embarca tech | Residência Tecnológica em Sistemas Embarcados

# Visual Studio Code

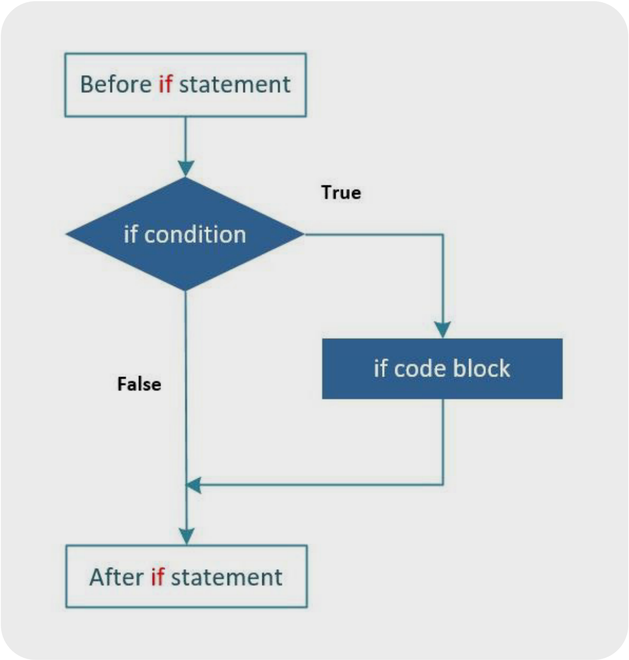
Unidade 3 | Capítulo 2

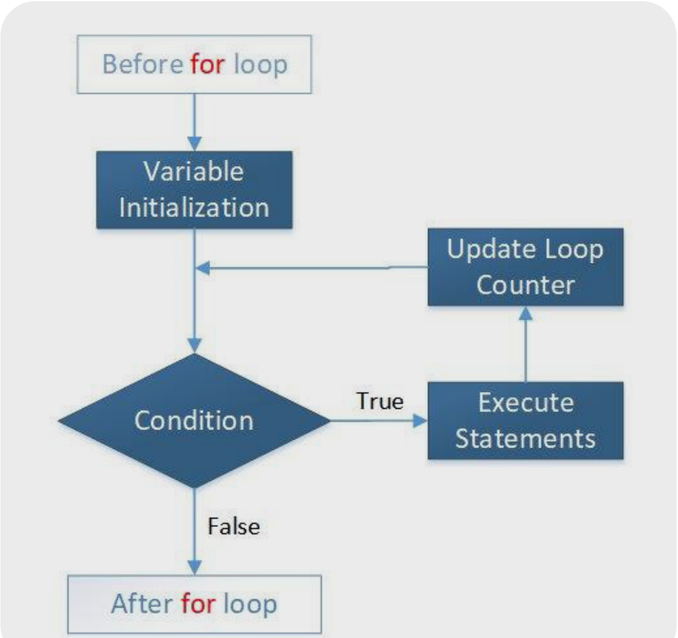
Executores:
INSTITUTO FEDERAL Piauí
INSTITUTO FEDERAL Rio Grande do Norte
INSTITUTO FEDERAL Maranhão
INSTITUTO FEDERAL Ceará
cepedi
INSTITUTO HARDWARE BR

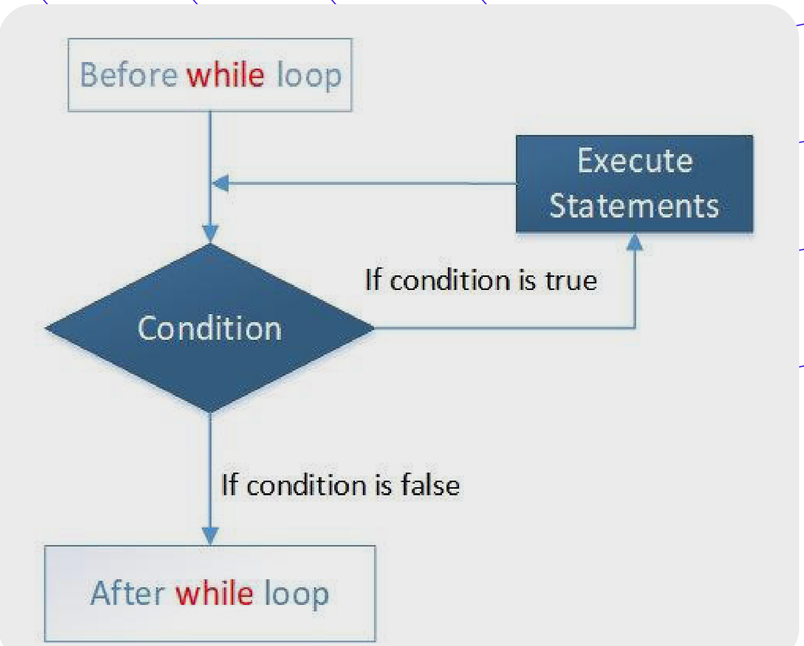
Coordenação: Softex

Iniciativa: MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO
GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

# Na aula passada...



# Nesta aula...

Compreender as funcionalidades e a correta forma de configuração e utilização do VS Code, importante para avaliar os códigos desenvolvidos inicialmente.

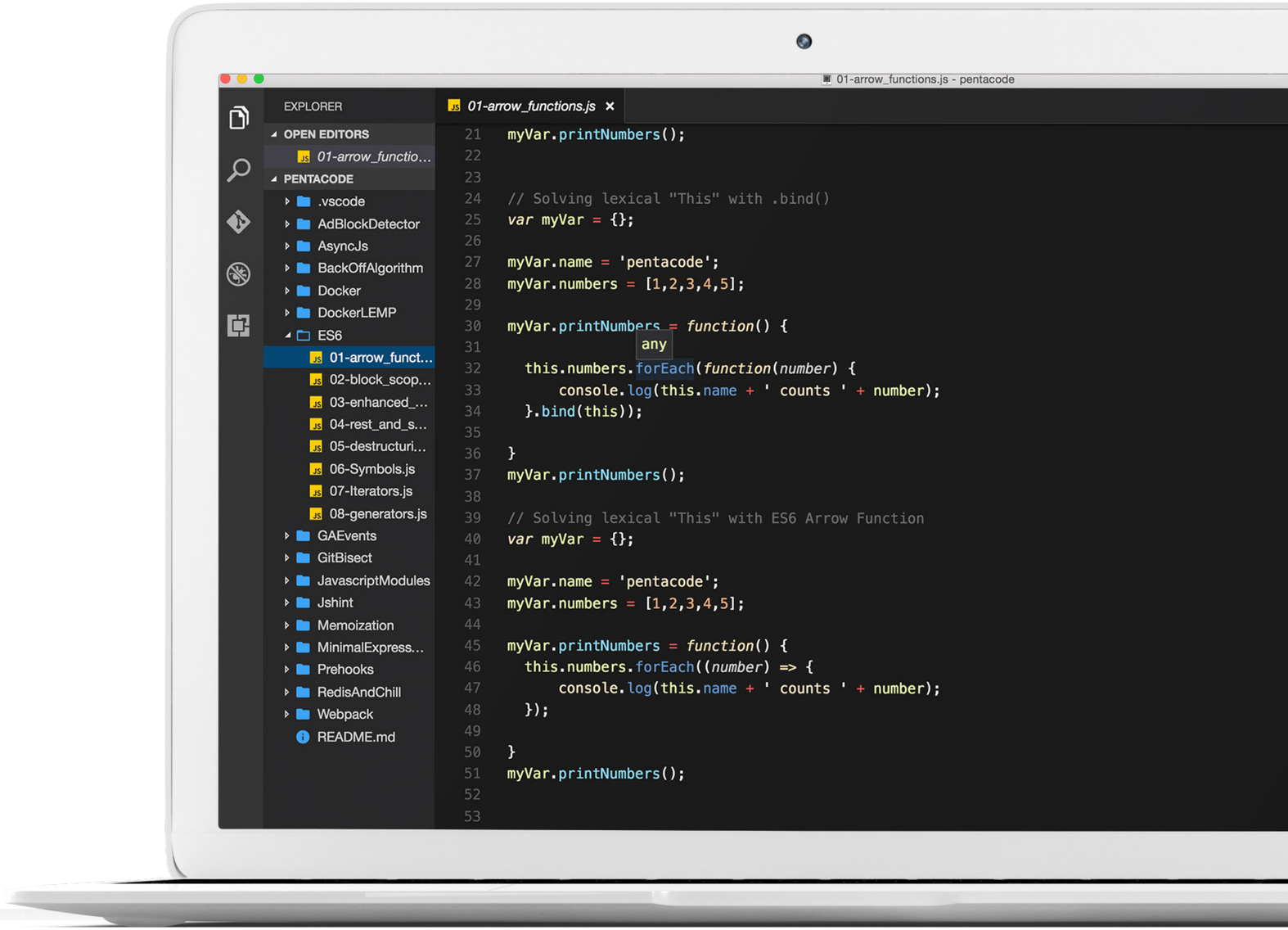
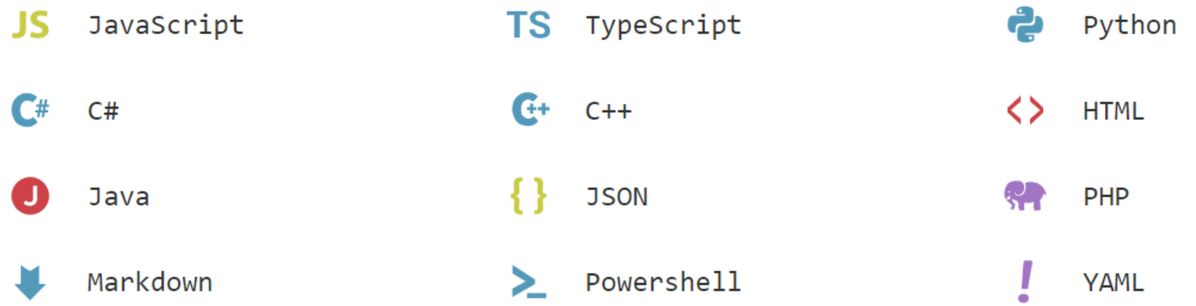


