03장 영상처리를 위한 Visual C++

- 디지털 영상 파일 포맷
- MFC AppWizard[exe]를 이용한 MFC 프로젝트 작성
- MFC를 이용한 영상처리 입.출력 프로그램 작성
- MFC를 이용한 영상 축소
- MFC를 이용한 영상 확대
- MFC를 이용한 양자화 영상처리

한빛미디어(주)

학습목표

- ✓ 영상처리에 사용되는 **RAW** 파일 포맷을 이해한다.
- ✓ **MFC AppWizard[exe]**를 이용한 영상처리 프로그램 기법을 익힌다.
- ✓ **MFC**의 기본 구조를 이해한다.

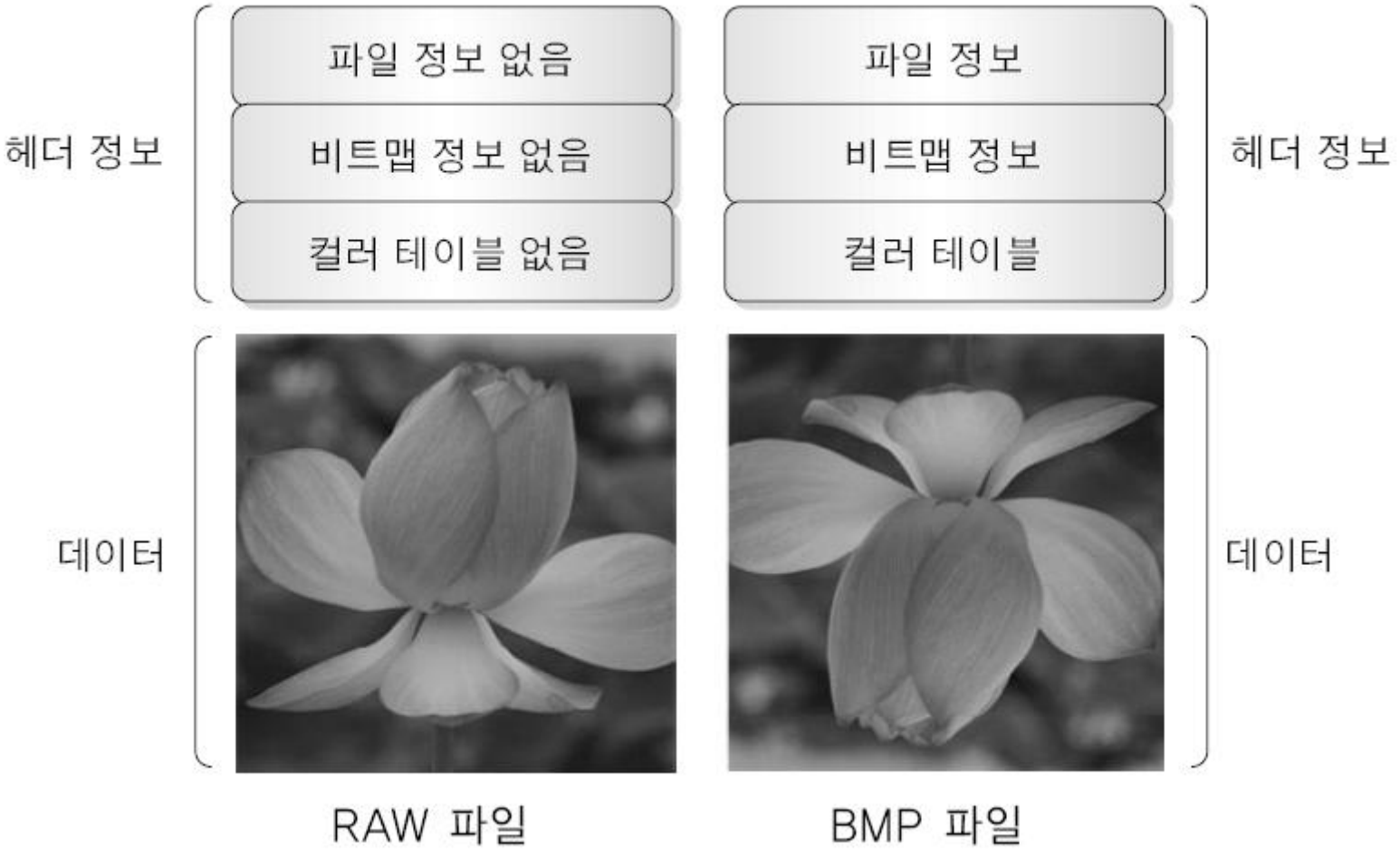
Section 01 디지털 영상 파일 포맷

- 영상 파일 포맷의 종류: **BMP, JPEG, RAW, GIF, PSD, TIFF** 등

- **BMP, JPEG** 파일 포맷
  - 영상의 색상 정보, 해상도 등을 알 수 있는 정보가 헤더**(Header)**에 포함되어 있음 → 추가 작업이 필요해 프로그램이 더 복잡해짐.

- **RAW** 파일 포맷
  - 헤더 정보 없이 완전한 데이터만으로 구성 → 복잡한 헤더 정보를 해석할 필요가 없어 영상처리가 복잡하지 않음.
  - 헤더 정보가 없어 영상의 색상 정보나 해상도 정보를 사용자가 미리 알아야 하는 단점이 있음.

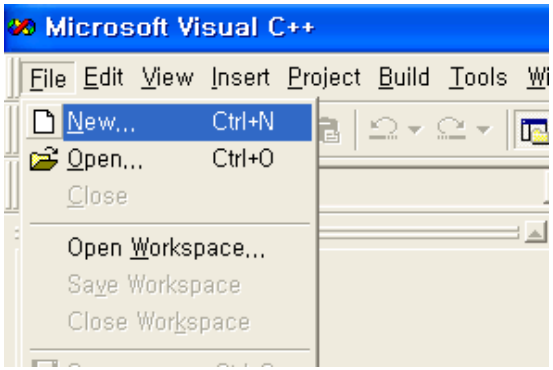
Raw와 BMP 파일 구조



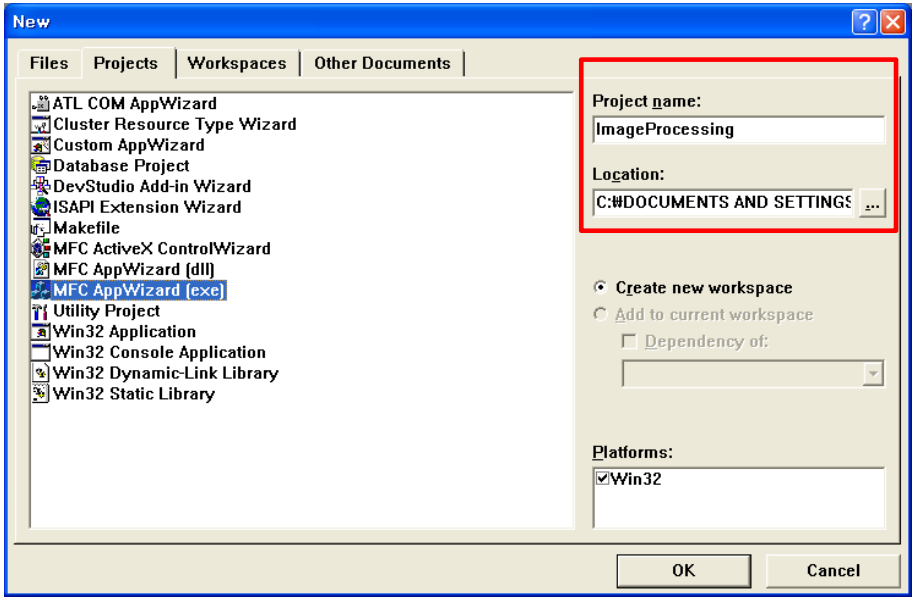
[그림 3-1] RAW와 BMP 파일 구조

[실습하기 3-1] MFC AppWizard[exe]를 이용한 MFC프로젝트 작성

① **[시작] 버튼 클릭**
→ [프로그램]-[Microsoft Visual Studio 6.0]-[Microsoft Visual C++ 6.0] 메뉴 클릭
→ Visual C++ 시작 화면에서 [File]-[New] 메뉴 클릭

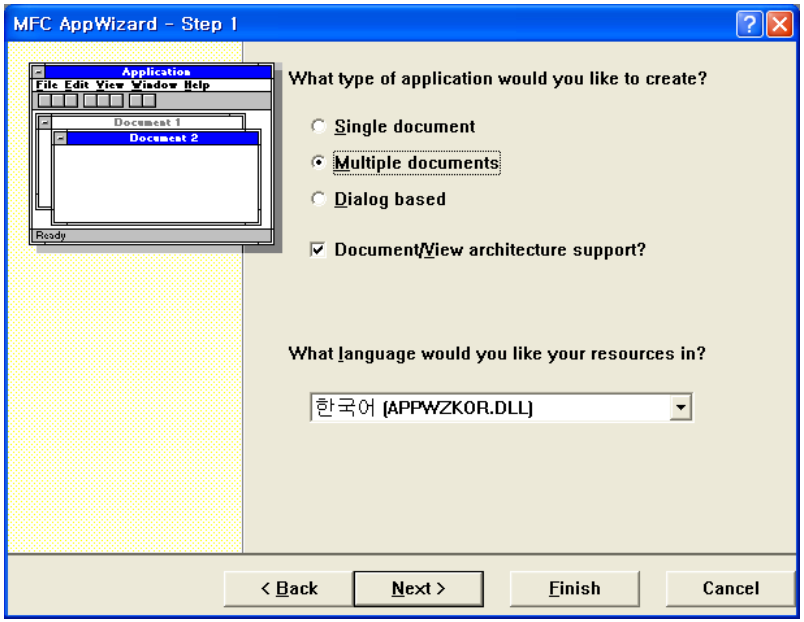


② **[New] 대화상자의 [Project] 탭에서 MFC AppWizard [exe] 항목 클릭**
→ Project name란에 작성할 프로그램 이름 입력, Location 항목에 프로그램을 저장할 위치 지정

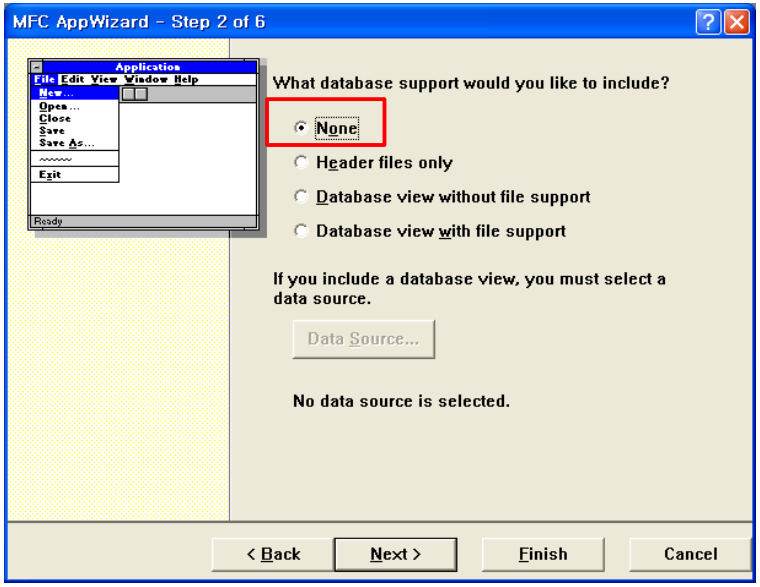


[실습하기 3-1] MFC AppWizard[exe]를 이용한 MFC프로젝트 작성

③ **[MFC AppWizard]** 대화상자에서 프로그램을 어떤 기반으로 작성할 것인지를 결정
→ 지원하는 언어 선택

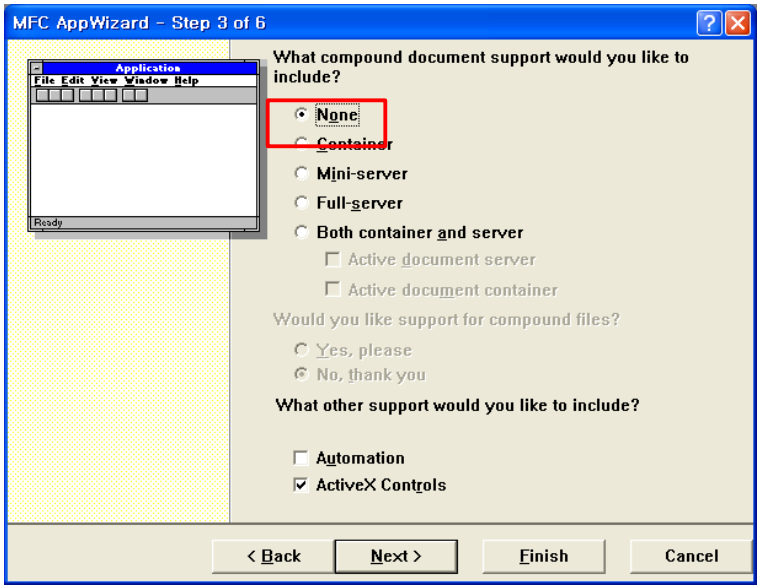


④ **[MFC AppWizard]** 대화상자의 2단계에서 데이터베이스 기능 사용 여부를 None으로 선택

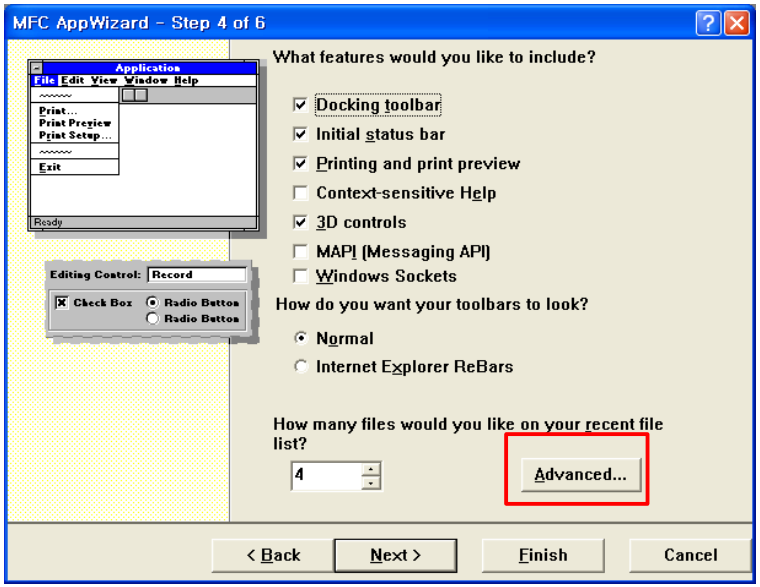


[실습하기 3-1] MFC AppWizard[exe]를 이용한 MFC프로젝트 작성

⑤ **[MFC AppWizard]** 대화상자의 3단계에서 **OLE** 기능의 사용 여부를 **None**으로 선택

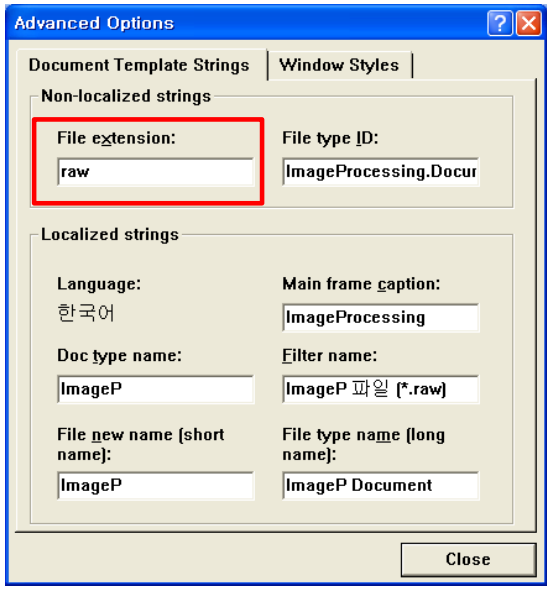


⑥ **[MFC AppWizard]** 대화상자의 4단계에서 메인 프레임의 디자인과 부가 기능을 설정하기 위해 **[Advanced]** 버튼 클릭

