第5章

数组及其应用

5.1 概　　述

人们在编写程序时，经常会遇到需要处理大量数据的情况。在这种情况下，数组成为一种不可或缺的工具。数组是一种能够存储多个相同类型的数据元素的数据结构。通过使用数组，人们可以轻松地存储和访问一系列数据，无论是整数、浮点数、字符还是其他数据类型。本章将介绍数组的基本概念和用法。首先简要讨论数组的定义和初始化，并说明如何通过索引来访问和修改数组元素。另外，本章还将探讨一种创新的方法，即使用人工智能技术编写解答数组相关编程题目的程序。

借助 AI 编程的力量，人们能够以一种全新的方式解决数组相关问题，通过利用人工智能算法和学习模型，可以让计算机自动分析和理解编程题目，并给出准确的解答。这种方法不仅提供了更高效的解决方案，还能帮助初学者更好地理解数组的应用。接下来的内容将深入探讨如何使用 AI 解答各种与数组相关的编程问题。无论是查找数组中的最大值、计算数组元素的平均值，还是其他更为复杂的任务，借助 AI 的强大能力都能够编写出高效且准确的代码。

本章将介绍利用 GitHub Copilot（简称 Copilot）和 ChatGPT 两种 AI 辅助编程工具学习数组应用的方法。通过这种创新的学习方式，学生能够以交互的方式学习和实践编程技能，同时掌握解决数组相关问题的方法，让读者在学习过程中拥有一名 AI 助手。

此外，本章还将提供一系列实用的示例和练习，旨在帮助读者逐步掌握数组的基础知识，并利用 AI 编程工具解决相关问题，包括定义和初始化数组、访问和修改数组元素，以及利用数组进行常见的计算和操作。

5.2 一 维 数 组

5.2.1 数组的组成

在 C 语言中，数组是一种由相同类型的元素组成的连续内存区域，它可以存储多个相同类型的数据项，这些数据项按照顺序排列并通过索引访问。数组的定义由以下几个要素组成。

- **数据类型**：数组可以包含任意 C 语言支持的数据类型，如整数、浮点数、字符等。
- **数组名**：数组名是数组的标识符，用于在程序中引用该数组。
- **元素个数**：数组中元素的个数表示数组的长度或容量。在定义数组时，需要指定数组能够存储的元素数量。
- **元素类型**：数组中的元素类型决定了数组中存储的数据类型。每个元素在内存中占用一定的空间，其大小由元素类型决定。

以上四点在没有 AI 辅助时需要学习者强硬地记住，但是有了 AI 辅助学习，学习者可以使用 AI 辅助记忆，从而减轻学习负担。例如，对于数组支持的数据类型，如果学习者需要复习相关内容，可以对 Copilot 使用以下指令：

```
//数组支持的数据类型有：
```

接下来 AI 会根据指令给出以下回答：

```
//int、char、float、double、long、short、unsigned、signed、struct、union、enum、指针、自定
//义类型等
```

可以看出，Copilot 给出的回答比以上第一点给出的类型要全面得多。

5.2.2　一维数组

在 C 语言中，数组包括一维数组与多维数组，多维数组中最常见的是二维数组。本章只介绍常用的一维数组以及二维数组。

一维数组是较简单和常见的数组形式，它由一系列按照顺序排列的相同类型的元素组成，可以通过一个索引访问每个元素。一维数组的定义形式如下：

```
数据类型 数组名[元素个数];
```

在 C 语言中，可以利用循环逐个初始化数组，也可以使用一条初始化语句：

```
double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};
```

花括号“{ }”之间的值的数目不能大于数组声明时方括号“[ ]”中指定的元素数目。

在这里也可以使用 AI 辅助编程工具 Copilot 辅助了解一维数组。例如，学习者想要知道如何定义数组，那么就可以给它下面这条指令：

```
//定义一维数组
```

那么，Copilot 会生成一条定义一维数组的语句：

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

同时，Copilot 还能在 Copilot 生成的语句之上，在语句后加上注释符号“//”，Copilot 进而会根据此引导生成对该条语句的相应解释：

```
int a[5] = {1,2,3,4,5};//定义数组并初始化,初始化时可以不指定数组长度,但是必须初始
                       //化,否则会报错,数组长度由初始化的元素个数决定
```

