패턴 인식

강미선

영상처리란?

- 스캐너, 디지털 카메라등의 장치를 통하여 획득되었거나 또는 컴퓨터로 생성된 **영상을 원하는 목적에 맞게 조작**하는 것

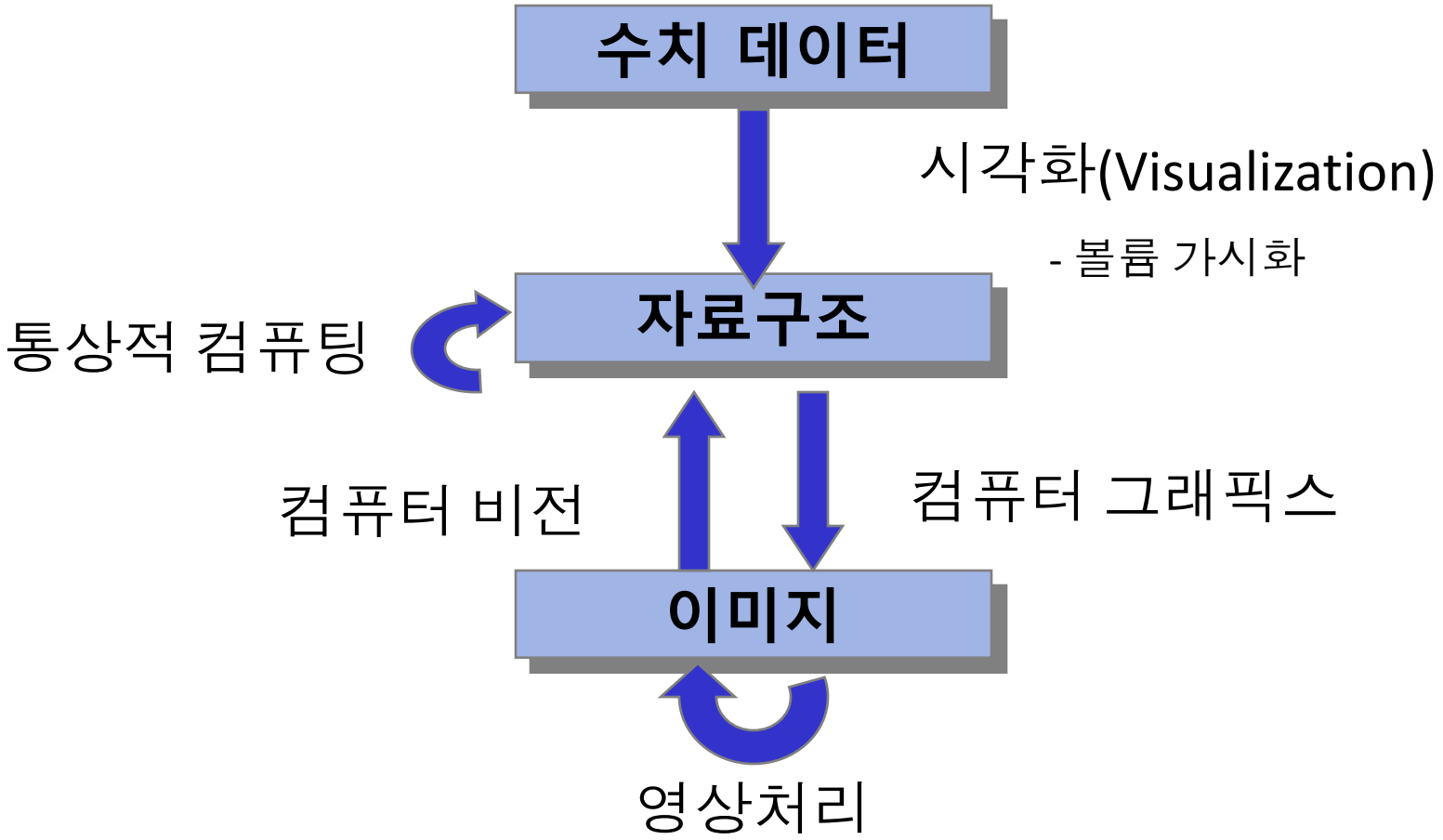
- 영상 처리 예
  - 영상의 화질 개선
  - 기하학적 변환
  - 영상의 특징 부각
  - 영상 압축/복원 등

영상 처리 필요성

- 수동 처리
  - ❖ 주관적
  - ❖ 소요시간↑

- 컴퓨팅 기술을 이용한 자동 처리
  - ❖ 정량적 측정
  - ❖ 빅데이터 분석

다른 분야와의 관계


수치 데이터
시각화(Visualization)
- 볼륨 가시화
자료구조
통상적 컴퓨팅
컴퓨터 비전
컴퓨터 그래픽스
이미지
영상처리

영상처리 응용

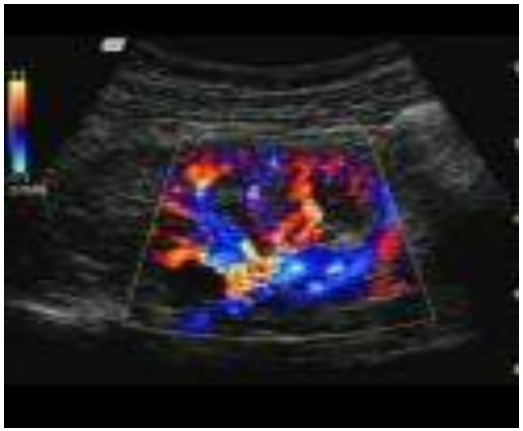
- 방송과 영화
  - 특수 효과 : 모핑, 워핑
- 의료
  - X-선, 초음파, CT, MRI 영상의 처리
    - 정확한 진단을 위해 영상을 확대, 변형, 조작
- 산업 현장
  - 결함 상품의 자동 검사
  - 산업용 로봇의 눈
- 보안 및 감시
  - 지문 인식, 얼굴 인식, 침입자 감지, 자동차 번호인식, 홍채인식
- 출판 및 문서 제작
  - 사진에 여러 가지 효과를 주기 위한 처리

영상처리 응용

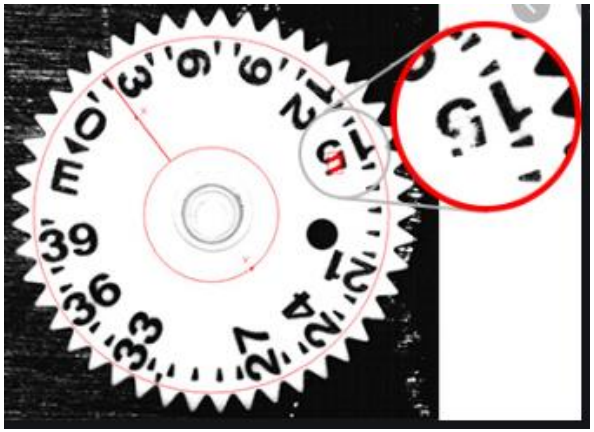
- 증강 현실

– 얼굴 인식

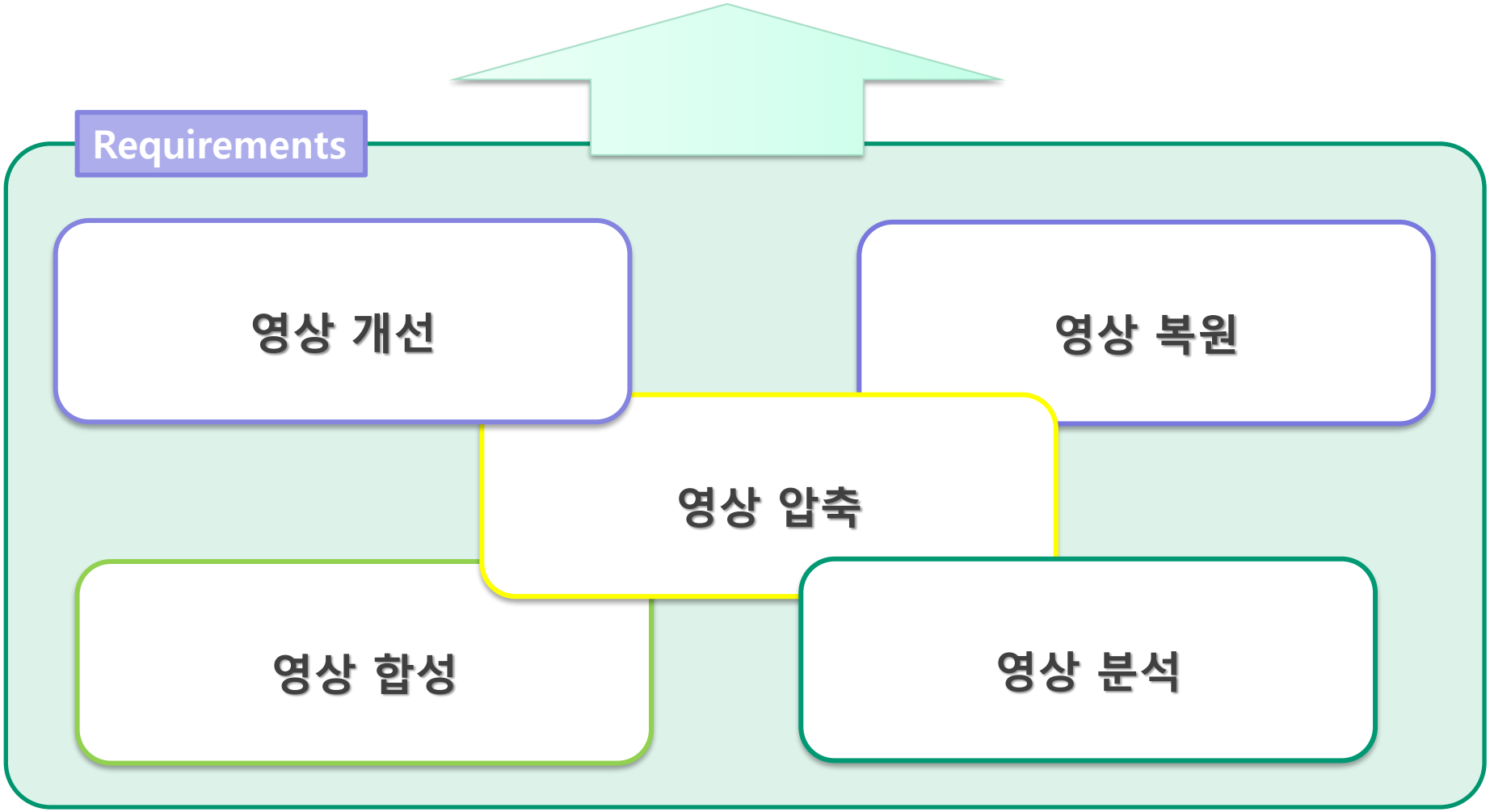
– 의료 영상



– 공장 자동화



영상 처리



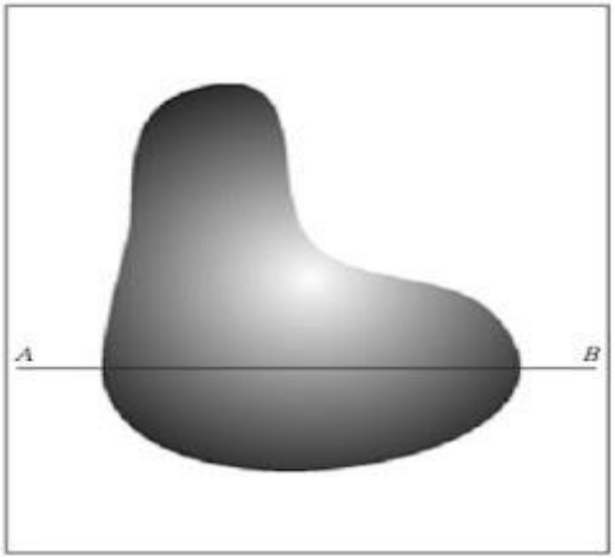
디지털 영상 생성

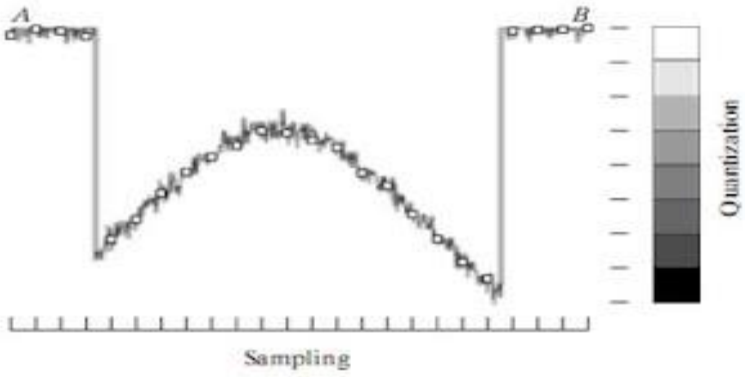
- 디지타이저
  - 아날로그 영상인 자연 영상을 컴퓨터에서 처리하기 위해서는 디지털 영상으로 변환

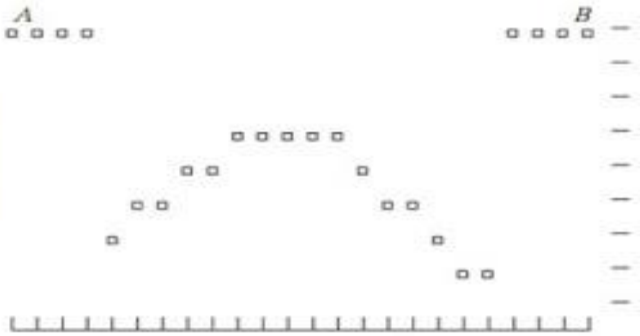
- 디지타이저의 두 가지 기능
  - 샘플링(sampling)
    - 하나의 영상을 표현하기 위하여 동등한 공간 크기로 데이터를 획득
  - 양자화(quantization)
    - 샘플링된 데이터에 수치값을 할당

- 디지타이저의 예
  - 스캐너, 디지털 카메라

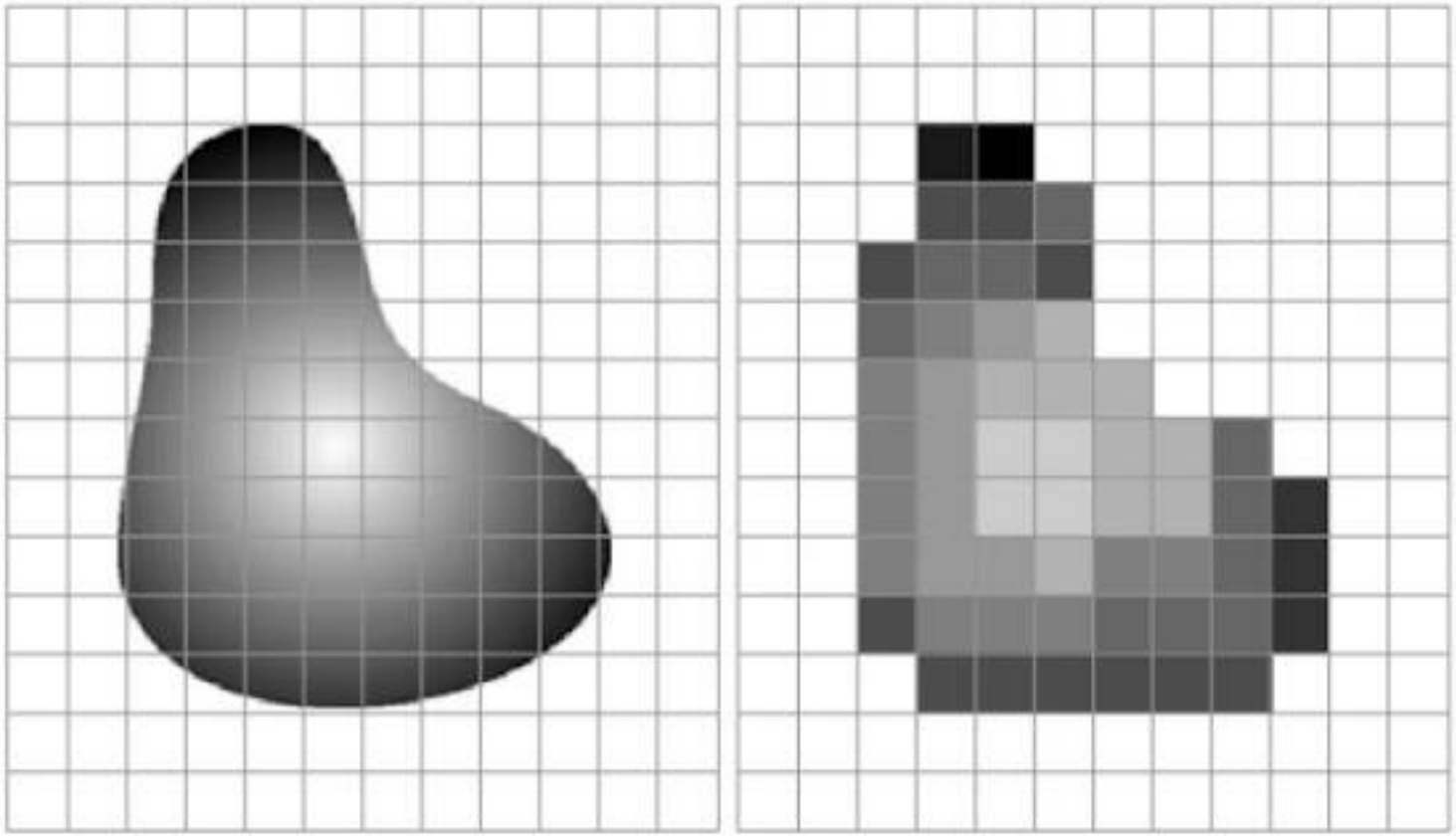
샘플링







샘플링



a b

샘플링크기에 따른 효과

다양한 양자화 수준 예

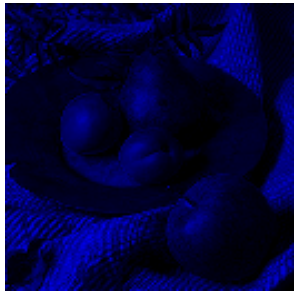
컬러 모델

- 응용분야에 따라 각기 다른 컬러모델을 사용
  - RGB
    - 컬러 모니터와 컴퓨터 그래픽스 시스템
  - CMY
    - 컬러로 된 그림을 출판하는 시스템
  - HSI
    - 색상, 채도, 명도를 다루어야 하는 시스템

