



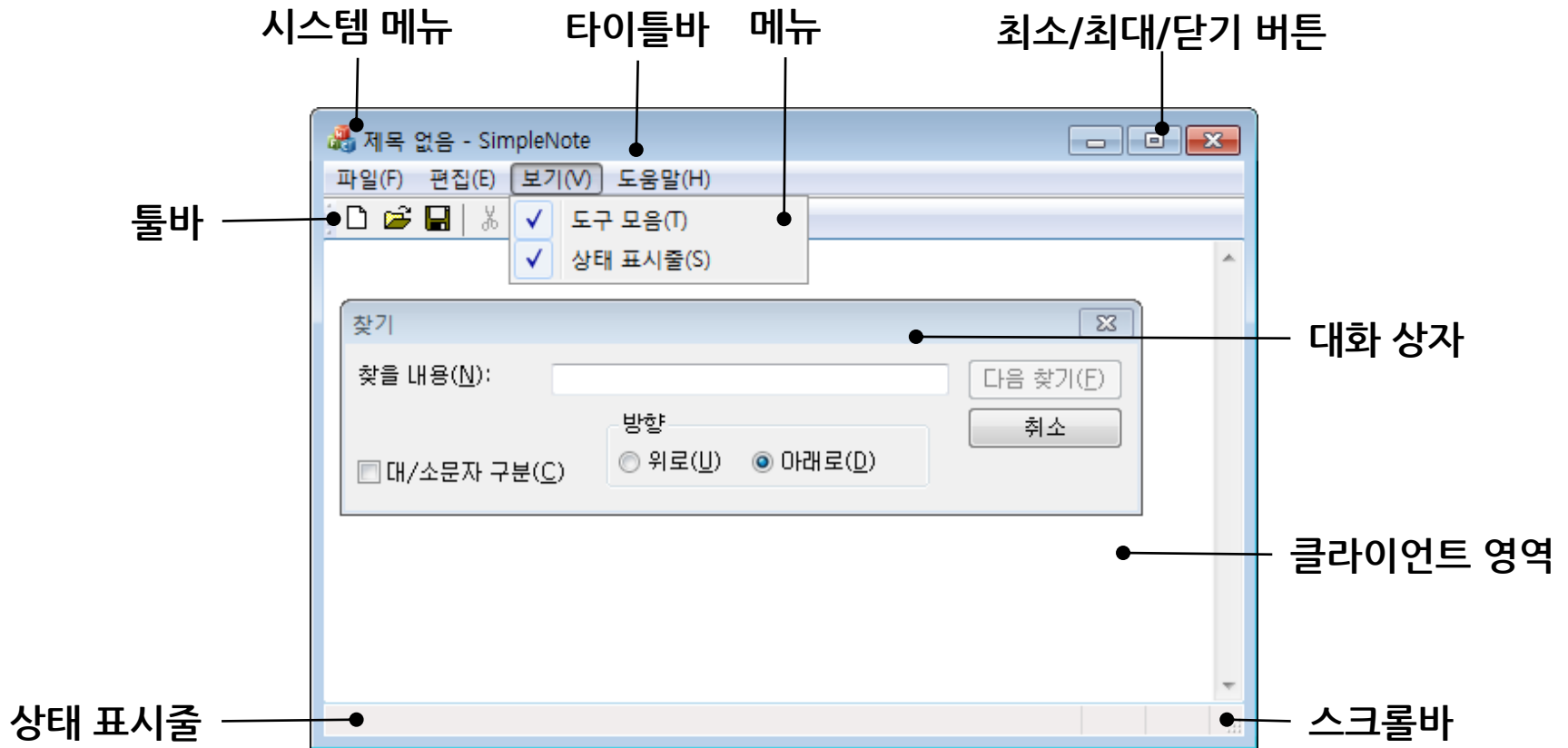
윈도우 프로그래밍

Visual C++ 2010 MFC Programming

1장. 윈도우 프로그래밍 기초

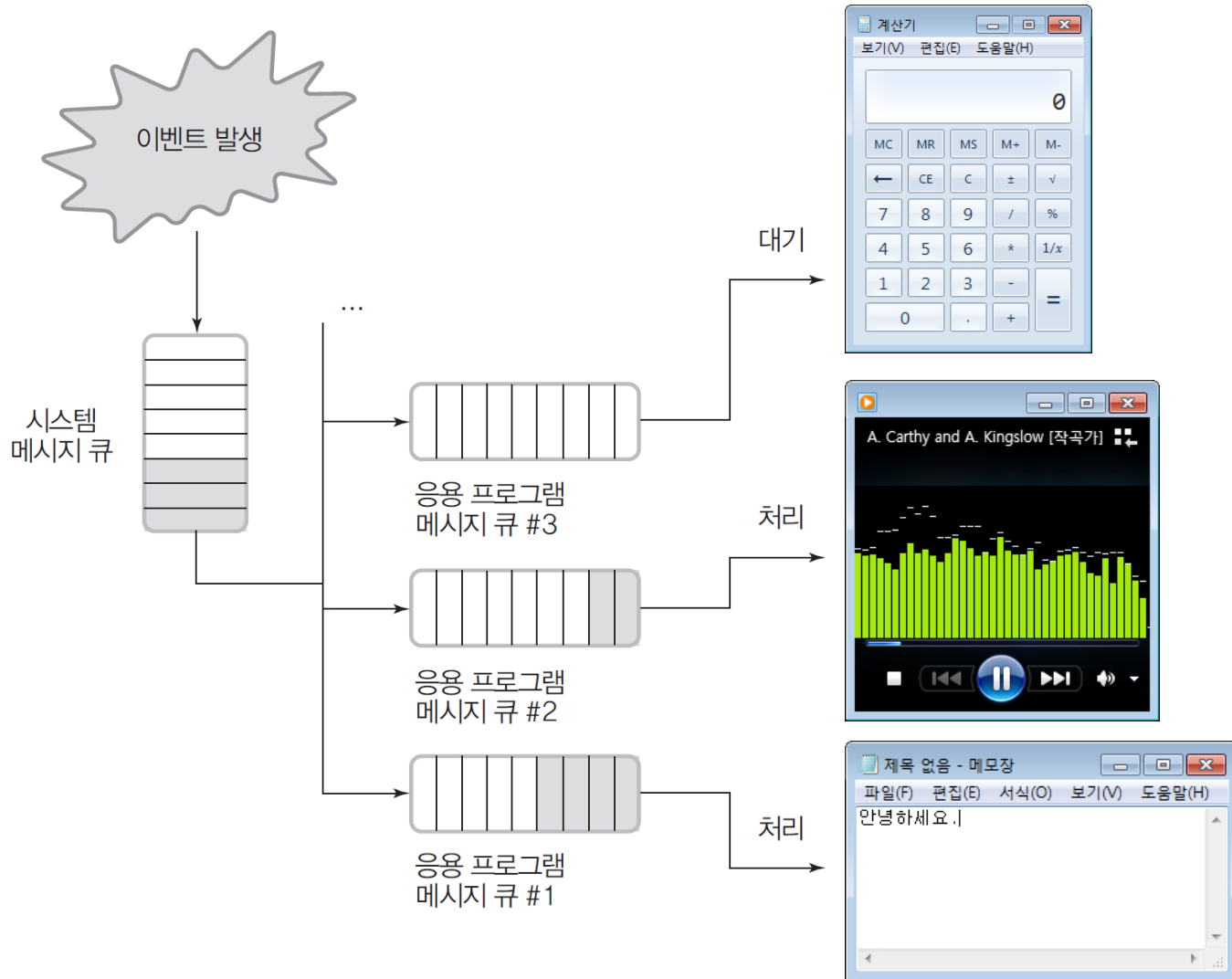
윈도우 운영체제의 특징

■ 그래픽 사용자 인터페이스



윈도우 운영체제의 특징

■ 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

윈도우 응용 프로그램의 특징

■ API 호출문 집합

- API (Application Programming Interface)
 - 윈도우 운영체제가 응용 프로그램을 위해 제공하는 각종 함수의 집합

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 (message handler)
 - 메시지를 받았을 때 동작을 결정하는 코드

응용 프로그램

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

윈도우 프로시저 : 메시지 핸들러의 집합

윈도우 응용 프로그램의 특징

■ 실행 파일과 DLL 집합

- DLL (Dynamic-Link Library)

- 프로그램이 실행 중에 결합하여 사용할 수 있는 코드와 리소스의 집합

응용 프로그램

실행파일

DLL #1

DLL #2

DLL #3

DLL #4

DLL #5

...

