

과제#2 소스 구현 설명

202304178 구서연

문제 정의 :

클래스를 이용하여 랜덤으로 정수를 생성할 줄 알고, 키보드에 특정 키가 들어왔을 때 액션을 취하고, 특정 조건을 만족하면 무한루프를 종료시킬 줄 안다.

문제 해결 방법 :

0. 선언부 / 구현부 / main 분리

클래스를 총 2개(Player, GamblingGame)을 만들 것이기 때문에 선언부를 담은 헤더파일을 2개(Player.h, GamblingGame.h)와 이 클래스들의 구현부를 넣을 cpp파일을 2개(Player.cpp, GamblingGame.cpp)와 main함수를 쓸 main.cpp파일까지 총 4개의 파일을 만들었다.

- Player.h / GamblingGame.h
- Player.cpp / GamblingGame.cpp
- main.cpp

1. 플레이어 클래스 선언하기 - Player.h

```
class Player {  
    string name;  
public:  
    void writeName(string name);  
    string getName();  
};  
  
#endif
```

Player 클래스를 작성하라 했기 때문에, 클래스 안에 플레이어의 이름과 함수들을 선언해준다.

writeName함수 : 문자열을 받아와서 클래스의 name에 넣는 함수다.

getName함수 : 플레이어 이름이 필요할 때 그걸 받아와서 반환하는 함수다.

함수들을 이용해서 name에 접근해 값을 넣고 가져오는 행동을 할 수 있다.

그러나 처음에 이렇게만 썼더니 string에서 오류가 났다. #include <string>을 안 해줬기 때문인데, include을 해줘도 오류가 나서 한참을 헤매었다. 이것을 해결해주려면,

```
#ifndef PLAYER_H  
#define PLAYER_H  
#include <string>  
using namespace std;
```

헤더파일에서도 using namespace std;도 해주어야 오류가 나지 않는다. 이 문장을 쓰지 않으려면 string 대신에 std::string라고 쓰는 것도 방법이다.

그리고 조건 컴파일문으로 여러 번 include해도 문제가 없도록 해준다.

2. GamblingGame 클래스 선언하기 - GamblingGame.h

```
#ifndef GAMBLINGGAME_H
#define GAMBLINGGAME_H
#include "Player.h"
using namespace std;

class GamblingGame {
    Player player[2];
public:
    void gamestart();
    bool turn(string name);
    ~GamblingGame();
};

#endif
```

게임을 하는 과정을 GamblingGame 클래스에서 처리할 것이기 때문에, Player 객체 player를 만들어준다. 이때, 플레이어는 총 2명이고 배열로 만들어주라고 문제에 있어 player[2]을 만들어줬다. Player 클래스의 객체를 만들기 때문에 #include "Player.h"을 해주어야 한다.

gamestart 함수를 이용하여 게임을 진행해준다.

turn 함수는 gamestart 안에서 게임이 진행되는 동안 사용될 엔터키를 입력받고 랜덤으로 수를 뽑아 그 숫자들이 동일한가 판별하여 bool값으로 반환하는 함수다.

Player.h와 마찬가지로 조건 컴파일문으로 여러 번 include해도 문제가 없도록 해준다.

3. Player 클래스 구현부 만들기 - Player.cpp

Player.cpp 앞서서 Player.h에서 #include <string>과 네임스페이스를 지정을 해주었기 때문에 #include "Player.h"로 그 과정은 생략해도 된다.

```
1  #include "Player.h"
2
3  void Player::writeName(string name)
4  {
5      this->name = name;
6  }
7  string Player::getName()
8  {
9      return name;
10 }
```

writeName(string Name)함수는 이름을 입력했을 때 그 값을 객체의 name에 넣어준다.

getName()함수는 객체의 name을 반환을 해주는 함수이다.

이 두 함수를 Player.c에서 구현을 해준다.

4. GamblingGame 클래스 구현부 만들기 - GamblingGame.cpp

```
1  #include <iostream>
2  #include <cstdlib>
3  #include <ctime>
4  using namespace std;
5
6  #include "GamblingGame.h"
7
```

먼저 필요한 것들을 include 해주고 `using namespace std;`도 선언해줘서 `std::`를 생략할 수 있게 한다. 그리고, 미리 만들어 둔 GamblingGame 헤더파일을 include 시켜준다.

