Procedura okna:

```
LRESULT CALLBACK WndProc(HWND hwnd, UINT message,
                         WPARAM wParam, LPARAM lParam);
```

UINT	– unsigned int
DWORD	– 32-bitowy unsigned int.
LPSTR	- wskaźnik do łańcuch znaków tzn. char *
LPCTSTR	– wskaźnik na stały łańcuch znaków tzn. const char*
LPVOID	- 32-bitowy wskaźnik na wartość o nieokreślonym typie.
LPARM	- 32-bitowa wartość przesyłana jako parametr funkcjom typu CALLBACK.
WPARAM	– Wartość przesyłana jako parametr funkcjom typu CALLBACK 16-bitowa w Windows w wersji 3.0 i 3.1; 32-bitowa w Win32.
HINSTANCE	– uchwyt wystąpienia – jednoznacznie identyfikuje program.
HWND	– Uchwyt okna
CALLBACK, WINAPI	– wskazują na specjalną sekwencję wywołań dla funkcji, jaka występuje między Windows i aplikacją.

```
int WINAPI WinMain(HINSTANCE hInst,HINSTANCE hPrevInst,
                   LPSTR cmdParam, int cmdShow);
```

- hInst - uchwyt wystąpienia - jednoznacznie identyfikujący program
- hPrevInstance – poprzednie wystąpienie. W wersjach Windows przed Windows 95 zwracał uchwyt ostatnio uruchomionego wystąpienia tego samego programu, który nadal działał w systemie. W Windows 95 zawsze NULL, jeśli zachodzi potrzeba sprawdzania czy program jest już uruchomiony to należy użyć funkcji `CreateMutex`.
- cmdParm - wskaźnik do ciągu znaków zakończonego zerem, zawierającego parametry przekazane do programu w chwili jego uruchomienia.
- cmdShow - określa sposób w jaki okno pojawi się po raz pierwszy na pulpicie Windows (SW_SHOWNORMAL, SW_SHOWMINNOACTIVE)

W typowej funkcji WinMain rejestruje się klasę okna (RegisterClassEx), tworzy się okno (CreateWindow), pokazuje się okno (ShowWindow) oraz tworzy się blok kodu zwany „pętlą komunikatów”.

```
ATOM RegisterClassEx(CONST WNDCLASSEX *lpwcx);
```

lpwcx - wskaźnik na strukturę WNDCLASSEX

WNDCLASSEX:

- `cbSize` - zwykle odpowiada: `sizeof(WNDCLASSEX)`
- `style` - style klasy składane przy pomocy operatora „or” np. CS_HREDRAW | CS_VREDRAW
- `lpfnWndProc` - daleki wskaźnik na procedurę obsługi okna (podajemy nazwę wcześniej zadeklarowanej procedury okna) np. (WNDPROC)WndProc
- `cbClsExtra` - rozmiar dodatkowego obszaru pamięci w strukturze klasy. Program może z niego korzystać na własny użytek. (zwykle 0)
- `cbWndExtra` - rozmiar dodatkowego obszaru pamięci w strukturze okna Program może z niego korzystać na własny użytek. (zwykle 0)
- `hInstance` - wystąpienie programu (hInstance z parametru wywołania WinMain)
- `hIcon` –wyznacza uchwyt ikony (HICON) dla wszystkich okien opartych na tej klasie. Można skorzystać z jednej z ikon zdefiniowanych w Windows lub utworzyć własną ikonę dla swojego programu. W celu uzyskania uchwytu ikony korzystamy z funkcji LoadIcon, której pierwszy parametr równa się NULL dla predefiniowanej ikony i hInstance – uchwyt wystąpienia programu - dla zaprojektowanej przez siebie ikony. Drugi parametr jest identyfikatorem ikony.(np. LoadIcon(hInstance, (LPCTSTR)IDI_FIRST).
- `hCursor` - postać kursora myszy jaką przyjmie wskaźnik myszy umieszczony nad oknem klasy (np. LoadCursor(NULL, IDC_ARROW)
- `hbrBackground` - kolor tła obszaru roboczego okna. Przypisujemy uchwyt pędzla (HBRUSH). Pędzel rozumiemy jako barwny wzorzec stosowany do wypełnienia dowolnego obszaru. Windows ma wiele standardowych pędzli.
  (np. `(HBRUSH)COLOR_WINDOW` lub
  `(HBRUSH)GetStockObject(WHITE_BRUSH)`
  `GetStockObject` – pobiera obiekt graficzny i zwraca uchwyt do niego. Tutaj zwraca uchwyt pędzla białego)
- `lpszMenuName` - nazwa menu okna ze skryptu zasobów np. (LPCSTR)IDC_FIRST
- `lpszClassName` - nazwa klasy np. *szNazwaKlasy*
- `hIconSm` - mała ikona wszystkich okien opartych na klasie (np. LoadIcon(hInstance,(LPCTSTR)IDI_SMALL)

