程序设计实训 练习4

全排列问题(form.cpp)

描述

输出自然数 1 到 n 所有不重复的排列，即 n 的全排列，要求所产生的任一数字序列中不允许出现重复的 数字。

输入

n(1≤n≤9)

输出

由 1～n 组成的所有不重复的数字序列，每行一个序列。

同一行的两个数字之间用一个空格隔开。

输入样例 1

```
3
```

输出样例 1

```
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
```

```
#include <bits/stdc++.h>
using namespace std;

int i,j;
const int maxn=10+5;
int digit[maxn],temp[maxn];
bool judge[maxn];                    //判断数字是否重复出现
int n,cnt;

inline void sequence(int cnt) {
    int i;
    if(cnt==n)   {        //n个数字全部填入完成,即找打一种排列方案
        for(i=0;i<n;i++)
            printf("%5d",temp[i]);
        cout<<endl;
    }
    for(i=0;i<n;i++)        //枚举数字1-n
        if(judge[i]==0)  {  //如果当前数字曾出现过，直接跳过
            judge[i]=1;
            temp[cnt]=digit[i];
            sequence(cnt+1);
            judge[i]=0;    //在回溯到上一层之前，清除本层的标记
        }
}
```

BY 162210107 蔡蕾

```
int main()    {
    cin>>n;
    for(int i=0;i<n;i++)
        digit[i]=i+1;

    sequence(0);
    return 0;
}
```

迷宫问题(migong)

描述

设有一个 N*N(2<=N<20)方格的迷宫，入口和出口分别在左上角和右上角。迷宫格子中 分别放 0 和 1，

0 表示可通，1 表示不能通过，入口和出口处肯定是 0。

迷宫走的规则如下所示:

即从某点开始，有八个方向可走，前进方格中数字为 0 时表示可通过，为 1 时表示不可通过， 要另找路径。

找出所有从入口（左上角）到出口（右上角）的路径(不能重复)，输出路径总数，如果无法到达，则输出 0。

输入

第一行一个整数n

接下来n行每行n个数字，仅包含0或1，用来描述迷宫

输出

仅一行，从入口到出口的路径总数

输入样例 1

```
3
0 0 0
0 1 1
1 0 0
```

输出样例 1

```
2
```

BY 162230217 陈梓鹏

```
#include <iostream>
#include <vector>
using namespace std;
int dx[8] = {0, -1, 0, 1, 1, 1, -1, -1};
int dy[8] = {1, 0, -1, 0, 1, -1, 1, -1}; //可以移动的八个方向
int count = 0;
int n = 0;
int arr[11][11];
int a[11][11];                          // 记录是否走过
void dfs(int x, int y)  {
    if (x == 1 && y == n)  {     //到达(1,n)点，说明找到一种方案
        count++;
        return;
    }
    for (int i = 0; i < 8; i++)  { //共8种移动方向，依次尝试
        a[x][y] = 1;
        int nx = x + dx[i];
        int ny = y + dy[i];
        if (nx >= 1 && nx <= n && ny >= 1 && ny <= n && arr[nx][ny] != 1
&& a[nx][ny] != 1)  {   //判断是否为地图边界或曾走过的点
            a[nx][ny] = 1;
            dfs(nx, ny);
            a[nx][ny] = 0;  //在回溯上一层前，清除本层标记
        }
    }
}
```

```
int main(void)  {
    cin >> n;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
            cin >> arr[i][j];
    dfs(1, 1);
    cout << count;
    return 0;
}
```