**计算机程序设计基础（C++)**

**实验报告**

专业班级：软工3班

学 号： 8209230307

姓 名：黄超俊

**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？

**四、算法分析，程序结果**

**1.** #include<iostream>;

using namespace std;

int gongyue(int a, int b) //求最小公约数的函数

{

int num = 0;

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

{

if ((a % i == 0) && (b % i == 0) && (i != 1))

{

num = i;

}

if ((i == 1) && (num == 0))

{

num = i;

}

}

return(num);

}

int main() {

int m, n, gy, gb;

cout << "请输入两个正整数：";

cin >> m >> n;

gy = gongyue(m, n);

gb = m \* n / gy;

cout << "两数最小公约数为：" << gy << endl;

cout << "两数最小公倍数为：" << gb << endl;

return 0;

}

**2.**

bool is\_prime(int num)

{

int num\_index = 0;

for (int i = num - 1; i > 1; i--)

{

if (num % i == 0)

{

return false;

}

num\_index = i;

}

if (num\_index == 1)

{

return true;

}

}

int main()

{

int num = 0;

int round = 1;

for (int i = 1; i <= 200; i++)

{

if (is\_prime(i))

{

num += 1;

cout << i << " ";

}

if ((num != 0) && (num % 10 == 0) && (num / 10 == round))

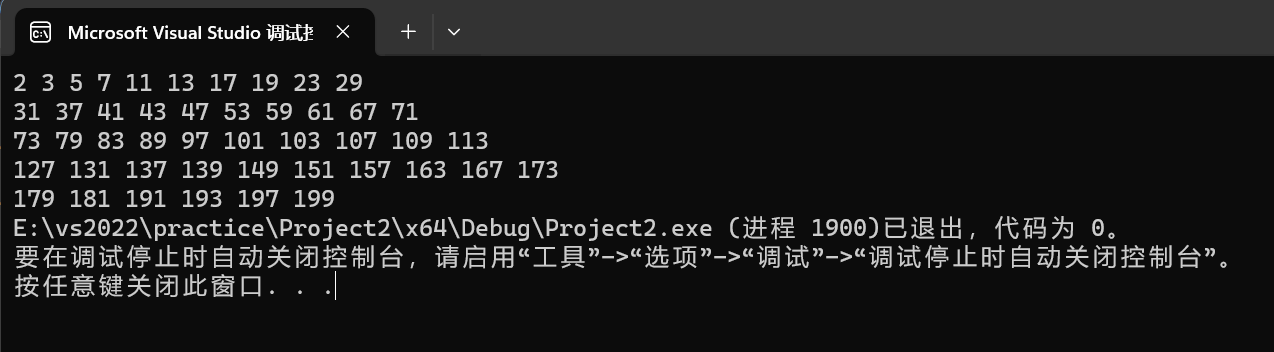
{

cout << endl;

round += 1;

}

}

}

**3.**

#include "mytemperature.h"

#include<iomanip>

int main()

{

cout << fixed << setprecision(2);

double cel0, fah0;

cout << "Celsius Fahrenheit | Fahrenheit Celsius" << endl;

