Estrutura de Dados

Conceitos Básicos e Vetores

Profa. Ana Cristina dos Santos

email: ana.csantos@sp.senac.br

Tópicos da Aula

- Tipos abstratos de dados
- Contiguidade física
- Armazenamento sequencial de dados
- Variáveis compostas homogêneas
- Manipulação de vetores

Estruturas de Dados

- As estruturas de dados têm o objetivo de **armazenar**, **manipular**, **organizar** e **gerenciar** dados na memória principal do computador
- O objetivo principal dessa disciplina é analisar as melhores alternativas para manipular eficientemente os dados de uma aplicação computacional

Estruturas de Dados

- As **Estruturas de Dados** diferem umas das outras pelo **relacionamento** e **manipulação** de seus dados e são intimamente ligadas aos algoritmos que as manipulam.
- **Relacionamento lógico entre dados**: se os dados estão organizados em ordem crescente ou de forma hierárquica.
- **Manipulação:** As Estruturas de Dados incluem operações para a manipulação de seus dados. Exemplos de operações: criação da Estrutura de Dados, inclusão de um novo elemento, remoção de um elemento, acesso a um elemento, destruição da Estrutura de Dados.

Estruturas de Dados

- **Diferentes tipos de Estrutura de Dados** são adequadas a **diferentes tipos de aplicações** e algumas são altamente especializadas, destinando-se a algumas tarefas específicas.
- Por exemplos, as **B-trees** são particularmente indicadas para a implementação de banco de dados, enquanto que a implementação de compiladores geralmente requer o uso de **tabela de dispersão** para a busca de identificadores.

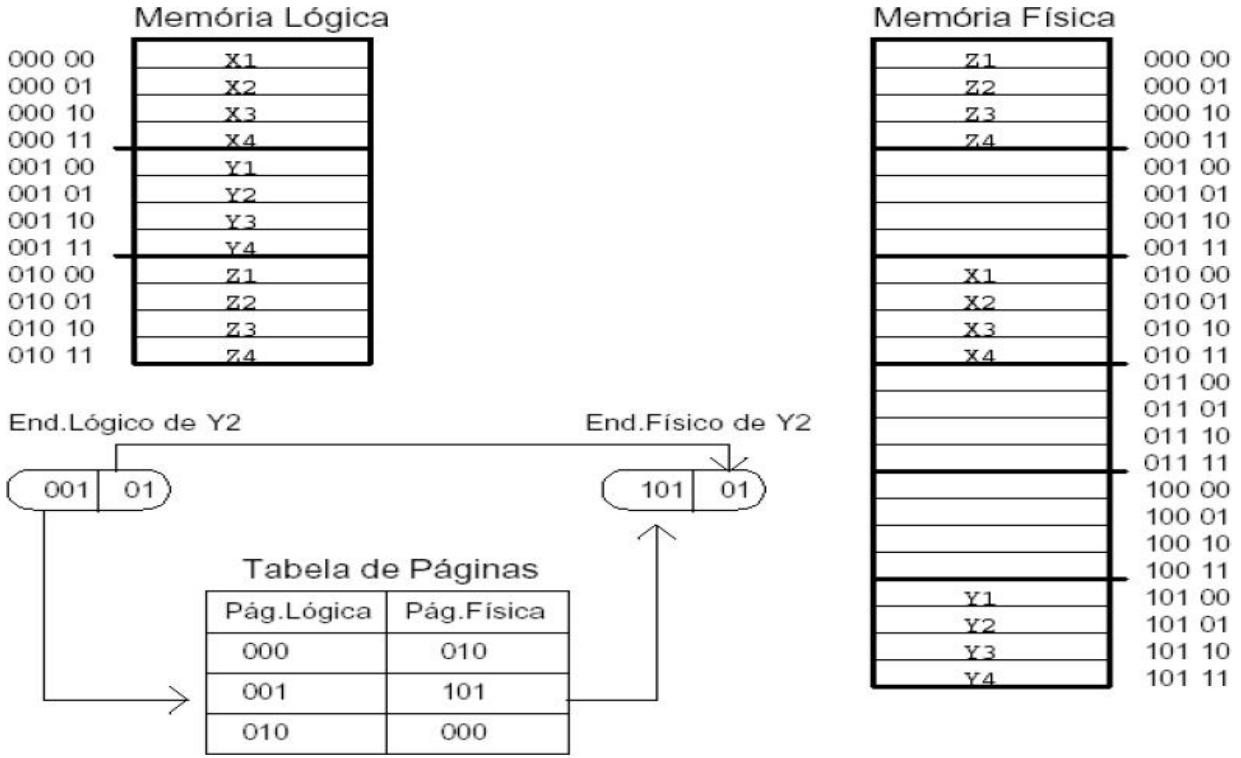
Estruturas de Dados

- Nesse semestre, vamos trabalhar apenas com as estruturas de dados **lineares** (Listas, Pilhas, Filas) e **não lineares (**Árvores e Grafos)
- Essas estruturas são representadas através de:
    - Estruturas contíguas (contiguidade física)
    - Estruturas encadeadas (contiguidade lógica)

