2.2. MATRIZES BIDIMENSIONAIS E MULTIDIMENSIONAIS

No início deste capítulo comparamos matrizes a tabelas, em que os dados estão dispostos em linhas e colunas. Esta é a melhor forma de compreender uma matriz de duas dimensões, a partir da qual matrizes de mais dimensões podem ser imaginadas, embora não possam ser representadas com a mesma facilidade.

Imaginemos uma pequena indústria de roupas e calçados, que vende para três cidades do país. Os produtos desta empresa também estão divididos em três categorias. A tabela a seguir, indica as vendas, para um determinado mês, das três categorias de produtos, em cada uma das três cidades atendidas:

	Camisetas	**Bermudas**	**Calçados**
São Paulo	R$ 9.260,52	R$ 7.086,58	R$ 9.588,66
Rio de Janeiro	R$ 12.063,36	R$ 11.412,25	R$ 8.221,74
Salvador	R$ 11.104,51	R$ 18.668,57	R$ 5.258,88

Podemos representar estes dados através de uma **matriz bidimensional** da seguinte maneira:

```
Dim Vendas(1 To 3, 1 To 3) As Currency
Vendas(1, 1) = 9260.52
Vendas(1, 2) = 7086.58
Vendas(1, 3) = 9588.66
Vendas(2, 1) = 12063.36
Vendas(2, 2) = 11412.25
Vendas(2, 3) = 8221.74
Vendas(3, 1) = 11104.51
Vendas(3, 2) = 18668.57
Vendas(3, 3) = 5258.88
```

Para encontrar o menor e o maior índice disponíveis em uma dada dimensão de uma matriz, utilizamos as funções pré-definidas `LBound` e `UBound`, como no caso de arranjos.

No entanto, como **cada dimensão** pode possuir os seus **próprios índices**, é preciso informar às funções a qual dimensão estamos nos referindo. Note que as dimensões de uma matriz no VBA são contadas **a partir de um, e não de zero**:

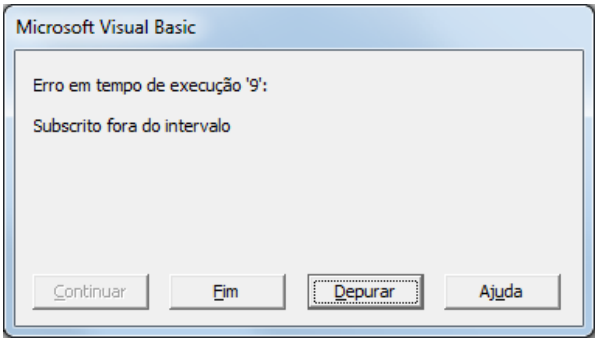
```
LBound(Vendas, 1)

UBound(Vendas, 1)

LBound(Vendas, 2)

UBound(Vendas, 2)
```

Caso tentemos acessar uma **dimensão inexistente de uma matriz**, ocorrerá o **mesmo erro** em tempo de execução que aprendemos anteriormente, e que é ativado quando tentamos acessar um **elemento inexistente** de um arranjo:



No VBA, além de matrizes bidimensionais, podemos criar matrizes de dimensões maiores, como matrizes **tridimensionais** (que você pode imaginar como um cubo de dados), **tetradimensionais**, **pentadimensionais** etc.

No nosso exemplo anterior, apresentamos uma matriz bidimensional que representava os dados de vendas **em um determinado mês**, de produtos vendidos em três cidades (dimensão 1) e pertencentes a três categorias (dimensão 2).

Podemos introduzir neste exemplo, informações relativas a **diversos meses consecutivos**, apresentando uma **nova dimensão** para os dados. Neste caso, a melhor forma de armazená-los temporariamente poderia ser uma **matriz tridimensional**:

Janeiro:

	Camisetas	Bermudas	Calçados
São Paulo	R$ 9.260,52	R$ 7.086,58	R$ 9.588,66
Rio de Janeiro	R$ 12.063,36	R$ 11.412,25	R$ 8.221,74
Salvador	R$ 11.104,51	R$ 18.668,57	R$ 5.258,88

Fevereiro:

	Camisetas	Bermudas	Calçados
São Paulo	R$ 12.588,35	R$ 11.258,96	R$ 6.147,35
Rio de Janeiro	R$ 17.863,19	R$ 13.473,28	R$ 8.100,25
Salvador	R$ 21.265,35	R$ 21.845,66	R$ 4.148,95

Março:

	Camisetas	Bermudas	Calçados
São Paulo	R$ 11.587,44	R$ 10.258,25	R$ 4.258,96
Rio de Janeiro	R$ 25.625,98	R$ 15.753,28	R$ 7.985,99
Salvador	R$ 30.918,48	R$ 29.785,23	R$ 4.025,58

Tente representar estes dados em uma matriz tridimensional em que a **primeira dimensão** represente os **meses**, a **segunda** represente as **cidades** e a **terceira** represente as **categorias de produtos**. Você consegue imaginar situações análogas em que matrizes de mais dimensões seriam úteis?

Antes de passarmos para o próximo assunto, é preciso ressaltar que matrizes de quaisquer dimensões, assim como os arranjos, podem possuir **qualquer tipo**, podem ser declaradas como variáveis **locais**, **privadas** ou **públicas** e podem ser **fixas** ou **dinâmicas**.

Quanto maior o número de dimensões (e de elementos comportados em cada dimensão), mais se torna importante a preocupação com o **espaço na memória**

que será ocupado e com o **tempo em que as matrizes permanecerão disponíveis**.

Assim, escolha em cada caso o **tipo de dados**, o **escopo** e as **dimensões** mais convenientes, levando em conta as limitações das máquinas em que suas rotinas serão executadas.

2.3. OBJETO DICTIONARY

Enquanto **arranjos e matrizes** permitem armazenar elementos e encontrá-los posteriormente através de **um ou mais índices numéricos** (um índice para cada dimensão), os dicionários possibilitam a associação entre **valores** e **chaves não necessariamente numéricas**, como no exemplo seguinte:
