



# Lógica de Programação

Leonardo Guimarães de Holanda  
Cinthya Cavalcanti Flório



**Curso Técnico em Desenvolvimento de Sistemas**  
Educação a Distância  
2019



# Lógica de Programação

Leonardo Guimarães de Holanda  
Cinthya Cavalcanti Flório

**Curso Técnico em Desenvolvimento de Sistemas**

Educação a Distância

Escola Técnica Estadual Professor Antônio Carlos Gomes da Costa

Recife - PE

1.ed. | out. 2019



Licença Pública Creative Commons  
Atribuição-NãoComercial-Compartilhável 4.0 Internacional

#### **Professor(es) Autor(es)**

Leonardo Guimarães de Holanda  
Cinthya Cavalcanti Flório

#### **Revisão**

Leonardo Guimarães de Holanda  
Cinthya Cavalcanti Flório  
José Américo Teixeira de Barros

#### **Coordenação de Curso**

José Américo Teixeira de Barros

#### **Coordenação Design Educacional**

Deisiane Gomes Bazante

#### **Design Educacional**

Ana Cristina do Amaral e Silva Jaeger  
Fernanda Paiva Furtado da Silveira  
Izabela Pereira Cavalcanti  
Jailson Miranda  
Roberto de Freitas Moraes Sobrinho

#### **Descrição de imagens**

Sunnye Rose Carlos Gomes

#### **Catálogo e Normalização**

Hugo Cavalcanti (Crb-4 2129)

#### **Diagramação**

Jailson Miranda

#### **Coordenação Executiva**

George Bento Catunda  
Renata Marques de Otero  
Manoel Vanderley dos Santos Neto

#### **Coordenação Geral**

Maria de Araújo Medeiros Souza  
Maria de Lourdes Cordeiro Marques

#### **Secretaria Executiva de Educação Integral e Profissional**

#### **Escola Técnica Estadual Professor Antônio Carlos Gomes da Costa**

#### **Gerência de Educação a distância**

#### **Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISDB**

H722I

Holanda, Leonardo Guimarães de.

Lógica de Programação: Curso Técnico em Desenvolvimento de Sistemas: Educação a distância / Leonardo Guimarães de Holanda, Cinthya Cavalcanti Flório. - 1.ed. - Recife: Escola Técnica Estadual Professor Antônio Carlos Gomes da Costa, 2019.  
86 p.: il.

Inclui referências bibliográficas.

Caderno eletrônico produzido em outubro de 2019 pela Escola Técnica Estadual Professor Antônio Carlos Gomes da Costa.

1. Lógica de Programação. 2. Informática. I. Título.

CDU – 005.1



## Sumário

Introdução .....	7
1.Competência 01   Conhecer os princípios de lógica de programação algorítmica .....	13
1.1 Sequência Lógica .....	13
1.2 Instruções .....	13
1.3 Algoritmo .....	13
1.4 Fases de um algoritmo .....	14
1.5 Desenvolvimento estruturado .....	14
1.6 Programas .....	15
1.7 Formas de Representação de Algoritmos .....	15
1.7.1 Descrição Narrativa .....	15
1.7.2 Fluxograma Convencional .....	16
1.7.3 Pseudocódigo .....	21
1.8 Comandos de Entrada e Saída .....	21
1.8.1 Comando de entrada de dados (leia) .....	21
1.8.2 Comando de saída de dados (escreva) .....	22
1.9 Representação e Armazenamento de Dados .....	23
1.9.1 Variáveis, constantes e tipos de dados .....	24
1.9.1.1 Variáveis (identificadores) .....	24
1.9.1.2 Declaração de variáveis .....	24
1.9.1.3 Constantes .....	25
1.9.1.4 Tipos de dados .....	28
2.Competência 02   Desenvolver um algoritmo para a realização de operações matemáticas .....	30
2.1 Manipulação de Dados .....	30
2.1.1 Operadores de Atribuição .....	30
2.1.2 Operadores Aritméticos Básicos .....	31



2.1.3 Prioridade dos Operadores Aritméticos Básicos.....	32
2.1.4 Operadores Relacionais .....	36
2.1.5 Expressões Aritméticas .....	38
2.1.6 Funções Primitivas .....	39
2.1.7 Operadores Lógicos.....	40
2.1.8 Expressões Lógicas .....	41
2.1.9 Tabela Verdade .....	42
2.2 Dicas adicionais para construção de programas.....	43
3.Competência 03   Desenvolver um algoritmo para resolução de um problema utilizando estrutura de decisão.....	46
3.1 Estrutura de controle se então / if ... then .....	46
3.2 Estrutura de controle se então senão / if ... then ... else.....	48
3.3 Estrutura de controle se-então-senão aninhada .....	50
3.4 Estrutura de controle caso selecione / select ... case .....	51
4.Competência 04   Desenvolver um algoritmo para resolução de um problema utilizando estrutura de repetição.....	55
4.1 Estrutura de repetição para-faça .....	55
4.2 Estrutura de repetição para-faça aninhada .....	57
4.3 Estruturas de repetição (enquanto-faça e faça-enquanto) .....	58
4.3.1 Estrutura de repetição enquanto-faça.....	59
4.3.2 Estrutura de repetição faça-enquanto.....	61
4.4 Dicas sobre estruturas de repetição. ....	63
4.5 Conceitos adicionais – Vetores, Matrizes, Procedimentos, Funções, Recursividade e Depuração .....	64
4.5.1 Vetores .....	64
4.5.2 Matrizes .....	69
4.5.3 Procedimentos .....	74
4.5.4 Funções .....	74



4.5.5 Recursividade .....	79
4.5.6 Depuração (Debug) .....	81
Conclusão .....	82
Referências .....	83
Minicurrículo do Professor .....	84
Minicurrículo do Professor .....	<b>Erro! Indicador não definido.</b>



## Introdução

É com satisfação que dou a vocês as boas-vindas à disciplina Lógica de Programação. Os assuntos que serão tratados nesta disciplina fazem parte de uma área da Informática conhecida como Desenvolvimento de Software e envolvem a solução de problemas através construção de algoritmos e programação de computador.

A programação proporciona mais organização, capacidade e agilidade em diversas aplicações, por exemplo no gerenciamento de alunos de uma escola ou dos pacientes de um hospital, no controle de estoque de uma empresa, nas comunicações de uma agência de notícias, a lista não tem fim, ela está vinculada a quase todos os aspectos da vida humana.

Nesta disciplina você aprenderá os fundamentos da construção de algoritmos e as principais estruturas que regem a programação. Realizando os exercícios adequadamente, o seu raciocínio evoluirá para ser capaz criar qualquer tipo de programa, assim, quando você estiver cursando uma disciplina posterior que envolva aprender uma linguagem de programação específica para computação a tarefa será bem mais fácil. Durante o decorrer desta disciplina será utilizada a ferramenta Portugol Studio, desenvolvida e mantida pelo LITE - laboratório de inovação tecnológica na educação da Universidade do Vale do Itajaí (Univali). Trata-se de uma ferramenta completa e atual para aprender programação que interpreta e executa o pseudocódigo Portugol. Esta seção dedicará mais esforço a apresentar a ferramenta e seu potencial para então, nas próximas começarmos a conhecer os conceitos básicos de algoritmos e programação.



Acesse: <http://lite.acad.univali.br/portugol/>  
E faça o download do Portugol Studio e instale o programa.

O Portugol Studio fornece um conjunto completo de bibliotecas de funções e exemplos de algoritmos. Todos os exemplos com copyright fornecidos pela ferramenta exibidos neste material de estudo são de autoria de *Giordana Maria da Costa Valle* e *Carlos Alexandre Krueger* e foram utilizados segundo a permissão fornecida nos próprios códigos. Este programa provê uma base para o desenvolvimento de aplicações complexas em pseudocódigo Portugol de forma similar à uma



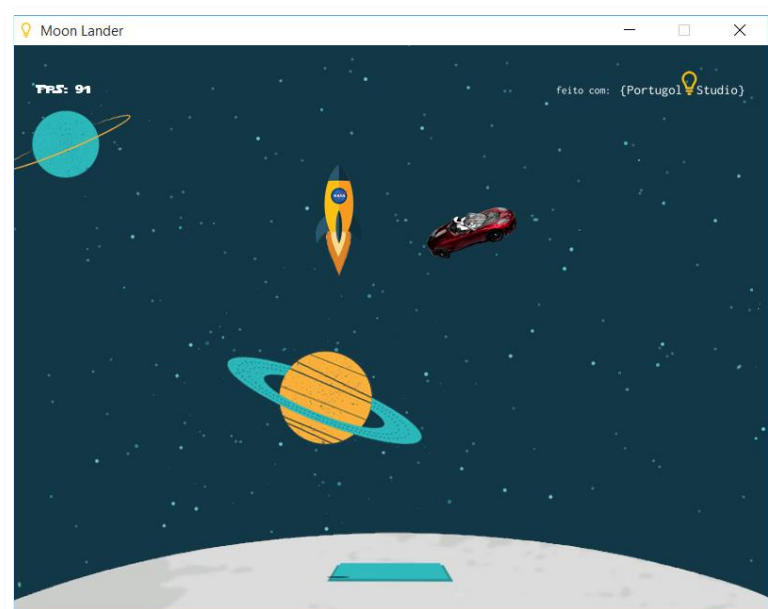
linguagem de programação genuína. Da Figura 1 à Figura 5 são apresentados alguns exemplos do que é possível com esta ferramenta de aprendizado.



**Figura 1- Tela inicial do jogo Moon Lander.**

**Fonte:** LITE – Univali em <http://lite.acad.univali.br/portugol/>

**Descrição:** Tela inicial do jogo Moon Lander, conteúdo dos exemplos do Portugol Studio, planetas e um foguete no espaço.

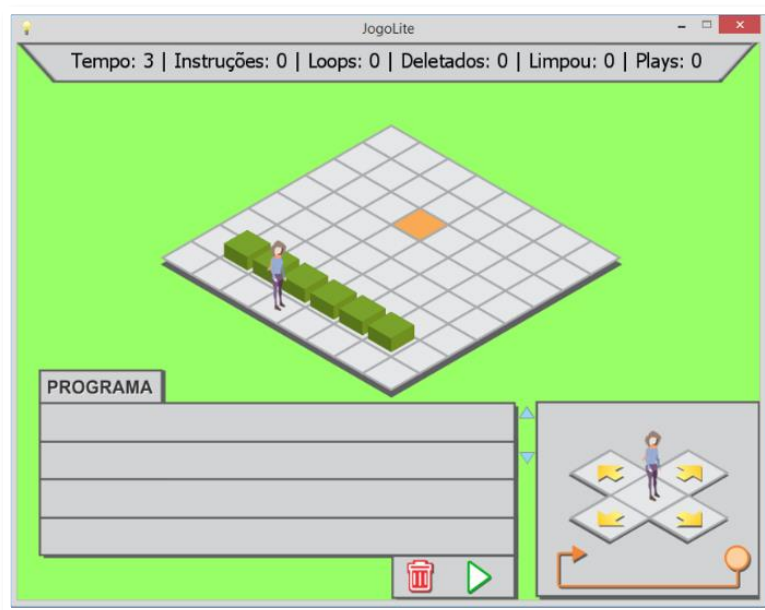


**Figura 2- Tela do jogo Moon Lander.**

**Fonte:** LITE – Univali em <http://lite.acad.univali.br/portugol/>

**Descrição:** Tela do jogo em funcionamento, planetas e um foguete e o carro da Tesla lançado no espaço pela empresa SpaceX.

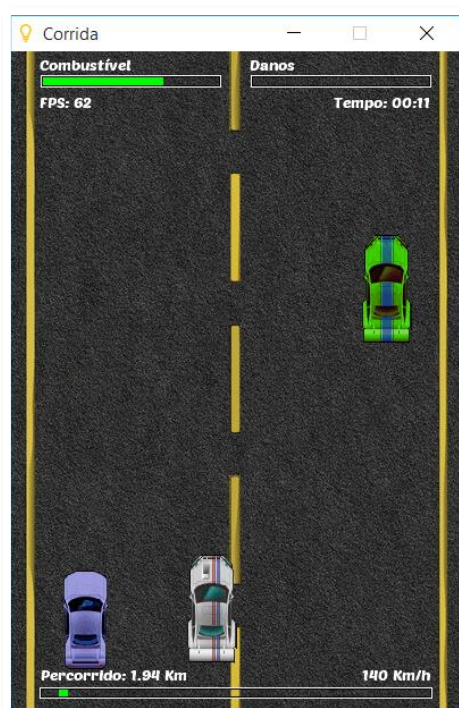




**Figura 3 - Telas do jogo Programa.**

**Fonte:** LITE – Univali em <http://lite.acad.univali.br/portugol/>

**Descrição:** Tela inicial do jogo Programa, conteúdo dos exemplos do Portugol Studio, um plano com objetos e um boneco cercado de botões de comando.



**Figura 4 - Telas do jogo Corrida.**

**Fonte:** LITE – Univali em <http://lite.acad.univali.br/portugol/>

**Descrição:** Tela do jogo Corrida, conteúdo dos exemplos do Portugol Studio, uma pista de corrida com carros competindo.



**Figura 5 - Telas do programa Bateria Eletrônica.**

**Fonte:** LITE – Univali em <http://lite.acad.univali.br/portugol/>

**Descrição:** Tela inicial, do programa Bateria Eletrônica, conteúdo dos exemplos do Portugol Studio, com três instrumentos e diversos botões de controle.

Como mencionado anteriormente, o Portugol Studio provê uma ampla biblioteca de funções, com os mais variados propósitos, para o aprendiz poder se familiarizar ao máximo com os aspectos de qualquer linguagem e ferramenta de programação que vier a ter contato futuramente. Figura 6 apresenta a tela inicial da interface do Portugol Studio.



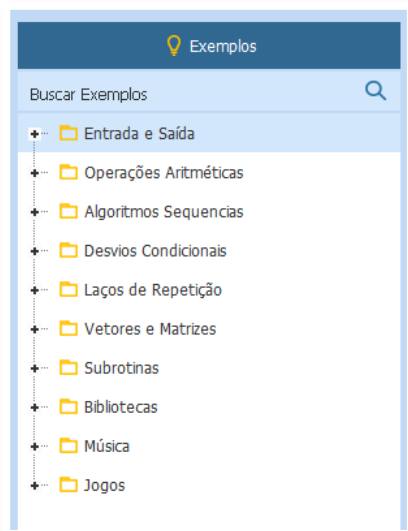
**Figura 6 - Tela inicial do Portugol Studio.**

**Fonte:** LITE – Univali em <http://lite.acad.univali.br/portugol/>

**Descrição:** Tela inicial do Portugol Studio com os botões de Novo Arquivo, Abrir Arquivo, Ajuda, Plugins, Dicas Interface (F3), Atalhos do Teclado (F11), Relatar um Bug, Contribuir e Sobre (F12), além de uma subtela de exemplos contendo uma estrutura de pastas em árvore com os exemplos nativos da ferramenta.



Além dos exemplos apresentados na Figura 6 e na Figura 7, a Figura 8 apresenta as bibliotecas fornecidas pela ferramenta.



**Figura 7 - Exemplos oferecidos pelo Portugol Studio.**

**Fonte:** LITE – Univali em <http://lite.acad.univali.br/portugol/>

**Descrição:** Subtela de exemplos oferecidos pela ferramenta apresentados numa estrutura de pastas em árvore.



**Figura 8 - Biblioteca de funções fornecida pelo Portugol Studio.**

**Fonte:** LITE – Univali em <http://lite.acad.univali.br/portugol/>

**Descrição:** Aba de Ajuda oferecida pela ferramenta com subtela de seleção de tópico à esquerda como uma estrutura de pastas em árvore contendo particularidades da linguagem e bibliotecas e à direita uma subtela de conteúdo do item selecionado.

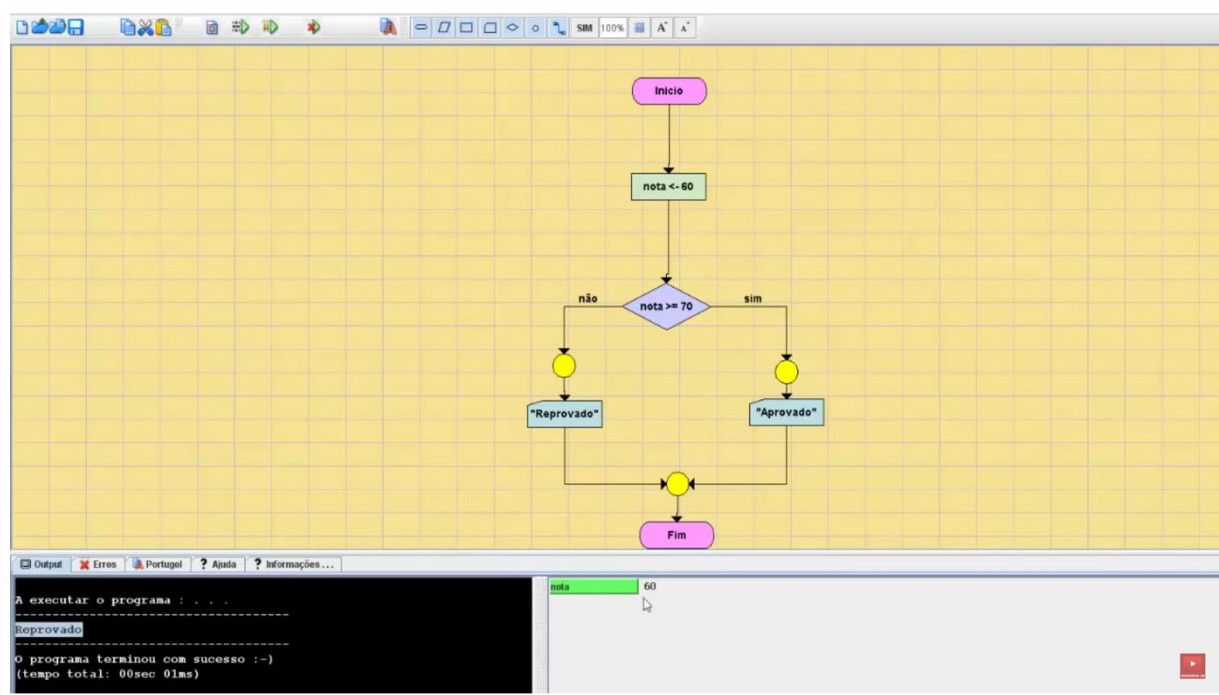


Indicamos que você assista esse vídeo como forma de ampliar o conceito de Portugol Studio.

<https://www.youtube.com/watch?v=6OIADpFlmtc>

Vale enfatizar que o potencial do Portugol Studio excede as expectativas e necessidades e escopo desta disciplina, neste material utilizaremos apenas uma parte do seu conteúdo, o que não impedirá o aluno interessado de buscar um aprofundamento maior para seu benefício.

Embora não utilizado durante o decorrer deste material recomendamos também outra ferramenta muito útil, o Portugol IDE (para construção de fluxogramas) apresentado na Figura 9.



**Figura 9 - Tela do programa Portugol IDE.**

**Fonte:** Departamento de Engenharia Informática - Instituto Politécnico de Tomar em <http://www.dei.estt.ipt.pt/portugol/>

**Descrição:** Tela do programa Portugol IDE, apresentando uma subtela superior com um fluxograma aberto e uma subtela inferior com a saída do algoritmo expresso pelo fluxograma.



Acesse: <http://www.dei.estt.ipt.pt/portugol/>

Você pode também fazer o download do Portugol IDE e instalar o programa.



## 1.Competência 01 | Conhecer os princípios de lógica de programação algorítmica

Nesta seção iniciaremos os estudos dos princípios básicos que compõem o a programação de algoritmos.

### 1.1 Sequência Lógica

Podemos entender uma sequência lógica como uma sequência de ações encadeadas para atingir um objetivo.

### 1.2 Instruções

No contexto computacional, podemos entender uma instrução como uma informação que sinaliza a um computador uma ação básica a ser executada.

### 1.3 Algoritmo

Podemos entender um algoritmo como uma sequência de passos (instruções) ordenadas de forma lógica para resolver um determinado problema ou tarefa. Eles podem ser executados por um computador, um autômato (robô) ou mesmo por um ser humano. Para um algoritmo que representa a solução de um problema poder ser definido geralmente é necessário que seja estabelecido um processo com algumas etapas a serem realizadas:

1. Ter o problema bem definido. É muito difícil e não recomendado misturar vários problemas em um e tentar resolver todos juntos. Esta etapa baseia-se no princípio de divisão e conquista.
2. Organizar um estudo da situação atual para listar quais as possíveis soluções do problema. A solução selecionada deve ser registrada, futuramente pode servir para outros problemas similares. Além disso se solução selecionada não servir, fica registrado para o desenvolvedor futuramente não tentar usá-la novamente para o mesmo problema.





3. Uma vez que a etapa de estudo esteja finalizada, o próximo passo é utilizar uma linguagem de programação para escrever o programa com o qual se pretende resolver o problema (implementar a solução).
4. Testar o programa com os usuários e demais interessados na solução do problema para verificar se ele foi resolvido.
5. Caso a solução testada não funcione, o processo deverá voltar à fase de estudo para se descobrir onde se cometeu o erro.

## 1.4 Fases de um algoritmo

Além das etapas do processo de definição de um algoritmo, vistos no item anterior, podemos dividir a execução de um algoritmo em fases fundamentais conforme a Figura 10.

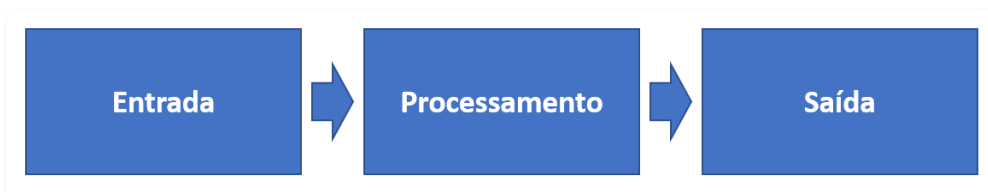


Figura 10- Fases de um algoritmo.

Fonte: o autor

**Descrição:** Três etapas sequenciais expressas por retângulos conectados através de setas, todos dispostos da esquerda para a direita, entrada, processamento e saída.

Sendo:

**ENTRADA:** São os dados de entrada para o algoritmo trabalhar.

**PROCESSAMENTO:** São os procedimentos ou manipulações realizadas nos dados para se chegar ao resultado final.

**SAÍDA:** São os dados já processados, os resultados apresentados ao usuário.

## 1.5 Desenvolvimento estruturado

O desenvolvimento estruturado é uma forma de pensar (paradigma) na qual a solução de um problema é alcançada através da sua quebra em problemas menores. Já vimos isso em algum lugar não vimos? Na primeira etapa de definição de um algoritmo no item Algoritmo. Em verdade este



paradigma é conhecido com diversos nomes (desenvolvimento estruturado, divisão e conquista, decomposição top-down...), todos conduzindo ao mesmo princípio, quebrar o problema em problemas menores.

