

쉽게 배우는 MFC 윈도우 프로그래밍

Visual C++ 2019

2015·2017 버전 사용 가능



1장. 윈도우 프로그래밍 기초

목차

- 01 윈도우 프로그래밍 개요
- 02 SDK 프로그램 기본 구조
- 03 MFC 프로그램 기본 구조
- 04 비주얼 C++ 개발 환경

윈도우 운영체제의 특징

■ 그래픽 사용자 인터페이스(GUI)

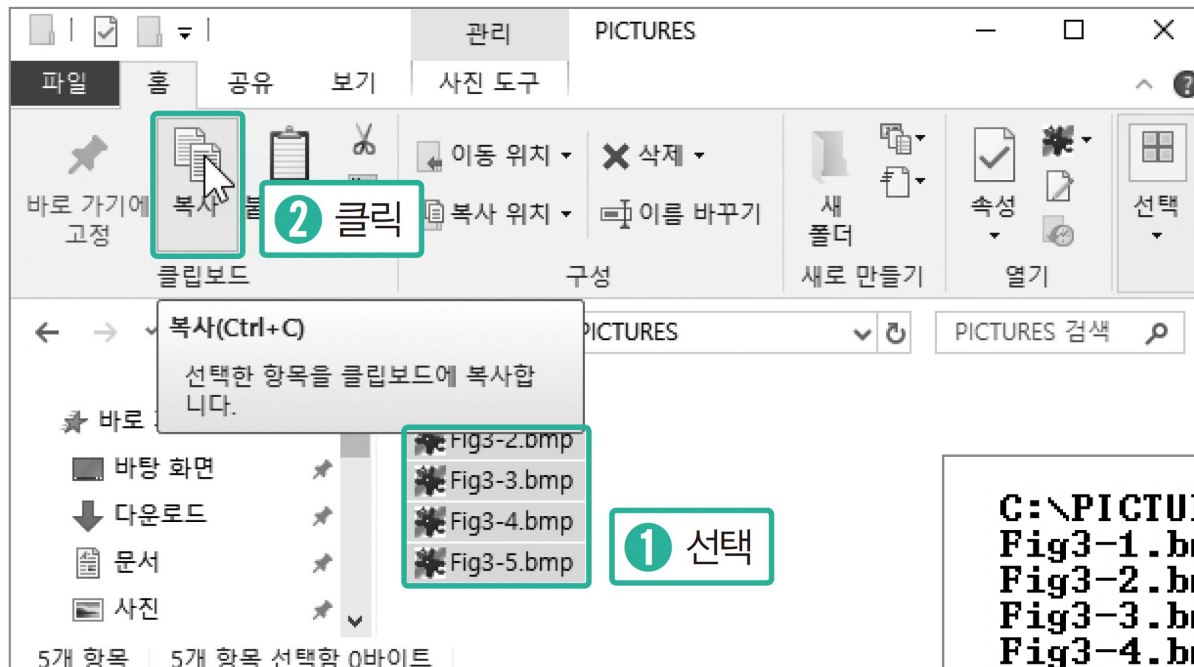


그림 1-2 그래픽 기반 운영체제 : 윈도우

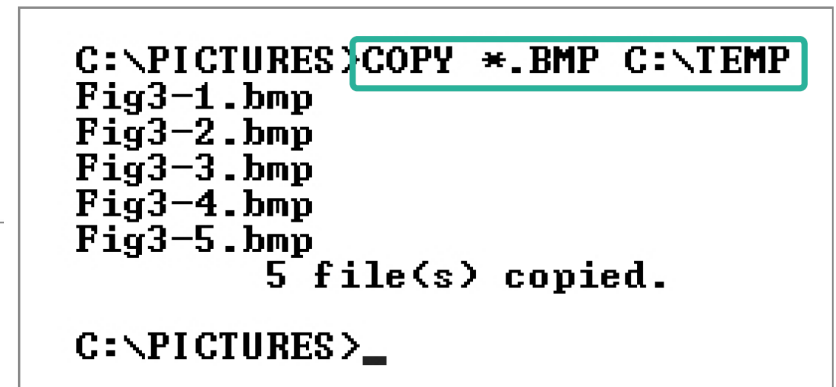


그림 1-1 텍스트 기반 운영체제 : 도스

윈도우 운영체제의 특징

■ 사용자 인터페이스의 구성 요소

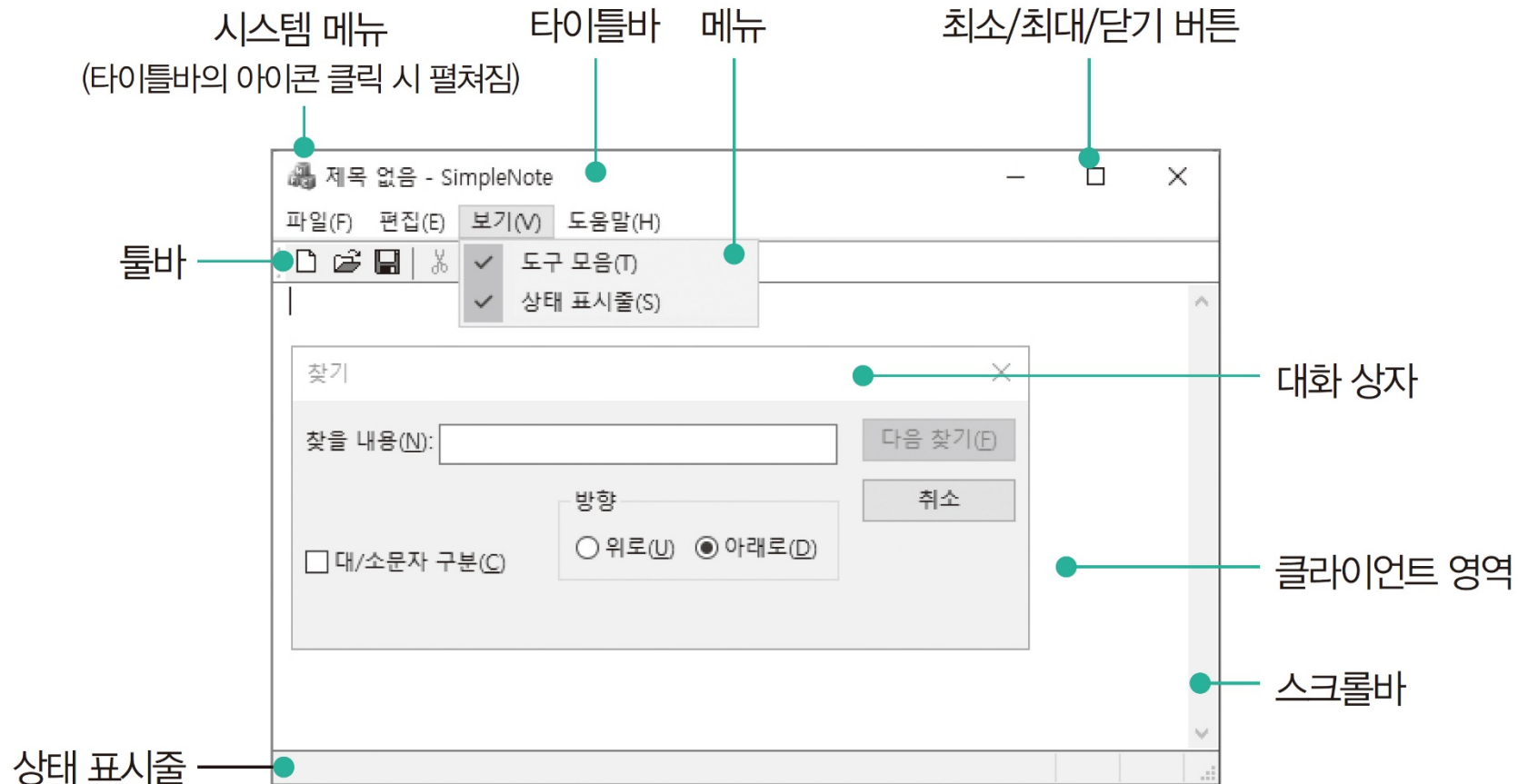


그림 1-3 사용자 인터페이스의 구성 요소

윈도우 운영체제의 특징

■ 메시지 구동 구조



그림 1-4 메시지 구동 구조

윈도우 운영체제의 특징

■ 멀티태스킹과 멀티스레딩

- 멀티태스킹(Multitasking) : 운영체제가 여러 개의 응용 프로그램을 동시에 실행
- 멀티스레딩(Multithreading) : 응용 프로그램 내부에서 여러 개의 실행 흐름(=스레드)을 동시에 진행

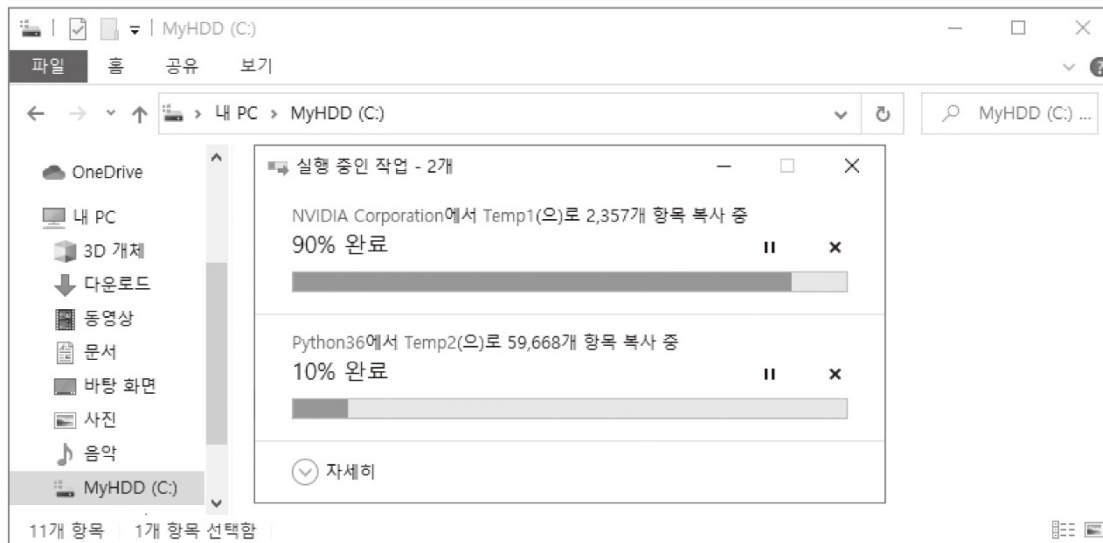


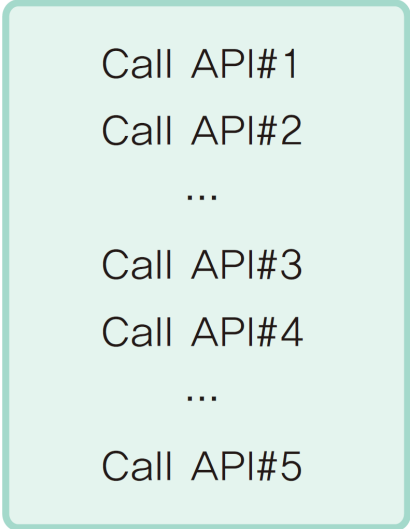
그림 1-5 멀티스레딩의 예

윈도우 응용 프로그램의 특징

■ API 호출문 집합

- API : Application Programming Interface
- 윈도우 운영체제가 응용 프로그램을 위해 제공하는 각종 함수의 집합
 - 16비트 윈도우 → Win16 API
 - 32 또는 64비트 윈도우 → Win32 API

응용 프로그램



```
Call API#1  
Call API#2  
...  
Call API#3  
Call API#4  
...  
Call API#5
```

윈도우 응용 프로그램의 특징

■ 메시지 핸들러 집합

- 메시지 핸들러 : 메시지를 받았을 때 동작을 결정하는 코드
- 윈도우 프로시저 : 메시지 핸들러의 집합

응용 프로그램

메시지 핸들러 #1
메시지 핸들러 #2
메시지 핸들러 #3
메시지 핸들러 #4
메시지 핸들러 #5
메시지 핸들러 #6
...

그림 1-7 메시지 핸들러 집합

윈도우 응용 프로그램의 특징

■ 실행 파일과 DLL 집합

- DLL(Dynamic-Link Library) : 프로그램 실행 중에 결합하여 사용할 수 있는 코드와 리소스 집합
- 윈도우 운영체제가 제공하는 API는 DLL 형태로 제공되며, 프로그래머가 직접 필요한 기능을 DLL로 제작하기도 함

응용 프로그램

실행 파일

DLL #1

DLL #2

DLL #3

DLL #4

DLL #5

...

