**计算机程序设计基础（C++)**

**实验报告**

专业班级： 软工2402

学 号： 8209240209

姓 名： 马子周

**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？

2. 本实验中主函数调用函数时采用的是何种传递方式？

**1.**

**2.值传递**

**四、算法分析，程序结果**

**1.** #include <iostream>

using namespace std;

int gys(int a,int b)//定义公因数函数

{

int c = a ? b : a < b;

while (c > 0)

{

if (a % c == 0 && b % c == 0)

break;

c--;

}

return c;

}

int gbs(int a, int b)//定义公倍数函数

{

int c = a ? b : a > b;

while (c > 0)

{

if (c % a == 0 && c % b == 0)

break;

c++;

}

return c;

}

int main()

{

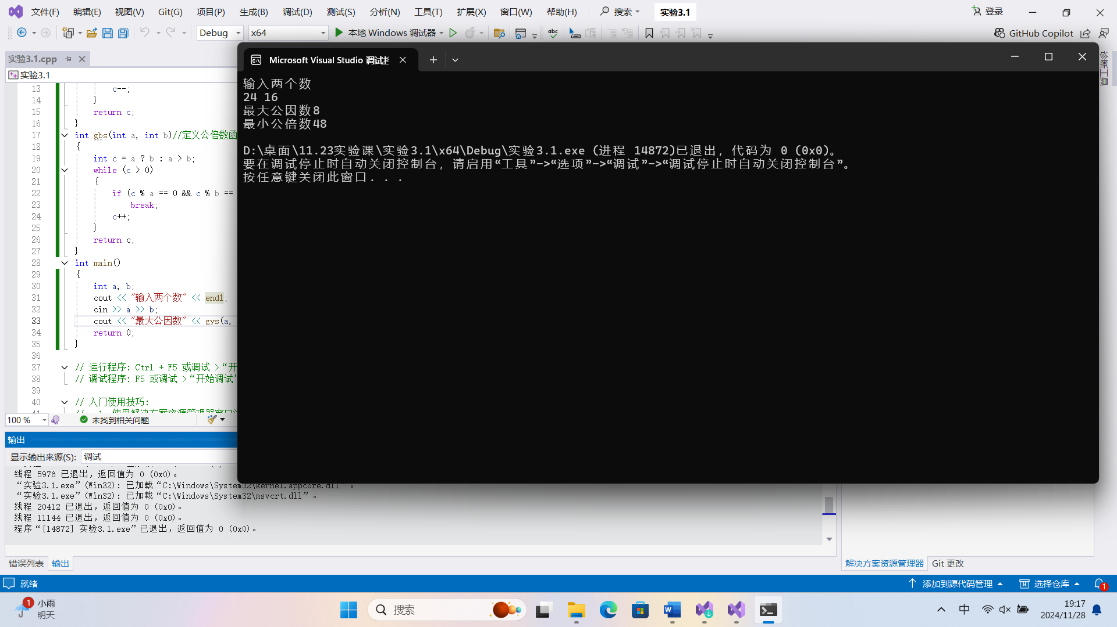
int a, b;

cout << "输入两个数" << endl;

cin >> a >> b;

cout << "最大公因数" << gys(a, b) <<endl<< "最小公倍数" << gbs(a, b) << endl;

return 0;

}****

**2.** #include <iostream>

using namespace std;

bool is\_prime(int num)

{

int a = 2;

while (a < num)

{

if (num % a == 0)

return false;

a++;

}

return true;

}

int main()

{

int a, i=0;

for (a = 2; i <= 200; a++)

{

if (is\_prime(a) == 1||a==2)

{

cout << a << " ";

i++;

}

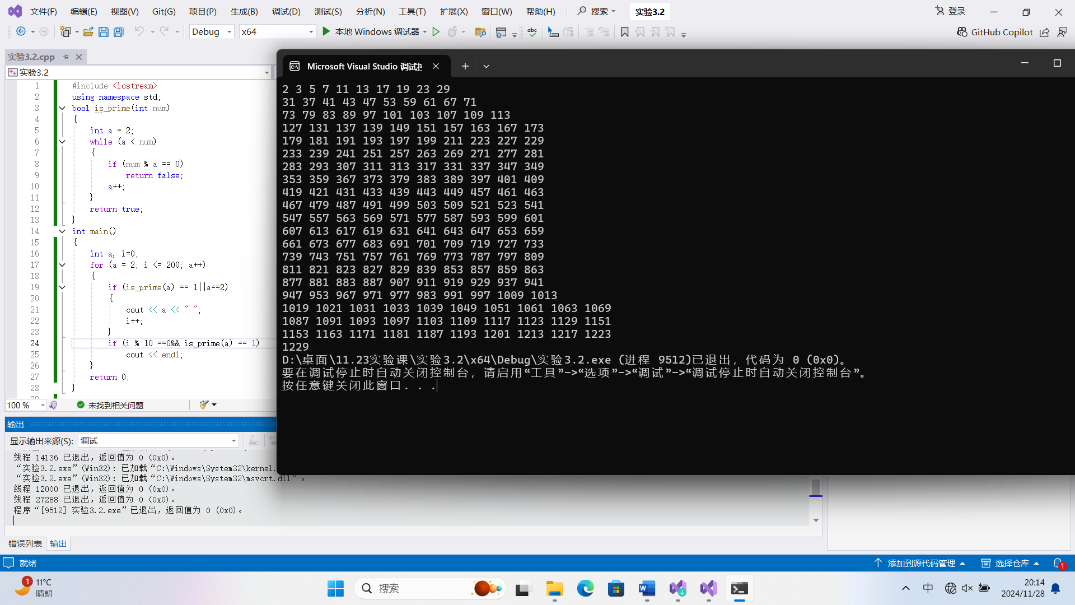
if (i % 10 ==0&& is\_prime(a) == 1)

cout << endl;

}

return 0;

