

LISTA DE EXERCÍCIOS PI-P001

Exercício 1: Criando um Projeto no MS-Code.

Criando a pasta no Git.

```
jucaa@LAPTOP-N9NQ56ED MINGW64 ~/Desktop/repo_helder/MeuProjeto (master)
$ cd ..

jucaa@LAPTOP-N9NQ56ED MINGW64 ~/Desktop/repo_helder (master)
$ mkdir Projeto1

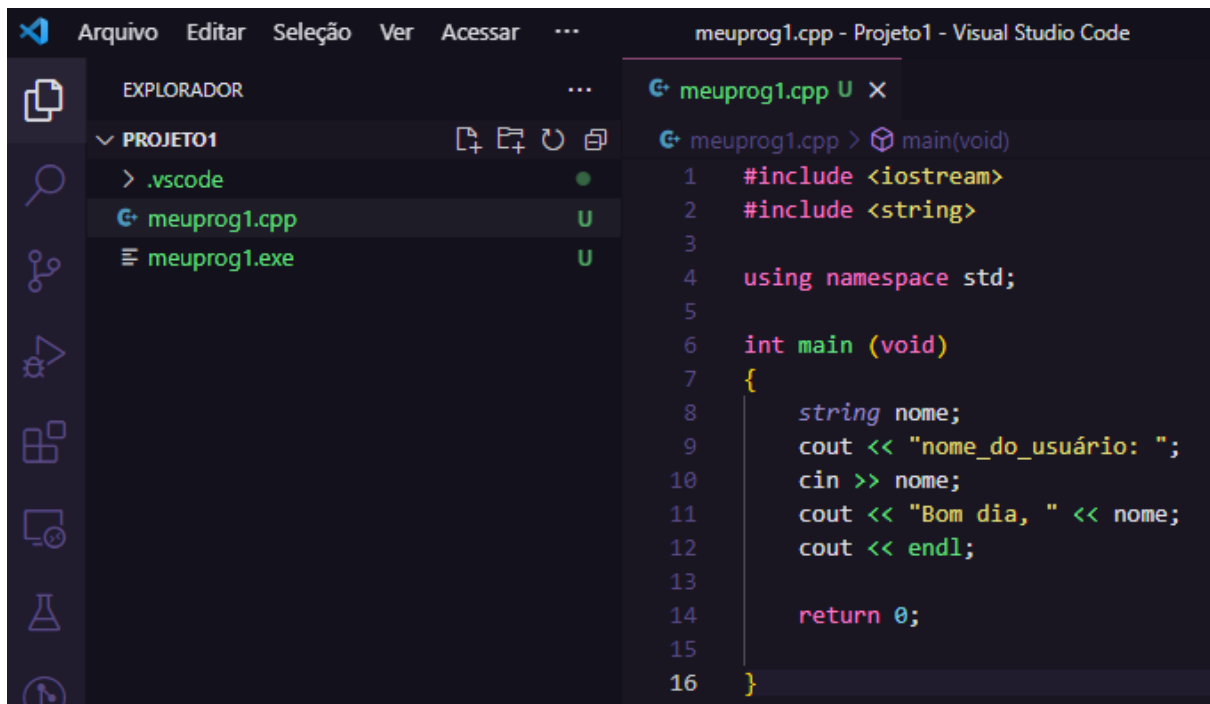
jucaa@LAPTOP-N9NQ56ED MINGW64 ~/Desktop/repo_helder (master)
$ cd Projeto1

jucaa@LAPTOP-N9NQ56ED MINGW64 ~/Desktop/repo_helder/Projeto1 (master)
$ code .

jucaa@LAPTOP-N9NQ56ED MINGW64 ~/Desktop/repo_helder/Projeto1 (master)
$ |
```

Exercício 2: Criando um programa básico.

