目录

[for与switch之间的配合： 3](#_Toc500270521)

[malloc（）动态数组 4](#_Toc500270522)

[Fn=Fn-1+Fn-2递加 5](#_Toc500270523)

[桶排序 6](#_Toc500270524)

[.杨辉三角 7](#_Toc500270525)

[十进制转十六进制 9](#_Toc500270526)

[十六进制转十进制 10](#_Toc500270527)

[检测是否存在相同的字符串 11](#_Toc500270528)

[Mallc与二维数组 11](#_Toc500270529)

[初始化数组 11](#_Toc500270530)

[题目1(空格断开) 12](#_Toc500270531)

[题目2（搬砖） 13](#_Toc500270532)

[题目3（最高分） 14](#_Toc500270533)

[题目4（做好事） 15](#_Toc500270534)

[题目5（保留小数位） 16](#_Toc500270535)

[题目6（找数目） 17](#_Toc500270536)

[取模运算法则 18](#_Toc500270537)

[零碎 18](#_Toc500270538)

[走楼梯 20](#_Toc500270539)

[整数拆分 21](#_Toc500270540)

[有规律 23](#_Toc500270541)

[Cout控制小数点输出 25](#_Toc500270542)

[题目7 01序列 27](#_Toc500270543)

[题目8 字母图形 29](#_Toc500270544)

[十进制转换为其他进制 31](#_Toc500270545)

[排序 32](#_Toc500270546)

[1.冒泡 32](#_Toc500270547)

[2.选择 33](#_Toc500270548)

[3.插入 34](#_Toc500270549)

[任意进制转任意进制 35](#_Toc500270550)

[题目9 中值排序 37](#_Toc500270551)

[题目十 第几个素数 38](#_Toc500270552)

[快速幂取模 40](#_Toc500270553)

## for与switch之间的配合：

#include<stdio.h>

int main()

{

int i,date=12,flag=0;

for(i=1;i<13;i++)

{

switch(i-1)

{

case 1:case 3: case 5: case 7: case 8:case 10:case12:

date+=31;break;

case 4:case 6 :case 9:case 11:

date+=30;break;

case 2:date+=29;break;

case 0:break;

}

if (date%7==5)

{

printf("2012%02d13",i);

}

}

return 0;

}

## malloc（）动态数组

#include<stdio.h>

#include<stdlib.h>

**int** main()

{

**while**(1)

{

**double** e;

**int** i,a=1;

**int** T=0;

**int** \*ch=(**int**\*)malloc(**sizeof**(**int**)\*(T\*4));

scanf("%d",&T);

**for** (i=0;(i+3)<(T\*4);i+=4)

{

scanf("%d/%d",&ch[i],&ch[i+1]);

scanf("%d/%d",&ch[i+2],&ch[i+3]);

}

i=0;

**while** (a<=T)

{

e=((**float**)ch[i]/ch[i+1])-((**float**)ch[i+2]/ch[i+3]);

**if** (e>0)

printf("1**\n**");

**else** **if** (e==0)

printf("0**\n**");

**else**

printf("-1**\n**");

a++;

i=i+4;

}

**break**;

}

**return** 0;

}

## Fn=Fn-1+Fn-2递加

**#include<stdio.h>**

**int Fibo(int n)**

**{**

**if(n==1||n==2)**

**return 1;**

**else**

**return Fibo(n-1)+Fibo(n-2);**

**}**

**int main()**

**{**

**int n=0;**

**scanf("%d",&n);**

**printf("%d\n",Fibo(n));**

## 桶排序

现在你可以请尝试一下输入n个0~1000之间的整数，将他们从大到小排序。提醒一下如果需要对数据范围在0~1000之间的整数进行排序，我们需要1001个桶，来表示0~1000之间每一个数出现的次数，这一点一定要注意。另外此处的每一个桶的作用其实就是“标记”每个数出现的次数，因此我喜欢将之前的数组a换个更贴切的名字book（book这个单词有记录、标记的意思），代码实现如下。

|  |  |
| --- | --- |
|  | #include <stdio.h>  int main()  {      int book[1001],i,j,t,n;      for(i=0;i<=1000;i++)          book[i]=0;      scanf("%d",&n);//输入一个数n，表示接下来有n个数      for(i=1;i<=n;i++)//循环读入n个数，并进行桶排序      {          scanf("%d",&t);  //把每一个数读到变量t中          book[t]++;  //进行计数，对编号为t的桶放一个小旗子      }      for(i=1000;i>=0;i--)  //依次判断编号1000~0的桶          for(j=1;j<=book[i];j++)  //出现了几次就将桶的编号打印几次               printf("%d ",i);      getchar();getchar();      return 0;  } |