Class Style CS_ (WNDCLASSEX)

- CS_BYTEALIGNCLIENT - pozioma współrzędna początku pola roboczego okna jest wielokrotnością liczby 8
- CS_BYTEALIGNWINDOW - pozioma współrzędna okna jest wielokrotnością liczby 8 (optymalizacja operacji na mapach bitowych)
- CS_CLASSDC - jeden kontekst urządzenia dla wszystkich okien klasy
- CS_DBLCLKS - wysyła komunikat o podwójnym kliknięciu do procedury okna gdy użytkownik kliknął podwójnie w obszarze okna.
- CS_GLOBALCLASS – ustawia, że klasa okna jest globalną klasą aplikacji tzn. jest dostępna dla wszystkich modułów programu.
- CS_HREDRAW - unieważnia cały obszar roboczy okna podczas zmiany szerokości okna, normalnie unieważniony jest tylko obszar zmiany (dzięki temu cały obszar roboczy będzie odświeżany przy każdorazowej zmianie szerokości okna)
- CS_NOCLOSE - wyłącza komendę „Zamknij” w menu systemowym okna.
- CS_OWNDC - alokuje osobny kontekst urządzenia dla każdego okna klasy, co zostanie zmienione w kontekście pozostaje zmienione do czasu jawnego odwołania np. pędzel, pióro, tryb odwzorowania. Styl ten wpływa jedynie na kontekst urządzenia otrzymany w wyniku wywołania GetDC oraz BeginPaint
- CS_PARENTDC –ustawia kontekst urządzenia dla okna typu *child* w oknie jego rodzica, aby okno *child* mogło rysować w oknie rodzica. Okno z ustawionym CS_PARENTDC otrzymuje kontekst urządzenie z systemowej pamięci kontekstów urządzeń. CHILD nie otrzymuje kontekstu urządzenia rodzica lub jego ustawień. (Pozbawia okna w stylu WS_CHILD własnego kontekstu urządzenia.)
- CS_SAVEBITS - powoduje, że system próbuje przed wyświetleniem okna zapamiętać poprzednią zawartość tego obszaru w postaci mapy bitowej. W chwili usunięcia okna następuje próba odtworzenia okna z zapamiętanej mapy nie wysyłając do okna komunikatu WM_PAINT.
- CS_VREDRAW - unieważnia cały obszar roboczy okna podczas zmiany wysokości okna, normalnie unieważniony jest tylko obszar zmiany.

```
HWND CreateWindow(LPCTSTR lpClassName, LPCTSTR lpWindowName,
                  DWORD dwStyle,int x, int y, int nWidth,
                  int nHeight, HWND hWndParent, HMENU hMenu,
                  HANDLE hInstance, LPVOID lpParam);
```

