객체지향설계 #Week05

제출일 : 20.10.04

학번 이름 : 201902721 유찬희

GitHub 주소 : https://github.com/HanCiHu/OOP

homework01 .

```
15    Animal(Animal &a) {//복사 생성자
16        age = a.age;
17        name = new char[strlen(a.name) + 1];
18        strcpy(name, a.name);
19    }
```

<< homework01.cpp 복사생성사 함수 코드

```
➜  week5 git:(master) ✗ g++ homework_05_01.cpp
➜  week5 git:(master) ✗ ./a.out
Name: Brown Age: 22
Name: Jenny Age: 10
```

<< 결과창

- 원래의 코드대로 실행을 하면 Animal 클래스의 name변수가 포인터 변수이기 때문에 복 복사생성자 실행될 때 포인터가 복사가 되어서 A가 바뀌면 B도 함께 바뀌게 됩니다.
- 위의 코드와 같이 코드를 고치게 되면 포인터를 복사하는게 아니라 name변수의 값만 복사하는 것이기 때문에 A가 이후에 바뀌게 되더라도 B는 바뀌지 않게 됩니다.

homework02 .

```
13    int foo(){
14        i = 3;
15        A *ob = new A();
16        delete ob;
17        return i;
18    }
```

<< homework02.cpp foo 함수 코드

```
➜  week5 git:(master) ✗ g++ homework_05_02.cpp
➜  week5 git:(master) ✗ ./a.out
10
```

<< 결과창

- 원래의 코드대로 실행하면 foo함수에서 A class의 동적할당이 이루어지지 않아서 소멸자가 호출되지 않아 3이 출력이 됩니다.
- 따라서 A class의 소멸자를 호출하기 위해 A를 동적할당 해주고 delete를 해줘서 소멸자를 호출해 10이 출력되게 하였습니다.

homework03 .

```
→  week5 git:(master) ✗ g++ example01.cpp -std=c++11
→  week5 git:(master) ✗ ./a.out
Name: Jenny Age: 10
-----1st push-----
Copy constructor is invoked!!
-----2nd push-----
Copy constructor is invoked!!
Copy constructor is invoked!!
Destructor!!
-----3rd push-----
Copy constructor is invoked!!
Copy constructor is invoked!!
Copy constructor is invoked!!
Destructor!!
Destructor!!
-----4th push-----
Copy constructor is invoked!!
-----5th push-----
Copy constructor is invoked!!
Copy constructor is invoked!!
Copy constructor is invoked!!
Copy constructor is invoked!!
Copy constructor is invoked!!
Destructor!!
Destructor!!
Destructor!!
Destructor!!
Name: Jenny Age: 10
Name: Jenny Age: 10
Name: Jenny Age: 10
Name: Jenny Age: 10
Name: Jenny Age: 10
Name: Jenny Age: 10
Destructor!!
Destructor!!
Destructor!!
Destructor!!
Destructor!!
Destructor!!
```

```
→  week5 git:(master) ✗ g++ example02.cpp -std=c++11
→  week5 git:(master) ✗ ./a.out
Name: Jenny Age: 10
-----1st push-----
Copy constructor is invoked!!
-----2nd push-----
Copy constructor is invoked!!
Move constructor is invoked!!
Destructor!!
-----3rd push-----
Copy constructor is invoked!!
Move constructor is invoked!!
Move constructor is invoked!!
Destructor!!
Destructor!!
-----4th push-----
Copy constructor is invoked!!
-----5th push-----
Copy constructor is invoked!!
Move constructor is invoked!!
Move constructor is invoked!!
Move constructor is invoked!!
Move constructor is invoked!!
Destructor!!
Destructor!!
Destructor!!
Destructor!!
Name: Jenny Age: 10
Name: Jenny Age: 10
Name: Jenny Age: 10
Name: Jenny Age: 10
Name: Jenny Age: 10
Name: Jenny Age: 10
Destructor!!
Destructor!!
Destructor!!
Destructor!!
Destructor!!
Destructor!!
```

example01.cpp & example02.cpp 실행결과

- 실행결과는 같아보이지만 example02에서는 이동생성자가 실행되었기 때문에 코드의 실행속도가 example01보다 더 빠를 것 같습니다.
- 벡터의 크기가 부족해서 벡터의 크기를 늘리고 원래의 벡터에 있던 값들을 크기를 늘린 벡터에 옮겨주는 과정에서 복사생성자가 실행이 되는데 이때 name을 동적할당을 계속 해주면서 성능이 저하되게 됩니다.
- 이동생성자를 이용하면 불필요한 동적할당을 없애줘 상대적으로 이동생성자를 사용하지 않았을 때보다 좋은 성능을 이끌어낼수 있습니다.
- 참고 사이트 : https://modoocode.com/227