The report of digital huarong road

## Abstract

We have used one semester to finish our project named digital huarong road. It is an old riddle in China and we want to design a method to realize the function of the game.

In addition to the basic function of the game. We also add some other functions in our programming. For example, the time recording and the choice of the difficulty is added to our project, which is a creative point of our project.

Our problems mainly lies in the generation of the initial interface and the movement of players. In the following part of the report, you will see how we solve the problems with our intelligence.

1. **Introduction/Problem Statement**

In our project, we want to realize the function of digital huarong road, which is a famous riddle with a long history in China. In our version, we first generate the initial interface(Picture 1 is an example of 4\*4 matrix),and then let the players move the numbers according to the rule. The rule is that the players could only move the numbers into the blank. If the player arranges the numbers in the correct order (Picture 2 is an example of 4\*4 matrix), he wins the game.

 Picture 1 Picture 2

The concrete functions we want to realize in our project are as follows:

1. the basic functions of the game digital huarong road(the functions have been introduced above);
2. the choice of difficulty or the order of the matrix by typing in different number;
3. the function of recording the time the players have used;

Compared to other similar games, our advantages mainly lies in the choice of difficulty and the time recording function.

And the following part will illustrate the main problems we have and the solution to each problem.

Problem 1:

What we can do to generate the initial interface with disordered numbers?

Solution:

We first generate the final matrix with numbers in order,and then we use the Random function to generate the random number and then use a switch structure to dispute the ‘up’ ’down’ ’left’ and ’right’ situation randomly to solve the problem. The flow chart of this part can be seen in the design part.

Problem 2:

What is needed to let the players move the numbers with keyboard?

Solution:

We use the function \_kbhit() \_getch() to realize the function.

Problem 3:

What can we do to realize the function of difficulty choice?

Solution:

To solve the problem, we use the list and an int value to control the difficulty

choice part. And the arrange of the number the players type in is limited.

All above are the introduction and the problem statement part of our project.

## **Group Division**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | student ID | College | Percentage of working |
| A | \*\*\*\* | Mathematics | 35% |
| B | \*\*\*\* | Mathematics | 25% |
| C | \*\*\*\* | Mathematics | 20% |
| D | \*\*\*\* | Mathematics | 20% |

## 3. Analysis

In this part,we will illustrate the functions we want to realize in detail. There are five main part in our project,and they are: the generation of the array,the random movement of the number,moving 0 to the right position,movement of players and the time recording and the choice of difficulty. And we will show the function according to the order.

the generation of the array

In this part , we are going to finish an array with the number 0-15 on the right place.

Function 1: Create a “1” on the first place.

Analysis: We just need to let the posite equals to 1,and the Element equals to 1, too. Now we got a “1” on the first place.

Function 2: Create the number 2-15.

Analysis: We firstly found L in the F\_list. We let T\_number equals to 16. In fact it’s 4\*4. Then we let the Element equals to 16+2-T\_number. After that, we let “L->Right=Tmp”, “Tmp->Right=Null”. And then “Tmp->Left=L”, “L=Tmp”. We let T\_number minus 1 each time. We did it over and over again, until T\_number equals to 2. Then we got the number 2-15 on the right place.

Function 3: Create the number 0 on the last place.

Analysis: In fact it’s quite easy. Let the posite equals to 16, and the Element equals to 0. Then we got a 0 on its place. It’s quite similar to function 1.

the random movement of the number

In this part, we want to move the numbers randomly, which means we can get an interface where the numbers are messed up.

Function1: get different random numbers

Analysis: It is easy to solve this problem, we get a function \_time() which can provide random numbers. Then we need to make sure the numbers we get each time are different. To make it come true, we use the function \_srand().

Function2: guarantee the random numbers we get can decide the direction of movement

Analysis: It obvious that there are four directions of movement——up, down, left or right. So we set a new variable called RD number .The RD number is the random number dividing 4.So there are four cases. After that, we use the switch structure. In that case, if the RD number is 2, then the 0 will move left. Similarly, we can make sure the random numbers can decide the movement. Apart from that, we use the for circulation to let the 0 can move for many times.

moving 0 to the right position

In this part, we want to move 0 to the lower right corner

Function1: find where the 0 is

Analysis: After the random moving, we don’t know where the 0 is. So we need to move it to the lower right corner . The first step to do it is to find where the 0 is. Then we use the for circulation and if structure to do that. If the number we are checking out is not 0, then checkout the next one until we find where the 0 is. The for circulation is used to make sure we can checkout the numbers one by one until we find the 0.