- `lpClassName` - nazwa klasy okna np. *szNazwaKlasy* lub predefiniowane: `BUTTON`, `COMBOBOX`, `EDIT`, `LISTBOX`, `SCROLLBAR`, `MDICLIENT`, `STATIC`. Przy pomocy tej nazwy wiążemy okno z klasą okna.
- `lpWindowName` - nagłówek okna np. ”Pierwszy program”
- `dwStyle` - styl okna np. `WS_OVERLAPPEDWINDOW`
- `x` – początkowe położenie lewego górnego rogu okna na osi x względem lewego górnego rogu ekranu. Używając CW_USEDEFAULT pozwalamy Windows na użycie swoich domyślnych ustawień.
- `y` – początkowe położenie lewego górnego rogu okna na osi y względem lewego górnego rogu ekranu
- `nWidth` - początkowy rozmiar okna wzdłuż osi x (można użyć CW_USEDEFAULT)
- `nHeight` - początkowy rozmiar okna wzdłuż osi y
- `hWndParent` - uchwyt okna nadrzędnego, NULL - brak okna nadrzędnego
- `hMenu` - uchwyt menu okna, może być NULL jeśli w definicji klasy okna zostało podane lpszMenuName. Jeżeli okno jest typu WS_CHILD *hMenu* specyfikuje identyfikator okna – wartość liczbową – używany przez okno do informowania jego rodzica o wystąpieniu zdarzenia. Aplikacja wyznacza identyfikator okna typu WS_CHILD, który musi być unikalny dla okien tego samego rodzica.
- `hInstance` - uchwyt wystąpienia programu
- `lpParam` - dodatkowe parametry inicjalizacji można nimi przekazać pewne dane oknu, z których można skorzystać w trakcie pracy programu, NULL - brak parametrów

Funkcja *CreateWindow* zwraca uchwyt do tworzonego okna.

WindowStyle WS_ (CreateWindow)

- WS_BORDER - okno z cienką ramką
- WS_CAPTION - jak WS_BORDER i dodatkowo pasek tytułowy okna
- WS_CHILD-okno potomne(nie może mieć paska menu, nie można używać z WS_POPUP)
- WS_CHILDWINDOW = WS_CHILD
- WS_CLIPCHILDREN - z obszaru unieważnionego (do odświeżenia) zostają usunięte obszary zakryte przez okna potomne (styl używany przy tworzeniu okien rodziców).
- WS_CLIPSIBLINGS – Spina okna potomne powiązane ze sobą. Kiedy konkretne okno otrzymuje WM_PAINT ten styl powoduje spięcie nachodzących na siebie okien potomnych i odświeżanie tylko obszarów nie zakrytych okna. Jeżeli ten styl nie jest wyspecyfikowany i okna potomne nachodzą na siebie, jest możliwe, że kiedy następuje rysowanie w obszarze roboczym jednego okna przemalowujemy obszar okna zakrywającego.
- WS_DISABLED - tworzy okno nieaktywne, tj. takie, które nie może otrzymywać komunikatów od użytkownika. Po utworzeniu okna jego włączanie i wyłączanie odbywa się za pomocą funkcji EnableWindow.
- WS_DLGFRAME - tworzy okno o ramce jak typowe okno dialogowe, Okno w tym stylu nie może posiadać paska tytułowego.
- WS_GROUP - wyznacza pierwsze okno z grupy okien. Grupa składa się z tego pierwszego okna i wszystkich następnych które są tworzone aż do następnego okna w stylu WS_GROUP. Zwykle dodawany jest styl WS_TABSTOP, aby użytkownik mógł przemieszczać się z jednej grupy do drugiej.
- WS_HSCROLL - tworzy okno z poziomym paskiem przewijania.
- WS_ICONIC - okno na starcie zminimalizowane, styl tożsamy z WS_MINIMIZE.
- WS_MAXIMIZE - okno na starcie rozwinięte na cały ekran.
- WS_MAXIMIZEBOX - okno wyposażone w przycisk rozwijania na cały ekran. Nie można łączyć z WS_EX_CONTEXTHELP. Musi być połączone z WS_SYSMENU.
- WS_MINIMIZE = WS_ICONIC.
- WS_MINIMIZEBOX - okno wyposażone w przycisk zwijania okna do ikony. Nie można łączyć z WS_EX_CONTEXTHELP. Musi być połączone ze stylem WS_SYSMENU.
- WS_OVERLAPPED = WS_TILED - okienko wyposażone w pasek tytułowy i ramkę.
- WS_OVERLAPPEDWINDOW - okienko w stylu WS_OVERLAPPED, WS_CAPTION, WS_SYSMENU, WS_THICKFRAME, WS_MINIMIZEBOX i WS_MAXIMIZEBOX. Tożsame z WS_TILEDWINDOW.
- WS_POPUP - nie można używać z WS_CHILD.
- WS_POPUPWINDOW = WS_BORDER, WS_POPUP i WS_SYSMENU.
  Styl WS_CAPTION musi być dodany żeby menu było widoczne.
- WS_SIZEBOX=WS_THICKFRAME - z ramką umożliwiającą zmianę wymiarów okna.
- WS_SYSMENU - dodaje do okna menu systemowe. WS_CAPTION musi być dodany.
- WS_TABSTOP - określa czy okno otrzymuje sterowanie (focus) po naciśnięciu klawisza TAB
- WS_THICKFRAME = WS_SIZEBOX.
- WS_TILED = WS_OVERLAPPED.
- WS_TILEDWINDOW = WS_OVERLAPPEDWINDOW.
- WS_VISIBLE - okno widoczne w momencie inicjalizacji. Ten styl może być włączany i wyłączany za pomocą funkcji ShowWndow i SetWindowPos
- WS_VSCROLL - okno wyposażone w pasek przewijania pionowego.

