



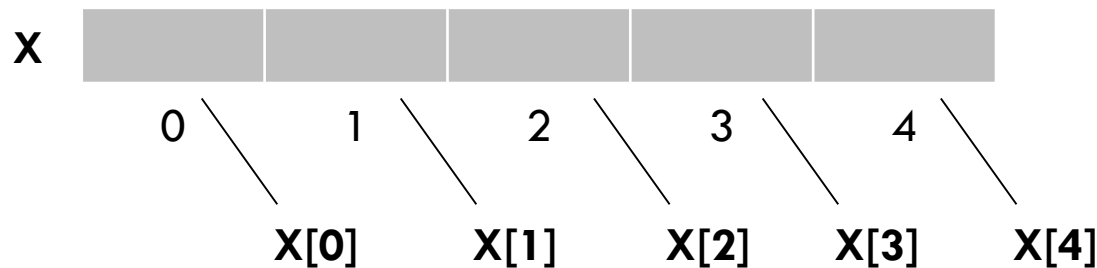
# PROGRAMAÇÃO A

Vetores

# DEFINIÇÃO

**Vetor** também é conhecido como **variável composta homogênea unidimensional**. Isso quer dizer que se trata de um **conjunto de variáveis de mesmo tipo**, que possuem o mesmo **identificador (nome)** e são **alocados sequencialmente na memória**. Como as variáveis têm o mesmo nome, o que as distingue é um **índice** que referencia sua localização dentro da estrutura.

Exemplo:



Acima podemos observar a criação de um **vetor chamado X**, que possui **cinco posições**. Ou seja, foram alocadas cinco posições de memória para armazenamento de números. Essas porções de memória são contíguas, isto é, seus endereços são sequenciais.

# INTRODUÇÃO

- Um **vetor** é um tipo de variável capaz de armazenar um coleção de dados do mesmo tipo. Cada um dos dados armazenados em um vetor, denominado **item** ou **elemento**, é identificado por um **número natural**, a partir de **0**, denominado **índice**.
- Para indicar que uma variável é um vetor, devemos declará-la com o sufixo **[n]**, sendo **n** um valor inteiro positivo que estabelece o tamanho do vetor.
- **Exemplos:**

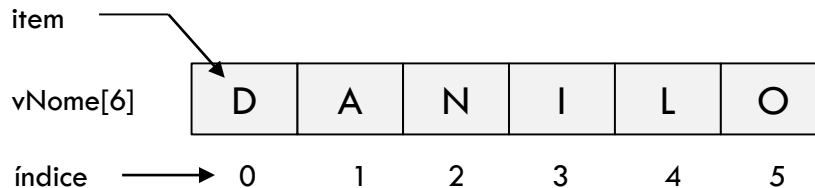
**char** letras[3]; // declaração de um vetor chamado c que pode armazenar até 3 caracteres.

**int** valores[5]; // declaração de um vetor chamado valores que pode armazenar até 5 números inteiros.

**float** precos[10]; // declaração de um vetor chamado precos que pode armazenar até 10 números reais.

# REPRESENTAÇÃO GRÁFICA DE UM VETOR

A declaração `char vNome[6];` cria um vetor com 6 posições, cada uma delas capaz de armazenar um caractere. Na verdade, com esta única declaração, criamos as variáveis `vNome[0]`, `vNome[1]`, ..., `vNome[5]`. Note que, como a indexação inicia-se em 0, o último item de um vetor de tamanho n é armazenado na posição n-1.



// Exemplo de atribuição de valores para o vetor de forma fixa

`vNome[0] = 'D';`

`vNome[1] = 'A';`

`vNome[2] = 'N';`

`vNome[3] = 'I';`

`vNome[4] = 'L';`

`vNome[5] = 'O';`

O que acontecerá se eu tentar atribuir um valor para a variável `vNome[6] = ?`

# EXERCÍCIOS

Se o vetor **V** for igual a:

Elemento	2	6	8	3	10	9	1	21	33	14
Índice:	0	1	2	3	4	5	6	7	8	9

e as variáveis **X = 2** e **Y = 4**, escreva o valor correspondente à solicitação:

a)  $V[X+1]$

d)  $V[X*4]$

g)  $V[X*3]$

i)  $V[8 - V[2]]$

m)  $V[V[0] * V[3]]$

b)  $V[X+2]$

e)  $V[X*1]$

h)  $V[V[X+Y]]$

k)  $V[V[3]]$

n)  $V[X+4]$

c)  $V[X+3]$

f)  $V[X*2]$

i)  $V[X+Y]$

l)  $V[V[V[6]]]$

o)  $V[X*Y]$

# INICIAÇÃO DE VETOR COM TAMANHO CONSTANTE

Um vetor de tamanho *constante* também pode ser iniciado ao ser declarado. Neste caso, os valores iniciais do vetor devem ser indicados entre chaves e separados por vírgulas. Por exemplo, a declaração:

```
char vogais[5] = {'a', 'e', 'i', 'o', 'u'};
```

cria um vetor chamados **vogais**, que armazena as letras 'a', 'e', 'i', 'o' e 'u', nesta ordem; ou seja, a variável `vogais[0]` vale 'a', `vogais[1]` vale 'e', `vogais[2]` vale 'i' e assim por diante. Analogamente, a declaração:

```
float moedas[6] = {1.00, 0.50, 0.25, 0.10, 0.05, 0.01};
```

 cria um vetor chamado **moedas**, que armazena os números reais 1.00, 0.50, 0.25, 0.10, 0.05 e 0.01 nesta ordem.

Se a quantidade de valores iniciais na declaração de um vetor for *menor* que o tamanho deste, as demais posições do vetor são automaticamente zeradas. Assim, por exemplo, a declaração:

```
int vetor[4] = {10, 20};
```

 cria um vetor cujos dois primeiros itens são 10 e 20 e cujos últimos dois itens são iguais a zero. Por outro lado, se a quantidade de valores iniciais é *maior* que o tamanho do vetor, uma mensagem de erro é exibida pelo compilador. Vale ressaltar que vetores de tamanho variável não podem ser iniciados.

# DECLARAÇÃO E ATRIBUIÇÃO DE VALORES

```
1  /*
2     Esse programa mostra as diversas formas de declaração
3     e utilização de vetores em linguagem C.
4  */
5
6  #include <stdio.h>
7
8  int main()
9  {
10     // declaração de vetores
11     char c[] = {'D', 'A', 'N', 'I', 'L', 'O'}; // vetor sem tamanho definido
12     char nome[4] = {'D', 'a', 'n', 'i'}; // vetor com tamanho definido
13     char n[6] = "Danilo"; // vetor com uma cadeia de caracteres
14     char v[3]; // vetor sem valores ainda
15     // char errado[]; // INCORRETO (não funciona)
16
17     // atribuição de valor
18     v[2] = 'n';
19
20     printf("c[0] = %c\n", c[0]);
21     printf("n[1] = %c\n", n[1]);
22     printf("v[0] = %c\n", v[0]);
23     printf("v[2] = %c\n", v[2]);
24     printf("c[8] = %c\n", c[8]);
25     printf("nome[5] = %c\n", nome[5]);
26
27     return 0;
28 }
29
```

exemplo\_slide\_7\_atribuição\_dados\_vetor.c

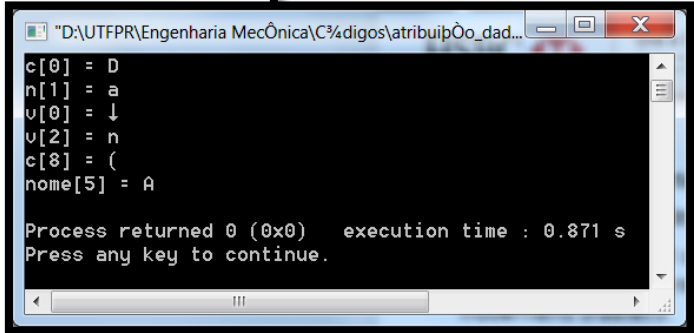
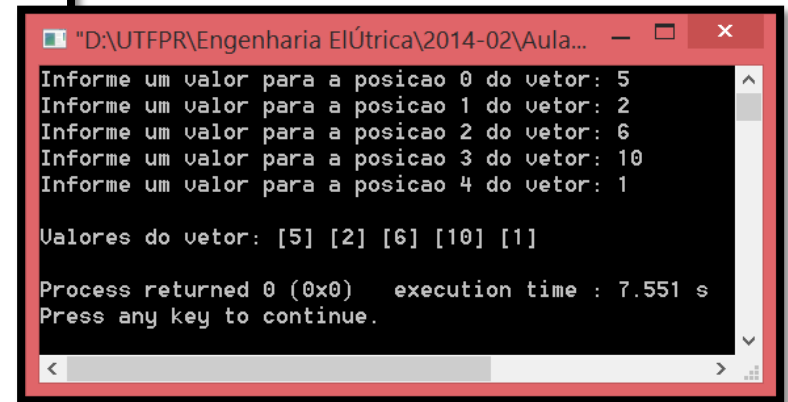