Como boa prática, foi incluído o “return 0;” para verificar a codificação,



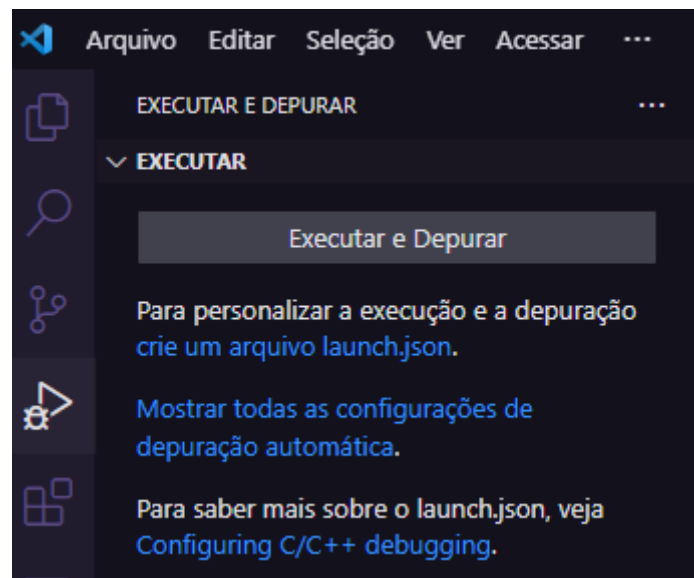
```
meuprog1.cpp - Projeto1 - Visual Studio Code

EXPLORADOR
▼ PROJETO1
  > .vscode
  meuprog1.cpp
  meuprog1.exe

meuprog1.cpp
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  int main (void)
7  {
8      string nome;
9      cout << "nome_do_usuario: ";
10     cin >> nome;
11     cout << "Bom dia, " << nome;
12     cout << endl;
13
14     return 0;
15
16 }
```

Exercícios 3: Compilando o programa.

Iniciando o compilador,



Compilando,

```
PS C:\Users\jucaa\Desktop\repo_helder\Projeto1> & 'c:\Users\jucaa\.vscode\extensions\ms-vscode.cpptools-1.17.5-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-iou2kztl.umi' '--stdout=Microsoft-MIEngine-Out-xdaad3yj.igr' '--stderr=Microsoft-MIEngine-Error-fcifaxzsp.wjv' '--pid=Microsoft-MIEngine-Pid-inc0gtzn.gwb' '--dbgExe=C:\msys64\mingw64\bin\gdb.exe' '--interpreter=mi'
nome_do_usuario: joseUlian
Bom dia, joseUlian
PS C:\Users\jucaa\Desktop\repo_helder\Projeto1> |
```

Retorno,

```
meuprog1.cpp U • meuprog2.cpp U
meuprog1.cpp > main()
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  int main ()
7  {
8      string nome;
9      cout << "nome_do_usuario: ";
10     cin >> nome;
11     cout << "Bom dia, " << nome;
12     cout << endl;
13
14     return 0;
15
16 }
```

Exercício 4: Criando outro programa.

O código correspondente,

```
meuprog1.cpp U  meuprog2.cpp U X
meuprog2.cpp > main()
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  int main ()
7  {
8      int numero1, numero2;
9
10     cout << "primeiro_numero: ";
11     cin >> numero1;
12
13     cout << "segundo_numero: ";
14     cin >> numero2;
15     cout << "numero1: " << numero1 << " numero2: " << numero2 << endl;
16
17     cout << "Soma           =: " << (numero1 + numero2) << endl;
18     cout << "Subtração        =: " << (numero1 - numero2) << endl;
19     cout << "Multiplicação     =: " << (numero1 * numero2) << endl;
20     cout << "Divisão          =: " << (numero1 / numero2) << endl;
21     cout << "Resto            =: " << (numero1 % numero2) << endl;
22
23     return 0;
24 }
```

