

Aula 7: Vetores

Disciplina: Fundamentos de Programação

Prof. Luiz Olmes

olmes@unifei.edu.br



Nas aulas anteriores...

▶ **O QUE JÁ ESTUDAMOS?**

- ▶ Algoritmos.
- ▶ Linguagem C.
- ▶ Variáveis, operadores e tipos.
- ▶ Estruturas de controle condicionais.
- ▶ Estruturas de controle iterativas.

▶ **OBJETIVOS:**

- ▶ Vetores.

Motivação

- ▶ Variáveis são capazes de armazenar um único valor.
- ▶ Sempre que um novo valor é atribuído, o valor anterior é perdido.
 - ▶ Isso ocorre porque cada variável está associada a uma única posição de memória.
 - ▶ Dentro dela, é possível armazenar apenas um valor do tipo especificado.
- ▶ Para armazenar mais de um valor, é preciso usar mais de uma variável.

Motivação

- ▶ Variáveis são capazes de armazenar um único valor.
- ▶ Sempre que um novo valor é atribuído, o valor anterior é perdido.
 - ▶ Isso ocorre porque cada variável está associada a uma única posição de memória.
 - ▶ Dentro dela, é possível armazenar apenas um valor do tipo especificado.
- ▶ Para armazenar mais de um valor, é preciso usar mais de uma variável.
- ▶ **Exemplo:** ler a nota de 3 alunos e mostrar as notas que são maiores que a média das notas.

Motivação

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      double media, n1, n2, n3;
5.      printf("Digite as 3 notas: ");
6.      scanf("%lf %lf %lf", &n1, &n2, &n3);
7.
8.      media = (n1 + n2 + n3) / 3.0;
9.
10.     if (n1 > media)
11.         printf("Nota: %lf\n", n1);
12.     if (n2 > media)
13.         printf("Nota: %lf\n", n2);
14.     if (n3 > media)
15.         printf("Nota: %lf\n", n3);
16.
17.     return 0;
18. }
```

Imagine o mesmo programa,
mas para 100 alunos.

Motivação

```
1. #include <stdio.h>
2. int main()
3. {
4.     double media, n1, n2, n3;
5.     printf("Digite as 3 notas: ");
6.     scanf("%lf %lf %lf", &n1, &n2, &n3);
7.
8.     media = (n1 + n2 + n3) / 3.0;
9.
10.    if (n1 > media)
11.        printf("Nota: %lf\n", n1);
12.    if (n2 > media)
13.        printf("Nota: %lf\n", n2);
14.    if (n3 > media)
15.        printf("Nota: %lf\n", n3);
16.
17.    return 0;
18. }
```

100 variáveis:
uma para cada aluno.

Imagine o mesmo programa,
mas para 100 alunos.

scanf gigantesco: 100 leituras.
Ou 100 scanf separados.

Motivação

```
1. #include <stdio.h>
2. int main()
3. {
4.     double media, n1, n2, n3;
5.     printf("Digite as 3 notas: ");
6.     scanf("%lf %lf %lf", &n1, &n2, &n3);
7.
8.     media = (n1 + n2 + n3) / 3.0;
9.
10.    if (n1 > media)
11.        printf("Nota: %lf\n", n1);
12.    if (n2 > media)
13.        printf("Nota: %lf\n", n2);
14.    if (n3 > media)
15.        printf("Nota: %lf\n", n3);
16.
17.    return 0;
18. }
```

100 variáveis:
uma para cada aluno.

Imagine o mesmo programa,
mas para 100 alunos.

scanf gigantesco: 100 leituras.
Ou 100 scanf separados.

Expressão gigantesca
para o cálculo da média.

Motivação

```
1. #include <stdio.h>
2. int main()
3. {
4.     double media, n1, n2, n3;
5.     printf("Digite as 3 notas: ");
6.     scanf("%lf %lf %lf", &n1, &n2, &n3);
```

100 variáveis:
uma para cada aluno.

Imagine o mesmo programa,
mas para 100 alunos.

scanf gigantesco: 100 leituras.
Ou 100 scanf separados.

100 comandos if:
um para cada variável.

Expressão gigantesca
para o cálculo da média.

