**C++프로그래밍**

**프로젝트**

|  |  |
| --- | --- |
| **프로젝트 명** | *스네이크 게임* |
| **팀 명** | *유레채* |
| **문서 제목** | 스네이크 결과보고서 |

|  |  |
| --- | --- |
| **Version** | 1.2 |
| **Date** | 2024-06-15 |

|  |  |
| --- | --- |
| **팀원** | 유채희 (팀장) |
| 박유나 |
| 레일라 |
|  |
|  |

|  |
| --- |
| **CONFIDENTIALITY/SECURITY WARNING**  이 문서에 포함되어 있는 정보는 국민대학교 소프트웨어융합대학 소프트웨어학부 및 소프트웨어학부 개설 교과목 C++프로그래밍 수강 학생 중 프로젝트 “스네이크 게임”를 수행하는 팀 “유레채”의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 “유래체”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다. |

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | 최종보고서-스네이크게임.doc |
| **원안작성자** | 유채희, 박유나, 레일라 |
| **수정작업자** | 유채희, 박유나, 레일라 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2024-05-11 | 유채희 | 1.0 | 최초 작성 | 개요 & 1,2,3,4 단계 개발 내용 |
| 2024-05-21 | 박유나 | 1.1 | 개발내용 | 4, 5단계 개발 내용, 각 단계별 개발 내용 수정사항 갱신, 추가사항 작성 |
| 2024-06-01 | 레일라 | 1.2 | 개발/마무리 | 4, 6, 7 단계 개발 내용, 최종 코드 개선, 코드 구조화 |

**본 양식은 소프트웨어학부 C++프로그래밍 과목의 프로젝트 결과보고서 작성을 위한 기본 양식입니다. 문서의 필수 항목을 제시하는 것이니 폰트, 문단 구조 등의 디자인 부분은 자유롭게 설정하기 바랍니다. 양식 내에 붉은 색으로 기술한 부분은 지우고 작성하기 바랍니다.**

**목 차**

[1 개요 4](#_gjdgxs)

[2 개발 내용 및 결과물 5](#_30j0zll)

[2.1 목표 5](#_1fob9te)

[2.2 개발 내용 및 결과물 6](#_3znysh7)

[2.2.1 개발 내용 6](#_2et92p0)

[2.2.2 시스템 구조 및 설계도 6](#_tyjcwt)

[2.2.3 활용/개발된 기술 6](#_3dy6vkm)

[2.2.4 현실적 제한 요소 및 그 해결 방안 6](#_1t3h5sf)

[2.2.5 결과물 목록 7](#_2s8eyo1)

[3 자기평가 8](#_17dp8vu)

[4 참고 문헌 8](#_3rdcrjn)

[5 부록 8](#_26in1rg)

[5.1 사용자 매뉴얼 8](#_lnxbz9)

[5.2 설치 방법 8](#_35nkun2)

# 개요

|  |
| --- |
| **평가기준 (10점)**  **프로젝트를 완성하기 위해 사용한 개발 방법을 기술하세요.**  **또한 사용하고 있는 외부 라이브러리와 해당 라이브러리를 획득/설치하는 방법을 기술하세요.** |

본 프로젝트는 윈도우 개발환경에서 비주얼 스튜디오를 사용하여 c++ 언어로 구현하였습니다.

프로젝트의 파일은 소스파일인 **Food.cpp, Gate.cpp, GateEmergeEnablePos.cpp, main.cpp, Map.cpp, Snake.cpp, UserInfo.cpp**와, 헤더파일인 **Gate.h, GateEmergeEnablePos.h, Map.h**로 이루어져 있습니다. 게임에서 사용되는 맵은 Map.h 파일에 3차원 배열을 선언 및 초기화 하여 임의로 만들어 놓았습니다.

main.cpp 파일에서 전반적인 게임을 관리하도록 하였습니다. 그렇기에 **Windosw.h, iostream, vector, ctime** 등의 라이브러리를 include 하였고, 직접 만든 클래스의 소스 파일과 헤더 파일인 GateEmergeEnablePos.h, Gate.h, Snake.cpp, Food.cpp, UserInfo.cpp, Map.h 를 include 하였습니다.

즉, main.cpp 파일 속에, 위에 나열되어 있는 클래스들의 인스턴스를 만들어 사용하였습니다. main() 함수 안에 선언해놓은 변수들의 목록으로는 다음과 같습니다. 다음의 변수들을 main() 함수내에서 이용합니다.

-------------------------------------------------------------------------------------------------------------

int main() {

// 게임 설명과 함께 시작

displayStartTable();

// 맵 생성

Map mapInstance;

mapInstance.setMap();

// UserInfo 색상 설정

UserInfo userInfo;

userInfo.ShowInfo();

userInfo.Mission(10, 5, 3, 2);

// 아이템 생성

Food food;

// 사용자 정보 생성

UserInfo Info;

// 아이템의 아이디 랜덤 변수

random\_device randomId;

mt19937 genItemId(randomId());

uniform\_int\_distribution<int> idDis(3, 6);

// 아이템의 위치 랜덤 변수

random\_device randomPosX;

random\_device randomPosY;

mt19937 genItemPosX(randomPosX());

mt19937 genItemPosY(randomPosY());

uniform\_int\_distribution<int> posXDis(2, 44);

uniform\_int\_distribution<int> posYDis(2, 21);

// 시간 관련 변수

clock\_t startTime = clock();

clock\_t tempTime{};

clock\_t itemEmergeTime{};

clock\_t gateEmergeTime{};

// 지렁이 생성

// 맨 처음 위치는 10, 10 으로..

Snake snake(10, 10);

// 게이트 생성

Gate gate1(2, 0);

Gate gate2(2, 22);

//GateEmergeEnablePos prevGate1(2, 0);

//GateEmergeEnablePos prevGate2(2, 23);

gate1.setGateEmergeVec(mapInstance.map, mapInstance.mapLevel);

gate2.setGateEmergeVec(mapInstance.map, mapInstance.mapLevel);

bool gateMapStarted = false;

// 게이트 위치 랜덤 변수

random\_device randomVecIndex;

mt19937 genVecIndex(randomVecIndex());

//uniform\_int\_distribution<int> vecIndexDis(0);

// Mission 초기값

int BGoal = 4;

int PlusGoal = 2;

int MinusGoal = 1;

int Ggoal = 1;

int Tick = 150;

스네이크 게임의 가장 기본이자 핵심이 되는 함수들은 개요에서 먼저 소개합니다 (세부적인 내용의 함수들은 개발 내용 및 결과물에 작성하겠습니다).

**[가장 기본이자 핵심인 함수]:**

1. Map 클래스의 draw() 함수:

이 함수는 맵의 변경 사항이 생길 때마다 main.cpp 파일의 main 함수에서 호출하는 함수입니다.

1. Food 클래스의 setFoodInfo(int x, int y, int id) 함수:

이 함수는 main.cpp 파일에서 호출되어 미리 main.cpp 파일에서 선언해놓은 food 인스턴스 의 아이템 위치와, 아이템 번호를 업데이트 하는 역할의 함수입니다.

1. Map 클래스의 changeMapItemState(int mapLevel, int x, int y, int prevX, int prevY, int itemId) 함수:

이 함수는 main.cpp 파일에서 호출되어 현재 맵의 아이템 상태를 바꾸는 역할을 하는 함수 입니다.

1. Map 클래스의 changeMapGateState(int mapLevel, GateEmergeEnablePos gate1, GateEmergeEnablePos gate2, int gate1PrevPosX, int gate1PrevPosY, int gate2PrevPosX, int gate2PrevPosY) 함수:

이 함수는 main.cpp 파일에서 호출되어 현재 맵의 게이트 상태를 바꾸는 역할을 하는 함수 입니다.

1. Gate 클래스의 setGateInfo (int map[][23][46], int mapLevel, Gate\* gate) 함수:

이 함수는 main.cpp 파일에서 일정 시간이 지나면 호출되어, main.cpp 파일에 만들어놓은 gate 인스턴스의 정보를 업데이트 하는 역할을 하는 함수입니다. Gate 클래스가 가지는 변 수인 bool 형 타입의 horizontalBlocked, verticalBlocked 의 값을 업데이트하여 게이트의 양 옆이 막혀있는지, 위아래가 막혀있는지의 여부를 저장합니다.

1. Gate 클래스의 setGateEmergeVec (int map[][23][46], int mapLevel) 함수:

이 함수는 main.cpp 파일에서 호출되어 매개변수로 주어진 맵의 현재 상태를 반영하여, 다 음으로 게이트가 생길 수 있는 위치를 저장하는 역할을 합니다.

1. Snake 클래스의 snakeTouchFreshApple() 함수:

이 함수는 main.cpp 파일에서 호출되어 뱀의 머리가 싱싱한 사과와 부딪혔을 때, 뱀의 꼬리 부분이 늘어나도록 동작하는 함수입니다.

1. Snake 클래스의 snakeTouchPoisonApple() 함수:

이 함수는 main.cpp 파일에서 호출되어 뱀의 머리가 썩은 사과와 부딪혔을 때, 뱀의 꼬리 부 분이 줄어들도록 동작하는 함수입니다.

1. Snake 클래스의 snakeTouchGate (Gate\* gate1, Gate\* gate2, int snakeMoveX, int snakeMoveY, Snake\* snake) 함수:

이 함수는 main.cpp 파일에서 호출되어 뱀의 머리가 게이트의 직전 부분에 도달했을 때, 뱀 을 다른 게이트로 내보내는 역할을 하는 함수입니다.

1. Snake 클래스와 main.cpp 파일에 정의된 gotoxy (itn x, int y) 함수:

콘솔창의 커서를 주어진 매개변수의 위치로 이동시키는 역할을 하는 함수입니다.

1. UserInfo 클래스의 ShowInfo() 함수:

사용자의 현재 스테이지의 점수를 보여주는 함수입니다. 뱀의 현재/최대 길이, 아이템 획득 횟수, 게이트 사용 횟수를 보여줍니다.

1. UserInfo 클래스의 Mission() 함수:

현재 스테이지의 각 항목별 미션 값을 랜덤으로 받아 사용자가 미션을 클리어 했는지 출력하는 함수입니다.

# 개발 내용 및 결과물

## 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 목표:** 스네이크 게임의 근간이 되는 맵을 만드는 것(23 X 46 사이즈의 배열을 만듦).

**2단계 목표:** 프로젝트에서 주어진 게임룰을 바탕으로 뱀이 맵 상에서 움직이도록 만드는 것(키 보드에서 입력 받은 키값을 토대로 스네이크의 이동방향과 이동값의 변수 값을 업 데이트하고 업데이트 된 정보를 이용하여 뱀이 화면 상에서 움직이는 것처럼 보이 도록 만든다.)

**3단계 목표:** 맵에 아이템(싱싱한사과, 썩은사과)을 추가하고, 이 아이템이 뱀과 상호작용하도록 만드는 것(아이템의 위치를 나타내는 Food 클래스의 변수들을 이용하여 맵 상에 아이템이 생기고 없어지도록 한다.), 일정한 시간이 지나면 아이템의 정보가 업데이트 됨.

**4단계 목표:** 맵에 게이트를 추가하고, 이 게이트가 뱀과 상호작용하도록 만드는 것(게이트가 가 지는 위치 변수, 게이트로 들어오는 뱀의 방향을 저장하는 변수, 게이트가 맵 가장 자리에 있는지 내부에 있는지 여부를 저장하는 변수, 게이트에 뱀이 들어왔는지 여 부를 저장하는 변수 등을 이용하여, 맵 상에 게이트가 생기고 없어지도록 하고, 맵 에 새로 나타난 게이트의 정보를 위의 변수에 저장함). 일정 시간이 지나면 게이트 의 정보가 업데이트 됨.

**5단계 목표:** 사용자 정보와 미션정보를 우측에 출력하는 것. 또한 미션을 클리어 할 시 새로운 스테이지로 진입하도록 함.

**6 단계 목표:** 스네이크 게임용 컬러 맵 개발하기. 음식, 게이트, 맵, 사용자 정보, 미션 정보 등의 요소에 색상을 추가한다.

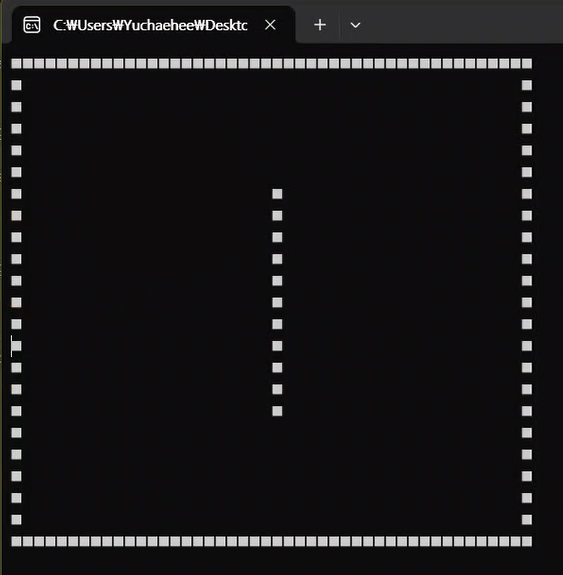
**7 단계 목표:** 스네이크 게임용 시작 창 개발하기.

## 개발 내용 및 결과물

### 개발 내용

|  |
| --- |
| **작성요령 (10점)**  **프로젝트의 수행의 내용을 구체적으로 기술한다. 세부 목표별로 어떤 결과를 어떤 방법으로 달성하였는지를 자세히 기술한다.** |

**[1단계(맵 구현)]:**



스네이크 게임의 맵을 만들기 위해 Map 클래스를 따로 만들었다. Map 클래스가 맵의 사이즈를 저장할 수 있도록 가로길이와, 세로길이를 나타내는 변수와, 맵의 상태를 나타내기 위한 3차원 배열을 선언 및 초기화 하였다. 가로길이는 46, 세로 길이는 23 으로 설정하였다.

맵을 3차원 배열로 선언한 이유는, 맵을 기본맵, 아이템 등장맵, 아이템&게이트 등장맵으로 구분하여 사용하기 위함이었다.

Main.cpp 파일의 main() 함수에서 Map 클래스의 인스턴스를 mapInstance 라는 이름으로 생성했다. 처음 인스턴스를 생성하고 난 후 바로 Map 클래스의 setMap() 함수를 호출하여 콘솔에 맵을 출력하였다.

Map 클래스에는 void setMap(), void gotoxy(int x, int y), void draw() 함수를 만들어 놓았다. SetMap 함수는 Map 클래스의 draw 함수를 호출하는 역할을 한다. draw 함수는 콘솔에 맵을 출력하는 역할을 한다. 이중포문을 돌면서 map 의 각 요소가 0, 1, 2, 3, 4, 5 중 무엇인지를 조건문을 통해 판단하여출력을 서로 다른 기호로 해주었다(0: 아무것도 출력X, 1&2:■, 3:★, 4:⊙, 5:♡).

------------------------------------------------------------------------------------------

0: 아무것도 없는 상태

1: 벽

2: 바뀔 수 없는 벽

3: 사과(먹으면 사이즈가 1 늘어남)

4: 독사과(먹으면 사이즈가 1 줄어듦)

5: 게이트

-----------------------------------------------------------------------------------------

처음엔 gotoxy 함수가 필요할 줄 알고 만들었는데, 반복문을 돌면서 출력하면 되는 것이라 사용하지 않았다. 삭제해도 상관없는 함수이다.

다만 이후3단계에 있을 item 종류의 추가사항 구현을 위해 SetMap 함수가 호출하는 draw 함수의 내용을 (0: 아무것도 출력X, 1&2:■, 3:★, 4:⊙, 5:♣, 6:§, 7:♡) 로 수정하였다.

------------------------------------------------------------------------------------------

0: 아무것도 없는 상태

1: 벽

2: 바뀔 수 없는 벽

3: 사과(먹으면 사이즈가 1 늘어남)

4: 독사과(먹으면 사이즈가 1 줄어듦)

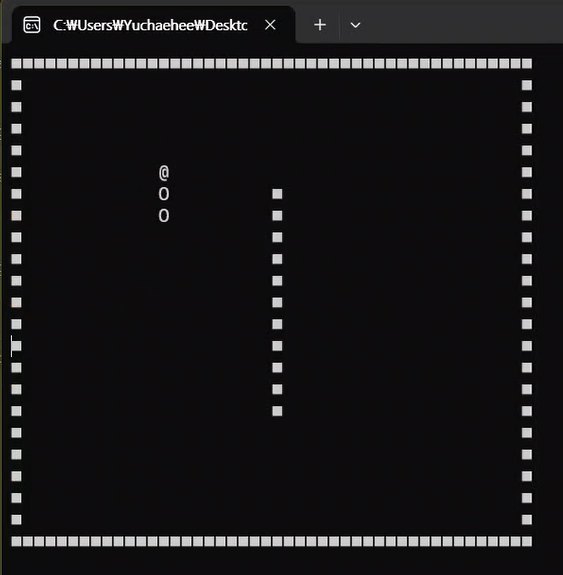
5: 황금사과(먹으면 사이즈가 2 늘어남)

6: 거꾸로사과(뱀의 방향을 바꿈)

7: 게이트

------------------------------------------------------------------------------------------

**[2단계(스네이크 움직임 구현 && 벽과 상호작용)]:**



**1. 스네이크 움직임 구현:**

스네이크를 구현하기 위해 Snake, SnakeHead, SnakeBody 클래스를 만들었다.

SnakeBody 는 Snake 클래스에서 vector 자료형을 통해 관리하고, SnakeHead 는 Snake 클래스에서 단일 변수로 관리하도록 하였다.

방향키를 누르면 main.cpp 의 main() 함수에서 snake 클래스의 setPos(snakeDirection dir) 함수를 호출하도록 했음. 이때, snakeDirection 은 Snake 클래스에 직접 정의한 enum 타입으로, LEFT=0 RIGHT=1 UP=2 DOWN=3 으로 설정했음. setPos(snakeDirection) 함수는 입력된 방향키에 따라 snake 클래스의 moveX, moveY의 값을 바꾸어 뱀이 이동하도록 하는 역할을 하도록 하였다.

setPos(snakeDirection) 함수로 snake 의 이동방향과 moveX, moveY 값을 설정한 후, snake 클래스의 move(int x, int y) 함수를 호출함. snake 클래스의 move 함수는 전체적인 뱀의 이동을 관리하도록 하였다.

SetPos 함수에서는 현재 뱀의 이동 방향과 반대되는 키가 입력되면 게임을 종료하도록 switch 문을 통해 주어진 입력에 따라 case 속으로 들어간 후, 그 속에서 조건문을 걸어 현재 뱀의 이동방향과 반대되는 값인지의 여부를 판단하도록 했다. 반대되는 값이라면 isDie=true 구문으로 변수의 값을 바꿔주고, return 문으로 함수를 바로 빠져나오도록 했다. 그리고 main() 함수로 돌아가면 바뀐 snake 의 죽음을 판별하는 변수의 값이 true 가 되었으므로 while 반복문을 break 해서 게임을 종료시키도록 했다.

이 함수에서 snakeHead 클래스의 move(int dx, int dy) 함수와 snakeBody 클래스의 move(int dx, int dy) 함수를 호출함. snakeHead 클래스의 move 함수는 snakeHead 의 위치를 전달된 매개변수의 값과 더해 다시 위치 변수에 대입하고, snakeBody 클래스의 move 함수는 body 의 위치 변수에 전달된 매개변수의 값을 대입함. snakeHead 의 위치를 덧셈연산을 수행하여 바꿔주면, snakeBody 는 이동 전 뱀의 머리 위치로 이동하기만 하면 되기 때문에 둘 함수의 행동을 다르게 구현하였음.

snake 클래스의 move 함수 마지막에는 removeTail() 과 draw() 함수가 호출되는데 removeTail 은, 미리 저장해둔 뱀의 꼬리 위치로 이동하여 공백을 출력해서, 뱀이 이동하는 것처럼 보이게 하기 위함이었다. draw 함수는 전체적인 뱀의 위치 이동이 끝나면 비로소 호출되어 뱀의 사이즈만큼 반복문을 돌아 커서를 이동시켜 출력하는 것이다. 즉, removeTail() 과 draw() 를 통해서 커서를 계속 움직이면서 뱀을 콘솔 창에 프린트해주었다.

게임 진행 도중 뱀의 머리랑 뱀의 몸통이 부딪히면 게임을 종료하도록 했다. 그러나 snake.moveX==0, snake.move==0 조건이 동시에 만족될 때(== 게임 시작 직후 방향키를 누르기 전 상태) 에는 머리랑 몸이 닿아도 죽으면 안 되므로 따로 조건문을 걸어주었다.

게임 시작 직후 방향키를 누르기 전 상태가 아니라면, 뱀의 사이즈-1 만큼 반복문을 돌면서 뱀의 머리와 뱀의 몸의 위치가 같은지를 판단하도록 했다. 만약 같다면 snake.isDie=true 구문을 통해 while 문을 빠져나와 게임을 종료하도록 했다.

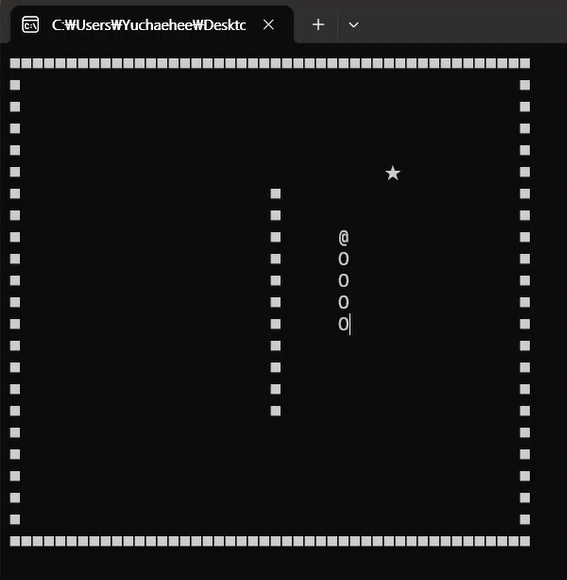
**2. 스네이트와 벽의 상호작용):**

게임 진행 도중 뱀의 머리랑 맵의 벽이 부딪히면 게임을 종료하도록 했다. Snake 의 snakeTouchWall 함수를 호출하여 만약 뱀의 머리 위치가 맵의 사이즈를 벗어나면 isDie 변수를 true 로 업데이트 하여 게임을 종료하도록 했다.

**3. Tick에 대한 변화 주기, 추가사항:**

뱀의 이동속도를 snake 클래스의 tick(int tick) 함수를 통해 조절한다. main 함수에서 tick 함수를 호출하여 tick함수 내의 sleep 함수를 통해 뱀 이동 속도를 정하고, 뱀의 이동 속도가 변하는 조건은 사용자가 스테이지를 클리어 할 때마다 main 함수 내의 시간 변수의 값을 줄여 이 값을 tick 함수의 매개변수로 보내 뱀의 이동속도가 증가하도록 했다.

**[3단계(아이템 요소 구현 && 뱀과 아이템 상호작용 구현)]:**



**1. 아이템 요소 구현:**

아이템을 구현하기 위해 Food 클래스를 만들었다. Food 클래스는 아이템 번호를 저장하는 foodId(3: 싱싱한 사과, 4: 썩은 사과) 변수와, 아이템의 위치를 저장하는 foodPosX, foodPosY 변수와, 이전 아이템의 위치를 저장하는 prevPosX, prevPosY 변수를 선언하였다.

SetFoodInfo(int x, int y, int id) 라는 함수를 만들어서 foodId 와 foodPosX, foodPosY 의 값을 업데이트 하는데 사용하였다.

Main.cpp 파일의 main() 함수는 startTime, tempTime, itemEmergeTime 의 시간변수를 가지는데 게임이 시작 된지 5초가 지나면 맵 레벨이 1로 증가하도록 했다.

맵 레벨이 1 또는 2이고, tempTime 과 itemEmergeTime 의 시간차가 7이면 조건 문이 만족되어서 아이템의 상태를 바꾸는 로직이 실행되도록 하였다, itemEmergeTime 에 tempTime 값을 다시 대입하여, 7초 차이가 날 때마다 계속해서 조건문이 실행될 수 있도록 하였다.

조건문에 진입하면 아이템이 새로 생성될 위치를 랜덤으로 만들도록 하였다. 그런데 만약 랜덤으로 얻은 위치로 map 변수에 접근했을 때, 그 값이 1(벽) 이라면 위치를 새로 받아야 하므로, 새로 받은 위치의 map 변수 요소의 값이 1이 아닌 0이 될 때까지 while 문을 돌도록 했다.

While 문을 빠져나온 후 main() 함수에 만들어 놓은 Food 클래스의 인스턴스의 setFoodInfo 함수를 실행시키도록 했다. 이때 매개변수로 새로 얻은 랜덤 위치값과, 랜덤 아이템 아이디(3 또는 4)를 전달해주었다.

아이템의 정보가 바뀌었으니, 이를 맵에 반영을 시켜주기 위해서 미리 만들어 놓은 Map 인스턴스인 mapInstance 의 chanageMapItemState 함수를 호출하였다.

changeMapState(int mapLevel, int x, int y, int prevX, int prevY, int itempId) 함수는 map[mapLevel][prevY][prevX]=0; 구문을 통해 이전 위치에 있던 음식을 없애고, map[mapLevel][y][x]=itemid; 구문을 통해 새로 입력 받은 x, y 위치에 아이템을 놓도록 구현하였다.

맵의 상태 변화를 마친 후 다음으로 mapInstance 의 setMap() 함수를 호출하여 콘솔창에 맵을 새로 그리도록 하였다.

**2. 뱀과 아이템 상호작용 구현:**

뱀과 아이템의 상호작용을 구현하기 위해 Snake 클래스에 snakeTouchFreshApple() 함수와 snakeTouchPoisonApple() 함수를 만들었다.

SnakeTouchFreshApple 함수는 게임 진행도중 뱀의 머리가 싱싱한 사과랑 부딪히면(==뱀의 머리와 싱싱한 사과의 위치가 같으면) 호출되는 함수로, 뱀의 사이즈를 1 늘려주고, body 벡터에 새로운 snakeBody 인스턴스를 생성하여 넣도록 했음

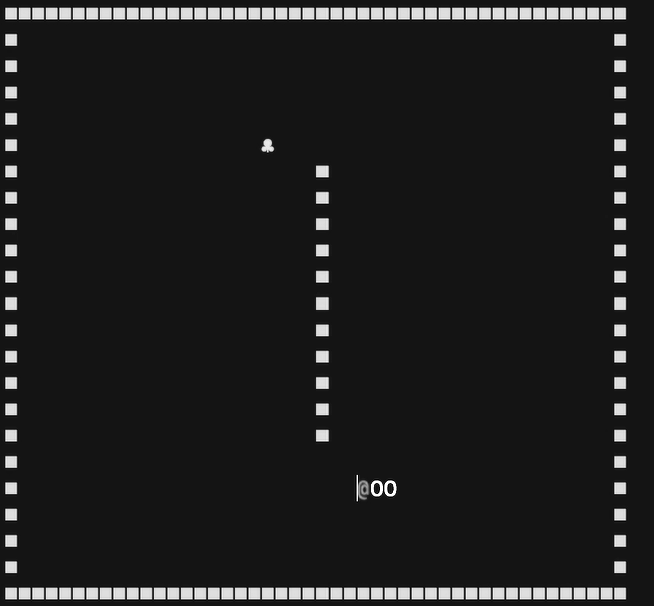
(\*이때 새로운 인스턴스의 위치는 꼬리 쪽이 늘어나야 하므로, 현재 뱀의 이동 방향과 반대되는 곳으로 늘어나도록 했다.

즉, 현재 뱀의 이동방향이 LEFT 일 때, moveX=-1, moveY=0 이므로 이 값에 -를 곱해서 각각 1, 0 으로 만들어 준 후 이를 새로운 인스턴스를 body 에 넣기 전 body 의 마지막에 있던 인스턴스의 위치에 더해주도록 해서 맨 마지막에 꼬리가 하나 붙도록 구현했다.)

SnakeTouchPoisonApple 함수는 게임 진행 도중 뱀의 머리가 썩은 사과와 부딪히면(==뱀의 머리와 썩은 사과의 위치가 같으면) 호출되는 함수로, 뱀의 사이즈를 1 감소시키고, 맨 마지막 꼬리 위치로 가서 공백을 출력하여 꼬리가 잘린 것처럼 보이도록 했다. 최종적으로 body.pop\_back() 구문을 통해 실질적으로 꼬리가 잘리도록 했다.

만약 뱀의 사이즈가 3보다 작아지면 snake.isDie = true 구문을 통해 while 문을 벗어나 게임을 종료하도록 했다.

**3. Item의 종류에 대한 추가. 추가사항:**



새로운 아이템 2개를 추가했다.

먹으면 뱀의 길이가 2 증가하는 황금사과(♣), 먹으면 뱀의 꼬리와 머리의 위치가 바뀌어 진행방향이 현재 진행방향과 반대로 바뀌는 거꾸로 사과(§) 을 구상함.

앞서 설정해둔 foodid를 각각 5,6 으로 설정. 그리고 뱀이 이 아이템과 상호작용 하기 위해 Snake 클래스에 snakeTouchGoldApple() 함수와 snakeTouchChangeApple(Snake\* snake) 함수를 추가했다.

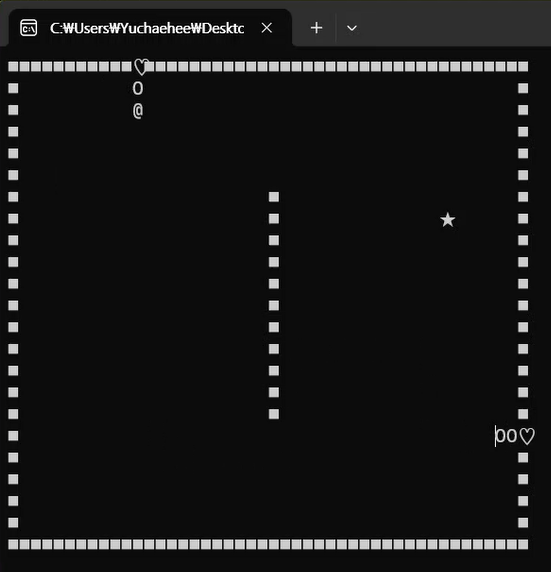
snakeTouchGoldApple 함수는 게임 진행 도중 뱀의 머리가 황금 사과와 부딪히면 호출되는 함수로, 뱀의 사이즈를 2 증가시키고, SnakeTouchFreshApple 과 동일한 방식인 body 벡터에 새로운 snakeBody 인스턴스를 생성하여 넣는 작업을 2번 진행하도록 했다.

snakeTouchChangeApple 함수는 게임 진행 도중 뱀의 머리가 거꾸로 사과와 부딪히면 호출되는 함수로, 뱀의 머리와 꼬리의 위치를 바꾸고 진행방향을 현재 진행방향과 반대가 되는 방향으로 바꾸는 함수이다. 매개변수로 현재 머리 위치와 스네이크를 받아오고, 머리를 teleportMove() 함수를 이용해 스네이크 꼬리 위치로 이동시키고, 꼬리에는 (body.back().x,body.back().y ) 에는 머리에 해당하는 위치를 넣어주도록 했다. 이때 teleportMove 함수는 Snake 클래스가 가지는 함수로 teleportMove() 함수에 이미 head.teleportMove(x, y); 가 선언되었기에 teleportMove(tempTailX, tempTailY);로 선언해주고, 꼬리 부분에 머리에 해당하는 위치를 넣어준다.

다만 여기서 그냥 하면 꼬리 부분이 떨어져 남게 되는데 이는 removeTail();을 호출하여 꼬리 부분을 공백으로 제거해주어 마치 바로 바뀌는 것 처럼 출력할 수 있다.

또한 현재 이동방향을 바꾸어주기 위해. 각각 방향에 따른 케이스를 if문으로 나누어 주고, (\*snake).setPos(Snake:://바꾸고 싶은 방향); 을 이용해 방향을 바꾸었다.

**[4단계(게이트 요소 구현 && 뱀과 게이트 상호작용 구현)]**



**1. 게이트 요소 구현:**

게이트를 구현하기 위해 Gate 클래스와 GateEmergeEnablePos 클래스를 만들었다.

Gate 클래스는 게이트의 위치를 저장하는 gatePosX, gatePosY 변수와, 이전 게이트의 위치를 저장하는 prevGatePosX, prevGatePosY 변수를 선언했다.

GateEmergeVec 이라는 이름의 GateEmergeEnablePos 클래스의 인스턴스를 담는 벡터 변수도 선언하였다.

Enum snakeInputDir 라는 이넘타입을 새로 정의하였는데, LEFT=0, RIGHT=1, UP=2, DOWN=3 을 의미하도록 했다.

뱀이 게이트의 어느 방향으로 접근할 수 있는 지의 상태를 저장하기 위해 horizontalBlocked, verticalBlocked 변수를 선언하였고, 뱀이 게이트에 들어왔는지 여부를 확인하기 위한 변수도 선언하였다(bool 타입).

게이트가 가장자리에 있는 지의 여부를 확인하기 위해 isEdgeGate 라는 변수를 선언하였다. 그리고 currSnakeInputDir 라는 이름의 snakeInputDir 타입 변수도 선언하였다.

GateEmergeEnablePos 클래스는 gateEmergeEnablePosX, gateEmergeEnablePosY 라는 변수를 가지도록 하였다.

Main.cpp 파일의 main() 함수는 startTime, tempTime, gateEmergeTime 의 시간변수를 가지는데 맵 레벨이 2이고 tempTime 과 gateEmergeTime 이 10 차이가 나면 조건문에 진입하여 게이트의 정보를 바꾸는 로직을 실행하도록 하였다.

조건문에 진입하면 gateEmergeTime 에 tempTime 의 값을 대입하여 10 차이가 날 때마다 조건문에 들어올 수 있도록 하였다.

조건문에 들어오면 또 하나의 조건문을 마주하게 되는데, 이는 미리 생성해놓은 게이트 클래스의 인스턴스의 isSnakeMoved 변수가 true 면 진입할 수 없도록 하여 뱀이 게이트 이동을 하지 않을 때만 게이트의 정보를 바꾸는 로직을 수행하도록 하였다.

게이트가 새로 생성될 수 있는 위치의 값들은 게이트 인스턴스의 gateEmergeVec 이라는 벡터에 저장되어 있다(한 쌍의 게이트의 벡터 값은 같음). 즉, 현재 gateEmergeVec 의 사이즈만큼 랜덤의 범위로 설정하여 랜덤으로 인덱스 값을 만들도록 했다.

랜덤으로 만들어진 인덱스 값으로 gateEmergeVec 의 요소에 접근해서 현재 게이트의 위치를 gateEmergeVec 요소의 값으로 업데이트 하도록 했다. 그런데 만약 랜덤으로 만들어진 인덱스의 값이 같아서 게이트 두 개가 서로 같은 요소의 값으로 업데이트 되면 안 되도록 해야 하기 때문에 로직을 추가했다.

랜덤으로 주어진 인덱스의 값이 같으면 두 개의 인덱스 값이 달라질 때까지 while 문을 돌면서 하나의 게이트의 인덱스 값만 계속해서 바꿔주었다. 결국 두 개의 인덱스 값이 달라지면 while 문을 빠져나와서 비로소 위에서 설명한, 랜덤으로 만들어진 인덱스 값으로 gateEmergeVec 의 요소에 접근해서 현재 게이트의 위치를 gateEmergeVec 요소의 값으로 업데이트 하도록 했다.

다음으로는 mapInstance 의 changeMapGateState 함수를 호출하여 현재 맵의 게이트 상태를 업데이트 하도록 했다. 함수의 원형은 다음과 같다.

changeMapGateState(int mapLevel, GateEmergeEnablePos gate1, GateEmergeEnablePos gate2, int gate1PrevPosX, int gate1PrevPosY, int gate2PrevPosX, int gate2PrevPosY)

매개변수로 주어진 gate1PrevPosY, gate1PrevPosX, gate2PrevPosY, gate2PrevPosX의 값으로 mapInstance 의 map 배열의 요소에 접근하여 값을 1로 바꿔주었다(게이트가 다시 벽으로 바뀌도록).

다음으로gate1.gateEmergeEnablePosY, gate1.gateEmergeEnablePosX 변수로 mapInstance 의 map 배열의 요소에 접근하여 값을 5로 바꿔주었다(벽이 게이트로 바뀌도록). gate2 도 마찬가지이다.

게이트가 새로 생성되면 게이트의 정보도 업데이트 되어야 하므로 게이트 정보 세팅 함수를 따로 만들었다. 게이트의 양옆이 막혀있는지, 위아래가 막혀있는지 판단해서 정보를 업데이트 하는 로직을 수행한다. 이 역할을 하는 함수는 Gate 클래스의 setGateInfo 함수이다. 함수의 원형은 다음과 같다.

setGateInfo(int map[][23][46], int mapLevel, Gate\* gate)

일단 이 함수가 호출되면 게이트의 horizontalBlocked, verticalBlocked 변수를 false 값으로 업데이트 해주었다. 그 다음으로 게이트의 위아래가 막혀있는지, 양옆이 막혀있는지를 판단하여 horizontalBlocked 와 verticalBlocked 변수의 값을 업데이트 했다.

이렇게 게이트의 정보 변경이 총체적으로 끝난 후 mapInstance 의 setMap 함수를 호출하여 콘솔에 맵을 새로 그렸다.

위 함수가 실행된 다음으로 수행되는 로직은 시간이 지나면 게이트를 없애기 위해서 이전 위치에 현재 게이트의 위치를 저장하는 로직이다.

**2. 뱀과 게이트 상호작용 구현:**

만약 뱀의 머리 위치에 뱀의 이동 방향을 더한 값이 게이트의 위치와 같다면 Snake 클래스의 인스턴스인 snake 의 snakeTouchGate 함수를 호출하도록 하였다. 함수의 원형은 다음과 같다.

snakeTouchGate(Gate\* gate1, Gate\* gate2, int snakeMoveX, int snakeMoveY, Snake\* snake)

Gate1 은 뱀이 들어가는 게이트, gate2 는 뱀이 나가는 게이트, snakeMoveX 랑 snakeMoveY 는 현재 뱀이 이동하는 방향을 의미한다.

SnakeTouchGate 함수는 나가는 게이트가 맵 가장자리에 있는지, 맵 내부에 있는 지에 따라 수행되는 로직이 다르도록 조건문을 달아주었다. 뱀이 나가는 게이트가 맵 가장자리에 있을 때, 뱀의 머리가 이동해야 하는 위치는 뱀이 나가는 게이트 위치의 바로 전 위치이다.   
  
즉, 뱀이 나가는 게이트의 x 위치가 0이면(뱀이 나가는 게이트가 맵의 좌측 가장자리에 있으면) 뱀의 이동방향을 RIGHT 로 바꿔주는 식으로 로직을 작성했다.

다음으로는 (\*snake).teleportMove(1, (\*gate2).gatePosY) 구문이 실행된다. 이때 teleportMove 함수는 Snake 클래스가 가지는 함수이다. 이 함수는 뱀의 머리랑 게이트가 만나면 실행되도록 했다. 기존 Snake 클래스의 move 함수와 전체적인 로직은 같으나 쓰이는 함수가 다르다.

TeleportMove 함수 속에는 head.teleportMove(x, y); 라는 구문이 존재하는데, 이는 SnakeHead 클래스의 teleportMove(int dx, int dy) 함수를 호출하는 것이다. SnakeHead 클래스의 teleportMove 함수는 기존의 move 함수와는 다르게 뱀 머리의 위치 변수에 직접 매개변수의 값을 대입하도록 했다.

즉, 함수가 실행되면 Snake 인스턴스가 가지는 SnakeHead 클래스의 head 인스턴스(뱀 머리) 의 위치 변수의 값이 입력 받은 위치로 업데이트 된다.

뱀 머리의 위치가 변하면 뱀 몸체의 위치도 바뀌어야 하는데 이는 기존 move 함수와 동일한 로직을 사용한다. PrevX, prevY 에 뱀 머리가 움직이기 전의 위치를 저장해놓고, 뱀 몸체 사이즈만큼 반복문을 돌면서 tempX, tempY 라는 변수를 이용해 순차적으로 뱀 몸체의 위치를 바꿔주도록 했다.

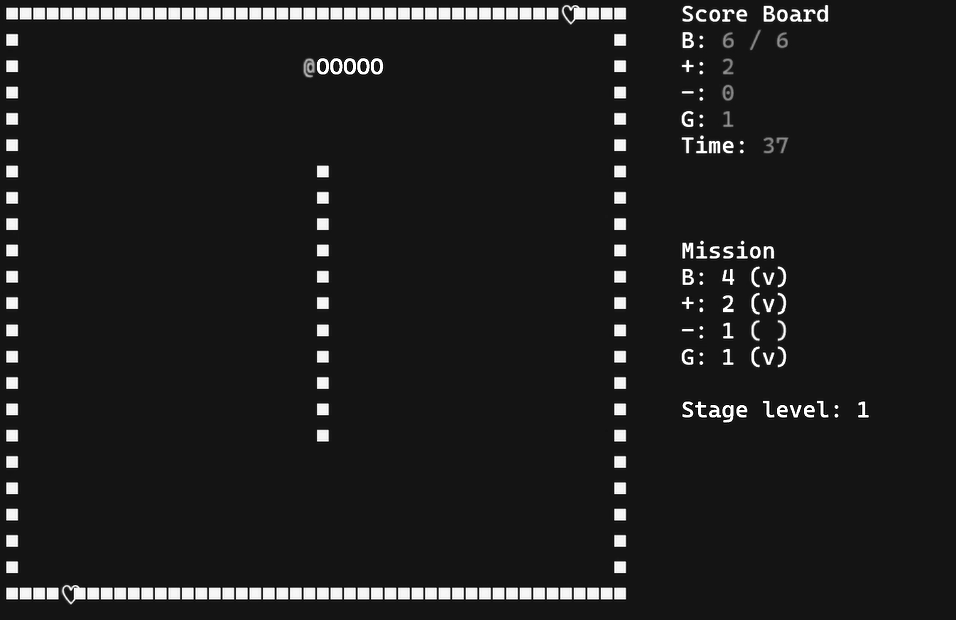
뱀의 꼬리가 나가는 게이트 바로 직전 위치에 도달할 때까지 뱀의 무한 while 반복문에서 move 함수를 호출한다. 도달하면 isSnakeMoved 변수의 값이 false 로 업데이트 되고 break 문으로 while 문을 종료한다.

즉, 이제 게이트의 위치를 바꿀 시간이 되었을 때 isSnakeMoved 변수의 값이 false 이므로, 게이트의 위치를바꾸는 로직이 수행되어 게이트의 위치가 바뀌도록 하였다.

뱀이 나가는 게이트가 맵 내부에 있는 경우에는 게이트 상태를 체크해야 한다. 나가는 게이트의 양옆과, 위아래가 막혀있지 않은 자유로운 상태, 나가는 게이트의 위아래가 막혀있는 상태, 나가는 게이트의 양옆이 막혀있는 상태와 같이 3가지의경우로 나누어서 조건문에 사용하도록 했다.

로직이 다 수행된 후, 나가는 게이트가 맵 가장자리에 있었던 경우와 마찬가지로 뱀의 꼬리가 나가는 게이트 바로 직전 위치에 도달할 때까지 뱀의 무한 while 반복문에서 move 함수를 호출하도록 했다.

**[5단계(유저 게임 점수 화면 & 미션 정보 화면)]:**



**1. 유저 게임 점수 화면 구현:**

유저 게임 점수 화면을 구현하기 위해 UserInfo 클래스를 따로 만들었다. UserInfo 클래스에서는 현재 길이와 최대길이, 아이템 먹은 개수, 현재 시간, 목표별 달성 여부 확인을 위한 bool 형 배열, 정보 출력 시작 좌표를 편하게 쓰기 위한 변수, 각 항목별 목표가 전부 달성되었는지 확인하는 bool형 변수들을 관리하도록 하였다.

ResetInfo() 함수는 목표 달성 시 현재 점수를 초기화하기 위한 함수로, 변수 값들을 초기화한다. setCurrentLength() 함수는 현재 길이를 지정하는 멤버 함수이다. setMaxLen() 함수는 최대 길이를 지정하는 멤버 함수이다. setGrowthItem, setPoisonItem 각각은 아이템 관련하여 지정하는 멤버함수이다.setGateUse() 는 게이트 사용 카운트를 지정하는 멤버함수이다.

IncreaseGetGrowthItem(), IncreaseGetPoisonItem(), IncreaseGateCount() 함수는 각각 성장 아이템, 감소 아이템을 먹었을 때, 게이트를 사용했을 때 필요한 함수로, 해당 카운트를 늘려준다.

이렇게 변수들을 지정하고 값을 증가하는 함수들을 이용한 후, ShowInfo() 함수를 이용해 현재 점수를 출력한다.

**2. 미션 정보 화면 구현:**

미션 정보는 UserInfo 클래스의 Mission(int Bgoal, int Pgoal, int MinGoal, int Ggoal) 함수를 통해 출력이 된다. 이 함수 내에는 PrintMissionInfo 함수가 각 목표마다 호출이 되고 있다.

PrintMissionInfo(const string& label, int current, int goal, int goalIndex) 함수는 현재 목표 이름, 현재 값, 목표 값, 목표 인덱스를 매개변수로 받는다.

이 함수에선 GetYPosition 함수가 호출되는데 이는 단순히 목표를 출력 할 y좌표 위치를 편하게 받아오게 하기 위한 함수이다.

PrintMissionInfo 함수는 만약 현재 값이 목표값보다 작으면 아직 목표 달성이 되지 않은 것으로 ( ) 을 출력하고, 만약 현재 값이 목표값과 같거나 크면 목표를 달성한 것으로 여겨 (v) 을 출력하고,

goalsCompleted 배열의 현재 목표 인덱스에 해당하는 부분을 true로 바꾸어준다. 그리고 만약 이 함수를 호출했을 때 goalsCompleted 해당 목표의 인덱스에 해당하는 부분이 true라면 (v) 을 호출하고 함수를 리턴한다. 만약 목표를 달성하지 못했다면 앞서 말한 목표 값과 현재 값을 비교하여 달성 여부를 출력하는 형식이다.

**3.미션 클리어 시 다음 스테이지 진입:**

UserInfo 클래스의 Mission 함수 안에서 스테이지 클리어 여부를 정한다. 이는 for문을 이용하여 판별한다. bool 형 배열 goalsCompleted 는 앞서 PrintMissionInfo() 함수에서 목표를 달성 할 시, 목표에 해당하는 인덱스 부분이 true로 바뀐다.

먼저 IsComplete 변수를 true로 선언해주고, goalsCompleted 를 for문으로 돌면서 만약 하나라도 false가 있으면 IsComplete 를 false로 바꾸고 아직 모든 목표가 달성되지 못한 것으로 판단한다. 만일 for문을 돌면서 false값이 없어 IsCompelet 변수가 여전히 true로 남아있다면 모든 목표를 달성한 것으로 판단한다. 그리고 IsGoal() bool형 함수에서 IsCompelet 의 값을 return하여 main함수에서 목표가 달성되었는지 판단할 수 있도록 하였다.

main함수에서 IsGoal() 함수를 호출했을 때 true라면 mapLevel을 높여 스테이지를 업데이트한다. 또한 현재 정보 값을 리셋하고, 스네이크의 길이도 원래 값인 3으로 초기화한다.

또한 목표값을 각각 난수로 변경하여 각 스테이지마다 다른 목표 값을 가질 수 있도록 하였다.

또한 2단계의 추가사항의 구현을 위해 미션 달성시 뱀 속도를 감소하도록 추가하였다.

setMap 함수를 이용해 현재 맵 상태를 업데이트 하였다.

만일 클리어한 스테이지가 현재 가지고 있는 스테이지 수 보다 많다면 게임을 클리어 한 것으로 판단해 게임을 종료한다.

**4. Score Board 에 대한 변경 추가사항 :**

Score Board에 표현되는 요소를 추가했다. 플레이한 시간과 현재 스테이지 레벨을 출력하도록 했다. setTime() 는 현재 시간을 지정하는 멤버함수이다. main함수에서 호출하여 플레이한 누적 시간을 지정하고, ShowInfo() 함수를 통해 시간 값이 출력된다.

현재 스테이지 레벨은 main 함수에서 특별한 함수 없이 출력한다. maplevel을 받아 stagelevel 변수를 선언. 단 maplevel은 Map을 나타내는 3차원 배열에서의 인덱스를 위한 변수였기에 0,1,2는 같은 스테이지를 의미한다. 그렇기에 0,1,2 일때는 1로 통일하고, 나머지는 각각 1을 빼서 2,3,4 로 받았다. 그렇게 우측 하단에 스테이지 레벨을 출력하였고, 만약 사용자가 마지막 스테이지에 도달했다면 Last Stage 라고 출력하도록 하였다.

**[6단계(색상 설정)]:**



setColor() 함수는 색상 효과를 생성하는 데 사용되었다. 이 기능에는 게임의 색상 테마에 관한 모든 정보가 포함되어 있다. 이 기능의 도움으로 게임의 시각적 품질을 향상시키고 게임을 더욱 재미있게 만들 수 있었다. 이 게임에 사용된 색상은 다음과 같:

. 검정- 0

어두운 파랑- 1

어두운 초록 - 2

어두운 하늘 - 3

어두운 빨강 - 4

어두운 보라 - 5

어두운 노랑 - 6

회색 - 7

어두운 회색 - 8

파랑 - 9

초록 - 10

하늘 - 11

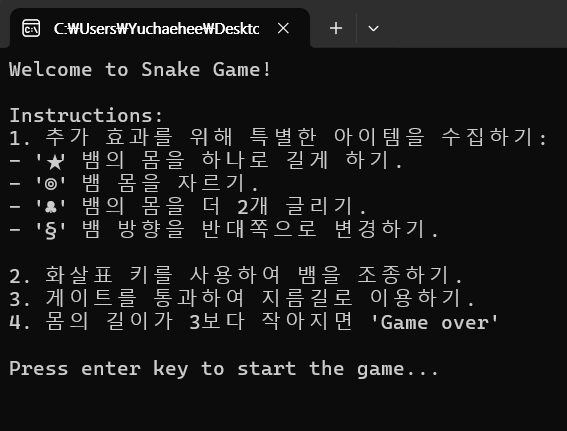
빨강 - 12

보라 - 13

노랑 - 14

하양 - 15

**[7단계(사용자 메뉴열)]:**



게임의 규칙을 정의하는 displayStartTable() 함수를 만들어졌다. 이 기능에는 게임의 모든 기본 규칙이 포함되어 있다. 덕분에 플레이어는 게임의 규칙을 편리하게 이용할 수 있게 된다.

### 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.** |

**[1단계]**

**개발자: 유채희**

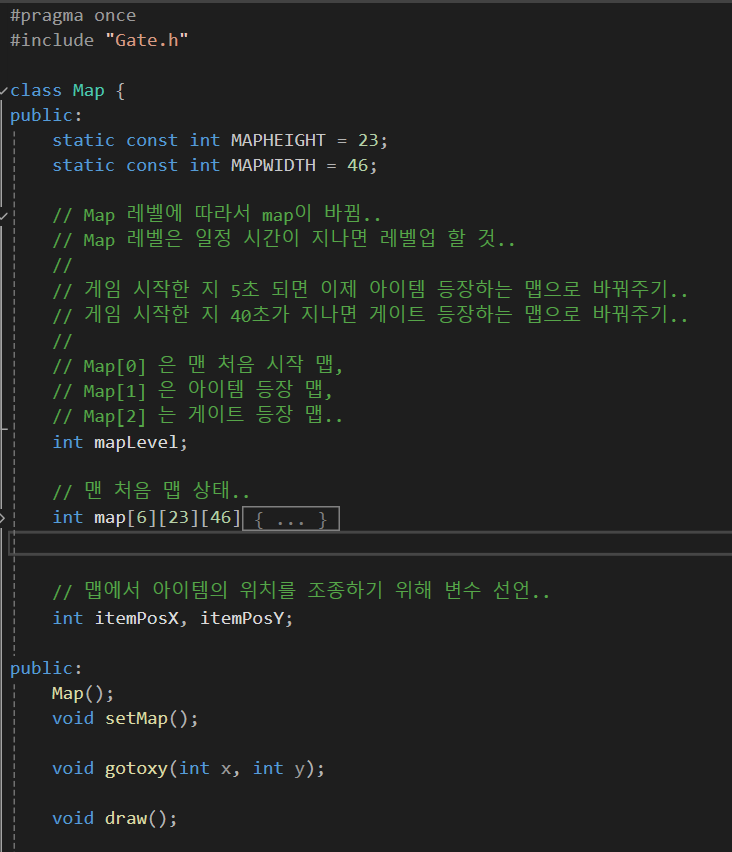
**사용한 소스 파일:** map.h, map.cpp

**사용한 클래스 및 함수:** Map 클래스, Map::setMap 함수



[Map 클래스의 draw 함수]:

이중 for 문을 돌면서 map 의 요소가 1, 2, 3, 4, 5,6,7 중 어디에 해당하는지 조건문을 통해 판단하고, 그에 맞는 기호를 출력하도록 하는 함수.



맵 클래스는 맵의 사이즈를 나타내기 위한 변수를 가진다(MAPHEIGHT, MAPWIDTH). 그리고 맵레벨을 나타내기 위한 변수를 가지고, 3차원 배열을 가지고 있다.

**[2단계]**

**개발자: 유채희**

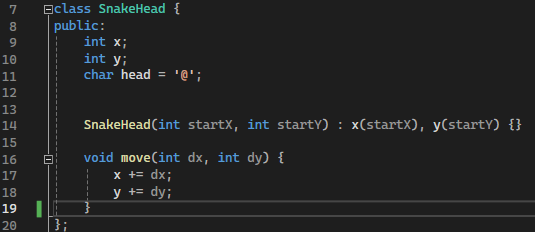
**사용한 소스 파일:** Snake.cpp

**사용한 클래스 및 함수:**

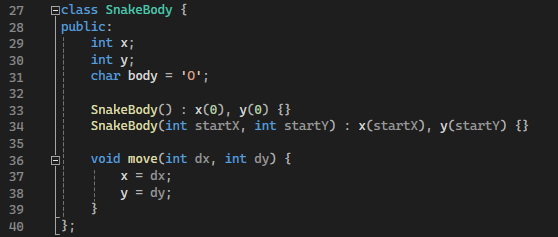
클래스 - SnakeHead, SnakeBody, Snake, Map

함수 - Snake::setPos, Snake::move, Snake::removeTail, Snake::draw, SnakeHead.move, SnakeBody.move

SnakeHead 클래스: 뱀의 머리 위치와, 뱀의 머리 모양을 가지고 있는 클래스. SnakeHead::move 함수는 Snake::move 함수에서 호출되는 함수이다(뱀의 머리 위치에, 뱀의 이동방향 만큼 값을 더해준다).



SnakeBody 클래스: 뱀의 몸통 위치와, 뱀의 몸통 모양을 가지고 있는 클래스. SnakeBody::move 함수는 Snake::move 함수에서 호출되는 함수이다(뱀의 몸통 위치의 값을 자기 앞에 있던 머리 또는 몸통의 위치로 바꾼다).

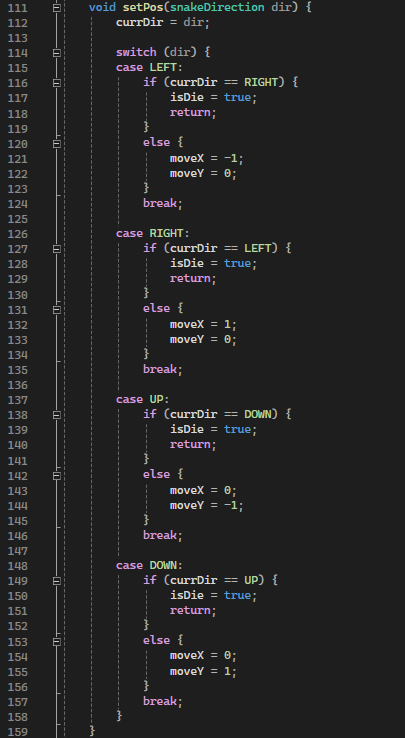


Snake 클래스: 뱀의 이동속도, 뱀 정보 초기화, 뱀의 이동 방향, 뱀의 머리 인스턴스, 뱀의 바디 인스턴스를 저장하는 벡터, 뱀의 생사 여부를 확인, 꼬리의 위치를 임시 저장하는 변수, 뱀의 크기, 뱀의 이동량 변수가 존재하는 클래스.



**Snake::setPos 함수:**

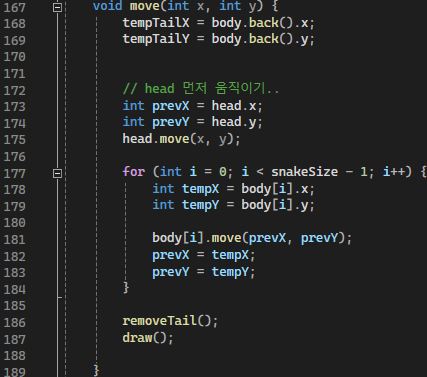
입력 받은 방향과 현재 뱀의 이동방향을 비교해서 서로 반대되는 값이면 isdie = true 구문을 통해 게임을 종료하도록 함. 서로 반대되는 값이 아니라면 뱀의 이동량을 입력 받은 방향에 따라 업데이트 해 준다.



**Snake::move 함수:**

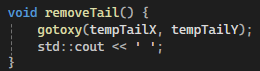
뱀을 이동시키는 함수. SnakeHead::move 함수와, SnakeBody::move 함수를 호출한다. 머리의 move 함수는 뱀의 머리 위치에 뱀의 이동량 만큼 더해줘서 뱀의 머리 위치를 업데이트하고, 바디의 move 함수는 뱀의 몸통 위치를 전달받은 매개변수의 값으로 업데이트 한다. 몸통의 move 가 받는 매개변수는 자기 몸통 이전 위치 머리 또는 몸통의 위치이다.

뱀의 꼬리 부분을 저장해 놓고 removetail 함수를 통해 꼬리를 자른후, draw 함수를 호출하여 콘솔창에 뱀을 출력한다.



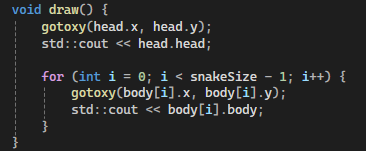
**Snake::removeTail 함수:**

맨 마지막 꼬리의 위치로 커서를 움직여서 공백을 출력한다.



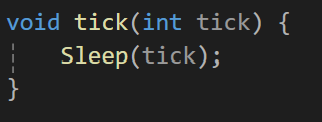
**Snake::draw 함수:**

뱀의 머리 위치로 커서를 움직여 머리를 먼저 출력하고, 뱀의 전체 사이즈-1 만큼 반복문을 돌면서 뱀의 몸통 위치로 커서를 움직여 몸통을 출력한다.



**Snake::tick 함수:**

뱀의 이동속도를 sleep을 주어 조절하는 함수.



**[3단계]**

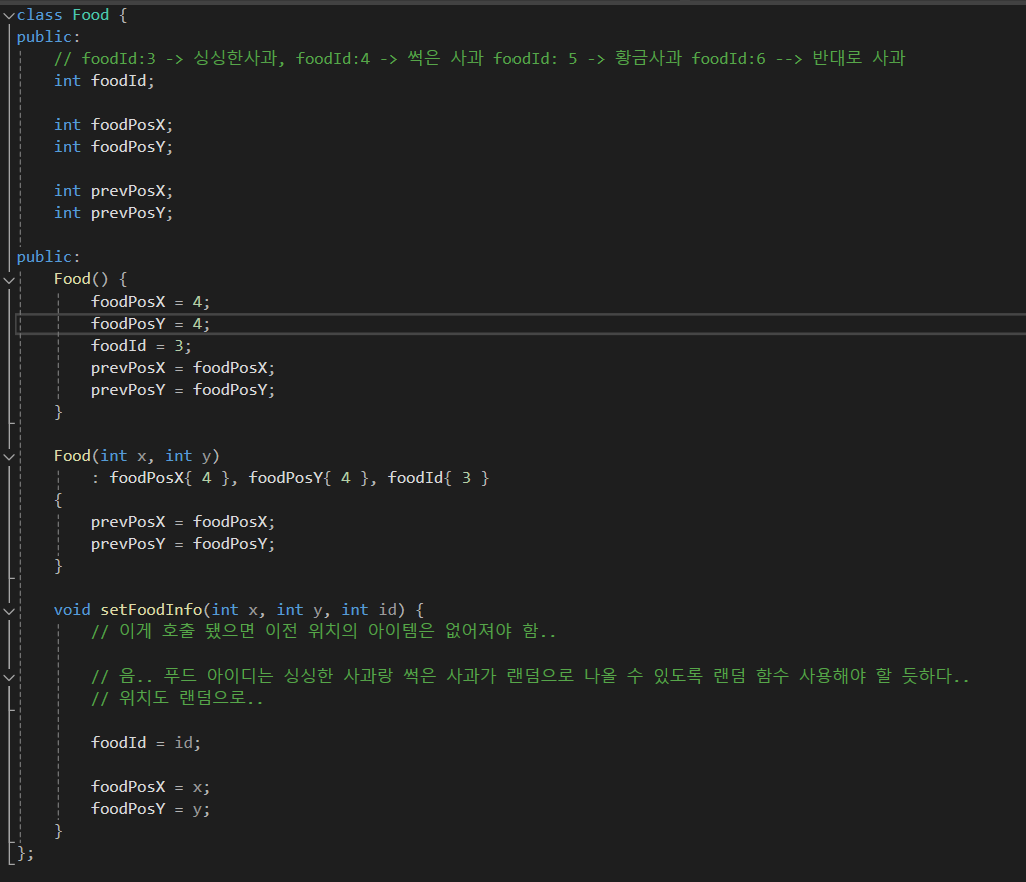
**개발자: 유채희**

**사용한 소스 파일:** Food.cpp, Snake.cpp, Map.h

**사용한 클래스 및 함수:** Food 클래스, Snake 클래스, Map 클래스, SnakeHead 클래스, SnakeBody 클래스, Food::setFoodInfo 함수, SnakeTouchFreshApple 함수, SnakeTouchPoisonApple 함수, Snake.removeTail 함수, Map::changeMapItemState 함수, snakeTouchGoldApple 함수, snakeTouchPoisonApple 함수

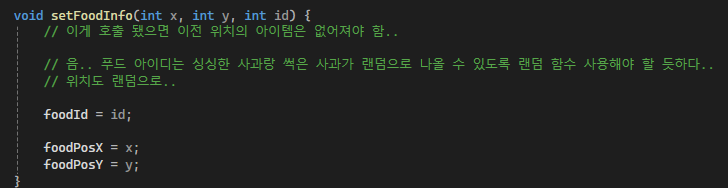
**Food 클래스:**

아이템 아이디, 아이템 위치, 아이템 이전 위치 변수를 가지는 클래스이다.



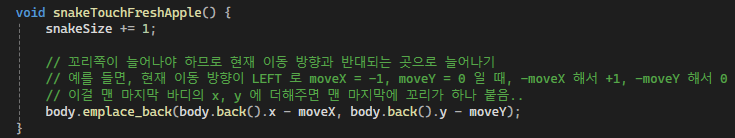
**Food::setFoodInfo 함수:**

Main.cpp 파일의 main() 함수에서 랜덤 아이템 위치와, 랜덤 아이템 아이디를 매개변수로 던져준다. 이 함수를 통해 아이템의 아이디와, 위치를 업데이트 해줄 수 있다.



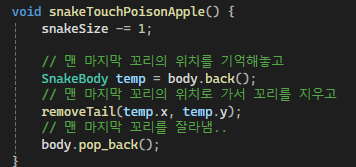
**Snake::SnakeTouchFreshApple 함수:**

뱀의 머리와 싱싱한 사과가 만날 때 호출되는 함수이다. 뱀의 사이즈를 1만큼 늘려준다. 꼬리쪽이 늘어나야 하므로 현재 이동 방향과 반대되는 곳으로 늘어나도록 해준다. 즉, 뱀의 이동량에 -를 곱한 값을 뱀의 꼬리의 위치 값에 더해주면, 꼬리가 늘어나게 된다.



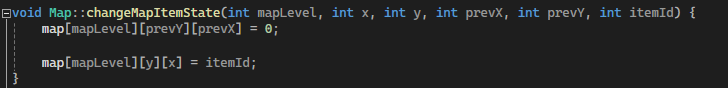
**Snake::SnakeTouchPoisonApple 함수:**

뱀의 머리와 썩은 사과가 만날 때 호출되는 함수이다. 뱀의 사이즈를 1만큼 줄여준다.



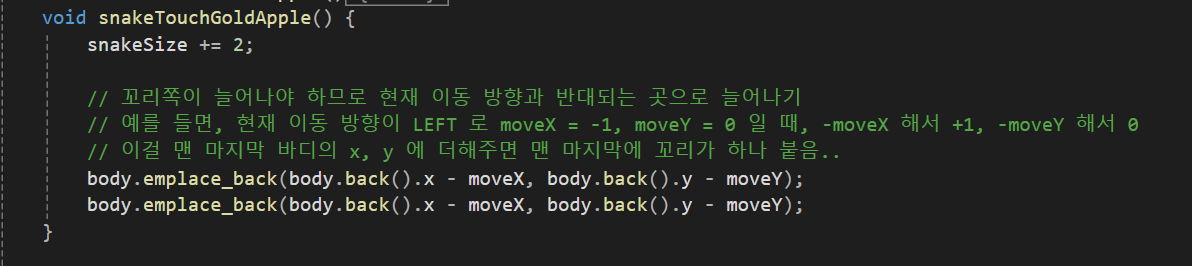
**Map::changeMapItemState 함수:**

아이템의 정보 업데이트가 끝난 후에 호출되는 함수이다. 이전 아이템의 위치를 받아서 맵의 요소를 0으로 바꿔주고, 현재 위치를 받아서 맵의 요소를 1로 바꿔준다.



**Snake::SnakeTouchGoldApple 함수:**

뱀의 머리와 황금 사과가 만날 때 호출되는 함수이다. 뱀의 사이즈를 2만큼 늘려준다.



**Snake::SnakeTouchChangeApple 함수:**

뱀의 머리와 거꾸로 사과가 만날 때 호출되는 함수이다. 뱀의 머리와 꼬리의 위치를 바꾸고, 진행 방향을 반대로 바꾼다.



**[4단계]**

**개발자: 공동개발(유채희, 박유나, 레일라)**

**사용한 소스 파일:** Gate.cpp, Gate.h, GateEmergeEnable.cpp, Snake.cpp, Map.cpp

**사용한 클래스 및 함수:**

클래스 - Gate 클래스, GateEmergeEnable 클래스, Snake 클래스(+SnakeHead 클래스, SnakeBody 클래스), Map 클래스.

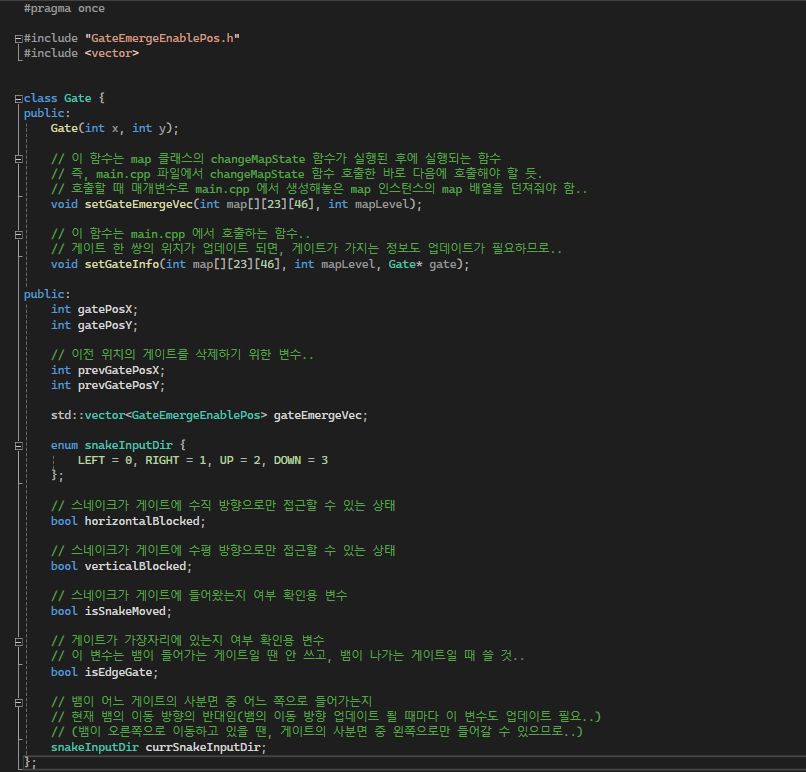
함수 - Snake::snakeTouchGate 함수, Snake::setPos 함수, Snake.teleportMove 함수, Snake::move함수(+SnakeHead::move, Snakebody::move 함수), Map::changeMapGateState 함수, Gate::setGateInfo 함수, Gate::setGateEmergeVec 함수, Map::draw(Map::setMap) 함수.

**Gate 클래스:**

스네이크 게임의 게이트 기능을 담담하는 클래스. 현재 게이트 위치, 이전 게이트 위치 변수를 이용하여 일정 시간이 지날 때마다 맵에 게이트 한 쌍이 생기고 없어지고를 반복하도록 했다.

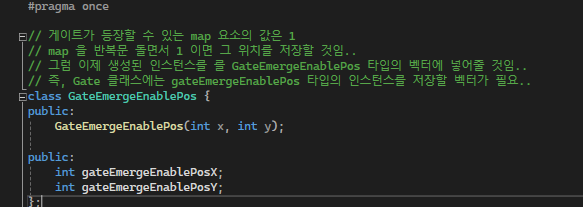
게이트가 등장할 수 있는 위치를 gateEmergeVec 이라는 타입의 인스턴스를 저장하는 벡터에 저장할 수 있게끔 하였다.

뱀이 게이트에 어떤 방향으로 접근할 수 있는지를 나타내는 상태 변수와, 뱀이 게이트를 이동하고 있는지의 여부를 나타내는 변수와, 게이트가 가장자리에 있는 게이트인지의 여부를 나타내는 변수와, snakeInputDir 타입의 currSnakeInputDir 변수를 통해 게이트와 뱀이 상호작용할 수 있도록 토대를 마련하였다.



**GateEmergeEnablePos 클래스:**

GateEmergeEnablePos 클래스는 게이트가 등장할 수 있는 위치를 저장하기 위한 클래스이다.



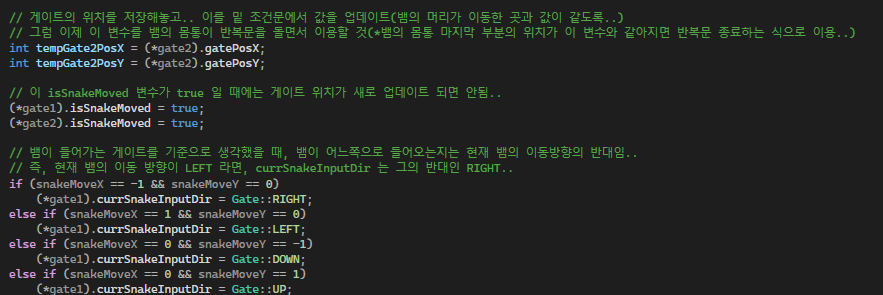
**Snake::snakeTouchGate 함수:**

너무 길어서 부분별로 설명할 것이다.

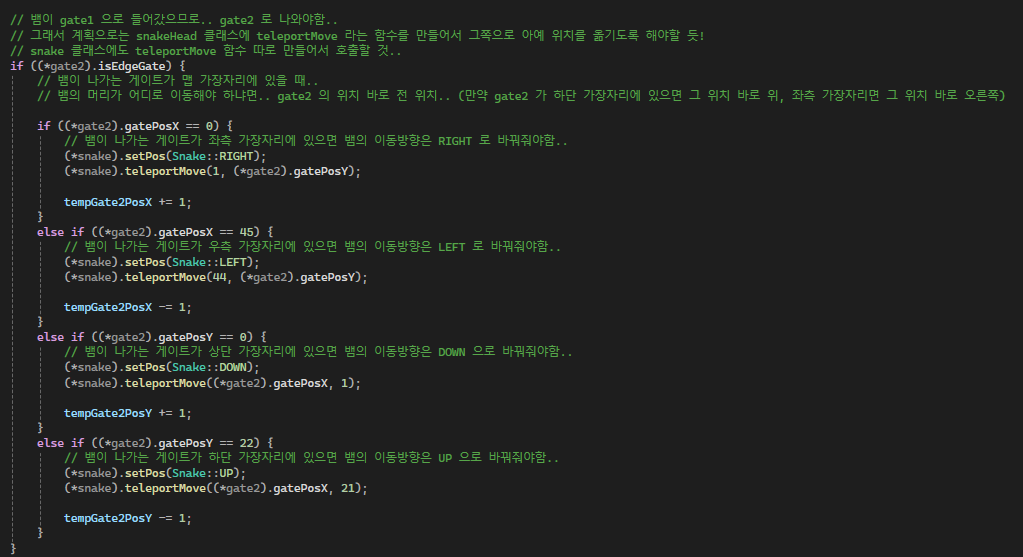
1. 함수 원형



2. 게이트의 위치를 저장해놓고, 게이트의 isSnakeMoved 변수의 값을 true 로 바꿔주어 main 함수에서 게이트 바꿀 시간이 되어도 안 바꾸도록 해주고, 뱀이 게이트의 어느 쪽에서 들어오는 지의 상태를 저장하는 변수 값을 뱀의 현재 이동량에 따라 다르게 업데이트 한다.



3. 뱀이 나가는 게이트가 맵 가장자리에 있는 경우를 처리하는 부분 로직이다. 게이트가 맵 가장자리에 있는지의 여부는 게이트의 isEdgeGate 변수를 통해 알 수 있다. 조건문에 진입하면 게이트의 위치에 따라서 뱀의 이동방향을 설정해주고, 뱀의 머리를 teleportMove 함수를 이용하여 게이트 직전 위치로 이동하도록 하였다.



4. 뱀이 나가는 게이트가 맵 내부에 있는 경우를 처리하는 부분 로직이다. IsEdgeGate 가 참인지 거짓인지를 판단하는 조건문에서 거짓이면 도달할 수 있는 부분이다. 이때, 비로소 뱀이 들어가는 게이트를 기준으로 설정한 currSnakeInputDir 변수를 사용한다.

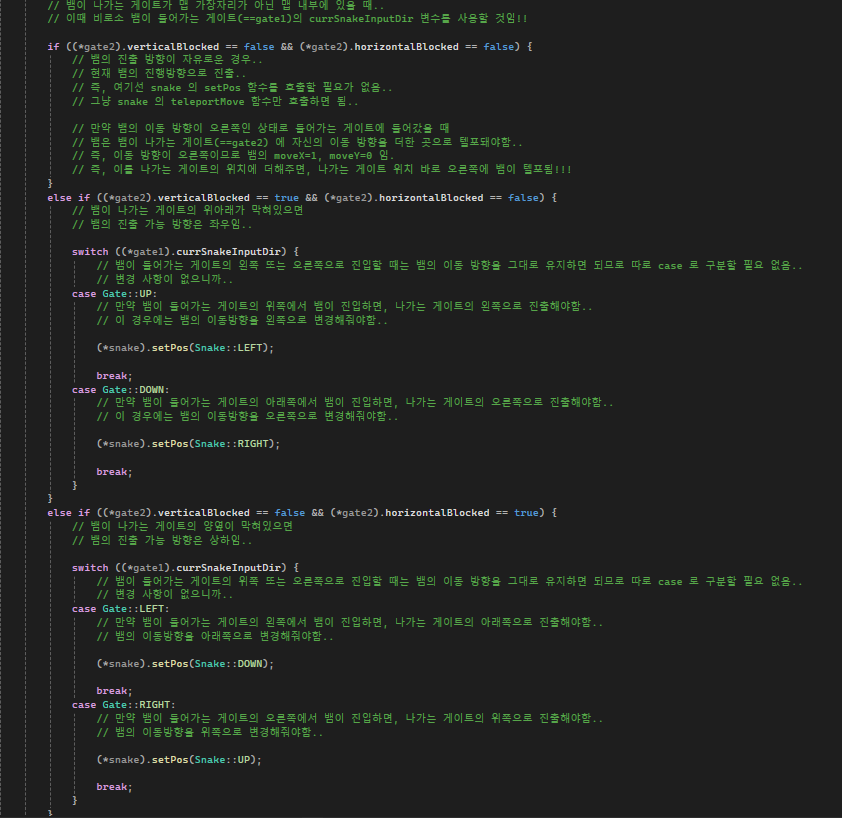
이때는 게이트의 세가지 상태에 따라 로직이 다른데 세가지 상태는 다음과 같다.

1. 게이트의 양옆과 위아래가 막혀 있지 않은 자유로운 상태
2. 게이트의 위아래가 막혀 있고 양 옆이 막혀 있지 않은 상태
3. 게이트의 양 옆이 막혀 있고 위아래가 막혀 있지 않은 상태

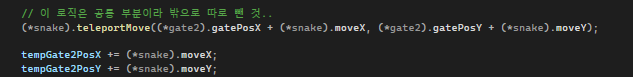
1. 자유로운 상태에서는 원래 뱀의 이동 방향을 유지한다.

2. 뱀이 나가는 게이트의 위아래가 막혀있으면 뱀의 진출 가능 방향은 좌우이다(currSnakeInputDir 의 방향에 따라 뱀의 이동방향을 설정해준다 -> 뱀이 나가는 게이트의 좌우로 진출 가능할 때 뱀이 들어가는 게이트로 오른쪽 또는 왼쪽 방향으로 들어왔다면, 그 방향을 그대로 유지하면 되기 때문에 switch 구문에서 case 로 다루지 않았다.)

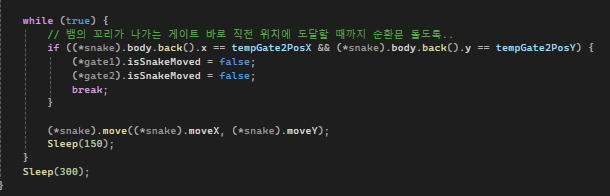
3. 뱀이 나가는 게이트의 양 옆이 막혀있으면 뱀의 진출 가능 방향은 위아래이다. 이 또한 currSnakeInputDir 의 방향에 따라 뱀의 이동 방향을 설정해준다.



5. 뱀의 이동 방향을 설정한 후에는 Snake 클래스의 teleportMove 라는 함수가 호출된다. TeleportMove 함수에는 SnakeHead 클래스의 teleportMove 함수가 사용되는데, 함수의 행동은 기존의 move 함수와 동일하다(SnakeHead 클래스의 move 가 사용되는 부분이 teleportMove 함수가 사용되도록 바뀐 것뿐이다).

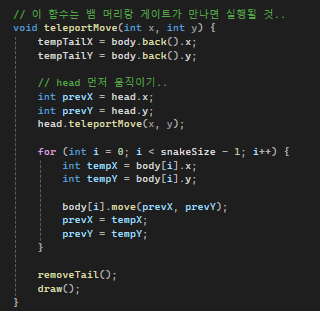


6. 뱀의 꼬리가 나가는 게이트 바로 직전 위치에 도달할 때까지 순환문을 돌도록 하여 뱀의 위치를 계속해서 이동시킨다. 도달하면 isSnakeMoved 라는 변수 값이 false 로 업데이트 한 다음 while 문을 종료한다. isSnakeMoved 변수가 false 가 되면 비로소, 일정 시간이 지나 게이트가 바뀔 시간이 되었을 때 무사히 게이트 변경 로직이 실행되게 된다.



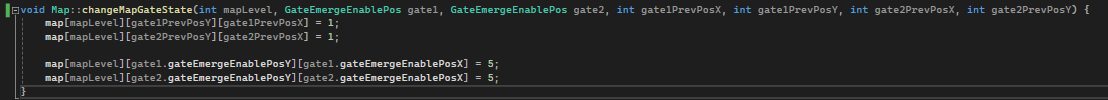
**Snake.teleportMove 함수:**

이 함수는 뱀의 머리 위치에 뱀의 이동량을 더했을 때의 위치가 게이트의 위치이면 호출되는 함수이다. 전체적인 함수의 로직은 Snake::move 함수와 같다. 단지 head 의 move 를 호출하던 부분이 teleportMove 를 호출하는 것으로 바뀌었다. 이 함수는 뱀의 머리의 위치를 먼저 바꾼 후, 머리부터 순차적으로 위치를 저장하는 임시 변수에 저장하고, 임시 변수의 값을 이용하여 뱀의 몸체의 위치를 바꾸는 역할을 한다. RemoveTail 함수로 커서가 뱀의 꼬리 부분에 가도록 해서 공백을 출력하고, 최종적으로 body 의 요소를 하나 빼서 꼬리를 없앤다. 마지막으로 draw 함수가 호출되어 변경된 뱀의 위치를 반영하여 콘솔에 출력한다.



**Map::changeMapGateState 함수:**

Main() 함수에서 랜덤으로 게이트가 등장할 수 있는 위치 값을 매개변수로 던져주면 이를 map 의 요소에 반영하는 역할을 한다. 이전 게이트의 위치로 map 의 요소에 접근하여 요소의 값을 1로 바꾼다(즉, 기존 게이트는 벽으로 바꾸고, 새로 받은 위치에 있는 벽에 게이트를 생성한다).

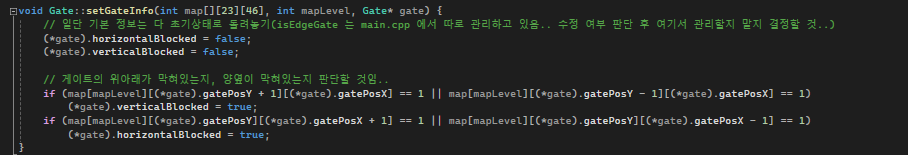


**Gate::setGateEmergeVec 함수:**

Main.cpp 파일의 main 함수에서 map 클래스의 changeMapGateState 함수가 호출된 다음으로 호출되는 함수이다. 함수로 진입하면 gateEmergeVec 벡터 변수에 들어 있던 인스턴스들을 clear() 를 이용하여 다 지워버린다(새로운 맵을 반영해야 하므로 기존의 게이트 생성 가능 위치 인스턴스를 다 버리는 것이다). 그 후 맵 사이즈만큼 이중 반복문을 돌면서 map 의 요소의 값이 1이면 gateEmergeVec.emplace\_back(j, i) 구문을 통해 새로운 인스턴스를 생성해 다시 벡터변수에 넣는다(게이트는 벽에만 생성될 수 있으므로 조건문에서 map 의 요소가 1인지 판단한 것이다).

**Gate::setGateInfo 함수:**

ChangeMapGateState 함수가 호출된 후 호출되는 함수이다. 게이트가 새로 업데이트 되었으므로 게이트가 가지는 정보도 업데이트 해줘야 하기 때문에 만든 함수이다. 이 함수에서는 게이트의 양옆이 막혀있는 지, 위아래가 막혀있는 지 판단해서 게이트의 변수(horizontalBlocked, verticalBlocked)를 업데이트 한다. 함수가 호출되자마자 변수 두개를 false 로 업데이트 해주는데, 일단 기본 정보를 다 초기 상태로 돌려놓기 위함이다. 조건문을 통해 게이트 y위치 위 또는 아래의 요소가 1이면 verticalBlocked 변수의 값을 true 로, 게이트 x위치 왼쪽 또는 오른쪽이 1이면 horizontalBlocked 변수의 값을 true 로 업데이트 해준다.



**Map::draw(Map::setMap) 함수:**

게이트와 관련된 모든 정보의 업데이트가 완료된 후 호출되는 함수이다. 이 함수는 위에서 설명하였으므로 넘어간다.

**[5단계]**

**개발자: 박유나**

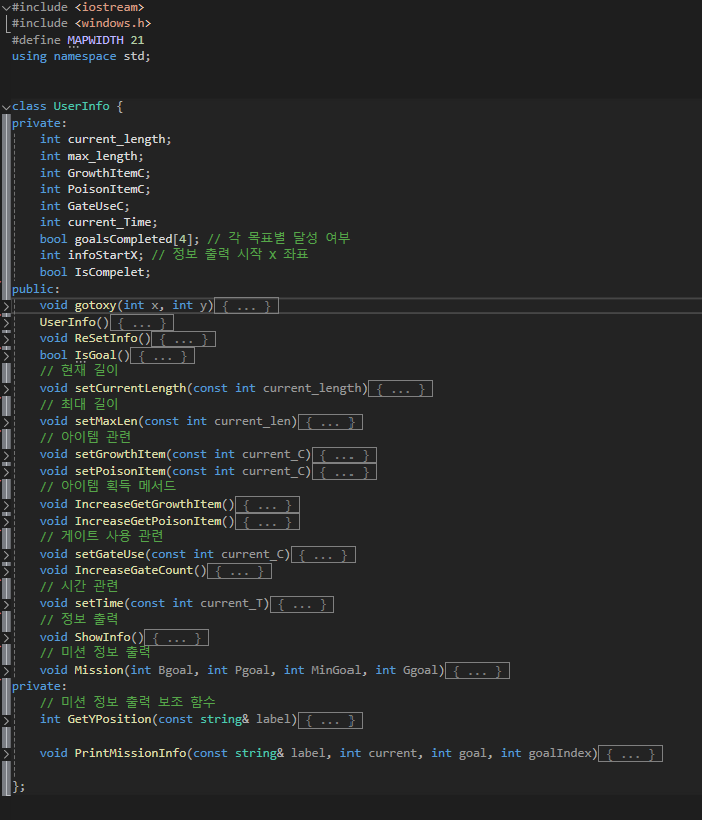
**사용한 소스 파일:** UserInfo.cpp, Snake.cpp

**사용한 클래스 및 함수:**

**UserInfo 클래스:** ReSetInfo(), IsGoal(), setCurrentLength(), setMaxLen(), setGrowthItem(), setPoisonItem(), IncreaseGetGrowthItem(), IncreaseGetPoisonItem(), setGateUse(), IncreaseGateCount(), setTime(), ShowInfo(), Mission(), GetYPosition(), PrintMissionInfo()

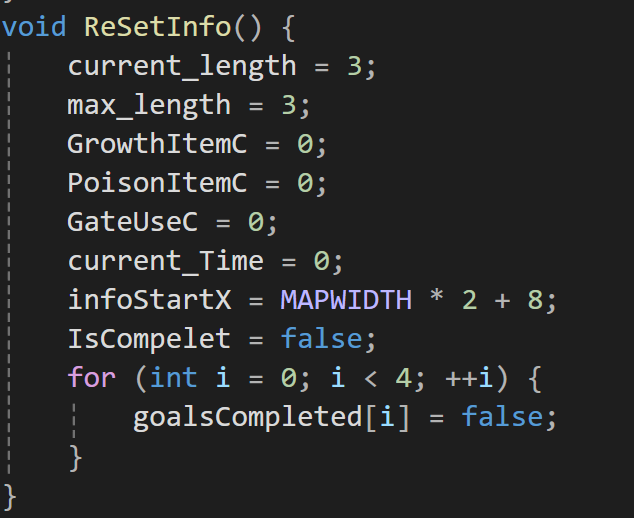
**Snake 클래스:** ReSetSnake()

사용자가 게임을 진행하면서 얻은 정보들을 담은 클래스. 현재 아이템을 먹은 횟수와 뱀의 길이, 게이트 이용 횟수의 변경을 함수로 관리하여 변경하고 출력하도록 하였다. 또한 미션 값을 달성하는 조건을 bool형 배열을 이용하여 판단할 수 있도록 기획하였다.

****

**UserInfo::ReSetSnake 함수:**

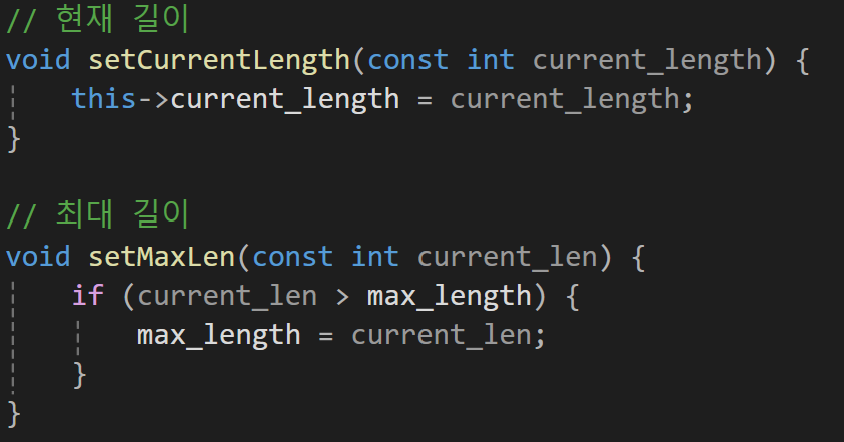
스테이지 클리어시 호출되는 함수. 현재 출력되고 있는 뱀의 길이, 뱀의 최대 길이, 아이템 획득 횟수, 게이트 이용 횟수, 각 미션별 달성 여부 등을 초기화한다.

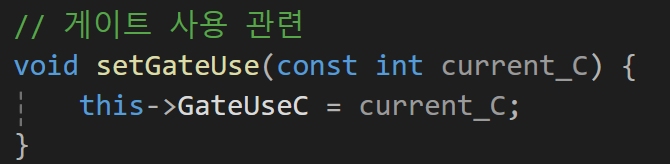
****

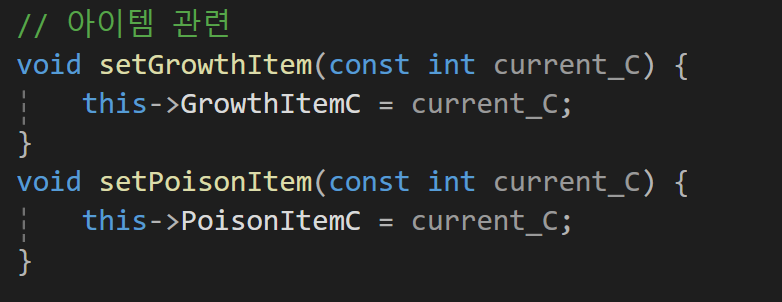
**UserInfo::setCurrentLength 함수, UserInfo::setMaxLength 함수, UserInfo::setGrowthItem 함수,**

**UserInfo::setPosionItem 함수, UserInfo::setGateUse 함수:**

아이템 획득 시 main 함수에서 호출되는 함수. 뱀의 길이와 최대 길이를 정하고, 각 아이템별 아이템 획득 개수를 정하며 게이트 사용 횟수를 정하는 함수들.

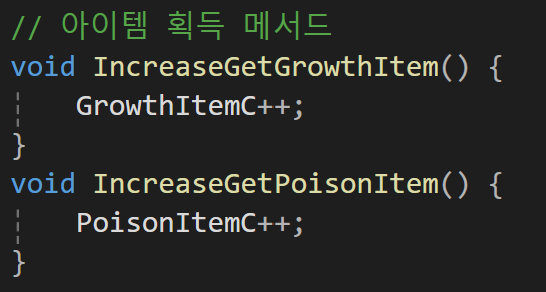
****

****

****

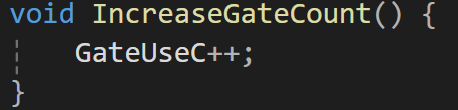
**UserInfo::IncreaseGeatGrowthItem 함수 , UserInfo::IncreaseGeatPosionItem 함수:**

아이템 획득 시 main 함수에서 호출되는 함수. 현재 출력되고 있는 각각의 아이템 획득 횟수를 증가시켜준다.

****

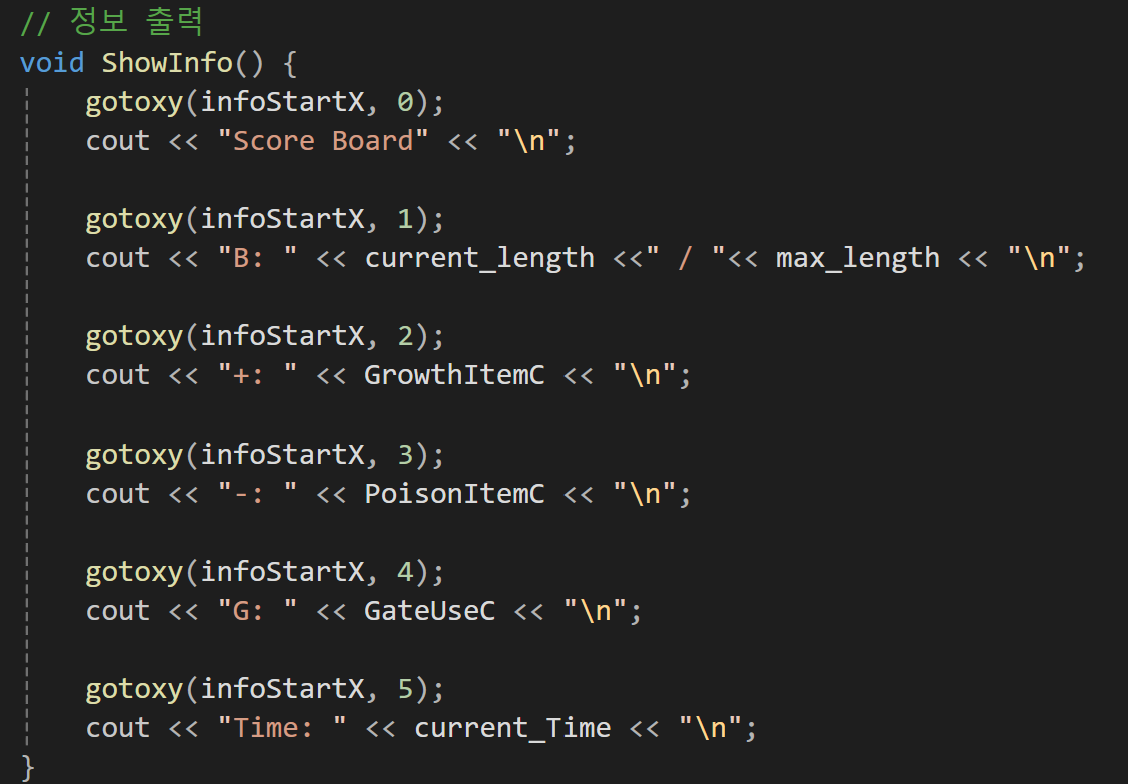
**UserInfo::IncreaseGateCount 함수 :**

게이트 사용시 main 함수에서 호출되는 함수. gate사용 횟수를 증가시킨다.

****

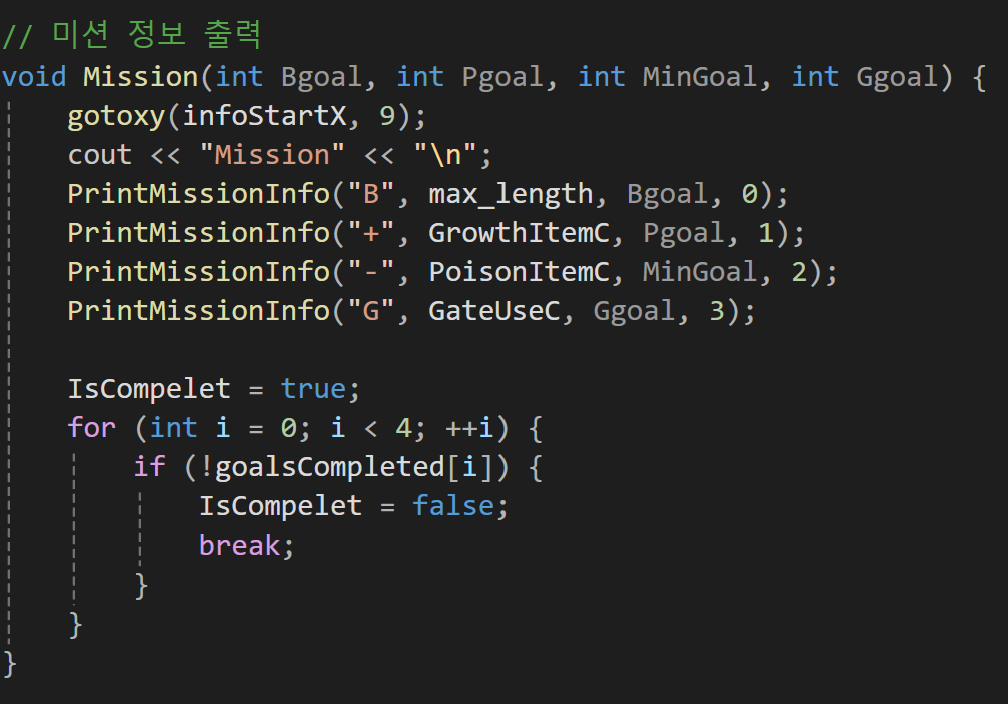
**UserInfo::ShowInfo 함수 :**

유저 정보를 출력하는 함수. 이때 infoStartX 변수는 출력하는 모든 문자들의 x좌표가 동일하기 때문에 변수로 관리함.

****

**UserInfo::Mission 함수 :**

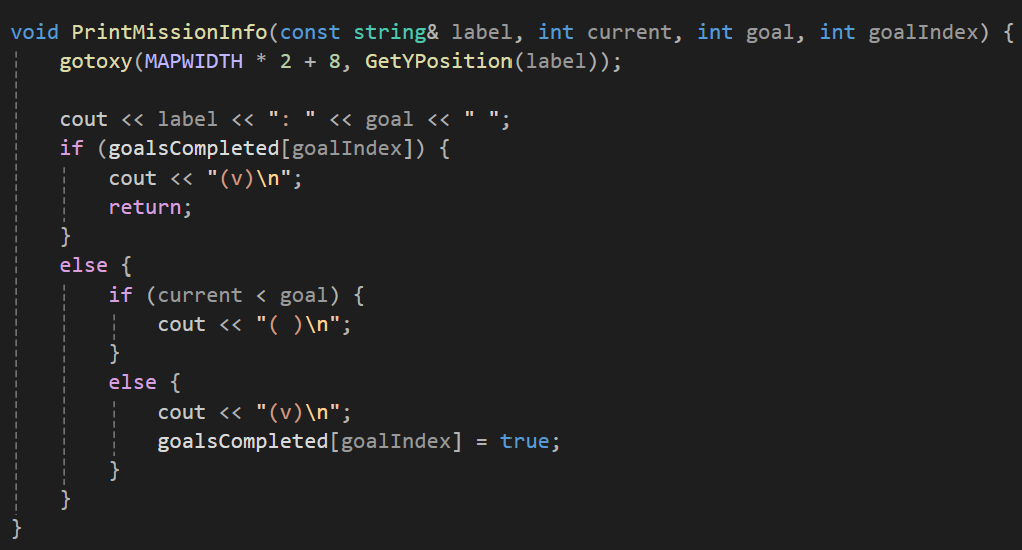
유저 정보를 출력하는 함수. 이때 infoStartX 변수는 출력하는 모든 문자들의 x좌표가 동일하기 때문에 변수로 관리함.

****

**UserInfo::PrintMission 함수 :**

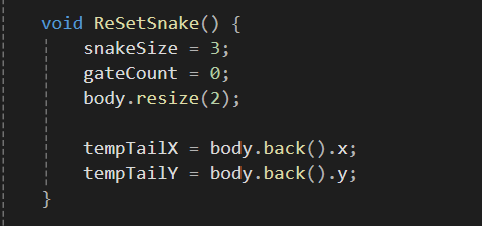
유저 정보를 출력하는 함수에서 호출하는 함수. current는 현재 값, goal은 목표값, goalIndex는 현재 목표를 식별하는 인덱스를 의미함. 만약 목표치 미만이면 ( )를 출력. 목표치 이상이면(v)를 출력하고 bool 형 배열 goalsCompleted의 해당 인덱스를 true로 바꿈.

만약 goalsCompleted 배열의 해당 인덱스 값이 true라면 (v)를 출력하고 return을 하여 이후에 있는 조건문에 들어가지 않도록 함.

****

**Snake::ReSetSnake 함수:**

스테이지 클리어시 뱀의 정보를 초기화 한다. 길이를 3으로 변경하고 몸의 길이를 2로 변경. 게이트 사용 횟수를 초기화 한다.

****

**[6 단계]**

**개발자: 레일라**

**사용한 소스 파일:** UserInfo.cpp, Snake.cpp, Map.cpp

**사용한 클래스 및 함수:**

**Snake 클래스:** setColor(),draw()

**Map 클래스:** setColor(), Map::draw()

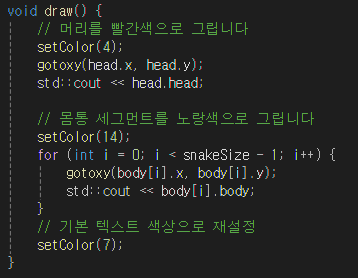
**UserInfo 클래스:** setColor(), ShowInfo(), Mission(), PrintMissionInfo()

**Snake.cpp:**

이 함수는 콘솔 출력의 텍스트 색상을 설정한다.

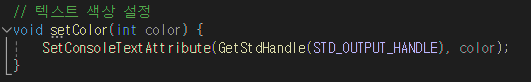
****

머리 그리기: 콘솔 텍스트 색상을 녹색(setColor(4))으로 설정한 다음 뱀의 머리를 나타내는 문자를 인쇄한다. 본문 그리기: 콘솔 텍스트 색상을 빨간색(setColor(14))으로 설정하고 본문 세그먼트를 나타내는 문자를 인쇄한다. 텍스트 색상 재설정: 모든 세그먼트를 그린 후 콘솔 텍스트 색상을 기본값(setColor(7))으로 재설정한다. 이는 일반적으로 후속 텍스트 출력이 기본 색 구성표를 따르도록 하기 위해 수행된다.

****

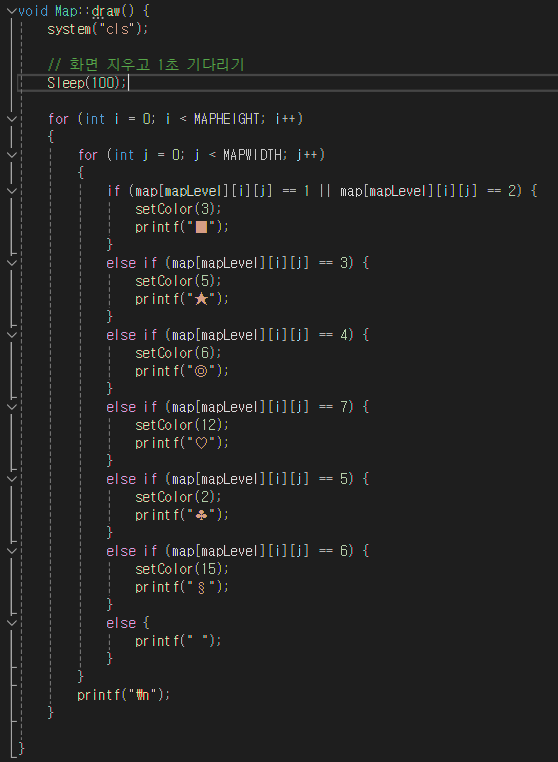
**Map.cpp:**

이 함수는 콘솔 출력의 텍스트 색상을 설정한다.

****

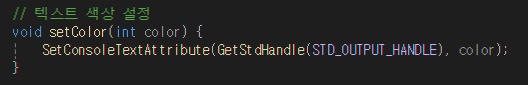
값에 따라 다른 문자(■ - 벽, ★- 음식(+1), ◎ - 몸체 삭제, ♡ - 문, ♣ - 음식(+2), § - 다른 쪽으로 변경)이 지도에 인쇄된다. 각 문자의 색상은 각 문자를 인쇄하기 전에 다양한 색상 코드(3 - 파란색, 5 - 보라색, 6 - 진한 노란색, 12 - 빨간색, 2 - 녹색, 15 - 회색)를 사용하여 setColor() 함수 호출을 사용하여 설정된다.

이 draw() 함수는 콘솔 화면에 게임 맵을 시각적으로 표현하는 역할을 합니다. 콘솔 색상을 사용하여 숫자 값을 기반으로 지도의 다양한 요소를 표시한다.

****

**UserInfo.cpp:**

이 함수는 콘솔 출력의 텍스트 색상을 설정한다.

****

이 코드는 텍스트 기반 콘솔 화면에 다양한 정보를 표시하는 'ShowInfo()' 함수를 정의한다.

이제 `ShowInfo()` 함수 내용을 설명하겠다.

1. 색상을 녹색(색상 코드 10)으로 설정한다. "Score Board"와 개행 문자를 인쇄한다.

2. 색상을 녹색으로 설정한다. "B:"를 녹색으로 인쇄한 다음 `current\_length` 값 뒤에 " / " 및 `max\_length`를 보라색으로 인쇄한다.

3. 색상을 녹색으로 설정한다. "+:"를 녹색으로 인쇄한 다음 'GrowthItemC' 값을 보라색으로 인쇄한다.

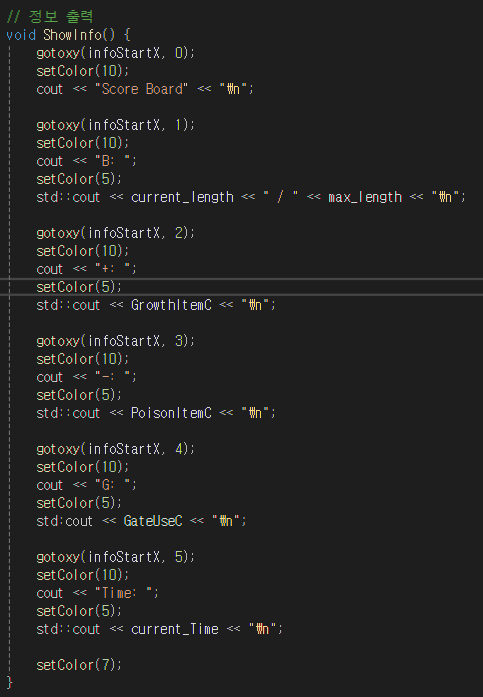
4. 색상을 녹색으로 설정한다. "-:"를 녹색으로 인쇄한 다음 `PoisonItemC` 값을 보라색으로 인쇄한다.

5. 색상을 녹색으로 설정한다. "G:"를 녹색으로 인쇄한 다음 'GateUseC' 값을 보라색으로 인쇄한다.

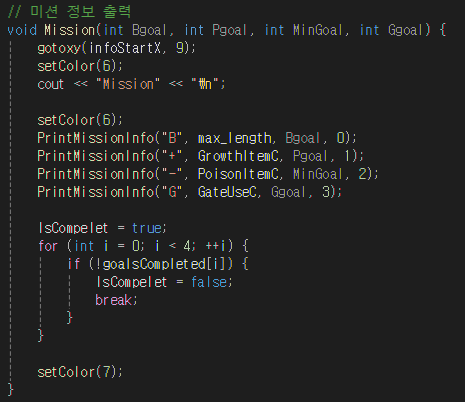
6. 색상을 녹색으로 설정한다. "Time:"을 녹색으로 인쇄한 다음 `current\_Time` 값을 보라색으로 인쇄한다.

7. 색상을 기본값(색상 코드 7)으로 다시 설정합니다. 그러면 모든 정보가 표시된 후 원래 텍스트 색상이 복원될 수 있다.

'ShowInfo()' 함수는 점수, 아이템 개수, 사용 횟수, 현재 시간 등 다양한 게임 관련 정보를 업데이트하여 콘솔 화면에 표시하는 기능이다. 텍스트 기반 환경에서 정보가 명확하고 깔끔하게 표시되도록 색상 설정 기능을 사용한다.

****

void Mission(int Bgoal, int Pgoal, int MinGoal, int Ggoal)은 Bgoal, Pgoal, MinGoal 및 Ggoal의 네 가지 정수 매개변수를 사용하는 Mission이라는 함수를 정의한다. 텍스트 색상을 진한 노란색으로 변경하는 setColor(6)에 의해 정의된 이 '미션' 텍스트이다. 이 'B', ''+', '-', 'G'는 텍스트 색상을 어두운 노란색으로 변경하는 setColor(6)에 의해 정의된다. setColor(7); 색상을 생각합니다(색상 코드 7)으로 다시 설정한다. 모든 정보가 후에 원래대로 색상이 복원될 수 있다.

****

이 **'PrintMissionInfo'** 기능은 콘솔 기반 애플리케이션에서 미션이나 작업에 대한 정보를 표시하도록 설계되었다.

1. 텍스트 색상 설정 및 라벨 인쇄:

- **setColor(6)** 텍스트 색상을 진한 노란색으로 변경한다.

- 라벨과 목표값이 노란색으로 인쇄되어 미션의 맥락을 제공한다.

2. 완료 상태 표시:

- 미션이 완료되었는지 확인하는 부분이다.

- `goalsCompleted[goalIndex]`가 `true`인 경우 텍스트 색상을 녹색 **setColor(10)**으로 설정하고 미션 완료를 나타내는 체크 표시 "(v)"를 인쇄한니다.

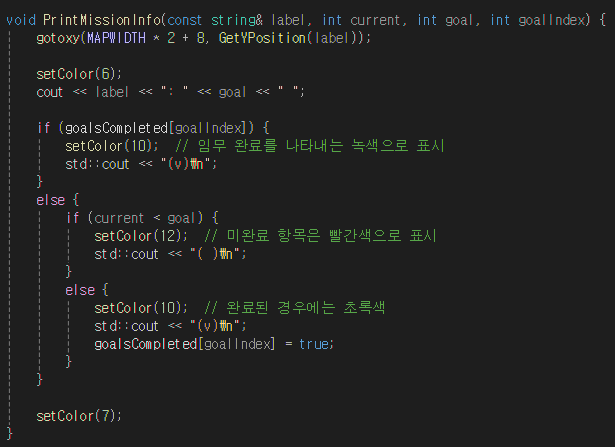
- 임무가 아직 완료로 표시되지 않은 경우(`goalsCompleted[goalIndex]`는 `false`입니다):

\* '현재'가 '목표'보다 작을 경우 미션이 아직 완료되지 않은 것이다. 이 함수는 텍스트 색상을 빨간색 **setColor(12)**로 설정하고 빈 체크박스 "( )"를 인쇄한다.

\* '현재'가 '목표'보다 크거나 같으면 이제 미션이 완료된다. 이 함수는 텍스트 색상을 녹색 **setColor(10)**로 설정하고 체크 표시 "(v)"를 인쇄하며 `goalsCompleted` 배열에 임무가 완료된 것으로 표시한다.

3. 텍스트 색상 재설정:

- **setColor(7)**는 텍스트 색상을 기본값(많은 콘솔 응용 프로그램에서 일반적으로 회색 또는 흰색)으로 재설정하여 후속 텍스트가 기본 색상으로 표시되도록 한다.

****

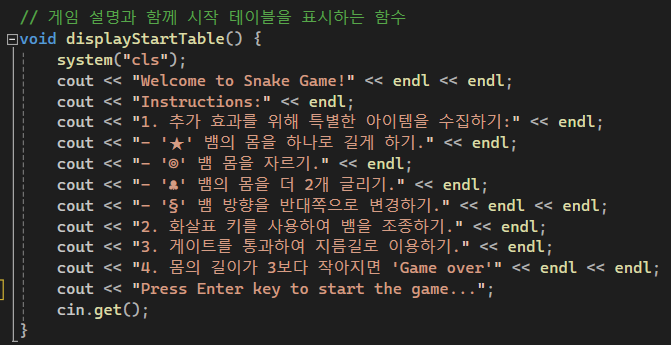
**[7 단계]**

**개발자: 레일라**

**사용한 소스 파일:** main.cpp

**사용한 클래스 및 함수:** displayStartTable()

코드는 스네이크 게임의 시작 화면이나 소개 테이블을 표시하는 역할을 한다.



### 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.** |

**Iostream:** 콘솔 창에 게임의 요소를 출력하기 위해 사용되었다. Cout 을 이용하여 이동된 커서의 위치에 요소를 출력하고, 반복문을 돌면서 맵을 출력하는 등 다양한 게임의 요소를 출력하는데 필수적이고 기본이 되는 역할을 하기 때문에 사용되었다.

**Windows.h :** 콘솔 창에서 커서가 위치를 바꾸면서 게임의 요소(뱀, 게임 정보 등)를 출력할 수 있도록 하기위해 사용되었다. 뱀의 위치에 따라 반복문을 돌면서 커서를 뱀의 위치로 움직여 뱀을 출력하는 기능을 수행하기 위해 사용되었다. 입력받은 방향키에 따라서 뱀의 이동 방향을 제어하기 위해 사용되었다.

**cTime :** 맵에 아이템 또는 게이트가 생성되는데 딜레이를 걸어주기 위해 사용되었다. 아이템과 게이트는 일정한 시간이 지나면 상태를 업데이트 하는 로직이 수행되어야 하는데 이를 위해서는 조건문을 걸어주어야 했다. 즉, 시간 변수를 선언하여 조건문에서 사용하기 위해 사용된 것이다(그냥 Sleep 함수를 이용하면 조건문에 따른 분기를 나눌 수 가 없어서 사용했다).

**vector :** 뱀의 몸통 인스턴스를 저장하고, 게이트가 생성될 수 있는 위치의 값을 가지고 있는 클래스의 인스턴스를 저장하기 위해 사용되었다. 라이브러리에서 제공하는 back, emplace\_back, pop\_back,clear 등의 함수를 이용하여 뱀의 꼬리가 잘리고, 뱀의 꼬리가 늘어나고 , 생성자를 호출하여 인스턴스를 생성함과 동시에 벡터에 넣고, 게이트가 생성될 수 있는 위치의 값을 저장한 벡터를 초기화 하는 등의 기능을 구현하였다.

**random :** random\_device, mt19937, unifom\_int\_distribution 을 사용하여 게임에서 필요로 하는 랜덤값을(게이트의 새로운 위치, 아이템의 새로운 위치, 아이템의 새로운 아이디, 게이트가 생성될 수 있는 값을 저장한 벡터에서 요소를 가져오기위한 인덱스 등) 생성하는 로직을 구현하기 위해 사용되었다.

### 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.** |

**1단계:**

맵을 출력하는 과정에서 y 축으로는 콘솔상에서 공백이 있는 것처럼 보이는데, x 축으로는 공백이 없이 딱 붙어있어서 맵의 모양새를 잡는 데에 처음에 시간을 허비했다. X 축에서 맵의 요소를 임의로 출력할 때마다 공백을 추가로 출력하게 했는데, 모양새는 그럴싸해도 2단계의 뱀과 맵이 상호작용하도록 하는 로직을 작성할 때 문제가 생길 것이 예상되었다. 결국 x 축과 y 축의 일관성을 포기하고, 맵의 가로 길이를 늘려서 모양새를 어느정도 갖추도록 했다.

**2단계:**

뱀의 움직임을 while 반복문에서 계속해서 move 함수를 호출하여 관리하였다. 함수를 호출하는 시간에 텀을 주기 위해서 Sleep 함수를 이용하였는데 값을 크게 줘버리면 방향키의 입력이 제대로 먹히지 않아서 값을 작게 넣어주었다(뱀은 0.15 초만큼의 텀을 줘서 움직이도록 했다).

**3단계:**

처음에는 아이템이 생성될 수 있는 곳을 맵 내부의 위치로 설정했기 때문에 랜덤 값의 범위 또한 맵 내부로 설정 해놓았다. 그러나 이렇게 하면 벽이 맵의 내부에 존재할 때, 벽 위치에 아이템이 생성되는 경우를 발견하였다. 즉, 어쩔 수 없이 랜덤 범위는 그대로 유지하고 만약 새로운 랜덤값으로 맵의 요소를 접근했을 때, 그 요소의 값이 1이면 1이 아닐 때까지 반복문을 돌도록 하는 로직을 작성하였다.

**4단계:**

뱀의 머리의 위치에 뱀의 이동량을 더한 위치가 게이트의 위치와 같으면 뱀의 snakeTouchGate 함수를 실행하여 게이트와의 상호작용을 수행하려고 했다. 계획과는 다르게 조건문 안으로 진입하지 않아서 뱀의 머리 위치와 게이트의 위치를 하나하나 출력하면서 디버그를 수행했다. 결국 main 함수의 while 반복문 내에서 뱀의 머리 위치+이동량 의 위치와 게이트의 위치를 비교하는 조건문의 위치를 옮겼더니 정상적으로 논리가 수행되었다.

**5단계:**

맵 출력 로직이 주기적으로 맵 상태를 지웠다가 다시 출력하는 형식으로 업데이트하는 형태로 맵과 함께 정보값을 출력을 할 경우 출력값이 지워지게 되는 문제가 있었다. 이를 맵 생성함수와, 정보값을 출력함수를 각각 따로 main함수에서 호출하여 맵의 상태가 변해도 정보값에는 영향이 가지 않도록 했다. 이로써 사용자의 정보값이 게임 도중 지워지지 않고 계속해서 현재 화면에 출력되게 했다.

**6 단계:**

게임에 색상을 추가할 때, 색상이 부여된 속성이 변경되는 것뿐만 아니라, 색상이 부여되지 않은 다른 속성이나 다른 색상이 부여되는 경우 심각한 문제가 발생했습니다. 코드에서 사용 가능한 모든 속성에 특정 색상을 추가하여 문제가 해결되었습니다.

**7 단계:**

7 단계를 개발할 때 아무런 문제가 없었습니다.

### 결과물 목록

|  |
| --- |
| **작성요령 (5점)**  **결과물 목록을 작성한다. 목록은 제출하는 파일과 각 파일의 역할을 간략히 설명한다.** |

다음은 이 프로젝트에서 사용된 파일들과 각 파일의 역할에 대한 간략한 설명이다:

1. **main.cpp**
   * 게임의 메인 루프와 주요 로직이 구현된 파일이다.
   * 게임의 시작 테이블을 표시하고 초기 맵 상태를 보여준다.
   * 뱀의 움직임, 아이템 생성, 게이트 등장 등의 게임 로직을 담고 있다.
2. **Map.h**
   * 맵 관련 클래스 및 함수 선언이 포함된 헤더 파일이다.
   * 게임 맵을 설정하고 그리는 데 사용된다.
3. **Gate.h**
   * 게이트 클래스의 선언이 포함된 헤더 파일이다.
   * 게이트의 위치 설정 및 관리를 담당한다.
4. **Snake.cpp**
   * 뱀 클래스의 구현이 포함된 소스 파일이다.
   * 뱀의 움직임 및 상호작용을 정의한다.
5. **Food.cpp**
   * 음식 클래스의 구현이 포함된 소스 파일이다.
   * 다양한 종류의 음식을 생성하고 그 위치를 관리한다.
6. **UserInfo.cpp**
   * 사용자 정보 클래스의 구현이 포함된 소스 파일이다.
   * 사용자의 게임 상태 및 진행 정보를 추적하고 표시한다.
7. **Gate.cpp**
   * 게이트 틀래스의 수현이 포함된 파일이다.
   * 게이트 위치 설정 및 관리 기능을 구현한다.
   * 게이트의 등의 위치를 랜덤으로 생성하고, 게이트의 상호작용 로직을 담당한다.
8. **Map.cpp**
   * 맵 틀래스의 구현이 포함된 소스 파일이다.
   * 게임 맵의 구조를 성정하고 초기화하는 기능을 담당한다.
   * 맵의 레벨 업데이트 및 맵의 상태를 그리는 기능을 구한다.

# 자기평가

|  |
| --- |
| **작성요령 (5점)**  **프로젝트를 수행한 자기 평가를 서술한다. 팀원 개개인의 자기 평가가 포함되어야 하며, 본인의 역할, 프로젝트 수행 시 어려운 점, 도움이 되었던 점, 이 프로젝트 운영에 개선이 필요하다고 생각하는 점을 충분히 서술한다.** |

**유채희:**

스네이크 게임 프로젝트에서 필요로 하는 클래스와, 클래스 간의 관계, 각 클래스가 필요로 하는 변수와 함수 등 전반적인 프로젝트의 설계를 작성하여 게임의 큰 틀을 잡았다. 프로젝트의 진행이 원활히 될 수 있도록 최대한 빨리 스네이크 게임의 기본 토대를 만들었다(맵 생성 및 뱀의 움직임, 뱀과 아이템의 상호작용 등).

직접 구성한 설계 내용을 팀원들에게 설명해주어 현재 프로젝트의 진행상황과 프로젝트에서 필요로 하는 클래스들의 내용, 서로의 관계를 팀원들이 파악할 수 있도록 하였고, 이를 토대로 팀원들이 개발을 진행할 수 있도록 하였다.

프로젝트 수행 시 어려웠던 점은 팀원들에게 역할을 분담시키는 것이었다. 조장으로써 팀원 개인에게 해야 할 내역을 정해줘서 개발을 진행해야 했는데, 세 명이서 동시에 똑같은 작업을 하느라 개발의 진도가 느리고, 한 사람이 성공하면 다른 팀원들의 개발 상황이 무의미해지는 감이 있어서 조절하기가 어려웠다. 다음 팀 프로젝트에서는 어떤 식으로 프로젝트를 진행해야 할지에 대한 반면교사가 되었다.

이 프로젝트를 진행하면서 도움이 되었던 점은 클래스 간의 관계를 설계하는 행위에 감을 잡을 수 있었던 것과, 클래스의 사용법을 단기간에 많이 배울 수 있었던 것이다. 그리고 헤더 파일과 소스 파일을 분리하는 행위를 이번 기회에 많이 연습할 수 있었다(분리를 안하면 오류가 생기는 경우가 많아서 저절로 연습을 할 수밖에 없었다).

**박유나:**

스네이크 게임의 사용자 정보와 미션값 출력, 각 단계별 추가 사항들을 구현하였다. 또한 스네이크 게임 구현시 생긴 주요 오류를 직접 디버깅하며 해결하였다.

협업의 장점은 혼자 했을 때는 하지 못했던 것을 할 수 있는 것인 것 같다. 조장이 먼저 로직을 잘 짜둔 덕분에 이를 차분히 이해하면서 나의 코드를 짤 수 있었다. 만일 조장이 먼저 로직을 짜지 않았더라면 갈피를 제대로 잡지 못했을 것 같다. 또한 이 프로젝트를 수행하면서 c++에 대한 깊은 공부가 되었다. 클래스의 사용법을 제대로 익힐 수 있었던 것 같다.

아쉬운점은 프로젝트를 시작할 때 역할분담을 따로 하지 않아서 생기는 문제들이었다. 이미 다른 조원이 코드를 완성해 내가 짜고 있던 코드가 한발 늦어 쓰이지 못했던 것이 아쉬웠던 것 같다. 또한 협업을 위해 깃허브를 사용했다. 아직 모두 사용법이 미숙하여 버전관리가 제대로 되지 않았던 것이 아쉬웠으나, 이번 기회를 통해 협업을 할 때 버전관리를 하는 법을 배우게 되어서 결과적으로는 좋았던 것 같다.

**레일라:**

스네이크 게임의 중요한 요소 중 하나인 게임 색상을 디자인을 했고 게임의 재미를 더하는 게이트 개발도 수행하였다. 플레이어들이 게임을 더 잘 이해할 수 있도록 게임 지침도 만들었다. 무엇보다 마지막 단계에서 제일 중요한 코드의 최종적인 개선과 구조화를 담당했다.

프로젝트에서 큰 틀을 짜면 그 방향에 맞게 맡은 역할을 해내야한다. 특히 마지막 마무리 단계에서 코드를 다듬고 구조화시킬 때 프로젝트의 방향에서 벗어나지 않도록 일관성을 유지하는 것이 쉽지 않았다. 만약 처음에 계획을 제대로 세우지 않았다면 마무리를 잘 하는 것도 어려웠을 것이다.

또한 처음에 역할 분담이 명확하게 되지 않아 효율적으로 프로젝트를 진행하는 것이 어려웠었다. 처음부터 팀원마다 정확하게 업무를 나누고 프로젝트를 진행했으면 더 좋았을 것 같다.

플레이어가 피로를 느끼지 않도록 색상을 조심히 선택하는 디자인은 어려웠지만 재미있었다. 게임에 색상을 추가할 때, 색상이 부여된 속성이 변경되는 것뿐만 아니라, 색상이 부여되지 않은 다른 속성이나 다른 색상이 부여되는 경우 심각한 문제가 발생했다. 코드에서 사용 가능한 모든 속성에 특정 색상을 추가하여 문제가 해결되었다.

또한 게임의 재미를 더 하기 위해 게이트 개발도 어려웠지만 흥미로웠다. 기존의 시스템에 통합이 잘 되면서 서로 코드가 충돌하지 않도록 해야 했다. 테스트를 반복하면서 조정해야했지만 이것이 나중에 해야할 개발자의 일이라는 생각도 들면서 뿌듯하기도 했다.

이번 프로젝트에는 아쉬운 점도 많았지만 그만큼 더 배우면서 성장할 수 있었던 좋은 기회였다고 생각한다. 다음에 프로젝트를 하게 된다면 더 잘 할 수 있을 것이라고 생각한다..

# 참고 문헌

**참고한 서적, 기사, 기술 문서, 웹페이지를 나열한다.:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **번호** | **종류** | **제목** | **출처** | **발행년도** | **저자** | **기타** |
| 1 | 웹 | [C/C++ 콘솔 강좌] 커서 이동하기 (gotoxy 함수) | [[C/C++ 콘솔 강좌] 커서 이동하기 (gotoxy 함수) : 네이버 블로그 (naver.com)](https://blog.naver.com/PostView.nhn?blogId=noksek0615&logNo=221555500856) | 2019.6.6 | 공부용 녹색이 |  |
| 2 | 웹 | 씹어먹는 C++ - <17 - 3. 난수 생성(<random>)과 시간 관련 라이브러리(<chrono>) 소개> | [씹어먹는 C++ - <17 - 3. 난수 생성(<random>)과 시간 관련 라이브러리(<chrono>) 소개> (modoocode.com)](https://modoocode.com/304#google_vignette) | 2020.1.6 | 이재범 |  |
| 3 | 웹 | C언어로 게임 만들기(1) - (콘솔창 세팅, window.h, conio.h) | [C언어로 게임 만들기(1) - (콘솔창 세팅, window.h, conio.h) (tistory.com)](https://jubrodev.tistory.com/1) | 2023.5.13 | 주브 |  |
| 4 | 웹 | API 탐방 2편. 키의 상태를 확인하는 GetKeyState, GetAsyncKeyState | [API 탐방 2편. 키의 상태를 확인하는 GetKeyState, GetAsyncKeyState (hexabrain.net)](https://blog.hexabrain.net/156) | 2012.10.20 | 끝나지 않는 프로그래밍 일기 |  |

# 부록

|  |
| --- |
| **작성요령 (15점)**  **프로젝트의 결과물을 사용하기 위한 방법에 대해서 작성하세요.** |

## 사용자 매뉴얼

프로젝트를 실행하면 게임 시작화면이 표시됩니다. 시작화면에는 게임의 설명과 함께 초기 맵 상태가 표시됩니다. 사용자는 엔터키를 눌러 게임을 시작할 수 있습니다.

게임 진행:

- 게임이 시작되면 뱀을 제어하기 위해 방향키를 사용합니다.

- 뱀은 벽이나 자신의 몸에 부딪히지 않도록 조심해야 합니다.

- 다양한 종류의 푸드를 먹어 뱀의 길이를 늘리거나 줄이고, 특수 아이템 효과를 얻을 수 있습니다.

- 게이트를 통과하여 단축 경로를 이용할 수 있습니다.

- 미션을 완료하여 스테이지를 진행합니다.

뱀의 크기가 3보다 작아지면 게임이 종료됩니다. 모든 스테이지의 미션을 완료하면 게임 클리어 메시지가 표시됩니다.

## 설치 방법

**[방법1. 프로그램 컴파일 및 실행 방법]:**

**1. 환경 요구사항:**

- Windows 운영체제

- C++ 컴파일러 (예: Visual Studio, MinGW 등)

**2. 컴파일 및 빌드:**

- 소스 코드 파일들을 컴파일러로 컴파일하여 실행 파일을 생성합니다.

- 예를 들어, Visual Studio에서는 솔루션 파일을 열고 빌드하여 실행 파일을 생성할 수 있습니다.

**3. 실행:**

- 생성된 실행 파일을 더블 클릭하여 게임을 실행합니다.

- 게임 실행 후에는 시작화면과 게임 설명이 표시됩니다. 사용자는 시작화면에서 아무 키나 눌러 게임을 시작할 수 있습니다.

**[방법2. 다운받은 실행파일 실행하기]:**

1. **환경 요구사항:**
   * Windows 운영체제
2. **실행:**
   * 다운 받은 파일 속 실행파일 더블 클릭하여 게임을 실행합니다.

\* 실행파일 경로: 스네이크게임\_팀K 의 Snake Game

이 가이드(보고서)를 참조하여 스네이크 게임 프로젝트를 성공적으로 사용할 수 있습니다.

감사합니다!