Aula 17: Introdução a vetores

Introdução a Programação

Túlio Toffolo & Puca Huachi
http://www.toffolo.com.br

BCC201 – 2019/1
Departamento de Computação – UFOP

Aula de Hoje

1. Exercícios da aula prática
2. Motivação
3. Vetores
4. Exemplos

Aula de Hoje

Questão 03

Faça um programa que desenha um trapézio retângulo de bases x e y usando asteriscos. O usuário deve informar os valores de x e y, que devem ser **pares** e tais que $x < y$. Certifique-se de que o usuário digitou valores válidos. Exemplo:

```
1  Digite os valores de x e y: 3 6
2  Apenas números pares são aceitos.
3
4  Digite os valores de x e y: 10 8
5  Erro: x deve ser menor que y.
6
7  Digite os valores de x e y: 4 18
8
9          ****
10        ******
11       ********
12      **********
13     ************
14    **************
15   ****************
16  ******************
```

Aula de Hoje

Motivação

Crie um programa que lê **3** notas e, em seguida, imprime quantas notas são maiores que a média.

Motivação

```c
int main()
{
    double nota1, nota2, nota3;
    printf("Digite a nota 1: "); scanf("%lf", &nota1);
    printf("Digite a nota 2: "); scanf("%lf", &nota2);
    printf("Digite a nota 3: "); scanf("%lf", &nota3);

    double media = (nota1 + nota2 + nota3) / 3;
    int contador = 0;
    if (nota1 > media) contador++;
    if (nota2 > media) contador++;
    if (nota3 > media) contador++;

    printf("Número de notas maiores que a média: %d\n", contador);
    return 0;
}
```

Motivação

Agora crie um programa que lê **20** notas e, em seguida, imprime quantas notas são maiores que a média.

Motivação

A solução abaixo é minimamente razoável?

```c
int main()
{
    double nota1, nota2, nota3, ···, nota20;
    printf("Digite a nota 1: "); scanf("%lf", &nota1);
    printf("Digite a nota 2: "); scanf("%lf", &nota2);
    printf("Digite a nota 3: "); scanf("%lf", &nota3);
    ···
    printf("Digite a nota 20: "); scanf("%lf", &nota20);

    double media = (nota1 + nota2 + nota3 + ··· + nota20) / 20;
    int contador = 0;
    if (nota1 > media) contador++;
    if (nota2 > media) contador++;
    if (nota3 > media) contador++;
    ···
    if (nota20 > media) contador++;

    printf("Número de notas maiores que a média: %d\n", contador);
    return 0;
}
```

Motivação

A solução abaixo é minimamente razoável? (**não**, não é...)

```
1  int main()
2  {
3      double nota1, nota2, nota3, ···, nota20;
4      printf("Digite a nota 1: "); scanf("%lf", &nota1);
5      printf("Digite a nota 2: "); scanf("%lf", &nota2);
6      printf("Digite a nota 3: "); scanf("%lf", &nota3);
7      ···
8      printf("Digite a nota 20: "); scanf("%lf", &nota20);
9
10     double media = (nota1 + nota2 + nota3 + ··· + nota20) / 20;
11     int contador = 0;
12     if (nota1 > media) contador++;
13     if (nota2 > media) contador++;
14     if (nota3 > media) contador++;
15     ···
16     if (nota20 > media) contador++;
17
18     printf("Número de notas maiores que a média: %d\n", contador);
19     return 0;
20 }
```

Motivação

Agora crie um programa que lê n notas e, em seguida, imprime quantas notas são maiores que a média. Assuma que $n \leq 1000$.

E agora José?

Motivação

Agora crie um programa que lê n notas e, em seguida, imprime quantas notas são maiores que a média. Assuma que $n \leq 1000$.

E agora José?

Aula de Hoje

1. Exercícios da aula prática
2. Motivação
3. Vetores
4. Exemplos

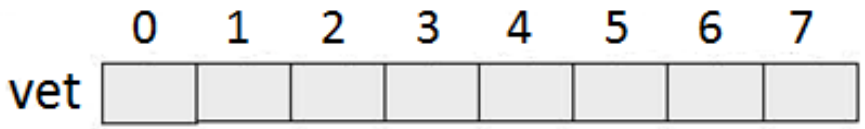
Introdução a vetores

- Um vetor é:
  - uma variável composta **homogênea unidimensional**,
  - formada por uma sequência de valores, todos do mesmo **tipo de dado**,
  - alocados sequencialmente na memória,
  - que são acessados usando um único identificador (nome).

