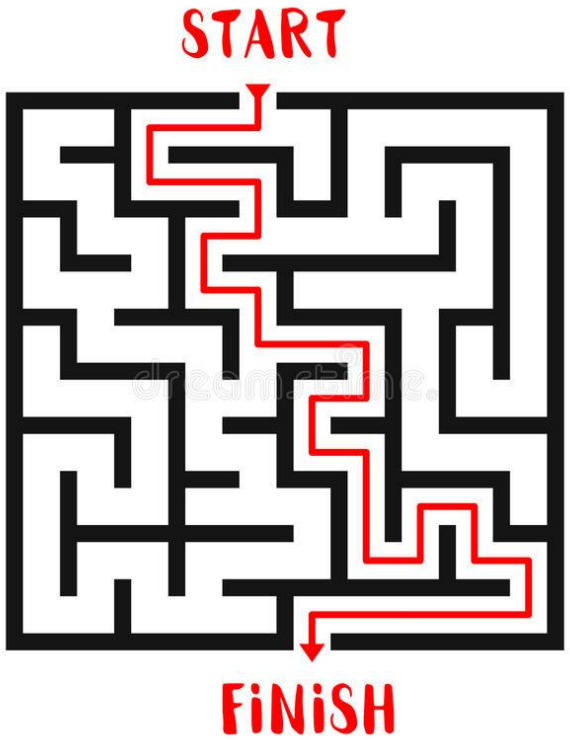
심화프로그래밍
프로젝트 1

2018학년도 1학기

미로 찾기 게임



- 시작 지점부터 도착 지점까지 도달하는 것이 목표.
- 미로의 전체 맵이 주어질 때 도착 지점까지의 경로를 찾는 것.

우리가 할 미로 찾기 게임

- 시작 지점부터 도착 지점까지 도달하는 것이 목표인 것은 동일.
- 그러나 미로의 전체 맵은 주어지지 않는다!
- “상 하 좌 우” 로 직접 이동해 보면서 도착 지점까지 도달해야 한다.

우리가 할 미로 찾기 게임

- 목표: 도착 지점에 도달하기.
- 플레이 방법: 턴 방식
- 한 턴에 이루어지는 행동:

1. 플레이어는 이동할 방향을 정한다. (상, 하, 좌, 우 중에서 하나)
2. 이동하려는 방향에 장애물이 있으면 플레이어에게 이동 불가능한 것을 알려줌.
3. 이동하려는 방향으로 이동이 가능한 경우 이동 후의 좌표를 플레이어에게 알려줌.

