学习项目三　基于循环结构实现学生成绩统计

学习情境：

一个班有 40 名学生，平均分成 5 个小组，参加了 C 程序设计期中考试，C 程序设计老师想统计每个小组的总分和平均分。

学习目标：

同学们通过本项目的学习，将了解三种循环语句的特点与区别，并学会使用三种循环语句编写程序；培养学生分析问题、解决问题的能力。由浅入深地安排教学内容，将学生被动接受知识变为主动探索知识。

学习框架：

任务一：统计单个小组 C 程序设计期中考试的总分及平均分
任务二：统计每个小组 C 程序设计期中考试的总分及平均分

3.1　任务一　统计单个小组 C 程序设计期中考试的总分及平均分

知识目标	（1）循环结构的三种语句语法及其执行的过程 （2）while 语句的使用 （3）for 语句的使用 （4）各种循环语句的区别
能力目标	（1）利用各种循环语句正确编写程序 （2）能够将 while 语句与 for 语句互相转换 （3）运用循环结构独立解决实际问题
素质目标	（1）培养学生自主学习的能力 （2）培养学生分析问题的能力 （3）培养学生解决问题的能力
教学重点	循环结构的应用及其执行的流程
教学难点	循环结构的正确使用
效果展示	 图 3-1　任务一运行效果图

3.1.1 任务描述

一个班有 40 名学生，平均分成 5 个小组，参加了 C 程序设计期中考试，老师计划设计一个简单的程序用来统计小组的总分和平均分；该任务的要求如下：

（1）新建 3-1.c 文件；

（2）在主函数中实现 8 名同学成绩的录入、计算，并输出总分及平均分。

3.1.2 任务实现

```
/*********************************************************************
* 任务一：统计单个小组 C 程序设计期中考试的总分及平均分
*********************************************************************/
#include <stdio.h>
main()
{
    int i=1,sum=0,score;
    float avg;
    printf("\n  计算小组 C 程序设计成绩的总分和平均分\n");
    printf("-------------------------------------------\n");
    printf("  请输入小组的 8 名学生的 C 程序设计成绩：\n");
    while (i<=8)
    {
        scanf("%d",&score);
        sum=sum+score;          //成绩累加
        i++;
    }
    avg=sum/8.0f;               //计算平均成绩
    printf(" 小组总成绩为：%d  平均成绩为：%.2f\n",sum,avg);
}
```

程序运行效果如图 3-1 所示。

本任务可以定义 8 个变量来接收 8 名学生的 C 程序设计成绩，完成平均分及总分的计算，但是显然这样做是不科学的。计算总分的过程是输入一个学生的成绩，累加到总分中，再输入第二个学生的成绩，再累加到总分中，直到最后一名学生的成绩输入并累加到总分中。成绩录入和成绩累加是重复相同的操作，使用循环结构更简单、合理。

本任务中需要学习的内容是：

- 学习循环结构的概念及执行过程
- 学习使用 while、for 循环结构完成程序设计
- 了解各种循环结构的区别

3.1.3 相关知识

一、循环结构的概述

循环结构是程序设计中一种很重要的结构。在需要重复执行一组步骤的情况下，应使用循环控制。例如，要输入全校学生成绩、求若干个数之和等。其特点是，在给定的条件成立时，

重复执行某程序段，直到条件不成立为止。给定的条件称为循环条件，重复执行的程序段称为循环体。

循环结构有两种类型：当型循环和直到型循环，其执行流程如图 3-2 所示。

（1）当型循环：先判断给定条件，若给定条件成立则执行循环体 A；然后重复“当判断条件成立时——执行 A”的过程；当条件不成立时退出循环结构。

（2）直到型循环：先执行循环体 A，再判断条件，如果条件不成立，继续执行循环体 A；然后重复“当判断条件不成立时——执行 A”的过程，当条件成立时，退出循环结构。