윈도우 응용 프로그램의 특징

■ 장치 독립성(Device-Independency)

- 주변 장치가 바뀌어도 장치 드라이버(Device Driver)만 설치하면 프로그램을 수정하지 않고 실행할 수 있음
- 응용 프로그램은 API를 통해 장치 드라이버와 간접적으로 통신하므로 장치 독립성을 가짐

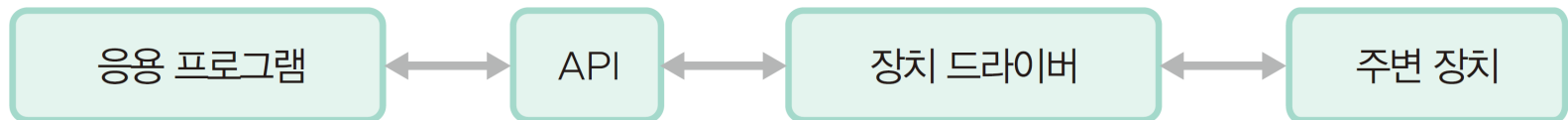


그림 1-9 장치 독립적으로 동작하는 응용 프로그램

윈도우 응용 프로그램의 개발 방식

■ SDK(Software Development Kit)

- MS에서 윈도우 응용 프로그램 제작을 위해 배포하는 개발 도구 모음
- 컴파일러를 비롯한 각종 개발툴, 헤더 파일, 라이브러리 파일, 도움말 등이 포함됨
- SDK 프로그램 개발은 C/C++ 언어로 윈도우 API를 직접 호출해서 프로그램을 구현한다는 것
- 장점
 - API를 직접 다루기 때문에 세부 제어가 가능함
 - 윈도우 운영체제가 제공하는 모든 기능을 사용 가능
 - 생성 코드의 크기가 작고 속도도 빠름
- 단점
 - 다른 개발 방식에 비해 생산성이 매우 낮음

윈도우 응용 프로그램의 개발 방식

■ RAD(Rapid Application Development)

- 화면을 시각적으로 디자인하고 코드를 추가해 프로그램을 개발
- 장점
 - 직관적으로 간편하게 프로그래밍할 수 있음
 - 빠른 시간 내에 원하는 기능의 프로그램 개발 가능
- 단점
 - SDK나 클래스 라이브러리를 이용한 개발 방식보다 생성 코드의 크기가 크고 실행 속도도 떨어지는 편임
 - 윈도우 운영체제가 제공하는 모든 기능을 활용해 세부적으로 제어하기 어려운 경우도 있음

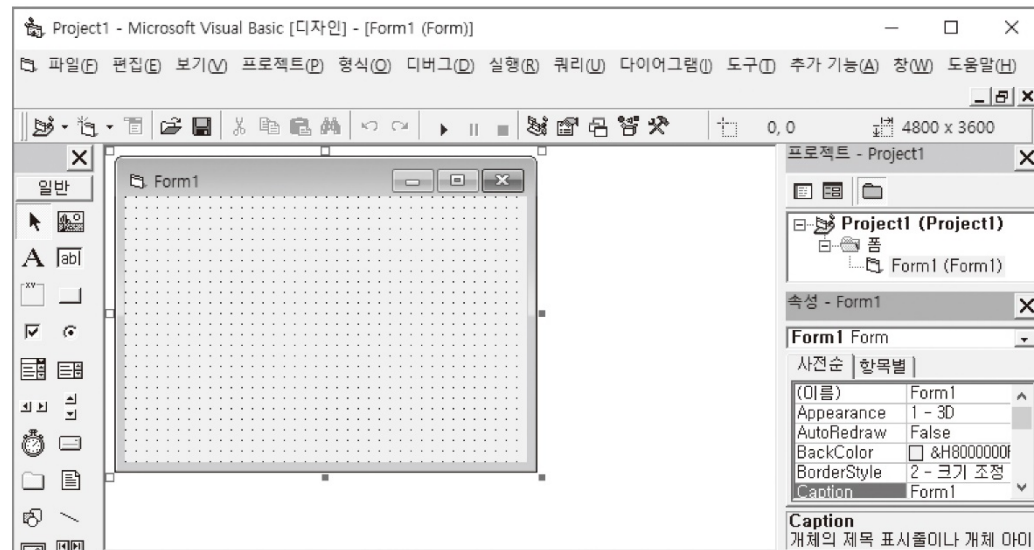


그림 1-10 RAD 개발툴의 예 : 비주얼 베이직

윈도우 응용 프로그램의 개발 방식

■ 클래스 라이브러리(Class Library)

- 윈도우 응용 프로그램 개발에 필수적인 기능을 C++와 같은 객체지향 언어를 이용하여 클래스화한 것(MFC, OWL)
- 장점
 - SDK를 이용한 방식보다 빠른 속도로 개발
 - API를 직접 사용해서 세부적으로 제어할 수 있음
 - RAD 개발 방식보다 코드 크기와 실행 속도 면에서 유리함
- 단점
 - 객체지향 프로그래밍에 익숙해야 함
 - 클래스 라이브러리의 구조와 각 클래스의 기능 및 관계를 파악하기 위한 초기 학습 기간이 긴 편임

윈도우 응용 프로그램의 개발 방식

■ .NET 프레임워크

- 윈도우 운영체제에 설치할 수 있는 소프트웨어 개발 및 실행 환경
- 특징
 - 공용 언어 런타임(CLR, Common Language Runtime)이라는 소프트웨어 가상 머신을 제공하며, 가상 머신 제어 하에서 응용 프로그램이 구동됨(장치 독립성)
 - 윈도우 API에 버금가는 방대한 라이브러리를 제공, 언어에 상관없이 라이브러리를 사용 가능(언어 독립성)

간단한 SDK 프로그램 작성 - HelloSDK

■ SDK 프로그램 기본 골격

- ① 윈도우 클래스를 정의(초기화)하고 운영체제에 등록
- ② 윈도우를 생성하고 화면에 나타냄
- ③ 메시지 루프를 구동함
- ④ 윈도우 프로시저에서 메시지를 처리함

간단한 SDK 프로그램 작성 - HelloSDK

■ 프로젝트 종류 선택



‘Windows 데스크톱 마법사’ 선택

그림 1-11 프로젝트 종류 선택

간단한 SDK 프로그램 작성 - HelloSDK

■ 프로젝트 이름과 위치 지정

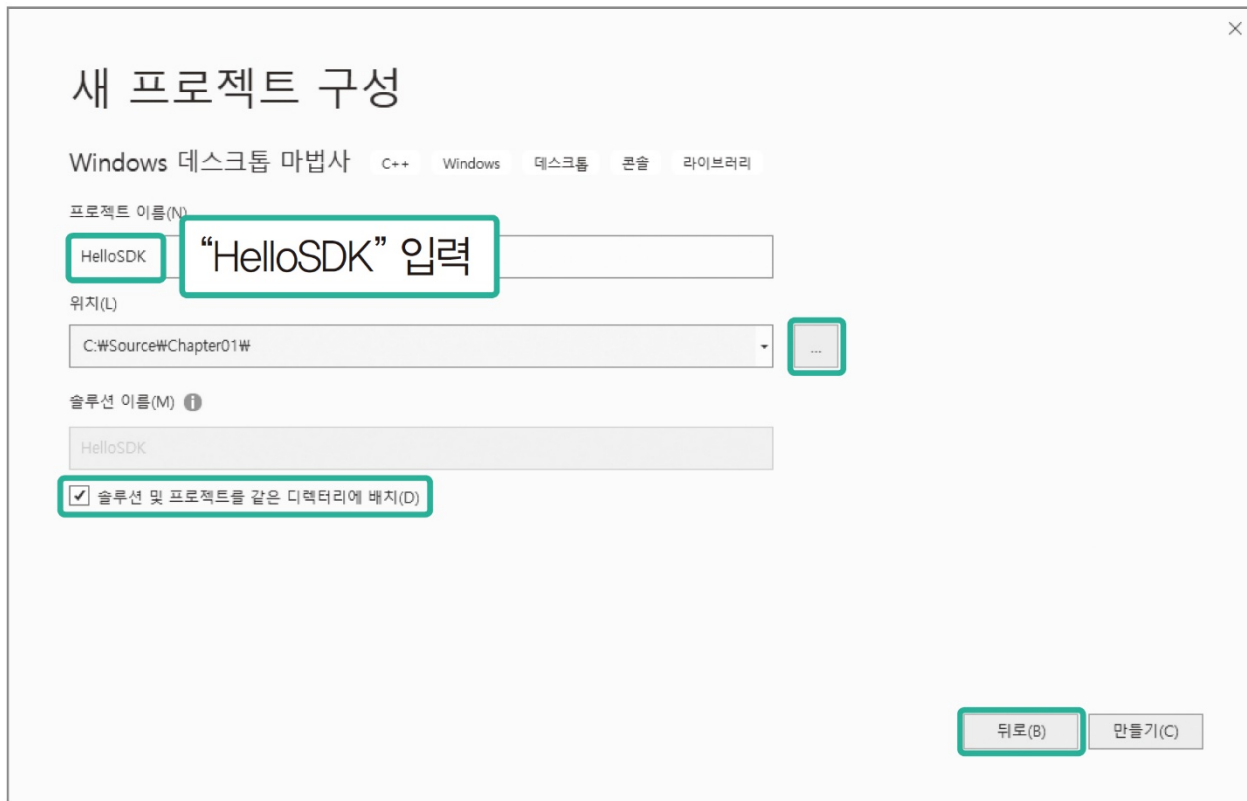


그림 1-12 프로젝트 이름과 위치 지정

간단한 SDK 프로그램 작성 - HelloSDK

■ 프로젝트 옵션 변경

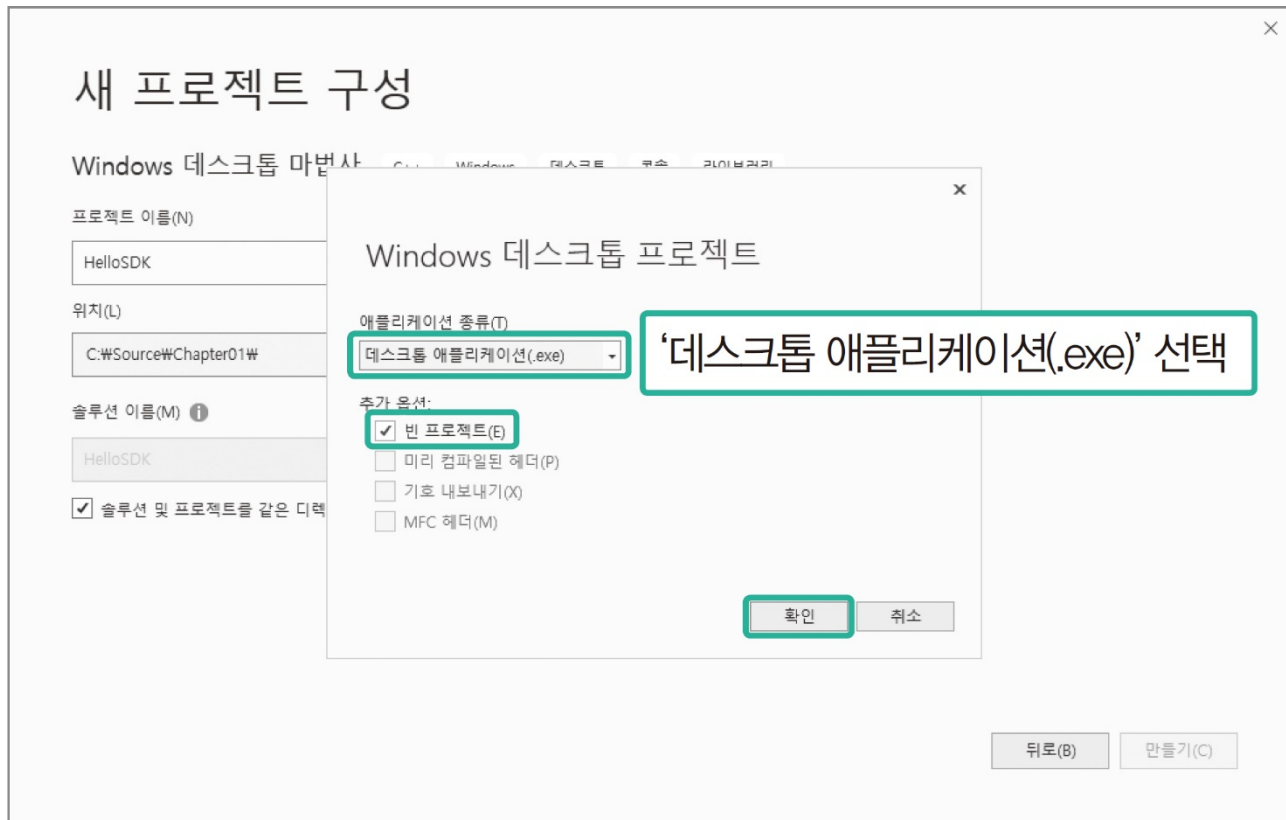


그림 1-13 프로젝트 옵션 변경

간단한 SDK 프로그램 작성 - HelloSDK

■ 소스 파일 추가(1)

