

2013년도 윈도우 프로그래밍

# 제4장 윈도우 메뉴

## ▶ 학습목표

- ▶ 리소스 파일을 만들고 윈도우에 메뉴를 등록하는 방법을 배운다.
- ▶ 메뉴에서 발생하는 메시지를 처리하는 방법을 배운다.
- ▶ 공용 대화상자의 종류를 알아보고 이를 적용해 프로그램의 완성도를 높인다.
- ▶ 윈도우에 등록된 메뉴를 프로그램 실행 중에 수정하는 방법을 배운다.

## ▶ 내용

- ▶ 메뉴 만들기
- ▶ 메뉴 사용하기
- ▶ 공용 대화상자 이용하기
- ▶ 메뉴 수정하기

# 리소스 개요

- ▶ 리소스(resource)는 메뉴, 아이콘, 커서, 다이얼로그, 액셀러레이터, 비트맵, 문자열, 버전정보 등 사용자 인터페이스를 구성하는 자원들로 읽기 전용 정적 데이터를 말한다.
- ▶ 리소스는 프로그램 실행 중 변경되지 않는 정적 데이터로 C/C++과 같은 언어로 관리하지 않고 리소스 스크립트(Resource Script; .rc)파일로 관리한다.
- ▶ 윈도우즈 프로그램의 큰 특징 중 하나가 소스코드와 리소스가 분리되어 있다는 것이다.
- ▶ DOS기반 프로그래밍에서는 리소스를 정의할 때 복잡한 배열 등을 만들어 사용하거나 외부 파일로 만들어 둔 후 프로그램 실행 시에 읽어 들이도록 했다.
- ▶ 그러나 윈도우즈 프로그래밍에서는 이런 데이터들을 리소스로 만들어 놓고 소스코드와 별도로 컴파일하며, 링크 시 최종 실행파일에 합쳐진다

# 리소스 개요

Workspace/기본 cpp  
파일 만들기

- Win32 Application 타입의 프로젝트 만들기
- 윈도우를 띄우는 기본 소스 코드 작성하기

File/New/Files/Resource  
Script

- Resource Script를 생성하기

Insert/Resource

- 생성한 리소스를 삽입하기

Resource 종류 선택

- 리소스 선택
  - 메뉴/아이콘/비트맵/툴바/커서 등

Resource 작성

- 리소스 편집기 사용
  - 캡션 및 id 설정하기

메인 코드 수정

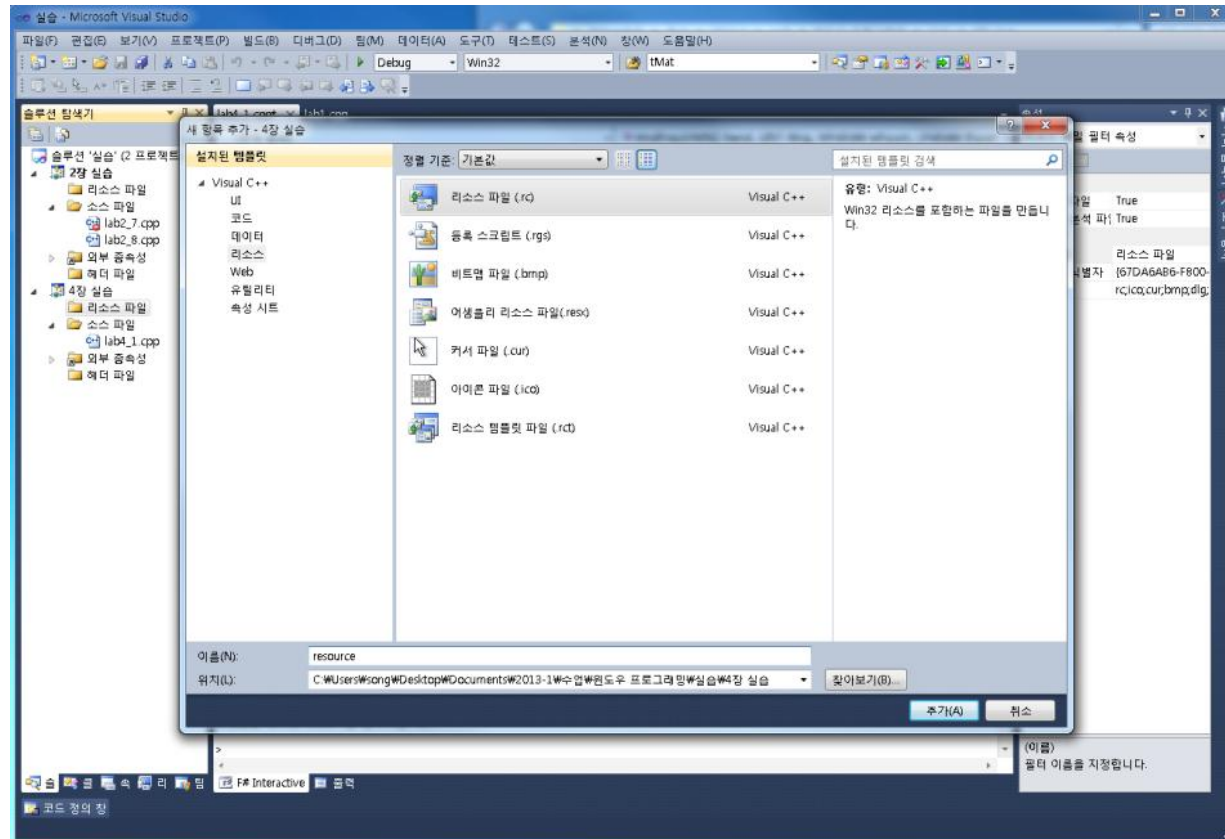
- 메인 프로그램에서 생성한 리소스 삽입
  - 리소스가 불러왔을 때 command 수행

# 1 절. 메뉴 만들기

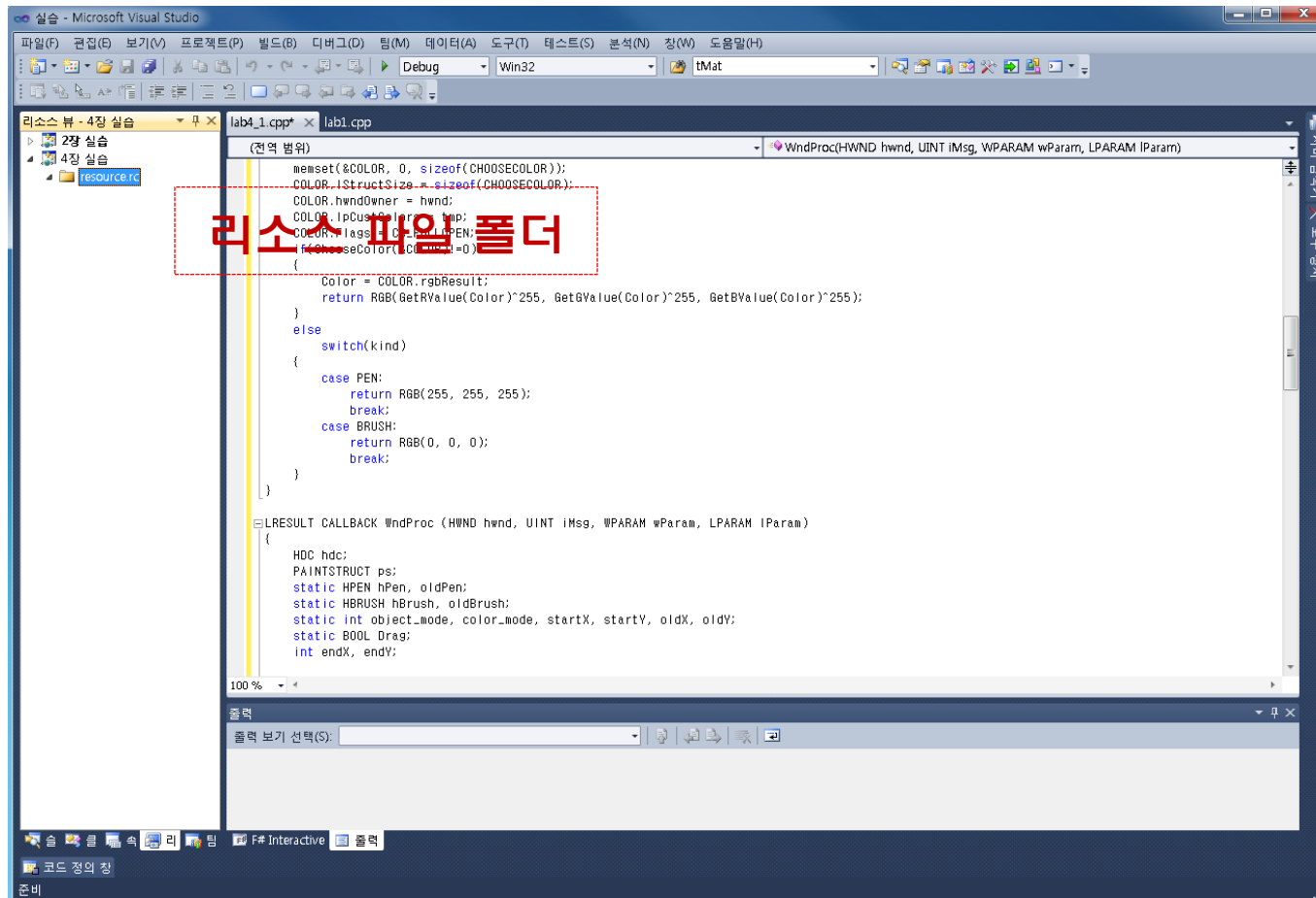
- ▶ 리소스 파일 작성
  - ▶ 방법: 소스 파일 작성과 유사
  - ▶ C++ 소스 대신에 **Resource Script** 선택
  - ▶ 리소스 파일 이름 명시
  - ▶ 리소스-파일.rc(리소스 정의), resource.h(리소스 ID 정의) 생성
- ▶ 메뉴화면 편집
  - ▶ 리소스 도구상자 이용
  - ▶ 속성 정의 : 프로그램과 연계
  - ▶ 메뉴 ID 정의 : 메뉴바에서
- ▶ 프로그램에서 메뉴 사용
  - ▶ include “**resource.h**”
  - ▶ 메뉴ID 등록 : 윈도우 클래스
  - ▶ 메뉴 처리 : **WM\_COMMAND** 메시지