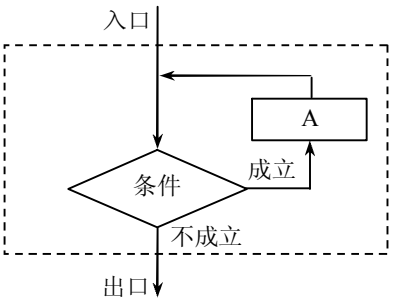


图 3-2　当型循环执行流程图

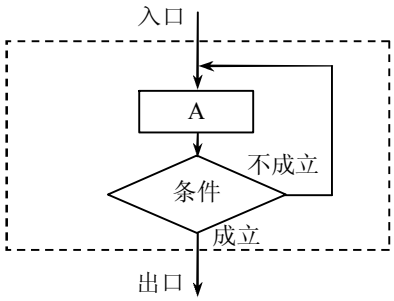


图 3-3　直到型循环执行流程图

C 语言提供了多种循环语句，常见的循环结构语句如下：

- while 语句
- do～while 语句
- for 语句

二、while 语句

while 语句用来实现“当型”循环结构。

1. while 语句一般形式

```
while(表达式)
    语句;          //循环体
```

2. while 语句执行过程

（1）判断 while 语句循环条件表达式的值，若为真，则转（2），否则转（3）；

（2）执行循环体语句组，然后转（1）；

（3）退出 while 循环。

while 循环结构执行的流程如图 3-4 所示。

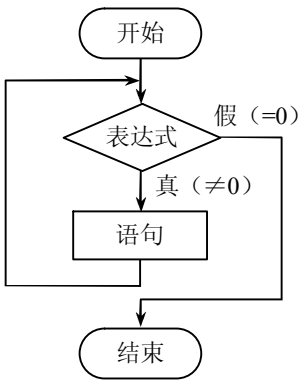


图 3-4　while 语句执行流程图

需要注意的是：

（1）while 语句先判断表达式的真假，然后决定是否执行循环体。

（2）while 语句中的表达式一般是关系表达式或逻辑表达式，只要表达式的值为真（非 0），即可继续循环。

【例 3-1】输出小于 n 的正的偶数。

问题分析：首先定义一个计数变量 n 接收输入的正整数，循环变量从最小的正整数 2 开始，若循环变量为偶数，则输

出之，而后循环变量自加；否则直接循环变量自加。当循环变量大于等于 n 时退出循环。

```
#include <stdio.h>
main()
{
    int i=1,sum=0,n;
    printf("\n  输出所有小于 n 的正的偶数 \n");
    printf("-----------------------------------\n");
    printf("  请输入 n 的值：");
    scanf("%d",&n);
    printf("  所有小于%d 的正的偶数为：\n",n);
    i=2;
    while (i<=n)
    {
        if(i%2==0)            //判断 i 是否为偶数
          printf("  %d  ",i);
        i++;
    }
    printf("\n");
}
```

程序运行效果如图 3-5 所示。

图 3-5　例 3-1 运行效果图

本例程序将执行 n 次循环，每执行一次，循环变量 i 值加 1。

（3）为了避免陷入“死循环（无休止的循环）”，while 语句的循环体中应含有使循环趋向于结束的语句（如 i++）。

例如：

```
int i=1,score=0,sum=0;
while (i<=8)
{
    scanf("%d",&score);
    sum=sum+score;
    i++;
}
```

如果循环体内没有“i=i+1;”或者“i++;”语句，i 值永远为初始值，则循环条件永远成立，循环将一直进行下去，造成死循环。

（4）如果循环体包括一个以上的语句，则必须用{}括起来，组成复合语句。如果不加花括号，则 while 语句循环体只包含 while 语句后的第一条语句。

例如：

```
i=1;
while(i<=100)
```

```
printf("%d",i);
i++;
```

从形式上看，while 循环本意要控制“printf("%d",i);”和“i++;”两条语句，其实只控制了第一条语句，因为循环体中没有修改循环变量 i 的值，循环条件恒为真，从而造成死循环。

【例 3-2】统计从键盘输入一行字符的个数。

问题分析：输入一行字符，以输入回车符表示输入结束。故引入一个计数变量 n，统计输入字符的个数。直到输入的字符为回车符，停止计数退出循环。

算法设计流程图如图 3-6 所示。

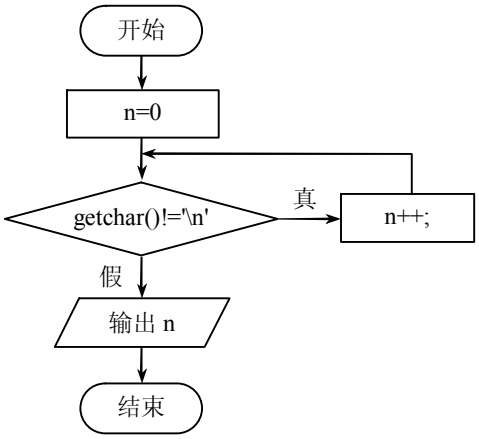


图 3-6　例 3-2 流程图

```
#include <stdio.h>
main()
{
    int n=0;                              //定义记录字符个数的变量
    printf("\n  统计从键盘输入一行字符的个数");
    printf("\n----------------------------------\n");
    printf("  请输入一个字符串：");
    while(getchar()!='\n')                //当从键盘输入的字符不是回车符
        n++;                              //字符个数加 1
    printf("  总字符数为%d\n",n);         //输出字符个数 n
}
```

