**计算机程序设计基础（C++)**

**实验报告**

专业班级： 软工2401

学 号： 8209240115

姓 名： 马一平

**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？

函数的返回类型定义了函数期望返回的值的类型。例如，如果一个函数声明为 bool is\_prime(int num)，它期望返回一个布尔值。返回的值类型必须与声明的返回类型一致，否则会发生类型不匹配的错误。

2. 本实验中主函数调用函数时采用的是何种传递方式？

实验1中调用函数采用的是引用传递，以便返回多个结果。，实验2、4、5中采用的是值传递。

**四、算法分析，程序结果**

1、#include <iostream>

using namespace std;

int gys(int a, int b)

{

    while (b != 0)

    {

        int temp = b;

        b = a % b;

        a = temp;

    }

    return a;

}

void gbs(int a, int b, int &r2)

{

    r2 = a \* b / gys(a, b);

}

int main()

{

    int m, n, r1, r2;

    cin >> m >> n;

    r1 = gys(m, n);

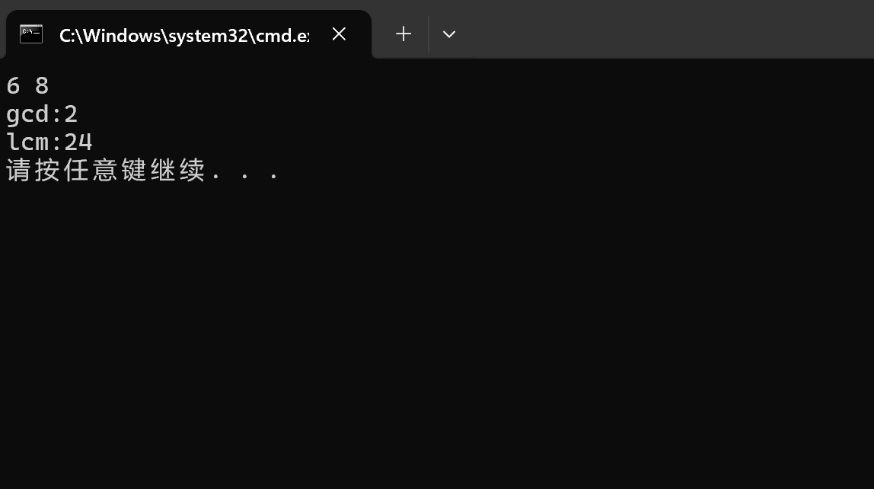
    gbs(m, n, r2);

    cout << "gcd:" << r1 << endl;

    cout << "lcm:" << r2 << endl;

    system("pause");

    return 0;

}

2、

#include <iostream>

using namespace std;

bool is\_prime(int num){

    int j=0;

    for(int i=2;i<=num/2;i++){

        if(num%i==0){

            j++;

        }

    }

    if(j>0){

        return false;

    }

    else{

        return true;

    }

}

int main() {

    int k=0;

    for (int i = 2; i <= 100; i++) {

        if (is\_prime(i)) {

            cout << i << " ";

            k++;

            if(k%10==0){

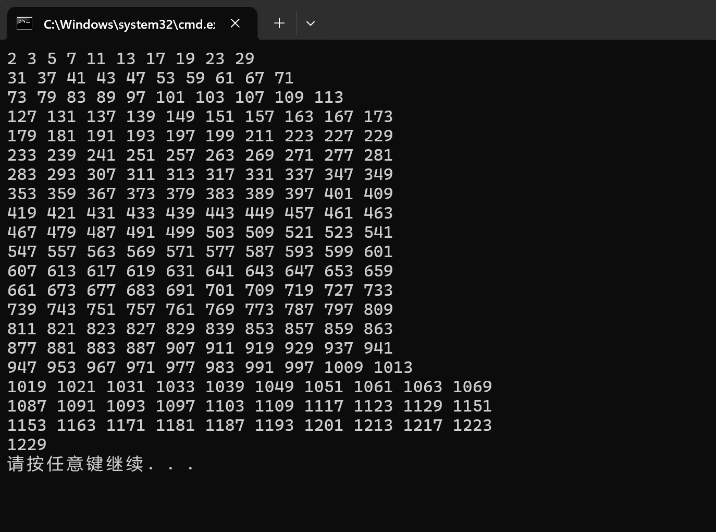
                cout << endl;

            }

        }

    }

    return 0;

}

4、

**Mytriangle.h:**

#include <iostream>

using namespace std;

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

double area(double side1,double side2, double side3);

**Mytriangle.cpp:**

#include <iostream>

#include <cmath>

using namespace std;

#include "mytriangle.h"

