Programação Estruturada

Aula 14 - Recursão: Arrays, Matrizes e Exercícios

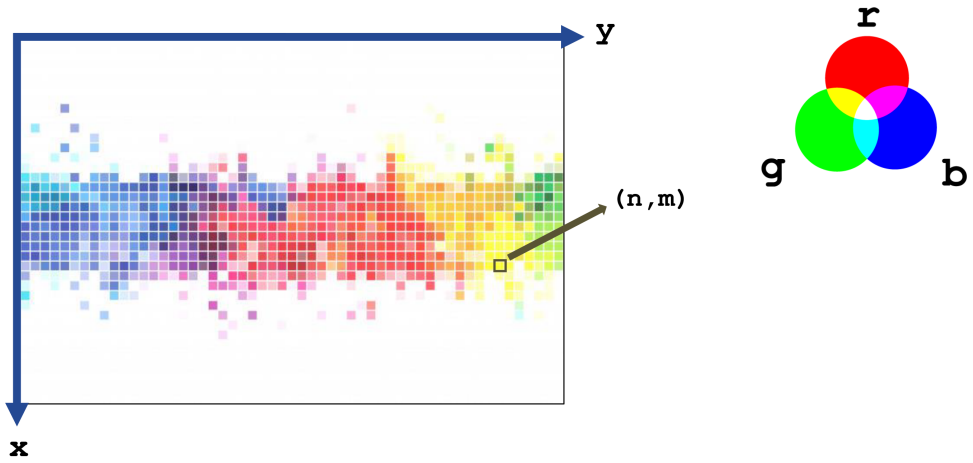
Videoaula 04: Recursão e Matrizes

Videoaula 04: Recursão e Matrizes

Olá! Nesta videoaula, você verá a aplicação de recursão a matrizes. Na prática, tem-se apenas uma extensão do que foi visto na primeira. Afinal de contas, matrizes são arrays, a única diferença é que seus elementos são arrays também. Mais uma vez, utilizarei exemplos para apresentar a aplicação de recursão a matrizes. Para começar, você conhecerá uma utilidade prática de matrizes.

As telas de computadores podem ser tratadas como matrizes. Isto porque cada ponto da tela tem uma coordenada `x` e uma coordenada `y`. Por exemplo, o quadrado amarelo marcado no slide (Figura 1) tem coordenadas `(n,m)`.

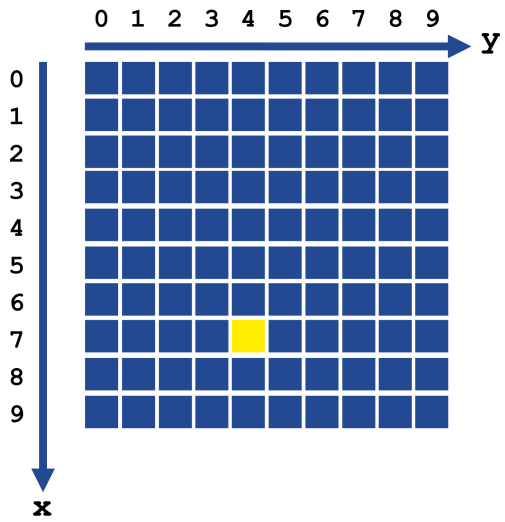
Figura 1 - Telas como Matrizes



É comum que nessa coordenada tenha-se três valores naturais que indicam a intensidade das cores vermelho (***r**ed* em inglês), verde (***g**reen* em inglês) e azul (***b**lue* em inglês). Por isso, esse é um valor RGB: ***R*** de ***r**ed*, ***G*** de ***g**reen* e ***B*** de ***b**lue*.

