CANAVARLI LABİRENT OYUNU

152120171029 İsmail Demircan

Eskisehir Osmangazi University

Faculty of Engineering and Architecure

Computer Engineering Departman

Computer Engineering LAB Lecture Project

Lecturer Yıldıray Anagun

May 2019

Part 1

Introduction

The program we wrote in this project is in a labyrinth defined in a 2-dimensional array (matrix) we have traced to find the way out before the monsters are caught. By the way finding the way out of the while collecting gold at the same time.If player meet the monster during the game , player have to beginning of the start point with 0 gold.

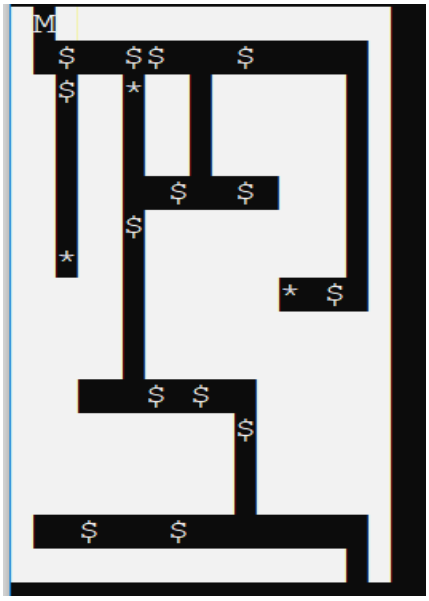
Problem

In this project I've dealt with loops in filing. Then I solved this problem using counters and I have create two arrays. Then I saved the motions to array elemans. After that I used loop and ı write in to the text.

What algorithm that you will apply the problem?

Create outer/Maze Wall and Space/ Gold / Monster

When preparing the map, I sketched the map by size, using loops in the form of spaces. I did this by pressing the spaces in the matrix according to the starting and ending points of the fights. Then I created a condition and pressed the gold into the spaces to random information .After that I put the monster on some locations. I showed the golds $, the monsters * symbol.



```
    for (i = 0; i < size; i++)
    {
        for (j = 0; j < size; j++)
        {
            A[1][j] = 1;
            A[0][1] = 0;
            A[size - 1][size - 2] = 0;
            A[size][size - 2] = 2;
            A[-1][1] = 1;
        }
    }


    for (i = 0; i < size; i++)
    {
        for (j = 1; j < size - 1; j++)
        {
            if (i == 1 || i == size - 2)
                A[i][j] = 0;
        }
    }
    for (i = 1; i < size / 3; i++)
        A[i][size / 2] = 0;

    for (j = size / 3; j < 3 * size / 4; j++)
        A[size / 3][j] = 0;
    for (i = size / 6; i < 2 * size / 3; i++)
        A[i][size / 3] = 0;
    A[size / 6][size / 3] = 3;
    for (j = size / 4; j < 2 * size / 3; j++)
        A[2 * size / 3][j - 1] = 0;
    for (i = 2 * size / 3; i < size - 2; i++)
        A[i][2 * size / 3 - 1] = 0;
    for (i = 1; i < size / 2; i++)
        A[i][size - 2] = 0;
    for (j = 3 * size / 4; j < size - 1; j++)
        A[size / 2][j] = 0;
    A[size / 2][3 * size / 4] = 3;
    for (i = 1; i < size / 2; i++)
        A[i][2] = 0;
    A[size / 2 - 1][2] = 3;

    int a;
    srand(time(NULL));
    for (i = 0; i < size; i++)
    {
        for (j = 0; j < size; j++)
        {
            a = rand() % 6 + 4;
            if (a == 5 && A[i][j] == 0)
            {
                A[i][j] = 5;
            }
        }
    }

    for (i = 0; i < size; i++)
    {
        for (j = 0; j < size; j++)
        {
            if (A[i][j] == 5)
                cout << "$";
            if (A[i][j] == 3)
                cout << "*";
            if (A[i][j] == 0)
                cout << " ";
            if (A[i][j] == 1)
                cout << char(219);
            if (A[i][j] == 2)
                cout << " ";
        }
        cout << endl;
```

Filing and Move Function

I have defined a coordinate plane to play the game. Then I created the conditions using the w, a, s, d keys. each key has allowed me to move on the coordinate. If I hit the wall, so if the value is equal to 1 x and y clean with the movement I have taken back, so I did not move.

