Introdução à programação e pensamento computacional

**Tipologias e variáveis**

Informações passadas ao computador são divididas em dois tipos: dados e instruções.

**Tipos primitivos de dados**

-Numérico: Números inteiros ou reais;

-Caracter: Letras (representadas entre aspas duplas), números e/ou caracteres especiais;

-Lógico (Booleano): Dentro dessa lógica só existem dois resultados possíveis: verdadeiro (representado pelo número um) ou falso (representado pelo zero). No português estruturado, equivalem a .Verdadeiro .V ou .S, .Falso, .F ou .N;

Variável: Estrutura mutável, inconstante, que pode ser reescrita, receber mais de um valor e modificar seu conteúdo. Pode assumir qualquer um dos valores de um determinado conjunto de valores. O que a restringe é, apenas, seu tipo de dados. Variáveis numéricas, por exemplo, só podem receber valores numéricos. É usada para modificar um estado (variável de ação) ou controlar uma estrutura (variável de controle).

Regras para nomear uma variável:

-Atribuir um ou mais caracteres;

-Primeiro caracter deve ser uma letra, não número;

-Sem espaços em branco;

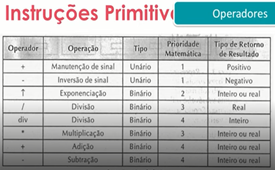
-Não pode usar palavras pré-reservadas;

-Pode incluir caracteres, números, e até mesmo dados do tipo lógico.

Constante: Algo inalterável, fixo ou estável; uma regra a ser seguida. Pode ser definida por variáveis de controle. Ex: Pi = 3,14 (e sempre será assim).

**Instruções primitivas**

Instruções definem as ações que nós iremos executar em cima dos nossos dados, e geralmente são cálculos matemáticos, porém também temos linguagem de palavra-chave. Dentro delas, nossos operadores podem ser tanto binários quanto unários.



A ordem de uma operação começa pela entrada (variáveis), em seguida estabelecemos constantes e operações (processo) para obter a saída (resultado).

Estrutura condicional: Implica uma suposição, condição, hipótese. A partir de um resultado satisfatório no qual a condição é seguida, realizamos determinada operação. Se o resultado foge à condição, temos uma exceção.

Estruturas condicionais podem ser do tipo simples (condição que resulta em operação “Se... então... fim\_se”), composta (condição que resulta em operação e uma exceção “Se... então... Senão... fim\_se”) e a encadeada (em que temos mais de uma condição com resultados de operação distintos, além da exceção “Se... então... Senão... Se... então... Senão... fim\_se”).



Operadores lógicos (And, or, not): Utilizamos para verificação de V ou F, substituindo o encadeamento de condições.

And: Todas as condições devem ser satisfeitas para que o resultado seja verdadeiro. (Interseção)

Or: Apenas uma das condições precisa ser verdadeira para que o resultado seja verdadeiro. (União)

Not: Faz a inversão do resultado lógico. Podemos utilizá-lo em operações excludentes. Ex: Not B -> Tudo aquilo que não está em B.

Estruturas de repetição: Repetem as mesmas variáveis e condições para diversas situações, porém é necessária uma condição de parada para que não haja um loop infinito. Assim, definimos um número de repetições pré-fixadas e/ou uma condição a ser satisfeita. As vantagens da estrutura de repetição são a redução de linhas, a compreensão facilitada e a redução de erro. Exemplos de estrutura de repetição: Enquanto... faça, repita... até, para... de... até... faça.

Exemplo de mescla de estruturas condicionais e de repetição:

Enquanto “grama alta” cortar

Se “grama curta” então

Parar de cortar

Fimse

....

Fim enquanto

**Vetores e matrizes**

Vetor é uma sequência de tipos de dados. É uma variável dimensionada com tamanho pré-fixado. A matriz, por sua vez, é uma tabela organizada em linhas e colunas no formato m x n, onde m = número de linhas (horizontal) e n = número de colunas (vertical). Nada mais é do que uma coleção de variáveis, de vetores. O índice indicará em qual coluna e qual linha cada variável está. Vetores aparecem entre colchetes, e o intervalo entre variáveis é representado por “..”. Ex: Vetor = [2..9]

**Funções**

São blocos de instruções que realizam tarefas específicas identificados por nomes e parâmetros (assinatura da função). Funções são compostas por definição, nome, invocação e variável local. Diminui o número de linhas do código, tornando-o mais eficiente. Ex:

Obtendo médias de alunos com função:

Resultado = 0

Resultado = (Nota 1 + Nota2)/2

Retorne resultado

Fim função

Obtendo média de alunos sem função:

aluno1 = mediaescolar(nota11,nota21)

aluno2 = mediaescolar(nota12,nota22)

aluno3 = mediaescolar(nota13,nota23)

**Instruções de entrada/saída**

Quais tipos de dados e como inseri-los no computador? Como exibir os resultados?

A entrada consiste na inserção e recebimento de dados do mundo real por meio de ação de alguma interface, ou seja, teclado, mouse, arquivos, entre outros.

A saída consiste na impressão dos dados do mundo abstrato digital por meio de ação de alguma interface. É impressa em uma tela, salva em arquivos, entre outros.

Existem dois tipos de saídas dentro de um programa (algoritmo): Programada e por interrupção.

Casos de saída na programação:

-Bem-sucedida;

-Erro de sintaxe ou outro;

-Erros de programação;

-Problemas com a interface.