파일 입출력 II

(13주차)

학습개요

- 학습 목표
  - CArchive 클래스를 이용한 직렬화 기법을 익힌다.

- 학습 내용
  - 직렬화
  - 실습

직렬화 기초

- 직렬화
  - 영속적인 저장 매체에 객체의 내용을 저장하거나 읽어오는 기법

- 데이터 쓰기 - 일반 파일 입출력

```
void CFileIOTestView::OnLButtonDown(UINT nFlags, CPoint point)
{
    CFile file;
    CFileException e;
    if(!file.Open(_T("mytest.dat"), CFile::modeWrite|CFile::modeCreate, &e))
    {
      e.ReportError();
      return;
    }
    double a = 1.23;
    double b = 4.56;
    file.Write(&a, sizeof(a));
    file.Write(&b, sizeof(b));
}
```

직렬화 기초

- 데이터 읽기 - 일반 파일 입출력

```
void CFileIOTestView::OnRButtonDown(UINT nFlags, CPoint point)
{
    CFile file;
    CFileException e;
    if(!file.Open(_T("mytest.dat"), CFile::modeRead, &e)){
      e.ReportError();
      return;
    }



    double a, b;
    file.Read(&a, sizeof(a));
    file.Read(&b, sizeof(b));
    TRACE(_T("a = %f, b = %f₩n"), a, b);
}
```

직렬화 기초

- 데이터 쓰기 - 직렬화

```
void CFileIOTestView::OnLButtonDown(UINT nFlags, CPoint point)
{
    CFile file;
    CFileException e;
    if(!file.Open(_T("mytest.dat"), CFile::modeWrite | CFile::modeCreate, &e))
    {
      e.ReportError();
      return;
    }

    double a = 1.23;
    double b = 4.56;
    CArchive ar(&file, CArchive::store);
    ar << a << b;
}
```

직렬화 기초

- 데이터 읽기 - 직렬화

```
void CFileIOTestView::OnRButtonDown(UINT nFlags, CPoint point)
{
    CFile file;
    CFileException e;
    if(!file.Open(_T("mytest.dat"), CFile::modeRead, &e)){
      e.ReportError();
      return;
    }


    double a, b;
    CArchive ar(&file, CArchive::load);
    ar >> a >> b;
    TRACE(_T("a = %f, b = %f₩n"), a, b);
}
```

직렬화 기초

- CArchive 클래스 생성자

```
CArchive::CArchive(CFile* pFile, UINT nMode,  int nBufSize=4096, void*lpBuf=NULL);
                          ①         ②         ③                  ④
```

  - pFile
    - CFile 객체
  - nMode
    - CArchive::load 또는 CArchive::store
  - nBufSize
    - CArchive 클래스 내부에서 사용할 버퍼 크기
  - lpBuf
    - 사용자 정의 버퍼의 주소

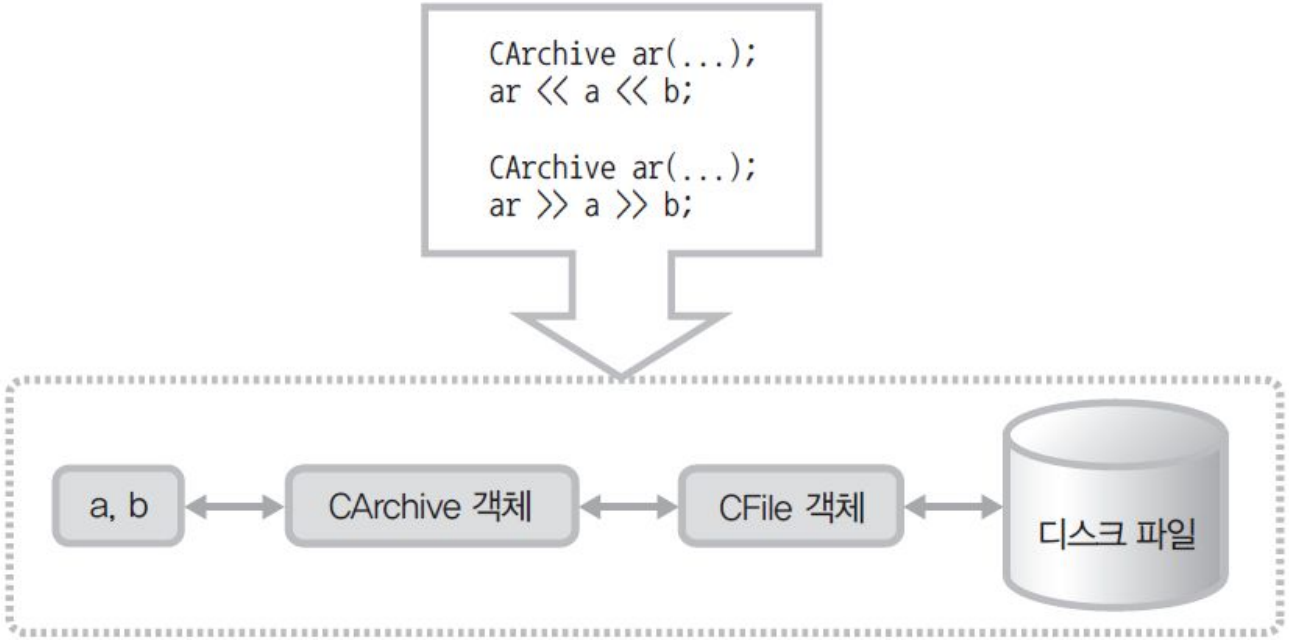
직렬화 기초

- 직렬화 가능한 데이터 타입

구분	데이터 타입
기본형	BYTE, WORD, LONG, DWORD, float, double, int, short, char, wchar_t, unsigned
비기본형	RECT, POINT, SIZE, CRect, CPoint, CSize, CString, CTime, CTimeSpan, COleVariant, COleCurrency, COleDateTime, COleDataTimeSpan

