第5章

迭代计算与循环结构

5.1 概　　述

利用分支语句或条件表达式可以实现从两个或3个数中找出最大数。但是当有更多的数据(例如50个数据或500个数据)时,如何找到其中的最大数呢?下面是一个最基本的解决思路。

步骤1:对第一个数,假设它可能是最大的数,将其标记为当前最大数。

步骤2:对下一个数,如果比标记的数大,则标记该数为当前最大数,否则什么都不做。

步骤3:重复步骤2,直到处理完所有的数。

步骤4:当前标记的数就是要找出的最大数。

其中,步骤2多次重复操作。程序设计将此类多次重复处理的逻辑结构称为循环结构,并引入循环语句实现循环结构。

循环结构一般可以表述为"当条件成立时,反复执行特定的操作序列,如果条件不成立则结束循环"(称为前判断结构、当型循环),或者表述为"执行特定的操作序列,如果条件成立,反复执行操作序列,直到条件不再成立时结束循环"(称为后判断结构、直到型循环)。其中的"条件"称为循环条件,可以为关系表达式、逻辑表达式及其他有确定值的表达式。条件作为一个判断结果,只要是非0值,其逻辑值就为真,否则为假。"操作序列"称为循环体,循环体作为一个顺序执行的操作序列,可以是空语句、表达式语句、块语句、选择结构语句或循环结构语句。在循环体执行过程中,如果不能改变循环条件的值,循环将永远执行,从而失去循环处理的意义。所以,循环体中通过修改与循环条件相关的变量值来改变循环条件,该变量称为循环控制变量。

当型循环结构的特点是先判断循环条件,循环体可能执行多次,也有可能一次也不执行;直到型循环结构的特点是先执行一次循环体,再判断循环条件,循环体至少会被执行一次。C语言提供的循环语句包括while、for和do…while。在某些特殊情况下,还可以通过break、continue、goto语句等C语言语句中断循环,直接改变循环迭代的执行过程。

5.2 while 语句

while 语句实现当型循环，先判断循环条件，后执行循环体语句。循环体可能一次也不执行。

语句格式：

```
while (循环条件表达式)
{
    语句块;
}
```

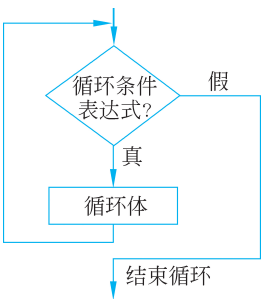

图 5-1 while 语句流程

while 为关键字，语句块为循环体。执行流程如图 5-1 所示。检查循环条件表达式，如果表达式值为真，执行循环体；否则，循环结束。

一般不能事先确定 while 语句控制循环的次数，需要根据循环条件判定，如果开始的循环条件为假，则循环体一次也不执行。

由多条语句构成的循环体务必放置在“{}”中，否则将导致逻辑上的错误。

例 5-1 编写一个程序，输出 1～100 的自然数之和。

```
#include<stdio.h>
int main()
{
    int i, sum=0;
    i=1;
    while (i<=100)                    /* 循环控制条件 i<=100 */
    {
        sum=sum+i;                    /* 累加和，还可以写成 sum+=i; */
        i++;                          /* 循环控制变量改变 i++ */
    }
    printf("%d\n",sum);
    return 0;
}
```

这是一个累加求和问题。声明 int 型变量 sum 用于存储计算自然数总和的值，其初始值设置为 0。如果没有初始化，声明 sum 时为其分配的存储空间内为随机值，该值将作为初始值参与求和计算从而导致计算结果错误。声明 int 型的循环控制变量 i 并初始化为 1，循环体内通过 i 值的自增将 1，2，…，100 加入到 sum 中。while 语句是否继续循环执行的条件，即判断循环控制变量 i 的值是否小于或等于 100。i++实现 i 从 1 至 100 的变化，每次变化满足 i 小于或等于 100 的循环条件，则执行循环体，并将当前变量 i 值累加到变量 sum 中。当 i 大于 100(此时 i 的值为 101)时，循环条件 i<=100 为假，循环终止，程序转而执行循环体后面的语句。