4-0-0. GamblingGame::gamestart() 함수 - 버퍼 지우기

게임을 시작하면 <Enter>를 받아 순서가 진행되기 때문에 gamestart함수에서 플레이어의 이름들을 받아 하나씩 `player[0]`, `player[1]`의 name에 넣은 후 플레이어 순서 시작 전, 버퍼를 지우는 과정을 거쳤다.

```
cout << "***** 캐블링 게임을 시작합니다. *****" << endl;
string name;
cout << "첫번째 선수 이름>>";
cin >> name;
player[0].writeName(name);
cout << "두번째 선수 이름>>";
cin >> name;
player[1].writeName(name);
cin.ignore();
```

위와 같이 `cin.ignore()`를 사용하여 지울 수 있었다.

4-0-1. GamblingGame::gamestart() 함수 - 게임 우승 판별

turn함수에서 만약 랜덤으로 뽑은 3개의 숫자(0~2)가 모두 같으면, 1을 반환하고 아니면 0을 반환하게 했다. 그리고 gamestart 함수에서 이 turn함수의 값이 1이면 우승을 했다고 출력을 해준다.

```
while (1) {
    for (int i = 0; i < 2; i++) {
        if ( turn(player[i].getName()) ) {
            cout << player[i].getName() << "님 승리!!!" << endl;
            end = 1;
            break;
        }
        else
            cout << "아쉽군요!" << endl;
    }
    if (end == 1)
        break;
}
```

그리고 게임을 종료해야하는데, 나는 while문 안의 for문에서 이 행동을 했기 때문에 break

로는 게임을 종료할 수 없어, 변수 end를 만들어서 디폴트값을 0으로 두고 게임을 우승했을 때 end를 1로 변경하여 while문에서 end가 1이면 게임을 종료하게 했다.

4-1-0. GamblingGame::turn(string name)함수 - <Enter>키를 받는 법

turn함수에서 키보드에서 엔터키가 눌리면 랜덤 뽑기를 진행했다.

```
while (1) {  
    char enter;  
    cin.get(enter);  
  
    if (enter == '\n')  
        break;  
}
```

while문으로 엔터키가 눌리기 전까지는 게임을 더 진행하지 않게 만들었다. 엔터키가 들어오면 랜덤으로 숫자를 뽑는 것을 진행했다.

4-1-1. GamblingGame::turn(string name)함수 - 숫자 랜덤 뽑기

```
srand((unsigned)time(NULL));  
int randomN[3] = { 0 };  
  
for (int i = 0; i < 3; i++) {  
    randomN[i] = rand() % 3;  
}
```

C언어에서 하는 것처럼 시간을 시드값으로 초기화해주고, 랜덤 숫자를 생성하기 위해서 랜덤 숫자를 넣을 배열 int randomN[3] = {0};을 만들어주었다.

그리고 for문으로 rand()%3의 값을 하나씩 배열에 넣어주었다.

rand()%3는 랜덤값을 0~2로 하기 위해서 3으로 나눈 나머지를 해준 것이다.

```
cout << "WtWt" << randomN[0] << "Wt" << randomN[1] << "Wt" << randomN[2] << "Wt";  
if (randomN[0] == randomN[1] && randomN[1] == randomN[2])  
    return 1;  
else  
    return 0;
```

그리고 랜덤으로 뽑은 이 3개의 값이 같다면 1을 반환하고 아니라면 0을 반환하여 위에서 설명한 것처럼 gamestart() 함수에서 이 turn함수의 리턴값을 사용하여 우승했는지를 판별하여 게임을 종료한다.

5. main함수 - main.cpp

```
#include "GamblingGame.h"  
#include "Player.h"
```

먼저 내가 구현한 클래스들을 이렇게 include 시켜준다.

```
int main(void) {
    GamblingGame game;
    game.gamestart();

    return 0;
}
```

그리고, main함수에서 GamblingGame 객체 game을 생성해주고 gamestart함수를 실행해주면 우리가 원하는 게임을 할 수 있다.

