

**INSTITUTO FEDERAL**

Paraná

Campus Assis Chateaubriand

# Algoritmos e Lógica de Programação

## Vetores (*arrays*)

# Vetores (*arrays*)

## Conteúdo

- Conceito de vetores;
- Manipulação de vetores e acesso indexado;
- Emprego de vetores em situações práticas;
- Uso dos mesmos índices para vetores diferentes (“vetores paralelos”);
- Vetores bidimensionais (*matrizes*);
- Vetores multidimensionais.

# Vetores (*arrays*)

## Objetivos de Aprendizagem

1. Compreender o que são vetores e como eles se distinguem das variáveis convencionais;
2. Identificar as situações em que os vetores são necessários;
3. Manipular os vetores por meio de *índices*, utilizando-os como *variáveis indexadas*;
4. Utilizar vetores em paralelo;
5. Utilizar as estruturas de repetição apropriadas para manipular vetores;
6. Realizar buscas sequenciais em vetores.

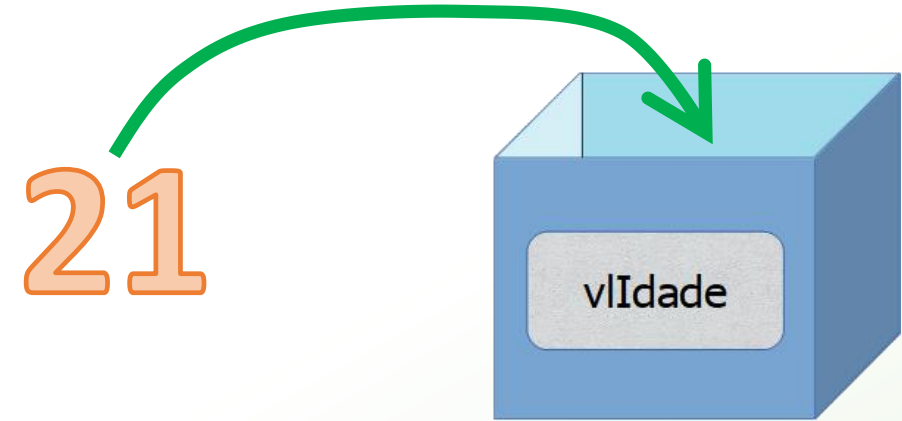
# O que são vetores e por que são úteis?

## Aplicando a Analogia da Caixinha para Entender os Vetores

### A Analogia da Caixinha

Sabemos que uma **variável** é uma forma conveniente de referenciar um valor armazenado na memória. Para assimilar essa ideia, é possível utilizar a analogia da caixinha que, embora não seja perfeita, ajuda a entender os conceitos envolvidos na definição de variável (*identificador*, *valor* e *tipo*).

Imagine que no meio do código de um programa, você tenha que armazenar a idade de uma pessoa, digamos, 21. Para isso, você teria que utilizar uma variável, digamos, *vlIdade*. Utilizando a analogia, isso equivaleria a colocar o valor "21" dentro de uma caixinha rotulada como "vlIdade", conforme mostrado na figura ao lado. Claramente, só é possível armazenar um valor de idade na caixinha; então, para armazenar a idade de mais pessoas serão necessárias outras caixinhas, uma por pessoa. A questão que surge nesse momento é: quantas caixinhas (variáveis) deveriam ser usadas no programa se você não souber de antemão quantas pessoas terão a idade registrada? E como manipular esses valores?



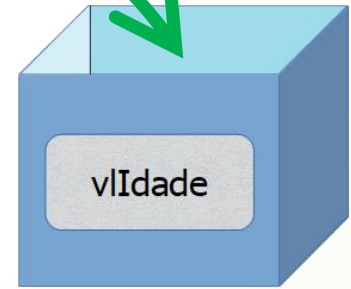
**Analogia da caixinha:** uma variável é utilizada num programa como se fosse uma caixa rotulada que utilizamos para guardar nossos pertences em nossas casas. Na analogia, os "pertences" são *valores* que precisam ser armazenados por alguma razão e o rótulo é um *identificador*. Uma das vantagens da analogia é destacar que *uma variável armazena apenas um valor de cada vez*. Então, como armazenar *vários* valores de uma vez? Claro, podemos utilizar muitas variáveis para isso, mas nem sempre essa é uma estratégia eficiente.