BOOL **ShowWindow**(HWND hWnd, int cmdShow)

- `hWnd` – uchwyt do utworzonego okna
- `cmdShow` – wartość przekazana w parametrach funkcji WinMain.

Pętla komunikatów:

```
while (GetMessage(&msg, NULL, 0, 0))
{
TranslateMessage(&msg);   // przekazanie komunikatu do systemu, w celu
                          // przekształcenia niektórych komunikatów wysłanych z
                          // klawiatury
DispatchMessage(&msg);
}
```

funkcja pobierająca komunikat z kolejki

```
BOOL GetMessage(LPMSG lpMsg, HWND hWnd, UINT wMsgFilterMin,
                UINT wMsgFilterMax)
```

`lpMsg` - adres na strukturę przechowującą komunikat (MSG)
`hWnd` - uchwyt okna dla którego nastąpiło zdarzenie. NULL oznacza, że funkcja pobiera komunikaty dla wszystkich okien aplikacji.
`wMsgFilterMin` – liczba całkowita określająca identyfikator pierwszego komunikatu, który ma być pobrany
`wMsgFilterMax` - ostatni komunikat
Jeżeli chcemy aby funkcja pobierała tylko zdarzenia związane z klawiaturą należy filtr ustawić na –WM_KEYFIRST I WM_KEYLAST, z myszą – WM_MOUSEFIRST, WM_MUSELAST. Jeżeli oba parametry mają wartość 0 funkcja pobiera wszystkie dostępne komunikaty.

GetMessage zwraca wartość różną od zera, z wyjątkiem przypadku gdy pole message równa się `WM_QUIT`. W przypadku błędu funkcja zwraca `-1`.

Definicja struktury MSG, przechowującej informacje na temat komunikatu, ma następującą postać.

```
typedef struct tagMSG
{
HWND hwnd;      // uchwyt okna adresata komunikatu
UINT message;  // identyfikator komunikatu. Każdy komunikat ma własny identyfikator.
WPARAM wParam;// 32 bitowy parametr komunikatu, którego znaczenie i wartość zależy
                // od samego komunikatu
LPARAM lParam;// dodatkowy 32 bitowy parametr komunikatu
DWORD time;     // czas określający moment umieszczenia komunikatu w kolejce
POINT pt;       // współrzędne myszy w chwili gdy komunikat został umieszczony w
                // kolejce. Struktura POINT składa się z dwóch pól: long x, long y.
}MSG;
```

Funkcja przekazująca komunikat do procedury okna:

```
LONG DispatchMessage(CONST MSG *lpmsg)
```

WM_DESTROY – komunikat ten oznacza, że Windows jest w trakcie usuwania okna. Komunikat jest wynikiem kliknięcia przycisku *Zamknij* lub klawiszy [ALT+F4]. Standardowa reakcja to zakończenie działania aplikacji po przez wysłanie komunikatu WM_QUIT. Jeżeli nie obsłużymy tego komunikatu Windows zamknie okno, ale aplikacja nie zakończy działania.

