

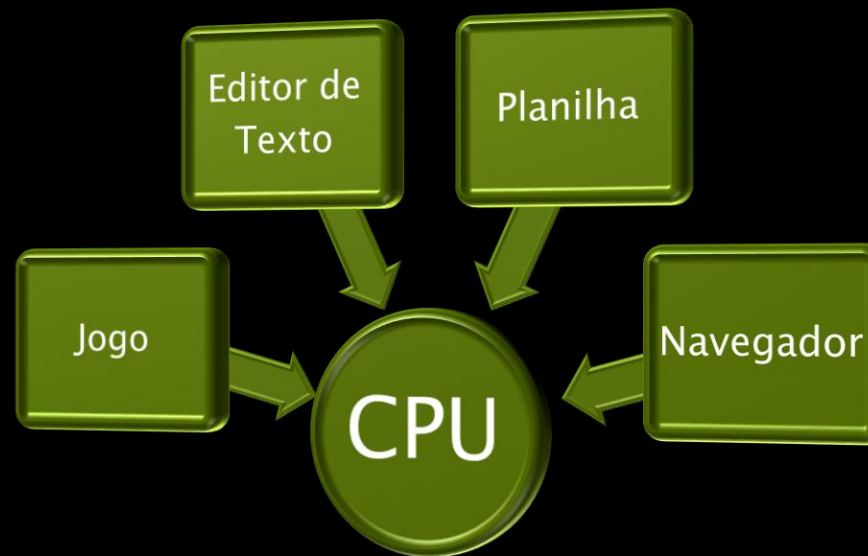
Programação Windows

Programação de Jogos

Judson Santos Santiago

Programação no Windows

- ▶ O Windows é um sistema operacional **multitarefa**
 - Vários **programas rodam concorrentemente** disputando por recursos:
 - Memória
 - Processador
 - Disco rígido
 - Acesso à periféricos:
 - Teclado
 - Mouse
 - Gamepad
 - Etc.



Programação no Windows

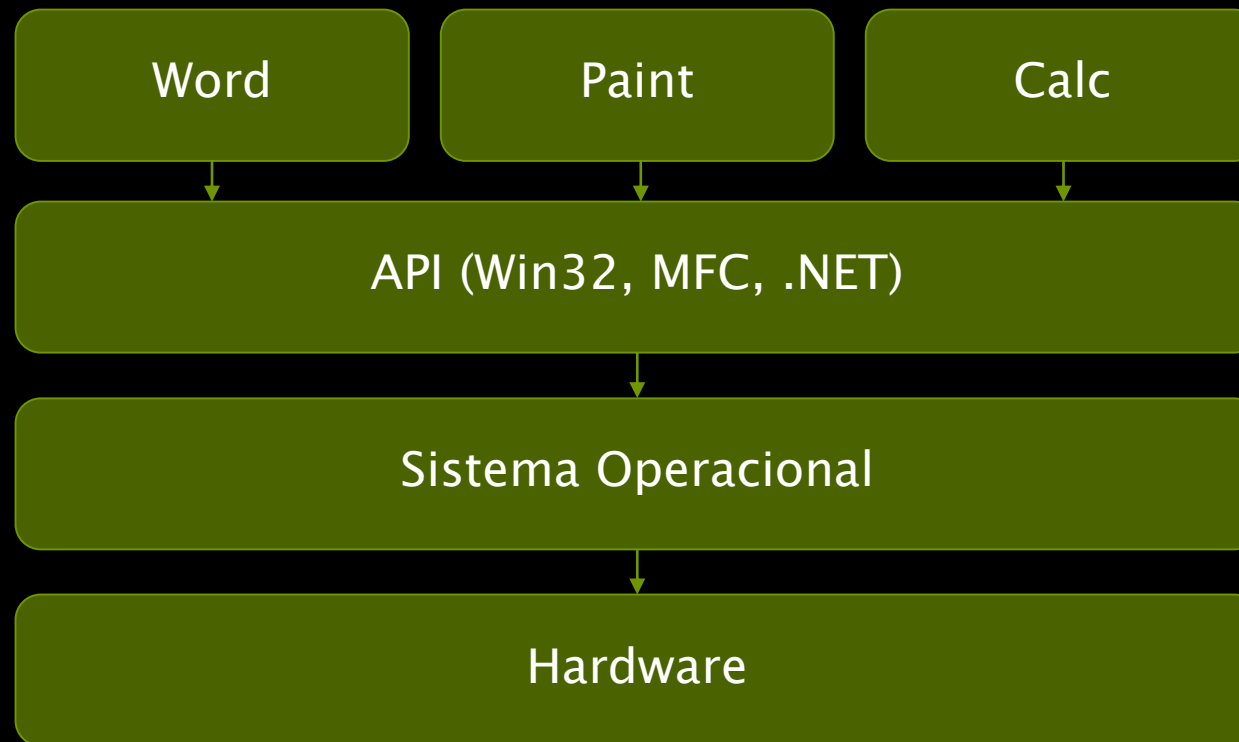
- Cada programa é executado por uma fração de segundos, fornecendo a **falsa impressão de que todos executam simultaneamente**



O sistema operacional aloca a cada processo entre 0.75 e 100 milissegundos.

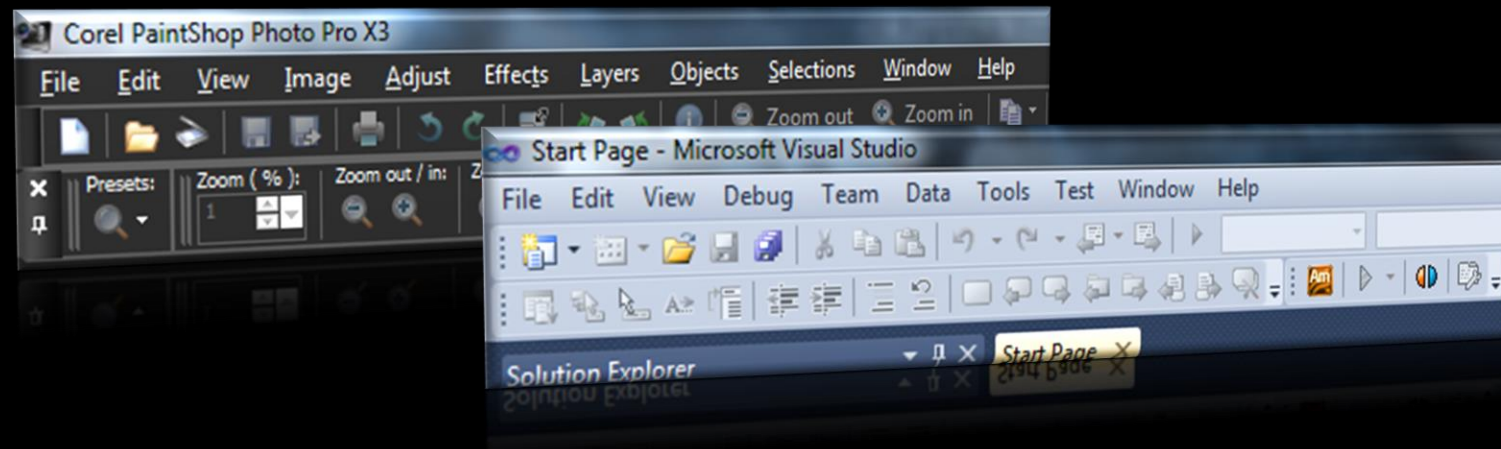
Programação no Windows

- ▶ Os programas escritos para Windows **não acessam o hardware diretamente**



Programação no Windows

- ▶ A **interface consistente** é o resultado do uso de rotinas padronizadas de uma API
 - As janelas têm a **mesma interface** porque elas são construídas pelas funções da API e não pelas aplicações
 - Estas funções padronizadas são **disponibilizadas** em arquivos DLL (Dynamic Link Libraries)



Programação no Windows

- ▶ Nas versões iniciais, a maior parte do Windows estava em três DLLs, contendo os principais subsistemas:
 - **Kernel** (kernel32.dll)
 - Gerenciamento de memória
 - Entrada/saída de arquivos
 - Escalonamento de tarefas
 - **User** (user32.dll):
 - interface do usuário
 - lógica de controle das janelas
 - **GDI** (gdi32.dll)
 - Desenho de texto e gráficos

