**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：8209240210

姓 名：王淇

**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**

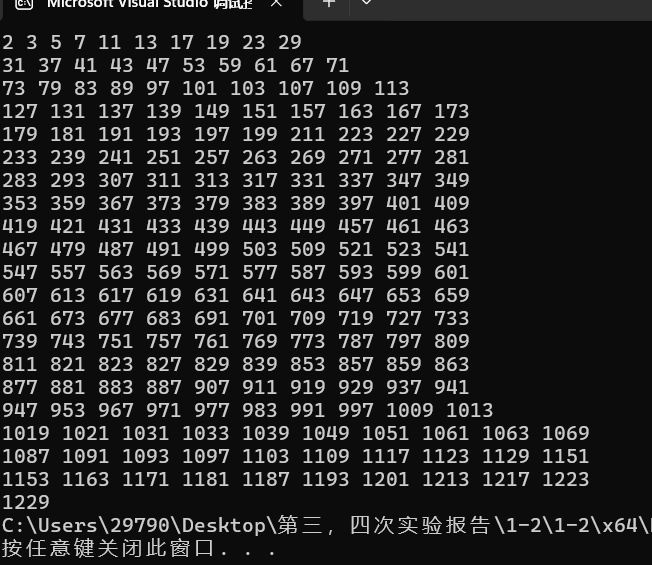
**1.**

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | #include<iostream>  #include<math.h>  using namespace std;  int m, n;  int yue1();  int bei1();  int main()  {      cin >> m >> n;//输入两个数      cout<<yue1()<<endl<<bei1();  **return** 0;  }  int yue1()//求最大公约数  {      int& m2 = m, & n2 = n;//引用      int minx = min(m2, n2);  **for** (int i = minx; i >= 1; i--)      {  **if** ((m2 % i== 0) && (n2 % i == 0))          {  **return** i;//一满足要求马上输出并停止循环          }      }  }  int bei1()//求最小公倍数  {      int& m1 = m, & n1 = n;      int maxx = max(m1, n1);  **while** (1)      {  **if** ((maxx % m1 == 0) && (maxx % n1 == 0))          {  **return** maxx;//一满足要求马上输出并停止循环          }          maxx++;      }  } | |



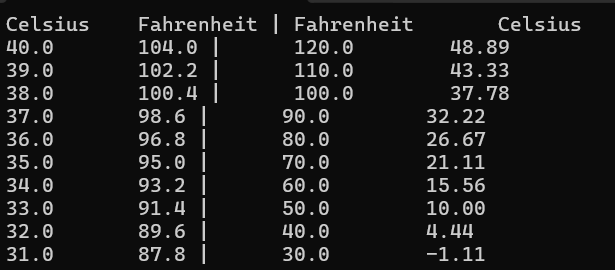
2.

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | #include<iostream>  using namespace std;  bool is\_prime(int num);  int main()  {      int number = 0,j=2;      bool tmp = 0;  **while** (number <= 200)      {          tmp = 0;  **if** (is\_prime(j) == **true**)          {              cout << j << " ";              number++;              tmp = 1;          }  **if** ((number % 10 == 0)&&(number>=10)&&(tmp==1))cout << endl;          j++;      }  **return** 0;  }  bool is\_prime(int num)  {  **for** (int i = 2; i < num; i++)      {  **if** (num % i == 0)          {  **return** **false**;          }      }  **return** **true**;  } | |



**3.**

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | #include<iostream>  using namespace std;  #include"mytemperature.h"  int main()  {      double j = 120.0,n=10;      cout << "Celsius    Fahrenheit | Fahrenheit       Celsius" << endl;  **for** (double i = 40.0; i >= 31.0; i--)      {          printf("%.1f", i);          cout << "       ";          printf("%.1f", celsius\_to\_fah(i));          cout<< " |      ";          printf("%.1f", j);          cout << "        ";          printf("%.2f", fahrenheit\_to\_cels(j));          cout<<endl;          j = j - n;      }  **return** 0;  } | |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | #include<iostream>  using namespace std;  double celsius\_to\_fah(double cel)  {      double fah;      fah = cel \* 1.8 + 32;  **return** fah;  }  double fahrenheit\_to\_cels(double fah)  {      double cel;      cel = (fah - 32) / 1.8;  **return** cel;  } | |

