Ambiente integrado de desenvolvimento

Programação II – Engenharia de Telecomunicações

Prof. Emerson Ribeiro de Mello

mello@ifsc.edu.br

INSTITUTO FEDERAL
Santa Catarina
Campus São José

Licenciamento

Sumário

Ferramentas para a disciplina

Ferramentas para essa disciplina

- **Compilador e automação de projetos**
  - gcc, clang ou mingw
  - make e cmake
- **Ambiente integrado de desenvolvimento**
  - CLion
  - Visual Studio Code
    - C/C++ Extension Pack
- **Materiais de referência sobre a linguagem C**
  - https://wiki.sj.ifsc.edu.br/images/f/fb/Mello-curso-de-c.pdf
  - https://devdocs.io/c
  - Livros da bibliografia complementar

Ambiente integrado de desenvolvimento

***Integrated Development Environment* (IDE)**

Jetbrains CLion

- IDE multiplataforma para C e C++
- Licença gratuita para estudantes do IFSC, clique aqui
- Tutoriais e documentação, clique aqui

Precisará ter instalado o compilador C (i.e. gcc, clang, minGW) e o aplicativo de construção *make*

Ambiente integrado de desenvolvimento

***Integrated Development Environment* (IDE)**

Microsoft Visual Studio Code

- Editor de propósito geral com grande coleção de extensões
  - C/C++ Extension Pack
- Tutoriais e documentação, clique aqui
-  VSCodium é versão de código aberto do VSCode
- Uma alternativa seria usar esse codespace que pode ser executado com o GitHub Codespace

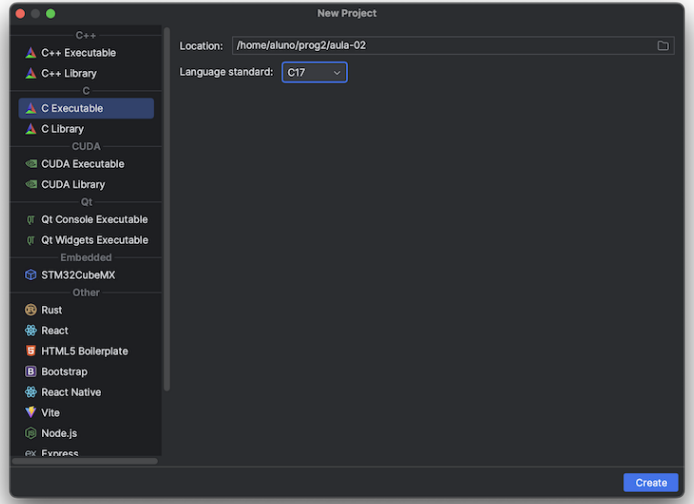
Precisará ter instalado o compilador C (i.e. gcc, clang, minGW), o aplicativo de construção *make* e o `cmake`