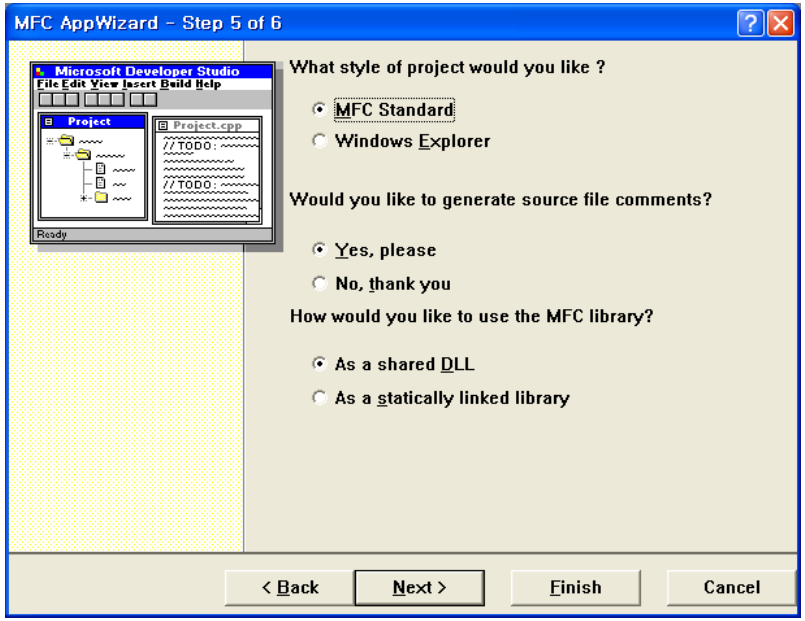


[실습하기 3-1] MFC AppWizard[exe]를 이용한 MFC프로젝트 작성

⑦ **[Advanced Option]** 대화상자에서는 불러올 영상 파일의 확장자를 디지털 영상처리에서 기본 파일 포맷인 **raw**로 지정

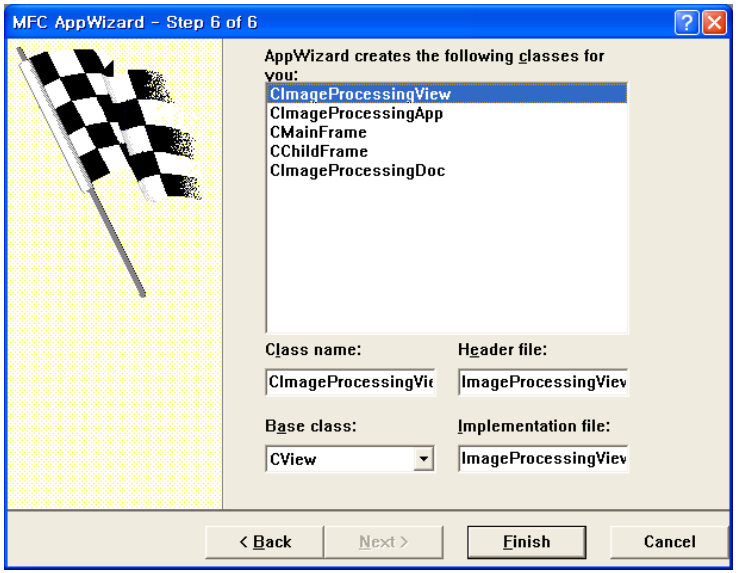


⑧ **[MFC AppWizard]** 대화상자의 5단계에서 프로그램의 스타일 설정
→ 주석문을 자동으로 삽입할 것인지를 결정

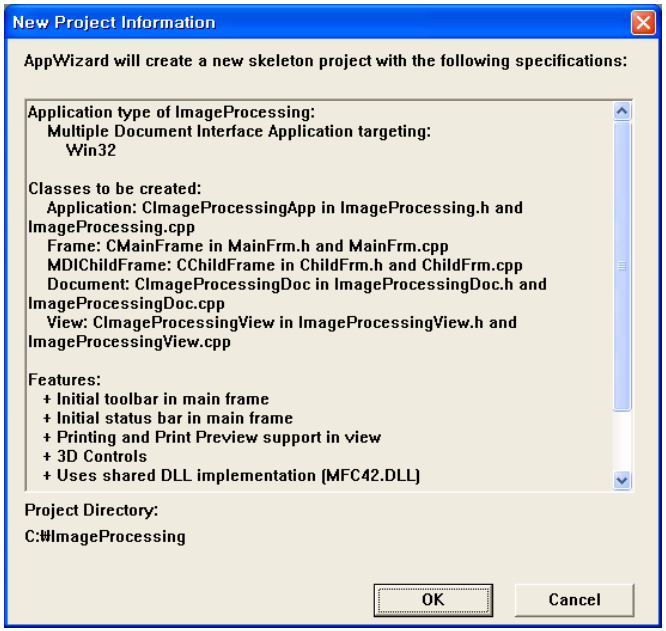


[실습하기 3-1] MFC AppWizard[exe]를 이용한 MFC프로젝트 작성

⑨ **[MFC AppWizard]** 대화상자의 마지막 6단계에서 생성된 기본 클래스의 정보 확인

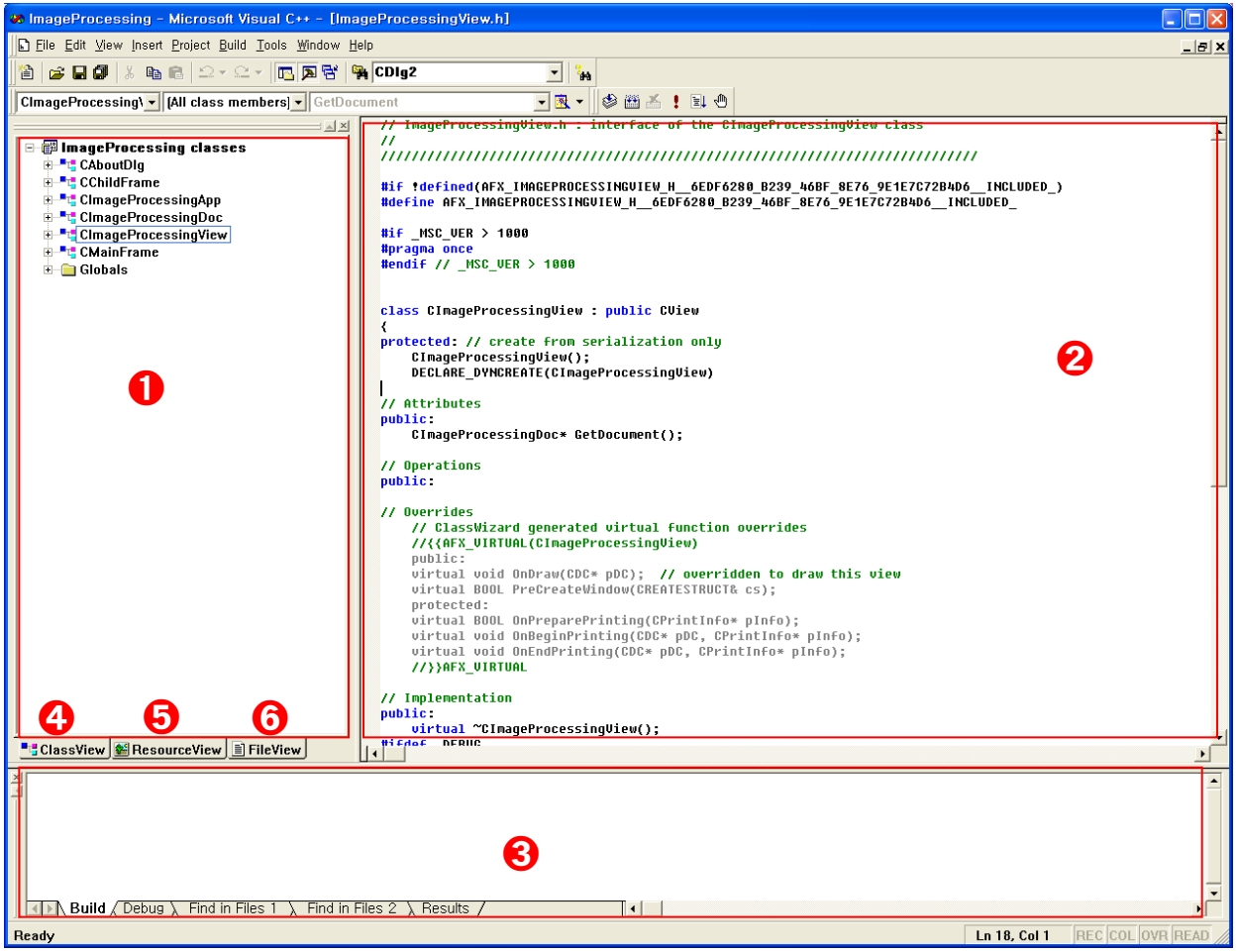


⑩ **[New Project Information]** 대화상자에서 설정 정보 확인
→ **[OK]** 버튼을 클릭



[실습하기 3-1] MFC AppWizard[exe]를 이용한 MFC프로젝트 작성

⑪ 새 창이 실행되고 **Workspace, Edit, Debug** 영역이 모두 활성화됨. 여기에서 프로그램 작성을 시작하면 됨



[실습하기 3-1] MFC AppWizard[exe]를 이용한 MFC프로젝트 작성

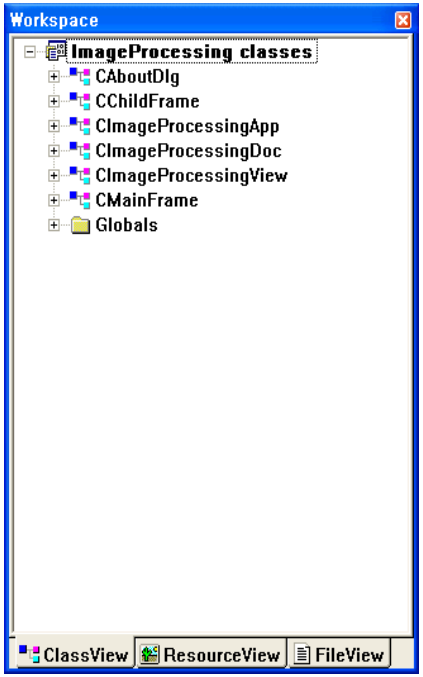
❶ **Workspace : 윈도우 탐색기와 같은 역할. 클래스, 대화상자, 파일 확인 가능**

❷ **Edit : 프로그램을 작성하는 부분**

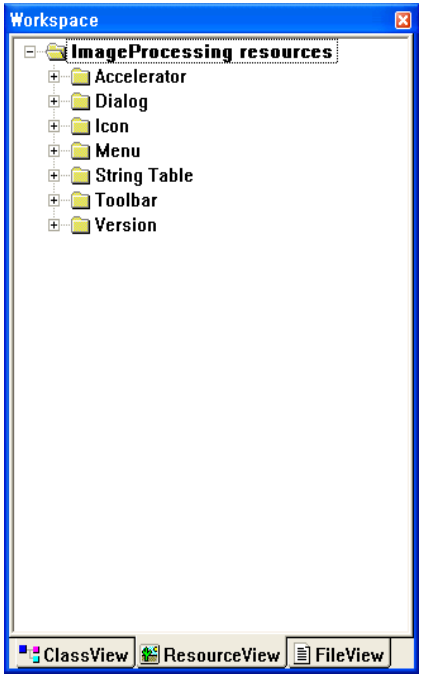
❸ **Debug : 오류 발생시 경고 및 오류 정보를 표시**

❹~ ❻ **Workspace : ClassView, ResourceView, FileView 영역으로 다시 나뉨**

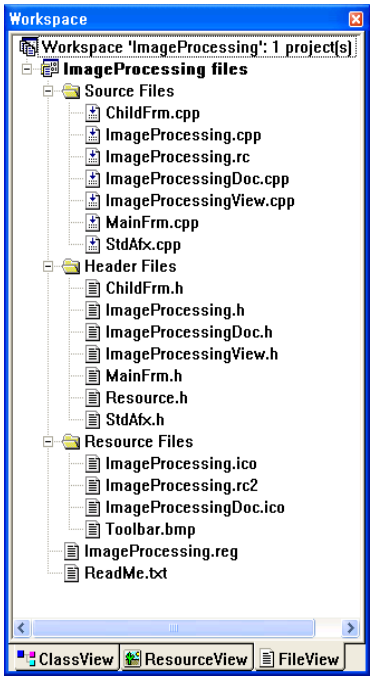
❹ **ClassView**



❺ **ResourceView**



❻ **FileView**



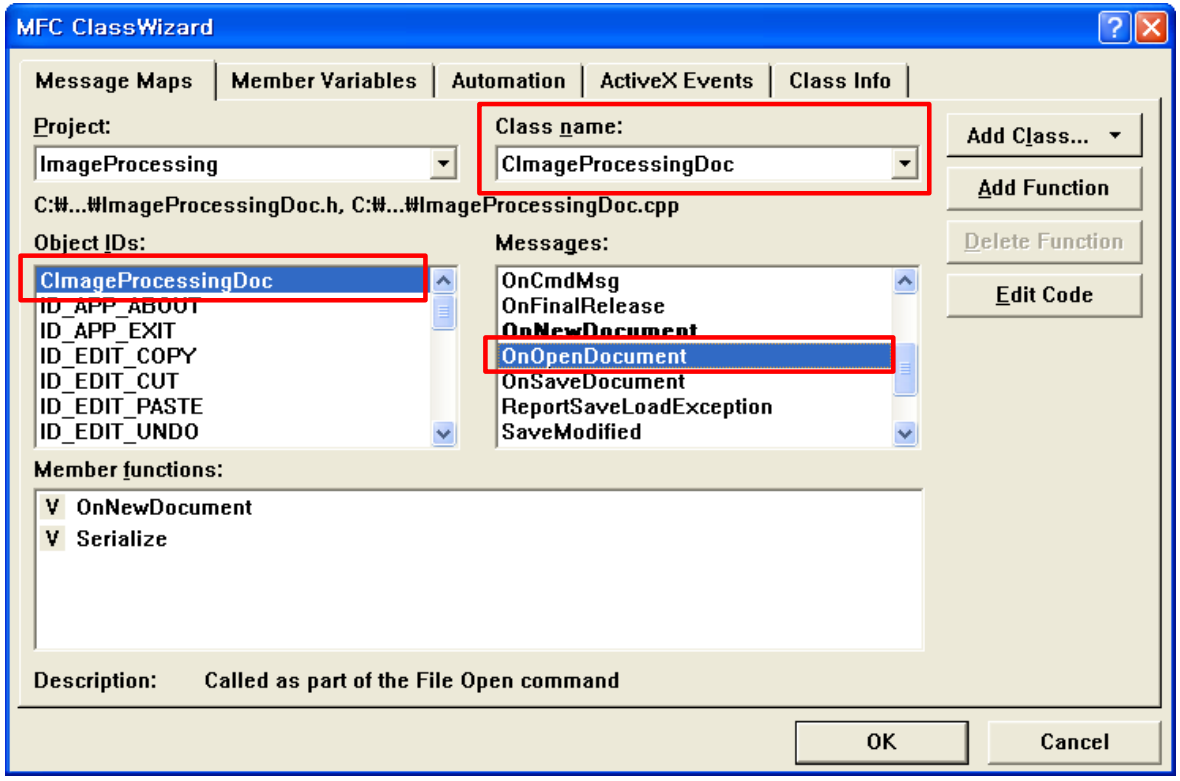
디지털 영상을 처리하려면

- 디지털 영상을 처리하려면
  - 디지털 영상을 **Visual C++**로 입력받은 뒤 일정한 루틴 따라 처리한 후 그 결과를 다시 디지털 영상 파일로 저장하는 과정이 필요.
  - **RAW** 포맷의 디지털 영상에는 **0~255**로 구성된 **8**비트 그레이 레벨이 있어 **C** 언어의 **unsigned char(0~255, 8**비트**)**로 저장 가능
  - 디지털 영상처리의 입력이나 출력 값은 모두 정수형이지만, 다양한 처리과정에서 실수 값이 나올 수도 있음. 이 때는 입력 영상의 데이터 값을 실수형**(double)**로 전환하여 영상을 처리해야 데이터가 손실되지 않음.