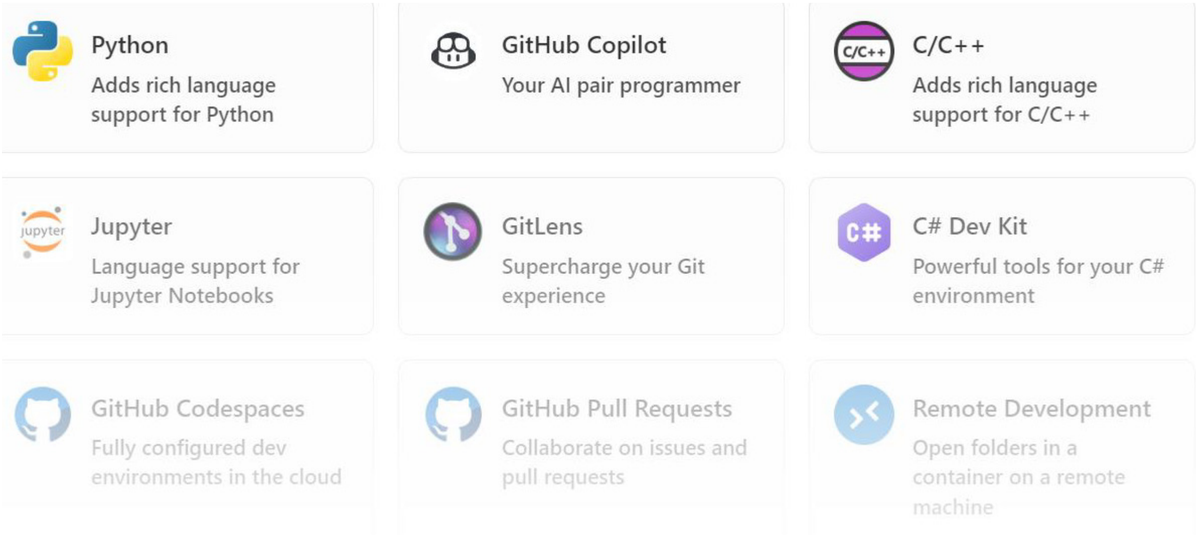

Imagem: fadguru.weebly.com

# Sobre o VS Code

Diversas Linguagens de Programação



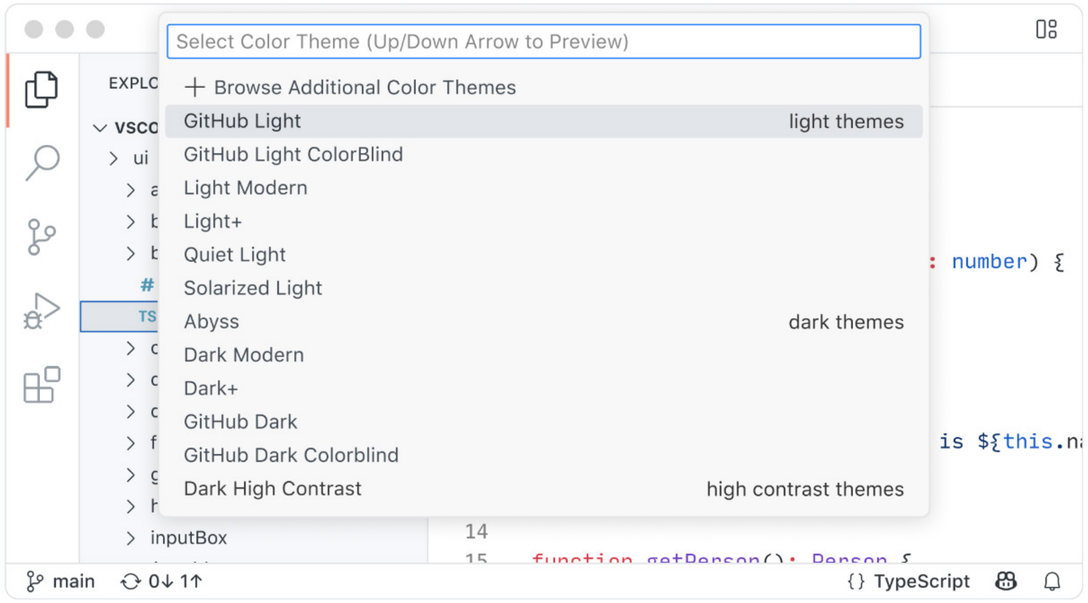

Plugins oficiais e da comunidade

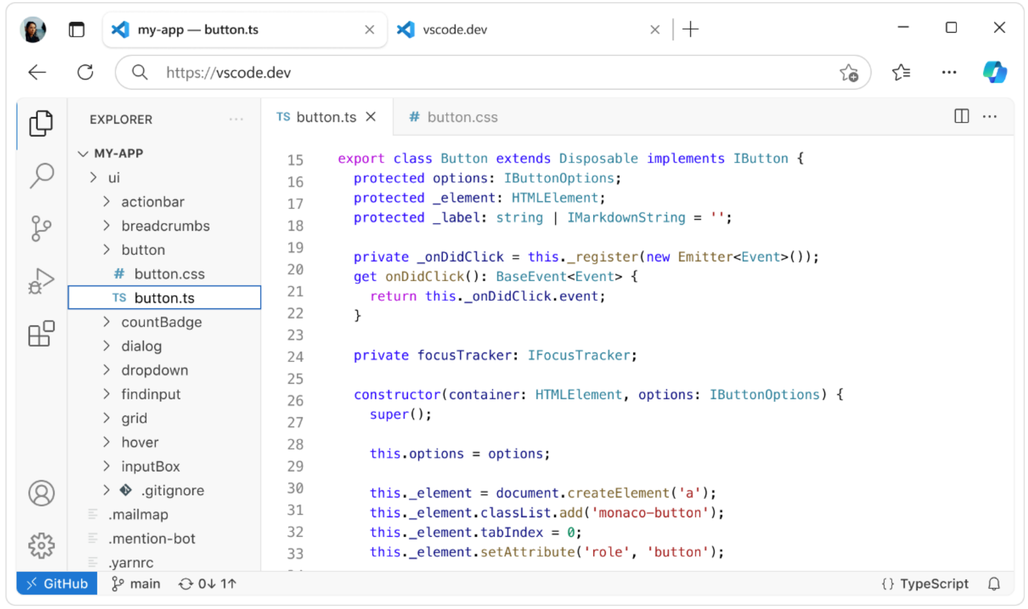



https://code.visualstudio.com

