쉽게 배우는 MFC 윈도우 프로그래밍

Visual C++ 2019

2015·2017 버전 사용 가능

7장. 파일 입출력

목차

파일 입출력 개요

파일 입출력 방법

- 일반 파일 입출력
  - CFile (파생) 클래스 객체를 생성
  - Read(), Write() 같은 멤버 함수를 이용하여 처리

- 직렬화
  - CArchive 클래스 객체를 생성
  - << 또는 >> 연산자 이용하여 처리

파일 입출력 개요

- MFC 클래스 계층도

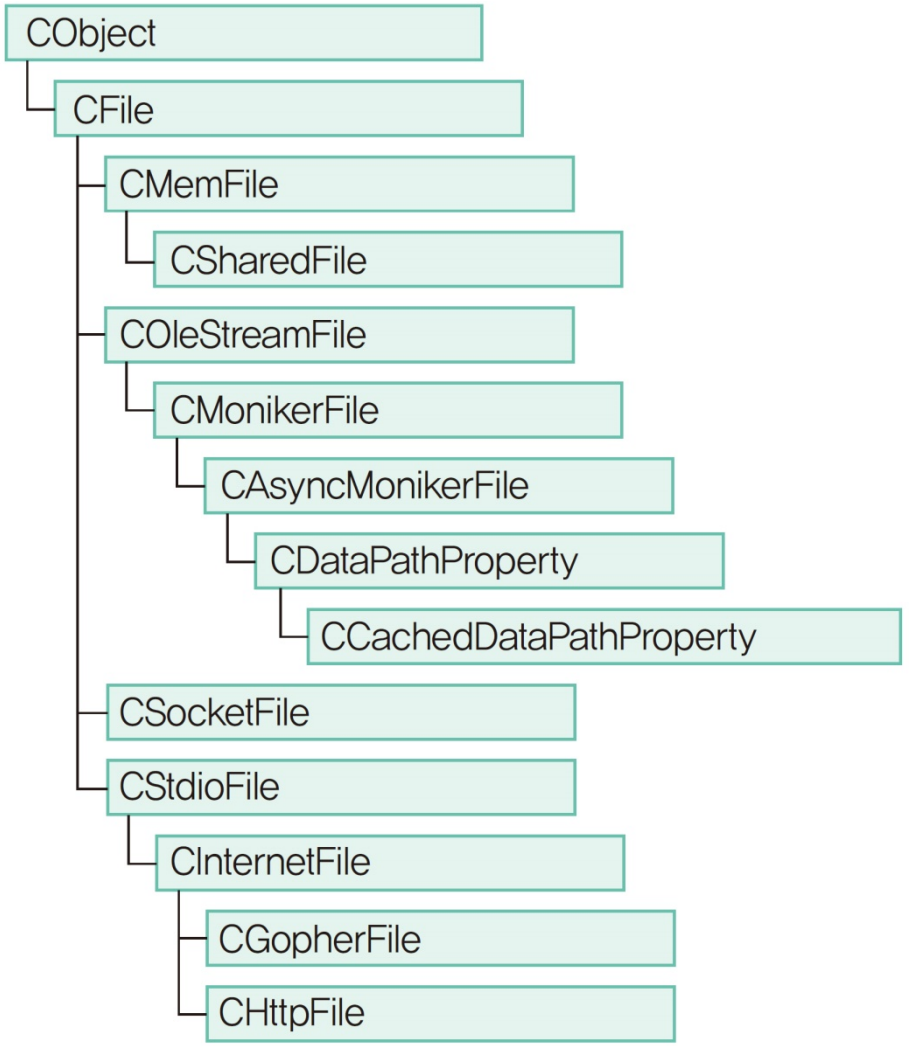


그림 7-1 MFC 클래스 계층도

CFile 클래스

- CFile 클래스가 제공하는 핵심 입출력 연산
  - 파일을 열거나 생성(Open)
  - 데이터를 읽음(Read)
  - 데이터를 씀(Write)
  - 입출력 위치를 변경(Seek)
  - 파일을 닫음(Close)

CFile 클래스

열기와 생성

```
try {
	CFile file(_T("mytest.txt"), CFile::modeReadWrite);
}
catch (CFileException* e) {
	e->ReportError();
	e->Delete();
}
```

```
CFile file;
CFileException e;
if (!file.Open(_T("mytest.txt"), CFile::modeReadWrite, &e))
	e.ReportError();
```

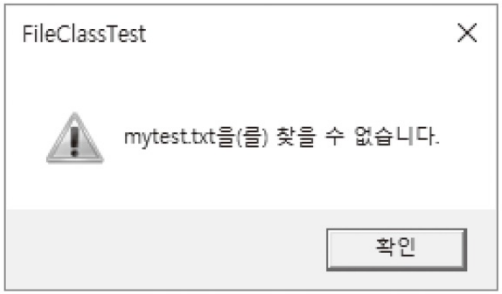


그림 7-2 오류 메시지

CFile 클래스

표 7-1 파일 접근과 공유 모드

플래그	의미
CFile::modeCreate	파일을 무조건 생성한다. 이름이 같은 파일이 있다면 크기를 0으로 바꾼다.
CFile::modeNoTruncate	| 연산자로 CFile::modeCreate 플래그와 조합해서 사용하면, 이름이 같은 파일이 있을 때 크기를 0으로 바꾸지 않고 그 파일을 연다.
CFile::modeRead	파일을 읽기 전용 모드로 열거나 생성한다.
CFile::modeReadWrite	파일을 읽기 및 쓰기 모드로 열거나 생성한다.
CFile::modeWrite	파일을 쓰기 전용 모드로 열거나 생성한다.
CFile::shareDenyNone	파일에 대한 읽기와 쓰기를 다른 프로세스에 허용한다.
CFile::shareDenyRead	다른 프로세스가 파일을 읽는 것을 금한다.
CFile::shareDenyWrite	다른 프로세스가 파일에 쓰는 것을 금한다.
CFile::shareExclusive	다른 프로세스가 파일을 읽거나 파일에 쓰는 것을 금한다.

CFile 클래스

- 닫기 : 방법1

```
void CChildView::OnLButtonDown(UINT nFlags, CPoint point)
{
	CFile file;
	CFileException e;
	if (!file.Open(_T("mytest.txt"), CFile::modeReadWrite, &e)) {
		e.ReportError();
		return;
	}
	...
} // CFile::~CFile() 소멸자가 호출된다.
```

CFile 클래스

- 닫기 : 방법2 (1/2)

```
void CChildView::OnLButtonDown(UINT nFlags, CPoint point)
{
    CFile file;
    CFileException e;
    if (!file.Open(_T("mytest1.txt"), CFile::modeReadWrite
        |CFile::modeCreate|CFile::modeNoTruncate, &e)) {
        e.ReportError();
        return;
    }
    ...
    file.Close();
```

CFile 클래스

- 닫기 : 방법2 (1/2)

```
if (!file.Open(_T("mytest2.txt"), CFile::modeReadWrite
        |CFile::modeCreate|CFile::modeNoTruncate, &e)) {
        e.ReportError();
    return;
    }
    ...
    file.Close();
    ...
}
```

CFile 클래스

읽기와 쓰기

```
UINT CFile::Read(void* lpBuf, UINT nCount);
void CFile::Write(const void* lpBuf, UINT nCount);
```

입출력 위치 변경하기

```
ULONGLONG CFile::Seek(LONGLONG lOff, UINT nFrom); // MFC 7.0 ~
LONG CFile::Seek(LONG lOff, UINT nFrom); // ~ MFC 6.0
```

표 7-2 nFrom 값

nFrom	의미
CFile::begin	파일의 처음 위치에서 lOff만큼 파일 포인터를 이동한다.
CFile::current	현재 파일 포인터 위치에서 lOff만큼 파일 포인터를 이동한다.
CFile::end	파일의 끝 위치에서 lOff만큼 파일 포인터를 이동한다.

CFile 클래스

기타 함수

- CFile::GetLength(), CFile::SetLength()
  - 파일의 현재 크기를 얻거나 변경
- CFile::GetPosition()
  - 파일 포인터의 현재 위치를 얻음
- CFile::LockRange(), CFile::UnlockRange()
  - 파일의 일정 영역을 잠그거나 해제. 잠근 영역은 다른 프로세스가 접근할 수 없음
- CFile::GetFilePath(), CFile::GetFileName()
  - 파일의 전체 경로(Full Path)와 이름을 얻음

CMemFile 클래스

- 사용 예

```
void CChildView::OnLButtonDown(UINT nFlags, CPoint point)
{
    CMemFile file;

    // 메모리 파일에 데이터 쓰기
    int a = 100;
    file.Write(&a, sizeof(a));

    // 메모리 파일에서 데이터 읽기
    file.SeekToBegin();
    int b;
    file.Read(&b, sizeof(b));

    // 읽은 데이터 출력하기
    TRACE(_T("b = %d\n"), b);
}
```

CStdioFile 클래스

- 사용 예

```
void CChildView::OnLButtonDown(UINT nFlags, CPoint point)
{
	CStdioFile file1;
	CFileException e;
	if (!file1.Open(_T("test1.txt"), CFile::modeRead, &e)) {
		e.ReportError();
		return;
	}
	CStdioFile file2;
	if (!file2.Open(_T("test2.txt"), CFile::modeWrite | CFile::modeCreate, &e)) {
		e.ReportError();
		return;
	}
	CString str;
	while (file1.ReadString(str)) {
		str.MakeUpper(); // 대문자로 바꾼다.
		file2.WriteString(str + _T('\n'));
	}
}
```

