**Texto base**

**8**

**Vetores**

Gilberto Alves Pereira

***Resumo***

*Usamos vetores no nosso dia-a-dia. Talvez o nome não seja bem esse. Conhecemos mais como listas. Neste texto é discutido o conceito de vetores, seus componentes e a sua forma de utilização. A grande vantagem dos vetores reside justamente na sua capacidade de armazenar uma grande quantidade de informação na memória. É esse justamente o seu objetivo como ferramenta: sempre que tivermos a necessidade de armazenar uma grande quantidade de informação vamos usar esse recurso. Como exemplificação mostra-se uma implementação de um exemplo usando fluxogramas e um segundo exemplo usando fluxogramas juntamente com uma simulação mostrando como percorrer um vetor. Ao final é mostrada a conversão de um fluxograma usando vetores e funções para Python.*

# 1.1. Introdução

A forma como os dados são organizados tem um impacto significativo sobre a eficácia com que podem ser usados. Uma das maneiras mais óbvias e úteis de organizar os dados é como uma lista ou um vetor de elementos (DIERBACH 2012).

Nós usamos listas no nosso dia-a-dia, fazemos listas de compras, listas de tarefas, listas de convidados (DIERBACH 2012).

Listas também ocorrem na natureza. Nosso DNA é essencialmente uma longa lista de moléculas na forma de uma dupla hélice, encontrada no núcleo de todas as células humanas e todos os organismos vivos. Sua finalidade também é armazenar informações - especificamente, as instruções usadas para construir todas as outras células do corpo - que chamamos de genes (DIERBACH 2012).

Dados os 2,85 bilhões de nucleotídeos que compõem o genoma humano, determinar seu sequenciamento (e, assim, entender nossa composição genética) é fundamentalmente um problema computacional (DIERBACH 2012).

Veremos o uso de listas na programação. O conceito de uma lista é semelhante à nossa noção cotidiana de uma lista. Nós podemos ler (acessar) itens em nossa lista de tarefas, adicionar novos itens, excluir itens e assim por diante (DIERBACH 2012).

Embora em programação existam algumas diferenças entre listas e vetores, devido à semelhança dos conceitos, estaremos usando os dois como sinônimos neste texto para facilitar o entendimento.

# 1.2 O QUE É UM VETOR?

Um vetor é uma **estrutura de dados linear** que mantém a posição dos seus elementos em uma ordem linear. Ou seja, tem o primeiro elemento, o segundo elemento e assim por diante.

Abaixo na Figura 8.1 nós temos um exemplo de uma lista de compras. Note que o primeiro elemento da lista é o Cereal, o segundo é o Leite, o terceiro é a Banana etc.

| ***Lista de compras*** |
| --- |
| 0 - Cereal |
| 1 - Leite |
| 2 - Banana |
| 3 - Maçã |
| 4 - Iogurte |

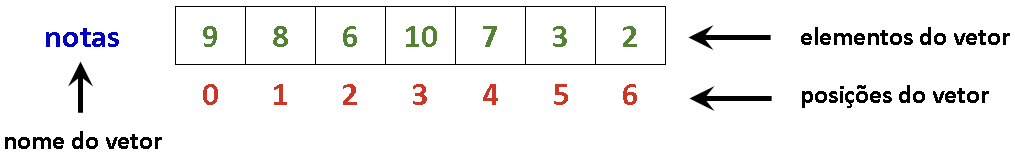
**Figura 8.1 - Exemplo elementos de um vetor.**

**Fonte: Autor**

A ordem em que os elementos aparecem na lista pode ser a ordem em que o autor planeja pegar os itens no mercado, ou pode ser simplesmente a ordem em que ele anotou os itens à medida que foi se lembrando.

# 1.3 VETORES EM PROGRAMAÇÃO

Vetores são variáveis com capacidade de armazenar várias informações. Cada informação é acessada ou armazenada através de sua posição.



**Figura 8.2 - Componentes de um vetor. Fonte: Autor**

No exemplo acima temos que o nome da variável (do tipo vetor) é notas. Esse vetor possui 7 posições. As posições começam de 0 até 6. Os elementos do vetor são os valores que o vetor armazena. Assim:

* O valor do vetor **notas** na posição 0 (zero) é 9.
* O valor do vetor **notas** na posição 3 (três) é 10.