程序运行效果如图 3-7 所示。

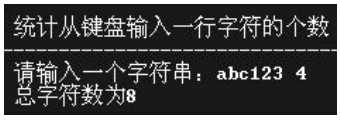


图 3-7　例 3-2 运行效果图

三、do ~ while 语句

do～while 语句用来实现“直到型”循环结构。

1. do ~ while 语句一般形式

```
do
    语句;  //循环体
while(表达式);
```

其中语句是循环体，表达式是循环条件。

2. do ~ while 语句执行过程

（1）执行循环体语句组；

（2）判断 while 语句循环条件表达式的值，若为真，则转（1），否则转（3）；

（3）退出 while 循环。

do～while 语句执行流程如图 3-8 所示。

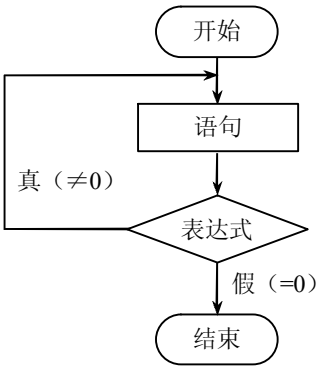


图 3-8　do～while 语句执行流程图

do～while 语句无论条件如何，首先执行一次循环体语句组，而后判断条件的真假，因此无论循环判断条件是否为真，do～while 语句中循环体语句组至少要执行一次。而 while 语句是先判断后执行，如果条件不满足，则循环体语句组一次也不执行。

while 语句和 do～while 语句一般都可以相互改写。如例 3-2 可以用 do～while 语句实现，见例 3-3。

【例 3-3】统计从键盘输入一行字符的个数。

```
#include <stdio.h>
main()
{
    int n=-1;                          //定义记录字符个数的变量
    printf("\n  统计从键盘输入一行字符的个数");
    printf("\n----------------------------------\n");
    printf("  请输入一个字符串：");
    do {
        n++;                           //字符个数加 1
    }while(getchar()!='\n') ;          //当从键盘输入的字符不是回车
    printf("  总字符数为%d\n ",n);     //输出字符个数 n
}
```

程序执行效果如图 3-7 所示。在本例中，比例 3-2 多执行一次循环体，所以计数变量 n 初值为-1。

在一般情况下，用 while 语句和用 do～while 语句处理同一问题时，若二者的循环体部分是一样的，当 while 循环体后面的表达式一开始就为真时，两种循环的结果是相同的。否则二者的结果是不同的。

需要注意的是：

（1）while 语句表达式后面不能加分号，而 do～while 语句的表达式后面则必须加分号。

（2）若 do 和 while 之间的循环体由多个语句组成时，必须用{}括起来组成一个复合语句。

（3）do～while 和 while 语句相互替换时，要注意修改循环控制条件。

将任务一中实现计算小组学生 C 程序设计考试的总分和平均分的 while 语句改写为 do～while 语句，两种语句实现的代码段如下：

```
while(i<=8)
    {   scanf("%d",&score);
        sum=sum+score;
        i++;
    }
```

```
do
{   scanf("%d",&score);
    sum=sum+score;
    i++;
} while(i<=8);
```

四、for 语句

for 语句是 C 语言所提供的功能最强、使用范围最广泛的一种循环语句，不仅可以用于循环次数已经确定的情况，而且可以用于循环次数不确定而给出循环结束条件的情况。

1. for 语句一般形式

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

（1）表达式 1：通常用来给循环变量赋初值，一般是赋值表达式。也允许在 for 语句外给循环变量赋初值，此时可以省略该表达式；

（2）表达式 2：判断循环终止的条件；

（3）表达式 3：改变循环变量的值，一般是赋值语句。

2. for 语句执行过程

（1）计算表达式 1 的值；

（2）计算表达式 2 的值，若值为真（非 0），则转（3）；否则（4）；

（3）执行循环体语句组及计算表达式 3 的值，转回第（2）步重复执行；

（4）跳出循环。

在整个 for 循环执行过程中，表达式 1 只计算一次，表达式 2 和表达式 3 则可能计算多次。循环体可能执行 0 次或多次。for 循环结构执行流程图如图 3-9 所示。

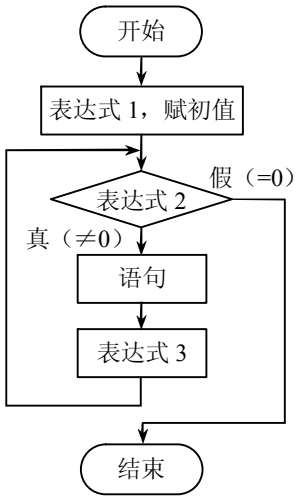


图 3-9　for 语句执行流程图

需要注意的是：

（1）三个表达式都可以是逗号表达式，即每个表达式都可由多个表达式组成，例如：

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

（2）三个表达式都是任选项，都可以省略，但表达式间的分号不能省略；

- 省略表达式 1，表示不需赋初值，初值可在进入 for 语句前完成
- 省略表达式 2，表示循环条件恒为真
- 省略表达式 3，表示没有循环条件修正部分，此时为确保循环正常结束，在循环体内部应该有循环条件的修正语句

