FAESA
CENTRO UNIVERSITÁRIO

Arrays em Java

Apresentação

A principal característica das estruturas arrays (vetores e matrizes) é a possibilidade de se criar muitas variáveis de mesmo tipo de dado e que serão acessadas pelo mesmo nome pelo deslocamento das posições em sua estrutura, os índices. Essa situação simplifica consideravelmente a escrita de programas de computador. O computador continua alocando o mesmo espaço de memória, mas o desenvolvedor, utilizando comandos de repetição apropriados, tem facilidade na programação. Fazendo uso das estruturas arrays, os desenvolvedores têm a possibilidade de transportar o mundo real para o mundo computacional.

Nesta Unidade de Aprendizagem, você aprenderá o que são vetores, matrizes e de que forma o computador manipula essas estruturas de dados. Utilizando a linguagem de programação Java, vai ser possível saber como utilizar essas estruturas de dados. Para você entender como vetores e matrizes auxiliam na informatização de situações do mundo real, serão mostrados alguns exemplos.

Bons estudos.

Ao final desta Unidade de Aprendizagem, você deve apresentar os seguintes aprendizados:

- Definir vetor e matriz.
- Explicar declaração, inicialização e acesso em *arrays*.
- Exemplificar a utilização de vetor e matriz em Java.

Padrão de resposta esperado
Para resolver o desafio, é necessário utilizar uma estrutura de dados matriz em Java para armazenar os dados do tabuleiro do jogo da velha. Como conteúdo da matriz, pode ser utilizado um valor (1, por exemplo) para representar a bolinha do jogo da velha e outro valor (2, por exemplo) para representar o X do jogo da velha.

- Para identificar a posição na qual o usuário deseja jogar, ele deverá informar a linha e coluna da posição da matriz (representando o jogo da velha) na qual será inserido o valor que representa o jogador da vez.

- A cada jogada, o programa deve controlar o jogador da vez, por uma variável de controle. A partir da 3ª jogada, o programa deve verificar se algum jogador venceu, varrendo as linhas, colunas e diagonais da matriz, verificando se existem 3 valores que representam um dos jogadores em alguma das varreduras realizadas. Caso existam 3 valores iguais em uma das varreduras, o jogo finalizou e o jogador ganhador é o da vez. Caso já tenha finalizado o número de jogadas permitidas e ninguém tenha vencido, acontece a "velha".

- Enquanto o usuário não finalizar o jogo, o programa deve pontuar, em um placar de 2 jogadores, o jogador que venceu a partida.

- Para controlar a jogada do computador, o programa deve varrer a matriz e identificar a posição da jogada baseado na probabilidade de vencer o jogo. Da mesma forma que o jogador humano avalia as possibilidades de jogada, o computador também deverá fazê-lo.

- Quando o usuário desejar parar o jogo, deve ser informado o placar final das jogadas.

Esse desafio pode ser resolvido de forma amadora, validando item a item, ou utilizando valores nas posições da matriz que direcionam para os resultados por somatório. De qualquer forma, as regras solicitadas devem fazer parte do programa em Java.

Desafio

Os *games* atualmente estão contagiando praticamente todas as faixas etárias. Eles podem ser utilizados em PCs, *tablets* e *smartphones*. Jogos antigos de tabuleiro ou até os jogos realizados em papel estão disponíveis de forma digital e conseguem encantar diversas gerações.

Mas como transpor os tabuleiros dos jogos para programas de computador? Uma das facilidades da disponibilidade de estruturas de dados na computação é justamente a possibilidade de poder organizar o mundo real na memória do computador.

Imagine que você trabalha em uma *startup* que desenvolve *games* clássicos e sua tarefa é **programar o jogo da velha na linguagem Java**. As regras são:

1. Um jogador é o usuário (jogador humano) e o outro é o computador.

2. O programa deve controlar o jogador da vez.

3. O programa deve identificar o final do jogo.

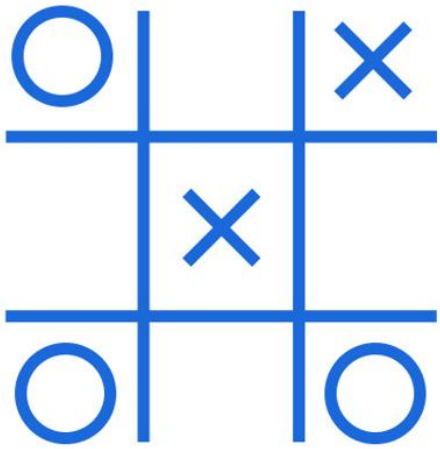
4. O programa deve informar qual jogador venceu a partida ou se deu "velha".

5. O programa deve permitir diversas rodadas do jogo, até o usuário desejar finalizar.

6. O programa deve informar um placar final quando o usuário finalizar as rodadas.

7. O computador deve "ser inteligente", buscando vencer a partida, e não jogar de forma aleatória.

8. Você deve utilizar, necessariamente, estrutura de dados do tipo *array* para desenvolver esse desafio.



A partir das regras apresentadas, explique como você pode programar o jogo da velha em Java.

- Cria uma matriz 2D com arrays vazios representando os espaços a preencher.
- Criar um função de exibição do tabuleiro
- Função de atividade do computador que busque ganhar e não preencher de forma aleatoria, então o foco seria - preencher as linhas e também bloquear o player
- Verificação da situação de empate
- Loop para repetição do jogo a critério do player
- Contagem para exibição na finalização do jogo

Infográfico

Cada linguagem de programação tem sua sintaxe específica. A sintaxe determina como o comando deve ser escrito na linguagem de programação. Particularmente, em linguagens de programação, a sintaxe dos comandos precisa ser obedecida à risca, para que o compilador consiga entender e realizar a operação solicitada.

Neste Infográfico, você verá a sintaxe da declaração e uso de vetores e matrizes bidimensionais em Java.

ESTRUTURA DE DADOS

As linguagens de programação são formadas por um conjunto de regras sintáticas e semânticas, que precisam ser seguidas para que o computador consiga compilar e gerar as instruções para processar o programa.

Para a escrita de programas, é necessário seguir as regras de sintaxe e semântica que são próprias de cada linguagem de programação.

Como é a sintaxe da linguagem java para os comandos de vetores?

- Declaração de vetores:

```
<tipo de dado> <nome do vetor> [];
```

- Alocação de espaço de memória para o vetor:

```
Vetor = new <tipo de dado> [<tamanho do vetor>];
```

Combinando a declaração do vetor + alocação de memória:

```
<tipo de dado> vetor [] = new <tipo de dado> [<tamanho do vetor>];
```

(*) em Java, o índice dos *arrays* inicia em 0.

DECLARAÇÃO DE VETORES. + ALOCAÇÃO DE ESPAÇO DE MEMÓRIA PARA O VETOR. → COMBINANDO A DECLARAÇÃO DO VETOR + ALOCAÇÃO DE MEMÓRIA.

E para acessar o conteúdo do vetor?