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

{

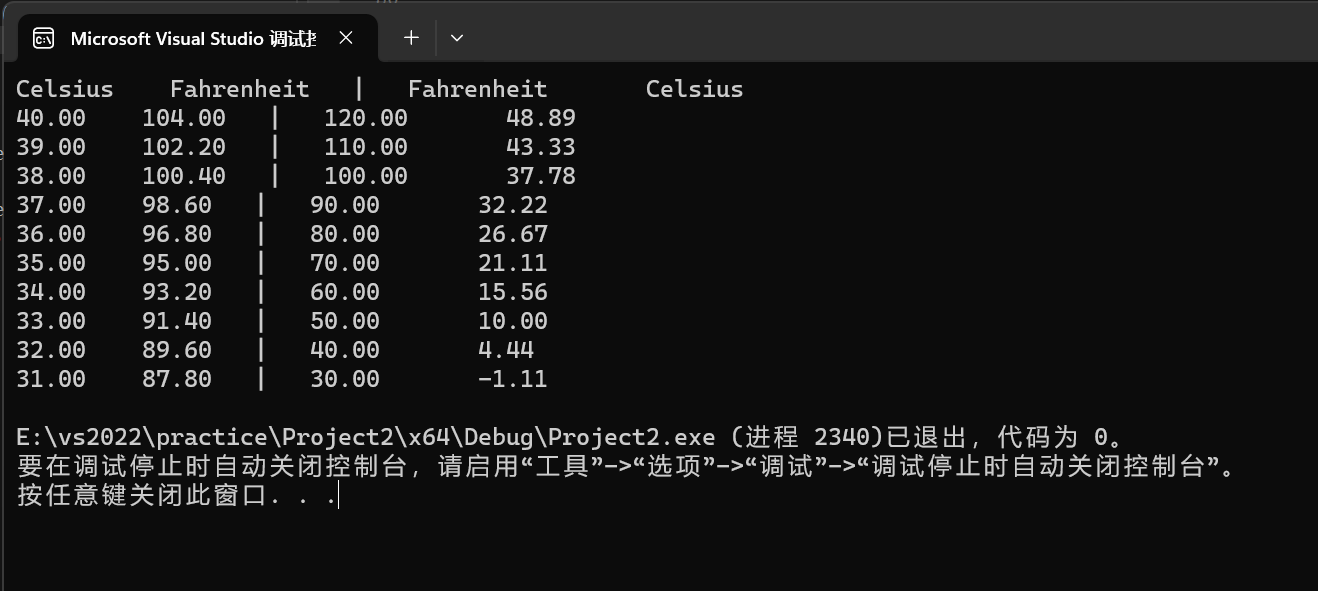
cel0 = 40.0 - i;

fah0 = 120.0 - 10 \* i;

cout << cel0 << " " << celsius\_to\_fah(cel0) << " | " << fah0 << " " << fahrenheit\_to\_cels(fah0) <<"\t"<< endl;

}

return 0;

}

**5.**

int main()

{

int m = 1;

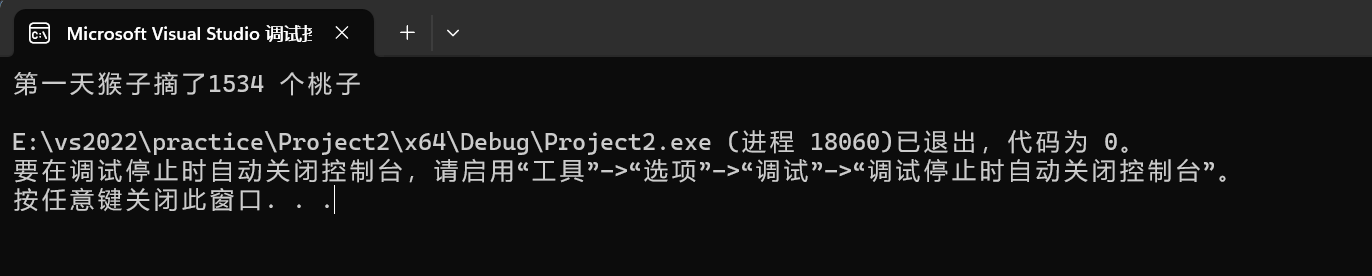
for (int i = 1; i < 10; i++)

m = (m + 1) \* 2;

cout << "第一天猴子摘了" << m << " 个桃子" << endl;

return 0;

}



**五、遇到的问题与解决方法**

1. **在第二个任务中，题目要求按照每行输出10个数据，原本认为实现比较困难，后来发现引入一个参数用于记录素数的个数，再配合if的判断即可确定何时换行**
2. **在第三个任务中，题目要求输出的数据只有最后一列是两位小数，其他则是输出一位小数，我最初写的版本是其他都符合，最后一列输出的则有四位小数，然后想到用setprecision来控制输出数据的有效位数，就索性全都按照两位有效数字输出了**
3. **在第五个任务中曾尝试用递归函数实现题目所需效果，发现自己对递归函数的理解并不够深入，很容易把自己绕进去，于是更改方式不使用递归函数而是直接在主函数中通过数学运算直接完成并得到预期结果**

**六、体会**

**函数能使主程序部分相当简洁，并且有需要时可以多次调用，十分方便。**

**在书写函数的过程中，如需实现一些额外的效果，引入变量通常会是一个不错的选择，能很大程度上简化代码，并增加可读性。**

**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<iostream>

using namespace std; //#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**

1. **数组**

**1.**

int main()

{

int arr[10];

int b = 1, num = 1,c=0;

cout << "Enter ten numbers:";

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

{

cin >> arr[(i >= b ? b : i)];

int c = 0;

for (int j = 0; j < (i >= b ? b : i); j++)