## 1.6 Programas

Um programa de computador é um algoritmo escrito numa linguagem de computador (Assembly, Basic, Pascal, C, C++, Cobol, Fortran, Visual Basic, Java, Python entre outras) e que é interpretado e executado por uma máquina, em geral um computador. Para melhorarmos o entendimento vamos nos aprofundar um pouco mais e conhecer também os conceitos de linguagem de programação.

Uma linguagem de programação, é uma forma padronizada para se construir instruções para um computador. É através de uma linguagem que um programador especifica precisamente sobre quais dados e instruções um computador deve trabalhar, como estes dados serão armazenados ou transmitidos e quais instruções devem ser executadas diante de várias situações.

## 1.7 Formas de Representação de Algoritmos

Existem diversas formas de representação utilizadas para descrever um algoritmo, dependendo do objetivo e do perfil profissional dos envolvidos pode-se desejar abstrair ou detalhar características diferentes do problema a ser resolvido. Dentre inúmeras formas de representação, as mais populares são listadas a seguir:

### 1.7.1 Descrição Narrativa

Nesta forma os algoritmos são expressos em linguagem natural (português, alemão, francês, inglês, espanhol, etc.). Como vantagem, é bem mais fácil construir um algoritmo utilizando uma linguagem com a qual já temos um certo domínio, do que através de linguagens as quais não temos familiaridade. A principal desvantagem é que sua tradução para outra linguagem utilizada por computadores é bastante trabalhosa. Além disso, linguagens naturais podem causar erros de interpretação (ambiguidade, muitas vezes uma palavra tem mais de um significado, dependendo do contexto em que são utilizadas).



Podemos considerar como exemplo de um algoritmo representado com descrição narrativa os algoritmos apresentados na Figura 11 e na Figura 12:

Uma receita de bolo (muitas vezes essa expressão é utilizada para referenciar uma solução básica e amplamente conhecida de um determinado problema):

- 1- Misture 500g de farinha, 250ml de leite e 2 ovos na batedeira.
- 2- Unte a forma com manteiga.
- 3- Despeje a mistura na forma.
- 4- Se houver raspas de chocolate então despeje sobre a mistura.
- 5- Leve a forma ao forno por 30 minutos.
- 6- Enfie uma faca na massa e verifique se ela cola no talher.
- 7- Enquanto a massa colar na faca deixe a forma no forno.
- 8- Retire do forno.
- 9- Deixe esfriar (se você gosta de bolo frio 😊).

**Figura 11- Descrição Narrativa – Receita de bolo.**

**Fonte:** o autor

**Descrição:** Exemplo muito utilizado para referenciar soluções básicas conhecido como “receita de bolo” devido à analogia do processo descrito narrativamente, passo a passo: 1-Misture 500g de farinha, 250ml de leite e 2 ovos na batedeira; 2-Unte a forma com manteiga. 3-Despeje a mistura na forma; 4-Se houver raspas de chocolate então despeje sobre a mistura; 5-Leve a forma ao forno por 30 minutos; 6-Enfie uma faca na massa e verifique se ela cola no talher; 7- Enquanto a massa colar na faca deixe a forma no forno; 8-Retire do forno; 9-Deixe esfriar (se você gosta de bolo frio).

Enviar uma carta (Sim! Tem gente que ainda usa! 😊):

- 1- Pegue um papel e um lápis.
- 2- Escreva uma mensagem no papel usando o lápis.
- 3- Pegue o papel com a mensagem e coloque num envelope.
- 4- Escreva o remetente e o destinatário no envelope e sele ele.
- 5- Leve o envelope até os correios e solicite o envio.

**Figura 12 - Descrição Narrativa - Enviar uma carta.**

**Fonte:** o autor

**Descrição:** Exemplo simples de processo descrito narrativamente, passo a passo: 1-Pegue um papel e um lápis; 2-Escreva uma mensagem no papel usando o lápis; 3-Pegue o papel com a mensagem e coloque num envelope; 4-Escreva o remetente e o destinatário no envelope e sele ele; 5-Leve o envelope até os correios e solicite o envio.


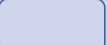





## 1.7.2 Fluxograma Convencional

Fluxograma é uma forma de representação gráfica onde formas geométricas padronizadas e combinadas descrevem os passos a serem executados pelo algoritmo representado. Trata-se de uma forma de representação utilizada para diversas aplicações, modelagem e representação de processos,





sistemas, algoritmos, etc. Cada forma geométrica tem um significado (ação) distinto conforme a Figura 13.

Símbolo	Significado
 ou 	Início / Fim
 ou 	Entrada de dados
	Saída de dados
	Atribuição / Processamento
	Decisão

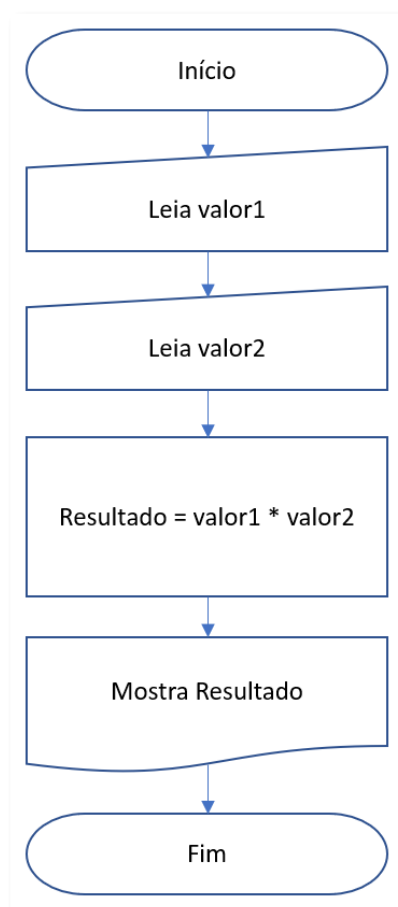
**Figura 13 - Formas/Símbolos que representam ações para construir fluxogramas.**

**Fonte:** o autor

**Descrição:** Tabela com duas colunas, a primeira com os símbolos que representam as ações num fluxograma, a segunda com o significado das ações. Início, um retângulo com as arestas abauladas. Entrada de dados, um retângulo com a aresta superior esquerda cortada. Saída de dados, um quadrilátero, com a base inferior curva. Atribuição, um retângulo normal. Decisão, um losango.

Como principal vantagem os fluxogramas são mais fáceis de serem compreendidos. Eles têm padrão mundial, o que os torna independentes das particularidades de uma linguagem natural.

Existem diversas maneiras de se representar ações, a forma de se construir um fluxograma não é rígida, não está escrita em pedra. A Figura 14 mostra um fluxo de execução linear simples onde são usados os verbos Leia e Mostra para representar as ações de entrada e saída.

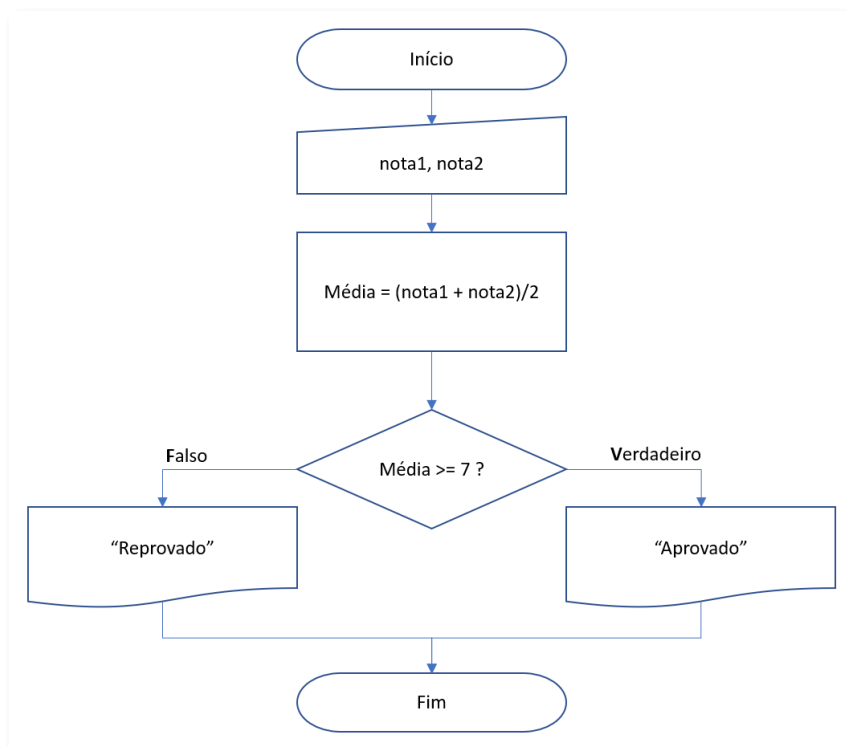


**Figura 14 - Fluxograma simples.**

**Fonte:** o autor

**Descrição:** Fluxograma com um fluxo linear simples com os passos Início, Leia valor1, Leia valor2, Resultado é igual a Valor1 vezes Valor2, Mostra Resultado e Fim. Todos dispostos sequencialmente de cima para baixo, cada um com sua respectiva forma que foi apresentada na Figura 13.

Já na Figura 15 o exemplo mostra um fluxo linear com uma estrutura de decisão que o divide e depois converge no fim do algoritmo. Perceba que as representações de entrada e saída estão diferentes, a entrada foi resumida com os dois valores requisitados dentro dela e a saída foi resumida com apenas a mensagem ou valor a ser exibido entre aspas.



**Figura 15 - Fluxograma bifurcado.**

**Fonte:** o autor

**Descrição:** Fluxograma bifurcado, onde os passos são: Início, Leia nota1 e nota2, Média é igual a abre parênteses nota1 mais nota2 fecha parênteses dividido por dois, então vem a condição "SE" a Média for maior ou igual a sete, que bifurca o fluxo, se sim Mostra "Aprovado", se não Mostra "Reprovado", então converge o fluxo para Fim. Todos com as respectivas formas apresentadas na Figura 13.



Indicamos que você assista esse vídeo como forma de ampliar o conceito de Fluxogramas

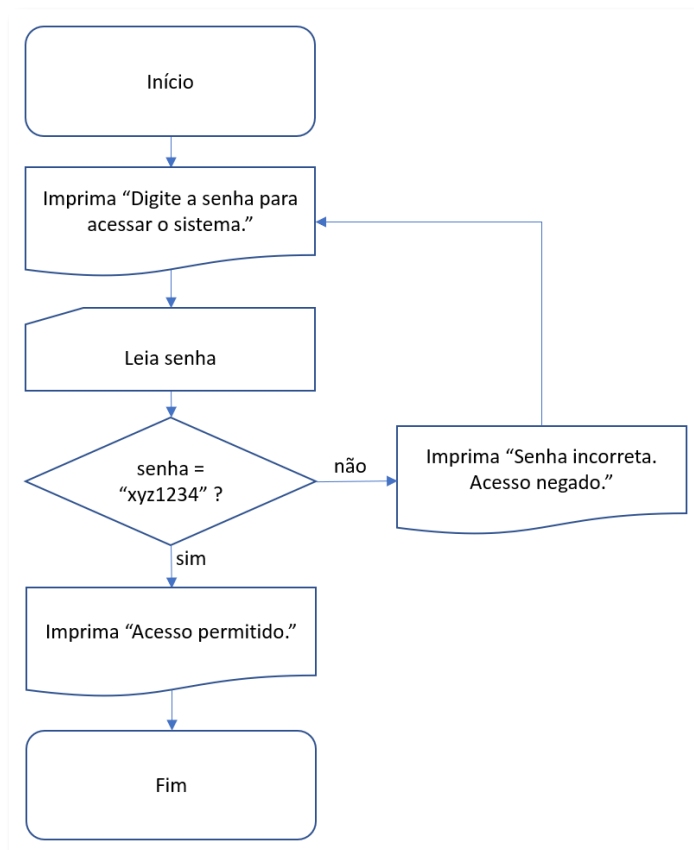
<https://www.youtube.com/watch?v=jbRzQVzH7ss&index=6&list=PLN2AgTfb0UJV69nIF8XMtWIRlpiBxxW0vDEBUG+%7C+DESCUBRA+COMOFUNCIONA+UM+PROGRAMA+POR+DENTRO+%7C+JEAN+VARGAS>



Acesse: <http://www.dei.estt.ipt.pt/portugol/>  
e faça o download do Portugol IDE



No exemplo da Figura 16 foram utilizadas formas geométricas alternativas para representar a entrada de dados, o início e o fim do algoritmo, para as saídas foi utilizado o verbo *Imprima*, na estrutura condicional foi utilizado sim e não no lugar de Verdadeiro e Falso.



**Figura 16 - Fluxograma bifurcado com repetição.**

**Fonte:** o autor

**Descrição:** Fluxograma bifurcado com repetição onde os passos são: Início, Imprima "Digite a senha para acessar o sistema.", Leia senha, então vem a condição "SE" a senha é igual a "xyz1234", se sim Imprima "Acesso permitido" e vai para o passo Fim, se não Imprima "Senha incorreta. Acesso negado." E retorna ao passo Imprima "Acesso permitido" antes da condição "SE". Todos com as respectivas formas apresentadas na Figura 13.



### DICA IMPORTANTE!

É possível construir fluxogramas através de várias ferramentas, até um editor de slides tem recursos de fluxograma.



## 1.7.3 Pseudocódigo

O pseudocódigo pode ser entendido como uma forma intermediária de representar um algoritmo que se situa entre a linguagem natural e uma linguagem de programação. Nesta forma é utilizado um conjunto restrito de palavras-chave que são equivalentes nas linguagens de programação. Geralmente este conjunto pertence à língua nativa do programador.

Como vantagem o pseudocódigo não exige toda a rigidez sintática numa linguagem de programação, permitindo que o estudante se foque na lógica dos algoritmos e não nos detalhes de sua representação.

Outra vantagem do pseudocódigo é que o estudante não precisa ter o conhecimento prévio de nenhuma linguagem de programação.

Depois que se conseguir mais familiaridade com os algoritmos, o pseudocódigo pode ser traduzido para uma linguagem de programação.

Nos países onde idioma é o português, o pseudocódigo é conhecido como Portugol. Ao longo deste ebook teremos todos os exemplos de pseudocódigo são apresentados em Portugol.

## 1.8 Comandos de Entrada e Saída

Entrada e saída é um conceito básico no contexto de computação, quase sempre um algoritmo precisa de interatividade com o usuário, desta forma, é necessário representar as trocas de informações que precisam ocorrer entre o computador e o usuário. Essa interatividade é conseguida por meio dos comandos de entrada e saída. O nome dos comandos varia entre as linguagens de programação e pseudocódigos. Além de podermos observar o uso deste comando na maioria dos exemplos deste material, as Figuras 17 e 18 apresentam um exemplo de utilização básica dos dois comandos, de entrada e de saída.

### 1.8.1 Comando de entrada de dados (leia)

Este comando recebe os valores digitados pelos usuários, atribuindo-os às variáveis cujos nomes são listados e passados como parâmetro (é respeitada a ordem das variáveis especificadas nos parâmetros passados no comando).



## 1.8.2 Comando de saída de dados (escreva)

Este comando imprime nos dispositivos de saída (geralmente uma tela de computador) o conteúdo de tudo que é listado como parâmetro, para que o usuário possa visualizá-lo (no Portugol e em outras linguagens de programação os conteúdos, variáveis e expressões devem ser separados por vírgulas e às vezes também por vírgulas combinadas com aspas ), observe o exemplo apresentado na Figura 17 e a saída dele na Figura 18.

```
/* Copyright (C) 2014 - UNIVALI - Universidade do Vale do Itajaí
 * Este arquivo de código fonte é livre para utilização, cópia e/ou modificação
 * desde que este cabeçalho, contendo os direitos autorais e a descrição do programa,
 * seja mantido.
 *
 * Descrição:
 * Este exemplo utiliza a entrada de dados do Portugol para ler e armazenar
 * um número inteiro em uma variável. Logo após, utiliza a saída de dados para
 * exibir o número digitado.
 */

programa
{
    funcao inicio ()
    {
        inteiro numero

        escreva("Digite um número inteiro: ")
        leia(numero)

        escreva("O número digitado foi: ", numero, "\n")
    }
}
```

**Figura 17 - Código exemplo de entrada e saída de dados no Portugol Studio.**

**Fonte:** LITE – Univali em <http://lite.acad.univali.br/portugol/>

**Descrição:** Este exemplo apresenta um programa de computador que possui uma função chamada início e nela exibe uma mensagem, com o comando “escreva”, exibe uma mensagem solicitando que o usuário forneça um valor exibindo a mensagem "Digite um número inteiro:", com o comando "leia", lê o valor fornecido pelo usuário, e por final exibe o número fornecido pelo usuário numa mensagem usando o comando "escreva".

```
Digite um número inteiro: 5
O número digitado foi: 5
```

**Figura 18 - Saída exemplo de entrada e saída de dados no Portugol Studio.**

**Fonte:** o autor

**Descrição:** Exibição da saída do programa de exemplo de entrada e saída de dados, apresentando o número digitado,



no caso é exibida a frase “O número digitado foi: 5”.



## DICA IMPORTANTE!

A indentação ou recuo (do inglês, indentation) é um recurso para ressaltar e definir a estrutura de um algoritmo aumentando sua legibilidade. Use sempre indentação!

## 1.9 Representação e Armazenamento de Dados

Para que o programador possa armazenar e manipular dados no computador é necessário representá-los de alguma maneira, no contexto de programação isto é feito através de variáveis e de constantes, conforme demonstrado na Figura 19 e na Figura 20.

```
programa
{
    funcao inicio()
    {
        real numero1, numero2, soma
        leia(numero1)
        leia(numero2)
        soma = numero1 + numero2
        escreva("A soma de ", numero1, " e ", numero2, " é ", soma, "\n")
    }
}
```

→ Variáveis

Figura 19 - Código exemplo de uso de variáveis no Portugol Studio.

Fonte: o autor

**Descrição:** Este exemplo apresenta um programa de computador que possui uma função início e nela declara três variáveis do tipo real, lê duas variáveis, com o comando “leia”, e atribui à terceira variável a soma das duas primeiras, depois apresenta na tela, com o comando “escreva”, o valor calculado da terceira variável.

```
3
4
A soma de 3.0 e 4.0 é 7.0
```

Figura 20 - Saída do exemplo de uso de variáveis no Portugol Studio.

Fonte: o autor

**Descrição:** É exibida a saída que foi calculada no programa de computador do exemplo 1 de declaração de variáveis. O resultado exibido é a frase “A soma de 3.0 e 4.0 é 7.0”.



## 1.9.1 Variáveis, constantes e tipos de dados

### 1.9.1.1 Variáveis (identificadores)

No contexto de computação, uma variável ou identificador representa um endereço da memória. Nela é possível armazenar temporariamente dados de todos os tipos. Ao usarmos uma variável, estamos atribuindo um nome simbólico, um apelido, a um endereço da memória. O conteúdo de uma variável pode ser alterado, consultado ou apagado quantas vezes forem necessárias durante a execução de um algoritmo. Quando o conteúdo de uma variável é alterado, a informação anterior é perdida, a última informação armazenada na variável é a que “conta”. Uma variável armazena apenas um conteúdo de cada vez. A Figura 19 e a Figura 20 apresentam um exemplo de utilização de variáveis no Portugol Studio.

### 1.9.1.2 Declaração de variáveis

Tanto em pseudocódigo como em diversas linguagens existem muitas maneiras de se declarar uma variável, podem ser declaradas individualmente, em conjunto (do mesmo tipo), com valores já carregados, etc. Podemos conferir dois exemplos nas Figuras 21 a 24.

```
programa
{
    funcao inicio()
    {
        real numero1
        real numero2
        real divisao
        leia(numero1)
        leia(numero2)
        divisao = numero1 / numero2
        escreva("A divisão de ",numero1," por ", numero2 , " é ", divisao ,"\n")
    }
}
```

→ Variáveis

Figura 21 – Código exemplo 1 de declaração de variáveis no Portugol Studio.

Fonte: o autor

**Descrição:** Este exemplo apresenta um programa de computador que possui uma função chamada início e nela declara três variáveis, lê duas variáveis, com o comando “leia”, e atribui à terceira variável a divisão da primeira pela segunda, depois apresenta na tela, com o comando “escreva”, o valor calculado da terceira variável.

```
7
2
A divisão de 7.0 por 2.0 é 3.5
```





**Figura 22 – Saída do exemplo 1 de declaração de variáveis no Portugol Studio.**

**Fonte:** o autor

**Descrição:** É exibida a saída que foi calculada no programa de computador do exemplo 1 de declaração de variáveis. O resultado exibido é a frase “A divisão de 7.0 e 2.0 é 3.5”.

```
programa
{
    funcao inicio()
    {
        real pi = 3.1415, numero1, multiplicacao
        leia(numero1)
        multiplicacao = numero1 * pi
        escreva("A multiplicação de ",pi," por ",numero1," é ",multiplicacao,"\n")
    }
}
```

→ Variáveis

**Figura 23 – Código exemplo 2 de declaração de variáveis no Portugol Studio.**

**Fonte:** o autor

**Descrição:** Este exemplo apresenta um programa de computador que possui uma função chamada início e nela declara três variáveis do tipo real, sendo elas "pi", "numero1" e "multiplicacao". A variável "pi" tem seu valor pré-definido na declaração sendo ele o valor da constante matemática pi. Após a declaração das variáveis o programa recebe um valor fornecido pelo usuário através do comando "leia" e atribui à variável "multiplicacao" a multiplicação das outras duas variáveis. Depois da multiplicação o valor calculado é exibido numa mensagem através do comando "escreva".

```
5
A multiplicação de 3.1415 por 5.0 é 15.707500000000001
```

**Figura 24 – Saída do exemplo 2 de declaração de variáveis no Portugol Studio.**

**Fonte:** o autor

**Descrição:** É exibida a saída que foi calculada no programa de computador do exemplo 2 de declaração de variáveis. O resultado exibido é a frase “A multiplicação de 3.1415 por 5.0 é igual a 15.707500000000001”.

## 1.9.1.3 Constantes

No contexto de algoritmos constantes são endereços de memória que armazenam informações fixas, inalteráveis durante a execução do programa. No Portugol, dependendo do tipo, uma constante pode ser classificada como sendo numérica, lógica ou literal. As Figuras 25 e 26 apresentam um exemplo de constante em Portugol.



```
programa
{
    funcao inicio()
    {
        real raio, area
        const real pi = 3.1415
        leia(raio)
        area = pi * raio * raio
        escreva("A área de um círculo de raio ", raio, " é ", area)
    }
}
```

Figura 25 - Código exemplo 1 de uso de constantes no Portugol Studio.

