Programação em C++

SENAI

Prof. Daniel Santos

daniel.sampaio@sp.senai.br

Senai Roberto Simonsen

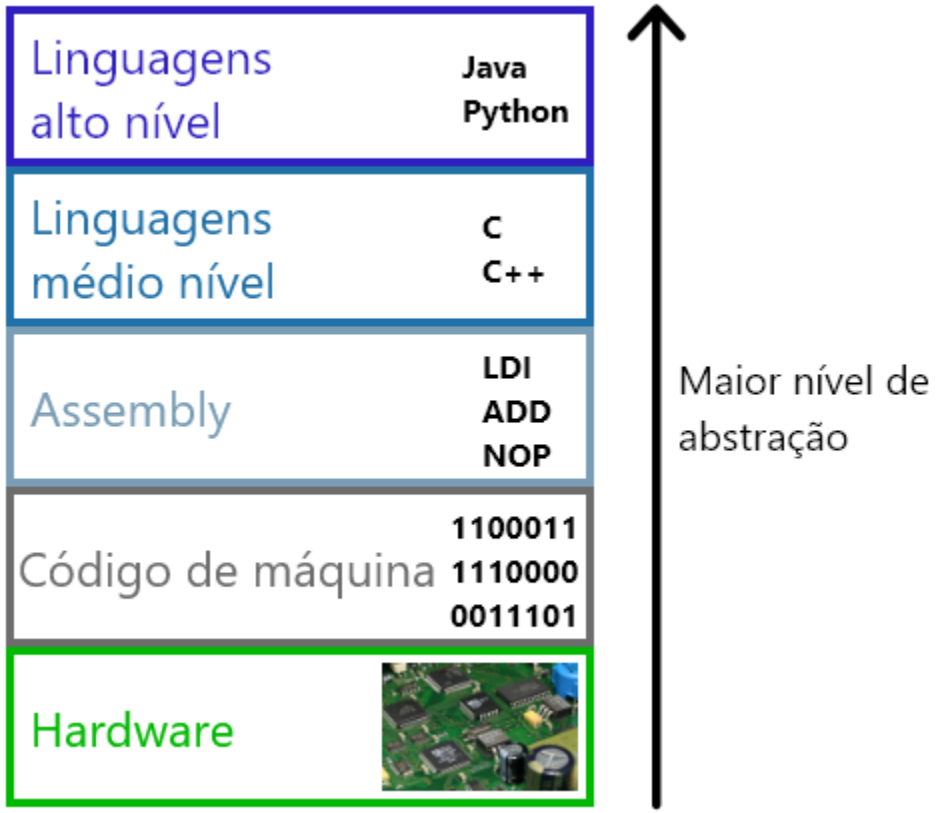
R. Monsenhor Andrade, 298 - Brás, São Paulo - SP

Prof. Daniel Santos

O que é C++?

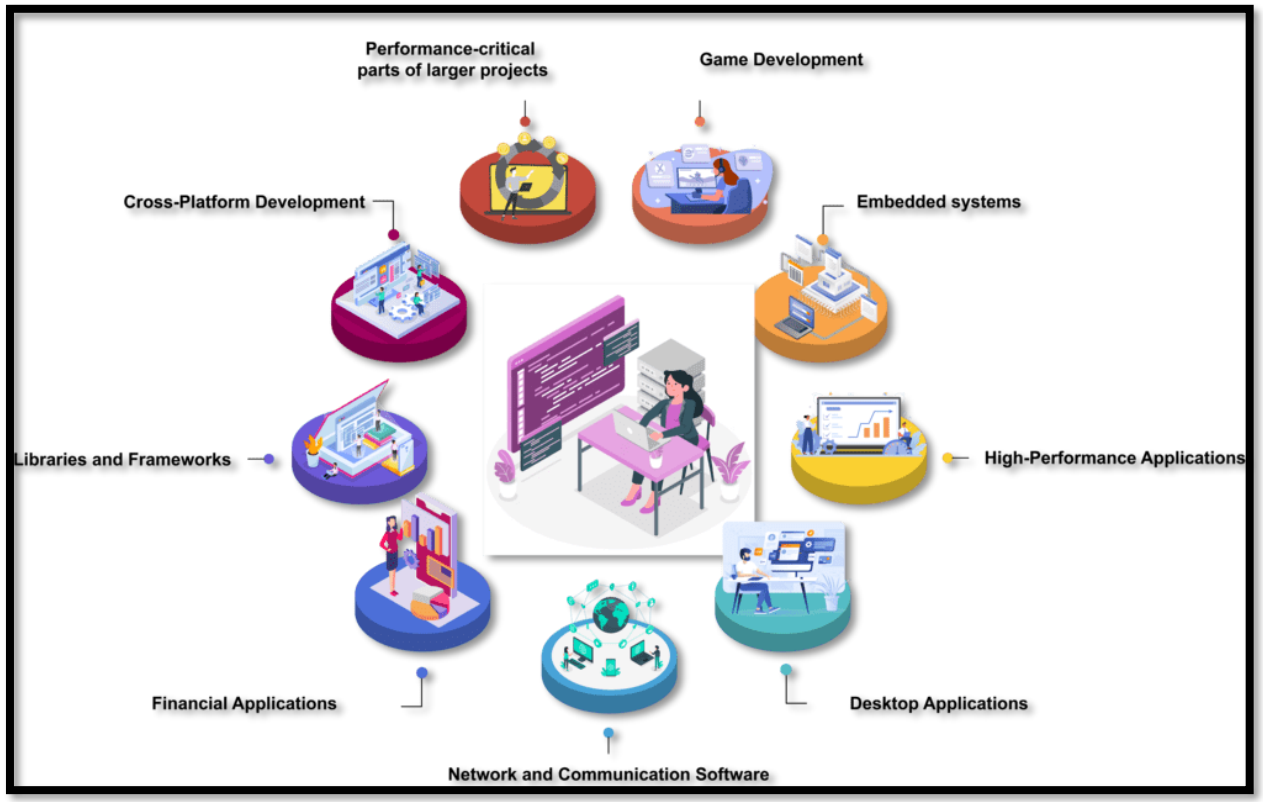
- C++ é uma linguagem de programação **compilada** de alto desempenho.
- Suporta programação estruturada, orientada a objetos e genérica.
- Muito usada em sistemas embarcados, jogos e aplicações de alto desempenho.

	Vantagens	Desvantagens
Compiladores	Execução mais rápida	Várias etapas de tradução
	Permite estruturas de programação mais completas para a sua execução	Programação final é maior necessitando mais memória
	Permite a otimização do código fonte	Processo de correção de erros e depuração é mais demorado
Interpretadores	Depuração do programa é mais simples	Execução do programa é mais lenta
	Consome menos memória	Estruturas de dados demasiado simples
	Resultado imediato do programa ou rotina desenvolvida	Necessário fornecer o programa fonte ao utilizador



O que posso fazer com C++?

Um de seus maiores atrativos é possuir uma ampla coleção de bibliotecas, tanto nativas quanto de terceiros, tornando-a muito popular e útil em diversas áreas, como desenvolvimento de **sistemas**, **jogos**, **computação de alto desempenho**, **Internet das Coisas (IoT)** e até em aplicações de **inteligência artificial** e **aprendizado de máquina**.



Por que aprender C++?

- É uma das linguagens de programação mais utilizadas no mundo.
- É uma linguagem com amplo controle de otimização para melhorar a performance.
- Esta linguagem pode ser aplicada em diversos campos, desde **sistemas embarcados**, **desenvolvimento de jogos** até **machine learning** e **visão computacional**.
- A comunidade é bem ativa, então se houver alguma dúvida especifica é possível pedir auxilio para outros desenvolvedores nos fóruns (ex: https://groups.google.com/g/ccppbrasil)

Mar 2025	Mar 2024	Change	Programming Language	Ratings	Change
1	1		Python	23.85%	+8.22%
2	3	▲	C++	11.08%	+0.37%
3	4	▲	Java	10.36%	+1.41%
4	2	▼	C	9.53%	-1.64%
5	5		C#	4.87%	-2.67%

Raciocínio lógico

- Para aprender qualquer linguagem de programação é primordial ter **raciocínio lógico**
- Ter **raciocínio lógico** é pensar seguindo uma **sequência lógica** de instruções para executar alguma tarefa, isto não precisa ser necessariamente aplicado a programação.

Exemplo: **Fazer um bolo.** Para fazer um bolo é necessário seguir uma sequência de tarefas ordenadas.

```
// RECEITA DE BOLO COMUM DE OVOS
INÍCIO
Passo 1: Separar os ingredientes
Ingredientes:
2 ovos;
3 xícaras de farinha de trigo;
1 e ½ colher de fermento;
¾ xícara de leite.
1/2 xícaras de açúcar;
250g de manteiga;

Modo de preparo:
Passo 2: Aqueça o forno a 180 graus;
Passo 3: Quebre os ovos e separe as claras da gema;
Passo 4: Bata as claras em neve e as deixe separadas;
Passo 5: Em uma vasilha, bata o açúcar, a manteiga e as gemas;
Passo 6: Misture a farinha e o leite;
Passo 7: Bata bem, até ficar bem homogêneo;
Passo 8: Acrescente o fermento;
Passo 9: Adicione as claras em neve e mexa cuidadosamente;
Passo 10: Unte uma forma com manteiga e farinha de trigo.
Passo 11: Coloque a massa na forma untada
Passo 12: Leve ao forno médio para assar por aproximadamente 35 minutos ou até que, ao
 espetar um palito, esse saia seco;
Passo 13: Após assado, desligue o forno e deixe o bolo esfriar;
Passo 14: Desenforme e saboreie.
FIM
```

O nome dado a esta sequência de passos para executar uma tarefa é **algoritmo**.

