**实验四、数组与指针**

**一、实验目的与要求**

1、进一步加深对数组的理解，掌握数组的定义方法；

2、掌握数组的处理方法、数组作为函数参数的使用方法，以及搜索与排序的应用。

3、掌握指针的概念、指针变量定义格式以及指针的运算；

4、掌握指针与数组、函数的关系；

5、理解内存动态分配的含义、熟练掌握内存动态分配方法；

6、掌握递归函数的定义方法

**二、实验内容**

**（一）数组**

1、打印不同的数：

编写一个程序，读入 10 个数，输出其中不同的数（即如果一个数出现多次，只打印一次）。

提示：读入的数如果是一个新的值，则将其存入一个数组。否则，将其丢弃。输入完毕后，

数组中保存的就是不同的数。

下面是一个运行样例：

Enter ten numbers: 1 2 3 2 1 6 3 4 5 2

The distinct numbers are: 1 2 3 6 4 5

2、起泡排序：

利用起泡排序算法编写一个排序函数。起泡排序算法分若干趟对数组进行处理。每趟处理中，

对相邻元素进行比较。若为降序，则交换；否则，保持原顺序。此技术被称为起泡排序（

bubble

sort）或下沉排序（

sinking sort），因为较小的值逐渐地“冒泡”到上部，而较大值逐渐下沉到底

部。

算法可描述如下：

bool changed = true;

do

{

changed = false;

for (int j = 0; j < listSize – 1; j++)

if (list[j] > list[j+1])

{

swap list[j] with list[j+1];

changed = true;

}

} while (changed);

很明显，循环结束后，列表变为升序。容易证明 do 循环最多执行 listSize – 1 次。

编写测试程序，读入一个含有 10 个双精度数字的数组，调用函数并显示排列后的数字。

3、游戏：存物柜问题：

一个学校有 100 个存物柜，100 个学生。开学第一天所有存物柜都是关闭的。第一个学生（记

为 S1）来到学校后，打开所有的存物柜。第二个学生 S2，从第二个存物柜（记为 L2）开始，每

隔两个存物柜，将它们关闭。第三个学生 S3 从第三个存物柜 L3 开始，每隔三个，将它们的状态

Enter10

改变（开着的关上，关着的打开）。学生 S4，从 L4 开始，每隔四个改变它们的状态。学生 S5，

从 L5 开始，每隔五个改变状态。依此类推，直至学生 S100 改变 L100 的状态。

当所有学生完成这个过程，那些存物柜是开着的？编写一个程序求解此问题，显示所有开着

的柜子号码，号码之间用一个空格隔开。

提示：使用一个 100 个布尔型元素的数组，每个元素代表存物柜是开（

true）或关（

false）。

最初所有的储物柜都是关闭的。

4、合并两个排列好的数组：

编写如下函数，合并两个排列好的数组，形成一个新的排列好的数组。

void merge(const int list1[], int size1, const int list2[], int size2, int list3[])

使用 size1+size2 次比较实现函数。编写测试程序，提示用户输入两个排列好的数组，并显示

合并以后的数组。下面是一个运行样例。注意，输入数据的第一个数字是数组的元素数，而不是

数组的一部分。假定数组大小不超过 80。

Enter list1: 5 1 5 16 61 111

Enter list1: 4 2 4 5 6

The merged list is 1 2 4 5 5 6 16 61 111

5、检验子串：

编写如下函数，检验 C 字符串 s1 是否是 C 字符串 s2 的子串。如果匹配，返回 s1 在 s2 中的

下标，否则返回–1。

int indexOf(const char s1[], const char s2[])

编写测试程序，读入两个 C 字符串，检验 C 字符串 s1 是否是 C 字符串 s2 的子串。下面是程

序的运行样例：

Enter the first string: welcome

Enter the second string: We welcome you!

indexOf(“welcome”, “We welcome you!”) is 3

Enter the first string: welcome

Enter the second string: We invite you!

indexOf(“welcome”, “We invite you!”) is –1

6、字符串中每个字母出现的次数：

请使用如下函数头编写函数，数出字符串中每个字母出现的次数。

void count(const char s[], int counts[])

counts 是一个有 26 个元素的整数数组。const[0]，const[1]，…，const[25]分别记录 a，b，…，

z 出现的次数。字母不分大小写，例如字母 A 和字母 a 都被看作 a。

编写测试程序，读入字符串并调用 count 函数，显示非零的次数。下面是程序的一个运行样

例：

Enter

Enter

Enter

Enter

Enter

Enter

Enter11

Enter a string: Welcome to New York!

c: 1 times

e: 3 times

k: 1 times

l: 1 times

m: 1 times

n: 1 times

o: 3 times

r: 1 times

t: 1 times

w: 2 times

y: 1 times

**（二）指针**

、程序设计

(1)编写函数检查字符串 s1 是否为字符串 s2 的子串，若是，返回第一次匹配的下标，否则返

回-1。在主程序中输入字符串 s1 与 s2，调用函数实现。

函数原型：int indexof(const char \*s1,const char \*s2);

