等于运算符
!= ，不等于运算符

关系运算符和关系表达式

2．关系表达式

关系表达式就是用关系运算符将两个或两个以上运算对象连接起来的式子。其中运算对象可以是常量、变量或表达式。

关系表达式的运算结果有两种：“真”或“假”。在C语言中用1表示“真”，用0表示“假”。

逻辑运算符和逻辑表达式

1．逻辑运算符

逻辑运算符用来进行逻辑运算。C语言提供了3个逻辑运算符，分别是：

!：逻辑非

- 如：!a ，若a为真，则!a为假，否则!a为真

&&：逻辑与

- 如：a&&b，若a，b均为真，则a&&b为真，否则a&&b为假

||：逻辑或

- 如：a||b，若a，b均为假，则a||b为假，否则a||b为真

问　题

- A说：“不是我”
- B说：“是C”
- C说：“是D”
- D说：“他胡说”　　char thisman=‘ ’;

说话人	说的话	写成关系表达式
A	不是我	
B	是C	
C	是D	
D	他胡说	

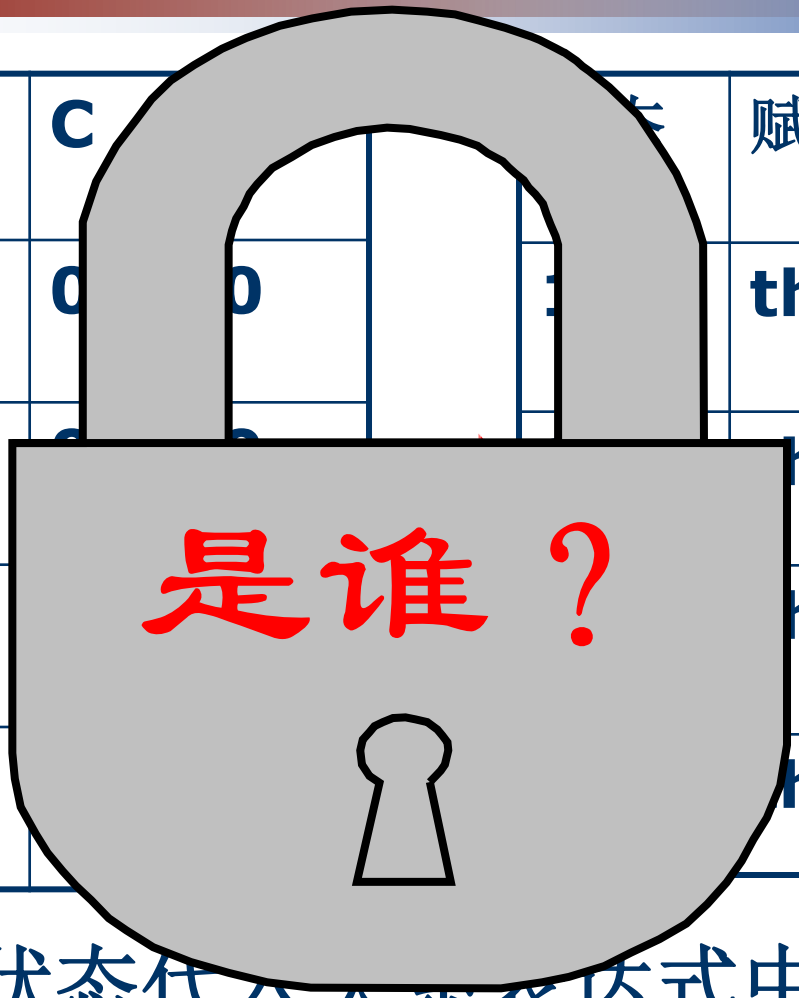
关系描述

- char thisman=‘ ’; //初始化为空

说话人	说的话	写成关系表达式
A	不是我	thisman!=‘A’
B	是C	thisman==‘C’
C	是D	thisman==‘D’
D	他胡说	thisman!=‘D’

枚举法过程分析

状态	A	B	C				赋值表达式
1	1	0	0	0		1	thisman=‘A’
2	0	1	[illegible]	[illegible]		[illegible]	hisman=‘B’
3	0	0	[illegible]	[illegible]		[illegible]	hisman=‘C’
4	0	0	[illegible]	[illegible]		[illegible]	hisman=‘D’



分别把四种状态代入关系表达式中分析结果。

假定让thisman='C'代入4句话中：

说话人	说的话	写成关系表达式	值
A	不是我	'C'!='A'	1
B	是C	'C'=='C'	1
C	是D	'C'=='D'	0
D	他胡说	'C'!='D'	1

