

# **ALGORITMOS & LÓGICA DE PROGRAMAÇÃO**

**Uso de estruturas simples**

**Mogi das Cruzes, SP**

**2025**

**CRÉDITOS**

Estas notas de aula foram elaboradas com base nas referências citadas no final e também considerando algumas contribuições dos professores listados a seguir: Marcia Bissaco, Cibele Mattos, Andréa Ono Sakai, Marco Antônio Porto Alvarenga, Marco H. Sawada (in memorium), Roberta Panzera, Sebastião Varotto, Valcir Orlando e Viviane Guimarães Ribeiro.

# SUMÁRIO

**falta fazer sumário e conferir os códigos**

# 1. CONCEITOS BÁSICOS

*“Programar é basicamente construir algoritmos”.*

## 1.1. LÓGICA DE PROGRAMAÇÃO E ALGORITMOS

A resolução de problemas envolve a identificação de uma ou mais soluções viáveis, por meio da aplicação de raciocínios lógicos e coerentes. Isso permite alcançar soluções eficazes e eficientes. Por exemplo:

- O armário está fechado.
- O prato está dentro do armário.
- Portanto, primeiro é preciso abrir o armário para depois pegar o prato.

Quando o problema a ser solucionado é de processamento de dados (computador), é necessário usar a lógica de maneira formal, ou seja, adequar os raciocínios a simbolização usada na programação, de modo a obter o resultado do que deseja programar com qualidade. Neste caso, usamos a expressão lógica de programação, que é representada por algoritmos.

Quando se trata de resolver problemas de processamento de dados em computadores, é essencial aplicar a lógica de forma sistemática e rigorosa. Isso envolve traduzir os raciocínios em uma linguagem simbólica compatível com a programação, garantindo resultados precisos e de qualidade. Nesse contexto, a expressão lógica é representada por algoritmos, que são fundamentais para o desenvolvimento de programas eficazes.

Os algoritmos estão presentes em nosso dia-a-dia, influenciando nossas ações e decisões. Exemplos disso incluem:

- Orientações para chegar a um local específico;
- Receitas culinárias que seguem etapas precisas;
- Plantas de construção que guiam a execução de projetos;
- Instruções para o uso correto de medicamentos e equipamentos;
- Manuais de montagem de aparelhos,
- Rotinas da vida diária.

Esses exemplos ilustram como os algoritmos são utilizados para fornecer sequências lógicas de ações e garantir resultados precisos, sem ambiguidade. Algoritmos são, portanto, uma sequência ordenada, finita e sem ambiguidade de passos executáveis (ou seja, instruções/ações), precisamente definidos, que manipulam um conjunto de dados a fim de resolver um determinado problema.

A seguir são descritas as formas de representação dos algoritmos.

## 1.2. FORMAS DE REPRESENTAÇÃO DE ALGORITMOS

Um algoritmo deve possuir ações bem definidas de modo a não dar margem à dupla interpretação. Ele parte de um ponto inicial, passando pelas ações até alcançar o resultado final. Não importando a representação a ser escolhida, para solucionar o problema devem ser empregadas corretamente as leis do pensamento para obter o resultado esperado.

Um algoritmo pode ser representado através de descrição narrativa, fluxograma, pseudocódigo e diagrama de Chaplin, formas de representação descritas nos itens seguintes.

### 1.2. 1. Descrição Narrativa

Nesta forma de representação, os algoritmos são expressos diretamente em Linguagem Natural. Por exemplo, observe a solução dos problemas apresentados abaixo referente à troca de uma lâmpada velha e ao cálculo da média aritmética de duas notas.

ALG01 – Trocar uma lâmpada velha
Pegar uma escada; Posicionar a escada embaixo da lâmpada; Buscar uma lâmpada nova; Subir na escada; Retirar a lâmpada velha; Colocar a lâmpada nova; Descer a escada; Guardar a escada; Descartar a lâmpada.

ALG02 – Calcular a média aritmética de um aluno
Pegar as notas N1 e N2; Calcular a média aritmética ( $M \leftarrow (N1 + N2)/2$ ); Mostrar o resultado (M);

A Linguagem Natural muitas vezes é PROLIXA (cansativa, longa, irritante) e IMPRECISA, podendo ocasionar problemas, tais como:

- **Má interpretação;**
- **Perda de informação;**
- **Dificuldade de se transmitir a informação desejada.**

Por exemplo, descreva um caminho para sair de onde você está no momento e ir até a estação rodoviária de São Paulo usando o metrô.

### 1.2. 2. Pseudocódigo

Nesta forma de representação, os algoritmos são expressos numa linguagem simplificada, também conhecida por Portugol ou português estruturado.

No momento do desenvolvimento de um algoritmo, é preciso saber quais dados serão recebidos do mundo externo e também quais informações exteriorizar. Não interessa, portanto, saber se os dados entrarão via teclado, pela leitura de um arquivo de dados ou por qualquer outro meio. Para dar entrada ou saída nesses dados, são utilizados os verbos LER, ESCREVER e IMPRIMIR. O comando ler (ou leia) espera receber um determinado dado (sem importar a origem). O comando escrever (ou escreva) mostra a informação produzida no vídeo. O comando imprimir (ou imprima) faz a impressão em papel da informação produzida. A sintaxe dos comandos é :

leia (variável1, variável2, ... , variável n);

escreva (lista de constantes, variáveis e/ou expressões);

imprima (lista de constantes, variáveis e/ou expressões);

### **Exemplos:**

Leia (numero1);

```
numero2 ← numero1 * 2;  
escreva ('O dobro do número é ', numero2);  
imprima_('O triplo do número é ', numero1 * 3);
```

Onde: ← equivale ao símbolo = (operador de atribuição)

A seguir são apresentados os algoritmos ALG01 e ALG02, que ilustram uma solução para a troca de lâmpada velha e para o cálculo da média aritmética de um aluno, respectivamente.

ALG01 – Pseudocódigo do algoritmo para trocar uma lâmpada velha
<pre>Algoritmo Troca_lampada; Início   Escreva ('Pegar uma escada');   Escreva ('Posicionar a escada embaixo da lâmpada');   Escreva ('Buscar uma lâmpada nova');   Escreva ('Subir na escada');   Escreva ('Retirar a lâmpada velha');   Escreva ('Colocar a lâmpada nova');   Escreva ('Descer da escada');   Escreva ('Guardar a escada') Fim</pre>

ALG02 – Pseudocódigo do algoritmo que calcula a média aritmética de duas notas
<pre>Algoritmo CalcMedia; Var   N1, N2, M : Numérico; Início   //entrada de dados   Leia (N1, N2);   // processamento   M ← (N1 + N2)/2;   // saída de dados   Escreva (M); Fim.</pre>

Note que ambos os algoritmos começam e terminam com as palavras **Algoritmo**, **início** e **fim**, sendo que a palavra algoritmo é seguida de um identificador, que corresponde a um nome representativo do problema a ser solucionado. Em ALG02 tem

também as palavras chaves **Var** e **Numérico**, que são utilizadas para declaração de variáveis e seus tipos de dados que fazem parte da solução do problema. Estas serão esclarecidas mais a frente. Por enquanto, é importante saber que estas palavras chaves compõem a sintaxe do pseudocódigo.

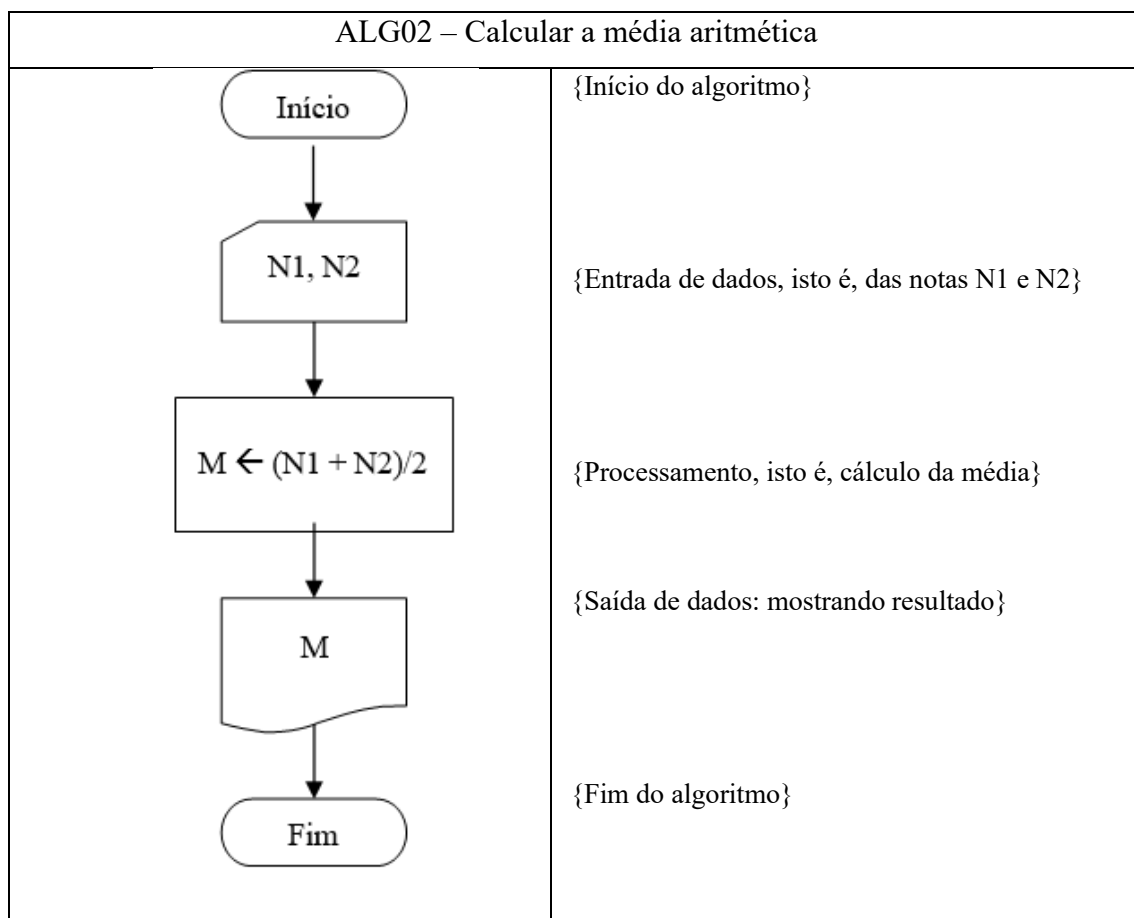
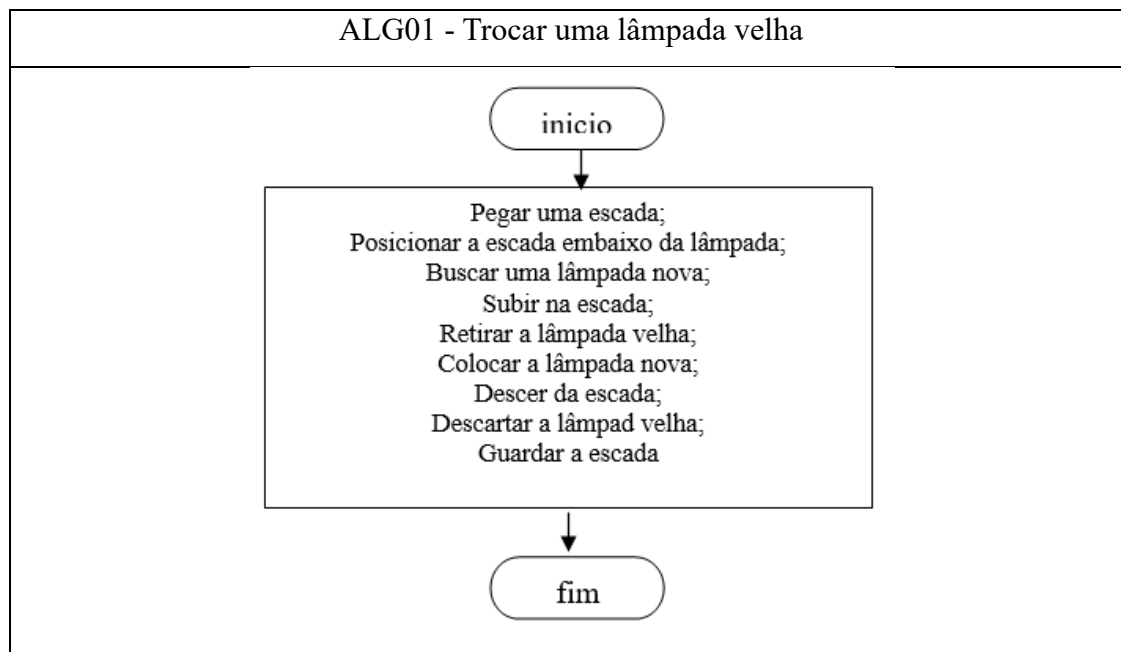
### **1.2. 3. Fluxograma**

Nesta forma de representação, os algoritmos são expressos numa linguagem gráfica, sendo que algumas das formas geométricas que simbolizam algumas das ações mais realizadas são estão representadas no quadro seguinte.



Descrição	Representação gráfica	Descrição	Representação gráfica
Início/Fim		Processamento	
Entrada de dados	ou	Decisão	 Com somente uma seta na direção do losango e duas saindo dele com indicação de falso (F)/verdadeiro (V)
Saída de dados	ou ou	Repetição	 Com duas setas na direção do losango e duas saindo dele com indicação de falso (F)/verdadeiro (V) Ou
Conectores	ou	Direção do fluxo	

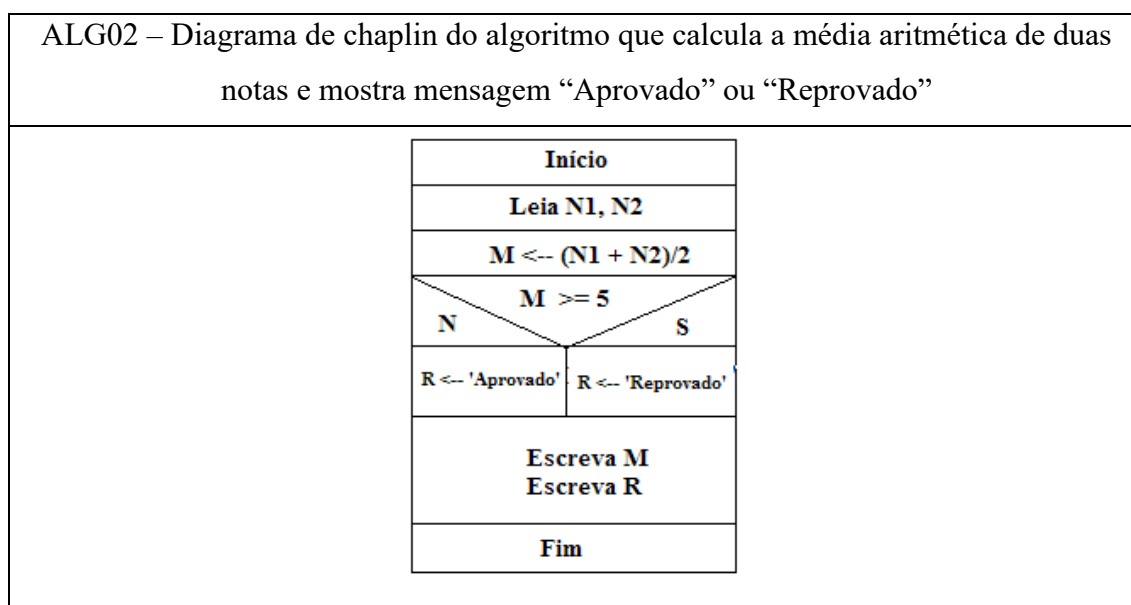
Veja a solução dos problemas da troca da lâmpada velha e do cálculo da média utilizando o fluxograma e mostrada a seguir.



O fluxograma é também conhecido pelos profissionais da área de Tecnologia da Informação de diagrama de blocos, embora tenham significados diferentes.

#### 1.2. 4. Diagrama de Chaplin

Nesta forma de representação, os algoritmos são expressos num diagrama de quadros, que apresenta uma visão hierárquica e estruturada da lógica do programa. Tem um ponto de entrada e um ponto de saída e é composto pelas estruturas básicas: sequência, seleção e repetição. Veja o exemplo a seguir.



#### 1.2. 5. Transformação do algoritmo em um programa

Quando pronunciamos o verbo **programar**, imediatamente vem à mente o termo **programa (software)**, que na realidade consiste em um algoritmo traduzido para uma forma inteligível para o computador, ou seja, a linguagem de máquina.

Como a linguagem binária é constituída por zeros e uns, sendo difícil transcrever o algoritmo para essa linguagem diretamente. Quem realiza essa tradução é um outro programa denominado compilador/interpretador, que converte em linguagem de máquina aquele código-fonte escrito pelo programador usando-se uma dada linguagem de programação. São exemplos de linguagem de programação: Pascal, Delphi, Visual Basic, C, etc.

A função básica da Linguagem de Programação é transcrever o algoritmo/lógica para um código-fonte cuja sintaxe é mais compreensível pelo programador.

Assim sendo, programa é uma combinação de algoritmo mais linguagem de programação, sendo que para elaborar o algoritmo é necessário pensar de forma lógica para solucionar o problema real que se apresenta (Figura 1).

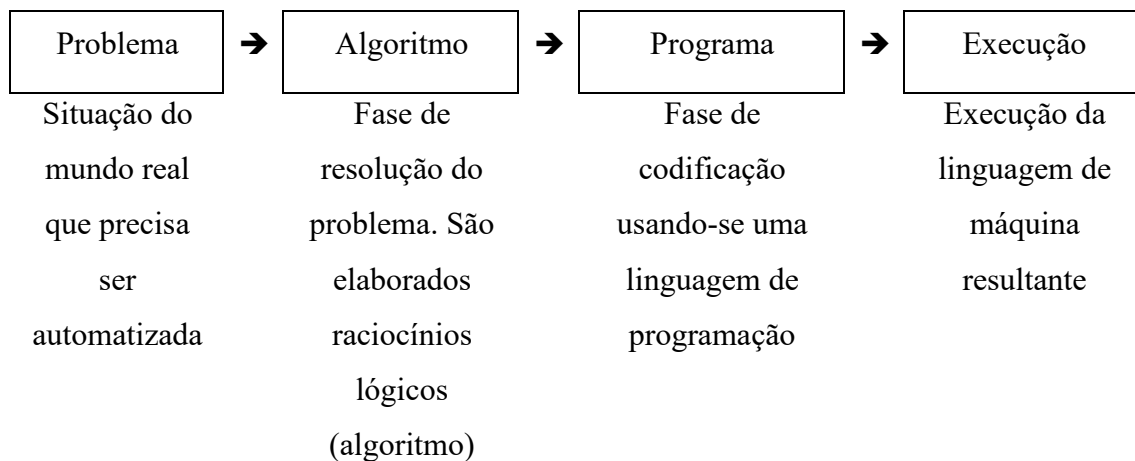


Figura 1 - Esquema algoritmo-programa

Os programadores devem iniciar o desenvolvimento de um programa pensando inicialmente na linha de raciocínio lógico para resolução do problema e, somente depois, gerar a documentação, ou seja, elaborar o fluxograma ou o pseudocódigo a fim de representar a sequência de operações a serem realizadas pelo computador.

