

# Системное программирование

## Лекция №16

# Программирование в Win32 API

## Венгерская нотация

- Каждое слово в имени переменной пишется с прописной буквы и слитно с другими словами.
- Каждый идентификатор предваряется несколькими строчными буквами, определяющими его тип.

`nMyVariable` — переменная целого типа

`cYourVariable` — символьная переменная (`char`)

`pszMyBuffer` — указатель на строку с нулевым ограничителем  
(`pointer to string terminated by zero`)

## Венгерская нотация

Префикс	Тип данных
b	BYTE (unsigned char)
cx, cy	short (используются, как ширина и длина объектов типа RECT или окон)
dw	DWORD (unsigned long)
fn	function
h	HANDLE
i	int
l	LONG
n	int or short
s	string
sz	string terminated by zero
w	WORD (unsigned int)
x,y	short (используются, как координаты)
c	char

# Очередь сообщений

- В Windows существует одна общесистемная очередь сообщений и очереди сообщений у каждого окна (First In First Out).
- Операционная система реализует циклы, в ходе которых опрашивается очередь и выбирается информация о сообщениях в них.
- При запуске каждой программы должно быть создано окно и запущен цикл обработки сообщений.
- Функция `WinMain()` является стандартной точкой входа в программы для Windows; функция окна от нее отделена.

## Программа для Windows

WinMain(список аргументов)

{

Подготовка и создание класса окон с заданными характеристиками

Создание экземпляра окна только что созданного класса

Пока не произошло необходимое для выхода событие

    Опрашивать очередь сообщений и передавать их оконной функции;

Возврат из программы;

}

WindowFunction(список аргументов)

{

Обработать полученное сообщение;

Возврат;

}

## Программа для Windows

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

```
LRESULT CALLBACK HelloWorldWndProc ( HWND, UINT, UINT, LONG);
```

```
int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR  
lpszCmdParam, int nCmdShow )
```

```
{
```

```
HWND hWnd;
```

```
WNDCLASS WndClass;
```

```
MSG Msg;
```

```
char szClassName[] = «HelloWorld»;
```

```
/*Регистрируем создаваемый класс*/
```

```
/*Заполняем структуру типа WNDCLASS */
```

```
WndClass.style = CS_HREDRAW | CS_VREDRAW;
```

```
WndClass.lpfnWndProc = HelloWorldWndProc;
```

```
WndClass.cbClsExtra = 0;
```

```
WndClass.cbWndExtra = 0;
```

## Программа для Windows

```
WndClass.hInstance = hInstance;  
WndClass.hIcon = LoadIcon (NULL, IDC_ APPLICATION);  
WndClass.hCursor = LoadCursor (NULL, . IDC_ ARROW);  
WndClass.hbrBackground = (HBRUSH) GetStockObject (WHITE_ BRUSH);  
WndClass.lpszMenuName = NULL;  
WndClass.lpszClassName = szClassName;  
  
if(!RegisterClass(&WndClass) )  
{  
    MessageBox(NULL, »Cannot register class», »Error», MB_ OK);  
    return 0;  
}
```

## Программа для Windows

```
hWnd = CreateWindow(szClassName, «Program No1»,  
                    WS_OVERLAPPEDWINDOW,  
                    CW_USEDEFAULT, CW_USEDEFAULT,  
                    CW_USEDEFAULT, CW_USEDEFAULT,  
                    NULL, NULL,  
                    hInstance, NULL);
```

```
if(!hWnd)  
{  
    MessageBox(NULL, »Cannot create window», »Error», MB_OK);  
    return 0;  
}
```



## Программа для Windows

```
/*Показать наше окно*/
```

```
ShowWindow (hWnd, nCmdShow);
```

```
UpdateWindow(hWnd);
```

```
/*Начало цикла сообщений*/
```

```
while (GetMessage (&Msg, NULL, 0, 0))
```

```
{
```

```
TranslateMessage(&Msg);
```

```
DispatchMessage(&Msg);
```

```
}
```

```
return Msg.wParam;
```

```
LRESULT CALLBACK HelloWorldWndProc ( HWND hWnd, UINT Message,  
                                     UINT wParam, LONG lParam);
```

```
{
```

# Программа для Windows

```
HDC hDC;  
PAINTSTRUCT PaintStruct;  
RECT Rect;  
switch(Message);  
{  
  case WM_PAINT:  
    hDC = BeginPaint(hWnd, &PaintStruct);  
    GetClientRect(hWnd, &Rect);  
    DrawText(hDC, »Hello, World!«, -1, &Rect,  
            DT_SINGLELINE | DT_CENTER | DT_VCENTER);  
    EndPaint(hWnd, &PaintStruct);  
    return 0;
```

## Программа для Windows

```
case WM_DESTROY:  
    PostQuitMessage(0);  
    return 0;  
}
```

```
return DefWindowProc(hWnd, Message, wParam, lParam);  
}
```

# Программа для Windows

**WINAPI** — определяет порядок передачи параметров при вызове процедуры

**hInstance** HINSTANCE = HANDLE = void\*; условный номер экземпляра программы  
(хэндл)

**hPrevInstance** не используется

**pszCmdLine** — указатель на командную строку, которая набирается после имени  
запускаемой программы

**nCmdShow** — определяет, в каком виде окно появится на экране

## Возможные значения параметра nCmdShow

<b>Параметр</b>	<b>Значение</b>	<b>Параметр</b>	<b>Значение</b>
SW_HIDE	0	SW_SHOW	5
SW_SHOWNORMAL	1	SW_MINIMIZE	6
SW_SHOWMINIMIZED	2	SW_SHOWMINNOACTIVE	7
SW_SHOWMAXIMIZED	3	SW_SHOWNA	8
SW_SHOWNOACTIVE	4	SW_RESTORE	9

## Функция CreateWindow

Аргумент 1 : указатель на строку с именем класса, к которому принадлежит создаваемое окно

Аргумент 2: указатель на строку текста - заголовка окна

Аргумент 3: стиль окна (индивидуальные характеристики конкретного окна). В файле `winuser.h` определено несколько десятков стилей; их идентификаторы начинаются с `WS`.

Аргументы 4-7: положение окна на экране (в пикселях) – отступы верхнего левого угла от левого края, верхней границы, ширина и высота окна.

Аргумент 8: хэндл окна, являющегося родительским по отношению к данному.

Аргумент 9: хэндл меню окна.

Аргумент 10: хэндл экземпляра запускаемой программы.

Аргумент 11: дополнительные данные для некоторых случаев запуска программы.

# Функция ShowWindow

Аргумент 1 : хэндл окна

Аргумент 2: вид отображения окна на экране

## Цикл обработки сообщений

Сообщение – это структура

Поле 1: хэндл окна-адресата

Поле 2: номер сообщения

Поле 3,4: параметры сообщения

Поле 5: время выдачи сообщения

Поле 6: позиция, на которой находится курсор в момент выдачи сообщения



## Функция GetMessage

Аргумент 1 : указатель на структуру типа MSG

Аргумент 2: хэндл окна, созданного программой

Аргумент 3,4: нижняя и верхняя границы номеров сообщений, которые разрешено передавать оконной функции

Функция всегда возвращает ненулевое значение – по нулевому значению работа программы прекращается. Это сообщение имеет имя WM\_QUIT