决定循环执行次数的因素包括循环条件、循环控制变量的初值和终值、循环控制变量每次变化的幅度等。循环控制变量每次增加或减少的值称为步长。如果循环控制变量具

有固定的步长，则循环次数的计算公式如下：

循环次数＝(终值－初值)/步长+1

例 5-1 中循环控制变量 i 从 1 循环到 100，步长为 1，循环次数为(100-1)/1+1=100 次。步长也可以是负数，改写例 5-1 为如下代码可得到同样的结果：

```
#include<stdio.h>
int main()
{
    int i,sum=0;
    i=100;                          /*初始值为 100*/
    while (i>=1)                    /*循环控制条件 i>=1*/
    {
        sum=sum+i;                  /*循环控制变量 i 参与求和计算*/
        i--;                        /*循环控制变量自减*/
    }
    printf("%d\n",sum);
    return 0;
}
```

如果循环体只有一条语句，可以省略包含循环体的花括号“{}”。常用的 while 语句有两种简单表示形式。

(1) 循环体为空语句。例如：

```
i=0;
while (++i<10);   /*判断循环条件是否成立的同时计算循环控制变量 i,即 i 先自增(++1)
                    再判断是否满足条件 i<10。若满足条件执行空语句;当++i 为 10 时,
                    i=10,循环条件不再成立,结束循环*/
```

(2) 循环体为表达式语句。例如：

```
i=0;
while (i<10)
++i;              /*当条件 i<10 成立时,执行循环体++i,直到 i=10,循环条件不再成立,
                    结束循环*/
```

但是仍然建议初学时，即使循环体只有一条语句也使用花括号包含该语句。例如：

```
int i=1;
while (i<=10)
{
    sum=sum+i++;          /*循环控制变量 i 参与求和计算*/
}
```

此外，while (1)表示循环条件为逻辑真，构成一个无限循环。

例 5-2 编写程序，输入 10 个整数，输出其中最大的数。

```
#include<stdio.h>
int main()
{
    int a,i,max;
    scanf("%d",&a);
    max=a;                          /*假设第一次输入的数为当前最大数*/
    i=1;
    while (i<10)
    {
        scanf("%d",&a);
        if (max<a)
```

```
            max=a;                          /*第 i 次输入的数为当前最大数*/
        i++;
    }
    printf("最大的数是%d\n",max);
    return 0;
}
```

首先，输入一个数并假设它为当前最大数，存储在 max 中；其次，依次输入其他 9 个数，对于每一次输入，都与之前输入数据中的最大值 max 进行比较，如果大于 max，则将此次输入数据存储于 max 中；循环执行结束后 max 中存储的即所有数中的最大数。如果需要找出 50 个数中的最大数，只需将上述程序中的 while(i<10)改为 while(i<50)即可。

设计循环结构时，通常将循环控制变量变化语句放在循环体的最后一行，虽然这不是语法上的强制要求，但有利于程序的阅读和理解。例 5-2 中的循环变量 i 所在语句只起到计数作用，并没参与计算，放在循环体的开始或结束位置效果是相同的。例 5-1 中的循环变量 i 所在语句也可以放在循环体的开始处，程序修改为：

```
#include<stdio.h>
int main()
{
    int i, sum=0;
    i=0;                    /*i 要从 0 开始*/
    while (i<=99)           /*也可写成 while(i<100)*/
    {
        i++;
        sum=sum+i;          /*循环控制变量 i 参与求和计算*/
    }
    printf("%d\n",sum);
    return 0;
}
```

程序中需要修改循环控制变量 i 的值(初始化值为 0)，循环的判定条件也要做相应修改(i<=99)，显然程序的可读性不如例 5-1 所示的程序。

5.3 do…while 语句

do…while 语句实现直到型循环，先执行一次循环体语句，再判断循环条件。即使循环条件不满足，循环体也至少被执行一次。

语句格式：

```
do
{
    语句块;
}while (循环条件表达式);
```

do、while 为关键字，语句块为循环体。while(循环条件表达式)的后面必须有分号。执行过程如图 5-2 所示。首先顺序执行循环体中的语句，然后检查循环条件表达式，如果表达式值为真，再次顺序执行循环体；否则，