A codificação do algoritmo pode ser realizada em qualquer Linguagem de Programação. Lembrando que estas apresentam sintaxes diferentes, mas são similares em muitas outras particularidades como, por exemplo, o paradigma de programação empregado (programação estruturada, programação orientada a eventos, programação orientada a objetos etc.). A seguir são apresentadas as implementações em PASCAL, C e C# dos algoritmos ALG01 e ALG02, mencionado o que é exibido no monitor quando cada uma das linhas de código é executada pelo computador.

Os programas geralmente têm uma estrutura padrão que consiste em: Entrada Processamento Saída. Os dados necessários à resolução do problema podem ser entrados via teclado e armazenados na memória do computador em posições denominadas de variável. O processamento é realizado pela Unidade Lógica e Aritmética (ULA) e a saída de dados processados (informações/valores calculados) ocorre através do monitor.

Para desenvolver um prlgrama com entrada, processamento e saída, é importante responder as seguintes perguntas durante a elaboração do algoritmo: O que tenho? O que quero (**objetivo**)? Como fazer para alcançar meu objetivo? Veja os exemplos apresentados a seguir, ou seja, ALG01 e ALG02.

ALG01 – Problema da troca da lâmpada	
Codificação em <b>PASCAL</b>	Aparece no monitor após execução do programa
<pre> Program Troca_lampada; Uses WinCrt; Begin   writeln ('Pegar uma escada;');   writeln ('Posicionar a escada embaixo da lâmpada; ');   writeln ('Buscar uma lâmpada nova;');   writeln ('Subir na escada;');   writeln ('Retirar a lâmpada velha;');   writeln ('Colocar a lâmpada nova. ');   writeln ('Descer da escada. ');   writeln ('Descartar a lâmpada velha');   writeln ('Guardar a escada. '); End. </pre>	<p>Nada</p> <p>Nada</p> <p>Nada</p> <p>Pegar uma escada;</p> <p>Posicionar a escada embaixo da lâmpada;</p> <p>Buscar uma lâmpada nova;</p> <p>Subir na escada;</p> <p>Retirar a lâmpada velha;</p> <p>Colocar a lâmpada nova.</p> <p>Descer da escada</p> <p>Descartar a lâmpada velha</p> <p>Guardar a escada</p> <p>Nada</p>
Codificação em linguagem <b>C</b>	Aparece no monitor após execução do programa
<pre> #include &lt;stdio.h&gt; main () {   puts ("Pegar uma escada;");   puts ("Posicionar a escada embaixo da lâmpada;");   puts ("Buscar uma lâmpada nova;");   puts ("Subir na escada;");   puts ("Retirar a lâmpada velha;");   puts ("Colocar a lâmpada nova;");   puts ('Descer da escada. ');   puts ('Descartar a lâmpada velha');   puts ('Guardar a escada. '); } </pre>	<p>Nada</p> <p>Nada</p> <p>Nada</p> <p>Pegar uma escada;</p> <p>Posicionar a escada embaixo da lâmpada;</p> <p>Buscar uma lâmpada nova;</p> <p>Subir na escada;</p> <p>Retirar a lâmpada velha;</p> <p>Colocar a lâmpada nova.</p> <p>Descer da escada</p> <p>Descartar a lâmpada velha</p> <p>Guardar a escada</p> <p>Nada</p>
Codificação em linguagem <b>C# (Console)</b>	Aparece no monitor após execução do programa
<pre> using System; namespace TrocaLampada {   class Program   {     static void Main(string[] args)     {       Console.WriteLine("Pegar uma escada; ");       Console.WriteLine("Posicionar a escada embaixo da lâmpada; ");       Console.WriteLine("Buscar uma lâmpada nova; ");       Console.WriteLine("Subir na escada; ");       Console.WriteLine("Retirar a lâmpada velha; ");       Console.WriteLine("Colocar a lâmpada nova; ");       Console.WriteLine('Descer da escada. ');       Console.WriteLine('Descartar a lâmpada velha');       Console.WriteLine('Guardar a escada. ');     }   } } </pre>	<p>Nada</p> <p>Nada</p> <p>Nada</p> <p>Nada</p> <p>Nada</p> <p>Nada</p> <p>Nada</p> <p>Pegar uma escada;</p> <p>Posicionar a escada embaixo da lâmpada;</p> <p>Buscar uma lâmpada nova;</p> <p>Subir na escada;</p> <p>Retirar a lâmpada velha;</p> <p>Colocar a lâmpada nova.</p> <p>Descer da escada</p> <p>Descartar a lâmpada velha</p> <p>Guardar a escada</p> <p>Nada</p> <p>Nada</p> <p>Nada</p>
Codificação em <b>PYTHON</b>	Aparece no monitor após execução do programa

<pre> print ("Pegar uma escada;") print ("Posicionar a escada embaixo da lâmpada;") print ("Buscar uma lâmpada nova;") print ("Subir na escada;") print ("Retirar a lâmpada velha;") print ("Colocar a lâmpada nova;") print ('Descer da escada.');</pre>	<pre> Pegar uma escada; Posicionar a escada embaixo da lâmpada; Buscar uma lâmpada nova; Subir na escada; Retirar a lâmpada velha; Colocar a lâmpada nova. Descer da escada Descartar a lâmpada velha Guardar a escada</pre>
---	--

ALG02 - Cálculo da média aritmética de 2 notas	
Codificação em PASCAL	Aparece no monitor após execução do programa
<pre> Program CalcMedia; Uses WinCrt; Var  N1, N2, M : real; Begin   Writeln ('Digite Nota 1:');   Readln (N1);   Writeln ('Digite Nota 2:');   Readln (N2);   M := (N1 + N2)/2;   Writeln ('A média é = ', M:5:2); End.</pre>	<pre> Nada Nada Nada Nada Nada Digite Nota 1: &lt;Deve ser digitado algum valor pelo teclado&gt; Digite Nota 2 &lt;Deve ser digitado algum valor pelo teclado&gt; Nada Nada A média é = &lt;valor armazenado em M&gt; Nada</pre>
Codificação em linguagem C	Aparece no monitor após execução do programa
<pre> #include &lt;stdio.h&gt; main () {   float N1, N2, M;   puts ("Digite Nota 1:");   scanf ("%f", &amp;N1);   puts ("Digite Nota 2: ");   scanf ("%f", &amp;N2);   M = (N1 + N2)/2;   printf ("A média é = %.2f", M); }.</pre>	<pre> Nada Nada Nada Nada Digite Nota 1: &lt;Deve ser digitado algum valor pelo teclado&gt; Digite Nota 2 &lt;Deve ser digitado algum valor pelo teclado&gt; Nada A média é = &lt;valor armazenado em M com duas casas decimais&gt; Nada</pre>

Codificação em linguagem C# (console)	Aparece no monitor após execução do programa
<pre> using System; namespace CalcMedia {   class Program   {     static void Main(string[] args)     {       float N1, N2, M;       Console.WriteLine("Digite Nota 1: ");</pre>	<pre> Nada Nada Nada Nada Nada Nada Nada Nada Nada Digite Nota 1: &lt;Deve ser digitado algum valor pelo teclado&gt;</pre>

<pre> N1 = float.Parse(Console.ReadLine()); Console.WriteLine("Digite Nota 2: "); N2 = float.Parse(Console.ReadLine()); M = (N1 + N2)/2 Console.WriteLine("A média é = {0} ", M);      } } </pre>	<p>Digite Nota 2 &lt;Deve ser digitado algum valor pelo teclado&gt; Nada A média é = &lt;valor armazenado em M&gt; Nada Nada Nada</p>
<b>Codificação em PYTHON (terminal)</b>	<b>Aparece no monitor após execução do programa</b>
<pre> N1 = float(input("Digite Nota 1: ")) N2 = float(input("Digite Nota 2: ")) M = (N1 + N2)/2 print(f"A média é {M:.2f}); </pre>	<p>Digite Nota 1:&lt;Deve ser digitado algum valor pelo teclado&gt;   Digite Nota 2 &lt;Deve ser digitado algum valor pelo teclado&gt;   Nada é mostrado na tela, mas ocorre o processamento na ULA (Unidade lógica e aritmética do computador, ou seja, o valor armazenada na posição de memória rotulada dde N1 é somado ao valoar armazenado em N2, é realizada a divição por 2 e o resulyado é colcoado na posição de memória rotulada de M  A média é &lt;mais o valor armazenado em M&gt;</p>

## 1.3. EXERCÍCIOS PROPOSTOS

### 1.3. 1. Responda as questões abaixo

1. O que é lógica de programação?
2. O que é algoritmo?
3. Como os algoritmos podem ser representados?
4. O que é descrição narrativa?
5. Dê um exemplo de algoritmo representado em descrição narrativa.
6. Qual a desvantagem de se utilizar a descrição narrativa para representar os algoritmos?
7. O que é pseudocódigo?
8. Dê um exemplo de pseudocódigo.
9. O que é fluxograma?
10. Dê um exemplo de utilização do fluxograma.
11. Fale sobre o diagrama de Chaplin.
12. Dê exemplo de um algoritmo representado pelo diagrama de Chaplin.
13. Como se transforma um algoritmo num programa?
14. O que é linguagem de programação?
15. O que é um programa?
16. Como os programadores devem proceder para desenvolver um programa?
17. Como pode ser realizada a codificação de algoritmo para transformá-lo num programa de computador?
18. Exemplifique a codificação do algoritmo da troca de lâmpada velha na linguagem de programação.
19. Exemplifique a codificação do algoritmo do cálculo da média em uma linguagem de programação.
20. Qual é a estrutura básica de um programa?
21. Durante a fase de elaboração do algoritmo, que perguntas devem ser respondidas?
22. Onde ocorre o processamento dos dados dentro do computador?
23. Como são entrados no computador os dados necessários à resolução do problema?
24. Cite alguns dispositivos utilizados para entrar dados na memória do computador.
25. Cite alguns dispositivos utilizados para saída das informações no computador.
26. O que é memória?



### 1.3.2. Coloque falso (F) ou verdadeiro (V)

( )	Programadores não devem iniciar o desenvolvimento de um programa elaborando antes o fluxograma ou o pseudocódigo
( )	Fluxograma ou pseudocódigo são construídos a fim de demonstrar a linha de raciocínio lógico para resolução do problema
( )	C++, C#, Python, Lua, Pascal, Java, Delphi, C e Windows são linguagens de programação
( )	O algoritmo pode ser codificado em qualquer Linguagem de Programação
( )	Os dados necessários à resolução do problema não são entrados via teclado
( )	Os dados são armazenados na memória do computador em posições denominadas de variável.

### 1.3.3. Elaboração de algoritmos descritivos

Elabore algoritmos descritivos para ensinar uma pessoa totalmente leiga a realizar as seguintes tarefas:

1. Chupar bala;
2. Estourar pipoca numa panela usando o fogão.
3. Trocar uma lâmpada queimada que está no teto, supondo que temos uma escada com altura suficiente e uma caixa com lâmpadas de diferentes potências.
4. Estourar pipoca numa panela usando o fogão, utilizando somente os verbos acender, colocar, desligar e misturar, bem como usar somente os objetos panela, tigela, fogo, sal, milho de pipoca, pipoca e manteiga.
5. Trocar a lâmpada queimada empregando somente os verbos colocar, descer, escolher, girar e subir. Além disso, usar também apenas os objetos escada, lâmpada queimada, lâmpada nova, soquete.
6. Colocar exatamente 4 litros de água num garrafão que tem capacidade para 5 litros. Pode-se usar também outro garrafão com capacidade de 3 litros e a fonte de água a vontade.
7. Atravessar três jesuítas e três canibais para o outro lado de um rio. Para isso, há um barco com capacidade para duas pessoas. Por medidas de segurança, não se deve permitir que em alguma margem a quantidade de jesuítas seja inferior à dos

canibais. Qual a solução para efetuar a travessia com segurança? Elabore um algoritmo mostrando a resposta, indicando as ações que concretizam a solução deste problema.

8. Solucionar o problema da Torre de Hanói, que envolve um ambiente formado por uma base com 3 pinos (pA, pB e pC). No pino pA há uma pilha de 3 discos furados com diâmetros diferentes (dP, dM e dG), ordenados de tal forma que o disco maior (dG) está em baixo e o menor (dP) em cima, formando assim uma torre conforme a figura a seguir:

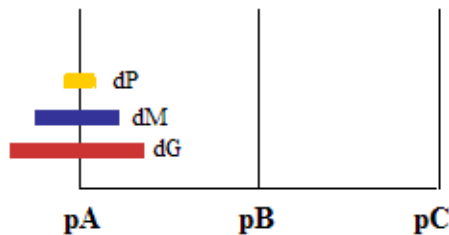


Figura 2 - Ambiente da Torre de Hanói.

O problema consiste em transferir-se a torre do pino pA para o pino pC obedecendo as seguintes restrições:

- a) Só é possível movimentar um disco por vez para qualquer pino;
  - b) Um disco maior nunca poderá ser colocado sobre um menor;
  - c) A solução deverá ser encontrada com o menor número de passos possível.
9. Mostrar os passos necessários para trocar um pneu furado, considerando as seguintes situações:
- a) Trocar o pneu (lado do motorista; traseira do carro);
  - b) Antes de efetuar a troca, verificar se o pneu reserva está em condições de uso;
  - c) Verificar se existe algum pneu furado, se houver verificar o pneu reserva e, então, trocar o pneu correto.

## 2. PROGRAMAÇÃO ESTRUTURADA

*“Algoritmo só se aprende fazendo”.*

### 2.1. INTRODUÇÃO

Segundo a literatura, qualquer programa de computador pode ser escrito utilizando apenas três estruturas básicas, a saber: Sequência, Condição e Repetição. Isto significa que, na construção de um algoritmo, deve ser criado um fluxo de ações a ser executado sequencialmente de cima para baixo e da esquerda para a direita.

Para isso, faz-se necessário formular as seguintes perguntas: quais dados devem ser usados para solucionar o problema? Qual é o tipo desses dados? (ou seja, são do tipo inteiro, real, literal ou lógico?) Qual é o tamanho desses dados? (ou seja, quanto de memória vão ocupar?).

Nos itens seguintes são abordados a descrição e os exemplos sobre tipos de dados e demais assuntos relacionados. São também apresentados exemplos sobre as estruturas básicas, bem como uma lista de exercícios para fixação dos conceitos.

### 2.2. TIPOS DE DADOS, VARIÁVEIS E OPERADORES

A seguir são mostrados os tipos de dados e os operadores de atribuição, aritméticos, lógicos e relacionais que podem ser usados pelo computador, bem como o conceito de variável.

#### 2.2.1. Tipos de dados

O computador recebe e manipula dados, transformando-os em informações através da execução de instruções. Uma instrução comanda o funcionamento da máquina e determina como devem ser tratados os dados, os quais podem ser dos seguintes tipos:

- Dados numéricos
- Dados literais
- Dados lógicos

Recordando:

- Conjunto dos números naturais (N)

$$N = \{ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, \dots \}$$

- Conjunto dos números inteiros (Z)

$$Z = \{ \dots, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, \dots \}$$

- Conjunto dos números fracionários (Q)

$$Q = \{ x/y \mid x, y \text{ pertencem a } Z \}$$

Conjunto dos números reais (R) é formado pela união do conjunto Q.

#### 2.2.1.1. Dados numéricos inteiros

Este tipo de dados são números pertencentes ao conjunto dos números inteiros (Z). Podendo assumir valores negativos, nulo e positivos. Exemplos:

número inteiro positivo	1
número inteiro	0
número inteiro negativo	-1

#### 2.2.1.2. Dados numéricos reais

Este tipo de dados são números pertencentes ao conjunto dos números fracionários (Q). Podendo assumir valores negativos, nulos e positivos. Exemplos:

número real positivo com duas casas decimais.	8.5
número real positivo com zero casas decimais.	128
número real negativo com uma casa decimal.	-7.2
número real com zero casas decimais.	0

### 2.2.1.3. Dados literais

São caracteres tais como letras, dígitos e/ou símbolos especiais. Podem ser chamados também como dados alfanuméricos, cadeias de caracteres ou string. No algoritmo, estes dados são delimitados pelas aspas (“cadeia de caracteres”). Exemplos:

literal de comprimento 6.	“Carlos”
literal de comprimento 1.	‘A’
literal de comprimento 6.	“2*4+3=”
literal de comprimento 7.	“KaDif”
literal de comprimento 1.	‘3’

### 2.2.1.4. Dados lógicos

São chamados de dados booleanos. Eles representam dados lógicos com verdadeiro e falso. Nos algoritmos seus valores são delimitados pelo ponto (.V.). Exemplos:

valor lógico verdadeiro	.V.
valor lógico falso	.F.

### 2.2.2. Variáveis

Variável é um nome (rótulo/etiqueta) colocado(a) em uma parte da memória, cujo conteúdo (valor) pode ser alterado durante a execução do programa. Embora seja possível alterar o valor armazenado na memória (na variável), é possível armazenar só um valor de cada vez. A variável é composta de dois elementos básicos: a) conteúdo – valor atual da variável e b) identificador – nome dado à variável.

O identificador deve obrigatoriamente iniciar com uma letra e os caracteres seguintes com letras ou números. Exemplos:

Armazena valor 4 na variável A	A 4
Armazena valor 6 na variável B	B 6
O valor que está em A é multiplicado pelo valor que está em B, e o resultado é armazenado na variável C	C A * B

Vale lembrar que quando um valor é armazenado em uma variável, o valor antigo será perdido. Alguns identificadores permitidos: K, NomeAluno, N1 e Nota\_1. Alguns identificadores que não são permitidos: 2T, K – Z, Aluno[3], X/2, J\*Y, “Nota”, média, sal-Liquido.

Todas as variáveis usadas em um algoritmo (pseudocódigo) devem ser declaradas antes de serem usadas. Isto faz necessário para permitir a reserva de espaço na memória.

Sintaxe da declaração:

**Var**

**<nome\_da\_variável> : <tipo\_de\_dados>;**

Regras:

- A palavra **Var** deve aparecer uma única vez;
- Para um mesmo tipo de dado, podem ser listadas várias variáveis (de modo que seus nomes devem ser separados por vírgula);
- Variáveis de tipos diferentes devem ser declaradas em linhas diferentes.

Exemplos:

Var

nome, endereco, e\_mail: literal;

idade, telefone: inteiro;

valor\_mensalidade, desconto: real;

tem\_desconto: logico;

### 2.2.3. Variáveis *versus* alocação de memória

O particionamento da memória para armazenamento de dados é realizado após análise do problema a ser resolvido. Deve-se responder as seguintes perguntas:

- Qual o problema a resolver?
- Quais os dados que devem ser armazenados?
- Que tipo de dado deve ser utilizado?
- Qual é o tamanho dos dados a serem utilizados?
- Como resolver o problema?

