

**APOSTILA
PORTUGOL STUDIO**

DISTRIBUIÇÃO DOS TEMAS

1.Considerações Iniciais.....	03
2.Conhecendo o Portugol Studio.....	04
2.1.Interface.....	04
2.2.Comentários.....	06
3.Tipos de Dados.....	07
3.1.Inteiro.....	07
3.2.Real.....	07
3.3.Carácter.....	07
3.4.Cadeia.....	07
3.5.Lógico.....	07
3.6.Vazio.....	07
4.Declaração.....	07
4.1.Declaração de Variáveis.....	08
4.2.Declaração de Constantes.....	08
5.Saída de dados.....	09
6.Entrada de dados.....	09
7.Limpar.....	10
8.Operadores.....	10
8.1.Operadores Aritméticos.....	10
8.2.Operadores Relacionais.....	11
8.3.Operadores Lógicos.....	12
8.3.1.e Lógico.....	12
8.3.2.ou Lógico.....	12
8.3.3.nao Lógico.....	13
Exercícios Bloco 01.....	13
9.Estruturas de Seleção.....	15
9.1.Se – Senao.....	15
9.2.Escolha.....	16
Exercícios Bloco 02.....	16
10.Estruturas de Repetição.....	18
10.1.Enquanto.....	18
10.2.Faça – Enquanto.....	19
10.3.Para.....	19
Exercícios Bloco 03.....	20
11.Vetores.....	23
Exercícios Bloco 04.....	24
12.Matrizes.....	25
Exercícios Bloco 05.....	25
13.Sub algoritmos e Bibliotecas.....	26
14.Recursividades.....	28
Exercícios Bloco 06.....	28
REFERÊNCIAS.....	29

CONSIDERAÇÕES INICIAIS

Este material foi elaborado para auxiliar na introdução e desenvolvimento da lógica de programação dos interessados e entusiastas pelo assunto. Parte material apresentado foi extraído da própria ferramenta Portugol Studio(exemplos de códigos), alguns sofrendo pequenas adaptações, bem como o conteúdo contido nesta apostila.

2.CONHECENDO O PORTUGOL STUDIO

O Portugol Studio é um ambiente para aprender a programar, voltado para os iniciantes em programação que falam o idioma português. Possui uma sintaxe fácil, diversos exemplos e materiais de apoio à aprendizagem. Também possibilita a criação de jogos e outras aplicações.

O Portugol Studio possui todos os recursos básicos de uma IDE (Integrated Development Environment): manipulação de arquivos de código-fonte (abrir, salvar, desfazer, etc.), execução e interrupção de programas, console para entrada e saída de dado, console para exibição dos erros de compilação e de execução, Syntax Highlight e Code Completion (em fase experimental).

Pode-se realizar o download do software no site oficial do projeto, em <http://lite.acad.univali.br/portugol/>.

2.1.INTERFACE

A interface inicial do Portugol Studio é simples e intuitiva.

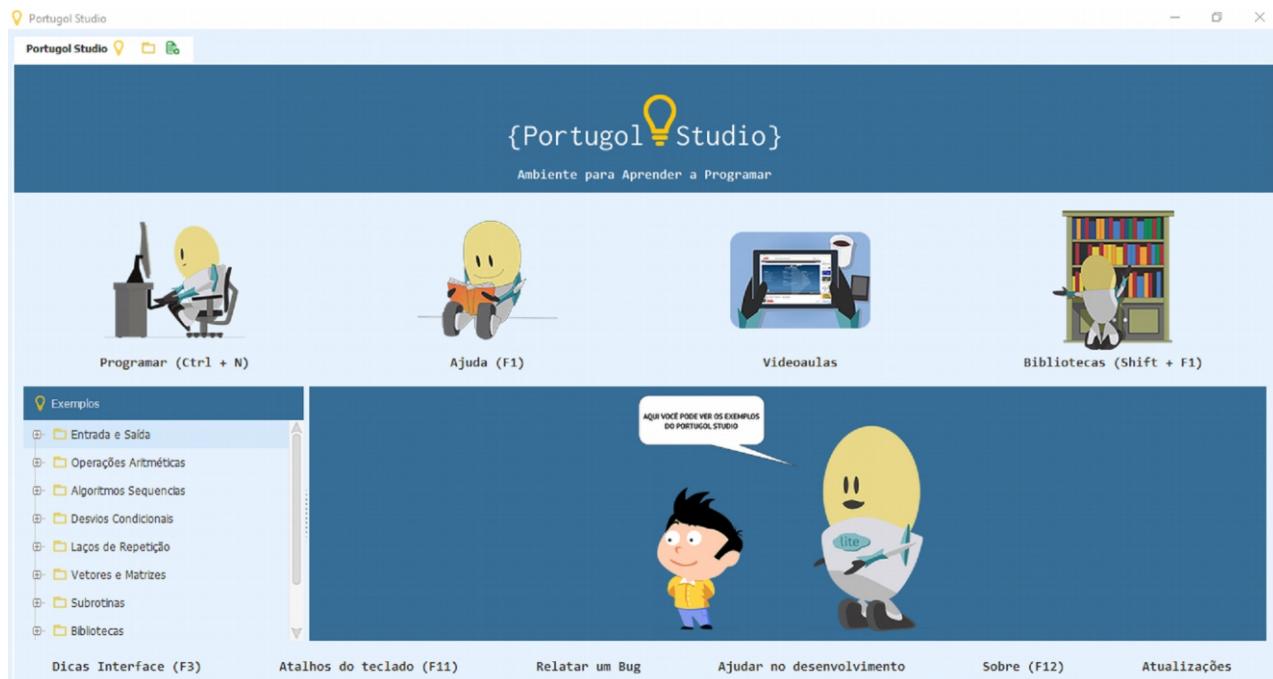


FIGURA 01

Temos então as opções de *Programar*(abre a interface para começar a programar), *Ajuda*(abre uma central de ajuda), *Videoaulas*(direciona para um canal de compartilhamento de vídeos online), *Bibliotecas*(mostra as bibliotecas padrões que podem ser invocadas na elaboração de códigos). Há ainda outras opções que ficam a encardo do leitor verificar suas funcionalidades.

Ao clicar em Programa, uma nova janela irá abrir-se, onde de fato, escreveremos os códigos. No centro temos a estrutura padrão do programa ou código, onde temos o início do **programa** e a chamada para a **funcao inicio()**.

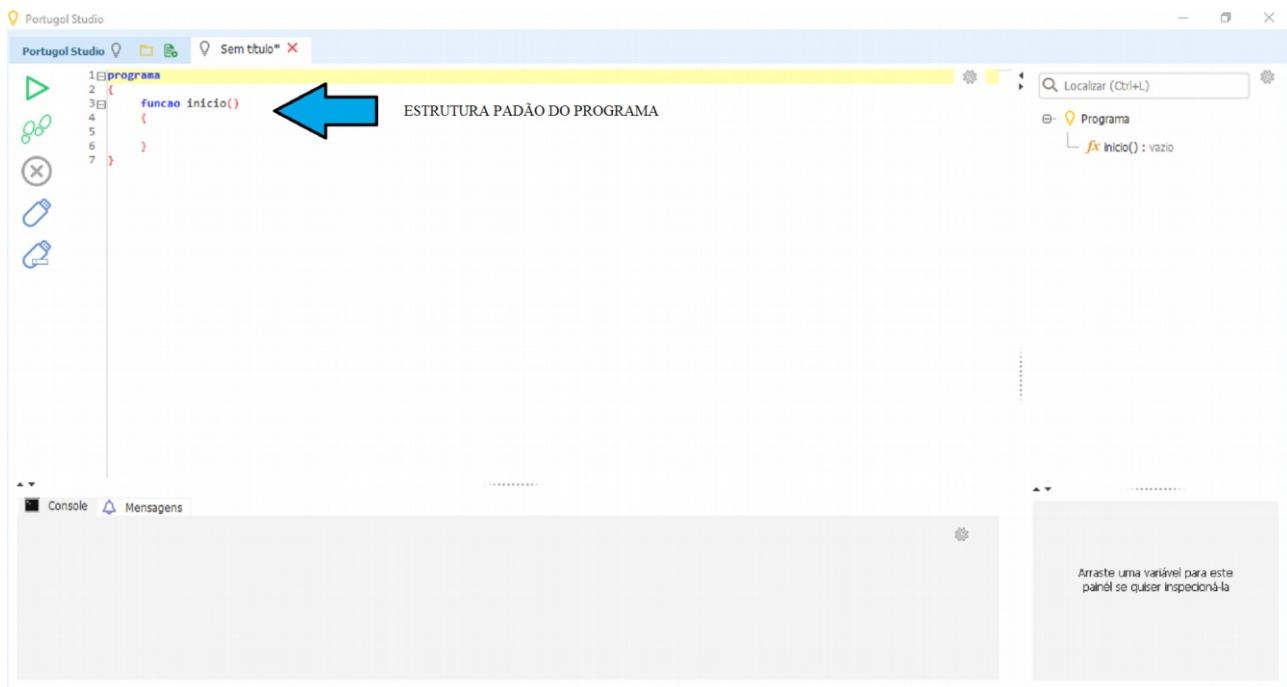


FIGURA 02

programa marca o início da estrutura do algoritmo, sendo delimitado por um abre chaves e fecha chaves.