```
7.     (n1 + n2 + n3) / 3.0;
8.
9.
10.    if (n1 > media)
11.        printf("Nota: %lf\n", n1);
12.    if (n2 > media)
13.        printf("Nota: %lf\n", n2);
14.    if (n3 > media)
15.        printf("Nota: %lf\n", n3);
16.
17.    return 0;
18. }
```

100 comandos printf:
um para cada variável.

Motivação

```
1. #include <stdio.h>
2. int main()
3. {
4.     double media, n1, n2, n3;
5.     printf("Digite as 3 notas: ");
6.     scanf("%lf %lf %lf", &n1, &n2, &n3);
7.
8.     media = (n1 + n2 + n3) / 3.0;
9.
10.    if (n1 > media)
11.        printf("Nota: %lf\n", n1);
12.    if (n2 > media)
13.        printf("Nota: %lf\n", n2);
14.    if (n3 > media)
15.        printf("Nota: %lf\n", n3);
16.
17.    return 0;
18. }
```

Imagine o mesmo programa,
mas para 100 alunos.

Solução inviável!

Variáveis Compostas Homogêneas

- ▶ **Variáveis compostas homogêneas**, também chamadas de **arrays**, são estruturas de dados na forma de um **grupo** de variáveis dispostas em posições contíguas de memória.
- ▶ Todos os valores de um array possuem, obrigatoriamente, o **mesmo tipo**.
- ▶ Os objetos armazenados em um array são chamados de **elementos** ou **componentes** do array.
- ▶ Um elemento específico de um array é acessado através de um **índice**.
- ▶ Os arrays que possuem **uma única** dimensão são chamados de **vetores**, enquanto que os arrays com **duas** dimensões são chamados de **matrizes**.

Vetores

- ▶ A ideia de um vetor é criar um conjunto de variáveis do mesmo tipo utilizando apenas um nome.
- ▶ Sintaxe: `tipo nome[tamanho];`

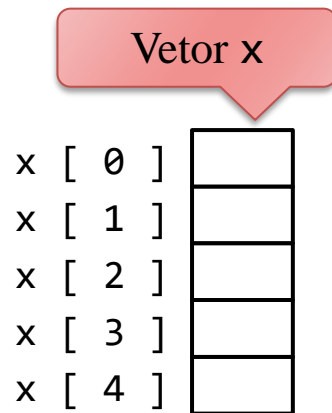
Vetores

- ▶ A ideia de um vetor é criar um conjunto de variáveis do mesmo tipo utilizando apenas um nome.

▶ Sintaxe: `tipo nome[tamanho];`

- ▶ **Exemplo:** declarando um array de inteiros.

```
1. int main()  
2. {  
3.     int x[5];  
4.     ...  
5.     return 0;  
6. }
```



Vetores

- ▶ A ideia de um vetor é criar um conjunto de variáveis do mesmo tipo utilizando apenas um nome.

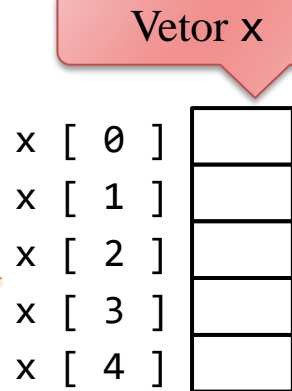
▶ Sintaxe: `tipo nome[tamanho];`

- ▶ **Exemplo:** declarando um array de inteiros.

```
1. int main()  
2. {  
3.     int x[5];  
4.     ...  
5.     return 0;  
6. }
```

Declarando um vetor
com 5 posições

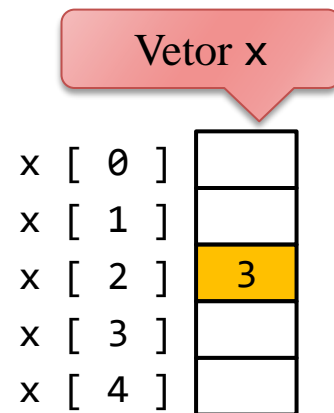
Posições variam entre
zero a 4.



Vetores

- ▶ O acesso ao valor de cada posição do vetor é feito através de um índice.

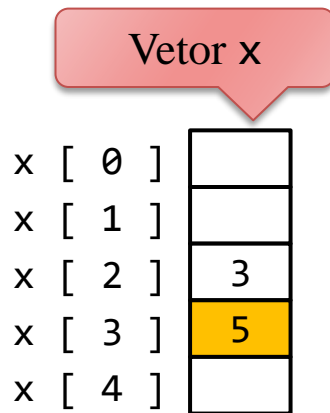
```
1. int main()  
2. {  
3.     int x[5];  
4.  
5.     x[2] = 3;  
6.  
7.  
8.  
9.     return 0;  
10. }
```



Vetores

- ▶ O acesso ao valor de cada posição do vetor é feito através de um índice.