프로젝트 내용

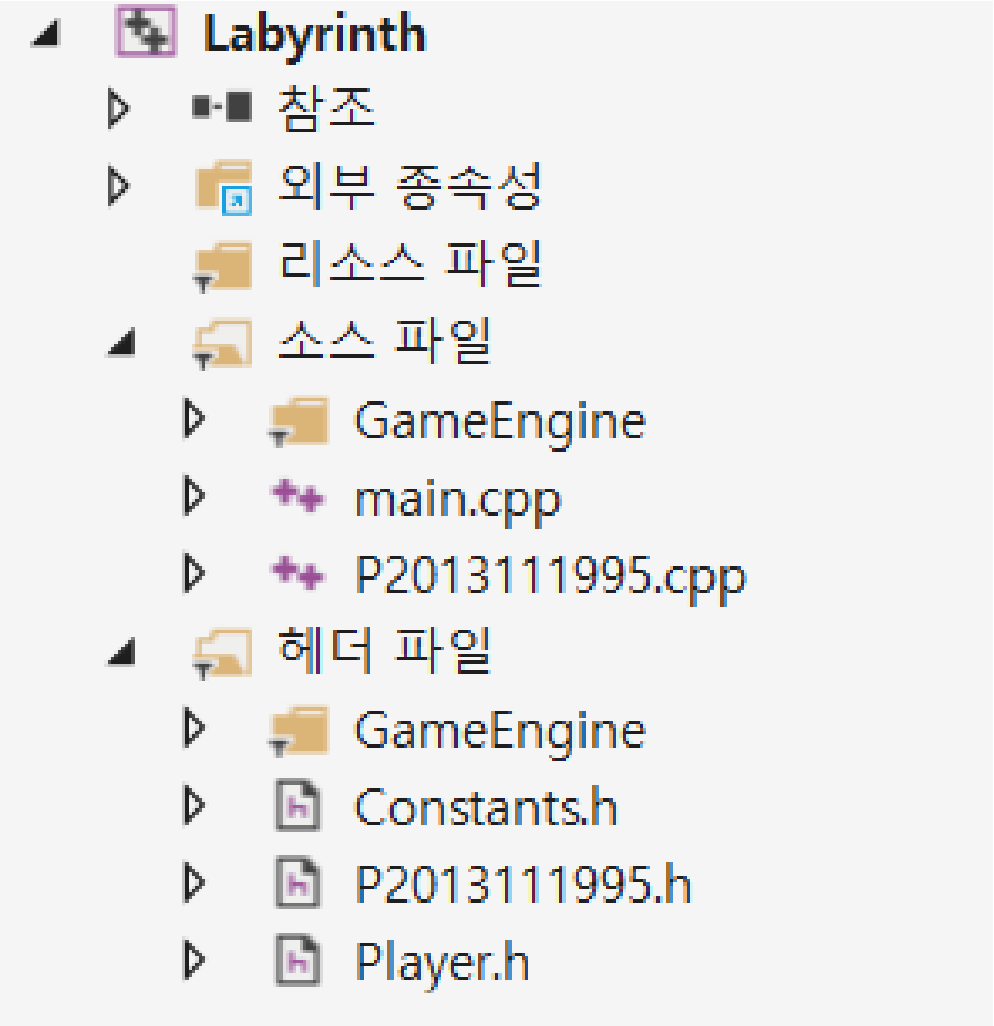
- 미로 찾기의 전체적인 틀은 구현되어 있습니다.
- 앞의 룰에 따라 **도착지점까지 도달할 수 있는 코드**를 작성하는 것이 목표입니다!

- **간단한 AI를 구현**하여 챔피언에 도전해봅시다! (미로의 도착지점에 빨리 도달할 수록 좋은 코드)
- **본인의 생각**을 구현해내야 합니다.
- 실제 제출한 과제로 대회가 치러집니다.

코드를 알아보기에 앞서

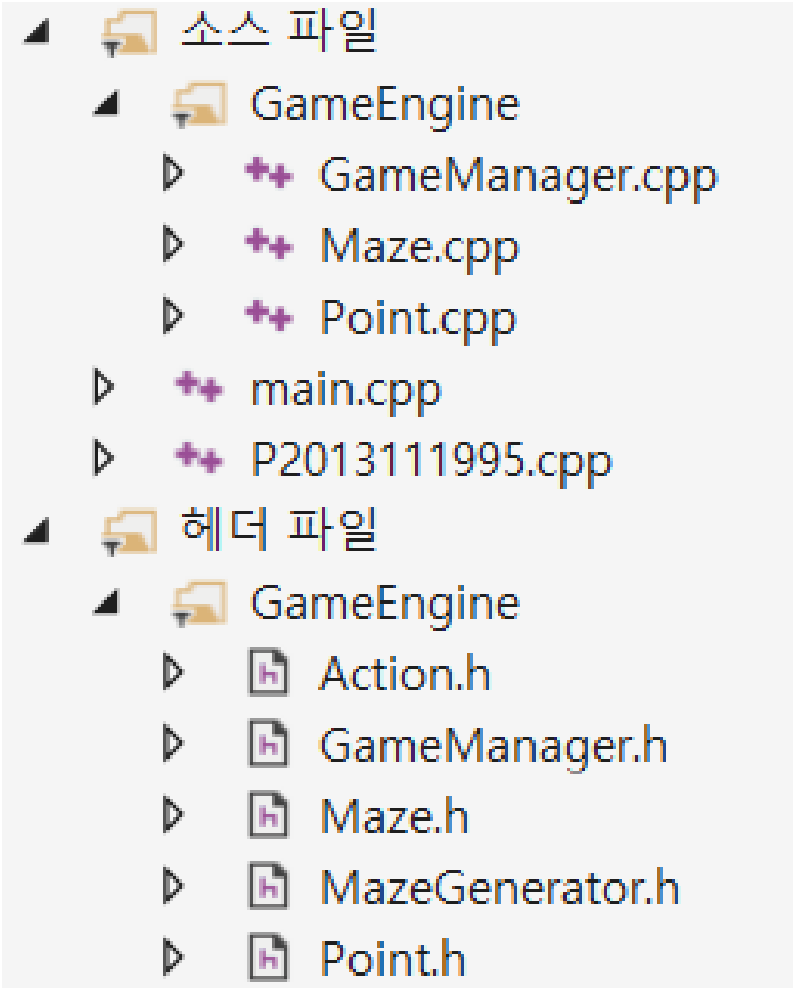
- 전체 코드를 이해할 필요는 없습니다.
- 전체를 몰라도 **어떤 기능을 구현해야 하는지 명확히** 알면 됩니다.
- 내 자신이 미로에 갇혔다면 어떻게 탈출할지 잘 고려해보세요.
- 어떻게 하면 **조금 더 효율적일지** 생각해보세요.

