**数据结构上机实习报告**

题目：2.4 马踏棋盘

班级：1403013 学号：14030130061 姓名：刘豪 完成日期：2015.12.04

**一、需求分析**

1.设计一个国际象棋的马踏遍棋盘的演示程序

2将数字1~64依次填入方格

3打印出指定数量的解

4求出解一共有多少个

**二、概要设计**

为了实现上述程序功能，我先写了一个栈的类，再写了一个棋盘类。

class Stack{

基本操作：

Stack()

初始化一个空对象;

~Stack()

析构一个对象;

void del()

删除栈顶元素;

void push(int \_val)

元素压栈;

int top()

返回栈顶元素;

void pop()

弹栈;

int size()

返回栈内元素个数;

}

class Board{

基本操作：

Board()

初始化一个空对象;

Board(int s,int cnt)

生成一个起始点为s的对象，求解数为cnt;

int solve()

dfs的栈展开;

void print()

打印结果;

}

**三、详细设计**

#include<iostream>

#include<cstdio>

#include<cstdlib>

#include<ctime>

using namespace std;

class Stack{

struct Node{

int val;

Node \*nex;

};

private:

Node \*head;

int \_sz;

public:

Stack();

~Stack();

void del();

void push(int \_val);

int top();

void pop();

int size();

};

void Stack::del(){

if(head==NULL)

return ;

\_sz--;

Node \*tmp=head;

head=head->nex;

delete tmp;

}

Stack::Stack(){

head=NULL;

\_sz=0;

}

Stack::~Stack(){

Node \*tmp;

while(head!=NULL){

(\*this).del();

}

}

int Stack::top(){

return head->val;

}

void Stack::pop(){

(\*this).del();

}

void Stack::push(int \_val){

\_sz++;

Node \*tmp=new Node;

tmp->val=\_val;

tmp->nex=head;

head=tmp;

}

int Stack::size(){

return \_sz;

}

class Board{

private:

static const int MAXN=8;

static const int MAXM=8;

int \_s,\_cnt;

int vis[MAXN][MAXM];

bool tag[MAXN][MAXM][8];

Stack Sta;

public:

Board();

Board(int s,int cnt);

int solve();

void print();

};

Board::Board(){

srand((unsigned)time(0));

for(int i=0;i<MAXN;i++)

for(int j=0;j<MAXM;j++){

vis[i][j]=-1;

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

tag[i][j][k]=false;

}

\_s=rand()%(MAXN\*MAXM-1);

\_cnt=3;

}

Board::Board(int s,int cnt){

for(int i=0;i<MAXN;i++)

for(int j=0;j<MAXM;j++){

vis[i][j]=-1;

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

tag[i][j][k]=false;

}

\_s=(s-1);

\_cnt=cnt;

}

int Board::solve(){

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

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

vis[\_s/8][\_s%8]=1;

Sta.push(\_s);

int icase=0;

while(Sta.size()!=0){

if(Sta.size()==64){

if(icase<\_cnt)

(\*this).print();

icase++;

}

int x=Sta.top()%8;

int y=Sta.top()/8;

int flag=-1;

int maxn=9;

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

int tx=x+dx[i];

int ty=y+dy[i];

if(tag[y][x][i]||vis[ty][tx]!=-1||(tx<0||tx>7||ty<0||ty>7))

continue;

int cnt=0;

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

int my=ty+dy[k];

int mx=tx+dx[k];

if(vis[my][mx]==-1&&(mx>=0&&mx<8&&my>=0&&my<8))

cnt++;

}

if(cnt<maxn){

maxn=cnt;

flag=i;

}

}

if(flag!=-1){

int ty=y+dy[flag];

int tx=x+dx[flag];

Sta.push(ty\*8+tx);

vis[ty][tx]=vis[y][x]+1;

tag[y][x][flag]=true;

}

else{

vis[y][x]=-1;

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

tag[y][x][i]=false;

Sta.pop();

}

}

return icase;

}

void Board::print(){

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

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

printf("%2d ",vis[i][j]);

}

printf("\n");

}

printf("\n");

}

int main(){

int n,cnt;

printf("请输入起始格子(范围1~64)和需要的解的个数：");

scanf("%d%d",&n,&cnt);

Board a(n,cnt);

printf("%d\n",a.solve());

return 0;

}**四、调试分析**

1.通过头插法和头结点删除法建立了一个栈。

2.通过贪心算法优化，可以在很快打印出解，但是计算出所有解的数量需要的时间很长还有进一步优化的空间。

**五、用户手册**

本程序的运行环境为WIN10，执行文件为:马踏棋盘.exe

**六、测试结果**