- **OnOpenDocument** 함수를 이용한 파일 입력
  - 영상 데이터를 파일에서 읽어 오려면 **OnOpenDocument** 함수를 재정의해야 함.
  - **OnOpenDocument** 함수는 **[**파일**]-[**열기**]** 메뉴를 클릭하면 파일을 입력받을 수 있는 대화상자를 실행시키는 역할 수행

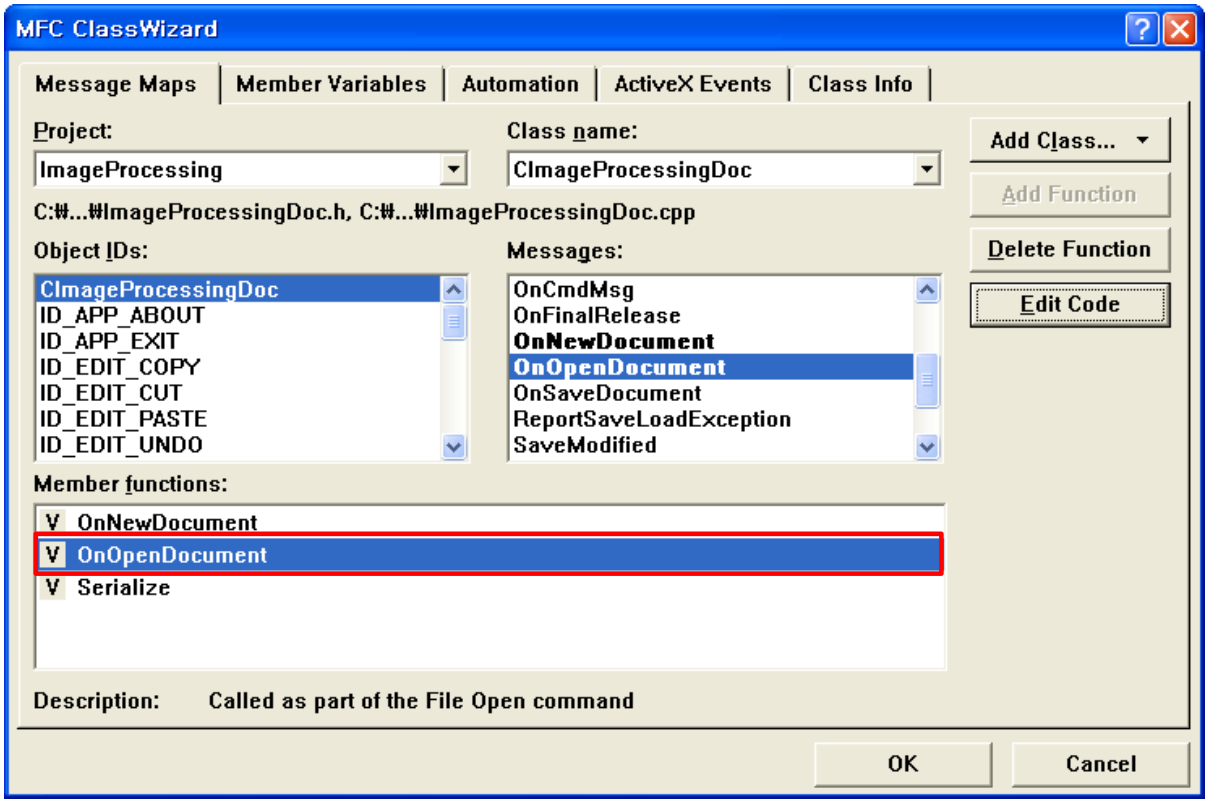
[실습하기 3-1] OnOpenDocument 함수를 이용한 파일 입력 프로그램

① **Visual C++ 프로그램에서 [View]-[ClassWizard] 메뉴를 클릭 → [MFC ClassWizard] 대화상자에서 다음과 같이 지정 → [Add Function] 버튼 클릭 → CImageProcessingDoc 클래스가 함수에 추가됨**



[실습하기 3-1] OnOpenDocument 함수를 이용한 파일 입력 프로그램

② **Member functions 항목에 OnOpenDocument 함수가 추가됐는지 확인 → [Edit Code 버튼] 버튼 → OnOpenDocument 함수로 이동**



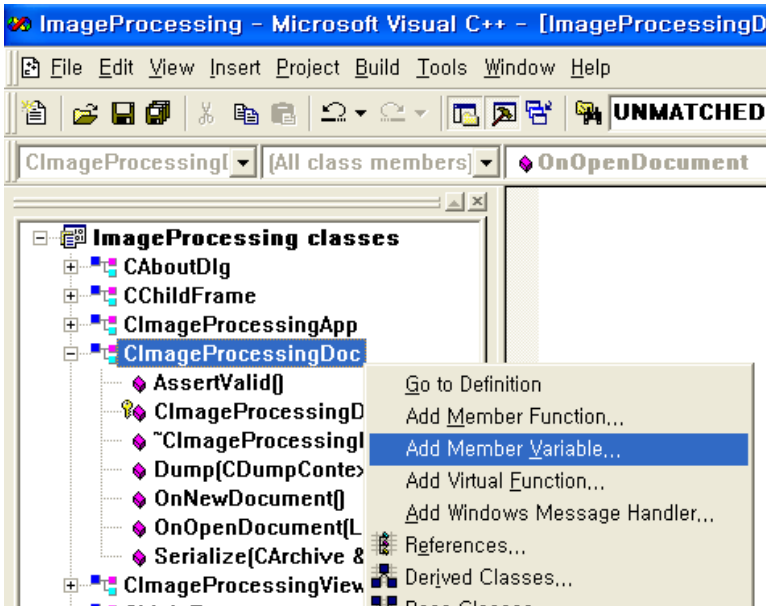
[실습하기 3-1] OnOpenDocument 함수를 이용한 파일 입력 프로그램

③ **OnOpenDocument** 함수가 다음과 같은 형태로 추가됨. 여기서 프로그램을 추가하여 영상 데이터를 입력하게 됨

```
BOOL CImageProcessingDoc::OnOpenDocument(LPCTSTR lpszPathName)
{
    if(!CDocument::OnOpenDocument(lpszPathName))
    return FALSE;
    // TODO: Add your specialized creation code here
    return TRUE;
}
```

[실습하기 3-1] OnOpenDocument 함수를 이용한 파일 입력 프로그램

④ 영상을 입력받으려면 변수를 추가해야 함. 영상 데이터를 저장하는 변수, 영상의 가로축과 세로축 크기 변수를 지정해야 함. **CImageProcessingDoc**에 변수를 추가하기 위해 **ClassView Workspace** 창으로 이동



	Variable Type	Variable Name	Access
입력 영상을 위한 포인터	unsigned char*	m_InputImage	Public
입력 영상의 가로축 크기	int	m_width	Public
입력 영상의 세로축 크기	int	m_height	Public
입력 영상의 전체 크기	int	m_size	Public

[실습하기 3-1] OnOpenDocument 함수를 이용한 파일 입력 프로그램

⑤ OnOpenDocument로 이동하여 함수 재정의

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

      CFile File; // 파일 객체 선언

      File.Open(lpszPathName, CFile::modeRead | CFile::typeBinary);
      // 파일 열기 대화상자에서 선택한 파일을 지정하고 읽기 모드 선택

      // 이 책에서는 영상의 크기 256*256, 512*512, 640*480만을 사용한다.
      if(File.GetLength() == 256*256){ // RAW 파일의 크기 결정

      m_height = 256;
      m_width = 256;
   }
   else if(File.GetLength() == 512*512){ // RAW 파일의 크기 결정
      m_height = 512;
      m_width = 512;
   }
   else if(File.GetLength() == 640*480){ // RAW 파일의 크기 결정
      m_height = 480;
      m_width = 640;
   }
   else{
      AfxMessageBox("Not Support Image Size"); // 해당 크기가 없는 경우
      return 0;
   }
   m_size = m_width * m_height; // 영상의 크기 계산

   m_InputImage = new unsigned char [m_size];
   // 입력 영상의 크기에 맞는 메모리 할당

   for(int i = 0 ; i<m_size ; i++)
      m_InputImage[i] = 255; // 초기화
   File.Read(m_InputImage, m_size); // 입력 영상 파일 읽기
   File.Close(); // 파일 닫기

   return TRUE;
}
```

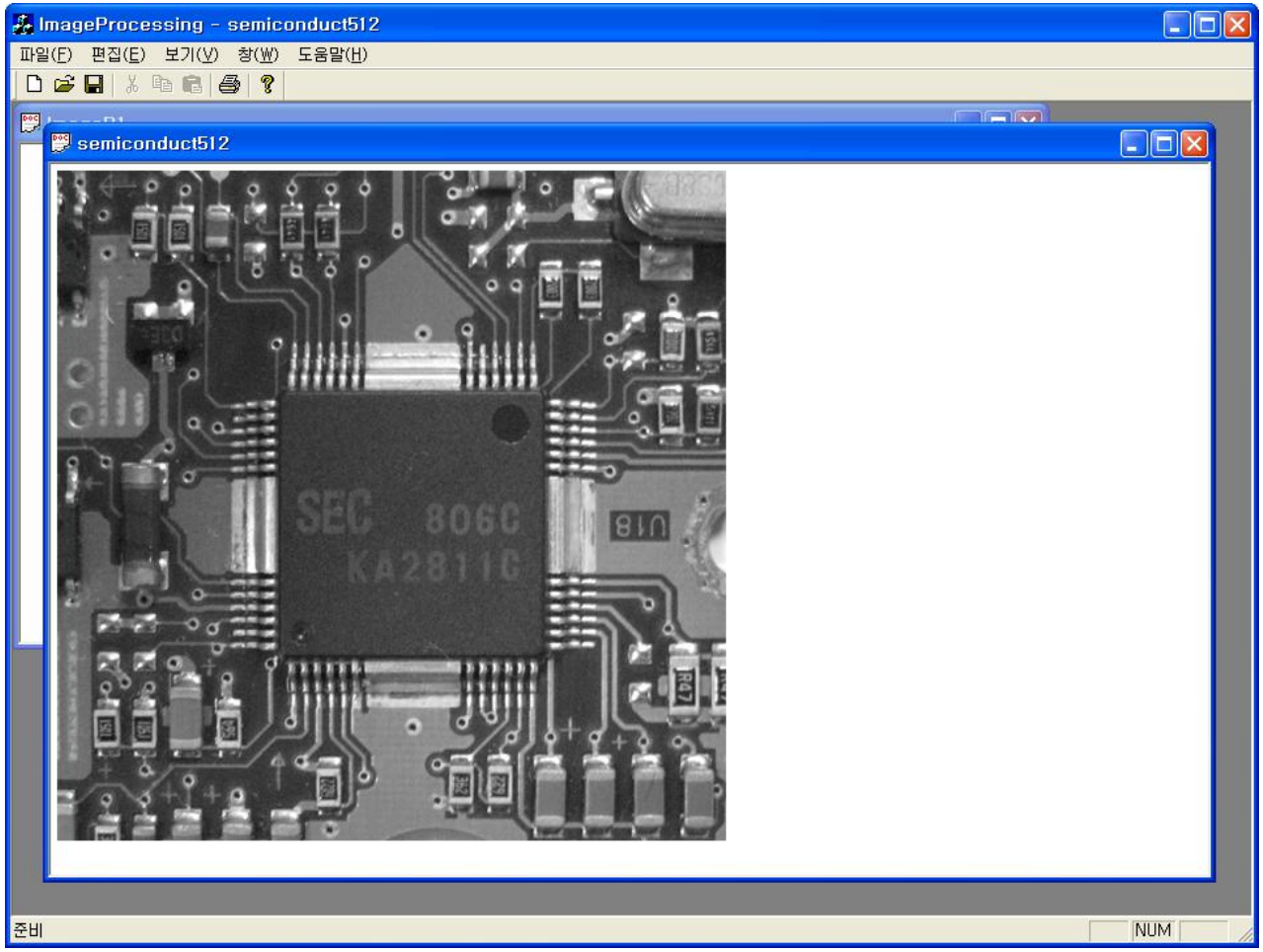
[실습하기 3-1] OnOpenDocument 함수를 이용한 파일 입력 프로그램

⑥ **CImageProcessingView** 클래스로 이동하여 **OnDraw** 함수를 재정의하여 영상 데이터를 출력함.

```
void CImageProcessingView::OnDraw(CDC* pDC)
{
   CImageProcessingDoc* pDoc = GetDocument(); // 도큐먼트 클래스 참조
   ASSERT_VALID(pDoc);
   int i, j;
   unsigned char R, G, B;

   for(i=0 ; i<pDoc->m_height ; i++){
      for(j=0 ; j<pDoc->m_width ; j++){
         R = G = B = pDoc->m_InputImage[i*pDoc->m_width+j];
         pDC->SetPixel(j+5, i+5, RGB(R, G, B));
      }
   }
}
```

[실습하기 3-1] OnOpenDocument 함수를 이용한 파일 입력 프로그램

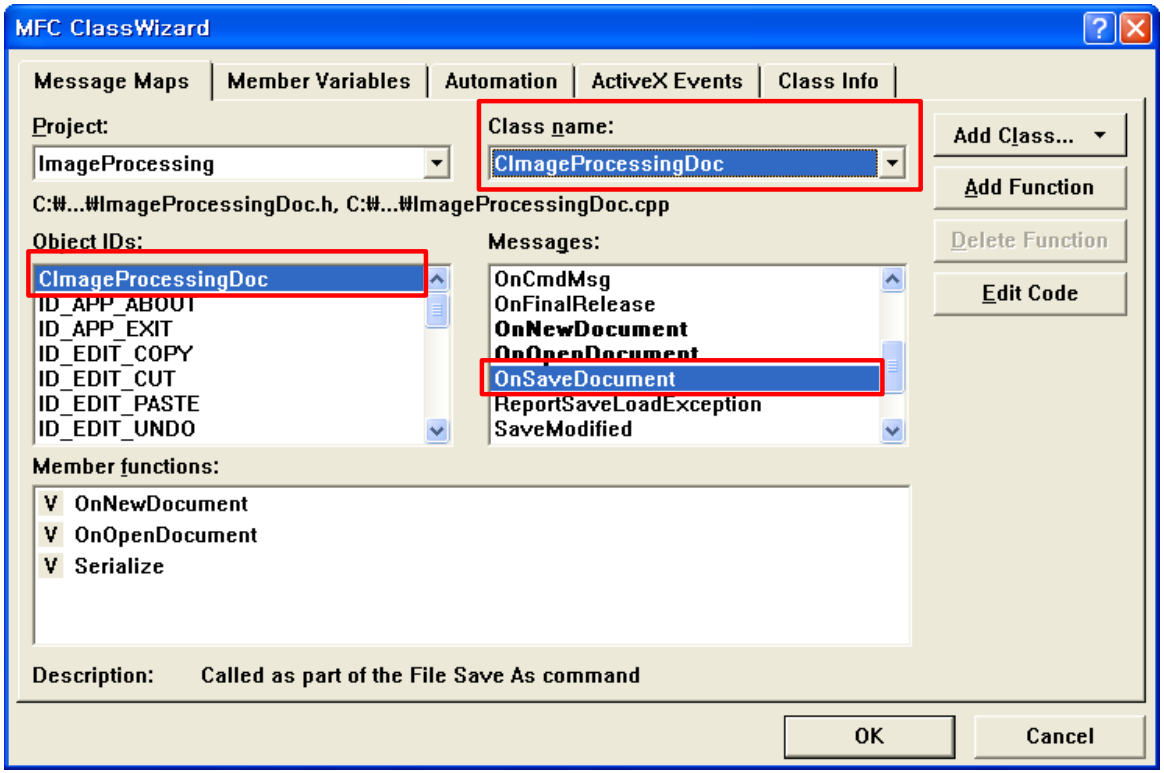
⑦ 결과 영상



- **OnSaveDocument 함수를 이용한 파일 출력**
  - **Visual C++에서 처리된 영상 데이터는 1차원이나 2차원 형태의 배열 데이터로 존재함.**
  - 이 배열 데이터를 확장자가 raw인 파일로 출력하려면 프로그램이 필요한데, 이것을 지원하는 함수가 바로 **OnSaveDocument**

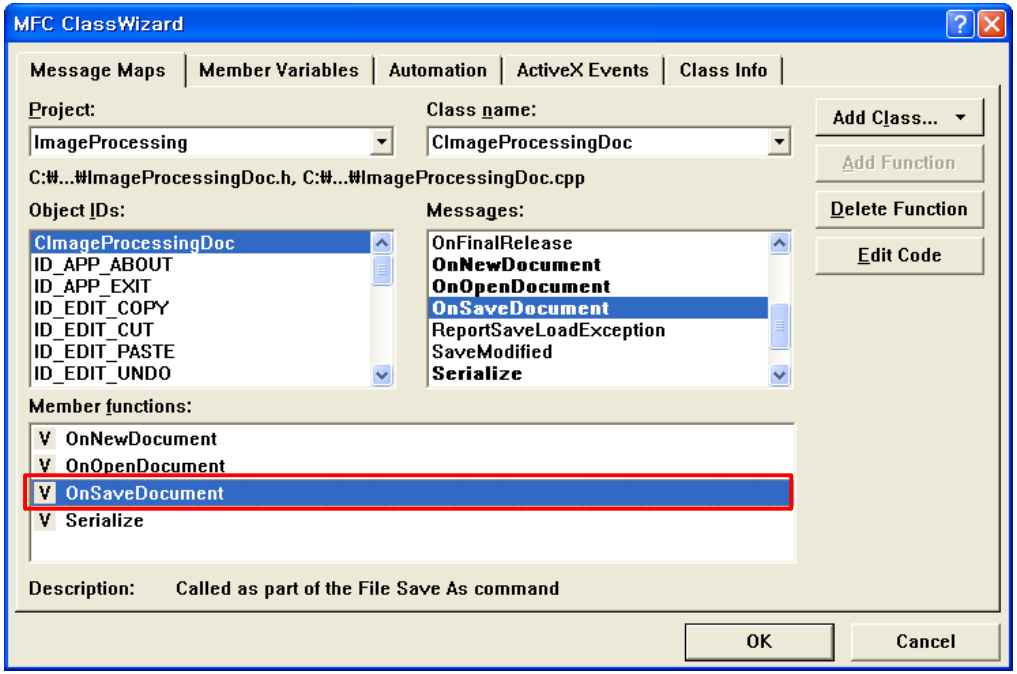
[실습하기 3-3] OnSaveDocument 함수를 이용한 파일 출력 프로그램

① Visual C++ 프로그램에서 [View]-[ClassWizard] 메뉴 클릭 → [MFC ClassWizard] 대화상자에서 다음과 같이 지정 → [Add Function] 버튼 클릭하여 OnSaveDocument 클래스를 함수에 추가



[실습하기 1-1] OnSaveDocument 함수를 이용한 파일 출력 프로그램

② Member functions 항목에 OnSaveDocument가 추가된 것 확인 → [Edit Code] 버튼을 클릭해 OnSaveDocument 함수 이동



③ OnSaveDocument 함수가 다음과 같은 형태로 추가됨. 여기에 프로그램을 추가하여 영상 데이터를 파일로 출력함.

