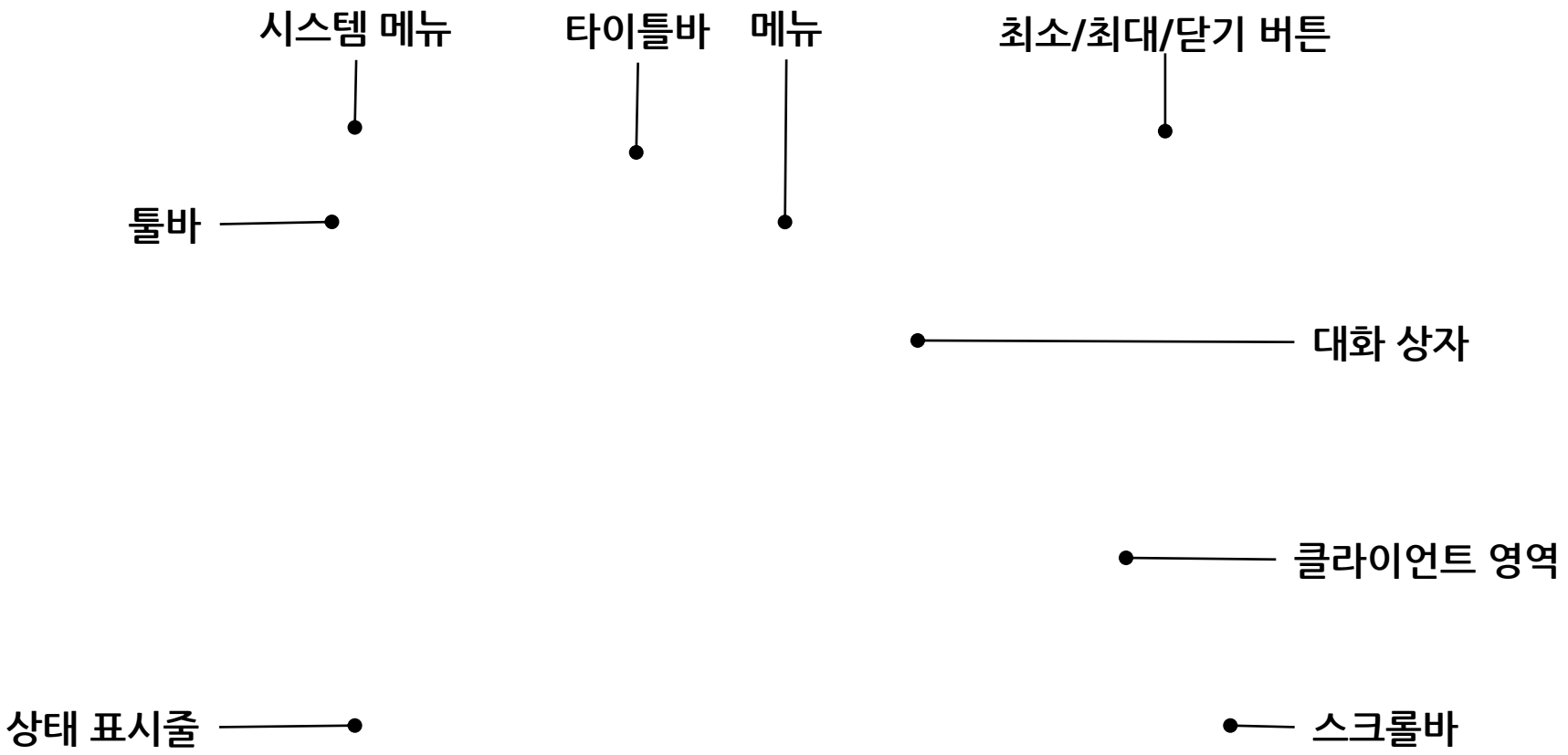
윈도우 프로그래밍
Visual C++ MFC Programming

윈도우 프로그래밍 기초

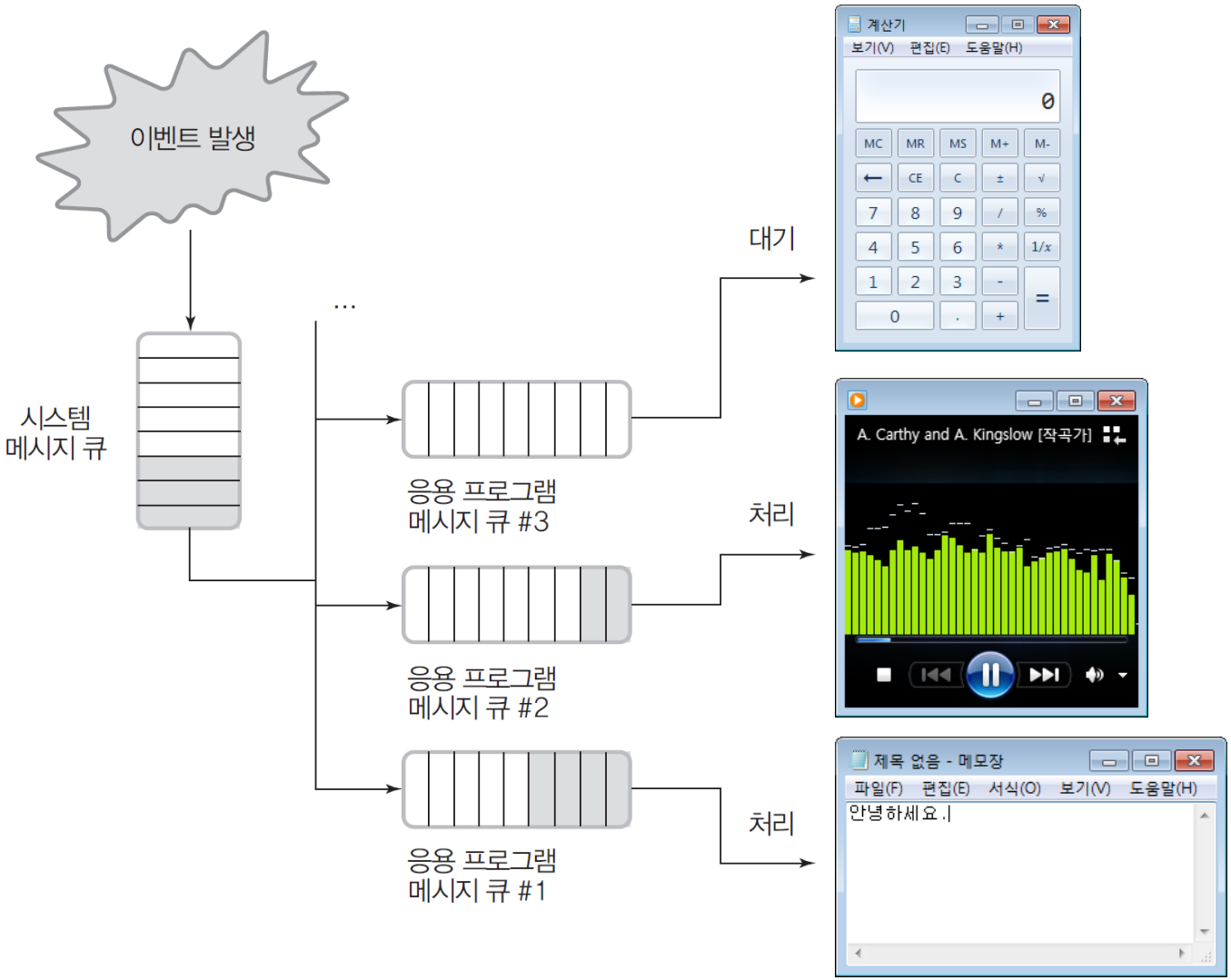
윈도우 운영체제의 특징

- 사용자 인터페이스의 구성 요소



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
    - 16비트 윈도우 → Win16 API
    - 32 또는 64비트 윈도우 → Win32 API

윈도우 응용 프로그램의 특징

- 메시지 핸들러 집합
  - 메시지 핸들러 : 메시지를 받았을 때 동작을 결정하는 코드

  - 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
  - 윈도우 운영체제가 제공하는 API는 DLL 형태로 제공되며, 응용 프로그래머는 필요한 기능을 DLL로 제작하기도 함

윈도우 응용 프로그램의 특징

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

윈도우 응용 프로그램의 개발 방식

- SDK
  - 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