78	234	54	-67	34	0	← Conteúdo
0	1	2	3	4	5	← Índice

- Sempre que se utilizar o conteúdo de uma posição do vetor, é preciso referenciar da seguinte forma:

```
<nome do vetor> [<índice>]
```

- Qualquer comando conhecido de Java pode ser utilizado com vetor, bastando referenciar.

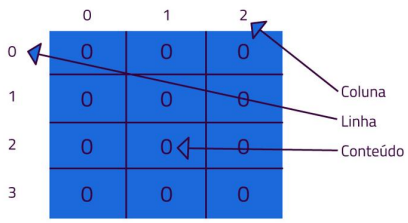
Como é a sintaxe da linguagem java para os comandos de matriz bidimensional?

- Declaração combinada de matrizes bidimensionais:

```
<tipo de dado> vetor [][] = new <tipo de dado> [<número de
linhas>][<número de colunas>];
```

(*) em Java, o índice dos *arrays* inicia em 0.

Para acessar o conteúdo de uma matriz



- Para acessar o conteúdo de uma matriz, é preciso indicar a linha e a coluna:

```
<nome da matriz> [<valor da linha>][<valor da coluna>]
```

- Aqui, também, qualquer comando da linguagem Java pode ser utilizado com matriz, sendo necessário indicar o valor da linha e da coluna desejada.

O compilador Java vai indicar erro de sintaxe e/ou semântica caso o comando não tenha sido utilizado de forma correta. Mas **conhecer a sintaxe da linguagem de programação é imprescindível para conversar com o computador**.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Conteúdo do Livro

A definição de estruturas de dados na programação envolve desde a criação do novo tipo de dado até as regras de manipulação dessas novas estruturas. Entender as regras de manipulação é o segredo para utilizar as estruturas de dados de forma eficiente nas aplicações computacionais. As estruturas de dados do tipo *array*, que envolvem vetores e matrizes, são as mais simples e entender o funcionamento delas é um início promissor para uma programação eficiente.

No capítulo ***Arrays* em Java**, base teórica desta Unidade de Aprendizagem, você entenderá as estruturas de dados *arrays*, aprenderá como declarar os *arrays* na linguagem Java e como aplicar em problemas do mundo real.

Boa leitura.

ESTRUTURA DE DADOS EM JAVA

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Arrays em Java

Patricia Rucker de Bassi

OBJETIVOS DE APRENDIZAGEM

- Definir vetor e matriz.
- Explicar declaração, inicialização e acesso em *arrays*.
- Exemplificar a utilização de vetor e matriz em Java.

Introdução

À medida que os desafios computacionais aumentam, as variáveis básicas dos tipos inteiro, real, *string* e *boolean* não são suficientes para podermos resolvê-los. Então, surge a necessidade de utilizarmos as estruturas de dados, que são basicamente a organização de conjuntos de dados na memória do computador. Entender as estruturas de dados e saber aplicá-las corretamente facilita o manuseio de uma grande quantidade de dados, melhorando o desempenho para a aplicação e tornando seus códigos eficientes.

Na maioria das linguagens de programação, há diversas estruturas de dados disponíveis, das mais simples às mais complexas. Os *arrays* são as estruturas de dados mais simples, e entender seu funcionamento e aplicação é muito importante para um desenvolvedor.

Neste capítulo, você vai ler sobre a criação de estruturas de dados no desenvolvimento de programas computacionais e o conceito de estrutura de dados do tipo *array*. Além disso, vai conhecer a linguagem de programação Java, como declarar vetores e matrizes e como manipular seus respectivos conteúdos para trazer respostas computacionais às necessidades dos usuários. Você vai ver alguns

exemplos de programas em Java utilizando vetores e matrizes bidimensionais com o objetivo de contextualizar a aplicação de *arrays* nas necessidades reais do desenvolvimento de sistemas computacionais.

Estruturas de dados do tipo *array*

Considere que um professor precise verificar quantos, dos 45 alunos da turma, receberam nota abaixo da média da turma em uma avaliação. Para obter essa informação, será necessário somar as notas da avaliação de cada aluno e dividir por 45, cujo resultado será a média da turma. Então, o professor vai comparar a nota da avaliação de cada aluno com a média da turma. Se a nota do aluno for menor em relação à média da turma, será preciso somar mais um no total de alunos que tiraram nota abaixo dessa média. Após manipular os dados, a informação que o professor precisa estará na variável que somou o número de alunos com nota menor em comparação à média da turma. Veja como fica esse programa em Java.

```
      double nota, soma=0, media;

       int i, abaixo=0;          // número de alunos abaixo
da média


       for (i=1; i<=45; i++) {
              nota = ler.nextDouble();
              soma = soma + nota;
              }
       media = soma/45;


       for (i=1; i<=45; i++) {
              nota = ler.nextDouble();
              if (nota < media)
                     abaixo++;
              }
```

```
        System.out.printf("Número de alunos abaixo da média
da turma é de %d", abaixo);
```

Você deve ter percebido que nesse código o professor vai precisar digitar as notas dos 45 alunos duas vezes: uma para calcular a média da turma e outra para comparar a nota de cada aluno com a média da turma. Esse é um programa pouco eficiente para o problema proposto.

O problema desse programa é que só temos uma variável para guardar a nota dos 45 alunos. Então, outra proposta de solução seria uma variável de nota para cada aluno. Veja como fica esse programa em Java.

```
      double n1,n2,n3, ..., n45, media;       // declarar as
45 variáveis

       int i, abaixo=0;                          // número de
alunos abaixo da média


       n1 = ler.nextDouble();

       n2 = ler.nextDouble();

       n3 = ler.nextDouble();

       .

       .                                    // ler as 45 notas

       .

       n45 = ler.nextDouble();


       soma = n1 + n2 + n3 + ... + n45;       //somar as 45
notas

       media = soma/45;


       if (n1 < media)

              abaixo++;

       if (n2 < media)
```

```
            abaixo++;
        if (n3 < media)
            abaixo++;
        .
        .
        .                                   // comparar as 45 notas
        if (n45 < media)
            abaixo++;

        System.out.printf("Número de alunos abaixo da média
da turma é de %d", abaixo);
```

Dessa forma, o professor só precisa digitar uma vez a nota dos 45 alunos. Porém, nessa proposta, o desenvolvedor precisou digitar um código muito mais longo e com comandos repetidos, e ficou pouco eficiente para o desenvolvedor. Esse exemplo mostra que é preciso termos na memória do computador um considerável conjunto de dados a serem manipulados sem dificultar o trabalho dos usuários ou dos desenvolvedores. Situações como essa trouxeram a necessidade de se criar estruturas de dados para manipular com mais facilidade conjuntos de dados.

A estrutura de dados também é chamada de estrutura abstrata de dados, pois o computador continua alocando espaço de memória para a variável igual a uma variável simples, sendo diferente apenas na forma de organização e manipulação dos conteúdos desses espaços de memória. É como se estivéssemos ditando uma regra, um comportamento para esse conjunto de dados. Basicamente, sempre que criarmos ou definimos uma nova estrutura abstrata de dados precisaremos definir também o comportamento que essa estrutura abstrata de dados terá; isto é, como iremos manipular o conteúdo que será armazenado nessa nova estrutura definida (TENENBAUM; LANGSAM; AUGENSTEIN, 1995).

As estruturas de dados mais simples são as conhecidas como *arrays*. Com as necessidades computacionais aumentando, tornando os códigos mais complexos, foi necessário criar outras formas abstratas de organização de conjunto de dados. Então, surgiram as estruturas abstratas de dados conhecidas como pilhas, filas, listas, árvores e grafos.

Os *arrays* são estruturas de dados compostas por um conjunto de dados básicos, como os inteiros, reais, caracteres e *booleans*. A particularidade dos *arrays* é que seu conteúdo é homogêneo, isto é, composto somente por um tipo de dado básico. Veja alguns exemplos a seguir.

1. Um *array* que contém os nomes dos 45 alunos de uma turma será um *array* de caracteres.
2. Um *array* que contém as 4 médias bimestrais de um aluno da turma será um *array* de valores reais.
3. Um *array* que contém as 4 médias bimestrais de todos os 45 alunos da turma será um *array* no formato de uma tabela de valores reais.

Nos exemplos 1 e 2, se percebe que é representado um conjunto em forma de lista de valores. Já no exemplo 3, o conjunto de valores é representado em forma de tabela. E aí está a diferença entre vetores e matrizes, as estruturas abstratas de dados do tipo *array*.

Um *array* do tipo **vetor** é uma estrutura de dados unidimensional homogênea. O vetor é unidimensional porque é uma lista, tem somente uma dimensão, e é homogêneo porque o conjunto de dados que ele pode armazenar precisa ser de apenas um tipo de dado básico – inteiros, reais, caractere ou *boolean* (TENENBAUM; LANGSAM; AUGENSTEIN, 1995). Um *array* do tipo **matriz** é considerado uma estrutura de dados multidimensional homogênea. Uma matriz pode ser bidimensional (uma tabela), tridimensional (um cubo) e até multidimensional. A forma mais comum de utilizar matriz é no formato bidimensional, como uma tabela formada por linhas e colunas. A matriz é dita homogênea porque, assim como o vetor, todo o seu conjunto de dados deve ser de um mesmo tipo básico de dados – inteiro, real, caractere ou *boolean* (TENENBAUM; LANGSAM; AUGENSTEIN, 1995).

Saiba mais

Você sabia que o tipo de dado *String* é um vetor de caracteres (uma letra, por exemplo)?

Uma *String* é um conjunto de caracteres dispostos em forma de lista — unidimensional, contendo um tipo de dado básico —, o *char*; logo, ela é um *array* do tipo vetor. As linguagens de programação, como o Java, têm uma biblioteca de funções para trabalhar a estrutura de dados *String*, pois esse é um recurso muito utilizado pelos desenvolvedores. Bibliotecas como essas facilitam e deixam os códigos mais eficientes.

Vetores e matrizes em Java

Vetores e matrizes são estruturas de dados que armazenam uma quantidade fixa de dados e de somente um tipo básico. Nessas estruturas, o dado é o conteúdo do vetor ou matriz, o que significa que temos muitos dados dentro de uma estrutura vetor ou matriz, e, para acessarmos um dado dentro dessas estruturas, utilizamos um índice. Pode-se dizer, então, que os *arrays* são estruturas indexadas, sendo necessário um índice para localizar e acessar o elemento do vetor ou matriz — o dado armazenado.

Observe o seguinte exemplo. A tele-entrega da pizzaria de sua cidade entrega uma pizza para o João, que mora na rua XV de Novembro. Se o entregador não tiver a informação do número da casa do João, ele não conseguirá entregar a pizza. Comparando a rua XV de Novembro com o nome de um vetor e o número da casa do João com o índice do vetor, entende-se que o compilador também precisa saber o nome do vetor (ou da matriz) e o índice da posição onde se encontra o dado que se deseja manipular.

Os índices são sempre variáveis inteiras e positivas para que possam fazer o papel de identificadores das posições nos *arrays*. O Java é uma linguagem com *arrays zero-based*, o que significa que as posições dos *arrays* iniciam a numeração a partir do valor 0. Por exemplo, um vetor com 10 posições (tamanho 10) teria índices de 0 até 9. É possível utilizar um elemento do vetor ou matriz de forma individual sem nenhuma regra ou ordem preestabelecida, e isso significa que as estruturas de dado do tipo *array* são de acesso aleatório.

Utilizando vetores

O vetor é um *array* unidimensional homogêneo, e o número de posições de um vetor é exatamente o tamanho dele. Assim, um vetor de tamanho 10 pode armazenar 10 elementos distintos. Para recuperarmos cada elemento distinto do vetor, é utilizado o índice. Portanto, pode-se dizer que cada elemento de um vetor é identificado de forma única por um índice inteiro e positivo, e que o vetor é uma estrutura linear sequencialmente numerada (ZIVIANI, 2006). A Figura 1 apresenta uma estrutura vetor de seis posições, seus conteúdos e respectivos índices.

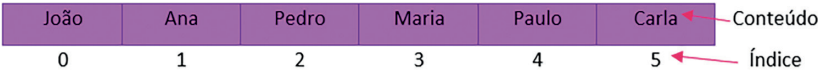


Figura 1. Exemplo de estrutura de dados vetor.

Declarando vetores

Para declararmos uma estrutura de dados vetor em um programa Java, é preciso fornecer algumas informações para o compilador criar uma nova estrutura de dados no programa (GOODRICH; TAMASSIA, 1997):

- o nome do vetor;
- o tamanho do vetor (quantas posições ele terá);
- o tipo de dado básico que o vetor poderá conter (*int*, *double*, *float*, *char*, *String* ou *boolean*).

O compilador precisa alocar espaço de memória para essa nova estrutura, e para fazer isso em Java é utilizado o operador *new*. Declarando um vetor de 10 posições inteiras:

```
int v[];                // declaração do vetor V

v = new int[10];        // alocação de espaço para V
```

Estas duas declarações podem ser combinadas em uma só. Veja como ficaria:

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

Veja outros exemplos de declaração de vetores:

```
String nome[] = new String[45];

float nota[] = new float[45];

double matricula[] = new double[45];
```

Os elementos dos vetores podem ser acessados de forma aleatória por meio de seus índices. Para fazer referência às posições do vetor, é preciso indicar o nome do vetor e o seu índice. Como demonstrado no exemplo da tele-entrega de pizza, precisamos saber o nome da rua e o número da casa para fazer a entrega.

