REPORT 1 차 설계서

KU 건국대학교 KONKUK UNIV.

과 목 명 | 전공기초프로젝트 1

담당교수 | 차 리 서 교수님

학과 | 소프트웨어학과

소속 | 1 팀

팀원 | 201714150 김 동 진

201714151 박 민 기

201714152 박 종 현

201714158 허 승 회

제출일 | 2019.04.10

미로 찾기 프로그램 1 차 설계서

(2019 전공기초프로젝트 1)

화 · 목 분반 1 팀

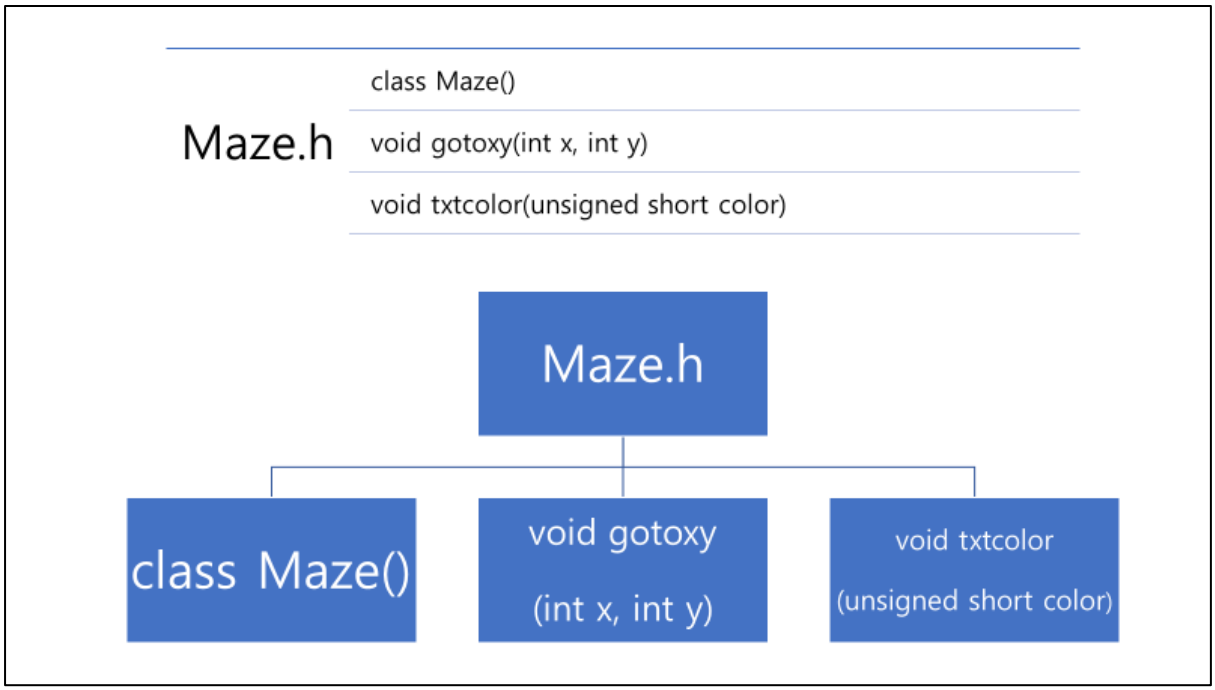
소프트웨어학과 201714150 김 동 진

소프트웨어학과 201714151 박 민 기

소프트웨어학과 201714152 박 종 현

소프트웨어학과 201714158 허 승 회

1. 계층 및 용어 정리



Maze Class Method

```
void makeMaze()
void checkMaze()
void startMaze()
void mazeSize()                          // 맵 크기 계산
void printMaze()                         // 화면에 맵 프린트
void storeInArray()                      // 2차 배열에 집어넣기
Maze(); // 생성자
```

Maze Class Variable

```
int i, j
int row, col
int **map
char tmp
ifstream file
string name
int exitcount, exit1_col, exit1_row, exit2_col, exit2_row, exit3_col, exit3_row
int entrance_col, entrance_row
int , exit4_col, exit4_row, exit5_col, exit5_row
```

Maze() Constructor

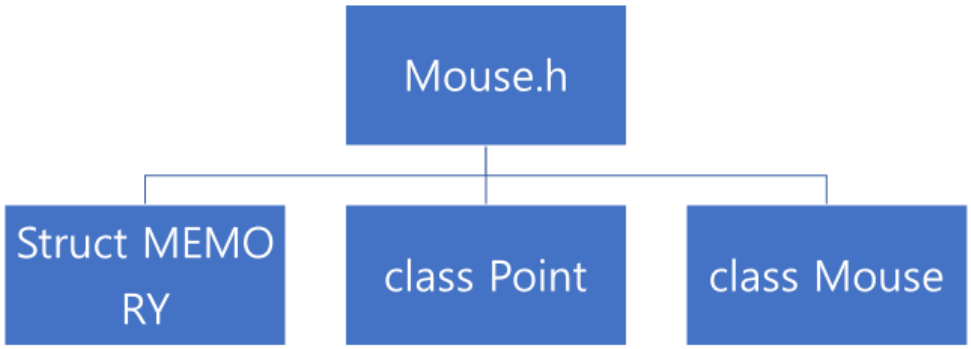
```
string name;

row = 1;

col = 0;

exitcount = 1;

exit1_col = exit1_row = exit2_col = exit2_row = exit3_col = exit3_row = e
xit4_col = exit4_row = exit5_col = exit5_row = 500;

entrance_col =1;

entrance_row = 1;
```

Mouse.h

Struct MEMORY

class Point

class Mouse



Mouse

(Maze &maze)

Constructor

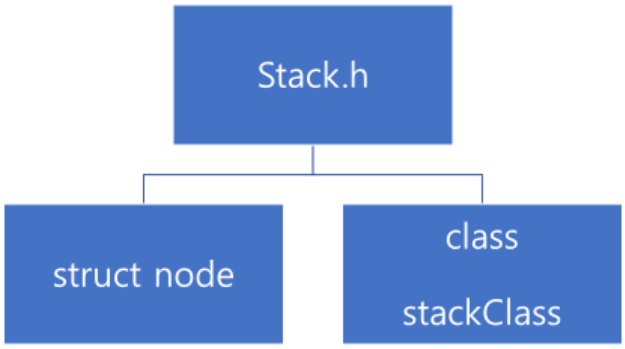
```
row = maze.row;
col = maze.col;
m_col = maze.entrance_col;
m_row = maze.entrance_row;
energy = row * col * 2;
memory = (MEMORY * *)malloc((row) * sizeof(MEMORY*));
맵사이즈 만큼 메모리 동적 할당 뒤
Memory 구조체에 알맞게 채워 넣는다
save = (Point * *)malloc((row * 100) * sizeof(Point*));
 // 갈림길 좌표 저장 포인터
save[tel] = new Point(m_row, m_col, teleport_count);
```

Class Mouse	
	int i, j
	int col, row // 좌표
	int m_col, m_row; // 쥐의 좌표
	int energy; // default = row*col*2
	int tpcol, tprow;
	int checkc, checkr;
	MEMORY **memory; // mouse가 지나왔던 길 저장
	Point **save;
	stackClass alreadyStack; // 최단거리 길을 저장 할 스택
	Mouse(Maze& maze); // 생성자
	bool asktp();
	void movePaint(int **map); // 쥐 움직이는 모습 시각화
	void lookAround(int **map); // 주변을 둘러보고 메모리에 저장 & 갈림길 수 return
	void strategy(int **map); // 전략

Class Point	
	int x;
	int y;
	int cnt;
	Point(int x, int y, int cnt)

Struct Memory	
	int state; // 0이면 길, 1이면 벽
	int trace; // 0이면 갈 수 있는 장소, 1이면 갔던 장소
	int crossroads; // 갈림길

Stack.h	struct node
	class stackClass

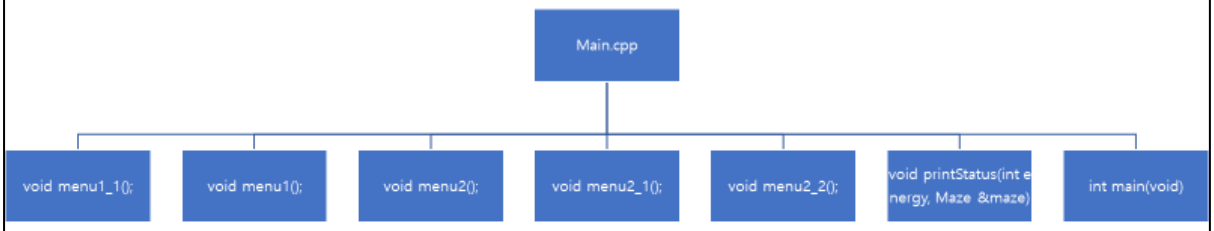


Struct node	int row;
	int col;
	node *Next;

class stackClass	int pointCol;
	int pointRow;
	int size;
	stackClass();
	stackClass(const stackClass & s);
	~stackClass();
	void Push(int m_row, int m_col);
	void Pop();
	int Size();
	boolean IsEmpty();
	node_pointer Top;

Main.cpp

```
void menu1_1();
void menu1();
void menu2();
void menu2_1();
void menu2_2();
void printStatus(int energy, Maze &maze)
int main(void)
```

