

# 6장 대화상자와 컨트롤

2022년도 1학기 윈도우 프로그래밍

- **학습목표**

- 대화상자를 만들고 사용할 수 있다.
- 컨트롤 종류를 알고 각 컨트롤을 사용할 수 있다.
- 다양한 컨트롤을 이용해 응용 프로그램을 개발할 수 있다.

- **내용**

- 대화상자 만들기
- 버튼 컨트롤
- 에디트 박스
- 체크버튼과 라디오버튼
- 콤보박스
- 리스트박스
- 모달리스 대화상자

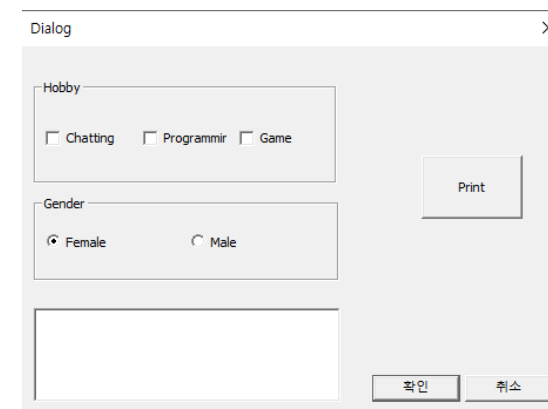
# 1. 대화상자 사용하기

## • 대화상자 (Dialog Box)

- 프로그램 수행 중 사용자와 간단한 입력/출력을 하기 위해 사용되는 윈도우
- 많은 양의 정보를 효율적으로 입/출력해주는 매개체. 혹은 말 그대로 사용자와 대화하는 상자
- 이 대화상자에서 사용하는 도구들: 컨트롤
  - 컨트롤은 사용자로부터 입력을 받거나 사용자에게 정보를 제공하기 위해 사용
  - 대표적 컨트롤: 버튼 컨트롤, 에디트 컨트롤, 콤보 박스 컨트롤, 리스트 컨트롤 등

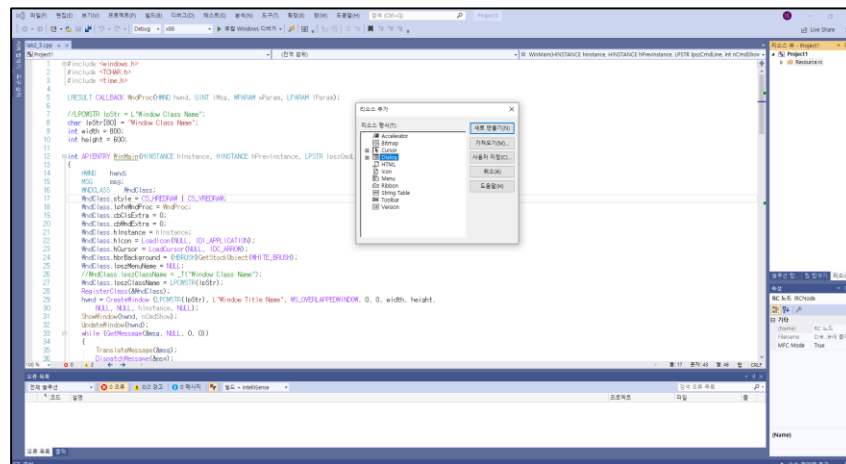
## • 사용방법

1. 리소스에서 새로운 대화상자 만들기
  - 리소스 형태로 대화상자를 만들고, 대화상자 편집기로 컨트롤들을 디자인한다.
2. 대화상자 띄우기
  - 대화상자를 메인 윈도우에서 띄운다.
3. 대화상자에 대한 메시지 처리 콜백 함수 **DialogProc()** 작성
  - 별도의 콜백함수를 가지고 대화상자 메시지 처리

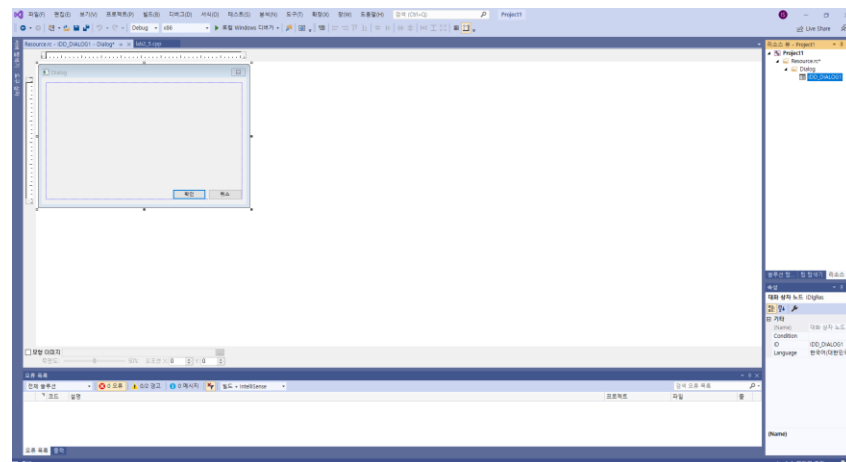


# 1) 대화상자 만들기

## 1) 리소스에서 대화상자 만들기

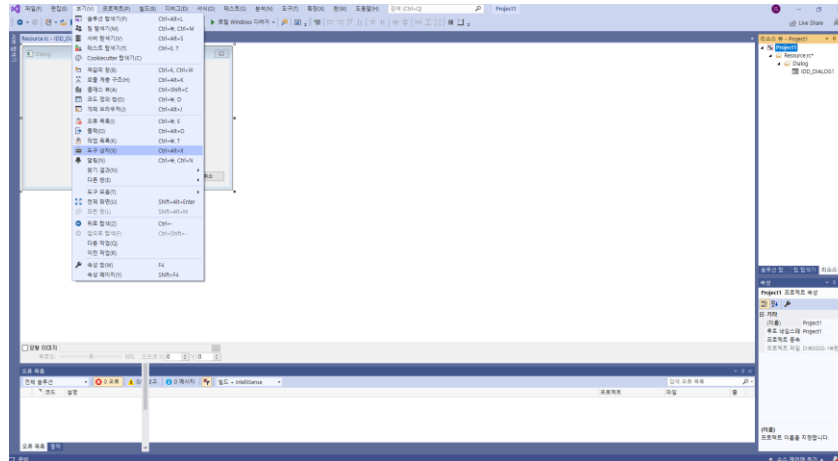


리소스에서 대화상자 추가하기

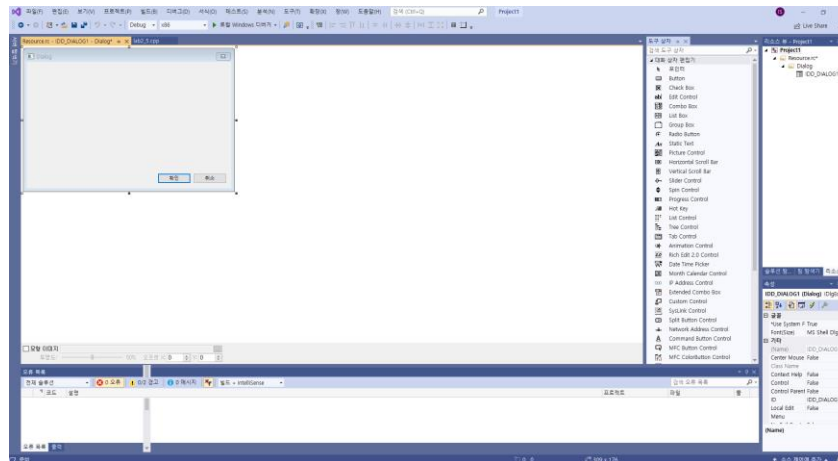


대화상자를 추가하면 리소스에  
IDD\_DIALOG1이 추가된 것을  
확인

# 대화상자 만들기

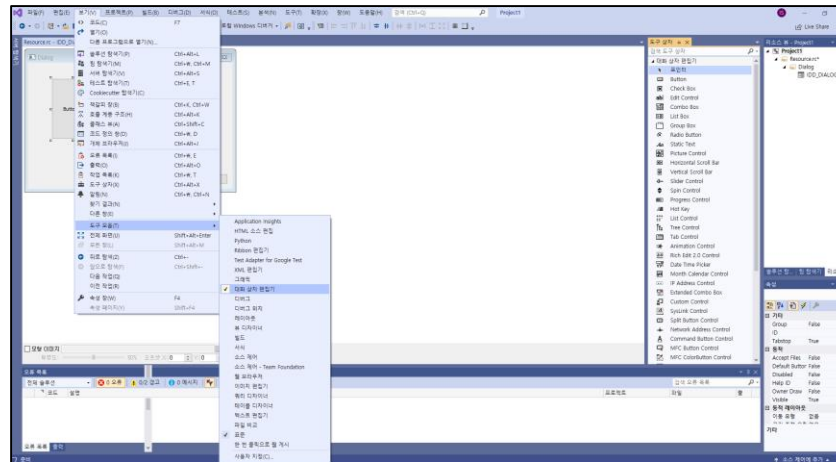


메뉴: 보기 → 도구상자  
사용 가능한 컨트롤들 확인

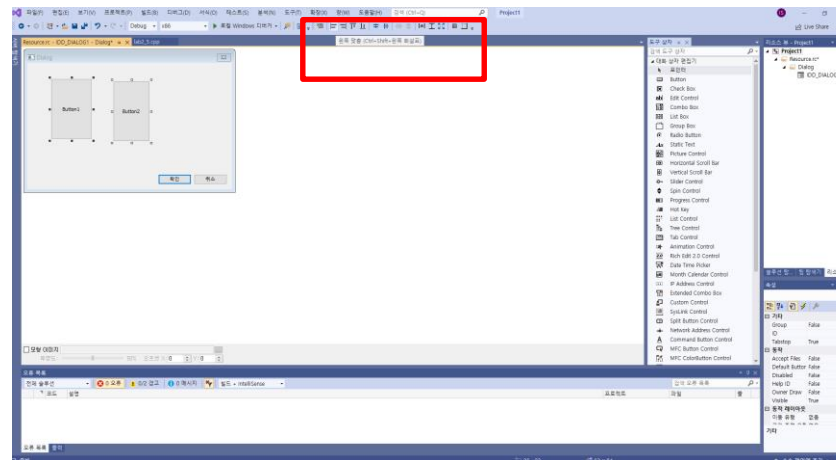


필요한 컨트롤을 선택한 후  
대화상자에 추가

# 대화상자 만들기

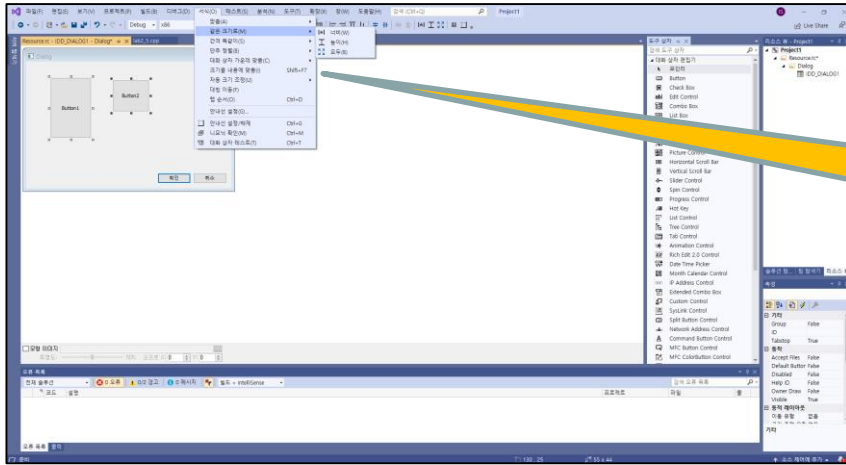


메뉴: 보기 → 도구모음 →  
대화상자 편집기 체크

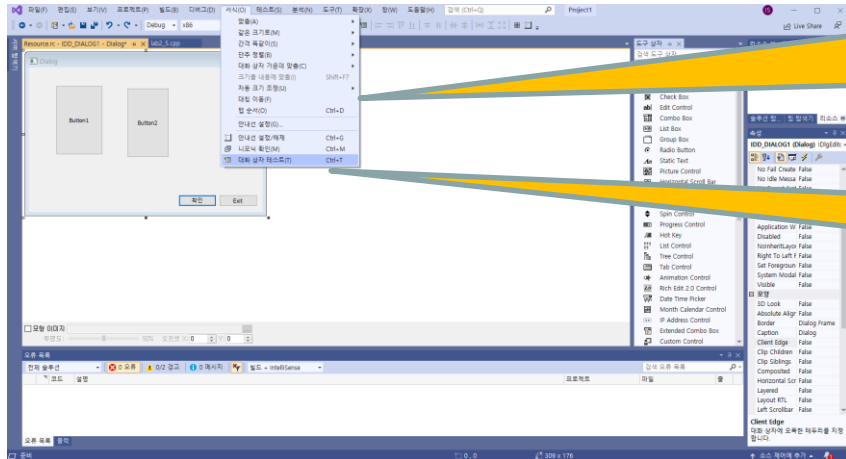


도구상자를 편집할 수 있다.