funkcja umieszczająca w kolejce komunikatów komunikat WM_QUIT (wraca bezpośrednio)
`VOID` **`PostQuitMessage`**`(int nExitCode)`
`nExitCode` - kod zakończenia działania aplikacji. Tą wartości przechowuje parametr wParam komunikatu WM_QUIT.

typowe użycie w procedurze okna:

```
case WM_DESTROY:
  PostQuitMessage(0);
  break;
```

Pętla komunikatów i procedura okna nie działają współbieżnie. Powrót z DispatchMessage nie nastąpi dopóki nie zakończy się przetwarzanie komunikatu w procedurze okna. Procedura okna może uruchamiać się rekurencyjnie przez wysyłanie komunikatów.

Wszystkie komunikaty, którymi procedura okna się nie zajmuje **muszą** być przekazane do funkcji *DefWindowProc*. Wartość zwracana przez DefWindowProc musi być jednocześnie wartością zwracaną przez procedurę okna.

```
LRESULT DefWindowProc(  HWND hWnd      // uchwyt okna
                        UINT msg       // identyfikator komunikatu
                        WPARM wParm    // pierwszy parametr komunikatu
                        LPARM lParm    //drugi parametr komunikatu
                        );
```

Podstawowe rodzaje okien:

Overlapped – okno najwyższego poziomu. Posiada pasek tytułowy, ramkę, obszar roboczy. Może służyć jako główne okno aplikacji. Może posiadać menu, przycisk minimalizacji i maksymalizacji oraz paski przewijania.
Pop-up – Specjalny rodzaj okna typu overlapped, używane do tworzenia okienek dialogowych, messagebox-ów i innych tymczasowych okienek używanych poza głównym oknem aplikacji. Pasek tytułowy jest opcjonalny dla okien typu *pop-up.* W przeciwnym wypadku okno typu pop-up jest takie same jak okno typu overlapped z stylem: WS_OVERLAPPED.
Child – okno potomne posiada styl: WS_CHILD i jest ograniczone ramami obszaru swojego rodzica. Okno typu CHILD musi posiadać rodzica. Rodzicem może być okno typu overlapped, pop-up lub nawet typu child. Okno tego typu posiada tylko obszar roboczy, chyba że zapotrzebowanie na inne właściwości okna będzie bezpośrednio zgłoszone. Okno typu CHILD nie może mieć menu. Pozycja okno tego rodzaju jest ustawiana względem górnego lewego rogu okna rodzica. Żadna z części okna typu child nie jest wyświetlana poza ramami okna jego rodzica. Niektóre akcje które dotyczą rodzica mogą mieć wpływ na jego potomstwo:
Zamykanie – okno potomne jest zamykane zanim rodzic zostanie zamknięty
Ukrywanie – okno potomne jest ukrywane przed rodzicem. Okno potomne jest widoczne tylko wtedy gdy jego rodzic jest widoczny
Przesuwanie – okno potomne jest przesuwane razem z rodzicem Okno typu child jest odpowiedzialne za odrysowanie swojego obszaru roboczego po przesunięciu.
Wyświetlanie - okno potomne ukazuje się po wyświetleniu jego rodzica.
System wysyła komunikaty bezpośrednio do okna potomnego. Jedyny wyjątek stanowi przypadek gdy okno jest nieaktywne, wtedy komunikat jest wysyłany do jego rodzica. Okno potomne wysyła komunikat o zajściu zdarzenia do okna nadrzędnego.
Owned – Okno typu overlapped lub pop-up może być własnością innego okna overlapped lub pop-up.
Okno typu owned jest zawsze przed oknem właściciela. Okno jest zawsze zamykane kiedy okno właściciela jest zamykane. Okno jest ukryte jeżeli okno właściciel jest zminimalizowane. Okno tego typu tworzymy wpisując uchwyt właściciela w parametrze hWndParent w funkcji CrateWindowEx.