```
BOOL CImageProcessingDoc::OnSaveDocument(LPCTSTR lpszPathName)
{
  // TODO: Add your specialized code here and/or call the base class
  return CDocument::OnSaveDocument(lpszPathName);
}
```

[실습하기 3-3] OnSaveDocument 함수를 이용한 파일 출력 프로그램

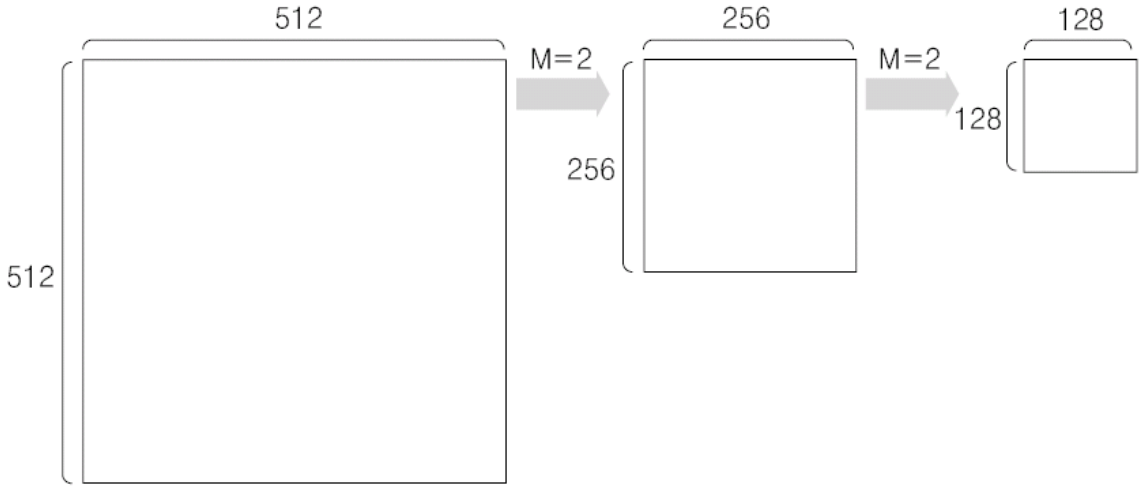
④ **OnSaveDocument 함수를 재정의함.**

```
BOOL CImageProcessingDoc::OnSaveDocument(LPCTSTR lpszPathName)
{
  CFile File; // 파일 객체 선언
  CFileDialog SaveDlg(FALSE, "raw", NULL, OFN_HIDEREADONLY);
  // raw 파일을 다른 이름으로 저장하기를 위한 대화상자 객체 선언

    if(SaveDlg.DoModal() == IDOK){
    // DoModal 멤버 함수에서 저장하기 수행
    File.Open(SaveDlg.GetPathName(), CFile::modeCreate |
    CFile::modeWrite);
    // 파일 열기
    File.Write(m_InputImage, m_size); // 파일 쓰기
    File.Close(); // 파일 닫기
    }

