MESTRADO PROFISSIONAL ENSINO E SUAS TECNOLOGIAS

INSTITUTO FEDERAL Fluminense *Campus* Campos Centro

MINISTÉRIO DA EDUCAÇÃO

Apostila de Lógica de Programação

Mestrado Profissional em Ensino e suas Tecnologias – 2019
Campos Centro - IFFluminense

Apresentação

A presente apostila faz parte de um curso de extensão sobre Lógica de Programação ministrado no Ambiente Virtual de Aprendizagem (AVA) gamificado AGILE (*Attractive, Gamified, Interactive Learning Environment*) em maio de 2019.

Esta apostila foi desenvolvida por Isadora Lopes Barbosa Vasconcellos e suas orientadoras, Dra. Annabell Del Real Tamariz e Dra. Silvia Cristina Freitas Batista, para experimentação do AGILE com alunos dos cursos Técnico de Informática, Integrado ao Ensino Médio e Concomitante, do *campus* Campos Centro do Instituto Federal de Educação, Ciência e Tecnologia Fluminense (IFFluminense).

Tal apostila é composta pelos conteúdos de Lógica de Programação e pelas atividades avaliativas utilizadas no curso do AGILE, em formato de *quiz*, assim como seus respectivos gabaritos.

O AGILE e o curso de Lógica de Programação foram desenvolvidos durante uma pesquisa do Mestrado Profissional em Ensino e suas Tecnologias (MPET) do *campus* Campos Centro do IFFluminense.

Sumário

Módulo 1 - Algoritmos 4
1.1 ALGORITMOS 5
1.1.1 ATIVIDADES ALGORITMOS 9
1.2 FORMAS DE REPRESENTAÇÃO 10
1.2.1 ATIVIDADES FORMAS DE REPRESENTAÇÃO 16
1.3 TIPOS DE DADOS 17
1.3.1 ATIVIDADES TIPOS DE DADOS 20
Módulo 2 – Variáveis, Constantes e Operadores 21
2.1 CONSTANTES E VARIÁVEIS 22
2.1.1 ATIVIDADES CONSTANTES E VARIÁVEIS 25
2.2 ENTRADA E SAÍDA DE DADOS 26
2.2.1 ATIVIDADES ENTRADA E SAÍDA DE DADOS 28
2.3 OPERADORES 29
2.3.1 ATIVIDADES OPERADORES 36
Módulo 3 – Estruturas de Seleção 38
3.1 ESTRUTURAS DE SELEÇÃO 39
3.1.1 ATIVIDADES ESTRUTURAS DE SELEÇÃO 41
3.2 ESTRUTURA DE SELEÇÃO IF/ELSE 42
3.2.1 ATIVIDADES ESTRUTURA DE SELEÇÃO IF/ELSE 47
3.3 ESTRUTURA DE SELEÇÃO SWITCH/CASE 48
3.3.1 ATIVIDADES ESTRUTURA DE SELEÇÃO SWITCH/CASE 51
Módulo 4 – Estruturas de Repetição 52
4.1 ESTRUTURAS DE REPETIÇÃO 53
4.1.1 ATIVIDADES ESTRUTURAS DE REPETIÇÃO 55
4.2 ESTRUTURA DE REPETIÇÃO FOR 57
4.2.1 ATIVIDADES ESTRUTURA DE REPETIÇÃO FOR 59
4.3 ESTRUTURA DE REPETIÇÃO WHILE 60
4.3.1 ATIVIDADES ESTRUTURA DE REPETIÇÃO WHILE 62
4.4 ESTRUTURA DE REPETIÇÃO DO/WHILE 63
4.4.1 ATIVIDADES ESTRUTURA DE REPETIÇÃO DO/WHILE 65
Questionário Final 66
Gabaritos 71

Módulo 1 - Algoritmos

Conteúdos:

- Conceitos de Algoritmo
- Formas de Representação dos Algoritmos
- Conceitos de Tipos de Dados

Ao final deste módulo, você será capaz de:

- Estruturar a codificação básica de um Algoritmo
- Reconhecer os elementos da estrutura de um Algoritmo

Recursos:

- Vídeos
- Textos explicativos
- Atividades avaliativas

1.1 ALGORITMOS

Vídeo 1- Apresentação do curso

Link: https://youtu.be/vtp9WRc1awA

▪ Vídeo - Algoritmos

Vídeo 2- Conceitos iniciais sobre Algoritmos

Link: https://youtu.be/ZR72CDXiDCw

▪ O que é um algoritmo?

- Você sabe o que é um Algoritmo?

  Um Algoritmo é...

  - É uma linguagem intermediária entre a linguagem humana e as linguagens de programação;
  - É utilizado para representar a solução de um problema;
  - Descrevem instruções a serem executadas pelos computadores.

É a especificação de uma sequência ordenada de instruções, finitas e não-ambíguas, que deve ser seguida para a solução de um determinado problema, garantindo a sua repetibilidade.

▪ Algoritmos no dia a dia

- Aplicamos o conceito de algoritmo diariamente sempre que estabelecemos um planejamento mental para realizar uma determinada tarefa, considerando que deveremos executar um conjunto de passos até atingir o objetivo desejado.

Exemplos de algoritmos no dia a dia:

- Receitas culinárias;
- Manuais de instrução;
- Roteiros realização de tarefas específicas.

- Um dos vários exemplos do uso de algoritmos no nosso dia a dia são as receitas culinárias, pois estas possuem um conjunto de passos que devem ser seguidos para obter o resultado esperado.

Receita de Brigadeiro
1. Separar os ingredientes: 1 lata de leite condensado 1 colher de sopa de manteiga 4 colheres de sopa de chocolate em pó **2.** Colocar todos os ingredientes em uma panela; **3.** Misturar os ingredientes; **4.** Cozinhar a mistura em fogo médio até começar a soltar do fundo da panela. **5.** Desligar o fogo; **6.** Colocar o brigadeiro em refratário de vidro; **7.** Esperar o brigadeiro esfriar; **8.** Enrolar o brigadeiro em formato esférico; **9.** Passar o brigadeiro enrolado no granulado; **10.** Colocar o brigadeiro na forminha de papel.

▪ Para que serve um algoritmo?

- O algoritmo é uma sequência de passos lógicos e finitos que permite solucionar problemas;
- O objetivo de aprender a criar algoritmos é que este é a base de conhecimentos para as linguagens de programação;
- Em geral, existem muitas maneiras de resolver o mesmo problema. Ou seja, podem ser criados vários algoritmos diferentes para resolver o mesmo problema;
- Assim, ao criarmos um algoritmo, indicamos uma dentre várias possíveis sequências de passos para solucionar o problema.

▪ Algoritmo computacional

- Para que um computador possa desempenhar uma tarefa é necessário que esta seja detalhada, passo a passo, em uma linguagem compreensível pela máquina, por meio de um... **Programa**.