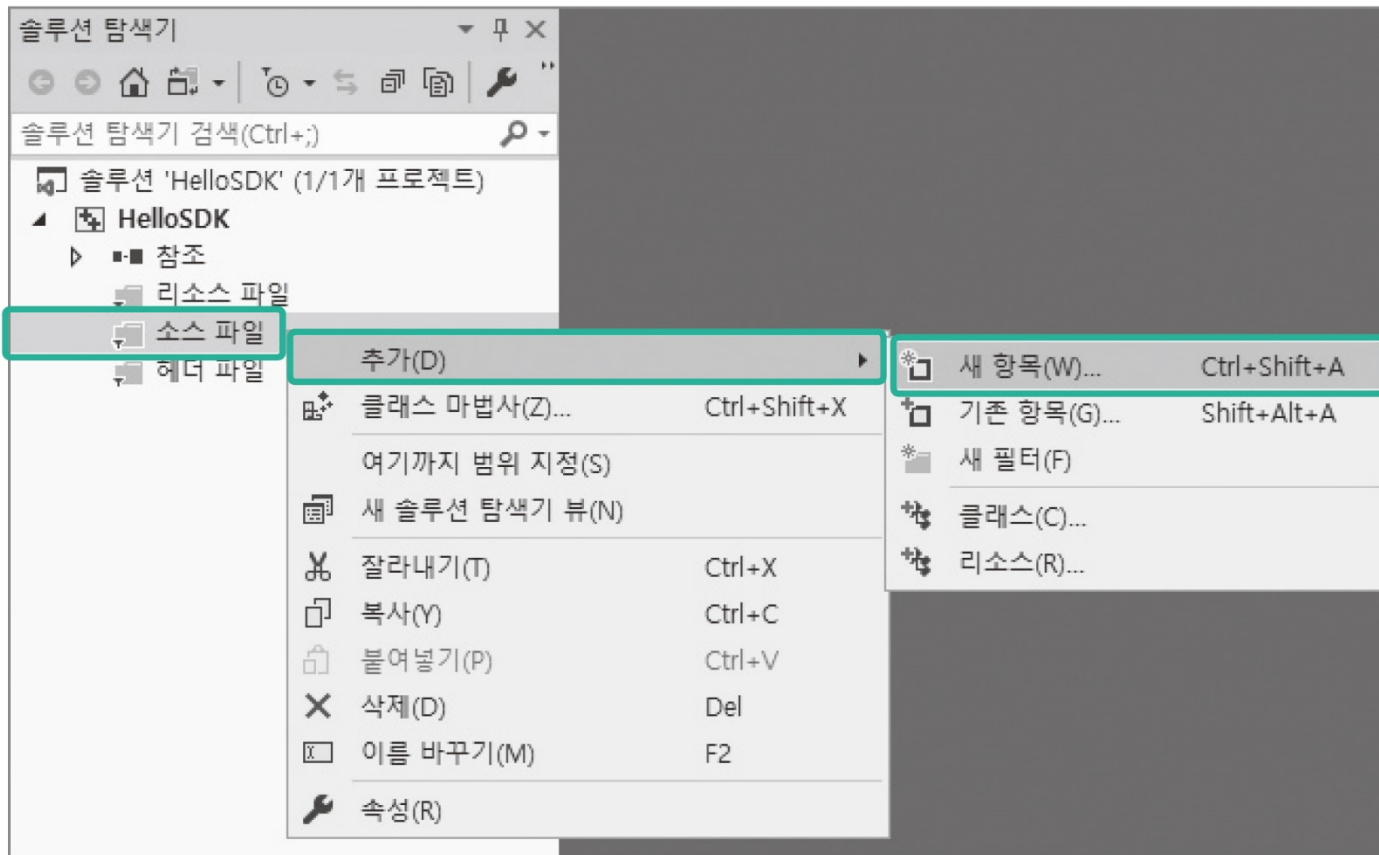


그림 1-14 소스 파일 추가(1)

간단한 SDK 프로그램 작성 - HelloSDK

■ 소스 파일 추가(2)

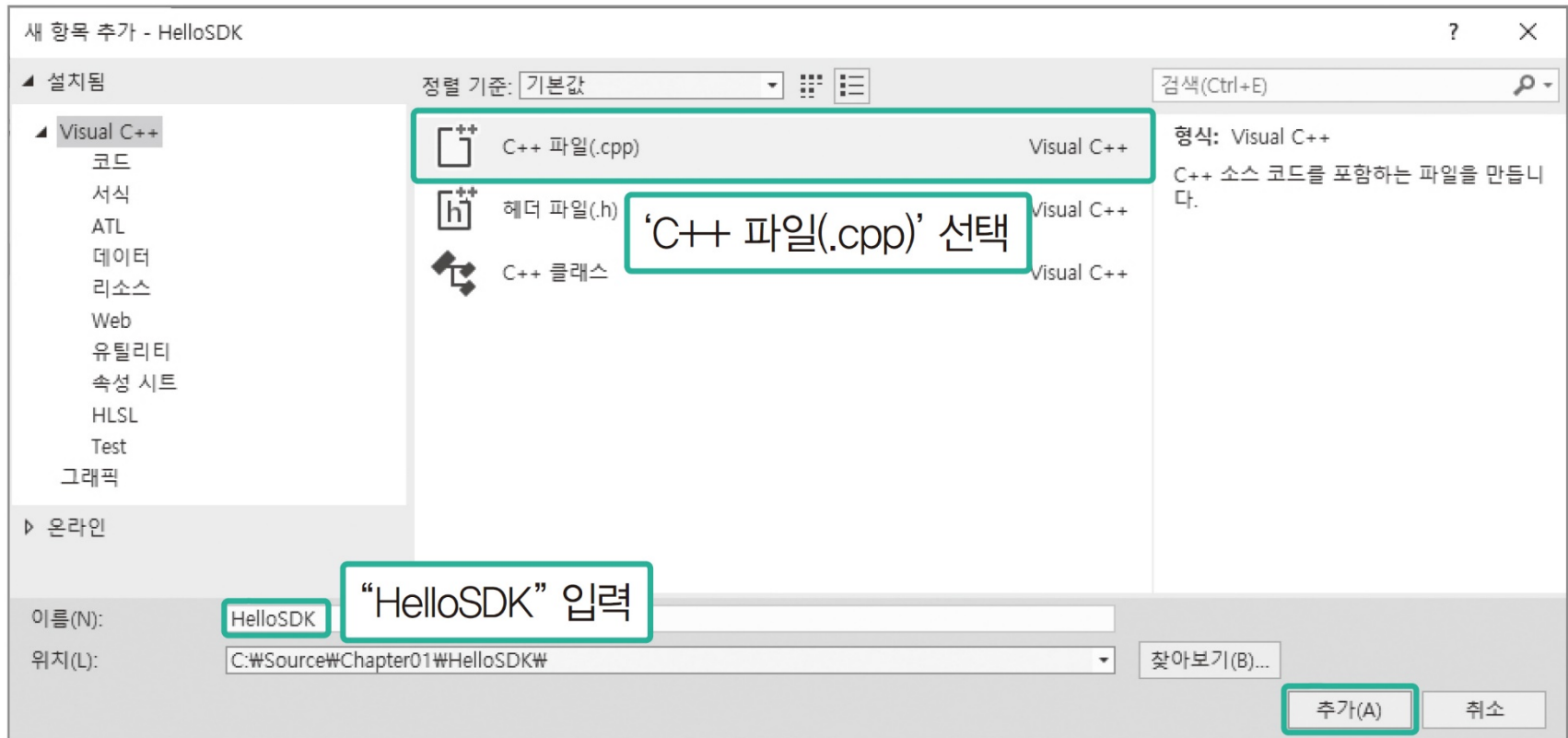


그림 1-15 소스 파일 추가(2)

간단한 SDK 프로그램 작성 - HelloSDK

■ 실행 결과

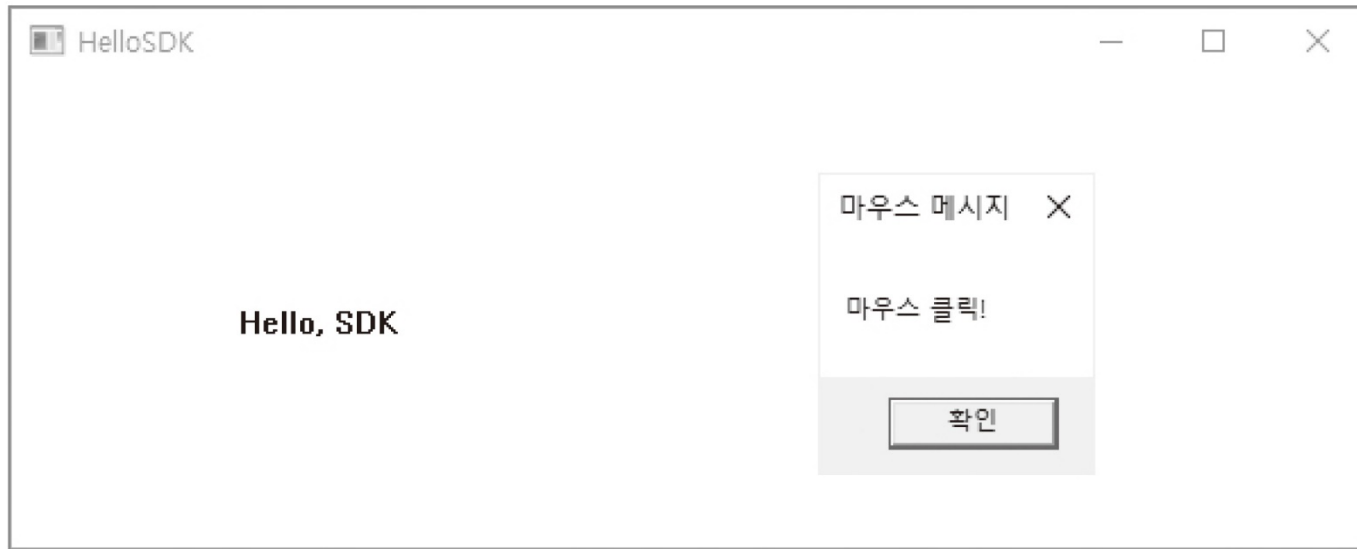


그림 1-16 실행 결과

간단한 SDK 프로그램 작성 - HelloSDK

■ HelloSDK 예제 코드 (1/5)

01 #include <windows.h>

02

03 // WinMain 함수에서 참조하므로 함수 원형을 선언한다.

