



UNINASSAU



INTRODUÇÃO DE ALGORITMOS E ESTRUTURAS DE DADOS



UNINASSAU



JOÃO FERREIRA DA SILVA JÚNIOR

- 1 Professor
- 2 Plano de Ensino
 - Conteúdo Programático
 - Bibliografia Sugerida
- 3 Introdução a Algoritmos
 - Definição de Algoritmo
 - Finalidades e Aplicações
 - Técnica de Abordagem
 - Representação do Algoritmo
 - Fluxograma
 - Pseudocódigo
- 4 Estruturas de Dados
 - Variáveis e Constantes
 - Tipo de Dado Primitivo
 - Definição
 - Lógico ou Booleano
 - Numérico



- Cadeia de Caracteres
- Tipo de Dado Não Primitivo
 - Definição
 - Vetor ou Array
 - Matriz

5 Aplicações Práticas

- Exemplo - Variáveis
- Exemplo - Constantes
- Exemplo - Booleano
- Exemplo - Cadeia de Caracteres
- Exemplo - Vetor
- Exemplo - Matriz

6 O que há além?

- Estruturas Lineares
- Estruturas Complexas

7 Agradecimento



8 Referências





João Ferreira da Silva Júnior

- <https://joaoferreirape.wordpress.com>
- <http://lattes.cnpq.br/8904695743376784>
- Formado em Análise e Desenvolvimento de Sistemas em 2013 (UNOPAR).

- Mestre em Engenharia Elétrica, título obtido em 2017 (UFPE).
- Doutorando em Engenharia Elétrica, previsão para 07/2022 (UFPE).
- Servidor Público na Compesa atuando como Analista de Tecnologia da Informação (Analista de Negócios, Administrador de Banco de Dados, Projetos de Inovação).
- Professor da UNINASSAU nos cursos de Análise e Desenvolvimento de Sistemas, Ciência da Computação Engenharia da Computação, Redes de Computadores e Sistemas de Informação.





Plano de Ensino

Conteúdo Programático

- Definição de algoritmos, tipos, aplicações e exemplos.
- Abordagem, pseudocódigo e fluxograma.
- Variáveis, estrutura de dados, tipos de dados primitivos e não primitivos.
- Exemplos de tipos de dados em linguagem de programação.
- O que há mais?



Bibliografia Básica

- GERSTING, J. L. Fundamentos Matemáticos para a Ciência da Computação. 7o ed. LTC, 2016. (GERSTING, 2016)
- COMEN, T. H. ALGORITMOS - TEORIA E PRÁTICA. 3o ed. GEN LTC, 2012. (COMEN, 2012)
- MEDINA, M.; FERTIG, C. Algoritmos e Programação - Teoria e Prática. 1o ed. Novatec Editora, 2005. (MEDINA; FERTIG, 2005)



Bibliografia Complementar

- ENGELBRECHT, A.; NAKAMITI, G.; JUNIOR, D. Algoritmos e Programação de Computadores. 2o ed. GEN LTC, 2019. (ENGELBRECHT; NAKAMITI; JUNIOR, 2019)
- LEVITIN, A. Introduction to the design and analysis of algorithms. 3o ed. Pearson Education, 2011. (LEVITIN, 2011)
- RAWLINS, G. J. E. Compared to What? An Introduction to the Analysis of Algorithms. W. H. Freeman, 1991. (RAWLINS, 1991)



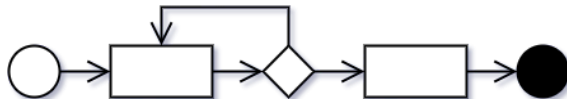


Introdução a Algoritmos

Definição de Algoritmo — I

Mas afinal, o que é algoritmo?