funcao inicio() delimita a escrita do algoritmo, sendo a função raiz do Portugol Studio, tudo que for escrito dentro desta função estará condicionada a abre chaves e fecha chaves. Observa-se que a função é uma estrutura aninhada a programa, sendo obrigatório o seu uso para o funcionamento dos algoritmos. Outras funções e procedimentos, que veremos mais a frente, podem ser utilizados, mas a **funcao inicio()** sempre deverá estar presente.

Na abra mais à esquerda temos cinco potões, como pode ser visto na FIGURA 03, importante para se conhecer, pois são muito utilizados.

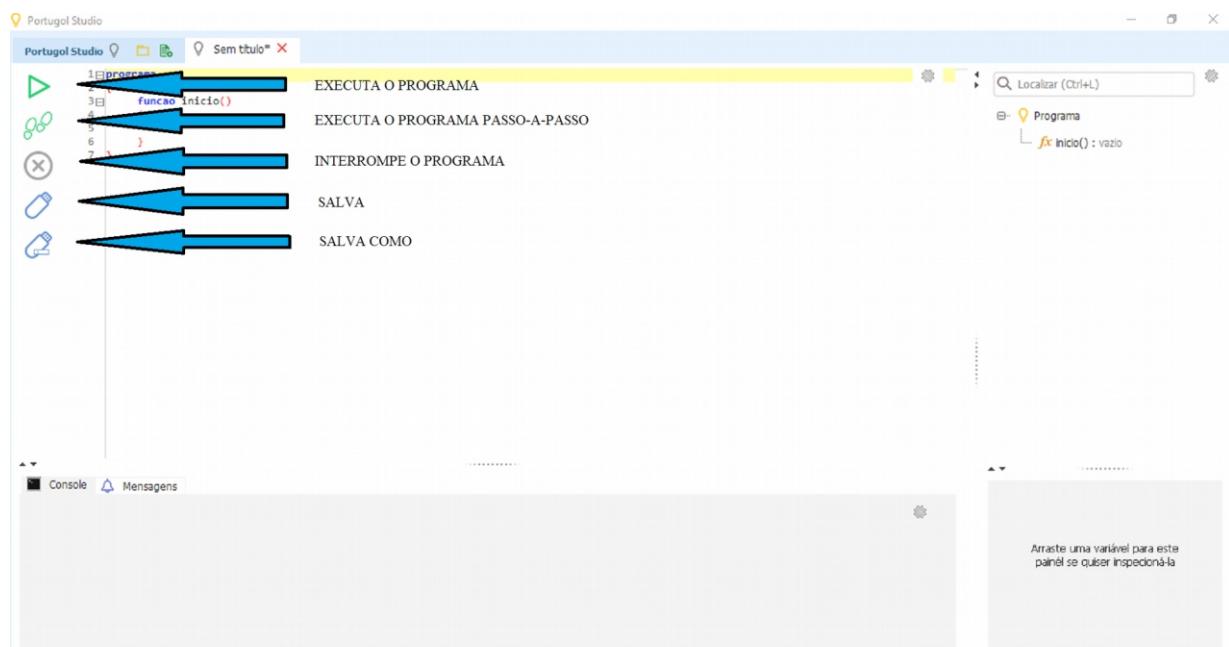


FIGURA 03

De cima para baixo temos:

Executar: Quando clicado inicia o algoritmo.

Passo a passo: Executa a mesma função do “De Bug”, ou seja, executa linha por linha do algoritmo.

Cancela: Ele interrompe a execução do algoritmo.

Salvar: Salva o algoritmo.

Salva como: Salva uma cópia do algoritmo.

Além destes, temos outras funções que serão exploradas ao logo do curso.

2.2.COMENTÁRIOS

Muitas vezes é bastante útil colocar comentários no código, por exemplo para esclarecer o que uma função faz, ou qual a utilidade de um argumento, etc. A maioria das linguagens de programação permite comentários; no Portugol Studio, eles podem aparecer de duas maneiras:

```
* Comentários que podem
ocupar várias
linhas
*/
```

ou

```
//Comentários de uma linha só, que englobam
//tudo desde as duas barras até o final da linha.
```

Tudo que estiver entre as marcas /* e */ ou entre // será ignorado pelo compilador.

```
programa
{
    funcao inicio()
    {
        inteiro a, b, soma // exemplo de comentário em uma linha
        a=2
        b=5
        soma=a+b /* exemplo de comentário em um bloco
        escreva("A soma é: ")
        escreva(soma)
    }

}
```

FIGURA 04

3.TIPOS DE DADOS

Nos algoritmos criados para realizar tarefas na computação utilizamos variáveis para manipular dados, por exemplo: nome, idade, altura, peso, data de nascimento, sexo, saldo, etc. Para otimizar a utilização da memória, cada variável armazena apenas um tipo de dados. A variável *nome*, deve armazenar textos, já a variável *idade* deve armazenar apenas números inteiros(sem casa decimal), na variável *sexo* podemos armazenar apenas um caractere (“M” ou “F”). Seria correto armazenarmos o valor “M” na variável *idade*? Não seria o mais usual, por isso devemos especificar em nossos algoritmos o tipo de cada variável.

3.1 INTEIRO

Dados numéricos positivos ou negativos não-fracionários. Em outras palavras, dados pertencentes ao conjunto dos números inteiros da matemática. Exemplos: 2, 4, -456, 0, 678, -5894.

3.2 REAL

Compreende os números inteiros e também os fracionários. Também pode-se dizer que são dados pertencentes ao conjunto dos números reais da matemática. Exemplos: 4.0, 70.67, -0.25, 8.1, -6.0

3.3 CARÁCTER

Compreende caracteres alfanuméricicos (a-z,A-Z,0-9). Deve estar delimitado por apóstrofes. Exemplos: ‘D’, ‘!’, ‘5’, ‘-‘, ‘:’, ‘y’.

3.4 CADEIA

Representa uma sequência de caracteres que pode ser constituída de letras, números ou símbolos. O tipo cadeia deve estar acompanhado de aspas duplas. Exemplo: “Meu nome é Homer”

3.5 LOGICO

Armazena os valores lógicos True (Verdadeiro) ou False (Falso). Exemplos: Lâmpada acesa ou apagada, computador ligado ou desligado.

3.6 VAZIO

Vazio é usado para o resultado de uma função que retorna normalmente, mas não fornece um valor de resultado a sua chamada. Normalmente, essas funções de tipo vazio são chamados por seus efeitos colaterais, como a realização de alguma tarefa ou escrevendo os seus parâmetros na saída de dados. A função com o tipo vazio termina ou por atingir o final da função ou executando um comando retorno sem valor retornado.

4.DECLARAÇÃO

Segundo ASCENCIO (1999, p.10), quando fazemos um programa, este recebe os dados que devem ser armazenados no computador para que possam ser utilizados no processamento e armazenado na memória do computador.

Um dado é classificado como variável quando tem a possibilidade de ser alterado em algum determinando momento do programa, conforme Forbellone (2000, p.17). Toda variável deverá ser

declarada antes de ser utilizada. Na sua declaração, informaremos um tipo para ela. Um tipo significa “informar” ao computador o que essa variável poderá receber e armazenar na memória do computador.

4.1. DECLARAÇÃO DE VARIÁVEIS

Programas de computador utilizam os recursos de hardware mais básicos para executar algoritmos. Enquanto o processador executa os cálculos, a memória é responsável por armazenar dados e servi-los ao processador. O recurso utilizado nos programas para escrever e ler dados da memória do computador é conhecido como variável, que é simplesmente um espaço na memória o qual reservamos e damos um nome. Por exemplo, podemos criar uma variável chamada “idade” para armazenar a idade de uma pessoa.

