

6장 대화상자와 컨트롤

2014년도 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.
- 모달리스 대화상자를 사용할 수 있다.

- **내용**

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

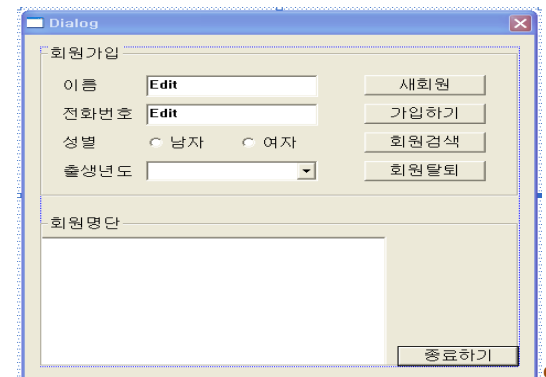
1절. 대화상자 이용하기

• 대화상자

- 프로그램 수행 중 사용자와 간단한 입력/출력을 하기 위해 사용되는 윈도우이다.
- 많은 양의 정보를 효율적으로 입/출력해주는 매개체, 혹은 말 그대로 사용자와 대화하는 상자

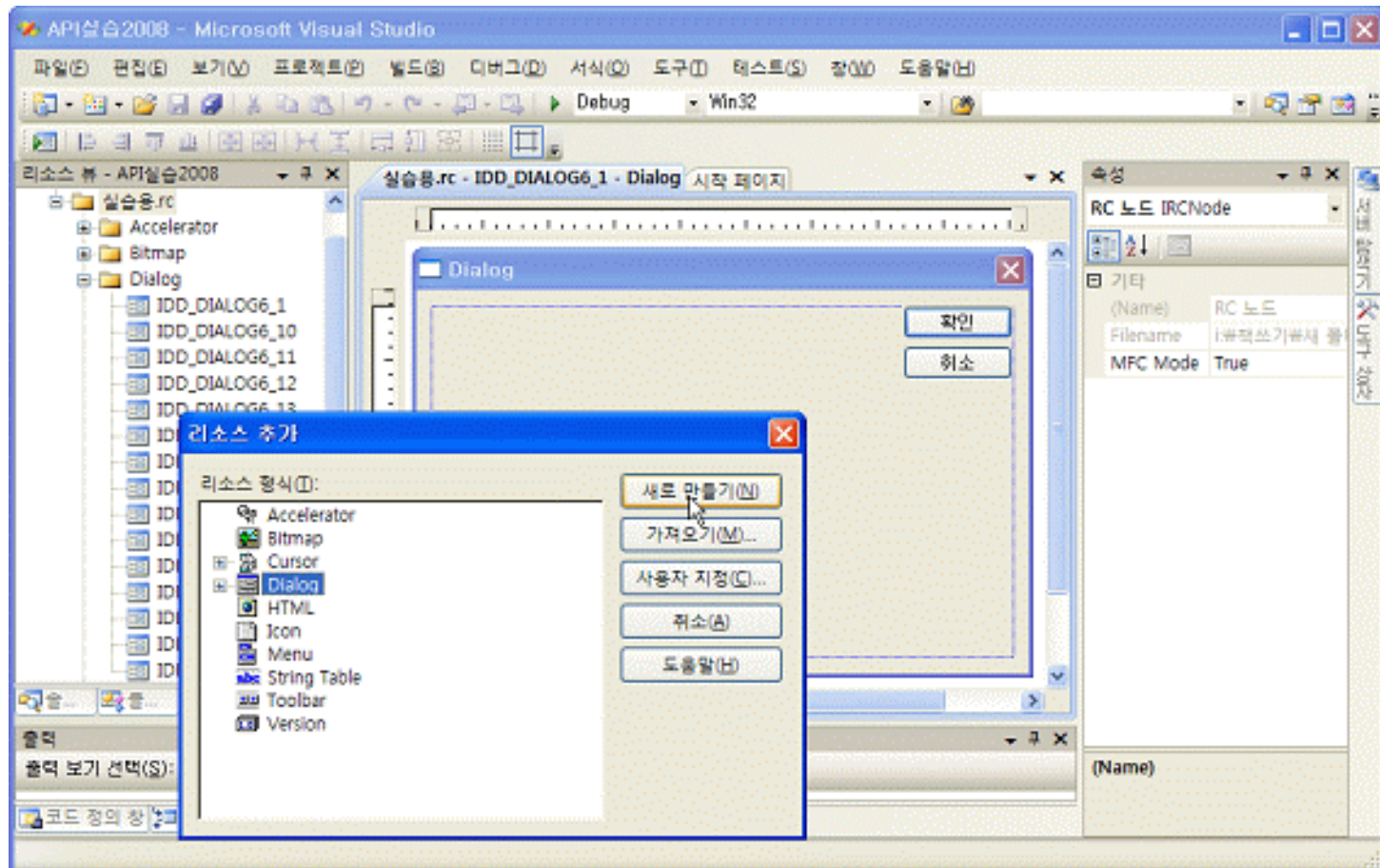
• 사용방법

- 리소스에서 새로운 대화상자 만들기
 - 리소스 형태로 대화상자 편집기로 컨트롤들을 디자인한다.
- 대화상자 띄우기
 - 대화상자를 메인 윈도우에서 띄운다.
- 대화상자에 대한 메시지 처리 함수 DialogProc()작성
 - 별도의 함수를 가지고 대화상자 메시지 처리



대화상자 만들기

- 리소스에서 대화상자 만들기 (Visual Studio 2010 환경)



대화상자 띄우기, 종료하기 함수

- 대화상자 띄우기

`DialogBox (hInst, MAKEINTRESOURCE(IDD_DIALOG1),
hWnd, DiagProc);`

- 대화상자를 생성하고 WM_INTDIALOG 메시지를 대화상자 프로시저로 보냄

- `int DialogBox (HINSTANCE hInstance, LPCTSTR lpTemplate, HWND hwnd, DLGPROC lpDialogFunc);`

- `hInstance` : 응용의 프로그램 인스턴스 값
- `lpTemplate` : 대화상자의 ID
- `Hwnd`: 윈도우의 핸들 값
- `lpDialogFunc` : 대화상자에서 발생하는 메시지 처리용 다이얼로그 함수
- 리턴값은 ID_OK 메시지

대화상자 띄우기, 종료하기 함수

- **메시지 처리용 다이얼로그 프로시저**
 - 대화상자 내에서 발생하는 메시지들을 처리하는 함수

BOOL CALLBACK DialogProc (HWND hDlg, UINT iMessage, WPARAM wParam, LPARAM lParam)

- **BOOL** 형을 반환한다. (메시지를 처리했으면 TRUE를 리턴하고 그렇지 않으면 FALSE를 리턴한다.)
- DefWindowProc 함수로 리턴하지 않는다. 다이얼로그에서는 내부에서 메시지를 처리했을 경우 TRUE를, 처리하지 않았을 경우 FALSE를 리턴한다.
- 메시지 처리:
 - **WM_INITDIALOG**를 사용한다. 윈도우 프로시저의 WM_CREATE 메시지 의미. 대화 상자에 필요한 초기화 작업
 - WM_COMMAND: 대화상자의 메시지 처리
 - **LOWORD (wParam):** 메시지를 보낸 컨트롤의 ID
 - **HIWORD (wParam):** 통지 코드

대화상자 띄우기, 종료하기 함수

- 대화상자 종료하기 함수

`EndDialog(hDlg, 0);`

- `BOOL EndDialog(HWND hDlg, int nResult);`
 - `nResult` : 0 (대화상자 종료상태 표시)

6-1 대화상자 띄우기

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

```
LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
BOOL CALLBACKDlg6_1Proc(HWND, UINT, WPARAM, LPARAM);
```

```
HINSTANCE hInst;
```

```
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
    LPSTR lpszCmdLine, int nCmdShow)
{
    HWND    hwnd;
    MSG     msg;
    WNDCLASS WndClass;
    hInst = hInstance;
    ...종락...
}
```


6-1 대화상자 띄우기

```

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

    static HINSTANCE hInstance;

    switch (iMsg) {
    case WM_CREATE :
        hInstance = hInst;
        break; ;

    case WM_LBUTTONDOWN :
        DialogBox (hInstance, MAKEINTRESOURCE (IDD_DIALOG6_1),
                  hwnd,Dlg6_1Proc);

        break;
    }
    return DefWindowProc (hwnd, iMsg, wParam, lParam) ;
}

```

6-1 메시지처리 함수

```

BOOL CALLBACK Dlg6_1Proc (HWND hDlg, UINT iMsg,
                          WPARAM wParam, LPARAM lParam)
{
    switch(iMsg){

    case WM_INITDIALOG:
        return true;

    case WM_COMMAND:
        switch (LOWORD(wParam)) {
            case IDOK: // 버튼
                EndDialog(hDlg,0);
                break;

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