Function 2:moving 0 to the lower right corner

Analysis:After we find the 0, we now need to move it to the right position.Then we need a program to move 0 to the right constantly until the position of it is a multiple of four. And now it is on the far right. Then we need to move it down constantly until the position of that is four times four, which means it is at the bottom right. On the other word, it is on the right position.

movement of players

In this part, we want to let the players move the number with the keyboard.

Function 1: get the character value from the keyboard and use it

Analysis: At first, we should have a function to judge whether the player is typing or not,and we found that \_kbhit() have the function. And then we use the function \_getch() to get the value from the keyboard when the player is typing.

Function 2: the move of number to different directions

Analysis: It is a bit easy compared to the random movement of the number. We just need to dispute different keyboard value to movement of different directions. So we need a switch structure to dispute the keyboard value and the relevant exchange of different number. We can also add a value to quit the game while playing.

Function 3: guarantee the players will keep playing until he win the game.

Analysis: we need a for circulation with a condition to judge whether the player has won the game or not. Thus we can that guarantee the players will keep playing until he win the game.

the time recording and the choice of difficulty

To satisfy players’ need for different difficulties of our game, we can have the players themselves choose the modes——which is the size of the matrix.

Normally we play the digital huarong road in a matrix whose size is from 3\*3 to 10\*10. It’s evident that the matrix has to be at least as big as 3\*3 to be created as our game, and we give the chance to challenge for more complex modes. A matrix smaller than 10\*10 can guarantee the initial interface is random enough for playing. (Though 6\*6 is enough.)

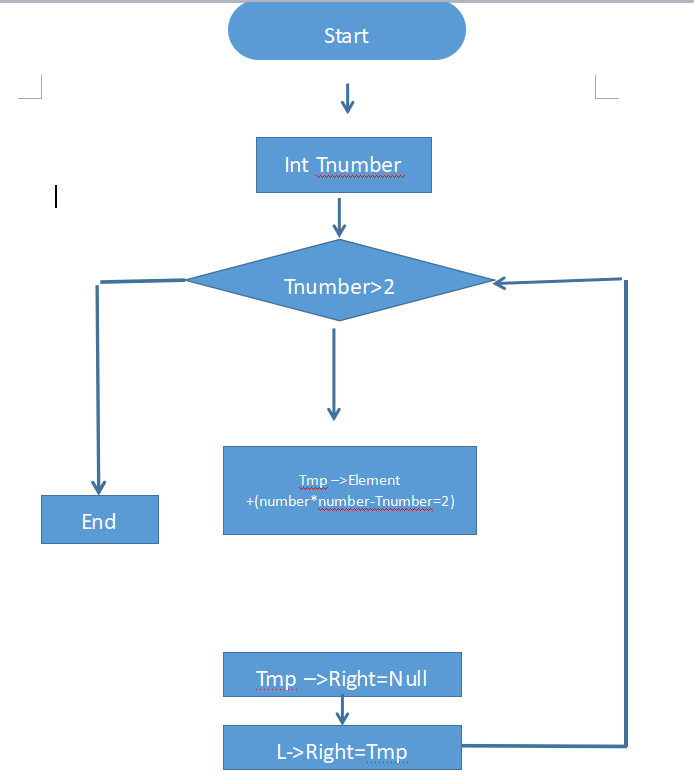
What’s more, we made a timer to record how much time it took for the player to solve the puzzle. However, in order not to disturb our players while they’re stuck at somewhere, we will show the time only after the game is finished.

## Design

In this part,we will illustrate the design we came up with during the project to realize the functions mentioned in the Analysis part or solve the problem we have in each part. In this part,we will also show what we came up with according to the same order, and that is:the generation of the array,the random movement of the number,moving 0 to the right position,movement of players and the time recording and the choice of difficulty. And we will show our design with the help of several flow charts.

the generation of the array

We firstly want to show the flow chart.

