**计算机程序设计基础（C++)**

**实验报告**

专业班级：软工2303班

学 号：8209230318

姓 名：李欣心

**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？

函数的返回值确保调用者能够正确地处理函数返回的值，若是函数返回值和函数类型不一，则会类型匹配错误，程序无法正常运行。

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

**四、算法分析，程序结果**

**1.**#include<iostream>

using namespace std;

int max(int& num1,int& num2)

{

for (int i = num1;; i++)

{

if (i % num1 == 0 && i % num2 == 0)

{

return i;

break;

}

}

}

int min(int& num1, int& num2)

{

for (int i = num2;; i--)

{

if (num1 % i == 0 && num2 % i == 0)

{

return i;

break;

}

}

}

int main()

{

int a, b;

cout << "请输入两个数值：";

cin >> a >> b;

if (a < b)

{

int temp = a;

a = b;

b = temp;

}

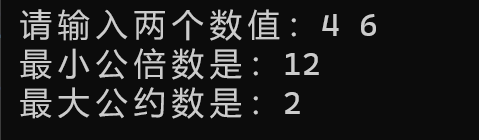
int c = max(a, b);

int d = min(a, b);

cout << "最小公倍数是："<<c<<'\n'<<"最大公约数是："<<d<<endl;

}

**运行结果如下：**



**2.**#include<iostream>

using namespace std;

bool is\_prime(int num)

{

for (int i = num-1; i >= 2; i--)

{

if (num % i == 0)

{

return false;

}

}

return true;

}

int main()

{

int k = 0;

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

{

if (is\_prime(i))

{

cout << i <<'\t';

k++;

if (k % 10 == 0)

{

cout << '\n';

}

while (k == 200)

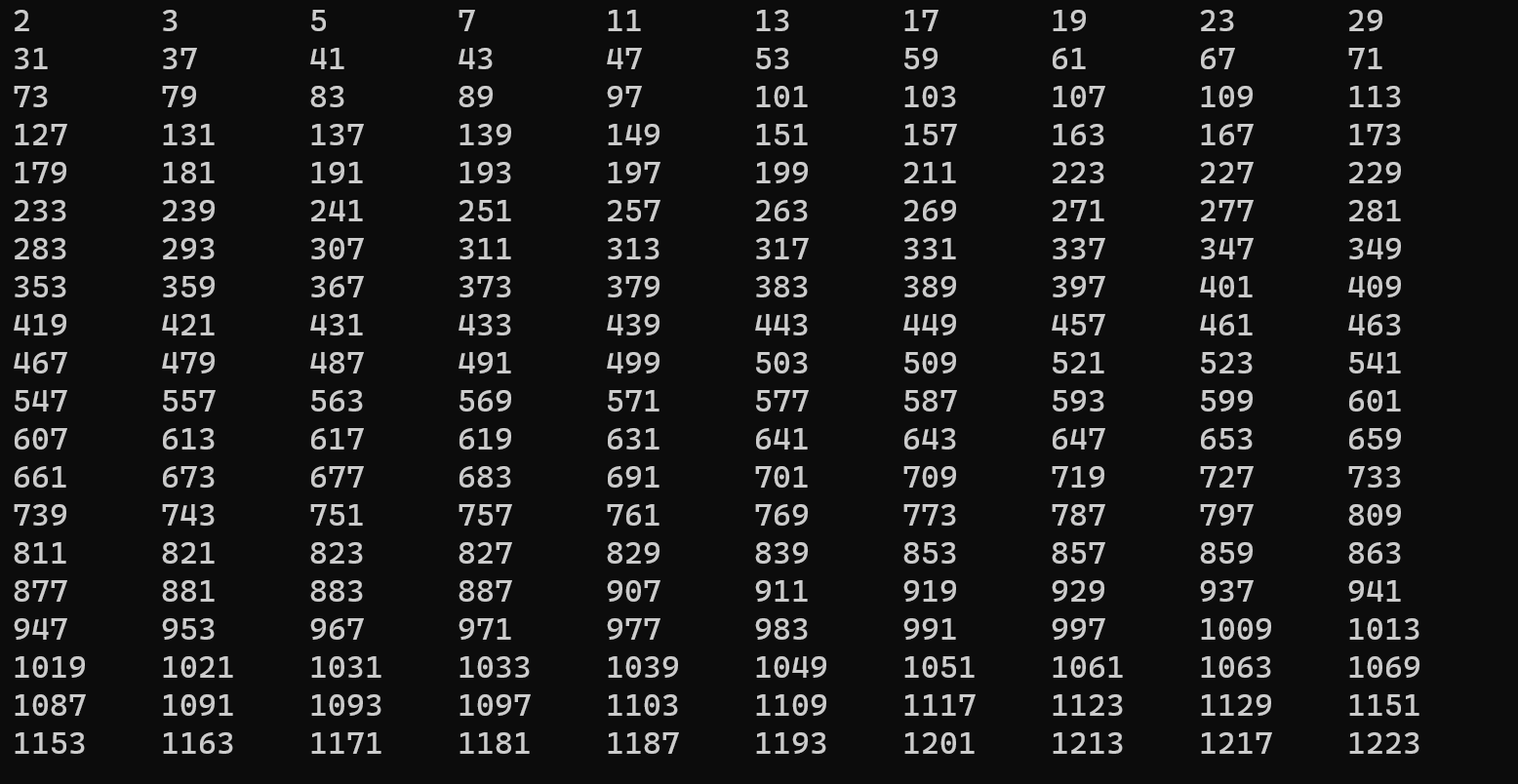
goto ret;