Quando criamos uma variável em nosso programa especificamos que tipo de dados pode ser armazenado nela (dependendo da linguagem de programação). Por exemplo, a variável nome só poderia armazenar valores do tipo texto, ou seja, o tipo de dado cadeia. Já a variável idade, só poderia armazenar valores do tipo número (inteiro).

Chamamos este espaço alocado na memória de **variável**, porque o valor armazenado neste espaço de memória pode ser alterado ao longo do tempo, ou seja, o valor ali alocado é “variável” ao longo do tempo.

```

programa
{
    funcao inicio()
    {
        //Variável Local "x" do tipo interira declarada
        inteiro x

        //Variável Local "frase" do tipo cadeira declarada e instanciada
        cadeia frase = "Hello World"

        //Variável Local "neper" do tipo real declarada e instanciada
        real neper = 2.718

        //Variável Local "V" do tipo logico declarada e instanciada
        logico V = verdadeiro

        //Variável Local "Sim" do tipo caracter declarada e instanciada
        caracter Sim = 'S'

        //Imprimi no console o valor armazenado na variável frase
        escreva(frase)
    }
}

```

FIGURA 05

4.2. DECLARAÇÃO DE CONSTANTES

Diferente das variáveis, as **constants** são um espaço reservado na memória para armazenar um valor que não muda com o tempo. Por exemplo, o valor PI (3.14159265359...) que nunca vai mudar.

```

programa
{
    //Valor constante "decaimento" do tipo real que não mudará nunca durante a execussão do programa
    const real taxa_decaimento = 0.5

    funcao inicio()
    {

        //Imprimi no console o valor armazenado na variável decaimento
        escreva(taxa_decaimento)

    }
}

```

FIGURA 06

5. SAÍDA DE DADOS

O **escreva** é empregado quando deseja-se mostrar informações no console, sendo assim um comando de saída de dados. Para utilizar-se o comando **escreva**, basta escrever o comando e entre parênteses colocar a(s) variável(eis) ou texto que se deseja mostrar no console. Textos devem estar entre aspas.

```

programa
{

    funcao inicio()
    {
        inteiro x
        caracter valor

        //Imprimi no console a mensagem solicitando um valor para a variável "x"
        escreva("Digite um valor inteiro para X: ")

        //Imprimi no console a mensagem solicitando um valor para a variável "valor"
        escreva("\nDigite um caracter: ")

    }
}

```

FIGURA 07

6.ENTRADA DE DADOS

O **leia** é empregado quando deseja-se obter informações a partir de um teclado, ou seja, é um comando de entrada de dados. O **leia** comando aguarda um valor a ser digitado e o atribui diretamente a uma variável. Para utilizar-se o comando **leia**, deve-se escrever o comando e entre parênteses colocar a(s) variável (eis) que se deseja(m) receber os valores digitados.

```

programa
{
    funcao inicio()
    {
        inteiro x
        caracter valor

        //Imprimi no console a mensagem solicitando um valor para a variável "x"
        escreva("Digite um valor inteiro para X: ")

        //Aguarda o usuário informar um valor para a variável "x", Logo após atribui este valor para a variável
        leia(x)

        //Imprimi no console a mensagem solicitando um valor para a variável "valor"
        escreva("Digite um caracter: ")

        //Aguarda o usuário informar um caracter para a variável "valor", Logo após atribui este caracter para a variável
        leia(valor)
    }
}

```

FIGURA 08

7.LIMPA

Responsável por limpar o console. Não requer nenhum parâmetro e não tem nenhuma saída.

```

programa
{
    //Valor constante "decaimento" do tipo real que não mudará nunca durante a execussão do programa
    const real taxa_decaimento = 0.5

    funcao inicio()
    {

        //Imprimi no console o valor armazenado na variável decaimento
        escreva(taxa_decaimento)

        //Limpa o que estava escrito no console
        limpa()

    }
}

```

8.OPERADORES

Os operadores são utilizados nas expressões matemáticas, lógicas e relacionais. A maioria dos operadores são iguais nas linguagens de programação. Observe com atenção as figuras nos próximos tópicos. Com certeza os operadores serão utilizados em todos os programas.

8.1.OPERADORES ARITMÉTICOS

Segundo Forbellone (2000, p.20), chamamos de operadores aritméticos o conjunto de símbolos que representa as operações básicas da matemática.

Operador	Descrição
+	Operação de adição
-	Operação de subtração
*	Operação de multiplicação
/	Operação de divisão
%	Operação de módulo (resto divisão)

FIGURA 10

A operação módulo encontra o resto da divisão de um número por outro. Dados dois números a (o dividendo) e b o divisor, a modulo b ($a \% b$) é o resto da divisão de a por b. Por exemplo, $7 \% 3$ seria 1, enquanto $9 \% 3$ seria 0.

```
programa
{
    funcao inicio()
    {
        inteiro x = 5, y = 3, soma, divisao, subtracao, multiplicacao, resto

        //Realiza a soma de x e y
        soma = x + y
        //Realiza a subtração de x e y
        subtracao = x - y
        //Realiza a multiplicação de x e y
        multiplicacao = x * y
        //Realiza a divisão de x e y
        divisao = x / y
        //Calcula o resto da divisão de x e y
        resto = x % y

        //Imprime o valor da soma de x e y
        escreva("O valor da soma é : ",soma)
        //Imprime o valor da subtração de x e y
        escreva("\nO valor da subtração é : ",subtracao)
        //Imprime o valor da multiplicação de x e y
        escreva("\nO valor da multiplicação é : ",multiplicacao)
        //Imprime o valor da divisão de x e y
        escreva("\nO valor da divisão é : ",divisao)
        //Imprime o valor do resto de x e y
        escreva("\nO valor do resto é : ",resto)
    }
}
```

FIGURA 11

8.2.OPERADORES RELACIONAIS

Toda vez que precisamos comparar valores, utilizaremos os operadores relacionais. Os operadores utilizados são < (menor que x), > (maior que x), <= (menor ou igual x), >= (maior ou igual x), == (igual x), != (diferente de x)

```
programa
{
    funcao inicio()
    {
        inteiro x = 5, y = 3
        //1º Caso - Realiza o teste de comparação entre os números
        se(x == y)
        //Escreva e mensagem se os x for igual a y
        escreva("X é igual a Y!")

        //2º Caso - Realiza o teste de comparação entre os números
        se(x <= y)
        //Escreva e mensagem se x for menor ou igual a y
        escreva("\nX é menor que Y!")

        //3º Caso - Realiza o teste de comparação entre os números
        se(x >= y)
        //Escreva e mensagem se x for maior ou igual a y
        escreva("\nX é maior a Y!")

        //4º Caso - Realiza o teste de comparação entre os números
        se(x != y)
        //Escreva e mensagem se x for diferente a y
        escreva("\nX é diferente a Y!")

        //Neste exemplo apenas as mensagens dos caso 3 e 4 serão impressas no console
    }
}
```

FIGURA 12

8.3.OPERADORES LÓGICOS

Segundo Mizrahi (1994, p.53), **operadores lógicos fazem comparações**. A diferença entre comparações lógicas e relacionais está na forma como os operadores avaliam seus operandos, esta avaliação resulta em verdadeiro ou falso. Os conectivos lógicos permitem estabelecer relações entre testes lógicos em uma estrutura de seleção. Com estes conectivos é possível a combinação de várias condições em uma única estrutura de tomada de decisão.