满足三个表达式结果为真即为答案

解决问题描述中的两个问题

- 如何判断假定情况是否为真？
- 如何描述分别假定ABCD四位同学为做好事的人，尝试推断假定结果是否成立？

3.2 C语言的三种基本结构

顺序结构

- 程序的执行流程按照程序中语句的先后，从第一条语句开始，顺序地执行每一条语句。

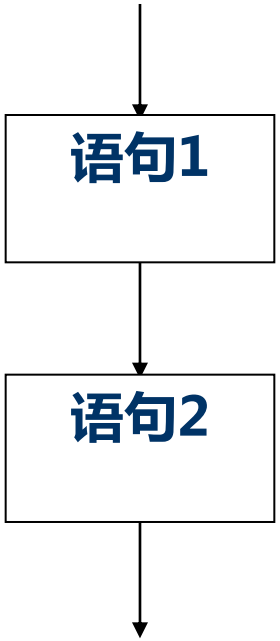
选择结构

- 对给定的条件进行判断，并根据逻辑结果选择执行语句。由选择结构可以派生出另一种基本结构，多路选择控制结构。

循环结构

- 根据特定的条件决定某个程序段重复执行。

顺序结构



先执行语句1，再执行语句2，两者是顺序执行的关系。

图3-1 顺序结构示意图

选择结构2-1

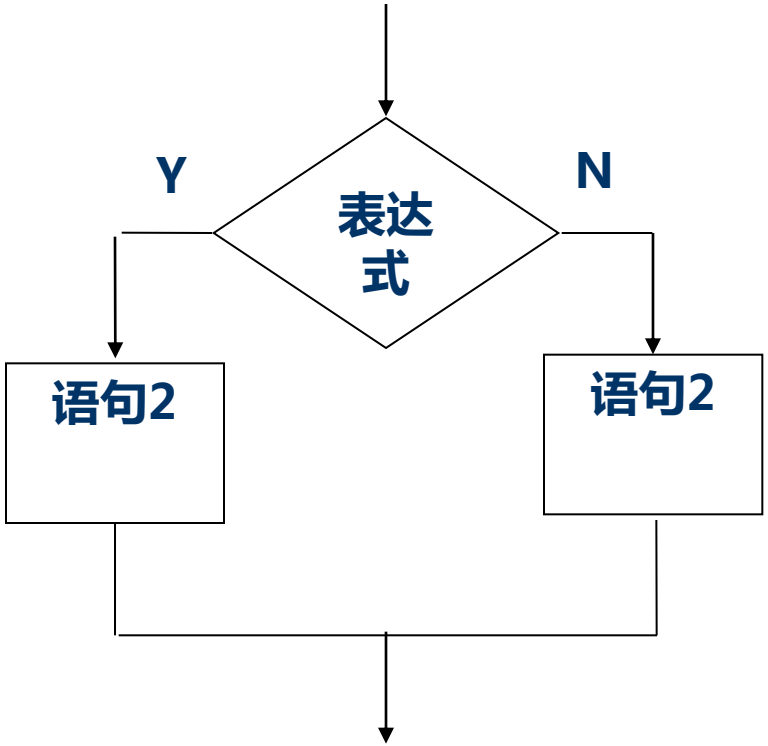


图3-2　选择结构示意图

先计算表达式的值，当表达式的值为“真”时，
执行语句1；否则，执行语句2。

选择结构2-2

根据表达式的值（k1，k2，…，kn）不同而分别执行语句1，语句2，…，语句n。

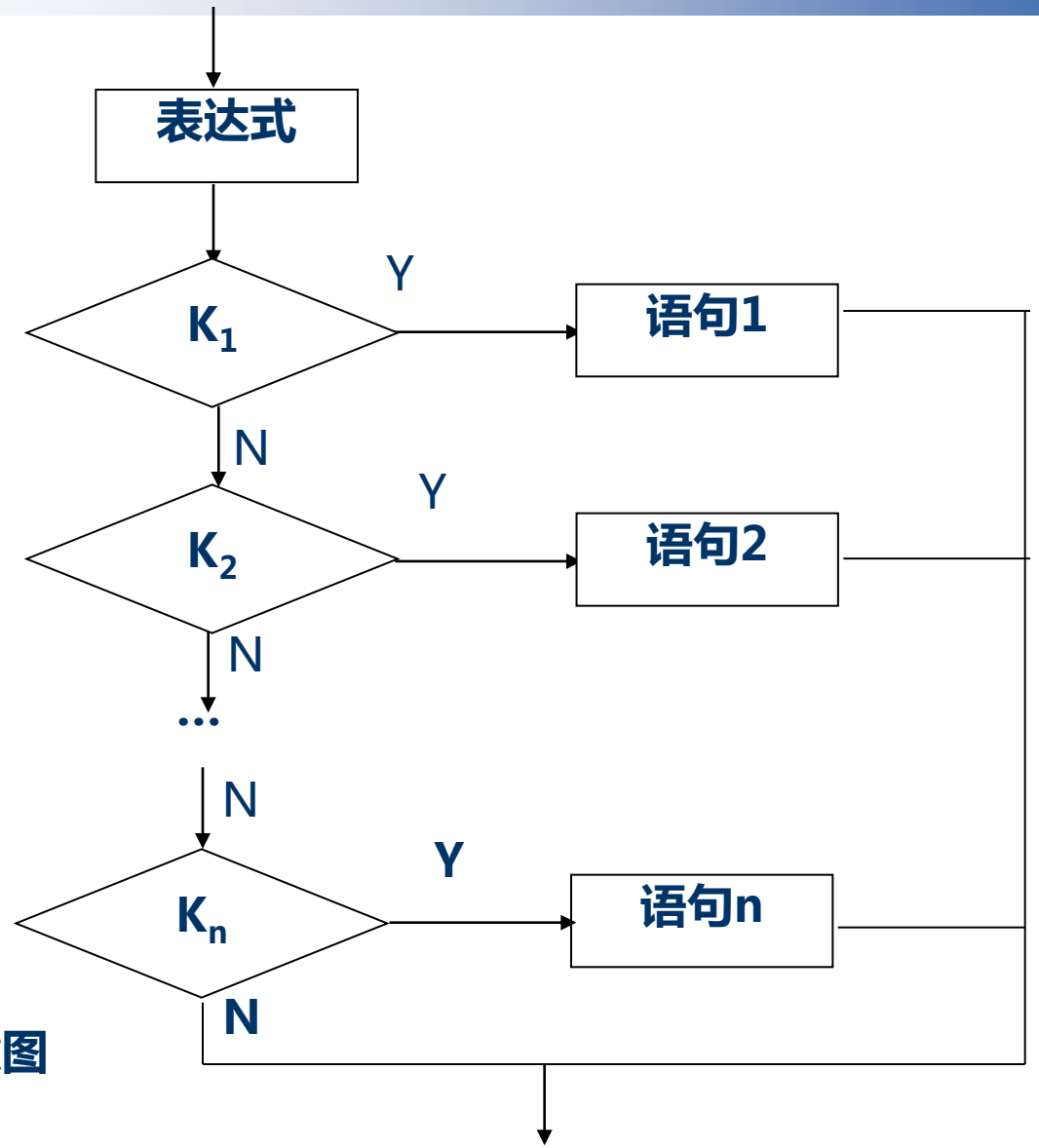


图 3-3 多路选择结构示意图

循环结构

先计算表达式的值，当表达式的值为“真”时，重复执行语句，当表达式的值为“假”时，跳出循环。所以，语句可能一次也不执行。

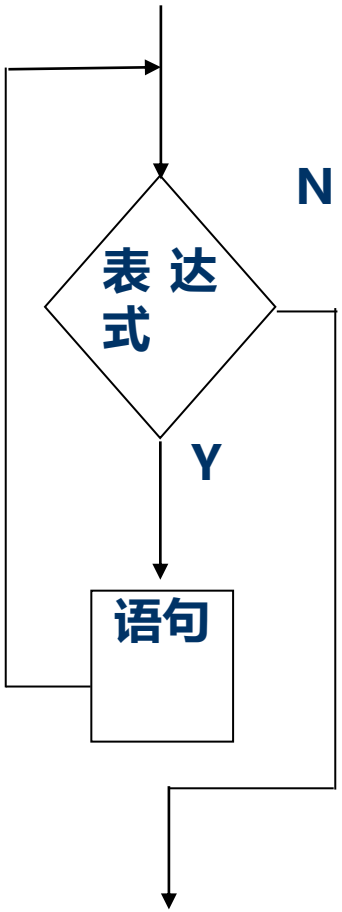


图3-4 “先判断后执行”结构示意图示意图

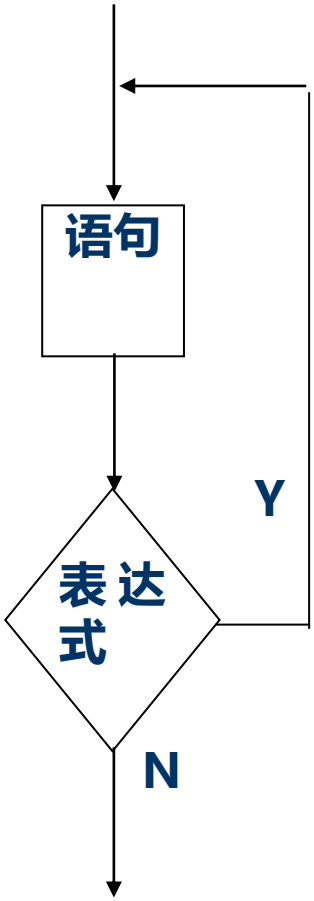


图3-5 “先执行后判断”结构

先执行语句，然后再计算表达式的值，当表达式的值为“真”时，重复执行语句，直到表达式的值为“假”时，跳出循环。循环体至少被执行一次。

3.3 选择结构程序设计

if 语句（条件语句）

- 简单分支语句
- 双重分支语句
- if 语句的嵌套

switch语句

- switch语句

条件结构1

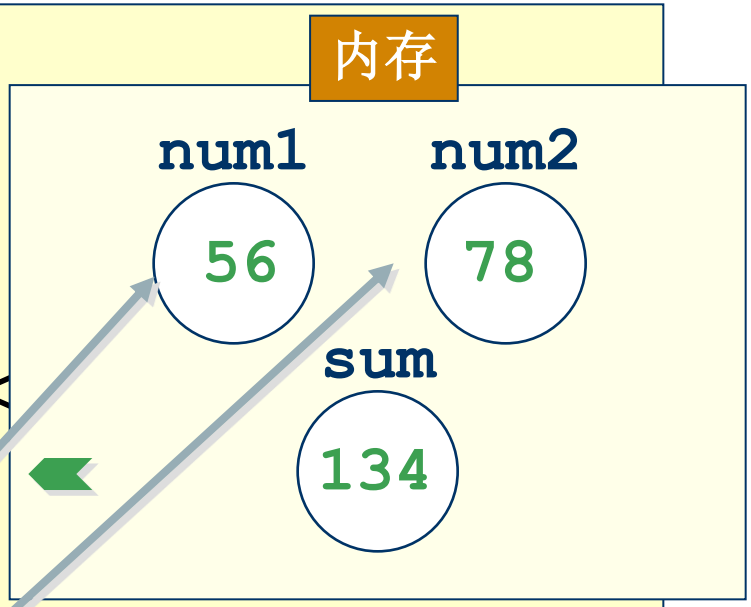
- 条件结构就是需要根据不同条件进行判断，然后执行不同的操作
- 简单 if 语句的一般形式为：

```
if (<条件>)
    <语句块>
```

如果条件为真，if 语句执行一个语句或一组语句；
如果条件为假，则执行 if 语句后面的语句（如果有）。

简单if语句的示例

```
# include <iostream.h>
void main()
{
  int num1, num2, sum;
  cout<<"\n请输入两个数："<<
  cin>> num1>>num2;
  sum = num1+ num2 ;
  if(sum > 100)
        cout<<"\n两数的和大于 100 \n"<<endl;
}
```




```
请输入两个数：56 78
两数的和大于 100
```

条件结构 2

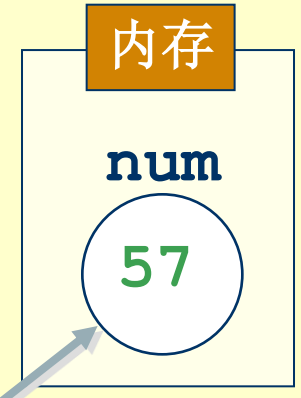
- **if-else 语句的一般形式为：**

```
if (<条件>)
{
   <语句块>
}
else
{
   <语句块>
}
```

如果条件为真，执行 **if** 后面的一个语句或一组语句；
如果条件为假，则执行 **else** 后面的语句。

if-else语句(判断输入整数是否为偶数)

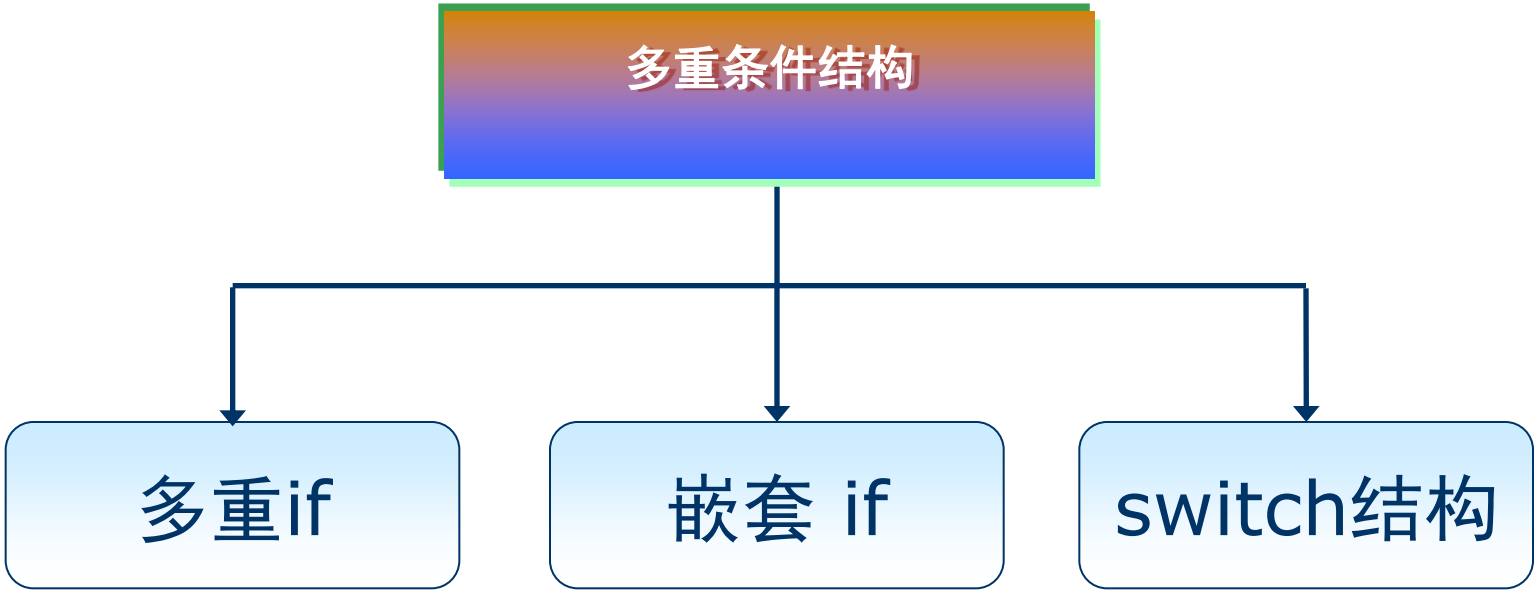
```
#include <iostream.h>
void main()
{
        int num;
        cout<<"请输入一个整数：" <<endl;
         cin>>num;
        if((num % 2) == 0)
                cout<<num<<" 是一个偶数"<<endl;
         else
                cout<<num<<" 是一个奇数"<<endl;
}
```




```
请输入一个整数：57
57 是一个奇数。
```