****

**5.**

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | #include<iostream>  using namespace std;  int taozi(int num);  int number = 0;  int main()  {      cout << taozi(1);//桃子最后只有1个  **return** 0;  }  int taozi(int num)  {      number++;  **if** (number == 10)//递归十次      {  **return** num;      }  **return** taozi((num + 1) \* 2);   //调用自己，同时加1乘2  } | |

****

**五、遇到的问题与解决方法**

**1.最重要的问题：实验报告的美观**

由于之前一直被代码的插入会打乱报告的格式而困扰，所以我先花了大概20分钟在csdn上寻找答案，最后找到了如图的哪些优秀网站，能把代码正确显色，还原其在编译器里的状态，然后我又想到可以利用表格将代码和文字分割开，创造出类似markdown里代码区的效果，使实验报告的美观和可读性大大提升

****

**2.粗心问题的存在极大减缓了做题的速度**

比如第二题输出时需要每十个换行，然而我发现前十个却是单独换行的，仔细思考后才明白输出条件中i%10==0，前十个都可以直接满足，所以必须加上一个额外限制i>=10，才能正常输出

**3.课后要及时复习新知识**

在写第三题时我发现已经不记得怎么调用头文件里写的函数了，所以在实验课结束后重新认真看了一遍那一节的知识才写出来

**六、体会**

从这次实验报告美观性的改进，我悟到我们软件开发中也要为客户考虑，才能被市场认可。

在当今竞争激烈的软件市场中，软件要为客户考虑已成为成功的关键要素。一款真正优秀的软件，从设计到开发，再到后续的维护与升级，都必须将客户的需求、体验和利益放在首位。

从需求角度出发，深入了解客户在特定场景下的工作流程和期望达成的目标至关重要。例如，一款办公软件若要获得青睐，就需要精准把握办公人员日常处理文档、数据统计、团队协作等方面的痛点，从而提供诸如便捷的文档编辑工具、高效的数据处理功能以及流畅的协作交流平台等。只有满足客户的实际需求，软件才能在众多同类产品中脱颖而出。

在使用体验上，软件的界面设计应简洁直观，操作流程要符合人体工程学和用户习惯。避免过于复杂的菜单设置和晦涩难懂的交互方式，确保无论是专业技术人员还是普通用户都能轻松上手。例如，许多知名的手机应用程序，其界面简洁美观，功能布局合理，用户仅需简单的滑动、点击操作就能完成各项任务，大大提高了用户的满意度和忠诚度。

软件的稳定性和可靠性也是为客户考虑的重要体现。频繁的崩溃、错误提示以及数据丢失等问题会给客户带来极大的困扰和损失。因此，开发团队必须进行严格的测试，确保软件在各种环境和条件下都能稳定运行。同时，及时的技术支持和更新服务也不可或缺，当客户遇到问题时能够迅速得到解决，并且软件能够根据客户反馈和技术发展不断优化升级。

总之，软件若想在市场中立足并取得长远发展，必须以客户为核心。将客户的需求贯穿于整个软件生命周期，用心打造出易用、稳定、高效的产品，才能赢得客户的信任与口碑，从而在激烈的竞争中立于不败之地。

**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 the first string: welcome

Enter the second string: We invite you!

indexOf(“welcome”, “We invite you!”) is –1

Enter

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;

}

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.**

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | #include<iostream>  using namespace std;  int number[20];//定义储存数的数组  bool search(int num,int length);  int main()  {      int num,length=0;//length为已填充进数组的数的数量  **for** (int i = 1; i <= 10; i++)      {          cin >> num;  **if** (search(num,length) ==**true**)//判断是否为新数          {              length++;              number[length] = num;          }      }  **for** (int i = 1; i <= length; i++)      {          cout << number[i]<<" ";      }  **return** 0;  }  bool search(int num, int length)  {  **for** (int i = 1; i <= length; i++)//遍历数组      {  **if** (number[i] == num)//如果相同，输出false          {  **return** **false**;          }      }  **return** **true**;  } | |