Descrição dos Conectivos Lógicos	
e	Conectivo e lógico
ou	Conectivo ou lógico
não	Conectivo nao lógico

FIGURA 13

8.3.1. e LÓGICO

É a operação lógica onde ambas as sentenças, ou afirmações, devem ser verdadeiras para que o resultado então também seja verdadeiro.

```

programa
{
    funcao inicio()
    {
        inteiro x = 5, y = 3
        //1º Caso - Realiza o teste de comparação entre os números
        se(x == y e x >3)
        //Escreva e mensagem ambas as condições forem verdadeiras
        escreva("Teste 1 positivo")

        //2º Caso - Realiza o teste de comparação entre os números
        se(x < y e x != y)
        //Escreva e mensagem ambas as condições forem verdadeiras
        escreva("\nTeste 2 positivo")

        //3º Caso - Realiza o teste de comparação entre os números
        se(x > y e x != y)
        //Escreva e mensagem ambas as condições forem verdadeiras
        escreva("\nTeste 3 positivo!")

        //4º Caso - Realiza o teste de comparação entre os números
        se(x > y e x < y)
        //Escreva e mensagem ambas as condições forem verdadeiras
        escreva("\nTeste 4 positivo!")

        //Neste exemplo apenas a mensaguem do caso 3 será impressa no console
    }
}

```

FIGURA 14

8.3.2. ou LÓGICO

É a operação lógica onde basta que apenas uma das sentenças, ou afirmações, seja verdadeira para que o resultado então também seja verdadeiro.

```

programa
{
    funcao inicio()
    {
        inteiro x = 5, y = 3
        //1º Caso - Realiza o teste de comparação entre os números
        se(x == y ou x > y)
        //Escreva e mensagem se pelo menos uma condição for verdadeira
        escreva("Teste 1 positivo")

        //2º Caso - Realiza o teste de comparação entre os números
        se(x < y ou x == y)
        //Escreva e mensagem se pelo menos uma condição for verdadeira
        escreva("\nTeste 2 positivo")

        //3º Caso - Realiza o teste de comparação entre os números
        se(x > y ou x != y)
        //Escreva e mensagem se pelo menos uma condição for verdadeira
        escreva("\nTeste 3 positivo!")

        //4º Caso - Realiza o teste de comparação entre os números
        se(x < y ou x != y)
        //Escreva e mensagem se pelo menos uma condição for verdadeira
        escreva("\nTeste 4 positivo!")

        //Neste exemplo as mensagens dos casos 1, 3 e 4 serão impressas no console
    }
}

```

FIGURA 15

8.3.3.nao LÓGICO

É o operador lógico que representa a negação (inverso) da variável atual. Se ela for verdade, torna-se falsa, e vice-versa.

```

programa
{
    funcao inicio()
    {
        inteiro x = 5, y = 3

        //1º Caso - Realiza a soma, o teste de comparação e após faz a negação do valor
        se(nao(x + y > 10))
        //Escreva e mensagem se a condição for verdadeira
        escreva("Teste 1 positivo")

        //2º Caso - Realiza a multiplicação, o teste de comparação e após faz a negação do valor
        se(nao(x * y > 10))
        //Escreva e mensagem se a condição for verdadeira
        escreva("Teste 2 positivo")

        //Neste exemplo apenas mensagem do caso 1 será impressa no console
    }
}

```

FIGURA 16

EXERCÍCIOS BLOCO 01

1. Elabore um algoritmo que leia dois números e mostre a soma deles.
2. Construa um algoritmo que leia 4 notas e mostre a média.
3. Desenvolva um algoritmo que receba dois valores numéricos inteiros, calcule e mostre a soma do quadrado desses dois números.

4. Escreva um algoritmo que leia uma medida em metros e converta para centímetros.
5. Elabore um algoritmo que leia o valor do lado do quadrado e calcule a área. Em seguida, calcule o dobro da área. Mostre a área e o dobro.
6. Faça um algoritmo que leia o valor que um funcionário ganha por hora e o número de horas trabalhadas no mês. Calcule e mostre o total do seu salário no referido mês.
7. Com base na altura de uma pessoa, construa um algoritmo que calcule seu peso ideal, usando a seguinte fórmula: $(72.7 \times \text{altura}) - 58$.
8. Faça um algoritmo para transformar uma distância expressa em milhas para quilômetros. Sabe-se que uma milha corresponde a 0,6214 km.
9. Desenvolva um algoritmo que receba o salário de um funcionário, calcule e mostre seu novo salário com reajuste de 15%.
10. Desenvolva um algoritmo que receba o valor de um depósito em poupança, calcule e mostre o valor após um mês de aplicação na poupança, sabendo que a poupança rende 5% ao mês.
11. Desenvolva um algoritmo que receba um valor numérico inteiro, calcule e mostre qual o quociente e o resto da divisão desse número por 3.
12. Desenvolva um algoritmo que receba uma quantidade de um alimento em quilos, calcule e mostre quantos dias durará esse alimento para uma pessoa que consome 50 gramas desse alimento por dia.
13. A turma C é composta de 60 alunos, e a turma D de 20 alunos. Escreva um algoritmo que leia o percentual de alunos reprovados na turma C, o percentual de aprovados na turma D, calcule e escreva:
 - a) O número de alunos reprovados na turma C.
 - b) O número de alunos reprovados na turma D.
 - c) A percentagem de alunos reprovados em relação ao total de alunos das duas turmas.
14. Um motorista de taxi deseja calcular o rendimento de seu carro na praça. Sabendo-se que o preço do combustível é de R\$ 2,50, escreva um algoritmo para ler: a marcação do odômetro (Km) no início do dia, a marcação (Km) no final do dia, o número de litros de combustível gasto e o valor total (R\$) recebido dos passageiros. Calcular e escrever: a média do consumo em Km/L e o lucro (líquido) do dia.
15. Uma loja vende bicicletas com um acréscimo de 50% sobre o seu preço de custo. Ela paga a cada vendedor dois salários-mínimos mensais, mais uma comissão de 15 % sobre o preço de custo de cada bicicleta vendida, dividida igualmente entre eles. Escreva um algoritmo que leia o número de empregados da loja, o valor do salário-mínimo, o preço de custo de cada bicicleta, o número de bicicletas vendidas, calcule e escreva: O salário final de cada empregado e o lucro (líquido) da loja.
16. Elabore um algoritmo que decomponha o número 1738, informando ao usuário o número de unidades, dezenas, centenas e milhares.
17. Uma escola deseja dividir os alunos de uma série em três turmas. Entretanto, deverá ocorrer um equilíbrio no número de alunos em cada turma. Caso ocorra diferença no número de alunos, esta

deverá ser a mínima. Escreva um algoritmo que leia o número de alunos da série, calcule e mostre o número de alunos em cada turma

9. ESTRUTURAS DE SELEÇÃO

Segundo Mizrahi (1994, p.87), uma das tarefas fundamentais de qualquer programa é decidir o que deve ser executado a seguir. Os comandos de seleção permitem determinar qual é a ação a ser tomada com base no resultado de uma expressão condicional. Podemos, dessa forma, selecionar alternativas, dependendo de critérios que foram estabelecidos no programa.

São três os tipos de seleção num programa:

- . Seleção simples
- . Seleção composta
- . Seleção de múltiplas escolhas.

9.1. SE – SENAO

O **se – senao** são estruturas de desvio condicional. Os desvios condicionais são estruturas onde o fluxo é alterado pelo programa se uma condição for satisfeita, ou seja, dependendo do resultado, é executado um fluxo de comandos. São utilizadas em algoritmos que necessitam tomar uma decisão, baseados em um teste lógico.

Para tanto, a instrução possui uma expressão lógica que será avaliada. Caso a expressão seja verdadeira uma lista específica de comandos será executada. Caso contrário, uma lista alternativa de comandos será executada (se houver). Para definir a lista de comandos que serão executados caso a condição não seja atendida, deve-se usar a palavra reservada **senao**.

É importante afirmar que podemos utilizar algum tipo de seleção ou todos num mesmo programa. Resumindo, a seleção simples serve para uma comparação simples ao passo que a seleção composta são conjuntos de seleções. Podemos inserir condições dentro de condições, isso é muito útil quando precisamos realizar vários testes. Há ainda a estruturas de seleções aninhadas, que basicamente são estruturas de seleções surgem dentro de outras estruturas de seleções.

