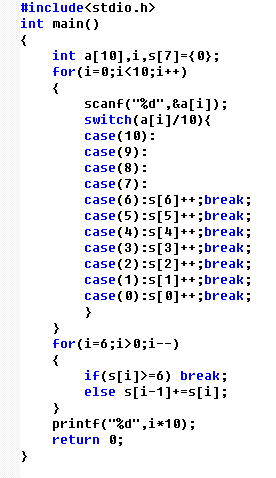
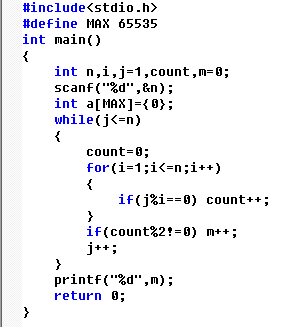
样题-初级题：从考试成绩中划出及格线  
10个学生考完期末考试评卷完成后，A老师需要划出及格线，要求如下：  
(1) 及格线是10的倍数；  
(2) 保证至少有60%的学生及格；  
(3) 如果所有的学生都高于60分，则及格线为60分



样题-中级题：亮着电灯的盏数   
一条长廊里依次装有n(1 ≤ n ≤ 65535)盏电灯，从头到尾编号1、2、3、…n-1、n。每盏电灯由一个拉线开关控制。开始，电灯全部关着。  
有n个学生从长廊穿过。第一个学生把号码凡是1的倍数的电灯的开关拉一下；接着第二个学生把号码凡是2的倍数的电灯的开关拉一下；接着第三个学生把号码凡是3的倍数的电灯的开关拉一下；如此继续下去，最后第n个学生把号码凡是n的倍数的电灯的开关拉一下。n个学生按此规定走完后，长廊里电灯有几盏亮着。  
注：电灯数和学生数一致。

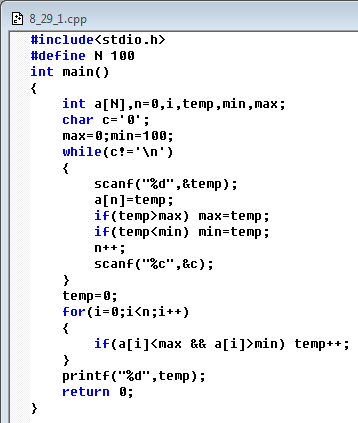


样题-高级题：地铁换乘  
已知2条地铁线路，其中A为环线，B为东西向线路，线路都是双向的。经过的站点名分别如下，两条线交叉的换乘点用T1、T2表示。编写程序，任意输入两个站点名称，输出乘坐地铁最少需要经过的车站数量（含输入的起点和终点，换乘站点只计算一次）。  
地铁线A（环线）经过车站：A1 A2 A3 A4 A5 A6 A7 A8 A9 T1 A10 A11 A12 A13 T2 A14 A15 A16 A17 A18  
地铁线B（直线）经过车站：B1 B2 B3 B4 B5 T1 B6 B7 B8 B9 B10 T2 B11 B12 B13 B14 B15

# 8.29去掉最大值、最小值之后剩下的个数

**1、输入一串数，以','分隔，输出所有数中去掉最大值、最小值之后剩下的个数。（其中最大值与最小值可能有多个）**

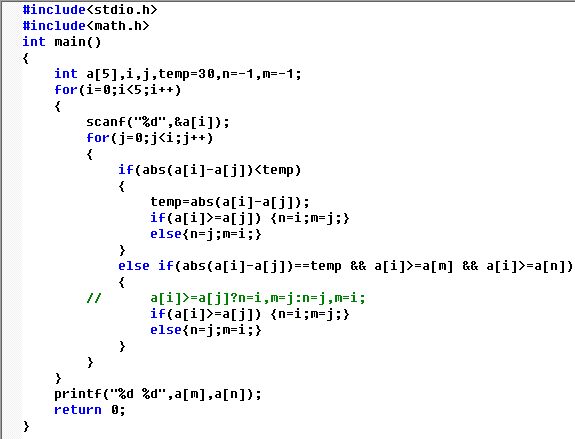
Smple input：3,3,5,3,6,9,7,9 Sample outPut: 3



# 8.29从5个人中选取2个人作为礼仪

**2、要从5个人中选取2个人作为礼仪，其中每个人的身高范围为160-190，要求2个人的身高差值最小（如果差值相同的话，选取其中最高的两人），以升序输出两个人的身高。**

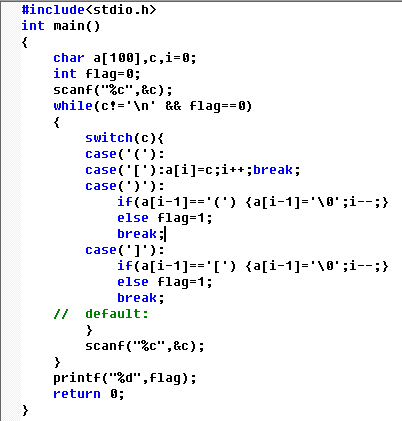
Smple input：161 189 167 172 188 Sample outPut: 188 189



# 8.29验证括号是否匹配

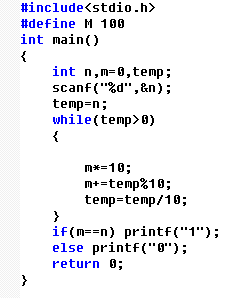
**3、输入一串字符串，其中有普通的字符与括号组成（包括‘（’、‘）’、‘[’,']'）,要求验证括号是否匹配，如果匹配则输出0、否则输出1.**

Smple input：dfa(sdf)df[dfds(dfd)] Smple outPut:0



# 8.31回文数

1. 判断回文数，是返回1



# 8.31将第一行中含有第二行中“23”的数输出并排序

2. 输入一行数字：123 423 5645 875 186523  
在输入第二行：23

将第一行中含有第二行中“23”的数输出并排序  
结果即：123 423 186523



# 8.31翻译电话号码

3. 将 电话号码 one two 。。。nine zero  
翻译成1  2 。。9 0

中间会有double

例如输入：OneTwoThree  
输出：123

输入：OneTwoDoubleTwo  
输出：1222

输入：1Two2 输出：ERROR

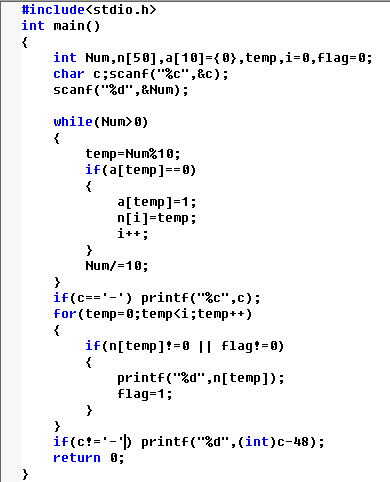
输入：DoubleDoubleTwo 输出：ERROR

第三题：有空格，非法字符，两个Double相连，Double位于最后一个单词 都错误



# 9.1.AM 将整数倒序输出，剔除重复数据

输入一个整数，如12336544，或1750，然后从最后一位开始倒过来输出，最后如果是0，则不输出，输出的数字是不带重复数字的，所以上面的输出是456321和571。如果是负数，比如输入-175，输出-571。