CFileFind 클래스

- MFC 클래스 계층도

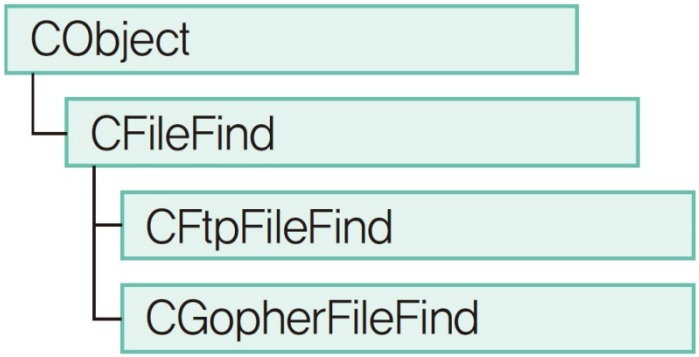


그림 7-3 MFC 클래스 계층도

CFileFind 클래스

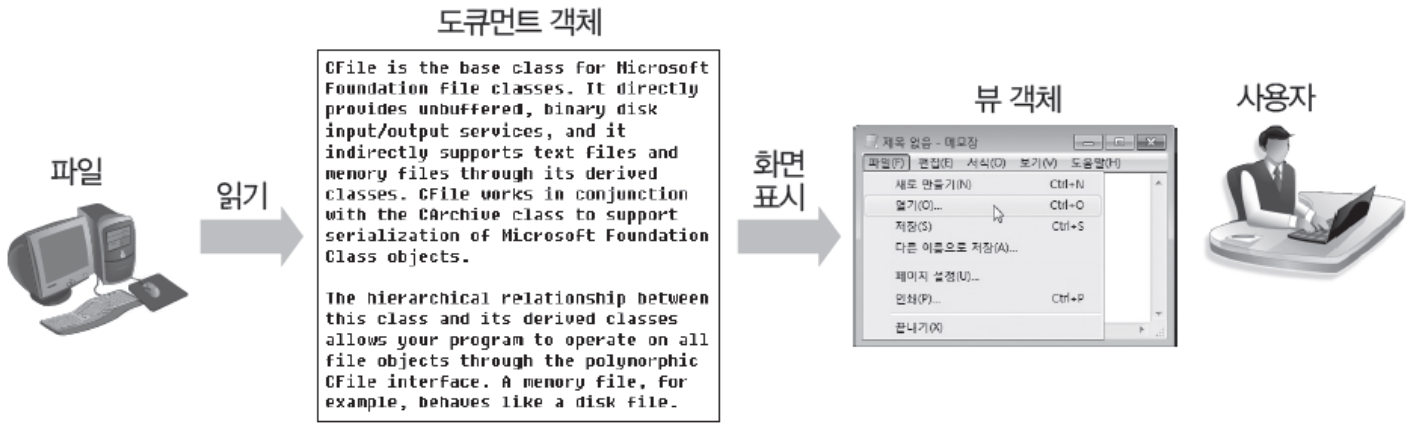
- 사용 예

```
void CChildView::OnLButtonDown(UINT nFlags, CPoint point)
{
	CFileFind finder;
	BOOL bWorking = finder.FindFile(_T("*.*"));
	while (bWorking) {
		bWorking = finder.FindNextFile();
		if(finder.IsDirectory())
			TRACE(_T("[%s]\n"), (LPCTSTR)finder.GetFileName());
		else
			TRACE(_T("%s\n"), (LPCTSTR)finder.GetFileName());
	}
}
```

도큐먼트/뷰 구조

도큐먼트와 뷰

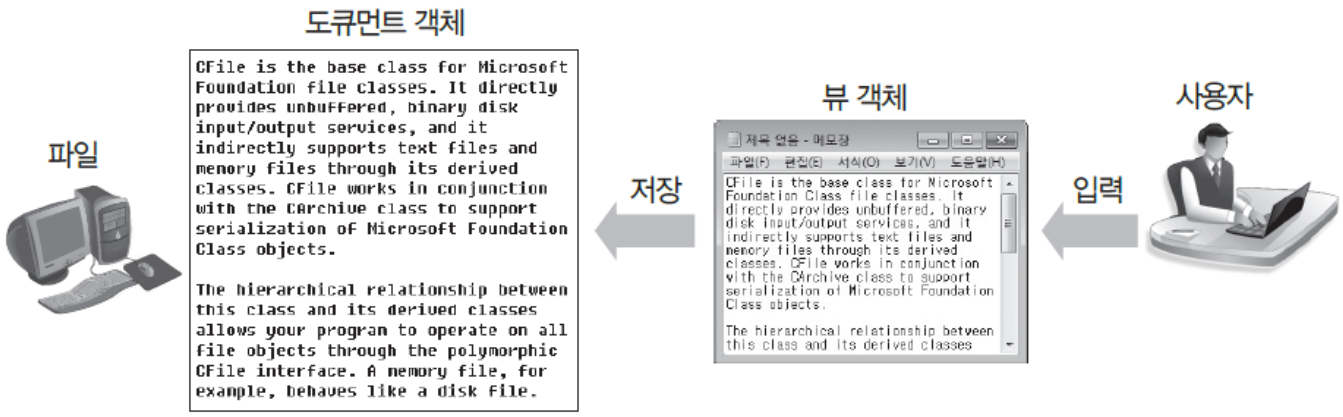
- 디스크에 저장된 파일 데이터를 읽는 경우



도큐먼트/뷰 구조

도큐먼트와 뷰

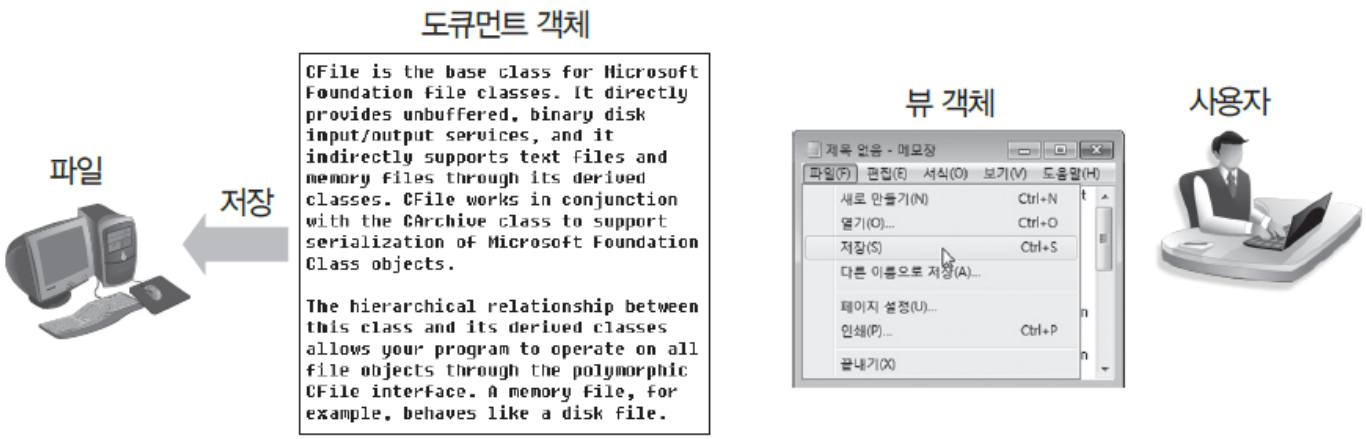
- 사용자가 데이터를 입력하는 경우



도큐먼트/뷰 구조

도큐먼트와 뷰

- 입력된 데이터를 디스크 파일에 저장하는 경우



도큐먼트/뷰 구조

■ 도큐먼트와 뷰 클래스의 역할

표 7-3 도큐먼트와 뷰 클래스의 역할

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

도큐먼트/뷰 구조

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++ 2010 편집창
- MFC에서 도큐먼트/뷰 구조를 위해 제공하는 부가 서비스를 이용할 수 있음
  - 예) 직렬화