{

if (arr[b] != arr[j])

{

c += 1;

}

}

if (c == num)

{

b += 1;

num += 1;

}

}

cout << "The distinct numbers are:";

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

{

cout << arr[k] << " ";

}

return 0;

}

**2.**

int main()

{

double arr[10];

const int N = 10;

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

{

cin >> arr[i];

}

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

{

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

{

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

{

double temp;

temp = arr[j+1];

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

arr[j] = temp;

}

}

}

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

{

cout << arr[k] << ' ';

}

return 0;

}

**3.**

int main()

{

bool arr[100];

int num = 1;

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

{

arr[i] = true;

}

for (int i = 2; i < 100; i++)

{

for (int j = i; j < 100; j += (i+1))

{

arr[j] = !arr[j];

}

}

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

{

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

if (((i+1) / 10) == num)

{

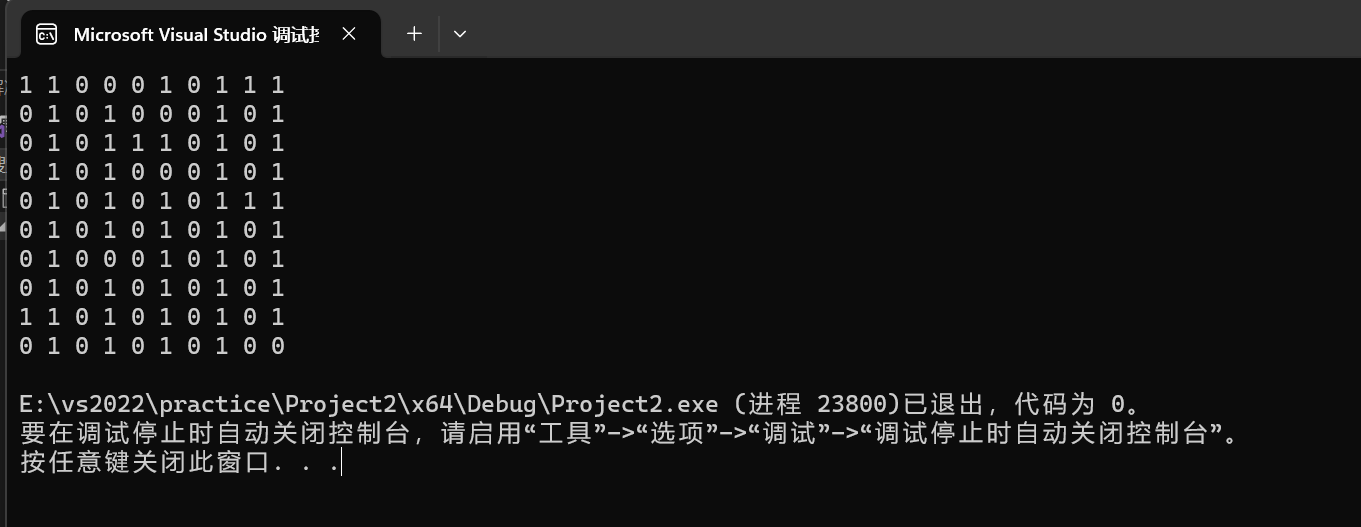
num += 1;

cout << endl;

}

}

return 0;

}

**4.**

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

{

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

{

list3[i] = list1[i];

}

for (int k = size1; k < size1 + size2; k++)

{

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

}

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

{

for (int j = 0; j < (size1 + size2) - 1 - i; j++)

{

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

{

int temp;

temp = list3[j + 1];

list3[j + 1] = list3[j];

list3[j] = temp;

}

}

}

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

{

cout << list3[i] << " ";

}

}

int main()

{

int list1[80] = {};

int list2[80] = {};

int list3[80] = {};

int size1,size2;

cout << "Enter listi:";

cin >> size1;

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

{

cin >> list1[i];

}

cout << "Enter list2:";

cin >> size2;