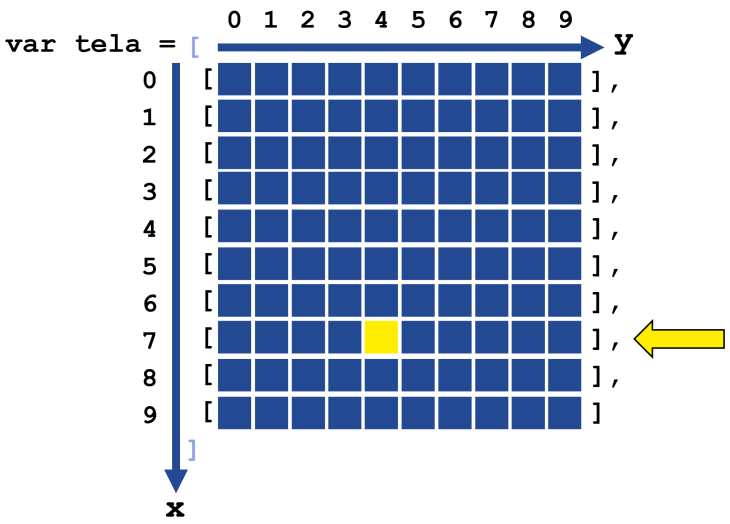
Para facilitar o entendimento, irei simplificar a "tela": considere uma tela 10 por 10, ou seja, que tenha apenas 10 linhas e 10 colunas. No slide (Figura 2), observe no exemplo de tela que apenas o ponto `(7,4)` tem o valor amarelo. Todos os outros são azuis.

Figura 2 - Uma tela 10x10



Você deve estar se perguntando: por que isso pode ser considerado uma matriz? Olhe novamente para o slide (Figura 3). Desta vez, coloquei os colchetes para marcar os arrays. Temos um array mais externo, delimitado com os colchetes azul-claros, cujos elementos são as linhas, os arrays mais internos.

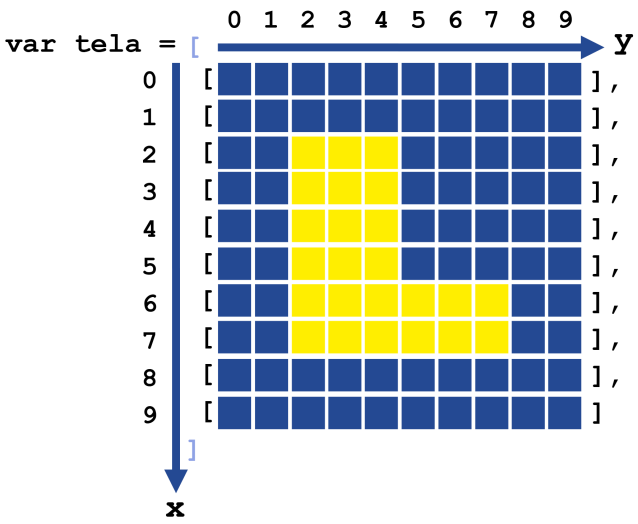
Figura 3 - Uma Matriz 10x10



Dessa forma, se acessarmos o elemento 7 do array `tela`, teremos a linha indicada pela seta no slide. Porém, esse elemento é também um array delimitado com colchetes pretos. Por isso, se você acessar o elemento 4 dessa linha, chegará ao ponto amarelo. Assim sendo, para referenciar o ponto amarelo, pode-se simplesmente usar a expressão `tela[7][4]`. Interessante, não é?

Vou escrever a letra L nesta tela (Figura 4).

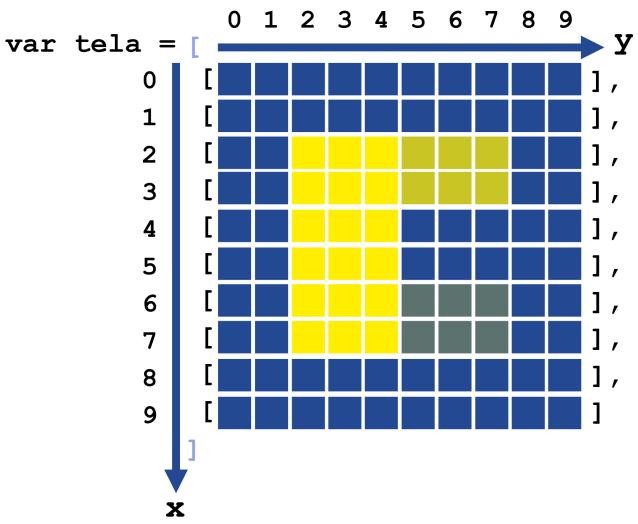
Figura 4 - Escrevendo um L na Tela



Olhe para o slide e tente responder: o que acontece se invertermos as linhas? Ou seja, o que acontece se a linha 0 se tornar a linha 9, a linha 1 se tornar a linha 8, e assim por diante?