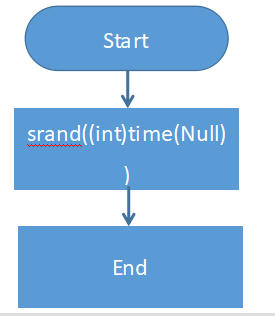


As you can see, to create the array, we divide the work into three part: create a 1; create the number 2-15; then create a 0. The above-mentioned flow part is about part 2. While creating the number 2-15, we’ d better making a circulation to make the thing easier. We also use the pointer during the process.

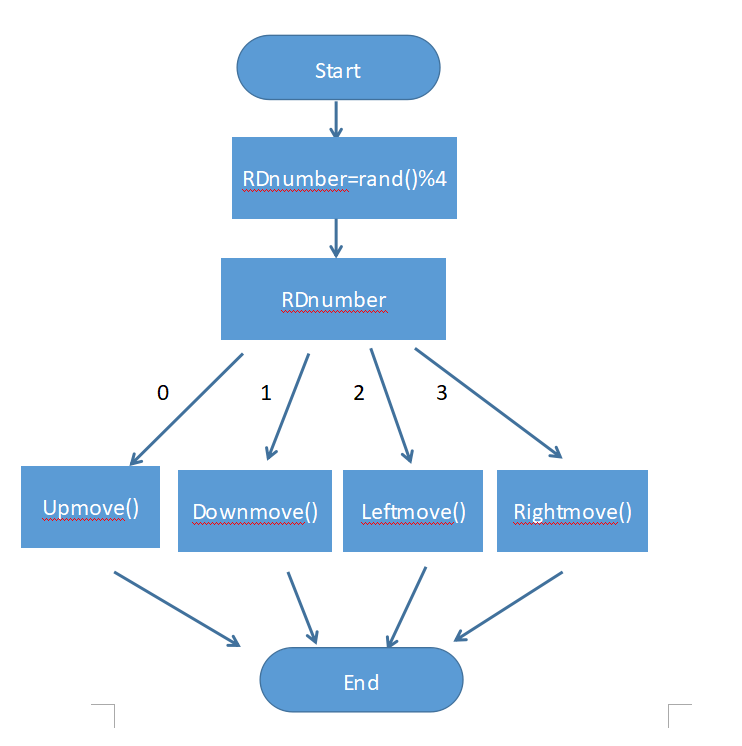
All the above are the design and the flow chart of the generation of the array.

the random movement of the number

At first, we want to show the flow chart of the function 1.

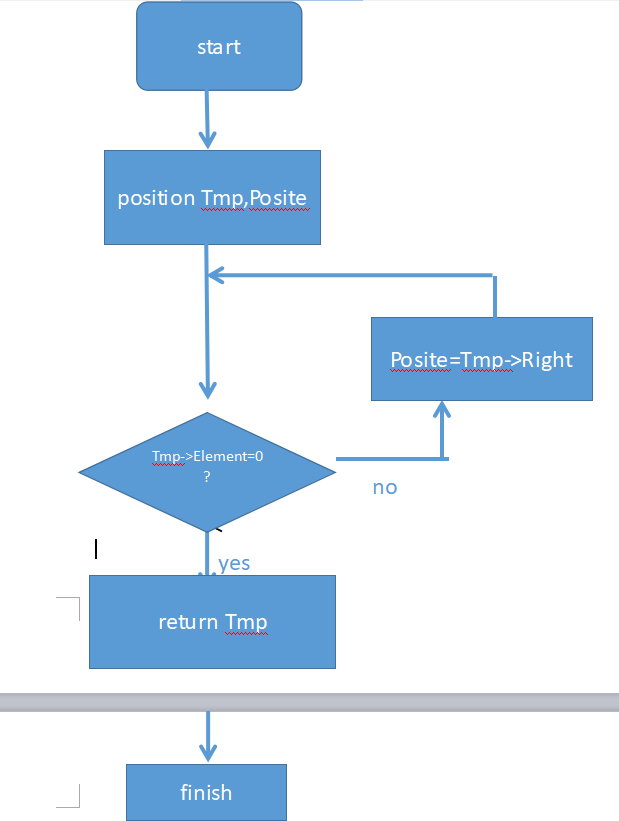


The flow chart above is the single operation of function 1. With the help of srand(), we can make sure the random number it produces each time is different.