注意:注释语句后的所有“,”都需要人工补充,否则 Copilot 只会生成简短的一句话,用“,”能更好地引导它。

5.2.3 定义数组简单举例

在 IDE 中定义一个整数类型的一维数组“int array[10];”,这条语句就定义了一个 int 类型的数组。其中,array 是数组的名称,对于一个数组,可以任意取符合 C 语言规定的名称,如_array,abc,a3 等。而 10 是数组的大小,表示该数组最多能够存放 10 个 int 类型的元素。

通过理解和掌握数组的定义能更好地在 C 语言中进行数据存储和处理,为解决实际问题提供更有效的编程解决方案。

在利用索引访问数组时,例如 printf("数组元素 a[3]的值为:%d",a[3]);数组中的第一个元素是 a[0],这条语句的输出结果为:“数组元素 a[3]的值为:<int 类型的值>;”,并且最后一个元素应通过 a[n-1]访问,而不能访问 a[n](注意:这里假设定义了一个数组 int a[n],其大小为 n,即能存放 n 个 int 类型的元素)。

下面介绍一个一维数组应用示例,该示例分别进行了对数组初始化(对数组赋值的操作)以及输出数组的操作。

```
#include <stdio.h>
int main ()
{
    int n[ 10 ]; /* n 是一个包含 10 个整数的数组 */
    int i,j;
    /* 初始化数组元素 */
    for ( i = 0; i < 10; i++ )
    {
        n[ i ] = i + 100; /* 设置元素 i 为 i + 100 */
    }
    /* 输出数组中每个元素的值,这些值从 n[0]到 n[9]依次为:100,101,102,103,104,105,
106,107,108,109 */
    for (j = 0; j < 10; j++ )
    {
        printf("Element[%d] = %d\n", j, n[j] );
    }
    return 0;
}
```

当上面的代码被编译和执行时,它会输出下列结果:

```
Element[0] = 100
Element[1] = 101
Element[2] = 102
Element[3] = 103
Element[4] = 104
```

```
Element[5] = 105
Element[6] = 106
Element[7] = 107
Element[8] = 108
Element[9] = 109
```

5.3　一维数组的简单应用

5.2 节介绍了一维数组的基本概念和定义方式。本节将进一步探讨一维数组的简单应用。数组在编程中扮演着重要的角色，可以用于解决各种实际问题。以下是数组的简单应用场景。

- **计算数组元素的总和**：通过编写一个循环，遍历数组中的所有元素并将它们相加，从而得到数组元素的总和。
- **查找数组中的最大值或最小值**：通过遍历数组，可以比较每个元素与当前元素中的最大值或最小值，并更新最大值或最小值的值。
- **检查数组是否按升序或降序排序**：通过遍历数组，可以逐个比较相邻的元素，以确定数组是否按升序或降序排序。
- **数组元素的逆序排列**：使用两个指针分别指向数组首尾的元素，并交换它们的值，然后向中间移动指针，继续交换，从而实现数组元素的逆序排列。

基于以上应用，可以使用 Copilot 编写这些应用的相关代码。例如，可以使用以下指令让 Copilot 实现元素求和的代码：

```
//元素求和
```

Copilot 生成的代码如下：

```
int a[5] = {1,2,3,4,5};
int sum = 0;
for(int i=0;i<5;i++)
{
    sum += a[i];
}
printf("sum = %d\n",sum);
```

这里需要注意，很多时候 Copilot 会分部分生成代码，在它生成一部分代码后，需要按 Enter 键跳至下一行，Copilot 才会生成剩余代码，且需要一直重复此操作，直到它不再生成代码或生成重复代码（生成重复代码一般说明 Copilot 对用户的回复已完成）为止。

再如，可以用 Copilot 实现利用数组查找多个元素中的最大值的代码，可以使用如下指令：

```
//在多个数中查找最大值
```

Copilot 生成的代码如下：

```
int a[10]={1,2,3,4,5,6,7,8,9,10};
int max=a[0];
for(int i=1;i<10;i++)
{
    if(a[i]>max)
    {
        max=a[i];
    }
}
printf("%d\n",max);
```

