

# 第5章 循环结构程序设计 (1)



# 第一次上机实验小结

	我班 ( 应届及转专业 )	全年级	所占比例
完成6题	13 ( 去年26 )	46 ( 去年37 )	28.2% ( 去年70.3% )
完成5题	6 ( 去年10 )	26 ( 去年26 )	23.0% ( 去年38.5% )
完成4题	8 ( 去年11 )	42 ( 去年58 )	19.0% ( 去年19.0% )
完成3题	6 ( 去年8 )	77 ( 去年112 )	7.8% ( 去年7.1% )
完成2题	6 ( 去年1 )	36 ( 去年26 )	16.7% ( 去年3.8% )
完成1题	1 ( 去年0 )	15 ( 去年18 )	6.7% ( 去年0% )
完成0题	1 ( 去年0 )	17 ( 去年22 )	5.9% ( 去年0% )
未统计	0 ( 去年0 )	?	N/A
总 计	40 ( 去年56 )	259/去年299+	15.4%

## •主要问题 ( 此外还有很多其他问题..... )

- 对字符常量理解有误
- 代码风格混乱

- 判断是否相等==写成赋值符号=
- 使用类似 $0 < a < 3$ 这样的错误表达式

# 复习回顾

## ➤ 上次课的内容：

### ◆ if语句的嵌套

#### ● if-else的配套

### ◆ switch语句

### ◆ 选择结构综合举例

#### ● 判断闰年

#### ● 运费计算

### ◆ 循环结构初探

### ◆ while语句

### ◆ 再解析一个现实场景：

#### 程序员买包子的故事

老婆给当程序员的老公打电话：“下班顺路买一斤包子带回来，如果看到卖西瓜的，就买一个。”

当晚，程序员老公手捧一个包子进了家门……

老婆怒道：“你怎么就买了一个包子？！”

老公答曰：“因为看到了卖西瓜的。”

#### 老婆的逻辑

```
买x包子，x=1斤
if (看到卖西瓜的)
{
    买y西瓜，y=1个
}
```

#### 程序员的逻辑

```
买x包子，x=1斤
if (看到卖西瓜的)
{
    x=1个
}
```

# 简单的while语句应用：求和

- 求  $2 + 4 + 6 + \dots + 100$ ，即一百以内偶数的和
- 解题思路：
  - ◆ 这是累加问题，需要先后将50个数相加
  - ◆ 要重复50次加法运算，可用循环实现
  - ◆ 后一个数是前一个数加2而得
  - ◆ 加完上一个数*i*后，使*i*加2可得到下一个数

# 简单的while语句应用：求和


```
1. #include <stdio.h>
2.
3. int main()
4. {
5.     int i=2, sum=0; //初始化必不可少
6.
7.     while (i<=100) //条件表达式别忘括号
8.     {
9.         sum=sum+i; //在循环体内必须变更i的值
10.        i=i+2; //第9、10行顺序不可颠倒
11.    }
12.    printf("sum=%d\n", sum);
13.
14.    return 0;
15.}
```

# 从此，编程求全班平均成绩不再痛苦

顺序结构是这样的

```
scanf("%f",&mark);    //输入学生1的成绩
total += mark;        // 累加学生1的成绩
scanf("%f",&mark);    //输入学生2的成绩
total += mark;        // 累加学生2的成绩
scanf("%f",&mark);    //输入学生3的成绩
total += mark;        // 累加学生3的成绩
.....               // 此处省略130行
scanf("%f",&mark);    //输入学生69的成绩
total += mark;        // 累加学生69的成绩
scanf("%f", &mark);   //输入学生70的成绩
total += mark;        // 累加学生70的成绩
average = total/70.0;
```