## .杨辉三角

#include<stdio.h>

int main()

{

int T[100][100] = { 0 }, n, i, j;

scanf("%d", &n);

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

T[j][0] = 1;

for (i = 1;i<n;i++)

for (j = 1;j<n;j++)

1. T[i][j] = T[i - 1][j] + T[i - 1][j - 1];

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

{

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

{

if (T[i][j])

printf("%d ", T[i][j]);

}

printf("\n");

}

return 0;

}

第n行的数字有n项。

第n行数字和为2n-1。

第n行的m个数可表示为 C(n-1，m-1)，即为从n-1个不同元素中取m-1个元素的组合数。

第n行的第m个数和第n-m+1个数相等 ，为组合数性质之一。

每个数字等于上一行的左右两个数字之和。可用此性质写出整个杨辉三角。即第n+1行的第i个数等于第n行的第i-1个数和第i个数之和，这也是组合数的性质之一。即 C(n+1,i)=C(n,i)+C(n,i-1)。

(a+b)n的展开式中的各项系数依次对应杨辉三角的第(n+1)行中的每一项。

前n行和2^n-1

N行中最大的数

n为奇数时，C(n-1，(n-1)/2)  
n为偶数时，C(n-1，n/2)

#include <stdio.h>

#include <stdlib.h>

int main()

{

int s = 1, h; // 数值和高度

int i, j; // 循环计数

scanf("%d", &h); // 输入层数

printf("1\n"); // 输出第一个 1

for (i = 2; i <= h; s = 1, i++) // 行数 i 从 2 到层高

{

printf("1 "); // 第一个 1

for (j = 1; j <= i - 2; j++) // 列位置 j 绕过第一个直接开始循环

//printf("%d ", (s = (i - j) / j \* s));

printf("%d ", (s = (i - j) \* s / j));

printf("1\n"); // 最后一个 1，换行

}

getchar(); // 暂停等待

return 0;

}

## 十进制转十六进制

#include<stdio.h>

int main()

{

int a;

scanf("%d", &a);

if (0 <= a&&a <= 2147483647)

printf("%X", a);

return 0;

}

#include<stdio.h>

int main()

{

int n,i=0,b,a;

char hex[16] = { '0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F' };

int ch[10000];

scanf("%d", &n);

if (n == 0)

printf("%d", n);

while (n > 0) {

b = n % 16;

ch[i++] = b;

n = n / 16;

}

for (i = i - 1;i >= 0;i--) {

a = ch[i];

printf("%c", hex[a]);}

return 0;

}

## 十六进制转十进制

#include<stdio.h>

#include<string.h>

int main() {

char s[50];

scanf("%s", &s);

int t;

int len = strlen(s);

long long sum = 0;

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

if (s[i] <= '9')

t = s[i] - '0';

else

t = s[i] - 'A' + 10;

sum = sum \* 16 + t; //合并同列项

}

printf("%lld", sum);

return 0;

}

## 检测是否存在相同的字符串

#include <stdio.h>

#include <string.h>

int main()

{

    char str1[] = "Hello World!";

    char str2[] = "Hello";

    if ( !strstr(str1,str2) )

        printf("existed!");

    else printf("Not find!");

    return 0;

}

## Mallc与二维数组

char (\*ka)[2] = (char\*)malloc(sizeof(char)\*(n \* 2));

char (\*ka1)[2] = (char\*)malloc(sizeof(char)\*(n \* 2));

(\*ka1)[2] = { 0 };

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

scanf("%s", ka[i]);

}

## 初始化数组

你还可以用memset函数在程序开始时初始化数组。这条命令这在你已经修改了数组之后又想将它重置为全0特别有用。

头文件<memory.h>或<string.h>

int arr[1024];

arr[5] = 67;

memset(ZEROARRAY, 0, 1024); //This will reinitialize all to ZERO

## 题目1(空格断开)

输入第一行为一个正整数N（1≤N≤50）表示接下来需要求和的行数。

接下来N行中每行输入若干个正整数（保证至少有一个），每两个数用一个空格隔开。

#include<stdio.h>

**int** main()

{

**long** **long** n,num,sum=0;

**char** ch;

scanf("%lld",&n);

