

# FULL STACK WEB DEVELOPER

FERNANDO LIRA

***FLAG***

# INTRODUÇÃO À PROGRAMAÇÃO

# APRESENTAÇÃO – FERNANDO LIRA

3



it.fernandolira@gmail.com



<https://www.linkedin.com/in/fernandolira74/>



+351 93 317 99 21



@fernandolira74





# Agenda

---

- Introdução à Programação
- Conceitos fundamentais
- Algoritmos e seus elementos



# Objetivos

- Saber o que é um algoritmo
- Reconhecer os principais elementos de um algoritmo
- Reconhecer conceitos fundamentais associados à programação

---

1

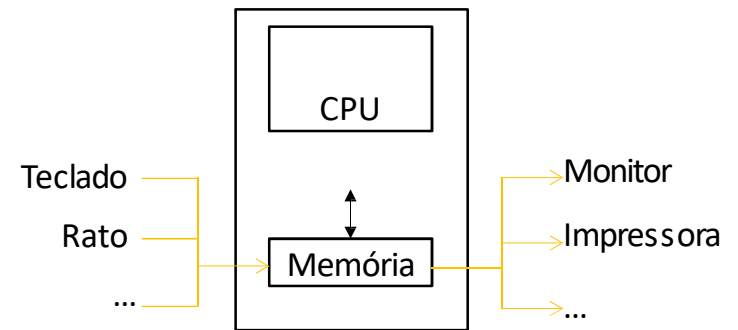
# Introdução

---

Conceitos Fundamentais

## 🗨️ – O que é um computador?

- Dispositivo elétrico/eletrônico que manipula dados
  - Recebe inputs através de rato, teclado, placa de rede ou muitas outras fontes
  - Pode ter em conta esses inputs e dados em memória para efetuar algum processamento/transformação
  - Devolve um output para um monitor, placa de rede, impressora, etc.
- Dados são manipulados de forma automática mas de acordo com um programa
- Output para um mesmo programa pode ser diferente consoante o input



## 🗨️ – Programa

- Conjunto de instruções que implementam uma tarefa específica
- É executado por um computador, sequencialmente, instrução a instrução
- É escrito por um **programador**, numa **linguagem de programação** específica, que é depois transformado em código máquina
  - Um computador apenas consegue “ler” instruções em código máquina
- Podem ser expressos sob a forma de um algoritmo

```
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
    return 0;
}
```

```
01010100000010100100101000101
10100001010100010010001010000
01000110001000001010000101001
00110100010101010110011000101
00000101000001001000100101010
10000101011000010011001011100
01101001001001001010010101010
```

8



## 🗨️ – Programar

---

- 🟡 Arte (ou ciência?) de escrever programas de computador para resolver um determinado problema

1. Abstrair o problema da sua descrição
2. Gerar sub-problemas
3. Transformar sub-problemas em sub-soluções
4. Integrar as sub-soluções num programa funcional
5. Avaliar/Testar (e repetir)

Perceber/Interpretar

Planejar

Implementar



## 🗨 - Linguagem de programação

- Uma linguagem de programação é utilizada para criar programas que implementam algoritmos específicos
- Como qualquer outra linguagem, tem
  - Um conjunto de palavras válidas (instruções)
  - Uma sintaxe (conjunto de regras que define como a linguagem deve ser utilizada)

```
14 #include <stdio.h>
15 #include <stdlib.h>
16
17 void imprimeLista(int lista[], int tam)
18 {
19     int i;
20     for (i=0; i<tam; i++)
21     {
22         printf("Posição: %d | Valor: %d\n", i, lista[i]);
23     }
24     printf("Fim da lista.");
25 }
26
27
28 int main(int argc, char** argv)
29 {
30     int lista[] = {3,5,3,1,2};
31
32     imprimeLista(lista, 5);
33
34     return (0);
35 }
```

## 🗨 – Algoritmos

- É uma sequência de passos para a realização de uma tarefa ou a resolução de um problema
- “Exemplos”
  - Receita num livro de culinária
  - Guia de reparação de uma oficina
  - ...
- Na escrita de um algoritmo é necessário identificar
  - Dados de entrada
  - Resultados de saída esperados
  - Processo pelo qual se obtém os resultados esperados