Veja agora!

Figura 5 - Invertendo as Linhas

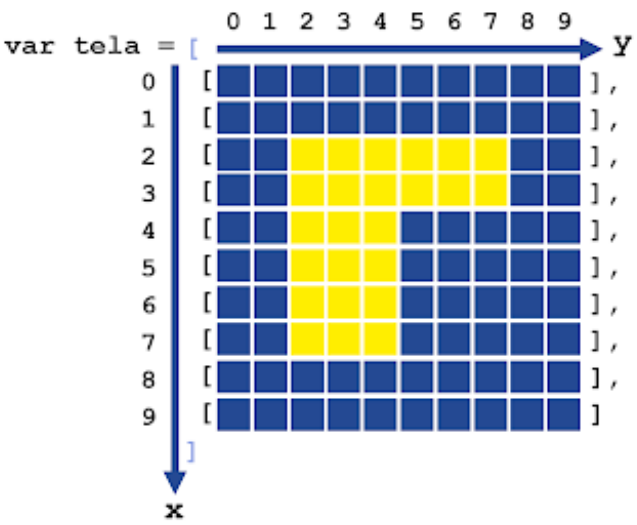


Isso mesmo, ao inverter o array mais externo, inverte-se a imagem verticalmente.

Agora, preste atenção e responda: o que acontece se não alterarmos a ordem das linhas, mas invertermos a ordem dos elementos de cada uma das linhas individualmente?

Obviamente, se invertermos as duas primeiras linhas, não veremos efeito algum. Mas veja só o que acontece se invertermos a terceira linha.

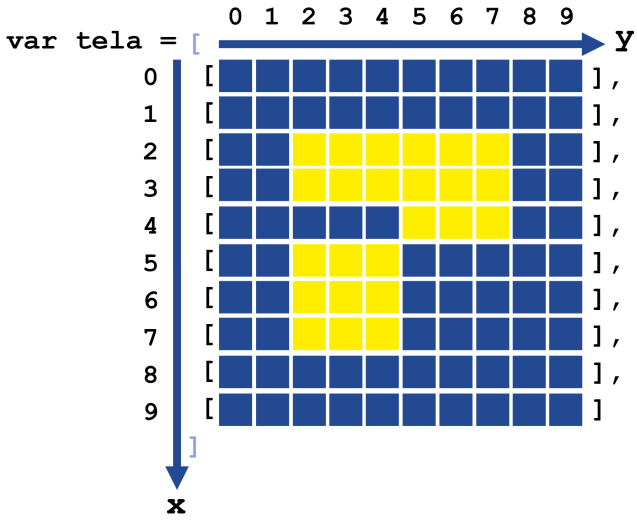
Figura 6 - Invertendo a Ordem das 3 Primeiras Linhas



Nada também! Isto porque a linha é simétrica. Fazendo um paralelo com *strings*, é como se tivéssemos um palíndromo onde a inversão da palavra é idêntica à palavra original. O mesmo acontece com a inversão da linha 3.