Figura 1 - Exemplo de declaração de vetores e atribuição de valores em linguagem C

# DECLARAÇÃO E ATRIBUIÇÃO DE VALORES

```
1  /*
2     Esse programa pede para a pessoa informar um valor
3     para cada posição de um vetor de cinco números inteiros.
4  */
5
6  #include <stdio.h>
7  #define N 5 // tamanho do vetor
8
9  int main()
10 {
11     // declaração do vetor
12     int v[N];
13
14     // índice do vetor
15     int i;
16
17     // entrada de dados
18     for(i=0; i<N; i++)
19     {
20         printf("Informe um valor para a posicao %i do vetor: ", i);
21         scanf("%i%c", &v[i]);
22     }
23
24     printf("\nValores do vetor: ");
25
26     // saída de dados
27     for(i=0; i<N; i++)
28         printf("[%i] ", v[i]);
29
30     printf("\n");
31
32     return 0;
33 }
34
```



```
"D:\UTFPR\Engenharia Elétrica\2014-02\Aula...
Informe um valor para a posicao 0 do vetor: 5
Informe um valor para a posicao 1 do vetor: 2
Informe um valor para a posicao 2 do vetor: 6
Informe um valor para a posicao 3 do vetor: 10
Informe um valor para a posicao 4 do vetor: 1

Valores do vetor: [5] [2] [6] [10] [1]

Process returned 0 (0x0)   execution time : 7.551 s
Press any key to continue.
```

exemplo\_slide\_8\_atribuição\_dados\_vetor2.c

Figura 2 - Exemplo de declaração de vetores e leitura de valores em linguagem C por meio do teclado (realizada pela própria pessoa).

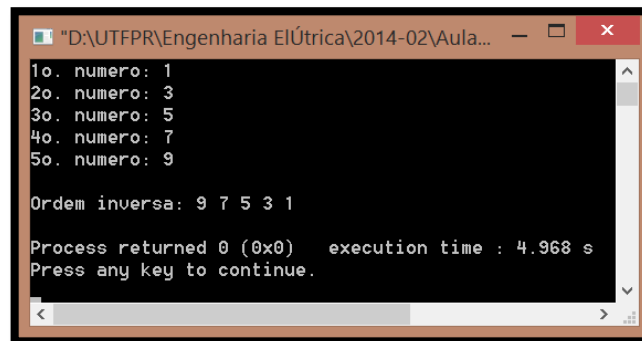


# VETORES - ESTUDO DE CASO

**Problema 1:** Leia uma sequência de cinco números e exiba-a em ordem inversa.

A lógica desse programa é simples. Basta ir lendo os números e guardando nas posições do vetor, da esquerda para direita; em seguida, após o completo preenchimento do vetor, os itens são acessados da direita para a esquerda e exibidos.

```
1  /* Exibe uma sequência de 5 números em ordem inversa */
2
3  #include <stdio.h>
4  #define N 5 // tamanho do vetor
5
6  int main()
7  {
8      // declaração de variáveis
9      int v[N], i;
10
11     // preenche o vetor
12     for(i=0; i<N; i++)
13     {
14         // entrada de dados
15         printf("10. numero: ", i+1);
16         scanf("%i&&c", &v[i]);
17     }
18
19     // saída de dados
20     printf("\nOrdem inversa: ");
21
22     // exibe os elementos do vetor
23     for(i=N-1; i>=0; i--)
24     {
25         // saída de dados
26         printf("%i ", v[i]);
27     }
28
29     printf("\n");
30
31     return 0;
32 }
```



```
"D:\UTFPR\Engenharia Elétrica\2014-02\Aula... - [x]
10. numero: 1
20. numero: 3
30. numero: 5
40. numero: 7
50. numero: 9

Ordem inversa: 9 7 5 3 1

Process returned 0 (0x0)   execution time : 4.968 s
Press any key to continue.
```