Contiguidade Física

- As estruturas de dados contíguas armazenam os dados computacionais em células adjacentes na memória principal do computador
- Suas principais características:
  - **Acesso direto** ao dado através de um índice
  - Existe a necessidade de **deslocar dados** durante os processos de inserção e remoção de dados
  - Na maioria das vezes, a estrutura contígua contém um **número fixo** de células

Armazenamento Sequencial



Tipos Abstratos de Dados

- Um Tipo Abstrato de Dado (TAD) é um modelo de estruturas de dados que especifica:
  - O tipo dos dados armazenados
  - As operações definidas sobre esses dados
  - Os tipos dos parâmetros dessas operações

Tipos Abstratos de Dados

- Dentre as principais características de um TAD, podemos citar que:
  - um TAD define ***o que*** cada operação faz, mas não ***como*** o faz
  - no Java, um TAD é modelado por uma classe
  - a modelagem de um TAD define os dados que serão armazenados e as operações suportadas pela estrutura

Tipos Abstratos de Dados

- Durante a construção das classes utilizadas nessa disciplina, vamos ter um cuidado especial ao escolher o **tipo de dado** utilizado por cada atributo
- O objetivo inicial é criar várias TAD (**Tipos Abstrato de Dado**) para especificar o conjunto de dados e de operações que podem ser executados sobre a estrutura manipulada

Tipos de dados da Linguagem Java

- Java possui duas categorias de tipos de dados:
  - Tipo Primitivo
  - Tipo de Instância ou Referência
- Tipo Primitivo
  - São tipos de dados predefinidos pela linguagem e correspondem a dados mais simples ou escalares
- Tipo de Instância ou Referência
  - Uma instância é um objeto do tipo definido pela classe

Tipos de dados da Linguagem Java

- Tipos primitivos suportados pela linguagem:

Tipo	Bits	
boolean	1	true ou false
char	16	
byte	8	Números Inteiros
short	16	
int	32	
long	64	
float	32	Números em ponto Flutuante
double	64	

Tipos de dados da Linguagem Java

- Intervalo de valores suportados pelo Java:

Tipo	Intervalo
byte	-128 ~ 127
short	-32.768 ~ 32.767
int	-2.147.483.648 ~ 2.147483.647
long	-922.337.203.685.475.808 ~ 922.337.203.685.475.807
float	-1.4e-45 ~ 3.4e38
double	-4.9e-324 ~ 1.7e308

Números Inteiros

Números em ponto Flutuante

Variáveis Compostas Homogêneas

- Como declarar variáveis compostas homogêneas de tipo primitivo e instância
- Como instanciar uma variável composta homogênea
- Inicialização na declaração de uma variável composta homogênea
- O atributo `length` em vetores

Variáveis Compostas Homogêneas

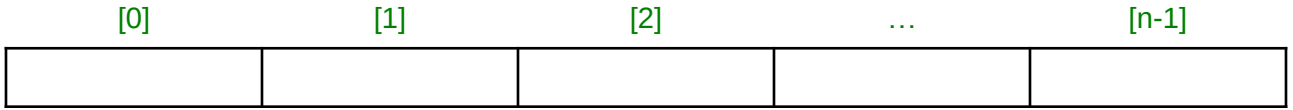
- Variáveis compostas homogêneas são **estruturas de dados** que podem armazenar vários valores de um **mesmo tipo de dado** simultaneamente
    - Enquanto uma variável consegue armazenar um único valor por vez, utilizamos vetores para armazenar um conjunto de valores ao mesmo tempo

Uso de vetores na Linguagem Java

- Em Java os vetores são objetos
    - Isso significa que eles não se comportam como as variáveis e sim como instâncias de classes
    - Por isso, eles precisam ser declarados, instanciados (criados) e iniciados