Saída do terminal,

```
PROBLEMAS  SAÍDA  CONSOLE DE DEPURACÃO  TERMINAL  GITLENS
PS C:\Users\jucaa\Desktop\repo_helder\Projeto1> & 'c:\Users\jucaa\.vscode\extensions\ms-vscode.cpptools-1.17.5-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-z03x2fjt.fsb' '--stdout=Microsoft-MIEngine-Out-wqze1cot.q5q' '--stderr=Microsoft-MIEngine-Error-25vkpomi.0bj' '--pid=Microsoft-MIEngine-Pid-dqp2leuk.mgb' '--dbgExe=C:\msys64\mingw64\bin\gdb.exe' '--interpreter=mi'
primeiro_numero: 5
segundo_numero: 3
numero1: 5 numero2: 3
Soma           =: 8
Subtração      =: 2
Multiplicação  =: 15
Divisão        =: 1
Resto          =: 2
PS C:\Users\jucaa\Desktop\repo_helder\Projeto1> |
```

Exercício 5: Transforme as variáveis em float.

Modificando as variáveis para float, chegamos ao seguinte código. Em especial, a inclusão da biblioteca `cmath` e o comando `fmod` que permite um retorno para uma divisão em float, bem como o resto da mesma,

```
meuprog2.cpp M    meuprog2_float.cpp U X
meuprog2_float.cpp > main()
1  #include <iostream>
2  #include <string>
3  #include <cmath>
4
5  using namespace std;
6
7  int main ()
8  {
9
10     float numero_1, numero_2, resto;
11
12
13     cout << "primeiro_numero: ";
14     cin >> numero_1;
15
16     cout << "segundo_numero: ";
17     cin >> numero_2;
18
19     cout << "numero1: " << numero_1 << " numero2: " << numero_2 << endl;
20
21     cout << "Soma          =: " << (numero_1 + numero_2) << endl;
22     cout << "Subtração       =: " << (numero_1 - numero_2) << endl;
23     cout << "Multiplicação    =: " << (numero_1 * numero_2) << endl;
24     cout << "Divisão          =: " << (numero_1 / numero_2) << endl;
25     cout << "Resto            =: " << (resto = fmod(numero_1, numero_2)) << endl;
26
27     return 0;
28 }
```

Pode-se observar alterações nos resultados, os quais deixaram de ser aproximados,

```
primeiro_numero: 5.0
segundo_numero: 3.0
numero1: 5 numero2: 3
Soma          =: 8
Subtração     =: 2
Multiplicação =: 15
Divisão       =: 1.66667
Resto         =: 2
PS C:\Users\jucaa\Desktop\repo_helder\Projeto1>
```

Exercício 6: Transforme o programa do exercício 4, mas as entradas e saídas devem ser conforme o exemplo abaixo:

Com a inclusão da biblioteca `iosmanip`, foi possível delimitar o número de casas decimais de todos os valores. O número de casas decimais foi definido por meio do `setprecision` e fixado com o `fixed`,

```
meuprog2.cpp M    meuprog2_float.cpp U    meuprog3_uma_casa_decimal.cpp U ●
meuprog3_uma_casa_decimal.cpp > ...
1  #include <iostream>
2  #include <iomanip>
3  #include <string>
4
5  using namespace std;
6
7  int main ()
8  {
9
10     float numero_1, numero_2;
11     cout << setprecision (1) << fixed;
12
13     cout << "primeiro_numero: ";
14     cin >> numero_1;
15
16     cout << "segundo_numero: ";
17     cin >> numero_2;
18
19     cout << "numero1: " << numero_1 << " numero2: " << numero_2 << endl;
20
21     cout << "Soma           =: " << (numero_1 + numero_2) << endl;
22     cout << "Subtração        =: " << (numero_1 - numero_2) << endl;
23     cout << "Multiplicação     =: " << (numero_1 * numero_2) << endl;
24     cout << "Divisão           =: " << (numero_1 / numero_2) << endl;
25
26     return 0;
27 }
```

As saídas,

```
C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.29.30133\bin\x86_x64\mingw64\bin\gdb.exe' '--interpreter=mi'
primeiro_numero: 5.0
segundo_numero: 3.0
numero1: 5.0 numero2: 3.0
Soma           =: 8.0
Subtração      =: 2.0
Multiplicação  =: 15.0
Divisão        =: 1.7
PS C:\Users\jucaa\Desktop\repo_helder\Projeto1>
```

Exercício 7: Verifique quais as extensões instaladas.