# O que são vetores e por que são úteis?

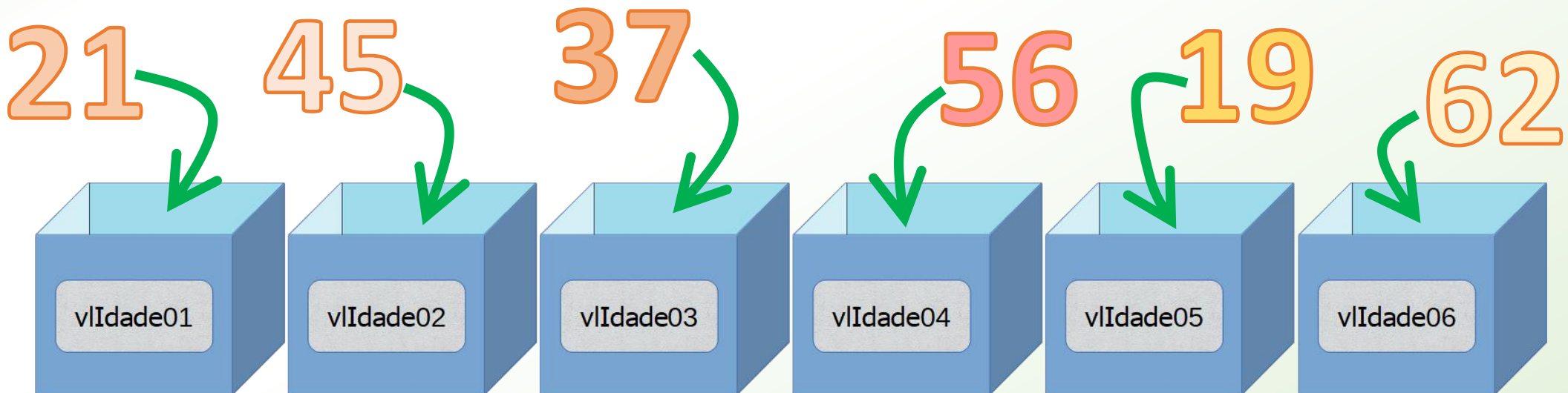
## Aplicando a Analogia da Caixinha para Entender os Vetores

Para armazenar um valor de idade, naturalmente utilizamos uma única variável

21



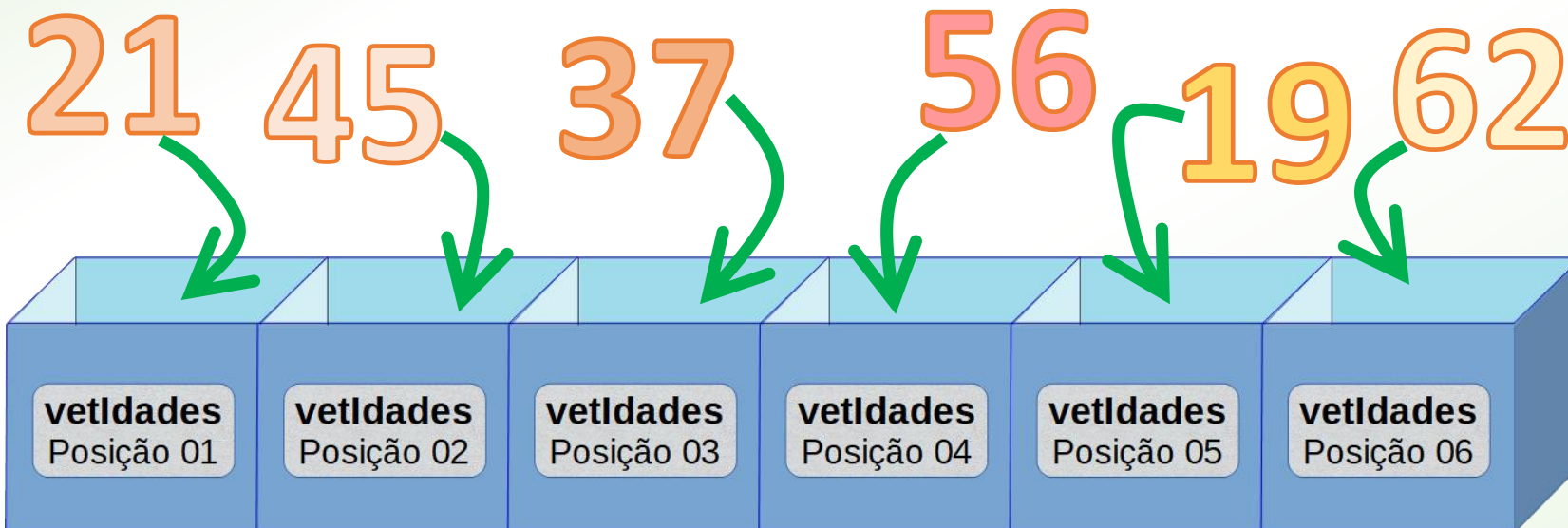
Para vários valores, poderíamos utilizar tantas variáveis quanto forem necessárias, digamos, 6 variáveis para armazenar 6 idades (como **21**, **45**, **37**, **56**, **19** e **62**). Mas essas variáveis teriam que ser declaradas e manipuladas individualmente, tornando o código redundante, extenso e penoso de entender. Isso se tornaria ainda pior se, ao invés de 6 pessoas, fossem 60 ou, digamos, 600.



