第5章 循环结构

【学习要点】

在程序设计中，如果一组操作需要反复执行，可用循环结构实现。C语言中实现循环结构的控制语句有三种：while语句、do-while语句和for语句。这三种语句在功能和用法上各有特点，熟练掌握将有助于复杂程序的编写。

- 循环语句：while语句、do-while语句、for语句。
- 流程控制语句：break语句、continue语句、goto语句。
- 嵌套循环。


while语句
课堂练习

5.1 while语句

5.1.1 while语句

while语句用于实现**当型循环**结构，语法格式为：

```
while(条件)
    循环体
```

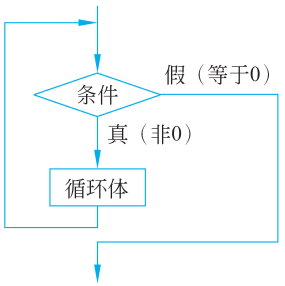


图5.1 while语句的流程图

while语句的执行过程是：**先判断循环条件**，如果条件成立(值不为0)则执行循环体，然后重复“判断条件-执行循环体”的过程，直到循环条件不成立(值为0)时退出while语句，结束循环，然后执行while语句的后续语句，如图5.1所示。

说明：

(1) 在循环条件为真的情况下，需要重复执行一个语句序列，被重复执行的语句序列称为**循环体**。循环体可以是单条语句或语句组，语句组是指两条或两条以上的语句，需要用花括号{ }括起来，称为**复合语句**。

(2) while语句的循环体有可能一次都不会被执行。

(3) 如果循环条件永远为真，循环将变成无限循环，又称为“**死循环**”。编程时应避免陷入“死循环”。

5.1.2 while 语句的应用

【例 5.1】 计算 1～n(含边界值)的所有整数之和，其中 n 为正整数，由键盘输入。

【问题分析】

要计算 1＋2＋3＋…＋n 的结果，实际上就是多次进行两个数相加的操作(操作次数与 n 有关)，只是每次相加的被加数和加数不一样。

规划数据结构如下：

(1) 定义 int 型变量 sum，用于存储求和结果，同时也作为被加数；

(2) 定义 int 型变量 i，用于存储加数，通过循环，i 依次取值 1、2、3、…、n。

算法的 N-S 流程图如图 5.2 所示。

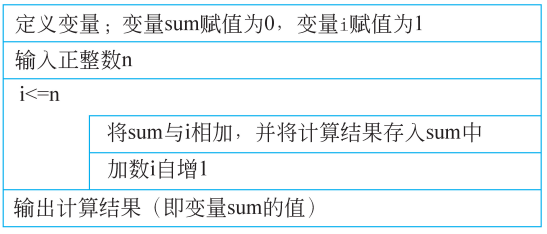
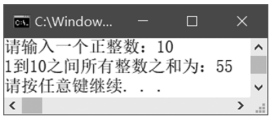


图 5.2 例 5.1 算法的 N-S 流程图

【编程实现】

```
#include <stdio.h>
int main()
{
    int n,i=1,sum=0;
    printf("请输入一个正整数：");
    scanf("%d",&n);
    while(i<=n)
    {
        sum=sum+i;
        i=i+1;
    }
    printf("1 到%d 之间所有整数之和为：%d\n",n,sum);
    return 0;
}
```

【运行结果】



【关键知识点】

(1) 循环体由两条语句构成,所以需要加{ }构成复合语句,如代码第 8～11 行所示。

(2) 代码第 10 行 i=i+1 修改了变量 i 的值,每执行一次循环,变量 i 的值增加 1。这样,当循环到一定次数的时候,循环条件 i<=n 不再成立,从而结束循环。

【延展学习】

(1) 如果只计算 1～n 的奇数和,该如何修改程序呢?

(2) 如果要计算 1～n 的所有整数的乘积,即求 n 的阶乘,该如何修改程序呢?

(3) 如果要计算实数 i 的 n 次方,该如何修改程序呢?