Fonte: o autor

**Descrição:** Este exemplo apresenta um programa de computador que possui uma função chamada início e nela declara três variáveis do tipo real, “raio”, “área” e “pi”. A variável “pi” é declarada como uma constante com o valor pré-definido 3.1415. Em seguida o programa, lê a variável “raio” através do comando "leia" e atribui à variável “area” a multiplicação do valor de “raio” por ela mesma e pela constante “pi”, depois escreve na saída o resultado de “area”. Note que o valor da variável área é dado pela fórmula geométrica da área de um círculo.

```
3
A área de um círculo de raio 3.0 é 28.2735
```

Figura 26 - Saída do exemplo 1 uso de constantes no Portugol Studio.

Fonte: o autor

**Descrição:** É exibida a saída que foi calculada no programa de computador do exemplo 1 de uso de constantes. O resultado exibido é a frase “A área de um círculo de raio 3.0 é 28.2735”.

Muitas vezes a própria ferramenta de programação fornece algumas constantes importantes, por exemplo, a constante matemática PI. As Figuras 27 e 28 apresentam um exemplo que demonstra a utilização da constante PI através do uso da biblioteca de funções matemáticas do Portugol Studio. No momento vamos focar em constantes, veremos mais sobre funções e sobre bibliotecas de funções em um tópico mais adiante nesta disciplina.



```
programa
{
    inclui biblioteca Matematica
    funcao inicio()
    {
        escreva("O valor de PI é ",Matematica.PI)
    }
}
```

Figura 27 – Código exemplo 2 de uso de constantes da biblioteca de funções do Portugol Studio.

Fonte: o autor

**Descrição:** Este exemplo apresenta um programa de computador que possui uma inclusão da biblioteca de funções matemáticas do Portugol Studio e uma função chamada início e nesta, através do comando "escreva", exibe o valor da constante matemática PI fornecido pela biblioteca de funções matemáticas do Portugol Studio com um grau de precisão de 15 casas decimais.

O valor de PI é 3.141592653589793

Figura 28 – Saída do exemplo 2 de uso de constantes da biblioteca de funções no Portugol Studio.

Fonte: o autor

**Descrição:** É exibida a saída que foi solicitada no programa de computador do exemplo 2 de uso de constantes da biblioteca de funções. O resultado exibido é a mensagem "O valor de PI é 3.141592653589793".

Constantes também podem ser de outros tipos, as Figuras 29 e 30 apresentam um exemplo com um grupo de constantes do tipo cadeia, declarados de uma forma diferente, todos de uma vez.

```
programa
{
    funcao inicio()
    {
        const cadeia AC = "Acre", AL = "Alagoas", AP = "Amapá",
                     AM = "Amazonas", BA = "Bahia", CE = "Ceará",
                     DF = "Distrito Federal", ES = "Espírito Santo",
                     GO = "Goiás", MA = "Maranhão", MT = "Mato Grosso",
                     MS = "Mato Grosso do Sul", MG = "Minas Gerais",
                     PA = "Pará", PB = "Paraíba", PR = "Paraná",
                     PE = "Pernambuco", PI = "Piauí", RJ = "Rio de Janeiro",
                     RN = "Rio Grande do Norte", RS = "Rio Grande do Sul",
                     RO = "Rondônia", RR = "Roraima", SC = "Santa Catarina",
                     SP = "São Paulo", SE = "Sergipe", TO = "Tocantins"

        escreva(PE," ", BA, " e ", CE," são estados do Nordeste brasileiro.")
    }
}
```

Figura 29 – Código exemplo 3 de uso de constantes do tipo "cadeia" no Portugol Studio.

Fonte: o autor

**Descrição:** Este exemplo apresenta um programa de computador que possui uma função chamada início e nela são declaradas vinte e sete constantes do tipo "cadeia" cujos nomes são as siglas dos estados do Brasil e os valores são os respectivos nomes dos estados. Em seguida são exibidos, através do comando "escreva", os valores de três das variáveis criadas.



Pernambuco, Bahia e Ceará são estados do Nordeste brasileiro.

**Figura 30 – Saída do exemplo 3 do uso de constantes do tipo “cadeia” no Portugol Studio.**

**Fonte:** o autor

**Descrição:** É exibida a saída que foi solicitada no programa de computador do exemplo 3 de uso de constantes do tipo “cadeia” no Portugol Studio. É exibida uma mensagem usando três das constantes declaradas, “Pernambuco, Bahia e Ceará são estados do Nordeste brasileiro”.

## 1.9.1.4 Tipos de dados

Para o conceito de algoritmo, os dados são classificados em três tipos primitivos básicos: *numéricos*, *caractere* e *lógicos*. Porém alguns desses tipos se subdividem. Para os tipos numéricos existem os *inteiros* e os *reais*, para os tipos caractere (no Portugol) existem o *caracter* e a *cadeia*, uma descrição desses tipos é apresentada na Figura 31.

Tipo	Descrição
inteiro	Consiste somente no conjunto de números inteiros. Sua utilização mais comum é para representar contagens (quantidade). Podem ser positivos, negativos ou nulos.
real	Consiste no conjunto de números reais (com ou sem casas decimais). Sua utilização mais comum é para representar uma medições. No Portugol os valores deste tipo de dado são números separados por pontos e não por virgulas.
caracter	Consiste em um valor contido dentro de um conjunto de caracteres alfanuméricos. Quando são declarados números como caracteres eles tornam-se representativos e perdem a atribuição de valor. Composto de APENAS UM carácter alfanumérico ou especial.
cadeia	Consiste numa uma sequência ordenada de caracteres (símbolos) escolhidos a partir de um conjunto de caracteres alfanuméricos, sendo texto ou uma quantidade MAIOR OU IGUAL A UM de caracteres.
logico	Contém um tipo de dado usado em operações lógicas, que possui somente dois valores, que são consideradas pelo Portugol como verdadeiro e falso.

**Figura 31 – Tipos de dados primitivos no Portugol Studio.**

**Fonte:** o autor

**Descrição:** Tabela com Tipo e Descrição dos dados primitivos usados no Portugol Studio, sendo eles inteiro, real, caracter, cadeia e lógico.



Indicamos que você assista esse vídeo como forma de ampliar o conceito de Tipos de Dados Primitivos.

<https://www.youtube.com/watch?v=9ybMQQZtOys>



## 2.Competência 02 | Desenvolver um algoritmo para a realização de operações matemáticas

### 2.1 Manipulação de Dados

#### 2.1.1 Operadores de Atribuição

No contexto de algoritmos, para determinar um valor a uma variável ou até mesmo para uma expressão, utilizamos a operação de Atribuição. Depois da declaração de uma variável, a manipulação dos dados que ela irá representar é realizada através da atribuição de valores à mesma. Uma variável pode armazenar apenas um único valor por vez, sendo que toda vez que um novo valor é atribuído à mesma, o valor anterior que estava armazenado na mesma é perdido. A Figura 32 apresenta os principais operadores de atribuição do Portugol no Portugol Studio.

Operador	Descrição	Exemplo
=	Atribuição simples de um valor ou expressão a uma variável	nota1 = 0 escreva(nota1) → 0
+=	Adição do valor de uma variável ao valor de outra variável ou constante. Equivale a: variavel1 = variavel1 + variavel2.	nota1 = 5, nota2 = 4 nota1 += nota2 escreva(nota1) → 9
-=	Subtração do valor de uma variável do valor de outra variável ou constante. Equivale a: variavel1 = variavel1 - variavel2.	nota1 = 9, penalidade = 1 nota1 -= penalidade escreva(nota1) → 8
*=	Multiplicação do valor de uma variável pelo valor de outra variável ou constante. Equivale a: variavel1 = variavel1 * variavel2.	nota1 = 9, peso = 3 nota1 *= peso escreva(nota1) → 27
/=	Divisão do valor de uma variável pelo valor de outra variável ou constante. Equivale a: variavel1 = variavel1 / variavel2.	media = 28, nProvas = 4 media /= nProvas escreva(media) → 7
%=	Resto da divisão do valor de uma variável pelo valor de outra variável ou constante. Equivale a: variavel1 = variavel1 % variavel2.	resto = 7 resto %= 2 escreva(resto) → 1
++	Adição de 1 ao valor de uma variável. Equivale a: variavel1 = variavel1+1.	contador = 2 contador++ escreva(contador) → 3
--	Subtração de 1 ao valor de uma variável. Equivale a: variavel1 = variavel1-1.	contador = 2 contador-- escreva(contador) → 1

**Figura 32 – Operadores de atribuição no Portugol Studio.**

**Fonte:** o autor

**Descrição:** Tabela com operadores de atribuição no Portugol Studio: recebe, recebe adição ou recebe concatenação, recebe subtração, recebe multiplicação, recebe divisão, recebe resto da divisão, auto incremento e auto decremento.



## 2.1.2 Operadores Aritméticos Básicos

Ao desenvolvermos algoritmos é comum utilizarmos expressões matemáticas para a resolução de cálculos. Neste tópico são apresentados os operadores aritméticos básicos necessários para a maioria das expressões. Os operadores aritméticos funcionam de forma muito semelhante aos operadores na matemática, por isso, podemos entendê-los como o conjunto de símbolos que representam as operações básicas da matemática. O exemplo das Figuras 33 e 34 demonstra a utilização de operadores aritméticos referentes às quatro operações básicas, adição, subtração, multiplicação e divisão.

```
/* Copyright (C) 2014 - UNIVALI - Universidade do Vale do Itajaí
 * Este arquivo de código fonte é livre para utilização, cópia e/ou modificação
 * desde que este cabeçalho, contendo os direitos autorais e a descrição do programa,
 * seja mantido.
 *
 * Descrição:
 * Este exemplo pede ao usuário que informe dois números. Logo após, calcula e exibe:
 * a) A soma entre os números
 * b) A subtração entre os números
 * c) A multiplicação entre os números
 * d) A divisão entre os números
 */
programa
{
    funcao inicio()
    {
        real a, b, soma, sub, mult, div

        escreva("Digite o primeiro número: ")
        leia(a)

        escreva("Digite o segundo número: ")
        leia(b)

        soma = a + b // Soma os dois valores
        sub = a - b // Subtrai os dois valores
        mult = a * b // Multiplica os dois valores
        div = a / b // Divide os dois valores

        // Exibe o resultado da soma
        escreva("\nA soma dos números é igual a: ", soma)
        // Exibe o resultado da subtração
        escreva("\nA subtração dos números é igual a: ", sub)
        // Exibe o resultado da multiplicação
        escreva("\nA multiplicação dos números é igual a: ", mult)
        // Exibe o resultado da divisão
        escreva("\nA divisão dos números é igual a: ", div, "\n")
    }
}
```

**Figura 33 – Código exemplo de uso de operadores aritméticos no Portugol Studio.**

**Fonte:** LITE – Univali em <http://lite.acad.univali.br/portugol/>

**Descrição:** Este exemplo apresenta um programa de computador que possui uma função chamada início e nela declara seis variáveis do tipo real, "a", "b", "soma", "sub", "mult" e "div". Em seguida solicita ao usuário que este forneça os valores de dois números através das mensagens que são realizadas com o comando "escreva", recebendo os valores após cada mensagem através do comando "leia". Por fim, calcula o resultado das quatro operações matemáticas





básicas, soma, subtração, multiplicação e divisão atribuindo-os às variáveis restantes com o nome semelhante e exibe os resultados através de mensagens realizadas com o comando "escreva".

```
Digite o primeiro número: 4
Digite o segundo número: 2

A soma dos números é igual a: 6.0
A subtração dos números é igual a: 2.0
A multiplicação dos números é igual a: 8.0
A divisão dos números é igual a: 2.0
```

**Figura 34 – Saída do exemplo de uso de operadores aritméticos no Portugol Studio.**

**Fonte:** o autor

**Descrição:** É exibida a saída que foi calculada no programa de computador do exemplo de operadores aritméticos no Portugol Studio. O resultado exibido, baseado no fornecimento dos números 4 e 2, são as mensagens “A soma dos números é igual a 6.0”, “A subtração dos números é igual a 2.0”, “A multiplicação dos números é igual a 8.0” e “A divisão dos números é igual a 2.0”.

## 2.1.3 Prioridade dos Operadores Aritméticos Básicos

Como na matemática, existe uma prioridade de execução entre os operadores aritméticos básicos usados em algoritmos caso esses operadores sejam utilizados em conjunto numa expressão aritmética.

Num algoritmo, sempre que uma expressão aritmética precisa ser avaliada, um analisador processa a expressão dando prioridade a operadores específicos. Primeiramente as sub-expressões que usam estes operadores serão processadas e a sub-expressão inteira será substituída pelo seu valor. Em seguida a próxima sub-expressão na ordem é processada e assim por diante até que toda a expressão corresponda a apenas um valor. A ordem de prioridade na avaliação dos operadores numa expressão aritmética é conhecida como precedência de operadores.

A Figura 35 apresenta os principais operadores aritméticos e suas prioridades.

Os operadores de multiplicação, divisão, potenciação e de módulo têm prioridade de execução em relação aos operadores de adição e subtração, e acima de todos, a utilização de parênteses podem alterar a prioridade de qualquer operador numa expressão.

Embora esses operadores sejam reconhecidos como universais, sua representação pode variar entre as linguagens de programação e mesmo nos pseudocódigos e ferramentas de aprendizagem, por exemplo, a potenciação no Portugol Studio é representada usando-se uma função e em algumas





outras ferramentas a função de módulo também. Portanto, sempre é interessante consultar o material de documentação relativo à ferramenta que está sendo usada.

Operador	Descrição	Prioridade
+	Operador de soma ou adição. Geralmente vem antes da subtração.	3ª
-	Operador de subtração.	3ª
*	Operador de multiplicação.	2ª
/	Operador de divisão.	2ª
%	Operador de módulo ou resto.	2ª
^	Operador de potenciação.	2ª
( )	Parênteses, operador de prioridade. Pode determinar prioridade de termos em expressões acima de qualquer outro operador	1ª

**Figura 35 – Operadores aritméticos básicos.**

**Fonte:** o autor

**Descrição:** Tabela com Tipo, Descrição e Exemplo dos operadores aritméticos básicos, sendo eles: adição, “+”, subtração, “-”, multiplicação, “\*”, divisão, “/”, módulo ou resto da divisão inteira, “%”, potenciação, “^” e parênteses, “( )”.



Indicamos que você assista esse vídeo como forma de ampliar o conceito de Operadores Aritméticos parte 1.

<https://www.youtube.com/watch?v=Zax00WPcbec>



Indicamos que você assista esse vídeo como forma de ampliar o conceito de Operadores Aritméticos parte 2.

[https://www.youtube.com/watch?v=SRiP2R8Ue\\_o](https://www.youtube.com/watch?v=SRiP2R8Ue_o)

O exemplo das Figuras 36 e 37 demonstra a priorização dos operadores aritméticos quando usados em expressões aritméticas, tópicos que veremos mais adiante.



```

/* Copyright (C) 2014 - UNIVALI - Universidade do Vale do Itajaí
 * Este arquivo de código fonte é livre para utilização, cópia e/ou modificação
 * desde que este cabeçalho, contendo os direitos autorais e a descrição do programa,
 * seja mantido.
 *
 * Descrição:
 * Este exemplo demonstra a prioridade das operações aritméticas no Portugol. As
 * operações de multiplicação (*), divisão (/) e módulo (%) têm prioridade sobre
 * as operações de soma (+) e subtração (-). Além disso, o exemplo demonstra como
 * os parênteses podem ser utilizados para alterar esta prioridade, fazendo com
 * que uma operação de soma ocorra antes de uma operação de multiplicação.
 */
programa
{
    funcao inicio()
    {
        real resultado
        // Neste exemplo, a operação de multiplicação (*) será executada primeiro
        resultado = 5.0 + 4.0 * 2.0
        escreva("Operação: 5 + 4 * 2 = ", resultado)

        // Neste exemplo, a operação de soma (+) será executada primeiro
        resultado = (5.0 + 4.0) * 2.0
        escreva("\nOperação: (5 + 4) * 2 = ", resultado)

        /*
         * Neste exemplo, a operação de divisão (/) será executada primeiro,
         * seguida pela operação de multiplicação (*). Por último, será
         * executada a operação de soma (+)
         */
        resultado = 1.0 + 2.0 / 3.0 * 4.0
        escreva("\nOperação: 1 + 2 / 3 * 4 = ", resultado)

        /*
         * Neste exemplo, a operação de soma (+) será executada primeiro,
         * seguida pela operação de multiplicação (*). Por último, será
         * executada a operação de divisão (/).
         */
        resultado = (1.0 + 2.0) / (3.0 * 4.0)
        escreva("\nOperação: (1 + 2) / (3 * 4) = ", resultado, "\n")
    }
}

```

**Figura 36 – Código exemplo de prioridade dos operadores aritméticos no Portugol Studio.**

**Fonte:** LITE – Univali em <http://lite.acad.univali.br/portugol/>

**Descrição:** Este exemplo apresenta um programa de computador que possui uma função chamada início e nela declara uma variável do tipo real chamada "resultado", em seguida atribui a esta variável uma expressão matemática com os operadores matemáticos básicos e exibe o resultado numa mensagem através do comando "escreva". Esta operação de atribuição de expressão e exibição de mensagem é repetida mais três vezes.



```
Operação: 5 + 4 * 2 = 13.0
Operação: (5 + 4) * 2 = 18.0
Operação: 1 + 2 / 3 * 4 = 3.6666666666666665
Operação: (1 + 2) / (3 * 4) = 0.25
```

Figura 37 – Saída do exemplo de prioridade dos operadores aritméticos no Portugol Studio.

Fonte: o autor

**Descrição:** São exibidas as saídas que foram calculadas no programa de computador do exemplo de prioridade dos operadores aritméticos no Portugol Studio, sendo elas, "Operação: 5 + 4 \* 2 = 13.0", "Operação: (5 + 4) \* 2 = 18.0", "Operação: 1 + 2 / 3 \* 4 = 3.6666666666666665" e "(1 + 2) / (3 \* 4) = 0.25".

As Figuras 38 e 39 apresentam um exemplo da utilização do operador aritmético de módulo, também conhecido como resto.

```
/* Copyright (C) 2014 - UNIVALI - Universidade do Vale do Itajaí
 * Este arquivo de código fonte é livre para utilização, cópia e/ou modificação
 * desde que este cabeçalho, contendo os direitos autorais e a descrição do programa,
 * seja mantido.
 *
 * Descrição:
 * Este exemplo demonstra o uso do operador aritmético módulo '%'.
 * Esta operação retorna o resto de uma divisão.
 */
programa
{
    funcao inicio()
    {
        inteiro valor_modulo
        valor_modulo = 7%3

        escreva ("O resultado de 7%3 é ", valor_modulo, "\n")
    }
}
```

Figura 38 – Código exemplo de uso do operador aritmético de módulo no Portugol Studio.

Fonte: LITE – Univali em <http://lite.acad.univali.br/portugol/>

**Descrição:** Este exemplo apresenta um programa de computador que possui uma função chamada início e nela declara uma variável do tipo inteiro chamada "valor\_modulo", atribui a ela o cálculo do resto da divisão de 7 por 3 e exibe o resultado numa mensagem na tela através do comando "escreva".

```
O resultado de 7%3 é 1
```

Figura 39 – Saída do exemplo de uso do operador aritmético de módulo no Portugol Studio.

Fonte: o autor

**Descrição:** É exibida a saída que foi calculada no programa de computador do exemplo de uso do operador aritmético de módulo no Portugol Studio. Retorna a mensagem "O resultado de 7%3 é 1", interpretando-se "O resto da divisão de 7 por 3 é igual a 1".



## 2.1.4 Operadores Relacionais

É comum, na construção de algoritmos, precisamos comparar uma variável, constante, valor ou expressão em relação à outra variável, constante, valor ou expressão. Para estas tarefas, são usados os operadores relacionais. Expressões que utilizam este tipo de operador resultam num valor lógico (Verdadeiro ou Falso).



### DICA IMPORTANTE!

Os dados comparados por operadores relacionais devem ser sempre do mesmo tipo primitivo.



### DICA IMPORTANTE!

Qualquer tipo primitivo de dado pode ser comparado através de um operador relacional.

A Figura 40 apresenta os principais operadores relacionais.

Operação	Símbolo	Exemplo	Resultado
Igual	==	$12 == 10 + 2$	Verdadeiro
Diferente	!=	$9 != 9$	Falso
Maior	>	$2 > 5$	Falso
Menor	<	$5 + 3 < 20 - 2$	Verdadeiro
Maior ou igual	>=	$7 >= 5 + 2$	Verdadeiro
Menor ou igual	<=	$25 <= 90$	Verdadeiro

**Figura 40 – Operadores relacionais básicos.**

**Fonte:** o autor

**Descrição:** Tabela com Tipo, Descrição e Exemplo dos operadores relacionais básicos, sendo eles: “igual a”, “diferente de”, “maior que”, “menor que”, “maior ou igual a” e “menor ou igual a”.

As Figuras 41 e 42 apresentam um exemplo que demonstra como os operadores relacionais realizam comparações entre valores do mesmo tipo (no caso, numérico).



```
/* Copyright (C) 2014 - UNIVALI - Universidade do Vale do Itajaí
 * Este arquivo de código fonte é livre para utilização, cópia e/ou modificação
 * desde que este cabeçalho, contendo os direitos autorais e a descrição do programa,
 * seja mantido.
 *
 * Descrição:
 * Este exemplo compara dois números utilizando os operadores relacionais.
 * São atribuídos valores a duas variáveis e em seguida são executadas
 * verificações com os operadores relacionais.
 */
programa
{
    funcao inicio()
    {
        inteiro a = 5, b = 3

        //Comparação entre A e B utilizando o operador igual a
        se(a == b){
            escreva("A é igual a B \n")
        }

        //Comparação entre A e B utilizando o operador diferente a
        se(a != b){
            escreva("A é diferente de B \n")
        }

        //Comparação entre valor A e B utilizando o operador maior que
        se(a > b){
            escreva("A é maior que B \n")
        }

        //Comparação entre valor A e B utilizando o operador menor que
        se(a < b){
            escreva("A é menor que B \n")
        }

        //Comparação entre A e B utilizando o operador maior ou igual a
        se(a >= b){
            escreva("A é maior ou igual a B \n")
        }

        //Comparação entre A e B utilizando o operador menor ou igual a
        se(a <= b){
            escreva("A é menor ou igual a B \n")
        }
    }
}
```

**Figura 41 – Código exemplo de uso de operadores relacionais no Portugol Studio.**

**Fonte:** LITE – Univali em <http://lite.acad.univali.br/portugol/>

**Descrição:** Este exemplo apresenta um programa de computador que possui uma função chamada início e nela declara duas variáveis do tipo inteiro com valores pré-definidos, sendo elas "a" = 5 e "b" = 3. Em seguida são realizadas seis comparações entre essas variáveis através da combinação de estruturas condicionais (que veremos mais adiante) e expressões relacionais (que utilizam operadores relacionais), sendo elas, comparação de "igualdade", comparação de "diferença", comparação de "maior que", comparação de "menor que", comparação de "maior ou igual a" e comparação de "menor ou igual a". Cada estrutura de repetição combinada à expressão relacional exibe, caso a expressão seja verdadeira, uma mensagem dentro de seu bloco através do comando escreva.



```
A é diferente de B
A é maior que B
A é maior ou igual a B
```

**Figura 42 – Saída do exemplo de uso de operadores relacionais no Portugol Studio.**

**Fonte:** o autor

**Descrição:** É exibida a saída que foi calculada no programa de computador do exemplo de uso de operadores relacionais no Portugol Studio. Retorna as mensagens as quais as expressões relacionais sejam verdadeiras, no caso, "A é diferente de B", "A é maior que B" e "A é maior ou igual a B".

