# WINAPI
# -CHAPTER 1-

SOULSEEK

# 목차

# 1. WINAPI의 특징

# 1. WINAPI의 특징

## Windows특징과 장점

1. 그래픽 기반(GUI)의 운영체제이다.
2. 멀티 태스킹이 가능하다.
3. 메시지 구동 시스템이다.
4. 장치에 독립적이고 일관성이 있다.
5. 리소스가 따로 분리되어 있다.

## WinAPI를 배우는 이유

1. **Windows OS**를 이해하기 위해.
2. **Class Library**의 이해를 위해.
3. **Library**와 **Visual tool, Engine**들의 사용의 이해를 위해.

# 1. WINAPI의 특징

**변수 명명(헝가리안 표기법)**

- 무조건 적으로 그렇게 사용하자가 아니다.
- 해당 변수의 자료형에 맞게 사용 할 수 있도록 이름만 보고파악 할 수 있게 만들기 위한 방법
- 현재 MS에서도 권장하지 않는 방법이지만 API에서는 사용하고 있다.

| 변수형 | 표현법 |
|---|---|
| bool | b (ex : bool bMove) |
| int | i,n (ex : int nNum) |
| float | f (ex : float fNum) |
| string | str (ex : string strName) |
| char(정수) | ch (ex : char chNum) |
| char(문자열) | sz(ex : char szName[10]) |
| handle | h(ex : HWND hWnd) |
| 포인터 | p(ex : int* ipNum) |
| 배열 | Arr(ex : int iarrNum[10]) |
| 전역 변수 | g_(ex : int g_nNum) |
| 맴버 변수 | m_(ex : int m_nNum) |

**BYTE, CHAR, WORD, DWORD, LONG, BOOL, NULL** : **WINAPI**에서 사용하는 표현법 들인데 기본적으로 해당 원형을 쉽게 표현한 것들이다.

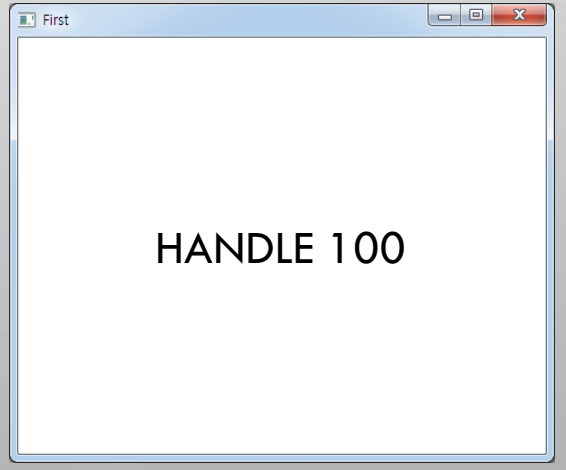
주의사항 – **NULL**의 경우 **VS2017**버전부터 **nullstr**를 권장하고 있다 인자로 쓰이는 경우 제외하고 **NULL**로 대입하는 경우에는 에러가 나는 경우가 있기 때문에 주의하자.

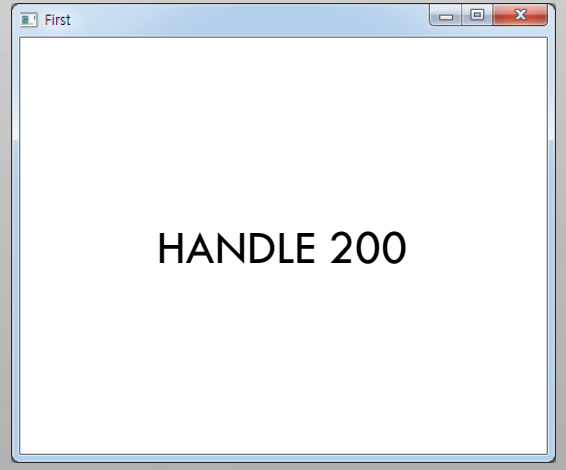
**EX)HANDLE hWnd = NULL -> nullprt**로 변경

# 1. WINAPI의 특징

## HANDLE

- 여러가지 응용프로그램들과 기능들의 메모리에 번호를 붙여 서로를 구분하기 위해 붙여진 정수 값 이다.
- 32비트 정수 값을 가진다.
- OS에서 발급받은 넘버를 사용하기만 한다.
- 같은 종류의 핸들과는 동일 값을 가지지 않는다.
- 멀티 태스킹을 위해 서로를 구분하기 위한 표식을 하는 숫자일 뿐 특정 값을 가지는 것이 아니므로 사용 법에 집중하자.