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

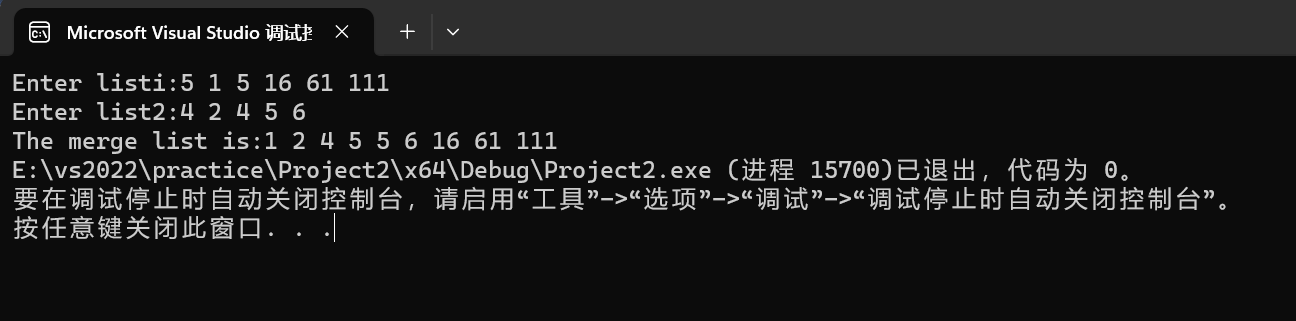
{

cin >> list2[i];

}

cout << "The merge list is:";

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

}

**5.**

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

{

int result = -1;

int size1 = 0, size2 = 0;

while (\*(s1 + size1)) //获取s1数组内有效字符个数

{

size1++;

}

while (\*(s2 + size2)) //获取s2数组内有效字符个数

{

size2++;

}

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

{

int num = 0; //用于记录两个数组连续相同字符的数量

if (result != -1)

{

break;

}

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

{

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

{

num++;

}

if (result != -1)

{

break;

}

if (num == size1)

{

result = i;

}

else

result = -1;

}

}

cout << "indexOf(" << '"';

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

{

cout << s1[i];

}

cout << '"' << ',' << '"';

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

{

cout << s2[i];

}

cout << '"' << ")is " << result;

return 0;

}

int main()

{

int size = 999;

char s1[999], s2[999];

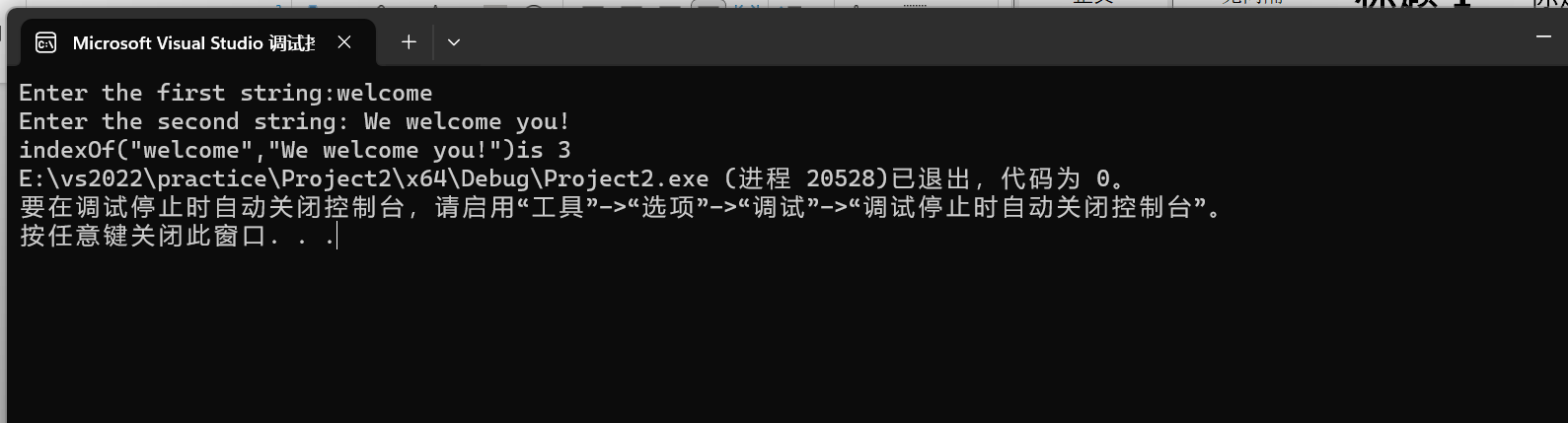
cout << "Enter the first string:";

cin.getline(s1, size);

cout << "Enter the second string: ";

cin.getline(s2, size);

indexOf(s1, s2);

 return 0;

}

**6.**

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

{

int size = 0;

while (\*(s + size)) //获取字符串长度

{

size++;