# 대화상자 만들기

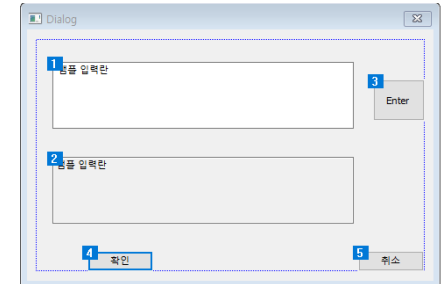
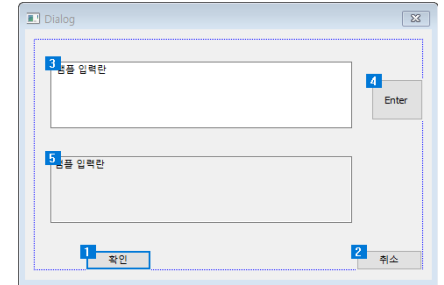


**메뉴: 서식  
도구상자의 서식을 편집할 수 있다.**

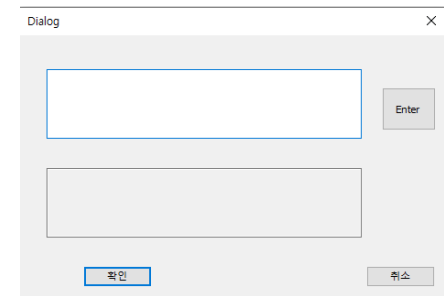


**메뉴: 서식 → 탭순서  
대화상자에서 탭을 이용하여 순서대로  
이동할 수 있는 순서를 설정해준다.  
마우스로 순서대로 클릭하여 순서를  
설정해준다.**

**메뉴: 서식 → 대화상자 테스트  
대화상자를 미리 테스트해볼 수 있다.**



**탭순서 실행한 결과**



**대화상자 테스트한 결과**

## 2) 대화상자 띄우기, 종료하기 함수

### 2) 대화상자 띄우기

- 리소스에서 대화상자를 만든 후, 함수 호출을 하여 대화상자를 화면에 띄운다.

**DialogBox** (hInstance, MAKEINTRESOURCE(IDD\_DIALOG1), hWnd, lpDialogFunc)

- 대화상자를 생성하고 **WM\_INITDIALOG** 메시지를 대화상자 프로시저로 보냄

```
int DialogBox (HINSTANCE hInstance, LPCTSTR lpTemplate, HWND hWnd, LGPROC lpDialogFunc );
```

- 대화상자를 생성하고 **WM\_INITDIALOG** 메시지를 대화상자 프로시저로 보냄
  - 리턴값: IDOK 메시지
  - hInstance : 응용의 프로그램 인스턴스 값
  - lpTemplate : 대화상자의 ID
  - hWnd: 윈도우의 핸들 값
  - lpDialogFunc : 대화상자에서 발생하는 메시지 처리용 다이얼로그 함수

- 대화상자 종료하기

**EndDialog**(hDlg, 0);

```
BOOL EndDialog (HWND hDlg, int nResult);
```

- hDlg: 종료할 대화상자 핸들
- nResult : 0 (대화상자 종료상태 표시)



### 3) 대화상자의 메시지 처리하기

#### 3) 메시지 처리 함수: 대화상자 프로시저

- 대화상자 내에서 발생하는 메시지들을 처리하는 함수로 대화상자 프로시저를 생성  
BOOL CALLBACK DialogProc (HWND hDlg, UINT iMessage, WPARAM wParam, LPARAM lParam)
- **BOOL 형을 반환: 메시지를 처리했으면 TRUE를 리턴, 그렇지 않으면 FALSE를 리턴**
  - DefWindowProc 함수로 리턴하지 않고 TRUE 또는 FALSE의 Boolean 값을 리턴한다.
- 메시지 처리:
  - 컨트롤에서 대화상자로 오는 메시지: WM\_COMMAND
    - HIWORD (wParam): 통지 코드
    - LOWORD (wParam): 메시지를 보낸 컨트롤의 ID

메시지를 보낸 곳	wParam		lParam
	HIWORD	LOWORD	
컨트롤	컨트롤에 따른 통지정보	컨트롤 ID	컨트롤 핸들값



# 대화상자의 메시지 처리하기

- 윈도우에 메시지를 보내는 함수: **SendMessage**

- 부모 윈도우와 차일드 컨트롤 간의 통신, 윈도우 간 데이터 전송 및 처리를 위한 통신
  - 부모 윈도우가 차일드 컨트롤에게 명령을 내리거나 상태를 조사할 때 사용
- 메시지를 메시지 큐에 넣지 않고 바로 윈도우 프로시저로 보내 처리
  - 메시지가 처리되기 전까지 반환되지 않음
  - 윈도우 프로시저가 값을 반환해야만 SendMessage도 반환하여 끝마칠 수 있음

```
LRESULT SendMessage (HWND hWnd, UINT iMsg, WPARAM wParam, LPARAM lParam);
```

- hWnd: 메시지를 전달받을 윈도우 핸들
- iMsg : 전달할 메시지
- wParam, lParam: 메시지의 추가적 정보, 메시지에 따라 다른 정보 반환

- 대화상자를 만들었을 때 발생 메시지: **WM\_INITDIALOG**

- 윈도우 프로시저의 WM\_CREATE 메시지 의미.
- 대화 상자에 필요한 초기화 작업
  - wParam: 대화상자에서 제일 먼저 키보드 입력을 받을 컨트롤의 핸들값
  - lParam: 부가적인 정보를 저장하는데 일반적으로 0의 값을 가짐

# 대화상자 띄우기

- 사용 예) 리소스에서 대화상자를 만든 후, 왼쪽 마우스 버튼을 클릭하면 2개 버튼 있는 대화상자 띄우기

```
#include <windows.h>
#include <TCHAR.h>
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM);
BOOL CALLBACK Dialog_Proc (HWND, UINT, WPARAM, LPARAM);

HINSTANCE g_hInst;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpszCmdLine, int nCmdShow)
{
    HWND      hwnd;
    MSG       msg;
    WNDCLASS  WndClass;

    g_hInst = hInstance;
    ...
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
    switch (iMsg) {
        case WM_CREATE :
            break; ;

        case WM_LBUTTONDOWN :    //--- 마우스 클릭하면 대화상자 띄우기
            DialogBox (g_hInst, MAKEINTRESOURCE(IDD_DIALOG1), hwnd, Dialog_Proc);
            break;
    }
    return DefWindowProc (hwnd, iMsg, wParam, lParam) ;
}
```

# 대화상자 띄우기

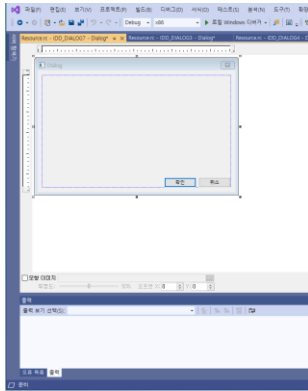
```
//--- 대화상자 메시지 처리함수
BOOL CALLBACK DIALOG_Proc (HWND hDlg, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
    switch(iMsg){

    case WM_INITDIALOG:
        break;

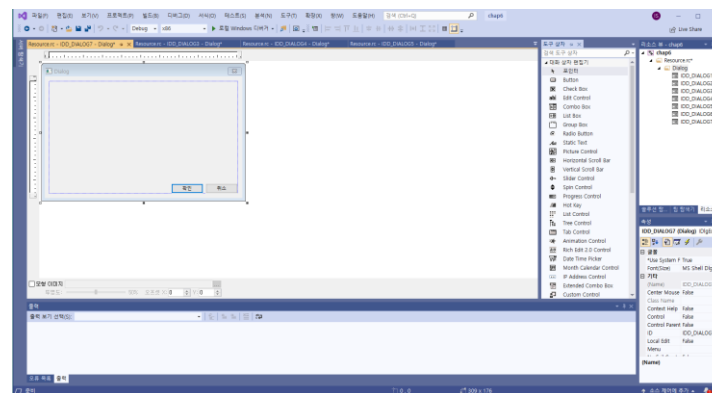
    case WM_COMMAND:
        switch (LOWORD(wParam)) {
            case IDOK:          //--- 버튼
                MessageBox (hDlg, L"test", L"test, ", MB_OK);
                break;

            case IDCANCEL:      //--- 버튼
                EndDialog(hDlg,0);
                break;
        }
        break;

    case WM_CLOSE:
        EndDialog(hDlg,0);
        break;
    }
    return 0;
}
```



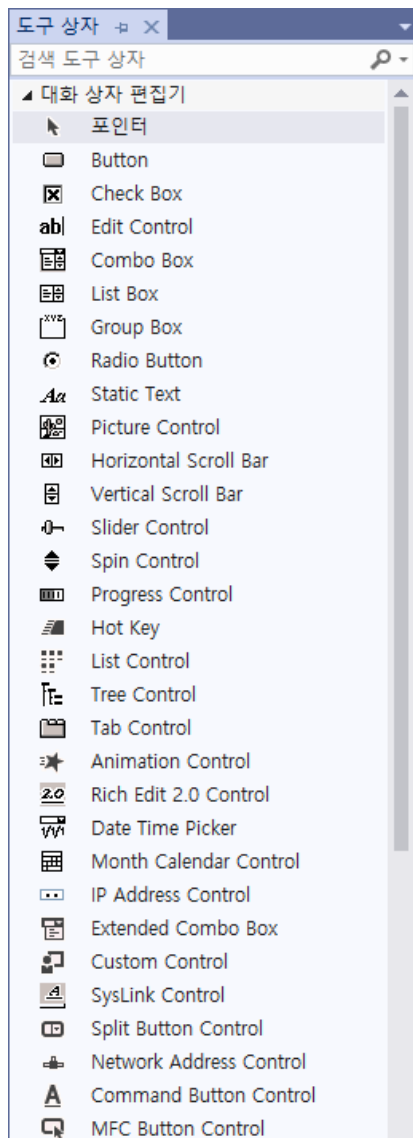
The screenshot shows a Windows application window titled 'Dialog'. Inside the window, there is a smaller dialog box with a title bar that says 'Dialog'. This inner dialog box has a single button labeled '확인' (OK) at the bottom right. The outer window has a standard Windows interface with a menu bar and a toolbar.



