

REPORT

#Maze C#으로 재구성

과 목 명 : 윈도우어플리케이션개발

담당교수 : 이완주 교수님

학 과 : 컴퓨터 과학과

학 번 : 201633019

이 름 : 신 소 연

차례

1.사용한 클래스

2.클래스 관계도

3.소스코드

4.실행화면

#1. 사용한 클래스

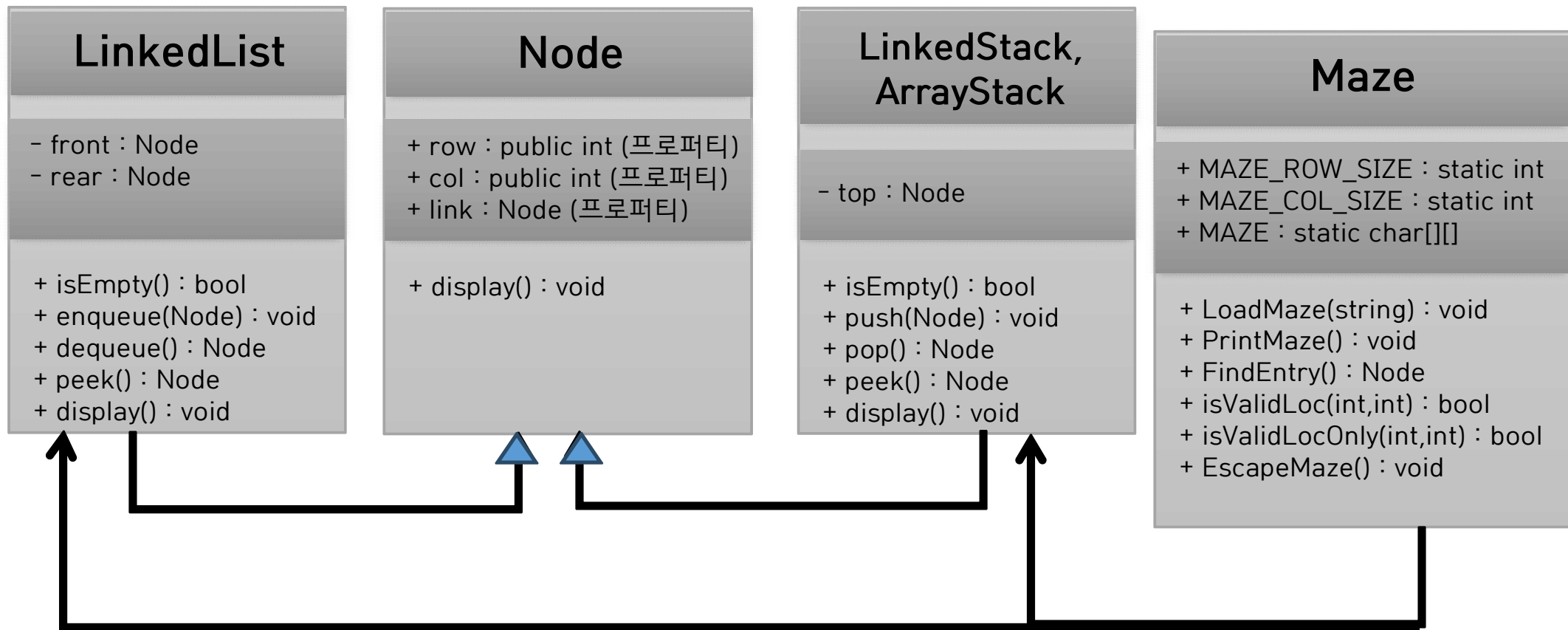
Maze	Node	LinkedList	LinkedStack, ArrayStack
<ul style="list-style-type: none"> + MAZE_ROW_SIZE : static int + MAZE_COL_SIZE : static int + MAZE : static char[][] 	<ul style="list-style-type: none"> + row : public int (프로퍼티) + col : public int (프로퍼티) + link : Node (프로퍼티) 	<ul style="list-style-type: none"> - front : Node - rear : Node 	<ul style="list-style-type: none"> - top : Node
<ul style="list-style-type: none"> + LoadMaze(string) : void + PrintMaze() : void + FindEntry() : Node + isValidLoc(int,int) : bool + isValidLocOnly(int,int) : bool + EscapeMaze() : void 	<ul style="list-style-type: none"> + display() : void 	<ul style="list-style-type: none"> + isEmpty() : bool + enqueue(Node) : void + dequeue() : Node + peek() : Node + display() : void 	<ul style="list-style-type: none"> + isEmpty() : bool + push(Node) : void + pop() : Node + peek() : Node + display() : void