多重条件结构

- 要处理多重条件判断的情况，需要使用多重条件结构



嵌套 if 结构 2-1

- 嵌套 if 结构就是将整个 if 块插入另一个 if 块中

```
if (表达式1)
    if (表达式2)
        if (表达式3)
            ✓ 语句;
else
    ✓ 语句;
```

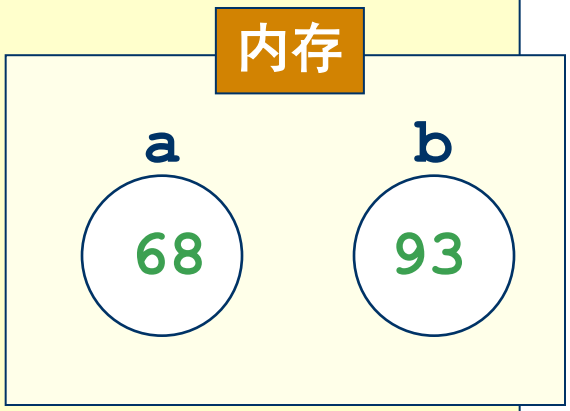
真 假

真

真

嵌套 if 结构的示例

```
#include<stdio.h>
void main()
{
        int a,b;
        cout<<" 请输入 A 和 B 的值: ";
        cin>>a>>b;
        if(a!=b)
                {if(a>b)
                        cout<<" A>B"<<endl;
                else
                        cout<<" A<B"<<endl;}
        else
                cout<<" A=B"<<endl;
}
```



输出:

A<B

嵌套 if 结构 2-2

```
if (x > 0)
    if (y > 1)
        z = 1;
    else /*这个 else 部分属于哪个 if? */
        z = 2;
```

C语言规定，每个 else 部分总属于前面最近的那个缺少对应的 else 部分的 if 语句。

提倡使用大括号括起来以避免看起来有二义性。

多重 if 结构

问题描述：

编写一个程序，根据用户输入的期末考试成绩，输出相应的成绩评定信息。成绩大于等于90分输出“优”；成绩大于等于80分小于90分输出“良”；成绩大于等于60分小于80分输出“中”；成绩小于60分输出“差”。

多重 if 结构

- 多重 if 结构是 if-else 的一种形式，这种形式也称为阶梯式 if-else-if

if (表达式1) 真 假
✓ 语句1;

else if (表达式2) 真 假
✓ 语句2;

else if (表达式3) 真 假
✓ 语句3;

.

.

else
✓ 语句n;

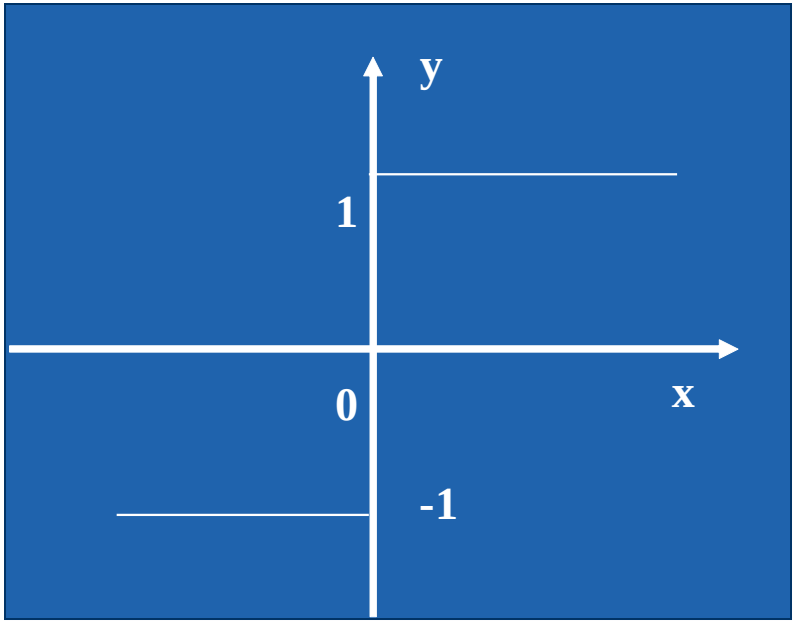
多重 if 结构示例

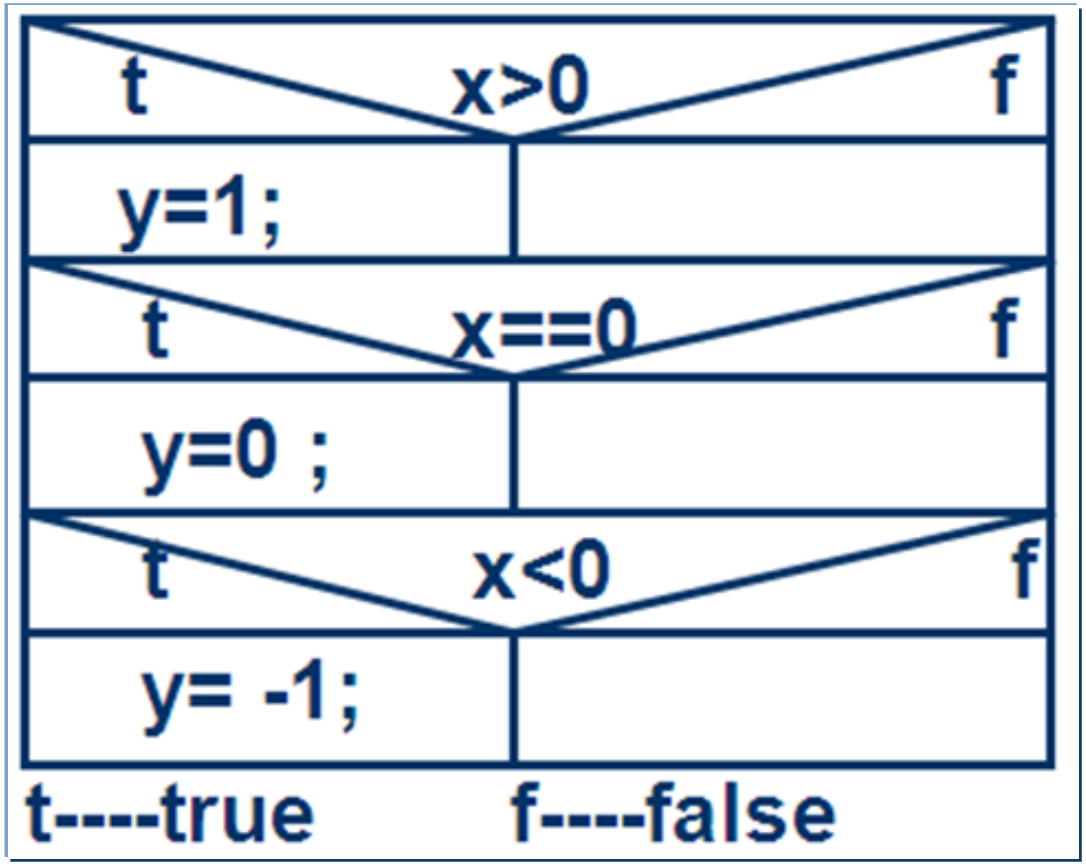
```
void main()
{
        float grade;
        cout<<"\n 请输入期末考试成绩:”<
        cin>>grade;
        if(grade>=90)
                {cout<<" 优"<<endl;}
        else if ((grade>=80) && (grade<90))
                {cout<<"良"<<endl;}
        else if ((grade>=60) && (grade<80))
                {cout<<" 中"<<endl;}
        else
                {cout<<"差"<<
}
```

