



中國人民大學
RENMIN UNIVERSITY OF CHINA

第4讲 控制语句

余力

buaayuli@ruc.edu.cn

内容提要

- 4.1 逻辑思维与计算机解题
- 4.2 C语言的分支控制语句
- 4.3 C语言的循环控制语句



中國人民大學
RENMIN UNIVERSITY OF CHINA



01. 逻辑计算思维解题

先从一个例子说起.....

- 人大附中有四位同学中的一位做了好事，不留名，表扬信来了之后，校长问这四位是谁做的好事。

A说：不是我。

B说：是C。

C说：是D。

D说：他胡说。

已知三个人说的是真话，一个人说的是假话。现在要根据这些信息，找出做了好事的人。

- 题目如何求解？
- 题目用计算机如何求解？

问题1：如何对同学说的话建模

■ 回顾：关系运算符和关系表达式

运算符	说明	
>	大于	优先级相同（高）
>=	大于等于	
<	小于	
<=	小于等于	
= =	等于	优先级相同（低）
!=	不等于	

- 关系表达式返回的值是一个逻辑值，即“真”（1）或“假”（0）

利用关系表达式建模四个人所说的话 (1)

- 结合任务，可以将四个人说的四句话写成关系表达式
 - 在声明变量时，我们让 **thisman** 表示要寻找的做了好事的人，定义它是字符变量

char thisman=""; // 定义字符变量并初始化为空

- 接着让 “==” 的含义为 “是”
- 让 “!=” 的含义为 “不是”

利用关系表达式建模四个人所说的话 (2)

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'

问题2：如何确定可能的答案

■ 枚举法

- 结合任务分析，A、B、C、D四个人，只有一位是做好事者。令做好事者为1，未做好事者为0，可以有如下4种状态

状态	A	B	C	D
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

四种状态的形式化表示

状态	赋值表达式
1	thisman='A'
2	thisman='B'
3	thisman='C'
4	thisman='D'

- 显然第一种状态是假定A是做好事者，第二种状态是假定B是做好事者， ...
- 所谓枚举是按照者四种假定逐一地去测试四个人的话有几句是真话，如果不满足三句为真，就否定掉这一假定，换下一个状态再试。
- 具体做法如下：

(1) 假定让thisman= 'A' 代入四句话中

说话人	说的话	关系表达式	值
A	thisman!='A';	'A'!='A'	0
B	thisman=='C';	'A'=='C'	0
C	thisman=='D';	'A'=='D'	0
D	thisman!='D';	'A'!='D'	1

四个关系表达式的值的和为1，不满足3句话为真，假设不成立，因此显然不是 'A' 做的好事。

(2) 假定让thisman= 'B' 代入四句话中

说话人	说的话	关系表达式	值
A	thisman!='A';	'B'!='A'	1
B	thisman=='C';	'B'=='C'	0
C	thisman=='D';	'B'=='D'	0
D	thisman!='D';	'B'!='D'	1

四个关系表达式的值的和为2，显然不是 'B' 做的好事。

(3) 假定让thisman= 'C' 代入四句话中

说话人	说的话	关系表达式	值
A	thisman!='A';	'C'!='A'	1
B	thisman=='C';	'C'=='C'	1
C	thisman=='D';	'C'=='D'	0
D	thisman!='D';	'C'!='D'	1

四个关系表达式的值的和为3，就是 'C' 做的好事。

枚举

- 按照上面的思路，一个人一个人去试，就是枚举。
- 从编写程序看，实现枚举最好使用

循环程序结构

- 对于每个人，实现逻辑判断最好使用

分支程序结构

学习目标

- 将实际问题抽象为逻辑关系
- 使用枚举法解题
- 通过实例串联起之前内容
 - 关系表达式与逻辑表达式
 - 程序的循环结构
 - 程序的分支结构

回顾示例

- 人大附中有四位同学中的一位做了好事，不留名，表扬信来了之后，校长问这四位是谁做的好事。

A说：不是我。

B说：是C。

C说：是D。

D说：他胡说。

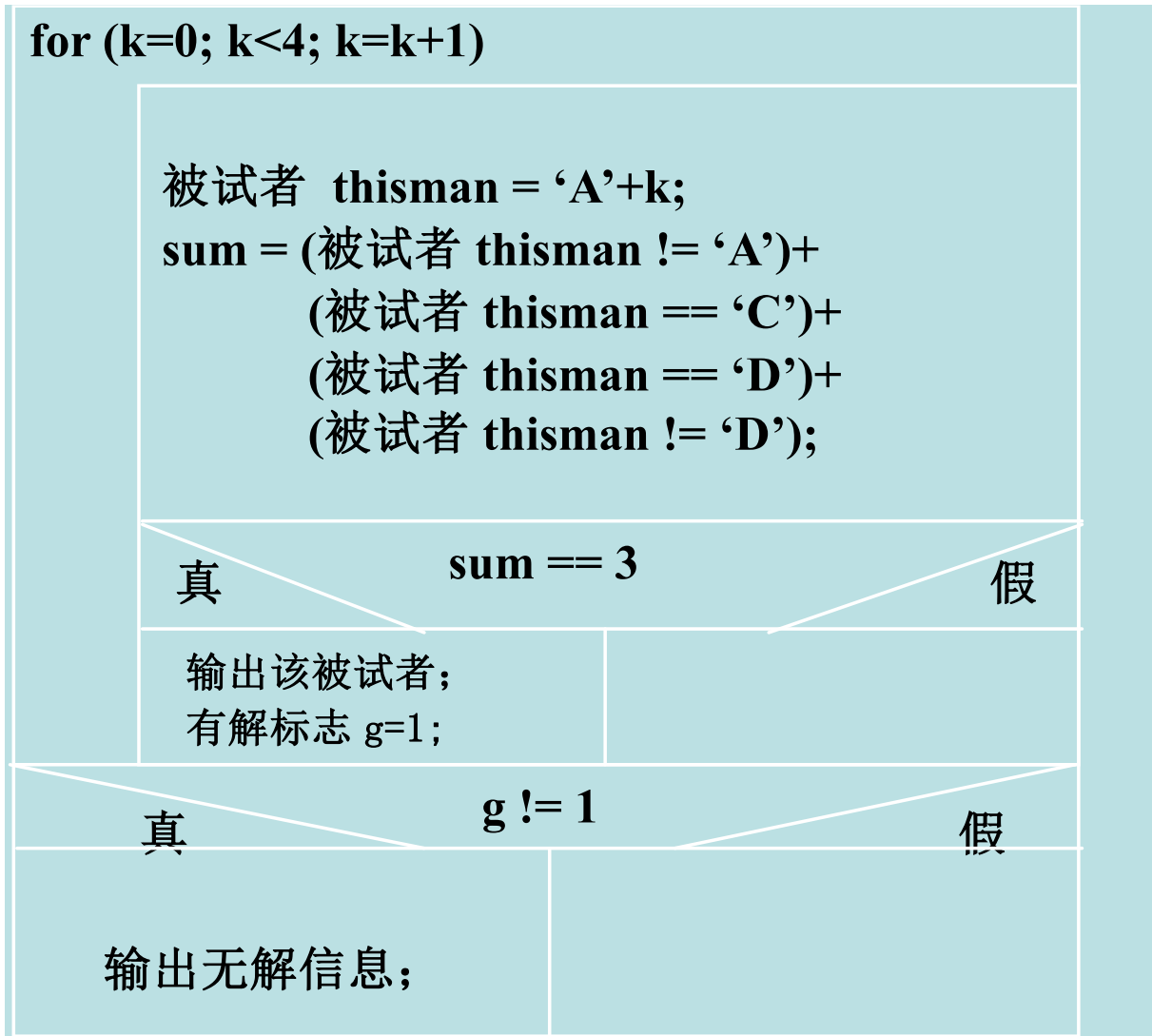
已知三个人说的是真话，一个人说的是假话。现在要根据这些信息，找出做了好事的人。

- 题目如何求解？
- 题目用计算机如何求解？

逻辑思维与计算机解题

- 一些逻辑问题必须转换成计算机能够看得懂的数学表达式和一定的程序指令
- 解题思路
 - 枚举法
- 数学表达式
 - 关系表达式与逻辑表达式
- 程序指令
 - 程序的循环结构
 - 程序的分支结构