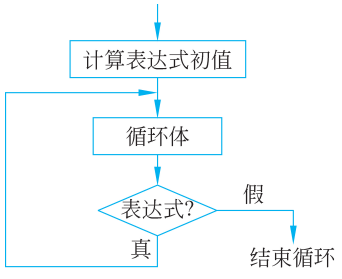

图 5-2 do…while 语句流程图

循环结束。

利用 do…while 语句改写例 5-1：

```
#include<stdio.h>
int main()
{
    int i, sum=0;
    i=1;
    do
    {
        sum=sum+i;          /*循环控制变量 i 参与求和计算*/
        i++;
    } while (i<=100);       /*循环控制条件分号;不能缺省*/
    printf("%d\n",sum);
    return 0;
}
```

do…while 语句同样有两种简单表示形式。

(1) 循环体为空语句。例如：

```
i=0;
do
{
}while (++i<10);     /*当++i 为 10 时,循环条件++i<10 不再成立,i=10 且循环结束*/
```

(2) 循环体为表达式语句。例如：

```
i=0;
do
{
    i++;
}while (i<10);       /*当++i 为 10 时,循环条件++i<10 不再成立,i=10 且循环结束*/
```

即使循环开始时没有满足循环条件,循环体仍然会执行一次。例如：

```
int i=100;
    do{
      printf("输出结果: i=%d\n",i);          /*输出结果: i=100*/
    }while(i<=10) ;
```

例 5-3 编写程序,输入一个整数,输出这个数的位数。

```
#include<stdio.h>
int main()
{
    int n,count=0;
    scanf("%d",&n);
    while (n!=0)
    {
        count++;
        n=n/10;
    }
    printf("这是一个%d 位整数。\n",count);
    return 0;
}
```

判断一个数的位数时,无论从左向右还是从右向左统计,都需要记住已经检查过的每一位数,可以利用位数计数器计数(加 1 操作),直到统计完所有的位数,位数计数器的值就是统计结果。基本解决思路如下：

(1) 输入一个整数。

(2) 计数加 1,去除一位,处理剩余位数。

(3) 重复执行步骤(2),直到没有剩余位数。

(4) 输出计数结果。

实际上,因为不知道左侧第一个数字是第几位(否则就不必编写程序了),所以从左侧开始去除一个位数很难。然而利用 C 语言整数除法的规则,从右侧去除一个数字的方法却很简单,只需要除 10 取整即可。假设输入 1234,第一次循环,count 值为 1,n 的值变为 123;第二次循环,count 值为 2,n 的值变为 12;第三次循环,count 值为 3,n 的值变为 1;第四次循环,count 值为 4,n 的值变为 0;此时不满足循环条件,循环结束(计数结果 4)。但是上述代码隐含着一个小错误,即当输入 0 时程序输出的计数结果为 0。因为当 n==0 时,由于不符合循环条件而没有执行循环体。虽然可以在循环之前加上一个 if 语句来处理 n 等于 0 的情况,例如:

```
if (n==0)
    count=1;
```

但是,使用 do…while 语句是一个更好的选择。例如:

```
#include<stdio.h>
int main()
{
    int n,count=0;
    scanf("%d",&n);
    do
    {
        count++;
        n=n/10;
    } while (n!=0);
    printf("这是一个%d位整数。\n",count);
    return 0;
}
```

当必须执行一次循环体时,采用 do…while 语句更好;而当有可能不需要执行循环体时,选择 while 语句更好。

5.4 for 语句

for 语句是使用最广泛的一种循环控制语句,特别适合已知循环次数的情况。

1. for 语句

语句格式:

```
for (表达式 1; 表达式 2; 表达式 3)
{
    语句块;
}
```

表达式 1 通常为赋值表达式,用于对循环控制变量进行初始化,又叫初值表达式;表

达式 2 为循环条件表达式；表达式 3 实现对循环控制变量的修改，多数情况采用++/--表达式；语句块是循环体。执行过程如图 5-3 所示。首先，计算表达式 1 的值，作为循环控制变量初值。其次，判断表达式 2 是否成立，如果成立，顺序执行循环体，否则退出循环。每一次循环体语句执行结束时，都要重新计算表达式 3 的值，然后重新判断表达式 2 是否成立，根据判断结果决定是否继续执行循环体。

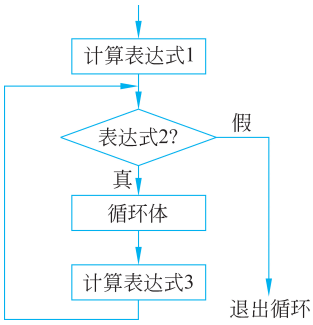

图 5-3　for 语句流程图

利用 for 语句改写例 5-1：

```
#include<stdio.h>
int main()
{
    int i, sum=0;
    for (i=1;i<=100;i++)
    {
        sum=sum+i;
    }
    printf("%d\n",sum);
    return 0;
}
```

i 为循环变量，赋值表达式 i=1 为循环控制变量 i 赋初值；表达式 i<=100 为循环控制条件；表达式 i++保证每一次循环之后改变循环变量 i 的值，使循环能够正常结束。

for 循环中常见的错误是循环控制条件使用不正确的表达式。例如，将上面代码中的 i<=100 错误地写成 i<100，则循环只进行 99 次。一个有效的解决方法是将问题的最终结果值包含在循环条件中。

