



국민대학교
소프트웨어융합대학
소프트웨어학부

C++프로그래밍 프로젝트

프로젝트 명	Snake Game
팀 명	21조
문서 제목	결과보고서

Version	1.2
Date	2022-JUN-15

팀원	20213000 박종혁
	20213021 신채원

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	21조	
	Confidential Restricted	Version 1.2	2022-JUN-15

CONFIDENTIALITY/SECURITY WARNING

이 문서에 포함되어 있는 정보는 국민대학교 소프트웨어융합대학 소프트웨어학부 및 소프트웨어학부 개설 교과목 **C++프로그래밍** 수강 학생 중 프로젝트 "**Snake Game**"을 수행하는 팀 "**21조**"의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 "**21조**"의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다.

문서 정보 / 수정 내역

Filename	최종보고서-Snake Game.doc
원안작성자	신채원
수정작업자	신채원

수정날짜	대표수정 자	Revision	추가/수정 항 목	내 용
2022-06-01	신채원	1.0	최초 작성	
2022-06-04	신채원	1.1	내용 수정	목표 달성 현황 수정
2022-06-15	신채원	1.2	내용 수정	자기평가 추가

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	21조	
	Confidential Restricted	Version 1.2	2022-JUN-15

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.....	7
2.2.3	활용/개발된 기술.....	26
2.2.4	현실적 제한 요소 및 그 해결 방안.....	27
2.2.5	결과물 목록.....	27
3	자기평가.....	29
4	부록.....	29
4.1	사용자 매뉴얼.....	30
4.2	설치 방법.....	31

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	21조	
	Confidential Restricted	Version 1.2	2022-JUN-15

1 개요

평가기준 (10점)

프로젝트를 완성하기 위해 사용한 개발 방법을 기술하세요.

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

해당 프로젝트는 C++를 사용하여 클래식 게임 중 하나인 Snake Game을 개발하는 것을 목적으로 하였다. 해당 게임은 정해진 공간 위에서 움직이며 아이템을 통해 몸 길이를 늘리고 줄이는 Snake Body, Snake가 통과할 수 없는 Immune Wall과 Wall, Snake의 몸 길이를 늘리는 Growth Item과 몸 길이를 줄이는 Poison Item, Snake가 통과하고 진출하게끔 하는 Gate로 구성되어 있다. 이때 Gate는 임의의 위치의 벽에서 나타나며, Immune Wall에서는 출현이 불가능하다.

해당 프로젝트는 ncurses 외부 라이브러리를 사용하여 개발하였다. ncurses는 Linux 환경에서 사용이 가능하게끔 만들어진 curses 라이브러리이다. ncurses는 터미널에서 sudo apt-get install 명령어를 사용하여 간단하게 설치가 가능하다. 다만 ncurses는 Windows 환경에서는 지원되지 않는 라이브러리이며, ncurses 라이브러리의 기능을 대부분 구현한 pdcurses를 설치해야만 한다. Windows에서 pdcurses를 설치하는 가장 흔한 방법은 Visual Studio 환경에서 vcpkg를 이용하는 것이다.

Visual Studio 환경이 갖춰져 있다는 가정 하에, vcpkg는 아래 Github (<https://github.com/Microsoft/vcpkg>)를 clone하는 방식을 통해 설치가 가능하다. 이후 명령 프롬프트에서 vcpkg install pdcurses를 입력해주면 pdcurses 설치가 완료된다. 다만 해당 명령어를 사용하여 설치하는 라이브러리는 32bit이기에, 64bit 버전으로 설치를 원할 시 vcpkg install pdcurses:x64-windows로 입력해야 정상적인 64bit 버전 설치가 완료된다.

Linux 환경과 Windows 환경에 대하여, 라이브러리 설치의 차이만이 있을 뿐, 실제 실행에는 문제가 되지 않으므로 주로 사용하는 환경에 맞춰 설치하는 것을 추천한다.

해당 프로젝트의 실현을 위해 Snake Body와 그 이동을 구현한 Snake.h와 Snake.cpp, 몸 길이를 늘이는 Growth.h와 Growth.cpp, 몸 길이를 줄이는 Poison.h와 Poison.cpp로 게임의 요소들을 각각 나누어 구현하였다. Snake Game의 실행을 위해서는 SanekGame.cpp를 실행시키면 되며, 해당 파일 실행 시 게임 화면 로딩과 함께 플레이가 가능하다. 각 단계 별 구현 및 설명은 아래에 이어진다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	21조	
	Confidential Restricted	Version 1.2	2022-JUN-15

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	미적용

1단계에서는 Snake Game을 플레이하는 공간이 되는 Map을 구현하고자 했다. Map은 ncurses를 사용하여 2차원 배열로 구현해야 하며, 해당 Map을 화면에 표시하도록 한다. 이때 Wall과 Immune Wall을 가시적으로 드러나도록 하여 플레이어가 구별할 수 있도록 하는 것을 목표로 하였다.

2단계에서는 1단계에서 구현한 Map 위에 Snake를 표시하고, 사전에 지정해 놓은 화살표를 통해 Snake의 이동을 조정하고자 하였다. 이때 Snake는 1단계에서 구현한 Immune Wall을 포함한 모든 Wall을 통과할 수 없다. 만일 Snake가 종류에 상관없이 Wall에 닿았을 경우 Game Over이 출력되도록 한다.

3단계에서는 Snake Game의 구성 요소들 중 Growth Item과 Poison Item의 출현 및 역할 수행을 목표로 하였다. 이때 Snake Body가 각각의 Item에 접근하는 것까지 고려하여, 각 아이템 흡수에 따라 Body가 늘어나거나 줄어드는 현상까지 수행되도록 하며, Item의 출현 위치는 Snake Body가 존재하지 않는 곳들 중 임의로 출현하게끔 해야만 한다.

4단계에서는 Wall의 임의의 위치에 Snake Body가 통과할 수 있는 Gate가 출현하게끔 하는 것을 목표로 정하였다. 이때 Gate는 Wall의 위치에 출현하지만, Wall과 시각적으로 구분될 수 있도록 구현 방법을 정해야 한다. 또한 Snake Game의 규칙에 따라 출현하는 Gate는 반드시 겹치지 않는 2개가 한 쌍이며, Snake가 Gate에 진입 시에는 사라지지 않아야만 한다. 또한 Gate를 통한 Snake의 이동은 정해진 규칙에 따라 이동하도록 해야만 한다.

마지막으로 5단계에서는 화면의 우측 부분에 현재까지의 게임 점수를 표시하는 부분을 구현하는 것을 목표로 한다. 게임 점수는 획득한 Item들의 수와 Gate 사용 횟수, Snake Body의 길이를 통해 정해진다. 이를 통해 각 Stage 별 Mission 달성 여부를 확인할 수 있으며, Mission 달성 시 다음 Stage로 진출할 수 있도록 한다.

프로젝트의 완성을 위한 세부 목표는 위와 같이 설정하였으며, 표에서도 알 수 있듯이 프로젝트 구현은 4단계까지 완료되었다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	21조	
	Confidential Restricted	Version 1.2	2022-JUN-15

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.

1. Map의 표현

1.1 Map의 크기

Row*Column의 21*21 크기의 정사각형으로 구현되었다. SnakeGame.cpp에서 integer형 변수인 mapsize를 통해 한 번의 길이(21)를 정의하였다.

1.2 Wall의 표현

Wall.h와 Wall.cpp를 통하여 Wall의 좌표를 설정하였다. Wall의 좌표는 [x][y]의 2차원 배열로 저장하였으며, [1][0]부터 [19][0], [0][1]부터 [0][19]와 같은 형식으로 좌측 세로, 우측 세로, 위측 가로, 아래쪽 가로의 순서로 저장되어 있다. Wall은 M을 이용하여 표현하였다.

2. Snake의 표현 및 조작

2.1 Snake 생성

Snake의 초기 몸 길이는 5로 지정하였으며, 해당 길이는 Snake.cpp에서 size 변수를 통해 확인할 수 있다. Map 내의 Snake Body의 표현은 0을 이용하여 "00000"과 같은 모습을 띄게끔 하였다.

2.2 방향키를 통한 Snake의 이동

a, w, d, s의 네 개의 키를 통하여 Snake의 움직임을 조정하도록 하였다. 일반적으로 생각하는 방향키에 대입한다면 a는 ←, w는 ↑, d는 →, s는 ↓에 연결시킬 수 있다.

a를 통하여 좌측으로, w를 통하여 위로, d를 통하여 우측으로, s를 통하여 아래로 1칸씩 이동할 수 있도록 하였다. 이동 방향과 같은 방향의 버튼이 눌렸을 때에는 동작하지 않도록 하였다.

2.3 Head 방향의 이동

Snake는 어떠한 버튼을 누르지 않더라도 일정 주기를 두고 자동으로 이동한다. 이때 주기는 time.h를 통해 계산하였으며, 이동 주기는 0.3초로 설정하였다.

2.4 Snake의 위치 및 길이를 통한 Game Over 여부 판단

Snake가 Wall에 닿으면 게임이 종료되도록 하였다. Wall에 닿았는지에 대해서는 Snake의 좌표를 저장하는 2차원 배열 body를 만들고, 이를 Map의 좌표와 비교하는 방식을 통해 판단하였다. 또한 Snake의 크기를 저장하는 size 변수를 생성하여 size가 1보다 작아졌을 경우에도 마찬가지로 게임 종료로 판단하도록 하였다.

3. Item 요소의 구현

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	21조	
	Confidential Restricted	Version 1.2	2022-JUN-15

- 3.1 Item의 출현
Item의 출현 주기는 3초로 설정하였으며, 마찬가지로 time.h를 통하여 주기를 계산하였다.
각 Item의 출현 위치는 rand() 함수를 통해 랜덤으로 결정해 주었으며, Map의 범위를 벗어나는 것을 방지하기 위해 모듈러 연산(rand() % 19)을 사용하였다.
- 3.2 Growth Item의 획득 시
Snake의 이동을 통해 Snake의 Head의 좌표와 Growth Item이 위치한 좌표가 동일해졌을 경우, Growth Item을 획득하였다고 판단하여 Snake의 Tail을 1만큼 증가시키고, Snake의 전체 몸 길이(size) 역시 1만큼 증가시킨다.
- 3.3 Poison Item의 획득 시
Growth Item과 같은 방식으로, Snake의 이동을 통해 Snake Head의 좌표와 Poison Item이 위치한 좌표가 동일해졌을 경우, Poison Item을 획득했다고 판단한다. Poison Item의 경우 Growth Item과는 반대로 Snake의 전체 몸 길이(size)를 1만큼 감소시키며, Tail 부분에서 1만큼 감소한다.

4. Gate 요소의 구현

- 4.1 Gate의 출현 위치 결정
Item의 출현 위치를 랜덤으로 결정했던 것과 마찬가지로, Gate의 위치 역시 rand() 함수와 모듈러 연산을 이용하여 랜덤으로 결정한다. 이때 Wall.cpp 내의 size 변수를 가지고 모듈러 연산을 수행해주며, Gate의 좌표는 Wall의 좌표를 표현하기 위해 사용한 2차원 배열 wall을 이용하여 표현하도록 하였다.
- 4.2 Gate를 통한 Snake의 이동
Gate를 통하여 Snake가 이동하는 것은 Gate.cpp가 아닌 Snake.cpp에서 구현되었다. 나오는 Gate가 좌측 Wall에 붙어있을 시에는 오른쪽으로 진출, 우측 Wall에 붙어있을 시에는 왼쪽으로 진출, 위쪽 Wall에 붙어있을 시에는 아래쪽으로 진출, 아래쪽 Wall에 붙어있을 시에는 위쪽으로 진출하도록 이동 방향을 설정하였다.

2.2.2 시스템 구조 및 설계도

작성요령 (30 점)

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.

아래에서 설명하는 모든 소스 파일에 대하여 20213000 박종혁 학우가 개발하였음을 미리 밝힌다.

게임을 진행하며 계속 길이를 변화하는 Snake 를 구현한 코드를 먼저 소개 후, 이후 Wall, Item, Gate 등 게임의 구성 요소를 구현한 코드를 설명하는 순서로 진행할 것이다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	21조	
	Confidential Restricted	Version 1.2	2022-JUN-15

1. Snake.h

```

#ifndef __Snake_h__
#define __Snake_h__

#include <iostream>
#include <vector>

#include "Growth.h"
#include "Poison.h"
#include "Gate.h"

using namespace std;

class Growth;
class Poison;
class Gate;

class Snake
{
private:
    int body[20][2];
    int size;
    int direction;

public:
    Snake();
    void move(int dir);
    int getSize();
    int getBody_X(int n);
    int getBody_Y(int n);

    void iseatG(Growth G);
    void iseatP(Poison P);
    void is_enter_gate(Gate G);

    bool SnakeIsDead(Gate G);
};

#endif

```

- body[20][2] : Snake 의 좌표를 [x][y]의 형태로 저장한 2 차원 배열이다. 이후 저장된 배열의 값을 통해 00000 와 같은 형태로 Snake 를 Map 에 표현한다.
- size : Snake 의 몸 길이를 의미하는 변수이며, 정수만이 가능하기에 int 로 지정하였다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	21조	
	Confidential Restricted	Version 1.2	2022-JUN-15

- direction : Snake 의 이동 방향을 지시하는 변수이다. 이때 char 이 아닌 int 로 선언한 이유는 이후 direction 에 입력되는 값이 아스키 코드로 변환된 값이기 때문이다.
- Snake() : Snake Class 의 생성자이다.
- void move(int dir) : 입력된 direction 을 받아 Snake 를 움직인다.
- getSize() : Snake 의 몸 길이를 return 한다.
- getBody_X(int n) : Head 를 기준으로 했을 때, Snake 의 n 번째 위치의 x 좌표를 return 한다.
- getBody_Y(int n) : Head 를 기준으로 했을 때, Snake 의 n 번째 위치의 y 좌표를 return 한다.
- iseatG(Growth G) : Snake 가 Growth Item 을 획득했을 때 수행되는 함수이다.
- iseatP(Poison P) : Snake 가 Position Item 을 획득했을 때 수행되는 함수이다.
- is_enter_gate(Gate G) : Snake 가 Gate 를 통과할 때 수행되는 함수이다.

2. Snake.cpp

```
#include "Snake.h"

#define mapsize 21

Snake::Snake ()
{
    size = 5;
    body[0][0] = 11;
    body[0][1] = 11;

    body[1][0] = 11;
    body[1][1] = 12;

    body[2][0] = 11;
    body[2][1] = 13;

    body[3][0] = 11;
    body[3][1] = 14;

    body[4][0] = 11;
    body[4][1] = 15;
}

int Snake::getSize()
{
```



```
return size;
}

int Snake::getBody_X(int n)
{
    return body[n][0];
}

int Snake::getBody_Y(int n)
{
    return body[n][1];
}

bool Snake::SnakeIsDead(Gate G)
{
    if(body[0][0] == G.getX1() && body[0][1] == G.getY1())
        return 0;

    if(body[0][0] == G.getX2() && body[0][1] == G.getY2())
        return 0;

    if(body[0][0] == mapsize-1 || body[0][0] == 0 || body[0][1] == 0
|| body[0][1] == mapsize -1)
        return 1;

    for(int i = 1; i < size; i++)
        if(body[0][0] == body[i][0] && body[0][1] == body[i][1])
            return 1;
    if(size < 1)
        return 1;

    return 0;
}

void Snake::move(int dir)
{
    direction = dir;
    for(int i = size-1; i > 0; i--)
    {
        body[i][0] = body[i-1][0];
        body[i][1] = body[i-1][1];
    }
    if(direction == 97) // a
        body[0][1]--;
    else if (direction == 119) // w
        body[0][0]--;
    else if (direction == 100) // d
        body[0][1]++;
    else if (direction == 115) // s
```



```
        body[0][0]++;
    }

    void Snake::iseatG(Growth G)
    {
        if(body[0][0] == G.getX() && body[0][1] == G.getY())
        {
            body[size][0] = body[size-1][0] + (body[size-1][0] -
body[size-2][0]);
            body[size][0] = body[size-1][0] + (body[size-1][0] -
body[size-2][0]);
            size++;
        }
    }

    void Snake::iseatP(Poison P)
    {
        if(body[0][0] == P.getX() && body[0][1] == P.getY())
            size--;
    }

    void Snake::is_enter_gate(Gate G)
    {
        if(body[0][0] == G.getX1() && body[0][1] == G.getY1())
        {
            if(G.getX2() == 0)
            {
                body[0][0] = G.getX2() + 1;
                body[0][1] = G.getY2();
                direction = 'd';
            }
            else if(G.getY2() == 0)
            {
                body[0][0] = G.getX2();
                body[0][1] = G.getY2() + 1;
                direction = 's';
            }
            else if(G.getX2() == 20)
            {
                body[0][0] = G.getX2() - 1;
                body[0][1] = G.getY2();
                direction = 'a';
            }
            else if(G.getY2() == 20)
            {

```



```
        body[0][0] = G.getX2();
        body[0][1] = G.getY2() - 1;
        direction = 'w';
    }
}

else if(body[0][0] == G.getX2() && body[0][1] == G.getY2())
{
    if(G.getX1() == 0)
    {
        body[0][0] = G.getX1() + 1;
        body[0][1] = G.getY1();
        direction = 'd';
    }
    else if(G.getY1() == 0)
    {
        body[0][0] = G.getX1();
        body[0][1] = G.getY1() + 1;
        direction = 's';
    }
    else if(G.getX1() == 20)
    {
        body[0][0] = G.getX1() - 1;
        body[0][1] = G.getY1();
        direction = 'a';
    }
    else if(G.getY1() == 20)
    {
        body[0][0] = G.getX1();
        body[0][1] = G.getY1() - 1;
        direction = 'w';
    }
}
}
```

- Snake() : Snake의 초기 크기는 5로 지정하였으며, 각 부분의 좌표를 (11, 11)에서 (11, 15)로 지정해주었다.
- getSize() : Snake의 몸 길이를 return 해주도록 하였다.
- getBody_X(int n), getBody_Y(int n) : Snake의 n번째 몸에 대한 x 좌표와 y 좌표를 각각 return 해주도록 하였다.
- SnakeIsDead(Gate G) : 몸 길이, 위치 이동 등 Snake의 다양한 변화에 대해 Snake의 죽음 여부를 판단해주었다.

첫 번째와 두 번째 경우는 Snake가 Gate를 통과할 때이므로 return 0을 해 주어 Snake가 죽지 않았음으로 판단한다. 세 번째 경우는 Snake가 Wall에 닿았을 경우로, 해당 경우에는 return 1을 해 주어 Snake의 죽음을 알린다. 그 아래의 for

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	21조	
	Confidential Restricted	Version 1.2	2022-JUN-15

반복문과 if 문은 모두 같은 상황을 이야기하는데, Snake 의 head 가 뒤의 몸통과 같은 좌표를 가지거나, Snake 의 몸 길이가 1 보다 작을 경우, return 1 을 통하여 Snake 의 죽음을 알린다.

- move(int dir) : 입력 받은 dir 에 따라 snake 을 움직여준다. a 가 입력되었을 시 왼쪽으로, w 가 입력되었을 시 위쪽으로, d 가 입력되었을 시 오른쪽으로, s 가 입력되었을 시 아래쪽으로 한 칸씩 snake 를 이동시켜 준다. 이를 Snake 의 각각의 좌표를 저장해 둔 2 차원 배열 body 의 값을 변경하여 Snake 의 좌표를 변경시켜 준다.
- iseatG(Growth G) : Snake 가 Growth Item 을 획득했을 때 실행되는 함수이다. 마찬가지로 2 차원 배열 body 의 값을 변경해 줌으로써 Snake 좌표를 변경시키는 방식을 사용하고 있다. 또한 Growth Item 획득 시 길이가 1 증가함에 따라, snake 의 몸 길이인 size 역시 1 증가시켜 주었다.
- iseatP(Poison P) : Snake 가 Poison Item 을 획득했을 때 실행되는 함수이다. iseatG(Growth G)와 원리는 동일하며, Poison Item 획득 시 길이가 1 감소하므로 snake 의 몸 길이인 size 를 1 감소시켜 주었다.
- Is_enter_gate(Gate G) : Snake 가 Gate 를 통과할 때 실행되는 함수이다. Gate 는 두 개가 한 쌍을 이루므로 경우의 수를 gate 1 을 통과할 때, gate 2 를 통과할 때의 두 가지로 나누었다. 즉 body[0][0] == G.getX1() && body[0][1] == G.getY1() 일 때에는 Snake 가 gate 1 을 통과 중인 것이며, body[0][0] == G.getX2() && body[0][1] == G.getY2() 일 때에는 Snake 가 gate 2 를 통과 중인 것이다.

두 경우에 대하여 내부 논리는 동일하다. 만일 통과하여 나오게 되는 gate 가 우측에 위치해 있을 시 Map 의 내부 방향인 좌측으로 이동하게 되며, 좌측에 있을 시 우측으로, 위쪽에 있을 시 아래쪽으로, 아래쪽에 있을 시 위쪽으로 이동하게 된다.

3. Wall.h

```
#ifndef __Wall_h__
#define __Wall_h__

#include <iostream>

class Gate;

class Wall
{
private:
    int stage;
    int wall[400][2];
    int size;
    friend class Gate;
```



```
public:
    Wall(int n);

};

#endif
```

- stage : Snake Game 의 현재 단계를 의미한다.
- wall : Wall 의 좌표를 [x][y] 형태로 저장한 2 차원 배열이다.
- friend class Gate : class Gate 에서 2 차원 배열 wall 에 접근해야 하기 때문에 friend 를 선언하였다. gate 는 wall 의 위치에 임의로 생성되기 때문에 Wall class 의 wall 에 접근하는 것이 효율적이라고 판단하였다.

4. Wall.cpp

```
#include "Wall.h"

Wall::Wall(int stage)
{
    int n = 0;
    for(int i = 1; i < 20; i++)
    {
        wall[n][0] = i;
        wall[n][1] = 0;
        size++;
        n++;
    }
    for(int i = 1; i < 20; i++)
    {
        wall[n][0] = i;
        wall[n][1] = 20;
        size++;
        n++;
    }
    for(int i = 1; i < 20; i++)
    {
        wall[n][1] = i;
        wall[n][0] = 0;
        size++;
        n++;
    }
    for(int i = 1; i < 20; i++)
    {

```

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	21조	
	Confidential Restricted	Version 1.2	2022-JUN-15

```

        wall[n][1] = i;
        wall[n][0] = 20;
        size++;
        n++;
    }
}

```

좌측 세로, 우측 세로, 위측 가로, 아래측 가로 순서로 wall 의 좌표를 저장하였다.

wall 은 Map 의 가장자리에 형성되기에 x 좌표와 y 좌표 중 하나를 0 또는 20 으로 고정시키고, for 반복문을 활용하여 해당 행 혹은 열을 전부 저장할 수 있도록 하였다.

5. Growth.h

```

#ifndef __Growth_h__
#define __Growth_h__

#include "Snake.h"

#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

class Snake;

class Growth
{
private:
    int X;
    int Y;

public:
    Growth();
    int getX();
    int getY();
    void set(Snake s);

};

#endif

```

- X, Y : Growth Item 이 나타나는 x 좌표와 y 좌표를 의미한다.
- getX(), getY() : 각각 Growth Item 의 x 좌표와 y 좌표를 return 한다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	21조	
	Confidential Restricted	Version 1.2	2022-JUN-15

- set(Snake s) : Snake 가 존재하지 않는 위치 중 Growth Item 이 출현할 위치를 랜덤으로 결정한다.

6. Growth.cpp

```
#include "Growth.h"

Growth::Growth()
{
    srand(time(NULL));
    X = 1;
    Y = 1;
}

int Growth::getX()
{
    return X;
}

int Growth::getY()
{
    return Y;
}

void Growth::set(Snake s)
{
    int n = 0;

    do
    {
        n = 0;
        X = rand() % 19 + 1;
        Y = rand() % 19 + 1;

        for(int i = 0; i < s.getSize(); i++)
        {
            if(X == s.getBody_X(i) && Y == s.getBody_Y(i))
            {
                n++;
                break;
            }
        }

    }while(n != 0);
}
```


 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	21조	
	Confidential Restricted	Version 1.2	2022-JUN-15

rand() 함수와 모듈러 연산(rand() % 19)을 이용하여 Growth Item 이 출현할 위치를 랜덤으로 결정하였다. rand() 함수로 Map 의 범위가 벗어나는 것을 모듈러 연산을 이용하여 방지해주는 방식을 선택하였다.

또한 for 반복문을 통해 Snake 의 각 좌표와 랜덤으로 결정된 Item 의 위치를 비교하며 Item 의 위치가 Snake 의 좌표와 겹칠 경우, 다시 rand()를 통해 재결정하는 방식을 do-while 문을 이용하여 구현하였다.

7. Poison.h

```
#ifndef __Poison_h__
#define __Poison_h__

#include "Snake.h"

#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

class Snake;

class Poison
{
private:
    int X;
    int Y;

public:
    Poison();
    void init();
    int getX();
    int getY();
    void set(Snake s);

};

#endif
```

- X, Y : Poison Item 이 나타나는 x 좌표와 y 좌표를 의미한다.
- getX(), getY() : 각각 Poison Item 의 x 좌표와 y 좌표를 return 한다.
- set(Snake s) : Snake 가 존재하지 않는 위치 중 Poison Item 이 출현할 위치를 랜덤으로 결정한다.



8. Poison.cpp

```
#include "Poison.h"

Poison::Poison()
{
    srand(time(NULL));
    X = 1;
    Y = 1;
}

int Poison::getX()
{
    return X;
}

int Poison::getY()
{
    return Y;
}

void Poison::set(Snake s)
{
    int n = 0;

    do
    {
        n = 0;
        X = rand() % 19 + 1;
        Y = rand() % 19 + 1;

        for(int i = 0; i < s.getSize(); i++)
        {
            if(X == s.getBody_X(i) && Y == s.getBody_Y(i))
            {
                n++;
                break;
            }
        }

    }while(n != 0);
}
```

rand() 함수와 모듈러 연산(rand() % 19 + 1)을 이용하여 Poison Item 이 출현할 위치를 랜덤으로 결정하였다. rand() 함수로 Map 의 범위가 벗어나는 것을 모듈러 연산을 이용하여 방지해주는 방식을 선택하였다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	21조	
	Confidential Restricted	Version 1.2	2022-JUN-15

또한 for 반복문을 통해 Snake 의 각 좌표와 랜덤으로 결정된 Item 의 위치를 비교하며 Item 의 위치가 Snake 의 좌표와 겹칠 경우, 다시 rand()를 통해 재결정하는 방식을 do-while 문을 이용하여 구현하였다.

9. Gate.h

```
#ifndef __Gate_h__
#define __Gate_h__

#include "Snake.h"
#include "Wall.h"

#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

class Snake;
class Wall;

class Gate
{
private:
    int X1, X2, Y1, Y2;

public:
    Gate();
    int getX1();
    int getY1();
    int getX2();
    int getY2();
    void set(Wall& w);

};

#endif
```

- X1, X2, Y1, Y2 : 한 쌍의 Gate 에 대하여 (X1, Y1), (Y1, Y2)의 위치를 갖는 Gate 의 좌표를 의미한다.
- getX1, getY1(), getX2(), getY2() : Gate 의 각각의 x, y 좌표를 return 한다.
- set(Wall& w) : gate 의 좌표를 Wall 의 좌표를 저장해 놓은 wall 에 저장한다.

10. Gate.cpp



```
#include "Gate.h"

Gate::Gate()
{
    srand(time(NULL));
}

int Gate::getX1()
{
    return X1;
}

int Gate::getY1()
{
    return Y1;
}

int Gate::getX2()
{
    return X2;
}

int Gate::getY2()
{
    return Y2;
}

void Gate::set(Wall& w)
{
    int n1 = rand() % w.size;
    int n2 = rand() % w.size;
    while(n2 == n1)
        n2 = rand() % w.size;

    X1 = w.wall[n1][0];
    Y1 = w.wall[n1][1];

    X2 = w.wall[n2][0];
    Y2 = w.wall[n2][1];
}
```

rand() 함수와 모듈러 연산(rand()%w.size)를 이용해 gate의 좌표를 임의로 선택한다.

이때 Gate는 반드시 2개가 1쌍으로 움직이기 때문에 n1과 n2를 별개로 생성하게 되며, n1과 n2는 동일한 값을 가져서는 안 된다. 이를 해결하기 위해 while문을 사용하여 n2가

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	21조	
	Confidential Restricted	Version 1.2	2022-JUN-15

n1 과 같은 값이 나올 동안은 계속해서 rand() 함수를 다시 수행해줌으로써 n1 과 n2 가 다른 값을 가지게끔 한다.

11. SnakeGame.cpp

```
#include <ncursesw/curses.h>
#include <locale.h>
#include <iostream>
#include <time.h>
#include <unistd.h>

#include "Snake.h"
#include "Growth.h"
#include "Poison.h"
#include "Wall.h"
#include "Gate.h"

#include <sys/select.h>
#include <unistd.h>
#include <termios.h>

static struct termios initial_settings, new_settings;
static int peek_character = -1;
void init_keyboard()
{
    tcgetattr(0, &initial_settings);
    new_settings = initial_settings;
    new_settings.c_lflag &= ~ICANON;
    new_settings.c_lflag &= ~ECHO;
    new_settings.c_cc[VMIN] = 1;
    new_settings.c_cc[VTIME] = 0;    tcsetattr(0, TCSANOW,
&new_settings);
}

void close_keyboard()
{
    tcsetattr(0, TCSANOW, &initial_settings);
}

int _kbhit()
{
    unsigned char ch;
    int nread;

    if (peek_character != -1)
        return 1;

    new_settings.c_cc[VMIN]=0;
```



```
tcsetattr(0, TCSANOW, &new_settings);

nread = read(0, &ch, 1);
new_settings.c_cc[VMIN]=1;

tcsetattr(0, TCSANOW, &new_settings);

if(nread == 1)
{
    peek_character = ch;
    return 1;
}
return 0;
}

int _getch()
{
    char ch;
    if(peek_character != -1)
    {
        ch = peek_character;
        peek_character = -1;
        return ch;
    }

    read(0, &ch, 1);
    return ch;
}

int _putch(int c)
{
    putchar(c);
    fflush(stdout);
    return c;
}

int main()
{
    Snake s;
    Growth G;
    Poison P;

    setlocale(LC_ALL, "");

    int mapsize = 21;

    WINDOW * snake_win;
```



```
initscr();
keypad(stdscr, TRUE);
curs_set(0);
noecho();

resize_term(25,80);
start_color();

border('|', '|', '-', '-', '+', '+', '+', '+');
mvprintw(1, 35, "SNAKE GAME!");

mvprintw(10, 35, "Choose stage 1~4!");

int stage;

Wall w(1);
Gate gate;
refresh();

mvprintw(10, 35, "
");
refresh();

snake_win = newwin(mapsize, mapsize, 2, 2);
wborder(snake_win, 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M');

for(int i = 0; i < s.getSize(); i++)
{
    mvprintw(snake_win, s.getBody_X(i), s.getBody_Y(i), "0");
}

wrefresh(snake_win);

clock_t start, finish;
double duration;

start = clock();

int ch = 97;
int mv_cnt = 0;

mvprintw(snake_win, gate.getX1(), gate.getY1(), "M");
mvprintw(snake_win, gate.getX2(), gate.getY2(), "M");
gate.set(w);

while(!s.SnakeIsDead(gate) && ch != '1')
{
```



```
finish = clock();
duration = (double)(finish - start) / CLOCKS_PER_SEC;

if(_kbhit())
{
    ch = _getch();
    _putch(ch);
}

for(int i = 0; i < s.getSize()+1; i++)
    mvwprintw(snake_win, s.getBody_X(i), s.getBody_Y(i), "
");

if(duration >= 0.3)
{
    s.move(ch);
    mv_cnt++;

if(mv_cnt % 10 == 0)
{
    mvwprintw(snake_win, G.getX(), G.getY(), " ");
    G.set(s);
    mvwprintw(snake_win, G.getX(), G.getY(), "G");

    mvwprintw(snake_win, P.getX(), P.getY(), " ");
    P.set(s);
    mvwprintw(snake_win, P.getX(), P.getY(), "P");
}

/* if(mv_cnt % 10 == 0)
{
    mvwprintw(snake_win, gate.getX1(), gate.getY1(), "M");
    mvwprintw(snake_win, gate.getX2(), gate.getY2(), "M");
    gate.set(w);
    mvwprintw(snake_win, gate.getX1(), gate.getY1(), "D");
    mvwprintw(snake_win, gate.getX2(), gate.getY2(), "D");
}*/
s.iseatG(G);
s.iseatP(P);
start = clock();
}

for(int i = 0; i < s.getSize(); i++)
    mvwprintw(snake_win, s.getBody_X(i), s.getBody_Y(i),
"0");
```




```
s.is_enter_gate(gate);

wrefresh(snake_win);
mvprintw(5, 60, "time : %d seconds ", mv_cnt/2);
mvprintw(6, 60, "score : %d !!!! ", s.getSize());
mvprintw(10, 60, "input : ");

mvwprintw(snake_win, gate.getX1(), gate.getY1(), "D");
mvwprintw(snake_win, gate.getX2(), gate.getY2(), "D");

refresh();
}
delwin(snake_win);

mvprintw(11, 40, "Game Over!!");
getch();
endwin();

return 0;
}
```

- init_keyboard(), close_keyboard(), _kbhit(), _getch(), _putch(int c) : 키보드 입력 이벤트를 감지하기 위해 생성된 함수들이다. Linux에서는 별도로 키보드 입력 이벤트를 감지할 수 있는 함수가 없어서, 따로 생성해 주어야만 했다.
- Snake s, Growth G, Poison P를 선언해주며 객체들을 초기화해준다.
- Wall w(1), Gate gate를 선언해주며 객체들을 초기화해준다.
- mapsize라는 변수를 선언 후 21로 초기화해 주고, 이후 게임이 진행되는 Map은 21*21의 정사각형 모양으로 형성한다. 이후 for 반복문을 사용해 Snake class에서 Snake의 좌표를 읽어와 Map 위에 00000의 형태로 시각화한다.
- 게임이 진행되는 Map은 중간중간, Map에 시각적인 변화가 일어날 때마다 refresh()를 통해 업데이트를 해 준다.
- Int ch = 97 : Snake의 기본 이동 방향(a, 아스키 코드로 97)이다. 어떠한 값도 입력되지 않았을 때 Snake는 왼쪽으로 계속 이동해야 하므로 ch를 97로 초기화한다.
- Int mv_cnt = 0 : Snake가 이동한 횟수를 의미한다. 이때 키보드 입력을 통해서가 아닌, 틱에 의한 자동적인 이동 역시 mv_cnt의 횟수로 포함시킨다.
- while(!s.SnakeIsDead(gate) && ch != 'I') : Snake가 살아있을 동안 아래의 코드가

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	21조	
	Confidential Restricted	Version 1.2	2022-JUN-15

계속해서 돌아간다. 또한 while loop의 시작과 끝에서 현재 시간을 clock()를 통해 계산해줌으로써 시간이 틱(주기)을 넘겼는지 확인 가능하게끔 한다.

if(duration >= 0.3)을 통해 0.3초가 지났을 경우 Snake를 기존의 이동 방향으로 계속 이동하게끔 한다.

If(mv_cnt % 10 == 0)을 통해 3초가 지났을 경우 Growth Item과 Position Item을 Map에 랜덤으로 세팅해준다. 또한 Snake가 해당 Item을 획득하였는지를 확인한다.

- 또한 Snake가 gate를 통과하였는지를 확인하고, 통과하였을 경우 그에 맞는 이동을 취하게끔 한다.
- Map의 우측 밖에 time와 score, input을 표시한다.
time은 게임이 시작한 후 현재까지 걸린 시간, score은 Snake의 몸 길이를 나타낸다. Input은 방향 버튼 중 어떤 것을 눌렀는지를 사용자에게 알리는 역할을 한다.
- 만일 Snake가 죽고 게임이 종료되었을 경우, 화면에 “Game Over!!”을 띄움으로써 사용자에게 게임이 종료되었음을 알린다.

2.2.3 활용/개발된 기술

작성요령 (10 점)

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. ncursesw/curses.h

- GUI 기반의 게임을 만들기 위해 사용한 라이브러리이다.
- Map 위에 Snake, Item, Gate, Wall 등 게임의 모든 요소를 시각적으로 구현하였다.
- 매번 refresh()를 통해 변경사항이 화면에 나타나도록 하였다.

2. cstdlib

- rand() 함수를 사용하기 위해 import 하였다.
- Growth Item, Poison Item, Gate 의 위치를 임의로 생성하기 위해 사용되었다.

3. time.h

- clock() 함수를 통해 시작 시간과 종료 시간을 계산하였으며, 이로부터 일정 주기(틱, 0.3 초로 설정)가 지났는지를 확인하였다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	21조	
	Confidential Restricted	Version 1.2	2022-JUN-15

- 계산한 시간 차이가 0.3 초 이상일 경우 snake 를 이동 방향으로 한 칸만큼 더 움직이게끔 하기 위해 사용하였다.

2.2.4 현실적 제한 요소 및 그 해결 방안

작성요령 (5 점)

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

키보드 입력 이벤트를 감지할 때, `getchar()` 함수는 타임아웃이라는 개념이 존재하지 않는다. 즉, 다음 입력이 들어올 때까지 무한히 대기하는 함수이며, 일정 시간 동안 입력이 들어오지 않을 시 기존의 진행 방향으로 계속 진출해야만 하는 Snake Game에서 사용하기는 다소 어려움이 있었다.

이를 해결하기 위해 키보드 입력 이벤트를 감지하는 함수인 `_kbhit()` 을 따로 구현하였다. Windows와 다르게 Linux에는 키보드 이벤트 발생을 감지하는 함수가 따로 존재하지 않아, 해당 함수를 작성해야만 했다.

`_kbhit()` 외에도 `_getch()`, `_putch(int c)` 의 함수가 사용되었는데, `_putch(int c)` 는 `_getch()` 호출 시 화면 출력 없이 바로 input buffer에서 읽어 오기 때문에, 따로 사용자에게 보여주기 위해 출력해주는 함수라고 볼 수 있다.

2.2.5 결과물 목록

작성요령 (5 점)

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

1. SnakeGame.cpp	아래의 모든 파일들을 포함한, Snake Game 을 실행하기 위한 main 파일이다. <code>main()</code> 함수가 이곳에 존재한다.
2-1. Snake.h	Snake Body 에 대한 조정을 다루고 있는 파일이다.
2-2. Snake.cpp	Snake 의 이동, 몸 길이의 증가 및 감소 등을 다룬다. Item 의 획득으로 인한 body 의 변화 역시 이곳에서 다룬다.
3-1. Wall.h	Map 내의 Wall 에 대한 구현을 다루고 있는 파일이다.
3-2. Wall.cpp	
4-1. Growth.h	출현하는 Item 들 중 Growth Item 에 대해 다루고 있는 파일이다. Growth Item 의 출현 위치 결정 및 Growth Item 의 출현을 담당하고 있다.
4-2. Growth.cpp	
5-1. Poison.h	

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	21조	
	Confidential Restricted	Version 1.2	2022-JUN-15

5-2. Poison.cpp	출현하는 Item 들 중 Poison Item 에 대해 다루고 있는 파일이다. Poison Item 의 출현 위치 결정 및 Poison Item 의 출현을 담당하고 있다.
6-1. Gate.h	Snake 의 이동이 가능한 Gate 의 구현에 대해 다루고 있는 파일이다. Gate 의 출현 위치 결정이 해당 파일에서 이루어진다.
6-2. Gate.cpp	

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	21조	
	Confidential Restricted	Version 1.2	2022-JUN-15

3 자기평가

작성요령 (5 점)

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.

20213000 박종혁

SnakeGame 을 메인 파일로 해서 snake, Growth, Poison, Wall, Gate 클래스를 구현해서 객체지향적 프로그래밍을 시도하였다. Snake 의 행동 반경이나, 아이템을 먹었을 때, 포탈에 진입했을 때의 상황을 움직이는 주체가 snake 이므로 여기에 몰아서 클래스를 작성하였다.

메인 파일인 snakeGame 이 읽기 복잡한 부분이 있어 이부분은 조금 아쉬웠다.

프로젝트 진행 중 가장 어려웠던 점은 키보드 입력 이벤트 감지 부분이었다. getchar() 함수는 키보드의 입력이 들어올 때까지 무한으로 대기하는 성질이 있는데 이는 키보드 입력을 하지 않아도 진행방향으로 계속 나아가야하는 snake 의 특성과 맞지 않아 관련 함수와 API 문서를 찾아가며 키보드 입력 이벤트 감지 함수를 작성하였다.

20213021 신채원

구현이 완료된 코드를 가지고 일부 주석 처리를 해 가며 흐름을 파악하였으며, 이를 바탕으로 보고서를 작성하였다. 각 요소들의 역할 및 동작 방식을 프로젝트 안내 pdf 를 분석하며 정리하였고, 이를 Snake, Growth, Poison 등의 class 와 비교해가며 각 변수 및 함수의 존재 이유를 파악하였다.


마찬가지로 메인 파일인 SnakeGame.cpp 를 해석할 때 다소 어려움을 겪었다. Snake Game 의 특성 상 고려해야 할 요소들(Item 획득, Gate 통과 등)이 많았던 것이 그 원인으로 추측된다.

그러나 본인의 것이 아닌 코드를 분석하며 C++를 활용한 객체지향적 구현에 대해 여러 방면으로 배워갈 수 있는 기회를 얻었다고 생각한다.

4 부록

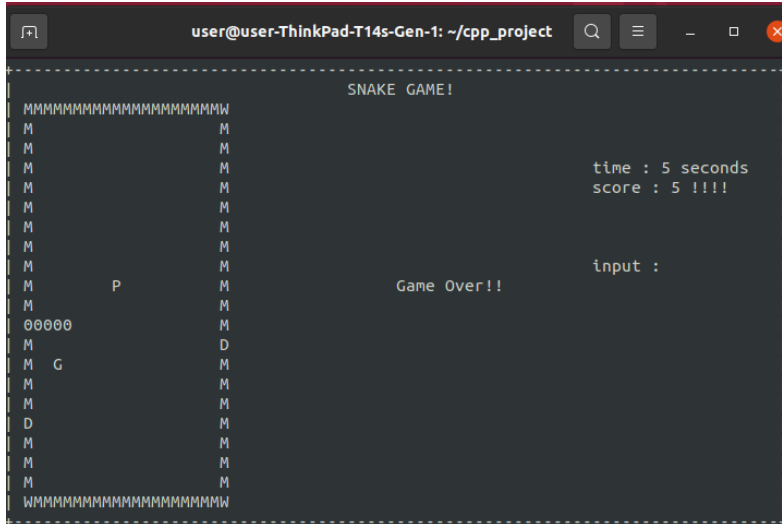
작성요령 (15 점)

프로젝트의 결과물을 사용하기 위한 방법에 대해서 작성하세요.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	21조	
	Confidential Restricted	Version 1.2	2022-JUN-15

- G 는 Gate 로, 한쪽 gate 로 진입 시 반대쪽 gate 로 나올 수 있다.
- M 은 Wall 로, Snake 가 Wall 에 닿게 되면 게임이 종료된다.

3. 게임 종료 화면



- Wall 에 닿았거나, 몸 길이가 일정 길이 이하가 되었을 때 게임이 종료된다.
- Game Over 시 다음과 같은 화면이 표시된다.
- 이후 아무 키나 입력하면 게임 화면을 끌 수 있다.

4.2 설치 방법

해당 프로젝트의 설치 및 실행을 위해서는 Linux Ubuntu 18.04, C++ 14, g++ 컴파일 환경이 요구된다.

결과물 목록에 작성되어 있는 모든 파일을 https://github.com/toddla23/cpp_project 에서 다운로드 후, 터미널에 다음과 같이 입력한다. 차례대로 컴파일, 실행에 해당된다고 할 수 있다.

- g++ -std=c++11 -o game SnakeGame.cpp Snake.h Snake.cpp Growth.h Growth.cpp Poison.h Poison.cpp Wall.h Wall.cpp Gate.h Gate.cpp -lncursesw
- ./game