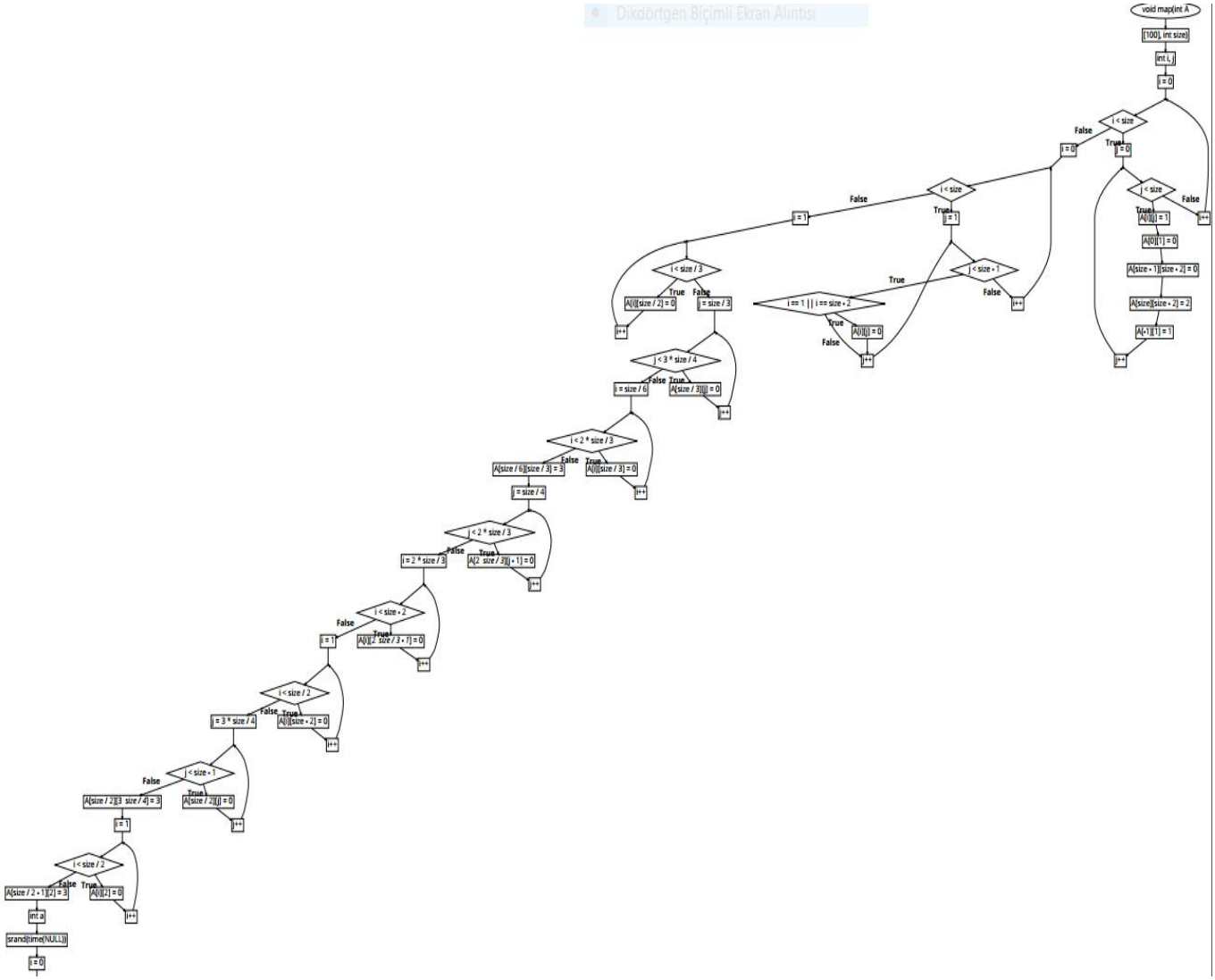
I counted the gold with a counter if it came under the cursor. If the cursor hit the monster, I reset the values and called the map function again. I used a counter for filing and counted the number of moves, and I created two arrays and saved each printed file in the file.