```

2절. 컨트롤 종류

컨트롤	설명
Static Text	정적 텍스트는 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	콤보박스는 데이터를 입력할 때 목록에서 하나를 선택하게 할 때 사용

포인터
 문자열
 그룹 박스
 체크 박스
 콤보 박스
 수평 스크롤
 스펜버튼
 슬라이더
 리스트
 탭
 리치 에디트
 달력
 사용자 정의

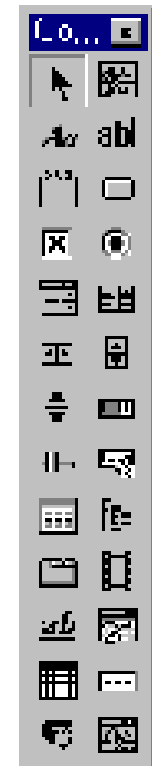


그림 상자
 에디트
 버튼
 라디오 버튼
 리스트 박스
 수직 스크롤
 프로그래스
 핫키
 트리
 애니메이션
 날짜선택
 IP 입력
 확장 콤보

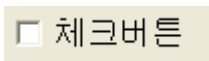
컨트롤 모양



버튼



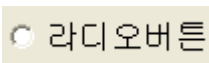
콤보박스



체크버튼



에디트박스



라디오버튼



트리컨트롤



리스트박스

3절. 버튼 컨트롤

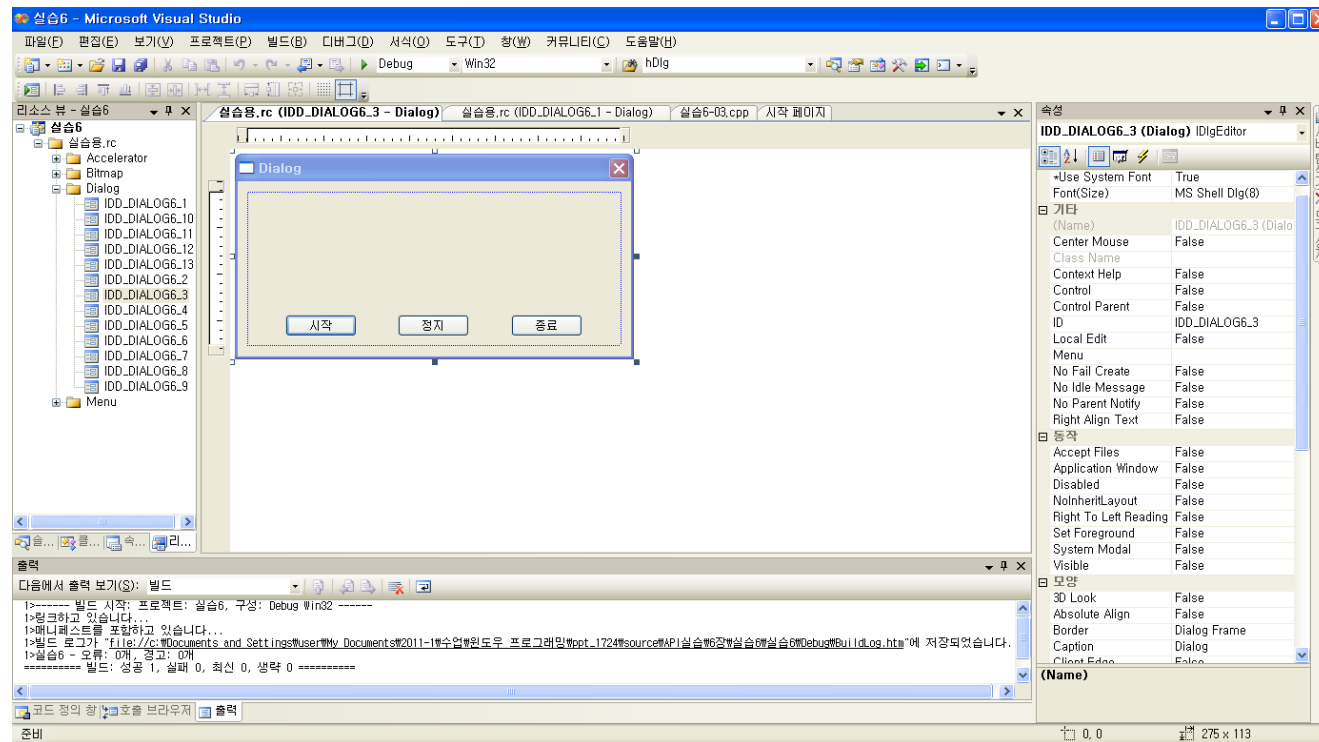
- **버튼 (Button)**
 - 버튼을 눌러 임의의 작업이 이루어진다.
 - 명령을 받아들이는 역할
- 대화상자의 컨트롤에서 발생하는 메시지는 WM_COMMAND
 - 컨트롤에서 오는 메시지 정보

메시지 보낸곳	wParam		lParam
	HIWORD	LOWORD	
컨트롤	컨트롤에 따른 통지 정보	컨트롤의 ID	컨트롤 핸들값

- **버튼 컨트롤의 통지 정보**
 - BN_CLICKED: 버튼이 클릭 되었음
 - BN_DBLCLK: 버튼이 더블클릭 되었음
 - BN_DISABLE: 버튼이 사용 불능 상태로 되었음
 - BN_HILITE: 사용자가 버튼을 선택했음
 - BN_SETFOCUS: 버튼이 포커스를 받았음
 - BN_KILLFOCUS: 버튼에서 포커스를 벗어날 때

6-2 버튼 이용하기

- 버튼의 편집 및 배치



6-2 버튼 클릭 메시지 처리

```

BOOL CALLBACK Dlg6_2Proc (HWND hDlg, UINT iMessage,
                          WPARAM wParam,LPARAM lParam)
{
    HDC hdc;
    switch(iMessage) {
        case WM_COMMAND:
            switch (LOWORD(wParam)) {
                case ID_BUTTON_PRINT:
                    hdc = GetDC(hDlg);
                    TextOut(hdc, 0, 0, "Hello World", 11);
                    ReleaseDC(hDlg, hdc);
                    break;
                case ID_BUTTON_END:
                    EndDialog(hDlg,0);
                    break;
                case IDCANCEL:
                    EndDialog(hDlg,0);
                    break;
            }
        }
    return false;
}

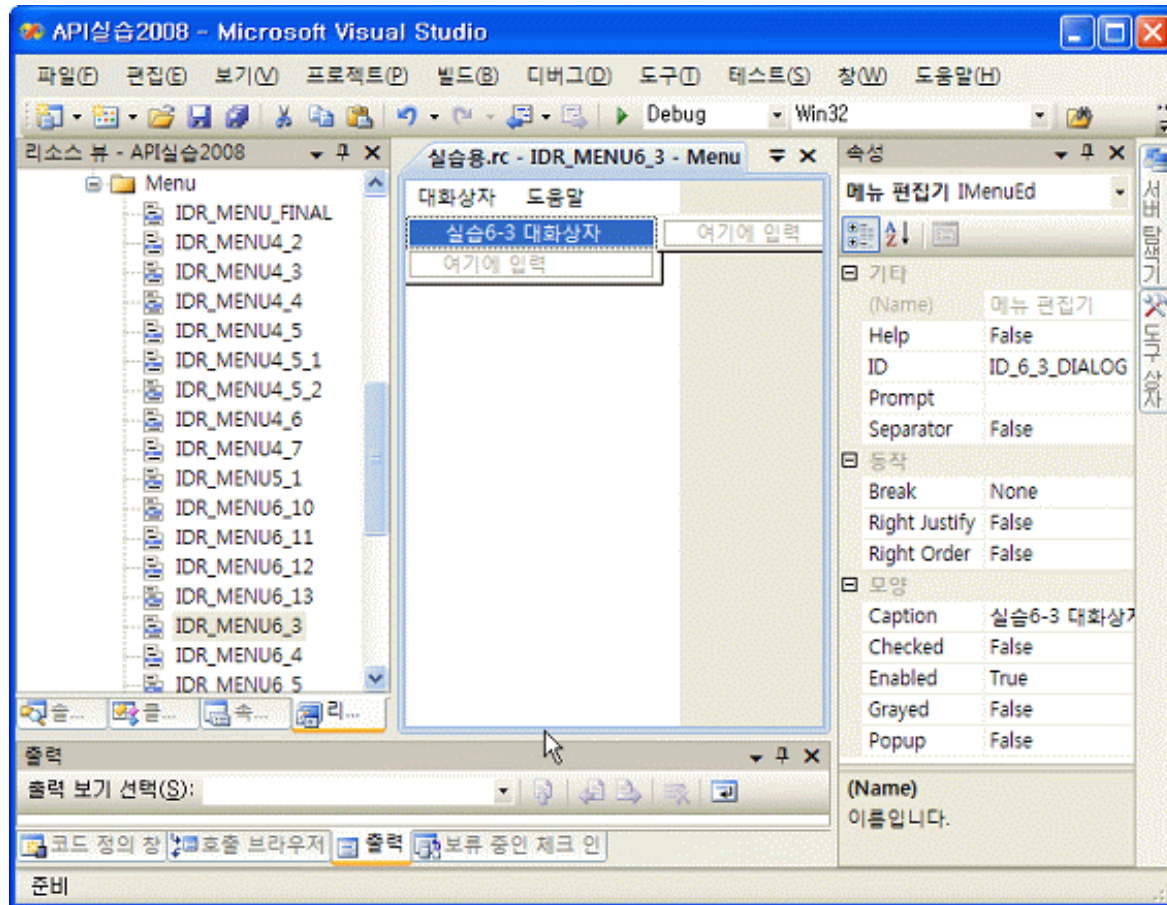
```