}

****

**3.头文件：** **#include <iostream>**

**using namespace std;**

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

**{**

**double fah = 32 + 1.8 \* cel;**

**return fah;**

**}**

**double fahrenheit\_to\_cels(double fah)//华氏温度到摄氏温度#pragma once**

**{**

**double cel = (fah - 32) / 1.8;**

**return cel;**

**}**

**主函数：#include"mytemperature.h"**

**int main()**

**{**

**double a, c, f;**

**cout << "你想输入华氏度还是摄氏度，华氏度扣1，摄氏度扣2" << endl;**

**cin >> a;**

**if (a == 1)**

**{**

**cout << "f="<<endl;**

**cin >> f;**

**c = fahrenheit\_to\_cels(f);**

**cout << c << endl;**

**}**

**else if (a == 2)**

**{**

**cout << "c=" << endl;**

**cin >> c;**

**f = celsius\_to\_fah(c);**

**cout << f << endl;**

**}**

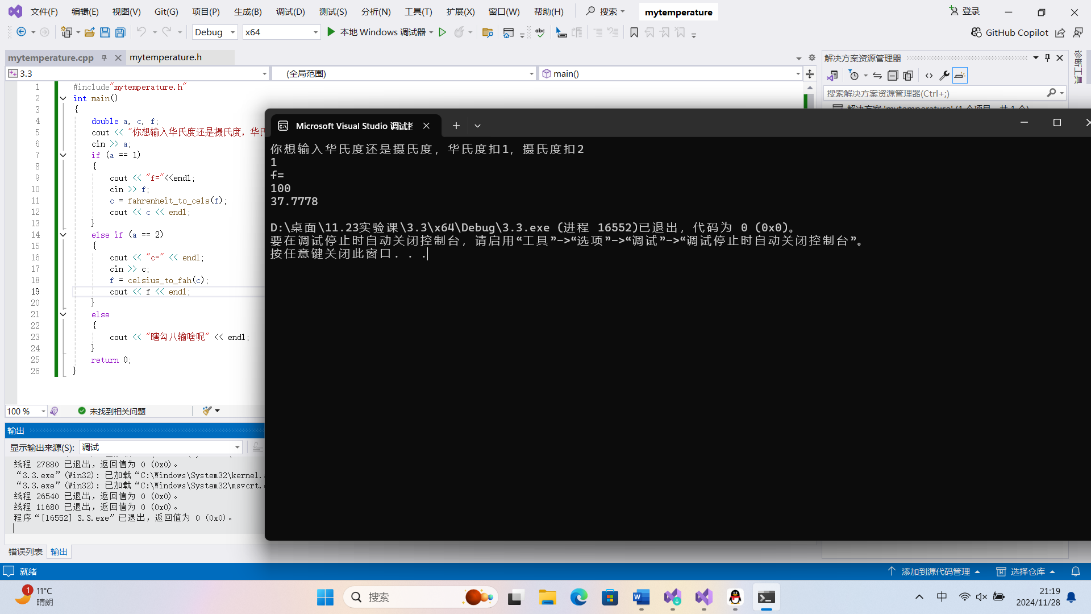
**else**

**{**

**cout << "瞎勾八输啥呢" << endl;**

**}**

**return 0;**

**}**

**5.** #include <iostream>

using namespace std;

int yesterday(int a)

{

int b = (a + 1) \* 2;

return b;

}

int main()

{

int a = 1;

for (int i = 1; i <= 10; i++)

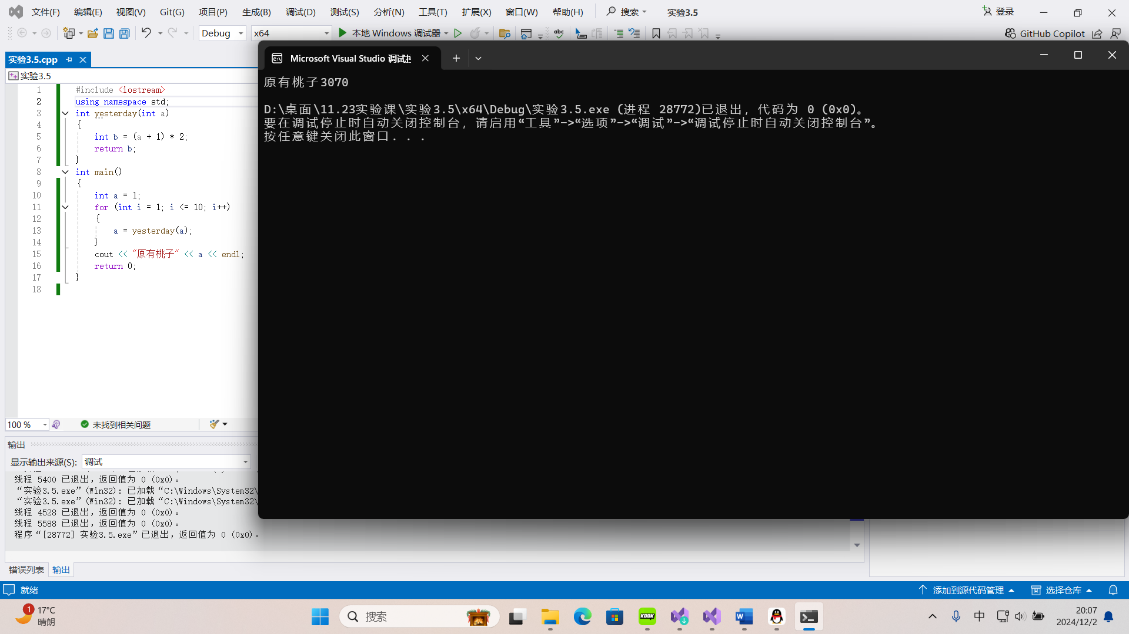
{

a = yesterday(a);

}

cout << "原有桃子" << a << endl;

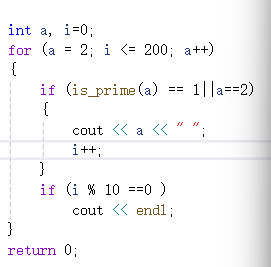
return 0;

****}

**五、遇到的问题与解决方法**

**3.2时每十个换行一开始出了错，i在被十整除会循环多次，不可单独输出换行。之后补充了a也为质数后成功输出。**

**3.3摄氏度与华氏度的换算忘记了，上网查询。**

**六、体会**

**在本次 C++函数章节的上机实践中，我收获颇丰。**

**函数的运用让程序的结构变得更加清晰。在处理复杂问题时，通过将不同功能封装成函数，使得代码的逻辑层次分明，易于理解和维护。比如在编写一个包含多种数学运算的程序时，把加法、减法、乘法等运算分别定义为独立的函数，主函数中只需调用这些函数，降低了代码的混乱度。**

