

# Métodos Numéricos com C++: Fundamentos

Semana Universitária UnB 2024

**Professor: Adriano Possebon Rosa (aprosa@unb.br)**

Laboratório de Energia e Ambiente (lea.unb.br)

Departamento de Engenharia Mecânica

Faculdade de Tecnologia  
Universidade de Brasília

- 1 Introdução
- 2 O que é Programar?
- 3 Linguagens de Programação. C++?
- 4 GitHub e IDEs
- 5 O que vamos ver neste curso?
- 6 Aula 01
- 7 Aula 02
- 8 Aula 03
- 9 Aula 04
- 10 Até mais, e obrigado pelos peixes!



# 24<sup>a</sup> Semana Universitária da UnB

---

O mundo em nós:  
construindo um  
presente sustentável

4 - 10 NOV | 2024

Realização:



O **professor**, quem é?

**Formação:**

- **Graduação** em Engenharia Mecânica (UnB, 2012)
- **Mestrado** em Ciências Mecânicas (UnB, 2014)
- **Doutorado** em Ciências Mecânicas (UnB, 2018)

Professor na **UnB** desde 2016, no Departamento de **Engenharia Mecânica**.

Pesquisador do **Laboratório de Energia e Ambiente (LEA)**.



**Laboratório de Energia e Ambiente**  
Energy and Environment Laboratory



# Projeto Lambari

[Saiba mais](#)





# Valorização de resíduos lignocelulósicos do DF

[Saiba mais](#)



## PESSOAS

[ver todas](#) ➤



Energia e Sustentabilidade, Energia...

Professora Adjunta, Doutora

Simone Monteiro



Ecologia Industrial, Energia da Bio...

Professora Associada, Doutora

Sandra M. Luz



Energia Eólica, Hidro, Sistemas Té...

Professor Titular, Doutor

Antonio Brasil Junior



Ecologia Industrial, Energia da Bio...

Professor Adjunto, PhD

Edgar Amaral Silveira



Energia e Sustentabilidade, Energia...

Professor Adjunto, Doutor

Rafael Castilho Faria Mendes



Energia da Biomassa, Ecologia Ind...

Professor Titular, Doutor

Armando de Azevedo Caldeira-Pires



Camada Limite Atmosférica, Siste...

Professor Associado, Doutor

Mario Benjamin Baptista de Siqueira



Energia Eólica, Hidro, Sistemas Té...

Professor Associado, Doutor

Taygoara F. Oliveira



Energia da Biomassa, Energia e S...

Professor Associado, Doutor

Carlos Alberto Gurgel Veras



Energia Eólica, Hidro

Professor Adjunto, Doutor

Adriano Possebon Rosa

Minha área de **pesquisa**: **Transferência de Calor** e **Mecânica dos Fluidos**.

Atualmente ministro as **disciplinas**:

- Transporte de Calor e Massa
- Transferência de Calor
- Métodos Numéricos em Termofluidos
- Introdução à CFD

Trabalho com **programação numérica** desde 2010 (+-).

**Programação Numérica:** uso do computador para resolver equações complicadas.

O **computador** é usado para resolver equações que governam o movimento do fluido.

Simulação de partículas em um fluido magnético.

# The influence of dipolar particle interactions on the magnetization and the rotational viscosity of ferrofluids

---

Cite as: Phys. Fluids 31, 052006 (2019); doi: 10.1063/1.5093267

Submitted: 19 February 2019 • Accepted: 5 May 2019 •

Published Online: 29 May 2019

---



A. P. Rosa<sup>a)</sup> and F. R. Cunha<sup>b)</sup> 

---

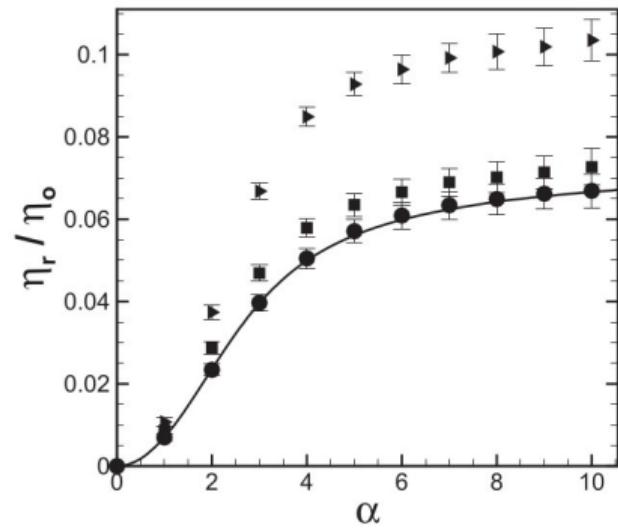
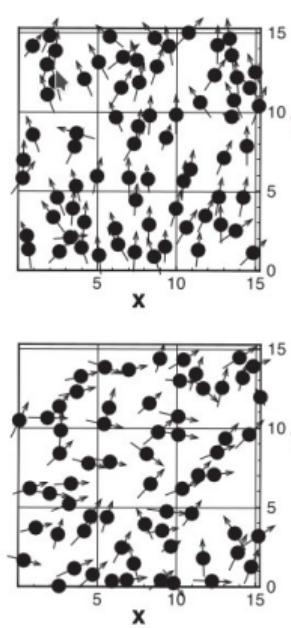
## AFFILIATIONS

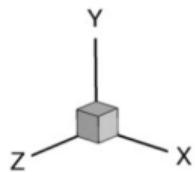
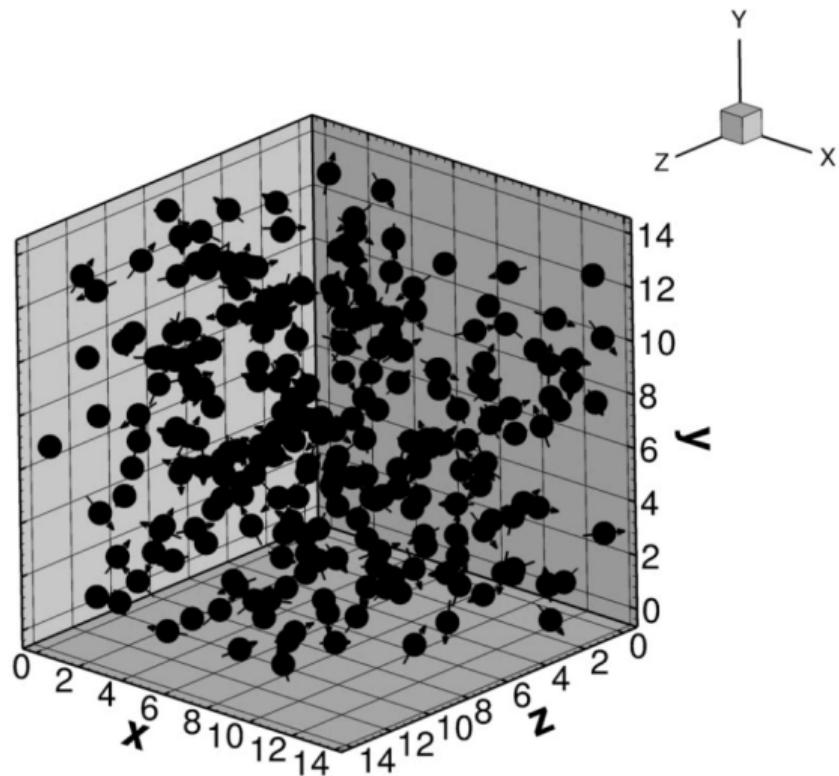
Fluid Mechanics of Complex Flows–VORTEX, Lab of Microhydrodynamics and Rheology, Department of Mechanical Engineering, University of Brasília, Campus Universitário Darcy Ribeiro, 70910-900 Brasília-DF, Brazil

<sup>a)</sup> Email: [possebon.adriano@gmail.com](mailto:possebon.adriano@gmail.com)

<sup>b)</sup> Author to whom correspondence should be addressed: [frcunha@unb.br](mailto:frcunha@unb.br)

# Simulação de partículas em um fluido magnético.





Simulação do comportamento de um fluido.

# Influence of a nonuniform magnetic field on the flow and heat transfer of a thermosensitive ferrofluid

---

Cite as: Phys. Fluids **36**, 103104 (2024); doi:[10.1063/5.0228839](https://doi.org/10.1063/5.0228839)

