ALGORITMOS E PROGRAMAÇÃO DE COMPUTADORES

Disciplina: 113476

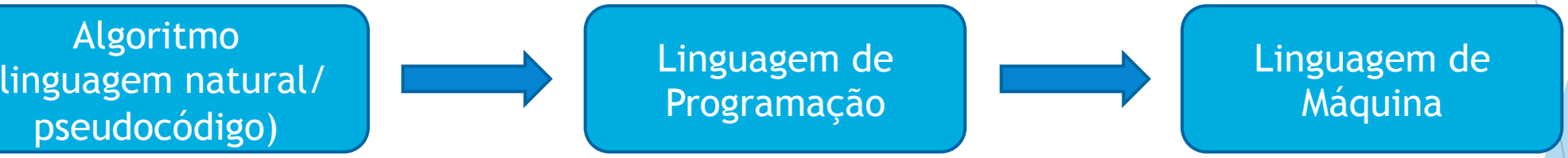
Profa. Carla Denise Castanho

Universidade de Brasília – UnB
Instituto de Ciências Exatas – IE
Departamento de Ciência da Computação – CIC

2. ALGORITMOS COMPUTACIONAIS

Algoritmos Computacionais

- Um **algoritmo computacional** é uma sequência de **instruções** que manipulam e geram **dados** e podem ser executadas por um **computador**.



- Existem algoritmos que não podem ser executados por um computador, seja porque o computador não tem **recursos** (trocar um pneu), seja porque ele são **indecidíveis** (você vai estudá-los na disciplina *Autômatos e Computabilidade*).

Tipos de Dados

- Para que sejam manipulados por computadores, os **dados** devem estar em um **formato** adequado e devem obedecer a um conjunto de **restrições**.
- Por isso os dados possuem **tipos**:
  - Numérico – guarda números inteiros ou reais;
  - Literal – guarda caracteres (texto);
  - Lógico – guarda valores lógicos (verdadeiro ou falso), também chamados de ***booleanos***, em homenagem ao matemático George Boole.

Tipos de Dados – Numérico

- Dados do tipo **inteiro** são números, positivos ou negativos, que não possuem parte fracionária, *i.e.*, pertencentes ao conjunto $\mathbb{Z}$.
  - Ex.: `36`; `0`; `-18`.

- Dados do tipo **real** são aqueles que podem possuir componentes decimais ou fracionários, e também podem ser positivos ou negativos, *i.e.*, pertencentes ao conjunto $\mathbb{R}$.
  - Ex.: `36.01`; `0.0`; `0`; `-18.5`; `42`.

Tipos de Dados – Literal

- Dados **literais** são sequências de **caracteres** que podem conter letras, dígitos e/ou símbolos especiais. Também são chamados de ***alfanumérico***, ***cadeia de caracteres*** ou, ainda, ***string***. Observe que também é válida uma string **vazia**!
- Usualmente são representados nos algoritmos pela coleção de caracteres delimitada por aspas (“ ”). Em algumas linguagens de programação, existe a diferenciação entre um caractere único (‘A’) e um conjunto de caracteres (“Olá Mundo”).
- Ex.: `“Quem?”; “”; “algoritmos e programacao de computadores”; “42”; “F”.`

Tipos de Dados – Lógicos

- São caracterizados como tipos **lógicos** (ou **booleanos**), os dados que podem assumir apenas os valores **verdadeiro** ou **falso** (mas nunca os dois ao mesmo tempo).
- Se definimos que Verdadeiro = 1, e Falso = 0, então um dado booleano pode ser representado em apenas 1 bit.

Algoritmos Computacionais - Conceitos

- Os conceitos básicos necessários para escrever um algoritmo computacional são:
  - Constantes
  - Variáveis
  - Palavras Reservadas
  - Expressões Aritméticas
  - Expressões Lógicas
  - Comandos de Atribuição e de Entrada/Saída

Constantes

- Constantes são valores que **não se modificam** ao longo da execução do programa.
- Uma constante criada pelo usuário deve ter um **nome**, um **tipo** e um **valor**. Nenhum desses três elementos pode ser modificado pelo programa. Ex.:

```
PI : real = 3.141522,
NOME_CHAR : string = "PEDRO".
```

- É convencional utilizar apenas letras maiúsculas e *underscore* ('_') no nome de constantes.