### INGREDIENTES

1,5 kg amêijoas depuradas  
1 cebola  
6 dentes de alho  
4 c. sopa azeite  
1 molho de coentros  
3 c. sopa vinho branco  
1 limão



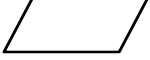
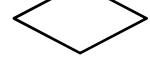

### PREPARAÇÃO

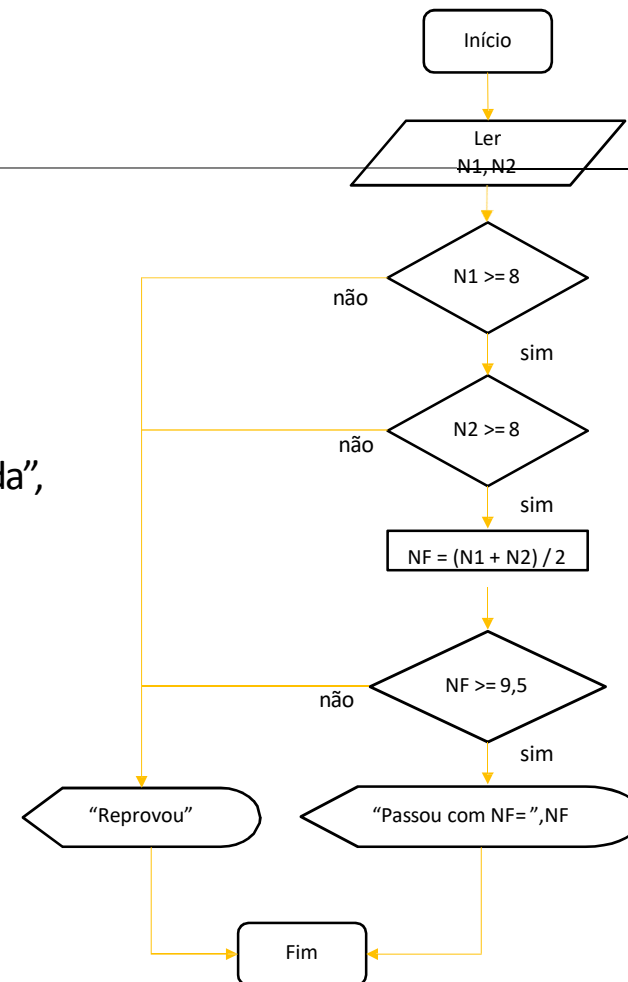
1. Certifique-se de que as amêijoas estão bem depuradas, lave-as em água corrente e deixe-as escorrer dentro de um coador de rede.
2. Leve a cebola e os dentes de alho, finamente picados, ao lume com o azeite até começarem a alourar.
3. Adicione os pés dos coentros picados, incorpore as amêijoas e mexa.
4. Regue com o vinho branco, tape e cozinhe em lume moderado cerca de 5 minutos ou até as amêijoas abrirem, agitando o tacho de vez em quando para que todas recebam calor por igual.
5. Polvilhe com as folhas dos coentros e regue com o sumo do limão.



## – Algoritmos

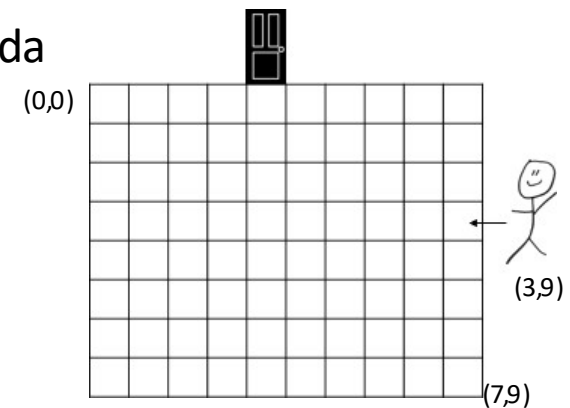
- Um algoritmo pode ser expresso de diferentes formas
  - Fluxograma
  - Pseudocódigo – linguagem informal, “inventada”, próxima da natural (próxs. slides)

	Utilizado no início e fim de um processo
	Utilizado para processamento de um módulo
	Utilizado para operações de entrada e saída de dados
	Utilizado para tomada de decisões
	Utilizado para mostrar informações e resultados



