1. 环状表示 ： s[p+i]%n 或 p = (p+k+n-1) % n +1 // k为步长，n为周期
2. #include <ctype.h>:

Isalpha,idigit,isprint(是否为可输出字符)

1. char buff[200];  
   sprintf(buf,”%d%d”,&a,&b); // 将int型变量a，b装到字符数组buf中
2. char s[200],  
   if(strchr(s,’A’) == NULL); //判断s字符串中是否有A字符  
   int \*p = strchr(s,’A’); //将s串中首次出现b字符的位置赋值给p指针
3. int a[100],b[100];  
   memspy(b,a,sizeof(a));//将a中所有元素从b的首地址开始复制到b中
4. #include <algorithm>:  
    hypot函数：计算三角形斜边
5. map： hdu 1075

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <string>

#include <cstring>

//#include <input.h>

#include <map>

using namespace std;

int main()

{

map<string,string>dic;

string a,b;

cin>>a;

while(cin>>a &&a != "END"){

cin>>b;

dic[b] = a;

}

cin>>a;

getchar();

char str[1002];

while(gets(str)){

if(strcmp(str,"END") == 0)

break;

a = "";

for(int i=0 ; i<strlen(str) ; i++){

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

a += str[i];

else if(a == "")

printf("%c",str[i]);

else{

if(dic[a] == "")

printf("%s",a.c\_str());

else

printf("%s",dic[a].c\_str());

a = "";

printf("%c",str[i]);

}

}

printf("\n");

}

return 0;

}

1. 辗转相除法：  
   int f(int x, int y){

return y == 0 ? x : f(y,x%y);

}

1. C++输出2进制数：

(oct,dec,hex分别代表八进制,十进制,十六进制; bin代表二进制,但是不能cout<<bin;)

#include <bitset>

using namespace std;

int main(){

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

cout<<bitset<4>(i)<<" "; // < >中代表输出位数 I 代表输出的数

return 0;

}

1. i<<8 //表示i \* 2的8次方 <<为移位符
2. sort(\*p , \*p+k , cmp); // \*p + k 为超出所要排的元素的下一个元素

bool cmp(int a,int b) {  
      return a<b;   //升序排列(默认)，如果改为return a>b，则为降序

}

1. 走迷宫/在数组中递归前进时利用：

int dx[4] = {-1,0,1,0};

int dy[4] = {0,1,0,-1};

for(int i=0 ; i<4 ; i++) {……} //前进

if(x<1 || x>X || y<1 || y>Y ||…(其他控制条件)…) {……} //判断是否出边线

1. 任意一个正整数a的位数

等于(int)log10(a) + 1；推导：

对于任意一个给定的正整数a，

假设10^(x-1) <= a < 10^x，那么显然a的位数为x位，

又因为

log10(10^(x-1))<=log10(a)<(log10(10^x))

即x-1<=log10(a)<x

则(int)log10(a)=x-1,

即(int)log10(a)+1=x

即a的位数是(int)log10(a)+1

1. 斯特林公式:

https://imgsa.baidu.com/baike/s%3D112/sign=4aec460be7cd7b89ed6c3e823d264291/4bed2e738bd4b31c19ead27a86d6277f9f2ff819.jpg 准确：https://imgsa.baidu.com/baike/s%3D149/sign=61f142202e2eb938e86d7ef6ec6385fe/9d82d158ccbf6c81b51c6d29bd3eb13532fa40c3.jpg

1. scanf(“%s”,str); //一定要记住memset(首地址输入不代表把数据清除)
2. stack 栈地简单应用：hdu 1022
3. 卡特兰数及应用：

**递推式[2]****：h(n)=h(n-1)\*(4\*n-2)/(n+1);**

**递推关系的解为：h(n)=C(2n,n)/(n+1) (n=0,1,2,...)**

**递推关系的另类解为：h(n)=c(2n,n)-c(2n,n-1)(n=0,1,2,...)**

**问题描述：卡塔兰数，是组合数学中一个常出现在各种计数问题中出现的数列。输入一个整数n，计算h(n)。其递归式如下：h(n)= h(0)\*h(n-1)+h(1)\*h(n-2) + ... + h(n-1)h(0) (其中n>=2，h(0) = h(1) = 1)    该递推关系的解为：h(n)=C(2n,n)/(n+1) (n=1,2,3,...)**

