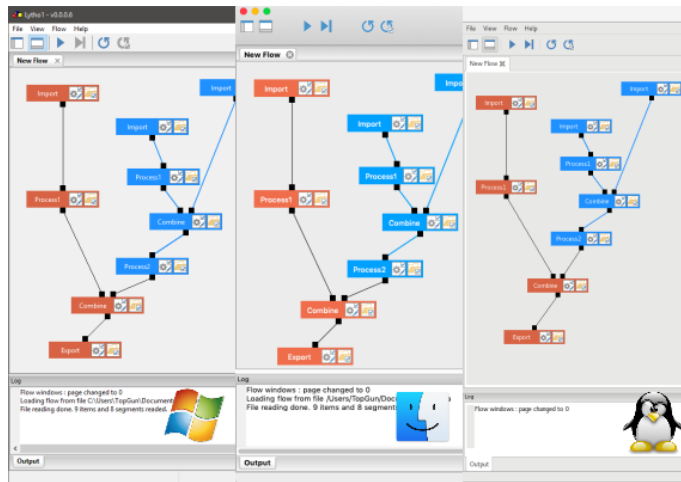




# Кроссплатформенная разработка программного обеспечения

Лабораторная работа №1

Нативные API. WinAPI



## Кроссплатформенный код: соглашение POSIX

**POSIX** (*portable operating system interface*) — системный API, описывающих интерфейсы между операционной системой и прикладной программой.

```
FILE *fopen(const char *path, const char *mode);
```

```
int fclose(FILE *stream);
```

```
int printf(const char *format, ...);
```

```
int fprintf(FILE *stream, const char *format, ...);
```

```
int sprintf(char *str, const char *format, ...);
```

```
int snprintf(char *str, size_t size, const char *format, ...);
```

# Код для программирования GUI: Windows

```
#include <windows.h>

static wchar_t windowClass[] = L"win32app";
static wchar_t windowTitle[] = L"Win32API - Пример 1";

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

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

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

    if (!RegisterClassEx(&wcx)) {
        MessageBox(nullptr, L"Call to RegisterClassEx failed!", L"Win32 API Test", MB_OK);
        return 1;
    }

    HWND hWnd = CreateWindow(windowClass, windowTitle, WS_OVERLAPPEDWINDOW,
                             CW_USEDEFAULT, CW_USEDEFAULT, 500, 400, nullptr, nullptr, hInstance, nullptr);

    if (!hWnd) {
        MessageBox(nullptr, L"Call to CreateWindow failed!", L"Win32 API Test"), MB_OK);
        return 1;
    }

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

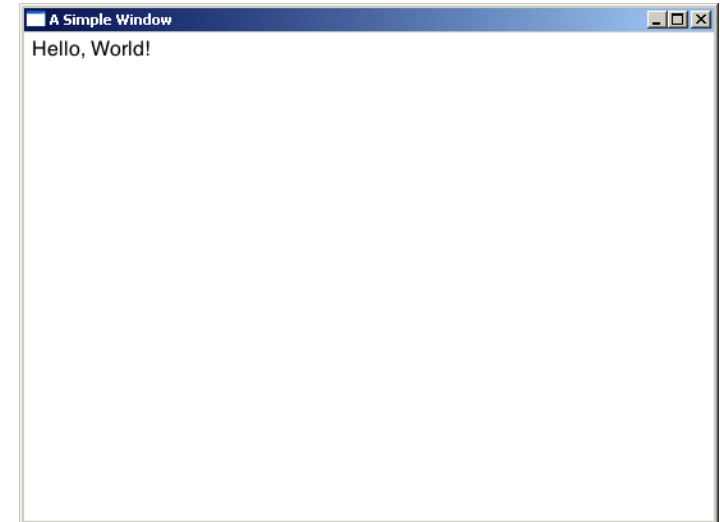
    MSG msg;
    while (GetMessage(&msg, nullptr, 0, 0)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }

    return (int)msg.wParam;
}
```

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam) {
    HDC hdc;
    PAINTSTRUCT ps;

    switch (message) {
        case WM_DESTROY:
            PostQuitMessage(0);
            break;
        case WM_PAINT:
            hdc = BeginPaint(hWnd, &ps);
            TextOut(hdc, 5, 5, L"Hello, world!", 13);
            EndPaint(hWnd, &ps);
            break;
        default:
            return DefWindowProc(hWnd, message, wParam, lParam);
    }

    return 0;
}
```



# Код для программирования GUI: X Window System

```
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/Xos.h>
#include <stdio.h>
#include <string.h>

#define WIDTH_MIN 50
#define HEIGHT_MIN 50
#define BORDER_WIDTH 5
#define TITLE "Example"
#define ICON_TITLE "Example"
#define PRG_CLASS "Example"

void main(int argc, char *argv[]) {
    Display * display;
    int screen_number;
    GC gc;
    XEvent report;
    Window window;

    if ((display = XOpenDisplay(NULL)) == NULL) {
        puts("Can not connect to the X server!\n"); exit(1);
    }
    screen_number = DefaultScreen(display);

    window = XCreateSimpleWindow(display,
        RootWindow(display, screen_number),
        X, Y, WIDTH, HEIGHT, BORDER_WIDTH,
        BlackPixel(display, screen_number),
        WhitePixel(display, screen_number));

    SetWindowManagerHints(display, PRG_CLASS, argv, argc,
        window, 0, 0, 400, 200, WIDTH_MIN,
        HEIGHT_MIN, TITLE, ICON_TITLE, 0);

    XSelectInput(display, window, ExposureMask | KeyPressMask);
    XMapWindow(display, window);