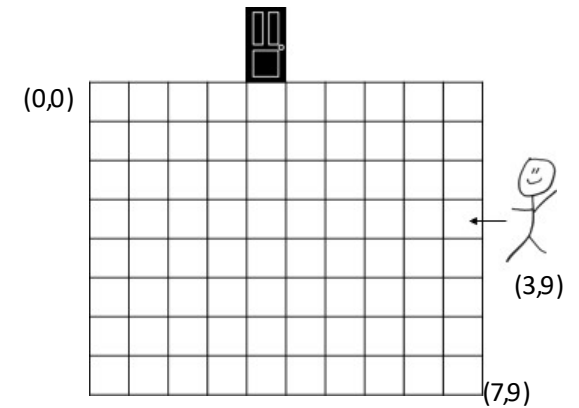
## ② – Algoritmos

- Considere o seguinte problema: O João está na posição (3,9) e quer sair da sala
- Considere uma “linguagem de programação” constituída pelas seguintes instruções
  - Caminhar em frente (1quadrícula)
  - Caminhar para trás (1quadrícula)
  - Rodar 90º esquerda
  - Rodar 90º direita
  - Abrir porta
  - Fechar porta
- Quais os dados de entrada?
- Qual o resultado esperado?
- Qual a sequência de instruções necessária para o João sair da sala?



## ② – Algoritmos

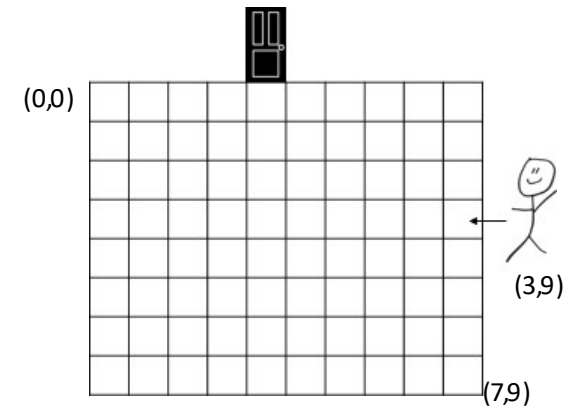
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Rodar 90° direita  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Abrir porta  
Fechar porta  
port



## ② – Algoritmos

- O que acontece à medida que cada linha do algoritmo é executada?
- Qual o resultado final?
- Pode dizer-se que o algoritmo resolve o problema?

Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Rodar 90° direita  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Abrir porta  
Fechar porta  
port

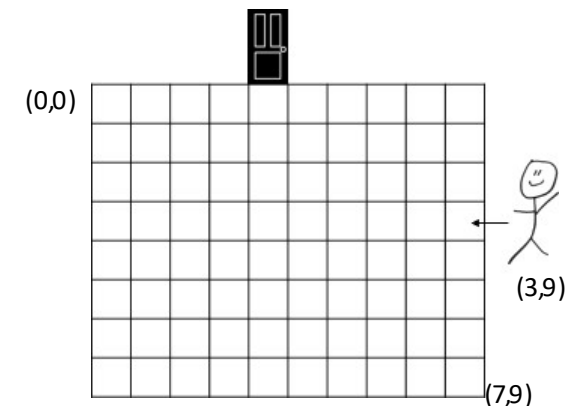




## – Correção (*Correctness*)

- Um algoritmo ou programa diz-se correto quando resolve o problema para o qual foi desenhado
- Um programa deve ser **testado** em diferentes cenários ou casos de uso
  - Há programas que resolvem apenas uma parte do problema ou que falham em determinadas condições
- O algoritmo à direita, resolve o problema?

Caminhar em frente  
Rodar 90° direita  
Caminhar em frente  
Rodar 90° direita  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Abrir porta  
Caminhar em frente  
Fechar porta





## ② – Outra linguagem

---

- Imaginemos agora que, em vez da linguagem apresentada, tínhamos ao dispor uma linguagem que nos permitia controlar cada músculo do João individualmente
  - Seria mais fácil ou mais difícil resolver o problema?
  - Teríamos maior ou menor controle sobre o João?
  - Implicaria maior ou menor conhecimento sobre o sistema para o qual estamos a programar? (O João)
  - O programa/algoritmo seria maior ou menor?
- E se tivéssemos que programar ao nível do cérebro e, controlando as sinapses, decidir que sinais elétricos enviar aos músculos das pernas e mãos, para assim controlar o João?