# 메뉴 만들기

- ▶ Visual Studio 2010 환경
  - ▶ [프로젝트] -> [리소스 파일] -> [새 항목 추가]

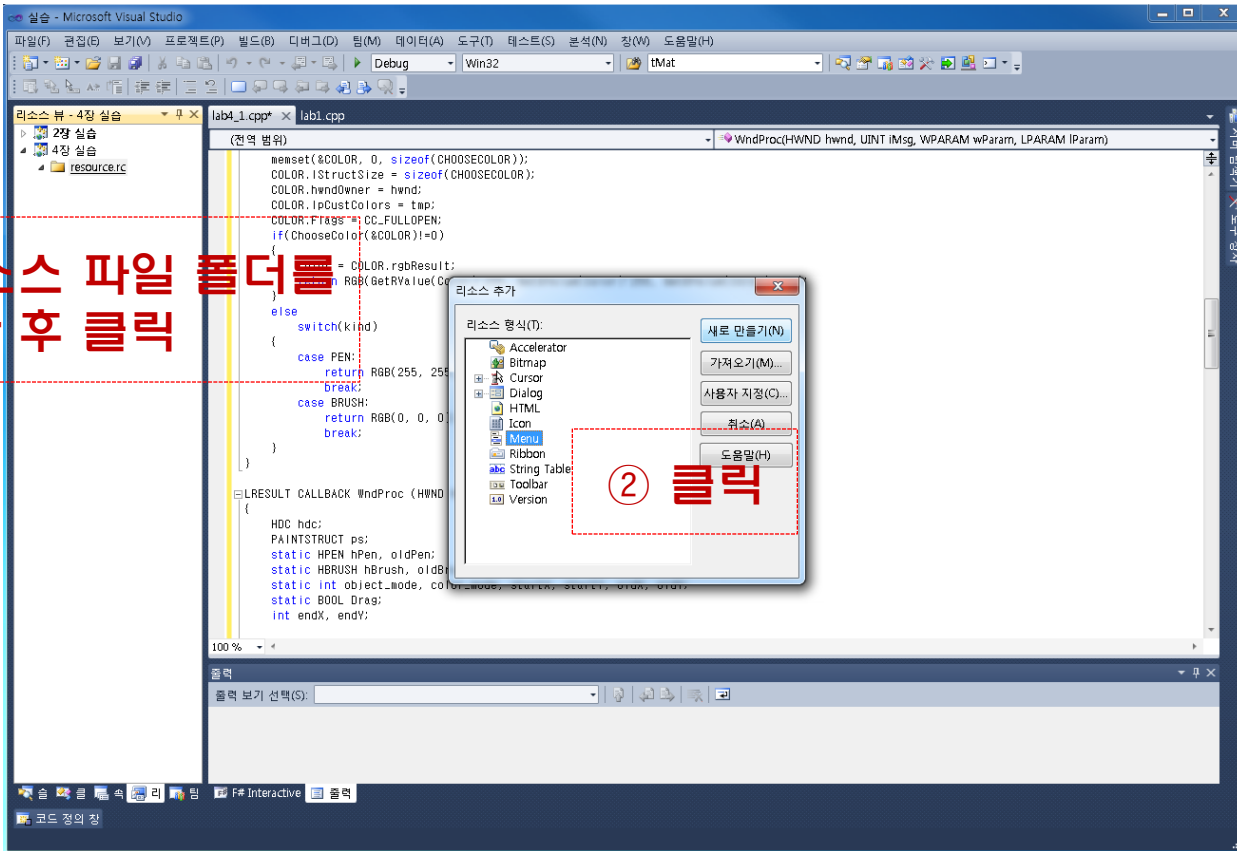


# 생성된 리소스 파일



# 메뉴 만들기

① 리소스 파일 폴더를  
선택한 후 클릭



② 클릭



# 메뉴 항목 편집

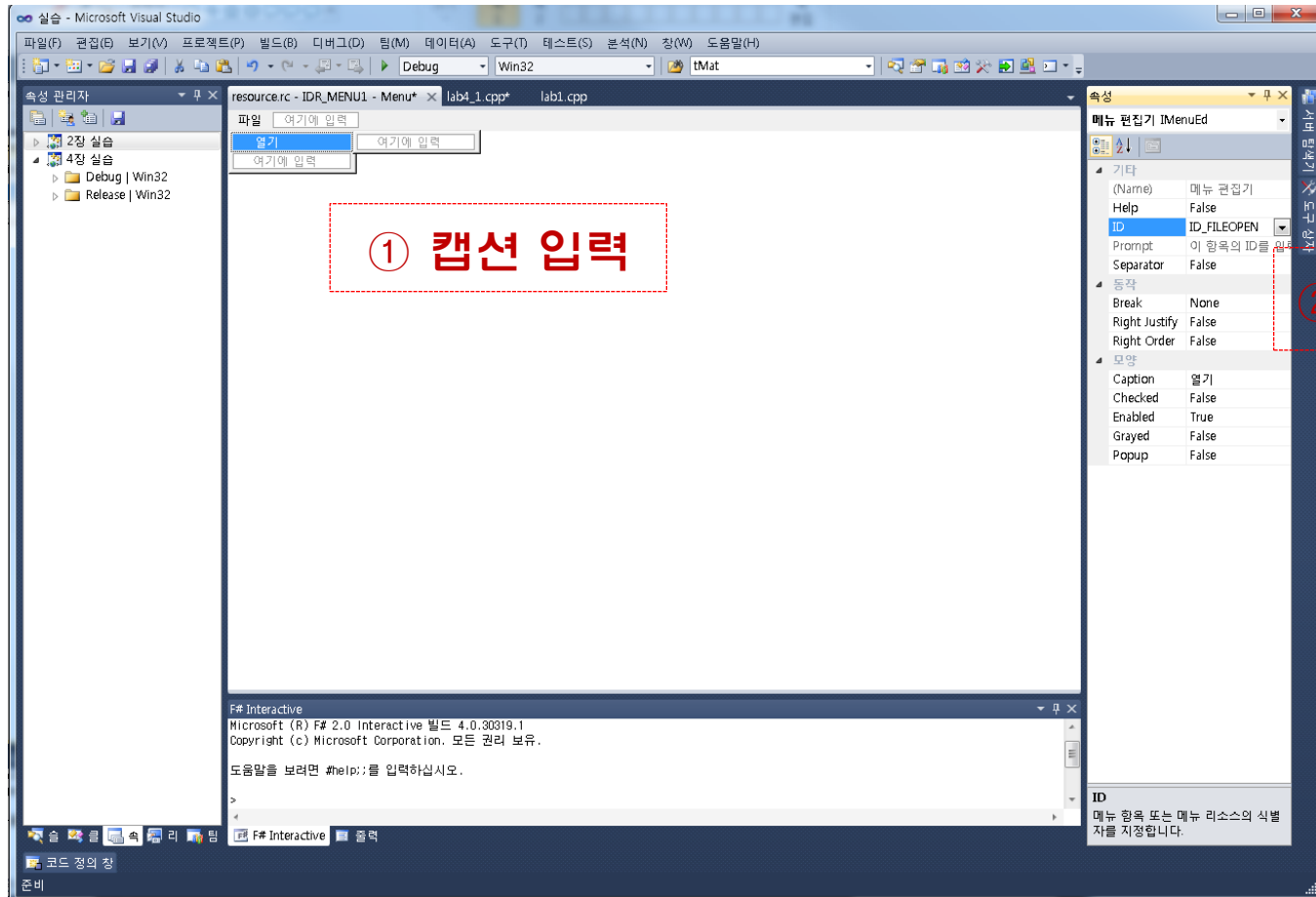
① 메뉴 생성 확인

② 캡션 입력

③ 속성 설정

The screenshot shows the Microsoft Visual Studio environment. The Solution Explorer on the left displays a project structure with folders for '2장 실습', '4장 실습', 'Debug | Win32', and 'Release | Win32'. The main editor window shows the 'resource.rc' file with a menu definition. The Properties window on the right is titled '메뉴 편집기 IMenuEd' and displays various properties for the selected menu item, including '기타' (General), '동작' (Behavior), and '모양' (Appearance) sections. The 'Caption' property is highlighted, corresponding to the '캡션 입력' step. The '속성 설정' step points to the Properties window. The '메뉴 생성 확인' step points to the menu definition in the resource file.

