计算机导论与程序设计 [CS006001-60]

段江涛

机电工程学院

西安电子科技大学 XIDIAN UNIVERSITY

2019 年 10 月

lecture-7 主要内容

循环结构程序设计

1. *while*(表达式){···}
2. *do*{···}*while*(表达式);
3. *for*(表达式 1; 表达式 2; 表达式 3){···}
4. 循环的嵌套
5. break,continue 改变循环执行的状态

为什么需要循环控制

- 要向计算机输入全班 50 个学生的成绩;(重复 50 次相同的输入操作)
- 分别统计全班 50 个学生的平均成绩; (重复 50 次相同的计算操作)

```
float score1,score2,score3,score4,score5,aver; // 5门课成绩及平均成绩
// 输入第1个学生5门课的成绩
scanf("%f%f%f%f%f",&score1,&score2,&score3,&score4,&score5);
// 求第1个学生平均成绩
aver=(score1+score2+score3+score4+score5)/5;
printf("aver=%7.2f",aver); // 输出第1个学生平均成绩
// 输入第2个学生5门课的成绩
scanf("%f%f%f%f%f",&score1,&score2,&score3,&score4,&score5);
// 求第2个学生平均成绩
aver=(score1+score2+score3+score4+score5)/5;
printf("aver=%7.2f",aver);// 输出第2个学生平均成绩
...
```

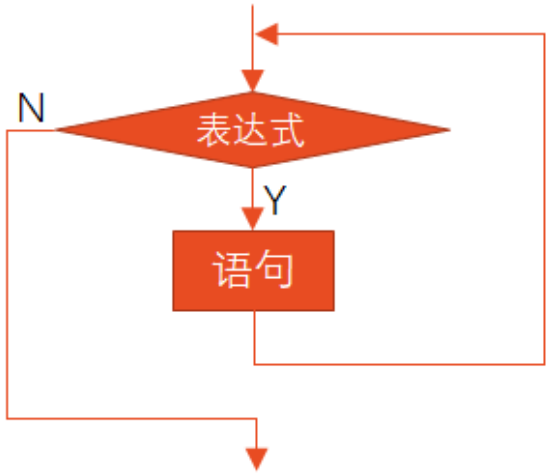
用循环控制处理重复操作

- 要向计算机输入全班 50 个学生的成绩;(重复 50 次相同的输入操作)
- 分别统计全班 50 个学生的平均成绩; (重复 50 次相同的计算操作)

```
float score1,score2,score3,score4,score5,aver; // 5门课成绩及平均成绩
int i=1; // 设整型变量i初值为1
while( i<=50 ) // 当i的值小于或等于50时执行花括号内的语句
{
   scanf("%f%f%f%f%f",&score1,&score2,&score3,&score4,&score5);
   aver=(score1+score2+score3+score4+score5)/5;
   printf("aver=%7.2f",aver);
   i++; // 每执行完一次循环使i的值加1
}
```

while(表达式){···}

```
while( 表达式 )
{
   // 循环体
   执行多条语句;
}
```



while 循环特点

每轮循环: 首先判断表达式的值, 若“真” (以非 0 值表示) 时, 就执行循环体语句; 为“假” (以 0 表示) 时, 就不执行循环体语句。

易犯错误

```
while(表达式);
{
   // 循环体
   执行多条语句;
}
```

变体

```
while(1)
{
  if(表达式) break; // 退出循环
  执行多条语句;
}
```

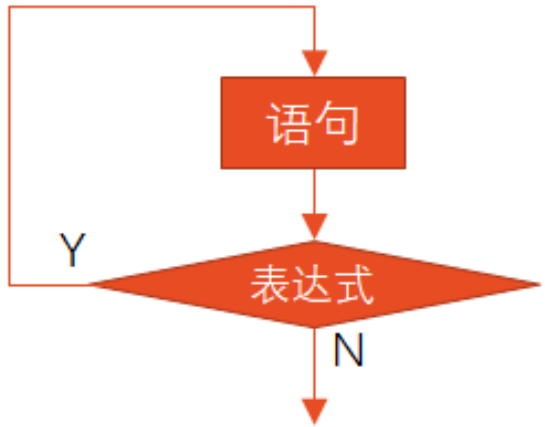
[例 5.1] 求 $1+2+3+\cdots+100$, 即 $\sum_{i=1}^{100} i$。

```
int i=1,sum=0; //定义变量i的初值为1,sum的初值为0
while(i <= 100) //当i>100，条件表达式i<=100的值为假，不执行循环体
{ //循环体开始
   sum=sum+i; //第1次累加后，sum的值为1
   i++; //加完后，i的值加1，为下次累加做准备
} //循环体结束
printf("sum=%d\n",sum); //输出1+2+3…+100的累加和
```

