

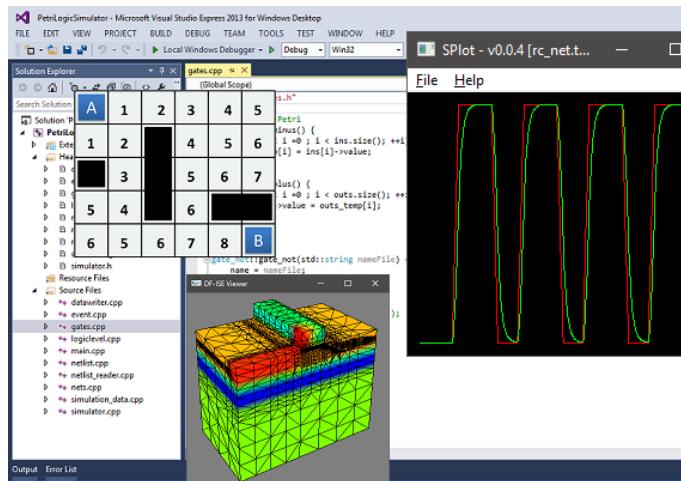


МИЭТ

Национальный исследовательский университет «МИЭТ»

Институт интегральной электроники (группы ЭН-34-35, каф. ПКИМС)

# Программные средства САПР



Лекция 1

Основы программирования приложений  
с графическим интерфейсом  
пользователя

# Консольные приложения сегодня

Command Prompt - run

```
06/16/2004 12:38 AM      7,572 phoenix-loader.jar
06/16/2004 12:38 AM      6,162 phoenix.sh
06/16/2004 12:38 AM      2,643 run.bat
06/16/2004 12:38 AM      854 run.sh
06/16/2004 12:38 AM     49,152 Wrapper.dll
06/16/2004 12:38 AM     98,384 Wrapper.exe
06/16/2004 12:38 AM     24,290 wrapper.jar
? File(s)          188,977 bytes
3 Dir(s)        962,379,776 bytes free

C:\projects\lib\james-2.2.0\bin>run
Using PHOENIX_HOME: C:\projects\lib\james-2.2.0
Using PHOENIX_TMPDIR: C:\projects\lib\james-2.2.0\temp
Using JAVA_HOME:      c:\J2sdk1.4.2_10

Phoenix 4.0.1

James 2.2.0
Remote Manager Service started plain:4555
POP3 Service started plain:110
SMTP Service started plain:25
NNTP Service started plain:119
Fetch POP Disabled
FetchMail Disabled
```

Terminal — bash — 90x27

```
PowerBook-G4:~ michaelhogg$ ioreg -l
+-o Root <class IORegistryEntry, retain count 11>
| {
|   "IOMaximumMappedIOByteCount" = 536870912
|   "IONDRVFramebufferGeneration" = <0000000200000002>
|   "IOKitDiagnostics" = {"Instance allocation"=1131794,"Classes"={"AppleFWOHCI_AsyncTr$}
|   "IORegistryPlanes" = {"IOdeviceTree"="IOdeviceTree","IOUSB"="IOUSB","IOService"="IO$}
|   "IOKitBuildVersion" = "Darwin Kernel Version 8.0.0: Sat Mar 26 14:15:22 PST 2005; r$}
|   "IOConsoleUsers" = {{"kCGSessionGroupIDKey"=501,"kCGSessionOnConsoleKey"=Yes,"kCG$}
| }

+-o PowerBook5,6 <class IOPlatformExpertDevice, registered, matched, active, busy 0, r$>
| {
|   "als-lgp-version" = <00000001>
|   "IONWInterrupts" = "IONWInterrupts"
|   "IOPlatformArgs" = <00d4b0000d4400000000000000000000>
|   "system-id" = <000000000000>
|   "graphics-setaggressiveness" = <>
|   "display-config-info" = <0000000000000000>
|   "name" = <"device-tree">
|   "AAPL,add-fcode-file" = <ff863b58>
|   "#size-cells" = <00000001>
|   "device_type" = <"bootrom">
|   "scb#" = <00000001>
|   "customer-sv-config" = <">
|   "powertreeDESC" = {{"service"="IOPMU8eMacRISC2 is n$}
|   "model" = <"PowerBook5,6">
```

Terminal 1

Welcome to the Be shell.

```
$ cd ../..
$ ls
Boot Disk boot      etc      system      var
bin       dev      pipe      tmp
$ ftp ftp.be.com
Connected to www.be.com.
220 www.be.com FTP server (Version wu-2.4.2-academ[BETA-12](3) Wed Mar 5 17:23:4
9 PST 1997) ready.
Name (ftp.be.com:demo): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password:230-Welcome to the Be FTP site! All transfers are logged.
230-
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> quit
221 Goodbye.
$
```

mgagne : bash - Konsole

Filesystem	1K-blocks	Used	Available	Use%	Mounted on
/dev/sdal	98429572	9506948	83922688	11%	/
udev	2965616	4	2965612	1%	/dev
tmpfs	1189620	1220	1188400	1%	/run
none	5120	0	5120	0%	/run/lock
none	2974048	4028	2970020	1%	/run/shm
none	102400	12	102388	1%	/run/user
cgroup	2974048	0	2974048	0%	/sys/fs/cgroup
/dev/sda6	614770712	79382432	504159660	14%	/home
//bvault/marcel	1937388032	1658634760	278753272	86%	/mnt/vault
//bvault/videos	1937388032	1658634760	278753272	86%	/mnt/videos
//bvault/share	1937388032	1658634760	278753272	86%	/mnt/share

```
mgagne@fullhouse ~ $ df
mgagne@fullhouse ~ $ date
Fri Mar 15 10:51:59 EDT 2013
mgagne@fullhouse ~ $ cal
            March 2013
Su Mo Tu We Th Fr Sa
                    1  2
3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
mgagne@fullhouse ~ $
```

