Vetores

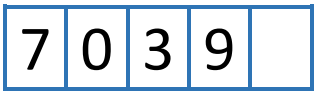
Hoje vamos conversar sobre os vetores, mas antes vamos falar sobre a aplicação. Imagine termos a necessidade de guardar, por um breve momento, sem sobrescrever, a idade de todos os alunos da nossa turma de FUP. Como temos mais de 80 matriculados, teríamos que ter mais que 80 variáveis. Nossa! Só de pensar em declarar e administrar tantas variáveis, já nos preocupa, né? Bom, trago uma ótima notícia! Há um tipo de variável em C que representa não só um elemento e sim, um conjunto de elementos. Legal?

Bom, quando declaramos uma variável informamos seu tipo e nome, e o sistema operacional armazena essa variável na memória. E sua representação se dar por:

int x=7;

Aprendemos que ao declararmos x como inteiro, ela só armazenará informações desse tipo e guardará apenas um valor a cada alteração que sofrer, no exemplo a cima x está armazenando o valor 7.

Bom, como havíamos conversado no início desse documento, em C podemos ter um conjunto de elementos do mesmo tipo, que são armazenados lado a lado na memória:



int x[5] = {7,0,3,9};

Para esse tipo de variável, que armazena o mesmo tipo de conteúdo, onde sua memória é alocada lado a lado uma da outra, de forma homogênea, chamados de Vetores (ou do inglês *Arrays*). Neste último exemplo temos uma variável x, inteira, com 5 espaços de memória alocados, em que na declaração guarda o conjunto de valores {7,0,3,9}.

A sintaxe de sua declaração é dada por:

```
<tipo> nome_variavel [<tamanho>];
```

Podendo, assim como as demais variáveis simples, que conhecemos até agora, inicializar ou não seu conteúdo no ato da declaração:

```
<tipo> nome_variavel [<tamanho>] = {<conjunto_de_valores>};
```

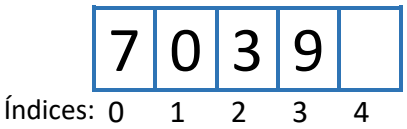
<conjunto_de_valores> são elementos, do mesmo tipo, separados por vírgula, que serão armazenados na variável vetor de maneira sequencial.

Após a variável ser declarada só podemos armazenar e manipular apenas um número determinado de elementos, correspondente ao <tamanho>, dessa variável. E esse <tamanho> deverá ser maior que zero, e uma vez definido, não poderá ter seu valor alterado. Ou seja, uma vez declarado o <tamanho> de um vetor, ele só poderá armazenar essa quantidade de elementos.

Bom, vamos conhecer um pouco mais sobre a estrutura de um vetor?

Estrutura

Fazendo uso do mesmo exemplo inicial:



int x[5] = {7,0,3,9};

Como nossa variável x possui 5 espaços de memória, uma forma de acessá-los é através de índices. Em C, os índices de um vetor começam com valor zero e vão até o valor de <tamanho> -1. Para nosso exemplo, em que x possui tamanho 5, os índices de acesso aos elementos de x vão de 0 a 4 (tamanho-1):

Elemento índice 0 → x[0] → 7

Elemento índice 1 → x[1] → 0

Elemento índice 2 → x[2] → 3

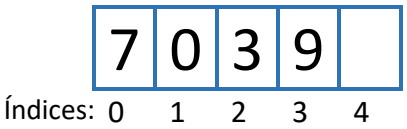
Elemento índice 3 → x[3] → 9

Elemento índice 4 → x[4] → lixo na memória

Bom, e para que serve os índices de um vetor? Vamos entender que eles são úteis no processo de identificação dos elementos, e ao identificarmos podemos atribuir valores.

