

소스 구현 설명

202304178 구서연

문제 정의 :

복사 생성자가 언제 호출이 되고, 깊은 복사를 위해서는 복사 생성자에 어떤 내용들이 들어가 있어야 하는지를 정확하게 알아야 하는 문제이다.

문제 해결 방법 :

문제 12-(1)번 아이디어 및 알고리즘 설명

1. 선언부 / 구현부 / main으로 나눈다.
2. 복사 생성자로 깊은 복사를 한다.
3. 소멸자에서 동적 할당한 배열을 반납해준다.
4. (문제 오류 발견 및 수정) 나머지 함수들 구현한다.

1. 선언부 / 구현부 / main으로 나눈다.

(1)번 문제와 (3)번 문제를 같은 프로젝트에 만들었기 때문에, 다른 파일이더라도 (1)번의 클래스 Dept는 편의상 Dept1로, (3)번의 클래스 Dept는 편의상 Dept3으로 구분해주었다.

그리고 클래스 Dept1을 선언부 Dept1.h와 구현부 Dept1.cpp로 나누고 main함수가 있는 Main1.cpp를 만들어 주었다.

Dept1.h에서 멤버 변수들과 모든 생성자, 복사 생성자, 소멸자, 함수들을 선언해준다. 그리고 Dept1.cpp에서 구현하는 내용들을 써준다. 이 내용은 핵심 아이디어 2, 3, 4번에서 쓰겠다. 그리고 Main1.cpp에 main함수를 구현한다. main함수는 이미 문제에서 모두 주어졌기 때문에 그대로 쓰면 된다.

조건 컴파일 문을 추가하여, dept1.h가 여러 번 include 되어도 문제가 없도록 해준다.

Dept1.h

```
#ifndef DEPT1_H
#define DEPT1_H

class Dept1 {
    int size;
    int* scores;
public:
    Dept1(int size);
    Dept1(const Dept1& dept);
    ~Dept1();
    int getSize();
    void read();
    bool isOver60(int index);
};

#endif
```

2. 복사 생성자를 구현해준다.

구현부 dept1.cpp에서 복사 생성자를 구현해주어야 한다. 이때, 왜 복사 생성자를 만드느냐를 알아야 한다.

```
class Dept1 {
    int size;
    int* scores;

    Dept1::Dept1(int size) {
        this->size = size;
        scores = new int[size];
    }
}
```

우리는 클래스에서 포인터를 만들어줬다. 그리고 이 포인터가 동적 할당한 정수형 배열을 가리키게 했다. 이때, 얇은 복사를 할 경우, 포인터만 복사되고 동적 할당한 정수형 배열은 복사가 되지 않는다. 그럼 소멸자로 삭제를 시킬 때 문제가 생기기 때문에, 직접 깊은 복사를 하게끔 복사 생성자를 만들어 주어야 한다.

```
Dept1::Dept1(const Dept1& dept)
{
    this->size = dept.size;
    this->scores = new int[size];
    for (int i = 0; i < size; i++) {
        scores[i] = dept.scores[i];
    }
}
```

구현부 dept1.cpp에서 깊은 복사하는 복사 생성자를 구현해야 한다. dept의 size를 복사하여 자신의 size에 넣고, 자신의 scores가 size만큼 동적 할당한 int형 배열을 가리키게 한 후, 그 배열에 dept의 int형 배열의 값들을 각각 넣는다.

이런 방식으로 하면, 포인터뿐만 아니라, 포인터가 가리키고 있는 동적할당 된 그곳도 복사해서 메모리가 별도로 잡히기 때문에 소멸될 때 문제가 없다.

3. 소멸자에서 필요한 내용들을 정리해준다.

이제 소멸자에서 필요한 것들을 정리해준다. 가장 필요한 것은 동적 할당된 배열을 다시 반납하는 것이다.

```
Dept1::~~Dept1() {
    delete [] scores;
}
```

소멸자에서 `delete [] scores;` 하면 포인터 scores는 그대로 유지된 상태로, scores가 가리키고 있는 동적할당 된 그 배열이 반납된다. 여기서 `delete scores;`가 아닌 `delete [] scores;`를 해주어 동적 할당한 배열을 해제해 주어야 한다.

4. 나머지 함수들 구현한다.