프로그램 구조



- 크게 두 부류로 나뉜다.
- GameEngine과 Player
- 과제로 구현할 것은 Player를 상속받은 **본인의 Player**입니다.

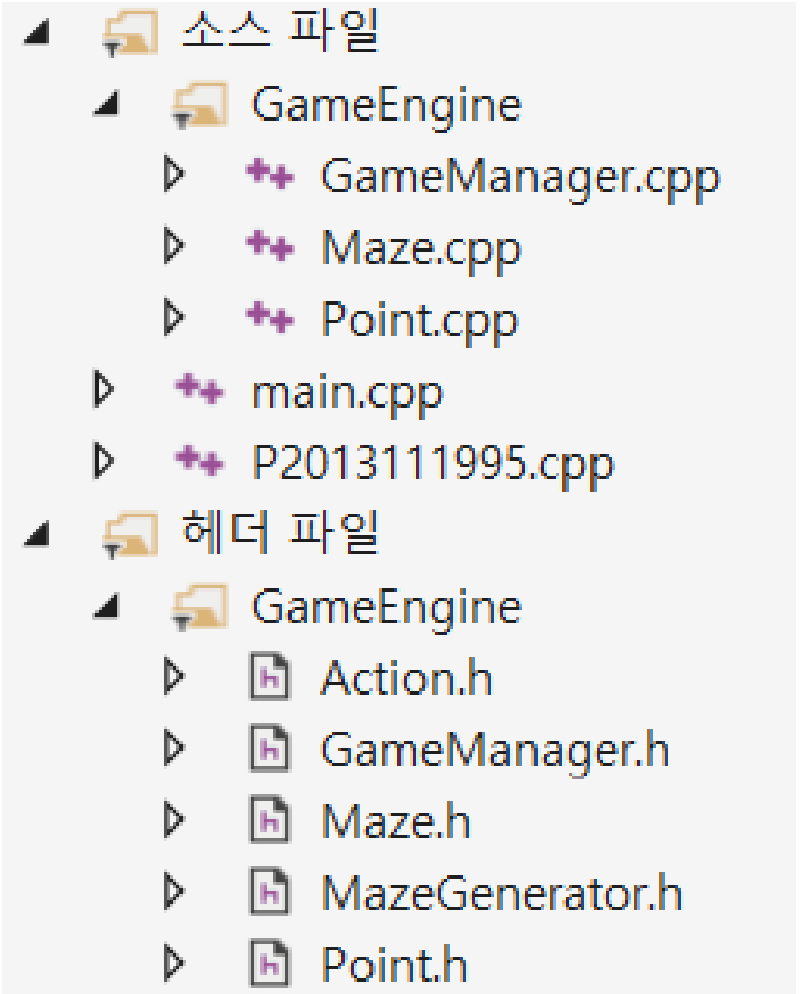
GameEngine



- GameManager: 게임 전체적인 것을 담당.
- main: 메인 함수, GameManager 호출
- Maze: 게임 중 미로 맵에 대한 것을 담당
- Point: 좌표를 나타내는 클래스.

