



CEEP – CENTRO ESTADUAL DE EDUCACAO PROFISSIONAL PEDRO BOARETTO NETO
RUA Natal, 2800 - JD. Tropical - 85807-100 - Cascavel- Paraná

Curso Técnico em Informática – 1º SEMESTRE

Professor: Ronye Peterson Cordeiro

Conteúdo com base na apostila do Professor Adair Santa Catarina - UNIOESTE (2000)

LINGUAGEM DE PROGRAMAÇÃO

CASCATEL - PR

2022

1.1 NOÇÕES DE LÓGICA	3
1.1.1 Conceituação:	3
1.1.2 Objetivos:	3
1.2 ALGORITMOS	3
Prática 1:	4
2 UNIDADE 2 – ALGORITMOS E PSEUDOCÓDIGO	4
2.1 INTRODUÇÃO	4
2.2 PROGRAMAÇÃO ESTRUTURADA	4
2.3 VANTAGENS	4
2.4 TIPOS PRIMITIVOS	4
2.5 VARIÁVEIS	5
2.6 FORMAÇÃO DE IDENTIFICADORES	5
2.6.1 Regras para formação de identificadores	5
2.7 EXPRESSÕES ARITMÉTICAS	6
2.8 FUNÇÕES MATEMÁTICAS	7
2.9 PRIORIDADES	7
2.10 EXPRESSÕES LÓGICAS	8
2.11 HIERARQUIA GERAL DOS OPERADORES	9
3 INSTRUÇÕES PRIMITIVAS	9
3.1 COMANDOS DE ATRIBUIÇÃO	9
3.2 ENTRADA E SAÍDA DE DADOS	10
3.3 BLOCOS	10
3.4 MÉTODO PARA A CONSTRUÇÃO DE ALGORITMOS	11
Prática 2:	12
4 UNIDADE 4 - ESTRUTURAS DE CONTROLE	14
4.1 ESTRUTURA SEQUENCIAL	14
4.2 ESTRUTURAS DE SELEÇÃO	14
4.3.1 SELEÇÃO SIMPLES	14
4.3.2 SELEÇÃO COMPOSTA	14
4.3.3 SELEÇÃO ENCADEADA	15
Prática 3:	16
4.4 ESTRUTURAS DE REPETIÇÃO	17
4.4.1 REPETIÇÃO COM TESTE NO FINAL	17
Prática 4:	18
4.4.2 REPETIÇÃO COM TESTE NO INÍCIO	19
Prática 5:	19
4.4.3 REPETIÇÃO COM VARIÁVEL DE CONTROLE	21
Prática 6:	22

UNIDADE 1 – CONCEITOS BÁSICOS EM ALGORITMOS

1.1 NOÇÕES DE LÓGICA

1.1.1 Conceituação:

- É a ciência que estuda as leis do raciocínio, coerência;
- Como técnica, nos ensina a usar corretamente as leis do pensamento;
- É a arte de pensar corretamente e vista que, a forma mais complexa de pensamento é o raciocínio, a lógica estuda a correção do raciocínio;
- Tem em vista a ordem da razão, uma vez que a razão pode funcionar desordenadamente, a lógica ensina a colocar ordem no pensamento.

1.1.2 Objetivos:

É o pensamento que traduz as relações dos conteúdos mentais, como tais suscetíveis, sendo pensados e identificados por muitos. O homem é, por natureza, apto para pensar retamente (lógica natural), mas precisa de lógica científica, principalmente quando se trata de por a prova seu pensamento em casos difíceis ou controversos.

Exemplos de raciocínio:

Conclusão: Os animais são dotados de instintos. Todo cachorro é um animal, portanto, todo cachorro tem um instinto.

Relação/Comparação: Maria tem 15 anos. João é mais velho que Maria, portanto, João tem mais de 15 anos.

Sequência: O paletó está dentro do guarda-roupa. O guarda-roupa está fechado. Preciso abrir o guarda-roupa para pegar o paletó.

Exclusão: Assaltaram um banco. Pegaram três suspeitos: Carlos, Antonio e Pedro. Pedro acabou de sair da lanchonete. Carlos é o vendedor de pipocas da rua. Quem assaltou o banco?

1.2 ALGORITMOS

Algoritmo: É a descrição de um conjunto de ações que, obedecidas, resultam em uma sucessão finita de passos, atingindo um objetivo.

Algoritmos não-computacionais: Todo algoritmo que não necessita de resolução computacional. Exemplos: Passos para atender um telefone; passos para trocar uma lâmpada; passos para chegar ao colégio.

Exemplo: Trocar uma lâmpada.

- pegue a escada;
- posicione-a embaixo da lâmpada;
- busque a lâmpada nova;
- suba na escada;
- retire a lâmpada velha;
- coloque a lâmpada nova;
- desça da escada;
- guarde a escada.

Algoritmos computacionais: É uma sequência ordenada e finita de operações bem definidas e eficazes que, quando executadas por um computador, operando sobre dados que caracterizam o estado atual do contexto e o estado desejado, sempre termina em um determinado tempo, produzindo uma solução ou indicando que a solução não pode ser obtida. Algoritmos são formas usadas para facilitar a programação de computadores. Por ser flexível, seu uso não está ligado a nenhuma linguagem específica. É um programa escrito em nosso idioma.

