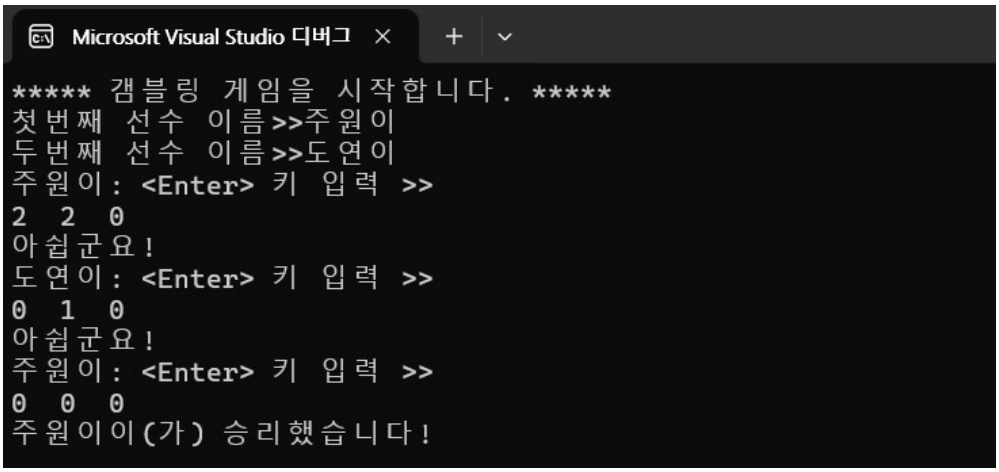
객체지향프로그래밍 과제#2

소프트웨어전공 202284012 김주원 / 소프트웨어전공 202304012 김도연

1. 소스 수행 결과 화면



2. 문제 정의

두 명의 플레이어가 번갈아 가며 무작위로 세 개의 숫자를 생성하고 모두 동일한 숫자가 나오면 승리하는 게임을 구현한다. 추상화를 통해 플레이어와 게임 클래스를 정의하고 캡슐화로 객체 내부 데이터를 보호한다. 게임 진행 중 플레이어 간의 상호작용과 제어 흐름을 관리하며 무작위성을 통해 랜덤한 결과를 생성한다.

3. 문제 해결 방법

<Player 클래스 선언부 – Player.h>

1. 기본 로직

Player 클래스는 GamblingGame에서 각 플레이어의 이름을 저장하고 게임 중 사용할 랜덤 숫자를 제공하는 역할을 한다.

2. 헤더 파일

#ifndef PLAYER_H, #define PLAYER_H, #endif로 구성된 헤더 가드는 헤더 파일이 여러 번 포함되더라도 컴파일 에러가 발생하지 않도록 방지함

#include <string>를 통해 플레이어의 이름을 std::string 타입으로 저장하기 위해 문자열 관련 라이브러리를 포함

3. 함수 구현

name 변수는 Player 클래스의 멤버 변수로 플레이어의 이름을 저장

Player(std::string name);

- 클래스의 생성자로 객체가 생성될 때 입력 받은 이름을 멤버 변수 name에 저장하여 각 플레이어를 고유하게 정의

멤버 함수 getRandomNumber()

- 플레이어가 게임 중 랜덤한 행동을 하거나 숫자를 얻어야 할 때 호출되며 0부터 2까지의 무작위 숫자를 반환함

<Player 클래스 구현부 – Player.cpp>

1. 기본 로직

Player.h에서 선언된 Player 클래스의 생성자와 멤버 함수 getRandomNumber()의 실제 동작을 구현한다. 생성자는 각 플레이어의 이름을 저장하는 역할을 하며 이를 통해 객체를 초기화한다. getRandomNumber() 함수는 게임 중에 필요한 랜덤한 숫자를 생성하고 반환하는 역할을 한다. 이를 통해 Player 클래스는 플레이어의 고유한 특성을 유지하면서 게임에서 랜덤한 행동을 할 수 있다.

