Estrutura da Informação

Por que estudar Estrutura da Informação ou Estrutura de Dados

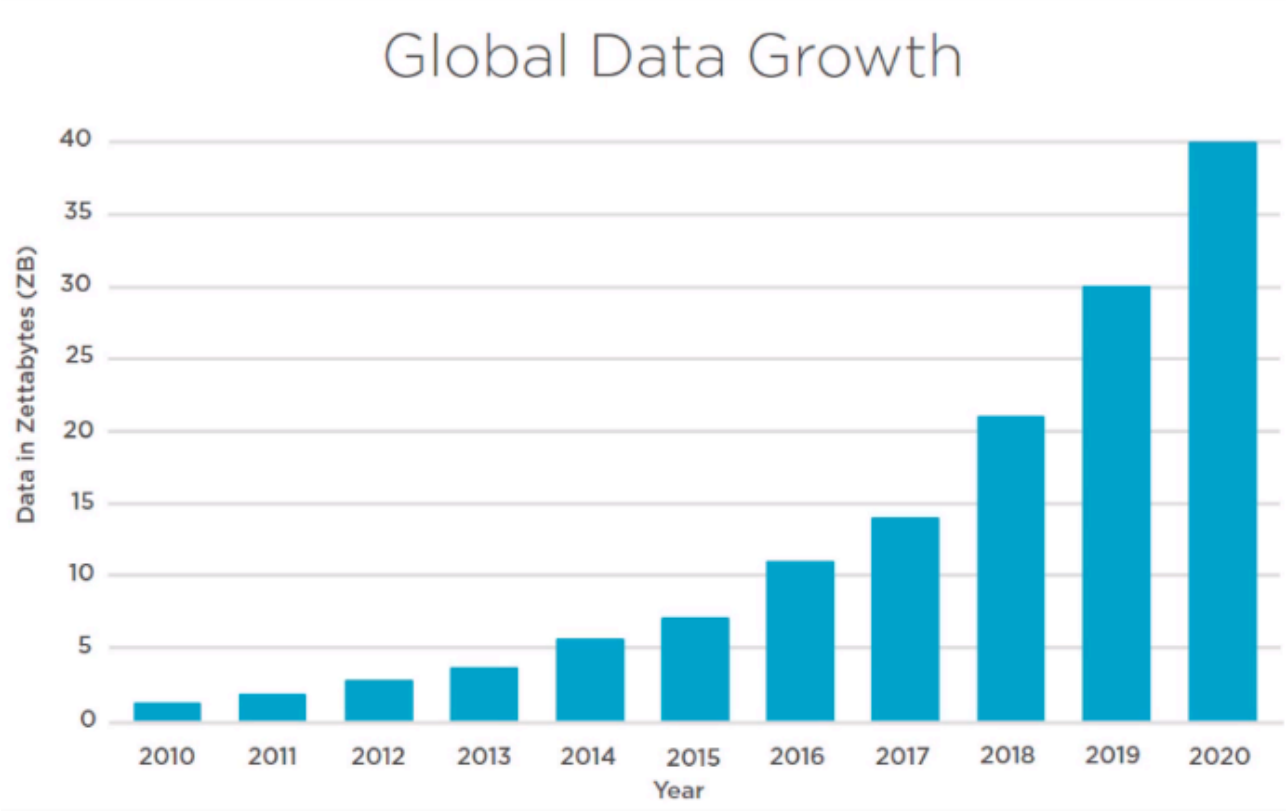
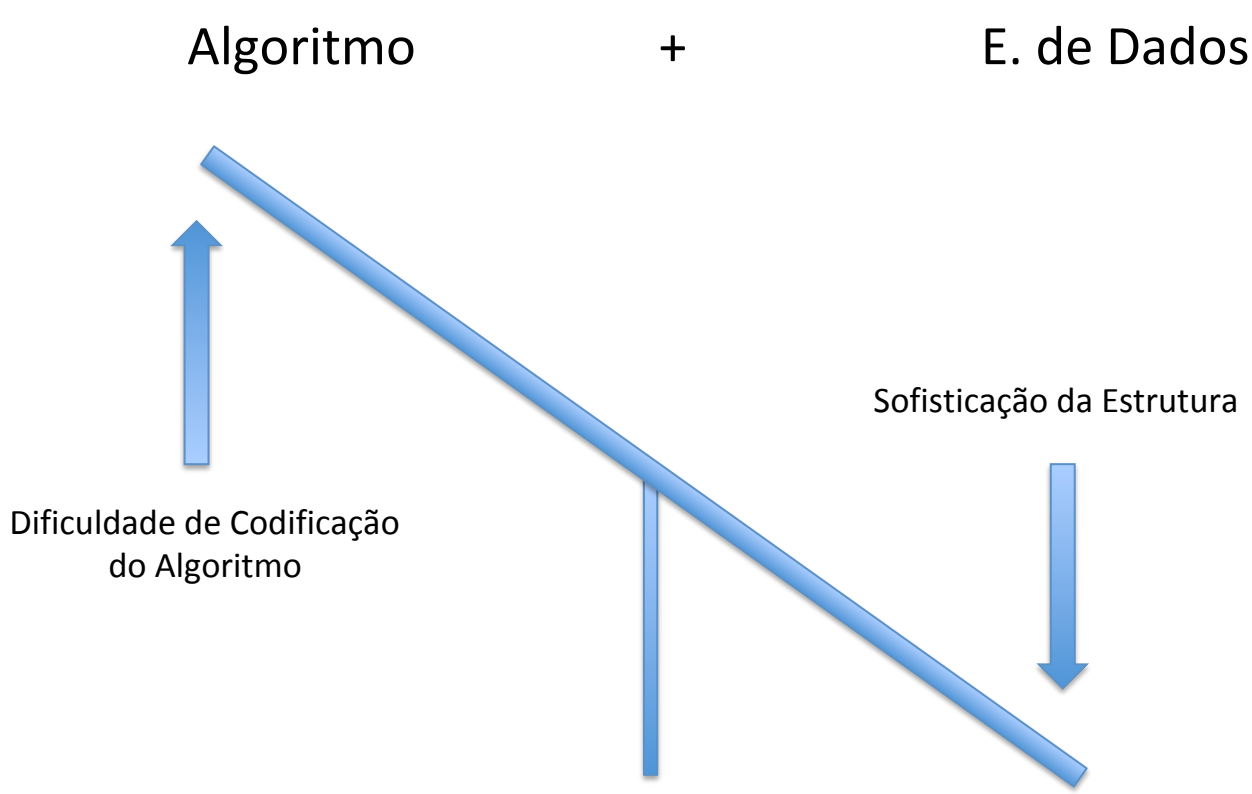