}

}

ret:return 0;

}

运行结果如下：

4.#include<iostream>

#include"mytriangle.h"

#include<cmath>

using namespace std;

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

{

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

return 1;

else return 0;

}

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

{

double s = (side1 + side2 + side3) / 2;

double square = sqrt(fabs((s - side1) \* (s - side2) \* (s - side3) \* s));

return square;

}

int main()

{

double num1, num2, num3;

cout << "请输入三角形三边长:";

cin >> num1 >> num2 >> num3;

bool c = is\_valid(num1, num2, num3);

if ((int)c==1)

cout<<"三角形的面积为"<< is\_area(num1, num2, num3)<<endl;

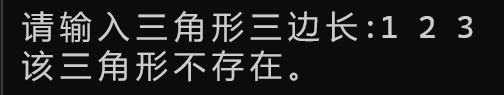
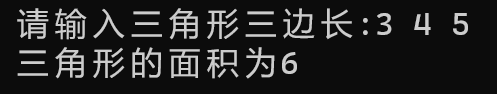
if ((int)c==0)

cout << "该三角形不存在。"<<endl;

return 0;

}

运行结果如下：

5.#include <iostream>

using namespace std;

int sum(int a)

{

a = (a + 1) \* 2;

return a;

}

int main()

{

int num = 1;

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

{

num=sum(num);

}

cout<<"第一天摘的桃子数量是："<<num << endl;

}

运行结果如下：



**五、遇到的问题与解决方法**

**1、在输出素数代码中，不知道如何通过if语句进行跳过步奏，导致了较长时间的卡顿，后想到了goto语句并成功运用其解决了问题。**

**2、三角形面积计算中的海伦公式在三角形不成立时为负值，导致数学符号错误，输出non，通过学习加入了fabs()，解决了问题。**

**六、体会**

**1、每道题目都有多种解决办法，不必拘泥于常规解法，思维多发散，可以收获更多的知识与喜悦。**

**2、知识是从前向后一路贯通的，要一步步学好，学以致用才能真正掌握。**

**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)加头文件#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; '\t'修改为'\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 “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;

}

改为：#include<iostream>

using namespace std;

int\* f()

{

int\* list = new int[4];

for (int i = 0, j = 1; i < 4; i++, j++)

list[i] = j;

return list;

}

int main()

{

const int size = 4;

int list1[size];