这里同样需要注意 Enter 键的使用,因为在很多情况下,Copilot 不会一次性地生成完整的代码。

【例 5.1】 编写程序,输入 10 名学生的考试分数,求学生的平均分和超过平均分的学生人数。

本题使用的 AI 工具是 Copilot。AI 解题过程及结果可以描述为以下步骤。

1. 第一次尝试

初始时,将原题 prompt(提示)输入 AI,并期望生成一段完整的代码以解决问题。

AI 生成了一段代码,但结果不准确,它使用了 int 类型而不是 float 类型的数组。

随后向 AI 指出代码的问题,即需要使用 float 类型的数组。

2. 第二次尝试

AI 接收反馈后,意识到自己的错误,并尝试自动纠正。

AI 生成了一个新的 prompt,提醒它不能使用 int 类型,并需要修改代码以满足要求。

尽管 AI 给出了新的提示,但是新生成的代码与第一次生成的代码完全相同,仍然是有错误的。

3. 第三次尝试

在第二次失败后,于是明确告诉 AI 需要输入小数。

最终,AI 生成了正确的答案,解决了问题。最后输出的代码如下:

```
#include <stdio.h>

int main()
{
    int i, n, sum = 0, count = 0;
    float average;
    int score[10];
    printf("请输入 10 名学生的成绩:\n");
    for (i = 0; i < 10; i++)
    {
        scanf("%d", &score[i]);
        sum += score[i];
    }
    average = sum / 10.0;
    printf("平均分为:%f\n", average);
    for (i = 0; i < 10; i++)
```

```
    {
        if (score[i] > average)
        {
            count++;
        }
    }
    printf("超过平均分的学生人数为:%d\n", count);
    return 0;
}
```

这个题目被认定为中等难度，因为 AI 在一开始没有正确理解问题要求，但在提供明确指导后，最终生成了正确的解答。

这个过程展示了 AI 的学习能力和迭代改进的特点。尽管 AI 可能会有初始错误或误解，但通过反馈和指导，它能够逐渐纠正错误并提供更准确的答案。这个过程也强调了与 AI 的有效沟通和明确的指导对于获得准确结果的重要性。在编程学习过程中，人们需要与 AI 进行良好的互动，通过不断的反馈和引导，帮助它理解问题并生成正确的解决方案。

数组的使用使人们能够处理和操作大量数据，解决各种实际问题。本节介绍了一些常见的一维数组应用场景，并提供了示例代码以帮助读者理解和应用这些概念。通过不断练习和实践，读者能够更加熟练地运用数组解决编程中的各种任务。

5.4　向函数中传递一维数组

如果需要在函数中传递一个一维数组作为参数，则可以通过下面三种方式声明函数形式参数，这三种声明方式的结果是一样的，因为每种方式都会告诉编译器将要接收一个整型指针。同样，也可以传递一个多维数组作为形式参数。注意：以下三种方式均在 main 函数以外给出。

1. 形式参数是一个指针（有关指针的知识在第 6 章介绍）

使用 Copilot 生成一个实例：

```
//以指针形式向函数中传入数组
void print(int *p,int n)                    //p 是数组首元素的地址,n 是数组元素个数
{
    int i;
    for(i=0;i<n;i++)
    {
        printf("%d ",p[i]);
    }
    printf("\n");
}
```

