

UNIDADE I – CONCEITOS BÁSICOS DE PROGRAMAÇÃO

MÓDULO 1 – CONCEITOS DE PROGRAMAS E ALGORITMOS

01

CONCEITO DE PROGRAMA

Nos dias de hoje o computador passou a ser a principal ferramenta em diversas áreas de trabalho, isso ocorreu porque o computador é rápido, não cansa e não erra, mas não tem nenhuma iniciativa, nenhuma criatividade e depende totalmente das instruções e comandos que recebe.

A finalidade do computador é receber, manipular dados e gerar resultados. A simples tarefa de digitar um texto no Word é um exemplo dessa função: o usuário digita o texto puro, define a formatação, as margens, insere figuras e, com o auxílio do computador, gera um resultado bem mais apresentável do que se tivesse que fazer esse texto manualmente.

Outros exemplos como cálculos, ordenação e resumo de informações, são tarefas que o computador consegue fazer com velocidade e precisão e em bases de dados com milhões de registros. Atividades que antigamente necessitariam de muitas pessoas ou de muito tempo para concluí-las são rapidamente realizadas por um único computador.

Como dito anteriormente, as instruções devem ser passadas detalhadamente para o computador e a forma como essas instruções são passadas se dá através de **programas**.

Portanto, o programa de computador é uma espécie de idioma usado para repassarmos as ações que o computador deve executar.

Um programa nada mais é que um tipo de **algoritmo**, ou seja, **é uma sequência finita de ações que descrevem como um problema pode ser resolvido**. Sua particularidade é que suas operações são específicas para o computador e restritas ao conjunto de instruções que o processador pode executar. Podemos considerar esse conjunto de instruções como a primeira linguagem de programação do computador, também chamada de linguagem de máquina.



02

Desenvolver um programa consiste em processar dados, isto é, recebê-los por um dispositivo de entrada (teclado, mouse, scanner), realizar operações com os mesmos (processamento) e gerar respostas que serão expressas em um dispositivo de saída (impressora, monitor) ou guardar os resultados em dispositivos de armazenamento (disco rígido, pen-driver, CDs).



As etapas para o desenvolvimento de um programa são:

- **análise** → Na análise estuda-se o problema para definir os dados de entrada, o processamento e os dados de saída.
- **algoritmo** → No algoritmo são utilizadas ferramentas para descrever o problema.
- **codificação** → Na codificação o algoritmo é transformado em códigos da linguagem de programação.

03

CONCEITO DE ALGORITMO

Segundo Cormen:

Algoritmo é qualquer procedimento computacional bem definido que toma algum valor ou conjunto de valores como entrada e produz algum valor ou conjunto de valores como saída.

Portanto, um algoritmo é uma sequência de passos computacionais, que transformam a entrada na saída. Também podemos visualizar um algoritmo como uma ferramenta para resolver um problema

computacional. O enunciado do problema especifica em termos gerais o relacionamento entre a entrada e a saída desejada. O algoritmo descreve um procedimento computacional específico para se alcançar esse relacionamento da entrada com a saída.

Em resumo, podemos dizer que algoritmo é um conjunto finito de regras que prevê uma sequência de operações destinadas à solução de um problema específico.

A importância do algoritmo está no fato de termos de especificar uma sequência de passos lógicos para que o computador possa executar a tarefa desejada pelo usuário do programa. Lembre-se de que o computador não tem vontade própria, isto é, faz apenas o que o algoritmo determina.

Características de um algoritmo



- 1) Deve ter início e fim (sequência finita de regras).
- 2) Não deve dar margem à dupla interpretação (não deve ser ambíguo).
- 3) Deve ter capacidade de receber dados de entrada e processá-los.
- 4) Deve ter capacidade de gerar informações de saída.
- 5) Deve ser efetivo (não basta gerar um resultado, o resultado deve estar correto).

04

Alguns autores consideram que o algoritmo pode ser considerado a solução de um problema. Mas, atenção:

No caso do algoritmo, os caminhos que levam à solução de um problema são muitos. Um determinado problema pode ser resolvido com mais de um algoritmo, assim como podemos ter algoritmos enormes ou pequenos para resolver o mesmo problema.

Um algoritmo que resolve um problema não pode ser considerado certo ou errado, mas ele pode ser mais ou menos demorado na resolução do problema. Um algoritmo pode ser especificado em linguagem comum, como um programa de computador, ou mesmo como um projeto de *hardware*.

Como exemplo, elencamos o algoritmo para ordenar números, onde tem uma entrada de números (45, 23, 85, 77) e deverá, depois de o algoritmo ser executado, ter uma saída (23, 45, 77, 85) ou seja, um algoritmo para reordenar a sequência de números.

O melhor algoritmo para uma determinada aplicação depende, entre outros fatores, do número de itens a serem ordenados, da extensão em que os itens já estão ordenados de algum modo, de possíveis restrições sobre os valores de itens e da espécie de dispositivo de armazenamento a ser usado: memória principal, discos ou fitas.

Ordenamos uma sequência de números pequenos, mas vamos agora citar o exemplo dado por Cormen sobre o **projeto Genoma**, que tem como objetivos:

- identificar todos os 100.000 genes do DNA humano,
- determinar as sequências dos 3 bilhões de pares de bases químicas que constituem o DNA humano,
- armazenar essas informações em bancos de dados e
- desenvolver ferramentas para análise de dados.

Cada uma dessas etapas exige algoritmos sofisticados. Um algoritmo eficiente pode gerar uma economia de tempo, dinheiro, tanto humano quanto da máquina, à medida que mais informações podem ser extraídas.

05

FORMAS DE REPRESENTAÇÃO DE UM ALGORITMO

Algoritmos podem ser representados pelas técnicas: descrição narrativa, fluxograma (diagramas de blocos), como veremos a seguir.

- **Descrição narrativa**

Na descrição narrativa utiliza-se a língua portuguesa para descrever a sequência finita de regras dos algoritmos.



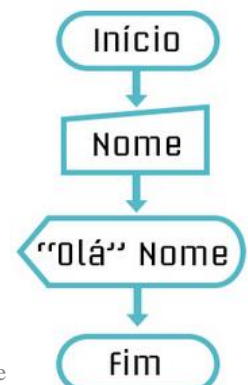
06

- **Fluxograma**

Os fluxogramas são uma apresentação do algoritmo em formato gráfico. Cada ação ou situação é representada por uma caixa. Tais caixas são ligadas por setas, que indicam o fluxo das ações.

Algumas caixas especiais indicam a possibilidade de o fluxo seguir caminhos distintos, dependendo de certas situações que podem ocorrer durante a execução do algoritmo.

Tomadas de decisões são indicadas por caixas especiais, possibilitando ao fluxo de ações tomar caminhos distintos. Também há representações para o início do



algoritmo e para o seu final, para que o fluxo do algoritmo possa ser determinado do seu princípio até o seu término. A representação de algoritmos por meio de fluxogramas tem uma série de vantagens, que serão apresentadas mais adiante.



Uma das buscas da humanidade é desenvolver uma máquina que pense e tome decisões como o ser humano. Apesar de avanços essa máquina ainda não existe. O exemplo representado pelo fluxograma acima obtém o nome do usuário e escreve uma saudação.

Descrição Narrativa

PASSO 1 – Receber o nome do usuário.

PASSO 2 – Mostrar a mensagem “Olá” nome do usuário.

Resumindo, os fluxogramas (diagramas de blocos) consistem na utilização de símbolos para representação da sequência finita de regras dos algoritmos.

Veja quais são as vantagens e desvantagens da utilização do fluxograma.

VANTAGENS	DESVANTAGENS
 <ol style="list-style-type: none"> 1) Ferramenta bem conhecida. 2) Figuras dizem mais do que palavras. 3) Padrão mundial. 	 <ol style="list-style-type: none"> 1) Não oferece recursos para descrever os dados da manipulação e saída. 2) Complica-se à medida que o algoritmo cresce.

07

- **Linguagem algorítmica**

A linguagem algorítmica ou pseudocódigo consiste na definição de uma pseudolinguagem de programação, cujos comandos são escritos em língua portuguesa, para representar estruturas algorítmicas.

O pseudocódigo visa a trazer o máximo possível de benefícios, tentando diminuir o ônus da utilização da linguagem de programação. A ideia é que o pseudocódigo seja um passo intermediário entre a linguagem natural e a de programação. Ao contrário da linguagem de programação, o pseudocódigo tem um grau de rigidez sintática intermediária, mas pode utilizar o idioma nativo.



As linguagens de programação são construídas utilizando no geral palavras em inglês. O pseudocódigo é independente de linguagem e pode ser traduzido de uma forma quase direta para uma gama de linguagens de programação. Um pseudocódigo bastante conhecido no Brasil é o **Portugol**, que

apresenta uma aceitação grande e tem suas razões para isso. Após a construção do algoritmo em pseudocódigo, é necessário que mais um passo seja executado para que o algoritmo possa ser transformado em programa, é necessário que seja transcrito para uma **linguagem de programação**. É a transformação do pseudocódigo em código de alguma linguagem de programação real.

Para representarmos nossos algoritmos, utilizaremos uma adaptação do Portugol, que incluirá algumas construções e mecanismos que julgamos adequados para o melhor aprendizado do programador, procurando facilitar o processo de construção do algoritmo.

08

Observe as vantagens e desvantagens da utilização da linguagem algorítmica.

VANTAGENS	DESVANTAGEM
 <ol style="list-style-type: none"> 1) Usa a Língua Portuguesa como base. 2) Pode-se definir os dados que serão estruturados 3) Passagem quase imediata da linguagem algorítmica para uma linguagem de programação. 	 <ol style="list-style-type: none"> 1) Exige a definição de uma linguagem não real para execução do trabalho.

Exemplo:

```
#incluir <biblioteca>
principal( )
inicio
```

```
    literal Nome;
    escreva("Digite seu nome");
    leia(Nome);
    escreva("Olá ", Nome);
```

```
fim
```

O deslocamento para a direita (identação) de uma instrução significa que esta está subordinada à instrução anterior. Esse recuo facilita muito a compreensão e manutenção do algoritmo e dos códigos fontes em uma linguagem de programação.

09

exemplo:
 $2 \times 6 = 12$

Na sua forma mais simples a multiplicação é uma maneira de se adicionar uma quantidade finita de números iguais. O resultado da multiplicação de dois números é chamado produto. Os números sendo multiplicados são chamados de coeficientes ou operandos,

e individualmente de multiplicando e multiplicador. O exemplo a seguir tem o objetivo de elaborar um algoritmo que mostre o resultado da multiplicação de dois números, utilizando descrição narrativa, fluxograma e linguagem algorítmica.

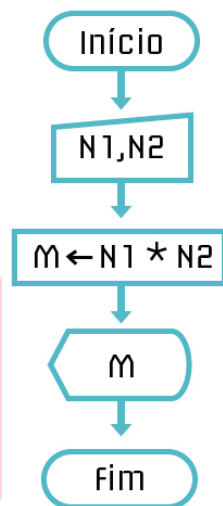
Descrição Narrativa

PASSO 1 – Receber os dois números que serão multiplicados.

PASSO 2 – Multiplicar os números.

PASSO 3 – Mostrar o resultado obtido na multiplicação.