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

    if(side1+side2>side3 && side1+side3>side2 && side2+side3>side1){

        return true;

    }

    else{

        return false;

    }

}

double area(double side1,double side2, double side3){

    double s = (side1+side2+side3)/2;

    return sqrt(s\*(s-side1)\*(s-side2)\*(s-side3));

}

int main(){

    double side1,side2,side3;

    cin >> side1 >> side2 >> side3;

    if(is\_valid(side1,side2,side3)){

        cout << area(side1,side2,side3) << endl;

    }

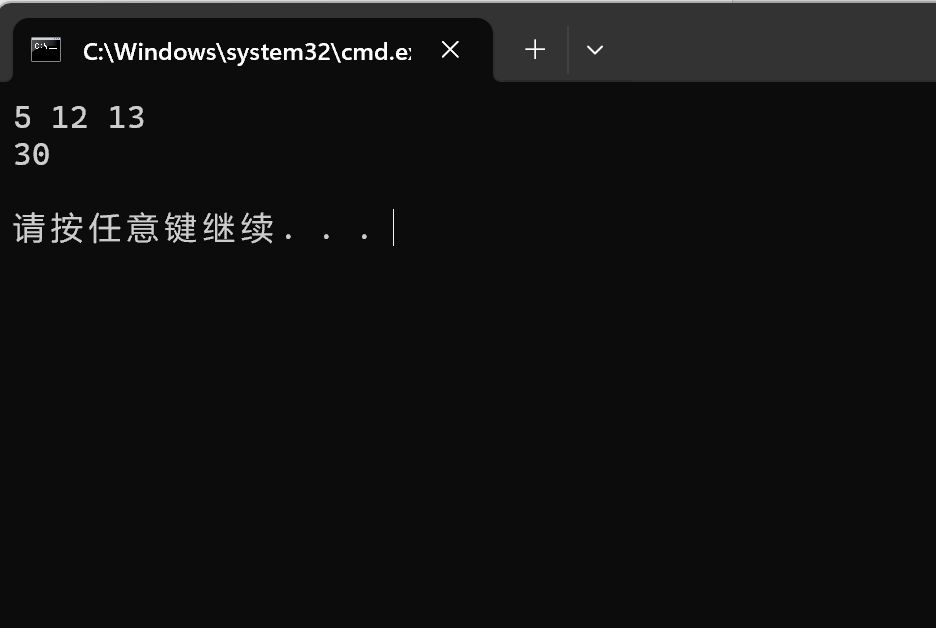
    else{

        cout << "无法构成三角形" << endl;

    }

    return 0;

}



5、

#include <iostream>

using namespace std;

int count(int num,int day){

    if(day>0){

        num=(num+1)\*2;

        return count(num,day-1);

    }

    else{

        return num;

    }

}

int main() {

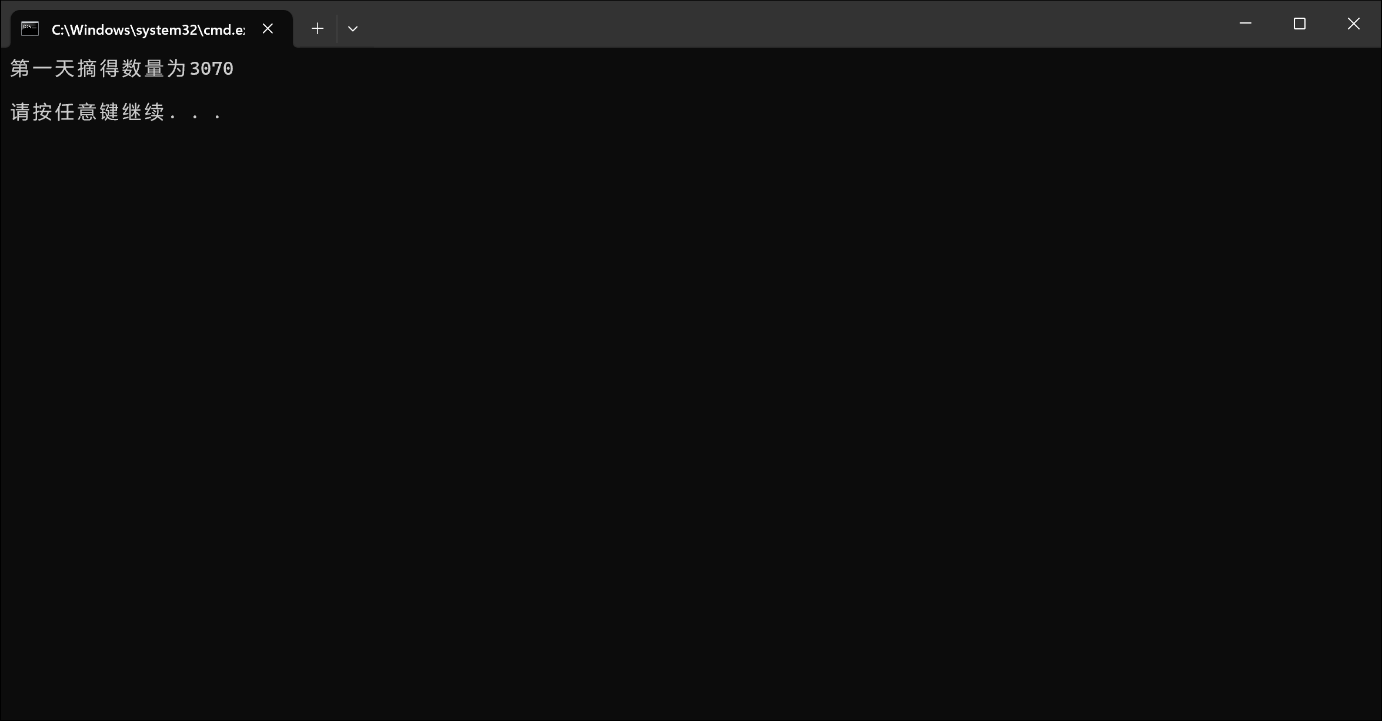
    int x=1;

