2013년도 1학기

7장 MDI 프로그래밍

MDI 프로그래밍
(Multiple Document Interface)

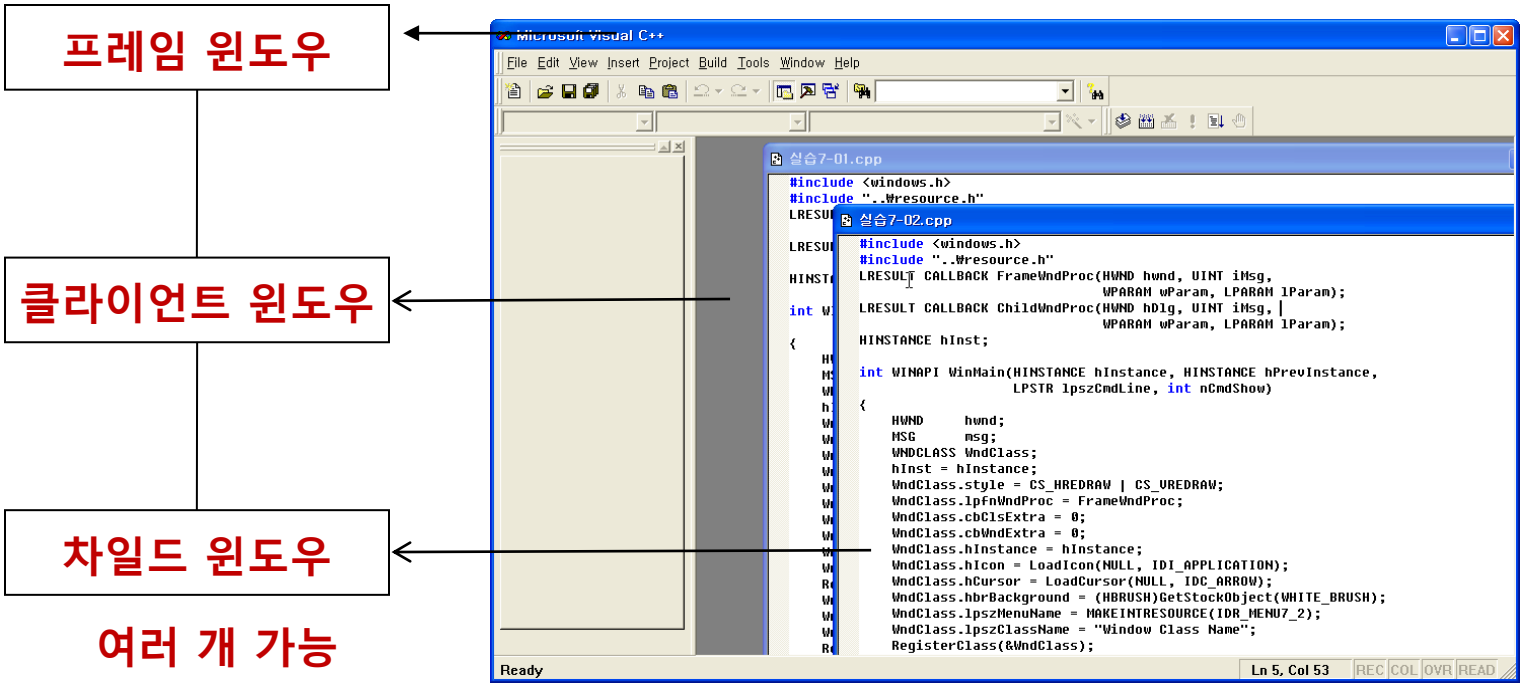
이제까지는 하나의 윈도우에 하나의 문서 처리 학습
(SDI : Single Document Interface)

- 학습목표
  - 문서창을 여러 개 열 수 있는MDI 기반 응용 프로그램을 작성할 수 있다.
  - 자식 윈도우 여러 개를 관리할 수 있다.
  - 윈도우 하나를 자식 윈도우 여러 개로 분할하여 독립적으로 활용할 수 있다.
  - 버튼, 에디트 박스, 콤보 박스, 리치 에디트 컨트롤 윈도우를 활용할 수 있다.

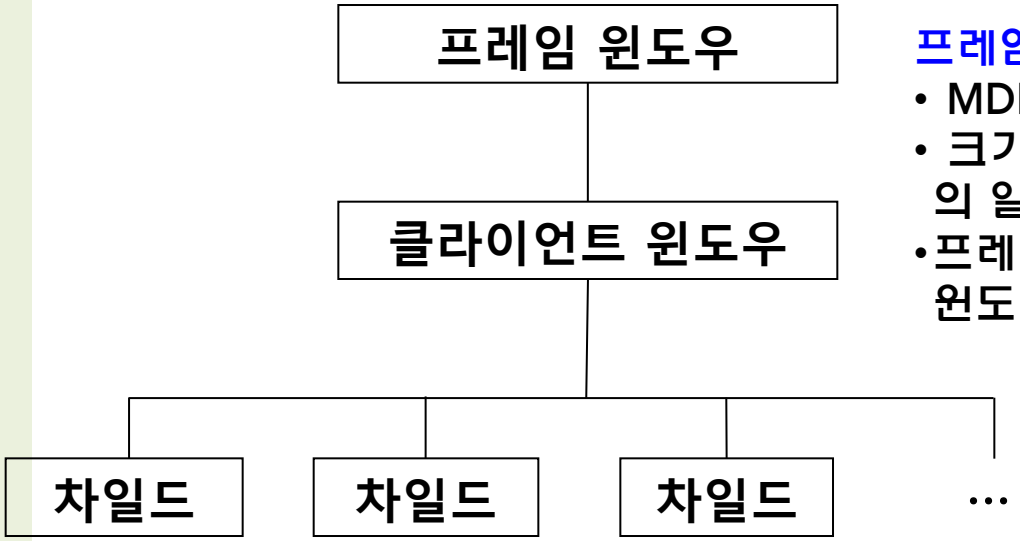
- 내용
  - MDI기반 응용 프로그램
  - 차일드 윈도우 관리
  - 윈도우 분할
  - 콘트롤 윈도우 활용하기

1절. MDI 기반 응용 프로그램

- MDI: 한 화면에 서로 다른 작업을 가능하게 여러 개의 윈도우를 생성하고 처리하도록 함



1절. MDI 기반 응용 프로그램



프레임 윈도우: 제일 바깥쪽의 윈도우
- MDI 프로그램의 형식적인 메인 윈도우
- 크기조절 경계선, 타이틀 바, 시스템 메뉴 등의 일반적인 모양
- 프레임 윈도우의 작업영역 전체는 클라이언트 윈도우가 가득 메우고 있음

클라이언트 윈도우: 짙은 회색의 가장 핵심 윈도우
- 차일드 윈도우는 클라이언트 윈도우 위에서만 움직일 수 있다.
- 차일드 윈도우를 파괴, 정렬 기능을 가지고 있음