미로 탐색을 위한 데이터와 메소드를 정의한 클래스

좌표정보와 Link 저장을 위한 노드 클래스

연결리스트, 배열을 구현한 클래스

#2. 클래스 관계도



#3. 소스코드 - 바뀐 점

```
public class Maze
```

```
{
```

```
    //필드를 static으로 선언
```

```
    public static int MAZE_ROW_SIZE = 0;
```

```
    public static int MAZE_COL_SIZE = 0;
```

```
    public static char[][] MAZE;
```

```
internal static void LoadMaze(string filename) //파일로부터 미로를 읽어오는 함수
```

```
{
```

```
    string mazeFile = File.ReadAllText(filename);
```

```
    //행, 열 정보 읽어오기
```

```
    foreach (char a in mazeFile)
```

```
    {
```

```
        if (a == '\n') break;
```

```
        MAZE_COL_SIZE++;
```

```
    }
```

```
    foreach (char a in mazeFile)
```

```
    {
```

```
        if (a == '\n') MAZE_ROW_SIZE++;
```

```
    }
```

```
    //이차원 배열 선언 및 미로 저장하기
```

```
    char[][] maze = new char[MAZE_ROW_SIZE][];
```

```
    for (int x = 0; x < MAZE_ROW_SIZE; x++)
```

```
    {
```

```
        maze[x] = new char[MAZE_COL_SIZE];
```

```
    }
```

```
    int i = 0, j = 0;
```

```
    foreach (char a in mazeFile)
```

```
    {
```

```
        if (j > MAZE_COL_SIZE) { i++; j = 0; }
```

```
        if (a != '\n')
```

```
            maze[i][j] = a;
```

```
            j++;
```

```
    }
```

```
    MAZE = maze;
```

```
}
```

2차원 배열의 동적할당을
다르게 변경하였습니다.

#3. 소스코드 - 바뀐 점

```
namespace MiroProgram_cs
{
    internal class Node
    {
        private Node link;
        public int row { get; set; }
        public int col { get; set; }
        internal Node Link { get; set; }
        internal Node() { }
        internal Node(int r, int c) { row = r; col = c; link = null; }
        internal void display()
        {
            Console.WriteLine($"{row}, {col} ");
        }
    }
}
```

기존의 Node 클래스에서 포인터를 사용하지 않고 선언하였습니다.

