

Lógica de Programação

Prof. Ms. Marcos Monteiro



O que veremos neste Treinamento?

- Conceito de Lógica de Programação e Algoritmo
- Estrutura de um pseudocódigo em VisualG
- Variáveis, Palavras Reservadas e Tipos de Dados
- Constantes, Comandos de Atribuição e Comentários
- Operadores Aritméticos, Relacionais, Lógicos e de Caracteres.
- Leitura de Dados e Saída de Dados
- Estruturas Condicionais Se, Senão Se e Senão
- Estruturas Escolha caso (switch-case)
- Estruturas de Repetição - para(for)
- Estruturas de Repetição - enquanto(while)
- Estruturas de Repetição – repita-ate(do-while).
- Estruturas de Dados – Vetor
- Estruturas de Dados - Matriz



Paylivre be.academy



Veremos nesta aula

- Conceito de Lógica e Algoritmo
- Padrão a ser adotado neste treinamento
- Exemplos de problemas como raciocínio Lógico.
- Dicas para o aprendizado.

Conceito de Lógica e Algoritmo

- Lógica, como diria o poeta, “É a arte de pensar corretamente”.
- Quando falamos em lógica, de pronto nos vem a mente, a indagação se algo faz sentido ou pode ser validado ou não.
- A lógica nos induz a adotar um raciocínio que faça sentido e que corrobore com os padrões que usamos normalmente.

Conceito de Lógica e Algoritmo

- Um **Algoritmo** é uma sequência de instruções ordenadas, de forma lógica, para a resolução de uma determinada tarefa ou problema.
- Vejamos aqui que a lógica deve estar presente na construção de nossos algoritmos pois ela dá sentido a melhor sequência para solução de uma tarefa.
- Como nos ensina o Wikipedia (2022):

*“Em [matemática](#) e [ciência da computação](#), um **algoritmo** é uma sequência finita de [ações executáveis](#) que visam obter uma solução para um determinado tipo de problema.^{[1][2]} Segundo Dasgupta, Papadimitriou e Vazirani; “Algoritmos são procedimentos precisos, não ambíguos, padronizados, eficientes e corretos.”.^[3]”*

Exercício Exemplo de raciocínio lógico

- Como vemos no exercício abaixo a lógica nos levará a identificar o raciocínio inicial e ajudar a resolver o exercício em questão.
- Descubra a lógica e complete o próximo elemento (GOUVEIA, 2022):

- a) 1, 3, 5, 7, ____
- b) 2, 4, 8, 16, 32, 64, ____
- c) 0, 1, 4, 9, 16, 25, 36, ____
- d) 4, 16, 36, 64, ____
- e) 1, 1, 2, 3, 5, 8, ____
- f) 2, 10, 12, 16, 17, 18, 19, ____

Solução - Exemplo de raciocínio lógico

- a) 1, 3, 5, 7, ____
- b) 2, 4, 8, 16, 32, 64, ____
- c) 0, 1, 4, 9, 16, 25, 36, ____
- d) 4, 16, 36, 64, ____
- e) 1, 1, 2, 3, 5, 8, ____
- f) 2, 10, 12, 16, 17, 18, 19, ____

- a) **9**. Sequência de números ímpares ou $+ 2$ ($1+2=3$; $3+2=5$; $5+2=7$; $7+2=9$)
- b) **128**. Sequência baseada na multiplicação por 2 ($2 \times 2=4$; $4 \times 2=8$; $8 \times 2=16$... $64 \times 2=128$)
- c) **49**. Sequência baseada na soma em uma outra sequência de números ímpares ($+1$, $+3$, $+5$, $+7$, $+9$, $+11$, $+13$)
- d) **100**. Sequência de quadrados de números pares (2^2 , 4^2 , 6^2 , 8^2 , 10^2).
- e) **13**. Sequência baseada na soma dos dois elementos anteriores: **1** (primeiro elemento), **1** (segundo elemento), $1+1=2$, $1+2=3$, $2+3=5$, $3+5=8$, $5+8=13$.
- f) **200**. Sequência numérica baseada em um elemento não numérico, a letra inicial do número escrito por extenso: **dois**, **dez**, **doze**, **dezesseis**, **dezessete**, **dezoito**, **dezenove**, **duzentos**.

Exemplo de Algoritmo

Exemplo: Trocar uma lâmpada queimada

- Pegue uma escada;
- Posicione-a embaixo da lâmpada;
- Busque uma lâmpada nova;
- Retire a lâmpada velha;
- Coloque a lâmpada nova

Exercício de Algoritmo

Faça um algoritmo para efetuar a troca de um pneu furado.

Pense nos passos que deve seguir para que tenha sucesso na troca de um pneu. Por exemplo você deve primeiro sinalizar a via antes de começar a troca. Depois verificar se tem um estepe, e assim por diante.

Boa sorte!

O poder do pensamento positivo

- Costumo repetir aos alunos que, como sempre dizia o Walter Disney, “*se você pode sonhar você pode realizar o seu sonho*”.
- Ou como nos ensinou Henry Ford “Se você acha que pode fazer alguma coisa ou se você acha que não pode, você sempre tem razão”.
- No desenvolvimento de sistemas entendo ser o mesmo, se podemos conceber uma solução de forma lógica em nosso pensamento podemos implementar a solução idealizada.
- Desta forma a ***Lógica*** nos diz se o nosso raciocínio sobre algo está certo ou não. O ***Algoritmo*** nos indica a forma, passo-a-passo, de atingirmos o objetivo visando resolver uma tarefa ou problema como no exemplo anterior.

Padrão adotado neste treinamento

Durante este treinamento adotaremos o seguinte padrão:

- Linguagem de elaboração dos exercícios propostos – **Portugol**.
- Aplicativo indicado para implementação dos exercícios – **VisualG**.
- O **Portugol** é uma forma de descrevermos as ações de nosso programa na língua portuguesa e desta forma facilitar o entendimento da lógica e conceitos iniciais de programação, uma vez que, sua implementação é muito próxima de nosso raciocínio em nossa língua.

Exemplo de um programa que deseja somar dois números que serão digitados pelo usuário:

```
1. PROGRAMA SomarDoisNumeros;  
2. VAR numero1, numero2 : REAL;  
3. INICIO  
   4.LER (numero1);  
   5.LER (numero2);  
   6.ESCREVER ('Soma dos números =', (numero1+numero2))  
7. FIM.
```

- Na primeira linha definimos o nome do programa (*SomarDoisNumeros*);
- Na segunda linha declaramos as variáveis que vamos utilizar nele (**numero1** e **numero2**) e dizemos que ela aceitará números fracionários (**REAL**);
- Na terceira linha indicamos o **início** da execução do programa;
- Na quarta e quinta linhas vamos **ler** os números que desejamos somar;
- Na sexta linha vamos **exibir** o resultado da soma;
- Na sétima linha **encerramos** o programa.

Evolução do aprendizado.

- Os programas que iremos desenvolver irão evoluir em complexidade mas sempre nos levarão a definir nossos algoritmos de forma a atingir o objetivo que será proposto como foi feito neste simples exemplo anteriormente citado.
- O mais importante, para que você se torne um programador e domine a linguagem que se propuser a aprender, é a sua **persistência**, ou seja, somente seu esforço e dedicação em repetir os exercícios e exemplos, várias vezes ao dia, o levará a dominar este mundo e ter sucesso com este mundo maravilhoso de desenvolvimento de sistemas.



Paylivre be.academy



Resumo da aula anterior

- Conceito de Lógica e Algoritmo
- Padrão a ser adotado neste treinamento
- Dicas para o aprendizado.

Veremos nesta aula

- Estrutura de um pseudocódigo em VisualG.
- Menu Manutenção
 - Listas
 - Personalizar
- Barra de Ferramentas do VisualG
- Variáveis
- Executando o primeiro programa em VisualG

Estrutura de um pseudocódigo em VisualG.

algoritmo "NomeDoPrograma"

// Função : Ao que o programa se destina.

//As “//” indicam um comentário para o programador

// Autor : Nome de quem está desenvolvendo o programa

// Data : Data

// Seção de Declarações das variáveis a serem utilizadas no programa

VAR numero1, numero2 : REAL;

Inicio

// Seção de Comandos

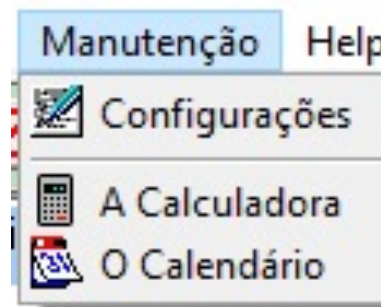
Fimalgoritmo

Destaquei acima a estrutura mínima que deveremos implementar para utilizar o VisualG.

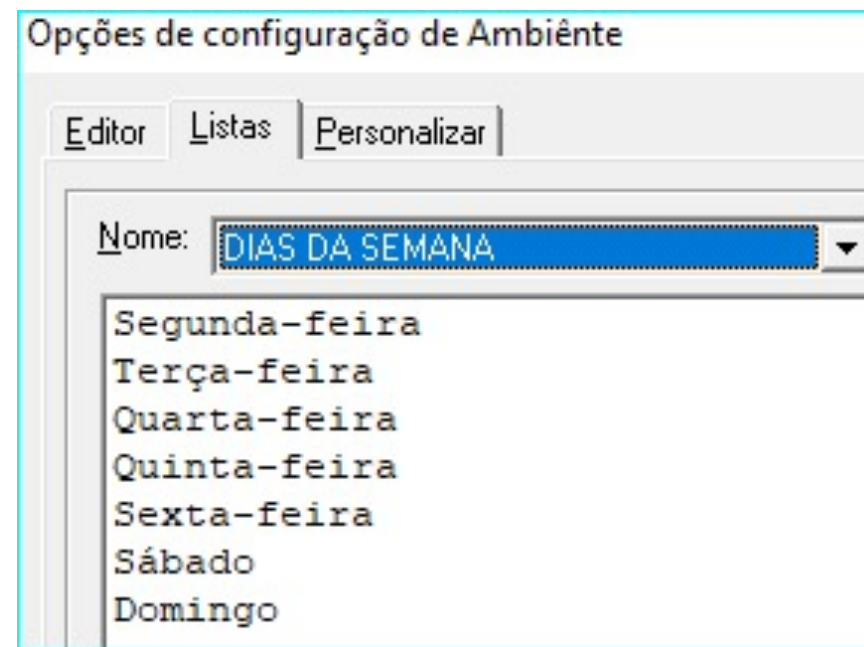
VisualG

Listas pré-existent

Para acessar as listas existentes precisamos clicar no Menu **Manutenção** → **Configurações**.