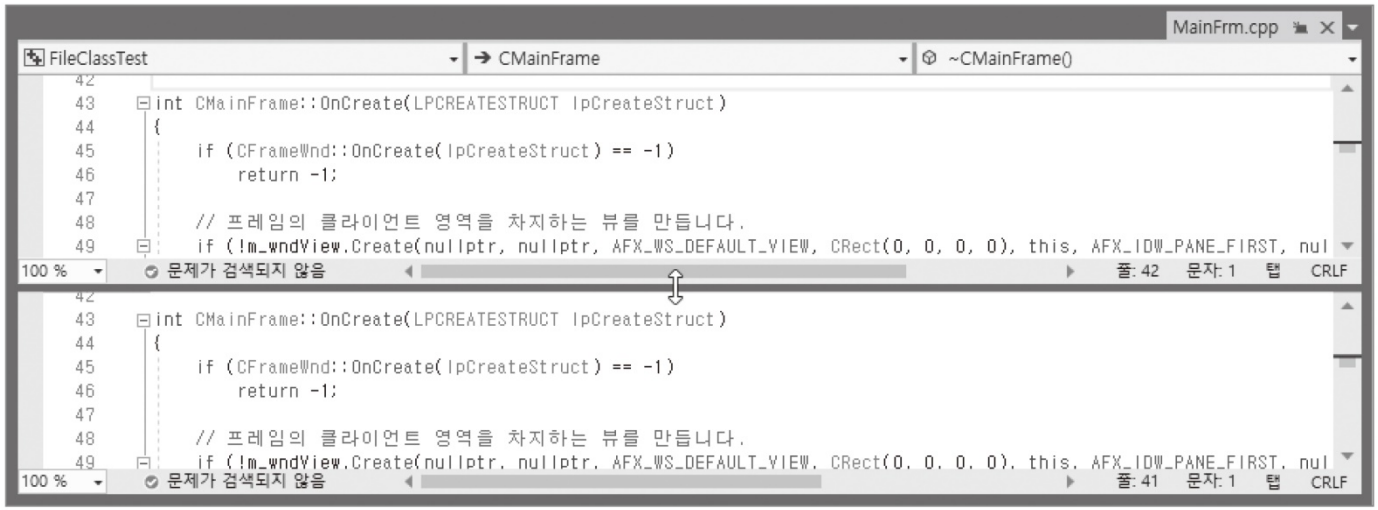
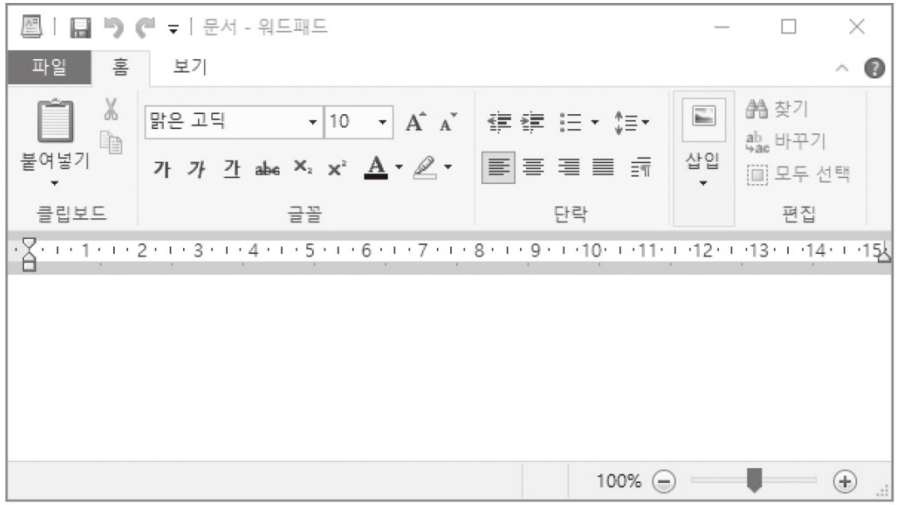


그림 7-5 두 개의 뷰

도큐먼트/뷰 구조

- SDI와 MDI
  - 다룰 수 있는 문서의 개수에 따라 구분



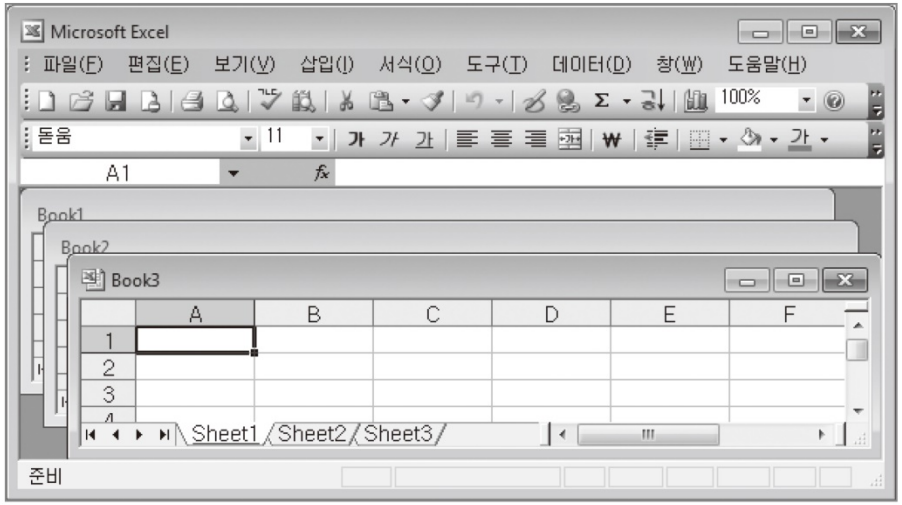


그림 7-6 SDI와 MDI 응용 프로그램의 예

도큐먼트/뷰 구조

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

- MFC 클래스 계층도

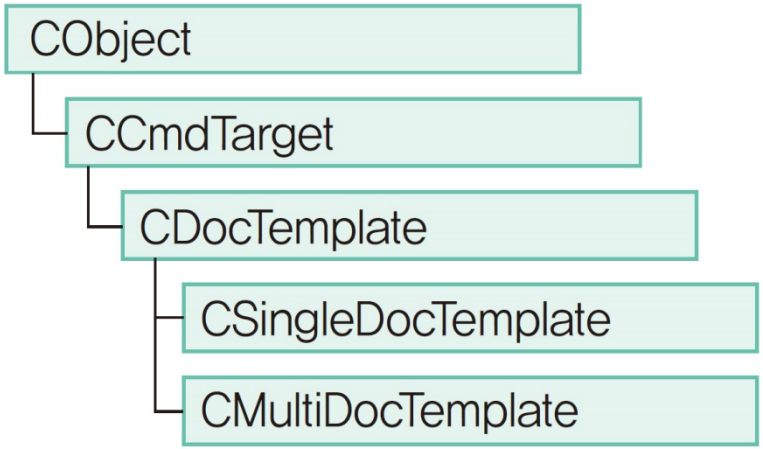


그림 7-7 MFC 클래스 계층도

도큐먼트/뷰 구조

- InitInstance() 함수

```
BOOL CFileIOTestApp::InitInstance()
{
    ...
    CSingleDocTemplate* pDocTemplate;
    pDocTemplate = new CSingleDocTemplate(
        IDR_MAINFRAME, // 리소스 ID
        RUNTIME_CLASS(CFileIOTestDoc), // 도큐먼트 클래스 정보
        RUNTIME_CLASS(CMainFrame), // 프레임 윈도우 클래스 정보
        RUNTIME_CLASS(CFileIOTestView)); // 뷰 클래스 정보
    if (!pDocTemplate)
        return FALSE;
    AddDocTemplate(pDocTemplate); // 응용 프로그램 객체에 도큐먼트 템플릿 등록
    ...
}
```

도큐먼트/뷰 구조

■ 주요 객체의 생성 관계(SDI 응용 프로그램)

표 7-4 주요 객체의 생성 관계(SDI 응용 프로그램)

생성 주체	생성되는 것
❶ 응용 프로그램 객체	❷ 도큐먼트 템플릿 객체
도큐먼트 템플릿 객체	❸ 도큐먼트 객체, ❹ 프레임 윈도우 객체
프레임 윈도우 객체	❺ 뷰 객체

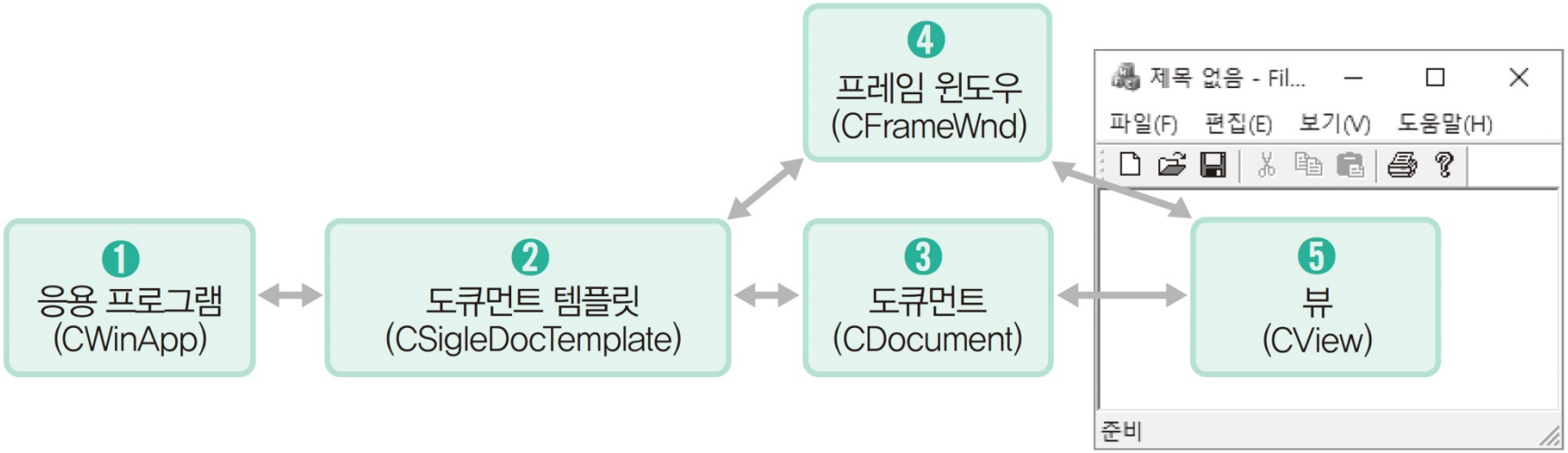


그림 7-8 주요 객체 간의 상호 작용

[실습 7-1] MFC 응용 프로그램 생성하기(도큐먼트/뷰 구조)