(2)编写一个函数将以字符串形式表示的一个 16 进制数转换为 10 进制数，并在主函数中测

试。函数原型 int parseHex(const char \*const hexString);

如：调用函数 parseHex(“A5”);返回 165

(3) 主程序中建立一动态数组（使用 new），数组元素及元素个数由键盘输入，动态调试观

察指针及指针指向的内容；设计一个函数对数组由小到大排序；主程序中用指针方式输出数组元

素；最后释放数组内存（

delete）。

1. **算法分析、程序与结果**

**(一）数组**

1.程序

#include <iostream>

using namespace std;

int main()

{

int i,j,k=0,a[10];

cout << " Enter ten numbers :";

for (i = 0; i < 10; i++)

{

cin >> a[i];

}

cout << " The distinct numbers are:";

for (i = 0; i < 9; i++)

{

bool c = true;

for(j=i+1;j<9;j++)

{

if (a[i] == a[j])

c = false;

}

if(c)

cout << a[i] << " ";

}

if (a[8] < a[9])

{

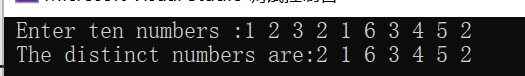
cout << a[9] << endl;

}

return 0;

}

结果



1. 程序

#include<iostream>

using namespace std;

void swap(double& a, double& b)

{

double temp = a;

a = b;

b = temp;

}

int main()

{

int listSize = 9; int i ;

double list[10];

bool changed = true;

cout << "Enter ten numbers:";

for (i=0;i<listSize+1;i++)

{

cin >> list[i];

}

do

{

changed = false;

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

if (list[j] > list[j + 1])

{

swap( list[j], list[j + 1]);

changed = true;

}

} while (changed);

cout << "The distinct numbers are:";

for (i = 0; i < listSize+1; i++)

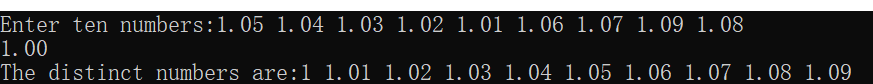
{

cout << list[i] << " ";

}

return 0.01;

}结果



3程序

#include<iostream>

using namespace std;

int main()

{

bool a[100];

for(int i=0;i<100;i++)

{

a[i] = true;

}

for (int j = 2; j < 100; j++)

{

for (int k = 0; k < 100; k++)

{

if(k-j+1>=0)

if ((k-j+1) % j == 0)

{

if (a[k] == true)

{

a[k] = false;

}

else { a[k] = true; }

}

}

}

for(int i=0;i<100;i++)

{

if (a[i])

{

cout << "L" << i+1 << " ";

}

}

return 0;

}

结果



**4程序**

#include<iostream>

using namespace std;

void merge( int list1[], int size1, int list2[], int size2, int \*list3,int size3)

{

int temp;

for (int i = 0; i < size1; i++)

{

list3[i] = list1[i];

}

for(int i=0;i<size2;i++)

{

list3[i + size1 ] = list2[i];

}

for(int i = 0; i < size3;i++)

{

for (int j = 0;j<size3-1;j++)

{

if(list3[j]>list3[j+1])

{

temp = list3[j];

list3[j] = list3[j+1];

list3[j+1] = temp;

}

}

}

for (int i = 0; i < size3; i++)

{

cout << list3[i] << " ";

}

}

int main()

{

int \*list3, list1[80], list2[80], size1, size2;

cout << "Enter list1:";

cin >> size1;

for(int i=0;i<size1;i++)

{

cin >> list1[i];

}

cout << "Enter list2:";

cin >> size2;

for (int i = 0; i < size2; i++)

{

cin >> list2[i];

}

int size3 = size1 + size2;

list3 = new int [size3];

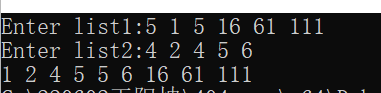
merge( list1, size1, list2, size2, list3,size3);

delete[] list3;

return 0;

}

结果



5程序

#include<iostream>

using namespace std;

int indexOf(char s1[], char s2[])

{

bool t = false; int a = 0,b=0;

for(int i=0;i<50;i++)

{

if(s1[0]==s2[i])

{

for (int j = 0; j < strlen(s1); j++)

{

++b;

if (s1[j] == s2[i + j])

{

t = true; a = i + j - strlen(s1)+1;

}

else { t = false; }

}

if (b == (strlen(s1)) && t == true)

break;

}

}

if (t == true)

{

return a;

}

if(t==false)

{

return -1;

