**计算机程序设计基础（C++)**

**实验报告**

专业班级： 软件工程2402

学 号： 8209240212

姓 名： 吴宗翰

**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？

因为定义函数类型就是定义返回值的类型。

2. 本实验中主函数调用函数时采用的是何种传递方式？

值传递，将实参的值复制一份传给实参。

**四、算法分析，程序结果**

1.

#include <iostream>

using namespace std;

int gy(int a, int b)

{

while (b != 0)

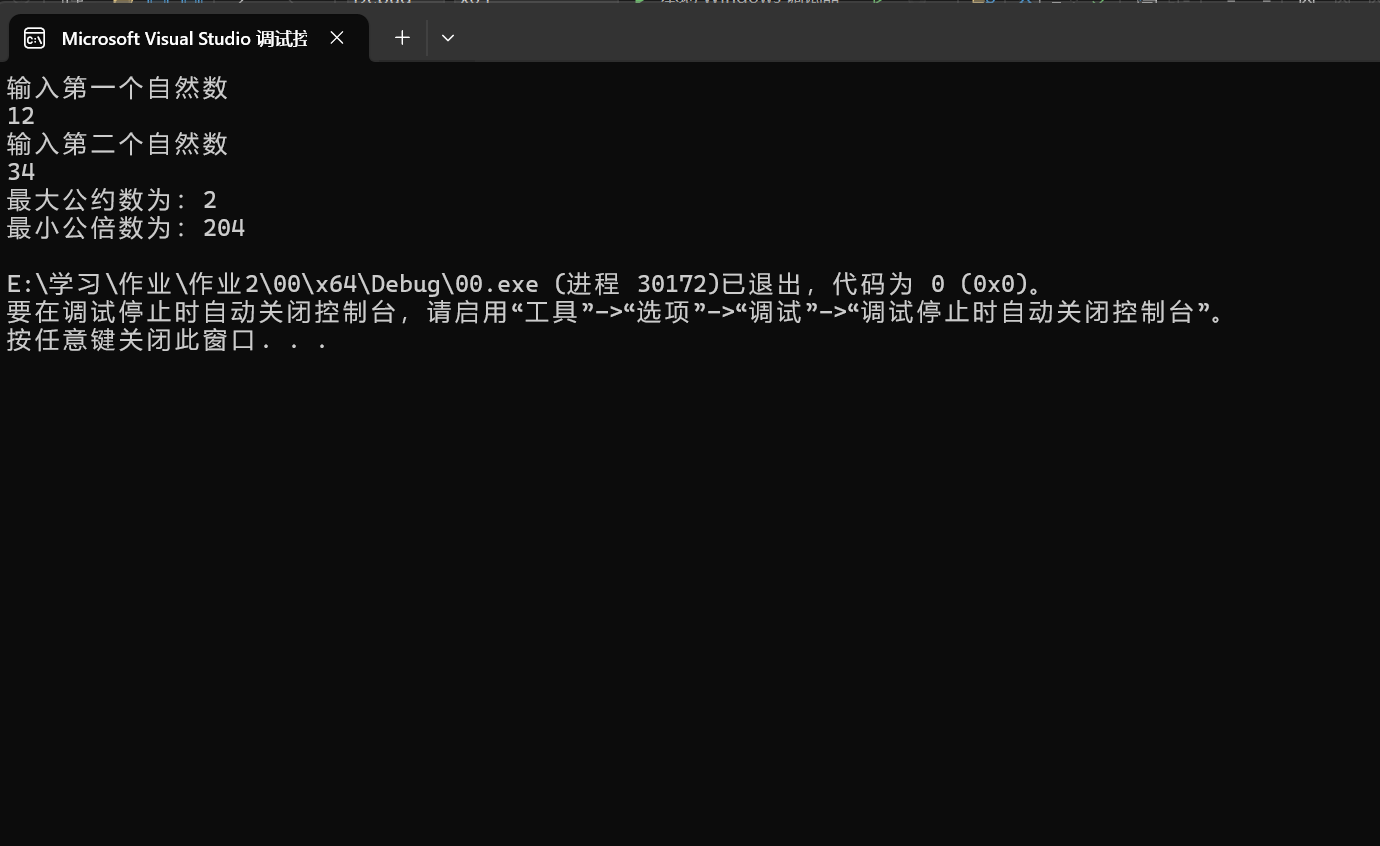
{

int temp = b;

b = a % b;

a = temp;

}

 return a;

}

int gb(int p, int q)

{

return p \* q / gy(p, q);

}

int main()

{

int m, n;

cout << "输入第一个自然数" << endl;

cin >> m;

cout << "输入第二个自然数" << endl;

cin >> n;

int p, q;

p = gy(m, n);

q = gb(m, n);

cout << "最大公约数为：" << p << endl << "最小公倍数为：" << q << endl;

}

通过辗转相除法来得到最大公约数，利用得到的最大公约数来获得最小公倍数。

2.

#include<iostream>

using namespace std;

bool is\_prime(int num)

{

if (num <= 1)

return false;

if (num == 2 || num == 3)

return true;

for (int m = 2; m < num; m++)

{

if (num % m == 0)

{

return false;

break;