Uso de vetores na Linguagem Java

- Os vetores (arrays unidimensionais) funcionam como arranjos de valores de um mesmo tipo de dado
    - Você pode imaginar um vetor como uma linha de uma tabela, composta por diversas células
    - Em cada célula é possível armazenar um valor
    - Na Linguagem Java, o primeiro índice do vetor é 0 (zero) e o último é n-1



Uso de vetores na Linguagem Java

- Vamos rever como realizar as quatro operações básicas com vetores:
  - Declarar
  - Instanciar
  - Iniciar
  - Consultar

Uso de vetores na Linguagem Java

- Declarar um vetor:
  - Podemos declarar um vetor de duas formas distintas, porém o resultado é o mesmo em ambos os casos.
    - <tipo>[] <nomeDoVetor>;
    - <tipo> <nomeDoVetor>[];
  - Exemplos:

```
int[] vetor;

int vetor[];
```

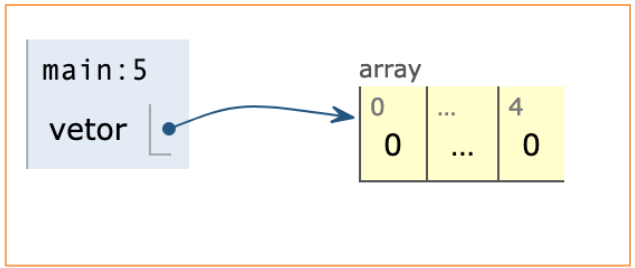
Uso de vetores na Linguagem Java

- Na Linguagem Java os **vetores são objetos**
  - Não basta declarar um vetor para que ele passe a existir
  - O nome do vetor é uma referência a posição inicial na memória onde os valores são armazenados
- A simples declaração **não cria** o vetor, mas apenas uma referência
  - Inicialmente a referência é `null`, ou seja, não aponta para nenhum lugar

Uso de vetores na Linguagem Java

- Instanciar um vetor:
  - Instanciar é o processo pelo qual você aloca um endereço de memória para um objeto
  - A instanciação é a criação do vetor
    - Instanciar um vetor significa lhe atribuir um endereço de memória onde ele possa armazenar seus valores
  - Sintaxe:
    - <nomeDoVetor> = new <tipo>[<posições>];
  - Exemplo:

```
vetor = new int[5];
```



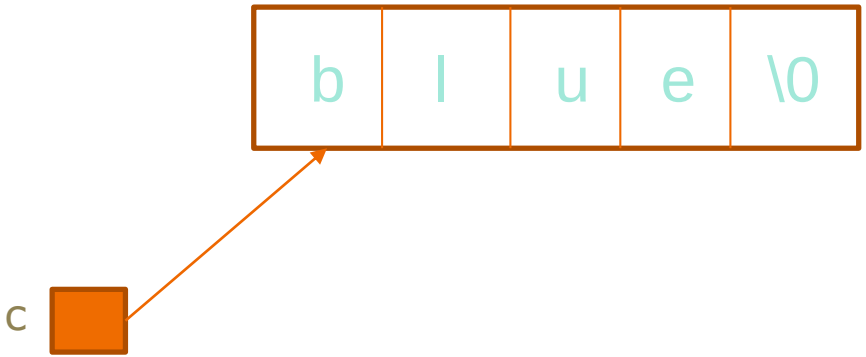
Uso de vetores na Linguagem Java

- Instanciar um vetor:

```
char []c;

c = new char[5];

c[0] = ‘b’;
c[1] = ‘l’;
c[5] = ‘s’;  // Runtime error
```

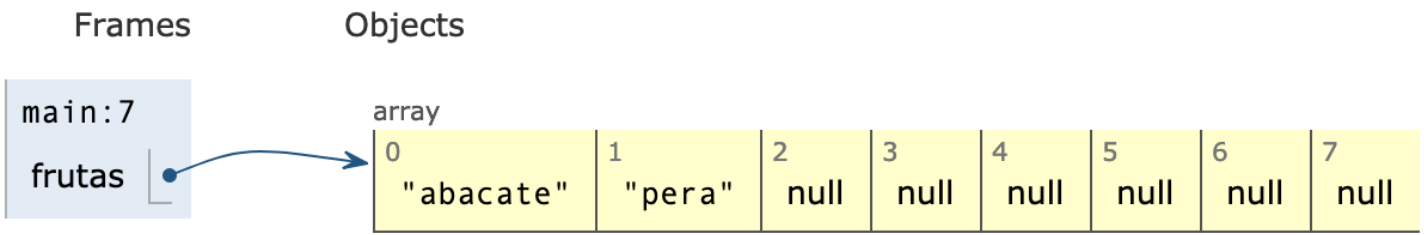


Uso de vetores na Linguagem Java

- Instanciar um vetor:

```
String []frutas = new String[8];

frutas[0] = “abacate”;
frutas[1] = “pera”;
```



