


# C++프로그래밍

## 프로젝트

프로젝트 명	Snake Game
팀 명	10팀
문서 제목	최종보고서 – Snake Game

Version	1.5
Date	2024-06-16

팀원	김민재 (20212970)
	윤성욱(20213029)


 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

이 문서에 포함되어 있는 정보는 국민대학교 소프트웨어융합대학 소프트웨어학부 및 소프트웨어학부 개설 교과목 C++프로그래밍 수강 학생 중 프로젝트 “Snake game”를 수행하는 팀 “10팀”의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 “10팀”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다.

## 문서 정보 / 수정 내역


Filename	최종보고서-Snake Game.doc
원안작성자	김민재, 윤성욱
수정작업자	김민재, 윤성욱

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2024-05-26	윤성욱	1.0		개요 작성 및 목표 작성
2024-05-26	김민재	1.1		개발 내용 및 시스템 구조 작성
2024-05-26	윤성욱	1.2		시스템 구조 작성
2024-06-16	김민재	1.3		활용 기술 및 제한 요소, 해결 방안 작성
2024-06-16	윤성욱	1.4		전체적인 수정 및 자기 평가 작성
2024-06-16	김민재	1.5		자기 평가 및 부록 작성

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

## 목 차

1	개요	4
2	개발 내용 및 결과물	5
2.1	목표	5
2.2	개발 내용 및 결과물	6
2.2.1	개발 내용	6
2.2.2	시스템 구조 및 설계도	6
2.2.3	활용/개발된 기술	6
2.2.4	현실적 제한 요소 및 그 해결 방안	6
2.2.5	결과물 목록	7
3	자기평가	8
4	참고 문헌	8
5	부록	8
5.1	사용자 매뉴얼	8
5.2	설치 방법	8

 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

# 1 개요

## 평가기준 (10점)

프로젝트를 완성하기 위해 사용한 개발 방법을 기술하세요.

또한 사용하고 있는 외부 라이브러리와 해당 라이브러리를 획득/설치하는 방법을 기술하세요.

이 프로젝트는 터미널 그래픽 라이브러리인 ncurses를 사용했다. 해당 외부 라이브러리를 설치하기 위해 MacOS 기준, homebrew의 brew update 명령어를 통해 패키지를 업데이트하고, brew install ncurses 명령어를 이용해 패키지를 내려받아 설치하였다. 이후, 각 소스파일에는 #include <ncurses.h>를 작성해 코드에서 ncurses 라이브러리를 사용할 수 있도록 적용하였다.

게임의 기능을 세분화해 여러 클래스와 내부의 여러 함수들로 구성하였고, 각자가 맡아 구현한 코드를 보다 효율적으로 공유하고 사용하기 위해 Git을 사용하였다.

이번에 구현한 프로젝트는 크게 main, Snake, SnakeGame의 세 가지 파일로 나눌 수 있으며, 각 부분의 구조와 개발 내용은 다음과 같다.

## <Snake>


이 부분은 말 그대로 Snake 자체를 담당하는 것으로, Snake 객체를 생성하여 해당 객체의 위치, 아이템 획득 정보, 진행 방향 등을 저장하고 변경한다. Snake의 멤버 함수인 moveTail을 통해 Snake의 head 방향으로 이동하고, 방향키를 통한 입력이 있을 경우 input 함수를 통해 진행 방향을 바꾸어준다. 또, Snake가 아이템과 만나면 checkitem을 통해 먹은 아이템의 종류를 확인하고 growth, poison 함수를 실행해, snake의 상태를 변경해 주는 기능 또한 포함하고 있다.

## <SnakeGame>

이 부분은 SnakeGame 클래스를 통해, stage에 따른 화면 구성, 아이템과 게이트의 생성, 배치와 같이 Snake를 제외한 게임의 배경적 요소와 실행을 담당하고 있다. 해당 클래스의 객체를 생성함으로써, map을 구성하고 미션 값을 고정으로 넣어 저장하며, 아이템과 게이트 그리고 시간을 초기화한다. 게임은 멤버 함수인 gamestart로부터 시작되며, 이는 미션 달성을 확인하는 checkMissionClear 함수의 결과값에 따라 stage를 넘겨가며 진행된다. 그리고 이때마다 renderMap 함수를 진행해 바뀐 map의 값에 해당하는 색을 넣어, 점수판과 함께 화면을 재구성한다. 이 이외에도 setGate와 setItem 함수를 통해 배경의 추가적 요소 역시 담당한다.

## <main>

게임 실행 전, 입출력에 대해 다양한 설정을 적용하고, 화면 구성을 위해 SnakeGame의 initColor 함수를

 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

불러 map의 값에 따른 색을 입혀 나타낸다. 이후, SnakeGame의 객체를 하나 만들어 해당 객체의 gameStart 함수를 실행함으로써 게임을 시작한다.

## 2 개발 내용 및 결과물

### 2.1 목표

#### 작성요령 (10점)

프로젝트의 목표를 기술하세요. 각 단계별 목표를 구체적으로 쓰세요.

적용단계	내용	적용 여부
1단계	Map의 구현	적용
2단계	Snake 표현 및 조작	적용
3단계	Item 요소의 구현	적용
4단계	Gate 요소의 구현	적용
5단계	점수 요소의 구현	적용

#### 1단계<Map의 구현>


NCurses Library 함수들을 사용해 2차원 배열로 된 Snake Map을 Game 화면으로 표시하는 프로그램을 완성한다.

- Wall과 Immune Wall을 구분해 구현한다.
- Stage는 최소 4개로 구성하며, 각 Stage 별로 Map을 다르게 구현한다.
- Map은 21X21을 최소 크기로 한다.

#### 2단계<Snake 표현 및 조작>

1단계의 맵 위에 Snake를 표시하고, 화살표를 입력받아 Snake가 규칙 #1을 준수하며 움직이도록 프로그램을 완성한다.

- 규칙#1
  - Snake는 진행 방향의 반대 방향으로 이동할 수 없다.
    - Head 방향 이동은 일정 시간(틱)에 의해 이동한다.
    - 방향키는 사용자가 직접 정한다.
    - 진행 방향(Head 방향)과 같은 방향키 입력은 무시한다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

- Tail 방향으로 이동할 수 없으며, 해당 방향과 같은 방향키 입력 시 실패한다.(Game over)
- 다른 실패 조건
  - Snake는 자신의 Body를 통과할 수 없다.
  - Snake는 벽을 통과할 수 없다.

### 3단계<Item 요소의 구현>


2단계 프로그램에서, Map 위에 Growth Item와 Poison Item을 출현시키고, Snake는 규칙#2를 준수하도록 한다.

- Growth Item과 Poison Item을 Map과 구분해 표현한다.
- 규칙#2
  - Growth Item과 Poison Item의 출현
    - Snake Body가 있지 않은 임의의 위치에 출현한다.
    - 출현 후 일정 시간이 지나면 다른 위치에 나타난다.
    - 동시에 출현할 수 있는 Item의 수는 3개로 제한한다.
  - Snake가 Growth Item을 획득한 경우
    - Snake의 몸의 길이가 진행 방향으로 1 증가한다.
  - Snake가 Poison Item을 획득한 경우
    - Snake의 꼬리 부분이 1 감소해 몸의 길이가 줄어든다.
    - 몸의 길이가 3보다 작아지면 실패한다.(Game over)

### 4단계<Gate 요소의 구현>

3단계 프로그램에서, Map의 Wall에 Gate를 규칙#3, #5에 따라 출현시키고, Snake가 통과해 규칙#4를 준수하며 이동할 수 있도록 한다.

- Wall, Immune wall과 Gate를 구분해 표현한다.
- 규칙#3
  - Gate 출현 설정
    - Gate는 두 개가 한 쌍이다.
    - Gate는 임의의 위치에 있는 벽에서 나타난다.
    - Gate는 한 번에 한 쌍만이 겹치지 않도록 나타난다.
  - Gate에 Snake가 진입 중인 경우
    - Gate는 사라지지 않는다.

 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

- 다른 위치에서 **Gate**가 나타나지 않는다.

#### - 규칙#4

- **Gate**가 나타나는 벽이 가장자리에 있을 때
  - 항상 **Map**의 안쪽 방향으로 진출한다.
  - 상단 벽 -> 아래 방향
  - 하단 벽 -> 위 방향
  - 좌측 벽 -> 오른쪽 방향
  - 우측 벽 -> 왼쪽 방향
- **Gate**가 나타나는 벽이 **Map**의 가운데에 있을 때 진출 우선순위
  - 진입 방향과 일치하는 방향
  - 진입 방향의 시계 방향으로 회전하는 방향
  - 진입 방향의 반시계 방향으로 회전하는 방향
  - 진입 방향과 반대 방향


#### - 규칙#5

- **Wall**
  - **Gate**로 변할 수 있다.
  - **Snake**가 통과할 수 없다.
  - **Snake Head**와 충돌 시 실패한다.
- **Immune wall**
  - **Gate**로 변할 수 없다.
  - **Snake**가 통과할 수 없다.
  - **Snake Head**와 충돌 시 실패한다.
- **Gate**의 출현 방법
  - 게임 시작 후 일정 시간이 지나면 출현한다.

### 5단계<점수 요소의 구현>

4단계 프로그램에서, 우측에 게임 점수와 **Mission**을 표시하는 화면을 구성하고, 게임 점수는 규칙#6을 준수한다.

#### -Mission

 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

- 구성된 Map의 정의에 고정 값을 주어 구현한다..
- Mission을 달성하면 다음 Map으로 넘어가 진행한다.

#### -규칙#6

- 게임 중 몸의 최대길이 계산
  - B: (현재 길이) / (최대 길이)
- 게임 중 획득한 Growth Item의 수 (+)
- 게임 중 획득한 Poison Item의 수 (-)
- 게임 중 Gate 사용 횟수 (G)

## 2.2 개발 내용 및 결과물

### 2.2.1 개발 내용

#### 작성요령 (10점)

프로젝트의 수행의 내용을 구체적으로 기술한다. 세부 목표별로 어떤 결과를 어떤 방법으로 달성하였는지를 자세히 기술한다.


#### 1단계 : Map의 구현

총 4단계의 stage와 map을 구현하기 위해서 21x21 크기의 map을 저장할 수 있는 'map'이라는 3차원 배열의 변수를 선언하였다. Map을 구성하는 요소로는 벽(WALL), 게이트가 생성될 수 없는 벽(IMMUNE WALL), 게이트(GATE), 성장 아이템(GROWTH), 독 아이템(POISON), Snake를 구성하는 머리(HEAD)와 몸(BODY), 그 외 나머지 빈칸들(BLANK) 등이 있다. 이 모든 요소들은 각각을 상수로 나타내 Map에서 이를 구분해 나타내었다.