请输入期末考试成绩:
差

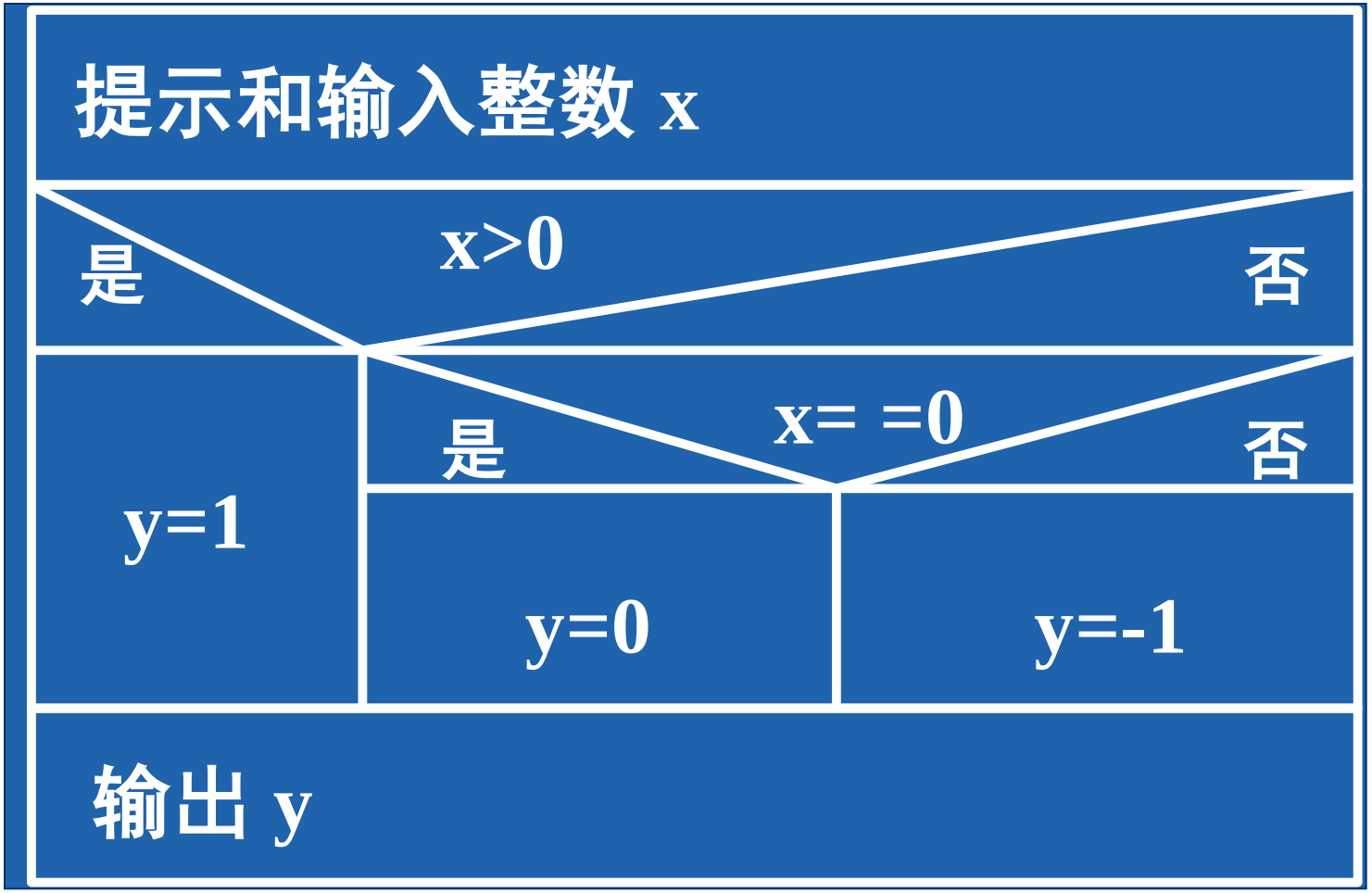
举例：实现如图函数

$$y=\begin{cases}1, & x>0\\ 0, & x=0\\ -1, & x<0\end{cases}$$




t
x>0
f
y=1;
t
x==0
f
y=0 ;
t
x<0
f
y= -1;
t----true
f----false

NS图



选择结构练习

- 求$ax^2+bx+c=0$方程的解
  - $a=0$，不是二次方程
  - $b^2-4ac=0$，有两个相等实根
  - $b^2-4ac>0$，有两个不等实根
  - $b^2-4ac<0$，有两个共轭复根
- **解题要求**
  - 画出程序流程图或NS图
  - 编写程序

<table>
<tr><td rowspan="4">输出“非二次方程”</td><td colspan="3">a=0</td></tr>
<tr><td>真</td><td colspan="2">假</td></tr>
<tr><td rowspan="3">相等实根
$-\frac{b}{2a}$</td><td colspan="2">b^2-4ac=0</td></tr>
<tr><td>真</td><td>假</td></tr>
<tr><td colspan="2">b^2-4ac>0</td></tr>
<tr><td>真</td><td>假</td></tr>
<tr><td>计算方程根
$x_1=-\frac{b+\sqrt{b^2-4ac}}{2a}$
$x_2=-\frac{b-\sqrt{b^2-4ac}}{2a}$</td><td>计算实部p
虚部q
$p=-\frac{b}{2a}$
$q=\frac{\sqrt{-(b^2-4ac)}}{2a}$</td></tr>
<tr><td>输出实根</td><td>输出虚根</td></tr>
</table>

数学函数

#include<math.h>

- 求绝对值函数　abs(-4)　labs(-41576)　fabs(3.14)
- 正弦函数　sin(3.141569265/2)=1
- 反正弦函数　asin(0.32696)=0.333085
- 余弦函数　cos(3.141569265/2)=4.489-011
- 反余弦函数　acos(0.32696)=1.23771
- 正切函数　tan(3.141569265/2)=1
- 反正切函数　atan(-862.42)=-1.56964
- 计算e^x函数　exp(1)=2.71828
- 计算lnx函数　log(2.71828)=0.999999
- 计算以10为底的对数函数$\log_{10}x$　log10(10)=1
- 计算x^y函数　pow(2,3)=8
- 计算不大于自变量x的整数值函数　floor(-2.8)=-3
- 计算x开方函数　sqrt(42.25)=6.5

多重分支语句

```
void main()
{
    float a=10;

    if(a>5)
    {
        cout<<"5"<<endl;
    }
    else if(a>4)
    {
        cout<<"4"<<endl;
    }
    else if(a>3)
    {
        cout<<"3"<<endl;
    }
}
```

5

```
void main()
{
    float a=10;

    if(a>5)
    {
        cout<<"5";
    }
    if(a>4)
    {
        cout<<"4";
    }
    if(a>3)
    {
        cout<<"3";
    }
}
```

543

多重分支语句

```
if(number>500) cost=0.15;
else if(number>300) cost=0.1;
else if(number>100) cost=0.075;
else cost=0;
```

多重分支语句

良好的编程风格

```
if(number>500)
{
   cost=0.15;
}
else if(number>300)
{
   cost=0.1;
}
else if(number>100)
{
   cost=0.075;
}
else
{
   cost=0;
}
```

switch 结构 3-1

嫌疑犯照片

不是

不是

是

警察记录

这是吉姆，身高 175cm，
墨西哥州海鸥路弗罗拉公寓，
于 2003 年 11 月 1 日在新泽西因抢银行被捕

加利

搜索

输出(“这是丹尼尔，身高 193cm，现居住地不明，1998 年 12 月 12 日在佛罗里达被捕”);

停止搜索

输出(“这是吉姆，身高 175cm，墨西哥州海鸥路弗罗拉公寓，于 2003 年 11 月 1 日在新泽西因抢银行被捕”);

停止搜索

default:

输出(“未发现匹配项。对不起！”);

停止搜索

switch 结构 3-2

- switch (表达式) 语句
- switch 语句计算条件表达式并对照多个常数值进行检查

计算表达式的值

```
{
    case 常量 1:
        语句;
        break;

    case 常量 2:
        语句;
        break;

    default:
        语句;
}
```

如果等于常量1

如果等于常量2

如果没有找到匹配的值

switch 结构 3-3

- **在使用switch结构时应注意以下几点：**
  - 在case后的各常量表达式的值不能相同，否则会出现错误；
  - 在case后，允许有多个语句，可以不用{}括起来；
  - 每个case语句后都必须有一个break语句，否则会出现错误；
  - 各case和default子句的先后顺序可以变动，而不会影响程序执行结果；
  - default子句可以省略；

switch结构

- 多路选择的switch语句

Example:

```
switch (month) {
    case 1:
            printf("Janu
            break;
    case 2:
            printf("Februa
            break;
    case 3:
            printf("March\n");
            break;
    default:
            printf("Others\n");
            break;
}
printf("End");
```

Assume month = 1, so …

this step will be …*case* is terminated here. Jump to …

```
January
End _
```

switch结构

- 多路选择的switch语句

Example:

```
switch (month) {
    case 1:
            printf("January\n");
            break;
    case 2:
            printf("February\n");
            break;
    case 3:
            printf("Marc
            break;
    default:
            printf("Others
            break;
}
printf("End");
```

```
March
End _
```

Assume month = 3, so …

…this step will be …*case* is terminated here. Jump to …

switch结构

- 多路选择的switch语句

Example:

```
switch (month) {
        case 1:
                printf("January\n");
                break;
        case 2:
                printf("Februar
                break;
        case 3:
                printf("March\n
                break;
        default:
                printf("Others\n
                break;
}
printf("End");
```

Now…what will happen if this *break* is taken out from the program?

switch结构

- 多路选择的switch语句

Example:

```
switch (month) {
        case 1:
                printf("January\n");
                break;
        case 2:
                printf("February\n");
                
        case 3:
                printf("March\n");
                break;
        default:
                printf("Others\n");
                break;
}
printf("End");
```

No more !

switch结构

- 多路选择的switch语句

Example: switch (month)

case 1:

printf("January\n");
break;

case 2:

printf("February\n");

case 3:

printf("March
break;

default:

Assume month = 2, so …

End _

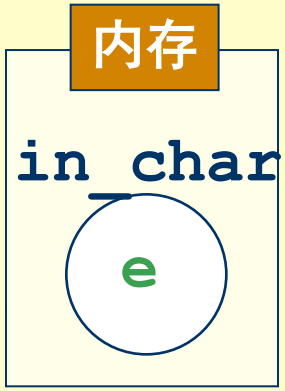
…this step will be executed. Later …

…execution continues. Thus, this step is executed . So …

…*case* is terminated here. Jump to …

switch 结构示例

```
char in_char;
cout<<"请输入一个小写字母：“<<endl;
cin>>in_char;
switch(in_char)
{
    case 'a':  cout<<" 您输入的是元音字母 a“<<
               break;
    case 'e':  cout<<"您输入的是元音字母 e“<<endl;
               break;
    case 'i':  cout<<"您输入的是元音字母 i“<<endl;
               break;
    case 'o':  cout<<" 您输入的是元音字母 o“<<endl;
               break;
    case 'u':  cout<<" 您输入的是元音字母 u“<<endl;
               break;
    default:  cout<<" 您输入的不是元音
}
```