Digamos que o problema a ser resolvido é o clássico **somar dois números quaisquer**, tais como  $2 + 3$ ,  $5 + 6$ ,  $7 + 10$ ,  $333 + 2$ , entre outros números. Neste caso, os dados que devem ser armazenados, considerando os valores, são 2 e 3:

Diagrama da equação  $2 + 3 = 5$ . O número 2 é rotulado como 'primeiro' (em português) com uma seta laranja apontando para ele. O número 3 é rotulado como 'segundo' (em português) com uma seta laranja apontando para ele. O número 5 é rotulado como 'resultado' (em português) com uma seta vermelha apontando para ele.

Onde 2 é o **Primeiro** número, 3 é o **Segundo** número e 5 é o **Resultado**, ou seja, o total da soma dos dois números. Então, posso rotular as partes da memória do computador, onde os valores 2, 3 e 5 serão armazenados, como sendo **Primeiro**, **Segundo** e **Resultado** respectivamente.

Estas partes de memória podem receber também outros valores dependendo dos valores escolhidos para realizar a operação. Estas partes da memória são, portanto, chamadas de variáveis e sempre que um novo valor for colocado nelas, o conteúdo anterior é perdido ou sobrescrito.

No exemplo acima, pode-se concluir que os dados são do tipo inteiro.

Diagrama da equação  $2 + 3 = 5$ . O número 2 é rotulado como 'Inteiro' (em português) com uma seta laranja apontando para ele. O número 3 é rotulado como 'Inteiro' (em português) com uma seta laranja apontando para ele. O número 5 é rotulado como 'Inteiro' (em português) com uma seta vermelha apontando para ele.

Portanto, a porção de memória a ser reservada para as variáveis rotuladas como **Primeiro**, **Segundo** e **Resultado** terá um dimensionamento apropriado para o tipo inteiro.

A memória do computador pode ser particionada para armazenar dados de outros tipos também, tais como dados do tipo real, literal e lógico. A diferença destes tipos de dados consiste apenas no tamanho da porção de memória a ser reservada, como ilustra tabela a seguir.

TIPO DE DADO	TAMANHO
Inteiro	2 bytes
Literal [1] (1 caracter)	1 byte

Real	4 bytes
Lógico	1 byte

Lembrando: 1 byte = 8 bits; 1 bit pode assumir apenas um de dois valores (0 ou 1). Cada tipo de dado pode assumir de um valor mínimo a um valor máximo, como mostra a tabela abaixo:

<b>Inteiro curto</b>	<b>-128 até +128</b>	<b>1 byte</b>
<b>Inteiro</b>	<b>-32.768 a 32.767</b>	<b>2 bytes</b>
<b>Inteiro longo</b>	<b>-2.147.483.648 a 2.147.483.647</b>	<b>4 bytes</b>
<b>Real</b>	<b>2.9 e-39 até 1.7 e38</b>	<b>4 bytes</b>
<b>Double</b>	<b>5.0 e-324 até 1.7e308</b>	<b>8 bytes</b>

Relembrando o exemplo da soma dos dois números, se os números a serem somados estão dentro da seguinte faixa de valores:

$$\begin{array}{c}
 \text{2} + \text{3} = \text{5} \\
 \begin{array}{ccc}
 \nearrow & \uparrow & \nwarrow \\
 -32.768 \text{ a } 32.767 & -32.768 \text{ a } 32.767 & -32.768 \text{ a } 32.767
 \end{array}
 \end{array}$$

Então, a porção de memória a ser reservada se enquadra no tipo inteiro. É importante ressaltar também que o processo de rotular as partes da memória com nomes, isenta o programador de conhecer os endereços da memória expressos em hexadecimal. A figura a seguir ilustra o particionamento da memória (a numeração hexadecimal foi substituída pela decimal apenas para facilitar o entendimento).



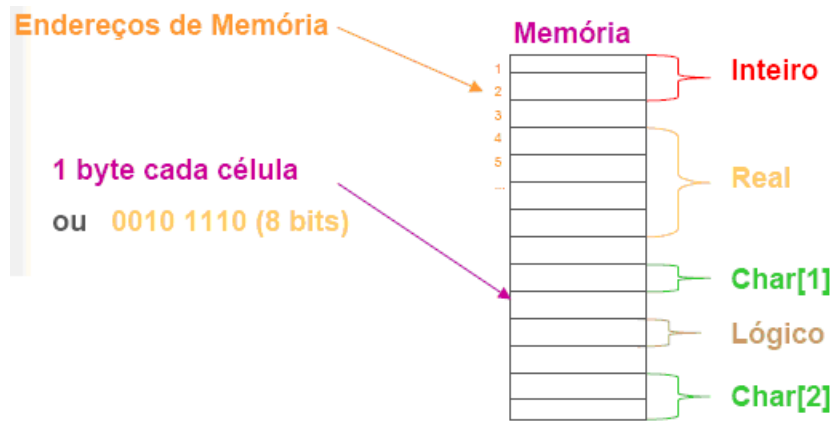


Figura 3 - Esquema de memória

Voltando ao problema da soma de dois números, pode-se agora responder a última pergunta “Como resolver o problema?”. A resposta é: definir variáveis e .... A figura a seguir ilustra todo o processo.

informar **passo a passo as ações** que serão realizadas.  
A esta definições de passos da-se o nome de algoritmo.



Nome Simbólico	Posição Inicial	Tipo Dado
primeiro	01	Inteiro
segundo	03	Inteiro
resultado	05	Inteiro

#### ■ O Algoritmo

##### Variáveis

primeiro : Inteiro;  
segundo : Inteiro;  
resultado : Inteiro;

##### Passo a Passo

- 1)Pergunte o primeiro número;
- 2)Armazene ele em primeiro;
- 3)Pergunte o segundo número;
- 4)Armazene ele em segundo;
- 5)Coloque em resultado a soma do valor armazenado em primeiro somado com o valor armazenado em segundo;



Figura 4 - Variáveis versus mapeamento de memória

## 2.2.4. Operadores

### 2.2.4.1. Operador de atribuição

O operador de atribuição (`=`) é usado para representar que uma dada variável está recebendo um valor como exemplificado a seguir:

**cont** 1 (Atribuindo o valor 1 à variável **cont**)

**T** **cont** (Atribuindo o valor da variável **cont**, valor igual a 1, à variável **T**)

**Soma** **Soma** + **Media** (Atribuindo o valor da soma dos valores armazenados nas variáveis **Soma** e **Media** à variável **Soma**)

### 2.2.4.2. Operadores aritméticos e funções

Para realizar cálculos matemáticos é necessário saber qual a representação dos símbolos de operações matemáticas, como mostrado no quadro abaixo:

Operador	Operação	Exemplo
+	Adição	$2 + 5 = 7$
-	Subtração	$12 - 10 = 2$
*	Multiplicação	$5 * 5 = 25$
/	Divisão (onde o resultado será um número real)	$7 / 2 = 3,5$
<b>DIV</b>	Divisão (onde o resultado será um número inteiro)	$13 \text{ div } 2 = 6$ $5 \text{ div } 2 = 2$
<b>MOD</b>	Resto de uma divisão	$13 \text{ mod } 2 = 1$
<b>**</b> ou <b>exp(a, b)</b>	Exponenciação	$5 ** 2$ ou $\text{exp}(5, 2)$
<b>Sqrt(x)</b>	Raiz quadrada	$\text{Sqrt}(9)$
<b>Exp(ln(x)*n)</b>	Exponenciação $x^n$	$\text{Exp}(\ln(5)*4)$

Exemplificando através da elaboração de um algoritmo que resolve:

a) $x = a^b$	b) $y = a$
Algoritmo A1;	Algoritmo A2;

Var x, a, b:real; início a 10; b 2; x Exp(ln(a)*b); escreva(x); fim.	var y, a: real; início a 16; y Sqrt(a); escreva(y); fim
--	--

### 2.2.4.3. Operadores Lógicos

Os operadores usados em expressões lógicas são os Operadores Lógicos apresentados na tabela a seguir.

Operador	Relação	Descrição	Exemplo
E (And)	E lógico	Conjunção (as duas ao mesmo tempo)	(Media > 5). E. (Freq > 75)
Ou (Or)	Ou lógico	Disjunção Inclusiva	(Flag = .V.) .Ou. (Cont > 20)
Não (Not)	Negação lógica	Negação	Não (.V.)
Ou-X (Xor)	Ou 'Exclusivo'		(Sal <= 1500) Ou-X (NumDep > 3)

Veja a tabela verdade apresentada a seguir para entender melhor os operadores lógicos, onde A e B são variáveis do tipo lógico.

A	B	A .e. B	A .ou. B	A .ou-X. B	Não (A .e. B)	Não (A .ou. B)	Não (A .ou-X. B)
F	F	F	F	F	V	V	V
F	V	F	V	V	V	F	F
V	F	F	V	V	V	F	F
V	V	V	V	F	F	F	V

#### 2.2.4.4. Operadores relacionais

Além de operações matemáticas, é frequente a comparação de informações. Por exemplo, se a média aritmética do aluno for maior ou igual a 5 então o professor toma a decisão de aprovar o aluno. Para isso, são usados os operadores relacionais listados na tabela a seguir.

Operador	Relação	Exemplo
=	Igualdade	Nome = 'Marcos de Souza'
ou <>	Diferente	Saldo 1100,00
>	Maior que	Salário >= 13000,00 Idade > = 18
ou >=	Maior ou igual que	Média 7,0
<	Menor que	RendaLiquida <= 7000
ou <=	Menor ou igual que	ValorMensalidade 390

#### 2.2.4.5. Prioridade para execução das operações no computador

O computador avalia as expressões e executa as operações mistas na seguinte ordem:

Ordem	Operações
1º	Parênteses e funções (resolvidos da esquerda para a direita)
2º	Multiplicação (*), Divisão (/ e div) e Resto (Mod) (resolvidos da esquerda para a direita)
3º	soma e subtração
4º	Operadores relacionais: >, <, =, <=, >=, <=, >=
5º	Operador Lógico Não
6º	Operador Lógico E
7º	Operador Lógico Ou

### 2.2.5. Exercícios propostos

1) Responda as questões abaixo:

- O que é uma variável?
- O que é tipo de dado?
- O que você entende por memória do computador?
- Quais as regras que devem ser consideradas para colocar o nome (rótulo ou identificador) de uma variável?
- Quais as regras que devem ser consideradas para a declaração de uma variável?

2) Relacione as colunas

Tipo de dado		Descrição	
a)	Numérico inteiro	( )	São caracteres tais como: letras, dígitos e/ou símbolos especiais. Pode-se também chamar estes dados de : alfanuméricos, cadeia de caracteres ou string. No algoritmo estes dados são delimitados pelo apóstrofo.
b)	Numérico real	( )	São chamados de booleanos. Pode assumir dois valores: Falso e Verdadeiro. No algoritmo seus valores são delimitados pelo ponto (.V.).
c)	Literal	( )	São números pertencentes ao conjunto dos números inteiros (Z), podendo assumir valores negativos, nulos e positivos.
d)	Lógico	( )	São números pertencentes ao conjunto dos números fracionários (Q), podendo assumir valores negativos, nulos e positivos.

3) Classifique os dados abaixo de acordo com seu tipo (inteiro, real, literal ou lógico).

Dado	Tipo de dado	Dado	Tipo de dado
0.9245		'xyz'	
'+25197'		.V.	
.F.		-0.5	
'V.'		+16	
322		-5	
12.0		'10'	
'-0.0'		'salario'	
34.21		141	
'3-1+7='		'XyZabC'	
'Onde?'		-11.5	

4) Na lista seguinte, assinale com **V** os nomes de variáveis válidos e com **I** os inválidos.

( )	Xyz	( )	3xybc	( )	N
( )	12b	( )	_k	( )	nota1
( )	_	( )	Num	( )	4
( )	B321	( )	_3	( )	T0143
( )	X1_a3	( )	_k173	( )	P2
( )	AB CWD	( )	cont...	( )	preco-unitario

5) Indique a prioridade na avaliação das expressões

	Soma e subtração
	Operador lógico E
	Operador Lógico Não
	Parênteses e funções (resolvidos da esquerda para a direita)
	Operadores relacionais: >, <, >=, <=, <>
	Multiplicação (*), Divisão (/ e Div) e Resto (Mod) (resolvidos da esquerda para a direita)
	Operador lógico Ou

6) Dadas as declarações de variáveis abaixo, monte as tabelas de símbolos correspondentes.

a) VAR  a,b,c : real;  delta : real;  positivo : lógico;  raiz1, raiz2 : real;		<i>Nome simbólico</i>	<i>Posição inicial</i>	<i>Tipo de dado</i>
		a	1	Real
b) VAR  idade: inteiro;  nome, profissao: literal [20];  endereço: literal [30];  numero: inteiro;  salario_liquido: real;		<i>Nome simbólico</i>	<i>Posição inicial</i>	<i>Tipo de dado</i>

7) Elabore um exercício sobre o conteúdo apresentado até o presente momento. Pode ser criativo.

8) Dado os enunciados abaixo, identifique as variáveis necessárias na elaboração do algoritmo. Dê nome para essas variáveis e identifique os tipos de dados possíveis:

8.1) Crie um algoritmo que calcule a soma de dois números inteiros.

8.2) Crie um algoritmo que calcule a média de um aluno composta por duas provas.

8.3) Crie um algoritmo que calcule o salário líquido de um funcionário, considerado:

a) os dados do funcionário: nome, RG e telefone.

b) salário bruto de R\$ 6700,00

c) descontos de R\$ 500,00

9) Dados os identificadores abaixo, assinale somente os que são aceitos em Algoritmos.

( ) ValorP	( ) 3 x 4 ( ) MARIA	( ) X1B2z3 ( ) 3	( ) Placa-Carro ( ) SALA25 ( ) 'NOTA'	( ) notadoaluno	( ) AB xyz ( ) Algoritmo34
---------------	---------------------------	------------------------	---	--------------------	----------------------------------

<input type="checkbox"/> C146	<input type="checkbox"/> KM/H	<input type="checkbox"/> 'Nota'	<input type="checkbox"/> Algoritmo_1	<input type="checkbox"/> _1aNota	<input type="checkbox"/> A426
<input type="checkbox"/> A*B	<input type="checkbox"/> 3M	<input type="checkbox"/> ABY	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> c145
<input type="checkbox"/> B2	<input type="checkbox"/> Data	<input type="checkbox"/> OH!	Nome_da_empresa	<input type="checkbox"/> _	<input type="checkbox"/> 7kx
<input type="checkbox"/> b	Pagto	<input type="checkbox"/> Nota1	<input type="checkbox"/> SALÁRIO-	<input type="checkbox"/> alg 01	
<input type="checkbox"/>	<input type="checkbox"/> DaTa	<input type="checkbox"/> _km	LÍQUIDO	<input type="checkbox"/> 1N	
<input type="checkbox"/>	<input type="checkbox"/> _x173	<input type="checkbox"/> axy4	<input type="checkbox"/> 1º algoritmo	<input type="checkbox"/> Média	
salário	<input type="checkbox"/> Cont		<input type="checkbox"/> Nota*do*aluno	<input type="checkbox"/> inicio	
<input type="checkbox"/>				<input type="checkbox"/> vetor	
Nome					

12) Indique os tipos de dados (inteiro, real, literal, lógico) que podem ser armazenados nas variáveis abaixo:

Var

cpf:

nome:

cidade:

renda:

contribuinte\_ativo:

13) Na hora de criar o nome da variável (identificador) é necessário seguir regras.

Indique com V à regra verdadeira e F para regra falsa.

<input type="checkbox"/>	Símbolos especiais (% , & , * , - , / , etc) NÃO podem ser usados;	<input type="checkbox"/>	O símbolo especial permitido é o ( _ ) underline.
<input type="checkbox"/>	Espaços em branco pode ser usados;	<input type="checkbox"/>	Não pode usar espaços em branco;
<input type="checkbox"/>	Não pode iniciar com número;	<input type="checkbox"/>	Pode usar acento agudo ou crase;
<input type="checkbox"/>	Só pode iniciar com uma letra;	<input type="checkbox"/>	É permitido iniciar com número
<input type="checkbox"/>	Símbolos especiais (% , & , * , - , / , etc) são permitidos;	<input type="checkbox"/>	Números podem ser usados para compor o o nome da variável após iniciar com letra;

14) Sendo A, B, C e D, variáveis numéricas, cujos conteúdos são iguais a 4, 5, 8.5 e 5, respectivamente, quais os valores fornecidos por cada uma das expressões abaixo.

a) $80 / A \text{ MOD } B + C$	
b) $A / B \text{ DIV } 2 - C * 5$	



c) $B * C - R / 4 * P - 1$	
d) $D \text{ MOD } (A + 5) - C * A$	
e) $A - D * (3 + 5 * C) / 2 - 4 * B$	

15) Dada a variável numérica Idade e as variáveis literais NOME e PROFISSAO, completar o quadro a seguir com os resultados lógicos (falso ou verdadeiro) obtidos como resultado das relações, tendo em vista os valores atribuídos a estas variáveis.

	Idade	Nome	Profissao	Idade > 18	Nome <> “Clara”	Profissao= “professor”
a)	16	“João”	“Estagiário”			
b)	98	“Marta”	“Professor”			
c)	24	“Juliana”	“Analista”			
d)	18	“Clara”	“Recepcionista”			
e)	20	“José”	“Técnico”			
f)	27	“Ricardo”	“professor”			

16) Considerando as variáveis numéricas A e B e as variáveis literais Nome e Profissão bem como a variável lógica ATIVO, contendo os valores 5, 16, “JOANA”, “CONTADOR” e VERDADEIRO, respectivamente, deve-se avaliar as expressões a seguir:

- a)  $B + 3 \geq A * 2$  .OU. NOME <> “CRISTINA”
- b)  $B / 2 \leq A * 2$  .OU. PROFISSÃO = “Analista”
- c) NOME <> “JOANA” .E. PROFISSÃO = “CONTADOR”
- d) PROFISSÃO = “ENGENHEIRO” .E. **não** ATIVO
- e)  $A + 4 > B / 2$  .E. PROFISSÃO = “PROFESSOR”
- f) **não** (B + 2 >= A .E. **não** ATIVO)

17) Sejam as variáveis lógicas P, A, X, B contendo, respectivamente, os valores verdadeiros, falso, falso, verdadeiro. Indique o resultado das operações:

a) <b>Não</b> X		f) $P \vee X$	
b) $P \wedge X$		g) $A \wedge B$	
c) $A \vee B$		h) $A \wedge X$	

d) A .ou. X		i) Não B	
e) P .ou. B		j) P .ou. B	

18) Dadas as expressões aritméticas abaixo, escreva-as de forma que o compilador de uma linguagem de programação como o C#, por exemplo, entenda:

a) $3(X + 2Y) - a$	
b) $\frac{X+2}{3} - 3\frac{Y-3B}{2}$	

## 2.3. TÉCNICA DE PROGRAMAÇÃO ESTRUTURADA

A seguir são apresentadas as estruturas que compõem a técnica de programação estruturada e alguns exemplos.

