{introcomp}

Working 01 : Algoritmos e Computação

Objetivos:

- Capacitar o aluno a descrever soluções, através de algoritmos, fundamentando-se na lógica e estruturas sequenciais, condicionais e de repetição;
- Conhecer e compreender o funcionamento superficial de sistemas computacionais;
- enteder como um computador consegue executar algoritmos.

Prazo de Envio: sexta, 15/10 17:00.

1 QUAIS AS ESTRUTURAS QUE PODEM FAZER PARTE DE UM ALGORITMO?

Um algoritmo pode ser constituído por três tipos de estruturas, você lembra quais? São elas:

- **Estrutura sequencial (incondicional):** corresponde a um conjunto indeterminado, porém limitados, de ações que devem ser executadas, todas passo a passo, uma após a outra;
- **Estrutura condicional (ou de seleção):** são blocos de comandos que serão executados ou não, dentro de um programa, dependendo de se uma determinada condição é verdadeira ou falsa. **Ex:** "se-entao-senao".

```
se (condicao) entao
  <conjunto de instrucoes 1>
 senao
  <conjunto de instrucoes 2>
fim-se
```

- **Estrutura de repetição:** corresponde a um conjunto de ações que deverá ser repetido inúmeras vezes até que a condição seja falsa, daí, o fluxo de execução dará continuidade ao restante das ações. **Ex:** estruturas "enquanto...faca".

```
enquanto (condicao) faca
  <conjunto de instrucoes>
fim-enquanto
```

A constituição de qualquer algoritmo baseia-se em qualquer combinação dessas três estruturas.

2 O QUE SÃO EXPRESSÕES NUMÉRICAS?

Expressões numéricas (também chamadas de expressões algébricas), já bastante conhecidas das aulas de matemática, são expressões que relacionam valores numéricos e tem como resultado um valor numérico.

Quando estamos programando, não podemos usar os mesmos símbolos para operadores numéricos que estamos acostumados na matemática. Assim, para fim de prática, iremos usar símbolos mais semelhantes aos usados em programação para nos acostumarmos. Na tabela abaixo estão os símbolos que iremos usar:

Nome	Operador
Soma	+
Subtração	-
Multiplicação	*
Divisão Real	/.
Divisão Inteira	/
Resto	%

Obs: Os operadores divisão inteira e resto só podem ser usados para números inteiros.

3 O QUE SÃO EXPRESSÕES LÓGICAS E RELACIONAIS?

Expressões lógicas são aquelas cujo resultado pode ser apenas **Verdadeiro** ou **Falso**. Estes dois valores são chamados de valores lógicos e são mais comumente falados em sua forma em inglês: True e False, respectivamente. Eles também podem ser representados como **1** (Verdadeiro) e **0** (Falso).

Como exemplos de expressão lógica poderíamos citar "Está chovendo", se no momento estiver chovendo, esta expressão assume o valor verdadeiro, caso contrário ela assume o valor falso. As expressões lógicas utilizam-se de **operadores lógicos** para a sua construção.

Uma classe de expressões lógicas são as expressões relacionais. Elas, como o próprio nome já diz, relacionam duas ou mais coisas e também possuem como resultado os valores lógicos. Por exemplo, "São Paulo é maior que Espírito Santo": esta afirmação é verdadeira e o dado São Paulo foi relacionado com o Espírito Santo com a grandeza de dimensão.

As expressões relacionais utilizam-se de operadores relacionais para a sua construção. Em algoritmos para computação, apenas os valores numéricos participam das expressões relacionais.

3.1 OPERADORES RELACIONAIS

Os operadores relacionais são os mesmos usados em inequações matemáticas, são eles:

Operador	Símbolo
Igual	==
Diferente	!=
Maior	>
Maior ou Igual	>=
Menor	<
Menor ou igual	<=

3.2 OPERADORES LÓGICOS

Os operadores lógicos são usados para relacionar duas ou mais expressões lógicas formando uma expressão maior. Por exemplo a expressão "Está chovendo e está fazendo calor" só será

verdade se ao mesmo tempo "Está chovendo" e "Está fazendo calor" forem verdadeiras. O operador e é um operador lógico, também chamado de conjunção. Temos três operadores lógicos:

- **E** → Também chamado de conjunção, a expressão só é verdade se as duas expressões forem verdadeiras, por exemplo "3>1 e 2*5==10" é verdade, mas "5>3 e 2*5<2+5" é falso. Na computação usamos o símbolo && para representá-lo;
- **OU** → Também chamado de disjunção, a expressão é verdadeira se pelo menos uma das expressões for verdadeira, por exemplo "5>3 ou 2*5<2+5" é verdadeira e "3*3=6 ou 2*5=7" é falsa. Na computação usamos o símbolo || para representá-lo;
- **NEGAÇÂO** → Transforma uma expressão verdadeira em uma falsa e vice e versa, por exemplo "não $(5 > 8)$" é verdadeira. Na computação usamos o símbolo ! para representá-lo. Para obter-se um melhor esclarecimento, pense que a expressão"não (5>8)" possui tal descrição narrativa: que o resultado da expressão 5>8 seja oinverso do resultado real. Portanto, o inverso de falso, já que 5 não é maior que 8, é verdadeiro;

A seguir temos a tabela verdade para as operações acima citadas:

Tabela 1: Tabela Verdade

a	b	a&&b	a||b	!a	!b
F	F	F	F	V	V
F	V	F	V	V	F
V	F	F	V	F	V
V	V	V	V	F	F

4 VARIÁVEIS

Como implementar um algoritmo computável, ou seja, que computa valores, sem a capacidade de manipular estes dados quantitativos!? Como obter operações lógicas e númericas sem a manipulação de valores? Como efetuar a execução de algoritmos sem a possibilidade de armazenar dados e, no espaço temporal, poder manipulá-lo até obter a saída desejada? Sempre que estivermos implementando algoritmos, teremos que manipular diversos dados. Em um sistema de computação, esses dados são memorizados/guardados e manipulados. Desta forma, definiu-se o conceito de variáveis.
Variáveis podem ser entendidas como "caixas" que podem guardar um valor de cada vez. Para trabalharmos com variáveis, damos um nome para ela (único para cada variável). Sempre que tivermos uma expressão com uma variável, iremos calcular essa expressão com o valor atual da variável.
Para atualizarmos o valor de uma variável usamos o comando de atribuição. A atribuição funciona do seguinte modo:

```
1 Variavel <- Expressao
```

O resultado da expressão que está a direita do operador de atribuição (<-) será guardada na variável que está à esquerda.

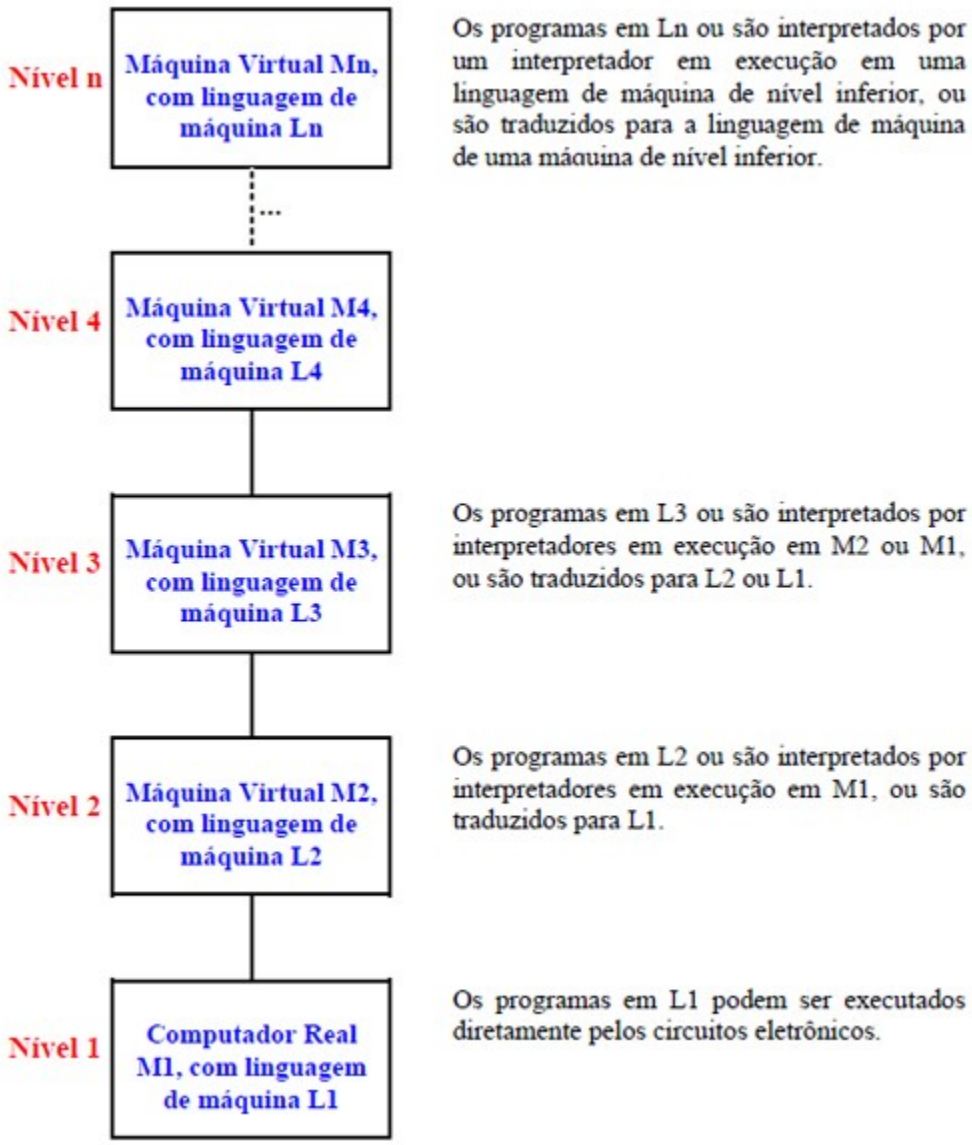
5 LINGUAGENS DE PROGRAMAÇÃO: O CONCEITO DE "MÁQUINA MULTINÍVEL"