Constantes

- Constantes criadas pelo usuário tipicamente ocupam uma **região de memória** cujo **conteúdo** nunca muda.
- Mas valores literais no código também são considerados constantes, por exemplo:

```
x ← 2 * x
escreva("Olá, mundo!")
```

- Nesses exemplos, `2` e `"Olá, mundo!"` são *constantes literais*, elas têm tipo e valor, mas não receberam um nome definido pelo programador.

Variáveis

- Uma variável é uma **região de memória**. Isto é, ao **declarar** uma variável, o programador está reservando memória da máquina para ser usada por seu programa.
- Para que não precisemos nos referir ao **endereço** de memória de uma variável diretamente, as linguagens de programação nos permitem atribuir-lhe um **nome**.
- Toda variável deve também ter um **tipo** declarado.
- **Atenção:** o **conteúdo**, *i.e.*, o valor armazenado em uma variável, pode **mudar** ao longo da execução, mas **seu nome e seu tipo são constantes!**

Variáveis

- Por exemplo, veja as declarações de variáveis abaixo:

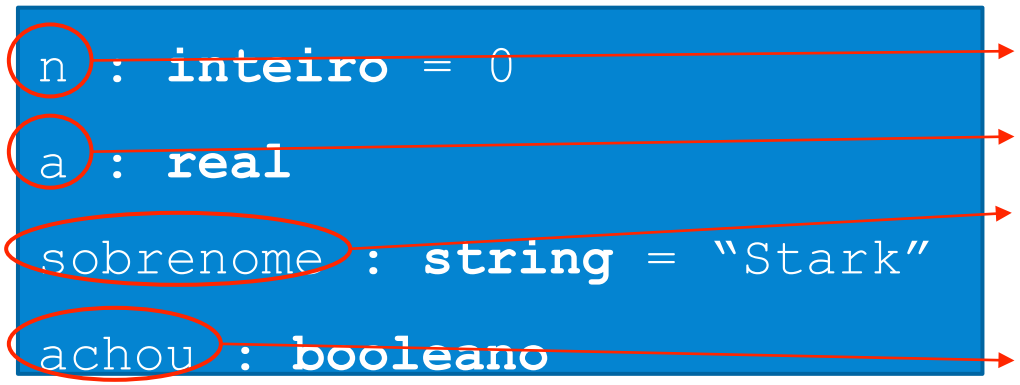
```
n : inteiro = 0
a : real
sobrenome : string = "Stark"
achou : booleano
```

- Observe que é possível definir um **valor inicial** para a variável.

Variáveis

- Por exemplo, veja as declarações de variáveis abaixo:

```
n : inteiro = 0
a : real
sobrenome : string = "Stark"
achou : booleano
```



endereço	valor
0x00000000	0
0x00000004	??
0x00000008	'S' 't' 'a' 'r'
0x0000000C	'k' ? ? ?
0x00000010	??
0x00000014	??
...	...

