The report of Digital Huarong Road

***Abstract***

In the second semester of our freshman year, we choose the project—Digital Huarong Road to practice and learn according to the requirements of C programming course. It takes us about two months to finish the project. In this process, we are active in practicing knowledge that our teachers have taught us, which we are deeply grateful for. To be honest, it is still challenging for us to do such a C programming project since we’ve never done such thing before. To complete this project, we take the initiative to learn extracurricular knowledge, to find a variety of books to read, to appreciate the code of predecessors, to gain inspiration and develop programming thinking and skills. Eventually, we make it.

***Problem Statement***

Since Digital Huarong Road is a game, a natural problem is how to generate the interface. Exactly speaking, it is how to generate a matrix where numbers are placed out of order. In addition, how can we implement keyboard to move numbers, how can we realize offering help to players, which means how can we search the right path? For this part, what algorithms can we use? Which is more appropriate? And how can we implement recording time? These are also problems we face. In fact, there are many other details in this game, such as the choice of difficulty, use which button to control which direction, overall game framework design and so on.

***Analysis***

※How to generate a matrix where numbers are placed out of order?

We had two choices at the beginning, one is to directly generate random Numbers and output them in the form of matrix, but this situation faces several problems. First, the matrix generated in this way cannot be guaranteed to have a solution after moving, so in this case, we need to make a function to determine whether there is a solution. The other case is that you take the matrix of the correct number order and then you scramble it to make a chaotic matrix, which is obviously a more reasonable case and some algorithm implementations are easier to understand. First, we generate a matrix where numbers are placed in order. Then we mess up it. As a result, we get the matrix where numbers are placed out of order. So now comes a question—how can we generate a matrix where numbers are placed in order?

We use a double linked list to generate a two-dimensional array—a matrix.

So now we need to do is to mess up it? How to do this?

It will be realized through the random movement of “0”.

So how to realize random movement of “0”?

Key words: ***Random number* *Switch structure Time***

More detailed explanation:

1. Different Random number will match with different direction.

2. Use Switch structure to judge and execute.

3. According time to gain different seeds from before to guarantee every time we will get a different matrix when we start the program

※How to implement keyboard to move numbers?

getch() function and kbhit() function.

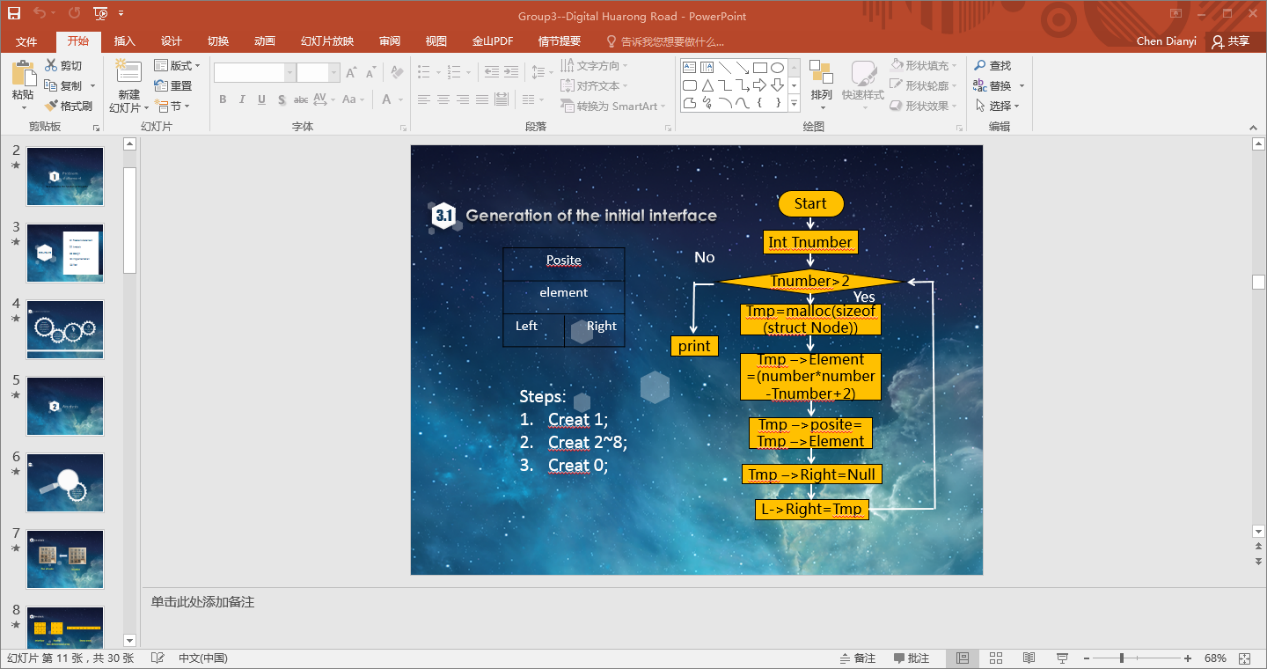
※How to realize offering help when players need it?