（3）循环体可以是空语句；

（4）for 语句可与 while 语句相互转换。

【例 3-4】统计从键盘输入一行字符的个数。

```
#include <stdio.h>
main()
{
    int n=0;                              //定义记录字符个数的变量
    printf("\n  统计从键盘输入一行字符的个数");
    printf("\n-----------------------------------\n");
    printf("  请输入一个字符串：");
    for(;getchar()!='\n';n++);            //当从键盘输入的字符不是回车符时，
                                          //字符个数加 1，此循环结构中无循环体
    printf("  总字符数为：%d\n ",n);      //输出字符个数 n
}
```

程序执行效果如图 3-7 所示。

程序采用 for 循环实现了从键盘输入一行字符的个数统计。例中省去了 for 语句的表达式 1，而表达式 3 也不是用来修改循环变量，而是用作输入字符的计数。这样，就把本应在循环体中完成的计数操作放在表达式 3 中完成了，因此循环体是空语句。应注意的是，空语句后的分号不可少，若缺少分号，则把后面的 printf 语句当成循环体来执行。

将任务一中实现计算小组学生 C 程序设计考试的总分和平均分的 while 语句改写为 for 语句，两种语句实现的代码段如下：

```
int i=1;
while(i<=8)
{
    scanf("%d",&score);
    sum=sum+score;
    i++;
}
```

```
int i=1;
for(;i<=8;)
{
    scanf("%d",&score);
    sum+=score;
    i++;
}
```

```
int i;
for(i=1;i<=8;i++)
{
    scanf("%d",&score);
    sum+=score;
}
```

for 语句中的表达式 1“i=1”可以省略，因为在变量 i 定义时已经赋初值为 1。表达式 3“i++”也可以省略，而将其移到循环体内部。其具体实现如右侧代码段所示。

五、break 语句

以上三种循环结构都是在循环体之前或之后通过一个表达式的值来决定是否中止对循环体的执行，也可以在循环体中利用 break 语句终止循环的执行，跳转到循环语句后的下一条语句处继续执行。

（1）break 语句的形式。

```
break;
```

（2）break 语句的作用。

终止 switch 语句或循环语句的执行，跳转到其后的语句处继续执行。

```
while(表达式 1)
{    ......
     if(表达式 2)
        break;
     ......
}
```

break 语句执行流程如图 3-10 所示。

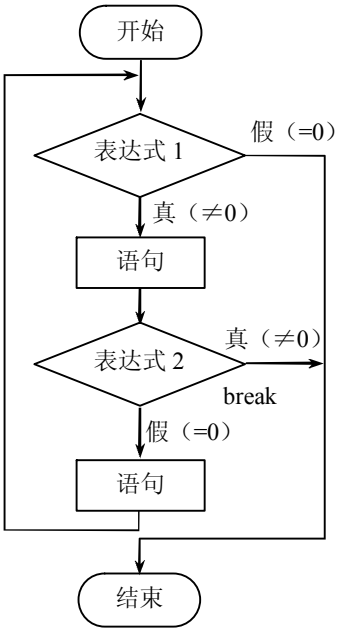


图 3-10　break 语句执行流程图

说明：

（1）在循环结构中，break 语句常常同 if 语句同时使用，表示当指定条件满足时，立即跳出循环结构。但 break 语句只能跳出循环语句，而不能跳出 if 语句。

（2）break 语句只能跳出当前一层循环结构。

【例 3-5】输入两个正整数并计算其最大公约数。

问题分析：（1）定义 3 个变量，x、y 和 z；

（2）x、y 分别接收输入的两个正整数，最大公约数 z 等于 x 和 y 中较小的；

（3）计算 x%z 与 y%z 的值，若二者中有值不等于 0，则令 z 自减，重复（3）；若都等于 0，则转（4）。

（4）x、y 的最大公约数即 z 的值，跳出循环；

```
#include <stdio.h>
main()
{
    int x,y,z;                          //x,y 为两个正整数；  z 为 x,y 的最大公约数
    printf("\n 计算输入的两个正整数的最大公约数");
    printf("\n-------------------------------------\n");
    printf(" 请输入 2 个正整数：");
    scanf("%d, %d",&x,&y);
    if (x>y)                            //令变量 z 取 x 与 y 中较小的值
        z=y;
    else
        z=x;
    while(z>1)
    {                                   //z 值逐一递减直至找到某一个值可同时被 x 与 y 整除
    if(x%z==0 && y%z==0)
        break;                          //z 不是 x 与 y 的公约数时，跳过本次循环
    else
        z--;
    }
    printf(" 最大公约数为：%d\n",z);  //输出最大公约数的值
}
```

程序运行效果如图 3-11 所示。

图 3-11　例 3-5 运行效果图

六、continue 语句

continue 语句只用在 for、while、do-while 等循环体中，常与 if 条件语句一起使用，用来加速循环。应注意的是，continue 只结束本层次的循环，并不跳出循环。