차일드 윈도우: 실제 작업 윈도우

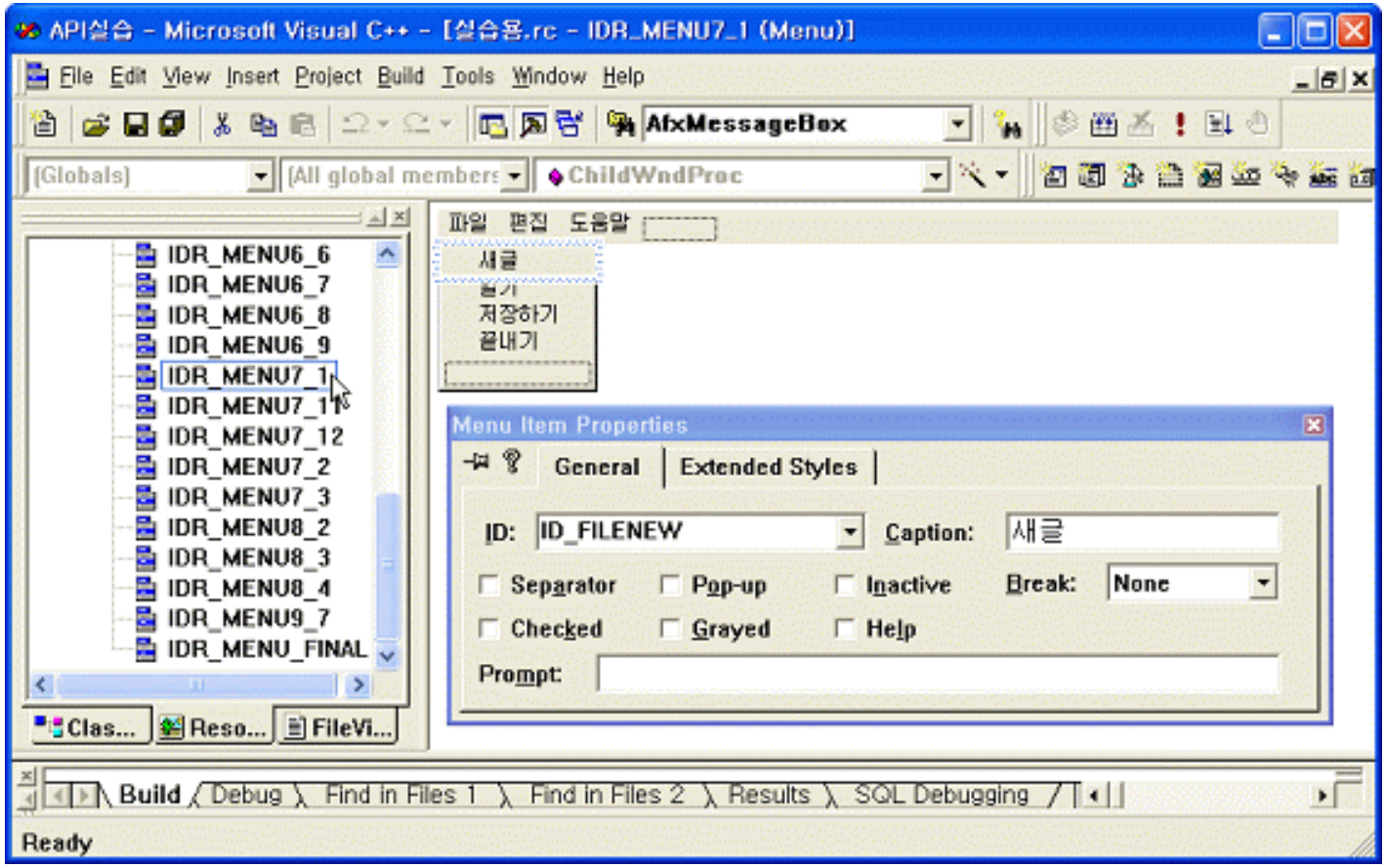
위의 3가지 윈도우를 항상 만들어야 한다.

작성방법

- 리소스 편집
  - 메뉴 작성 : 새문서 등 처리
- WinMain() 함수 작성
  - 프레임/차일드 윈도우 클래스 생성 및 등록
  - 프레임 윈도우 생성
- FrameWinProc() 함수 작성
  - 클라이언트 윈도우 생성
  - 차일드 윈도우 생성
- ChildWinProc() 함수 작성
  - 복수의 문서를 처리 가능
  - 문서간 처리 구분 필요

7-1 MDI 응용 프로그램 만들기

- 차일드 윈도우 생성 위한 “새글” 메뉴항목 추가



7-1 윈도우 클래스 생성 및 등록

```
// 프레임 윈도우 클래스 생성 및 등록

WNDCLASS  wndclass ;
wndclass.cbSize         = sizeof(wndclass) ;
wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
wndclass.lpfnWndProc    = FrameWndProc;    //  프레임 윈도우 프로시저
wndclass.cbClsExtra     = 0 ;
wndclass.cbWndExtra     = 0 ;
wndclass.hInstance      = hInstance ;
wndclass.hIcon          = LoadIcon(NULL,IDI_APPLICATION);
wndclass.hCursor        = LoadCursor(NULL,IDC_ARROW) ;
wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
wndclass.lpszMenuName   = MAKEINTRESOURCE(IDR_MENU1) ;
wndclass.lpszClassName =  "Window Class Name" ;
                                          // 프레임 윈도우 클래스 이름
wndclass.hIconSm        = LoadIcon(NULL,IDI_APPLICATION);

RegisterClass(&wndclass);          // 프레임 윈도우 클래스 등록
```

7-1 WinMain 함수 작성

- 윈도우 클래스 생성 : WNDCLASS
  - 프레임 윈도우 프로시저 등록 : SDI의 윈도우와 동일
  - 차일드 윈도우 프로시저 등록
  - 클라이언트 윈도우는 프로시저가 없음
- 두 개의 윈도우 클래스 등록 : RegisterClass()
  - 클래스 이름 (lpszClassName)으로 클래스를 구분한다.
- 프레임 윈도우 생성 : CreateWindow()
  - 클라이언트 윈도우는 WM_CREATE에서 생성

7-1 윈도우 클래스 생성 및 등록

```
// 차일드 윈도우 클래스 생성 및 등록 : 차일드를 위해 wndclass를 재사용

wndclass.lpfnWndProc   = ChildWndProc; // 차일드 윈도우 프로시저
wndclass.lpszMenuName  = NULL ;
wndclass.lpszClassName =  "Child Window Class Name" ;
                          // 차일드 윈도우 클래스 이름

RegisterClass(&wndclass);  // 차일드 윈도우 클래스 등록
```

7-1 어플리케이션 인스턴스 복사

- 응용 프로그램 인스턴스 공유
  - WinMain()함수에서 넘겨받은 hInstance 값을 FrameWndProc()함수에서도 사용
  - 전역변수를 이용하여 복사

```
HINSTANCE hInst;  // WinMain() 위에 global 변수로 위치

hInst = hInstance;  // WinMain() 함수 내에 위치
```

7-1 클라이언트 윈도우 생성

```
LRESULT CALLBACK FrameWndProc (HWND hwnd, UINT iMsg, WPARAM
    wParam, LPARAM lParam)
{
    static HWND   hwndClient ;              // 클라이언트 윈도우
    CLIENTCREATESTRUCT  clientcreate;  // 차일드 생성을 관리하기 위한 구조체

    switch (iMsg)
    {
     case WM_CREATE :        // 메뉴핸들 획득 -> 창 관리 부메뉴(0) 핸들 획득
        clientcreate.hWindowMenu = GetSubMenu(GetMenu(hwnd),0);
        clientcreate.idFirstChild = 100 ; // 첫 번째 차일드 ID

        hwndClient = CreateWindow ("MDICLIENT",
                    NULL,
                    WS_CHILD | WS_CLIPCHILDREN | WS_VISIBLE, // 윈도우 스타일
                    0, 0, 0, 0,         // 좌표
                    hwnd, NULL,
                    hInst, (LPSTR)&clientcreate) ;
        ShowWindow(hwndClient, SW_SHOW);  // 클라이언트 윈도우  출력
     ...
     }
}
```