# 9.1.AM 大数相减

输入两行字符串正整数，第一行是被减数，第二行是减数，输出第一行减去第二行的结果。

备注：1、两个整数都是正整数，被减数大于减数

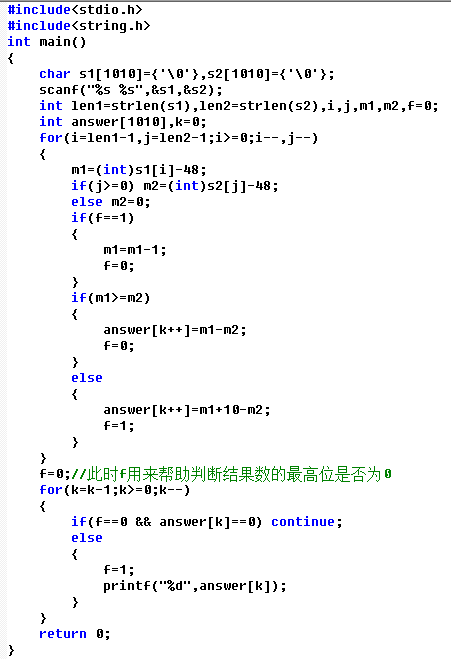
示例：

输入：1000000000000001

1

输出：1000000000000000

注意大数用char a[] 存储，用%s接收，一位一位的运算。注意a[0]里的正负号。



# 9.1.AM 判断if语句括号是否合法

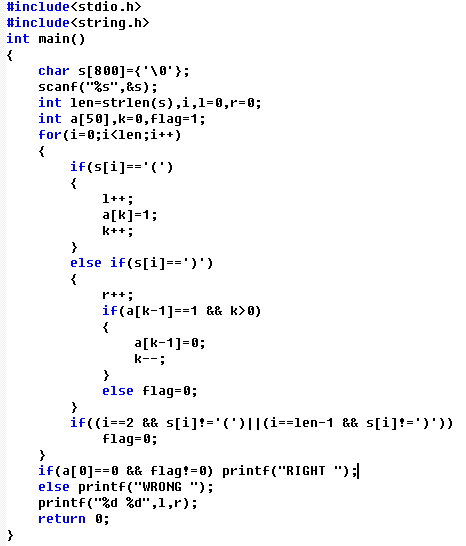
编程的时候，if条件里面的“(”、“)”括号经常出现不匹配的情况导致编译不过，请编写程序检测输入一行if语句中的圆括号是否匹配正确。同时输出语句中出现的左括号和右括号数量，如if((a==1)&&(b==1))是正确的，而if((a==1))&&(b==1))是错误的。注意if语句的最外面至少有一对括号。提示：用堆栈来做。

输入：if((a==1)&&(b==1))

输出：RIGTH 3 3

输入：if((a==1))&&(b==1))

输出：WRONG 3 4



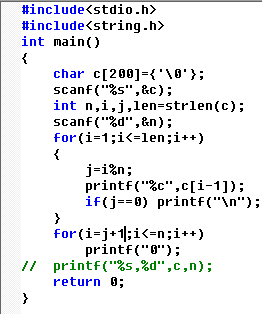
# 9.1.PM 字符串M化成以N为单位的段

输入m个字符串 和一个整数n, 把字符串M化成以N为单位的段，不足的位数用0补齐。

如 n=8 m=9 ，

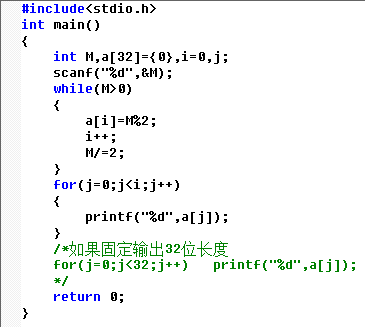
123456789划分为：12345678  
90000000

123化为 ：12300000



9.1.PM 整数化为2进制数，32位长度。然后逆序输出

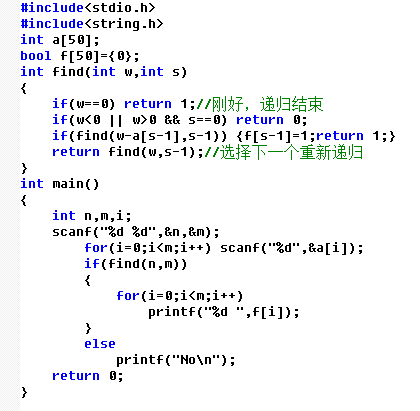
整数化为2进制数，32位长度。然后逆序输出



# 9.1.PM 背包

01背包

给定一个数，比如20  
然后 再给定几个数字 1 3 5 7 8  
1  3  5  7  8  
0  0  1  1  1   
因为5+7+8=20



 \*1、 输入一个字符串，将其各个字符对应的ASCII值加5后，输出结果。

 \* 要求：该字符串只包含小写字母，若其值加5后的字符值大于'z'，将其转换成从a开始的字符

 \*/

public class Demo1 {

public static void main(String[] args) {

Demo1 test = new Demo1();

String data = "";

String regex = "^[a-z]\*$";

if (data.matches(regex)) {

System.out.println(test.getString(data));

} else {

System.out.println("输入的字符串不合法");

}

}

public char[] getString(String data) {

char[] arr = data.toCharArray();

for (int i = 0; i < arr.length; i++) {

if (arr[i] + 5 > 'z') {

arr[i] = (char) (arr[i] - 21);

} else {

arr[i] = (char) (arr[i] + 5);

}

}

return arr;

}

}

/\*\*

 \*2、求一个整形数组中元素的平均值，并统计其中大于和小于此平均值的元素的个数

  要求：输入：整形数组中的元素个数及各个元素

             输出：整型数组中元素的平均值，大于和小于此平均值的元素个数

 \*/

public class Demo2 {

public static void main(String[] args) {

int[] arr ={1,100};

new Demo2().getavage(arr, 2);

}

public void getavage(int[] arr, int num){

if(num==0){

System.out.println("数组为空");

}else{

double avage = 0.0;

int sum = 0;

int count1=0;

int count2=0;

for(int i=0;i<arr.length;i++){

sum += arr[i];

}

avage = (double)sum/num;

for(int j =0;j<num;j++){

if(arr[j]>avage){

count1++;

}else{

count2++;

}

}

System.out.println("平均值:"+avage);

System.out.println("大于平均值的元素个数:"+count1);

System.out.println("小于平均值的元素个数:"+count2);