리소스에 처음 생성하면  
나타나는 대화상자 편집창

실행 시  
나타나는 대화상자

## 2. 컨트롤 종류

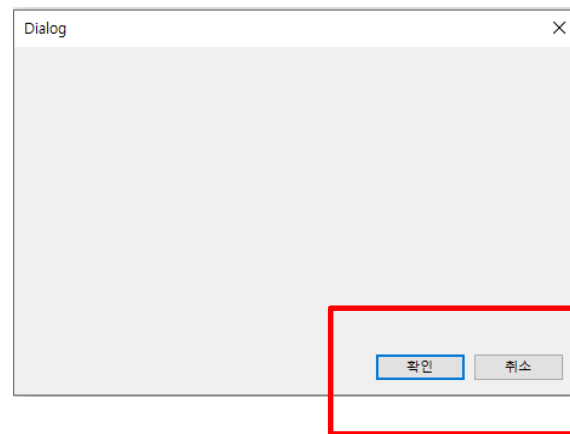
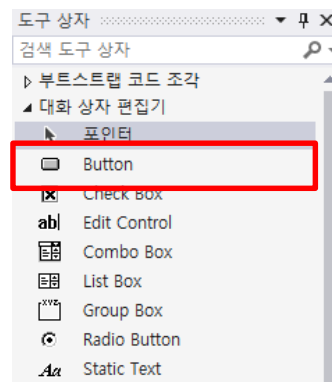


컨트롤	설명
Static Text	정적 텍스트로 보는 것만 가능하고 입력을 할 수 없음
Edit Box	텍스트 입출력을 위한 용도로 사용
Group Box	다른 컨트롤을 묶어 그룹 짓는 역할
Push Button	버튼을 클릭할 때 특정한 함수를 수행하게 할 때 사용
Check Box	특정한 기능을 선택하는 옵션에 사용
Radio Button	그룹 중에서 하나만 선택할 때 사용
List Box	리스트 박스는 여러 항목을 갖는 문자열 정보를 항목별로 보여주는 출력용 컨트롤
Combo Box	콤보 박스는 데이터를 입력할 때 목록에서 하나를 선택하게 할 때 사용

# 1) 버튼 컨트롤

## • 버튼 (Button)

- 버튼을 눌러 임의의 작업이 이루어진다.
- 명령을 받아들이는 역할



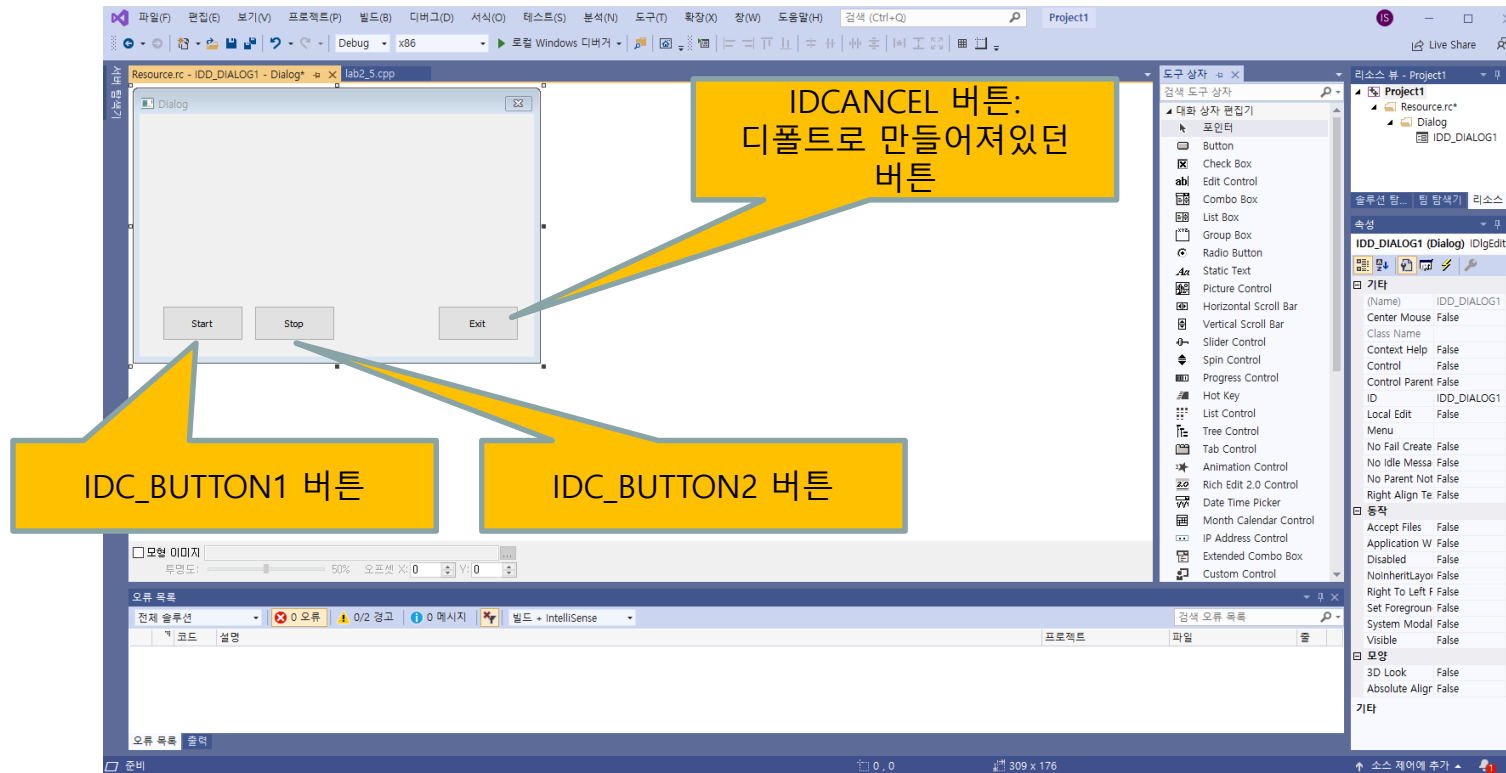
버튼 컨트롤

## • 버튼 컨트롤에서 오는 통지 정보

인자값		내용	
wParam	HIWORD(wParam)	컨트롤에 따른 통지 정보	<ul style="list-style-type: none"><li>• BN_CLICKED: 버튼이 클릭 되었음</li><li>• BN_DBLCLK: 버튼이 더블클릭 되었음</li><li>• BN_DISABLE: 버튼이 사용 불능 상태로 되었음</li><li>• BN_HILITE: 사용자가 버튼을 선택했음</li><li>• BN_SETFOCUS: 버튼이 포커스를 받았음</li></ul>
	LOWORD(wParam)	컨트롤 ID	
lParam		컨트롤 핸들값	

# 사용 예) 버튼 이용하기

- 버튼의 편집 및 배치
  - 버튼을 마우스로 선택하여 이동, 크기 변환. 캡션 변경 가능



- 사용 예) 위 그림의 3개의 버튼 메시지 처리
  - Start 버튼: 대화상자에 문자 출력,
  - Stop 버튼: 메시지 박스 출력
  - Exit 버튼: 대화상자 종료

```
LRESULT CALLBACK WndProc (HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
    switch (iMsg) {
        case WM_LBUTTONDOWN :    //--- 마우스 클릭하면 대화상자 띄우기
            DialogBox (g_hInst, MAKEINTRESOURCE(IDD_DIALOG1), hwnd, Dlalog_Proc);
            break;
    }
    return DefWindowProc (hwnd, iMsg, wParam, lParam) ;
}
```

```
BOOL CALLBACK Dlalog_Proc (HWND hDlg, UINT iMessage, WPARAM wParam, LPARAM lParam)
{
    HDC hdc;
    switch(iMessage) {
        case WM_INITDIALOG:
            break;
        case WM_COMMAND:
            switch (LOWORD(wParam)) {
                case IDC_BUTTON1:                                //--- Start 버튼
                    hdc = GetDC (hDlg);                          //--- 대화상자의 hdc를 가져옴
                    TextOut (hdc, 0, 0, L"Hello World", 11); //--- 대화상자에 출력함.
                    ReleaseDC (hDlg, hdc);
                    break;
                case IDC_BUTTON2:                                //--- Stop 버튼
                    MessageBox (hDlg, L"Stop Button", L"test", MB_OK);
                    break;
                case IDCANCEL:                                    //--- Exit 버튼
                    EndDialog(hDlg,0);
                    break;
            }
            break;
    }
    return 0;
}
```



# 컨트롤 관련 함수

## • 핸들 가져오기: **GetDlgItem** 함수

- 대화상자에 있는 컨트롤의 핸들(HWND)을 구함

**HWND GetDlgItem (HWND hDlg, int nIDDlgItem);**

- hDlg: 대화상자 핸들
- nIDDlgItem: 핸들을 구할 컨트롤의 ID
- 리턴값: 이 컨트롤의 윈도우 핸들을 리턴

- 예) IDC\_BUTTON1 아이디를 가진 컨트롤의 핸들 가져오기

```
HWND hButton;
```

```
hButton = GetDlgItem (hDlg, IDC_BUTTON1);
```



## • 컨트롤 ID 가져오기: **GetDlgCtrlID** 함수

- 특정 컨트롤의 윈도우 핸들로부터 컨트롤 ID 구함

**int GetDlgCtrlID (HWND hWndCtrl);**

- hWndCtrl: ID를 구할 컨트롤의 윈도우 핸들
- 리턴값: 컨트롤의 ID

- 예) hButton 컨트롤의 ID 가져오기

```
int id;
```

```
id = GetDlgCtrlID (hButton);
```

# 컨트롤 관련 함수

- 컨트롤의 사용 상태 변경하기: **EnableWindow** 함수
  - 컨트롤을 사용가능 상태 또는 사용불능 상태로 만들기

**BOOL EnableWindow (HWND hWnd, BOOL bEnable);**

- hWnd: 컨트롤의 핸들
- bEnable: 상태 설정 값, TRUE 면 사용 가능 상태, FALSE면 사용 불능 상태

- 예) hButton 컨트롤을 사용 불능 상태로 만들기

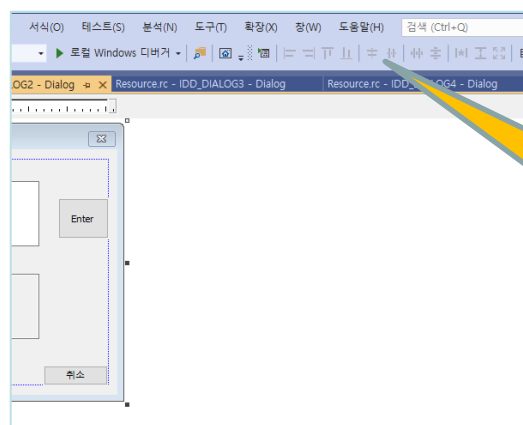
```
hButton = GetDlgItem (hDlg, ID_TEST);  
EnableWindow (hbutton, FALSE);
```

```
//--- ID_TEST 라는 id의 버튼 핸들 가져오기  
//--- 버튼을 사용 불능 상태로 만들기
```

- 예) hButton 컨트롤을 사용 가능 상태로 만들기

```
hButton = GetDlgItem (hDlg, ID_TEST);  
EnableWindow (hButton, TRUE);
```

```
//--- ID_TEST 라는 id의 버튼 핸들 가져오기  
//--- 버튼을 사용 가능 상태로 만들기
```



사용 불가능  
버튼

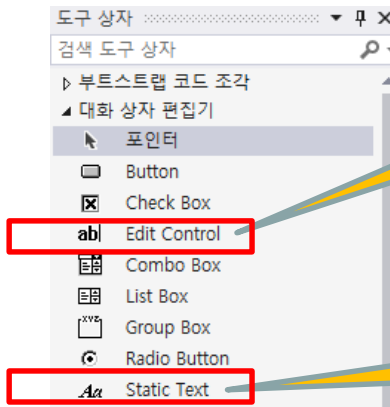


사용  
가능해졌다.