**在函数的参数传递和返回值设置方面，我也有了深刻体会。合理选择参数类型和传递方式，能够准确地在函数间传递数据。而返回值则能将函数的运算结果反馈给调用者，使函数具有更强的功能性和复用性。**

**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\_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[]={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**

**4.1.1**#include <iostream>

using namespace std;

int main()

{

int a[10];

int temp[10];

int b=0;

int t;

cout << "输入十个数" << endl;

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

{

cin >> temp[i];

t = 0;

for (int j = 0; j < i; j++)//检查有无相同的数字

{

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

t++;

}

if (t == 0)

{

a[b] = temp[i];

b++;

}

}

cout << "还剩下" << endl;

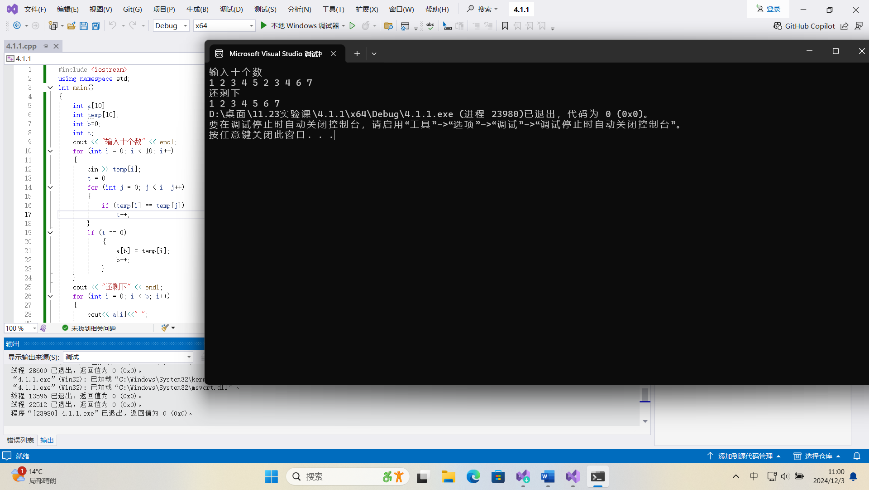
for (int i = 0; i < b; i++)

{

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

}

return 0;

}

**4.1.2** #include <iostream>

using namespace std;

void bubbleSort(int\* arr)

{

for (int i = 0; i < 9; i++)//比较轮数:比较一次 可以比出 一个最大值 要将10个数排序 仅需比出9次(10-1)最大值 即可确定排序

{

for (int j = 0; j < 9 - i; j++)//比较的数字:第一次要10个数前后比较 比较九回 选出最大值 第二次要选出次大值，以此类推

{

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

{

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

int main()

{

int arr[10];

cout << "输入10个数"<<endl;

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

{

cin >> arr[i];

}

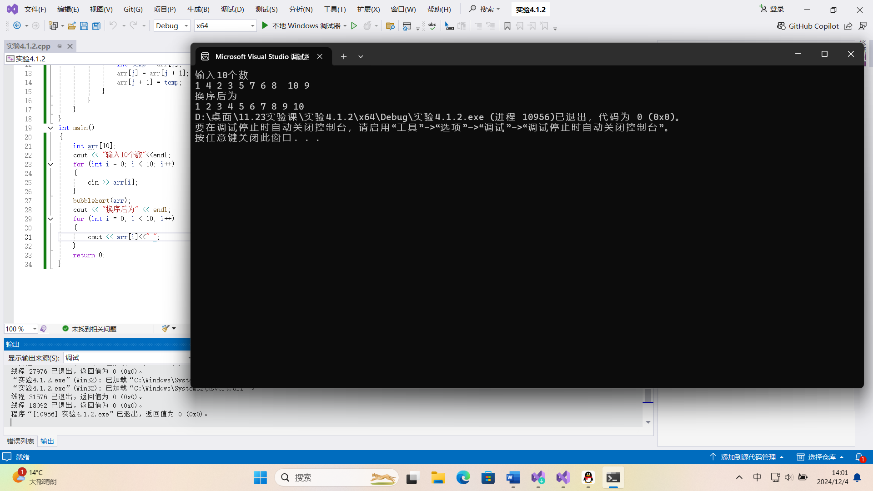
bubbleSort(arr);

cout << "换序后为" << endl;

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

{

cout << arr[i]<<" ";

 }

return 0;

}

**4.1.3** #include <iostream>

using namespace std;

void change(bool\* arr, int a)

{

for (int i = a; i <= 99; i = i + a + 2)//注意数组的编号小一位，如2号柜子实际为arr[1]，此时i=1,而间隔为3

{

arr[i] = (!arr[i]);

}

}

int main()

{

bool arr1[100];

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

{

arr1[i] = true;

}

for (int i = 1; i <= 99; i++)

{

change(arr1, i);

}

cout << "仍然打开的柜子是" << endl;

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

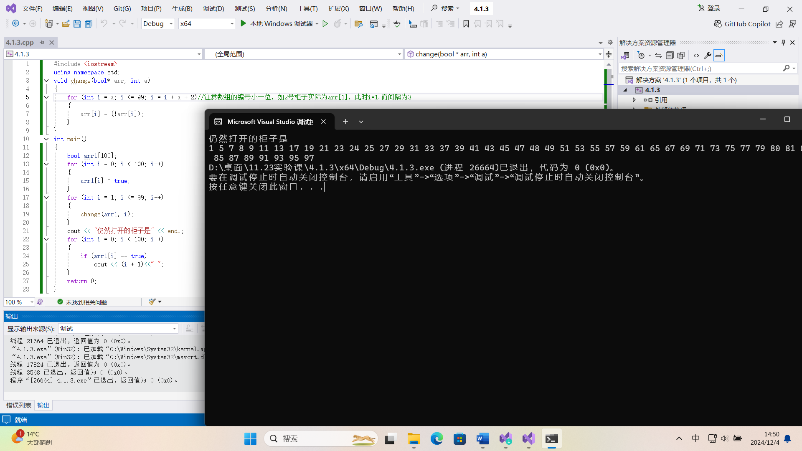
{

if (arr1[i] == true)

cout << (i + 1)<<" ";