... é qualquer procedimento bem definido que toma algum valor ou conjunto de valores como entrada e produz algum valor ou conjunto de valores como saída. Portanto, um algoritmo é uma sequência de etapas que transforma a entrada na saída ... (COMEN, 2012)



Definição de Algoritmo — II

Partes formais de um algoritmo

- Entrada: pode ser uma sequência de n números (a_1, a_2, \dots, a_n) .
- Processamento: os passos que realizam a tarefa desejada.
- Saída: uma permutação, ou reordenação $(a'_1, a'_2, \dots, a'_n)$ da sequência de entrada, tal que $(a'_1 \leq a'_2 \leq \dots \leq a'_n)$.



Definição de Algoritmo — III

Restrições para um algoritmo

Mas, de modo geral, um algoritmo conterá:

- uma sequência finita de instruções; que,
- não apresentam ambiguidade; por serem,
- bem definidas e elementares ou unitárias; e, por fim,
- que sejam executadas em um intervalo finito de tempo.



Definição de Algoritmo — IV

A definição mais importante

👉 Qual problema o algoritmo se propõe a resolver? 👈

- Deve estar bem definido.
- Deve haver de fato alguma solução.
- A sequência passos deve ser conhecida.
- Sem o problema não há o algoritmo.



Definição de Algoritmo — V

Construção de um algoritmo

Refere-se ao conjunto de processos para:

- identificar e analisar um problema, a fim de
- propor uma solução, através
- da definição de uma sequência de atividades, e
- sua posterior implementação.



Finalidades e Aplicações

Problemas resolvidos por algoritmos

- Ordenação, classificação e filtragem.
- Melhor caminho e busca de rotas.
- Simulação, avaliação de desempenho e otimização.
- ... uma infinidade de aplicações ...
- A famosa receita de bolo 🍰



Técnica de Abordagem — I

Aproximação *top-down*

- Problema bem definido.
- Característica cíclica de iterações.
- Parte do geral para o específico.
- A cada ciclo estende-se mais a definição.
- Objetiva aproximar-se dos detalhes para resolução do problema.



Técnica de Abordagem — II

e.g.: Substituir uma lâmpada queimada

Iteração 1

- 1 Buscar e posicionar a escada.
- 2 Buscar a lâmpada nova.
- 3 Retirar a lâmpada queimada.
- 4 Inserir a lâmpada nova.



Técnica de Abordagem — III

Iteração 2

① Buscar e posicionar a escada.

- ① Verificar se é seguro subir a escada para então poder subir até ficar próximo da lâmpada.

② Retirar a lâmpada queimada.

- ① Verificar as características da lâmpada queimada.

③ Buscar a lâmpada nova.

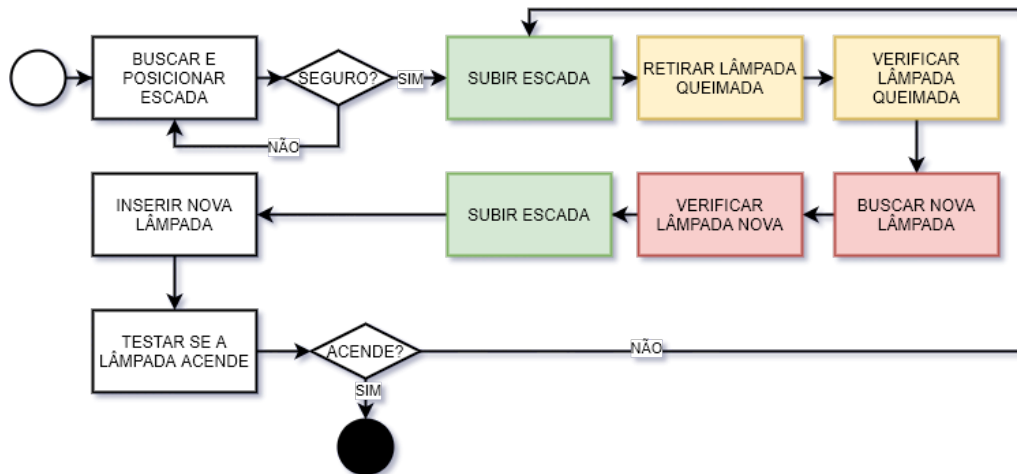
- ① Verificar se a lâmpada nova é igual a queimada.
- ② Subir a escada até ficar próximo do bocal.