}

}

}

3、手动输入一个存储整数的数组，要求输出数组里面的2个最大值

  例子：输入：1、2、5、9、84、3、2

      输出：84、9

public class Demo3 {

public static void main(String[] args) {

Demo3 test = new Demo3();

int[] arr = {1};

test.getMaxValue(arr);

}

public void getMaxValue(int[] arr){

int temp=0;

if(arr.length<1){

System.out.println("数组元素个数至少为2");

}

else{

for(int i=0;i<arr.length-1;i++){

for(int j=0;j<arr.length-1-i;j++){

  if(arr[i]>arr[i+1]){

temp = arr[i];

arr[i]=arr[i+1];

arr[i+1]=temp;

  }

    }

}

System.out.print(arr[arr.length-1]+"、"+arr[arr.length-2]);

}

}

}

 4、回文数字判断。题目描述：有这样一类数字，他们顺着看和倒着看是相同的数。

    如：121，656，2332等。这样的数字称为回文数字。

    编写一个函数，判断某数字是否是回文数字。

    public String isPalidrome(String strIn)

     输入：strIn:整数，以字符串表示

     返回：true:是回文数字

         false:不是回文数字

    只需完成该函数功能算法，中间不需要任何IO的输入和输出

    示例：输入：strIn="121",返回：true

public class Demo4 {

public boolean isPalindrome(String str) {

        char[] arr = str.toCharArray();

        int len = arr.length;

        boolean b = false;

        for(int i = 0; i <= len/2; i++ ) {

           if(arr[i] != arr[len-i-1]) {

              break;

           }

           b=true;

        }

        return b;

   }

   public static void main(String[] args) {

        String str = "223322";

        if(new Demo4().isPalindrome(str) ) {

               System.out.println("该字符串是回文字符串！");

        } else {

               System.out.println("该字符串不是回文字符串！");

          }

    }

}

6、手动输入一个字符串，仅限小写字母，统计并输出每个字符在字符串中出现的次数，并输出。提示可以用map

   例子：输入：aaabbbccc

       输出：a 3

                   b 3

                   c 3

public class Demo6 {

public static void main(String[] args) {

Demo6 test = new Demo6();

String data = "kjdfldfsaf";

String regex = "^[a-z]\*$";

if(data.matches(regex)){

Map<Character, Integer> map = test.getCharAount(data);

for(Map.Entry<Character, Integer> entry:map.entrySet()){

System.out.println(entry.getKey()+" "+entry.getValue());

}

}else{

System.out.println("输入的字符串不合法");

}

}

public Map<Character,Integer> getCharAount(String data){

Map<Character,Integer> tree = new TreeMap<Character,Integer>();

char[] arr = data.toCharArray();

for(int i=0;i<arr.length;i++){

if(!tree.containsKey(arr[i])){

tree.put(arr[i], new Integer(1));

}else{

Integer in = tree.get(arr[i])+1;

tree.put(arr[i], in);

}

}

return tree;

}

}

7、要求实现方法（这是在网上看的一个答案。没有处理负数相加。字符串相加竟然也能通过！）

public class Demo7\_bignum {

   public String addTwoBigNumber(String s1,String s2){

   //大数相加，注意处理异常

   String result="";

   //保证s1是最长的

   if(s1.length()<s2.length()){

   String temStr = s2;

   s2 = s1;

   s1 = temStr;

   }

   int sum =0;

   int flag = 0;//进位

   int deleW=s1.length()-s2.length();

   //从个位开始相加

   for(int i=s1.length()-1;i>=0;i--){

   if((i-deleW)>=0){

   sum = (s1.charAt(i)-'0')+(s2.charAt(i-deleW)-'0')+flag;

   }else{

   sum = (s1.charAt(i)-'0')+flag;

   }

   //每次相加后把进位置0

   flag = 0;

   //如果和大于等于10，再将进位置为1

   if(sum>9){

   sum-=10;

   flag = 1;

   }

   result = sum+result;

   }

   //如果最高位有进位

   if(flag == 1){

   result = "1"+result;

   flag = 0;

   }

   return result;

   }

   public static void main(String[] args) {

   Demo7\_bignum test = new Demo7\_bignum();

      String str = test.addTwoBigNumber("","0");

       System.out.println(str);

    }

  }

7.1、大数相加

import java.math.BigDecimal;

public class Demo7\_1 {

   public String addTwoBigNumber(String s1,String s2){

   //大数相加，注意处理异常

   String regex = "^\\-?[1-9][0-9]\*\\.?[0-9]\*$";

   if(!s1.matches(regex)||!s2.matches(regex)){

   return "输入的值不合法";

   }

   BigDecimal bd1 = new BigDecimal(s1);

   BigDecimal bd2 = new BigDecimal(s2);

   String result = bd1.add(bd2).toString();

   return result;

   }

   public static void main(String[] args) {

   Demo7\_1 test = new Demo7\_1();

       String r1 = test.addTwoBigNumber("123456789","987654321");

       String r2 = test.addTwoBigNumber("-10.5", "5");

       System.out.println(r1);

       System.out.println(r2);

  }

}

/\*\*

 \* 8、比较二维数组列最小值，组成一个新数组返回。

 \* (实现核心算法，不需要使用IO) 输入：{{5,6,1,16},{7,3,9}} 输出：intArrs

 \*  ={1,3}

 \*/

public class Demo8 {

public static void main(String args[]) {

// 写测试方法

int[][] a = { { 5, 6, 1, 16 }, { 7, 3, 9 }, { 2, 4, 56 } };

int[] ss = getColMin(a);

for (int i = 0; i < ss.length; i++) {

System.out.print(ss[i] + " ");

}

}

public static int[] getColMin(int a[][]) {

int[] res = new int[a.length];

for (int i = 0; i < a.length; i++) {

int[] s = a[i];

Arrays.sort(s);

res[i] = s[0];

}

return res;

}

}

/\*\*

 \* 10、功能描述：将字符串中的字母全部替换成字母的下一个字母， 要是最后一位是z或Z则替换为a或A。

 \*        输入：aBxyZ

 \*        输出：bCyzA

 \*/

public class Demo10 {

public static void main(String[] args) {

String data = "";

System.out.println(new Demo10().nextString(data));

}

public String nextString(String data) {

char[] arr = data.toCharArray();

for (int i = 0; i < arr.length; i++) {

if (arr[i] >= 'a' && arr[i] <= 'z' || arr[i] >= 'A'

&& arr[i] <= 'Z') {

if (arr[i] == 'z' || arr[i] == 'Z') {

arr[i] = (char) (arr[i] - 25);

} else if ((arr[i] >= 'a' && arr[i] < 'z')

|| (arr[i] >= 'A' && arr[i] < 'Z')) {

arr[i] = (char) (arr[i] + 1);

}

}

}

String res = String.valueOf(arr);

return res;