7-1 FrameWndProc 함수 작성

- 클라이언트 윈도우
  - 차일드 윈도우를 만들고 삭제하고 관리하는 윈도우
  - FrameWndProc의 WM_CREATE에서 생성됨: 프레임 윈도우가 만들어질 때 같이 만들어져야 한다.
  - 클라이언트 윈도우 특징:
    1. 윈도우 클래스가 MDICLIENT 로 고정되어있다. (따로 등록할 필요가 없다)
    2. CLIENTCREATESTRUCT 구조체를 만든 후 CreateWindow 의 인수로 전달해야한다.
    3. 클라이언트 윈도우의 특별한 스타일로 만들어야 한다. (WS_CHILD, WS_CLIPCHILDREN은 반드시)

- 클라이언트 윈도우 생성절차
  - 서브메뉴 윈도우 이용 : GetSubMenu()
  - WM_CREATE : CreateWindow(), 클라이언트 구조체 이용(아래)
  - ShowWindow()

- CLIENTCREATESTRUCT 구조체
  - 클라이언트 윈도우가 차일드 윈도우 관리를 위한 정보를 보유

7-1 FrameWndProc 함수 작성

hWindowMenu: 차일드 목록 관리에 사용될 메뉴 핸들

idFirstChild: 첫 번째 차일드 윈도우의 목록이 출력될 메뉴 항목

7-1 FrameWndProc 함수 작성

- 참고: CreateWindow 함수 구성

```
hwnd = CreateWindow // 윈도우가 생성되면 핸들(hwnd)이 반환됨
        (
              "MDICLIENT" ,                    // 윈도우 클래스 이름
              "Client Title Name",             // 윈도우 타이틀 이름
              WS_CHILD|WS_CLIPCHILDREN|WS_VISIBLE,
                                               // 윈도우 스타일
              0,                               // 윈도우 위치, x좌표
              0,                               // 윈도우 위치, y좌표
              CW_USEDEFAULT,                   // 윈도우 폭
              CW_USEDEFAULT,                   // 윈도우 높이
              NULL,                            // 부모 윈도우 핸들
              NULL,                            // 메뉴 핸들
              hInstance,                       // 응용 프로그램 ID
              NULL                             // 생성된 윈도우 정보
         ) ;
```

7-1 클라이언트 윈도우 생성

- **클라이언트 윈도우가 차일드 윈도우 관리**(생성, 유지, 삭제)
- CLIENTCREATESTRUCT 구조체: 차일드 윈도우 생성에 필요

```
typedef struct {
    HANDLE      hWindowMenu;
    UINT        idFirstChild;
} CLIENTCREATESTRUCT;
```

  - 차일드 윈도우 관리에 사용될 메뉴 ID에 대한 정보를 클라이언트 윈도우에게 제공한다.
  - 클라이언트 윈도우는 차일드 윈도우가 생성, 파괴될 때 차일드 목록을 메뉴 항목으로써 관리 –> 목록을 작성하고 관리할 위치를 알려주어야 한다.
    - hWindowMenu: 응용 프로그램의 메인 메뉴중 선택된 부메뉴에 대한 핸
    - idFirstChild: 생성될 차일드 윈도우의 ID로 사용될 정수. 클라이언트 윈도우는 이 값을 이용하여 첫번째 차일드 윈도우를 생성하며 두번째 차일드 윈도우부터는 이 값에 1씩 증가된 정수를 ID로 사용한다.

7-1 등록된 윈도우 클래스 종류

- button: 버튼 콘트롤 윈도우
- combobox: 콤보 박스 윈도우
- edit: 에디트 박스 윈도우
- listbox: 리스트 박스 윈도우
- MDICLIENT: MDI 클라이언트 윈도우
- RichEdit: 리치에디트 윈도우

7-1 차일드 윈도우 생성

```
MDICREATESTRUCT     mdicreate ;
HWND                hwndChild ;

switch (iMsg) {
case WM_COMMAND:
    switch(LOWORD(wParam))
    {
    case ID_FILENEW:        // 새문서
        mdicreate.szClass = "Child Window Class Name" ;
        mdicreate.szTitle = "Child Window Title Name" ;
        mdicreate.hOwner  = hInst ;// 프로그램 인스턴스 핸들, 전역번수로 선언
        mdicreate.x       = CW_USEDEFAULT ; // 자식 윈도우 X좌표
        mdicreate.y       = CW_USEDEFAULT ; // 자식 윈도우 Y좌표
        mdicreate.cx      = CW_USEDEFAULT ; // 자식 윈도우 폭
        mdicreate.cy      = CW_USEDEFAULT ; // 자식 윈도우 높이
        mdicreate.style   = 0 ;
        mdicreate.lParam  = 0 ;          // MDI 클라이언트 윈도우를 만들어라
        hwndChild = (HWND) SendMessage (hwndClient, WM_MDICREATE,
                        0,(LPARAM) (LPMDICREATESTRUCT) &mdicreate) ;
        return 0 ;
```

7-1 차일드 윈도우 생성

- 메뉴에서 새문서(ID_FILENEW)를 선택했을 때 차일드 윈도우 생성하는 코드를 FrameWndProc()의 WM_COMMAND에 추가
- 메뉴선택에 의한 차일드 윈도우 생성
  - WM_COMMAND에서 **SendMessage()**를 사용하여 WM_MDICREATE 메시지를 보내서 차일드 윈도우를 만든다.
  - 차일드 윈도우는 클라이언트 윈도우의 자식이므로 생성은 클라이언트가 직접 해야한다. (프레임 윈도우는 메뉴 명령만 입력받고, 차일드 생성에 필요한 정보를 클라이언트 윈도우에 전달한다.)
  - **CreateMDIWindow** (LPSTR lpClassName, LPSTR lpWindowName, DWORD dwStyle, int X, int Y, int nWidth, int nHeight, hWndParent, HINSTANCe hInstance LPARAM lParam) 함수를 사용하여 차일드 윈도우를 생성할 수 도 있다.

7-1 차일드 윈도우 생성

- typedef struct {
  LPCTSTR szClass; // 클래스 이름
  LPCTStR szTitle; // 타이틀 이름
  HANDLE hOwner; // 응용 프로그램 핸들
  int x; // x 값
  int y, // y 값
  int cx; // 폭
  int cy; // 높이
  DWORD style; // 윈도우 스타일
  LPARAM lParam; // 윈도우 생성 정보
  } MDICREATESTRUCT

스타일에 : MDIS_ALLCHILDSTYLES (WS_MINIMIZE, WS_MAXIMIZE, WS_HSCROLL, WS_VSCROLL 스타일을 모두 준 것과 같다)

