12-(1) 문제정의

주어진 문제는 학생들의 성적을 관리하는 Dept 클래스를 구현하고, 이 클래스를 사용하여 60점 이상인 학생의 수를 계산하여 출력하는 것입니다. 사용자는 10개의 성적을 입력해야 하며, 최종적으로 60점 이상인 학생의 수를 출력하는 결과를 얻어야 합니다.

12-(1) 문제 해결 방법

Dept 클래스를 설계하여 학생 수와 성적을 저장할 수 있는 배열을 동적으로 할당합니다. 클래스 설계 후 read()를 사용하여 사용자가 입력한 성적을 배열에 저장하고, isOver60()를 사용하여 특정 인덱스의 성적이 60점 이상인지 확인합니다. 또한 복사 생성자와 소멸자를 구현하여 메모리 관리와 객체 복사를 적절히 처리하여 멤버 함수를 구현합니다. 이후 countPass() 함수를 통해 60점 이상인 학생의 수를 계산하고, 프로그램의 시작점으로, Dept 객체를 생성하고 입력을 받고 결과를 출력합니다.

12-(1) 아이디어 평가

학생 수를 계산하는 과정의 코드에서 복사 생성자를 정의했지만, 매개변수를 Dept& dept로 설정하여 복사 시 오류가 발생할 가능성을 발견하였습니다. 그 과정에서 복사 생성자의 필요성을 인식하게 되었고, 참조를 활용하여 객체를 안전하게 전달하는 방법으로 수정하여 프로그램이 정상적으로 실행되었고, 60점 이상의 학생 수를 정확히 출력할 수 있었습니다.

12-(1) 문제를 해결한 키 아이디어 또는 알고리즘 설명

Dept 클래스의 복사 생성자를 const Dept&로 정의하여 객체를 안전하게 전달하는 방식으로 코드를 썼습니다. 이를 통해 원본 객체의 메모리와 데이터를 안전하게 유지할 수 있었습니다. 복사 생성자 내에서 깊은 복사를 구현하여 두 객체가 서로 독립적으로 동작하도록 하여 프로그램이 정상적으로 실행되었고, 60점 이상의 학생 수를 정확히 계산하여 출력할 수 있었습니다.

12-(3)문제정의

복사생성자를게거 하고 실행오류가 발생하지 않게 하려면 어떻게 해야 하는가 간단한 해결책이 있다고 말했습니다. countPass 함수는 Dept 객체를 값으로 전달받고 있습니다. 즉, Dept 객체가 복사되기 때문에 복사 생성자가 호출됩니다. 이 경우 복사 생성자를 정의하지 않으면, 기본 복사 생

성자가 호출되어 얕은 복사가 발생하게 됩니다. 따라서, 복사 생성자가 제거된 경우, 복사 생성자 자체를 사용하지 않도록 코드를 수정해야 합니다.

12-(3)문제 해결 방법

countPass 함수에서 Dept 객체를 값으로 전달하기 때문에 countPass(com)을 호출할 때 com 객체가 복사되고, 이 과정에서 복사 생성자가 호출됩니다. 복사 생성자가 없다면 기본 복사 생성자가 자동으로 사용되어 얕은 복사가 이루어지며 오류가 날 수 있습니다 참조 전달을 사용한다면 객체를 복사하지 않으므로 복사 생성자가 호출되지 않습니다. 그렇기 때문에 복사 생성자를 정의하지 않아도 문제가 발생하지 않습니다.

12-(3)아이디어 평가

복사생성자를 제거하는것만으로는 코드가 잘 구동되지 않음 그것의 이유가 무엇일까 생각해보니 깊은 복사를 하기위해 복사생성자를 두지 않으면 호출시 얕은 복사를 해 오류가 난다고 생각 그래서 호출이 아닌 다른 방식인 참조를 하기로 했고 성공적이였습니다.

12-(3)문제를 해결한 키 아이디어 또는 알고리즘 설명

아이디어 평가와 동일하게 (3)번 핵심 아이디어는 복사 생성자를 호출하지 않고 객체를 안전하게 함수에 전달하는 방법을 통해 불필요한 복사를 방지하기 위해 객체를 참조(&)로 전달하여 복사 생성자 호출을 회피하는것입니다.