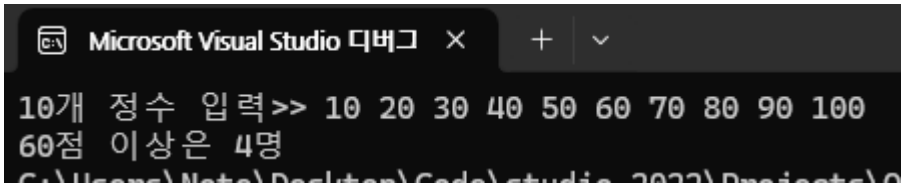
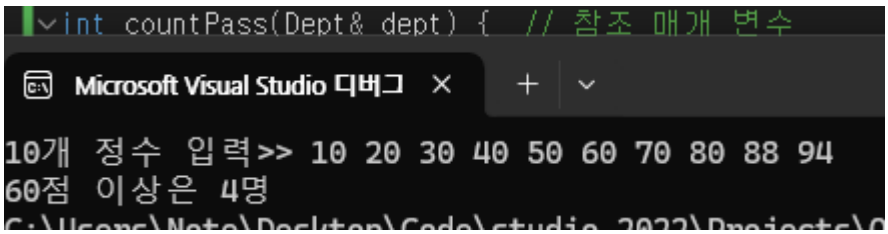
#12-1번


Microsoft Visual Studio 디버그
10개 정수 입력>> 10 20 30 40 50 60 70 80 90 100
60점 이상은 4명

#12-3번


int countPass(Dept& dept) { // 참조 매개 변수
Microsoft Visual Studio 디버그
10개 정수 입력>> 10 20 30 40 50 60 70 80 88 94
60점 이상은 4명

객체지향프로그래밍[02]
202304035 김유진
202104354 지현준

#문제 정의

12-1. 실행 결과에 맞게 멤버 함수들(Dept(), ~Dept(), read(), isOver60())을 구현하여 프로그램을 완성시키는 문제다. 우선 read()의 경우, 10개의 점수를 입력받도록 해야한다. countPass()는 com 학과에서 60점 이상으로 통과하는 학생 수를 리턴하도록 해야 한다.

12-3. 복사 생성자(Dept(const Dept& dept)) 없이 오류가 발생하지 않도록 극히 일부분의 코드를 수정하고 실행하는 문제다.

#문제 해결 방법

12-1.

- Dept(const Dept& dept) 함수

Dept 객체를 복사하는 복사생성자로, 같은 내용을 가지도록 새로운 객체를 만들면서 참조 매개변수의 기존 사이즈와 점수 정보를 복사한다.

- ~Dept() 함수

프로그램이 종료될 때 생성되었던 객체를 소멸하는 소멸자로 동적으로 메모리가 할당되었던 scores 배열을 if문을 사용하여 scores가 남아있다면 delete를 통해 힙으로 반환한다.

- read() 함수

지정한 size만큼 점수를 입력받는 함수로, Dept클래스의 객체인 com을 생성할 때 받은 size만큼의 점수를 입력함을 메시지로 출력해주며 이에 맞게 사용자가 점수를 입력할 시 scores 배열에 size개의 입력받은 값들을 size만큼 for문을 돌려 scores에 차례대로 저장하도록 한다.

- isOver60() 함수

60점 이상과 미만의 점수를 구분하는 함수로, 60점 이상인 학생이 몇 명인지 return scores[index] > 60;를 통해 count한다.(60점 이상일 경우 >=를 쓰는 것이 맞으나 문제의 실행결과에서 4명으로 표시해주었기에 >를 사용했다) scores[index]가 60보다 크면 true를 리턴하고 그렇지 않으면 false를 리턴한다. 60점 이상일 경우 countPass() 함수에서 if문을 사용했기 때문에 true이면 카운트를 하나씩 올린다.

