第5章

结构化程序设计二——设计学生成绩统计器二


视频讲解

本章的项目所实现的功能同第 4 章的项目相类似，都是实现一个用于学生成绩统计的程序。但是，在本项目中，系统功能通过循环结构和 switch 分支结构来实现学生成绩统计的功能。通过本项目的学习和实践，可以掌握通过循环结构，解决一些必须通过重复工作来完成的问题。

5.1 项目案例功能介绍

每门课程考试完成之后，任课教师要对考试成绩进行汇总、分析，成绩的汇总是分析的基础。假设全班人数为已知的，所需要统计的信息主要涉及全班的最高分、最低分、0～59 区间分数的人数及其所占的比例、60～69 区间分数的人数及其所占的比例、70～79 区间分数的人数及其所占的比例、80～89 区间分数的人数及其所占的比例、90～100 区间分数的人数及其所占的比例。

本项目采用控制台程序，实现依次接收每个同学的成绩，最后输出上述汇总信息。

5.2 项目设计思路

在本项目中，项目设计思路包括以下步骤。

(1) 项目分析与算法流程设计。

(2) 程序代码设计。

(3) 系统运行与效果测试。

5.3 关键技术

5.3.1 循环结构

程序设计中的循环结构，是指在程序中从某处开始有规律地反复执行某一语句块的结构，并

把重复执行的语句块称为循环体。

循环结构是一种常见的基本结构。循环结构按其循环体是否有嵌套从属的子循环结构，可分为单循环结构和多重循环结构。

循环结构可以在很大程度上简化程序设计，并解决采用其他结构而不易解决的问题。例如，计算 1＋2＋3＋…＋100 的和，采用顺序结构来解决非常不便。若采用循环结构来编写这种程序，就相当简单，只需要几条语句即可。C＃中提供了 4 种类型的循环结构，分别适用于不同的情形。

5.3.2 while 语句与 do…while 语句

while 语句和 do…while 语句用于循环次数不固定时，相当高效、简洁。

1. while 语句

while 语句的一般语法格式为：

```
while(条件表达式)
{
    循环体;
}
```

说明：while 语句在执行时，首先判定条件表达式的值。如果 while 后面括号中的条件表达式的值为 true，即执行循环体，然后再回到 while 语句的开始处，再判定 while 后面括号中的表达式的值是否为 true，只要表达式一直为 true，那么就重复执行循环体，直到 while 后面括号中的条件表达式的值为 false 时，才退出循环，并执行下一条语句。

while 语句的程序流程图如图 5-1 所示。

2. do…while 语句

do…while 语句与 while 语句功能相似，但和 while 语句不同的是，do…while 语句的判定条件在后面，这和 while 语句不同。do…while 循环不论条件表达式的值是什么，do…while 循环都至少要执行一次。do…while 语句的一般语法格式如下。

```
do
{
    循环体;
}
while(条件表达式);
```

说明：当循环执行到 do 语句后，先执行循环体语句；执行完循环体语句后，再对 while 语句括号中的条件表达式进行判定。若表达式的值为 true，则转向 do 语句继续执行循环体语句；若表达式的值为 false，则退出循环，执行程序的下一条语句。

