

GUIA DE CODIFICAÇÃO PARA A DISCIPLINA DE TÉCNICAS DE PROGRAMAÇÃO E ALGORITMOS



Flávio Gomes da Silva Lisboa, Ph.D.
flavio.lisboa@fgsl.eti.br

Versão 2.0

Versão 1.0: Capítulo 1 a 5

Versão 2.0: Capítulo 6, tradução dos comentários e formatação do código-fonte

Sumário

1. Código-fonte.....	4
2. Estruturas básicas de código-fonte.....	5
2.1 Linguagem C.....	5
2.2 Linguagem Java.....	5
2.3 Linguagem Javascript.....	5
2.4 Linguagem PHP.....	6
2.5 Linguagem Python.....	6
3. Estruturas de decisão.....	7
3.1 Linguagem C.....	7
3.2 Linguagem Java.....	7
3.3 Linguagem Javascript.....	8
3.4 Linguagem PHP.....	8
3.5 Linguagem Python.....	9
4. Estruturas de repetição.....	10
4.1 Linguagem C.....	10
4.2 Linguagem Java.....	10
4.3 Linguagem Javascript.....	11
4.4 Linguagem PHP.....	11
4.5 Linguagem Python.....	12
5. Funções.....	13
5.1 Linguagem C.....	13
5.2 Linguagem Java.....	13
5.3 Linguagem Javascript.....	13
5.4 Linguagem PHP.....	14
5.5 Linguagem Python.....	14
6. Arrays (Vetores ou Matrizes de uma dimensão).....	15
6.1 Linguagem C.....	15
6.2 Linguagem Java.....	15
6.3 Linguagem Javascript.....	16
6.4 Linguagem PHP.....	17
6.5 Linguagem Python.....	17
REFERÊNCIAS.....	18
APÊNDICE.....	19
A1.TIPOS DE DADOS.....	19
A1.1 Linguagem C.....	19
A1.2 Linguagem Java.....	19
A1.3 Linguagem Javascript.....	20
A1.4 Linguagem PHP.....	20
A1.5 Linguagem Python.....	20

1. Código-fonte

O código-fonte de um programa é o código escrito em uma linguagem de programação, que é uma linguagem legível por seres humanos.

As linguagens de programação usam palavras da língua inglesa, pelo fato de terem sido criadas predominantemente nos Estados Unidos da América. Assim, **programar computadores requer um conhecimento mínimo da língua inglesa**.

As linguagens de programação são linguagens formais, que exigem precisão na expressão das instruções. Programar é como dar ordens, mas de uma forma precisa, clara, com detalhes e sem deixar qualquer dúvida sobre o que deve ser feito.

Contudo, [...], os computadores funcionam por meio da manipulação de bits. Assim, é necessário converter os programas escritos para um código [de uma linguagem de programação] em uma linguagem de máquina, mais adequada para manipular bits, o que é feito por programas denominados **tradutores** (CARVALHO; LORENA, 2017, p. 99).

Existem dois tipos de tradutores: os compiladores e os interpretadores.

Os **compiladores** traduzem o código-fonte de uma vez só e geram um arquivo chamado executável, que se torna independente do código-fonte. Isso quer dizer que você não precisa mais do código-fonte depois que compila o programa para poder executá-lo.

Um compilador traduz todo o programa original (código-fonte) de uma só vez, gerando então um código-objeto (ou código de máquina) do programa. Nesse processo, ele também identifica se o programa possui erros de codificação. Em caso positivo, esses erros são apontados para que o programador possa então corrigi-los. O código-objeto é codificado em uma linguagem mais próxima à linguagem de máquina e pode então ser executado pelo computador, gerando as ações especificadas pelo programa original (CARVALHO; LORENA, 2017, p. 99).

Exemplos de linguagens compiladas são as linguagens C e Go.

Os **interpretadores** traduzem o código-fonte e o executam instrução a instrução, sem gerar um arquivo independente. Isso significa que você precisa ter o código-fonte sempre que for executar o programa.

O interpretador alterna os passos de tradução e execução para cada linha individual de um programa [...]. Assim, o interpretador sucessivamente decodifica unidades básicas do programa (exemplo: comandos) e as executa imediatamente. Se houver algum erro que impeça a sua execução, o programa tem sua execução encerrada (CARVALHO; LORENA, 2017, p. 99).

