Xidian University

C语言程序设计

Lec 3 程序流程控制

主要内容

- 语句与复合结构
- 再论变量
- 流程控制
  - 关系表达式
  - 逻辑表达式
  - 条件语句
  - 循环语句

3.1 语句与复合结构

语句

- 分号结束的一个字符序列构成语句
  - 变量声明语句： `double a,b,c;`
  - 表达式语句： `h=a*sin(3.1416*c/180);`
  - 函数调用语句： `printf("%f",s);`
  - return 语句：`return 0;`
  - ...

复合结构

- 多个语句由一对大括号包围起来构成复合结构（demo_3_fuhe. c）

```
int main ()
{
    double s=123.5;
    printf("s=%f!\n",s);
    return 0;
}
```

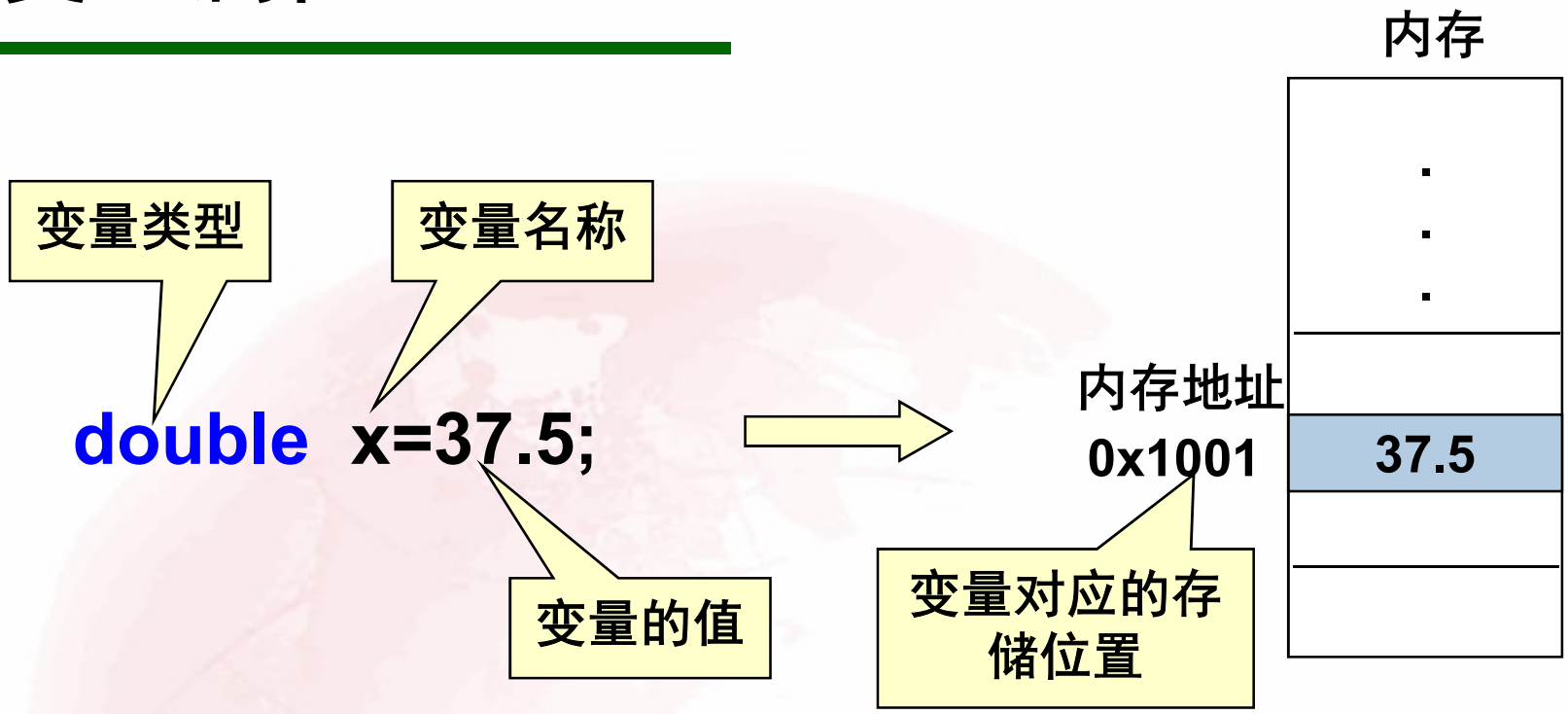
```
int main ()
{
    int i=0, sum=0,fac=1;
    for(i=1; i<=10; i++)
    {
        sum = sum + i;
        fac = fac * i;
    }
}
```

3.2 再论变量

变量属性

- 变量：用于存储程序的输入数据或计算结果的存储单元，一个变量具有以下4个属性
  - 变量的名称
  - 变量的类型
  - 变量的值
  - 变量的存储位置

变量属性



mov 0x1001, 37.5

变量对应的存储位置（内存地址）可以通过取地址运算符&得到：

&x;

变量声明（demo_3_var.c）

- 变量声明：给存储单元定义一个名称及类型，便于程序中引用
  - 变量必须先声明后使用
  - 变量名必须是合法标识符
  - 变量必须有确定数据类型
  - 可以在一条语句中定义多个同类型变量，变量之间用逗号分隔

```
int a, b, c;
double h, s;
```

变量声明（demo_3_var.c）

- 变量声明：给存储单元定义一个名称及类型，便于程序中引用
  - 在任何一个复合结构中都可以定义变量，但变量定义必须在该复合结构中的其他语句之前（注：与编译器采用的C语言标准有关，Dev C++支持C99标准，因此可以将变量声明放在语句之后；但VC6不支持C99，因此变量声明必须放在语句之前）

变量赋值

- 赋值操作——改变变量当前的值
- 赋值表达式：由赋值操作符“=”构成的表达式
- 赋值语句：赋值表达式加上分号

```
a=3.5
b=4.72
h=a*sin(c*3.1416/180)
s=0.5*b*h
```

