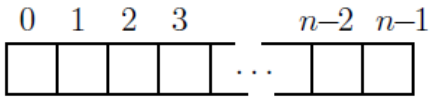
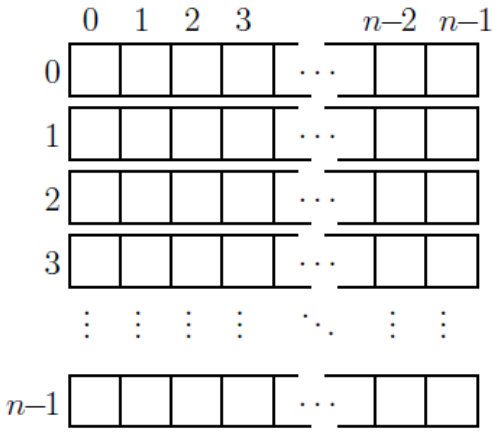
Universidade Federal do ABC	**BC0505 – Processamento da Informação** ***Assunto:*** **Vetores Bidimensionais (Matrizes)**

1. Introdução: vetores bidimensionais

Uma matriz é um vetor com duas dimensões e costuma ser usada para representar tabelas de valores que consistem nas informações dispostas em linhas e colunas. Para identificar um elemento de uma tabela particular devemos especificar dois índices. O primeiro índice se refere à linha do elemento e o segundo à sua coluna. O Java não suporta vetores multidimensionais (matrizes) diretamente, mas permite que o programador especifique vetores unidimensionais cujos elementos também são vetores unidimensionais.



a) Vetor Unidimensional



b) Vetor Bidimensional

A sintaxe para a definição de uma matriz é bem parecida com a definição de vetores. O que muda é que existe outra dimensão para definição de linhas. Um exemplo para a definição de uma matriz com 5 linhas e 5 colunas:

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

A primeira parte do comando, `int matrix[][]` corresponde à declaração da matriz, incluindo o tipo `int` e o nome da matriz `matrix[]`. Como no caso de vetores, o operador `new` é necessário para criar a instância da matriz, indicando o tipo de dados que está sendo instanciado `int` e o número de posições `[5][5]`. Para atribuir valores à matriz, veja o exemplo abaixo.

```
public class UsingMatrixJava {
    public static void main(String[] args) {
        int matrix[][] = new int[5][5];

        matrix[0][0]=1;
        matrix[0][1]=2;
        matrix[0][2]=3;
        matrix[0][3]=4;
        matrix[0][4]=5;

        matrix[1][0]=6;
        matrix[1][1]=7;
        matrix[1][2]=8;
        matrix[1][3]=9;
        matrix[1][4]=10;

         // imprime os elementos da matriz
        for (int line = 0; line < matrix.length; line++) {
            for (int column = 0; column < matrix[0].length; column++) {
                System.out.printf("%d ",matrix[line][column]);
            }
            System.out.println();
        }
    }
}
```

Para a matriz abaixo, o código poderia ser:

```
public class UsingMatrixJava {
    public static void main(String[] args) {
        int matrix[][] = { {34, 56, 4, 78, 89},{-7, 36, 81, 32, -23},
                           {2, 45, 53, 62, 25},{12, 17, 28, 74, 33},
                           {65, -34, 47, 8, 1} };

         // imprime os elementos da matriz
        for (int line = 0; line < matrix.length; line++) {
            for (int column = 0; column < matrix[0].length; column++) {
                System.out.printf("%d ",matrix[line][column]);
            }
            System.out.println();
        }
    }
}
```

34	56	4	78	89
-7	36	81	32	-23
2	45	53	62	25
12	17	28	74	33
65	-34	47	8	1

Neste caso, a matriz já é inicializada com os valores e, da mesma forma que em vetores, a definição de uma matriz em Java pode assumir qualquer tipo previamente definido para a linguagem. Uma possibilidade de programa para imprimir uma matriz "formatada" é dada a seguir. Note que, para percorrer as colunas, fixou-se a linha zero da matriz onde `matrix.length` dá o número de linhas da matriz.

Exercícios

Questão 1: Faça um programa que leia valores para preencher uma matriz 5x5 (5 linhas e 5 colunas) de números reais. O programa deve mostrar os valores da matriz, organizados por linha e coluna.

Questão 2: Faça um programa que leia uma matriz 5x5 de números reais, calcule e mostre sua matriz transposta correspondente.

Questão 3: Faça um programa que leia duas matrizes 4x4 de números reais, calcule e mostre a matriz resultante da multiplicação entre elas.

Questão 4: Faça um programa que receba como entrada o número de alunos de uma turma e o número de provas que eles fizeram ao longo de um trimestre. Em seguida, o programa deve pedir as notas de todos os alunos em cada uma das provas e calcular a média de cada aluno e a média de cada prova. Ao final o programa deve mostrar as médias dos alunos, as médias de cada prova e a média geral da turma.

2. Particularidades do Java

O Java não suporta matrizes diretamente, fazendo com que estas sejam vetores unidimensionais cujos elementos também são vetores unidimensionais. Isso permite que os elementos do vetor principal (que formariam as linhas de uma matriz) possam ser vetores de tamanhos diferentes.
A seguir, mostramos um exemplo em que temos um vetor bidimensional com número de colunas diferentes. O código para imprimir este vetor bidimensional é modificado, pois aquele anteriormente utilizado imprime apenas matrizes cujos números de colunas são iguais para todas as linhas.

```
public class UsingMatrixJava2 {
    public static void main(String[] args) {
        int matrix[][] = { {4, 78, 89},{-7, 36, 81, 32, -23},{2, 45, 53,25},
                {12, 17},{65, -34, 47, 8, 1} };
        for (int line = 0; line < matrix.length; line++) {
            for (int column = 0; column < matrix[line].length; column++) {
                System.out.printf("%d ",matrix[line][column]);
            }
            System.out.println();
        }
    }
}
```

3. A estrutura For aprimorada em JAVA para utilização com vetores multidimensionais

Há uma estrutura for aprimorada para iterar com vetores em Java. Esta estrutura pode, inclusive, trabalhar com vetores multidimensionais e fará a iteração pelos elementos do vetor, independentemente de se conhecer o tamanho ou os índices do vetor. Um exemplo de utilização desta estrutura for aprimorada aplicada a vetores multidimensionais pode ser visto no código abaixo. Note que o número de elementos (colunas) em cada linha é diferente.

```
public class UsingMatrixJava3 {

    public static void main(String[] args) {
        int matrix[][]={{4,78,89},{-7,36,81,32,-23},{2,45,53,25},{12,17},
                        {65,-34,47,8,1}};

        for (int vector[] : matrix) {
            for (int element : vector) {
                System.out.printf("%d ",element);
            }
            System.out.println();
        }
    }
}
```