직렬화 기초

- 직렬화 원리


CArchive ar(...);
ar << a << b;
CArchive ar(...);
ar >> a >> b;
a, b
CArchive 객체
CFile 객체
디스크 파일

도큐먼트/뷰 구조와 직렬화

- [파일]->[열기...] 메뉴를 선택한 경우

```
ON_COMMAND(ID_FILE_OPEN, &CWinApp::OnFileOpen)
```

```
BOOL CDocument::OnOpenDocument(LPCTSTR lpszPathName)
{
    …(CFile 객체를 생성한다; pFile은 CFile 객체의 주소값을 담고 있다.)
    …
    CArchive ar(pFile, CArchive::load | CArchive::bNoFlushOnDelete);
    …
    Serialize(ar)
    …
}
```

도큐먼트/뷰 구조와 직렬화

- [파일]->[저장] 또는 [다른 이름으로 저장...] 메뉴를 선택한 경우

```
ON_COMMAND(ID_FILE_SAVE, OnFileSave)
ON_COMMAND(ID_FILE_SAVE_AS, OnFileSaveAs)
```

```
BOOL CDocument::OnSaveDocument(LPCTSTR lpszPathName)
{
    …(CFile 객체를 생성한다; pFile은 CFile 객체의 주소값을 담고 있다.)
    …
    CArchive ar(pFile, CArchive::store | CArchive::bNoFlushOnDelete);
    …
    Serialize(ar)
    …
}
```

직렬화 클래스 구현

- 사용자 정의 클래스

```
class CMyData
{
public:
    CString m_str;
    COLORREF m_color;
public:
    CMyData(CString &str, COLORREF &color) { m_str = str; m_color = color; }
    virtual ~CMyData();
};
```

직렬화 클래스 구현

- 직렬화 ⇨ X
  - CFileIOTestDoc 클래스에서 CMyData 타입 변수 m_data에 데이터를 유지한다고 가정했을 때, 다음 코드는 동작하지않는다.

```
void CFileIOTestDoc::Serialize(CArchive& ar)
{
    if (ar.IsStoring())
    {
        … (다른 데이터의 직렬화 코드)
        ar << m_data;
    }
    else
    {
        … (다른 데이터의 직렬화 코드)
        ar >> m_data;
    }
}
```

직렬화 클래스 구현

- 사용자 정의 클래스 변경 (1/2)

```
/* 클래스 선언부 */
❶ class CMyData : public CObject
{
   ❷ DECLARE_SERIAL(CMyData)
public:
    CString m_str;
    COLORREF m_color;
public:
   ❸ CMyData() { }
    CMyData(CString &str, COLORREF &color) { m_str = str; m_color = color; }
    virtual ~CMyData();
   ❹ void Serialize(CArchive& ar);
};
```

직렬화 클래스 구현

- 사용자 정의 클래스 변경 (2/2)

```
/* 클래스 구현부 */
CMyData::~CMyData()
{
}

❺IMPLEMENT_SERIAL(CMyData, CObject, 1)

❻void CMyData::Serialize (CArchive& ar)
{
    CObject::Serialize(ar);
    if(ar.IsStoring())
      ar << m_str << m_color;
    else
      ar >> m_str >> m_color;
}
```

직렬화 클래스 구현

- 직렬화 ⇨ O

```
void CFileIOTestDoc::Serialize(CArchive& ar)
{
    if(ar.IsStoring())
    {
      ... (다른 데이터의 직렬화 코드)
      m_data.Serialize(ar);
    }
    else
    {
      ... (다른 데이터의 직렬화 코드)
      m_data.Serialize(ar);
    }
}
```

실습

학습정리

- 직렬화란 영속적인 저장 매체에 객체의 내용을 저장하거나 읽어오는 기법을 말합니다.
- MFC 객체 직렬화에서 객체를 파일에 저장하기 위해 CArchive 객체 생성 시 CArchive::store 모드를 사용하고, << 연산자를 사용합니다.
- MFC 객체 직렬화에서 파일에 저장된 객체의 정보를 읽어오기 위해 CArchive 객체 생성 시 CArchive::load 모드를 사용하고, >> 연산자를 이용합니다.
- 사용자 정의 직렬화 가능 클래스를 정의하기 위해서는 CObject를 상속받아야 하며, DECLARE_SERIAL – IMPLEMENT_SERIAL 매크로를 작성하고, 기본 생성자를 작성하며, 객체를 파일에 저장하고 읽어올 수 있는 기능을 가진 Serialize 함수를 재정의해야 합니다.