3
结构体及数组

工作情境

学期考试结束后，需要对全系所有的学生成绩进行统计（包括最高分、最低分、及格率和需要补考的学生）。考虑到学生数量多、统计工作量大的情况，系部决定组织学生开发一套学生成绩管理系统，该系统要求实现对学生信息进行录入、修改、删除，根据学号进行学生信息的查询，以及对学生成绩进行统计等功能，现指派你完成该系统中成绩统计部分的功能。

解决方案

使用第 2 章的知识点已经能完成学生成绩管理系统的统计部分，即能统计学生某门科目的最高分、最低分、平均分和总分，并能计算科目的及格率，能根据成绩进行排名，统计需要补考的学生信息。但统计只限于某门科目，不能进行多门科目的统计和总成绩的统计，另外，学生的基本信息也不完整。为此，需要使用结构体对学生的各种信息进行组合，形成一个有机的学生信息整体；并将全系所有学生信息定义为一个学生数组，便于使用循环或分支结构来简化数据的统计。

能力目标

通过该项目的学习和开发，学生应该掌握结构体的定义、结构体变量的引用、结构体变量赋值、数组的定义、数组的赋值和数组的使用，对于有内在联系且不同类型的数据，能根据程序需求，灵活运用自定义数据类型结构体，将其组合成一个有机的整体；对于有内在联系且类型相同的批量数据，可运用数组将其当作一个有序整体，并在数组的基础上使用循环或分支结构来简化数据的处理。

任务 3.1　结构体

任务分析

根据学生成绩管理系统的需求，要求完成系部所有学生基本信息的录入，基本信息包括学生班级、学号、姓名、高数成绩、大学英语成绩、C++成绩等。

相关知识

1. 结构体类型的定义
2. 结构体变量的定义
3. 结构体变量的初始化
4. 结构体变量的引用

实现方法

前面的课程中，我们学习了一些基本数据类型（如 int、float、char）的定义和应用。对这些数据类型来说有个共同的特点，即在特定类型的变量中只能使用该类型来表示数据。但在日常生活中，我们常常会遇到一些需要填写的登记表，如成绩表、通讯地址表、书籍资料表等。在这些表中，填写的数据是不能用同一种数据类型描述的，如在成绩表中，我们通常会登记学号、姓名、科目等项目；在通讯地址表中，我们会写下姓名、邮编、邮箱地址、电话号码、E-mail 等项目。这些表中集合了各种数据，无法用前面学过的任何一种数据类型完全描述。这时需要将基本数据类型组合起来，形成一种新的数据类型。这种利用基本数据类型组合形成新数据类型的方法，称为**用户自定义类型**，也可称为复合类型。在 C++中，用户自定义类型包括结构体（structure）类型、共用体（union）类型、枚举（enumeration）类型和类（class）类型等。本任务中我们只学习结构体类型。

3.1.1 结构体类型的定义

C++中提供了一种称为结构体（structure）的类型，可以帮助将若干个不同类型的数据组合成一个整体，从而形成一种新的数据类型，有效地反映数据间的内在联系，利于对复杂对象进行描述和操作。如上例中的数据可以声明为一个名为 Student 的结构体类型来表示：

```
struct Student
{
    string   classId;              //班级
    string   studentId;            //学号
    string   name;                 //姓名
    float    scoreOfComp;          //计算机应用成绩
    float    scoreOfCplusplus;     //C++程序设计成绩
};                                 //分号不能少
```

由此可以看到，我们声明了一种新的类型——Student，它由 classId、studentId、name 等不同类型的数据项组成。要注意的是，这时创建的是一种新的类型，其名字叫 Student，它与基本类型 char、int、float 等一样，可以用来定义变量。

下面给出声明一个结构体类型的一般形式

```
struct 结构体类型名 {
    <成员列表>;
};
```

- **struct**：是一个系统保留字，在声明新的结构体类型时必须以此关键字开始，它向编

译系统声明，其后是一种新的结构体类型；

- **结构体类型名**：作为新声明的结构体类型的名字，其命名应符合 C++的变量命名规则，如上例中的 Student；
- **一对大括号**：作为定界符，用以确定新结构体类型所包括的成员，在大括号范围内的就是组成新数据类型的全部成员列表；
- **成员列表**：在大括号内申明的变量不再作为一个单独的变量，而是作为新类型的一部分，称为**成员**，也可称为**域**（Field），因此成员列表也可称为**域表**（Field List）。每个成员都必须进行类似下面的类型声明：

  <类型名> <成员名>;

  这里的类型可以是基本类型，如 char、int 等，也可以是用户定义类型，如结构体类型、枚举类型等。
- **分号**：结构体类型声明是一条声明语句，必须以分号作为结束标记。

结构体的声明语句一般放在源文件的开头，以便于在其后的函数和语句中可以用来定义变量，当然也可以在某个函数内部进行声明，不过这时，其作用域范围将限定在函数内部。

3.1.2 结构体变量的定义

前面我们学会了如何定义一个新的结构体类型，但声明的只是一种新的类型，还没有用来定义变量，系统也没有进行内存分配，这时无法进行数据操作。就相当于已创建一个新的模板，但还没有进行生产，我们无法拿到具体的产品来使用。为了能够在程序中使用结构体类型的数据，应当像使用基本类型一样定义结构体类型变量，并在其中存放数据。结构体类型变量的定义与其他类型的变量的定义一样，但由于结构体类型需要针对问题事先自行定义，所以结构体类型变量的定义形式就增加了灵活性，共有三种方式。

1. 先定义结构体类型，再定义结构体类型变量

如在已定义结构体类型 Student 的情况下，可以像使用基本类型一样定义结构体变量，如图 3.1.1 所示。

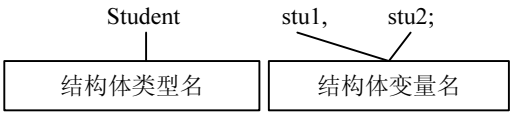

图 3.1.1 结构体定义

在 C++中，为保持与 C 语言的兼容，也允许在定义结构体变量时，在结构体类型前加上关键字 struct，例如：

```
struct Student stu1,stu2;
```

当用结构体类型定义变量后，系统将为变量分配相应的内存空间，这时结构体变量所占用的内存空间等于结构体类型中全部成员所占用内存空间之和，如 stu1、stu2 所占用的空间是 9+8+11+4+4=36。这可以使用 sizeof 运算符加以验证，例如：

```
cout<<sizeof(stu1)<<endl;          //sizeof 运算符中也可放数据类型，如 sizeof(Student)
```

这里要指明的是，相同类型的结构体变量所占用的内存空间大小是一样的，而不同类型

的结构体变量所占用的内存空间大小不一样。

下面给出这种方式的完整示例代码：

```
//定义结构体类型 Student
struct Student
{
    string  classId;              //班级
    string  studentId;            //学号
    string  name;                 //姓名
    float   scoreOfComp;          //计算机应用成绩
    float   scoreOfCplusplus;     //C++程序设计成绩
};
int main()
{
  …//其他语句
  Student stu1,stu2;              //定义结构体变量 stu1,stu2
  …//其他语句
}
```

2. 定义结构体类型的同时定义结构体类型变量

与方法 1 将结构体类型与结构体变量分别定义略有不同，方法 2 在定义结构体类型的同时定义结构体变量，示例如下。这两种方法只是为方便使用而加以区分，没有本质区别。

```
//定义结构体类型 Student
struct Student
{
    string  classId;              //班级
    string  studentId;            //学号
    string  name;                 //姓名
    float   scoreOfComp;          //计算机应用成绩
    float   scoreOfCplusplus;     //C++程序设计成绩
} stu1,stu2;                      //在这里定义结构变量
```

这种定义方法的一般形式如下：

struct 结构体类型名 {
<成员列表>;
} <变量名列表>;

3. 直接定义结构体类型变量

不定义结构体类型，直接定义结构变量，其一般形式如下：

struct {
<成员列表>;
} 变量名列表;

这种定义方法由于无法记录该结构体类型，所以除直接定义外，不能再定义该结构体类型变量，因此这种方法虽然合法，但较少使用。下面给出具体示例。

```
struct      //这里没有结构体类型名
{
```

```
    string   classId;              //班级
    string   studentId;            //学号
    string   name;                 //姓名
    float    scoreOfComp;          //计算机应用成绩
    float    scoreOfCplusplus;     //C＋＋程序设计成绩
} stu1,stu2;      //只能在这里定义结构变量
```

3.1.3 结构体变量的初始化

由于结构体类型变量汇集了各类不同数据类型的成员，所以结构体类型变量的初始化就略显复杂。具体来说，其初始化方式大致可以分为三种。

1. 定义时直接初始化

与其他类型的变量一样，可以在定义结构体变量时就指定其初始值，例如：

```
//定义结构体类型 Student
struct Student
{
    string   classId;              //班级
    string   studentId;            //学号
    string   name;                 //姓名
    float    scoreOfComp;          //计算机应用成绩
    float    scoreOfCplusplus;     //C＋＋程序设计成绩
} stu1={"计算机一班","0503001","张三",89,78},stu2;
```

如果采取的是类型声明与变量定义分离的情况，即在已定义好结构体类型的情况下（例如现已定义好结构体类型 Student），则可以对结构体变量 stu3 进行如下的初始化：

Student stu3 = {"计算机一班","0503003","李四",92,86};