}

return 0;

}

**4.1.4**#include <iostream>

using namespace std;

void bubbleSort(int\* arr,int a)//调用之前的升序函数

{

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

{

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

{

if (arr[j] > arr[j + 1]&&j+1<a)//这里注意要加上限，否则j+1是超小数就完了

{

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

int main()

{

int a, b;

cout << "请输入第一个数组的数目" << endl;

cin >> a;

cout << "请输入第二个数组的数目" << endl;

cin >> b;

int\* arr1,\*arr2,\*arr3;//弄三个动态数组

arr1 = new int[a];

arr2 = new int[b];

cout << "输入第一个数组" << endl;

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

{

cin >> arr1[i];

}

cout << "输入第二个数组" << endl;

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

{

cin >> arr2[i];

}

int c = a + b;

arr3 = new int[c];//将1，2整合入3，再对三升序

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

{

arr3[i] = arr1[i];

}

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

{

arr3[i] = arr2[i-a];

}

bubbleSort(arr3, c);

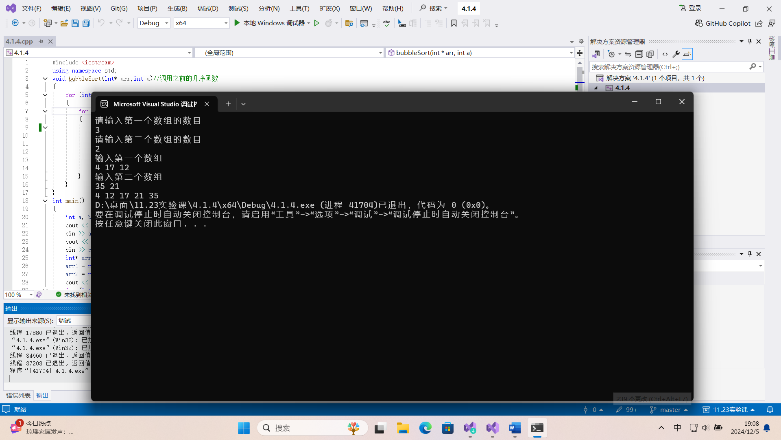
for (int i = 0; i < c; i++)

{

cout << arr3[i] << " ";

}

return 0;

****}

**4.1.5**#include <iostream>

using namespace std;

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

{

int len\_s1 = strlen(s1);

int len\_s2 = strlen(s2);

int equal = 0;//计数=0

for (int i = 0; i < len\_s2; i++)

{

equal = 0;//计数=0

if (s1[0] == s2[i] && len\_s1 <= (len\_s2 - i + 1))//如果子串的第一项与母串的某一项相等 且子串长度小于母串剩余长度

{

equal++;//计数+1

for (int j = 1; j < len\_s1; j++)

{

i++;//i+1

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

{

equal++;

}

else

{

break;

}

}

if (equal == len\_s1)//长度与各个字符都相等

{

return (i - len\_s1 + 1);

}

}

}

return -1;

}

int main()

{

char s1[100], s2[100];

cout << "Enter the first string: " << endl;

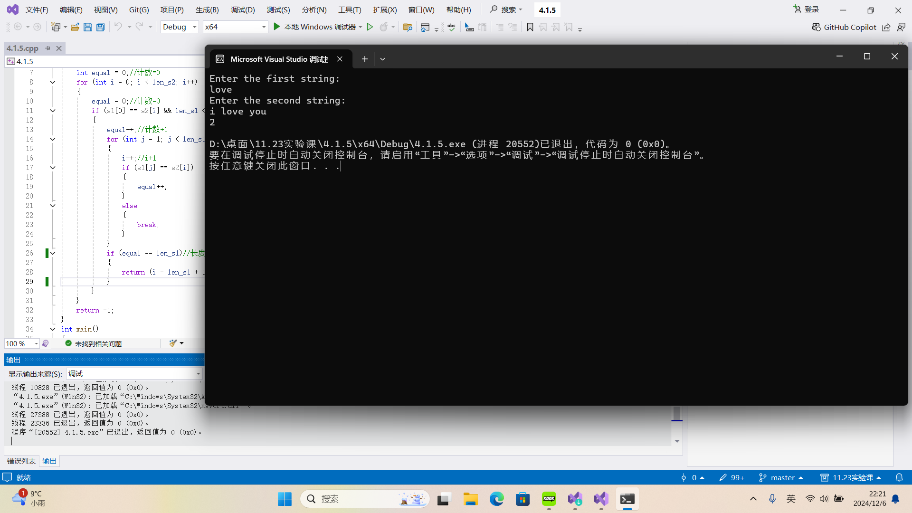
cin.getline(s1, 100);

cout << "Enter the second string: " << endl;

cin.getline(s2, 100);

cout<<indexOf(s1, s2)<<endl;

return 0;

}

**4.1.6** #include <iostream>

using namespace std;

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

{

int l\_s = strlen(s);

for (int i = 0; i < l\_s; i++)

{

int t = int(s[i]);

if (t >= 65 && t < 91)

counts[t]++;

if (t >= 97 && t <123 )

counts[t - 32]++;

}

}

int main()

{

char s[100];

int counts[100];

cout <<"Enter a string:" << endl;

cin.getline(s, 100);

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

counts[i] = 0;

count(s, counts);

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

{

if (counts[i] != 0)