## 1.4 Criando um vetor

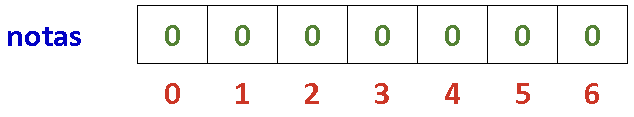
Embora a criação de vetores não seja definida para fluxogramas, vamos usar como padrão a criação de listas em Python 3. Podemos criar um vetor usando um dos comandos a seguir:

notas = 7 \* [0]

ou

notas = [0, 0, 0, 0, 0, 0, 0]

Isso cria a variável **notas** e atribui a ela um vetor de 7 posições, todas com o conteúdo 0 (zero) (Figura 8.3).



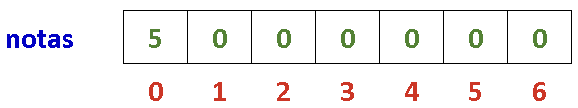
**Figura 8.3 - Resultados dos comandos de criação de listas.**

**Fonte: Autor**

## 1.5 Atualizando o conteúdo de uma posição do vetor

Para alterar o conteúdo de uma posição do vetor iremos usar o operador de atribuição “=”, indicando a posição que receberá o novo valor dentro de colchetes “[” e “]”:

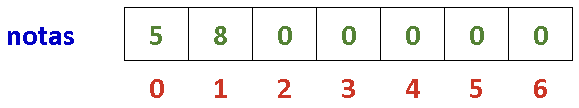
notas[0] = 5



**Figura 8.4 - Exemplo alteração dos elementos de vetor notas[0]=5.**

**Fonte: Autor**

notas[1] = 8



**Figura 8.5 - Exemplo alteração dos elementos de vetor - nota[1]=8.**

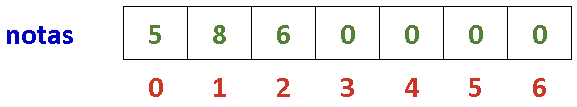
**Fonte: Autor**

Podemos também executar operações com o conteúdo do vetor em expressões:

notas[2] = notas[1] - notas[0] + 3

notas[2] = 8 - 5 + 3

notas[2] = 6



**Figura 8.6 - Resultado das operações no vetor nota.**

**Fonte: Autor**

A posição 2 do vetor notas recebe o valor 6. Pois notas[1] vale 8 e notas[0] vale 5.

## 1.6 Exibindo valores de um vetor

**Figura 8.7 - Exibição de um elemento do vetor.**

**Fonte: Autor**

Podemos exibir um elemento de um vetor indicando a posição a ser exibida (Figura 8.7)

Utilizando o vetor acima o resultado desse comando apresentará na tela:

**valor da nota é 8**

## 1.7 Atualizando o vetor com dados digitados pelo usuário

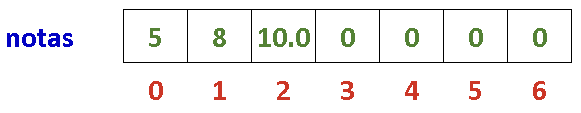
Podemos executar uma entrada de dados diretamente no vetor, bastando para isso indicar a posição do vetor onde o dado será armazenado:



**Figura 8.8 - Comando que recebe do usuário um valor e armazena na posição 2 do vetor notas.**

**Fonte: Autor**

Utilizando o vetor acima, caso o usuário digite o valor 10, o resultado desse comando deixará o nosso vetor como abaixo:

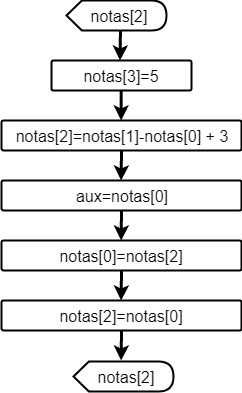
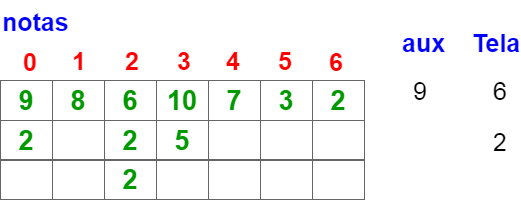


**Figura 8.9 - Resultado da entrada via teclado do valor 10 no vetor notas.**

**Fonte: Autor**

Ou seja, o valor da posição 2 que era 6 foi substituído por 10 (valor digitado pelo usuário)

### Exemplo 1 - Nesse exemplo mostramos a simulação de um trecho de fluxograma. A medida em que os comandos são executados as alterações nas posições do vetor notas, na variável aux e na tela são usadas para preencher a tabela abaixo:



**Figura 8.10 - Trecho de fluxograma e tabela com a simulação mostrando alteração dos dados das variáveis. Fonte: Autor**

## 1.8 Executando operações em vetores

**Alteração**: Para alterarmos um vetor temos que alterar cada um dos valores em cada posição. Não é possível alterar todos os valores de uma vez. Cada posição do vetor funciona como se fosse uma variável independente e dizemos que vamos percorrer o vetor.