Submitted: 15 July 2024 · Accepted: 13 September 2024 ·

Published Online: 3 October 2024



View Online



Export Citation



CrossMark

---

L. H. F. Castro, T. F. Oliveira, and A. P. Rosa<sup>a)</sup>

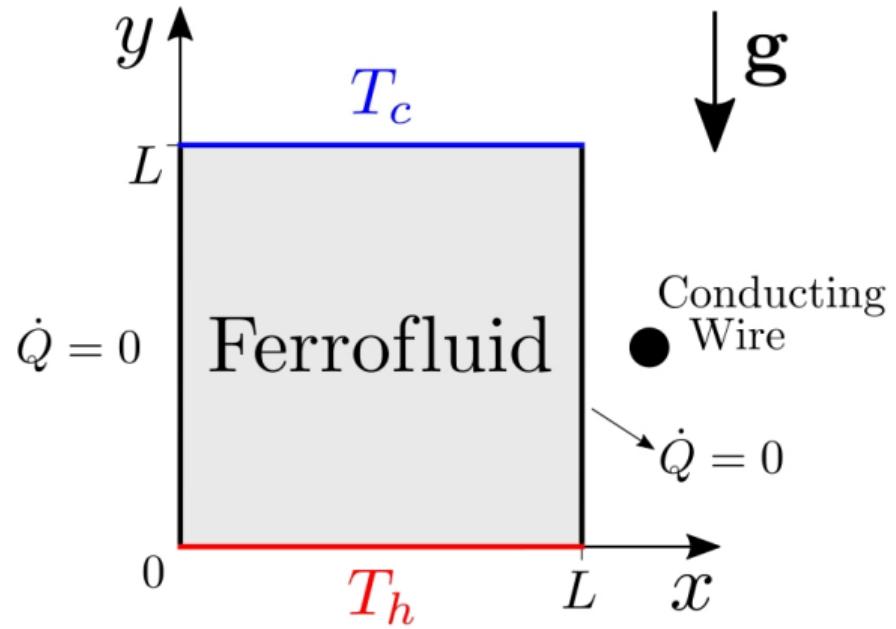
---

## AFFILIATIONS

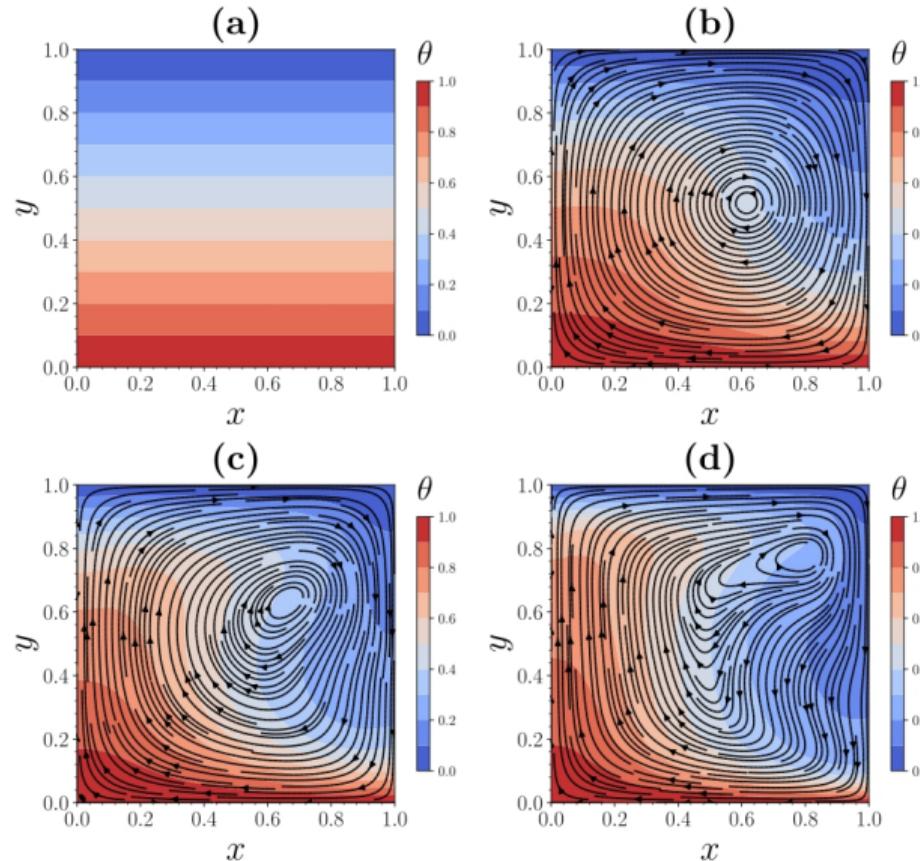
Energy and Environment Laboratory, University of Brasília, Brasília DF 70910-900, Brazil

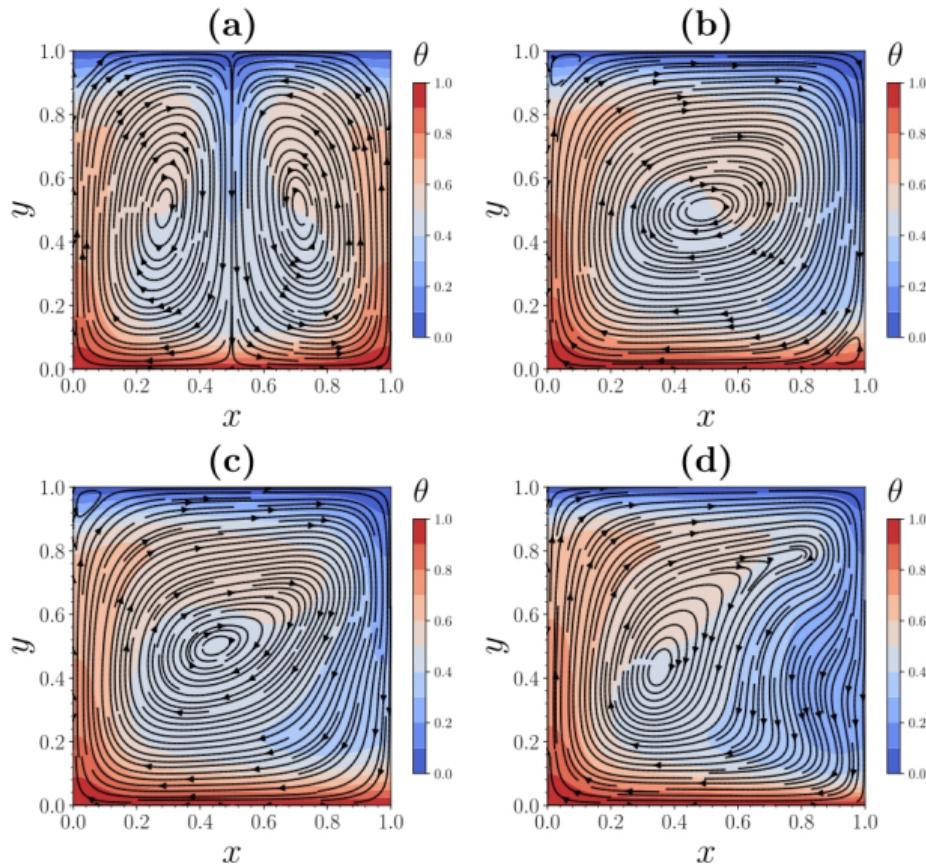
<sup>a)</sup>Author to whom correspondence should be addressed: [aprosa@unb.br](mailto:aprosa@unb.br)

---

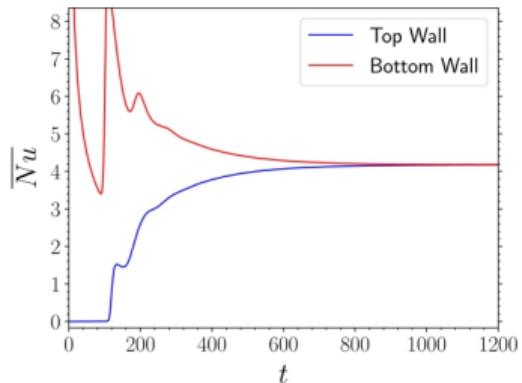


# Simulação do comportamento de um fluido.

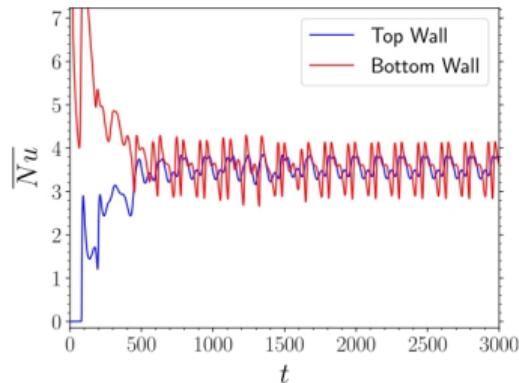




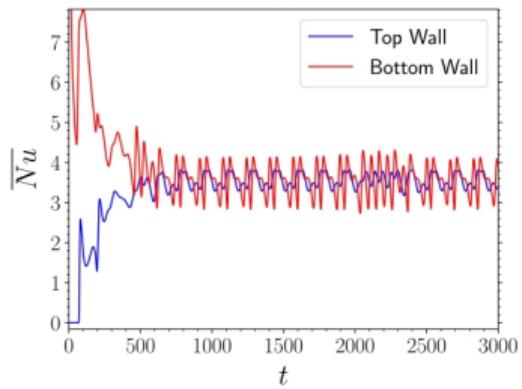
(a)



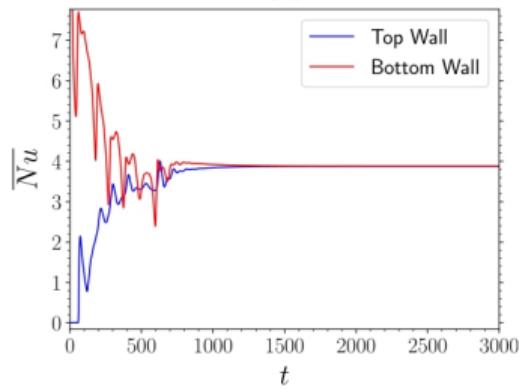
(b)



(c)



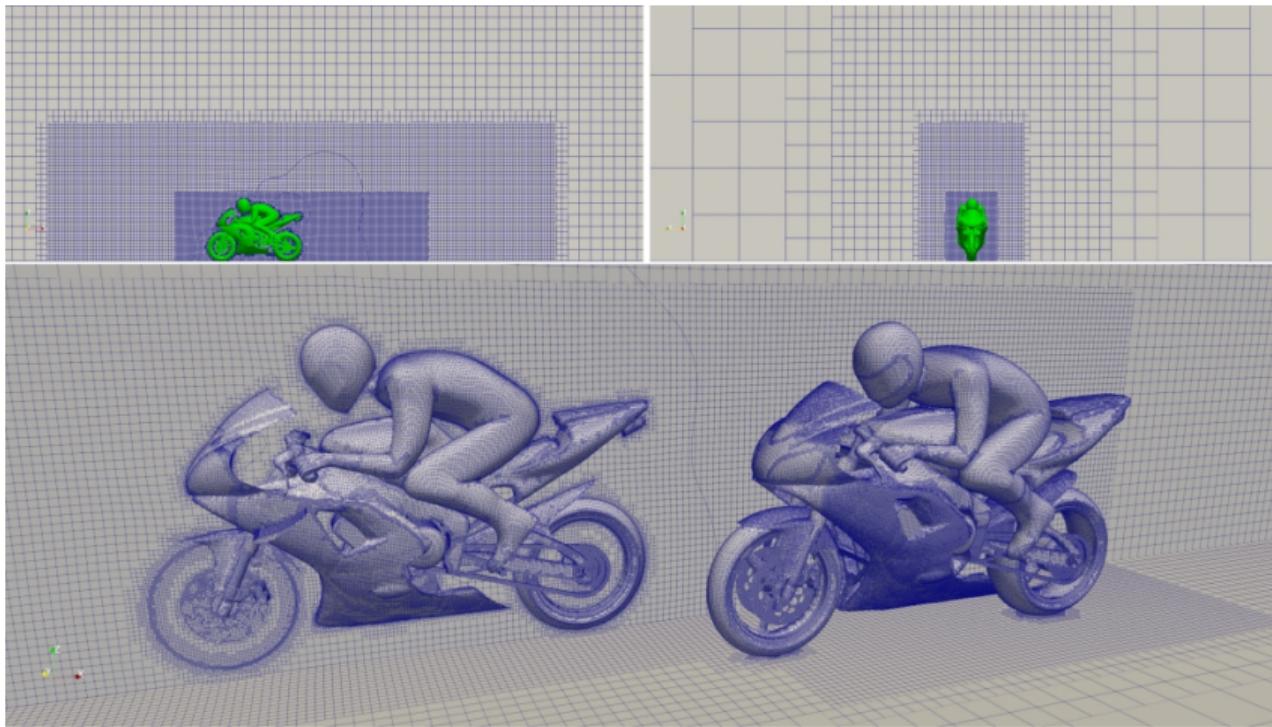
(d)