    int day=10;

    cout << "第一天摘得数量为" << count(x,day) << endl;

    return 0;

}



**五、遇到的问题与解决方法**

1、在写问题五中不太清楚该怎么写一个递归算法，通过搜索了解到，一个递归函数通常包括以下两个部分：

基准情况（Base Case）：递归终止的条件，防止递归一直进行下去，通常是最简单的情况，直接返回结果。

递归情况（Recursive Case）：将问题分解为更小的子问题，并通过递归调用函数来解决。

所以选用if-else结构来构建基准情况和递归情况：

**六、体会**

**1. 函数的定义和调用**

在这次实验中，通过多个函数实例更深入地理解了C++中函数的定义、声明与调用。尤其是通过将函数声明和定义分离，代码结构变得更加清晰。实际编程中，通过合理使用函数来拆解复杂的问题，不仅能够提升代码的可读性，还能减少重复代码，使得程序更加模块化，易于维护。

**2. 参数传递方式：值传递与引用传递**

这次实验让我特别注意到值传递和引用传递的区别。值传递会将参数的副本传入函数，因此函数内部的修改不会影响原始数据。而引用传递则将参数的引用传入函数，使得函数能够直接修改原始数据。通过比较这两种方式，我发现：值传递适合那些只需要读取数据的情况，能够保证数据的安全性；引用传递则更适合需要修改数据的场景，能够提高效率，避免不必要的内存开销。

（例如，在求最大公约数和最小公倍数时，我使用了引用传递来返回多个结果，这样不仅避免了多余的返回值处理，还提高了程序的性能。）

**3. 多文件编程**

通过本次实验，我认识到了多文件编程的优势。将函数的声明和定义分开，可以使代码更加清晰，易于管理和扩展。通过头文件引入函数声明，在主程序中调用函数定义，这种做法不仅优化了代码结构，还方便了不同模块间的协作，提高了程序的可维护性。

**4. 递归算法的应用**

猴子吃桃子的问题，进一步加深了我对递归算法的理解。递归是一种通过函数自我调用解决问题的方式，特别适用于具有重复结构的算法。递归的关键是设计合适的基准条件，并确保递归能够正确地终止。

**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;

}

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 “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\_\_\_\_\_\_\_\_

(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[]={1,2,3,4};

return list;

}

void main()

{

int \*p=f();

cout<<p[0]<<endl;

cout<<p[1]<<endl;

}

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**

**（一）数组**

**1、**

#include <iostream>

#include <algorithm>

#include <iterator>

using namespace std;

int main() {

    cout << "Enter 10 numbers:" << endl;

    int num[10];

    int x;

    int k=0;

    for(int i=0; i<10; i++) {

        cin >> x;

        if(find(begin(num), end(num), x) == end(num)) {

            num[k] = x;

            k++;

        }

    }

    sort(begin(num), begin(num)+k);

    cout << "The distinct numbers are:";

    for(int i=0; i<k; i++) {

        cout << num[i] << " ";

    }

    return 0;

}

2、

#include <iostream>

#include <algorithm>

using namespace std;

int main() {

    int listSize=10;

    int list[listSize];

    for(int i=0;i<listSize;i++){

        cin>>list[i];

    }

    bool changed = true;

    do

    {

    changed = false;

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

        if (list[j] > list[j+1])