}

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

{

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

{

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

{

counts[j] = counts[j] + 1;

}

}

}

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

{

if (counts[i] != 0)

{

cout << static\_cast<char>(97 + i) << ":" << counts[i] << " times" << endl;

}

}

}

int main()

{

int counts[26] = {0};

char s[999];

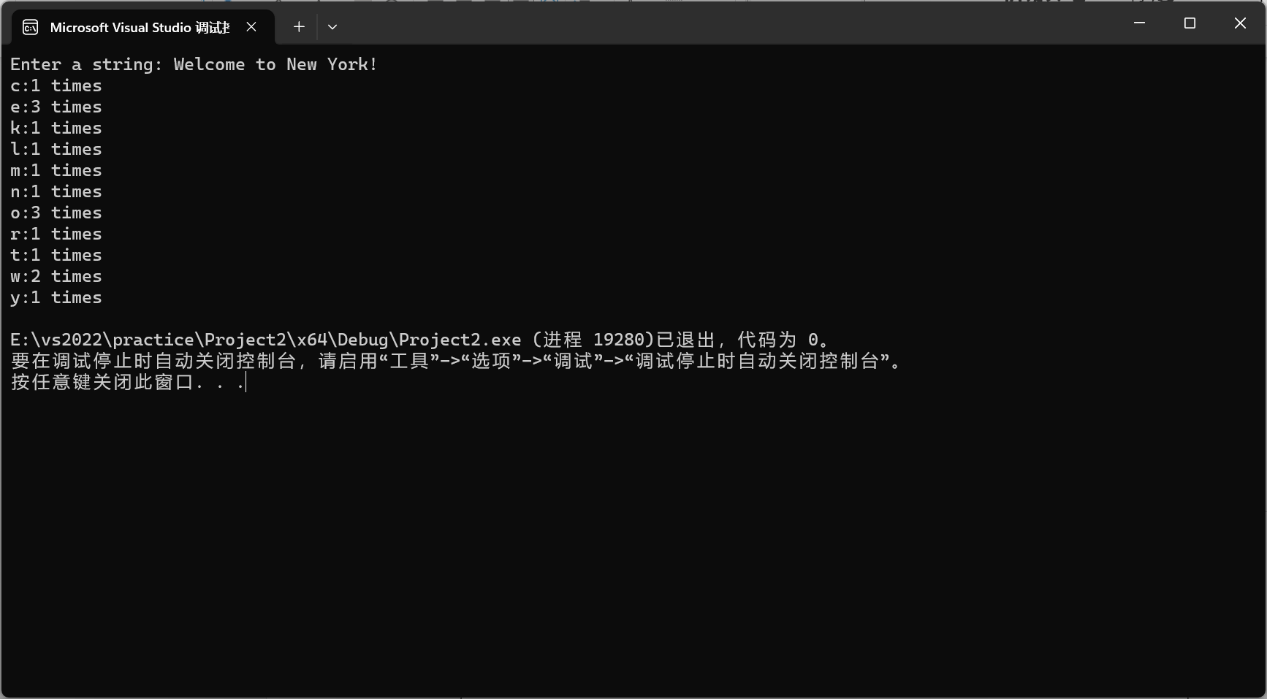
cout << "Enter a string: ";

cin.getline(s, 999);

count(s, counts);

return 0;

}



**（二）指针**

**1.**

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

{

int result = -1;

for (int i = 0; \*(s2+i) != 0; i++)

{

int num = 0;

if (result != -1)

{

break;

}

for (int j = 0; \*(s1+j) != 0; j++)

{

if(\*(s2+i+num)==\*(s1+j))

{

num += 1;

}

if (\*(s1+num) == 0)

{

result = i;

break;

}

}

}

return result;

}

int main()

{

char s1[999];

char s2[999];

int a;

cout << "Enter s1:";

cin.getline(s1, 999);

cout << "Enter s2:";

cin.getline(s2, 999);

a = indexof(s1, s2);

if (a != -1)

{

cout << "第一次匹配的下标为：" << a;