윈도우 응용 프로그램의 특징

■ 장치 독립성

- 주변 장치가 바뀌어도 장치 드라이버(Device Driver)만 설치하면 프로그램을 수정하지 않고 실행할 수 있음



- 하드웨어 장치 (예: 프린터, 모니터, 키보드, 네트워크) 의 제조사 및 모델에 관계없이 동일한 응용프로그램 작성 가능

윈도우 응용 프로그램의 개발 방식

■ SDK (Software Development Kit)

- 특징
 - 윈도우 API + 응용 프로그램 코드(C/C++ 언어)
- 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)

- 특징
 - 시각적 화면 디자인 + 응용 프로그램 코드 (Visual Basic 등)
- 장점
 - 간편하게 직관적으로 프로그래밍할 수 있음 (생산성이 높음)
- 단점
 - 생성 코드의 크기가 크고 실행 속도도 떨어지는 편임
 - 윈도우 운영체제가 제공하는 모든 기능을 활용한 세부적인 제어가 어려운 경우가 있음

윈도우 응용 프로그램의 개발 방식

■ 클래스 라이브러리

- 특징

- 클래스 라이브러리 + 응용 프로그램 코드(객체지향언어)

- 장점

- SDK를 이용한 방식보다 생산성이 높음
- RAD 개발 방식보다 코드 크기와 실행 속도 면에서 유리함

- 단점

- 초기 학습에 필요한 시간이 길다
 - 객체 지향 프로그래밍
 - 클래스 라이브러리의 구조와 기능, 각 클래스의 관계 파악

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 머신의 제어 하에 응용 프로그램이 구동됨(장치 독립성)
 - 윈도우 API에 버금가는 방대한 라이브러리를 제공하며, 언어에 상관없이 라이브러리를 사용 가능(언어 독립성)

간단한 SDK 프로그램 작성 - HelloSDK

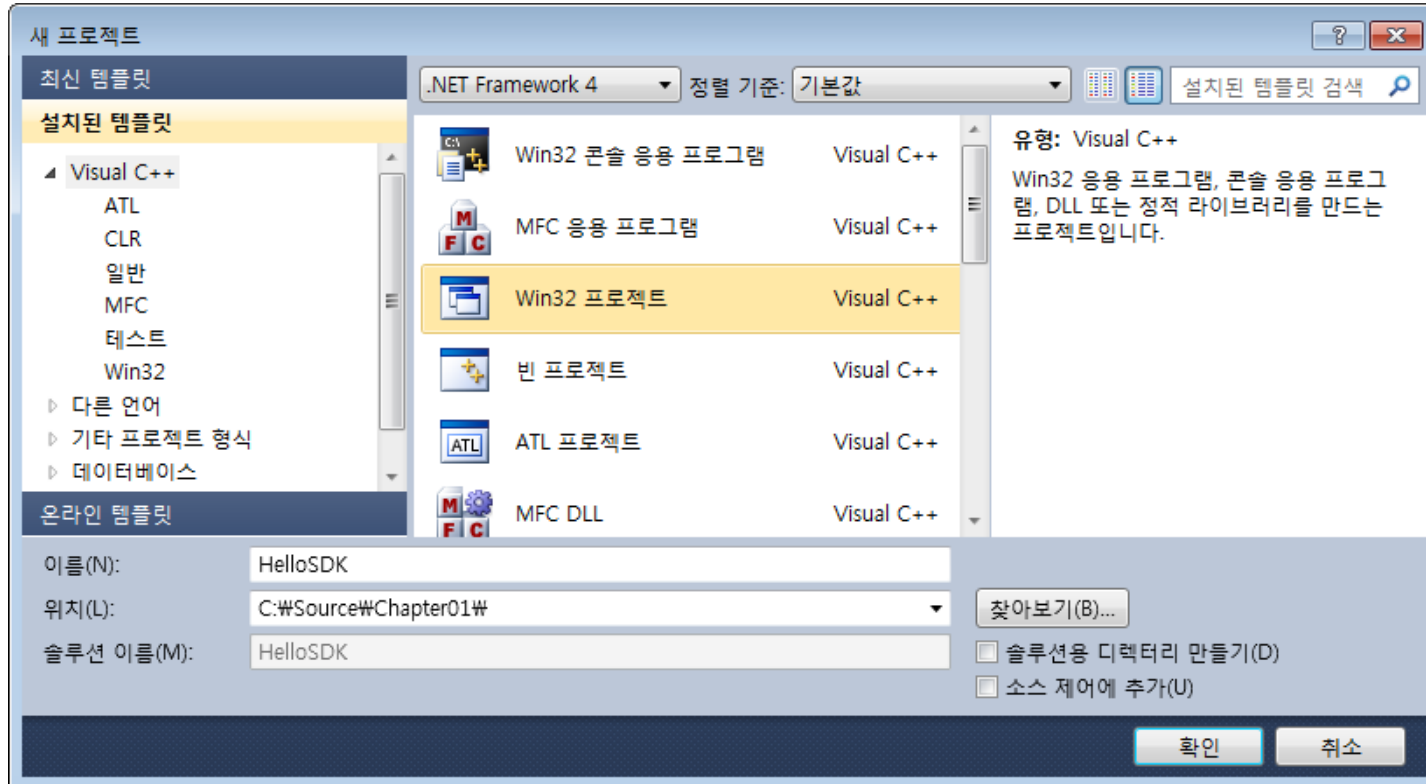
■ SDK 프로그램 기본 골격

실습 1-1 (p.36)