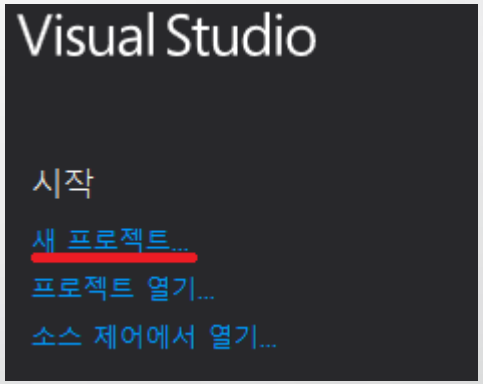


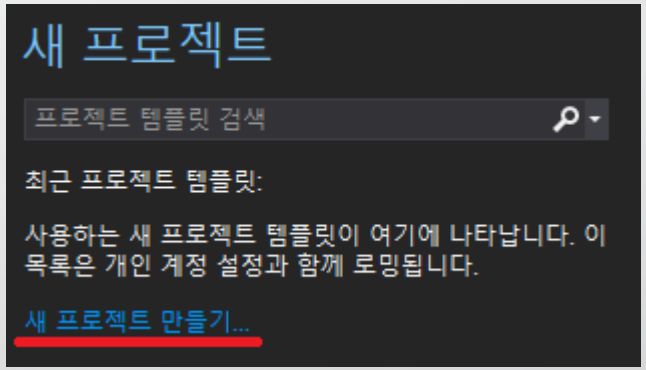



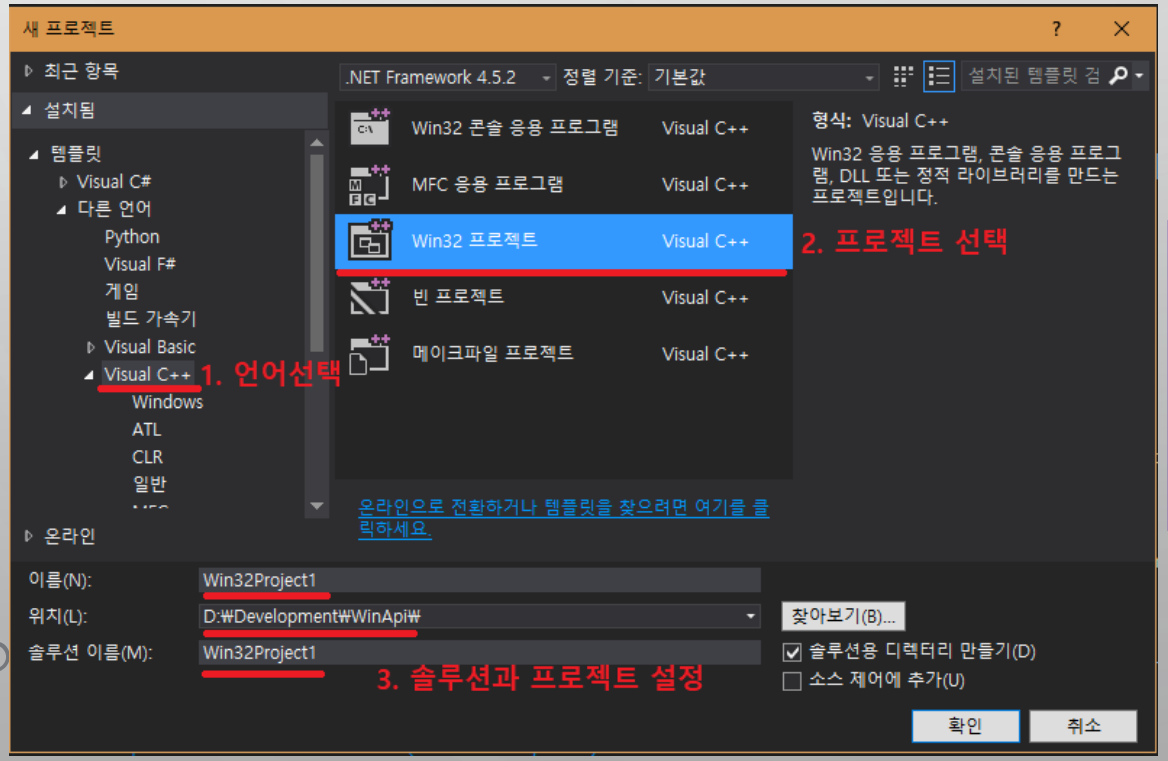
# 2. CREATE PROJECT

# 2. CREATE PROJECT

- Visual Studio 2015 or 2017을 실행한다.

