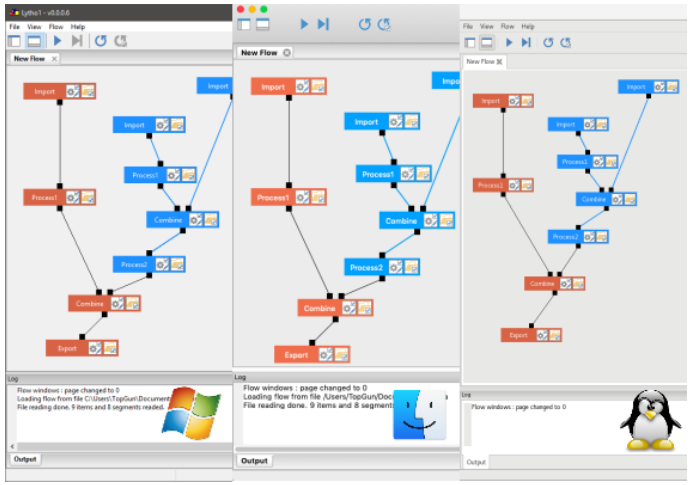
МИЭТ
Национальный исследовательский университет «МИЭТ»
Институт интегральной электроники (группы ИВТ-15М, ИВТ-16М, каф. ПКИМС)

Кроссплатформенная разработка программного обеспечения



Лабораторная работа №1

Нативные API. WinAPI

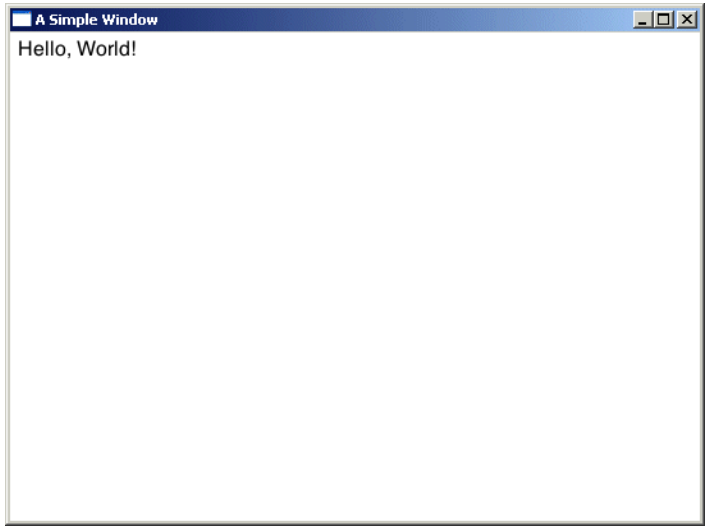
Кроссплатформенный код: соглашение POSIX

POSIX (***p**ortable **o**perating **s**ystem **i**nterface)* — 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

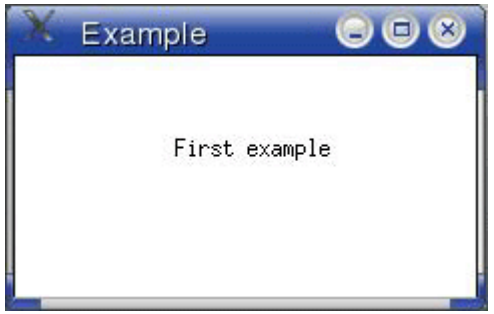
```
#include <windows.h>

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

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

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

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

  if (!RegisterClassEx(&wcex)) {
    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;
}
```

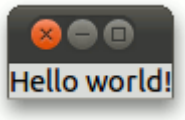
Код для программирования GUI: X Window System

```
#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



```c
#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, ...