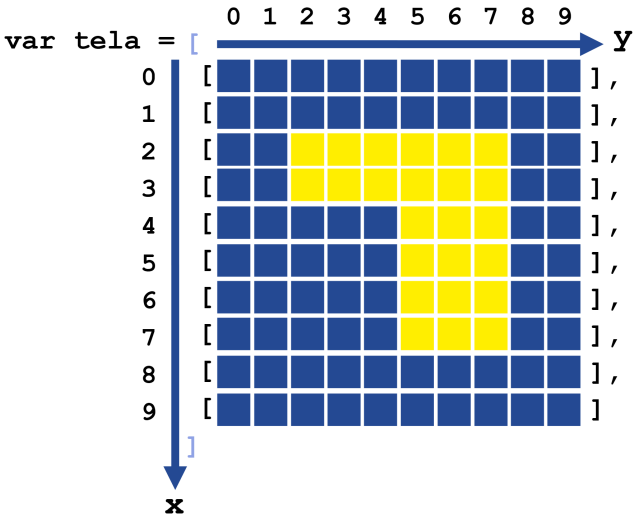
Porém, observe o que acontece se invertermos a ordem dos elementos da linha 4.

Figura 7 - Invertendo a Ordem da Linha 4



Notou a diferença? Vou continuar invertendo as linhas seguintes, ou seja, as linhas 5, 6 e 7.

Figura 8 - Invertendo a Ordem das Linhas 5, 6 e 7



Ao inverter a ordem dos elementos de cada uma das linhas individualmente, inverto a imagem horizontalmente. Legal, não acha?

Você aprendeu a inverter uma imagem verticalmente e horizontalmente, veja como fica o programa que faz essa inversão. Na aula de arrays, você aprendeu a invertê-los usando a função `reverse` e a aplicar uma mesma função a todos os elementos de um array usando a função `map`. Começo apresentando no exemplo uma solução que utiliza essas funções. No entanto, também irei mostrar uma implementação recursiva dessa solução. Vamos lá!

Código 1 - 14_7 Inverter Imagem.html

```html
<html >
   <head>
      <meta charset="UTF-8" />
      <title>Programação Estruturada - Aula 14</title>
   </head>
   <style>
      pre {
         font-family: "Courier New", Courier, monospace;
      }
   </style>
   <body>
      <noscript>Seu navegador não suporta JavaScript ou ele está desabilitado.</noscript>

      <h1>Inversão de Imagem</h1>

      <button onclick="inverter_h_rec()">INVERTER HORIZONTAL</button>
      <button onclick="inverter_v_rec()">INVERTER VERTICAL</button>

      <pre>
         <p id="resultado"></p>
      </pre>

      <script src="script.js"></script>
   </body>
</html>

```

```
var tela =
[
   ["+", "-", "-", "-", "-", "-", "-", "-", "-", "+"],
   ["|", " ", " ", " ", " ", " ", " ", " ", " ", "|"],
   ["|", " ", "+", "+", "+", " ", " ", " ", " ", "|"],
   ["|", " ", "+", "+", "+", " ", " ", " ", " ", "|"],
```

```
    ["|", " ", "+", "+", "+", " ", " ", " ", " ", " ", "|"],
    ["|", " ", "+", "+", "+", " ", " ", " ", " ", " ", "|"],
    ["|", " ", "+", "+", "+", "+", "+", "+", "+", " ", "|"],
    ["|", " ", "+", "+", "+", "+", "+", "+", "+", " ", "|"],
    ["|", " ", " ", " ", " ", " ", " ", " ", " ", " ", "|"],
    ["+", "-", "-", "-", "-", "-", "-", "-", "-", "+"]
];
document.getElementById("resultado").innerHTML = exibirMatriz(tela);

// Inversão usando reverse e map
function inverter_h() {
    tela = tela.map(inverter);
    document.getElementById("resultado").innerHTML = exibirMatriz(tela);
}
function inverter_v() {
    tela = inverter(tela);
    document.getElementById("resultado").innerHTML = exibirMatriz(tela);
}
function inverter(a) {
    return a.reverse();
}

// Inversão iterativa e recursiva (sem usar reverse e map)
function inverter_h_rec() {
    for (var i = 0; i < tela.length; i++) {
        tela[i] = inverter_rec(tela[i]);
    }
    document.getElementById("resultado").innerHTML = exibirMatriz(tela);
}

function inverter_v_rec() {
    tela = inverter_rec(tela);
    document.getElementById("resultado").innerHTML = exibirMatriz(tela);
}

function inverter_rec(a) {
    var retorno = [];
    if (a.length == 0) {
        retorno = [];
    }
    else {
        retorno = inverter_rec(a.slice(1));
        retorno.push(a[0]);
    }
    return retorno;
}

// Já vimos esta solução iterativa de exibição de arrays.
```

