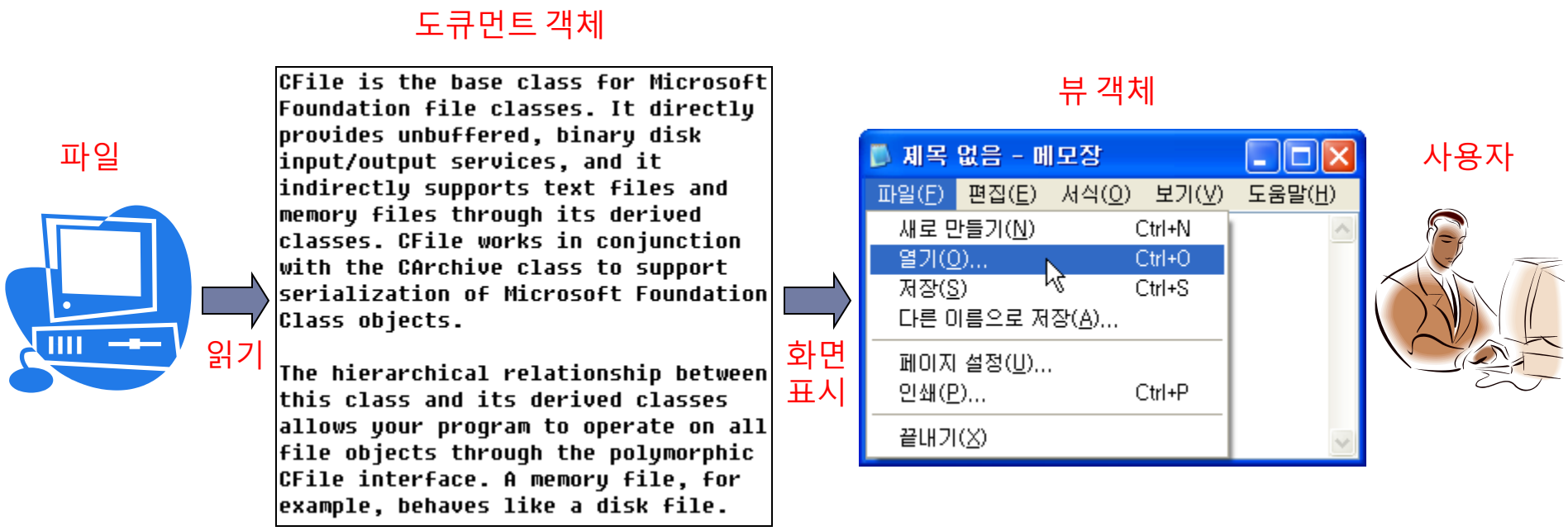
13장
직렬화

김성영교수
금오공과대학교
컴퓨터공학부

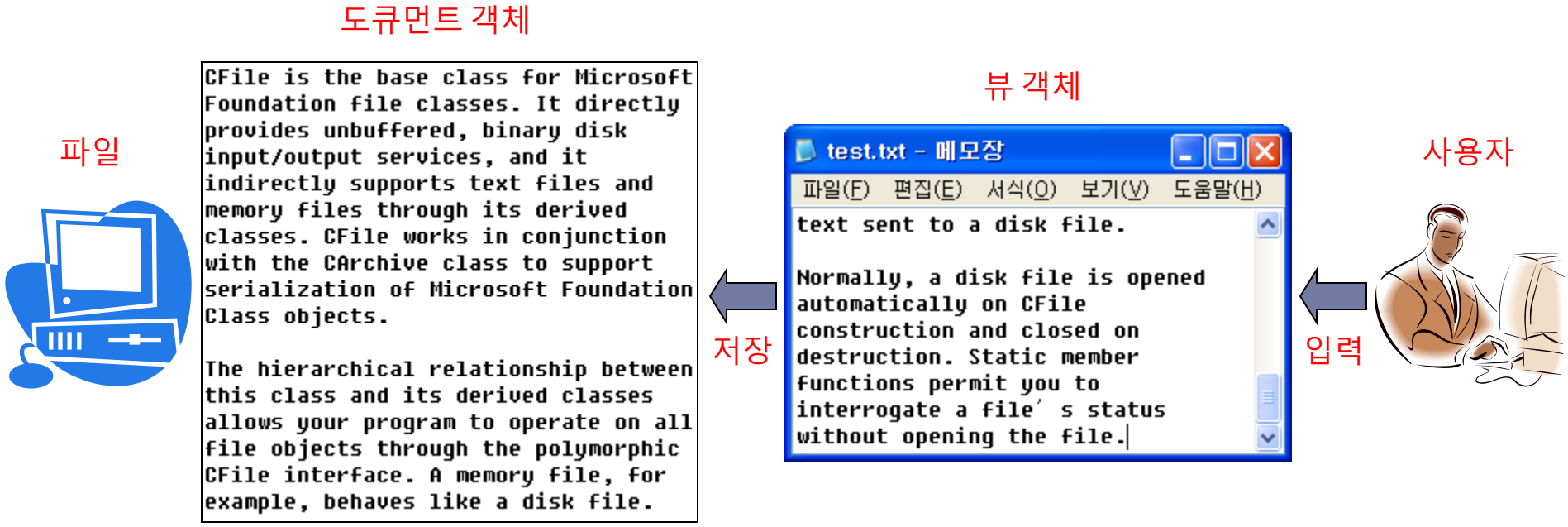
도큐먼트/뷰 구조 (1)