### 2.3.1. Estrutura Sequencial

Esta estrutura permite a execução de um grupo de ações sequencialmente.

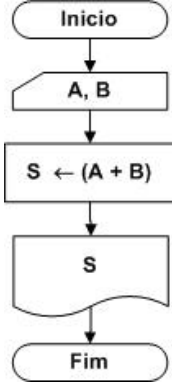
#### 2.3.1.1. Sintaxe

A seguir é apresentada a sintaxe utilizada para elaborar o pseudocódigo e o fluxograma.

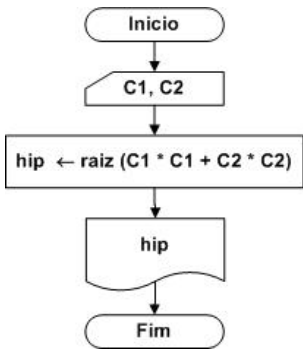
Pseudocódigo	Fluxograma
<pre>Comando_1; Comando_2; Comando_n;</pre>	<pre>graph TD; Start(( )) --&gt; C1[Comando-1]; C1 --&gt; C2[Comando-2]; C2 --&gt; Cn[Comando-n]; Cn --&gt; End(( ))</pre>

#### 2.3.1.2. Exemplos

Os algoritmos ALG01 a ALG17, apresentados acima, são exemplos de algoritmos elaborados utilizando a estrutura sequencial. Além deles, observe também os algoritmos apresentados a seguir, que realizam: soma de dois valores (ALG18); cálculo da hipotenusa (ALG19), conversão de um valor em horas e em minutos para um valor em minutos (ALG20), conversão da temperatura em graus Celsius para graus Fahrenheit (ALG21), aumento do salário de um funcionário de acordo com certo percentual de aumento (ALG22) e cálculo da área do círculo (ALG23).

ALG18 – Soma de dois números	
Análise do problema	Descrição narrativa
1 – O que tenho? dois valores (A e B) 2 – O que quero? Soma dos dois valores (S) 3 – Como? Fornecer valores A e B; $S \leftarrow A + B$ ; Mostra o resultado (S).	1 - Fornecer valores A e B; 2 - $S \leftarrow A + B$ ; 3 - Mostra o resultado (S).
Pseudocódigo	Fluxograma
Algoritmo SomaAB; Var A, B, S : inteiro; Início Leia (A, B); $S \leftarrow A + B$ ; Escreva (S); Fim	 <pre> graph TD     Inicio([Início]) --&gt; ReadAB[/A, B/]     ReadAB --&gt; CalcS[S ← (A + B)]     CalcS --&gt; WriteS[/S/]     WriteS --&gt; Fim([Fim]) </pre>
Codificação em PASCAL	Aparece no monitor quando o programa é executado
Program SomaAB; Uses WinCrt; Var A, B, S : integer; Begin Writeln ('Digite A:'); Readln (A); Writeln ('Digite B'); Readln (B); S := A + B; writeln ('Soma = ', S); End.	Nada Nada Nada Nada Nada Digite A: <Digitar o valor de A pelo teclado> Digite B: <Digitar o valor de B pelo teclado> Nada Soma = <valor armazenado em S> Nada
Codificação em linguagem C	Aparece no monitor quando o programa é executado
#include <stdio.h> main () { int A, B, S; puts ("Digite A:"); scanf ("%d", &A); puts ("Digite B"); scanf ("%d", B); S = A + B; printf ("Soma = %d", S); } }	Nada Nada Nada Nada Digite A: <Digitar o valor de A pelo teclado> Digite B: <Digitar o valor de B pelo teclado> Nada Soma = <valor armazenado em S> Nada
Codificação em linguagem C# (Console)	Aparece no monitor após execução do programa
using System; namespace SomaAB { class Program { static void Main(string[] args) { int A, B, S; Console.WriteLine("Digite A: "); A = int.Parse(Console.ReadLine()); Console.WriteLine("Digite B: "); B = int.Parse(Console.ReadLine());	Nada Nada Nada Nada Nada Nada Nada Nada Digite A: <Deve ser digitado algum valor pelo teclado> Digite B <Deve ser digitado algum valor pelo teclado>

<pre> S = (A + B)/2 Console.WriteLine("Soma = {0}", S); } } </pre>	Nada Soma = <valor armazenado em S> Nada Nada Nada
<b>Codificação em Python (Terminal)</b>	<b>Aparece no monitor após a execução</b>
<pre> A = int(input("Digite A: ")) B = int(input("Digite B: ")) S = (A + B) / 2 print(f"Soma = {S}") </pre>	Digite A:<Deve ser digitado algum valor pelo teclado> Digite B: <Deve ser digitado algum valor pelo teclado> Nada na tela, apenas o processamento na ULA, onde ocorre a soma dos valores armazenados em A e B, a divisão por 2 e a atribuição do resultado para a posição de memória rotulada de S Soma = <valor armazenado em S>

ALG19 – Cálculo da hipotenusa	
Análise do problema	Descrição narrativa
1 – O que tenho? Os catetos Oposto e Adjacente (A e B) 2 – O que quero? Hipotenusa (H) 3 – Como ? Fornece valores A e B; H raiz (A * A + B * B); Mostra o resultado (H).	1 - Fornece valores A e B; 2 - H raiz (A * A + B * B); 3 - Mostra o resultado (H).
Pseudocódigo	Fluxograma
Algoritmo CalcHip; Var C1, C2, hip : real; Início Leia (C1, C2); hip raiz (C1 * C1 + C2 * C2); Escreva (hip); Fim	 <pre> graph TD     Inicio([Início]) --&gt; Input[/C1, C2/]     Input --&gt; Process[hip ← raiz (C1 * C1 + C2 * C2)]     Process --&gt; Output[/hip/]     Output --&gt; Fim([Fim]) </pre>
<b>Codificação em Python</b>	<b>Aparece no monitor quando o programa é executado</b>
<pre> import math A = float(input ("Cateto Oposto: ")) B = float(input ("Cateto adjacente: ")) H = math.sqrt(A*A + B*B) print("O valor da hipotenusa é: {}".format(H)) </pre>	1. Nada no monitor, mas a biblioteca onde a função sqrt está programada é associada ao programa 2. Cateto Oposto:<Digitar o valor de A pelo teclado> 3. Cateto Adjacente:<Digitar o valor de B pelo teclado> 4. Nada na tela, mas ocorre o processamento da expressão matemática 5. O valor da hipotenusa é <valor armazenado em H>

ALG20– Conversão de um valor em Horas e Minutos para apenas minutos	
Análise do problema	Descrição narrativa
<p>1 – O que tenho? Tempo em horas (H) e minutos (M)</p> <p>2 – O que quero? Tempo apenas em minutos (M)</p> <p>3 – Como ? Fornece os valores H e M; <math>Mt = H * 60 + M</math>; Mostra o resultado (Mt).</p>	<p>1 - Fornece valores de Horas (H) e Minutos (M);</p> <p>2 - <math>Mt \leftarrow H * 60 + M</math>;</p> <p>3 - Mostra o resultado (Mt).</p>
Pseudocódigo	Fluxograma
<p>Algoritmo ConvHMin;</p> <p>Var</p> <p>H, M, Mt: inteiro;</p> <p>Início</p> <p>Leia (H, M);</p> <p><math>Mt \leftarrow H * 60 + M</math>;</p> <p>Escreva (Mt);</p> <p>Fim</p>	<pre> graph TD     Inicio([Início]) --&gt; Input[/H, M/]     Input --&gt; Process[Mt ← H * 60 + M]     Process --&gt; Output[Mt]     Output --&gt; Fim([Fim]) </pre>
Codificação em Python	Aparece no monitor quando o programa é executado
<pre> H = int(input("Horas :")) M = int(input("Minutos: ")) Mt = H * 60 + M print("Total de minutos é ", Mt) </pre>	<p>Horas: &lt;Digitar algum valor pelo teclado&gt;</p> <p>Minutos: &lt;Digitar algum valor pelo teclado&gt;</p> <p>Nada no monitor, mas a ULA processa a expressão matemática</p> <p>Total de minutos é &lt;valor armazenado em Mt&gt;</p>

### 2.3.1.3. Exercícios propostos

1. Resolva os algoritmos a baixo:

ALG21 – Conversão de uma temperatura em graus Celsius para Fahrenheit ( $32 + 1.8 * C$ )	
Análise do problema	Descrição narrativa
1 – O que tenho?  2 – O que quero?  3 – Como?	1 - Fornece valor da temperatura em Celsius (C)  2 - $F = 32 + 1.8 * C$ ;  3 - Mostra o resultado (F).
Pseudocódigo	Fluxograma
Codificação em Python	Aparece no monitor quando o programa é executado
	Digite graus Celsius: <Digitar algum valor pelo teclado> Nada Fahrenheit = <valor armazenado em F>

Análise do problema	Descrição narrativa
<p>1 – O que tenho? Salário atual (S) e percentual de aumento (p)</p> <p>2 – O que quero? Novo salário (NS)</p> <p>3 – Como ?</p> <p>Fornece os valores do salário atual (s) e do percentual de aumento (p);</p> <p>NS <math>S + S * p/100</math>;</p> <p>Mostra o resultado (NS).</p>	
Pseudocódigo	Fluxograma
	<pre> graph TD     Inicio([Inicio]) --&gt; Input[/S, p/]     Input --&gt; Process[NS ← S + S * p / 100]     Process --&gt; Output[/NS/]     Output --&gt; Fim([Fim]) </pre>
Codificação em Python	Aparece no monitor quando o programa é executado

ALG23 – Cálculo da área do círculo	
Análise do problema	Descrição narrativa



<p>1 – O que tenho?</p> <p>Raio do círculo (R) e valor do Pi</p> <p>2 – O que quero?</p> <p>Área do círculo (A)</p> <p>3 – Como ?</p> <p>Fornecer o valor do raio</p> <p>Encontrar o valor do Pi (3,1415);</p> <p><math>A = R * R * 3.1415</math>;</p> <p>Mostra o resultado (A).</p>	
Pseudocódigo	Fluxograma
<p>Algoritmo CalcAreaC;</p> <p>Var</p> <p>A, R: real;</p> <p>Início</p> <p>Leia (R);</p> <p><math>A = R * R * 3.1415</math>;</p> <p>Escreva (A);</p> <p>Fim</p>	
Codificação em Python	Aparece no monitor quando o programa é executado
	<p>Raio do círculo: &lt;Digitar valor do raio pelo teclado&gt;</p> <p>Nada no monitor</p> <p>Área do círculo = &lt;valor armazenado em A&gt;</p> <p>Nada</p>

2. Elabore um algoritmo que leia um valor em polegadas (Vp), calcule e exiba o valor correspondente em milímetros (Vm). Considere: 1 polegada (p) igual a 25,4 milímetros e  $V_m = V_p * p$ .

3. Elabore um algoritmo que leia a massa (m) de uma amostra de carbono (C), calcule e exiba o número de moles (nm) de carbono na amostra. Considere:  $\text{mol C} = 12$  e  $\text{nm} = m / C$ .
4. Elabore um algoritmo que leia a força (F) aplicada sobre um determinado corpo e a velocidade (V) desse corpo, calcule e exiba a potência (P). Considere:  $P = F * V$ .
5. O dono de uma padaria precisa calcular o total mensal de vendas. Elabore um algoritmo que o auxilie, que faça a leitura do total de vendas mensal em um ano e que, em seguida, mostre os totais de vendas por trimestre.
6. Na lanchonete “DeliciososLanches”, o atendente precisa anotar o nome dos clientes e a quantidade de cada item do menu (vide tabela abaixo) que ele vai pedir. Faça um programa que o auxilie, ler os dados necessários e que exiba o nome do cliente e o total da compra.

Código	ITEM	PREÇO
01	Hot dog	R\$ 11,20
02	Hamburguer	R\$ 16,60
03	Cheeseburger	R\$ 22,00
04	Refrigerante em lata	R\$ 8,00
05	Batatas fritas	R\$ 32,50
06	Misto quente	R\$ 13,00
07	Sucos naturais	R\$ 8,00

7. Faça um algoritmo que leia a cotação do dólar no dia e a quantidade de dólares que a pessoa deseja trocar por reais, calcule e exiba a quantidade correspondente em reais.
8. Elaborar um algoritmo para calcular e apresentar o valor do volume (v) de uma lata de óleo. Considere:  $v = \pi r^2 h$ ; ( $\pi$ ) é igual a 3,1415.
9. Faça um algoritmo que leia a idade de uma pessoa expressa em anos, meses e dias e mostre-a expressa apenas em dias.
10. Faça um algoritmo que leia as 3 notas (N1, N2 e N3) de um aluno e calcule a média final (MF) deste aluno. Considerar que a média é ponderada e que o peso

das notas é: 2, 3 e 5, respectivamente. Ou seja,  $MF = N1 * 0,2 + N2 * 0,3 + N3 * 0,5$ .

11. Faça um algoritmo que leia o tempo (ts) de duração de um evento expresso em segundos e mostre-o expresso em horas (H), minutos (M) e segundos (S).
12. Elabore um algoritmo para converter um valor em minutos (tM) num formato com dias (D), horas (H) e minutos (M).
13. Escrever um algoritmo que converta segundos em minutos e segundos.
14. Escrever um algoritmo que calcule a taxa de consumo (Tx) de um automóvel durante um período, conhecendo-se os valores de quilômetros percorridos (km) e a quantidade de combustível consumida (Lc).
15. Elabore um algoritmo que dados os litros gastos (L) e os quilômetros percorridos (km) por um automóvel, calcule os gastos de combustível em R\$/km.
16. Crie algoritmos para solucionar os problemas abaixo nas três formas de representação conhecidas (Descrição Narrativa, Fluxograma e Pseudocódigo):
  - Imprimir o seu nome.
  - Ler seu nome e sobrenome e imprimi-los na seguinte ordem: sobrenome e nome separados por vírgula.
  - Ler seu nome e idade e imprimi-los.
  - Imprimir o produto de 12 e 43.
  - Imprimir a média aritmética entre 8, 9 e 7.
  - Ler três valores numéricos do tipo inteiro e imprimir a média aritmética entre eles.
  - Ler um número inteiro e imprimir seu sucessor e seu antecessor.
  - Ler o lado de um quadrado e calcular a área deste quadrado e mostrar o resultado.
17. Diga qual é o valor armazenado em cada variável (TESTE DE MESA) após a execução de cada um dos comandos do algoritmo abaixo.

Algoritmo ALGXX;

Var

N, A, R, B, C : Inteiro;

V, X : real;

Início

A 10;

R 3;

V 14.0;

B 150;

C V MOD 2;

X B \* R;

X R / A;

X X - 2 \* A;

B B + 3;

N 30 + X;

B B DIV N;

V N - V + X \* R;

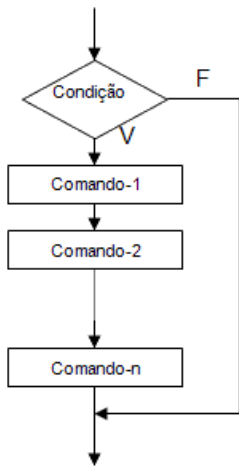
Fim.

### 2.3.2. Estrutura Condicional

Também conhecida como estrutura de tomada de decisão ou seleção, esta estrutura permite a escolha do grupo de ações a ser executadas quando determinada condição (expressão lógica) é ou não satisfeita.

#### 2.3.2.1. Sintaxe

Como pode ser observado nas sintaxes mostradas a seguir, tem-se a estrutura condicional simples, composta e aninhada.

Estrutura condicional simples	
Pseudocódigo	Fluxograma
<b>Se (Condição) Então</b> Comando_1; Comando_2; Comando_n; <b>Fim_se;</b>	
Em Python	
<b>if</b> Condição: Comando_1 Comando_2 Comando_n	Se fosse um comando só: <b>if</b> Condição: Comando_1  ---- Ou ---- <b>if</b> (Condição): Comando_1

### Estrutura condicional composta

Pseudocódigo	Fluxograma
<p><b>Se (Condição) Então</b></p> <p>    Comando_A1;     Comando_A2;     Comando_An</p> <p><b>Senão</b></p> <p>    Comando_B1;     Comando_B2;     Comando_Bn;</p> <p><b>Fim_se;</b></p>	<pre>graph TD; Entry(( )) --&gt; Cond{Condição}; Cond -- V --&gt; A1[Comando-A1]; A1 --&gt; A2[Comando-A2]; A2 --&gt; An[Comando-An]; Cond -- F --&gt; B1[Comando-B1]; B1 --&gt; B2[Comando-B2]; B2 --&gt; Bn[Comando-Bn]; An --&gt; Exit(( )); Bn --&gt; Exit;</pre>
	<b>Em Python</b>
	<pre>if Condição :     Comando_A1     Comando_A2     Comando_An else:     Comando_B1     Comando_B2     Comando_Bn</pre>

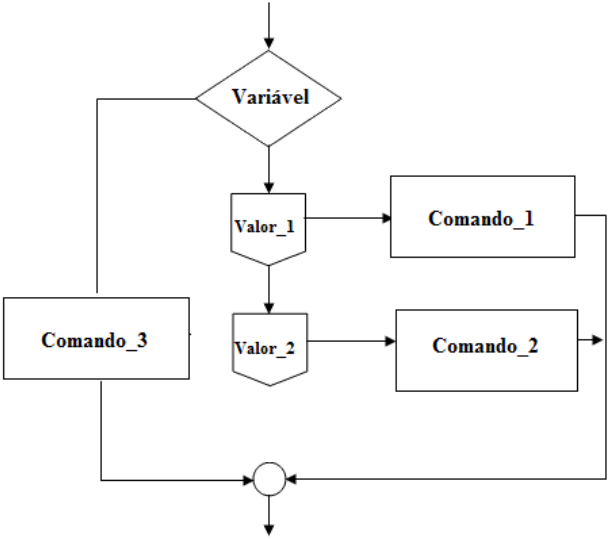
Estrutura condicional aninhada (ou encadeada)	
Pseudocódigo	Fluxograma
<pre> <b>Se</b> (Condição_1) <b>Então</b>   Comando_A1;   Comando_A2;   ...   Comando_An <b>Senão</b> <b>Se</b> (Condição_2) <b>Então</b>   Comando_B1;   Comando_B2;   ...   Comando_Bn; <b>Senão</b>   Comando_C1;   Comando_C2;   ...   Comando_Cn; <b>Fim_se;</b> <b>Fim_se;</b> </pre>	<pre> graph TD     Start(( )) --&gt; Cond1{condição_1}     Cond1 -- V --&gt; A1[Comando-A1]     A1 --&gt; A2[Comando-A2]     A2 --&gt; An[Comando-An]     An --&gt; Exit1(( ))     Cond1 -- F --&gt; Cond2{condição_2}     Cond2 -- V --&gt; B1[Comando-B1]     B1 --&gt; B2[Comando-B2]     B2 --&gt; Bn[Comando-Bn]     Cond2 -- F --&gt; C1[Comando-C1]     C1 --&gt; C2[Comando-C2]     C2 --&gt; Cn[Comando-Cn]     Bn --&gt; Merge(( ))     Cn --&gt; Merge     Merge --&gt; Exit1 </pre>
Em Python	Se for um único comando
<pre> <b>if</b> Condição_1 :     Comando_A1     Comando_A2     ...     Comando_An <b>elif</b> Condição_2 :     Comando_B1     Comando_B2     ...     Comando_Bn; <b>else:</b>     Comando_C1     Comando_C2     ...     Comando_Cn </pre>	<pre> <b>if</b> Condição_1 :     Comando_A1 <b>elif</b> Condição_2 :     Comando_B1 <b>else:</b>     Comando_C1 </pre>

Há também uma variação do condicional SE, que é utilizada quando uma situação de igualdade é testada para uma mesma variável.