注意：这种直接赋值方法如果不是在定义时使用，则是错误的，将无法通过编译，如已定义结构体变量 stu1，则下列代码是错误的：

```
stu1 = {"计算机一班","0503001","张三",89,78};  //注意前面没有类型 Student
```

2. 结构体变量赋值

在定义结构体变量时，可以使用一个已经存在的同类型结构体变量进行赋值，如在已定义好结构体类型 Student 且结构体变量 stu1 也已进行赋值的情况下，下列两种代码都是正确的。

代码 1：

```
Student stu3 = stu1;      //定义 stu3 的同时，将 stu1 的值赋给 stu3
```

代码 2

```
Student stu3;     //定义 stu3，但没有赋初值
stu3 = stu1;      //将 stu1 的值赋给 stu3
```

上面两段代码的作用是一致的，都定义了一个新的结构体变量 stu3，并将 stu1 的值赋给 stu3。执行完后，stu1 和 stu3 的每个成员都具有同样的值。

3. 使用成员引用的方式赋值

可以采用为结构体变量中每个成员单独赋值的方式来进行初始化。引用结构体变量中成员的一般方式为：

<结构体变量名>.<成员名>

“.”运算符表示从属关系，称为成员运算符，例如要表示学生 1 的姓名，则可以用 stu1.name 来表示。通过成员运算符对每个成员逐一赋值，也可以实现对结构体变量的初始化。如果要键盘接收 stu1 的各种情况，则可以用以下代码实现，设已定义好结构体类型 Student 和结构体变量 stu1：

```
cout << "请输入班级号        :";
cin >> stu1.classId;
cout << "请输入学号          :";
cin >> stu1.studentId;
cout << "请输入姓名          :";
cin >> stu1.name;
cout << "计算机应用成绩      :";
cin >> stu1.scoreOfComp;
cout << "C++程序设计成绩:";
cin >> stu1.scoreOfCplusplus;
```

3.1.4 结构体变量的引用

学习了定义结构体类型和结构体类型变量后，怎样正确地引用该结构体类型变量的成员呢？主要是使用成员运算符来实现，现将有关结构体变量的知识列举如下。

1. 整体引用

可以将一个结构体变量的值作为一个整体来使用，如 3.1.3 节的第二种方式所述，要注意的是这种整体引用方式仅适用于同类型结构体变量的赋值，例如：

```
stu1 = stu3;
```

不能使用标准输入/输出语句对结构体变量进行整体输入或输出，例如：

```
cin >> stu1;      //错误的代码
cout << stu1;     //错误的代码
```

因为结构体类型是自定义用户类型，在系统的基本输入/输出语句中无法事先针对自定义类型进行编程而实现整体输入/输出，当然用户可以自己编写特定代码来实现整体输入/输出。

2. 分别引用

除了进行整体引用外，我们还可以使用成员运算符“.”对结构体变量中的成员进行引用和赋值，这里列举几个在使用成员运算符时应注意的要点。

（1）成员运算符的优先级。

成员运算符的优先级在所有运算符中是**最高**的，因此可以将 stu1.name 作为一个整体来看待。

（2）成员运算符的嵌套使用。

如果成员本身也是一个结构体变量，则可以使用成员运算符逐级引用，直到找到最低一级的成员。如对下列成绩单而言：

班级	学号	姓名	出生年月			计算机应用	C++程序设计
			年	月	日		

则可以为出生年月定义一个结构体类型 date，那么 Student 的定义修改如下：

```
struct date
{
   int year;       //年
   int month;      //月
   int day;        //日
};
struct Student
{
    string   classId;              //班级
    string   studentId;            //学号
    string   name;                 //姓名
    date   birthday                //出生日期   这里是新增的
    float    scoreOfComp;          //计算机应用成绩
    float    scoreOfCplusplus;     //C++程序设计成绩
}stu1,stu2;
```

现在要引用学生 stu1 出生年月的年份，则可以按下列代码进行引用：

```
cout<<stu1.birthday.year<<endl;
```

（3）引用结构体变量中成员的地址。

结构体变量中的成员同一般变量一样，可以进行各种运算，包括使用取地址运算符“&”获取成员的地址，参与指针运算。例如：

```
cout << &stu1.name << endl; //输出 stu1.name 的地址
```

示例 3.1.1　编写代码，实现如下功能：

（1）定义结构体类型 Student，其结构如表 3.1.1 所示。

表 3.1.1　学生信息字段表

序号	字段名	字段类型	描述
1	studentId	string	学号
2	name	string	姓名
3	classId	string	班级
4	sex	string	性别
5	Math	float	高数成绩
6	English	float	大学英语成绩
7	Cplusplus	float	C++成绩
8	sum	float	总成绩
9	average	float	平均成绩

（2）定义结构变量 stu1，从键盘接收如表 3.1.2 所示的数据并输出。

表 3.1.2　学生信息表

学号	学生姓名	班级	性别	高数成绩	大学英语成绩	C++成绩	总成绩	平均成绩
0503001	张三	计算机一班	男	76.00	92.00	82.00		

程序清单：

```
/*****************************************
 * 程序名: 3_1_1.cpp
 * 功能: 定义学生结构体，并完成学生信息的输入
*****************************************/
#include <iostream >          //使用标准输入/输出库
using namespace std;          //使用标准命名空间
struct Student
{
      string studentId;       //学号
       string name;           //姓名
      string classId;         //班级
       string sex;            //性别
       float Math;            //高数成绩
       float English;         //大学英语成绩
       float Cplusplus;       //C++成绩
       float sum;             //总成绩
       float average;         //平均成绩
}stu1;
int main()
{
      cout<<"请输入学生的学号："<<' ';
      cin>>stu1.studentId;
      cout<<"请输入学生的姓名："<<' ';
      cin>>stu1.name;
      cout<<"请输入学生的班级："<<' ';
      cin>>stu1.classId;
      cout<<"请输入学生的性别："<<' ';
      cin>>stu1.sex;
      cout<<"请输入学生的高数成绩："<<' ';
      cin>>stu1.Math;
      cout<<"请输入学生的大学英语成绩："<<' ';
      cin>>stu1.English;
      cout<<"请输入学生的 C++成绩："<<' ';
      cin>>stu1.Cplusplus;
      //计算学生的总成绩
      stu1.sum=stu1.Math+stu1.English+stu1.Cplusplus;
      //计算学生的平均成绩
      stu1.average=stu1.sum/3;
      //输出学生信息
      cout<<"**************************************"<<endl;
      cout<<"学生的学号是："<<stu1.studentId<<endl;
      cout<<"姓名："<<stu1.name<<endl;
      cout<<"班级："<<stu1.classId<<endl;
```

```
    cout<<"性别："<<stu1.sex<<endl;
    cout<<"高数成绩："<<stu1.Math<<endl;
    cout<<"大学英语成绩："<<stu1.English<<endl;
    cout<<"C++成绩："<<stu1.Cplusplus<<endl;
    cout<<"总成绩："<<stu1.sum<<endl;
    cout<<"平均成绩："<<stu1.average<<endl;
    system("pause");          //暂停，按任意键结束
    return 0;
}
```

将上面程序编译、连接后，运行的结果如图 3.1.2 所示。

图 3.1.2　结构体变量程序运行结果

练习 3.1.1　编写代码，实现如下功能：

（1）定义结构体类型 Book，其结构如表 3.1.3 所示。

表 3.1.3　书籍信息字段表

序号	字段名	字段类型	描述	备注
1	id	string	书号	最大 8 位字符
2	name	string	书名	最多 15 个汉字
3	author	string	作者	最多 5 个汉字
4	publisher	string	出版社	最多 15 个汉字
5	price	float	价格	
6	flag	int	外借	1 表示外借，0 表示在库

（2）定义结构变量 book，从键盘接收如表 3.1.4 所示的数据并输出。

表 3.1.4　书籍信息表

书号	书名	作者	出版社	价格	外借
TP050001	C++程序设计	谭浩强	清华大学出版社	36.00	是

归纳小结

通过本节学习，我们应该掌握如下内容:

1. 用户自定义类型

当基本数据类型无法满足开发者的使用要求时，我们需要根据实际情况的要求，利用基本类型来声明一些新的类型，这些由用户自己声明的新的数据类型我们称为用户自定义类型，包括结构体类型、枚举类型和共用体类型等。

2. 结构体类型的定义

定义结构体类型的一般形式如下:

```
struct 结构体类型名 {
    成员列表;
};
```

3. 结构体变量的定义

结构体变量的定义可以分为三种类型，建议使用最规范的第一种。

- 先定义结构体类型，再定义结构体类型变量
- 定义结构体类型的同时定义结构体类型变量
- 直接定义结构体类型变量

4. 结构体变量的初始化

结构体的变量的初始化也可以分为三种方式:

- 定义时直接初始化
- 结构体变量赋值
- 使用成员引用的方式赋值

5. 结构体变量的引用

可分为整体引用和分别引用两种，要注意的是，整体引用时是无法进行整体输入/输出操作的。

作业

一、选择题

1．对如下结构体的说明和语句：

```
struct Time
{
    int hour,min,second;
};
struct workday
{
    Time worktime;
}shop;
```

对结构体变量 shop 的营业时间赋值时，下面正确的赋值语句是（　　）

A．hour=9;　　B．worktime.hour=9;

C．shop.worktime.hour=9;　　D．shop.hour=9;