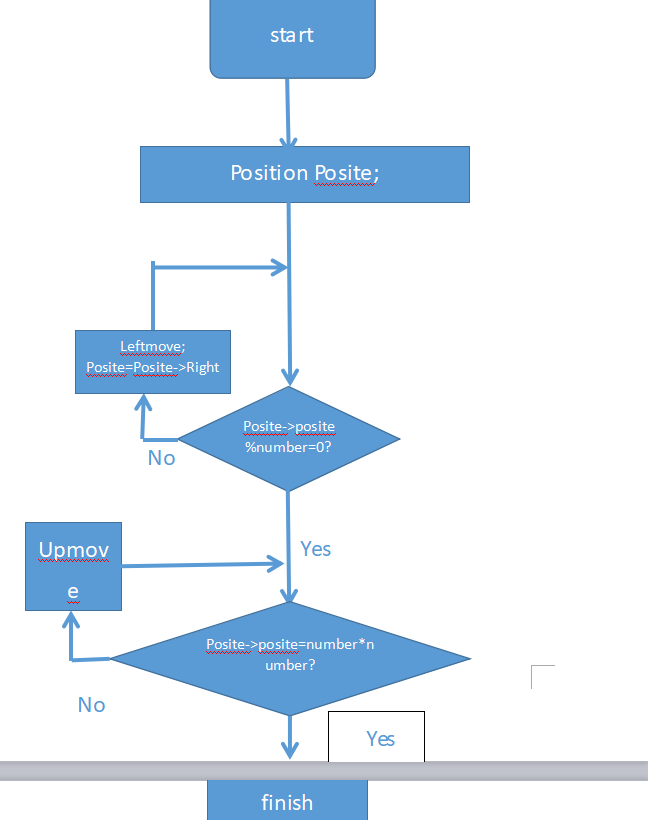
Then let’s show you the flow chart of the function 2. In this part,we want to guarantee the random numbers can decide the direction of movement. Let’s take the case of the number is equal to 4.

It is obvious that we have solve the problem. The RD number is the random number dividing 4. So there are 4 cases. The four cases can decide the direction of the movement. For example,if the RD is 1, the 0 will move down. Similarly, the direction can be up, left or right. If the 0 move for many times randomly, we will ensure that the array is scrambled enough.

the random movement of the number

Now we get the array which has been messed up. Then we need to find where the 0 is and move it to the bottom right corner. Let’s show you the flow chart.

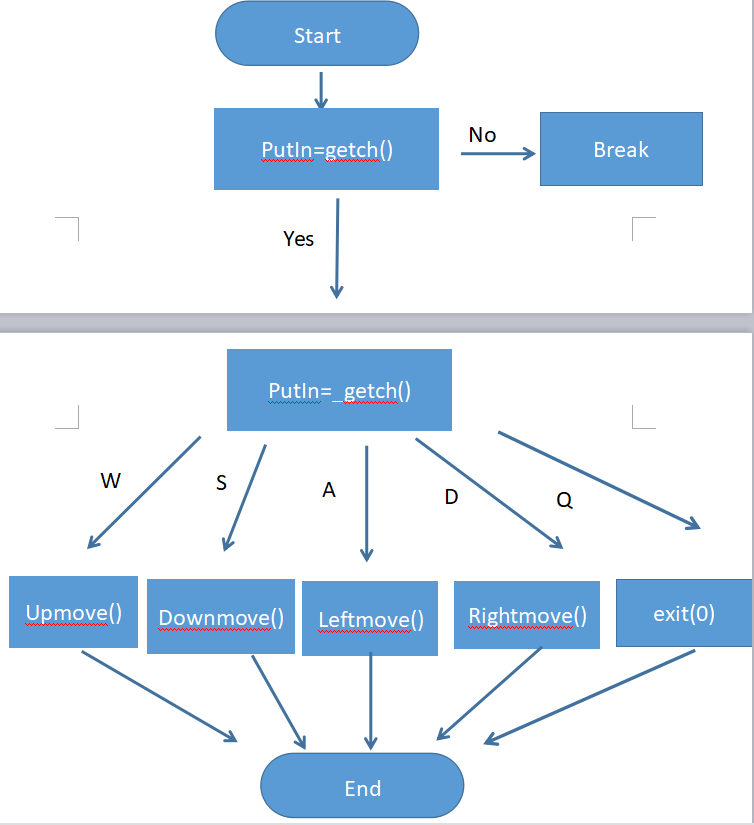
The flow chart is the single operation of the function 1. It will checkout each number until the 0 has been found.