“DFS Algorithm.”

Wide search gives you the best solution, while deep search gives you many solutions. Deep search keeps searching in every direction, and wide search is the first step of prioritizing all directions. From the level of Huarong road, search from a direction will only get the same results, and no matter from which direction search can find the results.As long as the other steps are the same can be regarded as the optimal solution, and finally compare the time so it is not necessary to find the optimal solution, only we want to do is to search in all directions at the same time, according to the player needs it can be the most among the first to find its optimal solution, but players can also be a step in the search to choose corresponding child nodes to search, in fact the difference between the deep search and wide search for this game, from the perspective of the level of game Digital Huarong Road, from one direction on the search will only get the same results, and no matter from which direction search can be found on the results. Actual operation when this algorithm we can in the first search to find the optimal solution of step, also can need to search to find other solution, as long as it can make the game can be closer to the actual when using this algorithm, need not too much difference, because in the main body of every step to search for and can show all the search steps, we can only call it deep search for a moment

※How to realize recording time until the player wins the game?

clock() function.

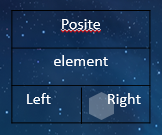
***Design***

※How to generate the matrix where numbers are placed in order?

Let’s come to see the flowchart on the right.

Maybe it make you a little puzzled. Take it easy. To make it

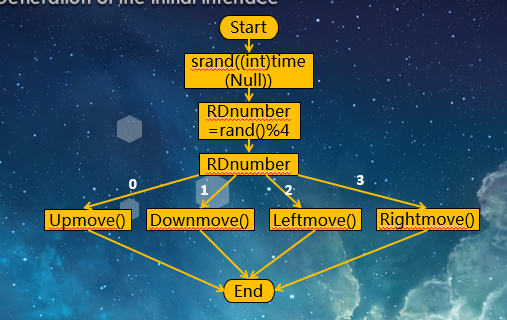
Clearer, let’s come to see what makes up the struct.

 The left picture shows the structure of the struct. It is the unit of the matrix, or to say, it is the unit of an array. The posite stores the address of itself for access. The element is a number the unit stores, such as “1”, “2” … The “Left” and the “Right” are respectively used to store the address of former node and the latter node for access. So we can get an array. Only if we let the numbers placed in order, we will get a matrix where numbers are placed in order. The key is in the mathematical expression in the flowchart. I believe you will be able to understand it after reading our code.

The struct

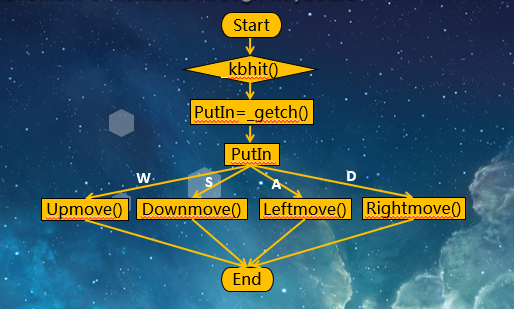
※How to mess up the matrix?

Let’s come to see the flowchart first.



Detailed explanation: We use srand() function to get random number. For a random number, no more than 4 possible results will be got through dividing it by 4. We call these four possible results RDnumber. They are 0,1,2,3. Every RDnumber will represent a direction as the flowchart shows. For example, “0” represents “Upmove”, which means move down “0”. Through several circles, we will get a matrix where numbers are placed at random.