2．对如下结构体的说明和语句：

```
struct  S
{
    int g;   char h;
}T;
```

下面叙述正确的是（　　）

A．可用 S 定义结构体变量　　B．可用 T 定义结构体变量

C．S 是 struct 类型的变量　　D．T 是 struct S 类型的变量

二、填空题

计算下列各结构体类型的长度。

①struct A{int a; double d;};　　长度：_____

②struct B{char s[10];int a ; A k;};　　长度：_____

三、编程题

1．定义一个结构体类型来描述下表

姓名 （字符串）	性别 （字符）	职业 （字符串）	年龄 （整型）	身份证号码 （字符串或长整型）

2．在第 1 题的基础上，完成一个可以接收并显示 3 个用户数据的程序。

3．以下程序为在结构体数组中查找分数最高与分数最低的学生姓名和成绩，请填空。

```
#include <iostream>          //使用标准输入/输出库
using namespace std;         //使用标准命名空间
struct sheet
{
  char name[10];
  int score;
};
int main()
{
    int max,min,i;
    sheet stud[5] = {{"丁一",88},{"王二",78},{"张三",69},{"李四",93},{"唐五",81}};
    max = min = 0;
    for(i = 0; i < 5; i++)
    {
        if(stud[i].score > stud[max].score)
        {
          _____A_____
        }
        if(stud[i].score < stud[min].score)
        {
          _____B_____
        }
    }
```

```
    cout<<"最高分姓名与成绩"<<  C  <<  D  <<endl;
    cout<<"最低分姓名与成绩"<<  E  <<  F  <<endl;
    system("pause");
    return 0;
}
```

任务 3.2　数组

任务分析

根据学生成绩管理系统的需求，要求完成系部所有学生的成绩统计，包括按高数、大学英语和 C++等单科分数排名，按总分或平均分排名，能统计最高总分或最低总分的学生信息、成绩及格率等。任务 3.1 已经实现了将单个学生的所有信息定义为一个结构体，本节任务要求在此基础上完善上述成绩统计管理模块。

相关知识

1. 一维数组的定义
2. 一维数组的初始化
3. 一维数组的应用
4. 二维数组的定义
5. 二维数组的初始化
6. 二维数组的使用

实现方法

使用数组是 C++语言与其他编程语言共同拥有的一大特点。通过数组，我们可以描述同一类具有相同数据类型的变量，方便我们对数据的查询和管理。

数组是由类型相同、逻辑意义相关的一组数据构成。从存储角度来看，如果一个变量代表一个存储单元，那么一个数组则代表连续的存储单元。每个数组都有一个由标识符充当的名字，也称为**数组名**。数组中的每个元素叫做**数组元素**，它们按顺序分配在内存中一片连续的内存单元中，每个元素占用相同数目的单元。数组元素是按顺序排列的，顺序号叫做**数组元素的下标**。数组元素通过数组名和下标来表示。数组可以有多个下标，下标的个数叫做**数组的维数**。数组按维数多少可分为一维数组和多维数组，按数组元素的类型可分为字符数组、整型数组、实型数组和指针数组等。

3.2.1　一维数组的定义

一维数组中的元素（单元）是线性排列的，并用下标 0,1,2…标识它的每个元素。如果一维数组共有 n 个元素，则其下标是 0～（n-1）范围内的一个无符号整数。一维数组只有一个下标。一般形式为：

数据类型 数组名[整常量表达式]

数据类型指明了数组元素的类别，可以是整型、实型、布尔型和字符型等简单数据类型，也可以是用户定义的复合数据类型，包括数组类型。数组名由标识符充当，代表了一个数组变量。“[ ]”是作为运算符处理的，其优先级与括号相同，从左向右结合；方括号中的整常量表达式表示数组元素的个数，**注意数组元素的下标是从零开始计数的**。例如：

```
int nNumber[10];
```

声明了 nNumber 是一个整型数组，它有 10 个整型元素，每个元素都是整型的变量，下标的变化范围是 0～9。

数组元素是通过数组名和下标来访问的。程序中数组元素的表示形式为：

数组名[下标表达式]

下标表达式是值为整型的表达式，它指明了拟访问的数组元素的下标。例如，数组 nNumber 的 10 个元素可以依次表示为：

nNumber [0], nNumber [1], nNumber [2], …, nNumber [9]

在定义一维数组时要注意以下几点：

（1）数据类型一般指的是该数组元素的类型。

（2）数组名的命名规则要遵循 C++ 关于标识符的命名规则。

（3）数组定义中的整常量表达式的值就是数组长度，是用方括号括起来的一个整数，不能用圆括号。

（4）必须要说明数组的长度，且这个长度要用一个确定的正整数来表示。例如：

```
int a[ ];     //不合法，没有定义数组的长度
int x;
int a[x]      //不合法，不能用一个变量定义数组的长度
```

（5）可以依次说明多个同类型数组。例如：

```
int a[10],b[12],c[16];          //定义 a, b ,c 三个整型数组
```

如果访问数组元素时提供的下标超出了元素的数目，会导致访问越界错误。

3.2.2 一维数组的初始化

数组同变量一样需要初始化，数组的初始化可采用以下 4 种方法实现：

（1）在声明数组时，可以对数组中开始若干个元素乃至全部元素进行初始化，其方法是把初始值按顺序放在花括号中，数值之间用逗号分开，例如：

```
int a[10]={1,2,3,};
float b[20]={1,3,5,5,7}
double c[6]={6,8,9,4,7}
```

（2）在定义数组时，既可以对所有元素进行初始化，也可以只对其中的一部分元素进行初始化。如果初始值的数目小于数组元素的数目，数组剩余的元素被自动初始化为 0，例如：

```
int d[6]={0};      //实际上将所有的数组元素都初始化为 0
```

（3）全部数组元素赋初始值时，可以不指定数组长度。计算机会根据初始化的数组元素的个数自动分配存储空间。如果声明数组时省略了数组元素的数目，那么系统将根据初始值的数目来确定数组元素的数目，这时数组元素的个数与初始值的个数相同。例如：

```
int e[ ]={1,2,3,4,5};
```

实际上是声明了一个包含 5 个整型元素的数组 f，并对每个数组元素进行了初始化，各元素的值为 e[0]=1，e[1]=2，e[2]=3，e[3]=4，e[4]=5。

（4）利用 for 循环语句赋初始值。例如：

```
for(int i=0;i<100;i++)
    a[i]=i+1;
```

3.2.3 一维数组元素的访问

系统对一维数组元素的访问使用下标法，即指出数组名和下标值，系统就会找到该元素。数组用其下标变化实行对内存中的数组元素进行处理。数组元素是通过数组名和下标来访问的。程序中数组元素的表示形式为：

数组名[下标表达式]

其中，下标表达式是值为整型的表达式，它指明了被访问的数组元素的下标。例如：

```
int nNumber[6];
nNumber[0] = 6;
nNumber[1] = 8;
nNumber[2] = 168;
//注意，数组中定义了 6 个元素，那么数组下标最大为 5，因为数组的开始下标为 0
nNumber[5] = 0;
```

如果访问数组元素时提供的下标超出了元素的数目，则会导致访问越界错误，C++不检查访问越界，这一点要特别注意。对于上述代码中的数组，nNumber：编译系统在一定内存区域为该数组分配了存放整型数据的 6 个连续空间，它们分别是 nNumber[0]、nNumber[1]、nNumber[2]、nNumber[3]、nNumber[4]、nNumber[5]。nNumber[i]表示从数组存储首地址开始的第 i 个元素变量。在程序中，通过 i 的变化就可以处理数组中的任何元素，nNumber[i]就是用下标法表示的数组元素。

示例 3.2.1 数组元素的引用。

程序清单：

```
/*******************************************
 * 程序名:3_2_1.cpp
 * 功能:数组元素的引用
 *******************************************/
#include <iostream>          //使用标准输入/输出库
using namespace std;         //使用标准命名空间
int main()
{
    int i,a[10];
    for(i=0;i<10;i++)
        a[i]=i;
    for(i=9;i>=0;i--)
        cout<<"  "<<a[i];
    cout<<endl;
```

```
    system("pause");
    return 0;
}
```

程序运行结果如图 3.2.1 所示。

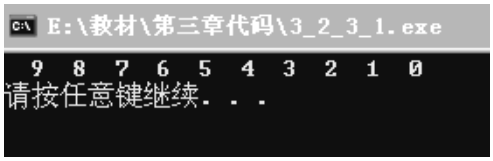

图 3.2.1　数组元素程序运行结果

在 C++ 语言中，数组之间不能整体赋值，要将一个数组赋给另一个数组，只能逐一元素地进行赋值。

示例 3.2.2　数组间的赋值。

程序清单：

```
/*******************************************
 * 程序名:3_2_2.cpp
 * 功能:数组间的赋值
 *******************************************/
#include <iostream>          //使用标准输入/输出库
using namespace std;         //使用标准命名空间
int main()
{
    int from[]={1,2,3,4,5},to[5];
    for(int i=0;i<5;i++)
        to[i] = from[i];
    cout<<"数组 from：\t";
    for(int i=0;i<5;i++)
        cout<<from[i]<<'\t';
    cout<<endl;
    cout<<"数组 to：\t";
    for(int i=0;i<5;i++)
        cout<<to[i]<<'\t';
    cout<<endl;
    system("pause");
    return 0;
}
```