# 메뉴 항목 추가

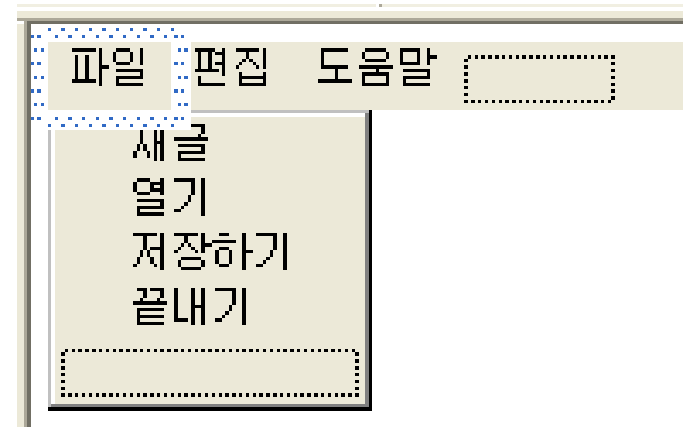


## 4-1 기본 메뉴 만들기

### ▶ 예제 내용

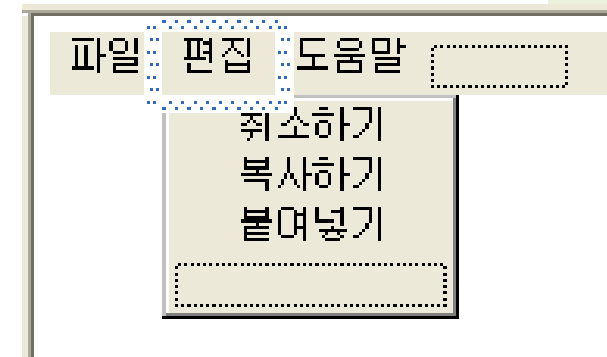
- ▶ 다음의 메뉴항목 설정표를 참고하여 기본 메뉴를 작성하기.  
-> 다음 페이지 설명 계속

Caption	ID	속성
파일		Pop-up
새글	ID_FILENEW	디폴트
열기	ID_FILEOPEN	디폴트
저장하기	ID_FILESAVE	디폴트
끝내기	ID_EXIT	디폴트

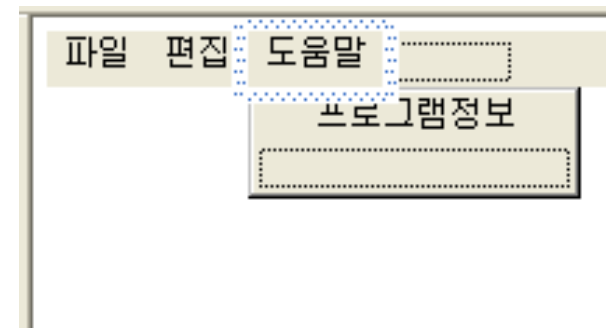


## 실습 4-1 (계속)

Caption	ID	속성
편집		Pop-up
취소하기	ID_EDITUNDO	디폴트
복사하기	ID_EDITCOPY	디폴트
붙여넣기	ID_EDITPASTE	디폴트

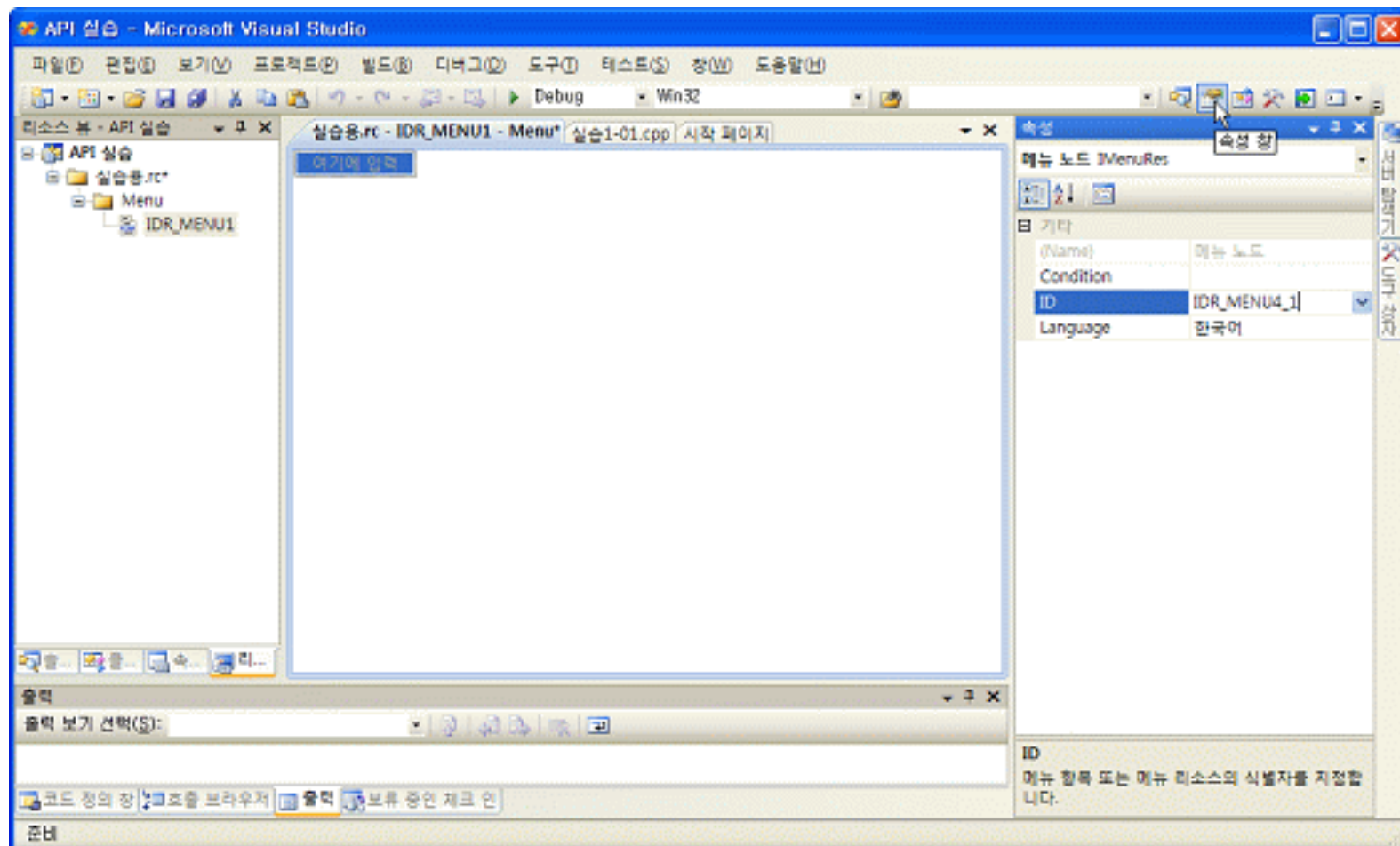


Caption	ID	속성
도움말		Pop-up
프로그램정보	ID_INFORM	디폴트



## 실습 4-1 (계속)

- 메뉴 ID 부여



## 2절. 메뉴 사용하기

- ▶ 메뉴를 선택했을 때 WM\_COMMAND 메시지가 전달
  - ▶ WM\_COMMAND
    - ▶ 메뉴의 메뉴항목을 선택하면 발생하는 메시지
    - ▶ 커맨드 메시지라 부름
  - ▶ LOWORD(wParam)
    - ▶ 선택된 메뉴항목의 ID가 정수로 들어 있음
  - ▶ HIWORD(wParam)
    - ▶ 0 (이벤트 소스)
  - ▶ lParam
    - ▶ 0