思路：直接根据递归式，写出相应的[**算法**](http://lib.csdn.net/base/datastructure)。

        参考代码：

1. //函数功能: 计算Catalan的第n项
2. //函数参数: n为项数
3. //返回值:   第n个Catalan数
4. **int** Catalan(**int** n)
5. {
6. **if**(n <= 1)
7. **return** 1;
9. **int** \*h = **new** **int** [n+1]; //保存临时结果
10. h[0] = h[1] = 1;        //h(0)和h(1)
11. **for**(**int** i = 2; i <= n; i++)    //依次计算h(2),h(3)...h(n)
12. {
13. h[i] = 0;
14. **for**(**int** j = 0; j < i; j++) //根据递归式计算 h(i)= h(0)\*h(i-1)+h(1)\*h(i-2) + ... + h(i-1)h(0)
15. h[i] += (h[j] \* h[i-1-j]);
16. }
17. **int** result = h[n]; //保存结果
18. **delete** [] h;       //注意释放空间
19. **return** result;
20. }

**应用1描述：n对括号有多少种匹配方式？**

       思路：n对括号相当于有2n个符号，n个左括号、n个右括号，可以设问题的解为f(2n)。第0个符号肯定为左括号，与之匹配的右括号必须为第2i+1字符。因为如果是第2i个字符，那么第0个字符与第2i个字符间包含奇数个字符，而奇数个字符是无法构成匹配的。

       通过简单分析，f(2n)可以转化如下的递推式 f(2n) = f(0)\*f(2n-2) + f(2)\*f(2n - 4) + ... + f(2n - 4)\*f(2) + f(2n-2)\*f(0)。简单解释一下，f(0) \* f(2n-2)表示第0个字符与第1个字符匹配，同时剩余字符分成两个部分，一部分为0个字符，另一部分为2n-2个字符，然后对这两部分求解。f(2)\*f(2n-4)表示第0个字符与第3个字符匹配，同时剩余字符分成两个部分，一部分为2个字符，另一部分为2n-4个字符。依次类推。

       假设f(0) = 1，计算一下开始几项，f(2) = 1, f(4) = 2, f(6) = 5。结合递归式，不难发现**f(2n) 等于h(n)**。

**应用2描述：矩阵链乘： P=a1×a2×a3×……×an，依据乘法结合律，不改变其顺序，只用括号表示成对的乘积，试问有几种括号化的方案？**

       思路：可以这样考虑，首先通过括号化，将P分成两个部分，然后分别对两个部分进行括号化。比如分成(a1)×(a2×a3.....×an)，然后再对(a1)和(a2×a3.....×an)分别括号化；又如分成(a1×a2)×(a3.....×an)，然后再对(a1×a2)和(a3.....×an)括号化。

       设n个矩阵的括号化方案的种数为f(n)，那么问题的解为

        f(n) = f(1)\*f(n-1) + f(2)\*f(n-2) + f(3)\*f(n-3) + f(n-1)\*f(1)。f(1)\*f(n-1)表示分成(a1)×(a2×a3.....×an)两部分，然后分别括号化。

       计算开始几项，f(1) = 1, f(2) = 1, f(3) = 2, f(4) = 5。结合递归式，不难发现**f(n)等于h(n-1)**。

**应用3描述：一个栈(无穷大)的进栈序列为1，2，3，…，n，有多少个不同的出栈序列?**

      思路：这个与加括号的很相似，进栈操作相当于是左括号，而出栈操作相当于右括号。n个数的进栈次序和出栈次序构成了一个含2n个数字的序列。第0个数字肯定是进栈的数，这个数相应的出栈的数一定是第2i+1个数。因为如果是2i，那么中间包含了奇数个数，这奇数个肯定无法构成进栈出栈序列。

       设问题的解为f(2n)， 那么f(2n) = f(0)\*f(2n-2) + f(2)\*f(2n-4) + f(2n-2)\*f(0)。f(0) \* f(2n-2)表示第0个数字进栈后立即出栈，此时这个数字的进栈与出栈间包含的数字个数为0，剩余为2n-2个数。f(2)\*f(2n-4)表示第0个数字进栈与出栈间包含了2个数字，相当于1 2 2 1，剩余为2n-4个数字。依次类推。

       假设f(0) = 1，计算一下开始几项，f(2) = 1, f(4) = 2, f(6) = 5。结合递归式，不难发现**f(2n) 等于h(n)**。

**应用4描述：n个节点构成的二叉树，共有多少种情形？**

       思路：可以这样考虑，根肯定会占用一个结点，那么剩余的n-1个结点可以有如下的分配方式，T(0, n-1),T(1, n-2),...T(n-1, 0)，设T(i, j)表示根的左子树含i个结点，右子树含j个结点。