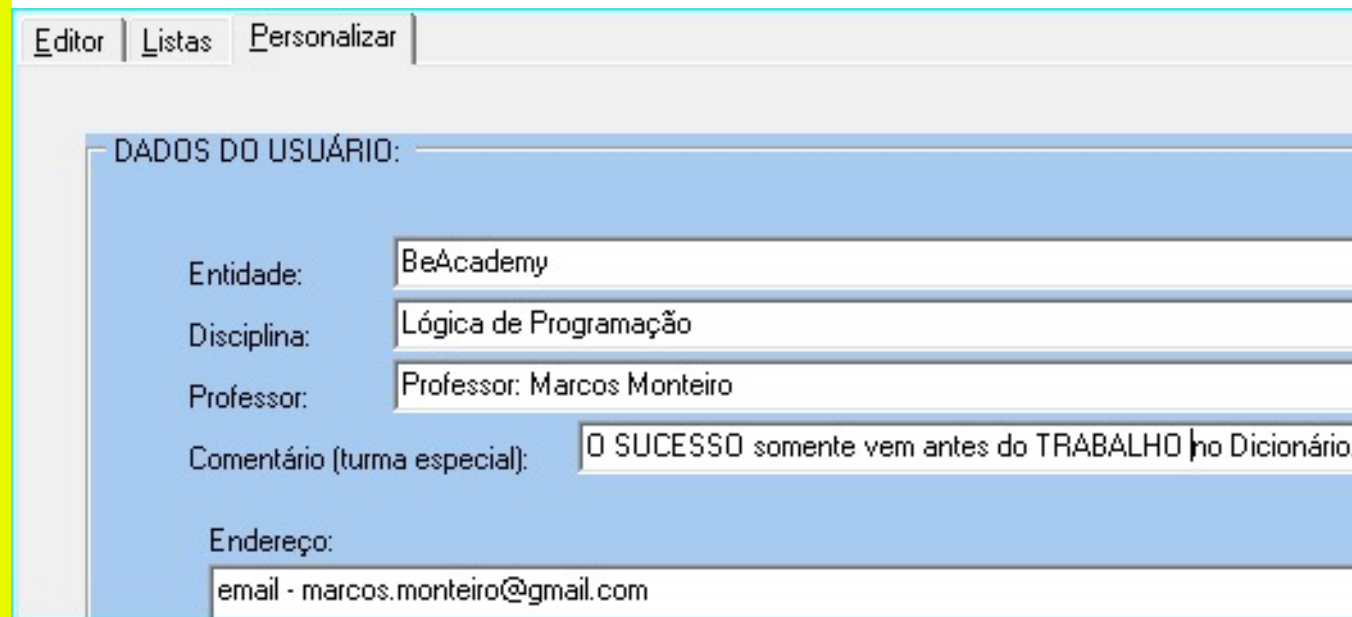
Após isto basta selecionar a guia Listas.



VisualG

Personalizar

A guia Personalizar do Menu de Manutenção nos permite inserir algumas informações .



The screenshot shows the 'Personalizar' (Customize) tab in the VisualG application's menu. The tab is highlighted with a blue border. Below the tab, there is a section titled 'DADOS DO USUÁRIO:' (User Data:). This section contains several input fields for user information:

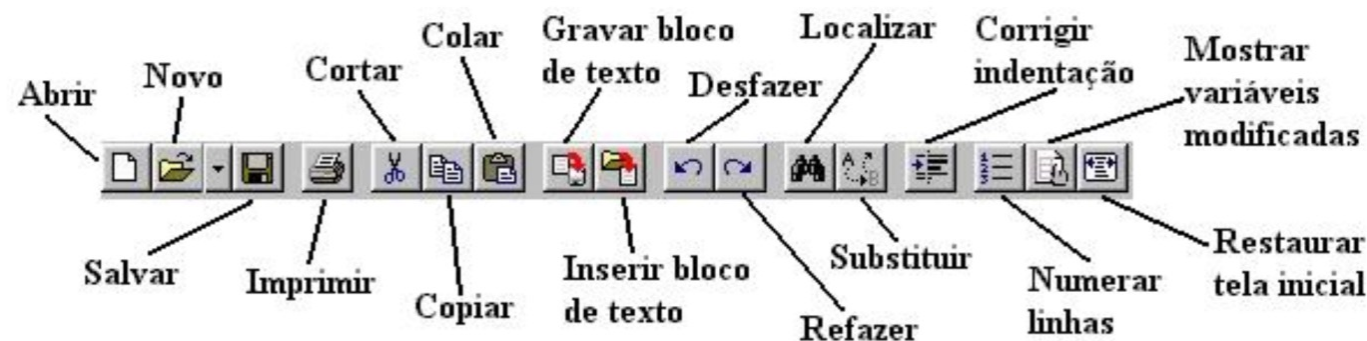
- Entidade:** BeAcademy
- Disciplina:** Lógica de Programação
- Professor:** Professor: Marcos Monteiro
- Comentário (turma especial):** O SUCESSO somente vem antes do TRABALHO no Dicionário.
- Endereço:** email - marcos.monteiro@gmail.com

VisualG

Barra de Ferramentas

A Barra de Ferramentas do VisualG contém os comandos mais comuns utilizados no dia-a-dia. Vou destacar aqui os botões:

- **Corrigir Indentação** que permite automaticamente deixar o código indentado.
- **Numerar linhas** para podermos identificar rapidamente uma linha que possa gerar um erro;
- Mostrar variáveis modificadas que Ativa ou desativa a exibição da variável que está sendo modificada muito interessante quando estamos depurando o programa.
- **Auto-digitação** - Para utilizar esta característica, basta escrever uma abreviatura da palavra-chave ou do comando a ser digitado e teclar *Ctrl-Espaço*. O VisuAlg completa então o comando automaticamente, Algoritmo "semnome"



Variáveis.

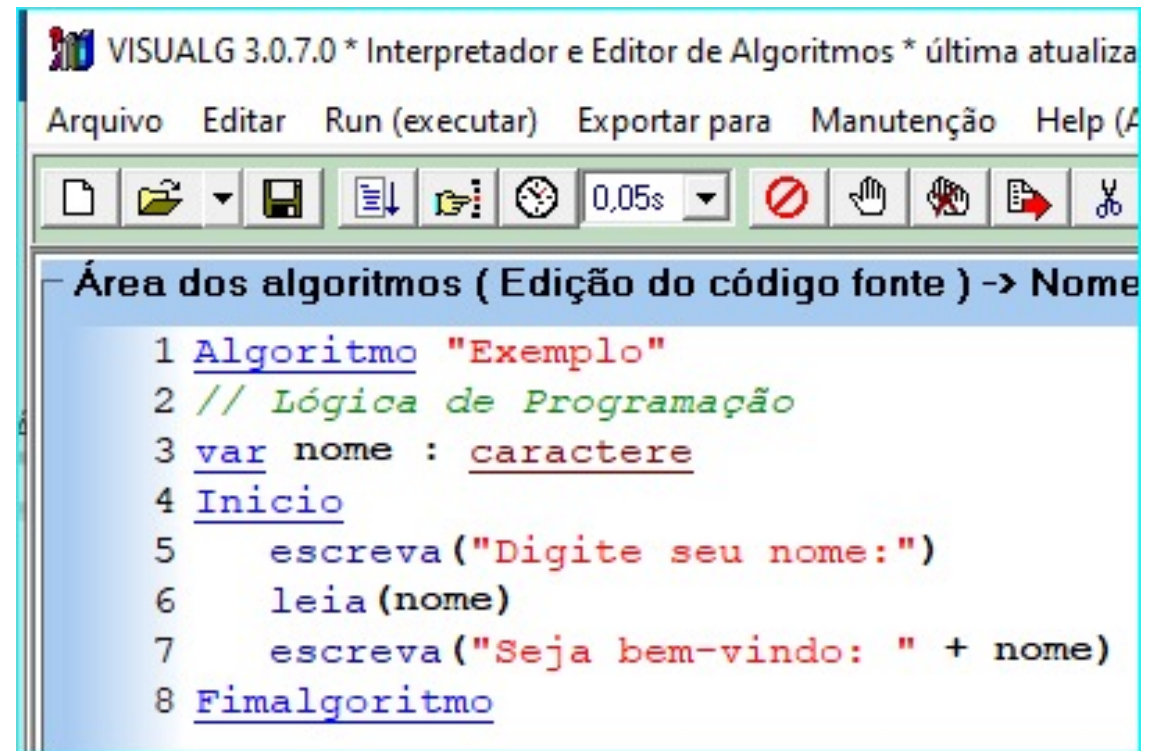
- Variáveis são espaços reservados na memória do computador, devidamente identificados com a finalidade de armazenar um valor.
- Praticamente todo programa precisa de uma variável.
- Por exemplo, se desejamos que o usuário digite seu nome para dar-lhe uma mensagem de boas-vindas precisamos armazenar o nome em uma variável para depois poder recuperá-la e utilizar como desejamos.

Variáveis.

- Os nomes das variáveis devem começar por uma letra e depois conter letras, números ou underline, até um limite de 30 caracteres. Não pode haver duas variáveis com o mesmo nome.
- A seção de declaração de variáveis começa com a palavra-chave **var**, e continua com as
- seguintes sintaxes:
- `<lista-de-variáveis> : <tipo-de-dado>`

Variáveis.

- Vejamos na tela do VisualG.

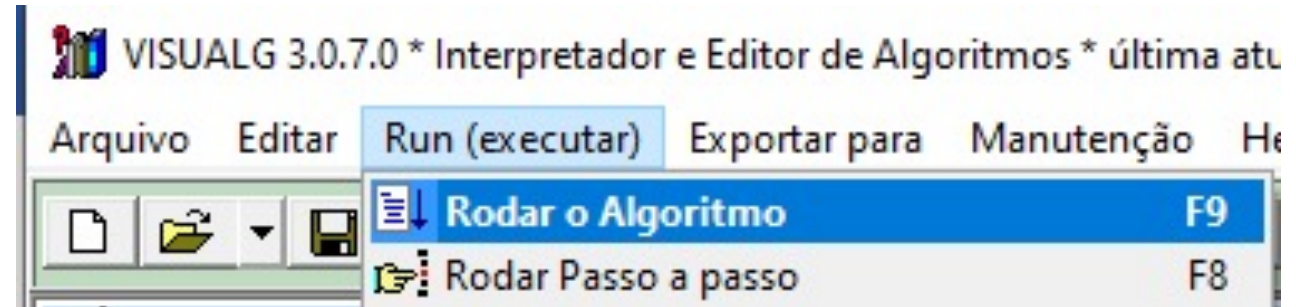


The screenshot shows the VisualG 3.0.7.0 application window. The title bar reads "VISUALG 3.0.7.0 * Interpretador e Editor de Algoritmos * última atualização". The menu bar includes "Arquivo", "Editar", "Run (executar)", "Exportar para", "Manutenção", and "Help (Ajuda)". The toolbar contains icons for file operations (new, open, save, print), execution (run, stop, pause), and other functions. The main area is titled "Área dos algoritmos (Edição do código fonte) -> Nome". It displays the following algorithm code:

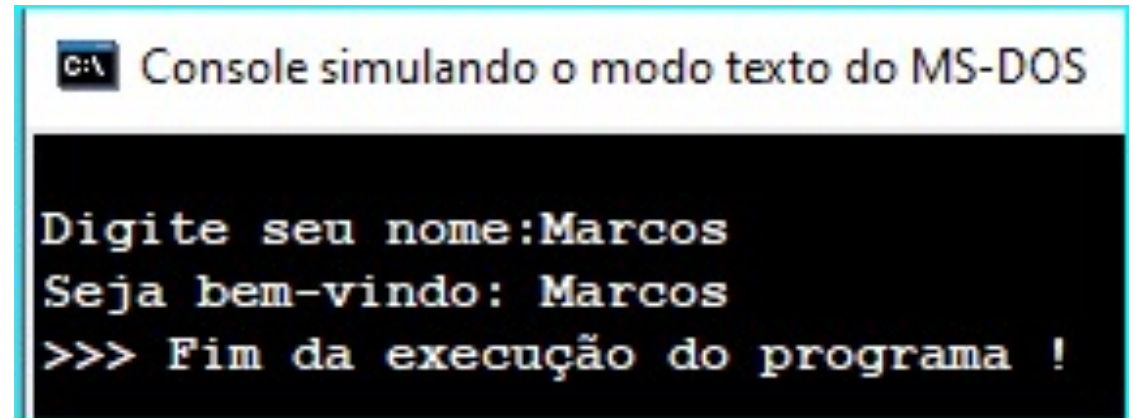
```
1 Algoritmo "Exemplo"  
2 // Lógica de Programação  
3 var nome : caractere  
4 Inicio  
5     escreva("Digite seu nome:")  
6     leia(nome)  
7     escreva("Seja bem-vindo: " + nome)  
8 Fimalgoritmo
```


Variáveis.