대화상자 초기화

- WM_INITDIALOG 메시지에 초기화
 - 대화상자가 처음 만들어질 때 발생하는 메시지
 - 대화상자 설정을 위한 변수의 초기화를 위해 주로 사용
 - wParam: 대화상자에서 제일 먼저 키보드 입력을 받을 컨트롤의 핸들값
 - lParam: 추가적인 정보를 저장하는데 일반적으로 0의 값을 가짐

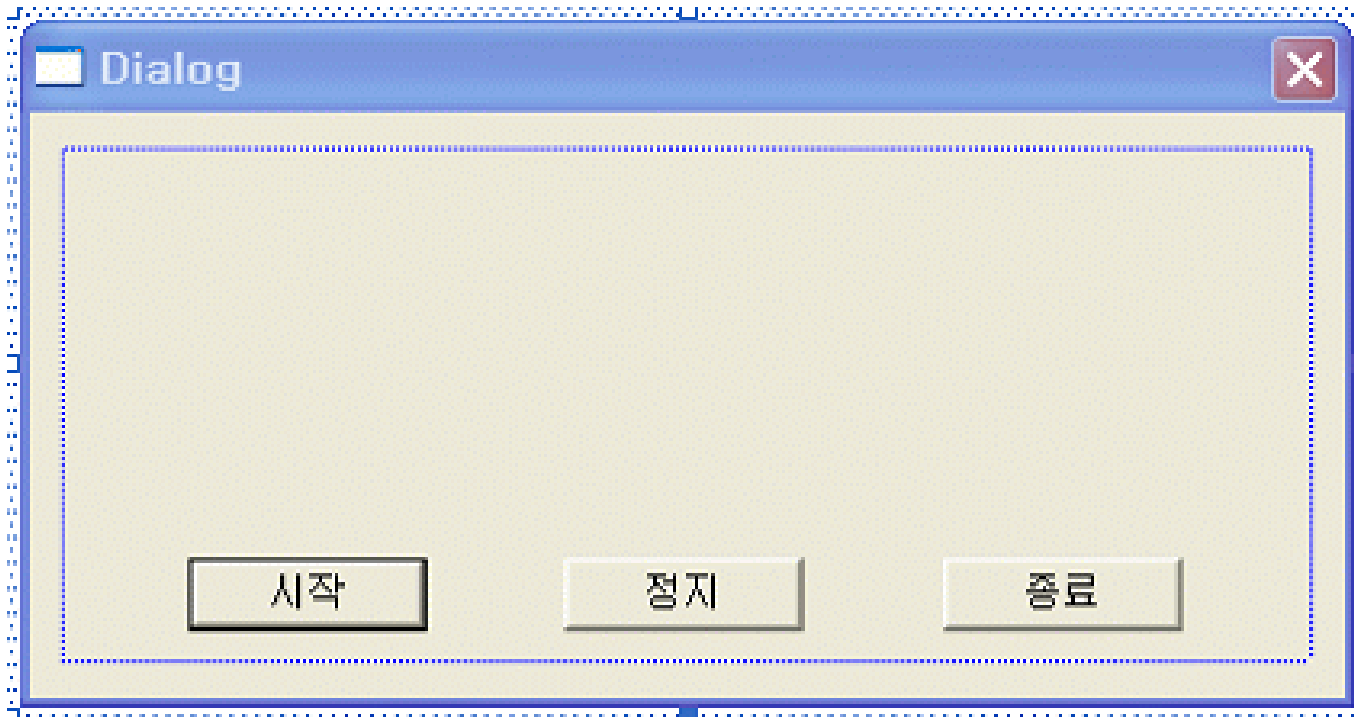
6-3 버튼 컨트롤 활성화/비활성화

- 메뉴 -> 메뉴항목 추가 : 대화상자 메뉴 작성(Visual Studio 2010 환경)



6-3 대화상자 편집하기

- ID가 IDD_DIALOG6_3인 대화상자를 새롭게 생성
- 세 개의 버튼을(시작, 정지, 종료) 배치



6-3 대화상자 띄우기

```
switch (iMsg) {  
  
case WM_CREATE :  
    break; ;  
  
case WM_COMMAND :  
    switch(LOWORD(wParam)) {  
        case ID_6_3_DIALOG:  
            DialogBox (hInst, MAKEINTRESOURCE (IDD_DIALOG6_3),  
                      hwnd, Dlg6_3Proc);  
            break;  
    }  
    break;  
}
```

6-3 대화상자 메시지처리 함수

```

BOOL CALLBACK Dlg6_3Proc (HWND hDlg, UINT iMsg,
                          WPARAM wParam, LPARAM lParam)
{
    HWND hButton;
    switch(iMsg)
    {
        case WM_INITDIALOG:
            hButton = GetDlgItem(hDlg, IDPAUSE);
            EnableWindow(hButton, FALSE); // 최초 정지 버튼은 비활성화
            break;
        case WM_COMMAND:
            switch (LOWORD(wParam))
            {
                case IDSTART: // 시작 버튼
                    hButton = GetDlgItem(hDlg, IDSTART);
                    EnableWindow(hButton, FALSE); // 시작버튼 비활성화
                    hButton = GetDlgItem(hDlg, IDPAUSE);
                    EnableWindow(hButton, TRUE); // 정지버튼 활성화
                    break;
            }
    }
}

```

대화상자 메시지처리 함수

```

    case IDPAUSE: // 정지 버튼
        hButton = GetDlgItem(hDlg, IDSTART);
        EnableWindow(hButton, TRUE); // 시작버튼 활성화
        hButton = GetDlgItem(hDlg, IDPAUSE);
        EnableWindow(hButton, FALSE); // 정지버튼 비활성화
        break;

    case IDCLOSE: // 종료 버튼
        EndDialog(hDlg,0);
        break;
    case IDCANCEL:
        EndDialog(hDlg,0);
        break;
    }
    break;
}
return 0;
}

```

컨트롤 관련 함수

• 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
- 사용 예: ID_START 아이디를 가진 컨트롤의 핸들 가져오기
 HWND hButton;
 hButton = GetDlgItem (hDlg, ID_START);

• GetDlgCtrlID 함수

- 특정 컨트롤의 윈도우 핸들로부터 컨트롤 ID 구함
- **int GetDlgCtrlID (HWND hWndCtrl);**
 - hWndCtrl: ID를 구할 컨트롤의 윈도우 핸들
 - 리턴값: 컨트롤의 ID
- 사용 예: hButton 컨트롤의 ID 가져오기
 int id;
 id = GetDlgCtrlID (hButton);

컨트롤 관련 함수

- **EnableWindow 함수**

- 컨트롤을 사용가능 상태 또는 사용불능 상태로 만듦
- **BOOL EnableWindow (HWND hWnd, BOOL bEnable);**
 - hWnd: 컨트롤의 핸들
 - bEnable: 상태 설정 값, TRUE 면 사용 가능 상태, FALSE면 사용 불능 상태
- 사용 예: hButton 컨트롤을 사용 불능 상태로 만듦
hButton = GetDlgItem (hDlg, ID_START);
EnableWindow (hbutton, FALSE);

4절. 에디트 박스 컨트롤

- **에디트 박스 컨트롤**

- 사용자의 키보드 입력 또는 출력을 위한 편집창
- 에디트 박스에서 오는 통지 정보
 - EN_CHANGE: 에디트 박스내의 내용이 변하였음
 - EN_HSCROLL: 에디트 박스의 수평스크롤바를 선택하였음
 - EN_VSCROLL: 에디트 박스의 수직스크롤바를 선택하였음
 - EN_SETFOCUS: 에디트 박스가 포커스를 받았음

컨트롤 관련 함수

- 컨트롤 윈도우에서 텍스트를 얻어오는 함수

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

- hDlg: 컨트롤을 가지고 있는 대화상자의 핸들
- nIDDlgItem: 컨트롤의 ID
- lpString: 얻어낸 텍스트 스트링을 저장할 버퍼의 주소
- nMaxCount: lpString이 가리키는 버퍼의 크기