2. 헤더 파일

#include "Player.h"를 통해 Player.h에서 선언된 Player 클래스의 정의를 포함하여 해당 클래스의 멤버 변수와 함수들에 접근할 수 있게 함

#include <cstdlib>를 통해 C++ 표준 라이브러리에서 제공하는 rand() 메소드를 사용하기 위해 포함하며 이때 rand() 메소드는 랜덤한 정수를 생성하는 함수

3. 함수 구현

생성자 Player::Player(std::string name)

- 생성자는 Player 객체가 생성될 때 호출되며 입력 받은 이름을 Player 클래스의 멤버 변수 name에 저장
- 각 플레이어의 이름을 관리하고 이름을 기반으로 객체를 구별할 수 있는 기능을 함
- 초기화 리스트 : name(name)을 사용하여 생성자 매개변수를 멤버 변수 name에 할당

멤버 함수 int Player::getRandomNumber():

- 0부터 2까지의 랜덤한 숫자를 반환하는 역할
- rand() 메소드는 매우 큰 범위의 숫자를 반환하는데 이 값을 0부터 2까지로 제한하기 위해 % 3 연산을 사용하여 0, 1, 2 중 하나를 반환
- GamblingGame에서 플레이어가 랜덤한 숫자를 필요로 할 때 호출됨

<GamblingGame 클래스 선언부 – GamblingGame.h>

1. 기본 로직

startGame() 메소드 호출로 게임이 시작되고 각 플레이어가 번갈아 가며 playTurn() 메소드를 호출하여 자신의 턴을 진행한다. 플레이어가 생성한 숫자가 승리조건을 만족하는지 체크하고 누군가 승리 조건을 만족하면 게임이 종료되고 결과가 출력된다.

2. 헤더 파일

헤더 파일이 중복으로 포함되는 것을 방지하여 컴파일 오류를 예방하기 위해 #ifndef GAMBLINGGAME_H와 #define GAMBLINGGAME_H로 중복 포함 방지용 전처리기 지시문 설정

3. 함수 구현

public 접근 지정자를 사용하여 GamblingGame 클래스의 멤버에 외부에서 접근 가능할 수 있도록 설정

Player* player1; Player* player2;으로 첫 번째, 두 번째 플레이어를 나타내는 포인터를 지정하고 이 속성을 통해 게임의 각 턴에서 어떤 플레이어 차례인지 쉽게 확인할 수 있도록 함

GamblingGame(Player* p1, Player* p2);을 통해 두 개의 Player 객체를 인수로 받아 GamblingGame 객체를 초기화하여 생성자 선언

startGame() 메소드

- 게임의 주 루프를 실행하며 플레이어가 번갈아 가며 숫자를 생성하고 승리 조건을 확인하는 기능을 포함
- 특정 값을 계산하거나 반환할 필요가 없으므로 void를 사용하여 메서드의 목적을 명확히 함
- 게임이 시작되면 이 메소드가 호출되어 전체 게임 진행을 관리함

bool playTurn(Player* player) 메소드

- 플레이어가 턴을 진행할 때 호출되며 플레이어에게 랜덤 숫자를 생성하게 하고 그 숫자가 승리조건을 만족하는지 체크
- bool을 사용하여 승리조건을 만족하면 true를 반환하고 그렇지 않으면 false를 반환하도록 설정
- 이 메소드를 통해 게임의 흐름을 제어하고 턴이 끝난 후의 상태를 확인

4. 헤더 파일 보호

헤더 파일이 여러 번 포함되는 것을 방지하고 코드의 안정성과 가독성을 높이기 위해 #endif 사용

<GamblingGame 클래스 구현부 – GamblingGame.cpp>

1. 기본 로직

두 플레이어의 턴을 반복적으로 진행하고 각 턴에서 무작위 숫자를 생성하여 승리

조건을 확인한다. startGame() 메서드로 무한 루프를 통해 플레이어의 턴을 차례로 호출하고 승리 시 게임을 종료한다. playTurn() 메서드로 각 플레이어가 턴을 수행하고 숫자가 모두 동일하면 승리 메시지를 출력한다.

2. 헤더 파일

#include "GamblingGame.h"를 통해 GamblingGame 클래스의 헤더 파일 포함

#include <iostream>을 통해 C++ 표준 입출력 스트림인 std::cin과 std::cout을 사용할 수 있도록 설정

#include <string>을 통해 C++ 표준 문자열 라이브러리를 포함하여 std::string, std::getline()과 같은 문자열 관련 작업을 쉽게 수행할 수 있도록 함