程序运行结果如图 3.2.2 所示。

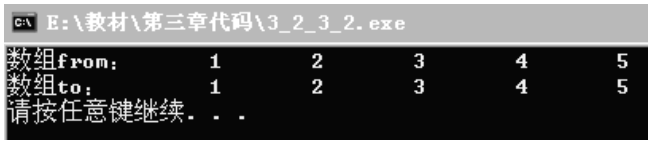

图 3.2.2　数据复制程序运行结果

3.2.4 一维数组的应用

示例 3.2.3 请定义一个存储 8 个整型数据的数组，数组中存放从 1 到 8 这 8 个连续的数据，并将其内容输出。

分析：前面已经学习了用数组存储相同类型的变量，在这个示例当中要求存放 8 个整型数据，因此，可以定义一个 int 类型的数组 nNumber[8]，数组的大小为 8，定义一个计数器 nCount 来指示数组的下标，通过 for 循环可以将数组中的数据按顺序输出。

程序清单：

```
/*******************************************
 * 程序名:3_2_3.cpp
 * 功能:数组的应用
 *******************************************/
#include <iostream>          //使用标准输入/输出库
using namespace std;         //使用标准命名空间
int main()
{
    int nNumber[8];          //声明一个整形数组，包括 8 个整型数据
    for(int nCount = 0; nCount < 8; nCount++)      //用循环语句将数组中的元素初始化
    {
        nNumber[nCount] = nCount+1;                //给第 nCount 个元素赋值
        //输出数组中的元素
        cout << "nNumber[" << nCount << "]   is : " << nNumber[nCount] << endl;
    }
    system("pause");
}
```

程序运行结果如图 3.2.3 所示。

图 3.2.3 数组应用程序运行结果

示例 3.2.4 请从键盘接收学生的个数，定义一个存储指定数目学生的 C++成绩的数组，接收所有学生的成绩并将其内容输出，统计最高分、最低分、总分和平均分。

程序清单：

```
/*******************************************
 * 程序名:3_2_4.cpp
 * 功能:接收学生的成绩并将其内容输出，统计最高分、最低分、总分和平均分
 *******************************************/
```

```
#include <iostream>          //使用标准输入/输出库
using namespace std;         //使用标准命名空间
int main()
{
    float stu[100];          //声明一个可用于存放 100 个学生成绩的变量
    //声明学生的最高分、最低分、总分和平均分变量，并将最高分和总分初始化为零，最低分初始化为满分 100
    float highest_score=0,lowest_score=100,total_score=0,average_score;
    int count;               //声明学生数目的变量
    cout<<"请输入学生的个数:";
    cin>>count;
    //使用循环，输入 count 个数目的学生成绩，在输入的同时比较最高分和最低分，并计算总分
    for(int i=0;i<count;i++)
    {
        cout<<"请输入第"<<i+1<<"个学生的 C++成绩:"<<endl;
        cin>>stu[i];
        //对比新输入的学生成绩与最高分，将两者的最高作为新的最高分
        if(highest_score < stu[i])
            {highest_score = stu[i];}
        //对比新输入的学生成绩与最低分，将两者的最低作为新的最低分
        if(lowest_score > stu[i])
            {lowest_score = stu[i];}
        //判断学生成绩是否及格，如果及格，则将及格个数变量加一
        total_score += stu[i];          //计算总分，每输入一个学生成绩就将其累加到总分
    }
    average_score =  total_score / count;    //计算平均分：平均分=总分/学生数目
    cout<<"最高分为："<<highest_score<<endl;
    cout<<"最低分为："<<lowest_score<<endl;
    cout<<"总成绩为："<<total_score<<endl;
    cout<<"平均成绩为："<<average_score<<endl;
    system("pause");
    return 0;
}
```

程序运行结果如图 3.2.4 所示。

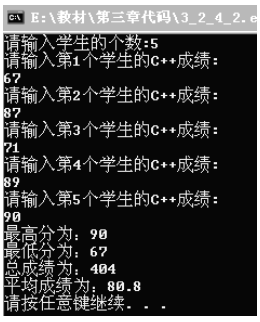

图 3.2.4　运行结果

练习 3.2.1　求一个数组 m[9]，使 m[i]的值为下标值的平方，然后按逆序输出。

3.2.5 二维数组的定义

前面学习的一维数组可以看成是数学上的向量在计算机中的表示方式，那么数学上的矩阵如何表示呢？在 C++中也可以构造多维数组来表示矩阵。C++中有多个下标的数组称为多维数组，具有两个下标表示的数组称为二维数组。多维数组可以有多于两个的下标，C++ 编译器支持最多 12 个下标的数组。例如：

```
int a [5][6];
char b [4][3][5];
```

声明 a 是一个整型二维数组，b 是字符型三维数组。二维数组是多维数组中最简单、最常用的多维数组。数学上的二维矩阵可以用二维数组表示出来。下面仅讨论二维数组，多维数组也可以类似地使用。

二维数组的一般形式：

数据类型 数组名[常量表达式 1][常量表达式 2]

在 C++ 中，二维数组也可以看成是一种特殊的一维数组，即它的元素又是一个一维数组。上面说明的二维数组 a 有两个下标，第一个下标称为行，变化的范围是 0～4；第二个下标称为列，变化范围是 0～5。因此，a 有 5 行 6 列，共 5*6=30 个元素，其逻辑结构如表 3.2.1 所示。每个数组元素用数组名和两个下标表示。例如，a [1][2]和 a [2][1] 分别表示第 1 行第 2 列的元素和第 2 行第 1 列的元素。

表 3.2.1 二维数组 a 的逻辑结构

	第 0 列	第 1 列	第 2 列	第 3 列	第 4 列	第 5 列
第 0 行	a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]
第 1 行	a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]
第 2 行	a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]	a[2][5]
第 3 行	a[3][0]	a[3][1]	a[3][2]	a[3][3]	a[3][4]	a[3][5]
第 4 行	a[4][0]	a[4][1]	a[4][2]	a[4][3]	a[4][4]	a[4][5]

注意：

（1）同一维数组一样，二维数组的下标也从 0 开始。

（2）常量表达式的要求与一维数组的要求一样，同时，常量表达式 1 和常量表达式 2 必须分开写在两个方括号内。

（3）二维数组中，元素在内存中存放的顺序是：先放第一行，然后放第二行，依此类推，直到存储完毕。

3.2.6 二维数组的初始化

二维数组元素的初始化方法很多，常用的有以下几种：

（1）将全部值连续写在一个花括号内，按照数组排列的顺序给各元素赋初值。例如：

```
int nNumber[2][3] = {1,2,3,4,5,6};
```

上面的例子给数组中六个元素赋初值，相当于：

```
nNumber[0][0] = 1;    nNumber[0][1] = 2;    nNumber[0][2] = 3;
nNumber[1][0] = 4;    nNumber[1][1] = 5;    nNumber[1][2] = 6;
```

（2）分行给数组元素赋初值。

```
int nNumber[2][3] = {{1,2,3},{4,5,6}};
```

显然，方法 2 是一行对一行，界限清楚；如果数据过多，用方法 1 很容易遗漏和增加数据，不易检查。

3．可以只给部分数组元素赋初值。例如：

```
int nNumber [4][3]={{1},{5},{6},{7}};
```

该语句的作用只对每行第一列的数组元素赋初值，其余元素为 0，相当于：

```
nNumber[0][0] = 1;    nNumber[0][1] = 0;    nNumber[0][2] = 0;
nNumber[1][0] = 5;    nNumber[1][1] = 0;    nNumber[1][2] = 0;
nNumber[2][0] = 6;    nNumber[2][1] = 0;    nNumber[2][2] = 0;
nNumber[3][0] = 7;    nNumber[3][1] = 0;    nNumber[3][2] = 0;
```

需要注意的是：给部分元素赋初值时，最好用这种方法。如果用方法 1 来赋初值，例如：

```
int nNumber[4][3]={1,5,6,7};
```

则相当于 a[0][0]=1，a[0][1]=5，a[0][2]=6，a[0][3]=7，其余的元素默认为 0。

（4）可以对某一行中的某一元素赋初值，也可以对某一行不赋值。例如：

```
int nNumber[4][3]= {{0,1},{}, {0,0,2},{0,1}};
```

该语句的作用相当于：

```
nNumber[0][0] = 0;    nNumber[0][1] = 1;    nNumber[0][2] = 0;
nNumber[1][0] = 0;    nNumber[1][1] = 0;    nNumber[1][2] = 0;
nNumber[2][0] = 0;    nNumber[2][1] = 0;    nNumber[2][2] = 2;
nNumber[3][0] = 0;    nNumber[3][1] = 0;    nNumber[3][2] = 1;
```

这种方法经常用于数组元素中 0 比较多的情况。

（5）如果对全部数组元素赋初值，则定义数组时，第一维的长度可以不指定，但第二维的长度不能省略。例如：

```
int nNumber[2][3]={1,2,3,4,5,6};
```

可以写成：

```
int nNumber[][3]={1,2,3,4,5,6};
```

注意：

（1）可以只对数组的部分元素赋初值，其他元素按照不同数据类型的默认值给数组元素赋初值。

（2）如果对数组中所有元素赋初值，则定义数组时可以不指定数组的行数，但是列的数目必须明确写出。例如：

```
int nNumber[][3] = {{1,2,3},{6,8,6}};      //数组的行数为 2
```

3.2.7 二维数组元素的访问

C++ 规定只能逐个引用数组元素而不能一次引用整个数组。数组元素的引用方式如下：

数组名[常量表达式 1][常量表达式 2]