（1）continue 语句的形式。

```
continue;
```

（2）continue 语句的作用。

结束本次循环，强行转入下一次循环条件的判断与执行，不再执行本次循环中 continue 语句之后的语句。

其执行流程如图 3-12 所示。

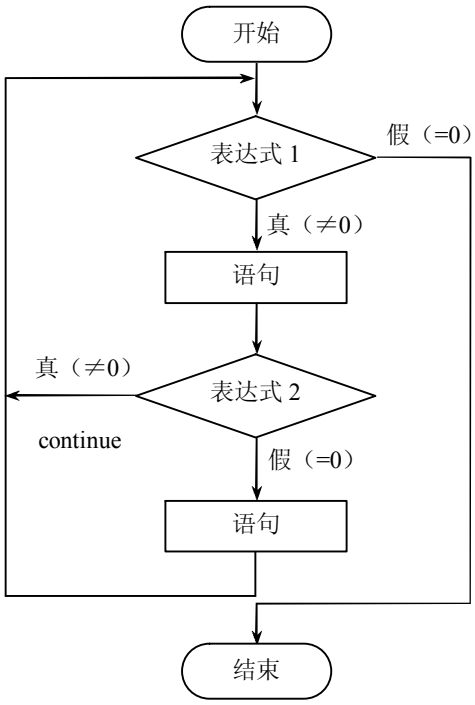


图 3-12　continue 语句执行流程图

【例 3-6】输出从键盘输入的除 Esc 键的字符，输入回车则结束程序。

问题分析：首先明确应使用循环结构，重复输入和输出两个操作。其次确定使用 while 语句实现，定义临时字符变量 c，并确定循环执行条件(c=getch())!=13。最后写出正确的程序代码。

```
#include <stdio.h>
#include <conio.h>
main()
{
    char c;
    while((c=getch())!=13)        //输入回车则循环结束
    {
        if(c==27)                 //Esc 的 ASCII 码值为 27
          continue;               //若按 Esc 键结束本次循环，进行下次循环
        printf("%c\n", c);
    }
}
```

【例 3-7】输出 100 以内能被 7 整除的数。

问题分析：首先确定在解决该问题时重复操作是判断某个数是否能够被 7 整除，且判断的次数确定，故选择用 for 语句实现。其次，确定循环变量、初值及循环结束的条件。最后，选择用 continue 语句结束本次循环来控制非 7 的倍数不打印输出。

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

```
    printf("\n                    输出 100 以内能被 7 整除的数\n");

    printf("------------------------------------------------------------\n");
    for(n=7;n<=100;n++)      //1 到 6 肯定不能被 7 整除，所以 n 从 7 到 100 循环
    {
        if(n%7!=0)           //如果 n 不能被 7 整除，退出本次循环，不打印
            continue;
        printf("%d\t",n);    //否则表示 n 能被 7 整除，打印输出
    }
    printf("\n");
}
```

程序运行效果如图 3-13 所示。

```
                    输出100以内能被7整除的数
------------------------------------------------------------------------
7       14      21      28      35      42      49      56      63      70
77      84      91      98
```

图 3-13　例 3-7 运行效果图

几种循环语句的区别如下：

（1）三种循环都可以用来处理同一问题，一般情况下它们可以互相代替；

（2）while 和 do～while 循环，只在 while 后面指定循环条件，在循环体中应包含使循环趋于结束的语句（如 i++或 i=i+1 等）；

（3）for 循环可以在表达式 3 中包含使循环趋于结束的操作，甚至可以将循环体中的操作全部放到表达式 3 中。因此 for 语句的功能更强，凡用 while 循环能完成的，用 for 循环都能实现；

（4）用 while 和 do～while 循环时，循环变量初始化的操作应在 while 和 do～while 语句之前完成。而 for 语句可以在表达式 1 中实现循环变量的初始化；

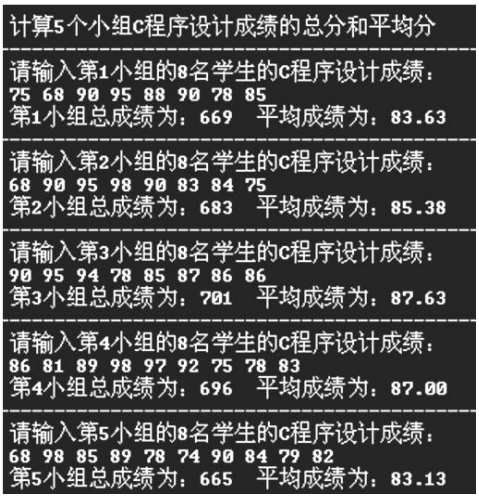
（5）三种循环都可以用 break 语句跳出循环，用 continue 语句结束本次循环。