④ Inserir nova lâmpada.

- ① Descer da escada e testar se a lâmpada nova acende.



Fluxograma



Pseudocódigo

Calcular a área de um retângulo

```
algoritmo "calculaAreaRetangulo"

var
    altura, comprimento: Inteiro
inicio
    escreval("Digite a altura do retângulo: ")
    leia(altura)
    escreval("Digite o comprimento do retângulo: ")
    leia(comprimento)
    escreval("A Area do Retangulo vale: ", altura * comprimento)
finalgoritmo
```





Estruturas de Dados

Variáveis e Constantes

- São espaços reservados na memória do computador que armazenam informações utilizadas pelos programas.
- Possuem nome, tipo de dado armazenado, valor e endereço.
- A variável pode ter seu valor modificado.
- A constant, por definição, não tem seu valor modificado.
- Por convenção o nome deverá iniciar sempre com uma letra, não conter espaço nem caractere especial.



Definição

- Tipo de dado básico implementado por todas as linguagens de programação.
- Representa algum valor do mundo real.
- Armazena valor lógico, numérico ou de texto.



Lógico ou Booleano

Variável booleana

- Possui a representação para apenas dois estados, ou valores.
- Representa os valores lógicos binários para verdadeiro ou falso.
- Pode conter o valor inteiro 0 para falso e 1 para verdadeiro.
- Pode conter o valor *false* para falso e *true* para verdadeiro.
- e.g.: `var lampadaQueimada = true;`



Numérico

Variável numérica

- Pode conter dados com valores numéricos inteiros \mathbb{Z} .
- Pode conter dados com valores numéricos reais \mathbb{R} , decimal ou de ponto flutuante.
- Existem vários subtipos de dados numéricos conforme a limitação do tamanho do dado e espaço alocado em memória.
- e.g.: *var inteiro* = -2; *var real* = $\sqrt{(42)}$;



Cadeia de Caracteres

Variável alfanumérica

- Armazena a sequência de um ou mais caracteres.
- Pode conter dados com valores numéricos, textual ou alfabético, caracteres especiais, símbolos ou caracteres de controle.
- Existem vários subtipos de cadeias de caracteres conforme a limitação do tamanho do dado e espaço alocado em memória.
- e.g.: *var rua = "Av. 13 de maio"; var simbolo = ©;*



Definição

- Tipo de dado composto por estrutura de dado primitiva.
- Poderá conter mais de um tipo de dado primitivo ao mesmo tempo.
- Poderá ter tamanho fixo ou variável.
- Comumente utilizam-se os tipo de dado em vetor, matriz ou ainda definido pelo programador.



Vetor ou Array

Variável vetor unidimensional

- Armazena um conjunto de dados de um determinado tipo.
- Contém um nome, ao qual se associa o tipo de dado e ao menos um elemento onde está armazenado o valor.
- Cada elemento do vetor é indexado obrigatoriamente através de um identificador numérico ou índice.
- Algumas linguagens permitem nomes em índices.



Matriz

Variável matriz ou vetor multidimensional

- É um vetor bidimensional, ou vetor de vetores.
- A matriz é composta por linhas, onde se define a dimensão 1, e por colunas, onde se define a dimensão 2.
- A cada associação de linha e coluna tem-se uma célula ou elemento.
- Pode conter mais de um tipo de dado.





Aplicações Práticas

Exemplo - Variáveis — I

Exemplo de variável com tipo inteiro

- Seu valor pode mudar.
- Tamanho depende da arquitetura do processador.
- Exemplo considerando a linguagem C.
- Inteiro com 2 bytes: $[-32.768 \text{ até } +32.767]$ ou $[0 \text{ até } +65.535]$.
- Inteiro com 4 bytes: $[-2.147.483.648 \text{ até } +2.147.483.647]$ ou $[0 \text{ até } +4.294.967.295]$.