exemplo\_slide\_9\_ordem\_inversa.c

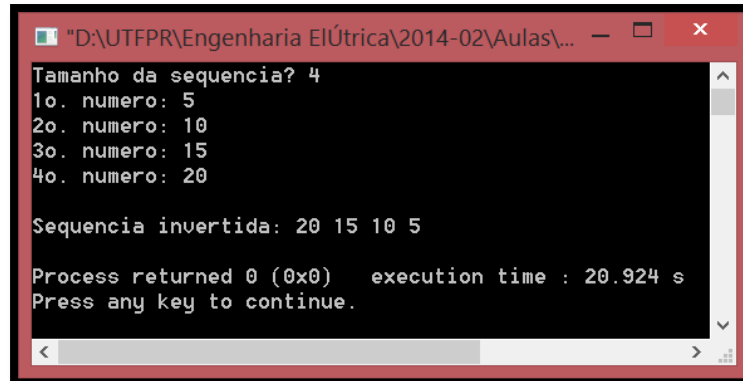
Figura 3 - Exemplo de programa em C que resolve o problema citado

# VETORES - ESTUDO DE CASO

**Problema 2:** Leia uma sequência de  $n$  números e exiba-a em ordem inversa.

De acordo com o padrão **ISO** (*International Organization for Standardization*), o tamanho de um vetor também pode ser indicado por uma variável.

```
1  /* Exibe uma sequencia de n numeros em ordem inversa */
2
3  #include <stdio.h>
4
5  int main()
6  {
7      // declaração de variáveis
8      int n, i;
9
10     // entrada de dados
11     // pergunta o tamanho do vetor
12     printf("Tamanho da sequencia? ");
13     scanf("%i%c", &n);
14
15     int v[n]; // criar o vetor com tamanho igual ao valor de n
16
17     // entrada de dados
18     for(i=0; i<n; i++)
19     {
20         // preenche o vetor
21         printf("%io. numero: ", i+1);
22         scanf("%i%c", &v[i]);
23     }
24
25     // saída de dados
26     printf("\nSequencia invertida: ");
27
28     // exibe os elementos do vetor
29     for(i=n-1; i>=0; i--)
30     {
31         // saída de dados
32         printf("%i ", v[i]);
33     }
34
35     printf("\n");
36
37     return 0;
38 }
```



```
"D:\UTFPR\Engenharia Elétrica\2014-02\Aulas\...
Tamanho da sequencia? 4
1o. numero: 5
2o. numero: 10
3o. numero: 15
4o. numero: 20

Sequencia invertida: 20 15 10 5

Process returned 0 (0x0)   execution time : 20.924 s
Press any key to continue.
```

exemplo\_slide\_10\_ordem\_inversa.c

Figura 4 - Exemplo de programa em C que resolve o problema citado

# EXERCÍCIOS

exercicio1\_slide\_11.c

exercicio2\_slide\_11.c

exercicio3\_slide\_11.c

exercicio4\_slide\_11.c

exercicio5\_slide\_11.c

exercicio6\_slide\_11.c

1. Leia um vetor com 10 números inteiros, mostre a quantidade de números pares e ímpares existentes e crie um novo vetor com o mesmo tamanho do vetor original mas do tipo float para armazenar a raiz quadrada dos números lidos. Apresente os resultados na tela.
2. Leia um vetor com 10 caracteres e mostre a quantidade de vogais e consoantes do vetor.
3. Leia um vetor com 10 números reais (float) e mostre o maior e menor número lido e também a média dos números lidos.
4. Leia dois vetores com 10 números do tipo inteiro cada um. Crie mais 4 vetores para armazenar as operações aritméticas básicas a serem realizadas entre os vetores criados. Armazene o resultado da adição em um vetor, o da subtração em outro e assim por diante. Apresente os resultados na tela.
5. Leia um vetor com 5 números inteiros e mostre o fatorial de cada um deles.
6. Leia um número inteiro qualquer e armazene a tabuada do mesmo em um vetor com 10 posições. Apresente os resultados na tela.

# REFERÊNCIAS BIBLIOGRÁFICAS

- ASCENCIO, A. F. G.; CAMPOS, E. A. V. D. **Fundamentos da Programação de Computadores: Algoritmos, Pascal, C/C++ (Padrão ANSI) e Java.** 3. ed. São Paulo: Pearson Education do Brasil, 2012. 569 p.
- FORBELLONE, A. L. V.; EBERSPACHER, H. F. **Lógica de Programação: A construção de algoritmos e estruturas de dados.** 3. ed. São Paulo: Prentice Hall, 2005. 218p.
- PEREIRA, S. D. L. **Algoritmos e Lógica de Programação em C: Uma abordagem didática.** 1. ed. São Paulo: Érica, 2010. 190 p.