赋值表达式

```
a=3.5;
b=4.72;
h=a*sin(c*3.1416/180);
s=0.5*b*h;
```

赋值语句

变量赋值（demo_3_assign.c）

- 赋值表达式的值就是等号右边的表达式的值
- 赋值运算符的优先级低于算术运算符
- 赋值运算符两边类型不同时将发生类型转换

```
#include <stdio.h>
int main(){
    int x, y;
    double z;
    y = (x=5) + 8;
    printf("x=%d, y=%d\n",x,y);
    y = x = 6 + 8;
   printf("x=%d, y=%d\n",x,y);
    z=(x+y+1)/2;
   printf("z=%f\n",z);
    return 0;
}
```

变量取值

- 取值操作——获得变量当前的值
  - 方法：直接引用变量名称
  - h=a*sin(c*3.1416/180)
  - s=0.5*b*h
  - printf("s=%d\n", s);

变量的相关问题

- 定义变量时初始化
  - int x=5, y=20;
- 赋值符号与数学意义上的等号
  - x = x + 1，表示将x的值增加1（等效于X++）

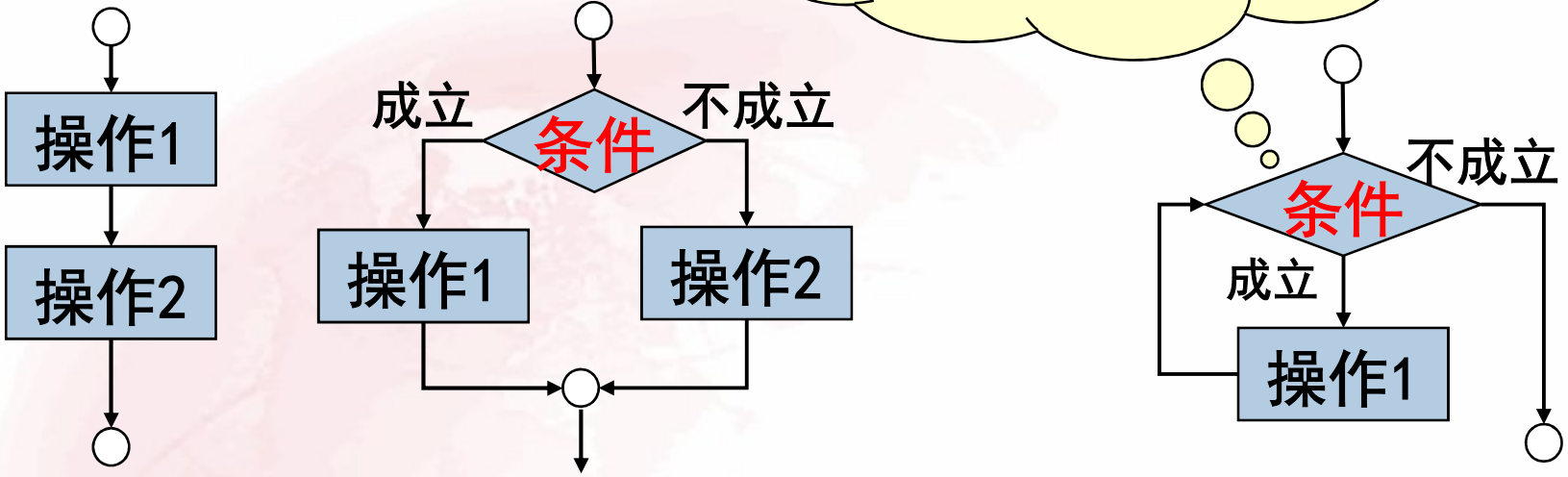
3.3 程序流程控制

引言

- 例1：写程序计算 $ax^2+bx+c=0$的两个实根，如果两个实根相同只输出一个，如果不存在实根输出"no real root"
  - 如何根据b^2-4ac的值来选择输出？
- 例2：计算 $\sum_{n=1}^{100}\sin(1/n)=?$
  - 重复做100次或更多次相同的事情如何解决？

三种基本流程控制模式

1. 条件是什么？
2. 条件是否成立如何判断？



顺序执行　　选择执行　　重复执行

三种模式都只有一个起点和一个终点，因此可以当作一个整体结构嵌入到其他流程模式中

为了控制程序流程需要用到的表达式

- 关系表达式：谁大谁小问题，是否成立问题
- 条件表达式：取舍问题　a=5>3?5:3
- 逻辑表达式：并且、或者、否定等问题

为了控制程序流程需要用到的表达式

- 关系表达式：谁大谁小问题，是否成立问题
- 条件表达式：取舍问题
- 逻辑表达式：并且、或者、否定等问题

```
if ( 3>1 ) {
        printf("It's true!");
}
else
{
        printf("It's wrong!");
}
```

```
if ( x==1 )
{ ……
}
else
{……
}
```

关系表达式

- 由关系运算符和数据构成的表达式，用来确定两个数据之间的关系

关系运算符	含　　义	示例
<	小于	x < 0
>	大于	x > 1
<=	小于等于	x <= 5
>=	大于等于	y >= 2
==	等于	y == 0 （y=0）
!=	不等于	y !=0

关系表达式

- 关系表达式的结果是一个逻辑值，其值取决于关系是否成立
  - 关系成立，表达式结果为逻辑“真”（true）
  - 关系不成立，表达式结果为逻辑“假”（false）
  - C语言没有专门的逻辑值类型，用数值1表示逻辑“真”，数值0表示逻辑“假”
  - 任何基本类型均可当作逻辑值使用，非0表示逻辑“真”，0表示逻辑“假”：

```
if(a){...}; 等效于：if(a!=0){...};
if(!a){...}; 等效于：if(a==0){...};
```

关系表达式示例

（demo_3_relation.c）

x	power	y	item	MIN_ITEM	gender	num
-5	1024	7	1.5	-999.0	'M'	999

运算符	关系表达式	含　　义	表达式的值
<=	x < = 0	x小于或等于0	1
<	power < 1024	power小于1024	0
>=	x >= y	x大于或等于y	0
>	item > MIN_ITEM	item大于MIN_ITEM	1
==	gender == 'M'	gender等于'M'	1
!=	num!=1000	num不等于1000	1