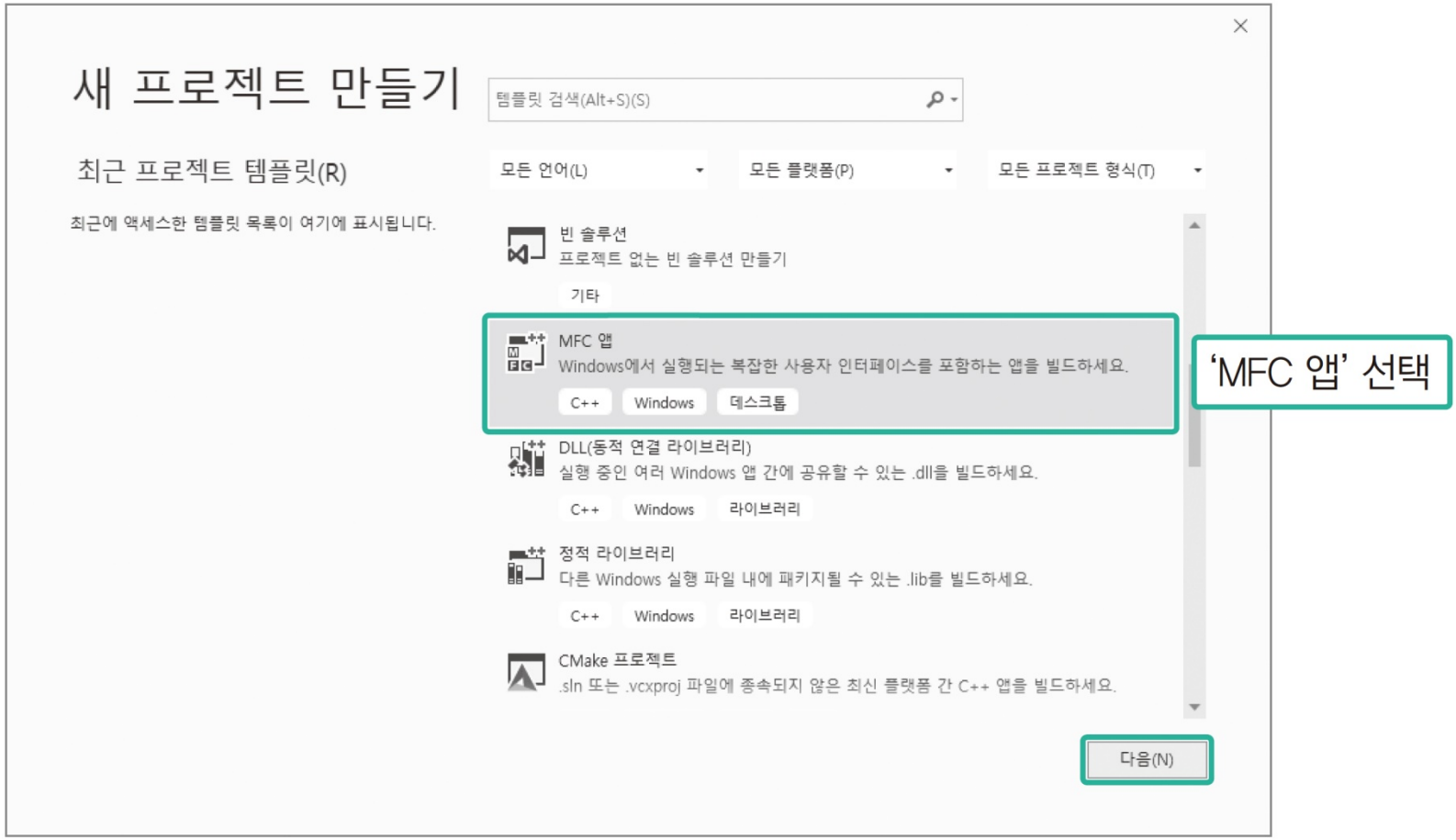


그림 7-9 프로젝트 종류 선택

[실습 7-1] MFC 응용 프로그램 생성하기(도큐먼트/뷰 구조)

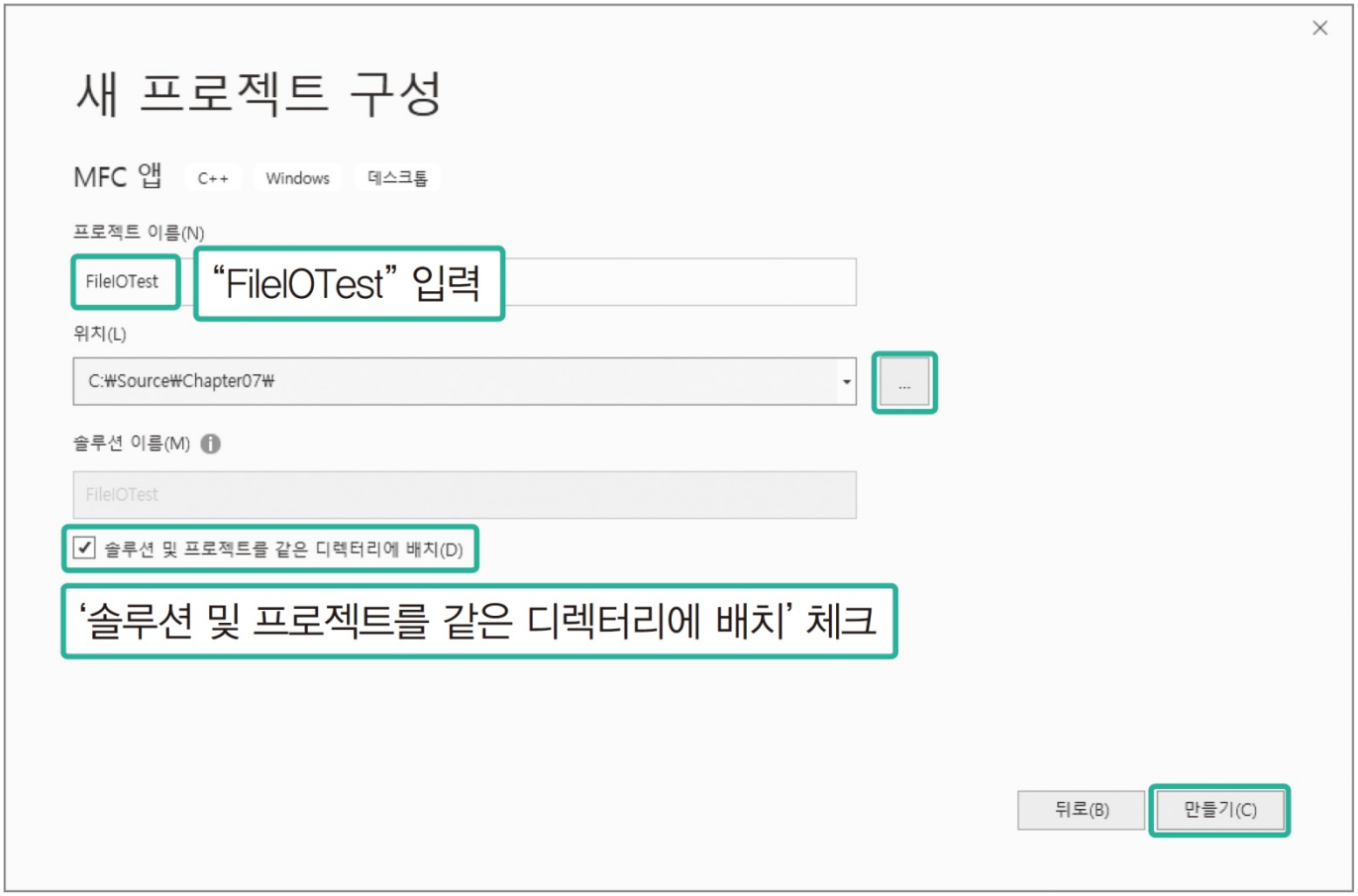


그림 7-10 프로젝트 이름과 위치 지정

[실습 7-1] MFC 응용 프로그램 생성하기(도큐먼트/뷰 구조)

표 7-5 응용 프로그램 마법사 단계별 변경 사항

단계	변경 사항
애플리케이션 종류	애플리케이션 종류로 '단일 문서' 선택 – '문서/뷰 아키텍처 지원' 선택되어 있음 프로젝트 스타일로 'MFC standard' 선택
문서 템플릿 속성	변경 없음 – 모든 기능 비활성화되어 있음
사용자 인터페이스 기능	변경 없음 – '초기 상태 표시줄'과 '클래식 도킹 도구 모음 사용' 선택되어 있음
고급 기능	'인쇄 및 인쇄 미리 보기', '공용 컨트롤 매니페스트'를 제외한 모든 옵션 해제

[실습 7-1] MFC 응용 프로그램 생성하기(도큐먼트/뷰 구조)