Fluxograma



Linguagem Algorítmica

#incluir <biblioteca>

principal()

inicio

inteiro N1, N2, M;

escreva("Digite dois números");

leia(N1, N2);

$M \leftarrow N1 * N2;$

escreva("Multiplicação = ", M);

fim

TÉCNICAS PARA RESOLVER UM ALGORITMO

Como visto anteriormente, algoritmo é um conjunto regras e operações que visam solucionar um determinado problema. Um mesmo problema pode ser resolvido de formas diferentes, a solução pode variar desde a forma mais simples até a mais complexa. A colaboração de membros da equipe ou colegas, se possível, é sempre vantajosa. A seguir listamos algumas técnicas que podem facilitar a elaboração do algoritmo.

- 1) Não se limite apenas às regras conhecidas: transgrida-as.
- 2) Não use apenas as suas próprias ideias: utilize as vantagens dos membros da equipe.
- 3) Adote uma definição flexível de solução: não se limite apenas às conhecidas.
- 4) Coopere com os membros da equipe para que o grupo saia ganhando.

Métodos para construção de algoritmos

- 1) Ler o enunciado, destacando pontos importantes.
- 2) Definir os dados de entrada (dados que serão fornecidos pelo usuário).
- 3) Definir o processamento (cálculos que serão realizados pelo algoritmo).
- 4) Definir os dados de saída (dados gerados após o processamento, que serão apresentados como respostas).
- 5) Construir o algoritmo.
- 6) Testar o algoritmo realizando simulações.

Depois de ter sido realizada a análise do problema (métodos de 1 a 4), pode-se utilizar a descrição narrativa para descrever a sequência dos passos do algoritmo, depois elaborar o fluxograma para facilitar o entendimento do fluxo de execução e, por fim, representar o algoritmo em linguagem algorítmica. Esta sequência permite o melhor entendimento do algoritmo. Com o aperfeiçoamento da análise de problemas e dependendo da complexidade de um algoritmo, a fase de elaboração da descrição narrativa e do fluxograma pode ser suprimida.

DESCRIÇÃO NARRATIVA

Veja, a seguir, exemplos de utilização da descrição narrativa na construção do algoritmo. As demais formas de representação serão apresentadas nas próximas unidades.

Exemplo A



Um sanduíche é um tipo de comida que consiste em duas fatias de pão, entre as quais é colocada carne, queijo e outros tipos de condimentos.

Elaborar um algoritmo para fazer um sanduíche utilizando a descrição narrativa

Solução:

PASSO 1: Pegar um pão.

PASSO 2: Cortar o pão ao meio.

PASSO 3: Pegar a maionese.

PASSO 4: Passar a maionese no pão.

PASSO 5: Pegar e cortar alface e tomate.

PASSO 6: Colocar alface e tomate no pão.

PASSO 7: Pegar o hambúrguer.

PASSO 8: Fritar o hambúrguer.

PASSO 9: Colocar o hambúrguer no pão.

Exemplo B

Dinheiro é o meio usado na troca de bens, na forma de moedas ou notas (cédulas), usado na compra de bens, serviços, força de trabalho, divisas estrangeiras ou nas demais transações financeiras, emitido e controlado pelo governo de cada país, que é o único que tem essa atribuição.

Elaborar um algoritmo para sacar dinheiro em um terminal de banco 24 horas, utilizando descrição narrativa.

Solução

12



Descrição narrativa para sacar dinheiro em um terminal de banco 14 horas:

PASSO 1: Ir até um banco 24 h.

PASSO 2: Colocar o cartão.

PASSO 3: Digitar a senha.

PASSO 4: Solicitar a quantia desejada.

PASSO 5: Se o saldo for maior ou igual à quantia desejada, sacar; senão, mostrar mensagem de impossibilidade de saque.

PASSO 6: Retirar o cartão.

PASSO 7: Sair do banco 24 h.

Exemplo C

Um bolo é um tipo de alimento a base de massa de farinha, geralmente doce e cozido no forno. Os bolos são um dos componentes principais das festas, por vezes ornamentados artisticamente e ocupando o lugar central da mesa.



Elaborar um algoritmo para preparação de um bolo de chocolate utilizando descrição narrativa.

Solução

Descrição narrativa para preparação de um bolo de chocolate:

PASSO 1: Bata quatro claras em neve.
 PASSO 2: Adicione duas xícaras de açúcar.
 PASSO 3: Adicione duas xícaras de farinha de trigo.
 PASSO 4: Adicione quatro gemas.
 PASSO 5: Adicione uma colher de fermento.
 PASSO 6: Adicione duas colheres de chocolate.
 PASSO 7: Bata por três minutos.
 PASSO 8: Unte uma assadeira com margarina e farinha de trigo.
 PASSO 9: Coloque o bolo para assar durante vinte minutos em temperatura média.

13

Exemplo D

Um pneu é um artefato circular feito de borracha, o qual pode ser inflado com ar ou com água. Utilizado por veículos em geral, como carros de passeio, caminhões, tratores, bicicletas, carros de mão etc.

Elaborar um algoritmo, utilizando descrição narrativa, para troca de um pneu furado.

Solução

Descrição narrativa para troca de um pneu furado:

PASSO 1: Verifica qual pneu está furado.
 PASSO 2: Pega o macaco, a chave de roda e o estepe.
 PASSO 3: Afrouxa os parafusos.
 PASSO 4: Com o macaco levanta o carro.
 PASSO 5: Retira os parafusos.
 PASSO 6: Substitui o pneu furado.

PASSO 7: Recoloca os parafusos.
 PASSO 8: Com o macaco desce o carro.
 PASSO 9: Aperta os parafusos.
 PASSO 10: Guarda o macaco, a chave de roda e o pneu furado.

Exemplo E

Carta é uma comunicação manuscrita ou impressa devidamente acondicionada e endereçada a uma ou várias pessoas.

Elaborar um algoritmo utilizando descrição narrativa com a sequência de ações para enviar uma carta pelo correio.

Solução



Descrição narrativa para enviar uma carta pelo correio:

PASSO 1: Escreva a carta.
 PASSO 2: Coloque a carta no envelope.
 PASSO 3: Feche o envelope.
 PASSO 4: Preencha o envelope.
 PASSO 5: Coloque o selo no envelope.
 PASSO 6: Vá até o correio.
 PASSO 7: Entregue a carta ao funcionário do correio.
 PASSO 8: Pague ao funcionário do correio.
 PASSO 9: Confira o troco.

14

REFINAMENTO SUCESSIVO

O refinamento sucessivo de um algoritmo consiste na lapidação do mesmo. Nesta fase, são adicionadas tarefas antes omitidas. São apresentadas soluções mais bem elaboradas.

O refinamento sucessivo é realizado após a construção do primeiro protótipo do algoritmo.

Em algoritmo, um problema pode ser resolvido de maneiras diferentes, porém, deve gerar sempre a mesma resposta.

Exemplo F

$$x^2 - 5x + 6 = 0$$

Apresentar as etapas algorítmicas necessárias para a solução da equação do 2º grau $x^2 - 5x + 6 = 0$.

$$ax^2 + bx + c = 0 \Rightarrow \begin{cases} \Delta = b^2 - 4ac \\ x = \frac{-b \pm \sqrt{\Delta}}{2a} \end{cases}$$

Solução:

PASSO 1: Determinação dos coeficientes a, b, c.

$$a = 1, b = -5, c = 6$$

PASSO 2: Cálculo do delta.

$$\Delta = b^2 - 4ac \leftarrow \Delta = (-5)^2 - 4.1.6 = 25 - 24 = 1$$

PASSO 3: Cálculo das raízes

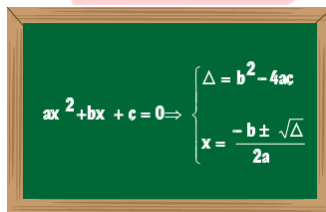
$$x = \frac{-b \pm \sqrt{\Delta}}{2a} = \frac{-(-5) \pm \sqrt{1}}{2.1} = \frac{5 \pm 1}{2}$$

PASSO 4: Apresentação da solução

$$x_1 = 2 \text{ ou } x_2 = 3.$$

15

Vejamos mais um exemplo.



$$ax^2 + bx + c = 0 \Rightarrow \begin{cases} \Delta = b^2 - 4ac \\ x = \frac{-b \pm \sqrt{\Delta}}{2a} \end{cases}$$

Demonstrar um algoritmo utilizando descrição narrativa que solucione uma equação do 2º grau.

Solução:

PASSO 1: Informe os coeficientes a, b, c.

PASSO 2: Calcule o delta.

PASSO 3: Calcule as raízes.

PASSO 4: Apresente os resultados na tela.

Abaixo seguem as etapas algorítmicas necessárias para a solução da equação do 2º grau $x^2 - 5x + 7 = 0$.

Atenção: para uma melhor aprendizagem, antes de clicar sobre cada passo para ver a solução, tente fazer o seu algoritmo.

Solução:

PASSO 1: Determinação dos coeficientes a, b, c.

$$a = 1, b = -5, c = 7$$

PASSO 2: Cálculo do delta.

$$\Delta = b^2 - 4ac \leftarrow \Delta = (-5)^2 - 4.1.7 = 25 - 28 = -3$$

PASSO 3: Verificação do sinal de delta.

$$\Delta < 0.$$

PASSO 4: Apresentação da solução

“Não existem raízes reais.”

16

Observe mais um exemplo de refinamento sucessivo.

Fazer o refinamento sucessivo do algoritmo que solucione uma equação do 2º grau utilizando descrição narrativa.

Solução:

PASSO 1: Informe os coeficientes a, b, c.

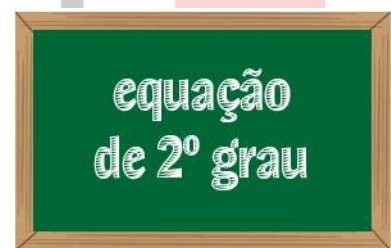
PASSO 2: Calcule o delta.

PASSO 3: Verifique se o delta é maior ou igual a zero

PASSO 4: Se o delta for maior ou igual a zero, calcule as raízes.

PASSO 5: Se o delta for menor do que zero, informe que não existem raízes reais.

PASSO 6: Apresente os resultados na tela.



RESUMO

Neste módulo apresentamos os conceitos básicos de programa, a importância do programa como ferramenta para execução de atividades. Apresentamos também os conceitos básicos de algoritmo, suas características e sua importância para o desenvolvimento de programas. Descrevemos as técnicas e métodos que auxiliam na análise e resolução de algoritmos, os quais devem seguir os passos:

- 1) Ler o enunciado, destacando pontos importantes.
- 2) Definir os dados de entrada (dados que serão fornecidos pelo usuário).
- 3) Definir o processamento (cálculos que serão realizados pelo algoritmo).
- 4) Definir os dados de saída (dados gerados após o processamento, que serão apresentados como respostas).
- 5) Construir o algoritmo.
- 6) Testar o algoritmo realizando simulações.

Apresentamos, também, uma introdução a respeito das formas de representação de algoritmos: Narração Descritiva, Fluxograma e Linguagem Algorítmica, exemplificando a descrição narrativa aplicada a diferentes tipos de problemas. Com estes conceitos é possível entender a importância de um programa de computador, a sua construção através de algoritmos e conhecer inicialmente as principais formas de representação de algoritmos.