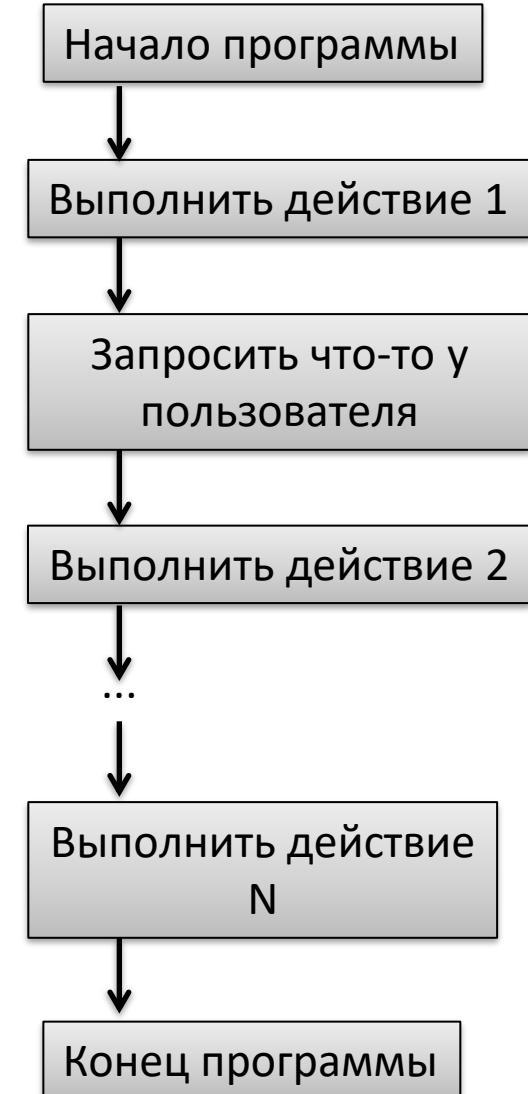
# Архитектура консольных программ

```
#include <stdio.h>
```

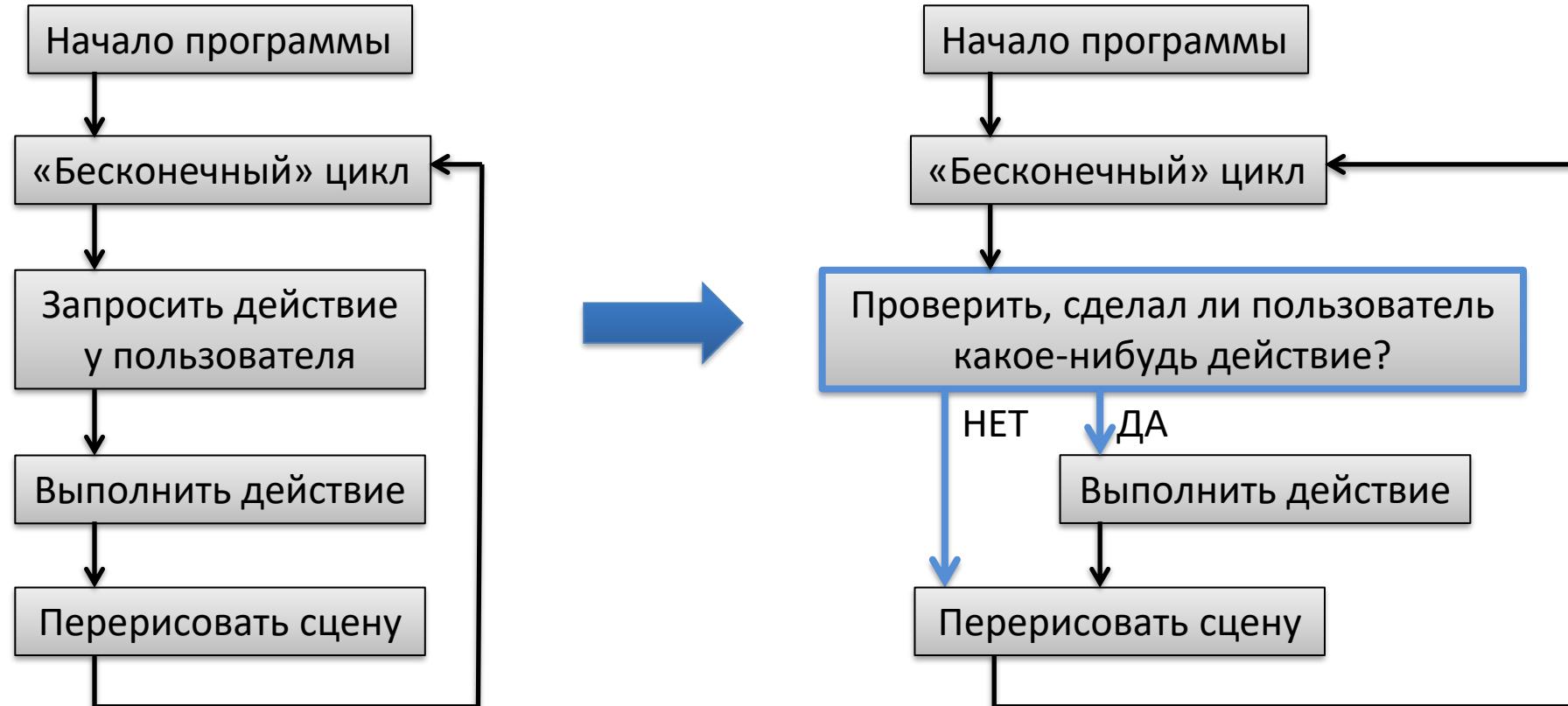
```
int main(int, char **) {
    printf("Hello!");
    return 0;
}
```

```
#include <stdio.h>
```

```
int main(int, char **) {
    printf("Enter your name : ");
    char userName[32];
    gets(userName);
    printf("Hello, %s!", userName);
    return 0;
}
```



# Как была реализована анимация?



Подождать нажатия клавиш:  
`getch()`

Проверить состояние клавиатуры:  
`kbit() + getch()`

# Сложность «оконных» программ – что нужно отслеживать

## Система

Окно:

- активировано/деактивировано
- нажали кнопку NC области
- действие мыши в NC области
- надо рисовать

...