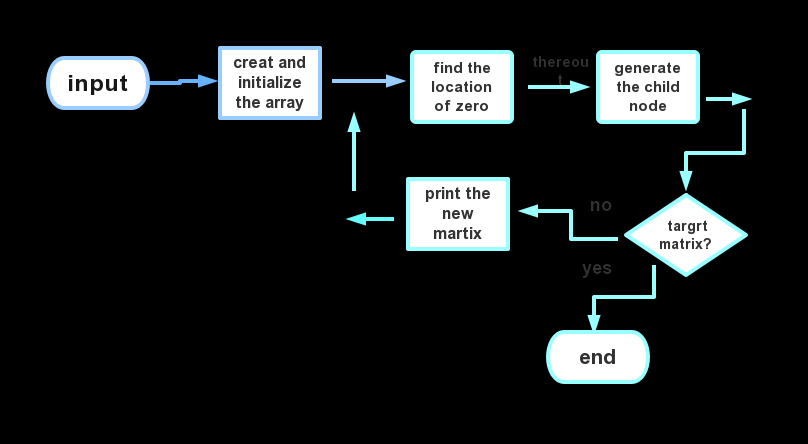
※How to implement keyboard to move numbers?



Detailed explanation: We use kbhit() function to “feel” whether the player is input. We use getch() function to read what the player input. We use ‘W’, ‘A’, ‘S’, ‘D’ to represent the upmove, downmove, leftmove and rightmove to the number that player can move.

※How to realize offering help?

When the player is unable to find a solution to a particular problem. Pressing “F” will help them print the right path. Depending on the player's needs, the program can search the corresponding steps and print out the sequence of operations when it finds a path. Let's look at the flowchart for details.



First, the player implements the Numbers to input the chaotic matrix, and the program generates the corresponding matrix accordingly. Next, the program begins to find the zero position to generate the corresponding child nodes. For example, if zero is in the upper left corner, this position generates two child nodes, one where the zero is swapped with the lower number, and one where the zero is swapped with the right number. The program starts to determine if these two child points are the target matrix, and if so, it prints out the order of the player's actions, but it still shows other child nodes and the player can continue to search for other child nodes. If not, repeat the process. Of course, the search process will determine whether the search is terminated based on the number of search steps entered by the player

***Implementation***

**Define a struct type(Structure 1)：**

struct Node

{

int Element;

Position Right;

Position Left;

int position;

};

1. Creatematrix：

List Creatematrix(int number)

{

int t = 2;

PtrToNode Tmp;

List L = CreatList(), Header;

Header = L;

while (t <= number\*number)

{

Tmp = malloc(sizeof(struct Node));//内存

if (Tmp == NULL)

{

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

Sleep(1000);

exit(0);

}

else

{

if (t < number\*number)

{

Tmp->Element = t;

Tmp->position = Tmp->Element;

Tmp->Right = NULL;

L->Right = Tmp;

Tmp->Left = L;

L = Tmp;

}

else

{

Tmp->Element = 0;

Tmp->position = number\*number;

Tmp->Right = NULL;

L->Right = Tmp;

Tmp->Left = L;

}

}

t++;

}

return Header;

}

1. FindZero：

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;

}

}

}

1. Movement

int Left(List L, int number)

{

Position P;

int MID, have = 0;

P = FindZero(L);

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

{

MID = P->Element;

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

P->Right->Element = MID;

have = 1;

}

return have;

}

1. Structure in DFS algorithm：

struct numberslider\_struct\_1

{

int NodeID;

int TreeDepth;

Enum\_Operate lastOperation;

short Array[3][3];

Type\_NumberNode \* preNode;

Type\_NumberNode \* nextNode[4];

};

1. Print operation path：

void printnode(Type\_NumberNode \* Node)

{

switch (Node->lastOperation)

{

case 0:

printf("S(down)<-");

break;

case 1:

printf("W(up)<-");

break;

case 2:

printf("D(right)<-");

break;

case 3:

printf("A(left)<-");

break;

default:

printf("NONE.<-");

break;

}

}

1. Createnode ：

int Createnode(Type\_NumberNode \* node, int Position\_X, int Position\_Y, Enum\_Operate Operate)//初始化节点，初始化结构体，接收0的位置

