Lógica de Programação

- É o modo como se escreve um programa de computador.
- ALGORITMO é uma seqüência lógica de instruções que podem ser executadas.
- Um exemplo de algoritmo, fora da computação, é uma receita de bolo.

Lógica de Programação

- A linguagem de programação é como uma língua normal, um grupo de palavras com significados.
- Estas linguagens fazem o computador assimilar cada comando e função de um algoritmo, depois executar cada função.

Lógica de Programação

- Entretanto ao montar um algoritmo, precisamos primeiro dividir o problema apresentado em três fases fundamentais.

- ENTRADA: São os dados de entrada do algoritmo
- PROCESSAMENTO: São os procedimentos utilizados para chegar ao resultado final
- SAÍDA: São os dados já processados

Lógica de Programação

- Na hora de programar alguns passos são indispensáveis, como Declarar Variáveis. Elas podem ser escritas por letras ou números, que representam um valor que pode ser mudado a qualquer momento. Cada variável tem um espaço na memória para armazenar seus dados.
- Porém existem vários tipos de dados, sendo os mais comuns:
  - Numérico: todo e qualquer tipo número, positivo ou negativo.
  - Reais: podem ser positivos ou negativos e decimais.
  - Caractere: são os textos e números (mas sem valor matemático).
  - Lógico: verdadeiro ou falso.

Lógica de Programação

Algoritmo "soma"

```
Var Num1, num2, num3, resultado:inteiro
Inicio
escreval("este programa irá somar 3 números inteiros de sua escolha:")
escreval("digite um número inteiro:")
leia(num1)
escreval("digite um numero para somar ao primeiro número:")
Leia (num2)
escreval("digite um terceiro número para somar aos outros 2 números:")
Leia (num3)
Resultado <- num1+num2+num3
escreval("O resultado é: ")
escreval (resultado)
fimalgoritmo
```

Lógica de Programação

- Os operadores relacionais são utilizados para comparar String de caracteres e números. Os valores a serem comparados podem ser caracteres ou variáveis.
- Estes operadores sempre retornam valores lógicos (verdadeiro ou falso/ True ou False).
- Para estabelecer prioridades no que diz respeito a qual operação executar primeiro, utilize os parênteses.

Lógica de Programação

- **Operadores Aritméticos**

  Os operadores aritméticos são os utilizados para obter resultados numéricos. Além da adição, subtração, multiplicação e divisão, podem utilizar também o operador para exponenciação.

  Os símbolos para os operadores aritméticos são:

OPERAÇÃO	SÍMBOLO
Adição	+
Subtração	-
Multiplicação	*
Divisão	/
Exponenciação	**

Lógica de Programação

- **Hierarquia das Operações Aritméticas**
  - 1 ° ( ) Parênteses
  - 2 ° Exponenciação
  - 3° Multiplicação, divisão (o que aparecer primeiro)
  - 4 ° + ou – (o que aparecer primeiro)

Lógica de Programação

```
algoritmo "semnome"
var
   EMPRESTIMO, JUROS, PARCELA: REAL
inicio
     ESCREVA ("QUAL O VALOR DO EMPRESTIMO? ")
     LEIA (EMPRESTIMO)
     ESCREVA ("EM QUANTAS PARCELAS VOCE QUER PAGAR? ")
     LEIA (PARCELA)
     JUROS <- (EMPRESTIMO *0,2)/PARCELA
     ESCREVA ("O VALOR DAS PARCELAS SERA DE ",JUROS)
fimalgoritmo
```

Lógica de Programação

- Os operadores relacionais são:

Descrição	Símbolo
Igual a	=
Diferente de	<> ou #
Maior que	>
Menor que	<
Maior ou igual a	>=
Menor ou igual a	<=

Lógica de Programação

```
algoritmo "TRIANGULO"
var
 A, B, C: REAL
 EQ, ES, TRI: LOGICO
inicio
    ESCREVA ("DIGITE O PRIMEIRO NUMERO: ")
    LEIA (A)
    ESCREVA ("DIGITE O SEGUNDO NUMERO: ")
    LEIA (B)
    ESCREVA ("DIGITE O TERCEIRO NUMERO: ")
    LEIA (C)
    TRI <- (A<B+C) E (B<A+C) E (C<A+B)
    EQ <- (A=B) E (B=C)
    ES <- (A<>B) E (B<>C) E (A<>C)
    ESCREVAL ("PODE FORMAR UM
    TRIANGULO? ",TRI)
    ESCREVAL ("O TRIANGULO E
     EQUILATERO? ",EQ)
    ESCREVAL ("O TRIANGULO E
     ESCALENO? ",ES)

    ESCREVA (A>B)
fimalgoritmo
```

Lógica de Programação

- Os operadores lógicos servem para combinar resultados de expressões, retornando se o resultado final é verdadeiro ou falso.
- Os operadores lógicos são:

  E / AND Uma expressão AND (E) é verdadeira se todas as condições forem verdadeiras

  OR/OU Uma expressão OR (OU) é verdadeira se pelo menos uma condição for verdadeira

  NOT Um expressão NOT (NÃO) inverte o valor da expressão ou condição, se verdadeira inverte para falsa e vice-versa.

