八皇后问题(C++)

1、问题描述：在一个 8*8 的棋盘上放置 8 个皇后，不允许任何两个皇后在棋盘的同一行、同一列和同一对角线上。

2、关键字：递归、上溯

3、技巧：

1）、

经观察发现，对 8 x 8 的二维数组上的某点 a[i][j](0<=i,j<=7)

其主对角线（即左上至右下）上的每个点的 i-j+7 的值（范围在(0,14)）均相等；

其从对角线（即右上至左下）上的每个点的 i+j 的值（范围在(0,14)）均相等；

且每个主对角线之间的 i-j+7 的值均不同，每个从对角线之间的 i-j+7 的值亦不同；

如 a[3][4]:

主：3-4+7=6

从：3+4=7

因此可设两个数组 b[15],c[15]分别表示主、从对角线是否安全

（为 1 表示有皇后，不安全；为 0 表示安全）

2）、

每行有且仅有一个皇后：

每 i 个皇后放在每 i 行(0<=i<=7)

void eightQueens(int line);

4、源码（C++）

```
//eight_queens.cpp
#include <iostream>
using namespace std;
int data[ 8 ][ 8 ]; //chess(double dimensional array)
int a[ 8 ];     //column(列)
int b[ 15 ];   //主对角线(左上至右下)
int c[ 15 ];   //从对角线(右上至左下)
int count = 0;

void eightQueens( int );
void output( const int [][ 8 ], int );

int main()
{
  int i, j;

  for( i = 0; i < 15; ++i ) //主、从对角线
```

```
    b[ i ] = c[ i ] = 0; //表示安全

  for( i = 0; i < 8; ++i )//chess
  {
    a[ i ] = 0;        //i 列安全
    for( j = 0; j < 8; ++j )
      data[ i ][ j ] = 0;
  }

  eightQueens( 0 );

  cout << "/ncount = " << count << endl;
  return 0;
}

void eightQueens( int line )
{
  if( 8 == line )//八个皇后安置就位，输出
  {
    output( data, 8 );
    cout << endl;
    return;
  }


  for( int column = 0; column < 8; ++column )
  {
    if( 0 == a[ column ] && 0 == b[ line - column + 7 ] && 0 == c[ line + column ] )
    {
      data[ line ][ column ] = 1; //
安置皇后
      a[ column ] = 1;      //此列被占
      b[ line - column + 7 ] = 1; //主对角线被占
      c[ line + column ] = 1;    //从对角线被占
      eightQueens( line + 1 ); //下一个皇后
      //重置
      data[ line ][ column ] = 0;
      a[ column ] = 0;
      b[ line - column + 7 ] = 0;
      c[ line + column ] = 0;
    }
  }
}
```

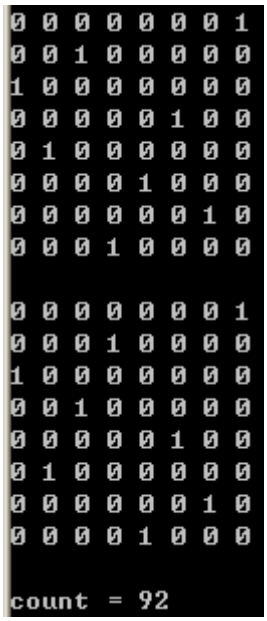
```
//output chess
void output( const int data[][ 8 ], int size )
{
  for( int i = 0; i < size; ++i )
  {
    for( int j = 0; j < size; ++j )
      cout << data[ i ][ j ] << ' ';
    cout << endl;
  }
  ++count;
}
```

5、性能：

时间复杂度 O(n^2)

6、测试

环境：VC++6.0



7、后记：

此算法是我在《程序员面试宝典》第 8 章面试例题 2 的基础上，做了一定的修改。

此外，我还想做进一步修改：把 eightQueens 函数做成一个封装的函数，

eightQueens(int a[][8], int n).