04 LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

05

06 int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
07 LPSTR lpCmdLine, int nCmdShow)

08 {

09 WNDCLASS wndclass;

10 HWND hwnd;

11 MSG msg;

12

① 헤더 파일

② 메인 함수

간단한 SDK 프로그램 작성 - HelloSDK

■ HelloSDK 예제 코드 (2/5)

```
13 // 윈도우 클래스를 초기화하고 운영체제에 등록한다.      ③ 윈도우 클래스 초기화와 등록
14 wndclass.style = CS_HREDRAW | CS_VREDRAW; // 스타일 지정
15 wndclass.lpfnWndProc = WndProc; // 윈도우 프로시저 이름
16 wndclass.cbClsExtra = 0; // 여분 메모리(0바이트)
17 wndclass.cbWndExtra = 0; // 여분 메모리(0바이트)
18 wndclass.hInstance = hInstance; // 인스턴스 핸들
19 wndclass.hIcon = LoadIcon(NULL, IDI_APPLICATION); // 아이콘 모양
20 wndclass.hCursor = LoadCursor(NULL, IDC_ARROW); // 커서 모양
21 wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH); // 배경(흰색)
22 wndclass.lpszMenuName = NULL; // 메뉴(NULL->메뉴 없음)
23 wndclass.lpszClassName = TEXT("HelloClass"); // 윈도우 클래스 이름
24 if(!RegisterClass(&wndclass)) return 1;
25
```

간단한 SDK 프로그램 작성 - HelloSDK

■ HelloSDK 예제 코드 (3/5)

```
26 // 윈도우를 생성하고 화면에 나타낸다.
27 hwnd = CreateWindow(TEXT("HelloClass"), TEXT("HelloSDK"),
28     WS_OVERLAPPEDWINDOW, CW_USEDEFAULT, CW_USEDEFAULT,
29     CW_USEDEFAULT, CW_USEDEFAULT, NULL, NULL, hInstance, NULL);
30 ShowWindow(hwnd, nCmdShow);
31
32 // 메시지 큐에서 메시지를 하나씩 꺼내서 처리한다.
33 while(GetMessage(&msg, NULL, 0, 0) > 0){
34     TranslateMessage(&msg);
35     DispatchMessage(&msg);
36 }
37
38 return msg.wParam;
39 }
40
```

④ 윈도우 생성

⑤ 메시지 루프

간단한 SDK 프로그램 작성 - HelloSDK

■ HelloSDK 예제 코드 (4/5)

```
41 LRESULT CALLBACK WndProc(HWND hwnd, UINT message,
42                             WPARAM wParam, LPARAM lParam)
43 {
44     HDC hdc;
45     PAINTSTRUCT ps;
46     const TCHAR *str = TEXT("Hello, SDK");
47
48     // 발생한 메시지의 종류에 따라 적절히 처리한다.
49     switch(message){
50     case WM_CREATE:
51         return 0;
52     case WM_LBUTTONDOWN:
```

⑥ 윈도우 프로시저

간단한 SDK 프로그램 작성 - HelloSDK

■ HelloSDK 예제 코드 (5/5)

```
53     MessageBox(hwnd, TEXT("마우스 클릭!"), TEXT("마우스 메시지"), MB_OK);
54     return 0;
55 case WM_PAINT:
56     hdc = BeginPaint(hwnd, &ps);
57     TextOut(hdc, 100, 100, str, strlen(str));
58     EndPaint(hwnd, &ps);
59     return 0;
60 case WM_DESTROY:
61     PostQuitMessage(0);
62     return 0;
63 }
64
65 // 응용 프로그램이 처리하지 않은 메시지는 운영체제가 처리한다.
66 return DefWindowProc(hwnd, message, wParam, lParam);
67 }
```