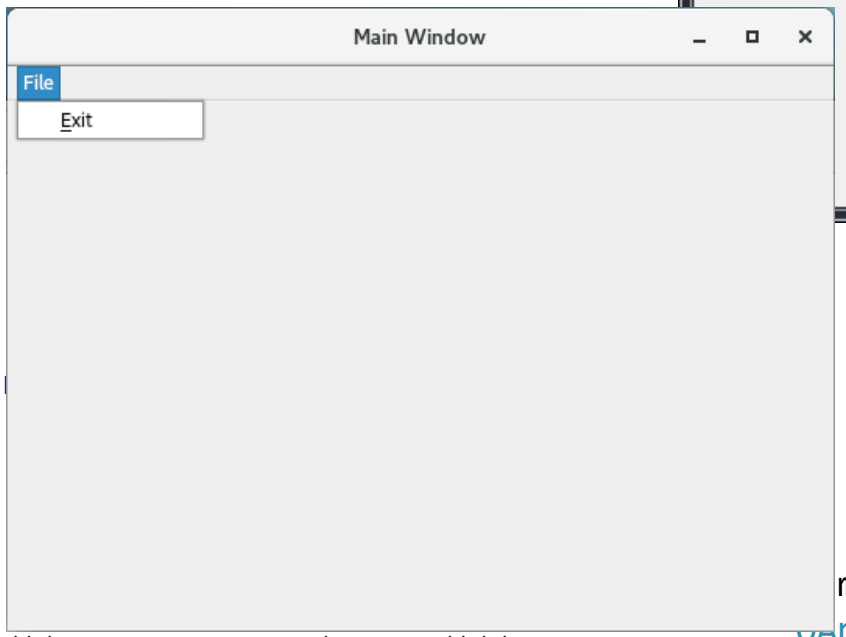
GTK

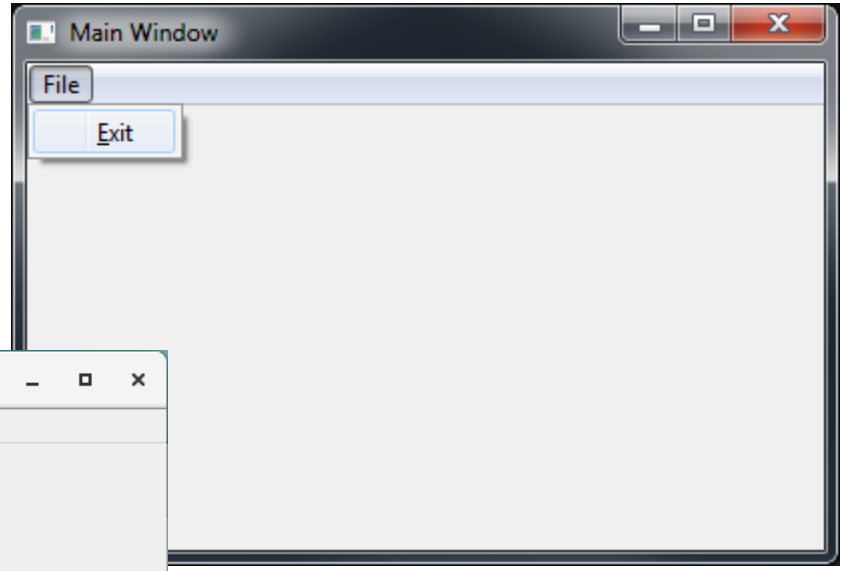
1998 – начало работы
2021 – последний релиз, 4.16.1

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  *actFileExit;
  QMenu    *menuFile;
public:
  MainWindow();
private slots:
  void onActionFileExit();
};