## 2.1.5 Expressões Aritméticas

No contexto de algoritmos uma expressão tem o mesmo significado de uma expressão da matemática comum, um conjunto de valores numéricos (inteiro ou real) que se relacionam por meio de operadores aritméticos, de atribuição e relacionais. Esses valores numéricos geralmente são acessados por meio de constantes ou de variáveis. O conjunto de valores e operadores aritméticos formam uma fórmula que quando solucionada fornece um resultado em específico que geralmente é numérico.

### DICA IMPORTANTE!

Divisões de números pelo valor 0 (zero) ou de raízes quadradas de números negativos não possuem um valor definido matematicamente para computadores. A execução de expressões gera erro. Evite estas situações.

### DICA IMPORTANTE!

Deixar parênteses não pareados (não fechados) nas expressões aritméticas é um erro difícil de se localizar depois de finalizado o código. Evite esta situação.

As Figuras 43 e 44 apresentam um exemplo de execução de uma expressão.





```
programa
{
    funcao inicio()
    {
        real variavel1 = 3, variavel2 = 5, variavel3 = 7, expressao
        expressao = ((variavel1 + variavel2) * 2.0) / (variavel3 + 1)
        escreva("( ( ",variavel1," + ",variavel2," ) * 2) / ( ",variavel3," + 1 ) = ",expressao)
    }
}
```

**Figura 43 – Código exemplo de uso de expressões aritméticas no Portugol Studio.**

**Fonte:** o autor

**Descrição:** Este exemplo apresenta um programa de computador que possui uma função chamada início e nela declara quatro variáveis do tipo real "expressao" e, com valor pré-determinado, "variavel1" = 3, "variavel2" = 5, "variavel3" = 7. Em seguida é atribuída à variável "expressao" a expressão  $((\text{variavel1} + \text{variavel2}) * 2.0) / (\text{variavel3} + 1)$ . Em seguida e exibe o resultado numa mensagem na tela através do comando "escreva".

```
( ( ( 3.0 + 5.0 ) * 2 ) / ( 7.0 + 1 ) ) = 2.0
```

**Figura 44 – Saída do exemplo de uso de expressões aritméticas no Portugol Studio.**

**Fonte:** o autor

**Descrição:** É exibida a saída que foi calculada no programa de computador do exemplo de uso de expressões aritméticas no Portugol Studio. Retorna o resultado da expressão aritmética na mensagem "(((3.0 + 5.0) \* 2) / (7.0 + 1)) = 2.0".

## 2.1.6 Funções Primitivas

Para a execução de muitas tarefas é necessária a realização de cálculos matemáticos que são muito difíceis e muitas vezes impossíveis, mesmo fazendo uso de operadores e expressões aritméticas. Como solução muitas ferramentas fornecem uma biblioteca de funções primitivas pré-definidas que podem ser usadas pelos programas. Desta forma estas funções podem ser usadas em conjunto com as expressões aritméticas ou de maneira única. A Figura 45 apresenta algumas das principais funções primitivas. Cada ferramenta de programação, que use uma linguagem de programação real ou pseudocódigo, tem um grupo dessas funções, no Portugol Studio são classificadas como a biblioteca de funções matemáticas.



Função	Descrição
Raiz(x) ou SQRT(x)	Raiz quadrada
ABS(x)	Valor absoluto
ROUND(x)	Valor arredondado
SIN(x)	Função seno
COS(x)	Função cosseno
TAN(x)	Função tangente
LOG(x)	Função logarítmica

**Figura 45 – Funções primitivas.**

**Fonte:** o autor

**Descrição:** Tabela com Função e Descrição das funções primitivas, sendo eles: “Raiz” ou “SQRT” (função raiz quadrada), “ABS” (função valor absoluto), “ROUND” (função valor arredondado), “SIN” (função seno), “COS” (função cosseno), “TAN” (função tangente) e “LOG” (função logaritmo).

## 2.1.7 Operadores Lógicos

Os operadores lógicos, também conhecidos como conectivos lógicos por serem capazes de unirem duas ou mais expressões simples em uma expressão composta, permitem que mais de uma condição seja testada em uma única expressão, ou seja, pode-se fazer mais de uma comparação (teste) ao mesmo tempo, tendo como resultado de saída um valor lógico (verdadeiro ou falso). Um operador lógico pode ser binário (comparação entre duas expressões) ou unário (negação de uma expressão). A Figura 46 apresenta os principais operadores lógicos e suas características.

Operador	Tipo	Operação	Descrição	Prioridade
OU (OR)	Binário	Disjunção	Apenas uma das expressões precisa ser verdadeira para retornar verdadeiro.	3
E (AND)	Binário	Conjunção	Só retorna verdadeiro se todas as expressões forem verdadeiras.	2
NÃO (NOT)	Unário	Negação	inverte o resultado lógico final da expressão, de verdadeiro passa para falso e vice versa.	1

**Figura 46 – Operadores lógicos.**

**Fonte:** o autor

**Descrição:** Tabela com Operador, Tipo, Operação, Descrição e Prioridade dos operadores lógicos, sendo eles: “OU”, “E” e “NÃO”.





## 2.1.8 Expressões Lógicas

Expressões lógicas são aquelas nas quais o valor resultante sempre é um valor lógico, verdadeiro ou falso. Elas podem ser compostas por operadores relacionais, aritméticos, por variáveis e constantes. As expressões lógicas também podem ser compostas por resultados de expressões aritméticas e outras expressões lógicas. As Figuras 47 e 48 apresentam um exemplo de uso de expressões lógicas no Portugol Studio.

```

programa
{
    funcao inicio()
    {
        real variavel1 = 2, variavel2 = 5, variavel3 = 7
        logico resultado, A, B, C, D
        A = (variavel1 < variavel2) // Verdadeiro
        B = (variavel3 > variavel2) // Verdadeiro
        C = (variavel1 > variavel2) // Falso
        D = (variavel1 > variavel3) // Falso
        resultado = (A) e (B)
        escreva(" ", variavel1, " < ", variavel2, " ) e ( ", variavel3, " > ", variavel2, " ) = ", resultado)

        resultado = (A) e (C)
        escreva("\n\n( ", variavel1, " < ", variavel2, " ) e ( ", variavel1, " > ", variavel2, " ) = ", resultado)

        resultado = (C) e (D)
        escreva("\n\n( ", variavel1, " > ", variavel2, " ) e ( ", variavel1, " > ", variavel3, " ) = ", resultado)

        resultado = (A) ou (B)
        escreva("\n\n( ", variavel1, " < ", variavel2, " ) ou ( ", variavel3, " > ", variavel2, " ) = ", resultado)

        resultado = (A) ou (D)
        escreva("\n\n( ", variavel1, " < ", variavel2, " ) ou ( ", variavel1, " > ", variavel3, " ) = ", resultado)

        resultado = (C) ou (D)
        escreva("\n\n( ", variavel1, " > ", variavel2, " ) ou ( ", variavel1, " > ", variavel3, " ) = ", resultado)

        resultado = nao(A) e (B)
        escreva("\n\n não( ", variavel1, " < ", variavel2, " ) e ( ", variavel3, " > ", variavel2, " ) = ", resultado)

        resultado = (A) e nao(C)
        escreva("\n\n( ", variavel1, " < ", variavel2, " ) e não( ", variavel1, " > ", variavel2, " ) = ", resultado)

        resultado = nao(A) ou (D)
        escreva("\n\n não( ", variavel1, " < ", variavel2, " ) ou ( ", variavel1, " > ", variavel3, " ) = ", resultado)

        resultado = nao(C) ou (D)
        escreva("\n\n não ( ", variavel1, " > ", variavel2, " ) ou ( ", variavel1, " > ", variavel3, " ) = ", resultado)
    }
}
    
```

Figura 47 – Código exemplo de uso de expressões lógicas no Portugol Studio.

Fonte: o autor

**Descrição:** Este exemplo apresenta um programa de computador que possui uma função chamada início e nela declara três variáveis do tipo real com valores pré-determinados variavel1 = 2, variavel2 = 5, variavel3 = 7, e cinco variáveis do tipo logico, A = (variavel1 < variavel2), B = (variavel3 > variavel2), C = (variavel1 > variavel2), D = (variavel1 > variavel3) e "resultado", sendo este último sem valor pré-definido. Em seguida é atribuída uma expressão lógica à variável "resultado" e é exibida na tela uma mensagem com o seu valor através do comando "escreva", sendo que esta operação de atribuição é repetida dez vezes com expressões lógicas diferentes. "resultado" = ((A) e (B)), "resultado" =



((C) e (D)), "resultado" = ((A) ou (B)), "resultado" = ((A) ou (D)), "resultado" = ((C) ou (D)), "resultado" = (nao(A) e (B)), "resultado" = ((A) e nao(C)), "resultado" = (nao(A) ou (D)), "resultado" = (nao(C) ou (D))).

```
( 2.0 < 5.0 ) e ( 7.0 > 5.0 ) = verdadeiro
( 2.0 < 5.0 ) e ( 2.0 > 5.0 ) = falso
( 2.0 > 5.0 ) e ( 2.0 > 7.0 ) = falso
( 2.0 < 5.0 ) ou ( 7.0 > 5.0 ) = verdadeiro
( 2.0 < 5.0 ) ou ( 2.0 > 7.0 ) = verdadeiro
( 2.0 > 5.0 ) ou ( 2.0 > 7.0 ) = falso
nao( 2.0 < 5.0 ) e ( 7.0 > 5.0 ) = falso
( 2.0 < 5.0 ) e nao( 2.0 > 5.0 ) = verdadeiro
nao( 2.0 < 5.0 ) ou ( 2.0 > 7.0 ) = falso
nao ( 2.0 > 5.0 ) ou ( 2.0 > 7.0 ) = verdadeiro
```

**Figura 48 – Saída do exemplo de uso de expressões lógicas no Portugol Studio.**

**Fonte:** o autor

**Descrição:** É exibida a saída que foi calculada no programa de computador do exemplo de expressões lógicas no Portugol Studio. Retorna as mensagens "( 2.0 < 5.0 ) e ( 7.0 > 5.0 ) = verdadeiro", "( 2.0 < 5.0 ) e ( 2.0 > 5.0 ) = falso", "( 2.0 > 5.0 ) e ( 2.0 > 7.0 ) = falso", "( 2.0 < 5.0 ) ou ( 7.0 > 5.0 ) = verdadeiro", "( 2.0 < 5.0 ) ou ( 2.0 > 7.0 ) = verdadeiro", "( 2.0 > 5.0 ) ou ( 2.0 > 7.0 ) = falso", "nao( 2.0 < 5.0 ) e ( 7.0 > 5.0 ) = falso", "( 2.0 < 5.0 ) e nao( 2.0 > 5.0 ) = verdadeiro", "nao( 2.0 < 5.0 ) ou ( 2.0 > 7.0 ) = falso" e "nao( 2.0 > 5.0 ) ou ( 2.0 > 7.0 ) = verdadeiro".

## 2.1.9 Tabela Verdade

A tabela verdade é uma tabela matemática muito utilizada na Álgebra Booleana, mostra a relação lógica entre duas ou mais expressões e seus respectivos resultados, facilita a visualização, e faremos

Operação 1	Operação 2	Operação 1 e Operação 2	Operação 1 ou Operação 2
Verdadeiro	Verdadeiro	Verdadeiro	Verdadeiro
Verdadeiro	Falso	Falso	Verdadeiro
Falso	Verdadeiro	Falso	Verdadeiro
Falso	Falso	Falso	Falso

**Figura 49 – Tabela Verdade 1, operadores lógicos E e OU.**

**Fonte:** o autor

**Descrição:** Tabela Verdade 1 com as colunas "Operação 1", "Operação 2", "Operação 1 e Operação 2" e "Operação 1 ou Operação 2".



o uso dela para compreendermos melhor os operadores lógicos. Pode ser uma ferramenta muito útil para visualizar o resultado de expressões e fluxos de execução nos algoritmos. As Figuras 49, 50 e 51 apresentam alguns exemplos de tabela verdade.

Operação 1	Operação 2	Operação 1 e Operação 2	Operação 1 ou Operação 2
$30 \geq 11$	$3 < 7$	Verdadeiro	Verdadeiro
$5 \leq 5$	$3 > 12$	Falso	Verdadeiro
$5 < 2$	$9 > 1$	Falso	Verdadeiro
$3+5 > 15$	$4/2 < 1$	Falso	Falso

**Figura 50 – Tabela Verdade 2, exemplo dos operadores lógicos E e OU.**

**Fonte:** o autor

**Descrição:** Tabela Verdade 2 com as colunas “Operação 1”, “Operação 2”, “Operação 1 e Operação 2” e “Operação 1 ou Operação 2”. O conteúdo das linhas são expressões lógicas de exemplo e valores lógicos correspondentes.

Operação 1	Não Operação 1	Exemplo Não Operação 1	Resultado
Verdadeiro	Falso	não( $15 \leq 20$ )	Falso
Falso	Verdadeiro	não ( $7 > 20$ )	Verdadeiro

**Figura 51 – Tabela Verdade 3, exemplo do operador lógico NÃO.**

**Fonte:** o autor

**Descrição:** Tabela Verdade 3 com as colunas “Operação 1”, “Não Operação 1”, “Exemplo Não Operação 1” e “Resultado”.

## 2.2 Dicas adicionais para construção de programas

Agora que vimos o básico do básico sobre algoritmos, esta seção do material é reservada para mencionar alguns aspectos relevantes de programação.

Muitas ferramentas de programação exibem o código do programa de forma a permitir que blocos de código sejam expostos ou ocultos de acordo com a indentação, semelhante a uma estrutura de árvores, onde se tem um sinal de “-” (menos) onde um bloco de código está exibido e um sinal de “+” (mais) onde um bloco de código está oculto (contraído). Este recurso é interessante quando o código é muito extenso e precisamos de uma visão mais simplificada, assim, blocos desnecessários podem ser ocultos da nossa vista para podermos focar no trecho que importa.

A Figura 52 apresenta um exemplo de um bloco expandido e de um bloco oculto.

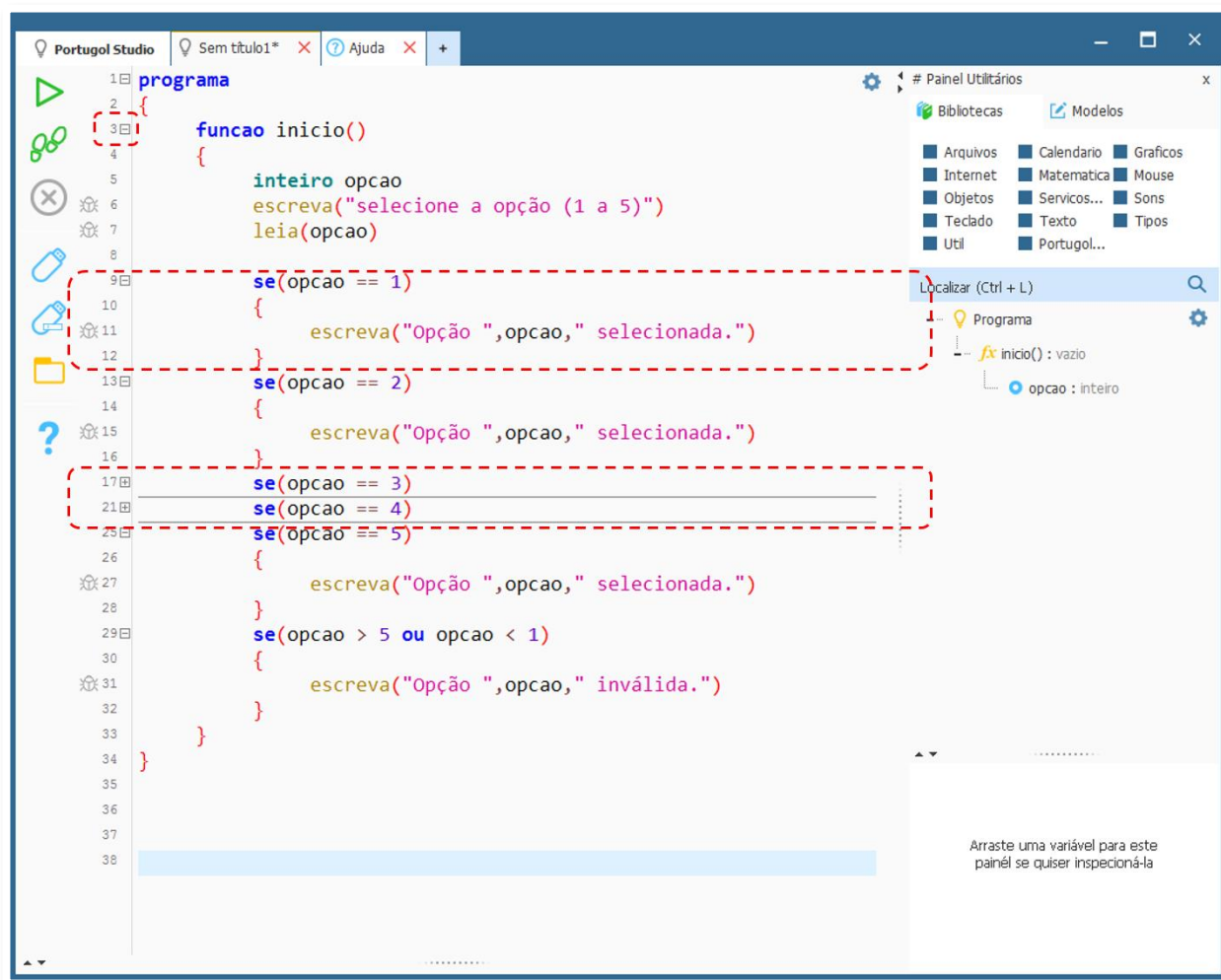


Figura 52 – Exemplo de resumo e exposição de blocos de código no Portugol Studio

Fonte: o autor

**Descrição:** Este exemplo apresenta um programa de computador que possui uma função chamada início e nela declara uma variável do tipo inteiro chamada "opcao", logo após, através dos comandos "escreva" e "leia", apresenta uma solicitação para o usuário fornecer o valor dentre as opções de 1 a 5 para valorar "opcao". Em seguida é criado um grupo com várias estruturas condicionais "SE" que verificam o conteúdo de "opcao" para cada valor possível segundo o intervalo dado. Além de possuir um comando escreva para exibir o valor da variável, cada bloco de estrutura "SE" possui, na interface, ao lado esquerdo da sua linha inicial, um sinal de mais se estiver oculto (área tracejada com início na linha dezessete e fim na linha vinte e um) ou de menos se estiver expandido (área tracejada com início na linha nove). Além disso pode-se observar também a linha três que também tem um sinal de menos indicando que o bloco da função inteira pode ser contraído ou oculto.

Outra dica interessante é a utilização de teclas de atalho. Toda ferramenta de programação possui teclas de atalho para facilitar e agilizar o processo de programação. Muitas ferramentas seguem o padrão básico de teclas de atalho do Windows. De acordo com o exemplo da Figura 53, embora o material do Portugol Studio não mencione, ao usar "Ctrl + S" o programa abre a rotina de salvar o arquivo selecionado, seja um novo arquivo, seja um arquivo já existente.

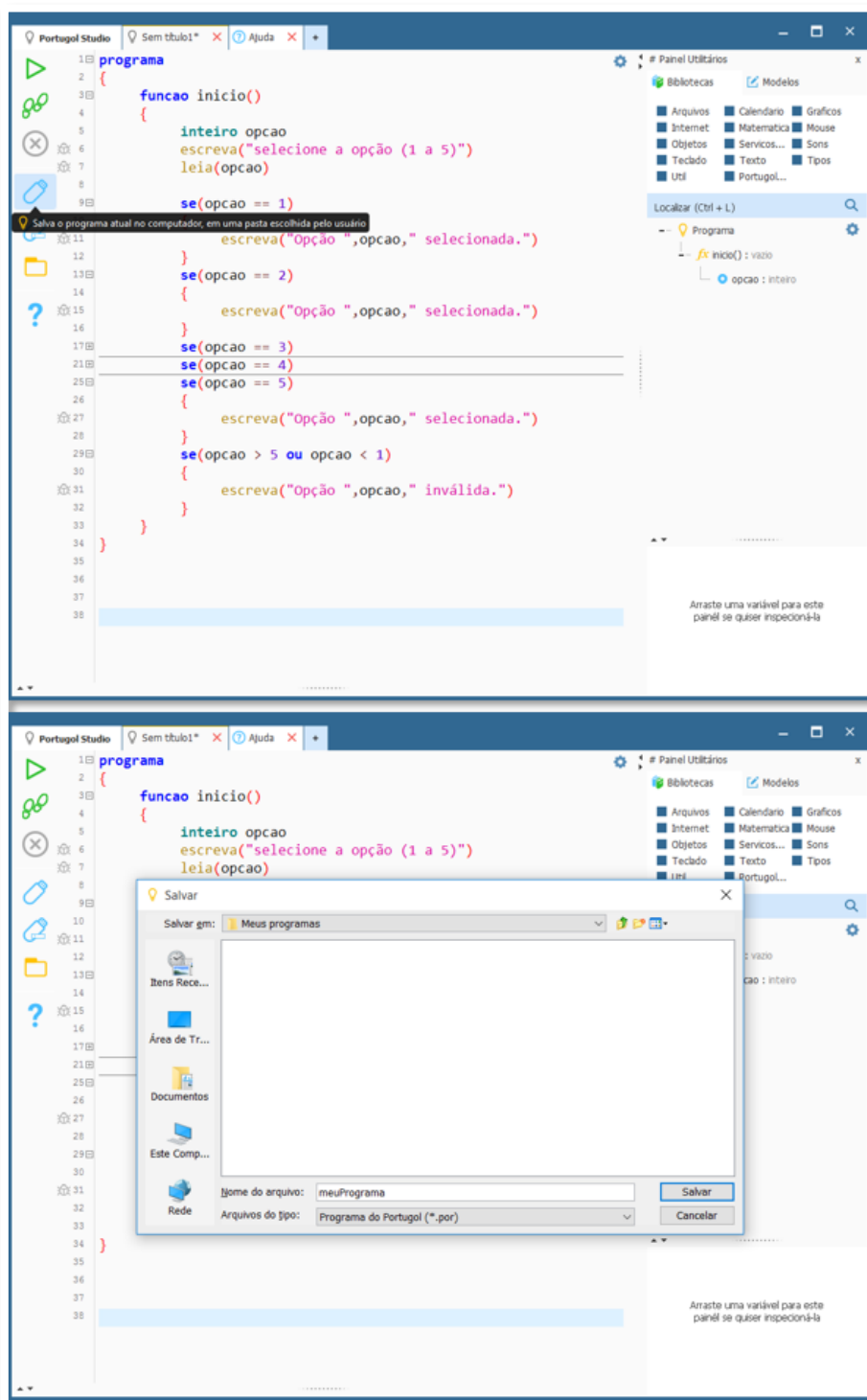


Figura 53 – Exemplo de uso de teclas de atalho no Portugol Studio

Fonte: o autor

**Descrição:** Neste exemplo são apresentadas duas telas do programa Portugol Studio, uma em cima da outra, representando a mesma tela em duas etapas distintas, com o mesmo código mostrado no exemplo da Figura 52. A tela superior apresenta o código e realça o botão "Salvar", localizado ao lado direito na tela de interface com o ícone de um pendrive. A tela inferior apresenta a mesma tela, com uma subtelas chamada através e exibida após a utilização das teclas de atalho "Ctrl + S", demonstrando a mesma função do botão apresentado da tela superior.