Algoritmo

Podemos representar um algoritmo de algumas maneiras diferentes:

Descrição Narrativa

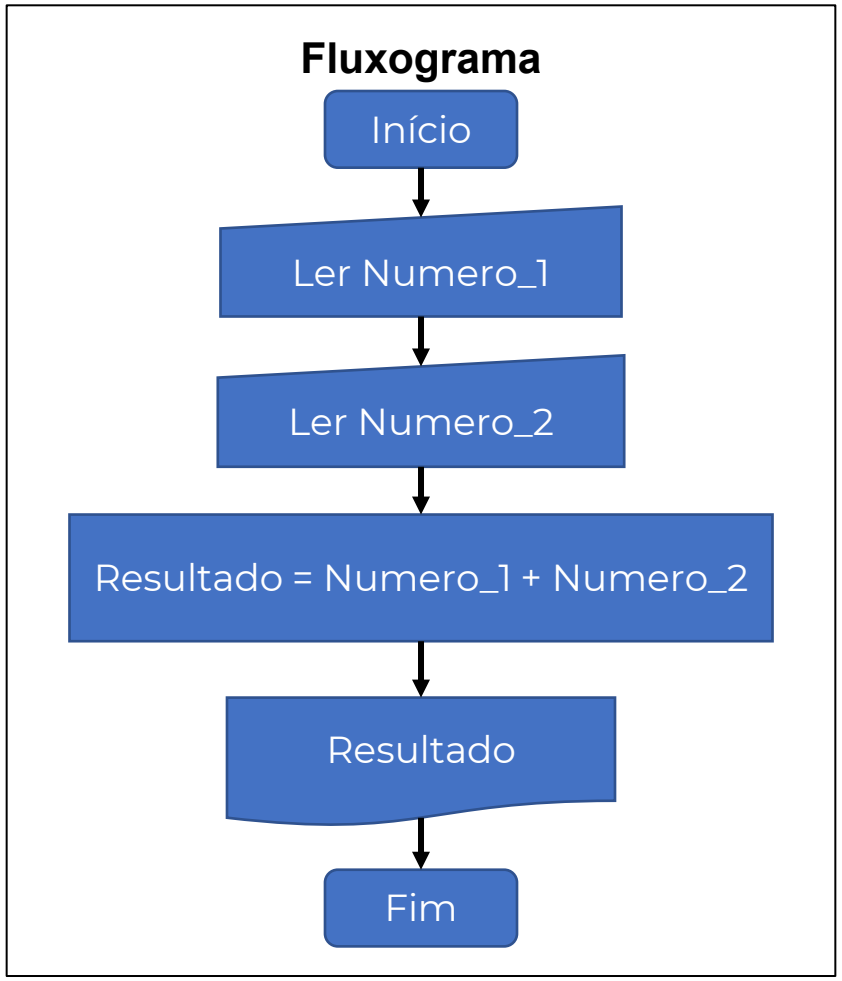
Passo 1: Receber o primeiro número

Passo 2: Receber o segundo número

Passo 3: Somar os dois números

Passo 4: Exibir resultado

Fluxograma



Pseudocódigo (PORTUGOL)

```
Declare Numero_1, Numero_2, Resultado;
Leia Numero_1, Numero_2;
Resultado = Numero_1 + Numero_2;
Escreva Resultado.
```

Linguagem C++

Assim como aprender um idioma, para aprender uma linguagem de programação é necessário entender a **sintaxe**, ou seja, como ela é escrita:

- A linguagem C++ é **case sensitive** → Diferencia letras Maiúsculas de letras minúsculas

- A linguagem C++ delimita os blocos por **chaves**:

Exemplo:

```
Este Bloco {
        Tem este comando dentro dele
        E este comando
        ...
}
Mas este comando não, pois está fora do bloco
```

Linguagem C++

- Assim como a língua portuguesa têm várias maneiras de expressar a mesma frase, algumas mais organizadas e mais simples de entender do que outras.

Exemplo: **cOnseGuimos enTeNder eSta FRaSE aPesar dE nÃO estAr BEm orGaNiZAda**

Podemos usar como um guia de **Boas Práticas**:
https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines

Este guia orienta como escrever a linguagem de forma organizada e padronizada, apesar da linguagem funcionar mesmo se não seguir à risca o guia.

IDE

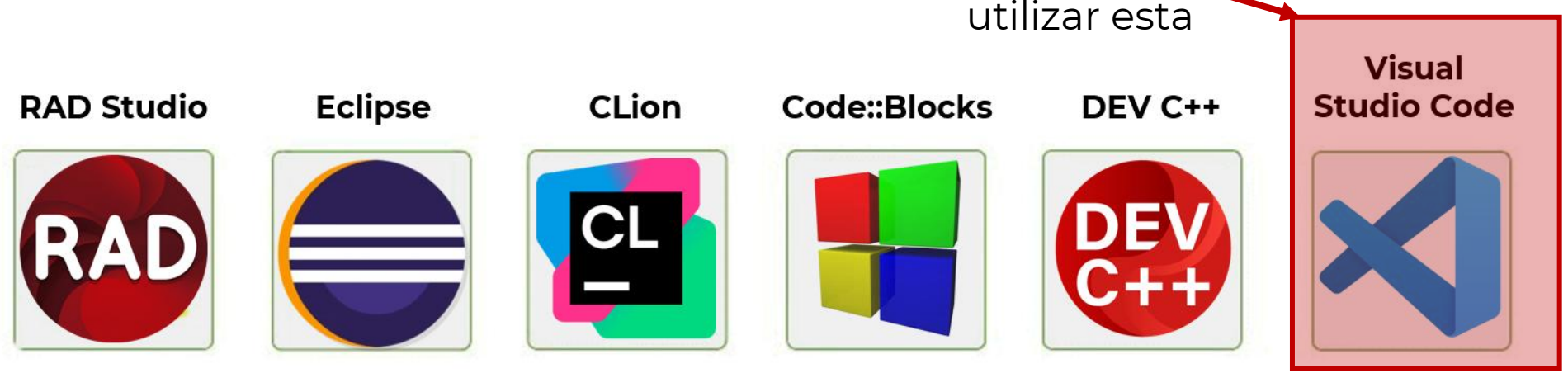
- Para executar programas em C++ é necessário um compilador, que transforma o código-fonte escrito em C++ em um arquivo executável que realiza as tarefas descritas

- Para escrever a linguagem, pode ser feito em qualquer **Editor de Texto**, até mesmo no bloco de notas

- Porém, existem programas que unem um **Compilador** junto com um **Editor de Texto** voltado a programação, estes softwares são chamados de **IDE** (**I**ntegrated **D**evelopment **E**nvironent → Ambiente de Desenvolvimento Integrado)

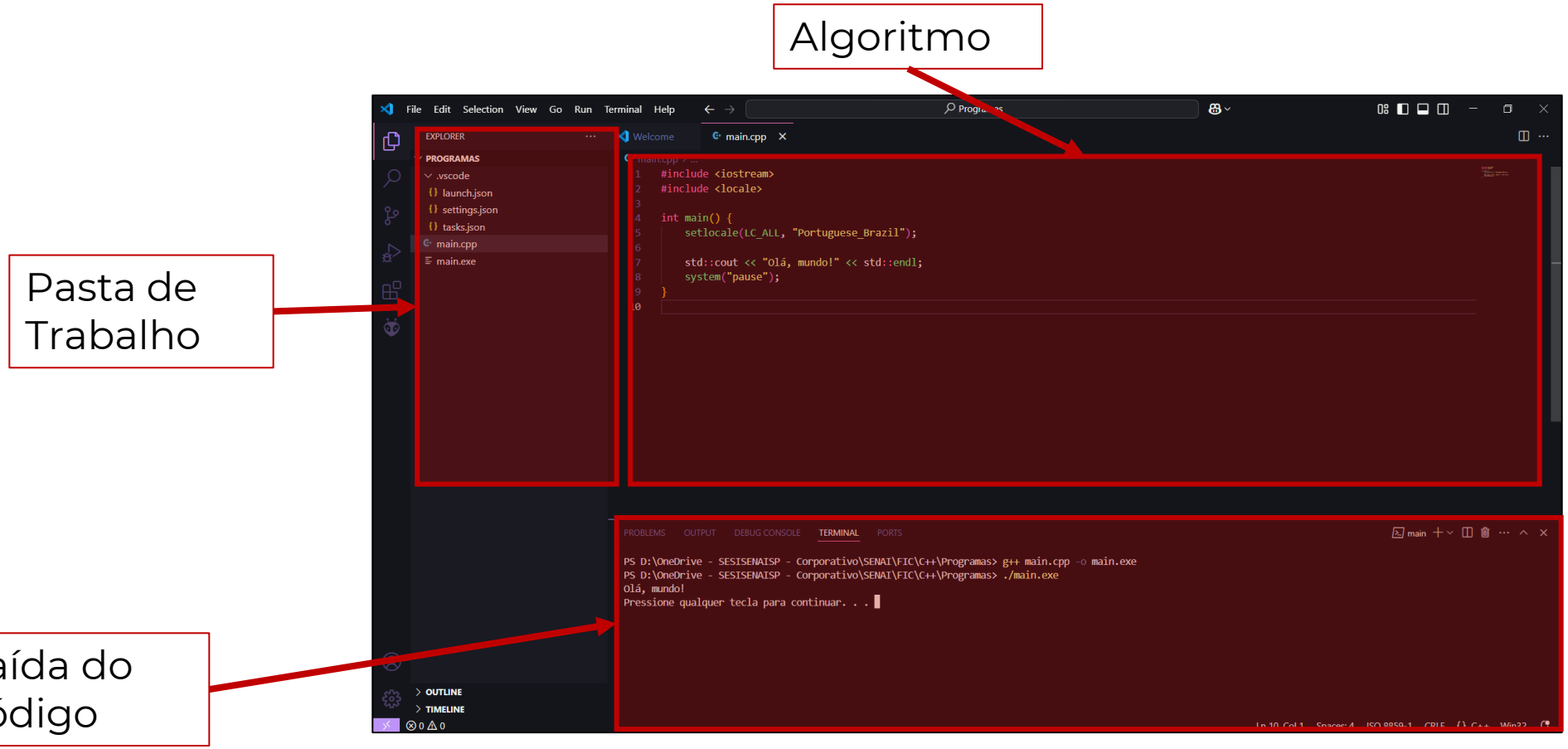
Algumas IDEs:

Vamos utilizar esta



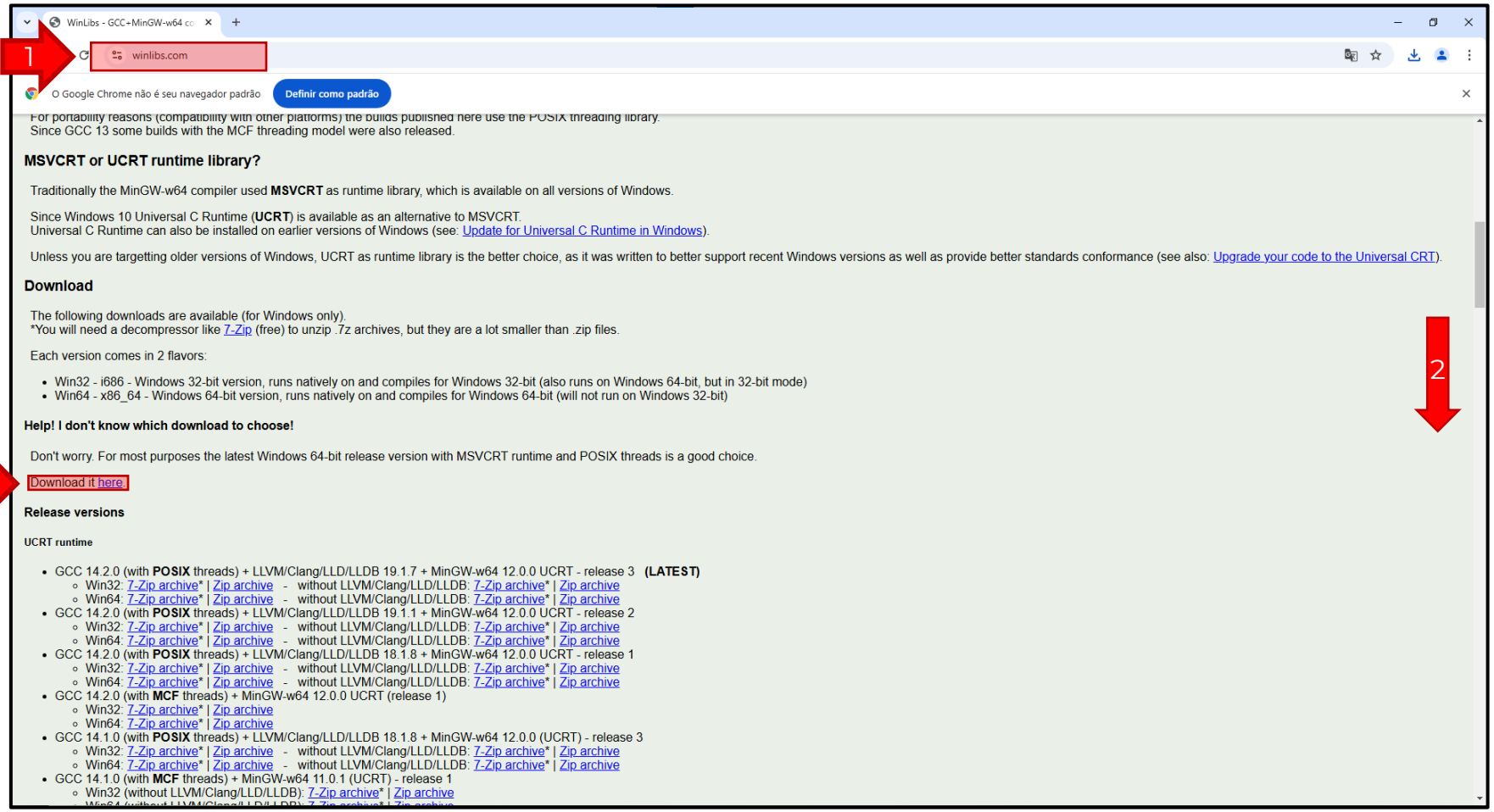
IDE

A IDE do Visual Studio Code tem o seguinte layout:



Instalação do Compilador

Entre no site: https://winlibs.com/ [1]
Role a página até seção **Download** [2]
Baixe o Mingw-w64 [3]



For portability reasons (compatibility with other platforms) the builds published here use the POSIX threading library.
Since GCC 13 some builds with the MCF threading model were also released.

MSVCRT or UCRT runtime library?

Traditionally the MinGW-w64 compiler used **MSVCRT** as runtime library, which is available on all versions of Windows.

Since Windows 10 Universal C Runtime (**UCRT**) is available as an alternative to MSVCRT.
Universal C Runtime can also be installed on earlier versions of Windows (see: Update for Universal C Runtime in Windows).

Unless you are targetting older versions of Windows, UCRT as runtime library is the better choice, as it was written to better support recent Windows versions as well as provide better standards conformance (see also: Upgrade your code to the Universal CRT).

Download

The following downloads are available (for Windows only).
*You will need a decompressor like 7-Zip (free) to unzip .7z archives, but they are a lot smaller than .zip files.

Each version comes in 2 flavors:

- Win32 - i686 - Windows 32-bit version, runs natively on and compiles for Windows 32-bit (also runs on Windows 64-bit, but in 32-bit mode)
- Win64 - x86_64 - Windows 64-bit version, runs natively on and compiles for Windows 64-bit (will not run on Windows 32-bit)

Help! I don't know which download to choose!

Don't worry. For most purposes the latest Windows 64-bit release version with MSVCRT runtime and POSIX threads is a good choice.

Download it here.