Um programa de computador é um algoritmo escrito em um formato compreensível pelo computador.

- Na elaboração de um algoritmo devem ser especificadas ações claras e precisas que resultem na solução do problema proposto;
- A lógica está na correta sequência de passos que deve ser seguida para alcançar um objetivo específico;
- O grau de detalhe do algoritmo dependerá da situação em que o programador se encontra.

▪ Propriedades essenciais

- Um Algoritmo deve ser:

Completo	Todas as ações precisam ser descritas e devem ser únicas.
Sem redundância	Um conjunto de instruções só pode ter uma única forma de ser interpretada.
Determinístico	Se as instruções forem executadas, o resultado esperado será sempre atingido.
Finito	As instruções precisam terminar após um número limitado de passos.

1.1.1 ATIVIDADES ALGORITMOS

1. **Dentre os exemplos abaixo, não pode ser considerado um algoritmo:**
   a) Guia de instalação do Ubuntu
   b) Manual de instruções de uso de micro-ondas
   c) Receita de sorvete
   d) Cardápio de restaurante

2. **A afirmação "O algoritmo é uma sequência de passos lógicos e infinitos e não-ambíguos que permitem solucionar problemas" é:**
   a) Verdadeira
   b) Falsa

3. **A afirmação "Um programa de computador é um algoritmo escrito em um formato compreensível pelo computador" é:**
   a) Verdadeira
   b) Falsa

1.2 FORMAS DE REPRESENTAÇÃO

▪ Formas de Representação

Você conhece alguma forma de representação (escrita) dos algoritmos?

- Existem diversas formas de representação de algoritmos, mas não há uma forma considerada a melhor;
- Entre as principais diferenças está o maior ou menor nível de detalhamento (grau de abstração).

Formas mais conhecidas de representação
Descrição narrativa
Fluxograma
Pseudocódigo (Linguagem estruturada ou Portugol)

- Cada uma das formas de representação possui vantagens e desvantagens;
- Cabe ao programador escolher qual forma oferece as melhores características de acordo com a situação/problema;
- É comum a combinação das representações, principalmente quando há a necessidade do entendimento por vários tipos de pessoas.

▪ Descrição Narrativa

- Os algoritmos são expressos diretamente em linguagem natural. Ou seja, a sequência de passos é descrita em nossa língua nativa (português).

Exemplo:

- Cálculo da média de um aluno:

- Obter as suas 2 notas de provas;
- Calcular a média aritmética;
- Se a média for maior ou igual a 7, o aluno foi aprovado;
- Senão o aluno foi reprovado.

Aspecto positivo	Aspecto negativo
Não é necessário aprender novos conceitos, pois a língua natural já é bem conhecida.	A língua natural dá oportunidade para várias interpretações e ambiguidades, dificultando a transcrição desse algoritmo para programa.

▪ Fluxograma

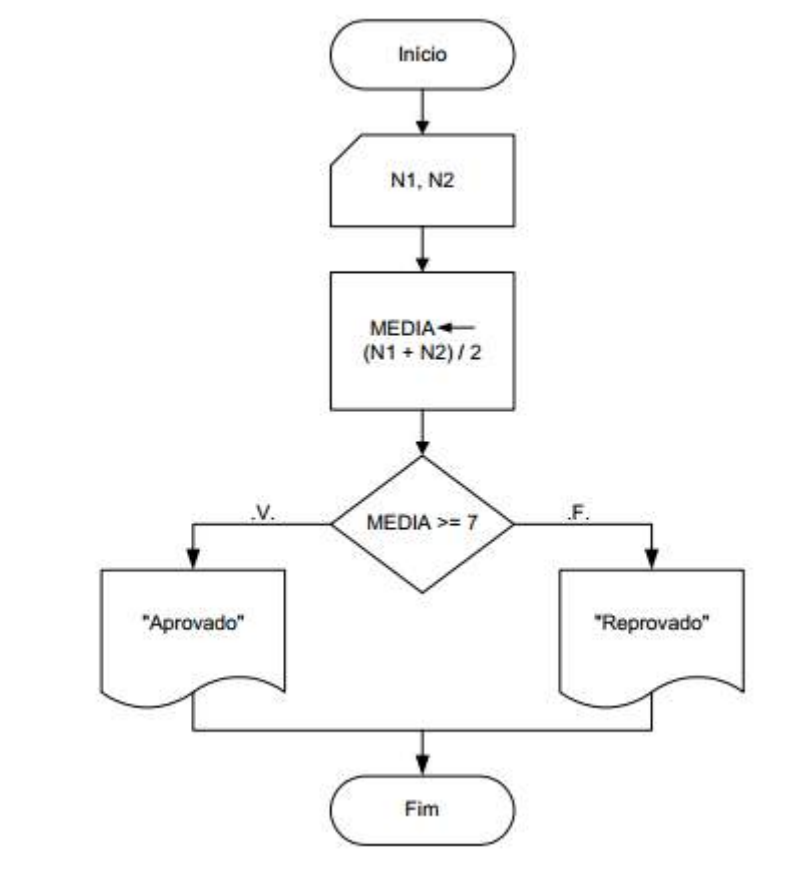
- É uma representação gráfica em que formas geométricas diferentes implicam ações (instruções, comandos) distintos;
- É mais precisa que a Descrição Narrativa, porém não se preocupa com detalhes de implementação do programa, como o tipo das variáveis utilizadas.

Elementos gráficos

- O fluxograma utiliza símbolos específicos para a representação gráfica dos algoritmos;
- Os símbolos sofrem algumas variações de acordo com o autor ou ferramenta em uso.

Exemplo:

- Cálculo da média de um aluno:



Aspecto positivo	Aspecto negativo
O entendimento de elementos gráficos é mais simples que o entendimento de textos.	Os fluxogramas devem ser entendidos e o algoritmo resultante não é detalhado, dificultando sua transcrição para um programa.

▪ Pseudocódigo

- É rico em detalhes, como a definição dos tipos das variáveis usadas no algoritmo.

Estrutura básica do pseudocódigo
Algoritmo <nome_do_algoritmo> <declaração_de_variáveis> **Início** <corpo do algoritmo> **Fim**

Algoritmo	Palavra que indica o início da definição de um algoritmo em forma de pseudocódigo.
<nome_do_algoritmo>	Nome simbólico dado ao algoritmo com a finalidade de distingui-lo dos demais.
<declaração_de_variáveis>	Parte opcional onde são declaradas as variáveis globais usadas no algoritmo.
Início e Fim	Palavras que delimitam o início e o término, respectivamente, do conjunto de instruções do corpo do algoritmo.

Exemplo:

- Cálculo da média de um aluno:

```
Algoritmo Calculo_Media
Var Nota1, Nota2, MEDIA: real;
Início
   Leia Nota1, Nota2;
   MEDIA ← (Nota1 + Nota2) / 2;
   Se MEDIA >= 7 então
      Escreva "Aprovado";
   Senão
      Escreva "Reprovado";
   Fim_se
Fim
```

Aspecto positivo	Aspecto negativo
Representação clara sem as especificações de linguagem de programação.	As regras do pseudocódigo devem ser aprendidas.