- 도큐먼트와 뷰
  - 디스크에 저장된 파일 데이터를 읽는 경우



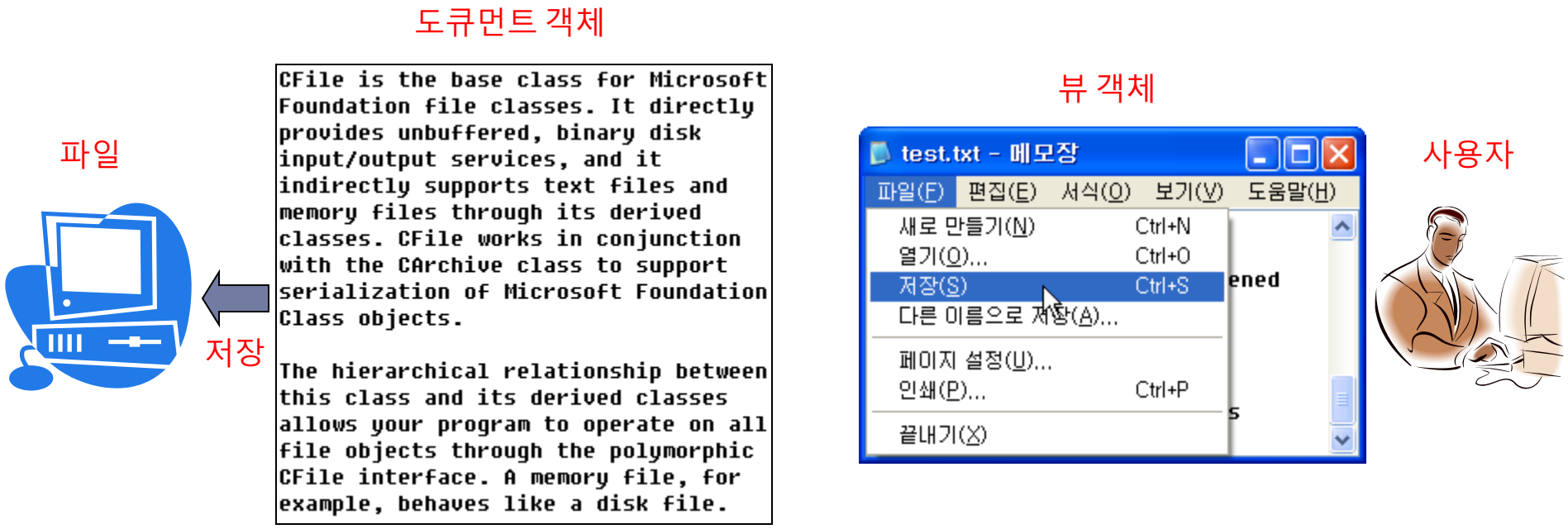
도큐먼트/뷰 구조 (2)

- 도큐먼트와 뷰
  - 사용자가 데이터를 입력하는 경우



도큐먼트/뷰 구조 (3)

- 도큐먼트와 뷰
  - 입력된 데이터를 디스크 파일에 저장하는 경우



도큐먼트/뷰 구조 (4)