Caso/escolha	
Pseudocódigo	Fluxograma
<p><b>Escolha (Variável)</b></p> <p><b>Caso</b> (valor_1) : Comando_1;  <b>Caso</b> (valor_2) : Comando_2;  <b>Caso</b> (valor_3) : Comando_3;  <b>Fim_escolha</b>;</p> <p><b>Ou desta outra forma:</b></p> <p><b>Caso</b> &lt;Variável&gt;          &lt;valor_1&gt; : &lt;Comando_1&gt;;          &lt;valor_2&gt; : &lt;Comando_2&gt;;          &lt;valor_3&gt; : &lt;Comando_3&gt;;  <b>Fim_caso</b>;</p>	<pre> graph TD     Start(( )) --&gt; Var{Variável}     Var --&gt; Val1{{Valor_1}}     Val1 --&gt; Com1[Comando_1]     Var --&gt; Val2{{Valor_2}}     Val2 --&gt; Com2[Comando_2]     Var --&gt; Val3{{Valor_3}}     Val3 --&gt; Com3[Comando_3]     Com1 --&gt; Join(( ))     Com2 --&gt; Join     Com3 --&gt; Join     Join --&gt; End(( ))   </pre>
Em Pascal	Em C#
<pre> <b>Case</b> &lt;Variável&gt; <b>of</b>   &lt;valor_1&gt; : &lt;Comando_1&gt;;   &lt;valor_2&gt; : &lt;Comando_2&gt;;   &lt;valor_3&gt; : &lt;Comando_3&gt;; <b>End</b>; </pre>	<pre> <b>switch</b> (Variável) {   case &lt;valor_1&gt; : &lt;Comando_1&gt;; break;   case &lt;valor_2&gt; : &lt;Comando_2&gt;; break;   case &lt;valor_3&gt; : &lt;Comando_3&gt;; break; } </pre>
	Em Python
	<pre> <b>match</b> &lt;Variável&gt;:   <b>case</b> &lt;valor_1&gt; :     &lt;Comando_1&gt;   <b>case</b> &lt;valor_2&gt; :     &lt;Comando_2&gt;   <b>case</b> &lt;valor_3&gt; :     &lt;Comando_3&gt; </pre>

Pseudocódigo	Fluxograma
--------------	------------



<p><b>Escolha (Variável)</b>  <b>Caso</b> (valor_1) :  Comando1;  <b>Caso</b> (valor_2) :  Comando2;  <b>Senão</b>  Comando3;  <b>Fim_escolha;</b></p> <p><b>Ou desta outra forma:</b></p> <p><b>Caso</b> &lt;Variável&gt;  &lt;valor_1&gt; :  &lt;Comando1&gt;;  &lt;valor_2&gt; :  &lt;Comando2&gt;;  <b>Senão</b>  &lt;Comando3&gt;;  <b>Fim caso;</b></p>	
Em Pascal	Em C#
<p><b>Case</b> &lt;Nome variável&gt; <b>of</b>  &lt;valor_1&gt; : &lt;Comando_1&gt;;  &lt;valor_2&gt; : &lt;Comando_2&gt;;  <b>Else</b>  &lt;Comando_3&gt;;  <b>End;</b></p>	<pre>switch (&lt;nome_variavel&gt;) {     case &lt;valor_1&gt; : &lt;Comando_1&gt;;                     break;     case &lt;valor_2&gt; : &lt;Comando_2&gt;;                     break;     default:         &lt;Comando_3&gt;;         break; }</pre>
	<b>Em Python</b>
	<pre>match &lt;nome_variavel&gt;:     case &lt;valor_1&gt; :         &lt;Comando_1&gt;     case &lt;valor_2&gt; :         &lt;Comando_2&gt;     case _ :         &lt;Comando_3&gt;</pre>

### 2.3.2.2. Exemplos

A seguir são apresentados alguns exemplos, onde foram utilizadas as estruturas: condicional simples, condicional composto, condicional encadeado e estrutura Escolha. No primeiro (ALG24) tem-se o algoritmo do cálculo da média aritmética (M), onde são consideradas apenas duas notas (P1 e P2) e é mostrada a média e uma mensagem de aprovação caso  $M \geq 5$ . Neste caso foi utilizada uma estrutura condicional simples.

ALG24 – Cálculo da média aritmética de duas notas, mostrando média e mensagem “Aprovado”, caso a média seja $\geq 5$ .	
Análise do problema	Descrição narrativa
<p>1 – O que tenho? Notas (P1 e P2)</p> <p>2 – O que quero?</p> <p>Média (M) e situação “Aprovado”</p> <p>3 – Como ?</p> <p>Fornece o valor das notas (P1 e P2)</p> <p>Calcula a média (<math>M = (P1 + P2)/2</math>);</p> <p>Mostra resultado (M);</p> <p>Verifica se aluno foi aprovado (<math>M \geq 5</math>)</p> <p>5)</p> <p>Mostra somente mensagem “Aprovado”.</p>	<p>1 - Fornece valor das notas (P1 e P2);</p> <p>2 - Calcula a média aritmética (M)</p> <p>3 - Verifica se <math>M \geq 5</math></p> <p>4 - Se sim, mostra média (M) e mensagem “Aprovado”.</p>
Pseudocódigo	Fluxograma
<p>Algoritmo CalcMedia1;</p> <p>Var P1, P2, M: real;</p> <p>Início</p> <p>  Leia (P1, P2);</p> <p>  M <math>(P1 + P2)/2</math>;</p> <p>  Escreva (M);</p> <p>  Se <math>M \geq 5</math> Então</p> <p>    Escreva ('Aprovado');</p> <p>  Fim_se;</p> <p>Fim</p>	<pre> graph TD     Inicio([Início]) --&gt; Input[/P1, P2/]     Input --&gt; Process[M ← (P1+P2)/2]     Process --&gt; OutputM[M]     OutputM --&gt; Decision{M ≥ 5}     Decision -- V --&gt; OutputA[Aprovado]     Decision -- F --&gt; Junction(( ))     OutputA --&gt; Junction     Junction --&gt; Fim([Fim]) </pre>
Codificação em PASCAL	Codificação em linguagem C# (Console)
<pre> Program CalcMedia1; Uses WinCrt; Var P1, P2, M: real; Begin Writeln ('Digite P1:'); Readln (P1); Writeln ('Digite P2:'); Readln (P2); M := (P1 + P2)/2; Writeln (M); If M &gt;= 5 then writeln ('Aprovado'); End. </pre>	<pre> using System; namespace CalcMedia1 {     class Program     {         static void Main(string[] args)         {             double P1, P2, M;             Console.WriteLine("Digite P1: ");             P1 = double.Parse(Console.ReadLine());             Console.WriteLine("Digite P2: ");             P2 = double.Parse(Console.ReadLine());             M = (P1 + P2)/2;             Console.WriteLine("média = {0}", M);             if (M &gt;= 5)                 Console.WriteLine("Aprovado");         }     } } </pre>
	Python

	<pre> P1 = float(input("Digite P1:")) P2 = float(input("Digite P2:")) M = (P1 + P2)/2 print(" a média é ", M) if M &gt;= 5 :     print(" Aprovado") </pre>
--	--

No segundo exemplo (ALG25), é utilizada uma estrutura condicional composta ao realizar um refinamento no algoritmo acima acrescentando também a mensagem de reprovado.

ALG25 – Cálculo da média aritmética de duas notas, mostrando a média e mensagem “Aprovado” ou “Reprovado” (M >= 5; aprovação) .	
Análise do problema	Descrição narrativa
<p><b>1 – O que tenho?</b> Notas (P1 e P2)</p> <p><b>2 – O que quero?</b> Média (M) e “Aprovado” ou “Reprovado”</p> <p><b>3 – Como ?</b> Fornece o valor das notas (P1 e P2) Calcula a média (M (P1 + P2)/2); Mostra o resultado (M); Verifica se aluno foi aprovado (M &gt;= 5); Mostra mensagem “Aprovado” ou “Reprovado”</p>	<p>1 - Fornecer valor das notas (P1 e P2); 2 - Calcular a média aritmética (M) 3 - Mostra o resultado (M) 4 - Verificar se M &gt;= 5 5 - Se sim, mostra mensagem “Aprovado”. 6 - Senão mostra mensagem “Reprovado”.</p>
<p>Algoritmo CalcMedia1; Var P1, P2, M: real; Início Leia (P1, P2); M (P1 + P2)/2; Escreva (M); Se M &gt;= 5 Então Escreva ('Aprovado') Senão Escreva ('Reprovado'); Fim_se; Fim</p>	<pre> graph TD     Inicio([Início]) --&gt; Input[/P1, P2/]     Input --&gt; Calc["M ← (P1+P2)/2"]     Calc --&gt; OutM[/M/]     OutM --&gt; Dec{"M &gt;= 5"}     Dec -- V --&gt; OutA[/Aprovado/]     Dec -- F --&gt; OutR[/Reprovado/]     OutA --&gt; Join(( ))     OutR --&gt; Join     Join --&gt; Fim([Fim]) </pre>
Codificação em PASCAL	Codificação em linguagem C# (Console)
<pre> Program CalcMedia1; Uses WinCrt; Var </pre>	<pre> using System; namespace CalcMedia1 {     class Program     {         static void Main(string[] args)         { </pre>

<pre> P1, P2, M: real; Begin   Writeln ('Digite P1:');   Readln (P1);   Writeln ('Digite P2:');   Readln (P2);   M := (P1 + P2)/2;   Writeln (M);   if M &gt;= 5 then     writeln ('Aprovado')   else     writeln ('Reprovado'); End. </pre>	<pre> double P1, P2, M; Console.WriteLine("Digite P1: "); P1 = double.Parse(Console.ReadLine()); Console.WriteLine("Digite P2: "); P2 = double.Parse(Console.ReadLine()); M = (P1 + P2)/2; Console.WriteLine("média = {0}", M); if (M &gt;= 5)   Console.WriteLine("Aprovado"); else   Console.WriteLine("Reprovado"); } } </pre>
	<b>Python</b> <pre> P1 = float(input("Digite P1:")) P2 = float(input("Digite P2:")) M = (P1 + P2)/2 print (" a média é ", M) if M &gt;= 5 :   print (" Aprovado") else:   print (" Reprovado") </pre>

Em seguida, é considerado também que o aluno pode ter ficado para exame (ALG26). Neste último caso foi utilizada a estrutura condicional encadeada.

ALG26 – Cálculo da média aritmética de duas notas, mostrando a média e mensagem “Aprovado”, “Reprovado” ou “Exame” ( $M \geq 5$ Aprovação; $M \geq 3$ Exame) .	
Pseudocódigo	Fluxograma
<p>Algoritmo CalcMedia3;  Var  P1, P2, M: real;  Início  Leia (P1, P2);  M <math>(P1 + P2)/2</math>;  Escreva (M);  Se <math>M \geq 5</math> Então  Escreva (‘Aprovado’)  Senão  Se <math>M \geq 3</math> Então  Escreva (‘Exame’)  Senão  Escreva (‘Reprovado’);  Fim_se;  Fim_se;  Fim</p>	<pre> graph TD     Inicio([Início]) --&gt; Input[/P1, P2/]     Input --&gt; Calc["M ← (P1+P2)/2"]     Calc --&gt; OutM[/M/]     OutM --&gt; Dec1{"M ≥ 5"}     Dec1 -- V --&gt; OutA[/'Aprovado'/]     Dec1 -- F --&gt; Dec2{"M ≥ 3"}     Dec2 -- V --&gt; OutE[/'Exame'/]     Dec2 -- F --&gt; OutR[/'Reprovado'/]     OutA --&gt; Join(( ))     OutE --&gt; Join     OutR --&gt; Join     Join --&gt; Fim([Fim]) </pre>
Codificação em PASCAL	Codificação em linguagem C# (Console)
<pre> Program CalcMedia3; Uses WinCrt; Var   P1, P2, M: real; Begin   Writeln ('Digite P1:');   Readln (P1);   Writeln ('Digite P2:');   Readln (P2);   M := (P1 + P2)/2;   Writeln (M);   If M &gt;= 5 then     Writeln ( 'Aprovado')   Else     If M &gt;= 3 then       Writeln ( 'Exame')     Else       Writeln ( 'Reprovado'); End. </pre>	<pre> using System; namespace CalcMedia1 {     class Program     {         static void Main(string[] args)         {             double P1, P2, M;             Console.WriteLine("Digite P1: ");             P1 = double.Parse(Console.ReadLine());             Console.WriteLine("Digite P2: ");             P2 = double.Parse(Console.ReadLine());             M = (P1 + P2)/2;             Console.WriteLine("média = {0}", M);             if (M &gt;= 5)                 Console.WriteLine("Aprovado");             else                 if (M &gt;= 3)                     Console.WriteLine("Exame");                 else                     Console.WriteLine("Reprovado");         }     } } </pre>
	<b>Python</b>
	<pre> P1 = float(input("Digite P1:")) P2 = float(input("Digite P2:")) M = (P1 + P2)/2 print (" a média é ", M) if M &gt;= 5 :     print (" Aprovado") </pre>

	<pre> elif M &gt;= 3:     print (" Exame") else:     print (" Reprovado") </pre>
--	--

Dois outros exemplos de utilização de estrutura condicional simples (ALG27) e encadeada (ALG28) podem ser vistos em seguida. Eles apresentam um algoritmo que lê um número inteiro entre 1 e 7 e mostra no monitor do computador o dia da semana correspondente. Caso o usuário digite um número fora desse intervalo, o algoritmo mostra uma mensagem informando “Número inválido”.

ALG27 – Algoritmo elaborado com <u>estrutura condicional simples e composta</u> que lê um número e imprime o dia da semana correspondente.	
Pseudocódigo	Fluxograma
<pre> programa DiaSemana_se; Var     Dia: inteiro; início     Escreva ('Digite um número de 1 a 7:');     Leia (Dia);     Se Dia = 1 então         escreva ('Domingo');     Fim_se;     Se Dia = 2 então         Escreva ('Segunda');     Fim_se;     Se Dia = 3 então         Escreva ('Terça');     Fim_se;     Se Dia = 4 então         Escreva ('Quarta');     Fim_se;     Se Dia = 5 então         Escreva ('Quinta');     Fim_se;     Se Dia = 6 então         Escreva ('Sexta');     Fim_se;     Se Dia = 7 então         Escreva ('Sábado');     Senão         Escreva ('Número inválido');     Fim_se; Fim. </pre>	<pre> graph TD     Inicio([Início]) --&gt; Dia[/Dia/]     Dia --&gt; D1{Dia = 1}     D1 -- V --&gt; Domingo[Domingo]     D1 -- F --&gt; D2{Dia = 2}     D2 -- V --&gt; Segunda[Segunda]     D2 -- F --&gt; D3{Dia = 3}     D3 -- V --&gt; Terca[Terça]     D3 -- F --&gt; D4{Dia = 4}     D4 -- V --&gt; Quarta[Quarta]     D4 -- F --&gt; D5{Dia = 5}     D5 -- V --&gt; Quinta[Quinta]     D5 -- F --&gt; D6{Dia = 6}     D6 -- V --&gt; Sexta[Sexta]     D6 -- F --&gt; D7{Dia = 7}     D7 -- V --&gt; Sabado[Sábado]     D7 -- F --&gt; NoInvalido[No. Inválido]     Domingo --&gt; J1(( ))     Segunda --&gt; J1     Terca --&gt; J1     Quarta --&gt; J1     Quinta --&gt; J2(( ))     Sexta --&gt; J2     Sabado --&gt; J2     NoInvalido --&gt; J2     J1 --&gt; Fim([Fim])     J2 --&gt; Fim </pre>
	Em Python
	<code>Dia = int (input("Informe um número de 1 a 7: "))</code>

	<pre> if (Dia == 1)     print ("Domingo") if (Dia == 2)     print ("Segunda") if (Dia == 3)     print ("Terça") if (Dia == 4)     print ("Quarta") if (Dia == 5)     print ("Quinta") if (Dia == 6)     print ("Segunda") if (Dia == 7)     print ("Sabado") else     print ("Número inválido") </pre>
--	--