Один из «компонентов» - текст:

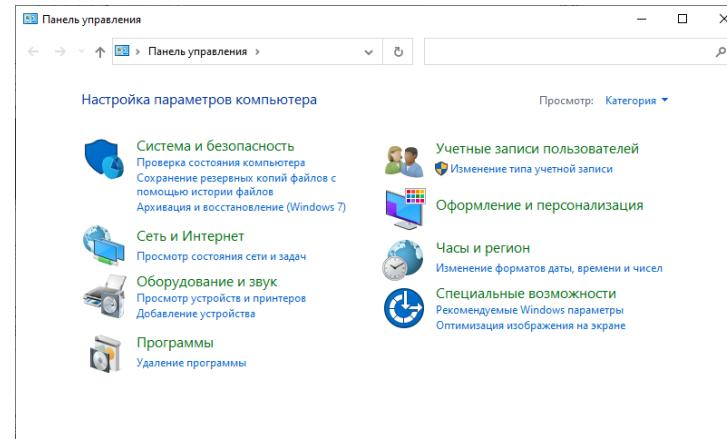
- выделили область
- скопировали/вырезали/вставили текст

...

События от таймера.

Межпроцессное взаимодействие.

...



## Пользователь

Данные от клавиатуры:

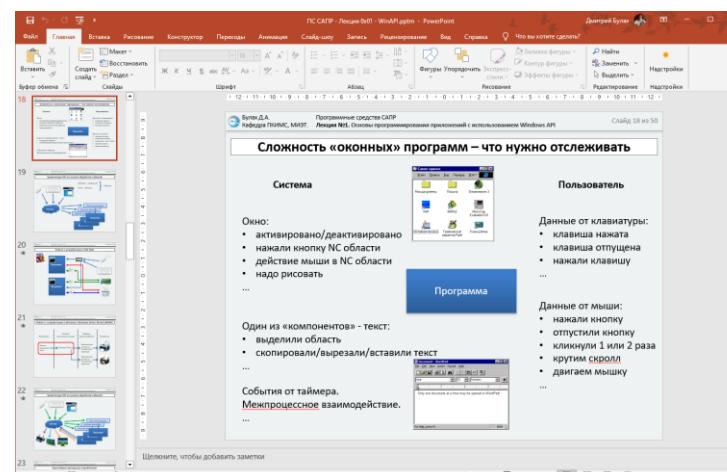
- клавиша нажата
- клавиша отпущена
- нажали клавишу

...

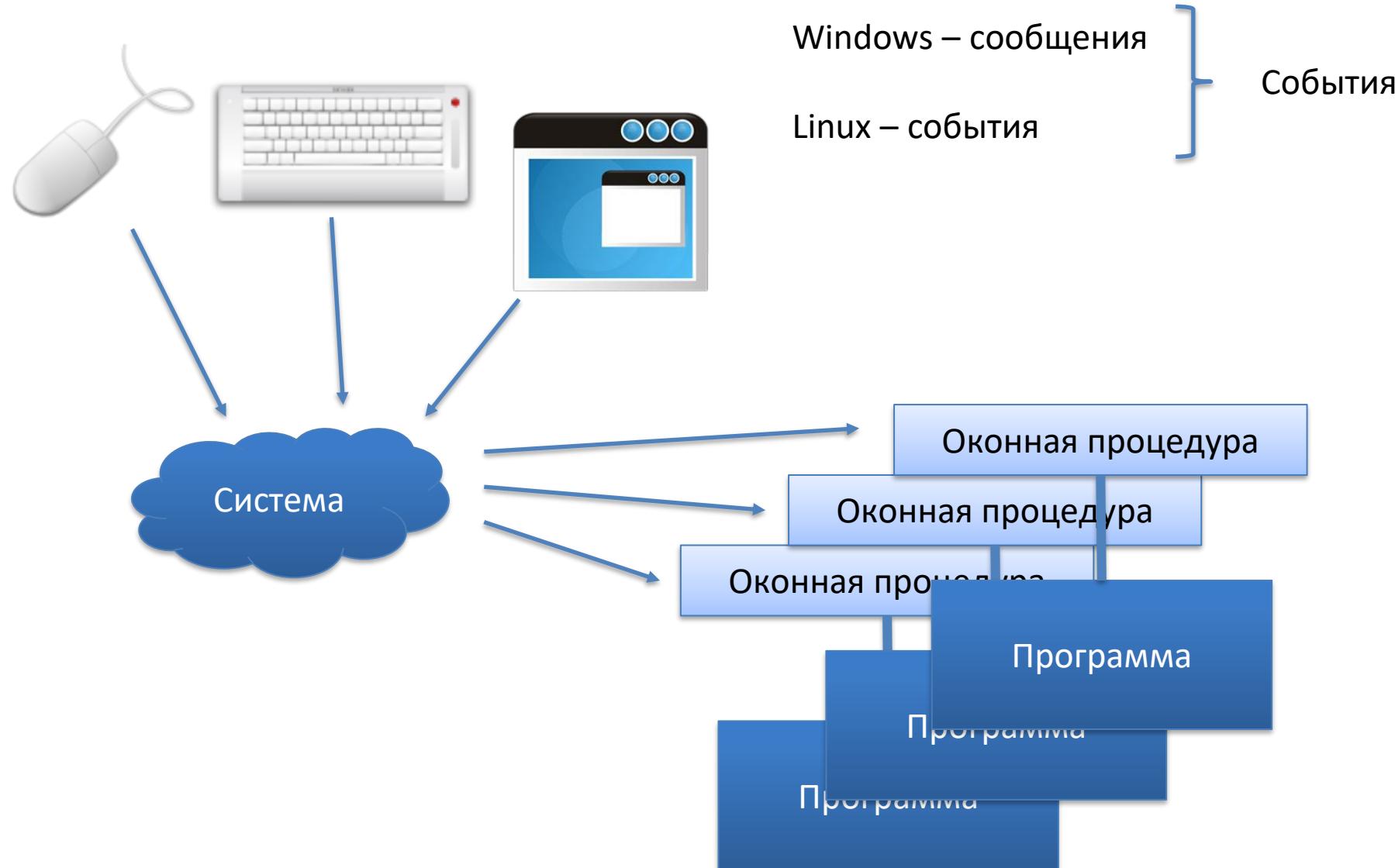
Данные от мыши:

- нажали кнопку
- отпустили кнопку
- кликнули 1 или 2 раза
- крутим скролл
- двигаем мышку

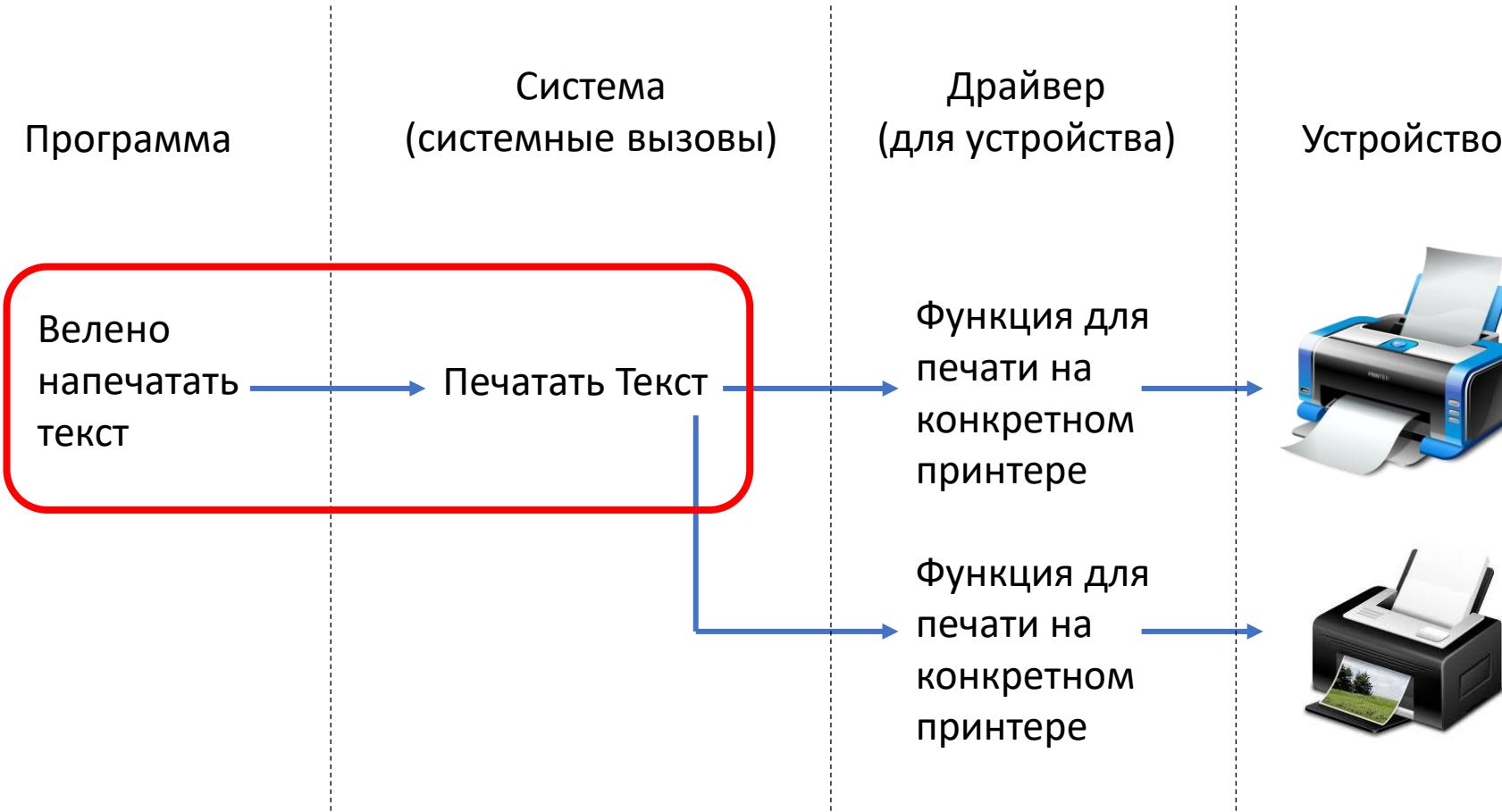
...



# Архитектура ПО на основе обработки событий



# Работа с устройствами в Windows: Windows Driver Model (WDM)



# Простейшая программа с Windows API

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

#define szWindowClass "MyWindow"
#define szTitle "A Simple Window"

int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {
    WNDCLASSEX wcex;

    wcex.cbSize = sizeof(WNDCLASSEX);
    wcex.style = CS_HREDRAW | CS_VREDRAW;
    wcex.lpfnWndProc = WndProc;
    wcex.cbClsExtra = 0;
    wcex.cbWndExtra = 0;
    wcex.hInstance = hInstance;
    wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI_APPLICATION));
    wcex.hCursor = LoadCursor(NULL, IDC_ARROW);
    wcex.hbrBackground = (HBRUSH)(COLOR_WINDOW + 1);
    wcex.lpszMenuName = NULL;
    wcex.lpszClassName = szWindowClass;
    wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI_APPLICATION));

    if (!RegisterClassEx(&wcex)) {
        MessageBox(NULL, "Can't register window class!", "Win32 API Test", NULL);
        return 1;
    }