Uso de vetores na Linguagem Java

- Declarar e instanciar um vetor:
  - A declaração e a instanciação de um vetor também podem ser feitas em uma única instrução
  - Sintaxe:
    - <tipo>[] <nomeDoVetor> = new <tipo>[<posição>];
    - <tipo> <nomeDoVetor>[] = new <tipo>[<posição>];
  - Exemplo:

```
int[] vetor = new int[5];

int vetor[] = new int[5];
```

Uso de vetores na Linguagem Java

- Iniciar um vetor:
  - A iniciação de um vetor pode ser feita posição a posição após sua declaração e instanciação.
  - Exemplo:

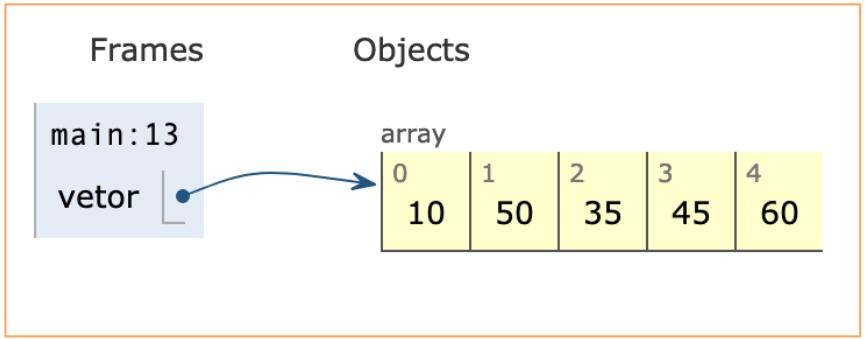
```
vetor[0] = 10;

vetor[1] = 50;

vetor[2] = 35;

vetor[3] = 45;

vetor[4] = 60;
```



Uso de vetores na Linguagem Java

- Consultar um vetor:
  - Abaixo temos a forma de como devemos consultar ou recuperar valores armazenados em um vetor

```
int soma = vetor[0] + vetor[1] + vetor[2];

int nota = vetor[4];

System.out.println("Posição 3 = " + vetor[3]);
```

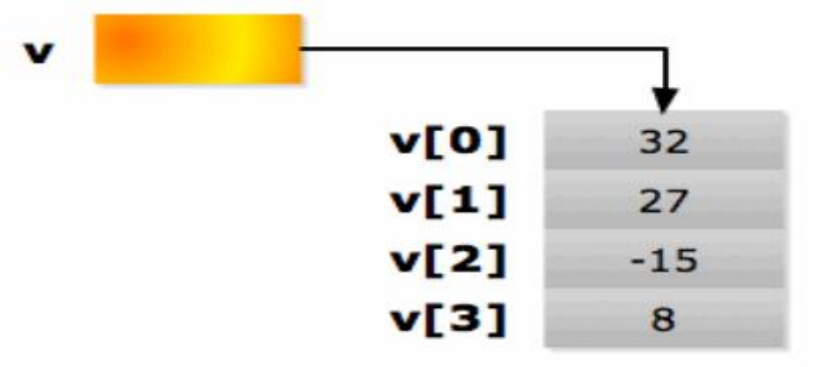
Uso de vetores na Linguagem Java

- Declarar, instanciar e iniciar um vetor:
  - Java possibilita ainda declarar, instanciar e iniciar um vetor em uma única instrução
  - Sintaxe:
    - <tipo>[] <nomeDoVetor>={<valor0>,...,<valorN>};
    - <tipo> <nomeDoVetor>[]={<valor1>,...,<valorN>};
  - Exemplo:
    - int[] vetor = {10, 50, 35, 45, 60};
    - int vetor[] = {10, 50, 35, 45, 60};

Uso de vetores na Linguagem Java

- Declarar, instanciar e iniciar um vetor:
  - Exemplo:

```
int[] v = {32, 27, -15, 8};
```



Uso de vetores na Linguagem Java

- Detectar o tamanho de um vetor
  - Para detectar o tamanho de um vetor a Linguagem Java disponibiliza a propriedade length
  - Sintaxe:
    - int tamanho = notas.length;

Uso de vetores na Linguagem Java

- Percorrer um vetor

```
void imprime(int []v) {
   for(int i = 0; i < v.lenght; i++)
      System.out.println(v[i]);
}

void imprime(int []v) {
   for(int n: v)
      System.out.println(n);
}
```

Problema inicial

- Escreva um programa que leia valores inteiros para um vetor de N posições, onde N ≤ 100. Em seguida, o seu programa deve encontrar e informar a quantidade de elementos do vetor que são maiores que a média dos elementos do vetor.

