Universidade Federal do Espírito Santo
Centro Tecnológico

{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, 05/08 04:00.

PET
Engenharia de Computação

1 INTRODUÇÃO

Bem-vindo ao primeiro Working do curso! Agora é a hora de você realmente se desenvolver, aprimorar sua lógica e praticar, construir suas habilidades.

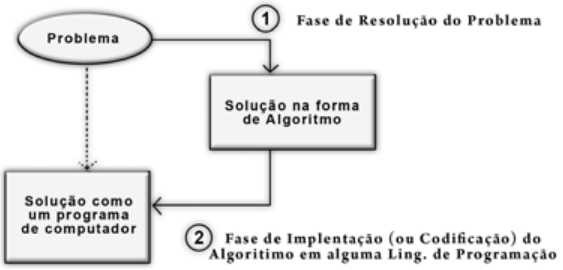
Este Working é muito importante, pois, afinal, de acordo com David Harel, algoritmos é o espírito/alma da computação. Antes de você estar aqui visualizando este pdf por meio do seu computador eletrônico, o qual executa milhares e milhares de algoritmos, dezenas de estudiosos a um bom tempo atrás estavam pesquisando e formulando a lógica, a automação do raciocínio e definindo o que é ou não é computável, definindo detalhadamente o que é um algoritmo. Enfim, há milhares de anos, a computação foi executada com caneta e papel, ou com giz e ardósia, ou mentalmente, por vezes com auxílio de tabelas ou utensílios artesanais.

Você já parou pra pensar que existem problemas que não há solução de maneira algorítmica, ou seja, não há uma sequência de passos que permita resolver estes problemas? Você também acredita que esses estudiosos nunca tiveram a oportunidade de tocar nem que seja no mais simples computador existente no atual século, mas conseguiram definir os alicerces pra atual e moderna computação? Aristóteles e Euclides nos primórdios formulando a lógica, Kurt Gödel, Alan Turing, o pai da computação, e muitos outros permitiram a revolução da computação no mundo. Em qualquer canto do globo, qualquer área, qualquer mercado, a computação está presente para resolver problemas, automatizar processos e satisfazer necessidades.

Neste Working trataremos de Algoritmos e Computação. Vamos começar.

2 QUAL O RELACIONAMENTO ENTRE RESOLUÇÃO DE PROBLEMAS, LÓGICA E ALGORITMOS?

A construção de programas de computador exige, entre outras habilidades, a capacidade de resolver problemas através da identificação de um conjunto ordenado e finito de etapas e/ou instruções que levam a sua resolução. Um exemplo disso seria a lista de instruções definidas para analisar as suas notas na disciplina e informar se você foi aprovado ou não. Também é esperado que você saiba que este conjunto de passos é a representação de um algoritmo e forma a base de um programa de computador. **Daí, falando em algoritmos estamos falando, também, em resolução de problemas, concorda?** É como apresentado na figura a seguir: primeiro é apresentado o problema, em seguida desenvolve-se uma solução na forma de algoritmo, e o passo seguinte é passar essa solução para uma linguagem de computador.



A lógica de programação, de acordo com Esmin (2000), consiste em aprender a pensar na mesma sequência de execução dos programas de computador. Aprende-se, dessa forma, a pensar como serão executadas as ações, partindo do estudo de um problema até chegar à construção de um algoritmo, que seria a solução deste mesmo problema.

3 MAS, O QUE É UM ALGORITMO MESMO?

"Algoritmo é a descrição de uma sequência de passos que deve ser seguida para a realização de uma tarefa" (Ascencio, 1999).

É fácil perceber que um algoritmo não representa, necessariamente, um programa de computador, e sim os passos necessários para realizar uma tarefa ou solucionar um problema. Corriqueiramente, executamos várias atividades que podem ser representadas através de algoritmos, por exemplo, algoritmo para escovar os dentes, tomar banho, vestir-se, ir ao trabalho, lavar o carro, etc.

4 QUAIS AS CARACTERÍSTICAS DE UM ALGORITMO?

Todo algoritmo deve apresentar, no mínimo, as características apresentadas a seguir.

- **Finitude:** um algoritmo deve sempre terminar após um número finito de passos.
- **Definição:** cada passo/instrução/etapa de um algoritmo deve ser precisamente definido. As ações devem ser definidas rigorosamente e sem ambiguidades.
- **Entradas:** um algoritmo deve ter zero ou mais entradas, isto é, quantidades que lhe são fornecidas antes do algoritmo iniciar. Por exemplo, em um algoritmo para computar a média de um aluno, a entrada seria as notas desse aluno.
- **Saídas:** um algoritmo deve ter uma ou mais saídas, isto é quantidades que tem uma relação específica com as entradas. No algoritmo para computar a média do aluno, a saída seria a média.

