202304105 이상은

1. 문제 정의

- 주어진 문제는 Dept라는 학과를 나타내는 클래스를 정의하고, 학생들의 성적을 입력 받아 성적이 60점 이상인 학생들의 수를 계산하는 프로그램을 작성하는 것이다.

2. 문제 해결 방법

1. **학생 성적 저장 및 조회**:
    a. 학생들의 성적을 저장하기 위해 Dept 클래스 내에 동적으로 할당되는 정수 배열을 사용한다.
    b. 성적 입력은 read() 멤버 함수에서 수행하며, 각 학생의 성적이 배열에 저장된다.
2. **통과 학생 수 계산**:
    a. isOver60() 멤버 함수에서 각 학생의 성적이 60점 이상인지를 확인할 수 있도록 구현한다.
    b. countPass() 함수에서 학과 내에서 60점 이상인 학생의 수를 계산하여 출력한다.
3. **복사 생성자 문제 해결**:
    a. 복사 생성자가 없을 때도 프로그램이 정상적으로 동작하도록 하기 위해, countPass() 함수가 객체를 참조(const Dept&)로 받아 처리하도록 수정하여 복사 생성자가 없어도 문제가 발생하지 않도록 한다.

3. 아이디어 평가

1. **동적 할당 사용**:
    a. 학생들의 성적을 저장하기 위한 배열은 Dept 객체가 생성될 때 크기를 정하고, 그 크기만큼 동적으로 메모리를 할당하는 방식으로 구현했다. 소멸자를 통해 메모리 누수를 방지할 수 있다.
2. **복사 생성자 사용 및 제거**:

a. 처음에는 복사 생성자를 구현하여 객체를 복사하는 과정에서 데이터를 안전하게 복사할 수 있도록 했지만, 복사 생성자를 제거한 이후에도 객체 복사가 필요하지 않도록 참조 전달 방식을 사용하여 프로그램이 정상적으로 작동하도록 수정했다.

3. **성적 비교 및 계산**:

   a. 성적이 60점 이상인 학생을 계산하는 로직은 간단한 반복문을 사용하여 `isOver60()` 함수로 판단한다

4. 문제를 해결한 키 아이디어 또는 알고리즘 설명

키 아이디어 1: 동적 메모리 할당과 관리

- 학생들의 성적을 저장하는 배열을 `Dept` 객체가 생성될 때 동적으로 할당하고, `delete[]`로 소멸자를 통해 할당된 메모리를 관리한다. 이 방식을 사용하면 학생 수가 고정되어 있지 않고, 필요한 크기만큼만 메모리를 할당할 수 있어 유연한 데이터 처리가 가능하다.

키 아이디어 2: 성적 계산을 위한 참조 전달 방식

- 복사 생성자를 제거하면서 객체 복사가 필요하지 않도록, `countPass()` 함수에서 `Dept` 객체를 참조로 전달받아 성적을 처리하는 방식으로 수정한다. 이 방식은 불필요한 객체 복사를 방지하며, 프로그램이 더 효율적으로 동작할 수 있도록 한다.

키 아이디어 3: const 멤버 함수 사용

- `countPass()` 함수에서 `Dept` 객체를 `const`로 참조하기 때문에, 객체의 상태를 변경하지 않는다는 보장을 주기 위해 `getSize()`와 `isOver60()` 함수를 `const` 멤버 함수로 선언하여 오류를 해결했다. 이는 객체의 불변성을 유지하면서도 데이터를 안전하게 처리할 수 있는 방식이다.