7-1 차일드 윈도우 생성

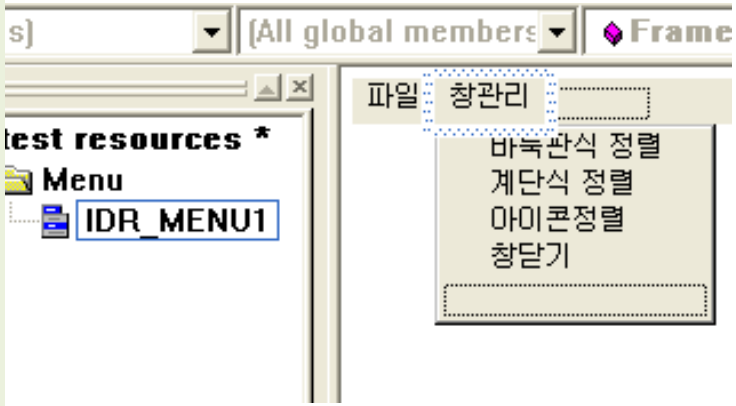
- MDI메시지 처리 (클라이언트 윈도우로 전달되는 메시지)
  - WM_MDIACTIVATE: 다른 차일드 윈도우를 활성화,
  - WM_MDICASCADE: 차일드 윈도우를 계단식으로 정렬
  - WM_MDICREATE: 차일드 윈도우를 만든다.
  - WM_MDIDESTROY: wParam 로 전달된 차일드 윈도우를 파괴한다.
  - WM_MDIGETACTIVE: 현재 활성화된 차일드의 핸들을 리턴
  - WM_MDIICONARRANGE: 아이콘 정렬
  - WM_MDIMAXIMIZE: wParam로 지정한 차일드 윈도우를 최대화
  - WM_MDINEXT: 지정한 차일드 윈도우의 앞 또는 뒤쪽 차일드 윈도우를 활성화
  - WM_MDIREFRESHMENU: 윈도우 메뉴를 리프레시시킨다.
  - WM_MDIRESTORE: 최대화 또는 최소화된 윈도우를 원래 크기대로 복구
  - WM_MDISETMENU:메뉴 변경
  - MW_MDITILE: 바둑판 식으로 재정렬

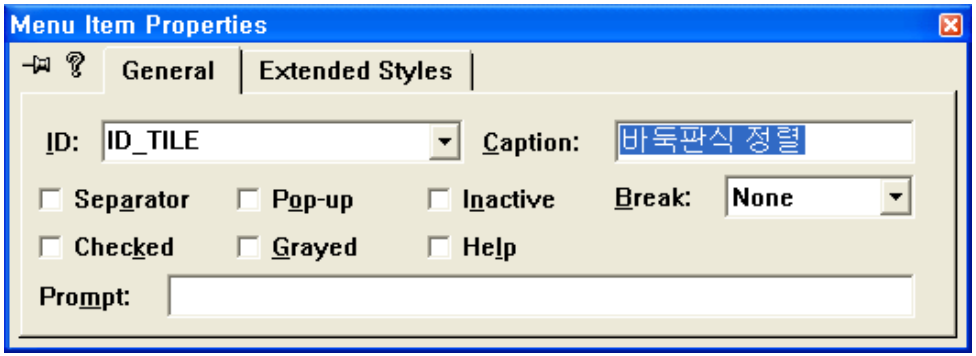
7-1 차일드 윈도우 메시지 처리함수

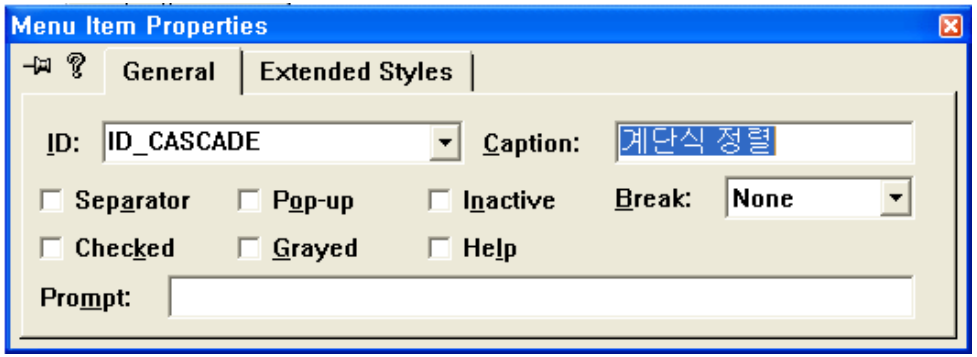
```
LRESULT CALLBACK ChildWndProc (HWND hwnd, UINT iMsg,
                               WPARAM wParam, LPARAM lParam)
{
  switch (iMsg)    // 현재 특별한 내용은 정의되어 있지 않음
  {
    case WM_CREATE:
          break;
    case WM_DESTROY:
          return 0;
  }

  return DefMDIChildProc (hwnd, iMsg, wParam, lParam);

    // 차일드 윈도우 메시지 함수에서는 메시지 처리한 후 마지막으로 호출하는
    // 함수가 DefMDIChildProc (hwnd, iMsg, wParam, lParam)이다
}
```

2절. 차일드 윈도우 관리

다음과 같이 메뉴항목을 변경

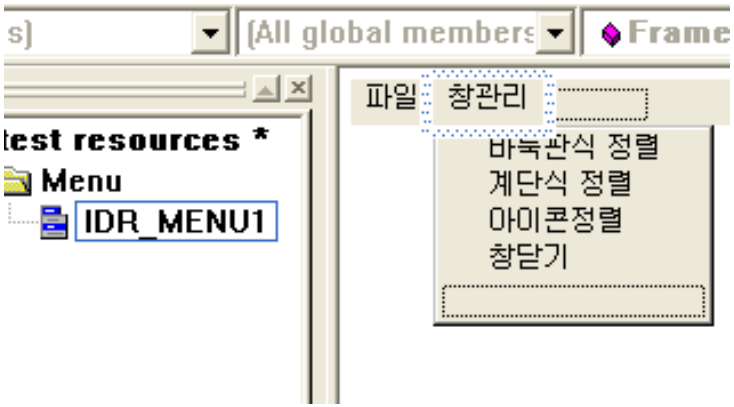


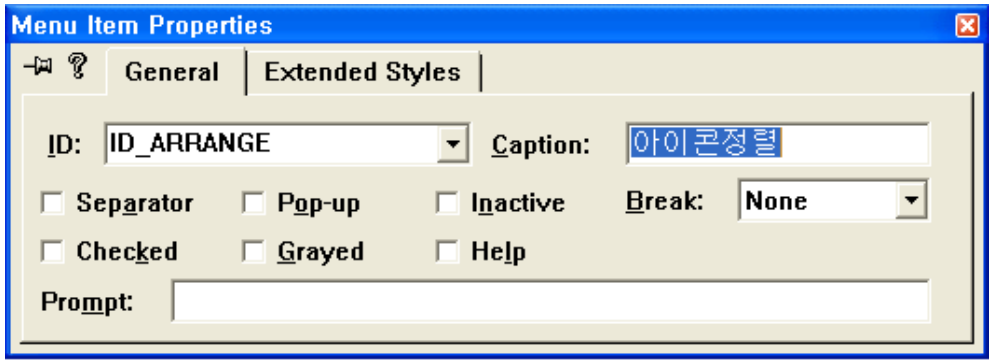


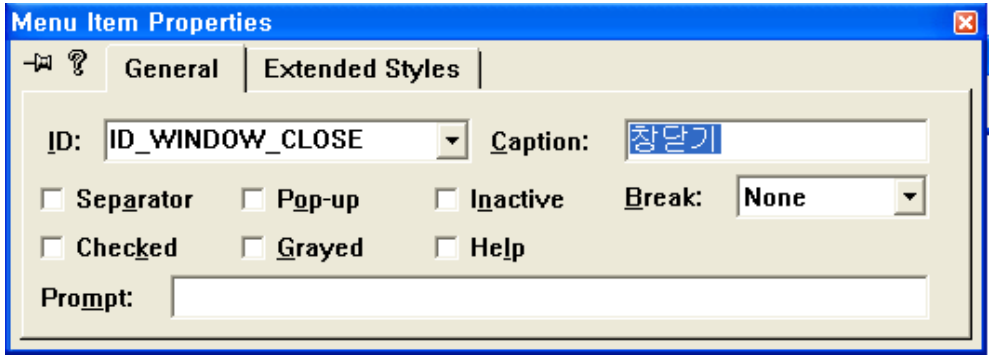


7-2 차일드 윈도우 관리

다음과 같이 메뉴항목을 변경







7-2 메뉴항목 처리 : FrameWndProc()