Percorrer os elementos de um vetor é acessar os elementos um a um, começando pelo primeiro e terminando no último elemento do vetor, posição a posição.

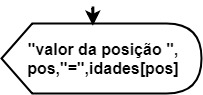
Esse percurso pelos elementos de uma sequência pode ser feito através de um laço de repetição.

**Exibição**: Veja abaixo um exemplo para a exibição dos valores de um vetor:

|  | |  | **Idades** | | | | | | | | --- | --- | --- | --- | --- | --- | --- | --- | | **posição** | **[0]** | **[1]** | **[2]** | **[3]** | **[4]** | **[5]** | **[6]** | | **valores** | 9 | 8 | 6 | 10 | 7 | 3 | 2 | |  |  |  |  |  |  |  |  |  | **pos** | **pos < 7** | **tela** | | --- | --- | --- | | 0 | 0 < 7 V | Valor da posição 0 = 9 | | 1 | 1 < 7 V | Valor da posição 1 = 8 | | 2 | 2 < 7 V | Valor da posição 2 = 6 | | 3 | 3 < 7 V | Valor da posição 3 = 10 | | 4 | 4 < 7 V | Valor da posição 4 = 7 | | 5 | 5 < 7 V | Valor da posição 5 = 3 | | 6 | 6 < 7 V | Valor da posição 6 = 2 | | 7 | 7 < 7 F |  | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

**Figura 8.11 - Fluxograma de exibição de um vetor e sua simulação. Fonte: Autor**

A execução deste bloco irá fazer com que a variável **pos** (que é a **variável de controle do laço**) assuma o valor da primeira posição do vetor (no caso 0 - zero) e, a cada execução do bloco de **instruções do laço**, assuma o valor da posição seguinte até que todas as posições sejam percorridas. Ao final, **pos** irá assumir o valor 7 e o laço será encerrado, pois o vetor não possui uma posição 7.

O comando de exibição na tela será repetido 7 vezes e para cada execução o valor da variável pos será diferente, começando em 0 e terminando em 6, e será impresso na tela o valor correspondente a cada posição. Dessa forma, quando **pos** vale 0 (zero) o valor exibido será aquele correspondente à primeira posição (posição zero) do vetor idades, e assim sucessivamente até a última posição.

**Atribuição**: Veja abaixo um exemplo para a atribuição de valores a um vetor a partir da leitura de tais valores do teclado:

Vamos supor que o usuário digite os seguintes valores: 3, 6, 23, 99, 1, 25, 81

|  | |  | **Idades** | | | | | | | | --- | --- | --- | --- | --- | --- | --- | --- | | **posição** | **[0]** | **[1]** | **[2]** | **[3]** | **[4]** | **[5]** | **[6]** | | **valores** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |  | 3 | 6 | 23 | 99 | 1 | 25 | 81 | |  |  |  |  |  |  |  |  |  | **pos** | **pos < 7** | **tela** | | --- | --- | --- | | 0 | 0 < 7 V | Digite o valor da posição 0 | | 1 | 1 < 7 V | Digite o valor da posição 1 | | 2 | 2 < 7 V | Digite o valor da posição 2 | | 3 | 3 < 7 V | Digite o valor da posição 3 | | 4 | 4 < 7 V | Digite o valor da posição 4 | | 5 | 5 < 7 V | Digite o valor da posição 5 | | 6 | 6 < 7 V | Digite o valor da posição 6 | | 7 | 7 < 7 F |  | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

**Figura 8.12 - Fluxograma da entrada de dados em um vetor e sua simulação. Fonte: Autor**

De uma maneira análoga a exibição, a execução deste bloco irá fazer com que a variável **pos** (que é a **variável de controle do laço**) assuma, a cada execução do bloco de **instruções do laço**, o valor da posição seguinte até que todas as posições sejam percorridas.

O comando  será repetido 7 vezes e, para cada execução, o valor da variável pos será diferente, começando em 0 e terminando em 6. Dessa forma quando **pos** vale 0 o valor digitado será armazenado na posição 0 do vetor **idade** e assim sucessivamente até a posição 6, pois ao receber o valor 7, o laço é encerrado pois não é possível acessar a posição 7 do vetor (ela não existe).