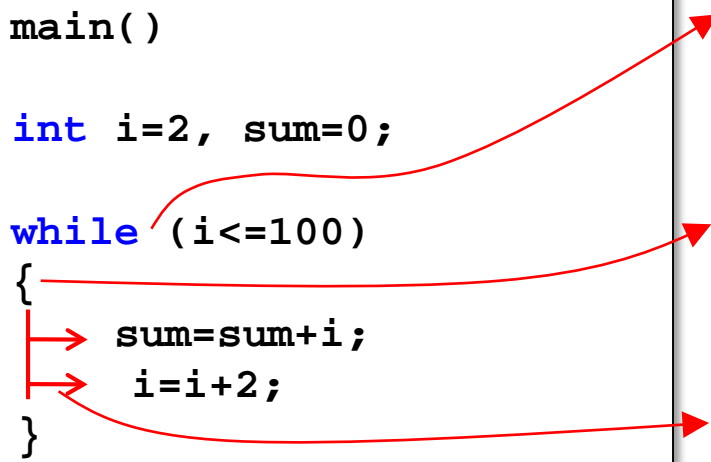
循环结构是这样的



```
i = 0;
while (i < 70)
{
    scanf("%f", &mark);
    total += mark;
    i++;
}
average=total/70.0;
```

# while语句的编程风格

```
1. #include <stdio.h>
2.
3. int main()
4. {
5.     int i=2, sum=0;
6.
7.     while (i<=100)
8.     {
9.         sum=sum+i;
10.        i=i+2;
11.    }
12.    printf("sum=%d\n", sum);
13.
14.    return 0;
15.}
```



- while关键字与条件表达式间留一个空格
- 花括号独立成行，垂直对齐
- 嵌套语句相对于花括号缩进

# 应用while语句的注意事项之一

- while语句中的表达式一般是关系表达式或逻辑表达式，但语法上所有表达式都可作为循环条件。

**但是新手慎用！**

◆ 比如

```
.....  
int x=10;  
while (y = x--)  
    printf("%d ",y);  
.....
```

```
.....  
char c;  
while ((c=getchar())!='\n')  
    putchar(c);  
.....
```



# 应用while语句的注意事项之二

- 只有位于判断条件之后的单个语句（简单语句或复合语句）才是循环体。缩进只是为了帮助阅读程序，不会改变程序的执行逻辑。

◆ 比如

```
.....  
int x=1;  
while (x<10)  
    printf("%d ",y);  
    x++;  
.....
```

应改成

```
.....  
int x=1;  
while (x<10)  
{  
    printf("%d ",y);  
    x++;  
}  
.....
```

# 用do...while实现循环

➤ 比如这个例子，执着地求输入零

## while实现

```
.....  
int z = 1; //若z=0则不执行循环  
while (z!=0)  
{  
    printf("Give me a zero:");  
    scanf("%d", &z);  
}  
.....
```

## do...while实现

```
.....  
int z; //不论z为何值，至少循环一次  
do  
{  
    printf("Give me a zero:");  
    scanf("%d", &z);  
} while (z!=0);  
.....
```

do...while循环的特点是：先执行循环体语句，后判断条件表达式，即**保证至少执行一次循环**。

# do...while的一般形式

do

语句

while (表达式);

即“循环条件”

.....

int z;

do

{

printf("Give me a zero:");

scanf("%d", &z);

} while (z!=0);

.....

真（非0）时再次执行循环体语句

假（为0）时不执行而结束循环

注意：括号是必须的！而且后面有分号！

# do...while的一般形式

do

语句

while (表达式);

循环体

可以是单个语句

也可以是复合语句

注意：复合语句必须首尾加大括号！

.....

int z;

do

{

```
printf("Give me a zero:");  
scanf("%d", &z);
```

} while (z!=0);

.....

# do...while更容易引起死循环

## ➤ 比如

```
.....  
int i=10;  
do  
    i--;  
while (i < 10);  
.....
```

### 程序段1

```
.....  
int i=10;  
do  
    i--;  
while (i++<10);  
.....
```

### 程序段2

```
.....  
int i=10;  
do  
    i--;  
while (i+=2<10);  
.....
```

## ➤ 要避免死循环需遵循两个原则

- ◆ 必须保证循环条件和循环体要同步地促使循环终止条件的产生 ( 如程序段1则死循环 )
- ◆ 促使循环终止条件的速度要比阻止的速度快 ( 如程序段2则不会发生死循环 )

# do...while和while的比较

```

1. int i,sum=0;
2. printf("i=?");
3. scanf("%d",&i);
4. while (i<=10)
5. {
6.     sum=sum+i;
7.     i++;
8. }
9. printf("sum=%d\n",sum);