#### 2단계 : Snake 표현 및 조작

Snake는 Node로 구성되어 있다. Node는 struct 구조체로 x, y 좌표값 이전 x, y좌표값과 다음 Node를 가리키는 Node\*인 next로 구성되어 있다. Snake는 Node\*인 Head를 가지고 있고, 이는 Snake의 몸을 가리키는 포인터이며, 이후 각각의 다음 몸을 가리키고 있다. 이외에도 Snake가 얻은 성장 아이템(growCount), 독 아이템의 카운트(poisonCount)와 지나온 게이트



 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

카운트(gateCount), 그리고 현재 Snake의 길이 정보(length) 또한 포함하고 있다.

Snake는 입력받은 키패드의 방향키에 따라 움직이며 기존 이동방향과 반대 방향으로 움직일 시에 게임이 종료된다.

### 3단계 : Item 요소의 구현

item은 struct 구조체로 x,y 좌표값과 성장 아이템인지 독 아이템인지 구분하는 boolean값의 isGrowth로 구성되어 있으며 게임에 표시되는 아이템은 2개이다.


item을 담는 itemList의 좌표값은 -1로 초기화해 시작하게 된다. 이후 itemTimer가 CREATE\_ITEM\_TIME(아이템 생성시간)과 같아지게 되면 아이템을 map에 배치한다. 이때 item은 무작위로 생성되어야 하기 때문에 rand()함수를 통해 이를 구현하였다. 각 변의 길이에서 immune wall을 제외한 길이로 module 연산을 해서 위치를 랜덤으로 지정해 주고 rand()가 짝수일 때 성장 아이템 홀수일 때 독 아이템으로 지정해 주었다. 이후 Map에 해당 위치에 대해 성장 아이템인지 독 아이템인지를 표시해 주고 다시 map을 update 해주는 방식으로 진행된다.

### 4단계 : Gate요소의 구현

2개의 gate가 쌍으로 나와야 하니 길이가 2인 gate배열을 통해 gate를 저장했다. gate는 x,y 좌표값을 담을 수 있는 멤버 변수를 가지고 있고, gate의 위치를 rand함수를 통해 랜덤으로 지정해 주며 이때, gate배열의 첫 번째와 두 번째의 위치값을 비교해 같은 곳에 gate가 생기지 않게끔 구현했다.

### 5단계 : 점수 요소 구현

화면에 출력하는 printf() 함수를 이용하여 Map 우측에 게임 점수를 표시하는 화면을 구성했다. 점수판 상단에는 현재 스테이지와 현재 상태에 대한 정보 그리고 하단에는 해당 각 스테이지 마다의 정해진 목표와 목표 달성 여부를 체크 표시로 나타내었다.

 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

## 2.2.2 시스템 구조 및 설계도

### 작성요령 (30점)

프로젝트의 각 세부 목표의 주요 기능(알고리즘 등)에 대해서 기술한다. 세부 목표별로 수정한 프로그램 소스 파일을 나열하고, 해당 파일에서 세부 목표를 달성하기 위해 작성한 클래스/함수에 대해 나열하고, 각 요소에 대해 간략한 설명을 작성한다. 또한 각 요소의 개발자를 명시한다.

### 1) Color.h


```
#include <ncurses.h>

// 후에 팔레트 편히 쓰기 위한 enum
enum Color {
    WHITE = 1,
    CYAN,
    BLUE,
    RED,
    YELLOW,
    MAGENTA,
    GREEN,
    BLACK
};
```

색 지정을 보다 편하게 위해 enum을 사용해 색깔 별로 상수를 지정해 주었다.

```
static void initColor() {
    // curses 모드 시작
    initscr();
    // Color 사용 선언
    start_color();

    // 1번 팔레트, 폰트색, 폰트배경색
    init_pair(Color::WHITE, COLOR_BLACK, COLOR_WHITE);
    init_pair(Color::CYAN, COLOR_BLACK, COLOR_CYAN);
    init_pair(Color::BLUE, COLOR_BLACK, COLOR_BLUE);
    init_pair(Color::RED, COLOR_BLACK, COLOR_RED);
}
```

 국민대학교 소프트웨어학부 <b>C++</b> 프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

```

init_pair(Color::YELLOW,COLOR_BLACK,COLOR_YELLOW);
init_pair(Color::MAGENTA,COLOR_BLACK,COLOR_MAGENTA);
init_pair(Color::GREEN,COLOR_BLACK,COLOR_GREEN);
init_pair(Color::BLACK,COLOR_WHITE,COLOR_BLACK);


// Attribute 적용
attron(COLOR_PAIR(Color::WHITE));

// 한 Attribute로 윈도우 전체 적용
wbkgd(stdscr, COLOR_PAIR(Color::WHITE));

// Attribute 해제
attroff(COLOR_PAIR(Color::WHITE));
}

```

curses 모드를 시작하고 init\_pair를 통해 각각의 팔레트를 폰트 배경색에 맞추어 지정해 주었으며, white로 윈도우 전체에 적용시켰다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16


## 2) Constant.h

```

1  #define MAX 21
2  #define BLANK 0
3  #define WALL 1
4  #define IMMUNE_WALL 2
5  #define HEAD 3
6  #define BODY 4
7  #define GROWTH 5
8  #define POISON (int)7
9  #define GATE 7
10 #define SPEED 600//
11 #define CREATE_ITEM_TIME 5000
12 #define CREATE_GATE_TIME 20000
13 #define LEFT_KEY 75
14 #define RIGHT_KEY 77
15 #define UP_KEY 72
16 #define DOWN_KEY 80

```

계속 쓰이게 될 상수들은 해당 헤더 파일에 따로 정리해둬으로써, 코드의 가독성을 높였다.

 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16


### 3) main.cpp

```

1  #include <cstdlib>
2  #include <ctime>
3  #include "SnakeGame.h"
4  // #include "Color.h"
5
6  using namespace std;
7
8  int main()
9  {
10     // Curses 모드 시작
11     initscr();
12     cbreak(); // 입력을 즉시 프로그램에서 사용
13     noecho(); // 입력을 출력하지 않음
14     scrollok(stdscr, TRUE); // 창의 커서가 창 영역을 벗어났을 때, 한 줄 위로 스크롤
15     nodelay(stdscr, TRUE); // getch 비 블로킹 (getch-> 입력이 없으면 -1 반환, 그렇지 않으면 키가 눌릴 때까지 기다림)
16
17
18     keypad(stdscr, TRUE); // 특수 키 입력 가능하게끔 설정
19     srand(time(NULL));
20     initColor();
21
22     SnakeGame game;
23     game.gameStart();
24 }
25

```

main 함수에서는 ncurses의 여러 함수를 사용하여 화면과 입출력에 대한 설정을 한다. initscr 함수로 화면을 초기화하고, curses모드를 시작한 후 cbreak를 통해 입력을 즉시 프로그램에서 사용하도록 설정한다. 그리고 noecho로 입력을 출력하지 않게끔 함께 설정해 주었다. scrollok를 통해 창의 커서가 창 영역을 벗어났을 때, 한 줄 위로 스크롤 하게끔 설정하고, nodelay모드를 통해 입력이 없으면 -1을 반환, 그렇지 않으면 키가 눌릴 때까지 기다리게 한다. keypad 함수를 사용해서 특수키가 입력 가능하게끔 설정해 주었고, 컬러를 초기화한 후 SnakeGame 객체 game 생성 후 게임을 시작한다.

 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

#### 4) SnakeGame.h

```

1  #include <ncurses.h>
2  #include <cstdlib>
3  #include "Color.h"
4  #include "Constant.h"
5  #include "Snake.h"

```


앞서 선언해둔 색깔과 상수 그리고 ncurses모드와 vector를 사용하기 위해 다음과 같이 include 해준다.

```

8  struct Item
9  {
10     int x;
11     int y;
12     bool isGrowth;
13 };
14
15 struct Mission
16 {
17     int maxLength;
18     bool isMaxLength;
19     int maxGrowth;
20     bool isMaxGrowth;
21     int maxPoison;
22     bool isMaxPoison;
23     int maxGate;
24     bool isMaxGate;
25 };

```

게임에 사용하게 될 item과 Mission struct를 구성해 준다. Item의 좌표를 담아둘 x,y 변수와 함께 bool타입의 isGrowth를 가지고 있으며, 이는 item struct가 Growth 아이템과 Poison 아이템

 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

모두에게 쓰일 수 있도록 둘을 구분해 준다. **Mission struct**는 **mission**의 목표들과 해당 목표의 달성 여부를 담고 있는 **bool**타입의 변수들을 포함하고 있다.

- **maxLength**는 Snake의 목표 최대길이, **isMaxLength**는 해당 목표 달성 여부
- **maxGrowth**는 Snake의 목표 성장 아이템 획득 수, **isMaxGrowth**는 해당 목표 달성 여부
- **maxPoison**은 Snake의 목표 독 아이템 획득 수, **isMaxPoison**은 해당 목표 달성 여부
- **maxGate**는 Snake의 목표 게이트 통과 수, **isMaxGate**는 해당 목표 달성 여부

```

27  class SnakeGame
28  {
29  private:
30      Mission mission[4];
31      Snake snake;
32      Gate gate[2];
33      Item itemList[2];
34
35      int stage;
36      int map[4][MAX][MAX];
37
38      int itemTimer;
39      int gateTimer;

```

다음은 **SnakeGame** 클래스가 가지고 있는 **private**한 멤버 변수들이다.

- 위에서 선언한 **Mission**타입의 **mission**을 각 스테이지 마다 설정하기 위해 길이가 4인 배열로 선언해 준다.
- **Snake** 클래스의 객체 **Snake**를 생성한다.
- **Gate**는 한 번에 두 개씩 나타나므로 길이가 2인 **Gate**배열로 생성한다.
- 전체적인 **map**을 담고 있을 배열을 **stage**의 수에 맞춰 3차원 배열로 생성한다.
- 각 스테이지의 장애물 벽을 담고 있을 **wallList**를 선언해 준다.
- **item**이 출현하고 난 이후의 시간을 저장하기 위해 **itemTimer**을 선언한다.
- **gate**가 출현하고 난 이후의 시간을 저장하기 위해 **gateTimer**을 선언한다.




```
41     public:
42         SnakeGame();
43         ~SnakeGame();
44
45         bool checkMissionClear();
46         void setGate(int Map[MAX][MAX]);
47         void resetItems(int Map[MAX][MAX]);
48         void setItem(int Map[MAX][MAX]);
49         void renderMap(int Map[MAX][MAX]);
50         void renderBlock(int color);
51         void gameStart();
52     };
53
```

SnakeGame의 public 함수이다.