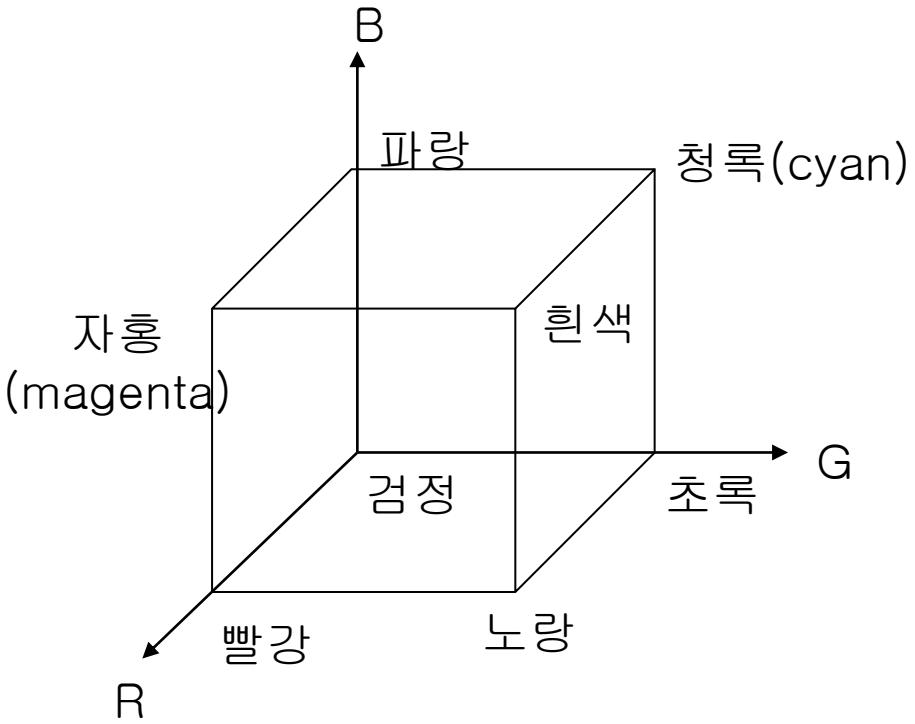
RGB 컬러 모델

R

G



B



CMY 컬러 모델

- 3원색 : 청록(Cyan), 자홍(Magenta), 노랑(Yellow)

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

보색 (complement)

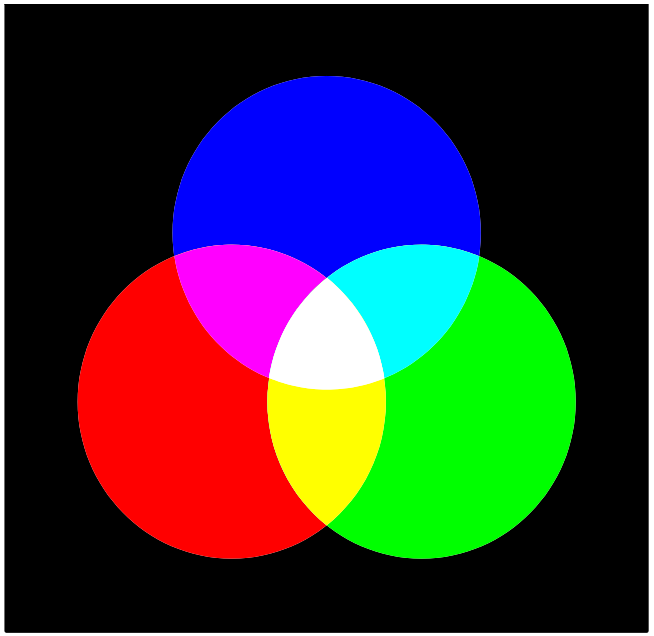
C

M

Y

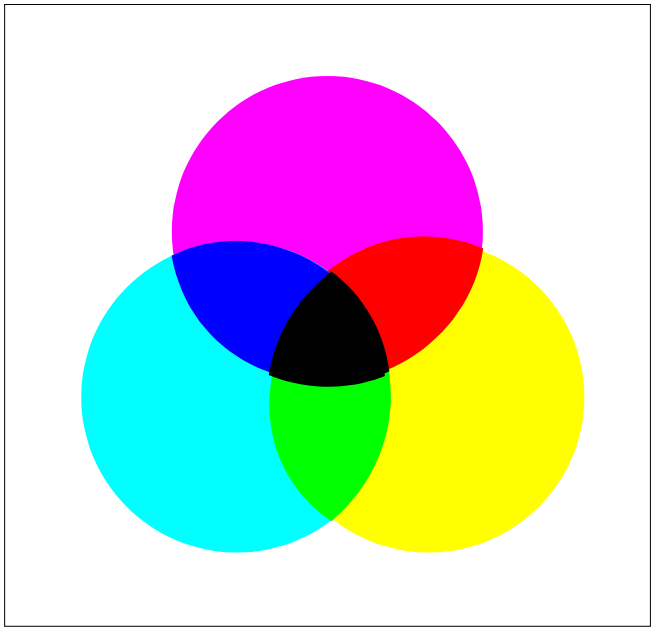
가산과 감산에 의한 컬러

가산 색 모형



RGB 컬러 모형

감산 색 모형



CMY 컬러 모형

CMYK 컬러 모델

- **CMYK = CMY + Black**
- **검정**
  - **다른 세 가지 컬러들의 조합에 의해 생성하는 것보다**
  - **순수 검정색이 더 좋기 때문에 프린팅 처리를 할 때 사용**

$$K = \min(C, M, Y)$$

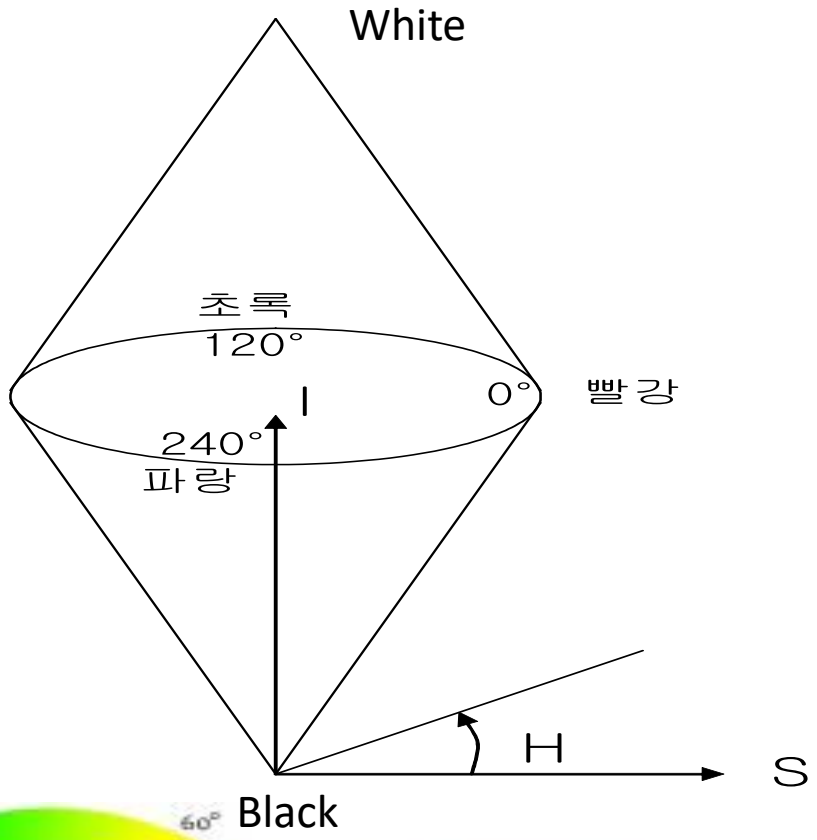
$$C = (C - K)/(1 - K)$$

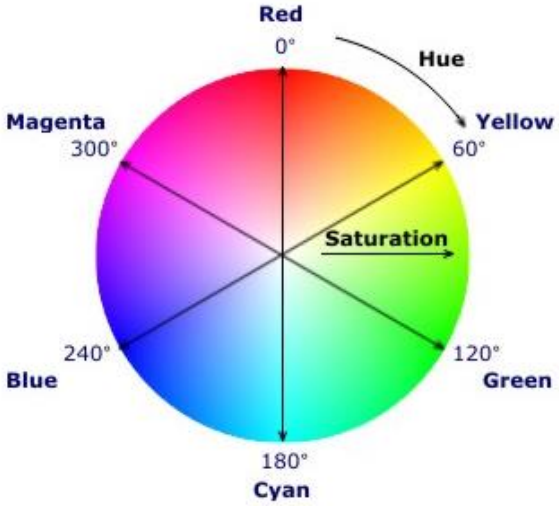
$$M = (M - K)/(1 - K)$$

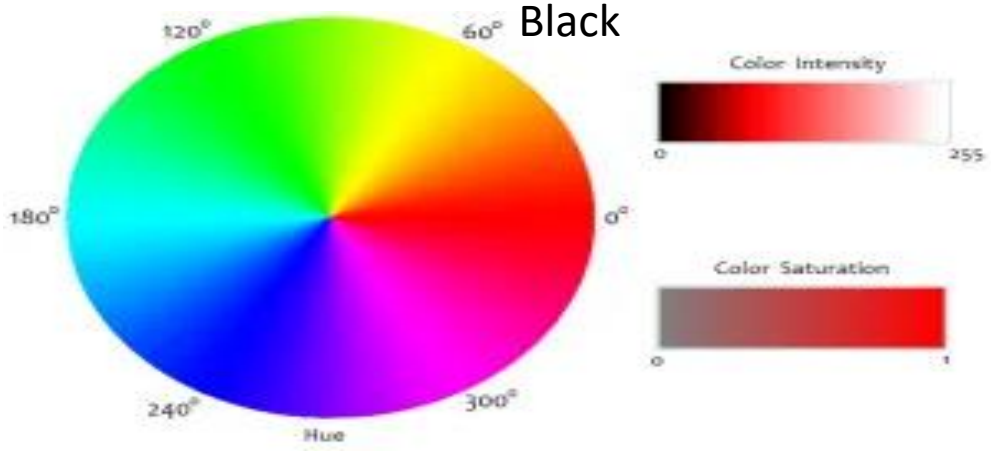
$$Y = (Y - K)/(1 - K)$$

HSI 컬러 모델

- 색상(Hue)
- 채도(Saturation)
  - 흰색으로 희석되지 않은 색깔의 정도를 나타냄
  - 색깔의 순수도를 나타냄
- 명도(Intensity)
- 많은 영상처리 어플리케이션이 HSI 모델 사용







Visual C++ 프로그램 작성

- 윈도우 프로그램 작성
- 메뉴 생성
- 부메뉴 생성 및 연결함수 작성
- 영상 출력 프로그램 작성
- 기본 클래스

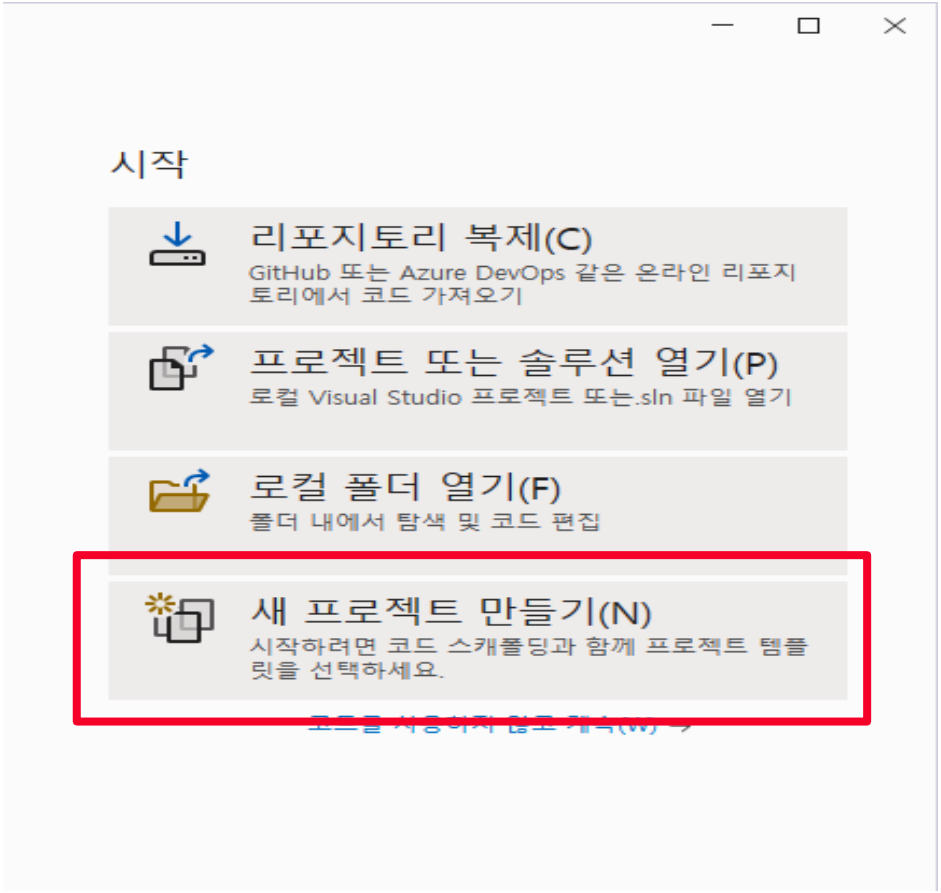
윈도우 프로그램 작성

- Visual C++에서는 윈도우 프로그램 작성을 도와주는 **마법사**를 제공
  - 몇 가지 선택 사항을 입력하면 자동으로 기본적인 윈도우 프로그램이 생성됨
  - 생성된 윈도우 프로그램을 **확장**하여 개별적인 응용 프로그램을 작성할 수 있도록 해주는 기능이 제공됨

윈도우 프로그램 작성 단계

1. Visual Studio를 실행한 다음 [새 프로젝트 만들기] 항목 선택

MFC 설치하기



윈도우 프로그램 작성 단계

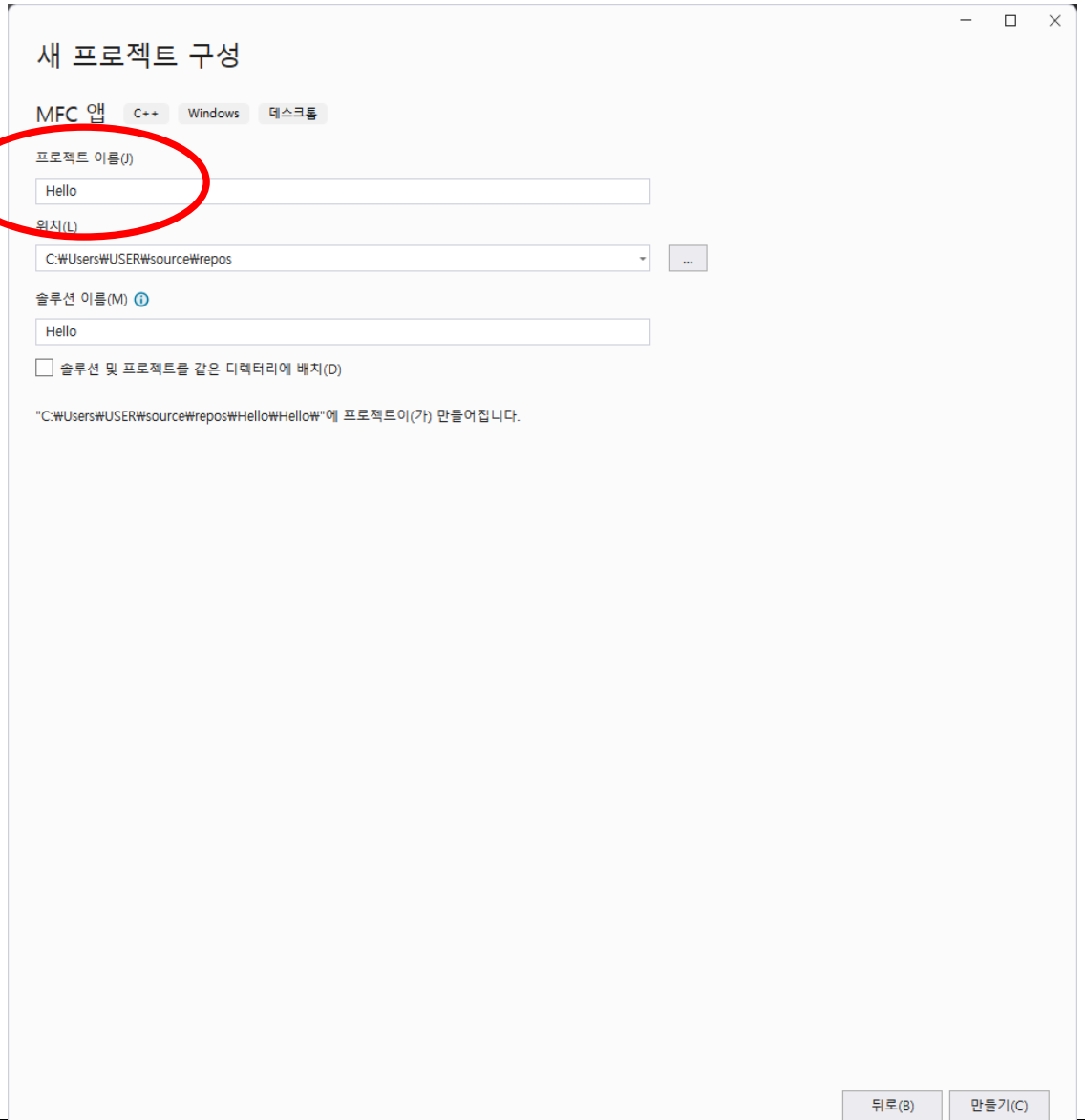
2. mfc 검색 시 없으면 “추가 도구 및 기능 설치 클릭 후 C++ mfc 설치