Exemplo - Variáveis — II

Exemplo de código escrito em linguagem C

```
// Exemplo com tipo de dado inteiro
// Definir e inicializar a variável
int numero = 0;
// Atribuir valor
numero = 8535;
// Acessar a variável imprimindo seu valor em tela
printf("Imprimindo o valor da variavel: %i", numero);
```



Exemplo - Constantes — I

Exemplo de constante com tipo decimal

- Seu valor nunca poderá mudar.
- Tamanho depende da arquitetura do processador.
- Exemplo considerando a linguagem C.
- Valor real, tipo *float*, com 6 bytes e precisão de 8 dígitos: $[10^{-38}$ até $10^{+38}]$.
- Valor real, tipo *double*, com 8 bytes e precisão de 15 dígitos: $[10^{-4932}$ até $10^{+4932}]$.



Exemplo - Constantes — II

Exemplo de código escrito em linguagem C

```
// Exemplo com tipo de dado decimal
// Definir e inicializar variáveis
float area = 0, raio = 3;
// Definir e inicializar a constante
const float PI = 3.14;
// Acessar a constante utilizando seu valor em um cálculo
area = ( PI * pow(raio , 2) );
// Imprimir o valor calculado em tela
printf("Imprimindo o valor calculado: %.2f", area);
```



Exemplo - Constantes — III

Ao tentar redefinir uma constante, resultará em erro do compilador.

Exemplo de código escrito em linguagem C

```
// Exemplo com tipo de dado decimal
// Definir e inicializar a constante
const float PI = 3.14;
// Tentativa de redefinir a constante
// Resultará em erro do compilador:
// |error: assignment of read-only variable 'PI'|
PI = 3.1415;
```



Exemplo - Booleano — I

Exemplo dado do tipo booleano

- Em C usa-se um artifício para definir um booleano a partir do inteiro.
- Em linguagens orientadas a objetos sempre haverá um tipo de dado específico para o tipo booleano.



Exemplo - Booleano — II

Exemplo de código escrito em linguagem C

```
// Exemplo com tipo de dado definido pelo programador
// Definir o tipo de dado booleano
typedef enum { false, true } bool;
// Definir e inicializar uma variável do tipo booleano
bool lampada = false;
// Acessar a variável imprimindo seu valor em tela
printf("A lampada esta acessa: %i", lampada);
```



Exemplo - Booleano — III

Exemplo de código escrito em linguagem js

```
// Define e inicializa a variável
var booleano = false;
// Operação lógica sobre o valor da variável
if (booleano === true) {
  // Se o valor da variável é true
  console.log('Valor se verdadeiro');
} else {
  // Se o valor da variável é false
  console.log('Valor se falso');
}
```



Exemplo - Cadeia de Caracteres — I

Exemplo dado do tipo cadeia de caracteres

- Em C deve-se determinar o tamanho máximo já na definição da variável.
- Em linguagens orientadas a objetos o tipo terá tamanho dinamicamente variável.



Exemplo - Cadeia de Caracteres — II

Exemplo de código escrito em linguagem C

```
// Exemplo com tipo de dado caractere
// Definir e inicializar a variável
char nome_aluno[60] = "Jack Daniel's";
// Acessar a variável imprimindo seu valor em tela
printf("Imprimindo o valor da variavel: %s", nome_aluno);
```



Exemplo - Cadeia de Caracteres — III

Exemplo de código escrito em linguagem js

```
// Define e inicializa a variável  
var nome_aluno = "Jack Daniel's";  
// Acessar a variável imprimindo seu valor em tela  
console.log("Imprimindo o valor da variável: %s", nome_aluno);  
// Imprimindo o tipo de dado da variável  
console.log("Tipo de dado da variável: %s", typeof nome_aluno);
```



