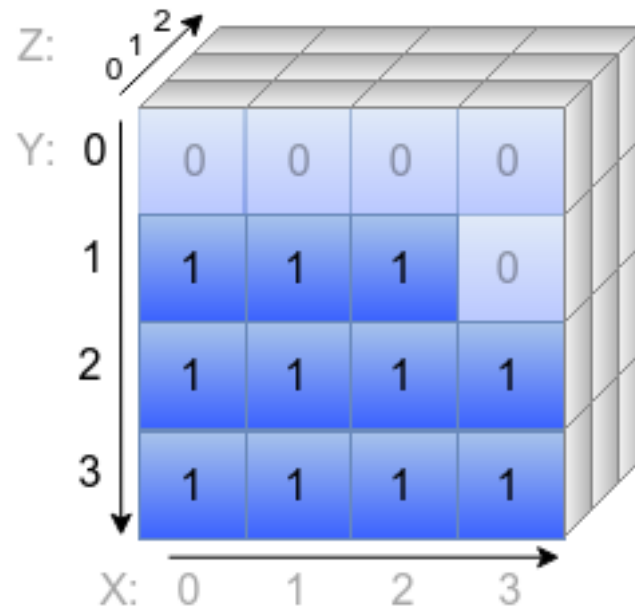


APROG – Algoritmia e Programação



Arrays

Emanuel Cunha Silva

ecs@isep.ipp.pt

Guardar uma garrafa de vinho



Guardar várias garrafas de vinho



Guardar uma garrafa de vinho



Guardar várias garrafas de vinho



Variável simples

guarda um valor

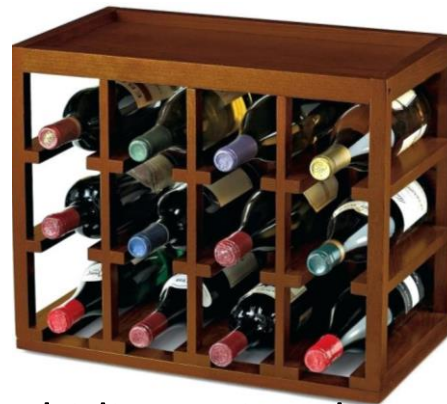


Variável composta (Array)

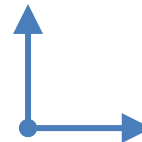
guarda vários valores do mesmo tipo



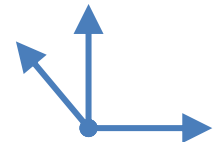
unidimensional



bidimensional



tridimensional



Arrays

Um array é uma estrutura de dados usada para armazenar uma coleção (possivelmente grande) de dados

- Permite armazenar múltiplos valores, **do mesmo tipo**, ao mesmo tempo
- Todos os valores estão associados a **um único identificador**
- Dimensão é fixa - não modificável em tempo de execução (run-time)

- Configuração

- Vetor – organização linear, unidimensional
 - Matriz – organização matricial – bidimensional
 - ...

12	20	14
----	----	----

11	24	27
5	56	18
1	8	34

- **Exemplos:**

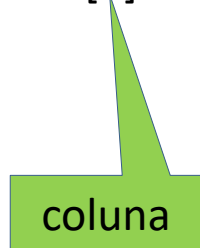
- notas de alunos (conjunto de reais)
 - nomes de alunos (conjunto de Strings)

Arrays

- O **Acesso** aos elementos é realizado através de **índices**
- Índices
 - Indicam as **posições** dos elementos
 - Números **inteiros** desde 0
 - **Arrays unidimensionais possuem apenas um índice**
- Exemplo:
 - `arr1[0] = 12`
 - `arr1[1] = 20`
 - `arr1[2] = 14`

0	1	2
12	20	14

arr1



Arrays

- O **Acesso** aos elementos é realizado através de **índices**
- Índices
 - Indicam as **posições** dos elementos
 - Números **inteiros** desde 0
 - **Arrays bidimensionais possuem dois índices**
 - **Índice para as linhas**
 - **Índice para as colunas**