The flow chart above is the single operation of function 2. The 0 will move to the far right firstly. Then it will move down constantly until the position of it is number times number. It means the 0 is in the right place.

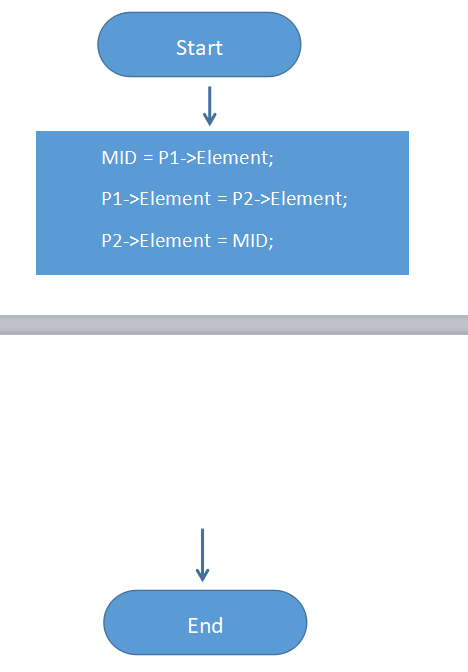
movement of players

At first, we want to show the flow chart of this part:



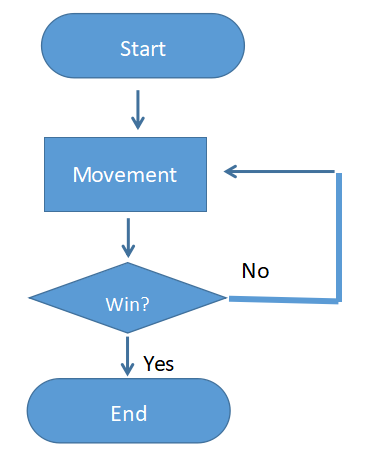
The flow chart shown above is the single operation of function 1 and function 2. As you can see, we used a switch structure to realize the assignment of the operation of directions and kbhit() and getch() to get the value from the keyboard.

The function of directions can be more detailed, and the idea of each function are almost the same. So we just take the Upmove() as an example. The following is the flow chart of Upmove():



In the function of directions, we all use the pointer to realize the exchange of relevant numbers.

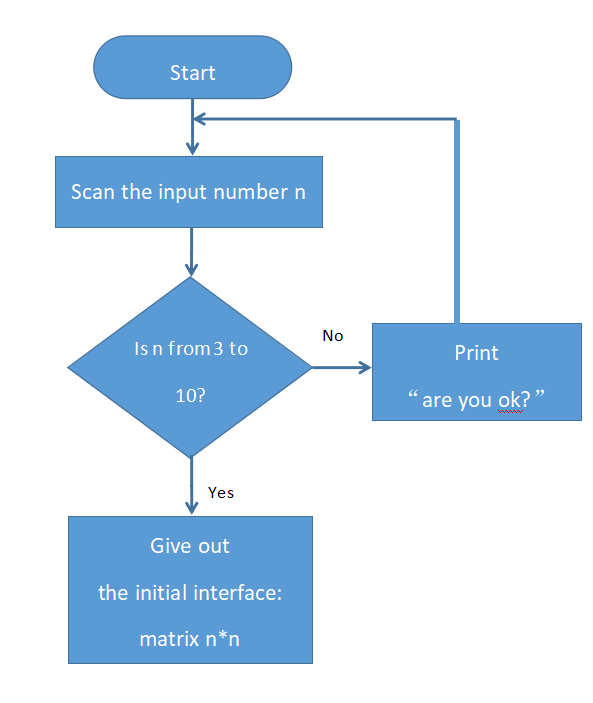
All the above are only once of the operation of players’, and that is unpractical, if we want to let the players play constantly, we must guarantee that the program won’t shut down until the player wins. And the following flow chart is the design of the circle(in this flow chat, the function of single movement is represented by Movement ):



