

RECURSIVIDADE

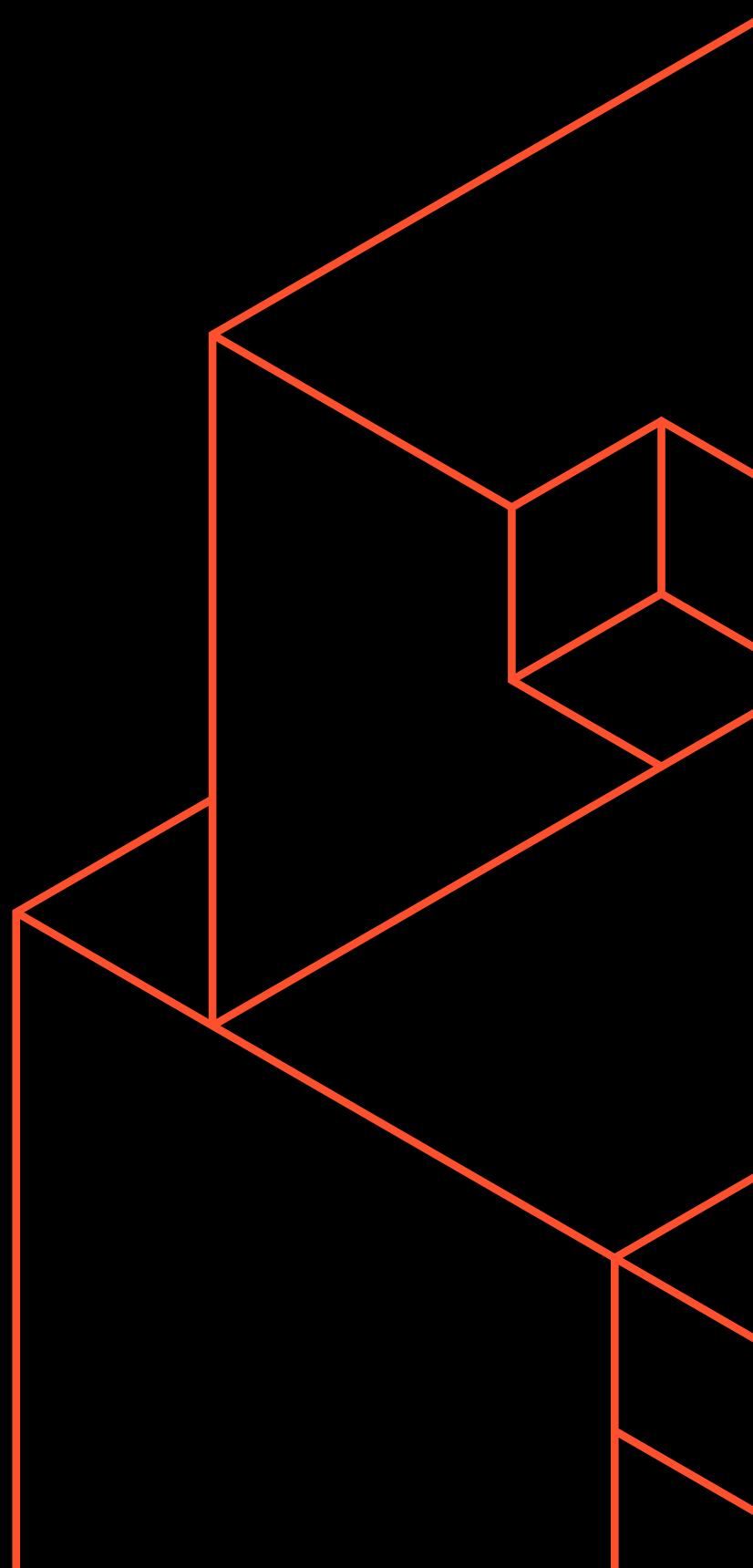
Professora Fernanda

Em ciência da computação, a recursividade é a definição de uma sub-rotina (função ou método) que pode invocar a si mesma. Um exemplo de aplicação da recursividade pode ser encontrado nos analisadores sintáticos recursivos para linguagens de programação. A grande vantagem da recursão está na possibilidade de usar um programa de computador finito para definir, analisar ou produzir um estoque potencialmente infinito de sentenças, designs ou outros dados.

- Um método comum de simplificação consiste em dividir um problema em subproblemas do mesmo tipo. Como técnica de programação, isto se denomina divisão e conquista, e constitui a chave para o desenvolvimento de muitos algoritmos importantes, bem como um elemento fundamental do paradigma de programação dinâmica.
- Praticamente todas as linguagens de programação usadas hoje em dia permitem a especificação direta de funções e procedimentos recursivos. Quando uma função é invocada, o computador (na maioria das linguagens sobre a maior parte das arquiteturas baseadas em pilhas) ou a implementação da linguagem registra as várias instâncias de uma função (em muitas arquiteturas, usa-se uma pilha de chamada, embora outros métodos possam ser usados). Reciprocamente, toda função recursiva pode ser transformada em uma função iterativa usando uma pilha.