1. 循环体如果包含一个以上的语句,应该用花括号括起来,作为复合语句出现。
2. 不要忽略给 i 和 *sum* 赋初值,否则它们的值是不可预测的,结果显然不正确。
3. 在循环体中应有使循环趋向于结束的语句。如本例中的 $i++;$ 语句。如果无此语句,则 i 的值始终不改变,循环永远不结束。

do{···}*while*(表达式);

```
do
{
  // 循环体
  执行多条语句;
} while( 表达式
       );
```



循环特点

先无条件地执行循环体,然后判断循环条件是否成立。。

易犯错误

```
do
{
   // 循环体
   执行多条语句;
} while( 表达式 )
```

变体

```
do
{
  执行多条语句;
  if(表达式) break; // 退出循环
} while(1);
```

[例 5.2] 求 $1+2+3+\cdots+100$, 即 $\sum_{i=1}^{100} i$。

```
int i=1,sum=0; //定义变量i的初值为1,sum的初值为0
do
{ //循环体开始
   sum=sum+i; //第1次累加后，sum的值为1
   i++; //加完后，i的值加1，为下次累加做准备
}while(i <= 100); //当i>100，条件表达式i<=100的值为假，不执行循环体
printf("sum=%d\n",sum); //输出1+2+3…+100的累加和
```

1. 在一般情况下,用 while(){···} 语句和用 do{···}while(); 语句处理同一问题时,若二者的循环体部分是一样的,那么结果也一样。
2. 但是如果 while 后面的表达式一开始就为假 (0 值) 时,两种循环的结果是不同的。

while(表达式){···} 与 *do*{···}*while*(表达式);

[例 5.3] 求 $\sum_{i=n}^{100} i$。考虑输入 $n > 100$ 时的情况,以下程序的不同。

```
int i,sum=0;
scanf("%d",&i);
while(i <= 100)
{
  sum=sum+i;
  i++;
}
printf("sum=%d\n",sum);
```

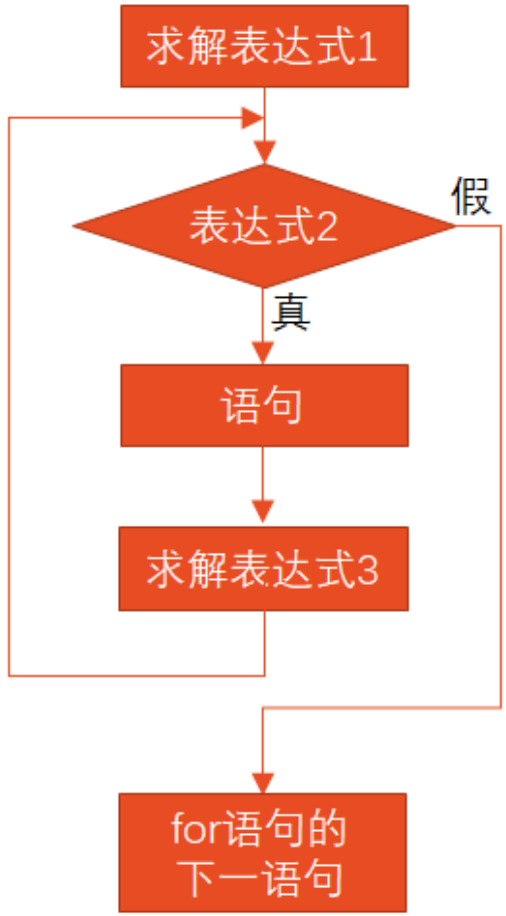
```
int i,sum=0;
scanf("%d",&i);
do
{
  sum=sum+i;
  i++;
}while(i <= 100);
printf("sum=%d\n",sum);
```

for(表达式 1; 表达式 2; 表达式 3){···}

```
for(表达式1;表达式2;表达式3)
{
   // 循环体
   执行多条语句;
}
```

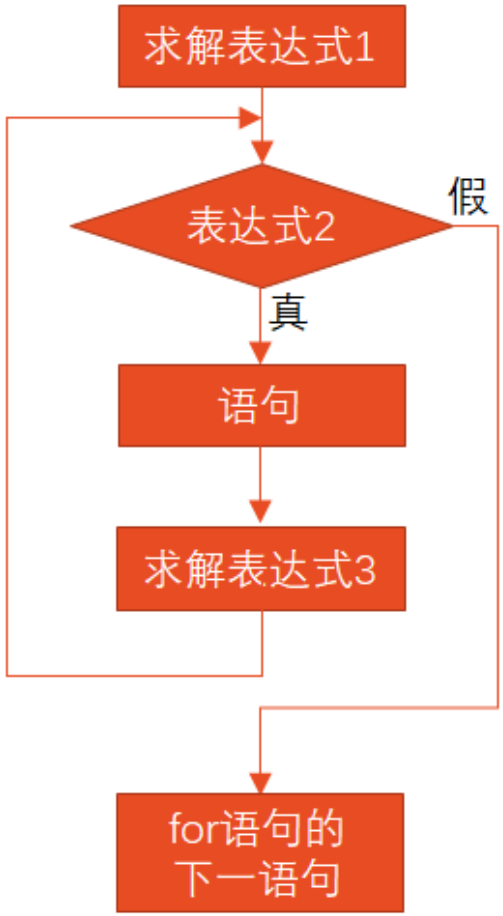
≡

```
表达式1;
while(表达式2)
{
   // 循环体
   执行多条语句;
   表达式3;
}
```