Um problema clássico na organização estruturada de computadores se baseia na grande lacuna do que é conveniente para os computadores e o que é conveniente para as pessoas. Pessoas querem fazer X, mas os computadores só podem fazer Y.

Um computador estritamente recebe "ordens", processa essas "ordens" e gera uma informação. Tais "ordens" são fornecidas como conjuntos de instruções logicamente ordenadas (algoritmo), de forma padronizada. Uma **linguagem de programação** é justamente uma forma padronizada de comunicação com o computador.

A maioria dos sistemas computacionais atuais operam utilizando a lógica digital binária. É a "linguagem" com que computadores interpretam instruções e se comunicam. Embora os primeiros computadores fossem programados em linguagem de máquina, a complexidade crescente dos programas obrigou a encontrar outras formas de se comunicar com a máquina.

O conceito de **máquina multinível** surgiu para resolver o problema, permitindo entender o computador em níveis bem definidos que teriam a função de abstrair do programador a complexidade eletrônica e digital envolvida no tratamento de instruções pela máquina.



A ideia fundamental do modelo é imaginar que o computador seja formado por um conjunto de "máquinas virtuais", que devem ser entendidas como computadores hipotéticos que estão preparados para receber uma determinada linguagem, tendo esta linguagem como sua linguagem de máquina.
O modelo tem a função de garantir ao programador uma abstração de níveis mais baixos, que essencialmente são mais complexos. Deste modo, poderiam existir agora linguagens que operassem sobre "máquinas virtuais" e que fossem muito mais humanas que a linguagem utilizada pela máquina real.
O **nível** de uma linguagem de programação é definido pela distância que a máquina virtual na qual esteja associada está da máquina real. Uma linguagem de programação pode ser de **alto nível** ou **baixo nível**.

5.1 Linguagens de baixo nível

As linguagens de baixo nível são conhecidas pela forte relação com o hardware e compreendem bem as características da arquitetura de um computador. São linguagens voltadas para a máquina e são escritas utilizando as instruções do microprocessador do computador. A linguagem de máquina e a linguagem de montagem (Assembly) são exemplos de linguagens de baixo nível.
- Linguagem de máquina: Linguagem composta somente por números, representados por cadeia de bits, que sob o ponto de vista do computador representam as operações e os operandos que serão usados no processamento das intruções.
- Linguagem de montagem (Assembly): Linguagem que surgiu para tornar mais legível a linguagem de máquina para os seres humanos, substituindo as sequências de bit's bruta em símbolos que tivessem significados equivalentes. Nesta linguagem, não só responsável de descrever um algoritmo, você também se preocupa com questões de movimento de dados da memória para o processador e vice-versa.
Por exemplo, enquanto um computador sabe o que a instrução em linguagem de máquina IA-21 (10110000 01100001) faz, para os programadores é mais fácil recordar a representação equivalente em instruções simbólicas "MOV AL, 10". Essa instrução, por exemplo, estaria ordenando que o valor 10 estivesse sendo movido para o registrador 'AL'.