PROGRAMAÇÃO RECURSIVA

- Em geral, uma definição recursiva é definida por casos: um número limitado de casos base e um caso recursivo. Os casos base são geralmente situações triviais e não envolvem recursão.
- Um exemplo comum usando recursão é a função para calcular o fatorial de um natural N. Nesse caso, no caso base o valor de $0!$ é 1. No caso recursivo, dado um $N > 0$, o valor de $N!$ é calculado multiplicando por N o valor de $(N-1)!$, e assim por diante, de tal forma que $N!$ tem como valor $N * (N-1) * (N-2) * \dots * (N-N)!$, onde $(N-N)!$ representa obviamente o caso base. Em termos recursivos:
-



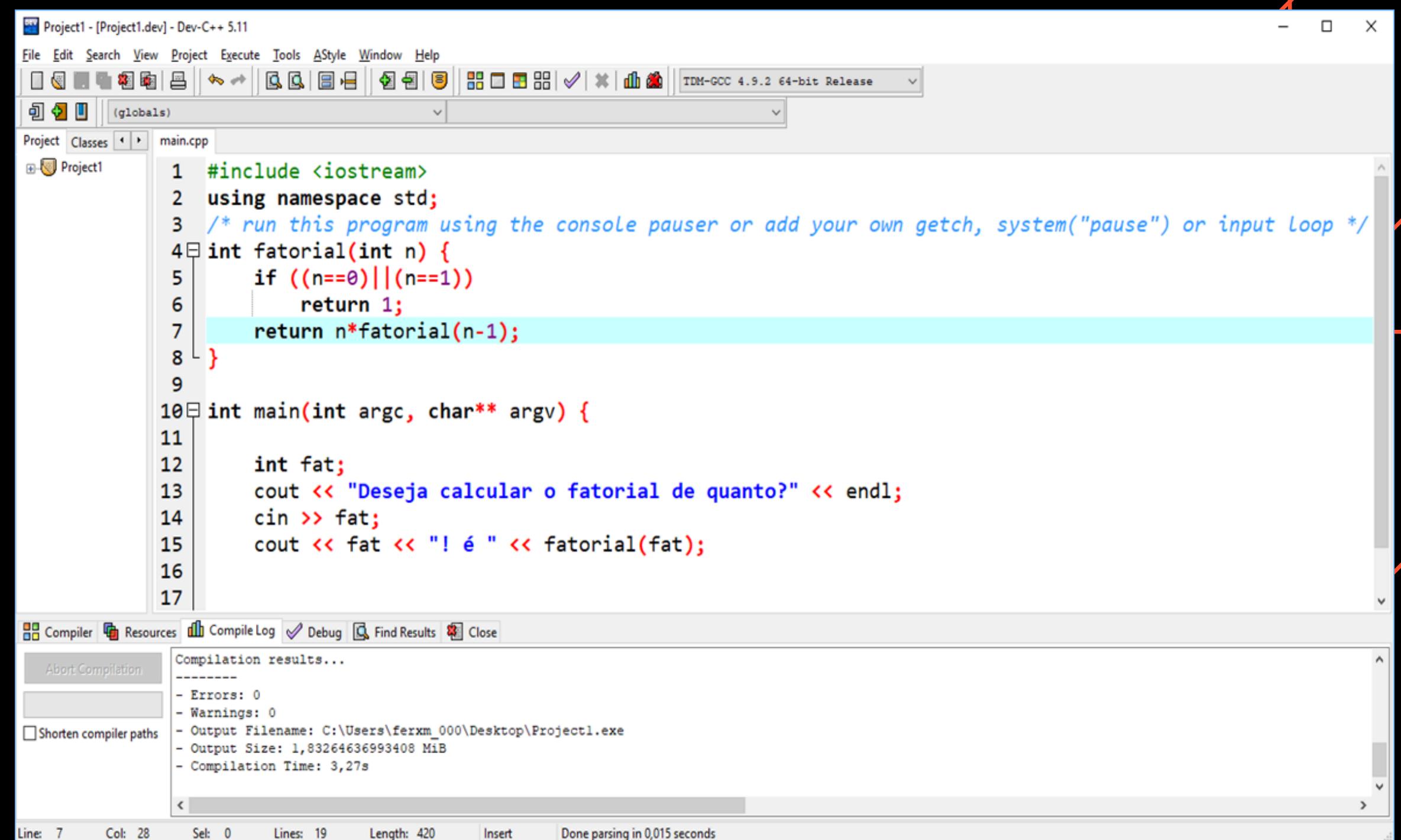
- Exemplo do processo do fatorial: $5!$

$$5 \times (5-1) = 20$$

$$20 \times (4 - 1) = 60$$

$$60 \times (3 - 1) = 120$$

$$120 \times (2-1) = 120$$



The screenshot shows the Dev-C++ IDE interface with the following details:

- Title Bar:** Project1 - [Project1.dev] - Dev-C++ 5.11
- Menu Bar:** File, Edit, Search, View, Project, Execute, Tools, AStyle, Window, Help
- Toolbar:** Includes icons for New, Open, Save, Build, Run, and others.
- Toolbox:** Includes icons for Project, Classes, and others.
- Compiler Status:** TDM-GCC 4.9.2 64-bit Release
- Project Explorer:** Shows "Project1" with "main.cpp" selected.
- Code Editor:** Displays the following C++ code:

```
1 #include <iostream>
2 using namespace std;
3 /* run this program using the console pauser or add your own getch, system("pause") or input loop */
4 int fatorial(int n) {
5     if ((n==0)||(n==1))
6         return 1;
7     return n*fatorial(n-1);
8 }
9
10 int main(int argc, char** argv) {
11
12     int fat;
13     cout << "Deseja calcular o fatorial de quanto?" << endl;
14     cin >> fat;
15     cout << fat << "!" << " é " << fatorial(fat);
16
17 }
```
- Compiler Log:** Shows compilation results:

```
Compilation results...
-----
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\ferxm_000\Desktop\Project1.exe
- Output Size: 1,83264636993408 MiB
- Compilation Time: 3,27s
```
- Status Bar:** Line: 7 Col: 28 Sel: 0 Lines: 19 Length: 420 Insert Done parsing in 0,015 seconds



VAMOS EXERCITAR

- Faça agora um método de Fibonacci usando a recursividade;