Problema inicial

- Para realizar a leitura do tamanho do vetor e dos elementos do vetor você pode utilizar a classe Scanner.

```
Scanner sc = new Scanner(System.in);
int N;

N =  sc.nextInt();
```

Problema inicial

- O problema deve ser resolvido considerando os seguintes passos:

1. Ler os valores inteiros para o vetor;
2. Calcular a soma e depois a média dos elementos;
3. Contar quantos elementos no vetor são maiores que a média calculada.

Problema inicial

- Agora tente escrever sua solução utilizando métodos para cada um dos passos abaixos:

1. Ler os valores inteiros para o vetor;
2. Calcular a soma e depois a média dos elementos;
3. Contar quantos elementos no vetor são maiores que a média calculada.

Métodos Estáticos

- Neste exemplo, os métodos devem ser declarados da seguinte forma:
  `public static <retorno> <nome> (<parametros>)`

- Vamos resolver este problema utilizando métodos estáticos

Métodos Estáticos

- Exemplo, método para ler o vetor:

```
public static void leVetor (int vet[]){
  Scanner sc = new Scanner(System.in);
  for( int i=0; i< vet.length; i++){
    System.out.println("Digite o valor para V["+i+"]:");
     vet[i]=sc.nextInt();
  }
}
```

Métodos Estáticos

- O método **`leVetor`** é estático, dessa forma ele pode ser invocado, mesmo que não haja nenhuma instância da classe.
- E para chamar o método poderíamos fazer uma simples invocação do método:

```
// lendo o vetor
leVetor( vet );
```

Exercícios

- Escreva um programa que:
  1. declare uma variável de um tipo vetor de 10 elementos inteiros
  2. leia 10 valores para esta variável
  3. encontre o maior valor do vetor
  4. encontre o menor valor do vetor
  5. Imprima o maior e menor valor encontrados

Exercícios

- Escreva um método conta(a, n, x) que devolve como resultado, o número de elementos iguais a x que aparecem no vetor a de n elementos.

Exercícios

- Escreva um programa que:
  - Leia dois vetores de números inteiros, contendo cada um, 5 elementos.
  - Intercale os elementos destes dois conjuntos formando um novo vetor de 10 elementos.
  - Apresente o novo conjunto, assim obtido.

Exercício

- Escreva um método **`inverter(int a[])`** em Java que receba um vetor com n elementos e devolva o vetor invertido.
  Ex: entrada: [1,3,6,4,5,9]
  saída: [9,5,4,6,3,1]

- Escreva o método principal para testar o método **`inverter(int a[])`**

Exercícios

- Dado n inteiro, 0 < n <= 100, e uma sequência de n números entre 0 e 99, determinar quantos estão entre 0 e 9, entre 10 e 19, ... , entre 90 e 99.

Exercícios

- Escreva um programa que leia dois vetores, A e B, com $n \leq 20$ e $m \leq 20$ elementos respectivamente. O programa deve efetuar INTERSECÇÃO entre os vetores, ou seja, os elementos em comum entre os dois vetores. Os dois vetores não contêm valores duplicados e não estão ordenados. Como resultado deve ser gerado o vetor C, que conterá a intersecção de A e B Exemplo:

  A = { 7, 2, 5, 8, 4} e B= {4, 2, 9, 5}, $C = A \cap B$ = {2, 5, 4}

  A = { 3, 9, 11} e B= {2, 6, 1}, $C = A \cap B$ = {}

Exercícios

- Escreva um programa que leia dois vetores, A e B, com $n \leq 20$ e $m \leq 20$ elementos respectivamente. O programa deve efetuar UNIÃO entre os vetores,. Os dois vetores não contêm valores duplicados e não estão ordenados. Como resultado deve ser gerado o vetor C, que conterá a intersecção de A e B Exemplo:

  A = { 7, 2, 5, 8, 4} e B= {4, 2, 9, 5}, $C = A \cup B$ = {7, 2, 5, 8, 4, 9}

  A = { 3, 9, 11} e B= {2, 6, 1}, $C = A \cup B$ = {3, 9, 11, 2, 6, 1}