# 实验三 指针与引用

## 题目 2 班级课程成绩计算

### 实验描述：

一个班有 5 名学生，每个学生修了五门课，1）求每个学生的平均成绩，并输出每个学生的学号，每门课程的成绩及平均值。2）求某门课程的平均分；

实现要求：

1、分别编写2 个函数实现以上2 个要求。

2、第1 个函数用数组名作参数。第2 个函数用指针用参数，并在函数体内用指针对数组操作。

### 实验代码：

/\*

 \* @Author: 王贤义

 \* @FilePath: \code\C+++C\c++\3\2score.cpp

 \* @Description: 一个班有 5 名学生，每个学生修了五门课，1）求每个学生的平均成绩，并输出每个学生的学号，每门课程的成绩及平均值。2）求某门课程的平均分；

 \*/

#include <iostream>

using namespace std;

/\*\*

 \* @description: 计算每个学生的平均分

 \* @param {int} sco 二维数组，表示每个学生的每门课程的成绩

 \* @param {int} avsco 一维数组，表示每个学生的平均分

 \* @return {\*}

 \*/

void averscore(int sco[][5], int avsco[]) // 二维数组的行数可以省略,但是列数不能省略

{

    for (int i = 0; i < 5; i++) // 遍历每个学生

    {

        int sum = 0;

        for (int j = 0; j < 5; j++) // 遍历每个学生的每门课程

        {

            sum += sco[i][j]; // 求每个学生的总分

        }

        avsco[i] = sum / 5; // 求每个学生的平均分

    }

}

/\*\*

 \* @description: 计算每门课程的平均分

 \* @param {int} n 课程的编号

 \* @return {\*} 返回课程的平均分

 \*/

int avercourse(int (\*psco)[5], int n) // 二维数组的行数不能省略,但是列数可以省略

{

    int sum = 0;

    for (int i = 0; i < 5; i++) // 遍历每个学生

    {

        sum += psco[i][n - 1]; // 求每门课程的总分

    }

    return sum / 5; // 求每门课程的平均分

}

int main()

{

    int num[5] = {1, 2, 3, 4, 5};

    int sco[5][5] = {0};

    int avsco[5] = {0};

    for (int i = 0; i < 5; i++)

    {

        for (int j = 0; j < 5; j++)

        {

            cout << "Student " << num[i] << ", Course " << j + 1 << ": ";

            cin >> sco[i][j];

        }

    }

    averscore(sco, avsco);      // 求每个学生的平均分

    for (int i = 0; i < 5; i++) // 输出每个学生的平均分

    {

        cout << "Student " << num[i] << "'s scores: ";

        for (int j = 0; j < 5; j++)

        {

            cout << sco[i][j] << " ";

        }

        cout << "Average: " << avsco[i] << endl;

    }

    int t;

    cout << "Which course do you want to know the average score of? ";

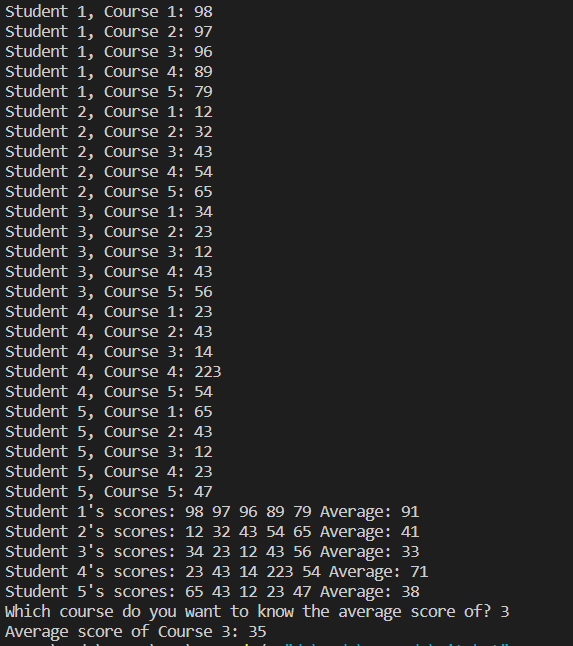
    cin >> t;

    cout << "Average score of Course " << t << ": " << avercourse(sco, t) << endl; // 求课程的平均分

    return 0;

}

### 实验结果：



### 思考与扩展：

1. 二维数组作为函数形参int avercourse( int（\*psco）[5], n )中为什么要指定第二维的

大小？

当你将二维数组作为函数形参传递时，指定第二维的大小可以使函数在处理数组时知道每行有多少元素。这是因为在C语言中，多维数组是以连续的内存块来存储的，指定第二维的大小有助于计算每个元素的内存地址。

1. 在函数体中用指针操作二维数组比如：\*（\*（psco +i）+j）就代表数组第i 行j 列

的元素即psco[i][j]。

以下是修改后的代码：

#include <iostream>

using namespace std;

void averscore(int (\*psco)[5], int avsco[])

{

    for (int i = 0; i < 5; i++)

    {

        int sum = 0;

        for (int j = 0; j < 5; j++)

        {

            sum += \*(\*(psco + i) + j);

        }

        avsco[i] = sum / 5;

    }

}

int avercourse(int (\*psco)[5], int n)

{

    int sum = 0;

    for (int i = 0; i < 5; i++)