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

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

  menuFile = menuBar()->addMenu(tr("File"));
  menuFile->addAction(actFileExit);
}
```

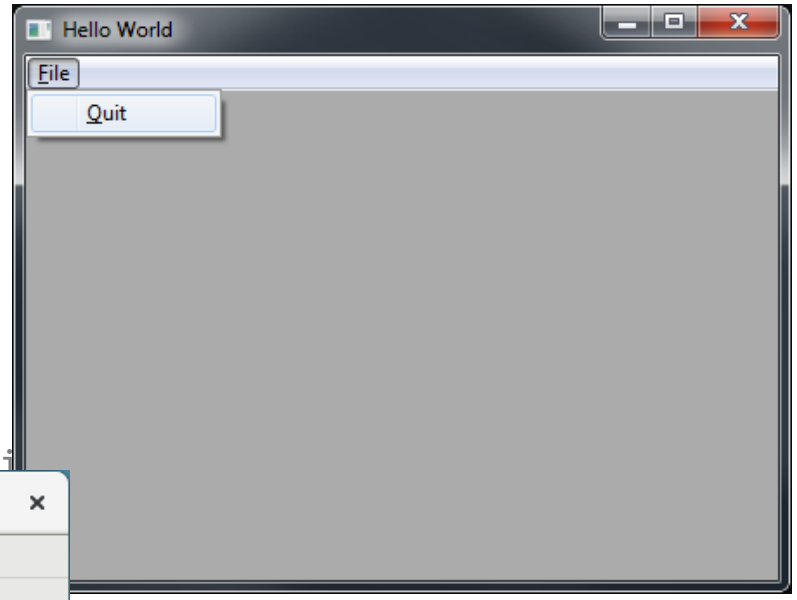


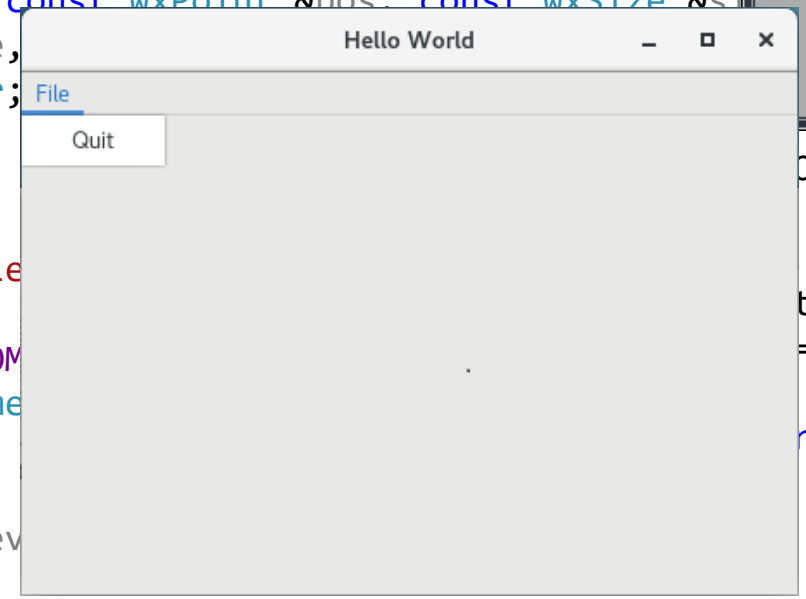



```
main(int argc, char *argv[]) {
Application app(argc, argv);
MainWindow *window = new MainWindow;
window->show();
return app.exec();
}
```

Кроссплатформенная разработка ПО: wxWidgets

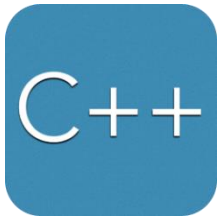
```
#include <wx/app.h>
#include <wx/frame.h>
#include <wx/menu.h>

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

Frame::Frame(const wxString &title, const wxPoint &pos, const wxSize &si
  : wxFrame(nullptr, wxID_ANY, title,
  wxMenuBar *menuBar = new wxMenuBar;
  SetMenuBar(menuBar);

  wxMenu *menuFile = new wxMenu;
  menuBar->Append(menuFile, wxT("File
  menuFile->Append(wxID_HIGHEST + 1,
  Connect(wxID_HIGHEST + 1, wxEVT_COM
          wxCommandEventHandler(Frame
}

void Frame::OnExit(wxCommandEvent &ev
  Close(true);
}
```






```
OnInit();

t() {
= new Frame("Hello World", wxPoint(50, 50),
                           wxSize(450, 340));
rue);

wxDECLARE_APP(App);
wxIMPLEMENT_APP(App);
```

Минимальный каркас оконной программы под WinAPI: WinMain



Task_01

