Matrizes e Vetores

Na aula de hoje, estudaremos sobre matrizes e vetores, e veremos alguns exemplos práticos. Lembra-se que o conceito de matrizes e vetores está presente na matemática, então será necessário analisarmos algumas formulas e fazer o uso da lógica.

1.0 Vetores

Um vetor é uma estrutura de dados unidimensional, capaz de armazenar uma sequência de elementos. Ao programar, conhecemos os vetores como array, que nada mais é do que uma coleção de elementos do mesmo tipo armazenados em locais de memória.

É importante saber que os vetores devem armazenam dados do mesmo tipo, não possibilitando que uma String seja armazenada juntos com dados tipo float ou int, por exemplo.

Os vetores possuem posições dentro de si, onde podemos localizar os dados armazenados em suas posições relativas. Veja no exemplo abaixo:

tipoDados[] ExemploVetor = {a, b, c, d, e}

tipoDados[] ExemploVetor = {0, 1, 2, 3, 4}

Perceba que o ExemploVetor tem dentro de si variáveis que estão posicionadas em ordem. Logo abaixo temos um exemplo, onde conseguimos observar as posições, onde:

A está na posição 0;

B está na posição 1;

C está na posição 2;

D está na posição 3;

E está na posição 4.

Abaixo está alguns exemplos de como declarar os vetores para diferentes tipos de dados. Analise:

int[] numeros = {1, 2, 3, 4, 5};

String[] nomes = {"Alice", "Bob", "Charlie", "Diana"};

char[] letras = {'A', 'B', 'C', 'D'};

float[] valores = {1.0f, 2.5f, 3.3f, 4.7f};

double[] grandesValores = {1.234567, 2.345678, 3.456789, 4.567890};

boolean[] respostas = {true, false, true, false};

Vamos praticar? Observe o código abaixo:

// Declarando e inicializando um vetor de números inteiros

int numeros[] = {1, 2, 3, 4, 5};

// Acessando elementos do vetor

System.out.println("Imprimindo o primeiro valor do vetor: " + numeros[0]);

System.out.println("Imprimindo o quarto valor do vetor: " + numeros[4]);

System.out.println("Imprimindo o segundo valor do vetor: " + numeros[1]);

//Interando sobre o vetor

for (int i = 0; i < numeros.length; i++) {

System.out.println("Contei " + numeros[i]);

}

No código acima, criamos uma array do tipo int chamado números[]. Nele, adicionamos 5 números, do 1 ao 5, onde o número 1 começa na posição 0 e a sequência dos números terminam na posição 4.

Em seguida, utilizamos System.out.println() para dizer em um texto qual é o valor do Array que será selecionado; foi concatenado a este texto o array com o número da posição onde está os valores que desejamos.

Em seguida, utilizamos a estrutura for para repetir a contagem até que os valores dentro da array acabem. Quando utilizamos o .lenght, estamos definindo que a estrutura se repetirá até que a quantidade de elementos acabe. A mensagem dentro da estrutura for nos mostra a quantidade de vezes que o número será contado.

Você talvez tenha visto algo similar nos seus estudos na matéria de matemática, e esses conceitos não são diferentes aqui. Um vetor é uma representação de uma lista de números, onde os espaços podem ser representados por [x*,* *y*, *z*].

LeonardoCAlves (github.com/Leon4rdoalves) listou os principais métodos da Classes Array:

Arrays.sort(array): Ordena os elementos de um array de primitivos em ordem crescente.

Arrays.sort(array, Collections.reverseOrder()): Ordena os elementos de um array de Objetos em ordem decrescente.

Para utilizar este Método em um vetor de primitivos, precisamos converter o vetor para um vetor de Objetos através das Classes Wrapper (veremos mais adiante).

Arrays.equals(array1, array 2): Compara 2 arrays retornando true se ambos tiverem o mesmo conteúdo, as mesmas referências, na mesma ordem e do mesmo tipo.

Arrays.binarySearch(array, chave de pesquisa): Retorna a posição do elemento procurado no array. O array deverá estar ordenado e sem valores duplicados.

Arrays.fill(array, valor): Preenche o array com o valor informado.

Arrays.compare(array1, array 2): Compara 2 arrays baseados no seu conteúdo, retornando true se ambos tiverem o mesmo conteúdo, as mesmas referências, na mesma ordem e do mesmo tipo.

Vamos fazer um exemplo com um desses métodos? Veja o código abaixo, onde comparamos as listas:

String petA[] = {"Gato", "Cachorro", "Jabuti"};

String petB[] = {"Pato", "Galinha", "Porco"};

int listaA[] = {1, 2, 3};

int listaB[] = {1, 2, 3};

System.out.println("Veja se as listas são iguais:\n");

// Comparando listas diferentes

if(Arrays.equals(petA, petB)) {

System.out.println("A lista A e B são iguais.");

}else {

System.out.println("A lista A e B são diferentes.");

}

// Comparando listas iguais

if(Arrays.equals(listaA, listaB)) {

System.out.println("A lista A e B são iguais.");

}else {

System.out.println("A lista A e B são diferentes.");

}

Criamos dois pares de array, onde duas delas tem valores do tipo String e as outras duas são do tipo Int. Abaixo foram criados dois if/else, onde o método .equals(petA, petB) foi utilizado para comparar as listas de String e de integer.