  return TRUE;
}
```

⑤ [저장] 버튼을 클릭 → [다른 이름으로 저장] 대화상자에서 저장할 파일 이름을 입력해 프로그램 저장

다운 샘플링(Down Sampling)

- 디지털 영상을 축소하는 가장 간단한 방법
- 다운 샘플링은 원 영상의 값을 일정한 좌표 단위로 버리는 것. 디지털 영상은 2차원이므로 수평축 샘플링과 수직축 샘플링이 모두 되어야 함.



1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

M=2

1	3
9	11

[그림 3-2] 디지털 영상의 다운 샘플링 예

[실습하기 3-4] 영상 축소 프로그램

① **Visual C++ 프로그램의 Workspace 창에서 [ResourceView] 탭 클릭**
→ [Menu]-[IDR_IMAGEPTYPE] 메뉴를 더블클릭해 [도움말] 메뉴 오른쪽에 프레임 메뉴를 하나 만듦
→ 프레임 메뉴 위에서 마우스 오른쪽 버튼을 누르 후[Properties] 메뉴 클릭
→ [Menu Item Properties] 대화상자에서 다음과 같이 설정
→ [Enter] 클릭

Caption	영상처리

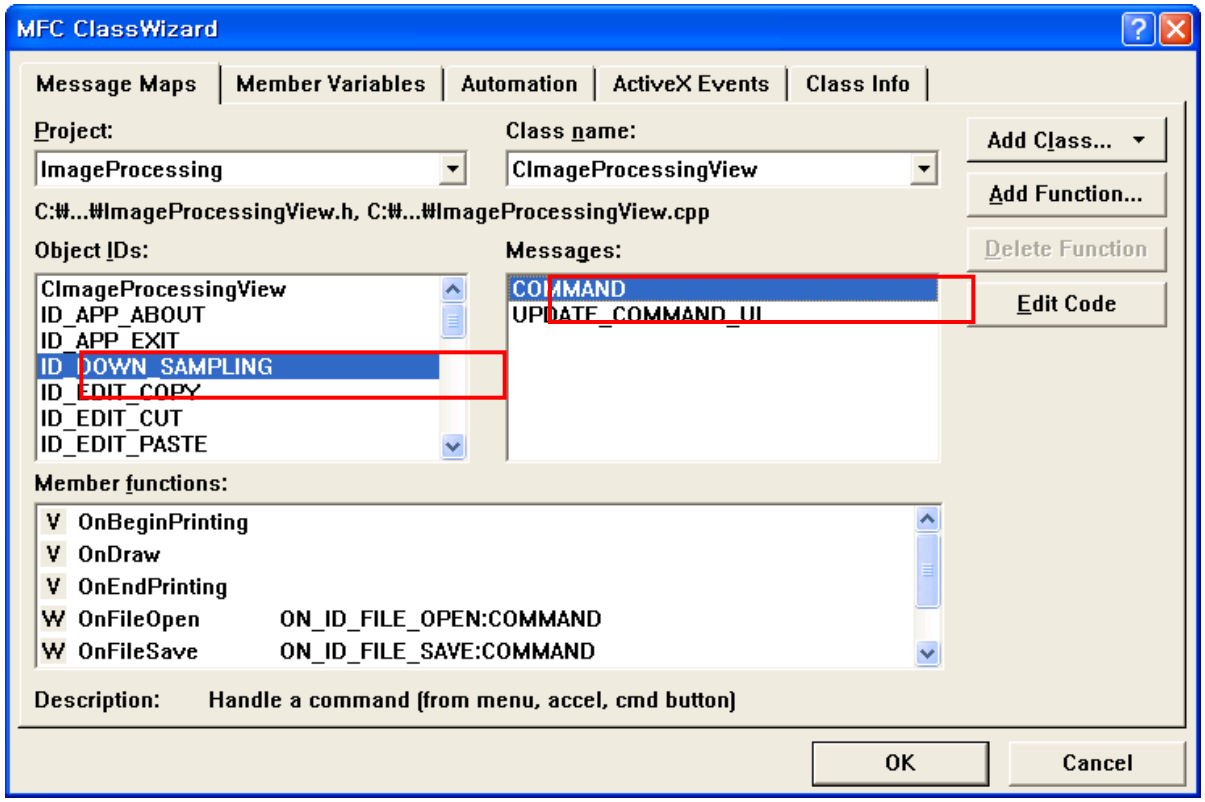
[실습하기 3-4] 영상 축소 프로그램

→ [영상처리] 메뉴 아래에 상자 선택 후 바로가기 메뉴 [Properties] 클릭
→ [Menu Item Properties] 대화상자에서 ID와 Captions를 입력
→ 생성된 [DownSample] 메뉴에서 바로가기 메뉴 [ClassWizard] 클릭

ID	ID_DOWN_SAMPLING
Caption	DownSample

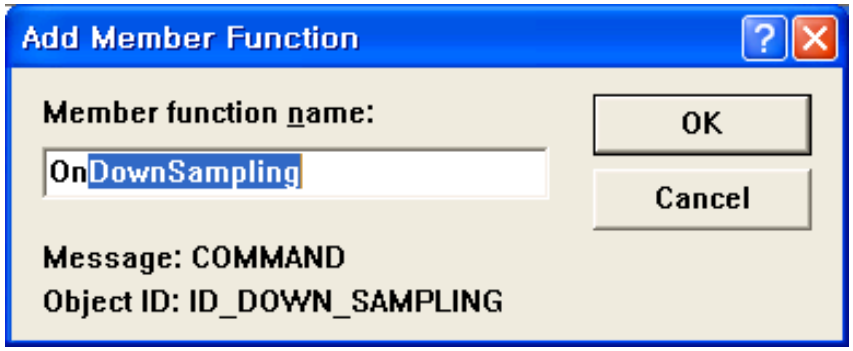
[실습하기 3-4] 영상 축소 프로그램

→ [MFC ClassWizard] 대화상자에서 Object Ids 항목을 ID_DOWN_SAMPLING, Message 항목을 COMMAND로 선택 → <Add Function> 버튼 클릭

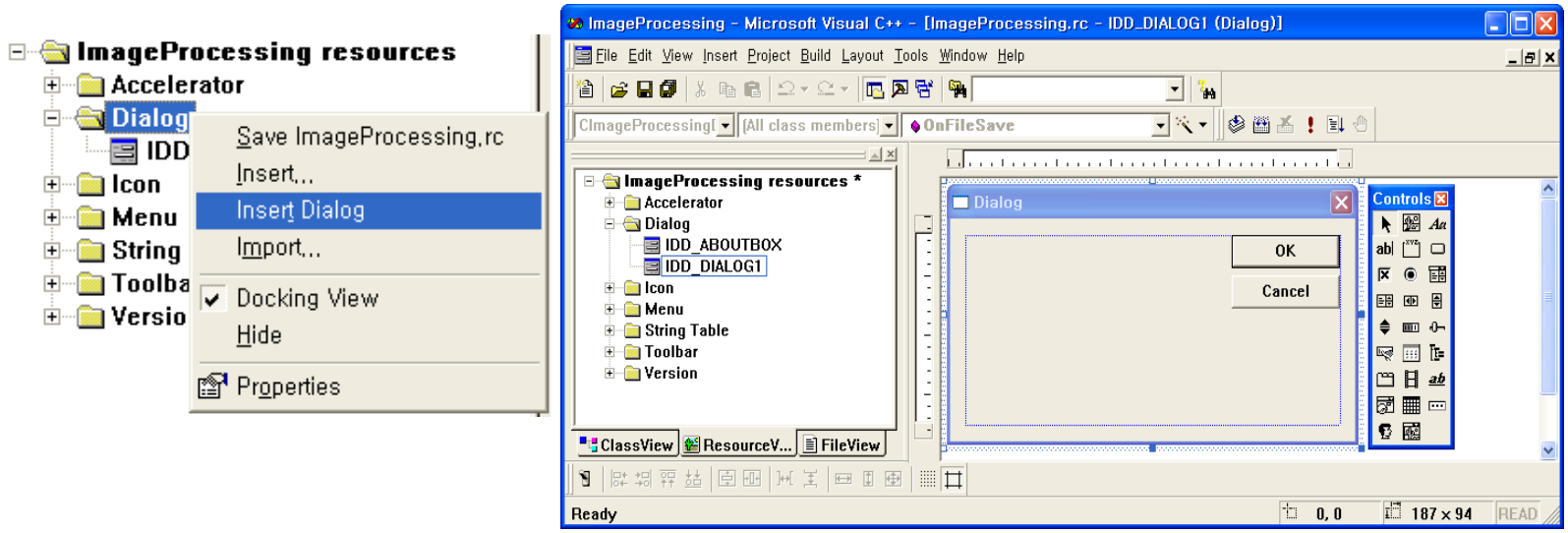


[실습하기 3-4] 영상 축소 프로그램

② **[Add Member Fuction]** 대화상자의 **View Class**에서 다운 샘플링을 담당할 함수가 생성됐는지 확인 → **[OK]** 버튼 클릭 → 다시 **[MFC ClassWizard]** 대화상자에서 **[Edit Code]** 버튼을 클릭해 함수로 이동

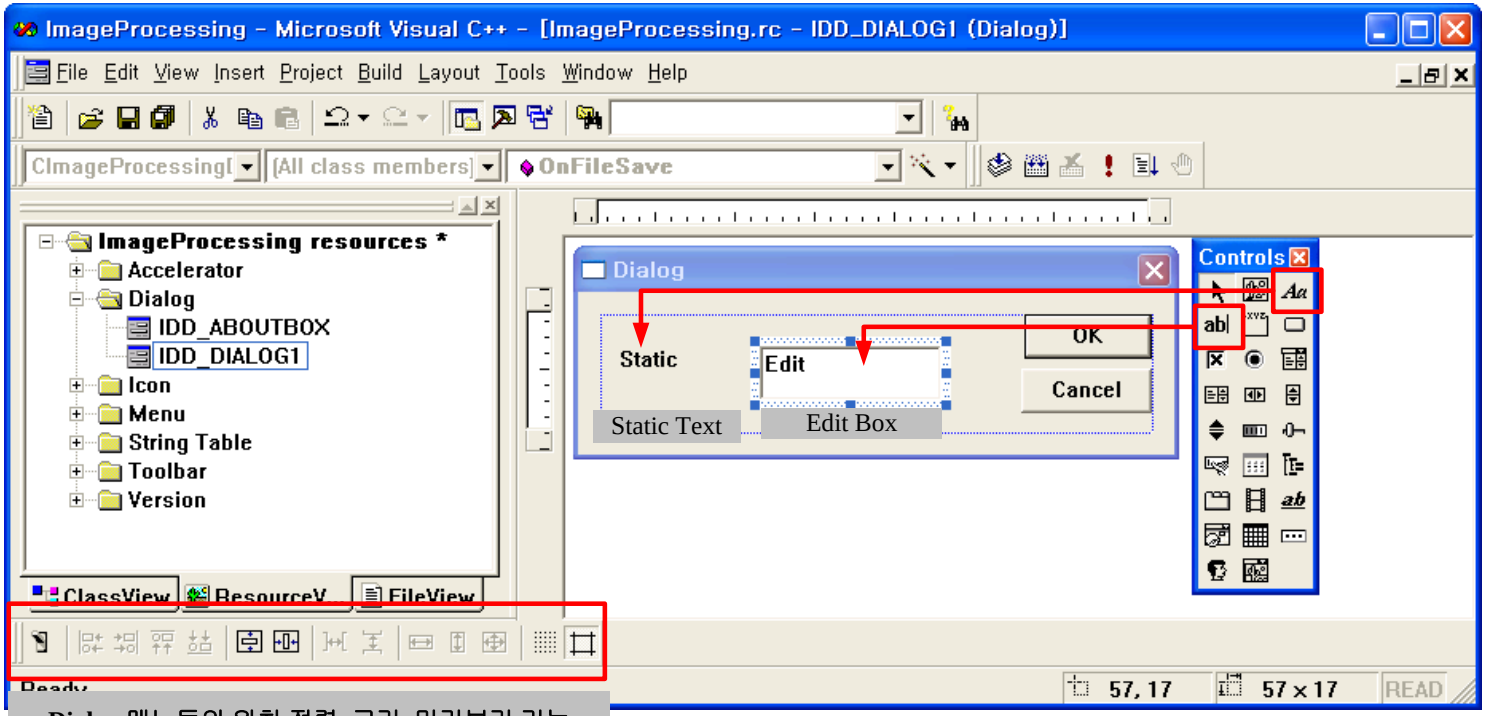


③ **ResourceView** 창에서 **[ImageProcessing resources]-[Dialog]** 폴더 선택 → 바로가기 메뉴 **[Insert Dialog]** 클릭 → 새 **[Dialog]** 대화상자가 추가됨.



[실습하기 3-4] 영상 축소 프로그램

④ **Controls 도구 상자를 이용하여 대화상자에 샘플링 비율을 입력할 수 있도록 대화상자 편집**

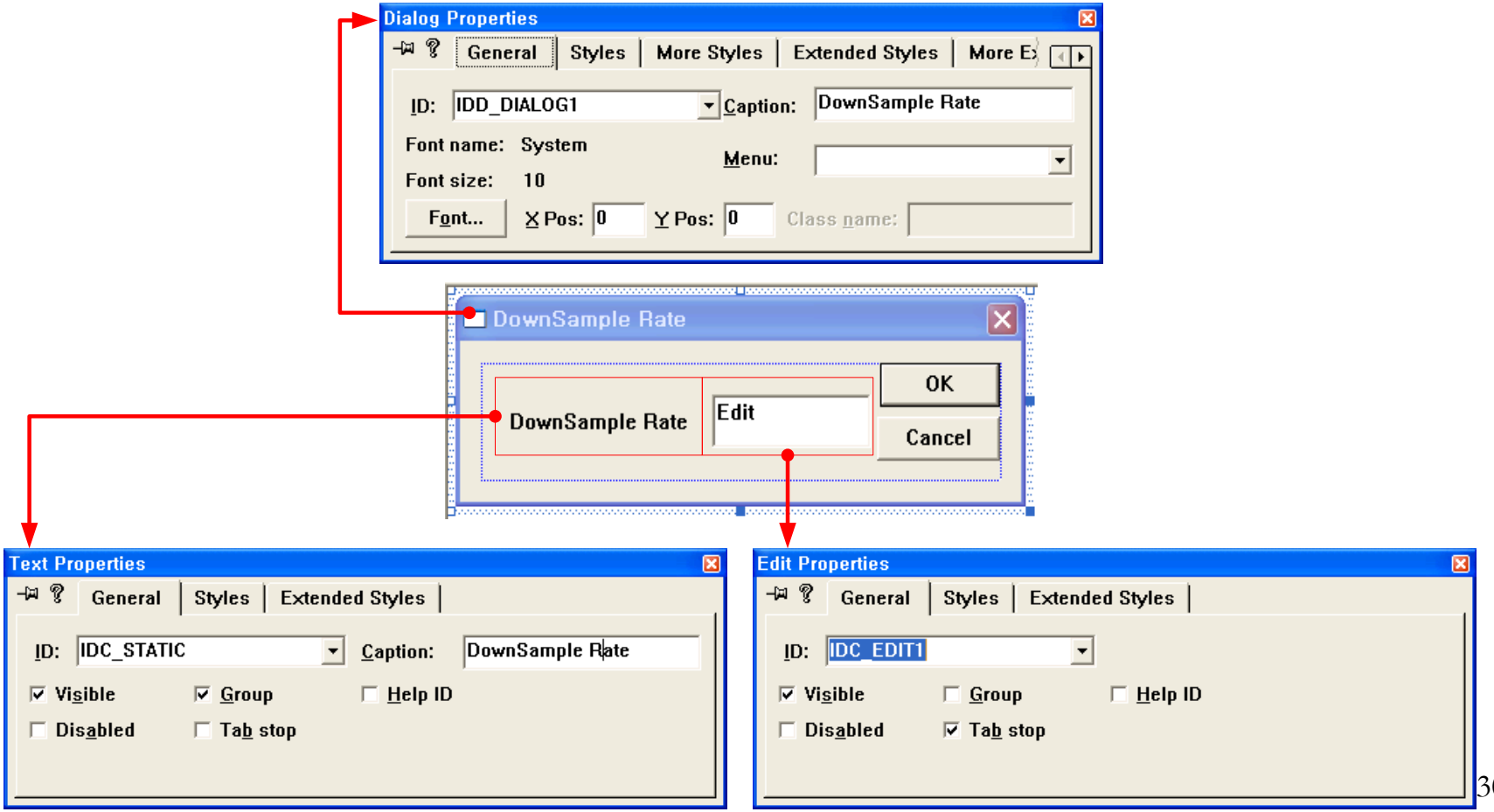


Dialog 메뉴들의 위치 정렬, 크기, 미리보기 기능

[실습하기 3-4] 영상 축소 프로그램

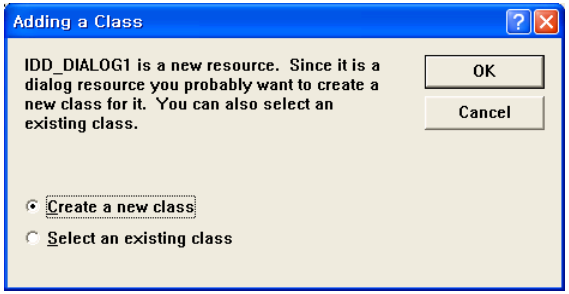
→ 대화상자에 삽입된 각 항목의 속성(Properties)을 다음과 같이 설정

	ID	Caption
Dialog Properties	IDD_DIALOG1	DownSample Rate
Text Properties	IDC_STATIC	DownSample Rate
Edit Properties	IDC_EDIT1	

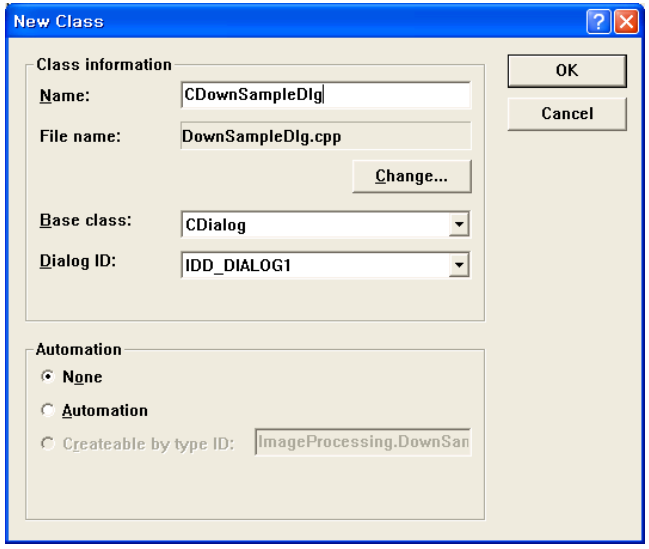


[실습하기 3-4] 영상 축소 프로그램

⑤ **[View]-[ClassWizard] 메뉴 클릭 → [MFC ClassWizard] 대화상자에서 새로 추가된 대화상자를 추가하기 위해 [Adding a Class] 대화상자에서 Create a new class 항목을 체크 → [OK] 버튼 클릭**



→ [New Class] 대화상자에서 다음과 같이 지정한 뒤 [OK] 버튼 클릭(클래스 이름 앞에는 반드시 'C'를 붙여주어야 함)→ 클래스의 이름을 등록하면 파일 이름은 자동으로 생성됨.



Name	CDownSampleDlg

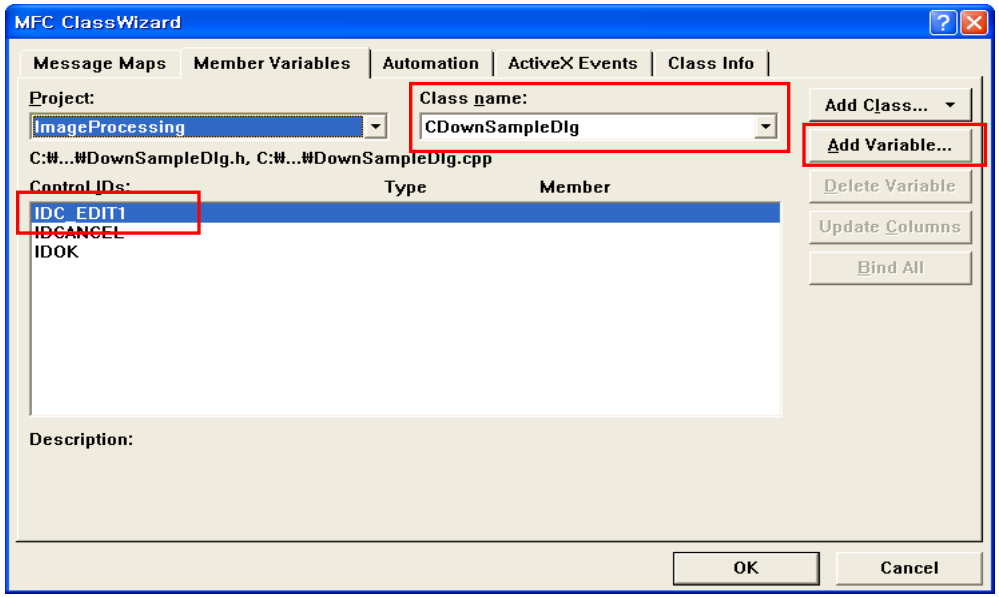
[실습하기 3-4] 영상 축소 프로그램

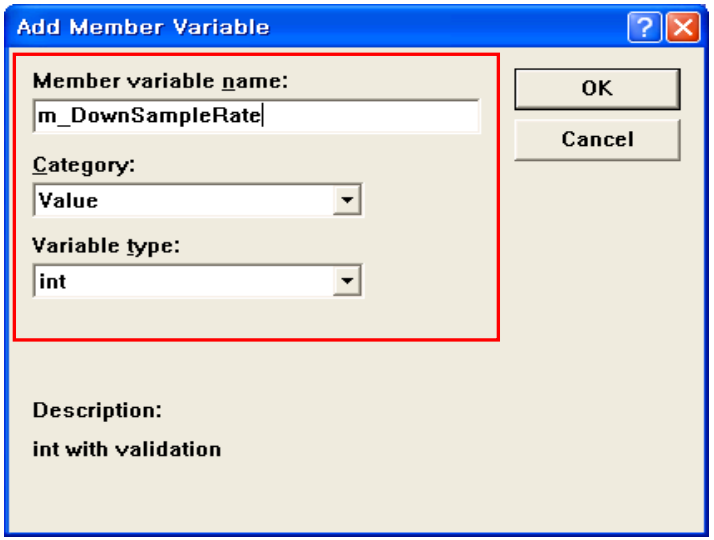
⑥ **[View]-[ClassWizard] 메뉴 클릭 → [MFC ClassWizard] 대화상자의 Object IDs 항목에서 DownSampleDlg 클래스를 선택한 뒤 [Member Variables] 탭 클릭 → 다음과 같이 세부 항목을 지정한 뒤 [Add Variable] 버튼 클릭 → [Add Member Variable] 대화상자에서 다음과 같이 지정하고 [OK] 버튼 클릭**

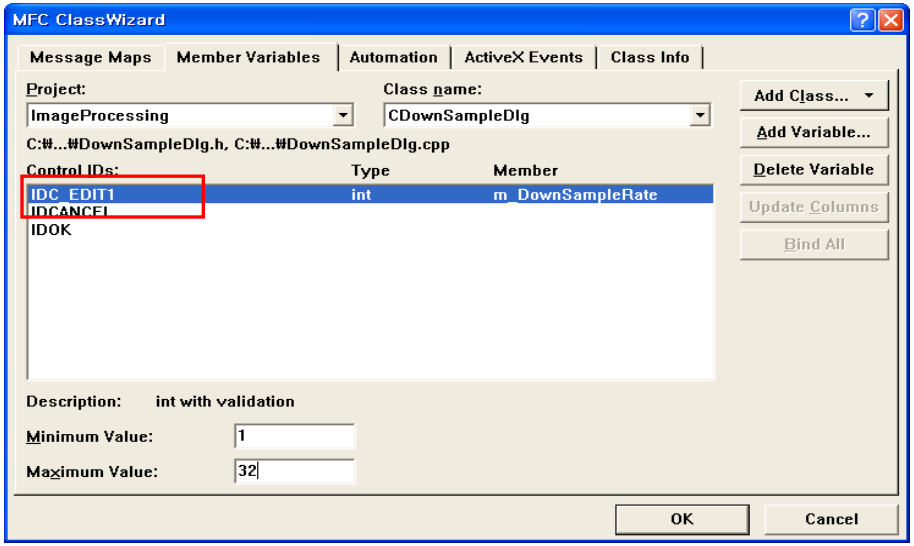
다시 **[MFC ClassWizard]** 대화상자에서 **IDE_EDIT1** 항목을 선택한 뒤 대화상자 아래쪽의 항목을 이용해 변수의 최소값과 최대값을 지정

Control IDs	Member variable name	Category	Variable type	Minimum Value	Maximum Value
IDC_EDIT1	m_DownSampleRate	Value	int	1	32

[실습하기 3-4] 영상 축소 프로그램







[실습하기 3-4] 영상 축소 프로그램

⑦ 메인 프레임에서 [영상처리]-[DownSample] 메뉴를 클릭했을 때 실제로 다운 샘플링이 발생하도록 프로그램을 작성해 주어야 함. 파일을 입·출력 때처럼 Doc 클래스에서 실제로 프로그램을 작성하고, View 클래스가 작성된 프로그램을 호출하여 화면에 출력할 수 있도록 만듦

⑧ 추가된 대화상자를 Doc 클래스에서 사용하려면 다음과 같은 선언 부분이 필요함. ImageProcessingDoc.cpp 파일의 윗부분에 DownSampleDlg.h 코드 추가

