**计算机程序设计基础（C++)**

**实验报告**

专业班级： 软工2206

学 号： 8202220718

姓 名： 齐文川

**实验报告成绩：**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **实验** | **实验一** | **实验二** | **实验三** | **实验四** | **实验五** | **总评** |
| **成绩** |  |  |  |  |  |  |

**批阅教师：**

**实验三 函数**

**一、实验目的**

本实验主要培养、训练学生对函数的理解，要求：

1. 掌握函数的定义、声明的方法；

2. 掌握函数的编写要求；

3. 掌握函数的调用方法；

4. 掌握函数参数的传递方法；

5. 掌握变量的作用域；

6. 掌握多文件编程方法。

**二、实验内容与要求**

1、输入自然数m和n，

（1）求他们的最大公约数（或称最大公因数）。

要求输入、输出在主函数中进行，求公约数由函数实现。

1. 在函数中求最大公约数与最小公倍数。（提示：使用引用参数）

2. 编写程序满足：声明一个函数，判断一个整数是否为素数，使用如下函数头：

bool is\_prime(int num) ,如果num是素数函数返回true，否则返回false；

利用函数is\_prime找出前200个素数，并按每行10个输出：

     2     3      5      7    11    13    17    19    23    29

3、编程实现摄氏温度到华氏温度的转换：

编写一个头文件，包含下面两个函数：

double celsius\_to\_fah(double cel)    //摄氏温度到华氏温度

double fahrenheit\_to\_cels(double fah) //华氏温度到摄氏温度

实现头文件，并编写测试程序，调用函数显示如下结果：

Celsius    Fahrenheit   |   Fahrenheit       Celsius

40.0       105.0        |   120.0            48.89

39.0       102.0        |   110.0            43.33

……       ……        |   ……             ……

31.0        87.8        |   30.0             -1.11

（测试程序为主模块，即main( )函数所在的CPP文件，头文件mytemperature.h只有函数声明；函数定义写在另一CPP文件mytemperature.cpp）

4、创建名为mytriangle.h的头文件，包括：

bool is\_valid(double side1,double side2,double side3)

double\_area(double side1,double side2, double side3)

面积=sqrt(s(s-side1)(s-side2)(s-side3))

其中s=(side1+side2+side3)/2

写测试程序：读取三角形三边长，如输入合法，计算面积，否则输出错误信息。

（测试程序为主模块，即main( )函数所在的CPP文件，头文件mytriangle.h只有函数声明；函数定义写在另一CPP文件mytriangle.cpp）

**3与4选一个完成**

5、猴子吃桃：猴子第一天摘若干桃子，当即吃了一半，还不过瘾，又吃了一个。第二天又将剩下的桃子吃掉一半，又多吃一个，以后每天如此，到第10天，发现只剩最后一个桃子，问，第一天猴子共摘多少桃子（用递归实现）。

**三、实验思考题**

1. 本实验中函数中返回的值为什么与函数类型一致？

函数中返回的值与函数类型一致是为了保证类型安全和语法规范性。

在函数定义时，我们会明确指定函数的返回类型，这样在函数体中编写代码时，就知道应该返回何种类型的值。如果函数返回的值与函数类型不一致，就会导致类型错误，从而引发编译错误或运行时错误。