}

}

/\*\*

 \* 11、判断一个字符串中是否只含有相同的子字符串（子字符串>=2）。 输入：abab 返回：true 输入：abcd 返回：false

 \*/

public class Demo11 {

public static void main(String[] args) {

String data = "ababab";

boolean flag = new Demo11().checkString(data);

System.out.println(flag);

}

public boolean checkString(String data) {

boolean flag = false;

if (data.length() == 2 || data.length() == 0 || data.length() % 2 != 0) {

return flag;

}

int subStrMaxLen = data.length() / 2;

for (int i = 2; i <= subStrMaxLen; i++) {

int subStrNumber = data.length() / i;

String[] temp = new String[subStrNumber];

int begin = 0;

int end = i;

for (int j = 0; j < subStrNumber; j++) {

temp[j] = data.substring(begin, end);

begin += i;

end += i;

}

for (int k = 0; k < subStrNumber - 1; k++) {

if (temp[k].equals(temp[k + 1])) {

flag = true;

} else {

flag = false;

break;

}

}

if (flag == true) {

break;

}

}

return flag;

}

}

1、对整形数据组按照和指定整数的差值大小进行排序，按照差值升序排列返回。

要求实现方法：

public static int[] calcTimes(int[] num, int value);

【输入】 num：整型数组；

         value 指定的整数

【返回】 按照升序返回整型数组，排序按照各个整数和指定整数的差值大小

【注意】只需要完成该函数功能算法，中间不需要有任何IO的输入输出

示例

输入：num = {1,2,3,4,5,10,17,18,19} value = 5

返回：{5,4,3,2,1,10,17,18,19}

#include<iostream>  
#include<cmath>  
using namespace std;  
  
void quickSort(int \*&num,int \*&data,int begin,int end){  
  
 if(begin < end){  
 int i = begin, j = end;  
 int flag = end;  
 int value = data[flag];  
 int value\_num = num[flag];  
 while(i < j){  
 if(flag == j){  
 if(value < data[i]){  
 data[flag] = data[i];  
 num[flag] = num[i];  
  
 flag = i;  
 j--;  
 }else  
 i++;  
 }else if(flag == i){  
 if(value > data[j]){  
 data[flag] = data[j];  
 num[flag] = num[j];  
  
 flag = j;  
 i++;  
 }else  
 j--;  
 }  
 }  
  
 data[flag] = value;  
 num[flag] = value\_num;  
 quickSort(num,data,begin,flag-1);  
 quickSort(num,data,flag+1,end);  
 }  
}  
  
int\* calcTimes(int num[], int n, int value){  
   
 int \*data = new int[n];  
 for(int i=0;i < n;i++){  
 data[i] = abs(num[i]-value);  
 }  
 quickSort(num,data,0,n-1);  
  
  
 cout<<endl;  
 delete[] data;  
 return num;  
}  
  
int main(){  
 int n,m;  
 cout<<"please input two numbers(n and m):"<<endl;  
 cin>>n>>m;  
 cout<<"please n numbers:"<<endl;  
 int \*num = new int[n];  
 for(int i=0;i<n;i++){  
 cin>>num[i];  
 }  
 int \*result = calcTimes(num,n,m);  
 for(int i=0;i<n;i++){  
 cout<<result[i]<<" ";  
 }  
 delete[] num;  
 system("pause");  
 return 0;  
}

题目描述(40分)：

2取出整型数据中出现次数最多的元素，并按照升序排列返回。

要求实现方法：

public static int[] calcTimes(int[] num, int len);

【输入】 num：整型数组；

         len 输入的整数个数

【返回】 按照升序排列返回整型数组中出现次数最多的元素

【注意】只需要完成该函数功能算法，中间不需要有任何IO的输入输出

示例

输入：num = {1,1,3,4,4,4,9,9,9,10} len = 10

返回：{4,9}

#include<iostream>  
#include<algorithm>  
#include<vector>  
#include<map>  
#include<climits>  
using namespace std;  
  
//方法一  
vector<int> calcTimes(int num[], int len){  
 vector<int> result;  
 map<int,int> sum;  
  
 map<int,int>::iterator it;  
 //get the num's count  
 for(int i=0;i<len;i++){  
 it = sum.find(num[i]);  
 if(it != sum.end()){  
 int t = it->second;  
 sum.erase(it);  
 sum.insert(pair<int,int>(num[i],t+1));  
 }  
 else  
 sum.insert(pair<int,int>(num[i],1));  
 }  
 //get the element  
 it = sum.begin();  
   
 int max = INT\_MIN;  
 for(;it != sum.end();it++){  
 if(it->second > max){  
 max = it->second;  
 result.clear();  
 result.push\_back(it->first);  
 }else if(it->second == max){  
 result.push\_back(it->first);  
 }  
 }  
 //sort  
 sort(result.begin(),result.end());  
 return result;  
}  
//方法二  
int compare (const void \* a, const void \* b)  
{  
 return ( \*(int\*)a - \*(int\*)b );  
}  
int \*calcTimes2(int num[], int len){  
  
 qsort(num,len,sizeof(int),compare);  
  
 vector<int> result;  
 int max = INT\_MIN;  
 int count=1,flag=num[0];  
 for(int i=1;i<len;i++){  
 if(flag == num[i]){  
 ++count;  
 continue;  
 }else{  
   
 if(count > max){  
 result.clear();  
 result.push\_back(flag);  
 max = count;  
 }else if(count == max){  
 result.push\_back(flag);  
 }  
 flag = num[i];  
 count = 1;  
 }  
 }  
 //最后一个数的次数  
 if(count > max){  
 result.clear();  
 result.push\_back(flag);  
 }else if(count == max){  
 result.push\_back(flag);  
 }  
 int size = result.size();  
 int \*data = new int[size];  
   
 for(int i=0;i<size;i++){  
 data[i] = result[i];  
 cout<<data[i]<<" ";  
 }  
   
 cout<<endl;  
 return data;  
}  
  
int main(){  
  
 int len;  
 cin>>len;  
 int \*data = new int[len];  
 for(int i=0;i<len;i++){  
 cin>>data[i];  
 }  
 vector<int> r = calcTimes(data, len);  
 int \*result = calcTimes2(data,len);  
 delete[] data;  
 system("pause");  
 return 0;  
}

3题目描述(附加题 50分)：

取出整型数据中存在指定整数的数据，并按照升序排列返回。

要求实现方法：

public static int[] calcTimes(int[] num, int value);

【输入】 num：整型数组；

         value 指定的整数

【返回】 按照升序排列返回整型数组中包含指定整数的元素

【注意】只需要完成该函数功能算法，中间不需要有任何IO的输入输出

示例

输入：num = {12345,654123,98764,123654,9876,4,6,97,8,123} value = 123

返回：{123,12345,123654,654123}

#include<iostream>  
#include<algorithm>  
#include<vector>  
#include<map>  
#include<climits>  
using namespace std;  
  