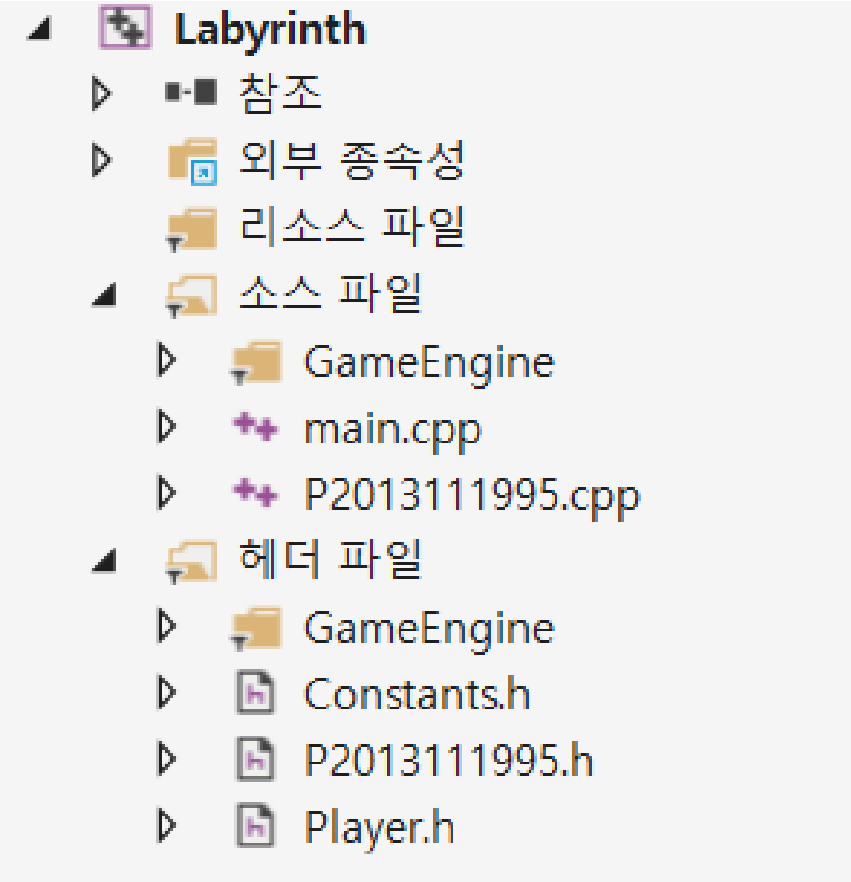
- Action: 상 하 좌 우로 이동 할 때의 enum
- MazeGenerator: 미로를 랜덤으로 생성.

GameEngine



- 코드를 자세히 알 필요는 없습니다.
- 어떤 식으로 돌아가는지만 알면 됩니다.

Player



- Player는 실제 게임을 플레이하는 사용자에 대한 내용입니다.
- 과제는 Player.h를 상속받아 자신만의 플레이어 클래스를 생성하는 것입니다.

Action.h

```
// 다음에 행동할 Action.
// MOVE_LEFT: 좌로 이동
// MOVE_RIGHT: 우로 이동
// MOVE_UP: 위로 이동
// MOVE_DOWN: 아래로 이동
enum Action
{
    MOVE_LEFT, MOVE_RIGHT, MOVE_UP, MOVE_DOWN
};
```

- 플레이어가 행동할 액션에 대한 enum입니다.

GameManager.h

```
class GameManager
{
public:
    // 게임 초기화
    // -> 미로 생성
    void init();
    // player 가 실제 게임 플레이 진행
    // -> 플레이어가 이동할 액션을 선택, 그에 따른 반응을 호출.
    // -> 도착지점을 일정 턴 안에 도달 시 게임 클리어
    void run(Player *player);

private:
    // 미로
    Maze maze;
};
```

- 전체적인 게임 플레이를 담당합니다.