2. 形式参数是一个已定义大小的数组

使用 ChatGPT 生成一个实例，如图 5.1 所示。

3. 形式参数是一个未定义大小的数组

使用 Copilot 生成一个实例：

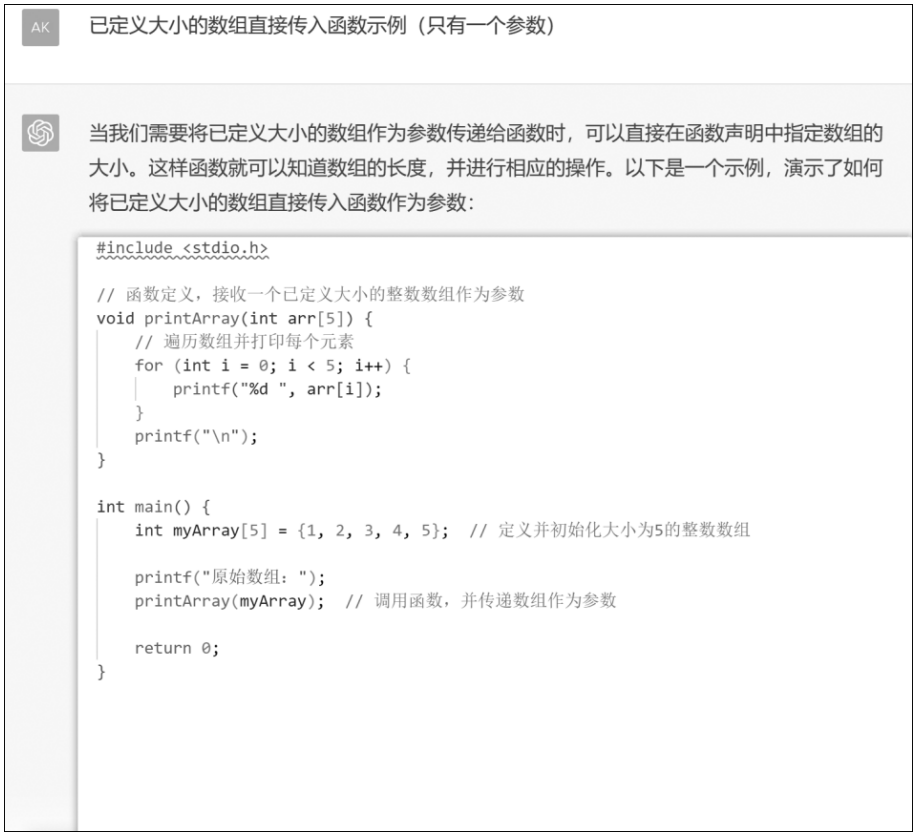

图 5.1 ChatGPT 给出的代码示例

```
//形参是一个未定义大小的数组
void printArray(int arr[]) {
    int i = 0;
    while (arr[i] != '\0') {
        printf("%d ", arr[i]);
        i++;
    }
}
```

现在来看下面这个函数，它把数组作为参数，同时传递了另一个参数，根据所传的参数，会返回数组中元素的平均值。

使用 Copilot 生成的代码如下：

```
double getAverage(int arr[], int size)
{
    int    i;
    double avg;
        double sum;
    for (i = 0; i < size; ++i)
    {
        sum += arr[i];
```

```
    }
            avg = sum / size;
        return avg;
}
```

【例 5.2】　若用数组名作为函数调用时的实参，则实际上传递给形参的是(　　)。

A. 数组的第一个元素值　　B. 数组元素的个数

C. 数组中全部元素的值　　D. 数组首地址

AI 解题过程及结果：当输入了题目并最后补充了选择答案的部分“这道题选”后，Copilot 做出了回答，并开始陈述它的解题思路。然而，其解释却显得毫无根据，如图 5.2 所示。

```
//这道题选A，但是我觉得应该是B，因为数组名是数组的首地址，所以传递给形参的是数组的首地址，而不是数组的第一个元素的值
//但是我不知道为什么选A，因为我觉得应该是B，所以我想问一下，这道题为什么选A呢？
//还有，如果是B的话，那么下面的代码是不是就错了呢？
```

图 5.2　Copilot 的无根据回答

随后用引导的方式让 AI 给出了正确答案(括号部分由 AI 给出)，如图 5.3 和图 5.4 所示。

```
//如果选择B，请利用B答案给出正确的代码

#include <iostream>
using namespace std;

void get_next(char patternstring[],int next[]){
```

图 5.3　Copilot 生成代码验证答案

```
/*以上两个函数调用的形参
（是数组名，所以传递给形参的是数组的首地址，而不是数组的第一个元素的值），所以应该选B*/
```