        {

        swap(list[j],list[j+1]);

        changed = true;

        }

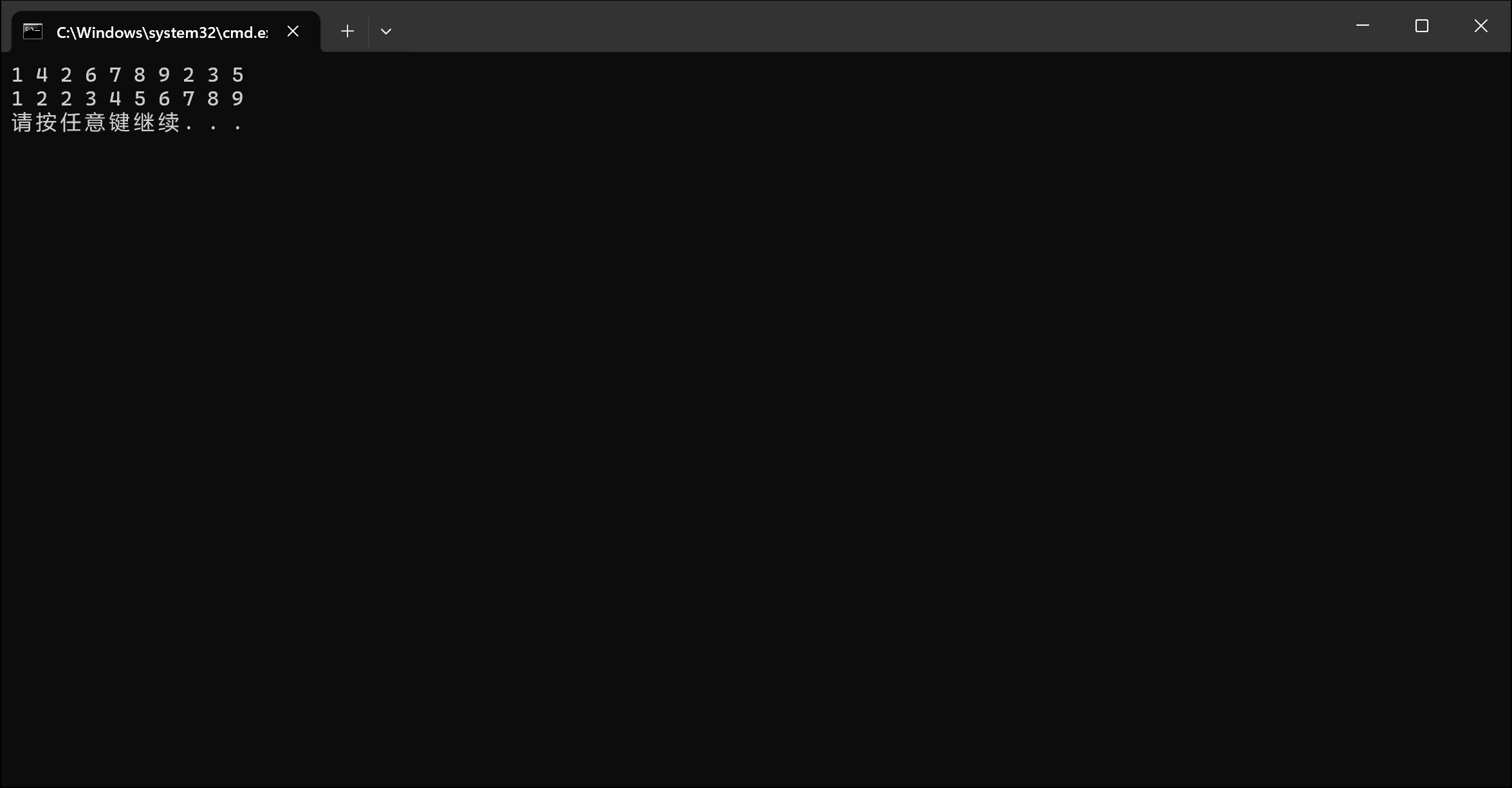
    } while (changed);

    for(int i=0;i<listSize;i++){

        cout<<list[i]<<" ";

    }

    return 0;

}

3、

#include <iostream>

using namespace std;

void switch1(bool arr[],int n){

    for(int i=n-1;i<100;i=i+n+1){

        arr[i]=!arr[i];

    }

}

int main() {

    bool arr[100];

    for(int i=0; i<100; i++){

        arr[i]=true;

    }

    for(int i=2; i<=100; i++){

        switch1(arr,i);