例如：int a[3][4] 定义了一个 3 行 4 列的整数数组，如果要引用其中的第 3 行第 1 列元素，应写为 a[2][0] 。

二维数组的常量表达式可以是整数常量或整型表达式。如果行下标为 m，列下标为 n，则该二维数组共包含 m*n 个数组元素。其中行下标最大值为 m-1，列下标最大值为 n-1。

注意：下标值要在定义的数组的最大范围内，即不要越界。

示例 3.2.5 请定义一个存储整型数据的二维数组（2 行 3 列），并将数组的元素输出。

分析：通过学习二维数组的基本知识，可以很容易地定义一个存储整型数据的二维数组，但是，如何将数组中的内容输出呢？从以上二维数组的定义可以发现，通过数组的行标与列标的不同，可以定位数组中的元素。先定位行，然后找到行中所有的元素，将行标下移，完成第二行中所有数据的输出，依次类推，直到将数组中的所有数据输出。这个过程可以通过循环嵌套来完成。

程序清单：

```
/*****************************************
 * 程序名:3_2_5.cpp
 * 功能:二维数组中数据的输出
 *****************************************/
#include <iostream>          //使用标准输入/输出库
using namespace std;         //使用标准命名空间
int main()
{
    int nNumber[2][3] = {1,2,3,4,5,6};        //声明一个整型二维数组，包括 6 个整型数据
    for(int i = 0; i < 2; i++)                //用循环语句将数组中的元素输出，先定位行
    {
        for(int j = 0; j < 3; j++)            //用循环语句将数组中的元素输出，再定位列
        {
         //输出数组中的元素
         cout << "nNumber[" << i << "][" << j << "] is : " << nNumber[i][j] << endl;
         }
    }
    system("pause");
}
```

程序运行结果如图 3.2.5 所示。

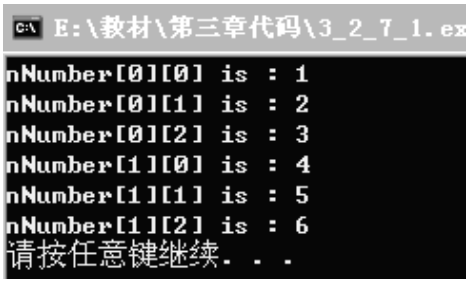

图 3.2.5 二维数组程序运行结果

3.2.8 二维数组的应用

示例 3.2.6 求一个 3×3 矩阵对角线元素之和。

分析：我们可以通过二维数组来存储矩阵，使用循环嵌套来定位行标和列标。在 3×3 的二维数组中，依次输入矩阵的每个元素。由于矩阵对角线上的元素对应的行标和列标是一致的，所以只要将所有 matrix[i][i]的元素累计即可得到对角线元素之和。

程序清单：

```
/*****************************************
 * 程序名:3_2_6.cpp
 * 功能:求一个 3×3 矩阵对角线元素之和
 *****************************************/
#include <iostream>          //使用标准输入/输出库
using namespace std;         //使用标准命名空间
int main()
{
    //定义 matrix 为 3×3 二维数组的数组名，sum 为累加和，初始化为 0
    float matrix[3][3],sum=0;
    cout<<"请输入矩阵元素："<<endl;
    //通过循环嵌套输入矩阵的元素
    for(int i=0;i<3;i++)          //变量 i 控制矩阵的行标
        for(int j=0;j<3;j++)      //变量 j 控制矩阵的列标
            cin>>matrix3 [i][j];
    cout<<endl;
    //matrix[i][i]为对角线上的元素
    for(int i=0;i<3;i++)
        sum=sum+matrix[i][i];
    cout<<"对角线元素之和为："<<sum<<endl;
    system("pause");
}
```

程序运行结果如图 3.2.6 所示。

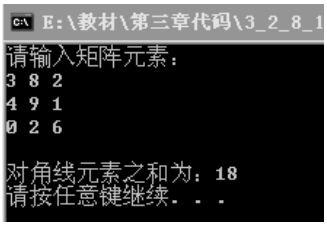

图 3.2.6 矩阵对角线元素求和程序运行结果

示例 3.2.7 建立一个九九乘法表。

分析：我们可以通过一个 9*9 的二维数组来存储九九乘法表。先使用两层循环嵌套来设置九九乘法表中的乘积，由于数组的下标是从 0 开始的，所以第 i 行第 j 列位置存储的乘积为(i+1)*(j+1)；将九九乘法表存储好后，再通过两层循环嵌套输出结果。

程序清单：

```
/*****************************************
 * 程序名:3_2_7.cpp
 * 功能:九九乘法表
 *****************************************/
#include <iostream>          //使用标准输入/输出库
#include<iomanip>            //输出格式控制
using namespace std;         //使用标准命名空间
int main()
{
    int tab[9][9];           //声明一个 9*9 的二维数组存储九九乘法表的乘积
    //利用循环嵌套，设置乘法表中的每个元素
    for(int i=0;i<9;i++)
    {
        for(int j=0;j<9;j++)
        {
            //由于数组下标都是从 0 开始的，所以第 i 行第 j 列的乘积应该是(i+1)*(j+1)
            tab[i][j]=(i+1)*(j+1);
        }
    }
    //利用循环嵌套，显示二维数组中存储的九九乘法表
    for(int i=0;i<9;i++)
    {
        for(int j=0;j<9;j++)
        {
            cout<<setw(4)<<tab[i][j];      //setw(4)的作用是设置显示的域宽为 4 个字符
        }
        //每显示完一行后，换行
        cout<<endl;
    }
    system("pause");
}
```

程序运行结果如图 3.2.7 所示。

图 3.2.7　九九乘法表运行结果

3.2.9 数组的应用

示例 3.2.8 根据学生成绩统计模块的需求，需要完成系部所有学生信息（学号、姓名、班级、性别、高数成绩，大学英语成绩、C++成绩）的录入，实现学生成绩的统计，包括按高数、大学英语和 C++等单科分数排名，按总分或平均分排名，能统计最高总分或最低总分学生信息、成绩及格率等。

分析：该程序需要解决如下几个问题。

（1）如何完成系部所有学生信息的录入。

解决方案：代码中我们先声明了一个变量名为 Student 的结构体，用于存放学生个人信息。然后声明了一个类型为 Student 的一维数组存放所有学生信息。当确定学生数量后，使用循环依次录入每个学生的数据。代码如下：

```
struct Student
{
    string studentId;       //学号
     string name;           //姓名
    string classId;         //班级
     string sex;            //性别
     float Math;            //高数成绩
     float English;         //大学英语成绩
     float Cplusplus;       //C++成绩
     float sum;             //总成绩
     float average;         //平均成绩
};
Student stud[200];
int number;                 //用于存放学生数量
……
//----------------录入学生信息开始------------------
cout<<"--------------录入学生信息开始-------------"<<endl;
cout<<"请根据你的需要输入学生的个数：";      //指定学生的数目
cin>>number;
//根据指定学生的数目，循环输入每个学生的基本信息，其中总成绩和平均成绩是在输入之后由程序计算得出
for(int i=0;i<number;i++)
{
    cout<<"请输入第"<<i+1<<"个学生资料："<<endl;
    cout<<"学号："<<' ';
    cin>>stud[i].studentId;
     ……
     cout<<"高数成绩："<<' ';   cin>>stud[i].Math;
     cout<<"大学英语成绩："<<' ';   cin>>stud[i].English;
     cout<<"C++成绩："<<' ';   cin>>stud[i].Cplusplus;
     //计算第 i 个学生的总成绩
     stud[i].sum=stud[i].Math+stud[i].English+stud[i].Cplusplus;
```

```
        //计算第 i 个学生的平均成绩
        stud[i].average=stud[i].sum/3;
    }
    cout<<"--------------录入学生信息结束-------------"<<endl<<endl;
    //----------------录入学生信息结束------------------
```

需要注意的是：这里的一维数组大小为 200，也就是说，学生的数量最多不能超过 200，否则数组下标越界。另外，每个学生的总成绩和平均成绩是在循环体中每次分别录入高数、大学英语和 C++成绩之后计算得出的。

（2）如何选择不同类别的统计，如单科排名、最高总分等。

解决方案：代码中我们定义了一个整型变量 choose，并设计了一个学生成绩统计管理模块的菜单，根据 choose 变量的输入值实现对菜单的选择，进而可以选择不同类别的统计；另外，为了控制程序可以根据需要多次统计，我们默认 choose 的值为 0 时退出成绩统计模块。因此使用 while 循环，判断只要 choose 的输入值不为 0，则可进入循环体进行成绩统计。代码如下：

```
int choose;    //菜单选择变量
while(choose!=0){
    cout<<"\t\t\t**************************************"<<endl;
    cout<<"\t\t\t**************************************"<<endl;
    cout<<"\t\t\t*              学生成绩统计管理              *"<<endl;
    cout<<"\t\t\t*                                            *"<<endl;
    cout<<"\t\t\t*              1.按总分排名                  *"<<endl;
    cout<<"\t\t\t*              2.按高数分数排名              *"<<endl;
    cout<<"\t\t\t*              3.按大写英语分数排名          *"<<endl;
    cout<<"\t\t\t*              4.按 C++分数排名               *"<<endl;
    cout<<"\t\t\t*              5.最高总分                    *"<<endl;
    cout<<"\t\t\t*              6.最低总分                    *"<<endl;
    cout<<"\t\t\t*              7.按平均分排名                *"<<endl;
    cout<<"\t\t\t*              8.及格率                      *"<<endl;
    cout<<"\t\t\t*              0.返回                        *"<<endl;
    cout<<"\t\t\t*                                            *"<<endl;
    cout<<"\t\t\t*              按 Enter 继续                  *"<<endl;
    cout<<"\t\t\t**************************************"<<endl;
    cout<<"\t\t\t**************************************"<<endl;
    cout<<"\t\t              请输入 0-8 之间的任意一数字:"<<' ';
    cin>>choose;
    switch(choose)
    {……}
}
```

