ALGORITMO

Um Algoritmo é uma seqüência de instruções ordenadas de forma lógica para a resolução de uma determinada tarefa ou problema.

ALGORITMO NÃO COMPUTACIONAL

Abaixo é apresentado um Algoritmo não computacional cujo objetivo é usar um telefone público.

1. Tirar o fone do gancho;
2. Ouvir o sinal de linha;
3. Introduzir o cartão;
4. Teclar o número desejado;
5. Se der o sinal de chamar
    - 5.1 Conversar;
    - 5.2 Desligar;
    - 5.3 Retirar o cartão;
6. Senão
    - 6.1 Repetir;

Fim.

PROGRAMA

Um programa é um Algoritmo escrito em uma linguagem computacional.

LINGUAGENS DE PROGRAMAÇÃO

São Softwares que permitem o desenvolvimento de programas. Possuem um poder de criação ilimitado, desde jogos, editores de texto, sistemas empresariais até sistemas operacionais.

Existem várias linguagens de programação, cada uma com suas características próprias.

Exemplos:

- Pascal

- Clipper
- C
- Visual Basic
- Delphi e etc.

TÉCNICAS ATUAIS DE PROGRAMAÇÃO

- Programação Seqüencial
- Programação Estruturada
- Programação Orientada a Eventos e Objetos

ALGORITMOS EM "PORTUGOL"

Durante nosso curso iremos aprender a desenvolver nossos Algoritmos em uma pseudo-linguagem conhecida como "Portugol" ou Português Estruturado.

"Portugol" é derivado da aglutinação de Português + Algol. Algol é o nome de uma linguagem de programação estruturada usada no final da década de 50.

OPERADORES ARITMÉTICOS

+ ➔ Adição

- ➔ Subtração

* ➔ Multiplicação

/ ➔ Divisão

OPERADORES RELACIONAIS

> ➔ Maior que

< ➔ Menor que

>= ➔ Maior ou Igual

<= ➔ Menor ou Igual

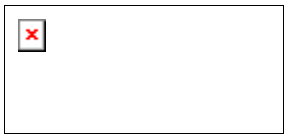
= ➔ Igual

<> ➔ Diferente

LINEARIZAÇÃO DE EXPRESSÕES

Para a construção de Algoritmos todas as expressões aritméticas devem ser linearizadas, ou seja, colocadas em linhas.

É importante também ressalvar o uso dos operadores correspondentes da aritmética tradicional para a computacional.



(2/3+(5-3))+1=

MODULARIZAÇÃO DE EXPRESSÕES

A modularização é a divisão da expressão em partes, proporcionando maior compreensão e definindo prioridades para resolução da mesma.

Como pode ser observado no exemplo anterior, em expressões computacionais usamos somente parênteses “( )” para modularização.

Na informática podemos ter parênteses dentro de parênteses.

Exemplos de prioridades:

(2+2)/2=2

2+2/2=3

OPERADORES ESPECIAIS (MOD e DIV)

MOD ➔ Retorna o resto da divisão entre 2 números inteiros.

DIV ➔ Retorna o valor inteiro que resulta da divisão entre 2 números inteiros.

```
13 DIV 2 = 6
```

```
13 MOD 2 = 1
```

13
2
6
1
MOD
DIV
Exemplo:

13 MOD 2=1

13 DIV 2=6

FUNÇÕES

Uma função é um instrumento (Sub-algoritmo) que tem como objetivo retornar um valor ou uma informação.

A chamada de uma função é feita através da citação do seu nome seguido opcionalmente de seu argumento inicial entre parênteses.

As funções podem ser predefinidas pela linguagem ou criadas pelo programador de acordo com o seu interesse.

BIBLIOTECAS DE FUNÇÕES

Armazenam um conjunto de funções que podem ser usadas pelos programas.

FUNÇÕES PRÉ-DEFINIDAS

ABS( )	VALOR ABSOLUTO
SQRT( )	RAIZ QUADRADA
SQR( )	ELEVA AO QUADRADO
TRUNC( )	VALOR TRUNCADO
ROUND( )	VALOR ARREDONDADO
LOG( )	LOGARITMO
SIN( )	SENO
COS( )	COSENO
TAN( )	TANGENTE

As funções acima são as mais comuns e importantes para nosso desenvolvimento lógico, entretanto, cada linguagem possui suas funções própias. As funções podem ser aritméticas, temporais, de texto e etc.

E	RETORNA VERDADEIRO SE AMBAS AS PARTES FOREM VERDADEIRAS.
OU	BASTA QUE UMA PARTE SEJA VERDADEIRA PARA RETORNAR VERDADEIRO.
NÃO	INVERTE O ESTADO, DE VERDADEIRO PASSA PARA FALSO E VICE-VERSA.