Exercícios:

- 1) Faça um algoritmo não computacional para o caminho que você faz para chegar ao colégio;
- 2) Faça um algoritmo não computacional para fritar um ovo.
- 3) Elabore três exemplos de raciocínio e classifique cada um conforme os tipos: (Conclusão, Relação/Comparação, Sequência e Exclusão).

Prática 1:

Abaixo um link com vídeo sobre um exercício envolvendo conceitos de lógica com o software GCompris:

Jogo Super Inteligencia no software GCompris:

<https://www.youtube.com/watch?v=O-mfgew2QCQ>

2 UNIDADE 2 – ALGORITMOS E PSEUDOCÓDIGO

2.1 INTRODUÇÃO

Um algoritmo é uma sequência ordenada, finita de operações bem definidas e eficazes que, quando executadas por um computador operando sobre dados que caracterizam o estado atual do contexto e o estado desejado, sempre termina em um determinado tempo, produzindo uma solução ou indicando que a solução não pode ser obtida.

Algoritmos são formas usadas para facilitar a programação de computadores. Por ser flexível, seu uso não está ligado a nenhuma linguagem específica. É um programa escrito em nossa língua.

2.2 PROGRAMAÇÃO ESTRUTURADA

Programação estruturada é aquela feita em blocos de comandos. Desta forma são evitados saltos incondicionais e a falta de clareza nos códigos fontes. É um método para a construção de algoritmos.

2.3 VANTAGENS

- Facilidade de se compreender o código fonte;
- Alterações do código fonte são mais fáceis e rapidamente efetuadas;
- Melhor controle do uso da memória;
- Programação mais fácil; e
- Portabilidade.

2.4 TIPOS PRIMITIVOS

Está relacionado com a maneira como o computador manipula dados. Podem ser dos seguintes tipos:

- a) **Inteiro:** Todo e qualquer dado numérico que $\in \mathbb{Z}$. São os números sem casas decimais.
- b) **Real:** Todo e qualquer dado numérico que $\in \mathbb{R}$, São números com casas decimais.
- c) **Caracter** ou **texto:** Todo e qualquer dado composto por um conjunto de caracteres alfa-numéricos e/ou especiais. São valores que colocamos com aspas, simples (' ') ou duplas (" ");
- d) **Lógico:** Toda e qualquer informação que pode apenas assumir duas situações (bi-estável). São representados em computação com valores *true* (verdadeiro) ou *false* (falso);

Exercícios:

4) Classificar os dados abaixo:

- | | | | | |
|--------------|----------|------------|----------|-----------|
| a) 516 | b) 516,7 | c) -516,8 | d) "516" | e) -3,19 |
| f) +15 | g) V | h) "4 - 2" | i) 0 | j) "Amém" |
| k) "A + B/2" | l) true | m) false | | |

5) Indicar o tipo do resultado (Inteiro ou Real) de cada uma das expressões onde: A = 5 B = 2,2 C = 8 D = 1,5.

- | | | | | |
|----------------|---------------|----------------|----------------|----------|
| a) $5 + 4 - 2$ | b) $10 - 3,4$ | c) $A + C$ | d) $4 + B - D$ | e) $5/2$ |
| f) $A * 2,4$ | g) $A * B$ | h) $A + B * B$ | | |

6) Expressar o tipo conforme o dado:

- | | | | | |
|-------------------|---------|-----------------------------|----------|-------------------|
| a) distância | b) peso | c) R\$ | d) idade | e) ano |
| f) nome de pessoa | | g) contador de página | | h) nome de cidade |
| i) temperatura | | j) sinalizador de passagem. | | |

2.5 VARIÁVEIS

Uma formação é classificada como variável quando tem a possibilidade de ser alterada em algum instante no decorrer do tempo. Num programa de computador, uma variável é uma entidade que possui um valor, sendo conhecida no programa por um nome. Uma variável pode receber muitos valores diferentes, mas, num dado instante, ela pode ter somente um valor. Nos algoritmos, cada variável corresponde a uma posição de memória cujo conteúdo pode variar de acordo com o tempo durante a execução.

a) Declaração de Variáveis:

No ambiente computacional, as informações variáveis são guardadas em dispositivos eletrônicos analogamente chamados de memória. Podemos chamar essa memória como sendo um armário repleto de gavetas, no qual as gavetas seriam os locais físicos responsáveis por armazenar objetos.

Os objetos que podem ser substituídos seriam os dados e as gavetas, as variáveis. Para identificar o tipo de uma ou mais variáveis é usada a declaração de variáveis. Além disso, no momento em que se declara uma variável, é feita a associação do nome escolhido com a respectiva posição de memória, que o mesmo passa a simbolizar. Uma vez declarada uma variável, qualquer referência ao seu identificador implica a referência ao conteúdo do local da memória representado pelo mesmo.

2.6 FORMAÇÃO DE IDENTIFICADORES

Um identificador é formado por um ou mais caracteres que servem para dar um nome para valores em mutação. Este nome também servirá para representar um espaço na memória.