1.2.1 ATIVIDADES FORMAS DE REPRESENTAÇÃO

1. **As formas de representação de algoritmo mais conhecidas são?**
   a) Fluxograma, Descrição narrativa, Pseudocódigo
   b) Diagrama de classe, Fluxograma, Pseudocódigo
   c) Pseudocódigo, Prototipagem, Fluxograma
   d) Pseudocódigo, Fluxograma, Modelagem de dados
   e) Descrição narrativa, prototipagem, fluxograma

2. **A afirmação "É um consenso entre os programadores que a melhor forma de representação de um algoritmo é a descrição narrativa" é:**
   a) Verdadeira
   b) Falsa

3. **A afirmação "O fluxograma utiliza símbolos específicos, que podem variar de acordo com a ferramenta, para representar graficamente os algoritmos" é:**
   a) Verdadeira
   b) Falsa

1.3 TIPOS DE DADOS

▪ Instruções X Dados

- As informações manipuladas pelo computador podem ser classificadas em:

Instruções	Dados
Coordenam o funcionamento do computador, determinando a maneira como os dados devem ser tratados.	São as informações a serem processadas pelo computador.

▪ Tipos de Dados

- Os dados podem ser do tipo:
  - Numérico;
  - Literal;
  - Lógico.

▪ Dados Numéricos

- Os dados numéricos representáveis em um computador são divididos em duas classes: **INTEIROS** e **REAIS**.

Dados numéricos Inteiros	Dados numéricos Reais
Os números inteiros são aqueles que não possuem componentes decimais ou fracionários, podendo ser positivos ou negativos.	Os números reais são aqueles que podem possuir componentes decimais ou fracionários, positivos ou negativos.

Exemplos:

- Dados Numéricos Inteiros:

10 - número inteiro positivo

0 - número inteiro

-10 - número inteiro negativo

- Dados Numéricos Reais:

20.05 - número real positivo com duas casas decimais

110. - número real positivo com zero casas decimais

-15.2 - número real negativo com uma casa decimal

0. - número real com zero casas decimais

▪ Dados Literais

- Os dados literais são sequência de caracteres que podem ser **letras**, **dígitos** e **símbolos especiais**.
- São representados nos algoritmos, pelo **delimitador aspas (“)** no seu início e término.

Exemplos:

"AbCdefGHi" - literal de comprimento 9

"1.2" - literal de comprimento 3

“0” - literal de comprimento 1

***Note que, "1.2" representa um dado do tipo literal, diferindo de 1.2 que é um dado do tipo real, devido às aspas.**

▪ Dados Lógicos

- Os dados lógicos são usados para representar os dois únicos valores lógicos possíveis: **Verdadeiro** e **Falso**.
- Seus pares valores podem representados por meio de outros tipos, como: **sim/ não**, **1/0**, **true/false**.

Exemplos:

V - valor lógico verdadeiro

F - valor lógico falso

▪ Esquema dos tipos de dados

1.3.1 ATIVIDADES TIPOS DE DADOS

1. **Os tipos de dados podem ser:**
   a) Inteiro, Literal, Lógico
   b) Numérico, Literal, Lógico
   c) Literal, Caractere, Imagem
   d) Real, Caractere, Lógico
   e) Numérico, Imagem, Caractere

2. **O tipo de dado Lógico pode assumir os valores: verdadeiro, falso e zero.**
   a) Verdadeiro
   b) Falso

3. **O tipo de dado literal é uma sequência de caracteres que podem ser:**
   a) Somente letras
   b) Somente letras e números
   c) Somente letras e caracteres especiais
   d) Letras, números e caracteres especiais
   e) Somente caracteres especiais e números

Módulo 2 – Variáveis, Constantes e Operadores

Conteúdos:

- Conceitos de Constantes e Variáveis
- Comandos de Entrada e Saída de Dados
- Conceitos de Tipos de Operadores

Ao final deste módulo, você será capaz de:

- Inicializar Constantes e Variáveis
- Utilizar Operadores

Recursos:

- Vídeos
- Textos explicativos
- Atividades avaliativas

2.1 CONSTANTES E VARIÁVEIS

▪ Vídeo - Constantes, Variáveis e Operadores

Vídeo 3- Conceitos iniciais sobre Constantes, Variáveis e Operadores

Link: https://youtu.be/AXkKQgPYkLQ

▪ O que é uma Constante?

Você sabe o que é uma Constante?

- Em programação, uma constante armazena um valor fixo, que NÃO mudará com o tempo de execução do programa. Ou seja, o valor será definido uma única vez e jamais será alterado durante a execução da aplicação;
- Uma constante deve ser utilizada quando uma informação NÃO tem qualquer possibilidade de alteração, ou variação, no decorrer da execução do algoritmo (programa).

Exemplos:

pi (π): 3,1415926

Velocidade da luz no vácuo: 299 792 458 m/s

▪ O que é uma Variável?

Agora que você sabe o que é uma Constante... O que seria uma Variável?

- É uma entidade destinada a guardar uma informação;
- Chama-se variável, pois o valor contido nesta varia com o tempo, ou seja, não é um valor fixo;
- O conteúdo de uma variável pode ser alterado, consultado ou apagado quantas vezes forem necessárias no algoritmo;
- Ao alterar o conteúdo de uma variável, a informação anterior é perdida. Ou seja, a variável armazena sempre a última informação recebida;
- Em geral, uma variável possui três atributos: **nome**, **tipo de dado** e a **informação** por ela guardada.

Nome	Deve começar com uma letra e não deve conter nenhum carácter especial, exceto o underline (_).
Tipo de dados	Pode ser do tipo numérico, literal ou lógico.
Informação	De acordo com o tipo de dado definido.

Exemplos:

VAR NOME **:literal[50]**

IDADE **:inteiro**

SALARIO **:real**

TEM_FILHOS **:lógico**

- **Regras para nomeação de variáveis:**

▪ Devem ser iniciadas sempre por uma letra;
▪ Não devem conter caracteres especiais;
▪ Não devem conter espaços em branco;
▪ Não devem conter hífen entre os nomes (utilize underline).

▪ Atribuição de valores

- É utilizada para atribuir um valor a uma variável, ou seja, para armazenar um determinado conteúdo em uma variável;
- A operação de atribuição, geralmente, é representada, nos algoritmos, por uma seta apontando para a esquerda.

