FUNDAMENTOS EM LÓGICA DE PROGRAMAÇÃO

Atividade: Começando a pensar logicamente e a desenvolver softwares utilizando o Portugol Studio

Tema: Coleção de dados – VETOR (ou ARRAY)

INDICADORES ASSOCIADOS

2 - Analisa e avalia o funcionamento de computadores e periféricos em ambientes computacionais.

3 - Codifica programas computacionais utilizando lógica de programação e respeitando boas práticas de programação.

5 - Desenvolver capacidades linguísticas de modo a saber usar adequadamente a linguagem oral e escrita em diferentes situações e contextos.

8 - Utilizar estruturas de dados definindo-as e aplicando-as adequadamente nos programas.

COLEÇÃO DE DADOS – VETOR OU (ARRAY)

- E se pudermos guardar todos os dados que queremos num único lugar?
- Essa é a função dos vetores e das matrizes;
- Basicamente, o *VETOR* é uma coleção de dados;
- A *MATRIZ*, por sua vez, é uma coleção de vetores;
- Mas, como assim?
- Para entender um, é preciso dominar o conceito do outro;
- Por isso, vamos seguir aos poucos;
- Por isso, nesse documento buscaremos a compreensão apenas do conceito e do funcionamento dos *VETORES*;
- Posteriormente, prosseguiremos com matrizes;
- Então, bora!
- Os *VETORES* também são conhecidos como *ARRAYS*:
    - Então, você pode se comunicar falando sobre VETORES ou sobre ARRAYS;

> Um *array* é uma estrutura de dados homogênea que mantém uma série de elementos de dados de mesmo tipo. Pode-se acessar os elementos individuais armazenados no *array* por meio de uma posição de índice associada, geralmente numérica (Bóson Treinamentos, 2013).

- Ou seja, um *ARRAY* guarda vários dados de um mesmo tipo:
    - Números, textos, dados lógicos.
- Vamos para um exemplo?
- Pense que você está querendo ir pra praia;
- E você está aguardando parado em meio a um congestionamento (seria o caminho para São Francisco do Sul?)
- Então, o que temos?
- Uma única fila até o caminho com vários veículos diferentes:
- Ou seja, numa analogia, temos um *VETOR* de congestionamento, repleto de *DADOS* de veículos;

Figura 1 - Congestionamento na BR-280

Fonte: ND Mais, 2013.

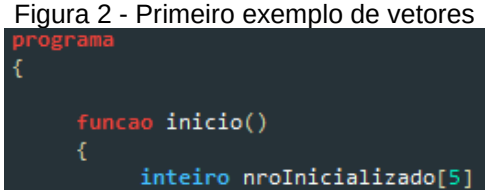
- Então vamos para a prática, no Portugol Studio;

Exemplo 1

- Inicialmente, criaremos uma variável chamada nroInicializado;
- Contudo, não estamos estudando variáveis, e sim vetores;

- Então, essa criação terá uma pequena diferença:
    - A quantidade de posições ou valores que eu quero receber.
- Como indicamos isso?
- Adicionando o número de posições que desejamos entre colchetes:
    - Ou seja, por exemplo, [5] representa que teremos 5 posições no nroInicializado.
- Então, por enquanto, temos o seguinte código-fonte:

Figura 2 - Primeiro exemplo de vetores



Fonte: Autores, 2023.

- Agora, vamos adicionar valores em cada uma das posições desse vetor;
- Temos 5 posições, certo?
- Contudo, na ampla maioria das linguagens de programação temos a seguinte situação:
    - Eu tenho 5 posições;
    - Porém, meu vetor não começa na posição 1, mas sim, na posição 0;
    - Por consequência, sua última posição não é 5, mas 4;
- Então, nesse nosso exemplo, para alimentar o vetor nroInicializado devemos:
    - Começar na posição 0;
    - E ir até a posição 4.
- Pode contar, tem 5 posições nesse intervalo;
- Por enquanto, vamos apontar valores individualmente para cada uma das posições do vetor;
- Vamos guardar os valores: 10, 20, 30, 40 e 50;
- Portanto, nosso código-fonte terá a seguinte estrutura:

Figura 3 - Primeiro exemplo de vetores

```
nroInicializado[0] = 10
nroInicializado[1] = 20
nroInicializado[2] = 30
nroInicializado[3] = 40
nroInicializado[4] = 50
```

Fonte: Autores, 2023.