# O que são vetores e por que são úteis?

## Aplicando a Analogia da Caixinha para Entender os Vetores

Ao invés de 6 variáveis diferentes, pode-se utilizar um único vetor ("vetIdades") de 6 posições, como mostrado abaixo. Assim, para armazenar as idades **21**, **45**, **37**, **56**, **19** e **62** usariamos comandos como `vetIdades[1] ← 21`, `vetIdades[2] ← 45`, `vetIdades[3] ← 37` e assim por diante. Note que *cada posição do vetor é identificada por um índice numérico* que varia de 1 a 6, e que *cada posição pode ser lida ou escrita como uma variável convencional*. A maior vantagem de um vetor é poder referenciar uma **coleção de valores** ao invés de um único valor, como acontece com as variáveis convencionais. Isso traz flexibilidade, pois **basta mudar o tamanho do vetor para acomodar mais ou menos valores**, sem necessidade de criar novas variáveis ou destruir as existentes.



Os vetores são úteis para lidar com vários dados diferentes que estão relacionados como, por exemplo, a idade de várias pessoas ou o nome delas. Sem vetores, teríamos que utilizar várias variáveis convencionais, tornando o código complexo, difícil de entender e – ainda pior – se o número de dados mudar, algumas variáveis terão que ser criadas ou destruídas.

Um vetor com 6 posições. Apenas uma variável para armazenar 6 valores!



# O que são vetores e por que são úteis?

## Muito Além da Analogia da Caixinha

Assim como acontece com qualquer variável, um vetor é armazenado fisicamente na memória RAM mas, para tornar o acesso indexado mais eficiente, *as posições desse vetor devem ser contíguas*. No exemplo da figura ao lado, **vetIdades** é armazenado a partir do endereço de memória 0xFB04 e cada um de seus elementos ocupa 2 bytes. Como o computador sabe qual é o endereço inicial do vetor e quanto espaço cada elemento ocupa, fica fácil calcular em qual endereço de memória está localizado cada um dos elementos. Tudo que o programador tem que fazer é especificar um determinado índice e o computador calcula o endereço correto. Contudo, essa forma de funcionamento tem um inconveniente: *para garantir que um vetor seja armazenado num espaço contíguo na memória, o seu número de posições é estático, ou seja, não pode ser modificado durante a execução do programa*. Nas situações em que isso é uma limitação, utiliza-se estruturas tais como as listas encadeadas, que são alocadas dinamicamente mas, em contra partida, não são indexadas (são percorridas sequencialmente).

Endereços de Memória	Valores do Vetor	
0xFB04	21	1
0xFB06	45	2
0xFB08	37	3
0xFB0A	56	4
0xFB0C	19	5
0xFB0E	62	6

Uma pequena área da Memória RAM

Índices do Vetor

Ao menos conceitualmente, um vetor ocupa um espaço contíguo na RAM; ou seja, seus valores ocupam posições vizinhas para garantir mais eficiência no acesso indexado.

# O que são vetores e por que são úteis?

Formalmente, um vetor consiste numa estrutura de dados composta por uma coleção de elementos indexados (ou seja, ordenados), independentes (são manipulados individualmente) e homogêneos (todos os elementos devem ser do mesmo tipo).

## Usando Vetores

Um vetor permite manipular convenientemente cada um de seus elementos por meio de índices numéricos, facilitando o processamento nas situações em que os dados são naturalmente dispostos em lista. Por exemplo, imagine que você tenha que construir um programa que precise manipular os dados de 50 pessoas que trabalham numa certa empresa. Como fazer isso com variáveis convencionais? E se contratassem mais dois empregados ou demitissem três deles, qual seria o impacto no código do programa?

## O Mundo Sem Vetores

**Algoritmo** semVetores;

**Declaração de Variáveis**

vlIdade01, vlIdade02, ..., vlIdade50: **inteiro**;

**Início**

**Escreva** ('Qual é a idade da pessoa 1?');

**Leia** (vlIdade01);

**Escreva** ('Qual é a idade da pessoa 2?');

**Leia** (vlIdade02);

⋮

**Escreva** ('Qual é a idade da pessoa 49?');

**Leia** (vlIdade49);

**Escreva** ('Qual é a idade da pessoa 50?');

**Leia** (vlIdade50);



# O que são vetores e por que são úteis?

50 variáveis, declaradas separadamente e manipuladas uma a uma. Se o número de empregados mudar, o número de variáveis tem que mudar. Isso parece razoável???

Esse tipo de abordagem prejudica a clareza do código, é redundante e inflexível (exige modificações significativas no programa se o número de empregados mudar). Muito claramente, é necessário uma abordagem mais racional!

## O Mundo Sem Vetores

**Algoritmo** semVetores;

**Declaração de Variáveis**

vlIdade01, vlIdade02, ..., vlIdade50: inteiro;

**Início**

**Escreva** ('Qual é a idade da pessoa 1?');

**Leia** (vlIdade01);

**Escreva** ('Qual é a idade da pessoa 2?');

**Leia** (vlIdade02);

⋮

**Escreva** ('Qual é a idade da pessoa 49?');

**Leia** (vlIdade49);

**Escreva** ('Qual é a idade da pessoa 50?');

**Leia** (vlIdade50);

# O que são vetores e por que são úteis?

50 variáveis, declaradas separadamente e manipuladas uma a uma. Se o número de empregados mudar, o número de variáveis tem que mudar. Isso parece razoável???

Esse tipo de abordagem prejudica a clareza do código, é redundante e inflexível (exige modificações significativas no programa se o número de empregados mudar). Muito claramente, é necessário uma abordagem mais racional!

O Mundo Sem Vetores

```
Algoritmo semVetores;  
Declaração de Variáveis  
  vlIdade01, vlIdade02, ..., vlIdade50: inteiro;  
Inicio  
  Escreva('Qual é a idade da pessoa 1?');  
  Leia(vlIdade01);  
  Escreva('Qual é a idade da pessoa 2?');  
  Leia(vlIdade02);  
  ...  
  Escreva('Qual é a idade da pessoa 49?');  
  Leia(vlIdade49);  
  Escreva('Qual é a idade da pessoa 50?');  
  Leia(vlIdade50);
```

Controle de Qualidade

Código Reprovado

Encaminhar para remanufatura

# O que são vetores e por que são úteis?

## O Mundo Com Vetores

**Algoritmo** comVetores;

**Declaração de Variáveis**

vetIdades: **vetor** [1..50] **de inteiro**;

cont : **inteiro**;

**Início**

**Para** cont  $\leftarrow$  1 **Até** 50 **Faça**

**Início**

**Escreva** ('Qual é a idade da pessoa ', cont, '?') ;

**Leia** (vetIdades[cont]) ;

**Fim**;

  :

**Fim.**

# O que são vetores e por que são úteis?

Aqui, a primeira posição de um vetor recebe o índice 1, mas em muitas linguagens de programação como o Java, esse índice é definido automaticamente como 0.

## O Mundo Com Vetores

A utilização de vetores torna o código mais claro e conciso!

**Algoritmo** comVetores;

**Declaração de Variáveis**

vetIdades: **vetor** [1..50] **de inteiro**;

cont : **inteiro**;

**Início**

**Para** cont ← 1 **Até** 50 **Faça**

**Início**

**Escreva** ('Qual é a idade da pessoa ', cont, '?') ;

**Leia** (vetIdades[cont]) ;

**Fim**;

**Fim.**

Tamanho do vetor. Indica quantas posições ele possui.