程序NS图

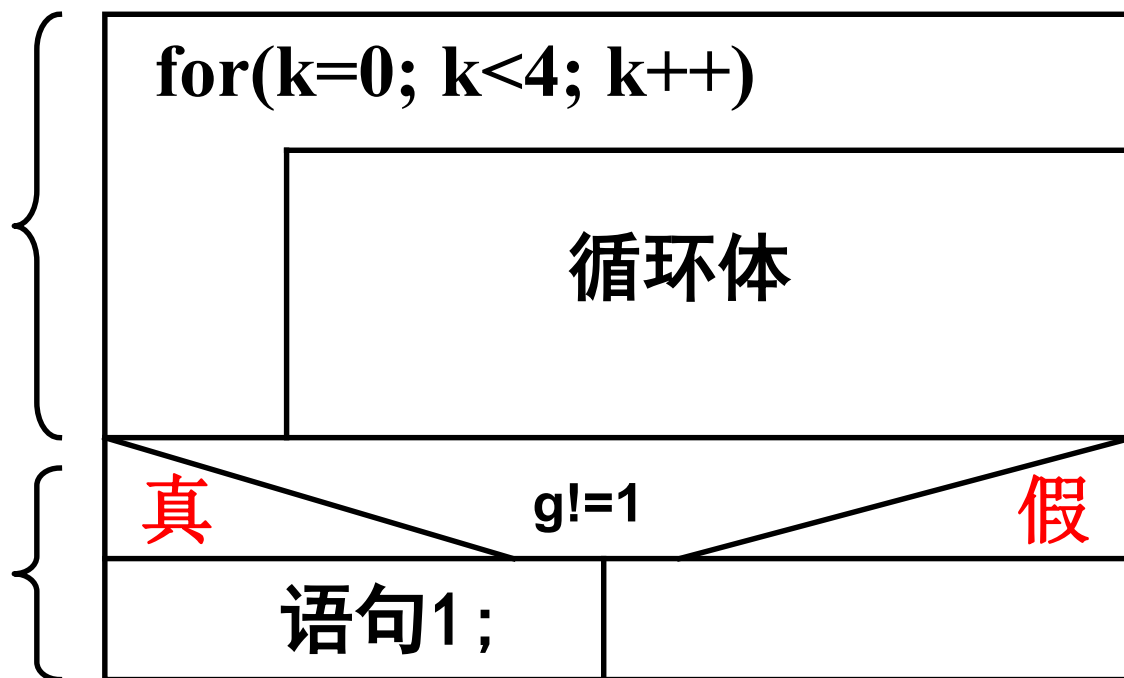


A说: 不是我。
B说: 是C。
C说: 是D。
D说: 他胡说。

程序NS图核心

第一块
循环结构

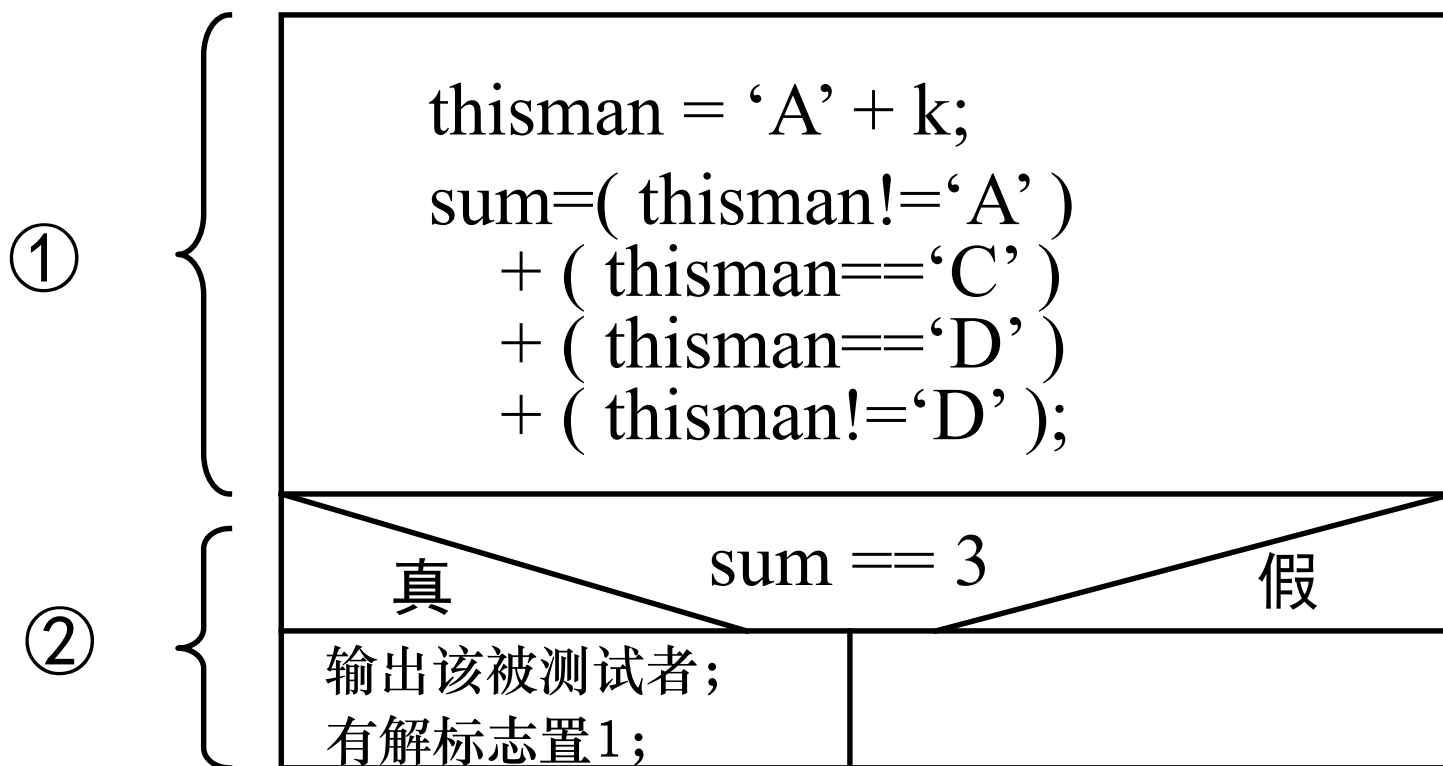
第二块
分支结构



- 再细看，第一块的循环体又由两块组成，如图4.9，

①中含两条赋值语句

②中含一条分支语句



```
1.  #include <stdio.h>    // 预编译命令
2.  int main() {          // 主函数
3.      int k=0,sum=0,g=0; //声明整数变量，且均初始化为0
4.      char thisman=' ';
5.      for(k=0; k<4; k++) { //k是计数器，表示第k+1个人
6.          thisman = 'A'+k;
7.          sum =( thisman!='A' )
8.              + ( thisman=='C' )
9.              + ( thisman=='D' )
10.             + ( thisman !='D' );
11.         if (sum==3) { // 如果4句话有3句话为真，则输出该人
12.             printf( "做好事者为%c\n ", thisman );
13.             g=1;      // 有解标志置1
14.         }
15.     }
16.     if (g!=1)          // 则输出无解信息
17.         printf( "Can' t found!\n ");
18.     return 0;
19. }
```

程序讨论

■ 变量设计

- 变量k: 为什么k初始化为0, 而不是1?
- 变量g: 为什么设计变量g?

■ 程序结构

- `for(k=0; k<4; k++)`
- 语句分号分隔的部分作用分别是什么?
- 根据题意, 能否提前终止循环?

示例2：找出作案人

- 某地刑侦队对涉及六个嫌疑人的一桩疑案进行案情分析：
 - A、B至少有一人作案；
 - A、D 不可能是同案犯；
 - A、E、F三人中至少有两人参与作案；
 - B、C或同时作案,或与本案无关；
 - C、D 中有且仅有一人作案；
 - 若 D 没参与，则 E 也不可能参与。
- 试编写程序将作案人找出来。

一脸懵逼



解题思路 - 「道」

■ 核心思路

枚举法

■ 比上例复杂的地方

- “一位做了好事” VS 人人皆可犯案
- “是”、“不是” VS “同案犯”、“有且仅有”、“若...则...”