```

```

i=?1
sum=55

```

```

i=?11
sum=0

```

```

1. int i,sum=0;
2. printf("i=?");
3. scanf("%d",&i);
4. do
5. {
6.     sum=sum+i;
7.     i++;
8. } while (i<=10);
9. printf("sum=%d\n",sum);

```

```

i=?1
sum=55

```

```

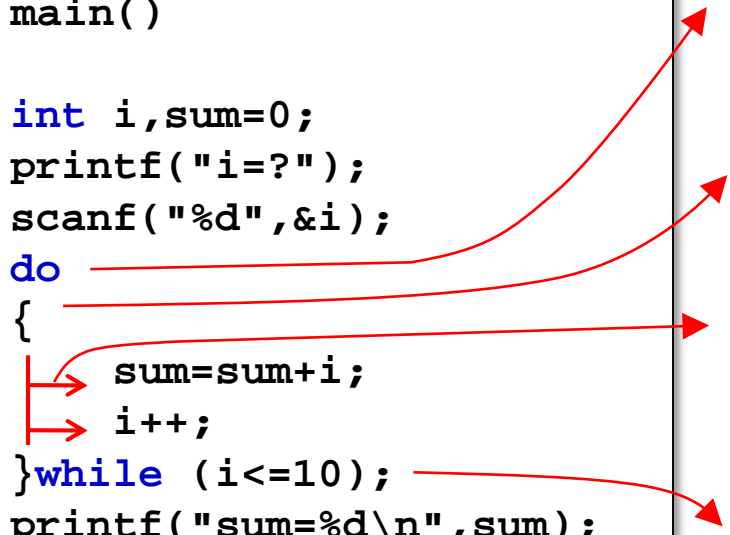
i=?11
sum=11

```

当while后面的表达式的第一次的值为“真”时，两种循环得到的结果相同；否则不相同

# do...while语句的编程风格

```
1. #include <stdio.h>
2.
3. int main()
4. {
5.     int i,sum=0;
6.     printf("i=?");
7.     scanf("%d",&i);
8.     do
9.     {
10.         sum=sum+i;
11.         i++;
12.     }while (i<=10);
13.     printf("sum=%d\n",sum);
14.
15.     return 0;
16. }
```



- do关键字独立成行
- 前花括号独立成行，与后花括号垂直对齐
- 嵌套语句相对于花括号缩进
- while关键字紧跟后花括号

# 不确定循环和计数循环

- **不确定循环**，循环体执行的次数不确定，while 和do...while常用于不确定循环，如

```
while ((c=getchar())!='\n')  
    putchar(c);
```

- **计数循环**，循环执行次数是能预先确定的，如

```
int end = 100;  
int count = 1;           //初始化  
while (count <= end)     //循环条件  
{  
    printf("%d\t",count); //动作  
    count++;             //更新计数  
}
```



# 计数循环的三个动作

1. 必须初始化一个计数器
2. 计数器与某个有限的值比较
3. 每次执行循环，计数器的值都要更新

```
1. int end = 100;  
2. int count = 1;  
3. while (count <= end)  
4. {  
5.     printf("%d\t", count);  
6.     count++;  
7. }
```

**使用while和do...while，即使是资深的程序员有时也会遗漏上面的某一个动作**

# 关于for语句的解释

- C语言中，提供了一种能将上述3个动作（初始化、判断、更新）放在一起的语句——for语句
- **for语句的一般形式：**

for (表达式1;表达式2;表达式3)

内嵌语句

设置初始条件，只执行一次。可以为零个、一个或多个变量设置初值执行。循环变量赋初值总是一个赋值语句，是用来给循环控制变量赋初值。在循环开始执行时执行，并且只执行一次。

```
for (i=1; i<100; i++)  
    sum += i;
```

# 关于for语句的解释

- C语言中，提供了一种能将上述3个动作（初始化、判断、更新）放在一起的语句——for语句
- **for语句的一般形式：**

for (表达式1; 表达式2; 表达式3)

