#include<windows.h>

typedef struct \_WNDCLASS {

UINT style;// 窗口类型

WNDPROC lpfnWndProc;//窗口处理函数

int cbClsExtra;//窗口扩展

int cbWndExtra;//窗口实例扩展

HINSTANCE hInstance;//实例句柄

HICON hIcon;//窗口的最小化图标

HCURSOR hCursor;//窗口鼠标光标

HBRUSH hbrBackground;//窗口背景色

LPCTSTR lpszMenuName;//窗口菜单

LPCTSTR lpszClassName;// 窗口类名

} WNDCLASS, \*LPWNDCLASS;

HWND CreateWindowEx(

DWORD DdwExStyle, //窗口的扩展风格

LPCTSTR lpClassName, //指向注册类名的指针

LPCTSTR lpWindowName, //指向窗口名称的指针

DWORD dwStyle, //窗口风格

int x, //窗口的水平位置

int y, //窗口的垂直位置

int nWidth, //窗口的宽度

int nHeight, //窗口的高度

HWND hWndParent, //父窗口的句柄

HMENU hMenu, //菜单的句柄或是子窗口的标识符

HINSTANCE hInstance, //应用程序实例的句柄

LPVOID lpParam //指向窗口的创建数据

);

typedef struct tagMSG {

HWND hwnd;

UINT message;

WPARAM wParam;

LPARAM lParam;

DWORD time;

POINT pt;

} MSG;

/\*GetMessage（

LPMSG lpMsg,//指向MSG结构的指针，该结构从线程的消息队列里接收消息信息

HWND hWnd,//取得其消息的窗口的句柄。当其值取NULL时，GetMessage为任何属于调用线程的窗口检索消息，线程消息通过PostThreadMessage寄送给调用线程。

UINT wMsgFilterMin,//指定被检索的最小消息值的整数。可省略

UINT wMsgFilterMax//指定被检索的最大消息值的整数。可省略

)

LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam);

HINSTANCE hinhwnd = 0;

HWND editor = 0;\*/

int WINAPI wWinMain(HINSTANCE hInstance, HINSTANCE, PWSTR pCmdLine, int nCmdeShow)

{

const wchar\_t CLASS\_NAME[] = L"test";//类名

WNDCLASS wc = {};

wc.style = 0;//缺省

wc.cbClsExtra = 0;//缺省

wc.cbWndExtra = 0;//缺省

wc.hIcon = LoadIcon(NULL, IDI\_APPLICATION);//缺省

wc.hCursor = LoadCursor(NULL, IDC\_ARROW);//箭头光标

wc.hbrBackground = (HBRUSH)(GetStockObject(WHITE\_BRUSH));//白色刷子

wc.lpszMenuName = NULL;//无菜单

wc.lpfnWndProc = WindowProc;//消息处理函数

hinhwnd=wc.hInstance = hInstance;

wc.lpszClassName = CLASS\_NAME;

RegisterClass(&wc);

HWND hwnd = CreateWindowEx(

0,

CLASS\_NAME,

L"test",//标题

WS\_OVERLAPPEDWINDOW,//一般风格

CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT, CW\_USEDEFAULT,//默认位置及尺寸

NULL,

NULL,

hInstance,

NULL

);

if (hwnd == NULL)

return 0;

ShowWindow(hwnd,1);

MSG msg = {};

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

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return 0;

}

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;

HDC hdc = BeginPaint(hwnd, &ps);

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

EndPaint(hwnd, &ps);

}

return 0;

}

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

}