//方法一  
int compare (const void \* a, const void \* b)  
{  
 return ( \*(int\*)a - \*(int\*)b );  
}  
int \*calcTimes(int num[], int len,int value){  
 vector<int> data;  
 qsort(num,len,sizeof(int),compare);  
  
 int n=1,tmp=value;  
 while(tmp!=0){  
 n\*=10;  
 tmp/=10;  
 }  
 int reminder=0;  
 for(int i=0;i<len;i++){  
 if(num[i] < n && num[i] != value)  
 continue;  
 else if(num[i] == value){  
 data.push\_back(num[i]);  
 }else{  
 tmp = num[i];  
 while(tmp >= value){  
 reminder = tmp%n;  
 if(reminder == value){  
 data.push\_back(num[i]);  
 break;  
 }  
 else{  
 tmp/=10;  
 }  
  
 }  
 }  
 }  
 int \*result = new int[data.size()];  
 for(int i=0;i<data.size();i++){  
 result[i] = data[i];  
 cout<<result[i]<<" ";  
 }  
 cout<<endl;  
  
 return result;  
}  
  
int main(){  
  
 int len,value;  
 cin>>len>>value;  
 int \*data = new int[len];  
 for(int i=0;i<len;i++){  
 cin>>data[i];  
 }  
  
 int \*result = calcTimes(data,len,value);  
  
 delete[] data;  
 system("pause");  
 return 0;  
}

---------------------------------------------------------------4------------------------------------------------------------------------------

4问题描述：

将输入的字符串（字符串仅包含小写字母‘a’到‘z’），按照如下规则，循环转换后输出：a->b,b->c,…,y->z,z->a；若输入的字符串连续出现两个字母相同时，后一个字母需要连续转换2次。例如：aa 转换为 bc，zz 转换为 ab；当连续相同字母超过两个时，第三个出现的字母按第一次出现算。

要求实现函数：

void convert(char \*input,char\* output)

【输入】  char \*input , 输入的字符串

【输出】  char \*output ，输出的字符串

【返回】 无

void convert(char \*input,char\*output){  
 if(input == NULL){  
 output[0]='\0';  
 return;  
 }  
  
 char \*p=output;  
 char flag = '0';  
 int count=1;  
 for(int i = 0;input[i] != '\0';i++){  
 //和前面一个不一样  
 if(input[i] != flag){  
 \*p++ = input[i]+1;  
 flag = input[i];  
 count=1;  
 }else{  
 count++;  
 //和前面一个字符一样，重复count次  
 if(count % 2 == 0){  
 \*p = \*(p-1)+1;  
 p++;  
 }else{  
 \*p++ = input[i]+1;  
 }  
 }  
 }  
 \*p='\0';  
  
}

------------------------------------------------------------5-----------------------------------------------------------------------

问题描述：      
在给定字符串中找出单词（ “单词”由大写字母和小写字母字符构成，其他非字母字符视为单词的间隔，如空格、问号、数字等等；另外单个字母不算单词）；找到单词后，按照长度进行降序排序，（排序时如果长度相同，则按出现的顺序进行排列），然后输出到一个新的字符串中；如果某个单词重复出现多次，则只输出一次；如果整个输入的字符串中没有找到单词，请输出空串。输出的单词之间使用一个“空格”隔开，最后一个单词后不加空格。  
要求实现函数：  
void my\_word(charinput[], char output[])  
【输入】  char input[], 输入的字符串  
【输出】  char output[]，输出的字符串  
【返回】 无  
示例  
输入：charinput[]="some local buses, some1234123drivers" ，  
输出：charoutput[]="drivers local buses some"  
输入：charinput[]="%A^123 t 3453i\*()" ，  
输出：charoutput[]=""

#include<iostream>  
#include<string>  
#include<vector>  
#include<algorithm>  
using namespace std;

bool myfunction (const string &str1,const string &str2) {   
   
 return (str2.length() < str1.length());  
}  
  
void my\_word(char input[], char output[]){  
   
 vector<string> str;  
 char \*cur = input;  
 char \*p = input;  
 int len;  
 bool flag = false;  
 //去除杂质，取单词  
 while(\*p != '\0'){  
 if((\*p >= 'a' && \*p <= 'z') || (\*p >= 'A' && \*p <= 'Z'))  
 p++;  
 else{  
 len = p-cur;  
 if(len > 1){  
 string tmp(cur,len);  
  
 for(int i=0;i<str.size();i++)  
 if(tmp == str[i])  
 flag = true;  
 if(flag != true)  
 str.push\_back(tmp);  
 }  
 cur = p+1;  
 p++;  
 flag = false;  
 }  
 }  
  
 len = p-cur;  
 if(len > 1){  
 string tmp(cur,len);  
  
 for(int i=0;i<str.size();i++)  
 if(tmp == str[i])  
 flag = true;  
 if(flag != true)  
 str.push\_back(tmp);  
 }  
  
 //sort  
 sort(str.begin(),str.end(),myfunction);  
 //拼接字符串  
 \*output='\0';  
 int i;  
 for(i=0;i<str.size()-1;i++){  
 strcpy(output+strlen(output),(str[i]+" ").c\_str());  
   
 }  
 strcpy(output+strlen(output),(str[i]).c\_str());  
 cout<<output<<endl;  
}

int main(){  
  
 char input[]="some ninja local buses, somes1234123drivers";  
 char output[500];  
 my\_word(input, output);  
 system("pause");  
 return 0;  
}

-----------------------------------------------------------------6----------------------------------------------------------

问题描述：      
两个任意长度的正数相减，这两个正数可以带小数点，也可以是整数，请输出结果。 输入的字符串中，不会出现除了数字与小数点以外的其它字符，不会出现多个小数点以及小数点在第一个字符的位置等非法情况，所以考生的程序中无须考虑输入的数值字符串非法的情况。   
详细要求以及约束：  
1.输入均为正数，但输出可能为负数；   
2.输入输出均为字符串形式；  
3.如果输出是正数则不需要带符号，如果为负数，则输出的结果字符串需要带负号  
例如：2.2-1.1 直接输出为“1.1”，1.1-2.2 则需要输出为“-1.1”  
 4.输出的结果字符串需要过滤掉整数位前以及小数位后无效的0，小数位为全0的，直接输出整数位  
例如相减结果为11.345，此数值前后均不可以带0，“011.345”或者“0011.34500”等等前后带无效0的均视为错误 输出。例如1.1-1.1结果为0.0，则直接输出0。  
要求实现函数：  
void Decrease(char \*input1, char\*input2, char \*output)  
【输入】 char \*iinput1 被减数  
char\*nput2 减数   
【输出】 char \*output 减法结果  
【返回】 无  
示例  
输入：char \*input1="2.2"   
char \*input2="1.1"  
输出：char\*output="1.1"  
输入：char \*input1="1.1"   
char \*input2="2.2"  
输出：char \*output="-1.1"

------------------------------------------------------------2013-------------------------------------------------------------