## 4-2 윈도우에 메뉴 붙이기

### ▶ 응용 프로그램에 자원 파일 불러오는 방법 3가지

#### ▶ 윈도우 클래스를 만들 때 메뉴를 정의

```
winclass.lpszMenuName = MAKEINTRESOURCE ( MYMENU );
```

#### ▶ 메인 윈도우를 생성할 때 메뉴를 첨부

```
winclass.lpszMenuName = NULL;  
CreateWindow (...,  
              LoadMenu ( hinstance, MAKEINTRESOURCE  
                          (MYMENU ) ),  
              ...);
```

#### ▶ 초기 윈도우 생성이 끝난 후에 붙인다.

```
HMENU hmenu = LoadMenu (hinstance,  
                          MAKEINTRESOURCE ( MYMENU ) );  
SetMenu (hwnd, hmenu);
```

## 4-2 윈도우에 메뉴 붙이기

```
#include "resource.h"  
... 중략 ...
```

// 리소스 파일 첨부

```
wndclass.lpszMenuName = // 메뉴 ID 등록 : 클래스 파일  
MAKEINTRESOURCE (IDR_MENU4_2);
```

- ▶ **MAKEINTRESOURCE**: 리소스에 대한 정수형 상수를 문자열로 변환하는 매크로 함수
  - ▶ LPTSTR MAKEINTRESOURCE(  
WORD wInteger // 리소스에 대한 정수형 상수  
);



## 4-3 커맨드 메시지 처리하기

```
int answer;
```

```
switch (iMsg)
```

```
{
```

```
case WM_COMMAND:
```

```
    switch (LOWORD(wParam)) {
```

```
        case ID_FILENEW:
```

```
            MessageBox (hwnd, "새파일을 열겠습니까 ?",  
                        "새파일 선택", MB_OKCANCEL );
```

```
            break;
```

```
        case ID_EXIT:
```

```
            answer = MessageBox (hwnd, "파일을 저장하고 끝내겠습니까 ? ",  
                                "끝내기 선택", MB_YESNOCANCEL );
```

```
            if (answer == IDYES || answer == IDNO)  
                PostQuitMessage (0);
```

```
            break;
```

```
    }
```

```
    break;
```

# 메시지박스

– 메시지박스: 사용자에게 경고나 알림 메시지를 주는 대화상자

```
int MessageBox (HWND hwnd, LPCTSTR lpText,  
                LPCTSTR lpCaption, UINT uType);
```

- **lpText**
  - 메시지 박스에 표시될 글
- **lpCaption**
  - 메시지 박스의 타이틀바에 표시될 글
- **uType**과 함수 반환값

<b>uType</b>	<b>반환 값</b>
<b>MB_OK</b>	<b>IDOK</b>
<b>MB_OKCANCEL</b>	<b>IDOK, IDCANCEL</b>
<b>MB_YESNO</b>	<b>IDYES, IDNO</b>
<b>MB_YESNOCANCEL</b>	<b>IDYES, IDNO, IDCANCEL</b>

# 리소스: 아이콘

- 아이콘은 프로그램의 메인 윈도우가 최소화(아이콘화)되었을 때나 배경화면에 등록 될 때 응용 프로그램을 나타내는 작은 그래픽 이미지이다.
- 아이콘과 커서는 크게 두 가지 종류의 리소스가 있다.
  - 내장(built-in) 리소스와 사용자 정의 리소스이다.
  - 내장 리소스는 윈도우즈에서 기본적으로 제공하는 것이다.
  - LoadIcon()함수의 첫 번째 인자에 "NULL"을 주고 두 번째 인자로 아이콘 이름 문자열을 지정하며 리턴된 핸들을 wc.hIcon에 대입한다.
- 손(IDI\_HAND), 느낌표(IDI\_WARNING) 등 9가지가 있으며 지금까지 사용한 표준 내장 아이콘(IDI\_APPLICATION)은 다음과 같은 윈도우 모양을 갖는다.

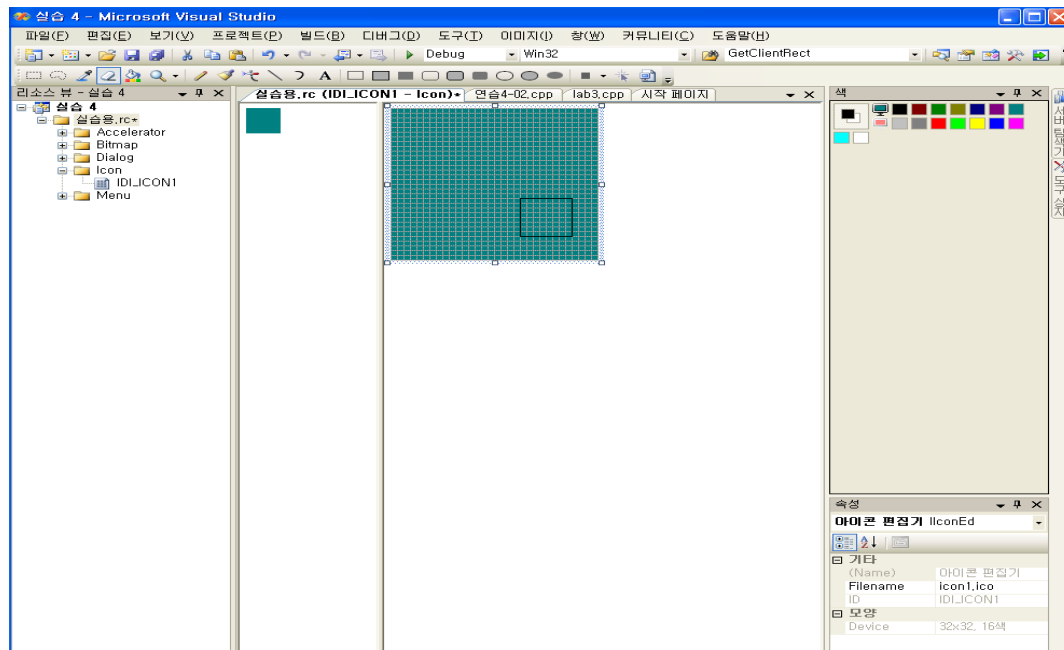


```
wc.hIcon = LoadIcon(NULL,IDI_APPLICATION);  
wc.hIconSm=LoadIcon(NULL,IDI_APPLICATION);
```

Value	Description
IDI_APPLICATION	Default application icon.
IDI_ASTERISK	Same as IDI_INFORMATION.
IDI_ERROR	Hand-shaped icon.
IDI_EXCLAMATION	Same as IDI_WARNING.
IDI_HAND	Same as IDI_ERROR.
IDI_INFORMATION	Asterisk icon.
IDI_QUESTION	Question mark icon.
IDI_WARNING	Exclamation point icon.
IDI_WINLOGO	Windows logo icon.

# 리소스: 아이콘 편집

- 32×32픽셀에 칼라로 원하는 아이콘을 그릴 수 있다.
- 화면 오른쪽에 그림판 프로그램에서 볼 수 있는 그래픽 편집 도구들이 보인다.
- 혹시 이 도구들이 보이지 않는 경우에는 캡션바를 제외한 비 클라이언트 영역에서 마우스 오른쪽 버튼을 클릭하여 나오는 팝업 메뉴에서 "색상창표시"를 체크하면 된다.