## 2) 에디트 박스 컨트롤

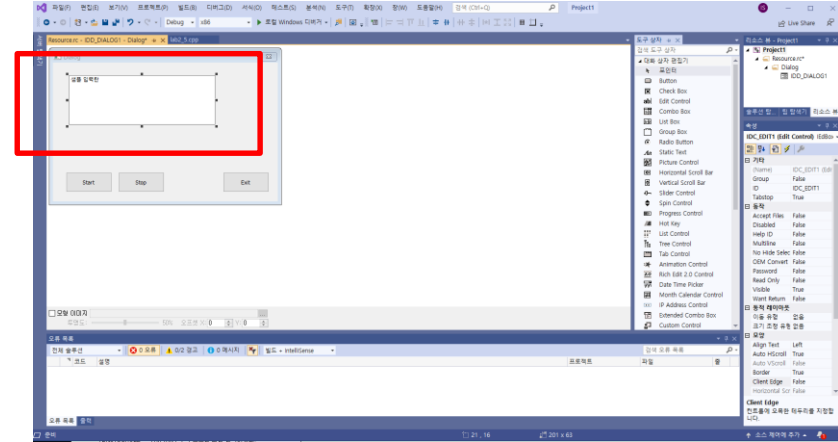
### • 에디트 박스 (Edit Box)

- 사용자의 키보드 입력 또는 출력을 위한 편집창
  - **Edit control**: 입력, 출력 가능
    - Edit Control의 속성으로 Read\_Only 로 설정 가능
  - **Static Text**: 문자열 출력만 가능



에디트 박스: 문자열을  
입력/출력

스태틱 에디트 박스: 문자열  
출력만 보여줌



### • 에디트 박스 통지 정보

인자값		내용	
wParam	HIWORD(wParam)	컨트롤에 따른 통지 정보	<ul style="list-style-type: none"> <li>• EN_CHANGE: 에디트 박스내의 내용이 변하였음</li> <li>• EN_HSCROLL: 에디트 박스의 수평스크롤바를 선택하였음</li> <li>• EN_VSCROLL: 에디트 박스의 수직스크롤바를 선택하였음</li> <li>• EN_SETFOCUS: 에디트 박스가 포커스를 받았음</li> </ul>
	LOWORD(wParam)	컨트롤 ID	
lParam		컨트롤 핸들값	

# 컨트롤 관련 함수

- **컨트롤 윈도우에서 텍스트 얻어오기: GetDlgItemText 함수**
  - 대화상자의 컨트롤에 WM\_GETTEXT 메시지를 보내서 텍스트를 얻어온다.

**UINT GetDlgItemText ( HWND hDlg, int nIDDlgItem, LPTSTR lpString, int nCount );**

- 리턴값: 성공하면 읽은 문자수 리턴, 실패하면 0 리턴
- hDlg: 컨트롤을 가지고 있는 대화상자의 핸들
- nIDDlgItem: 컨트롤의 ID
- lpString: 얻어낸 텍스트 스트링을 저장할 버퍼의 주소
- nCount: 버퍼의 길이 (문자열의 길이가 버퍼의 길이보다 길면 문자열은 잘린다)

- **컨트롤 윈도우에 텍스트를 출력하기: SetDlgItemText 함수**
  - 대화상자의 컨트롤에 텍스트를 출력한다.

**BOOL SetDlgItemText ( HWND hDlg, int nIDDlgItem, LPTSTR lpString );**

- 리턴값: 성공하면 0이 아닌 값 리턴, 실패하면 0 리턴
- hDlg: 컨트롤을 가지고 있는 대화상자의 핸들
- nIDDlgItem: 컨트롤의 ID
- lpString: 출력할 텍스트 스트링의 시작 주소

# 컨트롤 관련 함수

- 컨트롤 윈도우에서 문자열을 정수값으로 변환하여 읽어오기: **GetDlgItemInt** 함수
  - WM\_GETTEXT 메시지로 텍스트를 읽어 정수형으로 변환하여 리턴

**UINT GetDlgItemInt (HWND hDlg, int nIDDlgItem, BOOL\*lpTranslated, BOOL bSigned);**

- 리턴값: 변환된 정수값, 실패시 0 리턴
- hDlg: 컨트롤을 가지고 있는 윈도우 핸들
- nIDDlgItem: 컨트롤의 ID
- lpTranslated: 변환의 성공여부 리턴받는 변수. 변환되면 TRUE, 아니면 FALSE 로 설정 (에러 검사를 할 필요가 없을 때는 NULL로 설정)
- bSigned: 부호가 있는 정수인지 지정. 부호를 갖는 정수(int)이면 TRUE, 부호없는 정수(UINT)이면 FALSE

- 컨트롤 윈도우에 정수값을 출력하기: **SetDlgItemInt** 함수
  - 대화상자의 컨트롤에 정수값을 출력

**BOOL SetDlgItemInt ( HWND hDlg, int nIDDlgItem, UINT uValue, BOOL bSigned );**

- 리턴값: 성공하면 0이 아닌 값을 리턴, 실패하면 0 리턴
- hDlg: 컨트롤을 가지고 있는 윈도우 핸들
- nIDDlgItem: 컨트롤의 ID
- uValue: 컨트롤에 저장할 정수값
- bSigned: 부호가 있는 정수인지를 지정, 부호를 갖는 정수(int)이면 TRUE, 부호없는 정수(UINT)이면 FALSE

- **사용 예) 버튼과 에디트 박스 사용하여 문자/숫자 출력하기**
  - 버튼을 누르면 에디트 박스에 작성한 문장을 얻어 기존의 문자열에 붙여 다른 에디트 박스에 출력하기
    - 붙여넣기 하는 에디트 박스는 출력만 가능하게 한다
  - 2개의 에디트 박스에 각각 숫자를 입력받고, 더하기버튼 또는 빼기 버튼을 눌러 결과값을 출력하기
    - GetDlgItemText / SetDlgItemText 함수 사용하기
    - GetDlgItemInt / SetDlgItemInt 함수 사용하기

컨트롤 형태	ID	기능
에디트 박스	IDC_EDIT1	문자열을 입력받는다.
에디트 박스	IDC_EDIT2	입력받은 문자열을 출력한다.
에디트 박스	IDC_EDIT3	숫자를 입력받는다
에디트 박스	IDC_EDIT4	숫자를 입력받는다
에디트 박스	IDC_EDIT5	숫자의 덧셈 또는 뺄셈 결과를 출력한다
버튼	IDC_BUTTON1	입력받은 문자열을 출력 에디트 박스에 출력한다.
버튼	IDC_BUTTON2	문자열 에디트 박스를 클리어한다.
버튼	IDC_BUTTON3	두 숫자의 합을 결과 에디트 박스에 출력한다.
버튼	IDC_BUTTON4	두 숫자의 뺄셈을 결과 에디트 박스에 출력한다.
버튼	IDOK	모든 에디트 박스를 클리어한다.
버튼	IDCANCEL	대화상자 종료

Dialog

window programming

Enter

Editbotx Test: window programming

Clear

25

+

31

-

56

확인 취소

Dialog

window programming

Enter

Editbotx Test: window programming

Clear

25

+

31

-

-6

확인 취소

```

BOOL CALLBACK Dlalog_Proc (HWND hDlg, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
    static TCHAR editText[100];
    TCHAR resultText[100] = L"Editbotx Test: ";
    static int x, y, result;

    switch (iMsg)
    {
    case WM_COMMAND:
        switch (LOWORD(wParam)) {
            case IDC_BUTTON1:
                //--- 첫번째 에디트 박스에서 문자열을 가져옴
                GetDlgItemText (hDlg, IDC_EDIT1, editText, 100);
                //--- 가져온 문자열을 resultText 문자열에 덧붙임
                lstrcat (resultText, editText);
                //--- 두번째 에디트 박스에 합쳐진 문자열을 출력함
                SetDlgItemText (hDlg, IDC_EDIT2, resultText);
            break;

            case IDC_BUTTON2:
                SetDlgItemText (hDlg, IDC_EDIT1, L"");
                SetDlgItemText (hDlg, IDC_EDIT2, L"");
            break;
        }
    }
}

```

```

case IDC_BUTTON3:    //--- 더하기: +
    //--- 에디트 박스에서 숫자를 가져옴
    x = GetDlgItemInt (hDlg, IDC_EDIT3, NULL, TRUE);
    y = GetDlgItemInt (hDlg, IDC_EDIT4, NULL, TRUE);
    result = x + y;
    SetDlgItemInt (hDlg, IDC_EDIT5, result, TRUE);
break;

case IDC_BUTTON4:    //--- 빼기: -
    //--- 에디트 박스에서 숫자를 가져옴
    x = GetDlgItemInt (hDlg, IDC_EDIT3, NULL, TRUE);
    y = GetDlgItemInt (hDlg, IDC_EDIT4, NULL, TRUE);
    result = x - y;
    SetDlgItemInt (hDlg, IDC_EDIT5, result, TRUE);
break;

case IDOK:
    SetDlgItemText (hDlg, IDC_EDIT1, L"");
    SetDlgItemText (hDlg, IDC_EDIT2, L"");
    SetDlgItemText (hDlg, IDC_EDIT3, L"");
    SetDlgItemText (hDlg, IDC_EDIT4, L"");
    SetDlgItemText (hDlg, IDC_EDIT5, L"");
break;

case IDCANCEL:
    EndDialog(hDlg, 0);
break;
}
return 0;
}

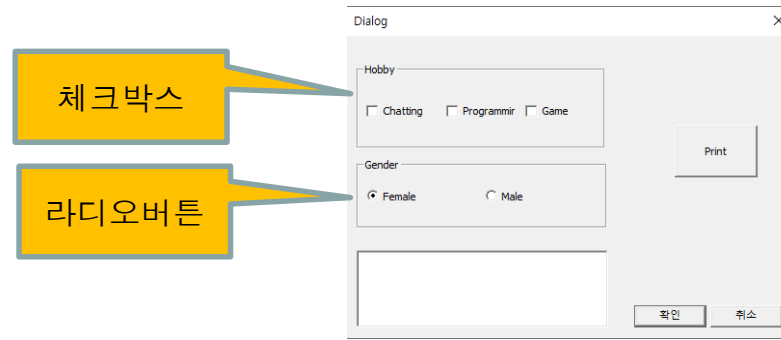
```



### 3) 체크박스과 라디오 버튼 컨트롤

- 체크버튼과 라디오 버튼

- 체크 박스: 복수 항목 선택 가능
- 라디오 버튼: 한 항목만 선택 가능

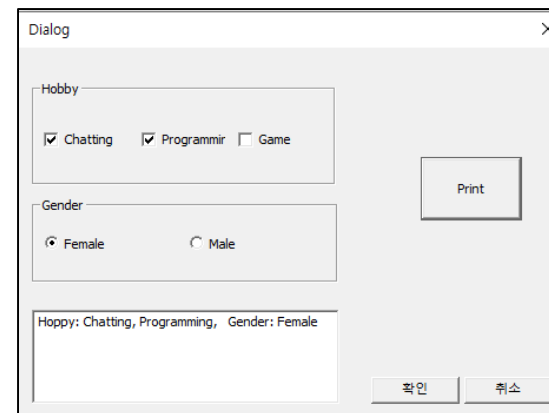
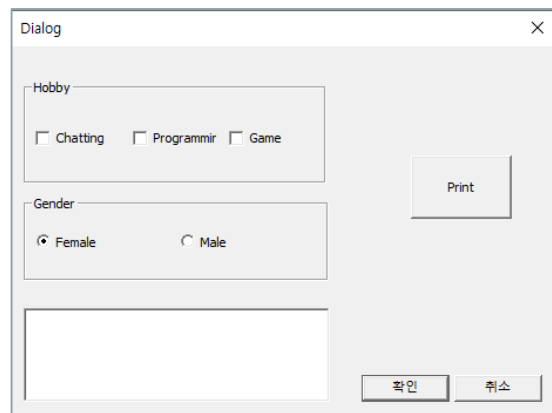


