**实验四、数组与指针**

**一、【实验目的】**

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、【实验内容与步骤】**

（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 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

2、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）。

**三、【实验步骤、算法与结果分析】**

1、程序：

#include <iostream>

using namespace std;

int main()

{

const int size=10;

int list[size];

cout << "Enter ten numbers:" << endl;

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

{

cin >> list[i];

}

cout << list[0] <<"\t";

for(int m=1;m<size;m++)

{

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

{

if(list[m]!=list[j])

{

if(j==m-1)

{

cout <<list[m]<<"\t";

}

else

{

continue;

}

}

else

{

break;

}

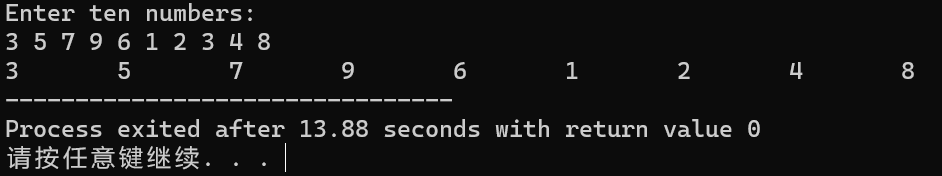
}

}

return 0；

}

结果：



2、程序：

#include<iostream>

using namespace std;

const int size=10;

void bubblesort(double list[10])

{

double temp;

bool changed=true;

do

{

changed=false;

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

{

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

{

temp=list[j];

list[j]=list[j+1];

list[j+1]=temp;

changed=true;

}

}

}while(changed);

}

int main()

{

double list[10];

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

{

cin >> list[i];

}

bubblesort(list);

for(int k=0;k<10;k++)

{

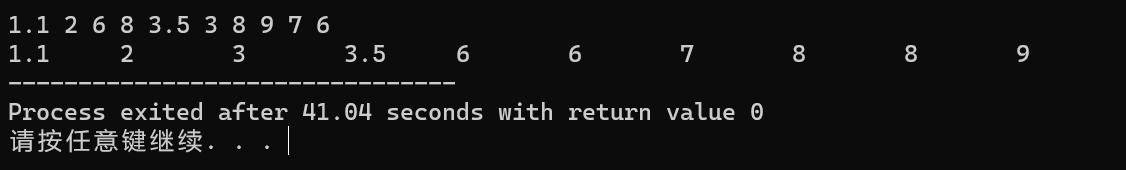
cout << list[k] << "\t";

}

return 0;

}

结果：



3、程序：

#include<iostream>

using namespace std;

const int q=1;

int main()

{

bool box[100];

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

{

box[k]=false;

}

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

{

for(int n=m;n<100;n+=m+1)

{

if(box[n]==true)

{

box[n]=false;

}

else

{

box[n]=true;

}

}

}

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

{

if(box[p]==true)

{

cout << p+1 <<"\t";

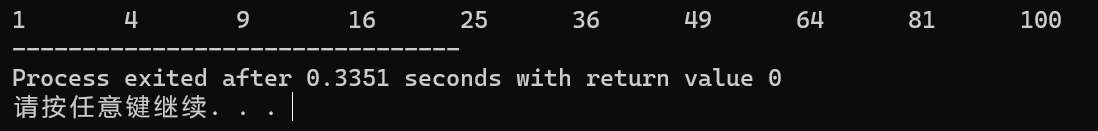
}

}

return 0;

}

结果：



4、程序：

#include<iostream>

#include<string>

using namespace std;

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

int main()

{

int size1,size2,size3;

cout<<"Enter size1: ";

cin>>size1;

cout<<"Enter list1: ";

int list1[size1];

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

cin>>list1[i];

}

cout<<"Enter size2: ";

cin>>size2;

cout<<"Enter list2: ";

int list2[size2];

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

cin>>list2[i];

}

size3=size1+size2;

int list3[size3];

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

cout<<endl;

return 0;

}

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

{

int temp;

for (int j=0;j<size1;j++){

list3[j]=list1[j];

}

for(int k=0;k<size2;k++){

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

}

cout<<"The merged list is : ";

for (int m=0;m<(size1+size2)-1;m++){

for(int b=0;b<(size1+size2)-1;b++){

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

{

temp=list3[b];

list3[b]=list3[b+1];

list3[b+1]=temp;

}

}

}

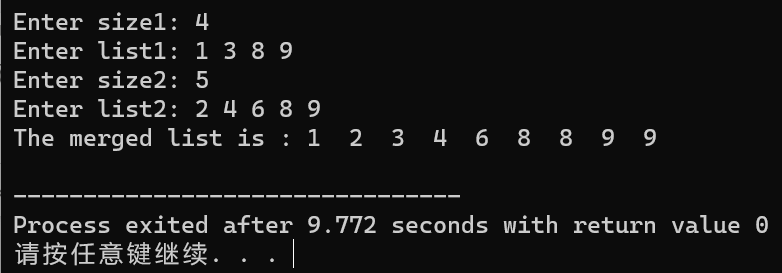
for (int a=0;a<(size1+size2);a++){

cout<<list3[a]<<" ";