回顾高中「排列组合」知识

- 有六个嫌疑人，人人皆可能犯案，总共有多少种可能的犯案情况？
- 乘法原理
 - 嫌疑人A有两种可能性：犯案、不犯案
 - 嫌疑人B有两种可能性：犯案、不犯案
 -
 - 总共有 $2^6 = 64$ 种可能性
- 如何设计变量？
 - 设计6个01变量：1表示参与；0表示没参与

回顾「逻辑表达式」

- 如何对复杂的“案情”进行建模？
- 现有两队进行篮球比赛，设变量A和B表示A和B两队是否到场，指出下面表达式的含义，判断比赛是否能顺利进行
 - 表达式1： $A \ \&\& \ B$
 - 表达式2： $A \ || \ B$
 - 表达式3： $A \ \&\& \ !B$
 - 表达式4： $!(!B \ || \ !A)$

变量设计

- 六个人每个人都有作案或不作案两种可能,因此有64 (2^6) 种组合,从这些组合中挑出符合条件的作案者。
- 定义6个整型变量,分别表示六个人A - F
- 如何枚举每个人的可能性?
 - 取值 0 表示没有参与作案;
 - 取值 1 表示参与作案

案情建模

- 基于定义的变量，如何做案情分析？
 - A、B至少有一人作案；
 - A、D 不可能是同案犯；
 - A、E、F三人中至少有两人参与作案；
 - 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。

用逻辑表达式建模

对案情逐一建模 (1)

- 已知：变量A和B分别表示A和B作案
 - 怎么表示 “A、B至少有一人作案” ？
 - 表达式：CC1 = A || B
- 已知：变量A和D分别表示A和D作案
 - 怎么表示 “A、D不可能是同案犯” ？
 - 考虑相反事件：A和D是同案犯 → A && D
 - 表达式：CC2 = !(A && D)

对案情逐一建模 (2)

■ 考虑三个变量A、E、F

- 怎么表示 “A、E、F三人中至少有两人参与作案” ？
- 可能有几种情况发生
 - 两个人A和E同时参与作案 $\rightarrow A \ \&\& \ E$
 - 两个人A和F同时参与作案 $\rightarrow A \ \&\& \ F$
 - 两个人E和F同时参与作案 $\rightarrow E \ \&\& \ F$
- 表达式： $CC3 = (A\&\&E) \ || \ (A\&\&F) \ || \ (E\&\&F)$

对案情逐一建模 (3)

■ 考虑两个变量B、C

- 怎么表示 “B、C或同时作案或同时与本案无关” ?
- 可能有几种情况发生
 - 两个人B和C同时参与作案 $\rightarrow B \ \&\& \ C$
 - 两个人A和F与本案无关 $\rightarrow !B \ \&\& \ !C$
- 表达式: $CC4 = (B\&\&C) \ || \ (!B\&\&!C)$

对案情逐一建模 (4)

■ 考虑两个变量C、D

- 怎么表示 “C、D中有且仅有一人作案” ？
- 可能有几种情况发生
 - 只有C作案，D没作案 $\rightarrow C \ \&\& \ !D$
 - 只有D作案，C没作案 $\rightarrow !C \ \&\& \ D$
- 表达式： $CC5 = (C\&\&!D) \ || \ (!C\&\&D)$

对案情逐一建模 (5)

■ 考虑两个变量D、E

- 怎么表示 “如果D没有参与作案，则E也不可能参与作案” ？
- 可能有几种情况发生
 - 如果D没有参与作案，E也没作案 $\rightarrow !D \ \&\& \ !E$
 - 如果D参与作案，则E既作案也没作案 $\rightarrow D$
- 表达式： $CC6 = (!D \ \&\& \ !E) \ || \ (D)$

案情建模

- 如何表示所有的案情分析都成立

$$CC1+CC2+\dots+CC6==6$$

- 与“谁做了好事”案例的区别
 - 简单的关系表达式：“=”与“!=”
 - 复杂的逻辑表达式
- 相同点：逻辑建模的思路

万事俱备，只欠.....

如何进行枚举呢？

- 方法1：沿用上例的单循环语句
 - `for(k=0; k<4; k++)`
- 方法2：

多重循环！

什么是多重循环？

- 考虑下面的例子：
- 编一个程序输出所有的排列数
 - 000000
 - 000001
 - 000010
 - 000011
 -
 - 111111

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);

真

CC1+CC2+CC3+CC4+CC5+CC6==6

假

输出

思路小节

- 枚举思想
 - 完备考虑所有可能性(体现在代码中)
 - 逐一处理 (使用循环语句)
- 自然语言描述-->数学语言描述
 - 以实现其 “可计算性”
 - 关系表达式与逻辑表达式
- 数学语言表达-->程序语言表达
 - 满足其 “可行性” ,变成可执行的程序

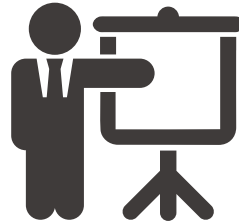
编程思考题

- 假设有8间教室，容量分别为C1 – C8。现考虑四个班级，人数分别为N1 – N4。如果班级人数小于等于教室容量，就可以把教室分配给该班级，共有多少种可行的分配方案
- 编写一个程序，输出下面的内容

A
ABA
ABCBA
ABCDcba
ABCDEDcba



中國人民大學
RENMIN UNIVERSITY OF CHINA



02. 分支控制语句

如何判断三角形

```
int main() {
    int a, b, c;
    double p, area;
    scanf("%d%d%d", &a, &b, &c);
    if (a + b <= c || a + c <= b || b + c <= a)
        printf("不能构成三角形, 请重新输入! ");
    else {
        p = (a + b + c) / 2.0;
        area = sqrt(p * (p - a) * (p - b) * (p - c));
        printf("%.2f\n", area);
    }
    return 0;
}
```

```
int judge(int a,int b,int c)
{
    int t;
    if(a>b)
    {
        t=a;
        a=b;
        b=t;
    }
    if(b>c)
    {
        t=b;
        b=c;
        c=t;
    }

    if(a+b>c){ return 1; }
    else{ return 0; }
}
```


三角形面积

```
int main()
{
    int a, b, c;
    double s, q, sum;
    scanf("%d %d %d", &a, &b, &c);
    while(a<=0||b<=0||c<=0||a+b<=c||a+c<=b||b+c<=a)
    {
        printf("构成不了三角形，请重新输入：");
        scanf("%d %d %d", &a, &b, &c);
    }
    q = (a + b + c) / 2.0;
    sum = q * (q - a) * (q - b) * (q - c);
    s = sqrt(sum);
    printf("%.2lf", s);
    return 0;
}
```

为什么要有分支控制语句

- 一般情况下，程序中的语句是从头到尾按顺序逐条执行的
- 分支计算：有些情况下，不同的条件对应着不同的计算过程，需要根据条件来决定程序的执行步骤
- 举例：输出一个月有几天
 - 一、三、五、七、八、十、十二月：31天
 - 二、四、六、十一月：30天
 - 二月：28天（非闰年）、29天（闰年）

条件语句

- 分支与一定的条件密切相关
 - 在某种条件得到满足的时候执行一种操作在该条件不满足的时候执行另一种操作
- 如何表示“条件”？

逻辑表达式

- 逻辑表达式可以是一个简单的关系表达式
- 也可是逻辑运算符连接的多个关系表达式

关系表达式

- 用**关系运算符**将两个表达式连接起来的式子，称关系表达式
- 关系表达式的值是一个逻辑值，即“真”或“假”（实现上判断是否为0）

运算符	说明	
>	大于	优先级相同（高）
>=	大于等于	
<	小于	
<=	小于等于	
=	等于	优先级相同（低）
!=	不等于	

逻辑运算

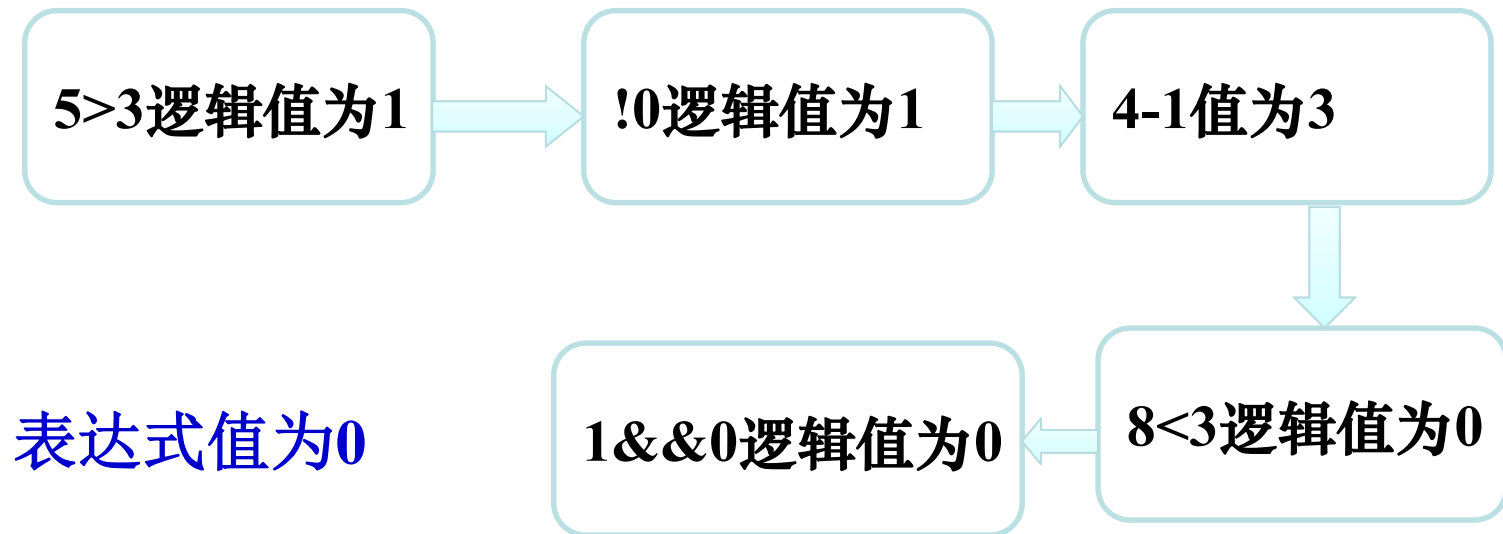
- 逻辑运算符如下：

运算符	说明
!	逻辑非 (NOT)
&&	逻辑与 (AND)
	逻辑或 (OR)