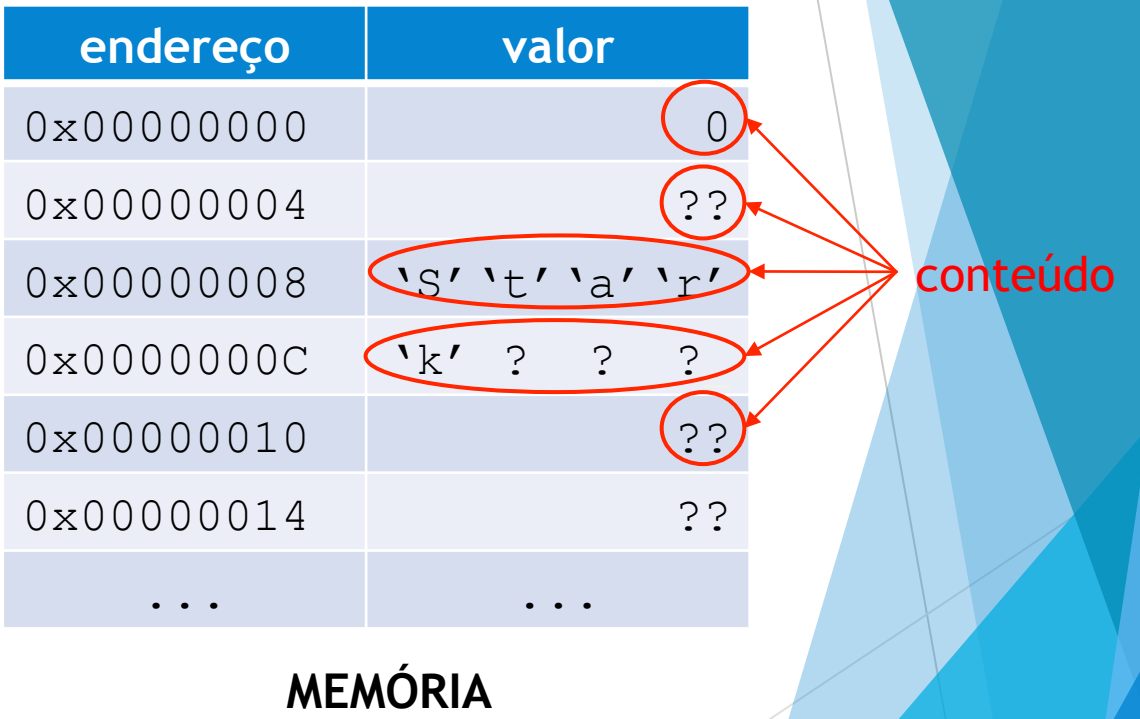
MEMÓRIA

- Observe que é possível definir um **valor inicial** para a variável.

Variáveis

- Por exemplo, veja as declarações de variáveis abaixo:

```
n : inteiro = 0
a : real
sobrenome : string = "Stark"
achou : booleano
```



- Observe que é possível definir um **valor inicial** para a variável.

Palavras Reservadas

- Palavras reservadas, são **identificadores** que têm um **significado especial** para o compilador, por isso não podem ser utilizadas pelo programador para nomear variáveis e constantes.
- Por exemplo, na Linguagem ANSI C, as palavras reservadas são:

```
auto  double  int  struct  break  else  long
switch  case  enum  register  typedef  char
extern  return  union  const  float  short
unsigned  continue  for  signed  void  default
goto  sizeof  volatile  do  if  static  while
```

Nomes de Variáveis

- Ao nomear variáveis, definimos algumas regras básicas:
  - O **primeiro caractere** deve ser uma letra;
  - Se houver mais de um caractere, só poderemos usar: **letras**, **undescore** (‘_’), ou **algarismos**;
  - Nomes de variáveis escritas com letras **maiúsculas** serão diferentes de letras **minúsculas**;
  - Nenhuma **palavra reservada** poderá ser usada como nome de uma variável;
  - Duas variáveis diferentes não poderão ter o **mesmo nome**;
  - Procure dar nomes **mnemônicos**, *i.e.*, significativos para a variável, mas não muito longos.

Nomes de Variáveis

- Exemplos de nomes **válidos**:

```
nota, salario, x, W, valor, Total,
valor_min, cont_alunos, n1, n2
```

- Exemplos de nomes **inválidos**:

`5B` - não começa com letra
`X+Y` - utiliza caractere não permitido '+'
`while` - palavra reservada

Declaração de Variáveis

- Variáveis são declaradas no **ínicio** do programa.
- Devem ter um nome e um tipo, que pode ser: **inteiro**, **real**, **literal** (string) ou **lógico** (booleano).
- Ex.:

```
codigo : literal (ou string)
quantidade : inteiro
preço, valortotal : real
vlr_conta, tx_multa, vlr_multa,
vlr_total : real
fim : lógico (ou booleano)
```

Expressões e Operadores

- Para realizar cálculos, podemos escrever **expressões aritméticas**. Nessas expressões, podem aparecer apenas variáveis e constantes **numéricas** e os seguintes **operadores**:

Prioridade	Operador	Operações	Tipo	Exemplo
maior ↓ menor	( )	altera prioridade	unário	`5 * (1 + 2) = 15`
	-, +	troca de sinal, manutenção de sinal	unário	`-(-1) = 1` `+(-5) = -5`
	^	exponenciação	binário	`2^3 = 8`
	*, /	multiplicação, divisão	binário	`1.0 / 2 = 0.5`
	+, -	soma, subtração	binário	`1 + 1 = 2`

Expressões e Operadores

- Observe que os operadores têm **prioridades**, que mudam a **ordem de avaliação** da expressão.
- Operadores com a **mesma prioridade** (ex.: `+` e `-`, `*` e `/`) são avaliados **da esquerda para a direita**.