****

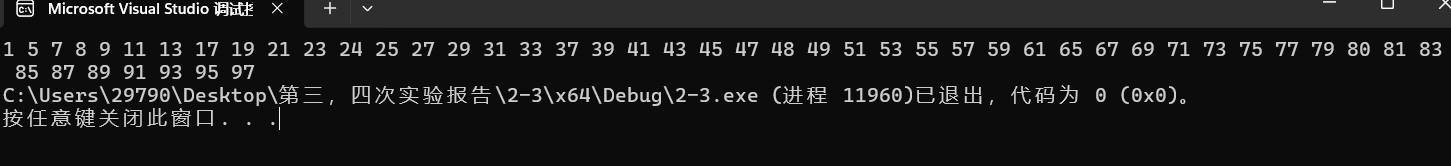
**2.**

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | #include<iostream>  using namespace std;  double num[20];  int main()  {      double tmp;//提前定义中间值tmp  **for** (int i = 0; i < 10; i++)      {          cin >> num[i];      }  **for** (int i = 0; i < 10; i++)      {  **for** (int j = i; j < 9; j++)//注意最大是9不是10          {  **if** (num[j] > num[j + 1])//如果前面的数比后面的数大              {                  tmp = num[j];                  num[j] = num[j + 1];                  num[j + 1] = tmp;              }//交换位置          }      }  **for** (int i = 0; i < 10; i++)      {          cout << num[i] << " ";//输出      }  **return** 0;  } | |

****

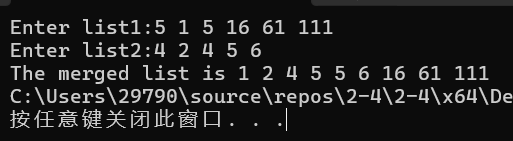
**3.**

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | #include<iostream>  using namespace std;  bool box[110];  void change(int begin, int gap);  int main()  {  **for** (int i = 1; i <= 100; i++)box[i] = **true**;  **for** (int i = 2; i <= 100; i++)      {          change(i, i+1);//改变状态，注意gap要加1      }  **for** (int i = 1; i <= 100; i++)      {  **if** (box[i] == **true**)cout << i << " ";      }  **return** 0;  }  void change(int begin, int gap)  {  **for** (int i = begin; i <= 100; i += gap)      {          box[i] = !box[i];//易错点      }  } | |

****

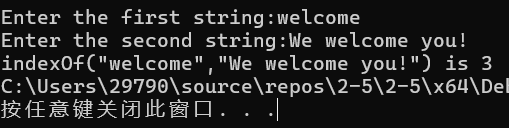
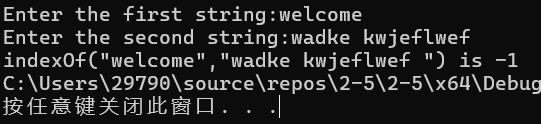
**4.**

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46 | #include<iostream>  using namespace std;  int num[1000];  int main()  {      int l1, l2,max=0,min=1002;      int n1[100], n2[100];      cout << "Enter list1:";      cin >> l1;  **for** (int i = 1; i <= l1; i++)      {          cin >> n1[i];          num[n1[i]]++;//运用计数排序的思路  **if** (n1[i] > max)//找出n1和n2数组中的最大值          {              max = n1[i];          }  **if** (n1[i] < min)          {              min = n1[i];          }      }      cout << "Enter list2:";      cin >> l2;  **for** (int i = 1; i <= l2; i++)      {          cin >> n2[i];          num[n2[i]]++;  **if** (n2[i] > max)          {              max = n2[i];          }  **if** (n2[i] < min)//找出n1和n2数组中的最小值          {              min = n2[i];          }      }      cout  << "The merged list is ";  **for** (int i = min; i <= max; i++)      {  **for** (int j = 1; j <= num[i]; j++)//最终输出          {              cout << i<< " ";          }      }  } | |