请输入一个小写字母：e
您输入的是元音字母 e

【例】 计算器程序

- 编程设计一个简单的计算器程序，要求用户从键盘输入如下形式的表达式：

操作数1　　运算符op　　操作数2

然后，计算并输出表达式的值

指定的运算符为

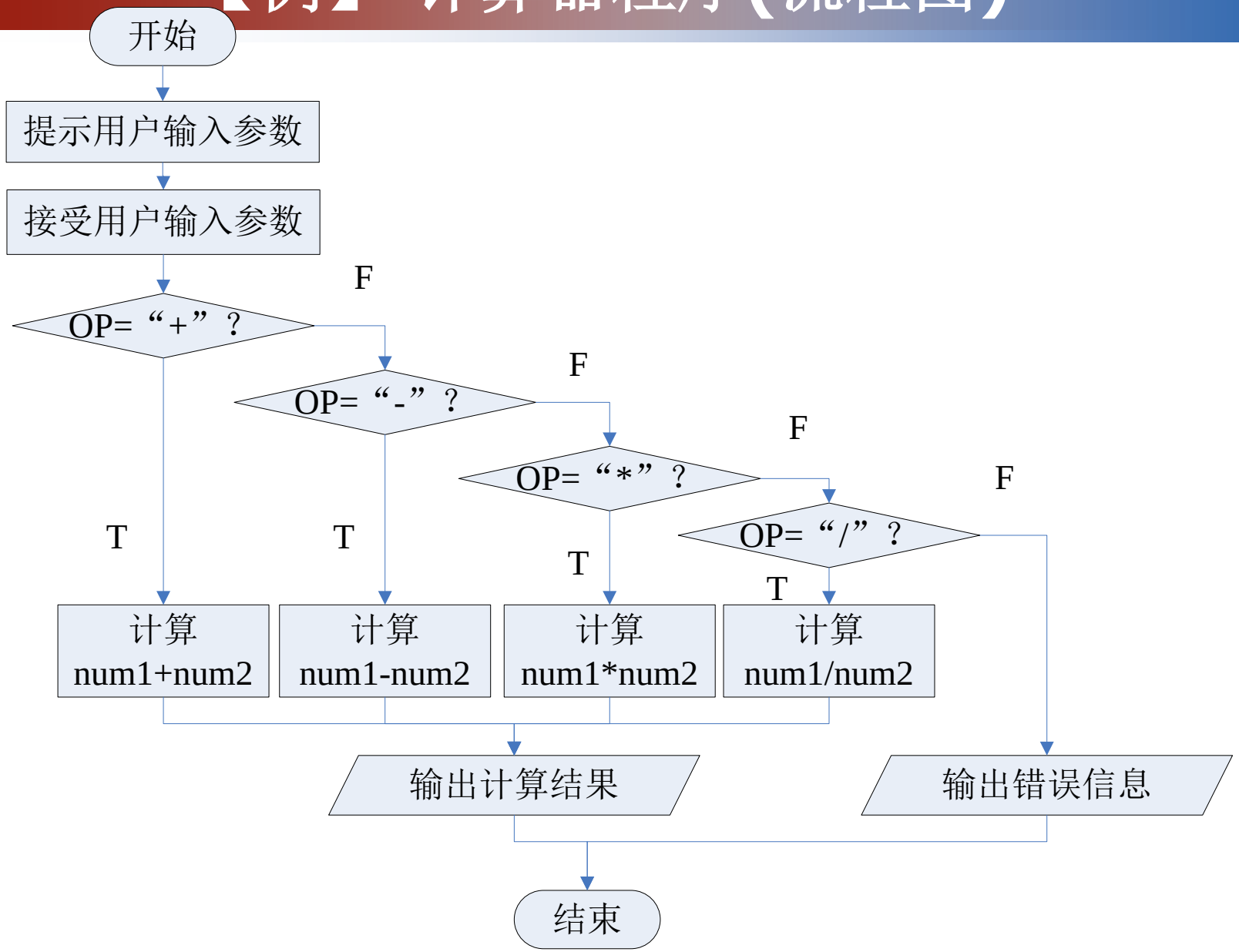
加（+）

减（-）

乘（*）

除（/）

【例】 计算器程序(流程图)


开始
提示用户输入参数
接受用户输入参数
OP=“+”？
F
T
OP=“-”？
F
T
OP=“*”？
F
T
OP=“/”？
F
T
计算
num1+num2
计算
num1-num2
计算
num1*num2
计算
num1/num2
输出计算结果
输出错误信息
结束

【例】 计算器程序(源程序)

```
switch (op) {
    case '+':                    /*处理加法*/
        cout<<num1<<"+"<<num2<<"="<<num1+num2<<endl;
        break;
    case '-':                    /*处理减法*/
        cout<<num1<<"-"<<num2<<"="<<num1-num2<<endl;
        break;
    case '*':                    /*处理乘法*/

        cout<<num1<<"*"<<num2<<"="<<num1*num2<<endl;
        break;
    case '/'                     /*处理除法*/
        if (0 == num2)
            cout<<"Division by zero!"<<endl;
        else
        cout<<num1<<"/"<<num2<<"="<<num1/num2<<endl;
        break;
    default:
        cout<<"Invalid operator!"<<endl;
}
```

比较多重 if 和 switch 结构

- 多重if结构和switch结构都可以用来实现多路分支
- 多重if结构用来实现两路、三路分支比较方便，而switch结构实现三路以上分支比较方便
- 在使用switch结构时，应注意分支条件要求是整型表达式，而且case语句后面必须是常量表达式
- 有些问题只能使用多重if结构来实现，例如要判断一个值是否处在某个区间的情况

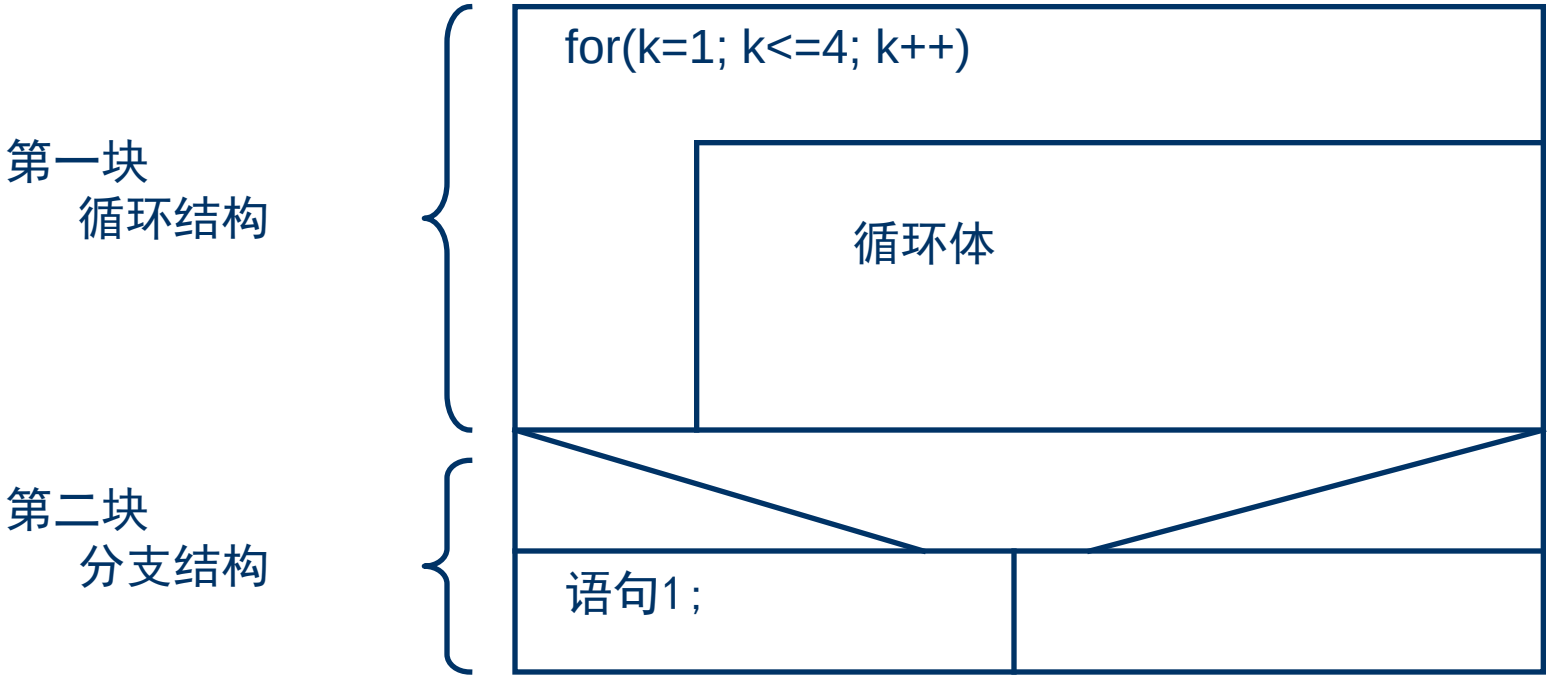
思考：谁做了好事问题中的条件表达

假设对i为做好事的同学这种情况进行判断

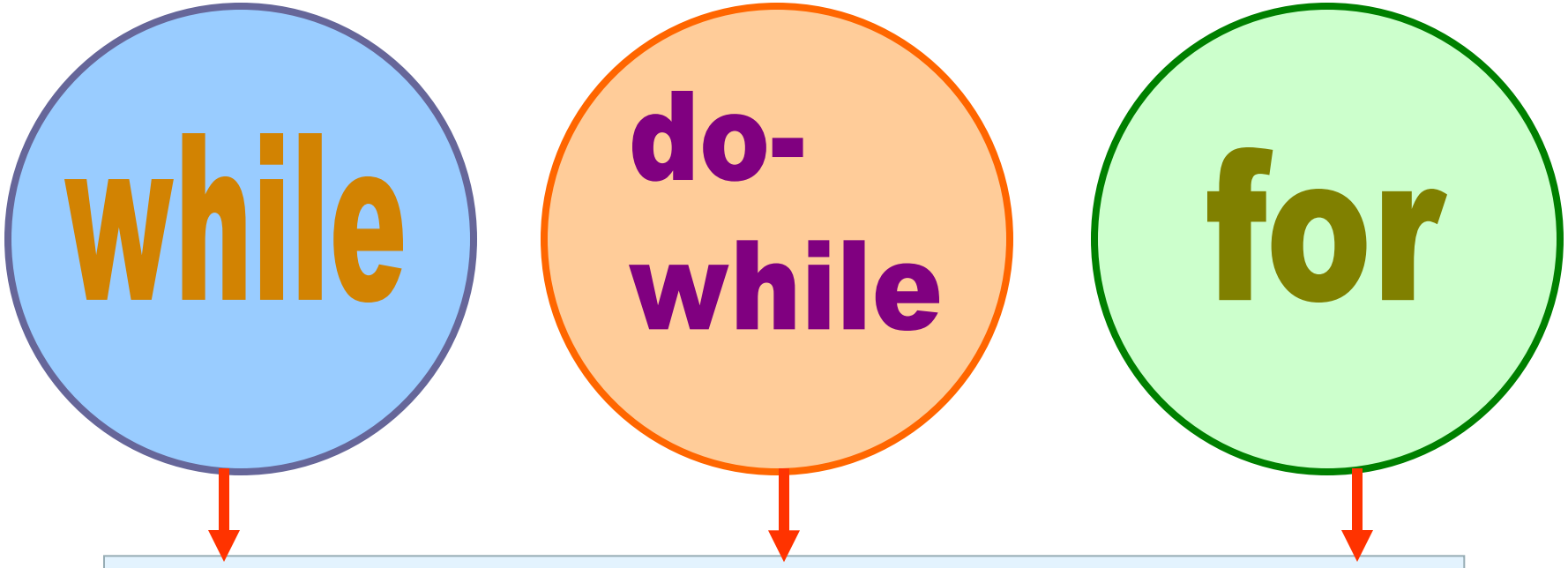
(i可取值为ABCD)

```
int sum=0;
char thisman=‘ i’;
sum=(thisman!=‘A’)+(thisman==‘C’)+
        (thisman==‘D’)+(thisman!=‘D’)
if(sum==3)
{
        cout<<i<<“做了好事” <<endl;
}
```

