项目说明文档
——N 皇后问题

姓名：吴桐欣
学号：1652677
同济大学 软件学院 软件工程专业

目录

1.项目概述

1.1 项目简介

八皇后问题是一个古老而著名的问题，是回溯算法的经典问题。该问题是十九世纪著名的数学家高斯在 1850 年提出的：在 8*8 的国际象棋棋盘上，安放 8 个皇后，要求没有一个皇后能够“吃掉”任何其它一个皇后，即任意两个皇后不能处于同一行，同一列或者同一条对角线上，求解有多少种摆法。
本实验拓展了 N 皇后问题，即皇后个数由用户输入。

1.1.1 功能分析

本项目要求对 N 皇后问题求解，最直接的功能就是
（1）输出所有的解题方案；
（2）计算出最多有多少种方案。

1.2 文件目录

（1）P04_1652677_吴桐欣_说明文档.docx（本文档）
（2）P04_1652677_吴桐欣.exe（可执行文件）
（3）P04_1652677_吴桐欣.cpp（源文件）
（4）P04_1652677_吴桐欣.h（头文件）

1.3 操作指南

运行程序后，将获得程序提示“输入皇后个数 n(0<n<11):”
用户输入一个数字，即可获得结果
[输入样例]
4
[输出样例]
皇后摆法:
[方案 #1]
0 x 0 0
0 0 0 x
x 0 0 0
0 0 x 0

[方案 #2]
0 0 x 0
x 0 0 0
0 0 0 x
0 x 0 0

共有 2 种解法

1.4 注意事项

（1）用户不得输入除数字以外的字符

（2）用户所输入的数字 n 需符合程序要求，大于 0 且小于 11

2.思路与设计

2.1 基本思路

此题用到回溯法，用栈对已放置的皇后进行存储。回溯时，依次弹出栈内的皇后。

2.2 设计

2.2.1 数据结构

为了存储及读取数据的方便，用 vector 来存储所放置的皇后的位置，vector 的 back()和 pop_back()函数可以模拟 stack 的 top()和 pop()函数。

2.2.2 成员与成员函数

```
struct Point{
    int x;//行
    int y;//列
};
```

为了存储数据直观，用 Point 结构将棋盘上每一个位置的行(x)、列(y)数都放在一个 Point 中，使一个棋盘位置只对应一个 Point。

```
struct List{
    int n;//棋盘大小
    vector<bool> row;//行
    vector<bool> col;//列
    vector<bool> mainDiag;//主对角线（左上-右下）
    vector<bool> minorDiag;//副对角线（右上-左下）

    bool onList(Point p)const;//检查某一位置是否可放皇后
    void initialize();//初始化 4 个数组
    void addPoint(Point p);//新增一个皇后，更新数组
    void deletePoint(Point p);//回溯时移走一个皇后，更新数组
};
```

每一个皇后放置之后，都会对后面的皇后的放置造成影响，为了方便检查棋盘上的位置是否可放，用 List 结构包装 4 个 vector 数组，为节省空间，存储值为 bool 值。值为 true 表示这一行或这一列或这一对角线上没有皇后，值为 false 则表示已经放有皇后。
initialize(),addPoint(),deletePoint()函数是对内部数据进行修改，也代表着棋盘上的变化。
onList()函数是外部对内部数据的读取、比较、做出判断，不对任何值做修改，因此设置为 const。

3.具体实现

3.1 函数的实现

3.1.1 onList()函数

函数代码

```
bool List::onList(Point p)const
{
    if (row[p.x] == true
        && col[p.y] == true
        && mainDiag[p.x-p.y+n-1] == true
        && minorDiag[p.x+p.y] == true) {
        //当且仅当 p 位置在 4 个数组中对应元素值均为 true 时，p 位置才可放置皇后
        return true;
    } else {
        return false;
    }
}
```

说明

返回值为 bool 值，返回 true 表示位置 p 可以放置皇后，返回 false 表示不可以。
主对角线上，所有位置的 x 与 y 的差是相等的，x-y 最小值是-(n-1)，要使每一条对角线在数组中有对应的下标，需要将 x-y 值全部加上(n-1)；
副对角线上，所有位置的 x 与 y 的和是相等的，x+y 的取值范围是 0~(2n-2)，可直接作为数组对应的下标。

3.1.2 initialize()函数

函数代码

```
void List::initialize(){
    for (int i=0; i<n; i++) {
        row.push_back(true);
        col.push_back(true);
    }
    for (int i=0; i<(2*n-1); i++) {
        mainDiag.push_back(true);
        minorDiag.push_back(true);
    }
}
```

说明

根据数组大小的不同，用两个 for 循环对数组进行初始化

3.1.3 addPoint()函数

函数代码

```
void List::addPoint(Point p){
    row[p.x] = false;
    col[p.y] = false;
    mainDiag[p.x-p.y+n-1] = false;
    minorDiag[p.x+p.y] = false;
}
```