}

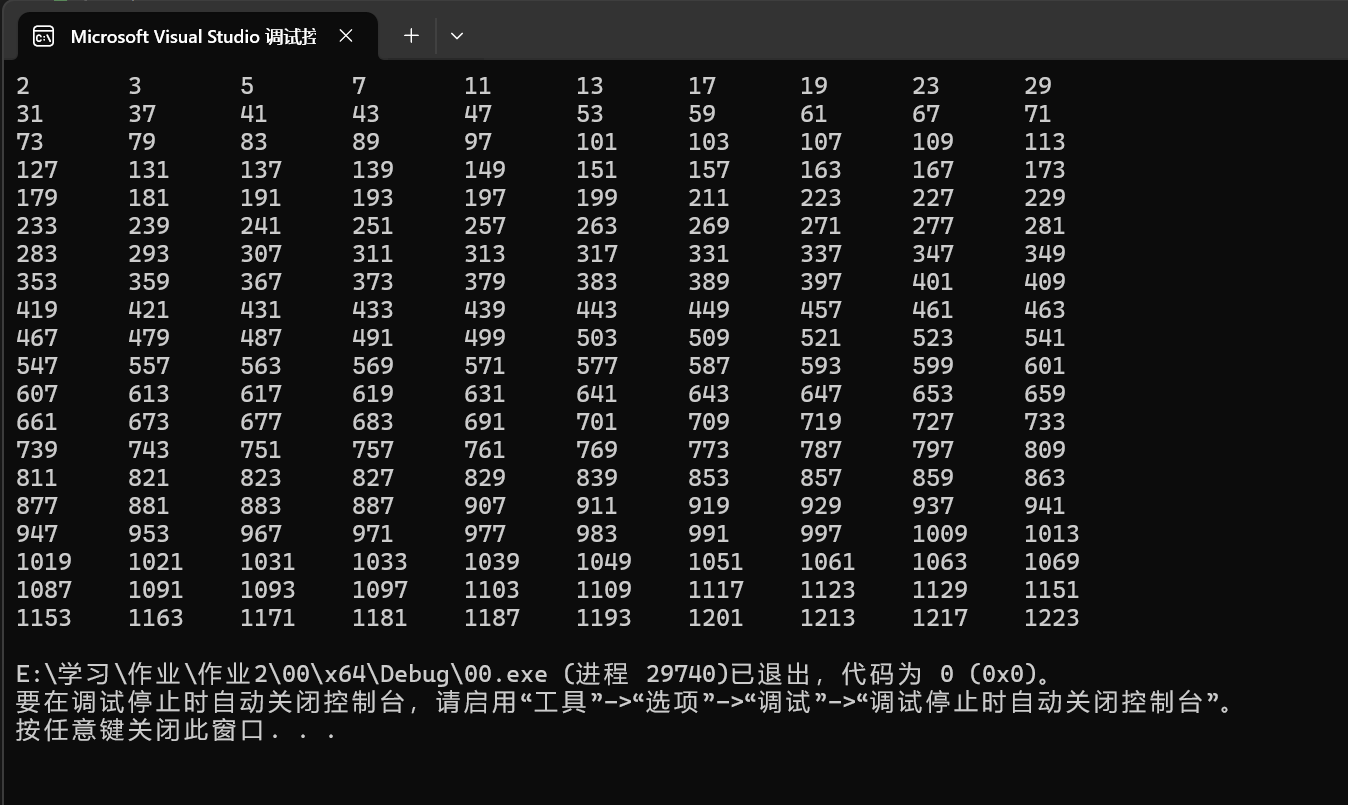
}

return true;

}

int main()

{

 int a=0,b=0;

while (b < 200)

{

if (is\_prime(a))

{

cout << a << '\t';

b++;

if (b % 10 == 0)

{

cout << endl;

}

}

a++;

}

return 0;

}

首先在函数中分1，2，3这三种特殊情况，其他一般情况通过循环数列来判断这个数能不能被比他小的数整除，从而可以判断读入函数的数是不是质数。在主函数中建立循环，将每个数都执行前面设计的函数，每发现一个质数则让b+1直到b=200从而找到前200个质数。

3．

主函数：

#include<iostream>

#include "mytemperature.h"

#include <iomanip>

using namespace std;

int main()

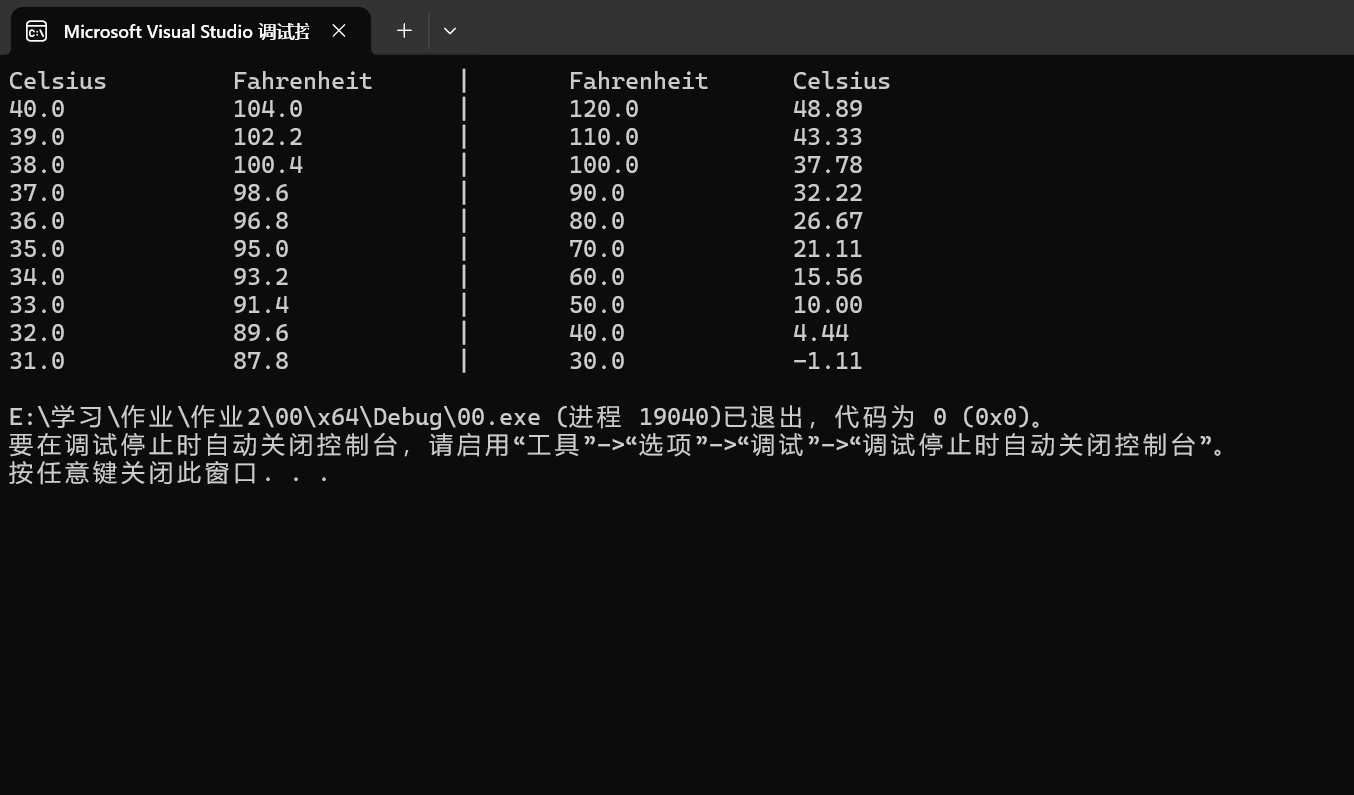
{

cout << "Celsius" << '\t' << '\t' << "Fahrenheit" << '\t' << "|" << '\t' << "Fahrenheit" << '\t' << "Celsius" << endl;

for (double m = 40.0,n = 120.0; m >= 31.0; m--)

{

cout<< fixed << setprecision(1) << m << '\t' << '\t' << celsius\_to\_fah(m) << '\t' << '\t' << "|" << '\t' << n << '\t' << '\t' << fixed << setprecision(2)<< fahrenheit\_to\_cels(n) << endl;

 n = n - 10.0;