do…while 语句的程序执行流程图如图 5-2 所示。

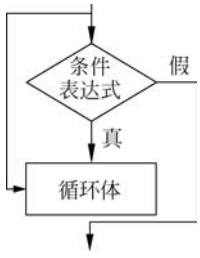

图 5-1　while 循环结构

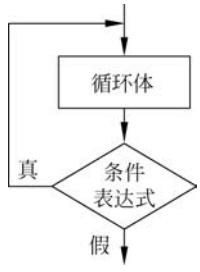

图 5-2　do…while 循环结构

5.3.3　for 循环语句和 foreach 语句

for 语句和 foreach 语句都是固定循环次数的循环语句。

1. for 语句

for 语句是一种功能强大的循环语句，它比 while 语句和 do…while 语句更为灵活。它的一般语法格式为：

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

for 语句的执行过程如下。

(1) 首先计算表达式 1 的值。

(2) 判定表达式 2 的值，若其值为 true，则执行循环体中的语句，然后求表达式 3 的值；若表达式 2 的值为 false，则转而执行步骤(4)。

(3) 返回步骤(2)。

(4) 结束循环，执行程序的下一条语句。

for 语句的程序执行流程图如图 5-3 所示。

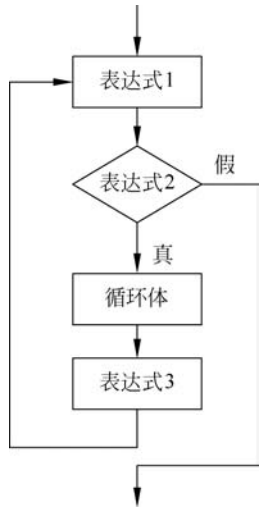

图 5-3　for 循环结构

2. foreach 语句

foreach 语句是 C＃中新增加的循环语句，它对于处理数组及集合等数据类型特别简便。foreach 语句用于列举集合中的每一个元素，并且通过执行循环体对每一个元素进行操作。foreach 语句只能对集合中的元素进行循环操作。foreach 语句的一般语法格式如下。

```
foreach(数据类型 标识符 in 表达式)
{
    循环体;
}
```

说明：foreach 语句中的循环变量是由数据类型和标识符声明的，循环变量在整个 foreach 语句范围内有效。在 foreach 语句执行过程中，循环变量就代表当前循环所执行的集合中的元素。每执行一次循环体，循环变量就依次将集合中的一个元素带入其中，直到把集合中的元素处理完毕，跳出 foreach 循环，转而执行程序的下一条语句。

5.4　项目实践

5.4.1　项目分析与算法流程设计

由前面所述的项目功能介绍可知，本项目包含以下几部分基本功能。

(1) 接收全班人数。

(2) 依次接收班级每个学生的成绩。

(3) 每接收一个成绩，依据要求进行统计汇总。

由此，可以得到问题的解决思路。该思路如下所述。

(1) 初始化统计信息各个变量。

(2) 接收班级学生人数 nStudents。

(3) 初始化已统计成绩的学生人数为 i=1。

(4) 判定 i<=nStudents，若判定式的值为 true，则执行(5)；若判定式的值为 false，则执行(9)

(5) 接收下一个成绩。

(6) 比较以前得到的最高、最低成绩与当前接收的成绩的大小关系，并修正统计信息。同时修正各区间成绩的统计信息。

(7) 已统计成绩的学生人数 i 自增。

(8) 执行(5)。

(9) 格式化输出统计信息。

程序的主流程图如图 5-4 所示。

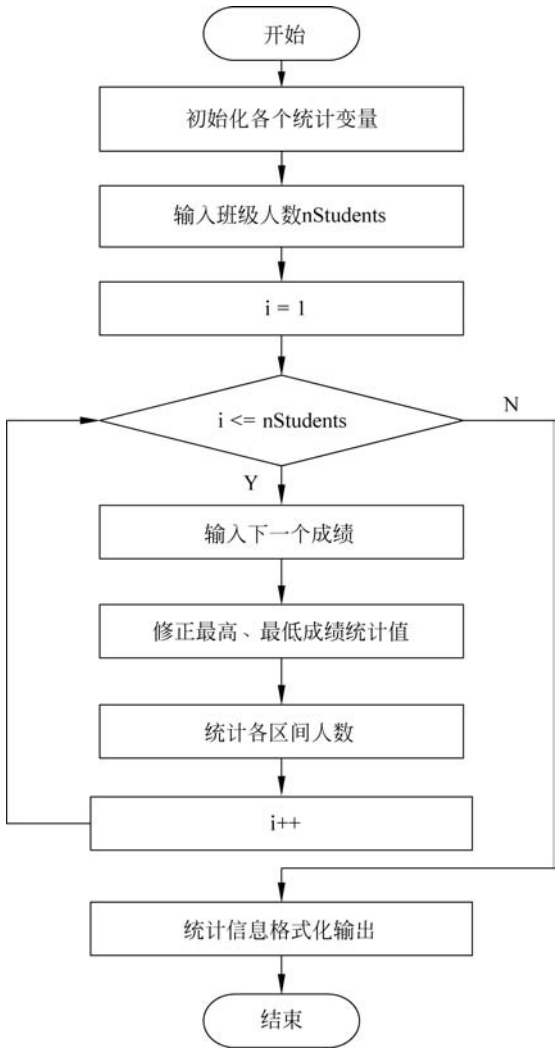

图 5-4　程序主流程图

其中，修正统计信息的执行策略如下。

（1）最高成绩修正：将接收得到的成绩 temp_score 与之前的统计值 max_score 比较，如果 temp_score＞max_score，则将新接收的成绩作为目前为止所有接收成绩中的最高值。