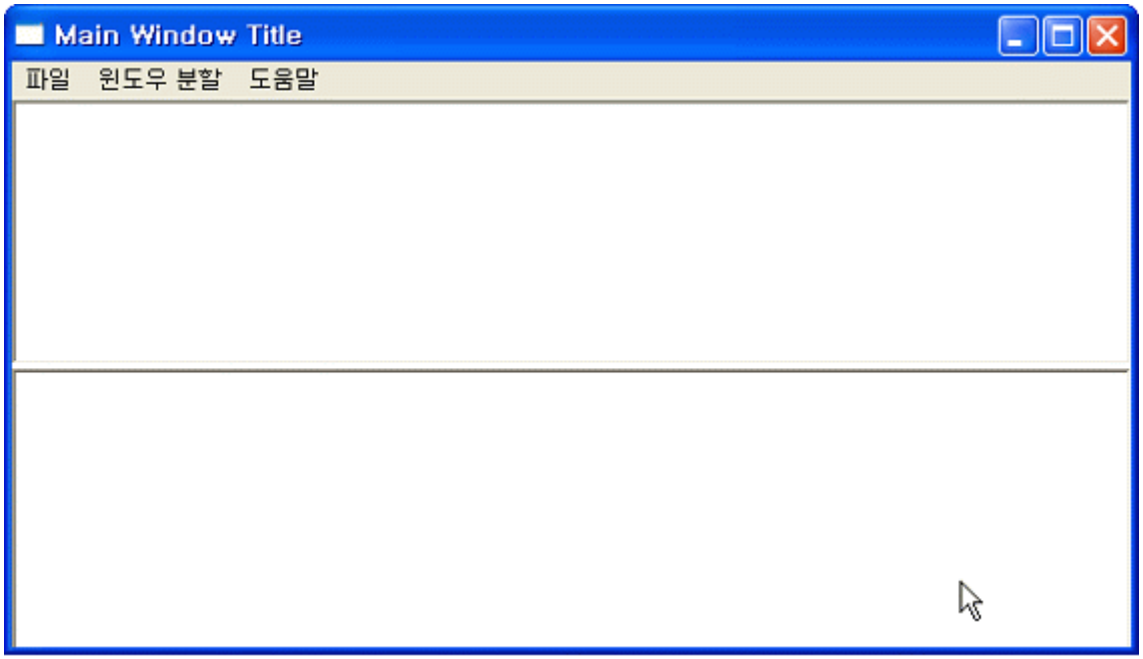
```
case WM_COMMAND:
    switch(LOWORD(wParam))
    {                   // 한번에 하나의 차일드 종료

    case ID_WINDOW_CLOSE : // 창닫기 처리, 모든 차일드 윈도우 화면 닫기
          // 클라이언트를 통해, 활성화(Active)된  차일드 윈도우 탐색
        hwndChild = (HWND) SendMessage (hwndClient,
                                        WM_MDIGETACTIVE, 0, 0) ;
                    // 클라이언트에서 특정 차일드를 종료해도 되나 문의
        if (SendMessage (hwndChild, WM_QUERYENDSESSION,0,0))
                    SendMessage (hwndClient, WM_MDIDESTROY,
                                        (WPARAM) hwndChild, 0) ; // 종료처리
        return 0 ;

    case ID_TILE : // 클라이언트에게 윈도우의 타입지정을 요청
        SendMessage (hwndClient, WM_MDITILE, 0, 0) ;
        return 0 ;
        …
```

7-2 메뉴항목 처리 : FrameWndProc()

```
case ID_CASCADE :
        SendMessage (hwndClient, WM_MDICASCADE, 0, 0);
        return 0 ;
case ID_ARRANGE :
        SendMessage (hwndClient, WM_MDIICONARRANGE, 0, 0);
        return 0 ;
case ID_EXIT :
        PostQuitMessage (0);
        return 0 ;
```

3절. 윈도우 분할

- 프레임 윈도우를 분할하여 차일드 윈도우 관리
  - 분할된 윈도우는 자식 윈도우이지만 팝업 윈도우는 아니므로 자식 윈도우에 타이틀 바를 포함하는 독립적인 프레임이 존재하지는 않는다.
  - 분할 윈도우도 메인 윈도우와 같은 방법으로 생성하고, CreateWindowEx 함수를 사용한다.



차일드 윈도우 생성 함수

```
HWND CreateWindowEx(
        DWORD dwExStyle,          // 생성되는 확장 윈도우의 스타일
        LPCTSTR lpClassName,      // 등록된 윈도우클래스
        LPCTSTR lpWindowName,     // 윈도우 타이틀 텍스트
        DWORD dwStyle,            // 기본 윈도우 스타일
        int x,                    // 생성 윈도우 위치의 x값
        int y,                    // 생성 윈도우 위치의 y값
        int nWidth,               // 생성 윈도우의 너비
        int nHeight,              // 생성 윈도우의 높이
        HWND hWndParent,          // 부모 윈도우 핸들
        HMENU hMenu,              // 사용될 메뉴의 핸들
        HINSTANCE hInstance,      // 어플리케이션 인스턴스
        LPVOID lpParam
        );
```

차일드 윈도우 생성 함수

dwExStyle 스타일

스타일	내용
WS_EX_DLGMODALFRAME	이중 경계선을 가진 윈도우를 만든다
WS_EX_WINDOWEDGE	양각 모양의 경계선을 가진 윈도우를 만든다.
WS_EX_CLIENTEDGE	작업영역이 쑥 들어간 음각 모양으로 만든다.
WS_EX_MDICHILD	MDI 차일드 윈도우를 만든다.
WS_EX_OVERLAPPEDWINDOW	(WS_EX_WINDOWEDGE | WS_EX_CLIENTEDGE)복합 속성

7-3 윈도우 분할 실습

```
HWND    ChildHwnd[2];
LRESULT CALLBACK FrameWndProc (HWND hwnd, UINT iMsg,
          WPARAM wParam, LPARAM lParam)
{
  RECT rectView;
  switch (iMsg)
  {
     case WM_CREATE:
           GetClientRect(hwnd, &rectView);
           ChildHwnd[0] = CreateWindowEx( WS_EX_CLIENTEDGE,
                          "Child Window Class Name",         NULL,
                          WS_CHILD | WS_VISIBLE, 0, 0, rectView.right,
                          rectView.bottom/2-1, hwnd, NULL, hInst, NULL );
           ChildHwnd[1] = CreateWindowEx( WS_EX_CLIENTEDGE,
                          "Child Window Class Name",         NULL,
                          WS_CHILD | WS_VISIBLE, 0, rectView.bottom/2+1,
                          rectView.right, rectView.bottom/2-1,
                          hwnd, NULL, hInst, NULL );
```

윈도우 크기 재조정 함수

윈도우의 위치와 크기를 변경하는 가장 일반적인 함수이다. X,Y 인수로 윈도우의 위치를 지정하며 nWidth, nHeight 인수로 윈도우의 폭과 높이를 지정하므로 이 함수로 위치와 크기를 한꺼번에 변경할 수 있다.

```
BOOL MoveWindow(
        HWND hWnd,          // 윈도우의 핸들
        int X,              // 윈도우의 새로운 위치 x좌표
        int Y,              // 윈도우의 새로운 위치 y좌표
        int nWidth,         // 윈도우의 새로운 너비
        int nHeight,        // 윈도우의 새로운 높이
        BOOL bRepaint       // 재조정후 재출력 여부,
                            // TRUE 면 다시그리기
        );
```