Exemplos:

variável ← **constante** Ex.: idade ← 12

//Variável recebe valor constante

variável ← **variável** Ex.: preço ← valor

//Variável recebe valor de outra variável

variável ← **expressão** Ex.: A ← B + C

//Variável recebe valor de uma expressão

2.1.1 ATIVIDADES CONSTANTES E VARIÁVEIS

1. **A afirmação “Uma constante armazena um valor fixo, que mudará com o tempo de execução do programa” é:**
   a) Verdadeira
   b) Falsa

2. **É um nome válido para a declaração de uma variável:**
   a) *nome
   b) data de nascimento
   c) data_de_inicio
   d) 1ºnumero
   e) novo-salario

3. **A afirmação “O conteúdo de uma variável pode ser alterado, consultado ou apagado quantas vezes forem necessárias no algoritmo” é:**
   a) Verdadeira
   b) Falsa

2.2 ENTRADA E SAÍDA DE DADOS

▪ Entrada e Saída de dados

- Existem basicamente duas instruções principais em algoritmos que são: **Leia** e **Escreva**.

Leia	Escreva
A instrução **Leia** é utilizada quando se deseja obter informações do usuário por meio do teclado, ou seja, é um **Comando de Entrada de Dados**.	A instrução **Escreva** é utilizada para mostrar informações na tela do computador, ou seja, é um **Comando de Saída de Dados**.

▪ Lendo instruções

- Usa-se a instrução **Leia**, quando é necessário que o usuário do algoritmo digite algum dado;
- A instrução de **entrada de dados (Leia)** será responsável pela leitura e armazenamento desses dados na **variável** indicada.

Sintaxe:

leia (variável);

▪ Escrevendo instruções

- Usa-se a instrução **Escreva** quando é necessário mostrar algum dado do algoritmo para o usuário;
- A instrução de **saída de dados (Escreva)** será responsável pela exibição dos dados da **variável**, **constante** ou **expressão** na tela do computador.

Sintaxe:

escreva (variável);

▪ Comentários

- A inserção de comentários no decorrer do algoritmo facilita a leitura deste por outros programadores;
- Os comentários também servem para auxiliar o programador a relembrar o próprio código depois de um tempo sem utilizá-lo.

Sintaxe:

//comentário

▪ Sugestões

- **Na escrita do algoritmo (pseudocódigo):**
  - Incluir **comentários** nas linhas mais importantes do programa;
  - Utilizar **nomes significativos** (que ajudem a identificar o conteúdo) para as variáveis e constantes;
  - Efetuar a **indentação** (alinhamento) das linhas para facilitar a leitura.

Algoritmo de exemplo:

```
Algoritmo entrada_saida_dados
Início
   var nome :literal; //Cria a variável nome do tipo literal
   escreva ("Digite seu Nome"); //Solicita que seja digitado o nome
   leia (nome); //Lê e armazena na variável nome o valor digitado
   escreva ("Bom dia", nome); //Escreve a mensagem + nome
Fim
```

2.2.1 ATIVIDADES ENTRADA E SAÍDA DE DADOS

1. **São comando utilizados nos algoritmos para representar as instruções de entrada e saída de dados:**
   a) Entrada; Saída
   b) Open; Close
   c) Leia; Escreva;
   d) Informe; Leia
   e) Escreva; Importe

2. **Qual alternativa abaixo corresponde a uma maneira adequada de inserir comentários em algoritmos:**
   a) *comentário*
   b) “comentário”
   c) !comentário
   d) %comentário
   e) //comentário

3. **A afirmação “É utilizada quando se deseja obter informações do usuário por meio do teclado” se refere a instrução de:**
   a) Entrada de dados (Leia)
   b) Saída de dados (Escreva)

2.3 OPERADORES

▪ O que são Operadores?

Você sabe o que são Operadores?

- Operadores são símbolos que representam atribuições, cálculos e ordem dos dados;
- As operações possuem uma ordem de prioridades (alguns cálculos são processados antes de outros);
- Os operadores são utilizados nas expressões matemáticas, lógicas, relacionais e de atribuição.

▪ Tipos de Operadores?

Quanto ao número de operandos sobre os quais atuam	
Unários: quando atuam sobre um único operando.	**Binários:** quando atuam sobre dois operandos, que podem ser: **duas variáveis**, **duas constantes**, ou **uma variável** e **uma constante**.

Exemplos:

Unário:

-x (o valor armazenado no operando x passa a ser negativo)

x++ (incrementa +1 na variável x).

Obs.:

++ significa adicionar +1 ao valor da variável

-- significa diminuir -1 do valor da variável

Binário:

z= x+y (somatória entre as variáveis x e y)

z=x+7 (somatória entre uma variável e uma constante)

Quanto ao tipo de dado dos operandos e do valor resultante de sua avaliação
✓ Operadores Aritméticos; ✓ Operadores de Atribuição; ✓ Operadores Lógicos; ✓ Operadores Relacionais.

▪ Operadores Aritméticos

- Conjunto de símbolos que representa as operações básicas da matemática como: somar, subtrair, multiplicar, dividir e etc.
- Esses operadores somente poderão ser utilizados entre variáveis com os tipos de dados numéricos inteiros e/ou numéricos reais.

Operadores Aritméticos		
Adição **+**	**Divisão** **/**	**Negativo unário** **-**
Subtração **-**	**Restou ou módulo** **%**	**Incremento** **++**
Multiplicação *****	**Positivo unário** **+**	**Decremento** **--**

- **Obedecem às regras matemáticas comuns:**
  - As expressões de dentro de parênteses são sempre resolvidas antes das expressões fora dos parênteses;
  - Quando existe um parêntese dentro de outro, a solução sempre inicia do parêntese mais interno até o mais externo (de dentro para fora);
  - Quando duas ou mais expressões tiverem a mesma prioridade, a solução é sempre iniciada da expressão mais à esquerda até a mais à direita.

- **Obedecem às regras matemáticas comuns:**

Operador	Operação	Prioridade
^, **	Exponenciação	1
/	Divisão	2
*	Multiplicação	2
+	Adição	3
-	Subtração	3

Exemplo:

```
Algoritmo Calculo_Area_Quadrado
var lado, area :real;
Início
   Leia lado;
   area ← (lado * lado);
   Escreva "A área do quadrado é" + area;
Fim
```

▪ Operadores de Atribuição

- Têm como função retornar um valor atribuído de acordo com a operação indicada;
- A operação é feita entre os dois operandos, sendo atribuído o resultado ao primeiro.

Operadores de Atribuição		
Atribuição simples **=**	**Atribuição com subtração** **-=**	**Atribuição com divisão** **/=**
Atribuição com adição **+=**	**Atribuição com multiplicação** ***=**	**Atribuição com módulo** **%=**

Exemplo:

```
Algoritmo Calculo_Area_Circulo
var raio, area :real;
real PI = 3.14;
Início
   Leia raio;
   area ← (pi) * (raio)**2;
   Escreva "A área do círculo é" + area;
Fim
```

▪ Operadores Lógicos

- Fazem comparações com o objetivo de avaliar expressões em que o resultado pode ser **verdadeiro** ou **falso**, ou seja, implementando a lógica booleana;
- O retorno desta comparação é sempre um valor do tipo booleano (lógico).