Se as listas do tipo String e int forem iguais, será exibido a mensagem configurada no System.out.println("A lista A e B são iguais."); se não, a mensagem exibida será a contida no else: System.out.println("A lista A não é igual a lista B.").

Não é muito difícil trabalhar com Vetores, mas é algo que exige um pouco de paciência porque as estruturas do que faremos com essas informações podem ficar grandes.

2.0 Matrizes

Uma matriz é uma estrutura de dados bidimensional que pode armazenar elementos em uma grade de linhas e colunas.

Na programação, podemos definir uma matriz como um array de arrays. Veja no código abaixo:

// Declarando e inicializando uma matriz 2x3 de inteiros

int[][] matriz = {

{1, 2, 3},

{4, 5, 6}

};

// Acessando elementos da matriz

System.out.println("1º valor da linha 1: " + matriz[0][0]);

System.out.println("3º valor da linha 2: " + matriz[1][2]);

for (int i = 0; i < matriz.length; i++) {

for (int j = 0; j < matriz[i].length; j++) {

System.out.println(matriz[i][j]);

}

}

No exemplo acima, foi criado uma matriz com dois arrays, onde o primeiro contém valores de 1, 2 e 3, e o segundo contém valores de 4, 5 e 6.

Em seguida usamos o System.out.println("1º valor da linha 1: " + matriz[0][0]) para selecionar o primeiro valor do primeiro array; e também usamos System.out.println("3º valor da linha 2: " + matriz[1][2]) para selecionar o terceiro valor do segundo array.

Por último trabalhamos com a estrutura for, utilizada para percorrer todos os elementos da matriz:

Primeiro loop: criamos um for externo e iniciamos uma variável i com o valor de 0, para controlar o índice das linhas da matriz, onde o seu cumprimento será do tamanho total da matriz (em nosso caso, 3 posições). A cada loop, 1 será incrementado a variável.

Segundo loop: criamos um loop interno que controla o índice das colunas da matriz para a linha atual i. foi iniciado uma variável j com o valor de 0, e o seu tamanho será menor que o comprimento da linha i. A cada loop, 1 será incrementado a variável.

Dentro dessa estrutura interna temos um System.out.println(matriz[i][j]), que imprime o elemento da matriz em suas respectivas posições, sendo i a linha e J a coluna.

Mas como podemos entender melhor a lógica por trás da estrutura for? Vamos analisar a estrutura executada abaixo:

for (int i = 0; i < matriz.length; i++) {

for (int j = 0; j < matriz[i].length; j++) {

System.out.println("i: " + i + ", j: " + j + " => " + matriz[i][j]);

}

}

Quando executamos o código acima, temos o seguinte resultado para:

Primeira iteração com o loop externo:

i: 0, j: 0 => matriz[0][0] = 1

i: 0, j: 1 => matriz[0][1] = 2

i: 0, j: 2 => matriz[0][2] = 3

O loop inicia com a variável J = 0 e termina com J = 2. Durante o loop, a variável i permanece com o valor de 0 e a variável J assume o valor: 0 no primeiro loop, 1 no segundo loop e 2 no terceiro loop.

Segunda iteração com o loop Externo:

i: 1, j: 0 => matriz[1][0] = 4

i: 1, j: 1 => matriz[1][1] = 5

i: 1, j: 2 => matriz[1][2] = 6

Onde o programa executa as mesmas operações, mas o valor de i muda de 0 para 1 por causa da operação anterior. O loop interno continua fazendo as mesmas operações.

Tudo bem que pode não ser fácil entender, porque existe um jogo de lógica envolvido, e para entender o que foi criado, exige paciência.

Agora falando sobre matemática, uma matriz é uma tabela retangular de números, símbolos ou expressões, organizada em linhas e colunas. Por exemplo, uma matriz 2x3 pode ser representada como:

1 2 3

4 5 6

Para realizar operações básicas com vetores, precisamos somar dois vetores, sendo:

A = [a1, a2, a3]

B = [b1, b2, b3]

A + B = [a1 + b1, a2 + b2, a3 + b3]

A soma de duas matrizes da mesma dimensão é realizada somando os elementos correspondentes. Para realizar a multiplicação de matrizes, precisamos de uma matriz A (com dimensão de *m* x *n*) multiplicada por uma matriz B (com dimensão de *n* x *p*), onde resulta em uma matriz C de *m* x *p*.

Vamos entender na prática como podemos passar esses conceitos para o programa?

int[] vetorA = {1, 2, 3};

int[] vetorB = {4, 5, 6};

int[] vetorC = new int[3];

for (int i = 0; i < vetorA.length; i++) {

vetorC[i] = vetorA[i] + vetorB[i];

}

System.out.println("Soma dos vetores: ");

for (int num : vetorC) {

System.out.println(num);

}

No código acima, criamos uma soma de dois conjuntos de vetores que serão agregados a apenas um vetorC. O vetorC, ao ser criado, não tem nenhum valor porque ele será usado para armazenar a soma do vetorA + vetorB.