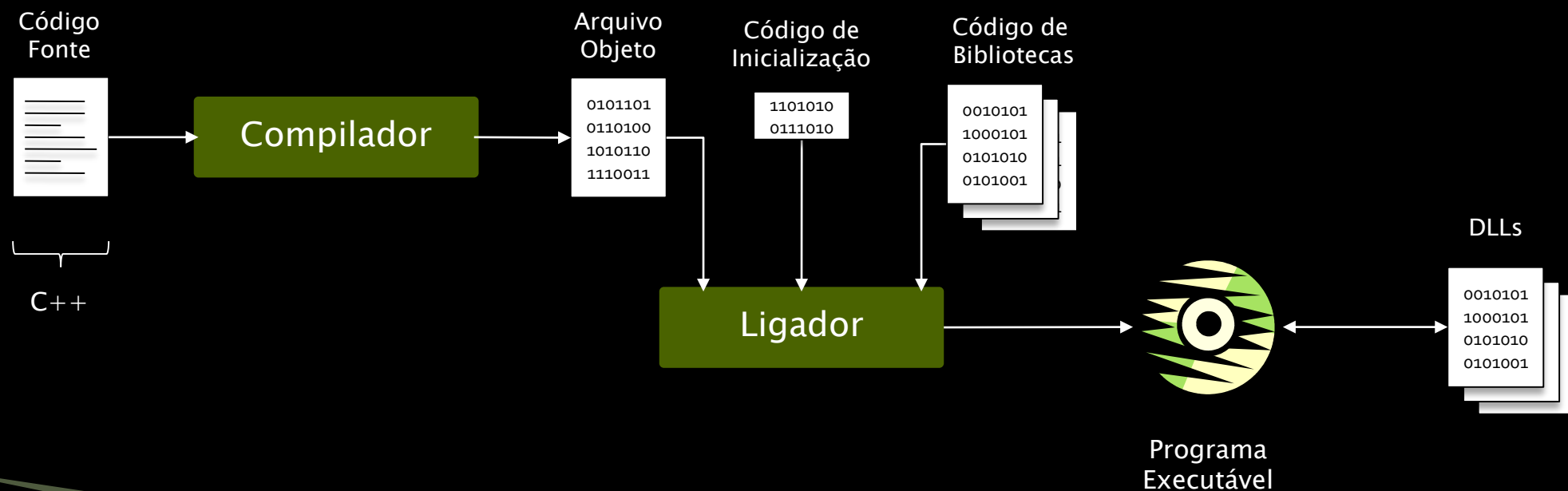


Programação no Windows

- ▶ Hoje o Windows possui **milhares de funções** disponibilizadas em suas DLLs
 - As funções são declaradas em **arquivos de cabeçalho**
Sendo “windows.h” o principal
 - A **documentação** está disponível no MSDN
Microsoft Developer Network
- ▶ O uso destas funções é semelhante ao uso das funções da biblioteca padrão da linguagem C++
Ex.: strlen

Programação no Windows

- ▶ A principal diferença entre as funções da biblioteca padrão* do C++ e as DLLs do Windows é que **as DLLs não são incluídas no código executável**



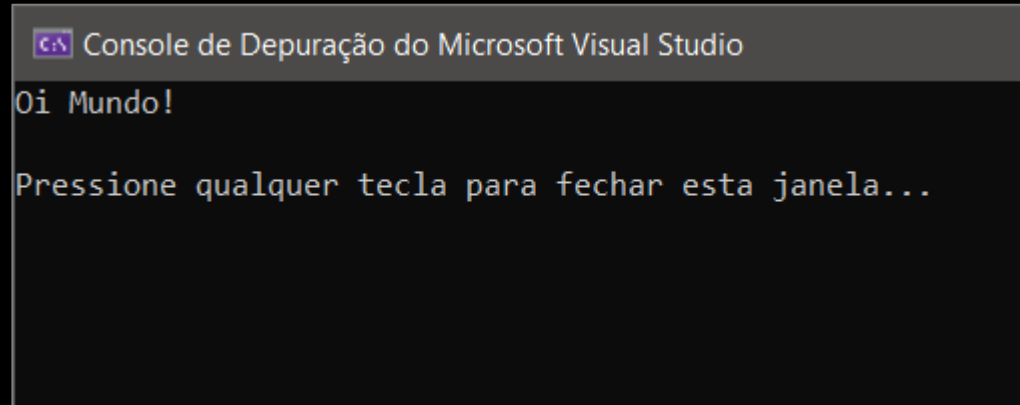
Primeiro Programa

- ▶ O primeiro programa em modo texto:

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Oi Mundo!" << endl;

    system("pause");
    return 0;
}
```



Console de Depuração do Microsoft Visual Studio

```
Oi Mundo!

Pressione qualquer tecla para fechar esta janela...
```

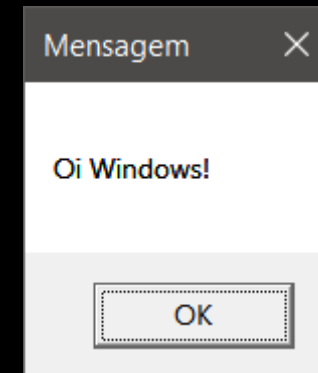
Primeiro Programa

- O primeiro programa para Windows:

```
#include <windows.h>

int APIENTRY WinMain (_In_ HINSTANCE hInstance,
                     _In_opt_ HINSTANCE hPrevInstance,
                     _In_ LPSTR lpCmdLine,
                     _In_ int nCmdShow)
{
    MessageBox(NULL, "Oi Windows!", "Mensagem", 0);