Operadores Lógicos		
Conjunção **e/and/&&**	**Disjunção** **ou/or/||**	**Negação** **não/not**
As duas condições devem ser verdadeiras para que o resultado seja verdadeiro.	Pelo menos uma condição deve ser verdadeira para que o resultado seja verdadeiro.	Inverte o valor do resultado da condição.

- **Retorno das expressões:**

Retorno de cada expressão		**E**	**OU**
Expressão A	Expressão B	A e B	A ou B
F	F	F	F
F	V	F	V
V	F	F	V
V	V	V	V

Exemplo:

```
Algoritmo Verifica_Aluno_Aprovado
var nota, frequencia :real;
Início
  Leia nota, frequencia;
  if nota >=7 e frequencia >= 70%
    Escreva “Aprovado”;
  else
    Escreva “Reprovado”;
Fim
```

▪ Operadores Relacionais

- São utilizados para comparar valores entre variáveis e expressões do mesmo tipo;
- O retorno desta comparação é sempre um valor do tipo booleano (verdadeiro/falso).

Operadores Relacionais		
Igual **==**	**Maior** **>**	**Maior ou Igual** **>=**
Diferente **!= ou <>**	**Menor** **<**	**Menor ou Igual** **<=**

Exemplo:

```
Algoritmo Pode_Tirar_Carteira_de_Motorista
var idade :inteiro;
Início
  Leia idade;
  if idade >= 18
    Escreva “Pode tirar carteira de motorista.";
  else
    Escreva “Não pode tirar carteira de motorista.";
Fim
```

2.3.1 ATIVIDADES OPERADORES

1. **São tipos de operadores, <u>exceto</u>:**
   a) Aritméticos
   b) Lógicos
   c) Relacionais
   d) Interpretativos
   e) De atribuição

2. **A afirmação "É um conjunto de símbolos que representa as operações básicas da matemática, como somar e subtrair" se refere a que tipo de operador:**
   a) Lógico
   b) Relacional
   c) De atribuição
   d) Booleano
   e) Aritmético

3. **São exemplos de operadores de atribuição, <u>exceto</u>:**
   a) +=
   b) *=
   c) %=
   d) #=
   e) =

4. **Qual opção abaixo contém apenas tipos de operadores lógicos:**
   a) Conjunção, negação, afirmação
   b) Conjunção, afirmação, disjunção
   c) Conjunção, disjunção, negação
   d) Afirmação, disjunção, negação
   e) Conjunção, abdução, disjunção

5. São exemplos de operadores relacionais, <u>exceto</u>:

a) <>

b) !=

c) =

d) ==

Módulo 3 – Estruturas de Seleção

Conteúdos:

- Conceitos de Estruturas de Seleção
- Funcionamento das Estruturas de Seleção If/Else e Switch/Case

Ao final deste módulo, você será capaz de:

- Identificar a necessidade de utilizar Estruturas de Seleção
- Utilizar as Estruturas de Seleção If/Else e Switch/Case

Recursos:

- Vídeos
- Textos explicativos
- Atividades avaliativas

3.1 ESTRUTURAS DE SELEÇÃO

▪ Vídeo - Estruturas de Seleção

Vídeo 4- Conceitos iniciais sobre Estruturas de Seleção

Link: https://youtu.be/bob7VJo72Sw

▪ O que são?

Você sabe o que são Estruturas de Seleção, também conhecidas como Estruturas Condicionais?

- São comandos que auxiliam no direcionamento da sequência de execução de um programa por meio da avaliação de **condições lógicas**;
- Têm como função validar condições e comparar o resultado destas.

▪ Algoritmo condicional

- Permite a **escolha** de um grupo de **ações** a ser executado quando determinadas **condições**, representadas por expressões lógicas, são ou não satisfeitas.

▪ Para que servem?

- Permitem alterar o **Fluxo de Execução** do algoritmo, de forma a selecionar qual parte deve ser executada;
- Essa “decisão” de execução é tomada a partir de uma condição, que pode resultar apenas dois valores: **verdadeiro** ou **falso**;
- Uma condição é representada por **expressões relacionais** ou **lógicas**.

▪ Funcionamento

- Após executar as funções de validação e comparação, as estruturas de seleção irão executar os blocos de comando, definidos de acordo com o resultado da comparação (**verdadeiro** ou **falso**).

▪ Tipos de Estruturas de Seleção

- ✓ If/Else (Se/Então);
- ✓ Switch/Case (Escolha/Caso)

3.1.1 ATIVIDADES ESTRUTURAS DE SELEÇÃO

1. **Qual das alterativas abaixo contém <u>apenas</u> Estruturas de Seleção:**
   a) For; If/Else
   b) If/Else; Switch/Case
   c) While; Switch/Case
   d) While; If/Else
   e) Do/While; While

2. **A afirmação "Um algoritmo sequencial permite a escolha de um grupo de ações a ser executado quando determinadas condições, representadas por expressões lógicas, são ou não satisfeitas" é:**
   a) Verdadeira
   b) Falsa

3. **A afirmação "As estruturas de seleção permitem alterar o Fluxo de Execução do algoritmo, de forma a selecionar qual parte deve ser executada" é:**
   a) Verdadeira
   b) Falsa

3.2 ESTRUTURA DE SELEÇÃO IF/ELSE

▪ Classificação

- **Tipos de estruturas IF/ELSE:**
- Simples;
- Compostas;
- Aninhadas.

▪ Estruturas de Seleção Simples

Sintaxe no Algoritmo:

Se <comandos>

Então <instruções>;

FimSe

- **Como funciona?**
  - A condição é verificada a cada passagem pela estrutura **IF/SE**;
  - Se a condição for satisfeita (**verdadeira**), são executadas as instruções entre chaves (**então**);
  - Se a condição **NÃO** for satisfeita (**falso**), as instruções entre chaves não são executadas, sendo executado o código logo após as chaves;
  - O **IF/SE** sempre executará o bloco de comando ou instrução única se a condição entre parênteses retornar um resultado booleano **verdadeiro**. Caso contrário, o bloco de comando ou a instrução única não serão executadas.

Exemplo:

```
Algoritmo verifica_numero
Início
  var x, y :inteiro
  x ← 10
  y ← 20
  Se (x < y) Então
    Escreva "X é menor que Y.";
  FimSe
Fim
```

▪ Estruturas de Seleção Composta

Sintaxe no Algoritmo:

```
Se <condição> Então
{
  <instruções>;
}
Senão
{
  <instruções>
}
FimSe
```

- **Como funciona?**
  - A condição é verificada a cada passagem pela estrutura **IF/SE**;

- Se a condição for satisfeita (**verdadeira**), são executadas as instruções entre chaves do **IF/SE**;
- Se a condição **NÃO** for satisfeita (**falso**), são executadas as instruções dentro das chaves do **ELSE/SENÃO**;
- As instruções do ELSE/SENÃO serão executadas somente quando o valor da condição do IF/SE for falso.