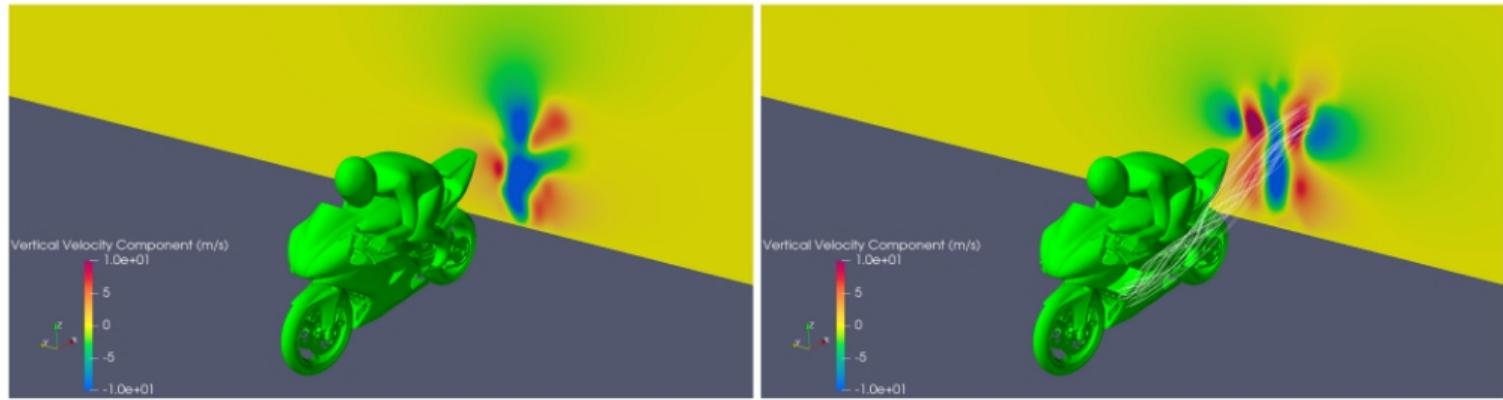


# Estudo do efeito de asas frontais na aerodinâmica de motocicletas esportivas.



**Figure 2**





**Figure 13**

Comecei a programar com o **MATLAB**.

Depois passei muitos anos com o bom e velho **Fortran**.  
(Ainda convivo muito com ele, e acho que vou conviver pra sempre.)

Tive uma rápida temporada com o **C++** em 2016.

Comecei a usar o **Python** em 2019. É a linguagem que eu mais uso no dia a dia.

Agora estou de volta com o **C++** (por causa do OpenFOAM).

Os **alunos**, quem são?

Quero saber:

- **curso/instituição**
- **experiência com programação** (zero, pouca, média, avançada)
- e o **motivo de você estar nesse curso**

## Pra quem é este curso?

Este curso é para quem

- tem interesse em programação numérica;
- nunca programou e quer ter uma primeira experiência com programação;
- já conhece um pouco de programação e quer saber um pouco mais;
- já programa usando outra linguagem e quer conhecer o famoso C++.

- 1 Introdução
- 2 O que é Programar?
- 3 Linguagens de Programação. C++?
- 4 GitHub e IDEs
- 5 O que vamos ver neste curso?
- 6 Aula 01
- 7 Aula 02
- 8 Aula 03
- 9 Aula 04
- 10 Até mais, e obrigado pelos peixes!

Textão do **Wikipédia**:

**Programação** é o processo de escrita, teste e manutenção de um programa de computador. O programa é escrito em uma linguagem de programação, embora seja possível, com alguma dificuldade, o escrever diretamente em linguagem de máquina. Diferentes partes de um programa podem ser escritas em **diferentes linguagens**.

Diferentes **linguagens** de programação funcionam de diferentes modos. Por esse motivo, os **programadores** podem criar programas muito diferentes para diferentes linguagens; muito embora, teoricamente, a maioria das linguagens possa ser usada para criar **qualquer programa**.

Há várias décadas se debate se a programação é mais semelhante a uma **arte**, a uma **ciência**, à **matemática**, à **engenharia**, ou se é um campo completamente novo.

**Programa:** sequência de instruções que dizem ao computador o que fazer.

**Linguagem:** traduz o que está escrito no programa em linguagem de máquina (zeros e uns).

```
37     double delta_x = 1.0/N;
38     double delta_y = delta_x;
39     double delta_t = 0.25*Re*delta_x*delta_x;
40
41 //Restriction For delta_t
42 if (delta_t > 0.5*delta_x)
43 { delta_t = 0.25*delta_x;};
44
45 //X-velocity Array
46 double u[N+2][N+2], u_s[N+2][N+2];
47
48 for (int i=0; i<N+2; i++)
49 {
50     for (int j=0; j<N+2; j++)
51     {
52         if (j==N+1)
53         {
54             u[i][j] = 2.0;
55             u_s[i][j] = 0.0;
56         }
57         else
```

```
207 def calculate_v_star(v_star, u, v, p, Re, N_x, N_y, dx, dy, dt):
208 """
209     Calcula v_star seguindo o algoritmo do método de
210     projeção de primeira ordem. O cálculo de v_star é
211     explícito.
212 """
213
214 # Pontos do interior do domínio.
215 # a1, a2, a3 e a4 são variáveis auxiliares.
216 for i in range(0, N_x):
217     for j in range(1, N_y):
218         a1 = 0.25*(u[i+1, j] + u[i, j] + u[i+1, j-1] + u[i, j-1])
219         a2 = (a1*(v[i+1, j] - v[i-1, j])/(2.0*dx) +
220               v[i, j]*(v[i, j+1] - v[i, j-1])/(2.0*dy))
221         a3 = (v[i+1, j] - 2.0*v[i, j] + v[i-1, j])/(Re*dx*dx)
222         a4 = (v[i, j+1] - 2.0*v[i, j] + v[i, j-1])/(Re*dy*dy)
223         v_star[i, j] = v[i, j] + dt*(-a2 + a3 + a4)
```

- 1 Introdução
- 2 O que é Programar?
- 3 Linguagens de Programação. C++?
- 4 GitHub e IDEs
- 5 O que vamos ver neste curso?
- 6 Aula 01
- 7 Aula 02
- 8 Aula 03
- 9 Aula 04
- 10 Até mais, e obrigado pelos peixes!

**Pergunta:** qual é a melhor linguagem de programação?

**Resposta:**



# BIRL



O BIRL (*Bambam's "It's show time" Recursive Language*) é a linguagem de programação mais treze já inventada.

Deve ser utilizada apenas por quem realmente constrói fibra e não é água com código. É uma linguagem extremamente simples porém com poder para derrubar todas as árvores do parque Ibirapuera.

Programando em BIRL, é verão o ano todo!

## INÍCIO DO CÓDIGO

HORA DO SHOW  
//código aqui  
BIRL

### TIPOS DE DADOS

FRANGO FR = 'a'; (*char*)  
MONSTRINHO M1 = 13; (*short*)  
MONSTRO M2 = 37; (*int*)  
MONSTRAO M3 = 666; (*long*)

### TIPOS DE DADOS

TRAPEZIO T = 0.13; (*float*)  
TRAPEZIO DESCENDENTE TD = 0.37; (*double*)  
BICEPS FRANGO BF = 200; (*unsigned*)

### PRINTAR NA TELA

CE QUER VER ESSA PORRA? ("Hello, Mutante");

### LER DA TELA

MONSTRO X;  
QUE QUE CE QUER MONSTRAO? ("%d", &X);

## IF

ELE QUE A GENTE QUER? ( $3 > 2$ )

//código a ser executado

BIRL

## IF/ELSE

ELE QUE A GENTE QUER? ( $X > 2$ )

//caso verdadeiro

NAO VAI DAR NAO

//caso falso

BIRL

## ELSE IF

ELE QUE A GENTE QUER? ( $X > 2$ )

// $X > 2$

QUE NAO VAI DAR O QUE? ( $X < 2$ )

// $X < 2$

NAO VAI DAR NAO

// $X = 2$

BIRL

## WHILE

MONSTRO X = 5;

NEGATIVA BAMBAM ( $X > 2$ )

//rodar código

X--;

BIRL

## FOR

```
MONSTRO M;  
MAIS QUERO MAIS (M = 0; M < 5; M++)  
    CE QUER VER ESSA PORRA? ("%d", M);  
BIRL
```

## BREAK/CONTINUE

```
MONSTRO M;  
MAIS QUERO MAIS (M = 0; M < 5; M++)  
    //continue  
    VAMO MONSTRO;  
    //break  
    SAI FILHO DA PUTA;  
BIRL
```

## DECLARAR FUNÇÃO

```
OH O HOME AI PO (MONSTRO NOMEFUNC(MONSTRO A,  
MONSTRO B))  
    //código da função  
    BORA CUMPADE 1;  
BIRL
```

## CHAMAR FUNÇÃO

```
MONSTRO A = 5;  
MONSTRO B = 8;  
MONSTRO C = AJUDA O MALUCO TA DOENTE SOMAR(A, B);
```

OH O HOME AI PO (MONSTRO SOMAR(MONSTRO A, MONSTRO B))  
BORA CUMPADE A + B;  
BIRL

### HORA DO SHOW

MONSTRO A, B, RES;

CE QUER VER ESSA PORRA? ("Entra com a e b ai cumpade!!\n");

QUE QUE CE QUER MONSTRAO? ("%d %d", &A, &B);

RES = AJUDA O MALUCO TA DOENTE SOMAR(A, B);

CE QUER VER ESSA PORRA? ("Oh o resultado ai po: %d\n", RES);

ELE QUE A GENTE QUER? (RES == 37)

CE QUER VER ESSA PORRA? ("É 37 anos caralho!\n");

### NAO VAI DAR NAO

CE QUER VER ESSA PORRA? ("Manda o double biceps!\n");

BIRL

BORA CUMPADE 0;

BIRL

Piadas à parte, não existe uma resposta para essa pergunta.

E, na verdade, essa não é a pergunta mais importante.

A pergunta que realmente importa é:

**O que você quer fazer? Qual é o seu projeto? Por que você quer usar um programa de computador?**

Linguagens mais famosas.

## Front-end:

- HTML, CSS, JavaScript

## Back-end:

- C, C++, C#, Ruby, Java, Perl, Python

## Desenvolvimento de Apps:

- Swift, Java, Objective-C, C++, C#, JavaScript, Python

## Programação Científica:

- C, C++, Fortran, Matlab, Julia, Octave, R, Python

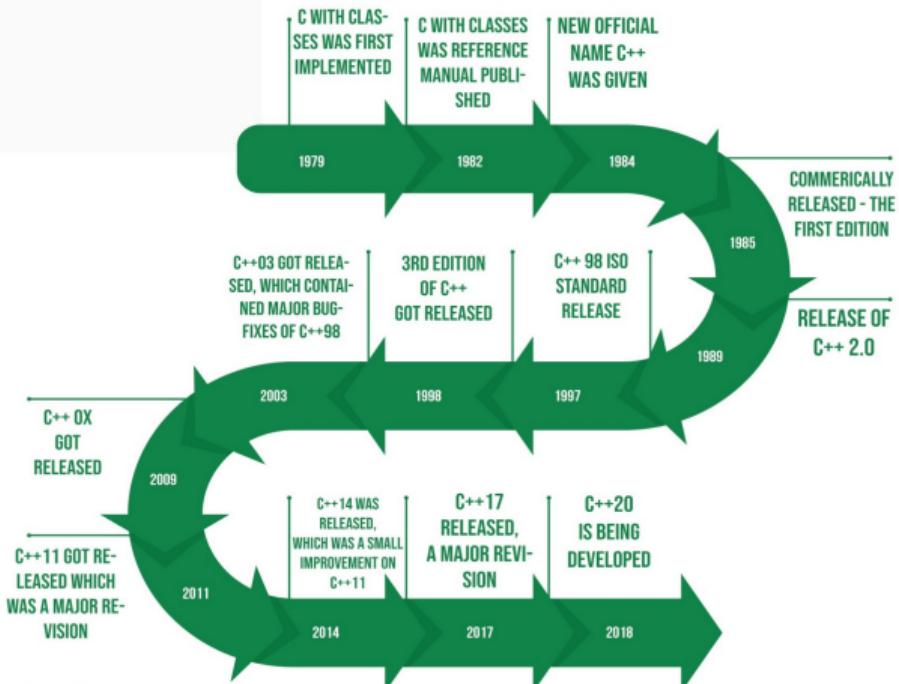
Temos mais de **1000** linguagens de programação!

# Por que C++?



Bjarne Stroustrup.

# History of C++



# Onde C++ é usada.

- **Sistemas Operacionais**
  - Windows
  - macOS e iOS
  - Linux
- **Navegadores Web**
  - Google Chrome
  - Mozilla Firefox
  - Safari
- **Softwares de Design e Animação 3D**
  - Autodesk Maya
  - Blender
  - Adobe Photoshop

- **Games e Motores de Jogo**
  - Unreal Engine
  - Unity (componentes específicos)
  - Jogos famosos: *Counter-Strike*, *World of Warcraft*, *Fortnite*
- **Softwares de Computação Científica e Engenharia**
  - MATLAB
  - Wolfram Mathematica
  - ANSYS
  - **OpenFOAM**

# Blender.

blender / source / blender / geometry / intern / **mesh\_boolean.cc** ↑ Top

**Code** **Blame** 1184 lines (1103 loc) · 46.4 KB Raw   

```
48     * so this is a hack to clean up such matrices.
49     * Would be better to change the transformation code itself.
50     */
51 v static float4x4 clean_transform(const float4x4 &mat)
52 {
53     float4x4 cleaned;
54     const float fuzz = 1e-6f;
55     for (int i = 0; i < 4; i++) {
56         for (int j = 0; j < 4; j++) {
57             float f = mat[i][j];
58             if (fabsf(f) <= fuzz) {
59                 f = 0.0f;
60             }
61             else if (fabsf(f - 1.0f) <= fuzz) {
62                 f = 1.0f;
63             }
64             else if (fabsf(f + 1.0f) <= fuzz) {
65                 f = -1.0f;
66             }
67             cleaned[i][j] = f;
68         }
69     }
70     return cleaned;
71 }
72
73 /* 'MeshesToIMeshInfo' keeps track of information used when combining a number
74    * of 'Mesh'es into a single 'IMesh' for doing boolean on.
75    * Mostly this means keeping track of the index offsets for various mesh elements. */
76 v class MeshesToIMeshInfo {
77     public:
78     /* The input meshes, */
79     Span<const Mesh *> meshes;
80     /* Numbering the vertices of the meshes in order of meshes,
```

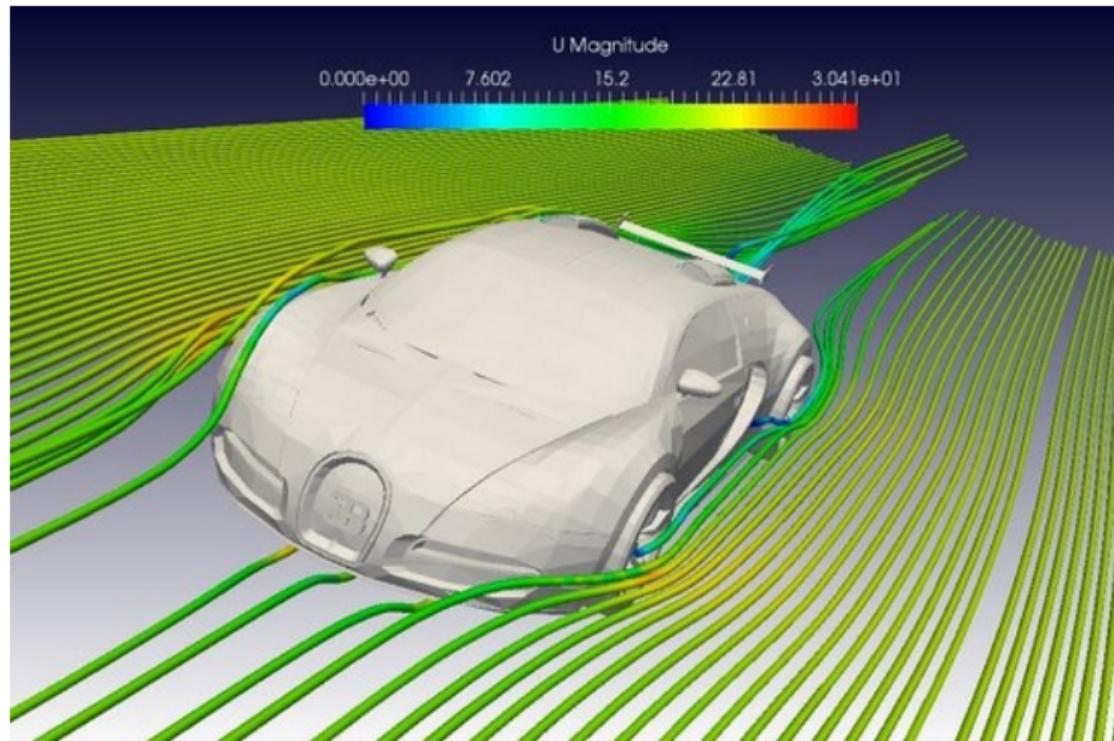
# OpenFOAM.

The screenshot shows the official OpenFOAM website. At the top left is the OpenFOAM logo, which consists of a grid of green squares. To its right is the text "OpenFOAM" in a large, bold, black font, with "The OpenFOAM Foundation" in a smaller, gray font below it. A navigation bar at the top right includes links for "Download", "Development", "Resources", "Funding" (which is highlighted in red), and "Contact Us", each with a dropdown arrow, followed by a search icon. The main content area features a large, bold "OPENFOAM 12" title in white and yellow. Below it is the text "FLUID SIMULATION SOLID FOUNDATION" in yellow, and a URL "https://openfoam.org/12" in white. At the bottom left is a circular icon with a recycling symbol and the text "REDESIGN REPAIR PUBLISH". Next to it is a teal button with the text "Read More". To the right is another circular icon with a recycling symbol and the text "100% FREE OPEN SOURCE GPL v3". Below these icons are four small white dots.

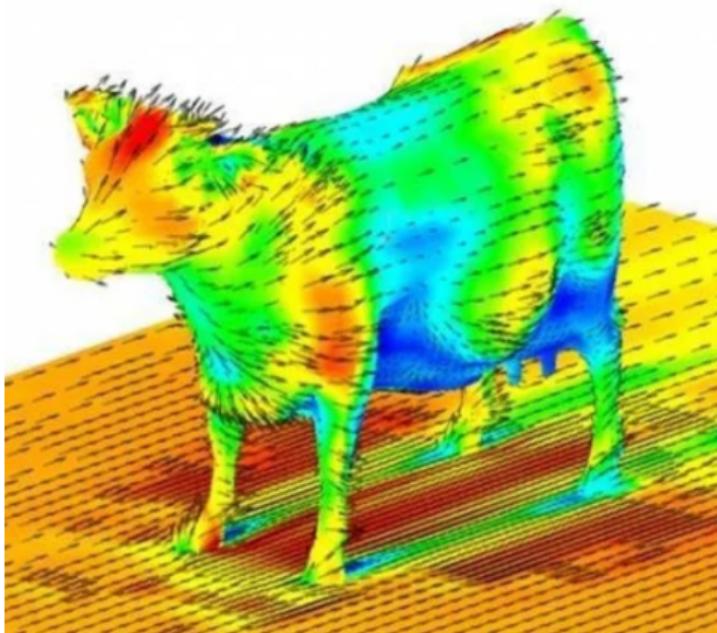
## OpenFOAM and The OpenFOAM Foundation

OpenFOAM is free, open source software for CFD from the OpenFOAM Foundation.