Exemplos de linguagens interpretadas são as linguagens Javascript, PHP e Python.

A linguagem Java é **semi-compilada**: o compilador Java gera um arquivo independente do código-fonte, mas ele não é executado diretamente pelo sistema operacional, mas por uma máquina virtual. Se a máquina virtual não estiver instalada no sistema operacional, o programa Java não pode ser executado.

2. Estruturas básicas de código-fonte

2.1 Linguagem C

- O arquivo de código-fonte em C tem a extensão **.c**.
- Todo programa em C precisa ter uma função **main()**¹.
- As variáveis precisam ser declaradas. A declaração é feita com a expressão: [tipo] [nome da variável];²
- A compilação de um programa em C no Visual Studio Code é feita assim:

```
gcc [nome do arquivo de código-fonte] -o [nome do programa]
```

- A execução de um programa em C no Visual Studio Code é feita assim:

```
.\[nome do programa]
```

2.2 Linguagem Java

- O arquivo de código-fonte em Java tem a extensão **.java**.
- Todo programa em Java precisa ter uma **classe** com um método estático **main()**³.
- O nome do arquivo é o nome da classe contida nele. O nome do arquivo deve seguir a convenção **CamelCase**.
- As variáveis precisam ser declaradas. A declaração é feita com a expressão: [tipo] [nome da variável];⁴
- A compilação de um programa em Java no Visual Studio Code é feita assim:

```
javac [nome do arquivo de código-fonte]
```

- A execução de um programa em Java no Visual Studio Code é feita assim:

```
java [nome do programa]
```

- Para ler do teclado em um programa escrito em C é necessário usar a função scanf(), assim:

```
scanf("%[tipo de dado]", &variavel);5
```

- Para usar a função scanf() é necessário importar a biblioteca stdio.h, assim:

```
#include <stdio.h>
```

2.3 Linguagem Javascript

- O arquivo de código-fonte em Javascript tem a extensão **.js**.
- Javascript pode estar embutido dentro de uma página HTML, que terá a extensão **.html**.
- Se o código Javascript em uma página HTML não estiver dentro de uma função, ele será executado assim que a página for carregada.
- As variáveis precisam ser declaradas. A declaração é feita com a expressão: var [nome da variável] **ou** let [nome da variável]⁶

1 A função main() pode retornar int, se for necessário devolver um número, ou void, se não for necessário devolver nada.

2 Os tipos de dados de C estão no apêndice A1.1.

3 O método main() em Java tem a seguinte assinatura: *public static void main(String args[])*.

4 Os tipos de dados de Java estão no apêndice A1.2.

5 O [tipo de dado] em scanf() pode ser *d* para números inteiros, *f*, para números fracionários e *c* para caracteres.

6 A diferença entre *var* e *let* é o escopo. Variáveis declaradas com *var* são globais, visíveis por qualquer parte do código Javascript. Variáveis declaradas com *let* são visíveis apenas dentro do bloco onde foram declaradas.

- A execução de um programa Javascript embutido em uma página HTML é feita pelo browser (navegador).
- A execução de um programa em Javascript no Visual Studio Code pelo NodeJS é feita assim:

```
node .\[nome do arquivo de código-fonte]
```

- Para ler do teclado em um programa executado com NodeJs é necessário importar a biblioteca prompt-sync, assim:

```
const prompt = require("prompt-sync") ();
```

E depois usar a função prompt, assim:

```
var variavel = prompt('Mensagem');
```

2.4 Linguagem PHP

- O arquivo de código-fonte em PHP tem a extensão **.php**.
- Todo programa em PHP precisa começar com **<?php**.
- As variáveis não precisam ser declaradas, mas devem ter o nome iniciando com o caractere \$ (cifrão). A variável em PHP é criada quando um valor é atribuído a um identificador.
- A execução de um programa em php no Visual Studio Code é feita assim:

```
php.exe [nome do arquivo de código-fonte]
```

- Para ler do teclado em um programa executado com PHP é necessário usar a função fgets(), assim:

```
$variavel = fgets(STDIN);
```

2.5 Linguagem Python

- O arquivo de código-fonte em Python tem a extensão **.py**.
- Todo programa em Python **deve ser indentado ou não executa**.
- As variáveis não precisam ser declaradas. A variável em Python é criada quando um valor é atribuído a um identificador.
- A execução de um programa em Python no Visual Studio Code é feita assim:

```
python.exe [nome do arquivo de código-fonte]
```

- Para ler do teclado em um programa executado com Python é necessário usar a função input(), assim:

```
variavel = input("Mensagem");
```

- A função input() retorna um tipo string. Para convertê-lo para um número inteiro usamos a função int(). Para converter um número em texto, usamos a função str().

