



Introdução à Programação de Computadores

Computação para Engenharia – Tópico 1

Daniel Guerreiro e Silva

Departamento de Engenharia Elétrica (ENE), Faculdade de Tecnologia (FT)

Roteiro

Organização de um computador

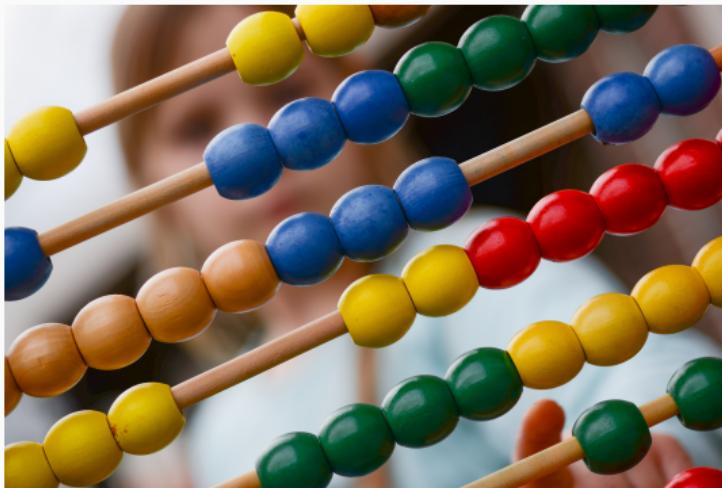
Algoritmos

A linguagem C++

Organização de um computador

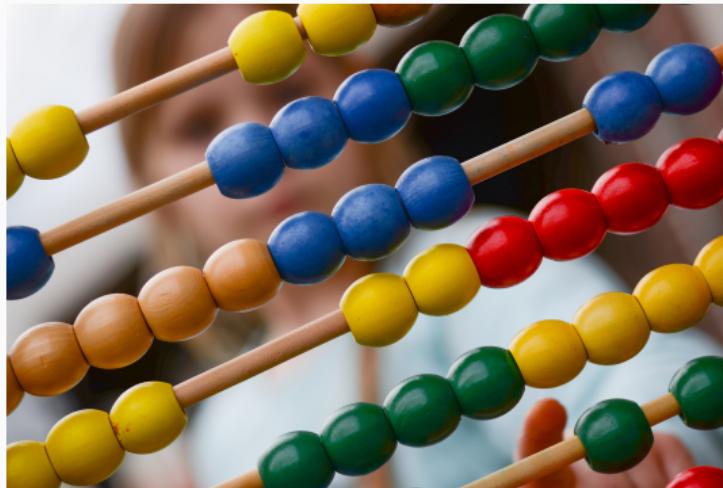
O que é um computador?

- Computador: o que computa, calculador, calculista (dicionário Houaiss).



O que é um computador?

- Computador: o que computa, calculador, calculista (dicionário Houaiss).
- Os primeiros “computadores” eram humanos que calculavam tabelas de logaritmos ou trajetórias para canhões, seguindo procedimentos bem definidos.



O que é um computador?

- Um computador é uma máquina que, a partir de uma entrada, realiza um número muito grande de cálculos matemáticos e lógicos, gerando uma saída.

O que é um computador?

- Um computador é uma máquina que, a partir de uma entrada, realiza um número muito grande de cálculos matemáticos e lógicos, gerando uma saída.
- **Máquina de propósito geral**, que executa tarefas e resolve problemas como cálculos e simulações complexas, geração de relatórios, comando de outras máquinas, controle de contas bancárias, processamento e comunicação de informações, etc..

Hardware e dispositivos

Um computador é um dispositivo eletrônico digital. A linguagem nativa do computador é codificada numericamente, isto é, informações como imagens, vídeos, textos, são representadas por números em formato **binário**.

Hardware e dispositivos

Um computador é um dispositivo eletrônico digital. A linguagem nativa do computador é codificada numericamente, isto é, informações como imagens, vídeos, textos, são representadas por números em formato **binário**.