윈도우 프로그램 작성 단계

2. MFC 설치 후 [MFC앱] 항목 선택

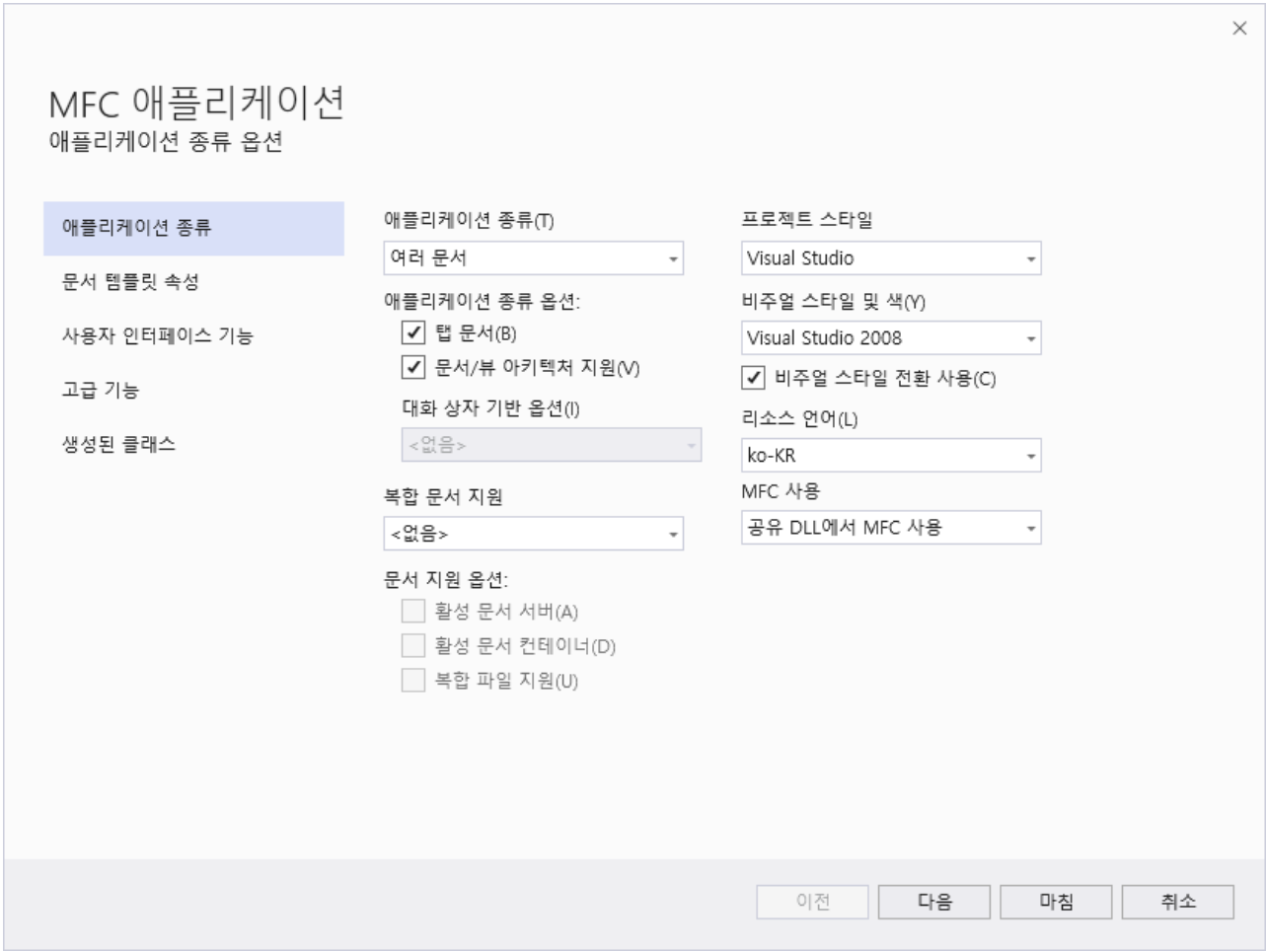
윈도우 프로그램 작성 단계

3. 이름 상자에 프로그램 이름 “Hello”를 입력.
   위치 상자에서 폴더 위치 지정



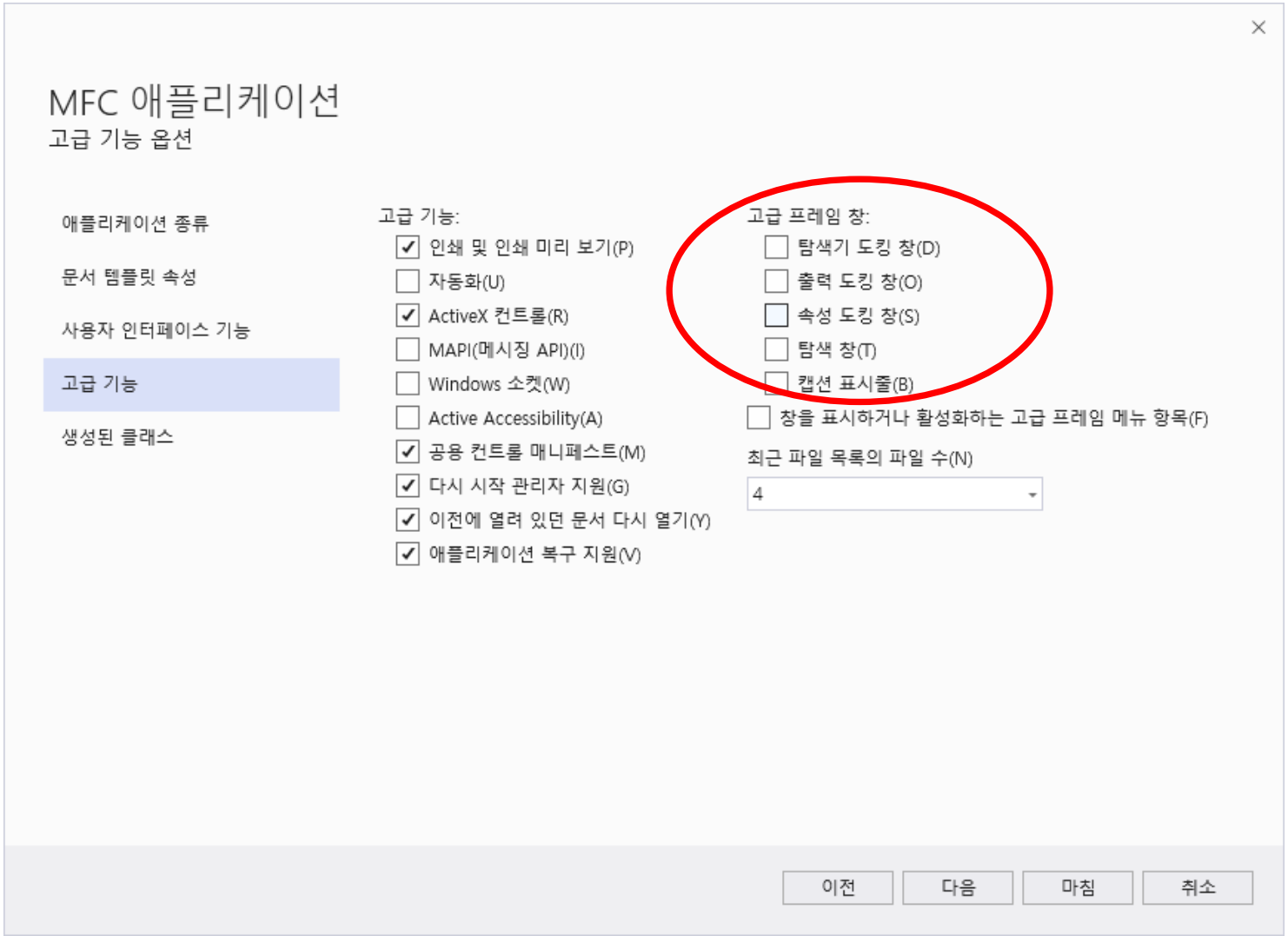
윈도우 프로그램 작성 단계

4. [MFC 애플리케이션 종류 옵션]을 설정하는 대화상자가 나타남
   - [다음] 버튼과 [이전]으로 설정 대화상자를 이동하거나
   - 대화상자의 왼쪽에 나열되어 있는 항목을 선택하여 이동 가능



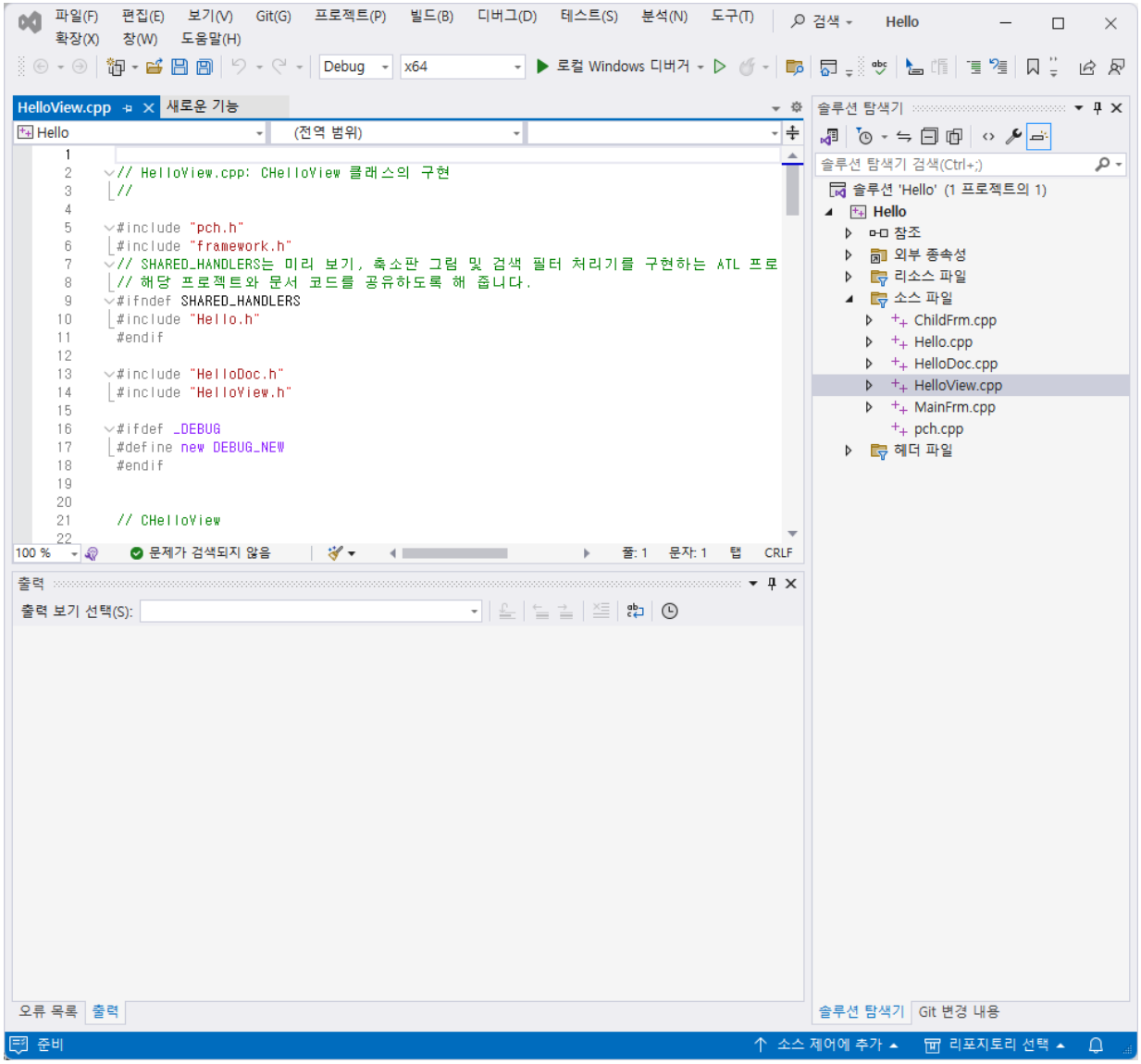
윈도우 프로그램 작성 단계

5. [고급기능] 항목을 선택하고 [고급프레임 창] 항목의 선택을 해제한 다음에 [마침] 버튼을 선택



윈도우 프로그램 작성 단계

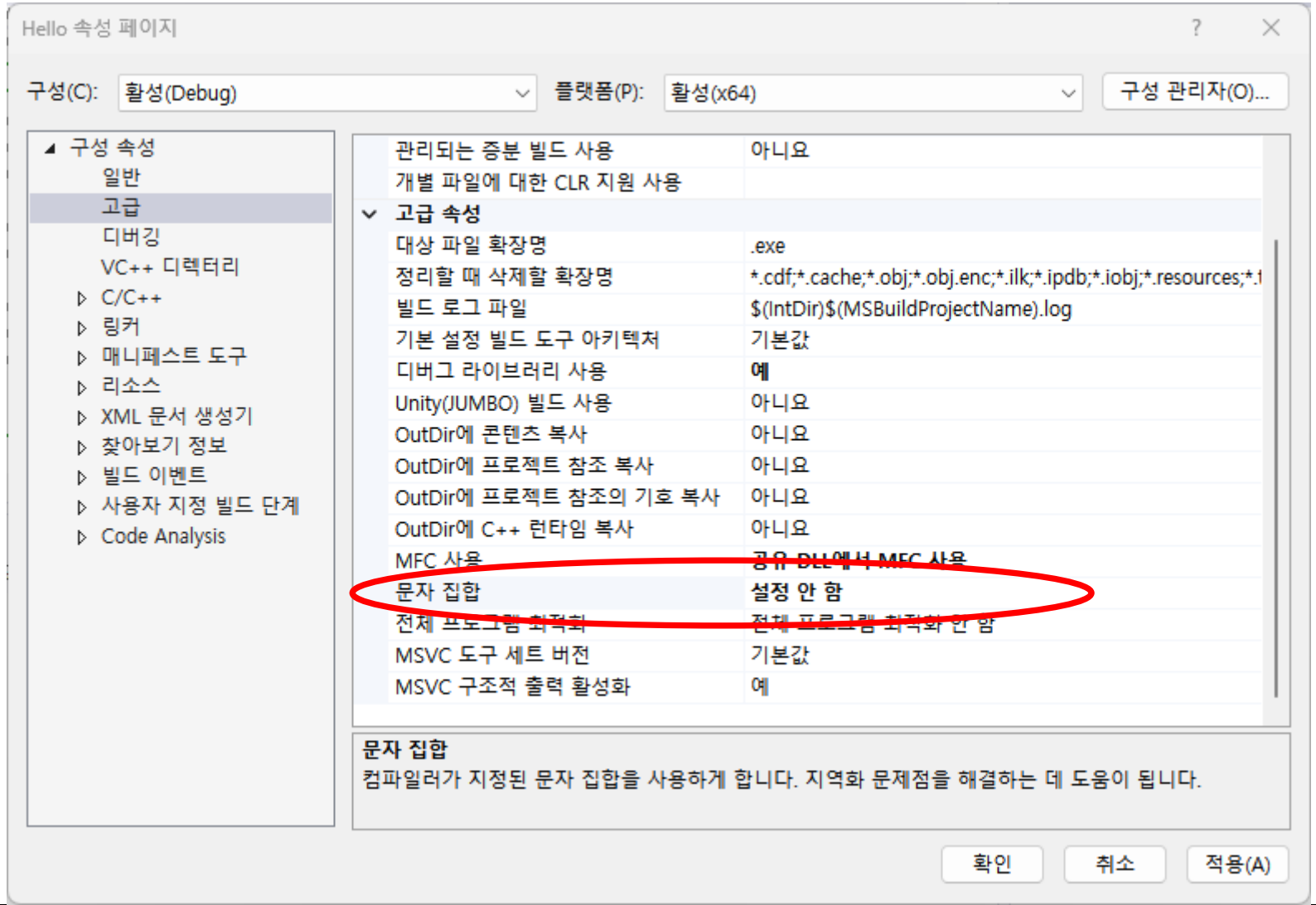
6. 설정한 사양에 따라 프로젝트가 생성되고 다음과 같은 작업 환경이 생성됨