문제에서 구현해주지 않은 read() 함수와 isOver60 (int index)함수를 마저 구현해주어야 한다. 이들 역시 구현부 dept1.cpp에서 구현해주도록 한다.

```
void Dept1::read()
```

```
void Dept1::read()
{
    cout << size << "개 점수 입력>> ";
    for (int i = 0; i < size; i++)
    {
        cin >> scores[i];
    }
}
```

read함수는 size만큼 키보드에서 정수를 읽어 scores 배열에 저장하는 함수라고 문제에서 제시해주었다. 일단 `#include <iostream>`와 `using namespace std;`를 해주고 난 후에 cout으로 몇 개의 점수를 입력해야 하는지 알려준다. 그리고 for문으로 cin을 사용하여 scores[i]에 입력받는 수를 넣어준다. void이니 반환 값이 필요가 없다.

```
bool Dept1::isOver60(int index)
```

```
bool Dept1::isOver60(int index)
{
    if (scores[index] >= 60)
        return true;
    else
        return false;
}
```

(문제오류 발견) 문제에서 함수에 대한 설명은 index의 학생의 성적이 60보다 크면 true 아니면 false를 리턴하는 함수라고 되어 있지만 밑에 출력 결과에서는 60점 이상이라고 되어 있기 때문에 **문제 오류**다. 따라서 나는 60점 이상일 경우라고 판단하고 문제풀이를 진행했다. if를 사용하여 scores[index]가 60 이상이면 true를 반환하게 한다. 그 외는 false를 반환한다.

문제 12-(3)번 아이디어 및 알고리즘 설명

핵심 아이디어 : 복사 생성자를 제거하기 위해 복사 생성자를 호출하던 함수를 참조에 의한 호출로 객체를 복사하지 않고, 원본 객체를 참조.

1. 복사 생성자 제거

문제 12-(3)번의 조건에서 복사 생성자를 제거하라고 했다. 만약, 복사 생성자를 제거하면 컴퓨터가 디폴트 복사 생성자를 삽입해주고, 얇은 복사가 일어나서 소멸자 호출 시 오류가 난다.

2. 원인 분석

복사 생성자를 제거하는 것이 문제 12-(3)번의 조건이므로, 나는 복사 생성자를 만들지 못하고, 디폴트로 삽입되는 복사 생성자는 건들지를 못한다. 따라서 복사 생성자가 애초에 호출되지 않는 상황을 만드는 것이 오류가 나지 않게 하는 방법이다. 그렇다면 복사 생성자는 언제 호출되는 것일까?

`int countPass(Dept3 dept)`를 보면 main함수에서 `countPass(com)`로 함수를 호출하면서 com 객체를 dept가 복사하며 복사 생성자가 호출되는 것이다. 그럼 복사 생성자가 호출되지 않도록 `int countPass(Dept3& dept)`를 수정해주면 된다.

3. 참조를 이용하여 오류 없게 수정

일단 countPass함수의 목적부터 보자. dept에서 60점이 이상인 학생 수를 세는 함수다. 그리고 dept는 12-(1)번에서는 com 객체를 깊은 복사하여 완전히 내용물은 그대로고 메모리가 다른 것일 뿐이었다. 그렇다면 결국 우리가 원하는 것은 인자인 객체의 성적이 60점 이상인 학생 수를 알아내는 것이고, countPass에서 com 객체를 참조하면 com에서 60점 이상인 학생 수를 셀 수 있고, dept는 복사 생성자를 호출하지 않게 된다.

```
Dept3 com(10);

int countPass(Dept3& dept)
{
    int count = 0;
    for (int i = 0; i < dept.getSize(); i++)
    {
        if (dept.isOver60(i))
            count++;
    }
    return count;
}
```

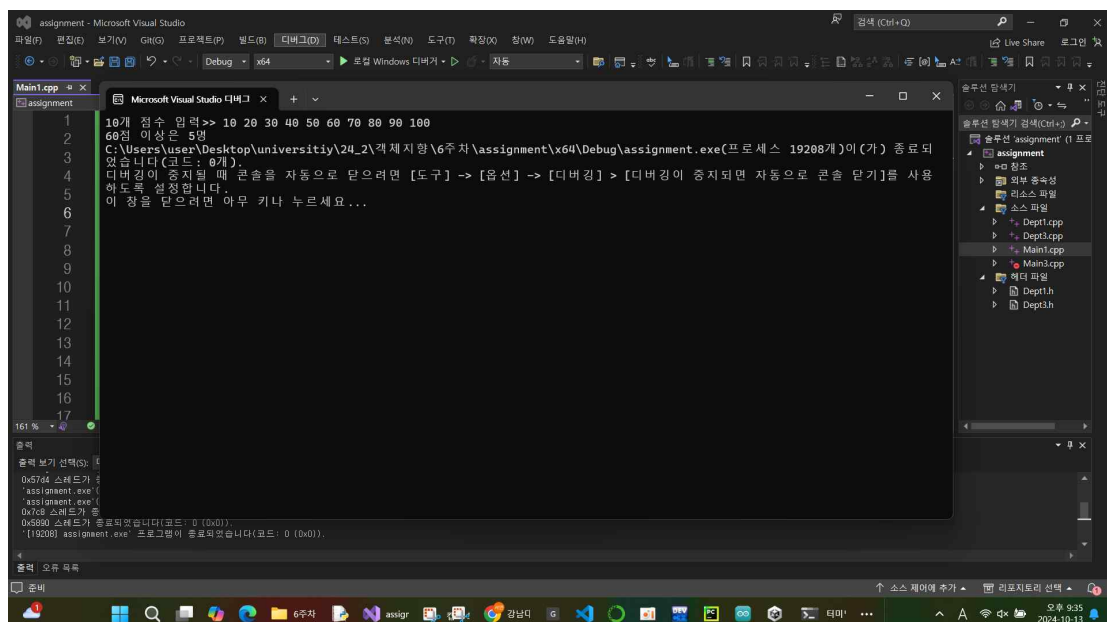
그럼 얕은 복사로 인해 소멸자에서 이미 삭제된 내용을 또 삭제하려는 오류가 발생하지 않게 된다.

