

Лекция 5

Содержание

Архитектура приложений win32 -

- стандартное оконное приложение,
- ресурсы приложения,
- предопределённые стили окон, диалоги.

Шаблон оконного приложения Windows

```
#include <windows.h>
LRESULT CALLBACK MyWndProc(HWND, UINT, WPARAM,
LPARAM);
HDC hdc;
RECT rt;
int WINAPI WinMain(HINSTANCE hInstance,
                   HINSTANCE hPrevInstance,
                   LPSTR lpCmdLine,
                   int nCmdShow){
    HWND hWnd;
    MSG msg;
    WNDCLASS wc;
    LPCSTR lpszAppName="CTemplate1";
    BOOL ret;
```

//ИНИЦИАЛИЗАЦИЯ КЛАССА ОКНА

wc.lpszClassName = lpszAppName; //Имя класса окна

wc.hInstance=hInstance; //дескриптор экземпляра приложения

wc.lpfnWndProc = (WNDPROC)MyWndProc; //указатель
//на процедуру окна

wc.hCursor = LoadCursor(NULL, IDC_ARROW); //вид курсора над
//окном

wc.hIcon = 0;

//LoadIcon(hInstance,(LPCTSTR)IDI_CTEMPLATE); //идентификатор
//пиктограммы

wc.lpszMenuName = 0; //идентификатор ресурса меню

wc.hbrBackground = (HBRUSH)(COLOR_WINDOW+1); //цвет
//закраски окна

wc.style = CS_HREDRAW | CS_VREDRAW; //стиль окна

wc.cbClsExtra = 0; //рудимент, инициализируется нулём

wc.cbWndExtra = 0; //рудимент, инициализируется нулём

if(!RegisterClass(&wc)) //Регистрация класса окна

return 0;

```
hWnd = CreateWindow( lpszAppName, //Имя класса окна
                    lpszAppName, //Имя окна
                    WS_OVERLAPPEDWINDOW, //Стиль
                    //окна (перекрывающееся окно)
                    100, //CW_USEDEFAULT,
                    //х-коорд. верхнего-левого угла
                    100, //у-коорд. верхнего-левого угла
                    400, //CW_USEDEFAULT, ширина
                    200, //высота
                    NULL, //дескриптор родительского окна
                    NULL, //дескриптор меню
                    hInstance, //дескриптор экземпляра приложения
                    NULL); //указатель на структуру, содержащую
//дополнительные параметры окна
```

```
ret=RegisterHotKey(hWnd,0xB001, MOD_CONTROL |
                    MOD_ALT, 'W');
```

```
ShowWindow(hWnd,SW_HIDE); //SW_SHOW...
                        //способ представления окна
UpdateWindow(hWnd);    //прорисовывает клиентскую область окна,
                        //генерирует сообщение WM_PAINT

while (GetMessage(&msg, NULL, 0, 0)) { //Извлечение сообщения
                        //из очереди
                        //сообщений
    TranslateMessage(&msg);    //трансляция сообщений
                                //виртуальных ключей WM_KEYDOWN,
                                //WM_KEYDOWN, WM_KEYUP и т.п.
                                //в сообщение символа WM_CHAR
    DispatchMessage(&msg);    //направляет сообщения оконной процедуре
}

return msg.wParam;
}
```

Процедура окна.

```
LRESULT CALLBACK
```

```
MyWndProc(HWND hWnd, UINT message,  
          WPARAM wParam,  
          LPARAM lParam){
```

```
PAINTSTRUCT ps;
```

```
switch (message){
```

```
case WM_DESTROY: //сообщение генерируется при уничтожении окна  
    PostQuitMessage(0); //указывает системе штатно выполнить  
                        //выход из программы
```

```
    break;
```

```
case WM_HOTKEY:
```

```
    ShowWindow(hWnd,SW_SHOWNORMAL);
```

```
    break;
```

```
case WM_PAINT:
```

```
    hdc = BeginPaint(hWnd, &ps);
```

```
    GetClientRect(hWnd, &rt);
```

```
    DrawText(hdc, "From Paint", strlen("From Paint"), &rt,  
            DT_CENTER);
```

```
    EndPaint(hWnd, &ps);
```

```
    break;
```

```
case WM_COPYDATA:
```

```
    GetClientRect(hWnd, &rt);
```

```
    hdc = GetDC(hWnd);
```

```
    DrawText(hdc, (char*)(
```

```
        ((COPYDATASTRUCT*)lParam)->lpData),
```

```
        ((COPYDATASTRUCT*)lParam)->cbData, &rt,
```

```
        DT_LEFT);
```

```
    ReleaseDC(hWnd, hdc);
```

```
    break;
```

```
default:
```

```
    return DefWindowProc(hWnd, message, wParam, lParam);
```

```
    //Обеспечивается обработка сообщений по умолчанию
```

```
}
```



```
typedef struct {  
    HWND hwnd; //Дескриптор окна-получателя  
    UINT message; //Идентификатор сообщения (WM_...)  
    WPARAM wParam; //Дополнительная информация (зависит  
                    //от сообщения)  
    LPARAM lParam; //Дополнительная информация (зависит от  
                    //сообщения)  
    DWORD time; //Время послыки сообщения  
    POINT pt; //Положение курсора, когда посылалось сообщение  
} MSG, *PMSG;
```

```
typedef struct {  
    ULONG_PTR dwData; //пересылаемые данные  
    DWORD cbData; //размер данных  
    PVOID lpData; // пересылаемые данные (можно  
                  //NULL)  
} COPYDATASTRUCT, *PCOPYDATASTRUCT;
```