【例 5.2】 输入一个正整数,求该数的各位数字之和。

【问题分析】

由于不知道输入的正整数有几位,所以只能从个位开始,向高位方向逐一取出各位数字。以整数 2345 为例,第一次循环通过 2345%10 得到 2345 的个位数 5,通过 2345/10 去掉最低位得到 234;第二次循环通过 234%10 取出 2345 的十位数 4,通过 234/10 去掉最低位得到 23;同理,第三次循环得到 2345 的百位数 3 和整数 2;第四次循环得到 2345 的千位数 2,且此时 2/10 变为 0,说明已经取完整数所有数位的数字,循环结束。

规划数据结构如下:

(1) 定义 int 型变量 number,用于存储输入的正整数;

(2) 定义 int 型变量 sum,用于存储正整数各位数字之和,初始值为 0。

算法的 N-S 流程图如图 5.3 所示。

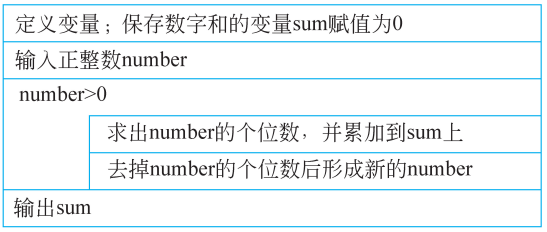
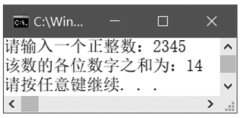


图 5.3 例 5.2 算法的 N-S 流程图

【编程实现】

```
#include <stdio.h>
int main()
{
    int number,sum=0;
    printf("请输入一个正整数: ");
    scanf("%d",&number);
    while(number>0)
    {
        sum=sum+number%10;
        number=number/10;
    }
    printf("该数的各位数字之和为: %d\n",sum);
    return 0;
}
```

【运行结果】



【关键知识点】

(1) number%10 可以得到正整数 number 的个位。

(2) number/10 可以去掉正整数 number 的个位。

【延展学习】

(1) 如果要得到正整数的反序数(也称逆序数,例如 2345 的反序数为 5432),该如何编程实现?

分析:以 number=2345 为例,第 1 次取出个位数 5,进行计算 0 * 10+5 得到 5 后存入 sum 中;第 2 次取出十位数 4,进行计算 5 * 10+4 得到 54 后存入 sum 中;第 3 次取出百位数 3,进行计算 54 * 10+3 得到 543 存入 sum 中,以此类推,直到 number 等于 0 为止。因此,将例题代码的第 9 行修改为 sum=sum * 10+number%10; 即可求出 number 的反序数,再适当修改第 12 行代码中的提示信息即可得到正确的运行结果。

(2) 如果要判断某正整数是否为回文数(正读和反读都一样的数,例如 1231 不是回文数,1221 是回文数),该如何编程实现?

分析:先求出正整数 number 的反序数 sum,再判断原数与反序数是否相等即可。需要注意的是,当循环结束时,number 等于 0,原数已不存在,无法判断原数与 sum 是否相等。因此需要在循环开始前,将 number 的值赋值给变量 backup,循环结束后,再判断 backup 与 sum 是否相等。

5.2 do-while 语句


do-while 语句
课堂练习

5.2.1 do-while 语句

do-while 语句用于实现**直到型循环**,流程图如图 5.4 所示,语法格式为:

```
do
    循环体
while(条件);
```

do-while 语句的执行过程:**首先执行循环体**,然后重复“判断条件-执行循环体”的过程,直到循环条件不成立为止,退出 do-while 语句,结束循环,执行 do-while 语句的后续语句。

说明:

(1) do-while 语句的循环体可以是单条语句,也可以是加花括号后构成的复合语句。

(2) while(条件)后面的分号是 do-while 语句的结束标志,**不能省略**。

(3) do-while 语句的**循环体至少执行一次**。

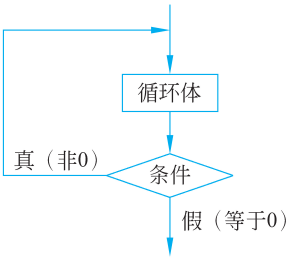


图 5.4　do-while 语句流程图


求圆周率
微视频

5.2.2　do-while 语句的应用

【例 5.3】　利用莱布尼茨公式 $\pi=4\left(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\cdots+(-1)^{n-1}\frac{1}{2n-1}+\cdots\right)$ 计算圆周率，直到最后一项的绝对值小于 0.00003 时结束运算。