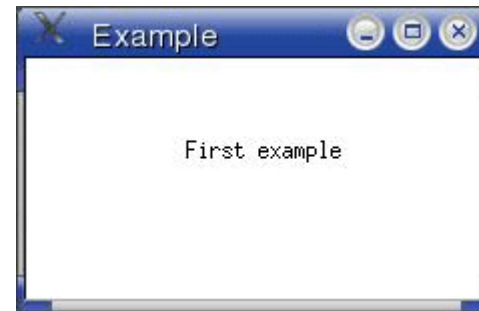
    while (1) {
        XNextEvent(display, &report);

        switch (report.type) {
            case Expose:
                if (report.xexpose.count != 0)
                    break;

                gc = XCreateGC(display, window, 0, NULL);

                XSetForeground(display, gc, BlackPixel(display, 0));
                XDrawString(display, window, gc, 20, 50, "First example", strlen("First example"));
                XFreeGC(display, gc);
                XFlush(display);
                break;

            case KeyPress:
                XCloseDisplay(display);
                exit(0);
        }
    }
}
```



## Библиотеки для GUI в XWS: GIMP ToolKit – GTK

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

int main( int argc, char *argv[]){
    GtkWidget *label;
    GtkWidget *window;

    gtk_init(&argc, &argv);

    window = gtk_window_new(GTK_WINDOW_TOPLEVEL);

    gtk_window_set_title(GTK_WINDOW(window), "Hello world!");

    label = gtk_label_new("Hello world!");

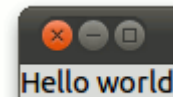
    gtk_container_add(GTK_CONTAINER(window), label);

    gtk_widget_show_all(window);

    g_signal_connect(G_OBJECT(window), "destroy", G_CALLBACK(gtk_main_quit), nullptr);

    gtk_main();

    return 0;
}
```



## Основные кроссплатформенные библиотеки



Qt

1991 – начало работы  
2024 – последний релиз, 6.7.3

PyQt, QtRuby, QtJambi,  
FreePascal bindings, ...



wxWidgets

1992 – начало работы  
2024 – последний релиз, 3.2.6

wxBasic,  
wx4j,  
wxD, wxPython, wxRuby,  
wxLua, wxHaskell, wxErlang, ...



GTK

1998 – начало работы  
2021 – последний релиз, 4.16.1

C/C++, Free Pascal, Free BASIC  
Fortran, Ada  
Java  
D, R, Go, Tcl, Python, LUA, LISP,  
Haskell, ...

# Кроссплатформенная разработка ПО: Qt

```
#include <QApplication>
#include <QMainWindow>
#include <QAction>
#include <QMenuBar>
```

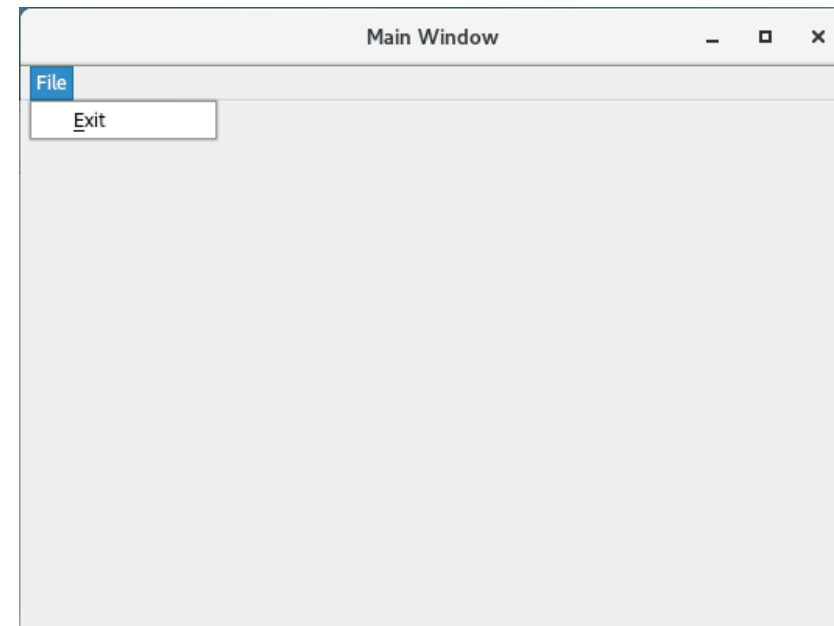
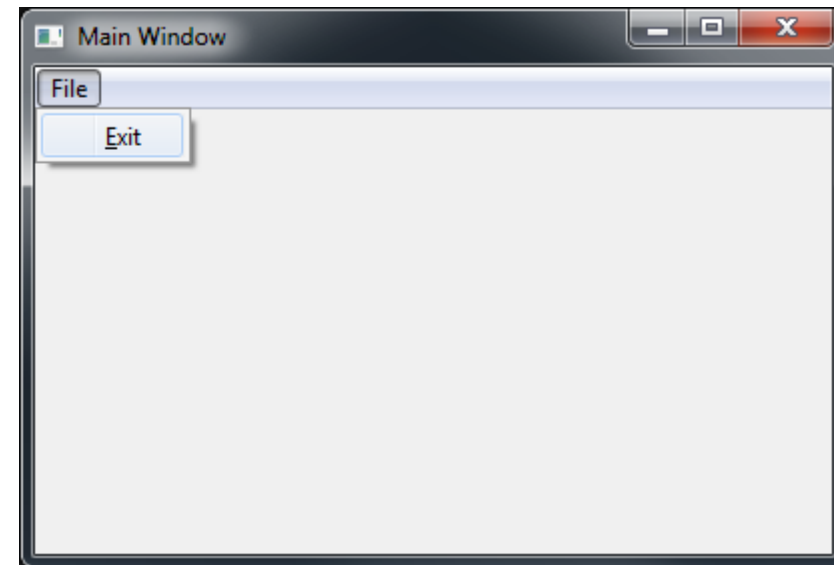
```
class MainWindow : public QMainWindow {
    Q_OBJECT
private:
    QAction *p_act_FileExit;
    QMenu *p_menu_File;
public:
    MainWindow();
private slots:
    void onActionFileExit();
};
```

```
MainWindow::MainWindow() {
    move(5, 5);
    resize(QSize(800, 600));
    setWindowTitle("Main Window");

    p_act_FileExit = new QAction(tr("&Exit"), this);
    connect(p_act_FileExit, SIGNAL(triggered()), this, SLOT(close()));

    p_menu_File = menuBar()->addMenu(tr("File"));
    p_menu_File->addAction(p_act_FileExit);
}
```

```
int main(int argc, char *argv[]) {
    QApplication app(argc, argv);
    MainWindow *p_window = new MainWindow;
    p_window->show();
    return app.exec();
}
```