    return 0;
}
```



Criando um Projeto

Nova janela de Boas
Vindas do Visual
Studio 2019



Criar um novo projeto

Escolha um modelo de projeto com scaffolding
de código para começar

O que você deseja fazer?

Abrir recente

À medida que usar o Visual Studio, quaisquer projetos, pastas ou arquivos que você abrir aparecerão aqui para acesso rápido.

Você pode fixar tudo que abre com frequência para que fique sempre na parte superior da lista.

Introdução



Clonar ou verificar código

Obter o código de um repositório online, como o GitHub ou o Azure DevOps



Abrir um projeto ou solução

Abrir um projeto local do Visual Studio ou arquivo .sln



Abrir uma pasta local

Navegar e editar o código dentro de qualquer pasta



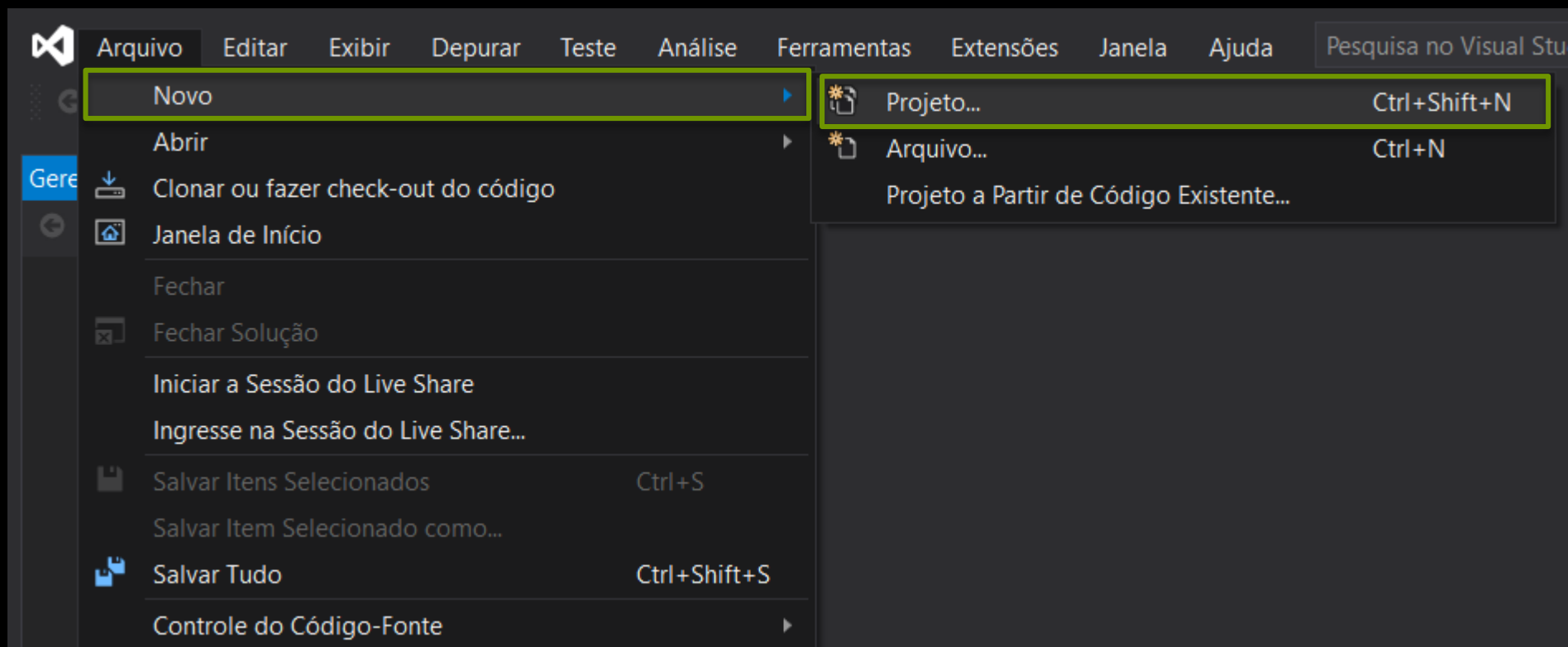
Criar um novo projeto

Escolha um modelo de projeto com scaffolding de código para começar

Criando um Projeto


Criar um projeto no Visual Studio:

Arquivo > Novo > Projeto...



Criando um Projeto

Na tela de seleção do projeto,
escolher **Assistente do
Windows Desktop**



Assistente do Windows Desktop




Crie seu próprio aplicativo do Windows usando um assistente.


C++ Windows Área de Trabalho Console Biblioteca

Criar um novo projeto

Pesquisar modelos de projeto 🔍 Idioma ▾ Plataforma ▾ Tipo de projeto ▾

Modelos de projeto recentes


-  Projeto Vazio C++
-  Aplicativo do Windows Desktop C++
-  Assistente do Windows Desktop C++



Projeto Vazio

Iniciar do zero com C++ para Windows. Não fornece nenhum arquivo inicial.


C++ Windows Console



Aplicativo de Console

Executar o código em um terminal Windows. Imprime "Olá, Mundo" por padrão.


C++ Windows Console



Assistente do Windows Desktop

Crie seu próprio aplicativo do Windows usando um assistente.


C++ Windows Área de Trabalho Console Biblioteca



Aplicativo do Windows Desktop

Um projeto para um aplicativo com uma interface gráfica do usuário que é executado no Windows.


C++ Windows Área de Trabalho