## – Alto nível vs. Baixo nível

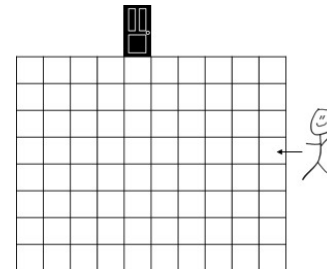
---

- As linguagens são frequentemente caracterizadas como sendo de alto nível ou baixo nível
- Uma linguagem de baixo nível
  - Está próxima do hardware, implica conhecimento do sistema
  - É geralmente mais eficiente
  - Permite maior controlo
- Uma linguagem de alto nível
  - É mais abstrata, mais próxima da linguagem natural
  - Permite fazer mais com menos código
  - Tende a ser mais fácil de aprender
- Não há uma linguagem melhor ou pior, tudo depende do problema a resolver

## 🗨️ – Múltiplas Soluções

- Os dois algoritmos à direita resolvem o problema, no entanto são diferentes
- Existem muitas soluções diferentes para o mesmo problema
  - Cada programador desenvolve mecanismos mentais e técnicas de resolução de problemas próprios
  - Algumas soluções são melhores que outras

Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Rodar 90° direita  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Abrir porta  
Caminhar em frente  
Fechar porta



Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Caminhar para trás  
Caminhar para trás  
Rodar 90° direita  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Abrir porta  
Caminhar em frente  
Fechar porta



## – Complexidade

---

- Os problemas (e a sua resolução) podem ser classificados de acordo com a sua dificuldade/complexidade
- Um problema (ou um programa) mais complexo vai requerer mais recursos para ser resolvido
  - No exemplo anterior, a solução da direita tem 4 instruções a mais
- Um programador deve ter como preocupação implementar programas que façam uma **gestão adequada da memória** e do **processamento** necessário para executar o programa

## 🗨️ – Condições

---

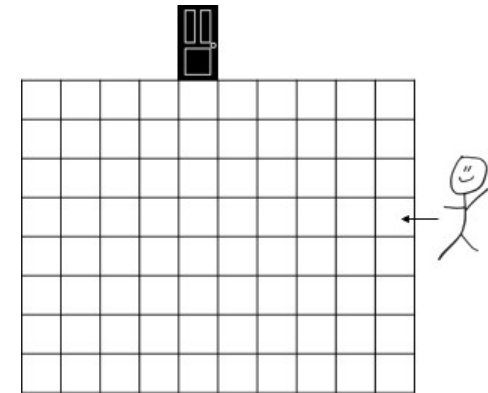
- Por vezes um programa (ou nós próprios) executa(mos) instruções diferentes consoante uma determinada condição é verdadeira ou não
  - “Se está a chover, levo casaco”
  - “Se está a chover, levo casaco, senão levo camisola”
  - “Se está a chover, levo casaco, senão, se está sol, levo t-shirt, senão levo camisola”
- No exemplo do João, poderia dar-se o caso de a porta já estar aberta quando ele lá chega
- Que alterações fazer ao algoritmo?



## – Condições

- Estruturas de controlo condicional como o “se/senão” permitem escolher **um** de vários caminhos de código, segundo uma condição
  - Tal como aconteceu no fluxograma apresentado atrás
- Condições são expressões que apenas admitem um valor verdadeiro ou falso

```
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Rodar 90° direita  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Se porta_aberta  
|   Caminhar em Frente  
Senão  
|   Abrir Porta  
|   Caminhar em frente  
Fechar porta
```

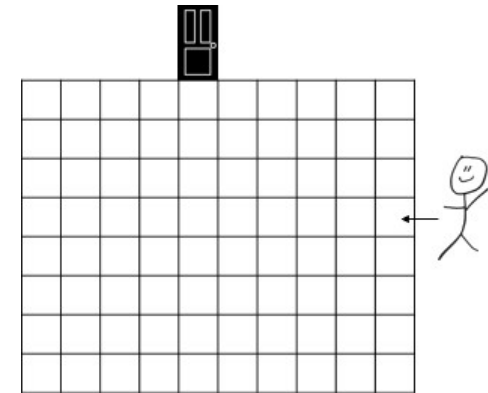




## – Condições

- Estruturas de controlo condicional como o “se/senão” permitem escolher **um** de vários caminhos de código, segundo uma condição
  - Tal como aconteceu no fluxograma apresentado atrás
- Condições são expressões que apenas admitem um valor verdadeiro ou falso

