#include <windows.h>

/\* Declare Windows procedure \*/

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

/\* Make the class name into a global variable \*/

char szClassName[ ] = "WindowsApp";

int WINAPI WinMain (HINSTANCE hThisInstance,

HINSTANCE hPrevInstance,

LPSTR lpszArgument,

int nFunsterStil)

{

HWND hwnd; /\* This is the handle for our window \*/

MSG messages; /\* Here messages to the application are saved \*/

WNDCLASSEX wincl; /\* Data structure for the windowclass \*/

/\* The Window structure \*/

wincl.hInstance = hThisInstance;

wincl.lpszClassName = szClassName;

wincl.lpfnWndProc = WindowProcedure; /\* This function is called by windows \*/

wincl.style = CS\_DBLCLKS; /\* Catch double-clicks \*/

wincl.cbSize = sizeof (WNDCLASSEX);

/\* Use default icon and mouse-pointer \*/

wincl.hIcon = LoadIcon (NULL, IDI\_APPLICATION);

wincl.hIconSm = LoadIcon (NULL, IDI\_APPLICATION);

wincl.hCursor = LoadCursor (NULL, IDC\_ARROW);

wincl.lpszMenuName = NULL; /\* No menu \*/

wincl.cbClsExtra = 0; /\* No extra bytes after the window class \*/

wincl.cbWndExtra = 0; /\* structure or the window instance \*/

/\* Use Windows's default color as the background of the window \*/

wincl.hbrBackground = (HBRUSH) COLOR\_BACKGROUND;

/\* Register the window class, and if it fails quit the program \*/

if (!RegisterClassEx (&wincl))

return 0;

/\* The class is registered, let's create the program\*/

hwnd = CreateWindowEx (

0, /\* Extended possibilites for variation \*/

szClassName, /\* Classname \*/

"Windows App", /\* Title Text \*/

WS\_OVERLAPPEDWINDOW, /\* default window \*/

CW\_USEDEFAULT, /\* Windows decides the position \*/

CW\_USEDEFAULT, /\* where the window ends up on the screen \*/

544, /\* The programs width \*/

375, /\* and height in pixels \*/

HWND\_DESKTOP, /\* The window is a child-window to desktop \*/

NULL, /\* No menu \*/

hThisInstance, /\* Program Instance handler \*/

NULL /\* No Window Creation data \*/

);

/\* Make the window visible on the screen \*/

ShowWindow (hwnd, nFunsterStil);

/\* Run the message loop. It will run until GetMessage() returns 0 \*/

while (GetMessage (&messages, NULL, 0, 0))

{

/\* Translate virtual-key messages into character messages \*/

TranslateMessage(&messages);

/\* Send message to WindowProcedure \*/

DispatchMessage(&messages);

}

/\* The program return-value is 0 - The value that PostQuitMessage() gave \*/

return messages.wParam;

}

/\* This function is called by the Windows function DispatchMessage() \*/

LRESULT CALLBACK WindowProcedure (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message) /\* handle the messages \*/

{

case WM\_DESTROY:

PostQuitMessage (0); /\* send a WM\_QUIT to the message queue \*/

break;

default: /\* for messages that we don't deal with \*/

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

}

return 0;

}