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**

1.#include<iostream>

using namespace std;

int main()

{

cout << "Enter ten numbers:";

int m[10];

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

cin >> m[i];

cout <<"The distinct numbers are:" << m[0] << " ";

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

{

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

{

if (m[i] != m[k])

{

if (k == 0)

{

cout << m[i] << " ";

}

}

else break;

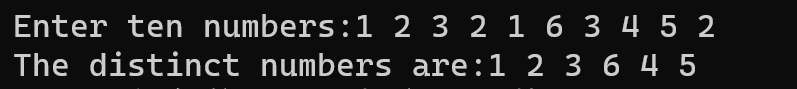
}

}

return 0;

}

运行结果如下：



2.#include<iostream>

using namespace std;

int main()

{

double list[10];

cout << "请输入十个要排序的数字：";

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

{

cin >> list[i];

}

bool changed = true;

do

{

changed = false;

for (int j = 0; j < 9; j++)

{

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

{

double temp = list[j];

list[j] = list[j + 1];

list[j+1] = temp;

changed = true;

}

}

}

while (changed);

cout << "排序后的值为：";

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

{

cout << list[i]<<" ";

}

cout << endl;

return 0;

}

运行结果如下：



3.#include<iostream>

using namespace std;

int main()

{

bool m[100];

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

{

m[i] = false;

}

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

{

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

{

if ((i+1)%j==0)

{

if (m[i])

m[i] = false;

else m[i] = true;

}

}

}

cout << "开着的柜子的编号是：";

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

{

if (m[i])

cout << i + 1 << " " ;

}

return 0;

}

运行结果如下：



4.#include<iostream>

using namespace std;

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

{

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

list3[j] = list1[j];

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

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

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

{

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

{

int temp;

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

{

temp = list3[j + 1];

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

list3[j] = temp;

}

if (list3[j] <= list3[j + 1])

{

}

}

}

}

int main()

{

int size1, size2, size3;

cout << "请分别输入两个数组元素个数，和由小到大排序号的数组元素" << endl;

cout << "Enter size1:";

cin >> size1;

cout << "Enter list1:";

int\* list1 = new int[size1+1];

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

cin >> list1[i];

cout << "Enter size2:";

cin >> size2;

cout << "Enter list2:";

int\* list2 = new int[size2+1];

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

cin >> list2[i];

size3 = size1 + size2;

int\* list3 = new int[size3+1];

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

cout << "The merged list is ";

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

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

delete[]list1;

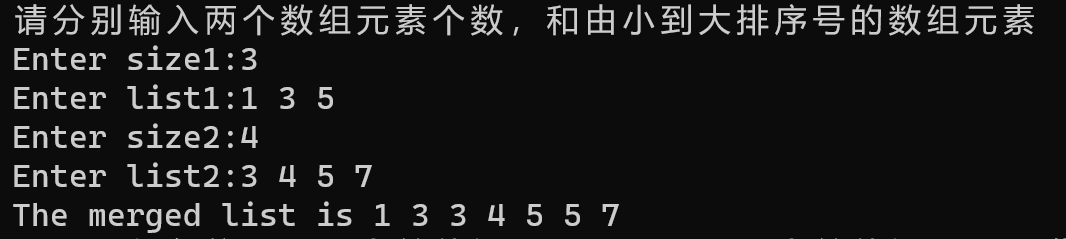
delete[]list2;

delete[]list3;

return 0;

}

运行结果如下：



5.#include<iostream>

#include<cstring>

using namespace std;

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

int main()

{

int size1 = 999, size2 = 99;

cout << "Enter the first string: ";

char\* s1 = new char[size1];

char\* s2 = new char[size2];

cin.getline(s1, size1);

cout << "Enter the second string: ";

cin.getline(s2, size2);

cout << "indexOf(\"" << s1 << "\", \"" << s2 << "\") is " << indexOf(s1, s2) << endl;

delete []s1;

delete []s2;

return 0;