- 도큐먼트와 뷰 클래스의 역할

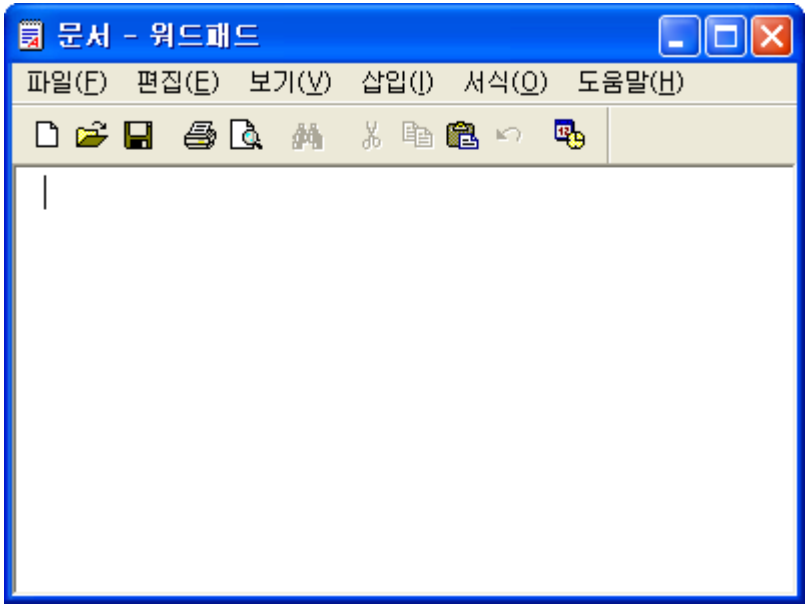
클래스	역할
도큐먼트	데이터를 저장하거나 읽어들인다. 데이터의 변경 사항이 생기면 뷰의 화면을 갱신한다.
뷰	데이터를 화면에 표시한다. 사용자와의 상호 작용을 담당한다.

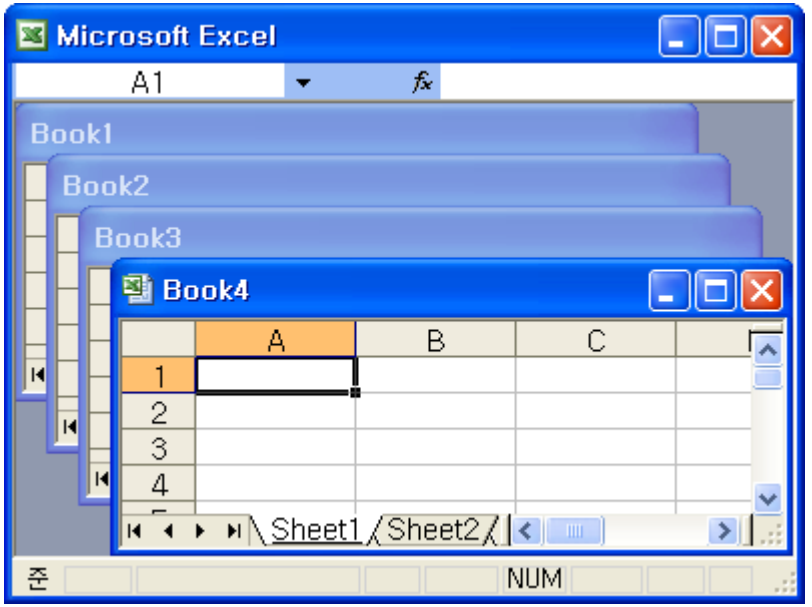
도큐먼트/뷰 구조 (5)

- 도큐먼트/뷰 구조의 장점
  - 서로 다른 기능을 도큐먼트와 뷰 클래스로 분리해서 구현하므로 개념적으로 이해하기 쉬움
  - 도큐먼트 하나에 뷰가 여러 개 존재하는 모델을 구현하기 쉬움
    - 예) 비주얼 C++ 편집창
  - MFC에서 도큐먼트/뷰 구조를 위해 제공하는 부가 서비스 이용 가능
    - 예) 직렬화

도큐먼트/뷰 구조 (6)

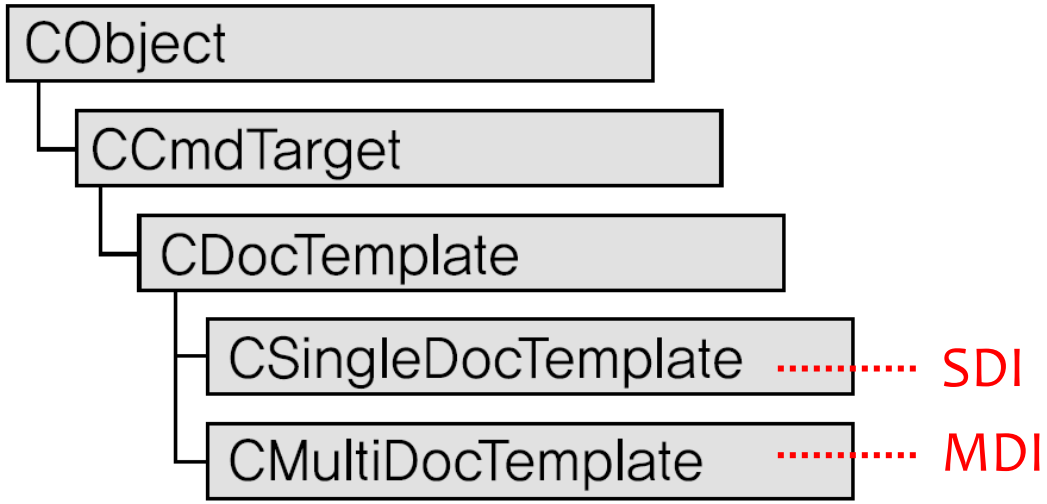
- SDI와 MDI
  - 다룰 수 있는 문서의 개수에 따라 구분





도큐먼트/뷰 구조 (7)

- 도큐먼트 템플릿
  - 도큐먼트, 프레임 윈도우, 뷰 클래스 정보를 유지
  - 필요에 따라 해당 객체를 동적으로 생성
- MFC 클래스 계층도



도큐먼트 객체에서의 파일처리

어느 메시지 핸들러에서 파일을 열고 닫을 것인가!