UNIDADE I – CONCEITOS BÁSICOS DE PROGRAMAÇÃO

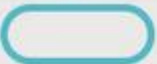








MÓDULO 2 – FLUXOGRAMAS

01

SIMBOLOGIA

O fluxograma é uma forma universal de representação, pois se utiliza de figuras geométricas para ilustrar os passos a serem seguidos para a resolução dos problemas. Também é chamado de **diagrama de blocos**.

Os principais diagramas de bloco encontram-se na tabela abaixo.

SÍMBOLO	NOME	AÇÃO
	Terminador	Representa o início e o fim do fluxograma.
	Entrada Manual	Representa a entrada de dados para as variáveis por meio do teclado.
	Entrada via Cartão	Símbolo muito utilizado antigamente para representar entrada de dados por meio de cartão.
	Processo	Representa a execução de operações ou ações como cálculos aritméticos, atribuição de valores a variáveis, abertura e fechamento de arquivo, entre outras.
	Decisão/ Repetição	Representa uma ação lógica que resultará na escolha de uma das sequências de instruções, ou seja, se o teste lógico apresentar o resultado 'verdadeiro', realizará uma sequência e, se o teste lógico apresentar o resultado 'falso', realizará outra sequência.
	Exibição	Representa a saída de informações (dados ou mensagens) por meio do monitor de vídeo ou outro dispositivo visual de saída de dados.
	Documento	Símbolo muito utilizado antigamente para representar a saída de informações por meio de impressão.
	Seta de orientação de fluxo	A sequência do fluxograma pode ser desenvolvida horizontalmente ou verticalmente.
	Conector	Utilizado para interligar partes do fluxograma ou pra desviar o fluxo corrente para um determinado trecho do programa.

O fluxograma é utilizado para organizar o raciocínio lógico a ser seguido para a resolução de um problema ou para definir os passos para a execução de uma tarefa, dentro das estruturas sequenciais, de decisão ou de repetição. Cada ação ou situação é representada por uma caixa. Tomadas de decisões são indicadas por caixas especiais, possibilitando ao fluxo de ações tomar caminhos distintos.

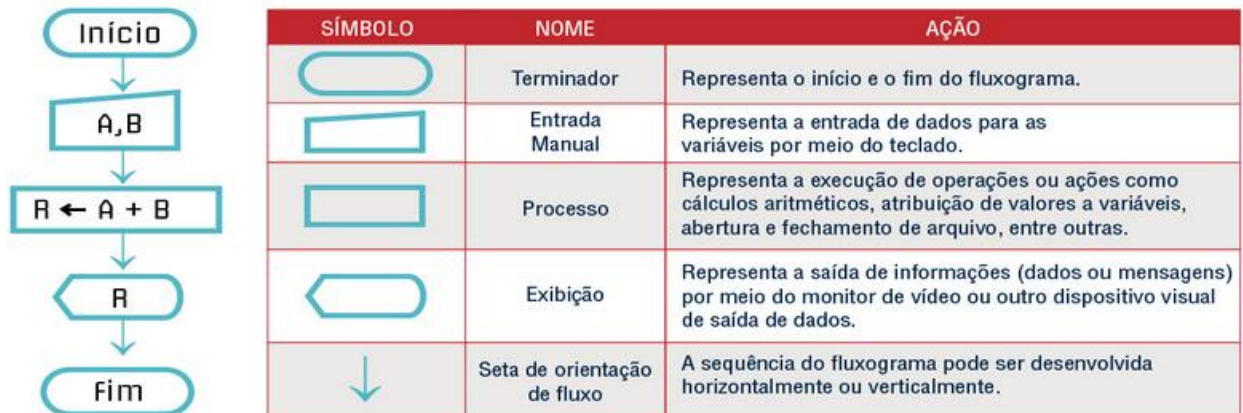
02

ESTRUTURAS DE CONDIÇÃO

Num processo geral de execução de um algoritmo implementado, a execução começa na primeira linha e vai avançando sequencialmente, executando o código, linha após linha, até chegar no final. Entretanto, frequentemente surge a necessidade de colocar instruções dentro de um programa, as quais só serão executadas caso alguma condição específica aconteça. Para esta finalidade, a maioria das linguagens dispõe de **estruturas de condição** para realizar esta tarefa, o que pode variar de acordo com a característica de cada linguagem.

O exemplo a seguir mostra um algoritmo utilizando fluxograma para ler dois números e apresentar no vídeo a soma entre eles. Ao lado do fluxograma, inserimos a simbologia, de modo que você possa identificar cada etapa do fluxo e habituar-se ao uso dos símbolos.

FLUXOGRAMA:



03

ESTRUTURAS DE DECISÃO

A estrutura de decisão permite que, durante o processamento, o algoritmo possa decidir qual será a próxima ação, entre uma ou mais opções, dependendo do valor lógico de um determinado teste (condição).

As estruturas de decisão podem ser simples ou compostas.

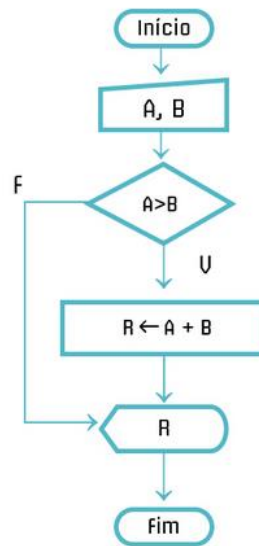
- **Estrutura de decisão simples**

É uma expressão que deverá retornar um valor de verdadeiro (V) ou de falso (F), e caso o resultado dessa expressão seja **verdadeiro**, será executado o bloco de comandos que está dentro da estrutura. Caso seja **falso**, a execução do programa ignora o bloco de comando e continua na linha seguinte à estrutura de condição.

se <expressão-lógica> então:

<bloco de comandos>

fim-se



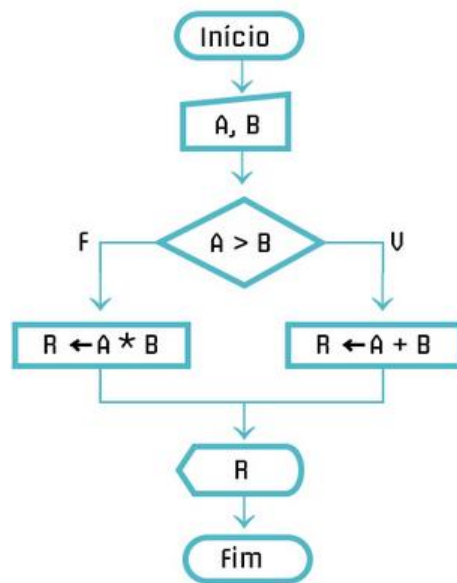
04

ESTRUTURA DE DECISÃO COMPOSTA

Oferece a possibilidade de executarmos uma determinada ação ou comando se o resultado da expressão lógica for verdadeiro e de executarmos uma ação diferente se o resultado da expressão lógica for falso.

O exemplo a seguir mostra um algoritmo utilizando fluxograma para ler dois números e apresentar a soma entre eles se o primeiro for maior, se o segundo número for maior, é feita a multiplicação desses números.

O fluxo com a identificação V executa caso o teste seja **verdadeiro**. Caso o teste resulte em **falso**, o fluxograma executa a outra parte do algoritmo.



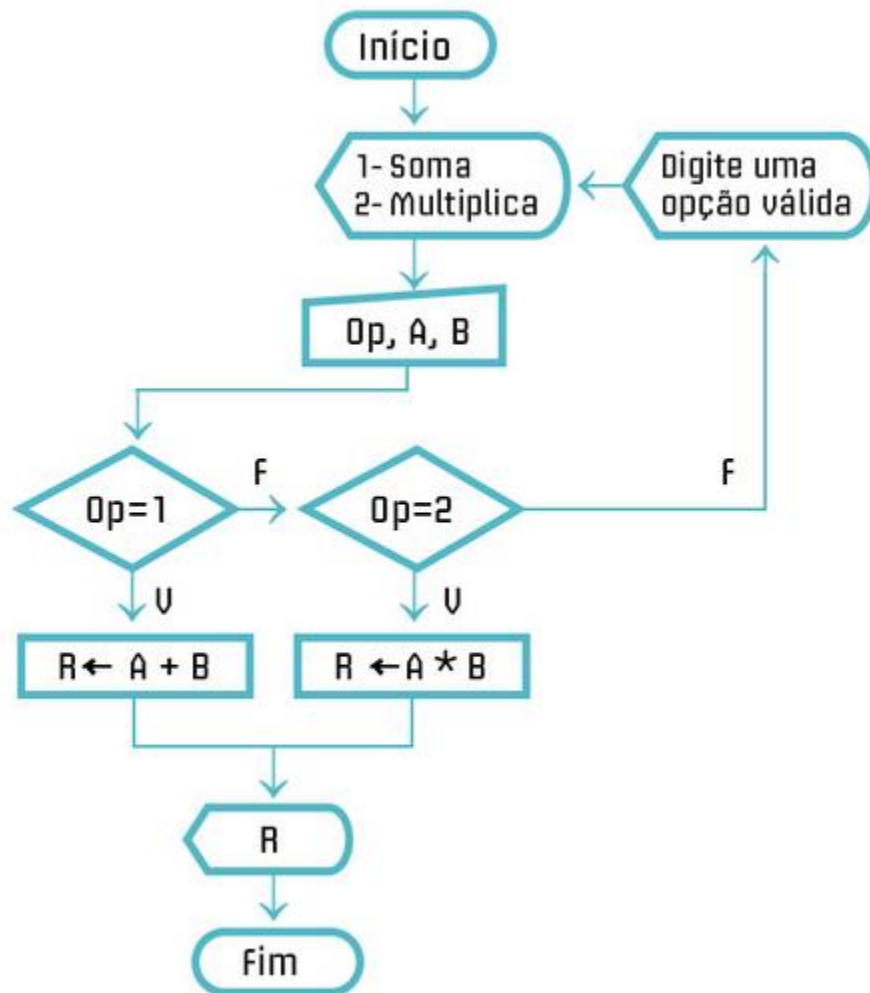
E se os números forem iguais, qual fluxo será executado? Qual é o resultado do teste $A > B$, verdadeiro ou falso?

Resposta

Resposta: Se os números forem iguais, o resultado de $A > B$ é falso.

05

Estruturas de decisão podem ser encadeadas. Por exemplo, se o usuário desejar escolher uma opção entre várias.



Se o usuário digitar qualquer coisa diferente de 1 ou 2, o que acontece?

Resposta

Resposta: O fluxograma mostra a mensagem “Digite uma opção válida” e volta para a tela inicial.

06

ESTRUTURAS DE REPETIÇÃO

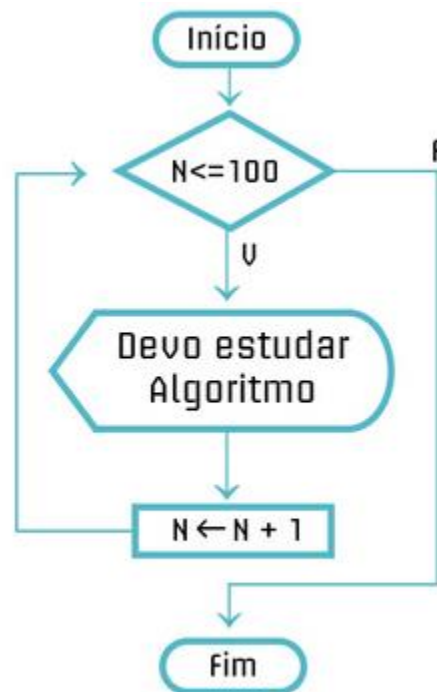
A estrutura de repetição permite que durante o processamento, o algoritmo repita uma mesma tarefa um número determinado de vezes.