컨트롤에 보내는 메시지	의미	리턴 값 또는 체크 박스 상태
BM_GETCHECK	체크박스가 현재 체크되어 있는 상태인지 조사	<ul style="list-style-type: none"><li>• BST_CHECKED: 현재 체크되어 있다.</li><li>• BST_UNCHECKED: 현재 체크되어 있지 않다.</li><li>• BST_INDETERMINATE: 체크도 아니고 비 체크도 아닌 상태</li></ul>
BM_SETCHECK	체크 박스의 체크 상태를 변경, wParam에 변경할 체크상태를 보내준다	

**LRESULT SendMessage (HWND hWnd, UINT Msg, WPARAM wParam, LPARAM lParam);**

- 메시지를 메시지 큐에 넣지 않고 바로 윈도우 프로시저로 보냄
- hWnd: 메시지를 전달받을 윈도우 핸들
- Msg : 전달할 메시지
- wParam, lParam 메시지의 추가적 정보, 메시지에 따라 다른 정보 반환

- 사용 예) 취미와 성별을 선택 후 선택한 결과 출력하기
  - 취미: 1개이상 선택 가능 - 체크박스
  - 성별: 1개 선택 - 라디오버튼 (그룹 설정: 첫 번째 라디오 버튼의 group을 true로 설정)
  - 그 외, 버튼과 에디트박스 사용
    - 출력 버튼, 클리어 버튼



**BOOL CheckRadioButton ( HWND hDlg, int nIDFirstButton, int nIDLastButton, int nIDCheckButton );**

- 처음 선택될 라디오 버튼 선택
- hDlg: 라디오 버튼을 가지는 부모 윈도우(또는 대화상자)의 핸들
- nIDFirstButton : 각 그룹의 시작 버튼 아이디
- nIDLastButton: 각 그룹의 끝 버튼 아이디
- nIDCheckButton: 선택될 버튼의 아이디

The image illustrates the use of Group Boxes and Radio Buttons in a Windows application. It consists of three main parts:

- Dialog Box:** A sample dialog box titled "Dialog" containing two Group Boxes. The "Hobby" Group Box contains three Check Boxes: "Chatting", "Programmir", and "Game". The "Gender" Group Box contains two Radio Buttons: "Female" (selected) and "Male". There is also a "Print" button and "확인" (OK) and "취소" (Cancel) buttons at the bottom.
- Toolbox:** A screenshot of the Windows Forms toolbox showing various controls. The "Group Box" control is highlighted in blue.
- Properties Window:** A screenshot of the Windows Forms Properties window for a control named "IDC\_RADIO1 (Radio-button Cor". The "기타" (Other) tab is selected, showing the following properties:

Property	Value
(Name)	IDC_RADIO1 (Rac
Group	True
ID	IDC_RADIO1
Tabstop	False

Callouts provide additional information:

- A yellow callout box labeled "그룹 박스" (Group Box) points to the "Hobby" and "Gender" Group Boxes in the dialog.
- A yellow callout box labeled "라디오 버튼의 그룹이 여러 개 있을 경우, 한 그룹의 첫번째 라디오 버튼 → 그룹을 true로 설정" (If there are multiple groups of radio buttons, set the first radio button of one group to true) points to the "Female" radio button in the "Gender" Group Box.

BOOL CALLBACK DIALOG\_Proc (HWND hDlg, UINT iMsg, WPARAM wParam, LPARAM lParam)

{

```
static int check[3], radio;
TCHAR hobby[][20] = { L"Chatting", L"Programming", L"Game" };
TCHAR gender[][20] = { L"Female", L"Male" };
TCHAR output[100];
HWND hCheck[3], hRadio[2];
```

switch(iMsg)

{

case **WM\_INITDIALOG**:

//--- 시작 버튼, 끝 버튼, 체크할 버튼

**CheckRadioButton** (hDlg, IDC\_RADIO1, IDC\_RADIO2, IDC\_RADIO1);

break;

case **WM\_COMMAND**:

switch (**LOWORD(wParam)**) {

case **IDC\_CHECK1**: //--- 취미1

check[0] = 1 - check[0];

break;

case **IDC\_CHECK2**: //--- 취미2

check[1] = 1 - check[1];

break;

case **IDC\_CHECK3**: //--- 취미3

check[2] = 1 - check[2];

break;

case **IDC\_RADIO1**: //--- 성별1

radio = 0;

break;

case **IDC\_RADIO2**: //--- 성별2

radio = 1;

break;

case **IDC\_BUTTON1**: //--- 출력 버튼

wsprintf (output, L"Hoppy: %s, %s, %s \n Gender: %s",

check[0] ? hobby[0] : L"",

check[1] ? hobby[1] : L"",

check[2] ? hobby[2] : L"",

gender[radio]);

**SetDlgItemText** (hDlg, IDC\_EDIT1, output);

break;

case **IDOK**: //--- 확인 버튼 (모든 값을 초기화)

lstrcpy (output, L" ");

**SetDlgItemText** (hDlg, IDC\_EDIT1, output);

radio = check[0] = check[1] = check[2] = 0;

hCheck[0] = **GetDlgItem** (hDlg, IDC\_CHECK1);

hCheck[1] = **GetDlgItem** (hDlg, IDC\_CHECK2);

hCheck[2] = **GetDlgItem** (hDlg, IDC\_CHECK3);

for ( int i = 0; i < 3; i++ )

**SendMessage** (hCheck[i], BM\_SETCHECK, BST\_UNCHECKED, 0);

**CheckRadioButton** (hDlg, IDC\_RADIO1, IDC\_RADIO2, IDC\_RADIO1);

break;

case **IDCANCEL**:

EndDialog(hDlg, 0);

break;

}

}

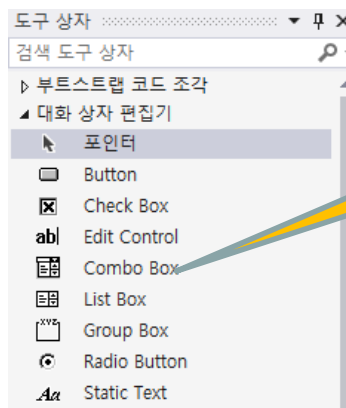
return 0;

}

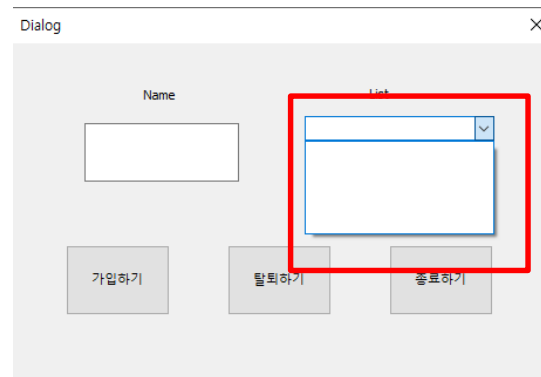
## 4) 콤보 박스 컨트롤

### • 콤보 박스 컨트롤

- 사용자의 키보드 입력 또는 출력을 위한 편집창
- 여러 항목들의 리스트를 나열하여 보여주는 컨트롤



콤보 박스: 여러 항목  
리스트 나열



### • 콤보 박스 통지 정보

인자값		내용	
wParam	HIWORD(wParam)	컨트롤에 따른 통지 정보	<ul style="list-style-type: none"><li>• CBN_DROPDOWN: 콤보 박스에 등록된 항목들이 아래로 펼쳐짐</li><li>• CBN_DBLCLK: 아래로 펼쳐진 항목 리스트에서 하나를 더블클릭으로 선택했음</li><li>• CBN_EDITCHANGE: 콤보 박스의 텍스트 편집 공간에 텍스트를 추가하거나 수정하였음</li><li>• CBN_SELCHANGE: 사용자가 항목 리스트에서 하나를 선택하였음</li></ul>
	LOWORD(wParam)	컨트롤 ID	
lParam		컨트롤 핸들값	

# 콤보박스에 보내는 메시지

- 콤보 박스 컨트롤에 보내는 메시지 (SendMessage 함수로 보내는 메시지)

메시지	의미	전달 값
CB_ADDSTRING	콤보 박스에 텍스트를 아이템으로 추가하는 메시지로써 리스트의 마지막에 추가된다.	wParam: 사용하지 않음 lParam: 텍스트 스트링의 시작 주소
CB_DELETESTRING	콤보 박스에 있는 아이템들 중 하나를 삭제하는 메시지	wParam: 삭제하기 원하는 아이템의 인덱스로 0부터 시작한다. lParam: 0
CB_GETCOUNT	콤보 박스의 아이템 리스트에 들어 있는 아이템의 개수를 얻기 위한 메시지로 개수 값은 SendMessage()함수가 리턴한다.	wParam: 0 lParam: 0
CB_GETCURRESEL	현재 선택된 아이템의 인덱스 번호를 얻기 위한 메시지로 인덱스 번호는 SendMessage()함수가 리턴한다.	wParam: 0 lParam: 0
CB_SETCURRESEL	콤보 박스 컨트롤의 텍스트 편집 공간에 지정한 항목의 텍스트를 보여준다.	wParam: 나타내고자 하는 항목의 인덱스 번호 lParam: 사용안함

# SendMessage()

- **SendMessage() 함수는**

- 메시지를 메시지 큐에 넣지 않고 바로 윈도우 프로시저로 보냄
- 윈도우 프로시저로 메시지를 보내 바로 처리
- 메시지가 처리되기 전까지 반환되지 않음, 즉 윈도우 프로시저가 값을 반환해야만 SendMessage 도 반환하여 끝마칠 수 있음

예) SendMessage (hCombo, CB\_ADDSTRING, 0, (LPARAM)name);

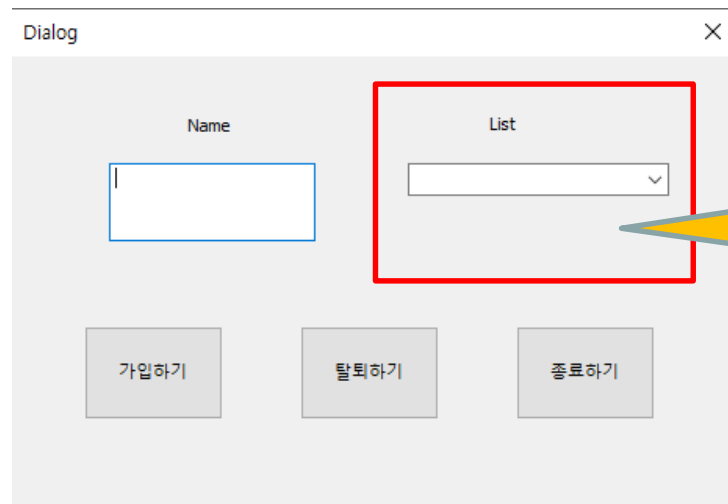
- hCombo 컨트롤에 CB\_ADDSTRING 메시지를 보내는데, 즉 문자열 name을 hCombo 에 추가하라는 메시지
- 윈도우에서 컨트롤로 메시지 전송 : ADD\_STRING, DELETE\_STRING
- 컨트롤에서 윈도우로 메시지 전송 : LBN\_DBLCLK, LBN\_SELCHG

**LRESULT SendMessage (HWND hWnd, UINT Msg, WPARAM wParam, LPARAM lParam);**

- hWnd: 메시지를 전달받을 윈도우 핸들
- Msg : 전달할 메시지
- wParam, lParam 메시지의 추가적 정보, 메시지에 따라 다른 정보 반환



