项目 3

循环结构程序设计

程序经常会重复执行某些相同的操作，比如求“1＋2＋…＋100”的和。可首先设置一个变量 sum，其初值为 0，用来存放计算之和。然后，解决以下 3 个问题即可。

(1) 将 k 的初值置为 1。

(2) 每执行 1 次“sum＋＝k”后，k 增 1。

(3) 当 k≤100 时，重复执行第(2)步；否则，结束。

这种根据某个条件重复执行相同算法的程序结构，称为循环结构。C 语言提供了 3 种实现循环结构的语句，即 while 语句、do-while 语句和 for 语句。

循环结构程序设计主要内容如下：

(1) while 语句、do-while 语句、for 语句等 3 种基本语句。

(2) break、continue、goto 语句。

(3) 循环结构程序嵌套。

重点与难点：

(1) 用 while、do-while、for 三种基本语句编写循环结构程序的方法。

(2) break 语句与 continue 语句的用法及区别。

任务 3.1　用 while 语句实现的循环结构

任务说明

在日常生活中遇到的有些循环问题，事前不知道循环次数，这时可以使用 while 语句实现循环。在本任务中，将学习 while 语句的格式、执行流程和应用方法。

相关知识

1. while 语句格式

while 语句也称为“当型”循环语句，它的格式为：

```
while(表达式)
{
    循环体语句;
}
```

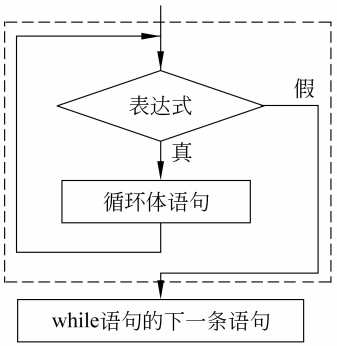

图 3-1　while 语句的执行流程

2. while 语句执行流程

while 语句执行流程如图 3-1 所示。

While 语句执行的流程如下。

(1) 求解表达式。如果其值为非 0,转第(2)步;否则转第(3)步。

(2) 执行循环体语句,然后转第(1)步。

(3) 执行 while 语句的下一条语句。

while 循环的特点是:先判断“表达式”是否成立,然后再决定是否执行循环体语句。

注意事项:

(1) 循环体如包括一个以上的语句,则必须用{}括起来,组成复合语句。

(2) while 语句的表达式一般是关系表达或逻辑表达式,也可以是其他表达式,只要表达式的值为非 0,即可使循环继续。

任务实施

1. 任务功能

编写程序,用 while 语句求 1+2+3+…+100 的值。

2. 编程思路

这是一个求多个有规律的数相加的问题,可以通过取数和求和两步来完成。

(1) 取数。第一个数为 1,其后的数都是在前一个数的基础上加 1 得到,直至 100。因此,可以在循环体中定义一个变量 i,每循环一次使 i 增 1,直到 i 的值超过 100,这样可以获得所要的每个数。

(2) 求和。可定义一个变量 sum 来存放和,给 sum 赋初值 0。当第 1 次判断表达式 i≤100 成立时,第一次执行循环体,sum+1 赋给 sum,i 的值变为 2;当第 2 次判断表达式 i≤100 成立时,第二次执行循环体时,sum+2 赋给 sum,i 的值变为 3;……。以此类推,直至循环结束,最后 sum 中存放的是 1+2+3+…+100 的值。执行流程如图 3-2 所示。

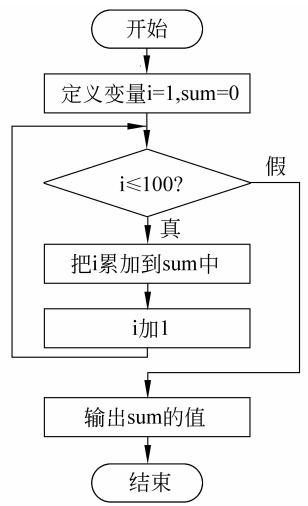

图 3-2　求 1+2+3+…+100 流程图

3. 源程序 EX3-1-1.c

```
#include<stdio.h>
void main( )
{
    int i=1,sum=0;                    //循环控制变量 i 和累加器 sum 初始化
```

```
    while( i<=100 )
    {
        sum+=i;                          //累加
        i++;                             //i自加
    }
    printf("sum=%d\n",sum);              //输出结果
}
```