- Agora vejamos a tela de entrada e saída do programa após pressionarmos **F9** ou utilizarmos o Menu como visto abaixo:



- A saída do programa será vista a seguir:

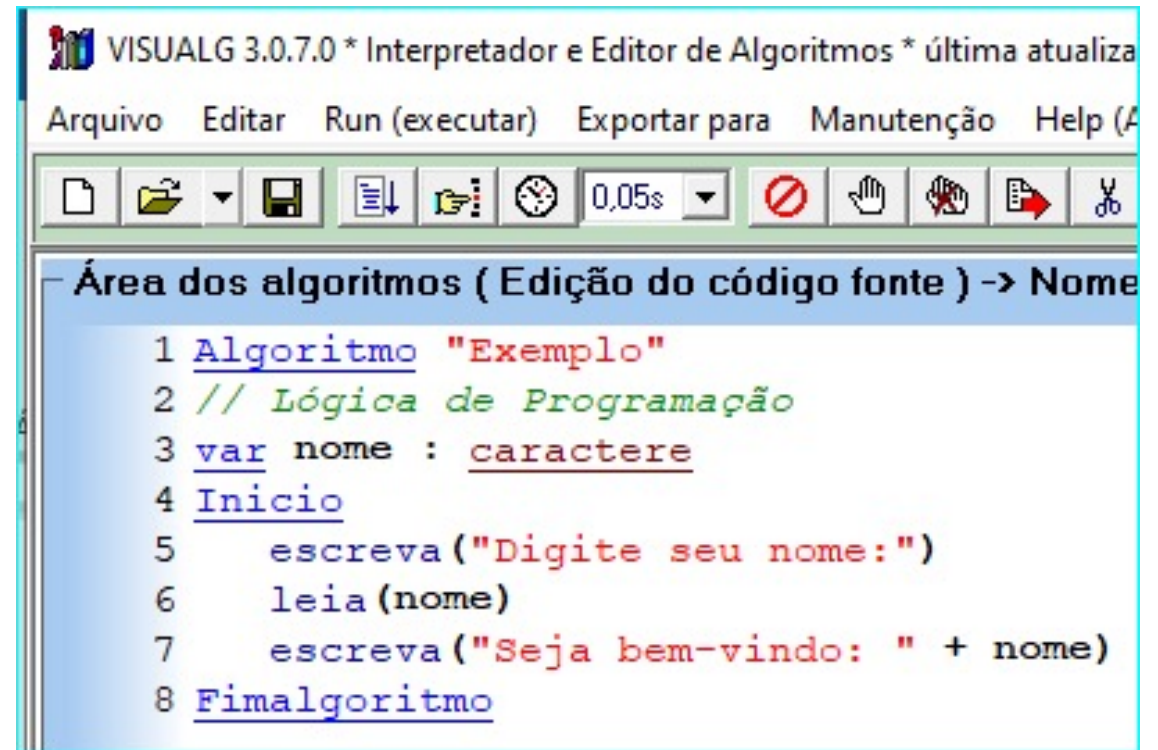
A screenshot of a DOS-style console window titled 'C:\ Console simulando o modo texto do MS-DOS'. The window has a black background with white text. The text displayed is: 'Digite seu nome:Marcos', 'Seja bem-vindo: Marcos', and '>>> Fim da execução do programa !'.

```
C:\ Console simulando o modo texto do MS-DOS

Digite seu nome:Marcos
Seja bem-vindo: Marcos
>>> Fim da execução do programa !
```

Exercício utilizando o VisualG.

- Altere o exemplo visto abaixo para que ao invés do nome solicite o endereço da pessoa e imprima o endereço na tela. Execute o programa e veja se a saída condiz com o endereço que você digitou.



The screenshot shows the VisualG 3.0.7.0 application window. The title bar reads "VISUALG 3.0.7.0 * Interpretador e Editor de Algoritmos * última atualização". The menu bar includes "Arquivo", "Editar", "Run (executar)", "Exportar para", "Manutenção", and "Help (A)". The toolbar contains icons for file operations (new, open, save, print), execution (run, stop, pause), and editing (undo, redo, cut, copy, paste). Below the toolbar, the window title is "Área dos algoritmos (Edição do código fonte) -> Nome". The main text area displays the following algorithm code:

```
1 Algoritmo "Exemplo"  
2 // Lógica de Programação  
3 var nome : caractere  
4 Inicio  
5     escreva("Digite seu nome:")  
6     leia(nome)  
7     escreva("Seja bem-vindo: " + nome)  
8 Fimalgoritmo
```

Veremos na próxima aula

- Palavras Reservadas e Tipos de Dados que podemos utilizar quando estamos criando as nossas variáveis.
- Constantes, Comandos de Atribuição e Comentários



Paylivre be.academy



Resumo da aula anterior

- A tela do VisualG
- Menus do VisualG
- Barra de Tarefas
- Variáveis – Conceitos e utilização em um programa.

Veremos nesta aula

- Palavras Reservadas e Tipos de Dados que podemos utilizar quando estamos criando as nossas variáveis.
- Constantes, Comandos de Atribuição e Comentários
- Exemplos e Exercícios

Palavras Reservadas.

- Palavras reservadas são palavras que a própria Linguagem utiliza para seu funcionamento. Exemplo: VAR, LEIA, ESCREVA, FAÇA, ENQUANTO, SE, SENÃO, CASO, SELECIONE, CONST, ETC.
- As palavras reservadas não podem ser utilizadas por nós para nomearmos uma variável pois, como já dissemos, estas são de uso exclusivo da linguagem.

Tipos de Dados.

- Os tipos de dados indicam a definição de quais valores irão ser armazenados em nossas variáveis.
- Cabe destacar que, na maioria das linguagens, eles são obrigatórios.
- Em nosso exemplo anterior utilizamos o tipo *caractere* quando declaramos a variável(*var*) *nome*.
- Revendo como fizemos:
var nome : caractere
- Desta forma vamos destacar a seguir os principais tipos que utilizaremos neste treinamento.

Tipos de Dados.

Caractere ou String.

- O tipo **caractere**, **literal** ou **String**(na maioria das linguagens) nos permite reservar um espaço de memória para armazenarmos um conjunto de caracteres (letras, números, símbolos, etc).
- Cabe ressaltar aqui que quando pedimos para um usuário digitar um valor que iremos ler este valor sempre será do tipo caractere (String). Vejamos o exemplo novamente:

Algoritmo "Exemplo"

var nome : caractere

Inicio

escreva("Digite seu nome:")

leia(nome)

escreva("Seja bem-vindo: " + nome)

Fimalgoritmo

Tipos de Dados.

Numéricos: Reais ou Inteiros

- Os tipos de dados numéricos serão utilizados quando necessitarmos receber um algum número (inteiro ou fracionário) e fazer algum cálculo com eles. Como por exemplo receber a nota de um aluno.

Algoritmo "NotasAluno"

VAR nota1, nota2: REAL;

Inicio

 escreva("Digite a 1ª nota:")

 leia(nota1)

 escreva("Digite a 2ª nota:")

 leia(nota2)

 escreva("Média obtida: ", ((nota1+nota2)/2))

Fimalgoritmo

Tipos de Dados.

Lógicos

- Os tipos de dados lógicos permitem que possamos receber o resultado de um teste feito. O resultado pode ser Verdadeiro(true) ou Falso(false).
- Desta forma se, por exemplo, perguntarmos a idade de uma pessoa podemos classifica-la como maior de idade, caso tenha 18 anos ou mais, ou como menor de idade, caso tenha menos de 18 anos.
- Este teste poderá ser armazenado em uma variável do tipo lógica(booleana) e quando lermos o seu valor o resultado será Verdadeiro ou Falso.

Constantes

- Constante é uma variável especial que ao receber um valor este não poderá ser alterado durante a execução de nosso programa.
- Um exemplo comum de constante é a variável PI. A variável PI tem o seu valor definido normalmente no momento de sua declaração e ele não poderá ser alterado durante o programa, até porque o valor do PI não varia nunca.
- Cabe destacar que uma constante é definida como a palavra CONST ou final na maioria das linguagens.

Constantes

- Vejamos o exemplo abaixo com o valor de PI que é uma constante interna do VisualG.