```
#include "stdafx.h"
#include "ImageProcessing.h"
#include "ImageProcessingDoc.h"
#include "DownSampleDlg.h" // 대화상자 사용을 위한 헤더 선언
```

[실습하기 3-4] 영상 축소 프로그램

⑨ **Doc 클래스에 OnDownSampling 함수 추가**

Function Type	void
Function Declaration	OnDownSampling
Access	Public

→ **ClassView Workspace에서 CimageProcessingDoc 항목에서 바로가기 메뉴 [Add Member Variable]를 클릭 → 처리 결과 저장 변수, 결과 영상의 가로축 크기, 세로축 크기 지정 변수 등의 사용할 변수 추가**

	Variable Type	Variable Name	Access
입력 영상을 위한 포인터	unsigned char*	m_OutputImage	Public
입력 영상의 가로축 크기	int	m_Re_width	Public
입력 영상의 세로축 크기	int	m_Re_height	Public
입력 영상의 전체 크기	int	m_Re_size	Public

[실습하기 3-4] 영상 축소 프로그램

→ 다음과 같이 **OnDownSampling** 함수 작성

```
void CImageProcessingDoc::OnDownSampling()
{
   int i, j;
   CDownSampleDlg dlg;
   if(dlg.DoModal() == IDOK) // 대화상자의 활성화 여부
   {
      m_Re_height = m_height / dlg.m_DownSampleRate;
      // 축소 영상의 세로 길이를 계산
      m_Re_width = m_width / dlg.m_DownSampleRate;
      // 축소 영상의 가로 길이를 계산
      m_Re_size = m_Re_height * m_Re_width;
      // 축소 영상의 크기를 계산

      m_OutputImage = new unsigned char [m_Re_size];
      // 축소 영상을 위한 메모리 할당

      for(i=0 ; i<m_Re_height ; i++){
       for(j=0 ; j<m_Re_width ; j++){
         m_OutputImage[i*m_Re_width + j]
          = m_InputImage[(i*dlg.m_DownSampleRate*m_width)+dlg.m_DownSampleRate*j];
                // 축소 영상을 생성
         }
     }
   }
}
```

[실습하기 3-4] 영상 축소 프로그램

⑩ 다음과 같이 View 클래스의 OnDownSampling 함수 작성

```
void CImageProcessingView::OnDownSampling()
{
    // TODO: Add your command handler code here
    CImageProcessingDoc* pDoc = GetDocument(); // Doc 클래스 참조
    ASSERT_VALID(pDoc);

    pDoc->OnDownSampling(); // Doc 클래스에 OnDownSampling 함수 호출

    Invalidate(TRUE); // 화면 갱신
}
```

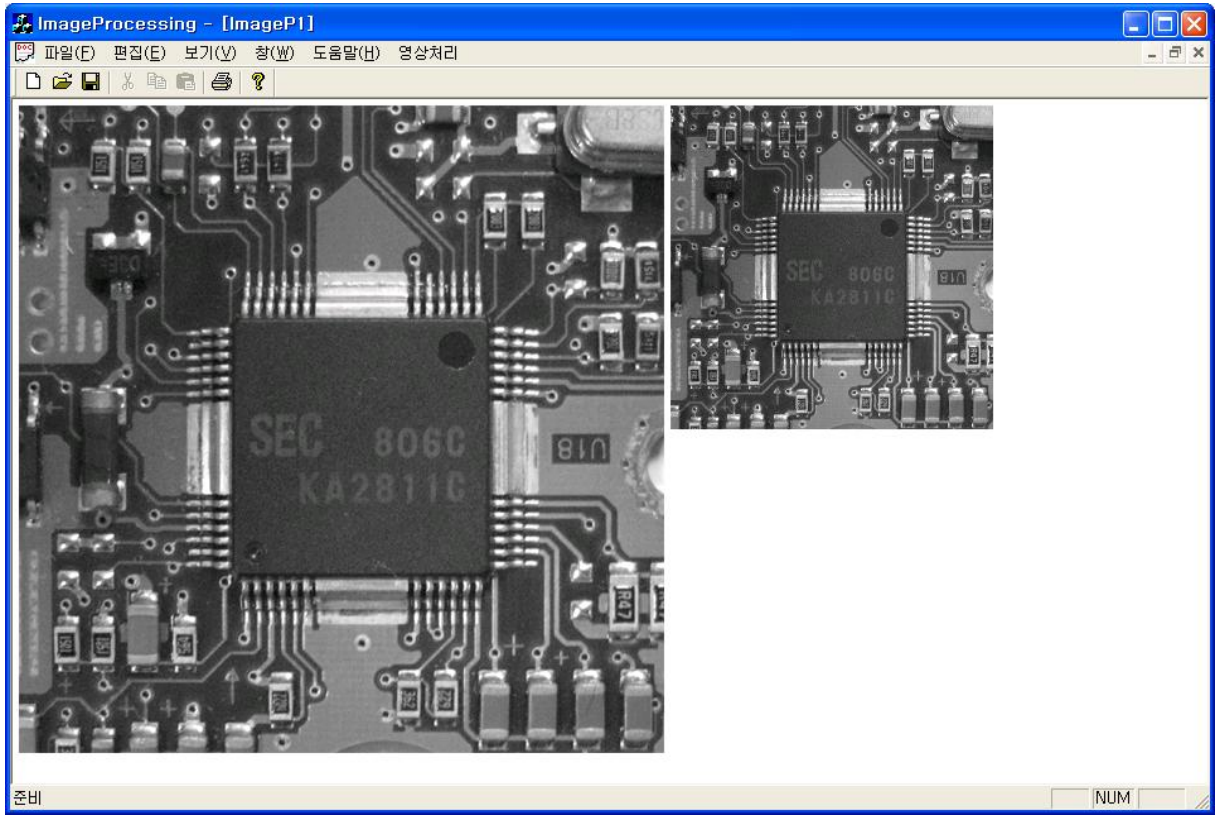
[실습하기 3-4] 영상 축소 프로그램

⑪ 처리된 결과를 화면에 표시하기 위해 **OnDraw** 함수를 다음과 같이 재정의

```
void CImageProcessingView::OnDraw(CDC* pDC)
{
   CImageProcessingDoc* pDoc = GetDocument(); // Doc 클래스 참조
   ASSERT_VALID(pDoc);
   // TODO: add draw code for native data here

   int i, j;
   unsigned char R, G, B;
   // 입력 영상 출력
   for(i = 0 ; i<pDoc->m_height ; i++){
      for(j = 0 ; j<pDoc->m_width ; j++){
          R = pDoc->m_InputImage[i*pDoc->m_width+j];
          G = B = R;
          pDC->SetPixel(j+5, i+5, RGB(R, G, B));
      }
   }