Introdução a vetores

O que distingue os diferentes valores armazenados em um vetor é um **índice**.

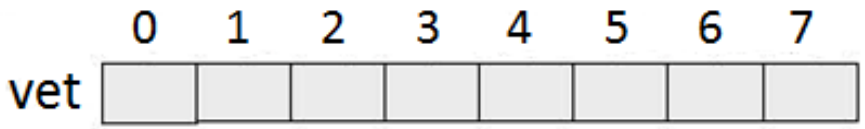
- Exemplo:



Introdução a vetores

O que distingue os diferentes valores armazenados em um vetor é um **índice**.

- Exemplo:



Benefícios do uso de vetores

- Ajudam a manter uma coleção de dados;
- Permitem manter as informações organizadas;
- Permitem operações com o volume de dados neles inseridos;
- Antes precisávamos de n variáveis pra guardar n valores, agora teremos uma **única variável para armazenar n valores**.

Benefícios do uso de vetores

- Ajudam a manter uma coleção de dados;
- Permitem manter as informações organizadas;
- Permitem operações com o volume de dados neles inseridos;
- Antes precisávamos de n variáveis pra guardar n valores, agora teremos uma **única variável para armazenar n valores**.

Benefícios do uso de vetores

- Ajudam a manter uma coleção de dados;
- Permitem manter as informações organizadas;
- Permitem operações com o volume de dados neles inseridos;
- Antes precisávamos de n variáveis pra guardar n valores, agora teremos uma **única variável para armazenar n valores**.

Benefícios do uso de vetores

- Ajudam a manter uma coleção de dados;
- Permitem manter as informações organizadas;
- Permitem operações com o volume de dados neles inseridos;
- Antes precisávamos de n variáveis pra guardar n valores, agora teremos uma **única variável para armazenar n valores**.

Vetores em C/C++

- Conhecidos em C/C++ como **arrays**.
- Correspondem a posições de memória.
- São identificados por um nome.
- Individualizadas por índices.
- Conteúdo do mesmo tipo.

- Resumindo: vetores são **posições de memória** identificadas por um mesmo **nome**, individualizadas por **índices** e cujo conteúdo é do **mesmo tipo**.

Vetores em C/C++

- Conhecidos em C/C++ como **arrays**.
- Correspondem a posições de memória.
- São identificados por um nome.
- Individualizadas por índices.
- Conteúdo do mesmo tipo.

- Resumindo: vetores são **posições de memória** identificadas por um mesmo **nome**, individualizadas por **índices** e cujo conteúdo é do **mesmo tipo**.

Vetores em C/C++

- Conhecidos em C/C++ como **arrays**.
- Correspondem a posições de memória.
- São identificados por um nome.
- Individualizadas por índices.
- Conteúdo do mesmo tipo.

- Resumindo: vetores são **posições de memória** identificadas por um mesmo **nome**, individualizadas por **índices** e cujo conteúdo é do **mesmo tipo**.

Vetores em C/C++

- Conhecidos em C/C++ como **arrays**.
- Correspondem a posições de memória.
- São identificados por um nome.
- Individualizadas por índices.
- Conteúdo do mesmo tipo.

- Resumindo: vetores são **posições de memória** identificadas por um mesmo **nome**, individualizadas por **índices** e cujo conteúdo é do **mesmo tipo**.

Vetores em C/C++

- Conhecidos em C/C++ como **arrays**.
- Correspondem a posições de memória.
- São identificados por um nome.
- Individualizadas por índices.
- Conteúdo do mesmo tipo.

- Resumindo: vetores são **posições de memória** identificadas por um mesmo **nome**, individualizadas por **índices** e cujo conteúdo é do **mesmo tipo**.

Vetores em C/C++

- Conhecidos em C/C++ como **arrays**.
- Correspondem a posições de memória.
- São identificados por um nome.
- Individualizadas por índices.
- Conteúdo do mesmo tipo.

- Resumindo: vetores são **posições de memória** identificadas por um mesmo **nome**, individualizadas por **índices** e cujo conteúdo é do **mesmo tipo**.