- !为单目运算符，&&和||为双目运算符，结合方向为从右至左。

逻辑表达式

- 用逻辑运算符将关系表达式或逻辑量连接起来的式子
- 求解： $5>3\&\&8<4-!0$ 的值



练习题 (1)

■ 1. 判断下面逻辑表达式的值

- `100 > 3 && 'a' > 'c'`
- `100 > 3 || 'a' > 'c'`
- `!(100 > 3)`

■ 2. 写出下述条件的逻辑表达式

- number is equal to or greater than 90 but smaller than 100.
- ch is not a q or a k character.
- number is between 1 and 9 (including the end values) but is not a 5.
- number is not between 1 and 9.

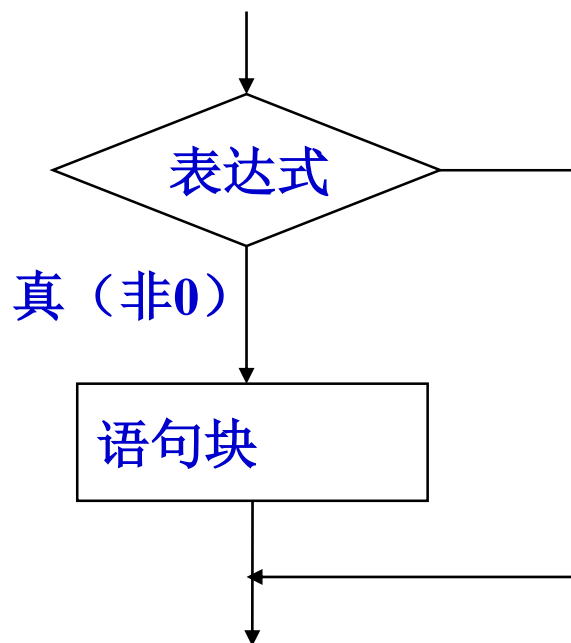
练习题 (2)

■ 判断下面表达式的值

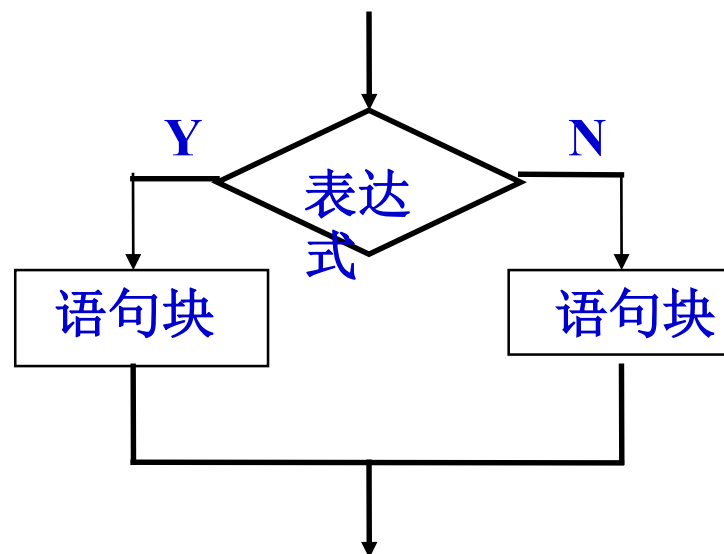
- $(5 > 2) + 2$
- $3 + 4 > 2 \ \&\& \ 3 < 2$
- $x \geq y \ || \ y > x$
- $d = 5 + (6 > 2)$
- $'X' > 'T' \ ? \ 10 : 5$
- $x > y \ ? \ y > x : x > y$

如何表示分支结构 (1)

■ 使用流程图表示

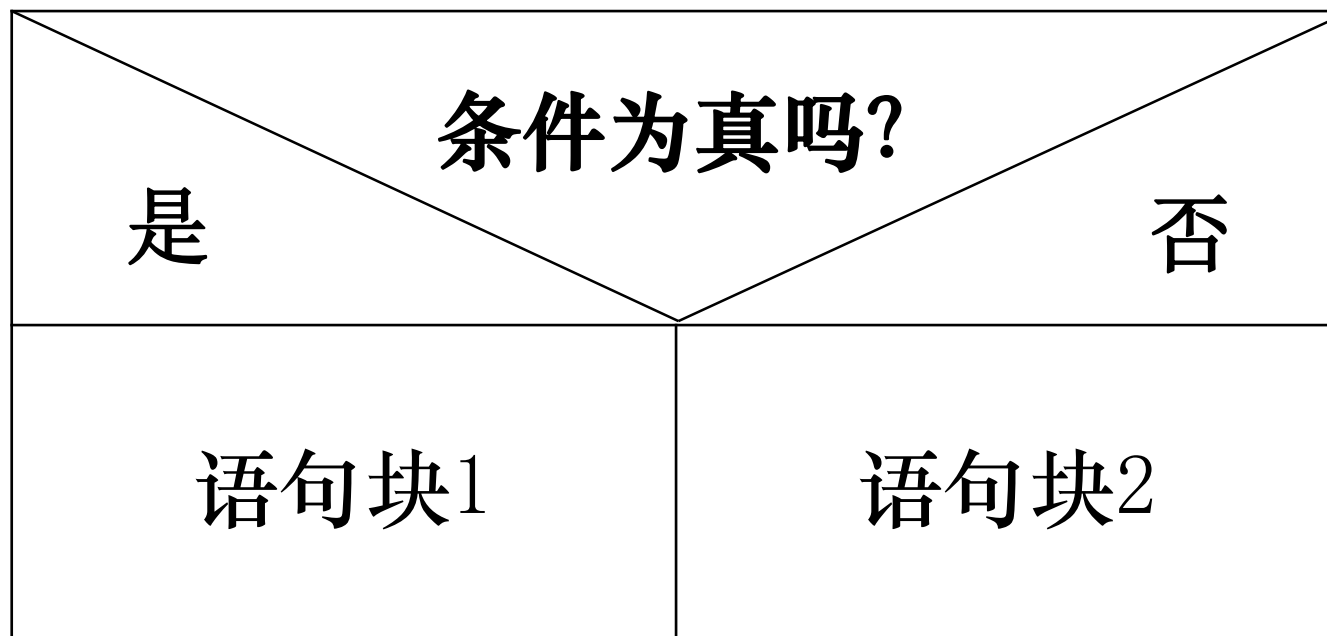


假
(0)



如何表示分支结构 (2)

- 使用N-S图表示



IF语句的格式

■ 情况1:

```
if ( 表达式 )  
    语句 1;
```

如果表达式为真，执行语句 1；否则什么都不做。

■ 情况2:

```
if ( 表达式 ) {  
    语句块 1;  
}
```

如果表达式为真，执行语句块 1；否则什么都不做。

- 情况3:

```
if ( 表达式 )
```

```
    语句 1;
```

```
else
```

```
    语句 2;
```

- 情况4:

```
if ( 表达式 ) {
```

```
    语句块 1;
```

```
}
```

```
else {
```

```
    语句块 2;
```

```
}
```

IF语句的嵌套和级联

```
if (<条件表达式1>) <语句1>  
  
else if (<条件表达式2>) <语句2>  
  
else if (<条件表达式3>) <语句3>  
  
.....  
  
else if (<条件表达式n>) <语句n>  
  
else <语句n+1>
```