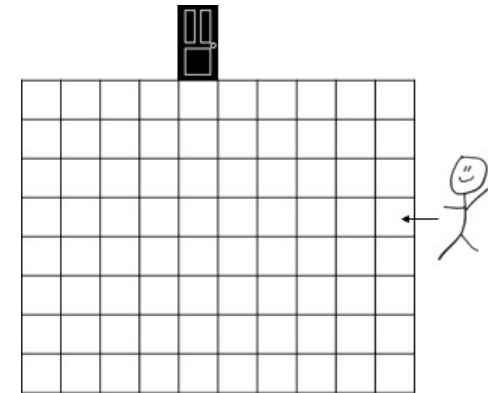
```
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Rodar 90° direita  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Se porta_fechada  
| Abrir Porta  
Caminhar em frente  
Fechar porta
```



## 🗨 – Ciclos

- Há situações em que uma instrução ou conjunto de instruções iguais são repetidas N vezes
- Se a sala tivesse uma dimensão de 1000 x 1000, teríamos que escrever a instrução “Caminhar em frente” centenas de vezes

Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Rodar 90° direita  
Caminhar em frente  
Caminhar em frente  
Caminhar em frente  
Abrir porta  
Caminhar em frente  
Fechar porta





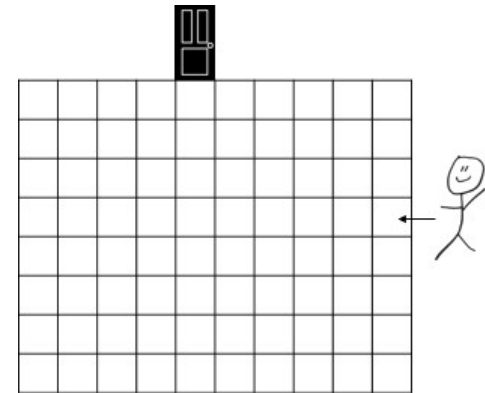


## – Ciclos

- Algumas linguagens de programação têm mecanismos que permitem **repetir uma instrução** ou conjunto de instruções
- Isto permite reduzir significativamente a quantidade de código produzida (entre outras vantagens a abordar posteriormente)
- Existem diferentes tipos de ciclos

```
Repetir 5 vezes  
| Caminhar em frente  
Rodar 90° direita  
Repetir 3 vezes  
| Caminhar em frente  
Abrir porta  
Caminhar em frente  
Fechar porta
```

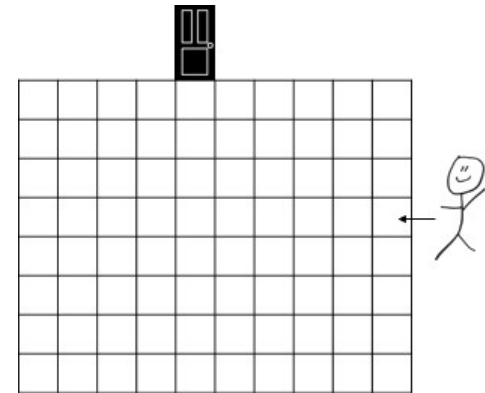
```
Enquanto passos < 5  
| Caminhar em frente  
Rodar 90° direita  
Enquanto passos < 8  
| Caminhar em frente  
Abrir porta  
Caminhar em frente  
Fechar porta
```



## – Ciclos

- Esta solução implica que o João tenha a capacidade de **contar passos**
  - Em cada momento o João tem que saber quantos passos já deu
- Algures, na memória do João, existe um espaço destinado a **guardar o número** de passos dados
  - Este número é atualizado a cada passo

```
Enquanto passos < 5  
  Caminhar em frente  
Rodar 90° direita  
Enquanto passos < 8  
  Caminhar em frente  
Abrir porta  
Caminhar em frente  
Fechar porta
```





