山东大学 软件 学院

c++程序设计 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201800301 | 姓名： 李成 | | 班级： 18.4 |
| 实验题目：实验二 | | | |
| 实验学时：4 | | 实验日期：2019.4.11 | |
| 实验目的：  熟悉c++的数组  熟悉c++的指针、引用。 | | | |
| 硬件环境：  PC机 | | | |
| 软件环境：  Dev C++ | | | |
| 实验步骤与内容：  1、创建源程序文件myFunctions.cpp，在其中定义下列函数：  1）findMaxElement1D(), 返回一维数组中的最大值。  int findMaxElement1D(int array[], int n){  int max = array[0];  for(int i = 0; i < n; i++)  if(array[i] > max)  max = array[i];  return max;  }  2）findMaxElement2D(), 返回二维数组中的最大值  int findMaxElement2D(int array[][5], int r){  int max = array[0][0];  for(int i = 0; i < r; i++)  for(int j = 0; j < 5; j++)  if(array[i][j] > max)  max = array[i][j];  return max;  与1）类似，不过由于是二维数组，需要用到双重循环，第一层循环代表数组的行，第二层代表数组的列，遍历数组。传入的参数为二维数组，以最大10列的数组为例。  3）sort1D(), 对一维数组用冒泡法排序  void sort1D(int array[], int n){  int temp;  for(int i = 0; i < n - 1; i++)  for(int j = 0; j < n - 1 - i; j++)  if(array[j] > array[j+1]){  temp = array[j];  array[j] = array[j+1];  array[j+1] = temp;  }  }  冒泡排序，利用双层循环，第一次循环为每一次比较大小时比较的元素个数，第二层循环从数组的第1个数开始，分别与其后一个数字作比较，如果该数字比其后的数字大，则作交换，否则无动作。由于每次第二层循环都会使相对最大值排到数组末，所以第二层循环比较的数字的个数每次减一，直到排完为止。  测试：  int main(){  int a[] = {1, 5, 3, 8, 4, 6};  int b[][5] = {{1,2,3,4,5},{2,5,9,4,7},{5,12,65,7,3}};  int c = findMaxElement1D(a, 6);  int d = findMaxElement2D(b, 3);  cout << c << "\n";  cout << d << "\n";  sort1D(a,6);  for(int i = 0; i < 6; i++)  cout << a[i];  return 0;  }  结果：    2、建立头文件myFunctions.h。  #include "myFunctions.cpp"  int findMaxElement1D(int array[], int n);  int findMaxElement2D(int array[][5], int r);  void sort1D(int array[], int n);  3、创建源程序文件main.cpp, 在其中#include “myFunctions.h”, 定义主函数main(),定义一个一维数组，定义一个二维数组，从键盘输入数组元素，输出这两个数组，并调用上述3个函数，输出函数返回的结果。  --begin--cpp--code--  #include "myFunctions.h"  #include<iostream>  using namespace std;  int main(){  int a[8];  int b[3][5];    cout << "enter a 1D array:" << endl;  for(int i = 0; i < 8; i++)  cin >> a[i];  for(int i = 0; i < 8; i++)  cout << a[i] << "\t";  cout << "\n";    cout << "enter a 2d array:" << endl;  for(int i = 0; i < 3; i++)  for(int j = 0; j < 5; j++)  cin >> b[i][j];  for(int i = 0; i < 3; i++)  for(int j = 0; j < 5; j++)  cout << b[i][j] << "\t";  cout << "\n";    sort1D(a, 8);  cout << "\n" << endl;  cout << "the max in 1D array is: " << findMaxElement1D(a, 8) << "\n";  cout << "the max in 2D array is: " << findMaxElement2D(b, 3) << "\n";  cout << "1D array sorted:";  for(int i = 0; i < 8; i ++)  cout << a[i] << "\t";    return 0;  }  --end--cpp--code--  结果如下图所示：    4. 有一字符串，包含n个字符。写一个函数，将此字符串中从第m个字符开始的全部字符复制成为另一个字符串。  