}

return 0;

}

mytemperature.cpp：

#include<iostream>

#include "mytemperature.h"

extern double celsius\_to\_fah(double cel)

{

return(cel \* 9 / 5) + 32;

}

extern double fahrenheit\_to\_cels(double fah)

{

return(fah - 32) \* 5 / 9;

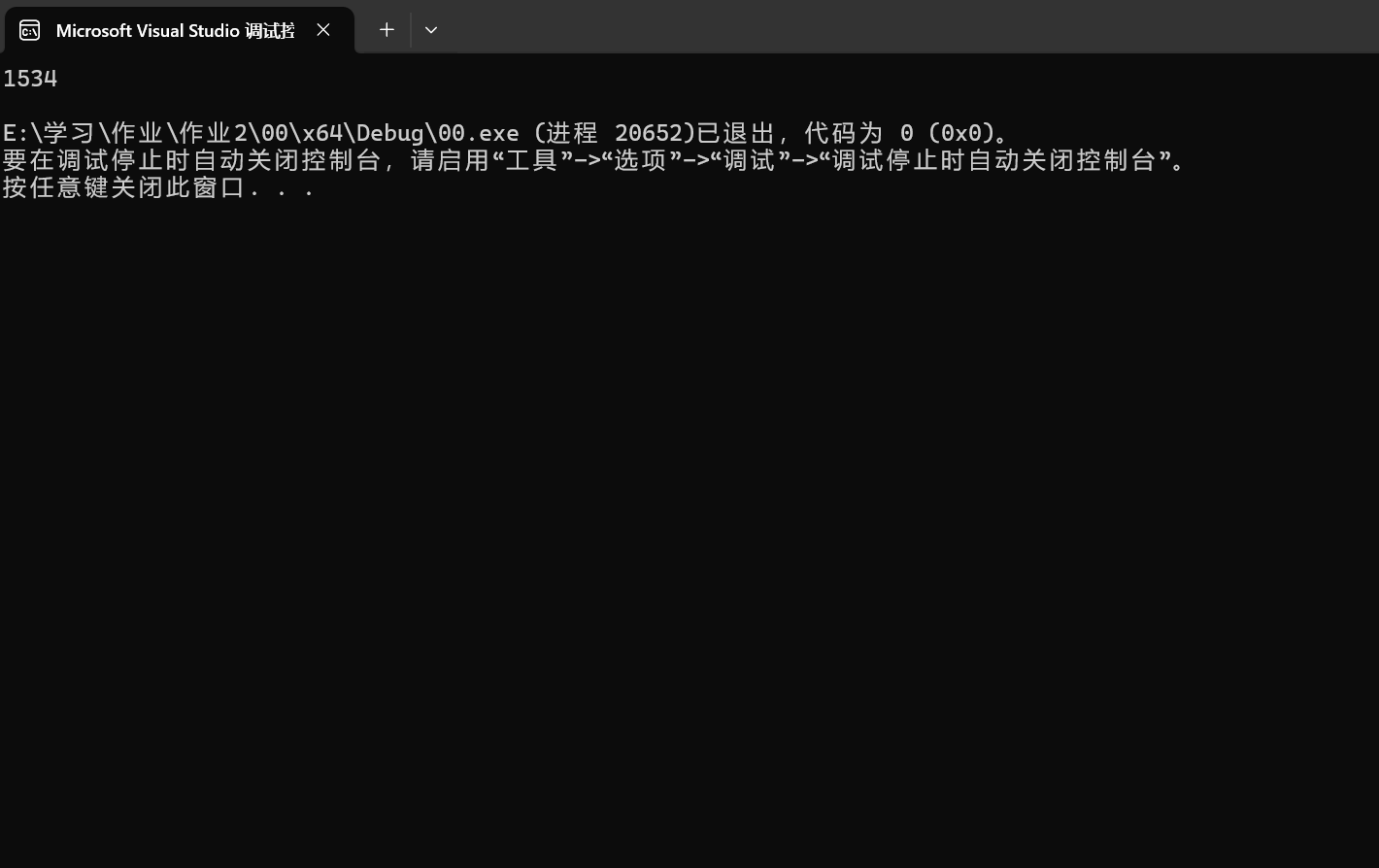
}

mytemperature.h：

double celsius\_to\_fah(double cel);

double fahrenheit\_to\_cels(double fah);

用extern和#...””来达到引用的目的。

5.

#include<iostream>

using namespace std;

int houzi(int m,int p)

{

if (p <= 9)

{

m = houzi(2 \* (m + 1), p + 1);

}

return m;

}

int main()

{

int a = 1, b = 1;

a = houzi(a,b);

cout << a << endl;

}

用倒推法，从1开始每次乘二加一一直到第九次，即倒推回第一天。用函数的递归来实现，条件就是p<=9。

**五、遇到的问题与解决方法**

1.在做第二题的时候，一开始未能正确分类导致1，2未能出现在结果中。最终通过观察实验结果得以发现问题并做出修改。

2.在做第三题时输出结果与预期相差甚远，主要是格式版面有问题，最终通过反复实验得出最终想要的结果。

3.在做第四题时想了很久，最终想起用倒推法解决了问题。

**六、体会**

1.在做一个程序时要先分析结果有几种可能性，详尽地将所有情况讨论出来才能达到实验目的，而不能没想好就开始敲代码。

2.在运行代码发现结果与预期不符时可以通过多次修改来达到目的

**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**

1.

#include<iostream>

using namespace std;

int main()

{

int a[10], c[10], n = 0;

cout << "Enter ten number:";

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

{

cin >> a[i];

bool h = true;