Representação Binária

- Bit → unidade elementar da representação binária, pode assumir valores 0 ou 1.
- Byte → agrupamento de 8 bits.

Hardware e dispositivos

Exemplo: números de 4 bits

Base 10

0	$= 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$	= 0000
1	$= 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$	= 0001
2	$= 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$	= 0010
3	$= 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$	= 0011
4	$= 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$	= 0100
5	$= 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$	= 0101
6	$= 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$	= 0110
7	$= 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$	= 0111
8	$= 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$	= 1000
9	$= 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$	= 1001
10	$= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$	= 1010

Base 2

Hardware e dispositivos

Uma sequência de instruções com alguma finalidade constitui um **programa** de computador.

```
3 require File.expand_path('../..', __FILE__)
4 # Prevent database truncation if the environment is test or development
5 abort("The test environment is running a database migration!")
6 require 'spec_helpers'
7 require 'capybara/spec'
8 require 'capybara/rspec'
9
10 Capybara.javascript_driver = :webkit
11 Category.delete_all
12 ShouldMatchers.configure do |config|
13   config.integrate do |builder|
14     builder.with_text_framework :rspec
15     builder.with_library :rspec
16   end
17 end
18
19 # Add additional requirements below this line if you need them.
20
21 # Requires supporting files with helper methods
22 # located in spec/support/ and its subdirectories
23 # spec/support/_ and its subdirectories
24 # run as spec files by default. You can also
25 # run as rspec files by adding
26 # in spec.rb will load the
27 # run twice. It is recommended
28 # end with _spec.rb. You can change
# results found for targeted
```

Hardware e dispositivos

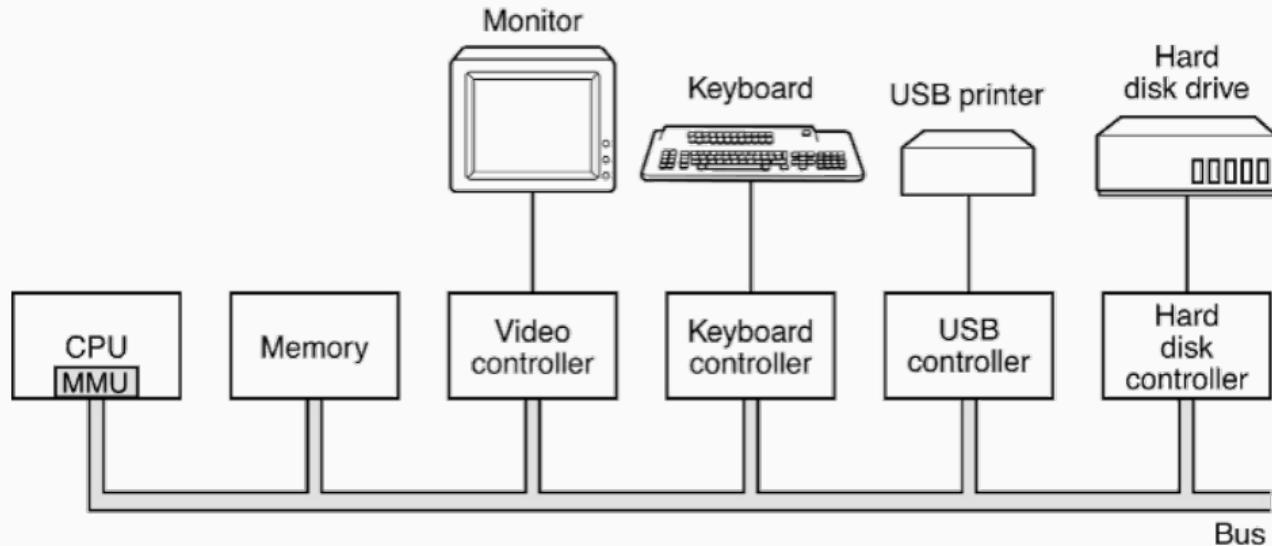


Figura 1: Arquitetura básica de um computador (A. Tanenbaum, H. Bos. *Modern Operating Systems*. 4th Edition.)