void copy(char c1[], int m){  int n = strlen(c1);  char \*p = &c1[m-1];  char c2[n-m+1];  for(int i = 0; i < n - m + 1; i++)  c2[i] = \*p++;  for(int i = 0; i < n - m + 1; i++)  cout << c2[i];  }  copy函数中传入的参数为被复制的字符数组（字符串）c1和从第m个字符开始复制。利用strlen（）函数获得字符串的长度。定义指针p指向c1中的第m个字符（下标为m-1），定义一个新的字符数组c2，长度为c1从第m个字符开始到最后一个字符de个数即n-m+1。再利用for循环将p所指向的字符赋值给c2[0], 指针p不断后移，以此类推。最后输出c2。  在主函数中调用，输出结果。    5. 写一个函数，将一个3×3的整型矩阵转置。  void transpose(int \*a){  int temp;  temp = \*(a + 1);  \*(a + 1) = \*(a + 3);  \*(a + 3) = temp;    temp = \*(a + 2);  \*(a + 2) = \*(a + 6);  \*(a + 6) = temp;    temp = \*(a + 5);  \*(a + 5) = \*(a + 7);  \*(a + 7) = temp;  }  由于是用一维数组表示3x3矩阵，根据3x3矩阵转置规律，只需将数组第二个元素与第四个元素互换，将第三个元素与第七个元素互换，第6个元素与第8个元素互换即可。  在主函数中调用，输出结果。用一维数组表示这个矩阵。  int main(){  int a[9],count = 0;  cout << "enter a matrix: ";  for(int i = 0; i < 9; i++)  cin >> a[i];  cout << "original matrix:" << "\n";  for(int i = 0; i < 9; i++){  cout << "\t" << a[i];  count++;  if(count % 3 == 0)  cout << "\n";  }  transpose(a);  cout << "transposed matrix:" << "\n";  for(int i = 0; i < 9; i++){  cout << "\t" << a[i];  count++;  if(count % 3 == 0)  cout << "\n";  }  }  结果如图：    6. 输入一个字符串，内有数字和非数字字符，将其中连续的数字作为一个整数，依次存放到一数组a中。  我们先定义字符串str[]，整形数组a[]，整形value和count。a[]用来存放字符串中的数字，value是从字符串中提取出来的整型数，count是计数器，用来计有多少数存在了a[]中。  char str[]="hu123nn5fja8;[]67";  int a[] = {0}, value = 0, count = 0;  bool hasValue = false;  char \*p;  p = &str[0];  之后进入双层while循环，第一层循环为a[count]赋值value并控制指向非数字字符的指针p后移，循环条件为\*p != ‘\0’。第二层循环给value赋值，hasValue变为true并控制指向数字的指针p后移直到p不再指向数字结束循环。循环条件为\*p >= '0' && \*p <= '9'  while(\*p != '\0'){  while(\*p >= '0' && \*p <= '9'){  value = value \* 10 + \*p - '0';  p = p + 1;  hasValue = true;  }  if(hasValue){  a[count] = value;  count++;  value = 0;  hasValue = false;  }  p = p + 1;  }  计算value时，用\*p-‘0’可直接得到\*p所指字符对应的数字。  最后输出：  cout << "字符串中的数字：" << "\n";  for(int j = 0; j < count; j++){  cout << a[j] << "\t";  }  cout << "共有" << count << "个";  return 0;  完整程序：  --begin--cpp--code--  #include<iostream>  #include<string.h>  using namespace std;  int main(){  char str[]="hu123nn5fja8;[]67";  int a[] = {0}, value = 0, count = 0;  bool hasValue = false;  char \*p;  p = &str[0];  while(\*p != '\0'){  while(\*p >= '0' && \*p <= '9'){  value = value \* 10 + \*p - '0';  p = p + 1;  hasValue = true;  }  if(hasValue){  a[count] = value;  count++;  value = 0;  hasValue = false;  }  p = p + 1;  }  cout << "字符串中的数字：" << "\n";  for(int j = 0; j < count; j++){  cout << a[j] << "\t";  }  cout << "共有" << count << "个";  return 0;  }  --end--cpp--code--  运行结果： | | | |
| 结论分析与体会：  1、要注意c++中二维数组的第二维必须声明具体的数，在将二维数组名作为函数参数时忽略了这一点，导致程序无法运行。  2、又学会了一种排序方法：冒泡排序，即将一个元素与其后的元素相比再做交换。  3、字符串相当于字符数组。  4、指针部分对于我们初学者，尤其是先学java的初学者来说有点难度，注意不要混淆p指向地址与\*p表示内容，例如p=&a[1]和\*p=a[1]。但指针相比数组更灵活。 | | | |