윈도우 프로그램 작성 단계

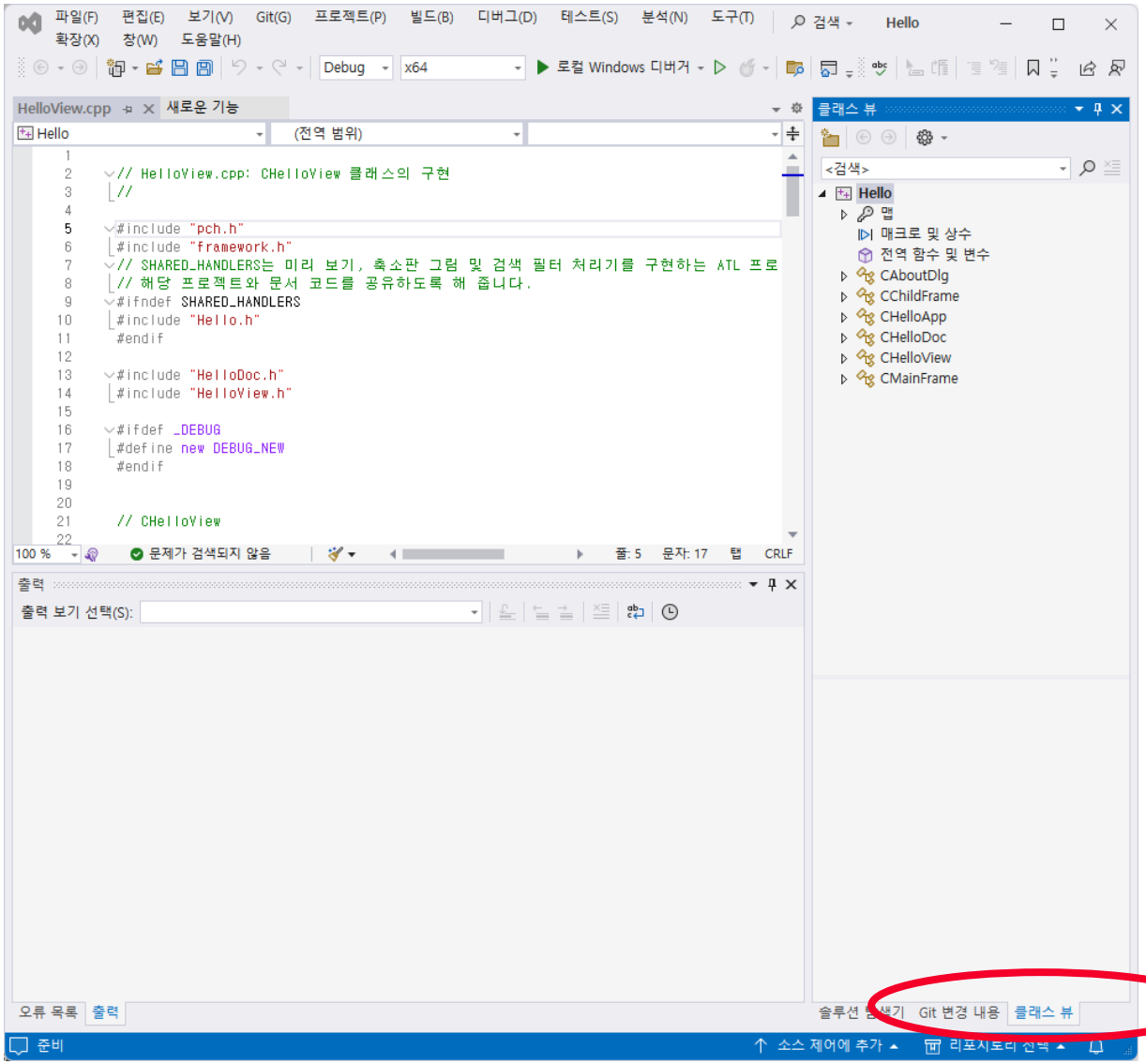
7. 문자 집합 설정

- [프로젝트] 메뉴 선택 ➔ 고급 ➔ [속성] 메뉴 선택
- [문자집합] 항목의 내용을 [설정안함]으로 선택



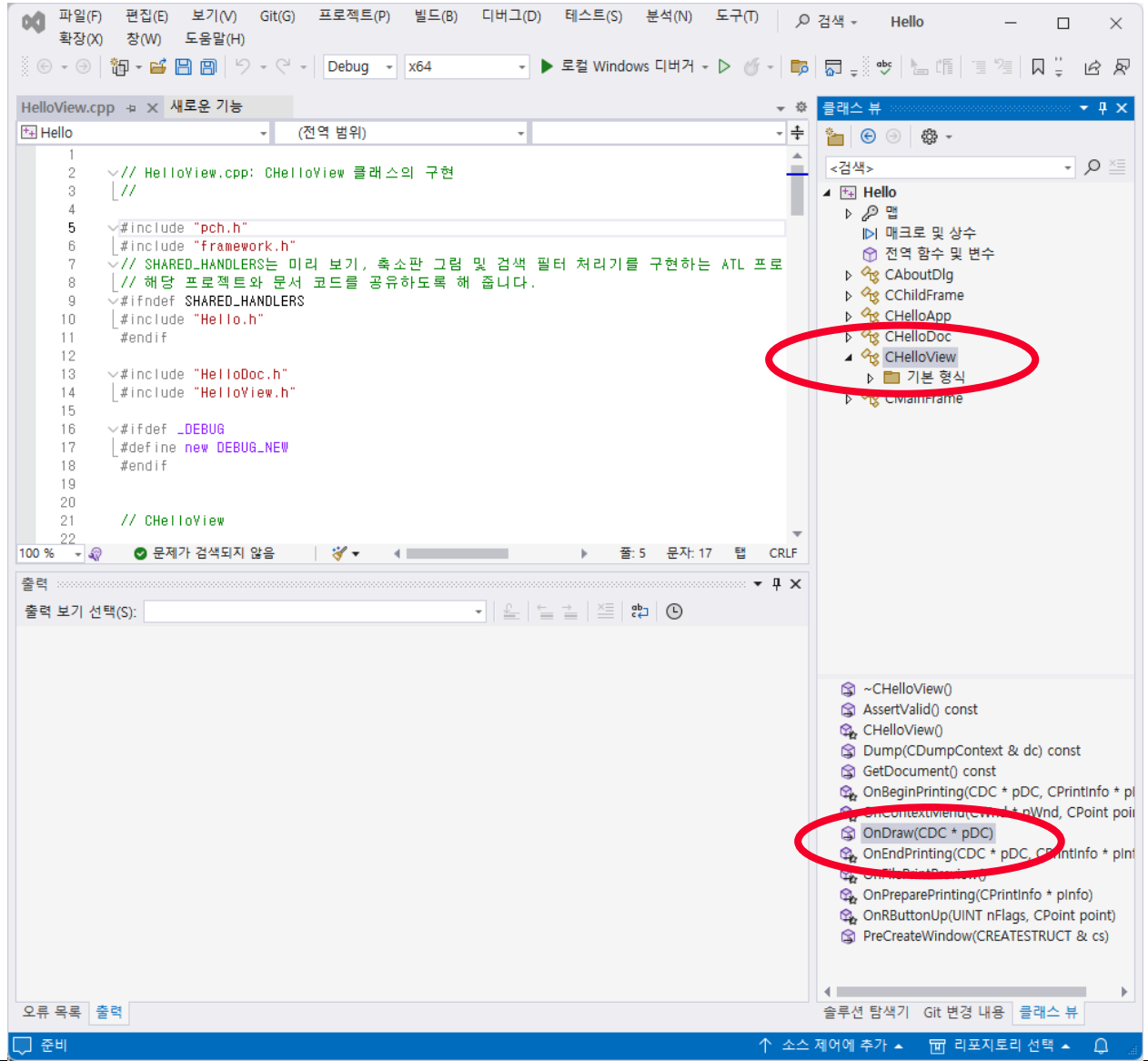
윈도우 프로그램 작성 단계

8. 오른쪽 영역에 [클래스] 탭이 나타나 있지 않으면 [보기] 메뉴에서 [클래스 뷰] 항목을 선택



윈도우 프로그램 작성 단계

9. [클래스 뷰] 탭을 선택한 다음, CHelloView 클래스를 클릭하면 아래 부분에 method 들이 나타남, 그 중에서 **OnDraw()** 함수를 두 번 클릭



10. OnDraw() 함수의 내용을 다음과 같이 편집

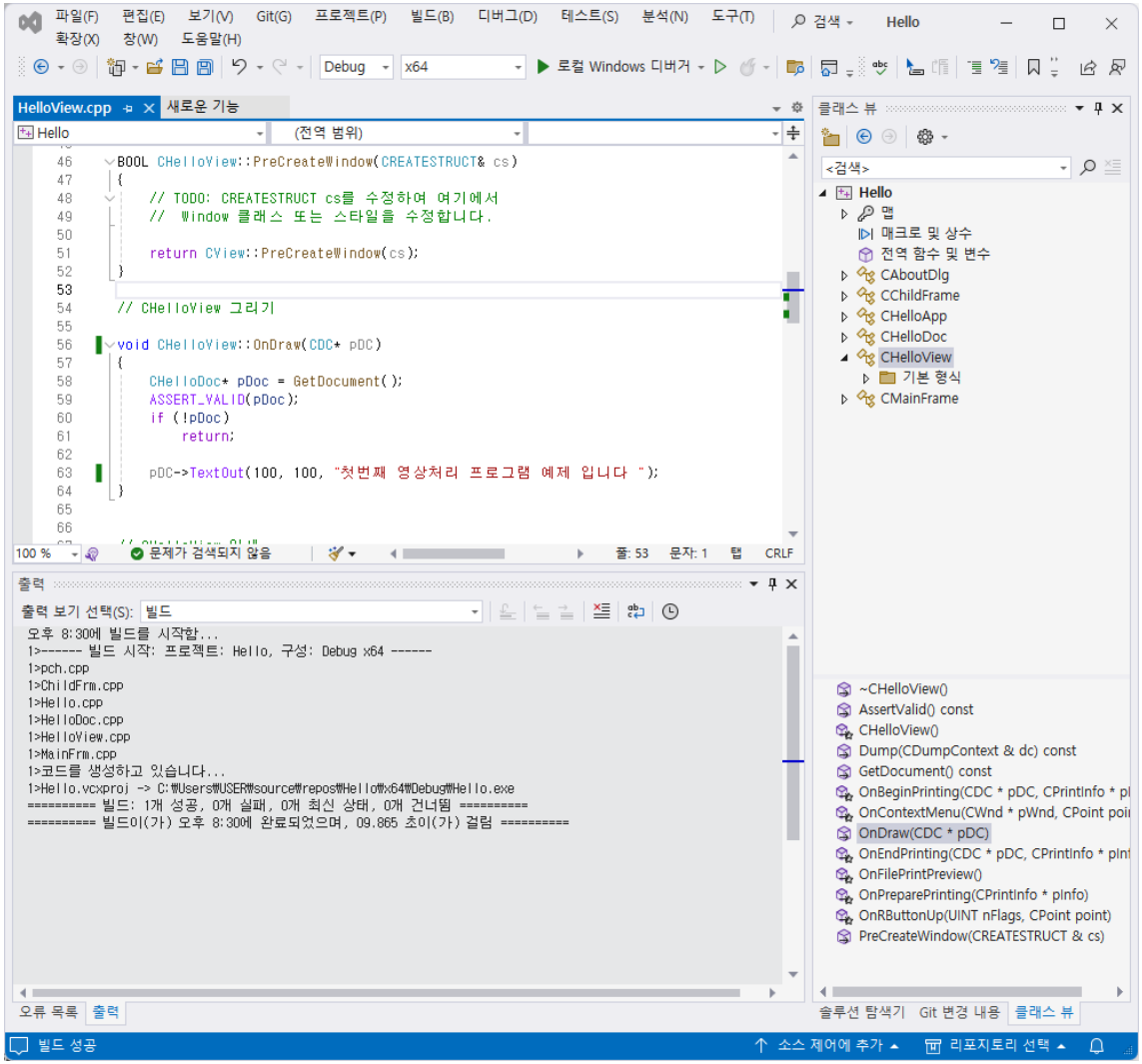
```
void CHelloView::OnDraw(CDC* pDC)
{
	CHelloDoc* pDoc = GetDocument();
	ASSERT_VALID(pDoc);

	if (!pDoc)
		return;

	pDC->TextOut(100,100, "첫번째 영상처리 프로그램 예제 입니다 ");
}
```

설명문 해제

윈도우 프로그램 작성 단계

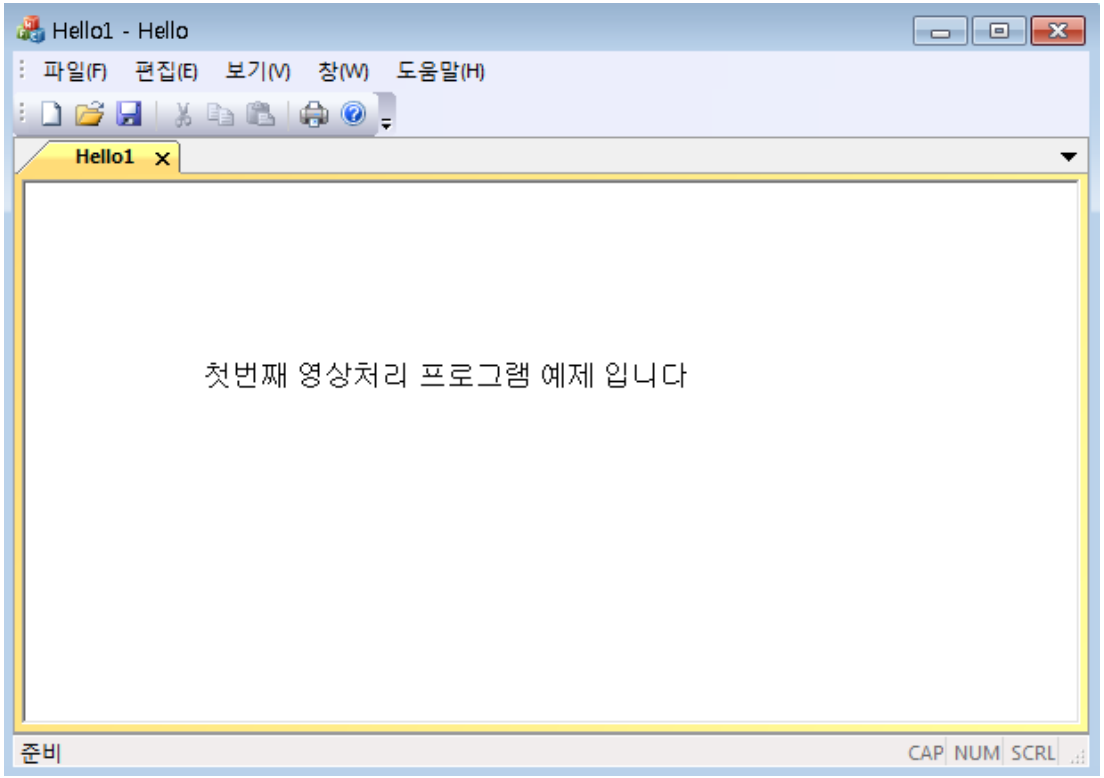
11. [Build] 메뉴의 [솔루션 빌드]를 선택

- 편집에 오류가 없으면 아래 창에 맨 마지막 줄에 “==빌드: 성공1, 실패0, 최신 0, 생략0 ==“ 이라는 메시지가 출력



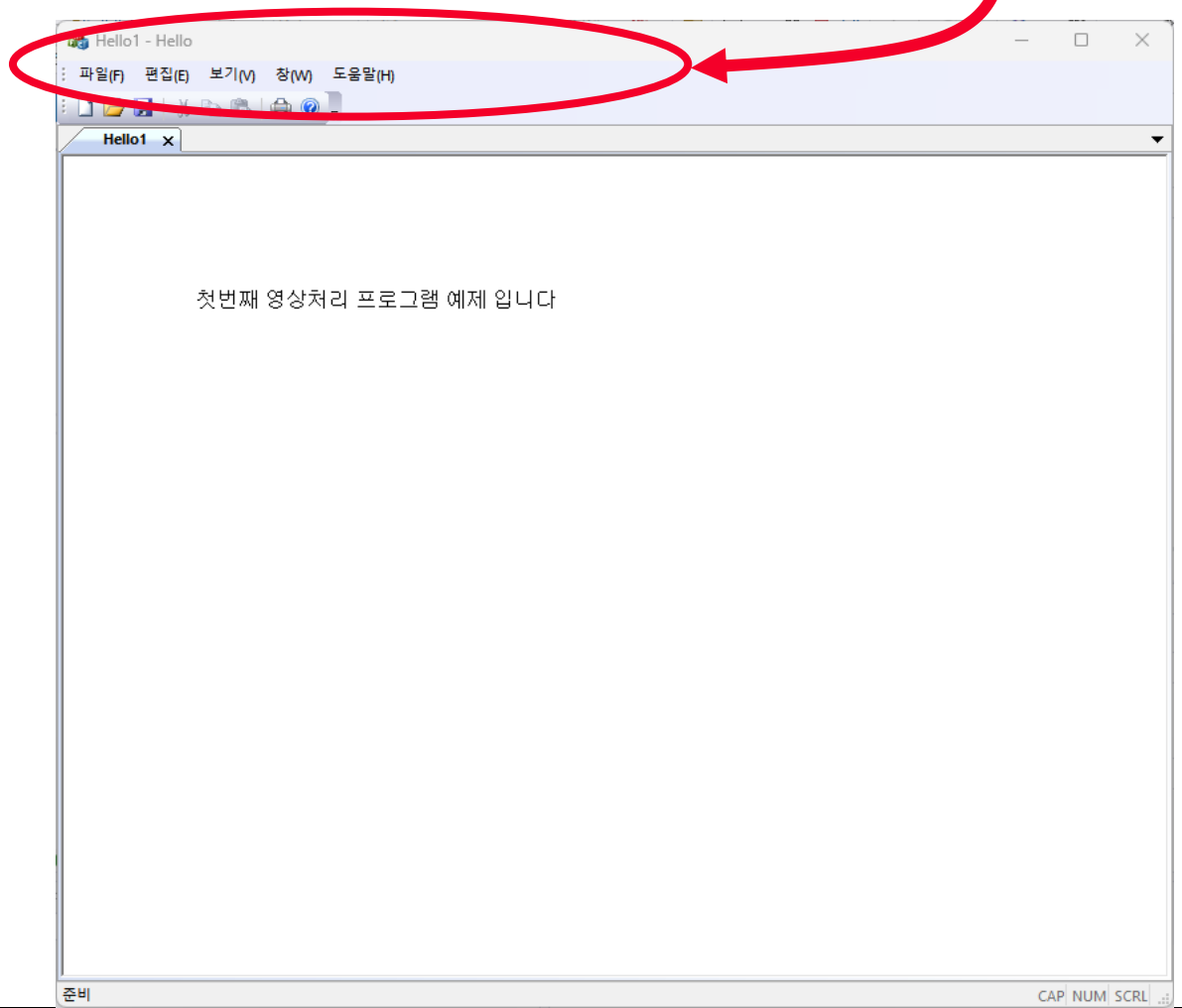
8. [디버그] 메뉴의 [디버깅 시작] 항목을 선택

- 그림과 같은 실행 결과 화면이 나타남



메뉴 생성

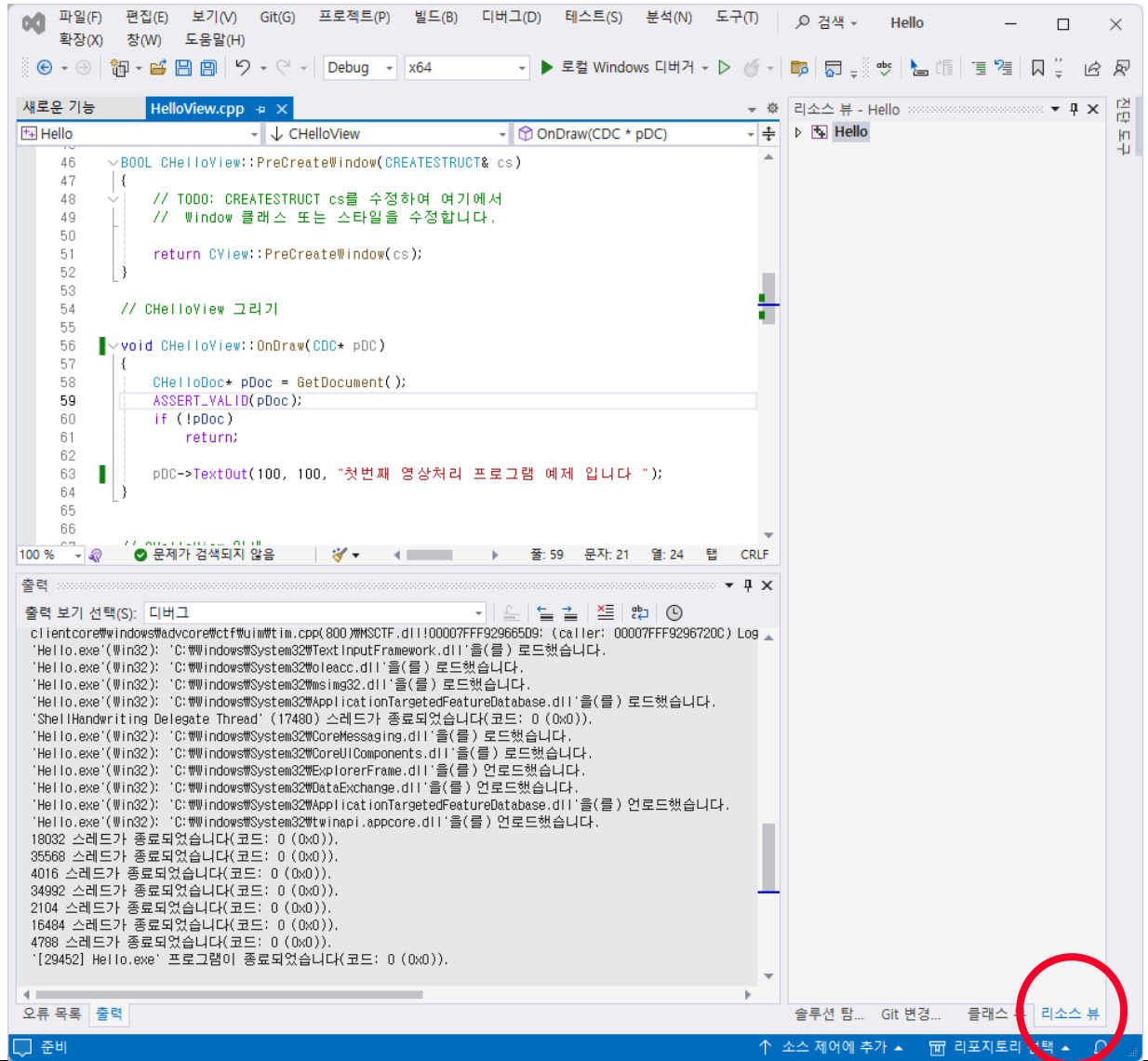
- 마법사를 이용하면 [파일], [편집], [보기], [창], [도움말]의 기본적인 메뉴가 생성됨



