

# Resolução de Problemas com o Computador

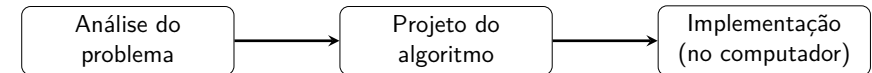
## Algoritmos e Programação de Computadores

Guilherme N. Ramos  
gnramos@unb.br

2018/2



## Análise do Problema



Definição *clara* do que se deseja.

- Qual é a entrada do problema?
- Qual é a saída desejada?
- Como transformar a entrada na saída?
- É necessário realizar algo durante a transformação?
- etc.

## Análise do Problema

### 1 Entendendo o problema.

- O que é desconhecido?
- Quais são os dados conhecidos?
- Estes dados são *suficientes* para resolver a questão?

### 2 Elabore um plano.

- Qual a ligação entre o conhecido e o desconhecido?
- Se não há uma ligação direta entre eles, há algum problema relacionado que pode ser útil na resolução?
- Se sim, este conhecimento é útil para o problema em questão?
- Se não, o problema pode ser apresentado de outra forma?

## Análise do Problema

### 3 Execute o plano.

- Verifique cada passo do plano.
- Está claro que o passo está correto?
- É possível provar que o passo está correto?

### 4 Examine a solução.

- É possível verificar o resultado?
- É possível obter o resultado de outra forma?
- É possível utilizar a solução ou o método em outro problema?

## Projeto Do Algoritmo

### Algoritmo

Sequência finita de instruções para executar uma tarefa.

- Bem definidas e não ambíguas.
- Executáveis com uma quantidade de esforço finita.
- Executáveis em um período de tempo finito.

#### Bolo simples

```
1 tigela ← ingredientes
2 bater(tigela) durante(5 min)
3 untar(forma)
4 forma ← conteúdo(tigela)
5 colocar(forma, fogão)
6 esperar(40 min)
```

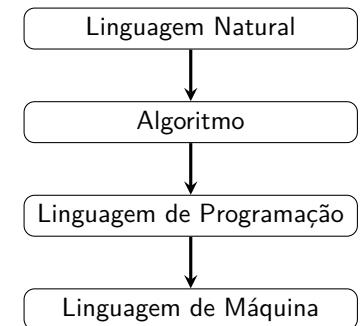
#### Soma 0 a 100

```
1 total ← 0
2 total ← total + 1
3 total ← total + 2
...
100 total ← total + 99
101 total ← total + 100
```

## Projeto Do Algoritmo

Um algoritmo descreve um padrão de comportamento com:

- instruções sequenciais;
- bifurcações;
- repetição.



*É conceito central da programação e da ciência da computação.*

## Projeto Do Algoritmo

*“Corretude é claramente a principal qualidade. Se um sistema não faz o que deveria fazer, então todas as outras coisas a seu respeito não têm importância.”*

**Bertrand Meyer**

*“Faça funcionar antes de fazer funcionar rápido.”*

**Bruce Whiteside**

## Representação de Algoritmos

Existem diversas formas de representar de um algoritmo.

- Descrição narrativa
- Diagramas (fluxograma)
- Pseudocódigo
- Linguagem de programação
- Linguagem de máquina
- etc.

Não existe um consenso de qual é a melhor forma, mas o algoritmo deve ser representado da forma mais clara possível.

## Representação de Algoritmos

### Exercício

Faça a descrição narrativa do algoritmo para trocar a lâmpada queimada de um abajur.

## Representação de Algoritmos

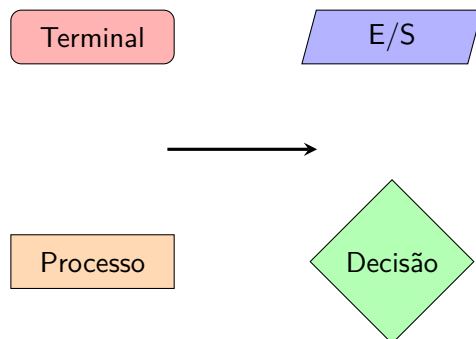
### Exercício

Faça a descrição narrativa do algoritmo para ler a idade de uma pessoa (em anos, meses e dias), e escrever a quantidade de horas vividas.

## Representação de Algoritmos

### Fluxograma

O algoritmo é expresso graficamente com auxílio de formas geométricas.



## Representação de Algoritmos

### Exercício

Faça o fluxograma do algoritmo para ler a idade de uma pessoa (em anos, meses e dias), e escrever a quantidade de horas vividas.

## Representação de Algoritmos

### Exercício

Monte um fluxograma para um algoritmo que leia os três coeficientes de uma equação de segundo grau e diga se as raízes são reais ou complexas, e se são iguais.

$$ax^2 + bx + c = 0 \Rightarrow r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

## Representação de Algoritmos

### Pseudocódigo

O algoritmo é expresso com estruturas de linguagem de programação, mas de forma a facilitar a compreensão humana (e não da máquina).