- checkMissionClear() : 해당 스테이지 목표 달성 여부를 확인한다.
- setGate() : 게이트를 map에 배치한다.
- resetItems() : item과 gate가 발생한 후 미리 지정해둔 일정 시간이 지나면 다시 setItem 또는 setGate를 불러 새로운 item과 gate를 생성한다.
- setItem() : 아이템을 map에 배치한다.
- renderMap() : map의 값에 맞춰 적절한 색을 입히고 점수판을 구성함으로써, 화면을 구성한다.
- renderBlock() : 한 칸씩 색칠한다.
- gameStart() : 게임을 시작하기 위한 메인 함수이다.



 <div> 국민대학교  소프트웨어부  C++프로그래밍 </div>	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

## 5) SnakeGame.cpp

```

1  #include "SnakeGame.h"
2
3  SnakeGame::SnakeGame()
4  {
5      stage = 1;
6
7
8      for(int i = 0; i < 4; i++)
9      {
10         for (int y = 0; y < MAX; y++)
11         {
12             for (int x = 0; x < MAX; x++)
13             {
14                 // 테두리 라인
15                 if (x == 0 || x == MAX - 1 || y == 0 || y == MAX - 1) {
16                     // 왼쪽 상단, 왼쪽 하단, 오른쪽 상단, 오른쪽 하단 -> immune wall
17                     if ((x == 0 && y == 0) || (x == 0 && y == MAX - 1) || (x == MAX - 1 && y == 0) || (x == MAX - 1 && y == MAX - 1))
18                         map[i][y][x] = IMMUNE_WALL;
19                     // 꼭짓점 제외 모두 그냥 벽
20                     else
21                         map[i][y][x] = WALL;
22                 }
23                 else
24                     map[i][y][x] = BLANK;
25             }
26         }
27     }


```

SnakeGame의 생성자이다. 초기 Map의 테두리를 벽으로 하되, 꼭짓점 4군데는 수정할 수 없는 벽으로 해준 뒤 나머지는 모두 빈칸으로 처리해 준다. for문을 통해 각 stage의 map 배열을 돌며 가장자리에 해당하는 좌표에는 앞에서 정의한 상수 WALL을 대입한다. 하지만 네 꼭짓점의 경우 IMMUNE\_WALL을 대입해 수정 불가능한 벽으로 설정한다. 이 이외에 나머지 좌표는 모두 BLANK로 처리해 빈칸으로 설정한다.



```
29 // 임의로 장애물 벽 생성하고 리스트에 저장
30 for(int i = 0; i < 4; i++)
31 {
32     if (i==0)
33         continue;
34     else if (i==1) {
35         for(int y = 5; y <= 10; y++)
36         {
37             map[i][y][5] = map[i][y][10] = WALL;
38         }
39     } else if (i==2) {
40         for(int x = 5; x <= 10; x++)
41         {
42             map[i][15][x] = map[i][8][x] = WALL;
43         }
44     } else if (i==3) {
45         for(int x = 3; x <= 10; x++)
46         {
47             map[i][x][15] = map[i][x][12] = WALL;
48         }
49     }
50 }
51 }
```

stage1을 제외한 각 스테이지에 사용될 장애물 벽을 생성해준다.. 가장 바깥 for문의 i=0일때 즉, stage1의 경우에는 continue를 통해 넘기고, 이후 상황에서는 하위 for문에서 지정해둔 임의의 범위를 돌며 해당 좌표를 WALL로 설정해 벽으로 나타낸다.


 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

```

53 mission[0].maxLength = 3;
54 mission[0].isMaxLength = false;
55 mission[0].maxGrowth = 1;
56 mission[0].isMaxGrowth = false;
57 mission[0].maxPoison = 1;
58 mission[0].isMaxPoison = false;
59 mission[0].maxGate = 1;
60 mission[0].isMaxGate = false;
61
62 mission[1].maxLength = 5;
63 mission[1].isMaxLength = false;
64 mission[1].maxGrowth = 2;
65 mission[1].isMaxGrowth = false;
66 mission[1].maxPoison = 2;
67 mission[1].isMaxPoison = false;
68 mission[1].maxGate = 2;
69 mission[1].isMaxGate = false;
70
71 mission[2].maxLength = 6;
72 mission[2].isMaxLength = false;
73 mission[2].maxGrowth = 3;
74 mission[2].isMaxGrowth = false;
75 mission[2].maxPoison = 3;
76 mission[2].isMaxPoison = false;
77 mission[2].maxGate = 3;
78 mission[2].isMaxGate = false;
79
80 mission[3].maxLength = 7;
81 mission[3].isMaxLength = false;
82 mission[3].maxGrowth = 4;
83 mission[3].isMaxGrowth = false;
84 mission[3].maxPoison = 4;
85 mission[3].isMaxPoison = false;
86 mission[3].maxGate = 4;
87 mission[3].isMaxGate = false;

```

각 스테이지의 목표 미션들을 각각 다르게, 점차 어려워지도록 고안해 설정해 주었다.

 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

```

90      // 아이템 초기화
91      for (int i = 0; i < 2; i++)
92      {
93          itemList[i].x = -1;
94          itemList[i].y = -1;
95      }
96      //게이트 초기
97      for (int i = 0; i < 2; i++)
98      {
99          gate[i].x = -1;
100         gate[i].y = -1;
101     }
102     // 아이템 시간 초기화
103     itemTimer = 0;
104     gateTimer = 0;
105 }

```

item과 gate들을 -1로 초기화 시켜주고, 이들을 일정 시간마다 새롭게 생성할 수 있도록, 시간 조건을 판단하는 데 사용할 Timer 변수를 0으로 초기화해 생성한다.

```

157     //실질적으로 색을 넣는 부분
158     void SnakeGame::renderBlock(int color)
159     {
160         attron(COLOR_PAIR(color));
161         printf(" ");
162         attroff(COLOR_PAIR(color));
163     }

```

다음은 renderBlock 함수이다. 인자로 받은 color를 이용해서 한 칸에 색을 입힌다.



```
106 //맵의 숫자에 맞춰 지정된 색을 입력
107 void SnakeGame::renderMap(int map[MAX][MAX])
108 {
109     clear();
110     for (int y = 0; y < MAX; y++)
111     {
112         for (int x = 0; x < MAX; x++)
113         {
114             int color = Color::WHITE;
115
116             if (map[y][x] == WALL)
117                 color = Color::YELLOW;
118             else if (map[y][x] == HEAD)
119                 color = Color::CYAN;
120             else if (map[y][x] == BODY)
121                 color = Color::BLUE;
122             else if (map[y][x] == GROWTH)
123                 color = Color::GREEN;
124             else if (map[y][x] == POISON)
125                 color = Color::RED;
126             else if (map[y][x] == IMMUNE_WALL)
127                 color = Color::BLACK;
128             else if (map[y][x] == GATE)
129                 color = Color::MAGENTA;
130
131             renderBlock(color);
132         }
133     }
```

```
133 //점수판
134 if (y == 0)
135     printf("    Score Board  %d Stage", stage);
136 else if (y == 1)
137     printf("    B : %d/%d", snake.getLength(), mission[stage - 1].maxLength);
138 else if (y == 2)
139     printf("    + : %d", snake.getGrowthCount());
140 else if (y == 3)
141     printf("    - : %d", snake.getPoisonCount());
142 else if (y == 4)
143     printf("    G : %d", snake.getGate());
144 else if (y == 7)
145     printf("    Mission");
146 else if (y == 8)
147     printf("    B : %d (%c)", mission[stage - 1].maxLength, mission[stage - 1].isMaxLength ? 'v' : ' ');
148 else if (y == 9)
149     printf("    + : %d (%c)", mission[stage - 1].maxGrowth, mission[stage - 1].isMaxGrowth ? 'v' : ' ');
150 else if (y == 10)
151     printf("    - : %d (%c)", mission[stage - 1].maxPoison, mission[stage - 1].isMaxPoison ? 'v' : ' ');
152 else if (y == 11)
153     printf("    G : %d (%c)", mission[stage - 1].maxGate, mission[stage - 1].isMaxGate ? 'v' : ' ');
154     printf("\n");
155 }
156 }
```

renderMap 함수는 전체적인 화면을 구성해 주는 함수로, map을 모두 순환하며 각 좌표의 값에 해당하는 색으로 칠해준다. 이때, 위의 Color.h에 선언되어 있는 팔레트를 사용한다. 또, 화면 우측에 점수판 및 미션, 미션 달성 여부를 printf()함수로 출력해 나타낸다.



```
207 //실질적으로 아이템을 만드는 부분
208 void SnakeGame::setItem(int Map[MAX][MAX])
209 {
210     for (int i = 0; i < 2; i++)
211     {
212         if (itemList[i].x != -1 && itemList[i].y != -1)
213             Map[itemList[i].y][itemList[i].x] = BLANK;
214
215         do
216         {
217             itemList[i].x = rand() % (MAX - 2) + 1;
218             itemList[i].y = rand() % (MAX - 2) + 1;
219         }
220         while(snake.checkNode(itemList[i].x, itemList[i].y)
221             || Map[itemList[i].y][itemList[i].x] != BLANK);
222
223         itemList[i].isGrowth = rand() % 2 == 0 ? true : false;
224         if (itemList[i].isGrowth)
225             Map[itemList[i].y][itemList[i].x] = GROWTH;
226         else
227             Map[itemList[i].y][itemList[i].x] = POISON;
228     }
229 }
```


setItem함수는 현재 item이 놓여있던 자리를 BLANK로 만들어 주고, 새로운 item의 위치를 rand 함수를 통해 발생시킨다. 이때, while문의 snake.checkNode 함수로 해당 item과 snake가 겹치는지 확인해, 겹치지 않거나 해당 좌표가 공백이라면 itemList의 item에게 발생된 좌표값을 그대로 할당해 준다. 해당 item의 종류는 rand()가 짝수이면 성장 아이템, 홀수라면 독 아이템으로 지정하며, 해당 아이템에 해당하는 숫자를 대입해 이후 화면에서 적절한 색이 적용되어 보일 수 있도록 한다.



```
178 //실질적으로 게이트를 만드는 부분
179 void SnakeGame::setGate(int map[MAX][MAX])
180 {
181     for (int i = 0; i < 2; i++)
182     {
183         int lastY = gate[i].y;
184         int lastX = gate[i].x;
185
186         do
187         {
188             if (rand() % 2 == 0) {
189                 gate[i].x = MAX - 1;
190                 gate[i].y = rand() % (MAX - 2) + 1;
191             }
192             else {
193                 gate[i].x = 0;
194                 gate[i].y = rand() % (MAX - 2) + 1;
195             }
196         }
197         while(i == 1 && gate[0].y == gate[1].y && gate[0].x == gate[1].x);
198
199         // 게이트가 있다면
200         if (lastY != -1 && lastX != -1)
201             // 게이트 제거
202             map[lastY][lastX] = WALL;
203         // 게이트 재할당
204         map[gate[i].y][gate[i].x] = GATE;
205     }
206 }
```