메뉴 생성

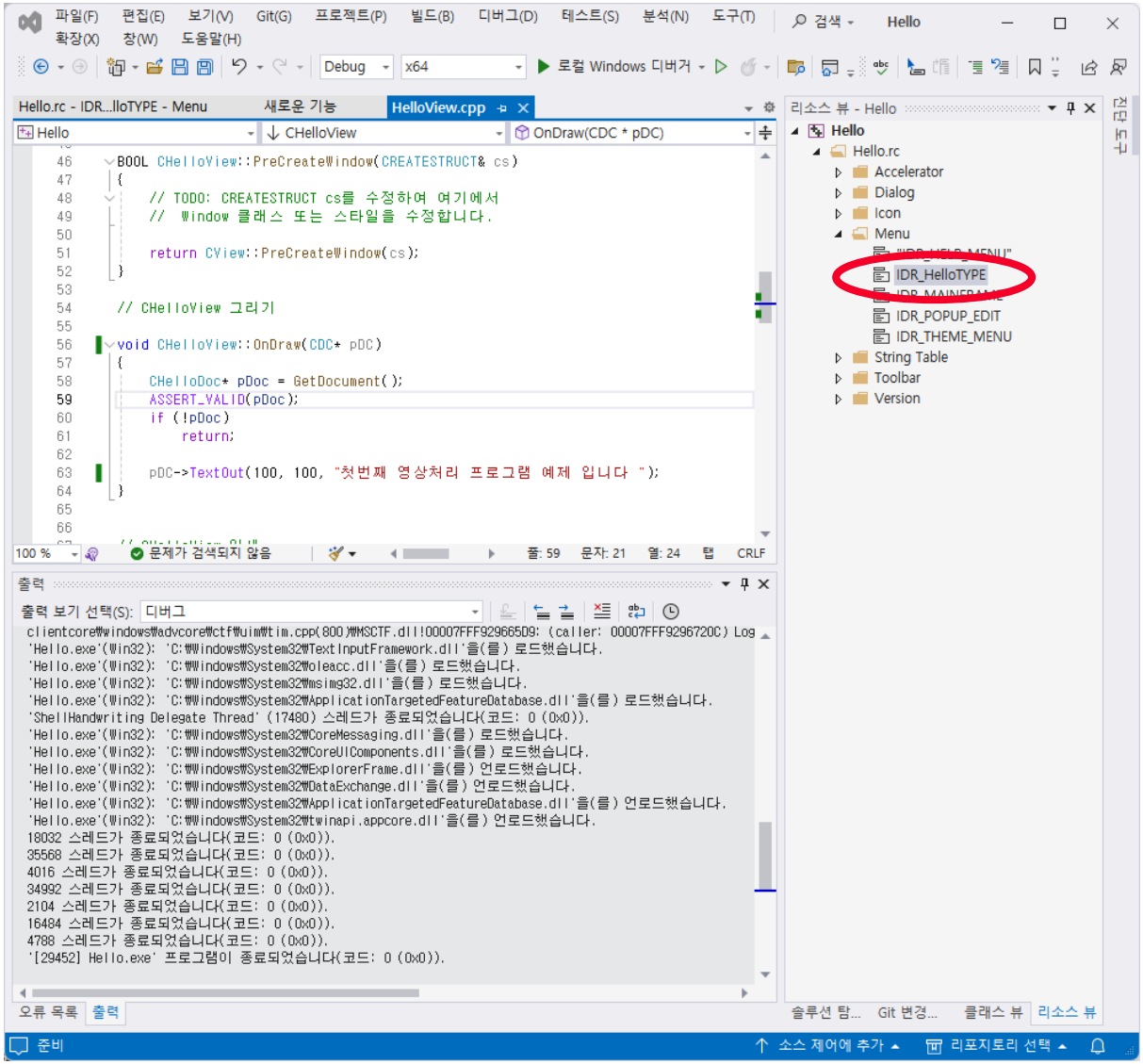
1. 프로젝트 작업 환경의 오른쪽창에서 [리소스뷰] 탭 선택

- [리소스 뷰] 탭이 없으면 [보기] ➔ [다른창]➔[리소스뷰] 선택



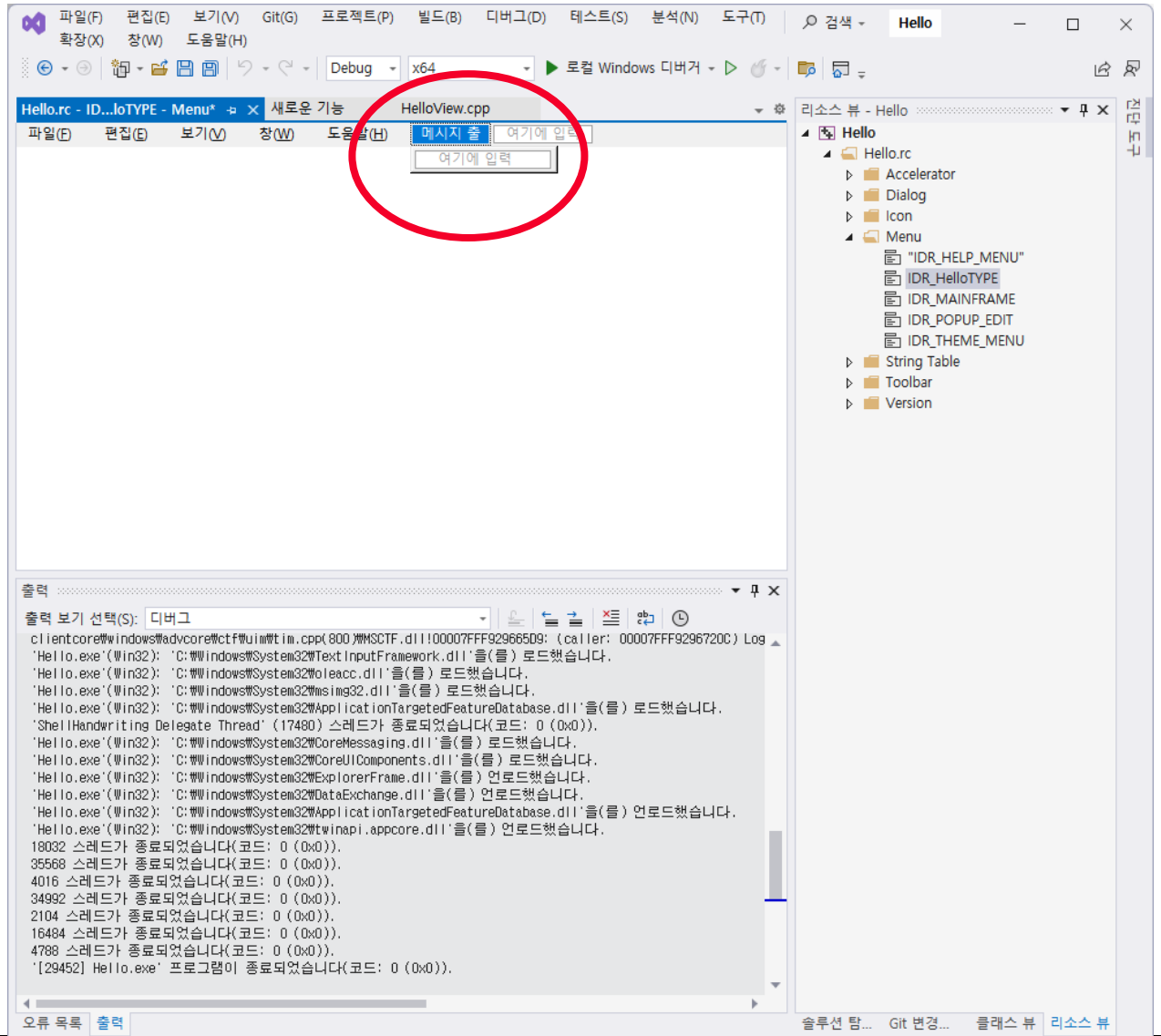
메뉴 생성

2. [리소스 파일] 목록에서 [Menu] 항목 선택
3. [Menu] 목록에서 “IDR_HelloTYPE” 항목을 두 번 클릭



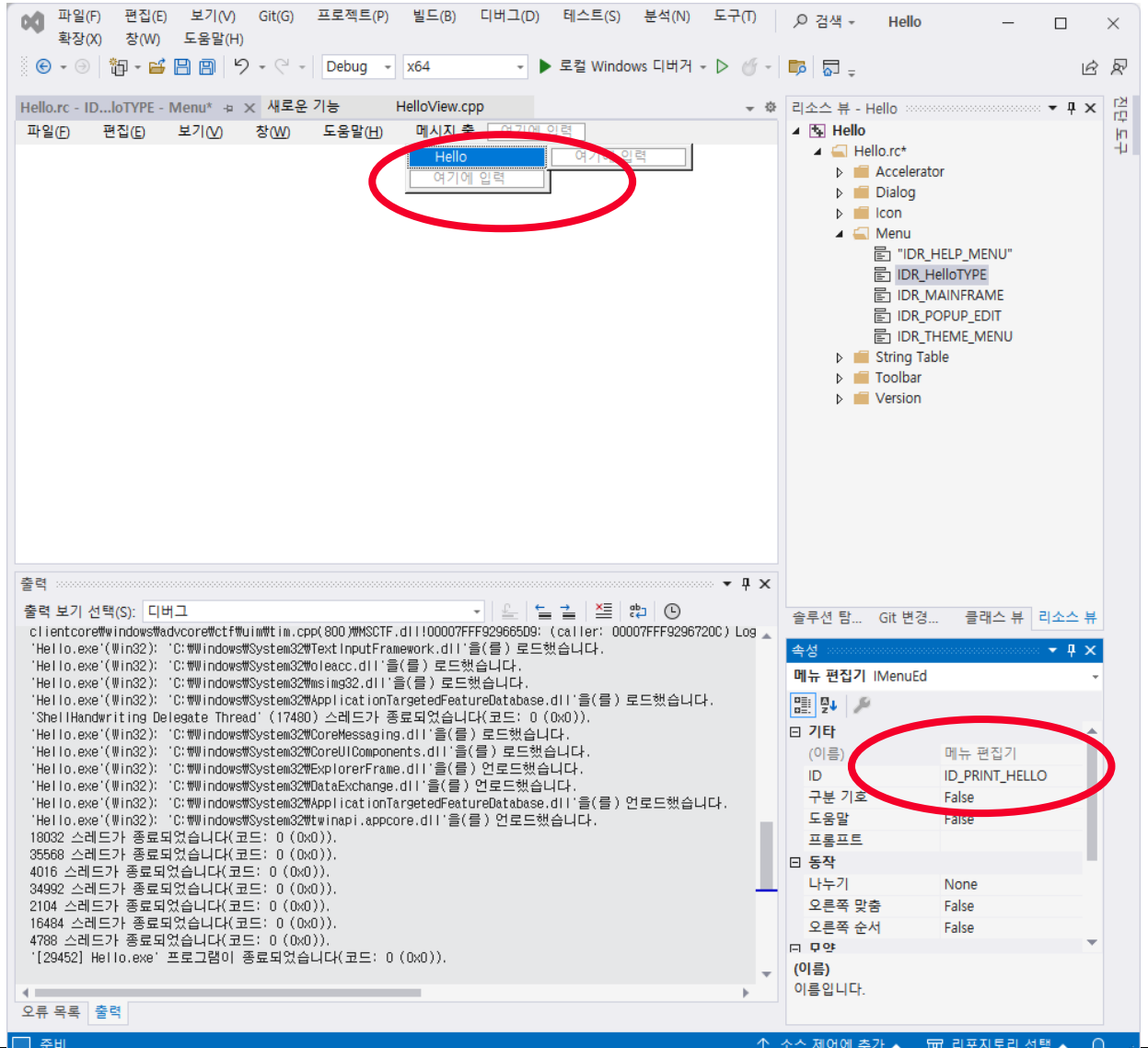
4. 새로운 메뉴 추가

- 메뉴 막대에서 점선 사각형에 메뉴 이름 “메시지 출력” 입력



부메뉴 생성 및 연결 함수 작성

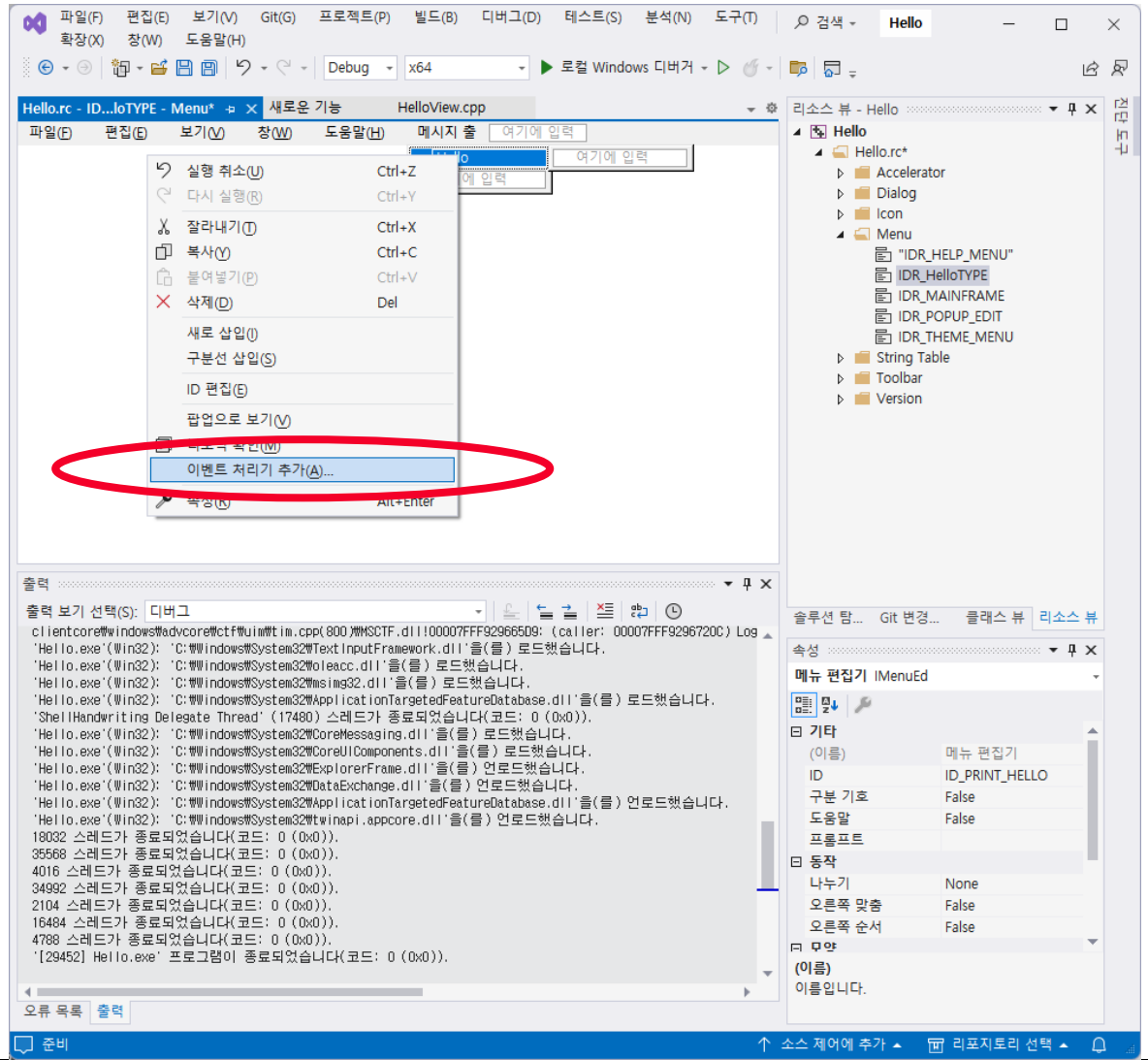
1. [메시지 출력] 메뉴아래 점선사각형에 부메뉴 이름을 “Hello”로 입력, 오른쪽 영역에 있는 속성창의 ID 항목에 “ID_PRINT_HELLO” 입력



