**Создание приложений Win32 (C++)**

Win32 API (также известный как Windows API) представляет собой платформу для создания приложений Windows, основанную на языке C. Эта платформа используется начиная с Windows 1.0. Подробную документацию по этому API можно найти по адресу [Windows API](http://msdn.microsoft.com/ru-ru/library/cc433218(v=vs.90).aspx).

В рамках данной процедуры создается простейшее приложений Win32, отображающее приветствие "Hello, World!" в отдельном окне. При создании любых приложений Win32 порядок действий одинаков. По завершении выполнения процедуры получившийся код можно использовать в качестве основы для создания любых других приложений Win32.

[Обязательные компоненты](javascript:void(0))

Этот раздел предполагает знание основ языка C++. Если вы приступаете к изучению C++, рекомендуем вам "Руководство по С++ для начинающих", написанное Гербом Шилдтом (Herb Schildt), которое доступно по ссылке <http://go.microsoft.com/fwlink/?LinkId=115303>.

ссылка на видеоДля просмотра видеодемонстрации см. [Видео: создание приложения Win32 (C++)](http://go.microsoft.com/fwlink/?LinkId=102471).

**Создание проекта Win32**

1. В меню Файл последовательно выберите команды Создать и Проект....
2. В узле Visual C++ области Типы проектов выберите CLR, а затем в области Шаблоны выберите Проект Win32.
3. Введите имя проекта, например win32app. Можно принять место размещения по умолчанию, ввести необходимое место размещения или перейти к каталогу, в который требуется сохранить проект.
4. В окне Мастер приложений Win32 нажмите кнопку Далее.
5. В окне Мастер приложений Win32 в поле Тип приложения выберите пункт Приложение Windows. В поле Дополнительные параметры выберите Пустой проект. Остальные параметры оставьте без изменений. Чтобы создать проект, нажмите кнопку Готово.
6. Добавьте в проект файл C++, в меню Проект выбрав команду Добавить новый элемент.... В диалоговом окне Добавление нового элемента выберите Файл C++ (.cpp). Введите имя файла, например GT\_HelloWorldWin32.cpp, и нажмите кнопку Добавить.

**Запуск приложения Win32**

1. Как известно, любое приложение на языках C и C++ должно иметь функцию **main**. Эта функция является начальной точкой для приложения. Подобным же образом любое приложение Win32 должно иметь функцию **WinMain**. Синтаксис функции **WinMain** выглядит следующим образом:

[Копировать](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_90d33807-1f14-4b3e-b30f-8514538e8c73');" \o "Копировать в буфер обмена.)

int WINAPI WinMain(HINSTANCE hInstance,

HINSTANCE hPrevInstance,

LPSTR lpCmdLine,

int nCmdShow);

Описание параметров и значений, возвращаемых этой функцией, см. в разделе [Функция WinMain](http://msdn.microsoft.com/ru-ru/library/ms633559(v=vs.90).aspx).

1. Наряду с функцией **WinMain**, в каждом приложении Win32 также должна быть определена еще одна функция, обычно называемая **WndProc** и представляющая собой оконную процедуру. Синтаксис функции **WndProc** выглядит следующим образом:

[Копировать](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_05bd46d6-99aa-44ab-94a8-874824f86b7b');)

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

Назначением этой функции является обработка сообщений, получаемых приложением от операционной системы. В каком случае приложение получает сообщения от операционной системы? Оно получает их постоянно! Например, представим, что было создано диалоговое окно с кнопкой ОК. Когда пользователь нажимает кнопку, операционная система посылает приложению сообщение, оповещающее о нажатии кнопки. Функция **WndProc** отвечает за реагирование на это событие. В этом примере соответствующей реакцией на это событие может быть закрытие диалогового окна.

Дополнительные сведения см. в разделе [Процедуры окна](http://msdn.microsoft.com/ru-ru/library/ms632593(v=vs.90).aspx).

**Расширение функциональности WinMain**

1. Для начала создайте внутри функции **WinMain** структуру класса окна типа [WNDCLASSEX](http://msdn.microsoft.com/ru-ru/library/ms633577(v=vs.90).aspx). Эта структура содержит информацию об окне, такую как используемые в приложении значки, цвет фона окна, отображаемое в заголовке окна название, имя функции процедуры окна и т.д. Типичная структура **WNDCLASSEX** выглядит следующим образом:

[Копировать](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_5637ce42-a4c7-45e0-9a68-fe3d059eb8fe');)

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));

Описание полей этой структуры см. в разделе [WNDCLASSEX](http://msdn.microsoft.com/ru-ru/library/ms633577(v=vs.90).aspx).