게이트를 Map에 세팅하는 함수로, lastY, lastX를 통해 이전 gate의 좌표값을 저장하고 rand()함수를 통해 새로운 좌표값을 gate의 x,y좌표값에 넣어준다. 두 번째 gate가 첫 번째와 다를 때까지 이를 반복한다. 이전 값을 저장해둔 변수를 사용해 기존에 있던 게이트를 일반 벽으로 초기화시키고 새로운 게이트를 map에 세팅시켜준다.

```
164 //일정 시간이 지나면 아이템과 게이트 배치
165 void SnakeGame::resetItems(int Map[MAX][MAX])
166 {
167     if (++itemTimer == CREATE_ITEM_TIME)
168     {
169         setItem(Map);
170         itemTimer = 0;
171     }
172     if (++gateTimer == CREATE_GATE_TIME)
173     {
174         setGate(Map);
175         gateTimer = 0;
176     }
177 }
```

 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

다음은 resetItem 함수로, item과 gate의 Timer를 각각 1씩 더하면서 일정 시간에 도달하면 setItem 또는 setGate 함수를 실행시켜 item과 gate를 다른 위치에 새로 만들어낸다. 이후, timer들은 다시 0으로 초기화한다.

```

231 //미션 달성률 확인
232 bool SnakeGame::checkMissionClear()
233 {
234     if (snake.getLength() >= mission[stage - 1].maxLength)
235         mission[stage - 1].isMaxLength = true;
236     if (snake.getGrowthCount() >= mission[stage - 1].maxGrowth)
237         mission[stage - 1].isMaxGrowth = true;
238     if (snake.getPoisonCount() >= mission[stage - 1].maxPoison)
239         mission[stage - 1].isMaxPoison = true;
240     if (snake.getGate() >= mission[stage - 1].maxGate)
241         mission[stage - 1].isMaxGate = true;
242
243     if (mission[stage - 1].isMaxGate && mission[stage - 1].isMaxGrowth && mission[stage - 1].isMaxLength && mission[stage - 1].isMaxPoison)
244         return true;
245     return false;
246 }
247

```

snake의 길이를 가져와 해당 stage의 목표값보다 크거나 같다면 목표 달성 여부를 true로 바꿔주는 함수이다. 이와 같이 각각의 미션에 대한 상태를 모두 판단한 후, 해당 값들이 전부 true로 바뀌었다면 stage의 미션을 모두 clear 했다는 뜻으로, true를 리턴한다.

```

250 void SnakeGame::gameStart()
251 {
252     snake.setSnake(MAX / 2, MAX / 2);
253     snake.updateMap(map[0]);
254

```


SnakeGame의 gameStart() 함수로, 우선 snake를 map의 한가운데에 세팅시켜주고 map을 다시 update시켜준다.

```

256 while (true)
257 {
258     // 방향키 입력
259     snake.input();
260
261     int check;
262     int (&currentMap)[MAX][MAX] = map[stage - 1];
263
264     // 아이템 재배치
265     resetItems(currentMap);
266
267     check = snake.update(currentMap, gate);
268     if(check == -1)
269         break;
270

```



 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

while문 안에서 전체적인 게임이 진행된다. 유저에게 방향키 입력을 받아 **snake**의 진행 방향을 결정하고, **stage**에 알맞은 맵을 설정한다. **snake.update** 함수를 통해 **snake**가 일정 시간 간격으로 **head**의 방향 그리고 **gate**의 출입에 따라 움직이도록 한다. 이때, 뱀이 움직이다가 종료 조건에 도달하면 -1을 리턴해 while문을 break하게 된다.

```

271 // 미션 다 통과하면
272 if (checkMissionClear())
273 {
274     stage++;
275     if (stage > 4)
276         return;
277
278     for (int y = 0; y < MAX; y++)
279     {
280         for (int x = 0; x < MAX; x++)
281         {
282             // snake 지우기
283             if (currentMap[y][x] == HEAD || currentMap[y][x] == BODY)
284                 currentMap[y][x] = BLANK;
285         }
286     }
287     // snake 재배치
288     snake.setSnake(MAX / 2 - 2, MAX / 2);
289     snake.updateMap(currentMap);
290
291     // gate 초기화
292     currentMap[gate[0].y][gate[0].x] = WALL;
293     currentMap[gate[1].y][gate[1].x] = WALL;
294
295     // item시간 초기화
296     itemTimer = 0;
297     gateTimer = 0;
298 }

```


Snake가 정상적으로 움직였다면 이후에 **checkMissionClear** 함수를 통해 미션 통과 여부를 확인한다. 통과했다면 다음 **stage**로 넘어가고 그를 위한 초기화 작업을 진행해 준다. 기존 Map에 있던 Snake를 모두 빈칸과 같이 처리해 주고 Snake를 다시 맵의 가운데에 배치해 준다. 이후에는 게이트의 위치, **itemTimer** 그리고 **gateTimer**를 0으로 다시 초기화 시켜준다.

```

299 // 맵 다시 그리기
300 renderMap(currentMap);
301
302 // 1ms 뒤에 실행 (실행 지연시키기)
303 napms(1);
304 }
305 }

```

만약 Snake가 정상적으로 움직였지만 미션은 아직 통과시키지 못했다면 다시 Map을 그려주고 **napms**함수를 써서 1ms동안 실행을 지연시킨 후 다시 while문의 처음으로 돌아가게 한다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

## 6) Snake.h

```

1  #include "Constant.h"
2  #include <ncurses.h>
3
4  struct Gate {
5      // Gate의 x,y 좌표
6      int x;
7      int y;
8  };
9
10 enum Direction {
11     LEFT, // 0
12     RIGHT, // 1
13     UP, // 2
14     DOWN // 3
15 };
16
17 struct Node
18 {
19     int x;
20     int y;
21     int lastX;
22     int lastY;
23     Node* next;
24 };
25

```

미리 지정해둔 상수와, `ncurses` 라이브러리를 사용하기 위해 `Constant.h`, `ncurses.h`를 include 해준다. `Gate struct`는 발생한 `gate`의 `x, y` 좌표를 저장하고, `enum`으로 정의한 `Direction`은 `Snake`의 진행 방향을 가독성 있게 나타내기 위한 것이며, 마지막으로 `struct Node`는 `Snake`의 몸을 구성하는 것으로, 해당 `node`의 `x, y` 좌표와 이전 `node` 즉, 한 칸 앞 몸의 좌표 그리고 다음 `node`를 가리키는 포인터로 구성되어 있다.


```

26 class Snake
27 {
28 private:
29     Node* Head;
30     Direction direction, last_direction;
31     int oldClock;
32     int length;
33     int growCount;
34     int poisonCount;
35     int gateCount;
36     // bool moveTic;
37

```

다음으로 `Snake` 클래스의 `private` 변수이다.

- `Node* Head` : `Snake`의 머리를 가리키는 `Node` 포인터이다.

 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

- Direction direction, last\_direction : 현재 진행 방향과, 이전 진행 방향을 저장하는 변수이다.
- int oldClock : Snake가 생성된 후의 시간을 기록하는 변수이다. (한 칸을 이동할때마다 초기화된다)
- int length : Snake의 전체 길이를 저장하는 변수이다.
- int growCount : 지금까지 먹은 growth item의 수를 나타내는 변수이다.
- int poisonCount : 지금까지 먹은 poison item의 수를 나타내는 변수이다.
- int gateCount : 지금까지 gate를 사용한 횟수를 나타내는 변수이다.


```

38 public:
39     Snake();
40     ~Snake();
41
42     bool checkTail(int x, int y);
43     void isMaxGate(int Map[][MAX], Gate gate[2]);
44     bool poison();
45     void growth();
46     bool checkItem(int Map[MAX][MAX]);
47     bool checkNode(int x, int y);
48     void input();
49     void deleteMap(int Map[][MAX]);
50     void deleteSnake(Node* Node);
51     void setSnake(int x, int y);
52     int update(int Map[][MAX], Gate gate[2]);
53     void updateMap(int Map[][MAX]);
54     bool move(int Map[][MAX], Gate gate[2]);
55     void moveTail(Node* Node);
56     Direction nextDirection(Direction direction, int count);
57
58     int getLength() { return length; }
59     int getGrowCount() { return growCount; }
60     int getPoisonCount() { return poisonCount; }
61     int getGate() { return gateCount; }
62
63
64 };
65


```

Snake 클래스의 public 멤버 함수이다.

- checkTail : 인자값의 좌표가 Snake의 head를 제외한 부분과 겹치는 지를 판단한다.
- moveGate : Snake가 gate를 이동할 때 게이트를 나와 진행 방향과 반대로 이동하게끔 해준다.
- poison : Snake가 poison item을 먹었을 때 몸을 줄여준다

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

- growth : Snake가 growth item을 먹었을 때 몸을 늘려준다.
- checkItem : Snake가 먹은 item의 종류를 판단해 준다.
- checknode : 인자값의 좌표가 Snake와 겹치는 지를 판단한다.
- input : 키보드의 입력을 통해 Snake의 진행 방향을 설정한다.
- deleteMap : map에서 Snake를 지운다.
- deleteSnake : Snake를 삭제한다.
- setSnake : Snake를 새로 생성한다.
- update : Snake가 진행 방향에 따라 일정한 시간 간격으로 이동하도록 한다.
- updateMap : Snake의 head 와 body에 해당하는 map 좌표에 적절한 상수를 주어 표현한다.
- move : gate를 통한 Snake의 이동을 담당한다.
- moveTail : Snake의 head를 따라 몸을 이동시킨다.
- nextDirection : Snake가 gate를 나와 어느 방향으로 가야 할지를 결정한다.
- getLength : private 변수인 length를 return 해준다.
- getGrowthCount : private 변수인 growCount를 return 해준다.
- getPoisonCount : private 변수인 poisonCount를 return 해준다.
- getGate : private 변수인 gateCount를 return 해준다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

## 7) Snake.cpp

```

1  #include "Snake.h"
2  #include <stdlib.h>
3
4
5  Snake::Snake()
6  {
7      Head = NULL;
8      direction = LEFT;
9      growCount = 0;
10     poisonCount = 0;
11     gateCount = 0;
12     // moveTic = false;
13 }