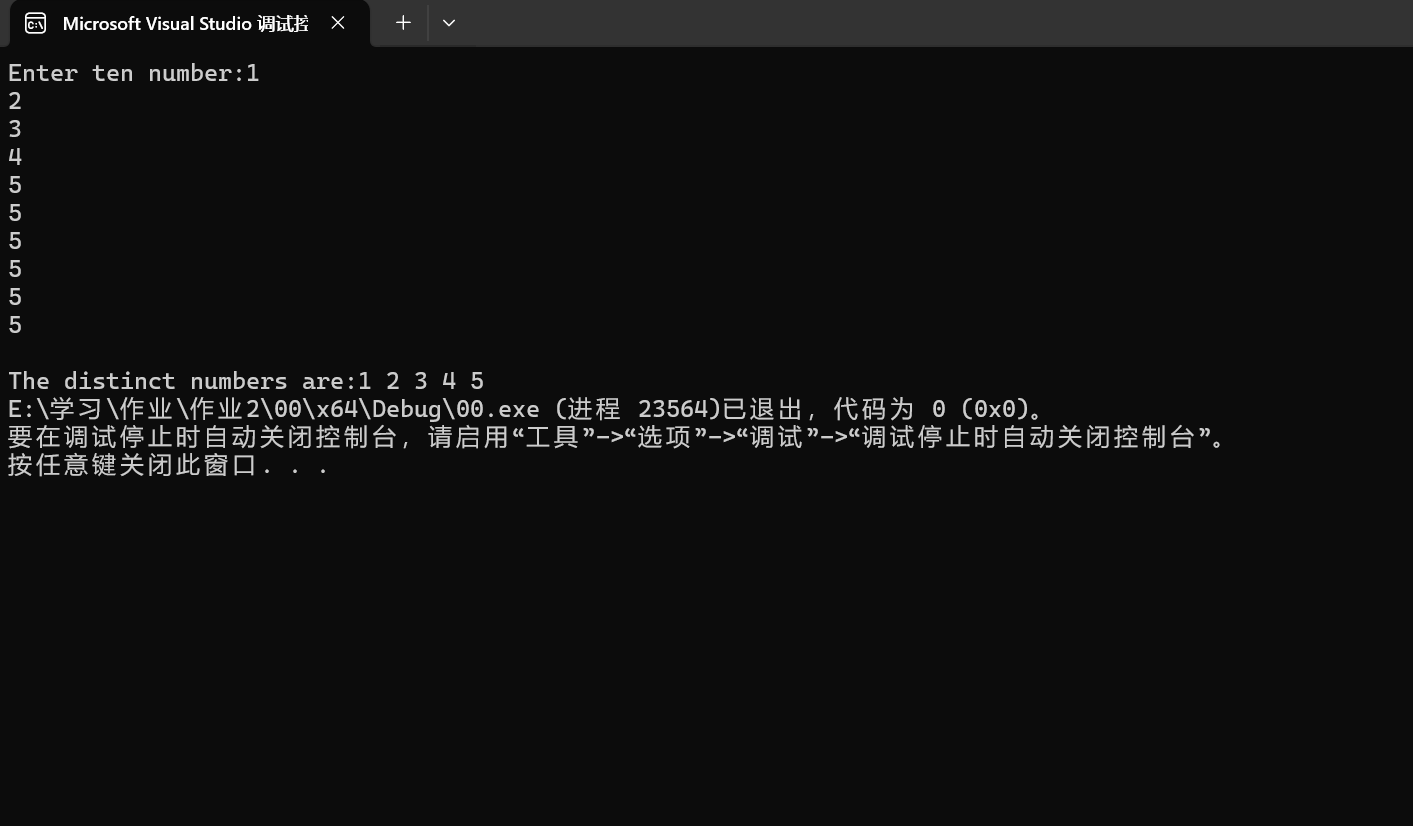
for (int j = 0; j < n; ++j)

{

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

{

h = false;

 break;

}

}

if (h == true)

{

c[n] = a[i];

n++;

}

}

cout << endl;

cout << "The distinct numbers are:";

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

{

cout << c[j] << " ";

}

return 0;

}

用循环与布尔类型来判断新输入的数有没有与数组中数相同的数。

2.

#include<iostream>

using namespace std;

int main()

{

double a[10];

cout << "输入十个数：" << endl;

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

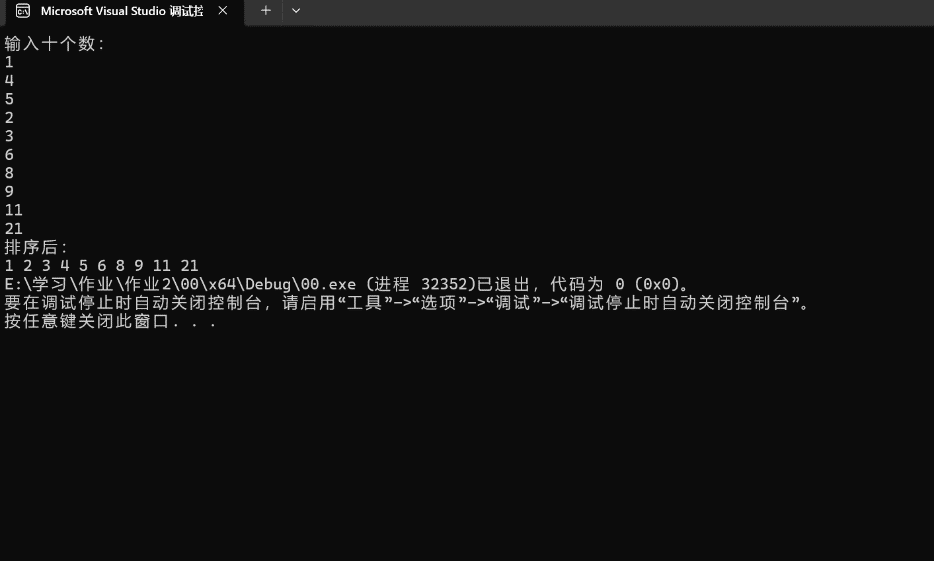
{

cin >> a[m];

}

for (int b = 9; b > 0; b--)

{

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

{

if (a[c] > a[c + 1])

{

int p = a[c];

a[c] = a[c + 1];

a[c + 1] = p;

}

}

}

cout << "排序后：" << endl;

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

{

cout << a[m] << " ";

}

}

从首个数字开始依次与后面的数字作比较，若前面的数字大，则前后换位置直到最后一个。然后重复此过程直到倒数第二个。依次法决定每个位置应该是哪个数字从而达到排序的目的。由此可知每次循环的次数来编写程序。

3.

#include<iostream>

using namespace std;

int main()

{

bool m[100];

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

{

m[a] = false;

}

for (int b = 1; b <= 100; b++)

{

for (int c = 1; c \* b <= 100; c++)

{

int q = c \* b;

m[q] = !m[q];

}

}

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

{

if (m[d] == true)

{

cout << d << " ";

}

}

}

用bool类型来模拟柜子开关，通过c\*b<=100的嵌套循环来模拟n个学生隔n个柜子开关柜子。

4.

