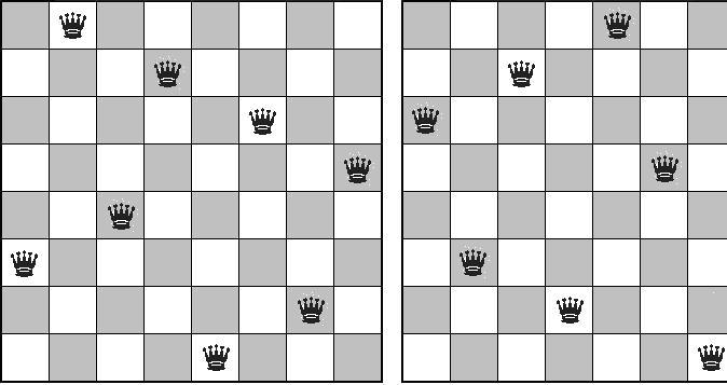
**实验三 八皇后问题**

**一.实验目的。**

编写实现输出八皇后问题所有解的程序

**二.实验分析。**

在8×8格的国际象棋上摆放八个皇后，使其不能互相攻击，即任意两个皇后都不能处于同一行、同一列或同一斜线上。



这次实验使用递归算法实现求解八皇后问题。

**三.实验内容。**

**源代码：C++实现**

#include <iostream>

using namespace std;

class ChessBoard {

public:

ChessBoard(); // 8 x 8 chessboard;自定义构造函数

ChessBoard(int); // n x n chessboard;带有参数的构造函数

void findSolutions();

private:

const bool available;

const int squares, norm;//squares代表棋盘格的边长

bool \*column, \*leftDiagonal, \*rightDiagonal;//定义列，左斜线以及右斜线

int \*positionInRow, howMany;//定义行以及方法的数量

char m[10][10];//记录棋盘格

void putQueen(int);

void printBoard();

void initializeBoard();

void Delete();

};

ChessBoard::ChessBoard() : available(true), squares(8), norm(squares - 1) {

initializeBoard();

}

ChessBoard::ChessBoard(int n) : available(true), squares(n), norm(squares - 1) {

initializeBoard();

}

void ChessBoard::initializeBoard() {

register int i;//将整数i寄存器，目的使的运算更快

column = new bool[squares];

positionInRow = new int[squares];

leftDiagonal = new bool[squares \* 2 - 1];//左斜线的数目

rightDiagonal = new bool[squares \* 2 - 1];//右斜线的数目

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

positionInRow[i] = -1;

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

column[i] = available;//将每一列都设置为可以放置皇后的情况

for (i = 0; i < squares \* 2 - 1; i++)

leftDiagonal[i] = rightDiagonal[i] = available;

howMany = 0;

}

void ChessBoard::printBoard() {

howMany++;//

cout << howMany << " way is:" << endl;

//为棋盘格赋值为1

for (int i = 0;i != squares;i++) {

for (int j = 0;j != squares;j++)

m[i][j] = '1';

}

//将皇后的位置在棋盘格上用'\*'标志出来

for (int row = 0;row != squares;row++)

m[row][positionInRow[row]] = '\*';

//打印棋盘格

for (int i = 0;i != squares;i++) {

for (int j = 0;j != squares;j++)

cout << m[i][j]<<" ";

cout << endl;

}

cout << endl;

}

void ChessBoard::putQueen(int row) {

for (int col = 0; col < squares; col++) {

if (column[col] == available &&

leftDiagonal[row + col] == available &&

rightDiagonal[row - col + norm] == available)

{

positionInRow[row] = col;

column[col] = !available;

leftDiagonal[row + col] = !available;

rightDiagonal[row - col + norm] = !available;

if (row < squares - 1)

putQueen(row + 1);

else printBoard();

column[col] = available;

leftDiagonal[row + col] = available;

rightDiagonal[row - col + norm] = available;

}

}

}

void ChessBoard::Delete() {

delete[]column;

delete[]positionInRow;

delete[]leftDiagonal;

delete[]rightDiagonal;

}

void ChessBoard::findSolutions() {

putQueen(0);

cout << howMany << " solutions found.\n";

Delete();

}

int main() {

ChessBoard board(8);

board.findSolutions();

return 0;

}

运行结果：