}

}

结果：



5、程序：  
#include<iostream>

#include<cstring>

using namespace std;

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

int main()

{

int size=999;

char s1[size],s2[size];

cout<<"Enter the first string : ";

cin.getline(s1,size);

cout<<"Enter the second string: ";

cin.getline(s2,size);

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

return 0;

}

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

{

int a,b,index,j,k;

a=strlen(s1);

b=strlen(s2);

bool s3[a];

for(int m=0;m<a;m++)

s3[m]=false;

for(int i=0;i<b-a+1;i++) //控制s2的首字母

{

for(j=0,k=i;(j<a)&&(k<i+a);j++,k++) //控制s2的总输出数

if(s1[j]==s2[k])

{

index=i;

s3[j]=true;

}

int h=0;

for(int n=0;n<a;n++)

{

if(s3[n]==false&&i==b-a)

{

return -1;break;

}

if(s3[n])

h++;

}

if(h==a)

{

return index;

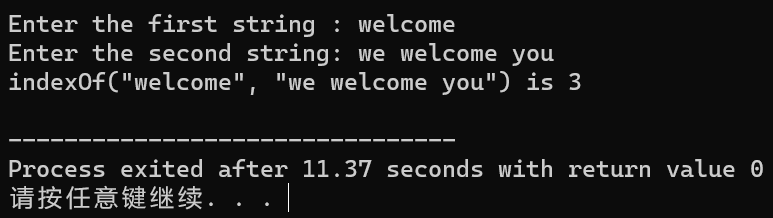
}

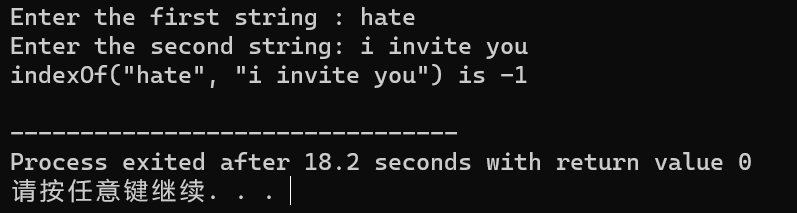
}

return 0;

}

结果：





6、程序：

#include <iostream>

using namespace std;

const int num1=26;

const int num2=99;

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

{

for(int i=0;i<num1;i++)

{

counts[i]=0;

}

for(int k=0;k<num2;k++)

{

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

counts[s[k]-'A']++;

}

for(int j=0;j<num1;j++)

{

if(counts[j]!=0)

{

cout << static\_cast<char>(j+'a') << ":" << counts[j] << "times" << endl;

}

}

}

int main()

{

char s[num2];

int counts[num1];

cout << "Enter a string:";

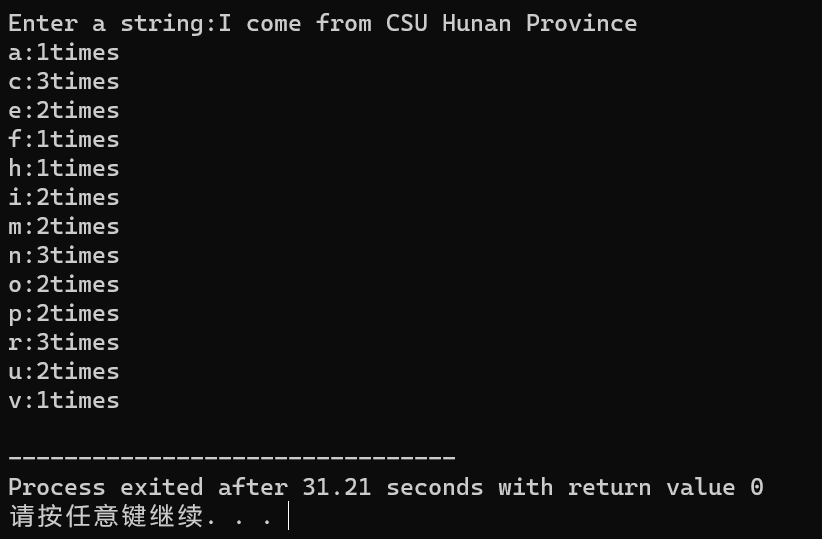
cin.getline(s,num2);

count(s,counts);

return 0;

}

结果：



7、程序：

#include<iostream>

#include<cstring>

using namespace std;

int indexOf(char \*s1, char \*s2);

int main()

{

int size=999;

char s1[size],s2[size];

cout<<"Enter the first string : ";

cin.getline(s1,size);

cout<<"Enter the second string: ";

cin.getline(s2,size);

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

return 0;

}

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

{

int a,b,index,j,k;

a=strlen(s1);

b=strlen(s2);

bool s3[a];

for(int m=0;m<a;m++)

s3[m]=false;

for(int i=0;i<b-a+1;i++) //控制s2的首字母

{

for(j=0,k=i;(j<a)&&(k<i+a);j++,k++) //控制s2的总输出数

if(s1[j]==s2[k])