부메뉴 생성 및 연결 함수 작성

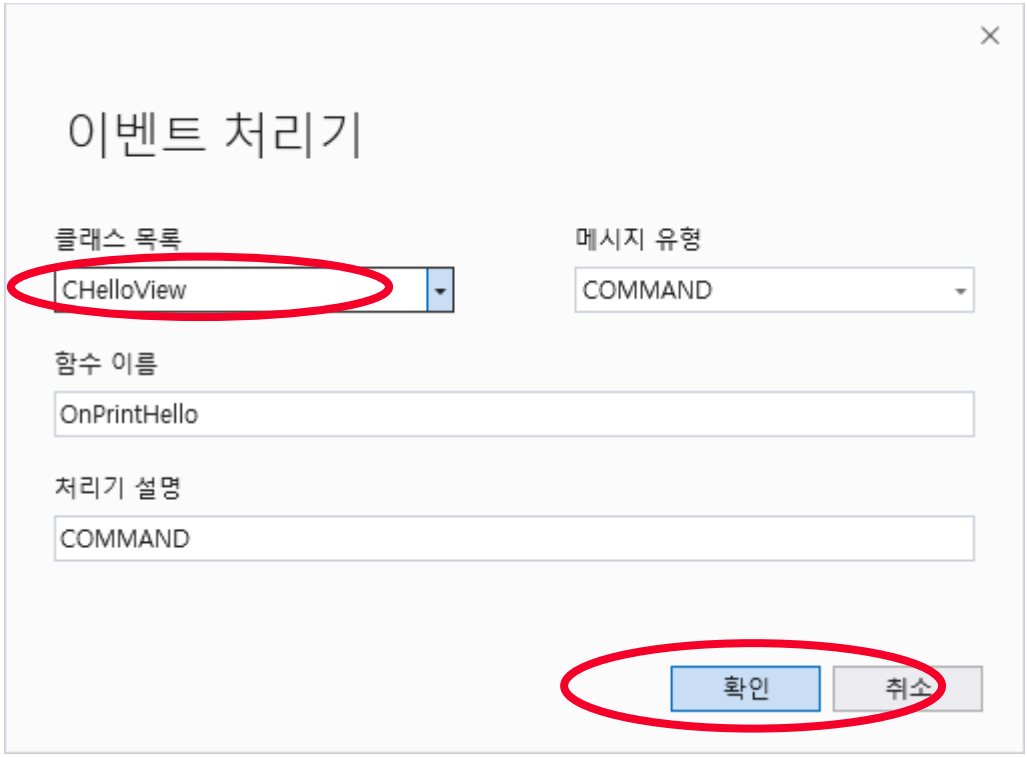
2. 부메뉴를 선택했을 경우에 수행될 함수 생성

- [Hello] 부메뉴를 마우스 오른쪽 버튼으로 클릭한 다음 [이벤트 처리기 추가] 항목 선택



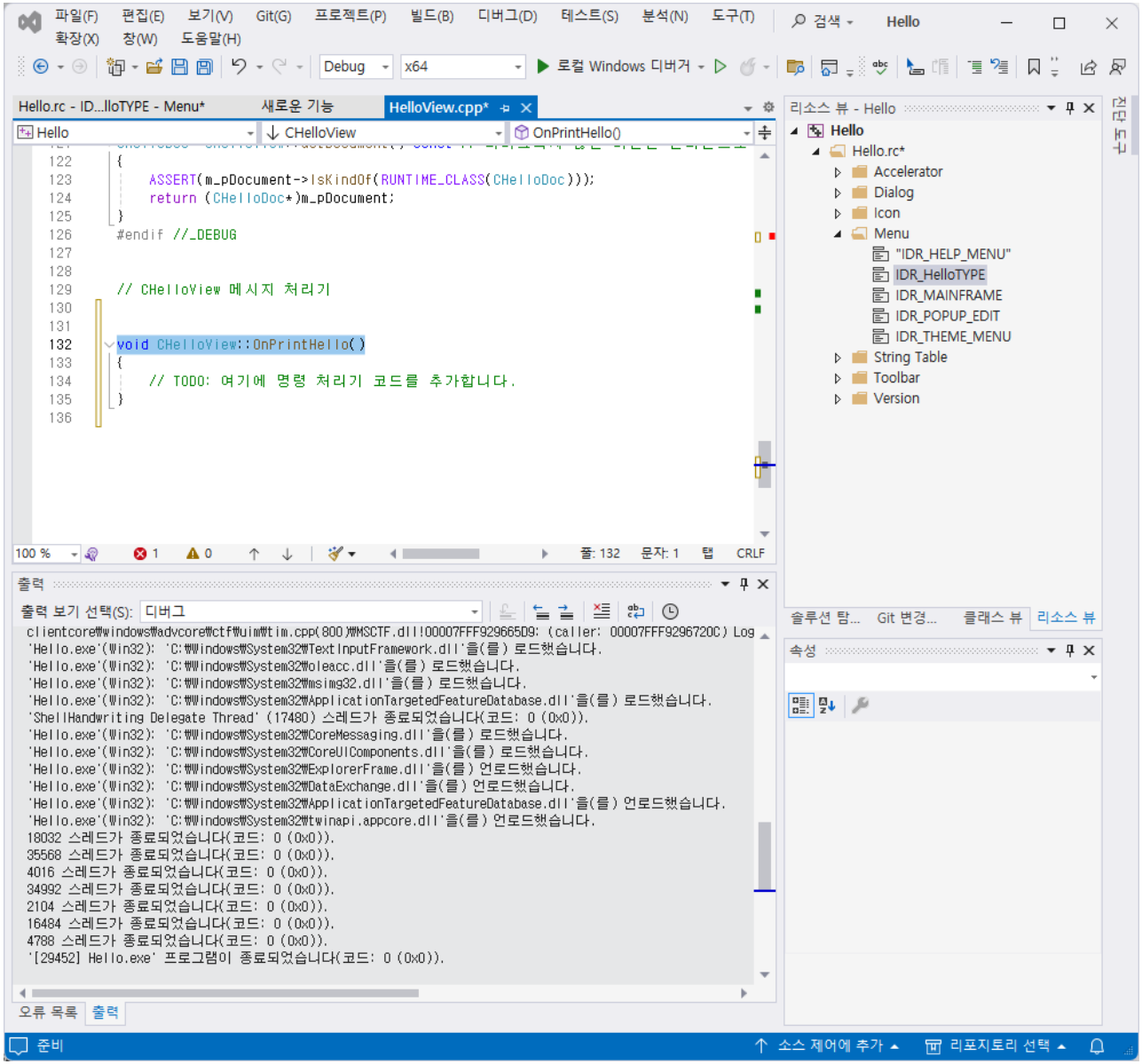
부메뉴 생성 및 연결 함수 작성

3. 클래스 목록에서 “CHelloView”를 선택한 다음에 [추가 및 편집] 메뉴 선택



부메뉴 생성 및 연결 함수 작성

4. 함수 편집 화면이 나타남



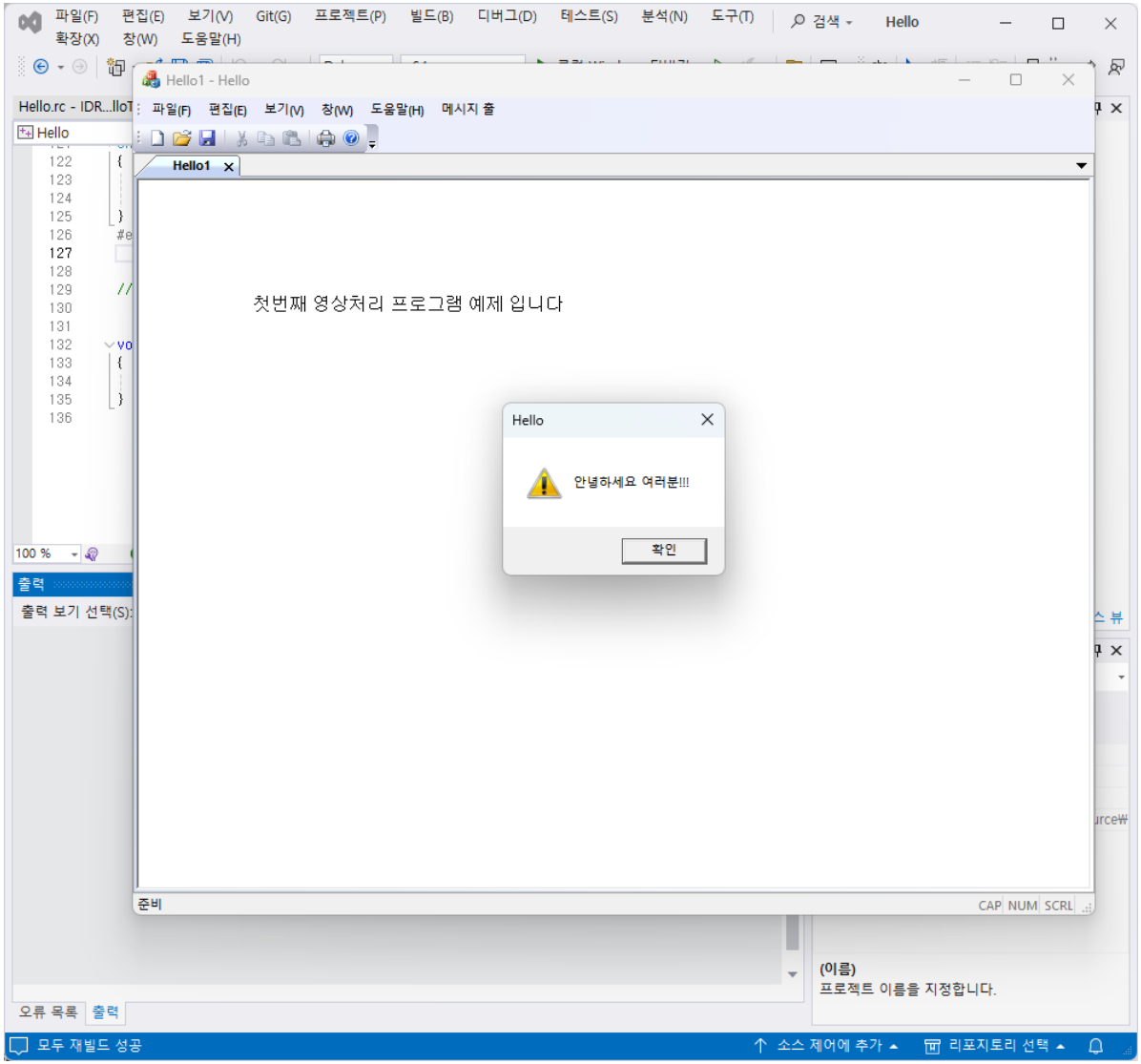
5. OnPrintHello() 함수 내용을 다음과 같이 편집

```
void CHelloView::OnPrintHello()
{
    AfxMessageBox("안녕하세요 여러분!!!");
}
```

Afx : Application Framework X
- DirextX, ActiveX

부메뉴 생성 및 연결 함수 작성

6. 프로그램을 컴파일하고 실행



영상 출력 프로그램 작성

- 프로젝트 생성
- 영상 저장을 위한 변수 선언
- 영상 파일 입출력
- 스크롤 윈도우 크기 설정
- 화면 출력
- 컴파일 및 실행

영상 출력 프로그램(프로젝트 생성)

- 프로젝트 생성
  1. [파일] 메뉴 -> [새로만들기] -> [프로젝트] 항목 선택
  2. [새 프로젝트] 대화 상자에서 [MFC 앱] 항목 선택
     - [이름] 상자에 프로젝트 이름을 입력
       - 영상 처리 프로그램이란 뜻으로 "ImagePro_영문이름" 을 사용
         - » 예) ImagePro_KilDong
     - [위치] 상자에 위치할 폴더를 입력한 다음에 [OK] 버튼 클릭

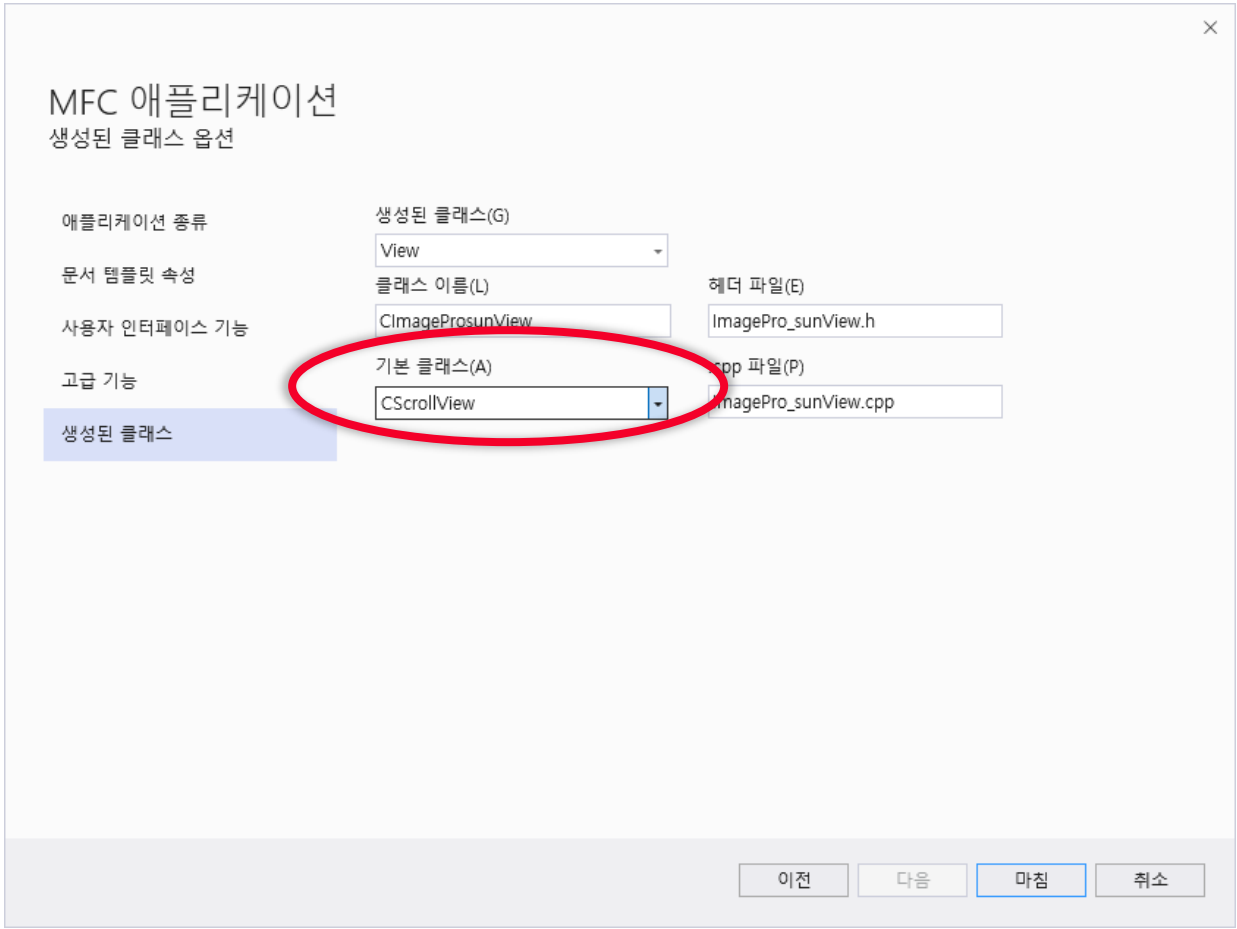
영상 출력 프로그램(프로젝트 생성)

3. [MFC 응용프로그램마법사 – 고급기능] 대화 상자에서 다음 항목이 체크되지 않도록 설정
   - 탐색기 도킹 창
   - 출력 도킹 창
   - 속성 도킹 창

영상 출력 프로그램(프로젝트 생성)