# Кроссплатформенная разработка ПО: wxWidgets

```
#include <wx/app.h>
#include <wx/frame.h>
#include <wx/menu.h>

class MyFrame : public wxFrame {
public:
    MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size);
private:
    void OnExit(wxCommandEvent& event);
};

MyFrame::MyFrame(const wxString& title, const wxPoint& pos, const wxSize& size) :
    wxFrame(NULL, wxID_ANY, title, pos, size) {

    wxMenuBar *p_menuBar = new wxMenuBar;
    SetMenuBar(p_menuBar);

    wxMenu *p_menuFile = new wxMenu;
    p_menuBar->Append(p_menuFile, wxT("File"));
    p_menuFile->Append(wxID_HIGHEST + 1, wxT("Quit"));

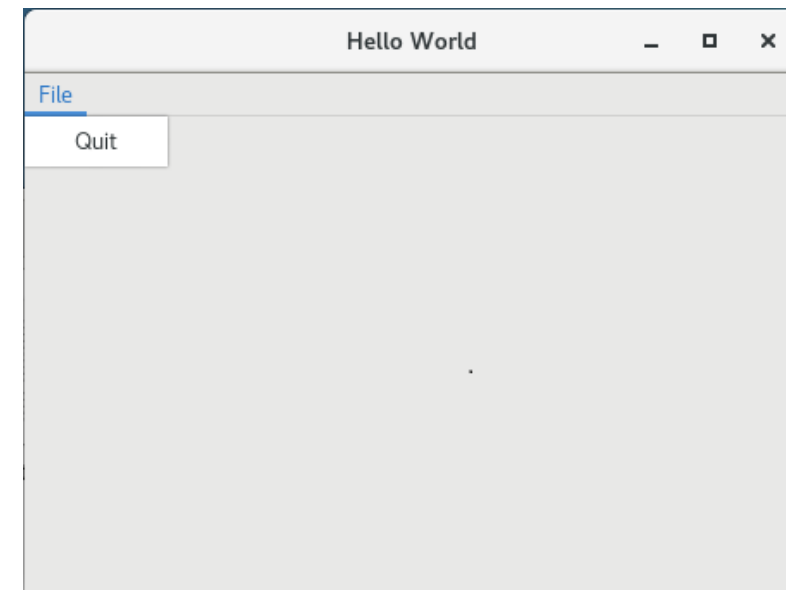
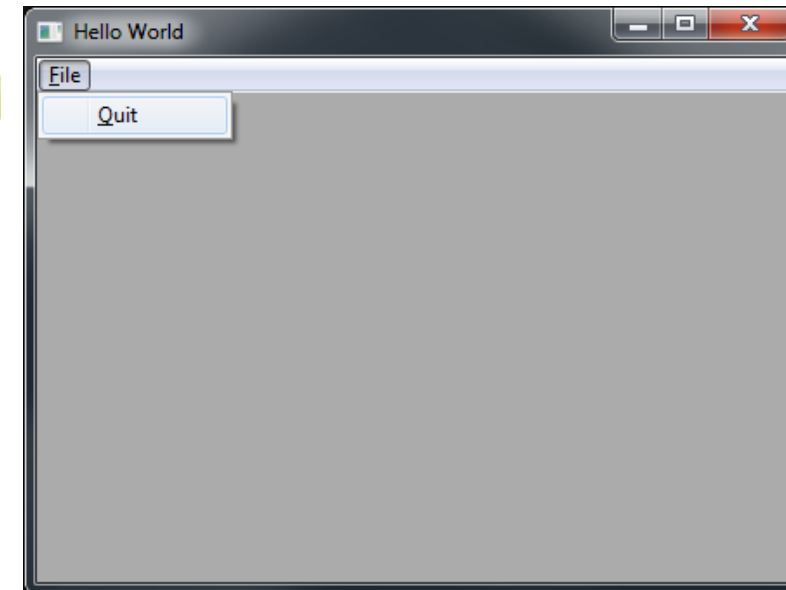
    Connect(wxID_HIGHEST + 1, wxEVT_COMMAND_MENU_SELECTED, wxCommandEventHandler(MyFrame::OnExit));
}

void MyFrame::OnExit(wxCommandEvent& event) {
    Close(true);
}

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

bool MyApp::OnInit() {
    MyFrame *frame = new MyFrame("Hello World", wxPoint(50, 50), wxSize(450, 340));
    frame->Show(true);
    return true;
}

wxDECLARE_APP(MyApp);
wxIMPLEMENT_APP(MyApp);
```





