第5章 循环结构程序设计

循环结构可以处理程序中需要被重复执行的操作。首先，通过一个例子说明在程序设计中使用循环结构的必要性。假设有如下程序：从键盘上输入10个整数，求它们的最大数。不难看出，如果只采用顺序结构和选择结构编写这个程序，将会出现冗长的代码，不仅执行效率低而且可读性差，使用循环结构则可以极大地提升程序的编写效率。

实现循环结构的语句称为循环语句，也叫重复语句。循环语句包括如下4类。

(1) while语句。

(2) do…while语句。

(3) for语句。

(4) 用goto语句和if语句构成循环。

本章重点：

(1) while语句、do…while语句、for语句的用法。

(2) 多重循环语句的用法及注意事项。

(3) break与continue语句的区别。

(4) 循环结构程序设计的方法。

本章难点：

(1) do…while语句中条件表达式的书写。

(2) 多重循环语句的执行流程。

(3) 循环结构算法的思维训练。

5.1 while语句

while语句也叫“当型循环语句”，其一般形式如下：

```
while(表达式)
{
    循环体语句
}
```

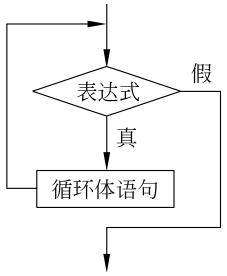

图5.1 while语句流程图

while语句流程图如图5.1所示。

当while语句执行时，首先计算“表达式”的值，若该值为假(即为0)，则终止执行while语句；否则，就执行“循环体语句”，执行完毕后，再次计算“表达式”的值，并根据该值的真假，决定

是否继续执行循环体语句，如此重复下去，直到“表达式”的值为假退出该循环结构。

【说明】

（1）while 语句括号中的“表达式”可以是任意类型表达式，必须用圆括号“()”括起来。

（2）“循环体语句”部分可以是单个语句，也可以是多个语句。如果是多个语句，必须用一对花括号“{ }”将它们括起来构成复合语句。

（3）while 语句的特点是：先执行表达式，后执行循环体。

【例 5.1】 从键盘上输入 10 个整数，求它们的最大数。

【分析】 对于需要使用循环结构的程序，读者需要弄清楚两个基本问题。

（1）需要被重复执行的操作是什么，换句话说，循环体是什么。

（2）循环体的执行需要满足的判别条件是什么。

基于以上两个问题，对这道题目的要求做出以下分析。

（1）要重复执行以下操作：输入一个数，把它与表示最大数的变量进行比较，如果它大于最大数，则将它的值赋值给最大数变量。

（2）要确保上述的循环体能够执行 10 次。那么，怎样确保这一点呢？

注意：表示最大值的变量的初始值应该如何处理？可以在循环结构之前先输入一个数，并将该数赋值给表示最大值的变量。为了能够确保输入的数据刚好是 10 个，循环体的执行次数将会相应地变为 9 次。为了便于更加清晰地理解该程序的执行过程，读者可以参照本程序的流程图，如图 5.2 所示。

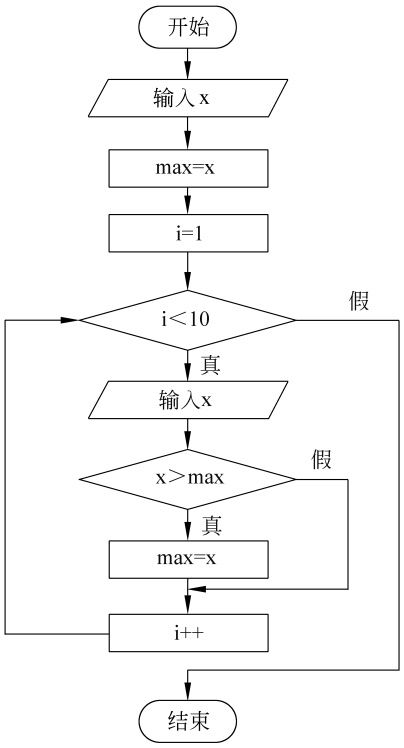

图 5.2　例 5.1 流程图

【程序】

```
#include <stdio.h>
int main()
{
    int x,max,i;
    scanf("%d",&x);
    max=x;
    i=1;
    while(i<10)
    {
        scanf("%d",&x);
        if(max<x)
            max=x;
        i++;
```

```
        }
        printf("%d",max);
        return 0;
    }
```