Figura 6—Tendência de crescimento do volume de dados durante os anos (Fonte: UNECE Statistics wikis)

Prefixo do SI		
Nome	**Símbolo**	**Múltiplo**
byte	B	10^{0}
kilobyte	kB	10^{3}
megabyte	MB	10^{6}
gigabyte	GB	10^{9}
terabyte	TB	10^{12}
petabyte	PB	10^{15}
exabyte	EB	10^{18}
zettabyte	ZB	10^{21}
yottabyte	YB	10^{24}
undecabyte	UB	10^{33}

Fonte: Wikipedia

Programa = Algoritmo + Estruturas de Dados



Introdução e Tipos Abstratos de Dados (Aula 1)

A menor parte de informação representada em um computador é o bit (binary digit). Um bit pode assumir apenas um estado: 1 “ligado” e 0 “desligado”. A partir de uma sequência de bits foram definidos sistemas de representação para tipos básicos de dados (inteiros, float, double, char, boolean, etc.).

Tipos Abstratos de Dados

Um Tipo Abstrato de Dados — TAD — consiste em um conjunto de valores (variáveis) e operações que podem ser realizadas sobre esses valores. A partir de tipos básicos de dados é possível definir um tipo abstrato de dados.

Dessa forma, podemos relacionar os elementos de um TAD da seguinte forma:

- Conjunto de variáveis (que podem ser de tipo primitivo, agregados ou outros TADs);
- Conjunto de métodos que manipulam as variáveis do TAD e devem necessariamente ser implementadas considerando *pré-condições* (verificação de estado que permite a execução da operação) e pós-condições (transformações no TAD).

Tipos de Dados

Em Java existem os seguintes tipos básicos de dados:

Tipo	Tamanho (bits)	Valor
Byte	8	-128 a 127
Short	16	-32.768 a 32.767
Int	32	-2.147.483.648 a 2.147.483.647
Long	64	-9223372036854775808 a 9223372036854775807
Float	32	Precisão Decimal Simples
Double	64	Precisão Decimal Dupla
Char	16	Um caracter
Boolean	1	true ou false

Array (Agregados Homogêneos)

A forma mais simples de uma array é a array unidimensional. Ela pode ser definida como um conjunto ordenado finito de elementos homogêneos. "Ordenado" porque os elementos de uma array estão dispostos de forma que há um primeiro, um segundo, etc. "Finito" porque toda array precisa ter especificado o seu tamanho e "homogêneo" porque todos os elementos de uma array possuem o mesmo tipo de dado.

A declaração

```
int[] a = int[100];
```

cria um array "a" de 100 elementos do tipo inteiro.

Uma array é utilizada quando é necessário armazenar uma grande quantidade de informação em memória e acessar todos os seus itens de maneira uniforme. A declaração anterior aloca 100 posições sucessivas de memória, cada uma contendo um simples inteiro. Suponha que a primeira posição seja referenciada por base(a) e que cada elemento da array ocupe um determinado espaço "size". Portanto a localização do primeiro elemento de "a" (a[0]) é dada por base(a), uma referência ao segundo elemento (a[1]) é dada por base(a) + size, b[2] é referenciado pela posição base(a) + 2 * size. Pode-se obter facilmente uma equação genérica para o cálculo de endereços de uma array: a[i] = base(a) + i * size, como ilustrado na Figura 1.

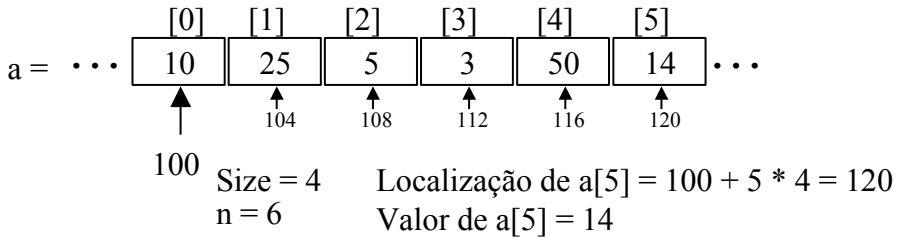


Figura 1: Representação de uma array de inteiros.

Arrays Bidimensionais

O tipo dos elementos de uma array pode ser uma outra array. Por exemplo, pode-se definir:

int[][] a = new int[3][5]

A declaração anterior define uma array com três elementos. Cada elemento da array é composto por 5 inteiros. A Figura 2 ilustra esta array.

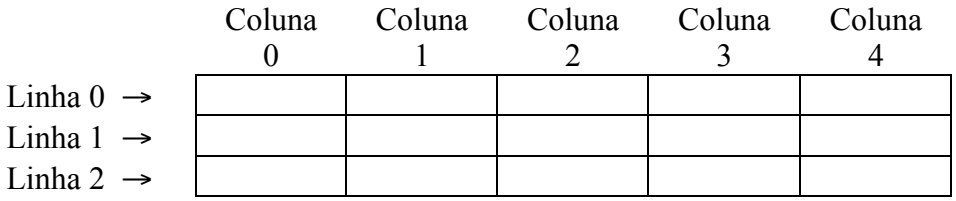


Figura 2: Array Bidimensional.