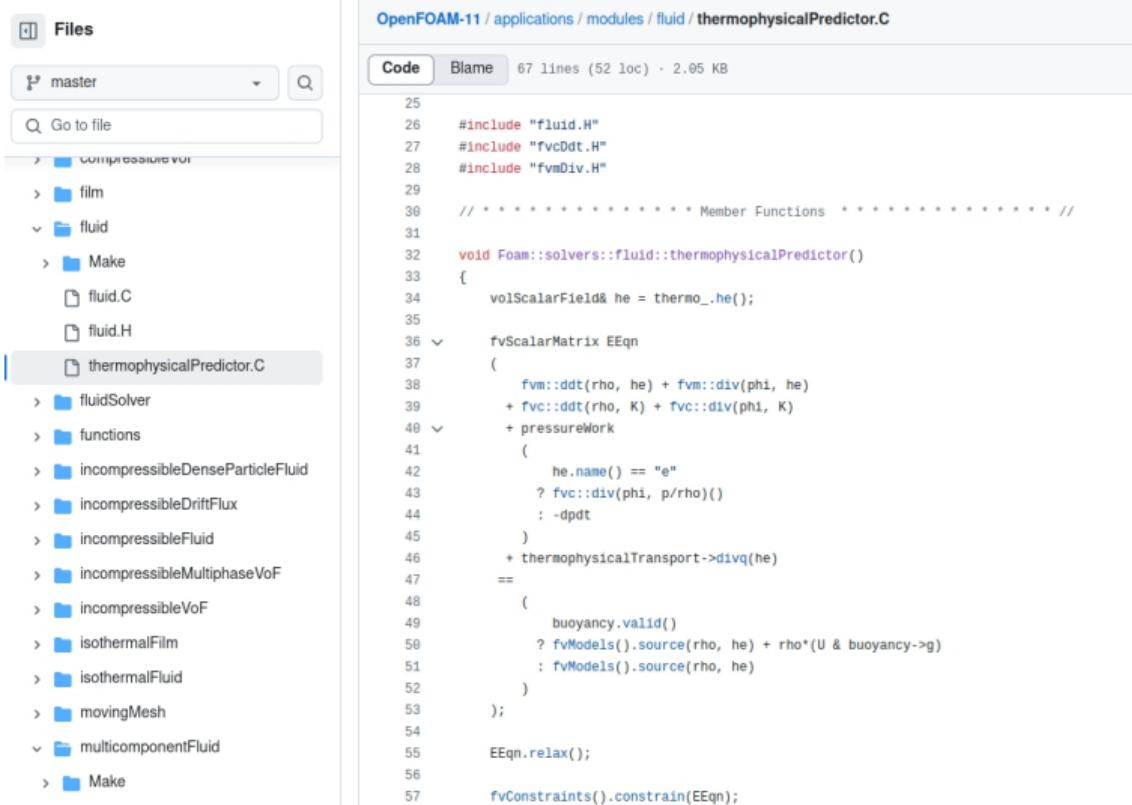
OpenFOAM.



OpenFOAM.



# OpenFOAM.



The screenshot shows a code editor interface with the following details:

- File Path:** OpenFOAM-11 / applications / modules / fluid / thermophysicalPredictor.C
- Code Tab:** The current tab is "Code".
- Blame Tab:** Shows 67 lines (52 loc) · 2.05 KB.
- Code Content:** The code is a C++ file containing member functions for the thermophysicalPredictor class. It includes #include statements for fluid.H, fvcDdt.H, and fvmDiv.H. The code uses fvScalarField he and fvScalarMatrix EEqn. It performs calculations involving rho, phi, K, and buoyancy, and calls fvModels().source() and fvConstraints().constrain().

```
25
26 #include "fluid.H"
27 #include "fvcDdt.H"
28 #include "fvmDiv.H"
29
30 // * * * * * Member Functions * * * * *
31
32 void Foam::solvers::fluid::thermophysicalPredictor()
33 {
34     volScalarField& he = thermo_.he();
35
36     fvScalarMatrix EEqn
37     (
38         fvm::ddt(rho, he) + fvm::div(phi, he)
39         + fvc::ddt(rho, K) + fvc::div(phi, K)
40         + pressureWork
41         (
42             he.name() == "e"
43             ? fvc::div(phi, p/rho]()
44             : -dpdt
45         )
46         + thermophysicalTransport->divq(he)
47     ==
48     (
49         buoyancy.valid()
50         ? fvModels().source(rho, he) + rho*(U & buoyancy->g)
51         : fvModels().source(rho, he)
52     )
53 );
54
55 EEqn.relax();
56
57 fvConstraints().constrain(EEqn);
```

## C++: Vantagens.

- **Alto Desempenho:** Controle direto sobre memória e recursos do sistema.
- **Programação Orientada a Objetos (OOP):** Suporte a herança, polimorfismo e encapsulamento.
- **Ampla Biblioteca Padrão (STL):** Estruturas de dados e algoritmos que facilitam o desenvolvimento.
- **Controle sobre Memória:** Gerenciamento manual que permite uso eficiente de recursos.
- **Linguagem Multi-Paradigma:** Suporte a programação estruturada, procedural, orientada a objetos e genérica.
- **Portabilidade:** Ampla compatibilidade com diversas plataformas.

## C++: Desvantagens.

- **Complexidade:** Linguagem complexa que pode dificultar o aprendizado para iniciantes.
- **Gerenciamento Manual de Memória:** Propenso a erros como vazamento de memória e uso indevido de ponteiros.
- **Compilação Lenta:** Tempos de compilação longos em projetos grandes.
- **Dependência de Bibliotecas Externas:** Necessidade de bibliotecas adicionais para tarefas modernas.
- **Depuração Difícil:** Complexidade que torna a depuração e manutenção mais desafiadoras.

Escrevendo “Hello World!” na tela, com C++, Fortran, Python e BIRL.

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     cout << "Hello World!" << endl;
8
9     return 0;
10 }
```

```
1 program hello_world
2
3 write(*,*) "Hello World!"
4
5 end program
```

```
1 print("Hello World!")
```

```
1 HORA DO SHOW
2     CE QUER VER ESSA PORRA? ("Hello, World! Porra!\n");
3     BORA CUMPADE 0;
4 BIRL
```

- 1 Introdução
- 2 O que é Programar?
- 3 Linguagens de Programação. C++?
- 4 GitHub e IDEs
- 5 O que vamos ver neste curso?
- 6 Aula 01
- 7 Aula 02
- 8 Aula 03
- 9 Aula 04
- 10 Até mais, e obrigado pelos peixes!

## GitHub

Repositório de códigos computacionais e uma rede social (para programadores).

Faça a sua conta hoje mesmo!!! E visite as contas de outras pessoas.

[github.com/adrianopossebon](https://github.com/adrianopossebon)



**Adriano Possebon**  
adrianopossebon

Follow

Professor do Departamento de  
Engenharia Mecânica da Universidade  
de Brasília.

57 followers · 25 following

Universidade de Brasília.

Overview

Repositories 7

Projects

Packages

Stars 9

Pinned

**CFD-OpenFOAM-Curso-Introductorio** Public

Curso introdutório de CFD (Dinâmica dos Fluidos Computacional), usando o software OpenFOAM. Este curso foi ministrado sob o nome de Tópicos Especiais em Sistemas Térmicos, durante o segundo semestre...

2 ⚡ 1

**Curso-Metodos-Numericos-em-Termofluidos** Public

Material do curso Métodos Numéricos em Termofluidos (MNT). O curso é voltado para alunos das engenharias e tem como objetivo introduzir conhecimentos sobre simulação numérica, com foco nas equações...

1

**TCM-Transporte-de-Calor-e-Massa** Public

Material do curso de Transporte de Calor e Massa.

5

**Cavidade-Cisalhante-Roteiro-MNT** Public

Roteiro e código em python para resolver o problema da cavidade cisalhante. Disciplina: Métodos Numéricos em Termofluidos.

Python 1 ⚡ 1

**SU21-Curso-de-Python** Public

Material do Curso de Python para Iniciantes ministrado na Semana Universitária de 2021 da Universidade de Brasília.

Jupyter Notebook 22 ⚡ 2

19 contributions in the last year

## IDEs

Integrated Development Environments ou Ambientes de Desenvolvimento Integrado.

Ferramentas que combinam várias funcionalidades em um único software para facilitar o desenvolvimento de programas

Ajudam a desenvolver, executar, testar, encontrar erros e integrar diferentes códigos de um mesmo projeto.

## Melhores IDEs:

- Visual Studio Code (VS Code)
- Code::Blocks
- Eclipse
- CodeLite
- Atom
- CLion

# VS Code

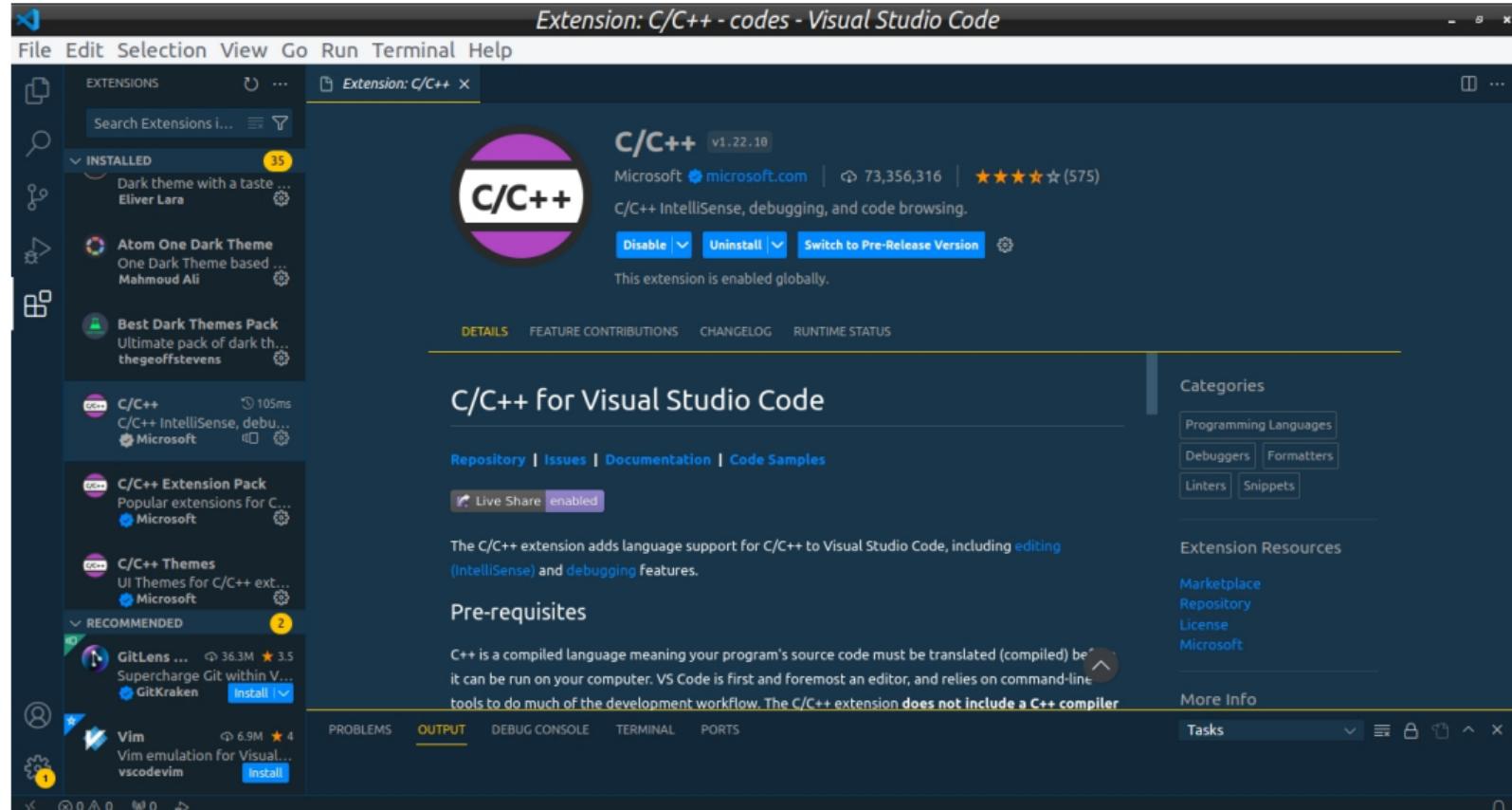
The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** aula04code01.cpp - codes - Visual Studio Code
- File Menu:** File Edit Selection View Go Run Terminal Help
- Explorer Bar (Left):**
  - CODES folder containing .vscode, aula01code01, aula01code01.cpp, aula02code01, aula02code02, aula03code01, aula04code01, aula04code01.cpp, tput.log, and zz.
  - A yellow circle with a '1' is visible next to the folder icon.
- Code Editor (Main Area):**