5 linhas de código para inicializar 50 valores. Se fossem 500 valores, bastaria mudar o tamanho do vetor e a condição de parada do laço; as 5 linhas permanecem iguais.

Com vetores, é possível utilizar estruturas de repetição para tornar o código mais conciso.

Sem o uso de um vetor, seria necessário: declarar 50 variáveis diferentes e inicializá-las uma a uma. Somente para inicializá-las, seriam necessárias 100 linhas de código! Com o vetor, declara-se uma única variável ("vetIdades") e utiliza-se 5 linhas de código para inicializá-la.

# Algoritmos com Vetores

## Problema

Imagine que você esteja desenvolvendo um programa para uma empresa que contrata vários profissionais autônomos (“free lancers”) que são pagos por hora trabalhada. Você sabe de antemão quantos profissionais prestam serviço para a empresa (digamos que sejam 50) e o seu programa precisa armazenar quantas horas cada um desses profissionais trabalhou. Como você criaria seu programa utilizando vetores? E como faria para criá-lo *sem* vetores?



# Algoritmos com Vetores

**Algoritmo** calculaSalarios01;

**Constantes**

numEmps  $\leftarrow$  50;

**Variáveis**

cont : inteiro;

horasEmps : vetor [1..numEmps] de real;

**Início**

**Para** cont  $\leftarrow$  1 **Até** numEmps **Faça**

**Início**

**Escreva** ("Quantas horas trabalhou o profissional", cont, "?");

**Leia** (horasEmps[cont]);

**Fim**;

**Para** cont  $\leftarrow$  1 **Até** numEmps **Faça**

**Início**

**Escreva** ("O profissional ", cont, " trabalhou ", horasEmps[cont], " horas");

**Fim**;

**Fim.**

# Algoritmos com Vetores

**Algoritmo** calculaSalarios01;

**Constantes**

numEmps ← 50;

**Variáveis**

cont : inteiro;

horasEmps : vetor [1..numEmps] de real;

**Início**

**Para** cont ← 1 **Até** numEmps **Faça**

**Início**

**Escreva** ("Quantas horas trabalhou o profissional", cont, "?");

**Leia** (horasEmps[cont]);

**Fim;**

**Para** cont ← 1 **Até** numEmps **Faça**

**Início**

**Escreva** ("O profissional ", cont, " trabalhou ", horasEmps[cont], " horas");

**Fim;**

**Fim.**

O tamanho do vetor pode ser configurado por uma variável ou constante inicializada apropriadamente. Note: o tamanho do vetor é estático, não pode ser modificado ao longo da execução do programa!

O tamanho do vetor é definido na construção do programa (em tempo de compilação). Não há como alterá-lo dinamicamente.

Basta mudar este valor para processar outra quantidade de profissionais. Usando vetores, essa quantidade pode ser alterada com um impacto mínimo no código.

Note que: (1) graças aos índices, os vetores podem ser percorridos com o uso de estruturas de repetição e (2) cada elemento dos vetores é tratado como uma variável qualquer.

# Algoritmos com Vetores

**Algoritmo** calculaSalarios01;

**Constantes**

numEmps ← 50;

Como seu programa teria que ser modificado para que, além do número de horas trabalhadas, ele armazene também o salário que cada um deve receber? Imagine que o valor da hora trabalhada é fornecida pelo usuário.

**Para** cont ← 1 **Até** numEmps **Faça**

**Início**

**Escreva** ("O profissional ", cont, " trabalhou ", horasEmps[cont], " horas") ;

**Fim;**

**Fim.**

Note que: (1) graças aos índices, os vetores podem ser percorridos com o uso de estruturas de repetição e (2) cada elemento dos vetores é tratado como uma variável qualquer.

# Algoritmos com Vetores

**Algoritmo** calculaSalarios01;

**Constantes**

numEmps  $\leftarrow$  50;

**Variáveis**

cont : inteiro;

vlHorTrab : real;

horasEmps : vetor [1..numEmps] de real;

vlHorasEmps: vetor [1..numEmps] de real;

**Início**

**Escreva** ("Quanto custa uma hora trabalhada?") ;

**Leia** (vlHorTrab);

**Para** cont  $\leftarrow$  1 **Até** numEmps **Faça**

**Início**

**Escreva** ("Quantas horas trabalhou o profissional", cont, "?") ;

**Leia** (horasEmps[cont]);

vlHorasEmps[cont]  $\leftarrow$  horasEmps[cont] \* vlHorTrab;

