객체지향 프로그래밍 과제 보고서

202104389 황형진, 202104309 이진용

1. 문제 정의

주어진 문제는 10명의 학생 점수를 입력받아, 60점 이상을 받은 학생의 수를 계산하는 것입니다. 복사 생성자를 통해 동적 메모리 할당과 값의 저장, 처리를 하면 되는 문제입니다.

2. 문제 해결 방법

1. Dept 클래스 설계: 학생 점수를 저장하고 관련 작업을 수행할 수 있도록 클래스를 설계합니다. 이 클래스는 점수의 크기와 점수를 저장할 배열을 멤버 변수로 가집니다.

2. 동적 메모리 사용: 학생 점수의 수를 동적으로 할당하기 위해 new 연산자를 사용하여 메모리 공간을 생성합니다. 생성된 메모리는 클래스 소멸자에서 해제하여 메모리 낭비를 방지합니다.

3. 깊은 복사 생성자 구현: 객체를 복사할 때, 점수를 저장하는 배열도 올바르게 복사되도록 깊은 복사 생성자를 구현하였습니다. 이를 통해 객체가 복사될 때 동적 메모리의 내용을 안전하게 복사할 수 있습니다.

4. 점수 입력 및 계산: read를 통해 사용자로부터 점수를 입력받고, countPass를 통해 60점 이상인 학생 수를 계산하여 출력합니다.

3. 아이디어 평가

- 동적 메모리 관리: 점수를 동적 배열로 관리함으로써 메모리 사용의 유연성을 높였습니다. 하지만 동적 메모리 관리는 복잡성을 증가시켜 메모리 낭비의 위험이 있습니다. 이 문제를 해결하기 위해 소멸자를 통해 할당된 메모리를 안전하게 해제하였습니다.

- 깊은 복사 생성자 사용: 깊은 복사를 샤용해 동적 배열을 복사하도록 하여 메모리 중복 해제와 같은 오류를 방지했습니다.

- 참조 전달 사용: countPass를 통해 객체를 복사하지 않고 참조로 전달함으로써, 불필요한 복사 비용을 줄이고 성능을 개선하였습니다.

4. 문제를 해결한 키 아이디어 또는 알고리즘 설명

이 문제를 해결하기 위한 핵심 아이디어는 깊은 복사 생성자의 사용입니다. Dept 클래스에서 점수 배열을 동적으로 할당하므로, 단순한 얕은 복사를 사용할 경우 복사된 객체와 원본 객체가 동일한 메모리 공간을 공유하게 됩니다. 이러한 경우, 한 객체가 소멸될 때

다른 객체의 메모리가 무효화되어 프로그램 에러가 생깁니다. 이를 방지하기 위해 깊은 복사 생성자를 구현하여 객체가 복사될 때 별도의 메모리 공간을 가지도록 하였습니다.