“谁做了好事”



C 语言中的各种循环



需要多次重复执行一个或多个任务的问题考虑使用循环来解决

while 循环 3-1

while 循环的一般语法：

```
while (表达式)
{
    语句;
}
```

工作原理

计算表达式的值，当值为真(非0)时，执行循环体语句，一旦条件为假，就停止执行循环体。如果条件在开始时就为假，那么不执行循环体语句直接退出循环。

示例 int index=0;

```
while (index < 100)
{
    . . .
    . . .
    index++;
}
```

while 循环 3-2

规则1：

```
[<初始化>]

while(循环条件)
{
    <循环体>
}
```

循环条件中使用的变量需要经过初始化

while 循环 3-3

规则2：

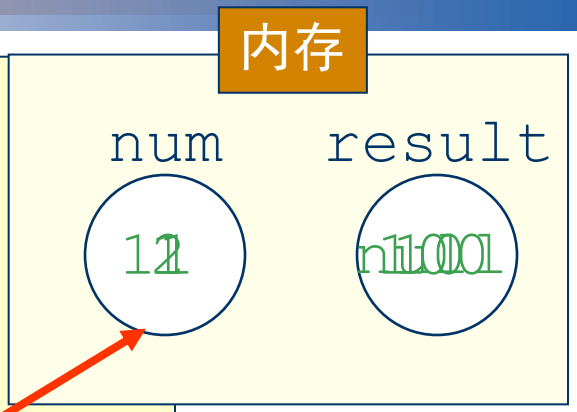
```
while (index < 100)
{
    . . .
    . . .
    index++;
}
```

while 循环主体中的语句必须修改循环条件的值，否则会形成死循环

while (1) — 真

while 循环示例1

```
#include<iostream.h>
void main ()
{
    int num=1,result;
    while (num<=10)
    {
      result=num*10;
      cout<<num<<" × 10 ＝ “<<result<<end
      num++;
    }
}
```



输出：

1 × 10 ＝ 10
2 × 10 ＝ 20
3 × 10 ＝ 30
4 × 10 ＝ 40
5 × 10 ＝ 50
6 × 10 ＝ 60
7 × 10 ＝ 70
8 × 10 ＝ 80
9 × 10 ＝ 90
10 × 10 ＝ 100

用while循环完成“谁做的好事”

thisman可取从A到D
即ADCII码从65-68
thisman =65 ;

注意

- 循环变量初始化
- 循环条件判定
- 循环变量控制

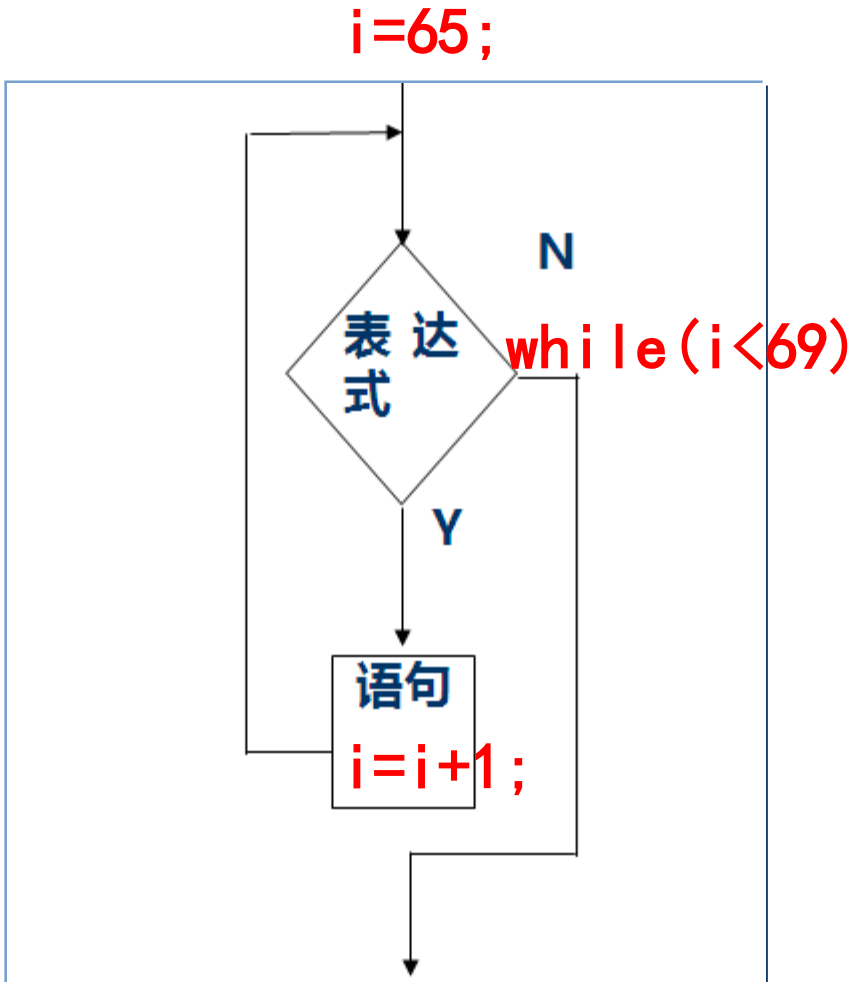


图3-4 “先判断后执行”结构示意图
示意图

do-while 循环

while 循环的一般语法：

```
do
{
    语句;
} while (表达式);
```

工作原理

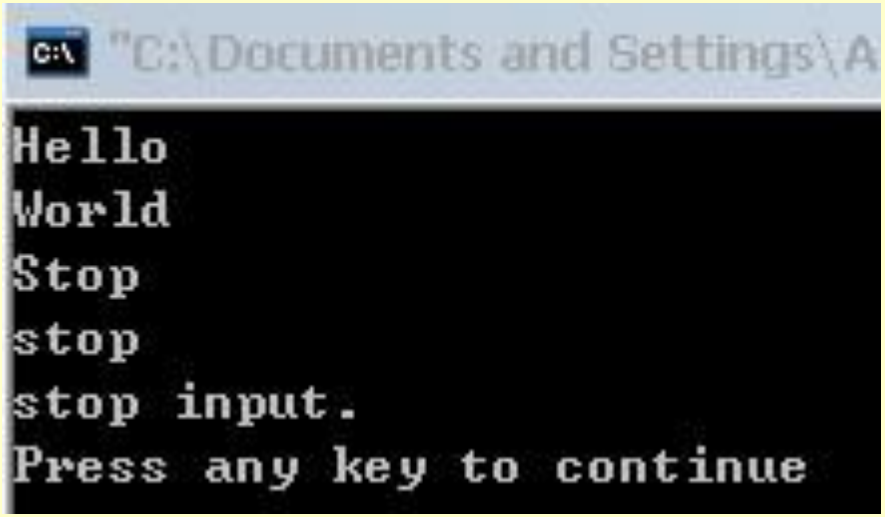
它先执行循环体中的语句，然后再判断条件是否为真，如果为真则继续循环；如果为假，则终止循环。

示例 string input;

```
do
{
  . . .
  . . .
  cin>>input;
} while (input != “stop”);
```

do-while 循环示例

```
#include <iostream>
#include <string>
using namespace std;
void main()
{
    string input;
    do
        {
            cin>>input;
        }while(input!="stop");
    cout<<"stop input."<<endl;
}
```



用do-while循环完成“谁做的好事”

thisman可取从A到D
即ADCII码从65-68
thisman =65 ;