Organização básica de um ambiente computacional

- Computadores realizam tarefas complexas por meio de um número enorme de operações simples, em geral **sequencialmente**.
- Para gerenciar a complexidade das soluções, existe uma hierarquia de funções, onde cada uma apresenta uma interface mais simples.



Programando computadores

- Como usuários, interagimos com os programas de aplicação.

Programando computadores

- Como usuários, interagimos com os programas de aplicação.
- O objetivo principal deste curso é descer nesta hierarquia, para construirmos novos programas de aplicação.

Programando computadores

- Como usuários, interagimos com os programas de aplicação.
- O objetivo principal deste curso é descer nesta hierarquia, para construirmos novos programas de aplicação.
- Estaremos interessados em **algoritmos** e em implementá-los como **programas de computadores**, utilizando em particular a linguagem de programação **C++**.

Por quê aprender a programar?

Computação é uma tecnologia ubíqua (onipresente).

- Controladores programáveis para controle de processos industriais.

Por quê aprender a programar?

Computação é uma tecnologia ubíqua (onipresente).

- Controladores programáveis para controle de processos industriais.
- Sistemas embutidos (embarcados) em carros, eletrodomésticos, celulares.

Por quê aprender a programar?

Computação é uma tecnologia ubíqua (onipresente).

- Controladores programáveis para controle de processos industriais.
- Sistemas embutidos (embarcados) em carros, eletrodomésticos, celulares.
- Todas as engenharias utilizam aplicações baseadas em computador para auxílio nas suas atividades.

Ainda assim, por quê tenho que programar?

Exemplos de atividades de um engenheiro de redes

- Desenvolvimento de aplicações / sistemas distribuídos.
- Desenvolvimento de algoritmos / protocolos de comunicação.
- Modelagem, análise e simulação de redes em computador.

Ainda assim, por quê tenho que programar?

Exemplos de atividades de um engenheiro de redes

- Desenvolvimento de aplicações / sistemas distribuídos.
- Desenvolvimento de algoritmos / protocolos de comunicação.
- Modelagem, análise e simulação de redes em computador.

Aprender a programar implica em desenvolver / aprimorar a habilidade em construir algoritmos para solucionar problemas.

Algoritmos

Algoritmos

Definição

Algoritmo é uma sequência de passos, precisos e bem definidos, para a realização de uma tarefa.

Como descrever um algoritmo

Algoritmos podem ser especificados de várias formas, inclusive em português.

Como descrever um algoritmo

Algoritmos podem ser especificados de várias formas, inclusive em português.

Exemplo

Como calcular $2345 + 4567$ usando lápis e papel?

Exercício

Escreva um algoritmo em português mostrando passo a passo como preparar um cachorro quente.

- Assuma que está em casa e que já possui os ingredientes.
 - Lembre-se: não pode ser ambíguo, pois o computador não sabe absolutamente nada sobre nada.
1. Ir até a cozinha.
 2. ...
 - N. ...

De algoritmos a programas

Como transformar um algoritmo em linguagem que o computador entenda?

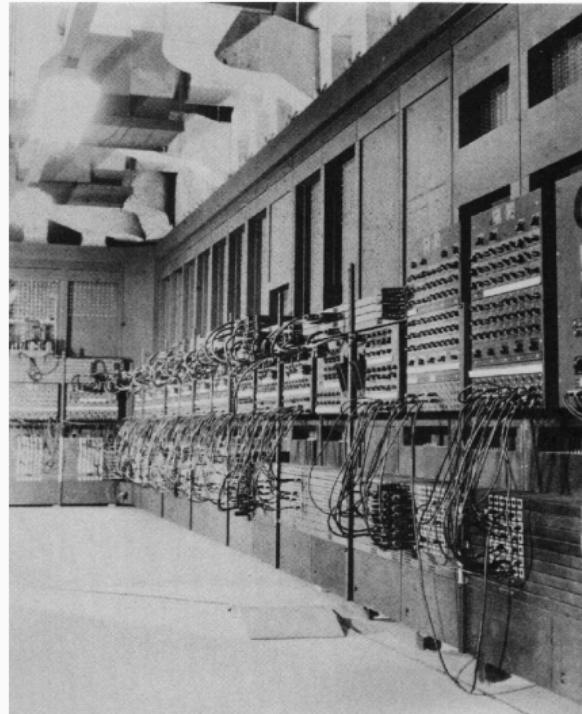
- Deve ser capaz de expressar tudo o que o computador pode fazer: cobre todo o **conjunto de instruções** da CPU.
- Não pode ser ambígua.