Lógica de Programação

Tabela com todas as possibilidades de combinação lógica:

1º Valor	Operador	2º Valor	Resultado
T	AND	T	T
T	AND	F	F
F	AND	T	F
F	AND	F	F
T	OR	T	T
T	OR	F	T
F	OR	T	T
F	OR	F	F
T	NOT		F
F	NOT		T

Lógica de Programação

- **Comandos de Decisão**
  - Os comandos de decisão ou desvio fazem parte das técnicas de programação que conduzem a estruturas de programas que não são totalmente seqüenciais.
  - Com as instruções de SALTO ou DESVIO pode-se fazer com que o programa proceda de uma ou outra maneira, de acordo com as decisões lógicas tomadas em função dos dados ou resultados anteriores. As principais estruturas de decisão são: **“Se Então”, “Se então Senão” e “Caso Selecione”**

Lógica de Programação

- **SE ENTÃO / IF ... THEN**

  A estrutura de decisão “SE/IF” normalmente vem acompanhada de um comando, ou seja, se determinada condição for satisfeita pelo comando SE/IF então execute determinado comando.

- **SE ENTÃO SENÃO / IF ... THEN ... ELSE**

  A estrutura de decisão “SE/ENTÃO/SENÃO”, funciona exatamente como a estrutura “SE”, com apenas uma diferença, em “SE” somente podemos executar comandos caso a condição seja verdadeira, diferente de “SE/SENÃO” pois sempre um comando será executado independente da condição, ou seja, caso a condição seja “verdadeira” o comando da condição será executado, caso contrário o comando da condição “falsa” será executado

Lógica de Programação

```
Algoritmo Comparacao
        var num1,num2: Real
        Inicio
        Ler (num1)
        Ler (num2)
        Se (num1 > num2) Então
                Escrever (‘O primeiro número é o maior’)
        Senão se (num1 < num2) Então
                Escrever (‘O segundo número é o maior’)
        Senão
                Escrever (‘Os números são iguais’)
         FimSe
FimAlgoritmo
```

Lógica de Programação

- **CASO SELECIONE / SELECT ... CASE**

  A estrutura de decisão CASO/SELECIONE é utilizada para testar, na condição, uma única expressão, que produz um resultado, ou, então, o valor de uma variável, em que está armazenado um determinado conteúdo. Compara-se, então, o resultado obtido no teste com os valores fornecidos em cada cláusula “Caso”.

Lógica de Programação

```
Algoritmo DobroOuTriplo
    Var
    OP: Inteiro
    Num, Resultado: Real;
    Início
        Escrever (‘ Opções:’)
        Escrever (‘ 1 – Calcular o dobro do número’)
        Escrever (‘ 2 – Calcular o triplo do número’)
        Escrever (‘ Escolha uma opção: ’)
        Ler (OP) Escrever (‘ Digite o número:’)
        Ler (Num)
            Caso (OP)
            1: Resultado := Num*2
            2: Resultado := Num*3
            Escrever (Resultado);
        Fim
FimAlgoritmo
```

Lógica de Programação

- **Comandos de Repetição**

Utilizamos os comandos de repetição quando desejamos que um determinado conjunto de instruções ou comandos sejam executados um número definido ou indefinido de vezes, ou enquanto um determinado estado de coisas prevalecer ou até que seja alcançado.

Lógica de Programação

- **Enquanto x, Processar (Do While ... Loop)**

  Neste caso, o bloco de operações será executado enquanto a condição x for verdadeira. O teste da condição será sempre realizado antes de qualquer operação.

  Enquanto a condição for verdadeira o processo se repete. Podemos utilizar essa estrutura para trabalharmos com contadores.

Lógica de Programação

```
Algoritmo Imprime1a100
Var Contador:Inteiro
 Início
        Contador := 1 {Contador é inicializado}
        Enquanto (Contador<=100)
                Faça Escrever(Contador)
                Contador := Contador +1 {Contador é incrementado} FimEnquanto
FimAlgoritmo
```

Lógica de Programação

- **Repita Até que x, processar ... (Do Until ... Loop)**

  Neste caso, o bloco de operações será executado até que a condição seja satisfeita, ou seja, somente executará os comandos enquanto a condição for falsa.

Lógica de Programação

```
Algoritmo Imprime1a100
Var Contador:Inteiro
    Início Contador:=1
        Repita
            Escrever(Contador)
            Contador := Contador +1
        Até (Contador >=100)
FimAlgoritmo
```

Lógica de Programação

- **Comando PARA**

  Permite construir estruturas de loop para casos onde se conhece de antemão o número de repetições que devem ser realizadas (ou seja, número finito de laços).

Lógica de Programação

```
algoritmo "FatorialComPARA"
var
  numero : INTEIRO
  fatorial : INTEIRO
  contador : INTEIRO
inicio
    ESCREVA ("Digite o número para calcular o fatorial: ")
    LEIA (numero)
    fatorial := 1
        PARA contador DE 1 ATE numero FACA
                fatorial := fatorial * contador
         FIMPARA
                ESCREVA ("O fatorial de ", numero, " é : ", fatorial)
fimalgoritmo
```