#include<iostream>

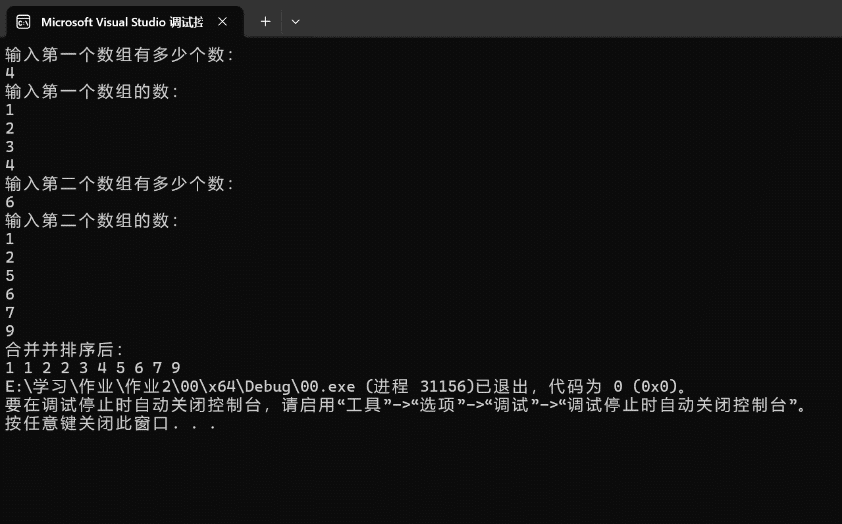
#include<string>

using namespace std;

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

{

int m = size1 + size2;

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

{

if (i < size1)

{

list3[i] = list1[i];

}

else

{

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

}

}

for (int i = m - 1; i > 0; i--)

{

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

{

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

{

int q = list3[c];

list3[c] = list3[c + 1];

list3[c + 1] = q;

}

}

}

}

int main()

{

int m, n;

cout << "输入第一个数组有多少个数：" << endl;

cin >> m;

int a[20];

int b[20];

cout << "输入第一个数组的数：" << endl;

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

{

cin >> a[i];

}

cout << "输入第二个数组有多少个数：" << endl;

cin >> n;

cout << "输入第二个数组的数：" << endl;

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

{

cin >> b[i];

}

int c[40];

merge(a, m, b, n, c);

cout << "合并并排序后：" << endl;

for (int i = 0; i < m + n; i++)

{

cout << c[i] << " ";

}

}

在函数中先计算两个数组的元素总个数，然后先后把第一个数组第二个数组的元素一个一个填进第三个数组里面，最后再进行排序。

5.

#include<iostream>

using namespace std;

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

{

bool h = false;

int len1 = 0, len2 = 0;

for (int i = 0; s1[i] != '\0'; i++)

{

len1++;

}

for (int i = 0; s2[i] != '\0'; i++)

{

len2++;

}

if (len1 >= len2)

{

return -1;

}

else

{

for (int i = 0; i < len2; i++)

{

for (int o = 0; o < len1; o++)

{

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

{

h = true;

return i;

break;

}

}

if (h == true)

{

break;

}

}

if (h == false)

{

return -1;

}

}

}

int main()

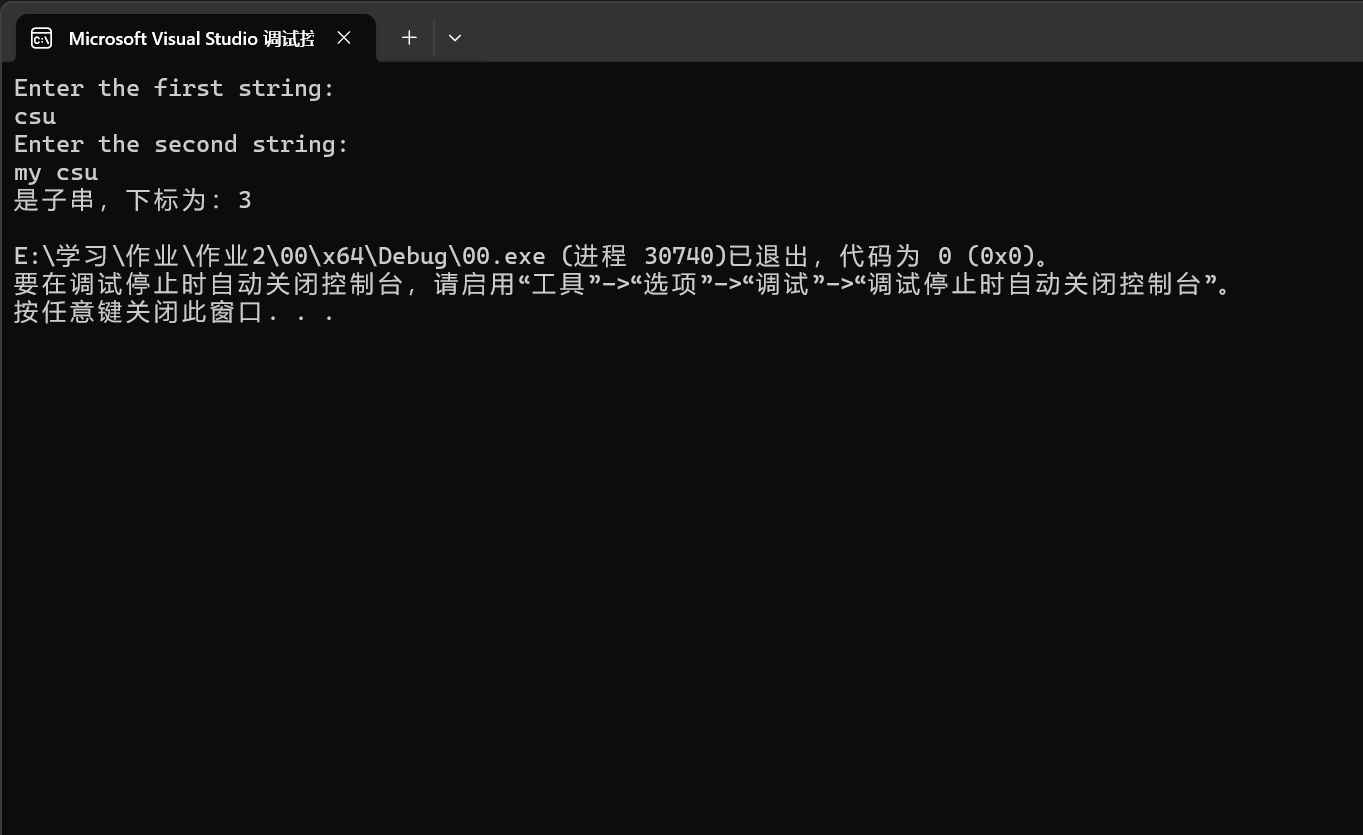
{

char m[30];

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

cin.getline(m, 30);

char n[30];

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

cin.getline(n, 30);

int k;

k = indexOf(m, n);

if (k == -1)

{

cout << "不是子串" << endl;

}

else

{

cout << "是子串，下标为：" << k << endl;

}

}

通过元素是否为’\0’来判断这个字符串的始末，再判断两个数组元素个数大小，前者大则不是子串，前者小则判断后者有无与前者相同的字符来判断是否为子串。

6.

