# 객체지향 프로그래밍 3차 과제

---

## 문제

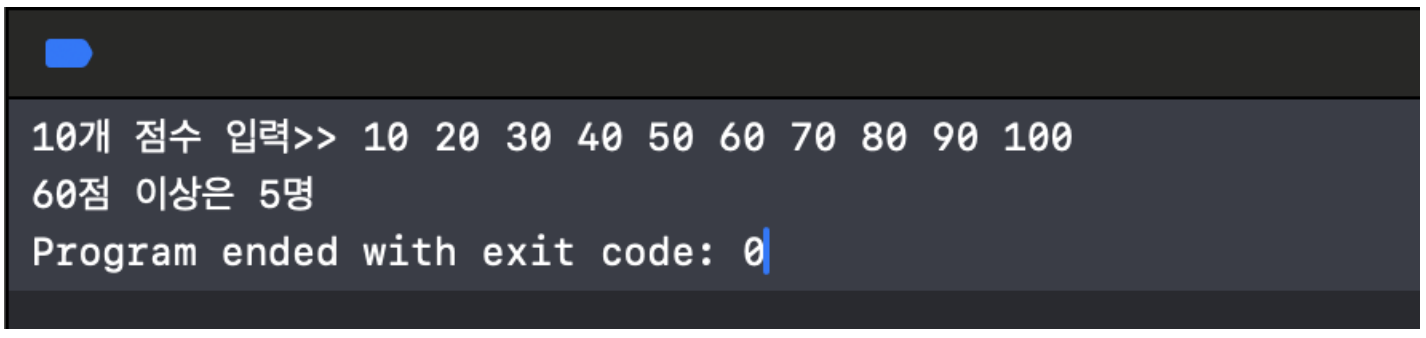
```
public:
    Dept(int size) {// 생성자
        this-›size = size;
        scores = new int[size];
}
Dept(const Dept& dept); 1 // 복사 생성자
~Dept(); // 소멸자
int getSize() { return size; }
void read(); 1 // size 만큼 키보드에서 정수를 읽어 scores 배열에 저장
bool isOver60(int index); 1 // index의 학생의 성적이 60보다 크면 true 리턴
};
int countpass(Dept dept) { 1 // dept 학과에 60점 이상으로 통과하는 학생의 수 리턴
    int count = 0;
    for (int i = 0; i ‹ dept-getSize(); i++) {
        if (dept.isOver60(i)) count++;
    }
return count;
}

int main() {
Dept com(10); // 총 10명이 있는 학과
com com.read(): // 총 10명의 학생들의 성적을 키보드로부터 읽어 scores 배열에 저장
int n = countPass(Com); 1 // com 학과에 60점 이상으로 통과한 학생의 수를 리턴
cout << "60점 이상은 << n << "명"
```

**(1)main()의 실행 결과가 다음과 같이 되도록 Dept 클래스에 멤버들을 모두 구현하고, 전체 프로그램을 완성하라**

```
10개 점수 입력>> 10 20 30 40 50 60 70 80 90 100
60점 이상은 4명
```

## 소스 수행 결과 화면 캡쳐




## 소스코드

Dept.h

```
#ifndef DEPT_H
#define DEPT_H

class Dept {
private:
    int size; // 학생 수
    int* scores; // 학생 성적 배열

public:
    Dept(int size); // 생성자
    ~Dept(); // 소멸자
    int getSize(); // 학생 수 반환
    void read(); // 성적 입력
    bool isOver60(int index); // 60점 이상인지 확인
};

#endif // DEPT_H
```