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

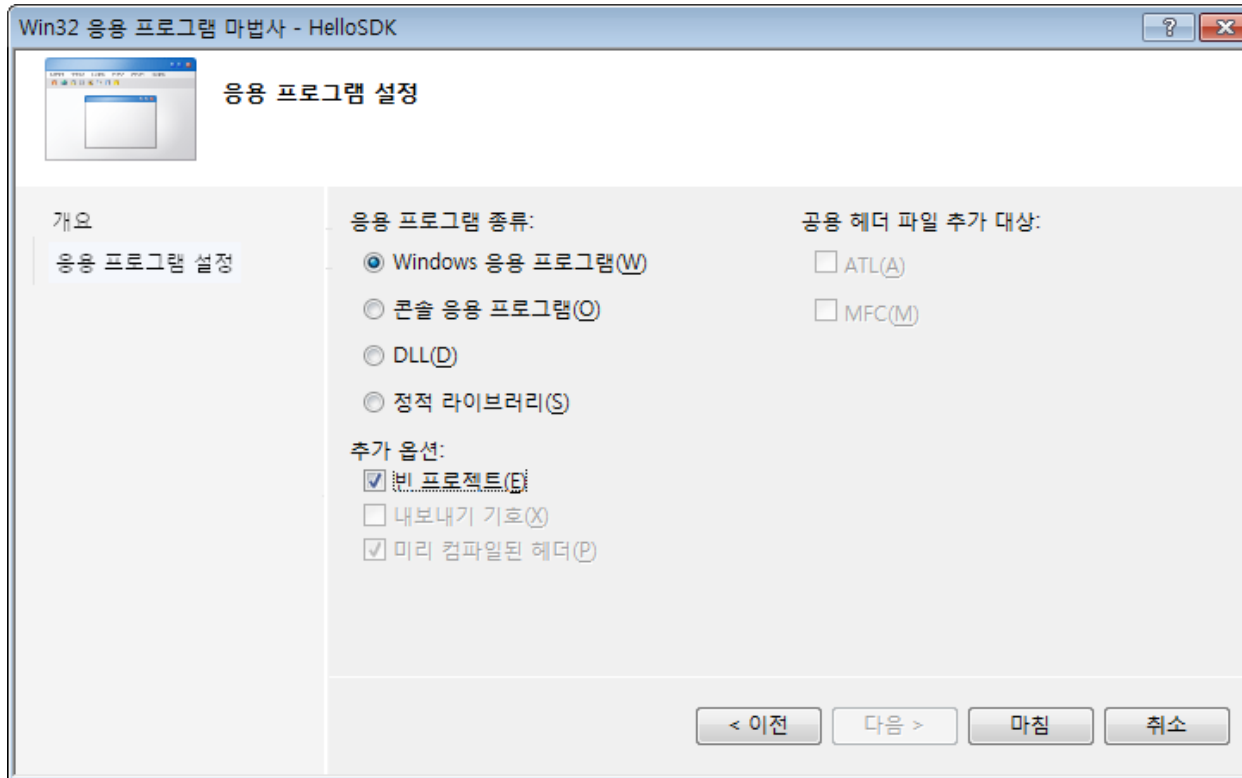
간단한 SDK 프로그램 작성 - HelloSDK

■ 프로젝트 종류 선택



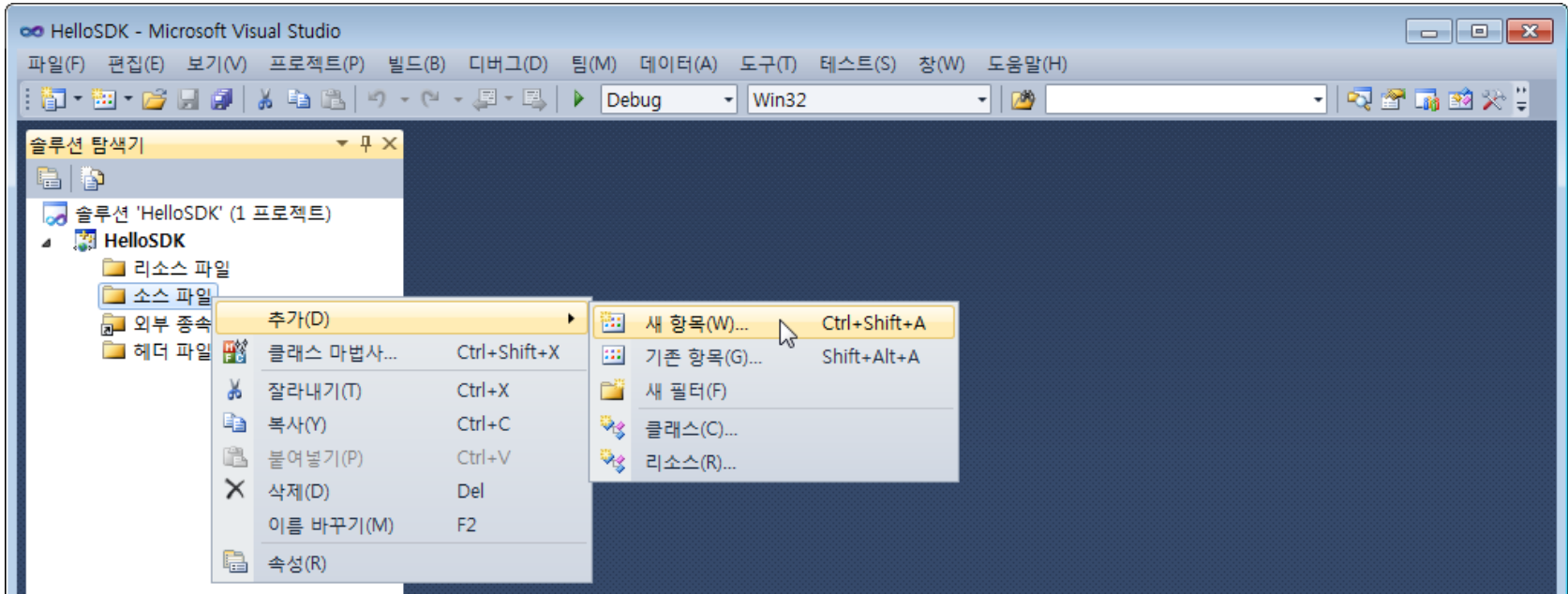
간단한 SDK 프로그램 작성 - HelloSDK

■ 프로젝트 옵션 변경



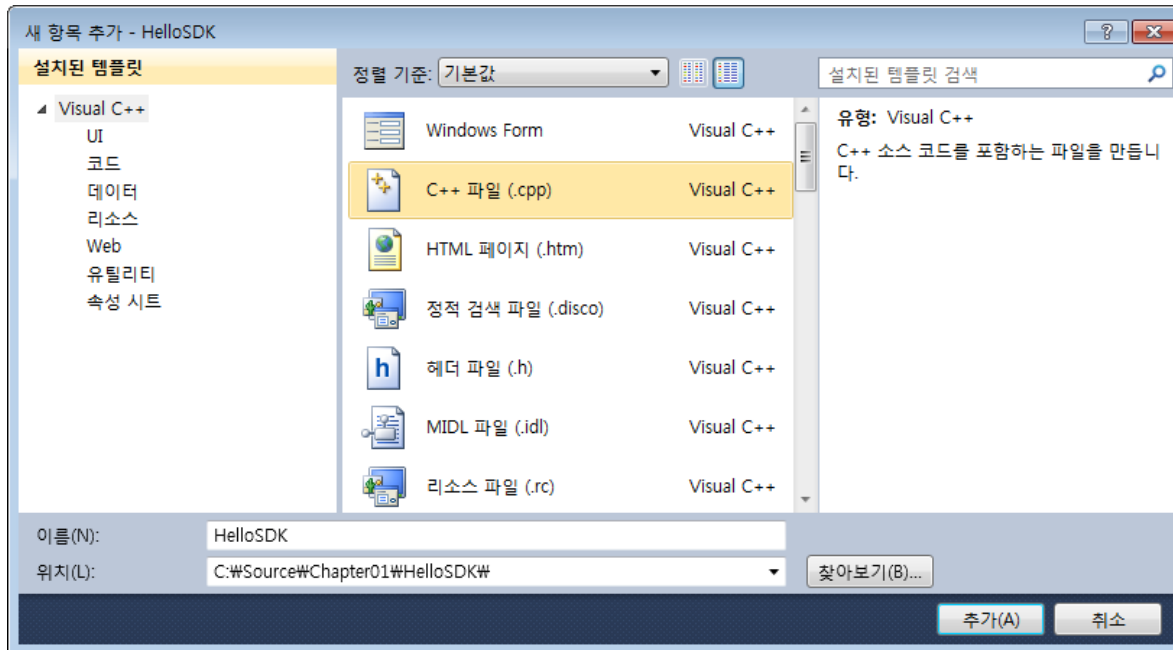
간단한 SDK 프로그램 작성 - HelloSDK

■ 소스 파일 추가(1)