# Sobre o VS Code

https://code.visualstudio.com

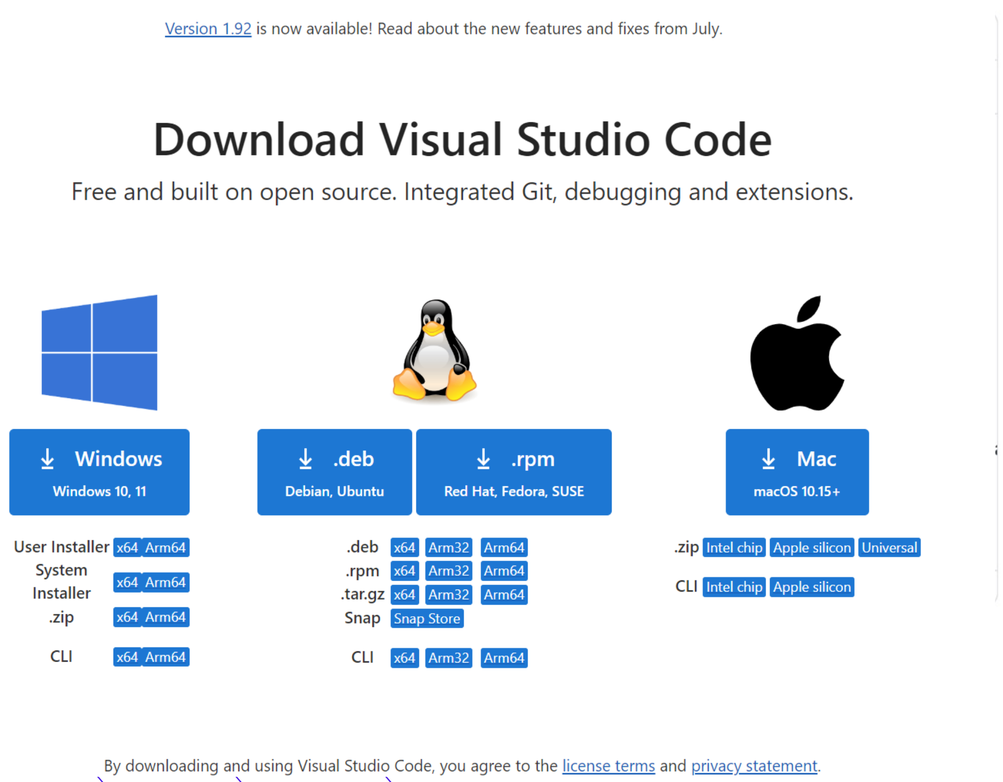


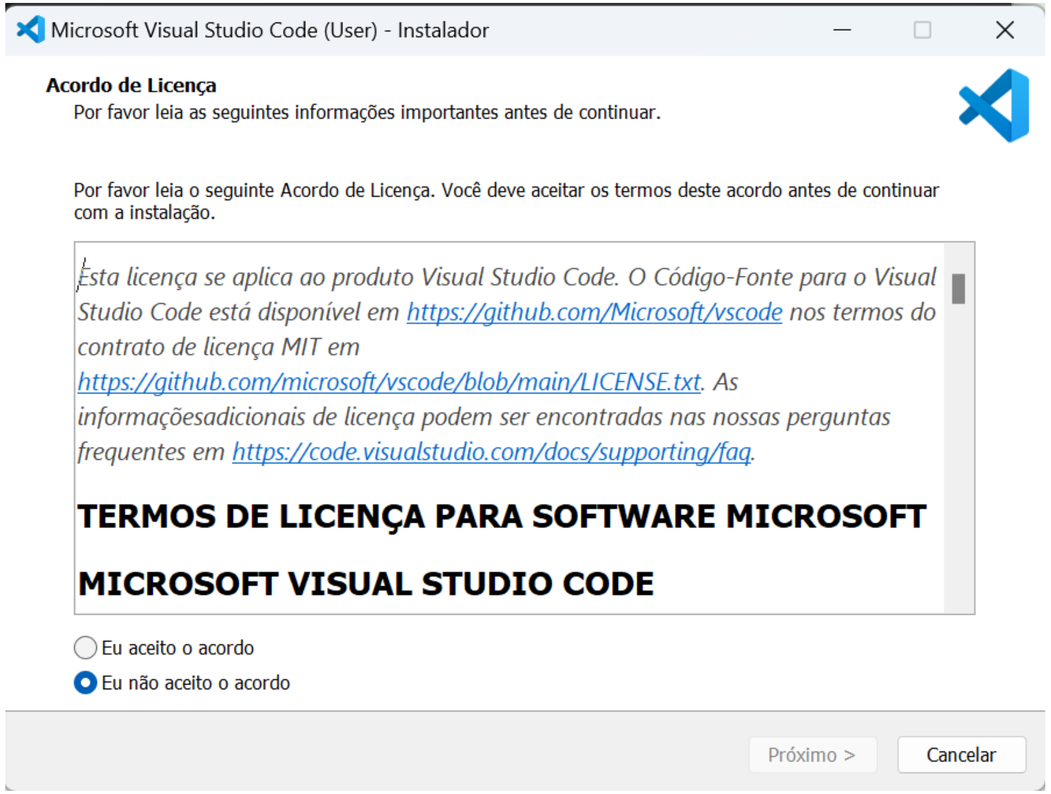




Tema da interface com muitas opções de customização

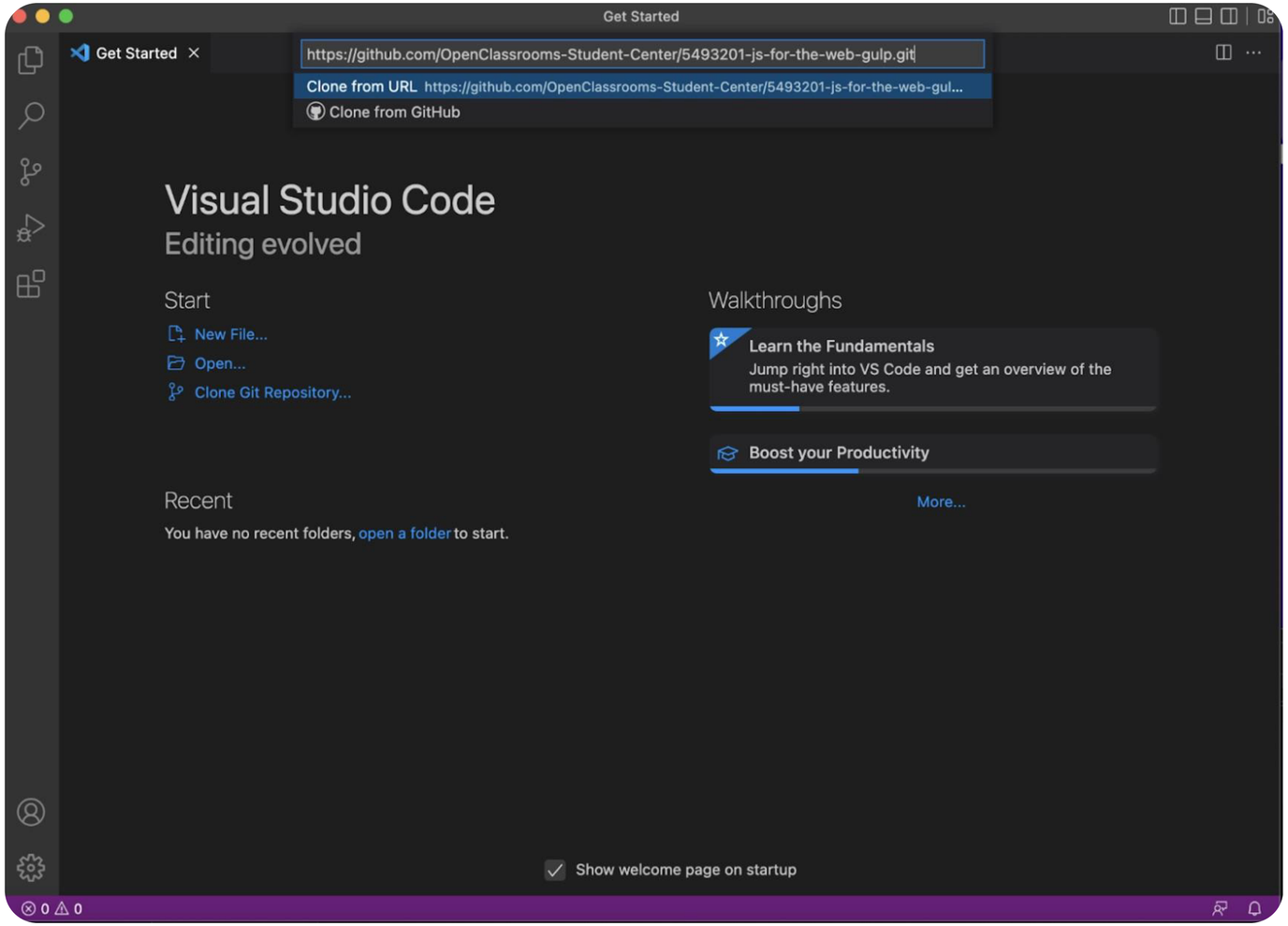
Integração multiplataforma, inclusive web

