

Создание GUI-приложений

Для регистрации класса окна используется функция:

```
int RegisterClass(LPWNDCLASS wc);
```

Для создания окна вызывается функция:

```
HWND CreateWindow(...);
```

Для отображения окна приложения используется функция:

```
BOOL ShowWindow(HWND hwnd, int nCmdShow);
```

Создание GUI-приложений

Для выборки сообщений вызывается функция:

```
BOOL GetMessage(  
    LPMSG lpMsg,    // ссылка на структуру для  
                    // получения сообщения  
    HWND hwnd,     // дескриптор окна, которому  
                    // адресуется сообщение  
    UINT wMsgMin, // первое сообщение (код)  
    UINT wMsgMax); // последнее сообщение (код)
```

Создание GUI-приложений

```
typedef struct {  
    HWND hwnd;        // дескриптор окна адресата  
    UINT message;    // код сообщения  
    WPARAM wParam;   // содержимое сообщения  
    LPARAM lParam;   // содержимое сообщения  
    DWORD time;      // время отправки  
    POINT pt;        // координаты места отправки  
} MSG;
```

Цикл обработки сообщений окна:

```
while(GetMessage(...)) {  
    DispatchMessage(...);  
}
```

Создание GUI-приложений

Передача сообщения оконной процедуре:

```
LRESULT DispatchMessage(CONST MSG *lpMsg);
```

Оконная процедура:

```
LRESULT CALLBACK <имя процедуры>(  
    HWND      hwnd,  
    UINT      msg,  
    WPARAM    wParam,  
    LPARAM    lParam);
```

Создание GUI-приложений

В ОС Window описаны несколько сот кодов сообщений.

Например,

1. WM_CREATE
2. WM_SIZE
3. WM_MOVE
4. WM_COMMAND
5. WM_DESTROY

Создание GUI-приложений

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

```
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM);
```

```
HINSTANCE hIns;
```

```
char NameClass[] = “WindowClass”;
```

```
int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrev, LPSTR  
    lpszCmdLine, int nCmdShow) {
```

```
    MSG msg;  HWND hwnd;  hIns = hInst;
```

```
    WNDCLASS wc;wc.style=wc.cbClsExtra=wc.cbWndExtra=0;
```

```
    wc.lpfnWndProc=WndProc;  wc.hInstance=hIns;
```

```
    wc.hIcon=LoadIcon(NULL, IDI_APPLICATION);
```

```
    wc.hCursor=LoadCursor(NULL, IDC_ARROW);
```

```
    wc.hbrBackground=(HBRUSH)(COLOR_WINDOW+1);
```

```
    wc.lpszMenuName=NULL;
```

```
    wc.lpszClassName=NameClass;
```

Создание GUI-приложений

```
if (!RegisterClass(&wc))
    return FALSE;
hwnd = CreateWindow(NameClass, "My Application",
    WS_OVERLAPPEDWINDOW,
    CW_USEDEFAULT, CW_USEDEFAULT,
    300, 300, 0, 0, hInst, NULL);
if (!hwnd)
    return FALSE;
ShowWindow(hwnd, SW_SHOW);
while(GetMessage(&msg, 0, 0, 0))
    DispatchMessage(&msg);
return msg.wParam;
}
```

Создание GUI-приложений

```
LRESULT CALLBACK WndProc (HWND hwnd,  
    UINT msg, WPARAM wParam, LPARAM lParam) {  
    switch(msg) {  
        case WM_DESTROY: {  
            PostQuitMessage(0);  
            return 0;  
        }  
    }  
    return DefWindowProc(hwnd, msg, wParam,  
        lParam);  
}
```

Создание GUI-приложений

ЭЛЕМЕНТЫ УПРАВЛЕНИЯ

Элементы управления

Элемент управления – это компонент окна приложения, который используется для управления работой этого приложения.

Отличительные особенности элементов управления:

1. для них описаны классы окон;
2. все они являются дочерними окнами;
3. для всегда необходимо описывать идентификаторы;
4. для них описаны дополнительные стили и набор сообщений.

Элементы управления

Классы для создания элементов управления

1. “button” - для реализации любых типов кнопок;
2. “edit” - для реализации поля ввода (редактирования);
3. “listbox” - для реализации списка;
4. “combobox” - для реализации комбинированного списка (поле ввода со списком);
5. “scrollbar” - для реализации полосы прокрутки;
6. “static” - для реализации статического поля.

Элементы управления

Алгоритм создания элемента управления:

1. Описание идентификатора

(например, `#define ID_BUTTON 200`);

2. Описание дескриптора элемента

(например, в теле оконной процедуры родительского окна:
`static HWND hButton`);

3. Создание элемента

(с помощью функции `CreateWindow()` со стилем `WS_CHILD`. Например,

```
hButton= CreateWindow("button", "OK",  
WS_CHILD|WS_VISIBLE, 0, 0, 40, 20, hwnd,  
(HMENU)ID_BUTTON, hIns, NULL) )
```

Элементы управления

```
#include <windows.h>
#define ID_BUTTON1 101
#define ID_BUTTON2 102
    // аналогично предыдущему примеру
LRESULT CALLBACK WndProc (HWND hwnd, UINT msg,
    WPARAM wParam, LPARAM lParam) {
    static HWND hBut1, hBut2;
    switch(msg) {
        case WM_CREATE: {
            CreateWindow("button", "K1", WS_CHILD|WS_VISIBLE,
                50, 50, 40, 20, hwnd, (HMENU)ID_BUTTON1,
                hIns, NULL);
            CreateWindow("button", "K2", WS_CHILD|WS_VISIBLE,
                130, 50, 40, 20, hwnd, (HMENU)ID_BUTTON2,
                hIns, NULL);
            return 0;
        }
    }
```

Элементы управления

```
case WM_COMMAND: {
    switch (LOWORD(wParam)) {
        case ID_BUTTON1: {
            MessageBox(hwnd, "Press K1", "Mes",
                MB_OK);
            return 0;
        }
        case ID_BUTTON2: {
            MessageBox(hwnd, "Press K2", "Mes",
                MB_OK);
            return 0;
        }
    }
    return 0;
}
```

Элементы управления

```
    case WM_DESTROY: {  
        PostQuitMessage(0);  
        return 0;  
    }  
}  
  
return DefWindowProc(hwnd, msg, wParam,  
lParam);  
}
```

Элементы управления

Работа с меню

Работа с меню приложения

Меню используются для изменения режимов работы приложения.

Состоит меню из пунктов, команд и сепараторов.

Виды меню:

- Системное;
- Главное;
- Контекстное (всплывающее).

Работа с меню приложения

Элемент меню может быть представлен:

- строкой символов;
- графическим объектом (растровое изображение).

Пункт меню, как и контекстное меню, создается в виде временного меню.

Работа с меню приложения

Существует два способа создания меню:

- Динамический (в период выполнения приложения: либо в главной функции, либо в оконной процедуре обработки сообщений);
- Статический (до запуска приложения на выполнение: использование файла ресурсов).