3. Estruturas de decisão

Nesta e nas próximas seções você verá EXEMPLOS de implementação dos elementos de um programa em várias linguagens. Um EXEMPLO serve para você se basear para codificar um programa, e não para copiá-lo sem entender o que ele está fazendo.

IF/ELSE (SE/ENTÃO)

3.1 Linguagem C

```
#include <stdio.h>

int main() {
    int number;

    printf("Digite um número inteiro: ");
    scanf("%d", &number);

    // Estrutura If-else
    if (number > 0) {
        printf("%d é um número positivo.\n", number);
    }
    else if (number < 0) {
        printf("%d é um número negativo.\n", number);
    }
    else {
        printf("O número é zero.\n");
    }

    return 0;
}
```

3.2 Linguagem Java

```
import java.util.Scanner;

public class IfElseExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Digite um número inteiro: ");
        int number = scanner.nextInt();

        // Estrutura if-else
        if (number > 0) {
            System.out.println(number + " é um número positivo.");
        }
        else if (number < 0) {
            System.out.println(number + " é um número negativo.");
        }
    }
}
```

```

    }
    else {
        System.out.println("O número é zero.");
    }

    scanner.close();
}
}

```

3.3 Linguagem Javascript

```

// Lê a entrada do usuário (em um navegador web)
let number = parseInt(prompt("Entre com um número inteiro:"));

// Estrutura If-else
if (number > 0) {
    console.log(`${number} é um número positivo.`);
}
else if (number < 0) {
    console.log(`${number} é um número negativo.`);
}
else {
    console.log("O número é zero.");
}

```

3.4 Linguagem PHP

```

<?php
// Lê a entrada do usuário
$number = (int) readline("Digite um número inteiro: "); // No
terminal
// $number = (int) $_POST['number']; // Em um formulário de uma
página HTML - as formas de leitura são mutuamente exclusivas

// Estrutura If-else
if ($number > 0) {
    echo "$number é um número positivo.\n";
}
elseif ($number < 0) {
    echo "$number é um número negativo.\n";
}
else {
    echo "O número é zero.\n";
}
?>

```

3.5 Linguagem Python

```
# Lê uma entrada do usuário (Python 3)
number = int(input("Digite um número inteiro: "))

# Estrutura If-elif-else
if number > 0:
    print(f"{number} é um número positivo.")
elif number < 0:
    print(f"{number} é um número negativo.")
else:
    print("O número é zero.")
```

4. Estruturas de repetição

WHILE (ENQUANTO) E FOR (PARA)

4.1 Linguagem C

while

```
#include <stdio.h>

int main() {
    int count = 5;  // Inicializa contador

    // Estrutura de repetição While
    while (count > 0) {
        printf("Contagem regressiva: %d\n", count);
        count--;    // Contador de decremento
    }

    printf("Decolar!\n");
    return 0;
}
```

for

```
#include <stdio.h>

int main() {
    // Estrutura de repetição For: inicialização; condição;
    incremento/decremento
    for (int count = 5; count > 0; count--) {
        printf("Contagem regressiva: %d\n", count);
    }

    printf("Decolar!\n");
    return 0;
}
```

4.2 Linguagem Java

while

```
public class WhileExample {
    public static void main(String[] args) {
        int count = 5;  // Inicializa contador

        // Estrutura de repetição While
        while (count > 0) {
            System.out.println("Contagem regressiva: " + count);
            count--;    // Contador de decremento
        }

        System.out.println("Decolar!");
    }
}
```

```
}
```

for

```
public class ForLoopExample {
    public static void main(String[] args) {
        // Estrutura de repetição For: inicialização; condição;
        incremento/decremento
        for (int count = 5; count > 0; count--) {
            System.out.println("Contagem regressiva: " + count);
        }
        System.out.println("Decolar!");
    }
}
```