Adicionalmente, um algoritmo deve sempre ser **especificado para resolver um problema**, ou seja, deve ter um objetivo, uma finalidade. Essas características são fundamentais quando construindo algoritmos, você precisa conhecê-las bem.

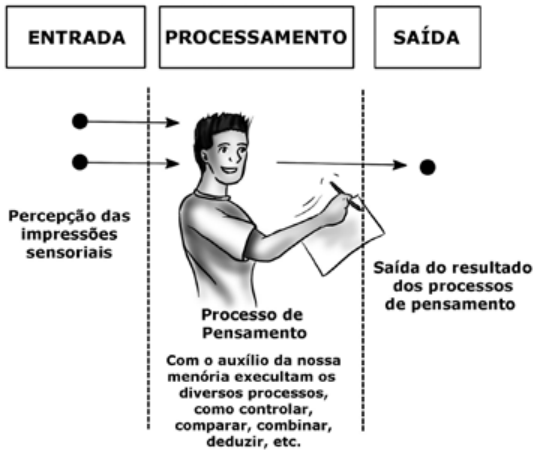
Além dessas características, você também precisa entender uma outra: **eficiência**. Um algoritmo deve ser eficiente. Isto significa que todas as operações devem ser suficientemente básicas de modo que possam ser executadas com precisão em um tempo finito. Além disso, deve-se levar em consideração o consumo de memória. Um bom programador é aquele que, para um determinado problema, sabe qual é o algoritmo mais eficiente e consegue aplicá-lo.

5 QUAIS AS FASES DE CONSTRUÇÃO DE UM ALGORITMO?

Diversas são as técnicas e métodos existentes para a construção de algoritmos, no entanto, todas elas possuem um mesmo objetivo: solucionar o problema. Para tornar essa resolução mais fácil, ao construir um algoritmo, precisamos primeiro dividir o problema apresentado em três fases fundamentais:

- **Entrada:** São os dados de entrada do algoritmo. As informações que ele vai precisar para poder solucionar o problema;
- **Processamento:** São os procedimentos utilizados para chegar ao resultado final, tais como cálculos, conversões, operações, etc;
- **Saída:** São os dados já processados que, geralmente, serão apresentados aos usuários.

Isso é parecido com a forma como o nosso cérebro funciona (vide Figura a seguir). Para realizar cada ação do dia-a-dia, para solucionar um problema ou executar uma atividade, primeiro, recebemos como **entrada** a percepção das impressões sensoriais (tais omo cheiros, sons, gostos, sensaçõs, etc). Depois, a partir dessa entrada, fazemos o **processamento** das informações, unindo as mesmas com outras informações que estão na nossa memória. A partir disso, temos a **saída** do processamento que resulta em uma ação, seja falar alguma coisa, realizar alguma atividade, produzir uma nova informação para ser armazenada na nossa memória, entre outras coisas.



A seguir um método simples, possivelmente já conhecido por você, para a construção de algoritmos, que possibilita a implementação dos mesmos, posteriormente, em um computador.

1. Procure conhecer e compreender, ao máximo, o problema a ser resolvido;
2. Identifique e defina os dados que, essencialmente, deverão ser informados para que o processamento seja realizado com sucesso (dados de entrada);
3. Descreva, detalhadamente, o processamento ou a transformação a ser executada sobre os dados de entrada, em busca dos resultados desejados (como chegar no objetivo);
4. Identifique e defina quais serão os dados resultantes do processamento ou transformação (dados de saída);
5. Construa o algoritmo que represente a solução encontrada com o detalhamento necessário para o seu entendimento;
6. Teste o algoritmo por meio de simulações e efetue as devidas correções que possam vir a ser necessárias na lógica proposta.

6 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.

7 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.

8 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.

8.1 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	<=

8.2 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

9 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.