```


다음은 Snake.h를 include 해 작성한 Snake.cpp의 Snake 생성자이다. Snake가 아직 생성되지 않았으므로, Head는 NULL로, direction은 기본 진행 방향인 left로 설정한다. 또, growCount와 poisonCount, 그리고 gateCount를 0으로 설정해 준다.

```

15 void Snake::deleteMap(int Map[][MAX])
16 {
17     Map[Head->y][Head->x] = BLANK;
18     Node* temp = Head->next;
19     while (temp != NULL)
20     {
21         Map[temp->y][temp->x] = BLANK;
22         temp = temp->next;
23     }
24 }

```

Snake의 head부터 다음 노드를 가리키는 next를 따라, map의 해당 좌표값을 모두 blank로 만들어 Snake를 지운다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

```

26  bool Snake::checkNode(int x, int y)
27  {
28
29      Node* temp = Head;
30      while (temp != NULL)
31      {
32          if (temp->x == x && temp->y == y)
33              return true;
34          temp = temp->next;
35      }
36      return false;
37  }

```

들어온 인자값을 map의 좌표로 하여, 해당 좌표가 head를 포함한 Snake와 겹치는 지를 판단한다. temp의 값을 다음 node를 가리키게 업데이트 하면서 Snake가 차지하는 전체와 비교하고 겹칠 경우 true를, 그렇지 않을 경우 false를 return 한다.



```
39 void Snake::isMaxGate(int Map[][MAX], Gate gate[2])
40 {
41     for (int i = 0; i < 2; i++)
42     {
43         if (gate[i].x == Head->x && gate[i].y == Head->y)
44         {
45             gateCount++;
46             Direction current_direction = direction;
47             const int dir[4][2] = {{0, -1}, {0, 1}, {-1, 0}, {1, 0}};
48             for(int j = 0; j < 4; j++)
49             {
50                 current_direction = nextDirection(current_direction, j);
51                 int next_y = gate[1 - i].y + dir[current_direction][0];
52                 int next_x = gate[1 - i].x + dir[current_direction][1];
53
54                 if(Map[next_y][next_x] != WALL) {
55                     Head->y = next_y;
56                     Head->x = next_x;
57                     break;
58                 }
59             }
60             direction = current_direction;
61             break;
62         }
63     }
64 }
```

Snake의 머리 좌표가 gate에 닿을 때, 즉 gate에 진입할 때 gate를 지나온 수를 +1 시켜주고 다음 방향을 계산해 준다. 이후, gate의 좌표값에 적절한 dir 방향값을 더해 Snake의 다음 위치를 정해주고 그 위치가 벽이 아니라면 머리를 이동시키고 Snake의 방향을 재설정해 준다.

```
66 //snake의 (머리빼고) 몸이 인자의 위치(x,y)를 포함하는지
67 bool Snake::checkTail(int x, int y)
68 {
69     Node* temp = Head->next;
70     while (temp != NULL)
71     {
72         if (temp->x == x && temp->y == y)
73             return true;
74         temp = temp->next;
75     }
76     return false;
77 }
```

checkNode와 동일한 방법으로 head를 제외한 Snake의 node를 돌며, 인자로 들어온 map의 좌표와 비교해, 겹치는 부분이 있다면 true를 그렇지 않다면 false를 return 한다.




```
78 //poison을 먹었을 때
79 bool Snake::poison()
80 {
81     Node* temp = Head;
82     while (temp->next != NULL)
83     {
84         if (temp->next->next == NULL)
85         {
86             delete temp->next;
87             temp->next = NULL;
88             length--;
89             if (length <= 2)
90                 return false;
91             return true;
92         }
93         temp = temp->next;
94     }
95     return true;
96 }
```

poison 함수는 Snake가 poison item을 먹었을 때 실행되는 함수로, next를 따라 Snake의 꼬리를 찾아 삭제하고, Snake의 전체 길이를 나타내는 변수 length를 1감소시킨다. 이때, Snake의 길이가 3보다 짧아졌을 경우, 게임 종료의 조건에 해당되므로 false를, 그렇지 않다면 true를 return 해준다.

```
97 //growth를 먹었을 때
98 void Snake::growth()
99 {
100     Node* temp = Head;
101     while (temp->next != NULL)
102     {
103         temp = temp->next;
104         temp->next = new Node;
105         temp->next->x = temp->lastX;
106         temp->next->y = temp->lastY;
107         temp->next->next = NULL;
108         length++;
109     }
```

growth 함수 역시 poison 함수와 같은 방식으로 Snake의 꼬리를 찾고, 해당 node에 새로운 node를 할당받아 이어줌으로써, Snake의 몸을 늘리고 몸의 길이를 나타내는 변수 length를 1증가시킨다.



 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

```

109 //먹은 아이템의 종류 확인
110 bool Snake::checkItem(int Map[MAX][MAX])
111 {
112     switch (Map[Head->y][Head->x])
113     {
114         case GROWTH:
115             growth();
116             growCount++;
117             break;
118         case POISON:
119             poisonCount++;
120             return poison();
121     }
122     return true;
123 }


```

Snake의 head가 접촉한 item의 종류를 판단하고 그에 해당하는 함수를 실행시키는 것으로, 만약 growth item이라면 growth 함수를 실행한 후 Snake가 얻은 growth item의 수를 저장하는 growCount 변수의 값을 1 증가시켜주고, poison item이라면 같은 방식으로 poisonCount 변수의 값을 1 증가시킨 후, poison 함수를 실행한다.



```
125 bool Snake::move(int Map[][MAX], Gate gate[2])
126 {
127     deleteMap(Map);
128     Head->lastX = Head->x;
129     Head->lastY = Head->y;
130     switch (direction)
131     {
132     case LEFT:
133         Head->x--;
134         break;
135     case RIGHT:
136         Head->x++;
137         break;
138     case UP:
139         Head->y--;
140         break;
141     case DOWN:
142         Head->y++;
143         break;
144     }
145     last_direction = direction;
146
147     if (checkTail(Head->x, Head->y))
148     {
149         deleteSnake(Head);
150         Head=NULL;
151         return false;
152     }
153     isMaxGate(Map, gate);
154     if (Map[Head->y][Head->x] == WALL)
155     {
156         deleteSnake(Head);
157         Head=NULL;
158         return false;
159     }
160     moveTail(Head);
161     if(!checkItem(Map))
162         return false;
163     updateMap(Map);
164     return true;
165 }
```

snake가 이동하는 동안 map에서의 snake를 계속해서 그려주는 함수이다. head의 이전 x, y좌표에 현재 x, y좌표값을 넣어준 후, 현재의 x,y좌표는 입력받은 방향에 맞춰 변경한 좌표로

 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

설정해 snake의 머리를 옮긴다. 만약, 이동 시에 꼬리와 부딪힌다면 snake를 없애고 false를 return해, 이동 실패임을 알려 게임 종료로 이어지도록 한다. gate와 벽과의 충돌에 대해서도 똑같이 진행한다. 이후 snake를 이동하며 만약 poison item을 먹게 된다면 checkItem 함수를 통해 poison을 먹고 난 후의 길이를 확인하게 되고 3보다 작게 된다면 false를 return해 결국 게임 종료로 이어진다. 종료 조건들을 모두 회피했다면 map을 다시 그려주고 true를 return하며, 이는 snake가 성공적으로 이동했음을 뜻한다.

```

166 //head 방향으로 이동
167 void Snake::moveTail(Node* Node)
168 {
169     if (Node->next == NULL)
170         return;
171     Node->next->lastX = Node->next->x;
172     Node->next->lastY = Node->next->y;
173     Node->next->x = Node->lastX;
174     Node->next->y = Node->lastY;
175     moveTail(Node->next);
176 }

```

멤버 함수 moveTail은 인자로 들어온 노드의 next 값의 값을 node의 지난 위치로 변경하고, 이를 재귀적으로 반복하며 Snake의 이동을 담당한다.

```

178 void Snake::deleteSnake(Node* Node)
179 {
180     if (Node == NULL)
181         return;
182     deleteSnake(Node->next);
183     delete Node;
184 }

```

‘node의 next 값을 따라 재귀적으로 반복하며 node의 할당 내용을 해제한다. 인자인 node에 snake의 head 값을 줌으로써, Snake를 삭제하는 기능을 하게 된다.



```
186 void Snake::setSnake(int x, int y)
187 {
188     direction = LEFT;
189     if (Head != NULL)
190     {
191         deleteSnake(Head);
192         Head = NULL;
193     }
194     Head = new Node;
195     Head->x = x;
196     Head->y = y;
197     Head->next = NULL;
198     for (int i = 1; i <= 2; i++)
199     {
200         Node* add = new Node;
201         add->x = Head->x + i;
202         add->y = Head->y;
203         add->next = NULL;
204         Node* temp = Head;
205         while (temp->next != NULL)
206             temp = temp->next;
207         temp->next = add;
208     }
209     oldClock = 0;
210     length = 3;
211     growCount = 0;
212     poisonCount = 0;
213     gateCount = 0;
214 }
```


Snake의 초기 방향은 왼쪽으로 설정해두었다. snake를 새로 세팅하기 전에 혹시 Snake의 머리가 남아있다면 완전히 지운 후, 새로운 head 노드를 만들고 2개의 body 노드를 만들어 포인터로 연결해 줌으로써, 새 Snake를 생성한다. 또, 이동을 위한 oldClock을 0으로 만들고 초기 length는 3, grow item, poison item, gate 획득 수를 나타내는 변수들은 모두 0으로 초기화 시켜준다.



```
216 int Snake::update(int Map[][MAX], Gate gate[2])
217 {
218     if(++oldClock >= 500) {
219         if(!move(Map, gate))
220             return -1;
221         oldClock = 0;
222         return 1;
223     }
224     return 0;
225 }
```

update 함수는 Snake가 생성된 후의 시간을 측정하는 oldClock 변수가 일정 시간에 도달하면 진행 방향으로 한 칸 이동시키고 이동 한 후에 다시 0으로 초기화시킨다. 이때, move 함수를 통해 Snake 이동 시 게임 종료 조건이 발생하였는지를 판단해, 이동 중 종료 조건에 걸리게 된다면 -1을 return하고 정상적으로 이동이 됐다면 oldClock을 0으로 초기화시킨 후 1을 return한다.

```
226 //키보드 입력을 통해 움직일 방향 설정
227 void Snake::input()
228 {
229     int ch = getch(); //키보드에서 입력 받아옴
230
231     switch (ch)
232     {
233     case KEY_LEFT:
234         if (last_direction == RIGHT)
235             exit(0);
236         if (last_direction != RIGHT)
237             direction = LEFT;
238         break;
239     case KEY_RIGHT:
240         if (last_direction == LEFT)
241             exit(0);
242         if (last_direction != LEFT)
243             direction = RIGHT;
244         break;
245     case KEY_UP:
246         if (last_direction == DOWN)
247             exit(0);
248         if (last_direction != DOWN)
249             direction = UP;
250         break;
251     case KEY_DOWN:
252         if (last_direction == UP)
253             exit(0);
254         if (last_direction != UP)
255             direction = DOWN;
256         break;
257     }
258 }
```