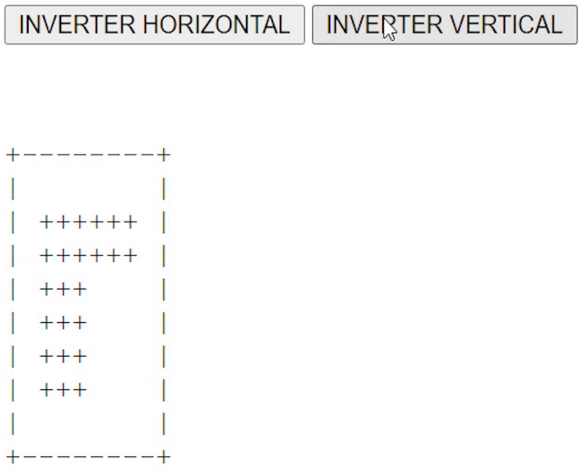
```
// Aqui vamos ver uma solução recursiva e removemos colchetes e vírgulas.

function exibirMatriz(matriz){
    var resposta = "";

    if (matriz.length == 0) {
        resposta = "";
    } else {
        resposta = resposta + exibirLinha(matriz[0]) + "<br>";
        resposta = resposta + exibirMatriz(matriz.slice(1)) + "<br>";
    }
    return resposta;
}

function exibirLinha(linha){
    var resposta = "";

    if (linha.length == 0) {
        resposta = "";
    } else {
        resposta = resposta + linha[0];
        resposta = resposta + exibirLinha(linha.slice(1));
    }

    return resposta;
}

```

Para esse exemplo fiz uma imagem com L, usando *ASCII* mesmo.

No JavaScript, para esse trecho do código (Linhas 2 a 13) criei um array de arrays, onde cada um deles tem a mesma quantidade de caracteres.

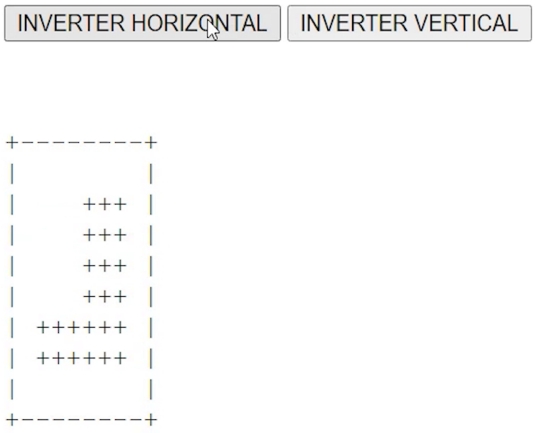
Caso deseje, você pode olhar o código fonte pra ver como foi criada essa imagem.

Figura 9 - Inversão de Imagem



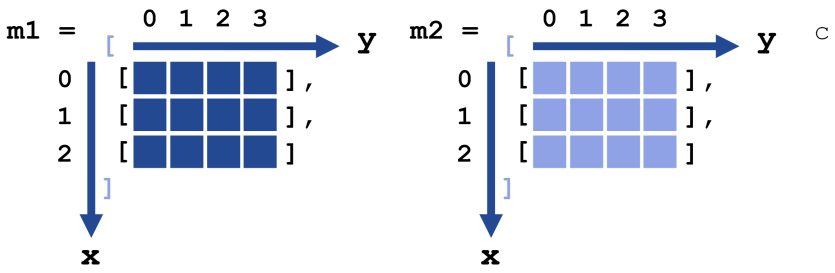
Voltando para a página, na Figura 9, tenho dois botões, um que inverte na vertical, como mostra a Figura 11, ou seja, o `L` sobe e desce.

Figura 10 - Invertendo na Vertical



E outro que inverte na horizontal, que espelha o `L`, como mostra a Figura 12.