    }

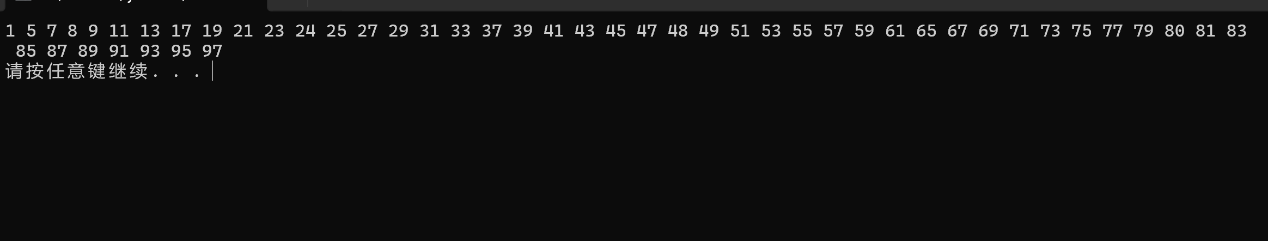
    for(int i=0; i<100; i++){

        if(arr[i]){

            cout<<i+1<<" ";

        }

    }

}

4、

#include <iostream>

using namespace std;

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

    int i = 0, j = 0, k = 0;

    while (i < size1 && j < size2) {

        if (list1[i] < list2[j]) {

            list3[k] = list1[i];

            i++;

        }

        else {

            list3[k] = list2[j];

            j++;

        }

        k++;

    }

    while (i < size1) {

        list3[k] = list1[i];

        i++;

        k++;

    }

    while (j < size2) {

        list3[k] = list2[j];

        j++;

        k++;

    }

}

int main() {

    int size1,size2;

    cout << "Enter list1:";

    cin >> size1;

    int \*list1 = new int[size1];

    for (int i = 0; i < size1; i++) {

        cin >> list1[i];

    }

    cout << "Enter list2:";

    cin >> size2;

    int \*list2 = new int[size2];

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

        cin >> list2[i];

    }

    int \*list3 = new int[size1 + size2];

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

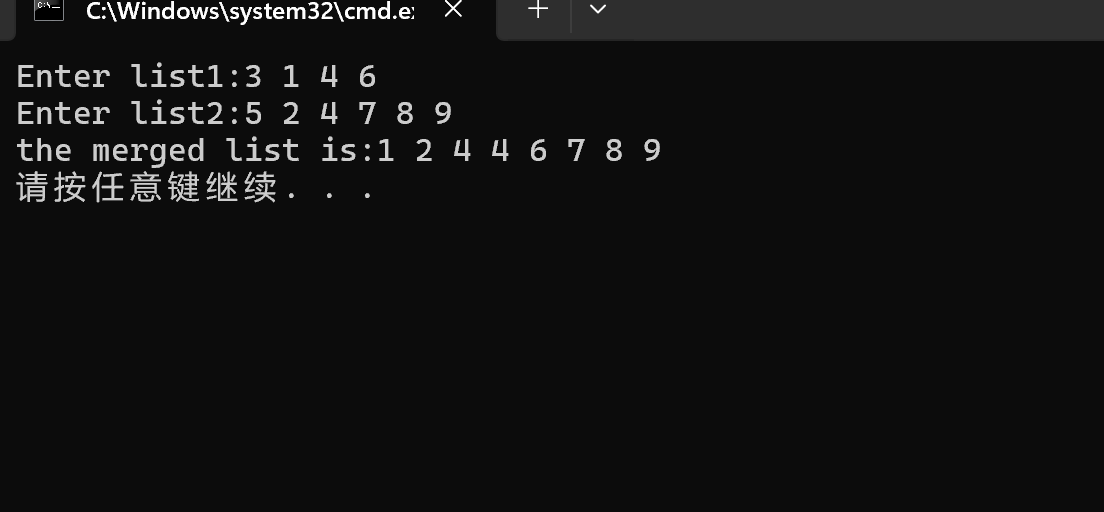
    cout << "the merged list is:";

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

        cout << list3[i] << " ";

    }

    return 0;

}

5、

#include <iostream>

using namespace std;

#include <cstring>

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

    int len1=strlen(s1);

    int len2=strlen(s2);

    for (int i = 0; i <= len2 - len1; i++) {

        bool match = true;

        for (int j = 0; j < len1; j++) {

            if (s2[i + j] != s1[j]) {

                match = false;

                break;

            }

        }

        if (match) {

            return i;

        }

    }

    return -1;

}

int main() {

    char str1[100], str2[100];