Um pouco de história

ENIAC

- Entrou em operação no final de 1945, 30 toneladas, 270m² de espaço, 5000 operações por segundo.
- Programação por meio de painéis de circuitos.

Saiba mais em <https://www.computerhistory.org/timeline/computers/>



Linguagem Simbólica ou de Montagem (Assembly)

Um programa, chamado montador ou *assembler*, faz a transformação para a linguagem de máquina.

```
LOOP:  MOV A, 3  
      INC A  
      JMP LOOP
```

Linguagens de alto nível

- Mais distantes da máquina e mais próximas de linguagens naturais (inglês, português, etc.).
- Mesmo mais compreensíveis, elas não são ambíguas.
- Um programa especial denominado **compilador** as transforma em código executável (linguagem de máquina).

Linguagens de alto nível

- Mais distantes da máquina e mais próximas de linguagens naturais (inglês, português, etc.).
- Mesmo mais compreensíveis, elas não são ambíguas.
- Um programa especial denominado **compilador** as transforma em código executável (linguagem de máquina).

Exemplos de linguagens

- C
- Python
- Java
- C++

A linguagem C++

A linguagem C++

- Desenvolvida a partir de 1979 por Bjarne Stroustrup



Figura 2: By Bjarne Stroustrup's homepage, GFDL, <https://commons.wikimedia.org/w/index.php?curid=188666>

A linguagem C++

- Desenvolvida a partir de 1979 por Bjarne Stroustrup
 - Contato durante o seu PhD com a linguagem Simula 67, a 1^a linguagem orientada a objeto.



Figura 2: By Bjarne Stroustrup's homepage, GFDL, <https://commons.wikimedia.org/w/index.php?curid=188666>

A linguagem C++

- Desenvolvida a partir de 1979 por Bjarne Stroustrup
 - Contato durante o seu PhD com a linguagem Simula 67, a 1^a linguagem orientada a objeto.
- Desenvolve o projeto da linguagem “C with Classes”.



Figura 2: By Bjarne Stroustrup's homepage, GFDL, <https://commons.wikimedia.org/w/index.php?curid=188666>

A linguagem C++

- Desenvolvida a partir de 1979 por Bjarne Stroustrup
 - Contato durante o seu PhD com a linguagem Simula 67, a 1^a linguagem orientada a objeto.
- Desenvolve o projeto da linguagem “C with Classes”.
- 1983: linguagem muda o nome para C++



Figura 2: By Bjarne Stroustrup's homepage, GFDL, <https://commons.wikimedia.org/w/index.php?curid=188666>

A linguagem C++

- Desenvolvida a partir de 1979 por Bjarne Stroustrup
 - Contato durante o seu PhD com a linguagem Simula 67, a 1ª linguagem orientada a objeto.
- Desenvolve o projeto da linguagem “C with Classes”.
- 1983: linguagem muda o nome para C++
- 1998: primeiro padrão internacional da linguagem, C++98



Figura 2: By Bjarne Stroustrup's homepage, GFDL, <https://commons.wikimedia.org/w/index.php?curid=188666>

A linguagem C++

- Desenvolvida a partir de 1979 por Bjarne Stroustrup
 - Contato durante o seu PhD com a linguagem Simula 67, a 1ª linguagem orientada a objeto.
- Desenvolve o projeto da linguagem “C with Classes”.
- 1983: linguagem muda o nome para C++
- 1998: primeiro padrão internacional da linguagem, C++98
- Padrão mais recente é o C++17