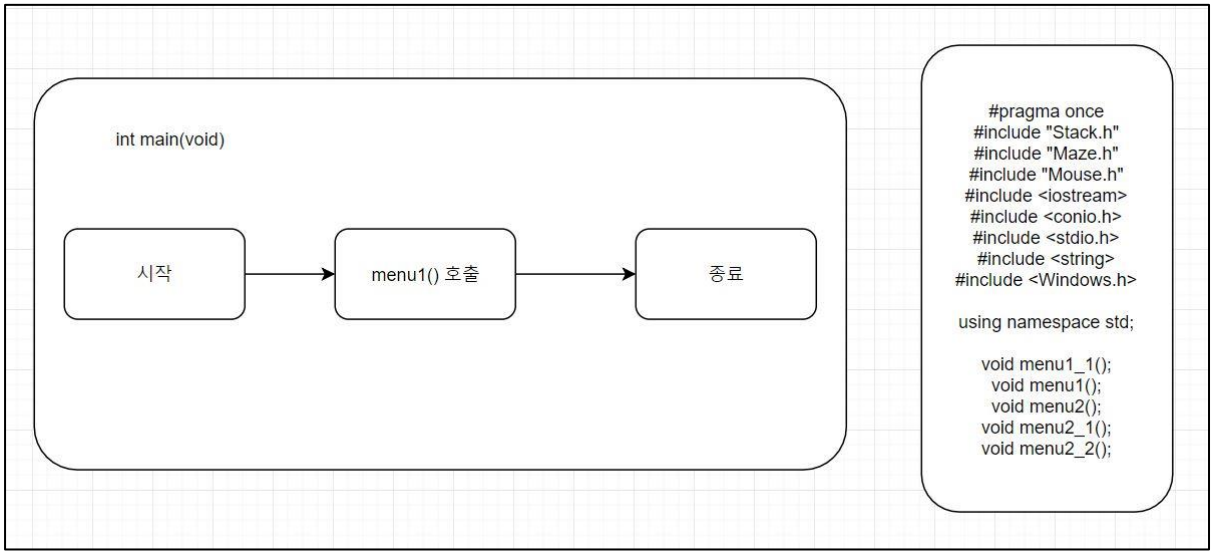


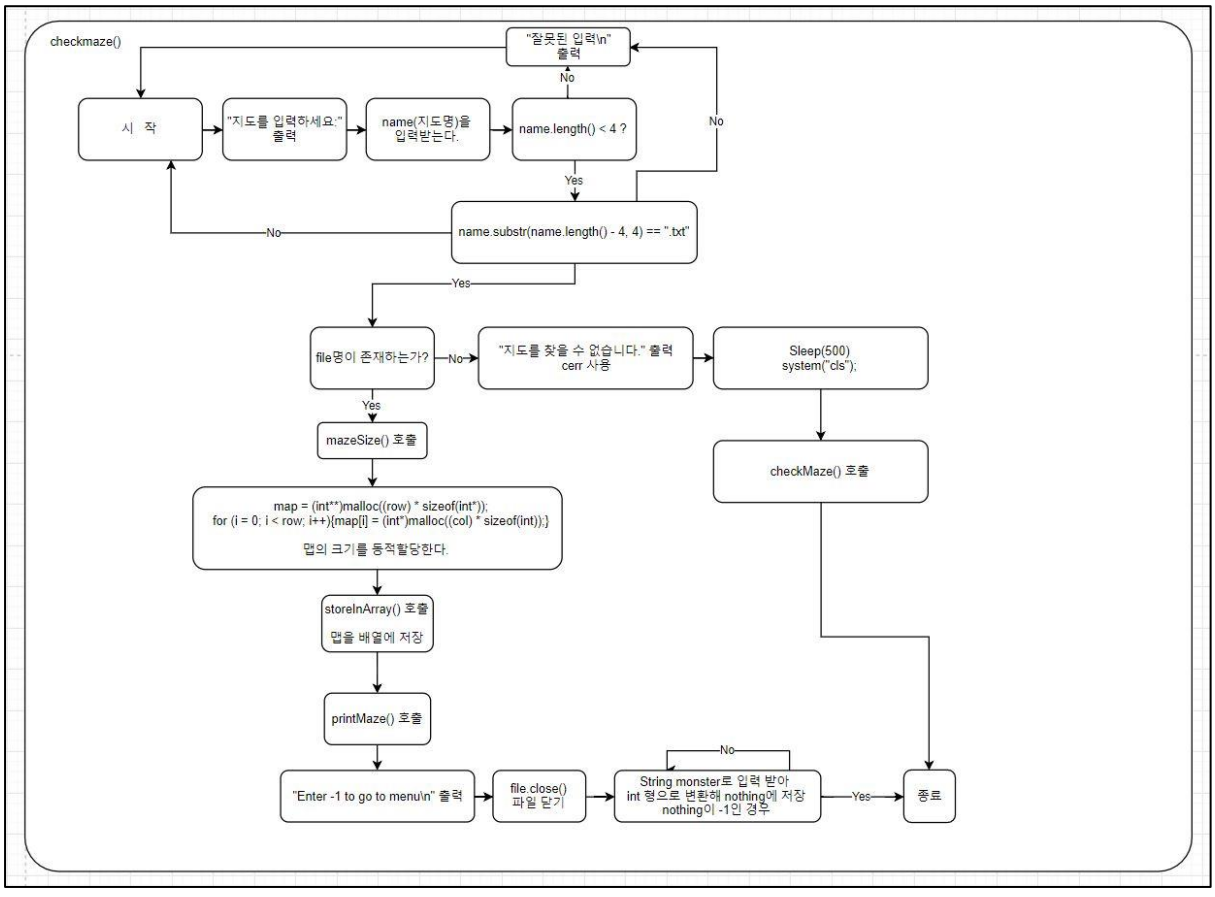
외부 참조 라이브러리

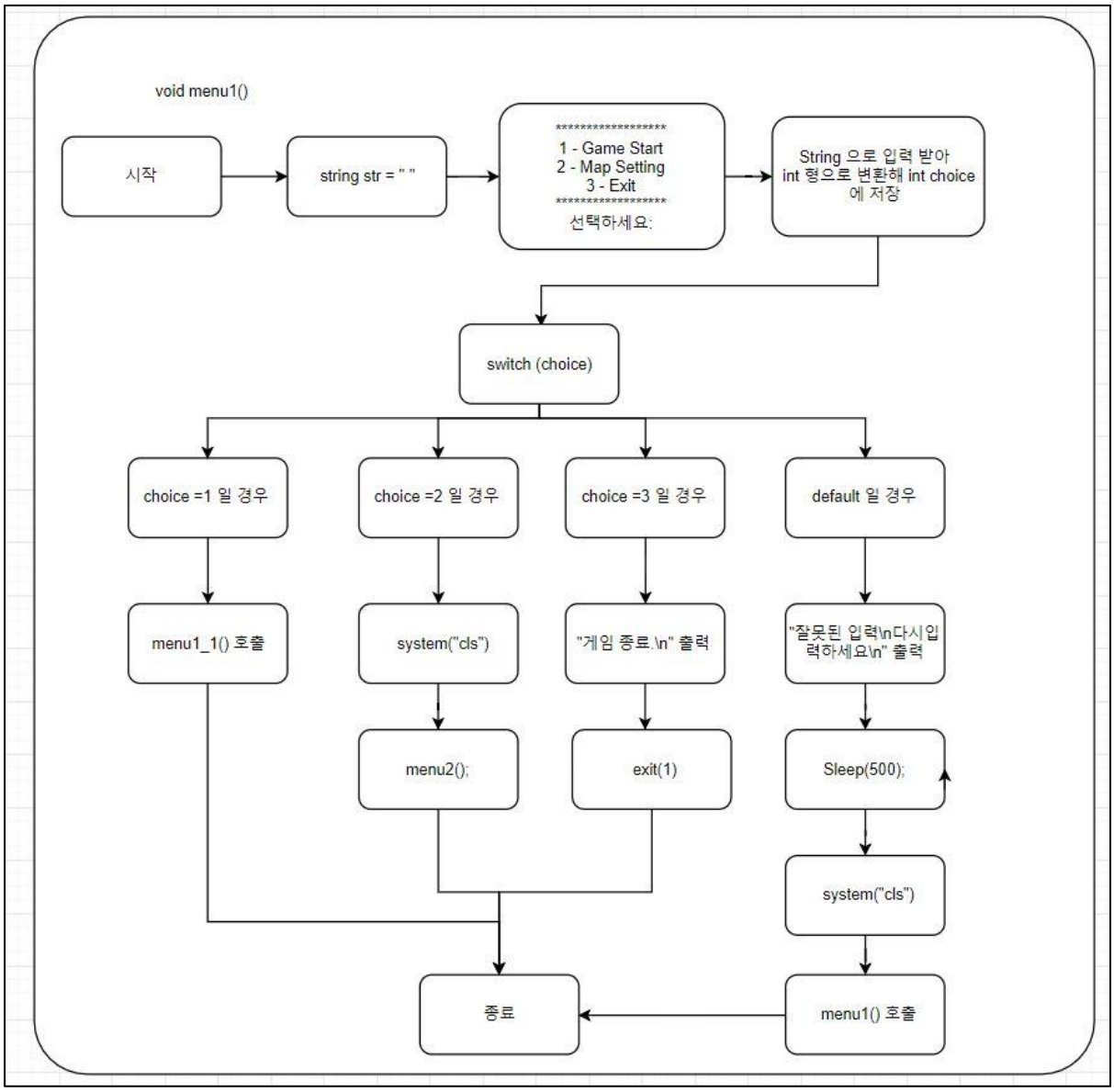
```
<iostream>
<conio.h>
<stdio.h>
<string>
<Windows.h>
```

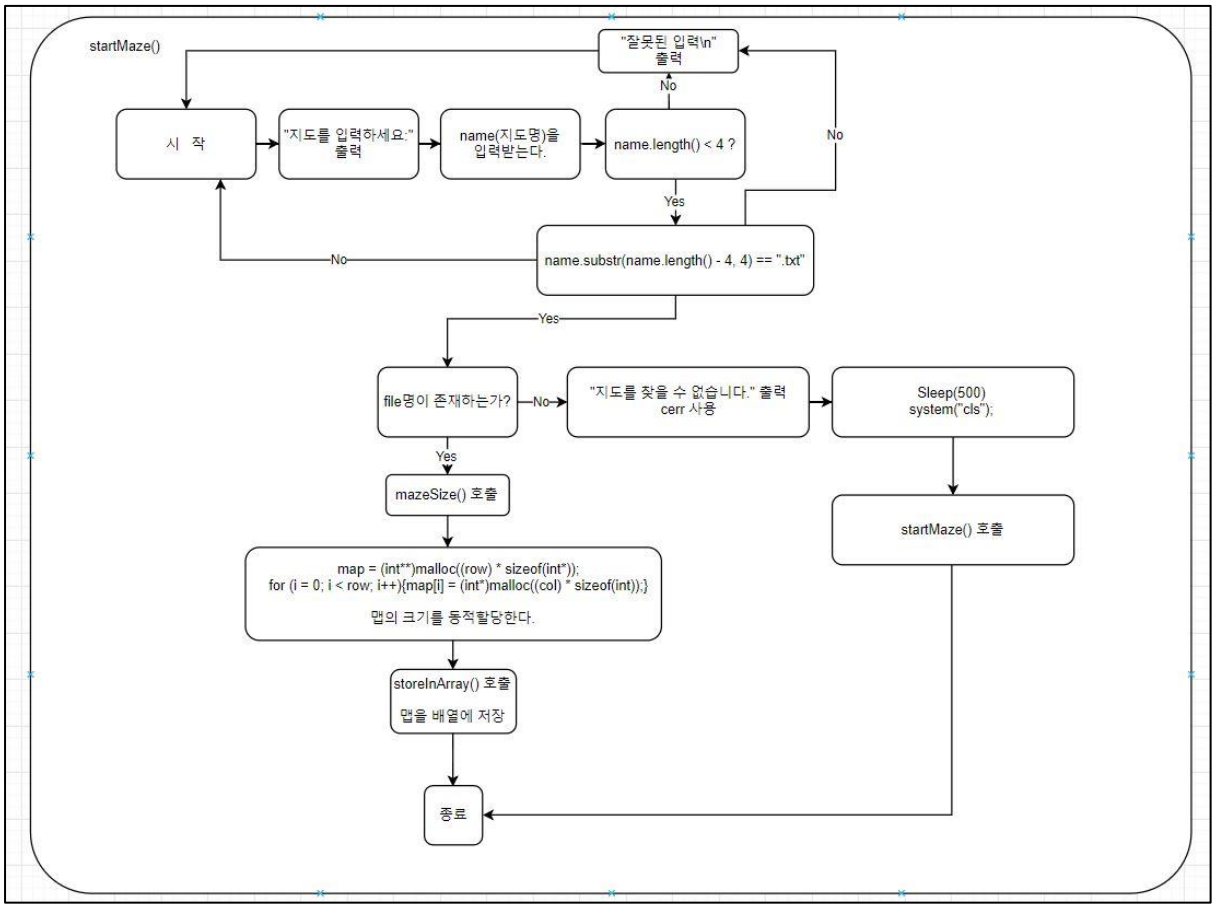
2. 맵 작성

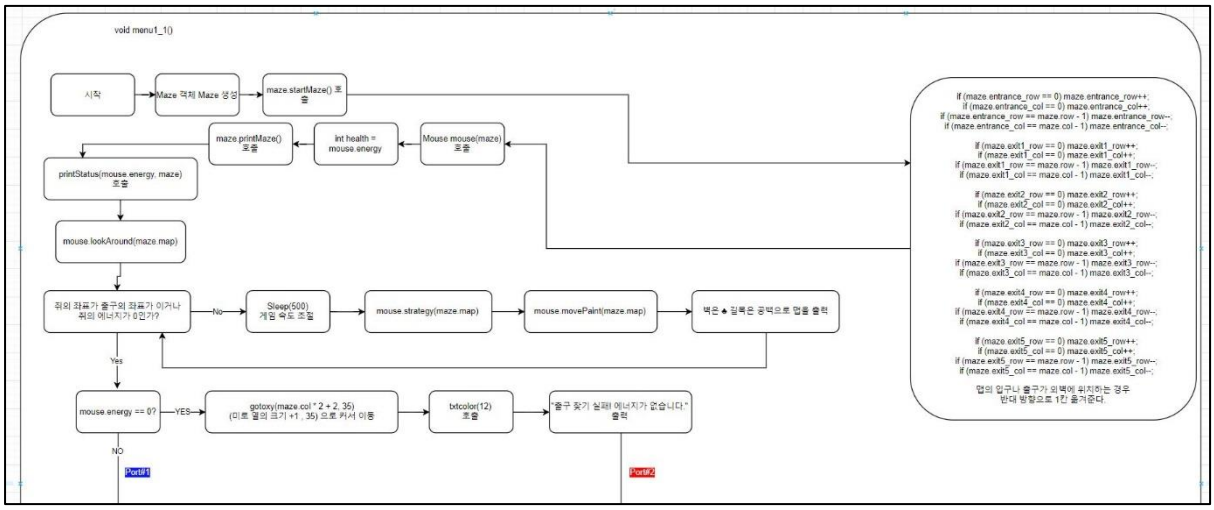
Main

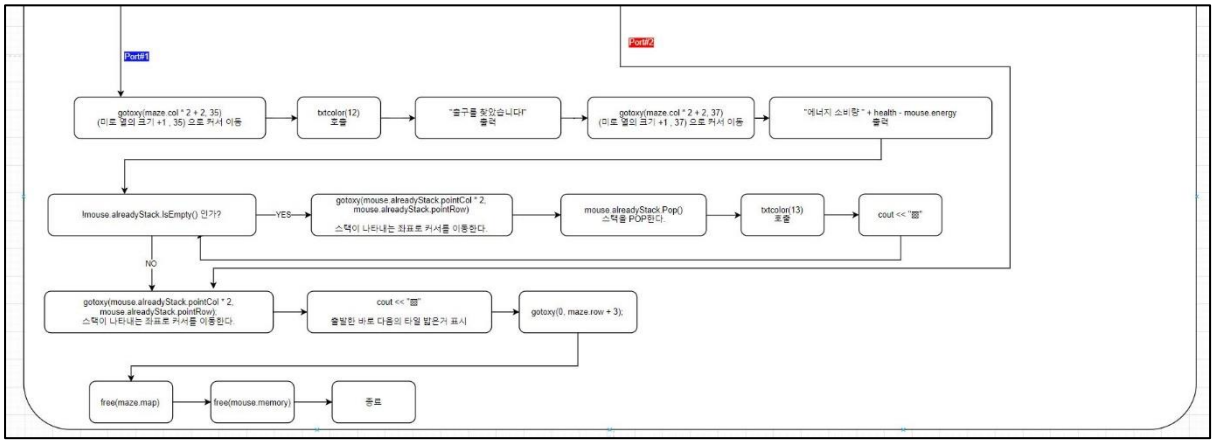

int main(void)
시작
menu1() 호출
종료
#pragma once
#include "Stack.h"
#include "Maze.h"
#include "Mouse.h"
#include <iostream>
#include <conio.h>
#include <stdio.h>
#include <string>
#include <Windows.h>
using namespace std;
void menu1_1();
void menu1();
void menu2();
void menu2_1();
void menu2_2();

