Algoritmo Computacional
Resumo Comandos/Estruturas

1. Tipos de Dados

Os dados manipulados por um algoritmo podem possuir natureza distinta, isto é, podem ser números, letras, frases, etc. Portanto os tipos de dados são:

Numérico: Podendo ser Inteiros: consiste dos números inteiros e das operações de adição, subtração, multiplicação, e Real: consiste dos números reais, na linguagem Portugol, os números reais são caracterizados por possuírem uma parte inteira e uma parte fracionária.

Literal: Inclui todas as letras do alfabeto, todos os dígitos, 0, 1, . . . , 9, símbolos, caracteres de pontuação, etc. Os elementos do conjunto de valores do tipo literal devem ser escritos, nos algoritmos, entre aspas duplas.

Lógico: Inclui apenas os valores lógicos: falso e verdadeiro.

2. Variáveis/Constantes

Variáveis e constantes, são espaços reservados na memória RAM do computador para guardar informações que serão utilizadas durante o código do programa. Podem ter valores de diversos tamanhos e tipos, tais como números inteiros, números reais, literais, enfim, diversas coisas.

Variável: O identificador é o nome pelo qual a variável é conhecida. O nome de uma variável deve iniciar sempre por uma letra, seguido de um conjunto de caracteres incluindo letras, números e alguns caracteres especiais, como é o caso do caractere '_' que pode ser utilizado para separar palavras que constituem o nome da variável (ex.: num_aluno). Por convenção, um nome de uma variável não deve conter espaços, não deve iniciar por números e não deve conter determinados caracteres especiais, tais como * , = ? , / * ; : . , } [ ] {, chama-se variável, pois o valor contido nesse espaço de memória do computador pode variar com o tempo, não é um valor fixo.

Constante: É um valor ÚNICO, um valor que NÃO mudará com o tempo de execução do programa. Pode ser declarada uma variável do tipo constante que irá reservar um espaço de memória do computador. Entretanto, esta variável do constante não pode ter seu valor alterado.

Exemplos de identificadores válidos e inválidos:

Válidos	**Inválidos**
Num	_num
valor3	3tigres
valor_x	oh!
a_3_xpto	valor-x
numero	número
maca	maçã
num_inteiro	inteiro

3. Comando DECLARE

O objetivo deste comando é para declarar as variáveis utilizadas no algoritmo computacional.

Sintaxe: **DECLARE** : Tipo de Dado;

Exemplo:

DECLARE
Nome, Cidade: Literal;
Teste, Idade: Numérico;
Sexo: Lógica;

4. Comando de Atribuição

O objetivo da atribuição é fornecer conteúdos as variáveis.

Sintaxe: **Variável** = Conteúdo;

Exemplo: **Nome** = “Maria”; **Teste** = 8.5; **Cidade** = “Salvador”; **Idade** = 19.

5. Entrada e Saída de Dados

O objetivo da entrada é permitir através do INPUT, teclado, fornecer valores as variáveis no momento em que o programa está sendo executado. E no caso da saída, é imprimir o resultado através de um dispositivo de saída, no curso adotaremos o monitor, tela do computador.

Comando de Entrada

Sintaxe:

Leia (Variável);

Exemplo:

Leia (NOME);

Leia (Cidade);

Leia (Teste);

Leia (Idade);

Comando de Saída

Sintaxe:

Escreva (Conteúdo);

Exemplo:

Escreva (“Resultado”, Media);

Escreva (“Vencedor”);

Escreva (“A média final do Aluno é”, MEDIA, “, portanto se obtive média >= 7, ele foi APROVADO”).

6. Estrutura de Seleção

Objetivo: usada para tomar decisões, ou seja, desviar a execução do algoritmo de acordo com uma condição, podendo: ser simples, composta, encadeada ou aninhada e múltipla escolha.

SIMPLES

Usada quando se pretende executar somente instruções quando a expressão for verdadeira.

Nota: Se (Expressão Lógica) for Verdade, executa instruções.

Sintaxe:

Se (Expressão Lógica) **então**
Verdade=> **Executa Instruções**;
Fim Se

Aplicação: Calcular 15% de IR para funcionários que ganham salários maiores do que 4000.

Leia (SALARIO);
Se (SALARIO > 4000) **então**
IR = SALARIO * 0.15;
Fim Se

COMPOSTA

Usada quando se pretende executar instruções quando a expressão for verdadeira e instruções para a negação da expressão verdadeira, ou seja, quando a expressão for FALSA.

Nota: Se (Expressão Lógica) for Verdade, executa instruções 1, se for FALSA, executa instruções 2.

Sintaxe:

```
Se (Expressão Lógica) então
        Verdade=> Instruções 1;
Senão
        Falsa=>     Instruções 2;
Fim Se
```

Aplicação: Dado duas notas, imprima se o aluno foi aprovado ou reprovado. Nota: Média maior ou igual a sete, aluno aprovado.

```
Leia (Nota1);
Leia (Nota2);
Media = (Nota1+Nota2) /2;
Se ( Media >=  7 ) então
        Escreva (“Aluno Aprovado”)
Senão
        Escreva (“Aluno Reprovado”);
Fim Se
```

ENCADEDA ou ANINHADA

Uma estrutura de seleção dentro da outra.