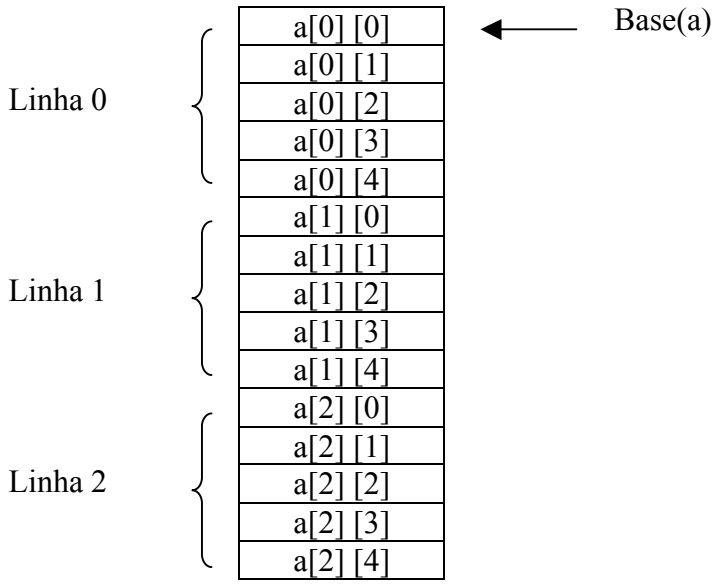


Figura 3: Array Bidimensional.

Assumindo que o tamanho de cada elemento é *size*, o cálculo de endereço de um elemento a[i1][i2] na array anterior (a[3][5]) é dado da seguinte forma:

Endereço de a[i1][i2] = base(a) + (i1 * 5 + i2) * size

Generalizando para uma array a[r1][r2], o endereço de a[i1][i2] é

a[i1][i2] = base(a) + (i1 * r2 + i2) * size

Considerando size = 1, o endereço de a[2][4] é dado por:

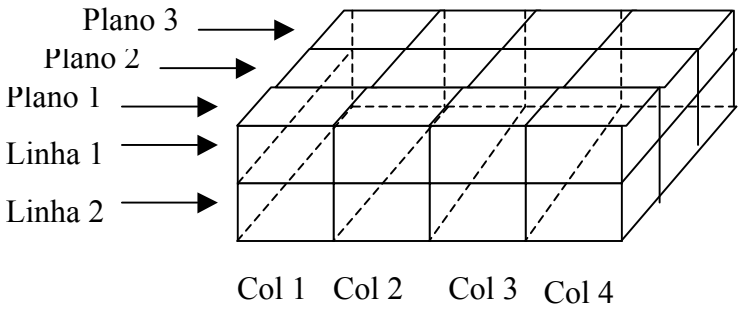
$$base(a) + (2 * 5 + 4) * 1 \Leftrightarrow base(a) + 14$$

Arrays Multidimensionais

Java também permite que arrays multidimensionais sejam declaradas. Por exemplo:

```
int[][][] a = new int[3][2][4]
```

Um elemento desta array é acessado através de três índices: a[2][1][3]. Um exemplo de array tridimensional pode ser a representação das coordenadas de temperatura indexadas por latitude, longitude e altitude.