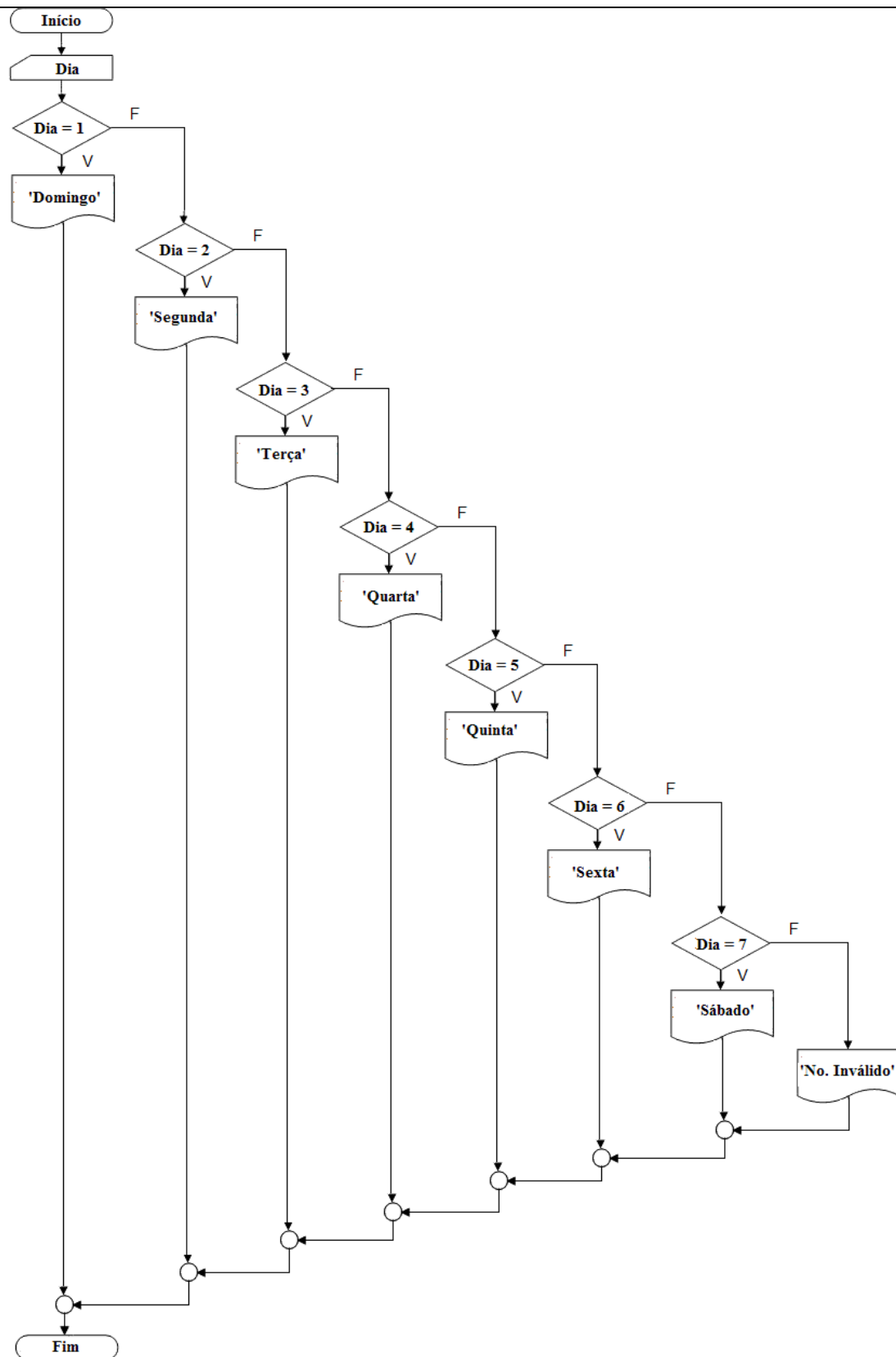
ALG28 – Algoritmo elaborado com <u>estrutura condicional encadeada</u> que lê um número e imprime o dia da semana correspondente.	
Pseudocódigo	
<pre> programa DiaSemana_se; Var     mes:inteiro; início     Escreva ('Digite um número de 1 a 7:');     Leia (Dia);     se Dia = 1 então         escreva ('Domingo')     senão         se Dia = 2 então             Escreva ('Segunda')         senão             se Dia = 3 então                 Escreva ('Terça')             senão                 se Dia = 4 então                     Escreva ('Quarta')                 senão                     se Dia = 5 então                         Escreva ('Quinta')                     senão                         se Dia = 6 então                             Escreva ('Sexta')                         senão                             se Dia = 7 então                                 Escreva ('Sábado')                             senão                                 Escreva ('Número inválido');                     fim_se;                 fim_se;             fim_se;         fim_se;     fim_se; fim. </pre>	
Em Python	
<pre> Dia = int (input("Informe um número de 1 a 7: ")) </pre>	

```
if (Dia == 1)
    print ("Domingo")
elif (Dia == 2)
    print ("Segunda")
elif (Dia == 3)
    print ("Terça")
elif (Dia == 4)
    print ("Quarta")
elif (Dia == 5)
    print ("Quinta")
elif (Dia == 6)
    print ("Segunda")
elif (Dia == 7)
    print ("Sabado")
else
    print ("Número inválido")
```



ALG28 – Algoritmo elaborado com estrutura condicional encadeada que lê um número e imprime o dia da semana correspondente.

### Fluxograma



Em Python

```
Dia = int (input("Informe um número de 1 a 7: "))
if (Dia == 1)
    print ("Domingo")
elif (Dia == 2)
    print ("Segunda")
elif (Dia == 3)
    print ("Terça")
elif (Dia == 4)
    print ("Quarta")
elif (Dia == 5)
    print ("Quinta")
elif (Dia == 6)
    print ("Segunda")
elif (Dia == 7)
    print ("Sabado")
else
    print ("Número inválido")
```

O exemplo a seguir mostra o algoritmo acima elaborado utilizando a estrutura Escolha/caso (ALG29) ao invés da estrutura condicional encadeada.

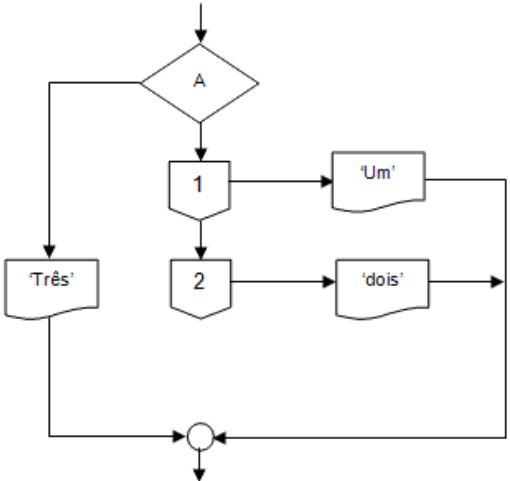
ALG29 – Algoritmo elaborado com estrutura ESCOLHA/CASO que lê um número e imprime o dia da semana correspondente	
Pseudocódigo	Fluxograma
<pre> programa DiaSemana_se; Var Dia : inteiro; inicio   Escreva ('Digite um numero de 1 a 7:');   Leia (Dia);   Escolha (Dia)     Caso (1) : Escreva ('Domingo');     Caso (2) : Escreva ('Segunda')     Caso (3) : Escreva ('Terça')     Caso (4) : Escreva ('Quarta')     Caso (5) : Escreva ('Quinta')     Caso (6) : Escreva ('Sexta')     Caso (7) : Escreva ('Sábado')   Senão     Escreva ('Número inválido');   Fim_escolha; fim. </pre>	<pre> graph TD     Inicio([Início]) --&gt; DiaInput[/Dia/]     DiaInput --&gt; DiaDecision{Dia}     DiaDecision -- V --&gt; 1[1]     1 --&gt; Domingo[Domingo]     1 --&gt; 2[2]     2 --&gt; Segunda[Segunda]     2 --&gt; 3[3]     3 --&gt; Terca[Terça]     3 --&gt; 4[4]     4 --&gt; Quarta[Quarta]     4 --&gt; 5[5]     5 --&gt; Quinta[Quinta]     5 --&gt; 6[6]     6 --&gt; Sexta[Sexta]     6 --&gt; 7[7]     7 --&gt; Sabado[Sábado]     DiaDecision -- F --&gt; NoInvalido[No. Inválido]     Domingo --&gt; Merge(( ))     Segunda --&gt; Merge     Terca --&gt; Merge     Quarta --&gt; Merge     Quinta --&gt; Merge     Sexta --&gt; Merge     Sabado --&gt; Merge     NoInvalido --&gt; Merge     Merge --&gt; Fim([Fim]) </pre>

Em Python

```
Dia = int (input("Informe um número de 1 a 7: "))
match Dia:
    case 1:
        print ("Domingo")
    case 2:
        print ("Segunda")
    case 3:
        print ("Terça")
    case 4:
        print ("Quarta")
    case 5:
        print ("Quinta")
    case 6:
        print ("Segunda")
    case 7:
        print ("Sabado")
    case _:
        print ("Número inválido")
```

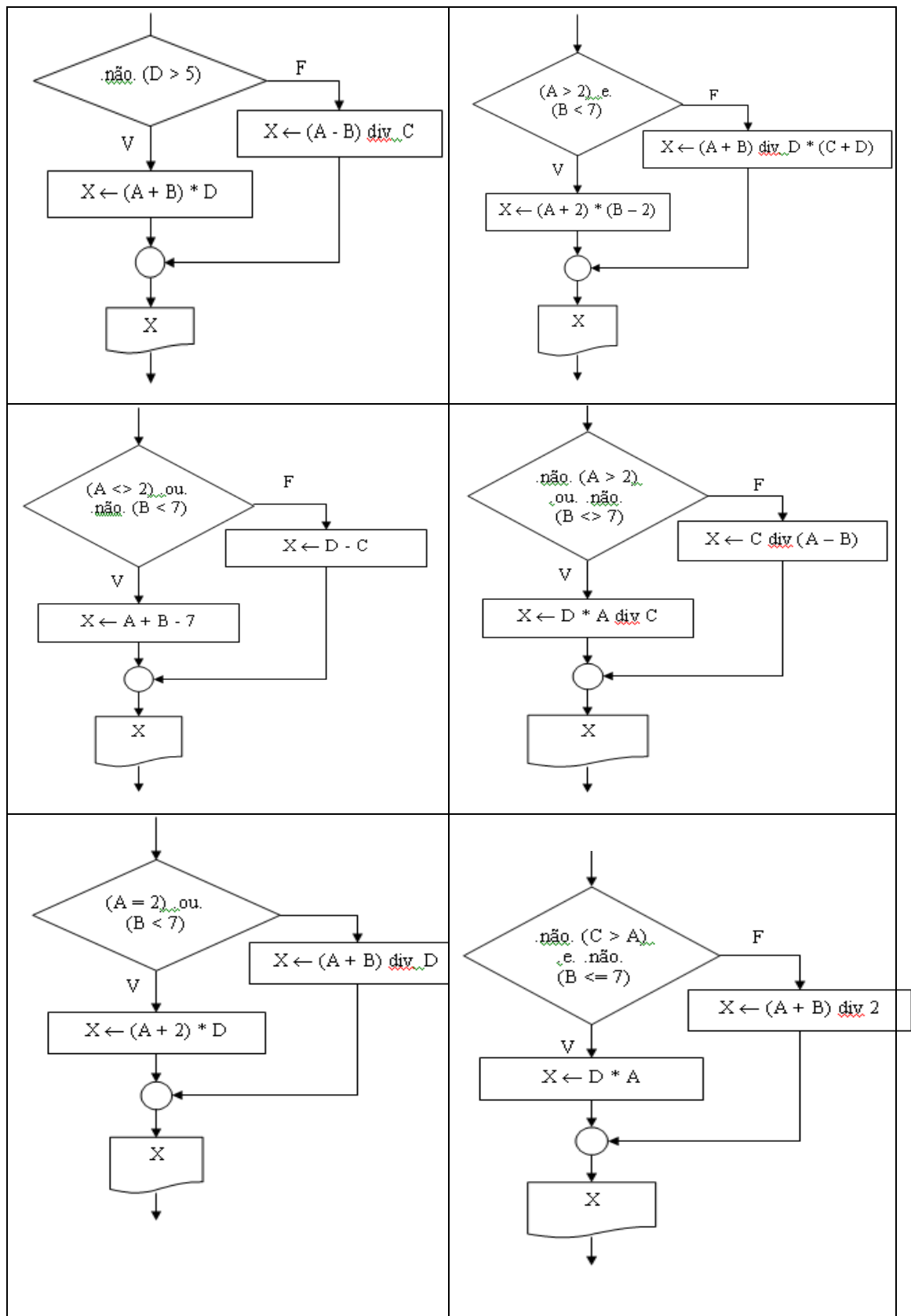
O exemplo seguinte apresenta apenas um trecho de algoritmo (elaborado com a estrutura ESCOLHA/CASO), que mostra no monitor a mensagem “Um”, “Dois” ou “Três” caso o valor digitado na variável A seja 1, 2 ou 3 respectivamente (ALG30).

ALG30 – Trecho de algoritmo elaborado com estrutura ESCOLHA/CASO que imprime a mensagem “Um”, “Dois” e “Três” caso seja digitado 1, 2 ou 3 respectivamente.	
Pseudocódigo	Fluxograma
<p><b>Exemplo 1</b></p> <p>Escolha (A)</p> <p>    Caso (1): Escreva (‘Um’);</p> <p>    Caso (2): Escreva (‘Dois’);</p> <p>    Caso (3): Escreva (‘Três’);</p> <p>Fim_escolha;</p>	

<p><b><u>Exemplo 2 (senão)</u></b></p> <p>Escolha (A)  Caso (1): Escreva ('Um');  Caso (2): Escreva ('Dois');  Senão  Escreva ('Três');  Fim_escolha;</p>	
<p><b>Em C# (exemplo 1)</b></p> <pre>switch (A) {     case 1: Console.WriteLine ("Um");             break;     case 2: Console.WriteLine ("Dois");             break;     case 3: Console.WriteLine ("Três");             break; }</pre>	<p><b>Em C#, Java (exemplo 2)</b></p> <pre>switch (A) {     case 1: Console.WriteLine ("Um");             break;     case 2: Console.WriteLine ("Dois");             break;     default:         Console.WriteLine ("Três");         break; }</pre>
<p><b>Em Python (exemplo 1)</b></p> <pre>match A:     case 1:         print ("Um")     case 2:         print ("Dois")     case 3:         print ("Três")</pre>	<p><b>Em Python (exemplo 2)</b></p> <pre>match A:     case 1:         print ("Um")     case 2:         print ("Dois")     case _:         print ("Três")</pre>

### 2.3.2.3. Exercícios propostos

- Indique a saída dos trechos de algoritmo (fluxograma) a seguir. Considere os seguintes valores: A = 2, B = 3, C = 5, D = 9. **Preste atenção na prioridade dos operadores.**



- Faça um algoritmo que leia dois números e apresente-os em ordem crescente. Use estrutura de desvio condicional simples.

- Faça um algoritmo que leia um número e diga se ele é divisível por 2 e por 3. Use estrutura de desvio condicional simples.
- Faça um algoritmo que leia um número inteiro e mostre uma mensagem indicando se este número é par ou ímpar e se é positivo ou negativo.
- Faça um algoritmo que leia um número e diga se ele é divisível por 5 ou por 7.
- Faça um algoritmo que leia 3 valores (A, B e C), encontra o valor maior e o escreve juntamente com a mensagem: "É o maior ".
- Faça um algoritmo que leia 2 valores A e B, verifica e escreve uma mensagem informando se “são múltiplos ou não são múltiplos”.
- Faça um algoritmo que leia 3 números inteiros e mostre o menor deles.
- Faça um algoritmo que calcule a média aritmética das 3 notas de um aluno e mostre, além do valor da média, uma mensagem de "Aprovado", caso a média seja igual ou superior a 6, ou a mensagem "Reprovado", caso contrário.
- Faça um algoritmo que dada a idade de um nadador, classifica-o em uma das seguintes categorias:

CATEGORIA	FAIXA ETÁRIA
infantil A	5 - 7 anos
infantil B	8 - 10 anos
juvenil A	11-13 anos
juvenil B	14-17 anos
Adulto	maiores de 18 anos

- Faça um algoritmo que leia o código de um aluno e suas três notas. Calcule a média ponderada do aluno, considerando que o peso para a maior nota seja 4 e para as duas restantes, 3. Mostre o código do aluno, suas três notas, a média calculada e uma mensagem "APROVADO" se a média for maior ou igual a 5 e "REPROVADO" se a média for menor que 5.
- Faça um algoritmo que auxilie o gerente a calcular o valor do crédito especial que seu banco concederá aos clientes dependendo do saldo médio do cliente no último ano. O algoritmo deve ler o saldo médio do cliente e calcular o valor do crédito de

acordo com a tabela abaixo e, em seguida, mostrar uma mensagem informando o saldo médio e o valor do crédito.

SALDO MÉDIO	PERCENTUAL
de 0 a 200	nenhum crédito
de 201 a 1000	20% do valor do saldo médio
de 1001 a 1600	30% do valor do saldo médio
acima de 1601	40% do valor do saldo médio

- Faça um algoritmo para automatizar o caixa da Lanchonete do senhor JUCA. Este algoritmo deve ler o código do item pedido, a quantidade e calcular o valor a ser pago por aquele lanche. O cardápio da lanchonete é o seguinte:

CÓDIGO	ESPECIFICAÇÃO	PREÇO UNITÁRIO (R\$)
1	Cachorro quente	8,10
2	Bauru simples	11.30
3	Bauru c/ovo	15,50
4	Hamburger	13.10
5	Cheeseburger	14.30
6	Refrigerante	5.00

- Um professor quer um programa que o auxilie. A calcular a média de seus alunos e que lhe permita também escolher o tipo de média que deseja calcular a partir de 3 notas. Assim sendo, faça um algoritmo que leia as notas, a opção de média escolhida pelo professor (vide tabela abaixo), calcule e mostre a média do aluno.

OPÇÃO	TIPO DE MÉDIA
1	aritmética
2	ponderada (3,3,4)
3	harmônica

- Cristiano comprou um microcomputador para controlar o rendimento diário de seu trabalho. Toda vez que ele traz um peso de peixes maior que o estabelecido pelo regulamento de pesca do estado dele deve pagar uma multa de R\$ 14,00 por quilo excedente. Cristiano precisa que você faça algoritmo que leia a variável P (peso de peixes), Peso máximo permitido (MP) e verifique se há excesso. Se houver, gravar na variável E (Excesso) e na variável M o valor da multa que ele deverá pagar.



- Faça um algoritmo que leia o código (C) de um trabalhador e o número de horas trabalhadas (NH) por ele, calcule o salário (S) sabendo-se que ele ganha R\$ 37,00 por hora. Quando o número de horas exceder a 44, calcule o excesso de pagamento armazenando-o na variável E, caso contrário zerar tal variável. No final do processamento imprimir o salário total (ST) e o salário excedente (SE).
- Faça um algoritmo que leia 4 (quatro) números; calcule o quadrado de cada um; se o valor resultante do quadrado do terceiro for  $\geq 100$ , imprima-o e finalize; caso contrário, imprima os valores lidos e seus respectivos quadrados.
- Faça um algoritmo que o índice de poluição medido e emita a notificação adequada aos diferentes grupos de empresas. Considere o seguinte problema: “A Secretaria de Meio Ambiente que controla o índice de poluição mantém 3 grupos de indústrias que são altamente poluentes do meio ambiente. O índice de poluição aceitável varia de 0,05 até 0,25. Se o índice sobe para 0,3 as indústrias do 1º grupo são intimadas a suspenderem suas atividades, se o índice crescer para 0,4 as indústrias do 1º e 2º grupo são intimadas a suspenderem suas atividades, se o índice atingir 0,5 todos os grupos devem ser notificados a paralisarem suas atividades. “
- Faça um algoritmo que leia 3 valores inteiros e positivos, encontre o maior e o menor valor lidos, calcule a média dos números lidos e mostre os resultados.
- Faça um algoritmo que leia a altura e o sexo de uma pessoa, calcule e mostre seu peso ideal. Considere as seguintes fórmulas: Para homens:  $(72.7 * h) - 58$ ; Para mulheres:  $(62.1 * h) - 44.7$  ( $h$  = altura).
- Faça um algoritmo que determine o grau de obesidade de uma pessoa, sendo fornecido o peso e a altura da pessoa. O grau de obesidade é determinado pelo índice da massa corpórea ( $\text{Massa} = \text{Peso} / \text{Altura}^2$ ), conforme ilustrado na tabela abaixo.