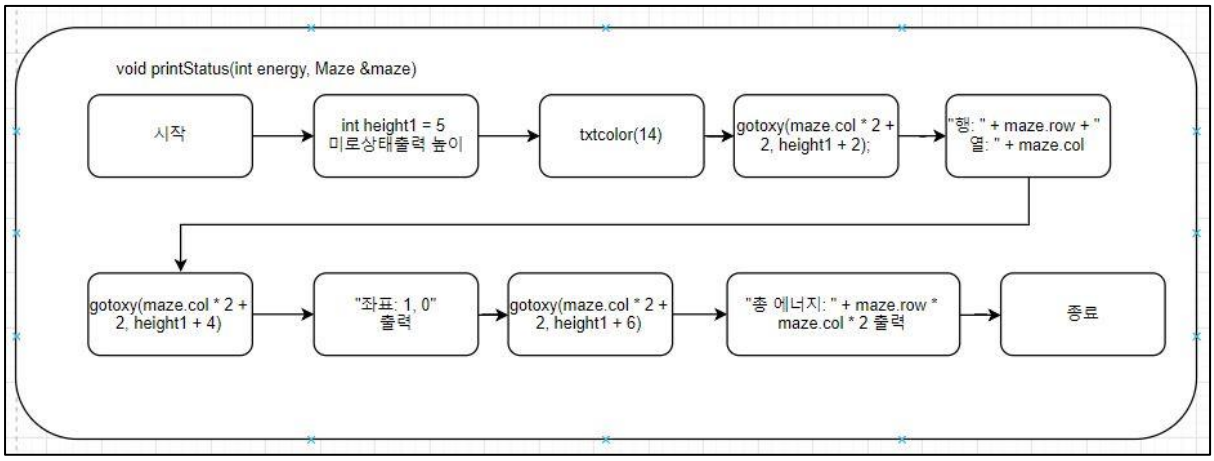

checkmaze()
"잘못된 입력\n" 출력
No
시 작
"지도를 입력하세요:" 출력
name(지도명)을 입력받는다.
name.length() < 4 ?
No
Yes
No
name.substr(name.length() - 4, 4) == ".txt"
Yes
file명이 존재하는가?
No
"지도를 찾을 수 없습니다." 출력
cerr 사용
Sleep(500)
system("cls");
Yes
mazeSize() 호출
checkMaze() 호출
map = (int**)malloc((row) * sizeof(int*));
for (i = 0; i < row; i++){map[i] = (int*)malloc((col) * sizeof(int));}
맵의 크기를 동적할당한다.
storeInArray() 호출
맵을 배열에 저장
printMaze() 호출
"Enter -1 to go to menu\n" 출력
file.close()
파일 닫기
No
String monster로 입력 받아
int 형으로 변환해 nothing에 저장
nothing이 -1인 경우
Yes
종료

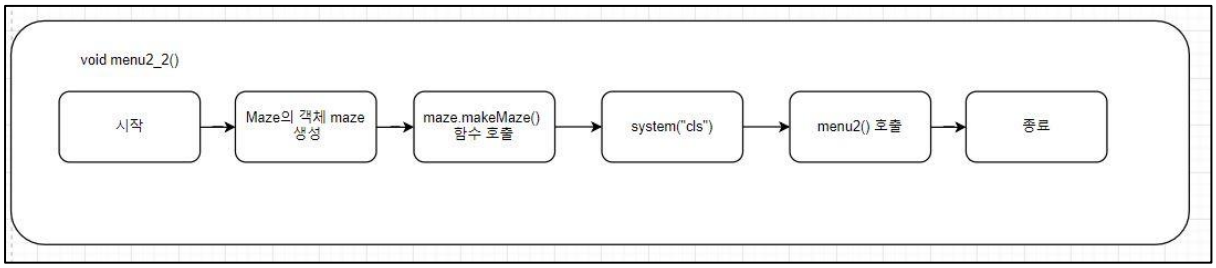

void menu1()
시작
string str = " "
****
1 - Game Start
2 - Map Setting
3 - Exit
****
선택하세요:
String 으로 입력 받아 int 형으로 변환해 int choice 에 저장
switch (choice)
choice =1 일 경우
choice =2 일 경우
choice =3 일 경우
default 일 경우
menu1_1() 호출
system("cls")
"게임 종료.\n" 출력
"잘못된 입력\n다시입력하세요\n" 출력
menu2();
exit(1)
Sleep(500);
system("cls")
종료
menu1() 호출

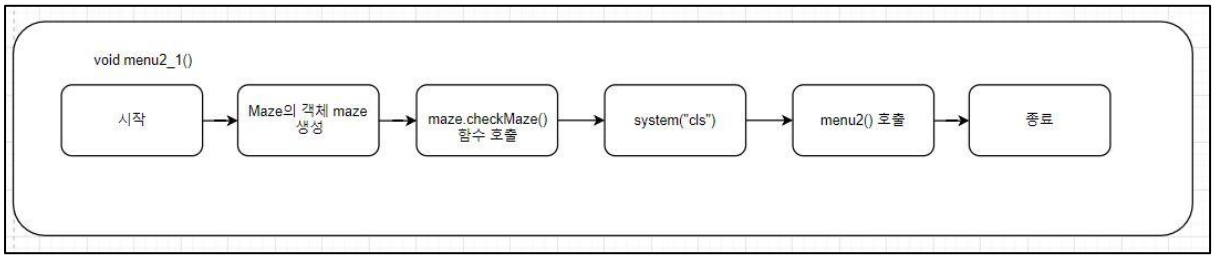

startMaze()
"잘못된 입력\n"
출력
No
시 작
"지도를 입력하세요:"
출력
name(지도명)을
입력받는다.
name.length() < 4 ?
No
Yes
No
name.substr(name.length() - 4, 4) == ".txt"
Yes
file명이 존재하는가?
No
"지도를 찾을 수 없습니다." 출력
cerr 사용
Sleep(500)
system("cls");
Yes
mazeSize() 호출
startMaze() 호출
map = (int**)malloc((row) * sizeof(int*));
for (i = 0; i < row; i++){map[i] = (int*)malloc((col) * sizeof(int));}
맵의 크기를 동적할당한다.
storeInArray() 호출
맵을 배열에 저장
종료

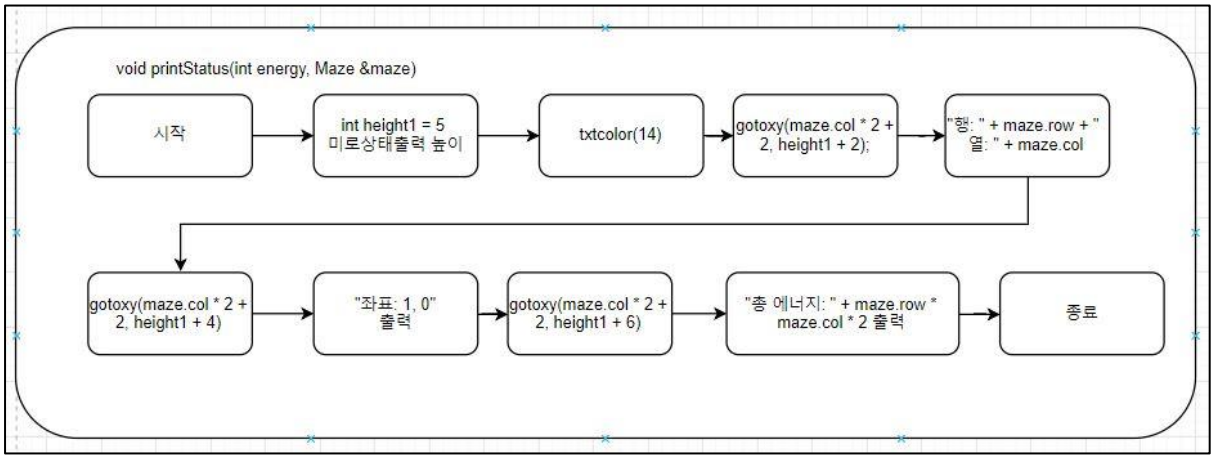

void menu1_1()
시작
Maze 객체 Maze 생성
maze.startMaze() 호출
maze.printMaze() 호출
int health = mouse.energy
Mouse mouse(maze) 호출
printStatus(mouse.energy, maze) 호출
mouse.lookAround(maze.map)
쥐의 좌표가 출구의 좌표가 이거나 쥐의 에너지가 0인가?
No
Sleep(500) 게임 속도 조절
mouse.strategy(maze.map)
mouse.movePaint(maze.map)
벽은 ■ 길목은 공백으로 맵을 출력
Yes
mouse.energy == 0?
YES
gotoxy(maze.col * 2 + 2, 35) (미로 열의 크기 +1 , 35) 으로 커서 이동
txtcolor(12) 호출
"출구 찾기 실패 에너지가 없습니다." 출력
NO
PortIf1
PortIf2
if (maze.entrance_row == 0) maze.entrance_row++;
if (maze.entrance_col == 0) maze.entrance_col++;
if (maze.entrance_row == maze.row - 1) maze.entrance_row--;
if (maze.entrance_col == maze.col - 1) maze.entrance_col--;
if (maze.exit1_row == 0) maze.exit1_row++;
if (maze.exit1_col == 0) maze.exit1_col++;
if (maze.exit1_row == maze.row - 1) maze.exit1_row--;
if (maze.exit1_col == maze.col - 1) maze.exit1_col--;
if (maze.exit2_row == 0) maze.exit2_row++;
if (maze.exit2_col == 0) maze.exit2_col++;
if (maze.exit2_row == maze.row - 1) maze.exit2_row--;
if (maze.exit2_col == maze.col - 1) maze.exit2_col--;
if (maze.exit3_row == 0) maze.exit3_row++;
if (maze.exit3_col == 0) maze.exit3_col++;
if (maze.exit3_row == maze.row - 1) maze.exit3_row--;
if (maze.exit3_col == maze.col - 1) maze.exit3_col--;
if (maze.exit4_row == 0) maze.exit4_row++;
if (maze.exit4_col == 0) maze.exit4_col++;
if (maze.exit4_row == maze.row - 1) maze.exit4_row--;
if (maze.exit4_col == maze.col - 1) maze.exit4_col--;
if (maze.exit5_row == 0) maze.exit5_row++;
if (maze.exit5_col == 0) maze.exit5_col++;
if (maze.exit5_row == maze.row - 1) maze.exit5_row--;
if (maze.exit5_col == maze.col - 1) maze.exit5_col--;
맵의 입구나 출구가 외벽에 위치하는 경우 반대 방향으로 1칸 옮겨준다.