内嵌语句

循环条件表达式，用来判定是否继续循环。在每次执行循环体前先执行此表达式，决定是否继续执行循环

```
for (i=1; i<100; i++)  
    sum += i;
```

# 关于for语句的解释

- C语言中，提供了一种能将上述3个动作（初始化、判断、更新）放在一起的语句——for语句
- **for语句的一般形式：**

for (表达式1;表达式2;表达式3)  
    内嵌语句

```
for (i=1; i<100; i++)  
    sum += i;
```

循环变量更新，定义循环控制变量每循环一次后，按什么方式变化，一般是一个递增或递减的语句

# for语句的执行步骤

for (表达式1;表达式2;表达式3)  
内嵌语句

## ➤ 执行过程：

**第1步**：求解表达式1；

**第2步**：求解表达式2，若其值为真（非0），则执行内嵌语句，然后执行第3步；若其值为假（0），则终止循环，转到第5步；

**第3步**：求解表达式3；

**第4步**：转回第2步继续执行

**第5步**：循环结束，执行for语句下面的一条语句。

# for与while之间的变换

```
for (i=1; i<100; i++)  
    sum += i;
```

等价于

```
i=1;  
while (i<100)  
{  
    sum += i;  
    i++;  
}
```

```
for (表达式1;表达式2;表达式3)  
    内嵌语句
```

等价于

```
表达式1  
while(表达式2)  
{  
    内嵌语句  
    表达式3  
}
```

**计数循环最好也是最常用的形式是for语句**

# 选择哪种循环实现方式？

```
1. #include <stdio.h>
2. int main()
3. {
4.     char answer;
5.     printf("你所要实现的循环是计数循环吗 ( 是请输入y , 否请输入n ) ? ");
6.     do {
7.         answer = getchar();
8.     } while (answer != 'y' && answer != 'n');
9.     if (answer == 'y')
10.        printf("你应该使用for循环\n");
11.    else
12.    {
13.        printf("你所要实现的循环至少需要循环一次吗 ( 是请输入y , 否请输入n ) ? ");
14.        do {
15.            answer = getchar();
16.        } while (answer != 'y' && answer != 'n');
17.        if (answer == 'y')
18.            printf("你应该使用do...while循环\n");
19.        else
20.            printf("你应该使用while循环\n");
21.    }
22.    return 0;
23.}
```

# 使用for语句的注意事项

- for循环中的 “**表达式1** ( 初始化 )” 、 “**表达式2** ( 循环条件 )” 和 “**表达式3** ( 循环变量更新 )” 都是可选的，即**可以省略**，但 “**;**” **不能省略**！
- 以下是省略了表达式1的例子，虽然合法，但注意 **i=1** 必须在for语句开始之前执行：

```
i=1;  
for ( ; i<100; i++)  
    sum += i;
```



# 使用for语句的注意事项

- for循环中的 “**表达式1** ( 初始化 )” 、 “**表达式2** ( 循环条件 )” 和 “**表达式3** ( 循环变量更新 )” 都是可选的，即**可以省略**，但 “**;**” **不能省略**！
- 以下是省略了表达式2的例子，但是发生死循环

```
for (i=1;;i++)  
    sum += i;
```

# 使用for语句的注意事项

- for循环中的 “**表达式1** ( 初始化 )” 、 “表达式2 ( 循环条件 )” 和 “表达式3 ( 循环变量更新 )” 都是可选的，即**可以省略**，但 “**;**” **不能省略**！

- 以下是省略了表达式3的例子，合法，但注意 **i++** 必须在循环体内执行

```
for (i=1;i<=100;)
{
    sum += i;
    i++;
}
```

# 使用for语句的注意事项

- for循环中的“**表达式1**（初始化）”、“**表达式2**（循环条件）”和“**表达式3**（循环变量更新）”都是可选的，即**可以省略**，但“**;**”**不能省略**！
- 甚至，3个表达式都可以省略，如下：

```
for (;;)
```

```
{
```