OBS: Considerações sobre criação de blocos - Blocos de instruções são identificados por um par de abre e fecha chaves. Quando o bloco é constituído por somente uma instrução, as chaves são opcionais.

```

programa
{
    funcao inicio()
    {
        inteiro a, b, c

        escreva ("Informe o 1º lado do triângulo: ")
        leia (a)

        escreva ("Informe o 2º lado do triângulo: ")
        leia (b)

        escreva ("Informe o 3º lado do triângulo: ")
        leia (c)

        se (a == b e a == c)//Uso das chaves opcional, visto que existe apenas uma instrução no bloco
        {
            escreva ("\nEste triângulo é Equilátero\n")
        }
        senao//Estrutura composta
        {
            se (a == b ou b == c ou c == a) //Estrutura aninhada ao senao
                escreva ("\nEste triângulo é Isósceles\n")
            senao
                escreva ("\nEste triângulo é Escaleno\n")
        }
    }
}

```

FIGURA 17

9.2.ESCOLHA

Quando precisamos testar a mesma variável com uma série de valores, podemos utilizar a Seleção de Múltipla Escolha: **Escolha**. A variável do teste deve ser sempre do tipo **inteiro** ou **caracter**. O comando **Escolha** é muito utilizado quando oferecemos várias opções ao usuário, deixando que escolha um valor dentre vários. A vantagem principal desse comando é que ele evita uma série de testes com o comando **SE**. A desvantagem é que os testes somente irão funcionar para variáveis inteiras ou do tipo carácter.

```

programa
{
    funcao inicio()
    {
        inteiro nota
        escreva ("O que você achou do nosso lanche: ")
        escreva("\n1 - Para Ótimo")
        escreva("\n2 - Para Bom")
        escreva("\n3 - Para Médio")
        escreva("\n4 - Para Regular")
        escreva("\n5 - Para Precisa Melhorar\n")
        leia (nota)

        escolha(nota){
            caso 1:
                escreva("O lanche estava ótimo!!!")
                pare
            caso 2:
                escreva("O lanche estava bom!!!")
                pare
            caso 3:
                escreva("O lanche estava comestível!!!")
                pare
            caso 4:
                escreva("O lanche estava regular!!!")
                pare
            caso 5:
                escreva("Precisa melhorar!!!")
                pare
            caso contrario:
                escreva("Nota não avaliada!!!")
        }
    }
}

```

FIGURA 18

EXERCÍCIOS BLOCO 02

1. Elaborar um algoritmo que lê o público total de um jogo de futebol e fornece a renda do jogo, sabendo-se que havia 4 tipos de ingressos assim distribuídos: popular 10% do público a R\$ 5,00, geral 50% do público a R\$ 10,00, arquibancada 30% do público a R\$ 20,00 e cadeiras 10% do público a R\$ 30,00.
2. Escreva um algoritmo que leia a quantidade de combustível abastecido em um automóvel em litros, bem como, a distância que o automóvel percorre por litro abastecido. O algoritmo deverá calcular e mostrar a distância máxima que o automóvel poderá atingir.
3. O valor unitário de um livro na promoção custa R\$ 12,00, caso o cliente comprar até dez livros. Caso contrário, o preço unitário do livro custa R\$ 8,00. Escreva um algoritmo que leia o número de livros comprados, calcule e mostre o valor total que o cliente deverá pagar.

4. Escreva um algoritmo que leia um valor para a variável **A** e um valor para a variável **B**. Logo após, o algoritmo deverá fazer com que a variável **A** guarde o valor da variável **B** e a variável **B** guarde o valor da variável **A**. Apresentar os valores das variáveis **A** e **B** antes e depois da troca.

5. **CALCULADORA:** escreva um algoritmo para ler 2 valores e uma das seguintes operações a serem executadas (codificada da seguinte forma: **(1.Adição, 2.Subtração, 3.Divisão, 4.Multiplicação)**). Calcular e escrever o resultado dessa operação sobre os dois valores lidos.

6. Em um torneio de atletismo as equipes são formadas por quatro jogadores. A equipe da Escola **SóDaCampeão** é formada pelos seguintes atletas: João, Chico, Pedro e Bola. Algumas restrições podem ocasionar a desclassificação da equipe, são elas:

- a) O jogador João não pode ter um número de pontos menor que a metade da soma dos pontos dos demais jogadores que compõem a equipe.
- b) O jogador Pedro não pode ficar com o triplo de pontos do jogador bola.
- c) O jogador chico não pode ficar com zero pontos.
- d) O jogador bola pode ficar com até metade dos pontos do jogador Pedro ou abaixo da soma de pontos dos jogadores João e chico.

Sendo assim, elabore um algoritmo que leia a pontuação de cada um dos jogadores citados. Após, verifique e mostre se a equipe foi ou não desclassificada.

7. Uma lancheria está com alguns problemas para atender aos pedidos dos clientes. Não vai ser possível atender pedidos que tenham algumas combinações do cardápio, são elas:

- a) O lanche bauru não poderá ser acompanhado da bebida guaraná.
- b) Se o cliente pedir X galinha, não poderá pedir água.
- c) Quando o cliente pedir Pizza, somente poderá beber vinho ou água.

Dessa forma, elabore um algoritmo que leia o lanche e a bebida e verifique se o pedido poderá ser ou não atendido.

8. Escreva um algoritmo que receba o nome, idade, sexo e salário fixo de um funcionário. Calcule e mostre o nome e salário líquido do funcionário de acordo com a tabela:

SEXO	IDADE	ABONO
M	≥ 30	100,00
	< 30	50,00
F	≥ 30	200,00
	< 30	80,00

9. Em uma competição o nadador é premiado de acordo com a distância máxima que percorrer. Se o nadador percorrer uma distância menor que 800 metros, recebe R\$ 5.000,00 reais de prêmio. Caso percorra uma distância entre 800 e 1500 metros, recebe R\$ 10.000,00. E se percorrer uma distância superior a 1500 metros, recebe R\$ 15.000,00 em prêmio. Escreva um algoritmo que leia a distância percorrida e mostre na tela o valor da premiação a ser recebida pelo nadador.

10. Escreva um algoritmo que leia a cidadania de uma pessoa, codificada da seguinte forma:

- 1 – Brasileiro,
- 2 – Alemão,
- 3 – Inglês,
- 4 – Italiano,
- 5 – Espanhol e

6 – Francês.

O algoritmo deverá mostrar na tela a língua nativa do cidadão, de acordo com a cidadania selecionada. Caso não seja informado nenhum código válido, informar na tela que a língua nativa da pessoa não pode ser verificada.

11. Elabore um algoritmo que tendo como entrada o preço e código de origem de um produto, calcule e mostre na tela o valor do desconto concedido. Os códigos são os seguintes:

- 1 – região norte com desconto de 5%,
- 2 – região sul com desconto de 15%,
- 3 – região sudeste com desconto de 7%,
- 4 – região nordeste com desconto de 12% e
- 5 – região centro-oeste com desconto de 20%.

Caso não seja informado nenhum código válido informar na tela que o produto é importado.

12. O cardápio de uma lanchonete é o seguinte:

LANCHE	CÓDIGO	VALOR
Cachorro Quente	101	1,20
Bauru Simples	102	1,30
Bauru Com Ovo	103	1,50
Hambúrguer	104	1,20
Cheeseburger	105	1,30
Refrigerante	106	1,00

Escrever um algoritmo que leia o código do item pedido e a quantidade. Calcule o valor a ser pago por aquele lanche. Considere que a cada execução somente será calculado um item. Caso não seja informado algum código da lista, deve-se informar que o código do lanche é inválido.

10. ESTRUTURA DE REPETIÇÃO

Todos os programas apresentados até agora não permitem que sejam executados mais de uma vez. Se deseja-se repetir algum programa necessita-se sair do programa e executá-lo novamente. Isso é inviável. Precisa-se permitir repetições. Outra necessidade: imagine-se no caos de precisar calcular a média das notas de 500 alunos sabendo-se que cada aluno tem 4 notas. Se fosse criar todas essas variáveis perderia-se um tempo enorme só na declaração dessas variáveis.

Segundo Forbellone (2000, p.49), uma estrutura de repetição é uma estrutura com controle de fluxo de execução, que permite repetir diversas vezes um mesmo trecho do programa. Do mesmo jeito que na estrutura de decisão, a estrutura de repetição depende do teste de uma condição.