Projeto de Itens Compartilhados

Um projeto de Itens Compartilhados é usado para compartilhar arquivos entre vários projetos.

C++ Windows Android iOS Linux Área de Trabalho Console Biblioteca UWP Jogos Celular



Solução em branco

Uma solução vazia sem projetos

Voltar Próximo

Criando um Projeto

Criar um projeto usando o Assistente:
HelloWindows

Configurar seu novo projeto

Assistente do Windows Desktop C++ Windows Área de Trabalho Console Biblioteca

Nome do projeto

HelloWindows

Localização

C:\Users\Judson\Source\Repos

Nome da solução ⓘ

HelloWindows

☐ Coloque a solução e o projeto no mesmo diretório

Criando um Projeto

Criar um Projeto do Windows Desktop:

Aplicativo da Área de Trabalho (.exe) > Projeto Vazio

Tipo de aplicativo:

Aplicativo de Console (.exe)
Aplicativo da Área de Trabalho (.exe)
Biblioteca de Vínculo Dinâmico (.dll)
Biblioteca Estática (.lib)

Projeto do Windows Desktop

Tipo de aplicativo:

Aplicativo da Área de Trabalho (.e

Opções Adicionais:

☒ Projeto Vazio

☐ Cabeçalho Pré-compilado

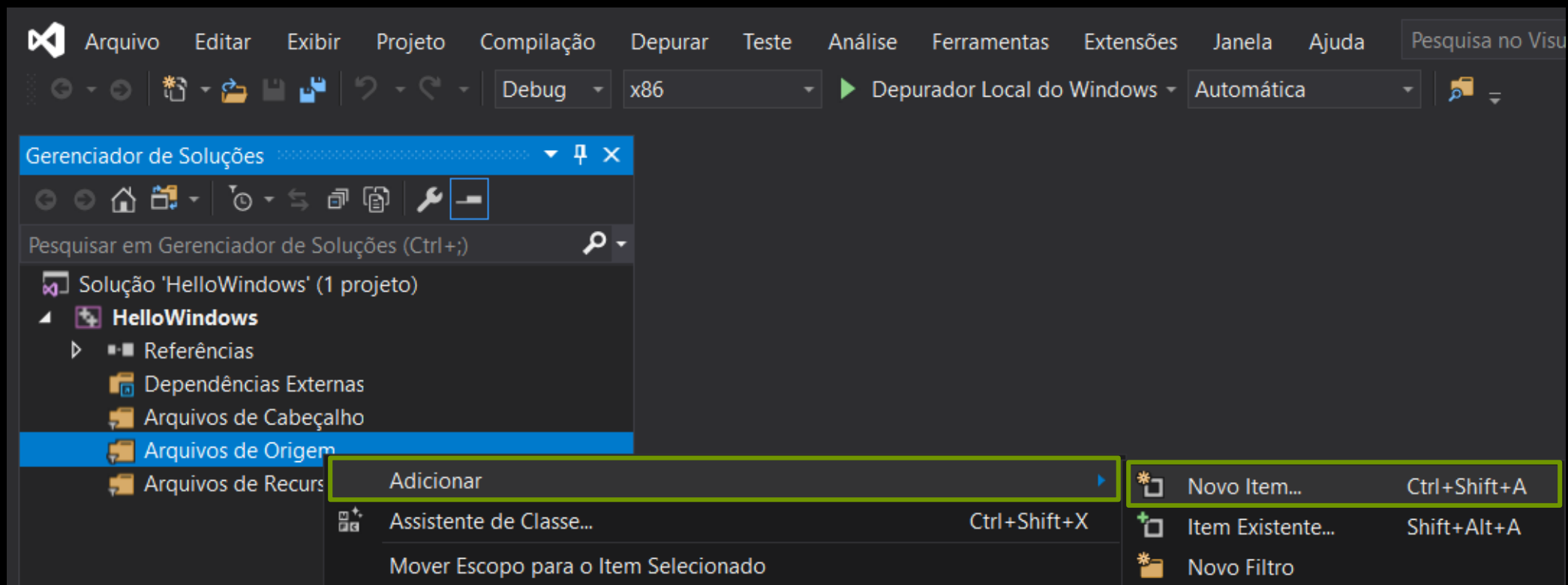
☐ Exportar Símbolos

☐ Cabeçalhos MFC

Criando um Programa

Inserir um novo arquivo fonte no projeto

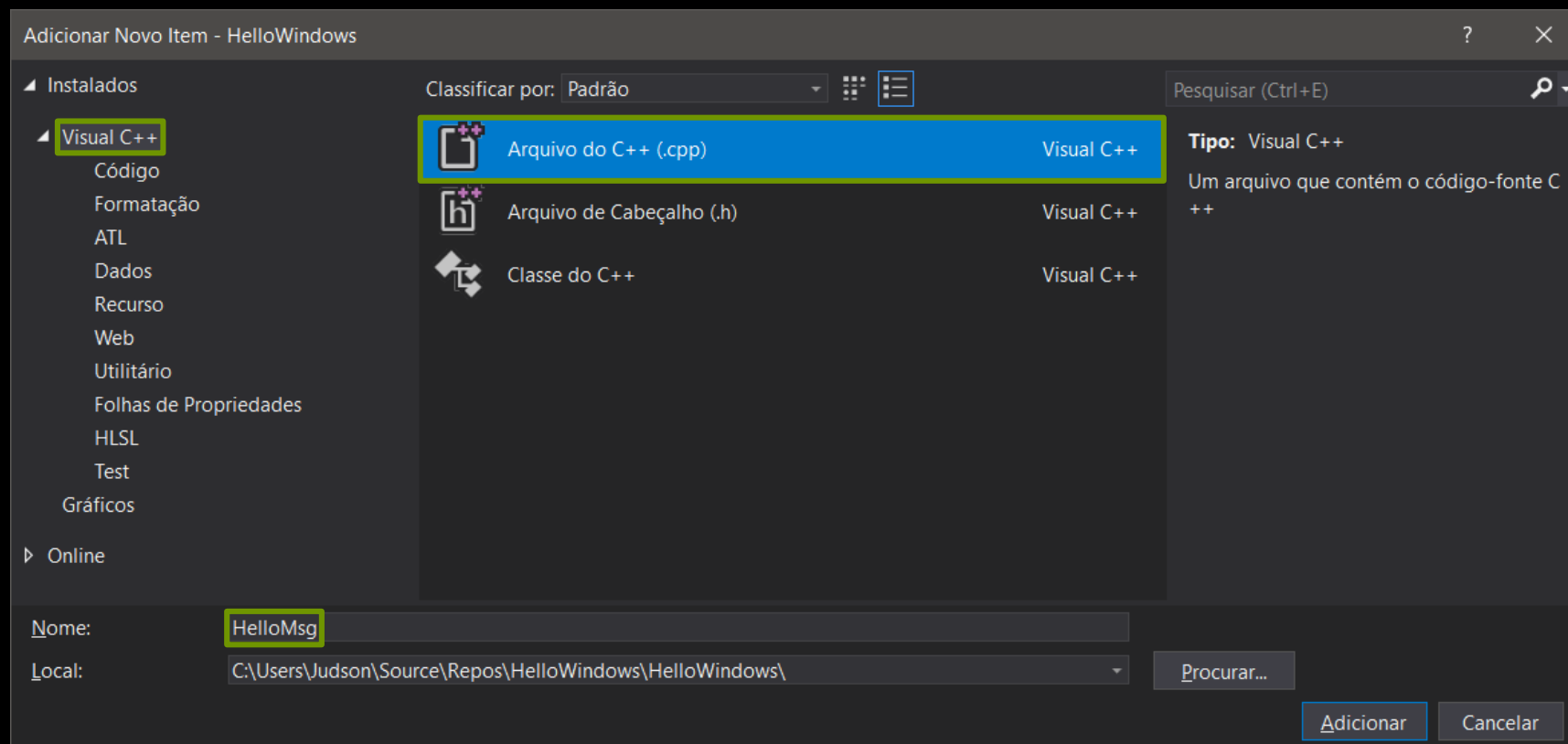
Clicar Botão Direito em Arquivos de Origem > Adicionar > Novo Item...



Criando um Programa

Inserir um novo arquivo fonte no projeto

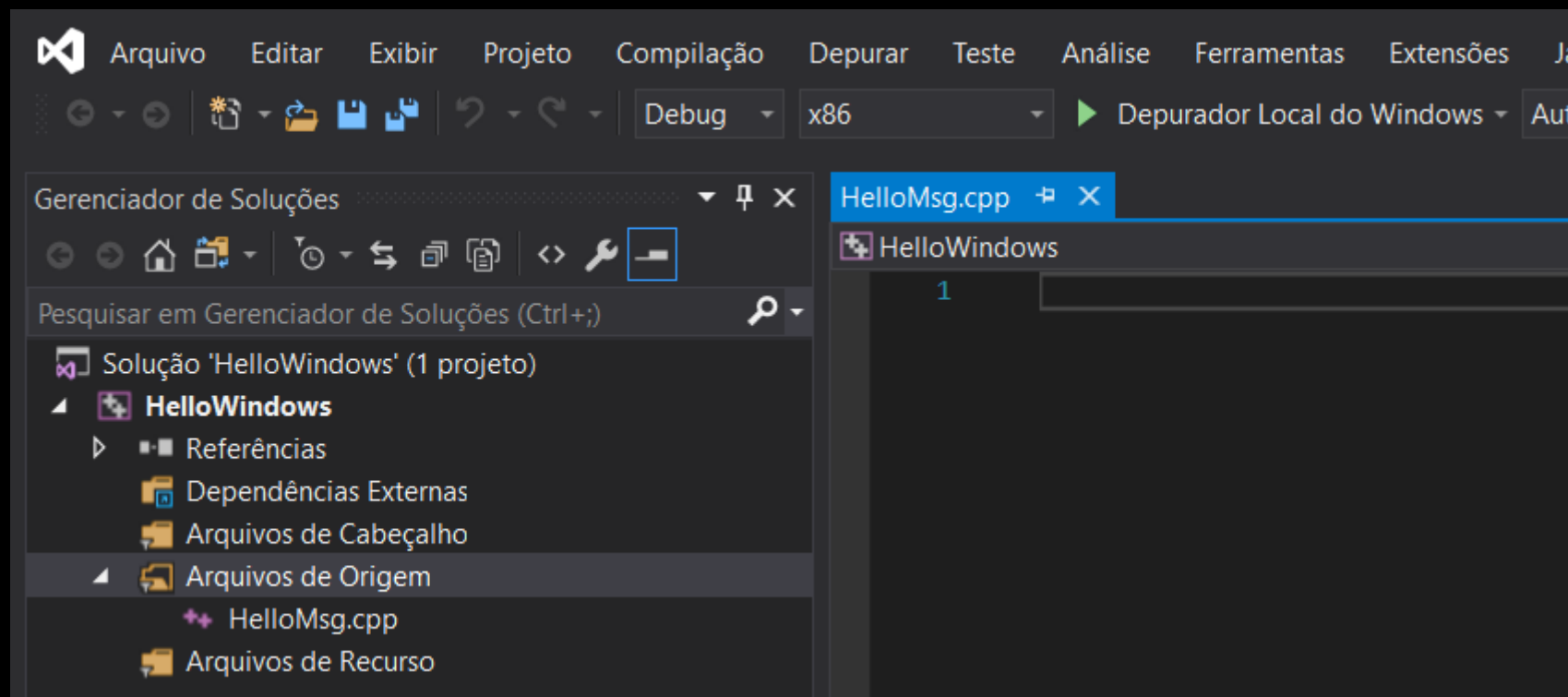
Visual C++ > Arquivo do C++ (.cpp) > HelloMsg



Criando um Programa

HelloMsg.cpp está pronto para edição

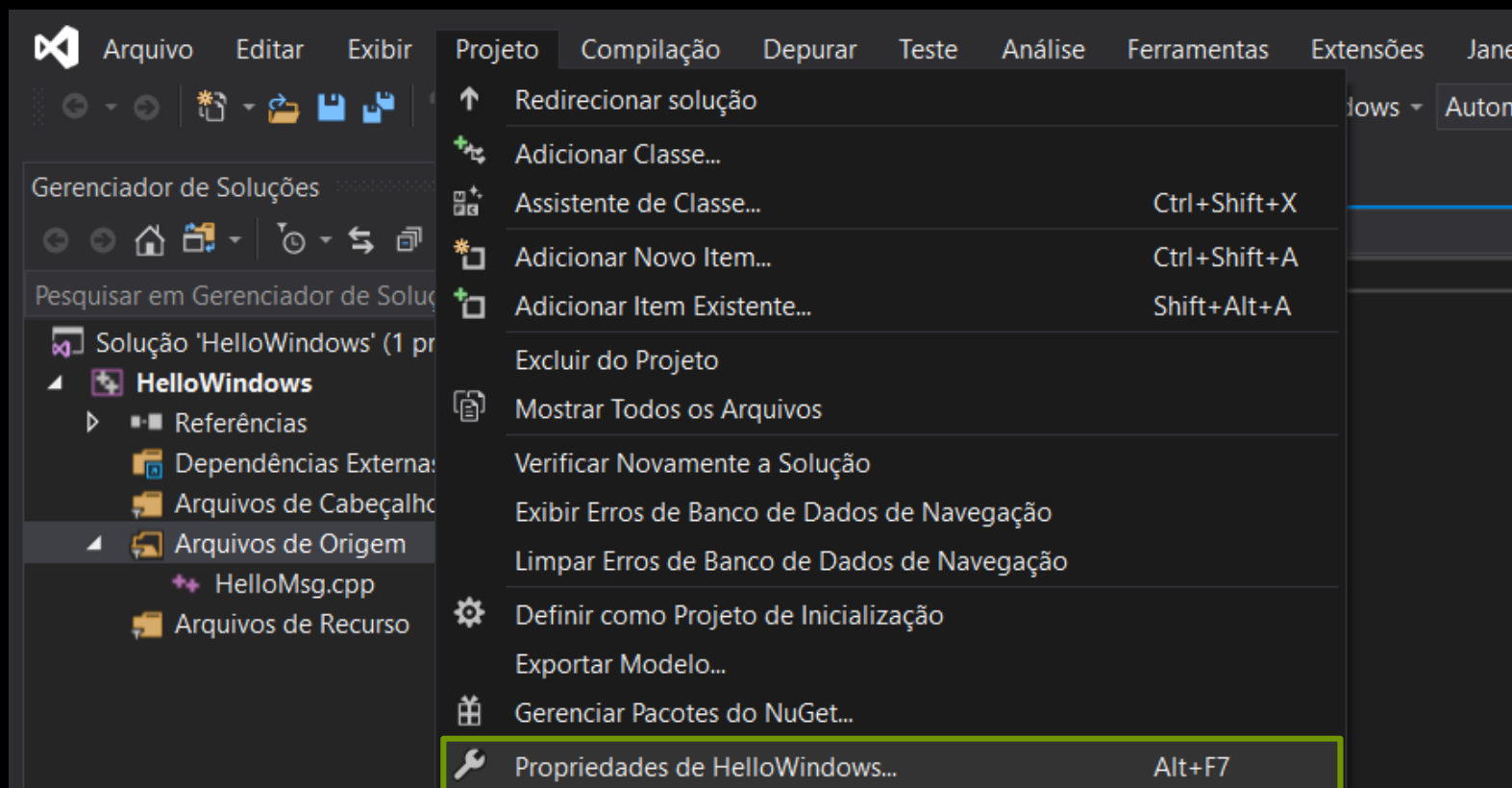
Um projeto pode conter vários arquivos fonte



Criando um Programa

Modificar a tabela de caracteres usada no projeto

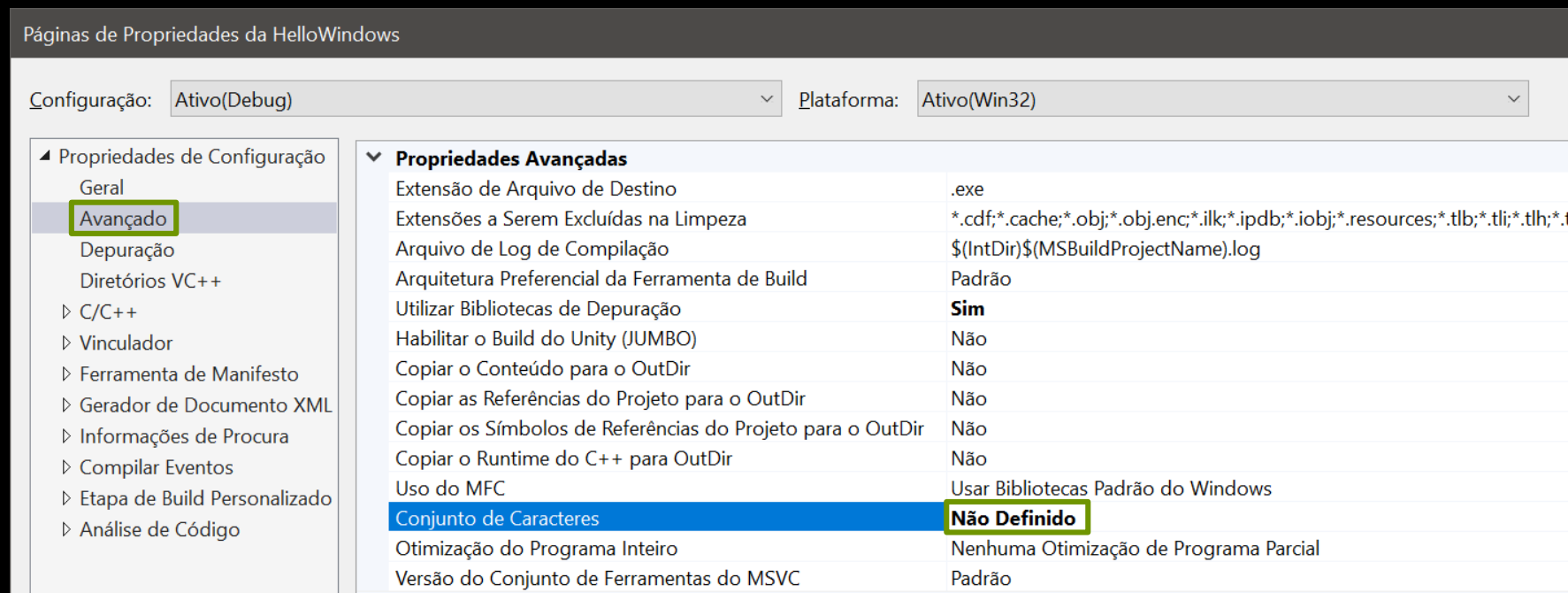
Projeto > Propriedades de HelloWorlds...



Criando um Programa

Modificar o conjunto de caracteres usado no projeto

Propriedades de Configuração > Avançado > Conjunto de Caracteres



Criando um Programa

- Insira e execute o programa:

```
#include <windows.h>
int APIENTRY WinMain (_In_ HINSTANCE hInstance,
                     _In_opt_ HINSTANCE hPrevInstance,
                     _In_ LPSTR lpCmdLine, _In_ int nCmdShow)
{
    MessageBox(NULL, "Oi Windows!", "Mensagem", 0);

    return 0;
}
```

C++ diferencia letras maiúsculas e minúsculas

Os Cabeçalhos

- ▶ A diretiva de pré-processamento inclui o arquivo de cabeçalho windows.h

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

- ▶ windows.h é um **arquivo de cabeçalho mestre** que inclui vários outros:
 - **windows.h** – definições de tipos básicos
 - **winbase.h** – funções do kernel
 - **winuser.h** – funções de interface com o usuário
 - **wingdi.h** – interface com o dispositivo gráfico

A Função WinMain

- ▶ **WinMain** é o ponto de entrada para todo programa **Windows**, assim como **main** é o ponto de entrada para programas no **console em modo texto**

```
int APIENTRY WinMain (_In_ HINSTANCE hInstance,  
                     _In_opt_ HINSTANCE hPrevInstance,  
                     _In_ LPSTR lpCmdLine,  
                     _In_ int nCmdShow)
```

- **APIENTRY** diz ao compilador como passar argumentos para WinMain
- As designações **_In_** e **_In_opt** fazem parte da linguagem de anotação de código-fonte da Microsoft

Hungarian Notation

- ▶ O Windows utiliza uma **convenção para os nomes de variáveis** conhecido como notação húngara

```
int APIENTRY WinMain (_In_ HINSTANCE hInstance,  
                      _In_opt_ HINSTANCE hPrevInstance,  
                      _In_ LPSTR lpCmdLine,  
                      _In_ int nCmdShow)
```

Prefixo	Significado	Tipo de Dado
h	handle	int ou unsigned
lp	long pointer	32 bits
n	number	short ou int

Os Parâmetros