실습 13.1

도큐먼트 클래스는 기본적으로 다음 두 함수 재정의

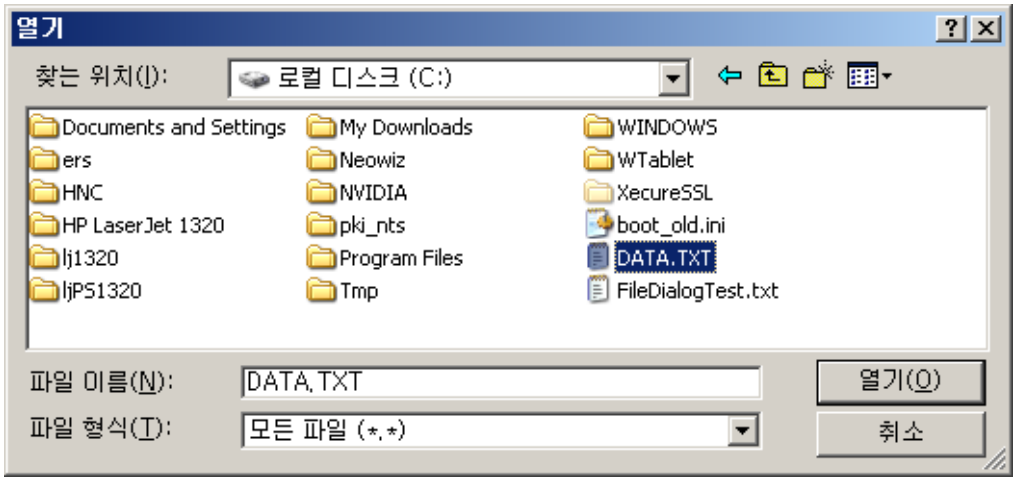
```
BOOL OnNewDocument();       //초기화
void Serialize(CArchive& ar);   //파일입출력
```

```
BOOL CMyDoc::OnNewDocument( )
{
        if (!CDocument::OnNewDocument())
                return FALSE;
        AfxMessageBox( "OnNewDocument" );
        return TRUE;
}

void CMyDoc::Serialize( CArchive& ar )
{
        if (ar.IsStoring())
        {
                AfxMessageBox( "Serialize: Storeing" );
        }
        else
        {
                AfxMessageBox( "Serialize: Loading" );
        }
}
```

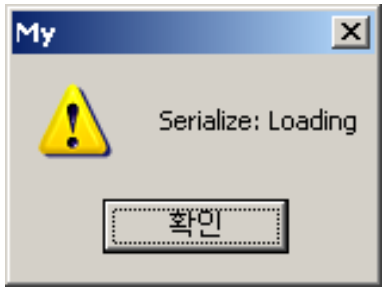
```
BOOL CMyDoc::OnOpenDocument( LPCTSTR lpszPathName )
{
        if (!CDocument::OnOpenDocument(lpszPathName))
                return FALSE;
        AfxMessageBox("OnOpemDocument");
        return TRUE;
}
BOOL CMyDoc::OnSaveDocument( LPCTSTR lpszPathName )
{
        AfxMessageBox("OnSaveDocument");
        return CDocument::OnSaveDocument(lpszPathName);
}
void CMyDoc::OnCloseDocument()
{
        AfxMessageBox("OnCloseDocument");
        CDocument::OnCloseDocument();
}
void CMyDoc::DeleteContents()
{
        AfxMessageBox("DeleteContents");
        CDocument::DeleteContents();
}
```

- 프로그램 실행 전 메모장에서 Hello라고 입력하고 C:/DATA.TXT로 저장
- 프로그램 실행 시 프레임 윈도우가 나오기 전에 DeleteContents와 OnNewDocument 메시지 박스 출력 (파일 메뉴에서 새파일을 클릭해도 마찬가지)