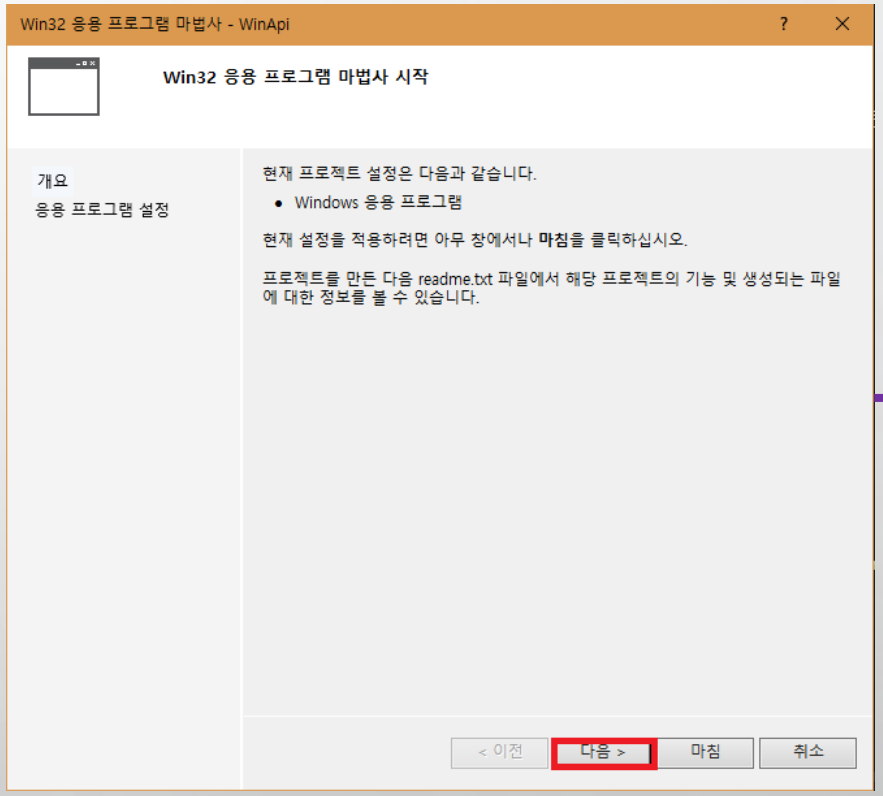


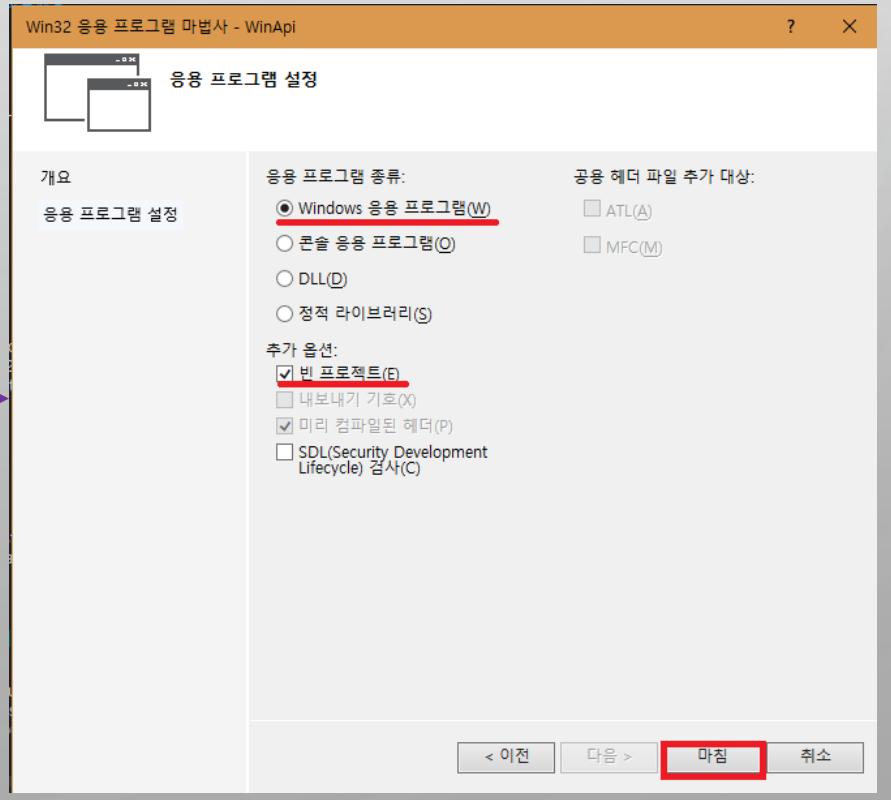



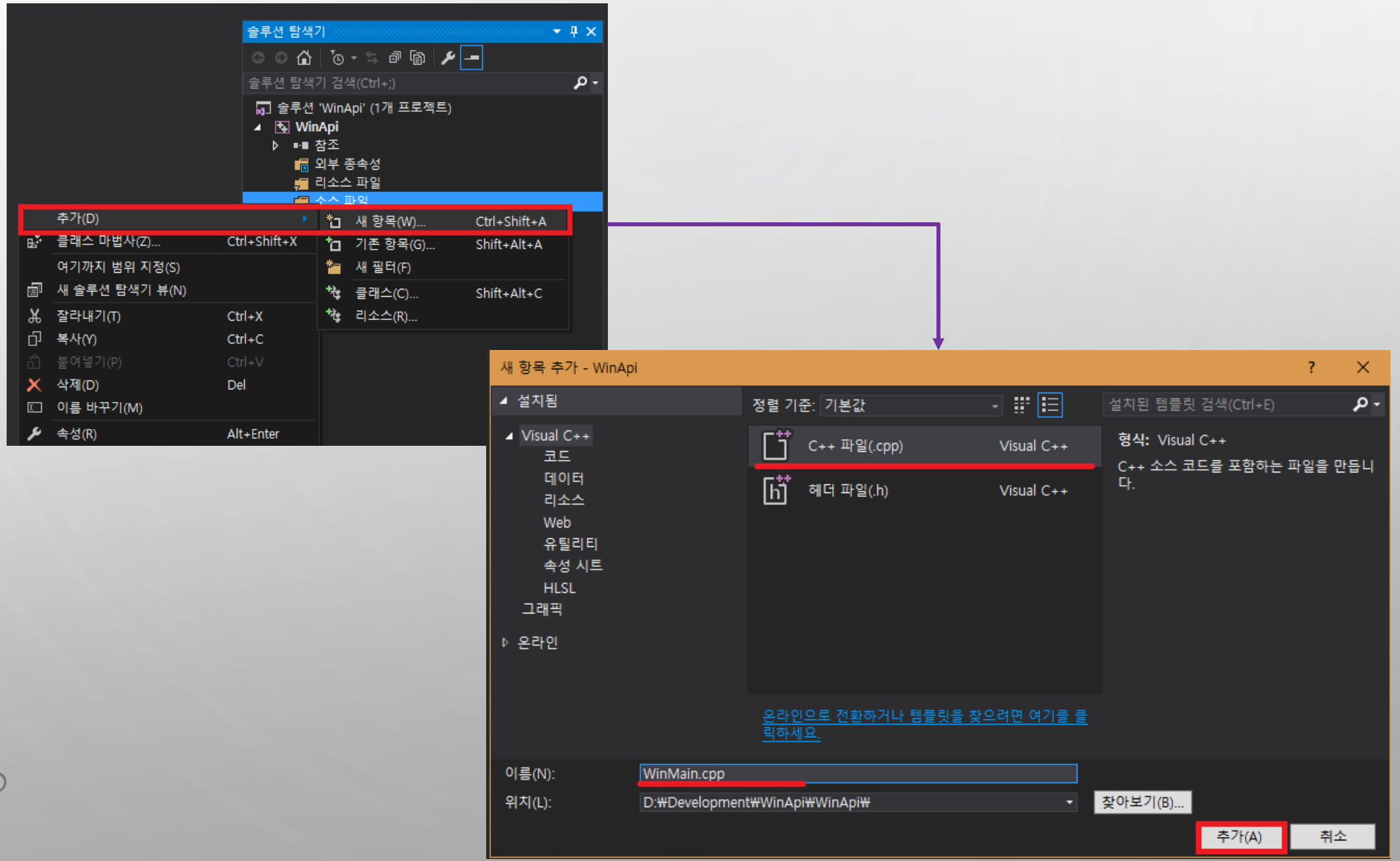

1. C++로 언어를 설정한다.
2. Win32 프로젝트를 설정한다.
3. 솔루션 경로와 프로젝트를 설정한다.
   - 솔루션은 프로젝트들의 그룹이라고 생각하면 된다.

# 2. CREATE PROJECT

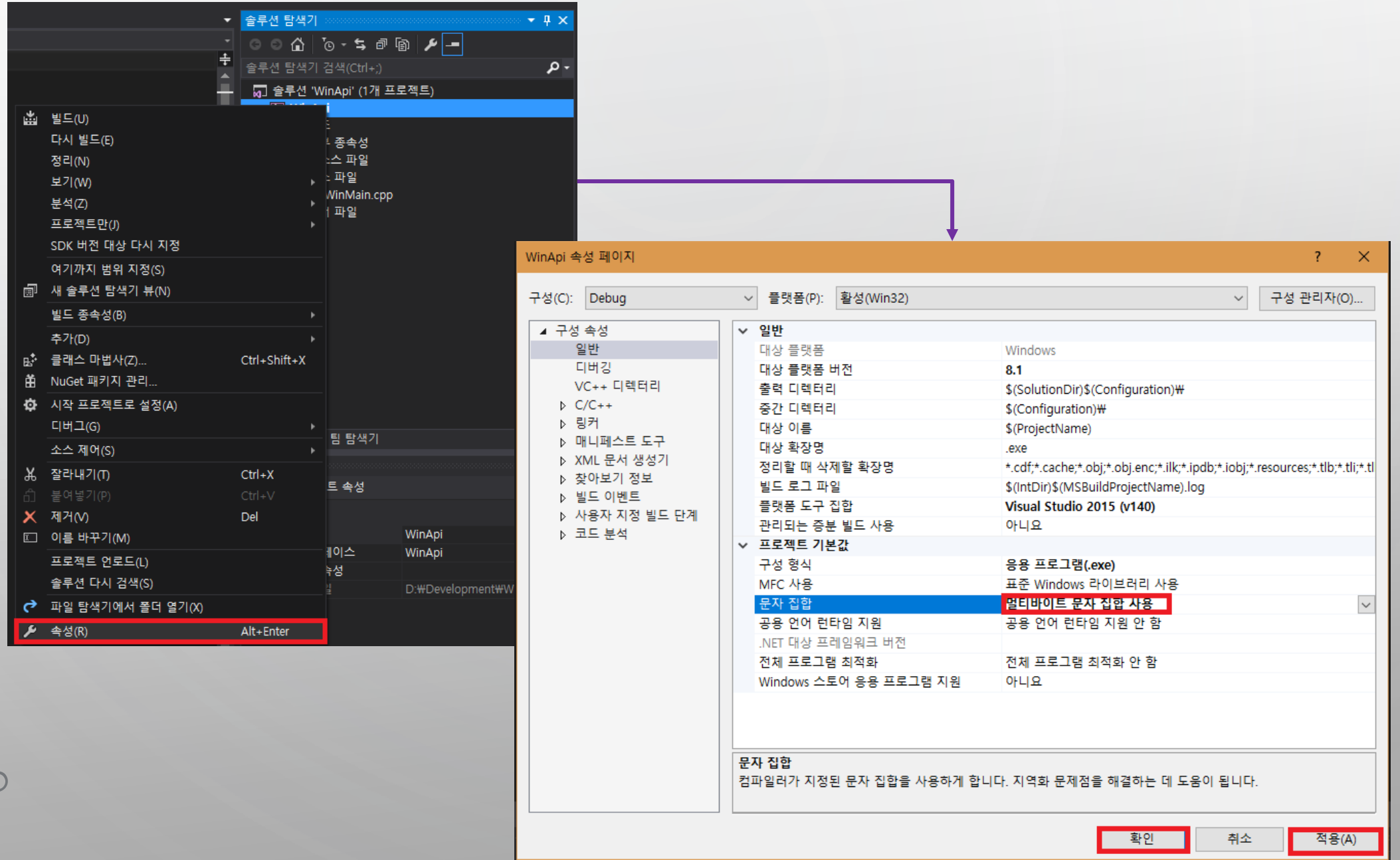


# 2. CREATE PROJECT



# 2. CREATE PROJECT



# 3. HELLO WORLD WINMAIN

# 3. HELLO WORLD WINMAIN

```
#include<Windows.h>
LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
HINSTANCE g_hInst;//글로벌 인스턴스핸들값
LPCTSTR lpszClass = TEXT("HelloWorld"); //클래스 명 : 창이름