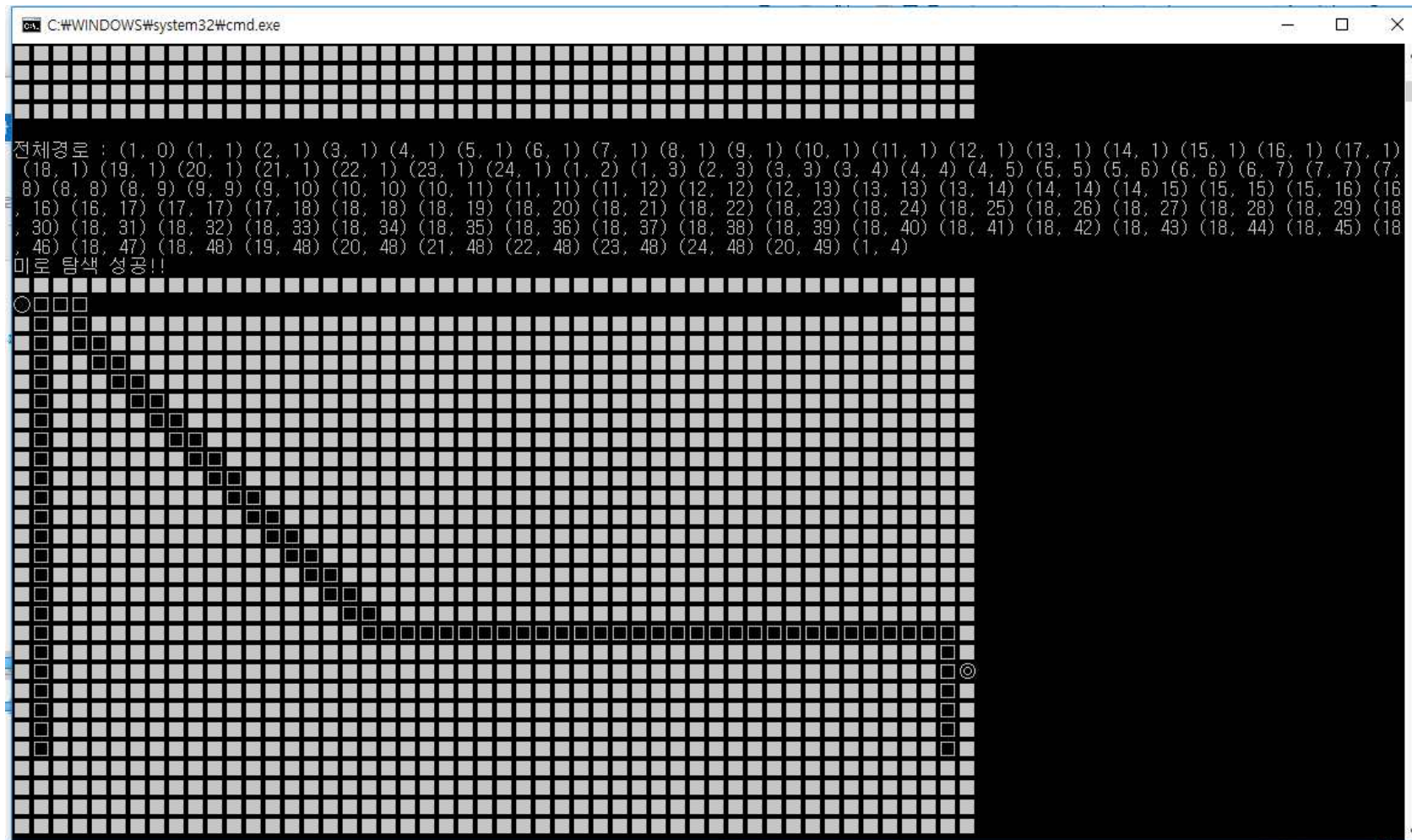
#3. 소스코드 - 바뀐 점

```
internal static Node FindEntry() //미로의 입구를 탐색하는 함수
```

```
{  
    Node entry = new Node();  
    Node exit = new Node();  
    for (int i = 0; i < Maze.MAZE_ROW_SIZE; i++)  
    {  
        for (int j = 0; j < Maze.MAZE_COL_SIZE; j++)  
        {  
            if (MAZE[0][j] == '0') { entry.row = 0; entry.col = j; }  
            else if (MAZE[i][0] == '0') { entry.row = i; entry.col = 0; }  
            if (MAZE[Maze.MAZE_ROW_SIZE - 1][j] == '0') { exit.row = Maze.MAZE_ROW_SIZE; exit.col = j; }  
            else if (MAZE[i][Maze.MAZE_COL_SIZE-2] == '0') { exit.row = i; exit.col = Maze.MAZE_COL_SIZE-2; }  
        }  
    }  
  
    MAZE[entry.row][entry.col] = 'e'; //미로의 입구를 e로 표시(출력시 가독성을 위해)  
    MAZE[exit.row][exit.col] = 'x'; //미로의 출구를 x로 표시  
    return entry;  
}
```

미로의 입구와 출구를 탐색하는 함수를 추가하였습니다.