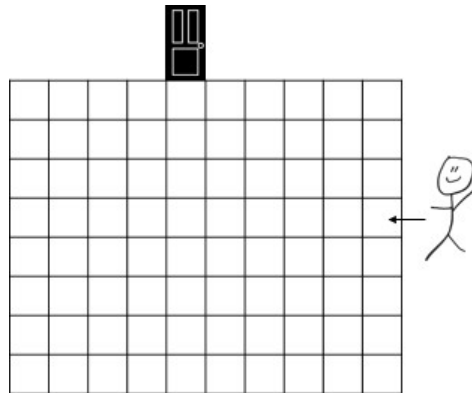
## – Variável

---

- Uma variável é um espaço em memória no qual é possível guardar um valor (numérico ou outro)
- A memória é constituída por muitos destes “espaços” que, enquanto programadores, podemos utilizar
- Variáveis são declaradas sempre que for necessário guardar um determinado valor ou conjunto de valores
  - Há linguagens de programação (tal como o C) em que é necessário definir o tipo de uma variável no momento da sua declaração
- Uma variável é identificada por um nome (passos, no exemplo)
- Uma variável pode ser lida ou escrita pelo programa, durante a sua execução

## ❓ – Variável

- Qual o valor da variável passos ao longo do programa?
  - Admita que quando o programa começa João não tinha dado qualquer passo
- Quando o programa termina, qual o valor da variável passos?



```
Enquanto passos < 5
    Caminhar em frente
Rodar 90° direita
Enquanto passos < 8
    Caminhar em frente
Abrir porta
Caminhar em frente
Fechar porta
```



## – Traçagem (*DryRun*)

---

- O que foi feito no slide anterior, ainda que de forma simples, é uma **traçagem**
  - Testar o algoritmo numa determinada situação, anotando o valor de cada variável à medida que cada instrução é executada
  - Permite perceber como evolui o **estado** do programa
- Geralmente é feita uma tabela
  - Em cada coluna está uma variável
  - Existe uma linha por cada instrução executada
  - Em cada célula está o valor de cada variável, após a execução da instrução dessa linha



## – Estado

- Quando um programa está em execução, tem dados guardados em variáveis que estão em memória
- O conteúdo de todas estas localizações de memória constituem o estado do programa
- Quando um programa não funciona como esperado, é fundamental saber o seu estado no momento do erro para perceber em que situação e por que razão o erro ocorre
- Existem ferramentas que ajudam na inspeção do estado do programa

```
14 #include <stdio.h>
15 #include <stdlib.h>
16
17 void imprimeLista(int lista[], int tam)
18 {
19     int i;
20     for (i=0; i<tam; i++)
21     {
22         printf("Posição: %d | Valor: %d\n", i, lista[i]);
23     }
24     printf("Fim da lista.");
25 }
```

imprimeLista

Variables	Call Stack	Breakpoints	Output												
<table border="1"><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td colspan="2">&lt;Enter new watch&gt;</td></tr><tr><td>i</td><td>2</td></tr><tr><td>lista</td><td>0xffffcb40</td></tr><tr><td>*lista</td><td>3</td></tr><tr><td>tam</td><td>5</td></tr></tbody></table>		Name	Value	<Enter new watch>		i	2	lista	0xffffcb40	*lista	3	tam	5		
Name	Value														
<Enter new watch>															
i	2														
lista	0xffffcb40														
*lista	3														
tam	5														



# Recapitulando

- O que é um **algoritmo**?
- Quais os principais elementos de um algoritmo?
- Como corre um programa?
- Quais os passos para programar?

*The psychological profiling [of a programmer] is mostly the ability to shift levels of abstraction, from low level to high level. To see something in the small and to see something in the large.*

Donald Knuth

“





---

1

## Conceitos fundamentais



## Em suma:

### Linguagens + Algoritmos

Linguagem: sistema de comunicação através de signos

Algoritmo: sequência finita de instruções para a realização de uma tarefa

### Computador digital

Máquina programável que processa dados, eletrônica e automaticamente, para a realização de tarefas

### Programação

Arte de definir um conjunto de instruções que implementam, numa *máquina computacional*, a execução de tarefas

## Laboratório de programação

Linguagens + Algoritmos

O conhecimento



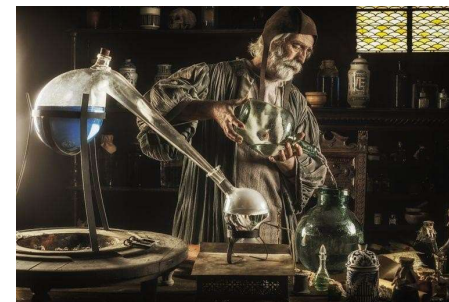
Computador digital

O instrumento



Programação

A arte





Linguagem de programação

## Linguagem de programação

Para que serve uma linguagem?