int APIENTRY WinMain(HINSTANCE hInstance,HINSTANCE hPervInstance,LPSTR lpszCmdParam, int nCmdShow)
{
	HWND hWnd;
	MSG Message;
	WNDCLASS WndClass;
	g_hInst = hInstance;

	//WndClass는 기본 윈도우환경을 만드는 구조체다. 맴버변수는 밑에와 같다.
	WndClass.cbClsExtra = 0; //예약영역
	WndClass.cbWndExtra = 0; //예약영역 (신경x)
	WndClass.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);	//배경색
	WndClass.hCursor = LoadCursor(NULL, IDC_ARROW);	//커서
	WndClass.hIcon = LoadIcon(NULL, IDI_APPLICATION);	//아이콘 모양
	WndClass.hInstance = hInstance; //(프로그램 핸들값(번호)등록)
	WndClass.lpfnWndProc = WndProc;	//프로세스 함수 호출
	WndClass.lpszClassName = lpszClass;	//클레스 이름
	WndClass.lpszMenuName = NULL;
	WndClass.style = CS_HREDRAW | CS_VREDRAW; //윈도우의 수직과 수평이 변경 시 다시 그린다.
	RegisterClass(&WndClass);  //만들어진 WidClass를 등록

	hWnd = CreateWindow(lpszClass, lpszClass, WS_OVERLAPPEDWINDOW, CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT,NULL,
			(HMENU)NULL, hInstance, NULL);

	ShowWindow(hWnd, nCmdShow);

	while (GetMessage(&Message,NULL,0,0)) //사용자에게 메시지를 받아오는 함수(WM_QUIT 메시지 받을 시 종료), 메시지가 오기까지 대기를 하고있다가 오면 작동.
	{
		TranslateMessage(&Message); //  키보드 입력 메시지 처리함수
		DispatchMessage(&Message); //받은 메시지를 WndProc에 전달하는 함수
	}
	return (int)Message.wParam;
}

LRESULT CALLBACK WndProc(HWND hWnd, UINT iMessage, WPARAM wParam, LPARAM lParam)
{
	switch (iMessage)
	{
	case WM_DESTROY : // 윈도우가 파괴되었다는 메세지
		PostQuitMessage(0); //GetMessage함수에 WM_QUIT 메시지를 보낸다.
		return 0; //WndProc의 Switch는 break 대신 return 0; 를 쓴다.
	}
	return(DefWindowProc(hWnd, iMessage, wParam, lParam)); // case에 있는 메시지를 제외한 나머지 메시지를 처리한다.
}
```

Win32에서 항상 NULL 오래된 구버전과의 호환을 위한 것

윈도우 실행 형태

프로그램 인스턴스의 핸들

명령행으로 입력된 프로그램 인수 값(보통 실행직후에 열 파일의 경로)

기본 프레임 **WINMAIN**으로 어떤 구성이 되어 있는지 살펴보고 익히세요.