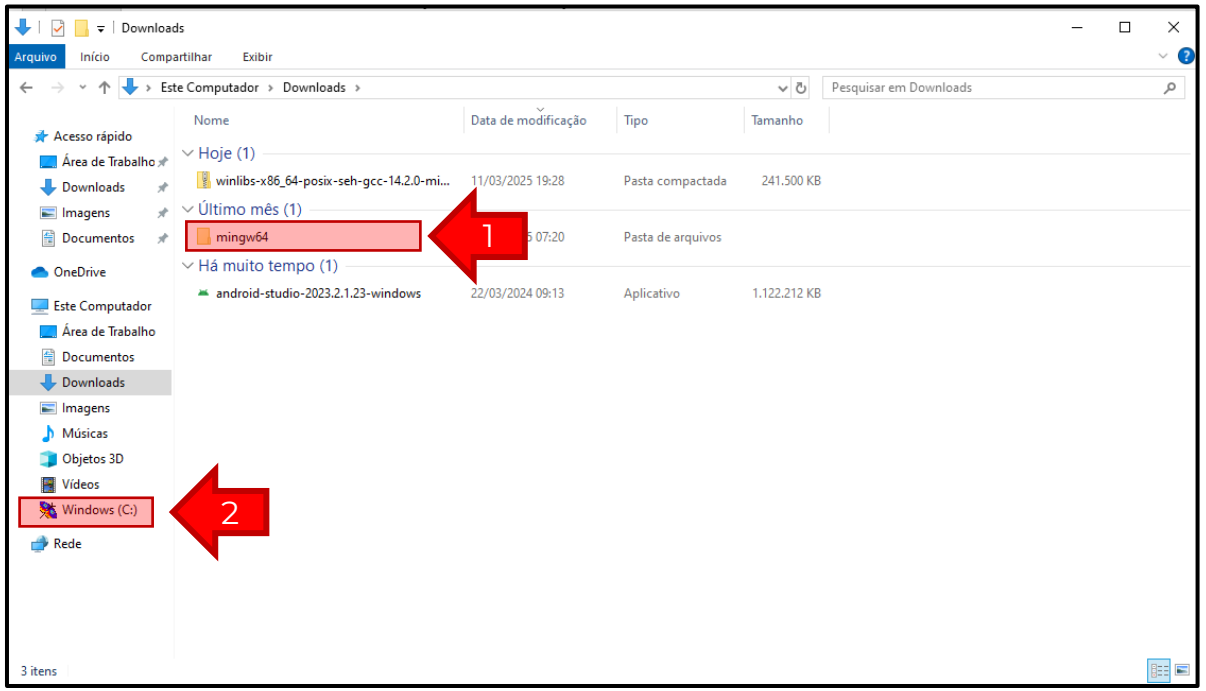
Release versions

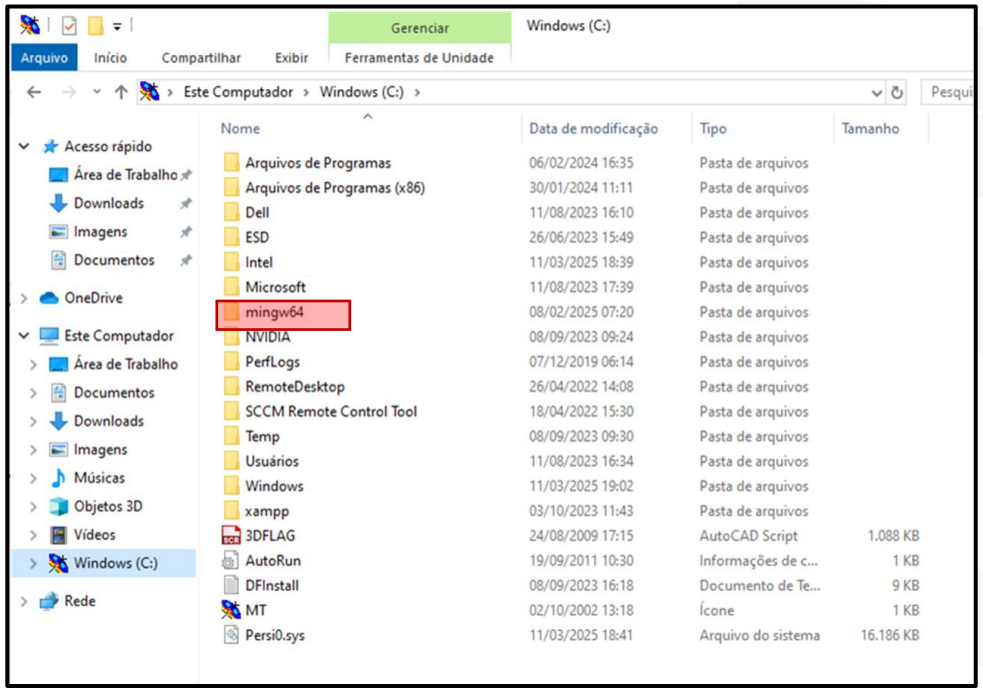
UCRT runtime

- GCC 14.2.0 (with **POSIX** threads) + LLVM/Clang/LLD/LLDB 19.1.7 + MinGW-w64 12.0.0 UCRT - release 3 **(LATEST)**
  - Win32: 7-Zip archive* | Zip archive - without LLVM/Clang/LLD/LLDB: 7-Zip archive* | Zip archive
  - Win64: 7-Zip archive* | Zip archive - without LLVM/Clang/LLD/LLDB: 7-Zip archive* | Zip archive
- GCC 14.2.0 (with **POSIX** threads) + LLVM/Clang/LLD/LLDB 19.1.1 + MinGW-w64 12.0.0 UCRT - release 2
  - Win32: 7-Zip archive* | Zip archive - without LLVM/Clang/LLD/LLDB: 7-Zip archive* | Zip archive
  - Win64: 7-Zip archive* | Zip archive - without LLVM/Clang/LLD/LLDB: 7-Zip archive* | Zip archive
- GCC 14.2.0 (with **POSIX** threads) + LLVM/Clang/LLD/LLDB 18.1.8 + MinGW-w64 12.0.0 UCRT - release 1
  - Win32: 7-Zip archive* | Zip archive - without LLVM/Clang/LLD/LLDB: 7-Zip archive* | Zip archive
  - Win64: 7-Zip archive* | Zip archive - without LLVM/Clang/LLD/LLDB: 7-Zip archive* | Zip archive
- GCC 14.2.0 (with **MCF** threads) + MinGW-w64 12.0.0 UCRT (release 1)
  - Win32: 7-Zip archive* | Zip archive
  - Win64: 7-Zip archive* | Zip archive
- GCC 14.1.0 (with **POSIX** threads) + LLVM/Clang/LLD/LLDB 18.1.8 + MinGW-w64 12.0.0 (UCRT) - release 3
  - Win32: 7-Zip archive* | Zip archive - without LLVM/Clang/LLD/LLDB: 7-Zip archive* | Zip archive
  - Win64: 7-Zip archive* | Zip archive - without LLVM/Clang/LLD/LLDB: 7-Zip archive* | Zip archive
- GCC 14.1.0 (with **MCF** threads) + MinGW-w64 11.0.1 (UCRT) - release 1
  - Win32 (without LLVM/Clang/LLD/LLDB): 7-Zip archive* | Zip archive

Instalação do Compilador

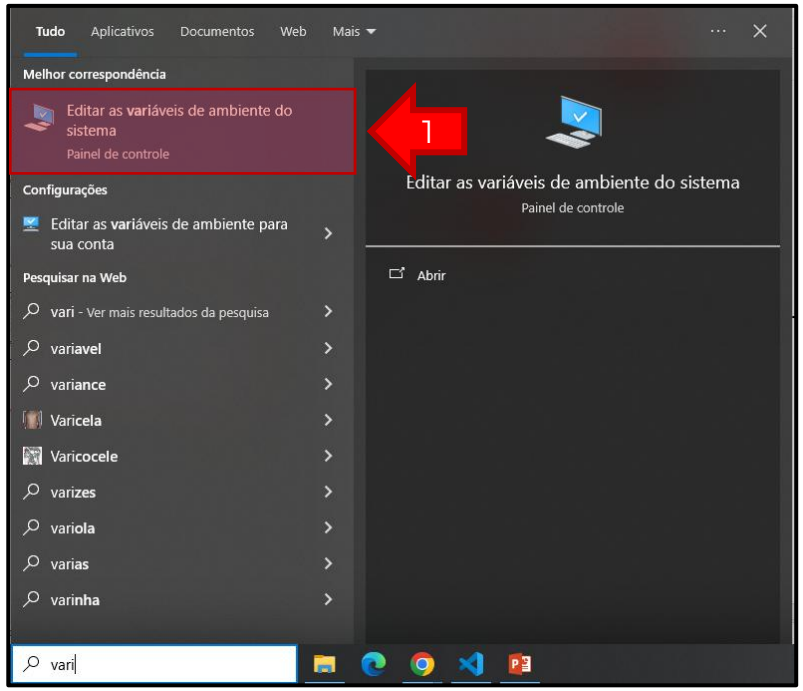
Extraia a pasta do arquivo **ZIP** [1]
Mova a pasta para o Diretório C/ [2]

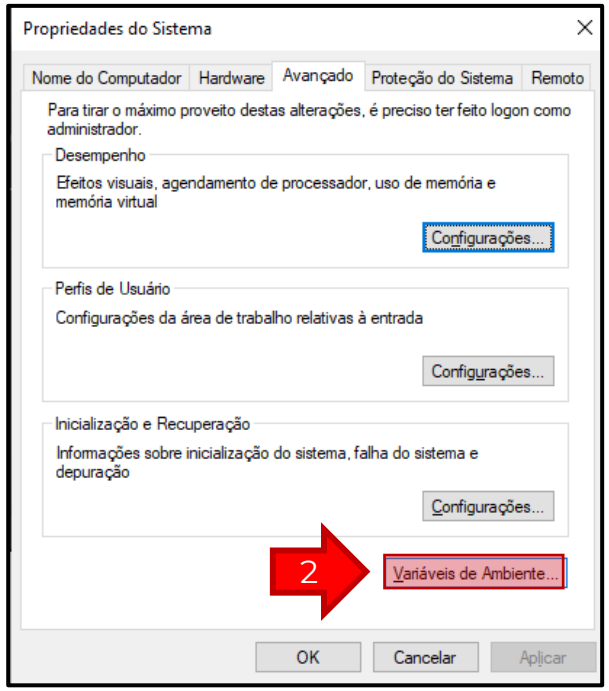


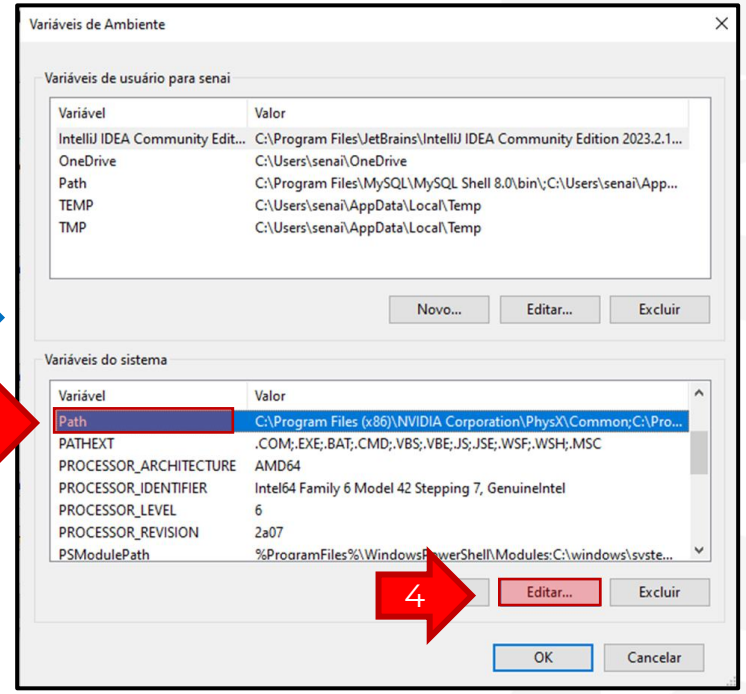


Instalação do Compilador

Abra a opção de **Editar Variáveis do Ambiente** [1]
Clique em **Variáveis do Ambiente...** [2]
Na parte de **Variáveis do Sistema**, selecione o campo **Path** [3] e clique em **Editar** [4]