```
int APIENTRY WinMain (_In_ HINSTANCE hInstance, _In_opt_ HINSTANCE hPrevInstance,  
                     _In_ LPSTR lpCmdLine, _In_ int nCmdShow)
```

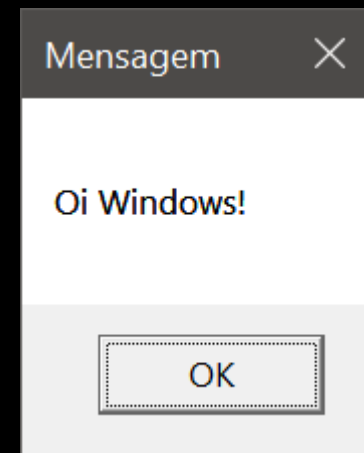
- **hInstance** – manipulador de instância, ou seja, um número para identificar a aplicação
- **hPrevInstance** – sempre nulo (parâmetro era usado em Win16 – Windows 3.1)
- **lpCmdLine** – linha de comando passada na chamada do programa
- **nCmdShow** – define como o programa deve ser inicializado (maximizado, minimizado, tela cheia)

A Caixa de Mensagem

- ▶ Caixas de mensagens foram projetadas para mostrar **mensagens curtas**

```
MessageBox (NULL, "Oi Windows!", "Mensagem", 0);
```

- ▶ **Parâmetros:**
 - Identificador da janela (window handle)
 - String de texto com a mensagem
 - Texto da barra de título
 - Define o estilo da caixa

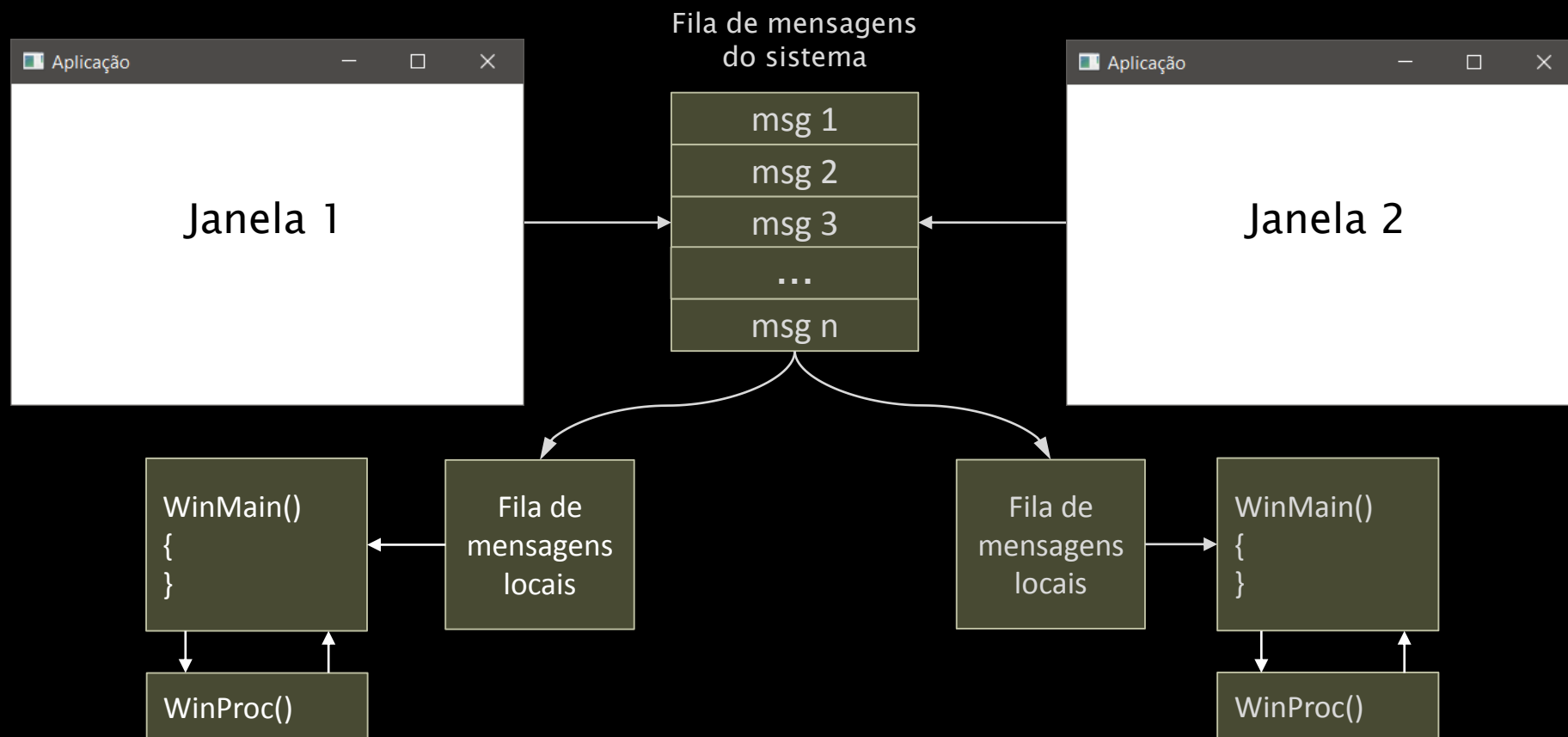


Criação de Janelas

- ▶ A caixa de mensagens é uma janela especial
- ▶ Para criar uma **janela de propósito geral** é preciso chamar a função `CreateWindow`
 - Uma janela é baseada em uma `Window Class`
 - Toda janela precisa de uma `Window Procedure`
- ▶ Para entender os argumentos da função `CreateWindow` é preciso conhecer a **arquitetura do Windows**

Criação de Janelas

► Arquitetura guiada por eventos:



Criação de Janelas

► A função Principal WinMain

```
int APIENTRY WinMain (_In_ HINSTANCE hInstance, _In_opt_ HINSTANCE hPrevInstance,
                     _In_ LPSTR lpCmdLine, _In_ int nCmdShow)
{
    // - Defina uma Window Class Name
    // - Registre a Window Class Name
    // - Crie uma janela baseada na Window Class Name

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

    return msg.wParam;
}
```

Criação de Janelas

► Definindo uma Window Class

```
WNDCLASS wndclass;  
  
wndclass.style          = CS_HREDRAW | CS_VREDRAW;  
wndclass.lpfnWndProc    = WinProc;  
wndclass.cbClsExtra     = 0;           // bytes extra  
wndclass.cbWndExtra     = 0;           // bytes extra  
wndclass.hInstance      = hInstance;  
wndclass.hIcon           = LoadIcon(NULL, IDI_APPLICATION);  
wndclass.hCursor         = LoadCursor(NULL, IDC_ARROW);  
wndclass.hbrBackground  = (HBRUSH) GetStockObject(WHITE_BRUSH);  
wndclass.lpszMenuName    = NULL;  
wndclass.lpszClassName  = "BasicWindow";
```

Criação de Janelas

- ▶ **Constantes** usadas para o **estilo da janela**

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

Constante	Significado
CS_HREDRAW	Redesenha se alterada a largura
CS_VREDRAW	Redesenha se alterado a altura

Criação de Janelas

► Constantes para o ícone da aplicação

```
wndclass.hIcon = LoadIcon(NULL, IDI_APPLICATION);
```

Constante	Significado
IDI_APPLICATION	Ícone padrão de aplicação
IDI_ASTERISK	Asterisco
IDI_EXCLAMATION	Ponto de exclamação
IDI_HAND	Ícone em formato de mão
IDI_QUESTION	Ponto de interrogação
IDI_WINLOGO	Logotipo do Windows

Criação de Janelas

► Constantes para o **cursor da aplicação**

```
wndclass.hCursor = LoadCursor(NULL, IDC_ARROW);
```

Constante	Significado
IDC_ARROW	Cursor padrão de aplicação
IDC_APPSTARTING	Cursor padrão com pequena ampulheta
IDC_CROSS	Em forma de cruz
IDC_UPARROW	Seta vertical
IDC_NO	Círculo cortado com uma barra
IDC_WAIT	Ampulheta

Criação de Janelas

- Constantes para a cor do pano de fundo

