模块五　循环结构程序设计

在许多问题中需要用到循环控制。例如，要求全班学生某单科成绩总分、平均分；迭代求根；累加累乘等问题；循环结构在程序设计中，应用非常广泛。

所谓循环结构就是指在一定条件下重复执行一组语句的程序结构。C 语言有四种实现循环结构的语句：

- while 语句
- do-while 语句
- for 语句
- goto 语句

5.1　while 循环语句

- 掌握 while 语句的使用。

案例 5-1

用 while 语句求解 1～100 所有整数的累计求和。

✧　**案例程序**

```
main()
{
    int i , sum = 0;            /*定义循环控制变量 i 和初始化累加器 sum*/
    i = 1;                      /*给循环控制变量 i  赋初值*/
    while( i <= 100 )
    {
        sum = sum + i;          /*实现累加*/
        i++;                    /*循环控制变量 i 加 1*/
    }
    printf("sum = %d\n", sum);
}
```

程序运行结果如下：

```
sum = 5050
```

✧　**案例分析**

（1）本案例是求 1～100 的累加和，因其加数是递增序列，即第 1 次做的是 0+1=1，第二次做的是 1+2=3，以此类推，当前结果作为下一次的被加数，因此，我们只需定义两个变量，一个是存放累加结果的变量 sum，另一个是存放加数的变量 i。

（2）本案例 i 除了用来表示加数，还用作循环执行条件控制。用来控制循环执行条件的变量就叫循环控制变量。循环执行条件是 i <= 100，当条件成立时，继续执行循环体，即 sum = sum + i 和 i++；否则，跳到循环体外，即 while 语句的下一条 printf("sum = %d\n", sum);语句。

（3）循环体中的 sum = sum + i 语句可用 sum += i 代替。

（4）while 语句的特点是先判断再执行，如果循环条件一开始就为假，则循环体一次也不执行。

✧ **随堂练习**

将案例 5-1 改为求 1～100 所有奇数或偶数之和。

✧ **知识链接**

（1）while 语句一般格式：

```
while( 循环执行条件 )
    { 循环体语句组；}
```

（2）执行过程：

1）求解“循环执行条件”表达式。如果其值非 0，表示条件为真，转 2)；否则转 3)。

2）执行循环体语句组，然后转 1)，再次进行循环条件判断。

3）执行 while 语句的下一条。

执行过程如图 5-1 所示。

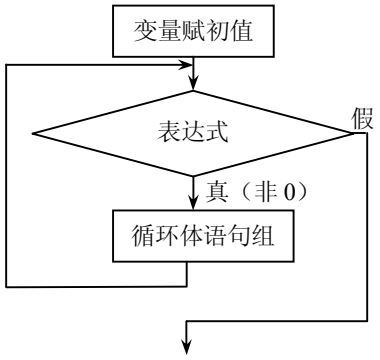

图 5-1

5.2 do-while 循环语句

- 掌握 do-while 语句的使用。
- 掌握 while 语句和 do-while 语句的区别。

案例 5-2

用 do-while 语句求解 1～100 所有整数的累计求和。

✧　**案例程序**

```
main()
{
    int i = 1, sum = 0;              /*定义并初始化循环控制变量，以及累加器*/
    do
    {
        sum+=i;                      /*实现累加*/
        i++;
    } while (i <= 100);              /*循环执行条件：i<=100*/
    printf("sum = %d\n",sum);
}
```

程序运行结果如下：

```
sum = 5050
```

✧　**案例分析**

（1）do-while 语句的特点是：先执行循环体语句组，再判断循环条件。假设循环条件一开始就为假，也至少执行一次循环体，这是 do-while 和 while 语句的主要区别。