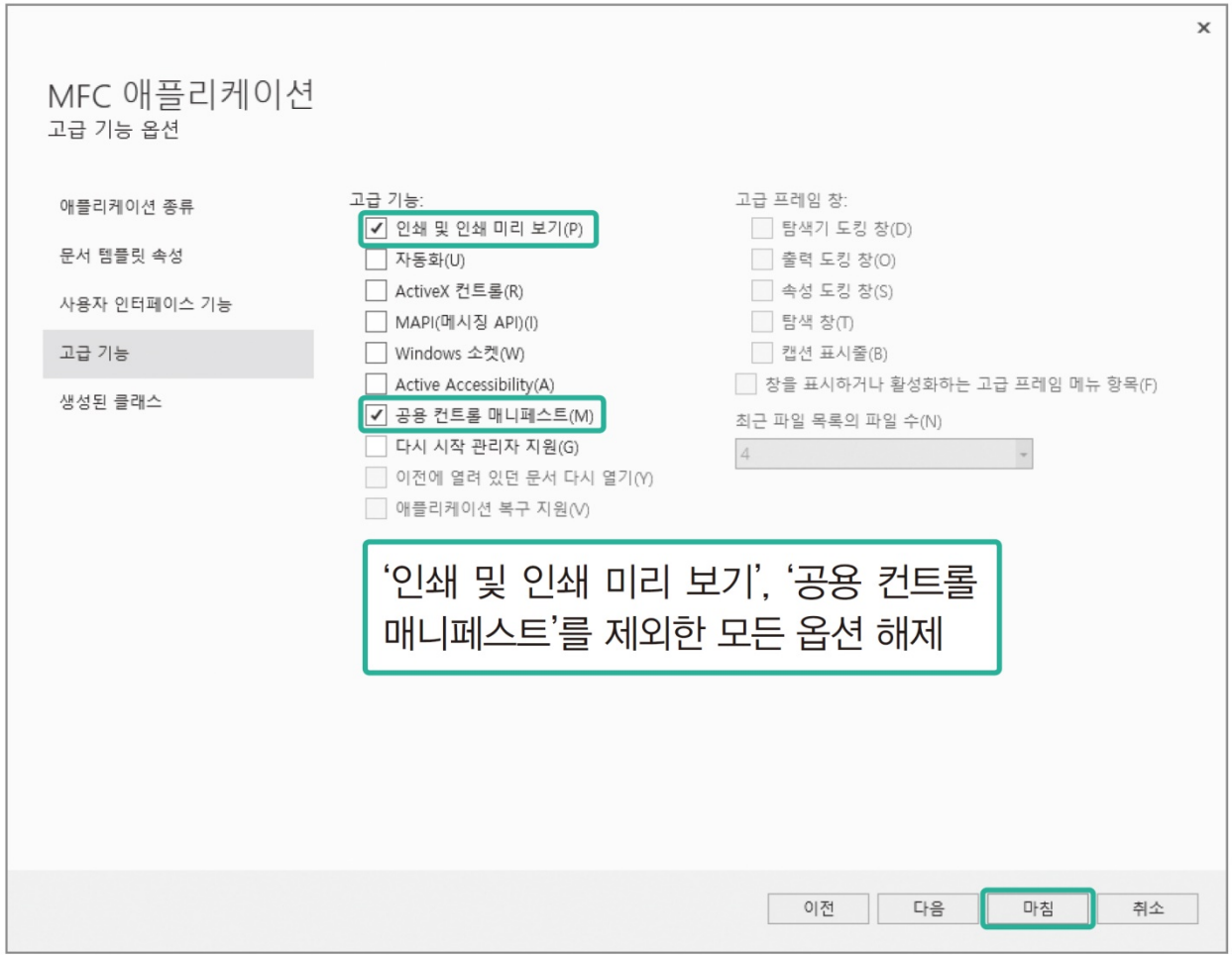


그림 7-11 고급 기능

[실습 7-1] MFC 응용 프로그램 생성하기(도큐먼트/뷰 구조)

```
void CFileIOTestView::OnDraw(CDC* pDC)
{
   CFileIOTestDoc* pDoc = GetDocument();
   ASSERT_VALID(pDoc);
   if (!pDoc)
      return;

   pDC->SetMapMode(MM_LOMETRIC);
   pDC->Rectangle(50, -50, 350, -350);
   pDC->Ellipse(500, -50, 800, -350);
}
```

[실습 7-1] MFC 응용 프로그램 생성하기(도큐먼트/뷰 구조)

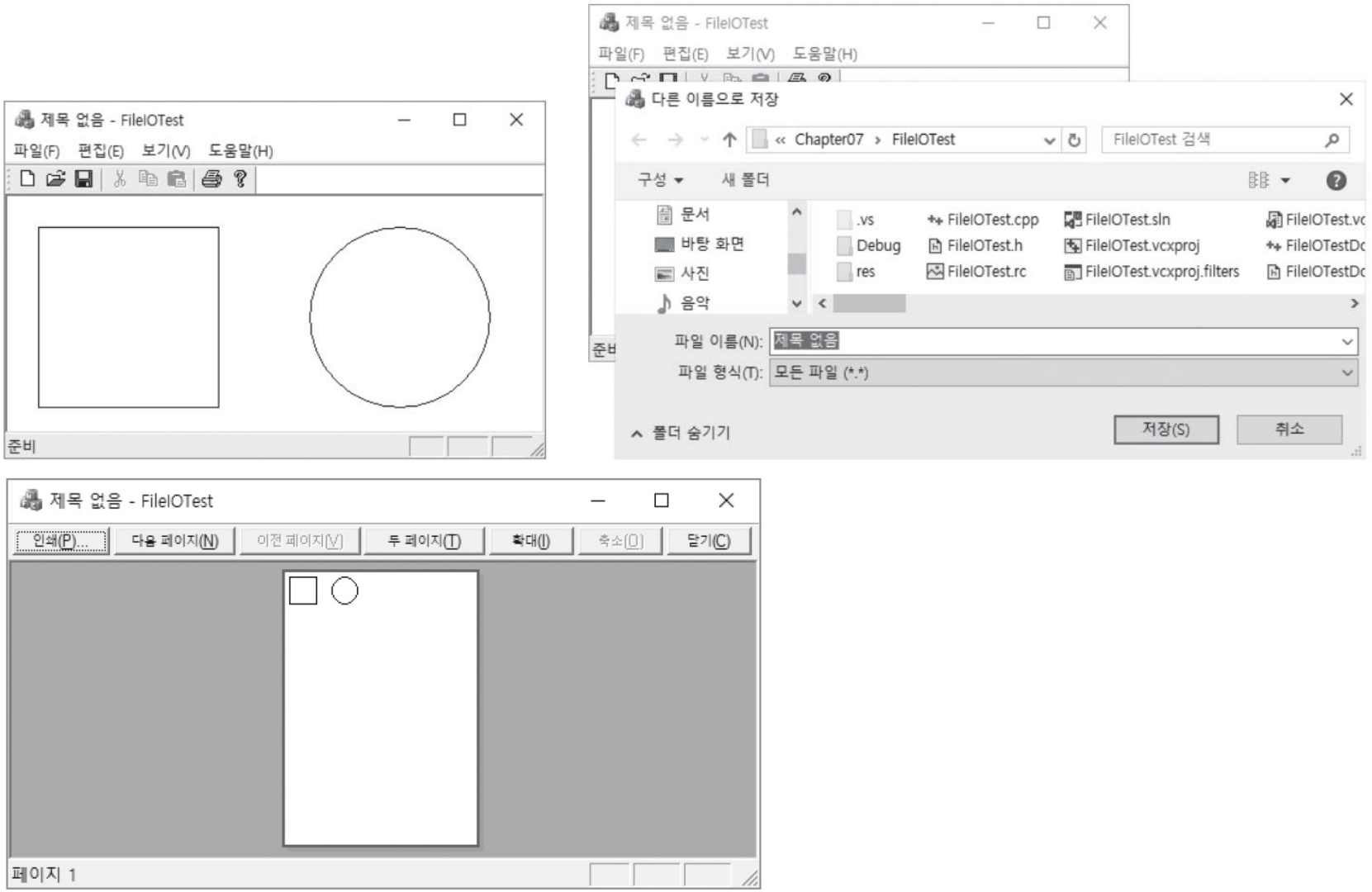


그림 7-12 실행 결과

도큐먼트/뷰 구조 분석

응용 프로그램 클래스 (1/2)

FileIOTest.cpP

```
...
BEGIN_MESSAGE_MAP(CFileIOTestApp, CWinApp)
   ON_COMMAND(ID_APP_ABOUT, &CFileIOTestApp::OnAppAbout)
   ON_COMMAND(ID_FILE_NEW, &CWinApp::OnFileNew) ❶
   ON_COMMAND(ID_FILE_OPEN, &CWinApp::OnFileOpen)
   ON_COMMAND(ID_FILE_PRINT_SETUP, &CWinApp::OnFilePrintSetup)
END_MESSAGE_MAP()
...
BOOL CFileIOTestApp::InitInstance()
{
   ...
   LoadStdProfileSettings(4); ❷

   CSingleDocTemplate* pDocTemplate; ❸
   pDocTemplate = new CSingleDocTemplate(
      IDR_MAINFRAME,
      RUNTIME_CLASS(CFileIOTestDoc),
```