```
wndclass.hbrBackground = (HBRUSH) GetStockObject(WHITE_BRUSH);
```

Constante	Significado
BLACK_BRUSH	Preto
WHITE_BRUSH	Branco
GRAY_BRUSH	Cinza
LTGRAY_BRUSH	Cinza Claro
DKGRAY_BRUSH	Cinza Escuro

Criação de Janelas

► Registrando uma Window Class

```
WNDCLASS wndclass;  
wndclass.style          = CS_HREDRAW | CS_VREDRAW;  
wndclass.lpfnWndProc    = WinProc;  
wndclass.cbClsExtra     = 0;           // bytes extra  
wndclass.cbWndExtra     = 0;           // bytes extra  
wndclass.hInstance     = hInstance;  
wndclass.hIcon          = LoadIcon(NULL, IDI_APPLICATION);  
wndclass.hCursor        = LoadCursor(NULL, IDC_ARROW);  
wndclass.hbrBackground = (HBRUSH) GetStockObject(WHITE_BRUSH);  
wndclass.lpszMenuName   = NULL;  
wndclass.lpszClassName = "BasicWindow";
```

```
if (!RegisterClass (&wndclass))  
{  
    MessageBox(NULL, "Erro na criação da janela!", "Aplicação", MB_ICONERROR) ;  
    return 0 ;  
}
```

Criação de Janelas

► A função `CreateWindow`

```
HWND hwnd; // identificador da janela

hwnd = CreateWindow(
    "BasicWindow", // nome da window class
    "Aplicação", // título da janela
    WS_OVERLAPPEDWINDOW, // estilo da janela
    CW_USEDEFAULT, // posição x inicial
    CW_USEDEFAULT, // posição y inicial
    CW_USEDEFAULT, // largura inicial
    CW_USEDEFAULT, // altura inicial
    NULL, // identificador da janela pai
    NULL, // identificador do menu
    hInstance, // identificador da aplicação
    NULL); // parâmetros de criação
```

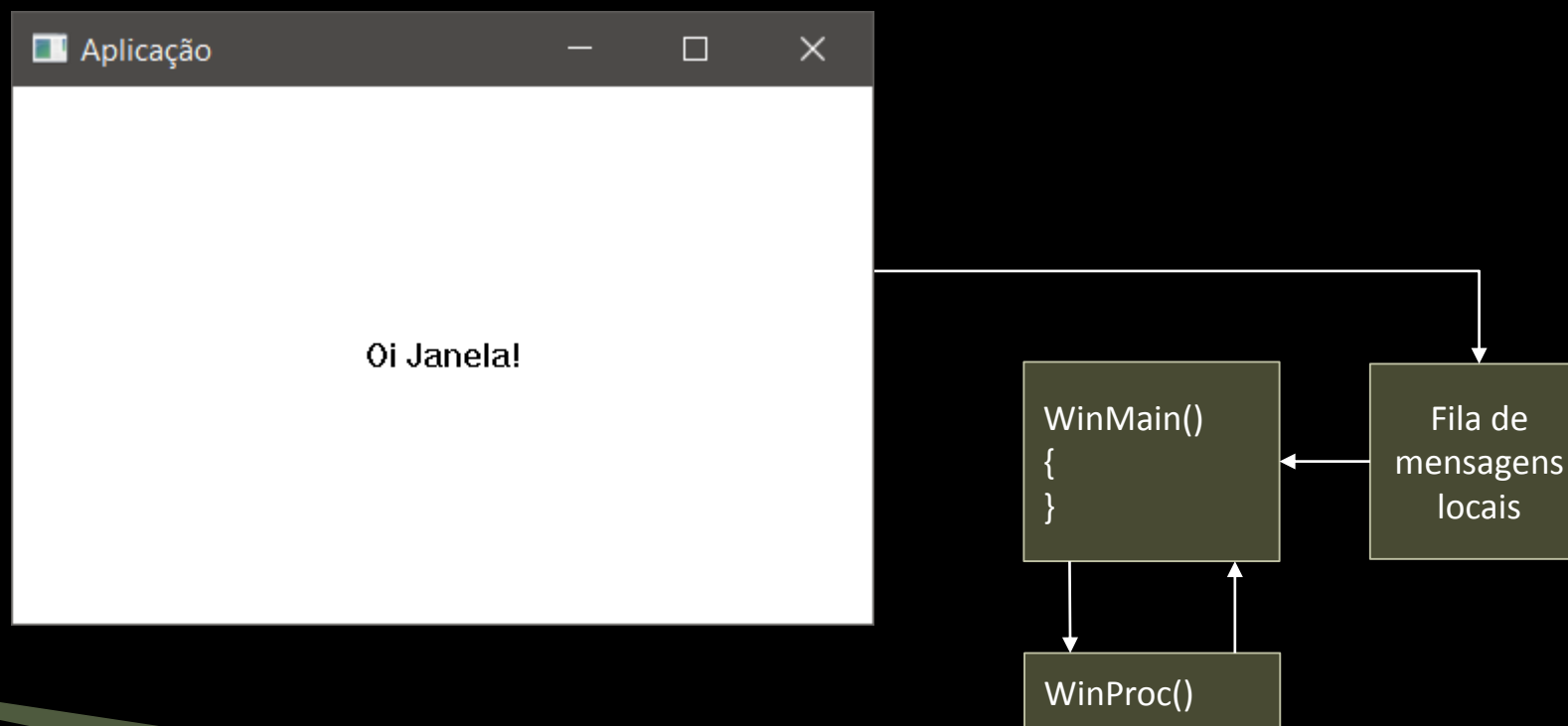
Criação de Janelas

► Constantes para o **estilo da janela**

Constante	Significado
WS_POPUP	Janela sem nenhuma barra ou borda
WS_OVERLAPPED	Janela com barra de título e bordas
WS_OVERLAPPEDWINDOW	Janela com os estilos WS_OVERLAPPED, WS_CAPTION, WS_SYSMENU, WS_THICKFRAME, WS_MINIMIZEBOX e WS_MAXIMIZE_BOX
WS_VISIBLE	Janela inicialmente visível
WS_SYSMENU	Janela com menu na barra de título (inclui WS_CAPTION)
WS_BORDER	Janela com uma borda fina
WS_CAPTION	Janela com uma barra de título (inclui WS_BORDER)
WS_MAXIMIZEBOX	Janela com o botão de maximizar
WS_MINIMIZEBOX	Janela com o botão de minimizar

Criação de Janelas

- ▶ A função `CreateWindow` cria uma janela cuja **interação** pode ser **capturada e tratada** em uma `Window Procedure`



Criação de Janelas

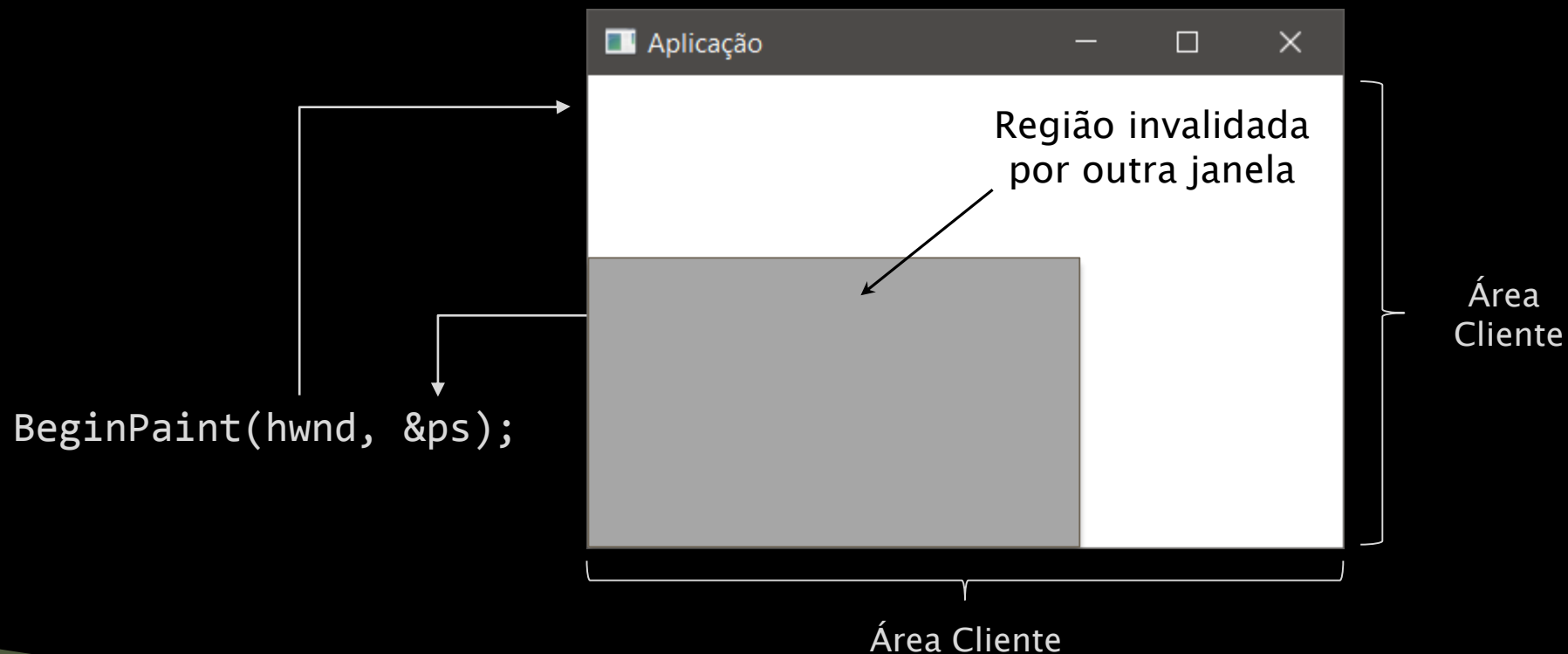
► Definindo uma **Window Procedure**