7-4 메인 윈도우 크기에 따라 조정

```
HWND    ChildHwnd[2];
LRESULT CALLBACK FrameWndProc (HWND hwnd, UINT iMsg,
                               WPARAM wParam, LPARAM lParam)
{
  static BOOL       split;
  RECT    rectView;
  switch (iMsg)
  {
    case WM_CREATE:
          split = FALSE;
          return 0;
    case WM_COMMAND:
          switch(LOWORD(wParam)) {
              case ID_SPLIT_2_1:
                    if (split == TRUE) break;
                    split = TRUE;
                    GetClientRect(hwnd, &rectView);
                    ChildHwnd[0] = CreateWindowEx(...);
                    ChildHwnd[1] = CreateWindowEx(...);
          return 0;
```

7-4 메인 윈도우 크기에 따라 조정

```
case ID_EXIT :
    PostQuitMessage (0);
    return 0 ;
  }
  return 0;
case WM_SIZE:
  if (split == TRUE)
  {
      GetClientRect (hwnd, &rectView);
      MoveWindow (ChildHwnd[0], 0, 0, rectView.right,
              rectView.bottom/2-1, TRUE);

      MoveWindow (ChildHwnd[1], 0, rectView.bottom/2+1,
             rectView.right, rectView.bottom/2-1, TRUE);
  }
  return 0;
```

마우스를 윈도우내에 캡춰하는 함수

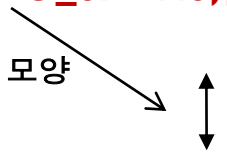
- 자식 윈도우 크기를 각각 조정할 때,
  - 자식 윈도우 사이의 경계를 움직일 수 있어야 한다.
  - 자식 윈도우 사이의 경계를 움직이게 하기 위해서 마우스 커서가 경계선에 있을 때와 크기를 조정 중일 때, 커서의 모양을 바꾼다.
  - 크기를 조정하기 위해서는 경계선에서 마우스를 드래그하여 경계선의 위치를 위나 아래로 이동시킨다.
  - 자식윈도우의 최소 크기는 보장해야 한다.

- 커서가 윈도우의 영역 밖을 벗어나더라도 드래그 동작을 할 때, 계속해서 마우스 메시지를 보내주어서 크기를 조정하게 한다. 마우스를 윈도우내에 캡춰하는 함수
  - HWND SetCapture (HWND hWnd);
  - 마우스 캡춰를 해제하는 함수

    ReleaseCapture();

7-5 차일드 윈도우 크기 조정

```
HWND ChildHwnd[2];
 LRESULT CALLBACK FrameWndProc (HWND hwnd, UINT iMsg, WPARAM wParam,
     LPARAM lParam)
{
  static BOOL       split;
  static HCURSOR  hCursor;
  static int          boundary = -1;

case WM_COMMAND:
     switch (LOWORD (wParam)) {
     case ID_SPLIT_2_1:
        split = TRUE;
        GetClientRect (hwnd, &rectView);
        ChildHwnd[0] = CreateWindowEx(...);
        ChildHwnd[1] = CreateWindowEx(...);
        boundary = rectView.bottom/2;
        hCursor = LoadCursor(NULL, MAKEINTRESOURCE(IDC_SIZENS));
        return 0;
```



7-5 차일드 윈도우 크기 조정

```
case WM_LBUTTONDOWN:
  if (split == TRUE)
  {
        SetCursor(hCursor);
        SetCapture(hwnd);
  }
  return 0;

case WM_LBUTTONUP:
  if (split == TRUE)
        ReleaseCapture();
  return 0;
```

7-5 차일드 윈도우 크기 조정

```
case WM_MOUSEMOVE: // y = HIWORD(lParam), x = LOWORD(lParam)

 if (HIWORD(lParam)>=boundary-1 && HIWORD(lParam)<=boundary+1)
        SetCursor(hCursor);

 if (wParam == MK_LBUTTON && split == TRUE)
 {
  GetClientRect(hwnd, &rectView);
  // 자식 윈도우 크기가 너무 작아지지 않도록 조건 조사
  if (rectView.top+5 < HIWORD(lParam) &&
                          HIWORD(lParam) < rectView.bottom-5)
        boundary = HIWORD(lParam);

  MoveWindow(ChildHwnd[0], 0, 0, rectView.right, boundary - 1, TRUE);
  MoveWindow(ChildHwnd[1], 0, boundary+1, rectView.right ,
                rectView.bottom - boundary + 1, TRUE);
 }
 return 0;
```

7-6 차일드 윈도우 메시지 처리

각 자식 윈도우에서 마우스 왼쪽 버튼을 눌렀을 때 타이머 이벤트가 발생하며 원을 그리게 하는 프로그램

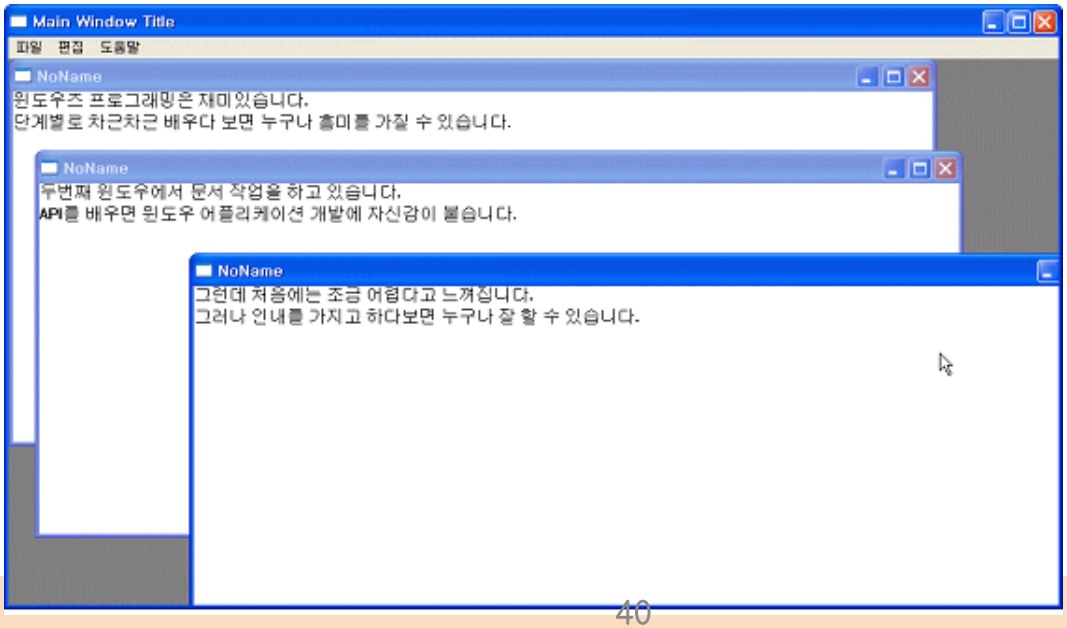
```
LRESULT CALLBACK ChildWndProc(HWND hwnd,UINT iMsg,
                              WPARAM wParam,LPARAM lParam)
{
    HDC hdc;
    static int x[2]={20,20}, y[2]={20,20}, flag[2];
    int select;


    case WM_TIMER:
        x[wParam] = x[wParam] + 20;
        hdc = GetDC(hwnd);
        Ellipse(hdc, x[wParam]-20, y[wParam]-20,
                              x[wParam]+20, y[wParam]+20);
        ReleaseDC(hwnd, hdc);
        break;
```

7-6 차일드 윈도우 메시지 처리

```
case WM_LBUTTONDOWN:
    if (hwnd == ChildHwnd[0])
        select = 0;
    else
        select = 1;
    flag[select] = 1 - flag[select];
    if (flag[select])
        SetTimer(hwnd, select, 100, NULL);
    else
        KillTimer(hwnd, select);
    break;
```