（2）最低成绩修正：将接收得到的成绩 temp_score 与之前的统计值 min_score 比较，如果 temp_score＜min_score，则将新接收的成绩作为目前为止所有接收成绩中的最低值。

（3）各区间人数修正：依次判定所接收的成绩在哪个区间范围内，然后将对应的区间统计值加 1。

5.4.2　程序代码设计

具体程序代码设计如下。

1. 代码设计

根据项目的描述，可在代码编辑器中完成如下代码的添加。

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Score
{
    class Program
    {
        static void Main(string[] args)
        {
            int nStudents;                                    //定义变量,用于统计学生的人数
            int temp_score;                                   //定义变量,接收每次输入的成绩
            //定义最大成绩与最小成绩
            int max_score = 0, min_score = 0;
            //统计成绩各区间内的成绩人数
            int score_0_59 = 0, score_60_69 = 0, score_70_79 = 0,
                score_80_89 = 0, score_90_100 = 0;
            Console.WriteLine("--------- 学生成绩统计 ----------- ");

            //接收全班人数
            Console.WriteLine("请输入班级人数: nStudents = ");
            string s = Console.ReadLine();
            nStudents = Int32.Parse(s);

            //接收并处理成绩
            for (int i = 1; i <= nStudents; i++)
            {
                //接收成绩
                Console.WriteLine("请输入第{0:d}个学生的成绩", i);
                temp_score = Int32.Parse(Console.ReadLine());

                //修改最高与最低成绩
```

```
        if (i == 1)
        {
            max_score = temp_score;
            min_score = temp_score;
        }
        else
        {
            if (max_score < temp_score)
                max_score = temp_score;
            if (min_score > temp_score)
                min_score = temp_score;
        }

        //接收成绩所在区间的人数
        int temp = temp_score / 10;
        switch (temp)
        {
            case 10:
            case 9:
                Console.WriteLine("优秀");       //成绩优秀
                score_90_100++;
                break;
            case 8:
                Console.WriteLine("良好");       //成绩良好
                score_80_89++;
                break;
            case 7:
                Console.WriteLine("中等");       //成绩中等
                score_70_79++;
                break;
            case 6:
                Console.WriteLine("及格");       //成绩及格
                score_70_79++;
                break;
            default:
                Console.WriteLine("不及格");     //成绩不及格
                score_0_59++;
                break;
        }
    }

    //统计信息输出
    Console.WriteLine("--------学生成绩统计信息输出----------");
    Console.WriteLine("全班共{0:d}人,其中最高成绩{1:f2},"+
                "最低成绩{2:f2}", nStudents, max_score, min_score);
    Console.WriteLine("成绩区间 90～100 的人数有{0:d}人," +
        "所占比例为:{1:f2}%", score_90_100, score_90_100 * 100 / nStudents);
    Console.WriteLine("成绩区间 80～89 的人数有{0:d}人," +
        "所占比例为:{1:f2}%", score_80_89, score_80_89 * 100 / nStudents);
    Console.WriteLine("成绩区间 70～79 的人数有{0:d}人," +
        "所占比例为:{1:f2}%", score_70_79, score_70_79 * 100 / nStudents);
```