    HWND hWnd = CreateWindow(szWindowClass, szTitle, WS_OVERLAPPEDWINDOW, CW_USEDEFAULT,
                           CW_USEDEFAULT, 500, 400, NULL, NULL, hInstance, NULL);

    if (!hWnd) {
        MessageBox(NULL, "Can't create window!", "Win32 API Test", NULL);
        return 1;
    }

    ShowWindow(hWnd, SW_SHOWNORMAL);
    UpdateWindow(hWnd);

    MSG msg;
    while (GetMessage(&msg, NULL, 0, 0)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }

    return msg.wParam;
}
```

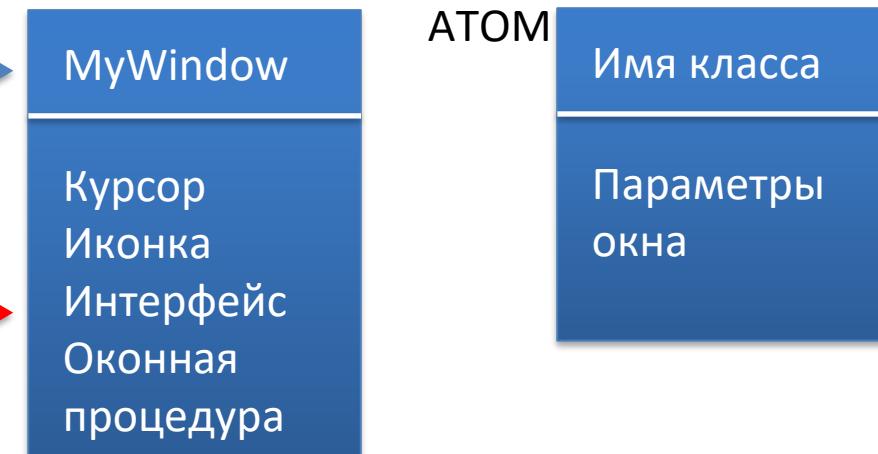
- + оконная процедура
- } Инициализация параметров создаваемого окна
- } Регистрация окна с нашими параметрами в системе
- } Создание экземпляра окна
- } Показать окно
- } Запуск обработчика событий

# Шаг 1. Регистрация окна в системе

```
LRESULT __stdcall WindowProc(...);

#define szWindowClass "MyWindow"

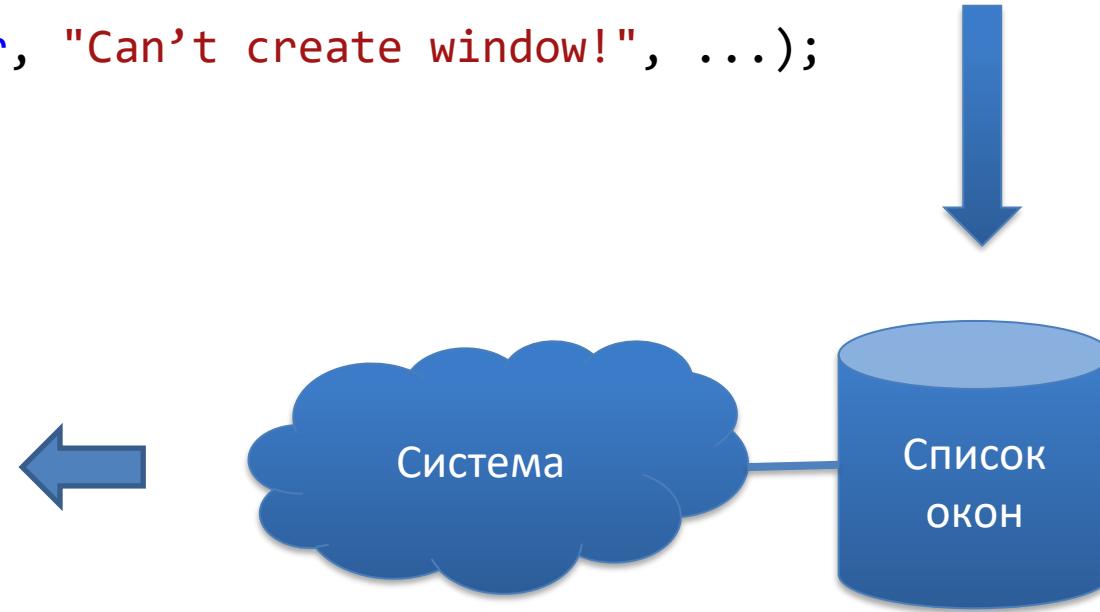
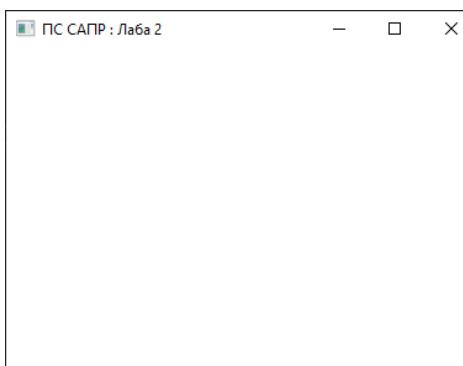
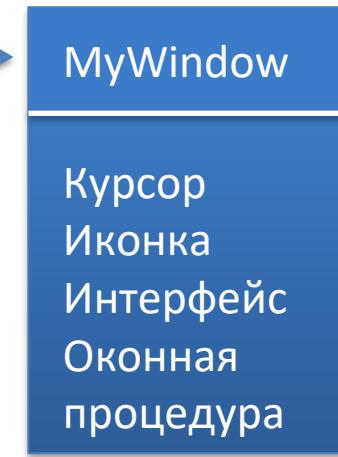
int WinMain(...) {
    WNDCLASSEX wc;
    ...
    wc.hIcon        = LoadIcon(nullptr, IDI_APPLICATION);
    wc.hCursor      = LoadCursor(nullptr, IDC_ARROW);
    wc.lpfnWndProc  = WindowProc;
    wc.lpszClassName = szWindowClass;
    ...
    RegisterClassEx(&wc);
    ...
}
```



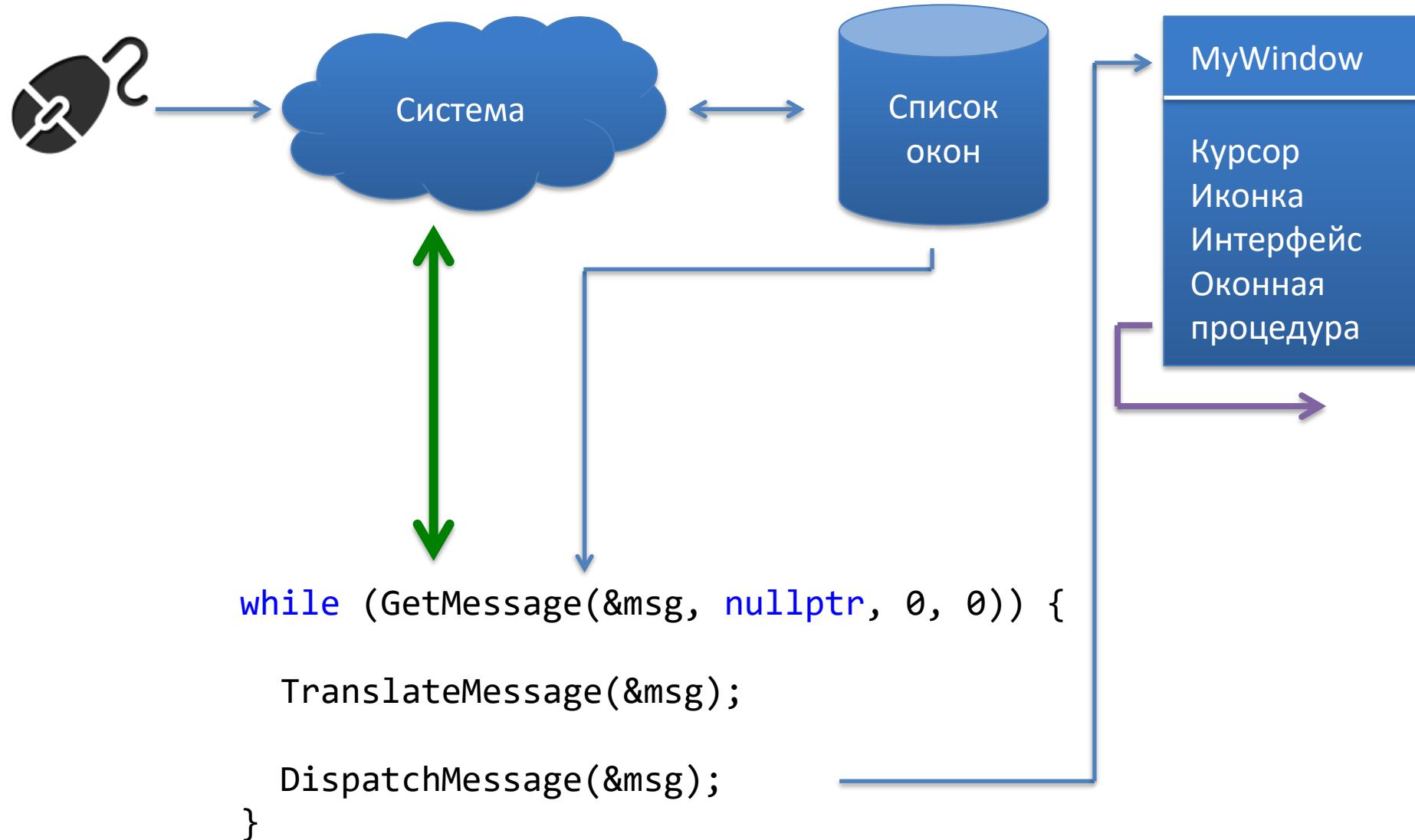
## Шаг 2. Создание экземпляра окна

```
HWND hWnd = CreateWindow(szWindowClass,
    "A Simple Window", ...);

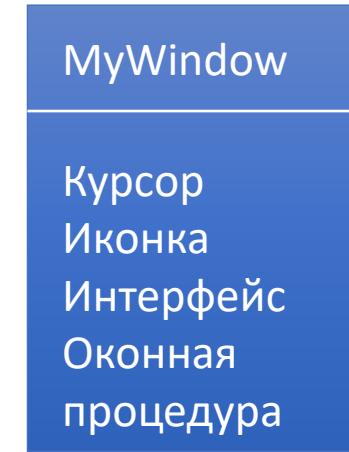
if (!hWnd) {
    MessageBox(nullptr, "Can't create window!", ...);
    return 1;
}
```



## Шаг 3. Ожидание сообщений от операционной системы



## Шаг 4 (основной). Обработка сообщений от операционной системы (1)



```
long __stdcall WndProcedure(HWND hWnd,  
                           UINT Msg,  
                           WPARAM wParam,  
                           LPARAM lParam) {  
  
    switch (Msg) {  
        case WM_DESTROY: PostQuitMessage(WM_QUIT);  
        break;  
  
        case WM_PAINT: ...  
  
        case WM_LBUTTONDOWN: ...  
  
        case WM_SIZE: ...  
  
        default: return DefWindowProc(hWnd, Msg, wParam, lParam);  
    }  
    return 0;  
}
```

## Шаг 4 (основной). Обработка сообщений от операционной системы (2)