**Escreva** ("O profissional", cont, "ganhará R\$ ", vlHorasEmps[cont]);

**Fim;**

**Fim.**

# Algoritmos com Vetores

**Algoritmo** calculaSalarios01; *Observe que esses vetores são indexados pela mesma variável ("cont"). É que cada elemento de um vetor possui um correspondente no outro. Por isso, esses vetores são chamados de "vetores paralelos".*

**Constantes**  
numEmps ← 50;

**Variáveis**  
cont : inteiro;  
vlHorTrab : real;  
horasEmps : vetor [1..numEmps] de real;  
vlHorasEmps: vetor [1..numEmps] de real;

**Início**  
**Escreva** ("Quanto custa uma hora trabalhada?") ;  
**Leia** (vlHorTrab);  
**Para** cont ← 1 **Até** numEmps **Faça**  
  **Início**  
  **Escreva** ("Quantas horas trabalhou o profissional", cont, "?") ;  
  **Leia** (horasEmps[cont]);  
  vlHorasEmps[cont] ← horasEmps[cont] \* vlHorTrab;  
  **Escreva** ("O profissional", cont, "ganhará R\$ ", vlHorasEmps[cont]) ;  
  **Fim**;  
**Fim**.

*Vetores paralelos: vetores diferentes indexados pela mesma variável ("cont").*



# Algoritmos com Vetores

**Algoritmo** calculaSalarios01; *Observe que esses vetores são indexados pela mesma variável ("cont"). É que cada elemento de um vetor possui um correspondente no outro. Por isso, esses vetores são chamados de "vetores paralelos".*

**Constantes**

numEmps ← 50;

**Var**

cont : inteiro; *chamados de "vetores paralelos".*

horasEmps : vetor [1..numEmps] de real;

vlHorasEmps : vetor [1..numEmps] de real;

**Início**

Escreva ("Quantas horas trabalhou o profissional?");

Leia (vlHorTrab);

Para cont ← 1 Até numEmps *Faça*

**Início**

Escreva ("Quantas horas trabalhou o profissional", cont, "?");

Leia (horasEmps[cont]);

vlHorasEmps[cont] ← horasEmps[cont] \* vlHorTrab;

Escreva ("O profissional", cont, "ganhará R\$", vlHorasEmps[cont]);

**Fim;**

**Fim.**

Utilizando os valores armazenados nos vetores "horasEmps" e "vlHorasEmps", como você faria para calcular quantas horas, em média, trabalharam os profissionais que receberam mais de R\$ 2.000,00?

# Algoritmos com Vetores

**Algoritmo** calculaSalarios02;

**Constantes**

numEmps  $\leftarrow$  50;

**Variáveis**

cont, contAux : inteiro;

vlHorTrab, mediaHrs : real;

horasEmps : vetor [1..numEmps] de real;

vlHorasEmps : vetor [1..numEmps] de real;

**Início**

:

contAux  $\leftarrow$  0;

mediaHrs  $\leftarrow$  0;

**Para** cont  $\leftarrow$  1 **Até** numEmps **Faça**

**Se** vlHorasEmps[cont] > 2000 **then**

**Início**

            mediaHrs  $\leftarrow$  mediaHrs + horasEmps[cont];

            contAux  $\leftarrow$  contAux + 1;

**Fim;**

mediaHrs  $\leftarrow$  mediaHrs / contAux;

**Escreva** ("Média de horas dos que ganharam mais que R\$ 2000 eh ", mediaHrs);

:

# Algoritmos com Vetores

**Algoritmo** calculaSalarios02;

**Constantes**

numEmps  $\leftarrow$  50;

**Variáveis**

cont, contAux : inteiro;

vlHorTrab, mediaHrs : real;

horasEmps : vetor [1..numEmps] de real;

vlHorasEmps : vetor [1..numEmps] de real;

**Início**

⋮  
contAux  $\leftarrow$  0;  
mediaHrs  $\leftarrow$  0;

→ Note que o vetor "vlHorasEmps" é percorrido do início ao fim, para que sejam localizados os índices que correspondem aos trabalhadores que receberam mais de R\$ 2.000,00. Isso é chamado de busca sequencial, pois as posições do vetor são lidas em sequência, uma após a outra.