- Para comunicar entre duas ou mais entidades

Linguagem natural

- Descreve ideias, ações, sentimentos, emoções, etc.
- Apresenta um vocabulário rico e regras gramaticais complexas
- É muitas vezes ambígua

## Linguagem de programação

Linguagem de programação:

- Descreve operações a serem executadas por um computador
- Apresenta um vocabulário limitado e regras gramaticais simples
- É sempre clara e concisa

Permite que um programador especifique precisamente sobre que dados um computador vai atuar, como serão armazenados ou transmitidos estes dados e que ações devem ser tomadas em diversas circunstâncias bem definidas (por outras palavras, expressa um algoritmo que possa ser executado por um computador)

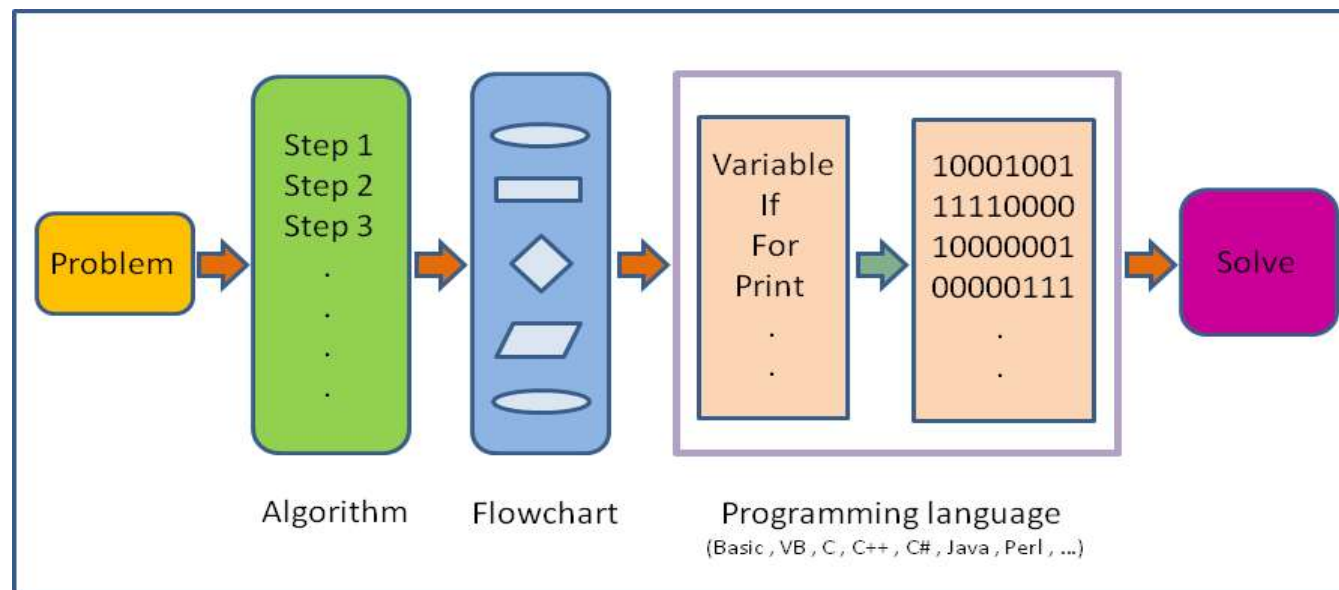
## Linguagem de programação

Linguagem caracteriza-se por possuir:

- Semântica ou terminologia – conjunto de termos, palavras ou sinais que assumem determinados significados para o processador
- Sintaxe ou conjunto de regras – estipulam o modo correto de utilizar e estruturar os termos da linguagem para formular instruções válidas para a máquina

## 🗨️ O que é um computador digital?

Linguagem + Algoritmo + Computador + Processo = Programação ?



Fonte: <https://en.wikiversity.org/wiki/File:Programming.png>

25





## Programação

## O processo de programação

Programar é a arte (ou ciência?) de escrever programas para resolver um determinado problema.