```
long __stdcall WndProcedure(HWND hWnd,
                           UINT Msg,
                           WPARAM wParam,
                           LPARAM lParam) {

    switch (Msg) {
        ...
        case WM_PAINT:
            hDC = BeginPaint(hWnd, &ps);
            OnPaint(hDC);
            EndPaint(hWnd, &ps);
            break;

        case WM_LBUTTONDOWN:
            x = LOWORD(lParam);
            y = HIWORD(lParam);
            OnLButtonDown(x, y);
            break;
        ...
    }
    return 0;
}
```

Функция – обработчик  
отрисовки

Функция – обработчик  
нажатия на кнопку мыши

# Аргументы оконной процедуры

```
long __stdcall WndProc(HWND hWnd,
                      UINT message,
                      WPARAM wParam,
                      LPARAM lParam)
```

```
PAINTSTRUCT ps;
HDC hDC;
```

```
switch (message) {
...
case WM_PAINT:
    hDC = BeginPaint(hWnd, &ps);
    OnPaint(hDC);
    EndPaint(hWnd, &ps);
    break;

case WM_LBUTTONDOWN:
```

```
x = ???
```

```
y = ???
```

```
...
```

```
break;
```

```
}
```

## WM\_LBUTTONDOWN message

05/31/2018 • 2 minutes to read •

Posted when the user presses the left mouse button while the cursor is in the client area of a window. If the mouse is not captured, the message is posted to the window beneath the cursor. Otherwise, the message is posted to the window that has captured the mouse.

A window receives this message through its [WindowProc](#) function.

C++

```
#define WM_LBUTTONDOWN 0x0201
```

Copy

### Parameters

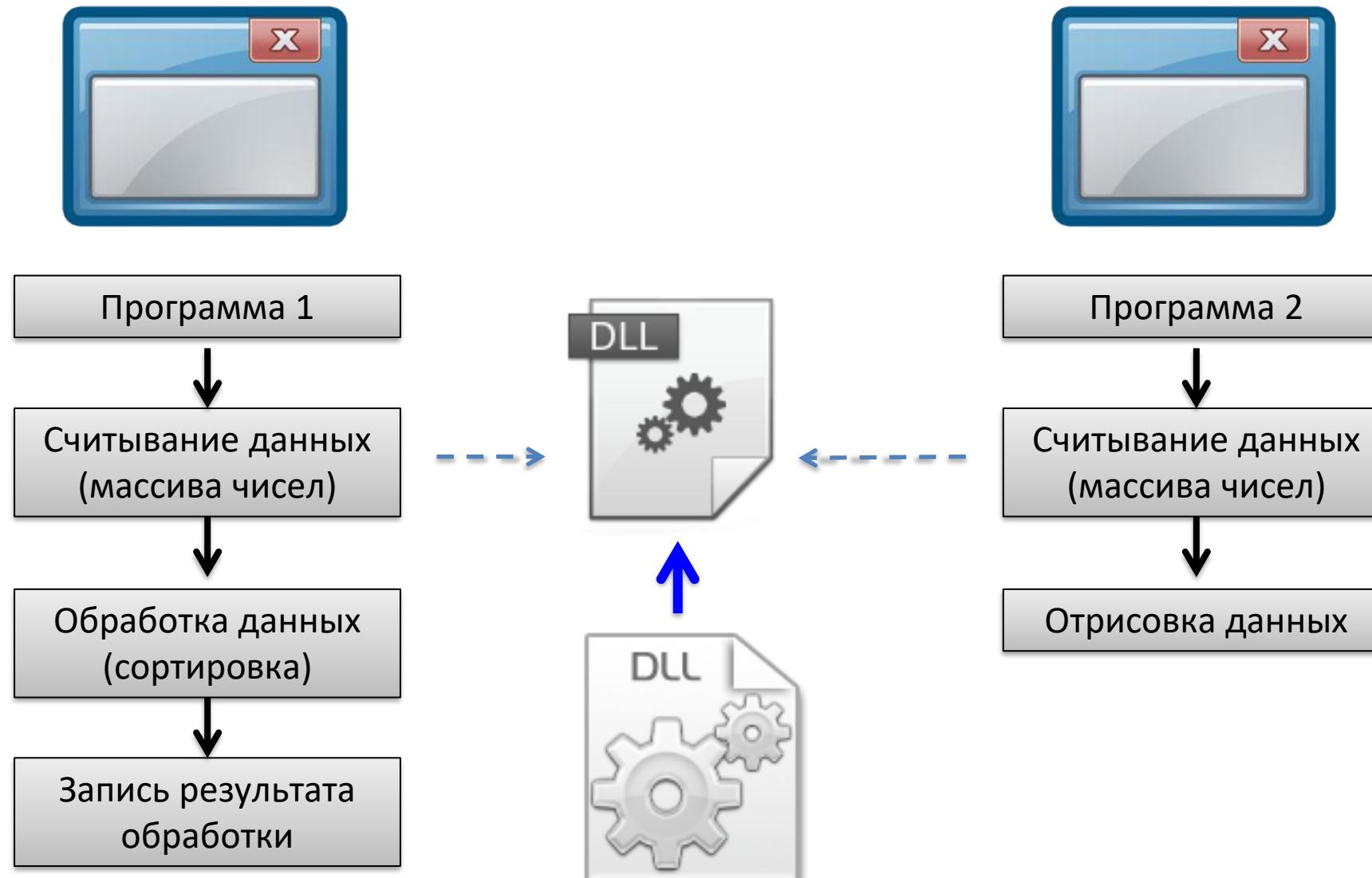
wParam

Indicates whether various virtual keys are down. This parameter can be one or more of the following values.

Value	Meaning
MK_CONTROL 0x0008	The CTRL key is down.
MK_LBUTTON 0x0001	The left mouse button is down.

\*<https://learn.microsoft.com/en-us/windows/win32/inputdev/wm-lbuttondown>

# Динамически подключаемые библиотеки



# ОС Windows: Windows API

```
#include <windows.h>
#include <stdlib.h>
#include <string.h>
#include <tchar.h>

static TCHAR szWindowClass[] = _T("win32app");
static TCHAR szTitle[] = _T("A Simple Window");