说明

放置一个皇后，其所在的行、列、对角线设置为 false，表示相应的行、列、对角线上不能再放皇后。

3.1.4 deletePoint()函数

```
void List::deletePoint(Point p){
    row[p.x] = true;
    col[p.y] = true;
    mainDiag[p.x-p.y+n-1] = true;
    minorDiag[p.x+p.y] = true;
}
```

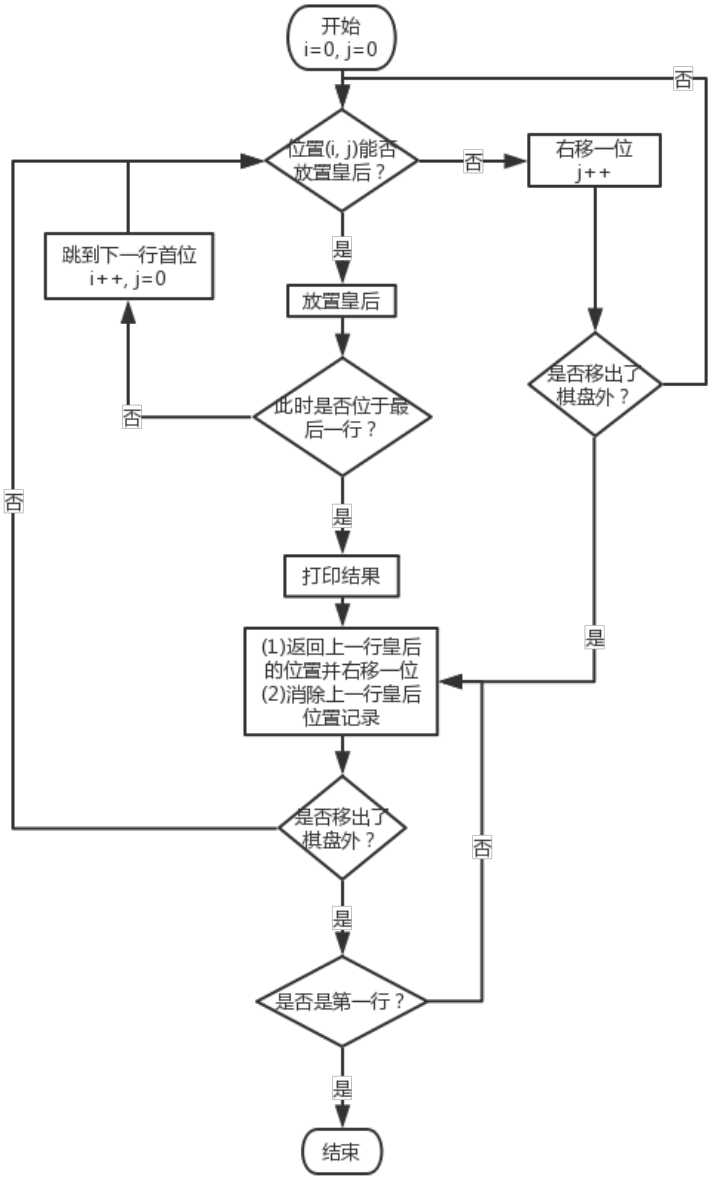
函数代码

说明

移走一个皇后，其所在的行、列、对角线设置为 true，表示相应的行、列、对角线上可以放皇后。

3.2 回溯法的实现

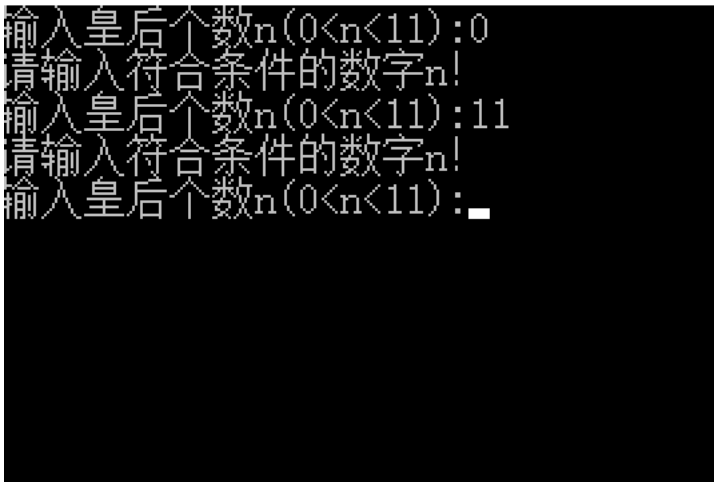
回溯法流程图



4.测试

4.1 边界测试

输入数字太大或太小



4.2 出错测试

输入非法字符