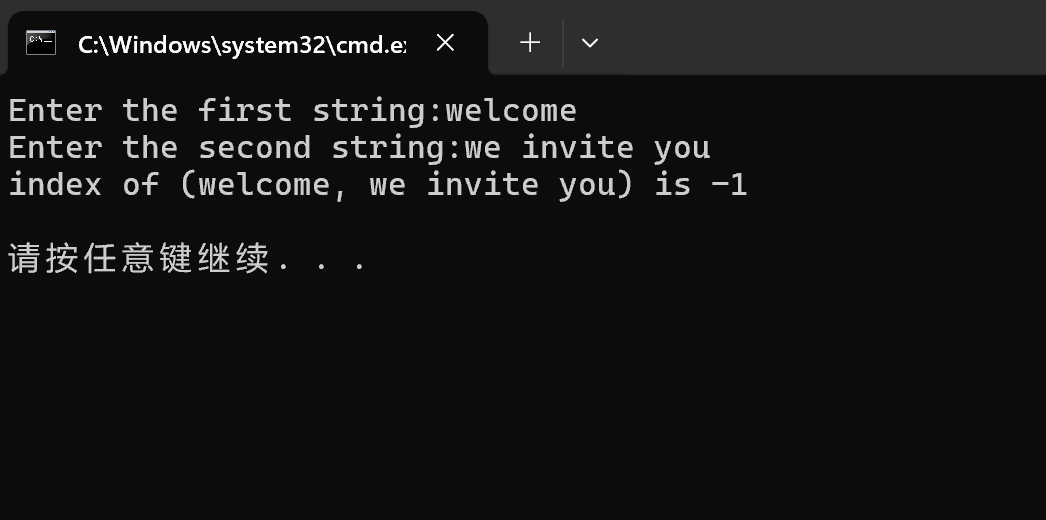
    cout << "Enter the first string:";

    cin.getline(str1, 100);

    cout << "Enter the second string:";

    cin.getline(str2, 100);

    cout << "index of (" << str1 << ", " << str2 << ") is " << indexOf(str1, str2) << endl;

}

6、

#include <iostream>

using namespace std;

#include <cstring>

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

    int len=strlen(s);

    for(int i=0;i<len;i++){

        if(s[i]>='a' && s[i]<='z'){

            counts[int(s[i])-97]++;

        }

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

            counts[int(s[i])-65]++;

        }

    }

}

int main(){

    char s[100];

    cout << "Enter a string:";

    cin.getline(s,100);

    int counts[26]={0};

    count(s,counts);

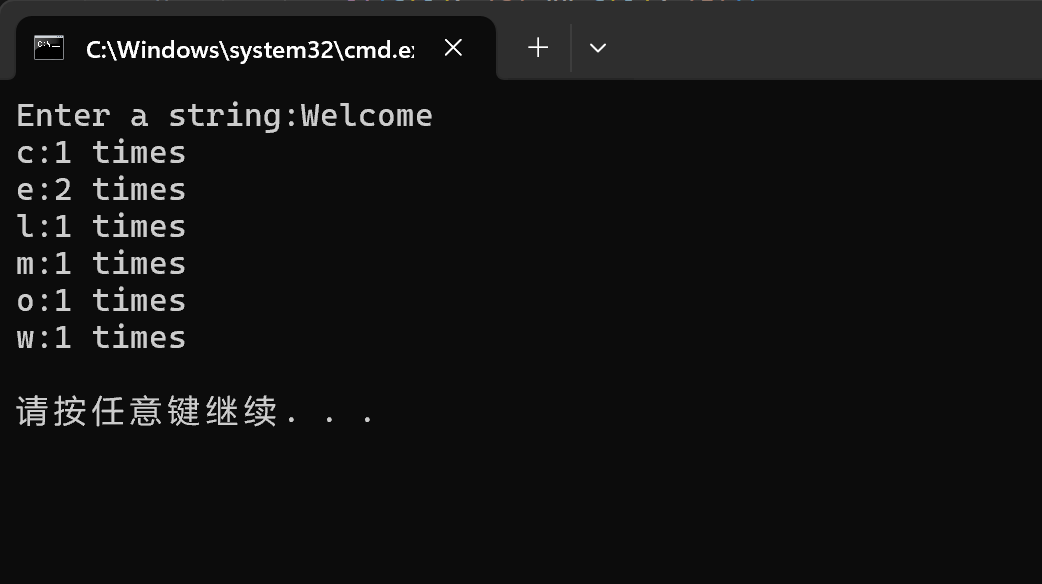
    for(int i=0;i<26;i++){

        if(counts[i]!=0){

            cout << char(97+i) << ':' << counts[i] << " times" << endl;

        }

    }

}

（二）指针

1、

#include <iostream>

using namespace std;

#include <cstring>

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

    int len1=strlen(s1);

    int len2=strlen(s2);

    for (int i = 0; i <= len2 - len1; i++) {

        bool match = true;

        for (int j = 0; j < len1; j++) {

            if (s2[i + j] != s1[j]) {

                match = false;

                break;