4.3 Linguagem Javascript

while

```
let count = 5; // Inicializa contador

// Estrutura de repetição While
while (count > 0) {
    console.log(`Countdown: ${count}`);
    count--; // Contador de decremento
}

console.log("Decolar!");
```

for

```
// Estrutura de repetição For: inicialização; condição;
incremento/decremento
for (let count = 5; count > 0; count--) {
    console.log(`Contagem regressiva: ${count}`);
}
console.log("Decolar!");
```

4.4 Linguagem PHP

while

```
<?php
$count = 5; // Inicializa contador

// Estrutura de repetição While
while ($count > 0) {
    echo "Contagem regressiva: $count\n";
    $count--; // Contador de decremento
}

echo "Decolar!\n";
?>
```

for

```
<?php
// Estrutura de repetição For: inicialização; condição;
incremento/decremento
for ($count = 5; $count > 0; $count--) {
    echo "Contagem regressiva: $count\n";
}
echo "Decolar!\n";
?>
```

4.5 Linguagem Python

while

```
count = 5 # Inicializa countador

# Estrutura de repetição While
while count > 0:
    print(f"Contagem regressiva: {count}")
    count -= 1 # Contador de decremento

print("Decolar!")
```

for

```
# Estrutura de repetição For com intervalo (início, fim, contador)
for count in range(5, 0, -1):
    print(f"Contagem regressiva: {count}")

print("Decolar!")
```

5. Funções

5.1 Linguagem C

```
#include <stdio.h>

//Declaração de função (protótipo)
//O protótipo permite colocar a função main() antes das funções
que ela chama, porque você fez uma promessa ao compilador de que
irá implementar a função
int add(int a, int b);

int main() {
    int num1 = 5, num2 = 3;
    int result;

    // Chamada da função
    result = add(num1, num2);
    printf("Soma: %d\n", result); // Saída: Sum: 8

    return 0;
}

// Definição função
int add(int a, int b) {
    return a + b; // Retorne a soma
}
```

5.2 Linguagem Java

```
public class Main {

    // Definição de método
    static String greet(String name) {
        return "Olá, " + name + "!";
    }

    public static void main(String[] args) {
        // Chamada de método
        System.out.println(greet("Alice")); // Saída: Olá, Alice!
    }
}
```

5.3 Linguagem Javascript

```
// Declaração de função
function greet(name) {
    return `Olá, ${name}!`;
}
```

```
// Chamada de função
console.log(greet("Alice")); // Saída: "Olá, Alice!"
```

5.4 Linguagem PHP

```
<?php
// Definição de função
function greet($name) {
    return "Olá, $name!";
}

// Chamada de função
echo greet("Alice"); // Saída: Olá, Alice!
?>
```

5.5 Linguagem Python

```
# Definição de função
def greet(name):
    return f"Olá, {name}!"

# Chamada de função
print(greet("Alice")) # Saída: Olá, Alice!
```

6. Arrays (Vetores ou Matrizes de uma dimensão)

6.1 Linguagem C

```
#include <stdio.h>
int main() {
    // Declara um vetor de inteiros com 10 elementos
    int numbers[10];
    int i; // Contador do laço de repetição
    printf("Digite 10 números inteiros:\n");
    // Laço de repetição para ler cada elemento do vetor
    for (i = 0; i < 10; i++) {
        printf("Digite o elemento %d: ", i + 1); // Mostra o
        número do elemento contando a partir de 1
        scanf("%d", &numbers[i]); // Lê um número inteiro e
        armazena no vetor
    }
    // Imprime os elementos do vetor
    printf("\nOs elementos do vetor são:\n");
    for (i = 0; i < 10; i++) {
        printf("%d ", numbers[i]); // Imprime cada elemento
        seguido por um espaço
    }
    printf("\n"); // Imprime um caractere de nova linha para
    formatação
    return 0; // Indica sucesso na execução do programa
}
```

6.2 Linguagem Java

```
import java.util.Scanner;

public class VetorNumeros {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Declara um vetor de inteiros com 10 elementos
        int[] numbers = new int[10];

        System.out.println("Digite 10 números inteiros:");

        // Laço de repetição para ler cada elemento do vetor
        for (int i = 0; i < 10; i++) {
            System.out.print("Digite o elemento " + (i + 1) + ":
");
            numbers[i] = scanner.nextInt(); // Lê um número
            inteiro e armazena no vetor
        }
    }
}
```

```

        // Imprime os elementos do vetor
        System.out.println("\nOs elementos do vetor são:");
        for (int i = 0; i < 10; i++) {
            System.out.print(numbers[i] + " "); // Imprime cada
elemento seguido por um espaço
        }

        System.out.println(); // Nova linha para formatação
        scanner.close();
    }
}

