**Introdução**

Este documento aborda uma estrutura muito comum na programação web, o vetor. Você saberá aqui o que ele é, como funciona, sua serventia e exemplos de uso.

**O que é um vetor?**

Um vetor é um tipo de variável que ocupa várias posições na memória, o que permite o armazenamento de diversos valores nele. Ele é constituído de vários **elementos** de **chaves** e **valores** (lembre-se dessas 3 palavras). Para que, dentre esses diversos **valores**, se possa encontrar o valor desejado ou armazenar um valor na chave correta, é necessário informar a respectiva **chave**. Aqui segue a sintaxe padrão:



Perceba que, como as variáveis simples, você precisa declará-la abaixo de **var**, informando seu nome e tipo. As novidades aqui são:

* usar a palavra **vetor** para indicar que se trata de um vetor, e;
* indicar entre colchetes 2 valores (**chaveinicio** e **chavefim**), que determinam a primeira e última chaves do array, e, consequentemente, informando a quantidade de elementos.

É importante ressaltar que no VisuAlg:

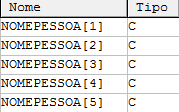
* Um vetor pode ser de qualquer tipo, porém não pode ser multitipo[[1]](#footnote-0);
* Você pode criar quantos vetores precisar em um programa, usando em cada um o tipo desejado;
* As chaves de um vetor podem ser apenas numéricas[[2]](#footnote-1);
* A quantidade de posições de um vetor é fixa[[3]](#footnote-2).

**Obs.: A essa altura do campeonato, acreditamos não ser mais necessário esclarecer que escrever de outra maneira essa sintaxe não terá resultado positivo!**

Para um exemplo prático, veja o código abaixo:

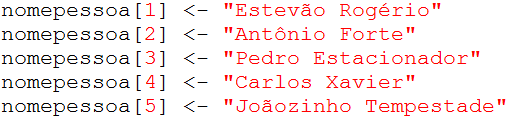


Perceba que, nesse caso, foi usado o nome **nomepessoa** e tipo **caractere**. Além disso, foi dito que a **chaveinicio** é **1** e a **chavefim** é **5**, o que representa que esse vetor terá 5 elementos compostos de chave e valor, onde suas chaves serão [1], [2], [3], [4] e [5]. Veja a imagem abaixo, que representa as posições que o programa disponibilizaria na memória:



Explicando a imagem acima, **nomepessoa** é o *nome* da variável e os valores em colchetes são as *chaves*, de 1 a 5, pois foram esses os valores informados dentro dos colchetes na declaração deste vetor.

Repare, assim, que com esse vetor poderíamos salvar na memória 5 nomes. Para fazer isso, precisamos indicar a chave na qual desejamos guardar a informação. Para preencher esse vetor *estaticamente*, podemos fazer assim:



O que deixaria a memória da seguinte maneira:



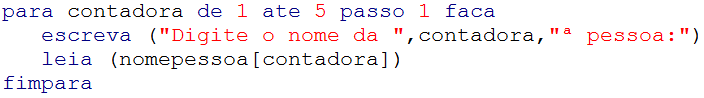
Para mostrar qualquer um desses valores, como já dito no início, é preciso informar a chave que contém a informação que queremos. Veja abaixo:



Qual nome apareceria na tela? O que fazer para mostrar todos os nomes na tela, um abaixo do outro? É bem óbvio, certo? Esperamos que sim, mas se não for assim tão óbvio, pense, releia o documento, lembrando bem das três palavras que lhe pedimos pra lembrar desde o início do documento, e você entenderá. Se ainda assim, você não entender isso como óbvio, chame um ORIENTADOR (sim, um orientador, não um colega de sala).

**Usar um vetor é vantajoso em relação a criar várias variáveis?**

Sim, e muito. No exemplo anterior, preenchemos um vetor estaticamente, ou seja, sem interferência do usuário. Veja agora como preencher o mesmo vetor de maneira *dinâmica* (interagindo com o usuário), usando uma estrutura de repetição:

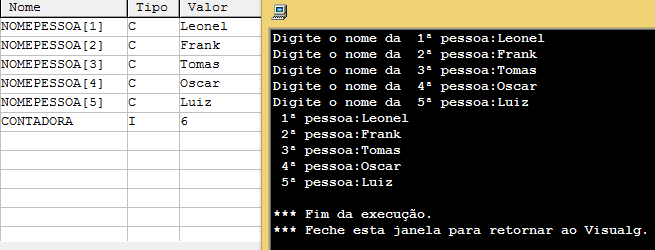


A única novidade para você deve ser a terceira linha, o **leia**. O raciocínio que você deve usar para entendê-la é quase o mesmo que você faria para explicar a segunda linha. Na segunda linha, usamos a variável contadora pelo valor que ela contém, pois na primeira execução do para-faca ela terá valor **1**, na segunda terá valor **2**, e assim por diante, e como você quer mostrar “**1ª**”, “**2ª**”, etc., você a utiliza no escreva. É justamente pelo mesmo motivo que a usamos no leia: na primeira vez que for executado o para-faca ela terá o valor **1**, então o programa gravará o que for digitado em **nomepessoa[1]**, e assim por diante.