内嵌语句

```
}
```

相当于

```
while (1)
```

```
{
```

内嵌语句

```
}
```

# 使用for语句的注意事项

- for循环中的“**表达式1**（初始化）”、“**表达式2**（循环条件）”和“**表达式3**（循环变量更新）”都是可选的，即**可以省略**，但“**;**”**不能省略**！
- 问题是为什么要省略呢？？？？？
  - ◆ 新手建议：**不要省略for循环语句的任何一个表达式！**
  - ◆ 编程道德守则之一：“**宽以待人，严于律己**”

# 使用for语句的注意事项

- 可以减小循环变量而不是增大它，例如

```
for (i=10;i>1;i--)  
    sum+=i;
```

- 需要的话可以让循环变量依次加或减2、3，例如

```
for (i=1;i<100;i+=2)  
    sum+=i;
```

- 可以让循环变量几何增长而不是线性增长，例如

```
for (i=1;i<100;i*=2)  
    sum+=i;
```

# 使用for语句的注意事项

- 用来计数的可以不是数字而是字符，例如

```
for (c='a';c<='z';c++)  
    printf("%c ",c);
```

**这个还OK**

- 表达式1可以赋值，也可以是其他表达式，例如

```
int i=1;  
for (printf("开始求和:");i<100;i+=2)  
    sum+=i;
```

**但是, 请尽量  
避免这  
些写法！！**

- 表达式1和3可以是逗号表达式，例如

```
for (i=0,j=100;i<=100;i++,j--)  
    sum+=i+j;
```

# 使用for语句的注意事项

- 表达式2可以是任意表达式，只要值非零就执行循环体，例如 `for (i=0;(c=getchar())!='\n';i+=c);`
- 循环中的动作可以改变循环表达式的参数，例如

```
for (i=1,step=1;i<100;i+=step)
{
    printf("%d\n",i);
    if (i>10)
        step = 10;
}
```

**这些也是怪咖的写法！  
还不如用while语句**

**注意，这里有危险，step指定为0，则死循环**

# for语句的编程风格

```
1. #include <stdio.h>
2.
3. int main()
4. {
5.     int i,n;
6.     double fac=1;
7.     scanf("%d", &n );
8.
9.     for (i=1; i<=n; i++)
10.    {
11.        fac = fac*i;
12.    }
13.    printf ("%d!=%.0f\n",n,fac);
14.
15.    return 0;
16.}
```

- for关键字之后留一个空格
- 花括号独立成行，垂直对齐
- 嵌套语句相对于花括号缩进



# Quiz Time , 如何公平的提问



# Quiz Time 1

## ➤ 写出下列程序的运行结果

1. `printf("5%%2=%d,2/5=%d",5%2,2/5);`
2. `printf("%f,%f",5/2.0,(float)(5/2));`
3. `printf("%c\n%d",'A'+3,'0'+5);`
4. `int n=0; if (n=1) printf("%d",n--); else  
printf("%d",n++);`
5. `int n=1; if (n%2||n=0) printf("%d",++n);  
else printf("%d",--n);`

# 循环结构简单编程举例 ( 1 )

## ➤ 输入两个正整数，分别输出他们的最大公约数和最小公倍数

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

2. int main() //辗转相除法
3. {
4.     int a,b,m,r;
5.     scanf("%d,%d",&a,&b);
6.     m = a*b;
7.     r = a%b;
8.     while (r!=0)
9.     {
10.         a=b;
11.         b=r;
12.         r=a%b;
13.     }
14.     printf("%d,%d",b,m/b);
15.     return 0;
16. }
```

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

2. int main() //按定义求解
3. {
4.     int a,b,p,q;
5.     scanf("%d,%d",&a,&b);
6.     p = (a>b)?a:b;
7.     q = a+b-p;
8.     while (a%q!=0 || b%q!=0)
9.     {
10.         q--;
11.     }
12.     while (p%a!=0 || p%b!=0)
13.     {
14.         p++;
15.     }
16.     printf("%d,%d",q,p);
17.     return 0;
18. }
```