**while**(n)

{

scanf("%lld%c",&num,&ch);

sum+=num;

**if**(ch!=' ')

{

printf("%lld**\n**",sum);

sum=0;

n--;

}

}

**return** 0;

}

## 题目2（搬砖）

某工地需要搬运砖块，已知男人一人搬3块，女人一人搬2块，小孩2人搬一块。问用45人正好搬45块砖，有

多少种搬法？

#include<stdio.h>

int main(){

int a, b, c;

for (a = 0;a <= 15;a++)

for (b = 0;b < 23;b++) {

c = 45 - a - b;

if (a \* 3 + b \* 2 + c / 2 == 45)

printf("男人:%d,女人:%d,小孩:%d\n", a, b, c);

}

return 0;

}

## 题目3（最高分）

从键盘输入一批学生的成绩，找出并输出其中的最高分。

#include<stdio.h>

int main(){

int s = -1,max = -1;

printf("请输入一批成绩（0结束）:");

while (s) {

scanf("%d", &s);

if (s != 0 && s > max)

max = s;

}

printf("最高分=%d\n", max);

return 0;

}

## 题目4（做好事）

有四位同学中的一位做了好事，不留名，表扬信来了之后，校长问这四位是谁做的好事。

A说：不是我。  
B说：是C。  
C说：是D。  
D说：他胡说。‘’  
已知三个人说的是真话，一个人说的是假话。现在要根据这些信息，找出做了好事的人。

#include<stdio.h>

int main(){

int num;//说假话的人数

char x;//表示被假设做好事的人

for (x = 'A';x <= 'D';x++) {

num = 0;

if (x == 'A')

num++;

if (x != 'C')

num++;

if (x != 'D')

num++;

if (x == 'D')

num++;

if (num == 1) {

printf("做好事的是%c\n", x);

}

}

return 0;

}

## 题目5（保留小数位）

请编写一个函数float fun（double h），函数的功能是对变量h中的值保留2位小数，并对第3位进行四舍五入（规定h中的值为正数）。

#include<stdio.h>

float fun (double h){

int ih = (int)(h \* 1000);

if (ih % 10 > 4)

ih = ih / 10 + 1;

else

ih /= 10;

return ih / 100.0;

}

int main(){

double a;

printf("请输入a的值:");

scanf("%lf", &a);

printf("The original data is %lf\n", a);

printf("结果是:%.2f\n", fun(a));

return 0;

}

## 题目6（找数目）

编写一个函数，实现对一个字符串分别统计除其中英文字母，空格，数字和其他字符的数量。

#include<stdio.h>

void fun (char str[]){

int i = 0;

int w = 0, x = 0, y = 0, z = 0;

while (str[i] != '\0') {

if ((str[i] >= 'a'&&str[i] <= 'z') || (str[i] >= 'A'&&str[i] <= 'Z'))

w++;

else if (str[i] == ' ')

x++;

else

z++;

i++;

}

printf("w=%d,x=%d,y=%d,z=%d\n", w, x, y, z);

}

int main(){

char s[999999];

gets(s);

fun(s);

return 0;

}

## 取模运算法则

(a + b) % p = (a % p + b % p) % p

(a - b) % p = (a % p - b % p) % p

(a \* b) % p = (a % p \* b % p) % p

## 零碎

printf("%lld", (**long** **long**) n\*(n - 1));

scanf("%d",&T);

while(T--)

Cmn：

int isum=1;

for(k=1;k<=n;k++)

isum=(isum\*(m-n+k))/k;

四舍五入

double num；

int num1=floor（num+0.5）;

判断四位数num是否符合aabb：

Int hi=num/100;//取前两位

Int lo=num%100;//取后两位

If(hi/10==hi%10&&lo/10==lo%10)

在多数据的题目中，一个常见的错误是：在计算完一组数据后某些变量没有重置，影响到下组数据的求解。

const double pi = acos(-1.0);

不能读取空格：cin，cin.get(字符数组名，接收长度，结束符)，结束符为可选参数，默认为Enter，可以接受Space 、Tab，对于结束符的处理要注意，结束符并不会丢掉）