Exemplo:

Aplicação: Dado duas notas, imprima se o aluno foi aprovado ou reprovado. Nota: Média maior ou igual a sete, aluno aprovado.

```
Leia (Nota1);
Leia (Nota2);
Media = (Nota1+Nota2) /2;
Se ( Media >  7 ) então
        Escreva (“Aluno Aprovado”)
Senão
    Se ( Media =  7 ) então
        Escreva (“Aluno Aprovado”);
    Senão
        Escreva (“Aluno Reprovado”)
    Fim Se
Fim Se
```

ESCOLHA-CASO (MÚLTIPLA)

A estrutura ESCOLHA-CASO (em inglês SWITCH-CASE), é uma solução elegante quanto se tem várias estruturas de decisão (SE-ENTÃO-SENÃO) aninhadas. Isto é, quando outras verificações são feitas caso a anterior tenha falhado (ou seja, o fluxo do algoritmo entrou no bloco SENÃO). A proposta da estrutura ESCOLHA-CASO é permitir ir direto no bloco de código desejado, dependendo do valor de uma variável de verificação. Usada para situação em que requer múltipla escolha, cujas opções são inteiros os literais com um caractere, ou seja, de tamanho 1.

Sintaxe:

ESCOLHA <variável de verificação>
    CASO <valor1> **FAÇA**
        "instruções a serem executadas caso <variável de verificação> = <valor1>"
    CASO <valor2> **FAÇA**
        "instruções a serem executadas caso <variável de verificação> = <valor2>"
    CASO <valor3> **FAÇA**
    "instruções a serem executadas caso <variável de verificação> = <valor3>"
...
FIM-ESCOLHA

Aplicação:

```
DECLARE
        numero1, numero2 : NUMÉRICO;
        operacao : LITERAL;

ESCREVA ("Digite o primeiro número: "); LEIA (numero1);
ESCREVA ("Digite o segundo número: "); LEIA (numero2);
ESCREVA ("Digite a operação: "); LEIA (operacao);

ESCOLHA operacao
  CASO "+"
    resultado := numero1 + numero2;
  CASO "-"
    resultado := numero1 - numero2;
  CASO "*"
    resultado := numero1 * numero2;
  CASO "/"
    resultado := numero1 / numero2;
FIMESCOLHA

ESCREVA ("Resultado: ", resultado);
```

7. Estrutura de Repetição

Objetivo: Serve para efetuar um conjunto de ações repetidas vezes. Existem três tipos básicos de repetições, sendo elas: Para, Enquanto e Repita.

Estrutura de Repetição PARA, usada para quando se tem o controle quantas vezes o processo será executado.

Sintaxe:

Para X = Valor Inicial **Até** Valor Final **Faça**
Instruções;
Fim Para

Aplicação1: Imprima os números de 1 a 10, inclusive.

Para X = 1 **Até** 10 **Faça**
Escreva X;
Fim Para

Aplicação 2: Imprima os números de 5 a 20, inclusive.

Para Y = 5 Até 20 **Faça**
Escreva Y
Fim Para

Estrutura de Repetição ENQUANTO, usada para quando NÃO se tem o controle quantas vezes o processo será executado.

Sintaxe:

Nota: Se (Expressão Lógica) for VERDADE, executa instruções.

```
Enquanto (Expressão Lógica) Faça
      Instruções;
Fim Enquanto
```

Aplicação 1: Imprima os números de 1 a 10, inclusive.

```
X = 1;
Enquanto (X <= 10) Faça
   Escreva ( X );
   X = X + 1;
Fim Enquanto
```

Aplicação 2: Imprima os números de 1 a 10, inclusive.

```
X = 0;
Enquanto ( X < 10 ) Faça
    X = X + 1;
    Escreva ( X );
Fim Enquanto
```

Estrutura de Repetição REPITA, usada para quando NÃO se tem o controle quantas vezes o processo será executado.

A diferença desta estrutura com relação a estrutura ENQUANTO é que ela é um LOOP PÓS-TESTADO, isto é, o teste para verificar se o bloco será executado novamente, acontece no final do bloco. Isso garante que as instruções dentro deste bloco serão executadas ao menos uma vez.

Perceba, que além de ser pós-testada, esta estrutura testa o contrário do ENQUANTO. Na estrutura REPITA-ATÉ, as instruções do bloco são executadas repetidamente enquanto a expressão booleana resultar FALSO. A partir do momento que a expressão booleana resultar VERDADEIRO, o fluxo do algoritmo sairá do LOOP.

Sintaxe:

Nota: Se (Expressão Lógica) for FALSA, executa instruções.

```
Repita
      Instruções;
Até (Expressão Lógica)
```

Aplicação 1: Imprima os números de 1 a 10, inclusive.

```
X = 1;
Repita
    Escreva ( X );
    X = X + 1;
Até (X > 10); => Expressão FALSA
```