4. 运行、调试

在VC++ 6.0开发环境下,编辑、编译和调试源程序EX3-1-1.c。程序运行的结果为:

```
sum=5050
```

任务拓展

在源程序EX3-1-1.c的基础上,若把while的循环体改为:

```
{
    i++;                                 //i自加
    sum+=i;                              //累加
}
```

请同学们分析一下结果为多少?如果想得到正确的结果sum=5050,则如何修改源程序EX3-1-1.c?

任务3.2 用do-while语句实现的循环结构

任务说明

有时使用while语句实现循环时,循环体语句组一次都不能执行。这是因为,while语句在进入循环之前先判断循环条件。但在日常生活中,有时也需要先无条件执行一次循环体语句,然后再根据判断条件确定是否重复执行循环体语句。这时可以使用do-while语句实现循环控制。在本任务中,将学习do-while语句的格式、执行流程和应用方法。

相关知识

1. do-while语句格式

do-while语句也称为"直到"循环语句,它的格式为:

```
do
{
    循环体语句;
} while(表达式);
```

注意事项:

(1)"while(表达式)"后面的分号不能省。

(2) 当循环体语句仅由一条语句构成时,可以省略大括号{}。

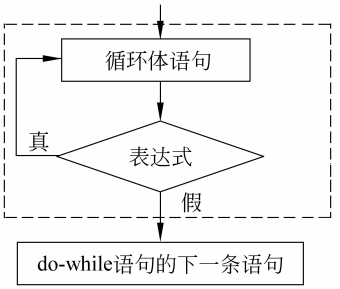

图 3-3　do-while 语句的执行流程

2. do-while 语句执行流程

do-while 语句执行流程如图 3-3 所示。

do-while 循环执行流程如下。

(1) 执行循环体语句。

(2) 求解表达式。如果表达式的值为非 0(真),则转向第(1)步继续执行;否则,转向第(3)步。

(3) 执行 do-while 的下一条语句。

do-while 循环语句的特点是:先执行循环体语句,然后再求解表达式,决定是否继续循环。

【例 3-1】 用 do-while 语句求 1+2+3+…+100 的值。

程序代码如下。

```
#include<stdio.h>
void main( )
{
    int i=1,sum=0;                          //循环控制变量 i 和累加器 sum 初始化
    do
    {
        sum+=i;                             //累加
        i++;                                //变量 i 自加
    }while( i<=100 );
    printf("sum=%d\n",sum);
}
```

程序运行结果:

```
sum=5050
```

注意:do-while 语句与 while 语句的区别。由于 while 语句是先判断表达式,后执行循环体语句,所以循环体语句有可能一次也不执行。由于 do-while 是先执行一次循环体语句,再判断表达式,所以循环体语句至少被执行一次。

任务实施

1. 任务功能

编写程序,计算数学式 $1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\cdots+\frac{1}{n}$ 的近似值。要求至少累加到 $1/n \leqslant 0.009\,84$ 为止。输出循环次数及累加和。

2. 编程思路

定义一个变量 n,取值分别为 1、2、3、4、…,可通过 n 自加来实现。定义一个变量 sum,初值为 0,用来存放累加后的结果,即近似值。n=1,sum=sum+1.0/n 得到数学式

的第1项；n＝2，当1/n＞0.009 84时，sum＝sum＋1.0/n得到数学式的第1项和第2项的和；……以此类推，直到1/n≤0.009 84时，累加结束，即可得到数学式的近似值。