读取空格：cin.get()，gets（）（对象是char \*而getline（）对象是string）

**cin.getline(字符数组名，接收长度，结束符)**

其用法与cin.get(字符数组名，接收长度，结束符)极为类似。cin.get()当输入的字符串超长时，不会引起cin函数的错误，后面若有cin操作，会继续执行，只是直接从缓冲区中取数据。但是cin.getline()当输入超长时，会引起cin函数的错误，后面的cin操作将不再执行。

**getline(istream is(如cin),string str,结束符)**

## 走楼梯

一个共有10个台阶的楼梯，从下面走到上面。一次只能走一个台阶或两个台阶。

请问一共有多少种走法？

#include<iostream>

using namespace std;

int sum(int n){

if(n==1)

return 1;

else if(n==2)

return 2;

return sum(n-1)+sum(n-2);

}

int main(){

cout << sum(10);

return 0;

}

上第十个台阶可以从第八个上，或者从第九个上，那么上第十级的方法数为  
a(10)＝a(9)＋a(8)  
同理，可推知上第n级台阶的方法数等于上第n－1、n－2级台阶的方法数之和  
所以，由于a(1)＝1，a(2)＝2  
故a(1)至a(10)的方法数为  
1，2，3，5，8，13，21，34，55，89  
即上第十级有89种方法

## 整数拆分

题目描述：

小明想将一个正整数拆分成若干个正整数之和。请你帮他算一下共有多少种分法。

输入：

输入包含多组测试用例。

第一行包含一个整数T表示测试用例的个数。接下来的T行输入这T个测试用例。

每个测试用例包含两个正整数N和M，N为待拆的数，M为可拆出的数的最大值。N和M之间用一个空格隔开。（N、M均小于100）

输出：

对于每一个测试用例，输出拆分方法的种类数。

思路：

举个例子：

我们假设现在输入是5 2，即5拆成的数最大是2的种类数。

我们容易知道：

5拆成的数最大是1的种类数是1。即1+1+1+1+1。

5拆成的数最大是2的种类数是3。即1+1+1+1+1 1+1+1+2 1+2+2

3拆成的数最大是2的种类数是2。即1+1+1 1+2

注意绿色字体和红色字体，容易发现他们是一样的。

于是我们由上面的举例可以得到一些规律。我们假设一个函数F(n,m)是求n拆成的数最大是m的种类数，可以得到：

F(n,m)=F(n,m-1)+F(n-m,m) 对于上例即为F(5,2)=F(5,1)+F(3,2)

F(n,1)=1 对于上例即为F(5,1)=1

F(n,m)=F(n,n)（m>n时）

F(n,n)=1+F(n,n-1) 对于上例即为F(3,3)=F(3,2)+1。（加一的情况就是不拆分，即3）

#include<stdio.h>

**int** q(**int** n,**int** m)

{

**if**(n<1||m<1)

**return** 0;

**if**(m==1||n==1)

**return** 1;

**if**(m>n)

**return** q(n,n);

**if**(n==m)

**return** q(n,m-1)+1;

**return** q(n,m-1)+q(n-m,m);

}

**int** main()

{

**int** N,a,b,i;

scanf("%d",&N);

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

{

scanf("%d%d",&a,&b);

printf("%d**\n**",q(a,b));

}

**return** 0;

}

## 有规律

输入n，计算s=1！+2！+3！+…+n!的末6位（不含前导0）。N<10e6，n！表示前n个正整数之积。

样例输入：

10

样例输出：

37913

#include<iostream>

using namespace std;

int main(){

long long s=0;

long long s1=1;

const int mod=1000000;

int n;

cin >> n;

if(n>25)

n=25;

for(int i=1;i<=n;i++){

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

s1=(s1\*j%mod);

}

s=s+s1;

s1=1;

}

cout << s%mod;

return 0;

}

在N行杨辉三角中，设所有数和为Q，其中最大数为R。

求Q和N\*R的关系。

输入

包含一个正整数N。（1≤N≤108）

输出

若Q>N\*R，输出">"；若Q=N\*R，输出"="；若Q<N\*R，输出"<"。

#include<iostream>

#include<math.h>

using namespace std;

int main(){

    int n;

    cin >> n;

    if(n<10){

        int q=pow(2,n)-1,r=1;

        int k,m1,m;

        if(n%2==0){

            m1=n/2;

            m=n-1;