Agora que você sabe disso, construa o restante do algoritmo e faça esse teste de mesa você mesmo. Para o teste de mesa, na parte da memória, crie o vetor e suas posições em forma de tabela, como no exemplo abaixo:

| nomepessoa | [1] | [2] | [3] | [4] | [5] |
| --- | --- | --- | --- | --- | --- |
| Estevão Rogério | Antônio Forte | Pedro Estacionador | Carlos Xavier | Joãozinho Tempestade |

Além disso, faça uma estrutura de repetição parecida com a anterior, mas agora para *mostrar* os 5 nomes digitados, na ordem em que foram digitados. Segue um exemplo do programa após sua execução:

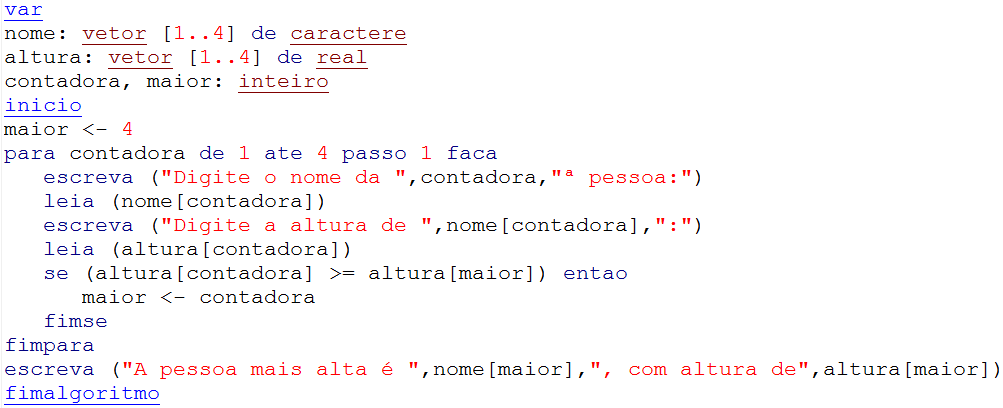


Fica muito mais simples programar assim, né?

**Quando utilizá-lo?**

A grande sacada de um vetor é utilizá-lo quando você precisar guardar várias informações em comum (vários nomes, vários documentos, várias “alturas”, etc.), permitindo que ao invés de criar *5 variáveis* chamadas **nomepessoa1**, **nomepessoa2**, **nomepessoa3**, **nomepessoa4** e **nomepessoa5**, você crie apenas *um vetor* chamado **nomepessoa** com 5 posições na memória.

Além disso, em algumas atividades anteriores, você teve que ficar sobrescrevendo os valores em uma só variável, pois cada valor lido era salvo em uma mesma variável (como no exemplo do aluno mais alto feito com o para-faca). Com o uso de um vetor, você pode guardar todos os nomes e alturas em dois vetores (um para cada informação), e para “saber o mais alto”, basta gravar o valor da contadora em uma variável separada. O único porém é que, como já dissemos, o vetor tem um número fixo de posições, portanto, por exemplo, *não poderíamos pedir ao usuário quantos alunos existem*, pois o vetor é criado antes disso. Como contraponto, sabemos que devemos executar a repetição por quantas chaves possuir o vetor, e como já temos o vetor criado, sabemos quantas vezes a repetição deve ser executada. Por completo, ele ficaria mais ou menos assim:



Faça o teste de mesa e tente entender esse código. Por fim, pense: qual a necessidade da linha 6 (maior <- 4)? Qual a lógica dela? Há algum alternativa viável para ela?

**Sugestões finais:**

Tente alterar o 1º programa, o dos 5 nomes, para fazer as seguintes ações:

* Mostrar os nomes na ordem inversa;
* Pedir ao usuário que ele escolha a pessoa da qual ela quer ver o nome.

1. Em diversas linguagens é permitido que os elementos de um array tenham valores de tipos diferentes. [↑](#footnote-ref-0)
2. A grande maioria das linguagens permite vetores com chaves não numéricas. [↑](#footnote-ref-1)
3. Boa parte das linguagens permitem que um vetor tenha um tamanho dinâmico, ou seja, que aumente ou diminua se for preciso. [↑](#footnote-ref-2)