- C/C++: A versão é v1.17.5;
- C/C++ Compile Run: A versão é v1.0.50;
- C/C++ Extension Pack: A versão é v1.3.0;
- C/C++ Themes: A versão é v2.0.0;
- CMake: A versão é v0.0.17;
- CMake Tools: A versão é v1.15.31;
- GitHub Codespaces: A versão é v1.15.0;
- GitLens – Git supercharged: A versão é v14.2.1;
- Jupyter notebook: a versão é v2023.7.1002162226;
- Jupyter Cell Tags: A versão é v0.1.8;
- Jupyter Keymap: A versão é v1.1.2;
- Jupyter Notebook Renderers: A versão é v1.0.17;
- Jupyter Slide Show: A versão é v0.1.5;
- Live Server: A versão é v5.7.9;
- Pylance: A versão é v2023.8.50;
- Python: A versão é v2023.14.0;

Exercício 8: Instalando Extensões.

Na barra de gerenciamento de contas foram escolhidas duas extensões para ampliar a utilização do VS Code.

- A primeira foi WSL (v0.81.0) a qual permite a utilização do Linux como um subsistema no Windows dentro do próprio terminal do VS code (também pode ser usado externamente, caso desejar). Após a instalação, o terminal de comando apresenta a opção de utilização da interface WSL simultaneamente a outras interfaces importantes como, por exemplo, o Git e GitHub.
- A segunda extensão foi a Omni Theme (v1.0.12) que, apesar de ser apenas um tema visual, facilita a leitura de códigos devido a organização colorida e correlacionada entre comandos introduzidos no VS Code.

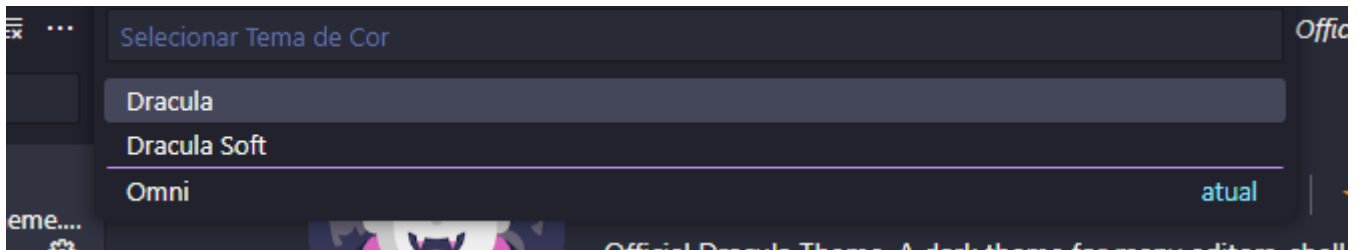


```
meuprog1.cpp > main()
You, há 1 segundo | 2 authors (Jose Ulian Cardoso and others)
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  int main ()
7  {
8      string nome1;
9      cout << "nome_do_usuario: ";
10     cin >> nome1;
11     cout << "Bom dia, " << nome1;
12     cout << endl;
13
14     return 0;
15 }
16
```

Exercício 9: Customizando a IDE – Temas.

Dentro da pesquisa de extensões, digitando dracula theme, foi possível encontrar o tema sugerido (Dracula Official) e iniciar a instalação.

A opção para definir o tema agora está disponível,



Após essas etapas, a interface foi modificada suavemente. O tema Omni Theme (v1.0.12) é bem semelhante com diferenças apenas em suas tonalidades. Fato percebido até o momento,

```
2_float.cpp U | G+ meuprog3_uma_casa_decimal.cpp U
G+ meuprog1.cpp > main()
You, há 20 segundos | 2 authors (Jose Ulian Cardoso and c
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  int main ()
7  {
8      string nome1;
9      cout << "nome_do_usuario: ";
10     cin >> nome1;
11     cout << "Bom dia, " << nome1;
12     cout << endl;
13
14     return 0;
15
16 }
```