## 3.Competência 03 | Desenvolver um algoritmo para resolução de um problema utilizando estrutura de decisão

Nos exemplos de código que vimos até agora, foi possível resolver apenas os problemas, nos quais todas as instruções eram executadas seguindo a ordem do algoritmo (de cima para baixo), portanto, puramente sequenciais. Agora começaremos a estudar os desvios condicionais. As Estruturas de Controle Condicionais, também conhecidas como Desvios Condicionais, permitem que determinados comandos sejam executados ou não, dependendo do resultado de um teste realizado (condição). Desta forma elas nos permitem “pular” blocos de código os quais não desejamos que sejam executados em determinadas situações, criando diferentes fluxos de execução e resultados possíveis num algoritmo. Cada estrutura condicional realiza um teste lógico (cujo resultado final da expressão é 0 ou 1, verdadeiro ou falso) para decidir se seu bloco de código é executado ou não.

### 3.1 Estrutura de controle se então / if ... then

Esta estrutura é utilizada quando queremos que uma instrução ou um conjunto de instruções seja executado apenas se uma condição específica for verdadeira, normalmente ela vem acompanhada de um comando, ou seja, se determinada condição for satisfeita pelo comando SE (IF) ENTÃO (THEN) execute determinado comando. Vale lembrar que a sintaxe do comando varia entre as ferramentas, às vezes subentendendo e escondendo o ENTÃO. A Figura 54 exemplifica a sintaxe básica de uma estrutura SE ENTÃO no Portugol Studio, as Figuras 55 e 56 apresentam um exemplo de utilização da estrutura SE ENTÃO na mesma ferramenta.

```
se(teste lógico)
{
    bloco de código que deve ser executado
    caso o teste lógico seja verdadeiro
}
```

Figura 54– Sintaxe da estrutura “SE ENTÃO” no Portugol Studio.

Fonte: o autor

**Descrição:** Pseudocódigo genérico da estrutura “SE ENTÃO” com o teste lógico delimitado por parênteses e o bloco de código a ser executado caso o teste seja verdadeiro delimitado por chaves.



```
programa
{
    funcao inicio()
    {
        real mediaFinal
        leia(mediaFinal)
        se(mediaFinal >= 7)
        {
            escreva("Aluno aprovado.")
        }
        se(mediaFinal < 7)
        {
            escreva("Aluno reprovado.")
        }
    }
}
```

Figura 55 – Código exemplo de uso de estrutura “SE ENTÃO” no Portugol Studio.

Fonte: o autor

**Descrição:** Este exemplo apresenta um programa de computador que possui uma função chamada início e nela declara uma variável do tipo real chamada "mediaFinal" e recebe do usuário um valor através do comando "leia". Em seguida são construídos dois blocos de estrutura condicional "SE ENTÃO", um em seguida do outro, que aplicam, cada um, um teste lógico através de operadores relacionais e casos seus testes lógicos sejam verdadeiros exibem uma mensagem através do comando "escreva". O primeiro "SE ENTÃO" usa o teste lógico (mediaFinal >= 7), o segundo "SE ENTÃO" utiliza o teste lógico (mediaFinal < 7), neste caso são cenários mutuamente excludentes e complementares, sendo impossível os dois ocorrerem juntos.

7.5  
Aluno aprovado.

Figura 56 – Saída do exemplo de estrutura “SE ENTÃO” no Portugol Studio.

Fonte: o autor

**Descrição:** É exibida a saída que foi calculada no programa de computador do exemplo de uso de estrutura “SE ENTÃO” no Portugol Studio. No caso o usuário forneceu o valor da variável "mediaFinal" = 7.5 e a mensagem "Aluno aprovado." é exibida.

Quando precisarmos combinar estruturas “SE ENTÃO” como tarefas opostas ou complementares é importante ter atenção às expressões das estruturas envolvidas. No exemplo das Figuras 55 e 56, se a expressão da segunda estrutura utilizasse o sinal de “<=” no lugar de “<” o algoritmo teria uma situação onde o aluno que tirou 7 receberia as duas mensagens, o que seria bem confuso.





## DICA IMPORTANTE!

Muita atenção em todos os sinais de todas as expressões das estruturas condicionais.

### 3.2 Estrutura de controle se então senão / if ... then ... else

Muitas vezes, queremos executar um bloco de comandos caso uma condição seja verdadeira e outro bloco de comandos caso essa condição seja falsa. Para isso, podemos utilizar as instruções SE ENTÃO e SENÃO. O funcionamento é o seguinte: Se a condição do teste lógico for verdadeira o bloco de instruções do SE é executado e depois o fluxo de execução do programa ignora, “pula”, o bloco de instruções do SENÃO (ELSE). Caso contrário, se a condição do teste lógico for falsa, o fluxo de execução do programa ignora o bloco de instruções do SE e o bloco de instruções do SENÃO será executado. A instrução SENÃO não pode ser utilizada sozinha no código sem estar vinculada a uma instrução SE.

A Figura 57 exemplifica a sintaxe básica de uma estrutura SE ENTÃO SENÃO no Portugol Studio, as Figuras 58 e 59 apresentam um exemplo de utilização da estrutura SE ENTÃO SENÃO na mesma ferramenta.

```
se(teste lógico)
{
    bloco de código que deve ser executado
    caso o teste lógico seja verdadeiro
}
senao
{
    bloco de código que deve ser executado
    caso o teste lógico NÃO seja verdadeiro
}
```

Figura 57– Sintaxe da estrutura “SE SENÃO” no Portugol Studio.

Fonte: o autor

**Descrição:** Pseudocódigo genérico da estrutura “SE ENTÃO SENÃO” com o teste lógico delimitado por parênteses e o bloco de código a ser executado caso o teste seja verdadeiro (“SE”) e um bloco de código alternativo a ser



executado caso o teste seja falso ("SENÃO"). Ambos os blocos de código são delimitados por chaves. O "SENÃO" sempre é precedido do "SE".

```
/* Copyright (C) 2014 - UNIVALI - Universidade do Vale do Itajaí *
 * Este arquivo de código fonte é livre para utilização, cópia e/ou modificação
 * desde que este cabeçalho, contendo os direitos autorais e a descrição do programa,
 * seja mantido.
 *
 * Descrição:
 * Este exemplo pede ao usuário que informe uma letra (caracter). Logo após
 * verifica se a letra digitada é uma vogal ou uma consoante e exibe o resultado
 * ao usuário.
 */
programa
{
    funcao inicio ()
    {
        caracter letra

        escreva("Digite uma letra: ")
        leia(letra)

        // O Portugol diferencia caracteres minúsculos e maiúsculos,
        // portanto é preciso verificar ambos os casos

        se
        (
            letra == 'A' ou letra == 'E' ou letra == 'I' ou letra == 'O' ou letra == 'U' ou
            letra == 'a' ou letra == 'e' ou letra == 'i' ou letra == 'o' ou letra == 'u'
        )
        {
            escreva("\nA letra '", letra, "' é uma vogal\n")
        }
        senao
        {
            escreva("\nA letra '", letra, "' é uma consoante\n")
        }
    }
}
```

Figura 58 – Código exemplo de uso de estrutura “SE SENÃO” no Portugol Studio.

Fonte: LITE – Univali em <http://lite.acad.univali.br/portugol/>

**Descrição:** Este exemplo apresenta um programa de computador que possui uma função chamada início e nela declara uma variável do tipo caracter chamada "letra", solicita que o usuário forneça uma letra através de uma mensagem na tela usando o comando "escreva" e recebe o valor através do comando "leia". Em seguida usa uma estrutura condicional "SE NÃO" com um teste lógico verificando se a letra fornecida é uma vogal, caso o resultado do teste seja verdadeiro, exibe uma mensagem apresentando a letra fornecida classificando-a como vogal, caso o resultado do teste não seja verdadeiro, exibe uma mensagem apresentando a letra fornecida classificando-a como consoante.

Digite uma letra: e

A letra 'e' é uma vogal

Figura 59 – Saída do exemplo de uso de estrutura “SE SENÃO” no Portugol Studio.

Fonte: o autor



**Descrição:** É exibida a saída que foi calculada no programa de computador do exemplo de uso de estrutura condicional “SE SENÃO” no Portugol Studio. No caso a letra digitada pelo usuário, “e”, é identificada como uma vogal e é exibida a mensagem “A letra 'e' é uma vogal”.z

## 3.3 Estrutura de controle se-então-senão aninhada

Existem casos em que os caminhos para uma tomada de decisão acabam formando uma espécie de árvore com diversas ramificações, onde cada caminho (ramo) é um conjunto de ações. Nestes casos, é preciso testar sequencialmente diversas condições para se realizar a tomada de decisão, aninhando (ou encadeando) condições dentro de condições, podendo esse tipo de estrutura ser constituído de diversos níveis de condições. As Figuras 60 e 61 apresentam um exemplo de estrutura SE ENTÃO aninhada.

```
/* Copyright (C) 2014 - UNIVALI - Universidade do Vale do Itajaí
 * Este arquivo de código fonte é livre para utilização, cópia e/ou modificação
 * desde que este cabeçalho, contendo os direitos autorais e a descrição do programa,
 * seja mantido.
 *
 * Descrição:
 * Este exemplo pede ao usuário que informe um número inteiro. Logo após, exibe uma
 * mensagem indicando se o número informado é positivo, negativo ou igual a zero.
 */
programa
{
    funcao inicio()
    {
        inteiro numero

        escreva("Digite um número inteiro: ")
        leia(numero)

        se(numero > 0) // Verifica se o número é positivo
        {
            escreva("O número é positivo")
        }
        senao se(numero < 0) // Verifica se o número é negativo
        {
            escreva("O número é negativo")
        }
        senao // Se não é positivo nem negativo, só pode ser igual a zero
        {
            escreva("O número é igual zero")
        }

        escreva("\n")
    }
}
```

Figura 60 – Código exemplo de uso de estruturas “SE SENÃO” aninhadas no Portugol Studio.

Fonte: LITE – Univali em <http://lite.acad.univali.br/portugol/>

**Descrição:** Este exemplo apresenta um programa de computador que possui uma função chamada início e nela declara uma variável do tipo inteiro chamada "numero", solicita que o usuário forneça um valor através de uma mensagem na tela usando os comandos "escreva" e recebe o valor para "numero" através do comando



"leia". Em seguida usa duas estruturas condicionais "SE NÃO SE" e "SE NÃO" aninhadas. Na cláusula "SE" é realizado um teste lógico para ver se o número é maior que 0, caso seja, exibe a mensagem "O número é positivo", caso contrário, dentro da cláusula "SE NÃO SE" é criada uma estrutura "SE NÃO" com o teste lógico de sua cláusula "SE" verifica se o número é menor que 0, caso seja, exibe a mensagem "O número é negativo", caso contrário, exibe a mensagem "O número é igual a zero".

```
Digite um número inteiro: 9
O número é positivo
```

**Figura 61 – Saída do exemplo de uso de estruturas “SE SENÃO” aninhadas no Portugol Studio.**

**Fonte:** o autor

**Descrição:** É exibida a saída que foi calculada no programa de computador do exemplo de uso de estruturas “SE SENÃO” aninhadas no Portugol Studio. Retorna a mensagem de acordo com o número digitado, no caso, foi digitado o número “9”, e é exibida a mensagem "O número é positivo".

## 3.4 Estrutura de controle caso selecione / select ... case

Para a tomada de uma decisão, existem casos em que apenas os desvios condicionais não são suficientes, ou seja, podem ocorrer casos em que é necessária uma série de testes sobre um mesmo bloco. Este tipo de estrutura é chamada de estrutura de decisão do tipo CASO SELECIONE, também conhecida como CASO SEJA ou ESCOLHA CASO.

A proposta do CASO SELECIONE é uma solução mais elegante para casos de múltiplas escolhas condicionais, levando o fluxo do programa direto ao bloco de código correto, dependendo do valor de uma variável de verificação. Da mesma forma que a estrutura de condição SE SENÃO permite que executemos algum comando quando o resultado do seu teste não é verdadeiro, a estrutura de condição CASO SELECIONE também nos oferece essa opção, chamada opção padrão. O bloco de comandos dentro da opção padrão (CASO CONTRÁRIO no Portugol Studio) será executado caso nenhuma das condições fornecidas seja verdadeira.

A Figura 62 exemplifica a sintaxe básica de uma estrutura CASO SELECIONE no Portugol Studio, as Figuras 63 e 64 apresentam um exemplo de utilização da estrutura CASO SELECIONE na mesma ferramenta.



```
escolha(parametro)
{
    caso valor1:
        bloco de código que deve ser executado
        caso o valor do parametro seja igual ao valor1.
        pare
    caso valor2:
        bloco de código que deve ser executado
        caso o valor do parametro seja igual ao valor2.
        pare
    caso valor3:
        bloco de código que deve ser executado
        caso o valor do parametro seja igual ao valor3.
        pare
    caso contrario:
        bloco de código que deve ser executado
        caso o valor do parametro NÃO seja igual
        nenhum dos valores nos casos anteriores.
}
```

Figura 62– Sintaxe da estrutura “CASO SELECIONE” no Portugol Studio.

Fonte: o autor

**Descrição:** Pseudocódigo genérico da estrutura “CASO SELECIONE” inicia com a cláusula “ESCOLHA” acompanhada da variável de parâmetro, neste caso “parametro”, delimitada entre parênteses, em seguida vem o conteúdo delimitado entre chaves, repetem-se diversos blocos de código com a cláusula “CASO”, cada uma acompanhada de um dos possíveis valores da variável “parametro” seguida de “:”. Dentro de cada um dos blocos “CASO” deve-se colocar o conjunto de instruções que devem ser executadas para o possível valor de “parametro” associado. Deve-se finalizar o conjunto de instruções dentro de cada um dos blocos “CASO” com a cláusula “PARE”. Por fim, um último bloco deve ser criado com a cláusula “CASO CONTRAÍRO” seguida de “:” com as instruções para caso o valor de “parametro” fornecido não seja nenhum dos previstos nos blocos anteriores, este bloco também é conhecido como bloco de opção padrão ou simplesmente padrão.



Indicamos que você assista esse vídeo como forma de ampliar o conceito de estrutura ESCOLHA CASO.

<https://www.youtube.com/watch?v=RQN3vZGKcp4>



```
/* Copyright (C) 2014 - UNIVALI - Universidade do Vale do Itajaí
 * Este arquivo de código fonte é livre para utilização, cópia e/ou modificação
 * desde que este cabeçalho, contendo os direitos autorais e a descrição do programa,
 * seja mantido.
 *
 * Descrição:
 * Este exemplo ilustra o uso da instrução "escolha". Para isso, o programa pede
 * ao usuário que escolha uma opção e exibe uma frase correspondente à opção
 * escolhida.
 */
programa
{
    funcao inicio()
    {
        inteiro opcao

        escreva("1) Elogio \n")
        escreva("2) Ofensa \n")
        escreva("3) Sair \n\n")
        escreva("Escolha uma opção: ")

        leia(opcao)
        limpa()

        escolha (opcao)
        {
            caso 1:
                escreva ("Voce é lindo(a)!")
                pare // Impede que as instruções do caso 2 sejam executadas
            caso 2:
                escreva ("Voce é um monstro!")
                pare // Impede que as instruções do caso 2 sejam executadas
            caso 3:
                escreva ("Tchau!")
                pare
            caso contrario: // Será executado para qualquer opção diferente de 1, 2 ou 3
                escreva ("Opção Inválida !")
        }
        escreva("\n")
    }
}
```

Figura 63 – Código exemplo de uso de estrutura “CASO SELECIONE” no Portugol Studio.

Fonte: LITE – Univali em <http://lite.acad.univali.br/portugol/>

**Descrição:** Este exemplo apresenta um programa de computador que possui uma função chamada início e nela declara uma variável do tipo inteiro chamada "opcao", em seguida utiliza o comando "escreva" para apresentar 3 opções "1) Elogio", "2) Ofensa" e "3) Sair" e solicita que o usuário escolha uma opção. Através do comando "leia" recebe a opção do usuário e inicia o bloco "CASO SELECIONE" com a variável "opcao" como parâmetro, em cada bloco de opção da estrutura é exibida uma mensagem diferente através do comando "escreva", no o bloco da opção 1 a mensagem é "Você é lindo(a)!", no bloco da opção 2 a mensagem é "Você é um monstro!", no bloco da opção 3 a mensagem é "Tchau!" e no bloco de opção padrão é exibida a mensagem "Opção inválida!".





```
1) Elogio  
2) Ofensa  
3) Sair
```

```
Escolha uma opção: 1
```

```
Voce é lindo(a)!
```

**Figura 64 – Saída do exemplo de uso de estrutura “CASO SELECIONE” no Portugol Studio.**

**Fonte:** o autor

**Descrição:** É exibida a saída que foi calculada no programa de computador do exemplo de uso de estrutura condicional “CASO SELECIONE” no Portugol Studio. No caso, a opção digitada foi “1”, o programa retorna a mensagem “Você é lindo(a)!”.



### DICA IMPORTANTE!

Sempre use a opção padrão da estrutura CASO SELECIONE, ela retorna informações interessantes em caso de nenhum dos testes (CASOS) funcionar.



### DICA IMPORTANTE!

A parada de execução PARE dentro dos CASOS evita que os testes seguintes sejam executados poupando processamento tornando seu algoritmo mais eficiente.





## 4. Competência 04 | Desenvolver um algoritmo para resolução de um problema utilizando estrutura de repetição

Comandos de repetição, também conhecidos como laços de repetição ou LOOPS são usados quando precisamos que um determinado conjunto de instruções ou comandos sejam executados um número definido ou indefinido de vezes, ou enquanto uma determinada condição seja atendida. Assim como as estruturas de decisão, toda estrutura de repetição (LOOP) executa um teste lógico para determinar se continua ou não repetindo o conjunto de instruções dentro de seu bloco de código. Os LOOPS dividem-se em dois tipos básicos de estruturas de repetição:

- **Repetição Determinada:** quando, de forma prévia, se tem o número de vezes que uma determinada sequência de comandos deve ser executada, um LOOP predefinido;
- **Repetição Indeterminada:** quando não se tem um número pré-determinado do número de vezes que a estrutura de comandos deve ser executada. Mesmo sem o conhecimento deste número, ele estará vinculado a uma determinada condição (teste lógico).



### DICA IMPORTANTE!

Cada tipo de laço se comporta de forma diferente, sendo mais adequado para determinadas soluções.

### 4.1 Estrutura de repetição para-faça

Esta estrutura é considerada um LOOP pré-definido e também é largamente conhecida por sua versão na língua inglesa, FOR. Para controlar as repetições (iterações) esta estrutura utiliza uma variável do tipo inteiro que implicitamente serve de “contador” para o número de iterações que foram executadas, incrementando seu valor a cada execução e comparando seu valor com o número de repetições pré-estabelecido no teste lógico. O incremento a cada iteração também é pré-estabelecido, por padrão seu valor é 1, ou seja, o valor contador cresce de 1 em 1, além disso pode-se usar outros valores de acordo com a necessidade.



A Figura 65 apresenta a sintaxe básica da estrutura PARA FAÇA no Portugol Studio. As Figuras 66 e 67 apresentam um exemplo de utilização de laço do tipo PARA FAÇA no Portugol Studio. No exemplo é implementado um somatório que combina seu valor com o do índice de repetição

```
para(inicio do contador ; condição de parada ; incremento do contador)
{
    bloco de código que deve ser executado caso o a condição de
    parada (teste lógico) seja verdadeiro. O “faça” está implícito
    neste bloco.
}
```

Figura 65– Sintaxe da estrutura “PARA FAÇA” no Portugol Studio.

Fonte: o autor

**Descrição:** Pseudocódigo genérico da estrutura “PARA FAÇA” com o valor inicial do contador, o teste lógico ou condição de parada e o valor de incremento, todos delimitados por um parênteses e separados por “;” e o bloco de código a ser executado caso o teste seja verdadeiro delimitado entre chaves “{}”.

```
/* Copyright (C) 2014 - UNIVALI - Universidade do Vale do Itajaí
 * Este arquivo de código fonte é livre para utilização, cópia e/ou modificação
 * desde que este cabeçalho, contendo os direitos autorais e a descrição do programa,
 * seja mantido.
 *
 * Descrição:
 * Este exemplo pede ao usuario que informe um valor inteiro. Logo após, calcula e
 * exibe a soma dos números de 1 e até o número digitado.
 */
programa
{
    funcao inicio()
    {
        inteiro soma = 0, numero, contador

        escreva("Digite o número até o qual deseja somar: ")
        leia(numero)
        escreva("\n")
        // Repete até o contador atingir o valor informado pelo usuário

        para (contador = 0; contador <= numero; contador ++ )
        {
            soma = soma + contador // Soma o valor atual do contador
            escreva("Na ", contador, "ª iteração o valor da variável soma é: ", soma)
            escreva("\n")
        }
        escreva("\n")
        escreva("A soma de 1 até ", numero, " é: ", soma, "\n")
    }
}
```

Figura 66 – Código exemplo de uso de estrutura “PARA FAÇA” no Portugol Studio.



**Fonte:** LITE – Univali em <http://lite.acad.univali.br/portugol/>

**Descrição:** Este exemplo apresenta um programa de computador que possui uma função chamada início e nela declara três variáveis do tipo inteiro chamadas "soma", "numero" e "contador", sendo "soma" com o valor pré-definido 0, em seguida, com o comando "escreva", solicita que o usuário forneça um número e recebe o valor com o comando "leia".