总之，函数返回的值与函数类型一致，能够保证类型安全和语法规范性，同时提升代码的可读性和可维护性。

2. 本实验中主函数调用函数时采用的是何种传递方式？

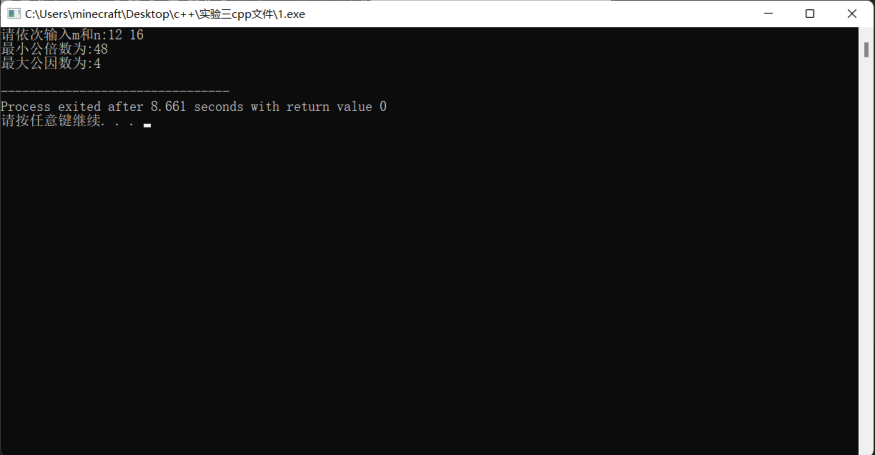
采用指针的方式进行参数的传递。

1. **算法分析，程序结果**

**第一题实验代码:**

1. #include <iostream>
2. using namespace std;
3. void count(int m,int n,int\*least\_common\_multiple,int\*pcd){
4. if(m<n){
5. for(int i=1;i<=m;i++){
6. if(m%i==0&&n%i==0){
7. \*pcd=i;
8. }
9. }
10. for(int j=m\*n;j>=n;j--){
11. if(j%m==0&&j%n==0){
12. \*least\_common\_multiple=j;
13. }
14. }
15. }
16. else{
17. for(int i=1;i<=n;i++){
18. if(m%i==0&&n%i==0){
19. \*pcd=i;
20. }
21. }
22. for(int j=m\*n;j>=m;j--){
23. if(j%m==0&&j%n==0){
24. \*least\_common\_multiple=j;
25. }
26. }
27. }
28. }
29. int main(){
30. int m,n;
31. cout<<"请依次输入m和n:";
32. cin>>m>>n;
33. int least\_common\_multiple,pcd;*//声明最小公倍数和最大共因数*
34. int\*p1,\*p2;
35. p1=&least\_common\_multiple;
36. p2=&pcd;
37. count(m,n,p1,p2);
38. cout<<"最小公倍数为:"<<least\_common\_multiple<<"\n";
39. cout<<"最大公因数为:"<<pcd<<"\n";
40. delete p1;
41. delete p2;
42. return 0;
43. }

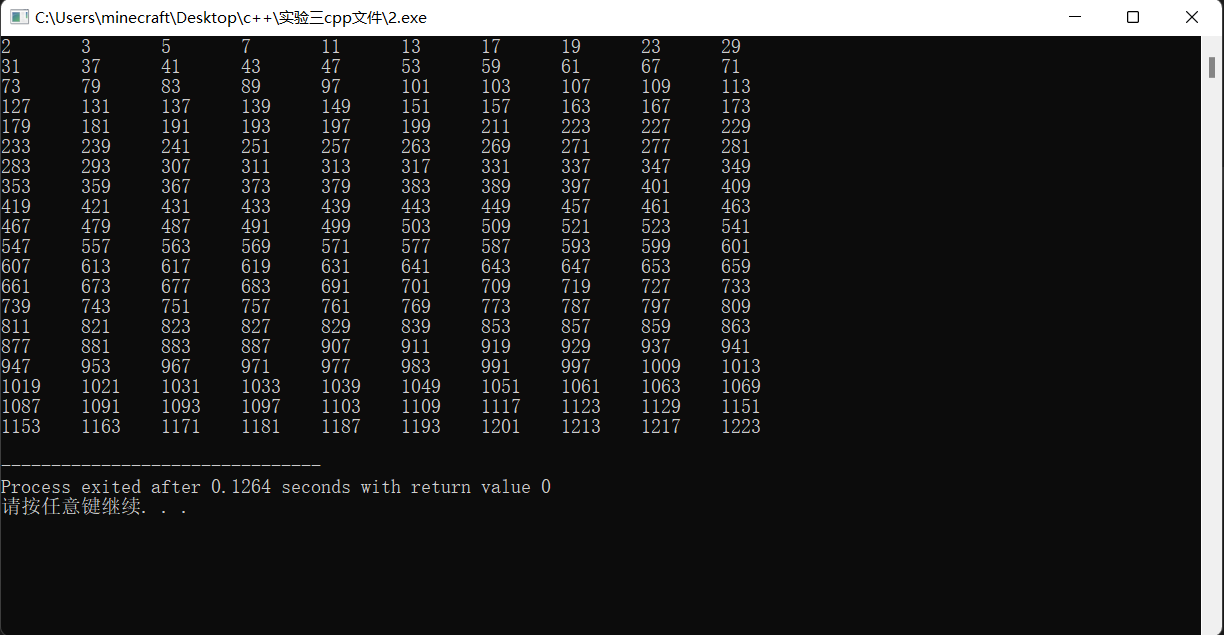
**实验结果:**

****

**第二题实验代码:**

1. #include <iostream>
2. using namespace std;
3. bool is\_prime(int num){
4. for(int i=2;i<num;i++){
5. if(num%i==0){
6. return false;
7. }
8. }
9. return true;
10. }
11. int main(){
12. int count=0,num=2;
13. while(count<200){
14. if(is\_prime(num)){
15. cout<<num<<"\t";
16. count++;
17. if(count%10==0){
18. cout<<"\n";
19. }
20. }
21. num++;
22. }
23. return 0;
24. }

**实验结果:**

****

**第四题实验代码:**

**mytriangle.h**

1. bool is\_vaild(double side1, double side2, double side3);
2. double area(double side1, double side2, double side3);

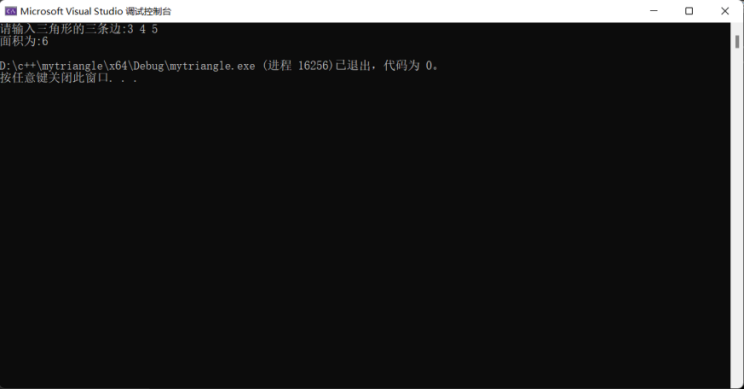
**main.cpp**

1. #include <iostream>
2. #include "mytriangle.h"
3. using namespace std;
4. int main() {
5. cout << "请输入三角形的三条边:";
6. int side1, side2,side3;
7. cin >> side1 >> side2 >> side3;
8. if (is\_vaild(side1, side2, side3)) {
9. cout << "面积为:" << area(side1, side2, side3) << "\n";
10. }
11. else {
12. cout << "三条边无法构成三角形。" << "\n";
13. }
14. return 0;
15. }