2.6.1 Regras para formação de identificadores

- a) Devem começar por um caractere alfabético (letra);
- b) Podem ser seguidos por mais caracteres alfabéticos e numéricos ou '_';
- c) Não é permitido o uso de caracteres especiais com exceção do '_';

Os itens abaixo não são regras, mas sim boas práticas para algoritmos:

- d) Os caracteres alfabéticos devem, preferencialmente, ser escritos em maiúsculas;
- e) Devem possuir nomes que lembrem o conteúdo que representam.

Sintaxe:

lista de variáveis : tipo;

Exemplos:

ANO, DIA, MES, CONTA, SOMA : inteiro;
NOME, CIDADE, BAIRRO, EMAIL : caracter;
MEDIA_NOTAS, PESO, DOLAR : real;

Exercícios:

7) Assinale os identificadores válidos:

- | | | | |
|---------------------------------------|----------------------------------|-------------------------------------|--------------------------------------|
| <input type="checkbox"/> (X) | <input type="checkbox"/> #AB | <input type="checkbox"/> ANTERIOR | <input type="checkbox"/> U2 |
| <input type="checkbox"/> KM/H | <input type="checkbox"/> B45 | <input type="checkbox"/> "NOME" | <input type="checkbox"/> CANTOS |
| <input type="checkbox"/> NOME CLIENTE | <input type="checkbox"/> CLIENTE | <input type="checkbox"/> MEDIA*ARIT | <input type="checkbox"/> MEDIA_ARIT. |
| <input type="checkbox"/> UF | <input type="checkbox"/> B+A | <input type="checkbox"/> DATA | <input type="checkbox"/> A>B |
| <input type="checkbox"/> 0 | <input type="checkbox"/> ALTURA | <input type="checkbox"/> EXISTE | |

8) Suponha as seguintes variáveis, declare-as corretamente (dê um nome e coloque o tipo):

EXEMPLOS:

Média dos alunos: MED_ALUNO : real

Nome do professor: NOME_PROF : caracter

- Estado Civil:
- Idade:
- Dia/mês/ano:
- Nome da Empresa:
- Telefone:
- Nome de pessoa para contato:
- Salário Bruto do mês:
- Código do Funcionário:
- Valor do ICMS a ser recolhido:
- Valor total dos descontos:
- Contador de registros no arquivo:
- Presença/ausência de registro:
- Quantidade de alunos da turma A:
- Série da Nota Fiscal:
- Existência de arquivo:

2.7 EXPRESSÕES ARITMÉTICAS

Denominamos expressões aritméticas aquelas cujos operadores são aritméticos e cujos operandos são constantes e/ou variáveis do tipo numérico.

a) Operadores Aritméticos:

Chamamos de operadores aritméticos o conjunto de símbolos que representam as operações básicas da matemática, a saber: +, -, *, /.

b) Operações Matemáticas não convencionais:

mod: resto da divisão inteira;

div: quociente da divisão inteira.

c) Funções auxiliares

exp() ou **^** -> Potência.

raizQ() -> Raiz Quadrada

d) Símbolos auxiliares:

Os símbolos auxiliares são os parênteses, cuja principal função é modificar a prioridade da execução.

2.8 FUNÇÕES MATEMÁTICAS

Apesar das melhores intenções dos projetistas de linguagens, não é possível especificar todas as operações requeridas nos programas na forma de operadores convencionais tais como: +, -, *, /. Frequentemente, o conjunto convencional de operadores é complementado por operadores especiais denominados funções embutidas. Funções embutidas são rotinas pré-escritas para auxiliar o programador na execução de cálculos que requeiram mais do que o conjunto convencional de operadores.

Ex: **Abs (x)** = valor absoluto de x;

$$\text{abs}(-3) = 3 \quad \text{abs}(-3,5) = 3,5 \quad \text{abs}(5) = 5 \quad \text{abs}(8,5) = 8,5 \quad \text{abs}(-0,5) = 0,5$$

Int (x) = parte inteira de x;

$$\text{int}(-3) = -3 \quad \text{int}(-3,5) = -3 \quad \text{int}(5) = 5 \quad \text{int}(8,5) = 8 \quad \text{int}(-0,5) = 0$$

Frac (x) = parte fracionária de x.

$$\text{frac}(-3) = 0,0 \quad \text{frac}(-3,5) = -0,5 \quad \text{frac}(5) = 0,0 \quad \text{frac}(8,5) = 0,5 \quad \text{frac}(-0,5) = -0,5$$

OBS: x pode ser um número, variável, expressão aritmética ou função matemática.

2.9 PRIORIDADES

Na resolução de expressões aritméticas e funções matemáticas os elementos guardam entre si uma hierarquia.

- Parênteses mais internos;
- Funções matemáticas;
- exp ou ^, raizQ;
- *, / (div ou mod);
- +, -;
- relações.

Para as operações de mesma prioridade, seguimos a ordem especificada, isto é, primeiro resolvemos as operações da esquerda para a direita. Para alterarmos a prioridade utilizaremos os parênteses.