- Para finalizar essa primeira parte do treinamento, precisamos, como sempre, mostrar os dados para o usuário, que desconhece o interior do código-fonte;
- Portanto, faremos o seguinte trecho:

Figura 4 - Primeiro exemplo de vetores

```
escreva("nroInicializado[0]: " + nroInicializado[0] + "\n")
escreva("nroInicializado[1]: " + nroInicializado[1] + "\n")
escreva("nroInicializado[2]: " + nroInicializado[2] + "\n")
escreva("nroInicializado[3]: " + nroInicializado[3] + "\n")
escreva("nroInicializado[4]: " + nroInicializado[4])
```

Fonte: Autores, 2023.

- Por fim, seu programa estará estruturado da seguinte forma:

Figura 5 - Primeiro exemplo de vetores

```
programa
{

    funcao inicio()
    {
        inteiro nroInicializado[5]

        nroInicializado[0] = 10
        nroInicializado[1] = 20
        nroInicializado[2] = 30
        nroInicializado[3] = 40
        nroInicializado[4] = 50


        escreva("nroInicializado[0]: " + nroInicializado[0] + "\n")
        escreva("nroInicializado[1]: " + nroInicializado[1] + "\n")
        escreva("nroInicializado[2]: " + nroInicializado[2] + "\n")
        escreva("nroInicializado[3]: " + nroInicializado[3] + "\n")
        escreva("nroInicializado[4]: " + nroInicializado[4])
    }
}
```

Fonte: Autores, 2023.

- Como sempre, salve tudo, teste e veja se está tudo OK.

Exemplo 2

- Agora vamos continuar nesse desenvolvimento, mas utilizaremos um outro recurso para resolver nossos problemas;

- Agora, ao invés de indicarmos individualmente os valores de nroInicializado, faremos isso em bloco;
- Vamos utilizar um método simples para resolver;
- Então, primeiramente:

SALVE O ARQUIVO ANTERIOR E FECHE
CRIE UM NOVO ARQUIVO PARA TRABALHARMOS
NÃO SUBSTITUA O QUE ACABAMOS DE FAZER

- Agora vamos começar novamente;
- Faremos denovo a criação de um vetor chamado nroInicializado;
- Porém agora nós indicaremos logo de início os valores que desejamos;
- Abaixo, indicaremos o código-fonte completo;
- Repare que o vetor criado possui {} para englobar os valores que ele receberá;
- Dentro dele, deve existir exatamente a quantidade de valores que serão comportados pelo vetor:
    - No nosso exemplo, não podemos ter mais ou menos do que 5 (cinco) valores:

Figura 6 - Segundo exemplo de vetores

```
programa
{

    funcao inicio()
    {
        inteiro nroInicializado[5] = {60, 70, 80, 90, 100}

        escreva(nroInicializado[0])
        escreva("\n"+nroInicializado[1])
        escreva("\n"+nroInicializado[2])
        escreva("\n"+nroInicializado[3])
        escreva("\n"+nroInicializado[4])

    }
}
```

Fonte: Autores, 2023.

Exemplo 3

- Novamente, vamos continuar, mas sem substituir aquilo que acabamos de criar;
- Então:

SALVE O ARQUIVO ANTERIOR E FECHE

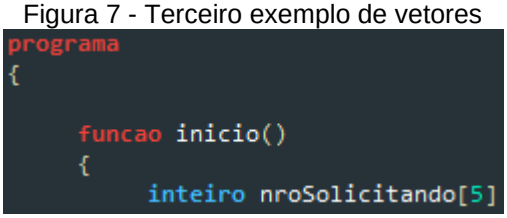
CRIE UM NOVO ARQUIVO PARA TRABALHARMOS

NÃO SUBSTITUA O QUE ACABAMOS DE FAZER

Vamos começar criando um vetor chamado nroSolicitando;

Ele terá o tipo inteiro e 5 posições;