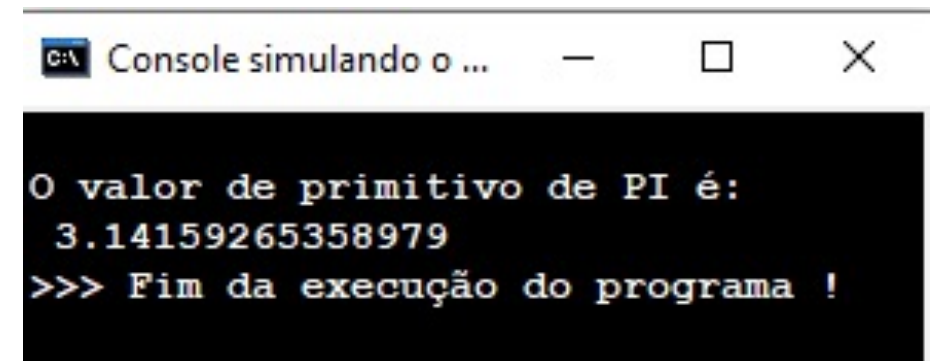
Algoritmo "Contantes"

Inicio

 escreva("O valor de primitivo de PI é:",
 PI)

Fimalgoritmo

- Abaixo vemos o resultado da execução do programa. Observe que sequer definimos a variável PI em nosso programa.



```
C:\> Console simulando o ...  
O valor de primitivo de PI é:  
3.14159265358979  
>>> Fim da execução do programa !
```

Comandos de Atribuição

- A atribuição de valores a uma variável em VisualG é feita utilizando os símbolos <- (menor e hífen). Vejamos no exemplo:

Algoritmo "Tipos de Dados"

VAR

Total, Maior, Menor : INTEIRO;

Nota, Media, altura : REAL;

Resposta : LOGICO;

INICIO

Maior <- 50;

Menor <- 40;

Total <- (Maior-Menor);

Resposta <- Verdadeiro;

escreva(Maior, " - ", Menor, " = ", Total)

escreva("Resposta = ", Resposta)

Fimalgoritmo

Comentários

- Os comentários em VisualG são feitos utilizando // .
- Comentários, em uma linguagem de programação, são orientações e explicações que o programador insere no código para facilitar o entendimento de quem necessitar dar manutenção no programa.
- Os comentários deixam, como já vimos, o texto na cor verde em VisualG. Cabe ressaltar que eles não interferem na execução do programa. Vejamos:

Algoritmo "Tipos de Dados" //Nome do programa

VAR //Definição das variáveis

Constantes, Comandos de Atribuição e Comentários

Exercícios

1. Crie um programa que declare 3 variáveis para receber o Nome, o peso e a altura de uma pessoa. Ao final imprima os dados na tela.
2. Com base no programa anterior calcule e exiba ao final o Índice de Massa Corpórea(IMC) da pessoa sabendo que a fórmula para o cálculo é:

$$\text{IMC} = \text{peso}/(\text{altura}*\text{altura});$$

OBS: Insira um comentário na linha em que se encontra a fórmula para esclarecer como o cálculo é feito. Abaixo a saída esperada

```
Digite o nome: Marcos
Digite a altura: 1.84
Digite o peso: 85
Nome: Marcos Altura: 1.84mts Peso: 85kg IMC: 25.1063327032136
>>> Fim da execução do programa !
```


Constantes, Comandos de Atribuição e Comentários

Exercícios – Solução.

```
Algoritmo "ExercicioIMC" //Nome do programa
VAR //Definição das variáveis
    Nome : CARACTERE;
    peso, altura, imc : REAL;

INICIO
    escreva("Digite o nome:")
    leia(nome)
    escreva(" Digite a altura: ")
    leia(altura)
    escreva(" Digite o peso: ")
    leia(peso)
    imc <- peso/(altura*altura)
    escreva("Nome:", nome, " Altura:", altura, "mts
Peso:", peso, "kg")
    escreva("IMC:", imc)
Fimalgoritmo
```

**Veremos na próxima
aula**

- Operadores Aritméticos, Relacionais, Lógicos e de Caracteres.



Paylivre be.academy



Resumo da aula anterior

- Palavras Reservadas e Tipos de Dados que podemos utilizar quando estamos criando as nossas variáveis.
- Constantes, Comandos de Atribuição e Comentários.

Veremos nesta aula

- Operadores Aritméticos, Relacionais, Lógicos e de Caracteres.
- Leitura de Dados e Saída de Dados.

Operadores Aritméticos, Relacionais, Lógicos e de Caracteres.

Operadores Aritméticos

- Operadores são comumente utilizados em nossos programas .
- São eles **+**, **-**, *****, **/**, **MOD** ou **%** e **^**.
- Vejamos exemplos de sua utilização supondo que a variável **A** inicie com valor 10:

$A - 3;$

$A + 3;$

$A * 5;$

$A / 2;$

$A \% 3;$

Operadores Aritméticos, Relacionais, Lógicos e de Caracteres.

Operadores Relacionais

- Operadores são utilizados em comparações.
- São eles =, >, <, >=, <=, <>.
- Cabe destacar que os operadores relacionais retorna Verdadeiro ou Falso após a comparação feita. Vejamos exemplos de sua utilização supondo que a variável A inicie com valor 10:

A = 3;

A > 3;

A < 5;

A >= 2;

A <= 3;

A <> 2;

Operadores Aritméticos, Relacionais, Lógicos e de Caracteres.

Operadores Lógicos

- Operadores são utilizados normalmente em testes condicionais que veremos adiante.
- São eles **não(!)** , **ou(or ou ||)** , **e(and ou &&)** , **xou** .
- Cabe destacar que o **xou** retorna **verdadeiro** se seus operandos lógicos forem **diferentes**, e **falso** se forem **iguais**.
- Vejamos exemplos de sua utilização supondo que a variável **A** inicie com valor 10 e **B** = 15:

nao(A = 3) ;

(A > 3) ou (A < 5);

(A > 3) e (A < 5);

(A > 3) xou (A < 5);

Operadores Aritméticos, Relacionais, Lógicos e de Caracteres.

Operadores de caracteres.

- Operadores são utilizados em concatenações (uniões) de palavras ou caracteres.
- Representado pelo sinal de positivo +.
- Vejamos um exemplo:
- “São " + " Paulo!!!”
- O resultado será “São Paulo!!!”

Leitura de Dados e saída de Dados.

- Você deve ter observado que, desde o início deste treinamento, estamos realizando a **Leitura** e a **Saída** dos dados.
- Então vamos explicar de forma bem objetiva como ela funciona.
- O método **escreva** escrever uma mensagem na tela que pode ser um “**Olá!**” ou o resultado de alguma operação que fizemos
- Já o método **leia** irá ler o que foi solicitado ao usuário digitar, após um método **escreva**, e normalmente armazenar em uma variável. Vejamos um exemplo:

```
escreva("Digite o nome:")  
leia(nome)
```

Operadores Exemplo

Algoritmo "ExemploOperadores" //Nome do programa

VAR //Definição das variáveis

nome : CARACTERE;

salario : REAL;

INICIO

escreva("Digite o nome:")

leia(nome)

escreval(nome + " Monteiro ")

escreval("Digite o salário: ")

leia(salario)

escreval("Salário: ", salario)

escreval("Salário menor que o mínimo: ", (salario<1200))

escreval("Salário maior que o mínimo: ", (salario>1200))

escreval("Salário diferente do mínimo: ", (salario<>1200))

escreval("Salário igual ao mínimo: ", (salario=1200))

escreval("Resto da divisãod de 5 por 2: ", (5%2))

escreval("Salario é maior que o mínimo e menor que 5000:", ((salario>1200) E (salario<5000)))

Fimalgoritmo

```
Digite o nome:Marcos
Marcos Monteiro
Digite o salário:
1500
Salário: 1500
Salário menor que o mínimo: FALSO
Salário maior que o mínimo: VERDADEIRO
Salário diferente do mínimo: VERDADEIRO
Salário igual ao mínimo: FALSO
Resto da divisãod de 5 por 2: 1
Salario é maior que o mínimo e menor que 5000: VERDADEIRO
```

Operadores Exemplo2

Algoritmo "ExemploOperadores2" //Nome do programa

VAR //Definição das variáveis

nome : CARACTERE;

salario : REAL;

INICIO

escreva("Digite o nome:")

leia(nome)

escreval(nome + " Monteiro ")

escreval("Digite o salário: ")

leia(salario)

escreval("Salário: ", salario)

escreval("Salario é menor que o mínimo OU menor que 5000:", ((salario<1200) OU (salario<5000)))

escreval("Salario NAO é maior que o mínimo e menor que 5000:", (NAO((salario>1200) E (salario<5000))))

escreval("Salario é IGUAL ao mínimo OU diferente 5000:", ((salario=1200) OU (salario<>5000)))

escreval("Salario é IGUAL ao mínimo e é diferente 5000:", ((salario=1200) XO (salario<>5000)))

Fimalgoritmo

```
Digite o nome:
Marcos
Marcos Monteiro
Digite o salário:
1500
Salário: 1500
Salario é menor que o mínimo OU menor que 5000: VERDADEIRO
Salario NAO é maior que o mínimo e menor que 5000: FALSO
Salario é IGUAL ao mínimo OU diferente 5000: VERDADEIRO
Salario é IGUAL ao mínimo e é diferente 5000: VERDADEIRO
```

Operadores Aritméticos, Relacionais, Lógicos e de Caracteres.

Exercícios

1. Crie um programa que receba o nome e a idade de uma pessoa e exiba:
 1. O **nome** e a **idade** informada;
 2. **Verdadeiro** Se a idade é maior que 18;
 3. **Falso** se a idade é diferente que 25;
 4. **Falso** se a idade é diferente que 25 **E** o nome é igual a Marcos;
 5. **Verdadeiro** se a idade é diferente que 25 **OU** o nome é igual a Marcos;
 6. **Verdadeiro** se a idade dividida por 2 é igual a ZERO;

Operadores Aritméticos, Relacionais, Lógicos e de Caracteres.

Exercícios

Crie um programa que receba um valor de depósito do usuário e atualize o seu saldo. Ao final exiba o valor inicial o depósito e o saldo atual.

Veremos na próxima aula

- Estruturas Condicionais Se, Senão Se e Senão.



Paylivre be.academy



Resumo da aula anterior

- Operadores Aritméticos, Relacionais, Lógicos e de Caracteres.
- Leitura de Dados e Saída de Dados.

Veremos nesta aula

- Estruturas Condicionais Se, Senão Se e Senão.

Estruturas

Condicionais **Se, Senão Se e Senão.**

• Em uma linguagem de programação temos as **estruturas condicionais** que nos permitem **testar** algo em nosso programa. Por exemplo recebemos a idade de uma pessoa e queremos saber se a idade é maior que 18 para classificá-la como adulto. Neste caso podemos utilizar a estrutura condicional **se** que irá realizar tal teste. Vejamos sua sintaxe:

```
se <teste> entao  
    <comandos caso o teste seja verdadeiro>  
fimse
```

Estruturas

Condicionais **Se**, **Senão Se** e **Senão**.

- Ao encontrar este comando, o VisuAlg irá analisa a expressão de teste **<teste>** e, se o seu resultado for **verdadeiro**, os comandos da **<comandos se teste for verdadeiro>** são executados.
- Caso o resultado do teste seja **falso**, estes comandos são desprezados e a execução do algoritmo continua a partir da primeira linha depois do **fimse**.

Estruturas Condicionais **Se,** **Senão Se e Senão.**

Exemplo

- Vejamos agora a utilização da estrutura condicional **se** em VisualG.

```
Algoritmo "ExemploSE" //Nome do programa
VAR //Definição das variáveis
    nome : CARACTERE;
    salario : REAL;
INICIO
    escreva("Digite o nome: ")
    leia(nome)
    escreva("Digite o salário: ")
    leia(salario)
    se (salario>1200) entao
        escreval(nome, " seu Salário é maior que o salário
mínimo!!!")
    fimse
Fimalgoritmo
```

```
Digite o nome: Marcos
Digite o salário: 1500
Marcos seu Salário é maior que o salário mínimo!!!
```

Estruturas Condicionais **Se,** **Senão Se e Senão.**

- Uma outra forma de usarmos o condicional se é feita através da palavra **senao** que irá ser executada quando o teste for falso. Vejamos sua sintaxe:

```
se <expressão-lógica> entao  
    < comandos para teste verdadeiro >  
senao  
    < comandos para teste falso >  
fimse
```

- Nesta outra forma do comando, se o resultado da avaliação de <expressão-lógica> for **falso** serão executados os comandos da <**comandos para teste falso**> e a execução continua depois a partir da primeira linha após o fimse.