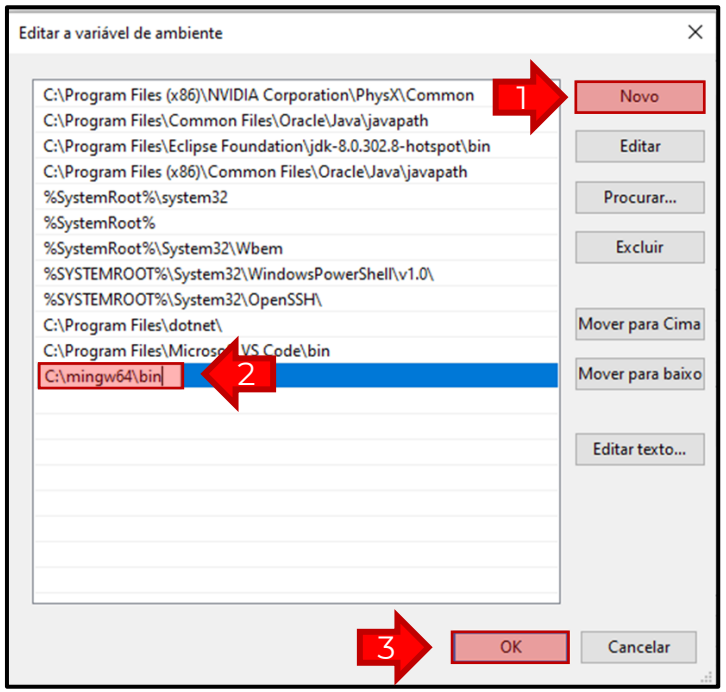


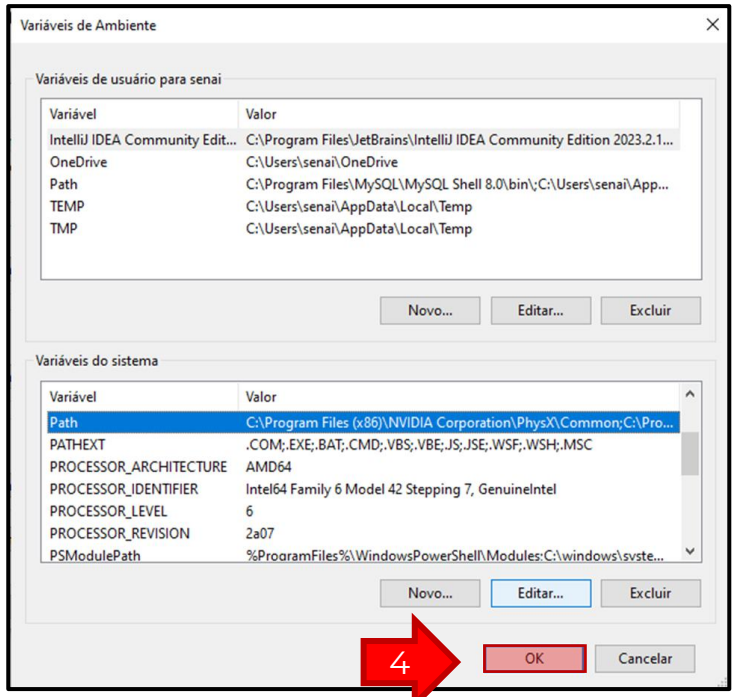


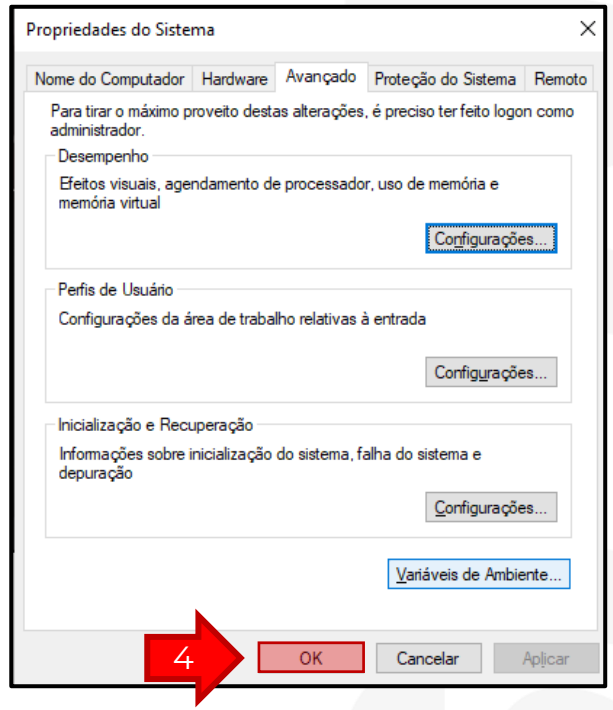


Instalação do Compilador

Clique em **Novo** [1]
Digite **C:\mingw64\bin** [2]
Clique em **OK** [3]
Clique em **OK** [4] nas outras janelas que abriram para finalizar essa configuração