# 循环结构简单编程举例 (2)

## ➤ 输入一个非负整数，并进行逆向显示

```
1. #include <stdio.h>
2. int main()
3. {
4.     int num;
5.     //如何确保输入非负整数?
6.     scanf("%d", &num);
7.
8.     while (num > 0)
9.     {
10.         printf("%d", num%10);
11.         num = num/10;
12.     }
13.     putchar('\n');
14.
15.     return 0;
16. }
```

```
1. #include <stdio.h>
2. int main()
3. {
4.     int num;
5.     do
6.     {
7.         printf("请输入一个非负整数: ");
8.         scanf("%d", &num);
9.         if (num < 0)
10.        {
11.            printf("请不要输入负整数。\\n");
12.        }
13.    } while (num<0);
14.    printf("该整数逆向显示的结果是");
15.    while (num > 0)
16.    {
17.        printf("%d", num%10);
18.        num = num/10;
19.    }
20.    putchar('\n');
21.    return 0;
22. }
```

# 循环结构简单编程举例 ( 3 )

- 读入用户输入的整数，至输入0结束，输出所有输入整数的平均值

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

2. int main()
3. {
4.     int count=0, sum=0, x;
5.     do
6.     {
7.         scanf("%d", &x);
8.         sum += x;
9.         count++;
10.    } while(x!=0);
11.    if (count > 1)
12.    {
13.        printf("%f", (double)sum/(count-1));
14.    }
15.    return 0;
16.}
```

# 循环结构简单编程举例 ( 4 )

➤ 输出显示10~150之间所有能被4或5整除的整数

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

2. int main()
3. {
4.     int i, count=0;
5.     for (i=10; i<=150; i++)
6.     {
7.         if (i%4==0 || i%5==0)
8.         {
9.             printf("%d", i);
10.            count++;
11.            putchar(count%5==0?'\n':' ');
12.        }
13.    }
14.    putchar('\n');
15.    return 0;
16.}
```

# 循环中的循环

- 即嵌套循环，指一个循环同时又是另一个循环的循环体，如

```
while (...)
{
    while (...)
    {
        .....
    }
}
```

内层循环

```
do
{
    do
    {
        .....
    } while (...);
} while (...);
```

内层循环

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

内层循环

```
while (...)
{
    do
    {
        .....
    } while (...);
}
```

内层循环

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

内层循环

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

内层循环

- 如果内嵌的循环再嵌套循环，那就成了多层嵌套循环

# 常见的嵌套循环

- 循环有3种实现方式，如果是两层循环，就可以组合成9种形式的嵌套循环
- 最常用的是两层for循环：外层for循环处理所有的行，内层for循环处理所有的列

```
*****
*****
*****
*****
*****
Press any key to continue
```

```
1. #include <stdio.h>
2.
3. int main()
4. {
5.     int i,j;
6.     for (i=0; i<5; i++)
7.     {
8.         for (j=0; j<10; j++)
9.         {
10.             printf("*");
11.         }
12.         printf("\n");
13.     }
14.     return 0;
15. }
```



# 内层循环可依赖于外层循环

```
1. #include <stdio.h>
2.
3. int main()
4. {
5.     int i,j;
6.     for (i=0; i<5; i++)
7.     {
8.         for (j=0; j<=i; j++)
9.         {
10.            printf("*");
11.        }
12.        printf("\n");
13.    }
14.    return 0;
15.}
```

```
*****
*****
*****
*****
*****
Press any key to continue_
```

# 循环结构程序设计方法

- **第一步**：根据题目要求和选择循环的原则选择一种或几种循环实现方式；
- **第二步**：确定每个循环的**终止条件**、**初始化语句**和**更新语句**；
- **第三步**：确定每个循环的循环体，**顺序应该从最内层循环开始，到最外层循环结束**；
- **第四步**：画出程序的流程图
- **第五步**：按照流程图将所有的程序块进行组合，并加上需要包含的头文件等其他内容，形成最终的程序。

# 作业 2017/11/8

## ➤ 按下列要求编写程序，提交手写源代码

1. 读入用户输入的正整数，至输入-1结束，输出所有输入整数的平均值。
2. 输出显示20~160之间所有能被4和5整除的整数。
3. 输入两个整数，分别输出他们的最大公约数和最小公倍数
4. 输入一个非负整数，并进行逆向显示，输出要求忽略最高位的0，如输入,2100，输出12，

## ➤ 上机练习（不用交）：编译运行本讲义例程，教材第五章 4、5、8、9、10