



Universidade Positivo

# Algoritmos de Programação

Vetores

Prof.<sup>a</sup> Mariane Cassenote

mariane.cassenote@up.edu.br

# Conteúdo

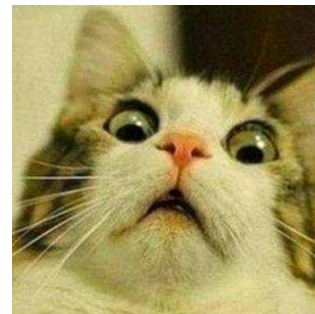


- Vetores

Ao final do componente curricular será possível aplicar estruturas homogêneas de armazenamento com uma dimensão em problemas algorítmicos.

# Por que usar vetores?

- Tipos básicos de dados não são eficientes para uma série de problemas
- Algoritmo para manipular as notas de um estudante
  - 4 avaliações = 4 notas = 4 variáveis
- Algoritmo para manipular as notas de uma turma de 50 estudantes
  - 4 avaliações = 4 notas = 4 variáveis \* 50 estudantes
  - seriam necessárias **200** variáveis só para armazenar as notas
- Utilizar vetores em casos em que tipos básicos de variáveis são ineficientes



# O que são vetores?



- Vetor é uma coleção de elementos de um **mesmo tipo** de compartilham o **mesmo nome** e ocupam posições consecutivas na memória do computador
- Ao invés de armazenar apenas um valor, as variáveis do tipo vetor armazenam vários valores do mesmo tipo simultaneamente
- Vetor é uma estrutura de dados homogênea

# O que são vetores?

- Vetores também são chamados de variáveis indexadas, variáveis compostas, arranjos, matrizes de uma dimensão ou **arrays**
- Mais utilizadas:



Vetores: uma dimensão



Matrizes: duas ou mais dimensões

# O que são vetores?

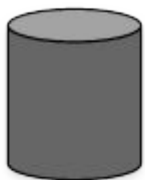


- Um vetor pode ser entendido como um conjunto de dados do mesmo tipo que são acessados através de um **nome comum** e um **índice** (que deve ser um número **inteiro**)
- O elemento com menor índice é o primeiro elemento do vetor e o elemento com maior índice é o último elemento do vetor

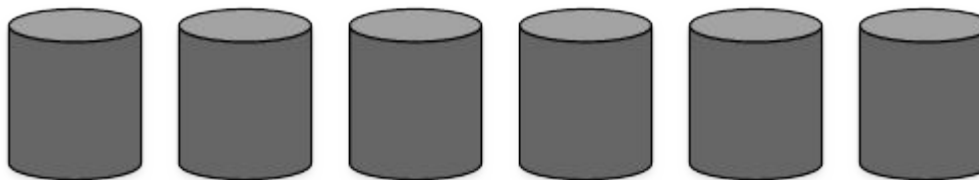
índice	0	1	2	3	4	5	6	7	8	9	10	11

# Vetores

- Um vetor precisa ter bem definidos:
  - o tipo dos valores armazenados
  - seu nome
  - seu tamanho (ou sua capacidade)