간단한 SDK 프로그램 작성 - HelloSDK

■ HelloSDK 예제 코드 실행 결과

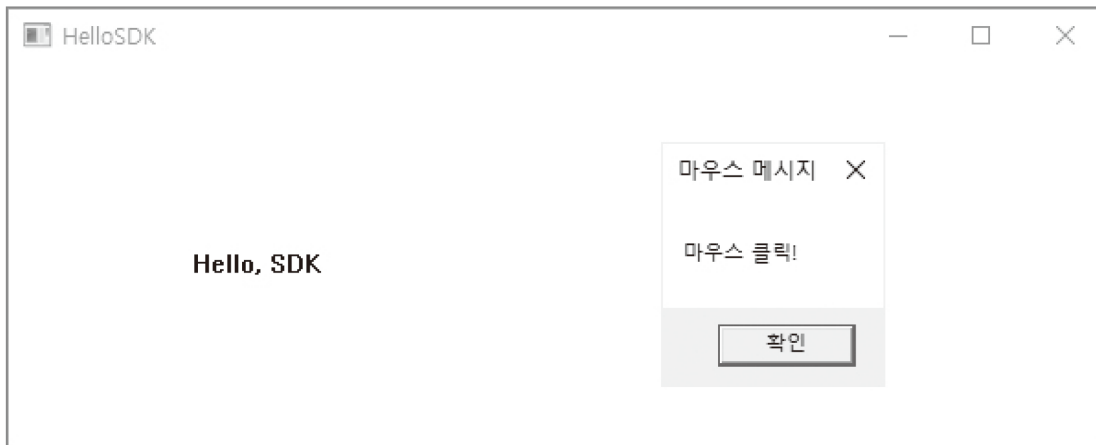


그림 1-16 실행 결과

SDK 프로그램 기본 구조

■ SDK 프로그램 동작 원리

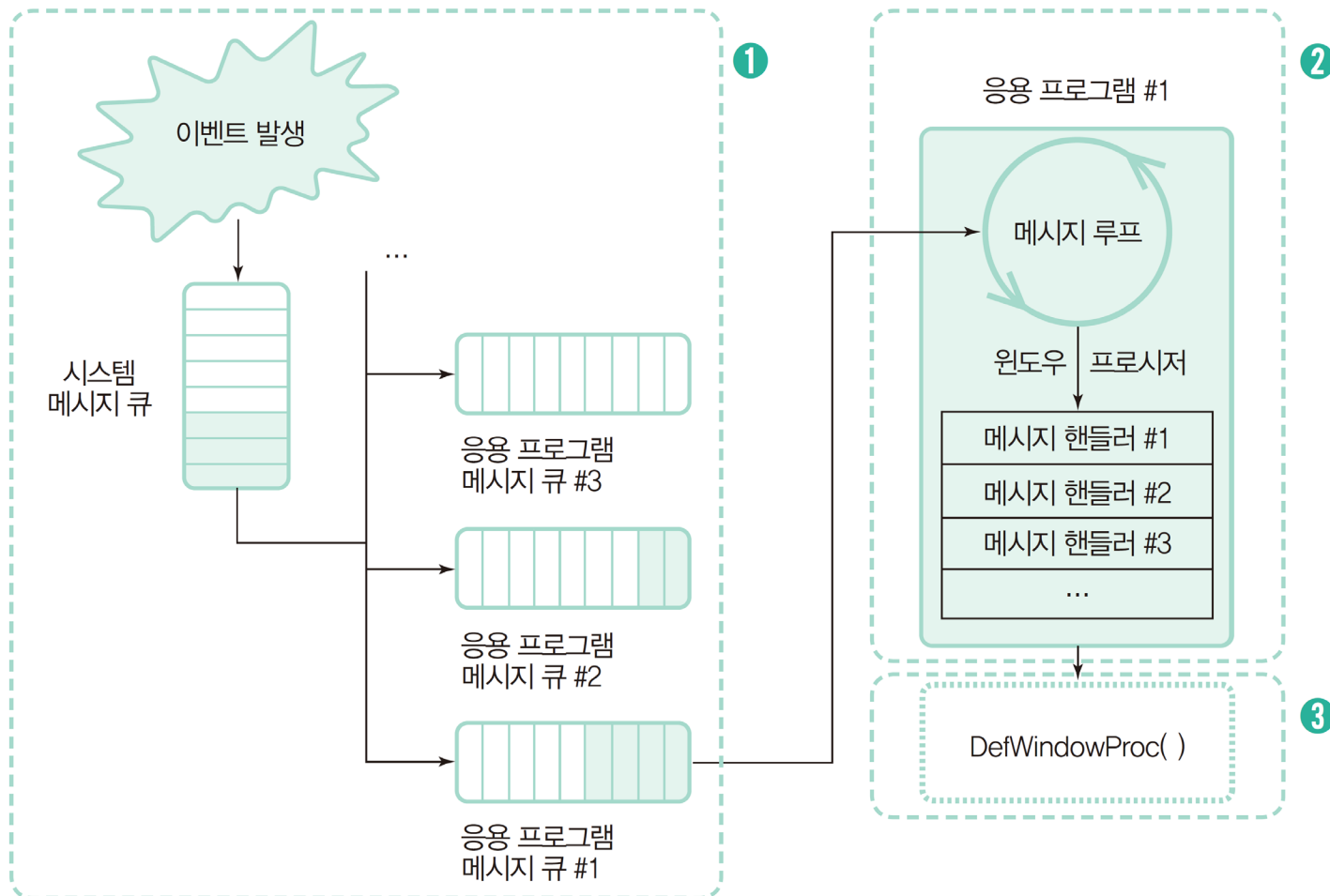


그림 1-17 SDK 응용 프로그램의 동작 원리

MFC 프로그램 기본 구조

■ MFC 프로그램 기본 골격

- ① 응용 프로그램 클래스 정의
- ② 메인(=프레임) 윈도우 클래스 정의
- ③ 응용 프로그램 객체 선언
- ④ 메시지 맵 선언

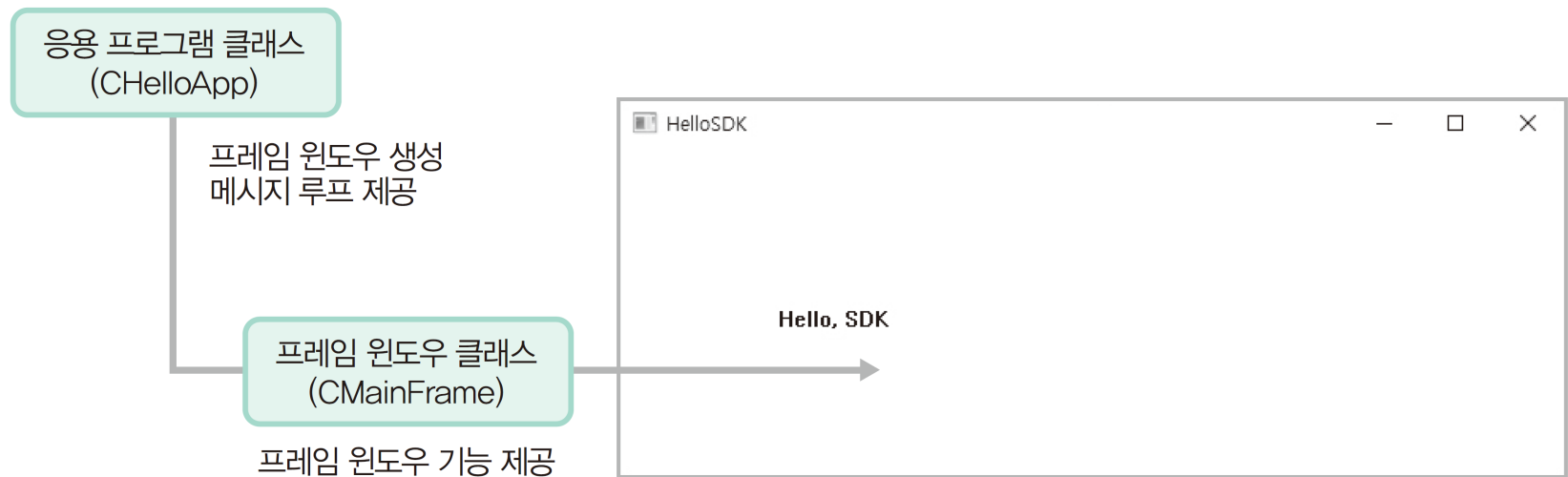


그림 1-18 주요 클래스

HelloMFC 프로그램 작성하기

■ 프로젝트 종류 선택



그림 1-19 프로젝트 종류 선택

HelloMFC 프로그램 작성하기

■ 프로젝트 이름과 위치 지정



그림 1-20 프로젝트 이름과 위치 지정

HelloMFC 프로그램 작성하기

■ 프로젝트 옵션 변경

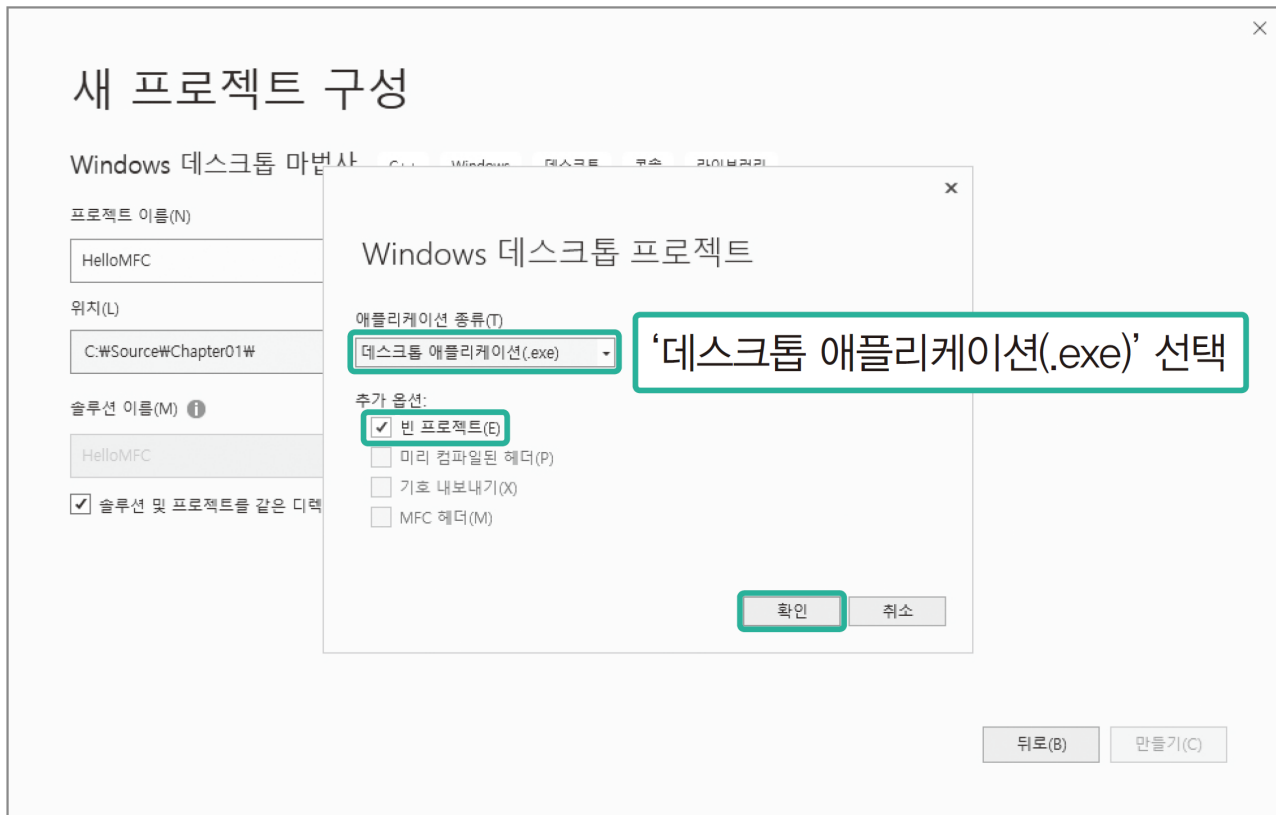


그림 1-21 프로젝트 옵션 변경

HelloMFC 프로그램 작성하기

■ 소스 파일 추가(1)

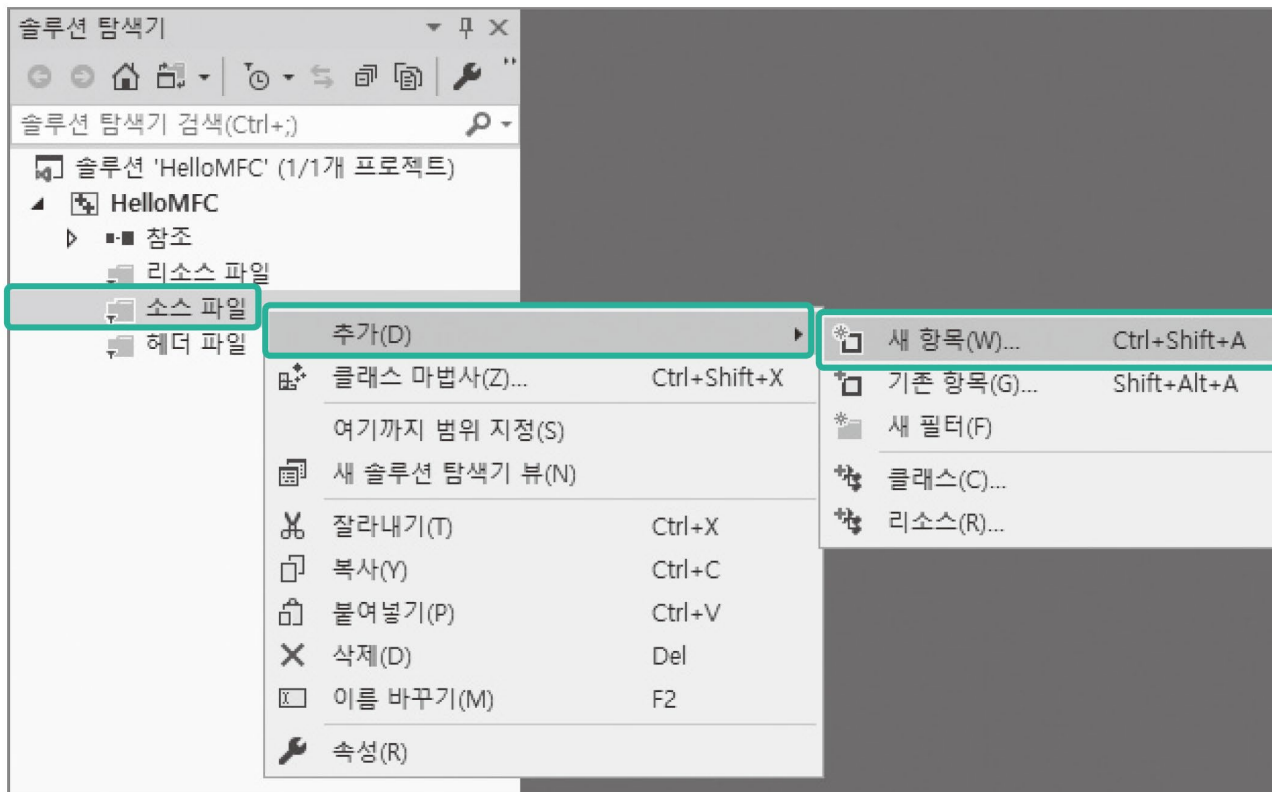


그림 1-22 소스 파일 추가(1)

HelloMFC 프로그램 작성하기

■ 소스 파일 추가(2)

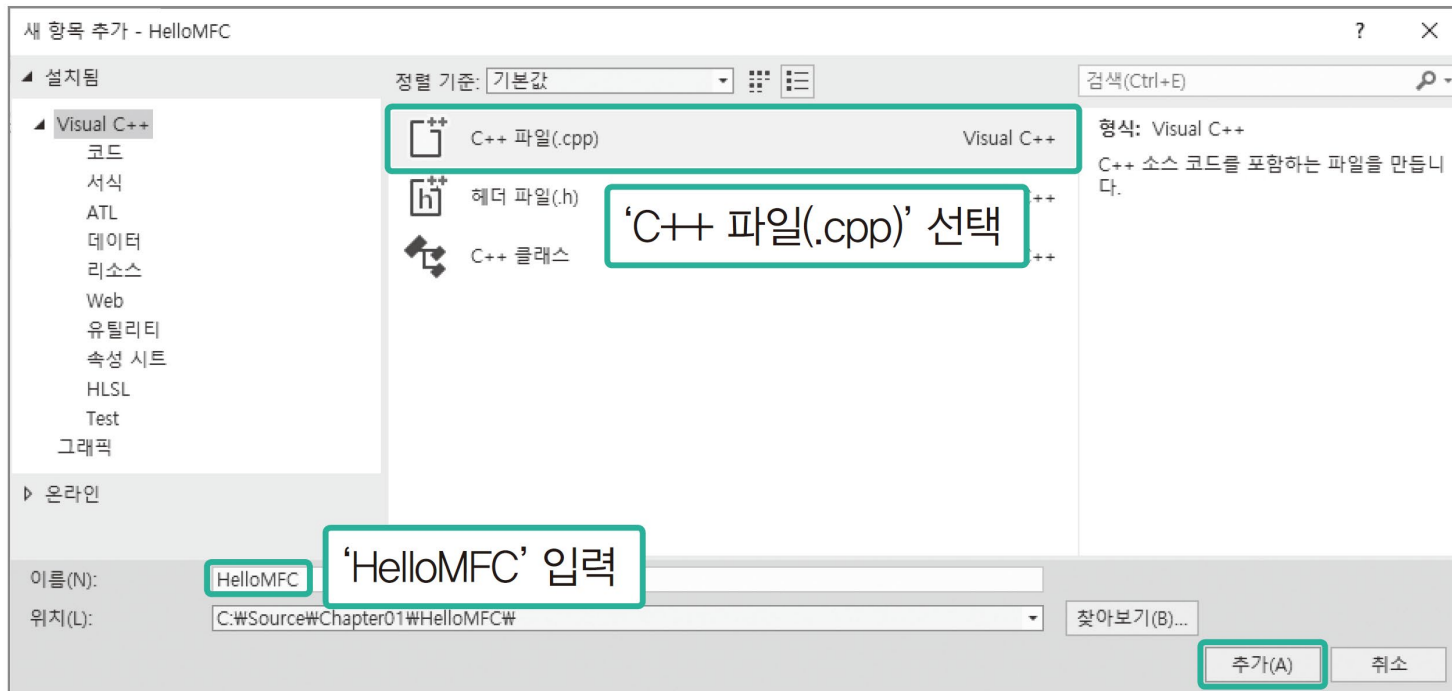


그림 1-23 소스 파일 추가(2)

HelloMFC 프로그램 작성하기

■ HelloMFC 예제 코드 (1/4)

```
01 #include <afxwin.h>
02
03 // 응용 프로그램 클래스를 선언한다.
04 class CHelloApp : public CWinApp
05 {
06 public:
07     virtual BOOL InitInstance();
08 };
09
10 // 메인 윈도우 클래스를 선언한다.
11 class CMainFrame : public CFrameWnd
12 {
13 public:
14     CMainFrame();
15
```

① 헤더 파일

② 클래스 선언부

HelloMFC 프로그램 작성하기

■ HelloMFC 예제 코드 (2/4)

```
16 protected:
17     afx_msg void OnPaint();
18     afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
19     DECLARE_MESSAGE_MAP()
20 };
21
22 // 응용 프로그램 객체를 선언한다.
23 CHelloApp theApp;
24
25 // 응용 프로그램 클래스를 정의한다.
26 BOOL CHelloApp::InitInstance()
27 {
28     m_pMainWnd = new CMainFrame;
29     m_pMainWnd->ShowWindow(m_nCmdShow);
30     return TRUE;
31 }
32
```

③ 응용 프로그램 객체

④ 클래스 정의부

HelloMFC 프로그램 작성하기

■ HelloMFC 예제 코드 (3/4)

```
33 // 메인 윈도우 클래스를 정의한다.
34 CMainFrame::CMainFrame()
35 {
36     Create(NULL, _T("HelloMFC"));
37 }
38
39 void CMainFrame::OnPaint()
40 {
41     CPaintDC dc(this);
42     const TCHAR *msg = _T("Hello, MFC");
43     dc.TextOut(100, 100, msg, lstrlen(msg));
44 }
45
46 void CMainFrame::OnLButtonDown(UINT nFlags, CPoint point)
47 {
48     MessageBox(_T("마우스 클릭!"), _T("마우스 메시지"));
49 }
50
```