    {

        sum += \*(\*(psco + i) + n - 1);

    }

    return sum / 5;

}

int main()

{

    int num[5] = {1, 2, 3, 4, 5};

    int sco[5][5] = {0};

    int avsco[5] = {0};

    for (int i = 0; i < 5; i++)

    {

        for (int j = 0; j < 5; j++)

        {

            cout << "Student " << num[i] << ", Course " << j + 1 << ": ";

            cin >> \*(\*(sco + i) + j);

        }

    }

    averscore(sco, avsco);

    for (int i = 0; i < 5; i++)

    {

        cout << "Student " << num[i] << "'s scores: ";

        for (int j = 0; j < 5; j++)

        {

            cout << \*(\*(sco + i) + j) << " ";

        }

        cout << "Average: " << avsco[i] << endl;

    }

    int t;

    cout << "Which course do you want to know the average score of? ";

    cin >> t;

    cout << "Average score of Course " << t << ": " << avercourse(sco, t) << endl;

    return 0;

}

## 题目 3 排序查找

### 实验描述：

用选择法对 15 个整数按从大到小的顺序排序。15 个整数用cin 输入。排序完成后输入一个数，要求用折半查找法找出该数是数组中第几个元素的值。如果该数不在数组中，则输出“无此数”。

要求：

1、把排序算法写成函数形式，用指针变量作为排序函数的参数，在排序算法中用指针法访问数组元素。

2、把查找算法写成函数形式，用指针变量作为查找函数的参数，在查找算法中用指针法访问数组元素。

### 实验代码：

/\*

 \* @Author: 王贤义

 \* @FilePath: \code\C+++C\c++\3\3sortfind.cpp

 \* @Description: 选择排序和折半查找，其中使用指针法访问数组

 \*/

#include <iostream>

using namespace std;

/\*\*

 \* @description: 选择排序，使用指针法访问数组

 \* @param {int} \*p 数组指针

 \* @param {int} n  数组长度

 \* @return {\*}

 \*/

void sort(int \*p, int n)

{

    int i, j, k, t;

    for (i = 0; i < n - 1; i++)

    {

        k = i;

        for (j = i + 1; j < n; j++)

        {

            if (\*(p + j) > \*(p + k))

            {

                k = j;

            }

        }

        t = \*(p + i);

        \*(p + i) = \*(p + k);

        \*(p + k) = t;

    }

}

/\*\*

 \* @description: 折半查找

 \* @param {int} \*p 数组指针

 \* @param {int} n 数组长度

 \* @param {int} x 要查找的数

 \* @return {\*}

 \*/

int search(int \*p, int n, int x)

{

    int low = 0, high = n - 1, mid;

    while (low <= high)

    {

        mid = (low + high) / 2;

        if (\*(p + mid) == x) // 找到

        {

            return mid + 1;

        }

        else if (\*(p + mid) > x) // 在左半部分

        {

            low = mid + 1;

        }

        else // 在右半部分

        {

            high = mid - 1;

        }

    }

    return -1;

}

int main()

{

    int a[15], i, x, result;

    for (i = 0; i < 15; i++)

    {

        cin >> a[i];

    }

    sort(a, 15); // 选择排序

    cout << "排序后的数组为：" << endl;

    for (i = 0; i < 15; i++)

    {

        cout << a[i] << " ";

    }

    cout << endl

         << "请输入要查找的数：";

    cin >> x;

    result = search(a, 15, x); // 二分查找

    if (result == -1)

    {

        cout << "无此数" << endl;

    }

    else

    {

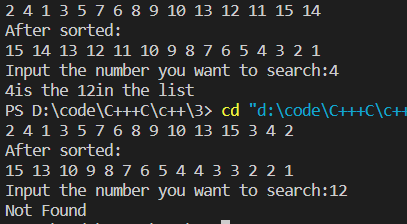
        cout << x << "是数组中第" << result << "个元素的值" << endl;

    }

    return 0;

}

### 实验结果：



### 思考与扩展：

1. 在调用函数时实参为什么只需给出数组名？

在调用函数时，实参只需给出数组名，是因为数组名可以被视为数组的指针。在C和C++中，数组名实际上表示了数组的第一个元素的地址。

1. 指针函数和函数指针有什么区别？

指针函数（Pointer to a Function）：指针函数是一个返回指针的函数。也就是说，指针函数的返回类型是一个指针类型。这意味着该函数执行后将返回一个指针，可以用于指向某个数据类型的变量或对象。指针函数可以用于实现动态分配内存、返回动态创建的对象等。

int\* createArray(int size) {

    int\* arr = (int\*)malloc(size \* sizeof(int));

    return arr;

}

函数指针（Function Pointer）：函数指针是指向函数的指针变量。函数指针可以用于存储函数的地址，从而使我们能够通过函数指针调用相应的函数。函数指针的声明需要指定函数的参数列表和返回类型。

int (\*sum)(int, int);

在上述示例中，sum是一个函数指针，它可以指向一个接受两个整型参数并返回整型结果的函数。我们可以将适当的函数的地址赋值给该函数指针，并通过函数指针来调用该函数。

int add(int a, int b) {

    return a + b;

}

// 将add函数的地址赋值给sum函数指针

sum = &add;

// 通过函数指针调用函数

int result = (\*sum)(2, 3);