All the above are the design and the flow chart of the movement of players.

the time recording and the choice of difficulty

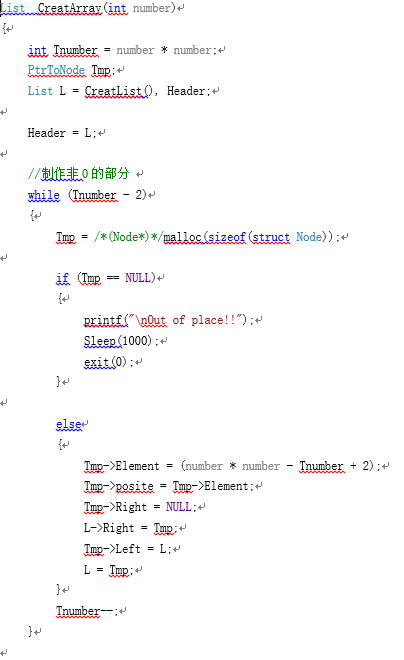
The following is the flow chart of the choice of the difficulty.

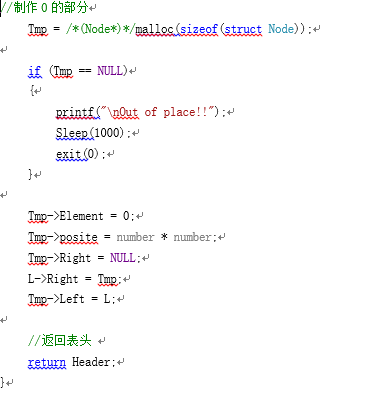


## 5.Implementation

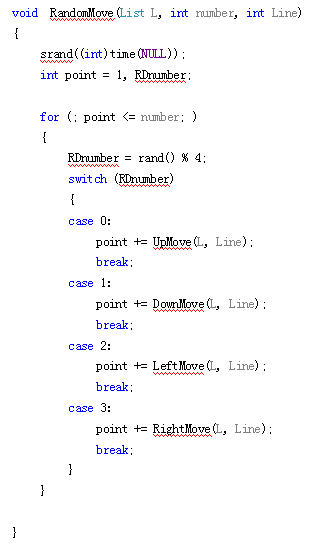
## In this part, we will show the key code of each part.