for(表达式 1; 表达式 2; 表达式 3){···}

```
for(表达式1;表达式2
    ;表达式3)
{
   // 循环体
   执行多条语句;
}
```



- 表达式 1: 设置初始条件,只执行一次。可以为零个、一个或多个变量 (逗号隔开) 设置初值。
- 表达式 2: 是循环条件表达式,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 (包括第 1 次循环),决定是否继续执行循环。
- 表达式 3: 作为循环的调整,例如使循环变量增值,它是在执行完循环体后才进行的。

for(表达式 1; 表达式 2; 表达式 3){···}, 省略表达式

```
int i;
for(i=0;i<=100;i++)
{
  printf("%d\n",i);
}
```

```
int i=0;
for(;i<=100;i++)
{
  printf("%d\n",i);
}
```

```
int i=0;
for(i=0;;i++)
{
  if(i>100) break;//退出循环
  printf("%d\n",i);
}
```

```
int i;
for(i=0;i<=100;)
{
  printf("%d\n",i);
  i++;
}
```

```
int i=0;
for(;i<=100;)
{
  printf("%d\n",i);
  i++;
}
```

```
int i=0;
for(;;)
{
  if(i>100) break;//退出循环
  printf("%d\n",i);
  i++
}
```

循环的嵌套

01

```
while()
{    ⋮
     while()
     {···}
}
```

内层循环

02

```
do
{    ⋮
     do
     {···}
     while();
}while();
```

内层循环

03

```
for( ; ; )
{
     for( ; ; )
     {···}
}
```

内层循环

04

```
while()
{    ⋮
     do
     {···}
     while();
     ⋮
}
```

内层循环

05

```
for( ; ; )
{    ⋮
     while()
     {···}
     ⋮
}
```

内层循环

06

```
do
{    ⋮
     for( ; ; )
     {···}
}while();
```

内层循环

几种循环的比较

1. 3 种循环都可以用来处理同一问题,一般情况下它们可以互相代替。
2. 在 while 循环和 do…while 循环中,只在 while 后面的括号内指定循环条件,因此为了使循环能正常结束,应在循环体中包含**使循环趋于结束的语句** (如 i++ 等)。
3. for 循环可以在表达式 3 中包含使循环趋于结束的操作,甚至可以将循环体中的操作全部放到表达式 3 中 (逗号隔开)。因此 for 语句的功能更强,凡用 while 循环能完成的,用 for 循环都能实现。
4. 用 while 和 do…while 循环时,循环变量初始化的操作应在 while 和 do…while 语句之前完成。而 for 语句可以在表达式 1 中实现循环变量的初始化。
5. while 循环、do…while 循环和 for 循环都可以用 break 语句跳出循环,用 continue 语句结束本次循环。

break,continue 改变循环执行的状态


```
while(表达式)
{
  printf("语句1");
  if(条件表达式) break; //提前终止循环
  printf("语句1");
}
```

```
while(表达式)
{
   printf("语句1");
   if(条件表达式) continue; //结束本次
        循环, 进入下轮循环
   printf("语句1");
}
```