```
#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 wcex;

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

  if (!RegisterClassEx(&wcex)) {
    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;
}
```

Инициализация параметров создаваемого окна

Регистрация окна с нашими параметрами в системе

Создание экземпляра окна

Показать окно

Запуск обработчика событий

\+ оконная процедура

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
```

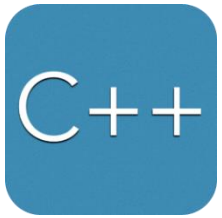
Значение параметра wParam:

MK_CONTROL
MK_SHIFT

Значение параметра lParam:

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

Работа с GDI



Task_02

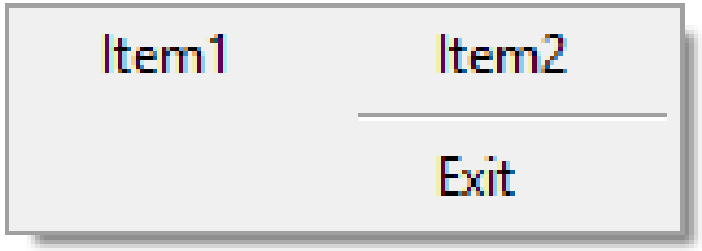
```
LRESULT CALLBACK WndProc(HWND hWnd,
                         UINT message,
                         WPARAM wParam,
                         LPARAM lParam) {
  PAINTSTRUCT ps;
  HDC hdc;

  switch (message) {

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

      ...

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

      ...

      EndPaint(hWnd, &ps);

    break;
```

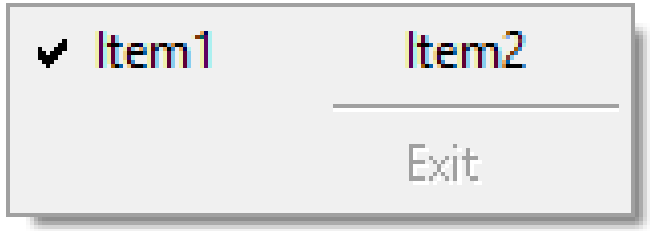
Вызов контекстного меню

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

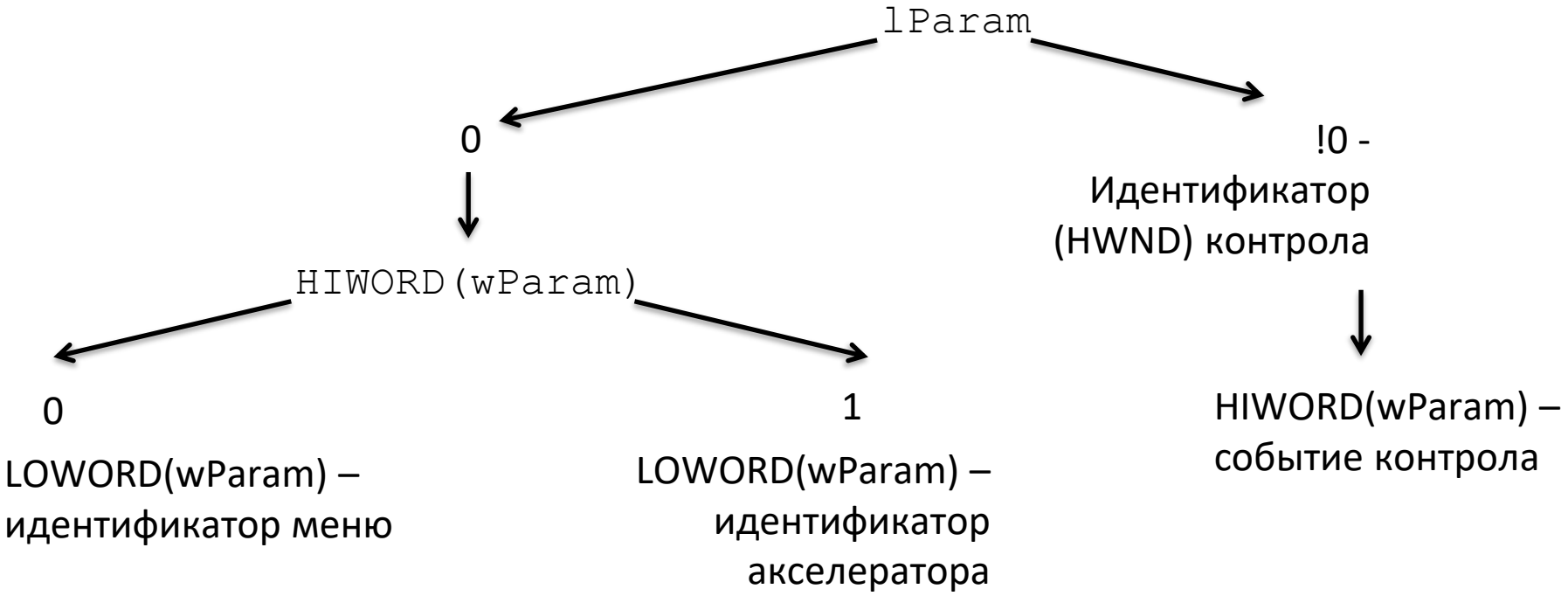





```
AppendMenu(popup, MF_STRING | MF_CHECKED, 0, L"Item1");
AppendMenu(popup, MF_MENUBREAK,           0, L"Item2");
AppendMenu(popup, MF_SEPARATOR,           0, nullptr);
AppendMenu(popup, MF_STRING | MF_GRAYED,  0, L"Exit");
```