Estruturas Condicionais **Se,** **Senão Se e Senão.**

Exemplo

- Vejamos agora a utilização da estrutura condicional **se e senão** em VisualG.

```
Algoritmo "ExemploSE" //Nome do programa
VAR //Definição das variáveis
    nome : CARACTERE;
    salario : REAL;
INICIO
    escreva("Digite o nome: ")
    leia(nome)
    escreva("Digite o salário: ")
    leia(salario)
    se (salario>1200) entao
        escreval(nome, " seu Salário é maior que o salário mínimo!!!")
    senao
        escreval(nome, " seu Salário é menor que o salário mínimo!!!")
    fimse
Fimalgoritmo
```

```
Digite o nome: Marcos
Digite o salário: 1000
Marcos seu Salário é menor que o salário mínimo!!!
```

Estruturas

Condicionais **Se,** **Senão Se e Senão.**

- A estrutura **senão se** deverá ser utilizada quando temos mais de um teste a ser feito. Neste caso deverá ser implementado tantos quantos forem os testes.
- Vejamos a seguir a utilização da estrutura condicional **se** e **senão se** e **senao** em VisualG.

Estruturas Condicionais **Se,** **Senão Se e Senão.**

Exemplo

```
Algoritmo "ExemploSE" //Nome do programa
VAR //Definição das variáveis
    nome : CARACTERE;
    salario : REAL;
INICIO
    escreva("Digite o nome: ")
    leia(nome)
    escreva("Digite o salário: ")
    leia(salario)
    se (salario>1200) entao
        escreval(nome, " seu Salário é maior que o salário
mínimo!!!")
    senao
        se (salario=1200) entao
            escreval(nome, " seu Salário é igual ao salário
mínimo!!!")
        senao
            escreval(nome, " seu Salário é menor que o salário
mínimo!!!")
        fimse
    fimse fimse
Fimalgoritmo
```

Estruturas Condicionais **Se,** **Senão Se e Senão.**

- Cabe destacar que o novo teste foi colocado dentro do senão e ao final deste inserido um fimse para encerrar a nova estrutura se que foi criada. Quando uma estrutura é colocada dentro de outra nós dizemos que ela está aninhada com a estrutura anterior. Vejamos agora as saídas que geramos: Com salário igual ao mínimo:

```
Digite o nome: Marcos  
Digite o salário: 1200  
Marcos seu Salário é igual ao salário mínimo!!!
```

- Com salário menor que o mínimo:

```
Digite o nome: Marcos  
Digite o salário: 900  
Marcos seu Salário é menor que o salário mínimo!!!
```

Estruturas Condicionais **Se, Senão Se e Senão.**

Exercícios

1. Crie um programa que receba a altura e o peso da pessoa e a classifique de acordo com a tabela:

Valor do IMC	Classificação
$IMC < 19$	Abaixo do Peso
$19 \leq IMC < 25$	Peso Normal
$25 \leq IMC < 30$	Sobrepeso
$30 \leq IMC < 40$	Obesidade Tipo I
$IMC \geq 40$	Obesidade Mórbida

2. Crie um programa que receba a idade da pessoa e a classifique de acordo com a tabela:

Menos de 18 anos	Menor de Idade
Maior que 18 e menor que 60	Adulto
Mais que 60	Idoso

Estruturas

Condicionais **Se,** **Senão Se e Senão.**

Exercícios

Crie um programa que permita ao usuário escolher a operação a realizar (depósito ou saque), receba a informação da operação escolhida e o valor do usuário e, em seguida, atualize o seu saldo. Ao final exiba o valor inicial, a operação realizada e o saldo atual.

**Veremos na próxima
aula**

- Estruturas Escolha caso (switch-case).



Paylivre be.academy



Resumo da aula anterior

- Estruturas Condicionais Se, Senão Se e Senão.

Veremos nesta aula

- Estruturas Escolha caso (switch-case).

Estruturas Condicionais **escolha** - **caso**.

- O comando condicional ou de **Seleção Múltipla** conhecido como estrutura **escolha caso** também é implementado no VisuAlg.
- Sua sintaxe é a seguinte:

escolha <variável ou valor>

caso <valor1>, <valor2>, ..., <valorn>

<comandos>

caso <valor21>, <valor22>, ..., <valor2n>

<comandos>

...

outrocaso

<comandos caso valor não encontrado>

fimescolha

Estruturas Condicionais **escolha** - **caso**.

Exemplo

- Vejamos a seguir um exemplo de como utilizar a estrutura escolha caso:

```
algoritmo "CalculadoraBasica"
var
    numero1, numero2 : REAL
    operacao : CARACTERE
    resultado : REAL
inicio
    ESCREVA ("Digite o primeiro número: ")
    LEIA (numero1)
    ESCREVA ("Digite a operação: ")
    LEIA (operacao)
    ESCREVA ("Digite o segundo número: ")
    LEIA (numero2)
    ESCOLHA operacao
        CASO "+"
            resultado := numero1 + numero2
        CASO "-"
            resultado := numero1 - numero2
        OUTROCASO
            resultado := -1
    FIMESCOLHA
    ESCREVA ("Resultado: ", resultado)
finalgoritmo
```

```
Digite o primeiro número: 5
Digite a operação: +
Digite o segundo número: 5
Resultado: 10
```

Estruturas Condicionais **escolha** - **caso**.

- Observamos no exemplo anterior que o usuário irá digitar um número, em seguida escolher a operação soma(+) ou subtração(-) e depois digitar o segundo número.
- Após isto o operador digitado (+ ou -) será passado como parâmetro(dentro dos parênteses) para a estrutura escolha(+).
- A partir daí o operador irá ser procurado dentro da estrutura e então, caso ele seja encontrado será realizado o cálculo senão(outro caso) o valor do resultado será negativo(-1). Vejamos a saída com o operador +:

```
Digite o primeiro número: 8
Digite a operação: -
Digite o segundo número: 3
Resultado: 5
```

Estruturas Condicionais **escolha** - **caso**.

- Agora vamos avaliar o resultado quando o usuário digita algum valor diferente de + ou -.

```
Digite o primeiro número: 9  
Digite a operação: *  
Digite o segundo número: 3  
Resultado: -1
```

- Observamos que o usuário digitou dois números porém não escolheu o caractere de soma ou subtração, desta forma não existe um caso para * e a instrução **outrocaso** será a executada lançando o valor -1 para a variável resultado.
- Concluimos que a estrutura **caso – escolha** funciona de forma semelhante a estrutura **se – senao se - senao**.

Estruturas Condicionais **escolha** - **caso**.

- A pergunta que fica então é quando devo usar uma e quando devo utilizar outra.
- Sempre questiono meus alunos sobre isto e raramente eles respondem da forma que eu espero.
- Vamos imaginar uma longa estrutura de **se senao se senao** e entender como ela funciona e depois compará-la com uma estrutura **escolha caso** então tiramos nossas conclusões.
- Analise o exemplo das **idades** e imagine que teremos que testar todas as idades de 1 a 128 para classificar uma pessoa como bebê(0 a 2), criança(2 a 11), adolescente(11 a 18), adulto(18 a 60) e idoso 60 a 128.

Estruturas Condicionais **escolha** - **caso**.

- Qual será a estrutura mais eficiente?
- A estrutura escolha – caso é a resposta correta.
- A explicação é simples a estrutura **escolha caso** conhece todos os casos que ela possui internamente e desta forma não precisa ficar testando um por um pra ver se corresponde ao digitado pelo usuário, ela vai direto do **caso** que o usuário digitou ou **outrocaso** caso o não encontre o valor digitado dentre os casos existentes.
- Daí vai minha dica a todos sempre que possível utilize a estrutura **escolha-caso** ao invés de uma longa sequencia de **se senao se senao**, pois a primeira irá ser executada mais rapidamente.

Estruturas Condicionais **escolha** - **caso**.

Exercícios

1. Crie um programa que solicite ao usuário a operação desejada e implemente as quatro operações matemáticas (soma, subtração, multiplicação e divisão)
2. Crie um programa que receba do usuário a figura geométrica que deseja calcular a área e o perímetro (Q-Quadrado ou T-Triângulo) e calcule e exiba a área e o perímetro da figura escolhida.

Estruturas Condicionais **escolha** - **caso**.

Exercícios

Crie um programa que, utilizando a estrutura **escolha caso**, permita ao usuário escolher a operação a realizar (depósito ou saque ou transferência) , caso a operação seja de transferência solicite o nome do banco, da agência e conta, receba as informações e, em ao final exiba o valor inicial, a operação realizada e o saldo atual, no caso de transferência exiba também os dados do banco, agência e conta.

**Veremos na próxima
aula**

- Estruturas de Repetição - para(for)



Paylivre be.academy



Resumo da aula anterior

- Estruturas Escolha caso (switch-case).

Veremos nesta aula

- Estruturas de Repetição - para(for)

Estruturas de repetição **para**.

- Normalmente em nossa vida repetimos várias ações rotineiramente. Isto ocorre em um simples caminhar(repetimos os passos), respirar, falar, etc.
- Como os programas expressam o que fazemos rotineiramente as linguagens possuem estruturas que implementam estas repetições.
- A primeira que estudaremos é a estrutura **para(for)**.
- A estrutura para é conhecida como uma estrutura que repete uma sequência de comandos por um determinado número de vezes que já é sabido anteriormente.

Estruturas de repetição **para**.

- Vejamos a sintaxe desta estrutura:

```
para <variável> de <valor-inicial> ate <valor-limite>  
[passo<incremento ou decremento>] faca  
  <sequência-de-comandos>  
Fimpara
```

- **<valor-inicial>** - é o valor que desejamos que a variável assuma inicialmente.
- **<valor-limite>** - é o valor máximo que desejamos que a variável assuma.
- **<incremento ou decremento>** - é o valor que desejamos que seja acrescentado ou diminuído do valor inicial da variável que controla nossa estrutura de repetição.
- Cabe ressaltar que os comando serão executados até que o valor limite seja atingido.

Estruturas de repetição **para**.

- Abaixo um melhor detalhamento da estrutura para(for):

<variável >	É a variável contadora que controla o número de repetições do laço. Deve ser necessariamente uma variável do tipo inteiro.
<valor-inicial>	É uma expressão que especifica o valor de inicialização da variável contadora antes da primeira repetição do laço.
<valor-limite >	É uma expressão que especifica o valor máximo que a variável contadora pode alcançar.
<incremento >	É opcional. Quando presente, precedida pela palavra passo, é uma expressão que especifica o incremento que será acrescentado à variável contadora em cada repetição do <incremento>. Quando esta opção não é utilizada, o valor padrão de <incremento> é 1.
fimpara	Indica o fim da seqüência de comandos a serem repetidos.

Estruturas de repetição **para**.