TABELA VERDADE

A	B	A E B	A OU B	NÃO (A)
V	V	V	V	F

V	F	F	V	F
F	V	F	V	V
F	F	F	F	V

EXPRESSÕES LÓGICAS

As expressões compostas de relações sempre retornam um valor lógico.

Exemplos:

2+5>4 ➔ Verdadeiro | 3<>3 ➔ Falso

De acordo com a necessidade, as expressões podem ser unidas pelos operadores lógicos.

Exemplo:

V
F
F

E
2+5>4 E 3<>3 ➔ Falso

VARIÁVEIS

Variáveis são endereços de memória destinados a armazenar informações temporariamente.

* Todo Algoritmo ou programa deve possuir variável!

VARIÁVEIS DE ENTRADA E SAÍDA

Variáveis de Entrada armazenam informações fornecidas por um meio externo, normalmente usuários ou discos.

Variáveis de Saída armazenam dados processados como resultados.

CONSTANTES

Constantes são endereços de memória destinados a armazenar informações fixas, inalteráveis durante a execução do programa.

Exemplo:

PI = 3.1416

IDENTIFICADORES

São os nomes dados a variáveis, constantes e programas.

Regras Para construção de Identificadores:

- Não podem ter nomes de palavras reservadas (comandos da linguagem);
- Devem possuir como 1º caractere uma letra ou Underscore (_);
- Ter como demais caracteres letras, números ou Underscore;
- Ter no máximo 127 caracteres;
- Não possuir espaços em branco;
- A escolha de letras maiúsculas ou minúsculas é indiferente.

Exemplos:

NOME	TELEFONE	IDADE_FILHO
NOTA1	SALARIO	PI
UMNOMEMUITOCOMPRIDOEDIFICILDELER UM_NOME_MUITO_COMPRIDO_E_FACIL_DE_LER		

TIPOS DE DADOS

Todas as Variáveis devem assumir um determinado tipo de informação.

O tipo de dado pode ser:

- Primitivo ➔ Pré-definido pela linguagem;
- Sub-Faixa ➔ É uma parte de um tipo já existente;
- PRIMITIVO

Escalar ➔ Definidos pelo programador.

Exemplos:

SUB - FAIXA
A : INTEIRO

TIPO NOTA=[1..10] DE INTEIRO

ESCALAR
TIPO SEMANA = (Segunda-feira, Terça-feira, Quarta-feira, Quinta-feira, Sexta-feira, Sábado, Domingo)

TIPOS PRIMITIVOS DE DADOS

INTEIRO	ADMITE SOMENTE NÚMEROS INTEIROS. GERALMENTE É UTILIZADO PARA REPRESENTAR UMA CONTAGEM (QUANTIDADE).
REAL	ADMITE NÚMEROS REAIS (COM OU SEM CASAS DECIMAIS). GERALMENTE É UTILIZADO PARA REPRESENTAR UMA MEDIÇÃO.
CARACTERE	ADMITE CARACTERES ALFANUMÉRICOS. OS NÚMEROS QUANDO DECLARADOS COMO CARACTERES TORNAM SE REPRESENTATIVOS E PERDEM A ATRIBUIÇÃO DE VALOR.
LÓGICO	ADMITE SOMENTE VALORES LÓGICOS(VERDADEIRO/FALSO).

COMANDOS DE I/O (INPUT/OUTPUT)

LER ➔ Comando de entrada que permite a leitura de Variáveis de Entrada.

ESCREVER ➔ Comando de saída que exibe uma informação na tela do monitor.

IMPRIMIR ➔ Comando de saída que envia uma informação para a impressora.

SINAL DE ATRIBUIÇÃO

Uma Variável nunca é eternamente igual a um valor, seu conteúdo pode ser alterado a qualquer momento. Portanto para atribuir valores a variáveis devemos usar o sinal de "**:=**".

Exemplos:

A **:=** 2;

B **:=** 3;

C **:=** A + B;

SINAL DE IGUALDADE

As constantes são eternamente iguais a determinados valores, portanto usamos o sinal de "=".

Exemplos:

PI = 3.1416;

Empresa = 'Colégio de Informática L.T.D.A.'

V = Verdadeiro

CORPO GERAL DE UM PROGRAMA

```
CONST
    <<identificador>> = <<dado>>
VAR
    <<identificador>> :  <<tipo>>;
ÍNICIO
    {
        COMANDOS DE ENTRADA,PROCESSAMENTO E SAÍDA
        <<comando1>>;
        <<comandoN>>
        }
FIM.
```

ESTRUTURAS SEQÜÊNCIAIS

Como pode ser analisado no tópico anterior, todo programa possui uma estrutura seqüencial determinada por um ÍNICIO e FIM.

; PONTO E VÍRGULA ;

O sinal de ponto e vírgula “;” indica a existência de um próximo comando (passa para o próximo).

Na estrutura ÍNICIO e no comando que antecede a estrutura FIM não se usa “;”.

PRIMEIRO ALGORITMO

Segue um Algoritmo que lê o nome e as 4 notas bimestrais de um aluno. Em seguida o Algoritmo calcula e escreve a média obtida.

```
PROGRAMA MEDIA_FINAL;
VAR
```

```
    NOTA1, NOTA2, NOTA3, NOTA4, MEDIA: INTEIRO;
    NOME : CARACTERE [35]
INICIO
  LER (NOME);
    LER (NOTA1, NOTA2, NOTA3, NOTA4);
  MEDIA := (NOTA1 + NOTA2 + NOTA3 + NOTA4) / 4;
  ESCREVER (NOME, MEDIA)
FIM.
```

SEGUNDO ALGORITMO

```
CONST PI = 3.1416;
VAR RAIO, AREA : REAL;
INICIO
  LER (RAIO); {PROCESSAMENTO}
  AREA := PI * SQR(RAIO); {ENTRADA}
  ESCREVER ('AREA =', AREA) {SAÍDA}
FIM.
```

{LINHAS DE COMENTÁRIO}

Podemos inserir em um Algoritmo comentários para aumentar a compreensão do mesmo, para isso basta que o texto fique entre Chaves "{}".

Exemplo:

LER (RAIO); **{ENTRADA}**

´ASPAS SIMPLES´

Quando queremos exibir uma mensagem para a tela ou impressora ela deve estar contida

entre aspas simples, caso contrário, o computador irá identificar a mensagem como Variável Indefinida.

Exemplo:

ESCREVER (‘AREA OBTIDA =’, AREA) {COMANDO DE SAÍDA}

AREA OBTIDA = X.XX {RESULTADO GERADO NA TELA}

ESTRUTURAS DE DECISÃO

Executa uma seqüência de comandos de acordo com o resultado de um teste.

A estrutura de decisão pode ser Simples ou Composta, baseada em um resultado lógico.

Simples:

SE <<CONDIÇÃO>>

ENTÃO <<COMANDO1>>

Composta 1:

SE <<CONDIÇÃO>>

ENTÃO <<COMANDO1>>

SENÃO <<COMANDO1>>

Composta 2:

SE <<CONDIÇÃO>>

ENTÃO INICIO

<<COMANDO1>>;

<<COMANDON>>

FIM;

SENÃO INICIO

<<COMANDO1>>; <<COMANDON>>

FIM;

Segue um Algoritmo que lê 2 números e escreve o maior.

```
PROGRAMA ACHA_MAIOR;

VAR A, B : INTEIRO;

INICIO

    LER (A, B);

    SE A>B

        ENTÃO ESCREVER (A)

        SENÃO ESCREVER (B)

FIM.
```

ALGORITMO QUATRO

Segue um Algoritmo que lê o nome e as 4 notas bimestrais de um aluno. Em seguida o Algoritmo calcula e escreve a média obtida pelo aluno escrevendo também se o aluno foi aprovado ou reprovado.

Média para aprovação = 6

```
 PROGRAMA MEDIA_FINAL;

VAR

        NOTA1, NOTA2, NOTA3, NOTA4, MEDIA: REAL;

        NOME : CARACTERE [35]

INICIO

    LER (NOME);

        LER (NOTA1, NOTA2, NOTA3, NOTA4);

    MEDIA := (NOTA1 + NOTA2 + NOTA3 + NOTA4) / 4;

    SE MEDIA>=6

                ENTÃO ESCREVER (‘APROVADO’)

                SENÃO ESCREVER (‘REPROVADO’)
```

ESCREVER (NOME, MEDIA)

FIM.

NINHOS DE SE

Usados para tomadas de decisões para mais de 2 opções.

Forma Geral:

SE <<CONDIÇÃO>>

ENTÃO **<<COMANDO1>>**

SENÃO SE <<CONDIÇÃO>>

ENTÃO **<<COMANDO1>>**

SENÃO **<<COMANDO1>>**

PROGRAMA ACHA_MAIOR;

VAR A, B, C : INTEIRO;

INICIO

LER (A, B, C);

SE (A>B) E (A>C)

ENTÃO ESCREVER (A)

SENÃO SE (B>A) E (B>C)

ENTÃO ESCREVER (B)

SENÃO ESCREVER (C)

FIM.

ESTRUTURAS DE CONDIÇÃO

A estrutura de condição eqüivale a um ninho de SE'S.