# Минимальный каркас оконной программы под WinAPI: WinMain

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

#define windowClass L"win32app"
#define windowTitle L"Win32API - Пример 1"

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

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

    if (!RegisterClassEx(&wcx)) {
        MessageBox(nullptr, L"Can't register window class!", L"Win32 API Test", MB_OK);
        return 1;
    }

    HWND hWnd = CreateWindow(szWindowClass, windowTitle, WS_OVERLAPPEDWINDOW,
        CW_USEDEFAULT, CW_USEDEFAULT, 500, 400, nullptr, nullptr,
        hInstance, nullptr);

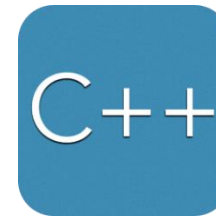
    if (!hWnd) {
        MessageBox(nullptr, L"Can't create window!", L"Win32 API Test", MB_OK);
        return 1;
    }

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

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

    return msg.wParam;
}
```

+ оконная процедура



Task\_01

Инициализация  
параметров создаваемого  
окна

Регистрация окна с нашими  
параметрами в системе

Создание экземпляра окна

Показать окно

Запуск обработчика  
событий



## Минимальный каркас оконной программы под WinAPI: WinProc

```
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;  
}
```



## Примеры сообщений ОС

```
/*  
 * Window Messages  
 */  
  
#define WM_NULL 0x0000  
#define WM_CREATE 0x0001  
#define WM_DESTROY 0x0002  
#define WM_MOVE 0x0003  
#define WM_SIZE 0x0005  
  
...  
  
#define WM_NCCREATE 0x0081  
#define WM_NCDESTROY 0x0082  
#define WM_NCCALCSIZE 0x0083  
#define WM_NCHITTEST 0x0084  
#define WM_NCPAINT 0x0085  
  
...  
  
#define WM_PAINT 0x000F  
#define WM_CLOSE 0x0010  
#define WM_QUERYENDSESSION 0x0011
```

### Клавиатура

```
WM_KEYDOWN  
WM_CHAR  
WM_KEYUP
```

### Отрисовка

```
WM_NCPAINT  
WM_ERASEBCKGND  
WM_PAINT
```

### Мышь

```
WM_LBUTTONDOWN  
WM_LBUTTONUP  
WM_LBUTTONDBLCLICK  
WM_LBUTTONUP
```



## Примеры сообщений ОС

C++

```
BOOL WINAPI MoveWindow(  
    HWND hWnd,  
    int X,  
    int Y,  
    int nWidth,  
    int nHeight,  
    BOOL bRepaint  
);
```

Функция посылает целевому окну следующую последовательность сообщений :

- |                      |   |  |
|----------------------|---|--|
| WM_WINDOWPOSCHANGING | - | начали двигать окно                          |
| WM_WINDOWPOSCHANGED  | - | закончили двигать окно                       |
| WM_MOVE              | - | окно было передвинуто                        |
| WM_SIZE              | - | окно получило новый размер                   |
| WM_NCCALCSIZE        | - | нужно рассчитать размер неклиентской области |

Если при создании окна в структуре WNDCLASSEX был установлен флаг  
`wcex.style = CS_HREDRAW | CS_VREDRAW;`  
то досылается ещё и сообщение WM\_PAINT

## Определение координат клика мышкой

C++

```
#define WM_LBUTTONDOWN 0x0201
```

### Значение параметра wParam:

```
MK_CONTROL  
MK_SHIFT
```

### Значение параметра lParam:

Координаты передаются в виде  
двойного слова, вытащить их можно с  
помощью макросов:

```
LOWORD, HIWORD
```

или

```
GET_X_PARAM, GET_Y_PARAM
```

## Работа с GDI

```
LRESULT CALLBACK WndProc(HWND hWnd,
                          UINT message,
                          WPARAM wParam,
                          LPARAM lParam) {

    PAINTSTRUCT ps;
    HDC hdc;

    switch (message) {

        case WM_PAINT:
            hdc = BeginPaint(hWnd, &ps);

            ...

            TextOut(hdc, 10, 10, "Hello, World!", strlen("Hello, World!"));

            ...

            EndPaint(hWnd, &ps);

    break;
```



Task\_02



## ВЫЗОВ КОНТЕКСТНОГО МЕНЮ

```
case WM_RBUTTONDOWN: {
    HMENU popup = CreatePopupMenu();

    AppendMenu(popup, MF_STRING, 0, L"Exit");

    POINT point = { LOWORD(lParam), HIWORD(lParam) };

    ClientToScreen(hWnd, &point);

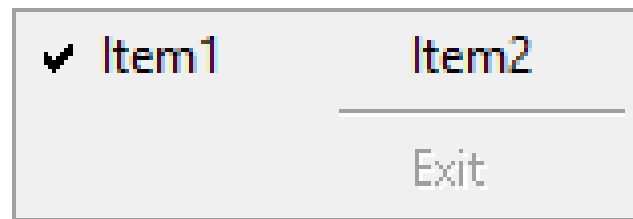
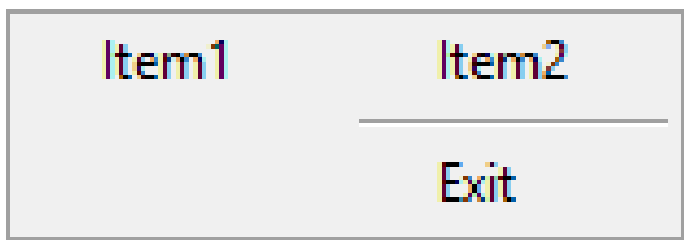
    TrackPopupMenu(popup,
        TPM_LEFTBUTTON,
        point.x, point.y,
        0,
        hWnd,
        nullptr);