Exemplo:

```
Algoritmo verifica_numero
Início
  var x, y :inteiro
  x ← 30
  y ← 20
  Se (x < y) Então
    Escreva "X é menor que Y.";
  Senão
    Escreva "X é maior que Y.";
  FimSe
Fim
```

▪ Estruturas de Seleção Aninhada

Sintaxe no Algoritmo:

```
Se <condição> Então
  Se <condição> Então
    <instruções>;
  FimSe
Senão
  <instruções>;
FimSe
```

- É utilizada, em geral, quando é necessário realizar várias comparações com a **mesma variável**;
- É chamada de aninhada porque na sua representação fica uma seleção **dentro** de outra seleção;
- Também é conhecida como seleção "**encadeada**";
- Permite fazer a escolha de apenas um entre vários comandos possíveis.

Exemplo:

```
Algoritmo novo_salario
Início
  var salario, novo_salario :real
  Se (salario < 500) Então
    novo_salario < -- salario 1.20;
  Senão
    Se (salario <= 1000) Então
      novo_salario ← salario 1.10;
    Senão
```

```
            novo_salario ← salario 1.05;
        FimSe
    FimSe
Fim
```

3.2.1 ATIVIDADES ESTRUTURA DE SELEÇÃO IF/ELSE

1. **A estrutura de seleção IF pode ser classificada em:**
   a) Simples; Composta; Refinada
   b) Simples; Composta; Derivada
   c) Simples, Composta; Aninhada
   d) Simples; Derivada; Aninhada
   e) Composta; Derivada; Aninhada

2. **A afirmação "Na estrutura de seleção If/Else a condição é verificada a cada passagem pela estrutura Else", é:**
   a) Verdadeira
   b) Falsa

3. **A estrutura de seleção aninhada também é conhecida como:**
   a) Estruturada
   b) Combinada
   c) Encadeada
   d) Distribuída
   e) Interpretada

3.3 ESTRUTURA DE SELEÇÃO SWITCH/CASE

▪ Para que serve?

- A estrutura **Switch/Case-Escolha/Caso** é utilizada quando é necessário testar a mesma variável com uma série de valores (**várias vezes**).

▪ Estrutura padrão

Sintaxe no Algoritmo:

Escolha <condição>
  Caso1: <expressão>
    <instruções>;
    Pare;
  Caso2: <expressão>
    <instruções>;
    Pare;
  Senão:
    <instruções>;
    Pare;
FimEscolha

▪ Como funciona?

- A variável a ser testada deve ser sempre do tipo **inteiro** ou **literal**;
- É utilizado para oferecer **várias opções** ao usuário, deixando que escolha um valor dentre vários;

- A principal **vantagem** desse comando é que ele evita uma série de testes com o comando **IF/SE**;
- Funciona de maneira semelhante ao **IF/SE** encadeado;
- A condição após o **SWITCH/ESCOLHA** informa o valor que será comparado em cada **CASE/CASO**;
- No primeiro **CASE/CASO** é verificado se o valor recebido como parâmetro é igual ao seu valor;
- Se o valor do parâmetro informado for o mesmo (igual) do **CASE/CASO**, será executado o trecho de código dentro do respectivo **CASE/CASO**;
- Se o valor do parâmetro informado for **diferente** do **CASE/CASO**, será testada a condição do **próximo CASE/CASO**;
- O comando **BREAK/PARE** é utilizado para forçar a saída do **SWITCH/ESCOLHA** ao se entrar em um **CASE/CASO**;
- Sem o **BREAK/PARE** todos os **CASE/CASO** serão testados, mesmo que algum **CASE/CASO** já tenha atendido a condição;
- O comando **DEFAULT/SENÃO** é **opcional** e define um **fluxo alternativo** para as situações não atendidas por nenhum **CASE/CASO**;
- O trecho de código dentro do **DEFAULT/SENÃO** será executado apenas quando o valor de nenhum **CASE/CASO** for igual ao valor do parâmetro informado.

Exemplo:

```
Algoritmo informa_sexo
Início
  var sexo :literal
  Escolha (sexo)
    Caso ("F"):
    Escreva "Sexo feminino";
    Pare;
  Caso ("M"):
    Escreva "Sexo masculino";
```

```
        Pare;
    FimEscolha
Fim
```

3.3.1 ATIVIDADES ESTRUTURA DE SELEÇÃO SWITCH/CASE

1. **A afirmação "A estrutura de seleção Switch/Case é utilizada quando é necessário testar a mesma variável várias vezes" é:**
   a) Verdadeira
   b) Falsa

2. **Na estrutura de seção Switch/Case a variável a ser testada deve ser sempre do tipo:**
   a) Inteiro ou Lógica
   b) Inteiro ou Real
   c) Inteiro ou Literal
   d) Lógica ou Literal
   e) Literal

3. **A afirmativa "O comando BREAK é utilizado para forçar a repetição do SWITCH ao se entrar em um CASE" é:**
   a) Verdadeira
   b) Falsa

Módulo 4 – Estruturas de Repetição

Conteúdos:

- Conceitos de Estruturas de Repetição
- Funcionamento das Estruturas de Repetição *For*, *While* e *Do/While*

Ao final deste módulo, você será capaz de:

- Identificar a necessidade de utilizar Estruturas de Repetição
- Utilizar as Estruturas de Repetição *For*, *While* e *Do/While*

Recursos:

- Vídeos
- Textos explicativos
- Atividades avaliativas

4.1 ESTRUTURAS DE REPETIÇÃO

▪ Vídeo - Estruturas de Repetição

Vídeo 5- Conceitos iniciais sobre Estruturas de Repetição

Link: https://youtu.be/_9YW3Yuxr7M

▪ O que são?

- São comandos que permitem que uma **sequência de instruções** seja executada **várias vezes** até que uma condição seja satisfeita;
- Se uma instrução ou uma sequência de instruções precisa ser **executada várias vezes**, deve-se utilizar uma **estrutura de repetição**.

▪ Para que servem?

- Servem para **repetir** um **conjunto de instruções** sem que seja necessário escrevê-las várias vezes;
- Permitem que um trecho do algoritmo seja repetido, em um número determinado ou indeterminado de vezes, sem que o código a ser repetido tenha que ser escrito novamente;
- As estruturas de repetição também são chamadas de **Laços** ou ***Loops***.