#4. 실행화면



```
C:\WINDOWS\system32\cmd.exe

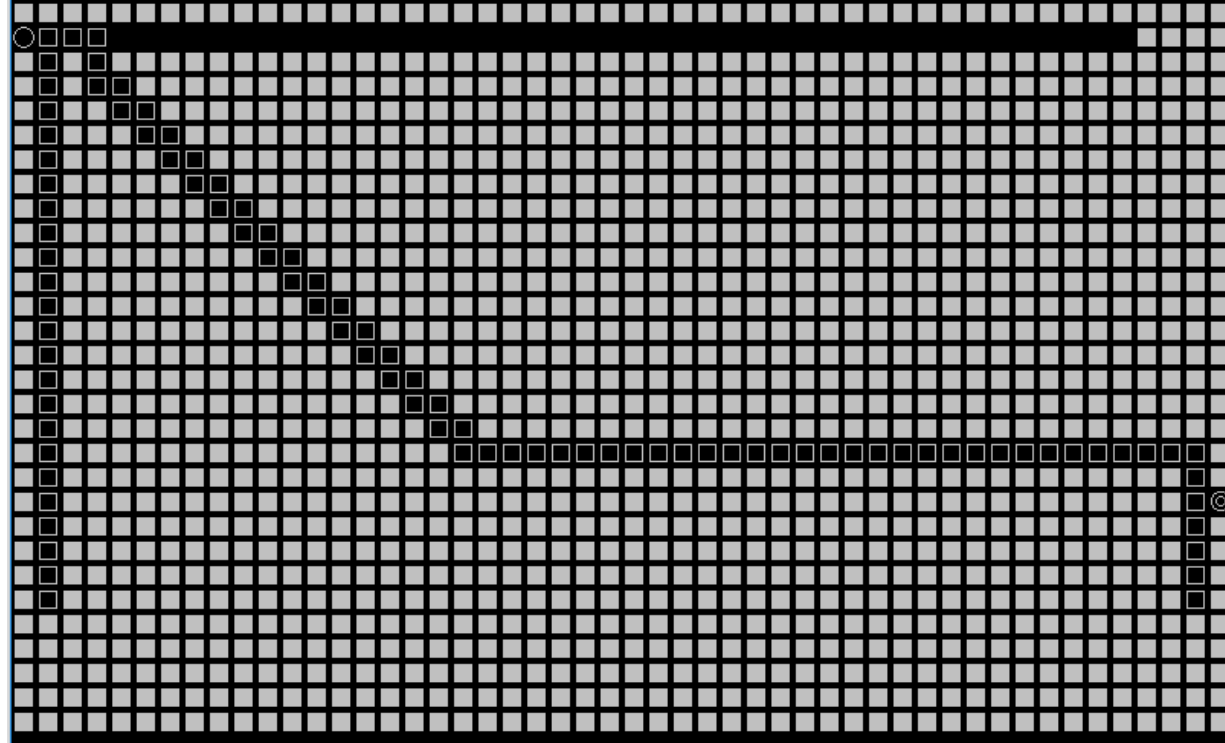
전체경로 : (1, 0) (1, 1) (2, 1) (3, 1) (4, 1) (5, 1) (6, 1) (7, 1) (8, 1) (9, 1) (10, 1) (11, 1) (12, 1) (13, 1) (14, 1) (15, 1) (16, 1) (17, 1)
(18, 1) (19, 1) (20, 1) (21, 1) (22, 1) (23, 1) (24, 1) (1, 2) (1, 3) (2, 3) (3, 3) (3, 4) (4, 4) (4, 5) (5, 5) (5, 6) (6, 6) (6, 7) (7, 7) (7,
8) (8, 8) (8, 9) (9, 9) (9, 10) (10, 10) (10, 11) (11, 11) (11, 12) (12, 12) (12, 13) (13, 13) (13, 14) (14, 14) (14, 15) (15, 15) (15, 16) (16
, 16) (16, 17) (17, 17) (17, 18) (18, 18) (18, 19) (18, 20) (18, 21) (18, 22) (18, 23) (18, 24) (18, 25) (18, 26) (18, 27) (18, 28) (18, 29) (18
, 30) (18, 31) (18, 32) (18, 33) (18, 34) (18, 35) (18, 36) (18, 37) (18, 38) (18, 39) (18, 40) (18, 41) (18, 42) (18, 43) (18, 44) (18, 45) (18
, 46) (18, 47) (18, 48) (19, 48) (20, 48) (21, 48) (22, 48) (23, 48) (24, 48) (20, 49) (1, 4)

미로 탐색 성공!!
```


#4. 실행화면

전체경로 : (1, 0) (1, 1) (2, 1) (3, 1) (4, 1) (5, 1) (6, 1) (7, 1) (8, 1) (9, 1) (10, 1) (11, 1) (12, 1) (13, 1) (14, 1) (15, 1) (16, 1) (17, 1) (18, 1) (19, 1) (20, 1) (21, 1) (22, 1) (23, 1) (24, 1) (1, 2) (1, 3) (2, 3) (3, 3) (3, 4) (4, 4) (4, 5) (5, 5) (5, 6) (6, 6) (6, 7) (7, 7) (7, 8) (8, 8) (8, 9) (9, 9) (9, 10) (10, 10) (10, 11) (11, 11) (11, 12) (12, 12) (12, 13) (13, 13) (13, 14) (14, 14) (14, 15) (15, 15) (15, 16) (16, 16) (16, 17) (17, 17) (17, 18) (18, 18) (18, 19) (18, 20) (18, 21) (18, 22) (18, 23) (18, 24) (18, 25) (18, 26) (18, 27) (18, 28) (18, 29) (18, 30) (18, 31) (18, 32) (18, 33) (18, 34) (18, 35) (18, 36) (18, 37) (18, 38) (18, 39) (18, 40) (18, 41) (18, 42) (18, 43) (18, 44) (18, 45) (18, 46) (18, 47) (18, 48) (19, 48) (20, 48) (21, 48) (22, 48) (23, 48) (24, 48) (20, 49) (1, 4)

미로 탐색 성공!!



#5. github url

- https://github.com/shinsoyeon16/MiroProgram_cs

이상입니다.