**mytriangle.cpp**

1. #include <cmath>
2. bool is\_vaild(double side1, double side2, double side3) {
3. if (side1 + side2 >side3&& side2 + side3 > side1 && side1 + side3 > side2) {
4. return true;
5. }
6. else {
7. return false;
8. }
9. }
10. double area(double side1, double side2, double side3) {
11. double area,s;
12. s = (side1 + side2 + side3) / 2;
13. area = sqrt(s \* (s - side1) \* (s - side2) \* (s - side3));
14. return area;
15. }

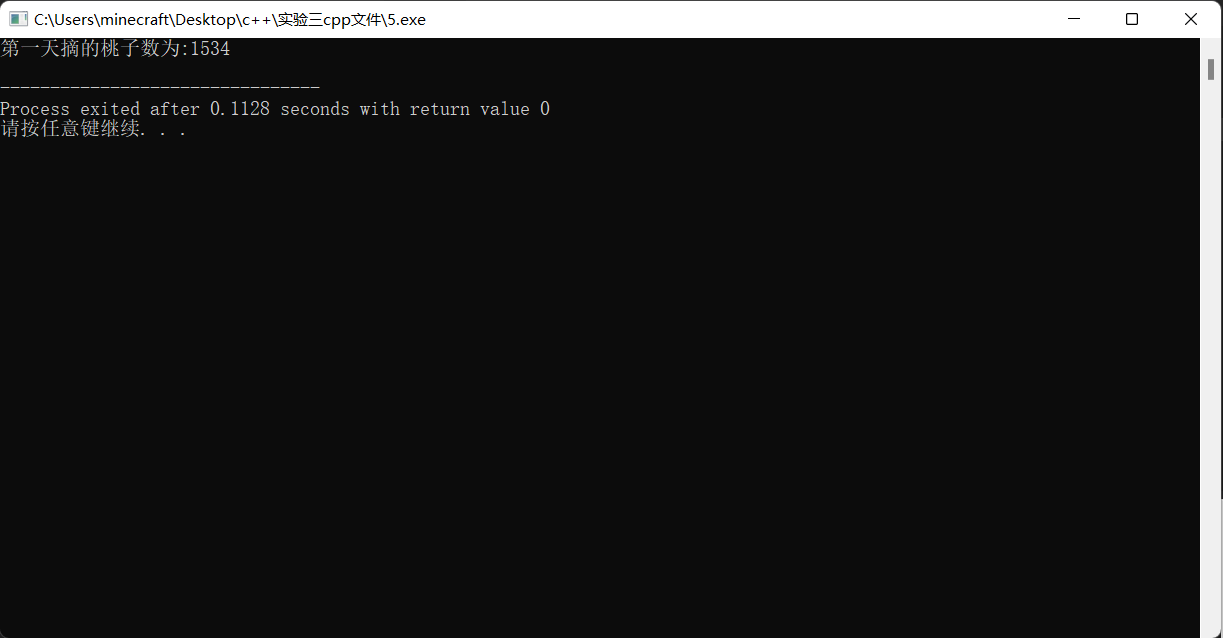
**实验结果:**

****

**第五题实验代码:**

1. #include <iostream>
2. using namespace std;
3. int apple\_num(int day){
4. int num=1;
5. while(day>1){
6. num=(num+1)\*2;
7. day--;
8. apple\_num(day);
9. }
10. return num;
11. }
12. int main(){
13. int day=10;
14. cout<<"第一天摘的桃子数为:"<<apple\_num(day)<<"\n";
15. return 0;
16. }

**实验结果:**

****

**五、遇到的问题与解决方法**

**在本次实验中，解决了一些与函数有关的问题，开始对使用指针作为函数参数并不是特别的了解，在本次实验中也遇到了相应的问题，不过在进行了相关的学习后，也算完整的了解了相关的内容，并且再结合了一定的运用后也较好的掌握了相关的使用方法，另外还有关于分文件编写有关的内容，以前只是粗略的了解和学习，现在真正运用并掌握这种编写程序的方式，使自己的编程能力得到了进一步的提升，其余就没有遇到什么显著的问题了。**

**六、体会**

**通过这次实验，使我的编程及解决问题的能力有了进一步的提升，也提升了我在遇到新的知识时自主学习的能力，也算是通过这个实验而获得的收获，这次实验使我对函数的相关知识有了进一步的了解，了解了怎么运用函数解决一些具体的问题，并且学到了以指针的方式来使一个函数能够返回多个值的方法，希望在接下来的实验中继续努力学到更多的知识。**

**实验四 数组与指针**

【**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。

【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

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

Enter

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

Enter

Enter

Enter the first string: welcome

Enter the second string: We welcome you!

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

Enter

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 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**

1、上机验证下列程序的运行结果（有错误的话自己补充完善）

(1) void main()

｛

int i,j,\*pi,\*pj; //此处的\*表示定义指针变量，而非间接运算符

pi=&i;

pj=&j;

i=5;j=7;

cout<<i<<”\t”<<j<<”\t”<<pi<<”\t”<<pj;

cout<<&i<<”\t”<<\*&i<<”\t”<<&j<<”\t”<<\*&j;

}

输出出现问题，制表符应为”\t”.

运行结果：

上述结果中，pi与&i,pj与&j是地址值，随编译程序而变化，不确定。

(2) int main() //C语言程序，要了解

{

int a[]={1,2,3};

int \*p,i;

p=a; //将数组a首地址送给p

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

printf("%d,%d,%d,%d\n",a[i],p[i],\*(p+i),\*(a+i)); //与cout功能差不多

}

运行结果：

1,1,1,1

2,2,2,2

3,3,3,3

通过这两道题目，希望学生掌握数组元素与指向数组的指针的不同。

a[i]表示数组中下标为i的元素。

a[i]←p[i]←\*(p+i)←\*(a+i)

a是数组名，表示数组首地址，(p+i)表示数组中第i个元素的地址，\*(p+i) 相当于a[i]。

(3)通过如下的问题理解递归函数的定义与调用（递归未讲，可以后做）

#include <iostream> //#include “stdio.h”

void f(char \*st,int i)

{

st[i]=’\0’;

cout<<st; // printf(“%s\n”,st);

if (i>1) f(st,i-1);

}

void main()

{

char st[]=”abcd”;

f(st,4);

}

补充完整，运行时输出为\_\_\_abcdabcaba\_\_\_\_\_

(4)下面程序的主函数中能保证p[0]输出1，p[1]输出2吗？如何修改以保证之（提示：在函数f中使用new生成动态数组；在main中用delete释放。）

#include<iostream>

using namespace std;

int\* f()

{

int\*list = new int[4];

list[0] = 1;

list[1] = 2;

list[2] = 3;

list[3] = 4;

return list;

}

void main()

{

int\* p = f();

cout << p[0] << endl;

cout << p[1] << endl;

delete[] p;

}

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

1. 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）。

【完成实验报告】

**实验报告只要求写程序设计部分**

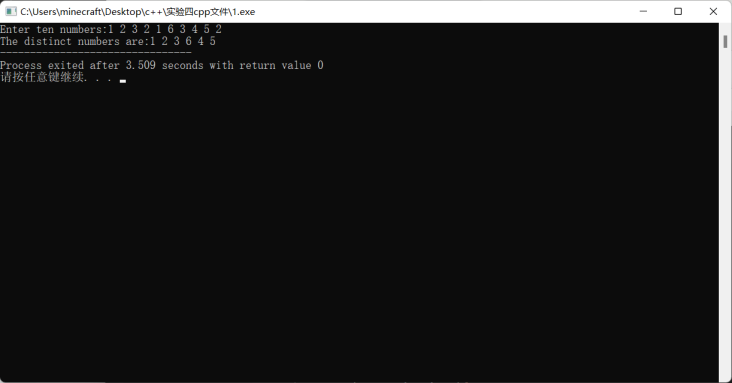
**三、算法分析，程序结果**

**数组:**

**第一题实验代码:**

1. #include <iostream>
2. using namespace std;
3. int main(){
4. int N=10;
5. int a[N];
6. cout<<"Enter ten numbers:";
7. for(int i=0;i<N;i++){
8. cin>>a[i];
9. bool flag=true;
10. for(int j=0;j<i;j++){
11. if(a[i]==a[j]){
12. flag=false;
13. }
14. }
15. if(flag==false){
16. i--;
17. N--;
18. }
19. }
20. cout<<"The distinct numbers are:";
21. for(int i=0;i<N;i++){
22. cout<<a[i]<<" ";
23. }
24. return 0;
25. }

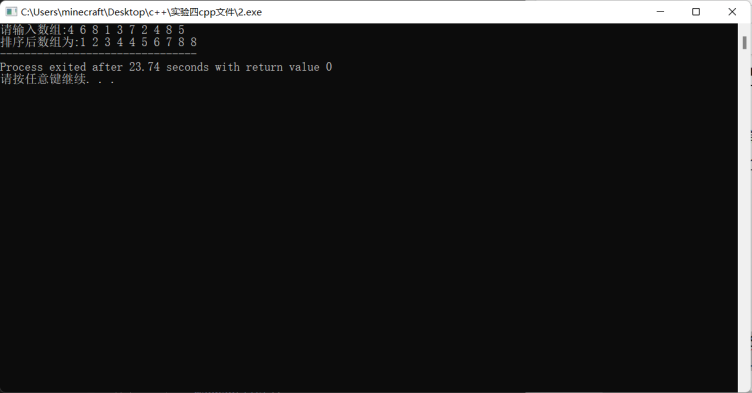
**实验结果:**

****

**第二题实验代码:**

1. #include <iostream>
2. using namespace std;
3. void maopao(double list[]){
4. bool changed=true;
5. int N=10;
6. do
7. {
8. changed = false;
9. for (int j = 0; j < N-1; j++)
10. if (list[j] > list[j+1])
11. {
12. double t=list[j];
13. list[j]=list[j+1];
14. list[j+1]=t;
15. changed = true;
16. }
17. } while (changed);
18. }
19. int main(){
20. int N=10;
21. double list[N];
22. cout<<"请输入数组:";
23. for(int i=0;i<N;i++){
24. cin>>list[i];
25. }
26. for(int i=0;i<N-1;i++){
27. maopao(list);
28. }
29. cout<<"排序后数组为:";
30. for(int i=0;i<N;i++){
31. cout<<list[i]<<" ";
32. }
33. return 0;
34. }