/\* 请在这里实现下列函数, c c++语法不限, 最后需要保证程序编译连接通过, 并生成test.exe文件. \*/

/\* 相关宏定义及函数声明见'func.h'头文件

\*/#include "stdafx.h"

#include <stdio.h>

#include <memory.h>

#include <string.h>

#include <stdlib.h>

#include "func.h"

#define MAXCHAR 100/\* 请按照要求实现下列函数，pOutputStr的长度足够大\*/#include "string.h"#include "stdlib.h"

/\*

题目描述（60分）：通过键盘输入一串小写字母(a~z)组成的字符串。请编写一个字符串过滤程序，若字符串中出现多个相同的字符，将非首次出现的字符

过滤掉。比如字符串“abacacde”过滤结果为“abcde”。要求实现函数： void stringFilter(const char \*pInputStr, long lInputLen, char

\*pOutputStr);

【输入】 pInputStr：  输入字符串

         lInputLen：  输入字符串长度

【输出】 pOutputStr： 输出字符串，空间已经开辟好，与输入字符串等长；

【注意】只需要完成该函数功能算法，中间不需要有任何IO的输入输出

示例

输入：“deefd”        输出：“def”

输入：“afafafaf”     输出：“af”

输入：“pppppppp”     输出：“p”

\*/20:29 2013/7/24

/\* main函数已经隐藏，这里保留给用户的测试入口，在这里测试你的实现函数，可以调用printf打印输出\*/

/\* 当前你可以使用其他方法测试，只要保证最终程序能正确执行即可 \*/

/\* 该函数实现可以任意修改，但是不要改变函数原型。一定要保证编译运行不受影响\*/

void stringFilter(const char \*pInputStr, long lInputLen, char \*pOutputStr)

{

}

【答案】

void stringFilter(const char \*pInputStr, long lInputLen, char \*pOutputStr)

{

char \*cur = NULL;

char \*r = pOutputStr;

\*pOutputStr='\0';

while(\*pInputStr != '\0'){

cur = pOutputStr;

while(\*cur != '\0'){

if(\*cur == \*pInputStr)

break;

cur++;

}

if(\*cur == '\0')

\*(r++) = \*(pInputStr++);

else

pInputStr++;

}

\*r='\0';

}

/\*

题目描述（40分）：

通过键盘输入一串小写字母(a~z)组成的字符串。请编写一个字符串压缩程序，将字符串中连续出席的重复字母进行压缩，并输出压缩后的字符串。

压缩规则：

1. 仅压缩连续重复出现的字符。比如字符串"abcbc"由于无连续重复字符，压缩后的字符串还是"abcbc".

2. 压缩字段的格式为"字符重复的次数+字符"。例如：字符串"xxxyyyyyyz"压缩后就成为"3x6yz"

要求实现函数：

void stringZip(const char \*pInputStr, long lInputLen, char \*pOutputStr);

【输入】 pInputStr：  输入字符串

         lInputLen：  输入字符串长度

【输出】 pOutputStr： 输出字符串，空间已经开辟好，与输入字符串等长；

【注意】只需要完成该函数功能算法，中间不需要有任何IO的输入输出

示例

输入：“cccddecc”   输出：“3c2de2c”

输入：“adef”     输出：“adef”

输入：“pppppppp” 输出：“8p”

\*/

void stringZip(const char \*pInputStr, long lInputLen, char \*pOutputStr)

{

int count = 0;

char curChar = \*pInputStr;

\*pOutputStr='\0';

//遍历

while(\*pInputStr != '\0'){

if(\*pInputStr == curChar)

count++;

else{

if(count == 1){

sprintf(pOutputStr+strlen(pOutputStr),"%c",curChar);

}else{

sprintf(pOutputStr+strlen(pOutputStr),"%d%c",count,curChar);

}

curChar = \*pInputStr;

count = 1;

}

pInputStr++;

}

//最后一组

if(count == 1){

sprintf(pOutputStr+strlen(pOutputStr),"%c",curChar);

}else{

//\*p++ = '0'+count;

sprintf(pOutputStr+strlen(pOutputStr),"%d%c",count,curChar);

}

\*(pOutputStr+strlen(pOutputStr))='\0';

}

/\*

题目描述（50分）：

通过键盘输入100以内正整数的加、减运算式，请编写一个程序输出运算结果字符串。

输入字符串的格式为：“操作数1 运算符 操作数2”，“操作数”与“运算符”之间以一个空格隔开。

补充说明：

1. 操作数为正整数，不需要考虑计算结果溢出的情况。

2. 若输入算式格式错误，输出结果为“0”。

要求实现函数：

void arithmetic(const char \*pInputStr, long lInputLen, char \*pOutputStr);

【输入】 pInputStr：  输入字符串

         lInputLen：  输入字符串长度

【输出】 pOutputStr： 输出字符串，空间已经开辟好，与输入字符串等长；

【注意】只需要完成该函数功能算法，中间不需要有任何IO的输入输出

示例

输入：“4 + 7”  输出：“11”

输入：“4 - 7”  输出：“-3”

输入：“9 ++ 7”  输出：“0” 注：格式错误

\*/

void arithmetic(const char \*pInputStr, long lInputLen, char \*pOutputStr)

{

char \*pch = NULL;

char \*cur = new char[lInputLen+1];

\*pOutputStr = '\0';

strcpy(cur,pInputStr);

pch = strtok(cur," ");

int a=0,b=0,count=0;

char \*flag = NULL;

while (pch != NULL)

{

if(count == 0)

a = atoi (pch);

else if(count == 2)

b = atoi (pch);

else

flag = pch;

pch = strtok (NULL, " ");

count++;

}

if(strcmp(flag,"+") == 0){

a = a+b;

sprintf(pOutputStr+strlen(pOutputStr),"%d",a);

}else if(strcmp(flag,"-") == 0){

a = a-b;

sprintf(pOutputStr+strlen(pOutputStr),"%d",a);

}else{

\*pOutputStr++ = '0';

\*pOutputStr = '\0';

}

delete[] cur;

}

void main\_i()

{

/\* TODO: 请测试时改变改用例 \*/

char pInputStr1[] = {"aaabbbcccdde"};

char pInputStr2[] = {"aaabbcddde"};

char pInputStr3[] = {"2 + 5"};

char pOutputStr1[MAXCHAR] = {0};

char pOutputStr2[MAXCHAR] = {0};

char pOutputStr3[MAXCHAR] = {0};

/\* TODO: 调用被测函数 \*/

stringFilter(pInputStr1, strlen(pInputStr1), pOutputStr1);

stringZip(pInputStr2, strlen(pInputStr2), pOutputStr2);

arithmetic(pInputStr3, strlen(pInputStr3), pOutputStr3);

/\* TODO: 执行完成后可比较是否是你认为正确的值 \*/

printf(pOutputStr1); //abcde

printf(pOutputStr2); //3a3b3c2de

printf(pOutputStr3); //7

return;