PortIf1
PortIf2
gotoxy(maze.col * 2 + 2, 35) (미로 열의 크기 +1 , 35) 으로 커서 이동
txtcolor(12) 호출
"출구를 찾았습니다!" 출력
gotoxy(maze.col * 2 + 2, 37) (미로 열의 크기 +1 , 37) 으로 커서 이동
"에너지 소비량 " + health - mouse.energy 출력
!mouse.alreadyStack.IsEmpty() 인가?
YES
gotoxy(mouse.alreadyStack.pointCol * 2, mouse.alreadyStack.pointRow) 스택이 나타내는 좌표로 커서를 이동한다.
mouse.alreadyStack.Pop() 스택을 POP한다.
txtcolor(13) 호출
cout << "■"
NO
gotoxy(mouse.alreadyStack.pointCol * 2, mouse.alreadyStack.pointRow) 스택이 나타내는 좌표로 커서를 이동한다.
cout << "■" 출발한 바로 다음의 타일 발온거 표시
gotoxy(0, maze.row + 3);
free(maze.map)
free(mouse.memory)
종료


void printStatus(int energy, Maze &maze)
시작
int height1 = 5 미로상태출력 높이
txtcolor(14)
gotoxy(maze.col * 2 + 2, height1 + 2);
"행: " + maze.row + " 열: " + maze.col
gotoxy(maze.col * 2 + 2, height1 + 4)
"좌표: 1, 0" 출력
gotoxy(maze.col * 2 + 2, height1 + 6)
"총 에너지: " + maze.row * maze.col * 2 출력
종료


void menu2_2()
시작
Maze의 객체 maze 생성
maze.makeMaze() 함수 호출
system("cls")
menu2() 호출
종료


void menu2_1()
시작
Maze의 객체 maze 생성
maze.checkMaze() 함수 호출
system("cls")
menu2() 호출
종료


void printStatus(int energy, Maze &maze)
시작
int height1 = 5 미로상태출력 높이
txtcolor(14)
gotoxy(maze.col * 2 + 2, height1 + 2);
"행: " + maze.row + " 열: " + maze.col
gotoxy(maze.col * 2 + 2, height1 + 4)
"좌표: 1, 0" 출력
gotoxy(maze.col * 2 + 2, height1 + 6)
"총 에너지: " + maze.row * maze.col * 2 출력
종료

Mouse

```
#include "Mouse.h"
int tel = 0;
int teleport_number = 1;
int teleport_count = 1;

Mouse::Mouse(Maze& maze)
{
 row = maze.row;
 col = maze.col;
 m_col = maze.entrance_col;
 m_row = maze.entrance_row;
 energy = row * col * 2;
 memory = (MEMORY * *)malloc((row) * sizeof(MEMORY*));
 for (i = 0; i < row; i++)
 {
  memory[i] = (MEMORY*)malloc((col) * sizeof(MEMORY));
 }
 for (i = 0; i < row; i++)
 {
  for (j = 0; j < col; j++)
  {
   memory[i][j].trace = 0;
   if (i == 0 || i == row - 1) // 외곽벽 (위, 아래)
   {
    memory[i][j].state = 1;
    memory[i][j].trace = 1;
   }
  }
  memory[i][0].state = 1; // 외곽벽 (왼쪽, 오른쪽)
  memory[i][0].trace = 1;
  memory[i][col - 1].state = 1;
  memory[i][col - 1].trace = 1;
 }

 save = (Point * *)malloc((row * 100) * sizeof(Point*));   // 갈림길 좌표 저장 포인터
 for (i = 0; i < row * 100; i++)
 {
  save[i] = (Point*)malloc((col * 100) * sizeof(Point));
 } // 동적할당
 save[tel] = new Point(m_row, m_col, teleport_count);
}
```

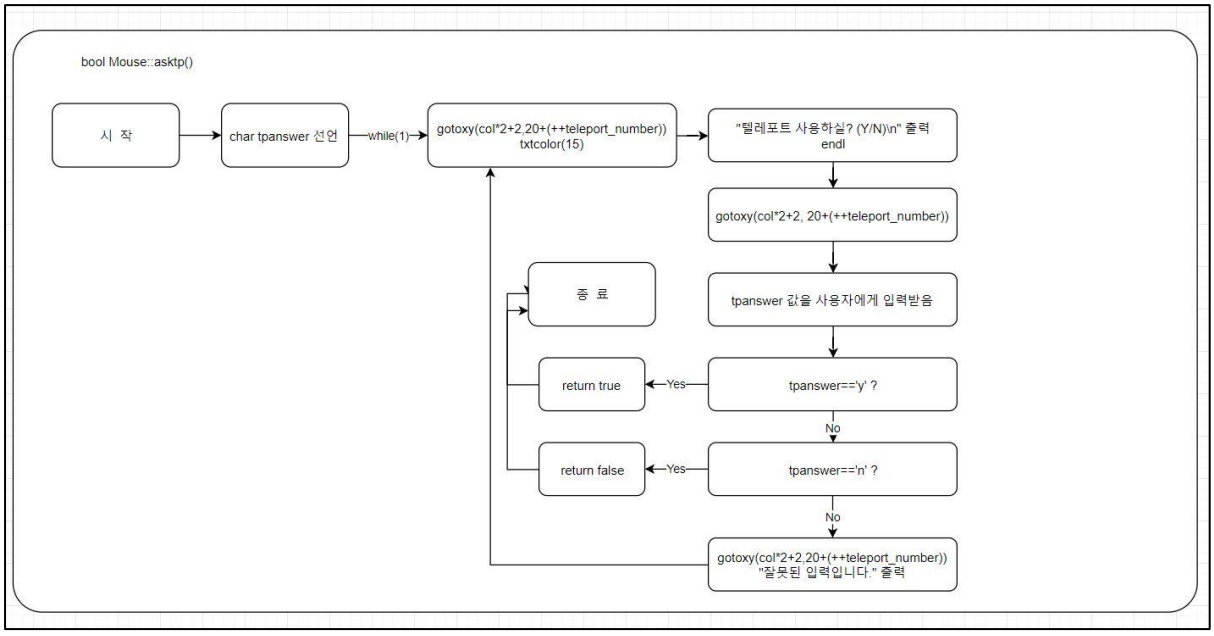
코드에 대한 설명부

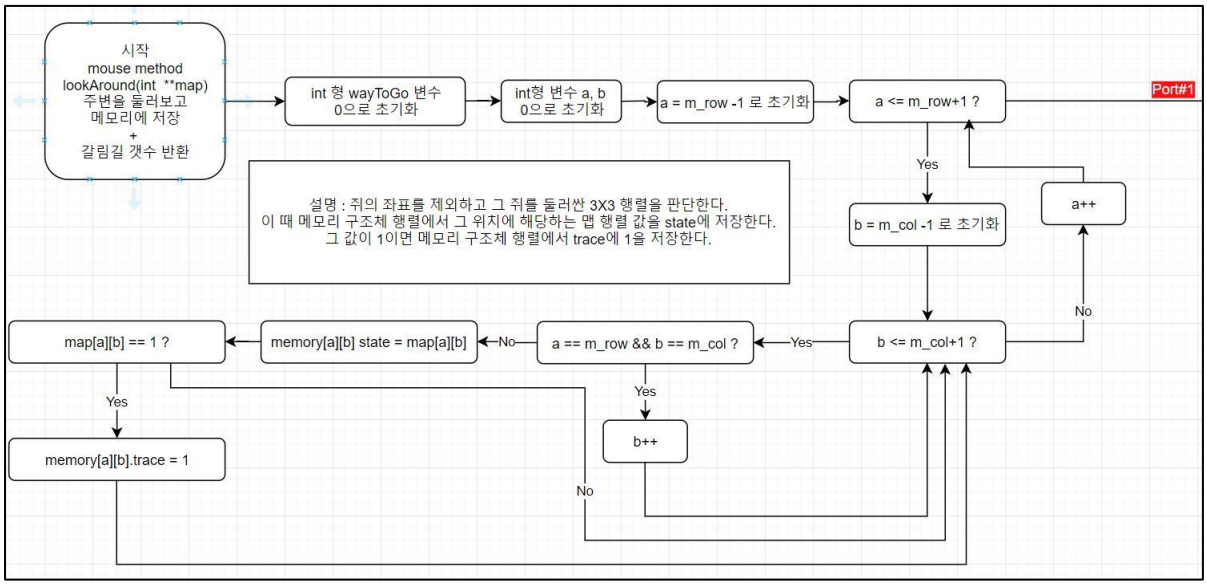
save 포인터 인덱스인 tel에 0을 저장
텔레포트를 사용한 횟수인 teleport_number에 1을 저장
텔레포트를 사용한 횟수인 teleport_count 에 1을 저장

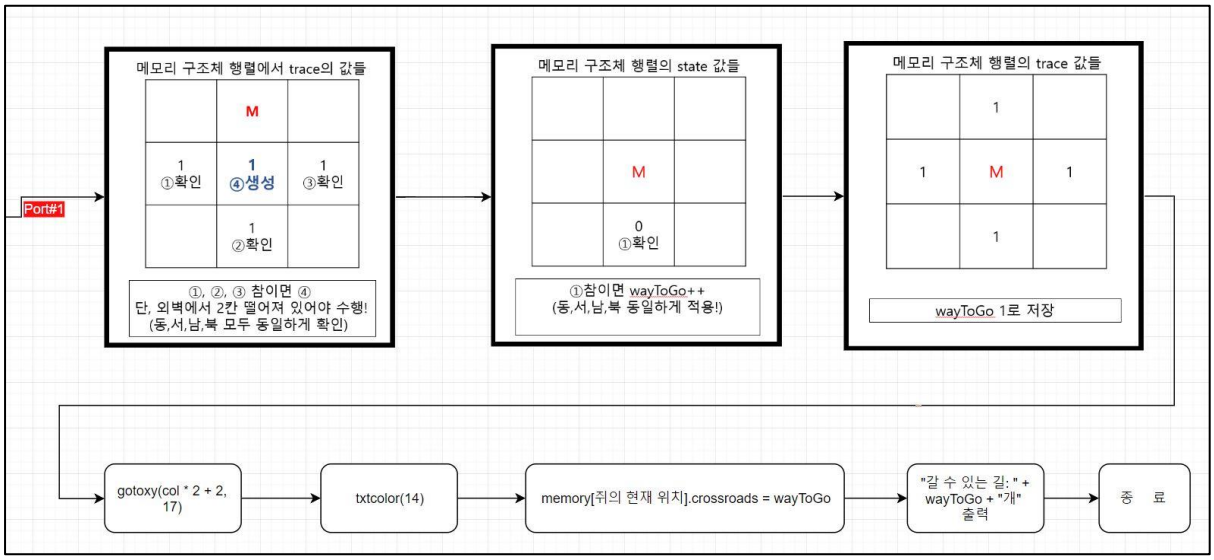
Mouse::Mouse(Maze &maze)
row 와 col에 미로의 행과 열을 저장
m_col과 m_row에 미로의 입구에 대한 행과 열을 저장
에너지는 미로의 행*열*2이다.
구조체 memory를 미로의 행과 열 크기 만큼 메모리 동적할당 한다.