Обработка событий меню: сообщение WM_COMMAND

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



Написание обработчиков событий меню

```
AppendMenu(popup, 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

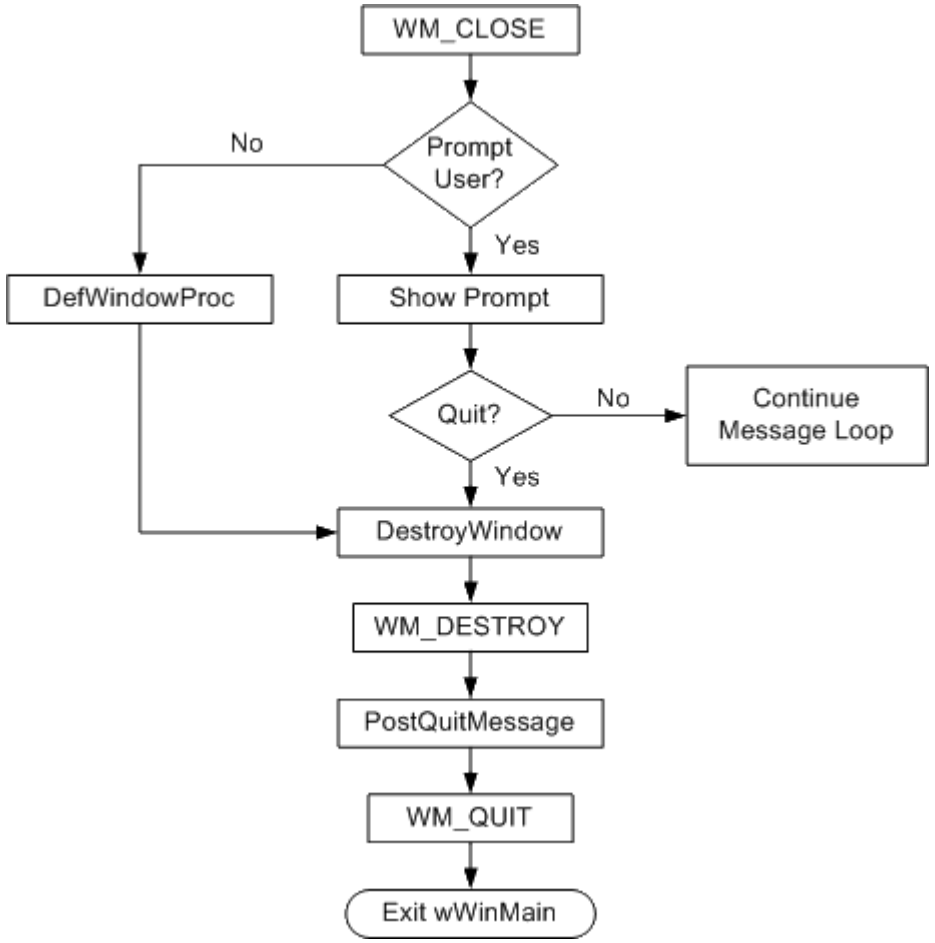
Неблокирующий вызов

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

Последовательность сообщений для закрытия окна

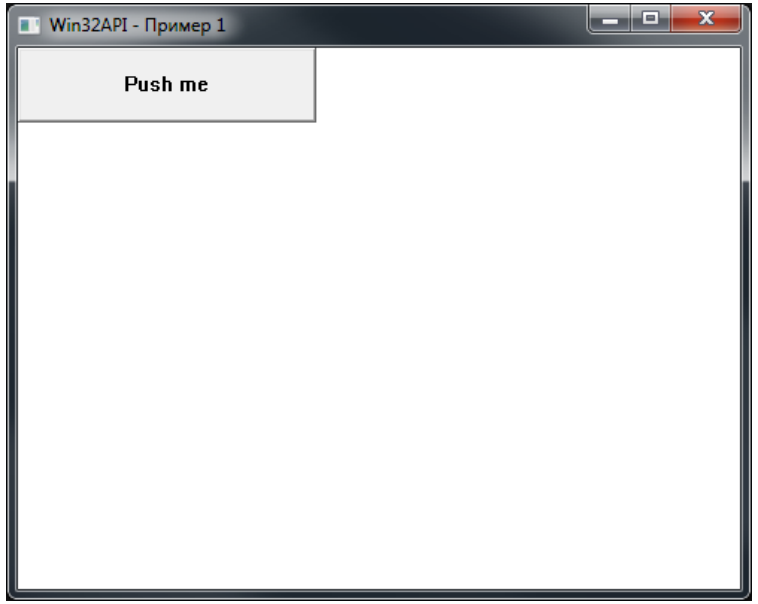
C++

```
#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

```
HWND CreateWindow(
        lpClassName,
        lpWindowName,
        dwStyle,
        x,
        y,
        nWidth,
        nHeight,
        hWndParent,
        hMenu,
        hInstance,
        lpParam)
```

BUTTON → **BS_3STATE**, **BS_CHECKBOX**, **BS_RADIOBUTTON**, **BS_TEXT**, **BS_SPLITBUTTON**, **BS_BITMAP**, **BS_GROUPBOX**