- 컨트롤 윈도우에 텍스트를 출력하는 함수

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

- hDlg: 컨트롤을 가지고 있는 대화상자의 핸들
- nIDDlgItem: 컨트롤의 ID
- lpString: 출력할 텍스트 스트링의 시작 주소

- 사용 예) char word[100];

GetDlgItemText (hDlg, IDC_EDIT_SOURCE, word, 100);

SetDlgItemText (hDlg, IDC_EDIT_COPY, word);

컨트롤 관련 함수

- 컨트롤 윈도우에서 문자열을 정수값으로 변환하여 읽어오는 함수

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

- hDlg: 컨트롤을 가지고 있는 윈도우 핸들
- nIDDlgItem: 컨트롤의 ID
- lpTranslated: 변환의 성공여부 리턴받는 변수, 변환되면 TRUE, 아니면 FALSE 로 설정된다. (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 출력하는 함수

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

- 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

컨트롤 관련 함수

- 사용 예)

int x, y

```
x = GetDlgItemInt (hDlg, IDC_X, NULL, FALSE);
```

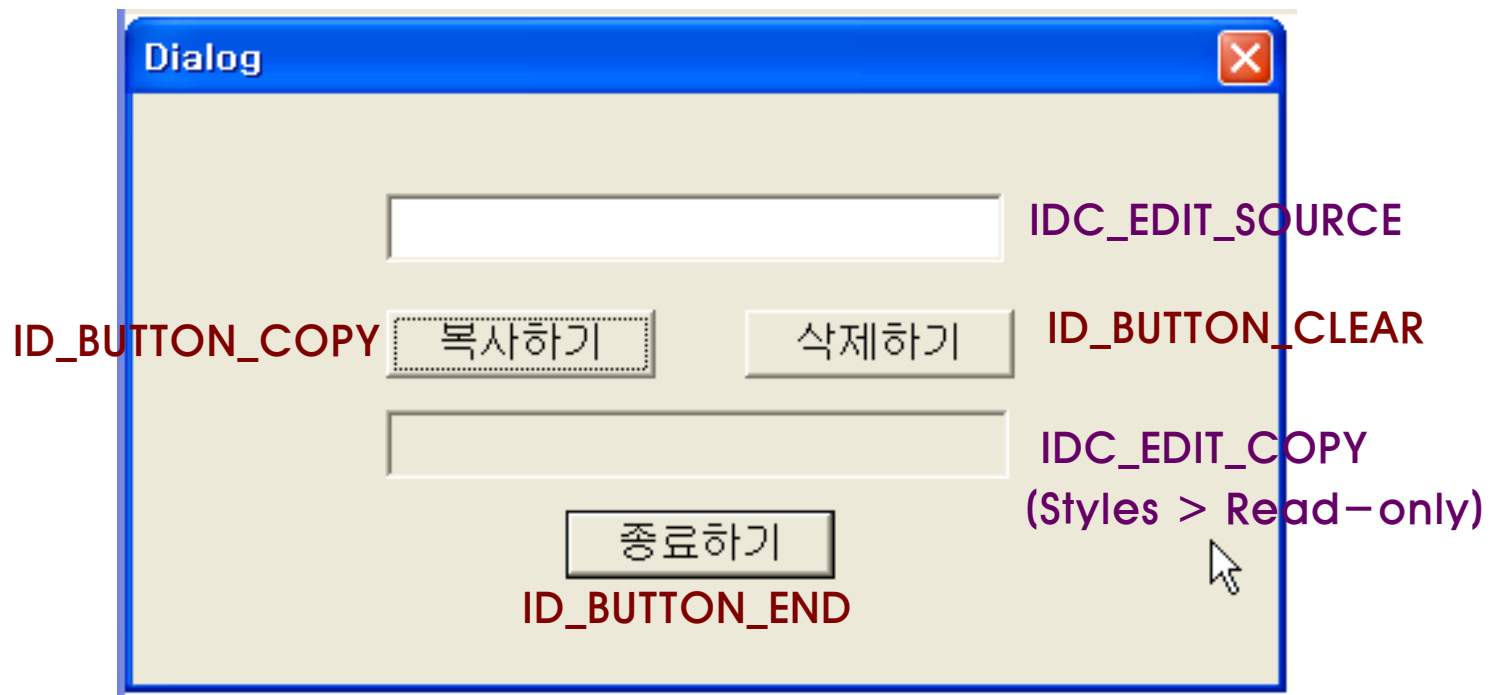
// hDlg 윈도우의 IDC_X라는 id를 가진 컨트롤에 저장된 값을 x에 읽어온다.

```
SetDlgItemInt (hDlg, IDC_Y, y, TRUE);
```

// hDlg 윈도우의 IDC_Y 라는 id를 가진 컨트롤에 y 값을 저장한다.

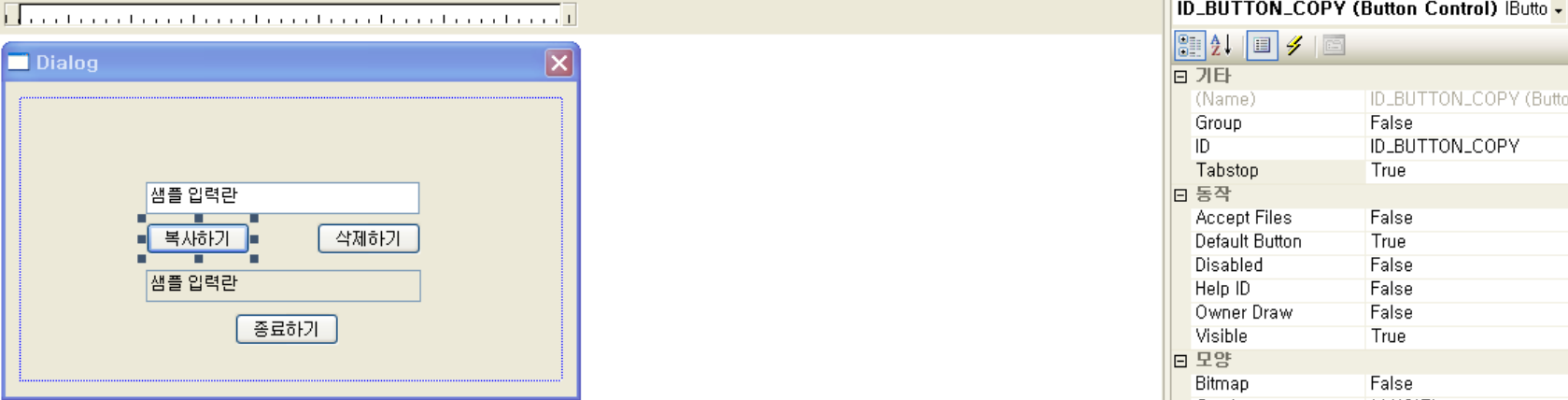
6-4 에디트박스에 문자열 복사하기

- 대화상자에 컨트롤 배치하기



6-4 에디트박스에 문자열 복사하기

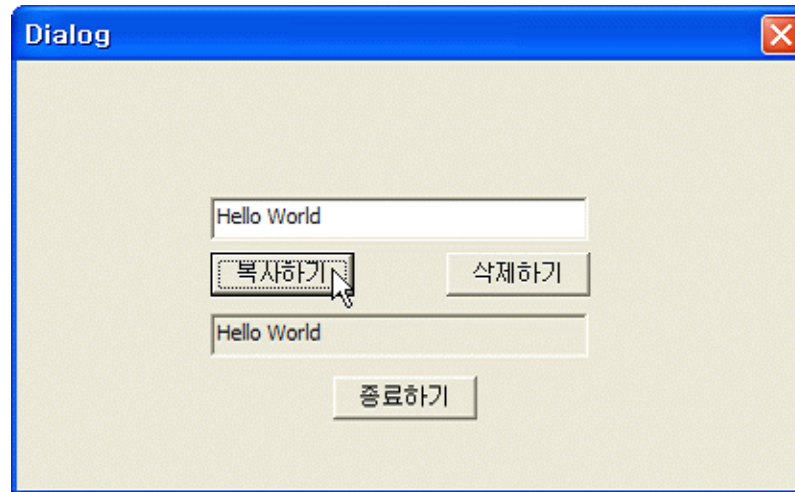
• 컨트롤 속성



The screenshot shows a Windows-style dialog box titled "Dialog" with a close button (X). Inside the dialog, there are two text input fields labeled "샘플 입력란" (Sample Input Field). Between them is a button labeled "복사하기" (Copy), which is currently selected with a mouse cursor. To the right of the "복사하기" button is another button labeled "삭제하기" (Delete). Below the input fields is a button labeled "종료하기" (End). The Properties window on the right displays the properties for the selected "ID_BUTTON_COPY (Button Control)".