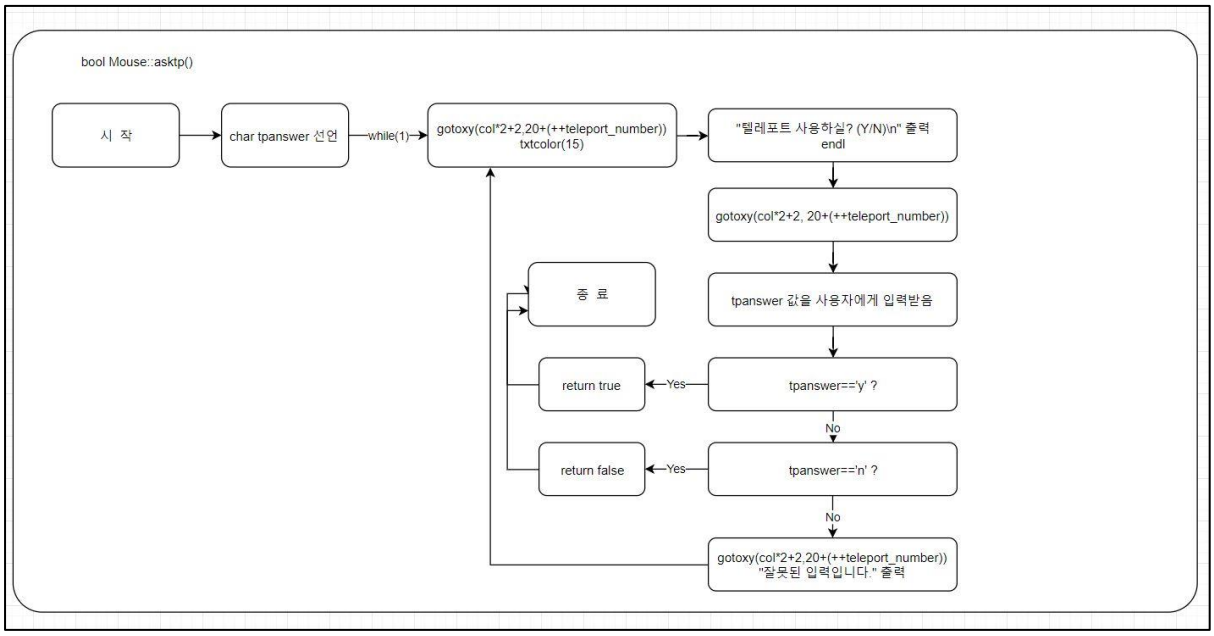
구조체 memory에 대하여 미로의 가장 바깥벽은 state와 trace를 1로 저장

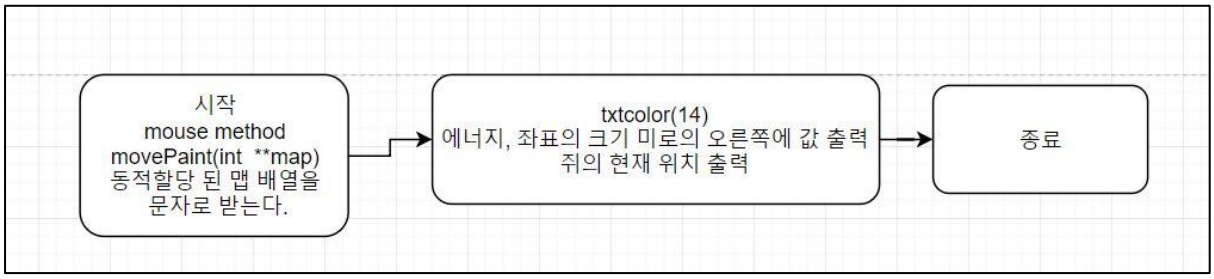
갈림길 좌표 저장 포인터인 save에 미로의 행과 열 만큼의 동적할당한다.

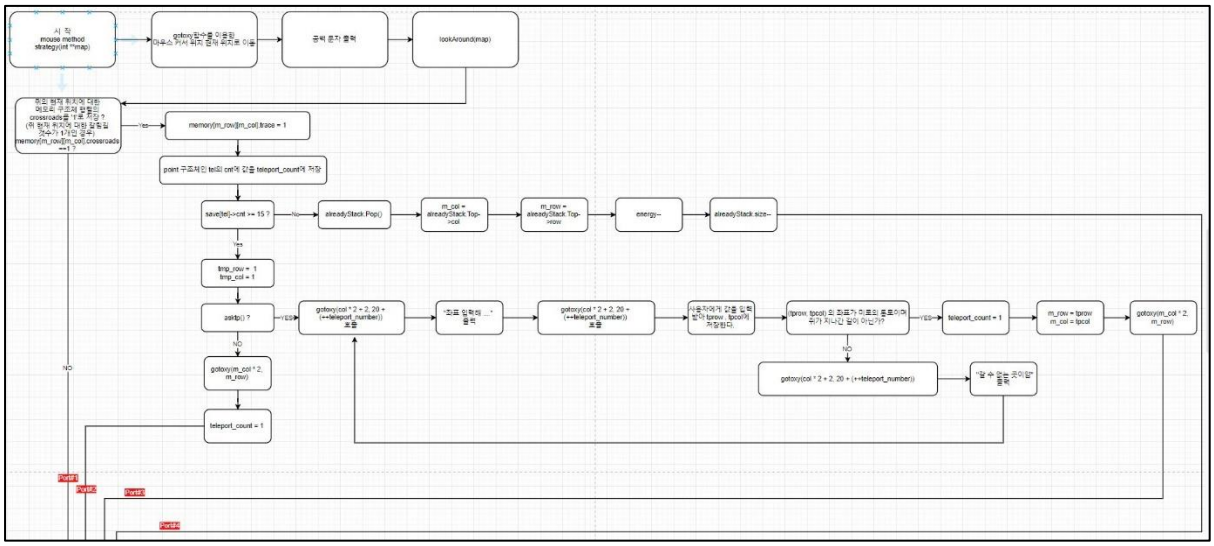


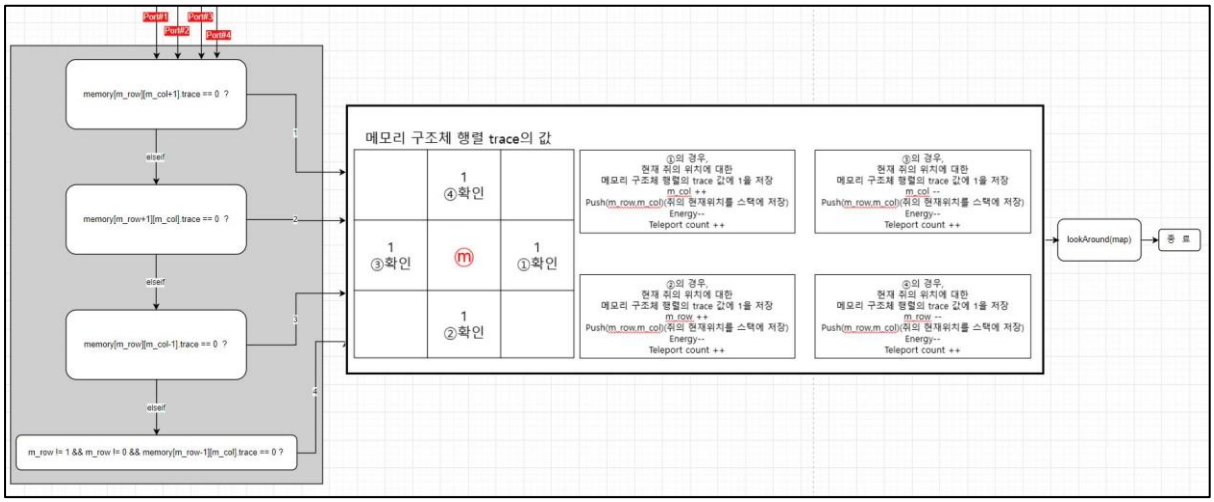

시작
mouse method
lookAround(int **map)
주변을 둘러보고
메모리에 저장
+
갈림길 갯수 반환
int 형 wayToGo 변수
0으로 초기화
int형 변수 a, b
0으로 초기화
a = m_row -1 로 초기화
a <= m_row+1 ?
Port#1
Yes
a++
b = m_col -1 로 초기화
No
b <= m_col+1 ?
Yes
a == m_row && b == m_col ?
No
memory[a][b] state = map[a][b]
map[a][b] == 1 ?
Yes
memory[a][b].trace = 1
Yes
b++
No
설명 : 쥐의 좌표를 제외하고 그 쥐를 둘러싼 3X3 행렬을 판단한다.
이 때 메모리 구조체 행렬에서 그 위치에 해당하는 맵 행렬 값을 state에 저장한다.
그 값이 1이면 메모리 구조체 행렬에서 trace에 1을 저장한다.


Port#1
메모리 구조체 행렬에서 trace의 값들
M
1
①확인
1
④생성
1
③확인
1
②확인
①, ②, ③ 참이면 ④
단, 외벽에서 2칸 떨어져 있어야 수행!
(동,서,남,북 모두 동일하게 확인)
메모리 구조체 행렬의 state 값들
M
0
①확인
①참이면 wayToGo++
(동,서,남,북 동일하게 적용!)
메모리 구조체 행렬의 trace 값들
1
1
M
1
1
wayToGo 1로 저장
gotoxy(col * 2 + 2, 17)
txtcolor(14)
memory[쥐의 현재 위치].crossroads = wayToGo
"갈 수 있는 길: " + wayToGo + "개" 출력
종 료

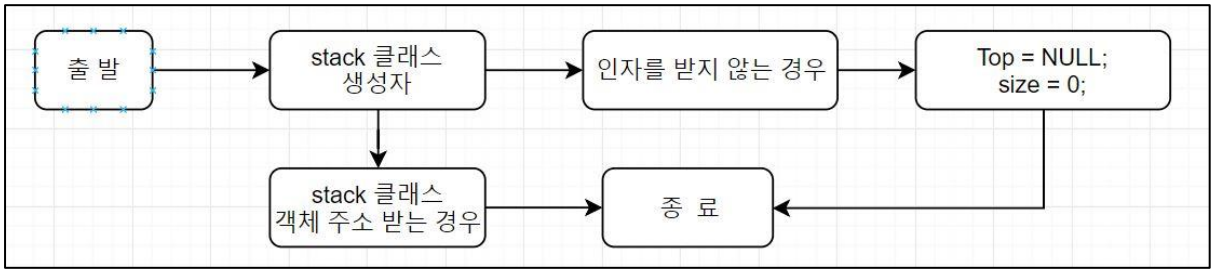

bool Mouse::asktp()
시 작
char tpanswer 선언
while(1)
gotoxy(col*2+2,20+(++teleport_number))
txtcolor(15)
"텔레포트 사용하실? (Y/N)\n" 출력
endl
gotoxy(col*2+2, 20+(++teleport_number))
tpanswer 값을 사용자에게 입력받음
tpanswer=='y' ?
Yes
return true
No
tpanswer=='n' ?
Yes
return false
No
gotoxy(col*2+2,20+(++teleport_number))
"잘못된 입력입니다." 출력
종 료

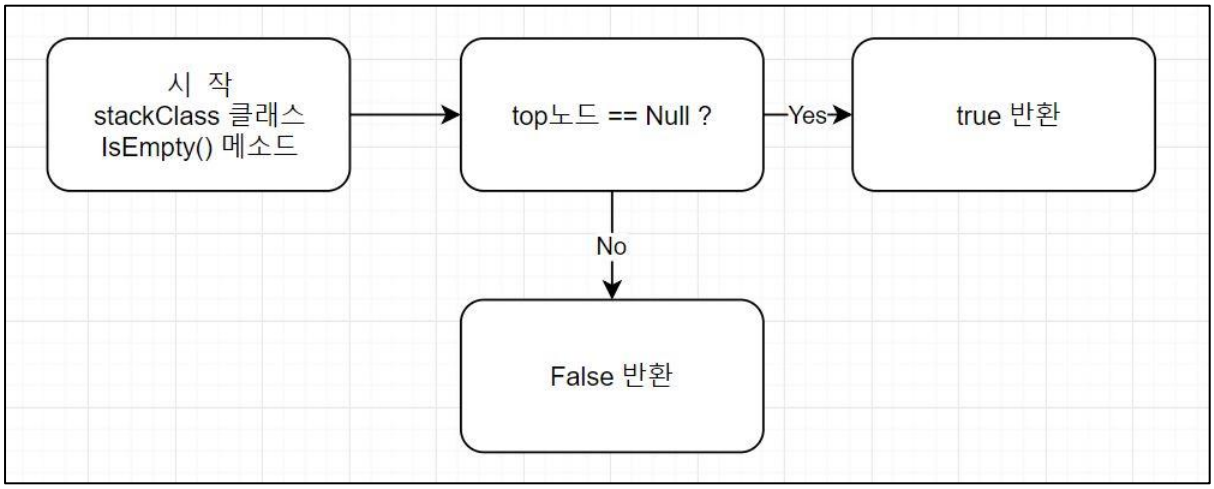

시작
mouse method
movePaint(int **map)
동적할당 된 맵 배열을
문자로 받는다.
txtcolor(14)
에너지, 좌표의 크기 미로의 오른쪽에 값 출력
쥐의 현재 위치 출력
종료

