#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include<windows.h>

#include<conio.h>

#ifndef \_List\_H

struct Node;

typedef struct Node \*PtrToNode;

typedef PtrToNode List;

typedef PtrToNode Position;

void PrintList(List L, int numebr);//打印链表

void RandomMove(List L, int number);//随机移动

void MoveBack(List L, int number);//将0移动回右下角

void StartGame(List L, int number);//游戏开始（输入链表和矩阵大小）

int RightMove(List L, int number);//向右移动

int LeftMove(List L, int number);//向左移动

int UpMove(List L, int number);//向上移动

int DownMove(List L, int number);//向下移动

List CreatList();//制作表头

List CreatArray(int number);//制作数组

Position FindZero(List L);//寻找0的位置

Position FindPosite(List L, int number);//寻找某个位置

#endif /\* \_List\_H \*/

struct Node

{

int Element;

Position Right;

Position Left;

int posite;

};

//制作包含头的链表

List CreatList()

{

PtrToNode F\_List;

F\_List = /\*(Node\*)\*/malloc(sizeof(struct Node));

if (F\_List == NULL)

{

printf("\nOut of place!!");

Sleep(1000);

exit(0);

}

else

{

F\_List->Element = 1;

F\_List->Left = NULL;

F\_List->Right = NULL;

F\_List->posite = 1;

return F\_List;

}

}

//制作数组

List CreatArray(int number)

{

int Tnumber = number \* number;

PtrToNode Tmp;

List L = CreatList(), Header;

Header = L;

//制作非0的部分

while (Tnumber - 2)

{

Tmp = /\*(Node\*)\*/malloc(sizeof(struct Node));

if (Tmp == NULL)

{

printf("\nOut of place!!");

Sleep(1000);

exit(0);

}

else

{

Tmp->Element = (number \* number - Tnumber + 2);

Tmp->posite = Tmp->Element;

Tmp->Right = NULL;

L->Right = Tmp;

Tmp->Left = L;

L = Tmp;

}

Tnumber--;

}

//制作0的部分

Tmp = /\*(Node\*)\*/malloc(sizeof(struct Node));

if (Tmp == NULL)

{

printf("\nOut of place!!");

Sleep(1000);

exit(0);

}

Tmp->Element = 0;

Tmp->posite = number \* number;

Tmp->Right = NULL;

L->Right = Tmp;

Tmp->Left = L;

//返回表头

return Header;

}

//打印链表

void PrintList(List L, int number)

{

int TmpNumber = number;

Position Posite;

Posite = L;

if (L->Right == NULL)

{

printf("\nList is Empty!!");

Sleep(1000);

exit(0);

}

while (Posite != NULL)

{

if (TmpNumber == 0)

{

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

TmpNumber = number;

}

printf("%4d ", Posite->Element);

Posite = Posite->Right;

TmpNumber--;

}

printf("\n 游戏注意事项：\n1、在进行游戏之前，请先按大写锁定键Capslk\n2、在游戏操作过程中，有效的操作键为'W''A''S''D'键，分别表示对零下方，右方，上方，左方数字进行上移，左移，下移，右移操作\n3、当数字从左到右，从上到下分别为1至15和0时，游戏成功\n4、按下'Q'键退出\n");

}

//寻找0的位置，返回位置.

Position FindZero(List L)

{

Position Tmp, Posite;

Tmp = L;

Posite = L;

while (Tmp != NULL)

{

if (Tmp->Element == 0)

{

return Tmp;

break;

}

else

{

Posite = Tmp->Right;

Tmp = Posite;

}

}

}

//寻找给定的位置，返回位置.

