Estruturas de Dados Indexadas - Arrays

- A descrição de algoritmos para a resolução de problemas, requer a representação da informação em elementos de dados.
- As diferentes formas como esses dados estão logicamente organizados definem diferentes estruturas de dados.
- As estruturas de dados primitivas são diretamente manipuláveis pela linguagem máquina.
- As estruturas de dados não primitivas são constituídas a partir das estruturas de dados primitivas logicamente relacionadas.

Estruturas de Dados

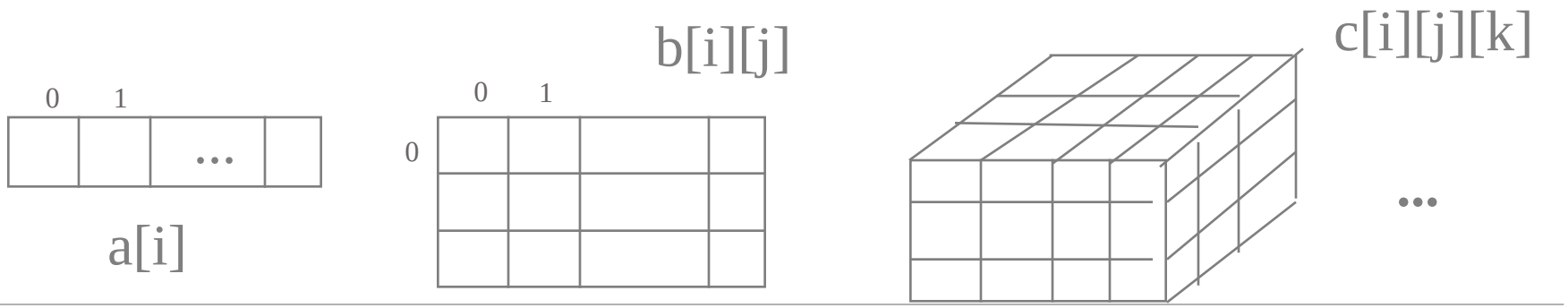
Estruturas de Dados Indexadas - Arrays

- **ARRAY-** é uma estrutura de dados que guarda um conjunto de elementos do mesmo tipo, armazenados contiguamente, e agrupados sobre o mesmo nome.

- Cada elemento do array é referenciado pelo nome do array e por um ou mais índices.
  Um array tem sempre associado índices, daí estas estruturas serem referidas como Estruturas de Dados Indexadas

- Quando se cria um array tem que se especificar o **número de elementos**. Esse **número** é **fixo**.

- **ARRAYS unidimensionais e multidimensionais**



Arrays em Java

- Em JAVA um array é um objeto, logo é criado com o operador new

```
int [] c;   // declara o array
c=new int[10]; //aloca espaço de memória e inicia valores
```

ou numa única instrução

```
int[] c=new int[10];
```

- **Uma vez criado um array ele tem tamanho (capacidade) fixo, neste caso 10 inteiros.**
- Os elementos são referidos por c[0]...c[9]
- Os elementos dos arrays são automaticamente iniciados:
  - a **0** os tipos de dados primitivos numéricos;
  - a **false** os booleanos;
  - a **null** as referências.
- Pode ser feita a criação de um array e a atribuição de valores numa só instrução

```
int [ ] n={10,20,30,40};
```

Arrays em Java

- Num array de N elementos, o índice tem que estar dentro dos limites, de 0 a N-1.

- Durante a execução o Java faz a validação automática dos valores dos índices e se não estiverem dentro dos limites lança uma exceção *ArrayIndexOutOfBoundsException*

- Cada array tem um **atributo público** designado por **length** que armazena o tamanho do array.

  Se int [ ] c = new int[100];

  - Os índices válidos variam entre 0 e 99
  - **c.length** guarda a capacidade do array, isto é, o nº total de elementos que pode guardar, neste caso 100

Arrays em Java

- Um Array é um objeto logo é uma variável do tipo referência

- **Tipos referência** não guardam valores, guardam um endereço de memória no qual se encontra um objeto