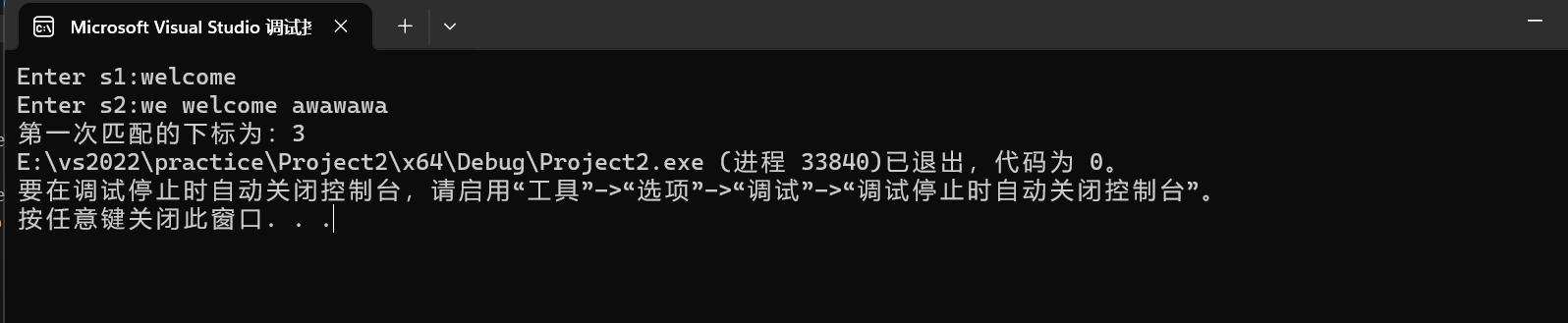
}

else

cout << a;

return 0;

}



**2.**

int parseHex(const char\* const hexString) //讲字符串表示的16进制数转化为10进制

{

int sum = 0, len = 0, j = -1, a = 0,b;

while (\*(hexString + len)) //获取字符串有效长度

{

len++;

}

for (int i = (len-1); \*(hexString + i) != 0; i--)

{

j++;

if (j > len - 1)

{

break;

}

switch (\*(hexString+i))

{

case 'A':

sum += 10 \* (pow(16, j));

break;

case 'B':

sum += 11 \* (pow(16, j));

break;

case 'C':

sum += 12 \* (pow(16, j));

break;

case 'D':

sum += 13 \* (pow(16, j));

break;

case 'E':

sum += 14 \* (pow(16, j));

break;

case 'F':

sum += 15 \* (pow(16, j));

break;

default:

a = static\_cast<int>(\*(hexString+i));

if (j == 0)

{

sum += (a - 48);

}

else sum += (a - 48) \* (pow(16, j));

break;

}

}

return sum;

}

int main()

{

char hex[999];

int a;

cout << "Enter your char:";

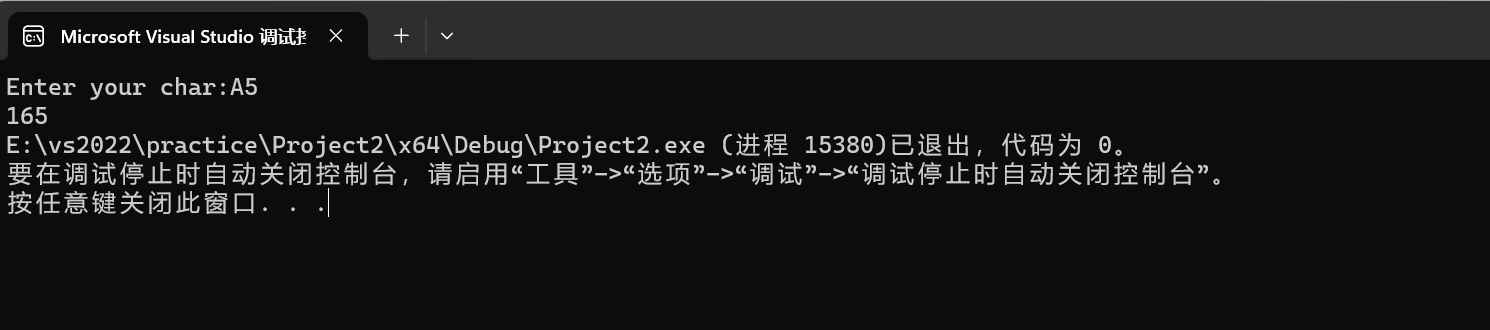
cin.getline(hex,999);

a = parseHex(hex);

cout << a;

return 0;

}



**3.**

void paixu(int size, int s[]) //冒泡序列

{

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

{

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

{

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

{

double temp;

temp = s[j + 1];

s[j + 1] = s[j];

s[j] = temp;