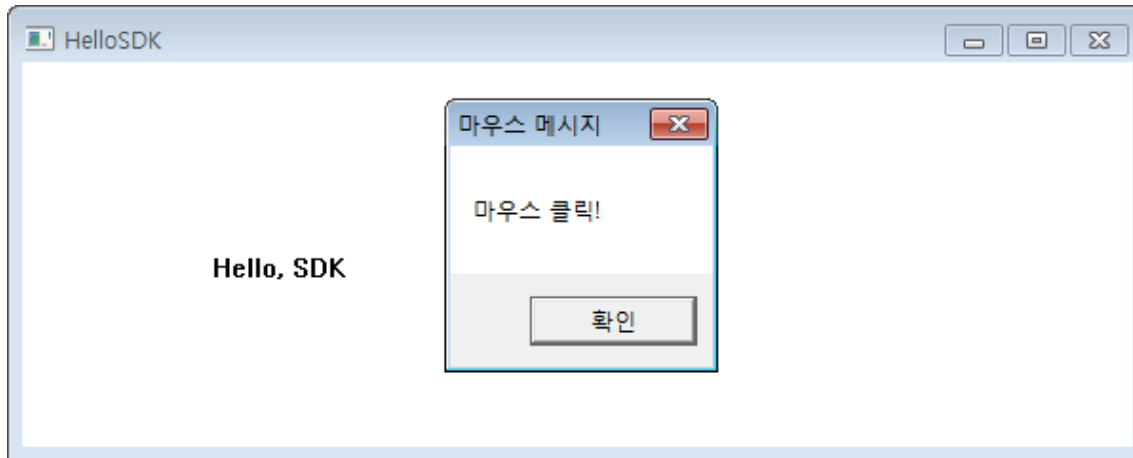
간단한 SDK 프로그램 작성 - HelloSDK

■ 소스 파일 추가(2)



간단한 SDK 프로그램 작성 - HelloSDK

■ 실행 결과



간단한 SDK 프로그램 작성 - HelloSDK

■ HelloSDK 예제 코드 (1/4)

```
#include <windows.h>
```

① 헤더 파일

```
// WinMain 함수에서 참조하므로 함수 원형을 선언한다.  
LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
```

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

② 메인 함수

```
{  
    WNDCLASS wndclass;  
    HWND hwnd;  
    MSG msg;
```

```
// 윈도우 클래스를 초기화하고 운영체제에 등록한다.  
wndclass.style = CS_HREDRAW | CS_VREDRAW; // 스타일 지정  
wndclass.lpfnWndProc = WndProc; // 윈도우 프로시저 이름  
wndclass.cbClsExtra = 0; // 여분 메모리(0바이트)  
wndclass.cbWndExtra = 0; // 여분 메모리(0바이트)  
wndclass.hInstance = hInstance; // 인스턴스 핸들  
wndclass.hIcon = LoadIcon(NULL, IDI_APPLICATION); // 아이콘 모양  
wndclass.hCursor = LoadCursor(NULL, IDC_ARROW); // 커서 모양
```

③ 윈도우 클래스
초기화와 등록

간단한 SDK 프로그램 작성 - HelloSDK

■ HelloSDK 예제 코드 (2/4)

```
wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH); // 배경(흰색)
wndclass.lpszMenuName = NULL; // 메뉴(NULL->메뉴 없음)
wndclass.lpszClassName = TEXT("HelloClass "); // 윈도우 클래스 이름
if(!RegisterClass(&wndclass)) return 1;
```

```
// 윈도우를 생성하고 화면에 나타낸다.
hwnd = CreateWindow(TEXT("HelloClass "), TEXT("HelloSDK "),
    WS_OVERLAPPEDWINDOW, CW_USEDEFAULT, CW_USEDEFAULT,
    CW_USEDEFAULT, CW_USEDEFAULT, NULL, NULL, hInstance, NULL);
ShowWindow(hwnd, nCmdShow);
```

④ 윈도우 생성

```
// 메시지 큐에서 메시지를 하나씩 꺼내서 처리한다.
while(GetMessage(&msg, NULL, 0, 0) > 0){
    TranslateMessage(&msg);
    DispatchMessage(&msg);
}
```

⑤ 메시지 루프

```
return msg.wParam;
}
```

간단한 SDK 프로그램 작성 - HelloSDK

■ HelloSDK 예제 코드 (3/4)

```
LRESULT CALLBACK WndProc(HWND hwnd, UINT message,
                          WPARAM wParam, LPARAM lParam)
{
    HDC hdc;
    PAINTSTRUCT ps;
    TCHAR *str = TEXT("Hello, SDK ");

    // 발생한 메시지의 종류에 따라 적절히 처리한다.
    switch(message){
    case WM_CREATE:
        return 0;
    case WM_LBUTTONDOWN:
        MessageBox(hwnd, TEXT("마우스 클릭! "), TEXT("마우스 메시지 "), MB_OK);
        return 0;
    case WM_PAINT:
        hdc = BeginPaint(hwnd, &ps);
        TextOut(hdc, 100, 100, str, lstrlen(str));
        EndPaint(hwnd, &ps);
        return 0;
    }
```

⑥ 윈도우 프로시저

간단한 SDK 프로그램 작성 - HelloSDK

■ HelloSDK 예제 코드 (4/4)

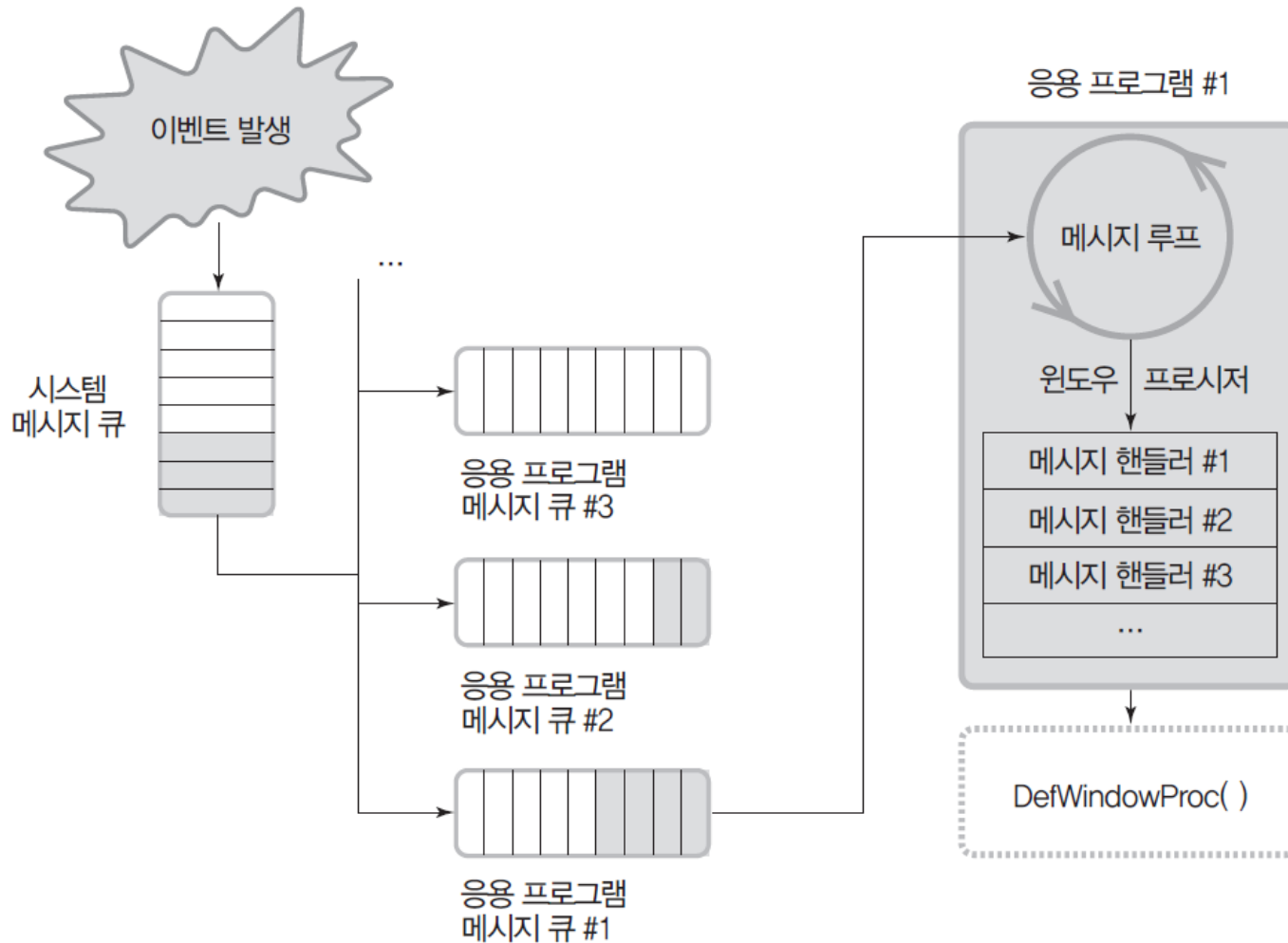
```
case WM_DESTROY:  
    PostQuitMessage(0);  
    return 0;  
}
```

```
// 응용 프로그램이 처리하지 않은 메시지는 운영체제가 처리한다.  
return DefWindowProc(hwnd, message, wParam, lParam);
```

```
}
```