Figura 7 - Terceiro exemplo de vetores



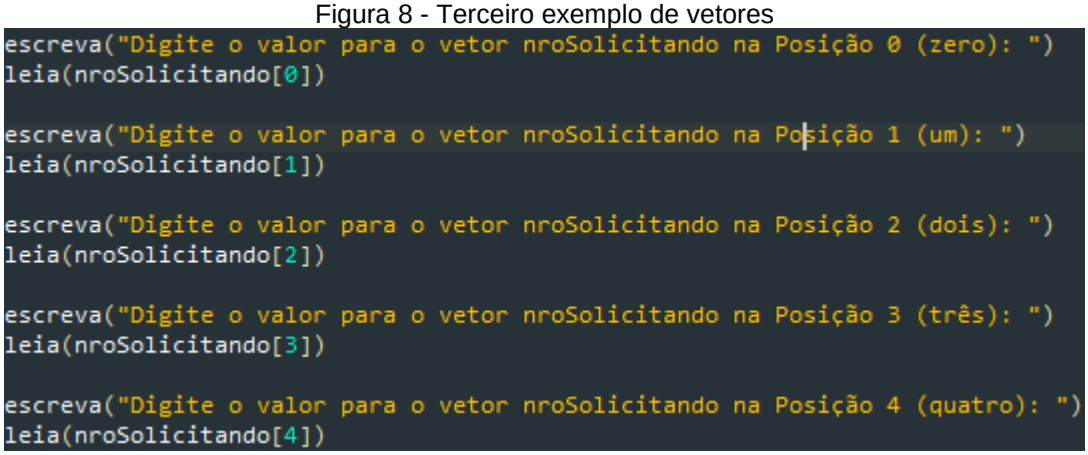
Fonte: Autores, 2023.

- Nossa ideia não é que nós informemos os valores ao vetor, como antes;
- Agora pediremos que o usuário informe os valores;
- Pois bem, sempre que queremos que o usuário informe um valor ao vetor uma coisa é necessária:

É PRECISO QUE INFORMEMOS QUAL A POSIÇÃO QUE SERÁ ALIMENTADA PELO USUÁRIO

- Então, prosseguiremos com o seguinte processo em nosso código-fonte:

Figura 8 - Terceiro exemplo de vetores



Fonte: Autores, 2023.

- Repare que duas coisas são feitas de maneira conjunta:
  - Usamos um escreva, com uma mensagem ao usuário:
    - Afinal, como ele vai saber o que deve informar?
  - Usamos um leia, para guardar o valor que o usuário informou na posição que desejamos.
- Por fim, nosso código-fonte ficará completo da seguinte forma:

Figura 9 - Terceiro exemplo de vetores

```
programa
{

    funcao inicio()
    {
        inteiro nroSolicitando[5]

        escreva("Digite o valor para o vetor nroSolicitando na Posição 0 (zero): ")
        leia(nroSolicitando[0])

        escreva("Digite o valor para o vetor nroSolicitando na Posição 1 (um): ")
        leia(nroSolicitando[1])

        escreva("Digite o valor para o vetor nroSolicitando na Posição 2 (dois): ")
        leia(nroSolicitando[2])

        escreva("Digite o valor para o vetor nroSolicitando na Posição 3 (três): ")
        leia(nroSolicitando[3])

        escreva("Digite o valor para o vetor nroSolicitando na Posição 4 (quatro): ")
        leia(nroSolicitando[4])

        escreva("nroSolicitando[0]: " + nroSolicitando[0] + "\n")
        escreva("nroSolicitando[1]: " + nroSolicitando[1] + "\n")
        escreva("nroSolicitando[2]: " + nroSolicitando[2] + "\n")
        escreva("nroSolicitando[3]: " + nroSolicitando[3] + "\n")
        escreva("nroSolicitando[4]: " + nroSolicitando[4])

    }
}
```

Fonte: Autores, 2023.

- Tudo funcionou?
- Revise o conteúdo, verifique o funcionamento e qualquer coisa chama a gente!
- Tudo dando certo, bola pra frente com os exercícios!

REFERÊNCIAS BIBLIOGRÁFICAS

BÓSON TREINAMENTOS em Ciência e Tecnologia. **Lógica de Programação - Vetores - Definição e Declaração**. São Paulo, SP: 2013. Disponível em: bosontreinamentos.com.br/logica-de-programacao/17-logica-de-programacao-vetores-definicao-e-declaracao/. Acesso em: 06 fev. 2023.

PEREIRA, Mariana. **Congestionamento na BR-280, em direção à São Francisco do Sul, já chega a 40 quilômetros**: Na BR-101, as filas no sentido Sul já chegam a 20 quilômetros. Joinville, SC: ND Mais, 2013. Disponível em: https://ndmais.com.br/transito/congestionamento-na-br-280-ja-chega-a-40-quilometros/. Acesso em: 06 fev. 2023.