```
1 #include <iostream>
2 #include <vector>
3 #include <cmath>
4
5 using namespace std;
6
7 // Classe para resolver EDOs usando o Método de Euler
8 class EulerSolver {
9 private:
10     double y0;           // Condição inicial
11     double t0;           // Tempo inicial
12     double h;            // Tamanho do passo
13     int steps;          // Número de passos
14
15 public:
16     // Construtor para inicializar o solver com condições iniciais e parâmetros
```
- Terminal Tab:** TERMINAL
- Terminal Output:**

```
1.7 0.022518
1.8 0.0180144
1.9 0.0144115
2 0.0115292
[1] + Done "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-lwi4jhe3.sah" 1>"/tmp/Microsoft-MIEngine-Out-fup3vley.bk0"
possebon@~/APR1/professor/extensao/2024/2_Semuni/ eventosMeus/00semanaDaSimulacaoComputacionalDaFT/00cursoCppPossebon/codes$:
```
- Status Bar:** Ln 12, Col 20 Spaces: 4 UTF-8 LF ( C++ Linux Prettier

## VS Code



Vou usar o **VS Code** e recomendo que você o faça também.

Não sei como instalar o VS Code no Windows, mas você pode encontrar vários vídeos no youtube ensinando a baixar o VS Code e a configurar para rodar códigos em C++.

Alguns vídeos:

[youtube.com/watch?v=3pfRvy\\_gfqY](https://www.youtube.com/watch?v=3pfRvy_gfqY)

[youtube.com/watch?v=pYcneq-aOaQ](https://www.youtube.com/watch?v=pYcneq-aOaQ)

- 1 Introdução
- 2 O que é Programar?
- 3 Linguagens de Programação. C++?
- 4 GitHub e IDEs
- 5 O que vamos ver neste curso?
- 6 Aula 01
- 7 Aula 02
- 8 Aula 03
- 9 Aula 04
- 10 Até mais, e obrigado pelos peixes!

Aqui vamos ver o básico de **C++**, voltado para programação numérica.

Teremos **8 horas de curso**, em **2 dias**.

Vamos tentar entender como a linguagem funciona.

O objetivo deste curso é começar a sua jornada em C++, e em programação numérica.

Vamos com calma, um passo de cada vez.

No final de cada aula serão propostos alguns **exercícios**.

**É muito importante** que você os faça.

**Só se aprende a programar programando.**

**Mais: só se aprende a programar errando... e errando muito!**

Nossas 4 aulas estão divididas da seguinte forma.

# **Aula 01. Introdução ao C++ e Método da Bissecção**

## **Objetivos**

- Apresentar uma visão geral do C++ e configurar o ambiente.
- Aprender sobre operadores, sintaxe básica, loops e condicionais.
- Introduzir o Método da Bissecção para encontrar raízes de funções.

# Conteúdo

- **Visão Geral do C++**
  - Estrutura básica de um programa em C++.
  - Entrada e saída com cin e cout.
- **Operadores e Sintaxe Básica**
  - Operadores aritméticos, de atribuição, comparação e lógicos.
- **Loops e Condicionais**
  - Estrutura if, else, while e for.
- **Método da Bissecção**
  - Explicar o conceito e a aplicação do método.
  - Implementar o algoritmo da bissecção para encontrar a raiz de uma função.

```
27     double f_c = 1.0;
28
29     // Implementação do método da bissecção
30     while (fabs(b-a) > tol && iter < maxIter)
31     {
32         c = (a + b) / 2.0; // Ponto médio
33         f_c = c * c - 5; // Avaliação da função f(c)
34
35         if (f_c == 0.0) {
36             break; // Raiz encontrada
37         } else if (f_c * (a * a - 5) < 0) {
38             b = c; // A raiz está no intervalo [a, c]
39         } else {
40             a = c; // A raiz está no intervalo [c, b]
41         }
42     }
43 }
```

## Aula 02. Funções, Cálculos de Médias e Método do Ponto Fixo

### Objetivos

- Explicar o uso de funções em C++.
- Praticar cálculos de média, somas e séries.
- Apresentar o Método do Ponto Fixo para encontrar raízes de funções.

# Conteúdo

- **Funções em C++**
  - Definição, chamada de funções, passagem de parâmetros e retorno de valores.
- **Cálculos de Médias, Somas e Séries**
  - Exemplos práticos de cálculo de média e somas usando loops.
- **Método do Ponto Fixo**
  - Conceito e aplicação do método.
  - Implementação de um exemplo de ponto fixo em C++.

```
5 double calcularMedia(double valores[], int tamanho)
6 {
7     double soma = 0.0;
8     for (int i = 0; i < tamanho; i++)
9     {
10         soma += valores[i];
11     }
12     return soma / tamanho;
13 }
```

## **Aula 03. Vetores, Matrizes e Método de Gauss-Seidel**

### **Objetivos**

- Introduzir vetores e matrizes em C++.
- Implementar operações básicas com vetores e matrizes.
- Apresentar o Método de Gauss-Seidel para sistemas de equações lineares.

# Conteúdo

- **Vetores e Matrizes**

- Declaração, inicialização e acesso a elementos.
- Exemplo de cálculo com vetores e matrizes.

- **Método de Gauss-Seidel**

- Explicação do método para resolver sistemas lineares.
- Implementação do método em C++.



# Aula 04. Programação Orientada a Objetos (OOP) e Diferenças Finitas

## Objetivos

- Apresentar os fundamentos de OOP (classes, objetos, encapsulamento).
- Implementar uma classe para resolver equações diferenciais usando diferenças finitas.

# **Conteúdo**

- **Introdução à OOP em C++**
  - Conceitos de classe, objeto, atributos e métodos.
  - Encapsulamento e uso de classes.
- **Diferenças Finitas**
  - Teoria de diferenças finitas para aproximação de derivadas.
  - Implementação de um exemplo usando uma classe.