4. MFC 응용프로그램 마법사 – 생성된 클래스대화 상자에서 CImageProView의 기본 클래스를 CScrollView로 설정
   - 영상이 창의 크기보다 큰 경우에 스크롤해가면서 볼 수 있음

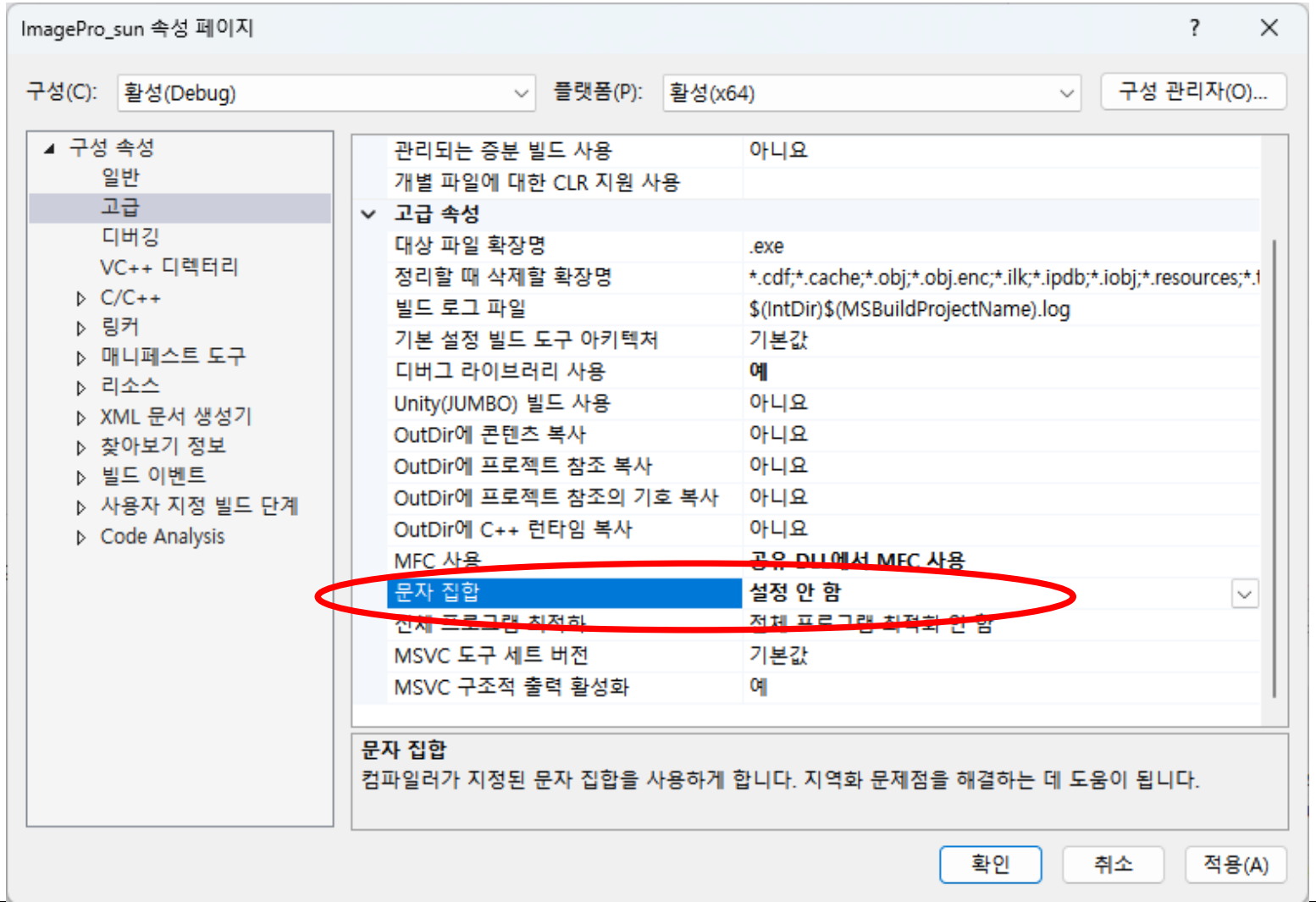
5. [마침] 버튼을 눌러 프로젝트 생성



윈도우 프로그램 작성 단계

6. 문자 집합 설정

- [프로젝트] 메뉴 선택 ➔ [속성] 메뉴 선택
- [문자집합] 항목의 내용을 [설정안함]으로 선택

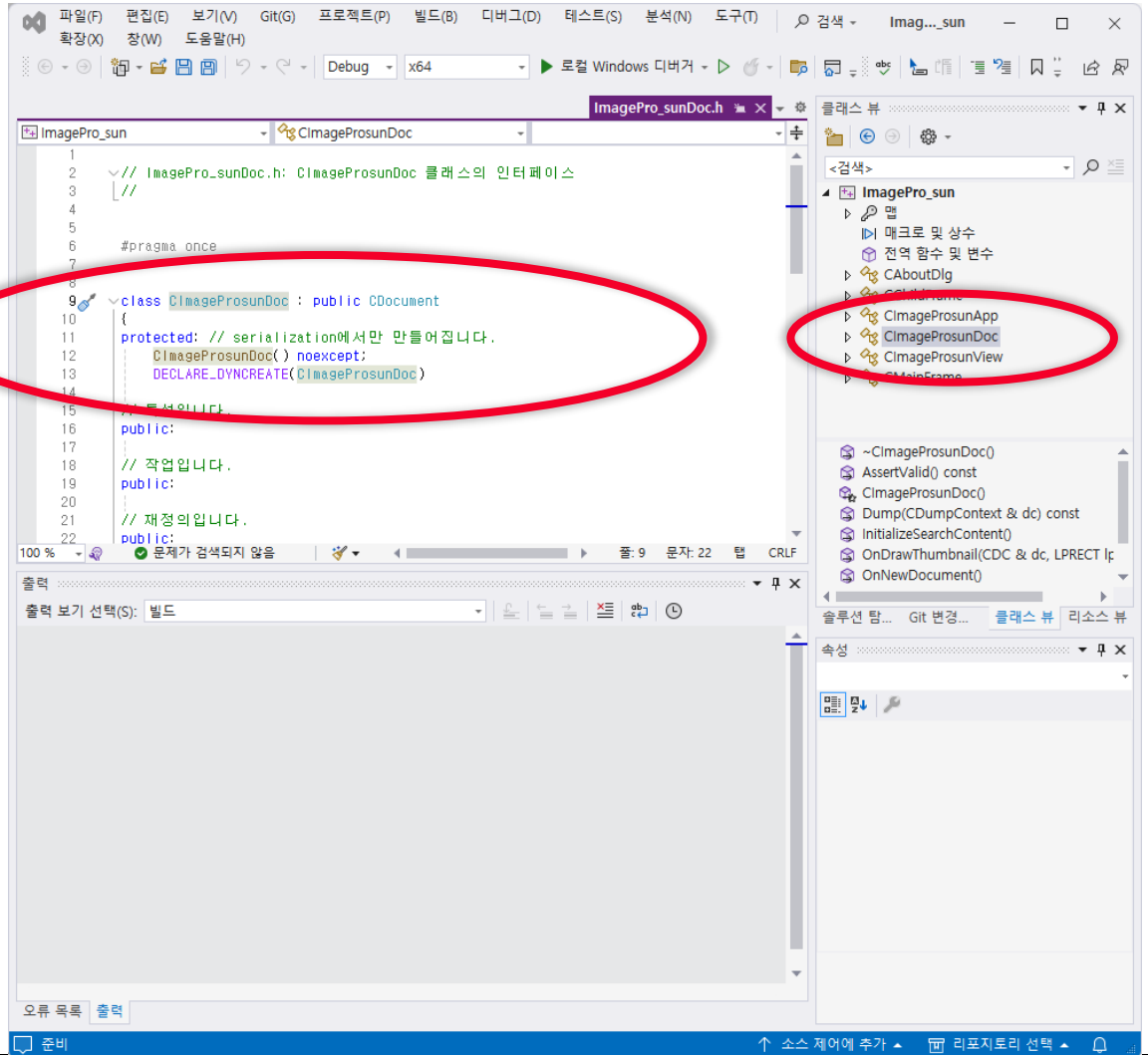


영상 출력 프로그램(변수 선언)

- 영상 저장을 위한 변수 선언
  - 영상을 읽어 들여 처리하기 위해서는 영상을 저장할 기억장소 공간이 필요
  - 영상은 픽셀 값들의 이차원 형태로 나열된 것이므로 이차원 배열을 사용

영상 출력 프로그램(변수 선언)

1. 프로젝트 작업 환경에서 [클래스 뷰] 탭을 선택하고 "CImageProDoc" 클래스를 더블 클릭하면 오른쪽 창에 CImageProDoc 클래스의 정의가 나타남



2. CImageProDoc 클래스에 변수 선언

```
class CImageProDoc : public CDocument
{
   ...
// 특성입니다
public:
   unsigned char  inputImg[256][256];
   unsigned char  resultImg[256][256];
   ...
}
```

영상 출력 프로그램(변수 선언)

- 변수 선언
  - 변수명
    - inputImg : 입력 영상을 위한 공간
    - resultImg : 영상 처리 결과 저장을 위한 공간

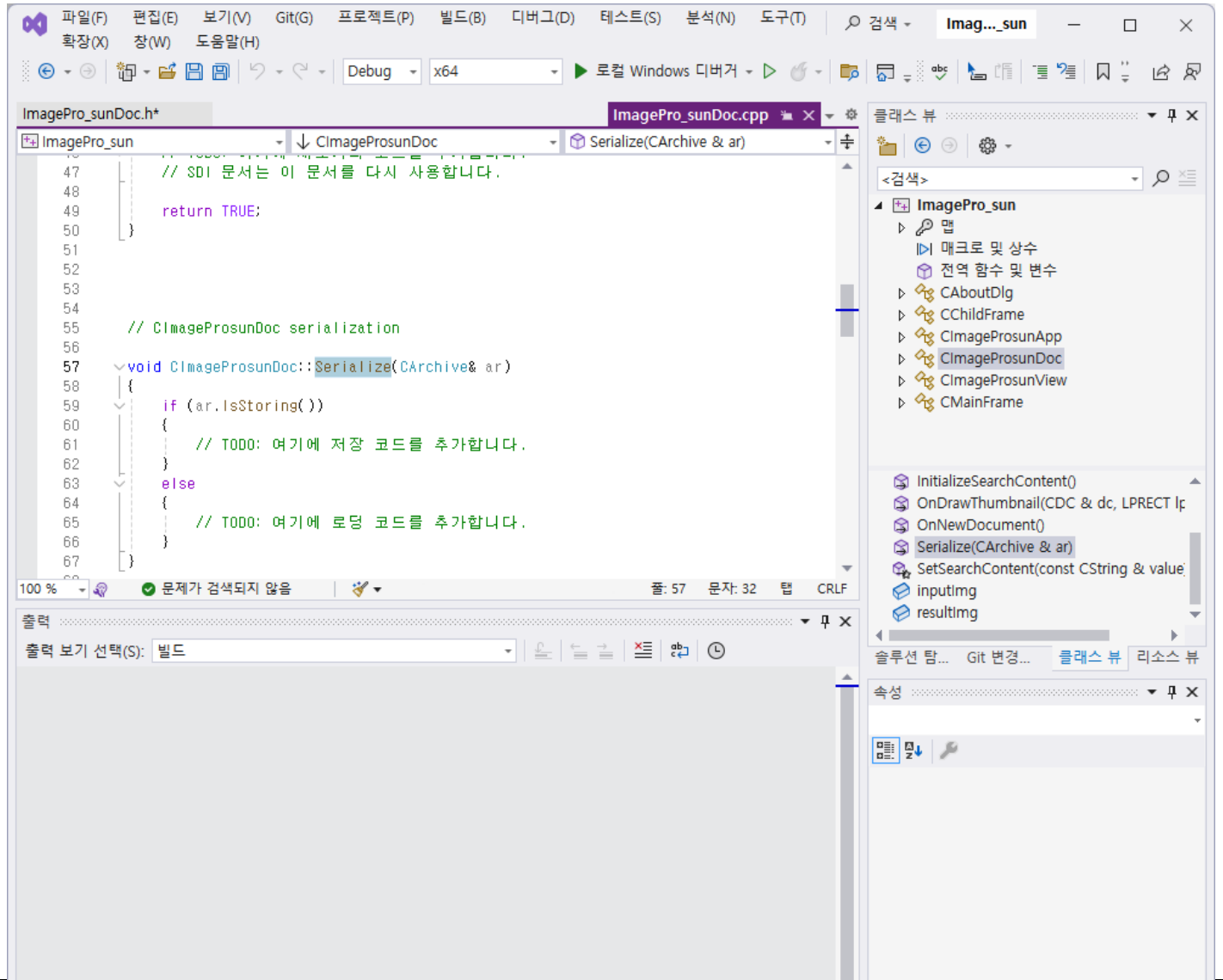
  - 크기가 256x256인 이차원 배열로 선언
  - 가장 단순한 형태의 선언으로서
    크기가256x256인 흑백 영상을 저장할 수 있음

영상 출력 프로그램(파일 입출력)

- MFC 응용프로그램 마법사에 의해 생성된 기본 프로그램에는 영상 파일 입출력을 위해 Serialize() 함수가 제공됨

영상 출력 프로그램(파일 입출력)

1. ImagePro 프로그램 작업환경에서 CImageProDoc 클래스의 Serialize() 함수를 클릭



2. Serialize() 함수의 내용을 다음과 같이 편집

```
void CImageProDoc::Serialize(CArchive& ar)
{
  if (ar.IsStoring() == TRUE)
  {
    ar.Write(resultImg, 256 * 256);
  }
  else
  {
    CFile *fp = ar.GetFile();
    if (fp->GetLength() == 256 * 256) ar.Read(inputImg, 256 * 256);
    else AfxMessageBox("256x256 크기의 파일만 사용가능합니다.");
  }
}
```

영상 출력 프로그램(파일 입출력)

- ar.Read(inputImg, 256 * 256) 함수 호출
  - 파일로부터 256 x 256 바이트의 데이터를 읽어들에 inputImg 배열에 저장

- GetLength() 함수 호출
  - 파일의 크기를 반환

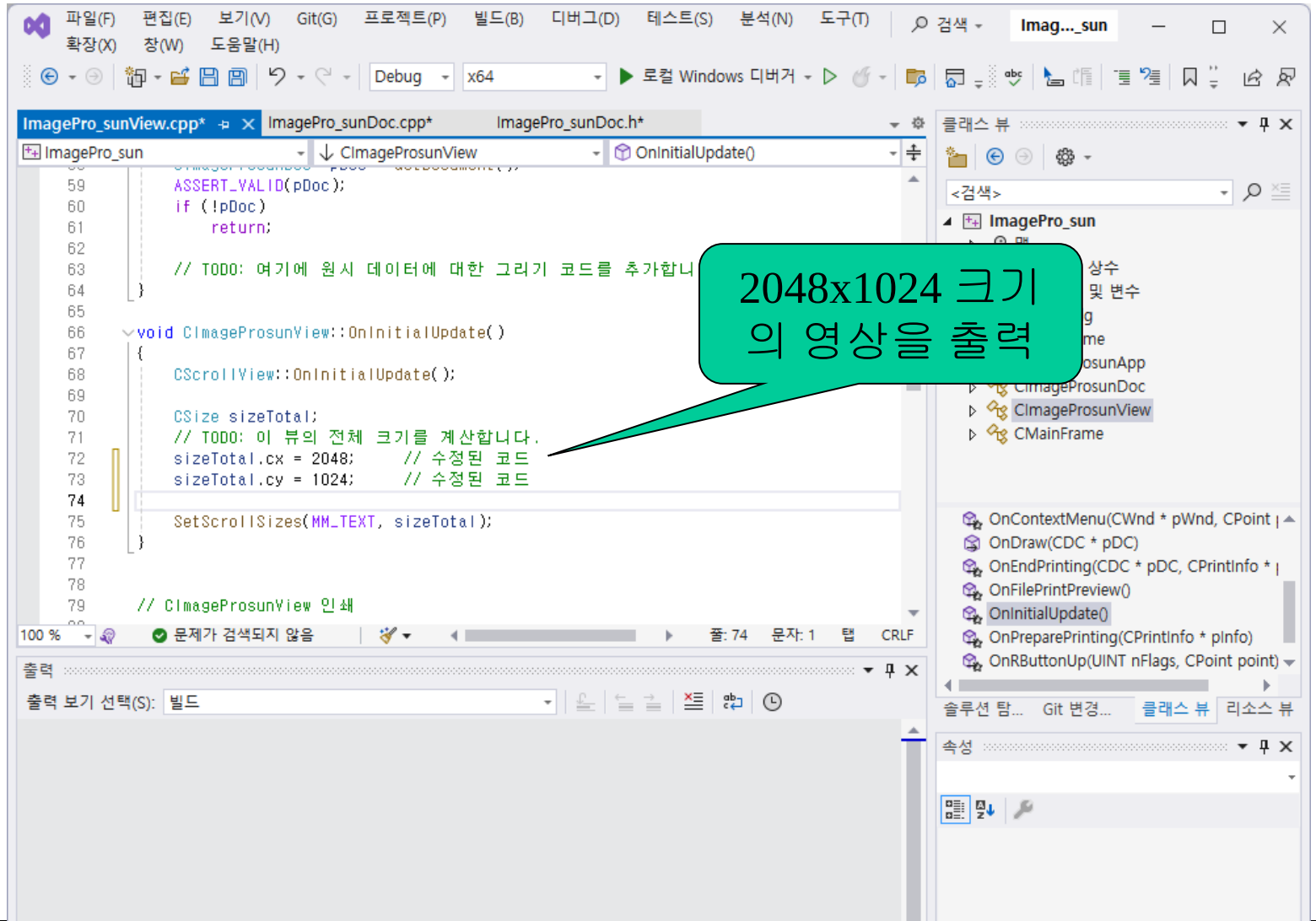
- ar.Write(resultImg, 256 * 256) 함수 호출
  - resultImg 배열의 영상을 출력 파일에 저장

영상 출력 프로그램(스크롤 설정)

- 스크롤 윈도우 크기 설정
  - 윈도우의 크기보다 영상의 크기가 클 경우에는 스크롤시키면서 영상을 볼 수 있도록 해야 함
  - 스크롤 크기는 CImageProView 클래스의 OnInitialUpdate() 함수에서 지정

영상 출력 프로그램(스크롤 설정)

1. CImageProView 클래스의 OnInitialUpdate() 함수를 선택
2. OnInitialUpdate() 함수의 내용을 다음과 같이 편집



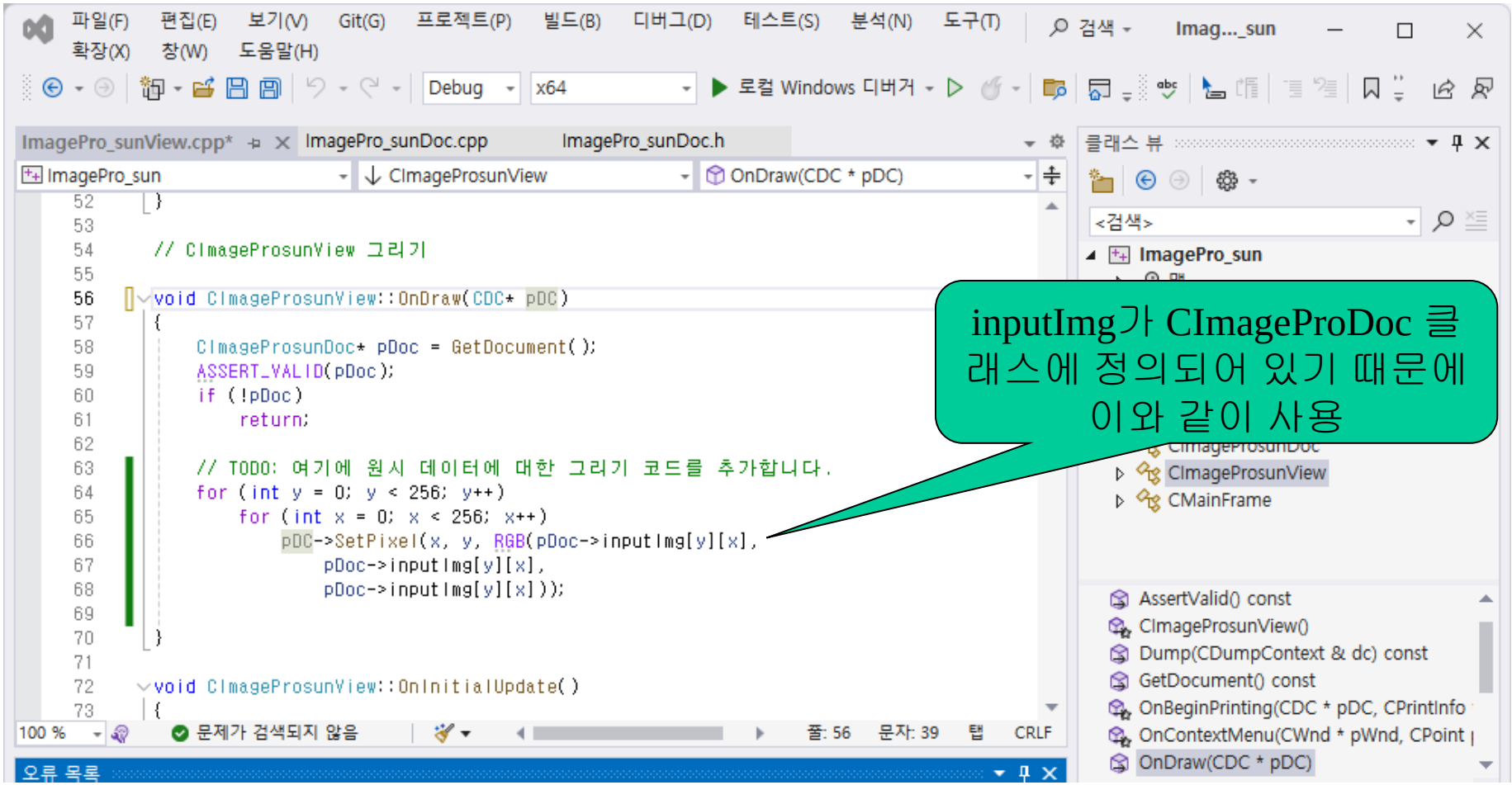
영상 출력 프로그램(화면 출력)