- Exemplo:
 - `arr2[0][0] = 11`
 - `arr2[0][1] = 24`
 - `arr2[0][2] = 27`
 - `arr2[1][0] = 5`
 - `arr2[1][1] = 56`
 - `arr2[1][2] = 18`

	0	1	2
0	11	24	27
1	5	56	18
2	1	8	34

arr2

Arrays

- Declaração

- A dimensão do array é definida na declaração

- Array unidimensional: ***tipo* nomeArray [dimensão]**

- Exemplo: ED: TEXTO nomes[20]
ED: REAL notasAlunos[15]
ED: INTEIRO alunosAprovados[15]

- Array bidimensional: ***tipo* nomeArray [dimensão1] [dimensão2]**

- Exemplo: ED: TEXTO moradoresDoPredio[3][20]
ED: REAL notasAlunosDasVariasTurmas[4][15]

Arrays

- Guardar/Atualizar um valor num array

- Exemplo: ED: TEXTO nomes[3]
nomes[0] ← "Zé"
nomes[1] ← "Maria"

nomeArray [índice] ← valor



- Obter um valor de um array

- Exemplo: ED: TEXTO nome1, nome2
...
nome1 ← nomes[1]
nome2 ← nomes[0]

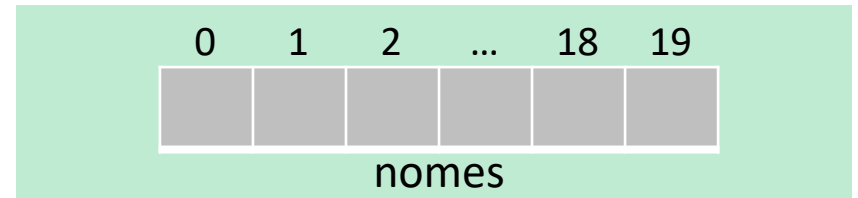
variável ← nomeArray [índice]

// "Maria"

// "Zé"

Arrays

Problema1: Ler e guardar o nome de 20 pessoas num array. De seguida mostre os nomes guardados pela ordem em que foram lidos.



INICIO

ED: TEXTO nomes[20], nome
ED: INTEIRO índice

REPETIR PARA índice \leftarrow 0 ATE 19 PASSO 1

LER (nome)

nomes[índice] \leftarrow nome

FIMREPETIR

REPETIR PARA índice \leftarrow 0 ATE 19 PASSO 1

nome \leftarrow nomes[índice]

ESCREVER (nome)

FIMREPETIR

FIM

INICIO

ED: TEXTO nomes[20], nome
ED: INTEIRO índice

REPETIR PARA índice \leftarrow 0 ATE 19 PASSO 1

LER (nomes[índice])

FIMREPETIR

REPETIR PARA índice \leftarrow 0 ATE 19 PASSO 1

ESCREVER (nomes[índice])

FIMREPETIR

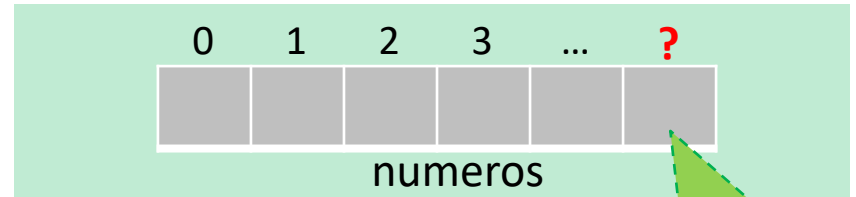
FIM



Arrays

Problema2: Ler e guardar números inteiros até ser inserido um número negativo.
De seguida, calcule e mostre a média dos números inseridos.