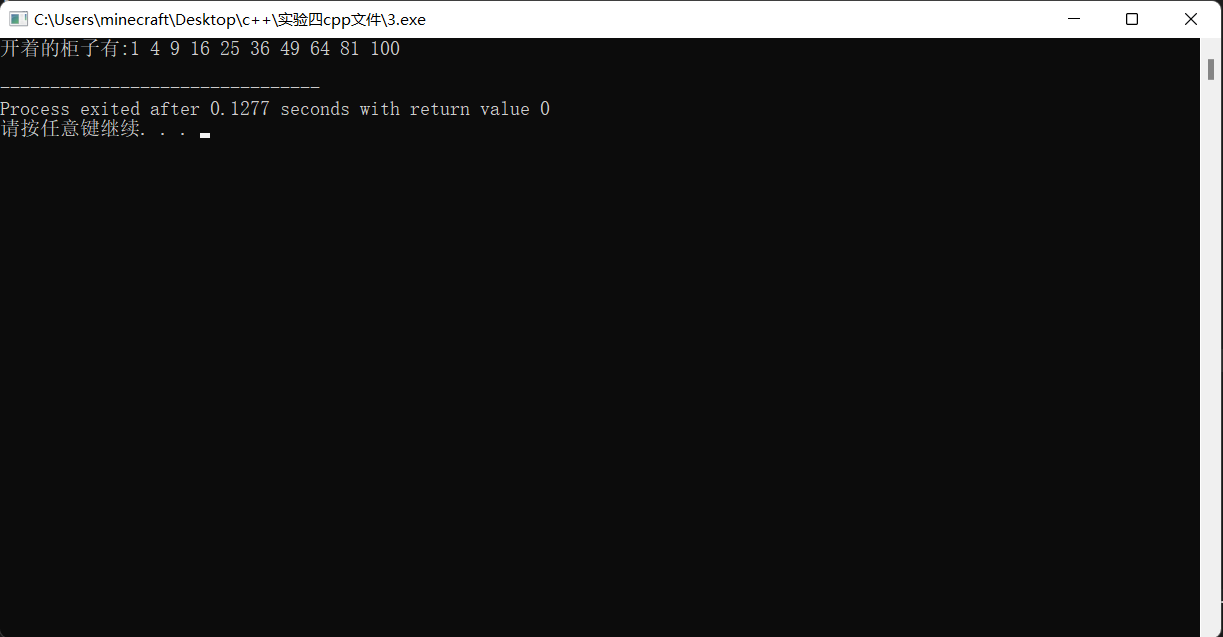
**实验结果:**

****

**第三题实验代码:**

1. #include <iostream>
2. using namespace std;
3. void change(bool desk[],int num){
4. if(desk[num]==true){
5. desk[num]=false;
6. }
7. else{
8. desk[num]=true;
9. }
10. }
11. int main(){
12. const int N=100;
13. bool desk[N]={false};
14. for(int i=1;i<=100;i++){
15. for(int j=1;j<=100;j++){
16. if(j%i==0){
17. change(desk,j-1);
18. }
19. }
20. }
21. cout<<"开着的柜子有:";
22. for(int i=0;i<100;i++){
23. if(desk[i]==true){
24. cout<<i+1<<" ";
25. }
26. }
27. cout<<"\n";
28. return 0;
30. }

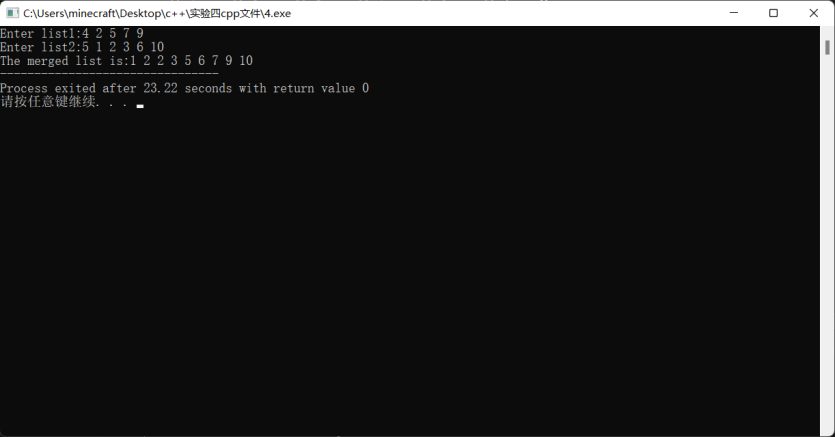
**实验结果:**

****

**第四题实验代码:**

1. #include <iostream>
2. using namespace std;
3. void merge(const int list1[], int size1, const int list2[], int size2, int list3[]){
4. int i=0,j=0,k=0;
5. while(i<size1&&j<size2){
6. if(list1[i]<list2[j]){
7. list3[k]=list1[i];
8. i++;
9. }
10. else{
11. list3[k]=list2[j];
12. j++;
13. }
14. k++;
15. }
16. if(i==size1){
17. for(int m=j;m<size2;m++){
18. list3[k]=list2[m];
19. k++;
20. }
21. }
22. if(j==size2){
23. for(int m=i;m<size1;m++){
24. list3[k]=list1[m];
25. k++;
26. }
27. }
28. }
29. int main(){
30. const int Maxsize=80;
31. int size1,size2;
32. cout<<"Enter list1:";
33. cin>>size1;
34. int list1[Maxsize],list2[Maxsize];
35. for(int i=0;i<size1;i++){
36. cin>>list1[i];
37. }
38. cout<<"Enter list2:";
39. cin>>size2;
40. for(int i=0;i<size2;i++){
41. cin>>list2[i];
42. }
43. int list3[size1+size2];
44. merge(list1,size1,list2,size2,list3);
45. cout<<"The merged list is:";
46. for(int i=0;i<size1+size2;i++){
47. cout<<list3[i]<<" ";
48. }
49. return 0;
50. }