Em seguida inicia uma estrutura de repetição "PARA FAÇA" com a cláusula "PARA" seguida de seu teste lógico entre parênteses e seus parâmetros separados por ";", sendo eles o valor inicial de "contador" = 0, a condição de parada que "contador" > "numero" (logo enquanto "contador" <= "numero" o laço deve se repetir) e o valor de incremento de "contador" sendo 1 (que é expresso pela expressão "contador ++"). Dentro do corpo da estrutura, delimitado por chaves "{}" a variável "soma" recebe o seu próprio valor somado ao valor da variável "contador" e em seguida, através do comando "escreva", é exibido na tela uma mensagem com o número da iteração(repetição) e o valor da variável soma. Depois da estrutura de repetição, através do comando "escreva", é exibida a mensagem "A soma de 1 até 'numero' é igual a 'soma'".

```
Digite o número até o qual deseja somar: 15

Na 0ª iteração o valor da variável soma é: 0
Na 1ª iteração o valor da variável soma é: 1
Na 2ª iteração o valor da variável soma é: 3
Na 3ª iteração o valor da variável soma é: 6
Na 4ª iteração o valor da variável soma é: 10
Na 5ª iteração o valor da variável soma é: 15
Na 6ª iteração o valor da variável soma é: 21
Na 7ª iteração o valor da variável soma é: 28
Na 8ª iteração o valor da variável soma é: 36
Na 9ª iteração o valor da variável soma é: 45
Na 10ª iteração o valor da variável soma é: 55
Na 11ª iteração o valor da variável soma é: 66
Na 12ª iteração o valor da variável soma é: 78
Na 13ª iteração o valor da variável soma é: 91
Na 14ª iteração o valor da variável soma é: 105
Na 15ª iteração o valor da variável soma é: 120

A soma de 1 até 15 é: 120
```

**Figura 67 – Saída do exemplo de uso de estrutura “PARA FAÇA” no Portugol Studio.**

**Fonte:** o autor

**Descrição:** É exibida a saída que foi calculada no programa de computador do exemplo de uso de estrutura de repetição “PARA FAÇA” no Portugol Studio. No caso, a opção digitada foi "15", o programa retorna 16 vezes a mensagem "Na 'contador' iteração o valor da soma é: 'soma'", sendo "contador" com valor inicial 0 e incrementado em 1 a cada iteração, exibindo no total de mensagens valores crescentes de 0 a 15, e "soma" o valor da soma no momento de cada repetição, exibindo valores de 0 a 120. E por fim exibe a mensagem "A soma de 1 até 'numero' é igual a 'soma'", sendo "numero" o valor fornecido "15" e "soma" o valor final "120".

## 4.2 Estrutura de repetição para-faça aninhada

Da mesma forma como vimos que é possível ter uma estrutura condicional SE dentro de outra, também podemos ter uma estrutura de repetição dentro de outra, dependendo da necessidade do problema a ser resolvido. Pode ser necessária uma estrutura de repetição dentro de outra do mesmo tipo ou de um tipo diferente, ou vice-versa, as combinações são infinitas. Nas Figuras 68 e 69



podemos ver um exemplo de uma estrutura de repetição PARA FAÇA dentro de outro PARA FAÇA, que é bastante utilizado para leitura e escrita de Matrizes.

```
programa
{
    funcao inicio()
    {
        inteiro contador1, contador2, numeroLinhas = 3, numeroColunas = 4
        para(contador1 = 0; contador1 <= numeroLinhas; contador1++)
        {
            para(contador2 = 0; contador2 <= numeroColunas; contador2++)
            {
                escreva(" ", contador1+contador2, " ")
            }
            escreva("\n") // pula linha
        }
    }
}
```

Figura 68 – Código exemplo de uso de estrutura “PARA FAÇA” aninhada no Portugol Studio.

Fonte: o autor

**Descrição:** Este exemplo apresenta um programa de computador que possui uma função chamada início e nela declara quatro variáveis do tipo inteiro chamadas "contador1", "contador2", "numeroLinhas" e "numeroColunas", sendo estas duas últimas com os valores pré-definidos 3 e 4. Em seguida começa a primeira estrutura de repetição, com o teste lógico tendo "contador1" = 0, "contador1" <= "numeroLinhas" e "contador1 ++". Dentro da primeira estrutura de repetição inicia-se a segunda, com o teste lógico "contador2" = 0, "contador2" <= "numeroColunas" e "contador2 ++". Dentro da segunda estrutura de repetição, através do comando "escreva" é exibida a soma das variáveis "contador1" e "contador2". Após fechar a segunda estrutura, ainda dentro da primeira, através do comando "escreva" e do parâmetro "\n", o programa pula uma linha na exibição e assim fecha a primeira estrutura.

0	1	2	3	4
1	2	3	4	5
2	3	4	5	6
3	4	5	6	7

Figura 69 – Saída do exemplo de uso de estrutura “PARA FAÇA” aninhada no Portugol Studio.

Fonte: o autor

**Descrição:** É exibida a saída que foi calculada no programa de computador do exemplo de uso de estrutura condicional “PARA FAÇA” aninhada no Portugol Studio. Retorna uma matriz numérica com 4 linhas e 5 colunas, sendo na primeira linha os valores "0, 1, 2, 3, e 4", na segunda linha os valores "1, 2, 3, 4 e 5", na terceira linha os valores "2, 3, 4, 5 e 6" e na quarta linha os valores "3, 4, 5, 6 e 7".

## 4.3 Estruturas de repetição (enquanto-faça e faça-enquanto)

Estes LOOP são do tipo Repetição Indeterminada, ou seja, são usados principalmente quando não se sabe com antecedência a quantidade de repetições que precisam ser realizadas. Estes tipos de LOOP possuem um *critério de parada* ou condição de controle expressa através de uma



expressão lógica que é testada a cada ciclo de repetição para determinar se o LOOP continua a se repetir ou não.

## 4.3.1 Estrutura de repetição enquanto-faça

Também conhecido como Repetição Indeterminada com Validação Inicial. Nesta estrutura, caso o resultado da expressão lógica (teste) seja VERDADEIRO, as instruções dentro do bloco de código da estrutura serão executadas. Depois da execução das instruções, o teste é realizado novamente, se o resultado da expressão for FALSO o loop é encerrado e o fluxo de execução do algoritmo segue para a próxima linha. A Figura 70 apresenta a sintaxe básica de uma estrutura ENQUANTO FAÇA no Portugol Studio. As Figuras 71 e 72 apresentam um exemplo de utilização do laço ENQUANTO FAÇA no Portugol Studio. No exemplo, o usuário fornece 10 valores que são somados a cada iteração para, após o fim do ciclo de repetições, então ser realizado o cálculo da média.

```
enquanto(condição de parada)
{
    bloco de código que deve ser executado caso o a condição de
    parada (teste lógico) seja verdadeiro. O "faça" está implícito
    neste bloco.
}
```

Figura 70– Sintaxe da estrutura “ENQUANTO FAÇA” no Portugol Studio.

Fonte: o autor

**Descrição:** Pseudocódigo genérico da estrutura “ENQUANTO FAÇA”, exibe a cláusula “ENQUANTO” seguida do teste lógico ou condição de parada delimitado entre parênteses e do bloco de código a ser executado caso o teste seja verdadeiro delimitado entre chaves "{}".



```
programa
{
    funcao inicio()
    {
        inteiro contador = 1

        real numero, media, soma = 0.0

        // Laço que verifica se já foram informados 10 valores

        enquanto(contador <= 10)
        {
            escreva("Digite o ", contador, "º número: ")
            leia(numero)

            soma = soma + numero // A variavel soma é o acumulador deste exemplo
            escreva("Na ", contador, "ª iteração o valor da variável soma é: ", soma)
            escreva("\n\n")
            contador = contador + 1 // Incrementa o contador
        }

        media = soma / 10
        escreva("A média dos números é: ", media, "\n")
    }
}
```

Figura 71 – Código exemplo de uso de estrutura “ENQUANTO FAÇA” no Portugol Studio.

Fonte: o autor

**Descrição:** Este exemplo apresenta um programa de computador que possui uma função chamada início e nela declara uma variável do tipo inteiro chamada "contador" e três variáveis do tipo real chamadas "numero" "media" e "soma", sendo "soma" com o valor predefinido 0, em seguida vem a estrutura de repetição “ENQUANTO FAÇA” com a cláusula "ENQUANTO" e seu teste lógico entre parênteses verificando se "contador" <= 10. Depois do teste lógico da estrutura de repetição vem o corpo delimitado entre chaves "{}", nele, através do comando "escreva", e usando "contador" como índice, é solicitado ao usuário que forneça um número. O número solicitado é armazenado na variável "numero" através do comando "leia". Em seguida a variável "soma" recebe o seu valor acrescido do valor da variável "numero" para então, através do comando "escreva" ser exibida uma mensagem com o valor de "soma" na iteração "contador". O contador é incrementado de 1 e o corpo da estrutura é fechado. Após a estrutura de repetição a variável "media" recebe o valor de soma dividido por 10 e, através do comando escreva é exibida uma mensagem com o valor da média dos números fornecidos.





```
Digite o 1º número: 4
Na 1ª iteração o valor da variável soma é: 4.0

Digite o 2º número: 5
Na 2ª iteração o valor da variável soma é: 9.0

Digite o 3º número: 6
Na 3ª iteração o valor da variável soma é: 15.0

Digite o 4º número: 7
Na 4ª iteração o valor da variável soma é: 22.0

Digite o 5º número: 4
Na 5ª iteração o valor da variável soma é: 26.0

Digite o 6º número: 5
Na 6ª iteração o valor da variável soma é: 31.0

Digite o 7º número: 6
Na 7ª iteração o valor da variável soma é: 37.0

Digite o 8º número: 4
Na 8ª iteração o valor da variável soma é: 41.0

Digite o 9º número: 3
Na 9ª iteração o valor da variável soma é: 44.0

Digite o 10º número: 4
Na 10ª iteração o valor da variável soma é: 48.0

A média dos números é: 4.8
```

**Figura 72 – Saída do exemplo de uso de estrutura “ENQUANTO FAÇA” no Portugol Studio.**

**Fonte:** o autor

**Descrição:** É exibida a saída que foi calculada no programa de computador do exemplo de uso de estrutura condicional “ENQUANTO FAÇA” no Portugol Studio. Retorna, sempre que uma iteração do laço é executada, a mensagem de solicitação de fornecimento do número da iteração, e o valor da soma na iteração. Foram fornecidos a sequência de números 4, 5, 6, 7, 8, 4, 5, 6, 4, 3 e 4, com a soma final igual a 48. Depois da exibição das mensagens geradas na estrutura de repetição é exibida a mensagem "A média dos números é: 4.8".

## 4.3.2 Estrutura de repetição faça-enquanto

Também conhecida como Repetição Indeterminada com Validação Final. Este comando de repetição executa o bloco de instruções antes de testar se o teste lógico é verdadeiro a cada ciclo, ou seja, o teste da condição é realizado apenas ao final da estrutura. Logo, a diferença da estrutura FAÇA ENQUANTO para a estrutura ENQUANTO FAÇA é que a primeira será sempre executada pelo menos uma vez, mesmo que o resultado do teste lógico seja falso. A Figura 73 apresenta a sintaxe básica de uma estrutura FAÇA ENQUANTO no Portugol Studio. As Figuras 74 e 75 apresentam um exemplo de utilização do laço FAÇA ENQUANTO.





```
faca
```

```
{
```

bloco de código que deve ser executado primeira vez sem a condição de parada (teste lógico) e a partir de então só será executado caso o teste seja verdadeiro. O “faça” NÃO está implícito neste bloco.

```
}
```

```
enquanto(condição de parada)
```

Figura 73– Sintaxe da estrutura “FAÇA ENQUANTO” no Portugol Studio.

Fonte: o autor

**Descrição:** Pseudocódigo genérico da estrutura “FAÇA ENQUANTO”, exibe a cláusula "FACA" seguida do bloco de código a ser executado delimitado entre chaves "{}" e, abaixo destes, a cláusula "ENQUANTO" seguida da condição de parada delimitada entre parênteses "()".

```
programa
```

```
{
```

```
funcao inicio()
```

```
{
```

```
inteiro idade
```

```
faca
```

```
{
```

```
escreva ("Informe sua idade (valores aceitos de 5 a 150): ")
```

```
leia (idade)
```

```
}
```

```
enquanto (idade < 5 ou idade > 150)
```

```
// A partir deste ponto do código é garantido que a idade
```

```
// terá um valor válido e não causará erros inesperados
```

```
escreva ("\nCorreto!\n")
```

```
}
```

```
}
```

Figura 74 – Código exemplo de uso de estrutura “FAÇA ENQUANTO” no Portugol Studio.

Fonte: o autor

**Descrição:** Este exemplo apresenta um programa de computador que possui uma função chamada início e nela declara uma variável do tipo inteiro chamada "idade". Em seguida é iniciada uma estrutura de repetição "FAÇA ENQUANTO" e, após a cláusula "FACA", dentro do corpo da estrutura, delimitado entre chaves "{}", através do comando "escreva", é solicitado que o usuário informe a sua idade delimitando que o intervalo aceito é de 5 a 150 e o valor é recebido na variável "idade" através do comando "leia". Depois do conteúdo do corpo da estrutura é posicionada a cláusula "ENQUANTO" seguida da condição de parada delimitada entre parênteses, sendo esta condição a expressão lógica "'idade' < 5 ou 'idade' > 150". Após a estrutura de repetição é exibida a mensagem "Correto!" através do comando "escreva".

```
Informe sua idade (valores aceitos de 5 a 150): 10
```

```
Correto!
```

Figura 75 – Saída do exemplo de uso de estrutura “FAÇA ENQUANTO” no Portugol Studio.



**Fonte:** o autor

**Descrição:** É exibida a saída que foi calculada no programa de computador do exemplo de uso de estrutura condicional “FAÇA ENQUANTO” no Portugol Studio. No caso, na primeira iteração, foi exibida a mensagem "Informe sua idade (valores entre 5 e 150)", foi fornecido o número 10 e foi retornada a mensagem "Correto !".

## 4.4 Dicas sobre estruturas de repetição.

Quando a condição de parada de uma determinada estrutura de repetição nunca é alcançada, as instruções do laço são executadas indefinidamente, e o programa nunca chega ao seu fim. Tal comportamento é algo que deve ser evitado em programação (na grande maioria das situações) e, por ser um problema tão recorrente, recebe um nome especial: laço infinito. A Figura 76 apresenta um comparativo com as diferenças entre as estruturas PARA FAÇA, ENQUANTO FAÇA e FAÇA ENQUANTO.

Estrutura de Repetição	Teste lógico em todas as execuções	Número de repetições pré-determinado
PARA FAÇA	SIM	SIM
ENQUANTO FAÇA	SIM	NÃO
FAÇA ENQUANTO	NÃO	NÃO

**Figura 76 – Tabela comparativa entre as estruturas de repetição apresentadas.**

**Fonte:** o autor

**Descrição:** Tabela com as colunas Estrutura de repetição, obrigatoriedade de Teste lógico em todas as execuções e presença de número de repetições pré-determinado. Além do cabeçalho das colunas a estrutura possui três linhas com os valores a seguir: linha 1, "PARA FAÇA", SIM, SIM; linha 2, "ENQUANTO FAÇA", SIM, NÃO; linha 3, "FAÇA ENQUANTO", NÃO, NÃO.



### DICA IMPORTANTE!

Em LOOPS infinitos, o algoritmo nunca chega ao seu fim, o que ocasiona uma série de problemas na maioria dos casos. Evite esta situação.



## DICA IMPORTANTE!

Em algoritmos com LOOPS, sejam do tipo PARA FAÇA, ENQUANTO FAÇA ou FAÇA ENQUANTO, é comum surgir a necessidade do uso variáveis do tipo contador e/ou acumulador.

## 4.5 Conceitos adicionais – Vetores, Matrizes, Procedimentos, Funções, Recursividade e Depuração

Nesta seção serão abordados conceitos adicionais presentes em todas as linguagens de programação. O universo de programação ainda tem muitos recursos a serem descobertos, por exemplo, até agora vimos que uma variável armazena um único valor por vez, porém existem maneiras mais práticas e eficientes de se lidar com esses problemas.

### 4.5.1 Vetores

Vetores, também conhecidos como Arrays são variáveis que podem armazenar um conjunto de valores indexadas (que possuem um índice). Podemos entendê-los como uma variável dividida em diversos "pedaços" ou "espaços", onde cada espaço é identificado através de um número, referente à posição no vetor em questão. O número de cada posição do vetor é chamado de índice e o número de índices, além de determinar as posições para leitura e escrita dentro de um vetor, determina o número de dimensões dele. Geralmente vetores são uni ou bidimensionais (matrizes, que veremos mais adiante). Um array é uma variável, ou seja, é referenciado através de um nome, e neste são associados os índices.

O conceito de vetores nas várias linguagens de programação é sempre o mesmo. Porém, em algumas linguagens alguns detalhes podem ser diferentes. Nas linguagens C e Java, por exemplo, o índice inicial é sempre zero. Já nas linguagens Pascal e R, o índice inicial é definido pelo próprio programador. A Figura 77 apresenta a sintaxe básica da declaração de um vetor no Portugol Studio. A Figura 78 apresenta uma representação gráfica que descreve suas estruturas de 3 vetores, “*alunos*”, “*notas*” e “*situação*”.



`tipo_do_vetor nome_do_vetor[quantidade_de_elementos_do_vetor]`

Figura 77– Sintaxe de “VETORES” no Portugol Studio.

Fonte: o autor

**Descrição:** Pseudocódigo genérico da declaração de “VETORES”, começa com o tipo de variável do vetor, seguido do nome e a quantidade de elementos delimitada entre colchetes “[ ]”.

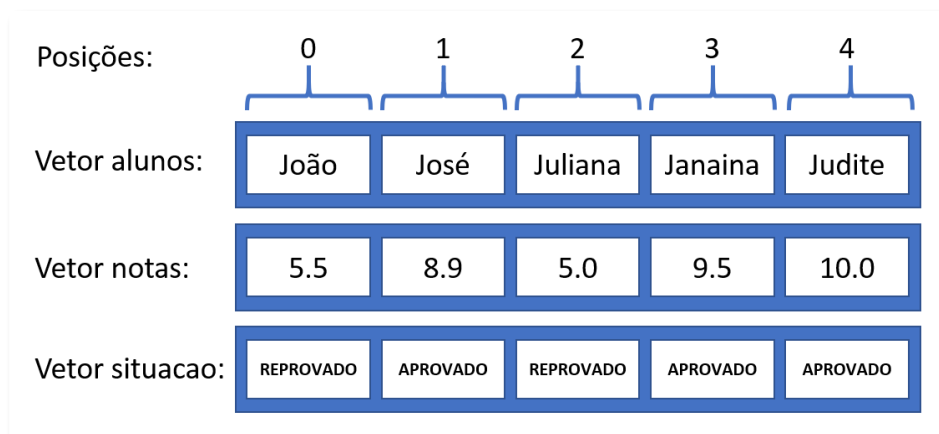


Figura 78– Exemplo gráfico de estruturas de “VETORES”.

Fonte: o autor

**Descrição:** Três vetores, "alunos", "notas" e "situacao", representados por três retângulos longos, um em cima do outro, com cinco retângulos menores subdividindo cada vetor e representando as posições ou espaços que os vetores possuem. Cada posição possui um índice, e estes vão de 0 a 4. Os vetores estão preenchidos. "alunos" possui "João", "José", "Juliana", "Janaina" e "Judite". "notas" possui "5.5", "8.9", "5.0", "9.5" e "10.0". "situacao" possui "reprovado", "aprovado", "reprovado", "aprovado" e "aprovado". Pode-se interpretar que o aluno "João" tem a nota "5.5" e a situação "reprovado".



## DICA IMPORTANTE!

No Portugol o índice de vetores e de matrizes começa com 0. Ao trabalhar com estas estruturas fique atento.



## DICA IMPORTANTE!

Podemos declarar vetores e matrizes apenas com as dimensões ou já atribuindo os seus valores.



As Figuras 79 e 80 apresentam um exemplo da utilização de vetores no Portugol Studio, nele três vetores de tipos diferentes são criados, podemos observar que não existe apenas uma forma de se declarar um vetor. A Figura 78 explica graficamente como os dados resultantes do algoritmo ficam distribuídos nos vetores criados.

```
programa
{
    funcao inicio()
    {
        real notas[] = {5.5,8.9,5.0,9.5,10.0}
        cadeia alunos[] = {"João","José","Juliana","Janaina","Judite"}, situacao[5]
        inteiro contador
        para(contador = 0; contador<5; contador++)
        {
            se(notas[contador]>7)
            {
                situacao[contador] = "APROVADO"
            }
            senao
            {
                situacao[contador] = "REPROVADO"
            }
        }
        para(contador = 0; contador<5; contador++)
        {
            escreva("ALUNO: ",alunos[contador],"\n")
            escreva("NOTA: ",notas[contador],"\n")
            escreva("SITUAÇÃO: ",situacao[contador],"\n\n")
        }
    }
}
```

Figura 79 – Código exemplo de uso de “VETORES” no Portugol Studio.

Fonte: o autor

**Descrição:** Este exemplo apresenta um programa de computador que possui uma função chamada início e nela declara dois vetores. O primeiro vetor é do tipo real e é chamado "notas", atribui a este os valores "5.5", "8.9", "5.0", "9.5" e "10.0". O segundo vetor é do tipo cadeia e é chamado "alunos", atribui a este os valores "João", "José", "Juliana", "Janaina" e "Judite". Logo após, declara uma variável do tipo inteiro chamada "contador". Em seguida inicia uma estrutura de repetição "PARA" com os parâmetros ("contador" = 0; "contador" < 5; "contador" ++), dentro do corpo desta estrutura de repetição é criada uma estrutura condicional "SE SENÃO" com o teste lógico da cláusula "SE" (notas[contador] > 7) e no seu corpo situacao[contador] = "APROVADO" e no corpo da cláusula "SENÃO" situacao[contador] = "REPROVADO". Depois do fechamento da primeira estrutura de repetição "PARA" é iniciada outra com os mesmos parâmetros ("contador" = 0; "contador" < 5; "contador" ++), dentro do corpo desta estrutura, através do comando "escreva" são exibidos os alunos, as notas e a situação dos alunos extraíndo estas informações das posições dos vetores com o índice igual ao valor de "contador"



```
ALUNO: João  
NOTA: 5.5  
SITUAÇÃO: REPROVADO  
  
ALUNO: José  
NOTA: 8.9  
SITUAÇÃO: APROVADO  
  
ALUNO: Juliana  
NOTA: 5.0  
SITUAÇÃO: REPROVADO  
  
ALUNO: Janaina  
NOTA: 9.5  
SITUAÇÃO: APROVADO  
  
ALUNO: Judite  
NOTA: 10.0  
SITUAÇÃO: APROVADO
```

Figura 80 – Saída do exemplo de uso de “VETORES” no Portugol Studio.