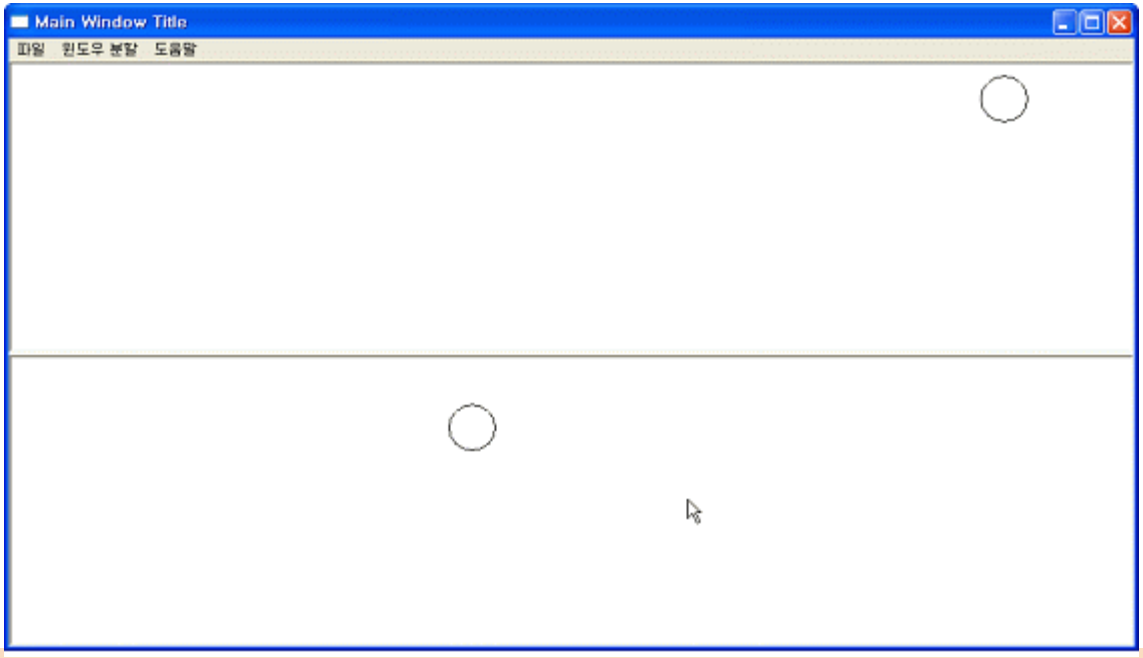
실습 7-1

- 제목
  - MDI기반 10라인 에디터
- 내용
  - 실습 7-1에 각 차일드 윈도우가 10라인 입력 에디터가 되도록 작성한다.
  - "새문서" 메뉴항목을 선택하면 새로운 차일드 윈도우가 뜨고 거기에 10줄짜리 글을 입력 및 편집할 수 있어야 한다.
  - 각 차일드 윈도우의 이름을 다르게 한다.



실습 7-2

- 제목
  - 분할된 윈도우에 공 바운드 하기
- 내용
  - 2개의 차일드 윈도우에서 원이 바운드 되는 프로그램을 작성한다.
  - 마우스 버튼의 한번 클릭하면 클릭한 차일드 윈도우의 원은 움직이기 시작하고 한번 더 클릭하면 원이 멈춘다.



실습 7-3

4절. 콘트롤 윈도우 활용하기

- 컨트롤도 윈도우로 부모 윈도우 아래의 자식 윈도우로 존재한다.
- 사용자가 만든 윈도우를 부모 윈도우로 두고 컨트롤을 차일드 윈도우로 만들어 본다.
  - 버튼 콘트롤
    - 윈도우 클래스: “button”
  - 에디트 콘트롤
    - 윈도우 클래스: “edit”
  - 콤보박스
    - 윈도우 클래스: “combobox”

7-7 버튼 콘트롤 윈도우 생성

- 버튼 생성: CreateWindow 함수로 버튼을 만든다.
- 버튼의 종류

BS_PUSHBUTTON	푸시 버튼
BS_DEFPUSHBUTTON	디폴트 푸시 버튼
BS_CHECKBOX	체크 박스
BS_3STATE	3가지 상태를 가지는 체크 박스
BS_AUTOCEHCKBOX	자동 체크 박스
BS_AUTO3STATE	3가지 상태를 가지는 자동 체크 박스
BS_RADIOBUTTON	라디오 버튼
BS_GROUPBOX	그룹 박스

- 버튼의 스타일:
  - WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON
- 메뉴: 컨트롤의 id로 사용한다.

7-7 버튼 콘트롤 윈도우 생성

- 버튼 이벤트 (버튼 통지)

Notify	Meaning
BN_CLICKED	버튼 위에서 마우스가 클릭되었을 때
BN_DBCLK	버튼 위에서 마우스가 더블 클릭되었을 때
BN_SETFOCUS	버튼 위 마우스 커서가 올 때
BN_KILLFOCUS	버튼 위에서 마우스가 벗어날 때
BN_PAINT	버튼 내부를 Drawing 할 때

- WM_COMMAND 메시지
  - HIWORD(wParam): 통지 코드
  - LOWORD(wParam): 컨트롤의 ID

7-7 버튼 콘트롤 윈도우 생성

```
#define IDC_BUTTON 100   // 버튼 콘트롤의 ID
LRESULT CALLBACK WndProc (HWND hwnd, UINT iMsg,
                          WPARAM wParam, LPARAM lParam)
{
  HDC hdc;
  HWND hButton;
  switch (iMsg)
  {
   case WM_CREATE:
      hButton = CreateWindow ( "button",  "확인",
                               // 버튼의 윈도우 클래스 이름은  “button”
                               WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON,
                               // 차일드 윈도우이고 누르는 형태의 버튼 스타일
                               200, 0, 100, 25, hwnd,
                               (HMENU)IDC_BUTTON, hInst, NULL);
      return 0;
```

7-7 버튼 콘트롤 윈도우 생성

```
case WM_COMMAND:
   switch(LOWORD(wParam)) {
   case IDC_BUTTON:
      hdc = GetDC(hwnd);
      TextOut(hdc, 0, 100, "Hello World", 11);
      ReleaseDC(hwnd, hdc);
      return 0;
   }
   return 0;
```

7-8 에디트 콘트롤 윈도우 생성

•에디트 컨트롤 윈도우 스타일

스타일	의미
ES_AUTOHSCROLL	수평 스크롤을 지원
ES_AUTOVSCROLL	여러 줄을 편집할 때 수직 스크롤을 지원
ES_LEFT	왼쪽 정렬
ES_RIGHT	오른쪽 정렬
ES_CENTER	중앙 정렬
ES_LOWERCASE	소문자로 변환하여 표시
ES_UPPERCASE	대문자로 변환하여 표시
ES_MULTILINE	여러 줄을 편집
ES_READONLY	읽기 전용, 편집할 수 없다.
ES_PASSWORD	입력되는 모든 문자를 *로 보여준다.