Exemplo - Vetor — I

Exemplo de código escrito em linguagem C

```
// Exemplo com tipo de dado caractere
// Definir e inicializar a variável
char nome_aluno[] = "Jack Daniel's";
// Acessando um caractere específico através do índice
int i;
for (i = 0; i < sizeof(nome_aluno); i++) {
    printf("Valor no indice %d: %c \r\n", i, nome_aluno[i]);
}
```



Exemplo - Vetor — II

Exemplo de código escrito em linguagem js

```
// Define e inicializa a variável
var nome_aluno = "Jack Daniel's";
// Acessando um caractere específico através do índice
for (i = 0; i < nome_aluno.length; i++) {
  console.log("Valor no índice %i: %s", i, nome_aluno[i]);
}
```



Exemplo - Matriz — I

Exemplo de código escrito em linguagem C

```
// Definir e inicializar variáveis
int matriz[2][2], i, j;
// Atribuir valores
matriz[0][0] = 2;   matriz[0][1] = 4;
matriz[1][0] = 3;   matriz[1][1] = 5;
// Acessar os elementos da matriz
for (i = 0; i < 2; i++) { for (j = 0; j < 2; j++) {
    printf("Valor na matriz[%i][%i]: %i\r\n", i, j, matriz[i][j]);
} }
```



Exemplo - Matriz — II

Exemplo de código escrito em linguagem js

```
// Definir e inicializar variáveis
var matriz = [ [2, 4], [3, 5] ];
// Acessar os elementos da matriz
for (i = 0; i < 2; i++) { for (j = 0; j < 2; j++) {
    console.log("Valor na matriz[%i][%i]: %i", i, j, matriz[i][j]);
} }
```





O que há além?

Estruturas Lineares — I

- Operando com lista ainda temos:
 - Vetores e suas operações para manipulação de elementos.
 - Lista encadeada.
 - Lista duplamente encadeada.
 - Lista circular.
- Tabela de dispersão ou *hash*.



Estruturas Lineares — II

- Listas do tipo LIFO, ou pilha:
 - LIFO, do inglês: Last In, First Out
 - O último a entrar na lista é o primeiro a sair.
 - e.g.: pilha de roupas uma sobre a outra.
- Listas do tipo FIFO, ou fila:
 - FIFO, do inglês: First In, First Out
 - O primeiro a entrar na lista é o primeiro a sair.
 - e.g.: pessoas em uma fila de banco.



Estruturas Complexas

- Teoria dos Grafos.
- Ramo da matemática, oriundo do trabalho de Euler sobre o problema das sete pontes de Königsberg.
- Grafo simples e completo, árvore e diversos outros modelos.
- Algoritmos de busca do melhor caminho.



Obrigado!



UNINASSAU



João Ferreira da Silva Júnior


- joaoferreirape@gmail.com
- <http://lattes.cnpq.br/8904695743376784>
- <https://joaoferreirape.wordpress.com>





Referências


Referências


 COMEN, T. H. *ALGORITMOS - TEORIA E PRÁTICA*. 3. ed. [S.l.]: GEN LTC, 2012. 944 p. ISBN 978-8535236996.

 ENGELBRECHT, A.; NAKAMITI, G.; JUNIOR, D. *Algoritmos e Programação de Computadores*. 2. ed. [S.l.]: GEN LTC, 2019. 528 p. ISBN 978-8535292480.

 GERSTING, J. L. *Fundamentos Matemáticos para a Ciência da Computação*. 7. ed. [S.l.]: LTC, 2016. 908 p. ISBN 978-8521632597.

 LEVITIN, A. *Introduction to the design and analysis of algorithms*. 3. ed. [S.l.]: Pearson Education, 2011. 593 p. ISBN 978-0-13-231681-1.

 MEDINA, M.; FERTIG, C. *Algoritmos e Programação - Teoria e Prática*. 1. ed. [S.l.]: Novatec Editora, 2005. 384 p. ISBN 978-8575220733.

 RAWLINS, G. J. E. *Compared to What? An Introduction to the Analysis of Algorithms*. [S.l.]: W. H. Freeman, 1991. 536 p. ISBN 978-0716782438.