（3）如何进行学生总分、平均分或者单科成绩排名。

解决方案：我们使用示例 2.2.12 中介绍的冒泡排序算法对学生的总分、平均分、每门单科成绩进行排序。因此代码基本相似，只是每次排序时比较的值不同而已。

（4）如何统计及格率。

解决方案：先定义三个用于统计及格人数的变量 Math_count、English_count 和 C_count，

用来分别存放学生的高数、大学英语和 C++及格学生数目。通过循环，依次遍历学生的三门单科成绩，当成绩大于 60 分时，则在对应科目的及格人数上进行累加。遍历完成后，用三门单科的及格人数除以学生总数便得到单科的及格率。需要注意的是：由于及格人数变量和学生总数变量 number 均是整型变量，所以计算及格率时要强行转换类型为 float 才行，否则及格率的值将不是 0 就是 1。

程序清单：

```
/*****************************************
 * 程序名:3_2_8.cpp
 * 功能:学生成绩统计模块
 *****************************************/
#include <iostream>          //使用标准输入/输出库
#include<iomanip>            //输出格式控制
using namespace std;         //使用标准命名空间
struct Student
{
      string studentId;      //学号
      string name;           //姓名
      string classId;        //班级
      string sex;            //性别
      float Math;            //高数成绩
      float English;         //大学英语成绩
      float Cplusplus;       //C++成绩
      float sum;             //总成绩
      float average;         //平均成绩
};
Student stud[200];
int number;
int main()
{
    //----------------录入学生信息开始------------------
    cout<<"--------------录入学生信息开始-------------"<<endl;
    //指定学生的数目
    cout<<"请根据你的需要输入学生的个数：";
    cin>>number;
    //根据指定学生的数目，循环输入每个学生的基本信息，
    其中总成绩和平均成绩是在输入之后由程序计算得出
    for(int i=0;i<number;i++)
    {
            cout<<"请输入第"<<i+1<<"个学生资料："<<endl;
            cout<<"学号："<<' ';
            cin>>stud[i].studentId;
            cout<<"姓名："<<' ';
            cin>>stud[i].name;
            cout<<"班级："<<' ';
```

```
            cin>>stud[i].classId;
            cout<<"性别："<<' ';
            cin>>stud[i].sex;
            cout<<"高数成绩："<<' ';
            cin>>stud[i].Math;
            cout<<"大学英语成绩："<<' ';
            cin>>stud[i].English;
            cout<<"C++成绩："<<' ';
            cin>>stud[i].Cplusplus;
            //计算第 i 个学生的总成绩
            stud[i].sum=stud[i].Math+stud[i].English+stud[i].Cplusplus;
            //计算第 i 个学生的平均成绩
            stud[i].average=stud[i].sum/3;
}
cout<<"--------------录入学生信息结束-------------"<<endl<<endl;
//----------------录入学生信息结束------------------


int choose;     //菜单选择变量
while(choose!=0){
cout<<"\t\t\t**************************************"<<endl;
 cout<<"\t\t\t**************************************"<<endl;
cout<<"\t\t\t*              学生成绩统计管理              *"<<endl;
cout<<"\t\t\t*                                            *"<<endl;
cout<<"\t\t\t*              1.按总分排名                  *"<<endl;
cout<<"\t\t\t*              2.按高数分数排名              *"<<endl;
cout<<"\t\t\t*              3.按英语分数排名              *"<<endl;
cout<<"\t\t\t*              4.按 C++分数排名              *"<<endl;
cout<<"\t\t\t*              5.最高总分                    *"<<endl;
cout<<"\t\t\t*              6.最低总分                    *"<<endl;
cout<<"\t\t\t*              7.按平均分排名                *"<<endl;
cout<<"\t\t\t*              8.及格率                      *"<<endl;
cout<<"\t\t\t*              0.返回                        *"<<endl;
cout<<"\t\t\t*                                             *"<<endl;
cout<<"\t\t\t*              按 Enter 继续                  *"<<endl;
cout<<"\t\t\t**************************************"<<endl;
cout<<"\t\t\t**************************************"<<endl;
cout<<"\t\t              请输入 0-8 之间的任意一数字:"<<' ';
cin>>choose;
switch(choose)
{case 1:
   {//----------------按总分排名开始---------------------
       cout<<"****************************************"<<endl<<endl;
       cout<<"--------------按总分排名开始-------------"<<endl;
       string temp_sid,temp_name,temp_cid,temp_sex;float m,e,c,z,p;
```

```
        for(int j=0;j<number;j++)              //冒泡排序
            for(int i=number-1;i>0;i--)        //以下按各单科成绩和平均分排名也是运用冒泡排序
                if(stud[i].sum<stud[i-1].sum)
                  {//当第 i 个学生的总分小于第 i-1 个学生的总分时，将两个学生位置交换
                    temp_sid=stud[i].studentId; stud[i].studentId=stud[i-1].studentId; stud[i-1].studentId=temp_sid;
                    temp_name=stud[i].name;    stud[i].name=stud[i-1].name;    stud[i-1].name=temp_name;
                    temp_cid=stud[i].classId; stud[i].classId=stud[i-1].classId;    stud[i-1].classId=temp_cid;
                    temp_sex=stud[i].sex; stud[i].sex=stud[i-1].sex; stud[i-1].sex=temp_sex;
                    m=stud[i].Math;      stud[i].Math=stud[i-1].Math;        stud[i-1].Math=m;
                    e=stud[i].English;    stud[i].English=stud[i-1].English;        stud[i-1].English=e;
                    c=stud[i].Cplusplus; stud[i].Cplusplus=stud[i-1].Cplusplus; stud[i-1].Cplusplus=c;
                    z=stud[i].sum;        stud[i].sum=stud[i-1].sum;        stud[i-1].sum=z;
                    p=stud[i].average;    stud[i].average=stud[i-1].average;    stud[i-1].average=p;
                   }
        cout<<"学生按总分排名:"<<endl;
        cout<<"学号"<<' '<<"姓名"<<' '<<"性别"<<' '<<"班级"<<' '<<"总分"<<endl;
        for(int i=0;i<number;i++)
        {
          cout<<stud[i].studentId<<"       "<<stud[i].name<<"       "
          <<stud[i].sex<<"       "<<stud[i].classId<<"       "
          <<stud[i].sum<<endl;cout<<endl;
        }
        cout<<"--------------按总分排名结束-------------"<<endl<<endl;
        //----------------按总分排名结束------------------
        break;
    }
case 2:
    {
        cout<<"*****************************************"<<endl<<endl;
        //-----------------按高数分数排名开始-------------------
        cout<<"--------------按高数分数排名开始-------------"<<endl;
        string temp_sid,temp_name,temp_cid,temp_sex;float m,e,c,z,p;
        for(int j=0;j<number;j++)       //冒泡排序
            for(int i=number-1;i>0;i--)
                if(stud[i].Math<stud[i-1].Math)
                  {//当第 i 个学生的高数分数小于第 i-1 个学生的高数分数时，将两个学生位置交换
                    temp_sid=stud[i].studentId; stud[i].studentId=stud[i-1].studentId; stud[i-1].studentId=temp_sid;
                    temp_name=stud[i].name;    stud[i].name=stud[i-1].name;    stud[i-1].name=temp_name;
                    temp_cid=stud[i].classId; stud[i].classId=stud[i-1].classId;    stud[i-1].classId=temp_cid;
                    temp_sex=stud[i].sex; stud[i].sex=stud[i-1].sex; stud[i-1].sex=temp_sex;
                    m=stud[i].Math;      stud[i].Math=stud[i-1].Math;        stud[i-1].Math=m;
                    e=stud[i].English;    stud[i].English=stud[i-1].English;        stud[i-1].English=e;
                    c=stud[i].Cplusplus; stud[i].Cplusplus=stud[i-1].Cplusplus; stud[i-1].Cplusplus=c;
                    z=stud[i].sum;        stud[i].sum=stud[i-1].sum;        stud[i-1].sum=z;
                    p=stud[i].average;    stud[i].average=stud[i-1].average;    stud[i-1].average=p;
```

```
                }
        cout<<"学生按高数分数排名:"<<endl;
        cout<<"学号"<<' '<<"姓名"<<' '<<"性别"<<' '<<"班级"<<' '<<"高数分数"<<endl;
        for(int i=0;i<number;i++)
        {
          cout<<stud[i].studentId<<"       "<<stud[i].name<<"       "
          <<stud[i].sex<<"       "<<stud[i].classId<<"       "
          <<stud[i].Math<<endl;cout<<endl;
        }
        cout<<"--------------按高数分数排名结束-------------"<<endl<<endl;
        //----------------按高数分数排名结束------------------
        break;
    }
case 3:
    {
        cout<<"****************************************"<<endl<<endl;
        //----------------按英语分数排名开始------------------
        cout<<"--------------按英语分数排名开始-------------"<<endl;
        string temp_sid,temp_name,temp_cid,temp_sex;float m,e,c,z,p;
        for(int j=0;j<number;j++)       //冒泡排序
            for(int i=number-1;i>0;i--)
                if(stud[i].English<stud[i-1].English)
                  {//当第 i 个学生的大学英语分数小于第 i-1 个学生的大学英语分数时，将两个学生位置交换