São três os tipos de estruturas de repetição:

- . Repetição com teste no início: **Enquanto**
- . Repetição com teste no fim: **Faça ... Enquanto**
- . Repetição com variável de controle **Para**

10.1. Enquanto

A estrutura de repetição **enquanto** é utilizada para repetir um bloco de comandos por várias vezes, sem sabermos ao certo a quantidade exata de vezes. Para isso criamos uma condição de teste,

que é testada já no início do bloco. O uso do comando **enquanto** é ideal, quando precisamos sair do comando de uma forma repentina ou assim que atingir algum objetivo. No comando **enquanto** primeiro a condição é avaliada e enquanto a condição for verdadeira, o comando **enquanto** é executado. No momento em que a condição deixar de ser verdadeira, o comando para.

```

programa
{
    funcao inicio()
    {
        inteiro cont = 1
        real num, media, soma = 0.0

        enquanto(cont <= 10)// Condição do Laço que verifica se já foram informados 10 valores
        {//Abre o bloco
            escreva("Digite o ", cont, "º número: ")
            leia(num)

            soma = soma + num // A variável soma é o acumulador, ou seja, acumula o valor somado
            cont = cont + 1 // Incrementa o contador para controlar o laço de repetição
        }//Fecha o bloco
        media = soma / 10
        escreva("A média dos números é: ", media, "\n")
    }
}

```

FIGURA 19

10.2.Faça – Enquanto

O comando **faça – enquanto** realiza o teste da condição no final do comando. Isso quer dizer que ao menos uma vez o comando será executado. Esse comando é o oposto do comando **enquanto**, visto que, o **enquanto** primeiro testa para entrar no bloco e o **faça – enquanto** primeiro executa para depois testar a condição.

```

programa
{
    funcao inicio()
    {
        inteiro id
        faca
        {//Abre o bloco
            escreva ("Informe a sua idade: ")
            leia (id)
        }//Fecha o bloco
        enquanto (id < 5 ou id > 150)//Testa a condição, se for verdadeira,
                                         //continua a executar o bloco,
                                         //caso contrário encerra o bloco.
            escreva ("\nSua idade é: ",id," anos\n")
    }
}

```

FIGURA 20

10.3.Para

A estrutura de repetição **para**, deve ser usada quando sabemos o **número exato** de repetições. Na estrutura de repetição **para** devemos usar uma variável de controle. Essa variável deve ser sempre do tipo inteiro ou carácter. Nesta estrutura de repetição informa-se uma variável de controle **x** onde a mesma deverá ser testada conforme uma condição determinada **y**, caso a condição seja verdadeira, a parte interna do código dentro da estrutura será executado. No final a variável

declarada em **x** deverá ser incrementada ou decrementada para então a condição em **y** ser novamente testada, assim sucessivamente até que ela seja falsa.

```

programa
{
    funcao inicio()
    {
        inteiro soma = 0, num, cont

        escreva("Digite o número até o qual deseja somar: ")
        leia(num)

        // Repete até o contador atingir o valor informado pelo usuário

        para (cont = 0; cont <= num; cont++)//Laço que controla o número de repetições conforme o
        {
            soma = soma + cont // Soma o valor atual do contador
        }
        escreva("A soma de 1 até ", num, " é: ", soma, "\n")
    }
}

```

FIGURA 21

EXERCÍCIOS BLOCO 03

1. Escreva um algoritmo que leia dez valores numéricos inteiros e apresente na tela o somatório dos valores.
2. Elabore um algoritmo que leia um determinado número e apresente na tela a tabuada de multiplicação deste número. Por exemplo, digamos que o número informado foi 2, o programa deverá apresentar na tela:

1 x 2 = 2
 2 x 2 = 4
 3 x 2 = 6
 ...
 10 x 2 = 20

3. Crie um algoritmo que leia um determinado número e apresente na tela a tabuada da divisão deste número. Por exemplo, digamos que o número informado foi 5, o programa deverá apresentar na tela:

5 : 5 = 1
 10 : 5 = 2
 15 : 5 = 3
 ...
 50 : 5 = 10

4. Escrever um algoritmo que lê um valor numérico inteiro. Após, escreva uma tabela com cabeçalho, contendo o valor, seu quadrado e seu cubo. Mostrar para todos valores entre o valor informado e 1. Por exemplo, digamos que o número informado seja 20, deve-se apresentar na tela:

Número	Dobro	Triple
20	400	8000
19	361	6859
18	324	5832

...

1 1 1

5. Chico tem 1,50 metro e cresce 2 centímetros por ano, enquanto Zé tem 1,10 metro e cresce 3 centímetros por ano. Construa um algoritmo que calcule e imprima quantos anos serão necessários para que Zé seja maior que Chico.

6. Construa um algoritmo que calcule a média aritmética das 3 notas dos alunos de uma classe. O algoritmo deverá ler, além das notas, o código (de três dígitos) do aluno e deverá ser encerrado quando o código for igual a zero.

7. Elabore um algoritmo que leia a medida em metros de frente e fundo de um número indeterminado de terrenos. O algoritmo deverá calcular e mostrar a área do terreno. Deverá parar somente quando a área de um terreno for inferior a 100 metros quadrados.

8. Escreva um algoritmo que leia 50 valores e encontre o maior e o menor deles. Mostre o resultado.

9. Faça um algoritmo que receba a idade, altura e peso de 10 pessoas. Calcule e mostre:

- a) a quantidade de pessoas com idade superior a 50 anos;
- b) a média das alturas das pessoas com idade entre 10 e 20 anos;
- c) a porcentagem de pessoas com peso inferior a 40 quilos entre todas as pessoas.

10. Crie um algoritmo que ajude o DETRAN a calcular o total de recursos que foram arrecadados com a aplicação de multas de trânsito. O algoritmo deve ler as seguintes informações para 20 motoristas:

- número da carteira de motorista (número inteiro de 4 dígitos);
- número de multas do motorista;
- valor da multa (considerando que todas as multas de um motorista tem o mesmo valor).

Ao final da leitura, deve-se apresentar o total de recursos arrecadados (somatório das multas de todos motoristas). O algoritmo deverá mostrar também o número da carteira do motorista que obteve o maior número de multas.

11. Elaborar um algoritmo que leia os limites inferior e superior de um intervalo e mostre todos os números pares no intervalo, bem como, o somatório dos pares. Suponha que os números digitados são um intervalo crescente. Exemplo:

Límite inferior: 3

Límite superior: 12

Pares: 4 6 8 10

Soma dos pares: 28

12. Faça um algoritmo que leia 20 números inteiros e positivos e calcule o produto dos números pares e o somatório dos ímpares. Além disso, deve-se verificar a quantidade de pares e ímpares informados. O algoritmo não poderá aceitar valores negativos, nem zero. O fim da leitura será indicado pelo número 0.

13. Foi realizada uma pesquisa entre os habitantes de uma região. Sendo assim, foram coletados os dados de idade, sexo (M/F) e salário. Encerre a entrada de dados quando for digitada uma idade zero. Faça um algoritmo que informe:

- a) a média de salário dos homens;
- b) a média de salário das mulheres;
- c) a média de salário do grupo;

- d) o homem mais velho;
 e) a mulher mais nova;

14. Em uma eleição presidencial existem quatro candidatos. Os votos são informados por meio de um código (representando os candidatos). Os códigos utilizados são:

- 1- João
 2- Maria
 3- Pedro
 4- Marcos
 5- Nulo
 6- Em branco

Escreva um algoritmo que calcule e mostre:

- a) o total de votos para cada candidato;
 b) o total de votos nulos;
 c) o total de votos em branco;
 d) o percentual de nulos em relação ao total de votos;
 e) o percentual de brancos em relação ao total de votos.

15. Elabore um algoritmo que, utilizando estruturas de repetição aninhadas, apresente na tela as tabuadas de multiplicação e divisão dos números de 1 a 9.

16. Construa um algoritmo que mostre na tela:

*
 **

17. Construa um algoritmo que mostre na tela:

 **
 *

18. Construa um algoritmo que mostre quatro zeros na tela, veja a saída abaixo:

***** * ***** * ***** *
 * * * * * * * * * *
 * * * * * * * * * *
 ***** * ***** * *****

19. Construa um algoritmo que mostre a palavra IOIO na tela, veja a saída abaixo:

**** * **** * **** *
 ** * * * * * * *
 ** * * * * * * *
 **** * **** * ****

20. Faça um algoritmo que leia um inteiro n e imprima um tabuleiro de xadrez ou damas com lado n. Represente casas pretas por um "X" e casas brancas por um "O". Ex: n = 4

O X O X
 X O X O
 O X O X
 X O X O

21. Faça um algoritmo que leia um inteiro n e imprima um triângulo com altura n. Ex: n = 5

```
*  
***  
*****  
*****  
*****
```

11. VETORES

Chamamos de vetores os conjuntos de variáveis agrupadas do mesmo tipo. Um exemplo prático da utilização de vetores. Imagine-se que precisasse armazenar as notas de 50 alunos. Na forma como vínhamos declarando as variáveis, teríamos que declarar 50 variáveis, ficando assim: nota1, nota2, nota3, nota4,.....até nota50. Na forma de vetores, a declaração fica da seguinte maneira:

`real nota [50].`

Isso é equivalente a criar 50 variáveis do tipo real.