       设问题的解为f(n)，那么f(n) = f(0)\*f(n-1) + f(1)\*f(n-2) + .......+ f(n-2)\*f(1) + f(n-1)\*f(0)。假设f(0) = 1，那么f(1) = 1, f(2) = 2, f(3) = 5。结合递推式，不难发现**f(n)等于h(n)**。

**应用5描述：在圆上选择2n个点，将这些点成对连接起来使得所得到的n条线段不相交的方法数？**

       思路：以其中一个点为基点，编号为0，然后按顺时针方向将其他点依次编号。那么与编号为0相连点的编号一定是奇数，否则，这两个编号间含有奇数个点，势必会有个点被孤立，即在一条线段的两侧分别有一个孤立点，从而导致两线段相交。设选中的基点为A，与它连接的点为B，那么A和B将所有点分成两个部分，一部分位于A、B的左边，另一部分位于A、B的右边。然后分别对这两部分求解即可。

       设问题的解f(n)，那么f(n) = f(0)\*f(n-2) + f(2)\*f(n-4) + f(4)\*f(n-6) + ......f(n-4)\*f(2) + f(n-2)\*f(0)。f(0)\*f(n-2)表示编号0的点与编号1的点相连，此时位于它们右边的点的个数为0，而位于它们左边的点为2n-2。依次类推。

       f(0) = 1, f(2) = 1, f(4) = 2。结合递归式，不难发现**f(2n) 等于h(n)**。

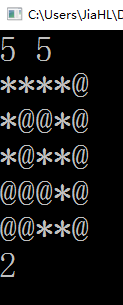
**应用6描述：求一个凸多边形区域划分成三角形区域的方法数？**

      思路：以凸多边形的一边为基，设这条边的2个顶点为A和B。从剩余顶点中选1个，可以将凸多边形分成三个部分，中间是一个三角形，左右两边分别是两个凸多边形，然后求解左右两个凸多边形。

      设问题的解f(n)，其中n表示顶点数，那么f(n) = f(2)\*f(n-1) + f(3)\*f(n-2) + ......f(n-2)\*f(3) + f(n-1)\*f(2)。f(2)\*f(n-1)表示三个相邻的顶点构成一个三角形，那么另外两个部分的顶点数分别为2和n-1。

      设f(2) = 1，那么f(3) = 1, f(4) = 2, f(5) = 5。结合递推式，不难发现**f(n) 等于h(n-2)**。

**应用7描述：有2n个人排成一行进入剧场。入场费5元。其中只有n个人有一张5元钞票，另外n人只有10元钞票，剧院无其它钞票，问有多少中方法使得只要有10元的人买票，售票处就有5元的钞票找零？**

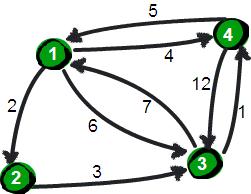
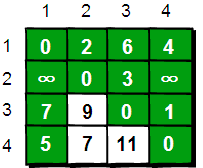
     思路：可以将持5元买票视为进栈，那么持10元买票视为5元的出栈。这个问题就转化成了栈的出栈次序数。由应用三的分析直接得到结果，**f(2n) 等于h(n)**。

1. scanf("%s",&field[i][1]); // 可以从field[i][1]位开始输入字符串，注意field[i][1]前&符号!

scanf(“%c”,xxxx)//尽量不使用，尽量用getchar(); 如右图所示: scanf(“%d%d”,&a,&b)后，一定要getchar( ) 怼换行符（ getchar()可以读取‘\n’ ）

而这种录入字符串的，用scanf("%s",&field[i][1]);较好

1. Floyd-Warshall算法：找最短路径 （hdu 4848）

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

    for(i=1;i<=n;i++)

        for(j=1;j<=n;j++)

            if(e[i][j]>e[i][k]+e[k][j])

                 e[i][j]=e[i][k]+e[k][j];

1. 若遇到dfs深搜边界 如: 1<= n <= 100000; 若数据能\*2 则数组要开到200000
2. 运算符重载: if(p\_nx == end) =>需加下面语句 (pn\_x,end 都是结构体)

bool operator==(pos &t1,pos &t2){

return (t1.x==t2.x)&&(t1.y==t2.y);}

22. struct pos{

int x,y;

pos(){ }

pos(int a,int b){ // 可以实现: p\_nx = pos(p\_nw.x+p\_x[i],p\_nw.y+p\_y[i]);

x = a; // 参见hdu2612 二解

y = b;

}

}Y,M,KFC[4005];