                    temp_sid=stud[i].studentId; stud[i].studentId=stud[i-1].studentId; stud[i-1].studentId=temp_sid;
                    temp_name=stud[i].name;    stud[i].name=stud[i-1].name;    stud[i-1].name=temp_name;
                    temp_cid=stud[i].classId; stud[i].classId=stud[i-1].classId;    stud[i-1].classId=temp_cid;
                    temp_sex=stud[i].sex; stud[i].sex=stud[i-1].sex; stud[i-1].sex=temp_sex;
                    m=stud[i].Math;      stud[i].Math=stud[i-1].Math;       stud[i-1].Math=m;
                    e=stud[i].English;    stud[i].English=stud[i-1].English;       stud[i-1].English=e;
                    c=stud[i].Cplusplus; stud[i].Cplusplus=stud[i-1].Cplusplus; stud[i-1].Cplusplus=c;
                    z=stud[i].sum;       stud[i].sum=stud[i-1].sum;       stud[i-1].sum=z;
                    p=stud[i].average;    stud[i].average=stud[i-1].average;    stud[i-1].average=p;
                  }
        cout<<"学生按大学英语分数排名:"<<endl;
        cout<<"学号"<<' '<<"姓名"<<' '<<"性别"<<' '<<"班级"<<' '<<"大学英语分数"<<endl;
        for(int i=0;i<number;i++)
        {
          cout<<stud[i].studentId<<"       "<<stud[i].name<<"       "
          <<stud[i].sex<<"       "<<stud[i].classId<<"       "
          <<stud[i].English<<endl;cout<<endl;
        }
        cout<<"--------------按英语分数排名结束-------------"<<endl<<endl;
        //----------------按英语分数排名结束------------------
        break;
    }
```

```
case 4:
{
    cout<<"*****************************************"<<endl<<endl;
    //----------------按 C++分数排名开始-------------------
    cout<<"--------------按 C++分数排名开始-------------"<<endl;
    string temp_sid,temp_name,temp_cid,temp_sex;float m,e,c,z,p;
    for(int j=0;j<number;j++)       //冒泡排序
        for(int i=number-1;i>0;i--)
            if(stud[i].Cplusplus<stud[i-1].Cplusplus)
              {//当第 i 个学生的 C++分数小于第 i-1 个学生的 C++分数时，将两个学生位置交换
                temp_sid=stud[i].studentId; stud[i].studentId=stud[i-1].studentId; stud[i-1].studentId=temp_sid;
                temp_name=stud[i].name;   stud[i].name=stud[i-1].name;   stud[i-1].name=temp_name;
                temp_cid=stud[i].classId; stud[i].classId=stud[i-1].classId;   stud[i-1].classId=temp_cid;
                temp_sex=stud[i].sex; stud[i].sex=stud[i-1].sex; stud[i-1].sex=temp_sex;
                m=stud[i].Math;     stud[i].Math=stud[i-1].Math;       stud[i-1].Math=m;
                e=stud[i].English;   stud[i].English=stud[i-1].English;       stud[i-1].English=e;
                c=stud[i].Cplusplus; stud[i].Cplusplus=stud[i-1].Cplusplus; stud[i-1].Cplusplus=c;
                z=stud[i].sum;       stud[i].sum=stud[i-1].sum;       stud[i-1].sum=z;
                p=stud[i].average;   stud[i].average=stud[i-1].average;   stud[i-1].average=p;
              }
    cout<<"学生按 C++分数排名:"<<endl;
    cout<<"学号"<<' '<<"姓名"<<' '<<"性别"<<' '<<"班级"<<' '<<"C++分数"<<endl;
    for(int i=0;i<number;i++)
    {
      cout<<stud[i].studentId<<"       "<<stud[i].name<<"       "
      <<stud[i].sex<<"       "<<stud[i].classId<<"       "
      <<stud[i].Cplusplus<<endl;cout<<endl;
    }
     cout<<"--------------按 C++分数排名结束-------------"<<endl<<endl;
     //----------------按 C++分数排名结束------------------
     break;
}
case 5:
{
     cout<<"*****************************************"<<endl<<endl;
     //----------------计算最高总分开始------------------
     cout<<"--------------计算最高总分开始-------------"<<endl;
     float max;
     max=stud[0].sum;
     for(int j=0;j<number;j++)
     {
          if(stud[j].sum>max)
               max=stud[j].sum;
     }
     cout<<"最高总分为:"<<max<<endl;
```

```
        cout<<"-------------计算最高总分结束-------------"<<endl<<endl;
        //---------------计算最高总分结束-----------------
        break;
    }
    case 6:
    {
        cout<<"*****************************************"<<endl<<endl;
        //---------------计算最低总分开始---------------------
        cout<<"-------------计算最低总分开始-------------"<<endl;
        float min;
        min=stud[0].sum;
        for(int j=0;j<number;j++)
        {
            if(stud[j].sum<min)
                min=stud[j].sum;
        }
        cout<<"最低总分为:"<<min<<endl;
        cout<<"-------------计算最低总分结束-------------"<<endl<<endl;
        //---------------计算最低总分结束-----------------
        break;
    }
    case 7:
    {
        cout<<"*****************************************"<<endl<<endl;
        //---------------按平均分排名开始----------------
        cout<<"-------------按平均分排名开始-------------"<<endl;
        string temp_sid,temp_name,temp_cid,temp_sex;float m,e,c,z,p;
        for(int j=0;j<number;j++)       //冒泡排序
           for(int i=number-1;i>0;i--)
              if(stud[i].average<stud[i-1].average)
                {//当第 i 个学生的平均分小于第 i-1 个学生的平均分时，将两个学生位置交换
                  temp_sid=stud[i].studentId; stud[i].studentId=stud[i-1].studentId; stud[i-1].studentId=temp_sid;
                  temp_name=stud[i].name;    stud[i].name=stud[i-1].name;    stud[i-1].name=temp_name;
                  temp_cid=stud[i].classId; stud[i].classId=stud[i-1].classId;    stud[i-1].classId=temp_cid;
                  temp_sex=stud[i].sex; stud[i].sex=stud[i-1].sex; stud[i-1].sex=temp_sex;
                  m=stud[i].Math;      stud[i].Math=stud[i-1].Math;       stud[i-1].Math=m;
                  e=stud[i].English;    stud[i].English=stud[i-1].English;       stud[i-1].English=e;
                  c=stud[i].Cplusplus; stud[i].Cplusplus=stud[i-1].Cplusplus; stud[i-1].Cplusplus=c;
                  z=stud[i].sum;       stud[i].sum=stud[i-1].sum;       stud[i-1].sum=z;
                  p=stud[i].average;    stud[i].average=stud[i-1].average;    stud[i-1].average=p;
                }
       cout<<"学生按平均分排名:"<<endl;
       cout<<"学号"<<' '<<"姓名"<<' '<<"性别"<<' '<<"班级"<<' '<<"平均分"<<endl;
       for(int i=0;i<number;i++)
       {
```

```
                cout<<stud[i].studentId<<"       "<<stud[i].name<<"       "
                <<stud[i].sex<<"       "<<stud[i].classId<<"       "
                <<stud[i].average<<endl;cout<<endl;
            }
            cout<<"--------------按平均分排名结束-------------"<<endl<<endl;
            //----------------按平均分排名结束------------------
            break;
        }
        case 8:
        {
            cout<<"****************************************"<<endl<<endl;
            cout<<"--------------计算及格率开始-------------"<<endl;
            int Math_count=0;int English_count=0;int C_count=0;float Math_rate,English_rate,C_rate;
            for(int i=0;i<number;i++)
            {//统计高数的及格人数
                if(stud[i].Math>=60)
                    Math_count++;
            }
            Math_rate=((float)Math_count/(float)number)*100;              //计算高数的及格率
            for(int j=0;j<number;j++)
            {//统计大学英语的及格人数
                if(stud[j].English>=60)
                    English_count++;
            }
            English_rate=((float)English_count/(float)number)*100;        //计算大学英语的及格率
            for(int k=0;k<number;k++)
            {//统计 C++的及格人数
                if(stud[k].Cplusplus>=60)
                    C_count++;
            }
            C_rate=((float)C_count/(float)number)*100;                    //计算 C++的及格率
            cout<<setiosflags(ios::fixed)<<"三科成绩的及格率分别为:"<<endl;
            cout<<setprecision(2)<<Math_rate<<"%,"
            <<setprecision(2)<<English_rate<<"%,"<<setprecision(2)<<C_rate<<"%"<<endl;
            cout<<"--------------计算及格率结束-------------"<<endl<<endl;
            break;
        }
        case 0:
            return 0;break;
        default:cout<<"\t\t            输入错误或返回主界面"<<'\n';break;
        }
        } // switch 语句结束
        system("pause");
}
```