    DestroyMenu(popup);
    break;
}
```



## Варианты вызова функции AppendMenu

```
AppendMenu(hwnd, MF_STRING, 0, L"Item1");  
AppendMenu(hwnd, MF_MENUBREAK, 0, L"Item2");  
AppendMenu(hwnd, MF_SEPARATOR, 0, nullptr);  
AppendMenu(hwnd, MF_STRING, 0, L"Exit");
```

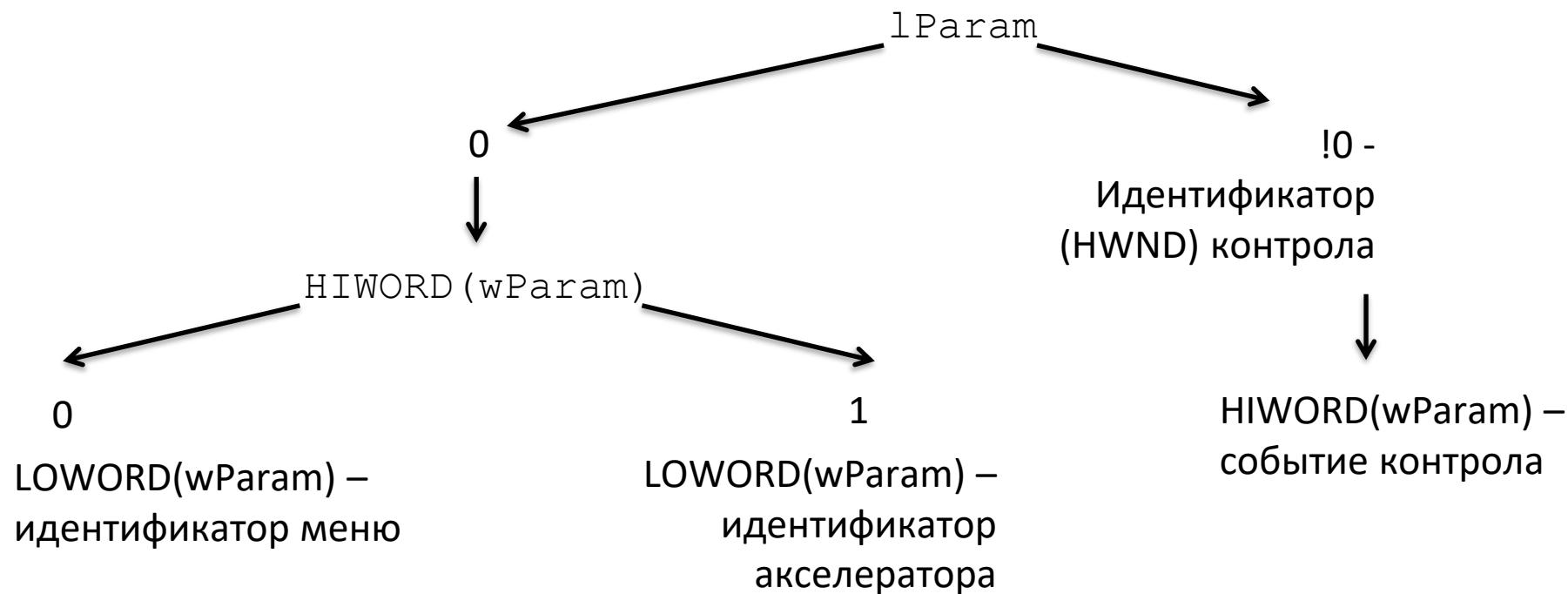


```
AppendMenu(hwnd, MF_STRING | MF_CHECKED, 0, L"Item1");  
AppendMenu(hwnd, MF_MENUBREAK, 0, L"Item2");  
AppendMenu(hwnd, MF_SEPARATOR, 0, nullptr);  
AppendMenu(hwnd, MF_STRING | MF_GRAYED, 0, L"Exit");
```



## Обработка событий меню: сообщение WM\_COMMAND

```
C++  
  
#define WM_COMMAND          0x0111
```





## Написание обработчиков событий меню

```
AppendMenu(hwnd, MF_STRING, 4131, L"Exit");
```

```
case WM_COMMAND:  
    if (LOWORD(wParam) == 4131)  
        SendMessage(hwnd, WM_CLOSE, 0, 0);  
    break;
```



## Функции отправки сообщений

### SendMessage

Блокирующий вызов