아이디어 평가 :

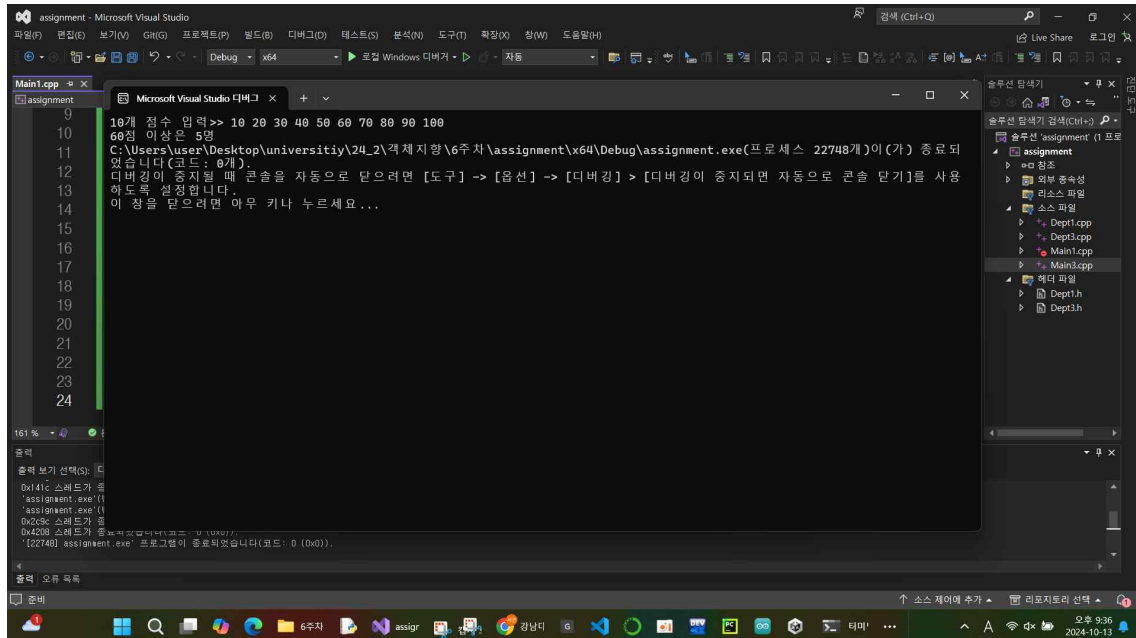
문제 12-(1)번 핵심 아이디어 : 복사 생성자로 깊은 복사를 하고, 소멸자에서 동적 할당한 배열 반납한다.

문제 12-(3)번 핵심 아이디어 : 복사 생성자를 제거하기 위해 복사 생성자를 호출하던 함수를 참조에 의한 호출로 객체를 복사하지 않고, 원본 객체를 참조한다.

문제 12-(1)번 결과 :



문제 12-(3)번 결과 :



```
10개 점수 입력 >> 10 20 30 40 50 60 70 80 90 100
60점 이상은 5명
C:\Users\user\Desktop\university\24_2\객체지향\6주차\assignment\x64\Debug\assignment.exe(프로세스 22748개)이(가) 종료되
었습니다(코드: 0).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

출력

출력 보기 선택(S):

- 0x141c: 스택드가...
- 'assignment.exe' (1)
- 'assignment.exe' (1)
- 0x2c9c: 스택드가...
- 0x4200: 스택드가...
- '[22748] assignment.exe' 프로세스가 종료되었습니다(코드: 0 (0x0)).

출력 오류 목록

준비

소스 제어에 추가

리포지토리 선택

오류 9:36
2024-10-13