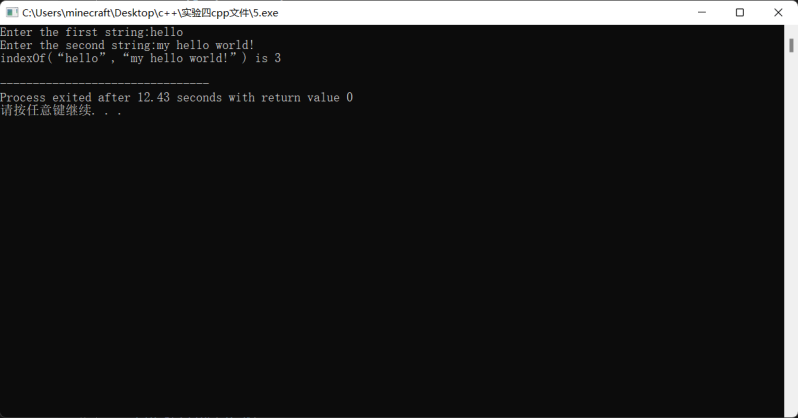
**实验结果:**

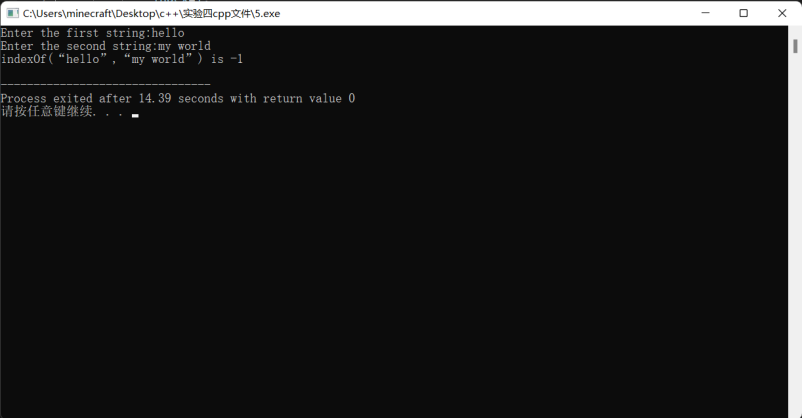
****

**第五题实验代码:**

1. #include <iostream>
2. using namespace std;
3. int indexOf(const char s1[], const char s2[]){
4. int i=0,j=0,index=-1;
5. while(s2[i]!='\0'){
6. if(s2[i]==s1[j]){
7. index=i;
8. int m=index;
9. while(s2[m]!='\0'&&s1[j]!='\0'){
10. if(s2[m++]==s1[j++]){
11. m++;
12. j++;
13. }
14. else{
15. break;
16. }
17. }
18. }
19. if(s1[j]=='\0'){
20. return index;
21. }
22. i++;
23. }
24. return -1;
25. }
26. int main(){
27. const int N=100;
28. char s1[N],s2[N];
29. cout<<"Enter the first string:";
30. cin.getline(s1,N);
31. cout<<"Enter the second string:";
32. cin.getline(s2,N);
33. int index=indexOf(s1,s2);
34. cout<<"indexOf(“"<<s1<<"”,“"<<s2<<"”) is ";
35. cout<<index<<"\n";
36. return 0;
37. }

**实验结果:**

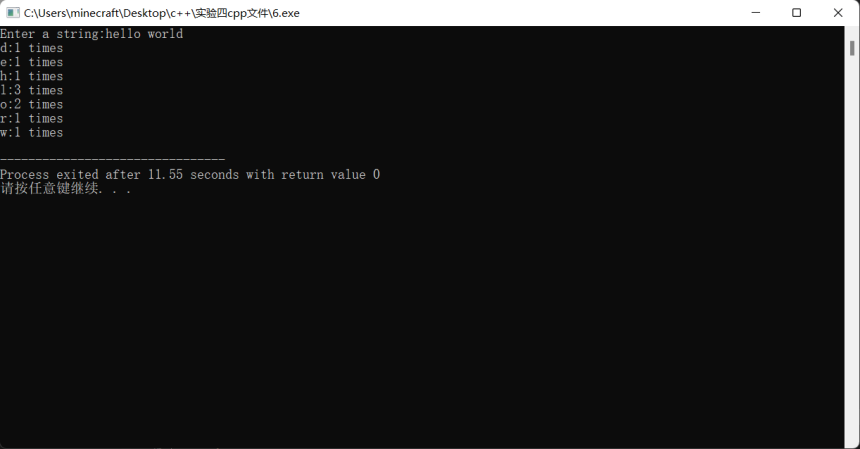
****

****

**第六题实验代码:**

1. #include <iostream>
2. using namespace std;
3. void count(const char s[], int counts[]){
4. int i=0;
5. while(s[i]!='\0'){
6. if(s[i]>='a'&&s[i]<='z'){
7. counts[s[i]-'a']++;
8. }
9. else if(s[i]>='A'&&s[i]<='Z'){
10. counts[s[i]-'A']++;
11. }
12. i++;
13. }
14. }
15. int main(){
16. const int N=100;
17. char s[N];
18. int counts[26]={0};
19. cout<<"Enter a string:";
20. cin.getline(s,N);
21. count(s,counts);
22. for(int i=0;i<26;i++){
23. if(counts[i]!=0){
24. cout<<char('a'+i)<<":"<<counts[i]<<" times"<<"\n";
25. }
26. }
27. return 0;
28. }