（2）do-while 循环中的 while (i <= 100);表达式要以分号结束，这是语法规定。

（3）当循环执行条件 i <= 100 不成立时，跳出循环体，继续执行后面的语句，即 printf("sum=%d\n",sum);语句。

✧　**随堂练习**

参考本案例，用 do-while 语句编写程序求 300～500 之间所有整数之和。

✧　**知识链接**

（1）do-while 语句一般格式：

```
do
        { 循环体语句组;}
while( 循环执行条件 );
```

（2）执行过程：

1）执行“循环体语句组”。

2）计算“循环执行条件”表达式。如果“循环执行条件”表达式的值为非 0（真），则转向 1)、继续执行；否则，转向 3)。

（3）执行 do-while 的下一条语句。

执行过程如图 5-2 所示。

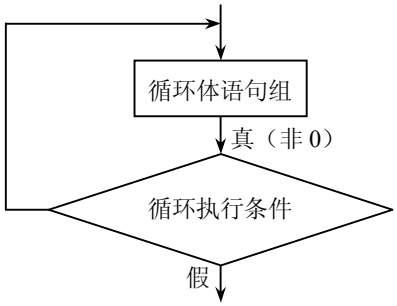

图 5-2

5.3 for 语句

学习目标

- 掌握 for 语句的几种形式。
- 掌握 for 语句的使用。

案例 5-3

用 for 语句求解 1～100 所有整数的累计求和。

✧ **案例程序**

```
main()
{
    int i, sum = 0;              /*将累加器 sum 初始化为 0*/
    for( i = 1; i <= 100; i++ )
    {
        sum += i;                /*实现累加*/
    }
    printf("sum = %d\n", sum);
}
```

程序运行结果如下：

```
sum = 5050
```

✧ **案例分析**

（1）for 语句中有三个表达式，两两之间用分号隔开。首先执行第一个表达式（变量赋初值表达式）i = 1。

（2）其次判断第二个表达式（循环执行条件表达式）i <= 100，表达式成立，因此执行循环体语句组，即 sum += i。

（3）再次，执行第三个表达式（循环变量增值）i++，再回到循环执行条件表达式进行下一次判断，成立则再次执行循环体语句组。

（4）在“循环执行条件表达式”、“循环体语句组”和“循环变量增值”三处形成一个环，直到变量 i 的值自增到 101 时，此时 i <= 100 不成立，跳出循环，即跳到 for 语句的下一条语句 printf("sum = %d\n", sum);继续执行，直到整个程序结束。

✧ **知识链接**

（1）for 语句的一般格式：

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

写成更易理解的形式：

```
for ([变量赋初值];[循环执行条件];[循环变量增值])
        { 循环体语句组;}
```

（2）执行过程：

1）求解“变量赋初值”表达式。

2）求解“循环执行条件”表达式。如果其值非 0，执行 3）；否则，转至 4）。

3）执行循环体语句组，并求解“循环变量增值”表达式，然后转向 2）。

4）执行 for 语句的下一条语句。

执行过程如图 5-3 所示。

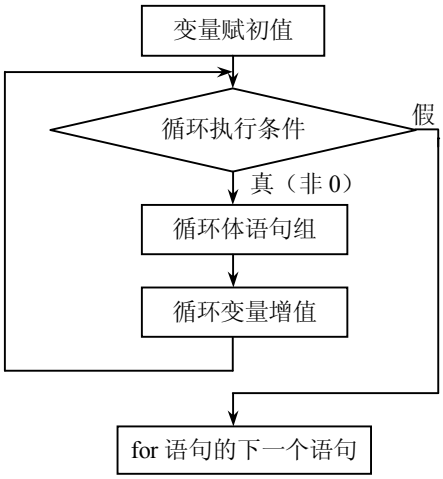

图 5-3

案例 5-4

for 语句的其他形式。

✧　**案例程序**

```
main()
{
    int i = 1, sum = 0;          /*将累加器 sum 初始化为 0*/
    for(   ; i <= 100;   )
    {
        sum += i;                /*实现累加*/
        i++;
    }
    printf("sum = %d\n", sum);
}
```

程序运行结果如下：

```
    sum = 5050
```

✧　**案例分析**

（1）“变量赋初值”、“循环执行条件”和“循环变量增值”部分均可缺省，甚至全部缺省，但其间的分号不能省略。当省略“循环执行条件”时，即为永真式，为避免死循环，通常会在循环里进行条件判断，当满足某一条件时跳出循环。

（2）“变量赋初值”表达式，既可以是给循环变量赋初值的赋值表达式，也可以是与此无关的其他表达式（如逗号表达式）。

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

（3）“循环执行条件”部分是一个逻辑量，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。

（4）“循环变量增值”可以是任何修改循环变量值的表达式，如 i += 2、i = x + y、i--等。

✧ **知识链接**

C 语言中实现循环结构的语句除了以上所学的 while 语句、do-while 语句和 for 语句以外，还有 goto 语句。因为使用 goto 语句，会破坏程序的结构化，降低程序的可读性，因此我们不提倡使用 goto 语句。在此，我们就 goto 语句的使用只作简单介绍。