1. После того как класс окна будет создан, необходимо зарегистрировать его. Воспользуйтесь функцией [RegisterClassEx](http://msdn.microsoft.com/ru-ru/library/ms633587(v=vs.90).aspx), которой следует передать структуру класса окна в качестве аргумента:

[Копировать](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_fa6e926f-9519-4e33-bc03-1f2c5612f1ab');)

if (!RegisterClassEx(&wcex))

{

MessageBox(NULL,

\_T("Call to RegisterClassEx failed!"),

\_T("Win32 Guided Tour"),

NULL);

return 1;

}

1. После того как класс будет зарегистрирован, можно приступать к созданию окна. Используйте функцию [CreateWindow](http://msdn.microsoft.com/ru-ru/library/ms632679(v=vs.90).aspx), как показано ниже:

[Копировать](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_28678d7e-e3cb-4d82-82a1-16b089d4f82a');)

static TCHAR szWindowClass[] = \_T("win32app");

static TCHAR szTitle[] = \_T("Win32 Guided Tour Application");

// The parameters to CreateWindow explained:

// szWindowClass: the name of the application

// szTitle: the text that appears in the title bar

// WS\_OVERLAPPEDWINDOW: the type of window to create

// CW\_USEDEFAULT, CW\_USEDEFAULT: initial position (x, y)

// 500, 100: initial size (width, length)

// NULL: the parent of this window

// NULL: this application dows not have a menu bar

// hInstance: the first parameter from WinMain

// NULL: not used in this application

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;

}

Эта функция возвращает объект HWND, являющийся дескриптором окна. Дополнительные сведения см. в разделе [Типы данных Windows](http://msdn.microsoft.com/ru-ru/library/aa383751(v=vs.90).aspx).

1. После того как окно будет создано, его можно вывести на экран с помощью следующего кода:

[Копировать](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_8f4f1c2a-67ce-4cb5-9fae-8fed3a635ee1');)

// The parameters to ShowWindow explained:

// hWnd: the value returned from CreateWindow

// nCmdShow: the fourth parameter from WinMain

ShowWindow(hWnd,

nCmdShow);

UpdateWindow(hWnd);

Это окно будет практически пустым, поскольку функция **WndProc** еще не реализована.

1. На последнем этапе в функции **WinMain** реализуется цикл обработки сообщений. Назначением этого цикла является прослушивание сообщений, посылаемых операционной системой. При получении приложением сообщения оно передается функции **WndProc** на обработку. Цикл обработки сообщений выглядит следующим образом:

[Копировать](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_54e58b75-bfae-4d29-a550-362747ab6ed0');)

MSG msg;

while (GetMessage(&msg, NULL, 0, 0))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return (int) msg.wParam;

Дополнительные сведения о структурах и функциях, используемых в цикле обработки сообщений, см. в разделах, посвященных [MSG](http://msdn.microsoft.com/ru-ru/library/ms644958(v=vs.90).aspx), [GetMessage](http://msdn.microsoft.com/ru-ru/library/ms644936(v=vs.90).aspx), [TranslateMessage](http://msdn.microsoft.com/ru-ru/library/ms644955(v=vs.90).aspx) и [DispatchMessage](http://msdn.microsoft.com/ru-ru/library/ms644934(v=vs.90).aspx).

1. Выполненные этапы типичны для создания большинства приложений Win32. На данном этапе функция **WinMain** должна выглядеть примерно следующим образом:

[Копировать](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_66e69f37-e12b-4880-a7c5-c1dbe029f854');)

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; // Store instance handle in our global variable

// The parameters to CreateWindow explained:

// szWindowClass: the name of the application

// szTitle: the text that appears in the title bar

// WS\_OVERLAPPEDWINDOW: the type of window to create

// CW\_USEDEFAULT, CW\_USEDEFAULT: initial position (x, y)

// 500, 100: initial size (width, length)

// NULL: the parent of this window

// NULL: this application dows not have a menu bar

// hInstance: the first parameter from WinMain

// NULL: not used in this application

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;

}

// The parameters to ShowWindow explained:

// hWnd: the value returned from CreateWindow

// nCmdShow: the fourth parameter from WinMain

ShowWindow(hWnd,

nCmdShow);

UpdateWindow(hWnd);

// Main message loop:

MSG msg;

while (GetMessage(&msg, NULL, 0, 0))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return (int) msg.wParam;

}

**Расширение функциональности WndProc**

1. Назначением функции **WndProc** является обработка сообщений, получаемых приложением. Обработка таких сообщений обычно реализуется путем использования функции Switch.