Atribuição

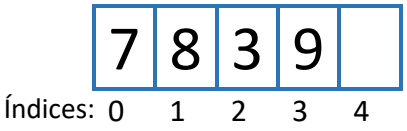
O processo de atribuição de valores pode ocorrer através de índices. Digamos que queremos alterar o segundo elemento da variável x:



x possui como segundo elemento o número zero, localizado no índice 1. Para altera-lo, é bem semelhante as variáveis simples, a diferença é que indicamos o índice que corresponde o elemento:

```
x[1] = 8
```

Agora o vetor x foi ajustado para:



Bom, mas como declaramos um vetor? Sempre deveremos dizer seus elementos iniciais?

Não, vamos lá conhecer o processo de inicialização.

Inicialização

Assim como as variáveis simples, um vetor pode ser inicializado no ato da declaração. Podemos inicializar sem expressarmos nenhum valor para eles:

```
int main(void){
    float y[3];
    int x[5];
    char v[4];
}
```

E assim como as variáveis comuns, os valores armazenados nos exemplos y, x e v são inicializados com lixo da memória. y contendo 3 espaços de memória, x possuindo 5 e v com 4.

Porém também podemos inicializar um vetor com um conjunto de elementos definido:

```
int main(void){
    float y[3] = {1.0,2.6,0.6};
    int x[5] = {7,0,3,9};
    char v[4] = {'F', 'U', 'P', '!'};
}
```

E ao dizer expressamente quais elementos estão nos nossos vetores no ato da declaração, podemos "ocultar" o tamanho:

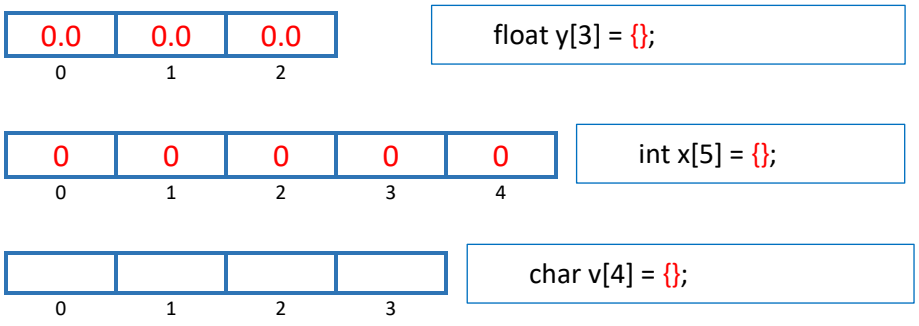
```
int main(void){
    float y[] = {1.0,2.6,0.6};
    int x[] = {7,0,3,9,8};
    char v[] = {'F', 'U', 'P', '!'};
}
```

Ao indicarmos com quais elementos o vetor deve ser inicializado, o compilador já realiza o processo de reserva de espaços sequenciais de memória para cada variável.

Ou podemos também iniciar um vetor com valores "zerados"

```
int main(void){
    float y[3] = {};
    int x[5] = {};
    char v[4] = { };
}
```

Isso representa que os vetores serão inicializados:



Bom, e quanto a manipulação?

Manipulação

A manipulação se dá através de um "percurso" sobre o conjunto de elementos. Como os elementos podem ser acessados através dos índices de um vetor, que sempre começa de zero e vai até o <tamanho>-1. Então, vamos pensar: qual estrutura é baseada em contagem e pode nos apoiar na contagem dos índices? Hm, isso mesmo, o for. Mas também nada nos impede de usar o while e do-while.

Vamos percorrer nosso vetor de exemplo x? Bora lá!

```
#include <stdio.h>
int main(){
	int x[5]={7,0,3,9};
	int i;

	for(i=0; i<5;i++)
		printf("%d \n", x[i]);
}
```

Desta forma, conseguimos com que i varie de 0 a 4, exatamente os índices do nosso vetor x. Com esta operação imprimimos os valores contidos em x (7,0,3,9). Vamos solicitar esses valores por scanf para incluir no nosso vetor?

```
#include <stdio.h>
int main(){
	int x[5];
	int i;

	for(i=0; i<5;i++){
		printf("Inform um valor: ");
		scanf("%d", &x[i]);
	}
}
```

Hm, então quer dizer que o uso do scanf para os vetores é igual para uma variável comum? Isso mesmo!

```
& <nome_variável>[<índice>]
```

E operações, será que podemos fazer operações aritméticas e lógicas com vetores? Sim, podemos:

```
#include <stdio.h>
int main(){
	int x[5]={5,6,3,8,1};
	int i;

	for(i=0; i<5;i++)
		x[i] = x[i]+1;
}
```

Os elementos de x que eram {5,6,3,8,1}, passaram a ser {6,7,4,9,2}, pois a cada iteração do for acrescemos um aos elementos no índice i do vetor x.