Ferramentas para a disciplina

Criando um projeto com CLion, VSCode e terminal

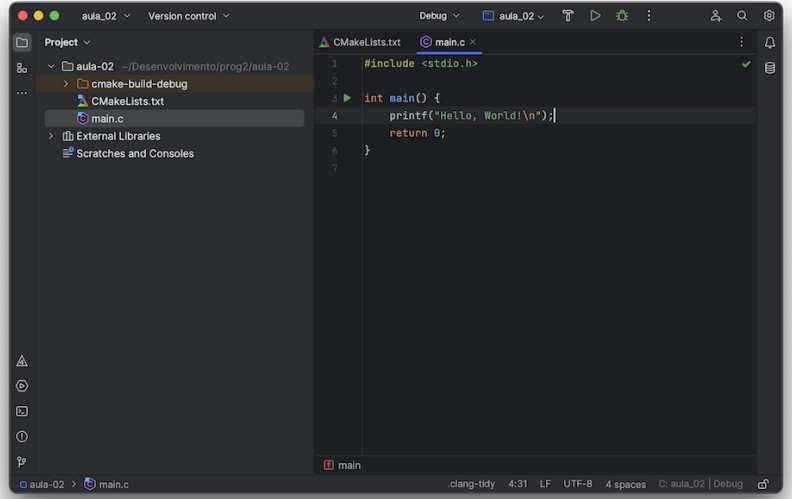
Criando um projeto com CLion

Clique no menu File → New → Project



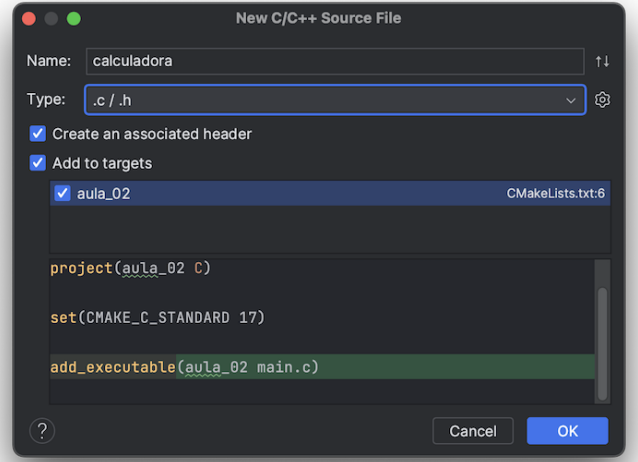
Criando um projeto com CLion

Clique no menu File → New → Project



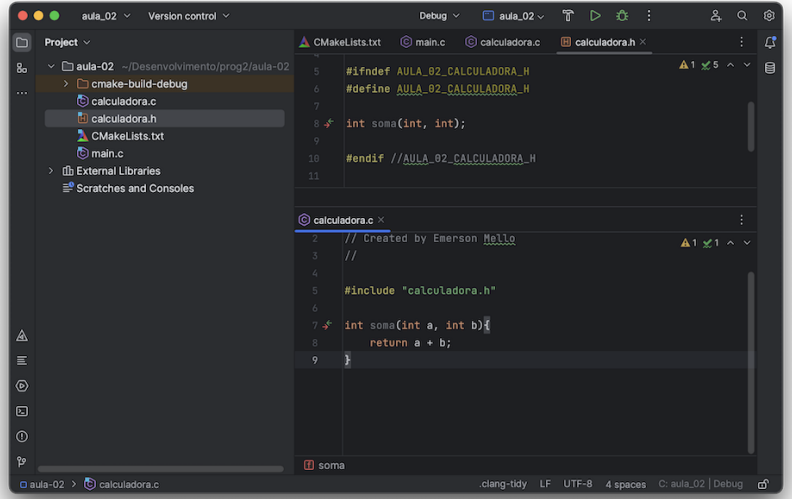
Adicionando o arquivo `calculadora.c` e seu respectivo `.h`

Menu File → New → C/C++ Source File



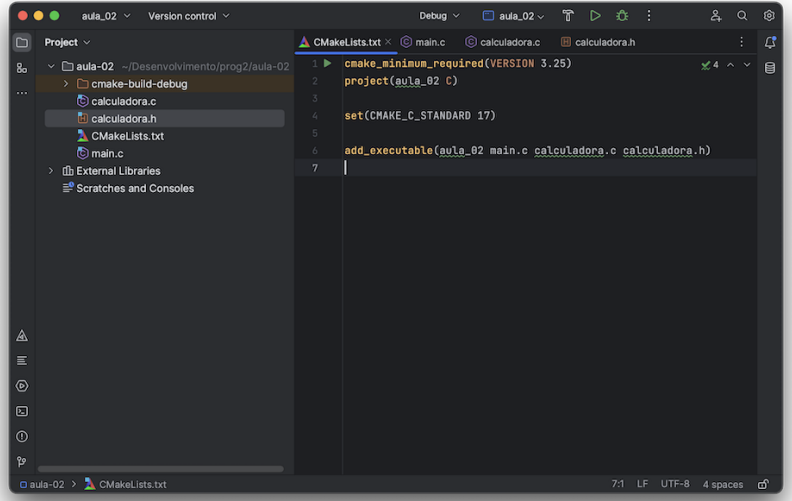
Adicionando o arquivo `calculadora.c` e seu respectivo `.h`