**实验结果:**

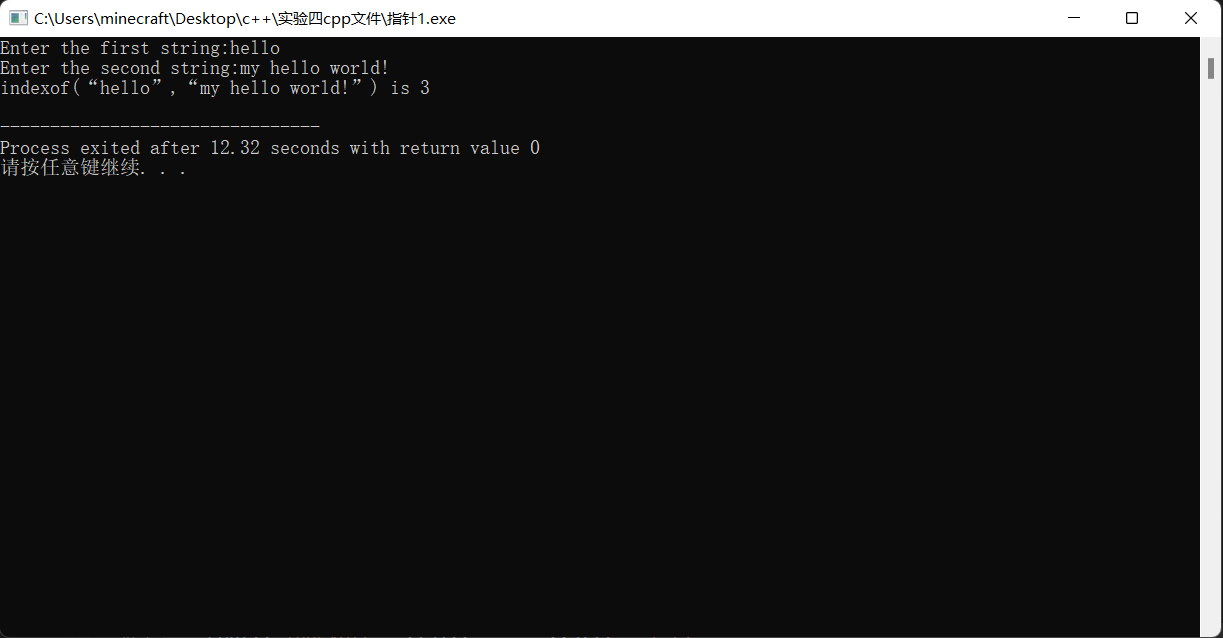
****

**指针:**

**第一题实验代码:**

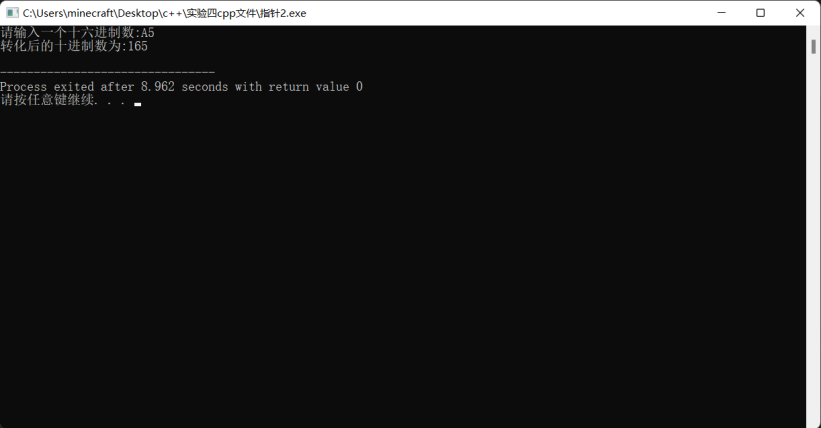
1. #include <iostream>
2. using namespace std;
3. int indexof(const char \*s1,const char \*s2){
4. int index=-1;
5. while(\*s2!='\0'){
6. if(\*s2==\*s1){
7. while(\*(s2+1)!='\0'&&\*(s1+1)!='\0'){
8. if(\*(s2+1)==\*(s1+1)){
9. s1++;
10. s2++;
11. }
12. else{
13. break;
14. }
15. }
16. }
17. if(\*(s1+1)=='\0'){
18. return (index+1);
19. }
20. s2++;
21. index++;
22. }
23. return -1;
24. }
25. int main(){
26. const int N=100;
27. char s1[N],s2[N];
28. cout<<"Enter the first string:";
29. cin.getline(s1,N);
30. cout<<"Enter the second string:";
31. cin.getline(s2,N);
32. char\*p1,\*p2;
33. p1=s1;
34. p2=s2;
35. int index=indexof(p1,p2);
36. cout<<"indexof(“"<<s1<<"”,“"<<s2<<"”) is ";
37. cout<<index<<"\n";
38. return 0;
40. }