ID_BUTTON_COPY (Button Control) IButto	
기타	
(Name)	ID_BUTTON_COPY (Butto
Group	False
ID	ID_BUTTON_COPY
Tabstop	True
동작	
Accept Files	False
Default Button	True
Disabled	False
Help ID	False
Owner Draw	False
Visible	True
모양	
Bitmap	False
Caption	복사하기
Client Edge	False
Flat	False
Horizontal Alignment	Default
Icon	False
Modal Frame	False
Multiline	False
Notify	False
Right Align Text	False
Right To Left Reading	False
Static Edge	False
Transparent	False
Vertical Alignment	Default

6-4 에디트박스에 스트링 복사



6-4 에디트 박스에 문자열 복사하기

```

BOOL CALLBACK Dlg6_4Proc(HWND hDlg,UINT iMsg,WPARAM
                                wParam,LPARAM lParam)
{
char word[100];
switch(iMsg) {

case WM_COMMAND:
switch (LOWORD(wParam)) {
case ID_BUTTON_COPY:
    GetDlgItemText (hDlg, IDC_EDIT_SOURCE, word, 100); // 스트링 복사
    SetDlgItemText (hDlg, IDC_EDIT_COPY, word);        // 스트링 출력
    break;

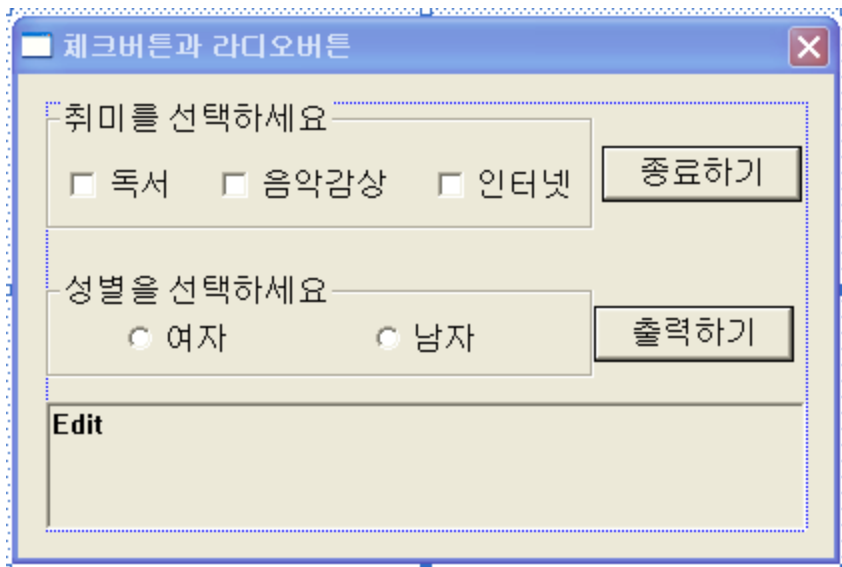
case ID_BUTTON_CLEAR:
    SetDlgItemText (hDlg, IDC_EDIT_SOURCE, "");         // 널 스트링 복사
    SetDlgItemText (hDlg, IDC_EDIT_COPY, "");           // 널 스트링 복사
    break;

case ID_BUTTON_END:
    EndDialog(hDlg,0);
    break;
}
break;
}
}

```

5절. 체크박스과 라디오버튼 컨트롤

- 대화상자에서 체크버튼과 라디오버튼 이용
 - 체크 버튼 : 복수 항목 선택 가능
 - 라디오 버튼 : 한 항목 만 선택



종류	ID
Static	IDC_STATIC
Static	IDC_STATIC
Check	IDC_CHECK_READING
Check	IDC_CHECK_MUSIC
Check	IDC_CHECK_INTERNET
Radio	IDC_RADIO_FEMALE
Radio	IDC_RADIO_MALE
Edit	IDC_EDIT_OUTPUT
Button	IDC_BUTTON_OUTPUT
Button	IDC_CLOSE

6-5 개인정보 선택, 출력하기

```

BOOL CALLBACK Dlg6_5Proc (HWND hDlg, UINT iMsg,
                          WPARAM wParam, LPARAM lParam)
{

    static int Check[3], Radio;
    char hobby[][30] = {"독서", "음악감상", "인터넷"};
    char gender[][30] = {"여자", "남자"};
    char output[200];

    switch(iMsg)
    {
        case WM_INITDIALOG:
            CheckRadioButton (hDlg,
                             IDC_RADIO_FEMALE, IDC_RADIO_MALE, IDC_RADIO_FEMALE);
            // 시작 버튼, 끝 버튼, 미리 체크할 버튼
            break;
    }
}

```

6-5 개인정보 선택, 출력하기

```

case WM_COMMAND:
    switch (LOWORD(wParam))
    {
    case IDC_CHECK_READING:
        Check[0] = 1 - Check[0];
        break;
    case IDC_CHECK_MUSIC:
        Check[1] = 1 - Check[1];
        break;
    case IDC_CHECK_INTERNET:
        Check[2] = 1 - Check[2];
        break;
    case IDC_RADIO_FEMALE:
        Radio = 0;                                // 0과 1 중에서 0선택(여성)
        break;
    case IDC_RADIO_MALE:
        Radio = 1;
        break;
    case IDC_BUTTON_OUTPUT:
        wsprintf (output, "선택한 취미는 %s %s %s입니다.
        \r\n선택한 성별은 %s 입니다.", Check[0]?hobby[0]:"",
        Check[1]?hobby[1]:"", Check[2]?hobby[2]:"", sex[Radio]);
        SetDlgItemText (hDlg, IDC_EDIT_OUTPUT, output);
        break;
    }
}

```

6-5 개인정보 선택, 출력하기

- 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 : 각각 그룹의 시작 버튼 아이디
 - nIDLastButton: 각각 그룹의 끝 버튼 아이디
 - nIDCheckButton: 선택될 버튼의 아이디

6-5 개인정보 선택, 출력하기

체크버튼과 라디오버튼

취미를 선택하세요

☒ 독서 ☐ 음악감상 ☒ 인터넷

종료하기

성별을 선택하세요

☒ 여자 ☐ 남자

출력하기

선택한 취미는 독서 인터넷 입니다.
선택한 성별은 여자 입니다.

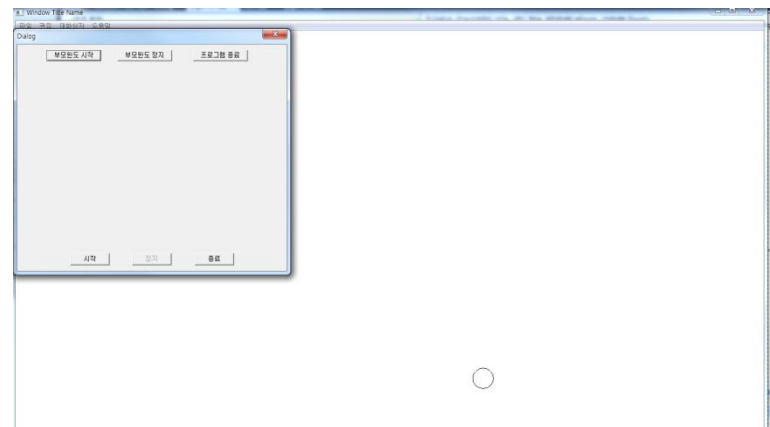
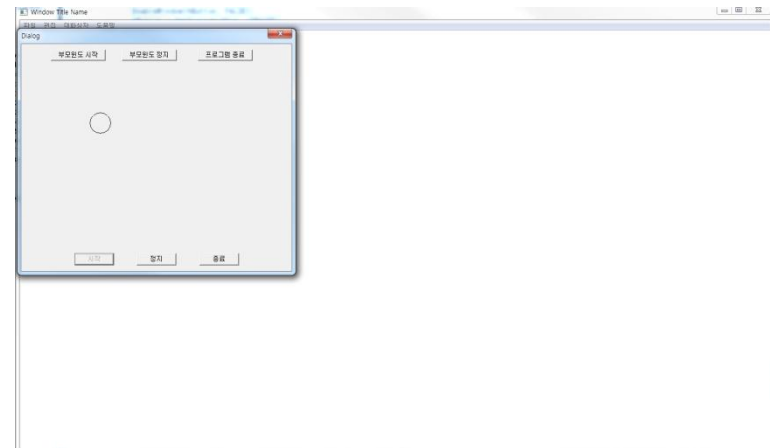
실습 6-1