Menu File → New → C/C++ Source File



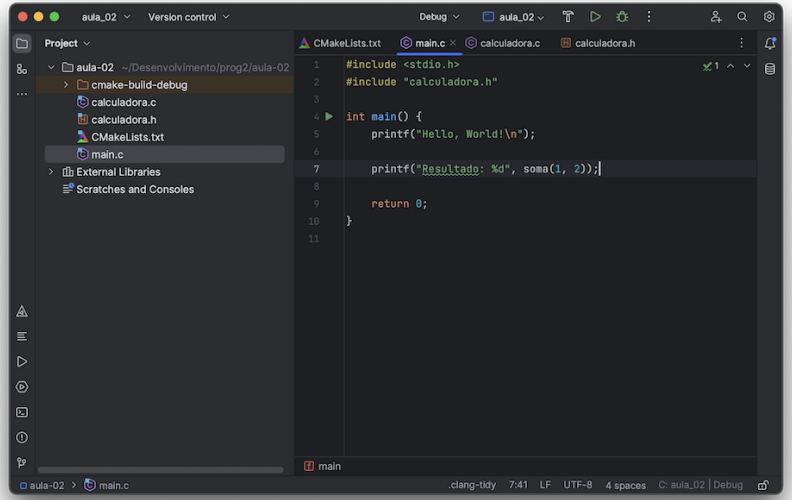
Adicionando o arquivo `calculadora.c` e seu respectivo `.h`

Menu File → New → C/C++ Source File



Adicionando o arquivo `calculadora.c` e seu respectivo `.h`

Menu File → New → C/C++ Source File



Para adicionar biblioteca `math.h`

```
cmake_minimum_required(VERSION 3.25)
project(aula_02 C)

set(CMAKE_C_STANDARD 17)

add_executable(aula_02 main.c calculadora.c calculadora.h)

# Para adicionar a biblioteca math.h
target_link_libraries(aula_02 m)
```

