

# Minicurso de C++ para Matemática

Pedro H A Konzen

10 de outubro de 2023

## Conteúdo

<b>1</b>	<b>Licença</b>	<b>1</b>
<b>2</b>	<b>Sobre a Linguagem</b>	<b>2</b>
2.1	Instalação e Execução . . . . .	2
2.1.1	IDE . . . . .	2
2.2	Olá, mundo! . . . . .	2
<b>3</b>	<b>Elementos da Linguagem</b>	<b>3</b>
3.1	Tipos de Dados Básicos . . . . .	3
	<b>Referências Bibliográficas</b>	<b>5</b>

## 1 Licença

Este trabalho está licenciado sob a Licença Atribuição-CompartilhaIgual 4.0 Internacional Creative Commons. Para visualizar uma cópia desta licença, visite [http://creativecommons.org/licenses/by-sa/4.0/deed.pt\\_BR](http://creativecommons.org/licenses/by-sa/4.0/deed.pt_BR) ou mande uma carta para Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

## 2 Sobre a Linguagem

C e C++ são **linguagens** de programação **compiladas** de propósito geral. A primeira é **estruturada e procedural**, tendo sido criada em 1972 por Dennis Ritchie<sup>1</sup>. A segunda foi inicialmente desenvolvida por Bjarne Stroustrup<sup>2</sup> como uma extensão da primeira. Em sua mais recente especificação, a linguagem C++ se caracteriza por ser **multi-paradigma (imperativa, orientada a objetos e genérica)**.

### 2.1 Instalação e Execução

Códigos C++ precisam ser compilados antes de serem executados. De forma simplificada, o **compilador** é um programa que interpreta e converte o código em um programa executável em computador. Há vários compiladores gratuitos disponíveis na web. Ao longo deste minicurso, usaremos a coleção de compiladores [GNU GCC](#) instalados em sistema operacional [Linux](#).

#### 2.1.1 IDE

Usar um **ambiente integrado de desenvolvimento** (IDE, em inglês, *integrated development environment*) é a melhor forma de capturar o melhor da linguagem C++. Algumas alternativas são:

- [Eclipse](#)
- [GNU Emacs](#)
- [VS Code](#)

### 2.2 Olá, mundo!

Vamos **implementar** nosso primeiro programa C++. Em geral, são três passos: 1. escrever; 2. compilar; 3. executar.

#### 1. **Escrever o código.**

Em seu IDE preferido, digite o código:

<sup>1</sup>Dennis Ritchie, 1941-2011, cientista da computação estadunidense. Fonte: [Wikipédia](#).

<sup>2</sup>Bjarne Stroustrup, 1950, cientista da computação dinamarquês. Fonte: [Wikipédia](#).

Código 1: ola.cpp

```
1 #include <iostream>
2
3 int main() {
4
5     std::cout << "Olá, mundo!"
6               << std::endl;
7     return 0;
8 }
```

## 2. Compilar.

Para compilá-lo, digite no terminal de seu sistema operacional

```
1 $ g++ ola.cpp -o ola.x
```

## 3. Executar.

Terminada a compilação, o arquivo executável `ola.x` é criado. Para executá-lo, digite

```
1 $ ./ola.x
```

# 3 Elementos da Linguagem

## 3.1 Tipos de Dados Básicos

Na linguagem C++, **dados** são alocados em **variáveis** com tipos declarados<sup>3</sup>.

**Exemplo 3.1.** Consideramos o seguinte código.

Código 2: dados.cpp

```
1 /* dados.cpp
2    Exemplo de alocação de variáveis.
3 */
4 #include <iostream>
5 #include <string>
```

<sup>3</sup>Consulte [Wikipedia: C data type](#) para uma lista dos tipos de dados disponíveis na linguagem

```
6
7 int main() {
8     // var inteira
9     int i = 1;
10    // var pto flutuante
11    double x;
12    // var string
13    std::string s;
14
15    x = 2.5;
16    s = "i + x";
17    double y = i + x;
18    std::cout << s << " = " \
19              << y << std::endl;
20    return 0;
21 }
```

Na linha 9, é alocada uma **variável do tipo inteira** com **identificador** `i` e **valor** 1. Na linha 11, é alocada uma **variável do tipo ponto flutuante** (64 *bits*) com **identificador** `x`.

Na linha 13, é alocada uma **variável do tipo *string*** (cadeia de caracteres). Antes, na linha 5, precisamos incluir a classe padrão ***string***.

Na linha 17, alocamos uma nova variável `y`. Uma vez declarada como ponto flutuante, o resultado da computação `i + x` é reinterpretado (*casting*) para ponto flutuante.

**Observação 3.1.** (**Comentários e Continuação de Linha.**) Códigos C++ admitem **comentários** e **continuação de linha** como no seguinte exemplo acima. Comentários em linha podem ser feitos com `//` e de múltiplas linhas com `/* ... */`. Linhas de instruções muito compridas podem ser quebradas em múltiplas linhas com a instrução de continuação de linha `\`.

**Observação 3.2.** (**Notação científica.**) Podemos usar **notação científica** em C++. Por exemplo  $5.2 \times 10^{-2}$  é digitado da seguinte forma `5.2e-2`.

Código 3: `notacaoCientifica.cpp`

```
1 #include <iostream>
2
3 int main() {
```

```
4
5  double x = 5.2e-2;
6
7  std::cout << "Padrão: "
8             << x << std::endl;
9
10 std::cout << "Fixada: " << std::fixed
11           << x << std::endl;;
12
13 std::cout << "Científica: " << std::scientific
14           << x << std::endl;;
15 return 0;
16 }
```

**Exercício 3.1.1.** Antes de implementar, diga qual o valor de  $x$  após as seguintes instruções.

```
1 int x = 1
2 int y = x
3 y = 0
```

Justifique sua resposta e verifique-a.

**Exercício 3.1.2.** Implemente um código em que a(o) usuá(ri)a entra com valores para as variáveis  $x$  e  $y$ <sup>4</sup>. Então, os valores das variáveis são permutados entre si.

## Referências

---

<sup>4</sup>A entrada de valores via console pode ser feita com o objeto `std::cin` da classe `iostream`.