Rejestracja klasy okna

```
void MyRegisterClass()
{
        WNDCLASSEX wcex;

        wcex.cbSize = sizeof(WNDCLASSEX);
        wcex.lpszClassName ="KLASA";
        wcex.lpfnWndProc = (WNDPROC)WndProc;
        wcex.style = CS_HREDRAW | CS_VREDRAW;
        wcex.cbClsExtra = 0;
        wcex.cbWndExtra = 0;
        wcex.hInstance = hInst;
        wcex.hIcon = LoadIcon(NULL,IDI_APPLICATION);
        wcex.hCursor = LoadCursor(NULL, IDC_ARROW);
        wcex.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
        wcex.lpszMenuName = NULL;
        wcex.hIconSm = NULL;

        RegisterClassEx(&wcex);  // Rejestracja klasy
}
                                        }
```

Rejestracja klasy okna – c.d.

Struktura WNDCLASSEX:

cbSize - rozmiar struktury
lpszClassName - nazwa rejestrowanej klasy (potrzebna przy tworzeniu okna)
lpfnWndProc - adres procedury okna
style - styl klasy (okno ma być odświeżane po zmianie szerokości lub wysokości)
cbClsExtra - liczba bajtów związanych z klasą do dyspozycji programisty
cbWndExtra - liczba bajtów przechowywanych z każdym oknem klasy do dyspozycji programisty
hInstance - uchwyt do aplikacji
hIcon - uchwyt do ikony aplikacji
hCursor - uchwyt do domyślnego kursora myszy
hbrBackGround - uchwyt do tła (ang. *brush*)
lpszMenuName - nazwa menu

Utworzenie okna

```
BOOL MyCreate(int nCmdShow)
{

HWND hWnd = CreateWindow("KLASA", "Pierwsza aplikacja",
                         WS_OVERLAPPEDWINDOW, CW_USEDEFAULT, 0,
                         CW_USEDEFAULT, 0, NULL, NULL, hInst, NULL);

if (!hWnd)
        return FALSE;

ShowWindow(hWnd, nCmdShow);
UpdateWindow(hWnd);
 return TRUE;
}
```

Procedura okna

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM
wParam, LPARAM lParam)
{
	static char szCzesc[]="Witaj w Windows";

	PAINTSTRUCT ps;
	HDC hdc;

	switch (message)
	{
		case WM_DESTROY:
			PostQuitMessage(0);
			break;

		case WM_PAINT:
			hdc = BeginPaint(hWnd, &ps);
			RECT rt;
			GetClientRect(hWnd, &rt);
			DrawText(hdc, szCzesc, strlen(szCzesc), &rt, DT_CENTER |
			DT_VCENTER| DT_SINGLELINE);
			EndPaint(hWnd, &ps);
			break;

		default:
			return DefWindowProc(hWnd, message, wParam, lParam);
	}
  return 0;
}
```

Komunikaty związane z myszą

WM_LBUTTONDOWN	wciśnięto lewy (ang. *left*) przycisk
WM_LBUTTONUP	puszczono lewy przycisk
WM_LBUTTONDBLCLK	podwójne kliknięcie l. przyciskiem

Dla prawego przycisku	**Dla środkowego przycisku**
WM_RBUTTONUP	**WM_MBUTTONUP**
WM_RBUTTONDOWN	**WM_MBUTTONDOWN**
WM_RBUTTONDBLCLK	**WM_MBUTTONDBLCLK**

WM_MOUSEMOVE – wysyłany podczas ruchu kursora myszy.

Parametry komunikatów (zakodowane przez **wParam**, **lParam**)

```
DWORD fwKeys = wParam          //   stan przycisków myszy i klawiszy
WORD xPos = LOWORD(lParam)     //   współrzędna x pozycji kursora
WORD yPos = HIWORD(lParam)     //   współrzędna y pozycji kursora
```

LOWORD, HIWORD – zwracają odpowiednio młodsze i starsze słowo (16 bitów) liczby 32-bitowej

fwKeys – kombinacja (logiczne OR) następujących flag

```
MK_CONTROL   - wciśnięty klawisz ctrl
MK_SHIFT     - wciśnięty klawisz shift
MK_LBUTTON   - wciśnięty lewy przycisk
MK_RBUTTON   - wciśnięty prawy przycisk
MK_MBUTTON   - wciśnięty środkowy przycisk
```

Krótko o obsłudze menu – c.d.

W procedurze okna **fragment kodu**:

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM
                                        wParam, LPARAM lParam)
{
int wmId;
switch (messge)
{
        case WM_COMMAND:
                wmId=LOWORD(wParam);
                switch(wmId)
                {
                        case IDM_EXIT:
                              // obsługa polecenia Exit
                               break;

                        case IDM_ABOUT:
                                // obsługa polecenia About
                                break;
                        // Uwaga: niebezpieczeństwo
                        default: return DefWindowProc(hWnd, message, wParam,
                                                        lParam);
                }
                break;
        // *********Tutaj kod obsługi kolejnych komunikatów
        default:  return DefWindowProc(hWnd, message, wParam, lParam);
}
return 0;
```