Em outras palavras, a estrutura de repetição propõe uma maneira de repetir um conjunto de procedimentos até que determinado objetivo seja atingido. Apenas se o objetivo for atingido, a repetição se encerra.

Vejamos alguns exemplos.

Demonstrar um algoritmo que escreve 100 vezes: “Devo estudar Algoritmo”.

O exemplo a seguir apresenta uma estrutura de repetição com uma condição de controle, essa condição é uma expressão lógica, que é executada, determinando se a repetição prossegue ou não. Em suma, a repetição acontecerá até que chegue ao valor 100.

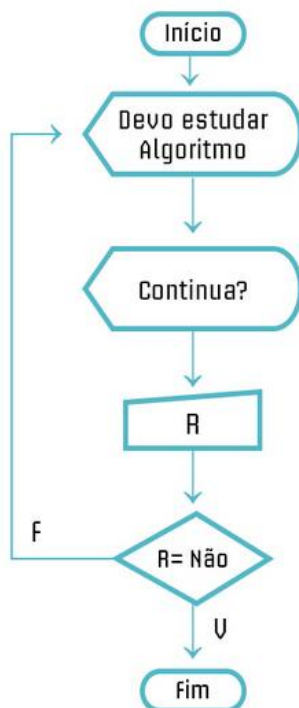


No fluxo anterior, N funciona como um contador de passos, após a exibição da mensagem, N passa a ter seu valor acrescido de 1 ($N \leftarrow N+1$), por isso, quando N for maior que 100, a expressão $N \leq 100$ será falsa e o fluxograma vai para o fim.

07

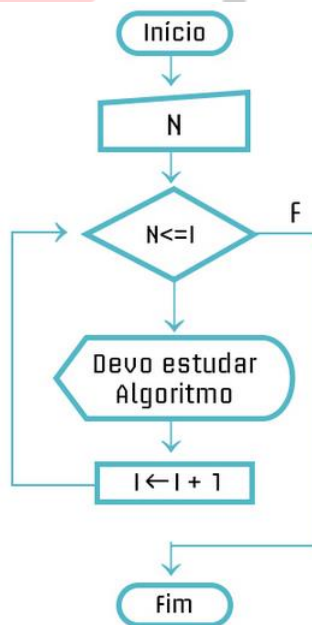
As estruturas de repetição permitem que os passos sejam executados por uma determinada quantidade de vezes ou até que se obtenha uma determinada condição.

No exemplo a seguir a mensagem “Devo estudar Algoritmo” será exibida até que o usuário escolha a opção Não.



08

No exemplo a seguir a mensagem “Devo estudar Algoritmo” será exibida por quantas vezes o usuário informar.

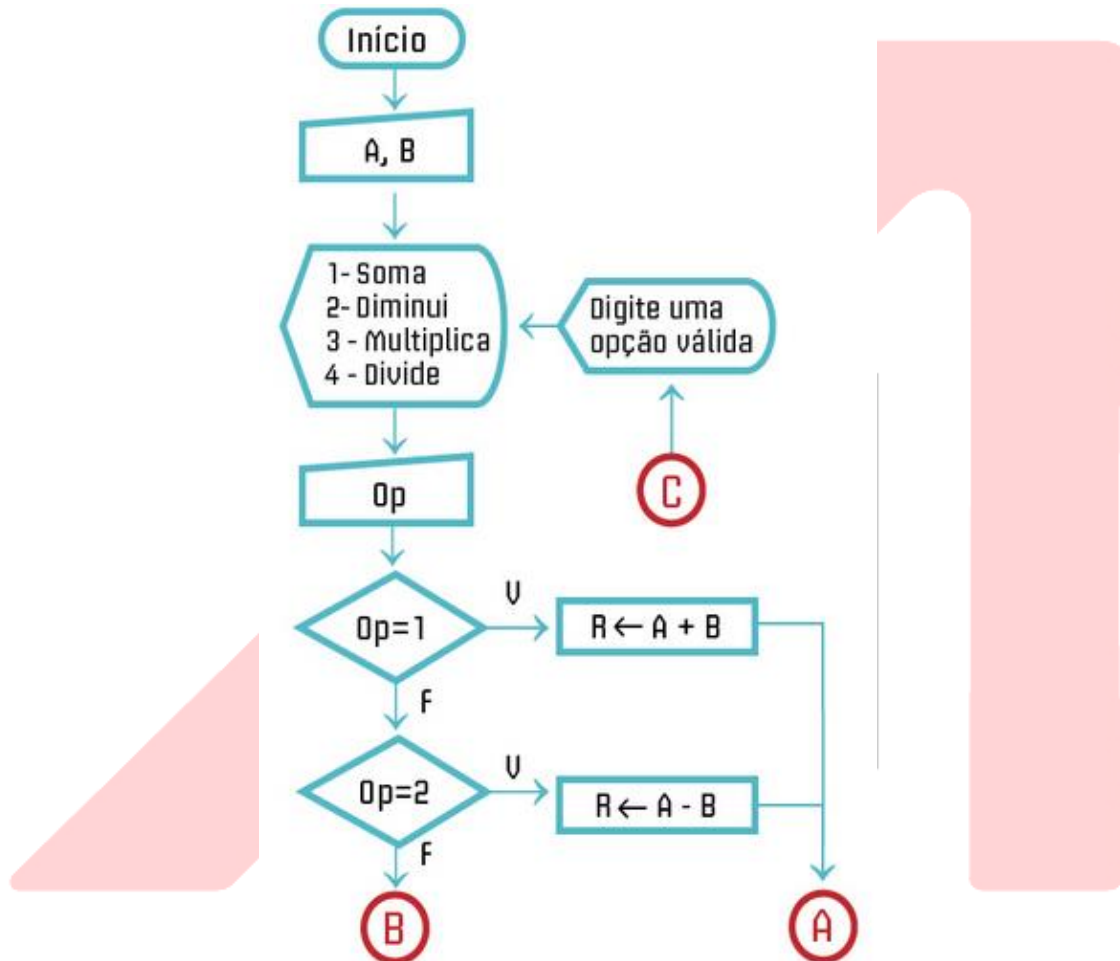


As estruturas de decisão e repetição serão explicadas com mais detalhes na próxima unidade quando serão aplicadas nos algoritmos.

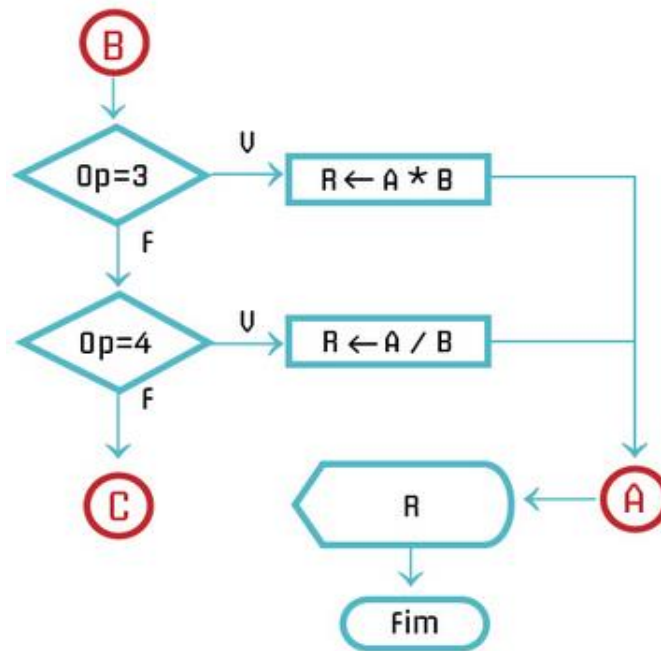
CONECTORES

Os conectores são usados quando o algoritmo se torna extenso e sua representação fica complexa. Nestes casos o conector é usado para ligar diversas partes do fluxograma.

Conectores podem ser identificados por números ou letras



Os conectores A, B e C ligam o fluxograma a outra parte que está logo a seguir.



10

RESUMO

Neste módulo apresentamos o Fluxograma ou Diagrama de Blocos como forma de representação do algoritmo, sua finalidade e o significado dos símbolos. Explicamos também a função de cada símbolo mostrando alguns exemplos. As estruturas de decisão, que podem ser simples e compostas, e as estruturas de repetição (que permitem que os passos sejam executados por uma determinada quantidade de vezes ou até que se obtenha uma determinada condição), com e sem blocos de comando, também foram demonstradas e exemplificadas, os exemplos mostram em quais situações cada estrutura pode ser utilizada. Foi mostrada ainda a importância da utilização dos conectores para organizar fluxogramas extensos visando facilitar sua visualização. Com essas informações é possível representar um algoritmo de forma gráfica com o objetivo de facilitar o entendimento e a visualização da ordem de execução dos passos do algoritmo.

UNIDADE I – CONCEITOS BÁSICOS DE PROGRAMAÇÃO

MÓDULO 3 - CONSTANTES, VARIÁVEIS, TIPOS E PSEUDOCÓDIGO

01

CONSTANTES E VARIÁVEIS

Dentro de um algoritmo podemos encontrar basicamente duas classes diferentes de dados, conhecidas como **constantes** e **variáveis**.

Uma **constante** é quando seu valor não se altera ao longo do tempo em que o algoritmo é executado,

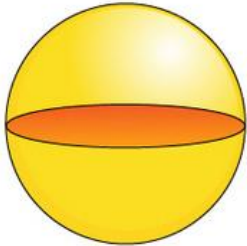
ou seja, permanece o mesmo valor do início até o final da execução.

Um dado que pode ter seu valor alterado durante a execução do programa é tido como uma **variável**.

Uma constante tem valor fixo e inalterável, podendo ser um caractere, uma cadeia de caracteres ou um número:

- uma **constante caractere** é escrita entre aspas simples,
- uma **constante cadeia de caracteres** entre aspas duplas e
- **constantes numéricas** como o número propriamente dito.

Uma constante muito comum no meio matemático e, por consequência, no meio computacional, é o número $\pi = 3,141592...$



O número carrega consigo uma propriedade geométrica intrínseca das formas circulares, tanto bidimensionais, como a circunferência, quanto tridimensionais, como a esfera. Este fato faz deste número uma **constante física**.

02

CONSTANTE GRAVITACIONAL



Normalmente uma pessoa diz que está pesando 60 quilos, no entanto, o que ela quer dizer é que sua massa é de 60 quilos.

Peso é uma força que é calculada pela massa multiplicada pela aceleração gravitacional da Terra.