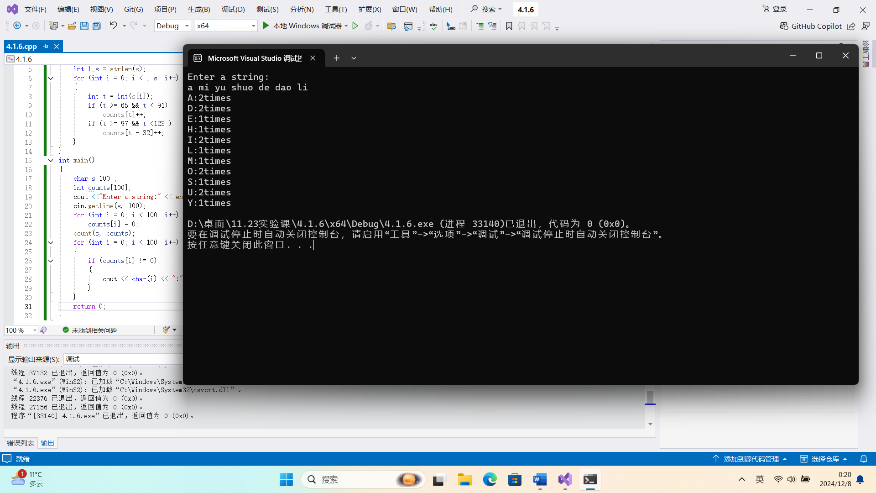
{

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

}

}

return 0;

}

**4.2.1.(1)** #include <iostream>

using namespace std;

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;

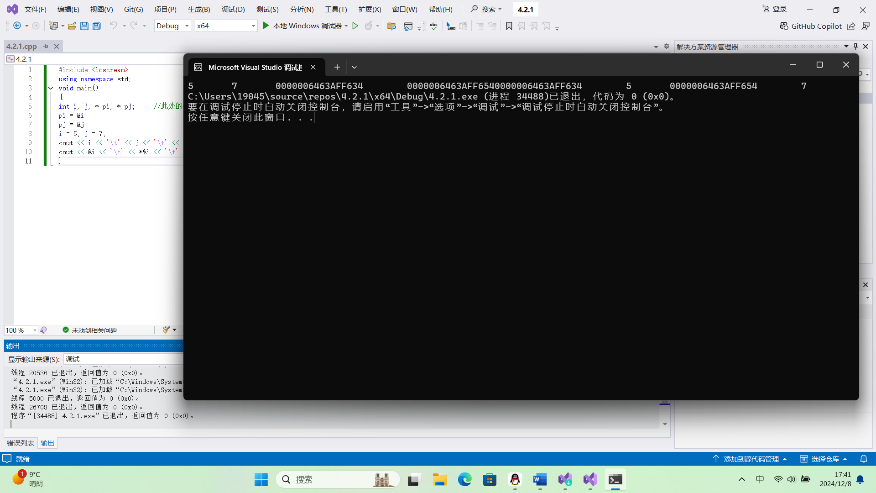
pj = &j;

i = 5; j = 7;

cout << i << '\t' << j << '\t' << pi << '\t' << pj;

cout << &i << '\t' << \*&i << '\t' << &j << '\t' << \*&j;

}

**4.2.1.(2)** #include <iostream>

using namespace std;

int main() //C语言程序，要了解

{

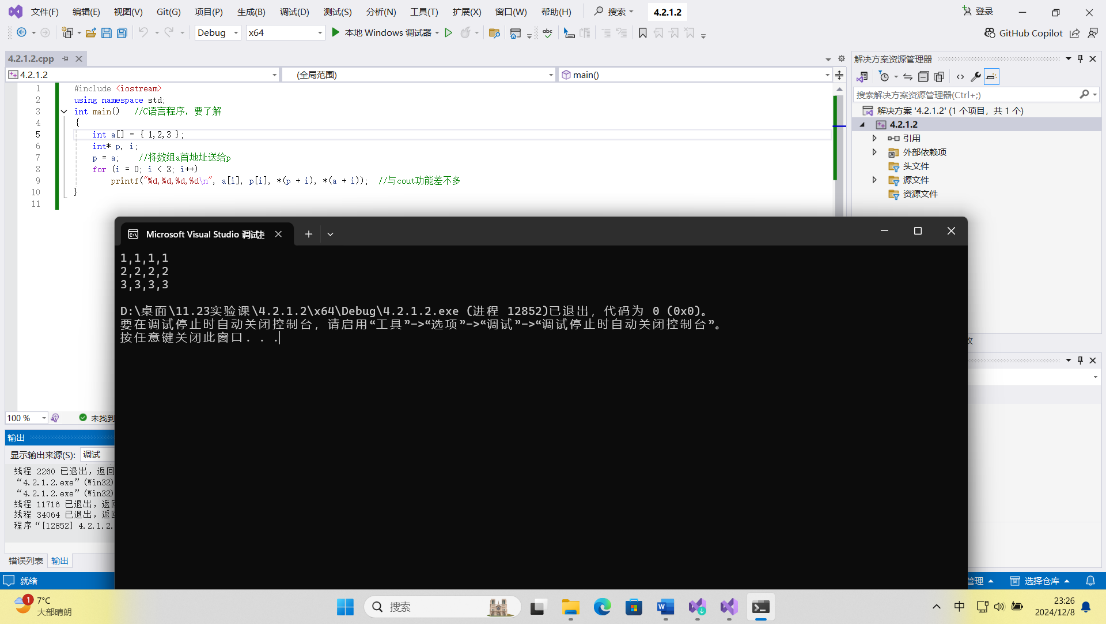
int a[] = { 1,2,3 };

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

}

**4.2.1.(3)** #include <iostream>

using namespace std;

void f(char\* st, int i)

{

st[i] = '\0';

cout << st; // printf(“%s\n”,st);

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

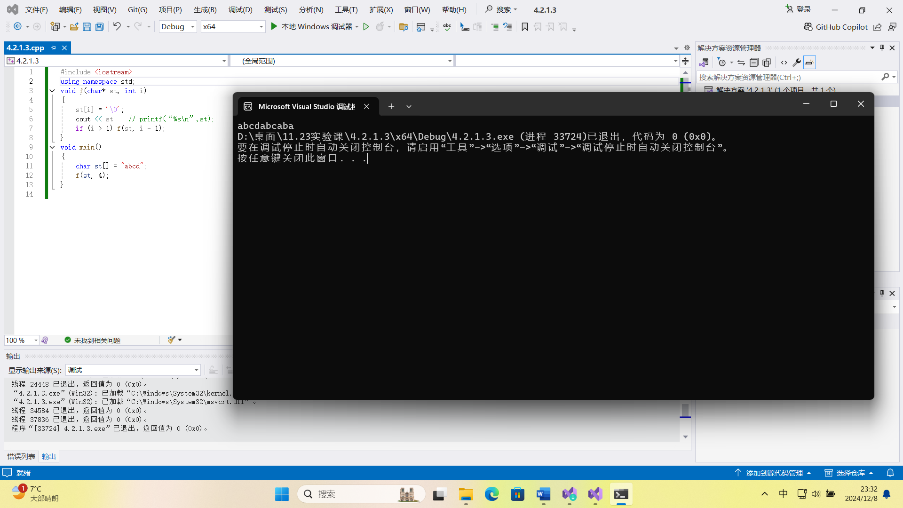
}

void main()