        }

        else{

            m1=(n-1)/2;

            m=n-1;

        }

        for(k=1;k<=m1;k++)

            r=(r\*(m-m1+k))/k;

        if(q>(n\*r))

            cout << ">";

        else if(q<(n\*r))

            cout << "<";

        else

            cout << "=";

    }

    else

      cout << "<";

    return 0;

}

## Cout控制小数点输出

保留小数点的功能需要**包含iomanip头文件**，并在输出数字前加上"<< fixed << setprecision(x)"，把x换成保留小数的位数。 fixed表示使用小数计数法;setprecision表示控制输出流显示浮点数的数字个数，如果和fixed合用的话，可以控制小数点右面的位数。

**例1：**cout << fixed << setprecision(4) << 3.14159 << endl;

3.1416

**例2：**cout << fixed << 3.14159 << endl;

3.141590

**例3：**cout << setprecision(4) << 3.14159 << endl;

3.142

然而却有了下面的问题。。。

|  |  |
| --- | --- |
| 1  2 | cout << fixed << setprecision(4) << 0.000101000110 << endl;  cout << 3.141593 <<endl; |

却得到了这样的输出结果

0.0001  
3.1416

没错，之后的输出被同样的输出格式所限制了。

那就要找到办法清除输出控制

遗憾的是 CPP 目前并没有办法消除 **setprecision**标签的效果方法，

但是消除 **fixed** 还是有的

在不需要 fixed 效果输出的前一行加上

|  |  |
| --- | --- |
| 1 | cout.unsetf( ios::fixed ); |

例如

|  |  |
| --- | --- |
| 1  2  3 | cout << fixed << setprecision(4) << 0.000101000110 << endl;  cout.unsetf( ios::fixed );  cout << 3.141593 <<endl; |

输出结果如下：

0.0001  
3.142

但可以通过再次设定 **setprecision**标签来达到控制位数

例如

|  |  |
| --- | --- |
| 1  2  3 | cout << fixed << setprecision(4) << 0.000101000110 << endl;  cout.unsetf( ios::fixed );  cout <<setprecision(6) << 3.141593 <<endl; |

输出结果如下：

0.0001  
3.14159

## 题目7 01序列

**问题描述**

　　对于长度为6位的一个01串，每一位都可能是0或1，一共有64种可能。它的前几个是：  
　　000000  
　　000001  
　　000010  
　　000011  
　　000100  
　　请按从小到大的顺序输出这64种01串。

**输出格式**

输出64行，每行一个01串。

代码1：

#include <stdio.h>

int main()

{

int a[6] = { 0,0,0,0,0,0 };

int i = 0;

for (i = 0;i<64;i++)

{

int j = i;

int k = 0;

for (k = 0;k<6;k++)

{

a[k] = j % 2;

j = j / 2;

}

k = 5;

while (k >= 0)

{

printf("%d", a[k]);

k--;

}

printf("\n");

}

return 0;

}

代码2：

#include <stdio.h>

int main()

{

int a1, a2, a3, a4, a5, a6;

for (a1 = 0;a1 <= 1;a1++)

for (a2 = 0;a2 <= 1;a2++)

for (a3 = 0;a3 <= 1;a3++)

for (a4 = 0;a4 <= 1;a4++)

for (a5 = 0;a5 <= 1;a5++)

for (a6 = 0;a6 <= 1;a6++)

printf("%d%d%d%d%d%d\n", a1, a2, a3, a4, a5, a6);

return 0;

}

## 题目8 字母图形

问题描述

利用字母可以组成一些美丽的图形，下面给出了一个例子：

ABCDEFG

BABCDEF

CBABCDE

DCBABCD

EDCBABC

这是一个5行7列的图形，请找出这个图形的规律，并输出一个n行m列的图形。

输入格式

输入一行，包含两个整数n和m，分别表示你要输出的图形的行数的列数。

输出格式

输出n行，每个m个字符，为你的图形。

样例输入

5 7

样例输出

ABCDEFG  
BABCDEF  
CBABCDE  
DCBABCD  
EDCBABC

数据规模与约定

1 <= n, m <= 26。

#include<stdio.h>

int main() {

int b, c, i, num, d;

char a[100][26];

scanf("%d %d", &b, &c);