{

index=i;

s3[j]=true;

}

int h=0;

for(int n=0;n<a;n++)

{

if(s3[n]==false&&i==b-a)

{

return -1;break;

}

if(s3[n])

h++;

}

if(h==a)

{

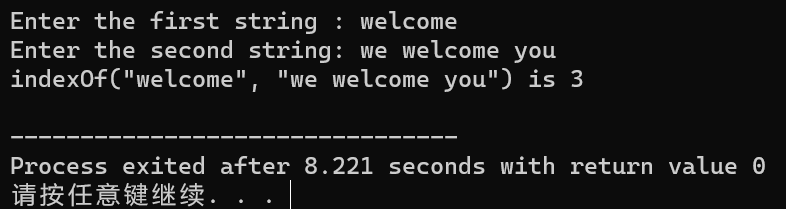
return index;

}

}

return 0;

}

结果：  


8、程序：

#include<iostream>

#include<string>

#include<cmath>

using namespace std;

int parseHex(const char\*const hexString)

{

int a,sum=0;

a=strlen(hexString);

int list[a];

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

{

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

{

list[i]=(static\_cast<int>(hexString[i])-55)\*(pow(16,a-1-i));

}

else

{

list[i]=(hexString[i]-48)\*pow(16,a-1-i);

}

sum=sum+list[i];

}

return sum;

}

int main(){

char \* str;

str= new char[20];

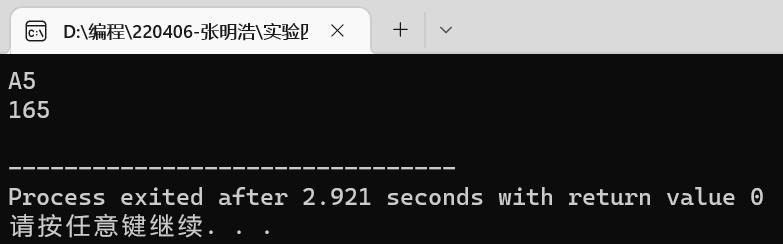
cin.get(str,20);

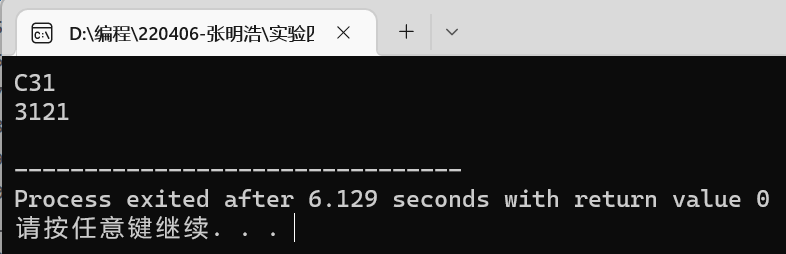
cout << parseHex(str) << endl;

return 0;

}

结果：





9、程序：

#include<iostream>

#include<string>

using namespace std;

int sort(int p[],int size)

{

int temp;

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

{

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

{

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

{

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

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

\*(p+i)=temp;

}

}

}

return 0;

}

int main()

{

int size,l;

cin >> size;

int \*p;

p=new int[size];

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

{

cin >> \*(p+m);

}

cout << "Enter the number of this arr:" << endl;

cin >> l;

cout << \*(p+l-1) << endl;;

sort(p,size);

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

{

cout << \*(p+m) << "\t";

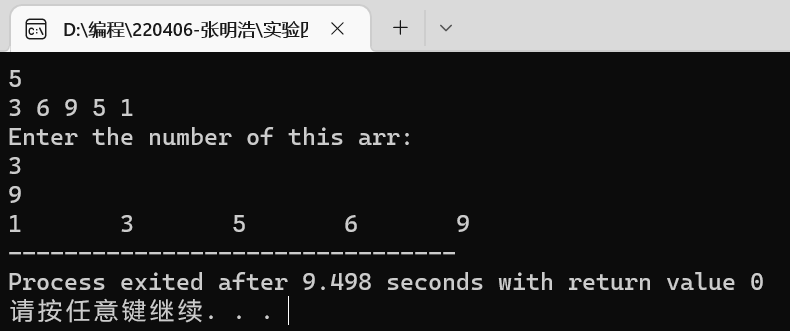
}

delete p;

return 0;

}

结果：



**四、遇到的问题与解决方法：**

这次实验几乎每道题都很难，每道题都至少写了两个小时，其中的问题也很多，最难的还是指针的运用。不会求取指针指向的动态整型数组的长度，查找资料后才知道这样的数组长度求不出来，只能使用全局变量事先给定。数组的排序看似简单，但稍微复杂一点的数组需要考虑的东西就很多，需要缜密的思维，需要将这些比较步骤转化成代码。有些细微的差错通过请教老师和同学解决。

五、体会：

数组和指针果然是C++中最难的部分之二，做完这个实验，发现还有很多数组和指针方面的知识没有掌握，没有理解，以至于做实验的时候不会灵活运用。应该马上把他们补起来，多花点时间看看书动手写代码帮助理解。