{

char st[] = "abcd";

f(st, 4);

}

**4.2.1.(4)** #include <iostream>

using namespace std;

int\* f()

{

int\* list = new int[4];

for (int i = 0; i < 4; i++)

list[i] = i + 1;

return list;

}

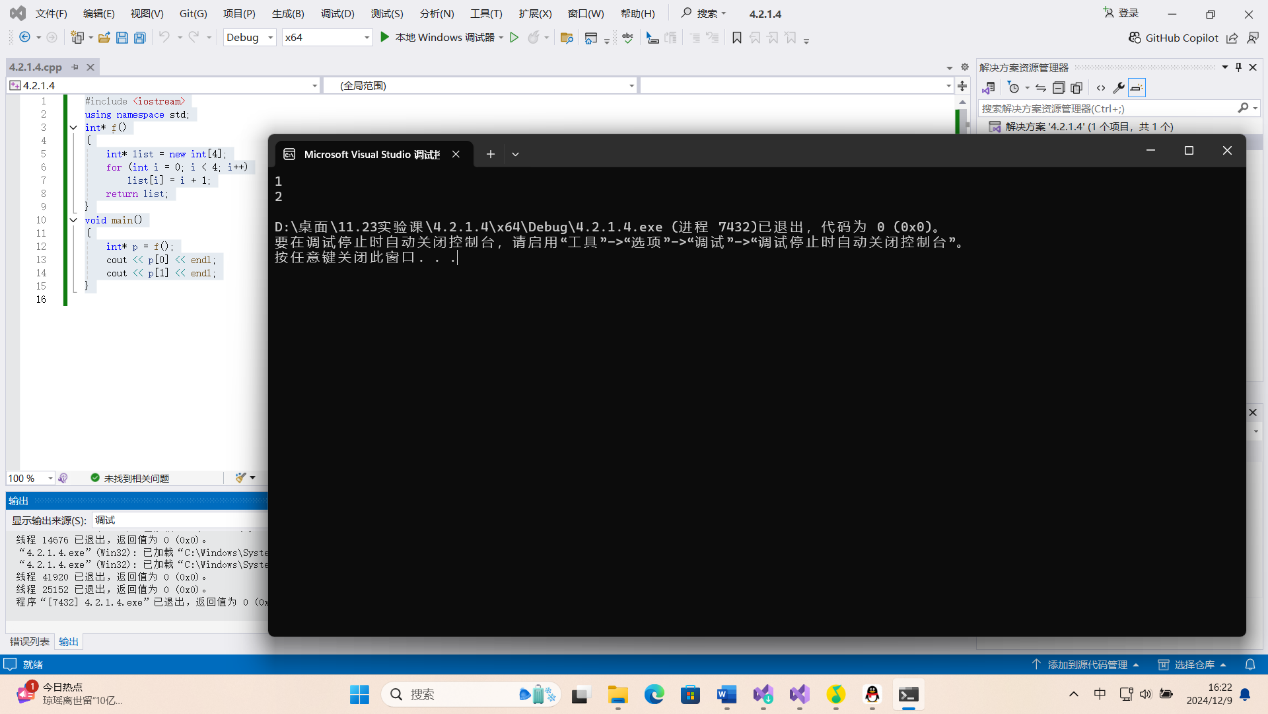
void main()

{

int\* p = f();

cout << p[0] << endl;

cout << p[1] << endl;

}

**4.2.2.(1) （这个题与4.1.5一模一样，就用一样的文件了）**

#include <iostream>

using namespace std;

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

{

int len\_s1 = strlen(s1);

int len\_s2 = strlen(s2);

int equal = 0;//计数=0

for (int i = 0; i < len\_s2; i++)

{

equal = 0;//计数=0

if (s1[0] == s2[i] && len\_s1 <= (len\_s2 - i + 1))//如果子串的第一项与母串的某一项相等 且子串长度小于母串剩余长度

{

equal++;//计数+1

for (int j = 1; j < len\_s1; j++)

{

i++;//i+1

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

{

equal++;

}

else

{

break;

}

}

if (equal == len\_s1)//长度与各个字符都相等

{

return (i - len\_s1 + 1);

}

}

}

return -1;

}

int main()

{

char s1[100], s2[100];

cout << "Enter the first string: " << endl;

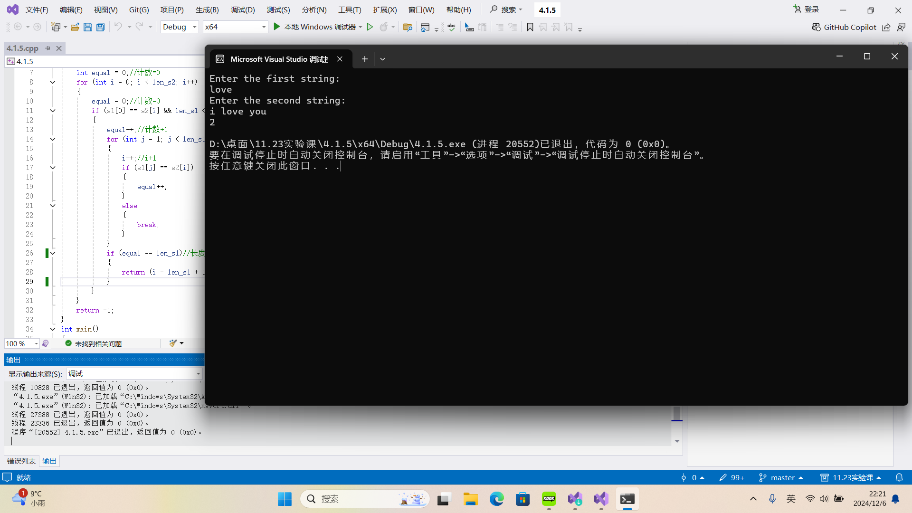
cin.getline(s1, 100);

cout << "Enter the second string: " << endl;

cin.getline(s2, 100);

cout<<indexOf(s1, s2)<<endl;

return 0;

