STL - 1주 2일 클래스 복습

Wulong

✔ **자원을 할당하는 클래스** - 동적으로 할당되는 데이터를 갖는 Model 클래스를 만들어 보자. 단계별로 코드를 추가해 가며 클래스를 완성해 보자.

```
class Model {
        char* data;             // 동적할당한 메모리의 시작번지
        size_t size;            // 메모리의 크기
};

int main()
{
        Model a;                // 기본 생성
}
```

[문제 1] 위의 코드가 문제없이 컴파일되도록 프로그램하라.
- 프로그램이 실행되는 최소한의 코드를 작성하라.

[문제 2] 객체의 동작을 관찰할 수 있는 코드를 추가한 후 실행결과를 관찰하라.
- 이 문제에서 말하는 객체의 동작은 스페셜 함수 호출을 의미한다.
- 클래스의 스페셜 함수는 생성/소멸/복사/복사할당/이동생성/이동할당 6가지이다

```
int main()
{
        Model a;                // 크기 정보없이 기본 생성
        Model b { 1000 };       // 1000 byte 데이터
}
```

[문제 3] 변경한 main 함수가 문제없이 컴파일되도록 최소한의 코드를 작성하고 동작을 관찰하라. 객체 b의 size = 1000, data는 1000 byte의 메모리를 가리키는 포인터 값이 저장되어야 한다.

```
int main()
{
        // 생략
        Model c = b;            // 추가한 코드
}
```

[문제 4] 위와 같이 main 함수에 코드를 다시 한 줄 추가하고 프로그램을 실행시키고 어떤 스페셜 함수가 호출되는 지 확인하라.
- 객체와 객체가 할당한 메모리 그림을 그려 실행 결과를 정확하게 설명해 보자.
- 메모리 관리 면에서 프로그램이 아무런 문제없이 실행되도록 프로그램하여야 한다.

```
int main()
{
        // 생략
        a = c;                  // 추가한 코드
}
```

[문제 5] 다시 한 줄을 추가 하였다. 어떤 스페셜 함수가 호출되는지 확인하라.
- 추가한 코드가 문제없이 컴파일되도록 코드를 작성하고 동작을 관찰하자.

이제 C++에서 **복사생성이 필요한 상황**을 잘 이해하였을 것이다.

지금부터 C++11에서 추가된 **클래스 데이터의 이동**을 복습해 보자. 먼저 객체의 생성과 소멸을 더 잘 관찰하기 위해 생성되는 객체마다 고유의 번호(id)를 붙여보자.

```
int main()
{
        Model a[10];
}
```

[문제 6] Model 객체 10개를 생성하는 코드이다. 객체마다 다른 id값을 갖도록 클래스를 수정하라.
- 프로그램을 실행하면 객체의 생성과 소멸시 객체의 id가 출력되도록 하라.

✔ **이동생성과 이동생성자** - 객체가 동적으로 확보한 데이터를 더 이상 사용할 필요가 없는 상황을 생각해 보자. 이것은 객체의 생명주기가 남아있는 상태에서, 더 이상 그 객체를 사용하지 않을 것이라고 단정할 수 있는 상황을 의미한다. (이러한 상황은 대개의 경우 프로그래머가 직접 코딩해야 하는 상황이 아니라는 점을 지적한다.) 이때, 이미 만들어진 객체를 이용해서 다른 객체를 생성하여야 할 필요가 있다면(이것은 기존 복사생성을 말한다), 새로 만들 객체는 기존 객체가 확보한 동적메모리의 내용을 복사하기 위하여 메모리를 할당한 후 일일이 내용을 복사하지 않고(이것을 깊은 복사라 한다) 메모리의 포인터만 복사하면 충분할 것이다. 기존 객체의 경우 확보한 메모리를 명시적으로 delete 하지 않고 포인터값을 nullptr로 만들어 주기만 하면 된다. 이와 같은 동작을 객체의 이동생성(move construction)이라 한다. 이 동작은 이동생성자(move constructor)에서 수행된다.

```
int main()
{
        Model a { 1000 };
        Model b = move( a );
}
```

[문제 7] 이 코드를 다음 조건에 맞게 실행되도록 하라.

- main이 끝나기 직전
  객체 a는 size = 0, data = nullptr
  객체 b는 size = 1000, data는 a가 할당한 1000 바이트 메모리의 시작번지를 저장하고 있어야 한다.

[문제 7]을 해결하려면 Model class에 다음과 같은 이동생성자를 추가해야 한다.

```
Model( Model&& other) : data(nullptr), size(other.size) {
        data = other.data;                  // data 주소만 가져온다.

        other.data = nullptr;
        other.size = 0;

        // 관찰 메시지와 id를 출력하도록 하자
}
```

[문제 8] 이동생성자를 추가한 후 [문제 6]의 객체 a와 b의 메모리 상황을 그림으로 설명하라.

위의 예에서 이동생성시 사용한 편의함수 move()의 역할은 객체의 타입을 &&로 만들어 주는 것이다.

```
Model b = move( a );
// 위의 코드는 아래와 동일하다.
// Model b = static_cast<Model&&>(a);
```

[문제 9] 클래스에 이동할당연산자를 추가하고 이동할당하는 코드를 호출하라.

- **이동생성자**는 프로그래머가 명시적으로 사용(호출)하는 것이 아니고 STL 컨테이너에서 내부 데이터를 관리하는 용도로 사용된다. 따라서 STL 컨테이너를 사용하는 프로그래머는 자기가 만든 class에서 메모리를 동적할당한다면 C++11에 새로 도입된 이동 의미론(move semantic)을 반드시 구현하여 프로그램의 성능이 향상되도록 할 책임이 있다는 것을 생각할 수 있어야 한다.

[문제 10] 이동의미론을 구현하혀면 프로그래머는 무엇을 하여야 하는가?

- 이번 학기 강의에서 STL 컨테이너 vector를 사용하여 이동의미론이 어떻게 사용되는 지 확인할 것이다.

- 이 문서의 내용을 이해하기 어렵다면 매우 열심히 복습해야 합니다.
- 처음 듣는 이야기였다면 C++을 먼저 공부하고 수강해야 합니다.