Neste caso a **constante gravitacional** é utilizada para saber qual a força que um corpo de 60 Kg exerce na superfície do planeta Terra. O cálculo do peso é feito através da fórmula:

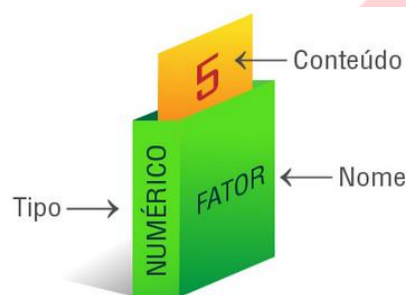
$$\text{Peso} = \text{Massa} * g;$$

onde $g = 9,78 \text{ m/s}^2$

03

Uma variável representa a posição da memória do computador. Um algoritmo recebe dados que precisam ser armazenados no computador, para que sejam utilizados no processamento. O armazenamento de dados é feito na memória do computador.

A variável possui nome e tipo, além disso, o conteúdo que ela armazena pode variar ao longo do tempo, durante a execução de um programa.



04

TIPOS DE DADOS

Para otimizar os recursos computacionais disponíveis e acelerar o processamento das informações, é necessário que os dados sejam armazenados no computador de acordo com o tipo de informação que se deseja disponibilizar, bem como o tipo de operação que será executada. Assim, os dados são representados pelas informações a serem processadas por um computador.

A seguir são definidos os tipos de dados mais comuns encontrados na maioria das linguagens de programação.

Nas variáveis, os tipos básicos de dados mais utilizados são:

- numérico,
- lógico e
- literal ou caractere.

Vejamos cada um a seguir.

- **Dado numérico**

Os dados numéricos dividem-se em dois grupos: inteiros e reais.

• Dados inteiros →

Podem ser positivos ou negativos e não possuem parte decimal. Os dados inteiros quando armazenados na memória do computador ocupam 2 bytes.

São exemplos de dados numéricos **inteiros**:

o 0
o 10
o -50
o 10000

• Dados Reais →

Podem ser positivos ou negativos e possuem parte decimal limitada, sendo por isto denominados dados **ponto flutuante**. Os dados reais quando armazenados na memória do computador ocupam 4 bytes. Os dados reais seguem a notação da língua inglesa, ou seja, a parte decimal é separada da parte inteira por um (ponto) e não por uma (vírgula).

São exemplos de dados numéricos **reais**:

o 3.1415
o 120.47
o -200
o 4096.333

05

• Dado lógico

Os dados lógicos também são chamados de dados booleanos. Podem assumir os valores lógicos verdadeiro ou falso. Os dados lógicos, quando armazenados na memória do computador ocupam um byte.

Os dados lógicos assumem os valores lógicos definidos pela Lógica Matemática, isto é, pelo princípio da não contradição e pelo princípio do terceiro excluído.

Segundo o princípio da não contradição, uma proposição não pode ser verdadeira e falsa ao mesmo tempo. E pelo princípio do terceiro excluído, não há uma terceira opção para que a proposição assuma. Sendo assim, ou a proposição assume o valor lógico VERDADEIRO ou assume o valor lógico FALSO.

São exemplos de dados lógicos:

- V ou Verdadeiro
- F ou Falso

Para saber mais sobre os princípios da não contradição e do terceiro excluído, acesse http://www.academia.edu/3658762/Principios_de_identidade_nao-contradicao_e_terceiro_excluido

06

- **Dado literal**

O dado literal ou caractere é formado por um único caractere ou por uma cadeia de caracteres.



Os dados literais podem ser as letras maiúsculas, as letras minúsculas, os números (não podem ser usados para cálculo) e os caracteres especiais (&, #, @, ?, +). O dado literal quando armazenado na memória do computador, ocupa 1 byte para cada caractere.

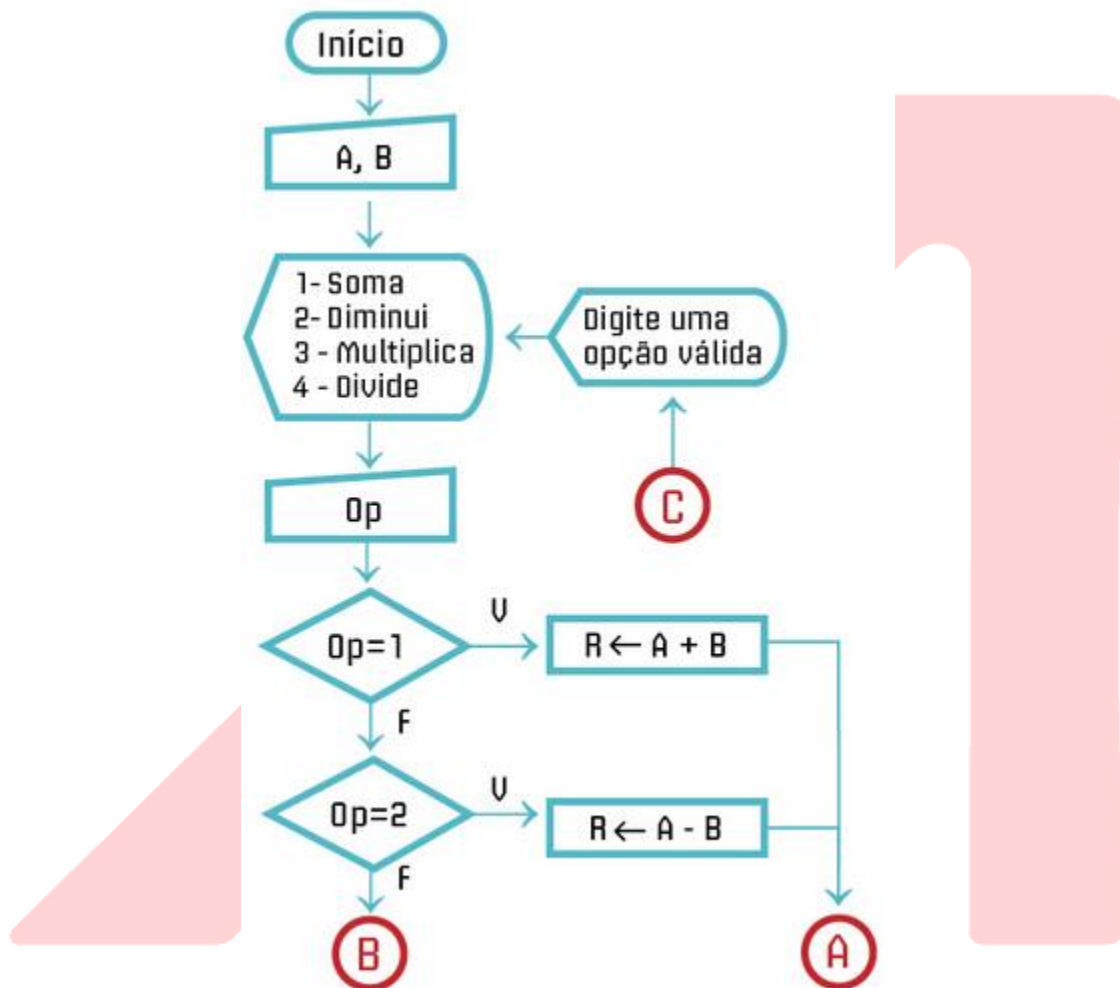
Agora, observe na tabela abaixo, alguns exemplos de tipos de dados, incluindo o dado literal.

DADO	TIPO DE DADO
"João Silva"	Literal
"576.221.786-01"	Literal
1999	Inteiro
11450300	Inteiro
"11.450-300"	Literal
9.78	Real
1.00001	Real
"Azul"	Literal
"São Paulo"	Literal
1000.574	Real
Falso	Lógico
Verdadeiro	Lógico
-10	Inteiro
"11 98118-1010"	Literal
"25/12/2010"	Literal
"JXY - 5060"	Literal
1024	Inteiro
"A"	Literal
3.141592	Real

07

Em um fluxograma, as variáveis identificadas no símbolo de entrada manual recebem as informações digitadas pelo usuário. No símbolo de processo a operação indica qual variável recebe o resultado do cálculo, conforme o exemplo a seguir.

	Entrada Manual	Representa a entrada de dados para as variáveis por meio do teclado.
	Processo	Representa a execução de operações ou ações como cálculos aritméticos, atribuição de valores a variáveis, abertura e fechamento de arquivo, entre outras.





No trecho do fluxograma anterior podemos identificar as variáveis: A, B, Op e R.

08

VARIÁVEIS DE ENTRADA E DE SAÍDA

É possível imaginar uma variável como sendo um local onde se pode colocar qualquer valor do conjunto de valores possíveis para o tipo definido para a variável.

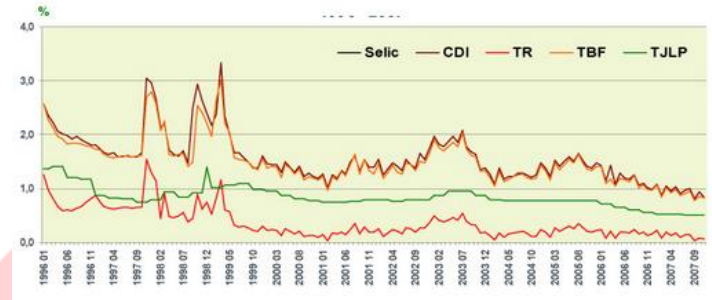
As variáveis podem ser denominadas de entrada ou de saída.

 VARIÁVEIS DE ENTRADA	 VARIÁVEIS DE SAÍDA
Correspondem àquelas que o usuário deve fornecer o conteúdo para que o processamento seja realizado.	São as que armazenam os resultados processados a partir das variáveis de entrada.

Observe os exemplos a seguir.

Exemplo 1

Quais são as variáveis de entrada e de saída de um algoritmo que recebe um valor de capital, uma taxa de juros e prazo, em meses, e calcula a prestação de empréstimo?



Resposta:

Dados de entrada:

Valor de capital
Taxa de juros
Prazo, em meses

Dados de Saída:

Prestação

09

Exemplo 2



Quais são as variáveis de entrada e de saída do algoritmo que calcula o valor líquido de uma fatura, dado o valor bruto e percentual de desconto?

Resposta:

Dados de entrada:

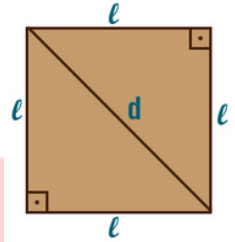
Valor bruto
Percentual de Desconto

Dados de Saída:
Valor Líquido

Exemplo 3

Quais as variáveis de entrada e de saída do algoritmo que calcula o quadrado de um número real?

Resposta:



Dados de entrada:
Número real

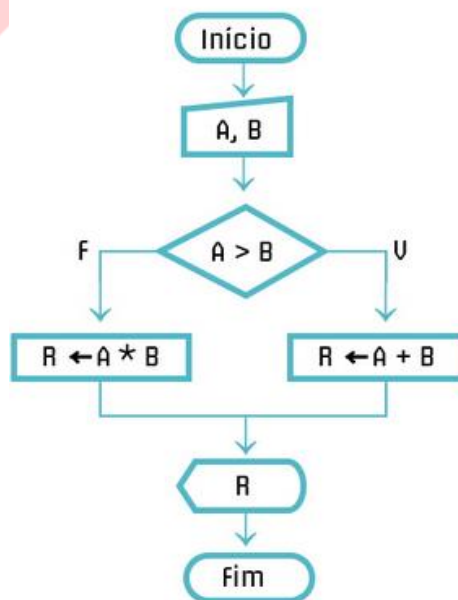
Dados de Saída:
Quadrado do número real

10

Exemplo 4

No fluxograma a seguir, quais os dados de entrada e de saída?