Acessando uma posição do vetor V:

```
v[i] = i * 5;

System.out.println(v[i]);

if (v[2] % 2 == 0)
```

Como os índices são valores inteiros, positivos e sequenciais, para percorrer linearmente as posições de um vetor o comando `for` é bastante útil. Veja um exemplo de atribuição de valores a um vetor:

```
for (i=0; i<10; i++)

     v[i] = i * 5;
```

O exemplo a seguir mostra um trecho de código em Java que atribui valor para as posições de um vetor e imprime seu conteúdo:

```
for (i=0; i<10; i++)

     v[i] = i * 5;



 for (i=0; i<10; i++)

     System.out.printf("%d --> %d", i, v[i]);
```

Também é possível inicializar os vetores no momento de fazer a sua declaração. Veja o exemplo dos dias da semana, que são valores pré-definidos e não vão mudar durante a execução do programa:

```
String semana[] = {"domingo", "segunda", "terça", "quarta",
"quinta", "sexta", "sábado"};
```

Utilizando matrizes

Arrays do tipo matriz são estruturas de dados multidimensionais homogêneas. As matrizes podem ter de 2 a *n* dimensões. Uma matriz de 2 dimensões são estruturas planas como as tabelas; na matriz de 3 dimensões há a profundidade, como nos cubos; e a partir de 4 dimensões fica difícil de visualizar, mas sua utilidade é imensa (ZIVIANI, 2006).

Aplicando essas matrizes na prática, imagine a representação de uma rede de franquias de uma loja de chocolates:

- Em uma tabela, podemos ter linhas representando o estado e, na coluna, a cidade. O conteúdo da tabela pode ser o número de lojas de chocolates em cada cidade de cada estado do Brasil.
- Em um cubo, podemos ter na linha, o estado; na coluna, a cidade; e na profundidade, o mês do ano. Seu conteúdo pode ser o faturamento de cada mês em cada cidade de cada estado do Brasil.

- Em uma matriz de quatro dimensões, podemos ter o estado, a cidade, o mês do ano e os produtos da loja. O conteúdo da matriz pode ser a quantidade vendida de cada produto, em cada mês, por cada cidade de cada estado do Brasil.

Pode-se perceber que um *array* do tipo matriz pode armazenar uma quantidade imensa de valores, e com significado. Isso tudo estará na memória do computador, podendo ser manipulado pelo programa. A Figura 2 mostra uma matriz bidimensional de tamanho 4 × 3, isto é, 4 linhas por 3 colunas. Lembre-se que em Java os *arrays* iniciam em 0 (*arrays zero-based*).

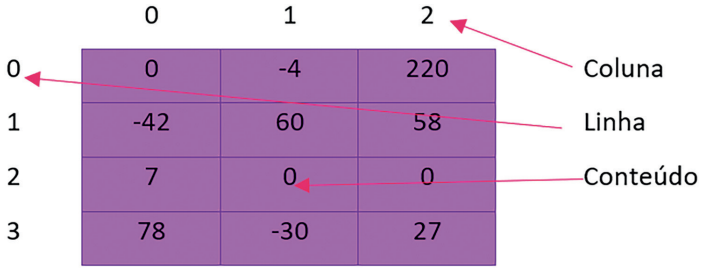


Figura 2. Exemplo de estrutura de dados matriz bidimensional.

Para acessar um elemento de uma estrutura de dados do tipo matriz, é preciso saber em que linha e coluna o elemento está. No exemplo da Figura 2, o conteúdo da matriz linha 0, coluna 1, é -4. Considerando que a matriz tenha o nome M, dizemos que M_{01} = -4. Se aplicarmos esse conceito em matrizes de 3 dimensões, será preciso informar o valor da linha, coluna e profundidade; no caso de matrizes de 4 dimensões, será preciso informar o valor das 4 dimensões da matriz para poder acessar o conteúdo; e da mesma forma em matrizes de *n* dimensões, em que precisaremos de *n* índices para acessar o conteúdo da matriz.

Fique atento

Lembre-se que as linhas, as colunas, a profundidade e outras dimensões da matriz serão controladas pelos índices. Para acessarmos o conteúdo de uma matriz, teremos a quantidade de índices e dimensões que a matriz tiver, e cada índice será representado no programa por uma variável inteira, positiva e diferente, iniciando em 0 (*arrays zero-based*) (ZIVIANI, 2006).

Declarando matrizes

Nesta seção, serão abordadas as matrizes bidimensionais, especificamente. Mas caso você precise utilizar matrizes com mais dimensões, é possível apenas aumentar o número de índices na indexação da matriz, e o conteúdo será acessado.

Em Java, não há uma declaração específica para tipos de dado matriz, então cria-se um vetor de vetor, de forma que dentro de cada posição do vetor houvesse um outro vetor. Para declararmos uma matriz bidimensional em um programa Java, são utilizados os mesmos conceitos da declaração de vetores (GOODRICH; TAMASSIA, 1997).

Declarando uma matriz 4 × 3 de conteúdo inteiro:

```
int m[][] = new int [4][3];  //declaração combinada da matriz M
```

Veja outros exemplos de declaração de matriz:

```
float produto [][] = new float [8][10];

String filiais [][] = new String [10][20];

double fatura [][][] = new double [3][4][2];
```

Nas matrizes, também são utilizados os índices para acessar seus conteúdos, e isso pode ser realizado de forma aleatória, sendo necessário saber apenas o valor dos índices. Veja alguns exemplos:

```
produto [0][2] = 120;

filiais [0][0] = "matriz";

fatura [0][j][1] = 0.5;
```

Aqui o comando `for` também é bastante útil para percorrer todos os elementos da matriz, utilizando-se um `for` para cada uma das dimensões da matriz associado à variável que representa o índice da dimensão. Em uma matriz bidimensional, geralmente são utilizados `i` e `j` como índices. Lembrando que para cada linha é preciso percorrer todas as colunas; assim, os comandos `for` são aninhados. Veja:

```
for (i=0; i<4; i++)

       for (j=0; j<3 ; j++)

              m[i][j] = ler.nextInt();
```

O exemplo a seguir mostra um trecho de código em Java que atribui valor para as posições de uma matriz bidimensional 4 × 3 e imprime seu conteúdo.

```
for (i=0; i<4; i++)
        for (j=0; j<3 ; j++)
                m[i][j] = i * 3;


for (i=0; i<4; i++) {
        System.out.println();
        for (j=0; j<3; j++)
                if (m[i][j] % 2 == 0)
                                {
                             System.out.print(m[i][j]);
                             System.out.print("   ");
                    }

        }
```

Também podemos inicializar as matrizes no momento em que fazemos a declaração dela. Veja os exemplos:

```
String filiais [][] = {{"matriz", "depósito"},{"praia central",
"central"}, {"norte", "sul"}};


  String filiais [][] = new String [3][2];
  filiais [0][0] = "matriz";
  filiais [0][1] = "depósito";
  filiais [1][0] = "praia central";
  filiais [1][1] = "central";
  filiais [2][0] = "norte";
filiais [2][1] = "sul";
```

Saiba mais

Em alguns casos, não temos a informação do tamanho do vetor ou da matriz. Mas você sabia que existe uma função que retorna o tamanho do *array*? É a função `length`. Veja.