Exemplo

- No exemplo abaixo os números de 1 a 10 são exibidos em ordem crescente. Utilizando a estrutura para e incrementando o valor inicial da variável j”:

```
algoritmo "ImprimeFor1a10"
```

```
var
```

```
  j: inteiro
```

```
inicio
```

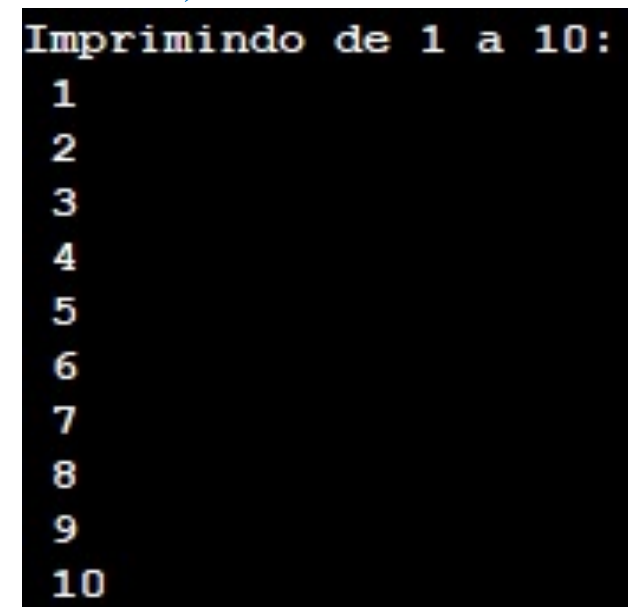
```
  escreval ("Imprimindo de 1 a 10:")
```

```
  para j de 1 ate 10 faca
```

```
    escreval(j)
```

```
  fimpara
```

```
finalgoritmo
```



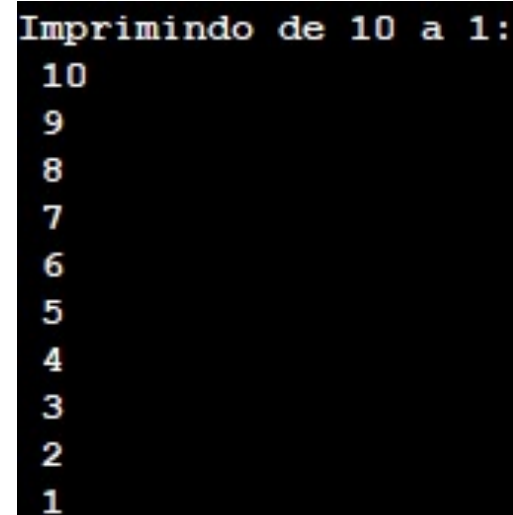
```
Imprimindo de 1 a 10:  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```


Estruturas de repetição **para**.

Exemplo

- No exemplo abaixo os números de 1 a 10 são exibidos em ordem decrescente. Utilizando a estrutura para e decrementando o valor inicial da variável j”:

```
algoritmo "ImprimeFor10a1"  
var  
    j: inteiro  
inicio  
    escreval ("Imprimindo de 10 a 1:")  
    para j de 10 ate 1 passo j-1 faca  
        escreval(j)  
    fimpara  
finalgoritmo
```



```
Imprimindo de 10 a 1:  
10  
9  
8  
7  
6  
5  
4  
3  
2  
1
```

Estruturas de repetição para.

- Analisando os exemplos anteriores verificamos que no primeiro não definimos o **passo** e que este foi incrementado automaticamente, aumentando o valor inicial da variável j em 1 a cada passagem pelo laço.
- No exemplo seguinte implementamos o decremento da variável j ($j-1$) a fim de que, iniciando o j em 10, ele atingisse o nosso objetivo que era subtrair 1 em cada passagem pelo laço(decrementar o j) até atingir 1.
- Entendemos desta forma que podemos controlar a estrutura da forma que desejarmos a fim de que se comporte como seja necessário para a lógica do programa.

Estruturas de repetição **para**.

Exercícios

Utilizando a estrutura **for** desenvolva os programas abaixo:

1. Crie um programa que receba do usuário um número e apresente a Tabuada deste.
2. Crie um programa que apresente os múltiplos de dois entre 0 e 100.
3. Crie um programa que imprima os múltiplos de 3 entre dois números digitados pelo usuário.

Estruturas de repetição **para**.

Exercícios

Crie um programa que, utilizando a estrutura **escolha caso**, permita ao usuário escolher a operação a realizar (depósito ou saque ou transferência), caso a operação seja de transferência solicite o nome do banco, da agência e conta, receba as informações e, em ao final exiba o valor inicial, a operação realizada e o saldo atual, no caso de transferência exiba também os dados do banco, agência e conta.

Altere o programa acima a fim de repetir a operação, por tantas vezes quanto o usuário desejar inicialmente, implemente a solução utilizando a estrutura **para**. Por exemplo o usuário quer fazer um depósito um saque e uma transferência então, no início do programa ele define que irá realizar 3 operações.

**Veremos na próxima
aula**

- Estruturas de Repetição - enquanto(while)



Paylivre be.academy



Resumo da aula anterior

- Estruturas de Repetição - para(for)

Veremos nesta aula

- Estruturas de Repetição - enquanto(while)

Estruturas de repetição Enquanto ... faça

-

Estrutura que repete uma sequência de comandos **enquanto** uma determinada condição(lógica) é satisfeita. Sua sintaxe é vista a seguir:

```
enquanto <expressão-lógica> faça  
    <comandos>
```

```
fimenquanto
```

<expressão-lógica> - expressão que é avaliada **antes** de cada repetição do laço. Enquanto seu resultado for **verdadeiro**, <comandos> são executados.

fimenquanto - Indica o fim da repetição dos <comandos>. Cada vez que a execução atinge este ponto, volta-se ao início do laço para que <expressão-lógica> seja avaliada novamente.

Se o resultado desta avaliação for **verdadeiro**, os <comandos> serão repetidos senão a execução prosseguirá a partir do primeiro comando após o **fimenquanto**.

Estruturas de repetição enquanto.

Reparamos que nesta estrutura não sabemos inicialmente quantas vezes o laço será repetido. Tal repetição dependerá do teste a ser feito e não de um valor já sabido como vimos na estrutura [para](#).

Desta forma esta estrutura é indicada para quando não sabemos, inicialmente, quando a repetição irá se encerrar e esta resposta virá durante a execução das repetições, quer por um incremento ou decremento de variável que torne o teste falso, quer por uma vontade do usuário de encerrar a repetição, sendo que tudo isto nós que definimos enquanto estamos planejando nossa aplicação.

Por exemplo enquanto o usuário responder que deseja realizar outra operação (depósito, transferência ou saque) nós repetimos os passos para ele.

Estruturas de repetição **enquanto**.

Exemplo

Vejamos um exemplo de sua utilização em VisualG.

```
algoritmo "ImprimeEnquanto1a10"
```

```
var
```

```
    j: inteiro
```

```
inicio
```

```
    escreval ("Imprimindo de 1 a 10 com enquanto:")
```

```
    j<-1
```

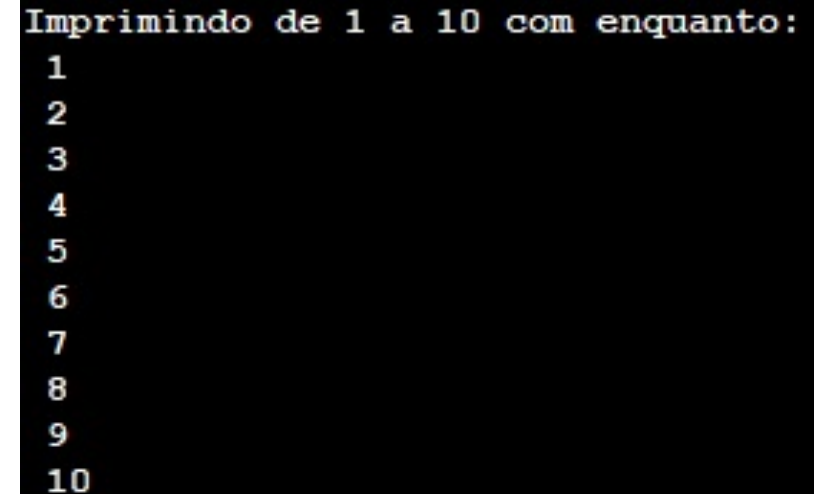
```
    enquanto j < 11 faça
```

```
        escreval(j)
```

```
        j <- j+1
```

```
    fimenquanto
```

```
fimalgoritmo
```



```
Imprimindo de 1 a 10 com enquanto:  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

Estruturas de repetição **enquanto**.

Repare que nesta estrutura por não sabermos quantas vezes o laço será repetido implementamos o incremento da variável `j` no corpo do laço (`j <- j+1`).

Assim a cada passagem o **teste** será feito e caso resulte em `verdadeiro(j < 11)` o processo de impressão será realizado `escreval(j)` senão, caso retorne **falso**, a execução continua após o **fimenquanto**.

Estruturas de repetição **enquanto**.

Exercícios

Utilizando a estrutura **de repetição while**,

1. Crie um programa que, imprima a tabuada de um número digitado pelo usuário.
2. Crie um programa que solicite ao usuário o seu nome e senha do cartão e valide se a senha e nome são corretos (Nome: Marcos e senha: paylivre) e, caso positivo, dê boas vindas ao usuário e, em caso negativo, solicite os dados novamente.
3. Crie um programa que realize as 4 operações matemática a partir de dois números que serão digitados pelo usuário. Após isto imprima os valores na tela e em seguida pergunte se ele quer realizar novo cálculo, repetido as operações enquanto o usuário desejar continuar.

Estruturas de repetição enquanto.

Exercícios

Crie um programa que permita ao usuário escolher a operação a realizar:

1. Depósito;
2. Saque;
3. Transferência;
4. Empréstimo;

Em seguida solicite os dados para concretizar a operação e imprima o nome da operação feita e os dados referentes a ela. Exemplo:

Depósito no Banco BOM agencia: 12-3 conta 1234-5 para Marcos Monteiro. Saldo inicial: R\$ 1000,00 Depósito: R\$500,00 Saldo final: R\$ 1500,00.

O programa deverá permitir ao usuário realizar quantas operações ele desejar, implemente a solução utilizando a estrutura **enquanto**.

**Veremos na próxima
aula**

- Estruturas de Repetição – repita-ate(do-while)



Paylivre be.academy



Resumo da aula anterior

- Estruturas de Repetição - enquanto(while)

Veremos nesta aula

- Estruturas de Repetição – repita-ate (faça-enquanto do-while)

Estruturas de Repetição – **repita - ate**(do-while)

- A Estrutura de Repetição – **repita - ate**(do-while) funciona de forma semelhante a estrutura **enquanto**(while).
- A grande diferença entre elas é que na estrutura **enquanto**(while) primeiro o teste é executado e depois a execução dos comandos, enquanto o teste for verdadeiro.
- Na estrutura **repita - ate** primeiro os comandos são executados e somente depois é que o teste é realizado. Desta forma os testes são executados pelo menos uma vez, e se repetirão até que a condição seja atendida.
- Desta forma podemos afirmar que na estrutura **repita - ate** os comandos sempre serão executados pelo menos uma vez.

Estruturas de Repetição – **repita - ate**(do-while)

- Sintaxe da estrutura repita-ate:

repita

<sequência-de-comandos>

ate <expressão-lógica>

repita	Indica o início do laço.
ate <expressão-lógica>	Indica o fim da <sequência-de-comandos> a serem repetidos. Cada vez que o programa chega neste ponto, <expressão-lógica> é avaliada: se seu resultado for FALSO, os comandos presentes entre esta linha e a linha repita são executados; caso contrário, a execução prosseguirá a partir do primeiro comando após esta linha.

Estruturas de Repetição – **repita - ate(do-while)**

Exemplo

- Estruturas de Repetição – repita - ate(do-while)

algoritmo "ImprimeRepitaAte1a10"

var

j: inteiro

inicio

escreval ("Imprimindo de 1 a 10 Repita Ate:")

j<-1

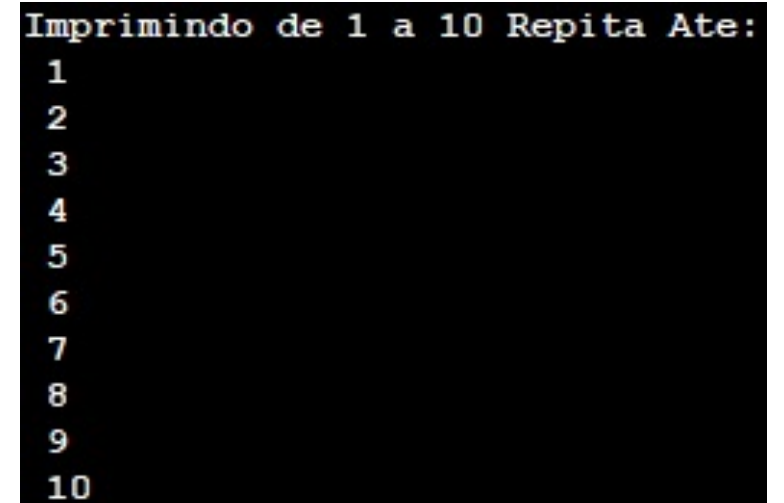
repita

escreval(j)

j <- j+1

ate j > 10

fimalgoritmo



```
Imprimindo de 1 a 10 Repita Ate:
1
2
3
4
5
6
7
8
9
10
```

Estruturas de Repetição – **repita - ate(do-while)**

- Observe que os comandos de repetição serão executados até que a condição(**ate** $j > 10$) seja atendida.
- Mesmo que inicializemos a variável **J** com o valor superior ao teste, por exemplo 11, ainda assim os comandos serão executados pelo menos uma vez.
- Esta estrutura é muito útil em situações em que pelo menos uma vez alguns comandos precisam ser executados, como o caso de um Login de usuário com senha.
- Neste caso a tela será exibida para o usuário digitar o Nome de login e senha e esta exibição irá se repetir até que o Login e senha estejam corretos.

Estruturas de Repetição – **repita - ate(do-while)**

- O comando **interrompa**(break) pode ser utilizado em qualquer das estruturas de repetição que vimos até aqui.
- Este comando irá interromper a estrutura de repetição e continuar o programa normalmente após esta estrutura.

Estruturas de Repetição – **repita - ate(do-while)**

Exemplo

- Estruturas de Repetição – repita - ate(do-while) com INTERROMPA.

algoritmo "ImprimeRepitaAte1a10"

var

j: inteiro

inicio

escreval ("Imprimindo de 1 a 10 Repita Ate:")

j<-1

repita

escreval(j)

j <- j+1

se j=5 entao

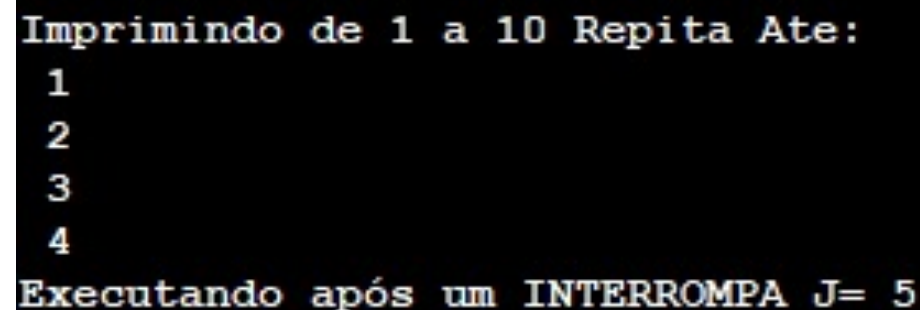
interrompa

fimse

ate j > 10

escreval ("Executando após um INTERROMPA J=", j)

fimalgoritmo



```
Imprimindo de 1 a 10 Repita Ate:
1
2
3
4
Executando após um INTERROMPA J= 5
```


Estruturas de Repetição – **faça-enquanto**(do-while)

- Verificamos que, no exemplo anterior, colocamos uma condição de teste (**se $j=5$**) onde quando isto ocorrer o programa deverá interromper a estrutura de repetição.
- Notamos também que mesmo após a interrupção da estrutura **repita-ate** o programa continua funcionando e imprime, ao final, o uma mensagem com o valor de j.
- Cabe reforçar que este recurso pode ser utilizado nas três estruturas de repetição que aprendemos.

Estruturas de Repetição – **repita - ate(do-while)**

Exercícios

1. Crie um programa que permita ao usuário tentar logar em seu Sistema informando seu nome e senha. Repita a operação até que o nome e senha correspondam a um valor armazenado(Marcos – 1234). Caso o usuário digite -1 interrompa a repetição e informe que o programa será finalizado por solicitação do usuário.
2. Solicitar a idade de várias pessoas e imprimir:
 - Total de pessoas com menos de 18 anos.
 - Total de pessoas com mais de 60 anos.
 - O programa termina quando idade for =-99.
3. Apresentar o total da soma obtida dos cem primeiros números inteiros.

Estruturas de Repetição – **repita - ate**(do-while)

Exercícios

- Crie um programa inicie o saldo do cliente com R\$ 1000,00 e que permita o saques consecutivos no valor de R\$ 150.00 até que seu saldo seja positivo.
- Quando isto ocorrer interrompa a operação e informe que o saque não pode ser efetuado em razão de que o saldo era insuficiente para efetuar a operação

**Veremos na próxima
aula**

- Estruturas de Dados - Vetor



Paylivre be.academy



Resumo da aula anterior

- Estruturas de Repetição – repita-ate(do-while) e o comando Interrompa.
- .

Veremos nesta aula

- Estruturas de Dados – Vetor.

Estruturas de Dados

– Vetores

- Vimos até aqui que o VisualG tem quatro tipos de dados: inteiro, real, cadeia de caracteres e lógico (ou booleano). Sendo:
- inteiro: permite definir variáveis numéricas sem casas decimais.
- real: permite definir variáveis numéricas do tipo real, ou seja, com casas decimais.
- caractere ou character: permite definir variáveis do tipo string(texto), ou seja, cadeia de caracteres.
- logico: permite definir variáveis do tipo booleano, ou seja, com valor verdadeiro ou falso.
- O VisualG também permite também a declaração de variáveis **estruturadas** através da palavra-chave **vetor**, como veremos nesta aula.

Estruturas de Dados

– Vetores

- **Vetor** é, conceitualmente falando, uma coleção de variáveis do mesmo tipo de dado, identificadas por um mesmo nome e acessadas através de seus índices.
- Os vetores são facilmente identificados em nosso programa por terem a sua frente um par de colchetes. Por exemplo `nome[1]<- "Marcos"`, neste caso estou armazenando o nome “Marcos” no índice 1 do vetor identificado como nome.

Estruturas de Dados

– Vetores

- Para criar um vetor agimos de forma semelhante a criação de uma variável comum porém para um vetor precisamos informar a quantidade de elementos(posições que ele terá). Vejamos:

`nomesAlunos: vetor[1..3] de caractere`

- A declaração cria um vetor de forma semelhante a uma linha de uma tabela no Excel contendo 3 colunas.

vt[1]	vt[2]	vt[3]
35	40	45

- Podemos guardar 3 valores de caracteres dentro deste vetor identificando cada posição pelo seu índice(número que irá dentro dos colchetes).

Estruturas de Dados

– Vetores

- Utilizaremos vetores sempre que necessitarmos de criar várias variáveis do mesmo tipo de dado. Desta forma, ao invés de declarar cinco variáveis do tipo caractere posso declarar apenas um vetor de cinco posições:

`nomesAlunos: vetor[1..5] de caractere`

- Para atribuir valores para o vetor acima eu uso a mesma lógica de atribuição para variáveis porém, no caso dos vetores, passo entre colchetes o índice onde desejo armazenar o valor dentro do vetor. Exemplo:

`nomesAlunos [1] <- “Marcos”`

Estruturas de Dados

– Vetores

- Como o **vetor** possui uma quantidade grande de variáveis, atribuir valores uma por uma, dentro do Código, seria um trabalho longo e muito repetitivo.
- Para facilitar nosso trabalho utilizamos uma **estrutura de repetição** do tipo **para** a fim de repetirmos todos os passos para **cadastro** de valores no vetor ou para **leitura** destes valores.
- Desta forma ao invés de termos 100 linhas para cadastrar os nomes de 100 alunos teremos um laço que a cada passagem irá incrementar o índice do vetor e fazer a atribuição dos nomes para ele.

Estruturas de Dados

– Vetores

Exemplo

Vejamos um exemplo:

Algoritmo "ExemploVetor"

Var

alunos: vetor [1..3] de caractere

i: inteiro //índice do vetor

Inicio

para i<-1 ate 3 faca //laço de cadastro

escreva("Digite o nome do ", i , "º aluno:")

leia(alunos[i])

fimpara

para i<-1 ate 3 faca //laço de impressão

escreval("Nome do ", i , "º aluno: ", alunos[i])

fimpara

Fimalgoritmo

```
Digite o nome do 1º aluno:Marcos
Digite o nome do 2º aluno:Maria
Digite o nome do 3º aluno:Manuel
Nome do 1º aluno: Marcos
Nome do 2º aluno: Maria
Nome do 3º aluno: Manuel
```

Estruturas de Dados

– Vetores

- Observamos no exemplo anterior que foi criado um vetor para armazenar nomes e o tamanho dele é de 3 posições.
- Na sequência criamos um laço para percorrer cada uma das posições do vetor utilizando a variável *i* como índice. Desta forma fizemos a atribuição dos nomes individualmente para cada índice do vetor no primeiro laço.
- No Segundo laço apenas percorremos todos os índices extraindo e escrevendo na tela os nomes no que estavam armazenados no vetor.