**Fonte:** o autor

**Descrição:** É exibida a saída que foi calculada no programa de computador do exemplo de uso de “VETORES” no Portugol Studio. No caso, são exibidos o nome, a nota e a situação de cada aluno. ALUNO: João, NOTA: 5.5, Situação: REPROVADO. ALUNO: José, NOTA: 8.9, Situação: APROVADO. ALUNO: Juliana, NOTA: 5.0, Situação: REPROVADO. ALUNO: Janaina, NOTA: 9.5, Situação: APROVADO. ALUNO: Judite, NOTA: 10.0, Situação: APROVADO.

Outro exemplo de utilização de vetores muito interessante que poderíamos ver é o da Figura 29 (no tópico constantes, na competência 1) combinado com os conceitos de estruturas de repetição e vetores, as Figuras 81 e 82 apresentam um exemplo de utilização dos 3 conceitos combinados, constantes, laços e vetores.

```
programa
{
    funcao inicio()
    {
        const cadeia estados[27] = {"Acre (AC) - Rio Branco", "Alagoas (AL) - Maceió",
                                     "Amapá (AP) - Macapá", "Amazonas (AM) - Manaus",
                                     "Bahia (BA) - Salvador", "Ceará (CE) - Fortaleza",
                                     "Distrito Federal (DF) - Brasília", "Espírito Santo (ES) - Vitória",
                                     "Goiás (GO) - Goiânia", "Maranhão (MA) - São Luís",
                                     "Mato Grosso (MT) - Cuiabá", "Mato Grosso do Sul (MS) - Campo Grande",
                                     "Minas Gerais (MG) - Belo Horizonte", "Pará (PA) - Belém",
                                     "Paraíba (PB) - João Pessoa", "Paraná (PR) - Curitiba",
                                     "Pernambuco (PE) - Recife", "Piauí (PI) - Teresina",
                                     "Rio de Janeiro (RJ) - Rio de Janeiro", "Rio Grande do Norte (RN) - Natal",
                                     "Rio Grande do Sul (RS) - Porto Alegre", "Rondônia (RO) - Porto Velho",
                                     "Roraima (RR) - Boa Vista", "Santa Catarina (SC) - Florianópolis",
                                     "São Paulo (SP) - São Paulo", "Sergipe (SE) - Aracaju",
                                     "Tocantins (TO) - Palmas"}

        inteiro contador = 0
        escreva("Os estados do Brasil são: \n\n")
        enquanto(contador<=26)
        {
            escreva(estados[contador], ", ")
            contador = contador+1
            se(contador%3==0)
            {
                escreva("\n")
            }
        }
    }
}
```





**Figura 81 – Código exemplo de uso de “VETORES” combinado com constantes no Portugol Studio.**

**Fonte:** o autor

**Descrição:** Este exemplo apresenta um programa de computador que possui uma função chamada início e nela declara um vetor com do tipo cadeia chamado "estados" 27 posições, com os valores predefinidos dos 27 estados brasileiros no formato "nome(sigla) - capital". Este vetor em sua declaração ficou também com um modificador, "const", posto antes do seu tipo de variável, que o definiu como uma constante. Em seguida é declarada uma variável do tipo inteiro chamada "contador", com o valor pré-definido 0 e, através do comando "escreva", é exibida a mensagem "Os estados do Brasil são:". Logo após é iniciada uma estrutura de repetição do tipo "ENQUANTO FAÇA" com a condição de parada ( $\text{contador} \leq 26$ ) e no corpo é exibido o conteúdo do vetor na posição de índice igual a "contador", depois a variável "contador" recebe ela mesma acrescida de 1, e por fim, é criada uma estrutura condicional "SE" com o teste lógico ( $\text{contador} \% 3 == 0$ ) e no corpo deste é estabelecida uma instrução para pular linha na exibição da saída através do comando "escreva" e do parâmetro "\n".

Os estados do Brasil são:

Acre (AC) - Rio Branco, Alagoas (AL) - Maceió, Amapá (AP) - Macapá,  
Amazonas (AM) - Manaus, Bahia (BA) - Salvador, Ceará (CE) - Fortaleza,  
Distrito Federal (DF) - Brasília, Espírito Santo (ES) - Vitória, Goiás (GO) - Goiânia,  
Maranhão (MA) - São Luís, Mato Grosso (MT) - Cuiabá, Mato Grosso do Sul (MS) - Campo Grande,  
Minas Gerais (MG) - Belo Horizonte, Pará (PA) - Belém, Paraíba (PB) - João Pessoa,  
Paraná (PR) - Curitiba, Pernambuco (PE) - Recife, Piauí (PI) - Teresina,  
Rio de Janeiro (RJ) - Rio de Janeiro, Rio Grande do Norte (RN) - Natal, Rio Grande do Sul (RS) - Porto Alegre,  
Rondônia (RO) - Porto Velho, Roraima (RR) - Boa Vista, Santa Catarina (SC) - Florianópolis,  
São Paulo (SP) - São Paulo, Sergipe (SE) - Aracaju, Tocantins (TO) - Palmas,

**Figura 82 – Saída do exemplo de uso de “VETORES” combinado com constantes no Portugol Studio.**

**Fonte:** o autor

**Descrição:** É exibida a saída que foi calculada no programa de computador do exemplo de uso de “VETORES” combinado com constantes no Portugol Studio. No caso, é exibida a mensagem "Os estados do Brasil são:" e em seguida são exibidos todos os estados armazenados no vetor, 3 a cada linha.



Indicamos que você assista esse vídeo como forma de ampliar os conhecimentos de vetores em algoritmos, parte 1.

<https://www.youtube.com/watch?v=Tp1iHwek9Hg>



Indicamos que você assista esse vídeo como forma de ampliar os conhecimentos de vetores em algoritmos, parte 2.

<https://www.youtube.com/watch?v=he9k99LOD64>





## 4.5.2 Matrizes

Podemos definir uma matriz, também conhecida como array bidimensional ou multidimensional (duas ou mais dimensões), como um vetor de vetores ou uma estrutura que forma uma tabela composta por várias linhas e várias colunas. Também podem existir matrizes com mais de duas dimensões dependendo da necessidade. Com matrizes diversos dados podem ser representados, tais como matrizes matemáticas, tabelas, um tabuleiro de xadrez ou de batalha naval, um espaço ou objeto tridimensional... A Figura 83 apresenta a sintaxe básica para a declaração de uma matriz no Portugol Studio. A Figura 84 explica graficamente como os dados resultantes do algoritmo ficam distribuídos na matriz criada.

```
tipo_do_matriz nome_do_matriz[quantidade_linhas_matriz][quantidade_colunas_matriz]
```

Figura 83– Sintaxe de “MATRIZES” no Portugol Studio.

Fonte: o autor

Descrição: Pseudocódigo genérico de “MATRIZES”.

		Colunas		
		C0	C1	C2
Linhas	L0	João	5.5	REPROVADO
	L1	José	8.9	APROVADO
	L2	Juliana	5.0	REPROVADO
	L3	Janaina	9.5	APROVADO
	L4	Judite	10.0	APROVADO

Figura 84– Exemplo gráfico de estruturas de “MATRIZES”.

Fonte: o autor

**Descrição:** Pseudocódigo genérico da declaração de “MATRIZES”, começa com o tipo de variável da matriz, seguido do nome e a quantidade de linhas delimitada entre colchetes “[ ]” e da quantidade de colunas, também delimitada entre colchetes.



## DICA IMPORTANTE!

Em algumas linguagens as matrizes são consideradas como um array de arrays, ou seja um vetor que contem vetores.



## DICA IMPORTANTE!

Podemos percorrer os elementos de uma matriz usando estruturas de repetição e contadores de índices/posição.

As Figuras 85, 86 e 87 apresentam um exemplo da utilização de matrizes no Portugol Studio, nele uma matriz de tipos cadeia é criada, podemos observar que não existe apenas uma forma de se declarar um vetor.

```
programa
{
    funcao inicio()
    {
        inteiro contadorLinhas, contadorColunas
        cadeia matriz[5][3]
        matriz[0][0] = "João"
        matriz[0][1] = "5.5"
        matriz[0][2] = "Reprovado"
        matriz[1][0] = "José"
        matriz[1][1] = "8.9"
        matriz[1][2] = "Aprovado"
        matriz[2][0] = "Juliana"
        matriz[2][1] = "5.0"
        matriz[2][2] = "Reprovado"
        matriz[3][0] = "Janaina"
        matriz[3][1] = "9.5"
        matriz[3][2] = "Aprovado"
        matriz[4][0] = "Judite"
        matriz[4][1] = "10.0"
        matriz[4][2] = "Aprovado"
        para(contadorLinhas=0; contadorLinhas<=4; contadorLinhas++)
        {
            para(contadorColunas=0; contadorColunas<=2; contadorColunas++)
            {
                escreva(matriz[contadorLinhas][contadorColunas], " ")
            }
            escreva("\n")
        }
    }
}
```

Figura 85 – Código exemplo 1 de uso de “MATRIZES” no Portugol Studio.

Fonte: o autor



**Descrição:** Este exemplo apresenta um programa de computador que possui uma função chamada início e nela declara duas variáveis do tipo inteiro chamadas "contadorLinhas" e "contadorColunas". Em seguida é declarada uma matriz do tipo cadeia com 5 linhas e 3 colunas e são atribuídos valores a cada posição da matriz. matriz[0][0]="João", matriz[0][1]="5.5", matriz[0][2]="Reprovado", matriz[1][0]="José", matriz[1][1]="8.9", matriz[1][2]="Aprovado", matriz[2][0]="Juliana", matriz[2][1]="5.0", matriz[2][2]="Reprovado", matriz[3][0]="Janaina", matriz[3][1]="9.5", matriz[3][2]="Aprovado", matriz[4][0]="Judite", matriz[4][1]="10.0", matriz[4][2]="Aprovado". Depois das declarações é iniciada uma estrutura de repetição "PARA" com os parâmetros (contadorLinhas=0; contadorLinhas<=4; contadorLinhas++). Dentro do corpo desta primeira estrutura é criada outra do mesmo tipo com os parâmetros (contadorColunas=0; contadorColunas<=2; contadorColunas++) e dentro do corpo desta, através do comando "escreva" é apresentado o elemento da matriz com o índice de linha igual a "contadorLinhas" e o índice de colunas igual a "contadorColunas". Após fechar a segunda estrutura de repetição, ainda no corpo da primeira, é dada uma instrução para pular linha através do comando "escreva" e do parâmetro "\n".

```
programa
{
    inclui biblioteca Tipos
    funcao inicio()
    {
        inteiro contadorLinhas, contadorColunas
        cadeia matriz[5][3] = {{"João", "5.5", "Reprovado"},
                                {"José", "8.9", "Aprovado"},
                                {"Juliana", "5.0", "Reprovado"},
                                {"Janaina", "9.5", "Aprovado"},
                                {"Judite", "10.0", "Aprovado"}}
        para(contadorLinhas=0; contadorLinhas<=4; contadorLinhas++)
        {
            para(contadorColunas=0; contadorColunas<=2; contadorColunas++)
            {
                escreva(matriz[contadorLinhas][contadorColunas], " ")
            }
            escreva("\n")
        }
    }
}
```

Figura 86 – Código exemplo 2 de uso de "MATRIZES" no Portugol Studio.

Fonte: o autor

**Descrição:** Este exemplo apresenta um programa de computador que possui uma função chamada início e nela declara duas variáveis do tipo inteiro chamadas "contadorLinhas" e "contadorColunas". Em seguida é declarada uma matriz do tipo cadeia com 5 linhas e 3 colunas e com os valores pré-definidos. linha 1: "João", "5.5", "Reprovado". linha 2: "José", "8.9", "Aprovado". linha 3: "Juliana", "5.0", "Reprovado". linha 4: "Janaina", "9.5", "Aprovado". linha 5: "Judite", "10.0", "Aprovado". Depois das declarações é iniciada uma estrutura de repetição "PARA" com os parâmetros (contadorLinhas=0; contadorLinhas<=4; contadorLinhas++). Dentro do corpo desta primeira estrutura é criada outra do mesmo tipo com os parâmetros (contadorColunas=0; contadorColunas<=2; contadorColunas++) e dentro do corpo desta, através do comando "escreva" é apresentado o elemento da matriz com o índice de linha igual a "contadorLinhas" e o índice de colunas igual a "contadorColunas". Após fechar a segunda estrutura de repetição, ainda no corpo da primeira, é dada uma instrução para pular linha através do comando "escreva" e do parâmetro "\n".



João	5.5	Reprovado
José	8.9	Aprovado
Juliana	5.0	Reprovado
Janaina	9.5	Aprovado
Judite	10.0	Aprovado

Figura 87 – Saída do exemplo de uso de “MATRIZES” no Portugol Studio.

Fonte: o autor

**Descrição:** É exibida a saída que foi calculada no programa de computador dos exemplos de uso de “MATRIZES” no Portugol Studio, sendo esta saída a mesma tanto para o exemplo 1 quanto para o exemplo 2. No caso, através do comando "escreva" e de estruturas de repetição "PARA" aninhadas são exibidas as informações dos alunos armazenadas nas matrizes, sendo um aluno por linha. linha 1: "João", "5.5", "Reprovado". linha 2: "José", "8.9", "Aprovado". linha 3: "Juliana", "5.0", "Reprovado". linha 4: "Janaina", "9.5", "Aprovado". linha 5: "Judite", "10.0", "Aprovado".

É possível combinar vetores e matrizes em algoritmos para trabalhar com informações de diversas maneiras para atender as necessidades do problema, como por exemplo uma tabela de alunos e suas notas. O exemplo apresentado pelas Figuras 88 e 89 demonstra como combinar vetores e matrizes e trabalhar com entrada e saída de dados nestas estruturas.

```
programa
{
    funcao inicio()
    {
        inteiro contadorLinhas, contadorColunas
        cadeia alunos[3]
        real notas[3][3]
        // loop para ler o vetor de nomes e a matriz de notas
        para(contadorLinhas=0; contadorLinhas<=2; contadorLinhas++)
        {
            escreva("Aluno",contadorLinhas+1," : ")
            leia(alunos[contadorLinhas])
            para(contadorColunas=0; contadorColunas<=2; contadorColunas++)
            {
                escreva("Nota",contadorColunas+1," : ")
                leia(notas[contadorLinhas][contadorColunas])
            }
            escreva("\n") // pula linha
        }
        // loops para mostrar a tabela de alunos e suas notas

        para(contadorLinhas=0; contadorLinhas<=2; contadorLinhas++)
        {
            escreva("Aluno",contadorLinhas+1," : ",alunos[contadorLinhas]," => ")
            para(contadorColunas=0; contadorColunas<=2; contadorColunas++)
            {
                escreva("Nota",contadorColunas+1," : ",notas[contadorLinhas][contadorColunas]," ")
            }
            escreva("\n") // pula linha
        }
    }
}
```



Figura 88 – Código exemplo de uso de “MATRIZES” combinada com “VETORES” no Portugol Studio.

Fonte: o autor

**Descrição:** Este exemplo apresenta um programa de computador que possui uma função chamada início e nela declara duas variáveis do tipo inteiro chamadas "contadorLinhas" e "contadorColunas". Logo após é declarado um vetor do tipo cadeia com 3 posições chamado "alunos" e uma matriz do tipo real, chamada "notas", com 3 linhas e 3 colunas. Depois das declarações é iniciada uma estrutura de repetição "PARA" com os parâmetros (contadorLinhas=0; contadorLinhas<=2; contadorLinhas++). Dentro do corpo desta primeira estrutura, através do comando "escreva" é solicitado ao usuário que forneça o nome do aluno e, através do comando "leia" o valor é armazenado em "alunos" na posição de índice igual a "contadorLinhas". Em seguida, ainda dentro do corpo desta primeira estrutura, é criada outra do mesmo tipo com os parâmetros (contadorColunas=0; contadorColunas<=2; contadorColunas++) e dentro do corpo desta, através do comando é solicitado ao usuário que forneça uma nota e, através do comando "leia" o valor é armazenado em "notas" na posição com índice de linhas = a "contadorLinhas" e índice de colunas = "contadorColunas". Após fechar a segunda estrutura de repetição, ainda no corpo da primeira, é dada uma instrução para pular linha através do comando "escreva" e do parâmetro "\n". Finalizando o recebimento de informações, é criado outro par de estruturas de repetição "PARA" aninhadas com os mesmos parâmetros do primeiro par. No corpo da primeira estrutura do par, antes da segunda, é exibido o nome do aluno, que é extraído de "alunos", na posição de índice = "contadorLinhas". Depois da apresentação do nome do aluno é iniciada a segunda estrutura de repetição "PARA" do par, no corpo dela, através do comando "escreva", é exibida a nota do aluno armazenada em "notas" com índice de linha = "contadorLinhas" e índice de coluna = "contadorColunas". Após fechar a segunda estrutura de repetição do segundo par aninhado, ainda no corpo da primeira, é dada uma instrução para pular linha através do comando "escreva" e do parâmetro "\n".

```
Aluno1: José
Nota1: 7.5
Nota2: 6.8
Nota3: 9.5

Aluno2: Maria
Nota1: 9.5
Nota2: 7.8
Nota3: 8.5

Aluno3: João
Nota1: 5.5
Nota2: 9.0
Nota3: 8.7

Aluno1: José => Nota1: 7.5 Nota2: 6.8 Nota3: 9.5
Aluno2: Maria => Nota1: 9.5 Nota2: 7.8 Nota3: 8.5
Aluno3: João => Nota1: 5.5 Nota2: 9.0 Nota3: 8.7
```

Figura 89 – Saída do exemplo de uso de “MATRIZES” combinada com “VETORES” no Portugol Studio.

Fonte: o autor

**Descrição:** É exibida a saída que foi calculada no programa de computador do exemplo de uso de “MATRIZES” combinada com “VETORES” no Portugol Studio. No caso, primeiramente são exibidas as mensagens e valores usados durante a fase de coleta dos dados. Em seguida, através do comando "escreva" e de estruturas de repetição "PARA" aninhadas são exibidas as informações dos alunos armazenadas nas matrizes, sendo um aluno por linha. Na linha 1, Aluno1: "José", Nota1: "7.5", Nota2: "6.8", Nota3: "9.5". Na linha 2, Aluno2: "Maria", Nota1: "9.5", Nota2: "7.8", Nota3: "8.5". Na linha 3, Aluno3: "João", Nota1: "5.5", Nota2: "9.0", Nota3: "8.7".



## 4.5.3 Procedimentos

Um procedimento é um tipo de subalgoritmo, também conhecido como sub-rotina, consiste num conjunto de instruções que realiza uma determinada tarefa específica e não retorna nenhum valor. Os parâmetros são variáveis para se proporcionar uma troca de informações entre o programa principal e o procedimento. Nem sempre o uso de parâmetros é necessário. Dois exemplos de procedimentos são as operações de entrada e saída de dados LEIA() e ESCRIVA() no Portugol Studio. Alguns materiais de estudo e trabalho referenciam procedimentos como funções, abstraindo a diferença. No Portugol Studio funções podem ou não retornar um valor.

## 4.5.4 Funções

Podemos definir uma função, ou um procedimento também, como um trecho reutilizável de código, um tipo de subalgoritmo, como um subprograma que retorna um valor, assim, as funções devem sempre utilizar um comando que forneça um retorno. No Portugol Studio, para retornar qualquer resultado usamos a cláusula “retorne”. A Figura 90 apresenta a sintaxe básica para a declaração e chamada de uma função no Portugol Studio. As Figuras 91 e 92 apresentam um exemplo de uma função simples que retorna uma soma. Logo após a palavra reservada “funcao” informamos o tipo de retorno, depois o nome da função seguido dos parâmetros necessários entre parênteses.

```
funcao tipo_retorno nome_função(tipo_parametro1 parametro1, tipo_parametro2 parametro2...)
{
    bloco de código da função.
}

tipo_dado variavel = nome_função(parametro1, parametro2...)

nome_função(parametro1, parametro2...)
```

Figura 90– Sintaxe básica da declaração de uma “FUNÇÃO” no Portugol Studio.

Fonte: o autor

**Descrição:** Pseudocódigo genérico da uso de “FUNÇÃO” no Portugol Studio, a declaração começa com a cláusula “funcao” seguida do tipo de retorno da função, o nome da função e os parâmetros delimitados entre parênteses “()”, em seguida vem o corpo da função delimitado entre chaves “{}”. Para o uso são demonstrados duas maneiras, a chamada para atribuir valor a uma variável e a execução direta. Na chamada para atribuir valor a uma variável





primeiramente vem o nome da variável seguida do sinal de atribuição "=", o nome da função e os parâmetros necessários delimitados entre parênteses, observando que, caso a função não demande parâmetros, deve-se usar os parênteses vazios. Na execução direta basta colocar o nome da função e os parâmetros da mesma maneira que a forma anterior.

```
programa
{
    funcao inicio()
    {
        real numero1 = 5.5, numero2 = 8.5
        real soma = soma2numeros(numero1, numero2)
        escreva("A soma de ", numero1, " e ", numero2, " é ", soma)
    }

    funcao real soma2numeros(real num1, real num2)
    {
        retorne num1+num2
    }
}
```

Figura 91 – Código exemplo de uso de “FUNÇÕES” no Portugol Studio.

Fonte: o autor

**Descrição:** Este exemplo apresenta um programa de computador que possui uma função chamada início e nela declara três variáveis do tipo real chamadas "numero1", "numero2" e "soma", com os valores pré-determinados, "numero1"="5.5", "numero2"="8.5" e "soma" recebe o valor de uma função chamada "soma2numeros" que recebe como parâmetros "numero1" e "numero2". Em seguida, através do comando "escreva" são exibidos os valores das variáveis na mensagem "A soma de 'numero1' e 'numero2' é 'soma'". Após o fechamento da função início é colocada a declaração da função "soma2numeros" recebendo duas variáveis do tipo real como parâmetros, no corpo da função uma única linha possui o comando "retorne" a soma do primeiro parâmetro com o segundo.

A soma de 5.5 e 8.5 é 14.0

Figura 92 – Saída do exemplo de uso de “FUNÇÕES” no Portugol Studio.

Fonte: o autor

**Descrição:** É exibida a saída que foi calculada no programa de computador do exemplo de uso de “FUNÇÕES” no Portugol Studio. No caso, é exibida a mensagem "A soma de 5.5 e 8.5 é 14.0".



Indicamos que você assista esse vídeo como forma de ampliar o conceito de Funções parte 1.

<https://www.youtube.com/watch?v=1c6VMiWmo1Q>



Indicamos que você assista esse vídeo como forma de ampliar o conceito de Funções parte 2.