• 제목

- 대화상자 안 또는 부모 윈도우에서 바운드 되는 원

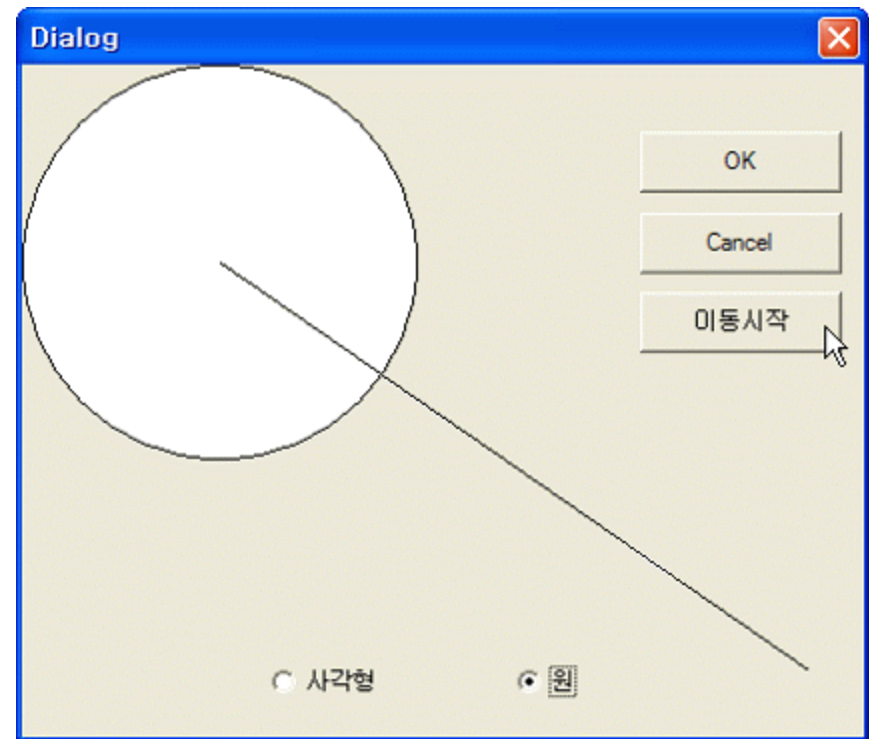
• 내용

- 먼저 대화상자를 추가하고 대화상자안에 여섯 개의 버튼을 만든다.
 - 버튼1: 대화상자에서 원의 바운드를 시작하기 위한 시작버튼
 - 버튼2: 대화상자에서 바운드 하고 있는 원을 멈추기 위한 정지버튼
 - 버튼3: 대화상자를 닫는 종료버튼
 - 버튼 4: 부모 윈도우에서 원의 바운드를 시작하기 위한 시작버튼
 - 버튼5: 부모 윈도우에서 바운드 하고 있는 원을 멈추기 위한 정지버튼
 - 버튼6: 프로그램을 닫는 종료버튼



실습 6-2

- 제목
 - 라디오버튼 연습
- 내용
 - 대화상자에는 사각형이 기본적으로 나타나고 **두개의 라디오 버튼**(사각형, 원)과 **세개의 버튼**(OK, Cancel, 이동시작)이 나타난다.
 - 라디오버튼 중 하나를 선택하면 선택한 버튼에 따라 도형 모양이 변한다.
 - 선택한 도형이 좌측 상단에 그려진다.
 - 대화상자의 클라이언트 영역 아무데나 왼쪽마우스버튼을 클릭하면 도형의 중심점부터 클릭한 위치까지 직선이 그려진다.
- 버튼
 - **OK**: 선을 그릴 수 있다.
 - **Cancel**: 도형이 제자리로 이동한다.
 - **이동시작**: 도형이 직선을 따라 자동으로 이동



실습 6-3

• 제목

- 버튼을 이용하여 캐릭터 이동하기

• 내용

- 다이얼로그 박스 왼쪽에는 화면을 설정하고 오른쪽에는 아래의 버튼을 만든다.

• 버튼

- 1: jump (점프)
- 2: Front (앞으로 이동)
- 3: Back (뒤로 이동)
- 4: Up (위로 이동)
- 5: Down (아래로 이동)

• 체크박스

- 6: 그리드 (화면에 그리드를 설정 / 해제)

• 라디오 버튼

- 7: 캐릭터 1 (이미지 사용)
- 8: 캐릭터 2 (이미지 사용)

• 라디오 버튼

- 9: 확대 (캐릭터 크기를 2배로 확대)
- 10: 축소 (캐릭터 크기를 0.5배로 축소)

• 라디오 버튼 (캐릭터 이동, 또는 정지)

- 11: 이동 (캐릭터가 현 위치에서 지그재그로 아래로/오른쪽으로 이동한다.)
- 12: 정지 (캐릭터가 정지한다.)
- 13: 제자리 (캐릭터가 초기 위치로 이동한다.)
- 14: 빨리 이동 (캐릭터가 이동 속도의 2배 속도로 이동한다)

• 버튼

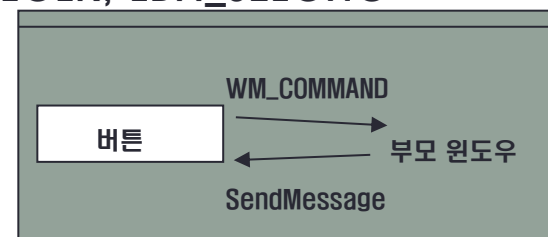
- 15: exit (프로그램 종료)

6절. 콤보 박스 컨트롤

- **콤보 박스 컨트롤은**
 - 사용자의 키보드 입력 또는 출력을 위한 편집창
 - 여러 항목들의 리스트를 나열하여 보여주는 컨트롤
 - 콤보 박스 컨트롤을 선택하면 WM_COMMAND 메시지 발생
- **콤보 박스에서 오는 통지 정보**
 - **CBN_DROPDOWN**: 콤보 박스에 등록된 항목들이 아래로 펼쳐짐
 - **CBN_DBLCLK**: 아래로 펼쳐진 항목 리스트에서 하나를 더블클릭으로 선택했음
 - **CBN_EDITCHANGE**: 콤보 박스의 텍스트 편집 공간에 텍스트를 추가하거나 수정하였음
 - **CBN_SELCHANGE**: 사용자가 항목 리스트에서 하나를 선택하였음
- **컨트롤에 메시지를 보내는 함수는 SendMessage**
 - 컨트롤이나 윈도우에 특정 명령을 내리기 위해 메시지를 보내고 그 결과는 함수가 반환하는 값

SendMessage()

- 메시지를 메시지큐에 넣지 않고 바로 윈도우 프로시저로 보냄
 - 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
 - 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



콤보박스에 보내는 메시지

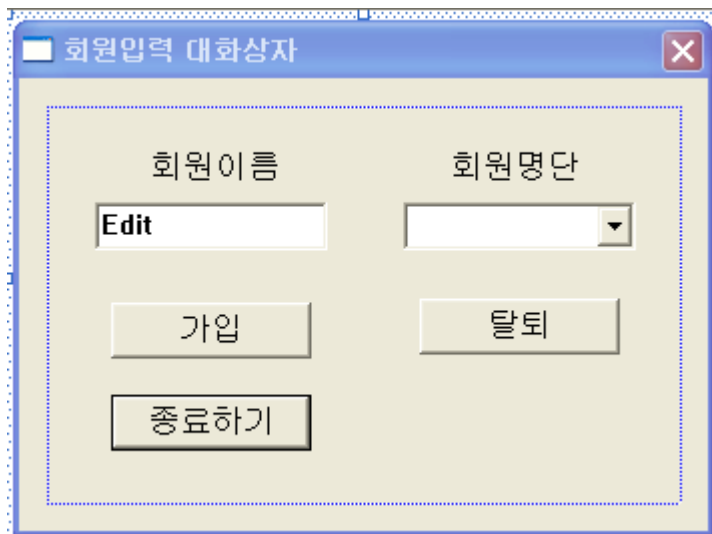
- **CB_ADDSTRING**: 콤보 박스에 텍스트를 아이템으로 추가하는 메시지로써 리스트의 마지막에 추가된다.
 - wParam: 사용하지 않음
 - lParam: 텍스트 스트링의 시작 주소
- **CB_DELETESTRING**: 콤보 박스에 있는 아이템들 중 하나를 삭제하는 메시지
 - wParam: 삭제하기 원하는 아이템의 인덱스로 0부터 시작한다.
 - lParam: 0

콤보박스에 보내는 메시지

- **CB_GETCOUNT**: 콤보 박스의 아이템 리스트에 들어 있는 아이템의 개수를 얻기 위한 메시지로 개수 값은 SendMessage()함수가 리턴한다.
 - wParam: 0
 - lParam: 0
- **CB_GETCURSEL**: 현재 선택된 아이템의 인덱스 번호를 얻기 위한 메시지로 인덱스 번호는 SendMessage()함수가 리턴한다.
 - wParam: 0
 - lParam: 0
- **CB_SETCURSEL**: 콤보 박스 컨트롤의 텍스트 편집 공간에 지정한 항목의 텍스트를 보여준다.
 - wParam: 나타내고자 하는 항목의 인덱스 번호
 - lParam: 사용않음

6-6 콤보박스로 회원명단 관리하기

- 대화상자에 콤보박스 그리기
 - 회원이름을 넣고 가입하면 회원명단에 추가됨



종류	ID
Static	IDC_STATIC
Static	IDC_STATIC
Edit	IDC_EDIT_NAME
Combo	IDC_COMBO_LIST
Button	IDC_BUTTON_INSERT
Button	IDC_BUTTON_DELETE
Button	IDC_CLOSE