 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

input 함수는 getch 함수를 통해 키보드에서 받아온 입력으로 Snake의 진행 방향을 결정하는 기능을 한다. 받아온 입력을 변수 ch에 저장하고 이를 switch문에 넣어 네 방향을 나타내는 상수와 비교해, 만약 현재 진행 방향과 동일하다면 게임 종료 조건에 해당하므로 exit(0)을 실행하고, 그렇지 않다면 현재 진행 방향을 입력받은 방향으로 변경해 준다.

```

259 //map에서 색표현을 위해 지정된 숫자 대입
260 void Snake::updateMap(int Map[][MAX])
261 {
262     Map[Head->y][Head->x] = HEAD;
263     Node* temp = Head->next;
264     while (temp != NULL)
265     {
266         Map[temp->y][temp->x] = BODY;
267         temp = temp->next;
268     }
269 }

```

map을 인자로 받아 Snake 부분에 색을 입히는 함수로, head 부분에 해당하는 map의 좌표에 상수로 저장해둔 head 값을 대입하고, Snake의 나머지 부분은 node 포인터 next를 따라가면서 상수 body를 대입해 해당하는 값을 지정해 준다.

```

271 Snake::~~Snake()
272 {
273     if (Head != NULL)
274     {
275         deleteSnake(Head);
276         Head = NULL;
277     }
278 }


```

Snake 객체의 소멸자로 head가 NULL이 아닐 경우 즉, Snake가 지워지지 않았을 경우 map에서 모두 제거한 후 객체를 소멸시킨다.



```
280 Direction Snake::nextDirection(Direction direction, int count)
281 {
282     const static Direction next[4] = {RIGHT, UP, LEFT, DOWN};
283     const static int dir_to_idx[4] = {2, 0, 1, 3};
284
285     if(count == 0)
286         return direction;
287     if(count == 1)
288         return next[(dir_to_idx[direction] + 3) % 4];
289     if(count == 2)
290         return next[(dir_to_idx[direction] + 1) % 4];
291     return next[(dir_to_idx[direction] + 2) % 4];
292 }
293
```

nextDirection 함수는 if문을 통해 gate가 들어왔던 방향과 반대 방향으로 return해주는 기능을 한다.

 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

### 2.2.3 활용/개발된 기술

#### 작성요령 (10점)

프로젝트 수행에 사용한 외부 기술/라이브러리를 나열하여 작성한다. 각각 기술을 이 프로젝트에 적용할 때, 도움 받거나 해결하고자 하는 기능에 대해 상세히 설명한다.

NCURSES / STL 라이브러리 등을 포함하여 설명한다.

또한, 이 프로젝트를 수행하면서, 새롭게 고안한 알고리즘 등이 있다면 설명한다.

#### 1. ncurses

ncurses란, 텍스트 모드에서 Window, Panel, Menu, Mouse, Color 등을 쉽게 사용할 수 있도록 도와주는 linux라이브러리로 curses의 새로운 버전이다. ncurses를 사용하기 위해서는 ncurses.h 헤더파일을 include해주어야 한다. 전체적인 게임의 화면을 출력하기 위해 사용했다.

#### 2. cstdlib

C표준 유틸리티 함수들을 모아놓은 헤더파일이다. 헤더파일 안에는 프로그래밍 시에 범용적으로 사용되는 여러 가지 함수들을 모아 놓고 있는데, 동적 할당 관련 함수, 난수 생성 함수, 정수의 연산 함수, 검색 및 정렬 함수 등이 있다. 게임에서 중요한 요소 중 하나인 랜덤 요소를 위한 rand() 함수의 사용이 필요해, 해당 헤더파일을 받아 사용하였다.

#### 3. ctime


C에서 날짜와 시간에 대한 정보를 가져오거나 다루는 함수들의 정의를 포함하고 있는 헤더파일이다. rand() 또한 시드를 바꿔주지 않으면 고정된 값으로 중복돼서 나와 완전히 랜덤이라고 할 수 없기 때문에 ctime을 이용해, 시간을 인자로 주어 항상 다른 랜덤값이 나오게끔 할 수 있었다.

### 2.2.4 현실적 제한 요소 및 그 해결 방안

#### 작성요령 (5점)

제안된 프로젝트의 단계 별 수행에 있어, 제한 요소를 찾아 작성한다. 해당 제한 요소를 해결하기 위해서 어떤 방법으로 해결하였는지 작성한다.



 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

Snake를 배열로 만들어 머리와 몸통을 삽입 삭제하면서 나타내고자 하였으나 실시간으로 이를 표현하는데 어려움이 있었다. 배열보다 **Node**를 구현한 후 포인터를 통해 다음 노드를 가리키는 형식으로 나타내는 것이 보다 자연스럽고 속도적인 측면으로도 이점이 있을 것 같아 포인터를 활용해 Snake의 몸과 머리를 구성하였으며, 의도한 바와 같이 구현할 수 있었다.


### 2.2.5 결과물 목록

#### 작성요령 (5점)

결과물 목록을 작성한다. 목록은 제출하는 파일과 각 파일의 역할을 간략히 설명한다.

- **Color.h**  
색깔을 쓰기 편하게 따로 팔레트를 만들어 정의해놓은 헤더파일이다.
- **Constant.h**  
이후 코드에서 상수들을 가독성 높고, 편하게 사용할 수 있도록 따로 정의해놓은 헤더파일이다.
- **main.cpp**  
게임을 시작하기 위한 초기설정과 게임을 시작하는 **main**함수를 포함하는 **cpp**파일이다.
- **Makefile**  
컴파일 명령을 순차적으로 실행하기 위한 파일이다.
- **Snake.h**  
게임 진행 중 발생하는 게이트와 스네이크의 구성 및 관련 함수들이 담겨있는 헤더파일이다.
- **Snake.cpp**  
Snake.h를 **include**해서 구현한 **cpp**파일이다.
- **SnakeGame.h**  
게임을 시작하고 진행을 담당하는 헤더파일이다.
- **SnakeGame.cpp**  
SnakeGame.h를 **include**해 게임의 시작과 전반적인 진행을 구현한 **cpp**파일이다.

## 3 자기평가

 국민대학교 소프트웨어부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

프로젝트를 수행한 자기 평가를 서술한다. 팀원 개인의 자기 평가가 포함되어야 하며, 본인의 역할, 프로젝트 수행 시 어려운 점, 도움이 되었던 점, 이 프로젝트 운영에 개선이 필요하다고 생각하는 점을 충분히 서술한다.

## 1. 김민재


이번 c++프로젝트에서 전체적인 게임의 진행 부분을 구현하는 역할을 맡아 구현했습니다. 수업 시간 때 배웠던 c++을 이론적으로 공부한 것에 그치는 것이 아니라 실제로 활용해서 프로젝트를 해볼 수 있어서 매우 유익했습니다. 하지만 직접 사용해서 게임 로직을 구현하고자 하니 생각보다 쉽지 않았고 익숙해지는 데에 많은 시간이 들었습니다. 이번 프로젝트를 통해서 포인터와 클래스 등 수업에서 배운 개념들을 직접 활용하면서 체득할 수 있어 좋았고 새롭게 ncurses 라이브러리를 사용해보며 명령어를 쳤을때 터미널에서 나오는 여러 화면효과들이 이런 라이브러리를 기반으로 만들어졌구나를 느끼며 나도 한번 만들어볼 수 있지 않을까라는 생각도 할 수 있게끔 된 것 같아 매우 뿌듯하고 많은 것을 느끼고 배울 수 있어 좋았습니다. 하지만 아직 c++실력과 알고리즘을 짜는 실력이 부족해서 게임 로직등을 구현하는데 어려움이 많았고 보기애나 기능적으로나 깔끔하게 짜지 못한 것 같아 제 실력의 미숙함을 느꼈습니다. 수업 시간 때 배운 레퍼런스나 클래스 관련 접근제어자등을 조금 더 알맞은 방향으로 사용하면 코드를 보다 더 개선시킬 수 있지 않을까 하는 아쉬움이 남습니다.

협업적인 측면에서는 게임 하나를 분업해서 협업하는 것에 있어서 어떤 식으로 분업을 진행해야 할지에 대해 많은 고민이 있었고 이번 프로젝트를 진행하면서 혼자 하는 개발이 아닌 팀원들과 함께하는 협업에 대해서 고민해 보고 느끼고 그 중요성에 대해서 알아갈 수 있는 좋은 기회가 된 것 같아서 아쉬움도 남지만 유익했던 것 같습니다.

## 2. 윤성욱

저는 이번 프로젝트 중 Snake 부분을 담당해 구현하였습니다. 클래스, 포인터 등 수업 시간에 배운 이론들을 실제로 프로젝트 구현에 사용해 봄으로써, 개념을 보다 확실하게 이해할 수 있었고, 코드를 여러 파일 또는 클래스로 세분화하는 것이 왜 중요한지 또한 깨달을 수 있는 시간이었습니다. 또, 외부 라이브러라인 ncurses 를 새롭게 사용해 보면서 저희가 이뤄낸 결과가 그래픽으로 보이는 점이 신기하고 더욱 뿌듯했으며, 다양한 외부 라이브러리의 존재와 기능들에 대해 살펴볼 수 있었습니다.