12-3.

countPass() 함수에서 Dept dept를 Dept& dept로 변경하여 즉, 객체를 복사 생성하지 않고 기존 객체의 주소를 참조하여 기존 객체를 사용하는 방식을 이용하여 문제를 해결했다.

객체지향프로그래밍[02]
202304035 김유진
202104354 지현준

#아이디어 평가

12-1.

(1) scores에 대해 원본 객체와 복사된 객체에 대해 동적으로 새로운 독립된 배열을 할당하여 소멸된 메모리를 다시 소멸하는 오류를 피할 수 있었다.

(2) cin은 공백 문자로 구분하여 값을 입력받아 저장하므로 read() 함수에서 cin 입력 스트림 객체를 사용했다. 또한 for문을 사용하여 이에 따라 주어진 실행 결과처럼 size개의 값들을 한줄로 띄어쓰기로 구분해 입력받을 수 있었다.

(3) delete를 사용하여 할당받은 동적 메모리를 반환하는 소멸자 함수를 구현했다. 이를 통해 메모리 누수가 발생하지 않게 할 수 있었다.

12-3.

객체가 값에 의해 호출되어 복사 생성자를 호출하지 않도록 참조 변수를 사용해 기존 객체를 그대로 쓸 수 있도록 했다. 그 결과로 반환된 메모리를 다시 반환하는 오류를 피할 수 있었다.

#문제를 해결한 키 아이디어 또는 알고리즘 설명

12-1.문제를 해결한 키 아이디어는 복사 생성자 부분에 대한 알고리즘이라고 생각한다. 디폴트 복사 생성자에 의한 얕은 복사에 대한 문제(반환된 메모리를 또 반환하는 문제)를 피하기 위해 깊은 복사 생성자를 구현해야 했다. 새롭게 생성된 객체의 멤버 변수인 this->size에 원본 객체의 값인 dept.size를 대입하여 원본 객체 dept의 size 값을 복사했다. 또한 복사한 size만큼의 새로운 동적 메모리로 배열을 할당하여 같은 메모리를 공유하는 일이 발생하지 않도록 했다. 새로운 메모리를 할당했으나 그 배열 안의 값들은 복사되지 않은 상태이므로 반복문을 size만큼 돌리도록 작성하여 원본 객체 dept의 배열 내용을 새로운 객체의 배열인 this->scores에 복사했다. 그 결과로 같은 배열 내용을 가지면서도 독립된 메모리를 가지도록 하여 반환된 것에 대한 반환으로 인한 충돌이 일어나지 않게 했다.

12-3.문제를 해결한 키 아이디어는 객체에 대한 참조이다. 전체적으로 보면 int n=CountPass(com); 이 코드에서 '값에 의한 호출'로 객체가 전달되므로 함수의 매개 변수 객체가 생성될 때 복사 생성자가 자동으로 호출된다. 그러나 구현한 복사 생성자를 제거한 상태이므로, 묵시적으로 컴파일러가 삽입한 디폴트 복사 생성자가 실행된다. 디폴트 복사 생성자의 경우, 얕은 복사를 수행하며 Dept의 scores가 동적 메모리 할당이 된다. 따라서 디폴트 복사 생성자가 생성하는 객체의 scores와 매개변수로 들어가는 객체인 com의 scores가 같은 메모리를 공유하게 된다. countPass의 값이 반환되며 소멸자가 실행될 때 com의 메모리가 반환되었다. 프로그램이 종료될 때, com의 소멸자가 실행되며 이미 반환한 메모리를 반환하려고 하기 때문에 실행 시간 오류가 발생하고 프로그램이 비정상적으로 종료하게 된다. 그렇기 때문에 countPass가 객체를 매개변수로 받지 않으면 되고 그 결과, 복사 생성자가 호출되지 않는다. countPass가 Dept에 대한 객체를 참조로 받으면 된다. 참조로 받기 때문에 com 원본에 접근할 수 있게 되어 복사 생성자가 없어도 실행 오류가 발생하지 않는다.