STATIC → **SS_BITMAP**, **SS_GRAYFRAME**

COMBOBOX

LISTBOX

EDIT

RichEdit

RICHEDIT_CLASS

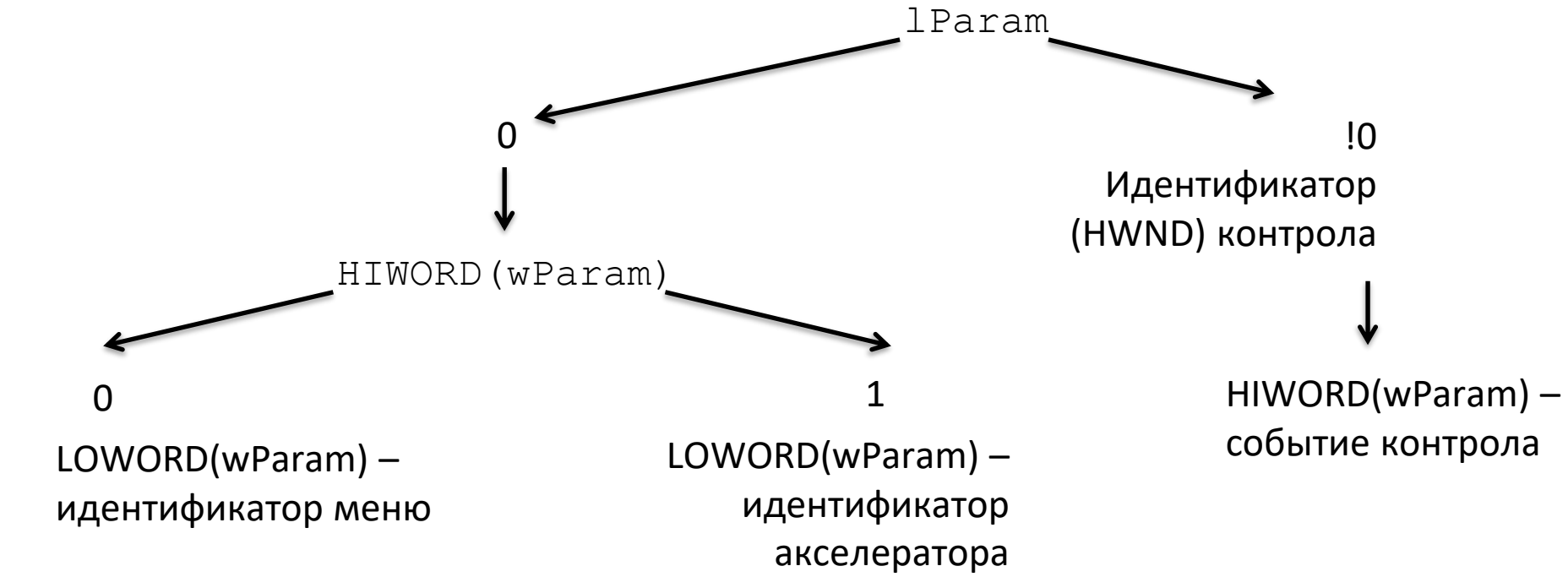
SCROLLBAR

MDICLIENT

Обработка событий меню: сообщение WM_COMMAND

C++

```
#define WM_COMMAND                      0x0111
```



Обработка событий от элементов управления

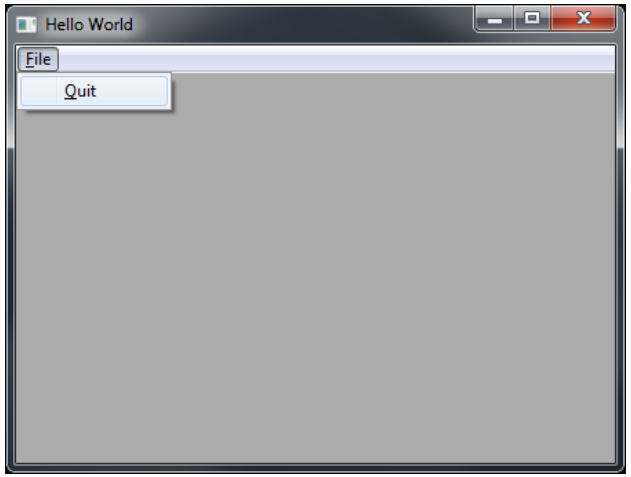
```
HWND hBtn;
...

 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;
    ...
```

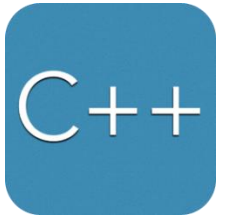
Реализация главного меню окна

```
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)



Task_03

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"
```

Использование файлов ресурсов (2)

```
#include "resource.h"

...

WNDCLASSEX wcex;

...

wcex.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 фала в качестве текста сообщения в MessageBox.

Задание на дом

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

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.

Так же, как и в программе MicroWind, при открытии файла топология должна рисоваться в полный масштаб.

Формат файла

Формат файла – упрощённая версия формата файла программы MicroWind MSK:

```
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
```