Maze.h

```
// 벽(WALL)인지 공간(SPACE)인지 골 지점(GOAL)인지를 나타내는 enum
enum Cell
{
    WALL, SPACE, GOAL
};

class Maze
{
public:
    // 미로의 크기, (높이 너비)
    // 상수 값은 Constants.h에 선언됨
    static const int WIDTH = MAZE_WIDTH;
    static const int HEIGHT = MAZE_HEIGHT;
```

Maze.h

```
    // 미로 출력, 플레이어 위치도 받아서 출력
    void printMaze(const Point& player) const;

    // 미로를 랜덤으로 생성
    void generateMaze();

    // 시작점 반환
    Point getStartPoint() const;

    // 해당 좌표에 이동 가능한가
    bool ableToMove(const Point& point) const;

    // 해당 좌표가 목표 지점인가
    bool isGoal(const Point& point) const;

private:
    // 실제 미로 맵
    Cell map[HEIGHT][WIDTH];
    // 시작점
    Point startPoint;
};
```

- 생성한 미로 맵을 관리합니다.

MazeGenerator.h

- 제가 작성한 코드가 아닙니다.
- 대략적으로는 DFS 방법을 이용해서 미로를 랜덤으로 생성합니다.
- 출처: https://codereview.stackexchange.com/questions/135443/c-maze-generator

Point.h

```
class Point
{
public:
    // 생성자
    Point(int x = 0, int y = 0);

    // getter
    int getX() const;
    int getY() const;

    // 출력 편하게 하기 위함
    friend std::ostream& operator<<(std::ostream& os, const Point& p);

    // 대입 및 비교 편하게 하기 위한 오버라이딩
    Point& operator = (const Point &p);
    bool operator == (const Point &p) const;
    bool operator != (const Point &p) const;

private:
    int x, y;
};
```

- 좌표 (x,y)를 표현하는 클래스입니다.

Constants.h

```
// 상수들. 자유롭게 조정 가능하지만 제출 및 채점 시에는 원래 값으로 진행합니다.
// MAZE_WIDTH, MAZE_HEIGHT는 반드시 5 이상의 홀수여야 합니다.
// 원래 값: 51, 21, 100000

// 미로의 너비
const int MAZE_WIDTH = 51;
// 미로의 높이
const int MAZE_HEIGHT = 21;
// 최대 플레이 가능한 턴 수.
const int MAX_TURN = 100000;
```

- 게임 플레이 시 사용하는 상수가 들어 있습니다.

- 테스트를 위해 값을 바꾸는 것은 가능하나 과제 채점 시에는 디폴트 값으로 진행합니다.

Player.h

```
class Player
{
public:
    Player() {}

    // 게임 처음 시작 시 호출됩니다.
    // 시작 플레이어 위치(매번 달라집니다), 미로의 높이와 너비가 매개변수로 들어옵니다.
    virtual void gameStart(Point player, int height, int width) = 0;

    // 다음에 "움직일 방향"을 반환하세요.
    // 해당 방향으로 한 칸 이동 시도 후 아래 두 함수 중 하나가 호출됩니다.
    //
    // Action enum안에는 { MOVE_LEFT, MOVE_RIGHT, MOVE_UP, MOVE_DOWN } 가 들어 있습니다.
    // ex) return Action::MOVE_LEFT; // 왼쪽으로 이동
    virtual Action nextMove() = 0;

    // 다음에 움직일 방향에 벽이 없으면 이 함수가 호출됩니다.
    // 입력 매개변수로 "움직인 후의 좌표", "이동 전의 좌표"가 들어옵니다.
    // ex) (3, 4)에서 MOVE_LEFT 한다면 point는 (3, 3) 입니다.
    virtual void ableToMove(Point point, Point prevPoint) = 0;

    // 다음에 움직일 방향에 벽이 있으면 이 함수가 호출됩니다.
    // 입력 매개변수로 "현재 위치의 좌표"가 들어옵니다.
    // ex) (3, 4)에서 MOVE_LEFT 해도 point는 (3, 4) 입니다.
    virtual void notAbleToMove(Point point) = 0;
};
```

- Player 클래스를 **상속받아** 내부를 구현해야 합니다.

