문제 12-(1)

main()의 실행 결과가 나오도록 하기 위해 우선 복사 생성자와 소멸자를 구현해야 한다. 또, read()와 isOver(int index) 등의 클래스 내부 함수를 구현해야 한다.

복사 생성자를 구현할 때 매개변수를 참조변수로 할당 받고 새로운 메모리에 똑같은 메모리 값을 복사 해주어 별도의 공간을 가지도록 하였다. 소멸자는 호출될 때 점수 배열에 들어있는 모든 값들을 지워주도록 delete[] 하였다.

read() 함수를 구현 할 때, stringstream을 사용하였다. <sstream> 헤더 파일에 포함되어 있는 것으로 이를 이용해 입력받은 점수들을 분리할 수 있도록 했다. getline을 통해 입력받은 한 줄을 모두 받아오고 이를 stringstream으로 변환하여 스페이스로 구분된 값들을 scores[] 배열에 저장되도록 하였다.

isOver 함수는 한 줄의 코드로 return 값을 반환할 수 있다. 인자값으로 받아온 index를 통한 배열의 값과 60을 비교하여 그 결과를 return하는 것이다. 만약 해당 값이 60보다 크거나 같다면 true가 리턴될 것이고, 그렇지 않다면 false가 리턴될 것이다. 이를 통해 comPass함수 조건문의 조건식으로 사용할 수 있다.

문제 해결의 키는 stringstream이다. 이것을 알기 전까지는 점수들을 한 줄로 입력하고 각각을 어떻게 한 번에 입력받을 수 있을지 고민이었다. 이 간단한 선언을 통해 쉽게 문제를 해결할 수 있었다. 또한 isOver함수에서 if-else 조건문을 통해 결과를 도출했었는데 이를 간단하게 조건식만을 리턴하는 방법으로 바꾸어 코드를 간략하게 만들었다. 결과는 같게 출력되지만 코드의 효율성을 더욱 높이는 방법이었다.

문제 12-(3)

복사 생성자를 없앤다면 객체를 복사하지 않고 원본 객체를 사용하게 하는 방법이 있다. 그 방법으로는 comPass함수에서 기존에 Dept 타입의 객체 매개변수를 받은 반면, 변경한 코드에서는 Dept 타입의 참조 매개변수를 받는 것이다. 이렇게 한다면 원본의 정보가 바뀌겠지만 복사 생성자가 없을 때 오류발생을 막을 수 있다. 객체 매개변수가 아닌 참조 매개변수 즉, 원본의 별명으로 정보를 입력받기 때문에 오류가 발생하지 않고 원본에 접근할 수 있다. 따라서 실행 결과화면에 이전과 같이 출력된다. 하지만 내부 메모리상으로는 복사된 데이터 값을 사용했던 전과 달리 원본 데이터 값에 접근하게 된 것으로 차이점이 생겼다.