Estruturas de Dados

– Vetores

Exemplo

- Vamos agora criar dois vetores.
- O primeiro vetor irá armazenar os nomes dos alunos;
- O Segundo vetor irá armazenar a nota;

Algoritmo "ExemploVetorNomesNotas"

Var

alunos: vetor [1..3] de caractere

notas: vetor [1..3] de real

i: inteiro //índice do vetor

Inicio

para i<-1 ate 3 faca //laço de cadastro

escreva("Digite o nome do ", i , "º aluno:")

leia(alunos[i])

escreva("Digite a nota do ", i , "º aluno:")

leia(notas[i])

fimpara

para i<-1 ate 3 faca //laço de impressão

escreval(alunos[i], " obteve a nota: ", notas[i])

fimpara

Fimalgoritmo

Estruturas de Dados

– Vetores

Exemplo

- Como saída de nosso programa teremos:

```
Digite o nome do 1º aluno:Marcos
Digite a nota do 1º aluno:7
Digite o nome do 2º aluno:Lucas
Digite a nota do 2º aluno:8
Digite o nome do 3º aluno:Marcela
Digite a nota do 3º aluno:10
Marcos obteve a nota: 7
Lucas obteve a nota: 8
Marcela obteve a nota: 10
```

- Analisando o Código do exemplo verificamos que apenas criamos um novo **vetor** de um tipo **real** chamado **notas[]** que serviu de base para cadastro e leitura das notas dos alunos.
- Desta forma a economia de linhas de Código é ainda maior.

Estruturas de Dados

– Vetores

Exercícios

1. Faça um programa que solicite ao usuário o nome e o preço de 10 produtos e armazene-os em um vetor. Ao final imprima o nome e os valores correspondentes dos produtos.
2. Crie um programa que permita cadastrar os seguintes dados de um Aluno: Nome, nota1 e nota2. Receba estes valores em vetores e calcule e exiba ao final todos os dados e a informação se o aluno foi aprovado(media maior ou igual a 6) ou reprovado(media inferior a 6).

Estruturas de Dados

– Vetores

Exercícios

1. Crie um programa que permita ao usuário cadastrar 5 clientes com os seguintes dados:
 - Nome;
 - CPF;
 - RG;
 - Endereço; e
 - Telefone.
2. Guarde os dados dos clientes em um vetor e ao final exiba-os.

**Veremos na próxima
aula**

- Estruturas de Dados - Matriz.



Paylivre be.academy



Resumo da aula anterior

- Estruturas de Dados - Vetores.

Veremos nesta aula

- Estruturas de Dados - Matriz.

Estruturas de Dados

– Matriz

- As matrizes são arrays ou vetores multidimensionais. Isso significa que elas podem ter duas ou mais dimensões.
- O uso mais comum de matrizes é para construção de estruturas de dados bidimensionais. Nesse caso, elas são usadas para armazenar valores em forma de linhas e colunas, tal como ocorre em uma tabela.

	vt[1]	vt[2]	vt[3]
vt[1]	10	20	35
vt[2]	30	60	55
vt[3]	50	80	75

Estruturas de Dados

– Matriz

- Na criação de uma matriz bidimensional, especifica-se a quantidade de linhas e de colunas que ela conterá.
- Em muitas linguagens uma Matriz é conhecida como vetor de vetores ou array de arrays.
- Assim como os vetores, as matrizes também precisam ser declaradas, iniciadas e consultadas. A sintaxe para realizar estas operações com matrizes é muito similar a sintaxe utilizada para vetores. Vejamos:

`notas: vetor [1..3 , 1..2] de real`

- A instrução acima cria uma Matriz com 3 linhas e duas colunas, ou seja, nos permitiria gravar duas notas para cada aluno em um total de 3 alunos.

Estruturas de Dados

– Matriz

Exemplo

- Vejamos um exemplo:

Algoritmo "ExemploMatrizNomesNotas"

Var

alunos: vetor [1..3] de caractere

notas: vetor [1..3,1..2] de real

l, c : inteiro //índice l para linhas e c para colunas

Inicio

para l<-1 ate 3 faca //laço de cadastro das linhas

escreva("Digite o NOME do ", l , "º aluno: ")

leia(alunos[l])

para c<-1 ate 2 faca //laço de cadastro das
colunas

escreva("Digite a ", c, "ª nota do ", l , "º aluno: ")

)

leia(notas[l , c])

fimpara

fimpara

Fimalgoritmo

Estruturas de Dados

– Matriz

Exemplo

- Vejamos o resultado do nosso exemplo:

```
Digite o NOME do 1º aluno: Marcos
Digite a 1ª nota do 1º aluno: 8
Digite a 2ª nota do 1º aluno: 7
Digite o NOME do 2º aluno: Marcio
Digite a 1ª nota do 2º aluno: 9
Digite a 2ª nota do 2º aluno: 6
Digite o NOME do 3º aluno: Maria
Digite a 1ª nota do 3º aluno: 9
Digite a 2ª nota do 3º aluno: 9.5
```

- Observe que isto foi apenas o cadastro a seguir iremos fazer a impressão.

Estruturas de Dados

– Matriz

Exemplo

- A impressão funciona de forma semelhante ao cadastro porém apenas teremos o comando de escreva visto que os valores já foram coletados. Vou colocar abaixo apenas o código que deve ser inserido abaixo da instrução **fimpara** de cadastro.

```
escreval("Dados dos Alunos:" )  
para l<-1 ate 3 faca //laço das linhas  
    escreva("NOME do ", l , "º aluno: " , alunos[l])  
    para c<-1 ate 2 faca //laço das colunas  
        escreva(c, "ª nota: " , notas[l , c])  
    fimpara  
    escreval()  
fimpara
```

```
Dados dos Alunos:  
NOME do 1º aluno: Marcos 1ª nota: 8 2ª nota: 7  
NOME do 2º aluno: Marcio 1ª nota: 9 2ª nota: 6  
NOME do 3º aluno: Maria 1ª nota: 9 2ª nota: 9.5
```

Estruturas de Dados

– Matriz

- Observamos que para o cadastro e leitura de uma Matriz necessitamos de criar um **laço para** dentro de outro.
- O primeiro **laço** irá percorrer as **linhas** da Matriz e o **segundo as colunas**.

Estruturas de Dados

– Matriz

Exercícios

- Uma matriz quadrada é uma matriz que tem a mesma quantidade de linhas e colunas.
- A diagonal principal é obtida pelos índices iguais de linhas e colunas.
- A diagonal secundária pela seguinte regra: a soma dos índices da linha e coluna é igual ao tamanho da matriz menos um, ou seja, em uma Matriz de dimensão(tamanho) 3 sempre que a soma de seus índice forem 3-1 estamos na diagonal secundária.
- Dada esta explicação crie um programa que permita o cadastro de uma Matriz e exiba a soma da diagonal principal e secundária. Exemplo:

• Soma da Diagonal Principal = 73

• Soma da Diagonal Secundário = 130

Estruturas de Dados

– Matriz

Exercícios

- Uma matriz quadrada é uma matriz que tem a mesma quantidade de linhas e colunas.
- A diagonal principal é obtida pelos índices iguais de linhas e colunas.
- A diagonal secundária pela seguinte regra: a soma dos índices da linha e coluna é igual ao tamanho da matriz menos um, ou seja, em uma Matriz de dimensão(tamanho) 3 sempre que a soma de seus índice forem 3-1 estamos na diagonal secundária.
- Dada esta explicação crie um programa que permita o cadastro de uma Matriz e exiba a soma da diagonal principal e secundária. Exemplo:

• Soma da Diagonal Principal = 73

• Soma da Diagonal Secundário = 130

Estruturas de Dados

– Matriz

Exercícios

- Considere uma planilha de **5 linhas por 4 colunas**, a qual será representada por uma matriz bidimensional.
- Desenvolva um programa que faça a leitura, a partir do teclado, dos valores numéricos das primeiras **4 linhas e 3 colunas** da planilha.
- Realizada a leitura, armazenar a soma dos valores de cada linha na linha correspondente da última coluna. Finalmente, armazenar a soma dos valores de cada coluna na coluna correspondente da última linha da planilha. Imprima a planilha ao final.

Bibliografia

WIKIPEDIA. **Algoritmo**. 2022. Disponível em: <https://pt.wikipedia.org/wiki/Algoritmo>. Acesso em: 10 abr. 2022.

GOUVEIA, Rosimar. **Exercícios de Raciocínio Lógico**. 2022. Disponível em: <https://www.todamateria.com.br/raciocinio-logico-exercicios/>. Acesso em: 10 abr. 2022.

ANTONIO, Professor. **Manual do Visualg 3.0**. 2017. Disponível em: <https://manual.visualg3.com.br/doku.php?id=manual>. Acesso em: 10 abr. 2022.

Exercícios de revisão

- Faça um programa para calcular a área e o perímetro de um círculo. O usuário deve informar o valor do raio do círculo. Caso o raio seja um valor negativo, deve-se informar uma mensagem de **erro** e solicitar ao usuário que informe um novo valor. Utilize o valor de 3,1416 para π (pi).
Área = $PI * Raio * Raio$
Perímetro = $2 * PI * Raio$. Exiba ao final o valor do raio e o perímetro e a área.
- Faça um programa para calcular o valor de cada parcela de um financiamento a ser escolhido pelo usuário. A parcela será formada pelas seguintes regra:
 - Caso o usuário escolha 2 parcelas os juros serão de 2% ao mês;
 - Caso o usuário escolha 3 parcelas os juros serão de 3% ao mês;
- Ao final imprima o valor financiado, o total de parcelas e o valor de cada parcela.

Exercícios de revisão

- Faça um programa receba o salário do funcionário e aplique um desconto conforme tabela abaixo.

Salário de contribuição (R\$)	Alíquota
até 1.212,00	7,5%
de 1.212,01 até 2.427,35	9%
de 2.427,36 até 3.641,03	12%
de 3.641,04 até 7.087,22	14%

- Ao final imprima o valor do salário bruto o valor do desconto e o salário líquido.

Exercícios de revisão

- Faça um programa receba o salário do funcionário e calcule o IRPF a ser pago conforme tabela abaixo.

Base de cálculo	Alíquota	Dedução
Até R\$1.903,98	Isento	R\$ 0,00
R\$ 1.903,99 até R\$ 2.826,65	7,5%	R\$ 142,80
R\$ 2.826,66 até R\$ 3.751,05	15%	R\$ 354,80
R\$ 3.751,06 até R\$4.664,68	22,5%	R\$ 636,13
Acima de R\$ 4.664,68	27,5%	R\$ 869,36

- Ao final imprima o valor do salário bruto o valor do Imposto de Renda(IRPF) e o salário líquido.

Exercícios de revisão

- Faça um programa que, com base na tabela abaixo, solicite ao usuário qual o tipo de IOF que deseja calcular e, em seguida, solicite o valor sobre o qual será aplicado o IOF.

Cheque especial: 0,38% + 3% ao ano	Operações de câmbio: 0,38%
Financiamentos: 0,38% + 3% ao ano	Uso do cartão pré-pago ou de crédito no exterior: 6,38%
Seguro de automóveis: 7,38%	Fatura atrasada do cartão de crédito: 0,38% + 3% ao ano

- Ao final imprima o Valor, o Tipo(chegue especial, Financiamentos, etc) e o IOF a ser recolhido.
- Após isto pergunte ao usuário se deseja calcular outro e repita os passos enquanto a resposta for verdadeira.



Paylivre be.academy