}

**4.2.2.(2)** #include <iostream>

using namespace std;

int parseHex(const char\* const hexString)

{

int num = 0;

int len = strlen(hexString);

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

{

if (hexString[i] >= '0' && hexString[i] <= '9')//对于0-9的数值

{

num = num + ((int(hexString[i])) - 48) \* pow(16, len - 1 - i);//转换成int类型的0-9 根据相应位数乘上16的x-1次方

}

else if (hexString[i] >= 'A' && hexString[i] <= 'F')//对于A-F的数值

{

num = num + ((int(hexString[i])) - 55) \* pow(16, len - 1 - i);//转换为int类型的10-15 根据相应位数乘上16的x-1次方

}

else//不正确输入

cout << "不要乱输" << endl;

}

return num;

}

int main()

{

int t;

char s[100];

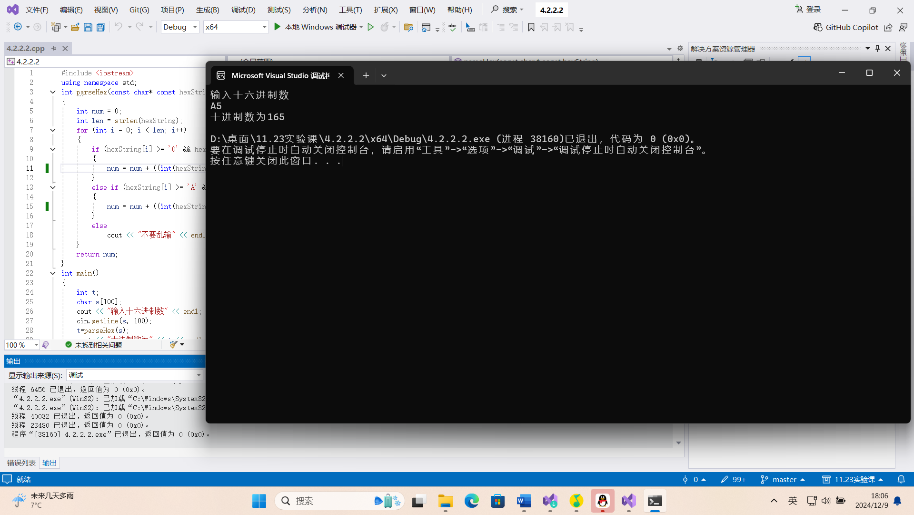
cout << "输入十六进制数" << endl;

cin.getline(s, 100);

t=parseHex(s);

cout << "十进制数为" << t << endl;

return 0;

}

**4.2.2.(3)本题的大致过程与我4.1.4的处理过程差不多，所以借鉴了大部分。**

#include <iostream>

using namespace std;

void bubbleSort(int\* arr, int a)//调用之前的升序函数

{

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

{

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

{

if (arr[j] > arr[j + 1] && j + 1 < a)//这里注意要加上限，否则j+1是超小数就完了

{

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

int main()

{

int a,\*arr;

cout << "请输入数组的数目" << endl;

cin >> a;

arr = new int[a];

cout << "请输入数组" << endl;

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

{

cin >> arr[i];

}

bubbleSort(arr, a);