Figura 2: By Bjarne Stroustrup's homepage, GFDL, <https://commons.wikimedia.org/w/index.php?curid=188666>

A linguagem C++

- Desenvolvida a partir de 1979 por Bjarne Stroustrup
 - Contato durante o seu PhD com a linguagem Simula 67, a 1ª linguagem orientada a objeto.
- Desenvolve o projeto da linguagem “C with Classes”.
- 1983: linguagem muda o nome para C++
- 1998: primeiro padrão internacional da linguagem, C++98
- Padrão mais recente é o C++17
 - <https://isocpp.org/>



Figura 2: By Bjarne Stroustrup's homepage, GFDL, <https://commons.wikimedia.org/w/index.php?curid=188666>

Primeiro programa em C++: hello.cpp

Um programa em C++ é um arquivo texto, contendo declarações e operações da linguagem. Isto é chamado de **código fonte**.

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

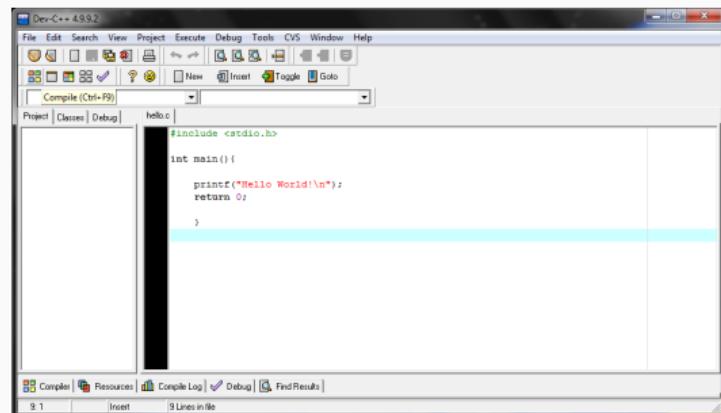
Como executar este programa

Para criar um programa a partir do seu código fonte é necessário compilá-lo, gerando o **código binário ou executável**. Este pode ser executado como qualquer outro programa de computador.

No Terminal do Linux

```
$ g++ hello.cpp -o hello  
$ ./hello  
Hello, world!
```

No Windows com o Dev-C++



O que são erros de compilação?

Caso o programa não esteja de acordo com as regras da linguagem, erros de compilação ocorrerão. **Ler e entender** estes erros é muito importante.

Programa

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     cout << "Hello , world!" << endl;
6     return 0;
```

O que são erros de compilação?

Caso o programa não esteja de acordo com as regras da linguagem, erros de compilação ocorrerão. **Ler e entender** estes erros é muito importante.

Programa

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     cout << "Hello , world!" << endl;
6     return 0;
```

Saída do compilador - Terminal

```
hello.cpp: In function `int main()':
hello.cpp:6:11: error: expected `}' at end of input
```

O que são erros de execução?

Acontecem quando o comportamento do programa diverge do esperado e podem acontecer mesmo quando o programa compila corretamente.

Programa

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     cout << "Hello, world! $#%#@%" << endl;
6     return 0;
7 }
```

O que são erros de execução?

Acontecem quando o comportamento do programa diverge do esperado e podem acontecer mesmo quando o programa compila corretamente.

Programa

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     cout << "Hello, world! $#%#@%" << endl;
6     return 0;
7 }
```

Saída do Terminal

\$ hello

Hello, world! \$#%#@%

Um exemplo mais complexo

```
1 /*CPE - Exemplo 2 topico 1*/
2 /*Prof. Daniel Guerreiro e Silva*/
3 #include <iostream>
4 using namespace std;
5
6 int main() {
7     int x, y;
8
9     cout << "x: ";
10    cin >> x;
11    cout << "y: ";
12    cin >> y;
13
14    if (x > y)
15        cout << "O maior numero e o x = " << x << endl;
16    else
17        cout << "O maior numero e o y = " << y << endl;
18
19    return 0;
20 }
```

Até o próximo tópico...