#include <iostream>

using namespace std;

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

{

for (int i = 0; s[i] != '\0'; i++)

{

for (int o = 0; o < 26; o++)

{

if (s[i] == char(65 + o) || s[i] == char(97 + o))

{

counts[o]++;

break;

}

}

}

for (int i = 0; i < 26; i++)

{

if (counts[i] != 0)

{

if (counts[i] == 1)

{

cout << char(65 + i) << ":" << counts[i] << " time" << endl;

}

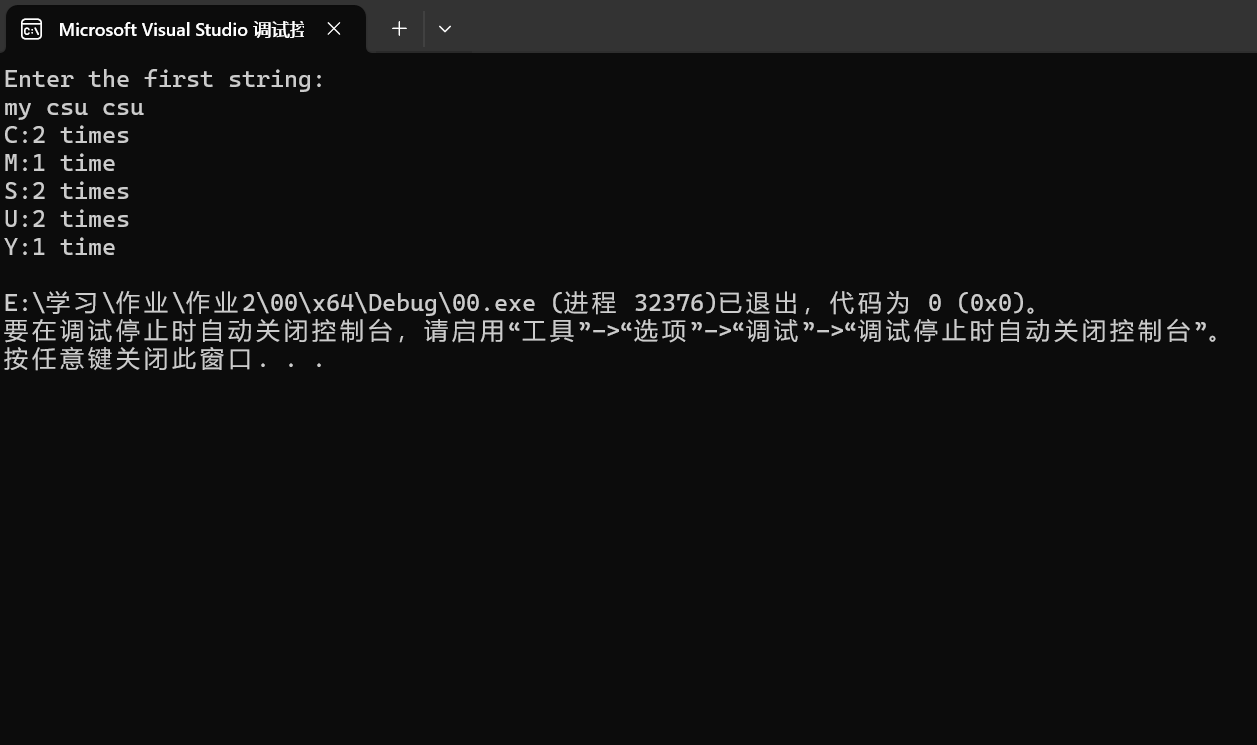
else

{

cout << char(65 + i) << ":" << counts[i] << " times" << endl;

}

}

 }

}

int main()

{

int counts[26];

for (int i = 0; i < 26; i++)

{

counts[i] = 0;

}

char m[30];

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

cin.getline(m, 30);

count(m, counts);

}

函数中通过两个数组，一个数组为字符串，另一个表示26个英文字母出现的次数，用ascii码来判断字符串的字符，从而实现统计。

（二）

1.

#include<iostream>

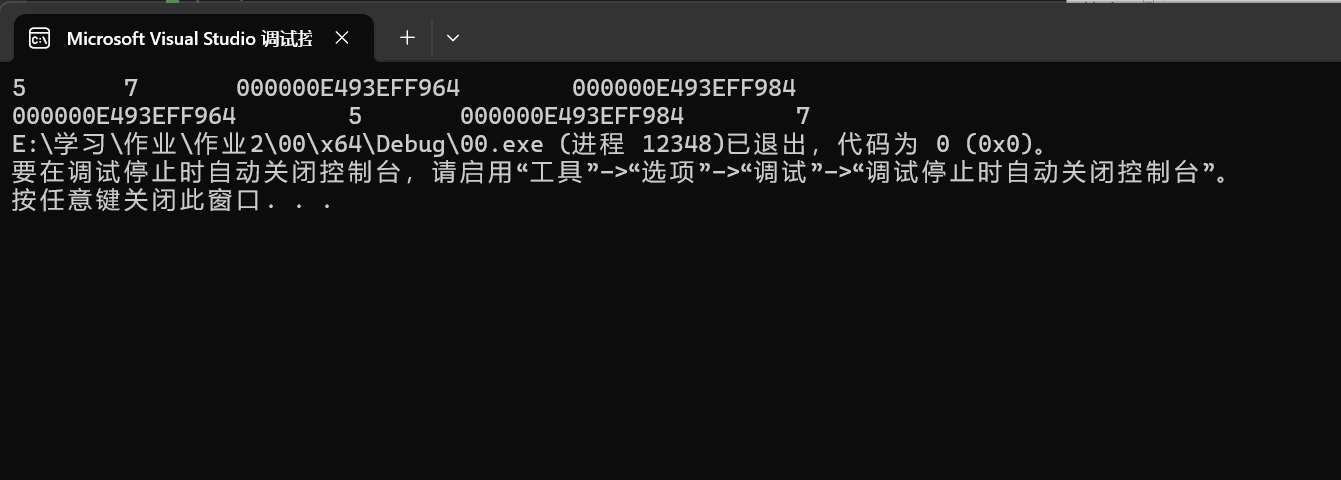
using namespace std;

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;

}

这里的pi与pj是分别指向p和j的指针，其值是p和j的地址，即&p和&j，所以可以看到输出结果pi pj和&i &j是相同的，而\*&p和\*&j与\*pi \*pj一样是代表该地址所指向的值。

2.

#include<iostream>

#include<stdio.h>

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

}

对于数组的指针，如上面p=a是指的a数组首元素的指针。而p+i则是指的数组首元素下一个元素的地址。

3.