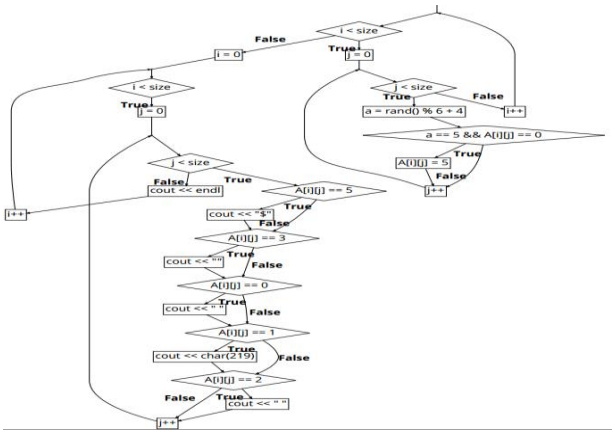
When the cursor came to the exit point, I printed the number of gold and the data that I saved to the array for filing.


```
x|y
0|1
1|1
1|2
1|3
1|4
1|5
1|6
1|7
2|7
3|7
4|7
5|7
5|6
5|5
5|4
6|4
7|4
8|4
9|4
10|4
11|4
11|5
11|6
11|7
11|8
11|9
12|9
13|9
14|9
15|9
15|10
15|11
15|12
15|13
15|15
16|15
```

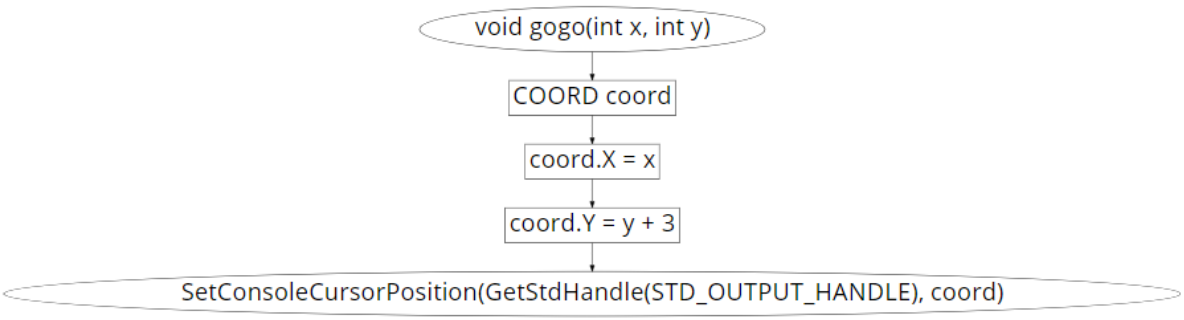
```
void gogo(int x, int y)
{
    COORD coord;
    coord.X = x;
    coord.Y = y + 3;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
}
void move(int A[][100], int size)
{
    int x = 1;
    int y = 0;
    int xcln = 1;
    int ycln = 0;
    int k = 0;
    int m1[250], m2[250], i = 0;

    int movecounter = 0;

    ofstream coordinat;
    coordinat.open("Coordinat.txt");
    coordinat << "x|y\n";

    while (true)
    {
        gogo(xcln, ycln);
        cout << " ";
        gogo(x, y);
        cout << 'M';
        start = _getch();
        xcln = x;
        ycln = y;


        if (start == 'w')
        {
            y -= 1;
            m1[i] = y;
            m2[i] = x - 1;
            i++;
            movecounter++;
        }
        else if (start == 's')
        {
            y += 1;
            m1[i] = y;
            m2[i] = x - 1;
            i++;
            movecounter++;
        }
        else if (start == 'a')
        {
            x -= 1;
            m2[i] = x - 1;
            m1[i] = y;
            i++;
            movecounter++;
        }
        else if (start == 'd')
        {
            x += 1;
            m2[i] = x - 1;
            m1[i] = y;
            i++;
            movecounter++;
        }

        m1[0] = 0;
        m2[0] = 1;

        if (A[y][x] == 5)
        {
            A[y][x] = 0;
            k++;
        }
        if (A[y][x] == 1)
        {
            x = xcln;
            y = ycln;
        }
        else if (A[y][x] == 3)
        {
            system("cls");
            x = 1;
            y = 0;
            k = 0;
            cout << "\n\n\n";
            map(A, size);
        }
        else if (A[y][x] == 2)
        {
            cout << endl << endl << "\nYou Win" << endl;
            cout << "Gold: " << k << endl;


            m2[movecounter - 3] = size - 2;
            m2[movecounter - 2] = size - 2;

            for (i = 0; i < movecounter - 1; i++)
            {
                for (int j = 0; j < 2; j++)
                {
                    if (j == 0)
                        coordinat << m1[i];
                    if (j == 0)
                        coordinat << "|";
                    if (j == 1)
                        coordinat << m2[i];
                }
                coordinat << "\n";
            }
            coordinat.close();
            break;
        }

}
```