Estimar valor. Pecar por excesso



INICIO

ED: INTEIRO **numeros[20]**, num, qtdNumeros, índice, soma

qtdNumeros ← 0

LER (num)

REPETIR ENQUANTO (num >= 0)

numeros [qtdNumeros] ← num

qtdNumeros ← **qtdNumeros + 1**

LER (num)

FIMREPETIR

soma ← 0

REPETIR PARA índice ← 0 ATE **qtdNumeros - 1** PASSO 1

soma ← soma + numeros [índice]

FIMREPETIR

ESCREVER (soma / qtdNumeros)

FIM

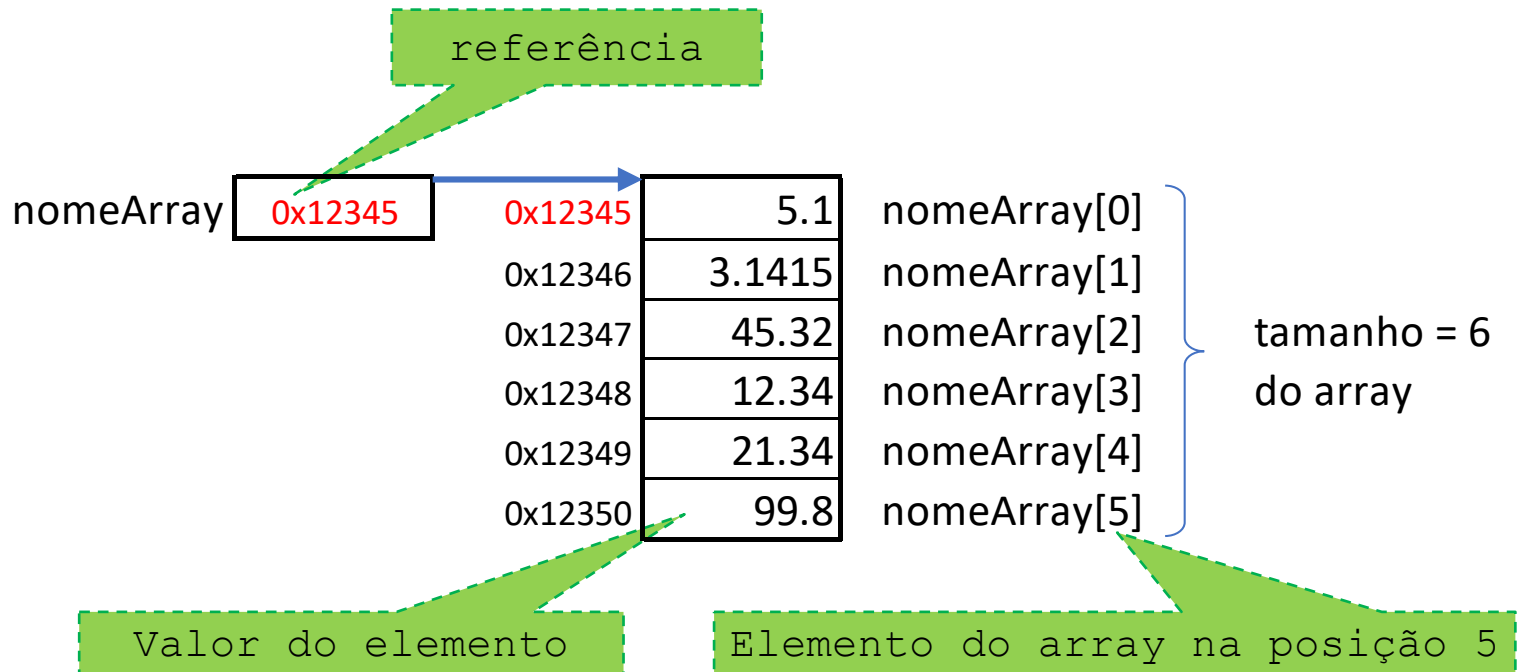
Que dimensão?
Quantos n^ºs é para guardar?