- Para saber o tamanho do vetor V, utiliza-se `v.length`.
- Para saber o número de linhas da matriz m, utiliza-se `m.length`.
- Para saber o número de colunas da linha i da matriz m, utiliza-se `m[i].length`. Lembre-se que essa função vai retornar um valor inteiro.

Aplicando vetores e matrizes

Nos exemplos a seguir, serão demonstrados alguns conceitos importantes em relação a vetores e matrizes, e um dos importantes é o de posição e conteúdo do vetor ou matriz.

Exemplo 1

Há 10 mesas em um restaurante, e cada grupo de pessoas que ocupa uma mesa tem seu nome associado a ela. O gerente do restaurante fez uma promoção no dia 1º de abril, oferecendo uma sobremesa aos ocupantes das mesas de números pares. Escreva um programa em Java que indique o nome do grupo das mesas de números pares que ganharão a sobremesa ao final do jantar.

Para resolver esse problema, precisamos ler o nome dos ocupantes das cinco mesas do restaurante e indicar na resposta somente o nome dos grupos cujos índices sejam pares. Aqui, estamos trabalhando com a **posição par**, que é indicada pelo índice `i`. Como em Java temos *array zero-based* e as mesas do restaurante não iniciam no número 0, e sim no número 1, vamos trabalhar com `i+1` para localizar as mesas pares (`i+1 % 2 == 0`).

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

int i;


for (i=0; i<5; i++) {

   System.out.printf("Grupo da mesa %d: ", i+1);

   grupo[i] = ler.next();

}
```

```
        System.out.println();

        for (i=0; i<5; i++)

                if ((i+1) % 2 == 0) {

                        System.out.printf("A mesa %d foi pre-
miada ", i+1);

                 System.out.printf("grupo do Sr(a) ");

                 System.out.printf(grupo[i]);

                 System.out.println();

                }
```

Resposta do programa:

- Grupo da mesa 1: Ana.
- Grupo da mesa 2: Pedro.
- Grupo da mesa 3: Maria.
- Grupo da mesa 4: Antonio.
- Grupo da mesa 5: João.

De acordo com o programa, a mensagem final é:

A mesa 2 foi premiada grupo do Sr(a) Pedro.
A mesa 4 foi premiada grupo do Sr(a) Antonio.

Exemplo 2

O jogo de corrida Runner trabalha com a pontuação de seus pilotos nas corridas, e a promoção do dia 1º de abril dará um crédito de 100 pontos aos pilotos que tiverem pontuação ímpar. O administrador do jogo precisa saber quantos pilotos ganharam os créditos. Cada bateria de corrida do Runner permite 5 pilotos.

Para resolver esse problema, precisamos encontrar as pontuações de cada piloto, que são conteúdos do vetor. Portanto, precisamos acessar o conteúdo de cada posição do vetor para saber se são impares, ou não. O acesso ao conteúdo do vetor é realizado pelo `<nome do vetor> [<índice>]`.

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

      int i, qtd=0;
```

```
for (i=0; i<5; i++) {
   System.out.printf("Pontos do piloto %d: ", i+1);
   point[i] = ler.nextInt();
}
System.out.println();
for (i=0; i<5; i++)
       if (i % 2 != 0) {
              qtd++;
              point[i]= point[i] + 100;

              }
System.out.printf("O número de pilotos que ganharam
100 pontos foi de %d ", qtd);
```

Resposta do programa:

- Pontos do piloto 1: 233.
- Pontos do piloto 2: 346.
- Pontos do piloto 3: 768.
- Pontos do piloto 4: 457.
- Pontos do piloto 5: 1234.

O número de pilotos que ganharam 100 pontos foi de 2.

Exemplo 3

Um professor aplicou uma avaliação objetiva de 10 questões entre seus 45 alunos, em que cada questão vale 1 ponto. Para facilitar a correção da prova, o professor criou um gabarito das respostas das questões, e ele precisa que o programa informe a nota que cada aluno obteve na prova.

Para resolver esse problema, trabalharemos com dois vetores: um do gabarito da prova e outro da resposta do aluno para a questão. As posições dos dois vetores (índices) estão associadas e dizem respeito à mesma ques-

tão. O conteúdo de cada vetor contém as respostas, então, se o conteúdo do gabarito for igual ao conteúdo da resposta do aluno, no mesmo índice, o aluno acertou a questão e tem 1 ponto a ser acrescido na nota. O resultado do processamento do programa considerou somente 3 alunos como exemplo.

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

String resposta[] = new String[10];

int i, aluno, nota;


  for (i=0; i<10; i++) {

          System.out.printf("Gabarito questão %d: ", i+1);

          gabarito[i] = ler.next();

       }

       System.out.println();

       for (aluno=0; aluno<45; aluno++) {

          nota=0;

         for (i=0; i<10; i++) {

            System.out.printf("Resposta do aluno %d para a
questão %d: ", aluno+1, i+1);

            resposta[i] = ler.next();

            if (gabarito[i].equals(resposta[i]))

               nota++;

            }

         System.out.println();

        System.out.printf("O aluno %d obteve nota %d \n\n",
aluno+1, nota);

         }

}
```

Resposta do programa:

- Gabarito questão 1: a.
- Gabarito questão 2: b.
- Gabarito questão 3: c.
- Gabarito questão 4: d.
- Gabarito questão 5: e.
- Gabarito questão 6: a.
- Gabarito questão 7: b.
- Gabarito questão 8: c.
- Gabarito questão 9: d.
- Gabarito questão 10: e.

- Resposta do aluno 1 para a questão 1: a.
- Resposta do aluno 1 para a questão 2: b.
- Resposta do aluno 1 para a questão 3: c.
- Resposta do aluno 1 para a questão 4: d.
- Resposta do aluno 1 para a questão 5: e.
- Resposta do aluno 1 para a questão 6: a.
- Resposta do aluno 1 para a questão 7: b.
- Resposta do aluno 1 para a questão 8: c.
- Resposta do aluno 1 para a questão 9: d.
- Resposta do aluno 1 para a questão 10: e.

O aluno 1 obteve nota 10.

- Resposta do aluno 2 para a questão 1: b.
- Resposta do aluno 2 para a questão 2: b.
- Resposta do aluno 2 para a questão 3: b.
- Resposta do aluno 2 para a questão 4: b.
- Resposta do aluno 2 para a questão 5: b.
- Resposta do aluno 2 para a questão 6: b.
- Resposta do aluno 2 para a questão 7: b.
- Resposta do aluno 2 para a questão 8: b.
- Resposta do aluno 2 para a questão 9: b.
- Resposta do aluno 2 para a questão 10: b.

O aluno 2 obteve nota 2.

- Resposta do aluno 3 para a questão 1: a.
- Resposta do aluno 3 para a questão 2: b.
- Resposta do aluno 3 para a questão 3: a.
- Resposta do aluno 3 para a questão 4: b.
- Resposta do aluno 3 para a questão 5: a.
- Resposta do aluno 3 para a questão 6: b.
- Resposta do aluno 3 para a questão 7: a.
- Resposta do aluno 3 para a questão 8: b.
- Resposta do aluno 3 para a questão 9: a.
- Resposta do aluno 3 para a questão 10: b.

O aluno 3 obteve nota 2.

Percebe-se que para associar vetores podemos utilizar o mesmo índice entre eles. Essa associação pode acontecer entre matrizes, e entre vetores e matrizes, dependendo da situação a ser resolvida.

Exemplo 4

Uma loja de *e-commerce* tem disponíveis 250 modelos diferentes de camisetas nos tamanhos PP, P, M, G e GG. O dono da loja sabe quantas camisetas foram vendidas de cada modelo por tamanho, mas agora ele precisa saber quanto foi vendido, no total, de cada modelo.

Para ajudar o proprietário da loja, podemos montar uma estrutura matriz bidimensional para manipular os dados e apresentar a informação que ele deseja. Como são 250 modelos de camiseta em 5 tamanhos diferentes, é possível criar uma matriz 250 x 5 e armazenar no seu conteúdo o dado que o dono da loja tem, que é o número de vendas de cada modelo em cada tamanho. Depois, é só somar todas as colunas de cada linha da matriz e teremos o total de vendas por modelo de camiseta. É possível armazenar a quantidade total de vendas em uma 6ª coluna da matriz. Para isso, vamos criar uma matriz 250 x 6. Nota: o resultado do processamento do programa considerou somente 5 modelos de camiseta como exemplo.