도큐먼트/뷰 구조 분석

- 응용 프로그램 클래스 (2/2)

```
    RUNTIME_CLASS(CMainFrame),
    RUNTIME_CLASS(CFileIOTestView));
  if (!pDocTemplate)
    return FALSE;
  AddDocTemplate(pDocTemplate);

  CCommandLineInfo cmdInfo; ❹
  ParseCommandLine(cmdInfo);
  if (!ProcessShellCommand(cmdInfo)) ❺
    return FALSE;
  m_pMainWnd->ShowWindow(SW_SHOW); ❻
  m_pMainWnd->UpdateWindow();
  return TRUE;
}
```

도큐먼트/뷰 구조 분석

- 프레임 윈도우 클래스

MainFrm.h

```
class CMainFrame : public CFrameWnd
{
protected:
    ...
   DECLARE_DYNCREATE(CMainFrame) ❶
...
};
```

MainFrm.cpP

```
...
IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd) ❷
...
```

도큐먼트/뷰 구조 분석

MainFrm.h

뷰 클래스 (1/5)

```
class CFileIOTestView : public CView
{
protected:
  ...
  DECLARE_DYNCREATE(CFileIOTestView) ❶

public:
  CFileIOTestDoc* GetDocument() const; ❷
...
public:
  virtual void OnDraw(CDC* pDC); ❸
   ...
  protected:
    virtual BOOL OnPreparePrinting(CPrintInfo* pInfo); ❹
    virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
    virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);
...
};
```

도큐먼트/뷰 구조 분석

뷰 클래스 (2/5)

```
#ifndef _DEBUG
  inline CFileIOTestDoc* CFileIOTestView::GetDocument() const ❺
  { return reinterpret_cast<CFileIOTestDoc*>(m_pDocument); }
#endif
```

도큐먼트/뷰 구조 분석

뷰 클래스 (3/5)

FileIOTestView.cpp

```
...
IMPLEMENT_DYNCREATE(CFileIOTestView, CView) ❻

BEGIN_MESSAGE_MAP(CFileIOTestView, CView)
   ON_COMMAND(ID_FILE_PRINT, &CView::OnFilePrint) ❼
   ON_COMMAND(ID_FILE_PRINT_DIRECT, &CView::OnFilePrint)
   ON_COMMAND(ID_FILE_PRINT_PREVIEW, &CView::OnFilePrintPreview)
END_MESSAGE_MAP()
...
void CFileIOTestView::OnDraw(CDC* pDC) ❽
{
    CFileIOTestDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    if (!pDoc)
       return;
```

도큐먼트/뷰 구조 분석

뷰 클래스 (4/5)

FileIOTestView.cpp

```
    pDC->SetMapMode(MM_LOMETRIC);
    pDC->Rectangle(50, -50, 350, -350);
    pDC->Ellipse(500, -50, 800, -350);
}
...
BOOL CFileIOTestView::OnPreparePrinting(CPrintInfo* pInfo) 9
{
    return DoPreparePrinting(pInfo);
}

void CFileIOTestView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
}

void CFileIOTestView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
}
```

도큐먼트/뷰 구조 분석

뷰 클래스 (5/5)

FileIOTestView.cpp

```
...
CFileIOTestDoc* CFileIOTestView::GetDocument() const ⑩
{
   ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CFileIOTestDoc)));
   return (CFileIOTestDoc*)m_pDocument;
}
```

도큐먼트/뷰 구조 분석

도큐먼트 클래스 (1/5)

FileIOTestDoc.h

```
class CFileIOTestDoc : public CDocument
{
protected:
   CFileIOTestDoc() noexcept;
   DECLARE_DYNCREATE(CFileIOTestDoc) ❶

public:
   virtual BOOL OnNewDocument(); ❷
   virtual void Serialize(CArchive& ar); ❸
#ifdef SHARED_HANDLERS
    ...
#endif
```

도큐먼트/뷰 구조 분석

도큐먼트 클래스 (2/5)

FileIOTestDoc.h

```
public:
    virtual ~CFileIOTestDoc();
#ifdef _DEBUG
   virtual void AssertValid() const;
   virtual void Dump(CDumpContext& dc) const;
#endif

protected:
   DECLARE_MESSAGE_MAP()

#ifdef SHARED_HANDLERS
   ...
#endif
};
```

도큐먼트/뷰 구조 분석

도큐먼트 클래스 (3/5)

FileIOTestDoc.cpp

```
...
IMPLEMENT_DYNCREATE(CFileIOTestDoc, CDocument) ❹

BEGIN_MESSAGE_MAP(CFileIOTestDoc, CDocument)
END_MESSAGE_MAP()

CFileIOTestDoc::CFileIOTestDoc() noexcept
{
}

CFileIOTestDoc::~CFileIOTestDoc()
{
}
```

도큐먼트/뷰 구조 분석

도큐먼트 클래스 (4/5)

FileIOTestDoc.cpp

```
BOOL CFileIOTestDoc::OnNewDocument() ❺
{
   if (!CDocument::OnNewDocument())
      return FALSE;

   return TRUE;
}

void CFileIOTestDoc::Serialize(CArchive& ar) ❻
{
   if (ar.IsStoring())
   {
   }
   else
   {
   }
}
```

도큐먼트/뷰 구조 분석

도큐먼트 클래스 (5/5)

FileIOTestDoc.cpp

```
#ifdef SHARED_HANDLERS
...
#endif

#ifdef _DEBUG
void CFileIOTestDoc::AssertValid() const
{
   CDocument::AssertValid();
}

void CFileIOTestDoc::Dump(CDumpContext& dc) const
{
   CDocument::Dump(dc);
}
#endif //_DEBUG
```

직렬화 기초

직렬화

- 영속적인 저장 매체에 객체의 내용을 저장하거나 읽어오는 기법
- CFile 클래스를 이용하여 객체의 내용을 저장하거나 읽어오는 기능을 구현하는 것도 직렬화에 속함
- 일반적으로 MFC에서는 CArchive 클래스를 이용한 입출력 기법