}

//判断回文数  
bool huiwen(const int &num){  
 int tmp=0;   
 int num2=num;  
 while(num2){  
   
 tmp = tmp\*10+num2%10;  
 num2 = num2/10;  
 }  
 if(num == tmp)  
 return true;  
 else  
 return false;  
}  
//判断回文字符串  
bool huiwen(const char \*str){  
  
 if(str == NULL)  
 return false;  
  
 int len = strlen(str);  
 cout<<len<<endl;  
 for(int i=0;i<len/2;i++){  
 if(\*(str+i) != \*(str+len-1-i))  
 return false;  
 }  
 return true;  
}  
//字符串逆序  
void reverse(char \*str){  
 if(str == NULL)  
 return;  
 int len = strlen(str);  
 for(int i=0;i<len/2;i++){  
 char tmp = \*(str+i);  
 \*(str+i) = \*(str+len-1-i);  
 \*(str+len-1-i) = tmp;   
 }  
}  
//素数  
int sushu(int num){  
 if(num == 1)  
 return 0;  
 if(num == 2)  
 return 1;  
 for(int i=2;i<=sqrt((double)num);i++){  
 if(num % i == 0)  
 return 0;  
 }  
 return 1;  
}  
//大于等于平均值的元素个数cnt  
int func(int iArray[],int nSize){  
 float avg = 0.0;  
 for(int i=0;i<nSize;i++){  
 avg += iArray[i];  
 }  
 avg = avg/nSize;  
 cout<<avg<<endl;  
 int count=0;  
 for(int i=0;i<nSize;i++){  
 if(avg <= iArray[i])  
 ++count;  
 }  
 return count;  
}  
  
//寻找num之内的所有素数  
bool isPrim[1000001];  
int primNumber[800000];  
void getPrim(int iNumber)   
{  
 int i = 0, j = 0,cnt = 0;  
 for(; i <= iNumber; ++i)  
 isPrim[i] = true;  
 for(i = 2; i <= sqrt((double)iNumber); ++ i)  
 {  
 for(j = i + i; j <= iNumber; j += i)  
 {  
 if(isPrim[i])  
 isPrim[j] = false;  
 }  
 }  
 for(i = 2; i <= iNumber; ++ i)  
 {  
 if(isPrim[i])  
 {  
 primNumber[cnt ++] = i;  
 }  
 }  
}

//最长回文字串  
int LongestPalindrome(const char \*s, int n){  
 int i,j,max;  
 if(s == NULL || n < 1)  
 return 0;  
 max = 0;  
 for(i = 0;i < n; ++i){  
 //odd  
 for(j = 0;(i-j >= 0) && (i+j < n); ++j)  
 if(s[i-j] != s[i+j])  
 break;  
   
 if(j\*2+1 > max)  
 max = j\*2+1;  
 //even   
 for(j = 0;(i-j >= 0) && (i+j < n); ++j)  
 if(s[i-j] != s[i+j+1])  
 break;  
   
 if(j\*2+2 > max)  
 max = j\*2+2;  
 }  
}

//删除要查找的字符串，并返回删除次数  
int delete\_sub\_str(char \*input,char \*sub\_str,char \*result){  
 int count=0;  
 int len = strlen(sub\_str);  
 char \*p = result;  
  
 while(\*input != '\0'){  
 int i=0;  
 for(;i<len;i++){  
 \*p = \*input;  
 if(\*input != \*(sub\_str+i)){  
 p++;  
 input++;  
 break;;  
 }  
 p++;  
 input++;  
 }  
  
 if(i == len){  
 p = p-len;  
 count++;  
 }  
  
 }  
 \*p='\0';  
 return count;  
}

笔试题目（机试，共两题）   
题目一：子串分离   
题目描述：     
通过键盘输入任意一个字符串序列，字符串可能包含多个子串，子串以空格分隔。请编写一个程序，自动分离出各个子串，并使用’,’将其分隔，并且在最后也补充一个’,’并将子串存储。   
如果输入“abc def gh i        d”，结果将是abc,def,gh,i,d,   
   
要求实现函数：     
void DivideString(const char \*pInputStr, long lInputLen, char \*pOutputStr);   
   
【输入】  pInputStr：  输入字符串   
          lInputLen：  输入字符串长度                     
【输出】  pOutputStr：  输出字符串，空间已经开辟好，与输入字符串等长；   
【注意】只需要完成该函数功能算法，中间不需要有任何IO 的输入输出   
示例     
输入：“abc def gh i        d”  
输出：“abc,def,gh,i,d,”  
转载请标明出处，原文地址：<http://blog.csdn.net/hackbuteer1/article/details/11150519>

**[cpp]** [view plaincopy](http://blog.csdn.net/hackbuteer1/article/details/11150519)

1. **void** DivideString(**const** **char**\* pInputStr , **long** lInputLen , **char**\* pOutputStr)
2. {
3. **int** i , j;
4. **bool** flag;
5. **for**(i = 0 ; pInputStr[i] == ' ' ; ++i)   //跳过字符串前面的空格
6. ;
7. flag = **true**;
8. **for**(j = 0 ; i < lInputLen ; ++i)
9. {
10. **if**(pInputStr[i] != ' ')
11. {
12. **if**(!flag)
13. flag = **true**;
14. pOutputStr[j++] = pInputStr[i];    //将各个子串分离保存下来
15. }
16. **else**
17. {
18. **if**(flag)
19. pOutputStr[j++] = ',';
20. flag = **false**;
21. }
22. }
23. pOutputStr[j++] = ',';
24. pOutputStr[j++] = '\0';
25. }

题目二：逆序链表输出。   
题目描述：     
将输入的一个单向链表，逆序后输出链表中的值。链表定义如下：   
typedef struct tagListNode   
{   
      int value;   
      struct tagListNode \*next;   
}ListNode;   
   
要求实现函数：     
void converse(ListNode \*\*head);   
【输入】head：    链表头节点，空间已经开辟好   
【输出】head：    逆序后的链表头节点  
【返回】无   
【注意】只需要完成该函数功能算法，中间不需要有任何IO 的输入输出

**[cpp]** [view plaincopy](http://blog.csdn.net/hackbuteer1/article/details/11150519)

1. // 返回新链表头节点
2. LinkNode \*reverse\_link(LinkNode \*head)
3. {
4. **if**(head == NULL)
5. **return** NULL;
6. LinkNode \*prev , \*curr , \*reverse\_head , \*temp;
7. prev = NULL , curr = head;
8. **while**(curr->next)
9. {
10. temp = curr->next;
11. curr->next = prev;
12. prev = curr;
13. curr = temp;
14. }
15. curr->next = prev;
16. reverse\_head = curr;
17. **return** reverse\_head;
18. }