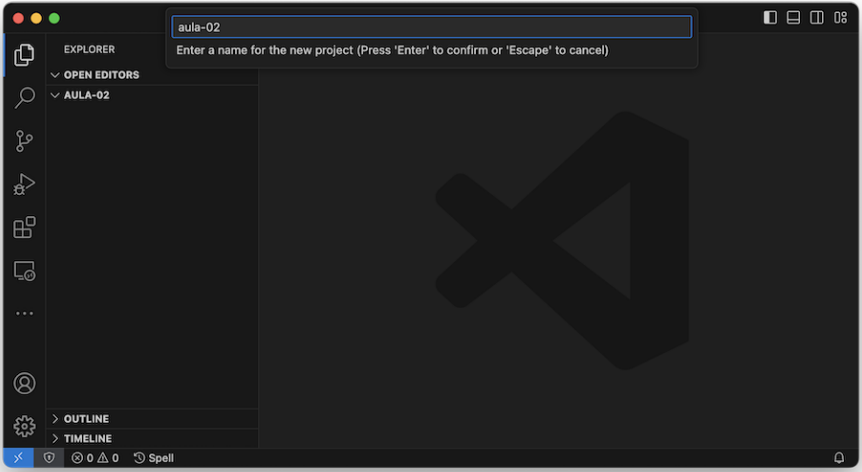
Criando um projeto com o VSCode

- Crie um diretório e abra o diretório com o VSCode

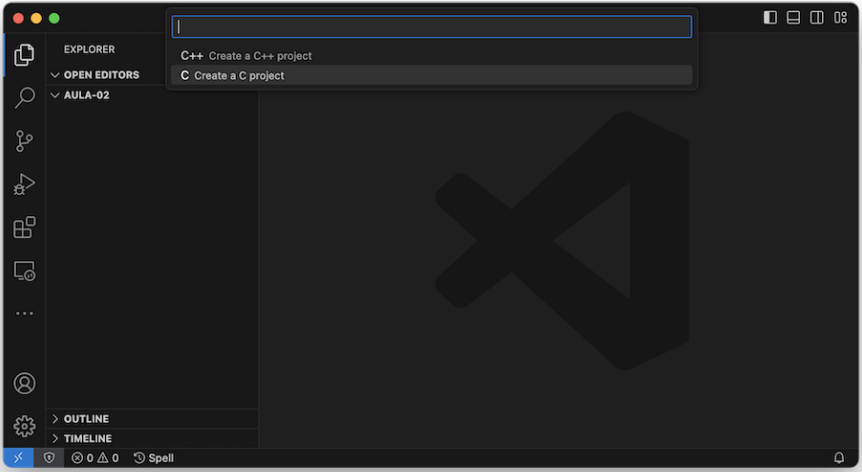
```
mkdir aula-02
cd aula-02
code .
```

- Abra o *prompt* de comando do VSCode (i.e. Ctrl + Shift ⇧ + P)
  - Digite: cmake quick start
  - Entre com o nome do projeto
  - Escolha criar um projeto na linguagem C
  - Escolha criar um executável

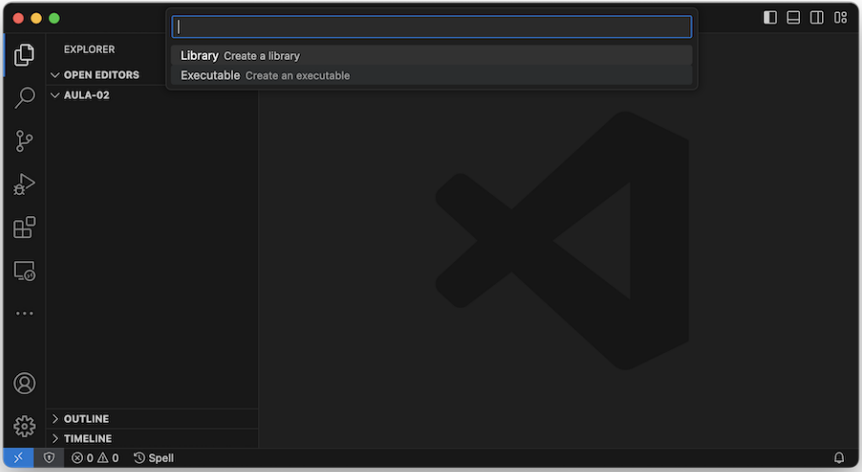
Criando um projeto com o VSCode



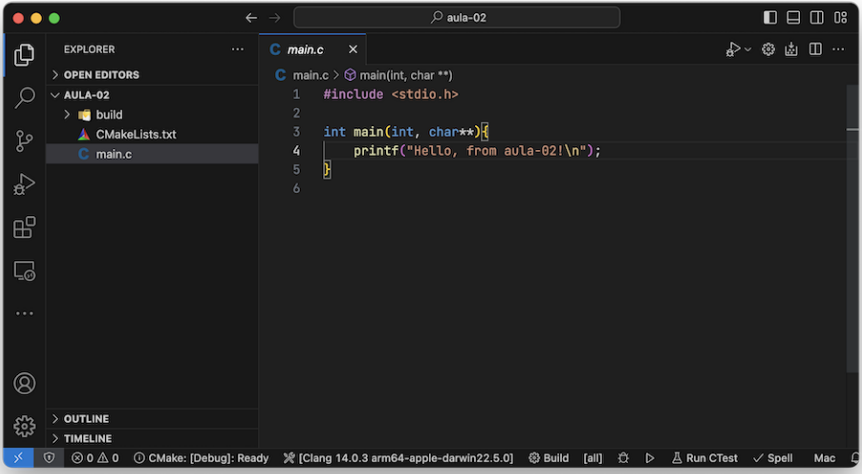
Criando um projeto com o VSCode



Criando um projeto com o VSCode



Criando um projeto com o VSCode



Criando um projeto com o VSCode

- O código gerado pelo VSCode não é compatível com o padrão ANSI C17
- Os parâmetros das funções precisam ser nomeados

```c
#include <stdio.h>

int main(int argc, char *argv[]){ // <-- ajuste aqui
    printf("Hello, from aula-02!\n");
    return 0;
}
```