```
LRESULT CALLBACK WinProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    PAINTSTRUCT ps;
    RECT rect;

    switch (message)
    {
    case WM_PAINT:
        HDC hdc = BeginPaint(hwnd, &ps);
        GetClientRect (hwnd, &rect);
        DrawText (hdc, "Minha Janela!", -1, &rect, DT_CENTER | DT_VCENTER);
        EndPaint (hwnd, &ps);
        return 0;
    }
    return DefWindowProc(hwnd, message, wParam, lParam);
}
```


Criação de Janelas

- ▶ **WM_PAINT** é recebida sempre que a janela precisa ser repintada



Criação de Janelas

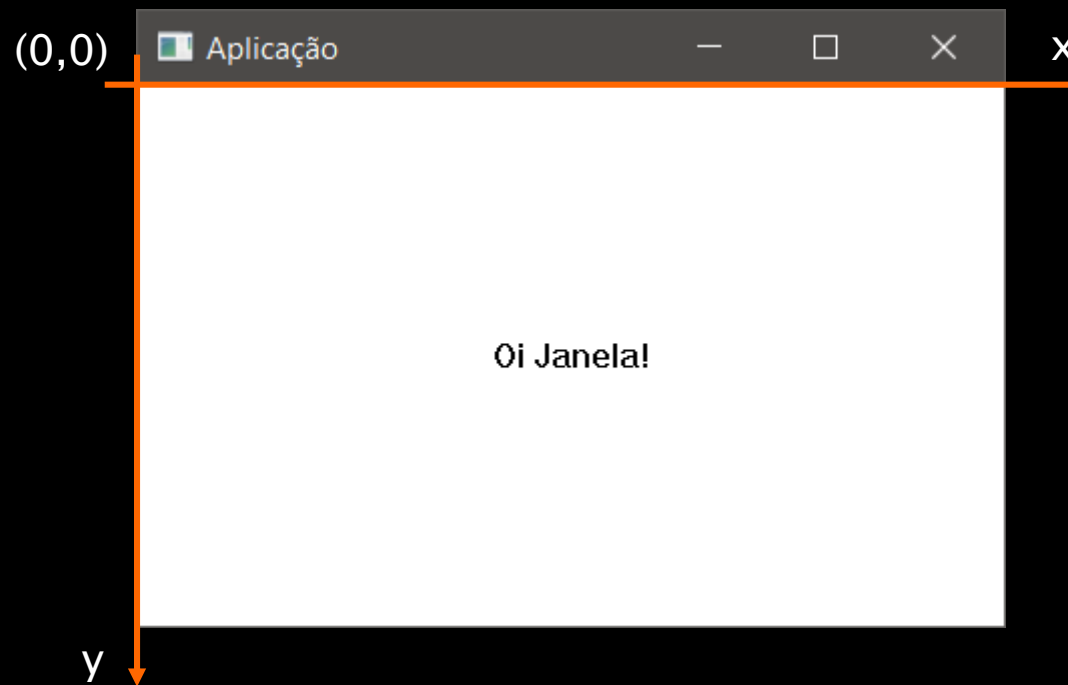
- ▶ A **saída de texto** em uma janela pode ser feita com a função TextOut ou DrawText

```
// pega o contexto do dispositivo
// HDC hdc = GetDC(hwnd);
HDC hdc = BeginPaint(hwnd, &ps);

// define o retângulo de destino
// RECT rect = {0, 0, 100, 25};
GetClientRect (hwnd, &rect);

// mostra o texto na janela
// TextOut(hdc, 50, 40, "Teste", -1);
DrawText(hdc, "Teste", -1, &rect,
    DT_SINGLELINE | DT_CENTER | DT_VCENTER);

// libera contexto do dispositivo
// ReleaseDC(hwnd, hdc);
EndPaint (hwnd, &ps);
```



Criação de Janelas

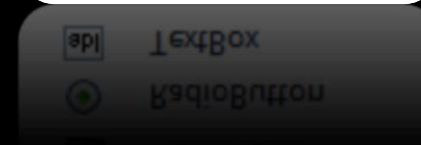
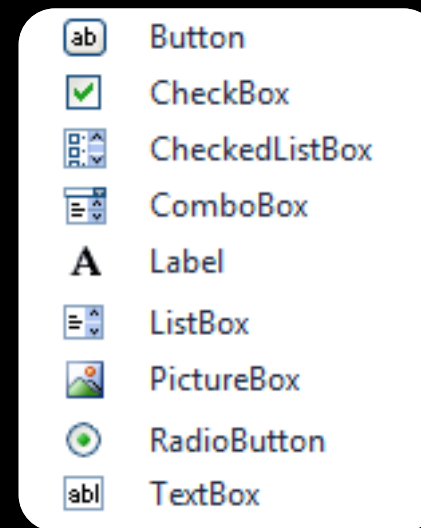
- Uma lista resumida de **mensagens**:

Mensagem	Significado
WM_ACTIVATE	Enviado quando uma janela recebe o foco
WM_CLOSE	Enviado quando uma janela é fechada
WM_CREATE	Enviado quando uma janela é criada
WM_DESTROY	Enviado quando uma janela está para ser destruída
WM_MOVE	Enviado quando uma janela foi movida
WM_MOUSEMOVE	Enviado quando o mouse é movimentado
WM_KEYUP	Enviado quando uma tecla é liberada
WM_KEYDOWN	Enviado quando uma tecla é pressionada
WM_PAINT	Enviado quando a janela precisa ser redesenhada
WM_SIZE	Enviado quando uma janela mudou de tamanho

Criação de Janelas

- ▶ O código apresentado pode ser usado como **ponto de partida** para qualquer programa Windows
- ▶ A função **CreateWindow** é usada também para criar **janelas filhas**:

Rótulos, botões,
barras de rolagem,
caixas de texto, agrupamentos,
listas, imagens, caixas de marcação,
botões de seleção, etc.



Resumo

- ▶ O Windows é um sistema operacional **multitarefa**
- ▶ A arquitetura de programação é **guiada por eventos**:
 - Para criar uma janela é preciso:
 - Criar e registrar uma Window Class
 - Criar uma Window Procedure
 - Capturar e tratar as mensagens geradas pelo usuário
- ▶ A função **CreateWindow** permite criar janelas
 - Todos os elementos de interação com o usuário são janelas