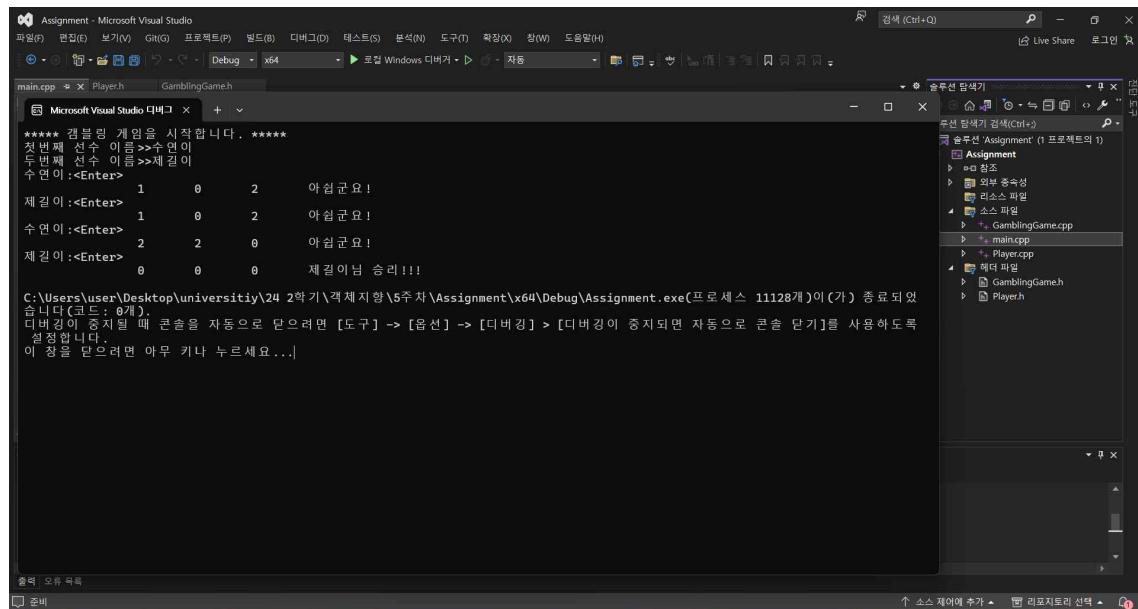
아이디어 평가 :

아이디어에 대한 설명은 위에서 했으며, 내가 생각한 내 아이디어에서의 키 포인트는 다음과 같다.

0. 선언부 / 구현부 / main 파일 분리 → 이때, 한 번 앞에서 선언한 거 생략해도 됨.
 - 조건 컴파일문으로 여러 번 선언 시에도 문제 없도록 한다.
1. Player 클래스 선언부 → public인 두 멤버함수로 private인 멤버변수 name에 접근할 수 있게 된다.
 - string 객체를 사용하기 때문에 #include <string>과 using namespace std;를 해주었다.
 - 여러 번 선언해도 오류가 안 나도록 #ifndef문과 #endif을 작성한다.
2. GamblingGame 클래스 선언부 → Player 객체 배열 player[2]를 만들어줬기 때문에 #include "Player.h"를 해준다.
3. Player 클래스 구현부 → Player.h에서 #include <string>과 네임스페이스를 지정해 주었기 때문에 #include "Player.h"을 함으로써 그 과정을 생략한다.
4. GamblingGame 클래스 구현부 → <Enter>를 받아 순서가 진행해야 해서 이전의 버퍼를 지웠다.
 - while문으로 엔터키가 들어올 때까지 게임을 진행하지 않도록 했다.
 - 랜덤값을 뽑기 위해 시드값을 시간으로 초기화하고, rand()를 3으로 나눈 나머지를 구했다.
5. main함수 - main.cpp → 앞선 파일에서 필요한 내용을 모두 include 했기 때문에 새로 include 않았다.
 - GamblingGame 객체 game을 생성해주고 게임을 실행한다.

위의 아이디어들로 코드를 작성하고 실행하면 다음 장의 화면과 같이 문제가 요구하는 올바른 화면을 도출해 낼 수 있다. 아래의 화면은 main.cpp를 실행했을 때의 화면이다.

실행 화면:



The screenshot shows the Microsoft Visual Studio IDE with a C++ project named 'Assignment'. The console window displays the following output:

```
***** 카펫팅 게임을 시작합니다. *****
첫 번째 선수 이름>>수연이
두 번째 선수 이름>>제갈이
수연이:<Enter>
제갈이:<Enter>
수연이:<Enter>
제갈이:<Enter>
0 0 0 제갈이님 승리!!!

C:\Users\user\Desktop\universitiy\24 2학기\객체지향\5주차\Assignment\x64\Debug\Assignment.exe(프로세스 11128개)이(가) 종료되었
습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록
설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

The right sidebar shows the project structure with files like main.cpp, Player.h, and GamblingGame.h.