소스 구현 설명

-강부경, 황호태-

문제 정의 :

겜블링 게임을 만드는 프로그램이다. 선수의 이름을 초기에 입력을 받으며, 입력을 받기 전에 '**** 겜블링 게임을 시작합니다. ****' 문구를 먼저 띄운다. 선수가 번갈아 자신의 차례에서 <Enter> 키를 치면 랜덤한 3개의 수가 생성되고, 모두 동일한 수가 나오면 게임에서 승리한다. 선수는 Player 클래스로, 2명의 선수는 배열로 구성하며, 게임은 GamblingGame 클래스로 작성한다.

문제 해결 방법 및 아이디어 평가 :

개인적으로 문제가 상당히 난도가 높았다고 생각한다. 어디서부터 소스를 구현해야 할지 그 접근하는 방식부터 고민해야 했다. 문제에서 '선수는 Player 클래스로'라는 조건이 있었기 때문에, Player 클래스에는 선수 고유의 정보들을 넣어야겠다고 설계하였고, GamblingGame에는 그 정보들을 바탕으로 두 명의 선수의 정보를 가져와서 게임을 구현해야겠다고 생각하였다.
메인에서는 어떤 부분만을 남기고 구현부로 넘길지에 대한 견적이 제대로 나오지 않아서 고민을 많이 했다.
문제 해결을 위해서는 먼저 풀이 순서를 정할 필요가 있었는데, 'main -> 선언부 -> 구현부' 순으로 하였다.
그리고 이 아이디어는 문제를 푸는 데에 생각하기 편안하게 한 역할을 하였다.

문제를 해결한 키 아이디어 또는 알고리즘 설명 :

문제를 해결한 키 아이디어를 뽑자면 두 클래스로 하나의 프로그램을 구현하는 과정이 가장 문제 해결 키였다고 생각한다. Player클래스에서 선수 고유의 정보를 가지고 GamblingGame::GamblingGame(string player1Name, string player2Name) {...} 생성자로 선언한 것이 가장 키포인트이다. 그 안에는 players를 포인터로 정의하고, 동적으로 할당해주었다. 그러고나서 GamblingGame 클래스에서 Player 클래스를 이어가서 Player에서 정의한 고유의 랜덤 맴버 함수들을 'play()'에 호출한 것이 중요했다.

<main.cpp>

```
#include <iostream>
#include<string>
#include<cstdlib>
#include<ctime>

using namespace std;

#include "Game.h"

int main() {

	string player1, player2;

	srand((unsigned)time(0)); //랜덤 숫자 생성

	cout << "***** 겜블링 게임을 시작합니다. *****" << endl;

	cout << "첫번째 선수 이름 >>";
	getline(cin, player1);
	cout << "두번째 선수 이름 >>";
	getline(cin, player2);

	GamblingGame setPlayer(player1, player2);//player class를 통해서
	setPlayer.play(); //게임시작
}
```

main 함수 구성은 상당히 많은 시행착오를 걸쳤다. 함수의 내용이 조금 길다 싶으면 묶어놓고 구현부로 넘겼다. 랜덤 함수를 쓸 예정이여서 'srand((unsigned)time(0));'를 우선으로 넣었고, 'cout << "***** 겜블링 게임을 시작합니다. *****" << endl;'를 작성하면서 시작을 알렸다.

그 다음은 이름을 입력받는 것이므로, 'string player1, player2;'를 정의하고, 'cout'과 'getline'을 통해서 선수의 이름을 입력받았다. 'GamblingGame' 생성자, 'setPlayer' 객체를 통해 두 플레이어에 대한 입력을 받고, 객체를 통하여 맴버 함수인 'play()'를 호출하였다. (※ 'play()'에는 두 선수의 고유 정보를 바탕으로 한 게임의 과정과 결과의 정보를 넣을 예정이다.)