```
1. int main()  
2. {  
3.     int x[5];  
4.  
5.     x[2] = 3;  
6.     x[3] = 5;  
7.  
8.  
9.     return 0;  
10. }
```



Vetores

- ▶ O acesso ao valor de cada posição do vetor é feito através de um índice.

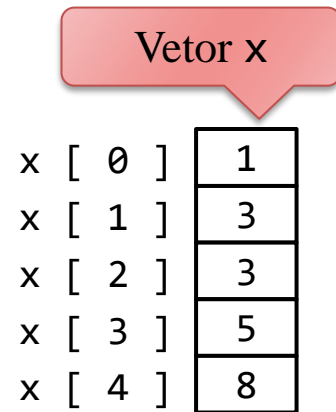
```
1. int main()
2. {
3.     int x[5];
4.
5.     x[2] = 3;
6.     x[3] = 5;
7.     x[4] = x[2] + x[3];
8.
9.     return 0;
10. }
```

Vetor x	
x [0]	
x [1]	
x [2]	3
x [3]	5
x [4]	8

Vetores

- ▶ Vetores podem ser inicializados assim que declarados:

```
1. int main()
2. {
3.     int x[5] = {1, 3, 3, 5, 8};
4.
5.     ...
6.
7.     return 0;
8. }
```



A diagram illustrating the memory layout of the array 'x'. A red speech bubble labeled 'Vetor x' points to a vertical stack of five boxes. To the left of each box is its index, from 0 to 4. The boxes contain the values 1, 3, 3, 5, and 8 respectively.

x [0]	1
x [1]	3
x [2]	3
x [3]	5
x [4]	8

Vetores

- ▶ A iteração sobre os elementos de um vetor utiliza, normalmente, o laço **for**:

```
1. #include <stdio.h>
2. int main()
3. {
4.     int x[5] = {1, 3, 3, 5, 8};
5.     int i;
6.
7.     for (i = 0; i < 5; i++)
8.         printf("%d \n", x[i]);
9.
10.    return 0;
11. }
```

Vetor x

x [0]	1
x [1]	3
x [2]	3
x [3]	5
x [4]	8

Observações

- ▶ Cada posição de um array é uma variável isolada, identificada por um índice.
- ▶ O tempo para acessar qualquer uma das posições do array é o mesmo.
- ▶ Em um array de n posições da linguagem C, a numeração do primeiro índice começa sempre pelo **zero** e termina em $n-1$.
- ▶ É responsabilidade do programador garantir que os limites do array sejam respeitados.
 - ▶ Acessar posições indevidas pode causar erros durante a execução.

Vetores

- ▶ **Exemplo 1:** ler a nota de 100 alunos e mostrar as notas que são maiores que a média.

Vetores

- ▶ **Exemplo 2:** ler em um vetor 8 números inteiros e mostrar os números pares presentes no vetor.

Vetores

- ▶ **Exemplo 3:** ler um vetor de números decimais com 10 posições e mostrar a média dos elementos.

Vetores

- ▶ **Exemplo 4:** ler um vetor de números inteiros com 10 posições e mostrar o maior elemento e a posição em que ele se encontra.

Vetores

- ▶ **Exemplo 5:** ler 6 números em um vetor A. A seguir, ler mais 6 números em um vetor B. Mostrar os números presentes simultaneamente em A e B.

Dúvidas?



Aula 7: Vetores

Disciplina: Fundamentos de Programação

Prof. Luiz Olmes

olmes@unifei.edu.br