【运行结果】

```
1 3 6 9 0 2 8 7 4 2
9
```

下面再看一个题目。

【例 5.2】 输入两个正整数 m 和 n,求它们的最大公约数。

【分析】 从最大公约数的概念出发,要找到能够既能整除 m 又能整除 n 的最大数。具体步骤如下:

(1) 输入两个整数 m 和 n。

(2) 判断这两个整数是否为正整数,如果不是,则给出错误提示信息;如果是,则转向第(3)步。

(3) 将变量 min 的值设置为 m 和 n 的较小数。

(4) 置初值 i=2。

(5) 若 i 小于或等于 min,则执行(6),否则转向(8)。

(6) 判断 i 是否能够整除 m 和 n,若能,则把 i 的值赋给变量 t。

(7) i 的值自增。

(8) 输出结果。

【程序】

```
#include <stdio.h>
int main()
{
    int m,n,i,min,t;
    printf("请输入整数 m 和 n:\n");
    scanf("%d%d",&m,&n);
    if(m<=0||n<=0)
        printf("输入错误!");
    else
    {
        min=m<n?m:n;
        i=2;
        while(i<=min)
        {
            if(m%i==0&&n%i==0)
                t=i;
            i++;
        }
        printf("%d 和%d 的最大公约数是 %d\n",m,n,t);
    }
    return 0;
```

```
}
```

【运行】

```
请输入整数 m 和 n:
9 15
9 和 15 的最大公约数是 3
```

5.2 do…while 语句

do…while 语句也叫“直到型循环语句”,其一般形式如下:

```
do
{
    循环体语句
}while(表达式)
```

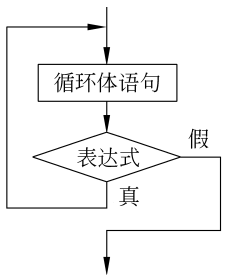

图 5.3 do…while 语句流程图

do…while 语句流程图如图 5.3 所示。

执行顺序如下:首先执行“循环体语句”,待“循环体语句”执行完毕后,再计算作为控制条件的“表达式”的值。若该值为真(即不等于 0),就再次执行“循环体语句”,否则就终止此循环。如此重复下去,直到“表达式”的值为假(即等于 0),则该循环语句执行完毕。

【说明】

(1) do…while 语句中的“表达式”可以是任意类型表达式,必须用圆括号“()”括起来。

(2) do…while 语句的特点是:先执行循环体,后判断表达式。

【例 5.3】 while 语句和 do…while 语句执行结果的比较。

(1) while 语句代码段。

```
#include <stdio.h>
int main( )
{
    int sum=0,i;
    scanf("%d",&i);
    while(i<5)
        sum+=i++;
    printf("sum=%d",sum);
    return 0;
}
```

(2) do…while 语句代码段。

```
#include <stdio.h>
int main( )
{
```

```
    int sum=0,i;
    scanf("%d",&i);
    do
    {
        sum+=i++;
    }while(i<5);
    printf("sum=%d",sum);
    return 0;
}
```

运行时,若两个程序都是输入1,则运行结果均为sum=10;如果都是输入5,则第一个程序得到结果sum=0,而第二个程序得到结果sum=5。

【例5.4】 输入一个正整数n,求n!。

【分析】 用变量jc表示阶乘计算的最终结果,并令其初值为1,然后把从2到n的所有整数逐个乘到jc中去。

(1) 输入一个正整数n。

(2) 置jc的初值为1,循环变量a的初值为1。

(3) 把a乘到jc中去,即执行jc*=a。

(4) a的值自增。

(5) 当a≤n时,转到(3),否则转到(6)。

(6) 输出jc的值。

【程序】

```
#include <stdio.h>
int main( )
{
    int n,a;
    long jc;
    printf("请输入一个正整数n:");
    scanf("%d",&n);
    a=1;
    jc=1;
    do
    {
        jc*=a;
        a++;
    }while(a<=n);
    printf("%d!=%ld\n",n,jc);
    return 0;
}
```