案例 5-5

使用 goto 语句实现求解 1～100 累计和。

✧ **案例程序**

```
main()
{
    int n=1, sum=0;
    loop:                    /*定义标号 loop*/
    sum += n;
    n++;
    if (n<=100)
    goto loop;               /*转向标号 loop 去执行*/
    printf("sum=%d\n", sum);
}
```

✧ **案例分析**

（1）loop:为语句标号，其命名遵循标识符命名规则。goto 语句格式：goto 标号，功能为：使系统转向标号所在的语句行执行。

（2）goto 语句通常和 if 语句构成循环。

5.4 循环语句的嵌套

- 掌握循环嵌套的几种形式。
- 掌握二重循环的执行流程。

一个循环体内又包含另一个完整的循环结构，称为循环嵌套。按嵌套的层次可分为二重嵌套、三重嵌套或多重循环。

三种循环语句 while、do-while 和 for 可以互相嵌套，构成 9 种基本形式。

案例 5-6

编写程序，在屏幕上显示如下图案：

* * * * *

✧　**案例程序**

```
main()
{
    int i;              /* 定义变量 i，i 用来控制循环的执行 */
    for ( i = 1; i <= 5; i++ )
    printf("*");        /* 循环体语句 */
}
```

✧　**案例分析**

（1）在没有学习循环语句之前，我们会用一个 printf()语句输出“* * * * *”，但学习循环语句后，我们可以用循环语句实现。

（2）当循环体语句组仅由一条语句构成时，可以不使用复合语句形式，即循环体可省略大括号。

案例 5-7

编写程序，在屏幕上显示如下图案：

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

✧　**案例程序**

```
main()
{
    int i,j;
    for (j=1 ;j <= 3; j++)
    {                           /* 行（4）左“{”是外循环的开始 */
    for (i=1; i <= 5; i++)
      printf("*   ");           /* 行（6）内循环语句 */
    printf("\n");               /* 行（7）外循环语句*/
    }                           /* 行（8）右“}”是外循环的结束 */
}
```

✧　**案例分析**

（1）本例中在 j 循环（外循环）中嵌套了 i 循环（内循环），即构成二重循环。程序的执行流程是：先执行外循环的“变量赋初值”表达式 j = 1，接着判断“循环执行条件”表达式 j <= 3，成立则进入内循环，否则转至行（4）。

（2）外循环条件成立，进入内循环，内循环的执行过程和案例 5-3 相似。本案例中内循环执行了 5 次。即行（6）内循环语句 printf("*   ")执行了 5 次，输出“* * * * *”，然后继续执行外循环的其余语句，本例是行（7）printf("\n")，即输出一个换行符。

（3）执行外循环变量自增表达式 j++，然后再进行“循环执行条件”j<=3 表达式的判断，成立，则转向行（2），否则转至行（4）。

（4）跳出外循环。

案例 5-8

编写程序，在屏幕上显示如下图案：

```
*
* *
```

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

✧ **案例程序**

```
main()
{
    for ( i = 1; i <= 5; i++ )
    {for ( j = 1; j <= i ;j++ )
        { printf("*   ");}
      printf("\n");
}
```

✧ **案例分析**

（1）本案例和案例 5-7 的区别是内循环的“循环执行条件”表达式不再是一个常量（如 5），而是变量 i（j <= i），其中 i 的值每执行完一次外循环都会发生变化。即每一趟外循环对应的内循环执行的次数是由外循环变量 i 决定的。

（2）当每趟外循环对应的内循环执行次数相同时，我们可用常量来控制循环执行次数；当每趟外循环对应的内循环执行次数都发生变化时，用常量就无法满足要求。进行多重循环程序设计时，首先要理解其执行流程，按照题目要求，正确设计出内、外循环的条件控制。

案例 5-9

输出所有“水仙花数”，所谓“水仙花”数是指一个三位数，其各位数字立方和等于该数本身（如 153 是一个水仙花数，因为 $153 = 1^3+5^3+3^3$）。

✧ **案例程序**

```
main()
{
    int num,bai,shi,ge;
    for( num = 100; num <= 999; num++ )     /*判断 100～999 之间的所有三位数*/
    {
        bai = num / 100;                    /*求 num 的百位数*/
        shi = num % 100 / 10;               /*求 num 的十位数*/
        ge = num % 10;                      /*求 num 的个位数*/
        if( num == bai * bai * bai + shi * shi * shi + ge * ge * ge)
            printf(" %5d",num);       /*如果 num 符合水仙花数条件,则输出*/
     }
}
```