P2013111995.h

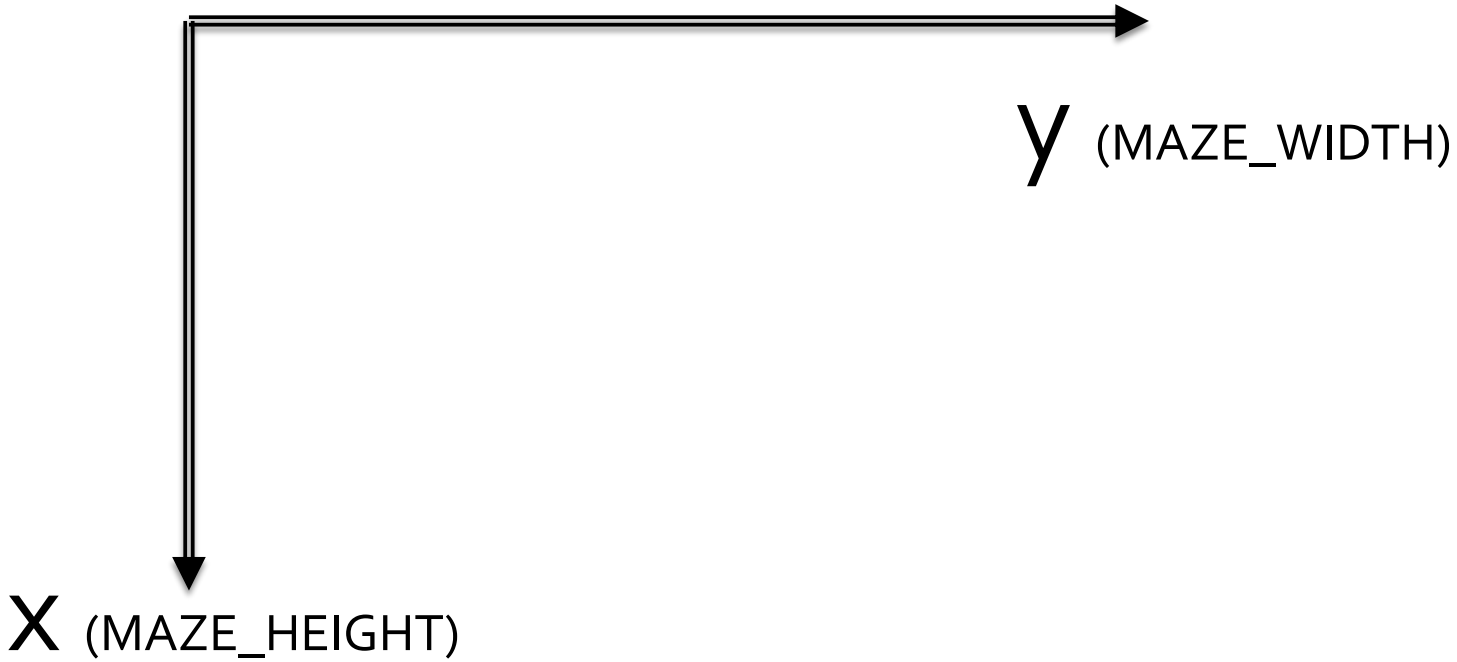
```
class P2013111995 : public Player
{
public:
    P2013111995();
    void gameStart(Point player, int width, int height);
    Action nextMove();
    void ableToMove(Point point);
    void notAbleToMove(Point point);

private:
    Point point;
};
```

- 이런 식으로 Player를 상속받아 자신만의 플레이어를 구현하면 됩니다.

주의사항!

- 본 미로 게임의 좌표는 **아래 방향이 +x축, 오른쪽 방향이 +y축** 입니다.
- **2차원 배열**을 생각하면 쉽습니다. maze[x][y]



주의사항!!

- **최종 과제 제출 시 입출력은 금지합니다**. (cin, cout, scanf, printf, 등)
- (소스코드 작성 중에 디버깅을 위한 입출력은 가능함.)

- 클래스 이름, 헤더파일 이름, 소스파일 이름은 "**P자신의학번**"으로 합니다.
- (ex, 학번이 2013111995일 경우 "P2013111995")

- 헤더 파일과 cpp 파일을 나눠서 둘 다 제출해 주세요.