SDK 프로그램 기본 구조

■ SDK 프로그램 동작 원리

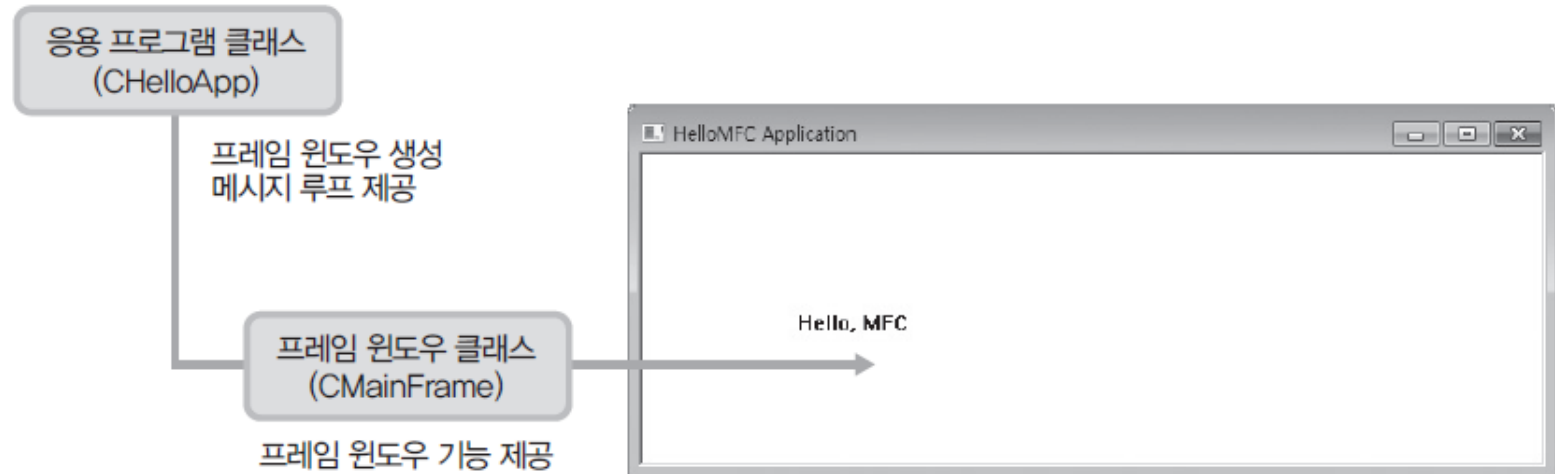


간단한 MFC 프로그램 작성 - HelloMFC

■ MFC 프로그램 기본 골격

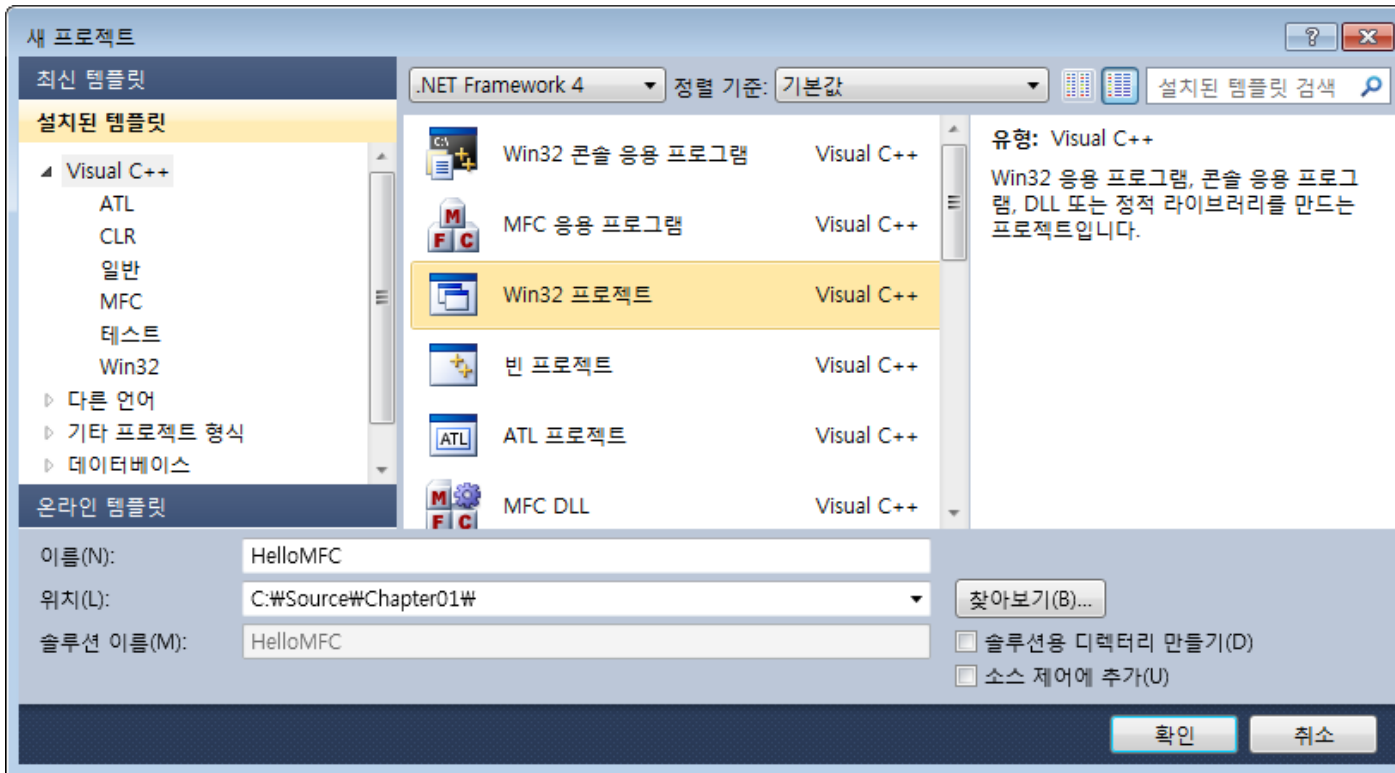
실습 1-2 (p.47)