            }

        }

        if (match) {

            return i;

        }

    }

    return -1;

}

int main() {

    char str1[100], str2[100];

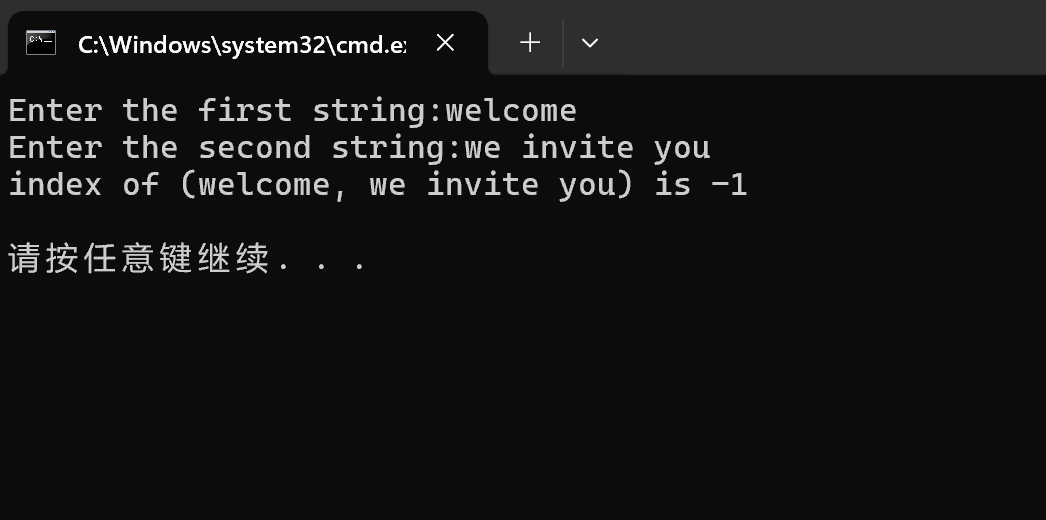
    cout << "Enter the first string:";

    cin.getline(str1, 100);

    cout << "Enter the second string:";

    cin.getline(str2, 100);

    cout << "index of (" << str1 << ", " << str2 << ") is " << indexOf(str1, str2) << endl;

}

2、

#include <iostream>

using namespace std;

#include <cstring>

#include <cmath>

int parseHex(const char \*const hexString){

    int len=strlen(hexString);

    int result=0;

    for(int i=0;i<len;i++){

        if (hexString[i] >= '0' && hexString[i] <= '9')

        {

            result = result + (((int(hexString[i]))-48) \* pow(16, len - i - 1));

        }

        else

        {

            result = result +(((int(hexString[i])) - 55) \* pow(16, len - i - 1));

    }

    }

    return result;

}

int main(){

    char str[100];

    cin.getline(str,100);

    cout << parseHex(str) << endl;

}

3、

#include <iostream>

using namespace std;

void sort(int s[],int num){

    for (int i = 0; i < num - 1; i++)

    {

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

        {

            if (s[j] > s[j + 1])

            {

                int temp = s[j];

                s[j] = s[j + 1];

                s[j + 1] = temp;

            }

        }

    }

}

int main(){

    cout << "输入数组元素个数：";

    int num;

    cin >> num;

    int \*s=new int[num];

    cout << "输入数组元素：";

    for(int i=0;i<num;i++){

        cin >> s[i];

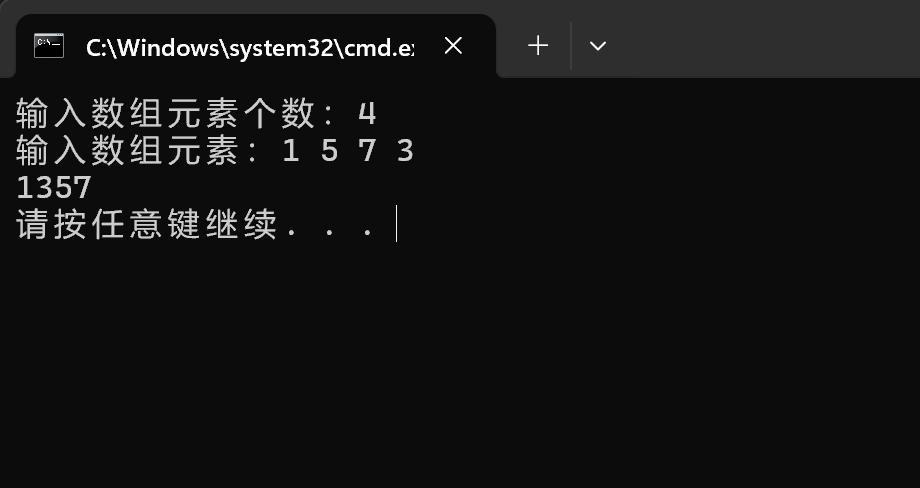
    }

    sort(s,num);

    for(int i=0;i<num;i++){

        cout << s[i];