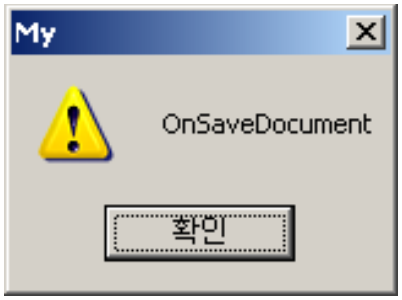
- 열기를 클릭하면 파일 대화상자가 나온다



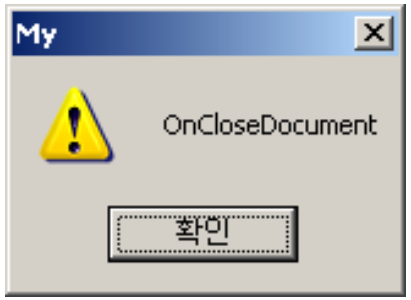
- DATA.TXT 파일을 선택하면 DeleteContents, Serialize:Loading, OnOpenDocument 메시지 출력



- 저장을 클릭하면 OnSaveDocument, Serialize:Storing 메시지 박스 차례대로 출력



- 다른이름으로저장 을 클릭하면 OnSaveDocument, Serialize:Storing 메시지 박스 차례대로 출력
- 프레임 윈도우를 닫으면 OnCloseDocument 메시지 박스 출력



파일 입출력 개요 (1)

- 파일 입출력 방법
  - 일반 파일 입출력
    - CFile (파생) 클래스
      - 바이너리 모드로 파일 입출력을 지원하는 MFC 클래스
    - Read(), Write() 등의 함수 이용
  - 직렬화
    - CArchive 클래스
    - << 또는 >> 연산자 이용

파일 입출력 개요 (2)

- 데이터 읽기 - 일반 파일 입출력

```
CFile file;
CFileException e;
if( !file.Open("mytest.dat", CFile::modeRead, &e) )
{
    e.ReportError();
    return;
}

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

파일 입출력 개요 (3)

- 데이터 쓰기 - 일반 파일 입출력

```
CFile file;
CFileException e;
if( !file.Open("mytest.dat", CFile::modeReadWrite
    |CFile::modeCreate, &e) )
{
    e.ReportError();
    return
}

double a = 1.23;
double b = 4.56;
file.Write( &a, sizeof(a) );
file.Write( &b, sizeof(b) );
```

파일 입출력 개요 (4)

- 데이터 읽기 - 직렬화

```
CFile file;
CFileException e;
if(!file.Open( "mytest.dat", CFile::modeRead, &e) )
{
    e.ReportError();
    return;
}

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

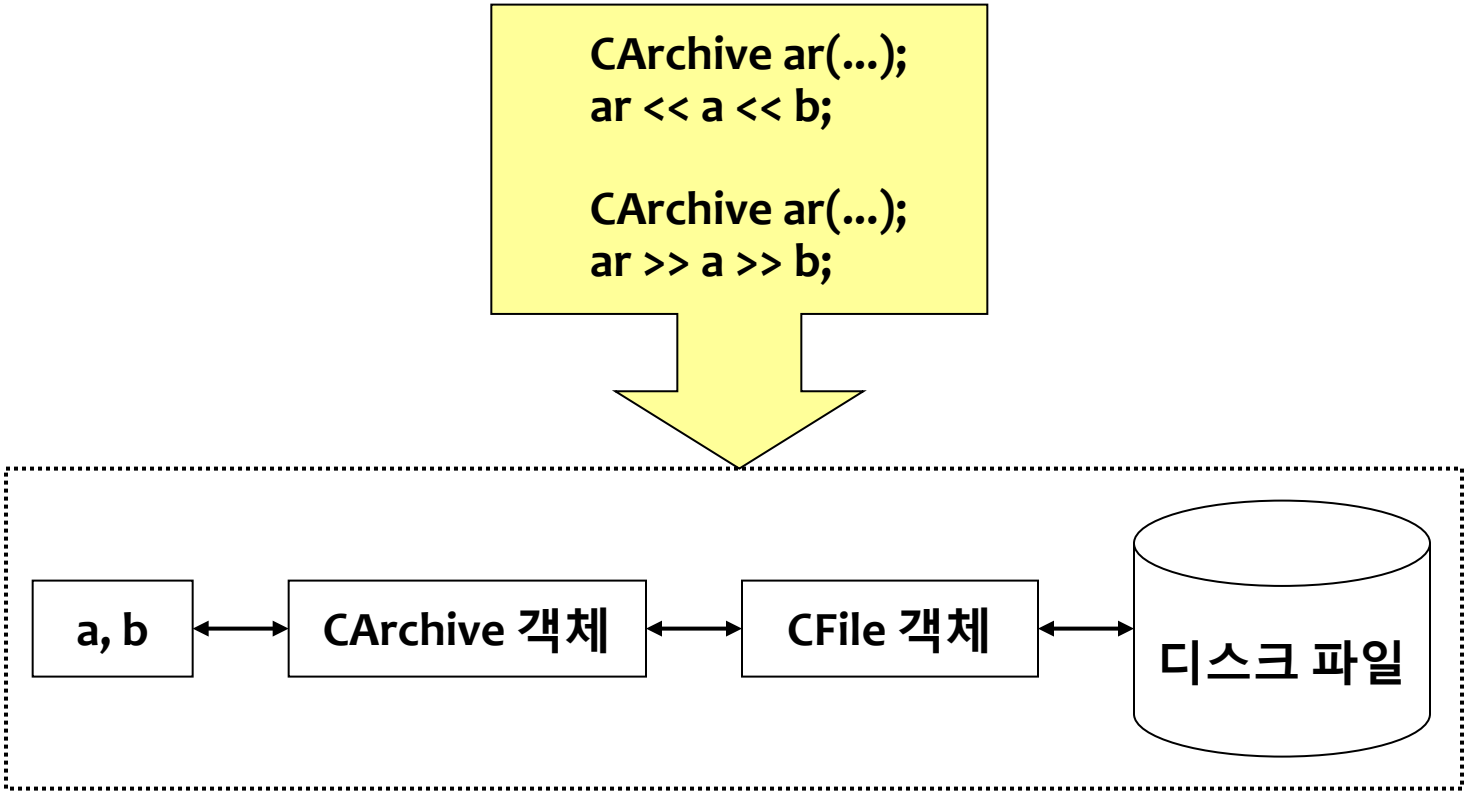
파일 입출력 개요 (5)