개인 프로젝트가 아닌 협업을 요하는 팀플레이였기 때문에 팀원들과 함께 개발하는 과정 자체에도 신경을 써서 진행하였습니다. 각자의 역할, 코드에서 통일 시킬 부분 그리고 코드를 공유하는 방법 등을 고민하고 결정하는 과정을 통해, 협업 프로젝트의 과정과 방법을 익히고 생각해 볼 수 있었습니다. 이번 프로젝트를 익숙하지 않은 언어와 라이브러리를 사용했을 뿐 아니라 프로젝트 형식으로 진행하면서 스스로에게 부족한 점도 많이 느꼈지만, 이번 활동을 통해 얻은 지식과 경험을 바탕으로 조금은 성장할 수 있었다고 생각합니다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

## 4 참고 문헌

번호	종류	제목	출처	발행년도	저자	기타
	서적					
	기사					

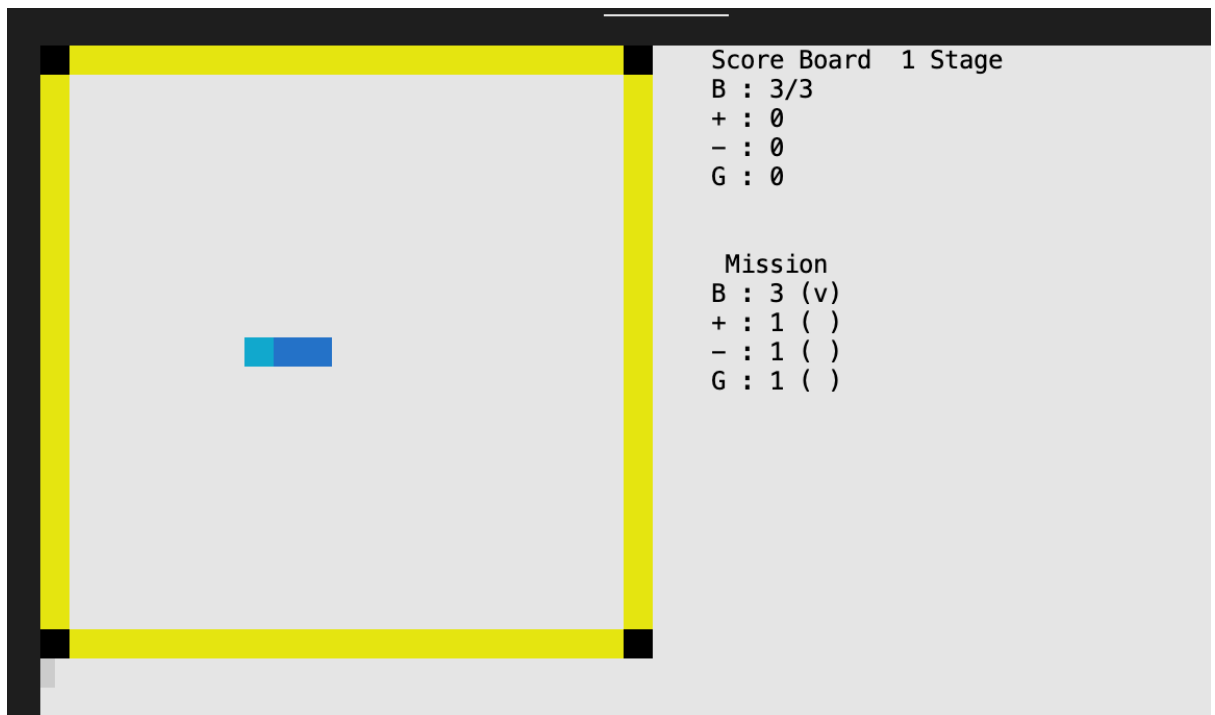
<https://widian.tistory.com/58> 의 ncurese 함수 참고

## 5 부록


작성요령 (15점)

프로젝트의 결과물을 사용하기 위한 방법에 대해서 작성하세요.

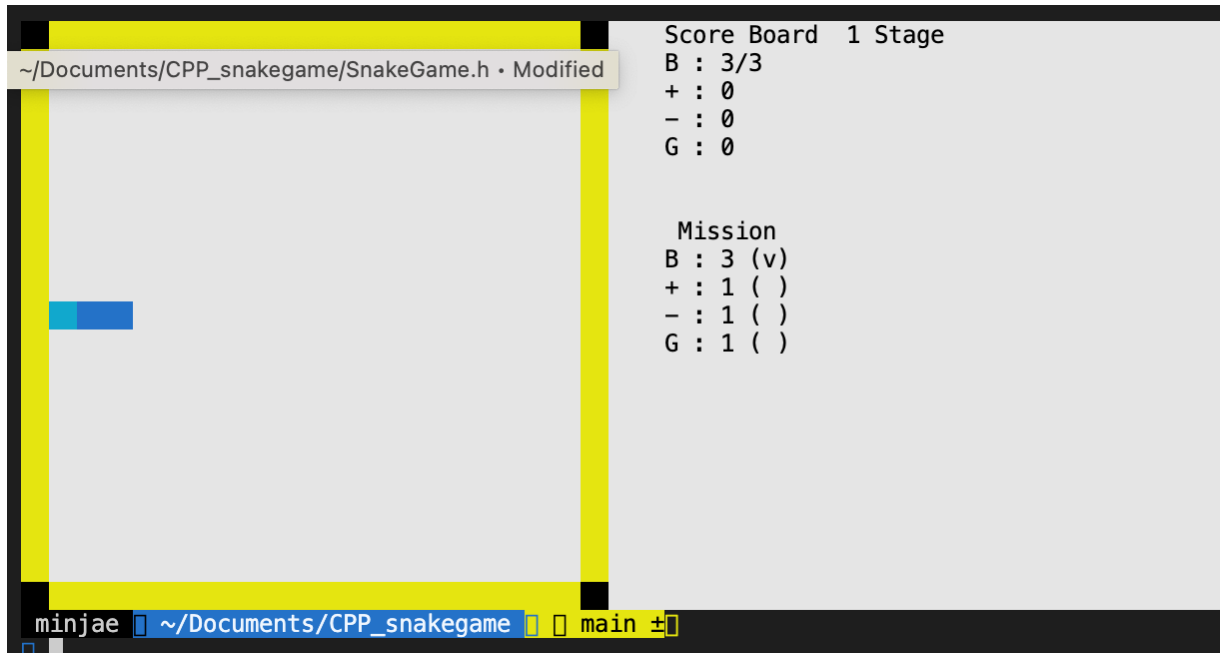
### 5.1 사용자 매뉴얼



snake 실행파일 실행모습


 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

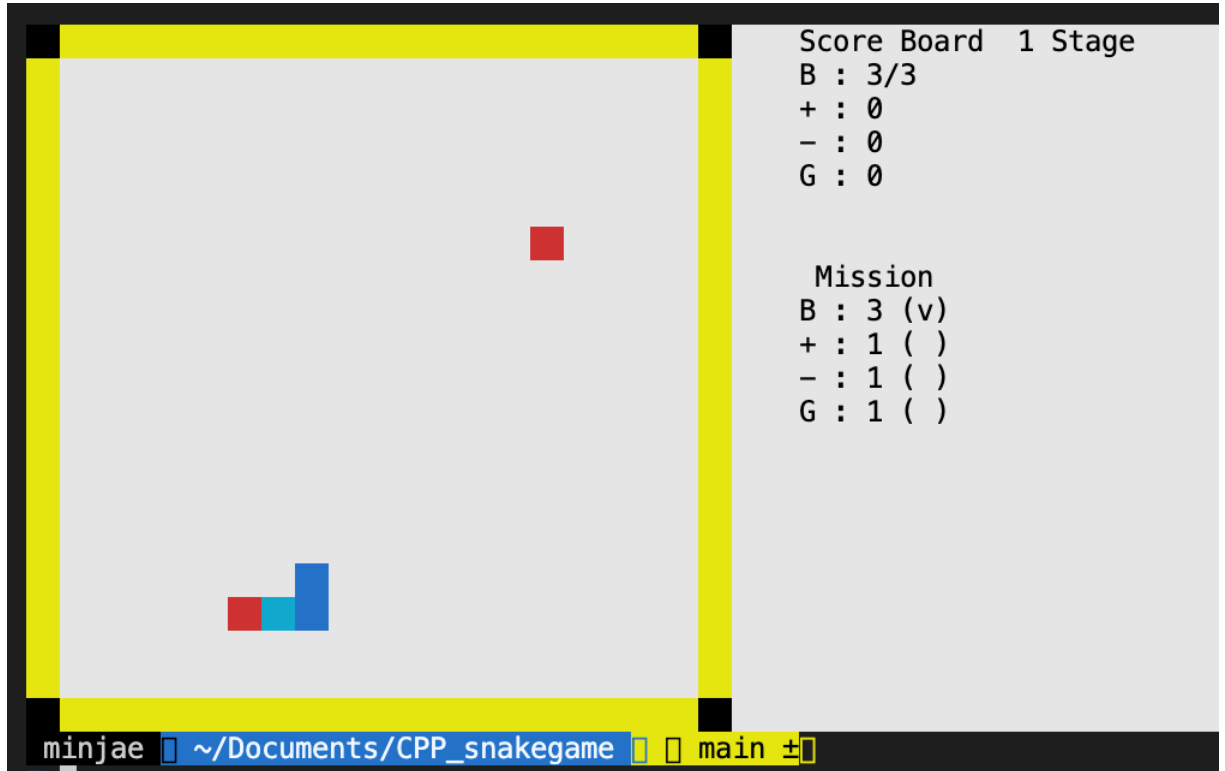
make명령어로 생성된 snake실행 파일을 실행시키면 곧바로 게임이 시작되고 snake는 기본 진행 방향인 좌측 방향으로 나아간다.



snake가 벽에 충돌한 모습


snake가 벽에 충돌하게 되면 규칙#2에 따라 프로세스가 종료된다.

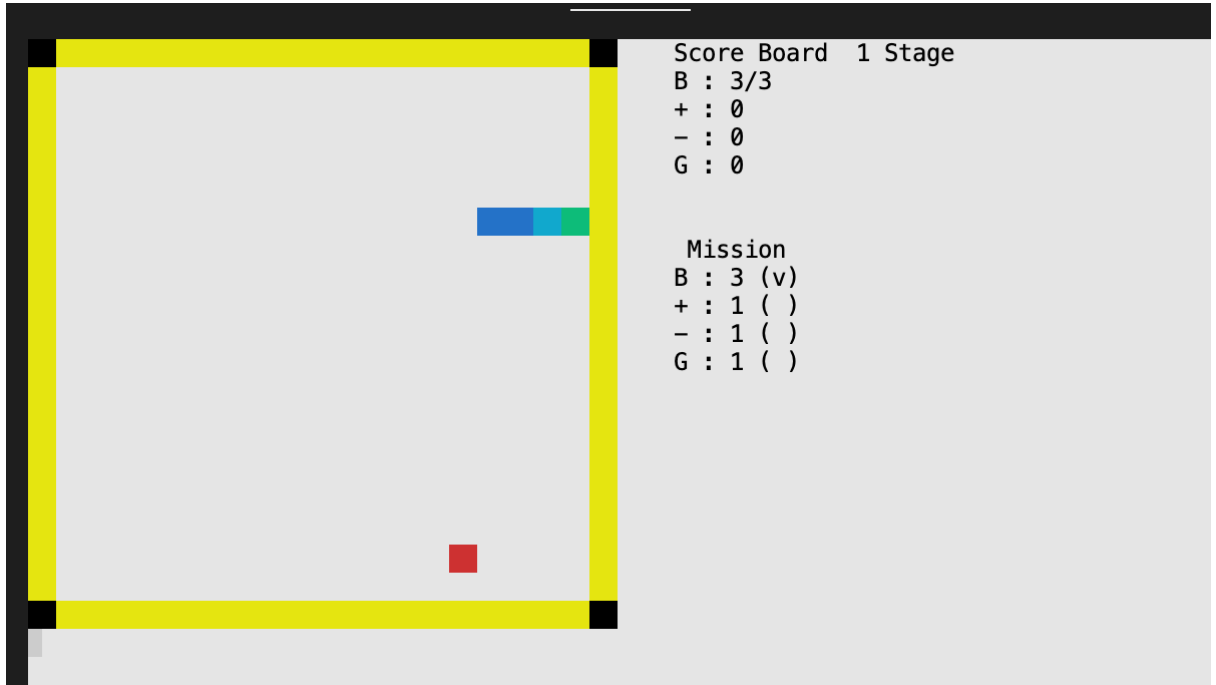
 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16