Map Flowchart:

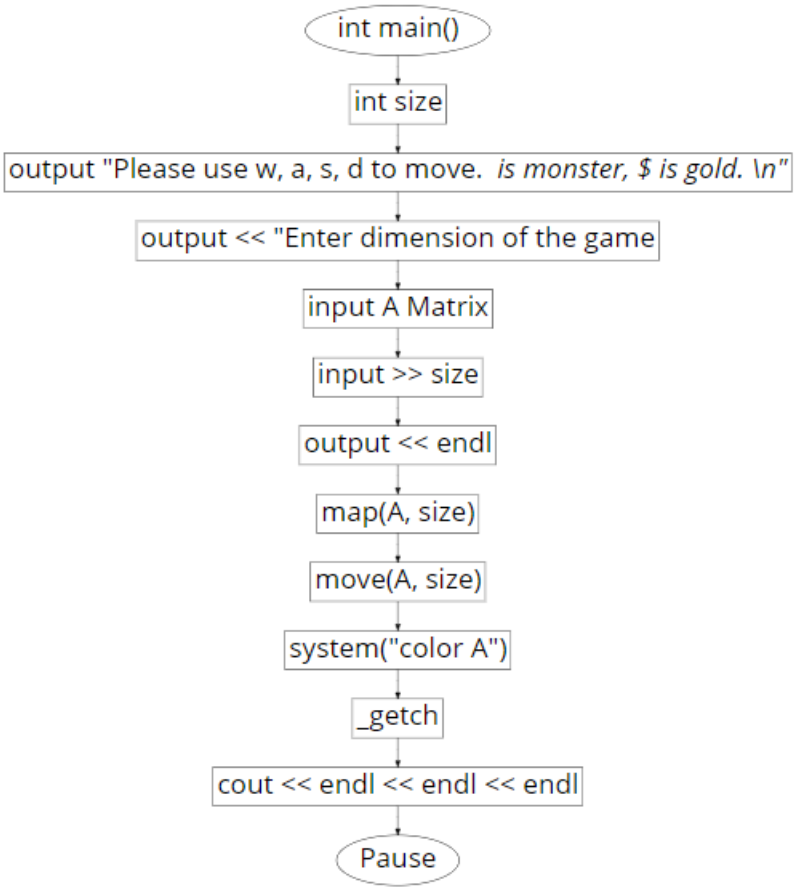




Coordinate Flowchart:



Main Flowchart:



Move Flowchart: