

C++ 프로그래밍 및 실습

C++ 미니게임

진척 보고서 #2

모든 기능은 main(임시)에서 확인할 수 있습니다!

제출일자: 2024-11-29

제출자명: 김휘승

제출자학번: 214958

1. 프로젝트 목표 (16 pt)

1) 배경 및 필요성 (14 pt)

수업시간에 terminal 환경에서 수행하는 기초적인 게임들을 배워봤습니다. 흥미로운 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 작은 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 나만의 '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 통해 게임에 신선함을 더했습니다.

2. 기능 계획

1) 기능 1 : 게임 구현하기

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

(1) 세부 기능 1: 목찌빠 & 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 게임에서 보너스를 제공합니다. 목찌빠는 패배 방어 1회, 틱택토는 추가 행동 1회, 숫자 배열하기는 추가 시간 5초, 단어 맞추기는 질문권을 3회 더 제공합니다.

2) 기능 2 : 메뉴 구현하기

- 설명: 메뉴에서 플레이할 게임을 선택합니다.

(1) 세부 기능 1: 랜덤 플레이

- 설명: 플레이할 게임이 랜덤으로 결정합니다.

(2) 세부 기능 2: 설명 & 점수 표시

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

3. 진척사항

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 메뉴 화면(현재 : main(임시))에서 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

```
class MCP : public CheckInput
{
public:
    int user_choice;
    int computer_choice;
    int turn;
    int winner;
    int GetUserInput();
    int GetComputerChoice();
    void ShowComputerChoice();
    int TurnCheck(int user_choic, int computer_choice);
    int CheckWin(int turn, int user_choice, int computer_choice);
    int PlayMCP();
};
```

목찌빠 헤더파일

// 유효검사를 완료한 사용자의 선택
// 컴퓨터의 선택
// 100이면 사용자의 공격, -100이면 컴퓨터의 공격, 0이면 공격자 미정
// 승자를 표시하는 변수, 사용자가 이기면 1000, 컴퓨터가 이기면 -1000을 반환 0이면 미정
// 사용자로부터 선택을 입력받는 함수
// 컴퓨터의 선택을 결정하는 함수
// 컴퓨터의 선택을 출력하는 함수
// 공격권을 결정하는 함수
// 승자 확인 함수
// 실제 게임을 실행시키는 함수

```
#include "Mukchippa.h"
#include "Bonus.h"
#include "tictactoe.h"
#include "Hangman.h"

// 게임을 확인하는 메인(임시)
int main(){
    MCP mcpGame; // 목찌빠 객체
    Bonus gameBonus; // 보너스 객체
    TTT tictactoeGame; // 틱택토 객체

    mcpGame.playMCP();
    tictactoeGame.playTTT();
}
```

메인 함수

각 게임의 객체를 생성 후
객체의 게임 실행 함수를 호출한다.

```
class CheckInput // 사용자의 입력과 관련된 필드들
{
public:
    string user_input; // 사용자로부터 입력받은 문자열
    string user_input2;
    bool is_bonus_used; // 보너스 사용 여부
    bool give_score; // 점수획득 여부를 결정
    bool IsNumber(string &choice); // 사용자의 입력이 숫자인지 확인하는 함수
    bool IsNumber(string &choice1, string &choice2); // 사용자의 입력 2개가 숫자인지 확인하는 함수
    bool IsAlphabet(char choice); // 사용자의 입력이 알파벳인지 확인하는 함수
    virtual bool IsValidNumber(int choice); // 사용자의 입력이 유효한 숫자인지 확인하는 함수
};
```

반복되는 필드들을 정의한
부모클래스

부모클래스에 정의된 숫자 유효성 검사 함수

```
// 사용자의 입력이 숫자인지 확인하는 함수
bool CheckInput::IsNumber(string &choice)
{ // 정수인지 확인 (소숫점 존재 여부 확인)
    if (choice.find('.') != string::npos)
    {
        cout << "입력한 값이 정수가 아닙니다. 다시 입력하세요." << endl;
        return false;
    }
    try
    { // stoi 예외처리를 위한 try-catch
        // 사용자의 String 입력을 int로 캐스팅한 값을 저장
        int int_choice1 = std::stoi(choice);
    }
    catch (const std::invalid_argument &e)
    { // 숫자가 아닌 값을 입력했을 때
        cout << "입력한 값이 숫자가 아닙니다. 다시 입력하세요." << endl;
        return false; // false를 반환
    }
    catch (const std::out_of_range &e)
    { // 너무 큰 숫자를 입력했을 때
        cout << "입력한 숫자가 너무 큼니다. 다시 입력하세요." << endl;
        return false; // false를 반환
    }
    return true; // 문자가 전부 숫자면 true를 반환
}
```

```
// 사용자의 입력 2개가 숫자인지 확인하는 함수
bool CheckInput::IsNumber(string &choice1, string &choice2)
{ // IsNumber를 오버로딩
    // 정수인지 확인 (소숫점 존재 여부 확인)
    if (choice1.find('.') != string::npos || choice2.find('.') != string::npos)
    {
        cout << "입력한 값이 정수가 아닙니다. 다시 입력하세요." << endl;
        return false;
    }
    try
    { // stoi 예외처리를 위한 try-catch
        // 사용자의 String 입력을 int로 캐스팅한 값을 저장
        int int_choice1 = std::stoi(choice1);
        int int_choice2 = std::stoi(choice2);
    }
    catch (const std::invalid_argument &e)
    { // 숫자가 아닌 값을 입력했을 때
        cout << "입력한 값이 숫자가 아닙니다. 다시 입력하세요." << endl;
        return false; // false를 반환
    }
    catch (const std::out_of_range &e)
    { // 너무 큰 숫자를 입력했을 때
        cout << "입력한 숫자가 너무 큼니다. 다시 입력하세요." << endl;
        return false; // false를 반환
    }
    return true; // 유효한 숫자면 true를 반환
}
```

보너스 객체 생성 & 보너스 사용 여부 확인

```
int MCP::playMCP(){
    srand(time(0)); // 컴퓨터의 무작위 선택을 위한 난수 생성
    Bonus mcpbonus; // 보너스 시스템을 위한 보너스 객체
    isBonusUsed = mcpbonus.checkUseBonus(); // 보너스 사용여부를 확인

    // 보너스를 사용여부가 true이면 보너스를 사용하는 함수를 호출
    if(isBonusUsed == true){
        |         mcpbonus.useBonus();
    }
}
```

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

```
void TTT::ClearMap()
{ // 틱택토의 보드판을 초기화하는 함수
  // 벡터의 1D, 2D 크기를 NUM_CELL로 설정하면서 초기화
  map = vector<vector<char>>>(NUM_CELL, vector<char>(NUM_CELL));
  for (int i = 0; i < NUM_CELL; i++)
  {
    for (int j = 0; j < NUM_CELL; j++)
    {
      map[i][j] = ' '; // 각 위치를 빈 칸(' ')으로 초기화
    }
  }
}

void TTT::GetComputerInput()
{ // 컴퓨터의 선택을 결정하는 함수
  while (1)
  { // 유효한 숫자를 결정하기위한 루프
    cout << "컴퓨터 (" << current_player << ")의 차례입니다. " << endl;
    // 컴퓨터의 입력은 0~NUM_CELL-1까지 즉 좌표 범위 안의 숫자만 랜덤으로 생성한다.
    int comx = (rand() % NUM_CELL);
    int comy = (rand() % NUM_CELL);
    // 유효한 좌표의 숫자를 선택하면 해당 좌표에 컴퓨터의 돌을 둔다.
    if (IsValid(comx, comy) == 0)
    {
      map[comx][comy] = current_player;
      break; // 성공적으로 돌을 두면 루프 종료
    }
  }
}

int TTT::IsValid(int x, int y)
{ // 입력한 좌표가 유효한 좌표인지 확인하는 함수
  if (x >= NUM_CELL || y >= NUM_CELL)
  { // 좌표 범위를 벗어날때 -100을 반환
    return -100;
  }
  else if (map[x][y] != ' ')
  { // 입력한 좌표에 돌이 놓여있으면 -200 반환
    return -200;
  }
  else
    return 0; // 유효한 좌표면 0을 반환
}
```

보드판 초기화 함수

컴퓨터의 선택을 결정하는 함수

돌을 놓으려는 좌표가 유효한 좌표인지
확인하는 함수

(1-2) 목찌빠 컴퓨터 행동 알고리즘

- **입출력** : 사용자의 선택(user_choice), 컴퓨터의 선택(computer_choice), 공격권(turn)

- **설명** : 1은 목, 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.

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

- **코드 스크린샷**

```
int MCP::getComputerChoice(){ // 컴퓨터의 선택을 결정하는 함수
    return (rand() % 3 + 1); // 컴퓨터는 1,2,3중 무작위로 하나를 선택한다.
}
```

```
int MCP::TurnCheck(int user_choice, int computer_choice)
{ // 공격권을 선택하는 함수
  // 비기면 0을 반환
  if (computer_choice == user_choice)
  {
    return 0;
  }
  // 플레이어가 첫번째 가위바위보를 이기면 플레이어가 공격권 획득(100을 반환)
  if ((user_choice == 1 && computer_choice == 2) || (user_choice == 2 && computer_choice == 3) || (user_choice == 3 && computer_choice == 1))
  {
    return 100;
  }
  // 컴퓨터가 첫번째 가위바위보를 이기면 컴퓨터가 공격권 획득(-100을 반환)
  if ((computer_choice == 1 && user_choice == 2) || (computer_choice == 2 && user_choice == 3) || (computer_choice == 3 && user_choice == 1))
  {
    return -100;
  }
  return 0; // 예외값
}
```

```
int MCP::CheckWin(int turn, int user_choice, int computer_choice)
{ // 승자가 결정되었는지 확인하는 함수
  // 유저가 이기면 1000을 반환
  if (turn == 100 && (user_choice == computer_choice))
  {
    return 1000;
  }
  // 컴퓨터가 승리하면 -1000을 반환
  if (turn == -100 && (user_choice == computer_choice))
  {
    return -1000;
  }
  // 승자가 결정되지 않으면 0을 반환
  return 0;
}
```

(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
- **코드 스크린샷**

```
// 텍스트 파일의 단어를 벡터에 담는 함수
void HM::LoadWords(vector<string> &wordList, string file_name)
{
    ifstream file(file_name); // 파일을 읽어옴
    string word;               // 텍스트에 담긴 단어들
    if (file.is_open())        // 파일을 읽는 동안
    {
        while (getline(file, word)) // 파일에 담겨있는 단어들에 대해서
        {
            wordList.push_back(word); // 원본 벡터에 단어 추가
        }
        file.close(); // 파일 읽기 종료
    }
}
```

(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 나뉜다.
- **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.

```
// 단어를 주어진 벡터로부터 랜덤한 단어를 선택하는 함수
void HM::SetAnswer(vector<string> &word_list)
{
    answer = new string(word_list[rand() % word_list.size()]); // 정답이 vector 내의 무작위 단어로 선택된다.
    hidden_answer = new string(answer->length(), '_');           // 정답 string의 길이와 같은 숨겨진 답을 생성한다.
}
```



```
// 주제, 단어를 결정하는 함수
void HM::SelectQuiz(int subject_num)
{
    switch (subject_num) // 사용자의 선택에 따라서 주제, 단어가 선택됨
    {
        case 1:
        {
            LoadWords(word_list, "animals.txt"); // 동물 텍스트파일 읽음
            SetAnswer(word_list);                // 랜덤 동물 단어 선택
            break;
        }
        case 2:
        {
            LoadWords(word_list, "fruits.txt"); // 과일 텍스트파일 읽음
            SetAnswer(word_list);                // 랜덤 과일 단어 선택
            break;
        }
        case 3:
        {
            LoadWords(word_list, "countries.txt"); // 나라 텍스트파일 읽음
            SetAnswer(word_list);                // 랜덤 나라 단어 선택
            break;
        }
    }
}
```

주제, 단어를 선택하는 함수

```
void HM::CheckAnswer()
{
    char guess; // 추측한 단어의 문자
    bool find = false; // 추측한 문자의 존재 여부 (초기화)
    while (true)
    {
        ShowCharacter(); // 지금까지 맞춘 문자들을 표시

        string input; // 사용자의 입력 (string)
        cout << "알파벳을 입력하세요. (남은 기회: " << chance << "번) : ";
        cin >> input;

        if (input.length() != 1) // 문자를 하나 이상 입력했을 때 다시 입력받음
        {
            cout << "하나의 문자만 입력하세요" << endl;
            continue;
        }

        guess = input[0]; // 추측한 문자 (char)

        if (!IsAlphabet(guess)) // 문자가 알파벳이 아니면 다시 입력받음
        {
            cout << "잘못된 입력입니다!" << endl;
            continue;
        }

        guess = tolower(guess); // 대문자를 소문자로 변환 (문자의 일관성 확보)

        for (int i = 0; i < answer->length(); i++) // 정답의 문자열에 대해서
        {
            if ((*answer)[i] == guess) // 정답의 문자열 중 추측한 문자가 존재하면
            {
                (*hidden_answer)[i] = guess; // 숨겨진 정답에 추측한 문자를 표시
                find = true; // 추측한 문자가 존재함
            }
        }

        if (!find) // 추측한 문자가 틀렸으면
        {
            cout << "틀렸습니다." << endl;
            --chance; // 기회를 1 차감함
        }

        cout << endl; // 줄바꿈

        return;
    }
}
```

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
```

```
bool HM::CheckWin()
```

```
{
```

```
    for (char i : *hidden_answer) // 숨겨진 정답의 문자에 대해서
```

```
    {
```

```
        if (i == '_') // 숨겨진 문자('_')가 있으면 승리 x
```

```
        {
```

```
            return false;
```

```
        }
```

```
    }
```

```
    return true; // 숨겨진 문자('_')가 없으면, 즉 정답의 문자를 모두 맞추면 승리
```

```
}
```

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**

```
// 배열의 요소를 크기순으로 sorting, merge
void SN::Merge(vector<int> &arr, int left, int mid, int right)
{
    int n1 = mid - left + 1;
    int n2 = right - mid;

    // 양쪽으로 나눈 부분 벡터들
    vector<int> left_array(n1), right_array(n2);

    // 부분 벡터 복사
    for (int i = 0; i < n1; i++)
        left_array[i] = arr[left + i];
    for (int i = 0; i < n2; i++)
        right_array[i] = arr[mid + 1 + i];

    int i = 0, j = 0, k = left;

    // 병합부분
    while (i < n1 && j < n2)
    { // 크기순으로 병합
        if (left_array[i] <= right_array[j])
        {
            arr[k++] = left_array[i++];
        }
        else
        {
            arr[k++] = right_array[j++];
        }
    }

    // 남은 요소 추가 (sorting이 되면 왼쪽이 더 작고 오른쪽이 더 크다)
    while (i < n1)
        arr[k++] = left_array[i++];
    while (j < n2)
        arr[k++] = right_array[j++];
}
```

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: 재귀적으로 배열을 나누고 병합
void SN::MergeSort(vector<int> &random_numbers, int left, int right)
{
    if (left < right)
    {
        int mid = (left + right) / 2; // 가운데를 중심으로 나눔

        // 나눈 벡터를 재귀적으로 호출
        MergeSort(random_numbers, left, mid);
        MergeSort(random_numbers, mid + 1, right);

        // 쪼갠 벡터를 크기순으로 sorting & 병합
        Merge(random_numbers, left, mid, right);
    }
}
```

Merge sort 함수

입력 받은 배열을 부분 배열로
쪼갠 후 쪼갠 배열에서 sorting 후
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 나뉘어 있다.

이 중 IsValidNumber 함수를 재정의하여 숫자 배열 크기의 입력을 조정하였다.

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

- **코드 스크린샷**

부모 클래스의 IsValidNumber 함수

```
virtual bool IsValidNumber(int choice); // 사용자의 입력이 유효한 숫자인지 확인하는 함수
```

```
bool CheckInput::IsValidNumber(int choice)
{ // 사용자의 입력이 유효한 숫자인지 확인하는 함수
    // 사용자가 1,2,3 중 하나를 입력했으면 true를, 아니면 false를 반환
    return choice == 1 || choice == 2 || choice == 3;
}
```

재정의한 IsValidNumber 함수

```
bool SN::IsValidNumber(int choice) // 부모 클래스의 유효한 숫자 검사 함수를 오버라이딩
{
    return choice == 5 || choice == 6 || choice == 7 || choice == 8 || choice == 9 || choice == 10;
}
```

```
// 문제 벡터의 길이 (문제의 수)를 설정하는 함수
void SN::SetArrayLength(int &array_length)
{
    while (true)
    {
        cout << "문제의 수를 결정하세요 (5개 ~ 10개)" << endl;
        cin >> user_input; // 사용자의 입력 (string)

        if (!IsNumber(user_input)) // 사용자의 입력이 숫자인지 확인
        {
            cout << "잘못된 입력입니다. 숫자를 입력하세요." << endl;
            continue;
        }
        if (!IsValidNumber(stoi(user_input))) // 유효한 숫자인지 확인
        {
            cout << "유효하지 않은 숫자입니다." << endl;
            continue;
        }
        array_length = stoi(user_input); // int로 반환
        break;
    }
}
```

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

```
// 문제 벡터를 설정하는 함수
void SN::MakeRandomNumber(vector<int> &random_numbers)
{
    for (int i = 0; i < array_length; ++i) // 설정한 문제 배열의 길이(문제의 수) 만큼 반복
    {
        int randomNum = (rand() % 201) - 100; // -100부터 100까지의 무작위 정수 생성
        random_numbers.push_back(randomNum); // 문제 벡터에 추가
    }
}
```

승리 여부를 확인하는 함수

사용자가 입력한 배열 순서와 정답 배열 순서가 같으면 정답

```
// 게임 결과를 확인하는 함수
bool SN::CheckWin(vector<int> &user_answer, vector<int> &random_numbers)
{
    for (int i = 0; i < user_answer.size() - 1; i++)
    {
        if (user_answer[i] != random_numbers[i]) // 사용자의 배열 순서와 정답의 배열 순서가 다르면 패배
        {
            return false;
        }
    }
    return true; // 사용자의 배열 순서와 정답의 배열 순서가 같으면 승리
}
```

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

```
void SN::GetUserAnswer(vector<int> &user_answer)
{
    cout << "정답을 입력하세요 (띄어쓰기로 구분) :" << endl;
    while (true)
    {
        string user_input;           // 사용자의 입력 문자열 (유효성 체크 x)
        string temp;                 // 사용자의 입력 문자열에 포함된 단어
        getline(cin, user_input);    // 사용자로부터 한줄을 입력 받음, user_input에 저장
        stringstream ss(user_input); // 사용자의 입력을 stringstream으로 저장
        bool is_valid = true;        // 잘못된 입력을 확인하는 변수

        user_answer.clear();         // 사용자의 정답을 담는 벡터 초기화
        while (getline(ss, temp, ' ')) // ss스트림에서 ' '를 구분하여 문자열 temp를 읽어냄
        {
            // 입력받은 각 숫자에 대해 유효성 검사
            if (IsNumber(temp))
            {
                user_answer.push_back(stoi(temp)); // 숫자이면 사용자 정답 벡터에 추가
            }
            else
            { // 숫자가 아닌 입력을 확인하면 반복문 break
                is_valid = false;
                break;
            }
        }
        if (!is_valid) // 잘못된 입력이 있었다면 루프 재시작
        {
            continue;
        }

        if (user_answer.size() == 0) // 유저가 입력이 공란이면 다시 입력받음
        {
            continue;
        }

        // 유저의 정답 벡터의 길이와 문제 벡터의 길이가 다르면 다시 입력받음
        if (user_answer.size() != random_numbers.size())
        {
            cout << "문제의 갯수와 정답의 갯수가 다릅니다. 다시 입력하세요." << endl;
            continue;
        }

        break; // 모든 유효성 검사를 통과하면 종료
    }
}
```

게임 실행 함수의 일부로 사용자가 정답을 입력하는 시간을 계산함

```
cout << endl << "주어진 시간은 " << array_length * 3 + bonus_time << "초 입니다." << endl;

// 설정한 문제 벡터 공개
ShowRandomNumbers(random_numbers);

time_t start = time(0); // 시간 측정 시작
GetUserAnswer(user_answer); // 사용자로부터 답을 입력받음
time_t end = time(0); // 시간 측정 종료

// 걸린 시간 계산 (초 단위)
double duration = difftime(end, start);
```

사용자가 정답을 일정 시간을 넘어서 입력 시 자동 실패

```
bool is_win = CheckWin(user_answer, random_numbers);
// 성공 여부 확인
if (is_win && duration <= array_length * 3 + bonus_time) // 성공
{
    cout << "정답입니다. " << "걸린 시간: " << duration << endl;
    give_score = true;
}
else if (is_win && duration > array_length * 3 + bonus_time) // 정답 but 시간 초과
{
    cout << "시간이 초과되었습니다. " << "걸린 시간: " << duration << endl;
    give_score = false;
}
else // 오답
{
    cout << "실패했습니다." << "걸린 시간: " << duration << endl;
    give_score = false;
}
```

(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

```
class Bonus
{
private: // 점수 관련 변수의 접근 제어를 위한 private
    // Bonus 객체를 사용하는 모든 객체에서 스코어, 보너스, 사용한 보너스를 공유하기 위한 static
    // static이 없으면 값이 공유되지 않는다.
    static int score;      // 점수
    static int bonus;      // 보너스
    static int usedbonus;  // 사용된 보너스
public:
    // 캡슐화한 score, bonus, usedbonus의 getter, setter
    int getScore();
    int getBonus();
    int getUsedBonus();
    void setScore(int record);
    void setBonus(int record);
    void setUsedBonus(int use);
    bool checkUseBonus(); // 보너스 사용여부를 물어보는 함수
    bool useBonus();      // 보너스 사용을 결정하는 함수
};
```

```
int Bonus::score = 0;
int Bonus::bonus = 0;
int Bonus::usedbonus = 0;

void Bonus::setScore(int record){ // 점수를 설정
    score = record;
}

int Bonus::getScore(){ // 점수를 반환
    return score;
}

void Bonus::setBonus(int record){ // 남은 보너스를 설정
    bonus = getScore()/200 - getUsedBonus(); // 남은 보너스 = 점수/200 - 사용한 보너스의 수
}

int Bonus::getBonus(){ // 남은 보너스를 반환
    return bonus;
}

void Bonus::setUsedBonus(int use){ // 사용한 보너스의 수를 더하는 함수
    usedbonus += use; // 매개변수의 값만큼 사용한 보너스의 수를 더함
}

int Bonus::getUsedBonus(){ // 사용한 보너스를 반환
    return usedbonus;
}
```

(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.

- 적용된 배운 내용 : 함수화, 캡슐화, 정적변수(static)
- 코드 스크린샷

```
bool Bonus::checkUseBonus(){ // 보너스 사용여부를 확인하는 함수
    string input; // 사용자의 입력
    while(true){
        if(getBonus() >= 1){ // 남은 보너스가 1 이상이면
            cout << "보너스를 사용하시겠습니까? 보유한 보너스:" << getBonus() << "개" << endl;
            cout << "사용하시려면 'y'를, 사용하지 않으시려면 'n'을 입력하세요."<<endl;
            cin >> input;
            if (input == "y" || input == "Y") { // 사용자의 입력이 y이면 사용여부 true를 반환
                return true;
            }
            else if (input == "n" || input == "N") { // 사용자의 입력이 n이면 사용여부 false를 반환
                return false;
            }
            else { // 사용자의 입력이 y,n이 아니면 루프를 반복하여 입력을 다시 받음
                cout << "잘못된 입력입니다. y 또는 n을 입력해주세요."<<endl;
            }
        }
        else
            return false; // 남은 보너스가 0개이면 false를 반환
    }
}

bool Bonus::useBonus(){ // 보너스를 사용하는 함수
    bool check = checkUseBonus();
    if(check){
        setUsedBonus(1); // 사용된 보너스를 1 증가
        setBonus(getScore()); // 남은 보너스를 갱신
        return true;
    }
    else
        return false;
}
```


묵찌빠, 틱택톡 객체에서 보너스 객체 생성

```
Bonus TTTbonus; // 보너스 시스템을 위한 보너스 객체
isBonusUsed = TTTbonus.useBonus(); // 보너스 사용여부를 확인
```

```
Bonus mcpbonus; // 보너스 시스템을 위한 보너스 객체
isBonusUsed = mcpbonus.useBonus(); // 보너스 사용여부를 확인
```

묵찌빠, 틱택토의 보너스 기믹 (패배 방어권, 돌 한번 더 두기)

```
// 보너스가 사용되었으면 처음으로 돌아감 (= 아직 turn이 넘어가지 않았으므로 플레이어가 돌을 한번 더 둔다.)
if(isBonusUsed == true){
    isBonusUsed = false; // 보너스가 중복으로 사용됨을 막기위해 false를 대입
    continue;
}
```

```
else if (Winner == -1000) { // 컴퓨터가 승리
    // 보너스 사용여부가 true 이면
    if(isBonusUsed == true){
        cout<<"패배 방어권을 사용합니다. 컴퓨터의 공격을 다시 막으세요."<< endl;
        isBonusUsed = false; // 패배 방어권 재사용을 막기 위해 if문의 조건을 false로 변경
        continue; // 패배하지 않고 다시 방어할 기회를 제공
    }
}
```

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

```
if (is_bonus_used == true)
{
    chance = 13; // 보너스 사용시 기회를 13번으로 바꿈
}
```

```
if (is_bonus_used)
{
    bonus_time = 5;
}
```

2) 테스트 결과

(1-1) 틱택토 컴퓨터 행동 알고리즘

- 설명 : 유저가 돌을 둔 후 컴퓨터도 랜덤으로 돌을 둔다.
- 테스트 결과 스크린샷

```
유저 (x)의 차례입니다. -> (x, y) 좌표를 입력하세요: 1 1
---|---|---
|   |   |
---|---|---
| x |   |
---|---|---
|   |   |
---|---|---
```

```
컴퓨터 (o)의 차례입니다.
---|---|---
|   |   |
---|---|---
| x |   |
---|---|---
o |   |   |
---|---|---
유저 (x)의 차례입니다. -> (x, y) 좌표를 입력하세요: |
```

```
---|---|---
| x | x |
---|---|---
|   |   |
---|---|---
| x | o |
---|---|---
o | x | o |
---|---|---
세로에 모두 돌이 놓였습니다!
플레이어 (x)의 승리입니다!
```

```
컴퓨터 (o)의 차례입니다.
---|---|---
| x | o |
---|---|---
x | o | x |
---|---|---
o | x | o |
---|---|---
오른쪽 위 -> 왼쪽 아래 대각선에 모두 돌이 놓였습니다!
컴퓨터 (o)의 승리입니다...
```

```
---|---|---
x | o | x |
---|---|---
o | x | x |
---|---|---
o | x | o |
---|---|---
모든 칸이 다 찼습니다. 종료합니다.
```

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

묵,찌,빠 중 원하는 손동작의 번호를 입력하세요 (1: 묵, 2: 찌, 3: 빠): 1
컴퓨터가 찌를 냈습니다.

묵,찌,빠 중 원하는 손동작의 번호를 입력하세요 (1: 묵, 2: 찌, 3: 빠): 2
컴퓨터가 찌를 냈습니다.

승자는 플레이어입니다!

묵,찌,빠 중 원하는 손동작의 번호를 입력하세요 (1: 묵, 2: 찌, 3: 빠): 3
컴퓨터가 찌를 냈습니다.

묵,찌,빠 중 원하는 손동작의 번호를 입력하세요 (1: 묵, 2: 찌, 3: 빠): 1
컴퓨터가 묵을 냈습니다.

승자는 컴퓨터입니다..

(2-1), (2-2) 단어 맞추기 게임& 단어 불러오기

- 설명 : 행맨 게임을 실행한다.
- 테스트 결과 스크린샷

2.2
입력한 값이 정수가 아닙니다. 다시 입력하세요.
주제를 선택하세요 (1. 동물, 2. 과일, 3. 나라)
-1
유효하지 않은 숫자입니다.
주제를 선택하세요 (1. 동물, 2. 과일, 3. 나라)
k
입력한 값이 숫자가 아닙니다. 다시 입력하세요.
주제를 선택하세요 (1. 동물, 2. 과일, 3. 나라)

유효성 검사

정수가 아닐 경우

유효한 숫자가 아닐 경우

숫자가 아닐 경우

k
입력한 값이 숫자가 아닙니다. 다시 입력하세요.
주제를 선택하세요 (1. 동물, 2. 과일, 3. 나라)
1
알파벳을 입력하세요. (남은 기회: 10번) :

알파벳을 입력 받음

알파벳을 입력하세요. (남은 기회: 10번) : a
틀렸습니다.
알파벳을 입력하세요. (남은 기회: 9번) :

틀리면 기회 1 차감

알파벳을 입력하세요. (남은 기회: 9번) : g
g
알파벳을 입력하세요. (남은 기회: 9번) :

맞추면 알파벳 표시
기회 차감 X

알파벳을 입력하세요. (남은 기회: 9번) : T
t_g_
알파벳을 입력하세요. (남은 기회: 9번) :

대문자 입력도 가능

tige_
알파벳을 입력하세요. (남은 기회: 7번) : r
정답 tiger을(를) 맞췄습니다!

기회 안에 정답을 맞출 경우

ko_a
알파벳을 입력하세요. (남은 기회: 1번) : w
틀렸습니다.
기회를 모두 사용했습니다. 정답은 korea입니다.

기회 안에 정답을 못 맞출 경우

(3-1), (3-2) 숫자 배열 알고리즘 & 숫자 배열 게임

- 설명 : 숫자 배열 게임을 실행한다.
- 테스트 결과 스크린샷

문제의 수를 결정하세요 (5개 ~ 10개)
1
유효하지 않은 숫자입니다.
문제의 수를 결정하세요 (5개 ~ 10개)
1.1
입력한 값이 정수가 아닙니다. 다시 입력하세요.
문제의 수를 결정하세요 (5개 ~ 10개)
k
입력한 값이 숫자가 아닙니다. 다시 입력하세요.

3 초 후에 문제가 나옵니다!
2 초 후에 문제가 나옵니다!
1 초 후에 문제가 나옵니다!

주어진 시간은 15초입니다.
-90 -10 -99 -98 -56

정답을 입력하세요 (띄어쓰기로 구분) :
-99 -98 -90 -56 -10

정답입니다. 걸린 시간: 15
최종 점수 : 500

주어진 시간은 18초입니다.
45 75 -47 -68 -19 -68

정답을 입력하세요 (띄어쓰기로 구분) :
-68 -68 -47 -19 45 75
-68 -68 -47 -19 45 75

시간이 초과되었습니다. 걸린 시간: 32
최종 점수 : 400

주어진 시간은 21초입니다.
92 -30 67 -34 75 35 -56

정답을 입력하세요 (띄어쓰기로 구분) :
0 1 2 3 4 5 6 7
문제의 갯수와 정답의 갯수가 다릅니다. 다시 입력하세요.
0 1 2 3 4 5 6
-56 -34 -30 35 67 75 92

실패했습니다. 걸린 시간: 8
최종 점수 : 400

유효성 검사

유효한 숫자가 아닐 경우

정수가 아닐 경우

숫자가 아닐 경우

5문제 제한시간 안에 맞춤

6문제 제한시간 안에 못 맞춤

7문제 제한시간 안에 정답 틀림

문제 수와 다르게 입력

(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

점수, 보너스 설정 후 목찌빠, 틱택토 연속으로 실행

```
// 게임을 확인하는 메인(임시)
int main(){
    MCP mcpGame; // 목찌빠 객체
    Bonus gameBonus; // 보너스 객체
    TTT tictactoeGame; // 틱택토 객체

    cout<<"초기 점수 : "<<gameBonus.getScore()<<endl;
    cout<<"초기 보너스 : "<<gameBonus.getBonus()<<endl;
    gameBonus.setScore(gameBonus.getScore()+400); // 점수 설정
    gameBonus.setBonus(gameBonus.getScore()); // 남은 보너스 갱신
    cout<<"설정된 점수 : "<<gameBonus.getScore()<<endl;
    cout<<"설정된 보너스 : "<<gameBonus.getBonus()<<endl;

    mcpGame.playMCP(); // 목찌빠 실행

    tictactoeGame.playTTT(); // 틱택토 실행

    cout<<"최종 점수 : "<<gameBonus.getScore()<<endl;
    cout<<"최종 남은 보너스 : "<<gameBonus.getBonus()<<endl;
    cout<<"사용된 보너스 : "<<gameBonus.getUsedBonus()<<endl;
}
```

```
초기 점수 : 0
초기 보너스 : 0
설정된 점수 : 400
설정된 보너스 : 2
보너스를 사용하시겠습니까? 보유한 보너스:2개
사용하시려면 'y'를, 사용하지 않으시려면 'n'을 입력하세요.
y
```

보너스 사용 여부 확인 (y)

```
목,찌,빠 중 원하는 손동작의 번호를 입력하세요 (1: 목, 2: 찌, 3: 빠): 3
컴퓨터가 빠를 냈습니다.
패배 방어권을 사용합니다. 컴퓨터의 공격을 다시 막으세요.
목,찌,빠 중 원하는 손동작의 번호를 입력하세요 (1: 목, 2: 찌, 3: 빠):
```

목찌빠 보너스 기믹 패배 방어권

목,찌,빠 중 원하는 손동작의 번호를 입력하세요 (1: 목, 2: 찌, 3: 빠): 3

컴퓨터가 빠를 냈습니다.

보너스 사용 여부 확인 (y)

승자는 플레이어입니다!

보너스를 사용하시겠습니까? 보유한 보너스:1개

사용하시려면 'y'를, 사용하지 않으시려면 'n'을 입력하세요.

y

유저 (x)의 차례입니다. -> (x, y) 좌표를 입력하세요: 0 0

```
---|---|---
X  |   |   |
---|---|---
   |   |   |
---|---|---
   |   |   |
---|---|---
```

유저 (x)의 차례입니다. -> (x, y) 좌표를 입력하세요: 0 1

```
---|---|---
X  |X  |   |
---|---|---
   |   |   |
---|---|---
   |   |   |
---|---|---
```

틱택토 보너스 기믹 추가 행동권

컴퓨터 (o)의 차례입니다.

```
---|---|---
X  |X  |   |
---|---|---
   |   |   |
---|---|---
O  |   |   |
---|---|---
```

```
X  |X  |O
---|---|---
```

```
X  |X  |
---|---|---
```

```
O  |X  |O
---|---|---
```

세로에 모두 둘이 놓였습니다!

플레이어 (x)의 승리입니다!

최종 점수 : 600

최종 남은 보너스 : 1

사용된 보너스 : 2

승리시 +100

2승 했으므로 남은 점수는 400+200

200점당 보너스 = 남은 보너스 1개

사용된 보너스 = 2개

왼쪽 위 -> 오른쪽 아래 대각선에 모두 돌이 놓였
플레이어 (X)의 승리입니다!
최종 점수 : 500
최종 남은 보너스 : 0
사용된 보너스 : 2

보너스 2번 사용, 목찌빠 패배, 틱택토 승리

승리시 +100

1승 했으므로 남은 점수는 400+100

200점당 보너스 = 남은 보너스 0개

사용된 보너스 = 2개

보너스를 사용하시겠습니까? 보유한 보너스:2개
사용하시려면 'y'를, 사용하지 않으시려면 'n'을 입력하세요.

y
문제의 수를 결정하세요 (5개 ~ 10개)

5
3 초 후에 문제가 나옵니다!
2 초 후에 문제가 나옵니다!
1 초 후에 문제가 나옵니다!

주어진 시간은 20초 입니다.
73 70 -38 19 -86

숫자 배열하기 보너스 사용

기존 시간 3*문제수 에서 보너스 사용으로
+5초 즉 20초로 제한시간 증가

정답을 입력하세요 (띄어쓰기로 구분) :
-86 -38 19 70 73
-86 -38 19 70 73

정답 맞춘 후 바로 행맨 게임 실행

정답입니다. 걸린 시간: 13
보너스를 사용하시겠습니까? 보유한 보너스:1개
사용하시려면 'y'를, 사용하지 않으시려면 'n'을 입력하세요.
y

주제를 선택하세요 (1. 동물, 2. 과일, 3. 나라)
3

보너스 사용으로 기회 +3 기존(10번)

알파벳을 입력하세요. (남은 기회: 13번) :

japa_알파벳을 입력하세요. (남은 기회: 1번) : s
틀렸습니다.

기회를 모두 사용했습니다. 정답은 japan입니다.
최종 점수 : 500
최종 남은 보너스 : 0
사용된 보너스 : 2

행맨 게임 실패, 점수 획득 X

기존 점수 400 + 숫자 배열 성공 100

최종 점수 500

보너스 2번 사용

남은 보너스는 0번

5. 프로젝트 일정

(진행한 작업과 진행 중인 작업 등을 표기)

업무		11/3	11/10	11/17	11/24	12/1	12/8
제안서 작성		완료					
기능1	세부기능1		완료				
	세부기능2			완료			
	세부기능3				완료		
	세부기능4		완료				
기능2	세부기능1					----->	
	세부기능2					----->	