Resposta:



Dados de entrada: A e B

Dados de Saída: R

11

PALAVRAS RESERVADAS E IDENTIFICADORES

Palavras reservadas são palavras que pertencem à sintaxe de um **pseudocódigo** ou uma linguagem de programação. Não podem ser utilizadas para mais nada, além do propósito inicial a que se destinam.

Exemplos de palavras reservadas em um pseudocódigo: inicio, declare, fim, literal, real, se, então, entre outras.

Exemplos de palavras reservadas da linguagem de programação C: case, char, if, else, entre outras.

Os **identificadores** são os nomes das variáveis, dos programas, das constantes, das funções e dos procedimentos.

Regras para formação dos identificadores:

- 1) Caracteres que podem ser utilizados: números, letras maiúsculas, letras minúsculas e o caractere sublinhado.
- 2) Primeiro caractere deve ser sempre uma letra ou o caractere sublinhado.
- 3) Não são permitidos os espaços em branco e caracteres especiais (@, \$, +, -, %, !).
- 4) Não se pode utilizar palavras reservadas.

Exemplos:

Na tabela a seguir estão indicados os identificadores de variáveis válidos e os inválidos.

1_codigo	Inválido, o identificador não pode começar com número.
codigo_1	Válido.
_1_codigo	Válido.
!_codigo	Inválido, o identificador não pode conter caracteres especiais, no caso "!".
codigo+1	Inválido, o identificador não pode conter caracteres especiais, no caso "+".
codigo um	Inválido, o identificador não pode conter espaço em branco.
aaaa	Válido.
a1234	Válido.
escrever	Inválido, o identificador não pode ser uma palavra reservada.
real	Inválido, o identificador não pode ser um tipo de dado, é uma palavra reservada.

12

PSEUDOCÓDIGO – PORTUGOL

O **pseudocódigo** é um padrão textual de representação de um algoritmo, pode ser denominado também de **português estruturado** ou **Portugol**.

Seus comandos e operações são semelhantes aos da linguagem de programação, o que facilita o entendimento da estrutura desses comandos e ajuda a traduzir o código para a linguagem de programação.

A representação de um algoritmo na forma de pseudocódigo é a seguinte:

Algoritmo Nome_Do_Algoritmo

Variáveis

Declaração das variáveis

Procedimentos

Declaração dos procedimentos

Funções

Declaração das funções

Início

Corpo do Algoritmo

Fim

Exemplo

Algoritmo “Média”

Variáveis

N1, N2, Média: **real**

Início

Escreva (“informe o primeiro valor”)

Leia (N1)

Escreva (“informe o segundo valor”)

Leia (N2)

Média <- (N1+N2)/2

Se Média >= 7 **Então**

Escreva (“Aprovado!!”)

Senão

Escreva (“Reprovado!!”)

Fim se

Fim.

As unidades que seguem utilizarão em seus exemplos o padrão da linguagem algorítmica PORTUGOL. O PORTUGOL foi desenvolvido com o intuito de aproximar o aluno da linguagem C, ainda durante a fase inicial do aprendizado, evitando dificuldades de aprendizagem com as diferenças entre as sintaxes do C e do PORTUGOL.

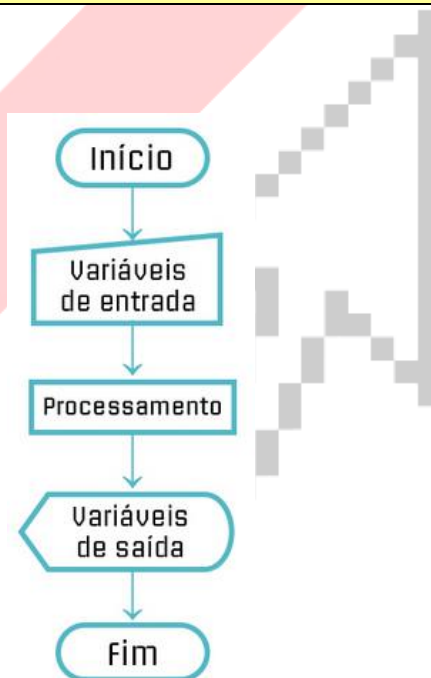
13

Apresentamos abaixo a sequência básica da sintaxe de um algoritmo escrito em PORTUGOL.

Sintaxe:

<pre>#incluir <biblioteca> principal() Inicio declare variáveis; bloco_de_comandos; Fim</pre>	<pre>// instrução que inclui bibliotecas de funções pré-definidas // instrução que caracteriza um nome ao algoritmo // instrução que marca o início da estrutura sequencial // instrução que declara as variáveis e os tipos de dados // instruções que definem os comandos sequenciais // instrução que marca o término da estrutura sequencial</pre>
--	---

Fluxograma:



Em fluxograma só são declaradas as variáveis de entrada, mas em linguagem algorítmica todas as variáveis são declaradas.

14

DECLARAÇÃO DE VARIÁVEIS E CONSTANTES

Pode-se considerar as variáveis como “apelidos” para endereços de memória que armazenam dados. Todas as variáveis de um programa C devem ser declaradas antes de serem usadas para que seja alocada memória para as mesmas.

As variáveis são declaradas conforme os tipos de dados utilizados (numérico, literal ou lógico). Ao longo do desenvolvimento do programa e do algoritmo, o dado será manipulado através do nome do seu identificador, que poderá ser uma constante ou uma variável. Assim, o primeiro passo para utilizarmos os dados é a nomeação do seu identificador e a definição do seu tipo ou do seu valor. A definição dos dados em algoritmos também é conhecida como **declaração**.

Um identificador, variável ou constante, declarado com um determinado tipo de dados ficará restrito a armazenar valores daquele tipo específico (inteiro, real, caractere, lógico).

Na maioria dos casos, se houver uma tentativa de atribuir a um identificador um tipo diferente daquele para o qual ele foi definido, o sistema não irá funcionar adequadamente.

Exemplo de declaração de variáveis e constantes:

```
var
marca, modelo: caractere;
ano: inteiro;
preco: real;
vendido: lógico;
constante
PI=3.141592654;
MAXIMO=100;
```

As variáveis receberam um TIPO específico e as constantes receberam um VALOR.

15

Como já vimos, para que os programas manipulem valores, estes devem ser armazenados em variáveis e para isso, devemos declará-las de acordo com a sintaxe:

NomeVariável : **Tipo**

Ex.: VARIÁVEIS

SalMes, Inss: REAL

Nome: CHARACTER[30] // Cadeia de caracteres

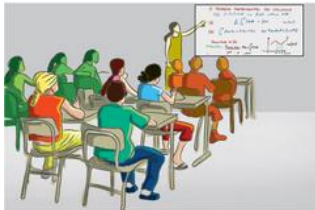
Sexo: CHARACTER // Caractere

Nesta definição, deveremos seguir algumas regras:

- VARIÁVEIS é a palavra chave, que deverá ser utilizada uma única vez na definição das variáveis e antes do uso das mesmas;
- Variáveis de tipos diferentes deverão ser declaradas em linhas diferentes;
- Em uma mesma linha, quando quisermos definir variáveis de mesmo tipo, deveremos usar o símbolo de vírgula (,) para separar as mesmas.
- Não se deve esperar para declarar variáveis quando precisar delas; isto pode confundir um programador imprudente e embaralhar o escopo do código.
- Não se deve declarar variáveis locais com nomes iguais a de outras variáveis de nível superior.

Observe a seguir exemplos de declaração de variáveis.

Exemplo A



Apresente a sintaxe para declaração da variável A, que armazena a quantidade de alunos de uma sala.

Solução

Solução:

```
inteiro A;
```

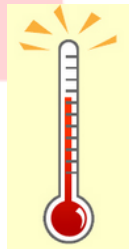
Exemplo B

Apresente a sintaxe para declaração da variável B, que armazena a temperatura de um determinado objeto.

Solução

Solução:

```
real B;
```



RESUMO

Neste módulo vimos que, dentro de um algoritmo, podemos encontrar basicamente duas classes diferentes de dados: constantes e variáveis. Foram apresentados os seus e a sua aplicação dentro de um algoritmo, detalhados os tipos de dados das variáveis: numérico, lógico e literal, conceituadas e exemplificadas as variáveis de entrada e saída nos algoritmos. As regras de identificação das variáveis foram apresentadas. Vimos que o pseudocódigo é um padrão textual de representação de um algoritmo, e pode ser denominado também de português estruturado ou Portugol. Foi demonstrada a sua estrutura básica e foi descrita e exemplificada a estrutura inicial que é a declaração de variável. Com estes conceitos é possível definir as constantes e variáveis de entrada e saída, e iniciar a construção de um algoritmo com a declaração de variáveis.

UNIDADE I – CONCEITOS BÁSICOS DE PROGRAMAÇÃO

MÓDULO 4 – OPERADORES

01

OPERADOR DE ATRIBUIÇÃO

A operação de atribuição permite que se forneça um valor numérico, literal ou lógico ou, ainda, um valor numérico de uma expressão a uma variável. O comando de atribuição é representado pelo símbolo “←”.

Como pode ser visto abaixo, no lado esquerdo do operador ← será colocado o nome da variável que irá receber o valor, e do lado direito o valor que será armazenado na mesma. A seguir são apresentados alguns exemplos de atribuições de valores a variáveis:

TipoVeiculo ← carro;

Cor ← Azul;

Sintaxe:

variável ← expressão

Exemplo de sintaxe de declaração de variáveis com a operação de atribuição.


```
#incluir <biblioteca>
principal( )
inicio
    inteiro x;
    literal y[4];
    lógico teste;
    x ← 4;
    x ← x + 2;
    y ← "aula";
    teste ← falso;
    bloco_de_comandos;
fim
```

02

Agora é sua vez. Responda:

Qual o valor das variáveis A, B, C ao final do processamento do pseudocódigo abaixo?

```
#incluir <biblioteca>
principal( )
inicio
    inteiro A, B, C;
    A ← 10;
    B ← 5;
    C ← 1;
    A ← B – C;
    B ← A + C;
    C ← A – B;
    A ← A + C – B;
    C ← B – C;
    B ← A + C – B;
    escreva(A, B, C);
fim
```



Resposta

Resposta:

A = -2; B = -1, C = 6)

03

COMANDO DE ENTRADA

Os algoritmos necessitam receber informações de fora do próprio algoritmo, a partir da entrada padrão (em geral, o teclado). Ferrari cita o exemplo de um sistema de locadora: sempre que alugamos um filme, o sistema irá necessitar de algumas informações, como o nosso código de cliente (ou o nome) e o nome da fita que estamos locando. Essas informações são fornecidas pelo sistema a partir de **comandos de entrada de dados**.

Para realizarmos a entrada de dados utilizaremos o comando **leia**. Ao utilizar o comando **leia**, o programador deve saber de antemão qual a variável que irá armazenar o valor que será fornecido pelo usuário. No caso do exemplo, os valores que seriam fornecidos pelo usuário são referentes ao código do cliente e ao nome da fita que o mesmo está locando. Sendo assim, é necessário declarar variáveis que possam armazenar valores que sejam compatíveis com as informações solicitadas ao usuário. **Saiba+**

Os cálculos do computador são de pouco valor a não ser que, primeiro, se possa fornecer os dados sobre os quais estes cálculos serão efetuados e, segundo, se for possível visualizar os resultados destes cálculos.