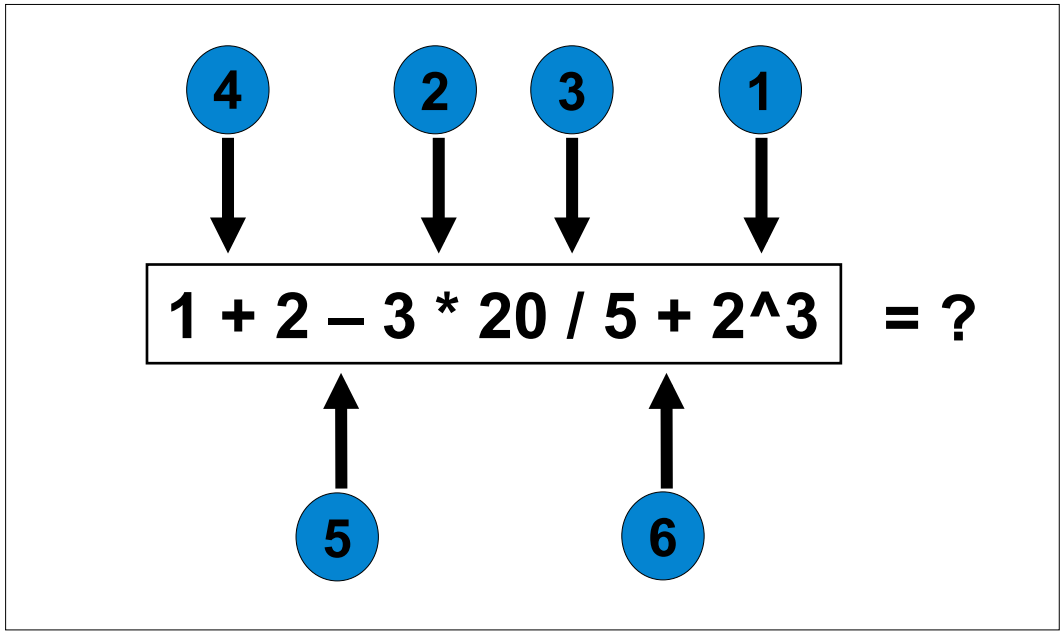
Expressões e Operadores

- Observe que os operadores têm **prioridades**, que mudam a **ordem de avaliação** da expressão.
- Operadores com a **mesma prioridade** (ex.: `+` e `-`, `*` e `/`) são avaliados **da esquerda para a direita**.

1 + 2 – 3 * 20 / 5 + 2^3 = ?

