

RECURSIVIDADE

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 $N * (N-1) * (N-2) * ... * (N-N)!$, onde $(N-N)!$ representa obviamente o caso base. Em termos recursivos:

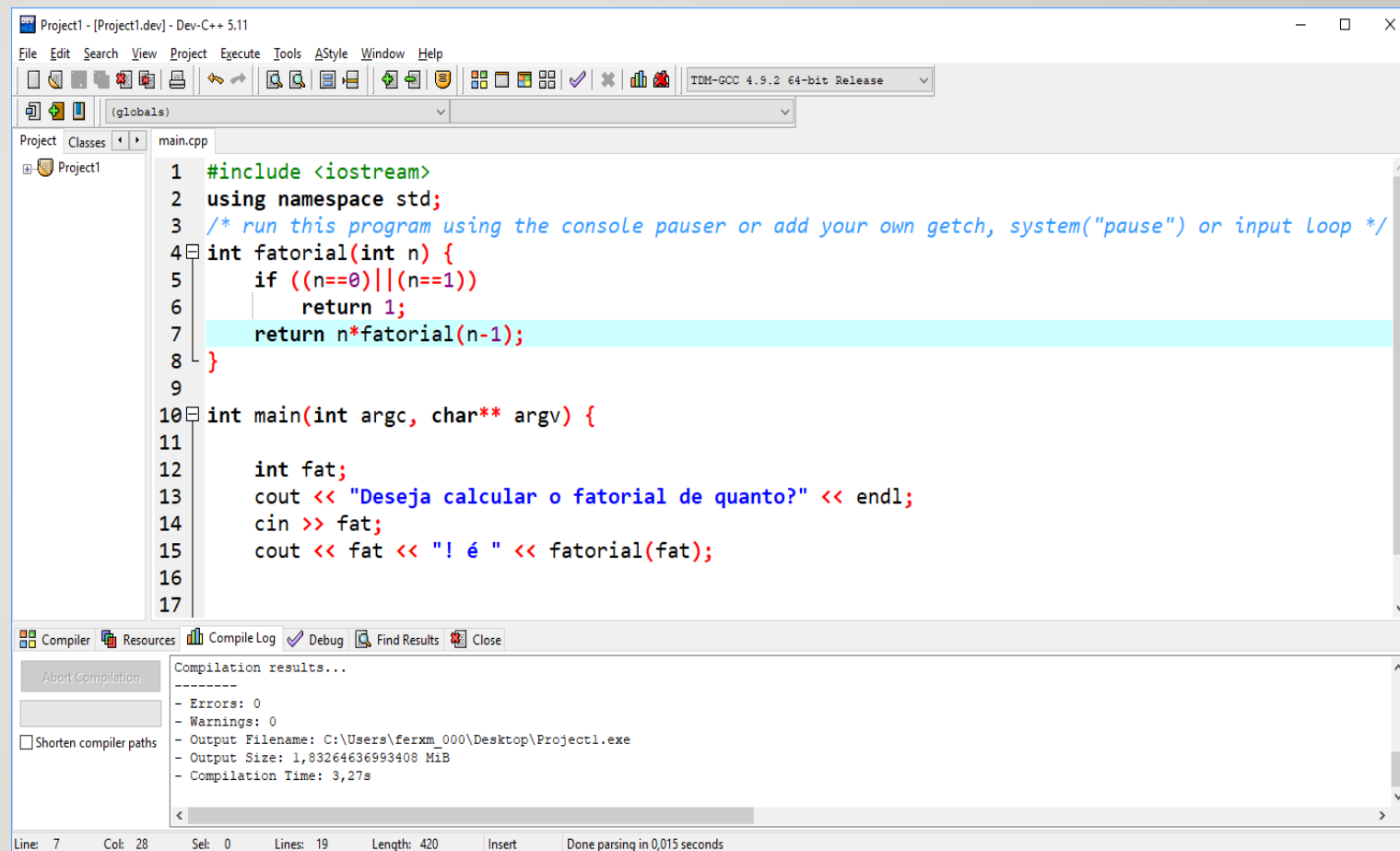
Exemplo do processo do fatorial: 5!

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

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

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

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



```
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 }
```

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\ferxm_000\Desktop\Project1.exe
- Output Size: 1,83264636993408 MiB
- Compilation Time: 3,27s

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;