注意

- 循环变量初始化
- 循环变量控制
- 循环条件判定

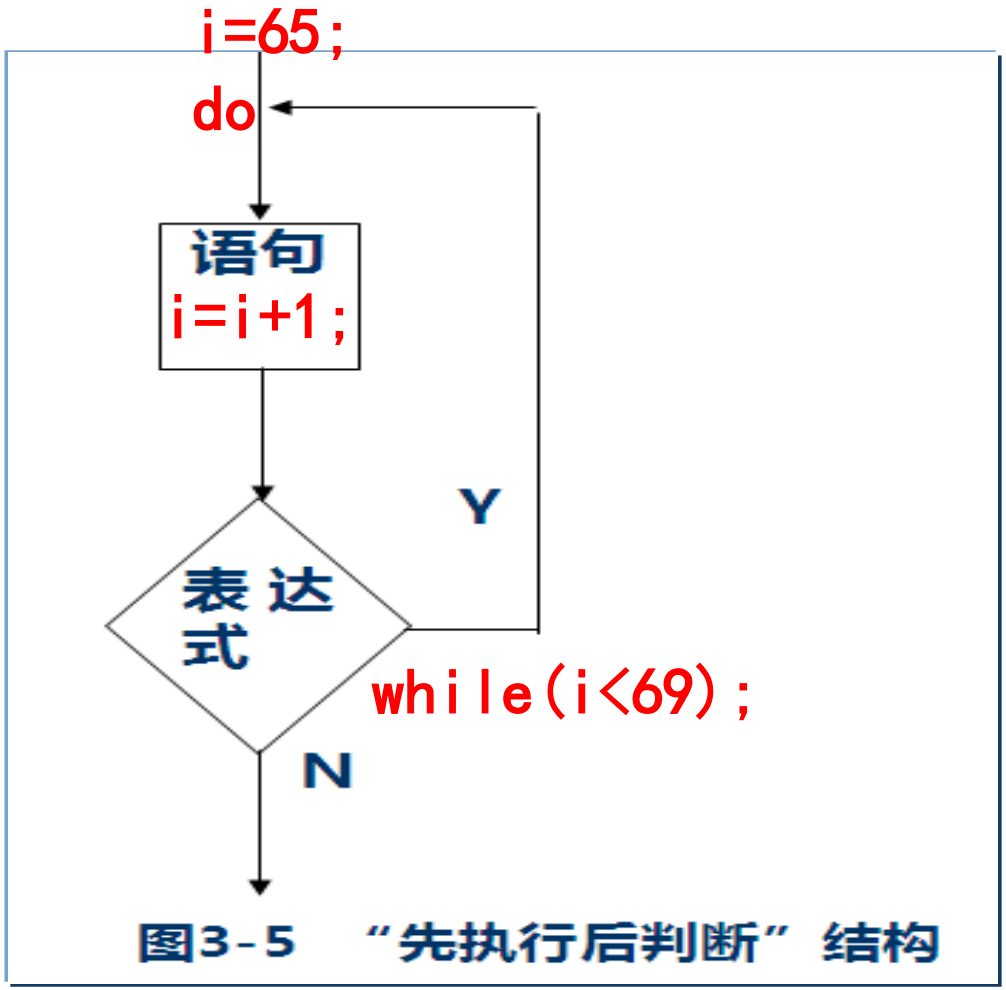


图3-5 “先执行后判断”结构

比较 while 和 do-while循环

- 比较 while 和 do-while 循环的工作原理

```
while(循环条件)
{
    循环体;
}
```

```
do
{
    循环体;
 }while( 循环条件);
```

while循环是先判断后执行，所以，如果条件为假，则循环体一次也不会被执行。

do-while 循环是先执行后判断，所以，即使开始条件为假，循环体也至少会被执行一次。

for 循环

for 循环的一般语法：

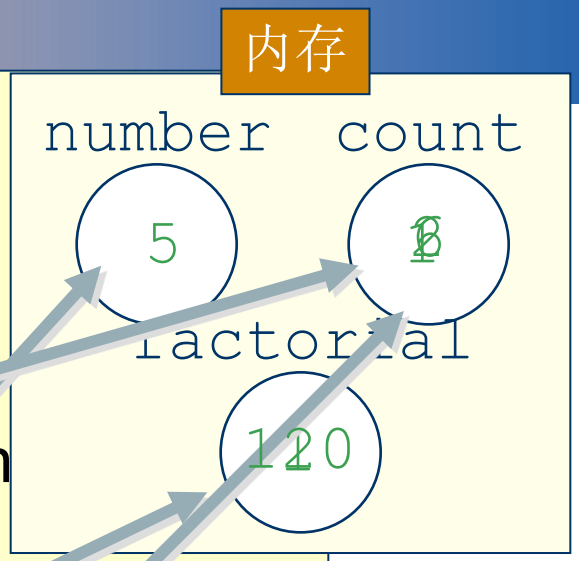
```
for( 表达式1 ; 表达式2 ; 表达式3 )
{
    语句;
}
```

工作原理

1、计算表达式1的值，通常为循环变量赋初值；
2、计算表达式2的值
3、即判断循环条件是否为真，若值为真则执行循环体一次， 否则跳出循环；
4、计算表达式3的值，这里通常写更新循环变量的赋值表达式，然后转回第2步重复执行；

for 循环示例1

```
#include <stdio.h>
void main()
{
    int number,count,factorial=1;
    cout<<" 请输入任意一个正整数：“<<en
    cin>>number;
    for(count = 1; count <=number; count++)
                factorial=factorial*count;
    cout<<number<<"的阶乘 <<factorial;
}
```



循环执行五次

```
请输入任意一个正整数：5
5的阶乘 = 120
```

```
请输入一个值
根据这个值可以输出以下加法表：5
0 + 5 = 5
1 + 4 = 5
2 + 3 = 5
3 + 2 = 5
4 + 1 = 5
```

```
#include <iostream.h>
void main()
{
        int i,j,max;
        cout<<"请输入一个值 "<<endl;
        cout<<"根据这个值可以输出以下加法表："<<endl;
        cin>>max;
        for(i = 0,j = max ; i <=max ; i++,j--)
            { cout<<i<<'+'<<j<<'= '<<(i + j)<<endl; }
}
```

初始化语句
只执行一次

循环执行max次

for 循环的表达式

- for 循环中有三个表达式
- for 语句中的各个表达式都可以省略
- 分号分隔符不能省略

可省略

```
for(<初始化循环变量>;<循环条件>;<修改循环变量的值>)
{
        <循环体语句>;
}
```

不能省略

用for循环完成“谁做的好事”

for(i=65; i<69; i++)

thisman可取从A到D
即ADCII码从65-68
thisman =65 ;

注意

- 循环变量初始化
- 循环条件判定
- 循环变量控制

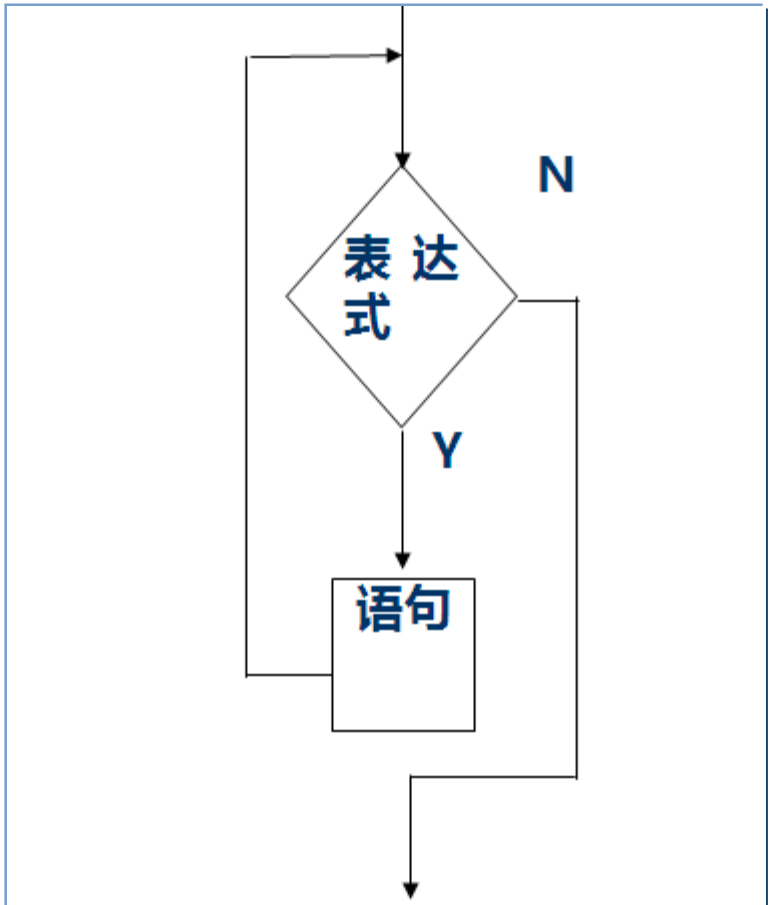


图3-4 “先判断后执行”结构示意图
示意图

练习

- **新年到，班级组织晚会需要采购坚果和水果，有100元钱供晚会购物。编写采购程序，按照市场价格确定坚果和水果的单价**

  - 方案一：100元钱物尽其能，尽量多的购买所需物品，输出购买方案（完成程序编写）

  - 方案二：成本控制在100元以内，输出所有可能的购买方案（思考并完成）

break 语句 2-1

- **break**语句可以改变程序的控制流
- **break**语句用于**do-while**、**while**、**for**循环中时，可使程序终止循环而执行循环后面的语句
- **break**语句通常在循环中与条件语句一起使用。若条件值为真，将跳出循环，控制流转向循环后面的语句
- 如果已执行**break**语句，就不会执行循环体中位于**break** 语句后的语句
- 在多层循环中，一个**break**语句只向外跳一层

break 语句 2-2

跳出 for 循环

```
for( ; ; )
{
    cout<<"这将一直进行下去";
    i = getchar();
    if(i == 'X' || i == 'x')
        break;
}
```

跳出 while 循环

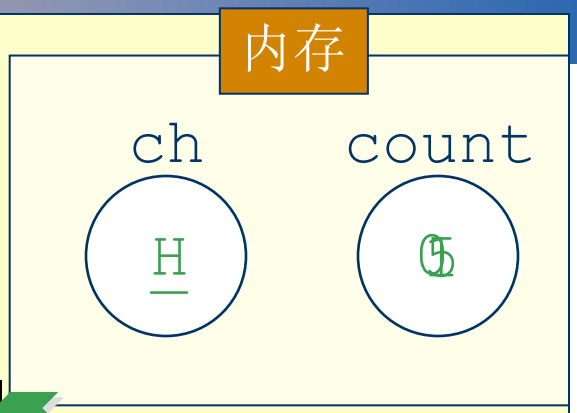
```
while(1)
{
    if(x == 10)
        break;
}
```

跳出 do-while 循环

```
do
{
    if (x == 10)
        break;
}while (x< 15);
```

break 语句示例

```
#include<stdio.h>
void main()
{
        int count=0,ch;
        cout<<" 请输入一行字符：“<<endl;
        while((ch=getchar())!='\n')
        {
                if(ch==' ')
                        break;
                count++;
        }
        cout<<" 共有 " <<count<<"个有效字符”<<endl;
}
```