#### Exemplo de Pseudocódigo

```
1 Leia (nome)
2 Escreva ("Olá ", nome)
3 Se Entende (PSEUDOCÓDIGO) Então
4     Escreva ("Ótimo! Vamos para o próximo assunto.")
5 Senão
6     Escreva ("Vá estudar.")
7 Enquanto Não Entende (PSEUDOCÓDIGO)
8     Estude (PSEUDOCÓDIGO)
9 FimEnquanto
10 FimSe
```

## Representação de Algoritmos

### Pseudocódigo

A construção do algoritmo faz parte do planejamento e execução do plano, conforme as seguintes etapas:

- 1 Definição dos dados a serem manipulados:
  - Quais informações serão necessárias? (variáveis/constantes)
  - Quais os tipos de cada?
- 2 Obtenção destes dados (entradas).
- 3 Definição da sequência de execução dos processos.
- 4 Exibição dos resultados (saídas).

## Representação de Algoritmos

### Pseudocódigo

A *precedência* define a ordem em que as operações são executadas quando não explicitado por parênteses. Em alguns casos, a indefinição de precedência pode levar a **comportamento não-definido**.

$$1 + 2 - 3 * 20/5 + 2^3 = -1$$

$$(1 + 2) - (3 * (20/5)) + (2^3) = -1$$

Na dúvida, *sempre use parênteses para explicitar a ordem das operações*.

## Representação de Algoritmos

### Pseudocódigo

Nesta disciplina, **todo** pseudocódigo deve seguir a seguinte estrutura:

```
1 Algoritmo NomeDoAlgoritmo
2 Constantes
3     // Definir constantes (se houver).
4
5 Variáveis
6     // Definir variáveis (se houver).
7
8 Função
9     // Definir funções (se houver).
10
11 Início
12     // Definir instruções.
13 Fim
```

## Representação de Algoritmos

### Pseudocódigo

```
1 Algoritmo Le_2_Numeros_e_Mostra_Divisao
2 /* O nome indica claramente o que faz! */
3
4 Variáveis
5     numerador, denominador : real
6     /* Nomes indicam claramente para que servem! */
7
8 Início
9     Escreva("Digite o numerador: ")
10    Leia(numerador)
11    Escreva("Digite o denominador: ")
12    Leia(denominador)
13    Escreva("A divisão é: ", numerador / denominador)
14 Fim
```

## Representação de Algoritmos

### Exercício

Faça a descrição narrativa do algoritmo para ler os catetos e calcular hipotenusa de um triângulo retângulo.

## Representação de Algoritmos

### Exercício

Monte um fluxograma para um algoritmo que leia as 3 notas de um aluno e calcule a média final deste aluno. Considerar que a média é ponderada e que os pesos das notas são: 1, 3 e 4, respectivamente.

Defina o pseudocódigo do algoritmo gerado.

## Representação de Algoritmos

### Exercício

Defina um algoritmo em pseudocódigo que leia a altura (em metros) e a massa (em kg) de uma pessoa e indique seu *Índice de Massa Corpórea*.

$$IMC = \frac{massa}{altura^2}$$

$18,5 > IMC$	Abaixo do peso ideal.
$18,5 \leq IMC < 25$	Peso normal
$IMC \geq 25$	Sobrepeso.

Defina o fluxograma do algoritmo gerado.

## Representação de Algoritmos

### Exercício

Calcule o valor líquido que um trabalhador recebe conhecendo o número de horas trabalhadas, a tarifa horária e a alíquota de impostos.

## Implementação

*“Conhecimento não tem valor a não ser que seja posto em prática.”*

**Anton Chekhov**

Um algoritmo computacional será executado por um computador de programa armazenado, é uma *sequência de instruções* que vai *manipular dados*.

**Instruções:** comandos que determinam a forma pela qual os dados devem ser tratados.

**Dados:** informações recolhidas/fornecidas por diversos meios e que serão processadas pelo computador através das instruções.

## Codificação

Codificação é a escrita do algoritmo em uma *linguagem de programação*.

### Exemplo de Código em Python

```
1 nome = input('Digite seu nome:')
2 print('Olá', nome)
3
4 if entende(PYTHON):
5     print('Ótimo!')
6 else:
7     print('Vá estudar.')
8 while not entende(PYTHON):
9     estude(PYTHON)
10    pratique(PYTHON)
```

### Exemplo de Código em C

```
1 printf("Digite seu nome:");
2 scanf("%s", nome);
3 printf("Olá %s.\n", nome);
4
5 if(entende(C))
6     printf("Ótimo!\n");
7 else {
8     printf("Vá estudar.\n");
9     while(!entende(C)) {
10         estude(C);
11         pratique(C);
12     }
13 }
```

## Codificação

Bjarne Stroustrup

“Programação”:

- dizer exatamente o que fazer a um idiota extremamente rápido
- um plano para resolver um problema no computador
- especificar a ordem de execução de um programa

### Programação

Especificar a estrutura e comportamento de um programa, e testar que o programa executa corretamente a tarefa com desempenho adequado.

## Codificação

Bjarne Stroustrup

Programar é, fundamentalmente, *simples*: basta dizer à máquina o que fazer.

Por que programação é tida como “difícil”?