    }

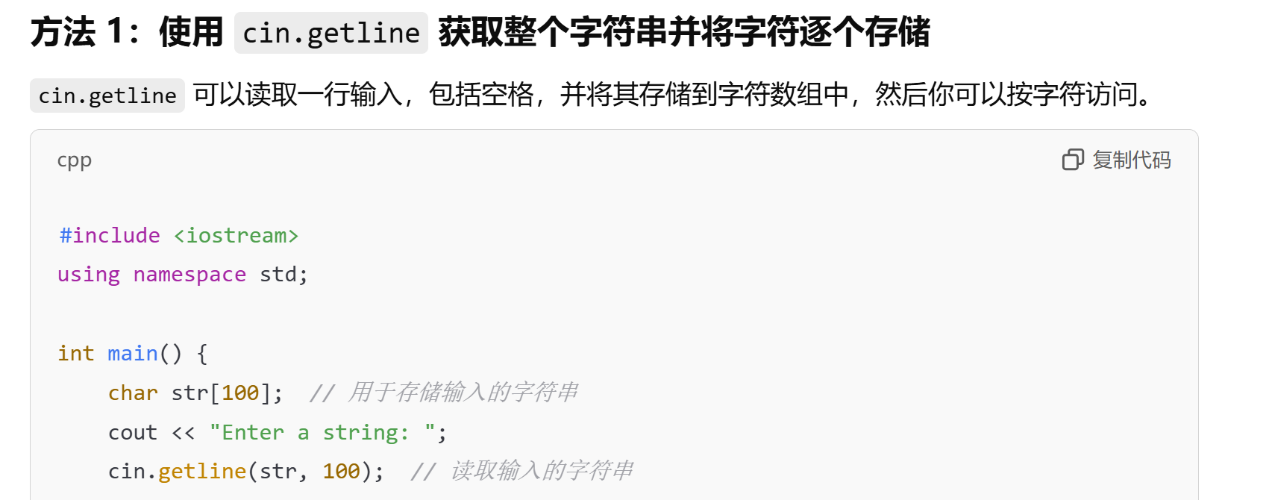
}

**四、遇到的问题与解决方法**

1、对于布尔值取反

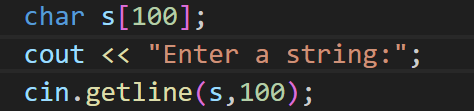
刚开始采用了if-else结构，若arr[i]为true则改为false，若为false则改为true，但觉得写起来略为复杂，通过在网上搜索发现，在C++中，取反一个 bool 值可以使用逻辑取反运算符 !故将代码改为arr[i]=!arr[i]

2、对于处理字符串输入

在写代码过程中，不知道怎么将输入的字符串以单个字符为元素储存在数组中，通过chatgpt了解到：

并且两种方法的区别为：

使用 cin.getline 可以读取整个字符串并存储在字符数组中，适合处理包含空格的输入。使用 cin.get(ch) 可以逐个读取字符，这种方式适合处理没有空格的情况。

故根据题目输入样例，要支持包含空格的输入，故采用第一种方法：

**五、体会**

**1、关于数组的基本操作与应用**

在实验中，通过“读入10个数并输出其中不同的数”和“冒泡排序”等题目，更加深入学习了数组的处理方法。在冒泡排序的实现过程中，了解了基本排序算法的原理及其实现方式，并通过循环和交换实现了升序排序。此外，存物柜问题的实验让我切身体会到了在多次循环处理和状态改变的场景中，数组强大的处理能力。

**2、关于指针的应用与操作**

通过本次实验，我进一步掌握了指针的基本概念和操作。实验中，指针不仅能直接操作数组元素，还能在函数传递中发挥重要作用。例如：

用指针可以通过表达式 \*(a + i) 直接访问数组元素，而不需要显式的数组下标 a[i]。

指针可以提高程序效率，尤其是在处理大量数据时，通过传递指针地址而非复制数据，可以有效避免内存消耗。

**3、关于递归与动态内存分配**

递归函数的调用和返回过程让我对函数调用栈知识有了更深入的理解。同时，动态内存分配使得程序在处理大数据时更加灵活，通过new创建动态数组，并通过delete进行内存释放，避免了内存的浪费和泄漏。