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



간단한 MFC 프로그램 작성 - HelloMFC

■ 프로젝트 종류 선택



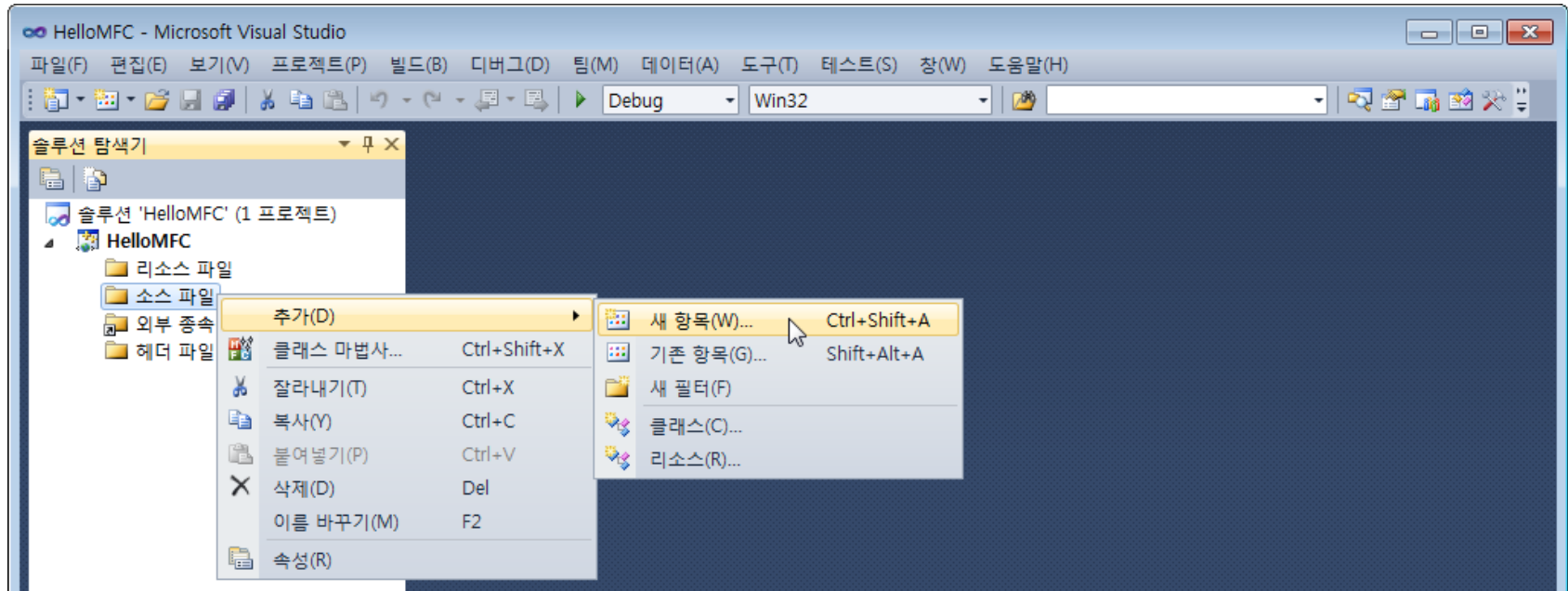
간단한 MFC 프로그램 작성 - HelloMFC

■ 프로젝트 옵션 변경



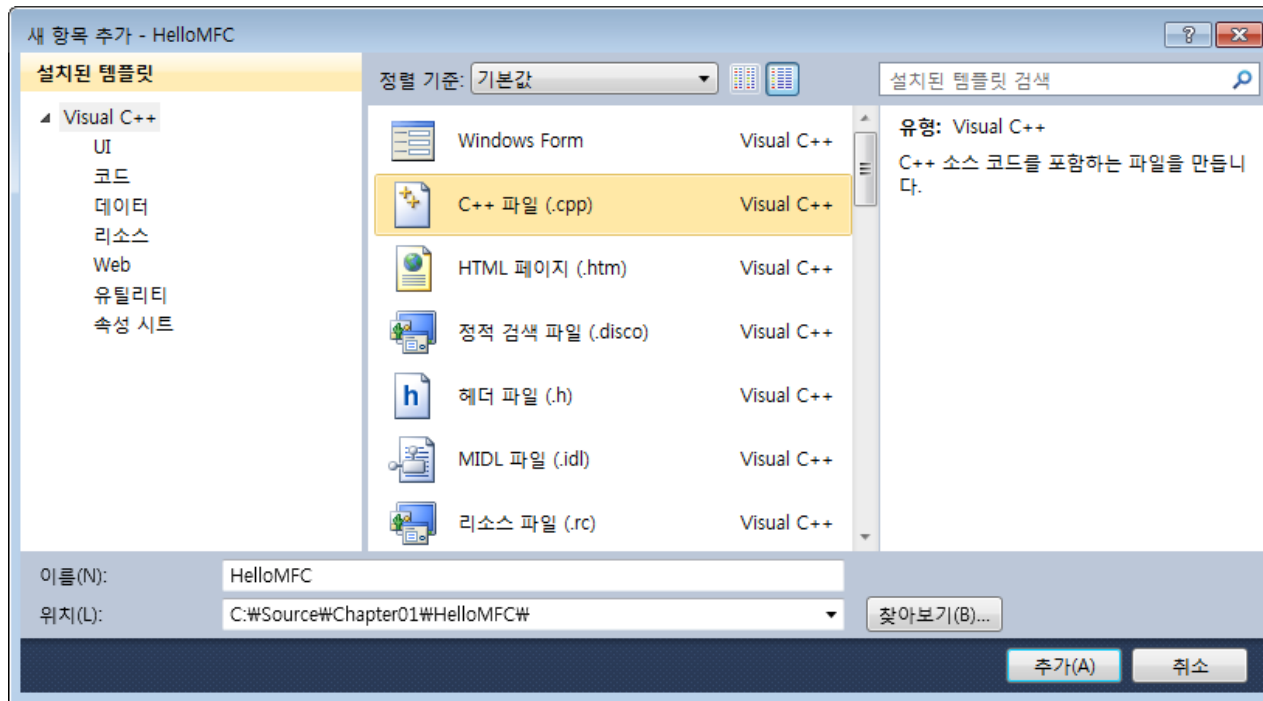
간단한 MFC 프로그램 작성 - HelloMFC

■ 소스 파일 추가(1)



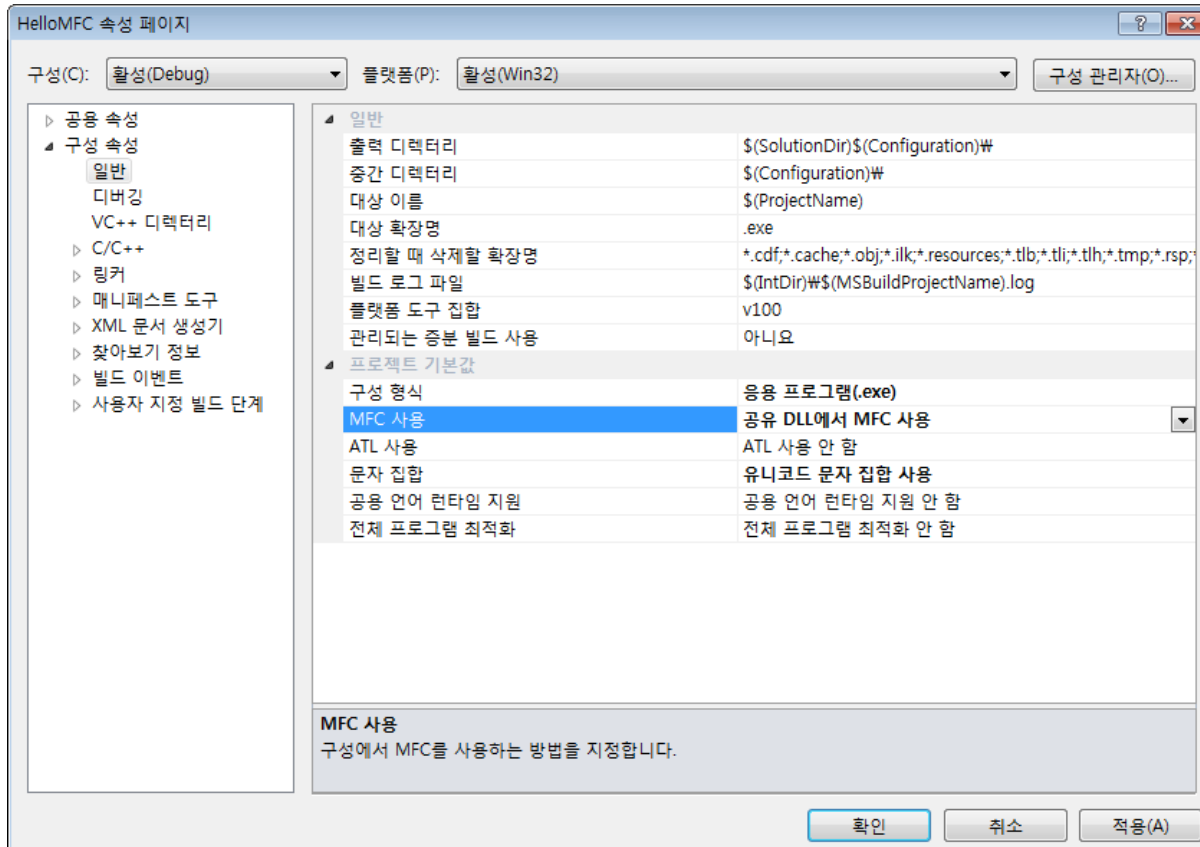
간단한 MFC 프로그램 작성 - HelloMFC

■ 소스 파일 추가(2)