程序运行结果如图 3.2.8 所示。

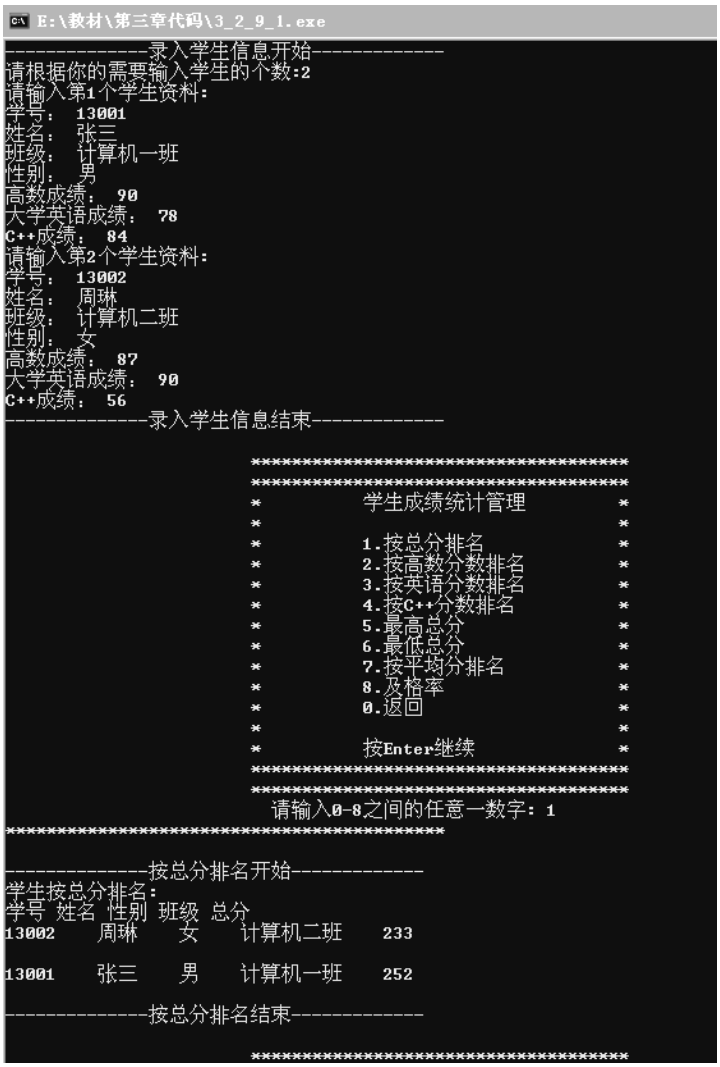

图 3.2.8　成绩统计程序运行结果

归纳小结

通过对本节的学习，我们基本掌握了数组的一般运用。主要包括如下要点:

1. 一维数组的定义、初始化与引用，一般使用循环结构去访问数组中的元素。

2. 二维数组的定义、初始化与引用，在使用二维数组时，要注意行与列所对应的实际存储的对象。在用循环嵌套语句访问二维数组的元素时，要注意数组下标的运用。

作业

一、选择题

1. 若有说明“int a[3][4];”，则 a 数组元素的非法引用是（　　）。

A. a[0][2*1]　　B. a[1][3]

C. a[4-2][0]　　D. a[0][4]

2．以下定义数组的语句中，正确的是（　　）。

A．int a(10);　　B．char str[];

C．int n=5; int a[4][n];　　D．#define SIZE 10;　char str 1[SIZE],str[SIZE +2];

3．在C++语言中，引用数组元素时，其数组下标的数据类型允许是（　　）。

A．整型常量　　B．整型表达式

C．整型常量或整型表达式　　D．任何类型的表达式

4．对以下说明语句的正确理解是（　　）。

int a[10]={6,7,8,9,10};

A．将5个初值依次赋给a[1]至a[5]

B．将5个初值依次赋给a[0]至a[4]

C．将5个初值依次赋给a[6]至a[10]

D．因为数组长度与初值的个数不相同，所以此语句不正确

5．以下二维数组c的定义形式正确的是（　　）。

A．int c[3][]　　B．float c[3,4]

C．double c[3][4]　　D．float c(3)(4)

6．若有说明“int a[ ][4]={0,0};”，则下面叙述不正确的是（　　）。

A．数组a的每个元素都可以得到初值0

B．二维数组a的第一维大小为1

C．当初值的个数能被第二维的常量表达式的值除尽时，所得商数就是第一维的大小

D．只有元素a[0][0]和a[0][1]可以得到初值，其余元素均得不到确定的初值

7．以下能对二维数组c进行正确初始化的语句是（　　）。

A．int c[3][]={{3},{3},{4}};　　B．int c[][3]={{3},{3},{4}};

C．int c[3][2]={{3},{3},{4},{5}};　　D．int c[][3]={{3},{},{3}};

8．已知对一维数组ns有如下说明：

int ns[10];

要求使ns 的所有元素都取值0。下面程序段不正确的是（　　）。

A．for(j=0;j<10;j++) ns[j]=0;

B．ns[0]=0;

　　for(j=1;j<10;j++) ns[j]=n[j－1];

C．for(j=1;j<=10;j++) ns[j]=0;

D．ns[0]=ns[1]=ns[2]=ns[3]=ns[4]=ns[5]=ns[6]=ns[7]= ns[8]=ns[9]=0;

9．有说明语句“int a[ ][4]={1,5,8,7,12,22,9,41,55,27};”，则数组a第一维的长度应该是（　　）。

A．2　　B．3　　C．4　　D．5

10．已知“int c[3][4];”，则对数组元素引用正确的是（　　）。

A．c[1][4]　　B．c[1.5][0]

C．c[1+0][0]　　D．以上引用都错误

11．下列语句中不正确的是(　　) 。

A．static int a[2][3]={1,2,3,4,5,6};　　B．static int a[2][3]={{1},{4,5}};

C．static int a[][3]={{1},{4}};　　D．static int a[][]={{1,2,3},{4,5,6}};

二、填空题

1．任何一个数组的数组元素都具有相同的名字和_____。

2．同一数组中，数组元素之间是通过_____来加以区分的。

3．若有说明“int a[][3]={1,2,3,4,5,6,7};”，则 a 数组第一维的大小是_____。

4．假定 int 类型变量占用两个字节，其有定义“int x[10]={0,2,4};”，则数组 x 在内存中所占字节数是_____。

5．已知数组 e 定义为“int e[ ][4]={{1,2,3,4},{5,6,7,8}};”，则 e 是一个_____行_____列的二维数组，总共有_____个元素，最大行下标是_____，最大列下标是_____，其首行各元素的值分别是_____。

6. 有定义语句“int aa[][3]={12,23,34,4,5,6,78,89,45};”，则 45 在数组 aa 中的行列坐标分别为_____。

7．定义如下变量和数组：

```
int k;
int a[3][3]={9,8,7,6,5,4,3,2,1};
```

则语句“for(k=0;k<3;k++) cout<<a[k][k];”的输出结果是_____。

8．有说明语句：

```
int x[ ][4]={{1},{2},{3}};
```

那么元素 x[1][1]的取值是_____。

9．要使 g 成为具有如下初始值的二维 int 型数组，则最简单的定义 g 的语句是_____。

```
1  0  0  0  0
1  2  0  0  0
1  2  3  0  0
1  2  3  4  0
1  2  3  4  5
```

10．下面的语句显示输出 6 行 6 列数组 h 的主对角线上的所有元素，请补充完整。

```
for(int i=_____;i<_____;_____)
cout<<_____<<' ';
```

11．下面的语句按行显式输出 5 行 3 列数组 m 的所有元素，请补充完整。

```
for(int i=_____;i<_____;_____)
    cout<<endl;
for(_____;_____;j++)
    cout<<_____<<' ';
```

三、程序阅读

1．写出下列程序的运行结果。

```
#include<iostream.h>
void main()
{
int i,a[3][3]={1,2,3,4,5,6,7,8,9};
for (i=0;i<3;i++)
cout<<a[i][2－i]<<"   ";
}
```

2．阅读程序，说明其功能及输出的结果。

```
#include<iostream.h>
void main()
{
   int k;
   char x,a[10]={'a','b','c','d','e','f','g','h','i','j'};
   for(k=0;k<5;k++)
   {
      x=a[k];
      a[k]=a[9 －k];
      a[9-k]=x;
   }
   for(k=0;k<10;k++)
      cout<<a[k];
   cout<<endl;
}
```

3．阅读程序，说明其功能及输出的结果。

```
#include<iostream.h>
void main()
{
      int j,k;
      int str[ ]={2,－4,5,15,19, －11,16,7};
      for(j=0,k=j;j<8;j++)
      if(str[j]>str[k])
      k=j;
      cout<<k;
}
```

4．阅读程序，说明其功能及输出的结果。

```
#include<iostream>
using namespace std;
int main()
{
      int i;
      int a[3][3]={1,2,3,4,5,6,7,8,9};
      for(i=0;i<3;i++)
      cout<<a[2-i][i];
      return 0;
}
```

四、编程题

1．有一个 3×4 的矩阵，要求编程并求出其中值最大的元素的值，以及其所在的行号和列号。

2．编程实现功能：从键盘上输入若干个学生的成绩，当输入负数时表示输入结束，计算学生的平均成绩，并输出低于平均分的学生成绩。

3．请用二维数组描述班级座位表，其中行标代表座位号码，列标代表学生姓名，并输出结果。

专业术语

Member	成员
Constructor	结构体
Array	数组
Array Index	数组下标
Character Set	字符集
Definition	定义
Initialization	初始化
Multidimensional Arrays	多维数组
One-Dimensional Arrays	一维数组