**Para** cont  $\leftarrow$  1 **Até** numEmps **Faça**  
**Se** vlHorasEmps[cont] > 2000 **then**

**Início**

mediaHrs  $\leftarrow$  mediaHrs + horasEmps[cont];

contAux  $\leftarrow$  contAux + 1;

**Fim;**

mediaHrs  $\leftarrow$  mediaHrs / contAux;

**Escreva** ("Média de horas dos que ganharam mais que R\$ 2000 eh ", mediaHrs);

⋮

# Vetores em Pascal

```
program vetores;  
const tamVetor = 50;  
var  
    vetNums : array [1..tamVetor] of integer;  
    cont     : integer;  
begin  
    randomize;  
    for cont := 1 to tamVetor do //inicializa o vetor  
        vetNums[cont] := 1+random(99); //gera valor aleatório  
    for cont:= 1 to tamVetor do  
        writeln('Pos[', cont, '] = ', vetNums[cont]);  
end.
```

# Vetores em Pascal

```
program vetores;
```

```
const tamVetor = 50;
```

```
var
```

```
    vetNums : array [1..tamVetor] of integer;
```

```
    cont     : integer;
```

```
begin
```

```
    randomize;
```

```
    for cont := 1 to tamVetor do //inicializa o vetor
```

```
        vetNums[cont] := 1+random(99); //gera valor aleatório
```

```
    for cont:= 1 to tamVetor do
```

```
        writeln('Pos[', cont, '] = ', vetNums[cont]);
```

```
end.
```

Em Pascal, é possível estabelecer qual é a faixa dos números que serão utilizados para indexar um vetor. Aqui, a faixa é de 1 a 50, mas poderia ser de 2 a 51, por exemplo.



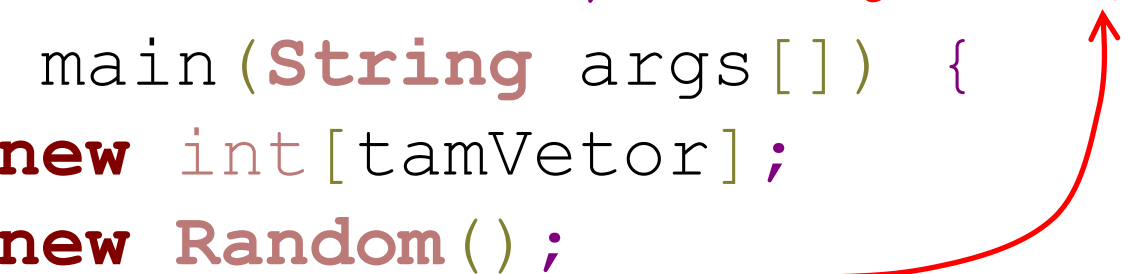
# Vetores em Java

```
import java.util.Random;
public class Vetores {
    static private int tamVetor = 50;
    public static void main(String args[]) {
        int[] vetNums = new int[tamVetor];
        Random rand = new Random();
        for (int cont=0; cont<tamVetor; cont++)
            vetNums[cont] = 1+rand.nextInt(100);
        for (int cont=0; cont<tamVetor; cont++)
            System.out.println("Pos["+cont+"]="+vetNums[cont]);
    }
}
```

# Vetores em Java

```
import java.util.Random;
public class Vetores {
    static private int tamVetor = 50;
    public static void main(String args[]) {
        int[] vetNums = new int[tamVetor];
        Random rand = new Random();
        for (int cont=0; cont<tamVetor; cont++)
            vetNums[cont] = 1+rand.nextInt(100);
        for (int cont=0; cont<tamVetor; cont++)
            System.out.println("Pos["+cont+"]="+vetNums[cont]);
    }
}
```

Note que em Java o primeiro elemento do vetor recebe o índice "0" ao invés de "1".



# Vetores com Mais de Uma Dimensão

```
program vetores;  
const tam = 50;  
var  
    matNums : array [1..tam, 1..tam] of integer;  
    lin, col: integer;  
begin  
    randomize;  
    for lin := 1 to tam do //inicializa a matriz  
        for col:= 1 to tam do  
            matNums[lin,col] := 1+random(99);//valores aleatórios  
    for lin := 1 to tam do //mostra a matriz  
        for col:= 1 to tam do  
            writeln('Pos[', lin, ', ', col, ']' = ', matNums[lin,col]);  
end.
```

# Discussão e Resolução de Exercícios Propostos