演示环节

分段函数值计算

见示例程序func. cpp

思考：画出该程序的N-S图

二元一次方程

```
#include <stdio.h>
```

```
#include <math.h>
```

```
int main()
```

```
{
```

```
    double a,b,c,disc,x1,x2,realpart, imagpart;
```

```
    scanf("%lf,%lf,%lf",&a,&b,&c);
```

```
    printf("The equation ");
```

```
    if( fabs(a) <= 1e-6 )
```

```
        printf("is not a quadratic\n");
```

实型不能用if (a==0)

else

```
{disc=b*b-4*a*c;
```

```
if(fabs(disc)<=1e-6) 不能用if (disc==0)
```

```
printf("has two equal roots:%8.4f\n", -b/(2*a));
```

else

```
if(disc>1e-6)
```

```
{x1=(-b+sqrt(disc))/(2*a);
```

```
x2=(-b-sqrt(disc))/(2*a);
```

```
printf("has distinct real roots:%8.4f and %8.4f\n",x1,x2);
```

```
}
```

else


```

{ realpart=-b/(2*a);
  imagpart=sqrt(-disc)/(2*a);
  printf(" has complex roots:\n");
  printf("%8.4f+%8.4fi\n "
          ,realpart,imagpart);
  printf("%8.4f-%8.4fi\n",
          realpart,imagpart);
}
}
return 0;
}

```

```
1,2,1
```

```
The equation has two equal roots: -1.0000
```

条件运算符

■ 条件表达式格式：

- 表达式1 ? 表达式2 : 表达式3
- 三目运算
- 如果表达式1成立，则条件表达式值为表达式2的值，否则为表达式3的值
- 例如： `max = (a > b) ? a : b ;`
 `if (a > b)`
 `max = a;`
 `else`
 `max = b;`
- 思考：将上述func.cpp的例子替换成三目运算符

switch语句

■ 格式:

switch (表达式)

```
{  
    case 常量1: 语句1;  
    case 常量2: 语句2;  
        ⋮  
    case 常量n: 语句n;  
    default:    语句n+1;  
}
```

● 执行过程:

- 求出“表达式”的值，并执行与其相匹“常量”对应的语句。
- 跟随case的语句将顺序执行，直到遇到break语句或case块结束；
- 如果没有与之相匹的“常量”就执行default块，若default不存在，就退出switch语句。

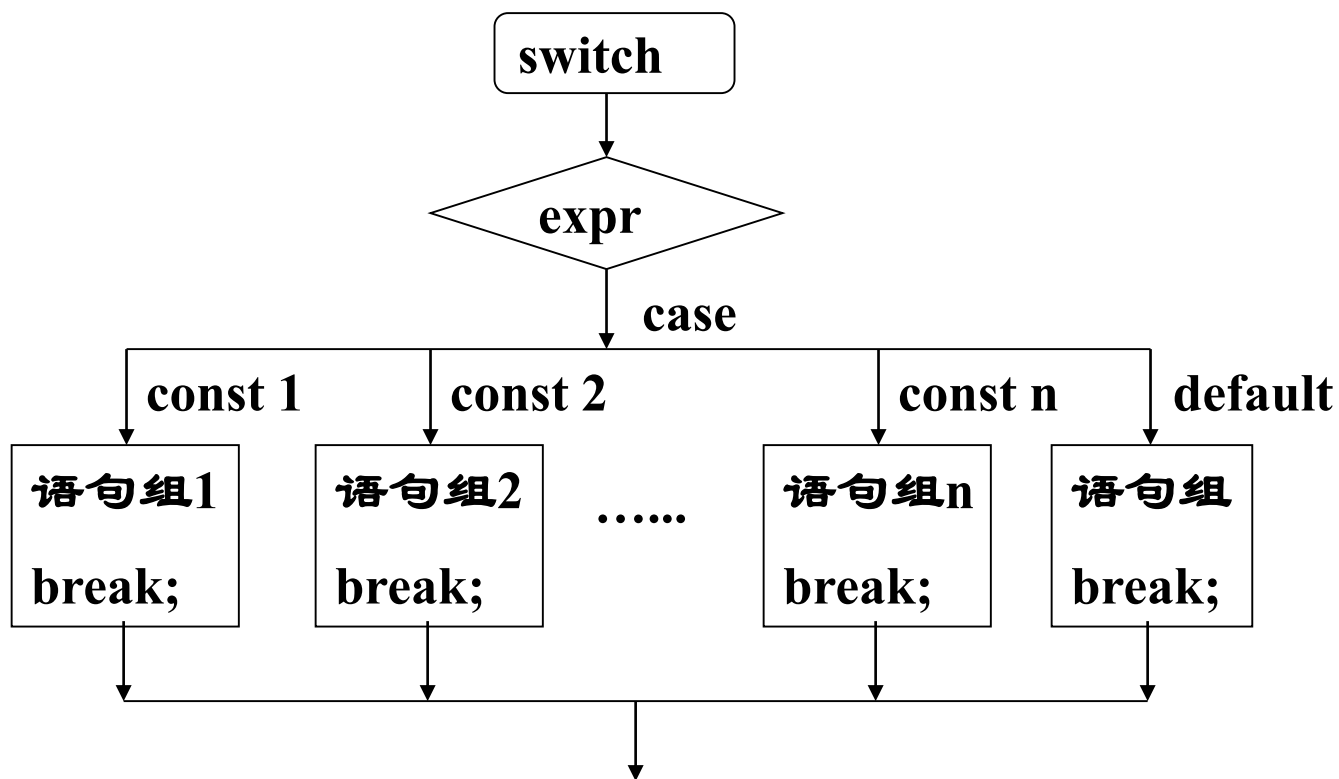
switch语句注意事项:

- switch中表达式可以是整型或字符型, 也可以是枚举型。
- 可以省略一些case和default。
- 每个case或default后的语句可以是语句块, 但不需要使用 “{” 和 “}” 括起来。
- case下也可以没有语句。

switch结构的应用

- switch结构与else if结构是多分支选择的两种形式。它们的应用环境不同：
 - else if 用于对多条件并列测试，从中取一的情况；switch 结构用于单条件测试，从其多种结果中取一种的情况。
 - else if的条件测试比switch复杂、功能更强；switch结构更为清晰、简单。

switch结构:



运费问题

例 运输公司对用户计算运输费用。路程(s km) 越远, 每吨·千米运费越低。

■ 标准如下:

$s < 250$	没有折扣
$250 \leq s < 500$	2%折扣
$500 \leq s < 1000$	5%折扣
$1000 \leq s < 2000$	8%折扣
$2000 \leq s < 3000$	10%折扣
$3000 \leq s$	15%折扣

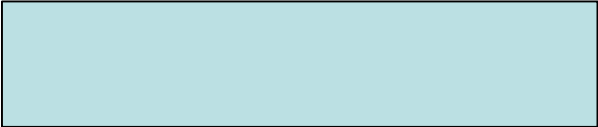
■ 解题思路：

- 设每吨每千米货物的基本运费为 p ，货物重为 w ，距离为 s ，折扣为 d
- 总运费 f 的计算公式为 $f=p \times w \times s \times (1-d)$

■ 折扣的变化规律：

- 折扣的“变化点”都是250的倍数
- 在横轴上加一种坐标 c ， c 的值为 $s/250$
- c 代表250的倍数
- 当 $c < 1$ 时，表示 $s < 250$ ，无折扣
- $1 \leq c < 2$ 时，表示 $250 \leq s < 500$ ，折扣 $d=2\%$
- $2 \leq c < 4$ 时， $d=5\%$ ； $4 \leq c < 8$ 时， $d=8\%$ ；
 $8 \leq c < 12$ 时， $d=10\%$ ； $c \geq 12$ 时， $d=15\%$


```

#include <stdio.h>
int main()
{
    int c,s;
    float p,w,d,f;
    scanf("%f,%f,%d",&p,&w,&s);
    if(s>=3000) c=12;
    else      c=s/250;
    
    f = p * w * s * (1 - d / 100);
    printf( "freight=%10.2f\n" ,f);
    return 0;
}

```

switch(c)

```

{ case 0:  d=0; break;
  case 1:  d=2; break;
  case 2:
  case 3:  d=5; break;
  case 4:
  case 5:
  case 6:
  case 7:  d=8; break;
  case 8: case 9: case 10:
  case 11: d=10; break;
  case 12: d=15; break;
}