Expressões e Operadores

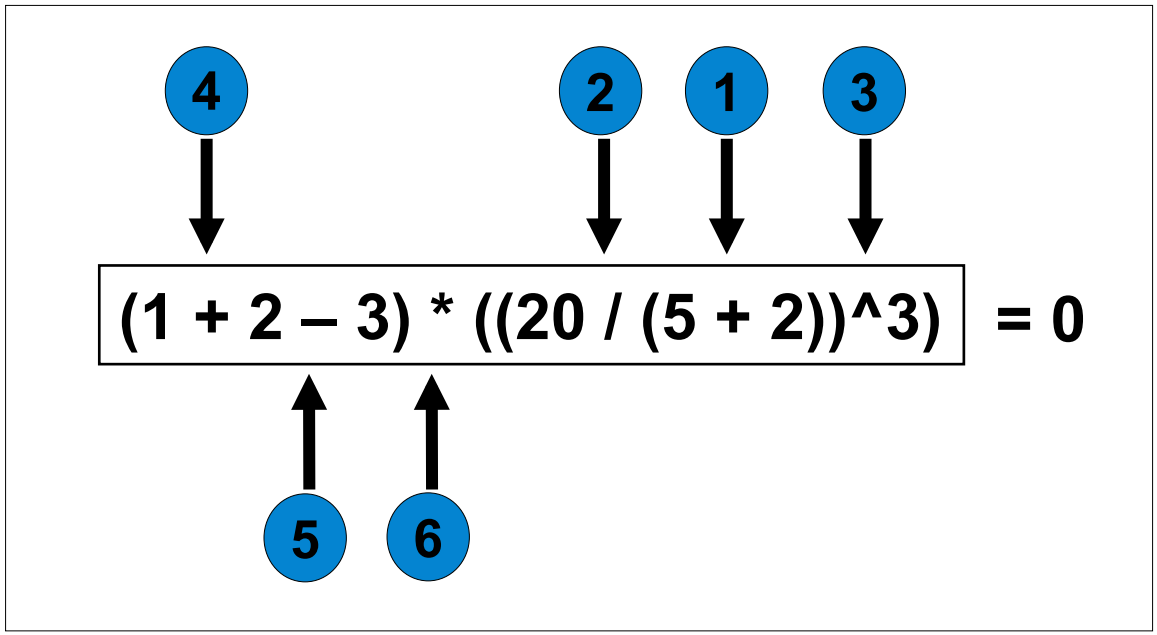
- Observe que os operadores têm **prioridades**, que mudam a **ordem de avaliação** da expressão.
- Operadores com a **mesma prioridade** (ex.: `+` e `-`, `*` e `/`) são avaliados **da esquerda para a direita**.



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

Expressões e Operadores

- Podemos utilizar parênteses para alterar a prioridade:



Expressões e Operadores

- Quando precisamos verificar condições, utilizamos **expressões relacionais e lógicas**.
- Expressões relacionais **comparam** valores, e retornam **verdadeiro** ou **falso**:

Prioridade	Operador	Operações	Tipo	Exemplo
maior ↓ menor	<, <=, >, >=	menor, menor ou igual maior, maior ou igual	binário	0 < 0 : F 1 <= 2 : V -1 > -5 : V 0 >= 7.8 : F
	=, ≠	igual, diferente	binário	1 = 1 : V 0 ≠ 0 : F

Expressões e Operadores

- Expressões lógicas têm operandos com o valor **verdadeiro** ou **falso** e geram um resultado também **verdadeiro** ou **falso**, de acordo com os operadores:

Prioridade	Operador	Operações	Tipo
maior ↓ menor	NÃO	não lógico (negação)	binário
	E	e lógico (conjunção)	binário
	OU	ou lógico (disjunção)	binário

Tabela Verdade

A	B	NÃO A	NÃO B	A E B	A OU B
V	V	F	F	V	V
V	F	F	V	F	V
F	V	V	F	F	V
F	F	V	V	F	F

Expressões e Operadores

- Exemplos:

Valores	Expressão	Resultado
sal: 1000, aluguel: 100	aluguel >= 0.1 * sal	V
A: 3, B: 4, C: 5	(A<B+C) E (B<A+C) E (C<A+B)	V
M: 1, N: 2, H: 3	H = M^2 + N^2	F
X: -7.8, Y: 3.141592	(X > 0) OU (Y > 0)	V
m: 50, g: 10, P: 500	P ≠ m * g	F
k: 4	k >= 1+2*1+0	V

Algumas Funções

- É possível utilizar algumas funções matemáticas predefinidas:

Função	Descrição	Exemplo	Linguagem C
t(A)	Retorna a parte inteira da expressão A.	`int(2.71) = 2`	Basta atribuir A a uma variável do tipo **int**, ou fazer: `(int) A`.
t(A/B)	Rt. a parte inteira da divisão de A por B.	`int(10/3) = 3`	Basta que os operandos sejam inteiros, caso não sejam, faça: `(int)(A/B)`.
ac(A)	Rt. a parte fracionária da expressão A.	`frac(10.345) = 0.345`	Subtraia a parte inteira do valor original: `A-(int)A`.
esto(A/B)	Rt. o resto da divisão (inteira) de A por B.	`resto(10/3) = 1`	Utilize o operador `%` (apenas para operandos inteiros): `A % B`.
^B	Exponenciação	`0.5 ^ 2 = 0.25`	`pow(A, B)`
iz(A)	Rt. a raiz quadrada de A.	`raiz(2) = 1.41421356`	`sqrt(A)`