the generation of the array



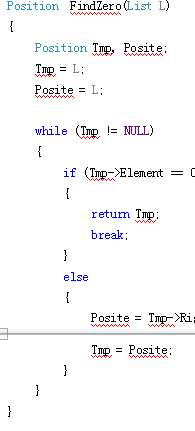


the random movement of the number

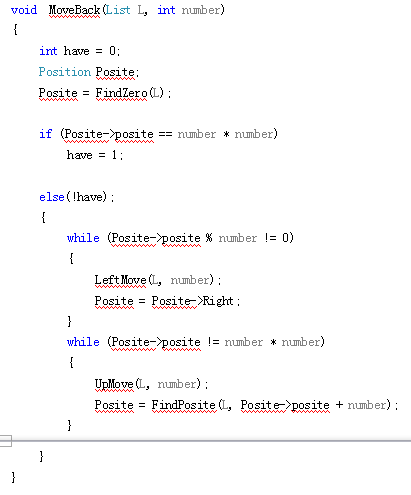


The code above is to move the numbers randomly

Moving 0 to the right position

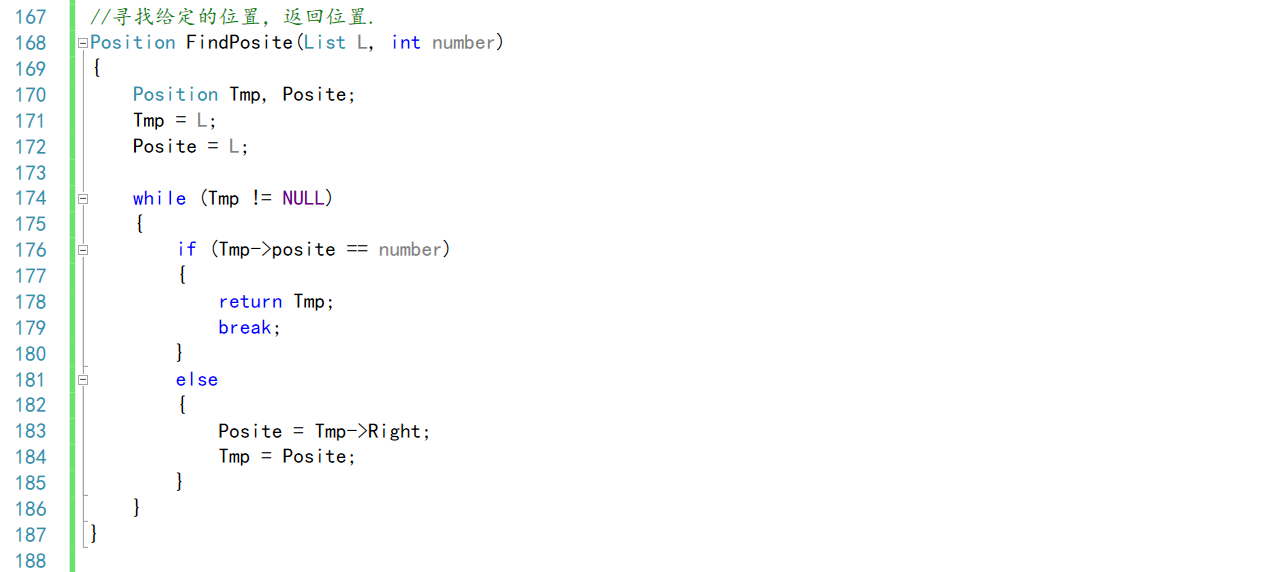
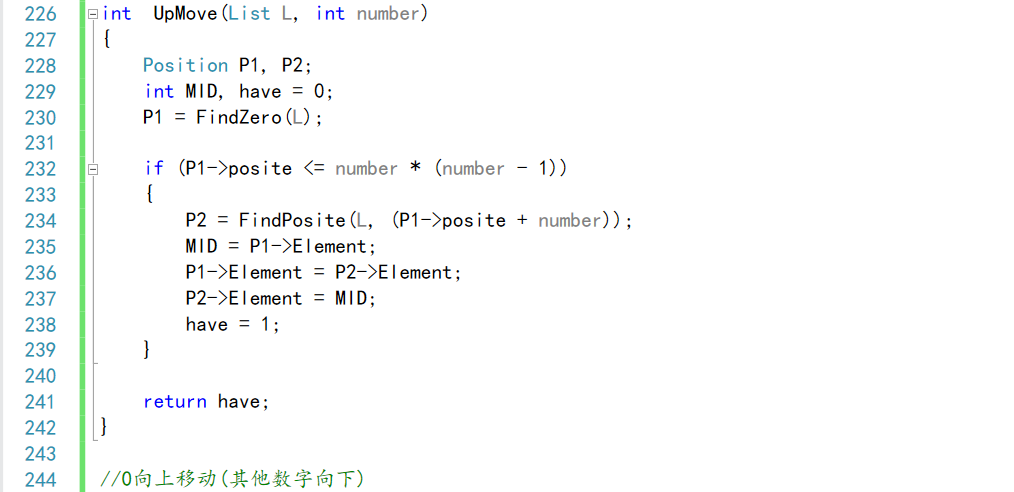