```
LRESULT SendMessage (  
    HWND    hWnd,  
    UINT    Msg,  
    WPARAM wParam,  
    LPARAM lParam  
);
```

### PostMessage

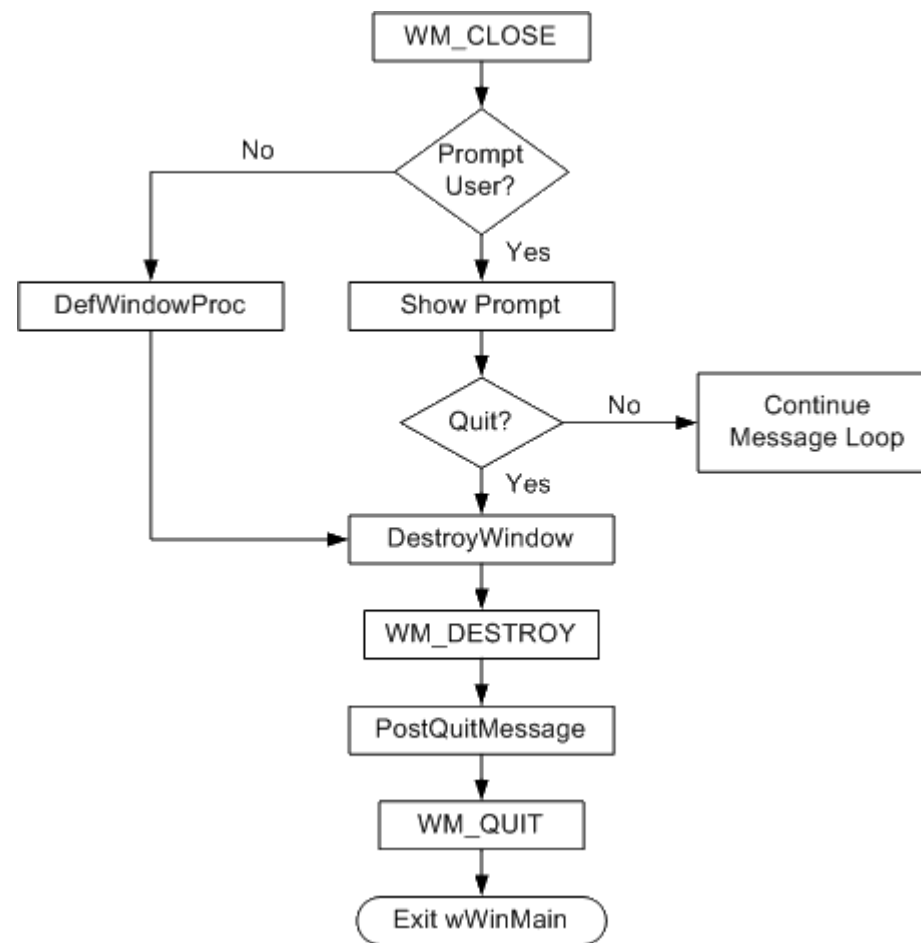
Неблокирующий вызов

```
LRESULT PostMessage (  
    HWND    hWnd,  
    UINT    Msg,  
    WPARAM wParam,  
    LPARAM lParam  
);
```

## Последовательность сообщений для закрытия окна

C++

```
#define WM_CLOSE 0x0010
```



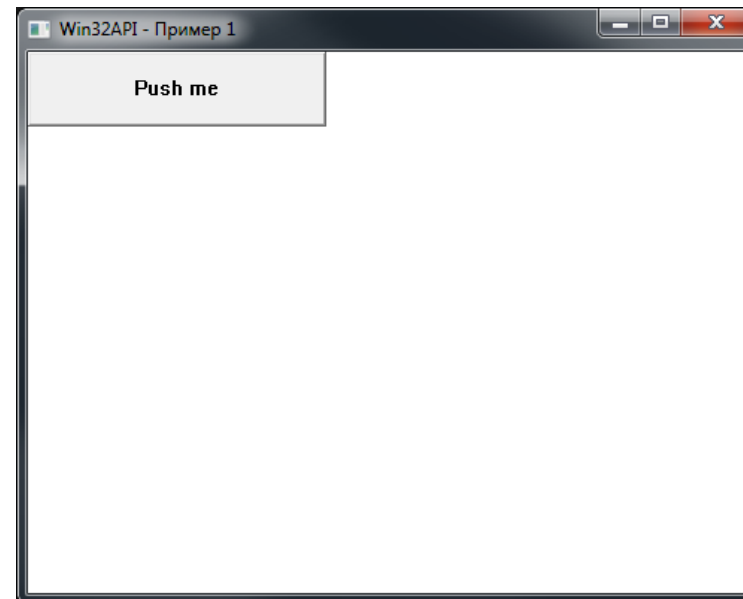
## Создание элементов управления