▪ Funcionamento

- As estruturas de repetição envolvem a avaliação de uma condição (**teste**);
- A avaliação resulta em valores **Verdadeiros** ou **Falsos**;
- Se o resultado da condição é **Falso, não é iniciada a repetição** ou, caso esteja em execução, é **encerrada a repetição**;
- Se o resultado da condição for **Verdadeiro**, é **iniciada** a repetição ou, caso esteja em execução, é **reiniciada** a execução das instruções dentro da Estrutura de Repetição;
- A **avaliação da condição** é sempre novamente realizada após a execução da **última instrução** dentro da estrutura de repetição;
- A única Estrutura de Repetição que **não** realiza a **avaliação da condição antes de iniciar** é a **Do/While (Faça/Enquanto)**.
- Desta forma, é assegurado que todas as instruções dentro da Estrutura de Repetição do Do/While serão **executadas pelo menos uma vez**.

▪ Tipos de Estruturas de Repetição

- For (Para/Faça);
- While (Enquanto/Faça);
- Do/While (Faça/Enquanto).

4.1.1 ATIVIDADES ESTRUTURAS DE REPETIÇÃO

1. **A afirmação "São comandos que permitem que uma sequência de instruções seja executada várias vezes enquanto uma condição é satisfeita (verdadeira)" se refere a:**
   a) Estruturas de Seleção
   b) Estruturas de Repetição
   c) Estruturas de Inicialização
   d) Estruturas de Condição
   e) Estruturas de Comparação

2. **Dentre as opções abaixo, qual contém <u>apenas</u> Estruturas de Repetição?**
   a) For; While; If
   b) For; Do/While; Switch/Case
   c) For; While; Do/While
   d) While; If/Else; Do/While
   e) While; If/Else; Switch/Case

3. **A afirmação "As estruturas de repetição envolvem a avaliação de uma condição (teste) que resulta em valores Verdadeiros ou Falsos" é:**
   a) Verdadeira
   b) Falsa

4. **As Estruturas de Repetição também são chamadas de:**
   a) Voltas/Enlaces
   b) Laços/Enlaces
   c) Reenvio/Loops
   d) Laços/Loops
   e) Recorrência/Loops

5. **A única Estrutura de Repetição que não realiza a avaliação da condição antes de iniciar é:**

a) Do/While
b) While
c) If/Else
d) For
e) Switch/Case

4.2 ESTRUTURA DE REPETIÇÃO FOR

▪ Características

- Deve ser usada quando o **número exato** de repetições é conhecido;
- Utiliza uma **variável de controle** que deve ser do tipo **Inteiro** ou **Literal**.

▪ Estrutura padrão

Sintaxe no Algoritmo:

Para <valor> **Até** <condição> **Faça**

{

<instruções>

}

FimPara

Obs.: Ao invés de **incremento** pode ser feito um **decremento** do valor da **variável de inicialização**.

▪ Como funciona?

- **For:** comando que inicializa a estrutura de repetição. Sua **condição** é **testada antes** de executar qualquer **instrução** dentro do laço;
- **Variável de inicialização:** comando de atribuição que **inicia** uma **variável de controle** do laço. É executada **apenas uma vez**, no início do laço;
- **Condição:** **determina** o **final** do laço (repetição). Normalmente é uma **expressão lógica**. É verificada **antes da execução** do laço. Se for

Verdadeira, as instruções dentro do laço são **executadas**. Se for **Falsa** o laço é **finalizado**;

- **Incremento/decremento:** é executado sempre no final do laço, **mudando** o valor da **variável de controle** a cada repetição do laço.

Exemplo:

```
Algoritmo Imprimir_numeros_de_1_a_100
Início
  var contador :inteiro;
  Para contador ← 1 Até 100 Faça
    Escreva (contador);
  FimPara
Fim
```

4.2.1 ATIVIDADES ESTRUTURA DE REPETIÇÃO FOR

1. **A estrutura de repetição For contém por padrão:**
   a) Variável de inicialização; Condição; Incremento/decremento
   b) Variável de finalização; Condição; Incremento/decremento
   c) Variável de inicialização; Operador; Incremento/decremento
   d) Variável de inicialização; Condição; Operador
   e) Variável de finalização; Condição; Operador

2. **Na estrutura de repetição For a condição é testada:**
   a) Após executar uma vez as instruções dentro do laço
   b) Antes de executar qualquer instrução dentro do laço
   c) Depois de executar as instruções do laço
   d) Durante a execução das instruções do laço
   e) Apenas uma vez no final da primeira execução do laço

3. **A afirmação "A estrutura de repetição For deve ser usada quando o número exato de repetições não é conhecido" é:**
   a) Verdadeira
   b) Falsa

4. **A afirmação "Na Estrutura de Repetição For, ao invés de incremento, pode ser feito um decremento do valor da Variável de Inicialização" é:**
   a) Verdadeira
   b) Falsa

5. **O comando que inicia uma variável de controle do laço na Estrutura de Repetição For é denominado:**
   a) Variável de finalização
   b) Variável de laço
   c) Variável de inicialização
   d) Variável de repetição
   e) Variável de conclusão

4.3 ESTRUTURA DE REPETIÇÃO WHILE

▪ Características

- É a estrutura de repetição **mais simples**;
- É ideal para situações em que **não** se sabe o **número exato** de vezes em que o bloco de instruções deve ser **repetido**;
- Pode ser utilizado para **substituir** laços **FOR**.

▪ Estrutura padrão

Sintaxe no Algoritmo:

Enquanto <condição> **Faça**

{

<instruções>

}

FimEnquanto

▪ Como funciona?

- A **condição** é validada antes de cada repetição do laço;
- Enquanto a condição for **Verdadeira**, o bloco de instruções dentro do laço é **executado**;
- Quando a condição se torna **Falsa**, o laço é **finalizado**.

Exemplo:

```
Algoritmo Imprimir_numeros_de_1_a_10
Início
  var contador :inteiro;
  contador← 1
  Enquanto (contador < 10) Faça
    Escreva contador;
    contador ← contador +1
  FimEnquanto
Fim
```

4.3.1 ATIVIDADES ESTRUTURA DE REPETIÇÃO WHILE

1. **A afirmação "A estrutura de repetição While deve ser usada quando o número exato de repetições não é conhecido" é:**
   a) Verdadeira
   b) Falsa

2. **A afirmação "Na estrutura de repetição While, a condição é validada antes de cada repetição do laço" é:**
   a) Verdadeira
   b) Falsa

3. **É a estrutura de repetição mais simples?**
   a) For
   b) If
   c) Do/While
   d) Switch/Case
   e) While

4.4 ESTRUTURA DE REPETIÇÃO DO/WHILE

▪ Características

- Testa a **condição de validação** do laço apenas no **final** do comando. Desta forma, é assegurado que as **instruções** dentro do laço serão **executadas** pelo menos **uma vez**;
- A diferença para a estrutura **WHILE** é que na **DO/WHILE** a **condição de validação** é **verificada após** a **execução** do bloco de instruções do laço.

▪ Estrutura padrão

Sintaxe no Algoritmo:

Faça

{

<instruções>

} **Enquanto** <condição>;

▪ Como funciona?