**实验结果:**

****

**第二题实验代码:**

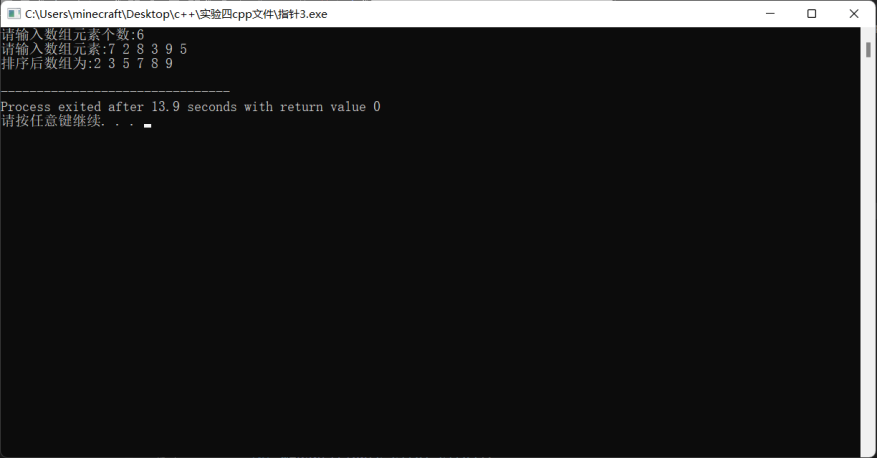
1. #include <iostream>
2. #include <cmath>
3. using namespace std;
4. int parseHex(const char \*const hexString){
5. int dec=0,size1=0;
6. while(hexString[size1]!='\0'){
7. size1++;
8. }
9. for(int i=0;i<size1;i++){
10. if(hexString[size1-i-1]>='A'&&hexString[size1-i-1]<='F'){
11. dec+=(hexString[size1-i-1]-'A'+10)\*pow(16,i);
12. }
13. else if(hexString[size1-i-1]>='0'&&hexString[size1-i-1]<='9'){
14. dec+=(hexString[size1-i-1]-'0')\*pow(16,i);
15. }
16. else{
17. return 0;
18. }
19. }
20. return dec;
21. }
22. int main(){
23. const int N=100;
24. char hexString[N];
25. cout<<"请输入一个十六进制数:";
26. cin.getline(hexString,N);
27. if(parseHex(hexString)){
28. cout<<"转化后的十进制数为:"<<parseHex(hexString)<<"\n";
29. }
30. else{
31. cout<<"这不是一个正确的十六进制数。"<<"\n";
32. }
33. return 0;
35. }

****

**第三题实验代码：**

1. #include <iostream>
2. using namespace std;
3. int Partition(int a[],int site1,int site2){
4. int i=site1,j=site2;
5. int x=a[site1];
6. while(true){
7. while(a[i]<=x&&i<site2){
8. i++;
9. }
10. while(a[j]>x){
11. j--;
12. }
13. if(i>=j){
14. break;
15. }
16. int k=a[i];
17. a[i]=a[j];
18. a[j]=k;
19. }
20. a[site1]=a[j];
21. a[j]=x;
22. return j;
23. }
24. void sort(int s[],int site1,int site2){
25. if(site1<site2){
26. int m=Partition(s,site1,site2);
27. sort(s,site1,m-1);
28. sort(s,m+1,site2);
30. }
31. }
32. int main(){
33. int N;
34. cout<<"请输入数组元素个数:";
35. cin>>N;
36. int\*p=new int[N];
37. cout<<"请输入数组元素:";
38. for(int i=0;i<N;i++){
39. cin>>p[i];
40. }
41. sort(p,0,N-1);
42. cout<<"排序后数组为:";
43. for (int i = 0; i < N; i++) {
44. cout << \*(p + i) << " ";
45. }
46. cout<<"\n";
47. delete[] p;
48. return 0;
50. }

**实验结果:**

****

**四、遇到的问题与解决方法**

**在这次实验中，解决了一些有关数组和指针相关的问题，在解决这些问题时也遇到了一些相关的问题，首先是在遇到以数组或者数组指针作为函数参数进行对数组进行相关的处理时，很多时候都会使用到数组的长度这一数量，而当该参数没有通过参数的形式从主函数中传递过来时，则需要在数组中先得到数组的长度，开始的时候我不是特别了解怎么得到数组的长度，现在看来就是没有弄清楚数组和数组指针的关系，在了解了数组名就是数组的一个常量指针时，后来就可以通过一个while循环来遍历数组中的元素，并计数，就可以得到数组的长度，然后进行相关的计算了。然后就没有遇到什么问题了。**

**五、体会**

通过这次有关数组和指针的实验，使我对数组和指针以及数组指针有了更加清楚的了解，

并且能够运用其解决相关的问题，让学到的知识成为具体的解决问题的能力，并且锻炼了遇到不懂的问题冷静处理的能力，下次遇到问题时能够更好的处理和解决，希望在接下来的实验中继续努力，努力将学到的知识能够更好地应用出来。