为了控制程序流程需要用到的表达式

- 关系表达式：谁大谁小问题，是否成立问题
- 条件表达式：取舍问题
- 逻辑表达式：并且、或者、否定等问题

条件表达式

- C语言中唯一一个三元运算符
  - 表达式1 ？ 表达式2 ： 表达式3
  - 首先计算表达式 1；如果这个表达式的值非 0（即，条件成立），那么接着计算表达式 2，并用它的值作为整个条件表达式的值；如果条件不成立（表达式 1 的值是 0），就计算表达式 3，并用它的值作为整个条件表达式的值。
  - 特别注意：在表达式 1非 0时不计算表达式 3；在表达式1值为 0时不计算表达式 2。

条件表达式示例(demo_3_condition.c)

```
//示例1：求两个整数的较大值或较小值的方法

int a=4, b=6, max, min;
max = a>b ? a : b;     //max=6
min = a<b ? a : b;     //min=4
printf( “max=%d, min=%d\n”, a, b);
```

```
//示例2：在表达式1的值非0时不计算表达式 3；
//在表达式1的值为0时不计算表达式 2

int a=4, b=6, c;
c=a>3 ? (a=a+1) : (b=b+1);
printf(“a=%d, b=%d, c=%d\n", a, b, c);
```

为了控制程序流程需要用到的表达式

- 关系表达式：谁大谁小问题，是否成立问题
- 条件表达式：取舍问题
- 逻辑表达式：并且、或者、否定等问题

```
if ( x>1 &&  x<100 ) {
        printf("It's true!");
}
else
{
        printf("It's wrong!");
}
```

逻辑表达式

- 用逻辑运算符连接多个关系表达式，用于描述多个关系的复杂组合
  - 例1：判断x是否在区间［ 3，5 ）之内
    - 即x大于等于3，并且x小于5
  - 例2：判断某年是否是闰年：
    - 年份能够被400整除
    - 或者年份能够被4整除并且不能被100整除
  - 例3：判断x不在区间［ 3，5 ）之内
    - 例1的条件取反

逻辑运算符

- C语言的3种逻辑运算符

X与区间 [3, 5)的关系

逻辑运算符	含　　义	示例
&&	并且	x >=3&&x<5
||	或者	x<3 ||x>=5
!	非	!(x >=3&&x<5)

Wrong!: 3<=x<5

逻辑表达式

- 逻辑与（&&）：　表达式1 && 表达式2
  - 只有两个表达式都非0时结果为1，否则为0

操作数1	操作数2	操作数1 && 操作数2
非零（true）	非零（true）	1（true）
非零（true）	0（false）	0（false）
0（false）	非零（true）	0（false）
0（false）	0（false）	0（false）

计算方式：先求表达式 1；若得到0则不计算表达式2，以0作为整个表达式的结果；否则（表达式 1非 0）就计算表达式 2，如果它为0则整个表达式以0为结果，否则以1为结果。

逻辑表达式

- 逻辑或（||）：　表达式1 || 表达式2
  - 两个表达式的值都为0时结果为0，否则为1

操作数1	操作数2	操作数1 || 操作数2
非零（true）	非零（true）	1（true）
非零（true）	0（false）	1（true）
0（false）	非零（true）	1（true）
0（false）	0（false）	0（false）

计算方式：先求表达式 1；若得到非0则不计算表达式2，以 1作为整个表达式的结果；否则（当表达式 1值是 0时）计算表达式2，如果它为 0则整个表达式以 0为结果，否则以 1为结果。

逻辑表达式

- 逻辑非（!）：　　！表达式1
  - 把表达式的值看作逻辑值，以该值的否定作为结果

操作数1	!操作数1
非零（true）	0（false）
0（false）	1（true）

计算方式：如果表达式的值非 1，则结果为 0；如果表达式值是 0则结果为 1。

运算符优先级

运算符	优先级
函数调用	高
! + - & （一元运算符）	
* / %	
+ -	
< <= >= >	
== !=	
&&	
||	
=（赋值运算符）	低

逻辑表达式示例

- 例1：判断x是否在区间［ 3，5 ）之内（demo_3_range.c）
  - 即x大于等于3，并且x小于5

```
x >= 3 && x < 5
```

```
3<= x < 5 是否正确？
```

逻辑表达式示例

- 例2：判断x是否在区间［ 3，5 ）之外
  - 即x小于3，或者x大于等于5

`x < 3 || x >= 5`　　或者　　`!(x >=3 && x < 5)`

（demo_3_range2.c）

逻辑表达式示例

- 例3：判断year表示的年份是不是闰年
  - 年份能够被400整除，或者年份能够被4整除并且不能被100整除

```
year%400==0 || (year%4==0 && year%100!=0)
```

（demo_3_leap year.c）

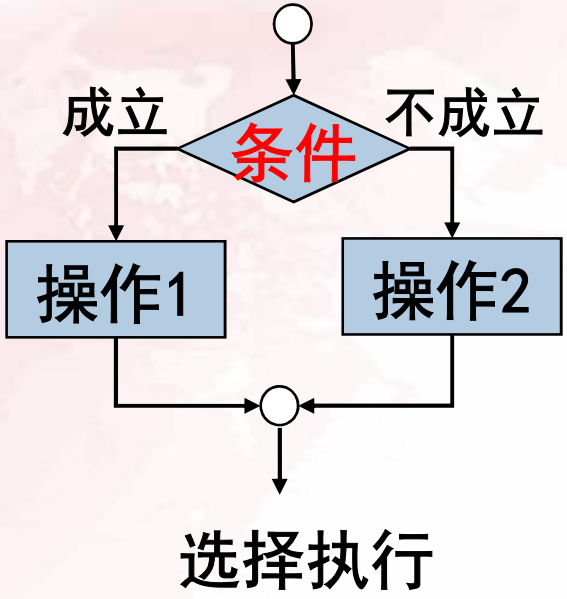
逻辑表达式示例

- 例4：判断字符ch是不是小写字母，大写字母，数字
  - 小写字母是'a'~'z'之间的所有字符　ASCII码表

ch是小写字母	ch >='a' && ch <= 'z'	ch >=97 && ch <=122
ch是大写字母	ch >='A' && ch <= 'Z'	ch >=65 && ch <= 90
ch是数字	ch >='0' && ch <= '9'	ch >=48 && ch <= 57