}

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

{

int a, b, r, h=0;

a = strlen(s1);

b = strlen(s2);

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

{

int c = 0;

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

{

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

{

c++;

}

if (c == a)

{

h = 1;

r = i;

goto re;

}

}

re:if (h == 1)

{

goto ret;

}

}

ret:

{

if (h == 1)

{

return r ;

}

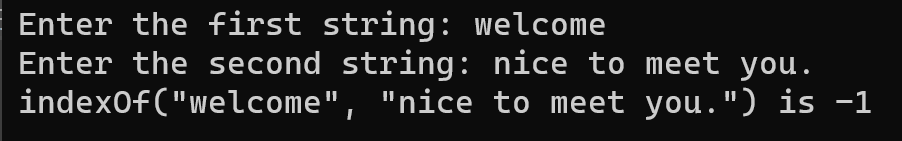
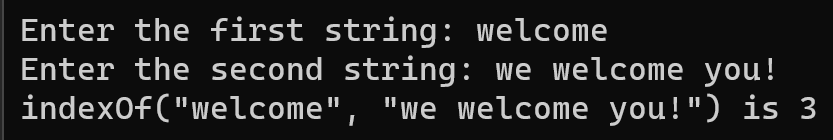
if (h == 0)

return -1;

}

}

运行结果如下：



6.#include<iostream>

using namespace std;

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

{

int b = 0;

const int num1 = 26;

const int num2 = 999;

for (int i = 0; i < num1; i++)

counts[i] = 0;

for (int i = 0; i < num2; i++)

{

counts[s[i] - 'a']++;

counts[s[i] - 'A']++;

}

for (int i = 0; i < num1; i++)

{

if (counts[i] != 0)

{

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

}

}

}

int main()

{

int size = 999;

int\* counts = new int[26];

cout << "Enter a string:";

char \*s = new char[size];

cin.getline(s, size);

count( s,counts);

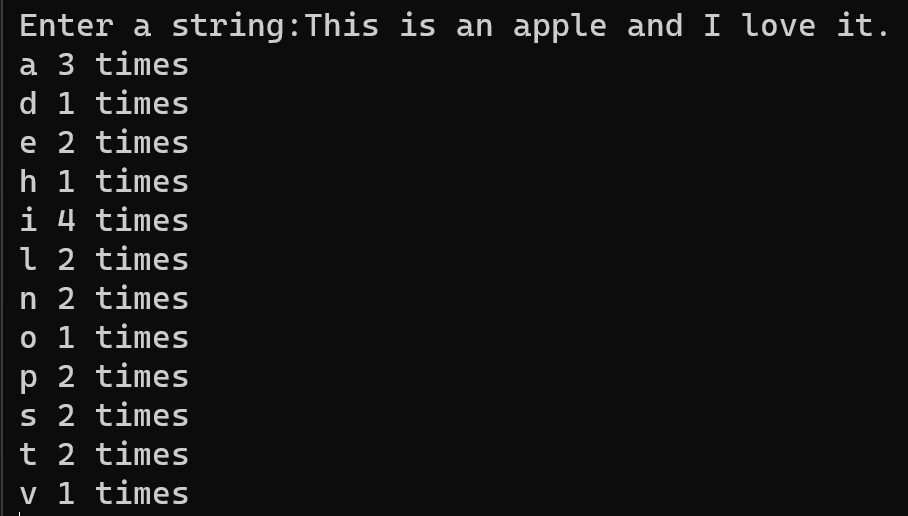
delete[]counts;

delete[]s;

return 0;

}

运行结果如下：



2.1#include<iostream>

#include<cstring>

using namespace std;

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

int main()

{

int size1 = 999, size2 = 99;

cout << "Enter the first string: ";

char\* s1 = new char[size1];

char\* s2 = new char[size2];

cin.getline(s1, size1);

cout << "Enter the second string: ";

cin.getline(s2, size2);

cout << "indexOf(\"" << s1 << "\", \"" << s2 << "\") is " << indexOf(s1, s2) << endl;

delete []s1;

delete []s2;

return 0;