{

int i, j;

PUBLIC\_NumberNodeID++;//每一次全局变量ID加1，记录ID

node->NodeID = PUBLIC\_NumberNodeID;//该子节点的ID

node->TreeDepth = node->preNode->TreeDepth + 1;//搜索深度为父节点加1

node->lastOperation = Operate;//记录上一次的操作

for (i = 0;i<3;i++)

{

for (j = 0;j<3;j++)

node->Array[i][j] = node->preNode->Array[i][j];//矩阵与父节点相同

node->nextNode[i] = NULL;//下一个子节点为空

}

if (IfFindgoal(node))

{

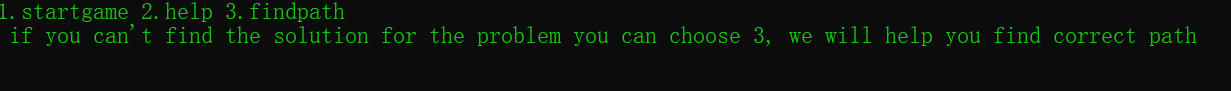
printf("\n\n The node is the target node and I will show you the formation process of the node in reverse order.\n");

printpath(node);

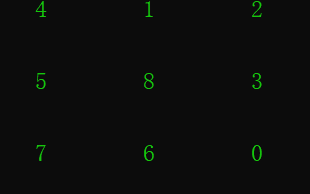
PUBLIC\_IsFinDCorrect = 1;