- 사용 예) 대화상자에 콤보박스 그리기
  - 회원이름을 넣고 가입하면 회원명단에 추가됨
  - 콤보박스 사용



리소스에서 콤보박스를 만들 때,  
높이를 충분히 길게 만들어준다.  
이 길이는 우측의 pull-down  
버튼을 눌렀을 때 보여지는  
콤보박스의 길이

컨트롤 종류	ID	
Static	IDC_STATIC	제목 보여주기
Static	IDC_STATIC	제목 보여주기
Edit	IDC_EDIT_NAME	이름 작성하기
<b>Combo</b>	<b>IDC_COMBO_LIST</b>	<b>작성한 이름 출력하기</b>
Button	IDC_BUTTON_INSERT	회원 이름을 명단에 넣기
Button	IDC_BUTTON_DELETE	이름 삭제하기
Button	IDC_CLOSE	대화상자 닫기



```
BOOL CALLBACK DIALOG_Proc (HWND hDlg, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
    static int selection;
    static HWND hCombo;
    TCHAR name[20];

    switch(iMsg)
    {
    case WM_INITDIALOG:
        hCombo = GetDlgItem (hDlg, IDC_COMBO_LIST);          //--- 회원명단
        break;

    case WM_COMMAND:
        switch (LOWORD(wParam))
        {
        case IDC_BUTTON_INSERT:                                //--- 가입 버튼이 눌러짐
            GetDlgItemText (hDlg, IDC_EDIT_NAME, name, 20);    //--- 이름 문자열 획득
            if (lstrcmp(name, L""))                               //--- 이름이 들어 왔으면, 그 값으로 채워라
                SendMessage (hCombo, CB_ADDSTRING, 0, (LPARAM)name);
            break;

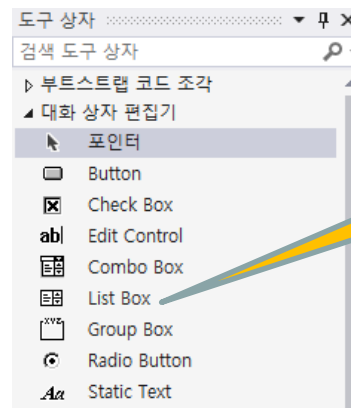
        case IDC_BUTTON_DELETE:                                //--- 탈퇴하라 버튼이 눌러짐
            SendMessage (hCombo, CB_DELETESTRING, selection, 0);
            break;

        case IDC_COMBO_LIST:                                    //--- 콤보박스가 눌러짐
            if (HIWORD(wParam) == CBN_SELCHANGE)                 //--- 하나가 선택됨(상태 변경)
                selection = SendMessage (hCombo, CB_GETCURSEL, 0, 0);
            break;
        case IDC_CLOSE:
            EndDialog(hDlg,0);
            break;
        }
    }
    return 0;
}
```

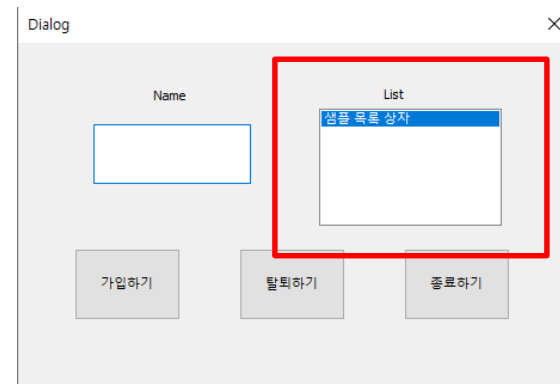
## 5) 리스트 박스 컨트롤

### • 리스트 박스

- 사용자의 키보드 입력 또는 출력을 위한 편집창
- 여러 항목들의 리스트를 나열하여 보여주는 컨트롤
- 콤보 박스 컨트롤은 버튼을 누르기 전에는 항목 리스트 컨트롤을 보여주지 않지만, 리스트 컨트롤은 외부 입력이 없어도 항목을 보여준다.



리스트 박스: 여러  
항목 리스트 나열



### • 리스트 박스에서 오는 통지 정보

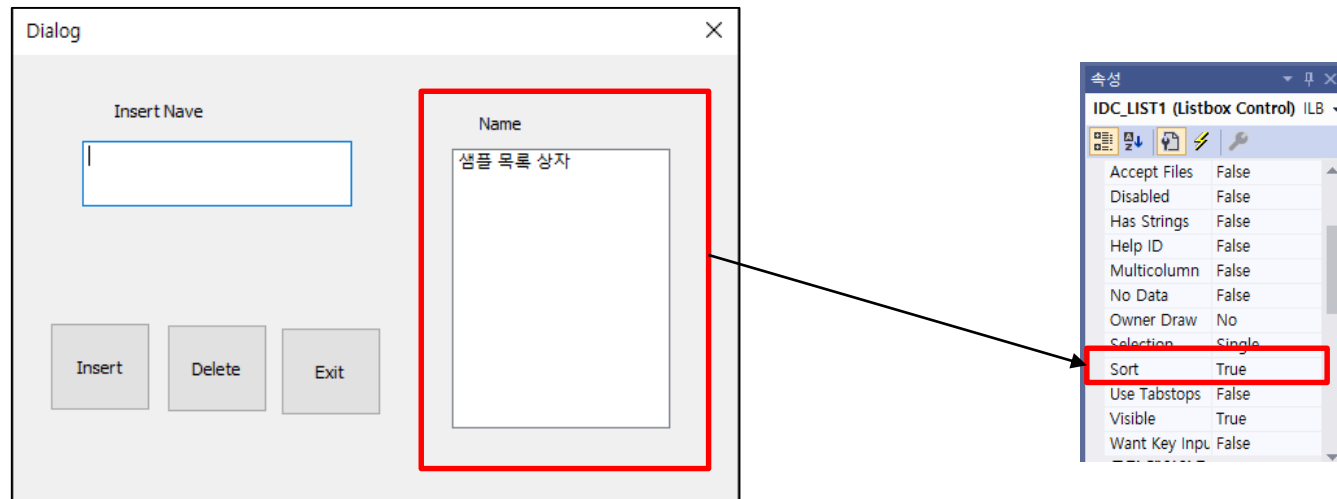
인자값		내용	
wParam	HIWORD(wParam)	컨트롤에 따른 통지 정보	<ul style="list-style-type: none"><li>• LBN_DBLCLK: 리스트 박스의 여러 아이템들 중 하나를 더블클릭 했음</li><li>• LBN_SELCHANGE: 아이템들중 하나가 선택되었음</li><li>• LBN_SETFOCUS: 리스트 박스가 포커스를 받았음</li><li>• LBN_KILLFOCUS: 리스트 박스가 포커스를 잃었음</li></ul>
	LOWORD(wParam)	컨트롤 ID	
lParam		컨트롤 핸들값	

# 리스트 박스에 보내는 메시지

- 리스트 박스에 보내는 메시지 (SendMessage 함수로 보내는 메시지)

메시지	의미	전달 값
LB_ADDSTRING	리스트 박스에 텍스트를 아이템으로 추가하는 메시지로써 리스트의 마지막에 추가된다.	wParam: 사용하지 않음 lParam: 텍스트 스트링의 시작 주소
LB_DELETESTRING	리스트 박스에 있는 아이템들 중 하나를 삭제하는 메시지	wParam: 삭제하기 원하는 아이템의 인덱스로 0부터 시작한다. lParam: 0
LB_GETCOUNT	리스트 박스의 아이템 리스트에 들어 있는 아이템의 개수를 얻기 위한 메시지로 개수 값은 SendMessage()함수가 리턴한다.	wParam: 0 lParam: 0
LB_GETCURRESEL	현재 선택된 아이템의 인덱스 번호를 얻기 위한 메시지로 인덱스 번호는 SendMessage()함수가 리턴한다.	wParam: 0 lParam: 0
LB_GETTEXT	아이템 리스트중 wParam에서 지정한 인덱스 아이템의 텍스트를 얻어 오는 메시지	wParam: 얻어올 아이템의 인덱스 번호 lParam: 얻어온 텍스트를 저장할 버퍼의 시작 주소
LB_INSERTSTRING	리스트 박스에 텍스트를 아이템으로 리스트 중간에 추가하는 메시지	wParam: 아이템 리스트중 추가될 위치의 인덱스 번호 lParam: 텍스트 스트링의 시작 주소

- 사용 예) 대화상자에 리스트박스 그리기
  - 리스트박스 사용



컨트롤 종류	ID	기능
Static	IDC_STATIC	Insert Name 제목 보여주기
Static	IDC_STATIC	Name 보여주기
Edit	IDC_EDIT_NAME	이름 입력하는 에디트 박스
<b>List Box</b>	<b>IDC_LIST_NAME</b>	<b>추가된 이름 리스트 보여주는 리스트 박스</b>
Button	IDC_BUTTON_INSERT	이름을 리스트박스에 추가하는 버튼
Button	IDC_BUTTON_DELETE	리스트박스에서 현재 선택된 아이템을 삭제
Button	IDC_CLOSE	대화상자 닫기

```
BOOL CALLBACK DIALOG_Proc (HWND hDlg, UINT iMsg, WPARAM wParam, LPARAM lParam)
{

    static int selection;
    static HWND hList;

    switch(iMsg)
    {
    case WM_INITDIALOG:
        hList = GetDlgItem (hDlg, IDC_LIST_NAME);
        break;

    case WM_COMMAND:
        switch (LOWORD(wParam))
        {
        case IDC_BUTTON_INSERT:
            GetDlgItemText (hDlg, IDC_EDIT_NAME, name, 20);
            if (lstrcmp(name, ""))
                SendMessage (hList, LB_ADDSTRING, 0, (LPARAM)name);
            break;

        case IDC_BUTTON_DELETE:
            SendMessage (hList, LB_DELETESTRING, selection, 0);
            break;

        case IDC_LIST_NAME:
            if (HIWORD(wParam) == LBN_SELCHANGE)
                selection = SendMessage (hList, LB_GETCURSEL, 0, 0);
            break;
        }
        break;
    }
    return 0;
}
```

### 3. 모달리스 대화상자

- **모달(Modal)형 대화상자**

- 이 대화상자를 닫지 않으면 다른 윈도우로 전환할 수 없는 특징을 갖는 대화상자
  - 대화상자가 떠있는 상태에서 해당 프로그램의 대화상자 이외의 부분을 클릭하면 "뽁"하는 소리가 나는 경우
- 해당 프로그램의 다른 윈도우로는 전환할 수 없으나, 다른 프로그램은 실행 가능
- 대표적인 예: MessageBox() 함수에 의해서 만들어진 대화상자