# Baixando e Instalando o VS Code

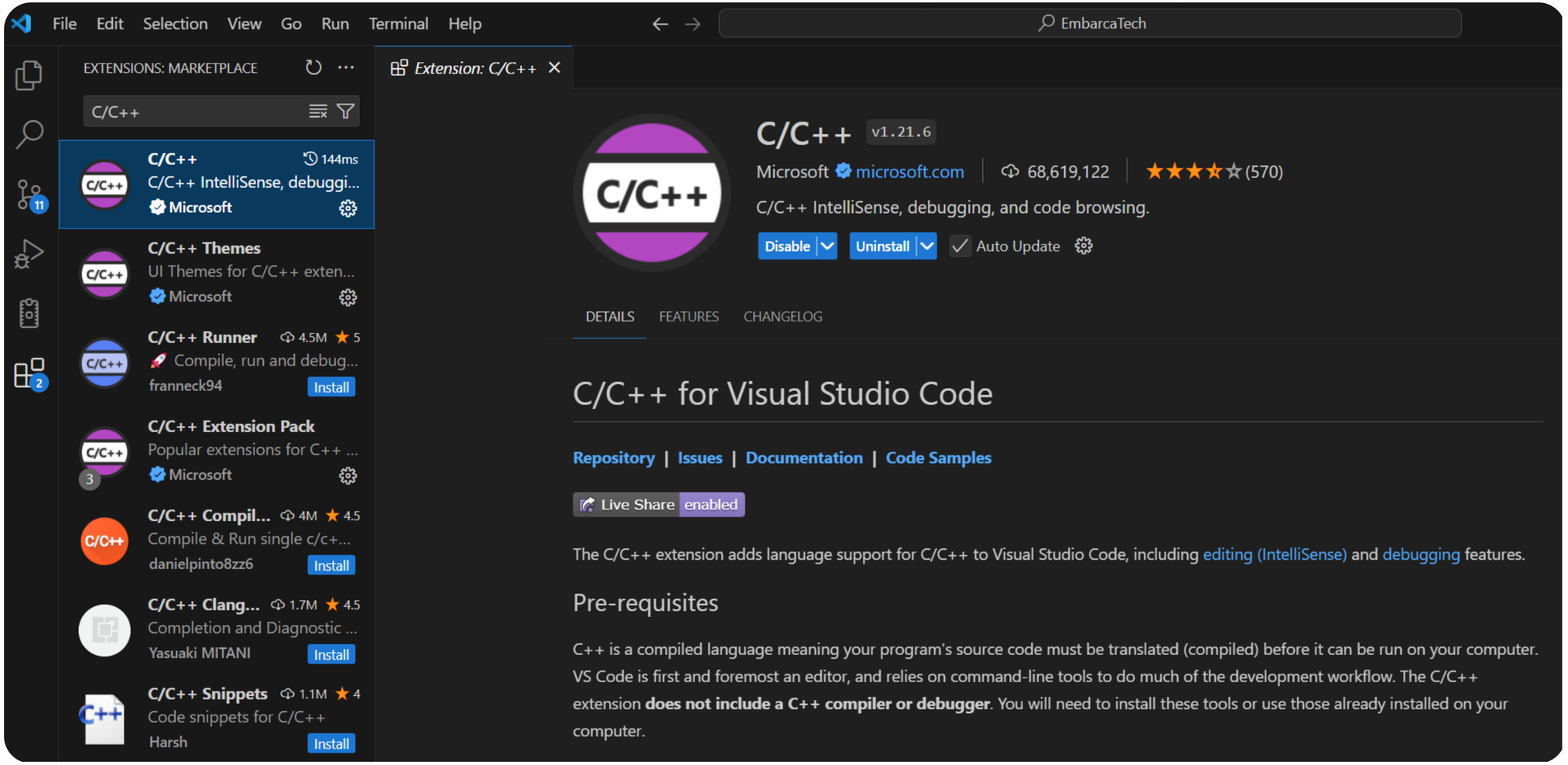


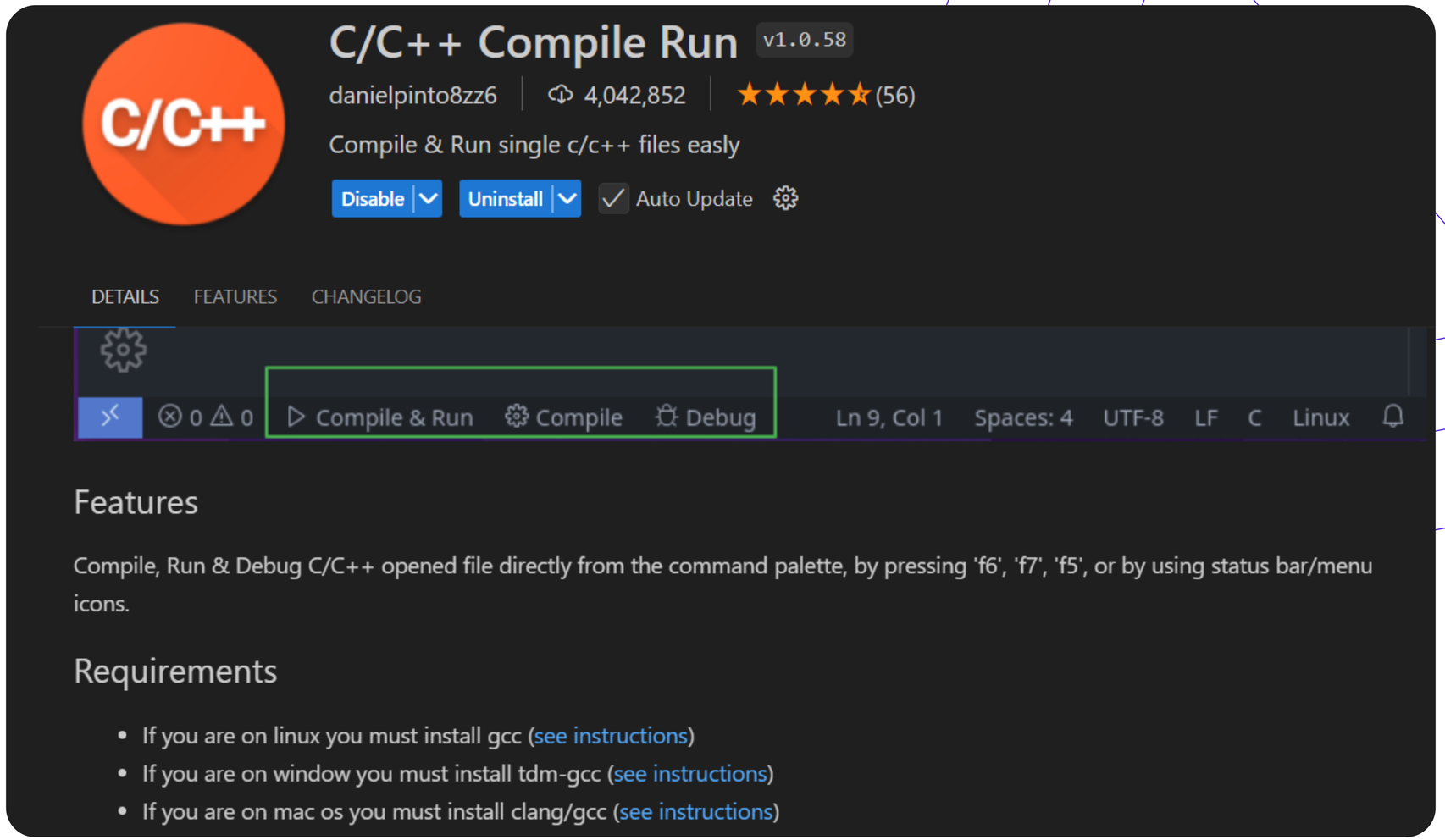
# Baixando e Instalando o VS Code



# Plugin


File Edit Selection View Go Run Terminal Help
EmbarcaTech
EXTENSIONS: MARKETPLACE
C/C++
C/C++ 144ms
C/C++ IntelliSense, debuggi...
Microsoft
C/C++ Themes
UI Themes for C/C++ exten...
Microsoft
C/C++ Runner 4.5M ★ 5
Compile, run and debug...
franneck94 Install
C/C++ Extension Pack
Popular extensions for C++ ...
Microsoft
C/C++ Compil... 4M ★ 4.5
Compile & Run single c/c+...
danielpinto8zz6 Install
C/C++ Clang... 1.7M ★ 4.5
Completion and Diagnostic ...
Yasuaki MITANI Install
C/C++ Snippets 1.1M ★ 4
Code snippets for C/C++
Harsh Install
Extension: C/C++
C/C++ v1.21.6
Microsoft microsoft.com 68,619,122 ★★★★☆ (570)
C/C++ IntelliSense, debugging, and code browsing.
Disable Uninstall Auto Update
DETAILS FEATURES CHANGELOG
C/C++ for Visual Studio Code
Repository | Issues | Documentation | Code Samples
Live Share enabled
The C/C++ extension adds language support for C/C++ to Visual Studio Code, including editing (IntelliSense) and debugging features.
Pre-requisites
C++ is a compiled language meaning your program's source code must be translated (compiled) before it can be run on your computer. VS Code is first and foremost an editor, and relies on command-line tools to do much of the development workflow. The C/C++ extension does not include a C++ compiler or debugger. You will need to install these tools or use those already installed on your computer.