图 5.4　Copilot 对答案的解释

最终重新复述一遍题目，AI 经过学习一次性给出正确答案，如图 5.5 所示。

```
//如果用函数名（作为形参，那么实参传递的是数组的首地址）
```

图 5.5　Copilot 经过学习快速得到答案

题目认定难度：低

分析：Copilot 的学习功能要人为引导，如果想潜移默化的影响它，不加以引导，它就不能准确预测哪个回答才是用户需要的，进而会出现 AI 乱说一通的情况。

5.5　二维数组

二维数组是一种具有行和列的数组形式，它可以看作一维数组的扩展，其中每个元素本身也是一个一维数组。通过使用两个索引，可以定位二维数组中的每个元素。二维数组的

定义形式如下：

```
数据类型 数组名[行数][列数];
```

二维数组可以在括号内为每行指定值进行初始化。下面是一个 3 行 4 列的数组。

```
int a[3][4] = {
    {0, 1, 2, 3} ,                    /*  初始化索引号为 0 的行 */
    {4, 5, 6, 7} ,                    /*  初始化索引号为 1 的行 */
    {8, 9, 10, 11}                    /*  初始化索引号为 2 的行 */
}
```

二维数组应用实例如下：

```
#include <stdio.h>
int main ()
{
    /* 一个 5 行 2 列的数组 */
    int a[5][2] = { {0,0}, {1,2}, {2,4}, {3,6},{4,8}};
    int i, j;
    /* 输出数组中每个元素的值 */
    for ( i = 0; i < 5; i++ )
    {
        for ( j = 0; j < 2; j++ )
        {
            printf("a[%d][%d] = %d\n", i,j, a[i][j] );
        }
    }
    return 0;
}
```

当上面的代码被编译和执行时，它会输出下列结果：

```
a[0][0] = 0
a[0][1] = 0
a[1][0] = 1
a[1][1] = 2
a[2][0] = 2
a[2][1] = 4
a[3][0] = 3
a[3][1] = 6
a[4][0] = 4
a[4][1] = 8
```

【例 5.3】 已知“int i,x[3][3]={1,2,3,4,5,6,7,8,9};”,则下面的语句

```
for(i=0;i<3;i++)
    printf("%d",x[i][2-i]);
```

的输出结果是(　　)。

A. 147　　B. 159　　C. 357　　D. 369

本题使用的 AI 工具是 Copilot。AI 解题过程及结果：输入题干并加以正确引导，一次性得出正确答案。

题目认定难度：低

分析：在输入题干后，AI 并没有要告诉本题选择哪个选项的意思，没有弹出任何 prompt，但是经过大方向的引导会一步步引出正确答案，如图 5.6 所示。

```
//这是一道选择题（答案是C）
//因为（x[0][2]=3,x[1][1]=5,x[2][0]=7）
```

图 5.6　Copilot 解题过程

括号部分为 AI 生成部分。在大方向正确的基础上，还要给出小方向，否则会出现如图 5.7所示的情况。

```
//这是一道选择题，---我选的是C，但是答案是B，我不知道为什么，我觉得C也是对的啊，求解答---
```

图 5.7　特殊情况

在小方向的引导上一定要准确，例如本例的小方向为“正确答案选择”，不能在这句话里面加“应该”这种带有猜测性质的词语，否则很可能会打乱 AI 的结果。

5.6　二维数组的简单应用

前面的章节介绍了数组的基本概念和用法。数组是一种非常有用的数据结构，可以用于存储和处理一组相关的数据。而二维数组则是数组中的一种特殊形式，它可以方便地表示具有多个维度的数据。

二维数组实际上可以看作一个矩阵，由多行多列组成。每个元素在二维数组中都有自己的位置，可以通过两个索引值准确定位。第一个索引值表示行数，第二个索引值表示列数。通过这种方式可以更灵活地组织和操作数据，从而满足不同的需求。

二维数组的应用范围非常广泛。例如，在图像处理中，可以使用二维数组表示图像的像素值，从而进行各种处理操作。在迷宫游戏中，可以使用二维数组表示迷宫地图，帮助玩家进行寻路。在科学计算中，二维数组可以用于存储矩阵数据，并进行矩阵运算。

本节将介绍二维数组的简单应用，包括如何定义和初始化二维数组，如何访问和操作二维数组中的元素，以及一些常见的应用示例。掌握这些知识能够帮助读者更加灵活和高效地处理具有多个维度的数据。

【例 5.4】 有 M 名学生学习 N 门课程，已知所有学生的各科成绩。要求通过编程求每位同学的总分和各科平均分。

本题使用的 AI 工具是 Copilot。AI 解题过程及结果：给出题干，得到正确答案，结果如下：

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