Queremos que a máquina faça algo complicado em um mundo mais complexo que parece ser e sem ter certeza das implicações do que se quer... e computadores são bestas estúpidas, cheias de caprichos e que não perdoam qualquer deslize.

## Codificação

“Programação é entendimento.”

Bjarne Stroustrup

- Quando você é capaz de programar uma tarefa, você a entende.
- Ao programar, você gasta um tempo significativo tentando entender a tarefa que quer automatizar.
- Programação é parte *prática*, parte *teoria*.
  - Se você é só prática, faz “gatos”.
  - Se você é só teoria, faz “brinquedos”.

## Linguagem de Programação C

Cada linguagem de programação possui suas próprias regras e formatos, que devem ser respeitados durante a codificação.

Mas por que?

- *Desempenho*
- *Gerenciamento de memória*
- “*Linguagem de baixo nível*”
- Portabilidade
- Maturidade
- Perl, PHP, Python, R, Matlab, Mathematica, etc. são escritos em C.

Sugestão: aprendam C  $\Rightarrow$  Python/R  $\Rightarrow$  LISP/Haskell

## Linguagem de Programação C

O algoritmo é expresso conforme a sintaxe da linguagem.

```
1      #include <stdio.h>
2
3      int main() {
4          char nome[100];
5
6          scanf("%s", nome);
7          printf("Olá %s.", nome);
8
9          if(entende(LINGUAGEM_C))
10             printf("Ótimo! Vamos para o próximo
assunto.");
11         else {
12             printf("Vá estudar.");
13             while(!entende(LINGUAGEM_C))
14                 estude(LINGUAGEM_C);
15         }
16         return 0;
17     }
18
```

## Linguagem de Programação C

Expressões lógicas resultam em um valor lógico booleano.

		Conjunção	Disjunção	Negação	
x	y	x && y	x    y	!y	(!x) && (x    y)
V	V				
V	F				
F	V				
F	F				

## Linguagem de Programação C

Nesta disciplina, **todo** código deve seguir a seguinte estrutura:

```
1 /* Identificação Disciplina/Aluno/Programa */
2
3 /* Incluir bibliotecas (se necessário). */
4
5 /* Definir funções (se houver). */
6
7 int main() {
8     /* Definir constantes (se houver). */
9
10    /* Definir variáveis (se houver). */
11
12    /* Definir instruções. */
13 }
```

## Linguagem de Programação C

00-hello\_world.c

```
1 /**      @file: 00-hello_world.c
2  *      @author: Guilherme N. Ramos (gnramos@unb.br)
3  * @disciplina: Algoritmos e Programação de Computadores
4  *
5  * Exemplo tradicional de programação, apenas escreve uma
6  * mensagem na tela. */
7
8 #include <stdio.h>
9
10 int main() {
11     printf("Hello World!\n");
12     return 0;
13 }
```

```
$ gcc -ansi -Wall 00-hello_world.c -o hello
```



## Linguagem de Programação Python

Cada linguagem de programação possui suas próprias regras e formatos, que devem ser respeitados durante a codificação.

Mas por que?

- *Utilidade*
- *“Linguagem de alto nível”*

## Linguagem de Programação Python

O algoritmo é expresso conforme a sintaxe da linguagem.

```
1 nome = input()
2
3 print('Olá', nome)
4
5 if entende(LINGUAGEM_PYTHON):
6     print('Ótimo! Vamos para o próximo assunto.')
7 else:
8     print('Vá estudar.')
9 while not entende(LINGUAGEM_PYTHON):
10     estude(LINGUAGEM_PYTHON)
```

**Tabulação!**

## Linguagem de Programação Python

PEP 8

<https://wiki.python.org.br/GuiaDeEstilo>

### PEP 8 -- Style Guide for Python Code

PEP:	8
Title:	Style Guide for Python Code
Author:	Guido van Rossum <guido at python.org>, Barry Warsaw <barry at python.org>, Nick Coghlan <ncoghlan at gmail.com>
Status:	Active
Type:	Process
Created:	05-Jul-2001
Post-History:	05-Jul-2001, 01-Aug-2013

## Linguagem de Programação Python

Expressões lógicas resultam em um valor lógico booleano.

		Conjunção	Disjunção	Negação	
x	y	x and y	x or y	not y	(not x) and (x or y)
V	V				
V	F				
F	V				
F	F				

# Linguagem de Programação Python

Nesta disciplina, **todo** código deve seguir a seguinte estrutura:

```
1 # Identificação Disciplina/Aluno/Programa
2
3 # Importar módulos (se necessário).
4
5 # Definir funções (se houver).
6
7 # Definir constantes (se houver).
8
9 # Definir variáveis (se houver).
10
11 # Definir instruções.
```

# Linguagem de Programação Python

00-hello\_world.py

```
1 # -*- coding: utf-8 -*-
2 # @package: 00-hello_world.py
3 # @author: Guilherme N. Ramos (gnramos@unb.br)
4 # @disciplina: Algoritmos e Programação de Computadores
5 #
6 # Exemplo tradicional de programação, apenas escreve
7 # uma mensagem na tela.
8
9 print('Hello World!')
```

```
$ python 00-hello_world.py
```