流程图如图3-4所示。由图3-4可知，先执行sum＝sum＋1.0/n，然后判断表达式1/n＞0.009 84是否为真，决定是否继续循环。

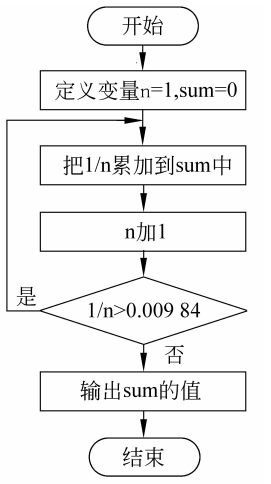

图3-4 计算数学式近似值的流程图

3. 源程序 EX3-2-1.c

```
#include<stdio.h>
void main( )
{
    int n=1,N;
    float sum=0;
    do{
        sum=sum+(float)1/n;              //将每一项累加到 sum 中
        n++;                             //变量 n 自加,求新的分母值
    }while((float)1/n>0.00984);
    N=n-1;                               //求循环的次数
    printf("N=%d\nsum=%.4f\n",N,sum);
}
```

4. 运行、调试

在VC++ 6.0开发环境下，编辑、编译和调试源程序EX3-2-1.c。程序运行结果为：

```
N=101
sum=5.1973
```

任务拓展

用while语句编写循环结构程序，重新计算本任务中的数学式$1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\cdots+\frac{1}{n}$的近似值。要求至少累加到1/n≤0.009 84为止。

任务3.3 用for语句实现的循环结构

任务说明

在3种循环语句中，for语句功能更强，最为灵活，不仅可用于循环次数已经确定的情况，也可用于循环次数虽不确定，但给出了循环继续条件的情况。本任务中将学习for语句的格式、执行流程和应用方法。

相关知识

1. for 语句格式

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

上述 for 语句格式中,“表达式 1”的功能是对循环变量赋初值,“表达式 2”的功能是判断循环变量是否满足继续循环的条件,“表达式 3”的功能是改变循环变量。

for 循环语句的执行流程如下。

(1) 求解表达式 1。

(2) 求解表达式 2。如果其值为非 0,执行第(3)步;否则,执行第(4)步。

(3) 执行循环体语句,并求解表达式 3,然后转向第(2)步。

(4) 执行 for 语句的下一条语句。

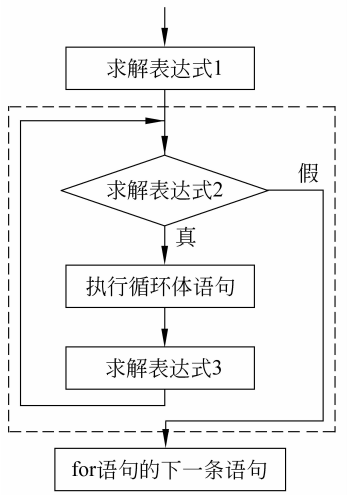

图 3-5　for 循环语句的执行流程

2. for 语句的执行流程

执行过程流程如图 3-5 所示。

【例 3-2】 用 for 语句求 1～100 的累计和。

程序代码如下。

```
#include<stdio.h>
void main( )
{
    int i,sum=0;                          //循环控制变量 i 和累加器 sum 初始化
    for(i=1;i<=100;i++)
    {
        sum+=i;                           //累加
    }
    printf("sum=%d\n",sum);
}
```

程序运行结果:

```
sum=5050
```

3. for 语句的几点说明

(1) “表达式 1”、“表达式 2”和“表达式 3”部分均可缺省,甚至全部缺省,但其间的分号不能省略,如 for(;;)

① 表达式 1 省略。例如:

```
    i=1;
```

```
for(;i<=100;i++)
    sun+=i;
```

由此可以看出表达式1并不是真的省略了，而是换了一个地方。

② 表达式3也可以省略。例如：

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

从这个意义来看，表达式3并没有省略，而是换了一个地方。如果一定要省略循环变量或改变这个表达式3，将不能使循环条件(表达式2)达到“假”，因而不能结束循环，该循环将成为死循环。

③ 表达式2省略。例如：

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

如果表达式2省略，就不判断循环条件，即认为循环条件始终为真，循环无终止地进行，如果没有别的办法退出循环，将成为死循环。循环条件2省略的流程如图3-6所示。

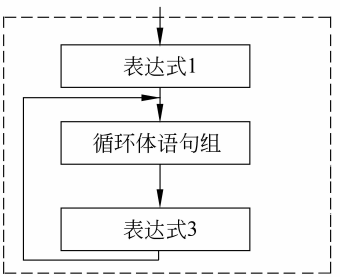

图3-6 循环条件省略流程图

(2) 当循环体语句仅由一条语句构成时，可以省略大括号{}。

(3) 表达式1，既可以是给循环变量赋初值的赋值表达式，也可以是其他表达式(如逗号表达式)。例如：

```
for(sum=0,i=1;i<=100;i++)
    sum+=i;
```