HelloMFC 프로그램 작성하기

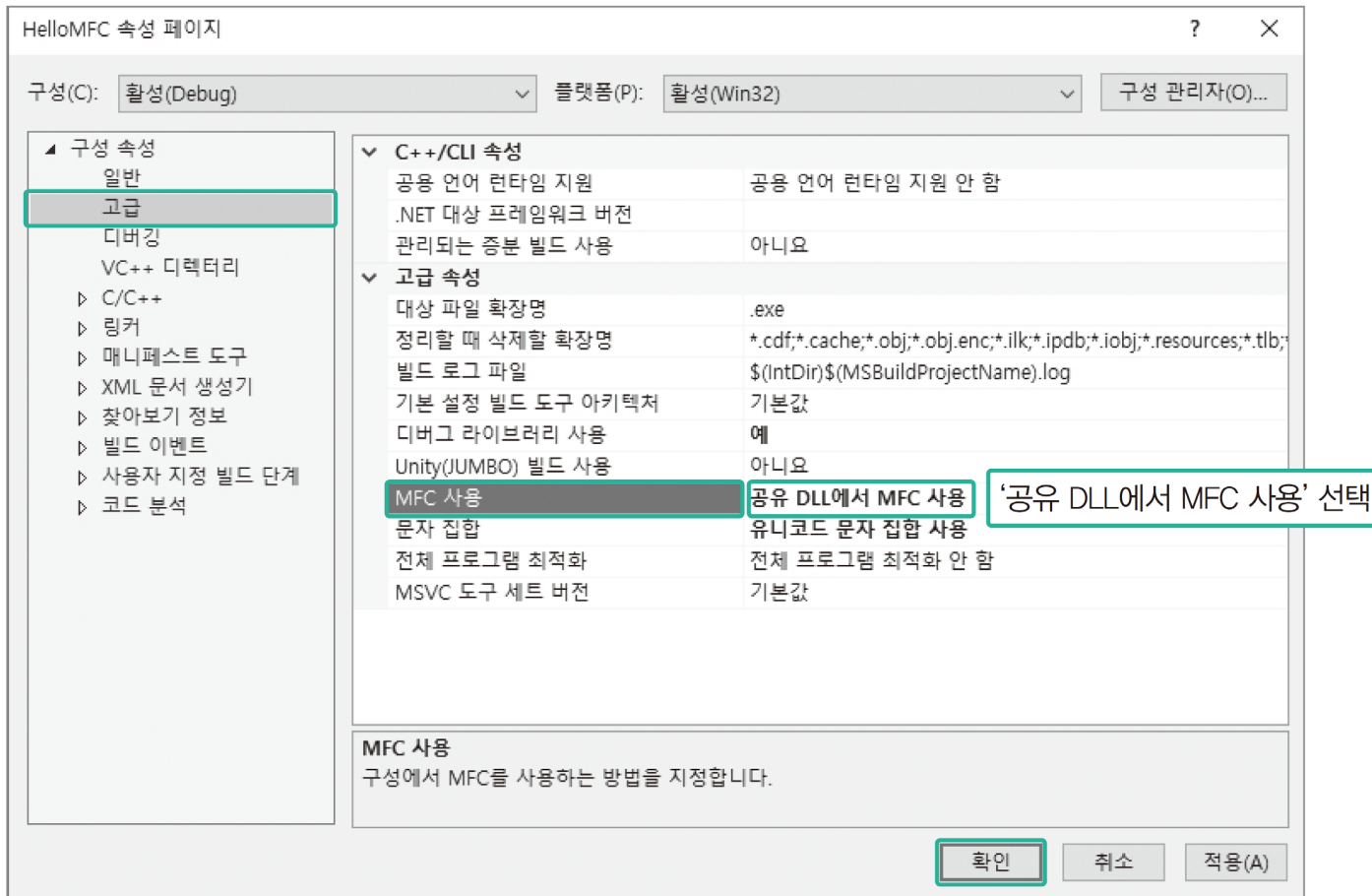
■ HelloMFC 예제 코드 (4/4)

```
51 // 메시지 맵을 선언한다.  
52 BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)  
53     ON_WM_PAINT()  
54     ON_WM_LBUTTONDOWN()  
55 END_MESSAGE_MAP()
```

⑤ 메시지 맵

HelloMFC 프로그램 작성하기

■ 프로젝트 속성 변경



HelloMFC 프로그램 작성하기

■ 실행 결과

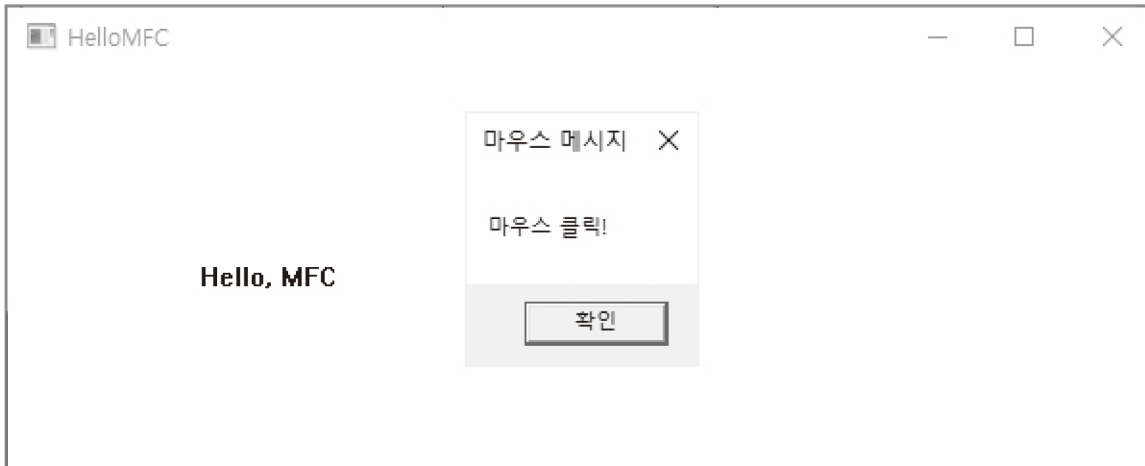


그림 1-25 실행 결과

MFC 프로그램 기본 구조

■ MFC 프로그램 동작 원리

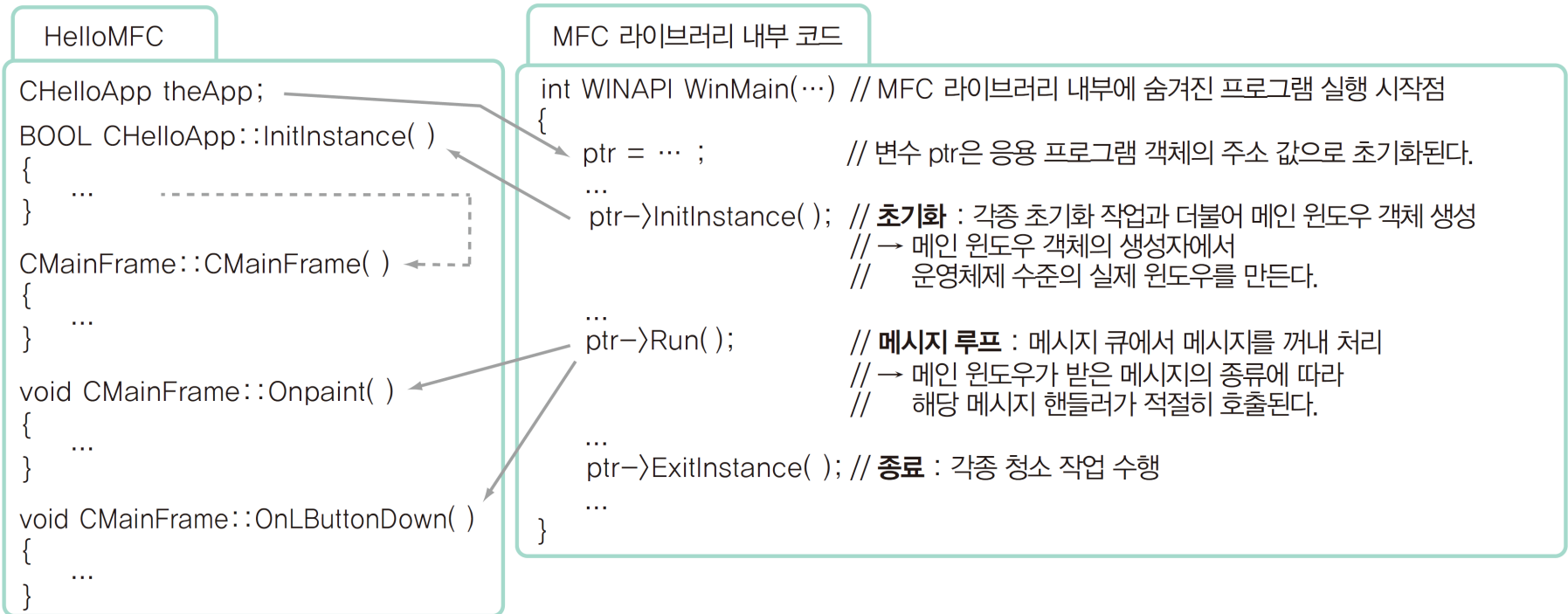


그림 1-27 MFC 응용 프로그램의 동작 원리

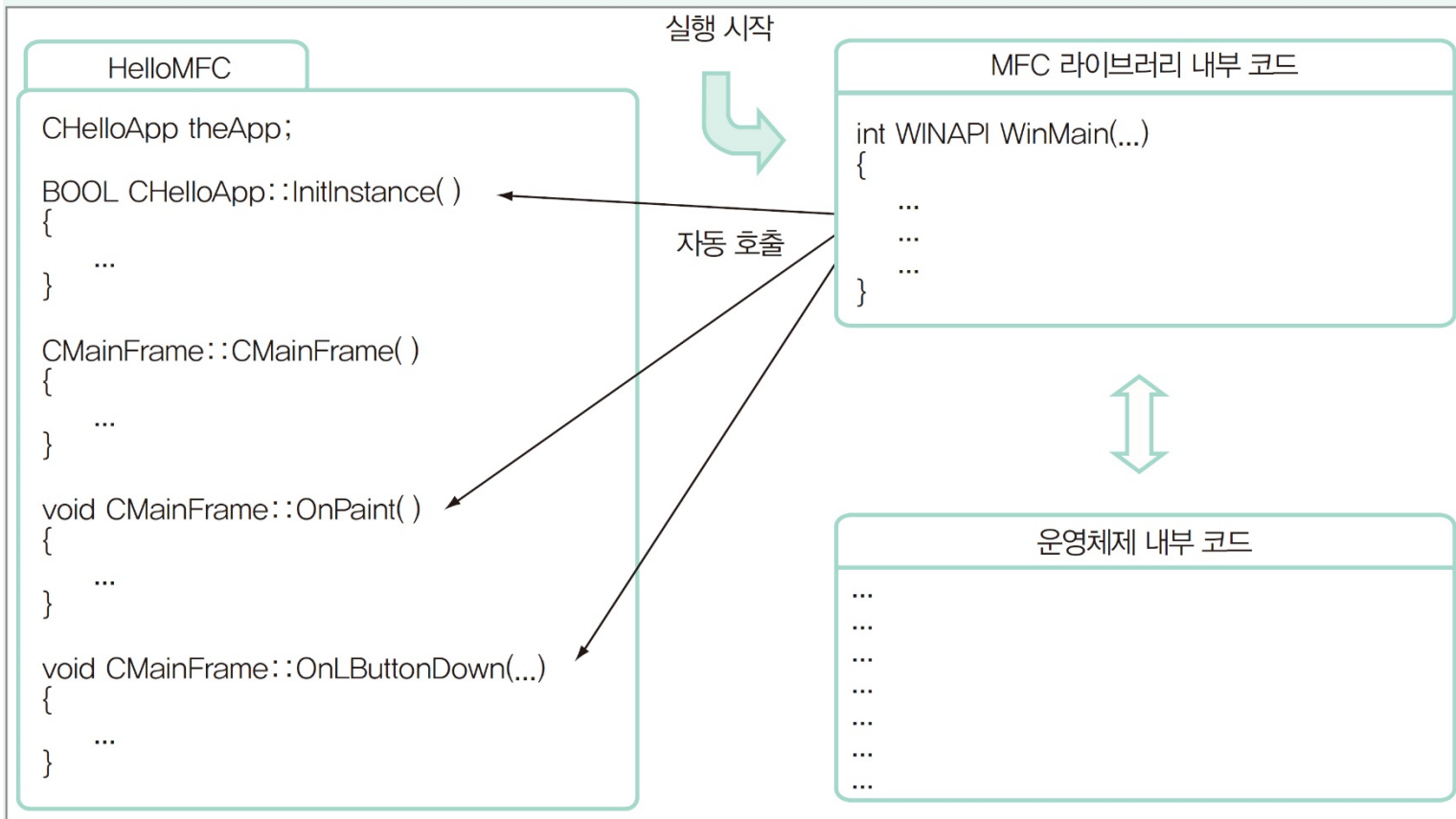
MFC 프로그램 기본 구조

■ MFC 프로그램의 특징

- WinMain() 함수가 존재하지 않음
- 사용자가 함수를 직접 호출하기보다 MFC 내부에 숨겨진 코드에서 사용자가 정의한 함수를 호출하는 경우가 많음(가상 함수)
- 각 메시지에 대한 처리 코드를 함수 단위(메시지 핸들러)로 따로 만들고, 해당 메시지와 메시지 핸들러를 연결하기 위해 메시지 맵 매크로를 사용