```
8     double y0;          // Condição inicial
9     double t0;          // Tempo inicial
10    double tf;          // Tempo final
11    double h;           // Tamanho do passo
12
13 public:
14     // Construtor para inicializar o solver com condições iniciais e parâmetros
15     EulerSolver(double y_init, double t_init, double t_final, double step_size)
16         : y0(y_init), t0(t_init), tf(t_final), h(step_size) {}
17
18     // Função que define a EDO  $dy/dt = -2y$ 
19     double odeFunction(double t, double y)
20     {
21         return -2 * y;
22     }
23
24     // Método para resolver a EDO usando o Método de Euler
25     void solve() {
```

**Alguma dúvida?**

**Comentário?**

Vamos começar!

- 1 Introdução
- 2 O que é Programar?
- 3 Linguagens de Programação. C++?
- 4 GitHub e IDEs
- 5 O que vamos ver neste curso?
- 6 Aula 01
- 7 Aula 02
- 8 Aula 03
- 9 Aula 04
- 10 Até mais, e obrigado pelos peixes!

# Introdução C++ e Método da Bissecção

## Objetivos

- Apresentar uma visão geral do C++ e configurar o ambiente.
- Aprender sobre operadores, sintaxe básica, loops e condicionais.
- Introduzir o Método da Bissecção para encontrar raízes de funções.

# Método da Bissecção

**Definição:** Método iterativo usado para encontrar o zero de uma função contínua  $f(x)$  em um intervalo  $[a, b]$  onde  $f(a) \cdot f(b) < 0$ .

## Condições Necessárias:

- $f(x)$  deve ser contínua em  $[a, b]$ .
- Sinais opostos em  $f(a)$  e  $f(b)$ :  $f(a) \cdot f(b) < 0$ .

## Ideia Principal:

- Dividir o intervalo  $[a, b]$  ao meio repetidamente.
- Selecionar o subintervalo onde ocorre a mudança de sinal.
- Repetir até atingir uma tolerância desejada.

# Passo a Passo do Método da Bissecção

- ① Calcule o ponto médio:  $c = \frac{a+b}{2}$ .
- ② Avalie a função em  $c$ :  $f(c)$ .
- ③ Verifique o sinal de  $f(c)$ :
  - Se  $f(c) = 0$ , então  $c$  é a raiz.
  - Se  $f(a) \cdot f(c) < 0$ , a raiz está em  $[a, c]$ .
  - Se  $f(c) \cdot f(b) < 0$ , a raiz está em  $[c, b]$ .
- ④ Atualize o intervalo  $[a, b]$  e repita até que  $|b - a|$  seja menor que a tolerância.

# Algoritmo do Método da Bissecção

- **Input:** Intervalo  $[a, b]$ , tolerância tol, função  $f(x)$ .
- **Passo 1:** Verifique  $f(a) \cdot f(b) < 0$ .
- **Passo 2:** Enquanto  $|b - a| > tol$ , repita:
  - Calcule  $c = \frac{a+b}{2}$ .
  - Se  $f(c) = 0$ , finalize ( $c$  é a raiz).
  - Se  $f(a) \cdot f(c) < 0$ , atualize  $b = c$ .
  - Se  $f(c) \cdot f(b) < 0$ , atualize  $a = c$ .
- **Output:** Valor aproximado da raiz  $c$ .

# Exemplo

**Exemplo:** Encontrar a raiz de  $f(x) = x^2 - 5$  no intervalo  $[1, 6]$  com tolerância de 0.0001.

# Programando

Agora que já estudamos a parte teórica e já temos um rascunho do código, vamos para o C++.

aula01code01.cpp

## **Exercícios para casa.**

- 1) Implemente um programa em C++ que leia dois números inteiros e calcule o valor da sua média. Exiba o resultado no formato A média é: ....
- 2) Escreva um programa em C++ que peça ao usuário para inserir um número e verifique se ele é positivo, negativo ou zero, utilizando condicionais if, else if, e else.

## Exercícios para casa.

- 3) Modifique o código do Método da Bissecção para resolver uma nova função, como  $f(x) = x^3 - x - 1$ . Peça ao usuário para inserir o intervalo  $[a, b]$  e a tolerância. Exiba a raiz encontrada pelo método.
- 4) (**Desafio**) Implemente uma versão aprimorada do Método da Bissecção que armazena todas as iterações em um vetor. No final do programa, exiba todas as aproximações de  $x$  feitas durante o cálculo da raiz. Inclua também uma condição para exibir uma mensagem se o número máximo de iterações for atingido sem que a tolerância seja alcançada.

- 1 Introdução
- 2 O que é Programar?
- 3 Linguagens de Programação. C++?
- 4 GitHub e IDEs
- 5 O que vamos ver neste curso?
- 6 Aula 01
- 7 Aula 02
- 8 Aula 03
- 9 Aula 04
- 10 Até mais, e obrigado pelos peixes!

# Funções, Cálculo de Médias e Método do Ponto Fixo

## Objetivos

- Explicar o uso de funções em C++.
- Praticar cálculos de média, somas e séries.
- Apresentar o Método do Ponto Fixo para encontrar raízes de funções.

# Funções em C++

- **Função:** Bloco de código que executa uma tarefa específica.
- **Componentes:**
  - Definição e chamada de função.
  - Parâmetros e retorno de valores.
- **Vantagens:**
  - Modularidade e reutilização de código.
  - Clareza na organização do programa.

# Cálculos de Médias e Séries

- **Cálculo de Média:** Soma dos elementos dividida pelo número de elementos.
  - Exemplo: Média de notas, valores de sensores, etc.
- **Séries:** Sequência de somas acumulativas.
  - Exemplo: Somatória de uma série matemática.
  - Útil em cálculos financeiros, estatísticos e de física.

$$S = \sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

# Programando

aula02code01.cpp

aula02code02.cpp

# Método do Ponto Fixo

- **Definição:** Método iterativo para encontrar raízes de equações da forma  $f(x) = 0$ .
- **Conceito:** Reescrever a equação  $f(x) = 0$  na forma  $x = g(x)$ .
- **Iteração:**
  - Começar com um chute inicial  $x_0$ .
  - Calcular  $x_{n+1} = g(x_n)$  até que a convergência seja atingida.

# Condições para Convergência

- A função  $g(x)$  deve ser contínua no intervalo.
- $g(x)$  deve ser contraída:  $|g'(x)| < 1$  no intervalo de interesse.
- A escolha do chute inicial deve ser feita com cuidado.

# Exemplo

**Exemplo:** Encontrar um zero da função  $f(x) = x - \cos(x)$ .

- **Função Iterativa:**  $g(x) = \cos(x)$
- **Chute Inicial:**  $x_0 = 0.5$
- **Parâmetros:**
  - Tolerância: 0.0001
  - Máximo de Iterações: 100

# Algoritmo

**Entrada:** Função  $g(x)$ , chute inicial, tolerância, número máximo de iterações.

**Passos:**

- ① Inicializar  $x$  com o chute inicial.
- ② Para  $i = 0$  até maxIter:
  - Calcular  $x_{\text{novo}} = g(x)$ .
  - Se  $|x_{\text{novo}} - x| < \text{tolerancia}$ , retornar  $x_{\text{novo}}$ .
  - Atualizar  $x = x_{\text{novo}}$ .

**Saída:** Raiz aproximada.

# Programando

aula02code03.cpp

## Exercícios para casa.

- 1) Crie uma função `calcular_media` que recebe três números e retorna a média aritmética deles. No `main`, peça ao usuário para inserir três valores e exiba a média calculada.
- 2) Implemente uma função `soma_naturais` que recebe um número inteiro  $n$  e retorna a soma dos  $n$  primeiros números naturais. No `main`, solicite um valor de  $n$  ao usuário e exiba a soma resultante.

## Exercícios para casa.

- 3) Modifique o código do Método do Ponto Fixo para resolver uma função diferente, como  $g(x) = \sqrt{2 + x}$ . Use uma condição inicial fornecida pelo usuário e exiba as iterações até a solução.
- 4) **(Desafio)** Crie um programa que implementa o Método do Ponto Fixo para a função  $g(x) = \cos(x)$ . Inclua uma verificação para garantir a convergência antes de iniciar o método. Exiba cada iteração até que a tolerância seja atingida ou o número máximo de iterações seja alcançado.

- 1 Introdução
- 2 O que é Programar?
- 3 Linguagens de Programação. C++?
- 4 GitHub e IDEs
- 5 O que vamos ver neste curso?
- 6 Aula 01
- 7 Aula 02
- 8 Aula 03
- 9 Aula 04
- 10 Até mais, e obrigado pelos peixes!

# Solução de Sistemas Lineares e Método de Gauss-Seidel

## Objetivos

- Introduzir vetores e matrizes em C++.
- Implementar operações básicas com vetores e matrizes.
- Apresentar o Método de Gauss-Seidel para sistemas de equações lineares.

# Sistemas Lineares de Equações

- Um sistema linear de equações é dado por:

$$A \cdot \mathbf{x} = \mathbf{b}$$

onde:

- $A$ : Matriz de coeficientes.
- $\mathbf{x}$ : Vetor incógnita.
- $\mathbf{b}$ : Vetor constante.

- Exemplo:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases}$$

