|  |
| --- |
| [**C++프로그래밍및실습**](https://sel.jnu.ac.kr/course/view.php?id=55996)  **C++ 미니게임**  **최종 보고서** |

|  |  |
| --- | --- |
|  | 제출일자: 2024-12-20  제출자명: 김휘승  제출자학번: 214958 |
|  |  |

**1. 프로젝트 목표 (16 pt)**

**1) 배경 및 필요성 (14 pt)**

수업시간에 terminal 환경에서 수행하는 기초적인 게임들을 배워봤습니다. 흥미로 운 게임들이지만, 2인 이상의 플레이어가 필요하거나 볼륨이 작은 게임이 많았습니다. 과제 주제를 고민하던 중, 어렸을 때 폴더폰으로 즐겼던 ‘미니게임 천국’이 떠올랐고, 이를 참고해 나만의 ‘미니게임 천국’을 만들면 컴퓨터와 함께 혼자서도 충분히 즐길 수 있는 환경을 제공할 수 있을 것이라 생각했습니다. 또한, 여러 작 은 게임을 하나의 프로그램으로 묶음으로써 볼륨 문제도 해결할 수 있다고 판단했습니다. 이와 같은 이유로 나만의 ‘미니게임 천국’을 만들어 보기로 했습니다.

**2) 프로젝트 목표**

터미널 환경에서 컴퓨터와 함께 혼자서 즐길 수 있고, 여러가지 게임을 담은 나 만의 ‘미니게임 천국’ 만들기

**3) 차별점**

1. 컴퓨터와의 상호작용: 모든 미니게임에서 컴퓨터가 플레이어의 행동에 반응하며 특정 결과를 제공합니다. 이를 통해 혼자서도 재미있게 즐길 수 있습니다.

2. 점수와 보너스 시스템: 각 미니게임에서 승리할 때마다 점수가 주어지며, 누적된 점수가 일정 기준에 도달하면 다음 게임에서 보너스를 받을 수 있습니다.

3. 랜덤 플레이 모드: 플레이할 게임이 무작위로 결정되는 랜덤 플레이 모드를 통 해 게임에 신선함을 더했습니다.

**2. 기능 계획**

**1) 기능 1 : 게임 구현하기**

**- 설명: 게임은 총 4개 (묵찌빠, 틱택토, 숫자 배열하기, 단어 맞추기)로 구성한다.**

(1) 세부 기능 1: 묵찌빠 & 틱택토 게임&컴퓨터 행동 알고리즘

- 설명: 플레이어의 행동 후 다음 행동은 컴퓨터가 수행합니다. 컴퓨터의 행동은 랜덤으로 결정되나, 룰을 준수해야 합니다. (예: 컴퓨터가 돌이 놓여있는 칸에 돌을 놓지 않는다.)

(2) 세부 기능 2: 단어 맞추기 게임& 단어 불러오기

- 설명: 컴퓨터가 제공할 단어는 텍스트 파일에서 불러오고, 텍스트 파일은 단어 주제에 따라 분류합니다. (동물, 과일, 나라이름)

(3) 세부 기능 3 숫자 배열하기 게임& 숫자 배열 알고리즘

- 설명: 컴퓨터가 무작위 숫자를 제공하고, 컴퓨터는 merge sort 알고리즘을 통해 숫자를 크기순으로 정렬합니다. 유저는 특정한 시간 안에 숫자를 크기순으로 정렬해야 합니다.

(4) 세부 기능 4: 누적 점수 기능

- 설명: 게임에서 이기면 점수가 주어지고 누적 점수가 일정 점수를 넘으면 다음 게임에 서 보너스를 제공합니다. 묵찌빠는 패배 방어 1회, 틱택토는 추가 행동 1회, 숫자 배열하 기는 추가 시간 5초, 단어 맞추기는 질문권을 3회 더 제공합니다.

**2) 기능 2 : 메뉴 구현하기**

**- 설명: 메뉴에서 플레이할 게임을 선택합니다.**

(1) 세부 기능 1: 게임 선택하기

- 설명: 플레이할 게임을 플레이어가 선택합니다.

(2) 세부 기능 2: 랜덤 플레이

- 설명: 플레이할 게임이 랜덤으로 결정합니다.

(3) 세부 기능 2: 설명 & 점수 표시

- 설명: 메인 메뉴에서 누적 점수를 확인하고, 각 게임의 설명을 제공합니다.

**3. 기능구현**

**0) 게임 구현의 전체적인 특징**

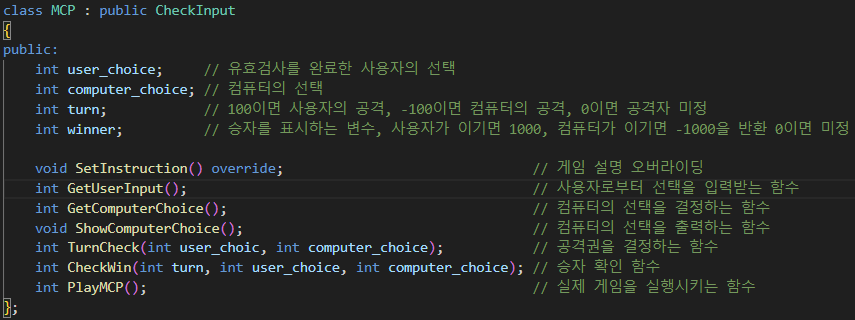
**- 설명 : 게임 구현이라는 추상적인 내용을 보충 설명하기 위한 내용이다.**

**- 특징 1 :** 모든 게임은 헤더파일(.h)과 실행파일(.cpp)로 구성되어 있고 게임 실행은 main에서 menu 객체를 통해 각 게임의 객체를 생성한 후 객체의 게임실행 메소드를 호출하는 방식으로 이뤄진다.

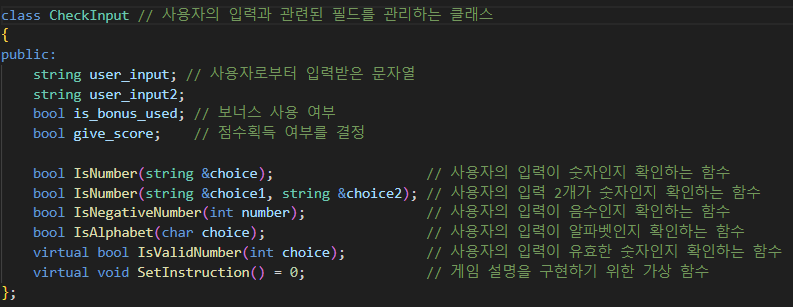
- **특징 2 :** 사용자로부터 입력을 받을 경우 모든 입력은 유효성검사를 거치게 된다. 가령 특정 범위의 숫자(1,2,3)를 입력해야 하는 경우 사용자의 입력한 문자가 숫자인지(IsNumber()), 사용자의 입력이 유효한 숫자인지(IsValidNumber()) 확인한다.

- **특징 3 :** 각 게임에서 반복적으로 사용되는 변수, 함수가 존재한다. 반복되는 필드들을 하나의 클래스에서 정의한 후에 상속을 통해 사용하는 것이 매번 새롭게 필드를 정의하는 것보다 코드를 작성하고 검토할 때 더 유리하기 때문에 위와 같이 구현했다.

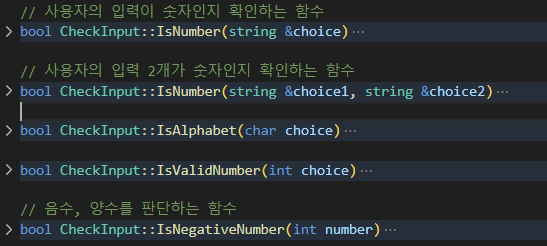
- **특징 4** **:** 각 게임 실행파일(.cpp)에서 보너스 객체를 생성한 후 보너스 사용 여부를 물어보고 사용 여부를 is\_bonus\_used (bool)에 저장 후 보너스 기믹 사용여부를 결정한다.

**- 코드 스크린샷**

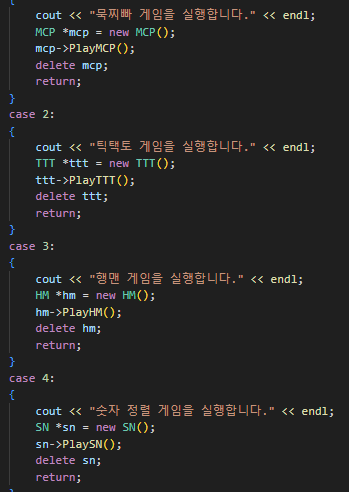
**묵찌빠 헤더파일**

****

**반복되는 필드들을 정의한 부모클래스 CheckInput**

****

**부모클래스에 정의된 유효성 검사 함수**

****

**Menu 객체에서**

**각 게임의 객체를 생성 후**

**객체의 게임 실행 함수를 호출한다.**

**1) 게임 구현하기**

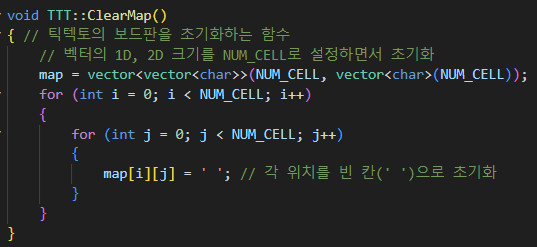
**(1-1) 틱택토 컴퓨터 행동 알고리즘**

**- 입출력 :** 보드판(map), 좌표(x, y), 보드의 칸 수(NUM\_CELL), 플레이어의 돌(current\_player)

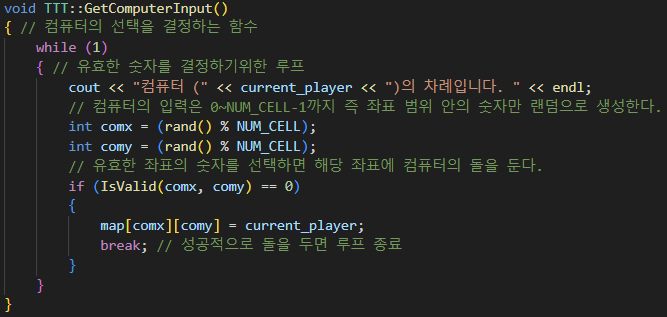
**-** **설명 :** 틱택토 보드판은 vector로 구현, 게임 시작 시 ClearMap()함수를 호출, 보드판의 크기를 설정 후, 각 칸을 빈칸(‘ ‘)로 초기화한다.

컴퓨터의 행동은 GetComputerInput()함수를 통해 결정된다. 컴퓨터는 보드의 좌표 범위 안의 숫자만 랜덤으로 선택하고, 선택한 수를 IsValid()함수를 통해 유효한 좌표를 선택했는지 확인한 후 유효한 좌표면 해당 좌표에 돌을 둔다.

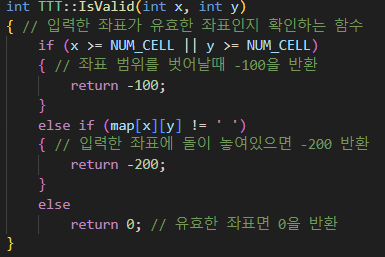
**- 적용된 배운 내용 :** vector, 조건문 & 반복문, 원하는 범위 내의 무작위 수 생성(rand() % NUM\_CELL, 게임 실행 함수에 srand(time(0))를 추가해 무작위 난수를 생성한다.)

**- 코드 스크린샷**

**보드판 초기화 함수**



**컴퓨터의 선택을 결정하는 함수**



**돌을 놓으려는 좌표가 유효한 좌표인지**

**확인하는 함수**

**(1-2) 묵찌빠 컴퓨터 행동 알고리즘**

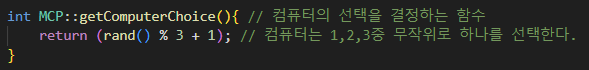
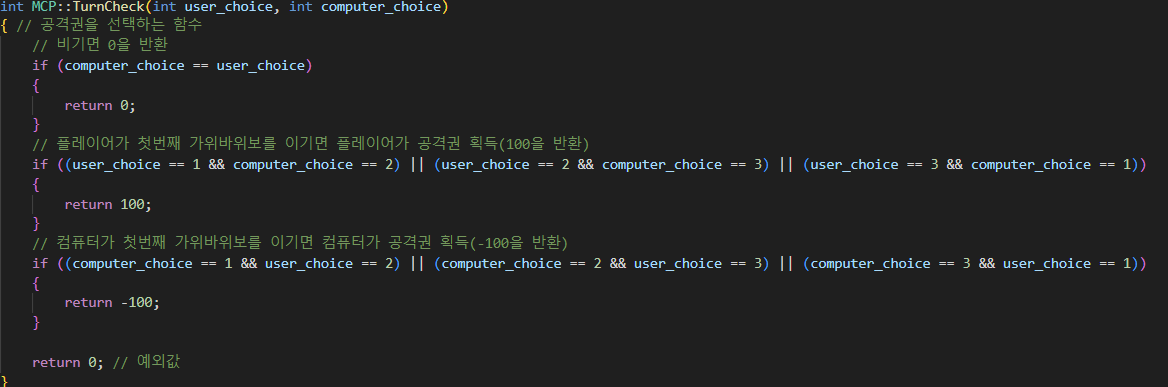
**- 입출력 :** 사용자의 선택(user\_choice), 컴퓨터의 선택(computer\_choice), 공격권(turn)

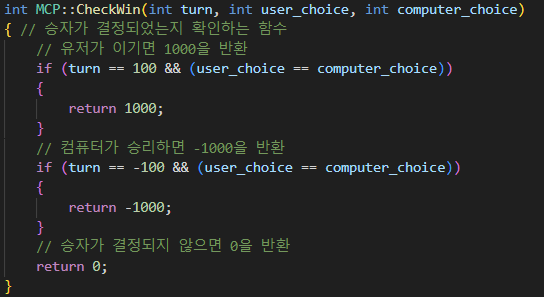
**-** **설명 :** 1은 묵, 2는 찌, 3은 빠를 나타낸다. 컴퓨터는 1~3까지 무작위의 수를 선택한다.

공격권은 첫번째 가위바위보를 이긴 사람이 가지며 이 상태에서 가위바위보를 비기면 승리, 이기면 공격권 유지, 지면 공격권을 상대에게 넘긴다.

공격권이 설정된 상태에서 두번째 가위바위보를 비기면(사용자의 선택과 컴퓨터의 선택이 같으면) 공격권이 있는 플레이어가 승리한다.

**- 코드 스크린샷**

****

****

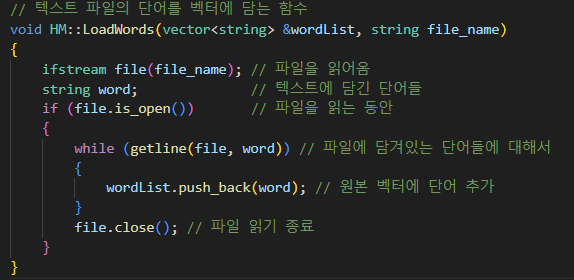
**(2-1)** 단어 불러오기

**- 입출력 :** 단어 파일(.txt), 단어들을 담을 배열 (vector<string> wordlist)

**-** **설명 :** .txt 파일에 적혀 있는 단어들을 배열에 담는 함수이다.

**- 적용된 배운 내용 :** vector push\_back, ifstream

**- 코드 스크린샷**

****

**(2-2)** 단어 맞추기 게임

**- 입출력 :** 정답, 숨겨진 정답(vector), 선택한 주제(int), 기회(in)

**-** **설명 :** 단어 맞추기 게임 코드로 크게 주제&단어선택, 정답 추측, 승리여부 확인으로 나눠진다.

**- 적용된 배운 내용 :** vector push\_back, switch, 포인터, 포인터 연산

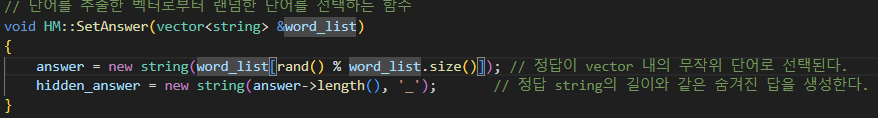
**- 코드 스크린샷**

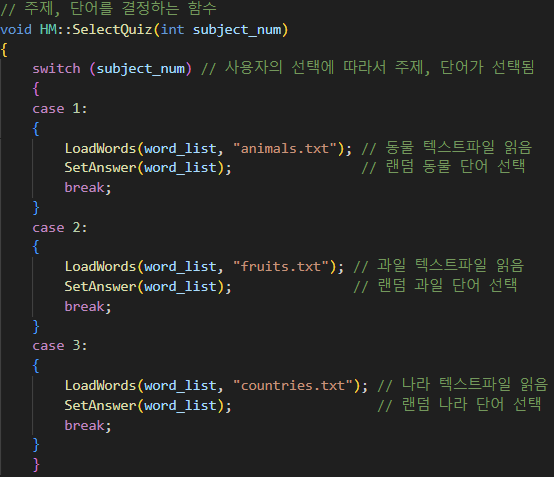
**텍스트 파일의 단어를 벡터에 담은 후 그중 무작위 단어가 정답으로 설정된다.**

**숨겨진 정답(실제로 유저에게 보이는 정답)은 정답의 길이만큼 \_로 설정된다.**

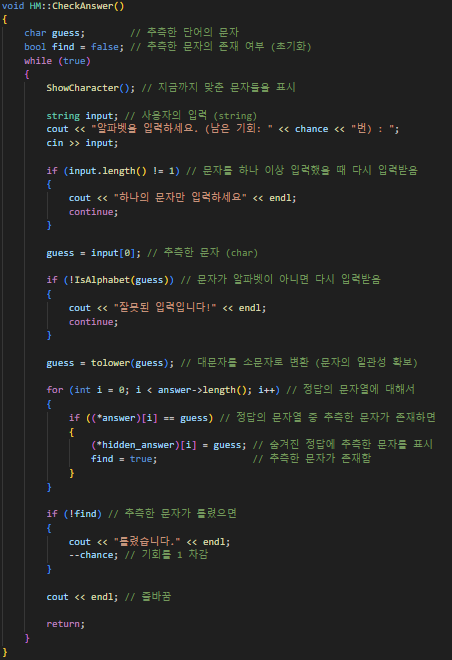
**Ex) word\_list에 rabbit, lion, dog가 들어 있다고 할 때 여기서 rabbit이 무작위 단어로 선택**

**숨겨진 정답은 rabbit의 길이만큼 즉 \_\_\_\_\_\_로 설정된다.**

****

****

**주제, 단어를 선택하는 함수**

****

**정답을 확인하는 함수**

**문자를 확인하는 함수(코드 주석 참조)**

**문자를 입력 받는 코드**

**(1개 이상의 문자를 입력 시 다시 입력)**

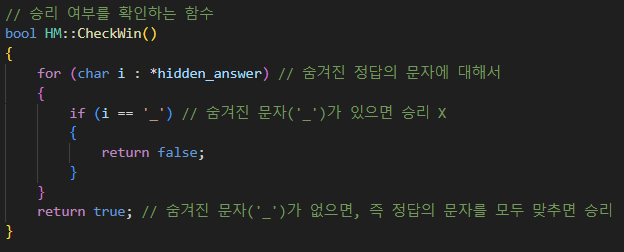
**(알파벳이 아닌 다른 문자를 입력 시**

**다시 입력)**

**추측한 문자가 존재하면 해당 문자에**

**해당하는 부분을 숨겨진 정답에 표시함**

**틀리면 기회를 1 차감함**

****

**승리 여부를 확인하는 함수**

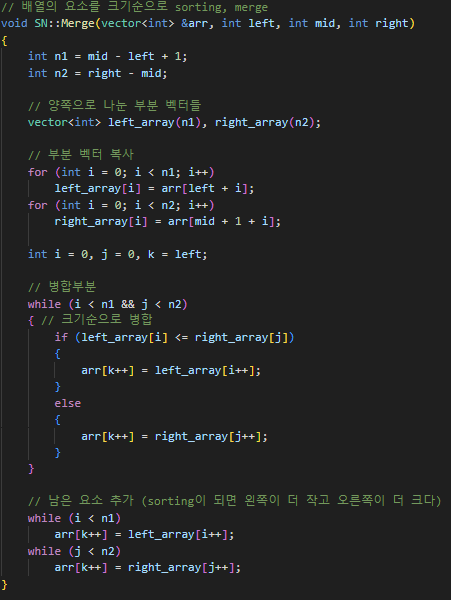
**(3-1)** 숫자 배열 알고리즘

**- 입출력 :** 생성한 무작위 숫자배열(vector<int>), 배열의 부분배열의 길이 (int)

**-** **설명 :** merge sort 알고리즘을 구현한 함수이다.

**- 적용된 배운 내용 :** vector, call by reference

**- 코드 스크린샷**

****

**Merge 부분**

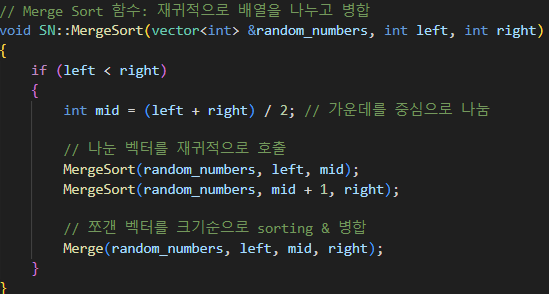
**부분 벡터로 나눔**

**나눈 벡터들의 각 요소를 비교**

**작은 요소를 벡터에 먼저 추가**

**비교 후 부분벡터에 남은 요소는**

**작은 요소별로 추가한다.**

****

**Merge sort 함수**

**입력 받은 배열을 부분 배열로 쪼갠 후 쪼갠 배열에서 sorting후 다시 하나의 배열로 Merge**

**이를 재귀적으로 수행**

**(3-2)** 숫자 배열 게임

**- 입출력 :** 생성한 무작위 숫자배열, 사용자 정답배열 (vector<int>), 문제 배열의 길이 (int)

**-** **설명 :** 숫자 배열 게임 코드로 크게 생성할 숫자 배열의 크기 입력, 무작위 숫자 배열 생성, 사용자로부터 정답 입력 받기, 무작위 숫자배열 MergeSort, 승리 여부 확인으로 나눠져 있다.

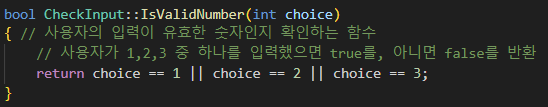
이 중 IsValidNumber 함수를 재정의하여 숫자 배열 크기의 입력을 조정하였다.

**- 적용된 배운 내용 :** virtual 함수, 부모 클래스의 함수 override, stringstream

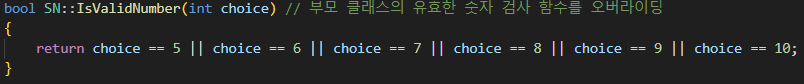
**- 코드 스크린샷**

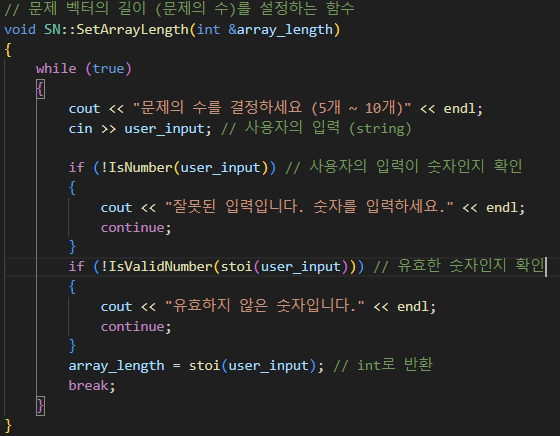
**부모 클래스 CheckInput의 IsValidNumber 함수(virtual)**

****

****

**재정의한 IsValidNumber 함수**

****

****

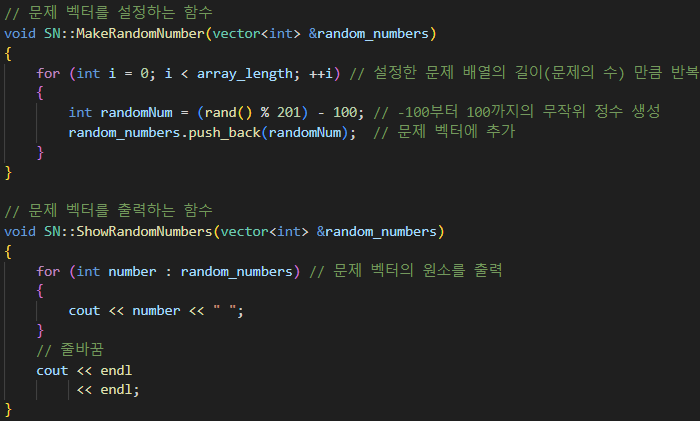
**생성할 문제의 수를 결정하는 함수**

**숫자 유효성 검사는 부모클래스의 함수를 이용함**

**부모클래스의 함수를 재정의하여 유효한 숫자의 범위를 재설정**

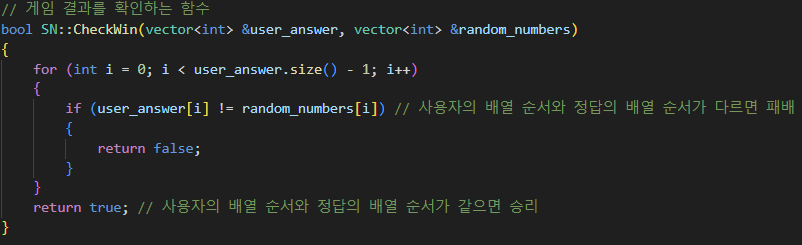
**문제 배열의 길이 설정**

**설정한 문제 배열의 길이만큼 무작위 숫자 생성 후 벡터에 담음**

****

**승리 여부를 확인하는 함수**

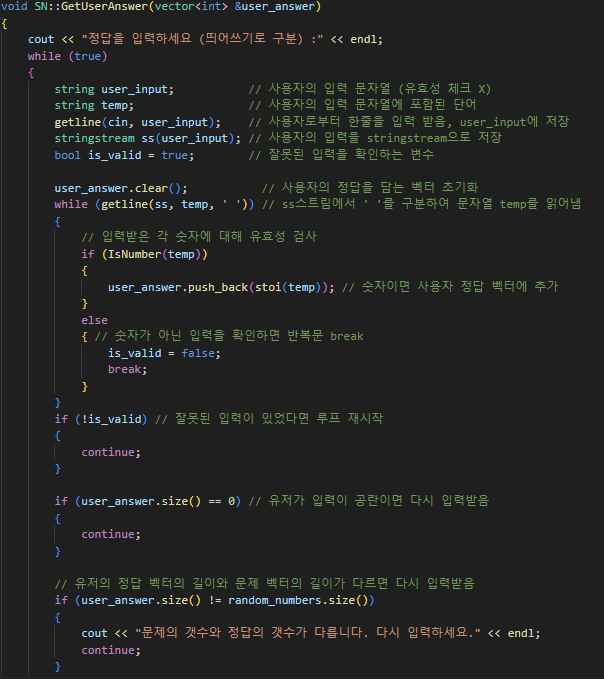
**사용자가 입력한 배열 순서와 정답 배열 순서가 같으면 정답**

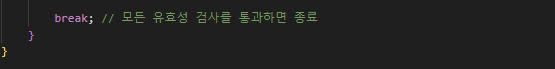
****

**s**

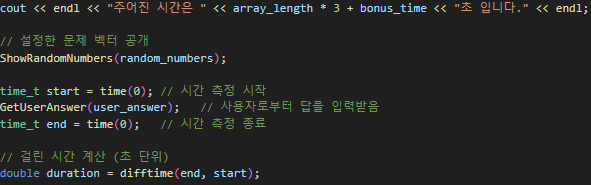
**사용자로부터 정답 배열을 입력 받는 함수 한 줄에 여러 개의 숫자를 구분하기 위해 stringstream 사용, ss의 각 문자열에 대해서 입력이 숫자가 아니면 다시 입력 받음**

**추가로 문제의 수와 입력한 숫자의 수가 다르거나 공란을 입력시에도 다시 입력 받음**

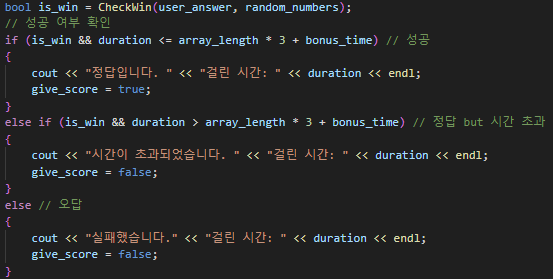
****

****

**게임 실행 함수의 일부로 사용자가 정답을 입력하는 시간을 계산함**

****

**사용자가 정답을 일정 시간을 넘어서 입력 시 자동 실패**

****

**(4-1) 누적 점수 기능**

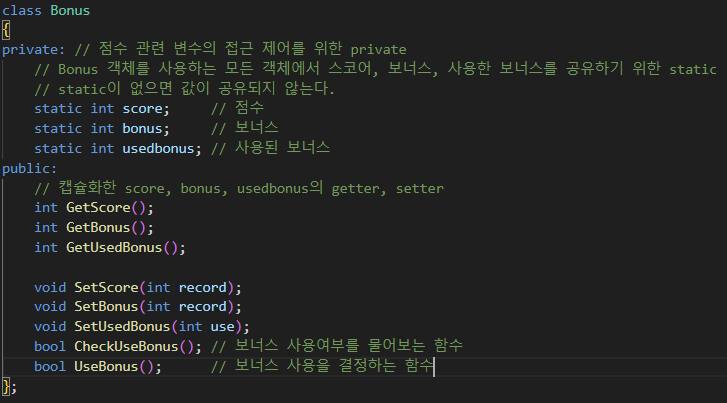
**- 입출력 :** 누적점수(score), 남은 보너스(bonus), 사용된 보너스(usedbonus)

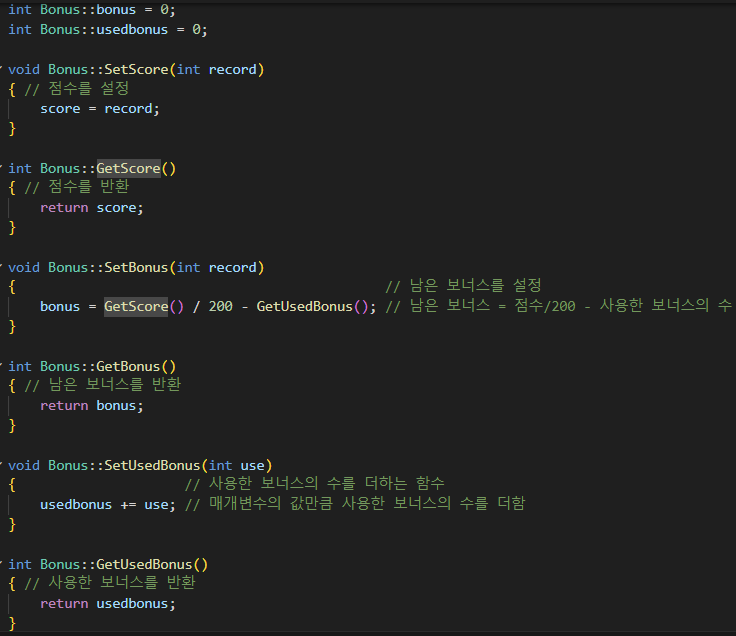
**-** **설명 :** Bonus 객체를 사용하는 모든 객체에서 스코어, 보너스, 사용된 보너스를 공유하기 위해 변수를 static으로 설정했으며, 변수의 값을 실수로라도 바꾸는 상황을 줄이기 위해 접근 제어자를 private로 하고 getter, setter로 접근하도록 구성했다.

**- 적용된 배운 내용 :** 접근제어자, 캡슐화, 정적변수(static)

**- 코드 스크린샷**

**보너스 객체와 보너스 관련 함수의 캡슐화**

****

****

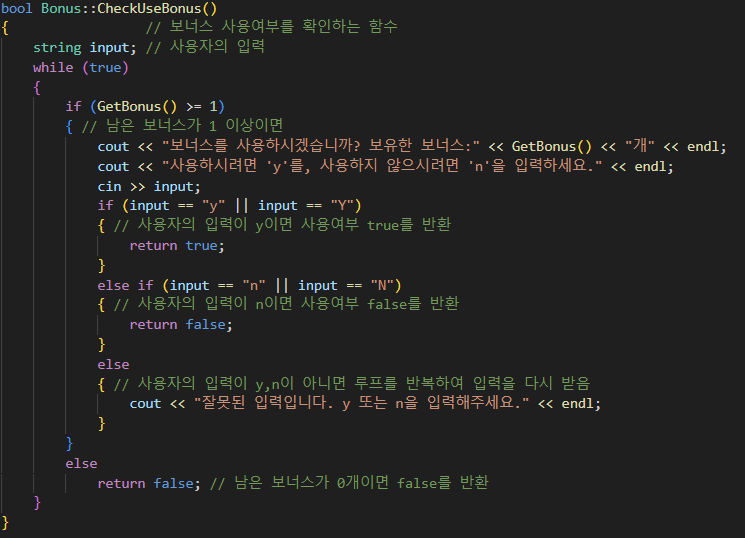
**(4-2) 보너스 사용 기능**

**- 입출력 :** 변수의 getter, setter

**-** **설명 :** 사용자로부터 유효한 입력(y,n 대문자 가능)을 받은 후 사용 여부를 결정한다.

보너스가 사용되면 사용된 보너스를 1 올린 후 남은 보너스를 갱신한다. 그 후 각 게임마다 보너스 기믹이 발동된다.

**- 코드 스크린샷**

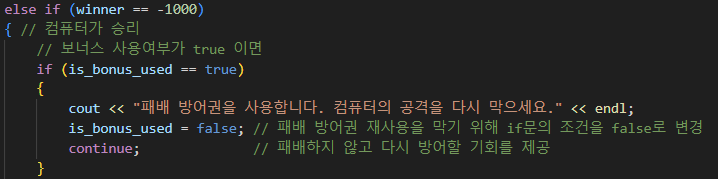
****

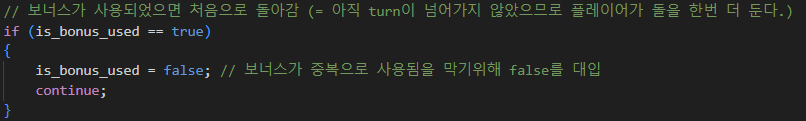
**묵찌빠, 틱택톡 객체에서 보너스 객체 생성 & UseBonus 함수 호출**

****

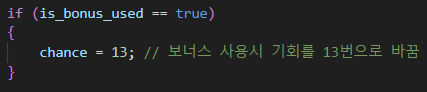
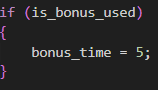
****

**묵찌빠, 틱택토의 보너스 기믹 (패배 방어권, 돌 한번 더 두기)**

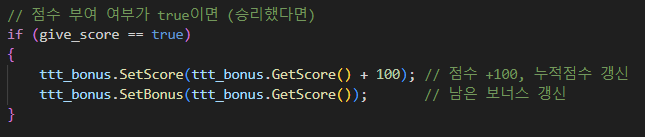
****

****

**단어 맞추기, 숫자 순서 맞추기 보너스 기믹 (기회 추가, 시간 추가)**

****

**게임에서 승리시 점수 획득 & 남은 보너스 갱신**

****

**2) 메뉴 구현하기**

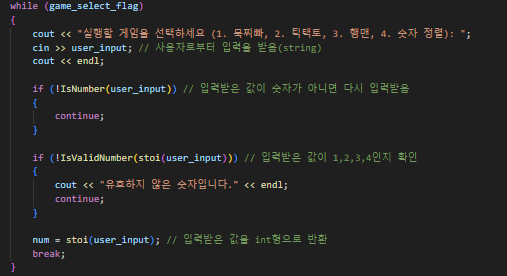
**(1) 플레이할 게임 선택하기**

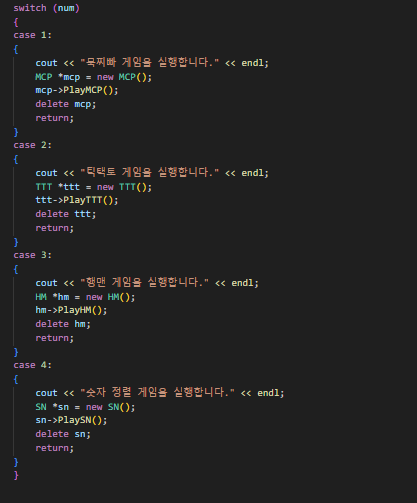
**- 입출력 :** 사용자의 입력(user\_input), 유효성 검사가 끝난 사용자의 입력(num), 각 게임의 객체

**-** **설명 :** 사용자의 입력에 따라 플레이할 게임을 결정한 후, 결정한 게임의 동적 객체를 생성한 후 생성한 객체의 PlayGame() 함수를 호출한다. 게임이 끝난 후에는 생성한 객체를 delete 한다.

**- 적용된 배운 내용 :** 동적 객체 생성, 호출

**- 코드 스크린샷**

****

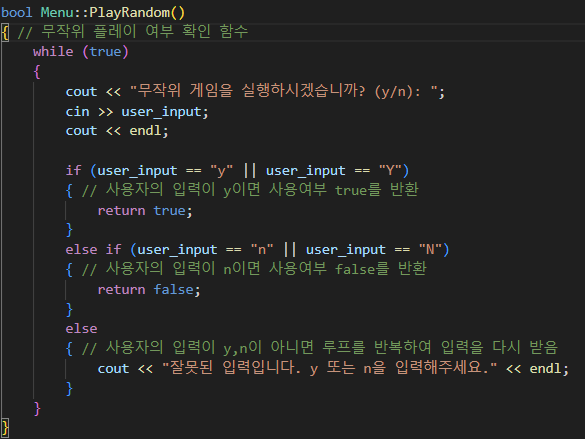
****

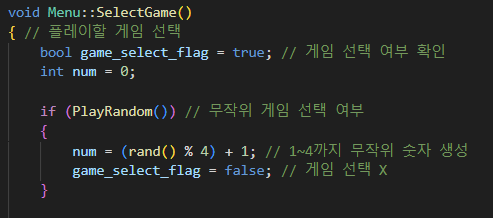
**(2) 랜덤 플레이 기능**

**- 입출력 :** 사용자의 입력(user\_input)

**-** **설명 :** 플레이할 게임을 무작위로 선택한다.

**- 코드 스크린샷**

****

****

**무작위 선택 시 SelectGame의 switch에 무작위 int값이 들어간다.**

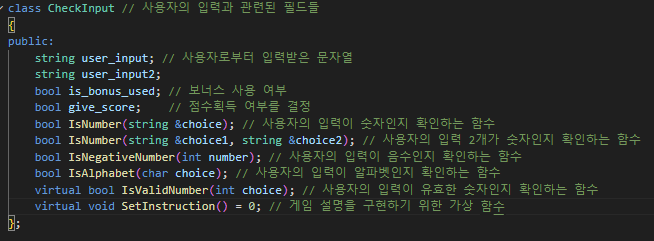
**(3-1) 게임 설명 기능**

**- 입출력 :** 사용자의 입력(user\_input)

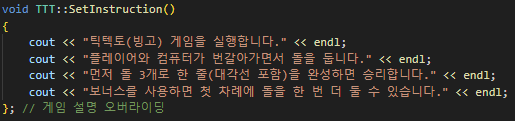
**-** **설명 :** 플레이할 게임을 무작위로 선택한다.

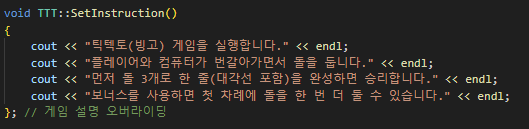
**- 적용된 배운 내용 :** 다형성, 가상 함수

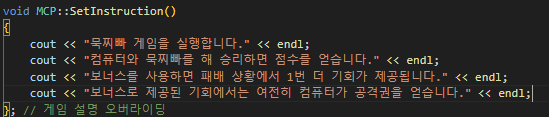
**- 코드 스크린샷**

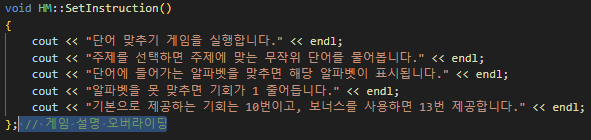
****

**게임 설명을 위한 가상함수**

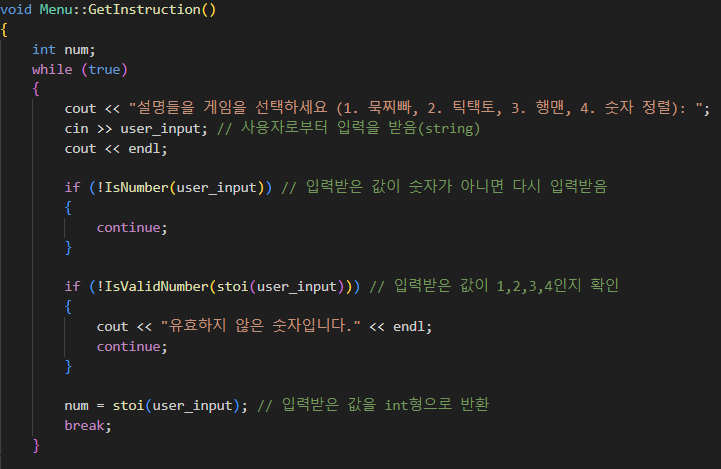
**CheckInput을 상속받은 각 게임에서 SetInstruction()을 재정의의**

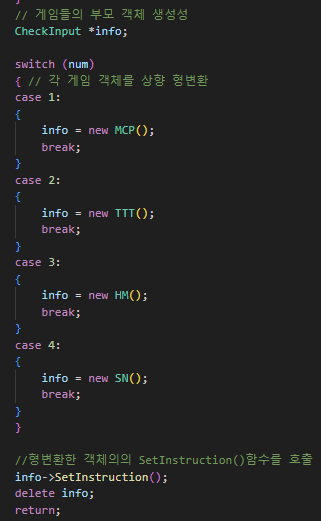
****

****

****

**각 게임을 CheckInput으로 상향 형변환 -> 형변환한 객체의 설명함수를 호출**

****

**(3-2) 점수 확인 기능**

**게임 객체의 부모 CheckInput 객체 생성**

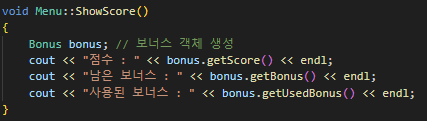
**Switch 문을 통해 해당하는 게임 객체를 부모 객체로 상향 형변환**

**형변환한 객체의 게임 설명 함수를 호출**

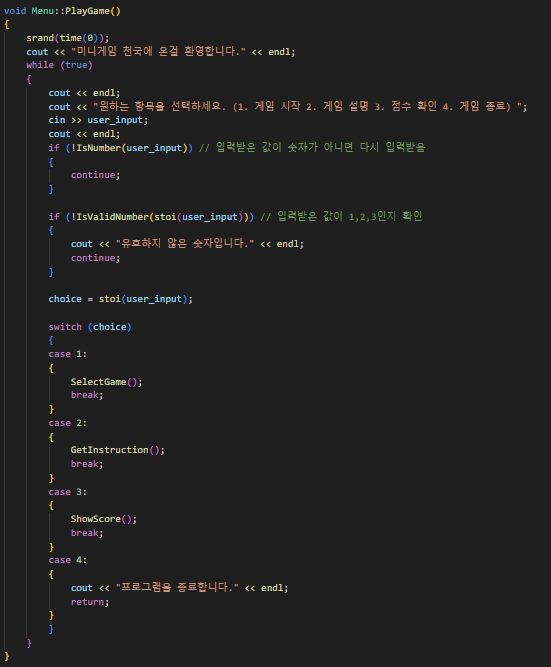
**생성한 객체 반환 (delete)**

**-** **설명 :** 남은 점수, 사용한 보너스, 남은 보너스를 표시한다.

**- 코드 스크린샷**

****

**메뉴 객체의 모든 기능을 담은 실행 함수 PlayGame()**

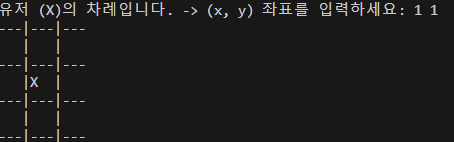
****

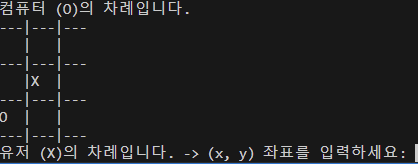
**4 테스트 결과**

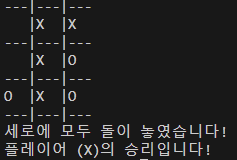
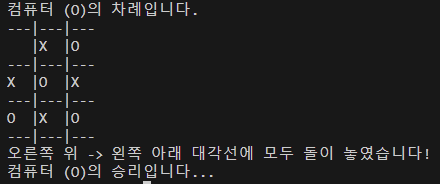
**(1-1) 틱택토 컴퓨터 행동 알고리즘**

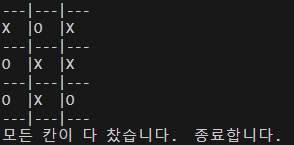
- 설명 : 유저가 돌을 둔 후 컴퓨터도 랜덤으로 돌을 둔다.

- 테스트 결과 스크린샷

****

****

****

****

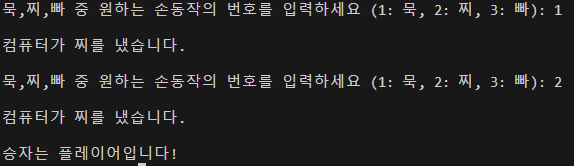
**틱택토의 3가지 상황 승리, 패배, 무승부**

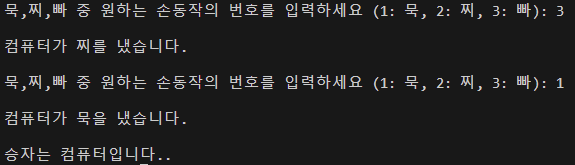
**(1-2) 묵찌빠 컴퓨터 행동 알고리즘**

- 설명 : 컴퓨터가 무작위로 가위바위보를 한다.

- 테스트 결과 스크린샷

**묵찌빠의 2가지 상황 승리, 패배**

****

****

**(2-1), (2-2) 단어 맞추기 게임& 단어 불러오기**

- 설명 : 행맨 게임을 실행한다.

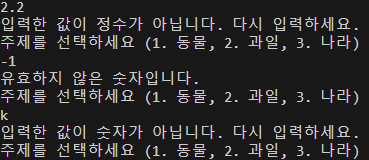
- 테스트 결과 스크린샷

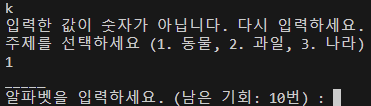
**유효성 검사**

**정수가 아닐 경우**

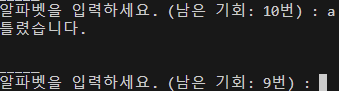
**유효한 숫자가 아닐 경우**

**숫자가 아닐 경우**

****

****

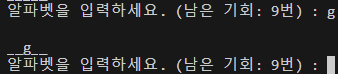
**알파벳을 입력 받음**

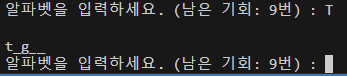
****

**틀리면 기회 1 차감**

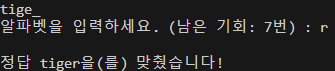
**맞추면 알파벳 표시**

**기회 차감 X**

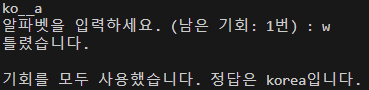
****

****

**대문자 입력도 가능**

****

**기회 안에 정답을 맞출 경우**

****

**기회 안에 정답을 못 맞출 경우**

**(3-1), (3-2) 숫자 배열 알고리즘 & 숫자 배열 게임**

- 설명 : 숫자 배열 게임을 실행한다.

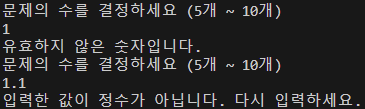
- 테스트 결과 스크린샷

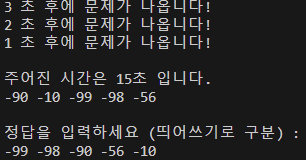
**유효성 검사**

**유효한 숫자가 아닐 경우**

**정수가 아닐 경우**

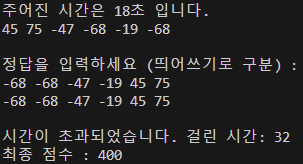
**숫자가 아닐 경우**

****

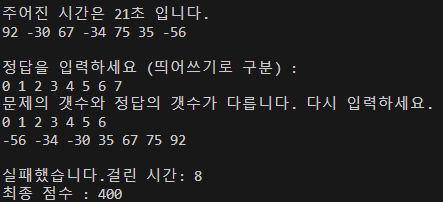
****

**5문제 제한시간 안에 맞춤**

****

****

**6문제 제한시간 안에 못 맞춤**

****

**7 문제**

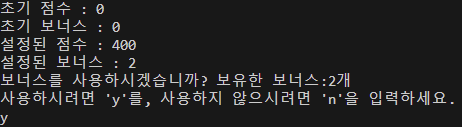
**문제 수와 다르게 입력**

**제한시간 안에 정답 틀림**

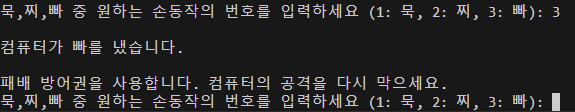
**(4-1), (4-2) 누적 점수기능 & 보너스 사용 기능**

- 설명 : 승리 시 점수 획득, 패배 또는 무승부 시 점수 획득 불가, 보너스 사용

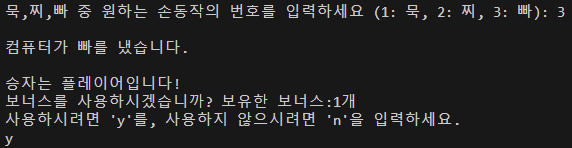
- 테스트 결과 스크린샷

****

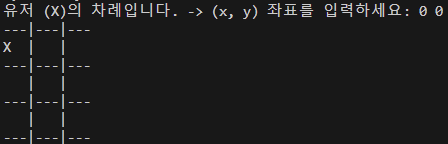
**보너스 사용 여부 확인 (y)**

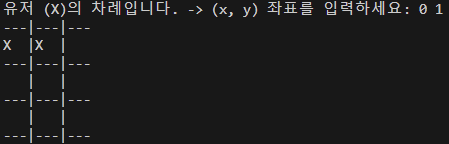
****

**묵찌빠 보너스 기믹 패배 방어권**

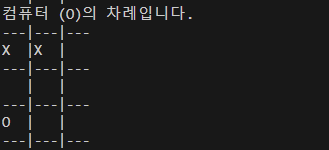
****

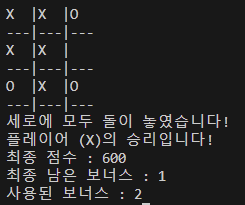
**보너스 사용 여부 확인 (y)**





**틱택토 보너스 기믹 추가 행동권**



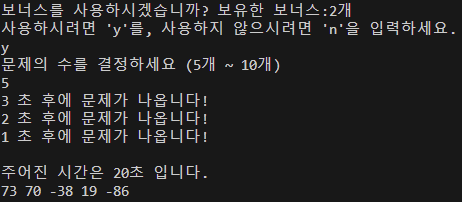


**승리시 +100**

**2승 했으므로 남은 점수는 400+200**

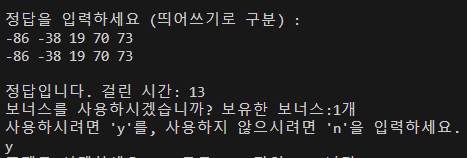
**200점당 보너스 = 남은 보너스 1개**

**사용된 보너스 = 2개**



**숫자 배열하기 보너스 사용**

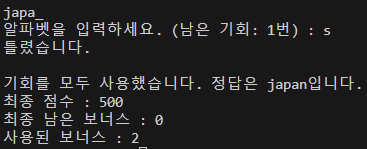
**기존 시간 3\*문제 수에서 보너스 사용으로 +5초 즉 20초로 제한시간 증가**



**정답 맞춘 후 바로 행맨 게임 실행**

**보너스 사용으로 기회 +3 기존(10번)**





**행맨 게임 실패, 점수 획득 X**

**기존 점수 400 + 숫자 배열 성공 100**

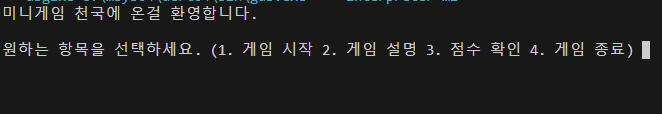
**최종 점수 500, 보너스 2번 사용**

**남은 보너스는 0번**

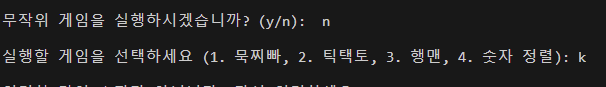
**2) 메뉴 구현하기 테스트 결과**

**(1) 플레이할 게임 선택하기**

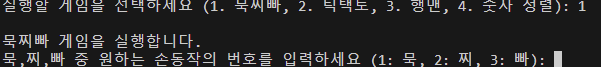
**맨 처음 실행 화면**

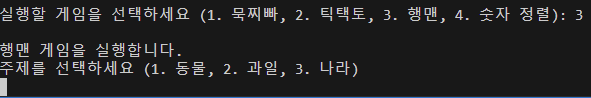
****

**무작위 게임 실행여부**

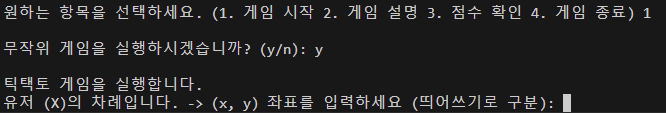


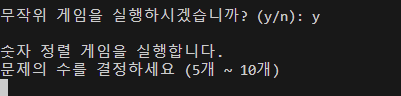
**유저의 입력에 따라 게임이 선택됨**





**(2) 랜덤 플레이 기능**

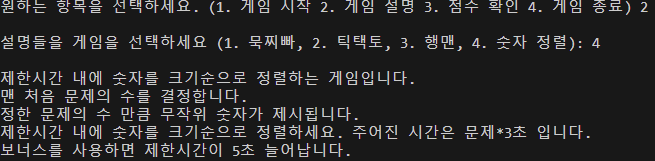
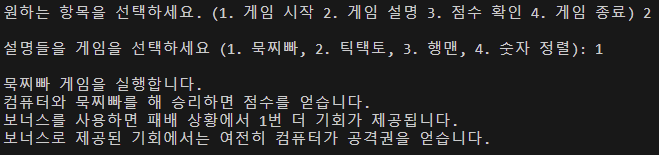


****

**무작위 게임 실행됨**

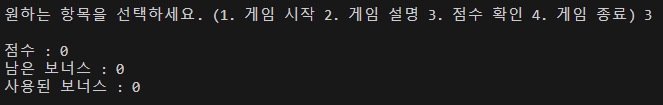
**(3-1) 게임 설명 기능**

**각 게임의 설명이 출력됨**

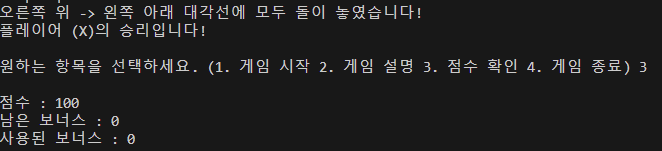
****

**(3-2) 누적점수 확인 기능**

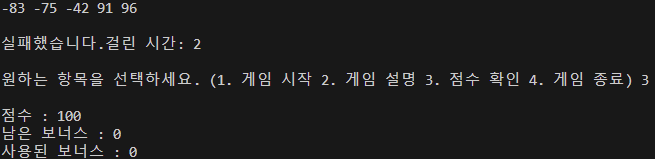
**맨 처음에는 점수, 보너스 0**

****

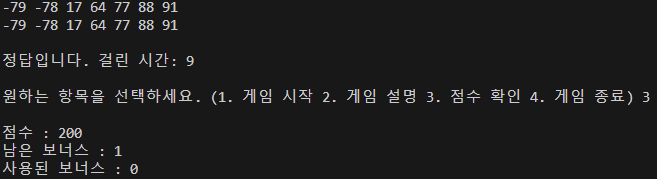
**틱택토 승리 후 100점 추가**

****

**게임 패배 후 점수 추가 X**

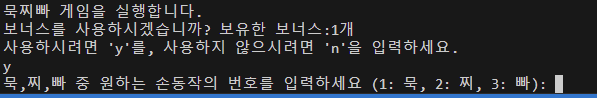
****

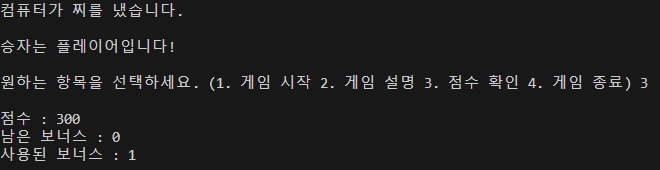
**점수 200점 당 보너스 1개 생성**

****

**보너스 1개 사용 후 게임 승리**

**점수 +100, 남은 보너스 -1, 사용한 보너스 +1**

****

****

**5. 느낀점**

이번 프로젝트를 진행하면서 가장 중요하게 생각했던 점은 **유효성 검사**였습니다. 제가 생각하기에 좋은 프로그램이란, 사용자가 실수를 하더라도 프로그램이 이를 보완하고 사용자가 원하는 옳은 결과로 자연스럽게 이끌어주는 프로그램이라고 생각했기 때문입니다. 따라서 다양한 예외 상황을 최소화하고 프로그램의 안정성을 높이는 데 주력했습니다.

비록 이번 프로젝트는 복잡한 프로그램은 아니었지만, 작은 게임 객체들을 하나로 모아 '미니게임 천국'이라는 더 큰 하나의 결과물을 만든다는 점에서 큰 의미를 느꼈습니다. 이를 통해 프로젝트란 **단계별로 작은 부품을 모아 더 큰 하나의 결과물을 만들어가는 과정**이라는 사실을 깨닫게 되었습니다.

학기 동안 배운 내용도 이와 비슷했습니다. 처음에는 단순한 틱택토(Tic-Tac-Toe)나 Mud 게임처럼 간단한 프로그램으로 시작했지만, 매 수업마다 배운 내용을 기반으로 기존 프로그램에 기능을 조금씩 추가하며 점점 더 완성도 있는 프로그램으로 발전시킬 수 있었습니다. 이러한 학습 방식은 새로운 내용을 즉시 활용하여 실질적인 개선을 이루는 경험을 제공했으며, 이를 통해 **자신의 프로그래밍 능력이 점진적으로 향상되고 있다는 것을 직접 체감할 수 있었습니다.** 이러한 점이 수업의 가장 큰 매력이라고 느꼈습니다.

결과적으로, 이번 프로젝트는 작은 부분을 모아 더 큰 결과물을 만들어가는 경험과 유효성 검사라는 소프트웨어 개발의 중요한 원칙을 다시 한 번 체감할 수 있는 좋은 기회였습니다. 이 경험은 앞으로 더 큰 프로젝트를 진행할 때에도 소중한 밑거름이 될 것이라고 믿습니다.