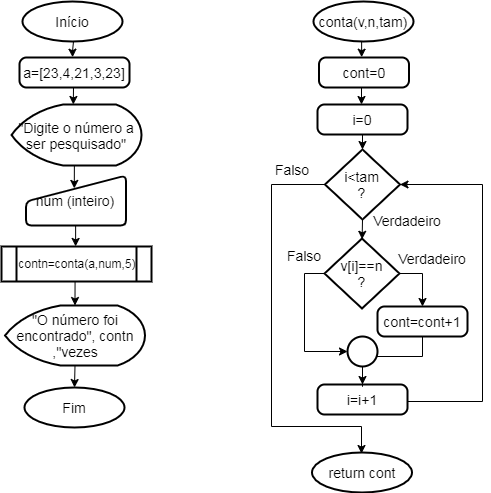
### Exemplo 2: Desenhar um fluxograma de uma função que recebe como parâmetro um vetor e um valor. A função deve retornar a quantidade de vezes que o valor aparece no vetor. Desenhar também um fluxograma que crie o vetor v=[23,4,21,3,23] , solicite um valor para o usuário e chame a função e exiba o valor retornado pela função.

Quando trabalhamos com funções é importante deixar bem claro qual parte do código fica na função e qual não fica. Nesse caso, vamos criar uma função que tem o objetivo de contar quantas vezes um determinado valor aparece em um vetor. Vamos passar por parâmetro o vetor **V** e o valor **n**. A grande vantagem de se fazer isso é que essa mesma função pode ser usada para outros valores e outros valores. Basta para isso passarmos essas informações como parâmetros da função. A reusabilidade é uma característica importante quando criamos uma função. O fato da função poder ser reutilizada em várias outras situações é um predicado importante para as funções. Pensando nisso vamos utilizar mais um parâmetro **tam**. Ele será usado para passarmos para a função a informação da quantidade de posições do vetor que estamos considerando para a nossa procura. Apesar de um vetor ter 100 posições, por exemplo, não precisamos utilizar todas as suas posições, podemos utilizar apenas 10 posições das 100 disponíveis.

Vamos primeiramente entender o funcionamento da nossa função **conta**. Nessa função vamos usar uma variável contadora **cont** para armazenar a quantidade de valores **n** encontrados no vetor **V**. Como todas as variáveis contadoras fazemos inicialmente sua inicialização em 0 (cont=0). Em seguida implementamos um loop utilizando uma variável de controle **i** de maneira que inicie valendo 0 - que é a primeira posição do vetor, até n-1 (usando i<tam) - que é a última posição do vetor.

Dentro desse loop usamos uma estrutura de seleção simples que tem a função de verificar se o valor da posição do vetor é igual ao valor procurado **V[i]==n** para cada uma das posições do vetor. Temos que lembrar que **i** é uma variável que na primeira iteração (passo) do loop vale 0 depois 1 e assim sucessivamente até atingir seu valor máximo (**tam**). Dessa forma, na primeira iteração fazemos o teste **V[0]==n**, pois **i** vale 0, depois **V[1]==n** e assim por diante. Assim, em cada iteração do loop testamos uma posição diferente. Caso o teste seja verdadeiro, incrementamos a variável **cont** através do comando **cont=cont+1** indicando e contabilizando que encontramos o valor na posição **i**. Assim ao final do loop a variável cont armazenará a quantidade de vezes que o valor foi encontrado no vetor. Por fim, o comando **return cont** devolve o valor **cont** ao comando que invocou a função no programa.

Perceba que na função não criamos o vetor e nem pedimos nenhuma informação ao usuário. Veja que isso é feito no programa. Observe que no fluxograma do programa criamos o vetor **a=[23,4,21,3,23]** (que será usado na função) e solicitamos a digitação do valor a ser encontrado **num**. Em seguida a função conta é invocada e são passados por parâmetro o vetor, o valor a ser procurado e o tamanho do vetor **contn=conta(a,num,5).** Perceba que nessa instrução a variável **contn** recebe o valor retornado pela função conta (a frequência de vezes que o valor foi encontrado no vetor). Finalmente a variável **contn** é exibida juntamente com uma mensagem.



**Figura 8.13. Fluxograma do exemplo 2 - contagem de quantas vezes um valor aparece em um vetor.**

**Fonte Autor.**

### 

### 1.9. Conversão de Fluxograma para Python

Abaixo podemos ver o exemplo 2 acima implementado em Python:

| **Fluxograma** | **Python** |
| --- | --- |
|  | |

**Figura 8..14. Esquema do fluxograma e seu respectivo código em Python referentes ao exemplo do uso da variável contadora.**

**Fonte Autor.**

**2.0. Referências**

DIERBACH, C. “Introduction to Computer Science Using Python: A Computational Problem-Solving Focus”1st Edition, New York: Wiley, 2012.