Dept.cpp

```
#include <iostream>
#include "Dept.h"
using namespace std;

// 생성자: 학생 수를 받아 scores 배열을 동적 할당
Dept::Dept(int size) {
    this->size = size;
    scores = new int[size]; // 동적 메모리 할당
}

// 소멸자: 동적 할당된 메모리를 해제
Dept::~Dept() {
    delete[] scores; // 메모리 해제
}

// 학생 수 반환
int Dept::getSize() {
    return size;
}

// 학생 성적 입력
void Dept::read() {
    cout << size << "개 점수 입력>> ";
    for (int i = 0; i < size; i++) {
        cin >> scores[i]; // 성적 입력
    }
}

// 특정 인덱스의 성적이 60 이상인지 확인
bool Dept::isOver60(int index) {
    return scores[index] >= 60; // 60점 이상이면 true 리턴
}
```

main.cpp

```
#include <iostream>
#include "Dept.h"
using namespace std;
```

```
// 60점 이상으로 통과하는 학생의 수를 계산
int countPass(Dept& dept) {
    int count = 0;
    for (int i = 0; i < dept.getSize(); i++) {
        if (dept.isOver60(i)) count++; // 통과한 학생 수 증가
    }
    return count;
}

int main() {
    Dept com(10); // 총 10명이 있는 학과
    com.read(); // 학생들의 성적 입력
    int n = countPass(com); // 60점 이상으로 통과한 학생의 수 리턴
    cout << "60점 이상은 " << n << "명" << endl; // 결과 출력
    return 0; // 프로그램 종료
}
```

## 문제 정의

학생 수와 성적을 동적으로 관리할 수 있어야 되고, 입력받은 성적을 기반으로 학생이 60점 이상인지 확인하는 기능을 포함해야되는 문제.

## 문제 해결 방법

- **클래스 설계**: `Dept` 클래스를 설계하여 학생 수와 성적을 관리한다.
  - 클래스의 멤버 변수로 학생 수와 동적 배열을 선언한다.
  - 생성자와 소멸자를 통해 메모리를 관리한다.
- **성적 입력**: `read()` 메서드를 구현하여 학생의 성적을 키보드로부터 입력받아 배열에 저장한다.
- **성적 확인**: `isOver60()` 메서드를 구현하여 특정 인덱스의 학생 성적이 60점 이상인지 확인한다.
- **60점 이상인 학생 수 계산**: `countPass()` 함수를 구현하여 60점 이상인 학생의 수를 세고 반환한다.
- **메인 함수**: `Dept` 객체를 생성하고 성적을 입력받은 후, 통과한 학생 수를 출력한다.

## 알고리즘 설명

### Dept 클래스

- **생성자**: 입력받은 학생 수에 따라 동적으로 성적 배열을 생성한다.
- **소멸자**: 할당된 메모리를 해제한다.
- **read()**: 학생의 성적을 입력받아 `scores` 배열에 저장한다.
- **isOver60()**: 특정 인덱스의 성적을 확인하여 60점 이상일 경우 `true`를 반환한다.

### countPass()

- `Dept` 객체를 인자로 받아, 반복문을 통해 각 학생의 성적을 확인하고 60점 이상인 경우 카운트하여 최종적으로 그 수를 반환한다.

## 흐름 설명

- **학생 수 입력**: 사용자가 입력한 학생 수에 따라 `Dept` 객체를 생성
- **성적 입력**: `read()` 메서드를 호출하여 사용자로부터 각 학생의 성적을 입력받고, 이를 `scores` 배열에 저장
- **성적 확인**: `countPass()` 함수가 호출되어 60점 이상의 학생 수를 카운트
- **결과 출력 및 메인함수 종료**

(3)

Dept 클래스에 복사 생성자를 제거하라. 복사 생성자가 없는 상황에서도 실행오류가 발생하지 않게 하려면 어느 부분을 수정하면 될까? 극히 일부분의 수정으로 해결된다. 코드를 수정해보라.

```
Dept(const Dept& dept); // 복사 생성자
```

**해당부분을 제거합니다.**

그리고 countPass의 매개변수에 참조로 전달하는 방식으로 처리합니다.

**기존 코드**

```
int countPass(Dept dept) { // dept 학과에 60점 이상으로 통과하는 학생의 수 리턴
```

**수정 코드**

```
int countPass(Dept& dept) { // 참조로 전달하여 복사 생성자 사용 안 함
```