```

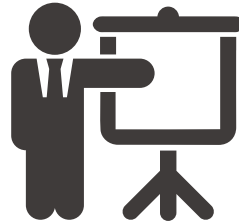
switch结构的应用

10086按键分布说明		
接入热线	一级菜单	二级菜单
10086	1. 话费、套餐使用情况、归属地及积分查询 (自动播报话费余额)	1. 话费查询
		2. 套餐及上网流量查询
		3. 归属地查询
		4. 账单查询
		5. 积分查询
		6. 查询其他手机信息
	2. 最新优惠活动	
	3. 停复机、手机上网及密码服务	1. 停复机
		2. 手机上网流量查询
		3. 手机上网套餐办理
		4. 密码修改
		5. 密码重置
		6. 他机办理
	4. 增值业务查询与退订	
	5. 无线宽带、家庭业务	1. 无线宽带
		2. 家庭业务
	8. 集团业务	1. 话费、套餐等信息查询
		2. 故障申告
		3. 业务咨询和办理
		0. 人工服务
	0. 人工服务	

- 请使用switch结构，
写一个10086电话语音服务的按键信息
- 接收用户按键，打印
出提示语句



中國人民大學
RENMIN UNIVERSITY OF CHINA



03. 循环控制语句

如何判断三角形

```
int main() {
    int a, b, c, g;
    double p, s;
    scanf("%d %d %d", &a, &b, &c);
    if (a + b <= c || b + c <= a || a + c <= b) {
        g = 1;
        for (g = 1; g != 0; g = g) {
            printf("不能构成三角形, 请重新输入\n");
            scanf("%d %d %d", &a, &b, &c);
            if (a + b <= c || b + c <= a || a + c <= b)
                g = 1;
            else {
                g = 0;
                p = (a + b + c) / 2.0;
                s = sqrt(p * (p - a) * (p - b) * (p - c));
                printf("%.2lf", s);
            }
        }
    }
}
```

演示环节

输入多个数字求和

见示例程序summing.c

表达循环的关键词 - while

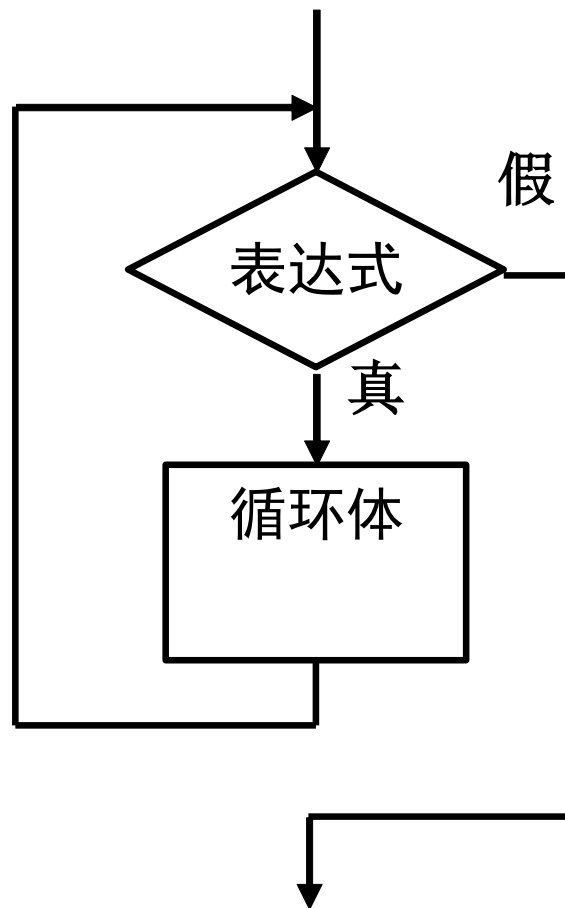
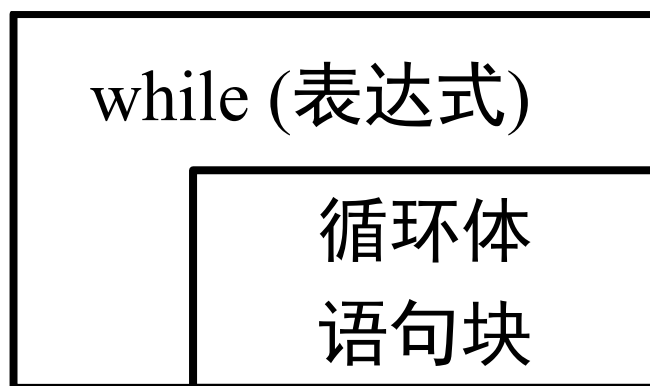
- 一般形式:

while (表达式)

{

语句块; (循环体)

}



思考：画出summing程序的流程图和N-S图

如何终止循环?

- 下面语句会产生什么结果?

```
index = 1;  
while (index < 5)  
    printf("Good morning!\n");
```

- 怎么终止循环?

- while(index++ < 5)
- break

死循环

思考题 (1)

■ 程序输出结果

```
#include <stdio.h>

int main(void) {
    int n = 0;
    while (n < 3)
        {printf("n is %d\n", n);
         n++;}
    printf("That's all this program does\n");
    return 0;
}
```


思考题 (2)

- 程序输出结果?

```
#include <stdio.h>

int main(void) {
    int n = 0;
    while (n++ < 3)    ;
        printf("n is %d\n", n);
    printf( "That 's all this program does.\n" );
    return 0;
}
```

练习：求两个整数的最小公倍数

- 分析：

- 假定有 x, y 且 $x > y$, 设最小公倍数为 z , 则：

- 1) z 一定会 $\geq x$

- 2) $z = kx$, $k = 1, 2, \dots$

- 3) z 一定会被 y 整除

- 用两个最简单的数试一下就可以找到算法，比如 $x=5$, $y=3$, 执行如下操作

```
#include <stdio.h>
```

```
int main(){
```

```
    int x=0, y=0, z=0, w=0; // 整型变量
```

```
    printf( "请输入两个整数，用空格隔开： " ); // 提示信息
```

```
    scanf( "%d" , &x);           // 键盘输入整数 x
```

```
    scanf( "%d" , &y);           // 键盘输入整数 y
```

```
    if ( x < y ) { // 让 x 表示两者中的大数
```

```
        w = x;  x = y;  y = w;
```

```
    }
```

```
    z = x; // 将一个大数赋给 z
```

```
    while ( z % y != 0 ) // 当z不能被y整除时，让z累加x
```

```
        z = z + x;
```

```
    printf( "最小公倍数为%d" , z); // 输出最小公倍数
```

```
    return 0;
```