7-8 에디트 콘트롤 윈도우 생성

•에디트 컨트롤 메시지 통지

메시지	의미
EN_CHANGE	Editbox의 내용이 변경된 후 발생 (화면에 갱신된 후)
EN_UPDATE	Editbox 내용이 변경되려고 할 때 발생 (사용자가 타이프한 후 화면에 갱신되기 직전에 발생)
EN_SETFOCUS	포커스를 받을 때 발생
EN_KILLFOCUS	포커스를 잃을 때 발생
EN_HSCROLL/EN_VSCROLL	수평 / 수직 스크롤바 클릭
EN_MAXTEXT	지정한 문자열 길이를 초과
EN_ERRSPACE	메모리 부족

7-8 에디트 콘트롤 윈도우 생성

```
#define IDC_BUTTON 100   // 에디트 콘트롤의 ID
#define IDC_EDIT   101
...생략...
  static HWND hButton, hEdit;
  char str[100];
      ...생략...
   case WM_CREATE:
      ...생략...
      hEdit = CreateWindow("edit", "에디팅", WS_CHILD | WS_VISIBLE | WS_BORDER,
                  // 에디트 윈도우 클래스 이름은  “edit”
                  // 차일드 윈도우이고 에디트 박스 주위에 테두리가 있는 스타일
                 0, 0, 200, 25, hwnd, (HMENU)IDC_EDIT, hInst, NULL);
      return 0;
```

7-8 에디트 콘트롤 윈도우 생성

```
case WM_COMMAND:
   switch(LOWORD(wParam)) {
   case IDC_BUTTON:
      GetDlgItemText(hwnd, IDC_EDIT, str, 100);
      hdc = GetDC(hwnd);
      TextOut(hdc, 0, 100, str, strlen(str));
      ReleaseDC(hwnd, hdc);
      return 0;
   }
```

7-9 콤보 박스 윈도우 생성

- 콤보 박스 스타일
  - CBS_SIMPLE — 에디트만 가진다.
  - CBS_DROPDOWN — 에디트와 리스트 박스를 가진다.
  - CBS_DROPDOWNLIST — 리스트 박스만 가지며 에디트에 항목을 입력할 수는 없다.
  - CBS_AUTOHSCROLL — 콤보 박스에서 항목을 입력할 때 자동 스크롤

콤보 박스에 전달되는 메시지 종류

- CB_ADDSTRING: 스트링 추가 메시지로
  - wParam은 사용하지 않음
  - lParam에 스트링의 시작주소
- CB_DELETESTRING: 스트링을 삭제하는 메시지
  - wParam에 삭제할 스트링의 인덱스 번호
  - lParam은 사용하지 않음
- CB_GETCOUNT: 저장되어 있는 스트링 아이템의
  - wParam, lParam은 사용하지 않음
  - 개수는 SendMessage()의 반환 값
- CB_GETCURSEL: 선택된 스트링이 리스트에서 몇 번째 것인지
  - wParam, lParam은 사용하지 않음
  - 개수는 SendMessage()의 반환 값

7-9 콤보 박스 윈도우 생성

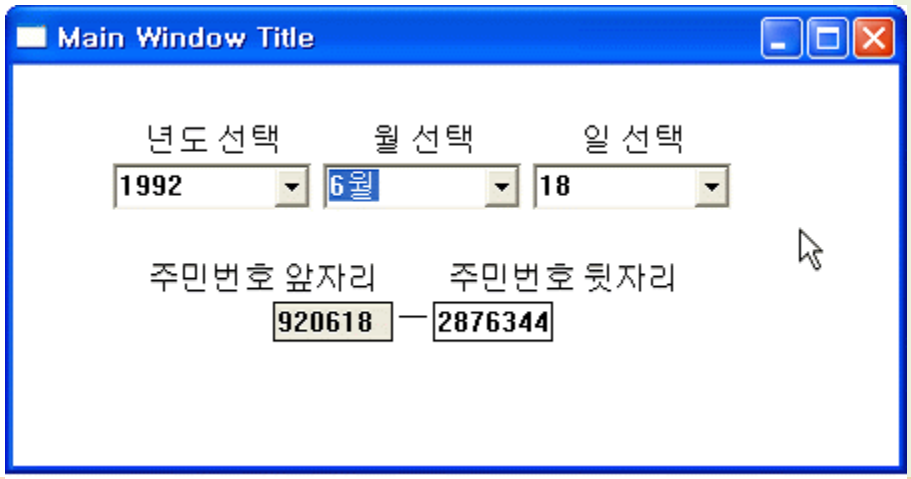
```
#define IDC_BUTTON 100
#define IDC_EDIT   101
#define IDC_COMBO  102 // 콤보 박스 콘트롤의 ID

static HWND hButton, hEdit, hCombo;
  char str[100];
  switch (iMsg)
  {
   case WM_CREATE:

      hCombo = CreateWindow("combobox", NULL,
                     WS_CHILD | WS_VISIBLE | CBS_DROPDOWN,
                     // 콤보 박스의 윈도우 클래스 이름은  "combobox"
                     // 차일드 윈도우이고 콤보 박스는 아래로 내용을 보여주는 형태
                     0, 100, 200, 300, hwnd, (HMENU)IDC_COMBO, hInst, NULL);
      return 0;
```

7-9 콤보 박스 윈도우 생성

```
case WM_COMMAND:
    switch(LOWORD(wParam)) {
    case IDC_BUTTON:
      GetDlgItemText(hwnd, IDC_EDIT, str, 100);
      if (strcmp(str, ""))
          SendMessage (hCombo,CB_ADDSTRING,0,(LPARAM)str);
      return 0;
    }
    return 0;
```

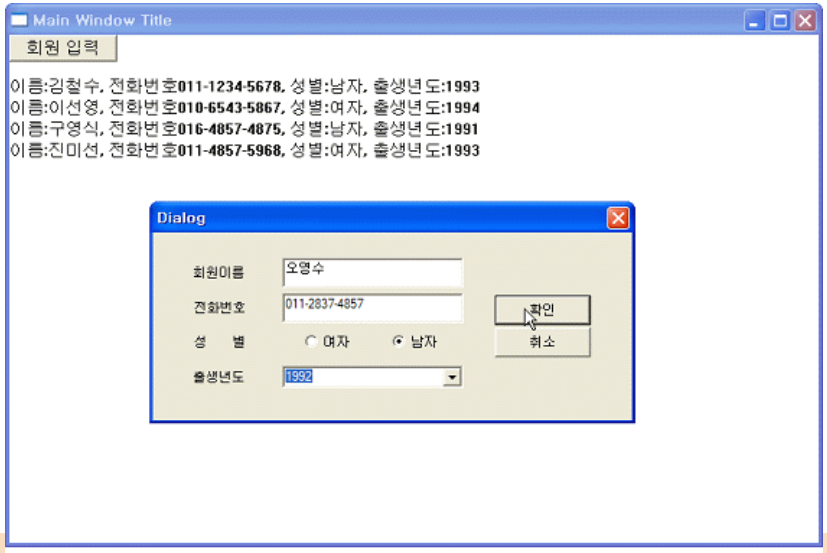
실습 7-4

- 제목
  - 주민번호 앞자리 자동 생성
- 내용
  - 년도와 월, 일을 선택하면 주민번호 앞자리가 완성되고 읽기 전용 에디트 박스에 출력한다.
  - 주민번호 뒷자리는 직접 입력해야 한다.
  - 월을 선택하고 나면 그에 따라 "일 선택"에 대한 콤보박스가 완성된다.
    즉 2월을 선택했다면 "일 선택" 콤보박스에는 1일부터 28일까지만 나온다.



실습 7-5

- 제목
  - 회원정보 입력하기
- 내용
  - "회원입력" 버튼을 클릭하면 대화상자가 나타난다. 대화상자에 회원정보를 입력하고 "확인"버튼을 누르면 회원 정보가 메인 윈도우 화면에 출력된다. 계속해서 "회원입력" 버튼을 눌러서 여러 명의 회원 정보가 입력되면 차례대로 메인 윈도우에 회원 정보가 출력된다.



실습 7-6