程序运行结果：

```
    153   370   371   407
```

✧ **案例分析**

（1）用循环变量 num 来表示所有三位数，其初值是 100，结束条件是 num <= 999（等价于 num < 1000）。判断完某个 num 后，num 自增 1，然后继续判断。

（2）循环体中包括了一个条件判断语句，即如果某个 num 等于它的各位数字立方和，该 num 是水仙花数，要输出。否则，不是水仙花数，不输出。

案例 5-10

求 Fibonacci 数列：1 , 1 , 2 , 3 , 5 , 8 , 13 ……的前 20 项，即

$F_1 = 1$　　（n = 1）

$F_2 = 1$　　（n = 2）

$F_n = F_{n-1} + F_{n-2}$　　（n≥3）

✧　**案例程序**

```
#include <stdio.h>
main( )
{
    int i , f1 , f2 , f3;
    f1 = 1; f2 = 1;                    /*为数列前两项赋初值*/
    clrscr();                          /*清屏函数*/
    printf("%10d%10d", f1 , f2);       /*输出前两项*/
    for ( i = 3 ; i <= 20 ; i++ )      /*求第 3 项到第 20 项*/
   {
        f3 = f1 + f2;                  /*从第 3 项起，每一项等于前两项之和*/
        f1 = f2;
        f2 = f3;
        printf(" %8d ", f3);
        if ( i%5 == 0)    putchar( '\n' );   /*每行输出 5 个数*/
    }
}
```

程序运行结果如下：

```
         1        1        2        3        5
         8       13       21       34       55
        89      144      233      377      610
       987     1597     2584     4181     6765
```

✧　**案例分析**

（1）for 循环求的是第 3 项至第 20 项，f3 = f1 + f2 表示当前项等于前两项之和，下次求下一项，下一项等于当前项和当前项的前一项之和，故要修改变量 f1 和 f2 的值，即将 f2 的值赋给 f1，将 f3 的值赋给 f2。

（2）在循环体中的判断语句 if (i%5 == 0) putchar('\n');表示当 i 能被 5 整除时，输出一个换行符。这里的换行是用语句 putchar('\n')实现的，也可以使用 printf("\n");语句。

（3）因为用到 putchar 函数，所以要用#include <stdio.h>预包含命令。

5.5　break 语句和 continue 语句

- 掌握 break 语句的使用。
- 掌握 continue 语句的使用。

为了使循环控制更加灵活，C 语言提供了 break 语句和 continue 语句。

5.5.1 break 语句

案例 5-11

计算半径 r=1 到 r=10 的圆面积，当面积大于 100 时结束。

✧ **案例程序**

```
#include <stdio.h>
main( )
{
    int r ;
    float area , pi = 3.14159;
    clrscr();
    for (r = 1 ; r <= 10 ; r ++ )
    {
        area = pi * r * r ;
        if(area >100)       /*当面积 area>100 时，跳出循环*/
          break;
        printf ("%10.2f", area );
      }
    getchar();
}
```

程序运行结果如下：

```
3.14        12.57        28.27        50.27        78.54
```

✧ **案例分析**

（1）从程序运行结果可知，循环执行次数没有 10 次，这是因为在循环体中加了判断语句 if(area >100) break; 当计算到 r=6 时，面积大于 100，执行了 break 语句跳出循环体。

（2）本案例在循环体后加了一个 getchar();语句，它的作用是等待用户输入，程序才继续执行、结束。目的是为了方便查看结果。同学们可比较有此语句和无此语句的效果。

✧ **随堂练习**

本案例若想查看最后一次没有输出的面积值，可以怎么修改程序？

✧ **知识链接**

（1）break 语句一般格式：break。

（2）功能：强行结束循环，转向执行循环语句的下一条语句。

（3）说明：

1）break 能用于循环语句和 switch 语句中。

2）循环嵌套时，break 只影响包含它们的最内层循环，与外层循环无关。

5.5.2 continue 语句

案例 5-12

输出 100～200 之间所有能被 8 整除的数。

✧ **案例程序**

```
main( )
{
```

```
    int n ;
    for (n = 100 ; n <= 200 ; n++)
    {
      if (n % 8 != 0)          /*如果 n 不能被 8 整除*/
            continue;          /*直接跳到 n++处执行*/
       printf ("%5d", n);
    }
 }
```