# Plugin


C/C++ Compile Run
v1.0.58
danielpinto8zz6
4,042,852
(56)
Compile & Run single c/c++ files easly
Disable
Uninstall
Auto Update
DETAILS
FEATURES
CHANGELOG
0 0
Compile & Run
Compile
Debug
Ln 9, Col 1
Spaces: 4
UTF-8
LF
C
Linux
Features
Compile, Run & Debug C/C++ opened file directly from the command palette, by pressing 'f6', 'f7', 'f5', or by using status bar/menu icons.
Requirements
If you are on linux you must install gcc (see instructions)
If you are on window you must install tdm-gcc (see instructions)
If you are on mac os you must install clang/gcc (see instructions)

# Compilador

C/C++ Compile Run
v1.0.58
danielpinto8zz6
4,042,852
(56)
Compile & Run single c/c++ files easly
Disable
Uninstall
Auto Update
DETAILS
FEATURES
CHANGELOG
0 0
Compile & Run
Compile
Debug
Ln 9, Col 1
Spaces: 4
UTF-8
LF
C
Linux
Features
Compile, Run & Debug C/C++ opened file directly from the command palette, by pressing 'f6', 'f7', 'f5', or by using status bar/menu icons.
Requirements
If you are on linux you must install gcc (see instructions)
If you are on window you must install tdm-gcc (see instructions)
If you are on mac os you must install clang/gcc (see instructions)

# Compilador: gcc

## GCC Setup

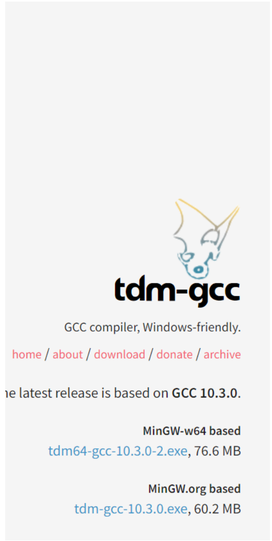
### Table of Contents


1. Windows
2. Linux
3. MacOS
4. WSL Example


### Windows

Install Tdm-gcc via the website. Click Tdm-gcc to download the Windows Tdm-gcc installer.