```

6.3 Linguagem Javascript

```

const readline = require("readline");

// Interface para leitura do console
const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout,
});

let numbers = [];
let i = 0;

console.log("Digite 10 números inteiros:");

function askNumber() {
    if (i < 10) {
        rl.question(`Digite o elemento ${i + 1}: `, (answer) => {
            numbers[i] = parseInt(answer); // Converte a string para
inteiro
            i++;
            askNumber(); // Chama de novo até completar 10
        });
    } else {
        console.log("\nOs elementos do vetor são:");
        console.log(numbers.join(" ")); // Imprime os números
separados por espaço
        rl.close();
    }
}

askNumber();

```

6.4 Linguagem PHP

```
<?php
// Declara um vetor de inteiros com 10 elementos
$numbers = array();

echo "Digite 10 números inteiros:\n";

// Laço de repetição para ler cada elemento do vetor
for ($i = 0; $i < 10; $i++) {
    echo "Digite o elemento " . ($i + 1) . ": ";
    $input = trim(fgets(STDIN)); // Lê uma linha do console
    $numbers[$i] = (int)$input; // Converte para inteiro
}

// Imprime os elementos do vetor
echo "\nOs elementos do vetor são:\n";
for ($i = 0; $i < 10; $i++) {
    echo $numbers[$i] . " "; // Imprime cada elemento seguido por
um espaço
}

echo "\n"; // Nova linha para formatação
?>
```

6.5 Linguagem Python

```
# Declara um vetor de inteiros com 10 elementos
numbers = []

print("Digite 10 números inteiros:")

# Laço de repetição para ler cada elemento do vetor
for i in range(10):
    num = int(input(f"Digite o elemento {i + 1}: "))
    numbers.append(num)

# Imprime os elementos do vetor
print("\nOs elementos do vetor são:")
for num in numbers:
    print(num, end=" ") # Imprime cada elemento seguido por
espaço

print() # Nova linha para formatação
```

REFERÊNCIAS

CARVALHO, André C. P. L. F de. LORENA, Ana Carolina. **Introdução à computação**: hardware, software e dados. Rio de Janeiro: LTC, 2017.

APÊNDICE

A1.TIPOS DE DADOS

A1.1 Linguagem C

Tipo	Num de bits	Intervalo	
		Início	Fim
char	8	-128	127
unsigned char	8	0	255
signed char	8	-128	127
int	16	-32.768	32.767
unsigned int	16	0	65.535
signed int	16	-32.768	32.767
short int	16	-32.768	32.767
unsigned short int	16	0	65.535
signed short int	16	-32.768	32.767
long int	32	-2.147.483.648	2.147.483.647
signed long int	32	-2.147.483.648	2.147.483.647
unsigned long int	32	0	4.294.967.295
float	32	3,4E-38	3.4E+38
double	64	1,7E-308	1,7E+308
long double	80	3,4E-4932	3,4E+4932

A1.2 Linguagem Java

Tipo	Tamanho	Valor
byte	8 bits	-128 a 127
short	16 bits	-32.768 a 32.767
int	32 bits	-2.147.483.648 a 2.147.483.647
long	64 bits	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807
float	32 bits	-3.40292347E+38 a +3.40292347E+38
double	64 bits	-1.79769313486231570E+308 a +1.79769313486231570E+308
char	16 bits	'\u0000' a '\uFFFF'
boolean	1 bit	true ou false

A1.3 Linguagem Javascript

String, Number, BigInt, Boolean.

Para saber o tipo de uma variável em Javascript você pode usar o operador **typeof**.

A1.4 Linguagem PHP

Tipo	Tamanho	Valor
string	1 byte para cada caractere	Depende da quantidade de caracteres
integer	32 bits	9^{18}
boolean	1 bit	true ou false

Para saber o tipo de uma variável em PHP, você pode usar a função **gettype()**.

A1.5 Linguagem Python

Categoria	Tipos
texto	str
numéricos	int, float, complex
sequência	list, tuple, range
mapeamento	dict
conjuntos	set, frozenset
booleano	bool
binários	bytes, bytearray, memoryview

Para saber o tipo de uma variável em PHP, você pode usar a função **type()**.