Приложение клиент.

```
#include <windows.h>
int main(){
    HWND hWnd;
    COPYDATASTRUCT data;
    char str[80];
    hWnd=FindWindow("CTemplate1","CTemplate1");
    do{
        gets(str);
        data.cbData=strlen(str);
        data.lpData=str;
        SendMessage(hWnd, WM_COPYDATA,
            (LPARAM)GetFocus(),
            (LPARAM)&data);
    }while(strcmp(str,"quit"));
    return 0;
}
```

Окна предопределенных стилей. Диалоги. Ресурсы.

Файл dl.c

```
#include <windows.h>
#include "resource.h"
LRESULT CALLBACK
DlgProc(HWND hDlg, UINT message, WPARAM wParam,
LPARAM lParam);
int WINAPI WinMain(HINSTANCE hInstance,
                   HINSTANCE hPrevInstance,
                   LPSTR lpCmdLine,
                   int nCmdShow){
    MSG msg;
```

```
DialogBox( hInstance, (LPCTSTR)IDD_DLGTEST, NULL,  
          (DLGPROC)DlgProc);
```

```
while (GetMessage(&msg, NULL, 0, 0)) {  
    TranslateMessage(&msg);  
    DispatchMessage(&msg);  
}  
return msg.wParam;  
}
```

LRESULT CALLBACK

```
DlgProc( HWND hDlg, UINT message, WPARAM wParam,  
        LPARAM lParam ){
```

```
    char strText[100];  
    switch (message){  
        case WM_INITDIALOG:  
            return TRUE;
```

```
case WM_COMMAND:
    switch (LOWORD(wParam) ){
        case IDOK:
            PostQuitMessage(0);
            return TRUE;
        case IDCANCEL:
            PostQuitMessage(0);
            return TRUE;
```

```
case IDC_BTN1:  
    GetDlgItemText(hDlg, IDC_EDIT1, strText, 100);  
    SetDlgItemText (hDlg, IDC_EDIT2, strText);  
    SetDlgItemText (hDlg, IDC_EDIT1, "");  
    break;
```

```
}
```

```
break;
```

```
default:
```

```
    return FALSE;
```

```
}
```

```
}
```

Файлы ресурсов

Файл d1.rc

```
#include <windows.h>
#include "resource.h"

IDD_DLGTEST DIALOG DISCARDABLE 0, 0, 186, 95
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION |
                                     WS_SYSMENU
CAPTION "Dialog"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON "OK", IDOK, 129, 7, 50, 14
    PUSHBUTTON    "Cancel", IDCANCEL, 129, 24, 50, 14
    PUSHBUTTON    "Down",  IDC_BTN1, 7, 49, 43, 15
    EDITTEXT IDC_EDIT1,7,7,77,18, ES_AUTOHSCROLL
    EDITTEXT IDC_EDIT2,7,24,77,18,ES_AUTOHSCROLL
END
```


Файл resource.h

```
#define IDD_DLGTEST          101
#define IDC_EDIT1            1000
#define IDC_EDIT2            1001
#define IDC_BTN1             1002
```

```
>rc d1.rc      (Компилятор ресурсов)
>cl d1.c d1.res user32.lib
```

Упражнение 1: протестировать программы, разобранные на лекции.

Упражнение 2: написать примитивный калькулятор с интерфейсом, представленным диалоговым окном.

Упражнение 3: организовать передачу данных на основе WM_COPYDATA между двумя приложениями с диалоговыми окнами.