****

**5.**

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48 | #include<iostream>  #include<string>  using namespace std;  char s1[100], s2[100];//定义好两个数组  int indexOf(int l1,int l2);  int main()  {      cout << "Enter the first string:";      cin.getline(s1, 100);//保证一句话能够完整输入      int l1 = strlen(s1);      cout << "Enter the second string:";      cin.getline(s2, 100);//保证一句话能够完整输入      int l2 = strlen(s2);      cout<<"indexOf(\"";  **for** (int i = 0; i < l1; i++)cout << s1[i];      cout << "\",\"";  **for** (int i = 0; i < l2; i++)cout << s2[i];      cout << "\") is ";      cout << indexOf(l1,l2);  **return** 0;  }  int indexOf(int l1,int l2)  {      bool tmp = **true**,tmp2=**false**;//标记      int gap = 0;//计算空格数  **for** (int i = 0; i < l2 - l1; i++)      {  **if** (s2[i] == ' ')gap++;//遇到空格加1  **if** (s2[i] == s1[0])          {              tmp2 = **true**;//标记改变  **for** (int j = 0; j < l1; j++)              {  **if** (s2[i + j] != s1[j])                  {                      tmp = **false**;  **break**;                  }              }          }  **if** ((tmp == **true**)&&(tmp2==**true**))//满足条件          {  **return** i -gap+1;          }          tmp = **true**; tmp2 = **false**;//标记重置      }  **return** -1;  } | |

**** ****

6.

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | #include<iostream>  #include<string>  using namespace std;  int number[30];  int main()  {      cout << "Enter a string:";      char str[100];      cin.getline(str, 100);//输入一句话      int len = strlen(str);//计算长度  **for** (int i = 0; i < len; i++)      {          int asc = (int)str[i];  **if** ((asc >= 65) && (asc <= 90))//如果是大写字母          {              int asc1 = asc - 65;              number[asc1]++;          }  **if** ((asc >= 97) && (asc <= 122))//如果是小写字母          {              int asc2 = asc - 97;              number[asc2]++;          }      }  **for** (int i = 0; i < 26; i++)//遍历数组      {  **if** (number[i] != 0)          {              cout << (char)(i + 97) << ":";//还原为小写字母              cout << number[i] << " times" << endl;          }      }  **return** 0;  } | |

****

**7.指针代码修改部分：**

**（1）void main()应该改成int main()**

**(3) 递归未讲，可留待后做**

**（4）**

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | #include<iostream>  using namespace std;  int\* f()  {      int \*list = **new** int(4);  **for** (int i = 0; i <= 3; i++)      {          list[i] = i+1;      }  **return** list;  }  void main()  {      int\* p = f();      cout << p[0] << endl;      cout << p[1] << endl;  **delete**[] p;  } | |

**1.**

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60 | #include<iostream>  #include<string>  using namespace std;  char s1[100], s2[100];  int indexOf(const char\* s1, const char\* s2);    int main()  {      cout << "Enter the first string:";      cin.getline(s1, 100);      int l1 = strlen(s1);      cout << "Enter the second string:";      cin.getline(s2, 100);      int l2 = strlen(s2);      cout << "indexOf(\"";  **for** (int i = 0; i < l1; i++)          cout << s1[i];      cout << "\",\"";  **for** (int i = 0; i < l2; i++)          cout << s2[i];      cout << "\") is ";      cout << indexOf(s1, s2);  **return** 0;  }    int indexOf(const char\* s1, const char\* s2)  {      int l1 = strlen(s1);      int l2 = strlen(s2);      bool tmp = **true**, tmp2 = **false**;      int gap = 0;  **for** (int i = 0; i <= l2 - l1; i++)      {  **if** (\*(s2 + i) == ' ')              gap++;  **if** (\*(s2 + i) == \*(s1 + 0))          {              tmp2 = **true**;  **for** (int j = 0; j < l1; j++)              {  **if** (i + j >= l2)  // 添加边界判断，防止越界访问  **break**;  **if** (\*(s2 + i + j) != \*(s1 + j))                  {                      tmp = **false**;  **break**;                  }              }          }  **if** ((tmp == **true**) && (tmp2 == **true**))          {  **return** i - gap + 1;          }          tmp = **true**;          tmp2 = **false**;      }  **return** -1;  } | |