- Na **primeira vez** que o laço for executado todas as **instruções** dentro deste serão **executadas**, **independente** da **condição** estabelecida;
- Somente após a **primeira execução** das instruções do laço é que a **expressão** será **testada**;
- Depois da primeira execução, as instruções dentro do laço só são **executadas novamente** se a **condição de validação** for **Verdadeira**.

Exemplo:

```
Algoritmo Imprimir_numeros_de_1_a_10
Início
  var contador :inteiro;
  contador ← 1
  Faz
    Escreva contador;
    contador ← contador +1
  Enquanto (contador < 10);
Fim
```

▪ Vídeo - Revisão dos conteúdos do curso

Vídeo 6- Revisão dos conteúdos do curso

Link: https://youtu.be/IMDWp1qlkZA

4.4.1 ATIVIDADES ESTRUTURA DE REPETIÇÃO DO/WHILE

1. **A afirmação "A principal diferença para a estrutura WHILE, é que na DO/WHILE a condição de validação é verificada após a execução do bloco de instruções do laço" é:**
   a) Verdadeira
   b) Falsa

2. **Na estrutura de repetição Do/While a condição é testada:**
   a) Após executar uma vez as instruções dentro do laço
   b) Antes de executar qualquer instrução dentro do laço
   c) Durante a execução das instruções do laço
   d) Apenas uma vez no final da primeira execução do laço

3. **A afirmativa "Na estrutura de repetição Do/While é assegurada que as instruções dentro do laço serão executadas pelo menos duas vezes" é:**
   a) Verdadeira
   b) Falsa

4. **A afirmação "Na Estrutura de Repetição Do/While, na primeira vez que o laço é executado, todas as instruções dentro deste serão executadas, independente da condição estabelecida" é:**
   a) Verdadeira
   b) Falsa

Questionário Final

1. **O retorno de uma comparação realizada por Operadores Lógicos é um valor do tipo:**
   a) Literal
   b) Booleano
   c) Positivo
   d) Neutro
   e) Nulo

2. **A partir da análise do algoritmo abaixo, informe os valores que serão exibidos (Escreva):**

```
Início

        var x,y :inteiro

        x← 5
        y ← 10

        Escreva x, y

        x ← 15
        y ← x

        Escreva x, y
Fim
```

   a) 5, 10, 5, 10
   b) 5, 10, 15, 10
   c) 15, 10, 15, 10
   d) 5, 10, 15, 15
   e) 15, 10, 15, 15

3. **Dado que A=5 e B=10, análise as expressões abaixo e informe o resultado correto para cada uma delas. Atenção aos Operadores Lógicos "E" e "OU".**

A<10 E B > 15
A = 5 E B > 5
A = 5 OU B < 5

a) Verdadeiro, Falso, Verdadeiro
b) Falso, Verdadeiro, Falso
c) Falso, Verdadeiro, Verdadeiro
d) Verdadeiro, Verdadeiro, Falso
e) Falso, Falso, Verdadeiro

4. **O tipo de dado Numérico pode ser dividido em: inteiro, real e fracionário. Esta afirmação é:**

a) Verdadeira
b) Falsa

5. **A afirmação "É uma boa prática incluir comentários nas linhas mais importantes do Algoritmo para facilitar sua leitura posteriormente por outros programadores ou pelo próprio programador" é:**

a) Verdadeira
b) Falsa

6. **Os Operadores Aritméticos podem ser utilizados apenas com variáveis do tipo:**

a) Numéricas
b) Literais e Numéricas
c) Numéricas e lógicas
d) Literais e lógicas
e) Numéricas, lógicas e literais

7. **A partir da análise do algoritmo abaixo, informe os valores que serão exibidos/impressos (Escreva):**

```
Início

        var: a,b,c :inteiro

        a ← 50
        b ← 20
        c ← A + B
        a ← 10

        Escreva a, b, c
Fim
```

a) 50, 20, 70
b) 10, 20, 70
c) 10, 20, 30
d) 50, 20, 0
e) 10, 20, 20

8. **Qual dos comandos abaixo pertence a Estrutura de Seleção Switch/Case e é de uso opcional:**

a) Switch
b) Case
c) If
d) While
e) Default

9. **A única Estrutura de Repetição que não realiza a avaliação da condição (teste) antes de iniciar é:**

a) Do/While
b) While
c) If/Else
d) For
e) Switch/Case

10. A partir da análise do algoritmo abaixo, informe o(s) valor(es) que será(ão) exibido(s)/impresso(s):

```
Início
        var: numero1, numero2, numero3 :inteiro

        numero1 ← 25
        numero2 ← 15
        numero3 ← 30

        Se numero1 > numero2 Então
                Escreva numero1
        Senão
                Escreva numero2

Fim
```

a) 25
b) 15
c) 25, 15, 30
d) 15, 25
e) 25, 15

11. A afirmação "Na estrutura de repetição <u>For</u>, ao invés de <u>incremento</u> pode ser feito um <u>decremento</u> do valor da variável de inicialização" é:

a) Verdadeira

b) Falsa

12. Informe o que será exibido/impresso (Escreva), a partir do algoritmo a seguir:

```
Início

    var numero :inteiro
    numero ←10

    Faça
            Escreva numero
            numero = numero +2
    Enquanto (numero < 20)
Fim
```

a) Nenhum número será impresso
b) Serão impressos os números 10, 12, 14, 16 e 18
c) Será impresso apenas o número 10
d) Serão impressos os números 12, 14, 16, 18 e 20
e) Serão impressos os números 10, 12, 14, 16, 18 e 20

Gabaritos

Módulo 1 - Algoritmos

Atividades Algoritmos:

1. d)
2. b)
3. a)

Atividades Formas de Representação

1. a)
2. b)
3. a)

Atividades Tipos de Dados

1. b)
2. b)
3. d)

Módulo 2 - Variáveis, Constantes e Operadores

Atividades constantes e Variáveis

1. b)
2. c)
3. a)

Atividades Entrada e Saída de Dados

1. c)
2. e)
3. a)

Atividades Operadores

1. d)
2. e)
3. d)
4. c)
5. c)

Módulo 3 - Estruturas de Seleção

Atividades Estruturas de Seleção

1. b)
2. b)
3. a)

Atividades Estrutura de Seleção IF/ELSE

1. c)
2. b)
3. c)

Atividades Estrutura de Seleção SWITCH/CASE

1. a)
2. c)
3. b)

Módulo 4 - Estruturas de Repetição

Atividades Estruturas de Repetição

1. b)
2. c)
3. a)
4. d)
5. a)

Atividades Estrutura de Repetição FOR

1. a)
2. b)
3. b)
4. a)
5. c)

Atividades Estrutura de Repetição WHILE

1. a)
2. a)
3. e)

Atividades Estrutura de Repetição DO/WHILE

1. a)
2. a)
3. b)
4. a)

Quiz Final

1. b)
2. d)
3. c)
4. b)
5. a)
6. a)
7. b)
8. e)
9. a)
10. a)
11. a)
12. b)