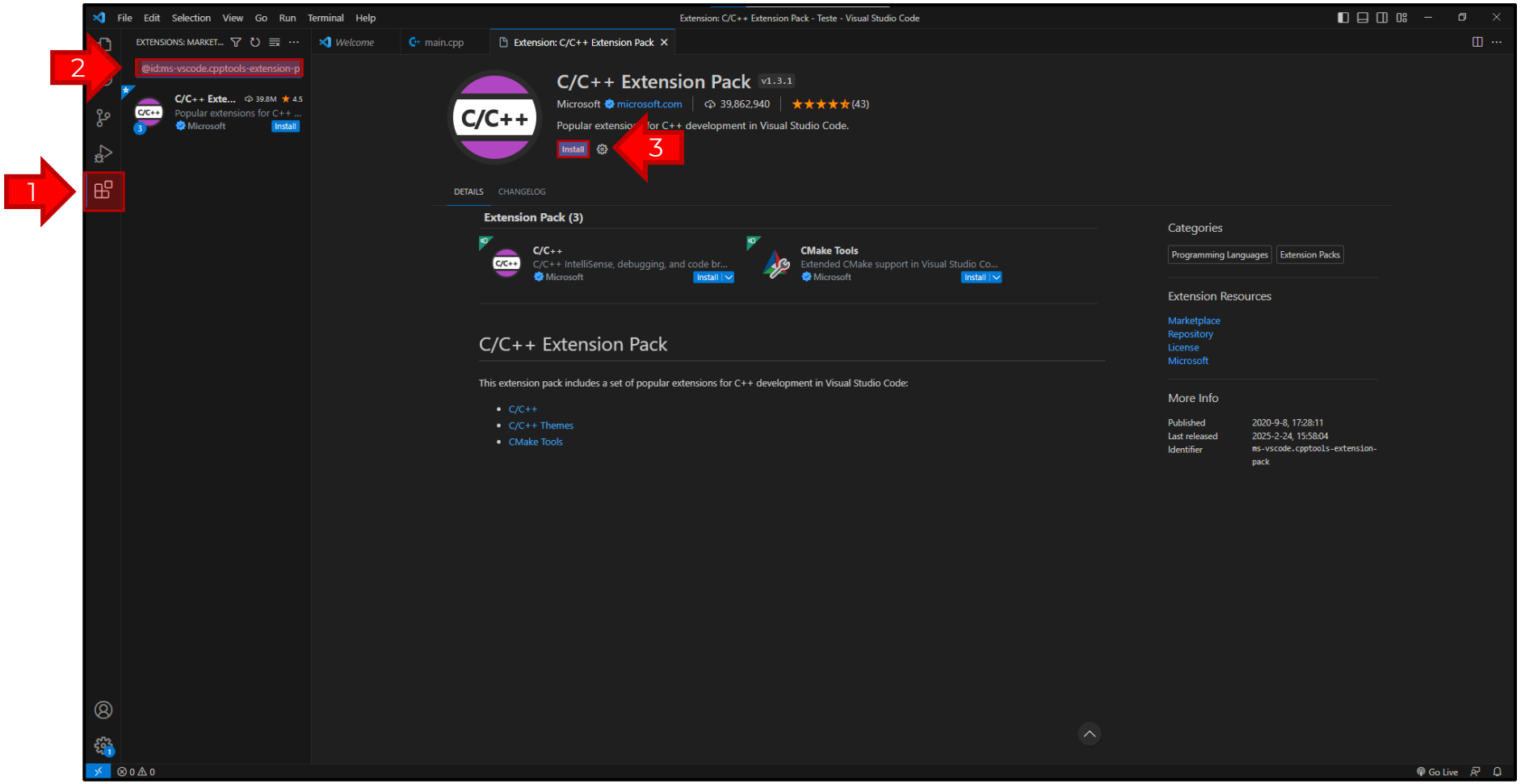






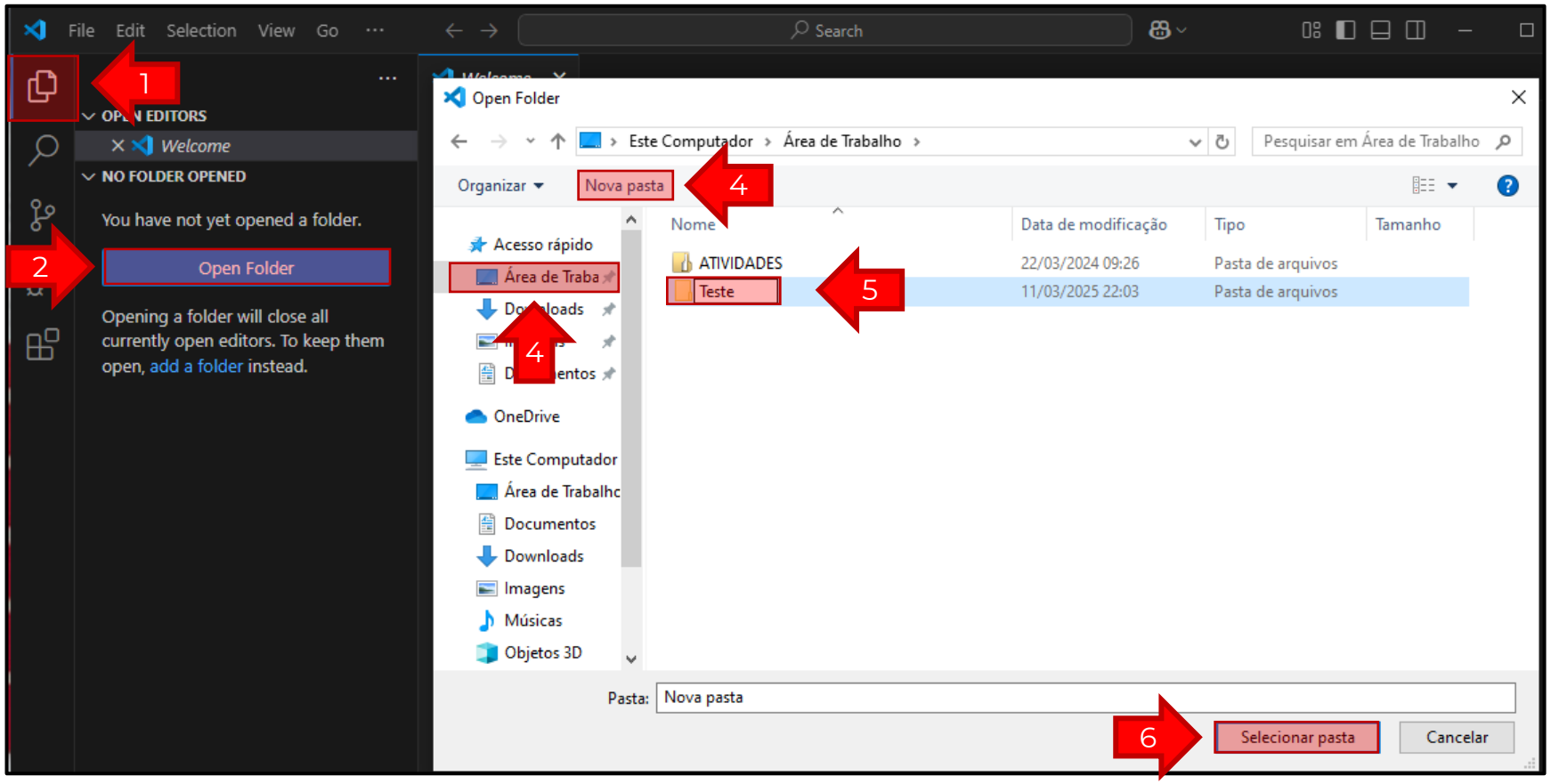
Configuração no Visual Studio Code

Abra o Visual Studio Code, clique em **Extensions** [1]
Procure por **@id:ms-vscode.cpptools-extension-pack** [2]
Clique em **Install** [3]



Primeiro Programa em C++

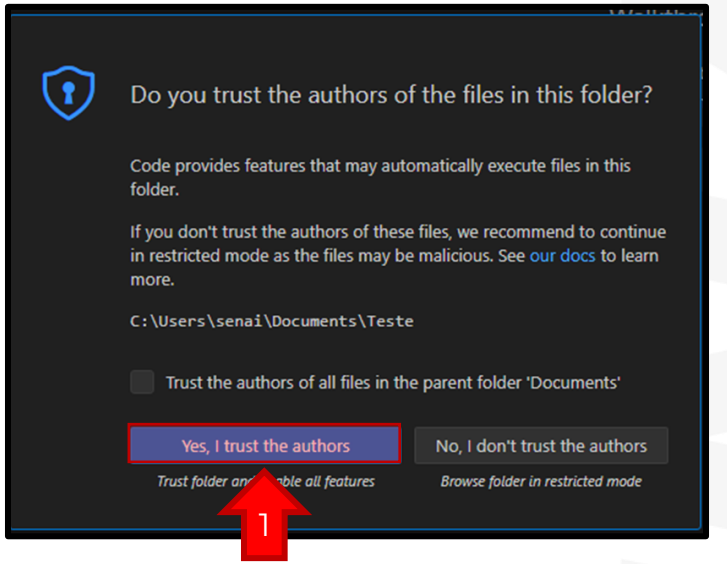
No **Visual Studio Code**, clique no **Explorer** [1]
Clique em **Open Folder** [2]
Clique em **Área de Trabalho** [3]
Clique em **Nova Pasta** [4]
Digite **Teste** [5] e clique **Selecionar pasta** [6]



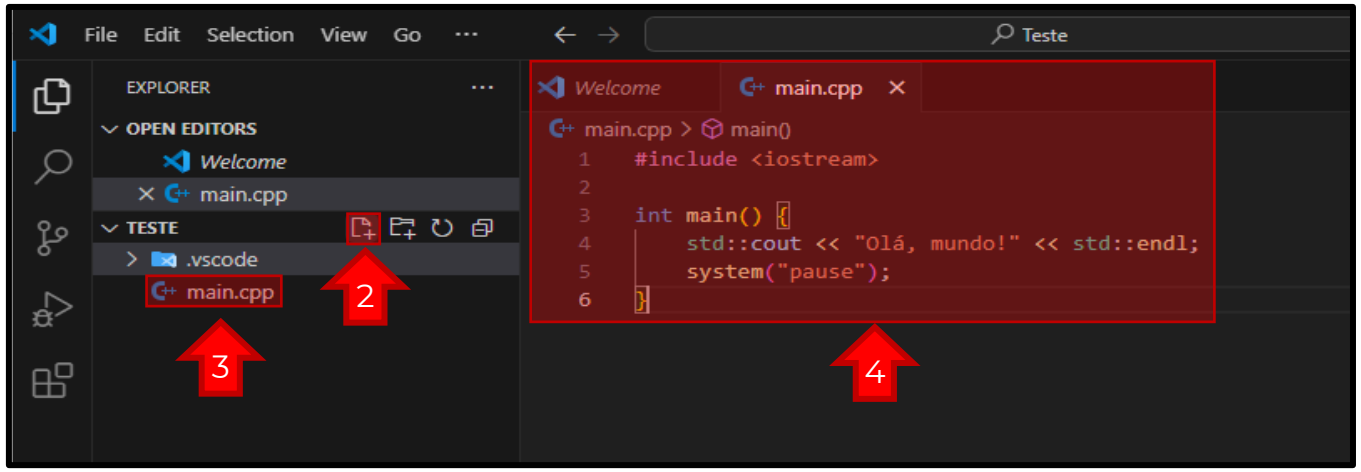
Primeiro Programa em C++

Se aparecer a mensagem perguntando se você confia no autor da pasta, clique em **Yes, I trust the authors** [1]
Clique em **New File...** [2]
Digite **main.cpp** [3]
Digite o código de teste [4]:

```
#include <iostream>

int main() {
    std::cout << "Olá, mundo!" << std:: endl;
    system("pause");
}
```



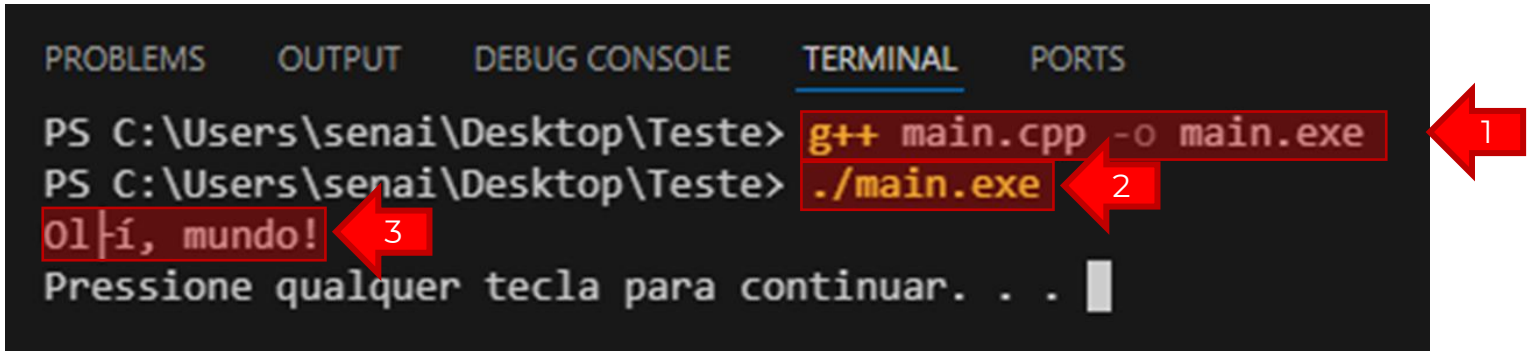
Pressione **CTRL + S** para salvar o arquivo



Primeiro Programa em C++

Pressione **CTRL + J** para abrir o Terminal
Para compilar, digite no terminal: **g++ main.cpp -o main.exe** [1] e pressione **ENTER**
Para executar, digite no terminal: **./main.exe** [2] e pressione **ENTER**

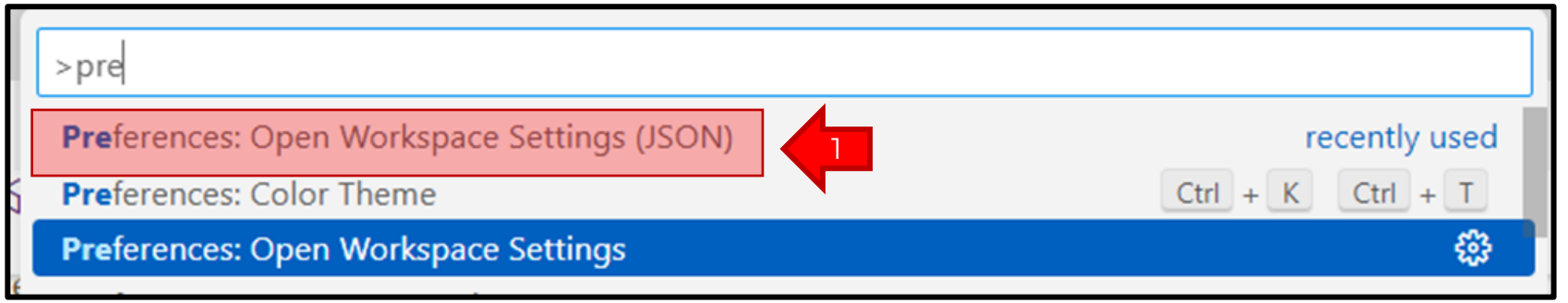
Se tudo foi instalado corretamente, deverá aparecer a mensagem abaixo [3]