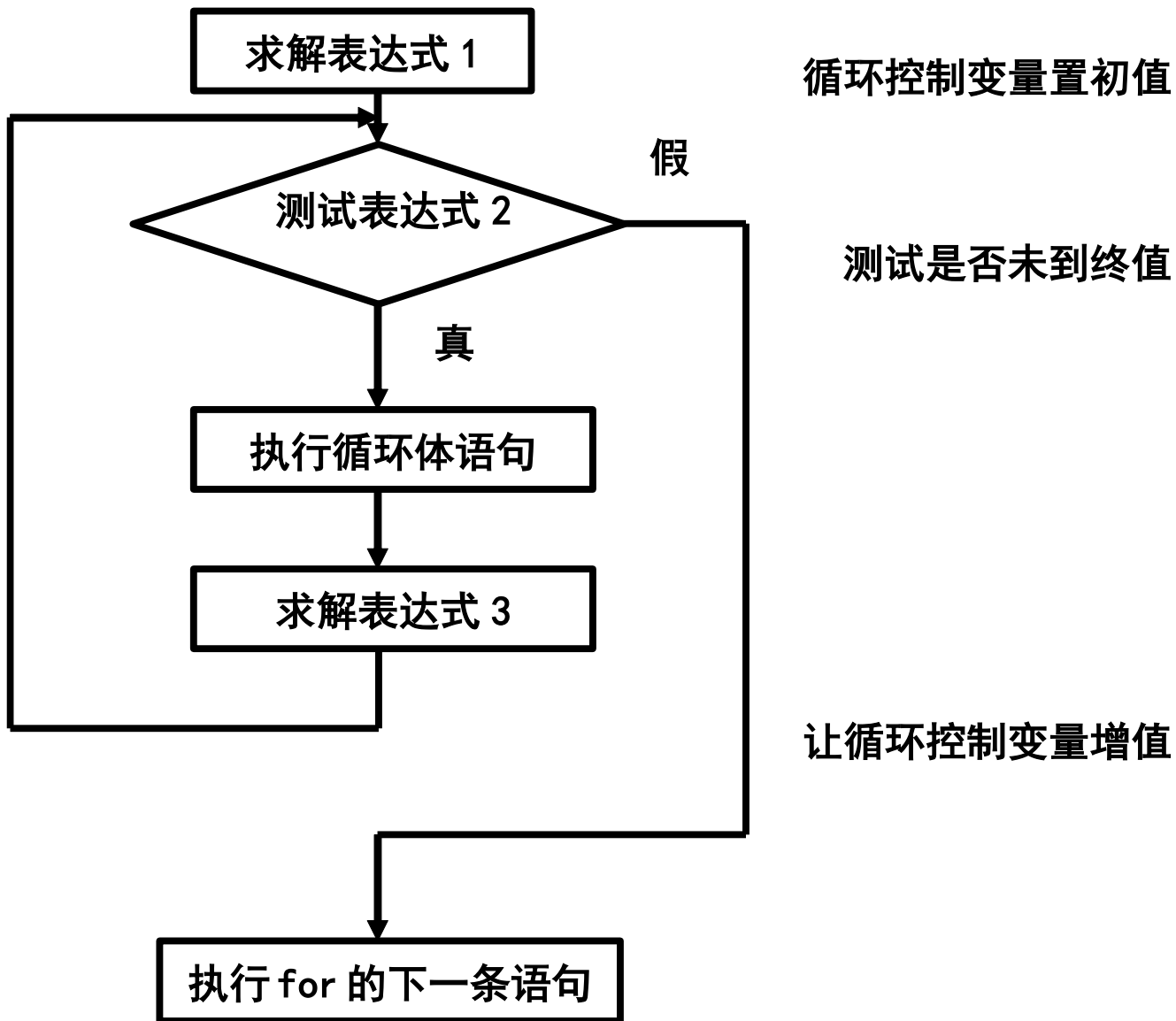
```
}
```

计数型循环关键词 - for

- 计数是程序中用得最多的一种，它发挥了计算机擅长重复运算的特点
- for：计数型循环的标识符

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

圆括号括起的是三个表达式。其下的大括号括起的部分是循环体



图：for 循环结构图

使用for语句

■ 倒计时器

```
#include <stdio.h>
```

```
int main(void) {
```

```
    int secs;
```

```
    for (secs = 5; secs > 0; secs--)
```

```
        printf("%d seconds!\n", secs);
```

```
    printf("We have ignition!\n");
```

```
    return 0;
```

```
}
```

思考1： 如果做到两秒两秒倒计时？

思考2： 如何输出小写字母表？

课后练习： 尝试for的表达式不同形式

数列求和

■ $1 + 1/2 + 1/4 + 1/8 + 1/16 + \dots$

```
int main(void) {
    int n;
    double time, x;
    int limit;
    printf("Enter number of terms you want: ");
    scanf("%d", &limit);
    time = 0;
    for (x = 1, n = 1; n <= limit; n ++, x *= 2.0) {
        time += 1.0 / x;
        printf("time = %f when terms = %d.\n", time, n);
    }
    return 0;
}
```

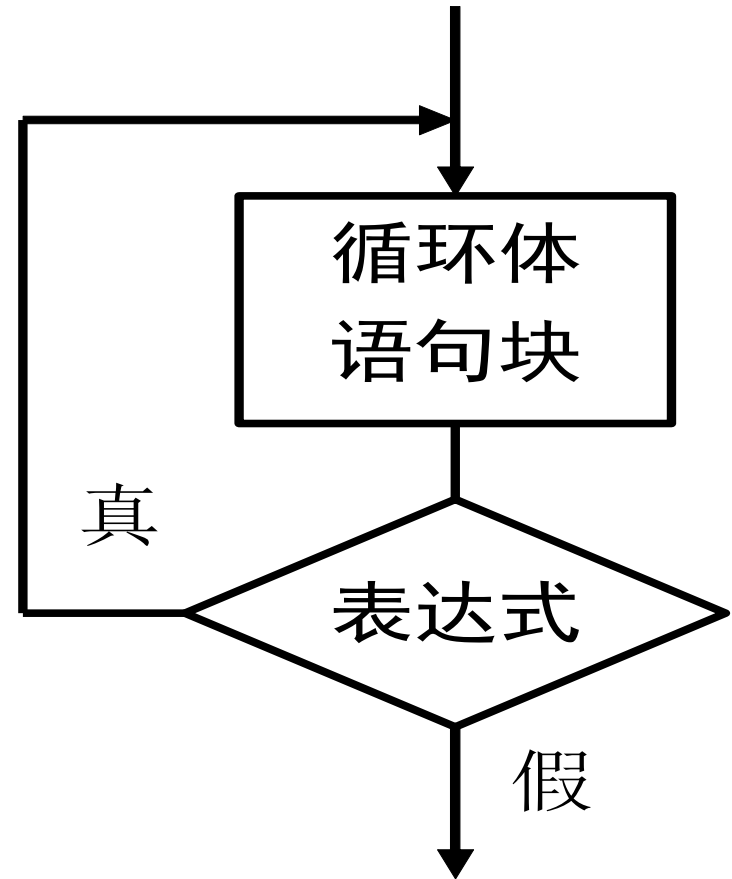
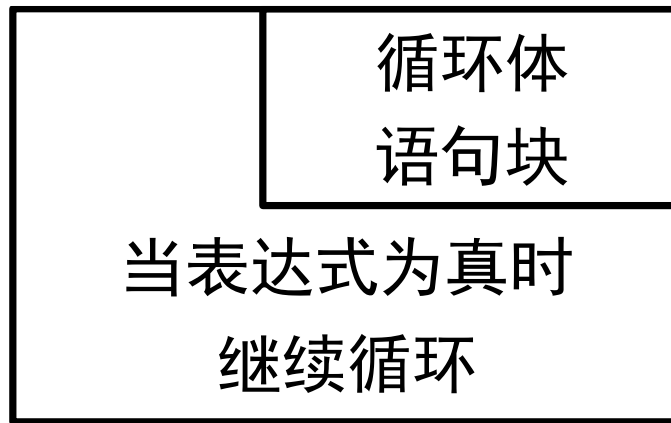
用for 语句实现循环

```
for(i=0; (c=getchar())!='\n'; i+=c)
    ;
```

```
for(    ; (c=getchar())!='\n';    )
    printf("%c", c);
```


退出条件循环 – do ... while

```
do {  
    循环体语句块;  
} while ( 表达式 )
```



直到表达式为假时才退出循环，所以循环体至少执行一次。

■ 猜数字：

➤ 让用户不断地猜一个数字，直到猜对为止

➤ 退出条件循环

```
int num = 31, guess = 0;  
do {  
    printf ("Guess a number: ");  
    scanf ("%d", &guess);  
} while (guess != num);
```

➤ 准入条件循环

```
printf ("Guess a number: ");  
scanf ("%d", &guess);  
while (guess != num) {  
    printf ("Guess a number: ");  
    scanf ("%d", &guess);  
}
```

如何选择循环表达式?

- 准入条件循环 vs. 退出条件循环
- 更为推荐使用准入条件循环
 - 先进行条件判断更为直观
 - 程序更容易阅读和理解
 - 如果条件不满足，则略去整个循环
- 如何选择for和while
 - 循环涉及变量的初始化与更新
 - 循环涉及其它条件

循环的嵌套

```
int main(void) {  
    const int ROWS = 6;  
    const int CHARS = 6;  
    int row;  
    char ch;  
    for (row = 0; row < ROWS; row++) {  
        for (ch = ('A' + row); ch < ('A' + CHARS); ch++)  
            printf("%c", ch);  
        printf("\n"); }  
    return 0;  
}
```

练习题

- 给定一个int类型的变量value，下面循环的输出是什么？如果是float类型呢？

```
for ( value = 36; value > 0; value /= 21)
    printf("%3d", value);
```

- 如何使用循环语句进行如下输出？

```
$
$$
$$$
$$$$
$$$$$
$$$$$$
$$$$$$$
$$$$$$$$
$$$$$$$$$
```

限定转向语句

- 这一类语句不形成控制结构，只是简单地使流程从其所在处转向另一处。
- 但是，转向也要按照“基本法”——系统事先规定的原则向某一点转移。
- 包括的语句有：
 - break
 - continue
 - return

break语句

- 格式:

- break;

- 功能:

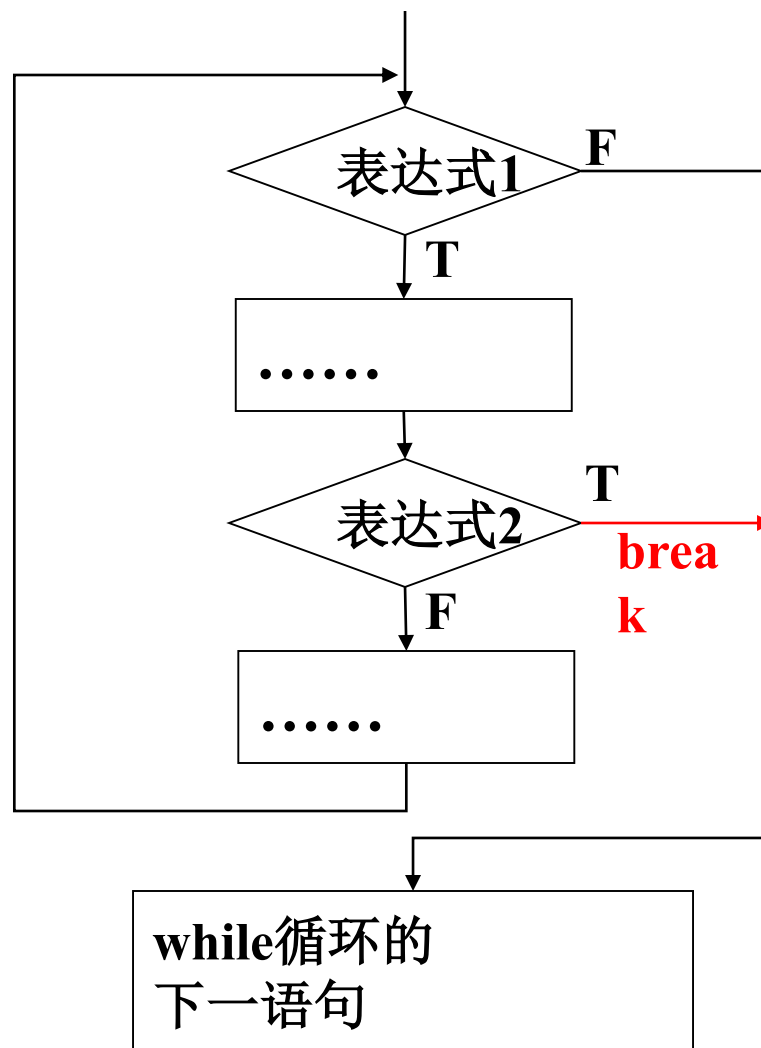
- 结束最近的循环 (do、for和while) 或switch语句, 把流程从该循环的内部跳到循环外部

- 思考:

- 什么是最近的循环语句?

break语句

```
while (表达式1)
{
    .....
    if(表达式2)
        break;
    .....
}
```



捐款

```
#include <stdio.h>
#define SUM 100000
int main()
{ float amount,aver,total;  int i;
  for (i=1,total=0;i<=1000;i++)
  { printf("please enter amount:");
    scanf("%f",&amount);
    total= total+amount;
    if (total>=SUM) break;
  }
  aver=total / i ;
  printf( "num=%d\naver=%10.2f\n ", i, aver);
  return 0;
}
```

continue语句

■ 格式:

- `continue;`

■ 功能:

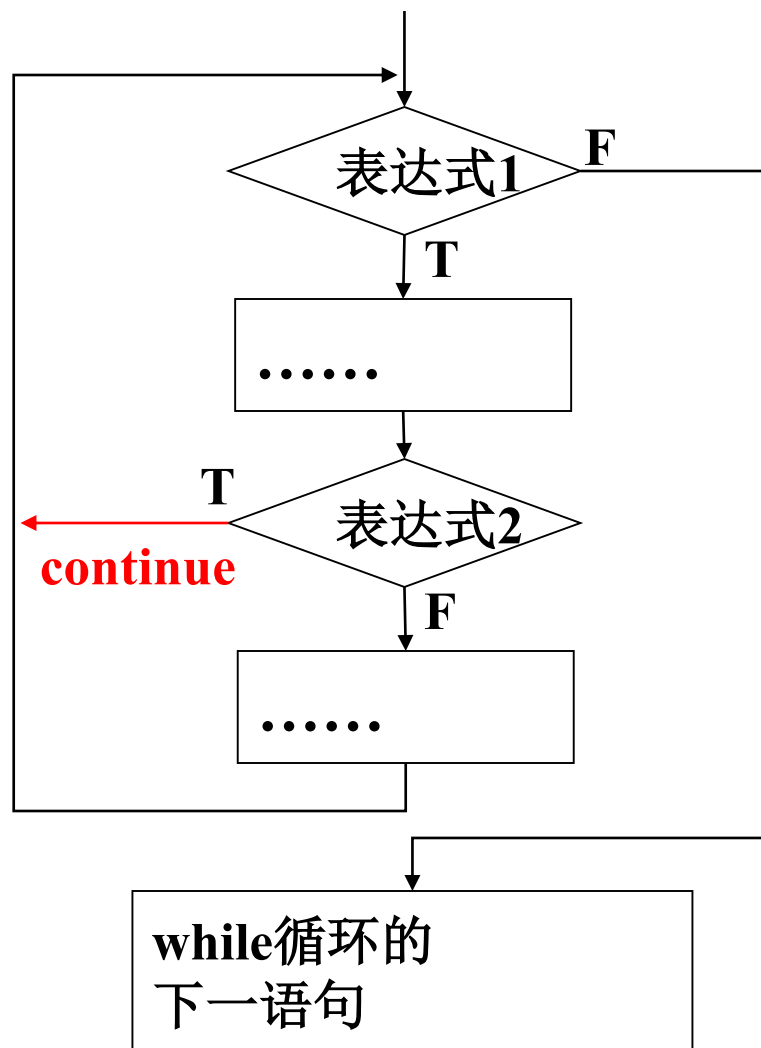
- 把控制转到最近的循环语句头,包括do、for和while

■ 说明:

- 中止本次循环, 即不执行continue后的循环体中的语句。
- continue与break的区别是: continue只结束本次循环, 而不是终止整个循环语句的执行; 而break则是结束整个循环, 不再进行条件判断。
- continue不用在for、do和while以外的其它语句中

continue示意

```
while (表达式1)
{
    .....
    if(表达式2)
        continue;
    .....
}
```



示例

■ 猜数字游戏

- 编写一个猜数字的游戏，由编程人员设定一个100以内的奇数，让用户来猜，每次反馈猜的大了还是小了，直到猜对停止。如果用户输入了偶数，提醒用户输入奇数。

```
int main () {  
    int num = 31, guess = 0;  
    while (1) {  
        printf (" Guess an odd number in [1,99]\n");  
        scanf ("%d", &guess);  
        if (guess % 2 == 0) {  
            printf ("Please guess an ODD number\n");  
            continue; }  
        if (guess > num) printf ( "Too big\n" );  
        else if (guess < num) printf (" Too small\n ");  
        else {  
            printf ( "Bingo! The number is %d" , num);  
            break;  
        }  
    }  
    return 0;  
}
```

思考题

- 编写一个程序，输出下面的内容

A
ABA
ABCBA
ABCD CBA
ABCDEDCBA

- 如果让用户循环输入一个行数（1-26之间），按照上述规律输出字母金字塔，直到用户输入q则退出程序。应该怎么编程？

思考题 (1)

```
for (int row = 0; row < line_num; row ++ ) {  
    //print the spaces in the left  
    for (int i = 0; i < line_num - row - 1; i ++ )  
        printf(" ");  
    // print the letters in the middle  
    for (int i = 0; i < row + 1; i ++ )  
        printf("%c", 'A' + i);  
    for (int i = row - 1; i >= 0; i -- )  
        printf("%c", 'A' + i);  
    printf("\n");  
}
```

演示环节

循环输入数字金字塔

参见pyramid. cpp

练习题：简单统计

- 读入一行字符，并统计其中元音字母的相对频率
 - 示例输入：I am a little teapot
 - 示例输出：
 - 15.0%,10.0%,10.0%,5.0%,0.0%
- 示例程序：参见vowel.cpp

迭代

- 迭代是一个不断用新值取代变量旧值，或由旧值递推出变量的新值的过程。
- 例6 兔子繁殖问题。

有一对兔子，从出生后第3个月起每个月都生一对兔子，小兔子长到第三个月后每个月又生一对兔子，假如兔子都不死，问每个月的兔子总数为多少对？

Fibonacci数列:

$$\text{fib}_1 = \text{fib}_2 = 1$$

$$\text{fib}_n = \text{fib}_{n-1} + \text{fib}_{n-2} \quad (n \geq 3)$$

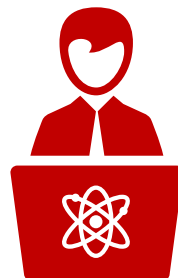
迭代的实例

■ 进制转换

- 读入一串二进制数，将它转换成对应的10进制数
- 使用迭代法，参见实例程序num_sys.cpp



中國人民大學
RENMIN UNIVERSITY OF CHINA



谢谢大家!