The code above is to find where the 0 is

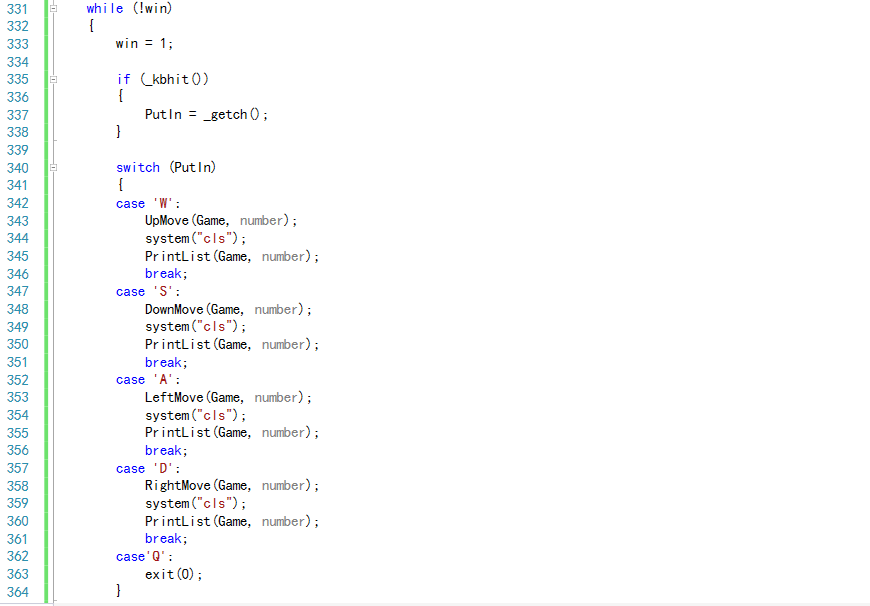


And the code above is to move the 0 to the right position

movement of players



The two pictures are the codes for an example of the function of directions.



The picture shows the code of whole process.

the time recording and the choice of difficulty

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\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);

## 6.Testing and Debugging

In this part, we will illustrate some part of the testing and debugging.

the random movement of the number

|  |  |  |
| --- | --- | --- |
| RD number | Expected result | Real result |
| 12 | Moving up | Moving up |
| 9 | Moving down | Moving down |
| 6 | Moving left | Moving left |

the random movement of the number

|  |  |  |
| --- | --- | --- |
| Posite----->posite | Expected result | Real result |
| 10 | Move left | Move left |
| 11 | Move left | Move left |
| 12 | End the first cycle | End the first cycle |

|  |  |  |
| --- | --- | --- |
| Posite | Expected result | Real result |
| 12 | Move down | Move down |
| 16 | End the second cycle | End the second cycle |

movement of players

The test of capitalizing

|  |  |  |
| --- | --- | --- |
| Input | Expected Result | Real Result |
| w | No effect | No effect |
| W | The relevant number move up | The relevant number move up |

The test of other characters

|  |  |  |
| --- | --- | --- |
| Input | Expected Result | Real Result |
| & | No effect | No effect |

The test of quitting the game

|  |  |  |
| --- | --- | --- |
| Input | Expected Result | Real Result |
| Q | Quit the game | Quit the game |

the time recording and the choice of difficulty

The test of mode choosing

|  |  |  |
| --- | --- | --- |
| Input | Expected Result | Real Result |
| 2 | Are you OK? | Are you OK? |
| 11 | Are you OK? | Are you OK? |
| 3.5 | Are you OK? | Are you OK? |
| x | Are you OK? | Are you OK? |
| 4 | A 4\*4 random matrix | A 4\*4 random matrix |

The test of timer

|  |  |  |
| --- | --- | --- |
| Input | Expected Result | Real Result |
| Finished the game  Took 4min 37sec | 277 seconds | 277 seconds |
| While unfinished | Nothing | Nothing |

**7.Result&Conclusion**

We eventually realized what we have wanted to achieve at the beginning of the semester. As is shown in the article, we realized the basic function of the riddle—digital huarong road, and we also add the time recording and the choice of difficulty function. Through the project, we learned how to solve a problem in reality, and also learned something about the cooperation.

The project is definitely imperfect. We faced problems such as the difficulty of gathering the team members and lack of sharp programming abilities. And we think we can solve the problems by being more responsible and keeping learning the knowledge of programming.

All in all, we all benefit a lot during the project.