**暨南大学本科实验报告专用纸**

课程名称 高级语言程序设计 成绩评定

实验项目名称 编程实验十二 指导教师 张鑫源

实验项目编号 108 实验项目类型 c语言编程 实验地点 家

学生姓名 张晓彤 学号 2019051119

学院 智能科学与工程学院 系 专业 信息安全 实验时间 2020 年 5 月 27 日 午～ 月 日 午 温度 ℃湿度

1. **实验目的**

指针的使用：以指针的方式实现对二维数组的冒泡排序。

示例:对于整型的二维数组a,设计函数sort实现对数组a中的元素进行排序。

1. **实验内容和要求**
   1. 设计合理的输出展示实验结果;
   2. 数组a的维度固定，排序函数sort的参数为指针变量(指针的类型为int \*)，从大到小排序;
   3. 数组a的维度由用户指定，a由随机数填充;
   4. 排序函数sort的参数为数组指针;
   5. 排序函数sort的参数为指针数组;
   6. 函数sort的排序规则由用户指定;
2. **主要仪器设备**

**仪器：**计算机

**实验环境：Visual Studio 2019**

1. **源程序**

1.数组a维度固定，排序函数sort的参数为指针变量，从大到小排序：

#include <stdio.h>

void sort1(int\* p);

int main()

{

int a[3][2] = {5,9,20,54,1,0};

printf("原数组a：\n");

for (int i = 0; i < 3; i++)

{

for (int j = 0; j < 2; j++)

{

printf("%5d", a[i][j]);

}

printf("\n");

}

int\* p=&a[0][0];

sort1(p);

printf("排序后:");

for (int i = 0; i < 6; i++)

{

printf("%5d", \*(p + i));

}

return 0;

}

void sort1(int\* p)

{

int temp;

for (int i = 0; i < 5; ++i) //比较5轮

{

for (int j = 0; j < 5 - i; ++j) //每轮比较5-i次,

{

if (\*(p+j) < \*(p+j + 1))

{

temp = \*(p+j);

\*(p + j) = \*(p + j + 1);

\*(p + j + 1) =temp;

}

}

}

}

2. 数组a的维度由用户指定，a由随机数填充：

#include <stdio.h>

#include <stdlib.h>

int \*sort3\_max(int \*p[100], int row, int clo);

int \*sort2\_min(int (\*p)[100], int , int );

int main()

{

//用户输入

int row, clo;

printf("输入行列数：");

scanf\_s("%d%d", &row, &clo);

int num = row \* clo;

//分配空间，定义a

int\*\* a;

a = (int\*\*)malloc(row \* sizeof(int\*));

a[0] = (int\*)malloc(sizeof(int) \* row \* clo);

for (int i = 1; i < row; i++)

a[i] = a[i - 1] + clo;

//随机数填充a

printf("原数组a：\n");

int\* p[100];

int(\*q)[100] = new int[row][100];

for (int i = 0; i < row; i++)

{

p[i] = (int\*)malloc(sizeof(int) \* clo);

}

for (int i = 0; i < row; i++)

{

for (int j = 0; j < clo; j++)

{

a[i][j] = (int)rand() % 100;

(\*(q + i))[j] = a[i][j];//数组指针赋值

printf("%5d", a[i][j]);

}

printf("\n");

}

for (int i = 0; i < row; i++)//指针数组赋值

{

p[i] = a[i];

}

//排序规则

int n;

scan:printf("从大到小：1\n从小到大：2\n退出：0\n");

scanf\_s("%d", &n);

if (n == 1)

{

sort3\_max(p, row, clo);

printf("排序后：\n");

for (int i = 0; i < row; i++)

{

for (int j = 0; j < clo; j++)

{

printf("%5d", \*(p[i] + j));

}

printf("\n");

}

goto scan;

}

else if (n == 2)

{

sort2\_min(q, row, clo);

printf("排序后：\n");

for (int i = 0; i < row; i++)

{

for (int j = 0; j < clo; j++)

{

printf("%5d", (\*(q + i))[j]);

}

printf("\n");

}

goto scan;

}

else if (n==0)

{

}

else {

goto scan;

}

return 0;

}

int \*sort3\_max(int \*p[100], int row, int clo)

{

int temp;

int num = row \* clo;

for (int m = 0; m < (num - 1); m++)

{

for (int i = 0; i < row; i++)

{

for (int j = 0; j < clo; j++)

{

if (j == clo - 1 && i != row - 1)

{

if (((\*(p + i))[j]) < ((\*(p + (i + 1)))[0]))

{

temp = (\*(p + i))[j];

(\*(p + i))[j] = (\*(p + (i + 1)))[0];

(\*(p + (i + 1)))[0] = temp;

}

break;

}

else if (j < clo - 1)

{

if ((\*(p + i))[j] < ((\*(p + i))[j + 1]))

{

temp = (\*(p + i))[j];

(\*(p + i))[j] = (\*(p + i))[j + 1];

(\*(p + i))[j + 1] = temp;