（demo_3_lower case letters.c）

选择执行--- if 条件语句



条件语句（if语句）

- 根据逻辑条件是否成立确定执行什么操作

```
...
if ( 条件 ){
    条件成立时执行的操作
}
...//无论条件是否成立均执行
```

```
...
if ( 条件 ){
    条件成立时执行的操作
}
else{
    条件不成立时执行的操作
}
...//无论条件是否成立均执行
```

demo_if_else.c

当操作只有一条语句时，if和else后的大括号可以不要

if和else对应的操作应该缩进，便于代码理解

嵌套条件语句

- 在if和else所对应的操作中可以再嵌套条件语句

```
if （ 条件1 ） {
    if （ 条件2 ）{
        //条件1成立且条件2成立
    }
    else {
        //条件1成立且条件2不成立
    }
}
else{
    if （ 条件3 ） {
        //条件1不成立且条件3成立
    }
    else {
        //条件1和条件均3不成立
    }
}
```

demo_if_else.c

嵌套条件语句

- 多个条件判断的另一种写法

```
if （ 条件1 ）{
    //条件1成立
}
else if （ 条件2 ） {
    //条件1不成立且条件2成立
}
else if （ 条件3 ） {
    //条件1不成立且条件2不成
    //立且条件3成立
}
else {
    //条件1，2，3均不成立
}
```

demo_if_else.c

（demo_3_lower case letters.c）

条件语句的匹配问题

- 就近匹配原则：else总是和最近的if匹配

```
int x=-1, y=2, z=3;

if (x > 0)
    if (y > 1) z = 1;
else
    z = 2;
printf("z=%d",z); //z=?
```

```
if (x > 0){
    if (y > 1) z = 1;
    else
        z = 2;
}
```

```
if (x > 0){
    if (y > 1) z = 1;
}
else
    z = 2;
```

(demo_if_else_match.c)

条件语句练习

- 练习1(demo_3_root.c)： 写程序计算 $ax^2+bx+c=0$的两个实根，如果两个实根相同只输出一个，如果不存在实根输出"no real root"
- 练习2(demo_3_swap.c)： 任意给定两个整数x和y，要求用x存储较小的数，y存储较大的数，然后输出。
- 练习3(demo_3_fee.c)： 阶梯电价计费，电价分三个档次，0～110度电，每度电0.5元；111～210度电，超出110部分每度电0.55元，超过210度电，超出210部分每度电0.70元，给出一个家庭一月用电量，请计算出应缴的电费。

```
//练习1
int main(){
    double a=1,b=4,c=4;// only one real root
    double condition;
    condition=pow(b,2)-4*a*c;
    if(condition<0){
        printf("no real root\n");
   }
    else if(condition>0){
        printf("the two real root is %f and %f\n",
            (-b+sqrt(condition))/(2*a),
            (-b-sqrt(condition))/(2*a));
    }
    else{
        printf(" the only one real root is %f\n",  -b/(2*a) );
    }
    return 0;
}
```

```c
//练习2-error
#include <stdio.h>
#include <stdlib.h>

int main(){
    int x=5,y=3;

    if(x>y){
        x=y;
        y=x;
    }
    printf( "x=%d, y= %d\n",
                x, y);
    return 0;
}
```

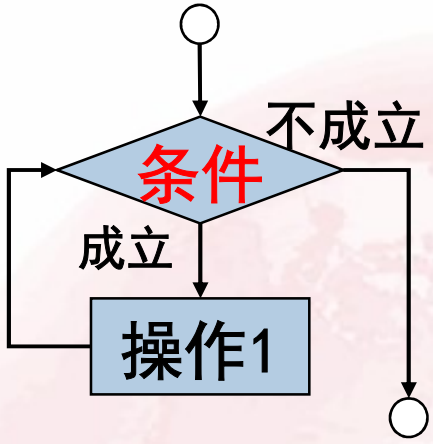
```c
//练习2-ok
#include <stdio.h>
#include <stdlib.h>

int main(){
    int x=5,y=3,tmp;

    if(x>y){
        tmp=x;
        x=y;
        y=tmp;
    }
    printf( "x=%d, y= %d\n",
                x, y);
    return 0;
}
```

//练习3-demo_3_fee.c

循环语句



循环（重复执行）

（1）**While**(条件)

{

……

}

（2）**do**

{

……

}**while**(条件);

（3）**for**(;;)

{

……

}

（1）**While** 循环

（2）do while 循环

（3）for 循环

循环语句——while循环

- 根据循环条件确定重复执行多少次相同的操作

```
while( 条件 ) {
    //条件成立时执行的操作（循环体）
}
```

  - 1. 对条件求值
  - 2. 条件成立（值非0），执行循环体回到步骤1， 条件不成立（值为0），结束循环

Note：当循环体只有一条语句时可以不写大括号，但为了避免逻辑错误，无论有多少条语句都加上大括号

while循环示例(demo_3_while.c)

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

int main(){
    double sum=0;
    int n=1;
    while(n<=100){
        sum=sum+sin(1.0/n);
        n=n+1;
    }
    printf("sum=%f\n",sum);
    system("pause");
    return 0;
}
```

$$\sum_{n=1}^{100} \sin(1/n) = ?$$

问题1：sum如果不初始化会怎样？

问题2：n如果不初始化会怎样？

问题3： 如果写成1/n会怎样？

问题4：如果没有这条语句会怎样？

使用循环语句的注意事项

- 初始化很重要
- 需要有改变条件的语句，否则可能出现死循环
- while循环可以划分为以下几个部分