(4) 表达式2是一个逻辑量，除一般的关系(或逻辑)表达式外，也允许是数值(或字符)表达式。C语言将非0值看成是逻辑真，将0值看成是逻辑假。

任务实施

1. 任务功能

编写程序，求n的阶乘n!(n!=1×2×…×n)。

2. 编程思路

由任务要求可知n!=1×2×…×n，若从键盘输入n的值为5，则5! = 1×2×3×4×5=120。编程时，可首先设一个累乘器fact用来存放乘积，设初值为1。然后分别把1,2,…,n与fact相乘后赋值给fact。具体重复运算如下。

第1次：fact=fact * 1

第2次：fact=fact * 2

第3次：fact=fact * 3

……

第 n 次：fact＝fact * n

执行流程如图 3-7 所示。

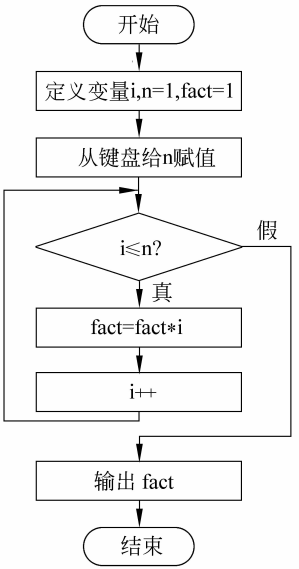

图 3-7　求 $n!$ 的流程图

3. 源程序 EX3-3-1.c

```
#include<stdio.h>
void main( )
{
    int i, n;
    long  fact=1;          //将累乘器 fact 初始化为 1
    printf("Input n: ");
    scanf("%d", &n);
    for(i=1;i<=n;i++)
        fact * =i;         //累乘
    printf("%d !=%ld\n", n, fact);
}
```

4. 运行、调试

在 VC++ 6.0 开发环境下，编辑、编译和调试源程序 EX3-3-1.c。程序运行结果为：

```
Input n: 5<回车>
5 !=120
```

任务拓展

用 while 循环求 n 的阶乘 n!(n!＝1×2×…×n)，体会与本任务中用 for 循环求阶乘 n! 的不同之处。

任务 3.4　循 环 嵌 套

任务说明

若在循环结构的循环体内，又出现了另一个完整的循环结构，则称为“循环嵌套”。嵌套式的结构表明各循环之间只能是“包含”关系，即一个循环结构完全在另一个循环结构的里面。通常把里面的循环称为“内循环”，外面的循环称为“外循环”。本任务主要学习循环嵌套编程的方法。