- Run the installer.
- Select create a new install
- Choose your **Architecture** and then select **Next** until complete.
- Restart your vscode




### Download

The easiest way to get TDM-GCC is via an installer.

**Download a TDM-GCC installer:**

| | |
|---|---|
| tdm-gcc-webdl.exe | Minimal online installer. Select the components you want, and it downloads and unpacks them. Either edition, latest release only. *(GCC 10.3.0)* |
| tdm64-gcc-10.3.0-2.exe | 64+32-bit MinGW-w64 edition. Includes GCC C/C++, GNU binutils, mingw32-make, GDB (64-bit), the MinGW-w64 runtime libraries and tools, and the windows-default-manifest package. |
| tdm-gcc-10.3.0.exe | 32-bit-only MinGW.org edition. Includes GCC C/C++, GNU binutils, mingw32-make, GDB (32-bit), the MinGW.org mingwrt and w32api packages, and t windows-default-manifest package. |

The following links are for TDM-GCC releases since the TDM-GCC 9 series. Older versions are still available on SourceForge: TDM-GCC files.

# Linux

- First, check to see wheter GCC is already istalled. To verify whether it is, open a Terminal window and enter the following command:

  `gcc -v`

- If GCC isn't installed, run the following command from the terminal window to update the Ubuntu package lists. An out-of-date Linux distribution can sometimes interfere with attempts to install new packages.

  `sudo apt-get update`

- Next install the GNU compiler tools and the GDB debugger with this command:

  `sudo apt-get install build-essential gdb`

# MacOS

On MacOS you can use gcc or clang, choose the one you prefer.

**gcc**

- Install Homebrew.
- After Homebrew installs, type in the terminal:

`brew install gcc gdb`

**Clang**

- Ensure Clang is installed
- Clang may already be installed on your Mac. TO verify that it is, open a macOS Terminal window and enter the following command:

  `clang - - version`

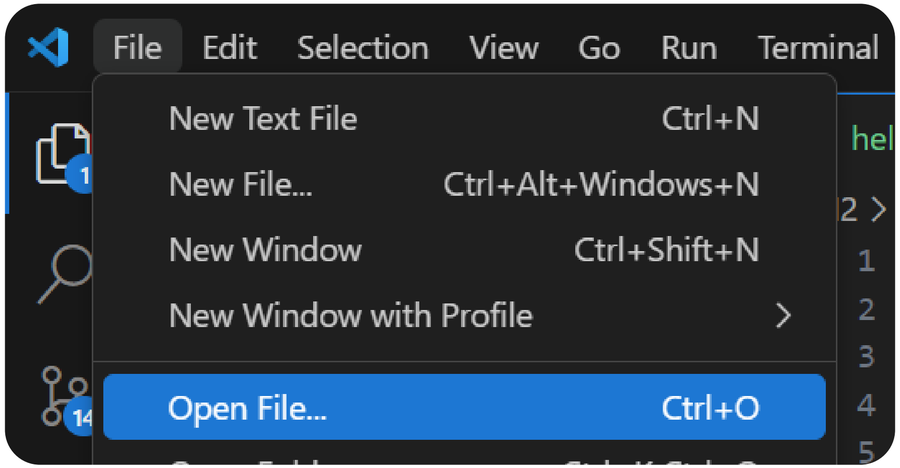
- If Clang isn't installed, enter the following command to install the command line developes tools:

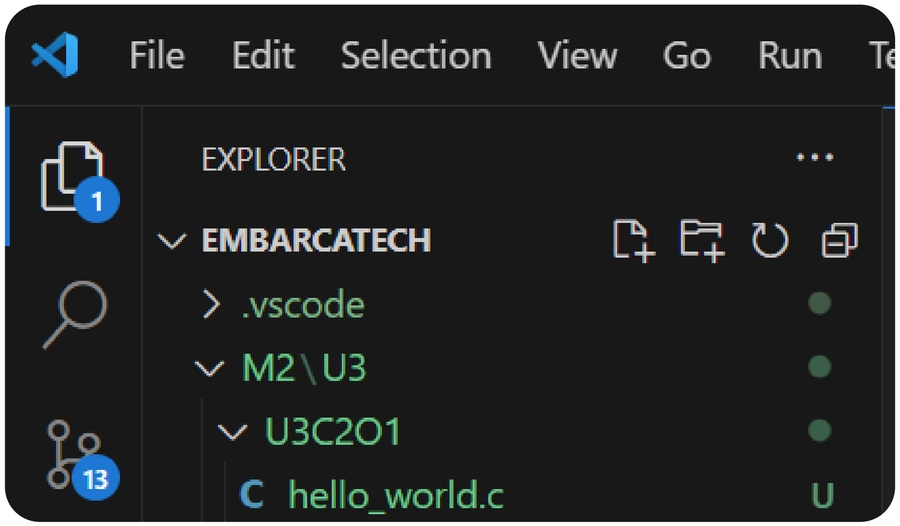
  `xcode-select - -install`

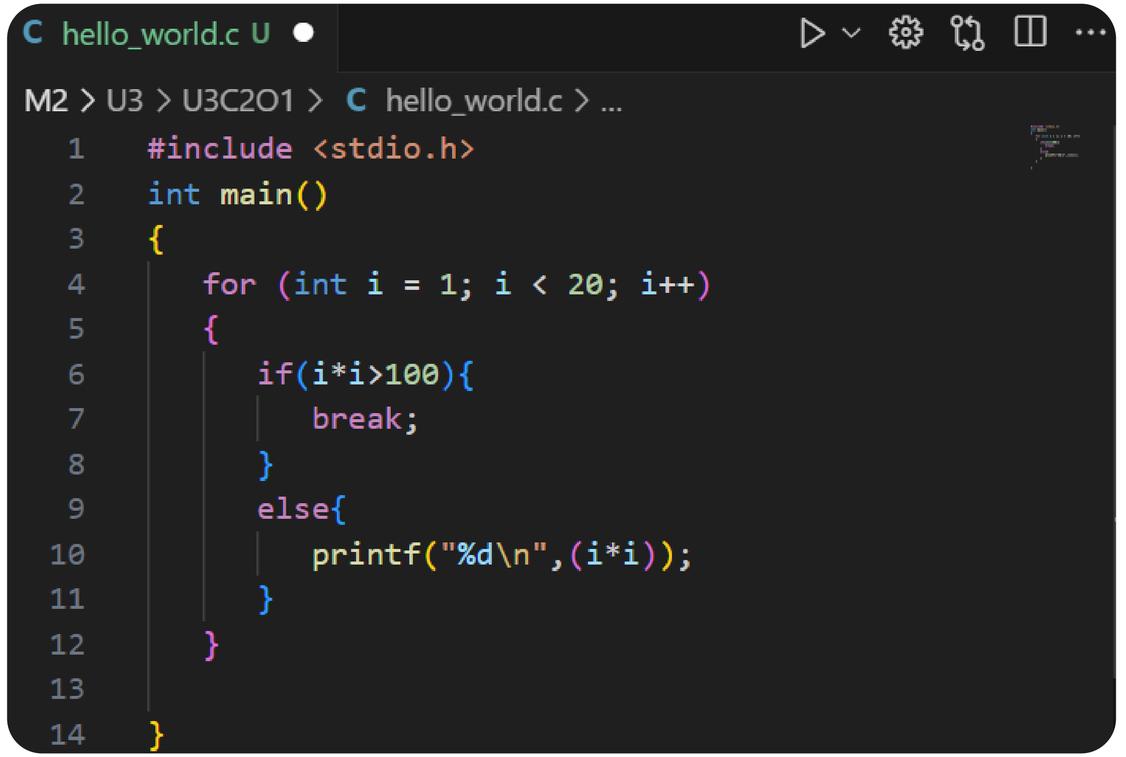
## Configurations

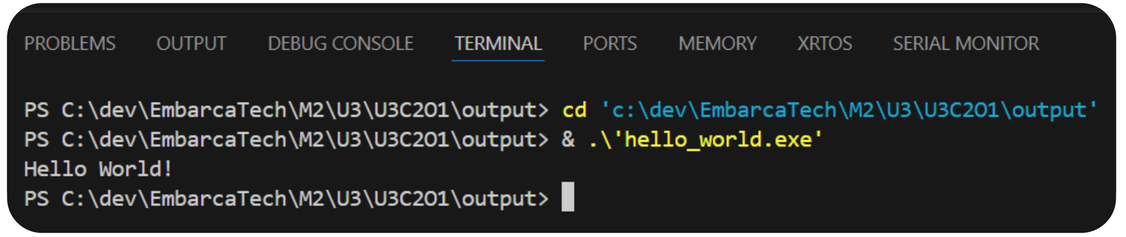
| Key | Description |
|---|---|
| c-cpp-compile-run.c-compiler | The C compiler path (e.g: /usr/bin/gcc or C:\TDM-GCC-64\bin\gcc.exe) |
| c-cpp-compile-run.cpp-compiler | The Cpp compiler path (e.g: /usr/bin/g++ C:\TDM-GCC-64\bin\gcc.exe) |
| c-cpp-compile-run.save-before-compile | Whether should save the file before compiling |
| c-cpp-compile-run.c-flags | The C flags: e.g. -Wall. default: -Wall -Wextra -g3 |
| c-cpp-compile-run.cpp-flags | The Cpp flags: e.g. -Wall. default: -Wall -Wextra -g3 |
| c-cpp-compile-run.run-args | The run arguments |
| c-cpp-compile-run.run-in-external-terminal | Whether should run in an external terminal |
| c-cpp-compile-run.should-show-notifications | Whether should show notifications |
| c-cpp-compile-run.output-location | Custom output location for the compiled file |
| c-cpp-compile-run.custom-run-prefix | Prefix command before run (e.g: valgrind ./foobar) |

# Exemplos: Hello World



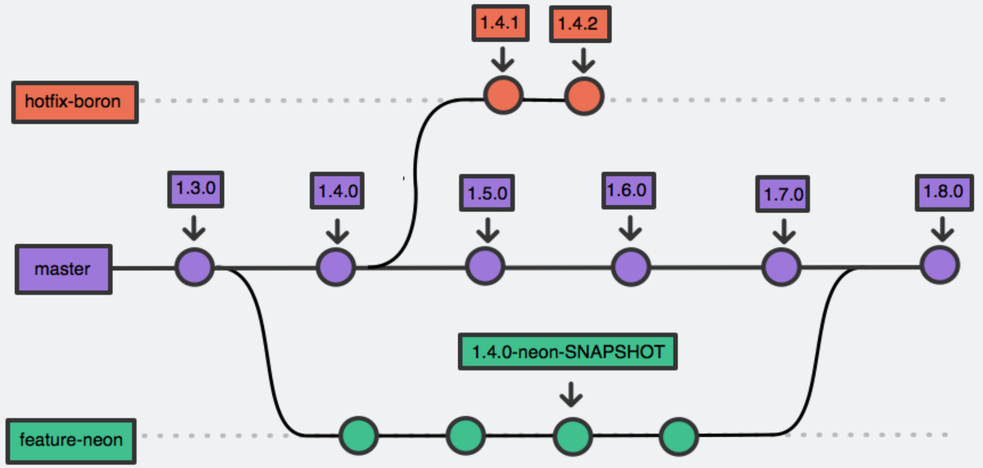
# Exemplos: Loops & Condicionais

```c
#include <stdio.h>

int main()
{
    for (int i = 1; i < 20; i++)
    {
        if(i*i>100){
            break;
        }
        else{
            printf("%d\n",(i*i));
        }
    }
}
```

# Conclusão

Visual Studio Code




https://simon-maxen.medium.com/branch-based-versioning-5ebf6ca2bccb

https://medium.com/@juniortrojilio/preparando-o-vs-code-para-compilar-c-c-no-windows-988f4a91a557

https://github.com/danielpinto8zz6/c-cpp-compile-run/blob/HEAD/docs/COMPILER_SETUP.md#Windows