循环执行 5 次

请输入一行字符：Hello world
共有 5 个 有效字符

continue 语句 2-1

- **continue** 语句只能用在循环里
- **continue** 语句的作用是跳过循环体中剩余的语句而执行下一次循环
- 对于**while**和**do-while**循环，**continue** 语句执行之后的动作是条件判断；对于**for**循环，随后的动作是变量更新

continue 语句 2-2

```
while(…)
{
    ……
    ……
     break;
    ……
    ……
}
```

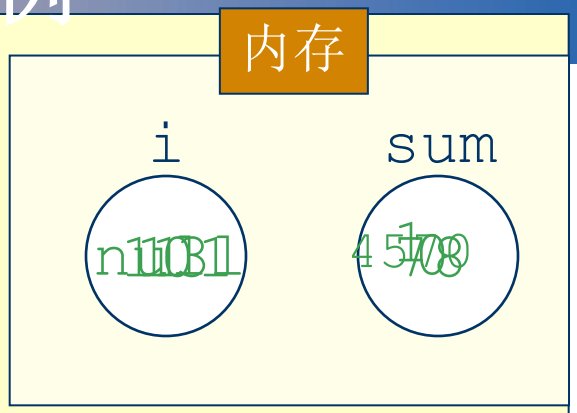
跳出整个循环

```
while(…)
{
    ……
    ……
     continue;
    ……
    ……
}
```

继续下一次循环

continue 语句示例

```
#include<stdio.h>
void main()
{
        int i,sum = 0;
        for(i=1; i<=100;i++)
        {
                if( i % 10 == 3)
                        continue;
                sum += i;
        }
        printf("sum = %d \n",sum);
}
```



循环执行到 i = 13

循环执行到 i = 101

输出：
sum = 4570

逻辑问题及其解法

任务讨论

某地刑侦大队对涉及六个嫌疑人的一桩疑案进行分析：

- A、B 至少有一人作案；
- A、E、F 三人中至少有两人参与作案；
- A、D 不可能是同案犯；
- B、C 或同时作案，或与本案无关；
- C、D 中有且仅有一人作案；
- 如果 D 没有参与作案，则 E 也不可能参与作案。

试编一程序，将作案人找出来。

思路

1、案情分析：

将案情的每一条写成逻辑表达式，
第一条用 CC1 表示，
第二条用 CC2 表示，
……

思路:案情分析1

CC1：A和B至少有一人作案

令 A 变量表示 A 作案，B 变量表示 B 作案

A	B	CC1
0	0	0
1	0	1
0	1	1
1	1	1

显然这是或的关系，有CC1= (A || B)

思路:案情分析2

CC2：A和D不可能是同案犯

可以分析为：

- A 和D是同案犯，写成

$$A \ \&\& \ D$$

- A 和 D 不是同案犯，写成

$$!(\ A \ \&\& \ D\ )$$

因此有

$$CC2 = !(\ A \ \&\& \ D\ )$$

思路:案情分析3

CC3：A、E、F 中至少有两人涉嫌作案

分析有三种可能

- 第一种，A 和 E 作案，（A && E）
- 第二种，A 和 F 作案，（A && F）
- 第三种，E 和 F 作案，（E && F）

这三种可能性是 或 的关系，因此有

CC3 = (A && E）||（A && F）||（E && F)

思路:案情分析4

CC4：B和C或同时作案，或都与本案无关

- **第一种情况：**
  - 同时作案（B && C）
- **第二种情况：**
  - 都与本案无关（!B && !C）

两者为或的关系，因此有

CC4 = (B && C) || (!B && !C)

思路:案情分析5

CC5：C、D中有且仅有一人作案

CC5=(C && !D) || (D && !C)

思路:案情分析6

CC6：如果D没有参与作案，则E也不可能参与作案。

分析这一条比较麻烦一些，可以列出真值表再归纳

$$
\begin{aligned}
& DE + D\bar{E} + \bar{D}\bar{E} \\
&= (DE + D\bar{E}) + (D\bar{E} + \bar{D}\bar{E}) \\
&= D(E + \bar{E}) + \bar{E}(D + \bar{D}) \\
&= D + \bar{E}
\end{aligned}
$$

CC6=D || !E

以上是案情分析，已经化成了计算机可解的逻辑表达式

2、采取枚举方法，枚举什么呢？枚举组合

6个人每个人都有作案或不作案两种可能，因此有 2^6 种组合，从这些组合中挑出符合6条分析的作案者。

定义 6 个整数变量，分别表示 6 个人A，B，C，D，E，F。

枚举每个人的可能性

- 让 0 表示不是罪犯；
- 让 1 表示就是罪犯。

思考

- 编一个程序输出
- 000000
- 000001
- 000010
- .....
- 111111
- 从高位到低位，分别用ABCDEF来表示

A	B	C	D	E	F
0	0	0	0	0	0
0	0	0	0	0	1
0	0	0	0	1	0
……	……				……
1	1	1	1	1	1

循环的应用

```
for( A=0;A<=1;A=A+1)
  for( B=0;B<=1;B=B+1)
    for( C=0;C<=1;C=C+1)
      for( D=0;D<=1;D=D+1)
        for( E=0;E<=1;E=E+1)
          for( F=0;F<=1;F=F+1)
        {  cout<<A<<B<<C<<D<<E<<F
                <<endl;      }
```

```
for( F=0;F<=1;F=F+1)
    {  cout<<A<<B<<C<<D<<E<<F
       <<endl;      }
```

循环体

```
for( E=0;E<=1;E=E+1)

    for( F=0;F<=1;F=F+1)
  {  cout<<A<<B<<C<<D<<E<<F
          <<endl;      }
```

循环体

```
for( D=0;D<=1;D=D+1)

  for( E=0;E<=1;E=E+1)
   for( F=0;F<=1;F=F+1)
{  cout<<A<<B<<C<<D<<E<<F
          <<endl;      }
```

循环体

```
for( A=0;A<=1;A=A+1)

  for( B=0;B<=1;B=B+1)
    for( C=0;C<=1;C=C+1)
     for( D=0;D<=1;D=D+1)
       for( E=0;E<=1;E=E+1)
         for( F=0;F<=1;F=F+1)
       {  cout<<A<<B<<C<<D<<E<<F
                <<endl;      }
```

循环体

作业

请完成输出**000000**到**111111**的程序并上机运行

画出根据案情分析采用枚举法寻找罪犯的程序框图

```
for (A=0; A<=1; A=A+1)
    for (B=0; B<=1; B=B+1)
        for (C=0; C<=1; C=C+1)
            for (D=0; D<=1; D=D+1)
                for (E=0; E<=1; E=E+1)
                    for (F=0; F<=1; F=F+1)
                        CC1=A||B;
                        CC2=!(A&&D);
                        CC3=(A&&E)||(A&&F) ||(E&&F);
                        CC4=(B&&C)||(!B&&!C);
                        CC5=(C&&!D)||(D&&!C);
                        CC6=D||(!E);
                        CC1+CC2+CC3+CC4+CC5+CC6==6
                        真 | 假
                        输出
```

讨　论

大家参与讨论的题

五位跳水高手将参加十米高台跳水决赛，有好事者让五个人据实力预测比赛结果。

A选手说：B第二，我第三；
B选手说：我第二，E第四；
C选手说：我第一，D第二；
D选手说：C最后，我第三；

E选手说：我第四，A第一;

决赛成绩公布之后，每位选手的预测都只说对了一半，即一对一错，请编程解出比赛的实际名次。

思路：

1、首先是将五个人的预测写成逻辑表达式：

- 让关系运算符 “==” 的含义是 “是” 。
- 让数字 1、2、3、4、5 分别表示名次第一、第二，…，第五。
- 让整型变量 A、B、C、D、E 分别表示每个选手所得名次。

A选手说：B==2，A==3；
B选手说：B==2，E==4；
C选手说：C==1，D==2；
D选手说：C==5，D==3；
E选手说：E==4，A==1；

2、考虑到每个人说的话是一对一错，

即一真一假，比如A说的

(B==2)+(A==3)　　应该是 1

1　+　0　　=　1

0　+　1　　=　1

我们可以归纳出要同时满足五个人所说的话都符合**一半对一半错的条件**是

ta= ((B==2)+(A==3))==1; 符合**A**选手的话

tb= ((B==2)+(E==4))==1; 符合**B**选手的话

tc= ((C==1)+(D==2))==1; 符合**C**选手的话

td= ((C==5)+(D==3))==1; 符合**D**选手的话

te = ((E==4)+(A==1))==1; 符合**E**选手的话

因为**ta,tb,…,te**非**1**即**0**，五个条件值都加在一起

$$0 \le ta + tb + tc + td + te \le 5$$

ta 的取值是 0 或 1

ta 的取值取决于下列关系

((B==2)+(A==3))==1

B==2---------可能是 0 或 1

A==3---------可能是 0 或 1

((B==2)+(A==3)) ---可能是 0 或 1 或 2

只有上式为 1 才能使 ta 为 1

3、只有等于5 时才都符合每个人所说的话，这仅只是符合题意的一个必要条件，同时还得考虑 A、B、C、D、E 的取值不得有相同者

$$A, B, C, D, E \in \{1, 2, 3, 4, 5\}$$

$$\text{且} A \neq B \neq C \neq D \neq E$$

可以考虑

$$A * B * C * D * E == 120$$

是一个条件

4、仍然可以用枚举的方法，让变量A、B、C、D、E 在 1~5 中取值，形成满足上述条件的 A~E 的组合，即是所求。这时可用循环结构，如下所示

```
for (A=1;A<=5;A=A+1)
    for (B=1;B<=5;B=B+1)
        for (C=1;C<=5;C=C+1)
            for (D=1;D<=5;D=D+1)
                E=15-(A+B+C+D);
                A*B*C*D*E==120
                真:
                    ta=((B==2)+(A==3))==1;
                    tb=((B==2)+(E==4))==1;
                    tc=((C==1)+(D==2))==1;
                    td=((C==5)+(D==3))==1;
                    te=((E==4)+(A==1))==1;
                    t=ta+tb+tc+td+te;
                    t==5
                    真:
                        输出 A, B, C, D, E
                    假:
                    [空]
                假:
                [空]
```

作业：

1 请你自己完成这个程序；

2 思考如何提高效率？

3 思考还有什么其它解法？

结　　束

总结

- 多重if结构就是在主if块的else部分中还包含其他if块
- 嵌套if结构是在主if块中还包含另一个if 语句
- C语言规定，嵌套if结构中每个else部分总是属于前面最近的那个缺少对应的else部分的if语句
- switch结构也可以用于多分支选择。用于分支条件是整型表达式，而且判断该整型表达式的值是否等于某些值（可以罗列的），然后根据不同的情况，执行不同的操作
- 条件运算符是 if–else 语句的另一种表现形式