相关知识

C 语言的 3 种循环语句都可以嵌套，既可以自身嵌套也可以相互嵌套。循环嵌套的层数没有限制，但一般用得较多的是二重循环或三重循环。

1. for 循环嵌套

for 循环嵌套如图 3-8 至图 3-10 所示。

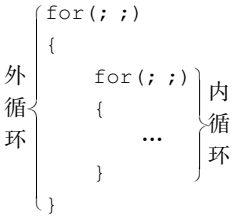

图 3-8 for 与 for 循环的嵌套

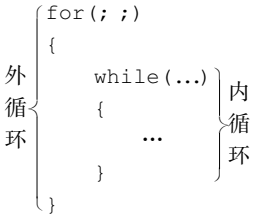

图 3-9 for 与 while 循环的嵌套

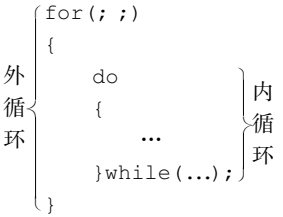

图 3-10 for 与 do-while 循环的嵌套

2. while 循环嵌套

while 循环嵌套如图 3-11 至图 3-13 所示。

图 3-11 while 与 while 循环的嵌套

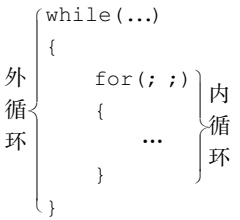

图 3-12 while 与 for 循环的嵌套

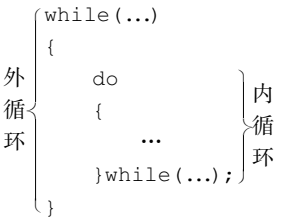

图 3-13 while 与 do-while 循环的嵌套

3. do-while 循环嵌套

do-while 循环嵌套如图 3-14 至图 3-16 所示。

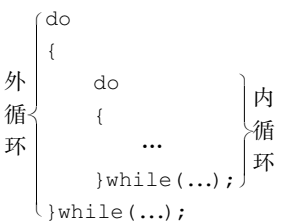

图 3-14 do-while 与 do-while 循环的嵌套

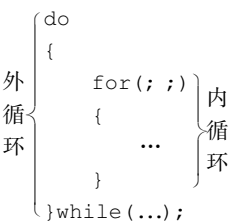

图 3-15 do-while 与 for 循环的嵌套

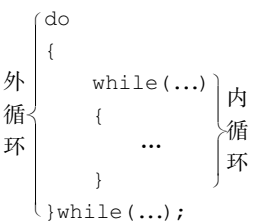

图 3-16 do-while 与 while 循环的嵌套

任务实施

1. 任务功能

用循环嵌套编写程序输出九九乘法口诀表。

2. 编程思路

乘法表的特点如下。

(1) 共有 9 行。

(2) 每行的式子数量很有规律,第几行就有几个式子。

（3）对于每一个式子，既与所在的行有关，又与所在的行上的位置（列）有关。首先看输出其中一行的情况。假设要输出的是第 i 行，我们知道 i 行共有 i 个式子，可用以下程序段实现：

```
for(j=1;j<=i;j++)
    printf("%3d*%d=%-2d",j,i, i*j);
printf("\n");
```

如果给上述程序段加一个外循环，使 i 从 1 取到 9，每执行一次内循环，就输出了乘法表中对应行的所有式子。执行流程如图 3-17 所示。

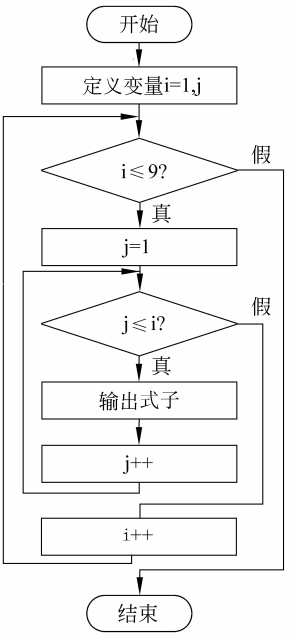

图 3-17　输出九九乘法口诀表执行流程

3. 源程序 EX3-4-1.c

```
#include<stdio.h>                    //包含头文件
void main( )
{
    int i,j,k=0;
    for(i=1;i<=9;i++)                //外循环执行 9 次
    {
        for(j=1;j<=i;j++)            //内循环每次的执行次数与 i 有关
        {
            k=i*j;
            printf("%3d*%d=%-2d",j,i,k);  //输出结果
        }
        printf("\n");                //换行
    }
}
```

4. 运行、调试

在 VC++ 6.0 开发环境下，编辑、编译和调试源程序 EX3-4-1.c。程序运行结果如下。

```
1*1=1
1*2=2  2*2=4
1*3=3  2*3=6   3*3=9
1*4=4  2*4=8   3*4=12  4*4=16
1*5=5  2*5=10  3*5=15  4*5=20  5*5=25
1*6=6  2*6=12  3*6=18  4*6=24  5*6=30  6*6=36
1*7=7  2*7=14  3*7=21  4*7=28  5*7=35  6*7=42  7*7=49
1*8=8  2*8=16  3*8=24  4*8=32  5*8=40  6*8=48  7*8=56  8*8=64
1*9=9  2*9=18  3*9=27  4*9=36  5*9=45  6*9=54  7*9=63  8*9=72  9*9=81
```

任务拓展

用双层 for 循环打印下面的图形。