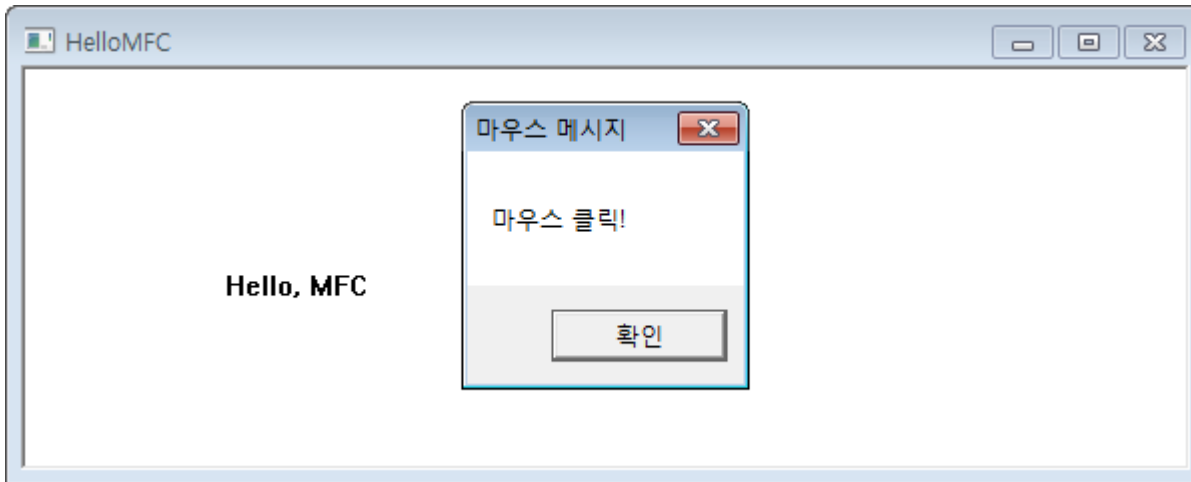
간단한 MFC 프로그램 작성 - HelloMFC

■ 프로젝트 속성 변경



간단한 MFC 프로그램 작성 - HelloMFC

■ 실행 결과



간단한 MFC 프로그램 작성 - HelloMFC

■ HelloMFC 예제 코드 (1/3)

```
#include <afxwin.h>
```

① 헤더 파일

```
// 응용 프로그램 클래스를 선언한다.
```

② 클래스 선언부

```
class CHelloApp : public CWinApp  
{  
public:  
    virtual BOOL InitInstance();  
};
```

```
// 메인 윈도우 클래스를 선언한다.
```

```
class CMainFrame : public CFrameWnd  
{  
public:  
    CMainFrame();
```

```
protected:
```

```
    afx_msg void OnPaint();  
    afx_msg void OnLButtonDown(UINT nFlags, CPoint point);  
    DECLARE_MESSAGE_MAP()  
};
```

간단한 MFC 프로그램 작성 - HelloMFC

■ HelloMFC 예제 코드 (2/3)

```
// 응용 프로그램 객체를 선언한다.  
CHelloApp theApp;
```

③ 응용 프로그램 객체

```
// 응용 프로그램 클래스를 정의한다.  
BOOL CHelloApp::InitInstance()  
{  
    m_pMainWnd = new CMainFrame;  
    m_pMainWnd->ShowWindow(m_nCmdShow);  
    return TRUE;  
}
```

④ 클래스 정의부

```
// 메인 윈도우 클래스를 정의한다.  
CMainFrame::CMainFrame()  
{  
    Create(NULL, _T("HelloMFC "));  
}
```


간단한 MFC 프로그램 작성 - HelloMFC

■ HelloMFC 예제 코드 (3/3)

```
void CMainFrame::OnPaint()
{
    CPaintDC dc(this);
    TCHAR *msg = _T("Hello, MFC ");
    dc.TextOut(100, 100, msg, lstrlen(msg));
}

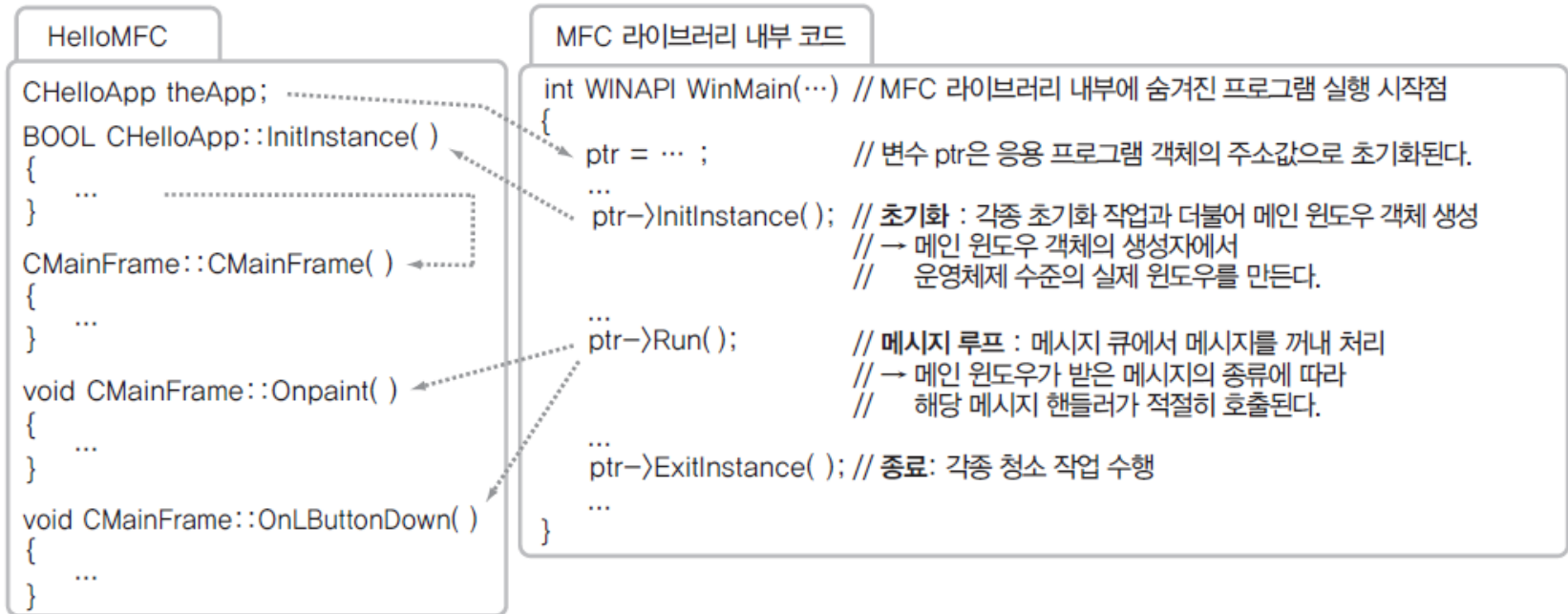
void CMainFrame::OnLButtonDown(UINT nFlags, CPoint point)
{
    MessageBox(_T("마우스 클릭! "), _T("마우스 메시지 "));
}

// 메시지 맵을 선언한다.
BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
    ON_WM_PAINT()
    ON_WM_LBUTTONDOWN()
END_MESSAGE_MAP()
```