- **Criação de um array**

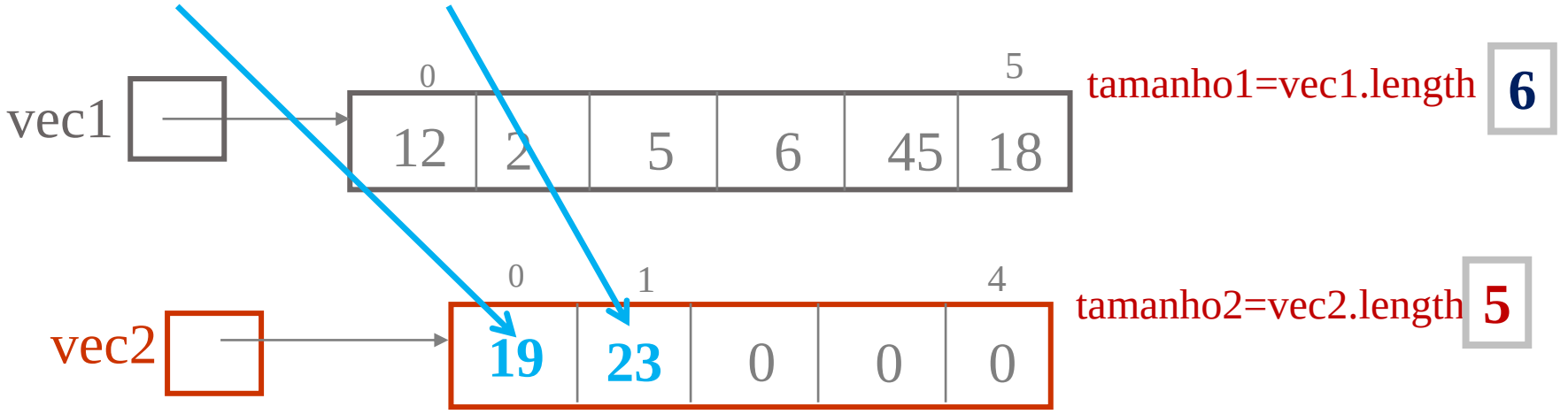
```
int[ ] vec1 = {12, 2, 5, 6, 45, 18};
int [ ] vec2 =new int[5];    geralmente requer gerir nº elementos
int nElem=0;
```

- Para manipular um array temos que ter presente
  - **Tamanho ou capacidade** (obtido do atributo length)
  - **Nº de elementos que contém** (o programador tem que gerir)
  - **Conjunto desses elementos**

Arrays em Java

- Atribuir valores a elementos de um array

vec2[0]=19; vec2[1]=23;



Arrays em Java

- Os elementos de um array podem ser do tipo primitivo ou referências para objetos.

Exemplos:

```
int [] nums=new int[10];

String[] nomes=new String[10];
```

Passagem de parâmetros por referência

- O nome de um array é uma referência para um objeto que contém os elementos do array.

```
int [ ] temperaturasHora=new int[24];
```

- Passar uma referência de um objeto para um método faz-se especificando a referência do objeto na chamada do método. Esta referência vai ser atribuída ao parâmetro do método (neste caso o parâmetro b)

```
actualizaArray(temperaturasHora);
```

- No corpo do método quando mencionamos o parâmetro (referência para o objeto), estamos a aceder ao objeto original, podendo alterar os seus atributos.

```
int [ ] temperaturasHora=new int[24];   //criar array

actualizaArray(temperaturasHora);      //chamada do método
    …
void actualizaArray(int [ ] b ) {
    b[1]=22;
 …}
```

Manipulação de Arrays unidimensionais

- Criar array
- Inserir elementos
- Actualizar elementos
- Listar elementos
- Apagar elementos
- Pesquisar elemento
- Ordenar vector

Cópia de arrays

Um array é uma estrutura de dados de tamanho FIXO. Se precisarmos de alterar a dimensão, teremos de criar um novo array e copiar o conteúdo para esse novo array.

Considere

```
int novaDim=???
int [ ] vec={1,2,3,4,5,6};
int [ ] vecCopia= new int[novaDim];
```

Copiar o array vec para o array vecCopia

System.arraycopy(vec,0,vecCopia,0,vec.length);

- O primeiro argumento de System.arraycopy é a referência do array origem, de onde serão copiados os elementos
- O segundo argumento é o índice que especifica o início dos elemento a copiar
- O terceiro argumento é o array destino, que armazena a cópia
- O quarto argumento é o índice do array destino para onde o primeiro elemento é copiado
- O quinto argumento especifica o número de elementos a copiar