1. bool函数返回默认不是false/true 因此要有return;
2. map <int,int> M;

map<int,int>::iterator it;

for(map<int,int>::iterator it=M.begin() ; it!=M.end() ; it++)

it -> first //指的前一个int(key) it -> second //指的后一个int(value)

1. typedef long long LL; //对类型定义
2. 对long long 类型初始化： long long a = 12345678901234LL
3. floor(),ceil() 分别为向下、向上取整

round() 四舍五入函数:

eg: double a = 1.23563; printf("%d",int(round(a)));

1. 如果数组开的过大(＞10^6)则需将数组生命在函数外面,因为函数内部申请的局部变量来自系统栈,允许申请的空间较小,而函数外部申请的全局变量来自静态存储区,申请的空间较大
2. memset是按字节赋值因此，赋值0或-1 不容易出错
3. gets()可以得到\n,因此在输入一个数字n后,要用getchar(),接收数字后的换行

eg: int a; char str[100] ; scanf(“%d”,&a); getchar(); gets();

1. **sscanf(),sprintf() 应用：**

int n1,n2;

char str1[100],str2[100];

sscanf(str1,"%d",&n1); //将str1 中的字符转换为整形复制给n1

sprintf(str2,"%s",n2); //将n2 这个整形数 换位 字符串 存在str2中

double m1,m2=3.14;

char s1={2048:3.14,hello} ,s2;

sscanf(s1,"%d:%lf,%s",&n1,&m1,&str1); //将s1“分解”给n1,m1,str,其中:与,为正则表达式

sprintf(s2,"%d:%lf,%s",n2,m2,str2); //将n2,m2,str2“合并”赋值给s2

1. scanf正则表达式: <http://blog.csdn.net/tujiaw/article/details/6139896>
2. 常用宏定义：

#define eps 1e-8

#define Pi acos(-1.0)

#define Equ(a,b) ((fabs((a)-(b)))<(eps))

#define More(a,b) (((a)-(b))>(eps))

#define Less(a,b) (((a)-(b))<(-eps))

#define MoreEqu(a,b) (((a)-(b))>(-eps))

#define LessEqu(a,b) (((a)-(b))<(eps))

1. 结构体初始化：

struct pos{

int x,y;

pos(){}

pos(int \_x,int \_y):x(\_x),y(\_y) { }

}A,B;

B = pos(3,4); //则此行可运行

A = B; //则此行可运行

1. char str[1000];

while(gets(str) != NULL){ … } ====while(scanf(“%d”,&) != EOF){ … }

1. max 需头文件#include <algorithm>
2. 结构体sort:

方法一: bool operator < (const node tmp)const {

if(u==tmp.u)

return v<tmp.v;

return u<tmp.u;

} sort(arr,arr+k); //重载法

方法二: bool cmp(node a,node b){

if(a.u != b.u)

return a.u > b.u;

return a.v > b.v;

} sort(arr,arr+k,cmp); //声明函数法

37.next\_permutation: 2 1 1 3 🡪 2 1 3 1 //两个1 不会重复排列

38. Cantor 展开

#include <iostream>

#include <stdio.h>

#include <algorithm>

#include <cstring>

#include <vector>

using namespace std;

typedef long long LL;

const int f[10] = {1,1,2,6,24,120,720,5040};

//康托展开逆运算

//康托展开

LL Work(char str[])

{

int len = strlen(str);

LL ans = 0;

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

{

int tmp = 0;

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

if(str[j] < str[i]) tmp++;

ans += tmp \* f[len-i-1];

}

return ans;

}

void Work(LL n,LL m)

{

n--;

vector<int> v;

vector<int> a;

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

v.push\_back(i);

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

{

LL r = n % f[i-1];

LL t = n / f[i-1];

n = r;

sort(v.begin(),v.end());

a.push\_back(v[t]);

v.erase(v.begin()+t);

}

vector<int>::iterator it;

for(it = a.begin();it != a.end();it++)

cout<<\*it;

cout<<endl;

}

int main(){

int a,b;

char c[100];

cin>>c;

int ans = Work(c);

cout<<ans<<endl;

cin>>a>>b;

Work(a,b);

return 0;

}

39: 01背包问题：<http://www.cppblog.com/tanky-woo/archive/2010/07/31/121803.html>

HDU 2062 一维数组需要逆序循环！

40. #define maxn 0x3f3f3f3f

int dp[1000];

memset(dp, 0x3f, sizeof dp);// 只有整形数组可以如此使用，double型数组不行

41.优先队列 结构体：

bool operator < (pos a,pos b){

return a.x < b.x;

}

42.POJ 1511 二解 结构体+优先队列