# 리소스: 커서

- 커서는 마우스의 위치를 나타내는 작은 그래픽 이미지이다.
- 내장 커서는 아래의 표와 같은 모양이 제공된다.
- LoadCursor()함수의 첫 번째 인자에 "NULL"을 주고 두 번째 인자로 커서 이름 문자열을 지정하며 리턴된 핸들을 wc.hCursor에 대입한다.
- 화살표, 모래시계 등 11가지가 있으며 지금까지 사용한 표준 내장 커서는 화살표 모양을 갖는다.
- 아이콘과 마찬가지로 새로운 리소스에서 커서를 선택하여 비트맵을 그린다.

```
wc.hCursor= LoadCursor(NULL,IDC_ARROW);
```

값	커서	모양
IDC_APPSTARTING	프로그램이 시작될 때 사용된다.	
IDC_ARROW	표준 화살표 커서	
IDC_CROSS	십자 모양의 커서. 정확한 선택을 해야 할 때 사용된다.	
IDC_IBEAM	I자 모양의 커서. 주로 문자열 입력 영역에 사용된다.	
IDC_ICON	Win32에서는 사용되지 않음	
IDC_NO	원 안의 빗금이 쳐진 커서이며 드래그 금지 구역을 나타낸다.	
IDC_SIZE	Win32에서는 사용되지 않음	
IDC_SIZEALL	4방향 화살표	
IDC_SIZENESW	좌하우상 크기조절 커서	
IDC_SIZENS	수직 크기조절 커서	
IDC_SIZENWSE	좌상우하 크기조절 커서	
IDC_SIZEWE	수평 크기조절 커서	
IDC_UPARROW	수직 화살표	
IDC_WAIT	모래 시계 커서. 시간이 오래 걸리는 작업을 할 때 사용된다.	

# 리소스: 커서 편집

- 사용자 정의 아이콘과 리소스를 만들었으면 소스에 새로 만든 리소스를 연결해주어야 한다.
  - `wc.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI_ICON1));`
  - `wc.hCursor = LoadCursor(hInstance, MAKEINTRESOURCE(IDC_CURSOR1));`
  - `wc.hIconSm=LoadIcon(hInstance, MAKEINTRESOURCE(IDI_ICON1));`
- 두 함수의 첫 번째 인자에 "NULL"대신 WinMain()함수의 첫 번째 인자인 "hInstance"를 쓰며 두 번째 인자에 만든 리소스 ID를 쓴다.
- 두 번째 인자로 리소스명을 바로 쓰면 안되고, MAKEINTRESOURCE매크로로 변환해야 한다.
  - 이 매크로는 정수형값을 LPCTSTR형으로 변환해준다.
  - ID는 정수 값이며 두 함수 모두 두 번째 인자는 LPCTSTR형을 받게 되어 있으므로 이 매크로를 사용해서 변환해야 한다.
- 해당 리소스 헤더파일을 추가한다.

# 실습 4-1

## ▶ 제목

- ▶ 공튀기기 프로그램 (실습 3-10) 업그레이드

## ▶ 내용

- ▶ 메뉴를 만든다.
- ▶ Game: Start/End
  - ▶ 공이 이동하기 시작/끝낸다
- ▶ Speed: Slow/Medium/Fast
  - ▶ 원이 천천히/중간속도/빠르게 이동하면서 벽에 맞으면 튕어서 다른 방향으로 이동한다.
- ▶ Ball: Small/Big
  - ▶ 공이 작거나/큰 크기로 그려진다.
- ▶ Color: Red / Green / Blue
  - ▶ 공의 색상을 변경한다.
- ▶ 그리드: 켜기 / 끄기
  - ▶ 화면에 모눈 종이 형태의 그리드를 그리거나 지우기

## 실습 4-2

### ▶ 제목

- ▶ 메뉴를 사용하여 그리기

### ▶ 내용

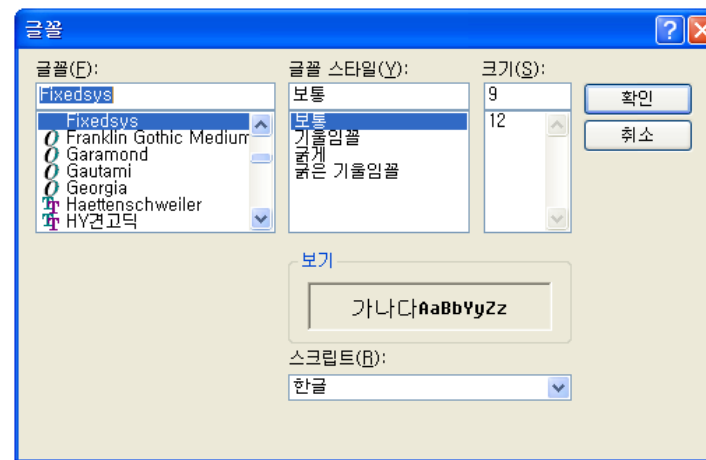
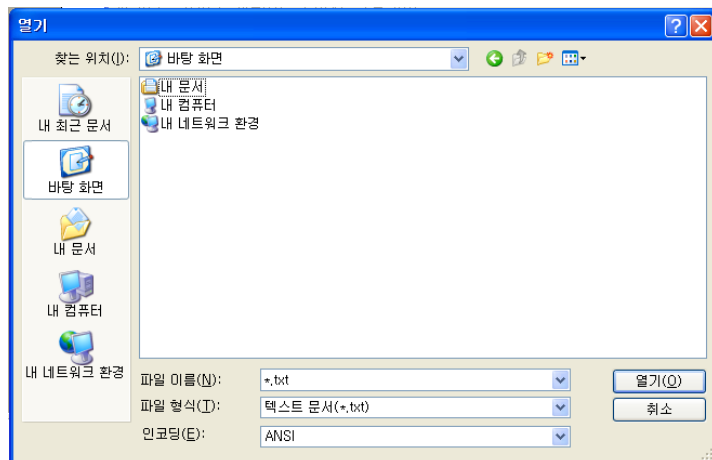
- ▶ 메뉴를 만든다.
- ▶ Color: Cyan/Magenta/Yellow/Random
  - ▶ 그릴 도형의 색상을 설정한다.
- ▶ Shape: Line/Rectangle/Ellipse
  - ▶ 고무줄 효과를 가진 선, 사각형, 원을 그린다.
- ▶ Border: On/Off
  - ▶ 도형의 외곽선을 검정색으로 그린다/외곽선 없이 도형을 그린다
- ▶ Help: Help/Exit
  - ▶ Help: 프로그램 이름을 보여주는 메시지 박스
  - ▶ Exit: 프로그램 종료 확인하는 메시지 박스 띄운 후 프로그램 종료
- ▶ 위의 메뉴가 실행되도록 구현한다.



## 실습 4-3

# 3절. 공용대화상자 이용하기

- 윈도우 프로그램에서 공통으로 사용되는 대화상자
  - 파일 열기
  - 파일 저장하기
  - 글꼴 선택하기



# 공용대화상자 – 파일 열기

- 파일열기 처리절차
  - OPENFILENAME 구조체 할당
  - 열기함수 호출 -> 파일이름 획득

```
OPENFILENAME OFN; // 구조체 할당
memset(&OFN, 0, sizeof(OPENFILENAME)); // (1) 구조체 초기화

OFN.lStructSize = sizeof(OPENFILENAME); // (2) 구조체 크기
OFN.hwndOwner = hwnd; // (3) 윈도우 핸들
OFN.lpstrFile = filepath; // (4) 선택한 파일경로 저장
OFN.nMaxFileTitle = 100; // (5) 파일경로 최대길이
OFN.lpstrFileTitle = filename; // (6) 선택한 파일이름 저장
OFN.nMaxFile = 100; // (7) 파일이름 최대길이

GetOpenFileName(&OFN); // (8) 열기할 파일이름을 획득
```

# 공용대화상자 – 파일 열기

- **memset 함수: 메모리를 지정한 문자로 채우는 함수**
  - `void *memset (void *s, int c, size_t n);`
    - s: 채울 메모리의 주소
    - c: 채울 문자
    - n: 채우려고 하는 문자 개수
  - 예) 20개 double 자료형을 요소로 갖는 배열 만들기 (초기값으로 0 채우기)
    - `double arr[20];`
    - `memset (arr, 0, 20*sizeof(double));`