직렬화 기초

- 데이터 읽기 - 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 쓰기 - 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);
```

  - pFile : 데이터를 저장하거나 읽어들일 파일과 연관된 CFile 객체
  - nMode : 객체를 저장할 것인지 혹은 읽은 것인지를 나타내는 값으로, CArchive::load 또는 CArchive::store 사용
  - nBufSize : CArchive 클래스 내부에서 사용할 버퍼 크기
  - lpBuf : 사용자 정의 버퍼의 주소 값

직렬화 기초

■ 직렬화 가능한 데이터 타입

표 7-6 직렬화 가능한 데이터 타입

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

직렬화 기초

- 직렬화 원리

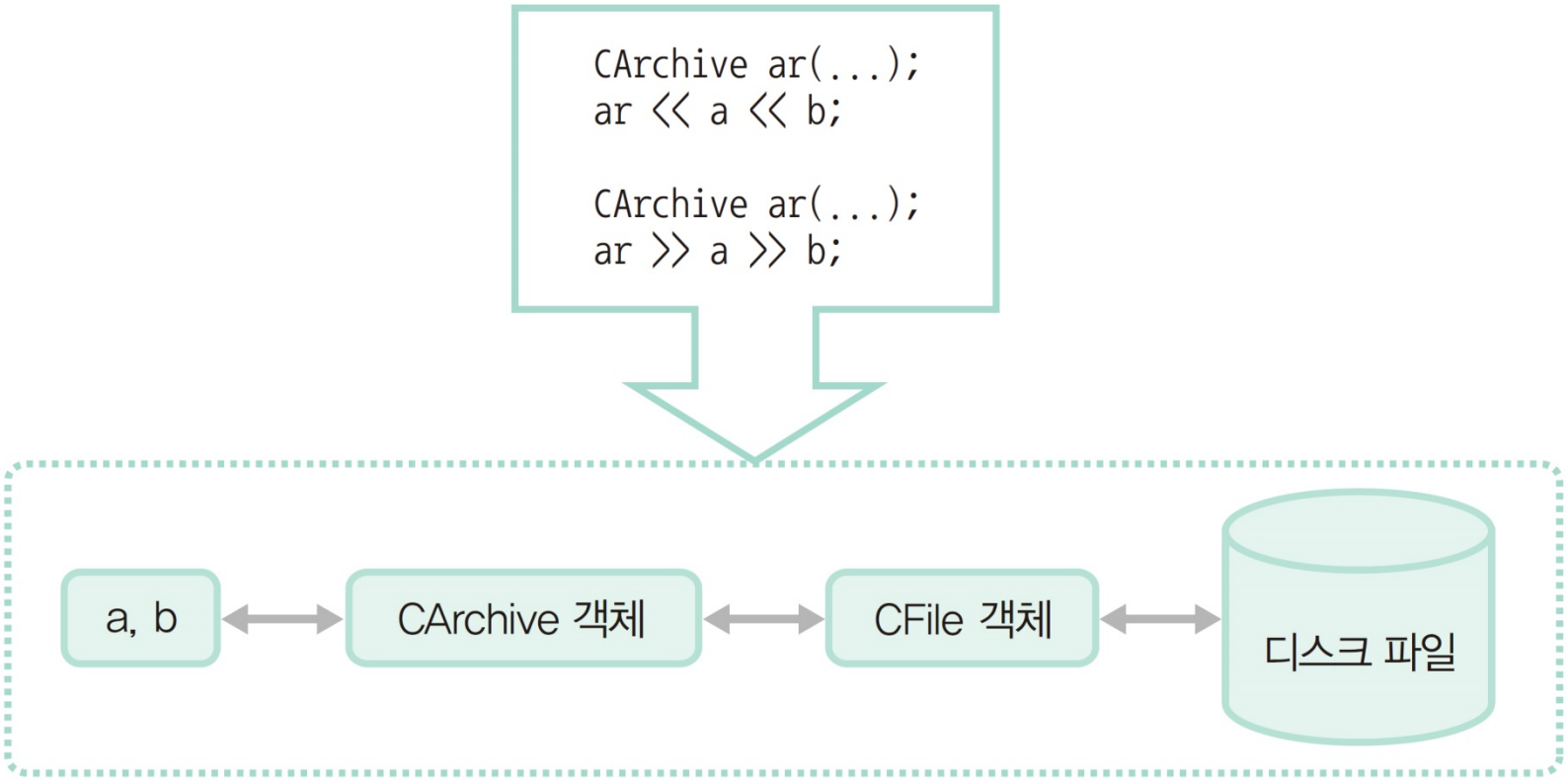


그림 7-13 직렬화 원리

도큐먼트/뷰 구조와 직렬화

- [파일]->[열기...] 메뉴를 선택한 경우

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

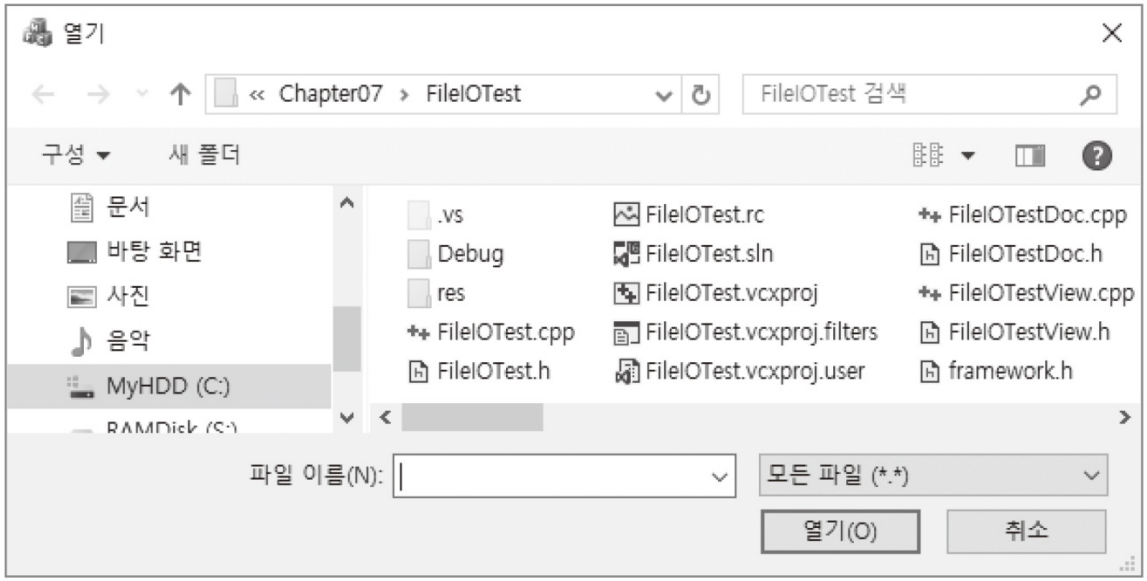


그림 7-14 파일 열기 대화 상자

도큐먼트/뷰 구조와 직렬화

- [파일]->[열기...] 메뉴를 선택한 경우

```
BOOL CDocument::OnOpenDocument(LPCTSTR lpszPathName)
{
    … (CFile 객체를 생성한다; pFile은 CFile 객체의 주소 값을 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)
```

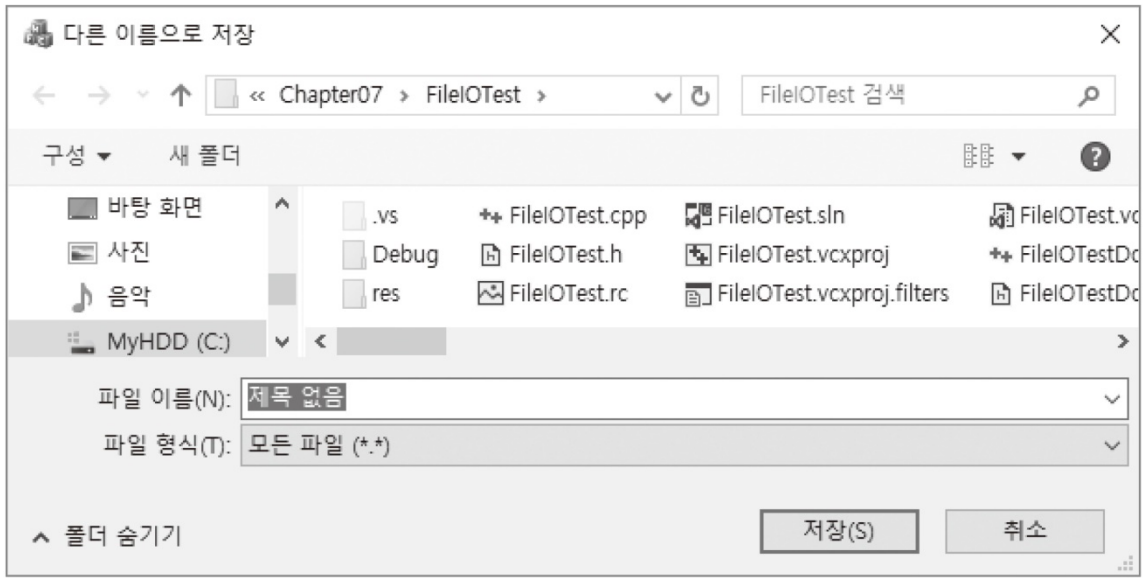


그림 7-15 파일 저장하기 대화 상자

도큐먼트/뷰 구조와 직렬화

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

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

도큐먼트/뷰 구조와 직렬화

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

```
void CFileIOTestDoc::Serialize(CArchive& ar)
{
    if (ar.IsStoring())
    {
        // TODO: 여기에 저장 코드를 추가합니다.
    }
    else
    {
        // TODO: 여기에 로딩 코드를 추가합니다.
    }
}
```

[실습 7-2] 직렬화 기능 구현하기

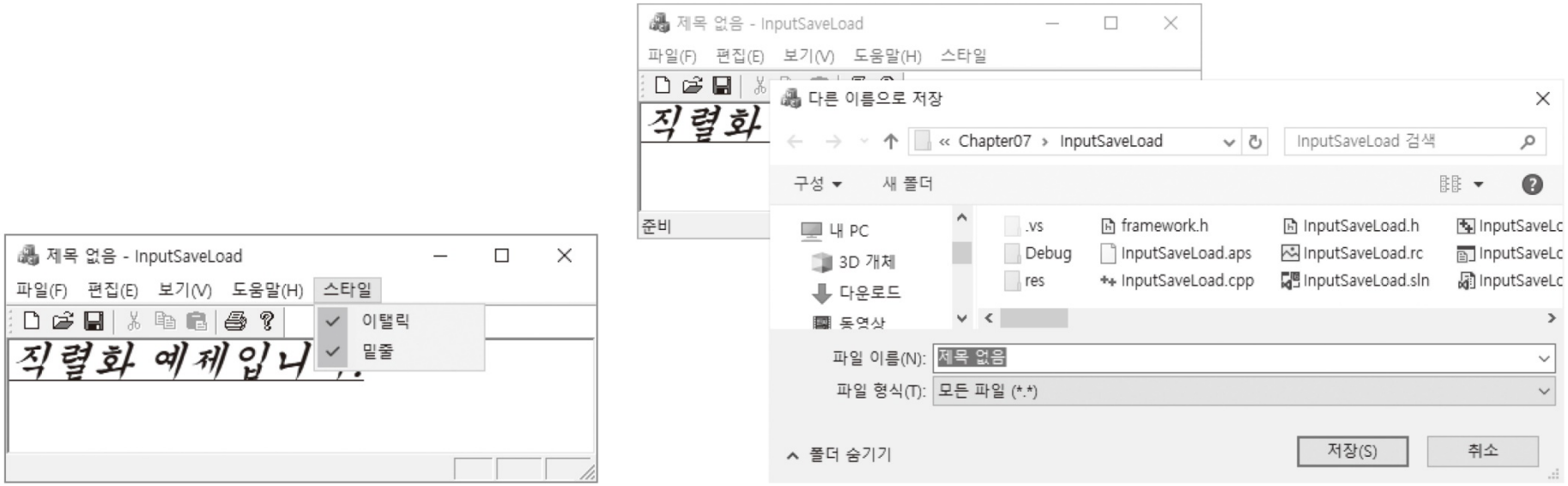


그림 7-16 실행 결과

[실습 7-2] 직렬화 기능 구현하기

```
#include <afxtempl.h>
class CInputSaveLoadDoc : public CDocument
{
protected: // serialization에서만 만들어집니다.
   CInputSaveLoadDoc();
   DECLARE_DYNCREATE(CInputSaveLoadDoc)