Se as letras com acento aparecerem como na imagem [3] siga os próximos passos

Primeiro Programa em C++

Pressione **CTRL + SHIFT + P** e procure por **Preferences: Open Workspace Settings** [1]
Insira o texto abaixo e pressione **CTRL + S** para salvar [2]
Feche a aba settings.json




```
{
    "files.encoding": "iso88591"
}
```

2

Primeiro Programa em C++

Altere o código para incluir a região, **compile** e **execute novamente** [1] :

```
#include <iostream>
#include <locale>

int main() {
   setlocale(LC_ALL, "Portuguese_Brazil");

   std::cout << "Olá, mundo!" << std::endl;
   system("pause");
}
```

```
PS D:\OneDrive - SESISENAISP - Corporativo\SENAI\FIC\C++\Programas> g++ main.cpp -o main.exe
PS D:\OneDrive - SESISENAISP - Corporativo\SENAI\FIC\C++\Programas> ./main.exe
Olá, mundo!
Pressione qualquer tecla para continuar. . .
```

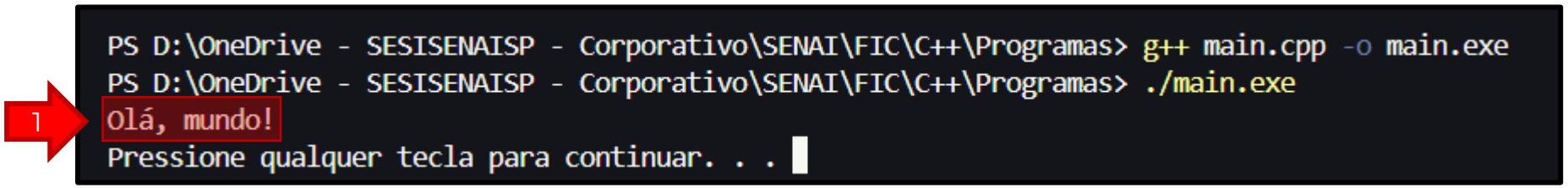
1

Primeiro Programa em C++

Caso ainda não dê certo, digite antes no terminal **chcp 65001** para mudar o idioma do console para UTF-8.

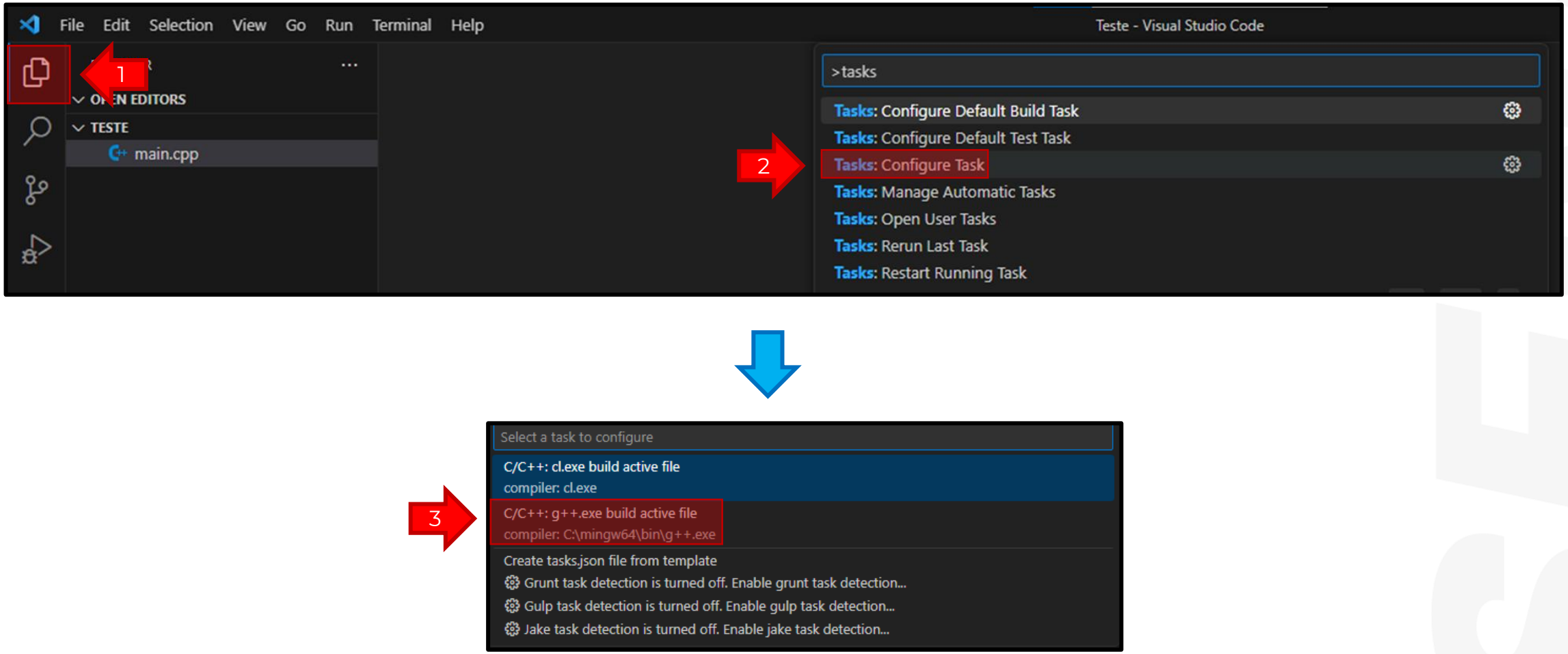
Altere o código para:

```
#include <iostream>
#include <locale>

int main() {
   setlocale(LC_ALL, “pt-BR");

   std::cout << "Olá, mundo!" << std::endl;
   system("pause");
}
```



Configuração adicional

Com a pasta de teste aberta, clique no **Explorer** [1]
Digite **CTRL + SHIFT + P** e busque por **Tasks: Configure Task** [2]
Selecione a opção que usa o **mingw64** como compilador [3]



Configuração adicional

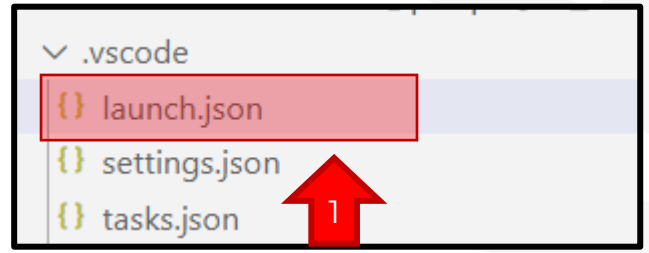
Substitua o texto do arquivo **tasks.json** [1] e pressione **CTRL + S** para salvar:

```
{
    "version": "2.0.0",
    "tasks": [
        {
            "label": "Compilar C++",
            "type": "shell",
            "command": "g++",
            "args": [
                "-fdiagnostics-color=always",
                "-g",
                "${file}",
                "-o",
                "${fileDirname}/${fileBasenameNoExtension}.exe"
            ],
            "options": {
                "cwd": "${fileDirname}"
            },
            "group": {
                "kind": "build",
                "isDefault": true
            },
            "problemMatcher": ["$gcc"]
        }
    ]
}
```

Configuração adicional

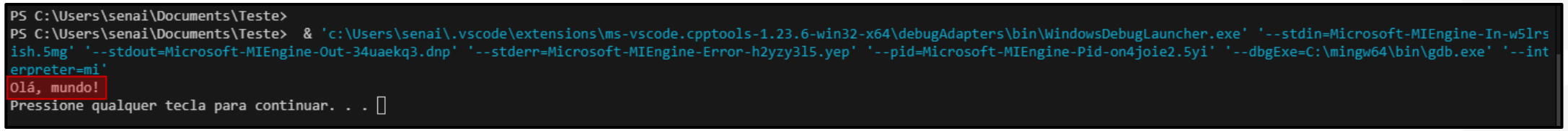
Crie um arquivo **launch.json** na pasta **.vscode** [1]
Insira o texto abaixo [2] e pressione **CTRL + S** para salvar :

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Executar C++",
            "type": "cppdbg",
            "request": "launch",
            "program": "${fileDirname}/${fileBasenameNoExtension}.exe",
            "args": [],
            "stopAtEntry": false,
            "cwd": "${fileDirname}",
            "environment": [
                {
                    "name": "LC_ALL",
                    "value": "pt_BR.UTF-8"
                }
            ],
            "externalConsole": true,
            "MIMode": "gdb",
            "setupCommands": [
                {
                    "description": "Ativar impressão de formato tradicional para gdb",
                    "text": "-enable-pretty-printing",
                    "ignoreFailures": true
                }
            ]
        }
    ]
}
```



Configuração adicional

De volta no arquivo **main.cpp**, pressione CTRL + SHIFT + B para compilar
Pressione F5 para Executar
Se tudo deu certo irá ficar como abaixo no **Terminal**:



OBS: Verifique se a parte debaixo está no **Terminal**, se não tiver altere para a janela do **Terminal**

Entendendo a sintaxe básica

Cabeçalho onde se incluem bibliotecas e determina algumas configurações do programa

```
#include <iostream>
#include <locale>
```

Importação da biblioteca **iostream** para controle de entrada e saída de texto

Importação da biblioteca **locale** para configuração de idioma e codificação de caracteres (á, ã, ç, etc)

Bloco da função principal (main)
*Obrigatório

```
int main() {
    setlocale(LC_ALL, "Portuguese_Brazil");
    std::cout << "Olá, mundo!" << std::endl;
    system("pause");
}
```

Configuração do idioma

Saída de texto

Comando para pausar o programa e não fechar automaticamente (Windows)

Operadores matemáticos

Para executar contas matemáticas em C++, pode-se utilizar os operadores descritos na tabela abaixo:

Operador	Operação	Exemplo	Resultado
+	Adição	5 + 3	8
-	Subtração	5 - 3	2
*	Multiplicação	5 * 3	15
/	Divisão	5 / 2	2.5
%	Módulo	5 % 2	1

Operadores matemáticos

A linguagem C++ segue os mesmos princípios da matemática ao realizar as contas, então, por exemplo, a multiplicação ocorre antes da adição.