# 공용대화상자 – 파일 열기

## OPENFILENAME 구조체

```
typedef struct tagOFN{
    DWORD      IStructSize;      // ofn
                                //구조체 크기
    HWND        hwndOwner;      //오너 윈도우 핸들
    HINSTANCE   hInstance;      //인스턴스 핸들
    LPCTSTR     lpstrFilter;      //파일 형식 콤보 박스에 나타낼 필터
    LPTSTR      lpstrCustomFilter; //커스텀 필터를 저장하기 위한 버퍼
    DWORD       nMaxCustFilter;  //커스텀 필터 버퍼의 길이
    DWORD       nFilterIndex;    //파일 형식 콤보 박스에서 사용할 필터의 인덱스
    LPTSTR      lpstrFile;       //파일 이름 에디트에 처음 나타낼 파일명
    DWORD       nMaxFile;       //lpstrFile 멤버의 길이
    LPTSTR      lpstrFileName;   //선택한 파일명을 리턴받기 위한 버퍼
    DWORD       nMaxFileName;   //lpstrFileName 멤버의 길이
    LPCTSTR     lpstrInitialDir; //파일 찾기를 시작할 디렉토리
    LPCTSTR     lpstrTitle;      //대화상자의 캡션
    DWORD       Flags;          //대화상자의 모양과 동작을 지정하는 플래그
    WORD        nFileOffset;     //lpstrFile 버퍼 내의 파일명 오프셋
    WORD        nFileExtension; //lpstrFile 버퍼 내의 파일 확장자 오프셋
    LPCTSTR     lpstrDefExt;     //디폴트 확장자
    DWORD       lCustData;       //쪽 프로시저로 보낼 사용자 정의 데이터
    LPOFNHOOKPROC lpfnHook;      //쪽 프로시저명
    LPCTSTR     lpTemplateName; //템플릿명
}OPENFILENAME;
```

# 공용대화상자 – 파일 열기

- 열 파일을 선택할 수 있도록 드라이브, 폴더, 파일 이름을 보여주는 열기 대화상자를 보여준다.

열기 / 대화상자를 초기화하며 사용자가 선택한 파일의 정보가 저장된다.

BOOL GetOpenFileName (LPOPENFILENAME lpofn);

- 저장할 파일 이름을 쓰거나 선택할 수 있도록 드라이브, 폴더, 파일 이름을 보여주는 대화상자를 보여준다.

BOOL GetSaveFileName (LPOPENFILENAME lpofn);

- 파일 입출력 관련 Win32 API함수와 유사 기능의 C언어에서 제공하는 표준 라이브러리 함수

기능	C언어 표준 라이브러리 함수	Win32 API함수
파일 열기	fopen()	CreateFile()
파일 닫기	fclose()	CloseHandle()
파일 포인터 위치 변경/획득	fseek()	SetFilePointer()
파일 읽기	fread()	ReadFile()
파일 쓰기	fwrite()	WriteFile()

# 필터 지정방법

- 필터의 용도

- 표시되는 파일이름을 걸러 줌
- 정의시 **공문자** 삽입 안하도록
- 매 필터마다 널 문자로 종료하며 하나의 필터는 “파일형식\0필터”로 표시한다.
- 여러 개의 패턴 지정하려면 ;로 연결

```
char filter[] = "텍스트 파일(*.txt)\0*.txt\0모든 파일(*.*)\0*.*\0";  
OFN.lpstrFilter = filter; // 필터 등록
```

표시되는  
문자열

파일 이름(N):

파일 형식(I):

The screenshot shows a file dialog box with a dropdown menu for file types. The dropdown is open, showing three options: "텍스트 파일 (\*.txt)", "모든 파일 (\*.\*)", and "모든 파일 (\*.\*)". The first option, "텍스트 파일 (\*.txt)", is currently selected and highlighted in blue. The text "파일 이름(N):" and "파일 형식(I):" are visible to the left of the dropdown.

## 4-4 열기 대화상자 이용하기

```
OPENFILENAME OFN;  
char str[100], lpstrFile[100] = "";  
char filter[] = "Every File(*.*) \0*.*\0Text File\0*.txt;*.doc\0";  
switch (iMsg)  
{  
case WM_COMMAND:  
    switch(LOWORD(wParam)) {  
        case ID_FILEOPEN:  
            memset(&OFN, 0, sizeof(OPENFILENAME));  
            OFN.lStructSize = sizeof(OPENFILENAME);  
            OFN.hwndOwner=hwnd;  
            OFN.lpstrFilter= filter;  
            OFN.lpstrFile=lpstrFile;  
            OFN.nMaxFile=100;  
            OFN.lpstrInitialDir="."; // 초기 디렉토리  
            if (GetOpenFileName(&OFN)!=0) {  
                wsprintf(str,"%s 파일을 여시겠습니까 ?",OFN.lpstrFile);  
                MessageBox(hwnd,str,"열기 선택",MB_OK);  
            }  
            break;
```



## 4-5 저장하기 대화상자 이용하기

OPENFILENAME SFN; // 파일열기와 저장하기는 동일한 구조체 사용

```
switch (iMsg) {
case WM_COMMAND:
    switch (LOWORD(wParam)) {
    case ID_FILESAVE:
        memset(&SFN, 0, sizeof(OPENFILENAME));
        SFN.lStructSize = sizeof(OPENFILENAME);
        ...

        SFN.lpstrInitialDir=".";
        if (GetSaveFileName(&SFN)!=0) {
            wsprintf(str,"%s 파일에 저장하시겠습니까 ?",
                    SFN.lpstrFile);
            MessageBox(hwnd,str,"저장하기 선택",MB_OK);
        }
        break;
```

# 공용대화상자 – 폰트 선택하기

- 폰트 선택하기 처리절차
  - CHOOSEFONT 구조체 할당
  - LOGFONT 구조체 변수 연결
  - 폰트대화상자 띄우기 -> 폰트정보 획득
  - 폰트 만들어 사용하기

```
CHOOSEFONT FONT;  
LOGFONT  LogFont;
```

```
FONT.lStructSize = sizeof(CHOOSEFONT); // 구조체 크기  
FONT.hwndOwner = hwnd; // 윈도우 핸들  
FONT.lpLogFont = &LogFont; // LOGFONT 구조체 변수 연결  
FONT.Flags = CF_EFFECTS | CF_SCREENFONTS; // 폰트대화상자 옵션  
ChooseFont(&FONT) // 폰트대화상자 띄우기  
hFont = CreateFontIndirect(&LogFont); // 선택된 폰트 핸들 생성  
OldFont = (HFONT)SelectObject(hdc, hFont); // 폰트 사용
```

# 공용대화상자 – 폰트 선택하기

## CHOOSEFONT 구조체

```
typedef struct {  
    DWORD IStructSize;      // 구조체 크기  
    HWND hwndOwner;  
    HDC hDC;                // 메인 DC 핸들  
    LPLOGFONT lpLogFont;    // LOGFONT 구조체 변수 값 (글꼴 선택하면  
                           // 설정된다.)  
    INT iPointSize;         // 선택한 글꼴의 크기 (글꼴 선택하면 설정된다)  
    DWORD Flags;            // 글꼴 상자 초기화  
    COLORREF rgbColors;     // 선택한 글꼴의 색상 정보 저장  
    LPARAM ICustData;  
    LPCFHOOKPROC lpfnHook;  
    LPCTSTR lpTemplateName;  
    HINSTANCE hInstance;  
    LPSTR lpszStyle;  
    WORLD nFontType;       // 선택한 글꼴을 가리키는 필드  
    INT nSizeMin;  
    INT nSizeMax;  
} CHOOSEFONT, *LPCHOOSEFONT;
```

# 공용대화상자 – 폰트 선택하기

## LOGFONT 구조체

```
Typedef struct {  
    LONG lHeight;           // 논리적 크기의 글꼴의 높이를 나타내는 정수  
    LONG lWidth;            // 글꼴의 너비  
    LONG lEscapement;       // 글꼴의 기울기 지정 (0 ~ 100 사이의 정수)  
    LONG lOrientation;      // 이탤릭 체 (TRUE/FALSE)  
    LONG lWeight;           // 글자에 밑줄 (TRUE/FALSE)  
    BYTE bItalic;           // 글자에 취소선 (TRUE/FALSE)  
    BYTE bUnderline;       // 글자에 밑줄 (TRUE/FALSE)  
    BYTE bStrikeOut;        // 글자에 취소선 (TRUE/FALSE)  
    BYTE bCharSet;          // 문자 배열로 글꼴 이름 저장  
    BYTE bOutPrecision;     // 문자 배열로 글꼴 이름 저장  
    BYTE bQuality;          // 문자 배열로 글꼴 이름 저장  
    BYTE bPitchAndFamily;   // 문자 배열로 글꼴 이름 저장  
    TCHAR lFaceName[LF_FACESIZE];  
} LOGFONT;
```