Agora para criar uma variável do tipo vetor com 10 posições, mas cada elemento do vetor deverá ser do tipo inteiro. A declaração da variável ficaria assim:

`inteiro numero [10]`

Cria-se assim um vetor com dez elementos inteiros. Para acessar uma variável, pode-se referir diretamente a ela: numero [1], numero [2], assim, se quiser atribuir um valor de 3000 ao elemento 8, do vetor número, deve-se no referir à variável da seguinte forma:

`numero [8] = 3000.`

No caso do numero [8], está se referindo à variável de posição 8. Observe a figura 22. Os quadrados representam cada elemento da variável numero [10]. O número que está dentro dos colchetes representa o índice do vetor ou a posição dele.

`vetor:numero[10]`



FIGURA 22

Trabalhar com vetor, significa trabalhar com estruturas de repetições. O motivo é simples: trabalharemos com muitas variáveis e deixaremos o trabalho mais pesado para a estrutura de repetição. A melhor estrutura de repetição para trabalhar com vetores é o **para**, pelo simples motivo de sempre ter-se o número exato dos elementos do vetor.

Agora um detalhe muito importante: no exemplo acima o índice do vetor começa do 1. No Portugol Studio e em outras muitas linguagens de programação, como o C por exemplo, o índice começa do 0. Veja a figura 23. Continua-se com dez elementos, porém o valor do índice começa do 0 (zero).

`vetor:numero[10]`

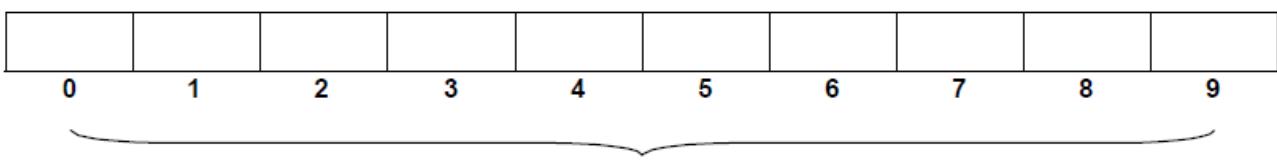


FIGURA 23

```

programa
{
    inclua biblioteca Util --> util

    funcao inicio()
    {
        inteiro vetor[10]

        // preenche o vetor
        para (inteiro posicao = 0; posicao < 10; posicao++)
        {
            vetor[posicao] = util.sorteia(1, 100) // Sorteia um número e atribui à posição do vetor
        }

        // Exibe o vetor na ordem original
        escreva ("Vetor na ordem original:\n")

        para(inteiro posicao = 0; posicao < 10; posicao++)
        {
            escreva (vetor[posicao], " ")
        }

        // Exibe o vetor na ordem inversa
        escreva ("\n\nVetor na ordem inversa:\n")

        para(inteiro posicao = 9; posicao >=0; posicao--)
        {
            escreva (vetor[posicao], " ")
        }
    }
}

```

FIGURA 24

BLOCO DE EXERCÍCIOS 04

1. Elabore um algoritmo que leia uma lista de 10 valores numéricos inteiros quaisquer. Feito isso, escreva na tela o maior e menor valor informado.
2. Construa um algoritmo que leia uma lista correspondente aos preços unitários dos produtos de uma loja de material esportivo. Ao final, o algoritmo deverá mostrar o valor total em estoque da loja.
3. Sabendo-se que a média de aprovação em uma disciplina é 6, escreva um algoritmo que leia uma lista de 10 médias de alunos. Logo a seguir, apresente na tela a quantidade de alunos que obtiveram aprovação e a quantidade de alunos que ficaram em recuperação na disciplina.
4. Escreva um algoritmo que leia uma lista de salários dos funcionários de uma empresa. Após, o algoritmo deverá aplicar um aumento de 10% somente sobre salários abaixo de R\$ 1.500,00. Mostrar na tela a lista dos salários.
5. Elabore um algoritmo que leia o nome, a quantidade e o valor de uma lista de 10 produtos. Ao final deverá calcular o subtotal de cada produto.
6. Faça uma modificação no exercício anterior para mostrar no final o total de todos os produtos.
7. Elabore um algoritmo que leia duas listas de 10 elementos:
 - a) a primeira lista contendo os nomes dos times.
 - b) a segunda lista contendo a pontuação dos times.

Ao final, apresentar na tela o time campeão (com pontuação) e o último colocado (com pontuação) no campeonato.

12. MATRIZES

As matrizes são estruturas de dados simples similar aos vetores, porém possui uma dimensão a mais, o que permite organizar os valores em linhas e colunas. Para acessar um valor individual é necessário dois índices, um para linha e outro para coluna. Durante a inicialização deve-se respeitar a quantidade de linhas e colunas informados.

Sua declaração é bastante similar aos vetores, onde deve-se informar o tipo, seu nome, e a quantidade de linhas e colunas.

```

programa
{
    inclua biblioteca Util --> u

    funcao inicio()
    {
        const inteiro TAMANHO = 5 // Define as dimensões (linhas e colunas) da matriz
        inteiro matriz[TAMANHO][TAMANHO] // Cria a matriz

        para (inteiro linha = 0; linha < TAMANHO; linha++)
        {
            para (inteiro coluna = 0; coluna < TAMANHO; coluna++)
            {
                matriz[linha][coluna] = u.sorteia(1, 9) // Atribui um valor aleatório à posição da matriz

                escreva("[", matriz[linha][coluna], "]") // Exibe o valor contido na posição da matriz

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

```

FIGURA 25

BLOCO DE EXERCÍCIOS 05

1. Escreva um algoritmo que leia os elementos de uma matriz 5x5 de inteiros. Ao final, o algoritmo deverá mostrar a soma de todos os elementos.
2. Elabore um algoritmo que leia uma matriz 4x4 de reais. Ao final, apresentar o resultado da subtração da soma dos elementos da primeira linha pela soma dos elementos da terceira coluna.
3. Escreva um algoritmo que leia um vetor de seis elementos numéricos inteiros. O algoritmo deverá calcular e mostrar:
 - a) a quantidade de números pares;
 - b) a quantidade de números ímpares.
4. Elabore um algoritmo que leia um vetor de 15 elementos inteiros. O algoritmo deverá verificar quantos valores são maiores que 10 e mostrar na tela.
5. Escreva um algoritmo que leia um vetor de 15 elementos do tipo inteiro. Após, calcular e mostrar:
 - a) o maior elemento do vetor e qual a posição que ele ocupa no vetor.
 - b) o menor elemento do vetor e qual a posição que ele ocupa no vetor.

6. Escreva um algoritmo que leia dois vetores de 10 elementos. O primeiro vetor será utilizado para ler o nome dos alunos de uma turma. O segundo vetor para ler as médias dos alunos da turma. Os índices dos dois vetores são correspondentes, ou seja, o aluno da posição 3 do vetor de nomes corresponde ao valor da posição 3 do vetor de médias. Ao final, o algoritmo deverá mostrar os nomes e médias dos alunos com média menor que 6.

7. Uma escola deseja saber, em uma turma de 10 alunos, quantos estão cursando, simultaneamente, as disciplinas de Lógica de Programação e Modelagem de Sistemas. O algoritmo deverá ler as matrículas dos alunos que estão cursando Lógica em um vetor de inteiros. Em outro vetor de inteiros ler as matrículas dos alunos que estão cursando Modelagem. Após, o algoritmo deverá calcular e mostrar a quantidade de alunos que estão cursando as duas disciplinas.

8. Faça um algoritmo para ler uma matriz 3X3 real e imprimir a soma dos elementos da Diagonal principal. Generaliza para uma matriz NXN.

9. Faça um algoritmo para ler uma matriz 2X3 real e depois gerar e imprimir sua transposta (matriz 3X2 equivalente).

10. Ler uma matriz 5X5 e gerar outra em que cada elemento é o cubo do elemento respectivo na matriz original. Imprima depois esta nova matriz.

11. Faça um algoritmo para ler uma matriz de 3×4 de números reais e depois exibir o elemento do canto superior esquerdo e do canto inferior.

12. Ler uma matriz 4X3 real e imprimir a soma dos elementos de uma linha L fornecida pelo usuário.

13. Ler uma matriz 4X3 real. Depois, mostre qual é o elemento armazenado em uma linha e coluna C fornecidos pelo usuário.

14. Crie uma matriz 7X8 onde cada elemento é a soma dos índices de sua posição dentro da matriz.

13.SUB ALGORITMOS e BIBLIOTECAS

Sub algoritmos são blocos, ou pequenas porções, de códigos os quais são chamados por um algoritmo.

Princípio da divisão e conquista.