}

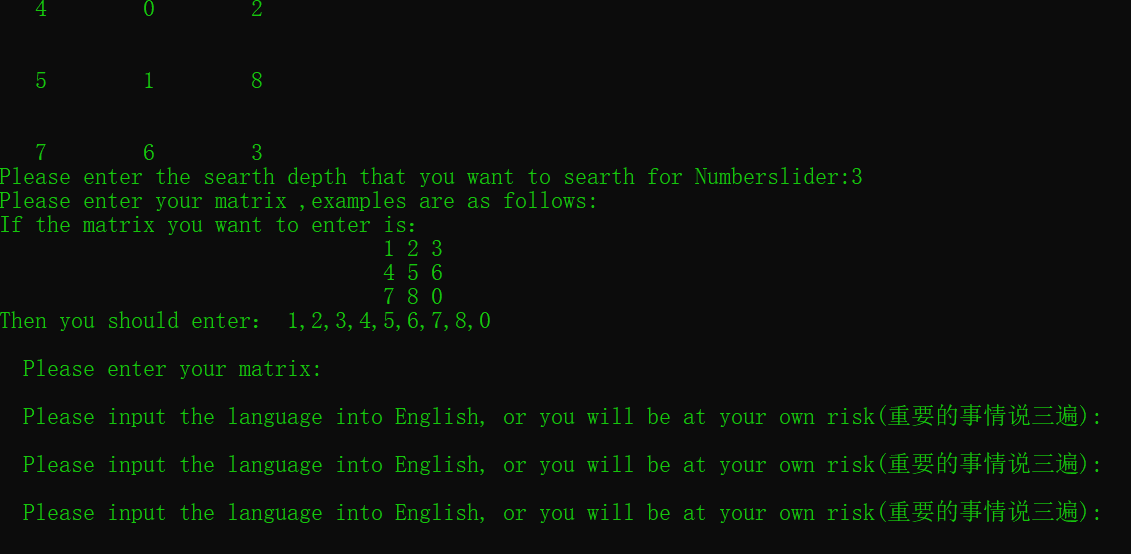
***Test***

******

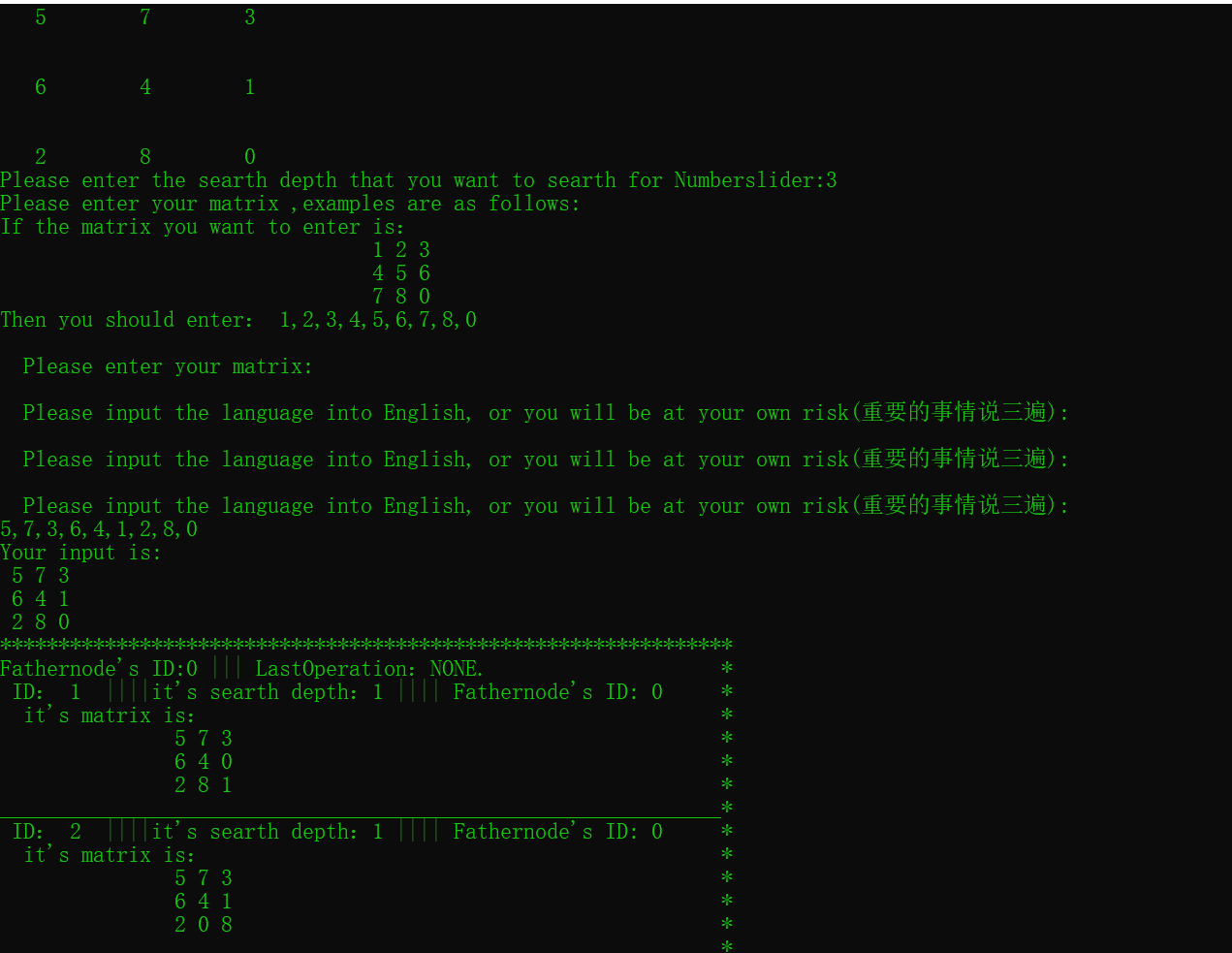
1.Initial interface:



2.DFS algorithm:



Search result:



***Group Member and Work Division***

|  |  |  |  |
| --- | --- | --- | --- |
| Group Member | College | Work Proportion | Work Description |
| \*\*\* | \*\*\* | 1/3 | Code writing(except DFS);  Project deployment  writing;  Slide modification |
| \*\*\* | \*\*\* | 1/3 | Code writing(DFS);  Code integration;  Slide modification;  Project report writing |
| \*\*\*i | \*\*\* | 1/3 | Code writing(except DFS);  Slide making;  Project report writing |

**Notes:**

The member list is no particular order.

**[Reference]**

Past TAQ C project; all kinds of blogs; all kinds of books

***Acknowledgement***

Thank our teachers for their precious guidance.

Thank senior schoolmates for their patient explanation.

Thanks to this era for providing us with learning resources for reference.