Figura 11 - Invertendo na Horizontal



Então, no meu HTML tem o `"resultado"` (linha 20) que eu estou escrevendo aquela imagem do L. Tem dois botões: INVERTER HORIZONTAL e INVERTER VERTICAL. O primeiro chama a função `inverter_h`, o segundo, a função `inverter_v`.

No JavaScript, vamos primeiro ver a solução mais elegante, já que temos uma, e depois apresento a solução recursiva para que você veja como podemos fazer isso de maneira recursiva.

Na linha 17, o `inverter_h`, inverte horizontalmente, aplicando a todos os elementos da matriz na primeira dimensão do array, onde cada elemento é um array, a função `inverter`. Fazendo isso, estou invertendo a imagem de maneira horizontal. Em seguida, peço para exibir a matriz na tela (linha 19). Essa função `inverter` nada mais é do que aplicar um *reverse* ao array (Linha 26).

Na linha 21, em `inverter_v`, inverto a matriz como um todo verticalmente. Ou seja, eu inverto a ordem dos elementos invertendo a tela de maneira vertical (linha 22).

Essa solução é a que acabamos de ver, que funciona perfeitamente na página HTML, no início do exemplo.

Agora, a gente vai passar para a versão recursiva dessa solução, nela, eu criei funções que terminam com `_rec`: `inverter_h_rec`, `inverter_v_rec`, tá certo?

A maneira recursiva funciona do mesmo modo na página HTML, tá certo? E como fazemos isso? Primeiro, vamos ver a inversão horizontal iterativa.

Na linha 30, para inverter horizontalmente, eu uso o `inverter_h_rec`. Na linha 31, para todo elemento da tela, digo que ele é o `inverter_rec` dele (linha 32), tá certo? Ou seja, tenho uma versão iterativa para passar em todos os elementos da tela, e vou aplicar uma inversão recursiva naquele array, tá certo?

O `inverter_v_rec` é a inversão recursiva dos elementos. E como faço a versão recursiva de inverter um array? Bom, para inverter um array, tenho uma variável de `retorno` (linha 43), e, se eu cheguei no caso base, o meu `retorno` é vazio, ou seja, se o `a.length` for igual a 0, meu `retorno` é vazio. Caso contrário, eu faço a inversão do resto (Linha 48) e concateno no final o primeiro elemento (linha 49).

Vejam que, na linha 48, eu faço `a.slice(1)`, ou seja, pego o resto do array e chamo a função `inverter` nele (linha 48). Só depois eu pego o primeiro elemento do array e coloco no final do `retorno` usando o `retorno.push(a[0])` (linha 49).

Já vimos uma versão iterativa para exibir a matriz, mas agora a gente vai ver uma solução recursiva. E para que a nossa imagem na página fique mais bonitinha, ou seja, apareça apenas a imagem, vou tirar os colchetes e as vírgulas entre os elementos, tá certo?

Para exibir a matriz, o que eu faço? Bom, se a matriz chegou ao final, ou seja, no caso base, se o tamanho da matriz for 0, não tenho mais nada para exibir, simplesmente tenho que minha resposta é uma *string* vazia (linha 61). Caso contrário, vou exibir a `matriz[0]`, ou seja, a primeira linha e depois o resto da matriz. E como é o resto da matriz? A `matriz.slice(1)`. Então, é uma função recursiva que vai exibir a primeira linha e vai chamar a mesma função para o resto, em seguida vai exibir a segunda linha até chegar no final, quando ela for vazia, coloca a *string* vazia.

Na linha 69, temos `exibirLinha`, que é bem parecido. Para exibir um array de maneira recursiva, se `linha.length` for igual a 0, tenho vazio, se não, exibo o primeiro elemento dessa linha, `linha[0]` (linha 75), depois chamo a função para exibir a calda

do array, ou seja `linha.slice(1)` (Linha 76), tá certo?

Existem diversos "truques" de programação que fazem com que as imagens se movimentem na tela e, para isso, geralmente, utilizam-se funções matemáticas. Esse é um bom exemplo da importância do conhecimento matemático para a programação.