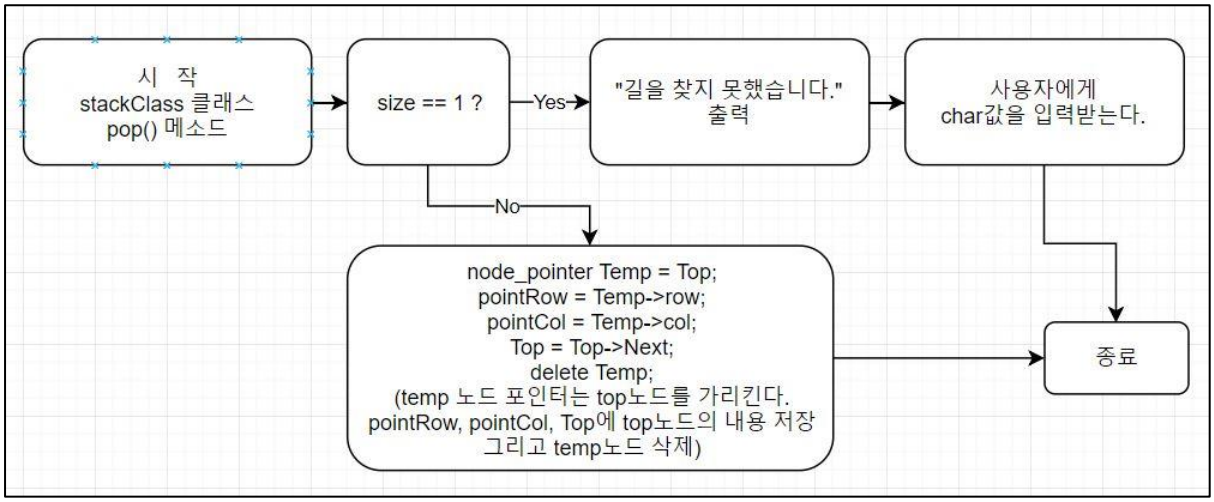

시 작
mouse method
strategy(int **map)
gotoxy함수를 이용해
마우스 커서 위치 현재 위치로 이동
유닉 문자 출력
lookAround(map)
현재 현재 위치에 대한
메모리 구조체 행렬의
crossroads를 1로 저장 ?
(이 현재 위치에 갈림 길등등의
갯수가 1개인 경우)
memory[m_row][m_col].crossroads
== 1 ?
yes
memory[m_row][m_col].trace = 1
point 구조체인 tel의 cnt에 갑을 teleport_count에 저장
save[tel]->cnt >= 15 ?
No
alreadyStack.Pop()
m_col =
alreadyStack.Top-
>col
m_row =
alreadyStack.Top-
>row
energy--
alreadyStack.size--
Yes
tmp_row = 1
tmp_col = 1
astar() ?
Yes
gotoxy(col * 2 + 2, 20 +
(++teleport_number))
보물
"보물 입력해 ..."
출력
gotoxy(col * 2 + 2, 20 +
(++teleport_number))
보물
사용자에게 값을 입력
받아 tprow, tpcol에
저장한다.
tprow, tpcol 의 좌표가 이로의 통로이며
유가 지나간 길이 아닌가?
Yes
teleport_count = 1
m_row = tprow
m_col = tpcol
gotoxy(m_col * 2,
m_row)
NO
gotoxy(m_col * 2,
m_row)
teleport_count = 1
NO
gotoxy(col * 2 + 2, 20 + (++teleport_number))
"갈 수 없는 곳이요"
출력
NO
Portlist
Portlist
Portlist
Portlist

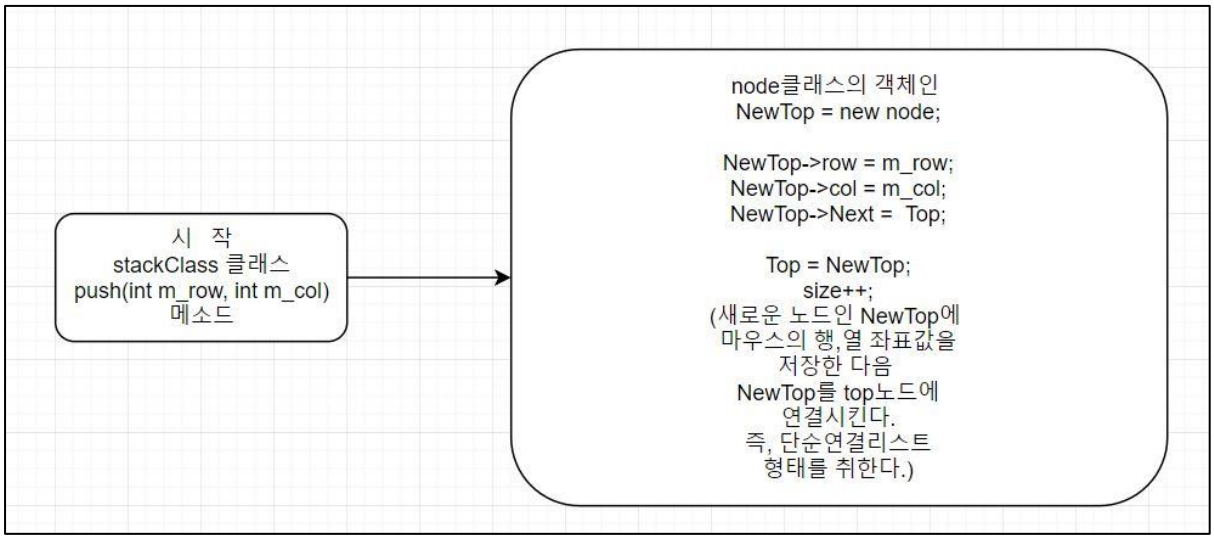

Portlist
Portlist2
Portlist4
memory[m_row][m_col+1].trace == 0 ?
elseif
memory[m_row+1][m_col].trace == 0 ?
elseif
memory[m_row][m_col-1].trace == 0 ?
elseif
m_row != 1 && m_row != 0 && memory[m_row-1][m_col].trace == 0 ?
1
2
3
4
메모리 구조체 행렬 trace의 값
1
④확인
1
③확인
ⓜ
1
①확인
1
②확인
①의 경우,
현재 쥐의 위치에 대한
메모리 구조체 행렬의 trace 값에 1을 저장
m_col ++
Push(m_row,m_col)(쥐의 현재위치를 스택에 저장)
Energy--
Teleport count ++
②의 경우,
현재 쥐의 위치에 대한
메모리 구조체 행렬의 trace 값에 1을 저장
m_row ++
Push(m_row,m_col)(쥐의 현재위치를 스택에 저장)
Energy--
Teleport count ++
③의 경우,
현재 쥐의 위치에 대한
메모리 구조체 행렬의 trace 값에 1을 저장
m_col --
Push(m_row,m_col)(쥐의 현재위치를 스택에 저장)
Energy--
Teleport count ++
④의 경우,
현재 쥐의 위치에 대한
메모리 구조체 행렬의 trace 값에 1을 저장
m_row --
Push(m_row,m_col)(쥐의 현재위치를 스택에 저장)
Energy--
Teleport count ++
lookAround(map)
종 료

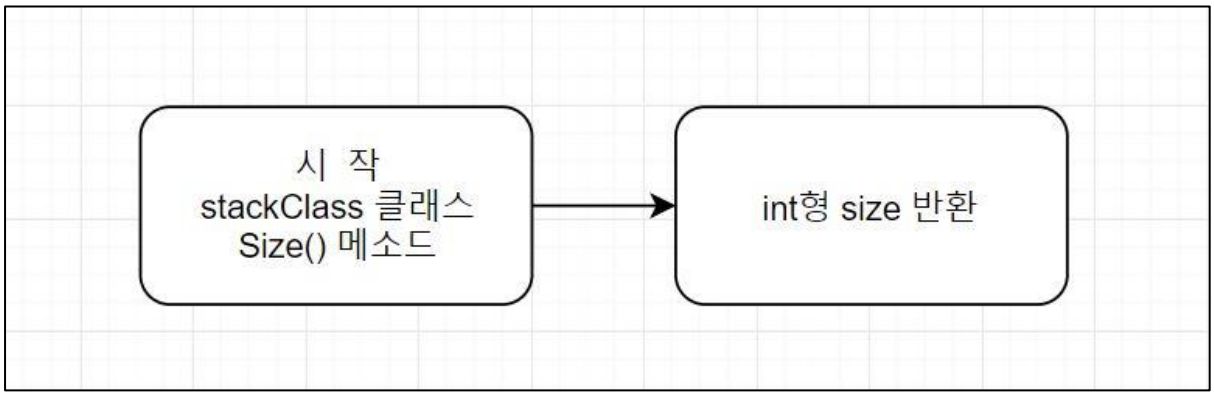
Stack


출 발
stack 클래스
생성자
인자를 받지 않는 경우
Top = NULL;
size = 0;
stack 클래스
객체 주소 받는 경우
종 료

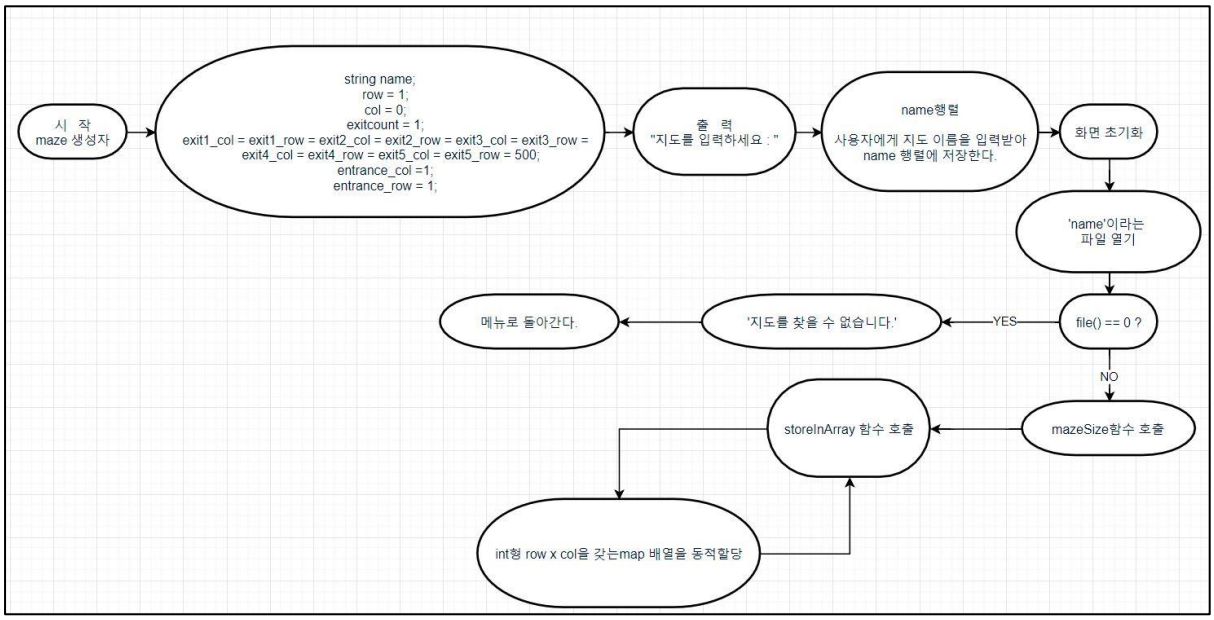

시 작
stackClass 클래스
IsEmpty() 메소드
top노드 == Null ?
Yes
true 반환
No
False 반환

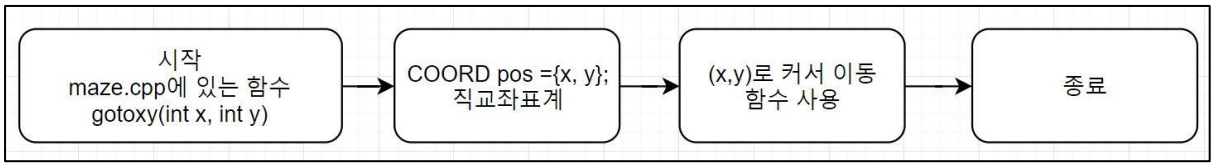

시 작
stackClass 클래스
pop() 메소드
size == 1 ?
Yes
"길을 찾지 못했습니다."
출력
사용자에게
char값을 입력받는다.
No
node_pointer Temp = Top;
pointRow = Temp->row;
pointCol = Temp->col;
Top = Top->Next;
delete Temp;
(temp 노드 포인터는 top노드를 가리킨다.
pointRow, pointCol, Top에 top노드의 내용 저장
그리고 temp노드 삭제)
종료

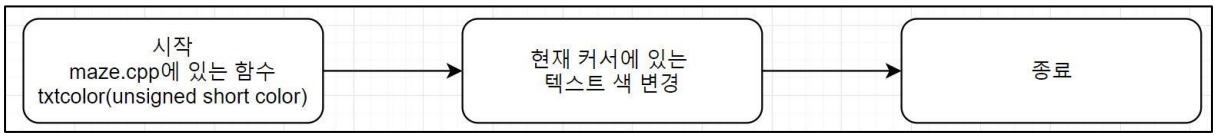

시 작
stackClass 클래스
push(int m_row, int m_col)
메소드
node클래스의 객체인
NewTop = new node;
NewTop->row = m_row;
NewTop->col = m_col;
NewTop->Next = Top;
Top = NewTop;
size++;
(새로운 노드인 NewTop에
마우스의 행,열 좌표값을
저장한 다음
NewTop를 top노드에
연결시킨다.
즉, 단순연결리스트
형태를 취한다.)

