소스 구현 설명 - 강부경, 황호태

문제 정의
: 10개의 점수를 입력하고 60점 이상으로 통과한 학생 수를 리턴하는 것이 문제이다.

문제를 해결한키 아이디어 또는 알고리즘 설명 :

이번 프로그램은 난이도가 그렇게 어렵지 않았다.
다만, 복사 생성자라는 개념이 들어갔는데, 복사하는 부분에서 막막하였고, 결국 해결하였다.

```
Dept::Dept(const Dept& dept) {
	this->size = dept.size;
	scores = new int[size];
	for (int i = 0; i < size; i++) {
		scores[i] = dept.scores[i];
	}
}
```

size는 그대로 size를 복사하면 된다.
scores를 공간할당을 또 해야한다는 사실을 알았고,

for문을 통해서 배열의 내용도 복사해야한다는 것이 문제 해결 키였다.
복사는 기존 socres의 배열에다 복사한 dept의 scores의 값을 복사하였다.

1. 복사 생성자 없애기 전(코드)

Dept.h

```
#pragma once
class Dept {
	int size;
	int* scores;
public:
	Dept(int size) {
		this->size = size;
		scores = new int[size];
	}
	Dept(const Dept& dept);
	~Dept();
	int getsize() { return size; }
	void read();
	bool isOver60(int index);
};
```

main.cpp

```
#include <iostream>

using namespace std;

#include "Dept.h"

int countPass(Dept dept) {
	int count = 0;
	for (int i = 0; i < dept.getsize(); i++) {
		if (dept.isOver60(i))
			count++;
	}
	return count;
}

int main() {
	Dept com(10);
	com.read();
	int n = countPass(com);
	cout << "60점 이상은 " << n << "명";
}
```

Dept.cpp

```
#include <iostream>

using namespace std;

#include "Dept.h"

Dept::Dept(const Dept& dept) {
	this->size = dept.size;
	scores = new int[size];
	for (int i = 0; i < size; i++) {
		scores[i] = dept.scores[i];
	}
}
```

main.cpp와 Dept.h는 교재에 나와있었고,

(1)번의 내용인 클래스에 맴버들을 모두 구현한 Dept.cpp를 설명하자면 다음과 같다.

1. Dept(const Dept& dept)

우선 Dept.h에 있는 것을 순서대로 구현하였다.
'Dept(const Dept& dept);' 복사생성자이므로 Dept(int size)의 내용을 깊은 복사할 필요가 있었다.

따라서 this size = dept이라는 객체의 size로 받고
scores도 따로 공간을 할당하였다.

또한 scores의 내용을 복사하기 위해
for문을 써서 배열의 내용을 복사하였다.

```
void Dept::read() {
        cout << "10개 점수 입력 >>";
        for (int i = 0; i < size; i++) {
                cin >> scores[i];
        }
}

bool Dept::isOver60(int index) {
        return scores[index] >= 60;
}

Dept::~Dept() {
  delete[]scores;
}
```

2. ~Dept()

'~Dept()'의 소멸자는 맨 마지막에 위치하였다.
배열을 할당받았기 때문에 위와 같은 delete 형태를 취했다.

3. void read()

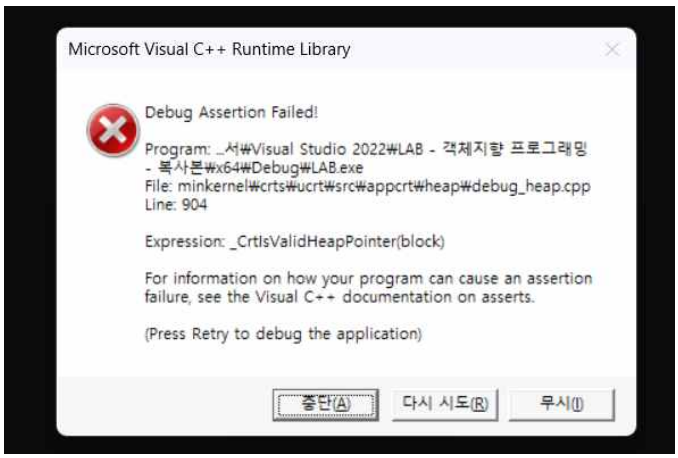
'void read();'는 'size만큼 키보드에서 정수를 읽어 scores 배열에 저장'을 구현해야 하므로 직관적으로 for문을 써야겠다고 생각했다. 범위는 size만큼, 키보드에서 정수를 읽어야 하므로 'cin'이라는 함수가 떠올랐다. 그리고 문제 실행결과에서 '10개의 점수 입력>>'이라는 문장을 보아서 'cout'이라는 함수도 떠올랐다.

따라서 read()는 직관적으로 바로 구현할 수 있었다.

4. bool Over60(int index)

index가 scores배열의 index임을 바로 알 수 있어서 바로 return에 'scores[index] >= 60;'를 넣었다.

다음은 복사자 생성자를 제거해보고, 오류를 분석해보았다.



복사자를 제거하였더니 다음과 같은 오류가 떴다.

이와 같은 오류가 왜 발생하였는지 고뇌한 결과,

main 함수에 있는 'int countPass(Dept dept){...}'가 눈에 걸렸다.

매개 변수 객체의 생성자는 실행되지 않고, 소멸자만 실행되기 때문에 비대칭 구조를 띤다고 교재에서 설명하였고,

현재 복사 생성자를 정의하지 않아서, 지금 현재 'int count()'안에 매개 변수 객체는 컴파일러에 의해 자동으로 삽입된 '디폴트 복사 생성자'이다.

얕은 복사이다.

따라서 Dept com과 Dept dept의 메모리를 반환하는 과정에서 실행 시간 오류가 발생하고 프로그램이 비정상 종료가 되는 것이다.

그렇다면 원인을 알았으니,

(3) 실행 오류가 발생하지 않게 하려면 어느 부분을 수정하면 될까?

정말 걸리는 것은 '극히 일부분의 수정으로 해결된다.'라는 부분이다.

처음 접근은 '기호 하나를 추가하면 될까?'이었다.
'*'이나 '&'이었다.

얕은 복사이기 때문에, 주소의 값을 어떻게 어떻게하면 되지 않을까라는 접근이었다.
하지만 포인터의 이해도 부족인지는 몰라도 문제는 해결되지 않았다.

두 번째 접근은 문제의 매개 변수 객체 Dept dept을 없애버리자는 생각을 하였다.

결국 문제의 매개 변수 객체를 없애서 소스를 구현하였다.

```
class Dept {
private:
	int size;
	int* scores;

public:
	Dept(int size);
	~Dept();
	int getSize() const;
	void read();
	bool isOver60(int index) const;
};
```

```
Dept::Dept(int size) {
	this->size = size;
	scores = new int[size];
}
```