<Game.h>

```
#pragma once
class Player {    //player 고유의 데이터를 저장
        string player;
        int dice[3] = { 0 };
public:
        Player(string player);
        void randomDice();
        void diceResult();
        bool diceCheck();
        string getName();
};
class GamblingGame {    //player 고유의 데이터를 가지고 게임 만들기
        Player* players[2];
public:
        GamblingGame(string player1Name, string player2Name);
        void play();
        ~GamblingGame();
};
```

선언부는 구현부랑 조금 왔다 갔다 하면서 구현하였다. 원래는 'dice[3]' 랜덤 게임 정보를 GamblingGame 클래스에 구현하려고 했는데, 선수 개인마다 정보로 하는 것이 편안하다고 판단하여 Player 클래스에 넣었고, 나중에 GamblingGame의 'play()' 멤버함수에서 이 모두를 넣을 생각이었다.

우선 Player 클래스부터 보면, 선수를 입력 받을 'string player;'가 필요했고, 랜덤 숫자를 돌릴 것이기 때문에 'int dice[3] = { 0 };'를 선언하였다. 'void randomDice();', 'void diceResult();', 'bool diceCheck();'부분은 구현부랑 왔다 갔다 하였는데, 결국 필요했던 것은 dice안에 랜덤 숫자를 넣는 함수와 그 넣은 값들을 보여주는 함수, 그리고 그 세 개의 숫자가 같은지 판단하는 함수가 필요했다. 그리고 구현부를 하면서 'string getName();'을 추가하였다.

GamblingGame 클래스는 Player의 정보를 가져올 필요가 있었고, 그 선수들의 정보를 문제에서 배열로 받으라고 하였으므로 'Player* players[2];' 포인터 배열로 받았다. 그리고 구현부로 넘어갈 것이다.

<Game.cpp>

```
#include <iostream>
#include<string>
#include<cstdlib>
#include<ctime>

using namespace std;

#include "Game.h"

Player::Player(string playername) {
	player = playername;
}
void Player::randomDice() {
	for (int i = 0; i < 3; i++) {
		dice[i] = rand() % 3;
	}
}
void Player::diceResult() {
	for (int i = 0; i < 3; i++) {
		cout << dice[i] << " ";
	}
}

bool Player::diceCheck() {
	return (dice[0] == dice[1]) && (dice[1] == dice[2]);
}

string Player::getName() {
	return player;
}
```

'Player::Player(string playername) { player = playername; }' 생성자, void Player::randomDice() {...} 랜덤을 for문으로 해서 배열로 입력받고, 'void Player::diceResult() {...}' 그 결과 값을 보여주고, 'bool Player::diceCheck() {...}' 세 개의 값이 같은지 다른지 확인하였다. 'string Player::getName() {...}'으로 player를 리턴해주었다.

```
GamblingGame::GamblingGame(string player1Name, string player2Name) {
          players[0] = new Player(player1Name);
          players[1] = new Player(player2Name);

}
void GamblingGame::play() {
    int turn = 0;
    while (true) {
         cout << players[turn]->getName() << ": <Enter>";
         cin.ignore();

         players[turn]->randomDice();
         players[turn]->diceResult();

         if (players[turn]->diceCheck()) {
              cout << players[turn]->getName() << "님 승리!!" << endl;
              break;
         }
         else {
              cout << "아쉽군요!" << endl;
         }
         turn = (turn + 1) % 2;
    }
}
GamblingGame::~GamblingGame(){
          delete players[0];
          delete players[1];
}
```

GamblingGame::GamblingGame(string player1Name, string player2Name) {...} 와 GamblingGame::~GamblingGame(){...}에서 players[]를 new delete를 통해 동적 할당해주었다.

마지막 'play()' 부분은 계속해서 반복해야 하므로, while문으로 시작하였고, '이름 : <Enter>' 다음에 Enter를 누르면 시작되어야 하므로 'cin.ignore()'라는 함수를 넣었다.

그리고 차례차례 3개의 랜덤 맴버함수를 호출하였고, 'diceCheck()'는 'boolean'함수로 받아서 참이면 '승리!!' 아니면 '아쉽군요!'하고 반복문을 계속 진행하게 하였다.

순서를 번갈아 가면서 돌아야 한다. 여기서 문제를 해결한 방법은 선수들의 이름을 배열로 받았다는 것을 활용하여 0과 1이 계속 돌 수 있도록 ' int turn = 0;'을 정의하였고, while문이 끝날 때마다 turn에 +1 그리고 2 이상이 되지 않도록 %2의 연산을 해주면서 'players[0]'과 'players[1]'이 번갈아 가면서 호출할 수 있게 되었다.