Adicionando o arquivo `calculadora.c` e seu respectivo `.h`

1. Crie e edite os arquivos `calculadora.h` e `calculadora.c`

```c
// calculadora.h
#ifndef AULA_02_CALCULADORA_H
#define AULA_02_CALCULADORA_H
int soma(int, int);
#endif
```

```c
// calculadora.c
#include "calculadora.h"

int soma(int a, int b){
    return a + b;
}
```

Adicionando o arquivo `calculadora.c` e seu respectivo `.h`

2. Edite o arquivo CMakeLists.txt e inclua os nomes dos arquivos dentro da função `add_executable`

```
cmake_minimum_required(VERSION 3.0.0)
project(aula-02 VERSION 0.1.0 LANGUAGES C)

include(CTest)
enable_testing()

# Alterar linha abaixo
add_executable(aula-02 main.c calculadora.c calculadora.h)

# Para adicionar a biblioteca math.h
target_link_libraries(aula_02 m)

set(CPACK_PROJECT_NAME ${PROJECT_NAME})
set(CPACK_PROJECT_VERSION ${PROJECT_VERSION})
include(CPack)
```

- Compilando com gcc um programa que dependa da biblioteca `math.h`: `gcc main.c calculadora.c -o saida -lm`

Criando um projeto pela linha de comando

1. Crie o diretório para o projeto e dentro dele crie o arquivo `CMakeLists.txt`

```
cmake_minimum_required(VERSION 3.14)
project(exemplo C)
set(CMAKE_C_STANDARD 17)
add_executable(exemplo main.c)
# Para adicionar a biblioteca math.h
target_link_libraries(exemplo m)
```

2. Crie o arquivo `main.c` (ou outro nome que desejar)

```
#include <stdio.h>

int main(int argc, char *argv[]) {
    printf("Hello, World!\n");
    return 0;
}
```

Usando o cmake pela linha de comando

3. Gere o sistema de construção nativo (i.e. Makefile)

```
cmake -S . -B build
```

4. Invoque o sistema de construção para compilar e gerar o binário da aplicação

```
cmake --build build
```

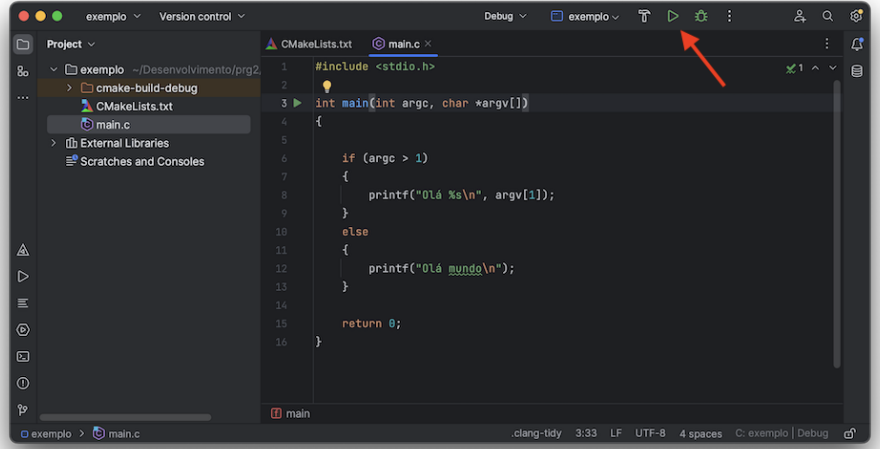
5. Execute a aplicação

```
./build/exemplo
```