6-6 콤보박스로 회원명단 관리하기

```
static int selection;
static HWND hCombo;

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 (strcmp(name, "")) // 이름이 들어 왔으면, 이 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;
    }
}
```

7절. 리스트 박스 컨트롤

- 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 박스에서 오는 통지 정보
 - LBN_DBLCLK: 리스트 박스의 여러 아이템들 중 하나를 더블클릭 했음
 - LBN_SELCHANGE: 아이템들중 하나가 선택되었음
 - LBN_SETFOCUS: 리스트 박스가 포커스를 받았음
 - LBN_KILLFOCUS: 리스트 박스가 포커스를 잃었음
 - WM_DELETEITEM: 리스트 박스의 여러 아이템들 중 하나가 삭제 되었음

리스트 박스에 보내는 메시지

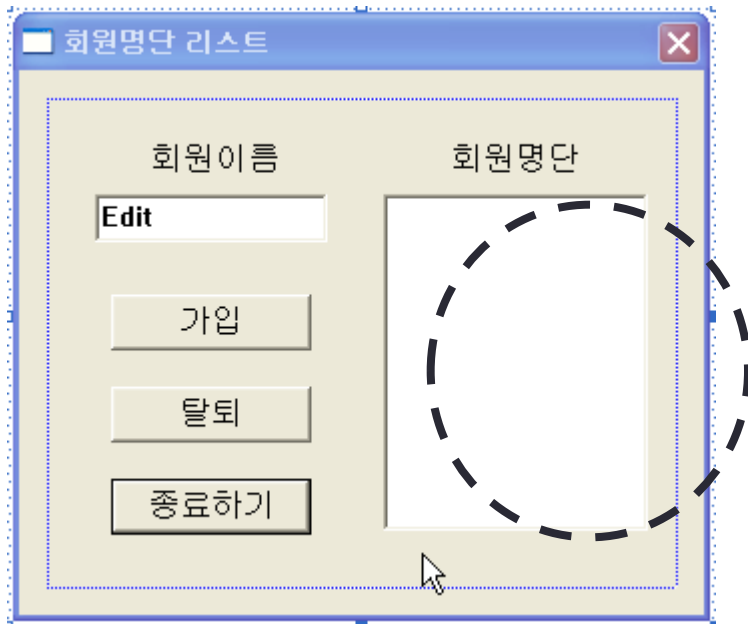
- **LB_ADDSTRING**: 리스트 박스에 텍스트를 아이템으로 추가하는 메시지로써 리스트의 마지막에 추가된다.
 - wParam: 사용하지 않음
 - lParam: 텍스트 스트링의 시작 주소
- **LB_DELETESTRING**: 리스트 박스에 있는 아이템들 중 하나를 삭제하는 메시지
 - wParam: 삭제하기 원하는 아이템의 인덱스로 0부터 시작한다.
 - lParam: 0
- **LB_GETCOUNT**: 리스트 박스의 아이템 리스트에 들어 있는 아이템의 개수를 얻기 위한 메시지로 개수 값은 SendMessage()함수가 리턴한다.
 - wParam: 0
 - lParam: 0

리스트 박스에 보내는 메시지

- **LB_GETCURSEL:** 현재 선택된 아이템의 인덱스 번호를 얻기 위한 메시지로 인덱스 번호는 SendMessage() 함수가 리턴한다.
 - wParam: 0
 - lParam: 0
- **LB_GETTEXT:** 아이템 리스트 중 wParam에서 지정한 인덱스 아이템의 텍스트를 얻어오는 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 리스트 중 추가될 위치의 인덱스 번호
 - lParam: 텍스트 스트링의 시작 주소

6-7 리스트 박스로 명단관리

- 대화상자에 리스트박스 그리기



종류	ID
Static	IDC_STATIC
Static	IDC_STATIC
Edit	IDC_EDIT_NAME
List Box	IDC_LIST_NAME
Button	IDC_BUTTON_INSERT
Button	IDC_BUTTON_DELETE
Button	IDC_CLOSE

6-7 리스트 박스로 명단관리

```
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 (strcmp(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;
    }
}
```

윈도우 클래스 구조체 값 변경

- 윈도우의 클래스를 등록한 후 WNDCLASS 구조체의 값을 변경할 때
 - DWORD **SetClassLong** (HWND hWnd, int nIndex, LONG dwNewLong);
 - hWnd: 윈도우 핸들
 - nIndex: 변경하고자 하는 값

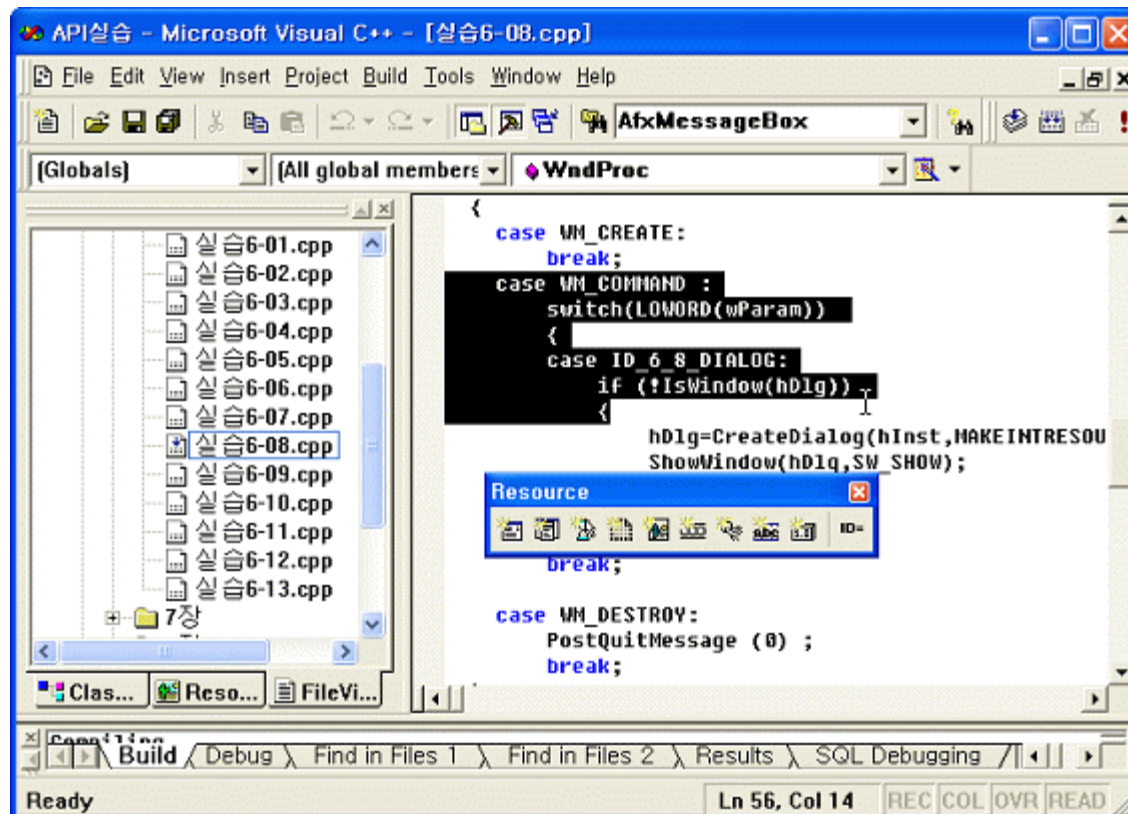
• GCL_CBCLSEXTRA	cbClsExtra 멤버값
• GCL_CBWNDXTRA	cbWndExtra 멤버값
• GCL_HBRBACKGROUND	hbrBackGround 멤버값
• GCL_HCURSOR	hCursor 멤버값
• GCL_HICON	hIcon 멤버값
• GCL_HICONSM	hIconSm 멤버값
• GCL_HMODULE	hInstance 멤버값
• GCL_MENUNAME	hMenu 멤버값
• GCL_STYLE	style 멤버값
• GCL_WNDPROC	lpfnWndProc 멤버값
 - 배경색 바꾸기
 - HBRUSH hGreen;
 - hGreen = CreateSolidBrush (RGB(0, 255, 0));
 - SetClassLong (hWnd, GCL_HBRBACKGROUND, (LONG)hGreen

8절. 모델리스 대화상자

- 모달(Modal)형 대화상자와 모델리스(Modaless)형 대화상자
- 모달형 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 실행할 수 있다.
 - 대부분의 대화상자가 이러한 특징을 가지고 있으며 대표적인 예로 MessageBox()함수에 의해서 만들어진 대화상자가 있다.
- 모델리스(Modaless)형 대화상자
 - 해당 대화상자를 닫지 않아도 다른 윈도우로 전환할 수 있는 특징을 갖는다.
 - 모델리스형 대화상자의 대표적인 예가 많은 프로그램에서 제공하는 "찾기"메뉴항목
 - "찾기"메뉴항목은 보통 해당 내용을 찾은 후 편집작업 등을 수행하고 다음 찾기를 하기 때문에 모델리스형 대화상자가 더 바람직하다.