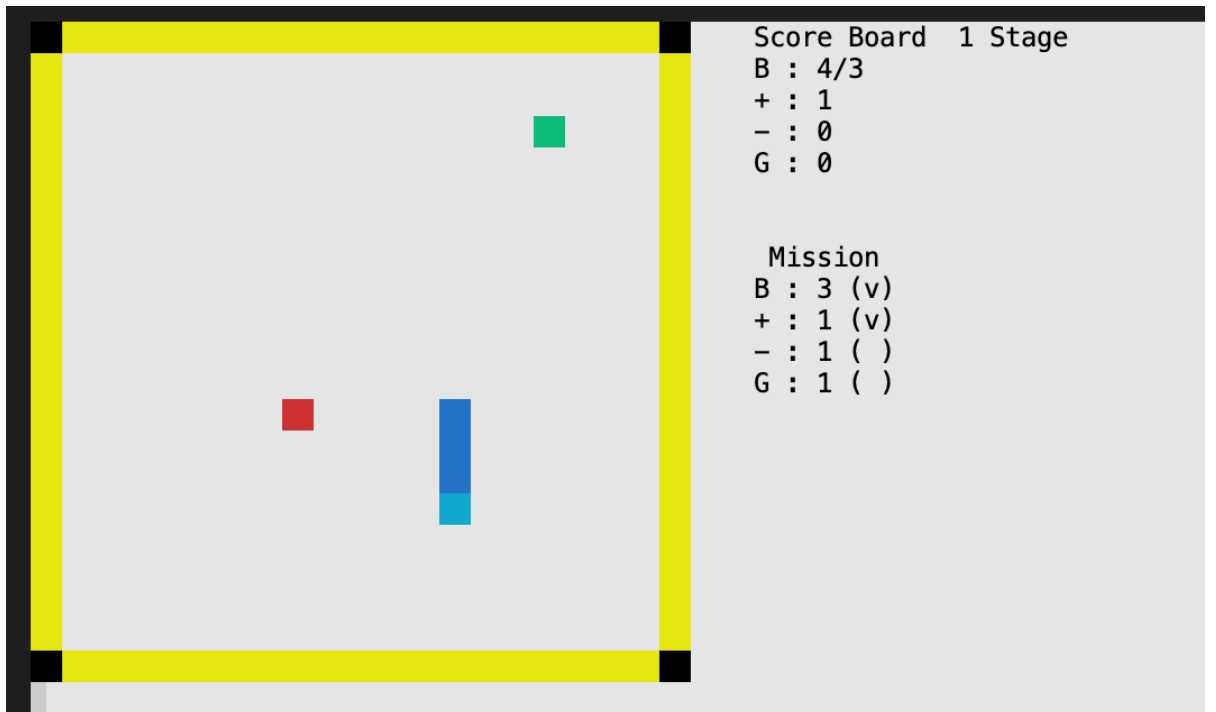
**snake**가 길이가 3일 때 **Poison**을 먹을 시 게임 종료

초기 설정인 길이 3인 상태에서 **Poison**을 먹게 되면 길이가 2가 되므로 규칙#2에 따라 게임이 종료된다.


 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

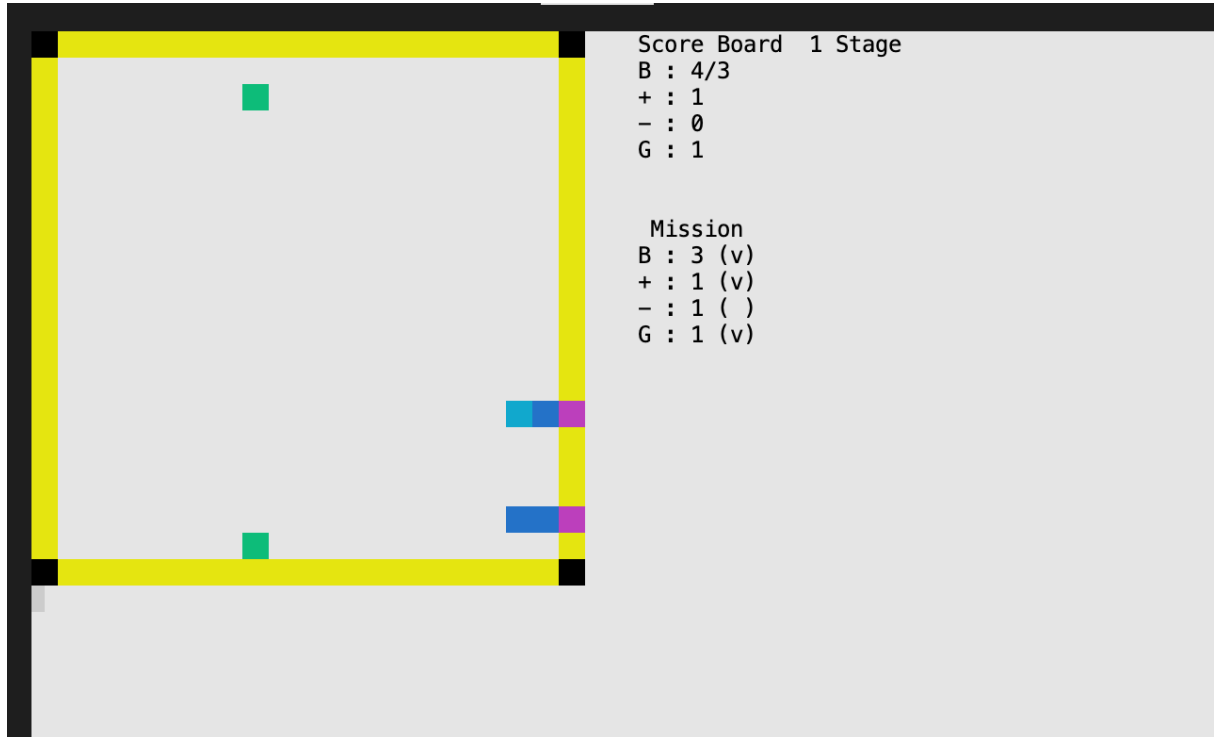


**Growth** 아이템을 먹기 직전 모습



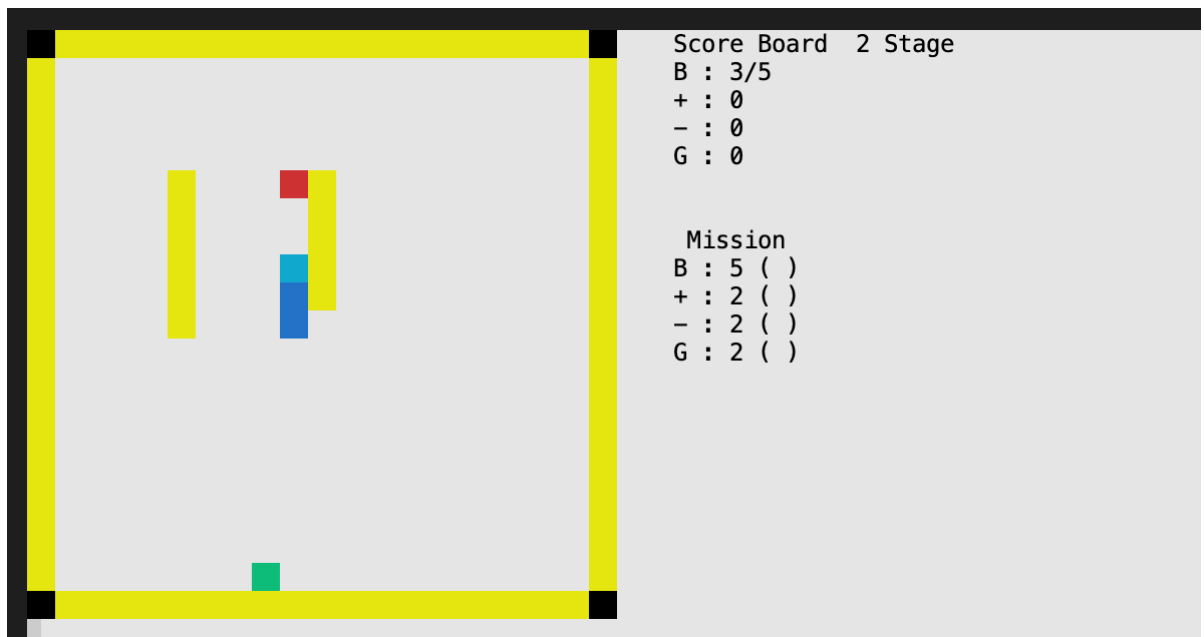
**Growth** 아이템을 먹고 길이가 1증가한 **snake**모습


 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16



### Snake가 Gate를 통과하는 모습

우측 진행 방향으로 들어가자 좌측 방향으로 나오는 걸 확인할 수 있다. 또한 우측의 점수판을 통해 현재 스코어와 미션 달성 여부가 표시된다.



 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	snakegame	
	팀 명	10팀	
	Confidential Restricted	Version 1.5	2024-06-16

미션을 달성하고 다음 스테이지로 넘어간 모습

미션을 달성하면 다음 스테이지로 넘어가게 되고 **map**에 장애물이 생긴 모습을 확인할 수 있으며,  
우측 점수판을 통해 현재 **2stage**가 진행 중임을 알 수 있다.

## 5.2 설치 방법

1. 터미널을 켜고 해당 폴더들이 있는 폴더로 경로 이동 (**cd** 명령어)
2. 해당 경로에서 **makefile**을 실행시키기 위한 명령어 실행

**make**

3. 생성된 실행파일(**snake**)을 실행시킨다.

**./snake**