Ferramentas para a disciplina

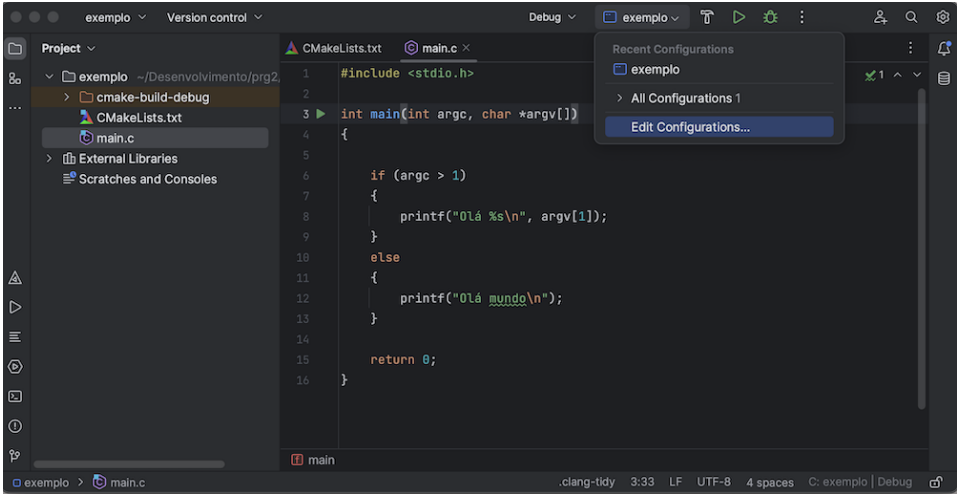
Executando e depurando projetos com CLion e VSCode