Exemplo: refazer o exemplo anterior supondo que o usuário tenha entrado:

a) A = -2, B = -1, C = 6.

b) A = -3, B = -2, C = -1

O comando de entrada é utilizado para receber os dados digitados pelo usuário, por meio de um dispositivo de entrada (teclado, mouse, monitor). Os dados recebidos são armazenados em variáveis, conforme os tipos que os comportem. O comando de entrada é representado pela palavra **LEIA**.

Sintaxe:

```
leia(nome_variável);
leia(variável-1, variável-2, ..., variável-n);
```

Saiba+

Por exemplo, a informação do código do cliente pode ser um valor do tipo inteiro, então é necessário

que declaremos no algoritmo uma variável desse tipo. Seguindo esse mesmo raciocínio, a informação do nome da fita pode ser uma informação do tipo **caractere**, sendo também necessário que declaremos no algoritmo uma outra variável para receber essa informação.

04

COMANDO DE SAÍDA

O comando de saída é utilizado para apresentar ao usuário perguntas necessárias ao processamento e respostas obtidas com o mesmo, nos dispositivos de saída (monitor, impressora). As informações apresentadas ao usuário podem ser conteúdos de variáveis ou mensagens. O comando de saída é representado pela palavra **ESCREVA**.

Sintaxe:

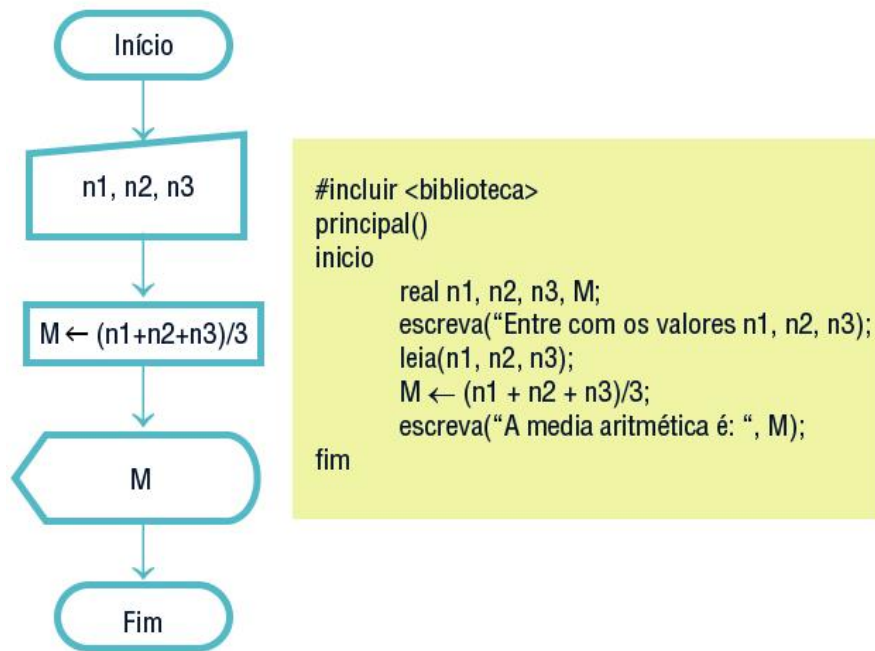
```
escreva("mensagens");
escreva(nome_variável);
escreva("mensagens", nome_variável);
```

Observe os exemplos a seguir. Tente fazer o que se pede e somente depois clique para ver a resposta.

Exemplo 1



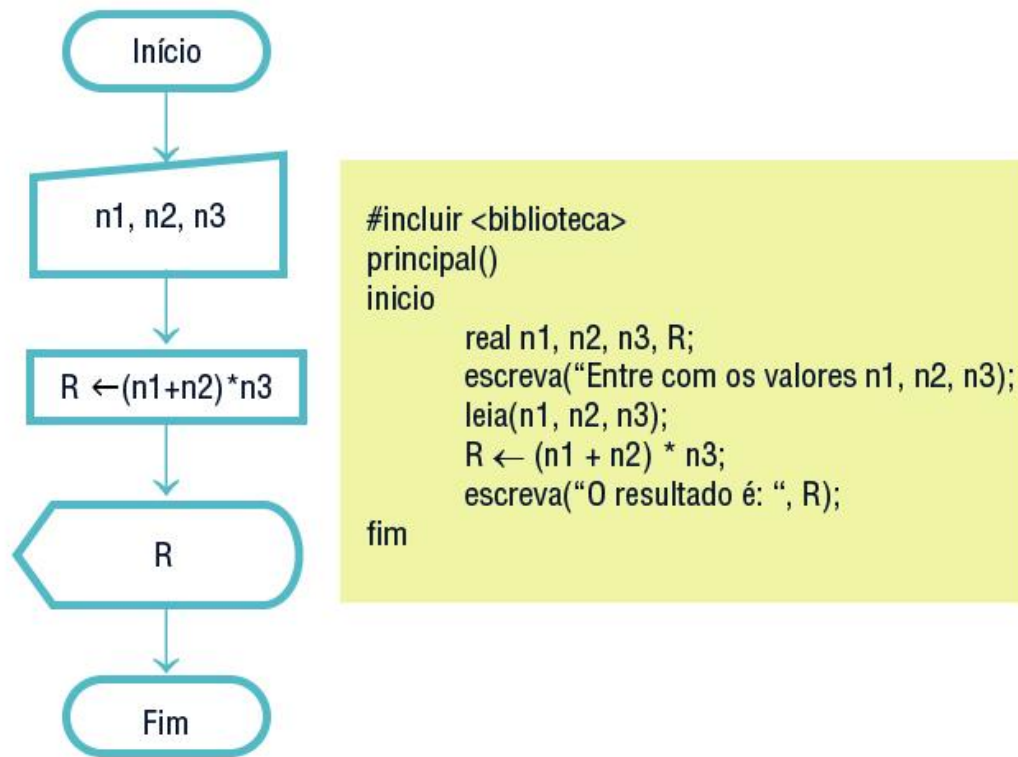
Elaborar um algoritmo que receba três notas de um aluno e retorne a média aritmética do mesmo, utilize fluxograma e linguagem algorítmica.

**Exemplo 2**

Elaborar um algoritmo que recebe três números reais some os dois primeiros e multiplique o resultado pelo terceiro, utilize fluxograma e linguagem algorítmica.

05





Descrevendo lembretes no corpo do programa

Para descrever um lembrete em um corpo de programa, utiliza-se o comando `//`.

Sintaxe:

```
// Lembrete
```

06

OPERADORES ARITMÉTICOS E ORDEM DE PRECEDÊNCIA

Os operadores aritméticos são os utilizados nos cálculos algébricos.

As operações aritméticas fundamentais são: adição, subtração, multiplicação, divisão, potenciação, divisão inteira e o resto (módulo).

A tabela a seguir apresenta os operadores aritméticos.

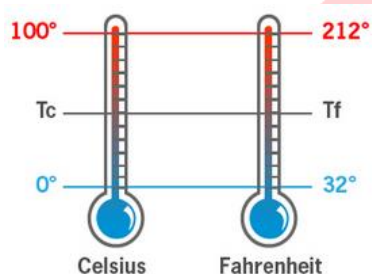
OPERADOR	AÇÃO	EXEMPLO
+	Adição	2+3, 9+x
*	Multiplicação	2*1, x*y, 2*g(x)
/	Divisão	1/x, a/z
-	Subtração	x-1, f(x)-2
DIV	Divisão Inteira	div 10 div 2, 120 div 10
MOD	Resto da Divisão Inteira	mod 10 mod 2, 120 mod 10

Durante a execução de uma expressão que envolve vários operadores, deve-se seguir uma ordem de prioridades, caso contrário, corre-se o risco de obter valores que não representem o resultado esperado.

Veja a seguir alguns exemplos de operadores aritméticos e, na sequência, a ordem de precedência desses operadores.

07

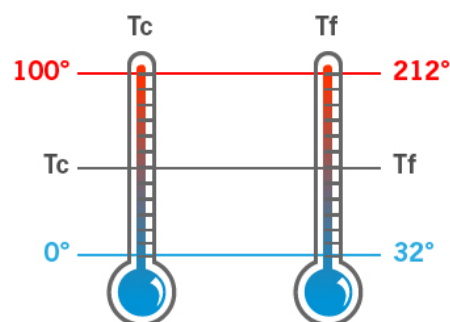
Exemplo A

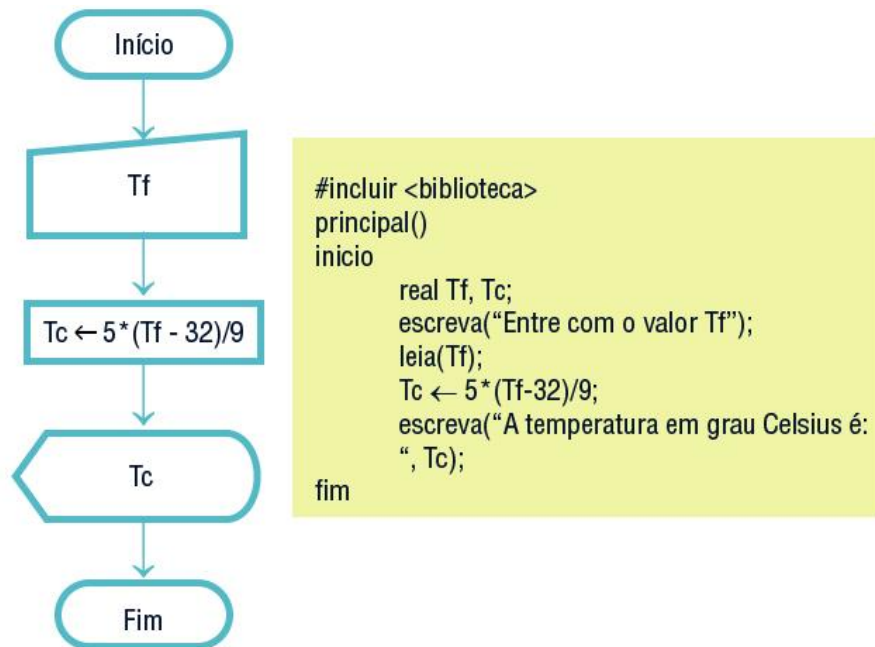


A temperatura é um parâmetro físico descritivo de um sistema que vulgarmente se associa às noções de frio e calor, bem como às transferências de energia térmica, mas que se poderia definir, mais exatamente, sob um ponto de vista microscópico, como a medida da energia cinética associada ao movimento (vibração) aleatório das partículas que compõem um dado sistema físico.