Для начала реализуем обработку сообщения [WM\_PAINT](http://msdn.microsoft.com/ru-ru/library/ms534901(v=vs.90).aspx). Приложение получает это сообщение, когда возникает необходимость в обновлении какой-либо области окна приложения. При создании окна данное сообщение передается для указания на необходимость обновления всего окна.

При обработке сообщения **WM\_PAINT** сперва необходимо вызвать функцию [BeginPaint](http://msdn.microsoft.com/ru-ru/library/ms534894(v=vs.90).aspx), а в завершение следует вызвать функцию [EndPaint](http://msdn.microsoft.com/ru-ru/library/ms534840(v=vs.90).aspx). Между вызовами этих двух функций обрабатывается логика по отображению текста, кнопок и других элементов управления в окне. Данное приложение отображает в окне строку "Hello, World!". Для отображения текста следует использовать функцию [TextOut](http://msdn.microsoft.com/ru-ru/library/ms534019(v=vs.90).aspx), как показано ниже:

[Копировать](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_60cb2344-495a-4cca-b044-a9feff701f20');)

PAINTSTRUCT ps;

HDC hdc;

TCHAR greeting[] = \_T("Hello, World!");

switch (message)

{

case WM\_PAINT:

hdc = BeginPaint(hWnd, &ps);

// Here your application is laid out.

// For this introduction, we just print out "Hello, World!"

// in the top left corner.

TextOut(hdc,

5, 5,

greeting, \_tcslen(greeting));

// End application-specific layout section.

EndPaint(hWnd, &ps);

break;

}

1. Обычно приложение обрабатывает множество других сообщений, таких как [WM\_CREATE](http://msdn.microsoft.com/ru-ru/library/ms632619(v=vs.90).aspx) и [WM\_DESTROY](http://msdn.microsoft.com/ru-ru/library/ms632620(v=vs.90).aspx). Ниже приведен код простой, но полноценной функции **WndProc**:

[Копировать](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_d6526337-38c9-40c6-a80b-634aed06bfeb');)

LRESULT CALLBACK 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);

// Here your application is laid out.

// For this introduction, we just print out "Hello, World!"

// in the top left corner.

TextOut(hdc,

5, 5,

greeting, \_tcslen(greeting));

// End application specific layout section.

EndPaint(hWnd, &ps);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

break;

}

return 0;

}

[Пример](javascript:void(0))

**Описание**

По завершении всех этапов код должен выглядеть следующим образом. Чтобы построить приложение, в меню Построение выберите команду Построить решение. Если компиляция приложения была выполнена без ошибок, можно запустить приложение с помощью клавиши F5. В верхнем левом углу экрана появится простое окно с текстом "Hello, World!".

**Код**

[Копировать](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_5628eb59-8981-43b7-bbe0-8ea5f876dd92');)

// GT\_HelloWorldWin32.cpp

// compile with: /D\_UNICODE /DUNICODE /DWIN32 /D\_WINDOWS /c

#include <windows.h>

#include <stdlib.h>

#include <string.h>

#include <tchar.h>

// Global variables

// The main window class name.

static TCHAR szWindowClass[] = \_T("win32app");

// The string that appears in the application's title bar.

static TCHAR szTitle[] = \_T("Win32 Guided Tour Application");

HINSTANCE hInst;

// Forward declarations of functions included in this code module:

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; // Store instance handle in our global variable

// The parameters to CreateWindow explained:

// szWindowClass: the name of the application

// szTitle: the text that appears in the title bar

// WS\_OVERLAPPEDWINDOW: the type of window to create

// CW\_USEDEFAULT, CW\_USEDEFAULT: initial position (x, y)

// 500, 100: initial size (width, length)

// NULL: the parent of this window

// NULL: this application dows not have a menu bar

// hInstance: the first parameter from WinMain

// NULL: not used in this application

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;

}

// The parameters to ShowWindow explained:

// hWnd: the value returned from CreateWindow

// nCmdShow: the fourth parameter from WinMain

ShowWindow(hWnd,

nCmdShow);

UpdateWindow(hWnd);

// Main message loop:

MSG msg;

while (GetMessage(&msg, NULL, 0, 0))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return (int) msg.wParam;

}

//

// FUNCTION: WndProc(HWND, UINT, WPARAM, LPARAM)

//

// PURPOSE: Processes messages for the main window.

//

// WM\_PAINT - Paint the main window

// WM\_DESTROY - post a quit message and return

//

//

LRESULT CALLBACK 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);

// Here your application is laid out.

// For this introduction, we just print out "Hello, World!"

// in the top left corner.

TextOut(hdc,

5, 5,

greeting, \_tcslen(greeting));

// End application-specific layout section.

EndPaint(hWnd, &ps);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

break;

}

return 0;

}