Exercício:

9) Monte as funções que se pedem (Somente montar, não resolver):

- a) Inteiro de 3,1416 menos o absoluto de -2,75.
- b) Fracionário de 2,7 mais o inteiro de X.
- c) A média aritmética de 80 e 60;
- d) O resto da divisão de 5 por 2.
- e) Quociente da divisão de 8 por 3.

2.10 EXPRESSÕES LÓGICAS

Denominamos “Expressão Lógica” aquela cujos operadores são lógicos e/ou relacionais e cujos operandos são relações e/ou variáveis e/ou constantes. Simplificando, são expressões que usam comparações para saber se um resultado é verdadeiro ou falso.

a) Operadores Relacionais:

Utilizamos os operadores relacionais para realizar comparações entre dois valores. Tais valores são representados por constantes, variáveis ou expressões aritméticas.

Ex: =, >, <, >=, <=, <>.

Exercício:

10) Analise as expressões e responda V ou F sendo: A = 1, B = 2, C = 3, D = 4, E = 5, G = -4

- a) () $A * B + C = D - E$
- b) () $C * D + E = E * B + C + D$
- c) () $A + E + D - C > C * B - E + C$
- d) () $B * (D - B) + \text{int}(G) < \text{ABS}(G) * C$
- e) () $G * C - B + C > E - C * B + A + D/B$

b) Operadores Lógicos:

Utilizaremos operadores lógicos para formar proposições. Os operadores lógicos são:

E = conjunção;

OU = disjunção (não exclusiva);

NÃO = negação;

c) Tabela Verdade:

É o conjunto de todas as possibilidades combinatórias entre os valores de diversas variáveis lógicas, as quais se encontram em apenas duas situações, e um conjunto de operadores lógicos.

1) Conjunção (e, and):

Com as proposições P e Q, temos que (P e Q) tem valor lógico verdade se ambas as proposições forem verdadeiras e valor lógico falso nos demais casos.

P	Q	P e Q
V	V	V
V	F	F
F	V	F
F	F	F

2) Disjunção (não exclusiva) (ou):

Com duas proposições P e Q, temos que (P ou Q) tem valor lógico verdade em todos os casos exceto se ambas as proposições forem falsas.

P	Q	P ou Q
V	V	V
V	F	V
F	V	V
F	F	F

3) Negação:

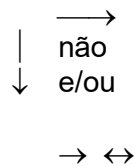
A negação de uma proposição P representada por (não P) é a verdade se P for F e falsa se P for V.

P	Não P
V	F
F	V

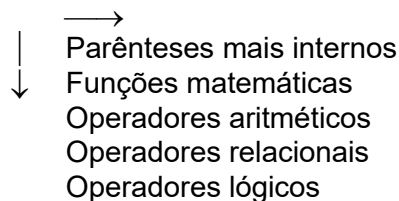
d) Proposições Compostas:

Para construir a tabela verdade de uma proposição composta analisamos as proposições simples que a compõe.

e) Hierarquia das Proposições Lógicas:



2.11 HIERARQUIA GERAL DOS OPERADORES



Exercício:

11) Obtenha V ou F observando a hierarquia

- a) não $(2^3 < 4^2)$ ou $ABS(INT(15/-2)) < 10$
- b) não $(5 < 10/2)$ ou V e $-2,5 > 5-2$ ou V
- c) $int(17/2) < 4+2$ ou $2+3*5/(3 \bmod 5) > 0$

3 INSTRUÇÕES PRIMITIVAS

3.1 COMANDOS DE ATRIBUIÇÃO

A operação de atribuição é uma forma para especificar que será dado um valor a uma variável. A operação de atribuição será indicada por “:=”.

Exemplo:

$A := 3$ { Lê-se A recebe 3 }

Esta é uma operação dita “destrutiva”, pois qualquer valor que a variável possua antes do processamento da operação de atribuição se perde, sendo substituído pelo novo valor. É importante lembrar que só é possível atribuir a variável um valor classificado do mesmo tipo que ela.

Exemplo:

Comandos no software Visual G

```
VAR //Declaracao de variaveis
    A: inteiro
    NOME: caracter
    AB: real

    // atribuindo valores as variaveis
    A := 7
    NOME := "Cascavel"
    AB := 5.433
```

3.2 ENTRADA E SAÍDA DE DADOS

Os cálculos do computador são de pouco valor, a não ser que: primeiro possamos fornecer os dados sobre os quais esses cálculos serão efetuados; segundo, ver os resultados desses cálculos.

Uma vez que as operações de E/S estão muito relacionadas às instruções dos programadores, sua forma é altamente dependente da linguagem de programação específica utilizada e, às vezes, do próprio sistema de computação.

a) Entrada:

O comando "leia" permite ler valores atribuindo-os as variáveis indicadas.

Sintaxe: leia (lista de variáveis);

Exemplo: leia (X, A, NOTA, NOME)
 leia (NOME)

b) Saída:

O comando "escreva" nos permite mostrar resultados. A saída pode aparecer no vídeo ou na impressora.