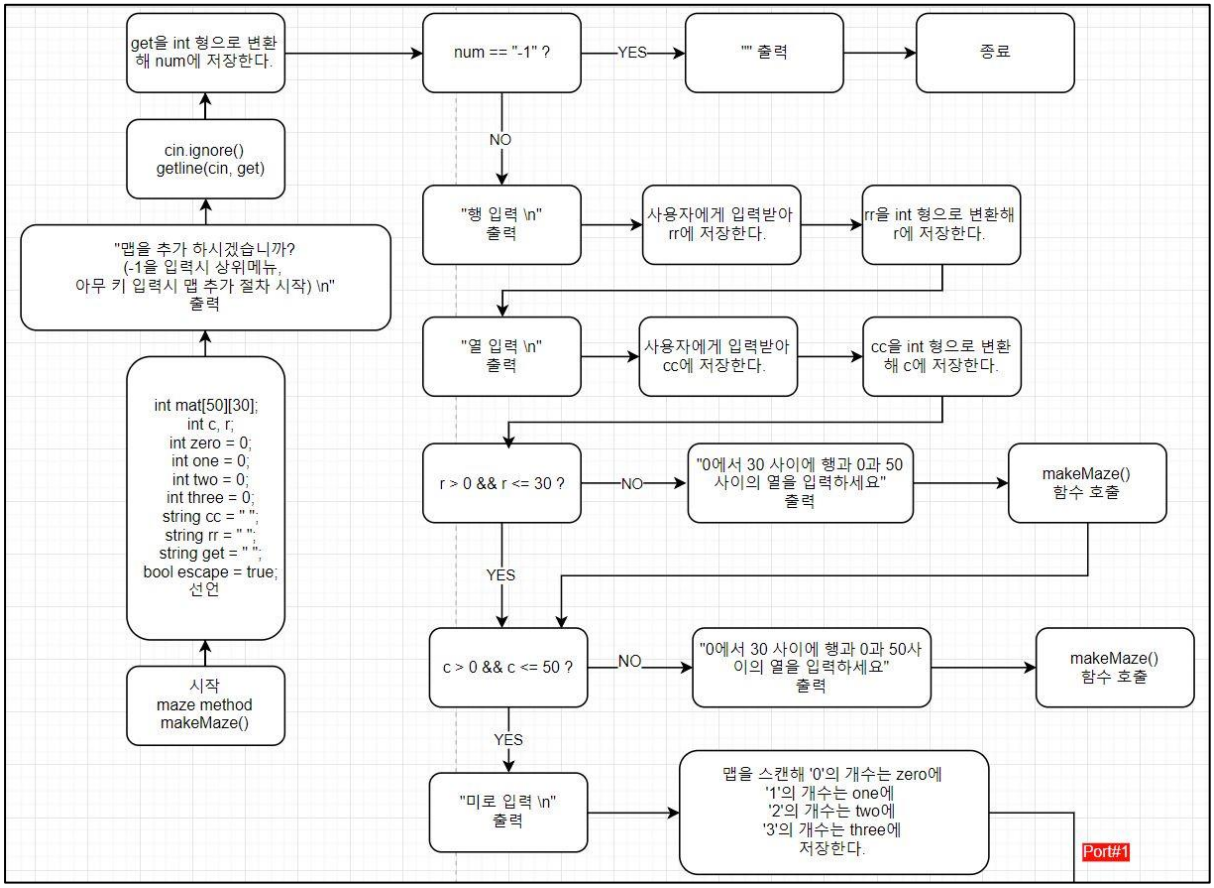

시 작
stackClass 클래스
Size() 메소드
int형 size 반환

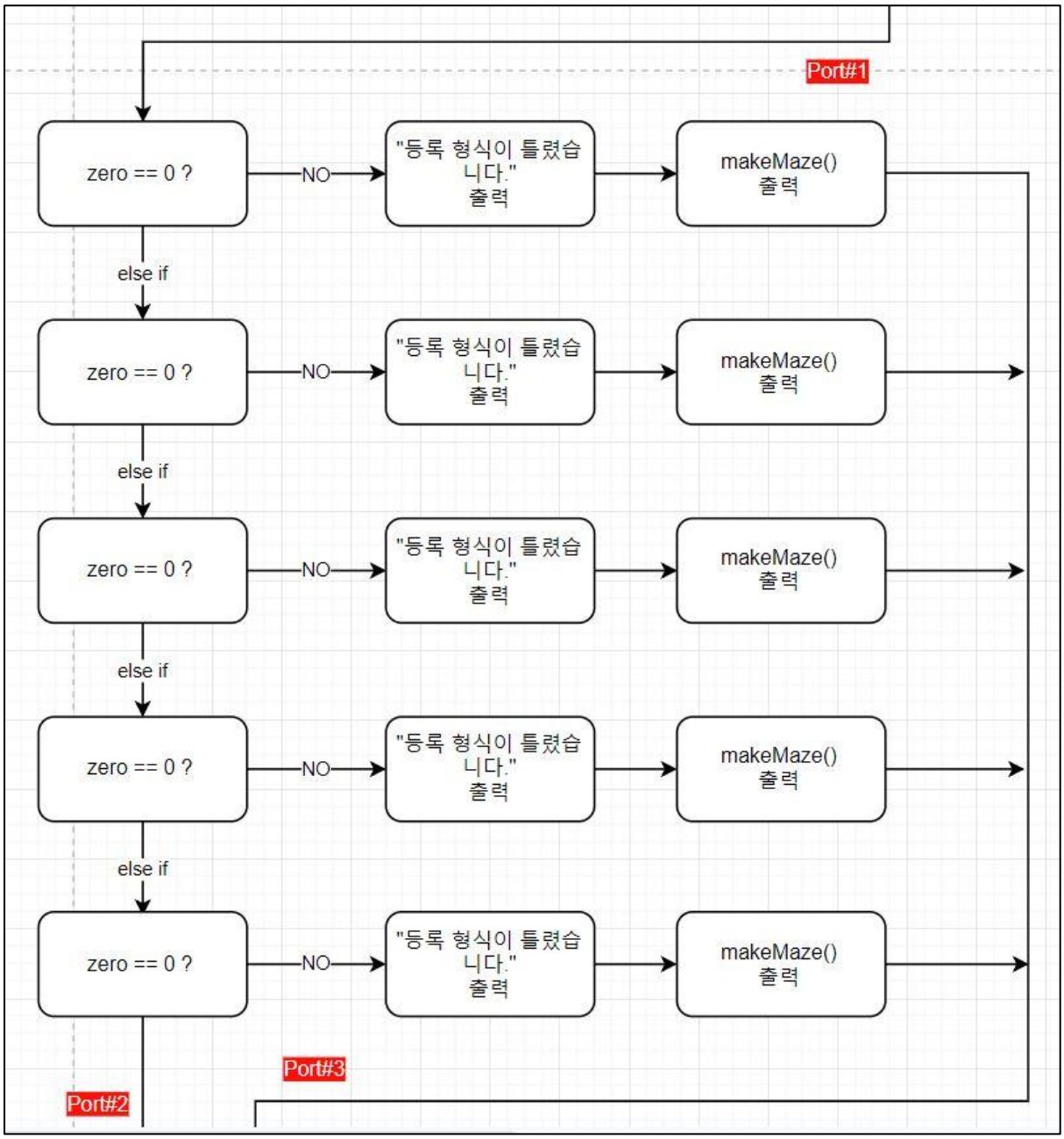
Maze

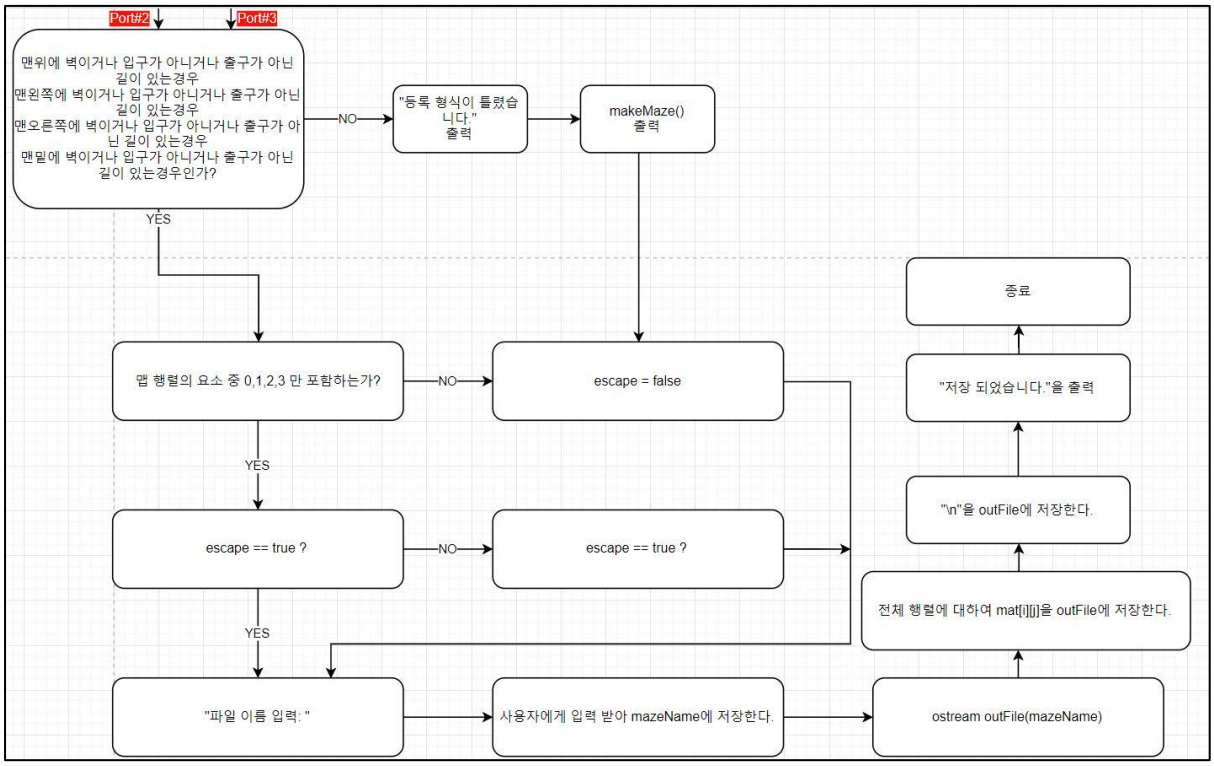

시 작
maze 생성자
string name;
row = 1;
col = 0;
exitcount = 1;
exit1_col = exit1_row = exit2_col = exit2_row = exit3_col = exit3_row =
exit4_col = exit4_row = exit5_col = exit5_row = 500;
entrance_col =1;
entrance_row = 1;
출 력
"지도를 입력하세요 : "
name행렬
사용자에게 지도 이름을 입력받아
name 행렬에 저장한다.
화면 초기화
'name'이라는
파일 열기
file() == 0 ?
YES
'지도를 찾을 수 없습니다.'
메뉴로 돌아간다.
NO
mazeSize함수 호출
storeInArray 함수 호출
int형 row x col을 갖는map 배열을 동적할당