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

for (num = 0;num < c;++num) {

d = num - i;

if (d < 0) {

d = -d;

}

a[i][num] = 'A' + d;

printf("%c", a[i][num]);

}

printf("\n");

}

return 0;

}

## 十进制转换为其他进制

#include <stdio.h>

void fun(int i,int k);

int main() {

int x,k;

scanf("%d %d",&x,&k);

fun(x,k);

return 0;

}

void fun(int x,int k) {

int num[1000];

int a,b,c=-1;

b=x;

do

{

a=b%k;

c++;

b=(b-a)/k;

x=x/k;

num[c]=a;

}

while(x>0);

for(;c>=0;c--)

{

if(num[c]>9)

printf("%c",num[c]+55);

else

printf("%d",num[c]);

}

}

## 排序

### 1.冒泡

#include<iostream>

using namespace std;

#define a 8

void Bubblesort(int \*arr, int b);

int main()

{

int num[a] = { 1,5,4,3,5,6,78,0 };

Bubblesort(num, a);

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

cout << num[i] << " ";

return 0;

}

void Bubblesort(int \*arr, int b)

{

int t;

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

{

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

{

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

{

int t = arr[j];

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

arr[j + 1] = t;

}

}

}

}

### 2.选择

#include<iostream>

using namespace std;

#define a 8

void SelectionSort(int \*arr, int b);

int main()

{

int num[a] = { 1,5,4,3,5,6,78,0 };

SelectionSort(num, a);

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

cout << num[i] << " ";

return 0;

}

void SelectionSort(int \*arr, int b)

{

int i, j;

for (i = 0;i < b-1;++i)

{

int min = arr[i],t,t1;

for (j = i+1;j < b;++j)

{

if (arr[j] <= min)

{

min = arr[j];

t1=j;

}

}

t = arr[i];

arr[i] = min;

arr[t1] = t;

}

}

### 3.插入

#include<iostream>

using namespace std;

#define num 8

void InsertSort(int \*arr);

int main()

{

int a[num] = { 1,5,4,3,5,6,78,0 };

InsertSort(a);

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

cout << a[i] << " ";

return 0;

}

void InsertSort(int \*arr)

{

int j, t;

for (int i = 1; i < num; i++) {

t = arr[i];

for (j = i - 1; j >= 0 && arr[j] > t; j--) {

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

}

arr[j + 1] = t;

}

}

## 任意进制转任意进制

#include <stdio.h>

#include<string.h>

#include<math.h>

void fun(int x, int k);//十进制转换成k进制

double fun1(char \*a, int num, int x);//k进制转换成十进制

int main()

{

int x, k;

char str[10000];

//scanf("%d", &x);

// scanf("%s", str);

//scanf("%d", &k);

while (scanf("%d %s %d", &x, str, &k)!=EOF)

{

int num = strlen(str);

if (k == 10)

printf("%0.0lf\n", fun1(str, num, x));

else

{

int temp = fun1(str, num, x);

fun(temp, k);

}

}

return 0;

}

void fun(int x, int k)

{

int num[10000];

int a, b, c = -1;

b = x;

do

{

a = b%k;

c++;

b = (b - a) / k;

x = x / k;

num[c] = a;

} while (x>0);

for (;c >= 0;c--)

{

if (num[c]>9)

printf("%c", num[c] + 55);

else

printf("%d", num[c]);

}

printf("\n");

}

double fun1(char \*a, int num, int k)

{

int i, t,t1=0;

double sum = 0;

for (i = num-1; i >= 0; i--)

{

if (a[i] >= '0' && a[i] <= '9')

t = a[i] - '0';

else if (k >= 11 && (a[i] >= 'A' && a[i] <= 'A' + k - 10))

t = a[i] - 'A' + 10;

sum = sum +(double)t\*pow(k, t1++);

}

return sum;