  - 특징
    - 시각적 화면 디자인 ✚ 응용 프로그램 코드
  - 장점
    - 간편하게 직관적으로 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 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
    - API를 직접 사용해서 세부적으로 제어할 수 있음
    - RAD 개발 방식보다 코드 크기와 실행 속도 면에서 유리함
  - 단점
    - 객체 지향 프로그래밍에 익숙해야 함
    - 클래스 라이브러리의 구조와 각 클래스의 기능 및 관계를 파악하기 위한 초기 학습 기간이 긴 편임

윈도우 응용 프로그램의 개발 방식

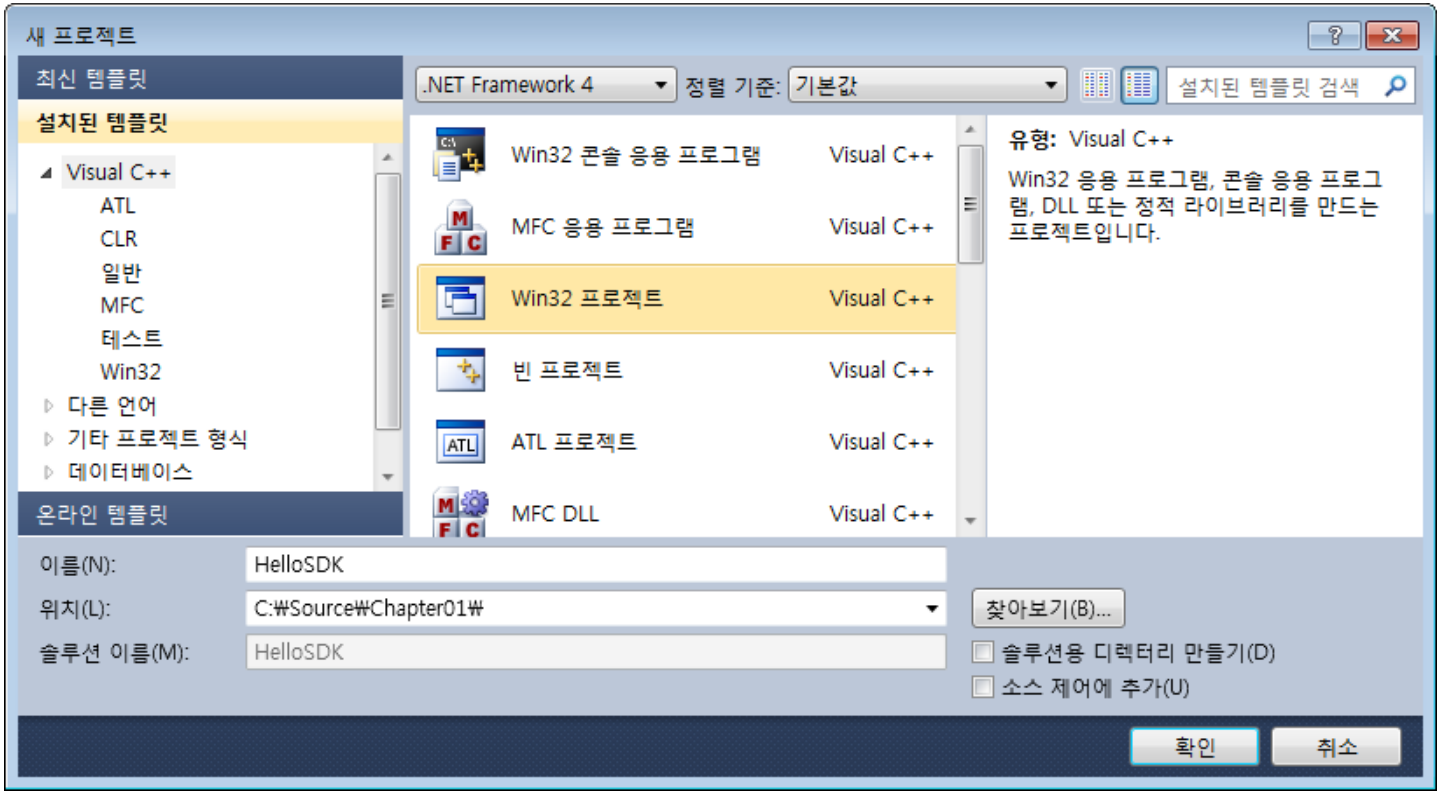
- .NET 프레임워크
  - 윈도우 운영체제에 설치할 수 있는 소프트웨어 개발 및 실행 환경
  - 특징
    - 공용 언어 런타임(CLR, Common Language Runtime)이라는 소프트웨어 가상 머신을 제공하며, 가상 머신의 제어 하에 응용 프로그램이 구동됨(장치 독립성)
    - 윈도우 API에 버금가는 방대한 라이브러리를 제공하며, 언어에 상관없이 라이브러리를 사용 가능(언어 독립성)