- 데이터 쓰기 - 직렬화

```
CFile file;
CFileException e;
if(!file.Open( "mytest.dat", CFile::modeReadWrite
    |CFile::modeCreate, &e) )
{
    e.ReportError();
    return;
}

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

직렬화 기초 (1)

- 직렬화 원리



직렬화 기초 (2)

- CArchive 클래스 생성자

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

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

직렬화 기초 (3)

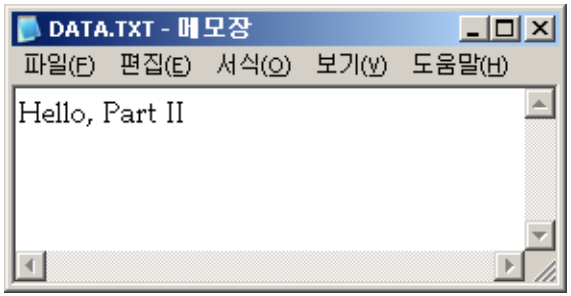
- 직렬화 가능한 데이터 타입

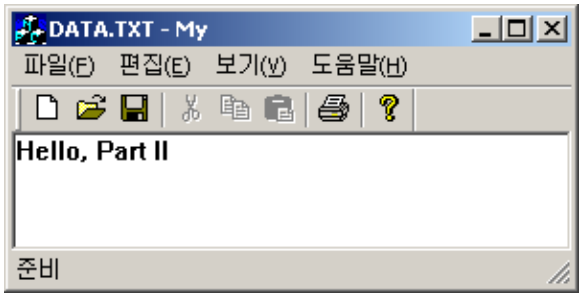
구분	데이터 타입
기본형	BYTE, WORD, LONG, DWORD, float, double, int, short, char, wchar_t, unsigned
비기본형	RECT, POINT, SIZE, CRect, CPoint, CSize, CString, CTime, CTimeSpan, COleVariant, COleCurrency, COleDateTime, COleDataTimeSpan

CFile 클래스를 이용한 입출력

MYDOC.CPP

```
BOOL CMyDoc::OnOpenDocument(LPCTSTR lpszPathName)
{
        if (!CDocument::OnOpenDocument(lpszPathName))
                return FALSE;

        CFile f;
        f.Open( lpszPathName, CFile::modeRead );
        f.Read( m_szMsg, 256 );
        f.Close( );

        UpdateAllViews( NULL );
        return TRUE;
}
```





```
BOOL CMyDoc::OnSaveDocument(LPCTSTR lpszPathName)
{
        char szMsg[] = "Hello, Part II";
        CFile f;

        f.Open( lpszPathName, CFile::modeWrite|CFile::modeCreate );
        f.Write( szMsg, strlen(szMsg) );
        f.Close( );