MFC 프로그램 기본 구조

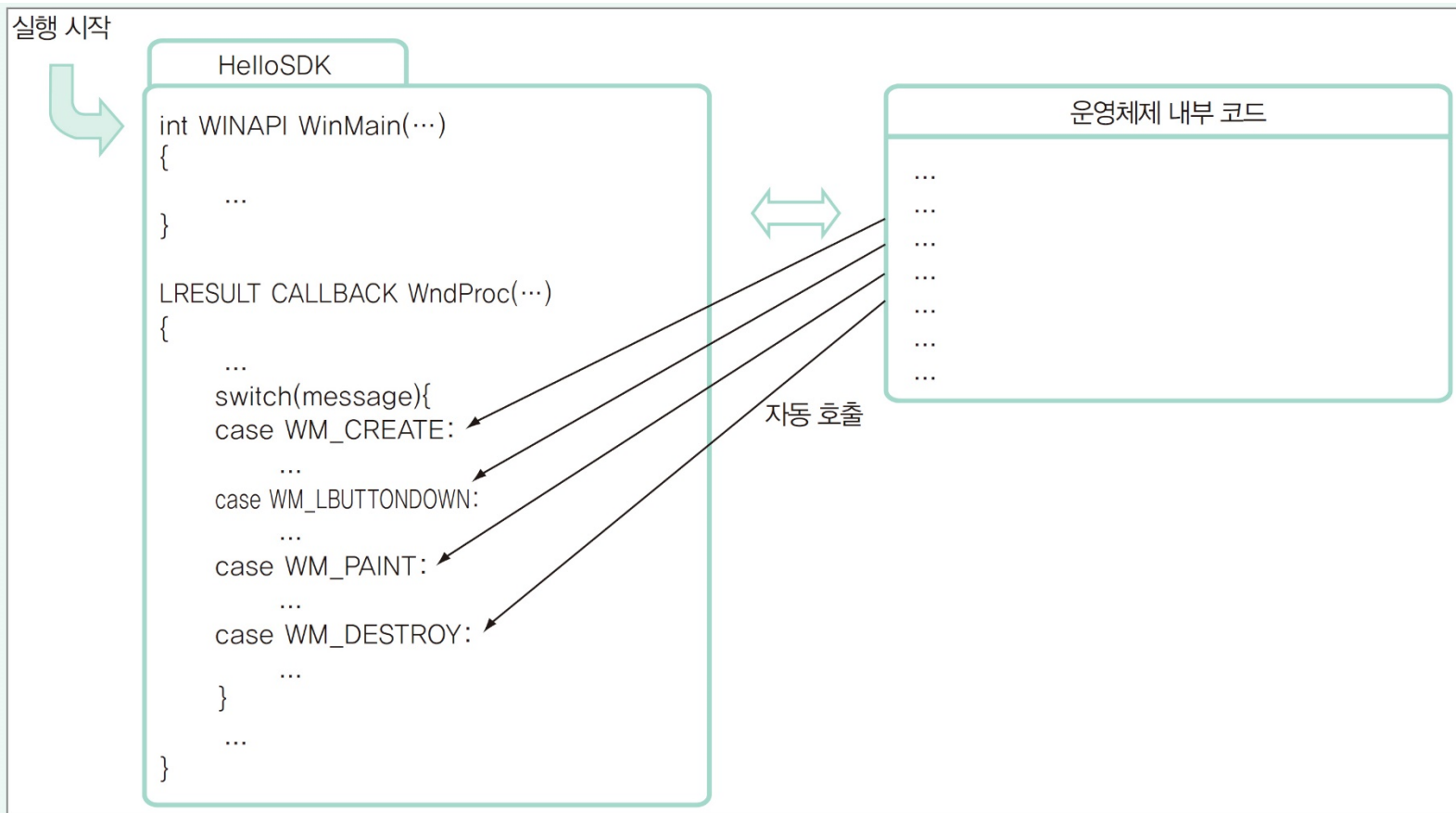
■ SDK/MFC 프로그램의 구조 비교



▲ MFC 응용 프로그램의 구조

MFC 프로그램 기본 구조

■ SDK/MFC 프로그램의 구조 비교



▲ SDK 응용 프로그램의 구조

비주얼 C++ 개발 환경

■ 비주얼 C++ 통합 개발 환경

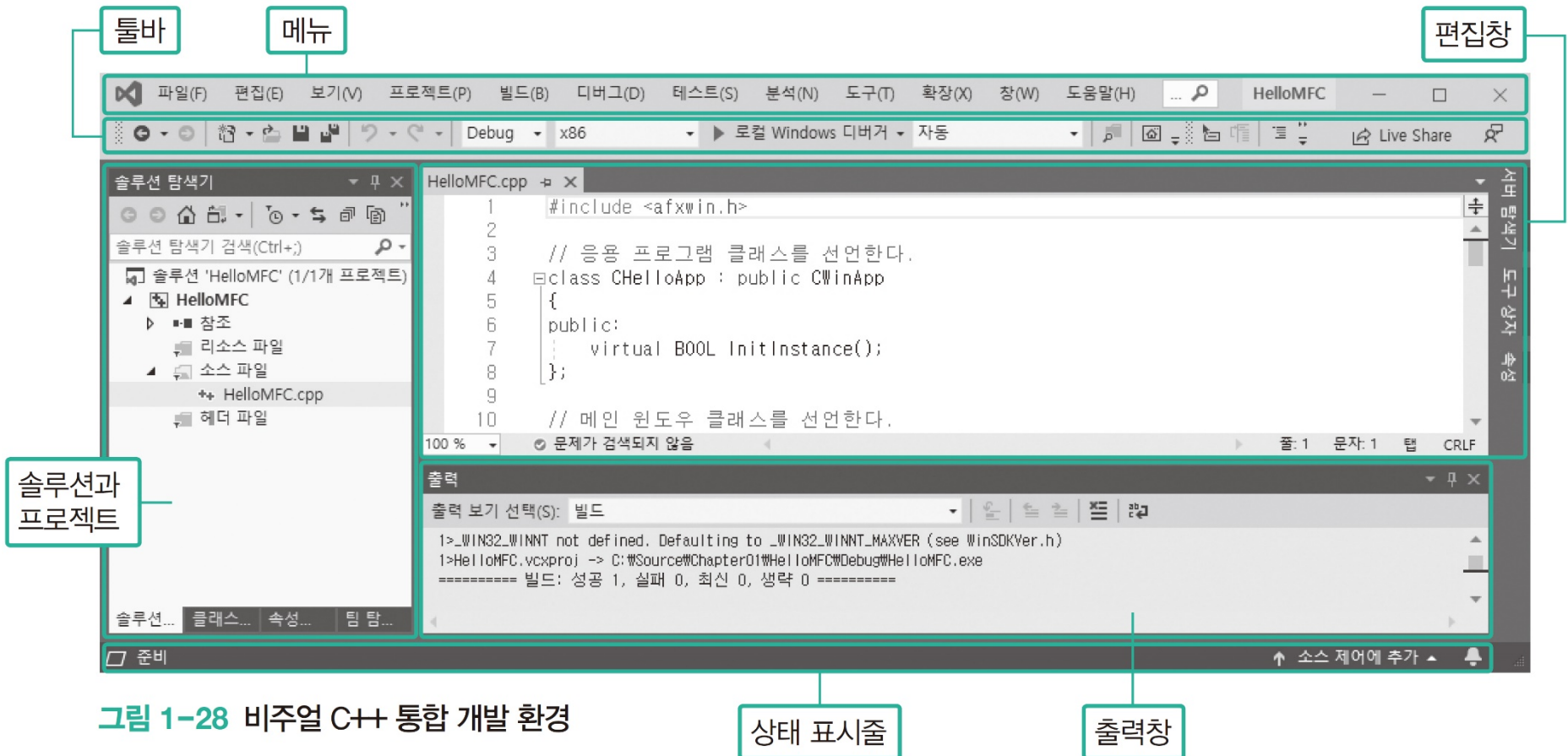


그림 1-28 비주얼 C++ 통합 개발 환경

상태 표시줄

출력창

비주얼 C++ 개발 환경

■ 메뉴

- 비주얼 C++는 다양한 기능을 메뉴로 제공
- 메뉴에서 제공하는 대부분 기능은 마우스 오른쪽 버튼을 클릭했을 때 나오는 팝업 메뉴를 통해서도 사용할 수 있음