간단한 SDK 프로그램 작성 – HelloSDK

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

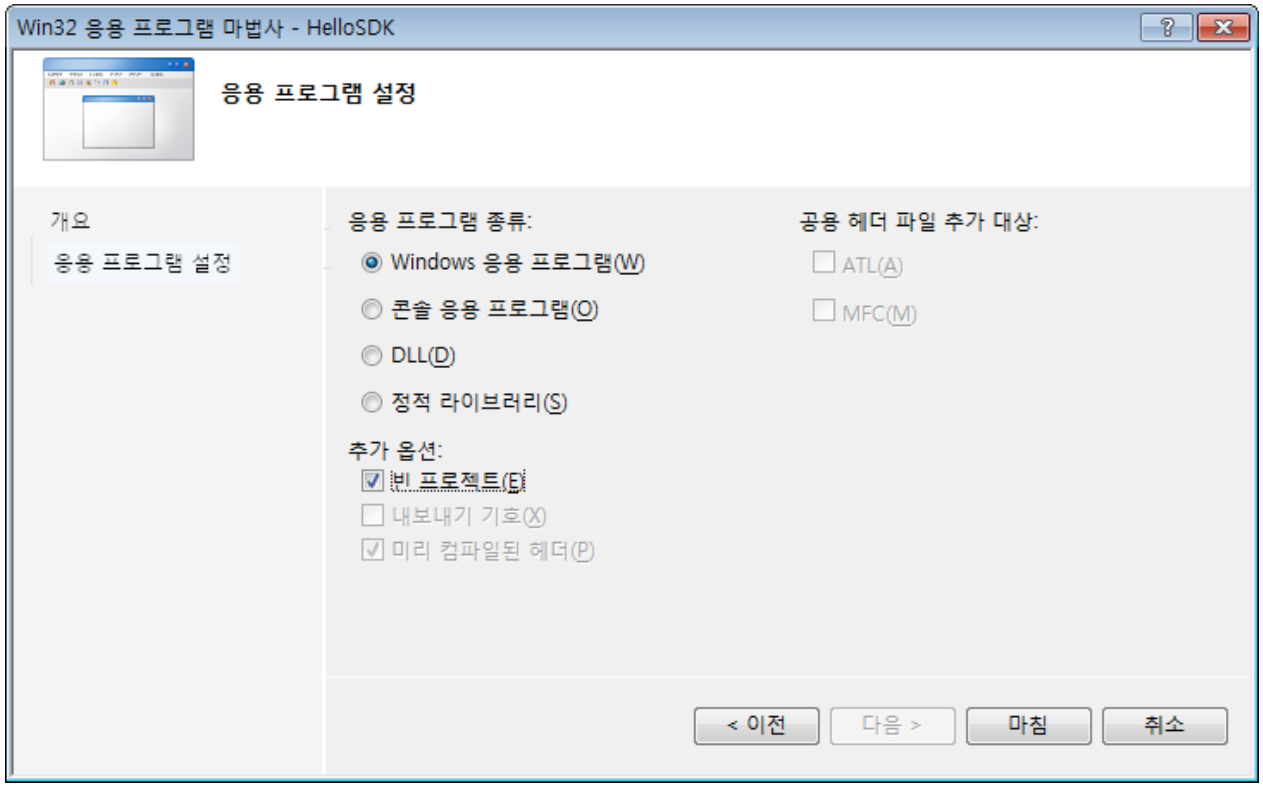
간단한 SDK 프로그램 작성 – HelloSDK

- 프로젝트 종류 선택



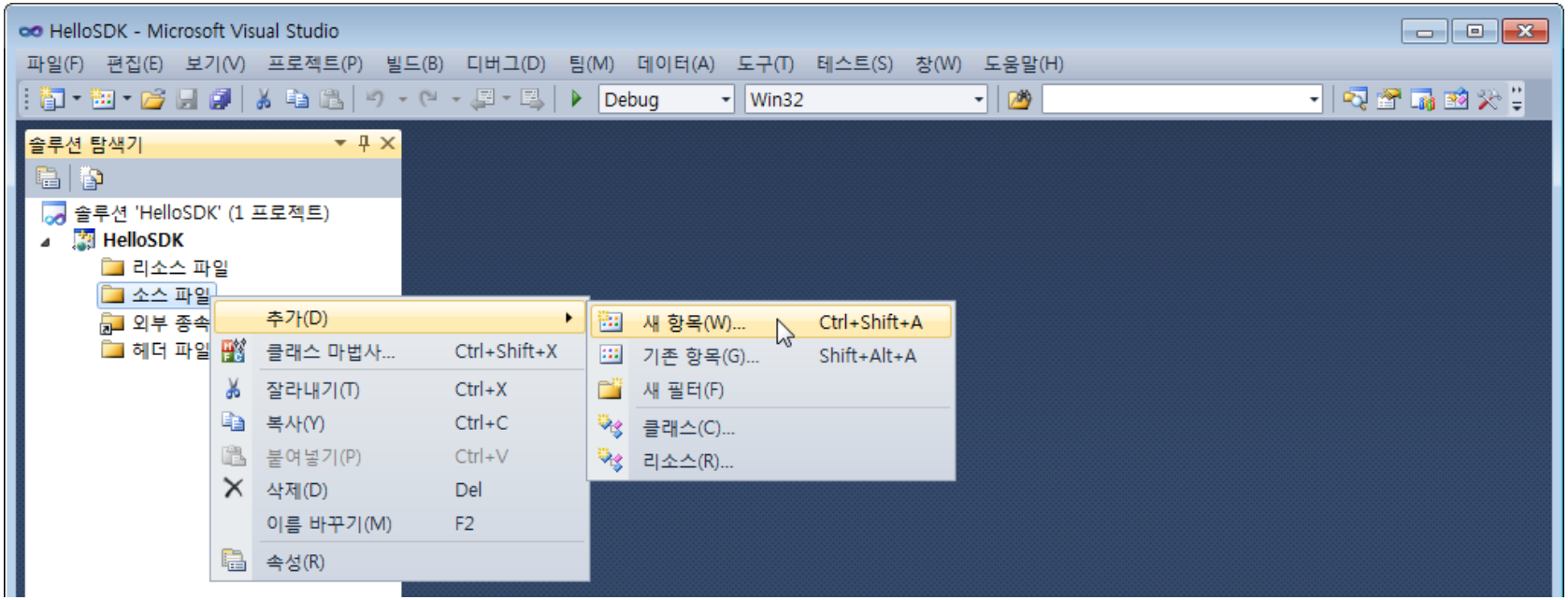
간단한 SDK 프로그램 작성 – HelloSDK

- 프로젝트 옵션 변경



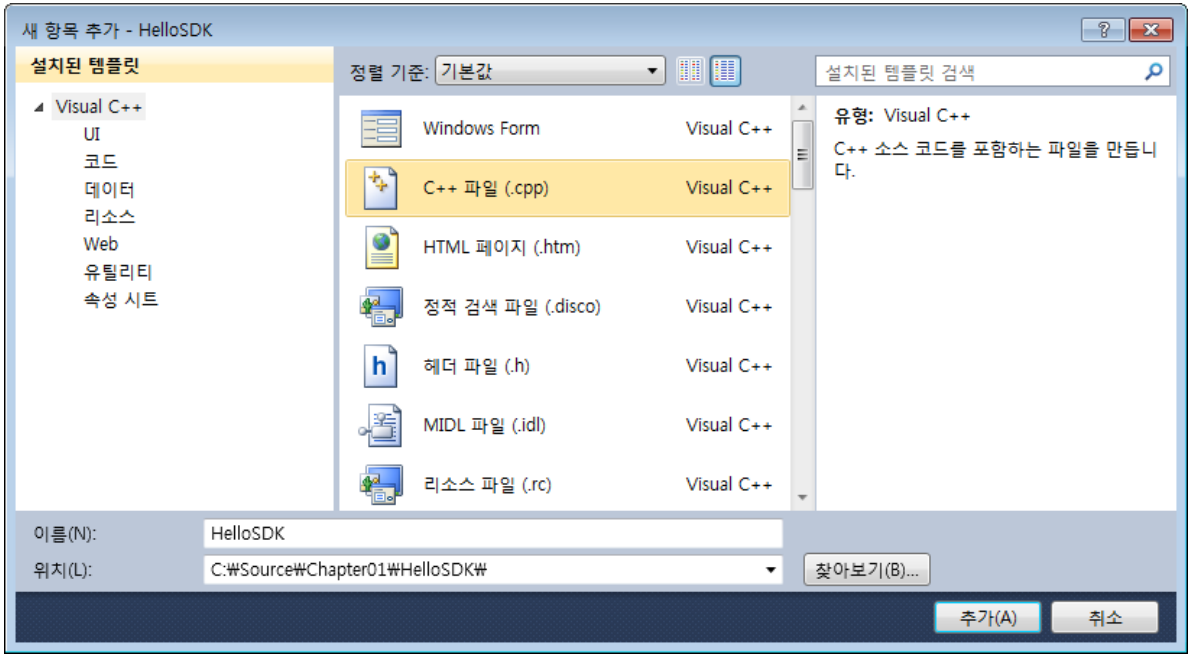
간단한 SDK 프로그램 작성 – HelloSDK

- 소스 파일 추가(1)



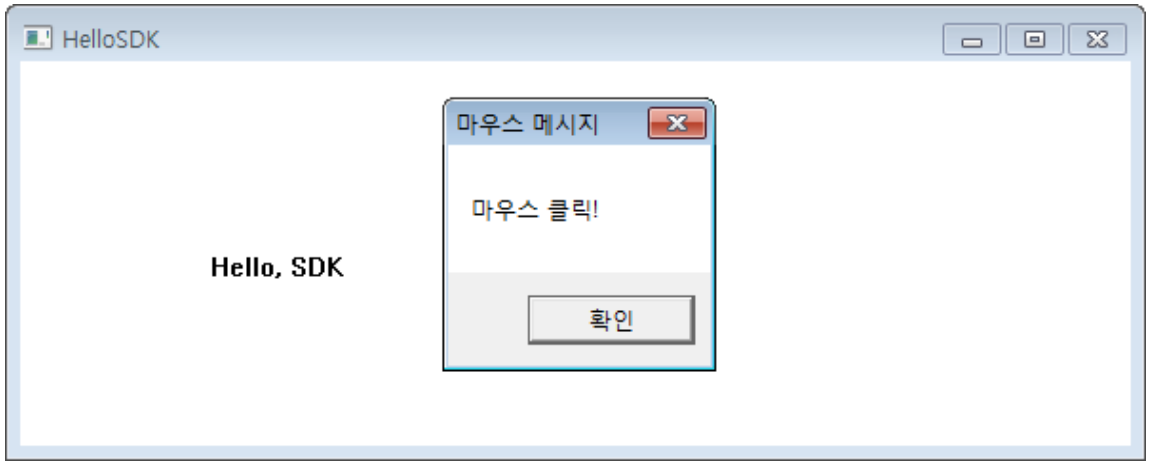
간단한 SDK 프로그램 작성 – HelloSDK

- 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.          ③ 윈도우 클래스
    wndclass.style = CS_HREDRAW | CS_VREDRAW; // 스타일 지정      초기화와 등록
    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
```

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.                    ④ 윈도우 생성
    hwnd = CreateWindow(TEXT("HelloClass“ ), TEXT("HelloSDK“ ),
        WS_OVERLAPPEDWINDOW, CW_USEDEFAULT, CW_USEDEFAULT,
        CW_USEDEFAULT, CW_USEDEFAULT, NULL, NULL, hInstance, NULL);
    ShowWindow(hwnd, nCmdShow);

    // 메시지 큐에서 메시지를 하나씩 꺼내서 처리한다.          ⑤ 메시지 루프
    while(GetMessage(&msg, NULL, 0, 0) > 0){
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }

    return msg.wParam;
}
```