[https://www.youtube.com/watch?v=8lvum\\_dHgLO](https://www.youtube.com/watch?v=8lvum_dHgLO)



Indicamos que você assista esse vídeo como forma de ampliar o conceito de Funções parte 3.

<https://www.youtube.com/watch?v=RhsXgFpcNII>



Indicamos que você assista esse vídeo como forma de ampliar o conceito de Funções parte 4.

<https://www.youtube.com/watch?v=TL46G0ajoJg>

As bibliotecas são conjuntos de funções desenvolvidas para um tipo de tarefa específica. Além da ferramenta de programação fornecer algumas bibliotecas básicas também é possível instalar ou criar novas bibliotecas para qualquer programa que o desenvolvedor deseje construir. As Figuras 93 e 94 apresentam um exemplo de função um pouco mais elaborada com o uso de bibliotecas nativas do Portugol Studio.

```
programa
{
    inclui biblioteca Matematica
    funcao inicio()
    {
        real raio, area
        escreva("Forneça o raio do circulo:\n")
        leia(raio)
        area = calculaAreaCirculo(raio)
        escreva("A área de um circulo com raio ",raio," é igual a ",area)
    }

    funcao real calculaAreaCirculo(real raio)
    {
        real area
        area = Matematica.PI * Matematica.potencia(raio, 2)
        retorne area
    }
}
```

Figura 93 – Código exemplo de uso de “FUNÇÕES DE BIBLIOTECA” no Portugol Studio.

Fonte: o autor

**Descrição:** Este exemplo apresenta um programa de computador, nele, antes do começo da função início é incluída a biblioteca "Matemática" do Portugol Studio. primeiramente é utilizado o comando "inclua", depois vem o tipo que se deseja incluir no programa, "biblioteca" seguido do seu nome "Matemática".

Depois de incluída a biblioteca de funções desejada é declarada uma função chamada início e nela declara duas variáveis do tipo real chamadas "raio" e "area", em seguida, através do comando "escreva" é solicitado que o usuário forneça o valor de "raio" e o valor é recebido através do comando "leia". Logo após, a variável "area" recebe o valor retornado pela função "calculaAreaCirculo" com "raio" fornecido como parâmetro para depois, através do comando "escreva", ser exibida a mensagem "A área de um circulo de raio 'raio' é igual a 'area'".

Após o fechamento da função início é colocada a declaração da função "calculaAreaCirculo" recebendo uma variável do



tipo real como parâmetro, no corpo da função é declarada uma variável do tipo real chamada "area" e esta recebe o valor da multiplicação da constante PI da biblioteca "Matematica" do Portugol Studio pelo valor de "raio" elevado ao quadrado e, por fim, utilizando o comando "retorne" retorna o valor de "area".

```
Forneça o raio do círculo:  
3  
A área de um círculo com raio 3.0 é igual a 28.274333882308138
```

Figura 94 – Saída do exemplo de uso de “FUNÇÕES DE BIBLIOTECA” no Portugol Studio.

Fonte: o autor

**Descrição:** É exibida a saída que foi calculada no programa de computador do exemplo de uso de “FUNÇÕES DE BIBLIOTECA” no Portugol Studio. No caso, é exibida a solicitação para o usuário fornecer o valor do raio e após fazê-lo é exibida a mensagem "A área de um círculo com raio 3.0 é igual a 28.274333882308138".

Através do uso de funções e procedimentos é possível modularizar as etapas de um algoritmo, simplificando a execução e o entendimento do fluxo de execução principal. As Figuras 95 e 96 apresentam um exemplo que demonstra a utilização de funções, com recebimento de parâmetros, para simplificar a função de execução principal no programa.

```
programa  
{  
    inclui biblioteca Util  
    funcao inicio()  
    {  
        inteiro contadorLinhas, contadorColunas  
        cadeia notasAlunos[5][3] = {"João", "5.5", "Reprovado"},  
                                    {"José", "8.9", "Aprovado"},  
                                    {"Juliana", "5.0", "Reprovado"},  
                                    {"Janaina", "9.5", "Aprovado"},  
                                    {"Judite", "10.0", "Aprovado"}  
        imprimeMatriz(notasAlunos)  
    }  
  
    funcao imprimeMatriz(cadeia matriz[][])  
    {  
        inteiro numLinhas = Util.numero_linhas(matriz)  
        inteiro numColunas = Util.numero_colunas(matriz)  
        inteiro contadorLinhas, contadorColunas  
        para(contadorLinhas=0; contadorLinhas<=4; contadorLinhas++)  
        {  
            para(contadorColunas=0; contadorColunas<=2; contadorColunas++)  
            {  
                escreva(matriz[contadorLinhas][contadorColunas], " ")  
            }  
            escreva("\n")  
        }  
    }  
}
```

Figura 95 – Código exemplo de uso de “FUNÇÕES” com recebimento de parâmetros no Portugol Studio.

Fonte: o autor



**Descrição:** Este exemplo apresenta um programa de computador, nele, antes do começo da função início é incluída a biblioteca "Util" do Portugol Studio. primeiramente é utilizado o comando "inclua", depois vem o tipo que se deseja incluir no programa, "biblioteca" seguido do seu nome "Util". Depois de incluída a biblioteca de funções desejada é declarada uma função chamada início e nela declara duas variáveis do tipo inteiro chamadas "contadorLinhas" e "contadorColunas", em seguida, declara uma matriz 5x3 do tipo cadeia com valores pré-determinados. linha 1: "João", "5.5", "Reprovado". linha 2: "José", "8.9", "Aprovado". linha 3: "Juliana", "5.0", "Reprovado". linha 4: "Janaina", "9.5", "Aprovado". linha 5: "Judite", "10.0", "Aprovado". E por fim chama diretamente uma função chamada "imprimeMatriz" com "notasAlunos" como parâmetro. Após o fechamento da função início é colocada a declaração da função "imprimeMatriz" recebendo uma matriz do tipo cadeia como parâmetro, no corpo da função são declaradas quatro variáveis do tipo inteiro chamadas "numLinhas", "numColunas", "contadorLinhas" e "contadorColunas", sendo "numLinhas" e "numColunas" com valores pré-determinados. "numLinhas" recebe o retorno da função numero\_linhas da biblioteca "Util", que recebe a matriz recebida como parâmetro na função "imprimeMatriz", desta forma a sintaxe fica Util.numero\_linhas(matriz). "numColunas" recebe o retorno da função numero\_colunas da biblioteca "Util", que recebe a matriz recebida como parâmetro na função "imprimeMatriz", desta forma a sintaxe fica Util.numero\_colunas(matriz). Em seguida é criado um par de estruturas de repetição "PARA" aninhadas, o primeiro com os parâmetros (contadorLinhas=0; contadorLinhas<=4; contadorLinhas++) e o segundo com os parâmetros (contadorColunas=0; contadorColunas<=2; contadorColunas++). No corpo da segunda estrutura, através do comando "escreva", é exibido o conteúdo da matriz passada como parâmetro, com índice de linha = "contadorLinhas" e índice de coluna = "contadorColunas". Após fechar a segunda estrutura de repetição do segundo par aninhado, ainda no corpo da primeira, é dada uma instrução para pular linha através do comando "escreva" e do parâmetro "\n".

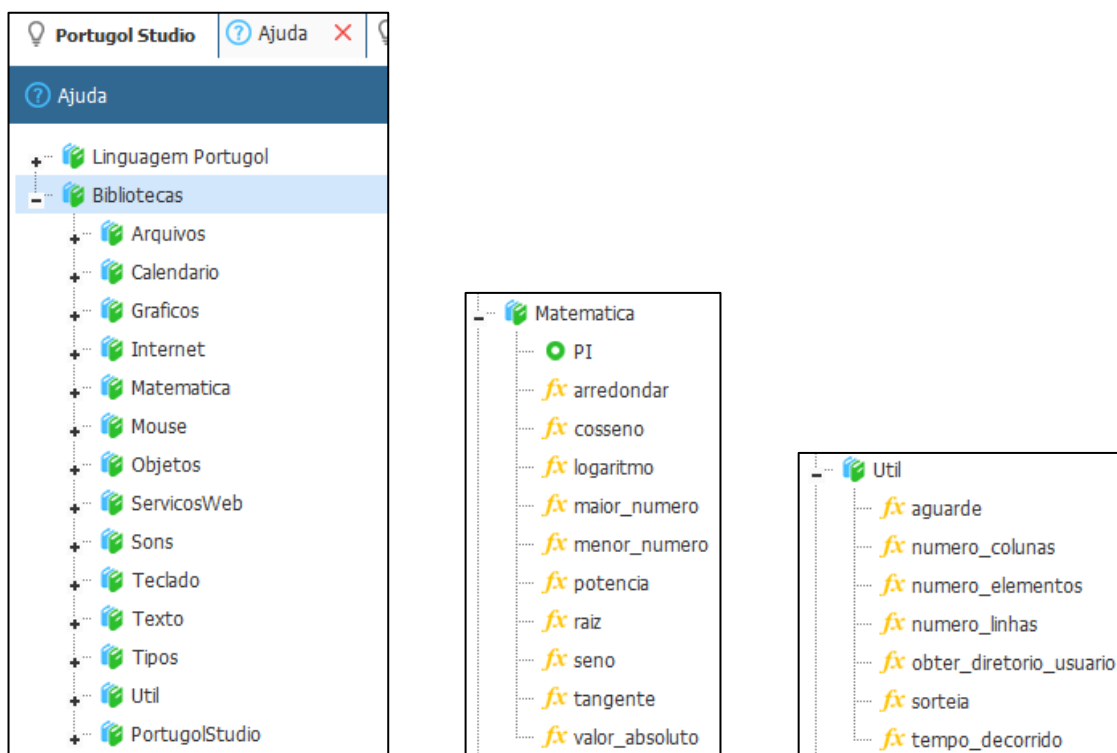
João	5.5	Reprovado
José	8.9	Aprovado
Juliana	5.0	Reprovado
Janaina	9.5	Aprovado
Judite	10.0	Aprovado

**Figura 96 – Saída do exemplo de uso de “FUNÇÕES” com recebimento de parâmetros no Portugol Studio.**

**Fonte:** o autor

**Descrição:** É exibida a saída que foi calculada no programa de computador do exemplo de uso de “FUNÇÕES” com recebimento de parâmetros no Portugol Studio. No caso, através do comando "escreva" e de estruturas de repetição "PARA" aninhadas são exibidas as informações dos alunos armazenadas nas matrizes, sendo um aluno por linha. linha 1: "João", "5.5", "Reprovado". linha 2: "José", "8.9", "Aprovado". linha 3: "Juliana", "5.0", "Reprovado". linha 4: "Janaina", "9.5", "Aprovado". linha 5: "Judite", "10.0", "Aprovado".

A Figura 97 apresenta as bibliotecas nativas na interface do Portugol Studio.



**Figura 97 – Exemplo de bibliotecas de funções disponíveis no Portugol Studio.**

**Fonte:** LITE – Univali em <http://lite.acad.univali.br/portugol/>

**Descrição:** Exemplos de bibliotecas de funções disponíveis no Portugol Studio. Três telas dispostas horizontalmente. À esquerda, todas as bibliotecas nativas disponíveis, no centro a biblioteca “Matematica” e suas funções, à direita a biblioteca “Util” e suas funções.



Indicamos que você assista esse vídeo como forma de ampliar o conceito de Bibliotecas de Funções no Portugol Studio.

<https://www.youtube.com/watch?v=rs8ihN08bgU>

## 4.5.5 Recursividade

Se você acha que isso é tudo que existe em lógica de programação, não se engane, existem diversas tarefas que são possíveis através da combinação das estruturas apresentadas neste material, por exemplo, digamos que precisemos implementar uma função que leia e escreva na tela os elementos de um vetor que pode mudar de tamanho a cada execução, por um motivo qualquer



você precisa adicionar ou remover elementos dele. Como realizar esta tarefa? Uma das soluções possíveis é o uso de recursividade. Recursividade é a implementação de uma função que pode chamar a si mesma. As Figuras 98 e 99 demonstram a solução proposta combinada a conceitos que já aprendemos anteriormente, observe como a função “*leVetorCadeia*” chama ela mesma.

```
programa
{
    inclui biblioteca Util
    funcao inicio()
    {
        cadeia alunos[] = {"João", "Maria", "José", "Pedro", "Teodora"}

        leVetorCadeia(alunos, 0)
    }

    funcao leVetorCadeia(cadeia vetor[], inteiro indice)
    {
        inteiro posicao = indice
        escreva(vetor[posicao])
        se(posicao < (Util.numero_elementos(vetor)-1))
        {
            escreva(", ")
            posicao = posicao + 1
            leVetorCadeia(vetor, posicao)
        }
        senao
        {
            escreva(".")
        }
    }
}
```

Figura 98 – Código exemplo de uso de “RECURSIVIDADE” no Portugol Studio.

Fonte: o autor

**Descrição:** Este exemplo apresenta um programa de computador, nele, antes do começo da função início é incluída a biblioteca "Util" do Portugol Studio. primeiramente é utilizado o comando "inclua", depois vem o tipo que se deseja incluir no programa, "biblioteca" seguido do seu nome "Util". Depois de incluída a biblioteca de funções desejada é declarada uma função chamada início e nela declara um vetor do tipo cadeia chamado "alunos", com os valores pré-determinados "João", "Maria", "José", "Pedro" e "Teodora". Em seguida chama diretamente uma função chamada "leVetorCadeia" com "alunos" e o inteiro 0 como parâmetro. Após o fechamento da função início é colocada a delaração da função "leVetorCadeia" recebendo um vetor do tipo cadeia chamado "vetor" e uma variável do tipo inteiro chamada "indice" como parâmetro, no corpo da função é declarada uma variável do tipo inteiro chamada "posicao", "posicao" recebe o valor do parâmetro do tipo inteiro e, através do comando "escreva" é exibido o conteúdo do vetor passado como parâmetro na posição de índice igual a "posicao". Em seguida é iniciada uma estrutura condicional do tipo "SE SENÃO" com o teste lógico (posicao < (Util.numero\_elementos(vetor)-1), desta forma utilizando uma função da biblioteca "Util" do Portugol Studio para obter o número de elementos do vetor. No corpo da cláusula "SE" o comando escreva é utilizado para imprimir uma virgula, a variável "posicao" recebe o seu valor acrescido de 1 e a própria função "leVetorCadeia" é chamada novamente com "vetor" e "posicao" como parâmetros. No corpo da cláusula "SENAO", através do comando "escreva" é exibido um ponto ".".





Indicamos que você assista esse vídeo como forma de ampliar o conceito de Depuração (Degub).

<https://www.youtube.com/watch?v=M7YxDQn7hRs&list=PLN2AgTfb0UJV69nIF8XMtWIRlpIBxxW0v&index=5ALGORITMO+%7C+MELHORFORMA+DE+RESOLVER+UM+PROBLEMA+%7C+JEAN+VARGAS>

João, Maria, José, Pedro, Teodora.

Figura 99 – Saída do exemplo de uso de “RECURSIVIDADE” no Portugol Studio.

Fonte: o autor

**Descrição:** É exibida a saída que foi calculada no programa de computador do exemplo de uso de “RECURSIVIDADE” no Portugol Studio. No caso, é exibida uma mensagem "João, Maria, José, Pedro, Teodora."

Muito interessante, não é? Porém devemos tomar cuidado com este recurso, é preciso ter um critério de parada bem estabelecido para não implementarmos um LOOP infinito.

## 4.5.6 Depuração (Debug)

Ao longo desta disciplina você conheceu muitos conceitos e, muito provavelmente, enfrentou algumas dificuldades ao tentar implementar alguns algoritmos, algumas vezes ao programarmos cometemos erros, seja de sintaxe seja de lógica, eles acabam atrapalhando o objetivo do programa. Em inglês estes erros são conhecidos pela expressão, “bug”, cuja tradução literal é inseto, nada mais desagradável que alguns insetos 🤔.

Quando construímos um algoritmo e não enxergamos nenhum erro nele antes da execução, ou seja, a sintaxe do código está aparentemente correta, ainda pode existir o risco de haverem erros na lógica do algoritmo. Erros de lógica podem se manifestar tanto impedindo completamente o programa de executar a tarefa desejada, quanto deixando de ser capaz de tratar alguma situação possível não pensada. Para analisar, detectar e corrigir erros em um código existe a atividade de depuração (em inglês também conhecida como Debug). A maioria das ferramentas de programação possui uma função de debug, com recursos como executar as instruções de um algoritmo uma de cada vez, observar os valores das variáveis em tempo de execução.



## Conclusão

Caro aluno chegamos ao fim das 4 competências previstas para esta disciplina, nela tivemos a oportunidade de aprender os princípios básicos de programação de computadores.

Conhecemos uma ferramenta de aprendizado que nos proporciona excelentes noções sobre programação, o Portugol Studio, em seguida começamos a conhecer o assunto.

Começamos com conceitos básicos como Entrada e Saída de dados, Tipos de dados e Variáveis. Conhecemos Formas de Representação e Estruturas de Controle Condicionais e Estruturas de Controle de Repetição. Por fim terminamos com conceitos adicionais importantes como Vetores, Matrizes e Funções. Ainda tivemos espaço para vislumbrar alguns conceitos adicionais de programação como Recursividade e Depuração.

Você acha que acabou? Não acabou! E isso é bom! A área de algoritmos e programação é virtualmente infinito (muitos recursos) e aplicável (útil) a incontáveis segmentos, educação, comercio, ciência e tecnologia, gestão, segurança... Impossível enxergar esta lista inteira, mas dá para imaginar. Você só precisa imaginar uma solução para um problema e programá-la.

As possibilidades dos conhecimentos adquiridos combinados com novos conhecimentos das próximas disciplinas, tais como Programação Orientada a Objeto e Banco de Dados, são promissoras.

Espero que tenha aproveitado bastante o material e a disciplina em si pois são base para as disciplinas Linguagem de Programação para Web, Programação WEB Orientada a Objetos e Projeto de Desenvolvimento de Software. Que tenha apreciado e queria aprender e se aprofundar mais, para que possa se tornar um profissional bem qualificado e com esse diferencial ser bem-sucedido.

Bons Estudos e Sucesso!



## Referências

De Moraes, Paulo Sergio. **Lógica de Programação**. São Paulo: Curso Básico de Lógica de Programação Unicamp - Centro de Computação – DSC, 2000.

Artero, Marcio Aparecido. **Algoritmos e lógica de programação**. Londrina: Editora e Distribuidora Educacional S.A., 2018.

De Souza, Bruno Jefferson; Dias Júnior, José Jorge Lima; Formiga, Andrei de Araújo. **Introdução a Programação**. João Pessoa: Editora da UFPB, 2014.

De Carvalho, Flávia Pereira. **Apostila de Lógica de Programação- ALGORITMOS** -. Taquara: FACCAT-FIT - Faculdade de Informática de Taquara - Curso de Sistemas de Informação., 2007.

Departamento de Computação e Automação. **Algoritmo e Lógica de Programação Algoritmos – Parte 1**. Natal: Universidade Federal do Rio Grande do Norte - Centro de Tecnologia, 2004.

Steinmetz, Ernesto Henrique Radis; Fontes, Roberto Duarte. **CARTILHA LÓGICA DE PROGRAMAÇÃO**. Brasília: Instituto Federal de Brasília, Editora IFB, 2013.

Sebesta, Robert W. **CONCEITOS DE LINGUAGENS DE PROGRAMAÇÃO**. Porto Alegre: Editora BOOKMAN, 2011.

Material de Apoio do Portugal Studio, LITE - laboratório de inovação tecnológica na educação da Universidade do Vale do Itajaí (Univali) em <http://lite.acad.univali.br/portugol/>



## Minicurrículo do Professor

### Leonardo Guimarães de Holanda

Possui graduação em Engenharia de Computação pelo Centro Universitário de Barra Mansa – (UBM), especialização em Segurança da Informação em Engenharia de Software pelo Centro de Estudos de Sistemas Avançados do Recife (C.E.S.A.R), mestrado em Engenharia da Computação pela Universidade de Pernambuco (UPE) e, atualmente possui doutorado em andamento em Ciência da Computação pela Universidade Federal de Pernambuco (UFPE).

Possui 6 anos de experiência na área de Tecnologia da Informação, já atuou profissionalmente exercendo atividades de: analista de sistemas, programador, testador e gerente de configuração. Atualmente é professor de ensino à distância do curso técnico de Desenvolvimento de Sistemas da Escola Técnica Estadual Professor Antônio Carlos Gomes da Costa (ETEPAC)

Possui conhecimento nas tecnologias: Java SE, JSP, JSF, DAO, JavaBeans, IDE Eclipse, Hibernate, JPA, Spring para o controle das transações, Struts para arquitetura em camadas (MVC), Container Apache Tomcat, Padrões de Projeto. Aplicação integrada com banco de dados Oracle, MySQL e PostgreSQL e utilização de PL/SQL. Uso de ferramenta ETL e Shell Script.

Suas áreas de interesse são: engenharia de software, priorização de software, ciclo de vida do software, metodologias ágeis, desenvolvimento de software, desenvolvimento web, desenvolvimento mobile, plataformas de desenvolvimento, internet das coisas, técnicas de inteligência artificial, técnicas de previsão de séries temporais, entre outros.

### Cinthy Cavalcanti Flório

Possui formação superior em Análise e Desenvolvimento de Sistemas pelo Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco (IFPE), curso sequencial de formação



complementar em Análise de Testes de Software pelo Centro de informática da Universidade Federal de Pernambuco (CIn-UFPE), especialização em Segurança da Informação pelo Centro de Estudos e Sistemas Avançados do Recife (C.E.S.A.R), mestrado em Engenharia da Computação pela Universidade de Pernambuco (UPE) e, atualmente possui doutorado em andamento em Ciência da Computação pela Universidade Federal de Pernambuco (UFPE).

Possui 6 anos de experiência na área de Tecnologia da Informação, já atuou profissionalmente exercendo atividades de: analista de sistemas, programadora, testadora e gerente de configuração. Atualmente é professora de ensino à distância do curso técnico de Desenvolvimento de Sistemas da Escola Técnica Estadual Professor Antônio Carlos Gomes da Costa (ETEPAC)

Possui conhecimento nas tecnologias: Java SE, JSP, JSF, DAO, JavaBeans, IDE Eclipse, Hibernate, JPA, Spring para o controle das transações, Struts para arquitetura em camadas (MVC), Container Apache Tomcat, Padrões de Projeto. Aplicação integrada com banco de dados Oracle, MySQL e PostgreSQL.

Suas áreas de interesse são: engenharia de software, priorização de software, ciclo de vida do software, metodologias ágeis, desenvolvimento de software, desenvolvimento web, desenvolvimento mobile, plataformas de desenvolvimento, internet das coisas, técnicas de inteligência artificial, entre outros.