【问题分析】

与例 5.1 类似，本例也是累和计算，但各累加项（也称为通项）正负交替变化。可以定义一个变量用来存储 1 或－1，代表通项的值为正或负，使用该变量乘以通项的值，可实现通项的正负交替变化。

规划数据结构如下：

（1）定义 double 型变量 sum，用于存储累加和（被加数）。

（2）定义 double 型变量 term，用于存储通项（加数）的绝对值。

（3）定义 int 型变量 sign，用于存储实现正负符号的数值 1 或－1。

算法的 N-S 流程图如图 5.5 所示。

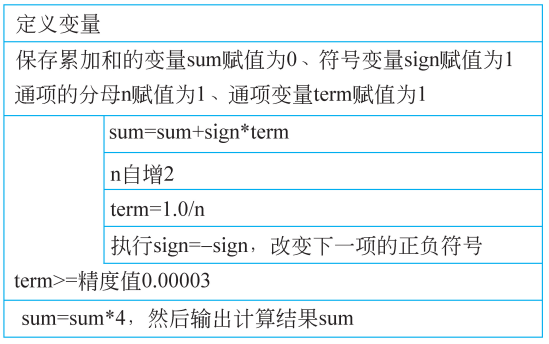


图 5.5　例 5.3 算法的 N-S 流程图

【编程实现】

```
1    #include <stdio.h>
2    int main()
3    {
4        double sum=0,term=1.0;
```

```
5       int sign=1,n=1;
6       do
7       {
8           sum=sum+sign*term;
9           n=n+2;
10          term=1.0/n;
11          sign=-sign;
12      }while(term>=3e-5);
13      sum=sum*4;
14      printf("PI=%f\n",sum);
15      return 0;
16  }
```

【运行结果】

【关键知识点】

(1) 第 10 行 term=1.0/n,其中分子 1.0 是 double 类型,分母 n 是 int 类型,计算时自动进行类型转换,将 n 的值转换为 double 类型,从而得到 double 类型的除法结果。若写成 term=1/n,则进行的是整除运算,如 1/5 得 0,不是 0.2。

(2) 第 11 行,对变量 sign 进行取负运算,从而实现正负交替变化。

(3) 本题的计算结果 3.141653,和圆周率的值有较大误差,若需减小误差,可修改循环条件,例如,将循环条件修改为 term>=1e-8,则计算结果为 3.141593。

【延展学习】

(1) 如果不使用变量 sign,而是用 if 语句处理通项正负符号的问题,即经过判断后,决定执行 sum=sum+term 或 sum=sum-term 的操作,该如何修改程序?

(2) 用沃利斯公式计算圆周率 $\frac{\pi}{2}=\frac{2}{1}\times\frac{2}{3}\times\frac{4}{3}\times\frac{4}{5}\times\frac{6}{5}\times\frac{6}{7}\cdots$该如何编程实现?

【例 5.4】 编程模拟计算机进行进制转换的过程,即从键盘输入一个十进制整数,将其转换为二进制输出。

【问题分析】

在 1.1.1 节中已介绍,十进制整数转换为二进制数的方法是:**除以基数倒取余数**,直到商为 0 为止。将得到的余数倒序排列(即最后得到的余数是最高位),即得到转换后的结果,如图 5.6 所示。

本题的难点是如何将每次得到的余数倒序存储构造成一个整数,即将图 5.6 中的余数按公式 $1\times10^0+0\times10^1+1\times10^2+1\times10^3+1\times10^4=11101$ 计算后,用 int 型变量存储转换结果 11101。

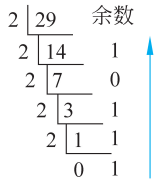


图 5.6 十进制整数转二进制整数计算过程

规划数据结构如下:

(1) 定义 int 型变量 n,用于存储输入的十进制整数。

(2) 定义 int 型变量 result,初值为 0,用于存储转换后的二

进制整数。

(3) 定义 int 型变量 w,表示位权,初值为 1,每循环一次,w 乘以 10,用以表示上述计算公式中的 10^0、10^1、10^2 等。

算法的 N-S 流程图如图 5.7 所示。

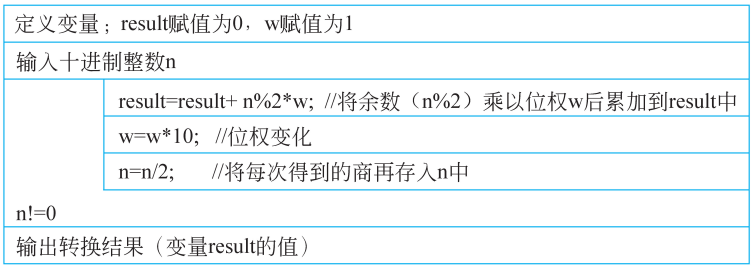
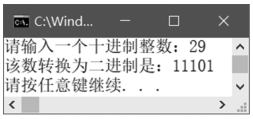


图 5.7 例 5.4 算法的 N-S 流程图

【编程实现】

```
#include <stdio.h>
int main()
{
    int n,w=1,result=0;
    printf("请输入一个十进制整数: ");
    scanf("%d",&n);
    do
    {
        result=result+n%2*w;
        w=w*10;
        n=n/2;
    }while(n!=0);
    printf("该数转换为二进制是: %d\n",result);
    return 0;
}
```

【运行结果】



【关键知识点】

在计算机中,图 5.6 中的转换结果 11101 实际上是按十进制数存储的,只是用户认为这是一个二进制数。由于 int 型变量的最大取值为 $2^{31}-1=2147483647$,因此,变量 result 能存储的最大数是 1111111111(计算机认为该数是十进制,用户认为该数是二进制),将二进制的 1111111111 转换为十进制是 1023。因此,本题的程序运行时,输入的十进制整数最大只能为 1023,否则 result 的值会溢出,得到错误的计算结果。

【延展学习】

(1) 将十进制整数转换成八进制数并输出。

(2) 将二进制整数转换成十进制数并输出。二进制转十进制的方法详见 1.1.1 节。

(3) 将八进制整数转换成十进制数并输出。


for 语句课堂练习

5.3 for 语句

5.3.1 for 语句

for 语句用于实现**当型循环**，流程图如图 5.8 所示，语法格式为：

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

for 语句的执行过程：首先求解表达式 1，然后判断表达式 2(即循环条件)是否为真，若为真则执行循环体，然后求解表达式 3；之后重复“判断表达式 2→执行循环体→求解表达式 3”的过程，当表达式 2 的值为假(即循环条件不成立)时，结束循环，执行 for 语句的后续语句。

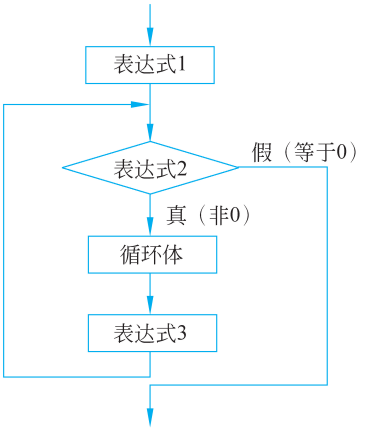


图 5.8 for 语句流程图

说明：

(1) 循环体可以是单条语句，也可以是加花括号后构成的复合语句。

(2) for 语句的圆括号中有且仅有 2 个分号，用于将 3 个表达式分开。表达式 1 用于为循环控制变量赋初值；表达式 2 是循环控制条件；表达式 3 常用于修正循环控制变量的值。

例如，以下代码可用于求 1＋2＋3＋…＋n 的和。

```
int n,i,sum=0;
scanf("%d",&n);
for(i=1;i<=n;i++)
    sum=sum+i;
```

(3) 当表达式 1 或表达式 3 包含多个表达式时，应使用**逗号表达式**。例如，若 n 为偶数，以下代码可求 1＋2＋3＋…＋n 的和；若 n 为奇数，则求和结果错误(正中间的数会被加两次)。

```
int n,i,j,sum;
scanf("%d",&n);
for(sum=0,i=1,j=n;i<=j;i++,j--)
    sum=sum+i+j;
```

(4) 表达式 1、表达式 2 和表达式 3 都可以省略，但各表达式间的**分号不能省略**。

如果省略表达式 1，表示无需赋初值(为循环控制变量赋初值可在 for 语句之前完

成);如果省略表达式 2,表示循环条件永远为真(在循环体中应有能跳出循环的流程控制语句,否则会产生死循环);如果省略表达式 3,表示没有修正部分(这时应在循环体内修正循环控制变量的值,以确保循环能正常结束)。