Declaração de um vetor

```
<tipo> identificador [<número de posições>];
```

- Tipo: `int`, `float`, `double`, etc.
- Identificador: é o nome da variável que identifica o vetor.
- Número de posições: é o tamanho do vetor!

Exemplos:

```
1 int vetor[5];
```

```
1 double notas[50];
```

```
1 char palavra[20];
```

Declaração de um vetor

`<tipo> identificador [<número de posições>];`

- Tipo: `int`, `float`, `double`, etc.
- Identificador: é o nome da variável que identifica o vetor.
- Número de posições: é o tamanho do vetor!

Exemplos:

```
1 int vetor[5];
```

```
1 double notas[50];
```

```
1 char palavra[20];
```

Declaração de um vetor

```
<tipo> identificador [<número de posições>];
```

- Tipo: `int`, `float`, `double`, etc.
- Identificador: é o nome da variável que identifica o vetor.
- Número de posições: é o tamanho do vetor!

Exemplos:

```
1 int vetor[5];
```

```
1 double notas[50];
```

```
1 char palavra[20];
```

Declaração de um vetor

`<tipo> identificador [<número de posições>];`

- Tipo: `int`, `float`, `double`, etc.
- Identificador: é o nome da variável que identifica o vetor.
- Número de posições: é o tamanho do vetor!

Exemplos:

```
1 int vetor[5];
```

```
1 double notas[50];
```

```
1 char palavra[20];
```

Declaração de um vetor

`<tipo> identificador [<número de posições>];`

- Tipo: `int`, `float`, `double`, etc.
- Identificador: é o nome da variável que identifica o vetor.
- Número de posições: é o tamanho do vetor!

Exemplos:

```
1 int vetor[5];
```

```
1 double notas[50];
```

```
1 char palavra[20];
```

Declaração de um vetor

- Ao declaramos um vetor, os seus elementos não são inicializados.
- Mas é possível atribuir valores iniciais.
- O valores iniciais são colocados entre chaves

Exemplos:

```
1 int vetor[5] = {0, 2, 5, 3, 9};
```

```
1 double notas[5] = {0.0, 10.0, 7.5, 8.5, 9.9};
```

Declaração de um vetor

- Ao declaramos um vetor, os seus elementos não são inicializados.
- Mas é possível atribuir valores iniciais.
- O valores iniciais são colocados entre chaves

Exemplos:

```
1 int vetor[5] = {0, 2, 5, 3, 9};
```

```
1 double notas[5] = {0.0, 10.0, 7.5, 8.5, 9.9};
```

Declaração de um vetor

Importante:

- A quantidade de valores entre chaves não deve ser maior que o número de elementos
- A fim de facilitar a inicialização, C/C++ permite deixar o número de elementos em branco [].
- Neste caso, o compilador vai supor que o tamanho do vetor é igual ao número de valores especificados entre chaves

```
1 int vetor[] = {0, 2, 5, 3, 9}; // tamanho = 5
```

```
1 double notas[] = {10.0, 9.5, 7.5}; // tamanho = 3
```

Declaração de um vetor

Importante:

- A quantidade de valores entre chaves não deve ser maior que o número de elementos
- A fim de facilitar a inicialização, C/C++ permite deixar o número de elementos em branco [].
- Neste caso, o compilador vai supor que o tamanho do vetor é igual ao número de valores especificados entre chaves

```
1 int vetor[] = {0, 2, 5, 3, 9}; // tamanho = 5
```

```
1 double notas[] = {10.0, 9.5, 7.5}; // tamanho = 3
```

Declaração de um vetor

Diferentes forma de declarar um vetor:

```
1  // declaração sem inicializar os valores do vetor (eles terão 'lixo')
2  int v1[3];
3
4  // declaração inicializando os valores do vetor
5  int v2[3] = {0, 2, 5};
6
7  // declaração alternativa inicializando os valores do vetor
8  int v3[] = {0, 2, 5};
```

Declaração de um vetor

Diferentes forma de declarar um vetor:

```
1  // declaração sem inicializar os valores do vetor (eles terão 'lixo')
2  int v1[3];
3
4  // declaração inicializando os valores do vetor
5  int v2[3] = {0, 2, 5};
6
7  // declaração alternativa inicializando os valores do vetor
8  int v3[] = {0, 2, 5};
```