HINSTANCE hInst;

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

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {
    WNDCLASSEX wcex;

    wcex.cbSize = sizeof(WNDCLASSEX);
    wcex.style = CS_HREDRAW | CS_VREDRAW;
    wcex.lpfnWndProc = WndProc;
    wcex.cbClsExtra = 0;
    wcex.cbWndExtra = 0;
    wcex.hInstance = hInstance;
    wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI_APPLICATION));
    wcex.hCursor = LoadCursor(NULL, IDC_ARROW);
    wcex.hbrBackground = (HBRUSH)(COLOR_WINDOW + 1);
    wcex.lpszMenuName = NULL;
    wcex.lpszClassName = szWindowClass;
    wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI_APPLICATION));

    if (!RegisterClassEx(&wcex)) {
        MessageBox(NULL, _T("Call to RegisterClassEx failed!"), _T("Win32 Guided Tour"), NULL);
        return 1;
    }

    hInst = hInstance;
    HWND hWnd = CreateWindow(
        szWindowClass,
        szTitle,
        WS_OVERLAPPEDWINDOW,
        CW_USEDEFAULT, CW_USEDEFAULT,
        500, 100,
        NULL,
        NULL,
        hInstance,
        NULL
    );
    if (!hWnd) {
        MessageBox(NULL, _T("Call to CreateWindow failed!"), _T("Win32 Guided Tour"), NULL);
        return 1;
    }

    ShowWindow(hWnd, nCmdShow);
    UpdateWindow(hWnd);

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

long __stdcall WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam) {
    PAINTSTRUCT ps;
    HDC hdc;
    TCHAR greeting[] = _T("Hello, World!");

    switch (message) {
        case WM_PAINT:
            hdc = BeginPaint(hWnd, &ps);
            TextOut(hdc, 5, 5, greeting, strlen(greeting));
            EndPaint(hWnd, &ps);
            break;
        case WM_DESTROY:
            PostQuitMessage(0);
            break;
        default:
            return DefWindowProc(hWnd, message, wParam, lParam);
            break;
    }
    return 0;
}
```

# \*nix : API Xlib под X Window System (1)

```
#include <X11/Xlib.h>
#include <stdio.h>
#include <stdlib.h>

int main(int, char *[]) {
    Display *p_display = XOpenDisplay(NULL);
    if (!p_display) {
        printf("Can't open display.\n");
        return EXIT_FAILURE;
    }
    Window window = XCreateSimpleWindow(p_display, XDefaultRootWindow(p_display),
                                         100, 100, 200, 200, 4, 0, 0);
    XMapWindow(p_display, window);
    XSelectInput(p_display, window, NoEventMask);

    XEvent event;
    for (;;) {
        XNextEvent(p_display, &event);
    }

    XDestroyWindow(p_display, window);
    XCloseDisplay(p_display);
    return EXIT_SUCCESS;
}
```



# Кроссплатформенные библиотеки: Gtk

```
#include <gtk/gtk.h>

void destroy() {
    gtk_main_quit();
}

int main(int argc, char *argv[]) {
    gtk_init(&argc, &argv);

    GtkWidget *window = gtk_window_new(GTK_WINDOW_TOPLEVEL);

    gtk_signal_connect(GTK_OBJECT(window),
                      "destroy",
                      GTK_SIGNAL_FUNC(destroy),
                      NULL);

    gtk_widget_show(window);

    gtk_main();

    return 0;
}
```



GTK

# Кроссплатформенные библиотеки: wxWidgets

```
#include <wx/wx.h>

class Frame : public wxFrame {
public:
    Frame(const wxString& title);
};

Frame::Frame(const wxString& title)
: wxFrame(NULL, wxID_ANY, title, wxDefaultPosition,
wxSize(250, 150)) {

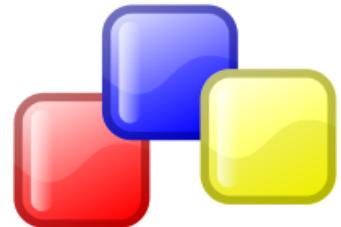
    Centre();
}

class SimpleApp : public wxApp {
public:
    virtual bool OnInit();
};

IMPLEMENT_APP(MyApp)

bool MyApp::OnInit() {
    SimpleApp *p_app = new SimpleApp(wxT("Simple"));
    p_app->Show(true);

    return true;
}
```



wxWidgets

# Кроссплатформенные библиотеки: Qt

```
#include <QMainWindow>
#include < QApplication >

class MyWindow : public QMainWindow {
public:
    MyWindow(QWidget *parent = 0);
};

MyWindow::MyWindow(QWidget *parent) : QMainWindow(parent) {
    setWindowTitle("Simple");
    resize(400, 200);
}

int main(int argc, char *argv[]) {
    QApplication app(argc, argv);

    MyWindow *win = new MyWindow;
    win->show();

    return app.exec();
}
```



Qt

# Пример портов кроссплатформенных C++ библиотек



C++

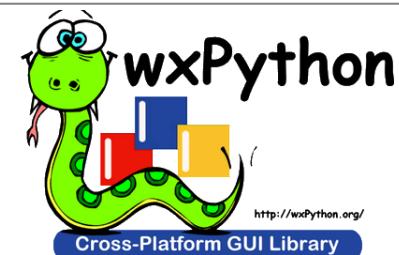
```
#include <wx/wx.h>

class MyApp : public wxApp {
public:
    virtual bool OnInit();
};

bool MyApp::OnInit() {
    wxFrame *frame = new wxFrame(nullptr, wxID_ANY, "Example");
    frame->Show(true);

    return true;
}

IMPLEMENT_APP(MyApp)
```



Python:

```
#!/usr/bin/python

import wx
app = wx.App()
frame = wx.Frame(None, -1, 'Example')
frame.Show()
app.MainLoop()
```