Comandos

- Comando de Atribuição:
  - Permite atribuir um valor a uma variável.
  - Indicado pelo símbolo: ←
  - Ex.:

```
nota ← (n1 + n2) / 2
nome ← "Joao"
```

- Comandos de Entrada e Saída:
  - Permitem a interação com o usuário - entrada pelo teclado e saída pela tela.
  - Comando de entrada: `leia(<lista de variáveis>)`
  - Comando de saída: `escreva(<lista de variáveis/constantes>)`
  - Ex.:

```
leia(nome, idade)
escreva("Nota: ", nota_final)
```

Pseudocódigo

- Vamos relembrar a forma geral da representação de um algoritmo em pseudocódigo:

Pseudocódigo - Estrutura Básica

```
Algoritmo <nome do algoritmo>.
<declaração de variáveis>
<subalgoritmos>
Início
<corpo do algoritmo>
Fim.
```

Pseudocódigo

- Declaração de variáveis:

Algoritmo - Declaração de Variáveis

```
Algoritmo DeclaraVariaveis.
Variáveis
    idade: inteiro
    altura: real
    sexo : caractere
    nome : literal
Início
...
Fim.
```

Pseudocódigo

- Atribuição de valores:

Algoritmo - Atribuição

```
Algoritmo AtribuiValores.
Variáveis
   nro: inteiro
Início
   nro ← 1
Fim.
```

Pseudocódigo

- Entrada e saída de dados:

Algoritmo - Entrada e Saída

```
Algoritmo Media.
Variáveis
    num1, num2, media: real
Início
    Escreva (“Informe o primeiro número:”)
    Leia (num1)
    Escreva (“Informe o segundo número:”)
    Leia (num2)
    media ← (num1 + num2) / 2
    Escreva (“A média dos números informados é: ”,
    media)
Fim.
```

Estrutura Sequencial

- Observe que todos os comandos entre **Início** e **Fim** são executados de maneira sequencial:

Algoritmo Sequencial

```
Algoritmo <nome>.
Variáveis
    <lista de variáveis>
Início
    <comando 1>
    <comando 2>
    ...
    <comando n>
Fim.
```

Estrutura Sequencial

- Observe que todos os comandos entre **Início** e **Fim** são executados de maneira sequencial:

Algoritmo Sequencial

```
Algoritmo <nome>.
Variáveis
    <lista de variáveis>
Início
    <comando 1>
    <comando 2>
    ...
    <comando n>
Fim.
```

Todos os exemplos anteriores são ALGORITMOS SEQUENCIAIS!

Estrutura Sequencial

- Vamos ver um exemplo mais complexo:

Estrutura Sequencial

- Vamos ver um exemplo mais complexo:
  - Sobre o salário bruto de um funcionário, são descontados:
    - 8% de INSS,
    - 10% de IR (imposto de renda).
    - e, sobre o restante, 0.5% referente à filiação sindical.
  - Para cada dependente (filhos), o funcionário ganha R$ 50,00.
  - Dado o salário bruto de um funcionário e a quantidade de dependentes, calcule e mostre o total de descontos, o total de acréscimos, e o salário líquido.

Estrutura Sequencial

Algoritmo - Folha de Pagamento

```
Algoritmo CalculoDeFolha.
Variáveis
     salariobruto, INSS, IR, FS, totaldescontos,
          totalacrescimos, salarioliquido: real
     dependentes : inteiro
Início
     Escreva ("Informe o salário bruto: ")
     Leia (salariobruto)
     Escreva ("Informe o número de dependentes: ")
     Leia (dependentes)
     INSS ← salariobruto * 0.08
     IR ← salariobruto * 0.10
     FS ← (salariobruto – (INSS+IR)) * 0.005
     totaldescontos ← INSS + IR + FS
     totalacrescimos ← dependentes * 50
     salarioliquido ← salariobruto – totaldescontos +
     totalacrescimos
     Escreva ("O total de descontos é: ", totaldescontos)
     Escreva ("O total de acréscimos é: ", totalacrescimos)
     Escreva ("O salario líquido é: ", salarioliquido)
Fim.
```