Dentro da estrutura for, foi iniciado um índice i com o valor de 0, e ele deverá ter um valor menor que o tamanho do vetorA (que é 3). Enquanto a condição da estrutura for é verdadeira, o loop mapeia as posições do vetorC com a soma dos vetorA + vetorB.

Por exemplo, para a seguinte soma “vetorC[i] = vetorA[i] + vetorB[i]” temos:

Primeiro loop:

vetorC[0] = vetorA[0] + vetorB[0] ou vetorC[0] = vetorA[1] + vetorB[4]

Resultado vetorC[0] = {5}

Segundo loop:

vetorC[1] = vetorA[1] + vetorB[1] ou vetorC[1] = vetorA[2] + vetorB[5]

Resultado vetorC[1] = {5, 7}

Terceiro loop:

vetorC[2] = vetorA[2] + vetorB[2] ou vetorC[2] = vetorA[3] + vetorB[6]

Resultado vetorC[2] = {5, 7, 9}

Após a soma dos vetores, usamos a estrutura for-each para imprimir em tela os números contidos no vetorC. Dentro do parâmetro do for-each foi iniciada uma variável num do tipo int que percorrerá todos os valores do vetorC. O System.out.println(num) imprimirá os valores percorridos no vetorC pela variável num.

Agora veja como realizamos essa operação com as matrizes com o código abaixo:

int[][] matrizA = {

{1, 2, 3},

{4, 5, 6}

};

int[][] matrizB = {

{7, 8, 9},

{10, 11, 12}

};

int[][] matrizC = new int[2][3];

// Soma das matrizes

for (int i = 0; i < matrizA.length; i++) {

for (int j = 0; j < matrizA[i].length; j++) {

matrizC [i][j] = matrizA[i][j] + matrizB [i][j];

}

}

// Imprime a soma das matrizes

System.out.println("Soma das matrizes: ");

for (int i = 0; i < matrizC.length; i++ ) {

for (int j = 0; j < matrizC[i].length; j++) {

System.out.println(matrizC[i][j] + " ");

}

}

No código acima, foram criadas 3 matrizes, onde a matrizA e matrizB recebem 2 vetores com 3 valores cada. A matrizC terá 2 linhas e 3 colunas, e receberá a soma das matrizes A e B.

A lógica no loop for interna é similar a lógica que observamos no exemplo anterior, com a diferença que o índice j também será agregado para mapear a posição dos valores na matrizC e a posição dos valores que serão somados.

Após a soma, foi realizada a impressão dos valores, onde o loop externo com o índice i fará o mapeamento das linhas, e o loop interno com o índice j fará o mapeamento das linhas. O System.out.println() fará a impressão dos valores contidos dentro da matrizC[i][j].

Isso quer dizer que, para cada interação com o matrizC[i][j], um valor é adicionado. No exemplo abaixo, veremos como funciona a iteração e soma do primeiro vetor de uma matriz.

Para o primeiro loop, temos:

matrizC[0][0] = matrizA[0][0] + matrizB[0][0];

Na primeira iteração, será armazenada na primeira posição da matrizC a soma das primeiras posições da matrizA + MatrizB, conforme vemos no exemplo abaixo.

matrizC[0] = matrizA[1] + matrizB[7];

Logo, matrizC[0] = 1 + 7 => {8}

Para o segundo loop, temos:

matrizC[0][1] = matrizA[0][1] + matrizB[0][1];

Nessa segunda iteração, será armazenada na segunda posição da matrizC a soma das segundas posições da matrizA + matrizB. Observe no exemplo abaixo.

matrizC[1] = matrizA[2] + matrizB[8];

Logo, matrizC[1] = 2 + 8 => {8, 10}

Para a terceira iteração, temos:

matrizC[0][2] = matrizA[0][2] + matrizB[0][2];

Na terceira iteração, será armazenada na terceira posição da matrizC a soma das terceiras posições da matrizA + matrizB. Observe no exemplo abaixo:

matrizC[2] = matrizA[3] + matrizB[9];

Logo, matrizC[2] = 3 + 9 => {8, 10, 12}

Você talvez se pergunte como o computador completará a segunda linha da nossa matriz. Depois de executar o loop interno, o nosso índice passará de i = 0 para i = 1.

Sendo assim, um novo array será criado dentro da matriz, e o loop interno novamente será executado, conforme observamos no exemplo anterior. Então a nossa matrizC terá dois arrays, sendo:

matrizC = {

{8, 10, 12},

{14, 16, 18} // Aqui está o resultado da segunda iteraração

}

Conclusão

Referências

Estruturas condicionais e estruturas de repetição em Java

https://www.treinaweb.com.br/blog/estruturas-condicionais-e-estruturas-de-repeticao-em-java#google\_vignette

Matrizes e Vetores

https://github.com/Leon4rdoalves/CookBook-Java/blob/main/08.md