// 특성입니다.
public:
   CArray<TCHAR, TCHAR> m_str; // 글자 저장
   BOOL m_bItalic, m_bUnderline; // 스타일 저장
```

[실습 7-2] 직렬화 기능 구현하기

```
BOOL CInputSaveLoadDoc::OnNewDocument()
{
   if (!CDocument::OnNewDocument())
      return FALSE;

   m_str.RemoveAll();
   m_bItalic = m_bUnderline = FALSE;

   return TRUE;
}
```

[실습 7-2] 직렬화 기능 구현하기

```
void CInputSaveLoadView::OnChar(UINT nChar, UINT nRepCnt, UINT nFlags)
{
   CInputSaveLoadDoc* pDoc = GetDocument();

   // [Backspace] 입력 시 맨 마지막 글자를 삭제한다.
   if(nChar == _T('\b')){
      if(pDoc->m_str.GetSize() > 0)
         pDoc->m_str.RemoveAt(pDoc->m_str.GetSize() - 1);
   }
   // 그 밖의 경우에는 동적 배열에 글자를 추가한다.
   else{
      pDoc->m_str.Add(nChar);
   }

   // 데이터가 수정되었음을 도큐먼트 객체에 알린다.
   pDoc->SetModifiedFlag();

   // 뷰의 화면을 갱신한다.
   Invalidate();
}
```

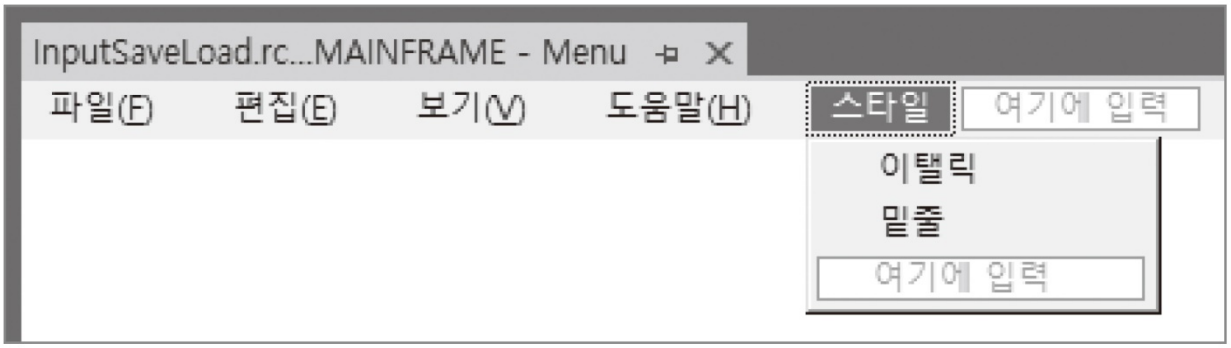
[실습 7-2] 직렬화 기능 구현하기

```
void CInputSaveLoadView::OnDraw(CDC* pDC)
{
   CInputSaveLoadDoc* pDoc = GetDocument();
   ASSERT_VALID(pDoc);
   if (!pDoc)
      return;

   // 화면 출력용 폰트를 선택한다.
   CFont font;
   font.CreateFont(30, 0, 0, 0, 0, pDoc->m_bItalic,
      pDoc->m_bUnderline, 0, 0, 0, 0, 0, 0, _T("궁서"));
   pDC->SelectObject(&font);

   // 현재까지 입력된 글자를 화면에 출력한다.
   CRect rect;
   GetClientRect(&rect);
   pDC->DrawText(pDoc->m_str.GetData(),
      pDoc->m_str.GetSize(), &rect, DT_LEFT);
}
```

[실습 7-2] 직렬화 기능 구현하기

그림 7-17 실행 결과

[실습 7-2] 직렬화 기능 구현하기



Caption	ID	나머지 속성
스타일	없음	변경 없음
이탤릭	ID_STYLE_ITALIC	변경 없음
밑줄	ID_STYLE_UNDERLINE	변경 없음

그림 7-18 '스타일' 메뉴 추가와 속성 변경

[실습 7-2] 직렬화 기능 구현하기

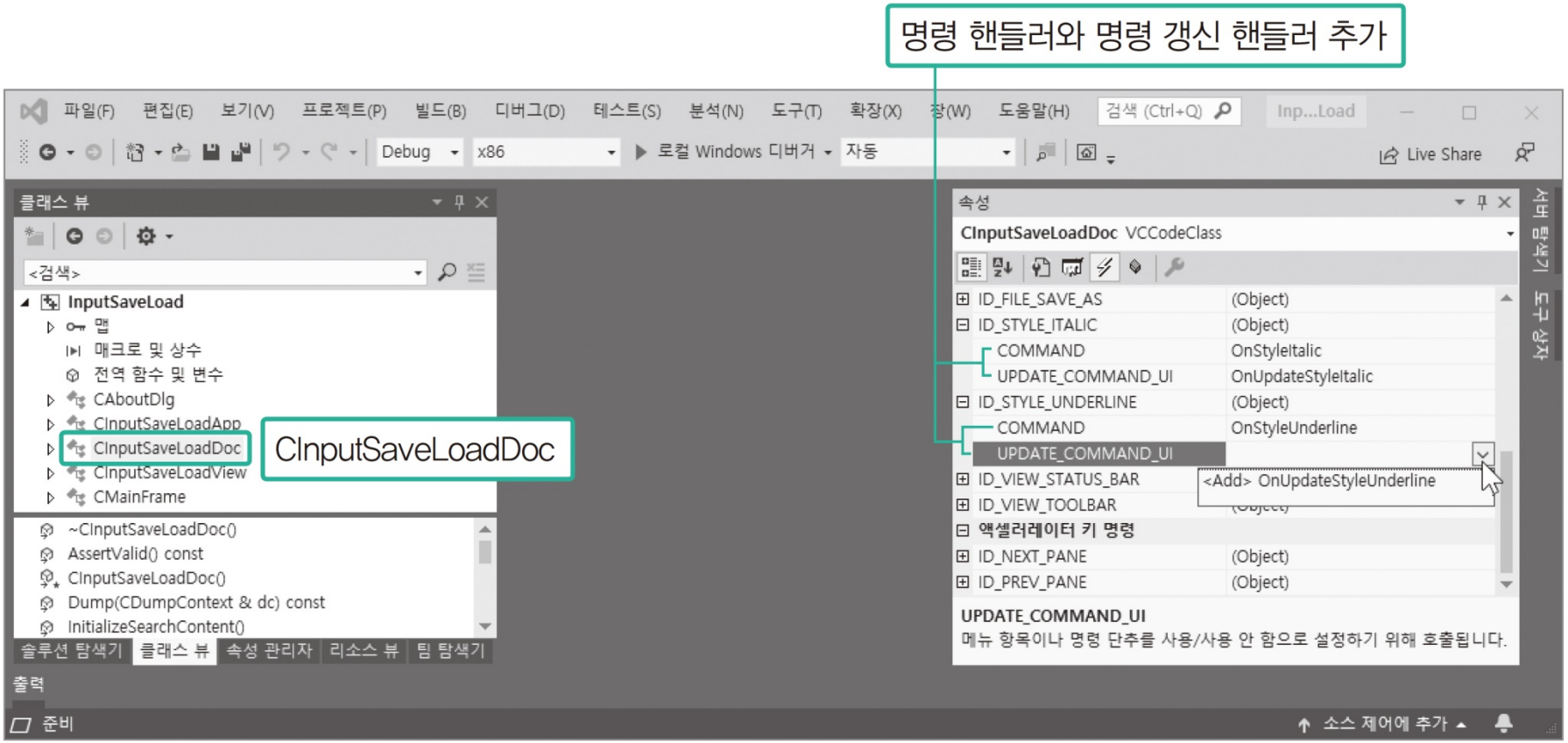


그림 7-19 명령 핸들러와 명령 갱신 핸들러 추가

[실습 7-2] 직렬화 기능 구현하기

```
void CInputSaveLoadDoc::OnStyleItalic()
{
   m_bItalic = !m_bItalic;
   SetModifiedFlag();
   UpdateAllViews(NULL); // 뷰의 화면을 갱신한다.
}

void CInputSaveLoadDoc::OnUpdateStyleItalic(CCmdUI* pCmdUI)
{
   pCmdUI->SetCheck(m_bItalic == TRUE);
}

void CInputSaveLoadDoc::OnStyleUnderline()
{
   m_bUnderline = !m_bUnderline;
   SetModifiedFlag();
   UpdateAllViews(NULL); // 뷰의 화면을 갱신한다.
}

void CInputSaveLoadDoc::OnUpdateStyleUnderline(CCmdUI* pCmdUI)
{
   pCmdUI->SetCheck(m_bUnderline == TRUE);
}
```

[실습 7-2] 직렬화 기능 구현하기

```
void CInputSaveLoadDoc::Serialize(CArchive& ar)
{
   if (ar.IsStoring())
   {
      ar << m_bItalic << m_bUnderline;
      m_str.Serialize(ar);
   }
   else
   {
      ar >> m_bItalic >> m_bUnderline;
      m_str.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 동작하지 않음

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

직렬화 클래스 구현

- 사용자 정의 클래스 변경 (2/2)

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

IMPLEMENT_SERIAL(CMyData, CObject, 1) ❺

void CMyData::Serialize (CArchive& ar) ❻
{
    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);
	}
}
```