Sintaxe: escreva (lista de variáveis);

Exemplo: escreva (X, Y, Z)
 escreva ("A media final eh: ", MEDIA)

3.3 BLOCOS

Um bloco pode ser definido como um conjunto de ações com uma função definida, neste caso, um algoritmo pode ser visto como um bloco. Ele serve para definir os limites nos quais as variáveis declaradas em seu interior são conhecidas.

Para delimitar um bloco, utilizaremos os delimitadores "início" e "fim".

Sintaxe:

```
    declaração de variáveis;
    início
        seqüência de ações;
    fim
```

Exemplo em Visual G:

```
algoritmo "exemplo1"  
var  
    IDADE: inteiro  
    NOME: caracter  
inicio  
    escreva ("Digite sua idade: ")  
    leia (IDADE)  
    escreva ("Digite seu nome: ")  
    leia (NOME)  
    escreva (NOME," ", IDADE)  
fimalgoritmo
```

3.4 MÉTODO PARA A CONSTRUÇÃO DE ALGORITMOS

- 1) Leia atentamente o enunciado do problema;
- 2) Procure incorporar comentários no algoritmo, pelo menos para descrever as atividades mais complexas.

Exemplo: { Classificação do arquivo CADCI }

- 3) Retire do enunciado a relação de entrada e saída de dados;
- 4) Escolha nomes de variáveis que sejam significantes, isto é, que traduzam o tipo de informação a ser armazenado.

Exemplo: NOTA1, NOME, CEP, CPF.

- 5) Grife as palavras-chave escritas em letras minúsculas, destacando a estrutura de controle.
- 6) Procure alinhar os comandos de acordo com o nível a que pertencem, isto é, destaque a estrutura a que estão contidos.

Exemplo: se A>B então

```
{ comando 1;  
  comando 2;  
  comando 3;  
  comando 4;  
  ...  
}
```

- 7) Construir o algoritmo determinando o que deve ser feito para transformar as entradas em saídas especificadas.
- 8) Executar o teste de mesa, ou seja, executar as ações descritas seguindo o fluxo de execução estabelecido.

Exercícios:

12) (EXEMPLO) Fazer um algoritmo para calcular a área de um retângulo (Área = base * altura)

Argumentos de entrada: BASE, ALTURA.

Argumentos de saída: AREA.

Em VisualG:

```
algoritmo "area_retangulo"
var
    BASE, ALTURA, AREA: real
inicio
    escreva ("Digite a base: ")
    leia (BASE)
    escreva ("Digite a altura: ")
    leia (ALTURA)
    AREA := BASE * ALTURA
    escreva ("A area do retangulo eh: ", AREA)
fimalgoritmo
```

Prática 2:

Veja o vídeo com o exemplo do algoritmo do exercício 16 no VisuAlg:

Visualg - Area de um retângulo

<http://youtu.be/Ty04H0DZ-Wo>

Este é um outro link sobre utilização do VisuAlg com vários comandos:

VisuALG - Aula 01 (Princípios Básicos)

<https://www.youtube.com/watch?v=dZq7l9Oj-c>

13) Faça um algoritmo para calcular a área de um triângulo.

14) Dadas 3 notas digitadas pelo usuário, faça um algoritmo para calcular a média aritmética.

Visualg - Média de 3 notas

Link: <http://youtu.be/HvHonFgYEcc>

15) Faça um algoritmo para calcular o salário líquido de um empregado onde:

SB = salário bruto DES = descontos AD = adicionais SL = salário líquido

Fórmula: $SL := SB + AD - DES$

16) Dadas duas notas, Nome e Matricula de um aluno, faça um algoritmo para calcular a soma e também a média aritmética das notas:

17) Dada uma idade (em anos), converter essa idade em horas e em meses.

18) Escreva um algoritmo para ler o salário mensal atual de um funcionário e o percentual de reajuste. Calcular e escrever o valor do novo salário.

- 19) Faça um algoritmo que dado um número qualquer obtenha a soma do número com seus 3 consecutivos.
- 20) Faça um programa que converta uma temperatura dada em Fahrenheit para Celsius. Fórmula: $C := 5/9*(F-32)$.
- 21) Faça um algoritmo para calcular a área de um círculo: $Area := \pi * R^2$
- 22) Há um erro no algoritmo da figura. Identifique o erro e indique a linha e o comando que deve ser o correto.

```
1 algoritmo "area retangulo"
2 var
3     BASE, ALTURA, AREA: real
4 inicio
5     escreva ("Digite a base: ")
6     leia (BA)
7     escreva ("Digite a altura: ")
8     leia (ALTURA)
9     AREA := BASE * ALTURA
10    escreva ("A area do retangulo eh: ", AREA)
11 fimalgoritmo
12
```

Linha: _____

Comando correto: _____

- 23) No algoritmo abaixo, acrescente uma variável para o usuário informar o salário e mostrar qual foi o salário informado.

```
1 algoritmo "Exercicio23"
2 var
3     IDADE: inteiro
4     NOME: caracter
5 inicio
6     escreva ("Digite sua idade: ")
7     leia (IDADE)
8     escreva ("Digite seu nome: ")
9     leia (NOME)
10    escreva (NOME, " ", IDADE)
11 fimalgoritmo
12
```

4 UNIDADE 4 - ESTRUTURAS DE CONTROLE

Na criação de algoritmos, utilizamos os conceitos de bloco lógico, entrada e saída de dados, variáveis, constantes, atribuições, expressões lógicas, relacionais e aritméticas, bem como comandos que traduzam estes conceitos de forma a representar o conjunto de ações. Através das estruturas básicas de controle do fluxo de execução (sequencialização, seleção e repetição) e da combinação delas, poderemos criar um algoritmo para solucionar qualquer problema.