E entre vetores podermos realizar operações aritméticas? Sim, olha só:

```
#include <stdio.h>
int main(){
        int x[5]={5,6,3,8,1};
        int y[5]={1,2,3,4,5};
        int r[5]={};
        int i;

        for(i=0; i<5;i++)
                r[i] = x[i] +y[i];

}
```

Neste exemplo, o vetor r inicializado com zeros {0,0,0,0,0} recebe a soma de cada elemento i do vetor x com o vetor y, ou seja, r terá como valores {6,8,6,12,6}.

Mas e as operações lógicas? Vamos comparar quais os elementos de x são pares?

```
#include <stdio.h>
int main(){
        int x[5]={5,6,3,8,1};


        for(i=0; i<5;i++)
                if(x[i] % 2 == 0)
                        printf("O elemento do indice %d eh par",i);

}
```

O resultado desse exemplo trará a indicação dos índices 1 e 3.

Bom, mas podemos comparar vetores? Sim, de índice em índice, como o exemplo:

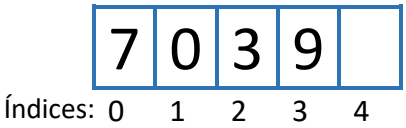
```
#include <stdio.h>
int main(){
        int x[5]={5,6,3,8,1};
        int y[5]={6,5,8,4,7};

        for(i=0; i<5;i++)
                if(y[i] > x[i])
                        printf("O elemento do indice %d de y eh maior que o de x",i);

}
```

Curiosidade

Bom, vimos no início desse material que os vetores são alocados na memória de maneira contínua:



int k[5] = {7,0,3,9};

E que o acesso ao primeiro elemento é dado pelo índice zero. Para esse exemplo k[0]. Ok?

Ótimo!

Então, se dizemos apenas k estamos falando não de todo o conjunto (o vetor), estamos tratando apenas do endereço do primeiro elemento de k, o &k[0]:

```
#include <stdio.h>
int main(){
        int k[5]={5,6,3,8,1};

        printf("%x -- %x", k, &k[0]);

}
```

Ou seja, falar do endereço do primeiro elemento do vetor é a mesma coisa de falar apenas o nome da variável que representa esse vetor.

Bom, o conteúdo do primeiro elemento desse vetor, nós conseguimos acessar por k[0], ok? Mas podemos acessar também por *k:

```
#include <stdio.h>
int main(){
        int k[5]={5,6,3,8,1};

        printf("%d -- %d", *k, k[0]);

}
```

Se k corresponde ao endereço do primeiro elemento e *k corresponde ao conteúdo desse primeiro elemento, isso te faz lembrar alguma coisa?
Hanram! Ponteiros!!!!!

Isso mesmo. Podemos dizer que ***vetores são conjuntos de endereços alocados de maneira homogênea no espaço de memória***. E podemos manipulá-los por meio de ponteiros. Super legal, não é mesmo?

Então fazer *(k + 2), corresponde ao mesmo de acessar o conteúdo do 3º elemento do nosso vetor:

```
#include <stdio.h>
int main(){
	int k[5]={5,6,3,8,1};

	printf("%d -- %d", *(k+2), k[2]);

}
```

Isso nos vai ser muito útil para quando formos trabalhar com vetores e funções, então guarda essa informação que logo em breve vamos utilizar, combinado? E enquanto isso vamos discutir sobre vetores no fórum e praticar bastante nos exercícios.