```
初始化
while(条件){
      循环体
      改变条件的语句
}
```

循环语句

（1）**While** 循环

（2）do while 循环

（3）for 循环

循环语句——do...while循环

- 根据循环条件确定重复执行多少次相同的操作
  - 1. 执行循环体
  - 2 .对条件求值，条件成立（值非0），执行循环体回到步骤1，条件不成立（值为0），结束循环

```
do {
   //循环体
} while( 条件 );
```

Note:
1. while语句后的分号不能少
2. 当循环体只有一条语句时可以不写大括号，但为了避免逻辑错误，无论有多少条语句都加上大括号

demo_3_dowhile.c

while循环与do..while循环的区别

- do..while循环先执行循环体，再判断条件，因此无论条件是否成立至少执行一次循环体
- while循环先判断条件再确定是否执行循环体，因此可能一次也不执行循环体 **demo_3_dowhile2.c**

```
int n=3;
while( n<3 ){
    printf("n=%d\n", n);
    n=n+1;
}
printf("n=%d\n", n);
```

```
int n=3;
do{
    printf("n=%d\n",n);
    n=n+1;
} while(n<3);
printf("n=%d\n", n);
```

循环语句

（1）**While** 循环

（2）do while 循环

（3）for 循环

循环语句——for循环

demo_3_for.c

- 将while循环的初始化，条件，改变条件三个部分合在一起的一种循环形式

```
for( 初始化表达式; 条件表达式; 改变条件的表达式)
{
    循环体
}
```

如：`for(i=1,sum=0;i<=3;i++) {sum=sum+i;}`

1. 执行“初始化表达式”（只执行一次）
2. 判断“条件表达式”是否成立，如果不成立结束循环
3. 如果条件成立，执行循环体
4. 执行“改变条件表达式”
5. 回到2继续相同的步骤

循环语句——for循环

demo_3_for2.c

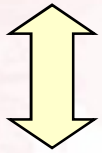
- 使用for循环的注意事项
  - 三个表达式都可以没有，但分号必须有

    ```
    for(;;){...};
    ```

  - 条件表达式没有，表示循环一直执行永不结束：for(;;){ … 跳出的语句(break) …};
  - 当循环体只有一条语句时可以不写大括号，但为了避免逻辑错误，无论有多少条语句都加上大括号

for循环和while循环相互转换

```
for( 初始化表达式; 条件表达式; 改变条件的表达式)
{
    循环体
}
```



```
初始化表达式;
while( 条件表达式 ) {
    //条件成立时执行的操作（循环体）
    改变条件的表达式;
}
```

for循环示例

demo_3_for3.c

$$\sum_{n=1}^{100} \sin(1/n) = ?$$

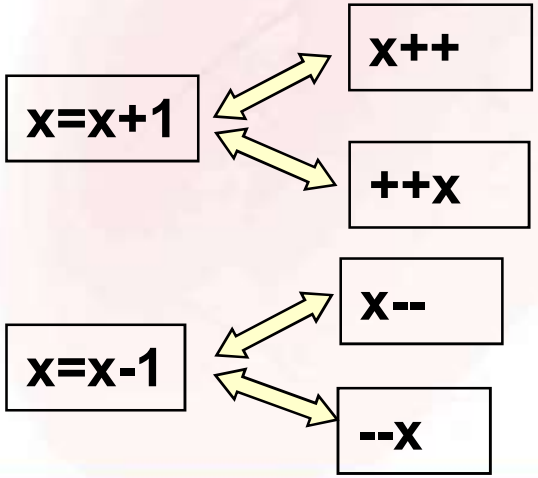
```
#include <stdio.h>
#include <math.h>

int main(){
    double sum=0;
    int n;
    for(n=1; n<=100; n=n+1 ){
        sum= sum + sin(1.0/n);
    }
    printf( "sum=%f\n", sum);
    return 0;
}
```

将变量值加1或减1可以有更简洁的写法

递增和递减运算符 ++ --

- 用于变量的递增（加1）和递减（减1）操作
- 递增运算符为“++”，递减运算符为“--”
- 递增和递减运算符可以出现在变量的前面或后面，但不能出现在表达式或常数的前面或后面
- 递增和递减运算符经常用在循环中改变条件




```
double sum=0;
int n=1;
while(n<=100){
    sum=sum+sin(1.0/n);
    n++ ;  // n =n +1
}
```

递增和递减运算符

- 递增和递减运算符出现在表达式中带来的问题

```
int x=1,y;
y= x++;
//y=?,x=?
```

⇩

先取x的值参与计算，
再递增x的值

⇩

```
int x=1,y;
y= x;
x=x+1;
//y=1,x=2
```

```
int x=1,y;
y= ++x;
//y=?,x=?
```

⇩

先递增x的值，
再取x的值参与计算

⇩

```
int x=1,y;
x=x+1;
y=x;
//y=2,x=2
```

demo_3_plusplus.c

复合赋值运算符

demo_3_value_fuhe.c

- x += 5 等价于 x = x + 5
- x -= 5 等价于 x = x - 5
- x *= 5 等价于 x= x * 5
- x /= 5 等价于 x= x / 5
- x %= 5 等价于 x= x % 5

```
//计算0-100每隔5度显示摄氏温度对应的华氏温度
int c;
for(c=0; c<=100; c+=5){
        f = c* 9.0/5+32;
        printf("c=%d ,f=%.1f\n", c, f);
}
```

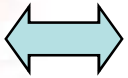
逗号运算符

demo_3_comma.c

- 通常在for循环的初始化表达式中使用
- 用来连接两个表达式，

  例如：表达式1 ， 表达式2;
- 计算形式：先计算表达式1，再计算表达式2

```
int main(){
    double sum=0;
    int n;
    for(n=1;n<=100;n=n+1){
        sum=sum+sin(1.0/n);
    }
}
```



```
int main(){
    double sum;
    int n;
    for( n=1, sum=0 ;n<=100;n=n+1){
        sum=sum+sin(1.0/n);
    }
}
```

流程控制语句：break continue

- break-退出当前循环，执行循环之后的语句