Executando com o CLion



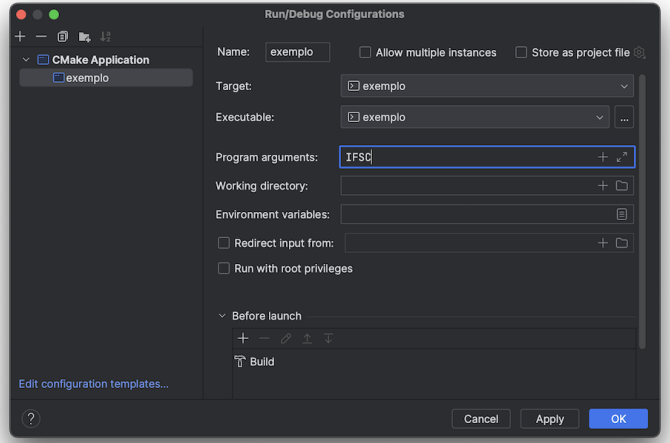
Depurando com o CLion

Fornecendo argumentos de linha de comando



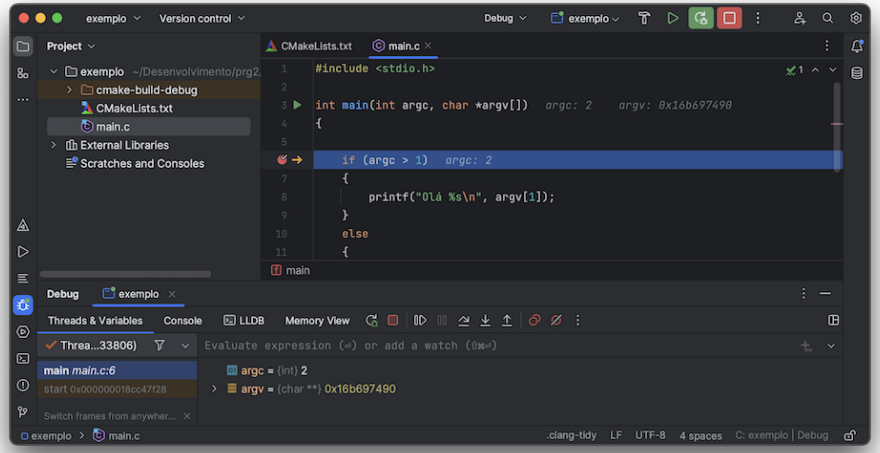
Depurando com o CLion

Fornecendo argumentos de linha de comando



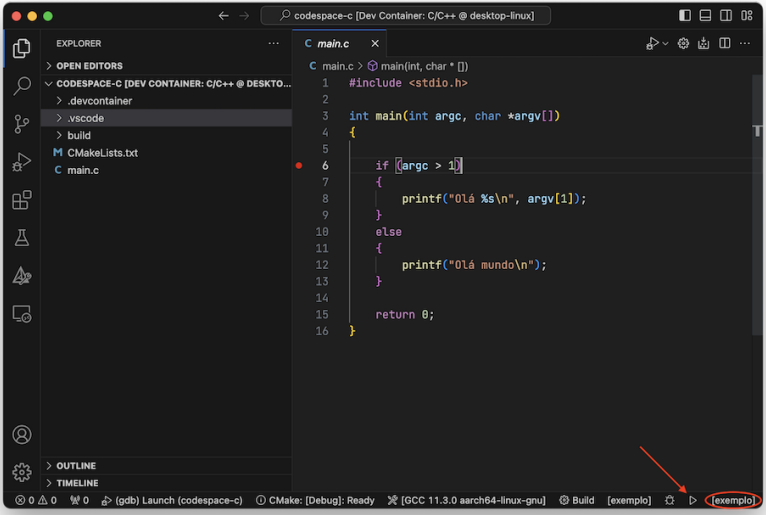
Depurando com o CLion

Fornecendo argumentos de linha de comando



Executando com o VSCode

Teclao de atalho Shift ⇧ + F5



Depurando com o CLion

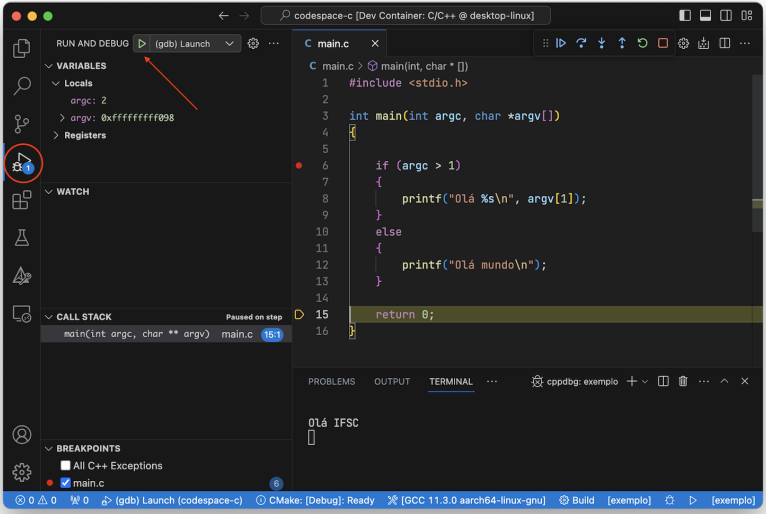
Fornecendo argumentos de linha de comando

- Se não quiser usar argumentos de linha de comando, então é possível executar a depuração de maneira rápida usando a tecla de atalho Ctrl + F5 .
- Se quiser usar argumentos de linha de comando, então será necessário criar uma configuração no arquivo `.vscode/launch.json` (veja próximo *slide*)

```
{"version": "0.2.0",
 "configurations": [{
            "name": "(gdb) Launch",
            "type": "cppdbg",
            "request": "launch",
            "program": "${command:cmake.launchTargetPath}",
            // Adicione ou remova os argumentos de linha de comando
            "args": ["IFSC"],
            "stopAtEntry": false,
            "cwd": "${workspaceFolder}",
            "environment": [{
                    "name": "PATH",
                    "value": "${env:PATH}:${command:cmake.getLaunchTargetDirectory}"
                }
            ],
            "MIMode": "gdb",
            "setupCommands": [{
                    "description": "Enable pretty-printing for gdb",
                    "text": "-enable-pretty-printing",
                    "ignoreFailures": true
                }
            ]
        }]
}
```

Depurando com o CLion

abrir o painel *Run and Debug* e depois clicar no botão *play* na parte superior do painel



Documentação e tutoriais CMake

- https://www.jetbrains.com/help/clion/quick-cmake-tutorial.html
- https://github.com/microsoft/vscode-cmake-tools/tree/main/docs#cmake-tools-for-visual-studio-code-documentation
- https://code.visualstudio.com/docs/cpp/cmake-linux
- https://cmake.org/cmake/help/latest/guide/tutorial/index.html
- https://coderefinery.github.io/cmake-workshop
- https://cliutils.gitlab.io/modern-cmake