**第五次实验报告**

**一.八皇后问题**

1.问题描述：八皇后问题是以国际象棋为背景的问题：有八个皇后（可以当成八个棋子），如何在 8\*8 的棋盘中放置八个皇后，使得任意两个皇后都不在同一条横线、纵线或者斜线上。

2.算法分析：从棋盘的第一行开始，从第一个位置开始，依次判断当前位置是否能够放置皇后，判断的依据为：同该行之前的所有行中皇后的所在位置进行比较，如果在同一列，或者在同一条斜线上（斜线有两条，为正方形的两个对角线），都不符合要求，继续检验后序的位置。

如果该行所有位置都不符合要求，则回溯到前一行，改变皇后的位置，继续试探。

如果试探到最后一行，所有皇后摆放完毕，则直接打印出 8\*8 的棋盘。最后一定要记得将棋盘恢复原样，避免影响下一次摆放。

**源代码：**

#include <stdio.h>

#include<stdlib.h>

int Queenes[8] = { 0 }, Counts = 0;

int Check(int line, int list){

//遍历该行之前的所有行

for (int index = 0; index<line; index++) {

//挨个取出前面行中皇后所在位置的列坐标

int data = Queenes[index];

//如果在同一列，该位置不能放

if (list == data) {

return 0;

}

//如果当前位置的斜上方有皇后，在一条斜线上，也不行

if ((index + data) == (line + list)) {

return 0;

}

//如果当前位置的斜下方有皇后，在一条斜线上，也不行

if ((index - data) == (line - list)) {

return 0;

}

}

//如果以上情况都不是，当前位置就可以放皇后

return 1;

}

//输出语句

void print()

{

for (int line = 0; line < 8; line++)

{

int list;

for (list = 0; list < Queenes[line]; list++)

printf("0");

printf("#");

for (list = Queenes[line] + 1; list < 8; list++){

printf("0");

}

printf("\n");

}

printf("================\n");

}

void eight\_queen(int line){

//在数组中为0-7列

for (int list = 0; list<8; list++) {

//对于固定的行列，检查是否和之前的皇后位置冲突

if (Check(line, list)) {

//不冲突，以行为下标的数组位置记录列数

Queenes[line] = list;

//如果最后一样也不冲突，证明为一个正确的摆法

if (line == 7) {

//统计摆法的Counts加1

Counts++;

//输出这个摆法

print();

//每次成功，都要将数组重归为0

Queenes[line] = 0;

return;

}

//继续判断下一样皇后的摆法，递归

eight\_queen(line + 1);

//不管成功失败，该位置都要重新归0，以便重复使用。

Queenes[line] = 0;

}

}

}

int main() {

//调用回溯函数，参数0表示从棋盘的第一行开始判断

eight\_queen(0);

printf("摆放的方式有%d种", Counts);

system("pause");

return 0;

}

### 结果：八皇后摆放方式有92种，这里也不再一一列举

**二.象棋中马遍历棋盘的问题**

1.问题描述：在n\*m的棋盘中，马只能走“日”字。马从位置(x,y)处出发，把棋盘的每一格都走一次，且只走一次。找出所有路径。

2.问题分析：马是在棋盘的点上行走的，所以这里的棋盘是指行有n条边，列有m条边。而一个马在不出边界的情况下有8个方向可以行走。如当前坐标为(x,y)则行走后的坐标可以为：

（x+1,y+2）,(x+1,y-2),(x+2,y+1),(x+2,y-1),

(x-1,y-2),(x-1,y+2),(x-2,y-1),(x-2,y+1)

3.算法设计：搜索空间是整个n\*m个棋盘上的点。约束条件是不出边界且每个点只经过一次，结点的扩展规则如问题分析中所述。

搜索过程是从任一点(x,y)出发，按深度优先的规则，从八个方向中尝试一个可以走的点，直到走过齐攀升所有n\*m个点。

**源代码：**

#include <stdio.h>

#include<stdlib.h>

int n = 5, m = 4;

int fx[8] = { 1, 2, 2, 1, -1, -2, -2, -1 };

int fy[8] = { 2, 1, -1, -2, -2, -1, 1, 2 }, a[5][4];

int dep, x, y, count;

void output()

{

count++;

printf("\ncount=%d", count);

for (y = 0; y<n; y++)

{

printf("\n");

for (x = 0; x<m; x++)

printf("%3d ", a[y][x]);

}

printf("\n");

}

void find(int x, int y, int dep)

{

int i, xx, yy;

for (i = 0; i<8; i++) //加上方向增量,形成新的坐标

{

xx = x + fx[i];

yy = y + fy[i];

if (check(xx, yy) == 1) //判断新坐标是否出界,是否已走

{

a[xx][yy] = dep; //走向新的坐标

if (dep == n\*m)

output();

else

find(xx, yy, dep + 1);//从新坐标出发,递归下一层

a[xx][yy] = 0;

}

}

}

int check(int x, int y)

{

if (x >= 0 && x<n&&y >= 0 && y<m&&a[x][y] == 0)

return 1;

else

return 0;

}

void main()

{

int i, j;

count = 0;

dep = 1;

printf("请输入x y的坐标: ");

scanf("%d%d", &x, &y);

if (x >= n || y >= m || x<0 || y<0)

{

printf("输入的坐标有误，请重新输入:\n");

scanf("%d%d", &x, &y);

}

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

for (j = 0; j<m; j++)

a[i][j] = 0;

a[x][y] = 1;

find(x, y, 2);

if (count == 0)

printf("No solution!\n");

else

printf("count=%d", count);

system("pause");

}

**结果示例：**