간단한 SDK 프로그램 작성 – HelloSDK

- HelloSDK 예제 코드 (3/4)

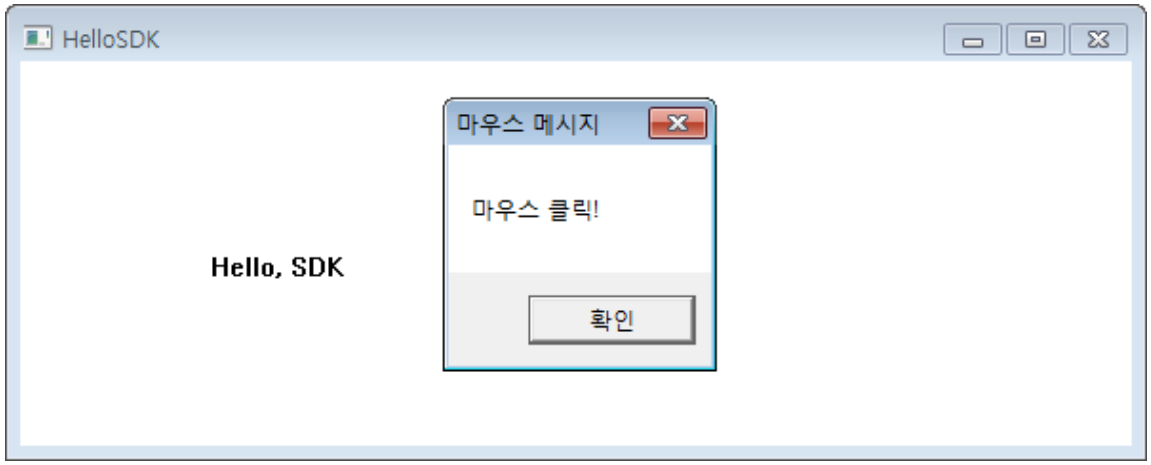
⑥ 윈도우 프로시저

```
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;
```