程序运行结果如下：

104　112　120　128　136　144　152　160　168　176　184　192　200

✧　**案例分析**

（1）本案例要求输出 100～200 之间所有能被 8 整除的数，即不能被 8 整除的数不用输出，所以可用 continue 语句跳过本次的输出 printf ("%5d", n);语句。

（2）本案例不使用 continue，仅用 if 语句也可实现要求：if (n%8 = = 0) printf ("%5d", n);，这里仅仅是举例说明 continue 的用法。

✧　**随堂练习**

若要输出 100～200 之间所有不能被 8 整除的数，程序该怎么修改？

✧　**知识链接**

（1）continue 语句一般格式：continue;。

（2）功能：在循环体中，当执行到 continue 语句时，跳过循环体内位于其后的其他语句，接着进行下一次是否执行循环的判定；即“提前结束本次循环”。

（3）说明：

1）continue 语句只能用于循环语句中。

2）循环嵌套时，continue 只影响包含它的最内层循环，与外层循环无关。

3）continue 与 break 不同：continue 语句只结束本次循环，而不是终止整个循环的执行；而 break 语句则是结束整个循环，不再判断循环条件是否成立。

课后练习

✧　**选择题**

1．以下程序的输出结果是（　　）。

```
# include <stdio.h>
main( )
{
int num = 0, i=1;
while ( i > = 2 )
        {
          num = num +i;
          i++;
        }
 printf("%d\n", num );
 }
```

A）1　　B）2　　C）3　　D）0

2．以下程序的输出结果是（　　）。

```
main( )
{
    int num = 0;
    while (num < = 2)
    {
        num = num + 1;
        printf("%d\n", num );
    }
}
```

A）1　　B）1
2　　C）1
2
3　　D）1
2
3
4

3．以下程序段执行（　　）。

```
x =  －1 ;
while (! x)
    {x = x * x;}
```

A）1 次　　B）2 次　　C）死循环　　D）0 次

4．以下程序段执行（　　）。

```
x = 0;
while (! x)
    {x = x * x;}
```

A）1 次　　B）2 次　　C）死循环　　D）0 次

5．以下程序段执行（　　）。

```
x = -1 ;
do
    {x = x * x;}
while (! x);
```

A）1 次　　B）2 次　　C）死循环　　D）0 次

✧ **程序分析**

1．下面程序的运行结果是：_____。

```
# include <stdio.h>
main( )
{
    int i , sum=0 ;
    for (i = 0 ; i <= 5 ; i++)
        {
            sum = sum + 1;
        }
     printf("sum = %d\n", sum );
}
```

2．以下程序的输出结果是：_____。

```
main( )
{
    int i , sum=0 ;
    for (i = 0 ; i <= 5 ; i++)
        sum = sum + i;
        printf("%d\n", sum);
}
```

3．程序的运行结果是：_____。

```
# include <stdio.h>
main( )
{
    int i, j,n, sum = 100;
    for(i = 5 ; i > 0 ; i--)
    {
        n = 0;
        for(j = 2 ; j>=0 ; j--)
        n = n + i +j;
        sum = sum – n;
    }
    printf ("%d" , sum );
}
```

4．以下程序的功能是从键盘上输入一行字符以回车结束。分别求数字、字母和其他字符个数。在横线处填上正确的语句。

```
# include <stdio.h>
main()
{
    char ch ;
    int digit =0, letter=0 , others=0;
    while (_____)
    {
        if（ch >= '0' && ch <= '9')
        digit= digit +1;
        else if ((ch >='A' && ch <='Z') && (ch >='a' && ch <='z'))
        letter = letter +1;
        else others = others +1 ;
        printf ("digit =%d, letter =%d, others=%d \n" ,digit, letter, others);
    }
}
```

上机实训

✧　**实训目的**

1．能用 while、do-while 和 for 语句编写循环结构程序。

2．能在循环结构程序中正确使用 break 和 continue 语句。

✧ **实训内容**

1．编写程序，求 n!，并输出最终结果。

2．求 $\sum_{k=1}^{200} k + \sum_{k=1}^{100} k^2 + \sum_{k=1}^{50} \frac{1}{k}$

3．有一分数序列：

$$\frac{2}{1},\frac{3}{2},\frac{5}{3},\frac{8}{5},\frac{13}{8},\frac{21}{13},\cdots\cdots$$

求出这个数列的前 20 项之和。

4．设计程序，在屏幕上输出一张九九表。

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

5．打印以下图案。

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

6．利用 continue 语句，编程完成 200 到 300 之间不能被 3 整除的数的输出。