```
int i;
for(i=0; i<8; i++){
    if( i%2 )break;
    printf("i=%d ", i);
}
printf("i=%d\n", i);
```

i=0
i=1

- continue-结束当次循环，继续下一次循环

```
int i;
 for(i=0; i<8; i++){
     if( i%2 ) continue;
   printf("i=%d\n", i);
}
printf("i=%d\n", i);
```

i=0
i=2
i=4
i=6
i=8

关于goto语句

demo_3_break_continue.c

流程控制语句：switch语句

- 一种多分支结构，根据一个整型表达式的值从多个分支中选择执行
- switch语句的执行：先求出整型表达式的值，然后用该值与各个case表达式的值比较。如果遇到相等的值，程序就从那里执行下去；如果找不到，那么就从“default:”之后执行；如果没有default部分，那么整个switch语句的执行结束。

```
switch (整型表达式) {
case 整型常量表达式1 :
    语句序列 1;
    break;
case 整型常量表达式2 :
    语句序列 2;
    break;
    ....
default :
    语句序列
}
```

switch语句

demo_3_switch.c

如果整形表达式和整型常量表达式1匹配，该case分支后有break语句

从匹配的case开始一直执行到该case后的break语句，然后结束switch语句

```
switch (整型表达式) {
case 整型常量表达式1 :
    语句序列 1;
    break;
case 整型常量表达式2 :
    语句序列2;
    break;
    ....
default :
    语句序列
}
```

demo_3_switch.c

switch语句

如果整形表达式和整型常量表达式1匹配，但该case分支没有break语句？

从匹配的case开始一直执行到下一个break语句，如果一直没有遇到break语句就执行到switch语句结束，不管后面的case是否匹配

```
switch (整型表达式) {
case 整型常量表达式1 :
    语句序列 1;
    //没有break
case 整型常量表达式2 :
    语句序列2;
    break;
    ....
default :
    语句序列
}
```

switch语句

- 例1：某一门课的成绩用'A','B','C','D','F'来表示，成绩为'A','B','C','D'表示通过，所得学分分别为4，3，2，1，成绩为'F'表示未通过，所得学分为0。给出某个同学的成绩请判断是否通过，及获得的学分。

demo_3_grade.c

- 例2：给定年和月，求出该月天数，例如2011年11月为30天。

demo_3_month_day.c

循环语句示例

demo_3_loop.c

- 1. 每5度输出一项，给出摄氏温度0-100度和华氏温度的对照表，转换公式为：华氏温度 = 摄氏温度×9/5+ 32

```
#include <stdio.h>

int main(){
    double f;
    int c;
    for(c=0; c<=100; c=c+5){
        f = c* 9.0/5+32;
        printf("c=%d ,f=%.1f\n", c, f);
    }
    return 0;
}
```

循环语句示例

demo_3_digital.c

- 2. 假设*n*是一个由最多10位数字（*d*10，*d*9，…，*d*1）组成的正整数(如：98671)。编写一个程序在一列中列出*n*的每一位数字。最右边的数字*d*1应该被列在这一列的顶端。用*n*等于6、3704和170498测试你的程序。

  - 对任何一个整数n，要得到其个位数字，只需要执行n%10
  - 对任何一个整数n，要舍弃其个位数字，只需要执行n=n/10
  - 数字位数不确定的问题如何解决？

循环语句示例

demo_3_perfect_number.c

- 3. 请写一个程序，给出指定整数范围[1，10000]内的所有完数。一个数如果恰好等于除它本身外的所有因子之和，这个数就称为"完数"。例如6是完数，因为6=1＋2＋3。

```
for(n=1;n<=10000;n++)
{
   //求出n的因子之和sum;
   if( sum == n )
      //输出n;
}
```

```
for(i=1,sum=0;i<=n/2;i++)
{
    if( n%i==0 )
        sum+=i;
}
```

循环语句示例

demo_3_gcd.c

4. 求两个整数的最大公约数（GCD）

```
min=m<n?m:n;
for(i=1;i<=min;i++){
    if(m%i==0&&n%i==0) //判断i是不是公约数
        gcd=i; //gcd更新为更大的约数
}
```

方法1：从小到大循环扫描

```
min=m<n?m:n;
for(i=min;i>=1;i--){
    if(m%i==0&&n%i==0){ //判断i是不是公约数
        gcd=i; //i就是最大公约数，退出循环
        break;
    }
}
```

方法2：从大到小循环扫描

如果不使用break语句，循环应该怎么结束？

循环语句示例

demo_3_prime.c

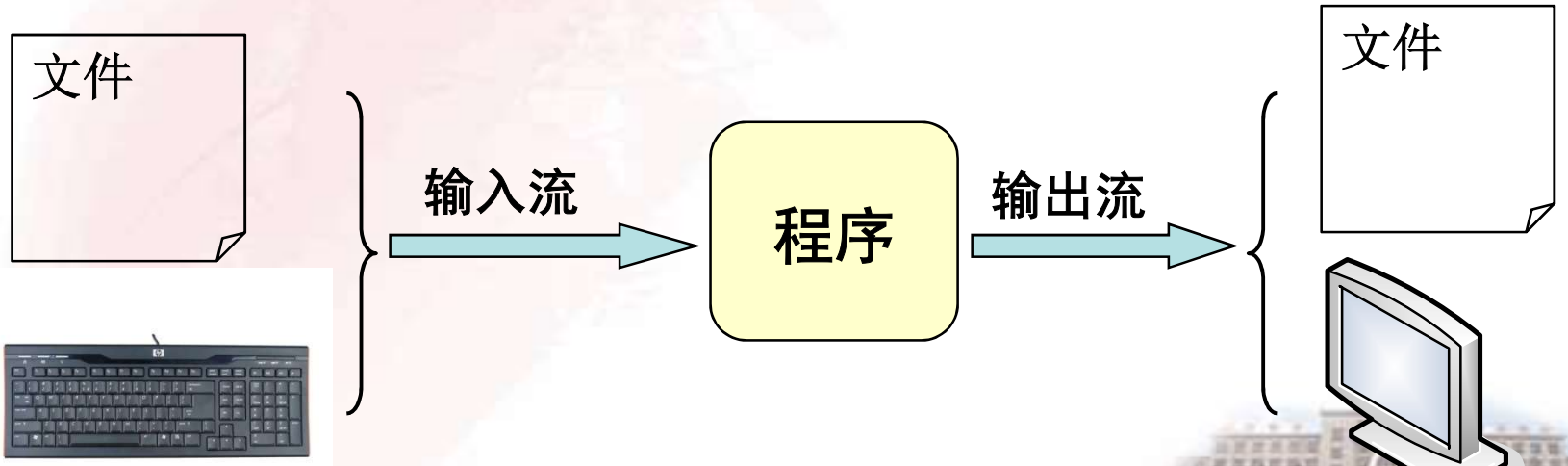
- 5. 判断一个数是不是素数