3.1.4　任务小结

本任务使用循环结构实现了统计小组 C 程序设计成绩的总分及平均分，通过该任务的学习，同学们能够掌握各种循环语句的特点与区别，能够独立使用三种循环控制语句解决实际问题。

3.2　任务二　统计每个小组 C 程序设计期中考试的总分及平均分

知识目标	（1）嵌套循环结构的语法 （2）嵌套循环结构的执行过程
能力目标	（1）掌握嵌套循环的使用 （2）使用嵌套循环结构解决实际问题
素质目标	（1）培养学生自主学习的能力 （2）培养学生独立分析问题的能力 （3）培养学生独立解决问题的能力

教学重点	利用嵌套循环解决实际问题
教学难点	利用嵌套循环解决实际问题
效果展示	 图 3-14　任务 3-2 运行效果图

3.2.1　任务描述

一个班有 40 名学生，平均分成 5 个组，参加了 C 程序设计期中考试，C 程序设计老师想统计每个小组的总分和平均分。该任务的要求如下：

（1）新建 3-2.c 文件；

（2）在主函数中实现 5 个小组学生的成绩录入；

（3）分别统计每个小组的总分和平均分，并输出计算结果。

3.2.2　任务实现

```
/*****************************************************************
* 任务二：统计每个小组期中考试 C 程序设计的总分及平均分
*****************************************************************/
#include <stdio.h>
main()
{
    int i,j=1,sum,score;
    double avg;
    printf("\n 计算 5 个小组 C 程序设计成绩的总分和平均分\n");
    printf("-------------------------------------------\n");
    while (j<=5)
    {
        printf(" 请输入第%d 小组的 8 名学生的 C 程序设计成绩：\n",j);
        i=1;                        //每个小组为 8 人，从 1 开始计数
        sum=0;                      //每个小组总分初值为 0
        while (i<=8)                //统计每个小组的平均分和总分
```

```
        {
            scanf("%d",&score);
            sum=sum+score;
            i++;
        }
        avg=sum/8.0;                    //统计每组成绩的平均分
        printf(" 第%d 小组总成绩为：%d  平均成绩为：%.2f\n",j,sum,avg);
        j++;
        if(j<=5)                        //控制在小组成绩之间输出横线
            printf("-----------------------------------------\n");
    }
}
```

程序运行效果如图 3-14 所示。

5 个小组的成绩计算方法相同，只是重复计算，故使用双层循环实现该功能。其中外层循环执行 5 次，变量 j 的值分别为 1、2、3、4、5。而 j 每取一个值，内层循环中变量 i 从 1 到 8 将执行 8 次内层循环体。故内层循环体共执行 40（5*8）次。

本任务中需要学习的内容是：

- 嵌套循环结构语法
- 嵌套循环执行过程

3.2.3 相关知识

在一个循环体内包含另一个完整的循环体，称为嵌套循环。C 语言中可以实现多层嵌套。其中内层循环的优先级高于外层循环，即先执行内层循环再执行外层循环。C 语言对循环嵌套深度没有限制，但嵌套的内、外层循环控制变量不得同名。并且使用嵌套循环时，一个循环必须完全包含在另一个循环内。

【例 3-8】将任务二功能改为由 while 语句嵌套 for 语句实现。

```
#include <stdio.h>
main()
{
    int i,j=1,sum,score;
    double avg;
    printf(" 计算 5 个小组 C 程序设计成绩的总分和平均分\n");
    printf("-----------------------------------------\n");
    while (j<=5)                    //变量 j 记录小组信息
    {
        printf(" 请输入第%d 小组的 8 名学生的 C 程序设计成绩：\n",j);
        for(i=1,sum=0;i<=8;i++)  //每个小组为 8 人，从 1 开始计数;每个小组总分初值为 0
        {
            scanf("%d",&score);
            sum=sum+score;
        }
        avg=sum/8.0;                //统计每组成绩的平均分
        printf(" 第%d 小组总成绩为：%d  平均成绩为：%.2f\n",j,sum,avg);
        j++;                        //一个小组信息统计完成
```

```
            if(j<=5)
                printf("-----------------------------------------\n");
        }
    }
```

程序运行效果如图 3-14 所示。

三种循环可以互相嵌套。循环结构的选择应根据实际情况灵活使用，做到结构清晰、简单明了。

【例 3-9】输出如图 3-15 所示下三角九九乘法表。

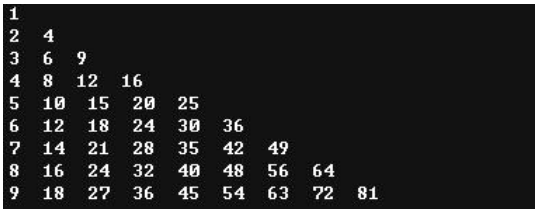


图 3-15　例 3-9 运行效果图

问题分析：首先确定共打印 9 行，每一行比上一行多一个数；其次确定每个数字是该数字所在的行号和列号的乘积；最后确定行号为 1 到 9，每一行中列号为从 1 到当前行号的值。

```
#include <stdio.h>
main()
{
    int i=1,j=1;
    for(i=1;i<10;i++)                  //控制输出的行
    {
        for(j=1;j<=i;j++)              //控制输出的列
            printf(" %d ",i*j);        //输出数值
        printf("\n");                  //输出一行后换行
    }
}
```

