Windows API

The communication between a Windows application and Windows itself uses the Windows Application Programming Interface, otherwise known as the Windows API. This consists of literally hundreds of functions that are provided with Windows as standard, to be used by your applications.

The WinMain() function is the equivalent of the main() function in a console program. It’s where execution starts and ends, and where the basic initialization for the rest of the program is carried out.

// SimApp.cpp : Defines the entry point for the application.

#include <windows.h>
#include <tchar.h>

// Function Prototype for WindowProc.
LRESULT CALLBACK WindowProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);

// *********************************
// Begining of the WinMain function.
int APIENTRY WinMain(HINSTANCE hInstance,
                     HINSTANCE hPrevInstance,
                     LPSTR     lpCmdLine,
                     int       nCmdShow)
{
// Could place C++ code here.

// Structure to hold your window's attributes.
WNDCLASSEX wc = {0};

// Error checking with the actual size of the structure.
wce.cbSize = sizeof(WNDCLASSEX);

// Redraw the window if the horizontal or vertical size changes.
wce.style = CS_HREDRAW | CS_VREDRAW;

// Define your program for message handling.
wce.lpfnWndProc = WindowProc;
wc.cbClsExtra = 0;
wce.cbWndExtra = 0;

// Handle to the application.
wc.hInstance = hInstance;

// Choose an application icon (default).
wc.hIcon = LoadIcon(NULL, IDI_APPLICATION);

// Choose the window cursor (standard arrow).
wc.hCursor = LoadCursor(NULL, IDC_ARROW);

// The brush defining the background color.
// A brush is an object used to fill an area (the window).
// Try changing to -5,-4,...-1,0,etc.
wc.hbrBackground = reinterpret_cast<HBRUSH>(COLOR_WINDOW + 1);

// The menu resource name (no menu used).
wc.lpszMenuName = NULL;

// Define the window class name.
static TCHAR szAppName[] = _T("SimpleApp");
wc.lpszClassName = szAppName;

// Choose a small application icon (default).
wc.hIconSm = LoadIcon(NULL, IDI_APPLICATION);

// Now tell the operating system about our window.
RegisterClassEx(&wc);

// Now create the window using the following attributes.
HWND hWnd = CreateWindowEx(
    WS_EX_CLIENTEDGE,
    szAppName,
    _T("Engineering Computing Methods"),
    WS_OVERLAPPEDWINDOW,
    CW_USEDEFAULT,
    CW_USEDEFAULT,
    CW_USEDEFAULT,
    CW_USEDEFAULT,
    NULL,
    NULL,
    hInstance,
    NULL
);

// Display the window without any of our program content.
ShowWindow(hWnd, nCmdShow);

// Redraw the window should the window be resized or moved.
UpdateWindow(hWnd);
// msg is a variable of a Windows message structure.
MSG msg;

// Get any messages pending, if there are none some other
// Window's application gets to run.
while (GetMessage(&msg,NULL,0,0) != 0)
{
    TranslateMessage(&msg);
    DispatchMessage(&msg);
}

return msg.wParam;

// ********************
// Function WindowProc.
// This is where your program code goes.
LRESULT CALLBACK WindowProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    switch(message)
    {
    // Code to deal with drawing the window area.
    case WM_PAINT:
    {
        PAINTSTRUCT paintst; // structure defining the area to be redrawn.

        // Request permission from Windows to redraw the window.
        // When we receive a handle to a display context (HDC) when continue.
        HDC hDC = BeginPaint(hWnd,&paintst);

        RECT rcClient; // work with a rectangle.

        // get the rectangle of our window.
        // (i.e. get the upper-left & lower-right coordinates of the window.)
        GetClientRect(hWnd, &rcClient);

        // Set the background color of the text to match the window background color.
        int nOldBkMode = SetBkMode(hDC, TRANSPARENT);

        // Set the text color and store the old text color for future use.
        COLORREF clrOldTextColor = SetTextColor(hDC, GetSysColor(COLOR_WINDOWTEXT));
// Now draw the text in the window.
DrawText(
    hDC,
    _T("What a great class!!!"),
    -1,
    &rcClient,
    DT_SINGLELINE | 
    DT_CENTER | 
    DT_VCENTER

);

// Restore the display context to it previous state.
// (i.e. replace the text color.)
SetTextColor(hDC, clrOldTextColor);

// and replace the text background color.
SetBkMode(hDC, nOldBkMode);

// Let Windows know that we are finished drawing in our window.
EndPaint(hWnd,&paintst);
return 0;
}

// Code to deal with the closing of the window.
case WM_DESTROY:
    PostQuitMessage(0);
    return 0;

// Code to handle any other messages that have not been dealt with.
// (i.e. send these extra messages back to Windows.)
default:
    return DefWindowProc(hWnd, message, wParam, lParam);
}