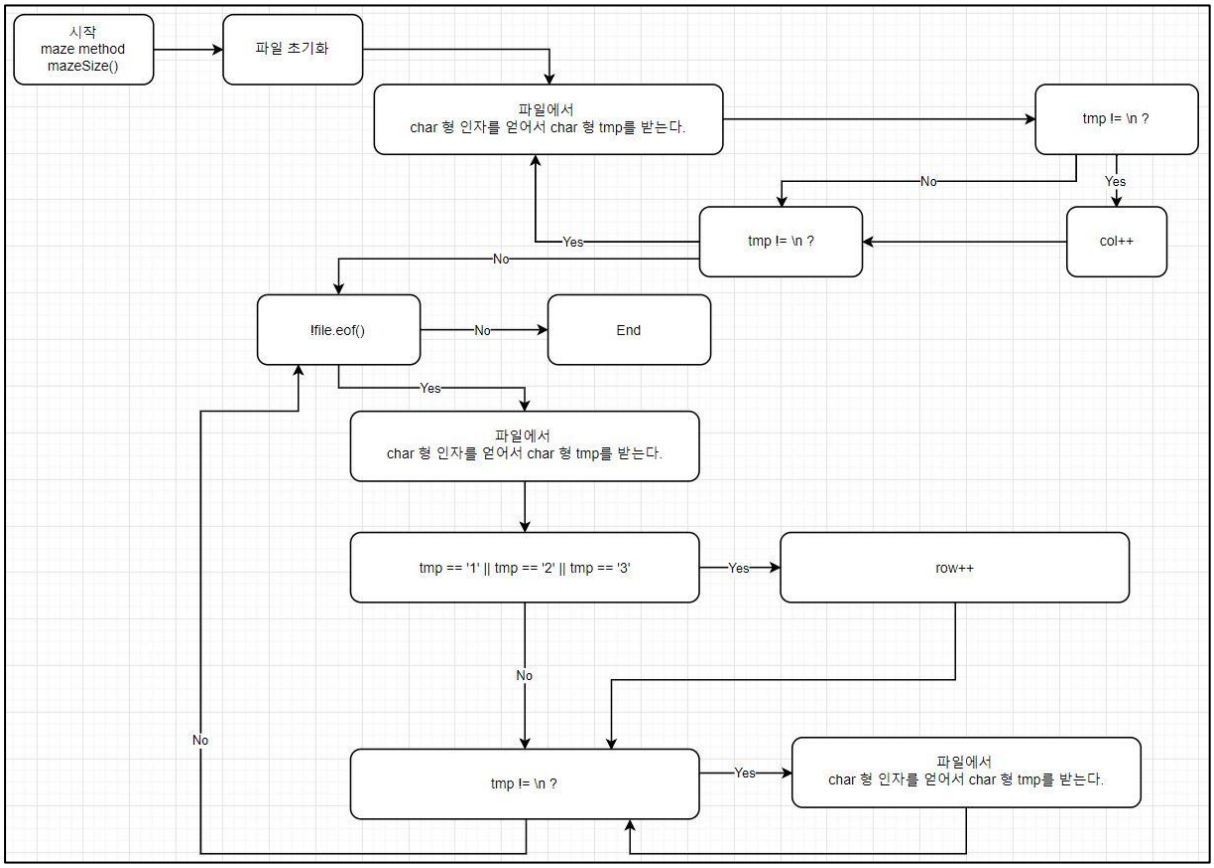

시작
maze.cpp에 있는 함수
gotoxy(int x, int y)
COORD pos ={x, y};
직교좌표계
(x,y)로 커서 이동
함수 사용
종료

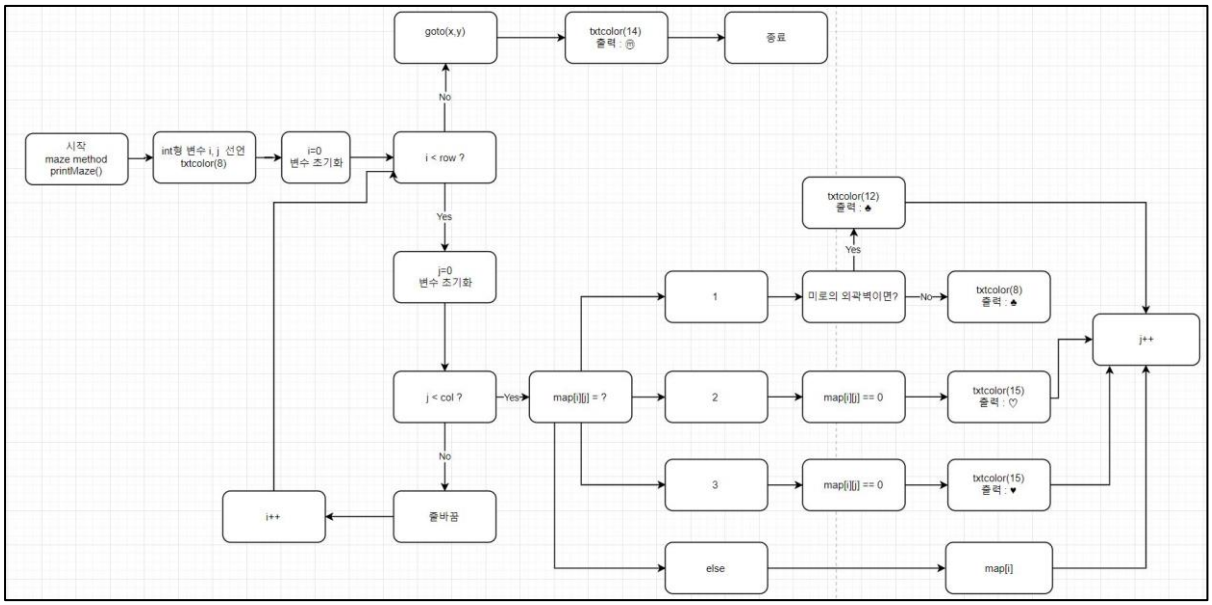

시작
maze.cpp에 있는 함수
txtcolor(unsigned short color)
현재 커서에 있는
텍스트 색 변경
종료

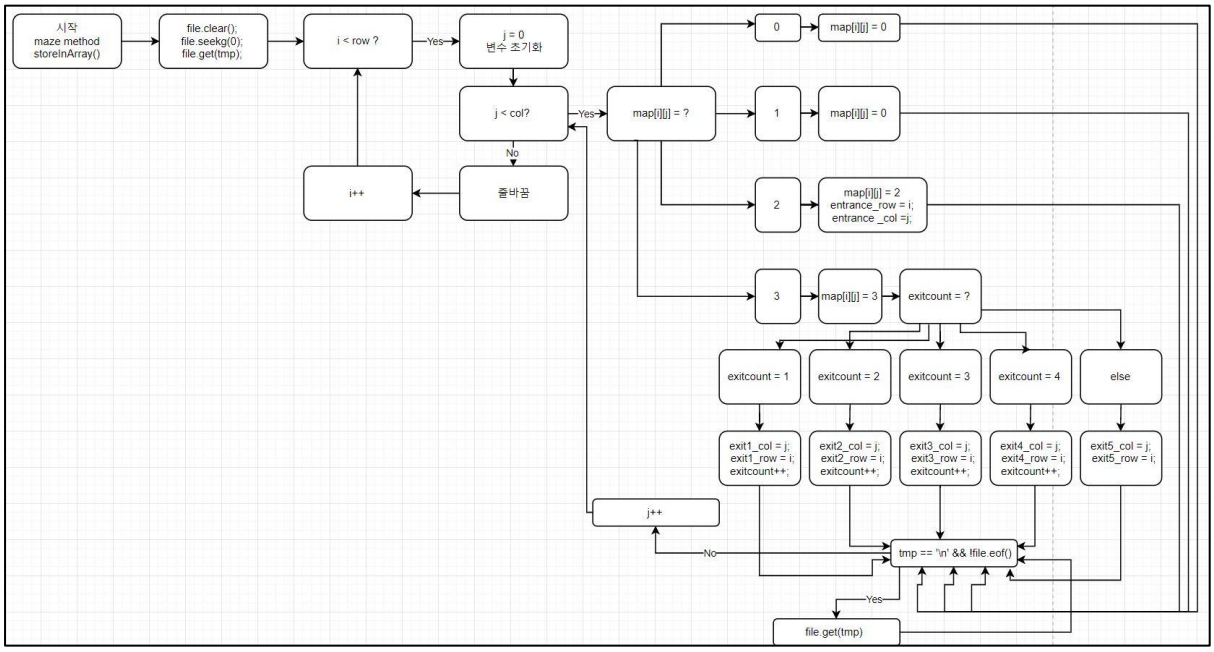

시작
maze method
makeMaze()
int mat[50][30];
int c, r;
int zero = 0;
int one = 0;
int two = 0;
int three = 0;
string cc = " ";
string rr = " ";
string get = " ";
bool escape = true;
선언
"맵을 추가 하시겠습니까?
(-1을 입력시 상위메뉴,
아무 키 입력시 맵 추가 절차 시작) \n"
출력
cin.ignore()
getline(cin, get)
get을 int 형으로 변환
해 num에 저장한다.
num == "-1" ?
YES
"" 출력
종료
NO
"행 입력 \n"
출력
사용자에게 입력받아
rr에 저장한다.
rr을 int 형으로 변환해
r에 저장한다.
"열 입력 \n"
출력
사용자에게 입력받아
cc에 저장한다.
cc을 int 형으로 변환
해 c에 저장한다.
r > 0 && r <= 30 ?
NO
"0에서 30 사이에 행과 0과 50
사이의 열을 입력하세요"
출력
makeMaze()
함수 호출
YES
c > 0 && c <= 50 ?
NO
"0에서 30 사이에 행과 0과 50사
이의 열을 입력하세요"
출력
makeMaze()
함수 호출
YES
"미로 입력 \n"
출력
맵을 스캔해 '0'의 개수는 zero에
'1'의 개수는 one에
'2'의 개수는 two에
'3'의 개수는 three에
저장한다.
Port#1


Port#1
zero == 0 ?
NO
"등록 형식이 틀렸습니다." 출력
makeMaze() 출력
else if
zero == 0 ?
NO
"등록 형식이 틀렸습니다." 출력
makeMaze() 출력
else if
zero == 0 ?
NO
"등록 형식이 틀렸습니다." 출력
makeMaze() 출력
else if
zero == 0 ?
NO
"등록 형식이 틀렸습니다." 출력
makeMaze() 출력
else if
zero == 0 ?
NO
"등록 형식이 틀렸습니다." 출력
makeMaze() 출력
Port#3
Port#2


Port#2
Port#3
맨위에 벽이거나 입구가 아니거나 출구가 아닌 길이 있는경우
맨왼쪽에 벽이거나 입구가 아니거나 출구가 아닌 길이 있는경우
맨오른쪽에 벽이거나 입구가 아니거나 출구가 아닌 길이 있는경우
맨밑에 벽이거나 입구가 아니거나 출구가 아닌 길이 있는경우인가?
NO
"등록 형식이 틀렸습니다." 출력
makeMaze() 출력
YES
맵 행렬의 요소 중 0,1,2,3 만 포함하는가?
NO
escape = false
YES
escape == true ?
NO
escape == true ?
YES
"파일 이름 입력: "
사용자에게 입력 받아 mazeName에 저장한다.
ostream outFile(mazeName)
전체 행렬에 대하여 mat[i][j]을 outFile에 저장한다.
"\n"을 outFile에 저장한다.
"저장 되었습니다."을 출력
종료


시작
maze method
mazeSize()
파일 초기화
파일에서
char 형 인자를 얻어서 char 형 tmp를 받는다.
tmp != \n ?
No
Yes
col++
tmp != \n ?
Yes
No
!file.eof()
No
End
Yes
파일에서
char 형 인자를 얻어서 char 형 tmp를 받는다.
tmp == '1' || tmp == '2' || tmp == '3'
Yes
row++
No
No
tmp != \n ?
Yes
파일에서
char 형 인자를 얻어서 char 형 tmp를 받는다.


시작
maze method
printMaze()
int형 변수 i, j 선언
txtcolor(8)
i=0
변수 초기화
i < row ?
No
goto(x,y)
txtcolor(14)
출력 : ㉿
종료
Yes
j=0
변수 초기화
j < col ?
Yes
map[i][j] = ?
1
미로의 외곽벽이면?
Yes
txtcolor(12)
출력 : ♠
No
txtcolor(8)
출력 : ♣
2
map[i][j] = 0
txtcolor(15)
출력 : ♡
3
map[i][j] = 0
txtcolor(15)
출력 : ♥
else
map[i]
j++
No
줄바꿈
i++


시작
maze method
storeInArray()
file.clear();
file.seekg(0);
file.get(tmp);
i < row ?
Yes
j = 0
변수 초기화
j < col?
Yes
No
줄바꿈
i++
map[i][j] = ?
0
map[i][j] = 0
1
map[i][j] = 0
2
map[i][j] = 2
entrance_row = i;
entrance_col =j;
3
map[i][j] = 3
exitcount = ?
exitcount = 1
exitcount = 2
exitcount = 3
exitcount = 4
else
exit1_col = j;
exit1_row = i;
exitcount++;
exit2_col = j;
exit2_row = i;
exitcount++;
exit3_col = j;
exit3_row = i;
exitcount++;
exit4_col = j;
exit4_row = i;
exitcount++;
exit5_col = j;
exit5_row = i;
j++
No
tmp == '\n' && !file.eof()
Yes
file.get(tmp)

3. 설계서 작성 시 발견한 기획서 추가 보완 사항

- 맵 만들기에서 파일 이름 크기 제한을 추가시켜야 하여 야 한다. 즉, char[20]을 사용하여 파일 이름을 만들게 제한한다.
- 맵 추가 기능에서 미로 만들 시 행렬의 크기는 최소 (3,3) 행렬 이상이 되야 한다.
- 텍스트 파일로 맵을 읽어 올 시 행렬의 크기는 최소 (3,3) 행렬 이상이 되야 한다.