4.1 ESTRUTURA SEQUENCIAL

É o conjunto de ações primitivas que serão executadas numa sequência linear, de cima para baixo e da esquerda para a direita.

4.2 ESTRUTURAS DE SELEÇÃO

Uma estrutura de seleção permite a escolha de um grupo de ações a serem executadas quando determinadas condições, representadas por expressões lógicas, são ou não satisfeitas.

4.3.1 SELEÇÃO SIMPLES

Visual G -> sintaxe

```
se condição A entao  
    comando 1  
    comando 2  
    comando 3  
fimse
```

OBS: O bloco de comandos só será executado caso o resultado da expressão lógica “Condição A” seja verdadeiro.

4.3.2 SELEÇÃO COMPOSTA

Visual G -> sintaxe

```
se condição A entao  
    comando A1  
    comando A2  
    comando A3  
senao  
    comando B1  
    comando B2  
    comando B3  
fimse
```

OBS: Os “Comandos B” só serão executados quando o resultado de “Condição A” for falso.

4.3.3 SELEÇÃO ENCADEADA

Quando, devido a necessidade de processamento, agruparmos várias seleções, formaremos uma seleção encadeada. Normalmente, tal seleção ocorre quando uma determinada ação ou bloco deve ser executado se um grande conjunto de possibilidades ou combinações de situações for satisfeito.

a) Seleção encadeada homogênea:

Chamamos de seleção encadeada homogênea quando a construção de diversas estruturas de seleção encadeada segue um determinado padrão lógico.

Vamos supor que num algoritmo um comando genérico dado deva ser executado apenas quando forem satisfeitas as condições COND1, COND2, COND3, ...

```
se condição 1 entao
    se condição 2 entao
        se condição 3 entao
            comando 1
            comando 2
            comando 3
        fimse
    fimse
fimse
```

ou

```
se condição 1 entao
    comando 1
senao
    se condição 2 entao
        comando 2
    senao
        se condição 3 entao
            comando 3
        fimse
    fimse
fimse
```

b) Seleção encadeada heterogênea

Podemos construir uma estrutura de seleção (se) de diversas formas, sendo que ao encadearmos várias seleções as possibilidades de construção diferentes tendem a um número elevado. Quando não conseguimos identificar um padrão lógico de construção em uma estrutura de seleção encadeada, dizemos que esta é uma estrutura de seleção encadeada heterogênea.

```
se condição entao
    comando 1
    se condição 2 entao
        comando 2
        comando 3
    fimse
senao
    se condição 3 entao
        comando 4
```

```

comando 5
se condição 4 entao
    se condição 5 entao
        comando 6;
    fimse
fimse
fimse
fimse

```

Prática 3:

Veja o vídeo com um exemplo do algoritmo utilizando o comando se-entao-senao no VisuAlg:

Visualg - Numero par ou ímpar

<http://youtu.be/HVNjuV72hA4>

Exercícios:

- 24) Fazer um algoritmo para verificar se um indivíduo pode fazer carteira de habilitação a partir da idade informada pelo usuário.
- 25) Fazer um algoritmo para calcular a média aritmética entre 2 notas digitadas pelo usuário e informe:
 - aprovado se a média $\geq 7,0$
 - reprovado se a média $< 4,0$
 - exame se a média $< 7,0$ e média $\geq 4,0$
- 26) Fazer um algoritmo para calcular o dobro de um número inteiro natural informado pelo usuário caso seja par, ou o triplo deste número caso o número seja ímpar.
- 27) Dados 3 números, imprimir o maior e o menor.
- 28) Elaborar um algoritmo em que dada a idade de um nadador, classifique-o em uma das seguintes categorias: infantil A (5 a 7 anos), infantil B (8 a 10 anos), juvenil A (11 a 13 anos), juvenil B (14 a 17 anos), senior (>17 anos).
- 29) No algoritmo abaixo, qual será o valor da variável N que será mostrado na tela? _____

```

1 algoritmo "SE-ENTAO-SENAO"
2 var
3     N: INTEIRO
4 inicio
5     N:=3
6     se N mod 2 = 1 entao
7         N:= N * 3
8     senao
9         N:= N * 2
10    fimse
11    escreva (N)
12 fimalgoritmo

```


30) Elaborar um algoritmo para ler dois valores e imprimir uma das três mensagens a seguir:

- 'Números iguais', caso os números sejam iguais
- 'Primeiro é maior', caso o primeiro seja maior que o segundo;
- 'Segundo maior', caso o segundo seja maior que o primeiro.

