C语言程序设计教学案例：数组的使用

一、 案例名称： 学生成绩统计与分析

二、 案例目标：

1. 掌握一维数组和二维数组的定义、初始化和使用方法。
2. 理解数组作为函数参数的传递方式。
3. 能够运用数组解决实际问题，例如数据存储、统计和分析。

三、 案例描述：

设计一个学生成绩统计与分析程序，可以录入多个学生的多门课程成绩，并实现以下功能：

1. **录入学生成绩：** 输入学生人数和课程门数，然后依次录入每个学生的每门课程成绩。
2. **计算每个学生的平均成绩：** 计算每个学生的平均成绩并输出。
3. **计算每门课程的平均成绩：** 计算每门课程的平均成绩并输出。
4. **查找最高分和最低分：** 查找所有学生成绩中的最高分和最低分，并输出对应的学生和课程信息。

四、 案例实现：

```
#include <stdio.h>

#define MAX_STUDENTS 50
#define MAX_COURSES 10

// 函数声明
void inputScores(float scores[][MAX_COURSES], int numStudents, int numCourses);
void calculateStudentAverages(float scores[][MAX_COURSES], int numStudents, int
numCourses);
void calculateCourseAverages(float scores[][MAX_COURSES], int numStudents, int
numCourses);
void findMaxMinScores(float scores[][MAX_COURSES], int numStudents, int
numCourses);

int main() {
    float scores[MAX_STUDENTS][MAX_COURSES];
    int numStudents, numCourses;

    printf("请输入学生人数（最多%d人）：", MAX_STUDENTS);
    scanf("%d", &numStudents);
    printf("请输入课程门数（最多%d门）：", MAX_COURSES);
    scanf("%d", &numCourses);

    inputScores(scores, numStudents, numCourses);
    calculateStudentAverages(scores, numStudents, numCourses);
    calculateCourseAverages(scores, numStudents, numCourses);
    findMaxMinScores(scores, numStudents, numCourses);

    return 0;
}

// 录入学生成绩
void inputScores(float scores[][MAX_COURSES], int numStudents, int numCourses) {
    for (int i = 0; i < numStudents; i++) {
```

```
        printf("请输入第%d个学生的%d门课程成绩：\n", i + 1, numCourses);
        for (int j = 0; j < numCourses; j++) {
            scanf("%f", &scores[i][j]);
        }
    }
}

// 计算每个学生的平均成绩
void calculateStudentAverages(float scores[][MAX_COURSES], int numStudents, int
numCourses) {
    printf("\n每个学生的平均成绩：\n");
    for (int i = 0; i < numStudents; i++) {
        float sum = 0;
        for (int j = 0; j < numCourses; j++) {
            sum += scores[i][j];
        }
        printf("第%d个学生的平均成绩：%.2f\n", i + 1, sum / numCourses);
    }
}

// 计算每门课程的平均成绩
void calculateCourseAverages(float scores[][MAX_COURSES], int numStudents, int
numCourses) {
    printf("\n每门课程的平均成绩：\n");
    for (int j = 0; j < numCourses; j++) {
        float sum = 0;
        for (int i = 0; i < numStudents; i++) {
            sum += scores[i][j];
        }
        printf("第%d门课程的平均成绩：%.2f\n", j + 1, sum / numStudents);
    }
}

// 查找最高分和最低分
void findMaxMinScores(float scores[][MAX_COURSES], int numStudents, int
numCourses) {
    float maxScore = scores[0][0], minScore = scores[0][0];
    int maxStudent = 0, maxCourse = 0, minStudent = 0, minCourse = 0;

    for (int i = 0; i < numStudents; i++) {
        for (int j = 0; j < numCourses; j++) {
            if (scores[i][j] > maxScore) {
                maxScore = scores[i][j];
                maxStudent = i + 1;
                maxCourse = j + 1;
            }
            if (scores[i][j] < minScore) {
                minScore = scores[i][j];
                minStudent = i + 1;
                minCourse = j + 1;
            }
        }
    }

    printf("\n最高分：%.2f（第%d个学生，第%d门课程）\n", maxScore, maxStudent,
maxCourse);
    printf("最低分：%.2f（第%d个学生，第%d门课程）\n", minScore, minStudent,
minCourse);
```

```
}
```

五、 案例解析：

1. **二维数组定义：**
   - 使用 `float scores[MAX_STUDENTS][MAX_COURSES]` 定义一个二维数组，用于存储学生成绩。
   - `MAX_STUDENTS` 和 `MAX_COURSES` 是宏定义，用于限制数组大小。
2. **数组作为函数参数：**
   - 将二维数组 `scores` 作为参数传递给各个函数，实现数据共享。
   - 在函数声明和定义中，需要指定数组的第二维大小。
3. **数组遍历：**
   - 使用嵌套 `for` 循环遍历二维数组，访问每个学生的每门课程成绩。
4. **数据统计：**
   - 使用累加器 `sum` 计算每个学生的总成绩和每门课程的总成绩。
   - 使用比较运算符查找最高分和最低分。

六、 案例拓展：

1. **增加功能：** 可以增加按学生或课程排序、统计成绩分布等功能。
2. **文件存储：** 可以将学生成绩保存到文件中，实现数据的持久化存储。
3. **图形界面：** 可以使用图形库（如EasyX）为学生成绩统计与分析程序添加图形界面，提升用户体验。

七、 教学建议：

1. 在讲解案例之前，先讲解一维数组和二维数组的定义、初始化和使用方法。
2. 引导学生分析案例需求，思考如何使用数组存储和处理学生成绩数据。
3. 鼓励学生尝试修改代码，例如增加功能、改进代码结构等，加深对数组的理解。
4. 可以布置类似的编程练习，例如编写一个“商品库存管理系统”程序，巩固学生对数组的掌握。

八、 总结：

本案例通过一个学生成绩统计与分析程序，帮助学生理解数组的使用方法，并能够运用数组解决实际问题。通过案例拓展和教学建议，可以进一步激发学生的学习兴趣，提高学生的编程能力。