- 화면 출력은 CImageProView 클래스의 OnDraw() 함수에서 관장함

- 화면 출력에 관한 기능은 OnDraw() 함수에 작성

영상 출력 프로그램(화면 출력)

- CImageProView 클래스의 OnDraw() 함수를 선택
- OnDraw() 함수의 내용을 다음과 같이 편집



영상 출력 프로그램(화면 출력)

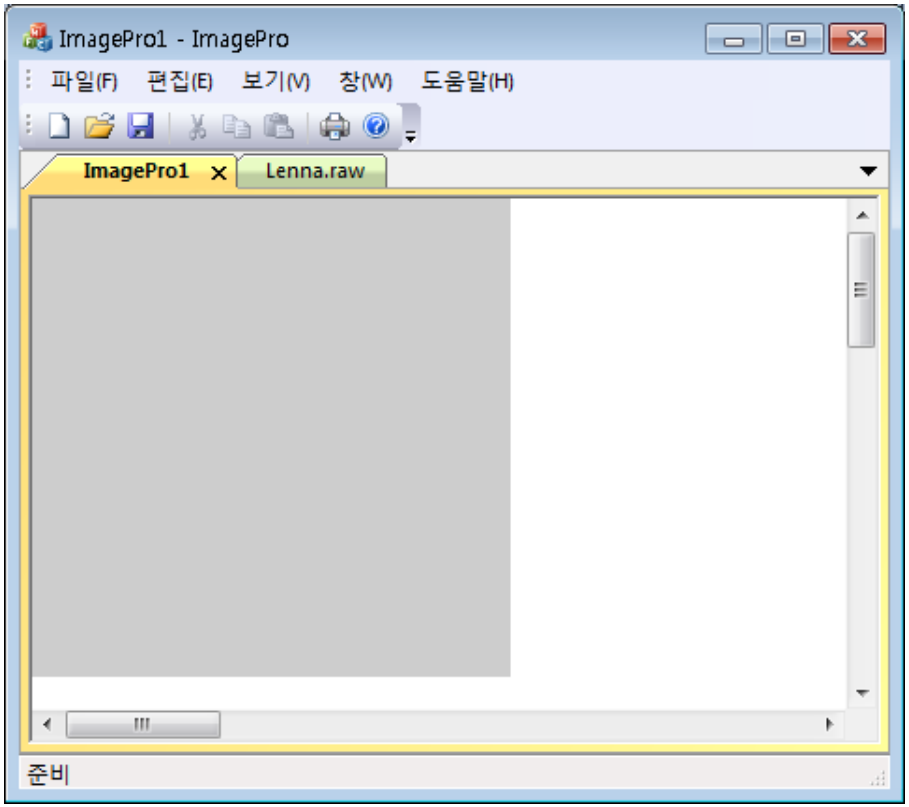
- **SetPixel(x, y, color) 함수 호출**
  - (x, y) 위치에 color 값의 점을 그려줌

- **RGB(r, g, b) 함수 호출**
  - 빨강색 성분의 값이 r이고 초록색 성분 값이 g 이고 파랑색 성분 값이 b인 색을 반환

- **GetDocument() 함수 호출**
  - CImageProDoc 클래스 객체에 대한 포인터를 반환
    - CImageProDoc 클래스에 선언되어 있는 변수나 함수를 사용하기 위해서는 이 포인터를 사용해야 함

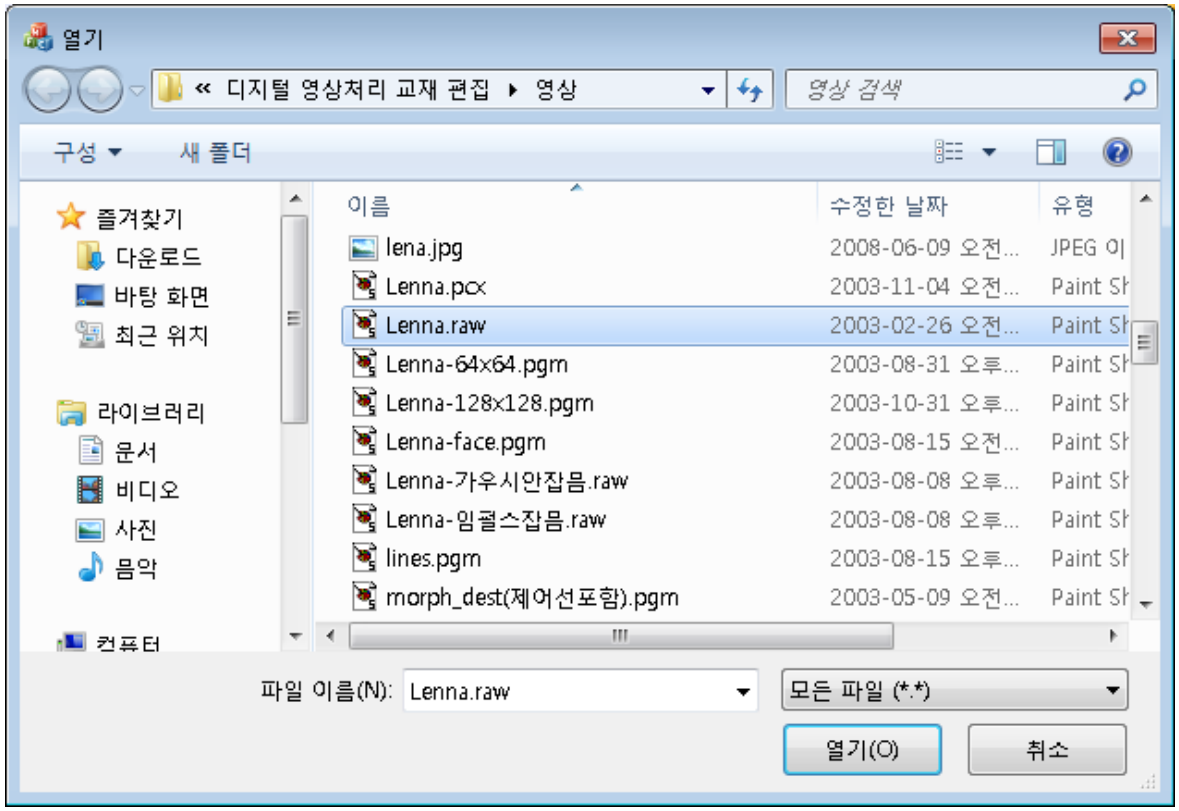
영상 출력 프로그램(컴파일 및 실행)

- 컴파일을 수행하고 프로그램을 실행하면 다음과 같은 프로그램 윈도우가 나타남



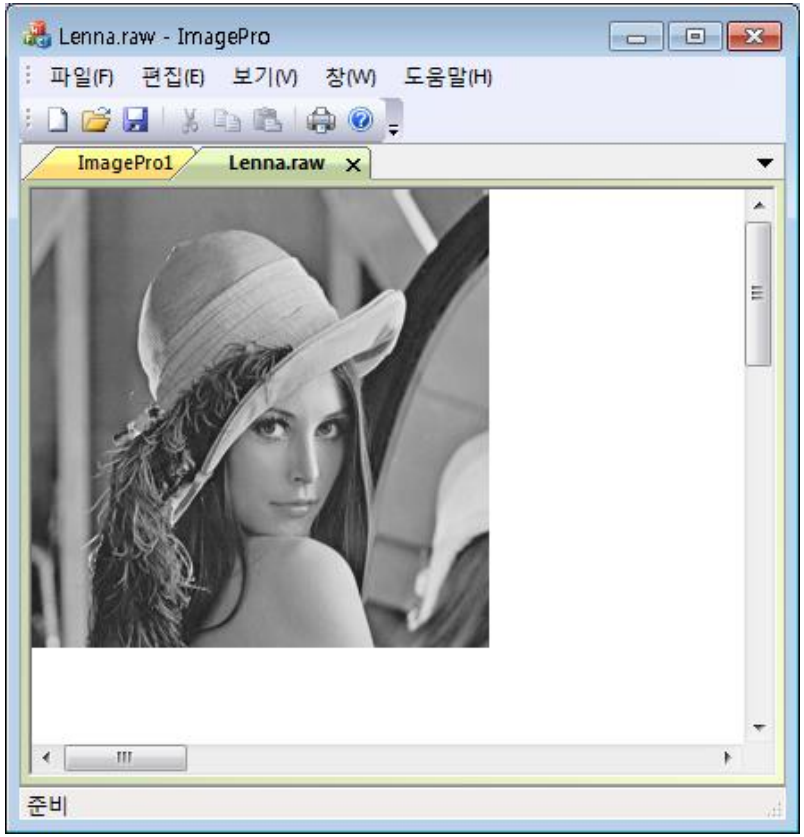
영상 출력 프로그램(컴파일 및 실행)

- [파일] 메뉴에서 [열기] 항목을 선택



영상 출력 프로그램(컴파일 및 실행)

- 파일 열기 대화 상자에서 영상이 저장되어 있는 폴더를 찾은 다음에 Lenna.raw 파일을 선택



기본 클래스

- MFC 응용프로그램 마법사를 이용하여 ImagePro 프로젝트를 생성하면 여섯 개의 클래스가 생성됨
  - CImageProDoc 클래스
    - 여기에서 Doc는 문서(Document)를 의미함
    - 데이터의 저장이나 변환 등과 같이 실질적인 데이터 처리를 담당
  - CImageProView 클래스
    - CImageProDoc 클래스에서 처리된 데이터를 출력 장치를 통하여 보여주는 역할 수행
  - CMainFrame 클래스
    - 프로그램 전체 윈도우에 대한 관리 담당

기본 클래스

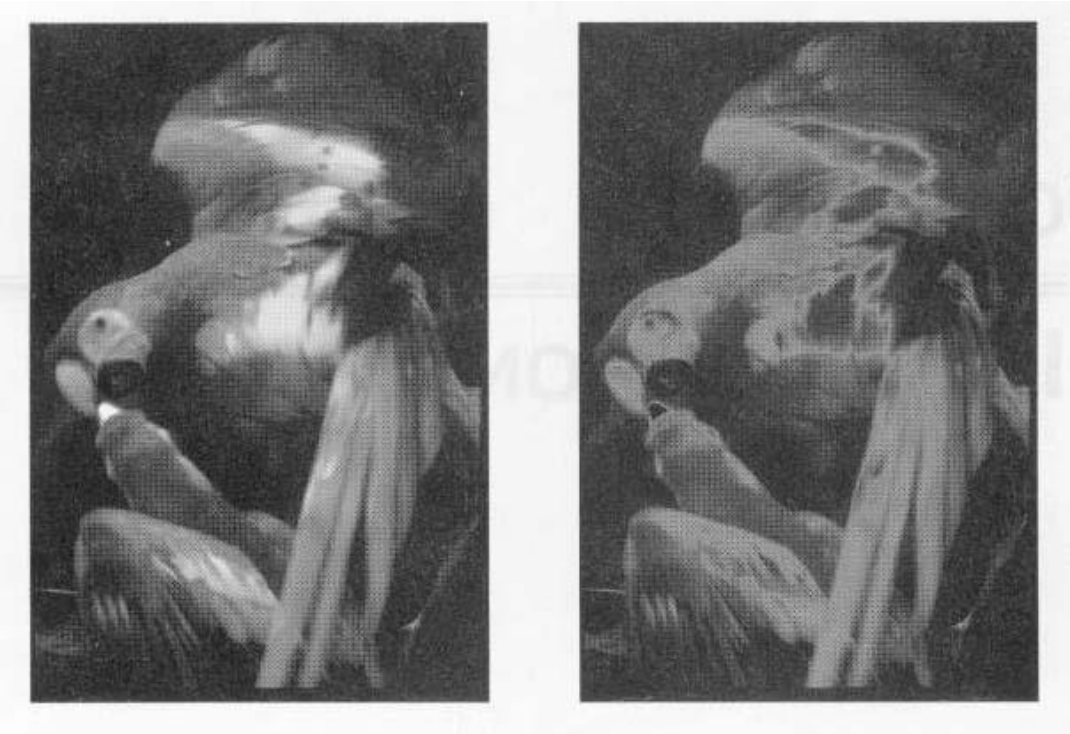
- CChildFrame 클래스
  - 프로그램 윈도우 안에 생성되는 여러 개의 서브윈도우에 대한 관리 담당
- CAboutDlg 클래스
  - 프로그램의 [도움말] 메뉴의 "ImagePro 정보"라는 항목을 선택하면 나타나는 대화상자에 대한 관리 담당
- CImageProApp 클래스
  - ImagePro 프로그램 전체에 대한 관리를 담당

기본적인 영상처리 알고리즘의 분류

- 픽셀 기반 처리
  - 단일 영상
  - 여러 영상
- 영역 기반 처리
- 기하학적 처리

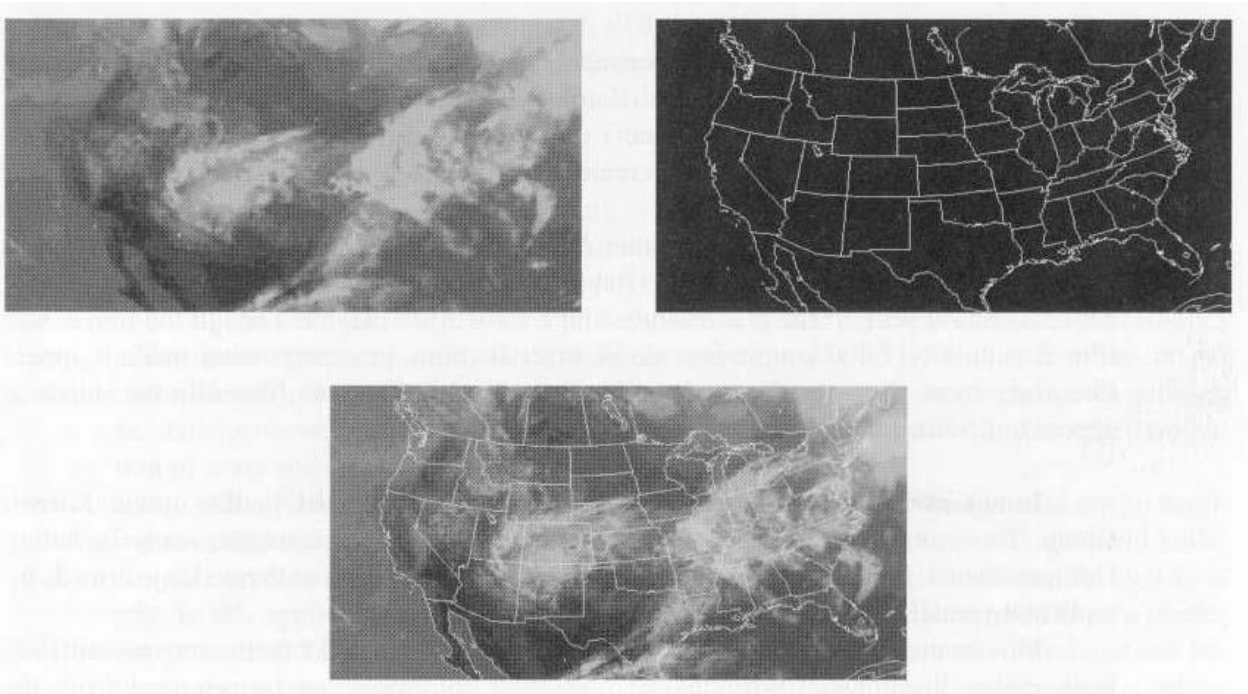
단일 영상에서의 픽셀 기반 처리

- 픽셀의 원래 값이나 위치에 기반한 픽셀 값을 변경

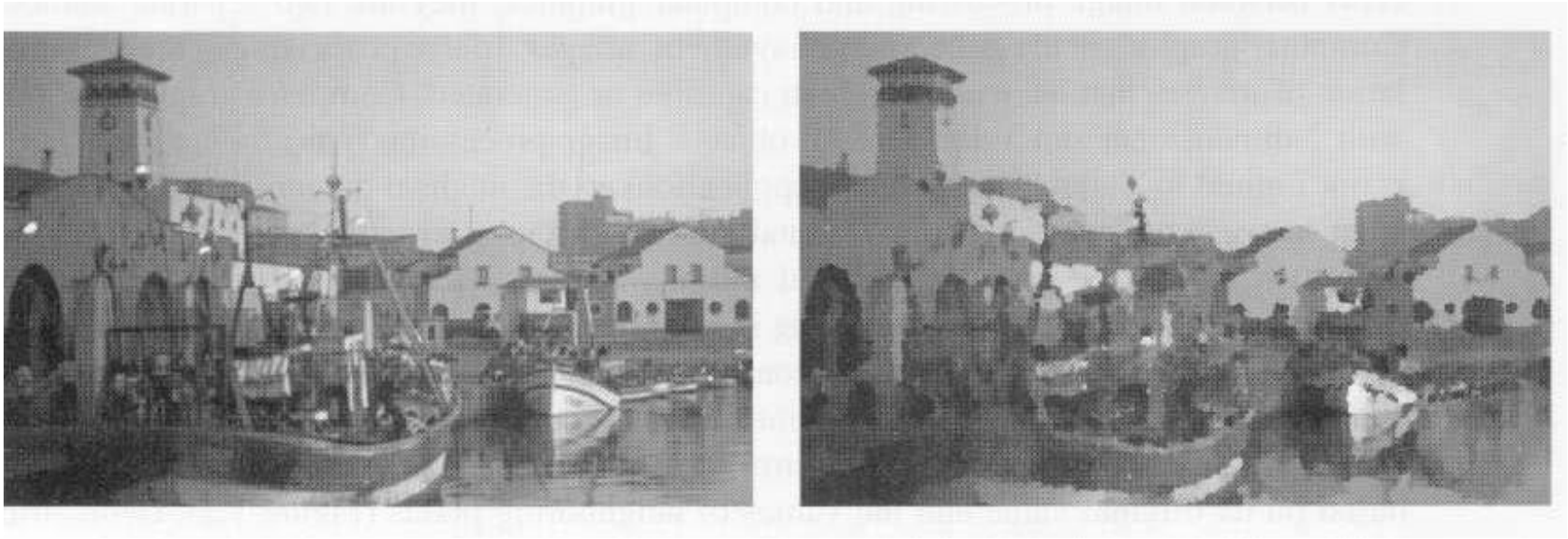


여러 영상을 이용한 픽셀 기반 처리

- **두 개 이상의 영상**들에 대한 연산을 기반으로 하여 픽셀 값들을 생성함



- 픽셀의 원래 값과 이웃하는 픽셀의 값을 기반으로 하여 픽셀 값을 변경



- 픽셀의 위치나 배열을 변화시킴