        return TRUE;
}
```

CArchive 클래스를 이용한 입출력

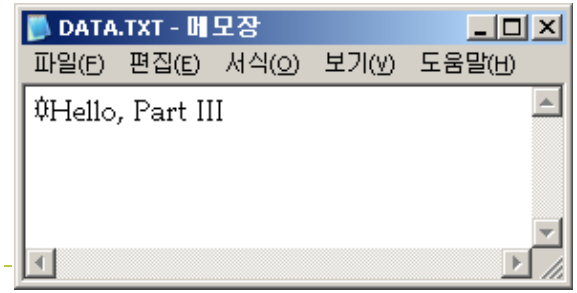
- 객체만 처리 가능 → 문자배열 대신 문자열 클래스 CString사용
- 실제로 파일을 처리하는 주체는 CFile 객체 → 먼저 CFile 객체 준비

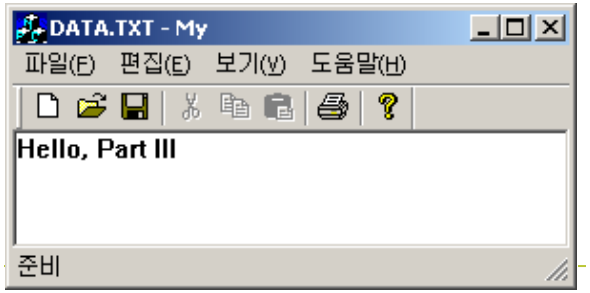
MYDOC.CPP

```
BOOL CMyDoc::OnOpenDocument(LPCTSTR lpszPathName) {
        if (!CDocument::OnOpenDocument(lpszPathName))
                return FALSE;

        CString strMsg;
        CFile f;

        f.Open( lpszPathName, CFile::modeRead );
        CArchive ar( &f, CArchive::load );
        ar >> strMsg; //문자열 객체로 읽음
        strcpy( m_szMsg, (LPCSTR)strMsg ); //문자배열에 복사

        ar.Close( );
        f.Close( );
```

```
        UpdateAllViews(NULL);
        return TRUE;
}
BOOL CMyDoc::OnSaveDocument(LPCTSTR lpszPathName) {
        CString strMsg;
        strMsg = "Hello, Part III";

        CFile f;
        f.Open( lpszPathName, CFile::modeWrite|CFile::modeCreate );
        CArchive ar( &f, CArchive::store );
        ar << strMsg; //문자열 객체로 저장
        ar.Close();
        f.Close();

        return TRUE;
}
```

- **프로그램 실행 결과**





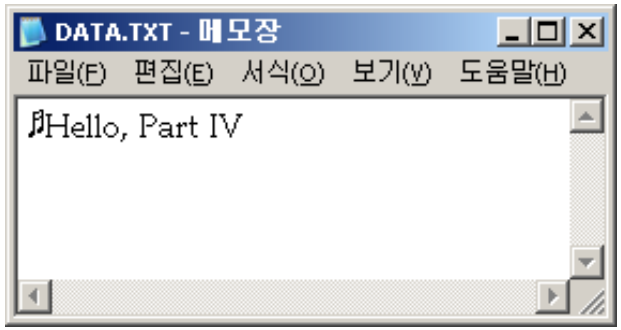
Serialize() 함수를 이용한 입출력

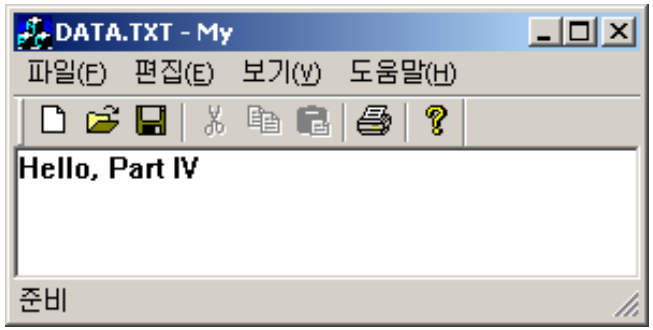
- 인자로 넘어오는 CArchive 객체를 이용해 입출력 작업

MYDOC.CPP

```
void CMyDoc::Serialize( CArchive& ar )
{
    CString strMsg;
    if( ar.IsStoring() )
    {
        strMsg = "Hello, Part IV";
        ar << strMsg;
    }
    else
    {
        ar >> strMsg;
        strcpy( m_szMsg, (LPCSTR)strMsg );
        UpdateAllViews( NULL );
    }
}
```

- 프로그램 실행 결과





- Serialize() 함수를 사용하여 처리될 수 있는 데이터는 Serialize()가 가능한 객체이어야함

- OnopenDocument()나 Serialize()함수를 사용하는 경우 드래그앤드롭(Drag & Drop) 기능 사용가능
- CMyApp 클래스의 InitInstance() 함수에 코드 추가

MY.CPP

```
BOOL CMyApp::InitInstance()
{
        m_pMainWnd->DragAcceptFiles(TRUE);
        return TRUE;
}
```

직렬화 가능 객체

실습 13.3

- 마우스 클릭 – 원을 그림
- 저장버튼 – 현재 좌표를 파일로 저장
- 열기버튼 – 파일에 저장해둔 좌표를 읽어와 원을 복원
- 마우스의 좌표를 직렬화 가능한 클래스로 구현

- 직렬화 가능한 클래스는 CObject 클래스로부터 상속받아 기본적으로 Serialize() 함수를 재정의하여 구현
- 클래스 정의와 구현부분에 직렬화 처리를 위한 매크로 추가