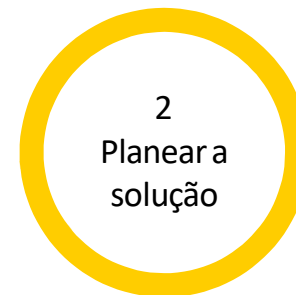
Para os resolver, de um modo organizado e sistematizado, podemos ter em consideração a seguinte abordagem:

1. Identificar o problema
2. Planear a solução
3. Escrever o programa
4. Verificar a solução

## 🗨 O processo de programação



- O que é pretendido fazer?
  - Requisitos
  - Especificação

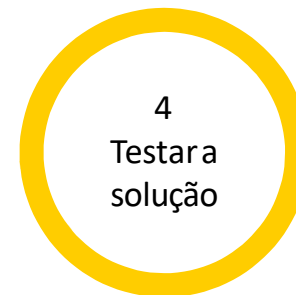


- Como será feito?

## 🗨️ O processo de programação

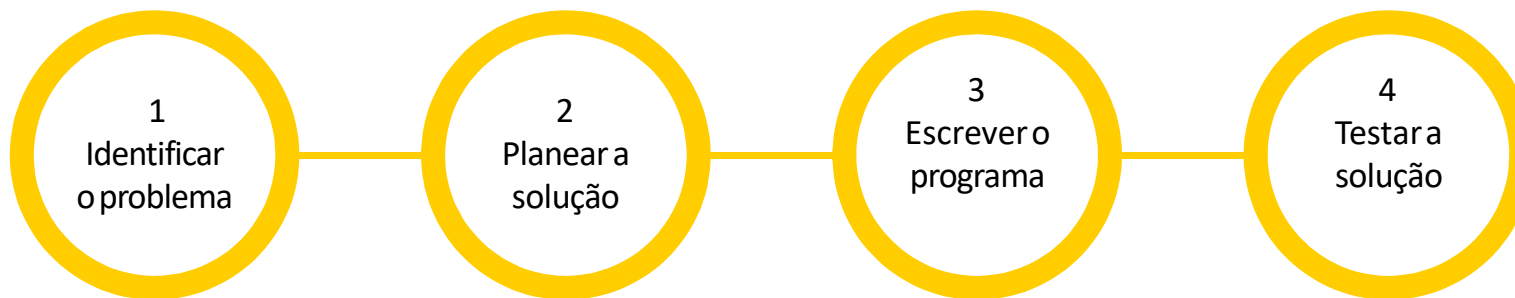


- "Instruir" o computador
  - Código
  - Compilação
  - *Debugging*



- Testar para verificar se o programa faz o pretendido

## 🗨️ O processo de programação



Destes passos, apenas o terceiro é habitualmente denominado de “programação”, mas como verá mais tarde, é provavelmente o passo menos importante do processo.

# EXERCÍCIOS

## Desafios

Dado um valor inteiro de  $x$  e  $y$ , calcular a soma dos dois valores.

Dado o valor da precipitação em polegadas, fazer a conversão desse valor para milímetros.

1 pol = 2.540 cm

Dada uma temperatura em graus Celsius, converter esse valor para graus Fahrenheit.

Calcule a área ( $A$ ) e o volume ( $V$ ) de uma esfera, sabendo que

- $A = 4 \times \pi \times r^2$
- $V = \frac{4 \times \pi \times r^3}{3}$

onde  $r$  identifica o raio da esfera.

## Desafios

Apresentar o maior de dois números inteiros  $x$  e  $y$ .

Apresentar o mínimo de três números reais  $x$ ,  $y$  e  $z$ .

Numa empresa o vencimento dos colaboradores é calculado a partir de um vencimento base (VB) e tem em consideração a sua idade (ID), número de filhos (NF) e anos de serviço (AS). O cálculo do salário final é feito de acordo com as seguintes parcelas:

- 1% de VB para cada ano de ID superior a 25;
- 3% por cada ano de serviço até perfazer 5 anos; sendo 5% por cada ano extra;
- 2% por cada filho, considerando o valor mínimo de 2 filhos.

Calcular o valor a receber por cada colaborador da empresa. Tome como exemplo um colaborador cujo  $VB = 900$  e  $ID = 27$ ,  $AS = 1$  e  $NF = 1$ . Para esta situação o valor final a receber será de

$$Vencimento\ Final = VB + 0,01 \times 2 \times VB + 0,03 \times 1 \times VB = 945.$$