- **모달리스(Modaless)형 대화상자**

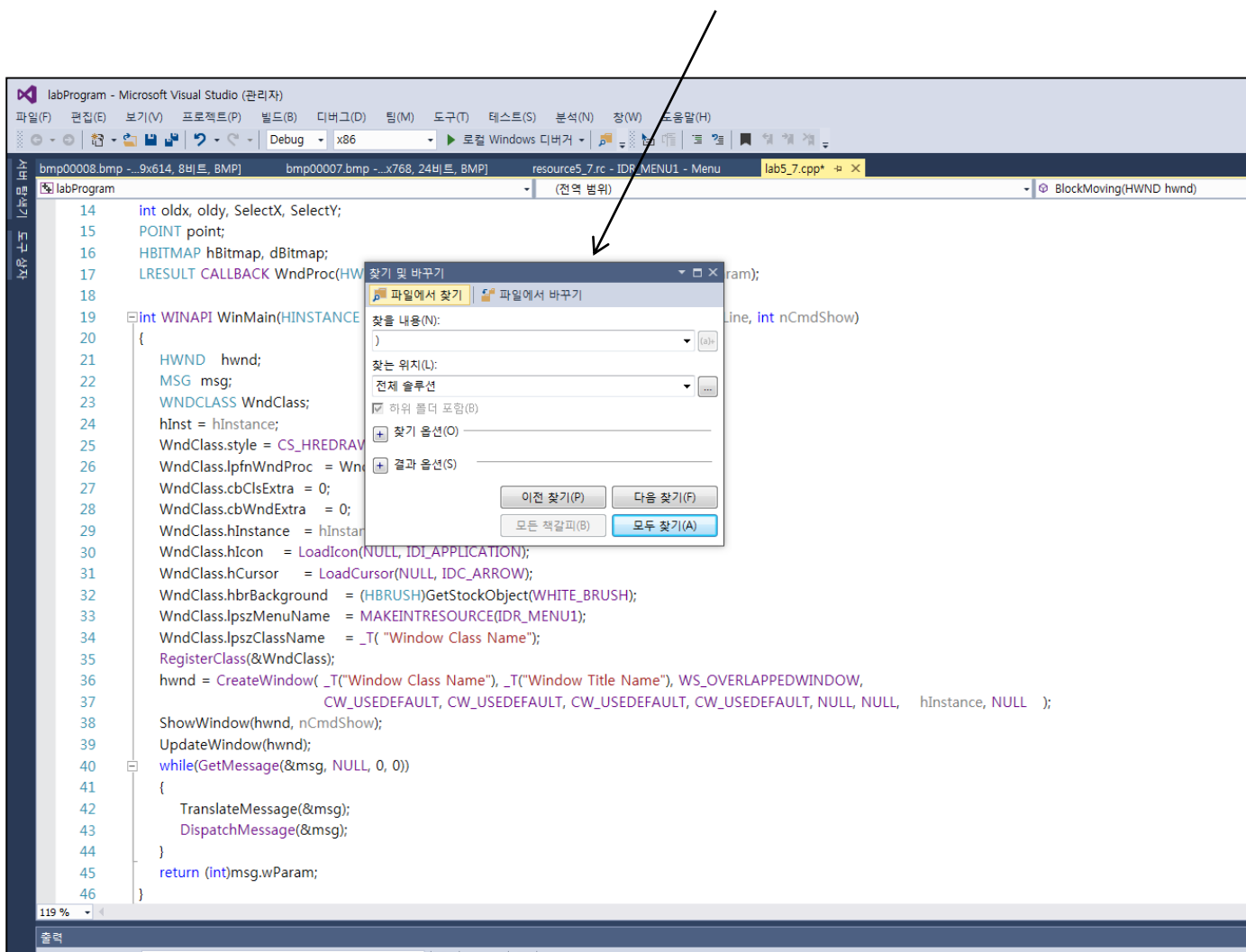
- 해당 대화상자를 닫지 않아도 다른 윈도우로 전환할 수 있는 특징을 갖는 대화상자
- 대표적인 예: 많은 프로그램에서 제공하는 "찾기" 대화상자
  - "찾기" 메뉴항목은 보통 해당 내용을 찾은 후 편집작업 등을 수행하고 다음 찾기를 하기 때문에 모달리스형 대화상자가 더 바람직

- **모달리스 대화상자 만들기**

- 모달형과 마찬가지로
  - 1) 리소스: 대화상자 만들기
  - 2) 대화상자 띄우기: 함수 호출
  - 3) 대화상자에서 발생하는 메시지 처리하기: 대화상자 프로시저 만들기

### 3. 모델리스 대화상자

- **모델리스 대화상자:** 대화상자가 나타나도 부모 윈도우를 선택할 수 있는 대화상자



# 모달리스 대화상자 관련 함수

## • 모달리스 대화상자를 생성하는 함수

**HWND CreateDialog** ( **HINSTANCE** hInstance, **LPCTSTR** lpTemplate,  
**HWND** hWndParent, **DLGPROC** lpDialogFunc);

- 대화상자를 만들고 바로 대화상자의 핸들값을 리턴한다.
- **HINSTANCE** hInstance: 인스턴스 핸들
- **LPCTSTR** lpTemplate: 대화상자 ID
- **HWND** hWndParent: 윈도우 핸들
- **DLGPROC** lpDialogFunc: 메시지 처리 함수

## • 모달리스 대화상자를 보이거나 숨기는 함수

**BOOL ShowWindow** ( **HWND** hwnd, **int** nCmdShow );

- **HWND** hwnd: 윈도우 핸들
- **int** nCmdShow: 윈도우 보이기 (SW\_SHOW: 나타냄, SW\_HIDE: 감춤)

## • 모달리스 대화상자 종료하기 함수

**BOOL DestroyWindow** ( **HWND** hwnd);

- **HWND** hwnd: 윈도우 핸들



# 모달리스 대화상자

//--- 부모 윈도우의 윈도우 프로시저

LRESULT CALLBACK WndProc (HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)

```
{
    HWND hDlg = NULL;
    switch (iMsg)
    {
        case WM_LBUTTONDOWN :    //--- 왼쪽 마우스 버튼을 누르면 모달리스 형태의 대화상자 열기
            hDlg = CreateDialog (hInst, MAKEINTRESOURCE(IDD_DIALOG6_8), hwnd, ModelessDlgProc );
            ShowWindow (hDlg, SW_SHOW);
            break;

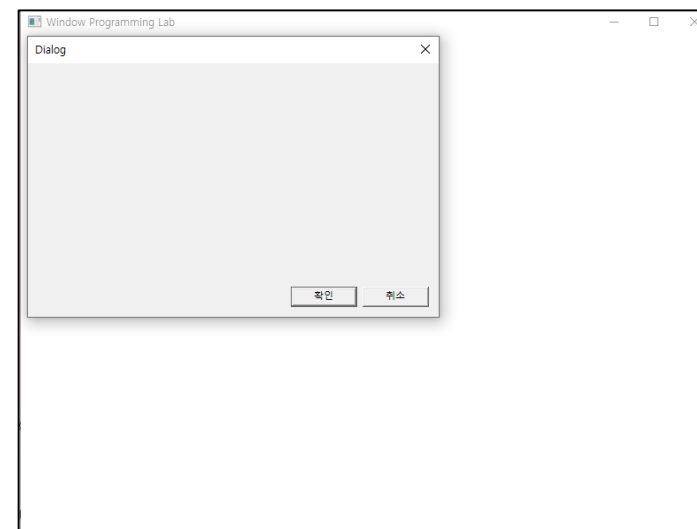
        return DefWndProc (hwnd, iMsg, wParam, lParam);
    }
}
```

//--- 모달리스 대화상자의 대화상자 프로시저

//--- 모달형과 마찬가지로, 대화상자에 설정한 컨트롤들을 처리

BOOL CALLBACK ModelessDlgProc (HWND hDlg, UINT iMsg, WPARAM wParam, LPARAM lParam)

```
{
    switch (iMsg) {
        case WM_COMMAND:
            switch (LOWORD (wParam)) {
                case IDOK:
                    DestroyWindow(hDlg);
                    hDlg=NULL;
                    break;
                case IDCANCEL:
                    DestroyWindow(hDlg);
                    hDlg=NULL;
                    break;
            }
        }
    return 0;
}
```



# 실습 6-1

## • 컨트롤 사용하기

- 대화상자에 라디오 버튼, 체크박스, 버튼을 만들고, 경로를 그리고 그 경로에서 원이 이동하는 애니메이션 만들기

### - 라디오 버튼

- 라디오 버튼1: 사인 곡선 그리기
- 라디오 버튼2: 사선을 이용한 지그재그 그리기
- 라디오 버튼3: 스프링 그리기
- 라디오 버튼4: 이중선 그리기 (위 아래에 각각 선이 그려진다)

### - 버튼

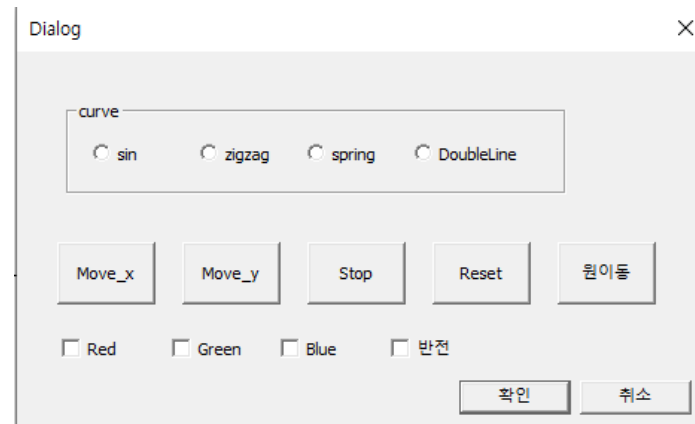
- 버튼1 (Move\_x 버튼): 경로를 좌측으로 움직이는 애니메이션
- 버튼2 (Move\_y 버튼): 경로를 위아래로 움직이는 애니메이션
- 버튼3 (Stop 버튼): 애니메이션이 멈춘다.
- 버튼4: 리셋 버튼
- 버튼5 (원 이동 버튼): 경로에 따라 원이 이동

### - 체크버튼 (1개 이상 선택 가능: 1개 이상이 선택되면 선택된 색이 합성되어 결정된다)

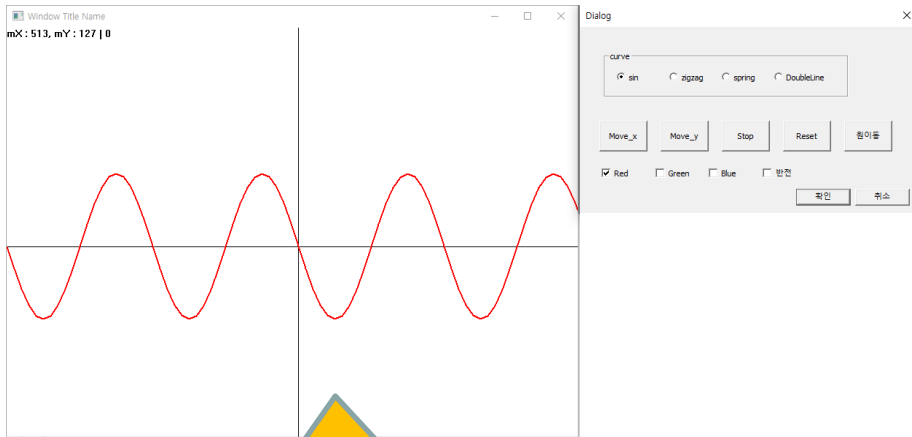
- 체크 버튼1: 선의 색이 빨강
- 체크 버튼2: 선의 색이 초록
- 체크 버튼3: 선의 색이 파랑
- 체크 버튼4: 반전 (버튼 1, 2, 3 선택의 반전색)