```
int i,n;
for(i=2;i<n;i++){
    if(n%i==0){ //判断i是不是n的因子
        break; // i是因子，退出循环
    }
}
if( n是素数 ) // “n是素数” 如何表示？
    printf(" %d is prime.\n",n);
else
    printf(" %d is not prime.\n",n);
```

if(n==i)

流重定向

- C语言提供三个标准流
  - 标准输入流(stdin)—通常是键盘缓冲区
  - 标准输出流(stdout)—通常是控制台缓冲区
  - 标准错误流(stderr)—通常是控制台缓冲区



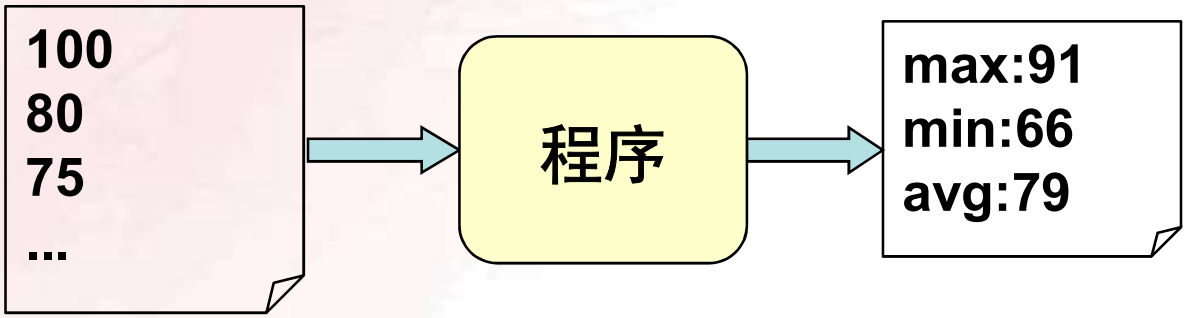
流重定向

- 标准输入流和标准输出流都可以重定向
  - stdin可以重定向到一个文件，此时读取数据从文件中读
  - stdout可以重定向到一个文件，此时输出数据均写入该文件
  - stdin重定向方法：可执行程序名 <文件名
  - stdout重定向方法：可执行程序名 >文件名

流重定向

redirect.c

- 流重定向示例：
  - 文件（input.txt）中有n+1行数据，第1行是n值，后面n行是n个整数(整数值不超过100)，从文件中读入这些数据，求出最大值，最小值和平均值并输出到文件(output.txt)



- 虽然还没有学习文件操作，但可以通过重定向完成
- 重定向对程序的编写没有任何影响，仍然假定是从标准输入流读数据，将结果写到标准输出流

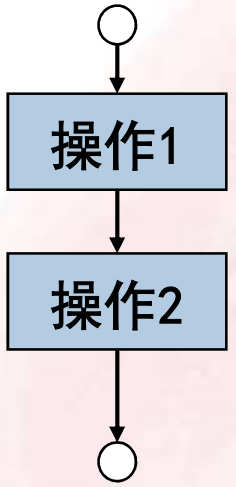
redirect.c

```
#include <stdio.h>
int main(){
    int i,n,max=0,min=100,avg=0;
    scanf("%d",&n);        //获得数据项数
    for(i=0;i<n;i++){
        int item;
        scanf("%d",&item); //读取一个数据
        avg+=item;         //累加和
        if(item>max) max=item; // 更新最大值
        if(item<min) min=item;  // 更新最小值
    }
    avg/=n;        //计算平均值
    printf("max:%d\n",max);
    printf("min:%d\n",min);
    printf("avg:%d\n",avg);
    return 0;
}
```

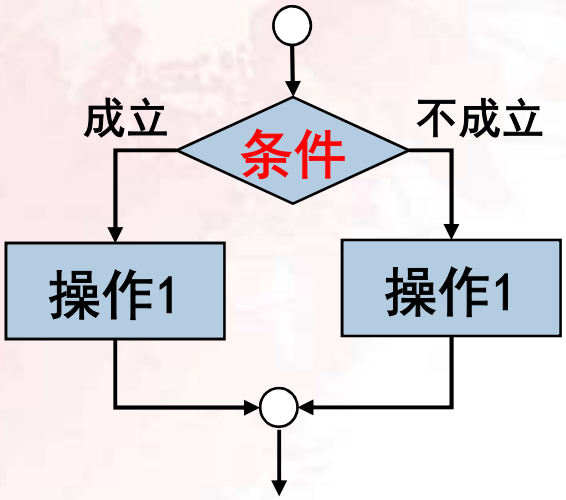
在命令行下执行：`redirect <input.txt >output.txt`

小结

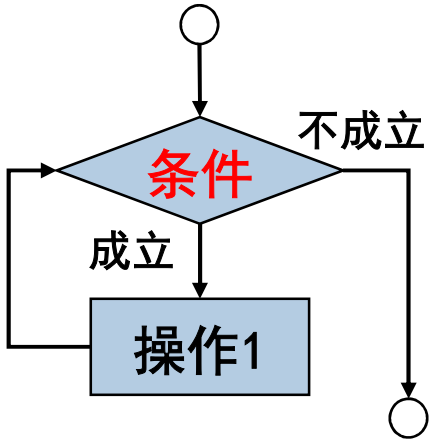
- 关系表达式和逻辑表达式
- 三种流程控制结构



顺序执行



选择执行



重复执行

小结

- 选择结构的几种形式