   // 축소된 영상 출력
   for(i = 0 ; i<pDoc->m_Re_height ; i++){
       for(j = 0 ; j<pDoc->m_Re_width ; j++){
           R = pDoc->m_OutputImage[i*pDoc->m_Re_width+j];
           G = B = R;
           pDC->SetPixel(j+pDoc->m_width+10, i+5, RGB(R, G, B));
       }
   }
}
```

[실습하기 3-4] 영상 축소 프로그램

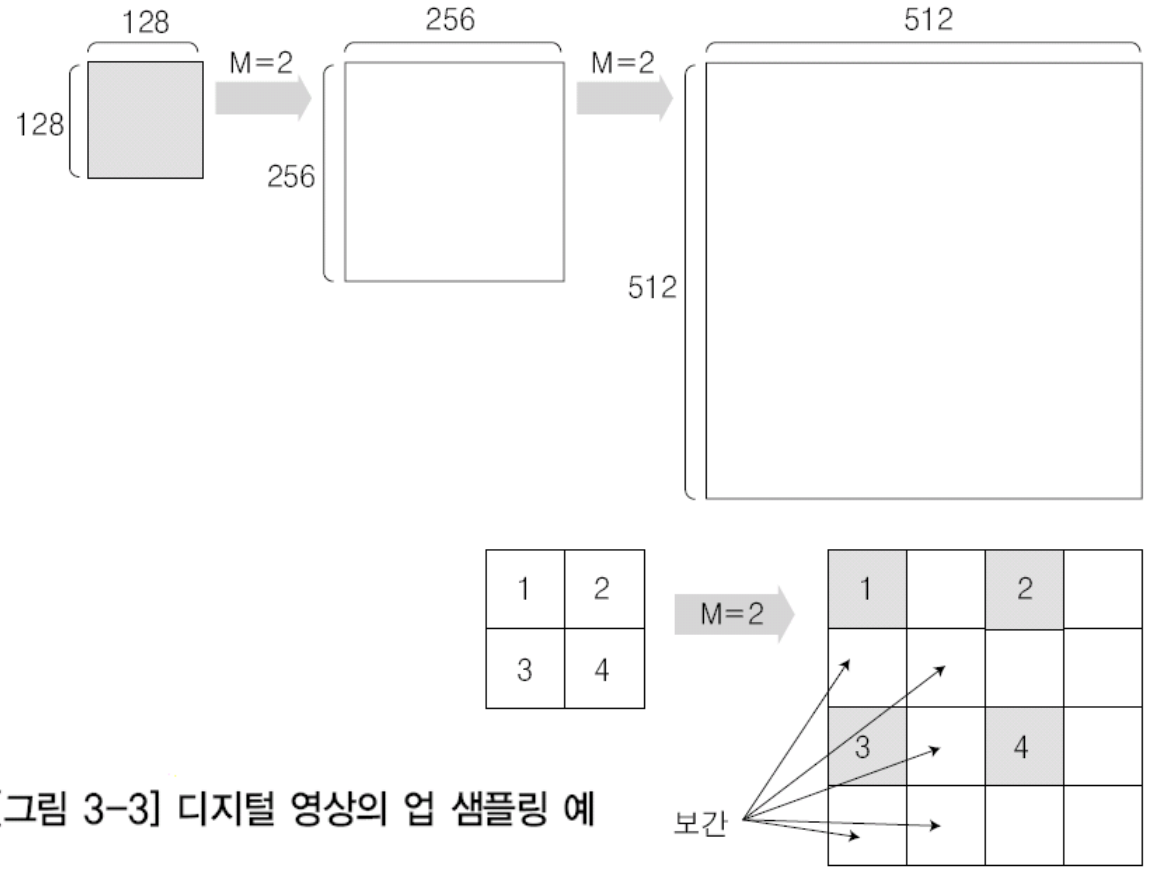
⑫ 실행 결과 영상. 이후 작성되는 프로그램에서는 결과 영상을 **m_OutputImage** 배열에 입력하고, 결과 영상의 가로축 크기와 세로축 크기를 **m_Re_width, m_Re_height**에 값을 지정하면 **OnDraw** 함수에서는 처리된 결과를 화면 오른쪽에 표시하게 됨



Sampling rate 2로 다운 샘플링된 결과 영상

업 샘플링(Up Sampling)

- 영상을 확대할 때는 먼저 일정한 배열 간격으로 재배열해야 하는 것
- 단순 업 샘플링을 사용하여 영상을 확대하면 영상의 품질이 현저히 떨어짐.
- 영상을 확대해도 선명한 품질을 얻고 싶다면 업 샘플링으로 얻은 데이터와 원본 영상의 데이터를 이용하여 보간(Interpolation)을 해야 함

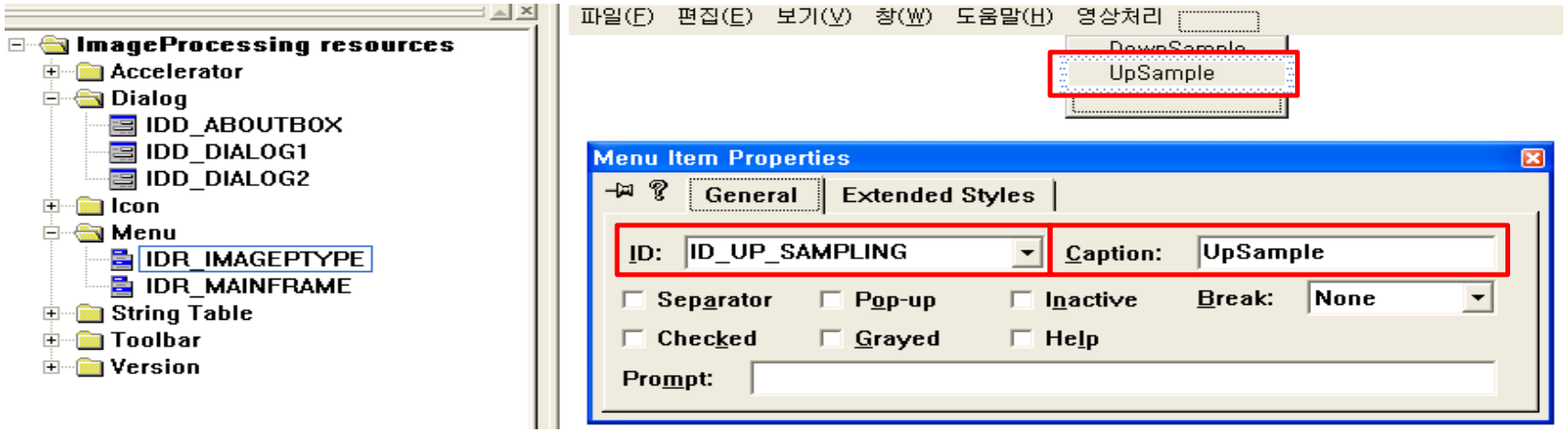


[그림 3-3] 디지털 영상의 업 샘플링 예

3장에서는 단순 재배열로 확대하는 것만 다루고 보간을 사용한 영상의 확대는 8장의 기하학 처리에서 학습함.

[실습하기 3-5] 영상 확대 프로그램

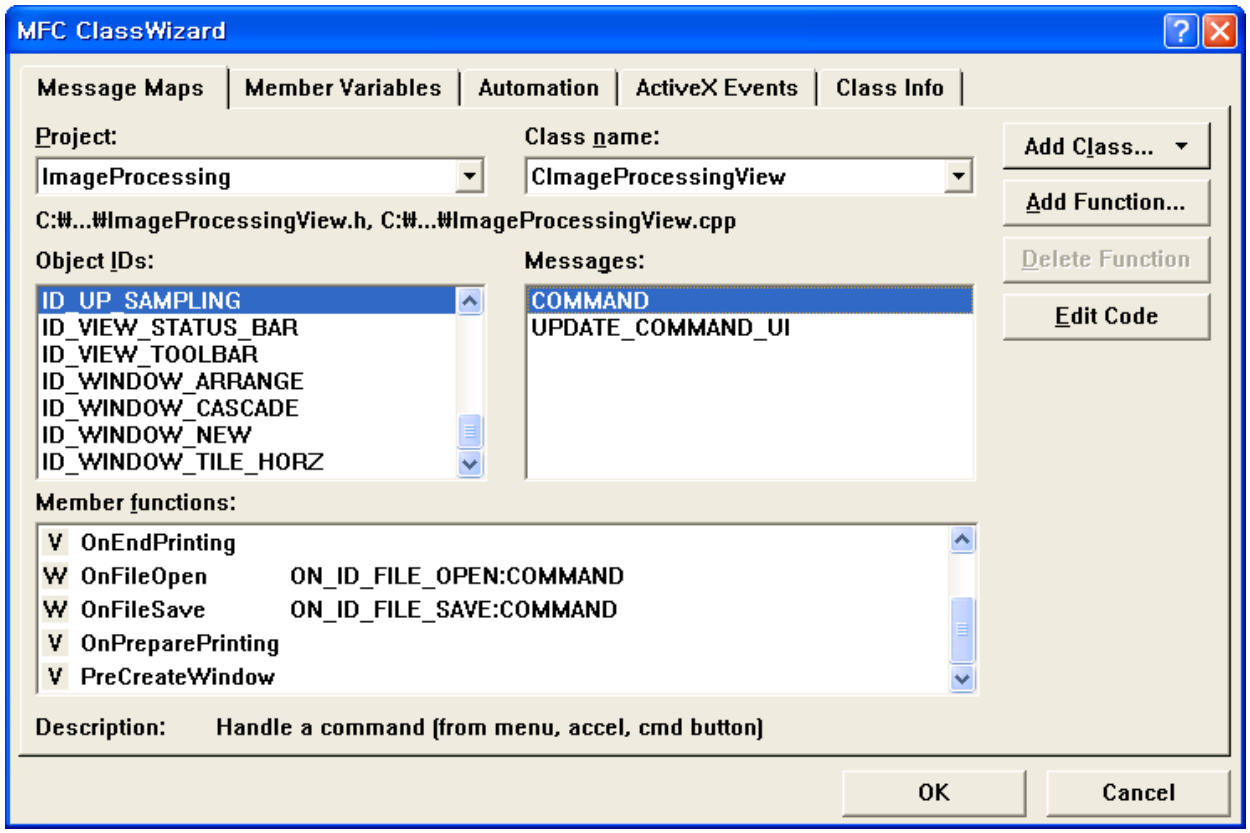
① **Visual C++ 프로그램을 실행한 뒤 Workspace 창에서 [ResourceView] 탭 클릭 → [Menu]-[IDR_IMAGEPTYPE] 폴더 더블클릭 → [영상처리] 하위에 [UpSample] 메뉴가 새로 추가 됨. →추가된 메뉴에서 [Properties] 클릭 → [Menu Item Properties] 대화상자에서 다음과 같이 설정**



ID	ID_UP_SAMPLING
Caption	UpSample

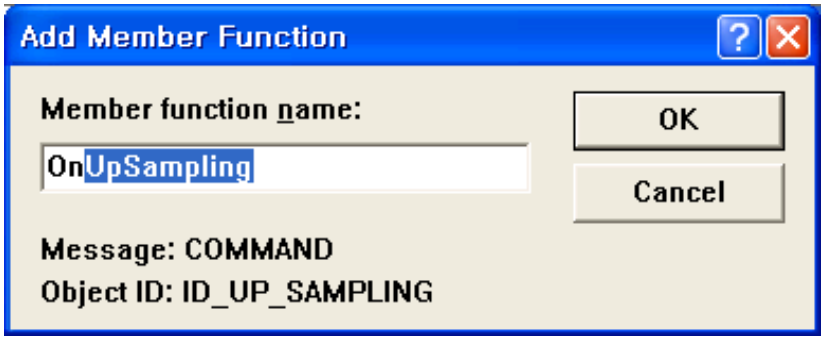
[실습하기 3-5] 영상 확대 프로그램

→ [View]-[ClassWizard] 메뉴 클릭 → [MFC ClassWizard] 대화상자에서 업 샘플링 함수를 다음과 같이 생성함.



[실습하기 3-5] 영상 확대 프로그램

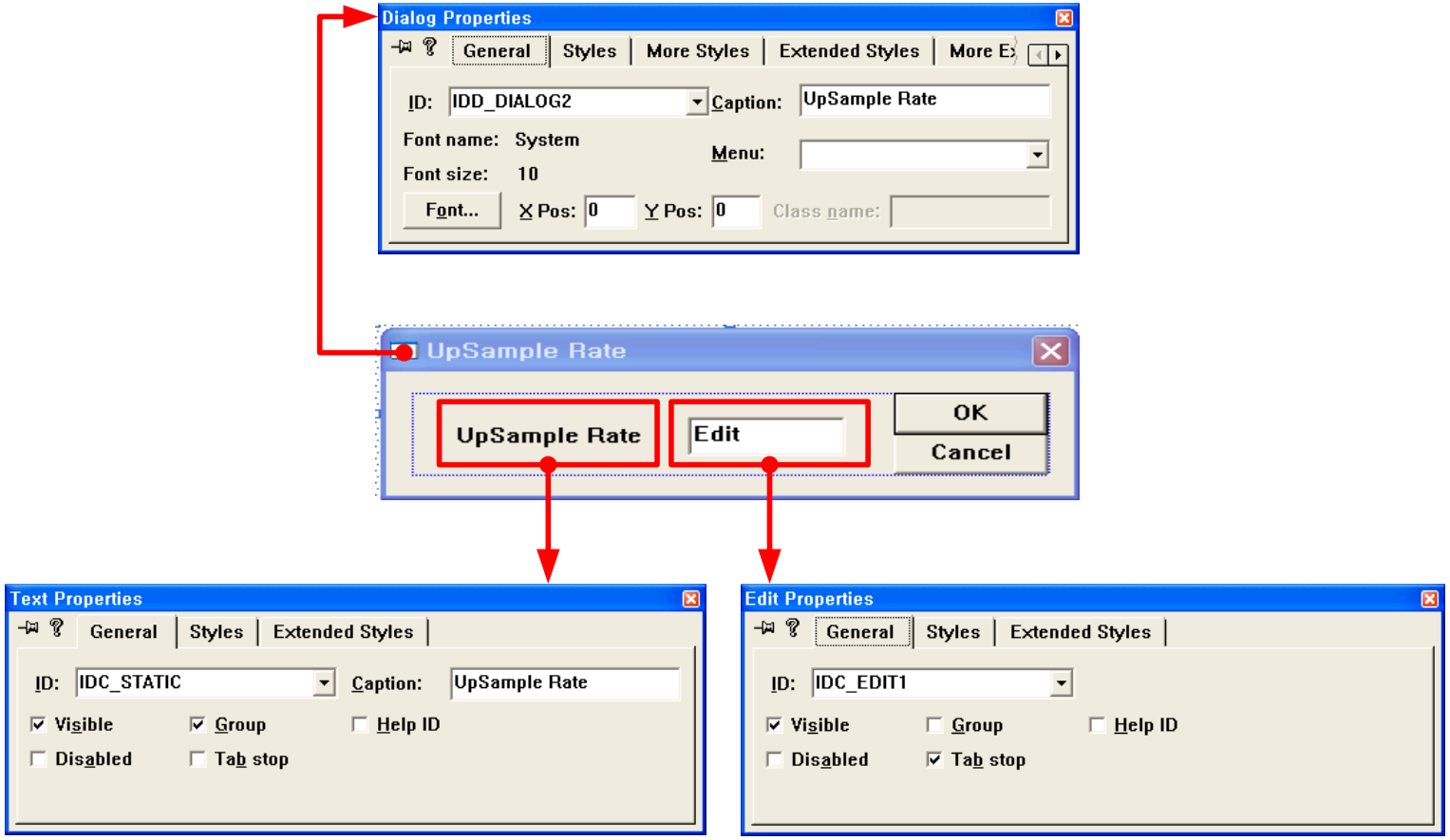
② **[Add Function]** 버튼 클릭 → **View Class**에 다운 샘플링을 담당하는 함수를 생성



→ 업 샘플링도 다운 샘플링처럼 다양한 샘플링 비율로 샘플링이 가능하며, 보간법의 사용 유무와 종류를 결정해 주어야 함. 따라서 업 샘플은 새로운 대화상자를 추가하여 업 샘플의 추가 정보를 대화상자에서 입력받도록 함

[실습하기 3-5] 영상 확대 프로그램

③ [View]-[ClassWizard] 메뉴를 클릭 → [MFC ClassWizard]대화상자에서 업 샘플링 함수를 다음과 같이 생성

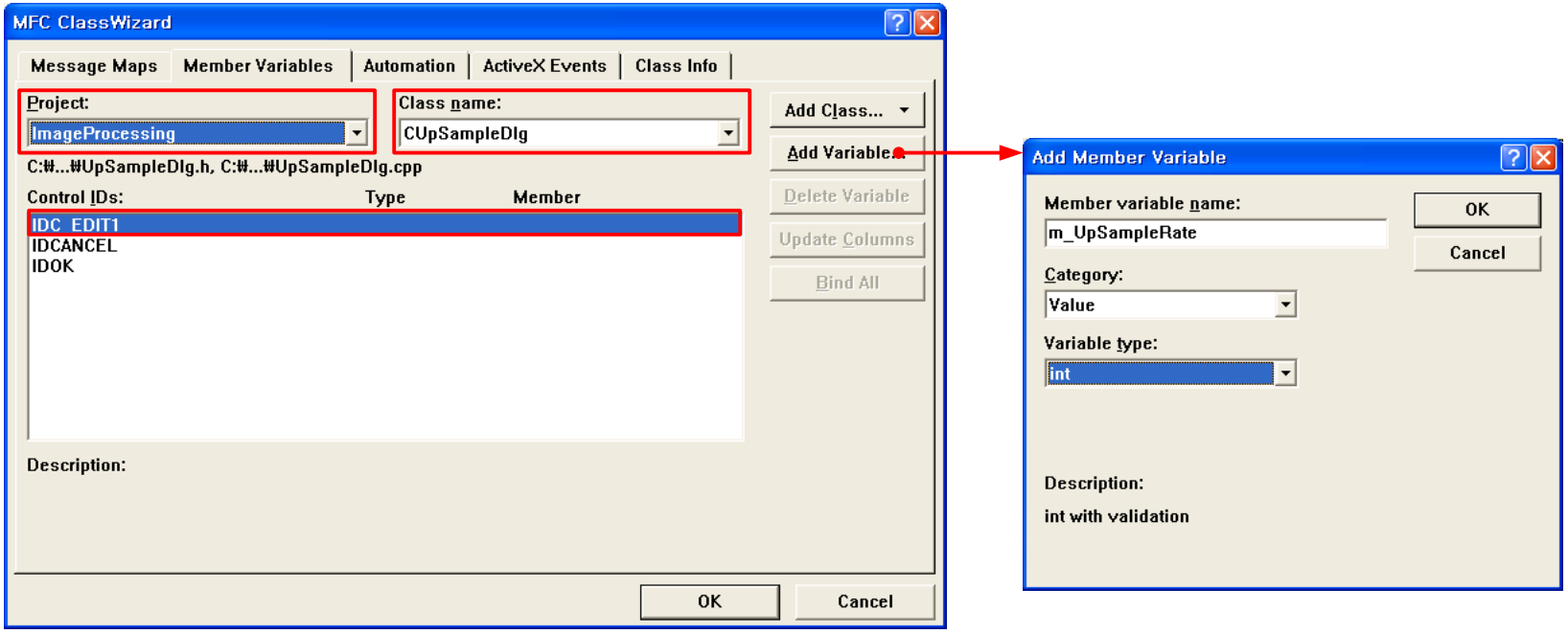


[실습하기 3-5] 영상 확대 프로그램

④ **[MFC ClassWizard] 대화상자에서 추가된 대화상자를 새로운 이름의 클래스로 등록함**

Name	CUpSampleDlg

→ **CUpSampleDlg의 IDC_EDIT1에 다음과 같이 변수 등록**



Control IDs	Member variable name	Category	Variable type
IDC_EDIT1	m_UpSampleRate	Value	int

[실습하기 3-5] 영상 확대 프로그램

⑤ **UpSampleDlg 대화상자를 Doc 클래스에서 사용하기 위해 ImageProcessingDoc.cpp 파일의 윗부분에 UpSampleDlg.h 코드를 포함시킴**

```
#include "stdafx.h"
#include "ImageProcessing.h"
#include "ImageProcessingDoc.h"
#include "DownSampleDlg.h"
#include "UpSampleDlg.h"
```

→ **Doc 클래스에 OnUpSampling 함수를 추가함**

Function Type	void
Function Declaration	OnUpSampling
Access	Public

[실습하기 3-5] 영상 확대 프로그램

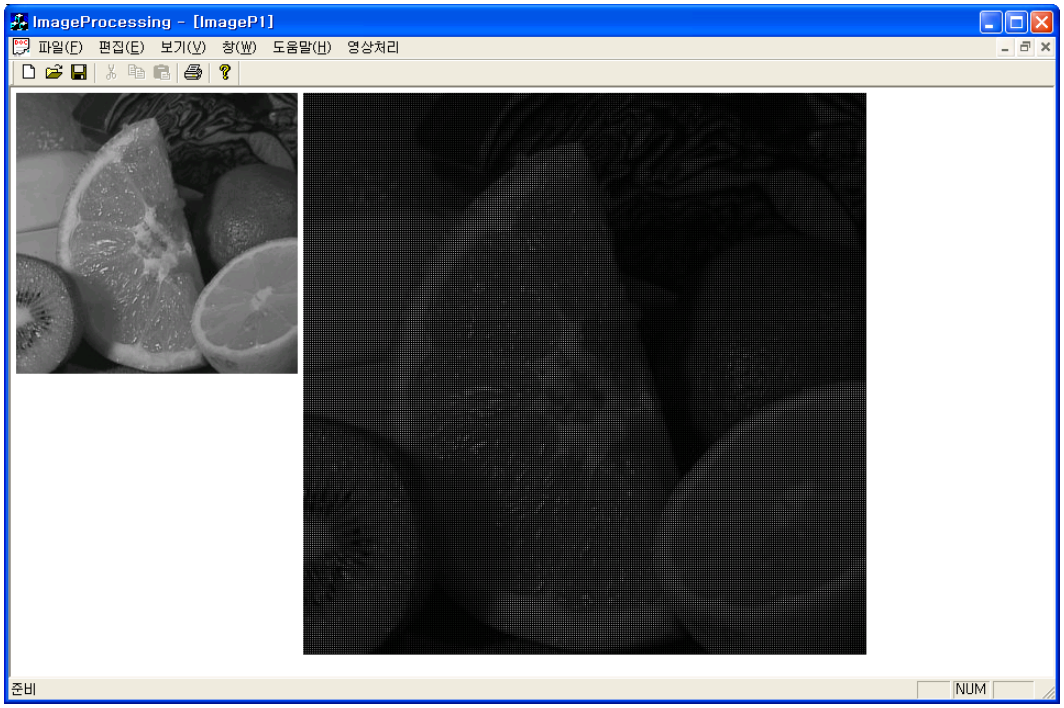
```
void CImageProcessingDoc::OnUpSampling()
{
    int i, j;

    CUpSampleDlg dlg;
    if(dlg.DoModal() == IDOK){ // DoModal 대화상자의 활성화 여부
        m_Re_height = m_height * dlg.m_UpSampleRate;
        // 확대 영상의 세로 길이 계산
        m_Re_width = m_width * dlg.m_UpSampleRate;
        // 확대 영상의 가로 길이 계산
        m_Re_size = m_Re_height * m_Re_width;
        // 확대 영상의 크기 계산
        m_OutputImage = new unsigned char[m_Re_size];
        // 확대 영상을 위한 메모리 할당

        for(i=0 ; i<m_Re_size ; i++)
            m_OutputImage[i] = 0; // 초기화

        for(i=0 ; i<m_height ; i++){
            for(j=0 ; j<m_width ; j++){
               m_OutputImage[i*dlg.m_UpSampleRate*m_Re_width +
               dlg.m_UpSampleRate*j]= m_InputImage[i*m_width + j];
            } // 재배치하여 영상 확대
        }
    }
}
```

[실습하기 3-5] 영상 확대 프로그램

⑥ View 클래스의 OnUpSampling 함수를 다음과 같이 작성