}

## 题目9 中值排序

现在有一个比较特别的排序需求：

现有包含N个数的数列需要按照接近M的程度排序。即越接近M的数我们认为它要排在前面，如果不相同但一样接近的话按数的大小排列。

如果还没看懂可以研究下样例及解释。

**输入**

输入包括两行。

第一行输入两个整数N、M。（1≤N≤106，,-106≤M≤106）

第二行输入N个整数为待排序的数列。

**输出**

输出排序后的序列。

**样例输入**

10 5

1 3 5 7 9 6 2 4 8 0

**样例输出**

5 6 4 7 3 8 2 9 1 0

**提示**  
对于样例：  
5跟5距离最近（0），所以在最前面。6,4与5的距离次之（1），但6>4，所以6在5后面，4在6后面。后面同理。

#include<iostream>

#include<cmath>

#include<cstdlib>

using namespace std;

int n, m;

int com(const void \* a, const void \* b)

{

return abs(\*(int\*)a - m) == abs(\*(int\*)b - m)? \*(int\*)b - \*(int\*)a : abs(((\*(int\*)a) - m)) - abs(((\*(int\*)b) - m));

}

int main()

{

cin >> n >> m;

int \*num = new int[n];

int i;

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

cin >> num[i];

qsort(num, n, sizeof(int), com);

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

cout << num[i] << " ";

delete[]num;

return 0;

}

## 题目十 第几个素数

给定两个整数N、M，请找出大于N且接近N的第M个素数。

**输入**

输出两个正整数N和M，两数之间用一个空格隔开。（1≤N、M≤106）

**输出**

输出题意要求的素数。

**样例输入**

5 10

**样例输出**

41

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

#define MAX 33000000

int prime[2100000] = { 1 };

void init()

{

//0-素数 1-合数

int \*num = (int\*)malloc(sizeof(int) \* MAX);

num[0] = 1;

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

num[i] = 0;

for (int i = 2; i < sqrt(MAX); i++)

{

if (!num[i])

{

for (int j = 2; j\*i <= MAX; j++)

{

if (num[i\*j] == 0)

num[i\*j] = 1;

}

prime[num[0]++] = i;

}

}

for (int i = sqrt(MAX) + 1; i < MAX; i++)

{

if (!num[i])

{

prime[num[0]++] = i;

}

}

}

int main()

{

init();

int n, m;

scanf("%d%d", &n, &m);

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

if (prime[i] > n)

{

printf("%d", prime[i + m - 1]);

break;

}

return 0;

}

## 快速幂取模

long long mi(long long a, int b)

{

long long ans = 1;

a %= mo;

while (b)

{

if (b & 1) ans = ans\*a%mo;

b >>= 1;

a = a\*a%mo;

}

return ans;

}

## Farey数列

void farey(int n)

{

    // We know first two terms are 0/1 and 1/n

    double x1 = 0, y1 = 1, x2 = 1, y2 = n;

    printf("%.0f/%.0f %.0f/%.0f", x1, y1, x2, y2);

    double x, y = 0;  // For next terms to be evaluated

    while (y != 1.0)

    {

        // Using recurrence relation to find the next term

        x = floor((y1 + n) / y2) \* x2 - x1;

        y = floor((y1 + n) / y2) \* y2 - y1;

        // Print next term

        printf(" %.0f/%.0f", x, y);

        // Update x1, y1, x2 and y2 for next iteration

        x1 = x2, x2 = x, y1 = y2, y2 = y;

    }

}

// Driver program

int main()

{

    int n = 7;

    cout << "Farey Sequence of order " << n << " is\n";

    farey(n);

    return 0;

}

## 高精度加法

#include<iostream>

#include<cstring>

using namespace std;

const int m=10000;

char num1[m]=;

char num2[m]=;

void move\_num(char num[])

{

int i;

int n=strlen(num);

int j=m-1;

for(i=1;i<=n;i++)

{

num[j]=num[n-i];

j--;

}

while(j>=0)

{

num[j]='0';

j--;

}

}

void conv\_num(char num[])

{

int i;

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

{

num[i]-='0';

}

}

void add(char num1[],char num2[])

{

int i;

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

{

num1[i]=num1[i]+num2[i];

if(num1[i]>=10)

{

num1[i-1]+=num1[i]/10;

num1[i]=num1[i]%10;

}

}

}

int main()

{

cin >> num1;

cin >> num2;

int i,temp;

move\_num(num1);

move\_num(num2);

conv\_num(num1);

conv\_num(num2);

add(num1,num2);

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

{

if(num1[i]!=0)

{

temp=i;

break;

}

}

for(i=temp;i<m;++i)

{

cout << (int)num1[i];

}

return 0;

}