```
DECLARE_SERIAL(class_name ) //헤더파일에 추가
IMPLEMENT_SERIAL(class_name, base_class_name, wSchema) //구현파일에 추가
```

직렬화를 구현하려는 클래스

기반 클래스

버전 정보 (보통 1)

- 마우수의 좌표를 저장할 수 있는 CMyPoint 클래스 작성
- Serialize()함수에서는 기반 클래스의 Serialize()함수를 먼저 호출

MYPOINT.H

```
#ifndef __MY_POINT__
#define __MY_POINT__

class CMyPoint: public CObject
{
public:
        CMyPoint();
        ~CMyPoint();
        long x;
        long y;
        void Serialize(CArchive& ar);
        DECLARE_SERIAL(CMyPoint)
};
#endif
```

MYPOINT.CPP

```
#include "stdafx.h"
#include "mypoint.h"

IMPLEMENT_SERIAL(CMyPoint, CObject, 1)
CMyPoint::CMyPoint()
{
        x = 0;
        y = 0;
}
CMyPoint::~CMyPoint()
{
}
void CMyPoint::Serialize(CArchive& ar)
{
        CObject::Serialize(ar); //기반 클래스의 Serialize() 호출
        if (ar.IsStoring())
        {
                ar << x << y;
        } else {
                ar >> x >> y;
        }
}
```

- 도큐먼트와 뷰 클래스에 마우스 이벤트와 파일 입출력 처리 추가

MYDOC.H

```
#include "mypoint.h"
class CMyDoc : public CDocument
{
public:
        CMyPoint m_MyPt; //직렬화 가능 객체
}
```

MYDOC.CPP

```
BOOL CMyDoc::OnNewDocument()
{
        if (!CDocument::OnNewDocument())
                return FALSE;
        m_MyPt.x = 0;
        m_MyPt.y = 0;
        return TRUE;
}
```

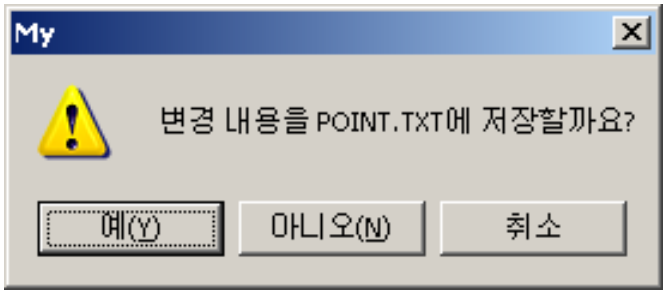
```
void CMyDoc::Serialize(CArchive& ar)
{
        m_MyPt.Serialize(ar); //m_MyPt객체의 파일 입출력 담당
        if (ar.IsStoring())
        { //CMyDoc 객체의 자체 데이터의 저장처리
        }
        else
        { //CMyDoc 객체의 자체 데이터 읽기처리
        }
        UpdateAllViews(NULL); //뷰 갱신
}
```

MYVIEWCPP

```
void CMyView::OnDraw(CDC* pDC)
{
        CMyDoc* pDoc = GetDocument();
        ASSERT_VALID(pDoc);
        CRect rt;
        rt.left = pDoc->m_MyPt.x - 50;
        rt.top = pDoc->m_MyPt.y - 50;
        rt.right = pDoc->m_MyPt.x + 50;
        rt.bottom = pDoc->m_MyPt.y + 50;
        pDC->Ellipse(rt);
}
```

```
void CMyView::OnLButtonDown(UINT nFlags, CPoint point)
{
        CMyDoc* pDoc=GetDocument();
        pDoc->m_MyPt.x = point.x; //도큐먼트의 멤버변수에 마우스 좌표 기록
        pDoc->m_MyPt.y = point.y;
        pDoc->SetModifiedFlag(); //도큐먼트의 내용이 변경되었음을 알림
        Invalidate();
        CView::OnLButtonDown(nFlags, point);
}
```

- **SetModifiedFlag()는 도큐먼트 내용 변경을 알림**
- **도큐먼트 내용이 저장 되지 않고 파괴되는 경우 대화상자 출력**



- **CMyDoc 클래스의 Serialize() 함수 수정 가능**

MYDOC.CPP

```
void CMyDoc::Serialize(CArchive& ar)
{
        //m_MyPt.Serialize(ar); //주석처리
        if (ar.IsStoring()) {
                ar << &m_MyPt;
        }
        else {
                CMyPoint* tmp;
                ar >> tmp;
                m_MyPt.x = tmp->x; //tmp가 참조하고 있는 것 복사
                m_MyPt.y = tmp->y; //tmp가 참조하고 있는 것 복사
        }
        UpdateAllViews(NULL);
}
```