31) Modifique o algoritmo abaixo para que seja mostrado na tela como "neutro" caso o número informado na variável N seja 0 (zero).

```
1 algoritmo "positivo_negativo_neutro"
2 var
3   N: inteiro
4 inicio
5   escreval ("Digite um numero inteiro: ")
6   leia (N)
7   se n > 0 entao
8     escreval ("positivo")
9   senao
10    escreval ("negativo")
11  fimse
12 fimalgoritmo
```

4.4 ESTRUTURAS DE REPETIÇÃO

4.4.1 REPETIÇÃO COM TESTE NO FINAL

Para realizar a repetição com teste no final, utilizaremos a estrutura "repita" que permite que um bloco ou ação primitiva seja repetida "até" que uma determinada condição seja verdadeira. Sintaxe:

```
repita
  comando A
  comando B
  comando C
  comando D
ate <expressão lógica>
```

Pela sintaxe da estrutura, perceberemos que o bloco é executado pelo menos uma vez, independente da validade da condição. Isto ocorre porque a inspeção da condição é feita após a execução do bloco.

Exemplo 1 em VisualG:

```
algoritmo "Exemplo_Repita"
var
N: inteiro
Inicio
N:=0
  repita
    escreva (N)
    N := N + 2
  ate N > 20
fimalgoritmo
```

Exemplo 2 em VisualG:

```
algoritmo "Exemplo_Repita_Soma"
var
N,SOMA: inteiro
Inicio
    N:=0
    repita
        escreva (N)
        SOMA:= SOMA + N
        N := N + 2
    ate N > 20
    escreva ("A Soma dos numeros pares entre 0 e 20 é: ", SOMA)
finalgoritmo
```

Prática 4:

Veja o vídeo com um exemplo do algoritmo utilizando o comando repita no VisualG:

Visualg - Exemplo com comando REPITA

<http://youtu.be/TfLbvZ88E2k>

Exercícios:

- 32) Fazer um algoritmo para escrever na tela os números ímpares no intervalo [7, 77].
- 33) Fazer um algoritmo para somar os números pares existentes entre 50 e 100. (O resultado da soma será 1950)
- 34) Fazer um algoritmo para escrever na tela os números pares no intervalo [25, 55].
- 35) Fazer um algoritmo para somar os números múltiplos de 4 entre 10 e 40. (O resultado da soma será 208)
- 36) Fazer um algoritmo para escrever seu nome na tela 15 vezes.
- 37) Fazer um algoritmo para informar 2 notas (que devem ser entre 0 e 10) e apresentar a média.
- 38) O algoritmo abaixo deveria apresentar os números pares entre 0 e 20, porém ele está com o chamado “looping infinito”. Identifique a linha e como ele pode ser resolvido para apresentar o que é desejado: 0 2 4 6 8 10 12 14 16 18 20

```
1 algoritmo "Execicio_38"
2 var
3 N: inteiro
4 Inicio
5     N:=0
6     repita
7         escreva (N)
8         N := N + 2
9     ate N = 21
10 finalgoritmo
```

4.4.2 REPETIÇÃO COM TESTE NO INÍCIO

Permite executar diversas vezes um mesmo trecho do algoritmo, porém, sempre verificando antes de cada execução se é permitido repetir o mesmo trecho.

Sintaxe:

```
enquanto <condição> faca  
    comando A  
    comando B  
    comando C  
fimenquanto
```

Para realizar a repetição com teste no início, utilizamos a estrutura “enquanto” que permite que um bloco ou uma ação primitiva seja repetida enquanto uma determinada condição for verdadeira.

Quando o resultado da condição for falso, o comando é abandonado. Se já da primeira vez o resultado for falso, os comandos não serão executados nenhuma vez.

Exemplo 1 em VisualG:

```
algoritmo "Exemplo_Enquanto"  
var  
N: inteiro  
Inicio  
    N:=0  
    enquanto N <= 20 faca  
        escreva (N)  
        N := N + 2  
    fimenquanto  
fimalgoritmo
```

Exemplo 2 em VisualG:

```
algoritmo "Exemplo_Enquanto_Soma"  
var  
N,SOMA: inteiro  
Inicio  
    N:=0  
    enquanto N <= 20 faca  
        escreva (N)  
        SOMA:= SOMA + N  
        N := N + 2  
    fimenquanto  
    escreva ("A Soma dos numeros pares entre 0 e 20 é: ", SOMA)  
fimalgoritmo
```

Prática 5:

Veja o vídeo com um exemplo do algoritmo utilizando o comando ENQUANTO no VisualG:

Visualg - Exemplo com comando ENQUANTO

<http://youtu.be/jImSvUKKE6Y>

Exercícios:

- 39) Fazer um algoritmo que mostre os números ímpares que são múltiplos de 3 que se encontram no conjunto dos números de 1 a 500.
- 40) Dada uma sequência de números, informados pelo usuário, obter a soma dos positivos e a quantidade dos negativos. O ponto de parada ocorrerá quando o número for zero.
- 41) Escreva um algoritmo para informar a altura de várias pessoas e indicar qual a maior altura entre as informadas. O ponto de parada ocorrerá quando a altura for zero.
- 42) Calcular o produtório de uma sequência qualquer dada, informando um número inicial e um número final. Sabe-se que:

Produtório entre 3 e 6

$$\prod_3^6 = 3 * 4 * 5 * 6 = 360$$

Produtório entre 2 e 7

$$\prod_2^7 = 2 * 3 * 4 * 5 * 6 * 7 = 5040$$

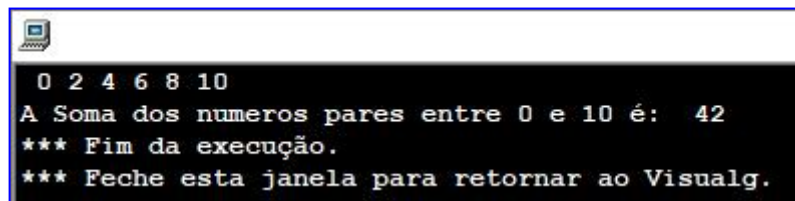
- 43) Calcular o fatorial de um número qualquer. O fatorial de 0 e 1 é igual a 1.

Exemplos: 5! -> 1*2*3*4*5 = 120

4! -> 1*2*3*4 = 24

- 44) O algoritmo abaixo deveria apresentar como resposta para a soma dos números pares entre 0 e 10 um resultado de 30 (0+2+4+6+8+10). Porém da forma que está desenvolvido não apresenta o desejado. Modifique o algoritmo para que tenha o resultado correto.

```
1 algoritmo "Exercicio_44"
2 var
3 N, SOMA: inteiro
4 Inicio
5     N:=0
6     enquanto N <= 10 faca
7         escreva (N)
8         N := N + 2
9         SOMA:= SOMA + N
10    fimenquanto
11    escreval
12    escreva ("A Soma dos numeros pares entre 0 e 10 é: ", SOMA)
13 fimalgoritmo
```



```
0 2 4 6 8 10
A Soma dos numeros pares entre 0 e 10 é: 42
*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```

4.4.3 REPETIÇÃO COM VARIÁVEL DE CONTROLE

Nas estruturas de repetição vistas até agora, ocorrem casos em que se torna difícil determinar quantas vezes o bloco será executado. Sabemos que ele será executado enquanto uma condição for satisfeita (enquanto) ou até que uma condição seja satisfeita (repita).

A estrutura “para” repete a execução do bloco de um número definido de vezes, pois ela possui limites fixos.

Sintaxe:

```
para V := Vi ate Vf faca  
    comando A  
    comando B  
    comando C  
    comando D
```

fimpara

Onde V é a variável de controle; Vi é o valor inicial atribuído a V e Vf é o valor final a ser atribuído a V. O incremento de V será sempre de uma unidade.

Possuímos então um contador de forma compacta em que sempre temos uma inicialização da variável de controle V, um teste para verificar se a variável atingiu o limite Vf e um acréscimo/decrécimo na variável de controle após cada execução do bloco de repetição.

Exemplos:

Exemplo 1 em VisualG:

```
algoritmo "Exemplo_Para"  
var  
N: inteiro  
Inicio  
    para N:=0 ate 20 faca  
        se N mod 2 = 0 entao  
            escreva (N)  
        fimse  
    fimpara  
Fimalgoritmo
```

Exemplo 2 em VisualG:

```
algoritmo "Exemplo_Para_Soma"  
var  
N, SOMA: inteiro  
Inicio  
    para N:=0 ate 20 faca  
        se N mod 2 = 0 entao  
            escreva (N)  
            SOMA:= SOMA + N  
        fimse  
    fimpara  
    escreva ("A Soma dos numeros pares entre 0 e 20 eh: ", SOMA)  
fimalgoritmo
```

Prática 6:

Veja o vídeo com um exemplo do algoritmo utilizando o comando PARA no VisualG:

Visualg - Exemplo com comando PARA

<http://youtu.be/oPT-DlqnyXE>

Exercícios:

- 45) Fazer um algoritmo para calcular e mostrar a tabuada de um número qualquer informado pelo usuário.
- 46) Fazer um algoritmo para o usuário informar uma frase e o número de vezes que o programa deve repetir essa frase na tela.
- 47) Fazer um algoritmo que simule a operação de um relógio (horas, minutos e segundos).
- 48) Fazer um algoritmo que escreva na tela os números ímpares de 10 a 25.
- 49) Fazer um programa que escreva os números de 10 em 10 até 1000 e imprima uma mensagem de chegada.
- 50) O algoritmo abaixo deveria apresentar como resposta para a soma dos números pares entre 0 e 10 um resultado de 30 (0+2+4+6+8+10). Porém da forma que está desenvolvido não apresenta o desejado. Modifique o algoritmo para que tenha o resultado correto.

```
1 algoritmo "Exercicio_50"
2 var
3 N, SOMA: inteiro
4 Inicio
5     para N:=1 ate 10 faca
6         se N mod 2 = 1 entao
7             escreva (N)
8             SOMA:= SOMA + N
9         fimse
10    fimpara
11    escreva (" A Soma dos numeros pares entre 0 e 10 eh: ", SOMA)
12 fimalgoritmo
13
```



```
1 3 5 7 9 A Soma dos numeros pares entre 0 e 10 eh: 25
*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```