```
void CImageProcessingView::OnUpSampling()
{
    // TODO: Add your command handler code here
    CImageProcessingDoc* pDoc = GetDocument(); // Doc 클래스 참조
    ASSERT_VALID(pDoc);

    pDoc->OnUpSampling(); // Doc 클래스에 OnUpSampling 함수 호출

    Invalidate(TRUE); // 화면 갱신
}
```

⑦ 결과 영상



Sampling rate 2로 업 샘플링된 결과 영상

양자화

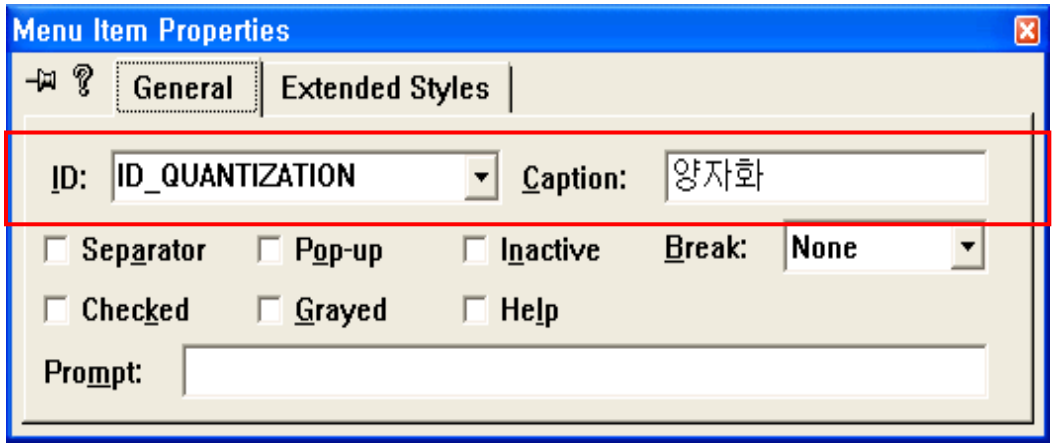
- 표본화된 화소의 밝기나 색상을 정해진 몇 단계의 값으로 근사화하는 과정
- 화소의 밝기나 색상이 숫자로 표현되어 화소는 양자화된 표본 값을 갖게 됨.
- 밝기나 색상이 몇 단계로 표현되느냐는 양자화 비트(Quantization Bits)로 결정됨.

[표 3-2] 양지화 비트 값에 따른 색상 표현

양자화 비트	표현 가능한 색상
1비트	2
2비트	4
3비트	8
4비트	16
5비트	32
6비트	64
7비트	128
8비트	256

[실습하기 3-6] 양자화 프로그램

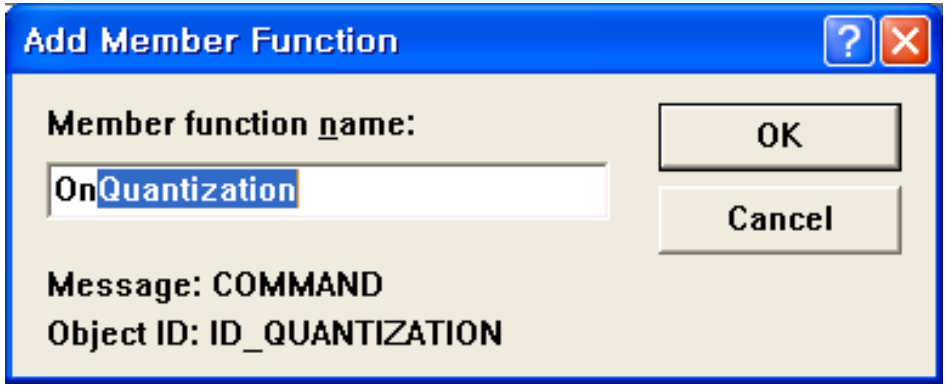
① **Visual C++ 프로그램의 Workspace 창에서 [ResourceView] 탭 클릭 → [Menu]-[IDR_IMAGEPTYPE] 메뉴 더블클릭 → 추가된 메뉴 프레임의 속성을 다음과 같이 지정**



ID	ID_QUANTIZATION
Caption	양자화

[실습하기 3-6] 양자화 프로그램

② **[MFC ClassWizard] 대화상자를 이용해 View 클래스에 OnQuantization 함수 생성**

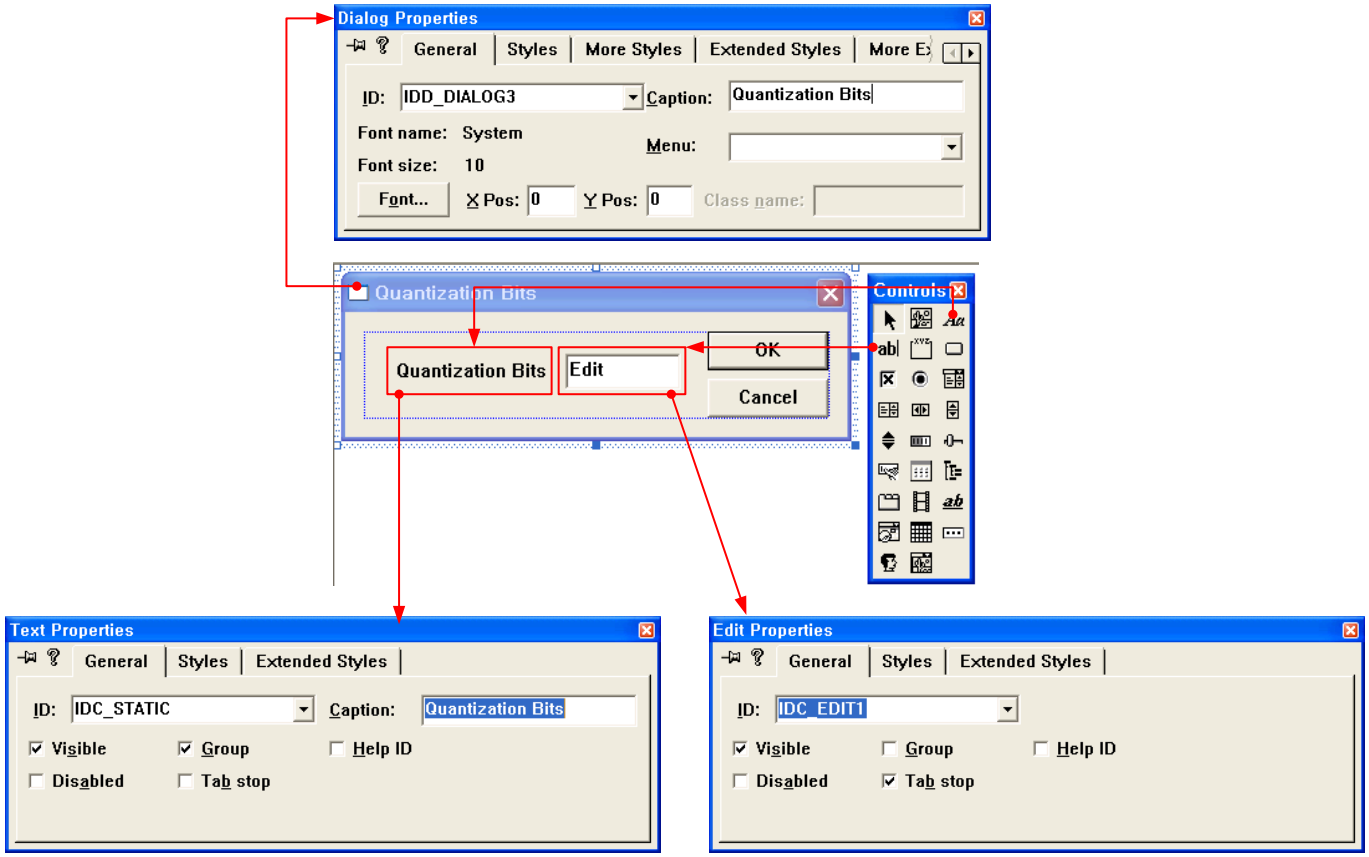


위치	CImageProcessingView
Member Function name	OnQuantization

[실습하기 3-6] 양자화 프로그램

③ **ResourceView 창에서 [ImageProcessing resources]-[Dialog] 폴더 클릭→ 바로가기 메뉴 [Insert Dialog] 클릭→ 대화상자의 속성을 다음과 같이 지정**

	ID	Caption
Dialog Properties	IDD_DIALOG3	Quantization Bits
Text Properties	IDC_STATIC	Quantization Bits
Edit Properties	IDC_EDIT1	

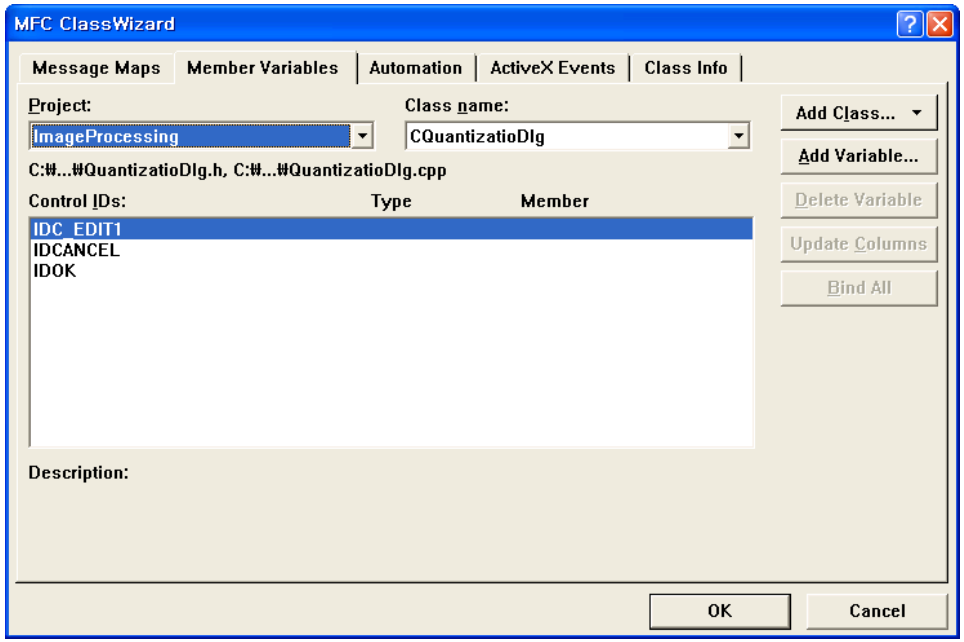


[실습하기 3-6] 양자화 프로그램

④ **[MFC ClassWizard]** 대화상자에서 양자화 비트 대화상자의 새로운 클래스를 등록

Name	CQuantizationDlg

→ **CQuantizationDlg** 클래스를 선택하고 **[Member Variables]** 탭에서 **IDC_EDIT1**에 다음과 같이 변수 할당

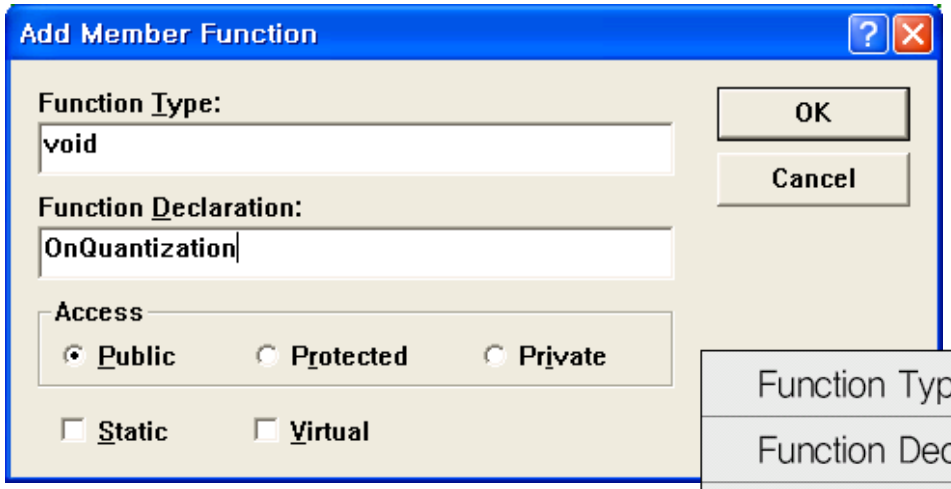


Control IDs	Member variable name	Category	Variable type	Minimum Value	Maximum Value
IDC_EDIT1	m_QuantBit	Value	int	1	8

[실습하기 3-6] 양자화 프로그램

- Doc 클래스에서 양자화 비트 대화상자를 사용하기 위해 양자화 비트 대화상자의 헤더 파일 선언 → 양자화 함수를 생성하고 양자화 프로그램 작성. 양자화 레벨의 수는 2bit로 계산되므로, NM을 계산하는 pow 함수를 사용함(math.h에 포함되어 있음)

```
#include "stdafx.h"
#include "ImageProcessing.h"
#include "ImageProcessingDoc.h"
#include "DownSampleDlg.h"
#include "UpSampleDlg.h"
#include "QuantizationDlg.h" // 대화상자 사용을 위한 헤더 선언
#include "math.h" // 수학 함수 사용을 위한 헤더 선언
```



Function Type	void
Function Declaration	OnQuantization
Access	Public

[실습하기 3-6] 양자화 프로그램

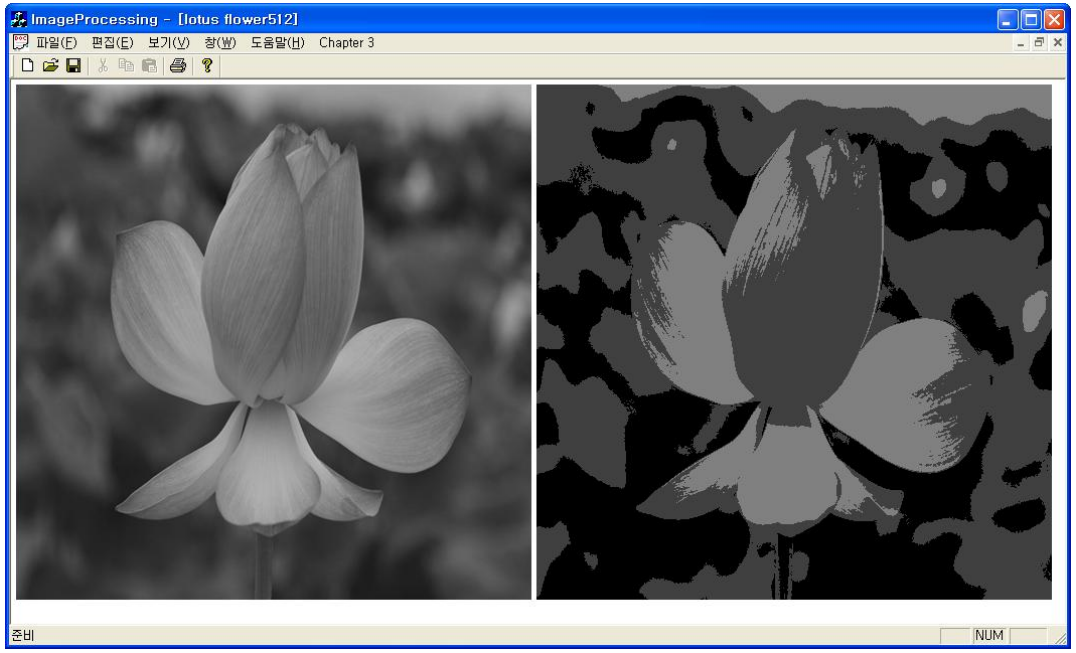
```
void CImageProcessingDoc::OnQuantization()
{
   CQuantizationDlg dlg;
   if(dlg.DoModal() == IDOK)
   // 양자화 비트 수를 결정하는 대화상자의 활성화 여부
   {
      int i, j, value, LEVEL;
      double HIGH, *TEMP;

      m_Re_height = m_height;
      m_Re_width = m_width;
      m_Re_size = m_Re_height * m_Re_width;

      m_OutputImage = new unsigned char[m_Re_size];
      // 양자화 처리된 영상을 출력하기 위한 메모리 할당

      TEMP = new double [m_size];
      // 입력 영상 크기(m_size)와 동일한 메모리 할당

      LEVEL=256; // 입력 영상의 양자화 단계(28=256)
      HIGH=256.;

      value = (int)pow(2, dlg.m_QuantBit);
      // 양자화 단계 결정(예 : 24=16)

      for(i=0 ; i<m_size ; i++){
         for(j=0 ; j<value ; j++){
            if(m_InputImage[i] >=(LEVEL/value)*j &&
               m_InputImage[i]<(LEVEL/value)*(j+1)){
                  TEMP[i]=(double)(HIGH/value)*j; // 양자화 수행
            }
         }
      }
      for(i=0 ; i<m_size ; i++){
         m_OutputImage[i] = (unsigned char)TEMP[i];
         // 결과 영상 생성
      }
   }
}
```

[실습하기 3-6] 양자화 프로그램

⑥ View 클래스에서 Doc 클래스의 OnQuantization 함수를 호출하고 화면을 갱신할 수 있도록 프로그램 작성

```
void CImageProcessingView::OnQuantization()
{
    // TODO: Add your command handler code here
    CImageProcessingDoc* pDoc = GetDocument(); // Doc 클래스 참조
    ASSERT_VALID(pDoc);

    pDoc->OnQuantization(); // Doc 클래스에 OnQuantization 함수 호출

    Invalidate(TRUE); // 화면 갱신
}
```

⑦ 결과 영상



4-level(2bit) 양자화된 결과 영상

Thank you