用 break 语句提前终止循环

[例 5.4] 在全系 1000 名学生中举行慈善募捐,当总数达到 10 万元时就结束,统计此时捐款的人数以及平均每人捐款的数目。

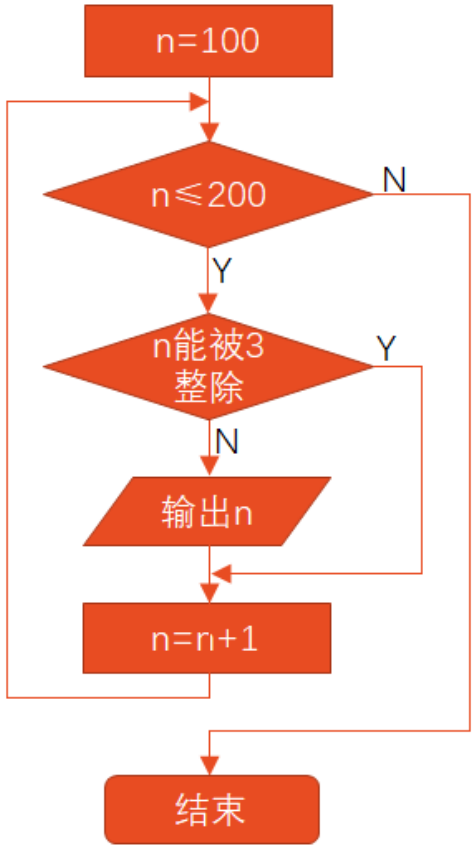
```
#define SUM 100000 //指定符号常量SUM代表10万
float amount,aver,total;
int i;
for (i=1,total=0; i<=1000; i++)// 表达式1给多个变量赋初值，用逗号隔开。
{
  printf("please␣enter␣amount:");
  scanf("%f",&amount);
  total = total + amount;
  if(total >= SUM) break;
}
aver=total/i;
printf("num=%d\naver=%10.2f\n",i,aver);
```

注意: break 语句只能用于循环语句和 switch 语句之中,而不能单独使用。

用 continue 语句提前结束本次循环

[例 5.4] 要求输出 100~200 之间的不能被 3 整除的数。

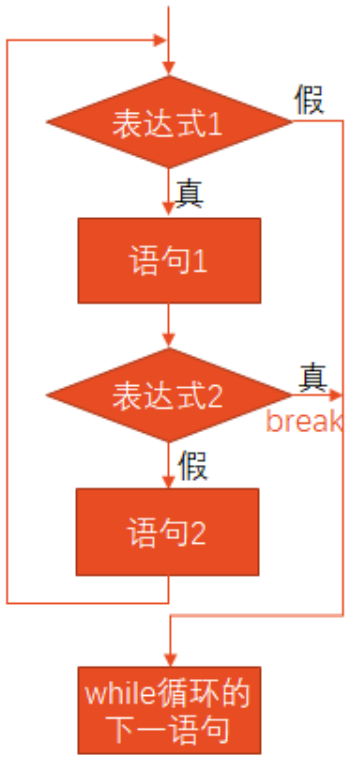
```
#include <stdio.h>
int main()
{
  int n;
  for (n = 100;n <= 200;n++)
  {
    if (n%3==0) continue;
    printf("%d␣",n);
  }
  printf("\n");
  return 0;
}
```



break 语句和 continue 语句的区别

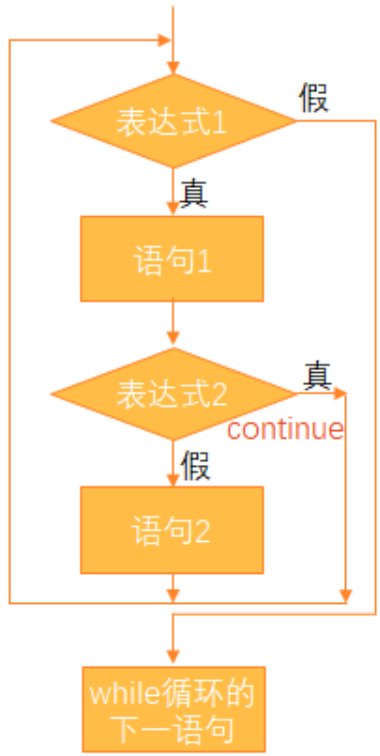
break;

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



continue;

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



注意: continue 语句只结束本次循环,而非终止整个循环。break 语句结束整个循环,不再判断执行循环的条件是否成立。

break 语句和 continue 语句的区别

[例 5.5] 输出以下 4 × 5 的矩阵。

```
int i,j,n=0;
for(i=1;i<=4;i++)
{
  for(j=1;j<=5;j++,n++) //n用来累计输
       出数据的个数
  {
    if(n%5==0) printf("\n"); //控制在
         输出5个数据后换行
    if (i==3 && j==1) break;
    printf("%d\t",i*j);
  }
}
```

```
int i,j,n=0;
for(i=1;i<=4;i++)
{
   for(j=1;j<=5;j++,n++) //n用来累计输
        出数据的个数
   {
     if(n%5==0) printf("\n"); //控制在
          输出5个数据后换行
     if (i==3 && j==1) continue;
     printf("%d\t",i*j); // \t就是Tab
          键，是特殊字符，表示多个空格
   }
}
```

注意事项小结

1. while( ){ }; do { } while( ); for(;;){ } 执行顺序;
2. 循环变量的开始和结束条件;
3. 循环体是复合语句时,必须用 { } 扩起来;
4. 必要时,用 break 结束整个循环,用 continue 结束本次循环;
5. 关键是找出循环规律,必要时设计流程图,指导代码实现。

欢迎批评指正！