# CHOOSEFONT 구조체 필드

- **lStructSize**: CHOOSEFONT 구조체의 크기 값을 넣어준다. 일반적으로 sizeof(CHOOSEFONT)를 넣어준다.
- **hwndOwner**: 대화상자의 주인 윈도우를 저장한다. 따라서 메인 윈도우 핸들인 hwnd를 넣어준다.
- **hpLogFont**: LOGFONT 구조체 변수의 주소값을 저장하는 곳으로 폰트 대화상자를 통해 선택된 폰트 정보를 얻어오는 공간이다.
- **Flags**: 폰트 대화상자를 초기화 하는데 사용되는 비트 플래그들의 조합을 저장하는 공간이다.
  - **CF\_EFFECTS**: 폰트 대화상자에 strikethrough, underline, 텍스트 컬러 등을 선택할 수 있는 컨트롤을 배치하게 한다. LOGFONT 변수에 설정된 정보는 폰트 대화상자에 나타나고 사용자가 선택하면 다시 LOGFONT 변수에 저장되어 돌아온다.
  - **CF\_SCREENFONTS**: 윈도우 시스템에서 제공하고 있는 폰트들을 폰트 대화상자에 나타나게 한다.

# CHOOSEFONT 구조체 필드

- **iPointSize**: 선택된 폰트의 크기 값을 저장하는 공간으로 포인트값의 10분의 1단위로 쓸 수 있다.
- **rgbColors**: Flag에 CF\_EFFECTS가 설정되어 있을 때 의미 있는 필드로서 선택된 폰트의 색상정보를 저장한다. 색상정보는 COLORREF타입으로 저장된다.
- **nFontType**: 선택된 폰트의 타입을 가리키는 필드이다.
  - **BOLD\_FONTTYPE**: 굵은 글씨체를 선택했을 때를 가리킴
  - **ITALIC\_FONTTYPE**: 이탤릭 글씨체를 선택했을 때를 가리킴
  - **REGULAR\_FONTTYPE**: 일반 글씨체를 선택했을 때를 가리킴

# 텍스트의 색상을 변경

```
COLORREF SetTextColor(
```

```
    HDC hdc,          // 변경할 디바이스 컨텍스트
```

```
    COLORREF crColor // 변경할 색상
```

```
);
```

```
// 디바이스 컨텍스트에 이미 등록되어 있던 텍스트 색상값  
// 을 반환
```

## 4-6 폰트 대화상자 이용하기

```
CHOOSEFONT FONT;  
static COLORREF  fColor;  
HFONT          hFont, OldFont;  
static LOGFONT   LogFont;  
  
case WM_COMMAND:  
    switch(LOWORD(wParam))  
    {  
        case ID_FONTDLG:  
            memset(&FONT, 0, sizeof(CHOOSEFONT));  
            FONT.lStructSize = sizeof(CHOOSEFONT);  
            FONT.hwndOwner = hwnd;  
            FONT.lpLogFont = &LogFont;  
            FONT.Flags = CF_EFFECTS | CF_SCREENFONTS;  
            if(ChooseFont(&FONT)!=0) {  
                fColor = FONT.rgbColors;  
                InvalidateRgn(hwnd, NULL, TRUE);  
            }  
            break;
```



## 4-6 폰트 대화상자 이용하기(계속)

```
case WM_PAINT:
```

```
    hdc = BeginPaint(hwnd, &ps);
```

```
    hFont = CreateFontIndirect(&LogFont);
```

```
    OldFont = (HFONT)SelectObject(hdc, hFont);
```

```
    SetTextColor(hdc, fColor);
```

```
    TextOut(hdc, 10, 10, "HelloWorld", 10);
```

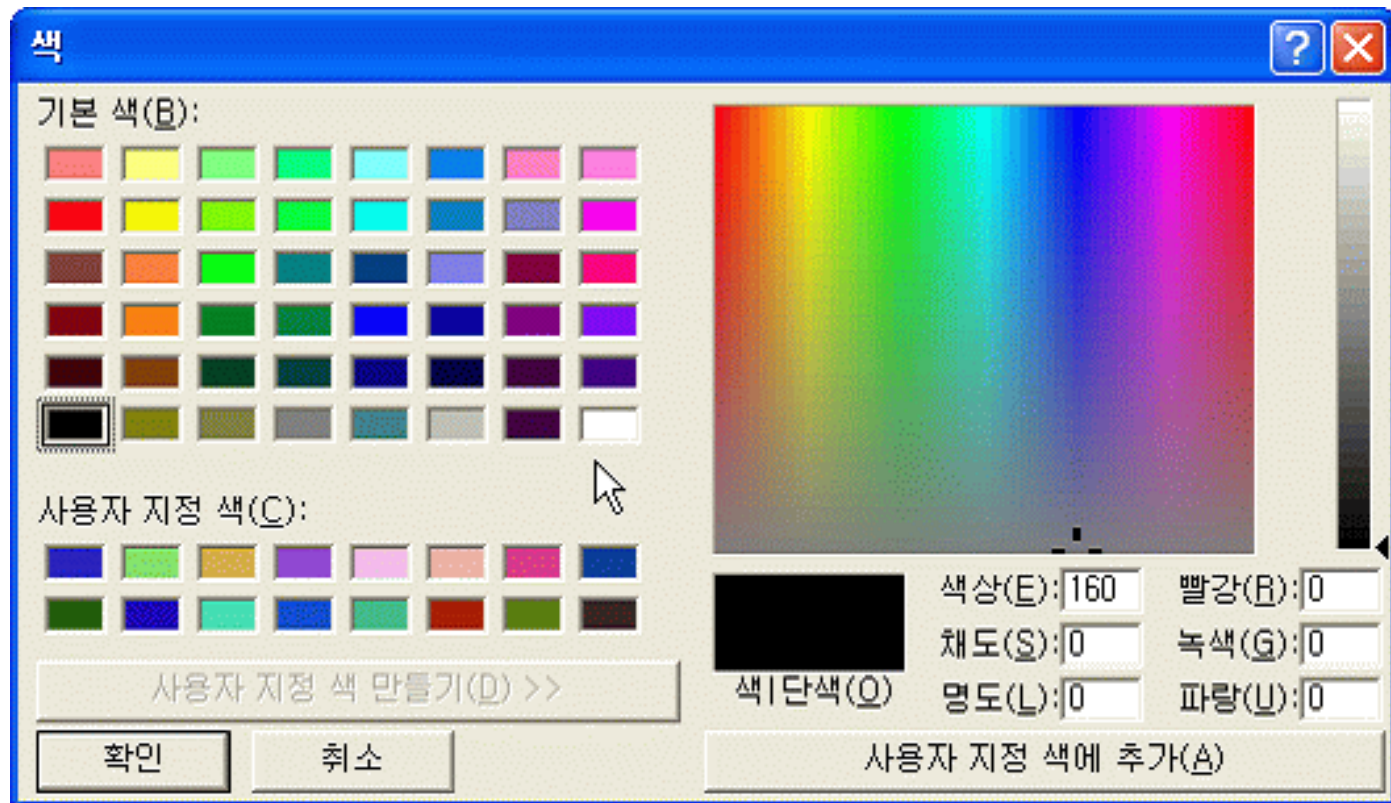
```
    SelectObject(hdc, OldFont);
```

```
    DeleteObject(hFont);
```

```
    EndPaint(hwnd, &ps);
```

```
    break;
```

# 색상 선택하기



# 공용대화상자 – 색상 선택하기

- 색상선택하기 처리절차
  - CHOOSECOLOR 구조체 할당
  - “사용자 지정 색” 만들기
  - 색상 대화상자 띄우기-> 색상 정보 획득

```
CHOOSECOLOR COLOR;  
static COLORREF tmp[16], color;  
for(i=0;i<16;i++)  
    tmp[i] = 사용자 지정 색상;  
memset(&COLOR, 0, sizeof(CHOOSECOLOR));  
COLOR.lStructSize = sizeof(CHOOSECOLOR);  
COLOR.hwndOwner = hwnd;  
COLOR.lpCustColors = tmp;  
COLOR.Flags = CC_FULLOPEN;  
ChooseColor(&COLOR); // COLOR.rgbResult에 색상정보 저장됨
```