⑤ 메시지 맵

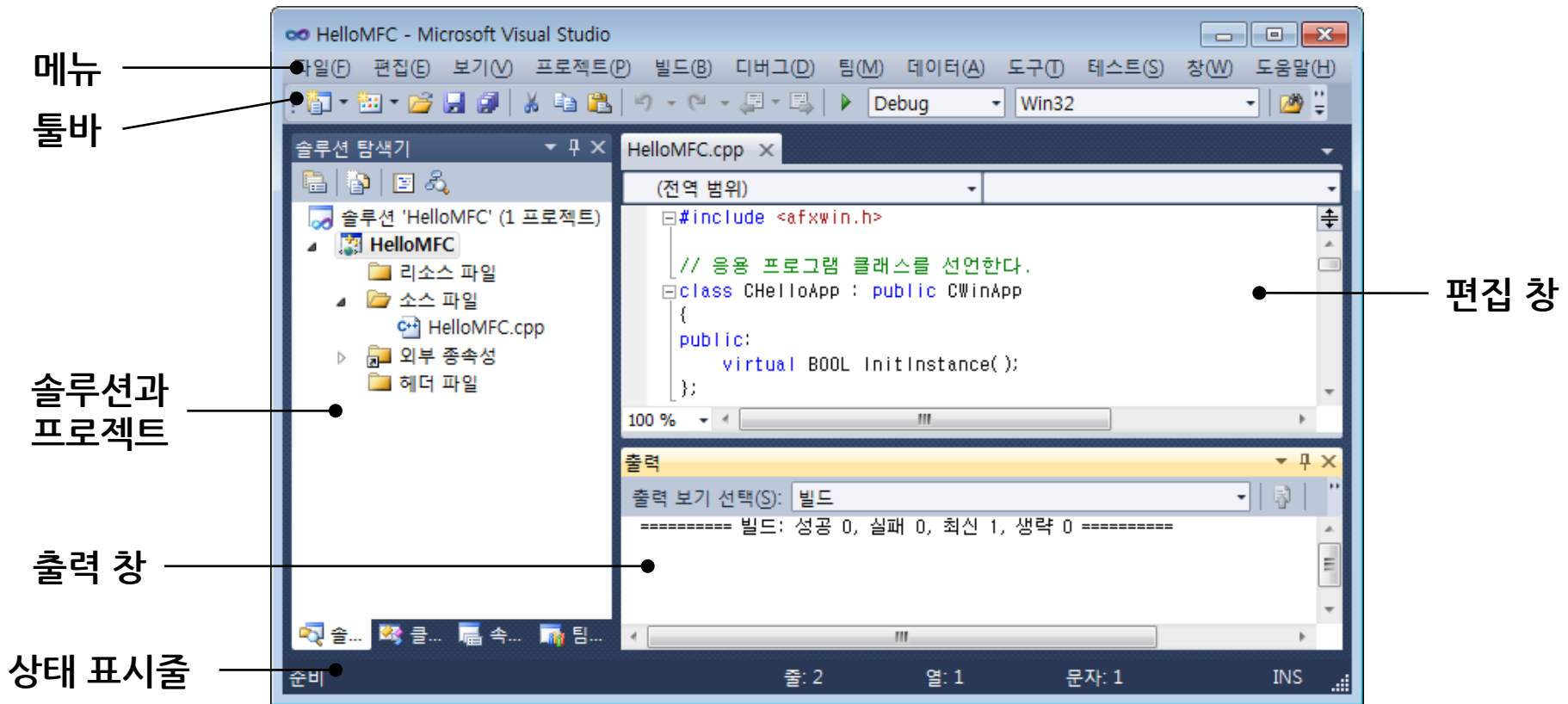
MFC 프로그램 기본 구조

■ MFC 프로그램 동작 원리



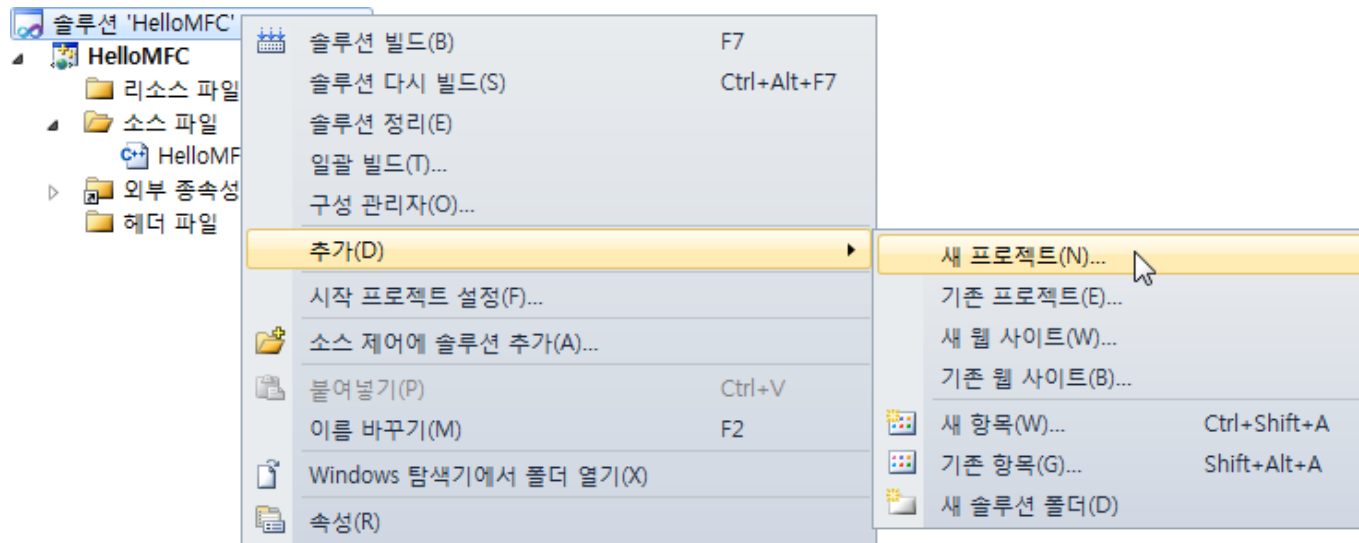
비주얼 C++ 개발 환경

■ 비주얼 C++ 통합 개발 환경



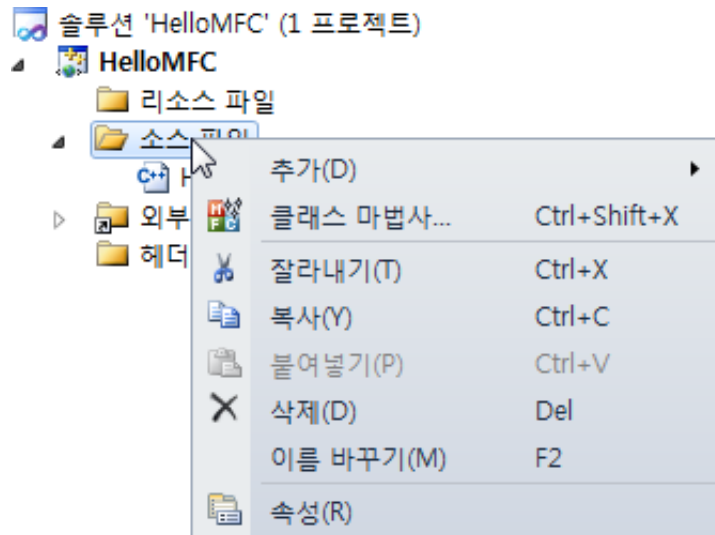
비주얼 C++ 개발 환경

- 솔루션과 프로젝트
 - 프로젝트 추가



비주얼 C++ 개발 환경

■ 소스 파일 추가



비주얼 C++ 개발 환경

- 솔루션 탐색기 / 클래스 뷰 / 리소스 뷰