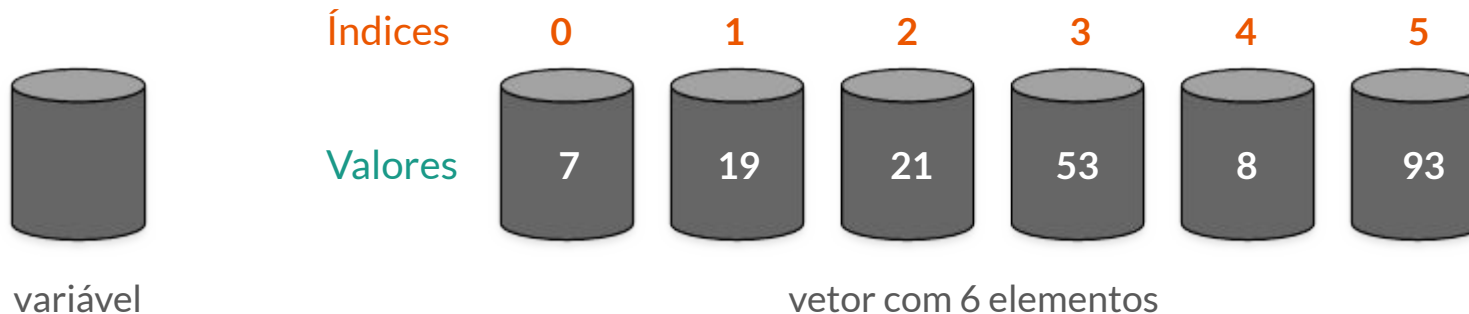
variável



vetor com 6 elementos

# Vetores

- Para acessar o valor armazenado em uma determinada posição de um vetor, precisamos saber exatamente em que posição esse valor se encontra
- A contagem do índice de um vetor começa em zero na maior parte das linguagens de programação





# Vetores em C



- Declaração:
  - `<tipo_vetor> <nome_vetor> [<tamanho_vetor>];`
- Exemplos:
  - `int vet[5];`
  - `float notas[40];`
  - `char email[120];`
  - `int leituras[100];` // vetor de 100 elementos inteiros, sendo leituras[0] o primeiro e leituras[99] o último

# Vetores em C

- Atribuição de valor:
  - `vet[0] = 20;`
  - `int vet[5] = {20, 12, -7, 56, 81};` *// declaração + atribuição (só funcionam juntas)*
  - `scanf("%d", &vet[0]);`
  - `for(int i = 0; i < 5; i++) {`  
    `scanf("%d", &vet[i]);`  
    `}`

	int vet[5];				
índice	0	1	2	3	4
valores	20	12	-7	56	81

# Vetores em C

- Acessar valor:
  - `int a = vet[0];`
  - `printf("%d", vet[0]);`
  - `for(int i = 0; i < 5; i++) {`  
    `printf("%d", vet[i]);`  
    `}`

	int vet[5];				
índice	0	1	2	3	4
valores	20	12	-7	56	81

# Vetores em C

Em ambos dos códigos a impressão na tela é a mesma

```
// uma forma de escrever
#include <stdio.h>

int main () {

    int vet[7]; // declaração

    // atribuições de valores
    vet[0] = 7;
    vet[1] = 35;
    vet[2] = 12;
    vet[3] = 76;
    vet[4] = 28;
    vet[5] = 3;
    vet[6] = 9;

    printf("%d", vet[3]); // acesso ao valor

    return 0;

}
```

```
// outra forma de escrever a mesma coisa
#include <stdio.h>

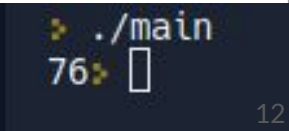
int main () {

    // declaração + atribuição de valores
    int vet[7] = {7, 35, 12, 76, 28, 3, 9};

    // acesso ao valor
    printf("%d", vet[3]);

    return 0;

}
```



```
./main
76
```

# Vetores em C

```
// atribuição de valor a uma posição
#include <stdio.h>

int main () {

    // declaração
    int vet[7];

    printf("Digite um valor para a posicao 3: ");
    scanf("%d", &vet[3]); // atribuição de valor

    printf("Valor da posicao 3: %d", vet[3]); // acesso ao valor

    return 0;
}
```

```
Digite um valor para a posicao 3: 5
Valor da posicao 3: 5
```

# Vetores em C



- Em C, não há operações envolvendo vetores completos
- Operações de atribuição de valores, comparação e outras deverão ser efetuadas elemento por elemento
- Para isso, é necessário utilizar laços de repetição