#include<iostream>

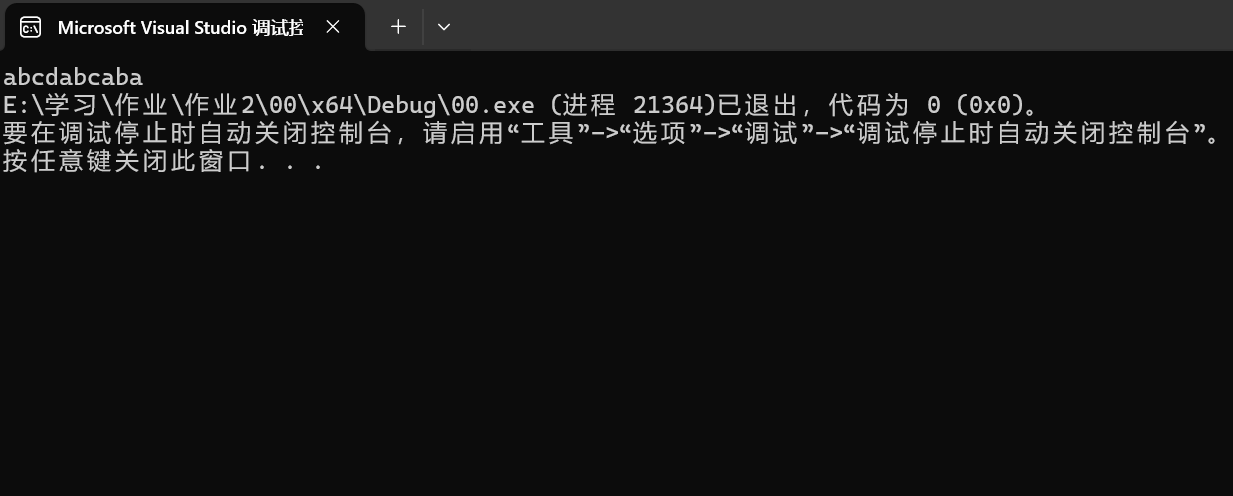
#include<stdio.h>

using namespace std;

void f(char\* st, int i)

{

st[i] = '\0';

 cout << st;

if (i > 1)

{

f(st, i - 1);

}

}

void main()

{

char st[] = "abcd";

f(st, 4);

}

本程序用函数的嵌套来依次递减输出数组元素，从前n个到前n-1个直到st[i]=‘\0’即结束。

4.

#include<iostream>

using namespace std;

int\* f()

{

int \*list = new int[4] ;

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

{

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

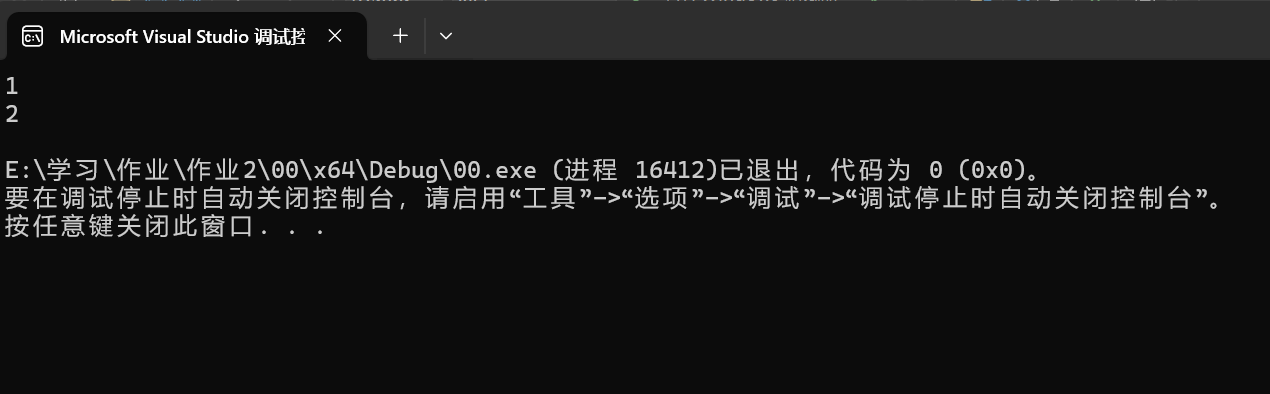
}

return list;

}

void main()

{

 int\* p = f();

cout << p[0] << endl;

cout << p[1] << endl;

delete[] p;

}

本程序在函数中创建了一个动态数组，该动态数组实则是{1，2，3，4}，用\*（list+i）来给数组每一个元素赋值。

2.程序设计

(1).

#include<iostream>

using namespace std;

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

{

bool h = false;

int len1 = 0, len2 = 0;

for (int i = 0; s1[i] != '\0'; i++)

{

len1++;

}

for (int i = 0; s2[i] != '\0'; i++)

{

len2++;

}

if (len1 >= len2)

{

return -1;

}

else

{

for (int i = 0; i < len2; i++)

{

for (int o = 0; o < len1; o++)

{

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

{

h = true;

return i;

break;

}

}

if (h == true)

{

break;

}

}

if (h == false)

{

return -1;

}

}

}

int main()

{

char m[30];

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

cin.getline(m, 30);

char n[30];

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

cin.getline(n, 30);

int k;

k = indexOf(m, n);

if (k == -1)

{

cout << "不是子串" << endl;

}

else

{

cout << "是子串，下标为：" << k << endl;

}

}

这里是指针传递而不是值传递，所以和上面的不一样，但结果一样。

(2).

#include<iostream>

using namespace std;

int parseHex(const char\* const hexString)

{

int len = 0;

int m = 0;

for(int i = 0; hexString[i] != '\0'; i++)

{

len++;

}

for (int i = len; i > 0 ; i--)

{

int o = 0;

if (hexString[i-1] > 47 && hexString[i-1] < 58)

{

o = hexString[i-1] - 48;

m += o \* pow(16,len-i);