Position FindPosite(List L, int number)

{

Position Tmp, Posite;

Tmp = L;

Posite = L;

while (Tmp != NULL)

{

if (Tmp->posite == number)

{

return Tmp;

break;

}

else

{

Posite = Tmp->Right;

Tmp = Posite;

}

}

}

//0向右移动(其他数字向左)

int LeftMove(List L, int number)

{

Position P;

int MID, have = 0;

P = FindZero(L);

if (P->posite % number != 0 && P->Right != NULL)

{

MID = P->Element;

P->Element = P->Right->Element;

P->Right->Element = MID;

have = 1;

}

return have;

}

//0向左移动(其他数字向右)

int RightMove(List L, int number)

{

Position P;

int MID, have = 0;

P = FindZero(L);

if (P->posite % number != 1 && P->Left != NULL)

{

MID = P->Element;

P->Element = P->Left->Element;

P->Left->Element = MID;

have = 1;

}

return have;

}

//0向下移动(其他数字向上)

int UpMove(List L, int number)

{

Position P1, P2;

int MID, have = 0;

P1 = FindZero(L);

if (P1->posite <= number \* (number - 1))

{

P2 = FindPosite(L, (P1->posite + number));

MID = P1->Element;

P1->Element = P2->Element;

P2->Element = MID;

have = 1;

}

return have;

}

//0向上移动(其他数字向下)

int DownMove(List L, int number)

{

Position P1, P2;

int MID, have = 0;

P1 = FindZero(L);

if (P1->posite >= number + 1)

{

P2 = FindPosite(L, (P1->posite - number));

MID = P1->Element;

P1->Element = P2->Element;

P2->Element = MID;

have = 1;

}

return have;

}

//将0移动到右下角

void MoveBack(List L, int number)

{

int have = 0;

Position Posite;

Posite = FindZero(L);

if (Posite->posite == number \* number)

have = 1;

else(!have);

{

while (Posite->posite % number != 0)

{

LeftMove(L, number);

Posite = Posite->Right;

}

while (Posite->posite != number \* number)

{

UpMove(L, number);

Posite = FindPosite(L, Posite->posite + number);

}

}

}

//随机移动

void RandomMove(List L, int number, int Line)

{

srand((int)time(NULL));

int point = 1, RDnumber;

for (; point <= number; )

{

RDnumber = rand() % 4;

switch (RDnumber)

{

case 0:

point += UpMove(L, Line);

break;

case 1:

point += DownMove(L, Line);

break;

case 2:

point += LeftMove(L, Line);

break;

case 3:

point += RightMove(L, Line);

break;

}

}

}

//游戏开始

void StartGame(List L, int number)

{

char PutIn = 'W';

int win = 0;

List Game, Tmp;

Game = CreatArray(number);

RandomMove(Game, number \* number \* number \* 3, number);

MoveBack(Game, number);

PrintList(Game, number);

Tmp = Game;

//初始化过程

while (!win)

{

win = 1;

if (\_kbhit())

{

PutIn = \_getch();

}

switch (PutIn)

{

case 'W':

UpMove(Game, number);

system("cls");

PrintList(Game, number);

break;

case 'S':

DownMove(Game, number);

system("cls");

PrintList(Game, number);

break;

case 'A':

LeftMove(Game, number);

system("cls");

PrintList(Game, number);

break;

case 'D':

RightMove(Game, number);

system("cls");

PrintList(Game, number);

break;

case'Q':

exit(0);

}

PutIn = '\0';

while (Tmp->Right != NULL)

{

if (Tmp->Element != Tmp->posite)

win = 0;

Tmp = Tmp->Right;

}

Tmp = Game;

}

printf("\nYou win this game!");

}

void main()

{

int k;

double duration;

clock\_t start, finish;

List list, position;

printf("请输入3到10之间的一个数字（该数字代表华容道阶数）：\n悄悄告诉你，如果你不听话的话，会受到惩罚哦，乖\n");

scanf\_s("%d", &k);

int point = 0;

while (k < 3 || k>10)

{

if(point < 3)

printf("Are you OK?");

else

{

printf("\nListen to me buddy!!!");

Sleep(1000);

exit(0);

}

point++;

scanf\_s("%d", &k);

}

list = CreatArray(k);

//PrintList( list , 4 ) ;

start = clock();

StartGame(list, k);

finish = clock();

duration = (double)(finish - start) / CLOCKS\_PER\_SEC;

printf("\n总用时： %f seconds\n", duration);

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

//PrintList(list, 4);

system("pause");

return 0;

}