3. 함수 구현

GamblingGame(Player* p1, Player* p2) : player1(p1), player2(p2) {}

Player 객체 포인터인 p1과 p2를 매개변수로 받아 player1과 player2 멤버 변수에 초기화하여 생성자 구현

startGame() 메서드로 while 무한 루프를 통해 게임이 진행되며 각 플레이어의 턴이 종료된 후 승리 여부를 확인하고 누군가 승리하면 루프를 종료시킴

playTurn() 메서드

- 사용자에게 엔터 키 입력을 요청하여 턴을 시작하기 위해 enterKey라는 문자열 변수를 선언하고 이때 std::string 클래스를 사용하여 문자열 처리를 쉽게 수행할 수 있도록 함
- 현재 턴을 진행하는 플레이어 이름과 사용자에게 엔터 키를 입력하라는 메시지 출력
- std::getline(std::cin, enterKey);를 사용하여 사용자로부터 입력받아 enterKey 변수에 저장하고 사용자가 엔터 키를 누를 때까지 대기
- 세 개의 숫자를 num1, num2, num3로 설정하고 랜덤 숫자를 생성하기 위해 Player 클래스의 getRandomNumber() 메소드 사용
- if-else 문을 사용하여 num1과 num2가 같고 동시에 num2와 num3가 같으면 승

리 문자열을 출력하고 true를 반환하여 게임을 종료
- 승리 조건을 만족하지 않으면 "아쉽군요!" 문자열을 출력하고 다음 턴으로 넘어가기 위해 false 반환

<main 함수 – main.cpp>

1. 기본 로직

사용자로부터 입력을 받아 게임을 설정하고 시작하는 역할을 한다. 각 클래스 간의 상호작용을 통해 게임의 흐름을 제어한다. 게임의 기본 설정과 실행을 담당하며 게임의 진행 과정을 제어하는 로직이 담겨있다.

2. 헤더 파일

#include "Player.h"를 통해 Player 클래스와 관련된 기능을 사용하기 위한 헤더 파일을 포함

#include "GamblingGame.h"를 통해 게임 로직을 제어하는 GamblingGame 클래스에 대한 정의를 포함

#include <iostream>를 통해 표준 입력 및 출력을 처리하기 위한 라이브러리 포함

#include <ctime>은 시간 관련 함수들을 사용하기 위한 헤더로 srand() 함수로 랜덤 시드를 초기화할 때 필요

#include <string>는 문자열을 다루기 위한 표준 라이브러리로 플레이어의 이름을 문자열로 저장하고 처리하기 위해 포함

3. 함수 구현

```
srand((unsigned int)time(0));
```

- rand() 메소드를 통해 랜덤한 숫자를 생성할 때 현재 시간을 기반으로 시드를 설정
- time(0)을 이용하여 현재 시간을 기반으로 시드를 설정하므로 매번 다른 랜덤 숫자를 생성할 수 있게 함
- 이 설정을 통해 게임의 랜덤 요소에 변동성을 부여하는 기능을 함

std::cout << "***** 갬블링 게임을 시작합니다. *****" << std::endl;를 통해 게임이 시작됨을 사용자에게 알리는 메시지를 출력

std::getline(std::cin, player1Name); std::getline(std::cin, player2Name);

- 각 플레이어의 이름을 사용자로부터 입력 받아 문자열로 저장하는 역할
- 첫 번째와 두 번째 플레이어의 이름을 각각 player1Name과 player2Name에 저장
- 사용자로부터 입력 받은 플레이어 이름을 이후에 Player 객체를 생성할 때 사용
- GamblingGame 객체가 두 플레이어 객체의 주소(포인터)를 전달받아 관리함
- 이를 통해 각 플레이어의 실질적인 객체를 조작할 수 있음

Player player1(player1Name); Player player2(player2Name);

- 입력 받은 이름을 사용하여 Player 객체를 생성
- 각 객체는 플레이어의 이름을 저장하고 게임에서 플레이어를 구분함

GamblingGame game(&player1, &player2);

- 두 명의 플레이어를 포인터로 전달하여 게임 객체를 생성
- GamblingGame 클래스는 게임의 전반적인 흐름을 제어하는데 이 객체를 통해 두 플레이어 간의 GamblingGame을 시작할 수 있게 됨

game.startGame();

- GamblingGame 객체의 startGame() 메소드를 호출하여 게임을 시작
- 게임의 진행 로직을 포함하며 게임이 어떻게 진행되는지에 대한 세부 사항은 GamblingGame 클래스에 정의되어 있음

return 0;으로 메인 함수가 정상적으로 종료되었음을 나타냄