<Java arrays>

Java - arrays

- Um array é um **objeto**
- O nome do array é uma **referência** para o objeto
- É uma estrutura indexada – são usados índices para aceder ao conteúdo
- Uma vez criado, o seu tamanho é imutável (fixo)
- Um array é criado com o operador **new**



Java - arrays

- Declarar e criar um array:

```
tipo[] nomeArray; //declarar array unidimensional  
nomeArray = new tipo[dimensão]; //criar
```

```
tipo[][] nomeArray; //declarar array bidimensional  
nomeArray = new tipo[dimensão1] [dimensão2]; //criar
```

- Declarar e criar um array numa instrução única:

```
tipo[] nomeArray = new tipo[dimensão];
```

Java - arrays

- Exemplo:

```
String[] nomes = new String[10];           //declarar e criar numa instrução única
```

```
float[] notas;                             //declarar  
notas = new float[15];                     //criar
```

```
float[][] temperaturas = new float[7][24]; //declarar e criar matriz
```

- Os elementos dos arrays são automaticamente inicializados
 - A **0** (zero) - os tipos de dados primitivos numéricos
 - A **false** - os booleanos
 - A **null** - as referências (Ex: String)
- Podem ser atribuídos elementos aos arrays na fase de criação

```
String[] nomes = {"Ana", "Berta", "Carla", "Daniela"};
```

```
int[] notas = {12, 14, 10, 9, 19};
```

Java - arrays

- Armazenar elementos num array
- **Exemplo:** armazenar até 5 nomes de pessoas

declarar

```
String[] nomes = new String[5];
```

nomes

null	null	null	null	null
0	1	2	3	4

armazenar

```
nomes[0]= "Ana";  
nomes[1]= "Berta";  
nomes[2]= "Carla";
```

nomes

Ana	Berta	Carla	null	null
0	1	2	3	4

Array não está totalmente
preenchido!!!

Java - arrays

- Inicializar um array com um valor (-1)
 - Inicializar um array, é colocá-lo num estado considerado inicial
- **Exemplo:** inicializar os valores da contagem da água dos 12 meses com o valor -1

declarar

```
double[] contagemDaAguaPorMes = new double[12];
```

contagemDaAguaPorMes	0	0	0	0	0	0	0	0	0	0	0	0
	0	1	2	3	4	5	6	7	8	9	10	11

inicializar

```
for( int mes = 0; mes<12; mes++)
```

```
    contagemDaAguaPorMes[ mes ] = -1;
```

contagemDaAguaPorMes	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	0	1	2	3	4	5	6	7	8	9	10	11

Java - arrays

- Aceder a um array
 - O índice usado tem de estar dentro dos limites, [0 ... tamanho-1]
 - O tamanho de um array é conhecido através do seu atributo **length**
 - A tentativa de acesso fora dos limites gera o erro *ArrayIndexOutOfBoundsException*
- Exemplo:

```
double[] notas = new double[15];
```

```
notas[-1] = 19;           // ERRO (ArrayIndexOutOfBoundsException)
notas[15] = 19;           // ERRO (ArrayIndexOutOfBoundsException)
notas[0] = 19;            // OK – guarda o número 19 na posição 0)
notas[1] = 20.5;          // OK – guarda o número 20.5 na posição 1)
notas[14] = 99;           // OK – guarda o número 99 na posição 14)
```

Java - passar arrays para métodos

- Como variáveis, também podemos passar arrays por parâmetro para os métodos.

```
class Test {
```

```
    public static void main(String[] args) {  
        int[] meuArray = {3, 1, 2, 5, 4};
```

Enviar variável

```
        somarArray( meuArray );  
    }
```

// passar array para o método *somarArray()*

```
    public static void somarArray( int[] arr ) { // somar os elementos do array  
        int soma = 0;  
        for (int i = 0; i < arr.length; i++)  
            soma += arr[i];
```