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

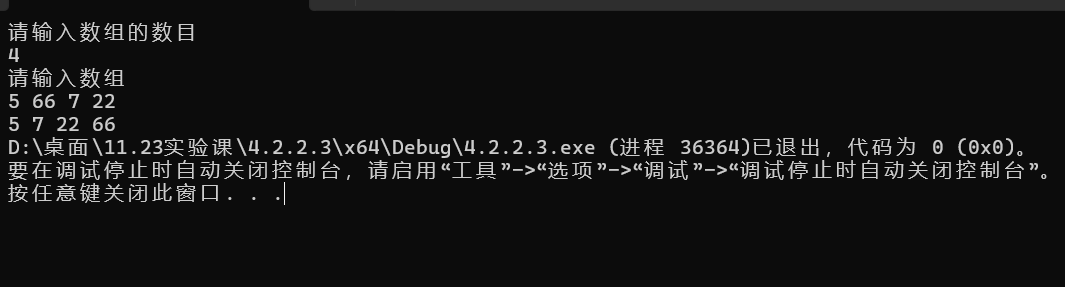
{

cout << \*&arr[i] << " ";//用指针输出

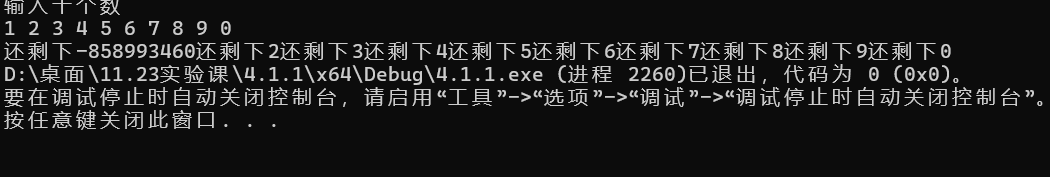
}

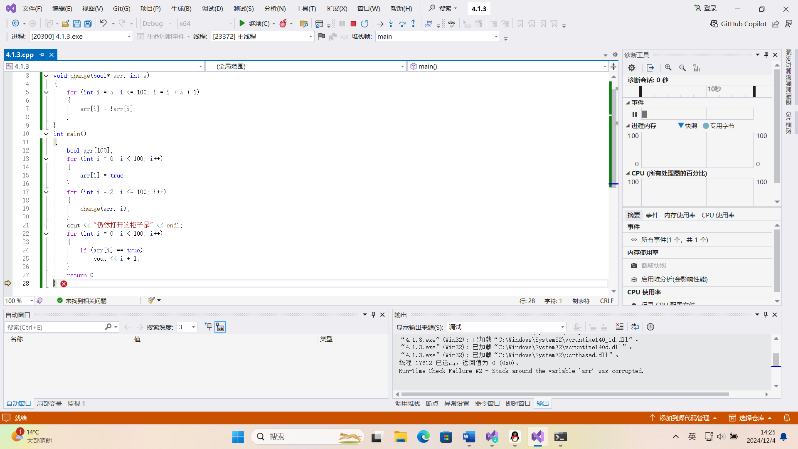
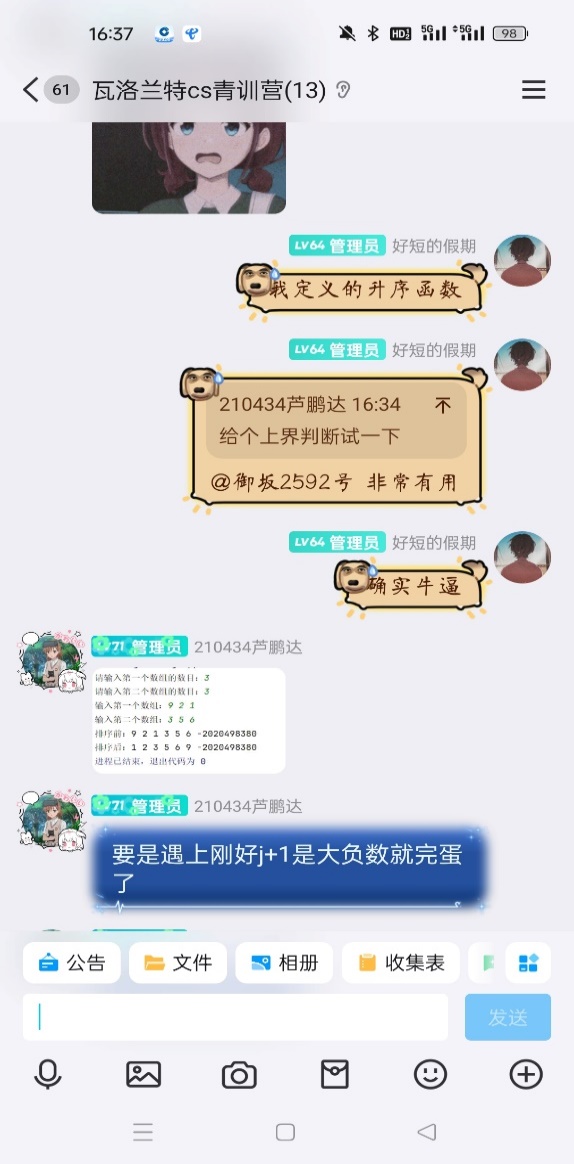
delete[]arr;//释放内存空间

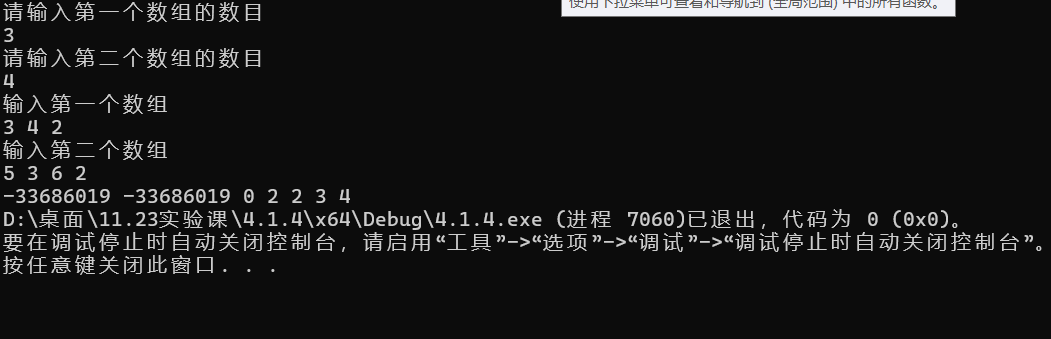
return 0;

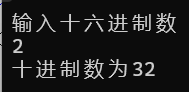
}

**四、遇到的问题与解决方法**

**4.1.1开始不知道怎么把重复的数剔除，后来引入了参数，将输入的数与之前的数一一比对，重复使参数加一，但还是出错了。分析发现，输入时a[i]可能赋值不到，而且循环的位置出错了，修改后解决。**

**4.1.3一开始写函数时没有考虑到数组的i只能取99，而对应柜子是100，而且编译器也找不出问题，后来自己再检查一遍才发现。注意数组的编号小一位，如2号柜子实际为arr[1]，此时i=1,而间隔为3。**

**4.1.4因为数组个数不能用变量，所以去搜索了一下动态数组的定义方法。然后又有bug。找了半天bug也找不到问题所在，遂向周围同学寻求帮助。最终向高中同学寻求帮助时指出了问题所在。我的冒泡函数是从4.1.2照搬而来，而4.1.2的数列是静态的，不需要管上限，而这里需要，最终改正。**

**4.2.2.(2)在调试程序时发现数据明显大了好多，如图。分析可知是最后一位，即个位乘的次数没算对。后了解到**(16^(len-i-1))**的表达并不正确，c++中没有乘方的表达。需用pow函数，最后解决。**

**五、体会**

在本次 C++数组与指针章节的上机实践中，我收获颇丰，也深刻认识到了自己的不足之处。

数组与指针的结合使用，让我看到了 C++语言在处理数据时的高效性与灵活性。通过指针操作数组元素，能够更加灵活地访问和修改数据，这使我对内存管理有了更直观的理解。

然而，在实践过程中，我明显感到对字符串类型的运用不够熟练。例如，在处理字符串数组时，常常混淆字符指针与字符数组的区别，导致一些字符串操作函数的使用出现错误。一些str类型的函数也不是特别了解。

反思自己的不足，主要是因为对字符串相关知识的学习不够深入系统，上课与做题都很少看到字符与字符串的题型，缺乏足够的实践练习。今后，我会加强对字符串类型的专项学习，仔细研究字符串处理函数的原理和用法，多做相关的编程练习，提高自己代码的简洁性和准确性。同时，在后续的学习中，我也会更加注重知识的系统性和深入性，不仅仅满足于代码能够运行，而是要追求更加高效、优雅的编程方式。