```
int venda[][] = new int[250][6];    // coluna 6 para o total
de vendas do modelo

int i, j;
```

```
String tam;


for (i=0; i<250; i++) {
              for (j=0; j<5; j++) {
                     switch ( j )
                     {
                         case 0:
                            tam = "PP";
                            break;
                         case 1:
                             tam = "P";
                             break;
                         case 2:
                             tam = "M";
                             break;

                         case 3:
                             tam = "G";
                             break;

                         default:
                             tam ="GG";
                     }
                     System.out.printf("Quantidade de vendas
do modelo %d no tamanho %s: ", i+1,tam);
```

```
                    venda[i][j] = ler.nextInt();
                  }
            venda[i][5]=0;
            System.out.println();
            }

      for (i=0; i<250; i++)
            for (j=0; j<5; j++)
                venda[i][5] = venda[i][5] + venda[i][j];

       System.out.println();
       for (i=0; i<250; i++)
           System.out.printf("O total de vendas do modelo
%d de camisetas é %d \n", i+1, venda[i][5]);
```

Resposta do programa:

- Quantidade de vendas do modelo 1 no tamanho PP: 1.
- Quantidade de vendas do modelo 1 no tamanho P: 1.
- Quantidade de vendas do modelo 1 no tamanho M: 1.
- Quantidade de vendas do modelo 1 no tamanho G: 1.
- Quantidade de vendas do modelo 1 no tamanho GG: 1.

- Quantidade de vendas do modelo 2 no tamanho PP: 1.
- Quantidade de vendas do modelo 2 no tamanho P: 1.
- Quantidade de vendas do modelo 2 no tamanho M: 1.
- Quantidade de vendas do modelo 2 no tamanho G: 1.
- Quantidade de vendas do modelo 2 no tamanho GG: 1.

- Quantidade de vendas do modelo 3 no tamanho PP: 1.
- Quantidade de vendas do modelo 3 no tamanho P: 1.
- Quantidade de vendas do modelo 3 no tamanho M: 1.

- Quantidade de vendas do modelo 3 no tamanho G: 1.
- Quantidade de vendas do modelo 3 no tamanho GG: 1.

- Quantidade de vendas do modelo 4 no tamanho PP: 1.
- Quantidade de vendas do modelo 4 no tamanho P: 1.
- Quantidade de vendas do modelo 4 no tamanho M: 1.
- Quantidade de vendas do modelo 4 no tamanho G: 1.
- Quantidade de vendas do modelo 4 no tamanho GG: 1.

- Quantidade de vendas do modelo 5 no tamanho PP: 1.
- Quantidade de vendas do modelo 5 no tamanho P: 1.
- Quantidade de vendas do modelo 5 no tamanho M: 1.
- Quantidade de vendas do modelo 5 no tamanho G: 1.
- Quantidade de vendas do modelo 5 no tamanho GG: 1.

- O total de vendas do modelo 1 de camisetas é 5.
- O total de vendas do modelo 2 de camisetas é 5.
- O total de vendas do modelo 3 de camisetas é 5.
- O total de vendas do modelo 4 de camisetas é 5.
- O total de vendas do modelo 5 de camisetas é 5.

Neste exemplo, foi possível demonstrar o uso de uma matriz para armazenar dados do mundo real e sua manipulação. Para criar a 6ª coluna da matriz, foi utilizado o recurso de fixar uma coluna da matriz. Em vez de utilizar uma variável, podemos fixar o número da linha ou coluna a ser manipulada na matriz.

Estruturas de dados do tipo *array* são as estruturas de dados mais simples, mas já auxiliam bastante no desenvolvimento de programas, trazendo eficiência para as aplicações computacionais.

Exemplo 5

Você precisa realizar o cálculo de multiplicação de matrizes. Para resolver esse problema em programação é muito simples. Considerando, por exemplo, a matriz A multiplicada pela matriz B, que resulta na matriz C, e obedecendo a regra matemática para o cálculo de multiplicação entre matrizes, o comando para realizar o cálculo é o seguinte:

```
for(i = 0; i < linhasdeA; i++){

            for(j = 0; j < colunasdeB; j++){
```

```
                for(k = 0; k < n; k++){
                    C[i][j] = C[i][j] + A[i][k] * B[k][j];
                }
            }
        }
```

São apenas 4 linhas para resolver todo aquele cálculo. Os valores de linhasdeA e colunasdeB podem ser obtidos pela função *length*.

Podem ser resolvidos muitos problemas envolvendo conceitos matemáticos de matrizes, como diagonal principal e diagonal secundária em matrizes quadradas, por exemplo. Basta utilizar os índices que representam as linhas e colunas das matrizes.

Referências

GOODRICH, M. T.; TAMASSIA, R. *Estruturas de dados e algoritmos em Java*. Porto Alegre: Bookman, 1997.

TENENBAUM, A. A.; LANGSAM, Y.; AUGENSTEIN, M. J. *Estruturas de dados usando C*. São Paulo: Makron Books do Brasil, 1995.

ZIVIANI, N. *Projeto de algoritmos com implementação em Java e C++*. São Paulo: Cengage Learning, 2006.

Leituras recomendadas

ESTRUTURA de dados (a famosa ED que todo dev tem que aprender): dicionário do programador. Petrópolis: [*S. n.*], 2020. 1 vídeo (12 min). Publicado pelo canal Código Fonte TV. Disponível em: https://www.youtube.com/watch?v=EfF1M7myAyY. Acesso em: 14 jan. 2021.

GUIA completo de Java. [*S.l.*]: DevMedia, [20--?]. Disponível em https://www.devmedia.com.br/guia/linguagem-java/38169. Acesso em: 14 jan. 2021.

Fique atento

Os *links* para *sites* da *web* fornecidos neste capítulo foram todos testados, e seu funcionamento foi comprovado no momento da publicação do material. No entanto, a rede é extremamente dinâmica; suas páginas estão constantemente mudando de local e conteúdo. Assim, os editores declaram não ter qualquer responsabilidade sobre qualidade, precisão ou integralidade das informações referidas em tais *links*.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Dica do Professor

É muito comum relacionar as estruturas de dados *array* na busca pela solução computacional de um problema da vida real. Esse relacionamento pode ocorrer entre matriz e vetor, vetor e matriz, entre diferentes matrizes e diferentes vetores. Nesses casos, é importante entender e saber aplicar os conceitos de indexação entre as estruturas de dados envolvidas.

Nesta Dica do Professor, você verá como fazer o relacionamento entre vetores e matrizes por meio de índices em Java.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Exercícios

1) **Os conceitos matemáticos de matrizes podem ser facilmente aplicados computacionalmente utilizando estruturas de dados. Um conceito bem comum de matrizes é o de diagonal principal. A diagonal principal ocorre em matrizes quadradas e une o seu canto superior esquerdo ao canto inferior direito.**

Considerando uma matriz bidimensional quadrada 3x3 e o código Java:

```
public static void main(String[] args) {

int i, j;

int m[][] = new int[3][3];

for (i = 0; i < 3; i++)

for (j = 0; j < 3; j++)

m[i][j] = i + j;

System.out.println();

System.out.println("Diagonal principal");

}
```

Indique qual dos itens descreve corretamente o código Java para imprimir os valores da diagonal principal da matriz m:

A)
```
for (i = 0; i < 3; i++) {

System.out.print(m[i][i]);

System.out.print(" ");

}
```

B)
```
for (i = 0; i < 3; i++) {

System.out.print(m[j][j]);
```

```
System.out.print(" ");
}
```

C)

```
for (i = 0; i < 3; i++) {
System.out.println();
for (j = 0; j < 3; j++) {
System.out.print(m[i][j]);
System.out.print(" ");
}
}
```

D)

```
for (i = 0; i < 3; i++) {;
System.out.println();
for (j = 0; j < 3; j++) {
System.out.print(m[3][3]);
System.out.print(" ");
}
}
```

E)

```
for (i = 0; i < 3; i++) {
System.out.print(m[3][3]);
System.out.print(" ");
}
```

2) Um vetor é uma estrutura de dados unidimensional que armazena diversos valores de mesmo tipo de dado. Para manipular o conteúdo dos vetores utilizamos os índices, que são sempre valores inteiros. Uma ação muito comum em programas computacionais é a troca de valores entre duas variáveis. Isto é, se tivermos as variáveis A=5 e B=25, trocando os valores entre estas duas variáveis, teremos A= 25 e B= 5.

Considerando um vetor v inteiro de 20 posições e o código Java:

```
public static void main(String[] args) {

int i, j, aux;

int v[]= new int[20];

for (i=0; i< 20; i++)

v[i] = i + 20;

System.out.println();

System.out.println("Vetor inicial");

for (i=0; i< 20; i++) {

System.out.print(v[i]);

System.out.print(" ");

}

System.out.println();

System.out.println("Vetor com conteúdo trocado");

for (i=0; i< 20; i++) {

System.out.print(v[i]);

System.out.print(" ");

}

}
```

Indique o trecho correto do código em Java para trocar o conteúdo entre duas posições consecutivas (uma seguida da outra) do vetor v, antes de imprimir o vetor com o conteúdo trocado:

A)

```
for (i=0; i < 19; i++) {

aux = v[i];

v[i] = v[i+1];

v[i+1] = aux;
```

```
}
```

B)

```
for (i=0; i < 19; i++) {
aux = v[i];
v[i] = v[j];
v[j] = aux;
}
```

C)

```
for (i=0; i < 19; i++) {
v[i+1] = v[i];
v[i] = v[i+1];
}
```

D)

```
for (i=0; i < 19; i++) {
v[i] = v[i+1];
v[i+1] = v[i];
}
```

E)

```
for (i=0; i < 20; i++) {
aux = v[i];
v[i] = v[i+1];
v[i+1] = aux;
}
```

3) Diversas operações da álgebra linear podem ser realizadas sobre matrizes, como adição (soma), subtração e multiplicação, cálculo de matriz inversa, ordenação e cálculo de determinantes.

Considerando o código em Java a seguir, assinale V (verdadeiro) ou F (falso) em relação aos *arrays* multidimensionais:

```
5 public static void main (String[] args) {
6
```

```
7 int i, j;
8 int m[][] = new int[2][3];
9
10 for (i = 0; i < 2; i ++)
11 for (j = 0; j < 3; j++)
12 m[i][j] = i + j;
13
14 }
```

( ) "m" é um *array* bidimensional de 2 linhas e 3 colunas.

( ) "m" é um *array* bidimensional de 3 linhas e 2 colunas.

( ) Cada posição da matriz "m" é preenchida com o valor relativo ao índice da sua linha.

( ) Todas as posições da matriz "m" recebem valor.

( ) Todas as posições da matriz "m" são apresentadas em tela.

A sequência correta de V e F referente aos itens acima é:

A) V, F, V, F, F.

B) V, F, V, V, F.

C) V, F, F, F, F.

D) F, V, V, F, F.

E) F, V, V, V, F.

4) **Considerando o ambiente de programação Java, um *array* multidimensional é uma estrutura em memória que permite o armazenamento de um conjunto de dados de um mesmo tipo, considerando mais de uma dimensão.**

Assinale V (verdadeiro) ou F (falso) para as afirmativas em relação aos vetores multidimensionais:

( ) Um *array* bidimensional é também chamado de *matriz*.

( ) Uma planilha eletrônica é um tipo de matriz.

( ) Para armazenar e recuperar o conteúdo de uma posição de um *array* bidimensional, deve-se referenciar essa posição pelos índices tanto da linha quanto da coluna.

( ) Na programação Java, é impossível declarar um *array* multidimensional com mais de duas dimensões.

A sequência correta de V e F referente aos itens acima é:

A) V, V, F, V.

B) V, V, F, F.

C) V, V, V, F.

D) V, F, V, F.

E) F, V, V, V.

5) **O setor pedagógico de uma universidade solicitou uma aplicação que verifique se existem alunos cursando, ao mesmo tempo, as disciplinas de Lógica de Programação e Estrutura de Dados. O programa Java que realiza essa verificação utiliza dois vetores; um chamado LOGIC, de 100 posições, para guardar as matrículas dos alunos que cursam Lógica de Programação, e outro chamado ESTRUT, que contém as matrículas dos 140 alunos que estudam Estrutura de Dados. Como resposta, a aplicação imprime a matrícula dos alunos que estão cursando, ao mesmo tempo, as disciplinas de Lógica de Programação e Estrutura de Dados. É importante citar que os vetores que armazenam os dados não estão ordenados.**

Considere o trecho do programa Java:

```
5 public static void main(String[] args) {
6 int i,j;
7 int LOGIC[] = new int[100];
8 int ESTR[] = new int[140];
9
10 for (i=0; i<100; i++)
11 for (j=0; j<140; j++)
12
13
14
```

15 }

Indique qual trecho de código Java deve ser incluído nas linhas 12, 13 e 14 do código acima, para que a aplicação retorne as matrículas dos alunos que estão cursando, ao mesmo tempo, as disciplinas de Lógica de Programação e Estrutura de Dados:

A) 12 if (LOGIC[i] == ESTR[j]) {

13 System.out.println (LOGIC[i]);

14 }

B) 12 if (LOGIC[j] == ESTR[i]) {

13 System.out.println (LOGIC[j]);

14 }

C) 12 if (LOGIC[i] == ESTR[j]) {

13 System.out.println (LOGIC[j]);

14 }

1. fff

D) 12 if (LOGIC[i] == ESTR[i]) {

13 System.out.println (LOGIC[i]);

14 }

E) 12 if (LOGIC[j] == ESTR[j]) {

13 System.out.println (LOGIC[j]);

14 }

Na prática

Desenvolvedores de aplicações computacionais estão a todo momento buscando entender as necessidades dos usuários para poder resolvê-las computacionalmente da melhor forma possível. Saber transportar problemas da vida real para soluções computacionais é um desafio.

Confira, Na Prática, o desafio de um game de corrida de carros, desenvolvido em Java, que precisa aplicar *arrays* para saber quem será o ganhador.

ALGORITMO DE MAIOR/MENOR

Em computação, se está o tempo todo **coletando dados, para transformá-los em informação**.

Essa informação precisa ser interessante para os tomadores de decisão nas organizações.

Um dado bruto ou sem uma organização **não é válido** para os tomadores de decisão.

Nas organizações é comum informar:

- Qual produto mais deu lucro?
- Quem é o cliente que mais está devendo?
- Qual aluno teve a maior nota?
- Qual jogador teve o maior número de pontos no *game*?
- Qual equipe demorou menos para chegar no resultado?
- Qual alimento tem o menor índice glicêmico?

As perguntas, basicamente, buscam saber o **maior ou menor valor contidos em vetores e matrizes** que estão guardando esses dados.

Case de *game*

Runner é um *game* de corrida de carros que apresenta desafios em seu percurso e o piloto vai **acumulando pontos a cada desafio realizado**.

Explicando melhor

- O *game* tem os dados de pontos da corrida realizada pelos pilotos armazenados em **vetores**.
- O vetor ***name*** guarda o nome dos pilotos e o vetor *point* guarda os pontos de cada piloto na corrida.
- Os vetores estão relacionados pelos índices. Isto é, posição 1 do ***name*** é o nome do piloto que tem os pontos da posição 1 do vetor ***point***, e assim por diante.
- O *game* permite **até 10 pilotos** por corrida.
- Ao final do *game*, é preciso informar o **nome do piloto ganhador** da corrida.

Analisando uma proposta de solução:

Para encontrar essa resposta, é preciso saber qual piloto obteve a maior quantidade de pontos durante a corrida.

Primeiro, é preciso encontrar a maior quantidade de pontos ▸ algoritmo de maior no vetor *point*. Nesse momento, é importante guardar a posição no vetor desse maior valor – o índice.

Os vetores *name* e *point* estão relacionados pelo índice. Por meio do índice que foi guardado do maior número de pontos, recuperando do vetor *name* o nome associado a esse índice, resultará no nome do piloto que ganhou a corrida.

```
package jogorunner;

import java.util.Scanner;

class winner {

public static void main(String[] args) {
Scanner ler = new Scanner (System.in);

int i, imaior, maior;
String name[]= new String[10];
int point[]= new int[10];

for (i=0; i<10; i++) {        //leitura dos dados
 System.out.printf("Informe nome do piloto %d: ",i+1);
 name[i] = ler.next();
     System.out.printf("Informe a pontuação: ");
     point[i] = ler.nextInt();
     System.out.println();
    }

imaior = 0;
maior = point[0];        // inicia o maior com o menor valor
possivel
for (i = 0; i < 10; i ++) // percorre o vetor
if (point [i]> maior) {  // busca um valor maior
maior = point[i];        // encontrou valor maior guarda em maior
imaior = i;              // guarda a posição do maior valor - índice i
}
    System.out.print("O piloto ");
System.out.print(name[imaior]);
System.out.printf(" venceu a corrida com %d pontos", maior );

}
}
```

Veja, a seguir, a execução do programa.

```
Informe nome do(a) piloto(a) 1: pedro
Informe a pontuação: 456

Informe nome do(a) piloto(a) 2: joão
Informe a pontuação: 123

Informe nome do(a) piloto(a) 3: ana
Informe a pontuação: 333

Informe nome do(a) piloto(a) 4: paulo
Informe a pontuação: 478

Informe nome do(a) piloto(a) 5: carlos
Informe a pontuação: 876

Informe nome do(a) piloto(a) 6: bruno
Informe a pontuação: 987

Informe nome do(a) piloto(a) 7: danilo
Informe a pontuação: 765

Informe nome do(a) piloto(a) 8: patricia
Informe a pontuação: 987

Informe nome do(a) piloto(a) 9: ana
Informe a pontuação: 999

Informe nome do(a) piloto(a) 10: antonio
Informe a pontuação: 765

O(a) piloto(a) ana venceu a corrida com 999 pontos
```

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Saiba mais

Para ampliar o seu conhecimento a respeito desse assunto, veja abaixo as sugestões do professor:

Central de ajuda do Java - perguntas gerais

Muitas vezes, ocorrem situações na programação nas quais precisamos de uma ajudinha extra. Neste link do Java, você tem disponíveis algumas respostas.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Desenvolvimento do jogo Pac-man: Uma estratégia para o ensino de estruturas de dados homogêneos — Primeira fase

Independente da linguagem de programação utilizada, entender como os arrays podem ser aplicados no desenvolvimento de games é muito interessante.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Como criar bots para responder no Discord

O Discord é uma plataforma de comunicação muito utilizada e tem vários recursos adicionais. Alguns desses recursos são programados em Java e é possível criar bots para ele, chamando o bot Java. Já imaginou você utilizando seu próprio bot no Discord?

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Amazon libera Alexa Skills Kit para o Brasil

Que tal programar em Java novos skills para a Alexa da Amazon? Aqui no link, você pode aplicar seus conhecimentos em Java numa aplicação real. No GitHub, você também pode participar de forma colaborativa em projetos da Alexa.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Conceitos de computação com Java

Veja, nesta obra, como implementar classes em Java, estruturas de decisão e iteração, além de arrays e listas de arrays.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.