```
5 * 3 + 2 = 17
```

A utilização de parênteses determina a ordem de resolução, assim como na matemática

```
5 * (3 + 2) = 25
```

Operadores relacionais

Para executar comparações é necessário utilizar um operador relacional, que irá retornar **verdadeiro** ou **falso** dependendo da comparação executada

Operador	Operação	Exemplo	Resultado
==	Igual a	5 == 3	0 (**falso**)
>=	Maior ou igual a	5 >= 3	1 (**verdadeiro**)
>	Maior que	5 > 3	1 (**verdadeiro**)
<=	Menor ou igual a	5 <= 2	0 (**falso**)
<	Menor que	5 < 2	0 (**falso**)
!=	Diferente de	5 != 2	1 (**verdadeiro**)

SENAI

Operadores relacionais

É possível realizar comparações com contas matemáticas

```
5 + 3 == 15 = 0
```

```
5 * 3 == 15 = 1
```

Também é possível comparar tipos de dados diferentes

```
'5' == 5 = 0
```

```
5.0 == 5 = 1
```

```
72 > 0052.2500 = 1
```

```
"DANIEL" == "Daniel" = 0
```

```
"Curso" != "C++" = 1
```

Variáveis e Tipos de Dados

Existem diversos tipos de dados aplicáveis dentro de um programa, abaixo segue alguns deles:

Tipo	Tamanho (bytes)	Intervalo de Valores | Exemplo
bool	**1 byte**	true (1) ou false (0)
char	**1 byte**	caracteres básicos (a, b, c)
wchar_t	**2 (Windows) ou 4 bytes (Linux)**	caracteres especiais (€, 你, ✓, →)
short	**2 bytes**	-32.768 a 32.767
unsigned short	**2 bytes**	0 a 65.535
int	**4 bytes**	-2.147.483.648 a 2.147.483.647
unsigned int	**4 bytes**	0 a 4.294.967.295
long	**4 bytes**	-2.147.483.648 a 2.147.483.647 (em MinGW)
unsigned long	**4 bytes**	0 a 4.294.967.295
long long	**8 bytes**	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807
unsigned long long	**8 bytes**	0 a 18.446.744.073.709.551.615
float	**4 bytes**	números reais (7 dígitos de precisão)
double	**8 bytes**	números reais (15-16 dígitos de precisão)
long double	**8 a 16 bytes (Depende do compilador e do sistema operacional 12 bytes no nosso caso)**	números reais (18-19 dígitos de precisão)
string	**não nativo, alocação dinâmica (1 byte por caractere)**	texto

Criação de variáveis

Variáveis são **identificadores** que armazenam dados, existem diversos tipos de variáveis, condizentes aos tipos de dados existentes.
O C++ é uma linguagem **fortemente tipada** e com **tipagem estática**, o que quer dizer que temos que definir o tipo da variável ao declarar ela.

```
short ano = 2025;
float dinheiro = 12.5;
short idade = 30;
```

Variáveis do tipo **string** não são nativas, então precisa incluir a biblioteca para utilizar.

```
#include <string>

std::string nome = "Daniel";
```

Criação de variáveis

Uma vez que as variáveis armazenam valores, é possível comparar as variáveis, fazer contas matemáticas com elas ou concatenar (no caso de textos).

```
short num1 = 5;
short num2 = 3;
std::cout << num1 + num2 << std::endl;
```

```
std::string nome = "João";
std::string sobrenome = "Cardoso";

std::string nomeCompleto = nome + " " + sobrenome;

std::cout << nomeCompleto << std::endl;
```

```
nome == "João" = 1
```

Criação de variáveis

Ao criar uma variável deve-se obedecer as seguintes regras:

- Só pode ser uma palavra (sem espaços)
- Só pode utilizar letras, números e sublinha (_)
- Não pode iniciar com um número
- Não pode ser uma palavra-chave (como int, return, include, etc)

Válido	Inválido	Motivo do Erro
idade	2idade	Não pode começar com número
nome_completo	nome completo	Não pode conter espaços
velocidadeMaxima	velocidade-Maxima	O hífen - não é permitido
_contador	int	int é uma palavra-chave
SOMA_TOTAL	$valor	O caractere $ não é permitido

Estilos de Programação

Por questão de organização e padronização, existem estilos predefinidos para criação de variáveis e funções, são eles:

- **camelCase**

Este estilo sempre começa com letras minúsculas e as próximas palavras tem a primeira letra maiúscula: **contaBancaria**, **nomeCompleto**.

- **PascalCase**

Este estilo sempre começa com a primeira letra Maiúscula em cada palavra: **ContaBancaria**, **NomeCompleto**.

- **snake_case**

Este estilo sempre utiliza letras minúsculas e separa as palavras com **sublinha (_)**: **conta_bancaria**, **nome_completo**

- **kebab-case**

Este estilo sempre utiliza letras minúsculas e separa as palavras com **hífen (-)**: **conta-bancaria**, **nome-completo**

Estilos de Programação

Para seguir um padrão, vamos utilizar:

- **camelCase** para **variáveis**, **funções** e **métodos**;
- **PascalCase** para **classes**;
- **SCREAMING_SNAKE_CASE** para constantes.

Namespace

Espaços de nomes (**namespaces**) que agrupam identificadores (funções, classes, variáveis) para evitar conflitos principalmente em projetos grandes.

Em projetos menores podemos utilizar a instrução para acessar um escopo de namespace para melhorar a legibilidade, mas isso deve ser evitado em projetos grandes.

```
#include <iostream>
#include <locale>

using namespace std;

int main() {
    setlocale(LC_ALL, "Portuguese_Brazil");
    cout << "Olá, mundo!" << endl;

    system("pause");
}
```

Entrada e Saída de Dados

No C++, a entrada e saída de dados é utilizada com a biblioteca **iostream**.

cin >>: Captura entrada do usuário.
cout <<: Exibe saída no console.
endl : (end line) Pula para a próxima linha (ENTER)

```
#include <iostream>
#include <locale>

using namespace std;

int main() {
    setlocale(LC_ALL, "Portuguese_Brazil");

    int idade;
    cout << "Digite sua idade: ";
    cin >> idade;
    cout << "Você tem " << idade << " anos." << endl;

    system("pause");
}
```

Entrada e Saída de Dados

Se for guardar algum texto composto em uma **string** é necessário utilizar a função **getline**.

```
#include <iostream>
#include <locale>
#include <string>

using namespace std;

int main() {
    setlocale(LC_ALL, "Portuguese_Brazil");

    cout << "Digite seu nome completo: ";

    string nome;
    getline(cin, nome);

    cout << "Seja bem vindo " << nome;

    system("pause");
}
```

Tamanho de texto

O método **.length()** serve para contar o número de caracteres de um texto.

```
#include <iostream>
#include <locale>
#include <string>

using namespace std;

int main() {
    setlocale(LC_ALL, "Portuguese_Brazil");

    string nome;

    cout << "Digite seu nome: ";
    getline(cin, nome);

    int numCaracteres = nome.length();

    cout << "Seu nome tem " << numCaracteres << " caracteres" << endl;

    system("pause");
}
```

Fixar precisão de um número real

Para fixar o número de casas decimais de um número real podemos utilizar a biblioteca **iomanip**.

```
#include <iostream>
#include <iomanip>
#include <locale>

using namespace std;

int main() {
    setlocale(LC_ALL, "Portuguese_Brazil");

    float temperatura = 27.2513587;

    cout << fixed << setprecision(2);
    cout << temperatura << endl;

    system("pause");
}
```

Comentários

Os comentários são utilizados para anotar alguma informação pro **programador**, será ignorado durante a **compilação** do **código**:

```
/*
================================================================
Programa: Formatação de Número com Casas Decimais
Autor: Daniel
Data: 18/03/2025
Descrição: Este programa exibe uma temperatura formatada com
           duas casas decimais usando `setprecision` em C++.
================================================================
*/

#include <iostream>  // Biblioteca padrão para entrada e saída de dados
#include <iomanip>   // Biblioteca para manipulação de formatação de saída
#include <locale>    // Biblioteca para configurar o idioma e regionalização

using namespace std;

int main() {
    setlocale(LC_ALL, "Portuguese_Brazil"); // Configura o idioma para exibir caracteres acentuados corretamente

    float temperatura = 27.2513587; // Declara uma variável float com um valor decimal

    cout << fixed << setprecision(2); // Define a saída para mostrar números fixos com 2 casas decimais

    cout << temperatura << endl; // Exibe a temperatura formatada com apenas 2 casas decimais

    system("pause"); // Pausa o sistema para visualizar a saída (apenas no Windows)
}
```