2023/02/23

(1) 직렬화(serialization)
CObject 클래스는 객체의 직렬화(serialization) 기능을 제공한다. 객체의 직렬화란 객체의 상태를 파일이나 메모리 버퍼에 저장하거나 불러오는 것을 말하며, 일반적으로 MFC 애플리케이션에서 사용자 인터페이스의 동적 생성(ex 대화 상자나 프레임의 동적 생성)에 유용하게 사용된다.

- Serialize() 함수를 사용하는 이유
: CObject 클래스에서 **객체의 직렬화 기능을 구현**하기 위해서는 Serialize() 함수를 사용 필요하다.여러가지 데이터 형식을 다루기 위해 CArchive 클래스의 Serialze() 함수를 호출하여 데이터를 파일이나 메모리 버퍼에 저장하거나 불러오는 방식으로 사용한다.
- Serialize() 함수는 가상 함수이기에, 파생 클래스에서 재정의할 수 있다.
- - Serialize() 함수를 사용할 때는 CArchive 클래스에서 제공하는 <<, >> 연산자를 사용하여 멤버 변수를 저장하거나 복원

- Serialize() 함수란
: MFC 클래스의 멤버 함수 중 하나로, 객체의 상태를 저장하거나 불러올 때 호출되는 함수
- CArchive 클래스를 사용하여 객체를 파일이나 메모리 버퍼에 저장하거나 불러 올 수 있다.
(CArchive 클래스 : 데이터를 압축/암호화 기능 제공)
- Serialize() 함수 기본 형태
CMyClass* pMyObject = RUNTIME_CLASS(CMyClass)->CreateObject();

1. DECLARE_DYNCREATE 매크로를 클래스의 선언부 or 정의부에 추가하여 클래스를 구현
CMyClass 클래스의 인스턴스를 동적으로 생성
-> 이렇게 생성된 인스턴스는 CObject* 형식으로 반환되며, 이를 원래의 클래스 형식으로 캐스팅하여 사용 가능

=> 클래스에서 serialize()함수를 재정의하여 문서의 데이터를 파일이나 메모리 버퍼에 저장하거나 불러올 수 있음

- Serialize() 함수 기본 예제
```
class CMyObject : public CObject //CObject 클래스를 상속받은 CMyClass
{
public:
    // 멤버 변수 선언
    int m_nValue;

    // Serialize 함수 재정의
    virtual void Serialize(CArchive& ar);
};

void CMyObject::Serialize(CArchive& ar)
{
    // CObject 클래스의 Serialize 함수 호출
    CObject::Serialize(ar);
```

```
    // 멤버 변수 Serialize를    if (ar.IsStoring())
    {
        ar << m_nValue;
    }
    else
    {
        ar >> m_nValue;
    }
}
```

- << 연산자, >>연산자 사용하여 Serialize 함수

- IsStoring() 함수 : CArchive 객체가 데이터를 저장하는 중인지 불러오는 중인지를 나타내는 함수.
기능 : 저장 중인 경우 << 연산자를 사용하여 데이터를 저장하고, 불러오는 중인 경우 >> 연산자를 사용하여 데이터를 불러옴

=> 출력 결과는 따로 나오지 않음.
이유 : Serialize() 함수는 객체를 바이트 스트림으로 변환하거나 바이트 스트림으로부터 객체를 생성하는 데 사용되는 함수이기에 해당 함수를 직접 실행하는 것은 의미 없음

(2) MFC에서 매크로를 클래스의 선언부와 정의부에 매크로를 선언하는 이유가 뭘까?
일반적으로 그렇다고 알려져있지만, 구체적으로 MFC가 내부적으로 많은 코드 생성 매크로를 사용하기 때문이다. MFC의 내부적인 코드 생성 기술을 사용하기 위함이고, 이는 코드의 가독성을 향상시키고, 클래스의 구현과 관련된 복잡한 코드를 단순화할 수 있다. 즉, 클래스의 메타 데이터를 정의하기 위해 사용되고, 주로 클래스의 구현을 정의하기 위해서 사용된다.

1. 클래스의 메타 데이터를 정의
예를 들어, DECLARE_DYNAMIC 매크로는 클래스가 동적으로 생성될 수 있음을 나타내는 메타 데이터를 생성하고, 이러한 메타 데이터는 컴파일러가 클래스를 처리할 때 필요한 정보를 제공한다.

2. 클래스의 구현을 정의
예를 들어, IMPLEMENT_DYNAMIC 매크로는 클래스의 동적 생성 함수를 정의한다. 이러한 매크로는 클래스의 구현과 관련된 내용을 제공하며, 템플릿과 같은 코드 생성 기술을 사용하여 컴파일러가 필요한 코드를 생성하도록 한다.