과제 제출

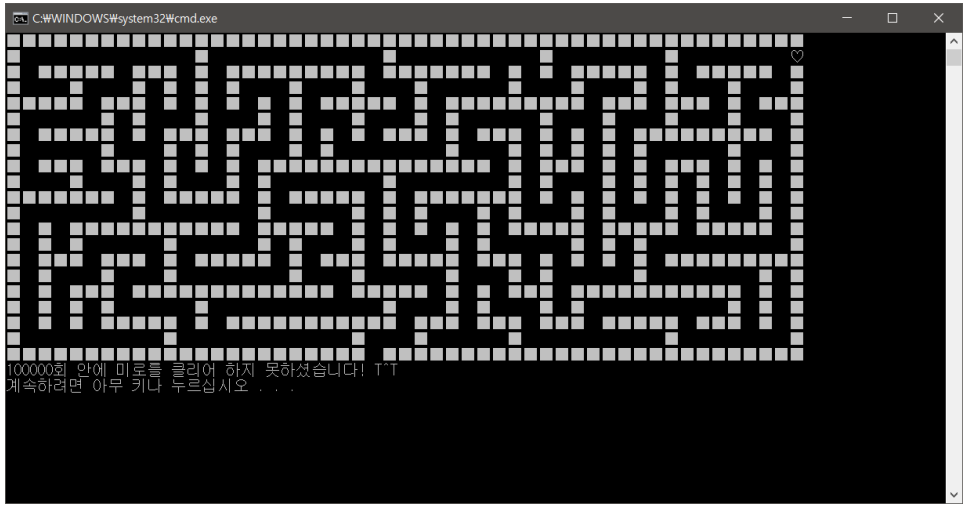
- 자신이 작성한 **헤더파일, cpp 파일(주석 포함)과 간단한 보고서를 제출하세요.**

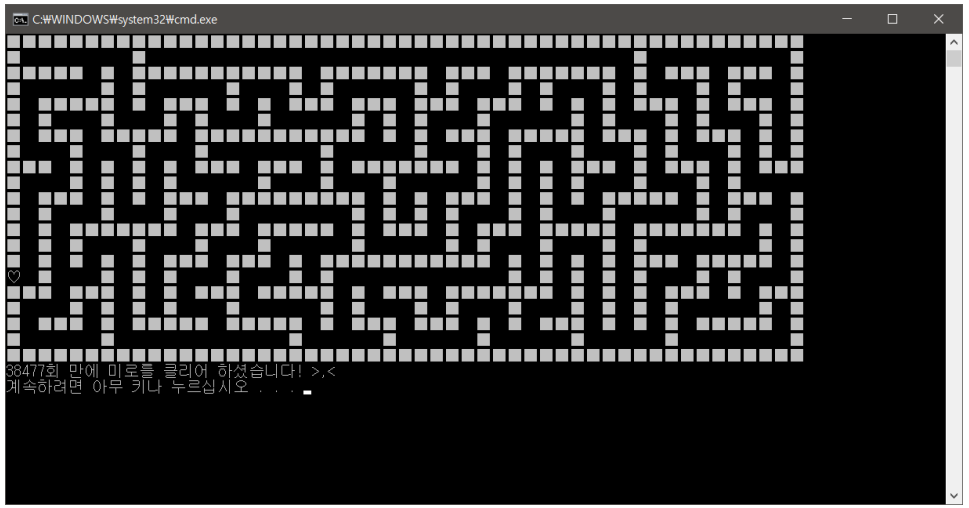
- 주석은 함수 위에 어떤 생각으로 어떻게 구현하였는지 적으세요.
- (ex, /*이러이러해서 저렇게 함*/)

- 보고서에는 각 함수별로 단 주석을 그대로 적으면 됩니다.
- (ex, 이러이러해서 저렇게 함)

- 추가로 **실행한 결과 화면**을 6개 캡처하여 넣습니다. (프로그램을 6번 돌리면 됩니다.)

실행 화면 예시





- **랜덤으로 4방향을 이동**하는 코드 실행 결과.
- 가끔 100000 턴 안에 클리어가 불가능하지만 클리어가 되기는 한다!
- 하지만 매우 효율이 좋지 못함.
- 잘 생각해서 **빠른 턴 안에 미로를 클리어**해 보세요!
- 만약 실행 시 미로 출력이 이상하다면 콘솔 창 크기를 키워보세요

질문 사항이 있다면

- 조교 이메일
- zeikar@naver.com

- 연구실
- 신공학관 5112호

- 감사합니다.