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);
}
```

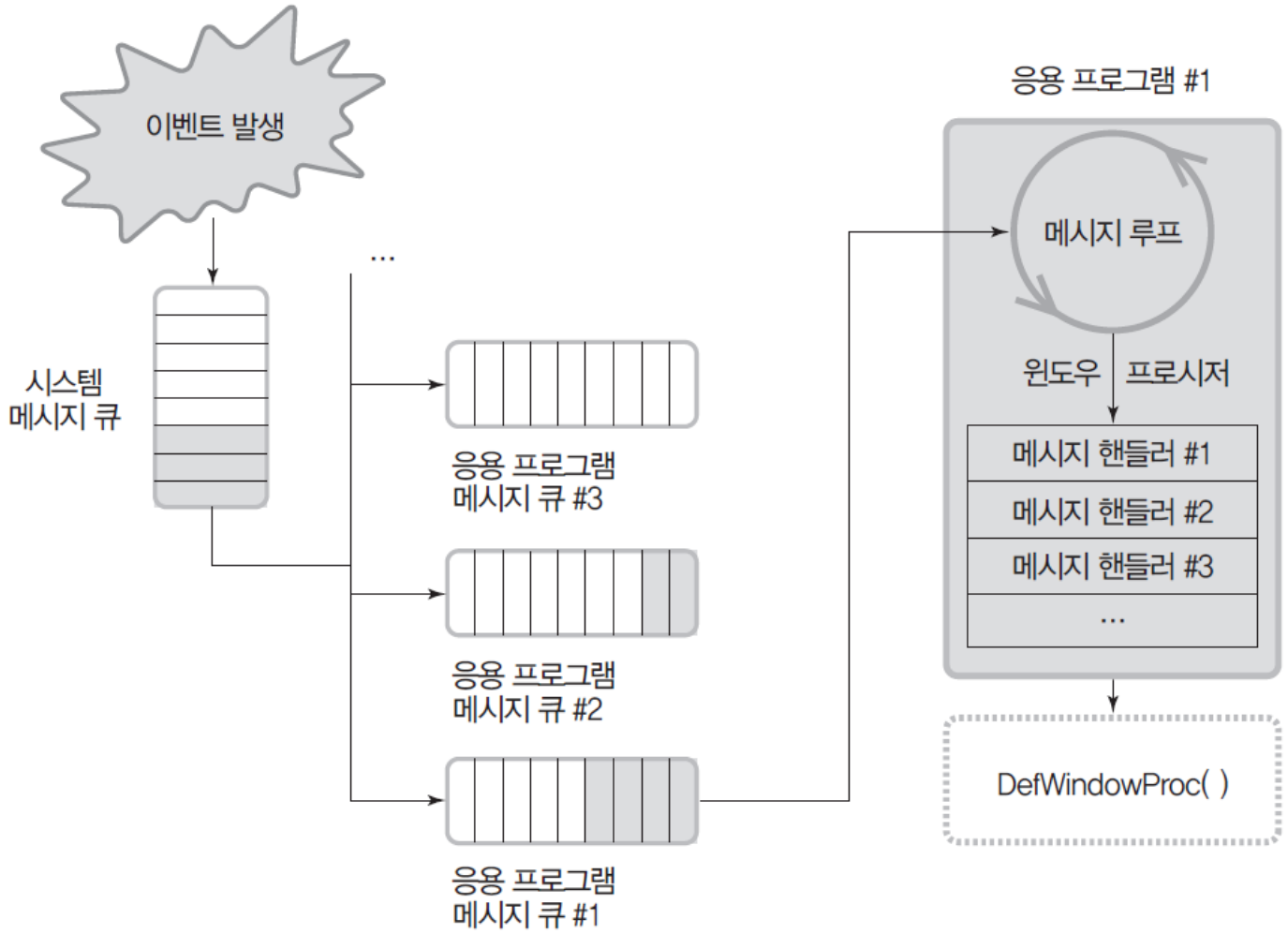


SDK 프로그램 기본 구조

- WM_CREATE
  - CreateWindow() 함수 호출
- WM_LBUTTONDOWN
  - 클라이언트 영역에서 마우스 왼쪽 버튼 누를때
- WM_PAINT
  - 클라이언트 영역에서 일부 또는 전체를 다시 그릴필요가 있을때
- WM_DESTROY
  - 종료 버튼을 눌렀을때

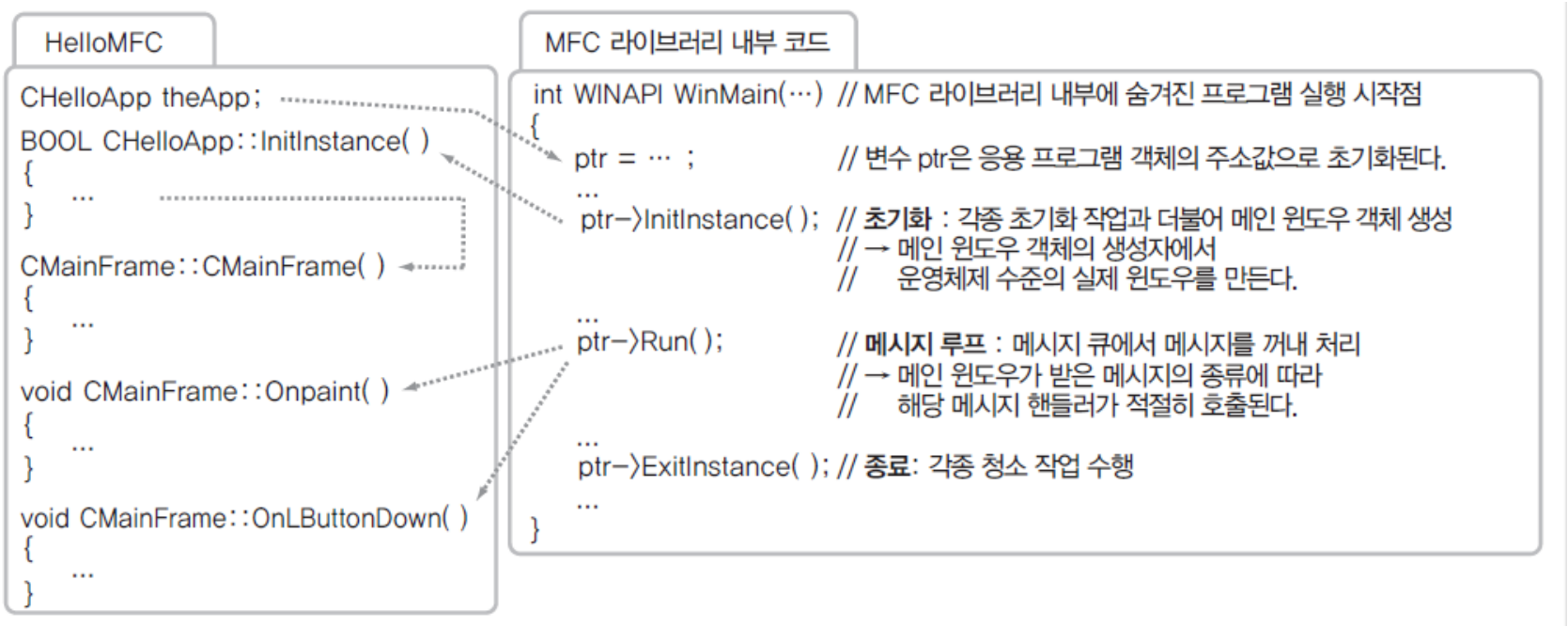
SDK 프로그램 기본 구조

- SDK 프로그램 동작 원리