MASSA CORPÓREA	GRAU DE OBESIDADE
$< 26$	Normal
$26 \leq < 30$	Obeso
$\geq 30$	Obeso Mórbido

- Faça um algoritmo que leia as três notas de um aluno, calcule e exiba a sua média final e o seu conceito. Considere que a média final é calculada pela média aritmética das 3 notas e que o conceito é determinado com base na tabela abaixo.

MÉDIA FINAL	CONCEITO	Comentário
9,0	A	Excelente
7,0 e < 9,0	B	Ótimo
5 e < 7,0	C	Bom
< 5,0	D	Vai para recuperação

- O dono de uma certa empresa deseja aumentar o salário de seus funcionários. O reajuste deve obedecer a tabela abaixo. Levando isso em consideração, faça um algoritmo que leia o nome e o salário atual do empregado, e exiba o nome, o salário atual e o salário reajustado.

SALÁRIO ATUAL (R\$)	AUMENTO
0,00 a 3.000,00	30%
3.000,01 a 7.000,00	20%
acima de 7.000,00	10%

- Faça um algoritmo para calcular a conta final de um hóspede de uma certa pousada. Essa conta deve conter o nome do hóspede, o tipo do apartamento, o número de diárias usadas, o valor unitário da diária, o valor total das diárias, o valor do consumo interno, o subtotal, o valor da taxa de serviço e o total geral. Considere que:

- Devem ser lidos o nome do hóspede, o tipo do apartamento utilizado (A, B, C ou D), o número de diárias utilizadas pelo hóspede e o valor do consumo interno do hóspede;
- O valor da diária é determinado pela seguinte tabela:

TIPO DO APTO.	VALOR DA DIÁRIA (R\$)
A	250,00
B	150,00
C	100,00
D	75,00

- O valor total das diárias é calculado pela multiplicação do número de diárias utilizadas pelo valor da diária;
- O subtotal é calculado pela soma do valor total das diárias e o valor do consumo interno;

- c. O valor da taxa de serviço equivale a 10% do subtotal;
- d. A total geral resulta da soma do subtotal com a taxa de serviço.

### 2.3.3. Estrutura Repetição

Também conhecida como LOOPING ou laço de repetição, esta estrutura permite repetir um grupo/conjunto de ações até que uma dada condição seja satisfeita.

#### 2.3.3.1. Sintaxe

Como pode ser observado nas sintaxes mostradas a seguir, tem-se a estrutura de repetição com teste no início (ENQUANTO), com teste no final (REPITA) e laços contados (PARA).

##### Com teste no início

Na estrutura de repetição ENQUANTO uma sequência de comandos é executada quando o teste lógico resulta em verdadeiro e quando o teste lógico resulta em falso a sequência de comandos é encerrada.

Essa estrutura é usada quando não se sabe o número de vezes que deseja repetir um determinado trecho de um programa. Embora ela também possa ser usada quando este número é conhecido. Veja a sintaxe:

Pseudocódigo	Fluxograma	Em Python
<b>Enquanto</b> <condição> <b>faça</b> Comando_1; Comando_2; Comando_n; <b>Fim_enquanto;</b>	<pre>graph TD     Entry(( )) --&gt; Cond{&lt;Condição&gt;}     Cond -- Sim --&gt; C1[Comando 1]     C1 --&gt; C2[Comando 2]     C2 -.-&gt; Cn[Comando n]     Cn --&gt; Cond     Cond -- Não --&gt; Exit(( ))</pre>	<b>while</b> <condição>: comando_1 comando_2 comando_3
Em Pascal	Em C	Em C# (console)

<b>While</b> <condição> <b>do</b> <b>Begin</b> Comando_1; Comando_2; Comando_n; <b>End;</b>	<b>while</b> <condição> { Comando_1; Comando_2; Comando_n; }	<b>while</b> <condição> { Comando_1; Comando_2; Comando_n; }
--	---	---

## Com variável de controle

A estrutura de repetição PARA (com variável de controle) é usada quando o número de vezes que deseja repetir um trecho do programa for conhecido ou quando puder entrar com ele durante sua execução. Veja a sintaxe

Pseudocódigo	Fluxograma
<b>para</b> <variável> <b>de</b> <valor de inicio> <b>até</b> <valor de fim> <b>com incremento de</b> <passo> <b>faça</b> Comando_1; Comando_2; Comando_n; <b>Fim_para;</b>	<pre> graph TD     Init{&lt;variável&gt; ← &lt;inicial&gt;; &lt;fim&gt;; &lt;passo&gt;} --&gt; C1[Comando 1]     C1 --&gt; C2[Comando 2]     C2 -.-&gt; Cn[Comando n]     Cn --&gt; Init </pre>
Em Pascal	Em C, C# e Java
<b>FOR</b> <variável> := <inicio> <b>TO</b> <fim> <b>DO</b> <b>Begin</b> Comando_1; Comando_2; Comando_n; <b>End;</b>  Obs.: pode-se ter repetição decrementando. Nesse caso o usa DOWNTO ao invés de TO e o valor inicial e final devem ser invertidos. Deve-se iniciar com maior valor.	<b>for</b> (<variável> = <valor inicial>; <variável> <= <valor de fim>; <variável> ++) { Comando_1; Comando_2; Comando_n; }  Obs.: pode-se ter repetição decrementando, ou seja: <b>Variavel --</b> <b>Variavel ← variavel - 1</b>  Nesse caso o valor inicial deveria ser o maior valor
Em Python	Em Python
<b>for</b> <variável> <b>in</b> range (<fim>): Comando_1 Comando_2 Comando_n	<b>for</b> <variável> <b>in</b> range (<inicio>, <fim>, <passo>): Comando_1 Comando_2

	Comando_n
<b>Exemplo em python</b>  <b>for</b> cont <b>in</b> range (10): N1 = float(input("número 1")) N2 = float(input("número 2")) S = N1 + N2 print ("Soma igual a ", S)	<b>Exemplo em python</b>  <b>for</b> cont <b>in</b> range (0, 10 + 1, 1): print ("repetição número ", cont+1) if cont % 2 == 0: print (cont, " contador é par")  para imprimir apenas os pares de 0 a 10, bastaria fazer: <b>for</b> cont <b>in</b> range (0, 10 + 1, 2): print (cont)

### Com teste no fim (Repita Até que)

Na estrutura de repetição com teste no Fim a repetição do trecho do programa será executado para a condição falsa, caso a condição passe a ser verdadeira, a repetição é finalizada. Nesta estrutura, o trecho do programa é executado pelo menos uma vez.

Pseudocódigo	Fluxograma
<b>repita</b> Comando_1; Comando_2; Comando_n; <b>Até que</b> <condição>;  ----- Ou -----  <b>faça</b> Comando_1; Comando_2; Comando_n; <b>enquanto</b> <condição>;	<pre> graph TD     Start(( )) --&gt; C1[Comando 1]     C1 --&gt; C2[Comando 2]     C2 -.-&gt; Cn[Comando n]     Cn --&gt; Cond{&lt;Condição&gt;}     Cond --&gt; C1     Cond --&gt; Exit(( ))       </pre>
Em Pascal	Em C, C#, Java
<b>REPEAT</b> Comando_1; Comando_2; Comando_n; <b>UNTIL</b> <condição>;  Obs: nesse caso os comandos 1 a N são repetidos até que a condição seja <b>VERDADEIRA</b> .	<b>do</b> { Comando_1; Comando_2; Comando_n; } <b>while</b> <condição>;  Obs: nesse caso os comandos 1 a N são repetidos até que a condição seja <b>FALSA</b> .
Em python	
<b>Não foi localizada essa estrutura em Python</b>	

### 2.3.3.2. Exemplos

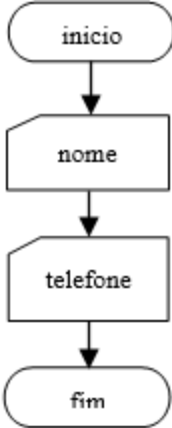
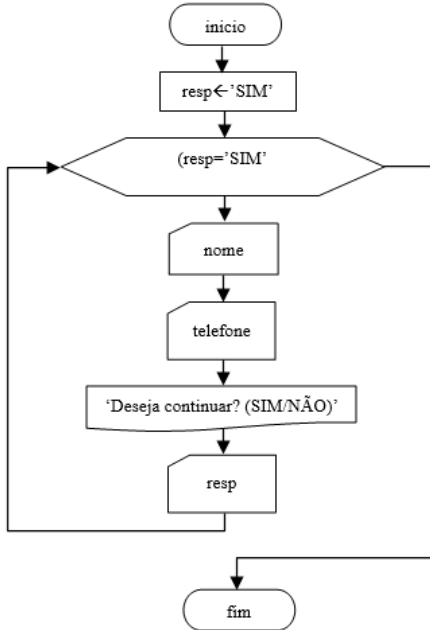
Veja alguns exemplos de algoritmos elaborados com e sem a estrutura de repetição. O primeiro exemplo (ALG31) mostra um algoritmo elaborado para ler nome e telefone. O segundo exemplo (ALG32) refere-se a um algoritmo que calcula a média global de 10 alunos de uma determinada sala de aula. Veja como é a resolução com e sem a utilização da estrutura de repetição.

O terceiro exemplo (ALG33) traz o algoritmo que imprime os números de 1 a 9 com incremento de 1. Neste exemplo foram empregadas as três estruturas de repetição para comparação das similaridades e diferenças.

Os dois primeiros exemplos (ALG31 e ALG32) demonstram a importância da estrutura de repetição e o terceiro (ALG33) mostra as diferenças e semelhanças entre as estruturas de repetição. Preste muita atenção nas diferenças. Por exemplo, na estrutura PARA não necessário inicializar o contador (variável N) fora do laço de repetição nem incrementar N dentro deste laço como ocorre com as estruturas ENQUANTO e REPITA. Na estrutura de repetição REPITA, como é codificada em PASCAL, a condição é o inverso do ENQUANTO, de modo que o laço de repetição só continua se a condição for falsa e o ENQUANTO só continua se a condição for verdadeira. Além disso, no REPITA o bloco de comandos dentro do laço de repetição é executado pelo menos uma vez antes de testar a condição. **Porém, em linguagens de programação que tem essa estrutura de repetição com teste no final, cuja sintaxe é `do..while`, ou seja, `FAÇA..ENQUANTO`, o laço de repetição só continua se a condição for verdadeira e o bloco de comandos a serem repetidos é executado pelo menos uma vez também.**

No segundo exemplo é utilizada uma variável para compor uma condição de parada para o looping, isto é, para as repetições. Ela é conhecida por **FLAG**.

ALG31 – Algoritmo para ler nome e telefone	
<u>SEM repetição</u>	<u>COM repetição</u>
Algoritmo Agenda; Var Nome, telefone : Literal; Início Leia(nome); Leia(telefone); Fim_algoritmo.	Algoritmo AgendaEnquanto; Var Nome, telefone, Resp : Literal; Início Resp 'S'; <b>Enquanto</b> (Resp='S') <b>faça</b> Leia(nome); Leia(telefone);

	<p>Escreva ('Deseja continuar? (S - continua)');          Leia (Resp);  <b>Fim_enquanto;</b>          Fim_algoritmo.</p>
 <pre> graph TD     inicio([inicio]) --&gt; nome[/nome/]     nome --&gt; telefone[/telefone/]     telefone --&gt; fim([fim])           </pre>	 <pre> graph TD     inicio([inicio]) --&gt; resp_init[resp&lt;-'SIM']     resp_init --&gt; cond1{resp='SIM'}     cond1 --&gt; nome[/nome/]     nome --&gt; telefone[/telefone/]     telefone --&gt; cond2{'Deseja continuar? (SIM/NÃO)'}     cond2 --&gt; resp[/resp/]     resp --&gt; cond1     cond1 --&gt; fim([fim])           </pre>
Em C# sem estrutura de repetição)	Em C# COM estrutura de repetição while e flag para controlar o looping)
<pre> 1. using System; 2. namespace AgendaEmCSharp; 3. { 4.     class Program 5.     { 6.         static void Main(string[] args) 7.         { 8.             string nome, telefone; 9.             Console.WriteLine("Nome: "); 10.            nome = Console.ReadLine(); 11.            Console.WriteLine("Telefone: "); 12.            telefone = Console.ReadLine(); 13.            Console.WriteLine("\n\npressione qualquer tecla para continuar "); 14.            Console.ReadKey(); 15.        } 16.    } 17. }           </pre>	<pre> 1. using System; 2. namespace AgendaWhileEmCSharp 3. { 4.     class Program 5.     { 6.         static void Main(string[] args) 7.         { 8.             string nome, telefone; 9.             char Resp = 'S'; 10.            while ((Resp == 'S')    (Resp == 's')) 11.            { 12.                Console.WriteLine("Nome: "); 13.                nome = Console.ReadLine(); 14.                Console.WriteLine("Telefone: "); 15.                telefone = Console.ReadLine(); 16.                Console.WriteLine("\n\n deseja continuar? (S - continua)"); 17.                Resp = Console.ReadLine(); 18.            } 19.            Console.WriteLine("\n\npressione qualquer tecla para continuar "); 20.            Console.ReadKey(); 21.        } 22.    } 23. }           </pre>
	Em Python
	<pre> 1. Resp = "S" 2. while (Resp=="S"): 3.     nome = str(input("Informe o nome")) 4.     telefone = str(input("Informe o telefone")) 5.     Resp = str(input("Deseja continuar? (S - continua)"))           </pre>



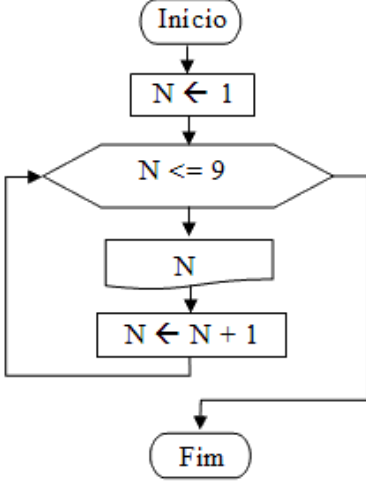
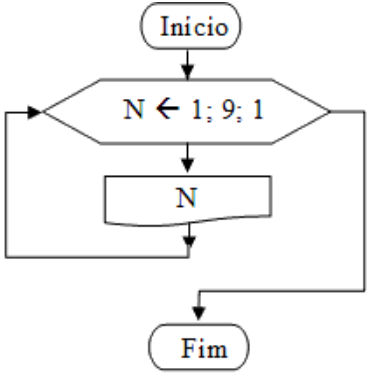
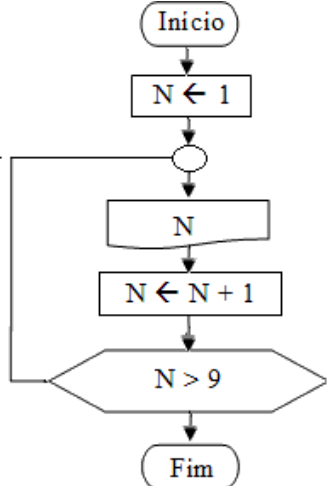
ALG32 – Algoritmo que calcula a média global de 10 alunos de uma determinada sala de aula. Veja como é a resolução com e sem a utilização da estrutura de repetição.

<u>Sem repetição</u>	<u>Com repetição</u>
<p><b>Algoritmo CalcMedia;</b></p> <p>Var</p> <p>N1,N2,N3,N4,N5,N6,N7,N8,N9,N10: real;</p> <p>Media, S: real;</p> <p><b>início</b></p> <p>leia (N1); leia (N2);</p> <p>leia (N3); leia (N4);</p> <p>leia (N5); leia (N6);</p> <p>leia (N7); leia (N8);</p> <p>leia (N9); leia (N10);</p> <p>S <math>\leftarrow</math> N1+N2+N3+N4+N5+ ... +N10;</p> <p>MEDIA <math>\leftarrow</math> S/10;</p> <p>Escreva (MEDIA);</p> <p>Fim.</p>	<p><b>Algoritmo CalcMedia;</b></p> <p>var</p> <p>N, SOMA, MEDIA: real;</p> <p>I: inteiro;</p> <p><b>início</b></p> <p>SOMA 0;</p> <p><b>para</b> I 1 até 10 <b>faça</b></p> <p>leia ( N ) ;</p> <p>SOMA SOMA + N;</p> <p><b>fim para;</b></p> <p>MEDIA SOMA/10;</p> <p>Escreva (MEDIA);</p> <p><b>Fim_algoritmo.</b></p>
<pre> graph TD     Inicio([Início]) --&gt; Input[/N1, N2, ..., N10/]     Input --&gt; S["S ← N1 + N2 + ... + N10"]     S --&gt; MediaCalc["MEDIA ← S/10"]     MediaCalc --&gt; Output[/MEDIA/]     Output --&gt; Fim([Fim]) </pre>	<pre> graph TD     Inicio([Início]) --&gt; Soma0["SOMA ← 0"]     Soma0 --&gt; Decision{"I &lt;= 1; 10; 1"}     Decision -- F --&gt; MediaCalc["MEDIA ← SOMA/10"]     MediaCalc --&gt; MediaOut[/MEDIA/]     MediaOut --&gt; Fim([Fim])     Decision -- V --&gt; Input[/N/]     Input --&gt; SomaAdd["SOMA ← SOMA + N"]     SomaAdd --&gt; Decision </pre>
<b>Em C# (sem repetição)</b>	<b>Em C# (com repetição)</b>
<pre> 1. using System; 2. namespace CalcMedia 3. { 4.     class Program 5.     { 6.         static void Main(string[] args) 7.         { 8.             double S, media, N1, N2, N3, N4, N5, N6, N7,               N8, N9, N10; 9.             Console.WriteLine("N1: "); 10.            N1 = double.Parse(Console.ReadLine()); 11.            Console.WriteLine("N2: "); </pre>	<pre> 1. using System; 2. namespace CalcMediaFor 3. { 4.     class Program 5.     { 6.         static void Main(string[] args) 7.         { 8.             double soma, media, N; 9.             soma = 0; 10. 11.             for (int i = 1; i &lt;= 10; i++) 12.             { </pre>

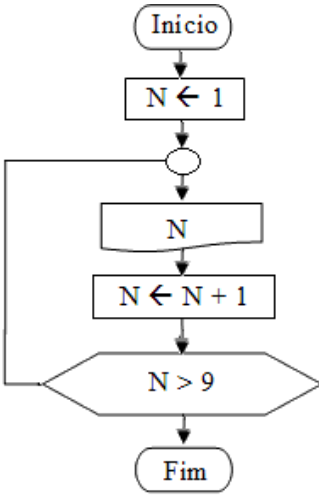
<pre> 12.     N2 = double.Parse(Console.ReadLine()); 13.     Console.WriteLine("\nN3: "); 14.     N3 = double.Parse(Console.ReadLine()); 15.     Console.WriteLine("\nN4: "); 16.     N4 = double.Parse(Console.ReadLine()); 17.     Console.WriteLine("\nN5: "); 18.     N5 = double.Parse(Console.ReadLine()); 19.     Console.WriteLine("\nN6: "); 20.     N6 = double.Parse(Console.ReadLine()); 21.     Console.WriteLine("\nN7: "); 22.     N7 = double.Parse(Console.ReadLine()); 23.     Console.WriteLine("\nN8: "); 24.     N8 = double.Parse(Console.ReadLine()); 25.     Console.WriteLine("\nN9: "); 26.     N9 = double.Parse(Console.ReadLine()); 27.     Console.WriteLine("\nN10: "); 28.     N10 = double.Parse(Console.ReadLine()); 29.     S = N1 + N2 + N3 + N4 + N5 + N6 + N7 + N8 +         N9 + N10; 30.     media = S/10; 31.     Console.WriteLine("\n\nmédia = {0}", media); 32.     Console.ReadKey(); 33. } 34. } 35. }</pre>	<pre> 13.     Console.WriteLine("\nDigite N: "); 14.     N = double.Parse(Console.ReadLine());  15.     soma = soma + N; 16. } 17.     media = soma / 10; 18.     Console.WriteLine("\n\nmédia = {0}", media); 19.     Console.ReadKey(); 20. } 21. } 22. }</pre>
Em Python SEM REPETIÇÃO	Em Python COM REPETIÇÃO
<pre> 1.     N1 = float(input("\nN1:")) 2.     N2 = float(input("\nN2:")) 3.     N3 = float(input("\nN3:")) 4.     N4 = float(input("\nN4:")) 5.     N5 = float(input("\nN5:")) 6.     N6 = float(input("\nN6:")) 7.     N7 = float(input("\nN7:")) 8.     N8 = float(input("\nN8:")) 9.     N9 = float(input("\nN9:")) 10.    N10 = float(input("\nN10:")) 11.    S = N1 + N2 + N3 + N4 + N5 + N6 +         N7 + N8 + N9 + N10; 12.    media = S/10; 13.    print("\n\nmédia = ", media);</pre>	<pre> 1.        soma = 0 2.        for cont in range (1, 10,1) 3.            N = str(input("\nDigite N: ")) 4.            soma = soma + N 5.        media = soma / 10 6.        print("\n\nmédia = {media}")</pre>

Já imaginou se não fossem apenas 10 notas e sim 50, 100 ou mais, como ficaria esse programa referente ao ALG32 sem repetição. Com repetição fica muito mais fácil.