Estrutura Sequencial

Algoritmo - Folha de Pagamento

```
Algoritmo CalculoDeFolha.
Variáveis
     salariobruto, INSS, IR, FS, totaldescontos,
          totalacrescimos, salarioliquido: real
     dependentes : inteiro
Início
     Escreva ("Informe o salário bruto: ")
     Leia (salariobruto)
     Escreva ("Informe o número de dependentes: ")
     Leia (dependentes)
     INSS ← salariobruto * 0.08
     IR ← salariobruto * 0.10
     FS ← (salariobruto – (INSS+IR)) * 0.005
     totaldescontos ← INSS + IR + FS
     totalacrescimos ← dependentes * 50
     salarioliquido ← salariobruto – totaldescontos +
     totalacrescimos
     Escreva ("O total de descontos é: ", totaldescontos)
     Escreva ("O total de acréscimos é: ", totalacrescimos)
     Escreva ("O salario líquido é: ", salarioliquido)
Fim.
```

Variáveis auxiliares: não são variáveis nem de entrada nem de saída.

Estrutura Sequencial

Algoritmo - Folha de Pagamento

```
Algoritmo CalculoDeFolha.
Variáveis
    salariobruto: real
    dependentes : inteiro
Início
    Escreva ("Informe o salário bruto: ")
    Leia (salariobruto)
    Escreva ("Informe o número de dependentes: ")
    Leia (dependentes)
    Escreva ("O total de descontos é: ", (salariobruto * 0.08) +
        (salariobruto * 0.10) + ((salariobruto - ((salariobruto * 0.08)  +
        (salariobruto * 0.10)) * 0.005)
    Escreva ("O total de acréscimos é: ", dependentes * 50)
    Escreva ("O salario líquido é: ", salariobruto - (salariobruto * 0.08)
        + (salariobruto * 0.10) + ((salariobruto - ((salariobruto * 0.08)
        + (salariobruto * 0.10))) * 0.005) + (dependentes * 50)
Fim.
```

Sem a utilização de variáveis auxiliares!

Estrutura Sequencial

Algoritmo - Folha de Pagamento

```
Algoritmo CalculoDeFolha.
Variáveis
    salariobruto: real
    dependentes : inteiro
Início
    Escreva ("Informe o salário bruto: ")
    Leia (salariobruto)
    Escreva ("Informe o número de dependentes: ")
    Leia (dependentes)
    Escreva ("O total de descontos é: ", (salariobruto * 0.08) +
        (salariobruto * 0.10) + ((salariobruto - ((salariobruto * 0.08)  +
        (salariobruto * 0.10)) * 0.005)
    Escreva ("O total de acréscimos é: ", dependentes * 50)
    Escreva ("O salario líquido é: ", salariobruto - (salariobruto * 0.08)
        + (salariobruto * 0.10) + ((salariobruto - ((salariobruto * 0.08)
        + (salariobruto * 0.10))) * 0.005) + (dependentes * 50)
Fim.
```

Sem a utilização de variáveis auxiliares!

O algoritmo fica "poluído" e de difícil compreensão!

Estrutura Sequencial - Exercícios

1. Faça um algoritmo que leia um número inteiro e mostre seu sucessor e antecessor.
2. Faça um algoritmo que leia a idade da pessoa expressa em anos, meses e dias e mostre-a expressa apenas em dias. Assuma que o ano tem sempre 365 dias e o mês sempre 30 dias.
3. O custo ao consumidor de um carro novo é a soma do custo de fábrica com a porcentagem do distribuidor e dos impostos (aplicados ao custo de fábrica). Supondo que a porcentagem do distribuidor seja de 28% e os impostos de 45%, escreva um algoritmo que leia o custo de fábrica e imprima o custo ao consumidor.
4. Faça um algoritmo que leia a idade de uma pessoa expressa em dias e mostre-a expressa em anos, meses e dias. Assuma que o ano tem sempre 365 dias e o mês sempre 30 dias.
5. Faça 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 o peso das notas é: 1, 3 e 4, respectivamente.