▪ 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:

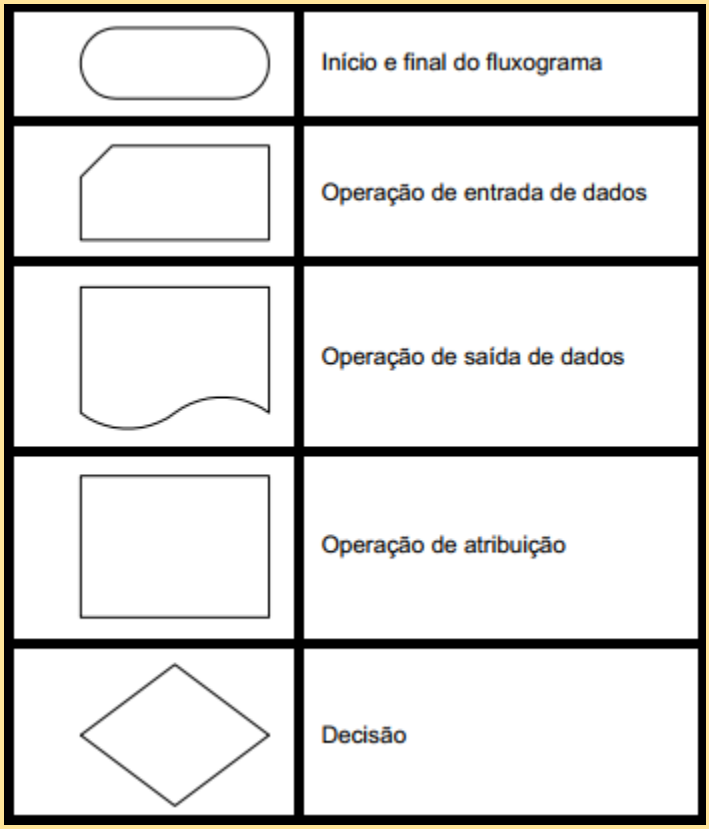
- Obter as suas 2 notas de provas;
- Calcular a média aritmética;
- Se a média for maior ou igual a 7, o aluno foi aprovado;
- Senão o aluno foi reprovado.

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.

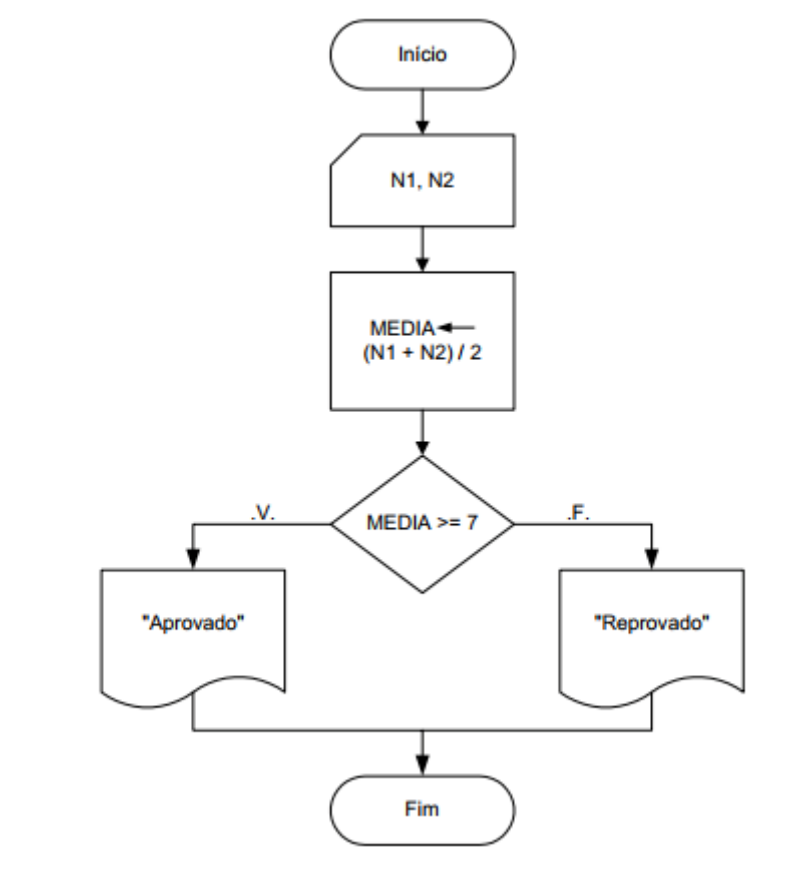
Elementos gráficos



- 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

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) ==

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