【运行】

```
请输入一个正整数n:5
5!=120
```

5.3 for 语 句

for 语句是一种使用频率较高的循环语句，其一般形式为：

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

for 语句流程图如图 5.4 所示。

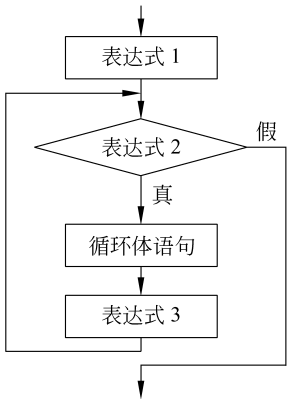

图 5.4 for 语句流程图

执行过程如下：

(1) 计算“表达式 1”。

(2) 计算并判断“表达式 2”，若该值为真(非 0)，则执行循环体语句；若为假(等于 0)，则转到第(5)步。

(3) 计算“表达式 3”。

(4) 转到第(2)步继续执行。

(5) 结束循环。

【说明】

(1) for 语句的圆括号内必须包含并且只能包含两个分号“;”。

(2) for 语句的 3 个表达式(即“表达式 1”“表达式 2”“表达式 3”)可以是任意类型表达式。“循环体语句”可以是一条语句，也可以是多条语句，如果是多条语句，必须用一对花括号“{ }”把它们括起来成为复合语句。

(3) “表达式 1”可以省略。比如：

```
for(;j<10;j++)
    x=x-j;
```

相当于

```
while(j<10)
{
    x=x-j;
    j++;
}
```

(4) “表达式 2”也可以省略。由于“表达式 2”起到判断并控制循环是否继续执行的作用，因此当它省略时，则不判断循环控制条件，也就是认为“表达式 2”始终为真，循环将无终止地执行下去。例如：

```
for(j=1; ;j++)
    x=x-j;
```

相当于

```
j=1;
while(1)
{
```

```
        x=x-j;
        j++;
    }
```


(5)“表达式3”也可以省略。例如：

```
for(j=1;j<10;)
    x=x-j;
```

相当于

```
j=1;
while(j<10)
  x=x-j;
```

【例5.5】 输入一个正整数，判断其是否为素数。

【分析】 素数除了能表示为它自己和1的乘积以外，不能表示为任何其他两个正整数的乘积。对于一个整数m，判断算法如下：

(1) 设置一个标志变量flag，令其初值为1。

(2) 将变量i的值置为2。

(3) 若$i>\sqrt{m}$，则转到(6)；若$i\leqslant\sqrt{m}$，则用m除以i，如果不能整除，则转到(4)，否则转到(5)。

(4) 将i的值自增，再转到(3)。

(5) 将flag的值置为0，再转到(4)。

(6) 如果flag等于1，则m是素数，否则就不是素数。

【程序】

```
#include <stdio.h>
#include <math.h>
int main( )
{
    int m,i,flag=1;
    printf("请输入一个正整数 m:");
    scanf("%d",&m);
    for(i=2;i<=sqrt(m);i++)
        if(m%i==0)
            flag=0;
    if(flag)
        printf("%d是素数\n",m);
    else
        printf("%d不是素数\n",m);

    return 0;
}
```

【运行】

```
请输入一个正整数 m:7
7 是素数
```

【说明】

(1) 在本例的程序中,使用库函数 sqrt()计算 m 的平方根,所以在程序的开头要声明它的头文件 math.h。

(2) 请读者参照 5.7 节中“break 语句”的作用思考一下,此程序能否进行简化?