ALG33 – Algoritmo que imprime os números de 1 a 9 utilização as 3 estruturas de repetição		
Pseudocódigo	Fluxograma	Em Python

<p>Algoritmo Num_1a9_Enquanto; Var N: Inteiro; Início   N 1;   <b>Enquanto</b> (N &lt;=9 ) <b>faça</b>     Escreva (N);     N N +1;   <b>Fim_enquanto</b>; Fim_algoritmo.</p>	 <pre> graph TD     Inicio([Início]) --&gt; N1[N ← 1]     N1 --&gt; Cond{N &lt;= 9}     Cond --&gt; PrintN[/N/]     PrintN --&gt; Nplus[N ← N + 1]     Nplus --&gt; Cond     Cond --&gt; Fim([Fim]) </pre>	<pre> 1. N = 1 2. while N &lt;=9: 3.     print(N) 4.     N = N + 1 </pre>
<p>Algoritmo Num_1a9_Para; Var N: Inteiro; Início   <b>Para</b> N 1 até 9 com   <b>incremento de 1 faça</b>     Escreva (N);   <b>Fim_Para</b>; Fim_algoritmo.</p>	 <pre> graph TD     Inicio([Início]) --&gt; Cond{N ← 1; 9; 1}     Cond --&gt; PrintN[/N/]     PrintN --&gt; Cond     Cond --&gt; Fim([Fim]) </pre>	<pre> for N in range (1,9,1):     print (N) </pre>
<p>Algoritmo Num_1a9_RepitaAte; Var N: Inteiro; Início   N 1;   <b>Repita</b>     Escreva (N);     N N +1;   <b>Até que</b> N &gt; 9; Fim_algoritmo.</p>	 <pre> graph TD     Inicio([Início]) --&gt; N1[N ← 1]     N1 --&gt; PrintN[/N/]     PrintN --&gt; Nplus[N ← N + 1]     Nplus --&gt; Cond{N &gt; 9}     Cond --&gt; PrintN     Cond --&gt; Fim([Fim]) </pre>	<p>Em C#</p> <pre> 1. using System; 2. namespace Num_1a9_DoWhile 3. { 4.     class Program 5.     { 6.         static void Main(string[] 7.             args) 8.         { 9.             int N; 10.            N = 1; 11.            do 12.            { 13.                Console.WriteLine("N = {0}", N); 14.                N = N + 1; 15.            } while (N &lt;= 9 ) 16.        } 17.    } 18. } </pre> <p># Em Python não encontrei a sintaxe dessa estrutura</p>

Como pode ser observada na codificação em C#, a condição colocada na estrutura **Repita..Até que** foi alterada ao usar a estrutura **do..while (faça ...enquanto)**, o que corresponde a:

<p>Algoritmo valores_4;</p> <p>Var N: Inteiro;</p> <p>Início</p> <p>    <math>N \leftarrow 1</math>;</p> <p>    <b>Faça</b></p> <p>        Escreva (N);</p> <p>        <math>N \leftarrow N + 1</math>;</p> <p>    <b>Enquanto</b> <math>N \leq 9</math>;</p> <p>Fim_algoritmo.</p>	 <pre> graph TD     Inicio([Início]) --&gt; N1[N ← 1]     N1 --&gt; Conector(( ))     Conector --&gt; N[N]     N --&gt; Nplus[N ← N + 1]     Nplus --&gt; Cond{N &gt; 9}     Cond -- Não --&gt; Conector     Cond -- Sim --&gt; Fim([Fim]) </pre>	<p>Em C#</p> <pre> 1. using System; 2. namespace CalcMediaFor 3. { 4.     class Program 5.     { 6.         static void Main(string[] args) 7.         { 8.             int N; 9.             N = 1; 10.            do 11.            { 12.                Console.WriteLine("N = {0}", N); 13.                N = N + 1; 14.            } while (N &lt;= 9 ) 15. 16.        } 17.    } 18. } </pre>
---	--	--

Voltando ao algoritmo ALG31, será mais útil se utilizarmos uma outra estrutura de dados, para armazenar todos os valores lidos, visto que as variáveis usadas comportam apenas um valor de cada vez que sobrepõe o anterior. A estrutura de dados ideal é o vetor. Pode ser estático ou dinâmico, como pode ser observado na codificação apresentada a seguir usando a estrutura de repetição Enquanto (While).

### 2.3.3.3. Exercícios propostos

1) Faça alguns algoritmos para resolver os seguintes problemas:

1. Imprimir os N primeiros números pares.
2. Determinar e imprimir o maior número em N números diferentes dados.
3. Encontrar e imprimir o menor dentre N valores lidos.
4. Calcular e imprimir o somatório de N valores lidos.
5. Calcular e imprimir a média aritmética de N Valores lidos.
6. Ler um número inteiro e positivo e apresentar o seu fatorial (exemplo: o fatorial de  $5! = 1*2*3*4*5 = 120$ ).

7. Auxiliar no controle de qualidade de uma fábrica de pisos a “PisosOtimos”, imprimindo os números das peças reprovadas, bem como o total de peças aprovadas e reprovadas no final do dia. Considere que a fábrica tem uma linha de produção capaz de produzir 800 peças por dia e para controlar a qualidade deve-se cadastrar o número da peça e o seu estado (aprovado ou reprovado).
8. Auxiliar na realização de uma pesquisa de opinião que visa saber se N pessoas gostaram ou não de um certo produto lançado no mercado. Os dados que devem ser registrados são o sexo do entrevistado e a sua resposta (Sim ou Não). O algoritmo precisa calcular e exibir as seguintes informações:
  - a) O número de entrevistados que responderam sim.
  - b) O número de entrevistados que responderam não.
  - c) A porcentagem de entrevistados do sexo feminino que responderam sim.
  - d) A porcentagem de entrevistados do sexo masculino que responderam não.
9. Auxiliar o senhor Fernando a verificar dentre os N bois de sua fazenda, qual é o número e o peso do boi mais gordo e do boi mais magro. Cada um dos bois dessa fazenda traz preso na sua orelha um cartão contendo seu número de identificação e seu peso. O algoritmo deverá ler os dados do cartão, realizar a verificação necessária e exibir a informação de interesse do dono da Fazenda.
  - a) Exibir de N números inteiros, quantos são pares, quantos são ímpares, quantos são positivos e quantos são negativos.
  - b) Calcular e exibir o valor de números, enquanto  $a \leq 1000$ .
10. Uma empresa decidiu fazer um recrutamento para preencher algumas vagas. Faça um algoritmo para cadastrar os candidatos, levando em conta o seguinte:
  - a) ler o número do candidato, a idade e o sexo, a experiência profissional (S - Sim/ N - Não);
  - b) mostrar a idade média dos candidatos;
  - c) mostrar o número total de candidatos e candidatas;

11. O cardápio da Lanchonete “Coma BEM” é o seguinte:

Código	Especificação	Preço Unitário(R\$)
1	Cachorro-quente	9,10
2	Bauru	10,30
3	Bauru c/ovo	13,50
4	Hamburger	14,10
5	Cheeseburger	16,30
6	Refrigerante	5,00

Escrever um algoritmo que vá lendo o código do item pedido, a quantidade e vá calculando o valor total a ser pago pelo pedido. **OBSERVAÇÃO:** Use uma flag para continuar no programa.

12. Faça um algoritmo para auxiliar um número indeterminado de contribuintes a fazer sua declaração do imposto de renda. Os cálculos devem ser feitos considerando as seguintes instruções:

- a) Desconto de R\$ 200,00 por dependente.
- b) Com base na renda líquida (renda anual menos descontos) é calculada a alíquota de contribuição (veja tabela a seguir);

Renda Líquida	Alíquota (%)
Até R\$ 1900,00	Isento
De R\$ 1901,00 até R\$ 7.000,00	5
De R\$ 7.001,00 até R\$ 15.000,00	15
Acima de R\$ 15.001,00	27

c) Na declaração devem constar os seguintes dados: nome do contribuinte, CPF, renda anual e número de dependentes.

- d) Use um flag para auxiliar no controle do laço de repetição.

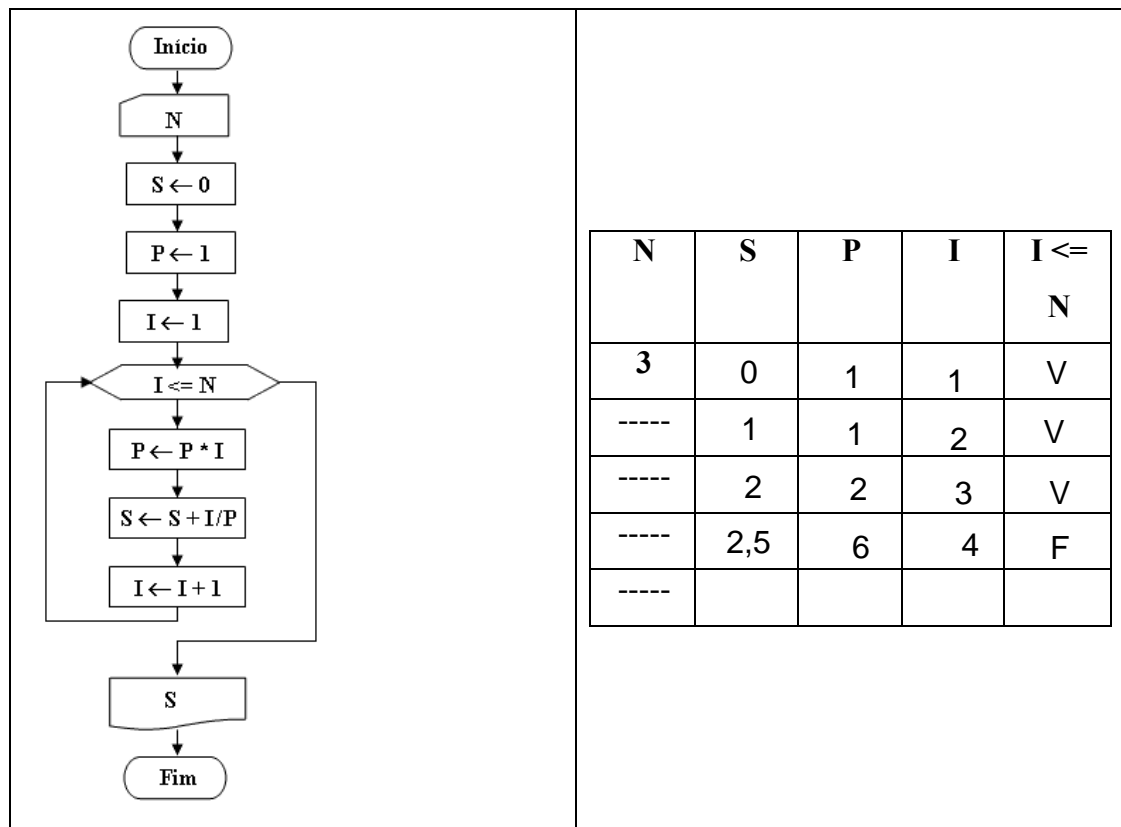
## 2.4. EXERCÍCIOS PROPOSTOS: simulado de prova

- Dado Fluxograma ao lado de um algoritmo que recebe dois números e mostra o menor, relacione o fluxograma com o pseudocódigo correto.

Pseudocódigo			Fluxograma
( )	(X)	( )	
Algoritmo	Algoritmo	Algoritmo	
Ex6_1;	Ex6_2;	Ex6_1;	
Var	Var	Var	
b:inteiro;	a,b:inteiro;	a,b:inteiro;	
Início	Início	Início	
Leia(a,b);	Leia(a,b);	Leia(a);	
Se a < b então	Se a < b	Se a < b então	
Escreva(a);	então	Escreva(a);	
Senão		Senão	
Escreva(b);	Escreva(a);	Escreva(b);	
Fim_se;	Senão	Fim_se;	
Fim.		Fim.	
	Escreva(b);		
	Fim_se;		
	Fim.		

2) Dado o fluxograma abaixo, faça o teste de mesa preenchendo a tabela de valores para N = 3.

Fluxograma	Teste de mesa
------------	---------------



3) Refazer o fluxograma do exercício anterior empregando a estrutura de repetição PARA.

4) Preencha as lacunas com V para as alternativas verdadeiras e F para as falsas.

V	Na estrutura de repetição PARA o contador deve ser declarado como inteiro.
F	ENQUANTO é uma estrutura de repetição com teste no final, e uma sequência de comandos só é executada quando o teste lógico resulta em verdadeiro.
V	Usa-se PARA normalmente quando o número de repetições é previamente conhecido.
F	A estrutura CASO/ESCOLHA pode ser utilizada sempre que o número de opções for maior que três.
V	Deve-se criar uma condição para executar comandos nas estruturas de repetição ENQUANTO ou REPITA.
F	Na estrutura CASO/ESCOLHA a condição testa a igualdade entre duas variáveis.
V	Dentro de uma estrutura de repetição é possível utilizar outras estruturas de repetição e/ou de decisão.
V	A estrutura condicional SE permite a escolha de uma sequência de comandos caso uma determinada condição (expressão lógica) é ou não verdadeira.
F	Na estrutura de repetição ENQUANTO o trecho de comandos é executado pelo menos uma vez antes de testar a condição.
V	Na estrutura de repetição REPITA o trecho de comandos será executado para a condição falsa.

5) Complete o algoritmo abaixo.



Algoritmo ExS5;

Var i: inteiro \_\_\_\_\_;

Início

i ← 30;

Enquanto (i > 15) faça

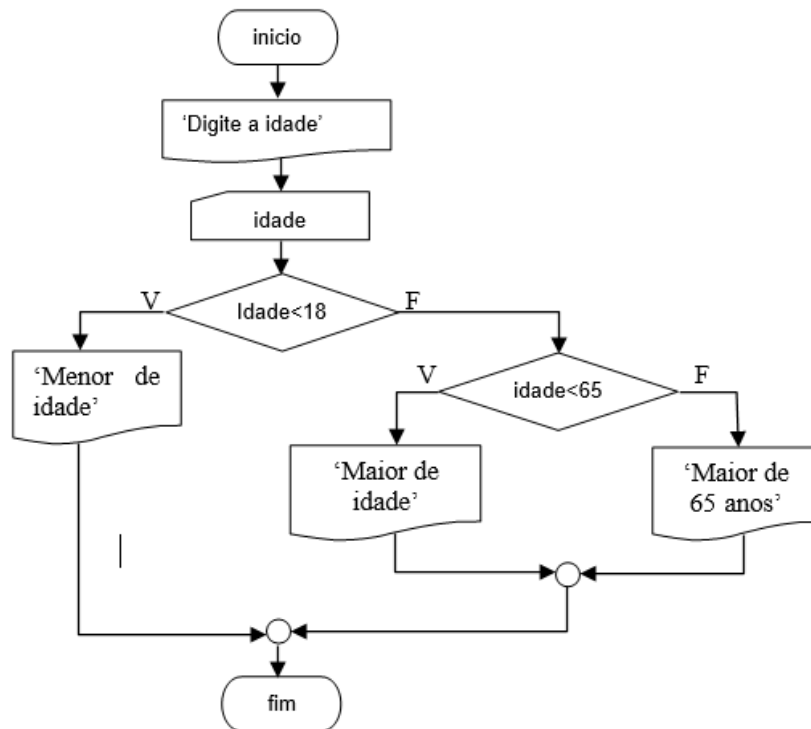
    Escreva (i);

    i ← i - 0,5;

Fim\_enquanto;

Fim\_programa.

6) Transcreva o fluxograma abaixo para o pseudocódigo.



7) Dado o fluxograma acima, acrescente uma estrutura de repetição para seja executada a verificação de idade para várias pessoas.

8) Simular o jogo de adivinhação: o jogador 1 escolhe um número entre 1 e 10; o jogador 2 insere números na tentativa de acertar o número escolhido pelo jogador 1. Quando ele acertar, o algoritmo deve informar que ele acertou o número escolhido pelo jogador 1 em x tentativas (quantidade de tentativas do jogador 2).

# REFERÊNCIAS BIBLIOGRÁFICAS

MANZANO, J.A.N.G., OLIVEIRA, J.F., **Algoritmos – Estudo Dirigido, Coleção P.D.**, Editora Érica, São Paulo, 1997.

MANZANO, J.A.N.G., OLIVEIRA, J.F., **Algoritmos – Lógica para desenvolvimento de Programação de Computadores**, Editora Érica, 15ª Ed., São Paulo, 2004.

FARRER, H, BECKER, C.G., FARIA, E.C., **Algoritmos Estruturados**, Editora LTC, 3ª Ed., 1999.