Declarar tipo e nome
da variável recetora

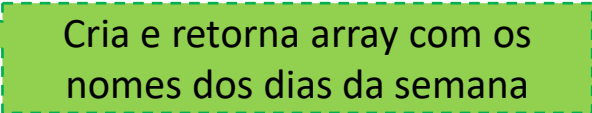
```
        System.out.println("soma dos elementos do array : " + soma);  
    }
```

```
}
```

Java – retornar array de métodos

- Um método também pode retornar um array.

```
class Test {  
    public static void main(String[] args) {  
        String[] arr;           // declarar array  
  
        arr = diasDaSemana();    // receber array retornado pelo método diasDaSemana()  
    }  
  
    public static String[] diasDaSemana() {  
        String[] result = new String[7];  
        result[0]="Domingo";  
        result[1]="Segunda";  
        result[2]="Terça";  
        result[3]="Quarta";  
        result[4]="Quinta";  
        result[5]="Sexta";  
        result[6]="Sábado";  
        return result;  
    }  
}
```



Cria e retorna array com os nomes dos dias da semana

Java – retornar array de métodos

Problema: Implemente um método que crie e retorne um array preenchido com números aleatórios. O tamanho do array é passado por parâmetro ao método

```
class Test {  
    public static void main(String[] args) {  
        double[] arr;  
  
        arr = gerarArrayComNumerosAleatorios(10);  
    }  
  
    public static double[] gerarArrayComNumerosAleatorios(int tamanho) {  
        double[] arrResultado = new double[tamanho];  
        for (int indice = 0; indice < tamanho; indice++) {  
            arrResultado[indice] = Math.random();  
        }  
        return arrResultado;  
    }  
}
```

criar array

preencher array

retornar array

Java – operações típicas com arrays

Criar array	<pre>double[] arr = new double[15];</pre>
Ler valores para o array	<pre>Scanner ler = new Scanner(System.in); for(int indice = 0; indice < arr.length; indice++) arr[indice] = ler.nextDouble();</pre>
Imprimir os elementos do array	<pre>for(int indice = 0; indice < arr.length; indice++) System.out.println(arr[indice]);</pre>
Calcular a média dos elementos do array	<pre>double soma = 0.0; for(int indice = 0; indice < arr.length; indice++) soma += arr[indice]; double media = soma / arr.length;</pre>

Java – operações típicas com arrays

Preencher o array com valores aleatórios	<pre>for(int indice = 0; indice < arr.length; indice++) arr[indice] = Math.random();</pre>
Encontrar o maior elemento do array	<pre>double maior = arr[0]; for(int indice = 1; indice < arr.length; indice++) if(arr[indice] > maior) maior = arr[i];</pre>
Inverter a ordem dos elementos do array	<pre>for(int idx1 = 0, idx2 = arr.length-1; idx1<idx2; idx1++, idx2--){ double aux = arr[idx1]; arr[idx1] = arr[idx2]; arr[idx2] = aux; }</pre>
Copiar os elementos do array para um novo array	<pre>double[] novoArr = new double[arr.length]; for(int indice = 0; indice < arr.length; indice++) novoArr[indice] = arr[indice];</pre>

Java – arrays multidimensionais

Arrays multidimensionais são arrays de arrays, com cada elemento do array mantendo a referência de outro array.

Um array multidimensional é criado anexando um conjunto de parêntesis retos ([]) por dimensão.

- **Exemplo:**

```
// criar um array 2D
```

```
int[][] arr = new int [2] [3];
```

```
// declarar e inicializar array 2D
```

```
Int[][] arr = { {2,7,9}, {3,6,1} };
```

```
// imprimir array 2D
```

```
for (int linha=0; linha< 2 ; linha++) {  
    for (int coluna=0; coluna < 3 ; coluna++)  
        System.out.print(arr[linha][coluna] + " ");  
    System.out.println();  
}
```