# Vetores em C

```
// operações entre vetores
#include <stdio.h>

int main() {

    int i;
    int vetX[5];
    int vetY[5] = {7, 1, 4, 9, 3};

    // vetX = vetY; // incorreto!!!

    for (i = 0; i < 5; i++) {
        vetX[i] = vetY[i]; // correto!!!
    }

    for (i = 0; i < 5; i++) {
        printf("VetorX [%d] : %d \n", i, vetX[i]);
    }

    return 0;
}
```

```
➤ ./main
VetorX [0] : 7
VetorX [1] : 1
VetorX [2] : 4
VetorX [3] : 9
VetorX [4] : 3
➤ □
```

# Vetores em C

*// povoar vetor usando laço de repetição*

```
#include <stdio.h>
```

```
int main () {  
    int vet[7], i;  
  
    printf("Digite valores para povoar o vetor: \n");  
  
    for (i = 0; i < 7; i++) {  
        printf("Vetor [%d]: ", i);  
        scanf("%d", &vet[i]);  
    }  
  
    printf("\nVetor povoado: \n");  
  
    for (i = 0; i < 7; i++) {  
        printf("Vetor [%d] : %d \n", i, vet[i]);  
    }  
  
    return 0;  
}
```

❏ ./main

Digite valores para povoar o vetor:

Vetor [0]: 4

Vetor [1]: 9

Vetor [2]: 1

Vetor [3]: 8

Vetor [4]: 2

Vetor [5]: 7

Vetor [6]: 5

Vetor povoado:

Vetor [0] : 4

Vetor [1] : 9

Vetor [2] : 1

Vetor [3] : 8

Vetor [4] : 2

Vetor [5] : 7

Vetor [6] : 5

❏



# Vetores em C

*// cálculo da média de um estudante*

```
#include <stdio.h>
```

```
int main() {
```

```
    float notas[4], soma = 0.0;
```

```
    printf("Digite valores para as notas: \n");
```

```
    for (int i = 0; i < 4; i++) {
```

```
        printf("Nota %d: ", i + 1);
```

```
        scanf("%f", &notas[i]);
```

```
        soma = soma + notas[i];
```

```
    }
```

```
    printf("Media das notas: %f", soma / 4.0);
```

```
    return 0;
```

```
}
```

```
➤ ./main
Digite valores para as notas:
Nota 1: 5.8
Nota 2: 6.1
Nota 3: 2.8
Nota 4: 3.9
Media das notas: 4.65 ➤
```

# Vetores em C



- Se vetores (ou algum elemento específico) não forem inicializados, eles podem conter lixo de memória
- Um tentativa de acesso (leitura ou escrita) fora dos limites do vetor não gera erro de compilação, mas pode retornar lixo de memória
- A linguagem C não tem verificação de limites em vetores. É de responsabilidade do programador fazer essa verificação

# Vetores em pseudocódigo



- Declaração:
  - `<tipo_vetor> <nome_vetor> [<tamanho_vetor>]`
- Exemplos:
  - `inteiro vet[5]`
  - `real notas[40]`
  - `caracter email[120]`
  - `cadeia leituras[100]`
  - `logico ligado[5]`

# Vetores em pseudocódigo

- Atribuição de valor:
  - `vet[0] = 20`
  - `inteiro vet[5] = {20, 12, -7, 56, 81}` *// declaração + atribuição (só funcionam juntas)*
  - `leia(vet[0])`
  - `para(inteiro i = 0; i < 5; i++) {`  
    `leia(vet[i])`  
    `}`

		inteiro vet[5]				
índice		0	1	2	3	4
valores		20	12	-7	56	81

# Vetores em pseudocódigo

- Acessar valor:
  - `inteiro a = vet[0]`
  - `escreva(vet[0])`
  - `para(inteiro i = 0; i < 5; i++) {`  
    `escreva(vet[i])`  
    `}`

inteiro vet[5]					
índice	0	1	2	3	4
valores	20	12	-7	56	81

# Vetores em pseudocódigo

*// atribuição de valor a uma posição com a sintaxe do Portugol WebStudio*

```
programa {  
    funcao inicio() {  
  
        inteiro vet[7]  
  
        escreva("Digite um valor para a posicao 3: ")  
        leia(vet[3])  
  
        escreva("Valor da posicao 3: ", vet[3])  
  
    }  
}
```

```
Digite um valor para a posicao 3: 4  
Valor da posicao 3:4  
Programa finalizado. Tempo de execução: 2497 ms
```