```
                Console.WriteLine("成绩区间 60～69 的人数有{0:d}人," +
                    "所占比例为: {1:f2}%", score_60_69, score_60_69 * 100 / nStudents);
                Console.WriteLine("成绩区间 0～59 的人数有{0:d}人," +
                    "所占比例为: {1:f2}%", score_0_59, score_0_59 * 100 / nStudents);
                Console.ReadLine();
            }
        }
    }
```

2. 代码分析

(1) 本项目的基本设计思想是依次接收成绩,每接收一个学生成绩后,紧接着做相关的统计、汇总工作。

(2) 对于每次成绩统计、汇总工作而言,其基本操作流程是相同的,因此可以将成绩处理作为一个循环体来处理。

(3) 程序中使用了一个 for 循环结构,其循环次数控制是通过计数器 i 来实现的。

(4) 程序中使用了一个 switch 分支结构,使得程序结构更为清晰,增加了程序的可读性。

5.4.3　系统运行

系统运行之后,依据相关提示信息,输入学生的人数及其相应成绩,其运行结果如图 5-5 所示。

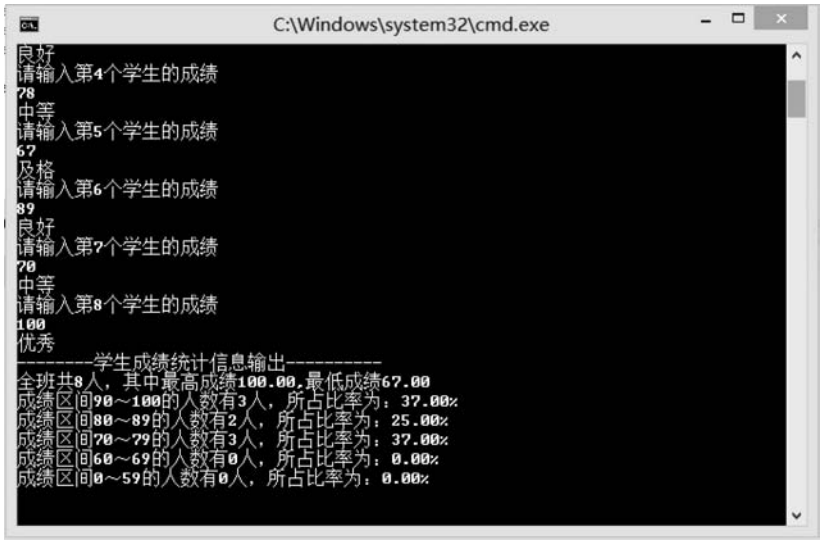

图 5-5　学生成绩统计的运行结果

5.5　小结

在 C# 中提供了 4 种循环结构语句：for 语句、while 语句、do…while 语句和 foreach 语句。

对于不同的实际运用,可以灵活选择不同的循环语句加以运用。对于 for 语句而言,一般用于循环次数明确的情况下；而 while 与 do…while 语句则常常用于循环次数未知的情况下；while 与

for 语句均为“预判定”或“预测试”语句，即在循环体执行前，首先进行是否满足循环的判定，然后才真正执行循环；而 do…while 循环则是“后循环”或“后测试”循环，即首先执行一次循环体，然后判定循环是否能够继续进行下去，如可以继续进行，则再次执行循环体；foreach 语句则是用于数组或集合元素中。

在所有的循环语句中，都应确保避免“死循环”的发生，这种情况的发生通常是由于循环条件的永久成立而引起的。一种有效控制循环进行的策略就是跳转语句的应用。C＃中提供了 goto 语句、continue 语句、break 语句和 return 语句来完成程序控制权的转移。

5.6 练一练

（1）分别用 for、while、do…while 语句编写程序，求出 100 之内的自然数之和。

（2）编写程序，要求输出如图 5-6 所示的图形。

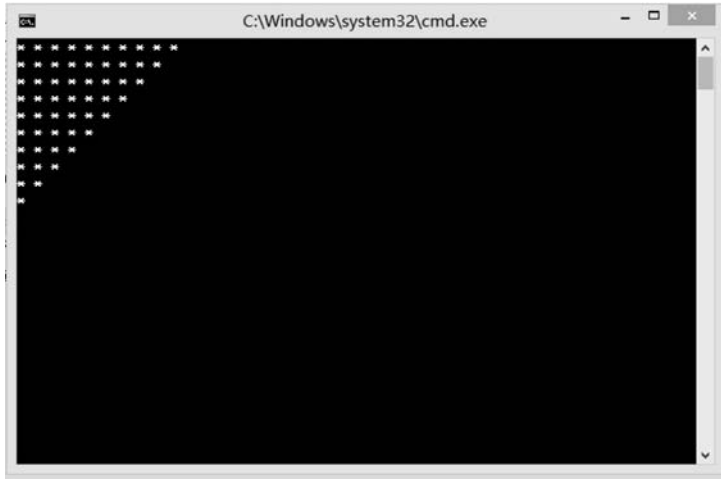

图 5-6　程序运行结果