객체지향프로그래밍 - 과제 #3

- 202184021 소프트웨어전공 박지민

1. 12-1 문제

1-1. 문제 정의

> (1) main() 의 실행 결과가 다음과 같이 되도록 Dept 클래스에 멤버들을 모두 구현하고, 전체 프로그램을 완성하라.

1. 10명분의 점수를 `cin`을 통해 입력받고서,
2. 그 중 60점을 초과하는 인원수를 출력한다.
3. `main()`의 실행 결과는 정상적으로 종료된다.

1-2. 문제 해결 방법

1-2-1. 클래스 설계

- **Dept 클래스** : 학생들의 점수를 저장하고, 핵심 로직이 들어있는 클래스
  - **구현 대상**
  - `Dept(int size)` : 생성자
  - `Dept(const Dept& dept)` : 복사 생성자
  - `~Dept()` : 소멸자
  - `int getSize()` : 학생 수를 반환하는 함수
  - `void read()` : cin을 통해 학생들의 점수를 입력받는 함수
  - `bool isOver60(int index)` : 학생점수의 index를 통해 60점을 넘었는지 확인하는 함수

1-2-2. 주의사항: 메모리 두 번 해제 에러

main() 함수에서 사용되는 `countPass(Dept dept)` 함수는
Dept 객체를 복사하여 사용하므로, 복사 생성자를 구현해야 한다.

이때, 복사 생성자는 기본적으로 **얕은 복사(shallow copy)** 를 수행한다.
하지만 `scores`는 동적 할당된 배열이므로, 두 객체가 같은 메모리를 가리키게 된다면,
소멸 시 `scores`에 대한 메모리 해제가 두 번 일어나게 되어, **에러**가 발생한다.
때문에 복사 생성자를 구현할 때, `scores` 배열을 동적으로 할당하여 깊은 복사를 수행한다.
이후, 메모리에서 누수가 발생하지 않도록, 생성 시 할당된 메모리를 소멸자에서 해제한다.

1-3. 아이디어 평가

- **메모리 해제 에러 방지**
  - C++의 메모리 관리를 이해하고, 복사 생성자와 소멸자를 구현하여 **깊은 복사(deep copy)** 를 수행하였다.
  - 동적 할당된 메모리를 소멸자에서 해제하여 메모리 누수를 방지하였다.
- **객체지향 프로그래밍**
  - Dept 클래스를 설계하여 **역할과 책임**을 명확히 하였다.
  - **캡슐화**를 위해 멤버 변수를 `private`로 선언하고, 접근을 위한 멤버 함수를 제공하였다.

> C++에서 기본 접근자는 `private`이다.

1-4. 문제를 해결한 키 아이디어 또는 알고리즘 설명

```
Dept::Dept(const Dept& dept) {
    size = dept.size;
    scores = new int[size];  // 깊은 복사
    for (int i = 0; i < size; ++i) {
        scores[i] = dept.scores[i];
    }
    cout << "[복사 생성자] 호출" << endl;
}
```

- 복사 생성자를 **깊은 복사(deep copy)**로 구현하여, 메모리 누수를 방지하였다.
- 또한, `countPass(Dept dept)` 부분에서 사용된 복사 생성자가 얕은 복사였다면,
    - 기존 객체와 복사된 객체가 동일한 scores 포인터를 가리키게 되고,
    - 두 번 소멸자가 호출되며 delete[]도 두 번 일어나서, 메모리 해제 에러가 발생했을 것이다.

2. 12-3 문제

2-1. 문제 정의

> Dept 클래스 복사 생성자를 제거하라. 복사 생성자가 없는 상황에서도 실행 에러가 발생하지 않게 하려면 어느 부분을 수정하면 될까? 극히 일부분의 수정으로 해결된다. 코드를 수정해보라.

1. 10명분의 점수를 `cin`을 통해 입력받고서,
2. 그 중 60점을 초과하는 인원수를 출력한다.
3. `main()`의 실행 결과는 정상적으로 종료된다.
    1. `main()` 함수에서 `countPass` 함수를 호출한다.
    2. 이때 복사 생성자가 호출되지 않도록 수정한다.

2-2. 문제 해결 방법

2-2-1. 클래스 설계

- **Dept 클래스** : 학생들의 점수를 저장하고, 핵심 로직이 들어있는 클래스
    - **구현 대상**
    - `Dept(int size)` : 생성자
    - `~Dept()` : 소멸자
    - `int getSize()` : 학생 수를 반환하는 함수
    - `void read()` : cin을 통해 학생들의 점수를 입력받는 함수
    - `bool isOver60(int index)` : 학생점수의 index를 통해 60점을 넘었는지 확인하는 함수

2-2-2. 주의사항: 복사 생성자 제거

main() 함수에서 사용되는 `countPass(Dept dept)` 함수는
Dept 객체를 복사하여 사용하므로, 복사 생성자를 구현해야 하지만, 현재는 복사 생성자를 구현할 수 없다.

이를 해결하기 위해, `countPass` 함수가 매개변수로 Dept 객체를 복사하여 전달 받는 것이 아닌, **참조** 변수로 전달 받도록 수정한다.

이렇게 수정하게 되면, 객체의 복사가 발생하지 않으므로 복사 생성자가 호출되지 않는다.
때문에 소멸자도 두 번 호출될 일이 없어, 실행 에러가 발생하지 않는다.

2-3. 아이디어 평가

- **복사 생성자 제거 후, 실행 에러 방지**
  - 복사 생성자를 제거하여, Dept 객체를 복사할 때 에러가 발생하지 않도록 하였다.
  - `countPass`의 매개변수를 **참조** 변수로 변경하여, 객체의 복사를 피하고, 객체의 참조를 전달하였다.
  - 이렇게 되면, 새로운 객체가 생성되지 않고, 기존 객체의 참조를 전달하므로 복사 생성자가 호출되지 않는다.
  - 복사 생성자가 호출되지 않으면, 소멸자가 두 번 호출될 일도 없기에 안전하다.

2-4. 문제를 해결한 키 아이디어 또는 알고리즘 설명

```
int countPass(Dept& dept) { ... }
```

- `countPass` 함수의 매개변수를 참조 변수로 변경하여, 객체의 복사를 피하고, 실행 에러를 방지하였다.

> `&` 키워드는 참조 변수를 선언하는 키워드다.