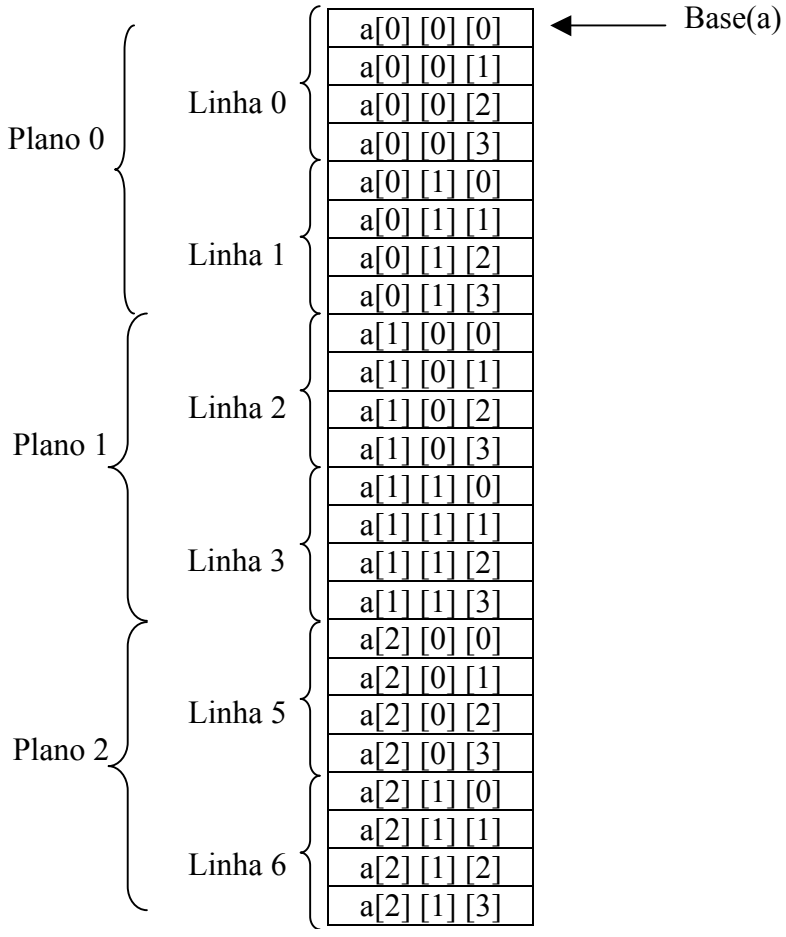


Figura 4: Array Tridimensional

Considere a declaração de uma array multidimensional

$$a[r_1][r_2] \ldots [r_n]$$

Uma forma de calcular os endereços de uma array multidimensioanl é atribuir pesos aos índices. Por exemplo, na declaração anterior, o índice mais à direita (r_n) possui peso *size*; o índice r_{n-1} possui peso r_n * size, o peso r_{n-2} possui peso peso(r_{n-1}) * r_{n-1}, o índice r_{n-3} possui peso peso(r_{n-2}) * r_{n-2}. A partir dos pesos, é possível calcular o endereço de qualquer elemento através da equação:

$$Endereço(a[j_1][j_2]...[j_n]) = base(a) + \sum_{i=1}^{n} peso(r_i) \times j_i$$

Por exemplo, considere a array da Figura 4 (a[3][2][4]). O endereço do elemento a[2][0][3] pode ser claculado da seguinte forma (considerando o *size* = 1 e base(a) = 0):

Pesos: r3 = 1, r2 = (4 * 1) = 4, r1 = (2 * 4) = 8

$$Endereço(a[2][0][3]) = 0 + (2 \times 8 + 0 \times 4 + 3 \times 1) = 19$$

Estruturas (Agregados Heterogêneos)

Uma estrutura consiste em um conjunto de variáveis. Cada uma é representada por um identificador próprio e é chamado de *membro*. Em algumas linguagens de programação uma estrutura é representada pelo identificador *record* (Pascal) e *struct* (linguagem C) e seus itens são chamados de *campos*. Por exemplo:

```
Estrutura Nome {
        String primeiro_nome;
        char meio ;
        String ultimo_nome;
}
```

A declaração anterior cria uma estrutura (classe) que representa nomes de pessoas. Esta estrutura apresenta três campos (membros).

Exercícios (Aula Teórica)

1. Dadas as declarações a seguir, e assumindo que a primeira posição das arrays são sempre o endereço 100, calcule a posição dos elementos:
   a) int[] a = int[100] endereço de a[10]:
   b) int[] a = int[200] endereço de a[100]:
   c) int[][] a = int[10][20] endereço de a[0][0]:
   d) int[][] a = int[10][20] endereço de a[2][1]:
   e) int[][] a = int[10][20] endereço de a[5][1]:
   f) int[][] a = int[10][20] endereço de a[1][10]:
   g) int[][] a = int[10][20] endereço de a[2][10]:
   h) int[][] a = int[10][20] endereço de a[5][3]:
   i) int[][] a = int[10][20] endereço de a[9][19]:

2. Assumindo que um inteiro necessita de 4 bytes, um numero real necessita de oito bytes, e um char necessita de 1 byte. Considere as seguintes declarações:

```
Estrutura Nome{
        char[] primeiro_nome = char[10];
        char meio;
        char[] ultimo_nome = char[20];
}

Estrutura Pessoa{
        Nome nome;
        int[] dia_aniversario = int[2];
        Nome[] nome_pais = Nome[2];
        int salario = 0;
        int numero_filhos = 0;
        char[] endereco = char[20];
        char[] cidade = char[10];
        char[] estado = char[2];
}
```

Considerando que o endereço inicial é 100, a partir da declaração *cad = Pessoa[100]*, quais são os endereços de cada um dos elementos a seguir?

a) cad[10]:
b) cad[20].nome.meio:
c) cad[20].salario:
d) p[20].endereco[5]:
   p[5].nome_pais[1].ultimo_nome[10]:

Exercícios

1. A **média aritmética** de uma array de números é dada pelo somatório do valor de seus elementos dividido pelo número de elementos. Escreva um programa que receba uma array e compute sua média.
2. A **moda** de uma array de números é o número *m* na array que é repetido mais vezes. Se há mais de um número com mais repetições do que os outros então não há moda única. Escreva um programa que receba uma array de números e retorne a moda da array. Caso a moda única não exista então informe o usuário.
3. Implemente um TAD Pessoa, com os seguintes campos: nome, sobrenome, idade, salário
4. Implemente uma função que:
   - Solicite do usuário o tamanho N de um vetor;
   - Crie um método que aloque um vetor de pessoas de tamanho N, faça entrada de dados e na sequência imprima de forma formatada as informações das pessoas cadastradas

Implementação de TADs (Tipos Abstratos de Dados) em Java

Verificou-se na seção anterior, como um tipo heterogêneo de dados poderia ser implementado utilizando estruturas. Em Java, uma estrutura é declarada como uma classe. Porém, uma classe é muito mais completa do que uma estrutura, pois pode apresentar, além dos campos e membros da estrutura, procedimentos internos chamados métodos. Esses procedimentos podem ser utilizados para manipular os valores armazenados nos membros das classes. Essas características tornam as classes uma ferramenta poderosa e ao mesmo tempo simples para a implementação de TADs.

Considere a seguinte declaração:

```
Public class Nome {
        public String primeiro_nome = new String();
        public char car ;
        public String ultimo_nome = new String();
}
```

A partir da declaração de uma classe é possível criar instâncias de objetos pertencentes àquelas classes.

```
public static void main(String args[]){
        Nome pai = new Nome();
        pai.Primeiro_nome = "João";
        pai.Ultimo_nome = "Silva";
}
```

O código anterior cria uma instância da classe Nome chamada pai. São atribuídos aos membros de pai os valores "João" e "Silva". Membros de uma classe podem ser outras classes, por exemplo:

```
public class Endereço{
        public String descrição = new String();
        public String cidade = new String;
        public char[] estado = new char[2];
        public int cep = new int;
}

public class Registro {
        Nome pessoa = new Nome();
        Endereço local = new Endereço();
}
```

A declaração anterior cria uma classe Registro que possui dois campos (Nome e Endereço) que também são classes.

```
public static void main(String args[]){
        Registro tupla = new Registro();
        tupla.pessoa.primeiro_nome = "Joaquim";
        tupla.local.cidade = "Curitiba";
}
```

Exercícios

Suponha que um número real é representado por uma estrutura declarada da seguinte forma:

```
public class Real {
    int esquerda = 0;
    int direita = 0;
    boolean positivo = true;
}
```

Essa estrutura representa um TAD de número real, onde esquerda representa a parte inteira do número e direita a parte decimal do número. Se o número é um número negativo, então o membro *positivo* apresenta o valor *false*.

a) Escreva um procedimento que leia um número real e crie uma estrutura representando aquele número;
b) Escreva uma função que receba tal estrutura e retorne o número real representado por ela;
c) Escreva métodos *soma*, *subtrai*, *multiplica* e *divide* para a classe Real que aceitam um outro objeto da mesma classe e retorne a referência para um terceiro objeto representando o valor obtido com as operações anteriores realizadas sobre a estrutura de entrada.