**Application Programs using Windows API**

I am Alvin Alexander, eighth standard student of Salvation Army HSS, Kowdiar, Thiruvananthapuram.

In the previous session, we learned about GUI and about how we will create a window. This session, we will be focusing on a window procedure that is needed for processing certain window messages sent by the system, like WM\_PAINT for painting our window. Without the window procedure, we cannot register our window class for creating a window.

This is a basic window procedure that handles basic messages and passes the rest to be processed by DefWindowProc(…) function:

LRESULT CALLBACK WindowProc(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam) {

switch (uMsg) {

case WM\_DESTROY:

PostQuitMessage(0);

return 0;

case WM\_PAINT:

PAINTSTRUCT ps;

ZeroMemory(&ps, sizeof(ps));

HDC hdc = BeginPaint(hWnd, &ps);

// All painting occurs between BeginPaint and EndPaint.

FillRect(hdc, &ps.rcPaint, (HBRUSH)(COLOR\_WINDOW + 1));

EndPaint(hWnd, &ps);

return 0;

case WM\_CLOSE:

DestroyWindow(hWnd);

return 0;

default:

return DefWindowProc(hWnd, uMsg, wParam, lParam);

}

}

Here, we handle three basic window messages: WM\_PAINT, WM\_DESTROY, and WM\_CLOSE. We can paint things into the window by handling WM\_PAINT. This will be useful later for painting GUI controls to the window. Currently, we just paint a white background for the window. WM\_CLOSE is sent to the window when the user closes the window. Here we destroy the window, which sends WM\_DESTROY. In WM\_DESTROY message handling, we sent WM\_QUIT, which stops the message loop. The program ends when the message loop is broken.

In the next session, we will delve into the message loop of a window, which passes messages to the window procedure to be subsequently handled.