【例 3-10】编程输出如图 3-16 所示的星型三角图形。

图 3-16　例 3-10 运行效果图

问题分析：该星型三角形共由 5 行“*”组成，每一行重复一个动作——输出“*”。故首先确定使用嵌套循环实现；其次找规律，第一行 9 个，以后每行减少 2 个，且后一行输出的第一个“*”比前一行退后 1 个位置。故内层循环应该有 2 个，一个循环控制输出“*”的位置即打印控制，另一个循环控制输出“*”。

```
#include <stdio.h>
main()
{
    int i,j,k;
    for(i=5;i>0;i--)                   //共输出 5 行，i 取 5，4，3，2，1
    {
        for(k=0;k<5-i;k++)
            printf(" ");               //输出空格
        for (j=1;j<2*i;j++)
```

```
            printf("*");          //输出当前行的*
        printf("\n");             //输出一行后换行
    }
}
```

【例 3-11】求 1+2+3+……+n 前 n 个数的和。

问题分析：这是一个累加的问题，相邻两个数相差 1，重复的操作是把后一个数加到前 m 项的和中。

```
#include <stdio.h>
main()
{
    int i=1,n;
    long int sum=0;
    printf("\n  统计 1+2+3+……+n 的和\n");
    printf("-------------------------- \n");
    printf("   请输入 n 的值：");
    scanf("%d",&n);
    printf("   计算结果为：");
    for(;i<=n;i++)
        sum=sum+i;                    //累加求和
    printf("  %ld  ",sum);            //输出自然数的和
    printf("\n");
}
```

程序运行效果如图 3-17 所示。

图 3-17　例 3-11 运行效果图

说明：该程序是求和运算，变量 sum 的初始值必须为 0，否则结果将多 1。

【例 3-12】计算 n!。

问题分析：参考例 3-11，只是把累加改为阶乘，同时注意变量的初始值问题。

```
#include <stdio.h>
main()
{
    int i=1,n;
    long int sum=1;
    printf("\n  ----------统计 n!--------\n");
    printf("  请输入 n 的值：");
    scanf("%d",&n);
    printf("  %d!计算结果为：",n);
    for(;i<=n;i++)
        sum=sum*i;                    //计算乘积
    printf("  %ld  ",sum);
    printf("\n");
}
```

程序运行效果如图 3-18 所示。

图 3-18　例 3-12 运行效果图

该程序实现时与例 3-11 最大的区别是变量 sum 的初始值不同，若 sum 的初始值为 0，则结果为 0。

【例 3-13】求 1+2!+3!+...+20!的和。

问题分析：首先求出 1 到 20 的每个数的阶乘，其次汇总求和，可利用嵌套循环实现。内层循环求每个数的阶乘。外层循环将每个数的阶乘累加求和，循环变量值从 1 到 20。

```
#include <stdio.h>
main()
{
    int i=1,n=1;
    long int sumall=0,sum=1;
    printf("\n 求 1+2!+3!+...+20!的和 \n");
    printf("------------------------\n");
    while (n<=20)
    {
        for(i=1;i<=n;i++)
            sum=sum*i;      //计算 n!
        sumall+=sum;        //计算 1+2!+3!+...+20!的和
        n++;
    }
    printf(" 计算结果：%ld \n ",sumall);
}
```

程序运行效果如图 3-19 所示。

```
求1+2!+3!+...+20!的和
------------------------
计算结果：1444231215
```

图 3-19　例 3-13 运行效果图

【例 3-14】编写程序，输出 1 到 20 之间的数，当其为 8 的倍数时不输出。

问题分析：输出小于 20 的非 8 的倍数，要从 1 到 20 逐个数判断其是否是 8 的倍数，若不是则输出其值，若是，则继续判断下一个数是否输出。

```
#include <stdio.h>
main()
{
    int i=1;
    printf("\n   输出 1 到 20 之间的非 8 的倍数\n");
    printf("------------------------------------------------------------
----------------\n");
    for(i=1;i<=20;i++)
    {
        if(i%8!=0)
            printf("%4d",i);
    }
    printf("\n");
}
```

程序运行效果如图 3-20 所示。

```
输出1到20之间的非8的倍数
------------------------------------------------------------
   1   2   3   4   5   6   7   9  10  11  12  13  14  15  17  18  19  20
```