Forma Geral:

```
FACA CASO
    CASO <<CONDIÇÃO1>>
        <<COMANDO1>>;
    CASO <<CONDIÇÃON>>
        <<COMANDO1>>;
    OUTROS CASOS
        <<COMANDO1>>;
FIM DE CASO
```

ALGORITMO SEIS

```
PROGRAMA ACHA_MAIOR;
VAR A, B, C : INTEIRO;
INICIO
    LER (A, B, C);
    FACA CASO
                CASO (A>B) E (A>C)
            ESCREVER (A);
        CASO (B>A) E (B>C)
                        ESCREVER (B);
                        OUTROS CASOS
                                ESCREVER (C);
    FIM DE CASO
FIM.
```

ESTRUTURA DE REPETIÇÃO DETERMINADA

A estrutura de repetição, assim como a de decisão, envolve sempre a avaliação de uma condição.

Na repetição determinada o algoritmo apresenta previamente a quantidade de repetições.

Forma Geral 1:

PARA <<VARIAVEL DE TIPO INTEIRO>>:=<<VALOR INICIAL>> ATE <<VALOR FINAL>> FAÇA

    <<COMANDO1>>;

Forma Geral 2:

PARA <<VARIAVEL DE TIPO INTEIRO>>:=<<VALOR INICIAL>> ATE <<VALOR FINAL>> FAÇA

    ÍNICIO

        <<COMANDO1>>;

        <<COMANDON>>

    FIM;

A repetição por padrão determina o passo do valor inicial até o valor final como sendo 1. Determinadas linguagens possuem passo –1 ou permitem que o programador defina o passo.

ALGORITMO SETE

Segue um algoritmo que escreve 10 vezes a frase “VASCO DA GAMA”

```
PROGRAMA REPETICAO;

VARIÁVEL IMPLEMENTADA DE 1 EM 1
VAR I:INTEIRO

INICIO

    PARA I :=1 ATE 10 FACA

        ESCREVER ('VASCO DA GAMA')

FIM.
```

ALGORITMO OITO

Segue um algoritmo que escreve os 100 primeiros números pares.

```
PROGRAMA PARES;

VAR I,PAR: INTEGER;

INICIO

    PAR:=0;

        PARA I:=1 ATE 100 FACA

        INICIO

            ESCREVER (PAR);

            PAR := PAR+2

        FIM

    FIM.
```

ESTRUTURA DE REPETIÇÃO INDETERMINADA COM VALIDAÇÃO INICIAL

É usada para repetir N vezes uma ou mais instruções. Tendo como vantagem o fato de não ser necessário o conhecimento prévio do número de repetições.

VALIDAÇÃO INICIAL

Forma Geral 1:

Texto explicativo 3: VALIDAÇÃO INICIAL

```
ENQUANTO <<CONDIÇÃO>> FACA

    <<COMANDO1>>;
```

Forma Geral 2:

ENQUANTO <<CONDIÇÃO>> FACA

ÍNICIO

<<COMANDO1>>;

<<COMANDON>>

FIM;

ALGORITMO NOVE

Segue um algoritmo que calcule a soma dos salários dos funcionários de uma empresa. O programa termina quando o usuário digitar um salário menor que 0.

```
PROGRAMA SOMA_SALARIOS;

VAR SOMA, SALARIO : REAL;

INICIO

    SOMA:=O;

    SALARIO:=1;

    ENQUANTO SALARIO>=0

        INICIO

            LER (SALARIO);

            SOMA:=SOMA+SALARIO

        FIM;

    ESCREVER (SOMA)

FIM.
```

TODAS AS VARIÁVEIS QUE ACUMULAM VALORES DEVEM

RECEBER UM VALOR INICIAL.

ESTRUTURA DE REPETIÇÃO INDETERMINADA COM VALIDAÇÃO FINAL

Assim como a estrutura ENQUANTO É usada para repetir N vezes uma ou mais instruções.

Sua validação é final fazendo com que a repetição seja executada pelo menos uma vez.

Forma Geral;

```
REPITA
    <<COMANDO1>>;
    <<COMANDON>>
ATE <<CONDIÇÃO>>
```

ALGORITMO DEZ

```
VAR
        SOMA, SALARIO : REAL;
INICIO
    SOMA:=O;
    REPITA
       LER (SALARIO);
       SOMA:=SOMA+SALARIO
    ATE SALARIO<0;
    ESCREVER (SOMA)
FIM.
```

ALGORITMO ONZE

Segue um algoritmo que escreve os 100 primeiros números pares.

```
PROGRAMA PARES_2;

VAR I, PAR, CONTADOR : INTEIRO;

INICIO

    CONTADOR := 0;

        PAR := 0;

        REPITA

       ESCREVER (PAR);

       PAR := PAR+2;

       CONTADOR := CONTADOR+1;

    ATE CONTADOR=100

FIM.
```

Programas Equivalentes

O algoritmo onze poderia ter sido criado com qualquer estrutura de repetição. Portanto podemos ter algoritmos que são escritos de maneiras diferentes, mas, funcionam realizando o mesmo objetivo.