5.2 Linguagens de alto nível

São linguagens que são mais próximas à linguagem natural do ser humano, tendo a característica de possuir um nível de abstração elevado, longe do código de máquina. Ao contrário da linguagem de baixo nível, não estão diretamente relacionadas à arquitetura do computador e não precisam entender detalhes de gerenciamento de memória e processamento. Nelas, você foca apenas na resolução de problemas, utilizando-se dos mais diversos algoritmos para construir programas. Conhecidas pela sua facilidade de manipulação, as linguagens de programação se tornaram muito mais acessíveis à comunidade e permitiram que problemas pudessem ser solucionados de forma rápida. Alguns exemplos de linguagens de alto nível:

- C/C++
- Java
- Python
- Haskell
- Lua
- C#
- Ruby
- Perl
- Perl
- PHP

6 Mas como uma linguagem de alto nível pode ser entendida pela máquina real?

É preciso lembrar que embora existam inúmeras linguagens de programação de alto nível, o computador continua a entender e ser capaz de executar somente programas em linguagens de baixo nível, a linguagem de máquina.

Como uma linguagem de alto nível, tão próxima dos seres humanos, poderia ser implementada em um computador que só pode "enxergar" uma linguagem de baixo nível?

Existem basicamente duas alternativas para esta implementação: **Interpretação** e **tradução**.

6.1 Interpretação

O programa fonte é traduzido e executado instrução a instrução, de forma iterativa. O interpretador cuida de traduzir todas as instruções para uma representação interna e as interpreta simulando o funcionamento do processador. Funciona de forma bem definida, executando repetidamente a seguinte sequência:

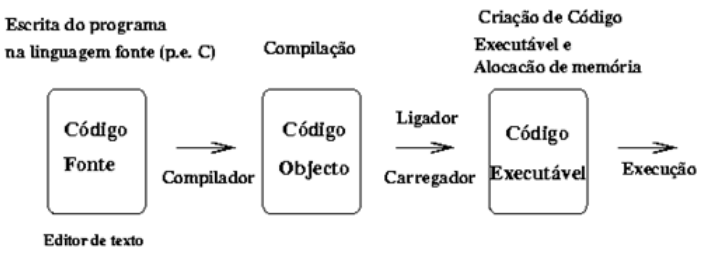
1. Obter um comando do programa;
2. Converter o comando em instruções a nível de linguagem de máquina, que poderão ser entendidas pela máquina real;
3. As instruções são executadas.

6.2 Tradução

Um tradutor é uma ferramenta de programação que possibilita traduzir um programa de uma determinada linguagem para uma outra linguagem. Os programas escritos em linguagem de alto nível são traduzidos para versões equivalentes em linguagens de máquina, mas diferentemente da interpretação, antes de serem executados. Um **compilador** é um exemplo de uma ferramenta de tradução.

6.3 Compilador

O compilador é uma ferramenta que traduz uma linguagem de alto nível (chamada de "código fonte") em uma linguagem de baixo nível (chamada de "código objeto") na forma de um executável.



A linguagem C, que iremos abordar no nosso curso, é um exemplo de uma linguagem que necessita ser compilada.
Exemplos de compiladores podem ser listados abaixo:

- GCC
- Free Pascal
- Visual C
- Delphi
- Visual Basic
- G++

Praticando

Agora vamos praticar! Para todos os praticandos deste Working escreva a resposta em um arquivo (.txt) para envio no site.

1. Desenvolva algoritmos em pseudocódigo que:
   (a) Verifique se um dado número é negativo;
   (b) Verifique se um dado número é par ou ímpar;
   (c) Crie três variáveis, leia seus valores do teclado, armazene o maior entre eles e imprima na tela;
2. Desenvolva algoritmos em pseudocódigo que:
   (a) Incremente o valor de uma variável de 1 até 10;
   (b) Some 10 números apresentados dados como entrada;

Desafios

1. Escreva um algoritmo que dados o peso em kg e a altura em metros de uma pessoa, classifique-a quanto ao seu IMC (Índice de Massa Corporal) segundo a tabela abaixo: **Obs:** IMC = massa/(altura*altura)

IMC	Classificação
<18,5	Abaixo do Peso
18,6 - 24,9	Saudável
25,0 - 29,9	Peso em Excesso
30,0 - 34,9	Obesidade Grau I
35,0 - 39,9	Obesidade Grau II
>= 40,0	Obesidade Grau III

NO PROXIMO ENCONTRO...

Como surgiu a linguagem de programação C? Qual sua sintaxe básica? Como construir meu primeiro programa em C? Cenas do próximo capítulo!