图 3-20　例 3-14 运行效果图

【例 3-15】输入一个二进制数，将其转换为十进制数输出。

问题分析：二进制数转化为十进制数的基本做法是把二进制数首先写成加权系数展开式，然后按十进制加法规则求和。例如，（101011）转化为十进制的公式是 $1*2^5+0*2^4+1*2^3+0*2^2+1*2^1+1*2^0$，十进制的值为 32+8+2+1=43。

```
#include <stdio.h>
#include <math.h>
main()
{
    int temp,sum=0,i=1;
    printf("\n 二进制数转化为十进制数\n");
    printf("----------------------------------\n");
    printf(" 请输入一个二进制数：");
    scanf(" %d",&temp);
    while(temp>0)
    {
        sum=sum+temp%10*i;
        temp=temp/10;
        i*=2;
    }
    printf(" 对应的十进制数为：%d\n",sum);
}
```

程序运行效果如图 3-21 所示。

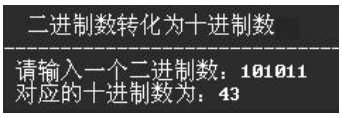


图 3-21　例 3-15 运行效果图

3.2.4　任务小结

本任务中，使用嵌套循环结构统计了每个小组 C 程序设计成绩的总分及平均分，并利用循环结构解决了 6 个不用类型的问题，加深了对循环结构的理解，使同学们能够熟练地利用循环结构解决实际问题。

习题三

一、填空题

1．常用的循环结构语句分别是_____、_____、_____。
2．循环体执行遇到 break 语句时，_____。
3．循环体执行遇到 continue 语句时，_____。

二、选择题

1．for(i=1;i<9;i+=1);该循环共执行了（　　）次。
A．7　　B．8　　C．9　　D．10
2．int a=2;while(a=0) a--;该循环共执行了（　　）次。
A．0　　B．1　　C．2　　D．3
3．执行完循环 for(i=1;i<100;i++);后，i 的值为（　　）。

A．99　　B．100　　C．101　　D．102

4．以下 for 语句中，书写错误的是（　　）。

A．for(i=1;i<5;i++);　　B．i=1;for(;i<5;i++);

C．for(i=1;i<5;) i++;　　D．for(i=1,i<5,i++);

5．（　　）语句，在循环条件初次判断为假，还会执行一次循环体。

A．for　　B．while　　C．do~while　　D．以上都不是

6．下列程序段执行后 s 的值为（　　）。

```
int i=1, s=0;  while(i++)  if(!(i%3)) break ;  else s+=i ;
```

A．2　　B．3　　C．6　　D．以上均不是

7．i、j 已定义为 int 类型，则以下程序段中内循环体的执行次数是（　　）。

```
for(i=5;i;i--)
for(j=0;j<4;j++){…}
```

A．20　　B．24　　C．25　　D．30

8．int a=1, x=1;　循环语句 while(a<10) x++; a++;循环执行（　　）。

A．无限次　　B．不确定次　　C．10 次　　D．9 次

三、读程序，写结果

1．下面程序的输出结果是_____。

```
#include <stdio.h>
void main( )
{
    int y=9;
    for( ;y>0; y--)
    if(y%3==0)
    {
        printf("%d", --y);
        continue;
    }
}
```

2．下面程序的输出结果是_____。

```
#include <stdio.h>
main()
{
    int k,n,m;
    n=10;m=1;k=1;
    while (k++<=n)
        m*=2;
    printf("%d\n",m);
}
```

3．下面程序的输出结果是_____。

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

```
    int i=5;
    do
    {
        switch (i%2)
        {
            case  4: i--; break;
            case  6: i--; continue;
         }
         i-- ;  i-- ;
         printf("i=%d  ", i);
    } while(i>0);
}
```

4．下面程序的输出结果是_____。

```
#include <stdio.h>
void main()
{
    int k=0; char c='A';
    do
    {
       switch (c++)
       {
          case 'A': k++; break;
          case 'B': k--;
          case 'C': k+=2; break;
          case 'D': k=k%2; break;
          case 'E': k=k*10; break;
          default: k=k/3;
       }
       k++;
    }
    while(c<'G');
    printf("k=%d\n", k);
}
```

5．下面程序输入数据 2，4 后，输出结果是_____。

```
#include <stdio.h>
main( )
{
    int  i=1,s=1,t=1,a,n;
    scanf("%d%d",&a,&n);
    for(;i<n;i++)
    {
       t=t*10+1;
       s=s+t;
    }
    s*=a;
    printf("SUM=%d\n",s);
}
```

四、程序设计题

1．求 10！的值。

2．求 1/2–2/3+3/4–4/5–5/6+…+79/80 的值。

3．输入一个正整数，计算输出各个位上所有数字之和。

4．若一个三位整数的各位数字的立方之和等于这个整数，称之为“水仙花数”。例如：153 是水仙花数，因为 $153=1^3+5^3+3^3$，求 1000 以内所有的水仙花数。