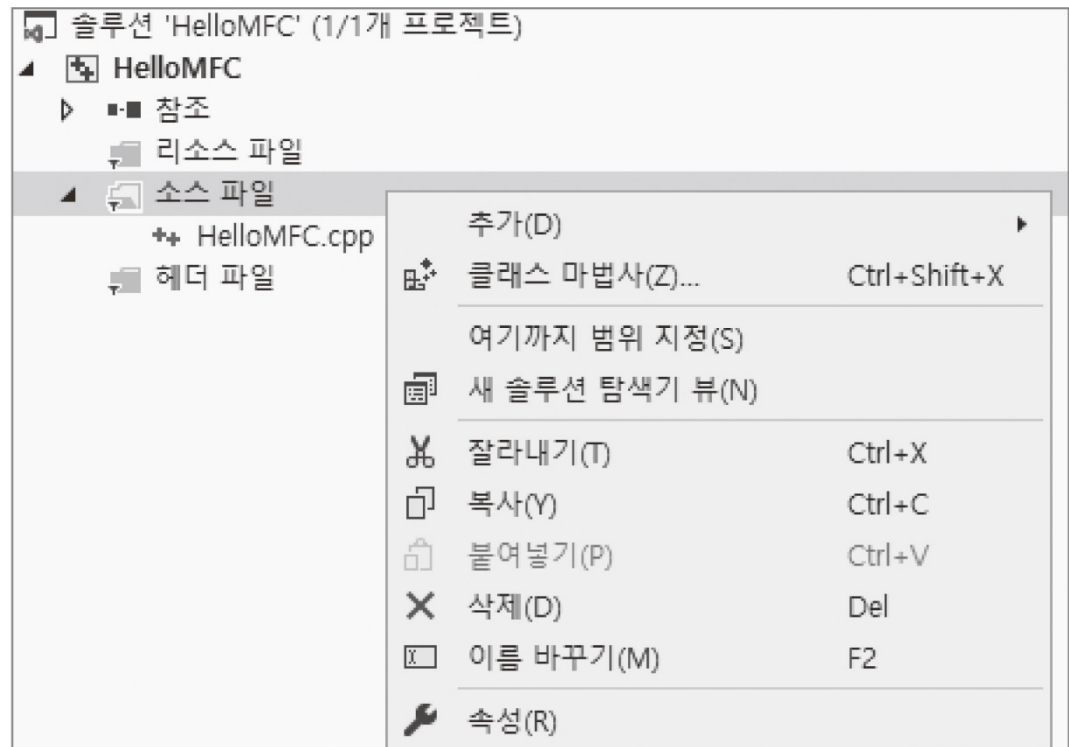


그림 1-29 팝업 메뉴

비주얼 C++ 개발 환경

■ 툴바

- 메뉴의 기능을 빠르게 사용할 수 있도록 다양한 툴바를 제공

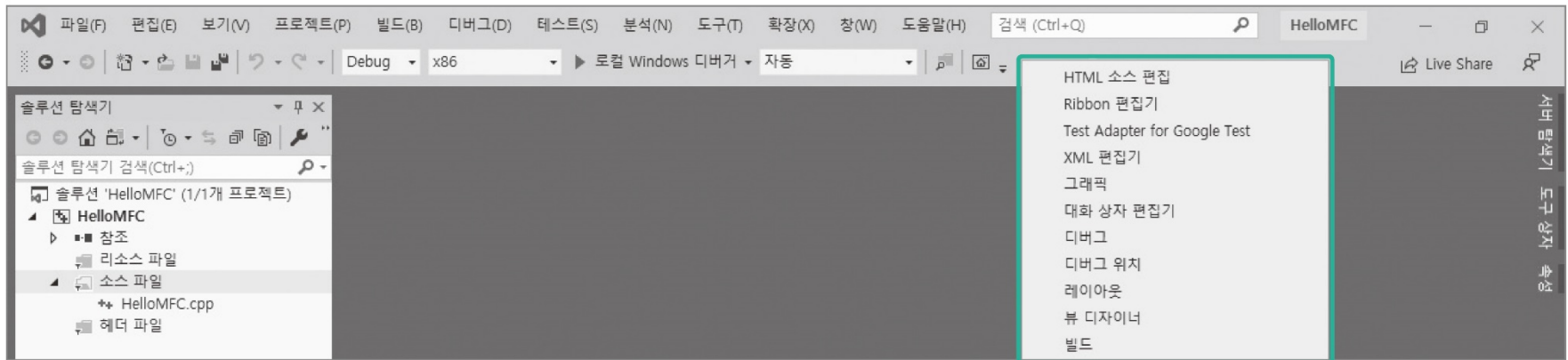


그림 1-30 툴바 선택

비주얼 C++ 개발 환경

■ 솔루션과 프로젝트

• 프로젝트 추가

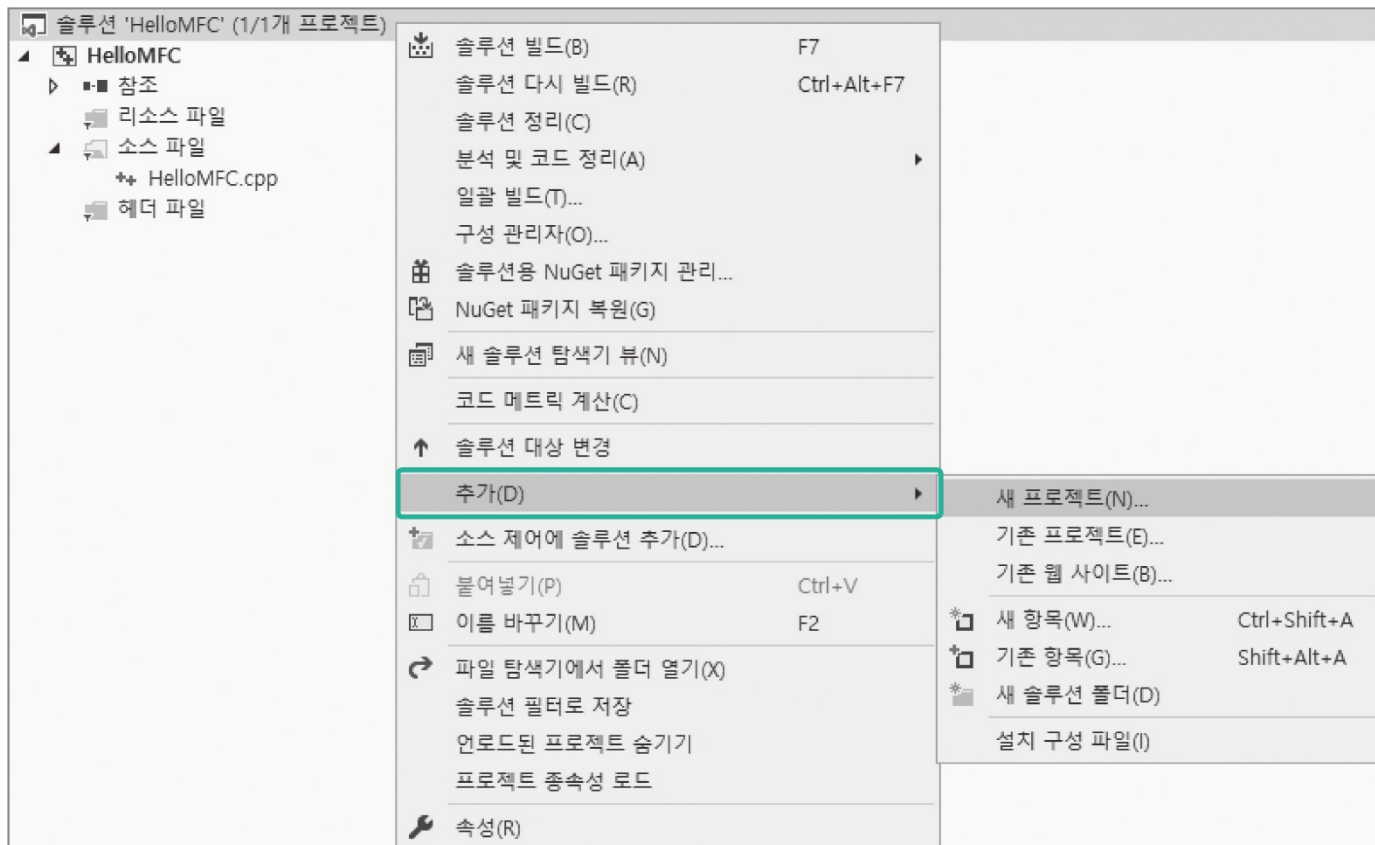


그림 1-31 프로젝트 추가

비주얼 C++ 개발 환경

■ 솔루션과 프로젝트

- 솔루션 탐색기, 클래스 뷰에서 멤버 함수 추가, 리소스 뷰

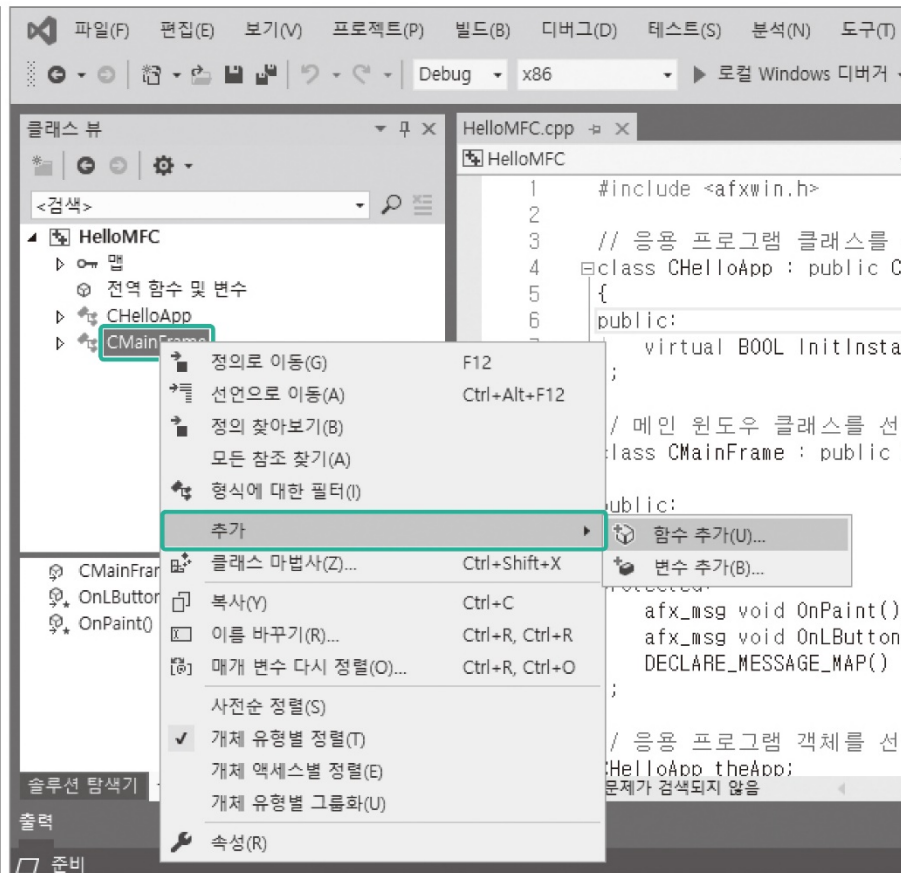
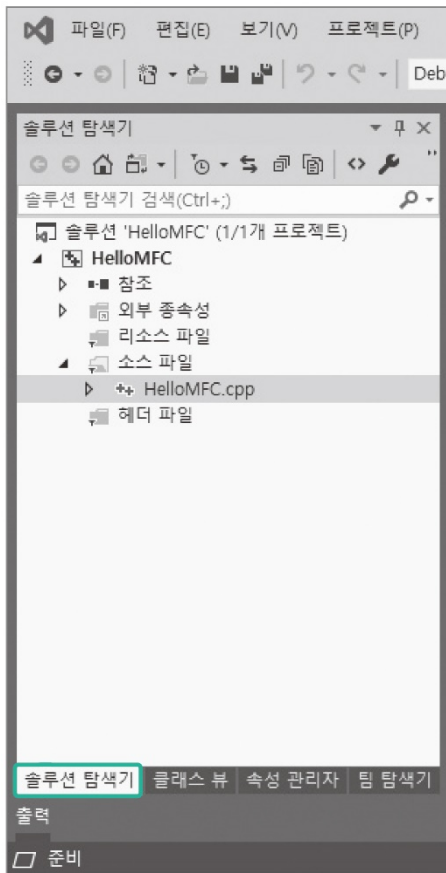


그림 1-32 솔루션 탐색기

그림 1-33 클래스 뷰에서 멤버 함수 추가

그림 1-34 리소스 뷰 49/52

비주얼 C++ 개발 환경

■ 편집



```
47 void CChildView::OnPaint()  
48 {  
49     CPaintDC dc(this);  
50     dc.Rectangle()  
51 }  
52  
53
```

▲ 1/2 ▼ BOOL Rectangle(LPCRECT lpRect)

그림 1-35 편집기

비주얼 C++ 개발 환경

■ 응용 프로그램 마법사



그림 1-36 응용 프로그램 마법사

비주얼 C++ 개발 환경

■ 속성 창과 MFC 클래스 마법사

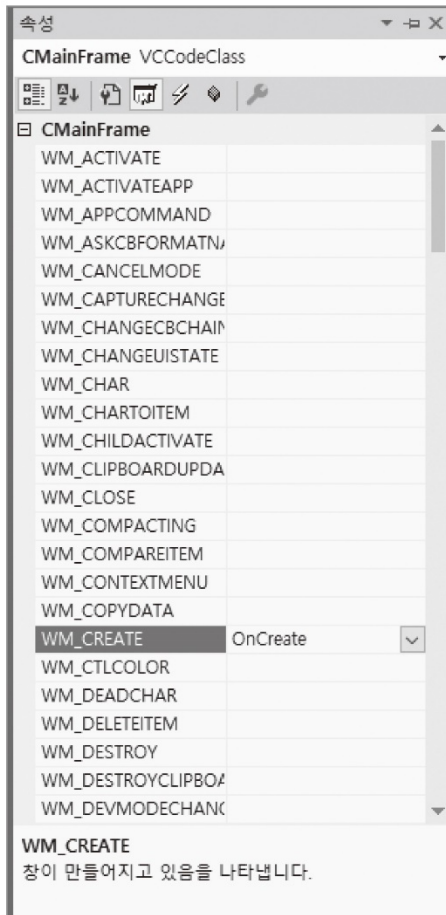


그림 1-37 속성 창

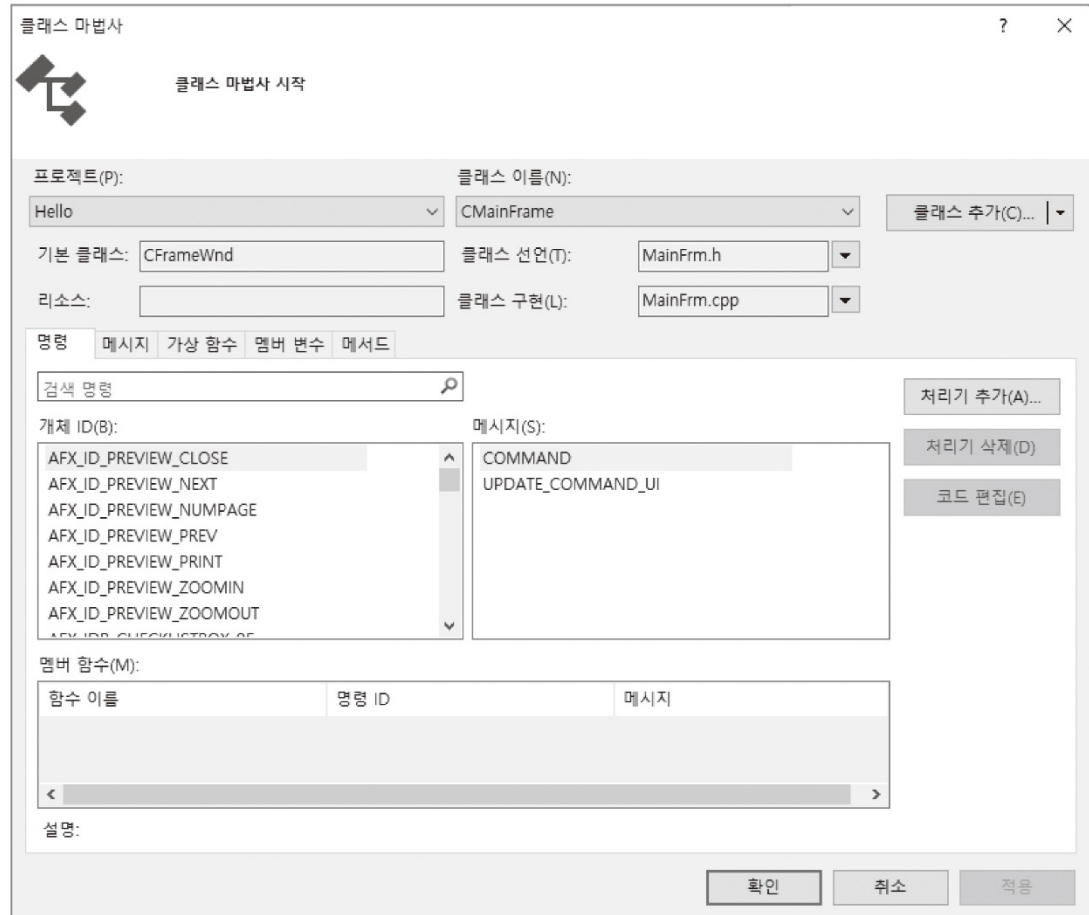


그림 1-38 MFC 클래스 마법사