# 공용대화상자 – 색상 선택하기

## CHOOSECOLOR 구조체

```
typedef CHOOSECOLOR {  
    DWORD IStructSize; // 구조체 크기  
    HWND hwndOwner; // 메인 윈도우 핸들  
    HWND hInstance;  
    COLORREF rgbResult; // 사용자가 대화상자에서 선택한 색상 정보  
    COLORREF *lpcustColors; // 색 대화상자에 사용자 지정색에  
                           //채울 색 정보 목록 (16가지)  
    DWORD Flags; // 색 대화상자 초기화 하는데 사용한 플래그  
    LPARAM lcustData;  
    LPCCHOOKPROC lpfnHook;  
    LPCTSTR lpTemplateName;  
} CHOOSECOLOR, *LPCHOOSECOLOR;
```

## 4-7 색상 대화상자 이용하기

```
CHOOSECOLOR COLOR;  
static COLORREF tmp[16], color;  
HBRUSH hBrush, OldBrush;  
int i;
```

```
case WM_PAINT:  
    hdc = BeginPaint(hwnd, &ps);  
    hBrush = CreateSolidBrush(color);  
    OldBrush = (HBRUSH)SelectObject(hdc, hBrush);  
    Ellipse(hdc, 10, 10, 200, 200);  
    SelectObject(hdc, OldBrush);  
    DeleteObject(hBrush);  
    EndPaint(hwnd, &ps);  
    break;
```

## 4-7 색상 대화상자 이용하기(계속)

```
case WM_COMMAND:
    switch(LOWORD(wParam))
    {
    case ID_COLORDLG:
        for(i=0;i<16;i++)
            tmp[i] = RGB(rand()%256,rand()%256,rand()%256);
        memset(&COLOR, 0, sizeof(CHOOSECOLOR));
        COLOR.lStructSize = sizeof(CHOOSECOLOR);
        COLOR.hwndOwner = hwnd;
        COLOR.lpCustColors = tmp;
        COLOR.Flags = CC_FULLOPEN;
        if(ChooseColor(&COLOR)!=0) {
            color = COLOR.rgbResult;
            InvalidateRgn(hwnd, NULL, TRUE);
        }
        break;
    }
    break;
```

## 4절. 메뉴 수정하기

### ▶ 수정 배경

- ▶ 새로운 메뉴의 추가
- ▶ 메뉴항목의 속성변경
  - ▶ 선택 가능 / 불가능 상태로 설정, 복사하기 이후 붙여넣기가 가능

### ▶ 수정 방법

#### ▶ 메뉴 불러오기

**메뉴** : HMENU GetMenu(HWND hWnd);

**부메뉴** : HMENU GetSubMenu(HMENU **hMenu**, int nPos);

반환값: 메뉴 핸들

hMenu: 부메뉴를 비롯해 메뉴의 핸들값

nPos: 메뉴에서 몇 번째 부메뉴를 지칭하는지를 알리는 정수  
(첫 번째 하위 메뉴가 0번)

#### ▶ 메뉴항목 활성화/비활성화 하기

BOOL EnableMenuItem(HMENU hMenu,  
                                UINT **uIDEnableItem**, UINT **uEnable**);

**uIDEnableItem** : 활성화/비활성화하고자 하는 메뉴 ID

**uEnable** = **MF\_GRAYED** : 비활성화  
          = **MF\_ENABLED** : 활성화

## 4-8 복사하기, 붙여넣기 비활성화

```
static HMENU hMenu, hSubMenu;
```

```
switch (iMsg) {
```

```
case WM_CREATE :
```

```
    hMenu = GetMenu(hwnd);
```

```
    hSubMenu = GetSubMenu(hMenu, 1);
```

```
    EnableMenuItem(hSubMenu, ID_EDITCOPY, MF_GRAYED);
```

```
    EnableMenuItem(hSubMenu, ID_EDITPASTE, MF_GRAYED);
```

```
    return 0 ;
```



## 4-9 객체 선택후 복사하기 활성화

```
case WM_LBUTTONDOWN :  
    mx = LOWORD(IParam);  
    my = HIWORD(IParam);  
    if (InCircle(x, y, mx, my))  
        Select = TRUE;           // 객체 선택 -> 복사하기 활성화  
    InvalidateRgn(hwnd, NULL, TRUE);  
    break;  
  
case WM_COMMAND :  
    if (LOWORD(wParam) == ID_EDITCOPY)  
    {  
        Copy = TRUE;  
        InvalidateRgn(hwnd, NULL, TRUE);  
    }  
    break;
```

# 복사하기/붙여넣기 활성화

case WM\_PAINT :

```
    EnableMenuItem(hSubMenu, ID_EDITCOPY,  
        Select?MF_ENABLED:MF_GRAYED);    // Select가 true냐 ?
```

```
    EnableMenuItem(hSubMenu, ID_EDITPASTE,  
        Copy?MF_ENABLED:MF_GRAYED);    // Copy가 true냐 ?
```

```
    hdc = BeginPaint (hwnd, &ps) ;
```

```
    if (Select)
```

```
        Rectangle(hdc, x-SIZE, y-SIZE, x+SIZE, y+SIZE);
```

```
    Ellipse(hdc, x-SIZE, y-SIZE, x+SIZE, y+SIZE);
```

```
    EndPaint (hwnd, &ps) ;
```

```
    return 0 ;
```

## 실습 4-4

### ▶ 제목

- ▶ 파일 읽고 화면에 출력하고, 윈도우에 출력된 텍스트를 파일에 저장하기

### ▶ 내용

- ▶ 파일 공용대화상자를 가지고 선택된 텍스트 파일을 읽어서 화면에 출력하는 프로그램을 작성해 본다.
- ▶ 파일 공용대화상자를 가지고 선택된 파일이름으로 화면에 출력중인 텍스트를 저장하는 프로그램을 작성해 본다.
- ▶ 파일 입출력은 표준 입출력 함수를 이용한다.
  - ▶ fopen / fclose
  - ▶ fgets
  - ▶ fputs

## 실습 4-4

### ▶ 표준 입출력 함수

- ▶ `FILE *fopen (const char *filename, const char *mode);`
  - ▶ filename: 열고자 하는 파일 이름, 경로 포함 가능
  - ▶ mode: 개방 방식, 액세스 모드
    - ▶ r (읽기 전용), w (쓰기 전용), a (추가 입력), r+, w+ (읽고 쓰는 것  
이 다 가능)
- ▶ `int fclose (FILE *stream);`
- ▶ `char fgetc (char *s, int n, FILE *stream);`
  - ▶ s: 문자열을 입력받을 문자 배열 포인터
  - ▶ n: 문자열의 길이
  - ▶ stream: 문자열을 읽어올 파일
- ▶ `int fputs (const char *s, FILE *stream);`
  - ▶ s: 출력하고자 하는 문자열
  - ▶ stream: 출력하고자 하는 대상 파일

## 4-4

```
#include <stdio.h>
#include <stdlib.h>

FILE *fPtr;
HDC hdc;
char filename[100] = "c:\\test.cpp" ;
char buffer[100];

fPtr = fopen(filename, "r");
while(fgets(buffer, 100, fPtr))
{
    printf ( "%s\n" , buffer);
}
fclose(fPtr);

fPtr = fopen(filename, "w" );
fputs (buffer, fPtr);
fclose (fPtr);
```

## 실습 4-5

### ▶ 제목

- ▶ 메뉴를 사용하여 그리기 복사, 취소하기

### ▶ 내용

- ▶ 메뉴를 만들고 실행하게 한다.
  - ▶ 그리기 객체: 직선 / 원 / 사각형
    - ▶ 선택한 객체를 임의의 위치에 임의의 크기로 그린다.
    - ▶ 디폴트 색상은 검정색이다.
    - ▶ 최대 10개의 그리기를 하도록 한다.
  - ▶ 색상: 빨강 / 초록 / 파랑 / 노랑 / 자주 / 청록 / 검정
    - ▶ 색상을 선택하면 그때부터 그리는 도형의 색상은 변경된다. (선택은 면)
  - ▶ 편집: 복사하기 / 붙여 넣기 / 취소하기
    - ▶ 복사하기: 도형의 번호가 서브메뉴로 나오고 그 중에서 선택하여 복사하도록 한다.
    - ▶ 붙여넣기: 복사해놓은 도형을 임의의 위치에 붙여넣기를 한다. 도형이 복사되어 있지 않으면 붙여넣기는 비활성화 되어 있고, 도형이 복사되어 있으면 활성화된다.
    - ▶ 취소하기: 마지막으로 복사된 것이 취소된다.

## 실습 4-6