# Método de Gauss-Seidel

- **Gauss-Seidel:** Método iterativo para resolver  $A \cdot \mathbf{x} = \mathbf{b}$ .
- **Ideia Principal:** Atualizar cada variável usando valores mais recentes.
- Adequado para sistemas grandes e esparsos.
- Condição de Convergência:
  - A matriz  $A$  deve ser diagonalmente dominante.

# Método de Gauss-Seidel

- Para resolver cada equação, isolamos uma incógnita em função das outras:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij}x_j^{(k)} \right)$$

- Passo a Passo:**
  - Usar valores atualizados assim que disponíveis.
  - Iterar até que a diferença entre iterações seja menor que uma tolerância.

# Algoritmo do Método de Gauss-Seidel

**Entrada:** Matriz  $A$ , vetor  $\mathbf{b}$ , tolerância  $\text{tol}$ , número máximo de iterações  $\text{maxIter}$ .

**Passos:**

- ① Inicializar  $\mathbf{x}^{(0)}$  com valores iniciais (ex: zeros).
- ② Para cada iteração  $k = 1, 2, \dots$ :
  - ① Para cada variável  $i = 1, 2, \dots, n$ :
    - Atualizar  $x_i^{(k+1)} = \frac{1}{a_{ii}} \left( b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right)$ .
  - ② Calcular erro relativo.
- ③ Parar se erro  $< \text{tol}$  ou atingir  $\text{maxIter}$ .

**Exemplo:** Resolver o sistema:

$$\begin{cases} 3x + y = 5 \\ x + 2y = 4 \end{cases}$$

**Passo a Passo:**

- Initialize  $x = 0, y = 0$ .
- Iterar usando as fórmulas do Gauss-Seidel para obter aproximações sucessivas.

# Programando

aula03code01.cpp

## **Exercícios para casa.**

- 1) Crie um programa que declare um vetor de inteiros com 5 elementos e preencha-o com valores aleatórios. Em seguida, exiba todos os valores do vetor.
- 2) Escreva um programa que declare uma matriz  $2 \times 2$  e preencha-a com valores fornecidos pelo usuário. Calcule a soma de todos os elementos da matriz e exiba o resultado.

## Exercícios para casa.

3) Implemente o Método de Gauss-Seidel para resolver o sistema linear

$$\begin{cases} 12x - 2y + 3z + w = 0 \\ x + 6y + 20z - 4w = 20 \\ -2x + 15y + 6z - 3w = 0 \\ 0x - 3y + 2z + 9w = 0 \end{cases}$$

Use uma condição de parada baseada em uma tolerância e exiba o número de iterações realizadas.

4) (**Desafio**) Expanda o Método de Gauss-Seidel para resolver sistemas lineares de dimensão  $n \times n$ . Solicite ao usuário os coeficientes do sistema e a condição inicial. Use uma tolerância para determinar a convergência e exiba as aproximações para cada variável.

- 1 Introdução
- 2 O que é Programar?
- 3 Linguagens de Programação. C++?
- 4 GitHub e IDEs
- 5 O que vamos ver neste curso?
- 6 Aula 01
- 7 Aula 02
- 8 Aula 03
- 9 Aula 04
- 10 Até mais, e obrigado pelos peixes!

# Diferenças Finitas e Método de Euler

## Objetivos

- Apresentar os fundamentos de OOP (classes, objetos, encapsulamento).
- Implementar uma classe para resolver equações diferenciais usando diferenças finitas.

## Conceitos Básicos

- **Classe:**
  - Estrutura que define um tipo de dado, incluindo atributos e métodos.
  - Serve como um molde para criar objetos.
- **Objeto:**
  - Instância de uma classe, que contém valores específicos para os atributos.
  - Pode utilizar os métodos definidos na classe.

## Exemplo de Classe e Objeto

- `class Carro { ... };`
- `Carro meuCarro; // Instância de um objeto da classe Carro`

# Vantagens OOP

- **Encapsulamento:**
  - Protege dados e métodos, permitindo acesso controlado.
  - Facilita a manutenção e evolução do código.
- **Reutilização de Código:**
  - Classes podem ser reutilizadas em diferentes partes do programa.
  - Facilita a criação de sistemas mais complexos com menos redundância.

# Diferenças Finitas

- **Diferenças Finitas:** Método para aproximar derivadas.
- Baseia-se na substituição de derivadas por diferenças entre valores próximos.
- **Exemplo:** Derivada primeira de  $y(x)$  em  $x = x_0$ :

$$y'(x_0) \approx \frac{y(x_0 + h) - y(x_0)}{h}$$

onde  $h$  é o **passo** da aproximação.

# Aplicação em Equações Diferenciais

- Usamos diferenças finitas para resolver EDOs numericamente.
- Método de Euler é uma aplicação simples de diferenças finitas para EDOs.
- Converte a EDO em uma sequência de operações aritméticas.

# Método de Euler

- **Ideia Principal:** Aproximar a solução de uma EDO inicial.
- Dada uma EDO da forma  $y'(t) = f(t, y)$  com condição inicial  $y(t_0) = y_0$ .
- O método aproxima  $y(t)$  em passos  $h$ :

$$y_{n+1} = y_n + h \cdot f(t_n, y_n)$$

- Quanto menor o passo  $h$ , maior a precisão (e maior o custo computacional).

# Algoritmo do Método de Euler

**Entrada:** Função  $f(t, y)$ , intervalo de tempo  $[t_0, t_f]$ , condição inicial  $y_0$ , passo  $h$ .

**Passos:**

- ① Inicializar  $t = t_0$ ,  $y = y_0$ .
- ② Enquanto  $t \leq t_f$ :
  - Calcular a derivada:  $f(t, y)$ .
  - Atualizar  $y = y + h \cdot f(t, y)$ .
  - Incrementar  $t = t + h$ .

**Saída:** Aproximação da solução  $y(t)$  no intervalo  $[t_0, t_f]$ .

# Exemplo

**Exemplo:** Resolver  $y' = -2y$  com  $y(0) = 1$  e  $h = 0.1$ .

- Condição inicial:  $t_0 = 0$ ,  $y_0 = 1$ .
- Iterar o método de Euler:

$$y_{n+1} = y_n + h \cdot (-2y_n)$$

- Obter valores aproximados de  $y(t)$  em intervalos de  $h$ .

# Programando

aula04code01.cpp

## **Exercícios para casa.**

- 1) Crie uma classe `Circulo` que possui um atributo `raio`. Adicione métodos para calcular a área e o perímetro do círculo. No `main`, solicite o valor do raio ao usuário e exiba a área e o perímetro.
- 2) Implemente uma classe `Retangulo` com atributos `largura` e `altura`. Adicione métodos para calcular a área e o perímetro. No `main`, crie um objeto `Retangulo`, defina suas dimensões e exiba os resultados.

## Exercícios para casa.

- 3) Implemente uma classe `EDOSolver` para resolver uma EDO usando o método de Euler para a função  $f(t, y) = -3y$ , com condição inicial  $y(0) = 1$ . Permita que o passo e o número de iterações sejam definidos pelo usuário e exiba a solução.
- 4) (**Desafio**) Expanda a classe `EDOSolver` para resolver a EDO  $y' = y \cdot \sin(t)$  com condição inicial  $y(0) = 1$ . Adicione um método `salvarDados` que salva as soluções  $t$  e  $y$  em um arquivo de texto `resultados.txt`.

- 1 Introdução
- 2 O que é Programar?
- 3 Linguagens de Programação. C++?
- 4 GitHub e IDEs
- 5 O que vamos ver neste curso?
- 6 Aula 01
- 7 Aula 02
- 8 Aula 03
- 9 Aula 04
- 10 Até mais, e obrigado pelos peixes!

## **Fechamos nosso curso!**

Vimos aqui o básico do básico.

Há muita coisa para explorar.

Mas lembre-se sempre do seu objetivo principal.

**Aprender a programar apenas para aprender a programar não funciona.**

Deixo aqui algumas dicas de cursos, livros e sites.

# Cursos:



## Beginning C++ Programming - From Beginner to Beyond

Obtain Modern **C++** Object-Oriented Programming (OOP) and STL skills. C++14 and C++17 covered. C++20 info see below.

Tim Buchalka's Learn Programming Academy, Dr. Frank Mitropoulos

**4.6 ★★★★★ (72,712)**

46 total hours • 305 lectures • All Levels

**Bestseller**



## Unreal Engine 5 C++ Developer: Learn C++ & Make Video Games

Created in collaboration with Epic Games. Learn **C++** from basics while making your first 5 video games in Unreal

Ben Tristem, GameDev.tv Team

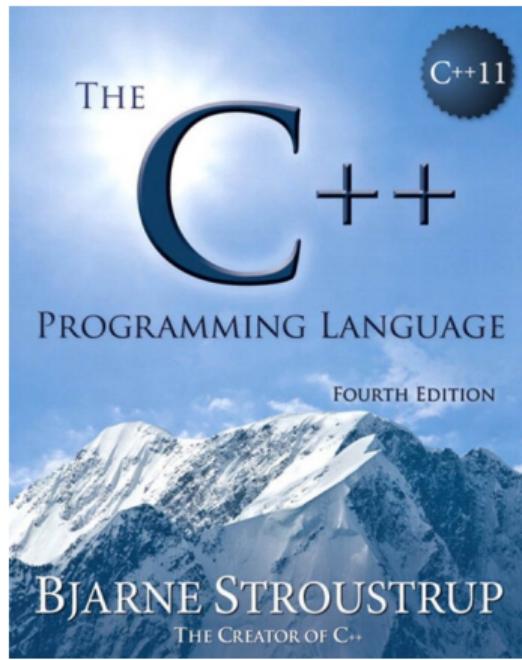
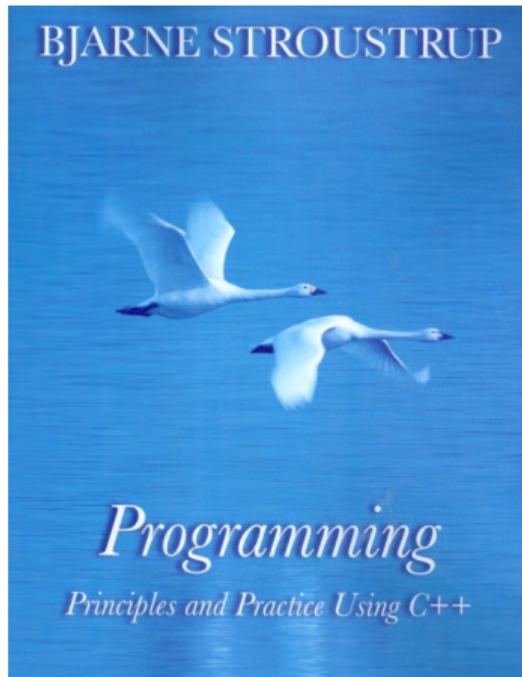
**4.6 ★★★★★ (71,062)**

29.5 total hours • 210 lectures • All Levels

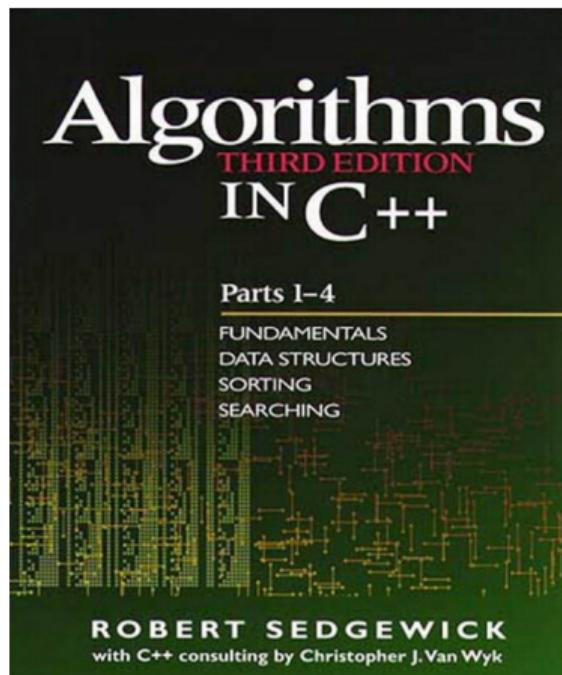
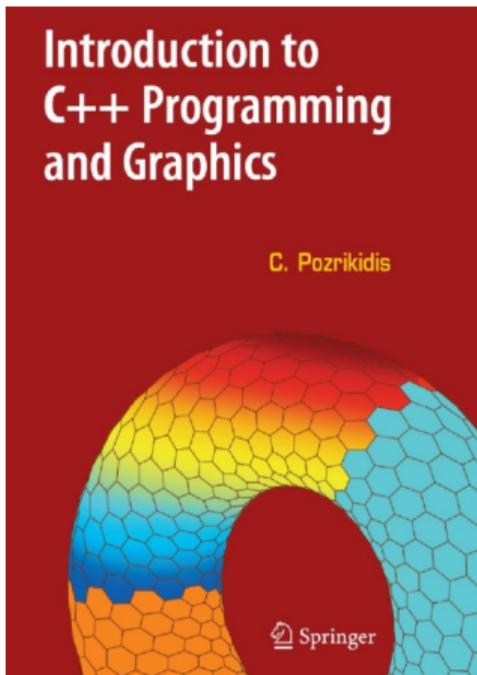
**Bestseller**

(Não entre no “Tutorial Hell” !!!)

Livros:



Livros:



Sites:

[learncpp.com](http://learncpp.com)

[en.cppreference.com/w/cpp/standard\\_library](http://en.cppreference.com/w/cpp/standard_library)

[codewars.com](http://codewars.com)

Muito obrigado!

aproса@unb.br