Elaborar um algoritmo que utilize os operadores aritméticos e converta uma temperatura Fahrenheit em uma correspondente na escala Celsius, utilize fluxograma e linguagem algorítmica.

$$T_c = \frac{5}{9} (T_f - 32)$$





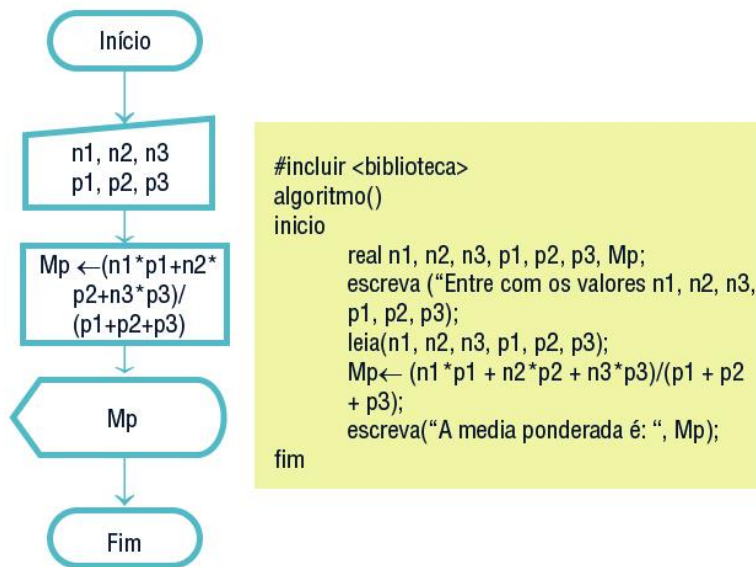
Exemplo B

Elaborar um algoritmo que receba três notas, e os respectivos pesos, calcule e mostre a média ponderada dessas notas, utilize fluxograma e linguagem algorítmica.

$$M_p = \frac{n_1p_1 + n_2p_2 + n_3p_3}{p_1 + p_2 + p_3}$$



08



09

ORDEM DE PRECEDÊNCIA

Quando uma expressão aritmética precisa ser avaliada num algoritmo, o analisador processa a expressão dando prioridade para certos operadores.

A tabela a seguir apresenta a ordem de prioridade dos operadores numa expressão aritmética, chamada de **precedência de operadores**.

ORDEM	OPERAÇÃO	SÍMBOLO
1 ^a	Parênteses	()
2 ^a	Potenciação	**
3 ^a	Multiplicação, Divisão, Resto e Divisão Inteira	*, /, mod, div
4 ^a	Adição, Subtração	+, -

Conforme o quadro acima, as primeiras subexpressões a serem resolvidas serão aquelas embutidas em parênteses mais internos, em seguida, as potências, depois as multiplicações e divisões, e assim por diante, seguindo a ordem.

Vejamos, a seguir, alguns exemplos.

Dica

A maneira mais fácil de verificar a ordem de execução das operações numa expressão aritmética é

seguir os parênteses, sendo que eles são executados antes de tudo, a partir dos mais internos para os mais externos.

Os operadores aritméticos devem obedecer à seguinte ordem de precedência

Parêntesis mais internos e mais a esquerda;
 Operações mais à esquerda, quando houver a mesma precedência;
 Operadores $*$, $/$, MOD, DIV possuem a mesma precedência;
 Operadores $+$, $-$ possuem a mesma precedência.

10

Exemplo:

Qual operador devo utilizar para descobrir se um número é par ou ímpar? Exemplifique.

Solução:

$A \text{ MOD } 2 = 0$; \rightarrow Número Par

$A \text{ MOD } 2 = 1$; \rightarrow Número Ímpar

Para fixar bem a ordem de prioridade dos operadores, efetue as expressões abaixo e clique em “Resposta” para verificar se você acertou.

Efetue as seguintes expressões:

a) $101 \text{ MOD } 3 * 2 + 9 * 5 \text{ MOD } 10$;

Resposta a

Resposta: 9.

b) $4 + 5 * 9 - 12 / 3$;

Resposta b

Resposta: 45.

c) $40 / 8 + 2 * (5 - 11 \text{ MOD } 3)$;

Resposta c

Resposta: 11.

d) $105 \text{ MOD } 3 * 4 - 3 * 4 \text{ MOD } (101 \text{ DIV } 10);$

Resposta d

Resposta: -2.

e) $A = 1, B = 2, C = 3, D = 4;$

$A \leftarrow 40 * B \text{ MOD } C;$

$C \leftarrow A * 10 - 5 \text{ DIV } D;$

escreva(A, B, C, D);

Resposta e

Resposta: 2, 2, 19, 4.

11

OPERADORES RELACIONAIS

Os operadores relacionais são utilizados para comparar duas variáveis.

A tabela a seguir apresenta os principais operadores relacionais.

OPERADOR	AÇÃO
>	Maior que
>=	Maior ou igual que
<	Menor que
<=	Menor ou igual que
==	Igual a
!=	Diferente de

Exemplo:

Se o usuário digitar o valor 3, qual é o resultado encontrado no seguinte algoritmo:

#incluir <biblioteca>

principal()

início

inteiro A, B, C, D, R;

$A \leftarrow 1; B \leftarrow 2; C \leftarrow 3;$

escreva("Digite D");

leia(D);

se $(D + A < C + 4)$

$R \leftarrow A + B + C + D;$

senão

$R \leftarrow A - B + C - D;$

escreva("O valor de R é: ", R);

fim

Solução

Solução:

$$D + A = 3 + 1 = 4$$

$$C + 4 = 3 + 4 = 7$$

$$(4 < 7) \rightarrow \text{verdade}$$

$$R = 1 + 2 + 3 + 3 = 9$$

Agora é sua vez. Resolva:

O que acontece no exemplo anterior se o usuário digitar 10?

Solução

Solução:

$$D + A = 10 + 1 = 11$$

$$C + 4 = 3 + 4 = 7$$

$$(11 < 7) \rightarrow \text{falso}$$

$$R = 1 - 2 + 3 - 10 = -8.$$

OPERADORES LÓGICOS

São utilizados para determinar o valor lógico de uma proposição composta.

Proposição é uma ideia de sentido completo.

As proposições podem ser simples ou compostas. As simples possuem um único sentido completo enquanto as compostas dois ou mais.

A palavra que une as proposições simples para formar as compostas é denominada **conectivo**. Os principais conectivos são: e (\wedge), ou (\vee), não (\sim), se... então (\rightarrow), se somente se (\leftrightarrow).

Estes conectivos geram as tabelas-verdades fundamentais

Tabelas-verdades fundamentais

p	q	$p \wedge q$	$p \vee q$	$\sim p$	$p \vee q$	$p \rightarrow q$	$p \leftrightarrow q$
V	V	V	V	F	F	V	V
V	F	F	V	F	V	F	F
F	V	F	V	V	V	V	F
F	F	F	F	V	F	V	V

Exemplo:

Em um triângulo retângulo, o quadrado da medida da hipotenusa é igual à soma dos quadrados das medidas dos catetos.

13

Em programação, os conectivos são representados pelos operadores lógicos.

OPERADOR LÓGICO MATEMÁTICO	OPERADOR LÓGICO COMPUTACIONAL	AÇÃO
$\sim p$!p	Negação
$p \wedge q$	p && q	E
$p \vee q$	p q	Ou
$p \rightarrow q$!p q	Se...então
$p \leftrightarrow q$	(!p q) && (!q p)	Se somente se

O recurso de comparar e associar expressões através de operadores lógicos permite a execução de ações dependendo da combinação de resultados das expressões.

Exemplo



Podemos incluir no algoritmo da troca de pneus um passo para verificar se o pneu estepe e as ferramentas estão no carro.

Se pneu estepe = Verdadeiro e macaco = Verdadeiro e chave_de_roda = Verdadeiro Então

A expressão acima verifica se todos os itens estão no carro para continuar a execução da troca de pneu. Se um dos itens for Falso, a expressão toda se torna falsa, portanto, impossibilitando a continuidade do processo de troca de pneu. Isso acontece porque o conector utilizado foi o “e”.

Para verificar se um dos pneus está furado, podemos utilizar a expressão:

Se pneu1 = “Furado” ou pneu2 = “Furado” Então

Significa que se um dos pneus estiver furado, a expressão será verdadeira. Inclusive se os dois estiverem furados, a expressão também será verdadeira.

Agora, observe os exemplos a seguir.

Atribuindo os valores às variáveis a, b e c conforme abaixo:

$a \leftarrow 30$, $b \leftarrow 40$ e $c \leftarrow 80$

Temos os seguintes resultados nas expressões a seguir.

$a > 30$ e $c = 80$	Falso
---------------------	-------

O valor de a é 30, portanto, a expressão “a é maior que 30” é falsa. O valor de c é 80, portanto, a expressão “c é igual a 80” é verdadeira. No entanto, a conjunção “e” resulta verdadeira apenas se todas as expressões resultarem verdadeiras, nesse caso, portanto, o resultado é falso.

$a \neq 20$ ou $b \leq 50$	Verdadeiro
----------------------------	------------

O valor de a é 30. A expressão “a é diferente de 30” é verdadeira. A expressão “b é menor ou igual a 50” é verdadeira, pois b tem o valor atribuído de 40. Como a disjunção “ou” resulta verdadeira se pelo menos uma das expressões resultar verdadeira, por isso o resultado final é verdadeiro.

$a = 30$ e não $b \leq 40$	Falso
----------------------------	-------

A expressão “a é igual a 30” é verdadeira. A expressão b é menor ou igual a 40 também é verdadeira, mas o operador de negação “não” inverte esse resultado, portanto a expressão “não b é menor ou igual a 40” torna-se falsa. O resultado é falso, pois na conjunção “e” basta um valor falso para tornar toda a expressão falsa.

$a \neq b$ e $b = c$	Falso
----------------------	-------

A expressão “a é diferente de b” é verdadeira, pois o valor de a é 30 e o valor de b é 40. A expressão b é igual a c é falsa, pois o valor de c é 80. O resultado da conjunção “e” neste caso também é falso.

$a > b$ ou $b > 50$	Falso
---------------------	-------

A expressão “a é maior que b” é falsa. A expressão “b é maior que 50” é falsa. Portanto, o resultado é falso, já que para a disjunção “ou”, todas as expressões tem que ser falsas para o resultado ser falso.

$c > 20$ ou $a < b$	Verdadeiro
---------------------	------------

A expressão “c é maior que 20” é verdadeira, pois o valor de c é 80. A expressão “a é menor que b” também é verdadeira. Na disjunção “ou” basta uma das expressões ser verdadeira para o resultado ser verdadeiro, nesse caso, ambas são verdadeiras, o que também resulta em verdadeiro.

15

Em resumo, podemos concluir que a conjunção “e” será falsa se pelo menos uma das expressões for falsa e somente será verdadeira se todas forem verdadeiras.

A disjunção “ou” será falsa se todas as expressões forem falsas e será verdadeira se pelo menos uma for verdadeira. A operação de negação sempre irá inverter o valor da expressão.

Tabela Verdade “e”, “ou”, “não”

p	q	$p \wedge q$	$p \vee q$	$\sim p$
V	V	V	V	F
V	F	F	V	F
F	V	F	V	V
F	F	F	F	V

A disciplina Lógica Matemática mostra detalhadamente a aplicação dos operadores lógicos e das tabelas verdade.

RESUMO

Neste módulo foram apresentados os operadores utilizados para manipular os dados em um algoritmo. Vimos que a operação de atribuição permite que se forneça um valor numérico, literal ou lógico ou, ainda, um valor numérico de uma expressão a uma variável. Estudamos também os operadores de entrada e saída, que são utilizados para atribuir valores de entrada e mostrar valores de saída nos algoritmos. Foram apresentados os operadores aritméticos que manipulam os valores numéricos, os operadores relacionais que comparam valores e os operadores lógicos que comparam expressões lógicas. Com esses conhecimentos é possível manipular valores e variáveis e compará-las logicamente.