# Vetores em pseudocódigo

```
// atribuição de valores
programa {
    funcao inicio() {

        inteiro vetX[5]
        inteiro vetY[5] = {4, 6, 21, 94, 9}
        inteiro i

        //vetX <- vetY incorreto!!!

        para(i = 0; i < 5; i++) {
            vetX[i] = vetY[i] //correto!!!
        }

        para(i = 0; i < 5; i++) {
            escreva("VetorX [", i, "]: ", vetX[i], "\n")
        }

    }
}
```

```
VetorX [0]: 4
VetorX [1]: 6
VetorX [2]: 21
VetorX [3]: 94
VetorX [4]: 9
```

Programa finalizado. Tempo de execução: 31 ms

# Vetores em pseudocódigo

*// povoar vetor usando laço de repetição*

```
programa {  
    funcao inicio() {  
  
        inteiro vet[7], i  
  
        escreva("Digite valores para povoar o vetor: \n")  
  
        para (i = 0; i < 7; i++){  
            escreva("Posicao", i, ": ")  
            leia(vet[i])  
        }  
  
        escreva("\nVetor povoado:")  
  
        para (i = 0; i < 7; i++){  
            escreva("Posicao", i, ":", vet[i])  
        }  
    }  
}
```

Digite valores para povoar o vetor:

Posicao0: 10

Posicao1: 20

Posicao2: 30

Posicao3: 40

Posicao4: 50

Posicao5: 60

Posicao6: 70

Vetor povoado:

Posicao0:10

Posicao1:20

Posicao2:30

Posicao3:40

Posicao4:50

Posicao5:60

Posicao6:70

Programa finalizado. Tempo de execução: 9663 ms



# Vetores em pseudocódigo

*// cálculo da média de um estudante*

```
programa {  
    funcao inicio() {  
  
        real notas[4], soma = 0  
        inteiro i  
  
        escreva("Digite valores para as notas: \n")  
  
        para(i = 0; i < 4; i++) {  
            escreva("Posicao", i, ": ")  
            leia(notas[i])  
  
            soma = soma + notas[i]  
        }  
  
        escreva("\nMedia das notas:", soma / 4.0)  
    }  
}
```

Digite valores para as notas:

Posicao0: 89.7

Posicao1: 67.4

Posicao2: 98.6

Posicao3: 76.8

Media das notas:83.125

Programa finalizado. Tempo de execução: 10002 ms

# Para praticar



1. Escreva um programa que leia do teclado valores para dois vetores de 4 posições de ponto flutuante, depois subtraia os dois vetores. Ao final da execução deverá ser impresso o resultado na tela.
2. Implemente um programa que exiba os conteúdos dos itens de índice par de um vetor.
3. Escreva um programa que apresente os dados de um vetor na ordem inversa em que eles foram inseridos.

# Para praticar



4. Escreva um programa que crie um vetor de números inteiros de 6 posições e povoe esse vetor com valores lidos do teclado. O programa deverá aceitar somente números pares entre 1 e 20. Ao final, imprima todos os valores salvos no vetor, além da média dos valores, o menor e o maior valor.
5. Escreva um programa que contenha um vetor de 8 valores lidos do teclado. Você deve garantir que os valores lidos sejam inteiros positivos. Em seguida, o programa deverá solicitar um número do teclado e verificar se esse número está contido ou não no vetor.



Universidade Positivo

# Algoritmos de Programação

Vetores

Prof.<sup>a</sup> Mariane Cassenote

mariane.cassenote@up.edu.br