}

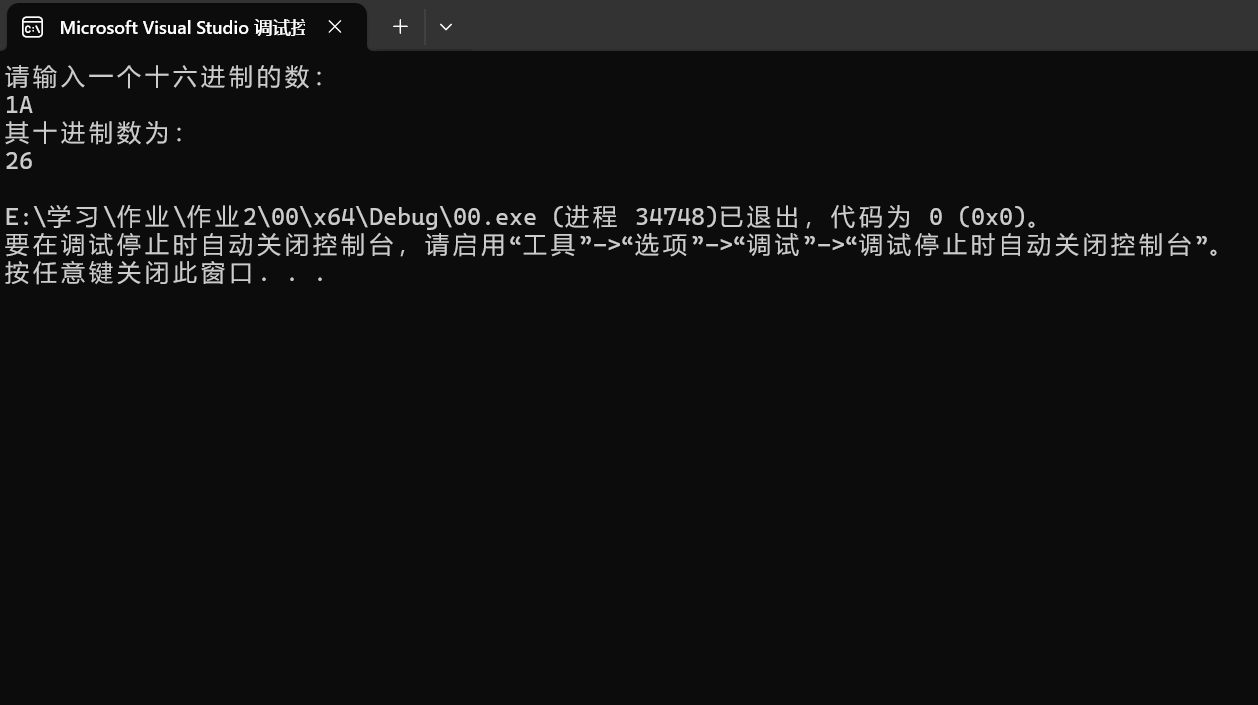
if (hexString[i-1] > 64 && hexString[i-1] < 71)

{

o = hexString[i-1] - 55;

m += o \* pow(16,len-i);

}

 }

return m;

}

int main()

{

int r;

char k[300];

cout << "请输入一个十六进制的数：" << endl;

cin.getline(k, 300);

char\* m = k;

r = parseHex(m);

cout << "其十进制数为：" << endl;

cout << r << endl;

}

函数核心是用ascii码来将每一位字符转化为相应十进制的数，并且用len来记录字符个数从而计算有该数有几位。

(3).

#include<iostream>

using namespace std;

void bubble(int\* a,int b)

{

for (int i = b - 1; i > 0; i--)

{

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

{

if (a[m] > a[m + 1])

{

int q = a[m];

a[m] = a[m + 1];

a[m + 1] = q;

}

}

}

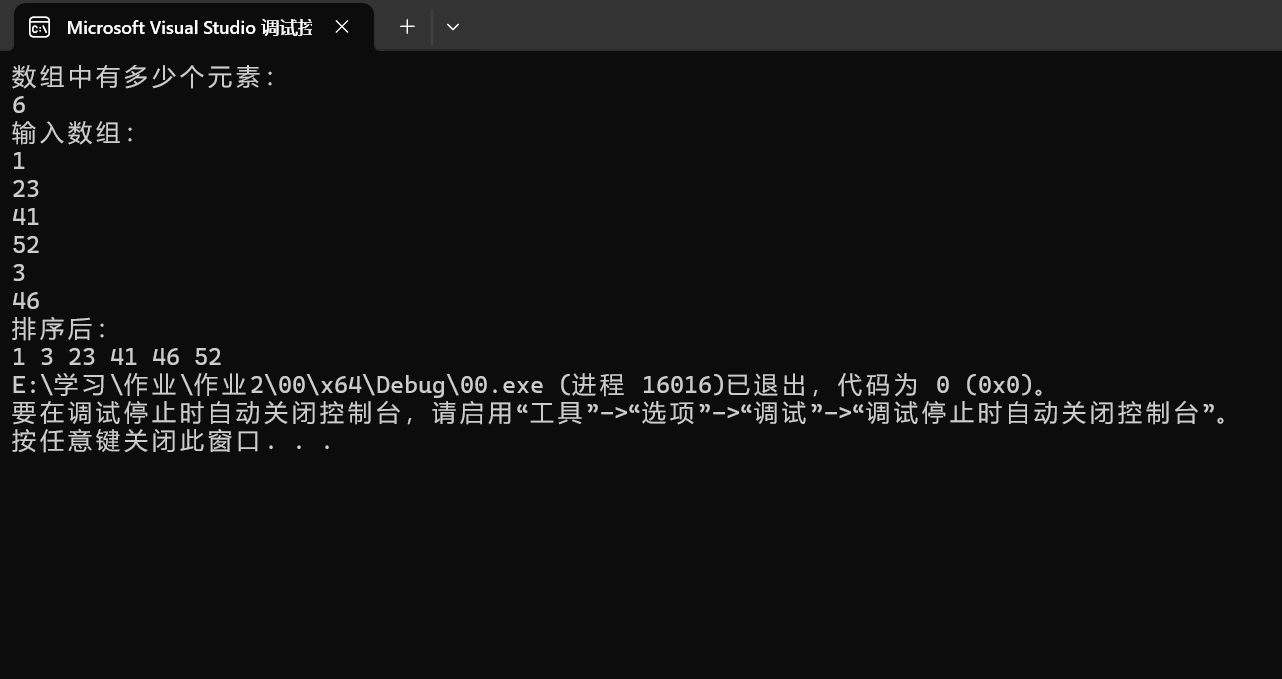
}

int main()

{

int b;

cout << "数组中有多少个元素：" << endl;



cin >> b;

int \*a=new int[b];

cout << "输入数组：" << endl;

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

{

cin >> a[i];

}

bubble(a, b);

cout << "排序后：" << endl;

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

{

cout << \*(a+i) << " ";

}

delete[] a;

}

仍然是用冒泡排序，但是区别是用了指针传递和用指针来输出数组元素。

**四、遇到的问题与解决方法**

1. 第一题之前的思路是输入十个数之后再比较十个数中有没有不一样的数 将不一样的数移入新数组 找到的是独立的数而不是题目意义上的每个数只输出一遍。

2．第三题思路一开始错误，找的关系出错了，后来再查询相关问题资料后改正思路重写了代码。

3．在程序设计板块第二题中输出一直是乱码，在函数板块思考很久结果发现是主函数定义数组时出现问题，导致耗费大量时间。

**五、体会**

1.在写代码前先要把思路理清，要不然容易理解错而写出的程序达不到真正目的，要训练自己抽象问题的能力。

2．在程序出现问题时不能只看一方面，要从头到尾都看，并且在编译过程中可以做好注释，使自己找错误时更清晰。