}

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

{

int a, b, r, h=0;

a = strlen(s1);

b = strlen(s2);

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

{

int c = 0;

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

{

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

{

c++;

}

if (c == a)

{

h = 1;

r = i;

goto re;

}

}

re:if (h == 1)

{

goto ret;

}

}

ret:

{

if (h == 1)

{

return r ;

}

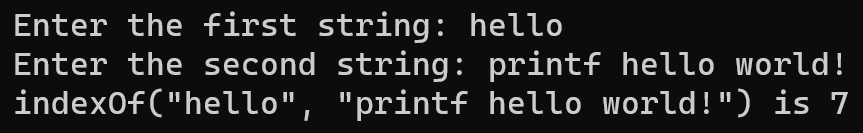
if (h == 0)

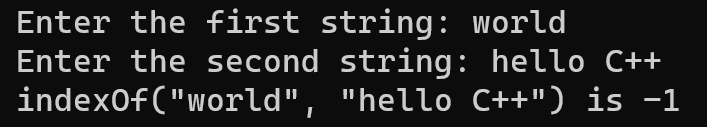
return -1;

}

}

运行结果如下：





2.2#include <iostream>

#include<cstring>

using namespace std;

int parseHex(const char\* const hexString)

{

int a, sum = 0;

a = strlen(hexString);

int \*list=new int [a];

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

{

if (hexString[i] >= 'A' && hexString[i] <= 'F')

list[i] = (static\_cast<int>(hexString[i]) - 55) \* (pow(16, a - 1 - i));

else

list[i] = (hexString[i] - 48) \* pow(16, a - i - 1);

sum = sum + list[i];

}

return sum;

}

int main()

{

int size = 999;

char \*a=new char[size];

cout << "请输入一个十六进制数:";

cin.getline(a, size);

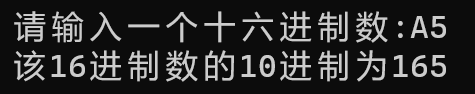
cout << "该16进制数的10进制为" << parseHex(a) << endl;

delete []a;

return 0;

}

运行结果如下：



2.3#include <iostream>

using namespace std;

void line( int size, int list[])

{

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

{

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

{

int temp;

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

{

temp = list[j + 1];

list[j + 1] = list[j];

list[j] = temp;

}

if (list[j] <= list[j + 1])

{

}

}

}

}

int main()

{

int size;

cout << "input size:";

cin >> size;

cout << "input list";

int\* list = new int[size];

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

{

cin >> list[i];

}

cout << "动态调试:" << endl;

int a;

cin >> a;

cout << "list[" << a << "]=" << \*(list + a - 1) << endl;

line(size, list);

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

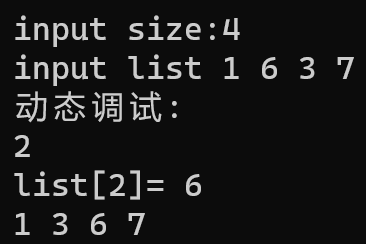
{

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

}

}

运行结果如下：



**四、遇到的问题与解决方法**

**1、在本次实验课中所写代码经常无报错但不可运行，提示栈出问题，后发现均是数组溢出情况导致，一一检查改正。在含有数组的程序一定要重视该问题。**

**2、无法用自己输入的变量对数组进行定义，后来学长统一指导，应当写为类型\* 数组名=new 类型 [变量]。**

**五、体会**

**1、写代码的时候一定要小心谨慎，相似内容在copy过程中一定要注意变量常量名字的更改，否则容易因为不该犯的小错误导致进度卡壳。**

**2、本次实验使对循环和指针有了进一步的了解，同时在每个小题中，题目所给的定义函数都是有具体意义的名称，这大大增强了代码的可理解性，我们不能单单为了自己的好写与否，就每次都将变量、函数名全部设为a,b,c,还是应当尽量使用恰当的名字。**