例 5-4　编写程序，计算 n 的阶乘 $n!$（$n!=1\times2\times\cdots\times n$）。

```
#include<stdio.h>
int main()
{
    int i, n;
    long fact=1;
    scanf("%d", &n);
    for (i=1; i<=n; i++)
        fact=fact*i;
    printf("%d! =%ld\n", n, fact);
    return 0;
}
```

计算 $n!$的过程是一个累乘求积的过程。调用 scanf 函数输入 n 的值，设 int 型循环变量 i，i 从 1 至 n 循环乘入 fact 中。为防止累乘结果溢出，采用 long int 型声明累乘器变量 fact。

2. for 语句的常用省略格式

使用 for 语句时，经常会省略 for 语句中的某些表达式，for 语句常用的几种省略格式如下：

(1) for(;;)。for 语句的 3 个表达式都省略(注意，分号绝对不能省略)，这是一个无

限循环语句，与 while(1)的功能相同，需要通过其他方式(如 5.7.1 节的 break 语句)结束循环。

(2) for(;表达式 2;表达式 3)。省略表达式 1，可以将表达式 1 写在 for 语句结构的外面。例如：

```
i=1;                          等价于        for(i=1; i<=n; i++)
for (;i<=n; i++)                                fact=fact * i;
    fact=fact * i;
```

如果循环控制变量的初值不是确定的值，而是需要通过 for 语句前面语句的执行计算得到，可使用该形式。

(3) for(表达式 1;表达式 2;)。省略表达式 3，C 语言允许在循环体内改变循环控制变量的值。该格式一般在循环控制变量呈非规则性变化，并且在循环体中有更新循环控制变量的语句时使用。例如：

```
for (n=1;n<=100;)
{
    …
    n=3 * n-1;          /* 循环控制变量的变化为 1,2,5,14,… */
    …
}
```

(4) for(逗号表达式 1;表达式 2;逗号表达式 3)。表达式 1 和表达式 3 可以是逗号表达式。例如：

```
for (n=1,m=100;n<m;n++,m--)
{
    …
}
```

表达式 1 同时为 n 和 m 赋初值，表达式 3 同时改变 n 和 m 的值。循环可以有多个控制变量，逗号表达式可以与循环有关，也可以与循环无关。

(5) for (表达式 1;;表达式 3)。省略表达式 2，即循环条件一直成立，等价于 while(1)格式。

3. 注意事项

使用 for 语句还需要注意以下几点。

(1) 循环初始值(表达式 1)、循环的条件(表达式 2)和循环控制变量改变语句(表达式 3)中可以包含算术表达式。例如：

```
n=10;                              等价于        for(i=11;i<=100;i++)
for (i=n+1;i<=n * 10;i++)
```

(2) 尽管可以在 for 语句的循环体中修改循环控制变量的值，但在一般情况下，循环控制变量仅用来控制循环过程，在循环体内尽量不改变其值，以免导致令人费解的结果和错误。

(3) 表达式 3 可以自增/自减，或是加/减一个整数等多种形式。例如：

```
for (i=100;i>=1;i--)          /* 循环控制变量从 100 递减到 1 */
for (i=0;i<=10;i+=2)          /* 循环控制变量从 0 变化到 10,每次增加 2 */
for (i=10;i>=0;i-=2)          /* 循环控制变量从 10 变化到 0,每次减少 2 */
```

当进行递增操作时，循环向上计数，表达式 1 的值要小于表达式 2 的值；当进行递减操作时，循环向下计数，表达式 1 的值要大于表达式 2 的值；否则将造成死循环。

(4) for 语句和 while 语句都是前判断结构，二者可以互相转换。

由于 for 语句将控制循环的 3 个要素写在一行中，看起来更为直观，因此更受欢迎。例如：

```
for (表达式 1;表达式 2;表达式 3)        等价于        表达式 1;
{                                                    while (表达式 2)
    语句块;                                          {
}                                                        语句块;
                                                         表达式 3;
                                                     }
```

