**1. Inclusion de l'en-tête Windows**

c

Copy

#include <windows.h>

* **Explication** :
  + windows.h est l'en-tête principal pour l'API Windows. Il contient les déclarations de fonctions, types et constantes nécessaires pour créer des applications Windows.

**2. Déclaration de la fonction WndProc**

c

Copy

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

* **Explication** :
  + WndProc est la fonction de rappel (callback) qui gère les messages envoyés à la fenêtre (comme les clics de souris, les touches du clavier, etc.).
  + Elle est déclarée ici pour pouvoir être utilisée dans WinMain avant sa définition complète.

**3. Fonction WinMain**

c

Copy

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {

* **Explication** :
  + WinMain est le point d'entrée de l'application Windows.
  + **Paramètres** :
    - hInstance : Handle de l'instance actuelle de l'application.
    - hPrevInstance : Toujours NULL dans les versions modernes de Windows.
    - lpCmdLine : Arguments de la ligne de commande.
    - nCmdShow : Mode d'affichage de la fenêtre (par exemple, maximisée, minimisée, etc.).

**4. Initialisation de la classe de la fenêtre**

c

Copy

WNDCLASSEX wc = { sizeof(WNDCLASSEX) };

wc.lpfnWndProc = WndProc;

wc.hInstance = hInstance;

wc.lpszClassName = TEXT("MaClasse");

RegisterClassEx(&wc);

* **Explication** :
  + WNDCLASSEX est une structure qui définit les propriétés de la classe de la fenêtre.
  + **Champs** :
    - cbSize : Taille de la structure (initialisée avec sizeof(WNDCLASSEX)).
    - lpfnWndProc : Pointeur vers la fonction WndProc pour gérer les messages.
    - hInstance : Instance de l'application.
    - lpszClassName : Nom de la classe de la fenêtre (ici, "MaClasse").
  + RegisterClassEx enregistre cette classe pour qu'elle puisse être utilisée pour créer des fenêtres.

**5. Création de la fenêtre**

c

Copy

HWND hwnd = CreateWindowEx(0, TEXT("MaClasse"), TEXT("Ma Fenêtre"), WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, CW\_USEDEFAULT, 500, 400, NULL, NULL, hInstance, NULL);

ShowWindow(hwnd, nCmdShow);

* **Explication** :
  + CreateWindowEx crée la fenêtre.
    - **Paramètres** :
      * 0 : Style étendu (aucun style supplémentaire ici).
      * "MaClasse" : Nom de la classe de la fenêtre (défini précédemment).
      * "Ma Fenêtre" : Titre de la fenêtre.
      * WS\_OVERLAPPEDWINDOW : Style de la fenêtre (avec bordure, titre, boutons de fermeture, etc.).
      * CW\_USEDEFAULT, CW\_USEDEFAULT : Position par défaut de la fenêtre.
      * 500, 400 : Largeur et hauteur de la fenêtre.
      * NULL, NULL : Pas de parent ni de menu.
      * hInstance : Instance de l'application.
      * NULL : Pas de données supplémentaires.
  + ShowWindow affiche la fenêtre à l'écran.
  + UpdateWindow met à jour la fenêtre pour s'assurer qu'elle est correctement dessinée.

**6. Boucle de messages**

c

Copy

MSG msg;

while (GetMessage(&msg, NULL, 0, 0)) {

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return msg.wParam;

* **Explication** :
  + La boucle de messages est le cœur de l'application Windows. Elle récupère les messages (comme les clics de souris ou les touches du clavier) et les envoie à la fonction WndProc pour traitement.
  + **Fonctions** :
    - GetMessage : Récupère un message de la file d'attente des messages.
    - TranslateMessage : Traduit les messages de touches en caractères.
    - DispatchMessage : Envoie le message à la fonction WndProc.
  + La boucle se termine lorsque GetMessage retourne 0, ce qui se produit lorsque le message WM\_QUIT est reçu.

**7. Fonction WndProc**

c

Copy

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

switch (msg) {

case WM\_CLOSE:

DestroyWindow(hwnd);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hwnd, msg, wParam, lParam);

}

return 0;

}

* **Explication** :
  + WndProc est la fonction de rappel qui gère les messages envoyés à la fenêtre.
  + **Messages traités** :
    - WM\_CLOSE : Reçu lorsque l'utilisateur clique sur le bouton "Fermer". La fenêtre est détruite avec DestroyWindow.
    - WM\_DESTROY : Reçu lorsque la fenêtre est détruite. PostQuitMessage est appelé pour quitter l'application.
  + Pour tous les autres messages, la fonction DefWindowProc est appelée pour fournir un traitement par défaut.

**Résumé**

* **WinMain** : Initialise l'application, crée la fenêtre, et entre dans la boucle de messages.
* **WndProc** : Gère les messages de la fenêtre (comme la fermeture de la fenêtre).
* **Boucle de messages** : Récupère et traite les messages pour interagir avec l'utilisateur.

Ce code crée une fenêtre Windows simple avec un titre "Ma Fenêtre". Lorsque l'utilisateur clique sur le bouton "Fermer", la fenêtre est détruite et l'application se termine.