### - 버튼

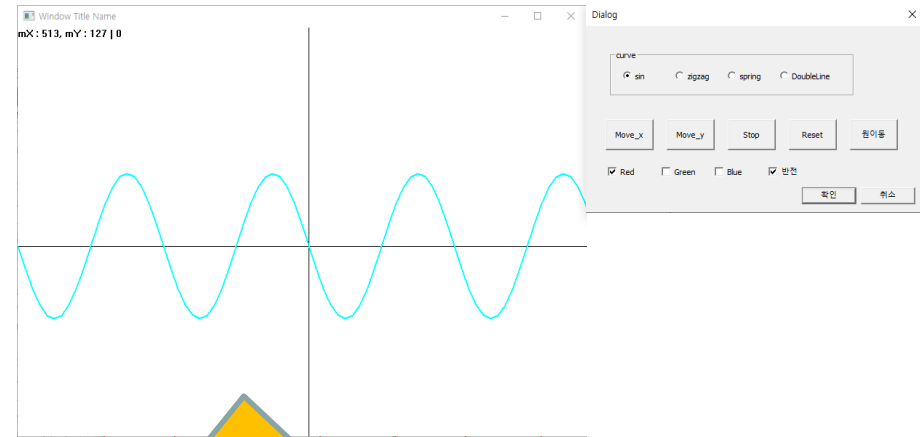
- 확인: 모든 값을 리셋
- 취소: 대화상자 종료



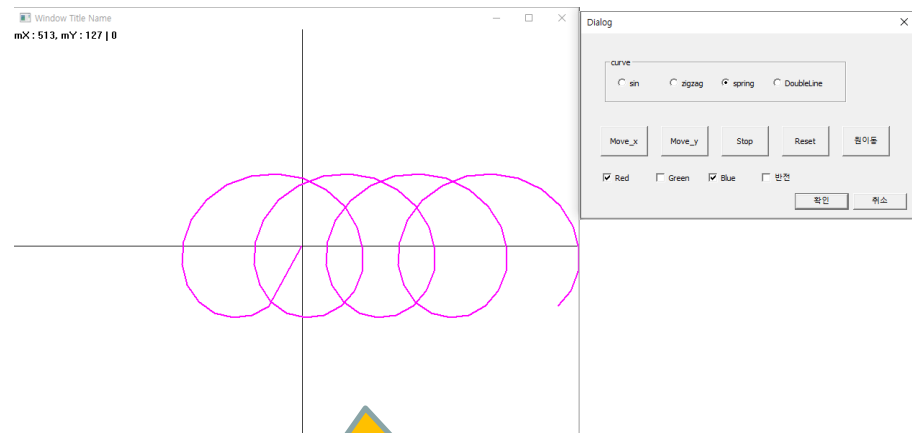
# 실습 6-1



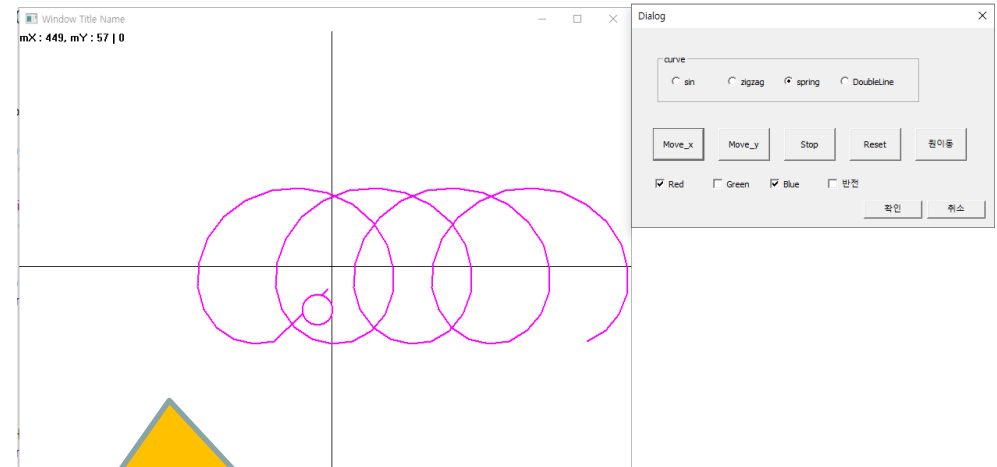
사인 곡선: 빨강색으로 그림



사인 곡선: 빨강색의 반전색으로 그려지고  
애니메이션 되어 그림이 바뀌었다.

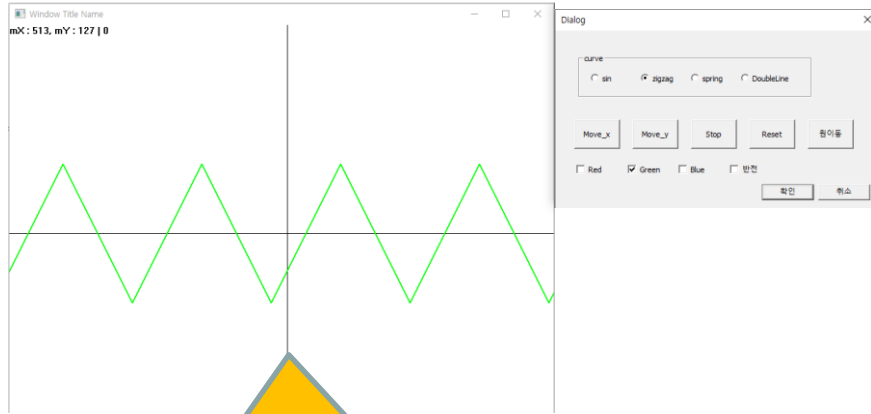


스프링 곡선: 빨강색+파랑색으로  
그림

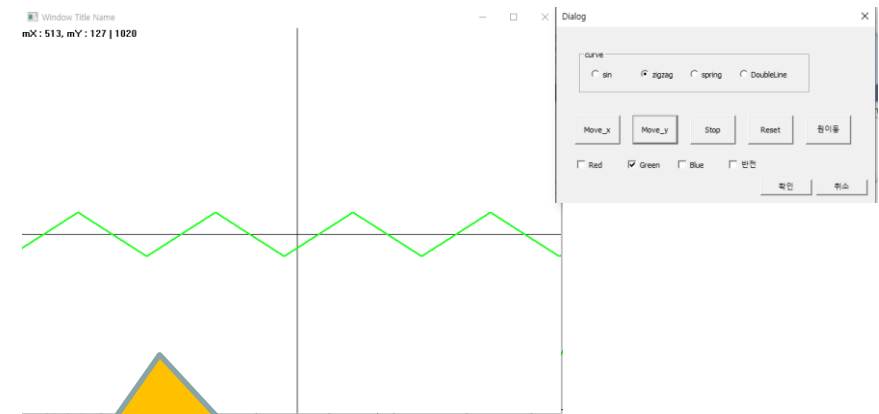


스프링 곡선과 원 애니메이션

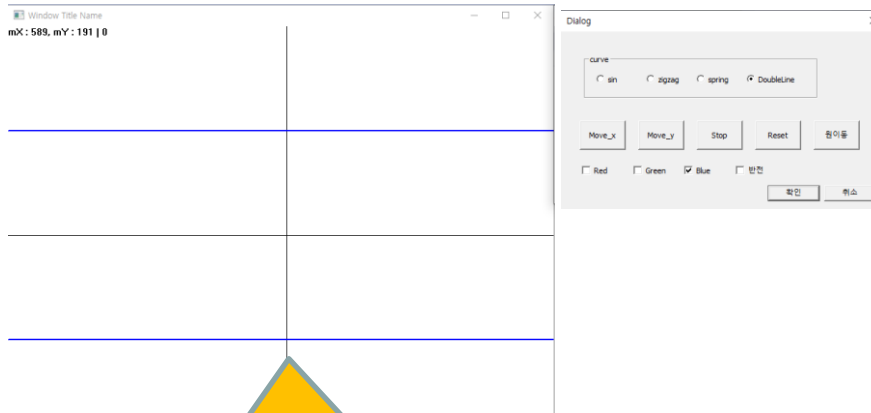
# 실습 6-1



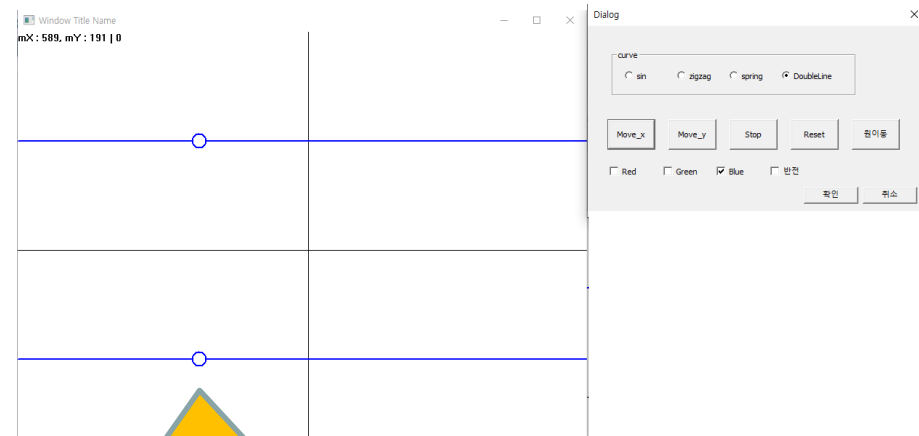
지그재그 곡선: 초록색으로 그림



지그재그 곡선 애니메이션 (왼쪽으로 이동함)



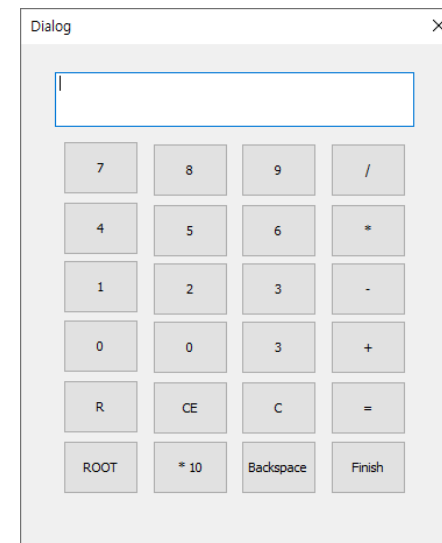
DoubleLine: 파랑색으로 그림



DoubleLine: 원 이동 애니메이션  
(원이 오른쪽으로 이동함)

## • 모달리스 대화상자를 이용하여 계산기 구현하기

- 에디트 박스 컨트롤에 숫자를 직접 입력하거나 숫자 버튼을 눌러 입력한다.
- 버튼으로 숫자를 입력하는 계산기
  - 기본 사칙연산 버튼: + / - / \* / / (나눗셈)
  - 대입 버튼: = (연산을 하고 대입 버튼을 누르면 결과가 나온다 → 입력 스타일에 따라 진행)
- 기존 계산기에 버튼 추가한다.
- **버튼 1 (R 버튼):** 입력된 숫자들의 순서를 바꾸는 버튼
  - 예)  $12+34 \rightarrow$  (R 버튼)  $\rightarrow 21 + 43$
- **버튼 2 (CE 버튼):** 마지막으로 입력한 값을 지운다.
  - 예)  $123 + 2 + 3 \rightarrow$  (CE 버튼)  $\rightarrow 123 + 2 +$  또는  $123 + 2$  (→ 모두 가능)
- **버튼 3 (C 버튼):** 모든 입력 값을 삭제한다.
- **버튼 4 (fac 버튼):** 에디트 박스에 출력된 숫자의 factorial 값을 출력한다.
- **버튼 5 (\*10 버튼):** 입력된 숫자에 10을 곱한다.
- **버튼 6 (← 버튼):** 입력된 숫자에서 마지막 한자리씩 삭제한다. (100자리 → 10자리 → 1자리)  
(실제 계산기에서 테스트해보기, 1자리에서 다시 버튼을 누르면 0이 된다.)
- **버튼 7 (10<sup>x</sup>):** 입력한 숫자의 10의 입력한 숫자승의 값을 출력한다. (예, 에디트 박스에 2가 있고, 버튼7을 누르면  $10^2 \rightarrow 100$  출력)
- **버튼 8 (종료하기):** 프로그램 (또는 대화상자)을 종료한다.



\*\* 주의

- 입력 숫자: 정수를 기본으로 입력
- 결과값 실수값도 출력
- 숫자와 연산자 입력은 본인이 결정
  - 예) 연산자 입력하면 바로 결과 출력 또는 = 입력하면 결과 출력

# 이번 주에는

- 대화상자와 컨트롤
  - 대화상자 만들기
  - 대화상자 위에 컨트롤 올리기
- 다음 주에는
  - 차일드 윈도우 만들기