}

}

}

}

}

return(p[100]);

}

int \*sort2\_min(int(\*p)[100], int row, int clo)

{

int temp;

int num = row \* clo;

for (int m = 0; m < (num - 1); m++)

{

for (int i = 0; i < row; i++)

{

for (int j = 0; j < clo; j++)

{

if (j == clo - 1 && i != row - 1)

{

if (((\*(p + i))[j]) > ((\*(p + (i + 1)))[0]))

{

temp = (\*(p + i))[j];

(\*(p + i))[j] = (\*(p + (i + 1)))[0];

(\*(p + (i + 1)))[0] = temp;

}

break;

}

else if (j < clo - 1)

{

if ((\*(p + i))[j] > ((\*(p + i))[j + 1]))

{

temp = (\*(p + i))[j];

(\*(p + i))[j] = (\*(p + i))[j + 1];

(\*(p + i))[j + 1] = temp;

}

}

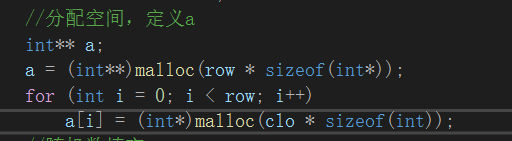
}

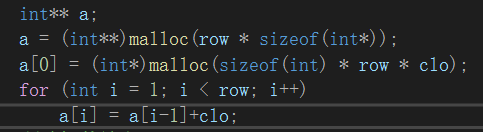
}

}

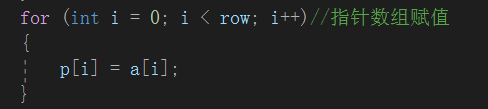
return (p[100]);

}

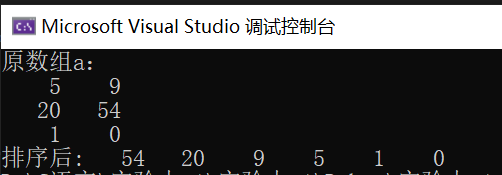
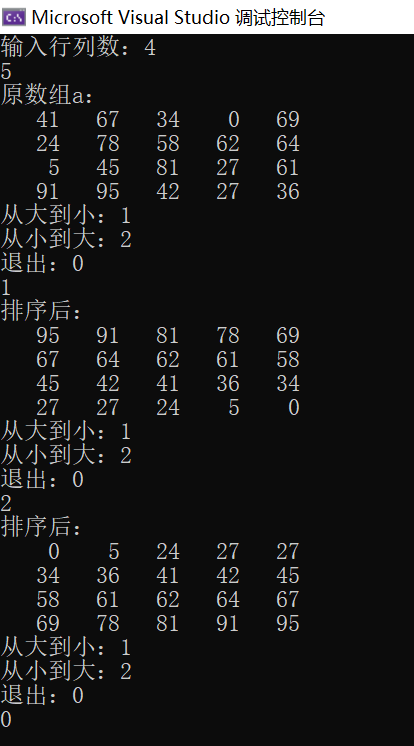
1. **实验步骤与调试**
   1. 数组a维度固定，排序函数sort的参数为指针变量，从大到小排序：
      1. 将**指针p的首地址设为数组a[0][0]的地址**，并传入函数sort1中。
      2. 因为数组a各元素的地址连续且递增，**\*（p+j）**为数组第j个元素的值，利用冒泡排序的方法，排序。
   2. 编写排序规则：设置两个选项，判断输入的数字，分别调用相应的函数，否则用goto回到输入命令的程序段。
   3. 排序后输出。
   4. 动态分配内存：
      1. 此方法分配内存发现**内存不连续**，对后面传递指针排序造成困难。
      2. **改进**分配方法：一次性分配所有空间，a的值初始化为m\* n的二维数组首地址, 且这块**内存连续**。



* 1. 因为从小到大和从大到小的排序方法差不多，为方便完成任务，将**从小到大**排序的函数参数设为**数组指针**，而**从大到小**排序的函数参数设为**指针数组**。
  2. 比较规则：（以从大到小为例）前一个数和后一个数比较，若前一个数较小则交换，判断当该数是**一行里的最后一个数**，且不**是最后一行**时则与**行数加一列数为0的数**比较，后跳出内层循环，另行数加一后继续，直至所有数都判断完。（从小到大排序则相反）
  3. 数组指针：
     1. **int(\*p)[]**是一个指向数组的指针，它其实还是**一个指针**，只不过是**指向数组**而已，将a[i][j]赋值于数组指针中： 
     2. 因为a的行列数不是一个常数，所以需要利用循环一个一个赋值
  4. 指针数组：
     1. **int \*q[]**是数组，而**数组的每一个元素都是指针**，用指针表示数组的一行数。



* + 1. 第i行第j列的表示方法：

1. **实验结果与分析**
   1. 数组a维度固定，排序函数sort的参数为指针变量，从大到小排序：
   2. 随机分配数组a结果：

**暨南大学本科实验报告专用纸(附页)**