}

}

int main()

{

char s1[50], s2[50];

cout << "Enter first string:";

cin.getline(s1, 50);

cout << "Enter second string:";

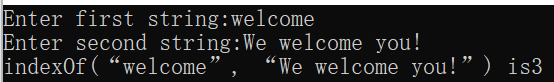
cin.getline(s2, 50);

cout << "indexOf(“" << s1 << "”, “" << s2 << "”) is" << indexOf(s1, s2)<<endl;

return 0;

}

结果



6程序

#include<iostream>

using namespace std;

void count(char s[],int counts[])

{

int num = 26;

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

{

counts[i] = 0;

}

for(int i=0;i<strlen(s);i++)

{

if(s[i]>='A'&&s[i]<='Z')

{

s[i] = s[i] + 'a' - 'A';

}

if (s[i] >= 'a' && s[i] <= 'z')

{

counts[s[i] - 'a']++;

}

}

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

{

if (counts[i] > 0)

{

cout << (char)('a' + i) << ":" << counts[i] << " " << "times" << endl;

}

}

}

int main()

{

char s[50]; int counts[26];

cout << "Enter a string:";

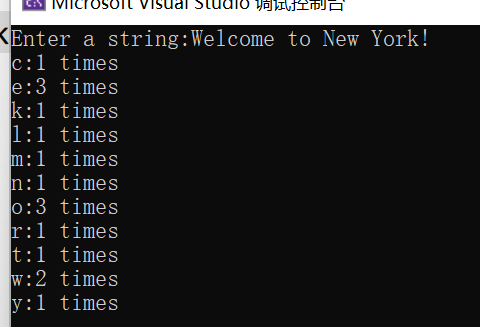
cin.getline(s, 50);

count(s, counts);

return 0;

}

结果



（二）指针

1程序

#include<iostream>

using namespace std;

int indexof(char\* s1, char\* s2)

{

bool t = false; int a = 0,b=0;

for (int i = 0; i < 50; i++)

{

if (s1[0] == s2[i])

{

for (int j = 0; j < strlen(s1); j++)

{

++b;

if (s1[j] == s2[i + j])

{

t = true; a = i + j - strlen(s1)+1;

}

else { t = false; }

}

if (b == strlen(s1) && t == true)

break;

}

}

if (t == true)

{

return a;

}

if(t==false)

{

return -1;

}

}

int main()

{

char s1[50], s2[50];

cout << "Enter first string:";

cin.getline(s1, 50);

cout << "Enter second string:";

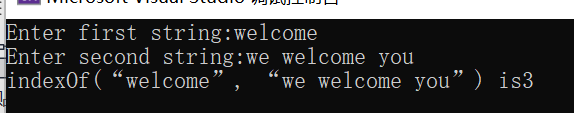
cin.getline(s2, 50);

cout << "indexOf(“" << s1 << "”, “" << s2 << "”) is" << indexof(s1, s2) << endl;

return 0;

}

结果



2程序

#include<iostream>

using namespace std;

int parseHex(char\* hexString)

{

int sum = 0;

for (int i = 0; i < strlen(hexString); i++)

{

if (hexString[i] >= 'A' && hexString[i] <= 'F')

{

hexString[i] = hexString[i]- 'A' + 10;

}

else { hexString[i] -= '0'; }

}

for (int i = 0; i < strlen(hexString); i++)

{

sum += hexString[i] \* pow(16, strlen(hexString) - i-1);

}

return sum;

}

int main()

{

char hexString[50];

cout << "Enter hex:";

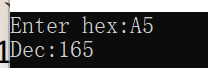
cin.getline(hexString, 50);

cout << "Dec:" << parseHex(hexString) << endl;

return 0;

}

结果



3程序

#include<iostream>

using namespace std;

void swap(int\* p, int size)

{

int temp;

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

for (int j = 0; j < size - 1; j++)

{

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

{

temp = \*(p + j);

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

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

}

}

}

int main()

{

int size,a;

cout << "输入元素个数";

cin >> size;

int\* p = new int[size];

cout << "输入数组";

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

{

cin >>\*(p+i);

}

cout << "输入你想输出的数组";

cin >> a;

cout << "p[" << a - 1 << "]=" << p[a - 1] << endl;

swap(p, size);

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

{

cout << p[i] << " ";

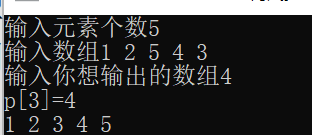
}

delete[] p;

return 0;

}

结果



1. 遇到的问题与解决办法

由于一段时间没有接触指针与数组一些地方有些生疏，比如一些字符型数组的输入和检测数组长度知识点遗忘，通过询问同学和网上查找得以解决

五、体会

指针和数组算是目前的一个难点，还需要多加练习巩固并进行学习来更好的掌握相关知识点