【例 5.6】 有一个分数序列 2/1,3/2,5/3,8/5,13/8,21/13,…,求出这个数列的前 10 项之和。

【分析】 首先分析这个数列的规律:从第 2 个数开始,该数项的分母是前一个数项的分子,该数项的分子是前一个数项的分子与分母之和。算法如下。

(1) 用 s 表示各个数项之和,令其初值等于 0;用 a 表示数项的分子,初值赋为 2;用 b 表示数项的分母,初值赋为 1。

(2) 用 i 表示加到 s 中的数项的个数,初值为 1。

(3) 若 i>10,则转到(8);否则转到(4)。

(4) 将 a/b 加到 s 中。

(5) 将前一个数项的分子与分母之和赋值给后一个数项的分子。

(6) 将前一个数项的分子赋值给后一个数项的分母。

(7) 将 i 的值自增,再转到(3)。

(8) 输出结果。

【程序】

```
#include <stdio.h>
int main( )
{
    int i;
    float a=2.0,b=1.0,s=0;
    for(i=1;i<=10;i++)
    {
        s+=a/b;
        a=a+b;
        b=a-b;
    }
    printf("该数列前 10 项之和为:%f\n",s);
    return 0;
}
```

【运行】

```
该数列前 10 项之和为 16.479906
```

5.4 用 goto 语句和 if 语句构成循环

5.4.1 goto 语句

goto 语句也叫转向语句，用于将程序的执行流程转移到由 goto 语句指定的位置（某个标号语句处）。它的一般形式为：

```
goto  语句标号;
```

其中，语句标号由标识符表示。

使用 if 语句和 goto 语句可以构成循环结构。但是，这样的循环形式不仅容易造成程序结构的混乱，而且容易降低程序的可读性，因此，在结构化程序设计中应当尽量避免使用 goto 语句。

5.4.2 带标号语句

带标号语句的一般形式为：

```
标号:语句;
```

带标号语句的作用是便于其他语句将程序的执行流程转移到标号所标记的位置。

【例 5.7】 用 if 语句和 goto 语句构成循环，求解 $\sum_{i=1}^{10} i$。

本题比较简单，在此直接给出参考程序：

```
#include <stdio.h>
int main( )
{
    int a=1,s=0;
    loop:
    if(a<=10)
    {
        s+=a;
        a++;
        goto loop;
    }
    printf("1 到 10 的和等于%d\n",s);
    return 0;
}
```

5.5 循环的嵌套

如果循环语句的循环体内包含另一个完整的循环结构，则称为循环的嵌套。如果一个循环体内只嵌套一层循环，这种结构就称作二重循环。对于二重循环而言，处于内部的循环叫作

内循环，处于外部的循环叫作外循环。前几节介绍的几种循环语句均可以互相嵌套。对于嵌套的循环结构，应当特别注意内外循环的执行过程。下面举例说明嵌套循环的程序设计方法。

【例 5.8】 编程实现下面图形的输出。

```
    *
   ***
  *****
   ***
    *
```

【分析】 对于要输出的这个图形，应当考虑输出行数、每行星号的个数、每行星号的位置 3 个因素。经过分析发现，第 i(－2≤i≤2)行星号的个数等于 5－2＊abs(i)。对于每行星号的位置，取决于它的前导空格个数，第 i 行的前导空格数等于 abs(i)。需要说明的是，abs(i)表示 i 的绝对值。算法如下。

(1) 将 i 的初值设置为－2。

(2) 当 i＞2 时，转向(7)，否则转向(3)。

(3) 输出 abs(i)个空格。

(4) 输出 5－2＊abs(i)个星号。

(5) 输出换行符。

(6) i 的值自增，并转向第(2)步。

(7) 程序结束。

【程序】

```
#include <stdio.h>
#include <math.h>
int main( )
{
    int i,j;
    for(i=-2;i<=2;i++)
    {
        for(j=1;j<=abs(i);j++)
            printf(" ");
        for(j=1;j<=5-2*abs(i);j++)
            putchar('*');
        printf("\n");
    }
    return 0;
}
```

【运行】

```
    *
  * * *
* * * * *
  * * *
    *
```