Uso de constantes em vetores

```
1   ...
2   #define TAM_MAX 10
3
4   int main()
5   {
6       double vetor[TAM_MAX];
7
8       // coloca os valores {TAM_MAX, TAM_MAX-1, ..., 1} no vetor
9       for (int i = 0; i < TAM_MAX; i++) {
10          vetor[i] = TAM_MAX - i;
11      }
12      ...
13      return 0;
14  }
```

Uso de constantes em vetores (2)

```
1   ...
2   const int TAM_MAX = 10;
3
4   int main()
5   {
6       double vetor[TAM_MAX];
7
8       // coloca os valores {0, 1, ..., TAM_MAX - 1} no vetor
9       for (int i = 0; i < TAM_MAX; i++) {
10          vetor[i] = i;
11      }
12      ...
13      return 0;
14  }
```

Criando uma cópia de um vetor

```
1   ...
2   #define TAM_MAX 20
3
4   int main()
5   {
6       double vetor[TAM_MAX];
7       for (int i = 0; i < TAM_MAX; i++) {
8           vetor[i] = i;
9       }
10
11      ...
12
13      // copiando cada posição do vetor
14      double copia[TAM_MAX];
15      for (int i = 0; i < TAM_MAX; i++) {
16          copia[i] = vetor[i];
17      }
18  }
```

Aula de Hoje

1. Exercícios da aula prática
2. Motivação
3. Vetores
4. Exemplos

Exemplo 1

Ler 5 notas de alunos e, em seguida, calcular e imprimir a maior nota.

```c
#define TAM 5

int main()
{
    double notas[TAM];

    // lendo as notas
    for (int i = 0; i < TAM; i++) {
        printf("Digite a nota %d: ", i+1);
        scanf("%lf", &notas[i]);
    }

    // obtendo a maior nota
    double maiorNota = 0; // inicializando com a menor nota possível
    for (int i = 0; i < TAM; i++) {
        if (notas[i] > maiorNota) {
            maiorNota = notas[i];
        }
    }

    printf("Maior nota: %lf\n", maiorNota);
    return 0;
}
```

Exemplo 2

Ler 10 números inteiros do teclado para depois imprimir os números na ordem inversa de leitura.

```c
#define TAM 10

int main()
{
    int numeros[TAM];

    // lendo os números
    for (int i = 0; i < TAM; i++) {
        printf("Digite o nro %d: ", i+1);
        scanf("%d", &numeros[i]);
    }

    // imprimindo em ordem inversa
    for (int i = TAM - 1; i >= 0; i--) {
        printf("%d ", numeros[i]);
    }
    printf("\n");

    return 0;
}
```

Exemplo 3

Ler 10 notas de alunos e imprimir quantas tem valor superior à média.

```c
#define TAM 10

int main()
{
    double notas[TAM], soma = 0;

    // lendo as notas (e acumulando valores para calcular a média)
    for (int i = 0; i < TAM; i++) {
        printf("Digite a nota %d: ", i+1);
        scanf("%lf", &notas[i]);
        soma += notas[i];
    }

    // calculando quantas notas são maiores do que a média
    double media = soma / TAM;
    int contador = 0;
    for (int i = 0; i < TAM; i++) {
        if (notas[i] > media)
            contador++;
    }

    printf("%d notas superam a média\n.", contador);
    return 0;
}
```

Exemplo 4

Ler n notas de alunos e imprimir quantas tem valor superior à média.
Assuma que $n \leq 1000$.

```c
#define TAM 1000

int main()
{
    int n;
    double notas[TAM], soma = 0;

    // perguntado ao usuario o nro de notas
    printf("Digite o nro de notas: ");
    scanf("%d", &n);

    // lendo as notas (e acumulando valores para calcular a média)
    for (int i = 0; i < n; i++) {
        printf("Digite a nota %d: ", i+1);
        scanf("%lf", &notas[i]);
        soma += notas[i];
    }

    // calculando quantas notas são maiores do que a média
    double media = soma / TAM;
    int contador = 0;
    for (int i = 0; i < n; i++) {
        if (notas[i] > media)
            contador++;
    }

    printf("%d notas superam a média\n.", contador);
    return 0;
}
```

Perguntas?