arr

2	7	9
3	6	1

Java – arrays multidimensionais

▪ Exemplo:

```
// criar um array 3D
```

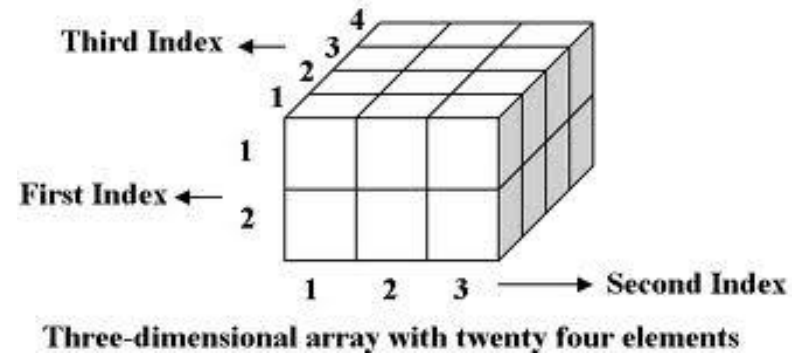
```
int[][][] arr = new int [2] [3] [4];
```

```
// declarar e inicializar array 3D
```

```
int arr[][][] = { { {1, 2, 3},  
                  {4, 5, 6} },  
                  { {7, 8, 9},  
                    {10, 11, 12} } };
```

```
// imprimir array 3D
```

```
for (int camada=0; camada < 2 ; camada++) {  
    for (int linha=0; linha< 2 ; linha++) {  
        for (int coluna=0; coluna < 3 ; coluna++)  
            System.out.print(arr[linha][coluna][camada] + " ");  
        System.out.println();  
    }  
    System.out.println();  
}
```



1	2	3
4	5	6

7	8	9
10	11	12

Java – arrays multidimensionais

Problema: Implemente um método que calcule o somatório dos elementos (números inteiros) de uma linha de uma matriz. O método recebe, por parâmetro, a matriz e o número da linha a somar e retorna o valor do somatório calculado.

```
public static void main(String args[]) {  
    int [][] matriz = { {2,7,9,1,3}, {3,6,1,2,2}, {2,3,4,5,6} };  
  
    System.out.println("soma =" + somarLinha(matriz, 2));  
}
```

arr	2	7	9	1	3	
	3	6	1	2	2	
→	2	3	4	5	6	= 20

```
public static int somarLinha(int[][] arr, int linha) {  
    int soma = 0;  
    for (int coluna=0; coluna < 5 ; coluna++)    // coluna < 5 ⇔ coluna < arr[linha].length  
        soma += arr[linha][coluna];  
    return soma;  
}
```

Java – exercícios com arrays

Problemas: Utilizando arrays e modularização implementar as seguintes funcionalidades:

String[]

1. Ler n nomes para array;
2. Visualizar o conteúdo do array;
3. Procurar um elemento (nome) no array e retornar a posição em que ocorre ou (-1) se não encontra;
4. Ordenar o array, alfabeticamente, de forma descendente.

Int[]

1. Gerar aleatoriamente 10 valores (entre 1 e 100) para array;
2. Visualizar o conteúdo do array;
3. Procurar um elemento no array e retornar a posição em que ocorre ou (-1) se não encontra;
4. Calcular e retornar a media dos elementos do array;
5. Calcular e retornar a quantidade de elementos superiores à média do array;
6. Ordenar o array de forma ascendente.

Java – exercícios com arrays

Problemas: Utilizar um array multidimensional para guardar as temperaturas registadas por hora ao longo do dia durante os 7 dias da semana. Implementar as seguintes funcionalidades usando modularização:

`double[][]`

1. Criar um array de 7 linhas e 24 colunas (para representar os dias da semana e as horas do dia) e preencher com valores aleatórios entre 0 e 50;
2. Visualizar o conteúdo do array (em forma de matriz);
3. Calcular e retornar a temperatura média de um dia (qualquer);
4. Calcular e retornar qual o dia da semana que registou a maior temperatura a uma determinada hora do dia;
5. Calcular e retornar quantos dias da semana registaram temperaturas negativas.