****

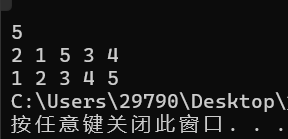
**2.**

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | #include<iostream>  #include<string>  using namespace std;  char num[100];  int parseHex(int l1);  int main()  {      cin.getline(num, 100);      int l1 = strlen(num);      cout << parseHex(l1);  **return** 0;  }  int parseHex(int l1)  {      int number = 0;  **for** (int i = 0; i < l1; i++)      {          int asc = (int)num[i];//转换成asc码          int tmp=1;  **for** (int j = 0; j < l1 - 1 - i; j++)          {              tmp = tmp \* 16;          }  **if** ((asc >= 65) && (asc <= 70))//如果是A到F          {              number += (asc - 55) \* tmp;          }  **else** {              number += (asc-48) \* tmp;//如果是1到9          }          tmp = 16;//复原      }  **return** number;  } | |

****

**3.**

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48 | #include<iostream>  using namespace std;  void NumberSort(int\* arr,int max);  int n;  int main()  {      cin >> n;      int max = 0;      int\* arr = **new** int[n];  **for** (int i = 0; i < n; i++)      {          int tmp;          cin >> tmp;  **if** (tmp > max)          {              max = tmp;//记录下最大值          }          \*(arr + i) = tmp;      }      NumberSort(arr,max);//使用选择排序  **for** (int i = 0; i < n; i++)      {          cout << \*(arr + i)<<" ";//输出      }  **delete**[] arr;//释放内存  **return** 0;  }  void NumberSort(int\* arr,int max)  {      int min = max,minn,tmp;  **for** (int i = 0; i < n; i++)      {  **for** (int j = i; j < n; j++)//记录下每轮最小值          {  **if** (\*(arr + j) <min)              {                  min = \*(arr+j);                  minn = j;              }          }          tmp = \*(arr + minn);          \*(arr + minn) = \*(arr + i);          \*(arr + i) = tmp;//将最小值丢到最前面          min = max;//还原      }  } | |

****

**四、遇到的问题与解决方法**

**1.不记得全局变量中bool默认是false还是true**

最后询问ai得知

****

**2.不确定空格在char中怎么表示**

在第五题判断是否为字串时，由于最后要输出索引，所以要计算之前空格的出现数，

因此在循环的开始要先判断是否为空格

****

**五、体会**

通过本次实验课，我深刻认识到编程能力的提高离不开大量的上机练习。

编程，作为一门实践性极强的技能，其能力的提升必须通过大量的上机练习来实现。理论知识固然重要，但它仅仅是构建编程大厦的基石，只有在实际的上机操作中，才能将这些知识转化为真正的编程能力。

上机练习能够让编程者更加深入地理解编程语言的语法规则和逻辑结构。在书本上看到的代码示例，只有亲手输入到计算机中，运行、调试，才能真切地体会到每个语句的作用和执行顺序。例如，对于循环结构的理解，仅仅背诵其语法格式是远远不够的，只有在编写各种不同类型的循环程序，如计算数列求和、遍历数据集合等，并且在调试过程中解决可能出现的索引越界、循环条件错误等问题时，才能真正掌握循环的精髓，从而在实际项目中灵活运用。

大量的上机练习有助于培养编程思维和解决问题的能力。在编程过程中，常常会遇到各种各样的错误和异常，这些问题的解决过程就是编程思维锻炼的过程。通过不断地尝试、分析错误信息、查阅资料、修改代码，编程者学会了如何将复杂的问题分解成一个个小的、可解决的部分，如何从不同的角度思考解决方案，以及如何优化代码提高效率。每一次成功解决一个编程难题，都是一次编程思维的成长，而这种成长离不开大量的实践积累。

总之，编程能力的提升没有捷径可走，只有通过坚持不懈地进行大量上机练习，在实践中不断积累经验、总结教训，才能逐步提高编程水平，从一个编程新手成长为一名优秀的开发者，在编程的世界里游刃有余地创造出高质量的软件作品。