Dividimos um algoritmo em pequenos módulos visando diminuir a complexidade do código. Reduzimos ao máximo o tamanho do algoritmo principal.

Construímos módulos que tratam aspectos específicos na solução de um problema.

Vantagens dos sub algoritmos

Reduzem o tamanho do algoritmo com a divisão em módulos.

Organizam e facilitam a compreensão do código.

Evitam a reescrita de código ao longo do algoritmo.

Permitem a reutilização de código.

Podem ser reutilizados através de bibliotecas.

Facilitam a manutenção do código.

O **algoritmo** executa as instruções sequencialmente. A cada ponto de chamada o fluxo principal é interrompido e o código do **sub algoritmo** é executado. Ao final da execução do sub algoritmo, o fluxo de execução principal continua a partir do ponto de chamada.

Tipos de sub algoritmos

Os sub algoritmos podem ser classificados de acordo com o retorno ao ponto de chamada.

Dessa forma, são classificados em:

funções;

procedimentos.

Funções

As funções são **sub algoritmos** que executam um conjunto de instruções e retornam um valor resultante ao algoritmo que realizou a chamada.

Funções podem ser embutidas em **bibliotecas**. As bibliotecas de funções podem ser utilizadas na construção de algoritmos. **Portugol Studio** contém uma série de bibliotecas de funções embutidas no ambiente de programação que podem ser utilizadas para construção de algoritmos.

Procedimentos

Já procedimentos são **sub algoritmos** que executam um conjunto de instruções e não retornam valor. Assim como funções, os procedimentos podem ser embutidos em **bibliotecas**.

Portugol Studio contém uma série de bibliotecas de procedimentos embutidos.

Ex:

Um exemplo típico de procedimento em Portugol é o **limpa()**. Procedimento utilizado para limpar a tela.

Importação de bibliotecas

As bibliotecas devem ser importadas por algoritmos para que suas funções possam ser utilizadas. A importação é realizada pela instrução **inclusa biblioteca** da linguagem Portugol. A instrução de importação deve ser inserida logo após o início do programa, antes da função início.

```

programa
{
    funcao inicio()
    {
        mensagem("Bem Vindo") // Chama o procedimento
        // Chama a função no escreva
        escreva("O resultado do primeiro cálculo é: ", calcula (3.0, 4.0))
        // Chama a função no escreva
        escreva("\nO resultado do segundo cálculo é: ", calcula (7.0, 2.0), "\n")

        mensagem("Tchau") // Chama o procedimento
    }
    funcao mensagem (cadeia texto)
    {
        inteiro i
        // Insere uma linha antes do texto da mensagem
        para(i = 0; i < 50; i++)
        {
            escreva ("--")
        }
        escreva ("\n", texto, "\n") // escreve a mensagem
        // Insere uma linha após do texto da mensagem
        para(i = 0; i < 50; i++)
        {
            escreva ("--")
        }
        escreva("\n")
    }
    // Função que realiza um cálculo e retorna o resultado
    funcao real calcula (real a, real b)
    {
        real resultado
        resultado = a * a + b * b
        retorno resultado
    }
}

```

FIGURA 25

14.RECURSIVIDADE

Em ciência da computação, a **recursividade** é a definição de uma sub-rotina (função ou método) que pode invocar a si mesma. Um exemplo de aplicação da recursividade pode ser encontrado nos analisadores sintáticos recursivos para linguagens de programação. A grande vantagem da recursão está na possibilidade de usar um programa de computador finito para definir, analisar ou produzir um estoque potencialmente infinito de sentenças, *designs* ou outros dados.

```

programa
{
    funcao inicio()
    {
        inteiro x
        escreva("\nInforme o número o qual deseja calcular o fatorial: ")
        leia(x)
        fatorial(x) //Faz a chamada da função fatorial passando x como parâmetro
        escreva("o valor do fatorial de ",x," é: ",fatorial(x))//Recebe o resultado da função fatorial

    }
    funcao inteiro fatorial (inteiro x){//Abre o bloco da função fatorial recebendo x como parâmetro
        inteiro resultado
        se(x==0 ou x==1) //Faz a comparação pela definição de fatorial
            retorno 1 //Retorno se a condição for verdadeira
        senao{
            resultado = x * fatorial(x-1)//Chamada recursiva da função fatorial
            retorno resultado//Retorna o resultado para a função inicio
        }
    }
}

```

FIGURA 27

BLOCO DE EXERCÍCIOS 06

1. Escreva um algoritmo para calcular e mostrar a área de uma circunferência. Para isso, o algoritmo deverá ler o valor do raio (r).

$$a = \pi \cdot r^2$$

2. Elabore um algoritmo para calcular e mostrar o volume de uma lata. Dessa forma, deve-se ler o valor do raio (r) e altura (a) da lata.

$$V = \pi \cdot r^2 \cdot a$$

3. Construa um algoritmo que leia três valores numéricos inteiros e positivos (A, B, C). Após, calcule a seguinte expressão e mostre o resultado.

$$D = \frac{R + S}{2}, \text{ onde } R = (A + B)^2 \\ S = (B + C)^2$$

4. Escreva um algoritmo que leia dois valores numéricos inteiros A e B. Após, apresente o resultado da subtração de A por B. Notem que mesmo A sendo menor que B, o resultado deverá ser sempre positivo.

5. Elabore um algoritmo que leia dois valores numéricos **A** e **B**. O algoritmo não deverá permitir que o valor de **A** seja maior que o valor de **B**. Caso os valores sejam válidos, apresentar o resultado da expressão A^B .

6. Elabore um algoritmo que receba um valor inteiro, logo implemente uma função recursiva que execute a soma de todos os valores de n até 0.

7. Elabore um algoritmo que receba um vetor de n posições, logo após implemente uma função que receba o vetor e o tamanho dele e calcule de forma recursiva a soma dos elementos deste vetor.

8. Escreva um algoritmo que simule uma eleição. O algoritmo deverá ficar lendo votos enquanto não for digitada a condição de saída. A condição de saída é o usuário digitar o valor 6. Limpar a tela e mostrar novamente a tela da eleição a cada voto.

Veja a tela do algoritmo

| E L E I Ç Ã O – sistema de votação |

1. Votar em João
2. Votar em Maria
3. Votar em Marcos
4. Votar em branco
5. Anular o voto
6. Finalizar a votação

Na construção do algoritmo devemos criar os seguintes procedimentos:

funcao vazio montarTela()

monta a tela apresentada acima.

funcao vazio computarVoto(inteiro voto)

acrescenta o voto para um candidato.

funcao vazio apurarEleicao()

mostra os resultados da eleição, após finalizar.

Quando finalizar a votação, o procedimento **apurarEleicao()** deve apresentar a tela de apuração, conforme exemplo abaixo (notem que os valores informados são hipotéticos):

| E L E I Ç Ã O – resultado final |

João: 10 (10%) votos.

Maria: 25 (25%) votos.

Marcos: 15 (15%) votos.

Branco: 30 (30%) votos.

Nulos: 20 (20%) votos.

Total de votos: 100

9. Escreva um algoritmo recursivo para o seguinte problema: encontre o valor de um elemento máximo de um vetor de inteiros. É claro que o problema só faz sentido se $n \geq 1$.

10. Escreva um algoritmo recursivo para o seguinte problema: verificar se um dado inteiro x é elemento de vetor. A seguinte função retorna *sim* se x está no vetor ou retorna *não* em caso contrário. Note que o problema faz sentido com qualquer $n \geq 0$ (O algoritmo trata da instância $n = 0$ com naturalidade).

REFERÊNCIAS

<https://sourceforge.net/projects/portugolstudio/>

FORBELLONE, André Luiz Villar, EBERSPASCHER, Henri Frederico. Lógica de Programação: a construção de algoritmos e estrutura de dados. 2.ed. São Paulo: Makron Books, 2000.

SALVETTI, Dirceu Douglas, BARBOSA, Lisbete Madsen. Algoritmos. São Paulo: Makron Books, 1998.

ASCENCIO. Ana Fernanda Gomes. Lógica de Programação com Pascal. São Paulo: Makron Books, 1999.

MIZRAHI. Victorine Viviane. Treinamento em Linguagem C++: curso completo módulo 1. São Paulo: McGraw-Hill, 1990.

[COR98] CORDEIRO, José. Transição para C++ - para programadores de C. Escola Superior de Tecnologia, Instituto Politécnico de Setúbal, 1998.

SEVERO, Carlos Emilio Padilha – PortugolStudio - APOSTILA