```
if ( 条件 ){
    条件成立时执行的操作
}
```

```
if ( 条件 ){
    条件成立时执行的操作
}
else{
    条件不成立时执行的操作
}
```

```
switch (整型表达式) {
case 整型常量表达式1 :
    语句序列 1;
    break;
case 整型常量表达式2 :
    语句序列2;
    break;
    ....
default :
    语句序列
}
```

小结

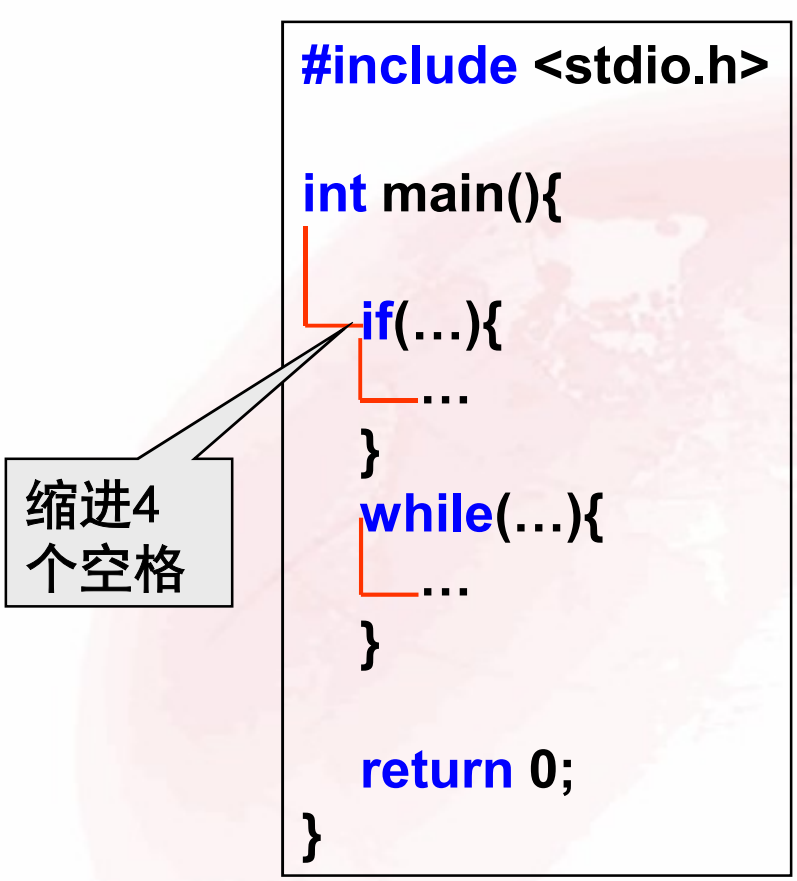
- 循环结构的几种形式

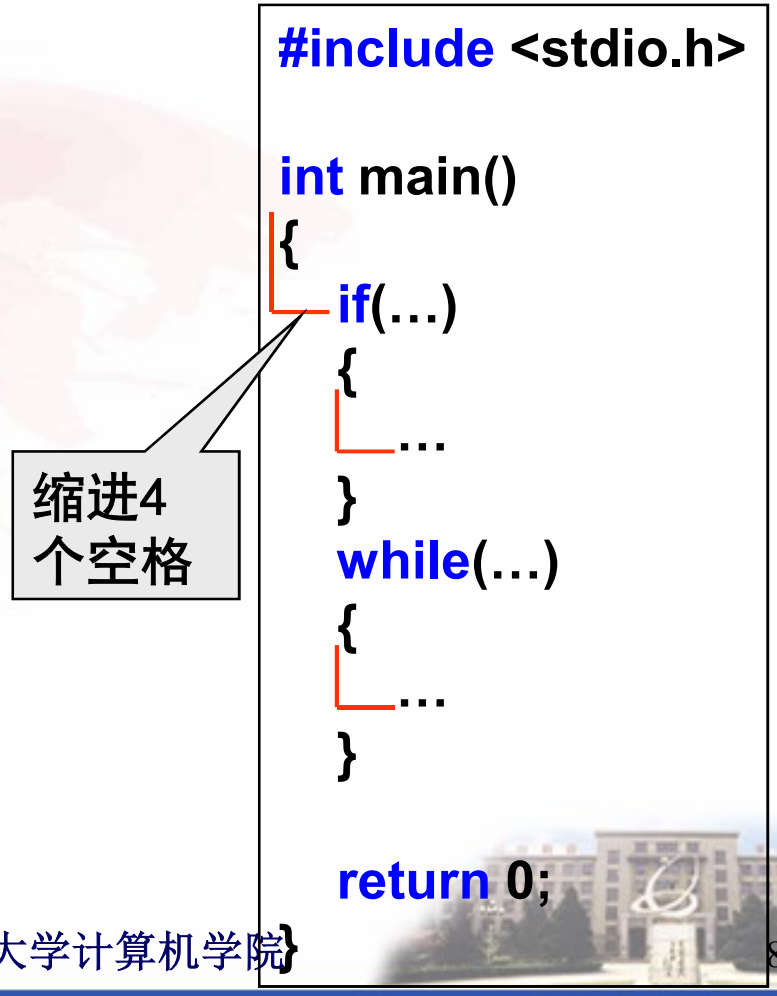
```
while( 条件 ) {
    //循环体
}
```

```
do {
    //循环体
} while( 条件 ) ;
```

```
for( 初始化表达式; 条件表达式; 改变条件的表达式)
{
    //循环体
}
```

代码风格


#include <stdio.h>
int main(){
if(…){
…
}
while(…){
…
}
return 0;
}
缩进4
个空格


#include <stdio.h>
int main()
{
if(…)
{
…
}
while(…)
{
…
}
return 0;
}
缩进4
个空格

作业

- 书面作业
  - P66：1，2，3，4

上机练习

第三章上机练习2

END