```
HWND hBtn;
```

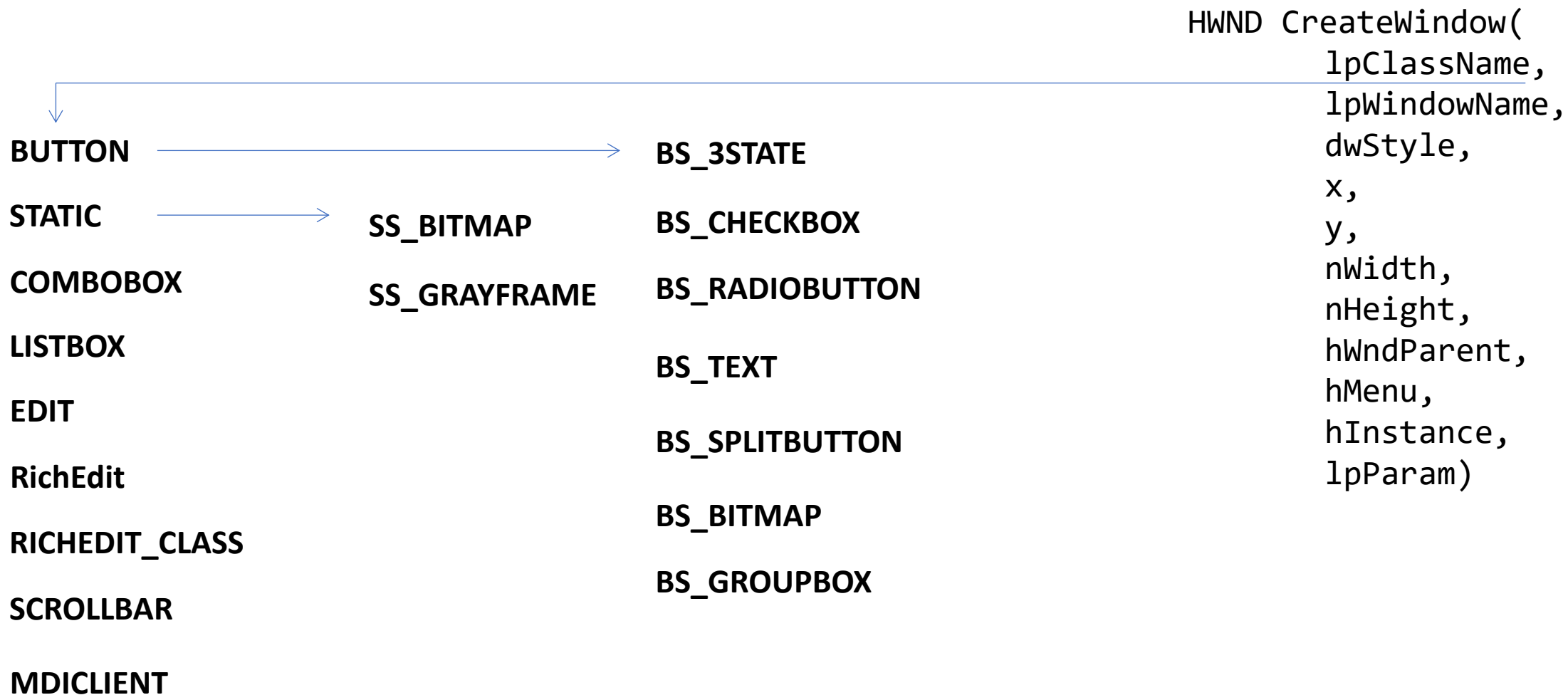
```
int __stdcall WinMain(...
```

```
...
```

```
hBtn = CreateWindow(L"button", L"Push me", WS_CHILD | WS_VISIBLE,  
0, 0, 200, 50, hWnd, nullptr, hInstance, nullptr);
```



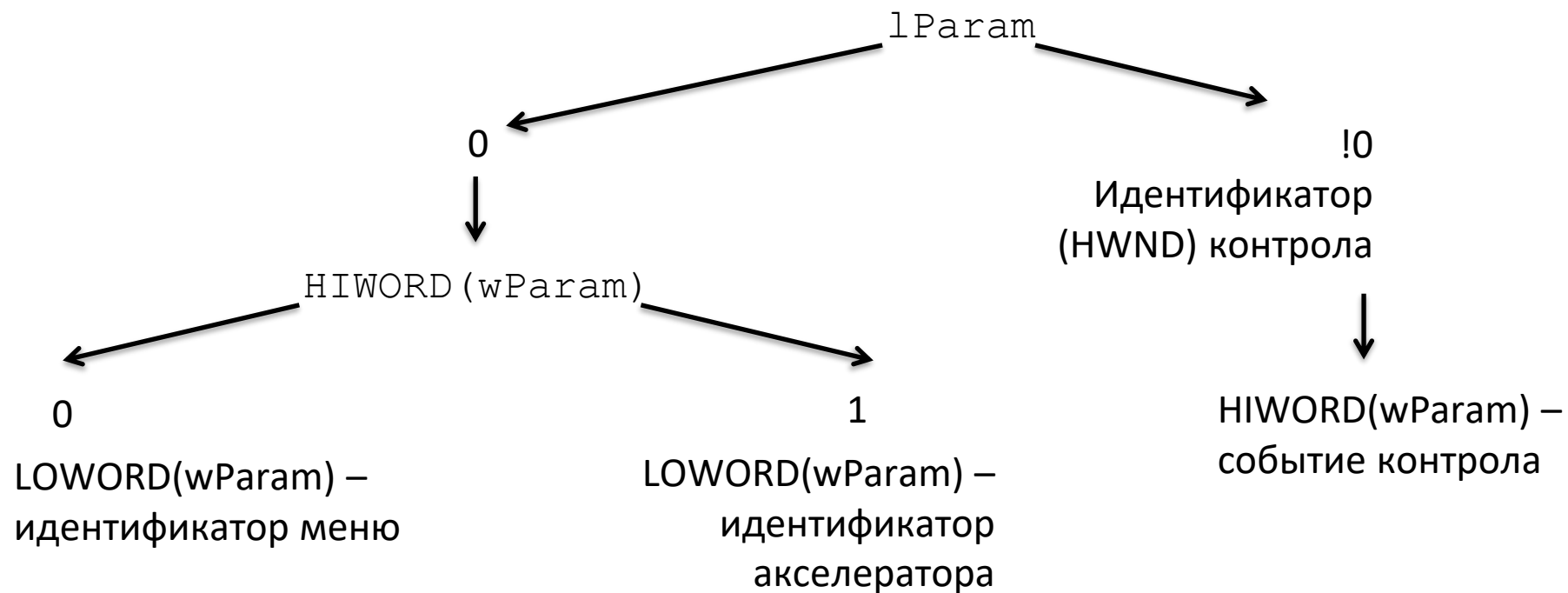
## Основные элементы управления Win32



## Обработка событий меню: сообщение WM\_COMMAND

C++

```
#define WM_COMMAND 0x0111
```



## Обработка событий от элементов управления (1)

```
HWND hBtn;
```

```
...
```

```
hBtn = CreateWindow(L"button", L"Push me", WS_CHILD | WS_VISIBLE,  
                  0, 0, 200, 50, hWnd, NULL,  
                  hInstance, NULL);
```