1 /\*

2

3 题目描述（60分）：

4 通过键盘输入一串小写字母(a~z)组成的字符串。请编写一个字符串过滤程序，若字符串中出现多个相同的字符，将非首次出现的字符过滤掉。

5 比如字符串“abacacde”过滤结果为“abcde”。

6

7 要求实现函数：

8 void stringFilter(const char \*pInputStr, long lInputLen, char \*pOutputStr);

9

10 【输入】 pInputStr： 输入字符串

11 lInputLen： 输入字符串长度

12 【输出】 pOutputStr： 输出字符串，空间已经开辟好，与输入字符串等长；

13

14 【注意】只需要完成该函数功能算法，中间不需要有任何IO的输入输出

15

16 示例

17 输入：“deefd” 输出：“def”

18 输入：“afafafaf” 输出：“af”

19 输入：“pppppppp” 输出：“p”

20 \*/

21

22 /\* main函数已经隐藏，这里保留给用户的测试入口，在这里测试你的实现函数，可以调用printf打印输出\*/

23 /\* 当前你可以使用其他方法测试，只要保证最终程序能正确执行即可 \*/

24 /\* 该函数实现可以任意修改，但是不要改变函数原型。一定要保证编译运行不受影响\*/

25

26 /\*

27 题目描述（40分）：

28 通过键盘输入一串小写字母(a~z)组成的字符串。请编写一个字符串压缩程序，将字符串中连续出席的重复字母进行压缩，并输出压缩后的字符串。

29 压缩规则：

30 1. 仅压缩连续重复出现的字符。比如字符串"abcbc"由于无连续重复字符，压缩后的字符串还是"abcbc".

31 2. 压缩字段的格式为"字符重复的次数+字符"。例如：字符串"xxxyyyyyyz"压缩后就成为"3x6yz"

32

33 要求实现函数：

34 void stringZip(const char \*pInputStr, long lInputLen, char \*pOutputStr);

35

36 【输入】 pInputStr： 输入字符串

37 lInputLen： 输入字符串长度

38 【输出】 pOutputStr： 输出字符串，空间已经开辟好，与输入字符串等长；

39

40 【注意】只需要完成该函数功能算法，中间不需要有任何IO的输入输出

41

42 示例

43 输入：“cccddecc” 输出：“3c2de2c”

44 输入：“adef” 输出：“adef”

45 输入：“pppppppp” 输出：“8p”

46 \*/

47

48 /\*

49 题目描述（50分）：

50 通过键盘输入100以内正整数的加、减运算式，请编写一个程序输出运算结果字符串。

51 输入字符串的格式为：“操作数1 运算符 操作数2”，“操作数”与“运算符”之间以一个空格隔开。

52

53 补充说明：

54 1. 操作数为正整数，不需要考虑计算结果溢出的情况。

55 2. 若输入算式格式错误，输出结果为“0”。

56

57 要求实现函数：

58 void arithmetic(const char \*pInputStr, long lInputLen, char \*pOutputStr);

59

60 【输入】 pInputStr： 输入字符串

61 lInputLen： 输入字符串长度

62 【输出】 pOutputStr： 输出字符串，空间已经开辟好，与输入字符串等长；

63

64 【注意】只需要完成该函数功能算法，中间不需要有任何IO的输入输出

65

66 示例

67 输入：“4 + 7” 输出：“11”

68 输入：“4 - 7” 输出：“-3”

69 输入：“9 ++ 7” 输出：“0” 注：格式错误

70 \*/



解答：

|  |  |
| --- | --- |
| 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  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136 | #include <stdio.h>  #include <stdlib.h>  #include <string.h>    #define MAXCHAR 256    //必须规定均为小写字母  void stringFilter(const char \*pInputStr, long lInputLen, char \*pOutputStr)  {      int a[26] = {0};      int pos = 0;      long i, j;      const char \*p = pInputStr;      for(i = 0, j = 0; i < lInputLen; i++)      {          pos = pInputStr[i] - 'a'; //将出现的字符标记为1，下次出现同样的字符时就不再存储          if(a[pos] == 0)          {              a[pos]++;              pOutputStr[j++] = pInputStr[i];          }      }        pOutputStr[j] = '\0';  }    void stringZip(const char \*pInputStr, long lInputLen, char \*pOutputStr)  {      int i, j, k = 0, cnt = 0;      char buffer[30];        for(i = 0; i < lInputLen;)      {          for(j = i + 1;;j++)          {              if(pInputStr[j] == pInputStr[i])                  cnt++;              else                  break;          }            if(cnt != 0)          {              memset(buffer, 0, sizeof(buffer));              itoa(cnt + 1, buffer, 10);              strcat(pOutputStr, buffer);              k += strlen(buffer);              i += cnt;          }          pOutputStr[k++] = pInputStr[i++];          cnt = 0;      }  }    void arithmetic(const char \*pInputStr, long lInputLen, char \*pOutputStr)  {      int i, cnt = 0, a, b, result;      char ch[1] = {'0'};      char op1[MAXCHAR], op[MAXCHAR], op2[MAXCHAR], buffer[4];      for(i = 0; i < lInputLen; i++)          if(pInputStr[i] == ' ')              cnt++;        if(cnt != 2)                    //空格数不等于2      {          strcat(pOutputStr, ch);          return;      }        sscanf(pInputStr, "%s %s %s", op1, op, op2);        if(strlen(op) > 1 || (op[0] != '+' && op[0] != '-'))        // 操作符有多个      {          strcat(pOutputStr, ch);          return;      }        for(i = 0; i < strlen(op1); i++)                            //操作数1是否有其他字符      {          if(op1[i] < '0' || op1[i] > '9')          {              strcat(pOutputStr, ch);              return;          }      }          for(i = 0; i < strlen(op2); i++)                            //操作数2是否有其他字符      {          if(op2[i] < '0' || op2[i] > '9')          {              strcat(pOutputStr, ch);              return;          }      }        a = atoi(op1);      b = atoi(op2);        switch(op[0])      {          case '+':              result = a + b;              itoa(result, buffer, 10);              strcat(pOutputStr, buffer);              break;          case '-':              result = a - b;              itoa(result, buffer, 10);              strcat(pOutputStr, buffer);              break;          default:              break;      }  }    int main()  {      char pInputStr1[] = {"aaabbbcccdde"};      char pInputStr2[] = {"aaabbcddde"};      char pInputStr3[] = {"3 + 4"};      char pOutputStr1[MAXCHAR] = {0};      char pOutputStr2[MAXCHAR] = {0};      char pOutputStr3[MAXCHAR] = {0};        /\* TODO: 调用被测函数 \*/      stringFilter(pInputStr1, strlen(pInputStr1), pOutputStr1);      stringZip(pInputStr2, strlen(pInputStr2), pOutputStr2);      arithmetic(pInputStr3, strlen(pInputStr3), pOutputStr3);        /\* TODO: 执行完成后可比较是否是你认为正确的值 \*/      printf(pOutputStr1); //abcde      printf(pOutputStr2); //3a3b3c2de      printf(pOutputStr3); //7      return;  } |