8절. 모델리스 대화상자

- 대화상자가 나타나도 부모 윈도우를 선택할 수 있는 대화상자



모델리스 대화상자 관련 함수

- 모델리스 대화상자를 생성하는 함수

```

HWND CreateDialog ( HINSTANCE hInstance,           // 응용의 핸들
                    LPCTSTR lpTemplate,           // 대화상자의 ID
                    HWND hWndParent,             // 윈도우 핸들
                    DLGPROC lpDialogFunc        // 메시지 처리 함수
                    );

```

대화상자를 만들고 바로 대화상자의 핸들값을 리턴한다.

- 모델리스 대화상자를 보이거나 숨기는 함수

```

BOOL ShowWindow ( HWND hwnd,                       // 윈도우 핸들
                  int nCmdShow    // SW_HIDE: 감춤, SW_SHOW: 나타냄
                  );

```

- 모델리스 대화상자 종료하기 함수

```

BOOL DestroyWindow (HWND hWnd);

```

6-8 모델리스 대화상자

```
LRESULT CALLBACK WndProc (HWND hwnd, UINT iMsg,
                          WPARAM wParam, LPARAM lParam)
{
    HWND hDlg = NULL;
    switch (iMsg)
    {
        case WM_COMMAND :
            switch(LOWORD(wParam))
            {
                case ID_6_8_DIALOG:
                    if (!IsWindow(hDlg))
                    {
                        hDlg = CreateDialog (hInst,
                                              MAKEINTRESOURCE(IDD_DIALOG6_8), hwnd,Dlg6_8Proc);
                        ShowWindow(hDlg,SW_SHOW);
                    }
                    break;
            }
        break;
    }
    break;
}
```

6-8 모델리스 대화상자

```
BOOL CALLBACK Dlg6_8Proc (HWND hDlg, UINT iMsg,  
                           WPARAM wParam,LPARAM lParam)  
{  
    ...생략...  
    case IDCLOSE:  
        DestroyWindow(hDlg);  
        hDlg=NULL;  
        break;  
  
    case IDCANCEL:  
        DestroyWindow(hDlg);  
        hDlg=NULL;  
        break;  
    ...생략...  
}
```


실습 6-4

- 제목
 - 콤보 박스와 리스트 박스 사용하기
- 내용
 - **에디트 박스**: 문자열 입력하기
 - **콤보 박스**: 에디트 박스에서 입력받은 문자열 출력하기
 - 버튼 1: 에디트 박스에 입력받은 도형을 콤보 박스에 출력하기
 - 버튼 2: 콤보 박스에서 선택된 문자열을 버튼 2를 누르면 에디트 박스에 출력하도록 한다.
 - **리스트 박스**: 색상 리스트 (빨강, ..., 검정, 8개 색상)
 - 버튼 3: 리스트 박스에서 선택된 색상을 윈도우 바탕색 색상 변경에 사용하도록 한다.
 - **버튼 4**: 종료하기 (프로그램 혹은 대화 상자 종료)

실습 6-5

• 제목

- 모델리스 대화상자를 이용하여 계산기 구현하기

• 내용

- 에디트 박스 컨트롤에 숫자를 직접 입력하는 형태
- 버튼으로 숫자를 입력하는 계산기
 - 사칙연산은 연속으로 할 수 있도록 한다.
 - $3+4+5+6\cdots$ 또는 $3+4=7+2=9\cdots$
- 기존 계산기에 버튼 추가한다.
- **버튼 1** (R 버튼): 입력된 숫자의 순서를 바꾸는 버튼
 - 예) $12345.67 \rightarrow$ (Reverse 버튼) $\rightarrow 76.54321$
- **버튼 2** (CE 버튼): 마지막으로 입력한 **숫자를 지운다.**
 - 예) $123 + 2 + 3 \rightarrow$ (CE 버튼) $\rightarrow 123 + 2$
- **버튼 3** (C 버튼): 리셋한다.
- **버튼 4** (Binary): 입력된 숫자를 2진수로 변환
- **버튼 5** (Decimal): 입력된 숫자 10진수로 변환
- **버튼 6** (종료하기): 프로그램을 종료한다.



실습 6-6

• 제목

- 회원 관리 프로그램 만들기

• 내용

- 다음의 내용을 입력받는다.

- **회원 이름, 전화번호**: 에디트 박스
- **성별**: 라디오 버튼
- **출생년도**: 콤보 박스
- **회원 명단**: 리스트 박스
- **새회원**: 버튼 – 회원 이름과 전화번호 에디트 박스가 비워져서 새 회원정보를 받을 수 있다.
- **가입**: 버튼 – 회원명단에 새로운 회원 정보가 추가된다.
- **정렬**: 회원들의 이름에 따라 정렬한다.
- **탈퇴**: 버튼: 회원 한 명을 선택하고 탈퇴 버튼을 누르면 해당 회원의 정보가 명단에서 사라진다.

이름	전화번호	성별	출생년도
이보석	010-8374-5867	여자	1990
김철수	011-1234-5678	남자	1990
이수정	010-9876-5432	남자	1990

실습 6-7

- 제목
 - 간단한 타일 툴 프로그램 구현하기
- 내용
 - 모델리스 대화상자
 - 맵의 크기 설정하는 에디트 박스 (가로와 세로의 크기)
 - 가로와 세로의 타일 개수를 설정하는 에디트 박스 (가로와 세로의 타일 개수)
 - 타일 이미지 선택 버튼 (버튼에 이미지를 입혀서 선택하게 한다)
 - 타일을 선택한 후 화면의 그리드 칸을 클릭하면 선택된 이미지가 그려진다.
 - 메뉴:
 - 읽기: 타일의 데이터가 저장된 파일을 읽어 화면에 출력한다.
 - 저장: 만들어진 타일 맵을 저장한다.
 - 종료: 프로그램 종료

실습 6-7

• 내용

• 버튼에 이미지 입히기

- 버튼의 속성에서 모양-이미지를 true로 바꿈

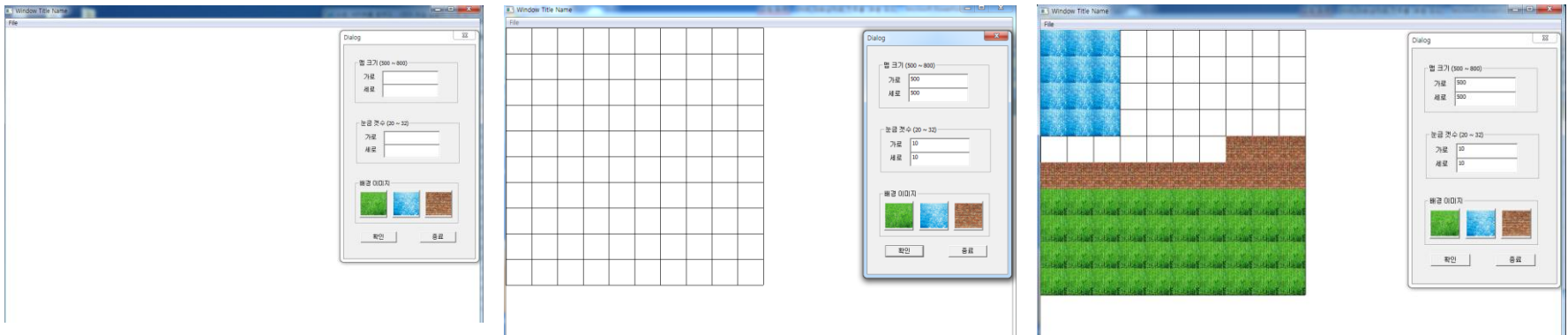
HBITMAP hBit;

HWND hButton;

hBit = LoadBitmap (hInst, MAKEINTRESOURCE(IDB_BITMAP1)); // 비트맵 로드

hButton = GetDlgItem (hwnd, IDC_BUTTON1); // 버튼의 핸들 얻기

SendMessage (hButton, BM_SETIMAGE, 0, (LPARAM)hBit); // 버튼에 메시지 보내기



실습 6-7

- 내용

- 파일에 저장하기

- 표준 입출력 함수 사용
 - 파일 열기/닫기: fopen 함수
 - 파일에 숫자 저장하기

```
int x, y, z;
```

```
FILE *fp = fopen ( "test.txt" , "w" );
```

```
fprintf (fp, "%d %d %d" , x, y, z);
```

- 파일에서 숫자 읽기

```
int x, y, z;
```

```
FILE *fp = fopen ( "test.txt" , "r");
```

```
fscanf (fp, "%d %d %d" , &x, &y, &z);
```