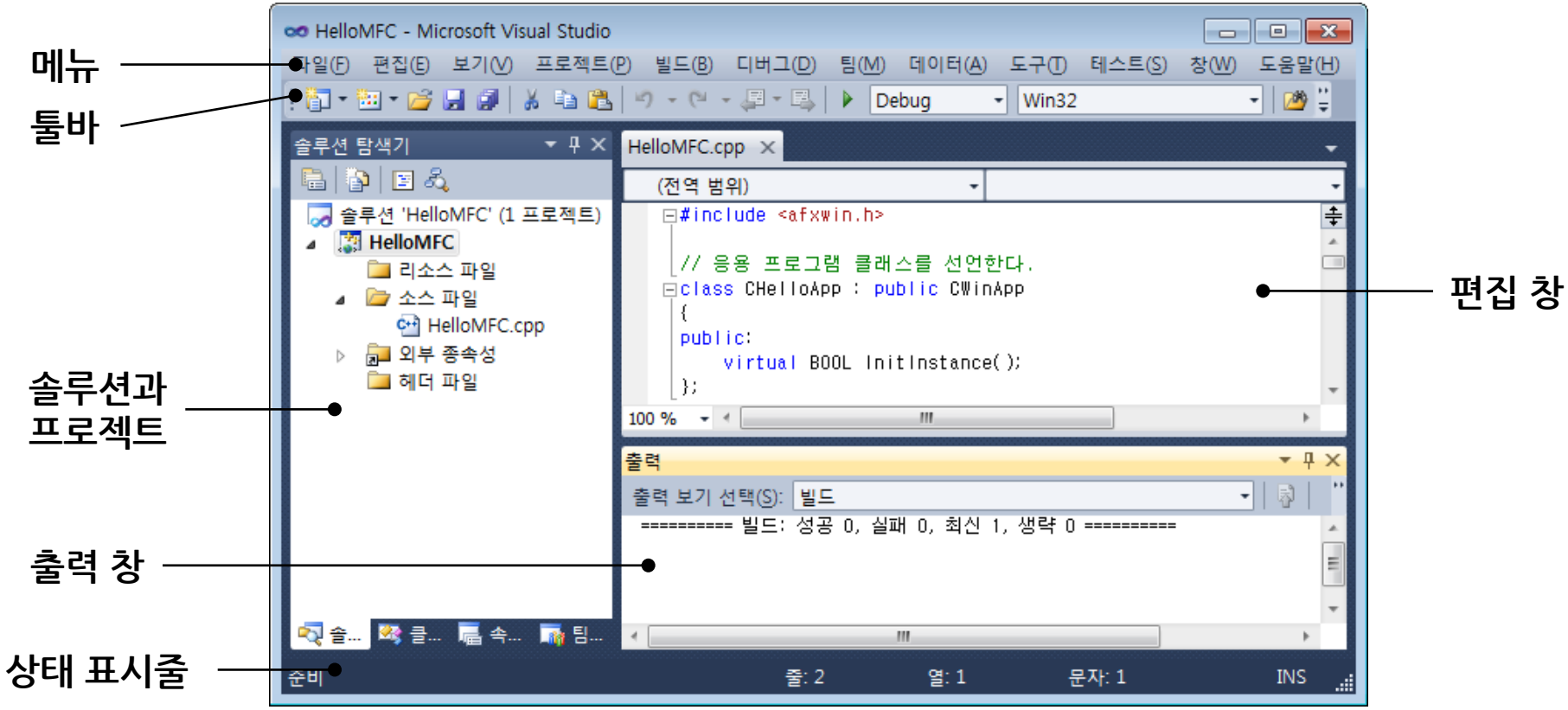
MFC 프로그램 기본 구조

- MFC 프로그램 동작 원리



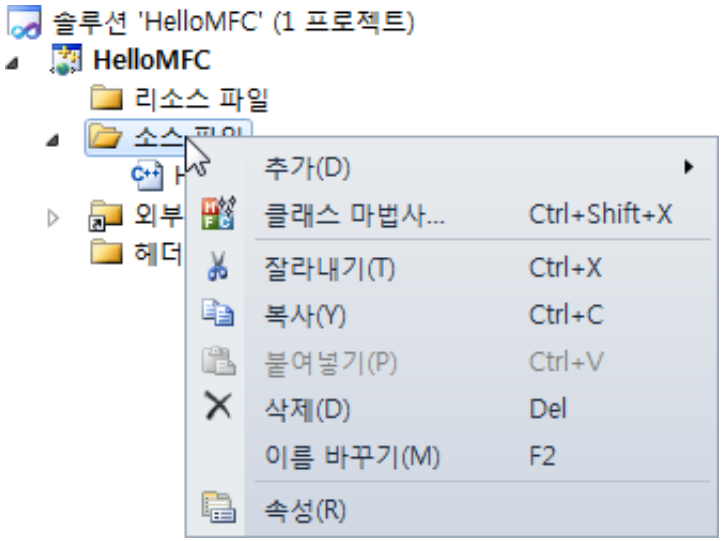
비주얼 C++ 개발 환경

- 비주얼 C++ 통합 개발 환경



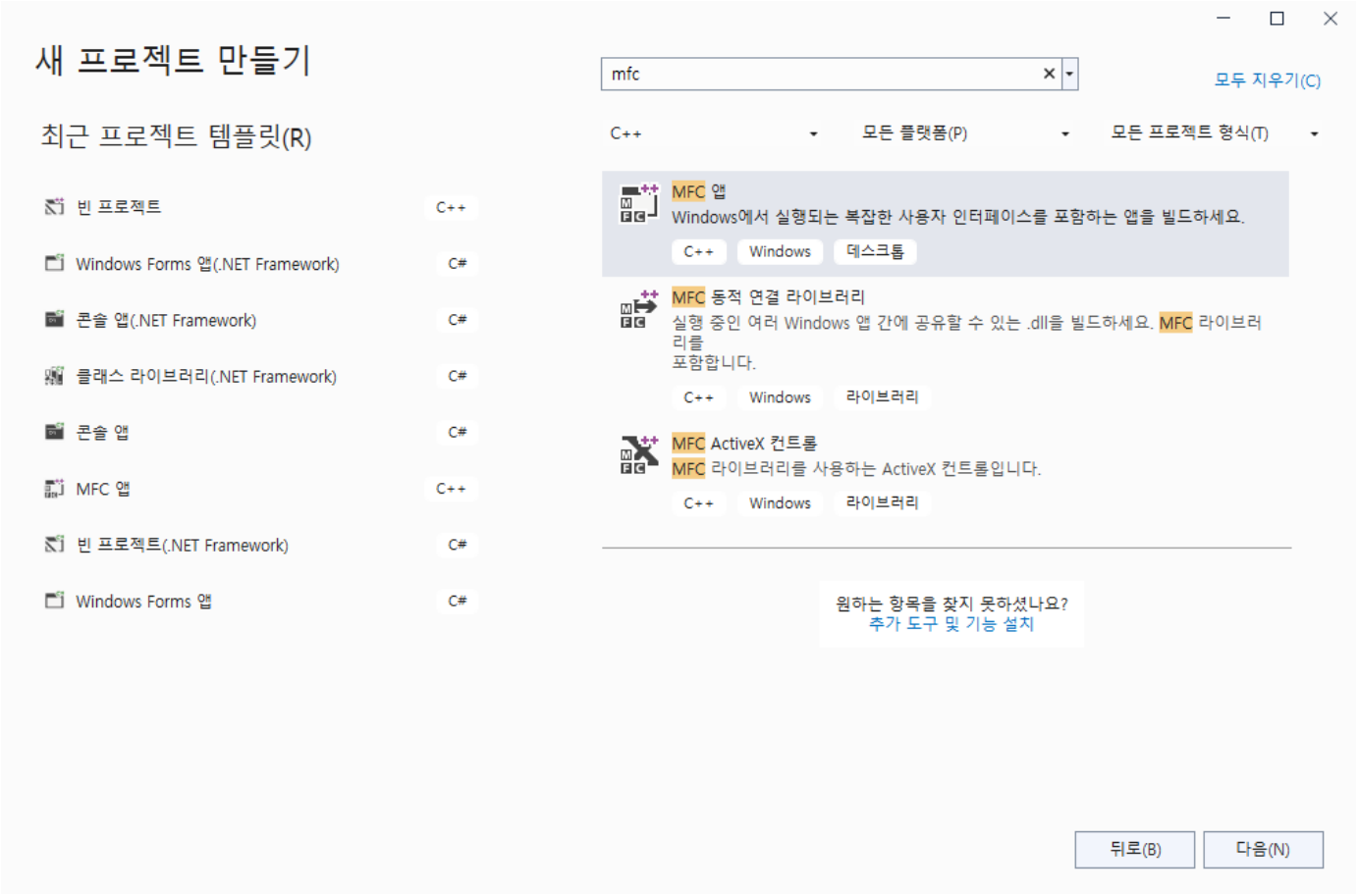
비주얼 C++ 개발 환경

- 팝업 메뉴

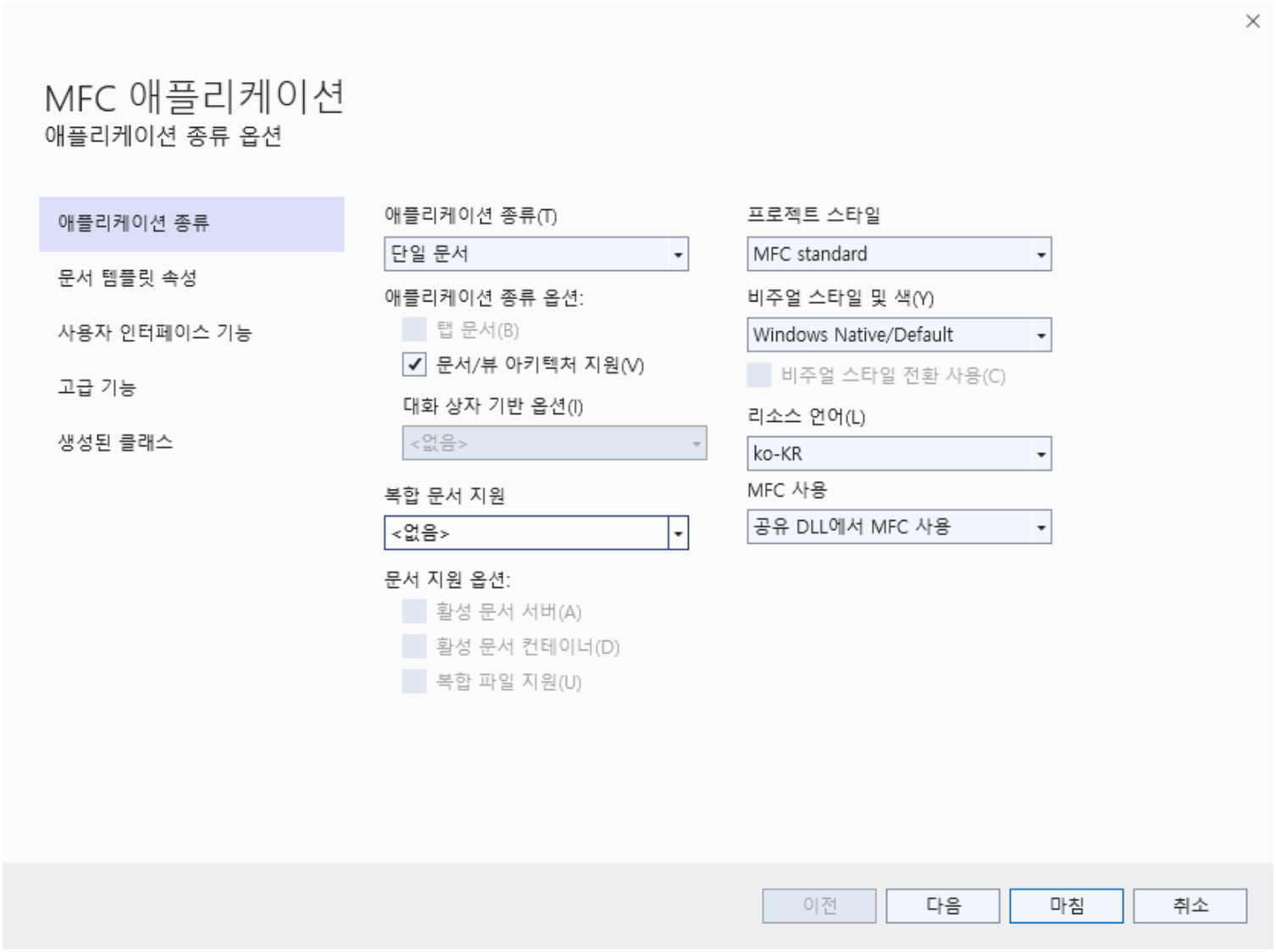


비주얼 C++ 개발 환경

- 응용 프로그램 마법사



비주얼 C++ 개발 환경


MFC 애플리케이션
애플리케이션 종류 옵션
애플리케이션 종류
문서 템플릿 속성
사용자 인터페이스 기능
고급 기능
생성된 클래스
애플리케이션 종류(T)
단일 문서
애플리케이션 종류 옵션:
탭 문서(B)
문서/뷰 아키텍처 지원(V)
대화 상자 기반 옵션(I)
<없음>
복합 문서 지원
<없음>
문서 지원 옵션:
활성 문서 서버(A)
활성 문서 컨테이너(D)
복합 파일 지원(U)
프로젝트 스타일
MFC standard
비주얼 스타일 및 색(Y)
Windows Native/Default
비주얼 스타일 전환 사용(C)
리소스 언어(L)
ko-KR
MFC 사용
공유 DLL에서 MFC 사용
이전
다음
마침
취소

비주얼 C++ 개발 환경

- 속성 창과 MFC 클래스 마법사 (Ctrl+Shift+X)