如果同时省略表达式 1 和表达式 3,则 for 语句等效于 while 语句,即 for(;条件;)等同于 while(条件)。

如果同时省略 3 个表达式,则循环条件永真,即 for(;;) 等同于 while(1)。

5.3.2 for 语句的应用

【例 5.5】 计算阶乘之和,即 1!+2!+3!+…+n!,n 从键盘输入。

【问题分析】

本题可以通过递推法完成。**递推法**是一种简单的算法,即通过已知条件,利用特定关系得出中间推论,直至得到最终结果的算法。递推法分为顺推和逆推两种。**顺推法**是从已知条件出发,利用特定关系逐步推算出问题的解;**逆推法**是从问题的结果出发,利用特定关系逐步推算出问题的开始条件,即顺推法的逆过程。

分析本题中各累加项的关系可知,后一项等于前一项的值乘以项数。

规划数据结构如下:

(1) 定义 int 型变量 sum,用于存储累加和(被加数),初值为 0。

(2) 定义 int 型变量 term,用于存储累加项(加数),初值为 1,即第 1 项的值。

(3) 定义 int 型变量 i,表示第几项,初值为 1。

算法的 N-S 流程图如图 5.9 所示。

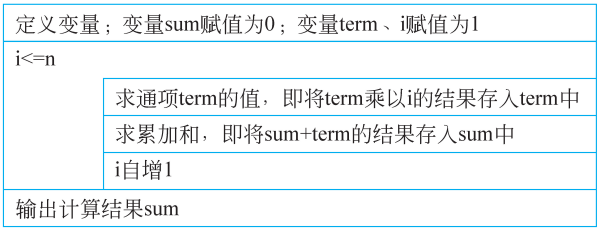


图 5.9 例 5.5 算法的 N-S 流程图

【编程实现】

```
#include <stdio.h>
int main()
{
    int sum=0,term=1,i,n;
    printf("请输入一个正整数: ");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        term=term*i;
        sum=sum+term;
```

```
11        }
12        printf("sum=%d\n",sum);
13        return 0;
14    }
```

【运行结果】

【关键知识点】

(1) 若代码第 4 行未给变量赋初值，则可将第 7 行修改为 for(sum=0, term=1, i=1; i<=n; i++)，即在 for 语句的表达式 1 中使用逗号表达式为多个变量赋初值。

(2) 第 9 行实现了顺推法，即第 1 次循环时计算 1×1 得到第 1 个累加项的值，存入 term 中。第 2 次循环时计算 1×2 得到第 2 个累加项的值，存入 term 中。以此类推，每次循环都用前一项的值乘以 i 得到当前项的值。

【延展学习】

(1) 除了用递推法之外，本题能否用其他方法编程实现？

(2) 当 n 超过 12 后，运行结果不正确，其原因是 13!=6227020800，已超出 int 型变量的数据表示范围，所以 sum 和 term 的值会溢出。请思考变量 sum 和 term 应定义为什么类型？第 12 行的输出语句应如何修改？

【例 5.6】 编程输出所有的水仙花数。水仙花数是一个三位数，其各位数字的立方之和等于该数本身，例如，$153=1^3+5^3+3^3$，因此 153 是水仙花数。

【问题分析】

枚举法是利用计算机运算速度快、精确度高的特点，对要解决问题的所有可能情况，一个不漏地进行检验，从中找出符合要求的答案，因此枚举法是通过牺牲时间来换取答案的全面性。

本题可采用枚举法，即定义 int 型变量 i，用 for 循环控制 i 的值从 100 变化到 999，对于每一个 i，取出它的个位、十位、百位数字，并用 if 语句判断它是否满足水仙花数的条件。

【编程实现】

```
1    #include <stdio.h>
2    int main()
3    {
4        int i,a,b,c;
5        printf("水仙花数有: ");
6        for(i=100;i<=999;i++)
7        {
8            a=i/100;                    //百位数字
9            b=i/10%10;                  //十位数字
10           c=i%10;                     //个位数字
11           if(i==a*a*a+b*b*b+c*c*c)
```