例 5-5 编写程序，统计某班 30 名学生 C 语言课程的平均成绩。

```
#include<stdio.h>
int main()
{
    double aver,sum;
    int i,score,n=30;
    sum=0;
    for (i=1; i<=n; i++)
    {
        scanf("%d",&score);          /*读入第 i 个学生成绩*/
        sum=sum+score;               /*成绩累加处理*/
    }
    aver=sum/n;                      /*计算平均成绩*/
    printf("班级的平均成绩是 %5.1 分。\n",aver);
    return 0;
}
```

例 5-5 程序对已经确定人数的学生进行成绩处理，已知道循环次数，适合使用 for 循环。循环控制变量的变化范围为 1～30，每执行一次循环体，读入一个数据并累加一次，直到 30 个数据全部输入后循环结束，计算全班的平均成绩。程序输出计算结果时采用的格式%5.1 表示占 5 位空间的双精度数且只保留一位小数。

5.5 循环语句对比

循环的本质就是当循环条件成立时反复执行循环体。C 语言可以实现计数式循环和标记式循环两种循环处理。

计数式循环用于处理知道准确循环执行次数的循环过程，又称为定数循环。在计数式循环过程中，循环控制变量用来计算循环的次数。在每次执行循环体语句后循环控制变量的值都要发生变化(递增或递减)，当循环控制变量的值经改变达到了预定的循环次数后结束循环。例 5-4 和例 5-5 都是计数式循环。

标记式循环适用于处理循环次数未知的循环过程，又称为不定数循环。在标记式循环过程中，由于事先不知道准确循环的次数，因此需要在循环体中包含获取数据的语句，以期望在某次数据处理后因不满足循环继续执行的条件而终止循环。这个确保退出循环

的数据被称为标记值。标记值是利用一个专门的数据表示正常处理数据的结束，应该在处理所有合法的数据后提供给程序，因此标记值必须不同于正常数据值。例 5-3 是标记式循环。

while、do…while、for 这 3 种循环语句形式各不相同，相互之间有一定区别，但主要的结构成分都是循环体和循环控制条件。

(1) while、do…while 循环一般用于标记式循环(循环次数未知)，for 循环通常用于计数式循环(循环次数已知)。

(2) while、do…while 循环通常将循环结束的条件放在 while 后面的表达式中，在循环体中除反复执行的操作语句外，还应包含能够保证循环结束的语句(例如 i++等循环变量控制语句)，for 循环则用表达式 2 和表达式 3 表示循环结束的条件以及循环控制变量的变化。

(3) while、for 循环是前判断结构。如果循环条件一开始就不满足，则不执行循环体而直接结束循环。因此循环体可能被执行 0 次(循环条件不满足)或多次(循环条件满足)。do…while 循环是后判断结构，循环条件无论是否成立，循环体至少要被执行 1 次。

(4) 采用 while、do…while 循环时，循环变量的初始值操作一般在 while 和 do…while 语句之前完成，for 语句中循环控制变量的初始化由表达式 1 实现。

3 种循环语句都是根据循环条件决定是否重复执行，所以在循环体内部或循环条件中必须存在改变循环条件的语句，否则会出现死循环等异常情况。实际应用中，同一个问题既可以用 while 语句解决，也可以用 do…while 语句或 for 语句解决，3 种循环语句格式之间可以相互转化。选用的一般原则如下。

- 如果循环次数在执行循环体之前就已确定，一般用 for 语句；如果循环次数是由循环体的执行情况确定，则采用 while 语句或 do…while 语句。
- 当循环体至少要执行 1 次时，采用 do…while 语句；反之则选用 while 或 for 语句。

5.6 循环嵌套

一个循环语句的循环体内包含另一个完整的循环语句，称为循环嵌套结构。3 种循环语句 while、do…while 和 for 语句可以互相嵌套，但被嵌套的循环结构一定是一个完整的循环结构，即两个嵌套的循环结构之间不能相互交叉。循环嵌套结构中，每一层循环的循环体都应该使用花括号“{}”括起来，即使循环体只有一条语句也应使用“{}”，以防止二义性。

```
do
{
    …
    for (;;)          /* do…while 结构与 for{…}结构发生交叉，错误 */
    {
        …
    }while ();
}
```