```
...
```

```
case WM_COMMAND:  
    if (lParam != 0) {  
        if ((HWND)lParam == hBtn)  
            if (HIWORD(wParam) == BN_CLICKED)  
                MessageBox(hWnd, L"Button clicked!", L"Cap", MB_OK);  
    }  
break;
```



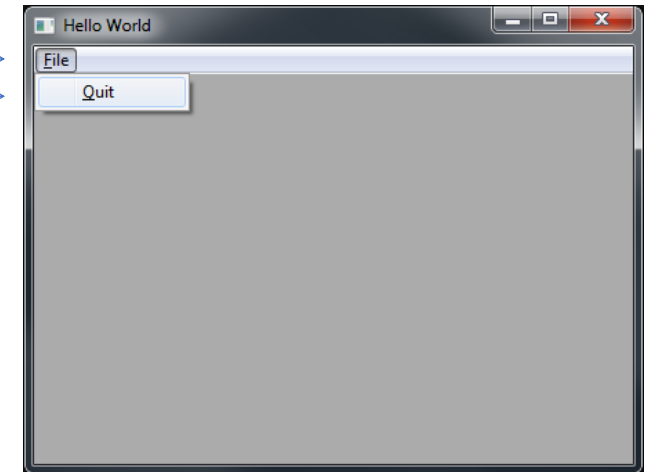


## Пример отправки сообщения окну

```
long __stdcall WndProc(HWND hWnd,  
                        UINT message,  
                        WPARAM wParam,  
                        LPARAM lParam) {  
    switch (message) {  
        ...  
        case WM_KEYDOWN:  
            if (wParam == ' ')  
                SendMessage(hWnd, WM_SYSCOMMAND, SC_MAXIMIZE, 0);  
            break;  
        ...  
    }
```

## Реализация главного меню окна

```
switch (message) {  
    case WM_CREATE:  
        HMENU hMenubar = CreateMenu();  
        HMENU hMenu     = CreateMenu();
```



```
    AppendMenu(hMenu, MF_STRING, 4131, L"&Quit");
```

```
    AppendMenu(hMenubar, MF_POPUP, (UINT_PTR)hMenu, L"&File");  
    SetMenu(hWnd, hMenubar);
```

```
break;
```

## Использование файла ресурсов (1)

Файл - Resource.h:

```
#define IDR_MYMENU 101
#define IDI_MYICON 201

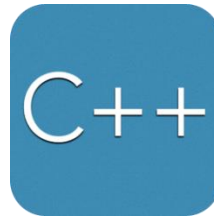
#define IDM_FILE_OPEN 9001
#define IDM_FILE_EXIT 9002
```

Файл - Resource.rc:

```
#include "resource.h"

IDR_MYMENU MENU
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&Open", IDM_FILE_OPEN
        MENUITEM "E&xit", IDM_FILE_EXIT
    END
END

IDI_MYICON ICON "menu_one.ico"
```



Task\_03

## Использование файлов ресурсов (2)

```
#include "resource.h"
```

```
...
```

```
WNDCLASSEX wcx;
```

```
...
```

```
wcx.lpszMenuName = MAKEINTRESOURCE(IDR_MYMENU);
```

## Использование диалога открытия файлов

```
OPENFILENAME ofn;  
char szFileName[MAX_PATH] = "";  
ZeroMemory(&ofn, sizeof(ofn));  
ofn.lStructSize = sizeof(ofn);  
ofn.hwndOwner = hWnd;  
ofn.lpstrFilter = "Text Files (*.txt)\0*.txt\0All Files (*.*)\0*.*\0";  
ofn.lpstrFile = szFileName;  
ofn.nMaxFile = MAX_PATH;  
ofn.Flags = OFN_EXPLORER | OFN_FILEMUSTEXIST | OFN_HIDEREADONLY;  
ofn.lpstrDefExt = "txt";  
if (GetOpenFileName(&ofn))  
    MessageBox(hWnd, ofn.lpstrFile, "Info", MB_OK);  
}
```

Результат возвращается в `ofn.lpstrFile`. Именно этот файл и нужно открывать и считывать из него данные. В данном примере я вывожу имя файла в качестве текста сообщения в `MessageBox`.

## Задание на дом

Написать программу с использованием WinAPI, которая рисует топологию из загружаемого файла (формат файла – на следующем слайде).

В программе должны быть пункты меню File и Draw.

В меню File есть пункты:

- «Open...» - загрузить файл с топологией (с вызовом диалога)
- «Save As...» - сохранить файл с топологией (с вызовом диалога)
- Разделитель
- «Exit»

В меню Draw есть пункты:

- «Metal» – рисуется металл (синий цвет)
- «Poly» – рисуется поликремний (красный цвет)

По умолчанию для рисования выбран металл.

Мышкой можно рисовать прямоугольники выбранного типа вещества.  
Нажали, провели, отпустили – нарисовалась область.

## Формат файла

Формат файла – упрощённая версия формата файла программы Microwind:

```
REC 16 3 2 14 POLY
REC 52 2 2 19 POLY
REC 16 21 38 3 POLY
REC 26 24 6 3 POLY
REC 16 17 3 4 POLY
REC 19 12 6 3 METAL
REC 45 11 6 9 METAL
REC 19 5 7 7 METAL
REC 44 4 7 7 METAL
REC 61 4 6 6 METAL
REC 55 0 6 36 METAL
REC 9 0 6 37 METAL
REC 25 21 7 7 METAL
```