}

}

}

}

int main()

{

int size, a, b, c = 0;

cout << "Enter the amount of your list:";

cin >> size;

b = size;

int\* s = new int[size];

cout << "Enter your list:";

while (b)

{

b--;

cin >> a;

\*(s + c) = a;

c++;

}

cout << "输入你想获取的元素下标:";

cin >> b;

cout << "s[" << b << "]为:" << \*(s + b) << endl;

paixu(size, s);

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

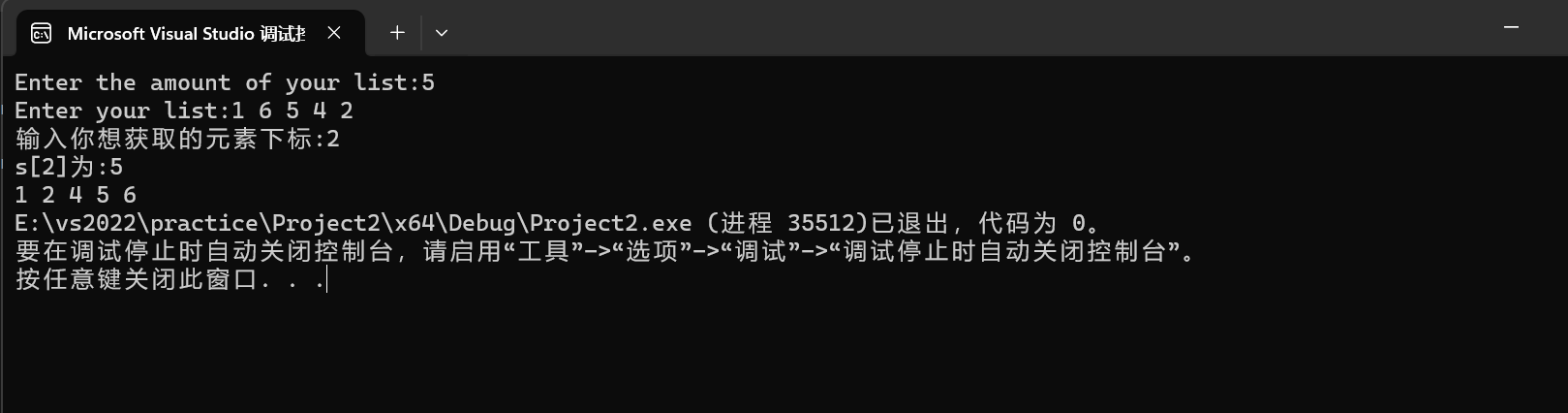
{

cout << \*(s + i) << ' ';

}

delete[] s;

return 0;

}

**四、遇到的问题与解决方法**

1. **数组**

**① 在检验字串的问题中，我首先遇到了获取数组有效长度的问题，即通过键盘键入了多少个元素。因为默认初始化的数组内容全都是0，未被赋值的部分则保留0，又因为0在循环条件的判定中被认为是false，利用以上两点，通过while循环来获取单个数组的有效长度，且代码简单有效。**

**② 在求字符串中每个字母出现的次数问题时，需要查ASCⅡ码对照，从而将字母与int型进行比较**

1. **指针**

**1.验证题**

**对于验证4：**

**2.程序设计题**

**① 原本直接写的例如s[1]之类的数据因函数参数类型的不同需改变形式，写法参考验证题2**

**② 对于第二题，我采取了switch case的模式，这一点本身应该可以优化成if的函数，有效缩短代码长度。除开这一点外，在写题过程中遇到了char类型转int类型的问题，通过引入新的参数来调试，获知与所需数字相差为48，然后在计算中提前减去这48即可**

**五、体会**

我在写程序的过程中难免出错，查错往往花费了大量的时间，通过引入新的参数来帮助查错能够有效缩短这个部分所需的时间（指针部分程序设计的第二题就是很好的例子）。除此之外，应该要对ASCⅡ码表中一些基本的东西进行记忆，比如大小写字母的差值，A代表的值等等。

遇到这些问题后，我的第一反应是自己反思代码问题所在，成功排除错误后，再去和同学讨论或者是在网络上搜索其他解法，通常能发现自己代码中不足的部分（还是指针部分程序设计的第二题），很多值得优化的地方。看多了之后，经验逐渐积累，日后写出来的程序应该也不会像现在一样冗余了吧。