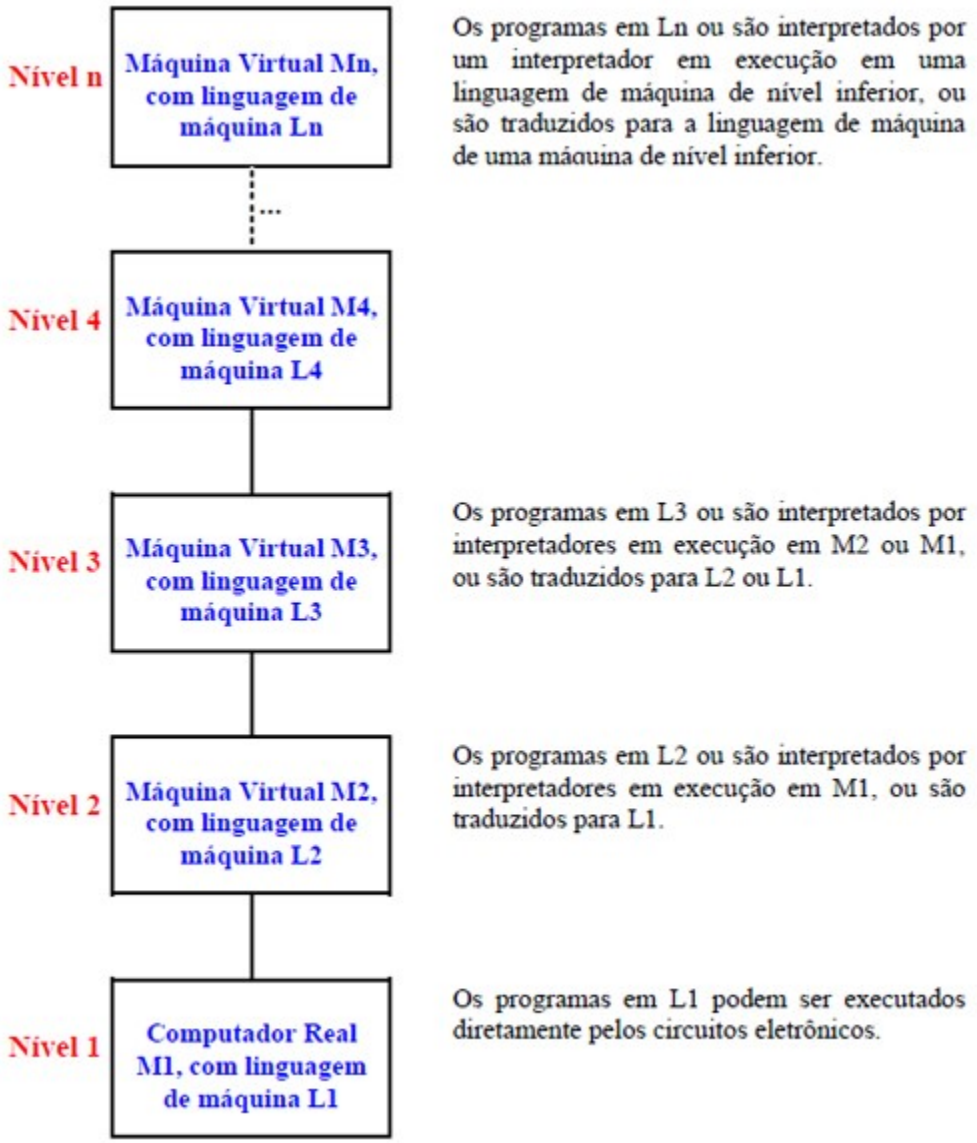
10 LINGUAGENS DE PROGRAMAÇÃO: O CONCEITO DE "MÁQUINA MULTINÍVEL"

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**.

10.1 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'.

10.2 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

11 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**.

11.1 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:

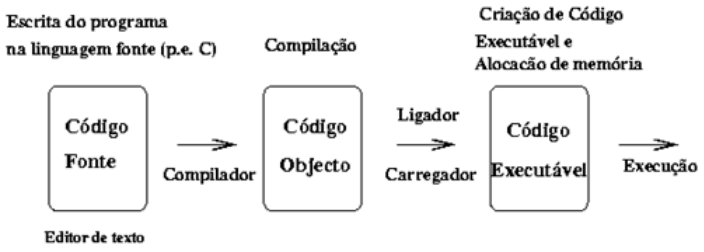
1. Obter um comando do programa;
2. Converter o comando em instruções a nível de linguagem de máquina, que poderão ser entendidas pela máquina real;
3. As instruções são executadas.

11.2 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.

11.3 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. Qual será o valor de cada variável ao final da execução da sequência de comandos?

   (a)
   ```
   x <- 3+3
   y <- x*2*3
   z <- x+y
   ```

   (b)
   ```
   a <- 10
   b <- 2*a+5
   c <- (a+2)*b
   ```

   (c)
   ```
   x <- 6
   y <- 2*x
   y <- 3*x*y
   ```

   (d)
   ```
   a <- 3*2
   b <- a*2+4
   a <- a*b*5
   ```

2. Escreva um algoritmo em pseudocódigo que execute os seguintes comandos:

   I. Faça com que uma variável x receba o valor 35;
   II. Faça com que outra variável y receba o valor 25;
   III. Some as variáveis x e y, e armazene o resultado em uma variável w;
   IV. Multiplique as variáveis x e y, e armazene o resultado em uma variável z;
   V. Faça a média aritmética das variáveis x, y, w, z, e armazene em uma variável soma.

3. Desenvolva algoritmos em pseudocódigo que:

   (a) crie três variáveis, some seus valores, e imprima na tela.
   (b) crie três variáveis, multiplique o valor da primeira por 2, some esse resultado com as duas outras variáveis, e imprima o resultado desta última operação.
   (c) verifique se um dado número é negativo.
   (d) verifique se um dado número é par ou ímpar.
   (e) incremente o valor de uma variável de 1 até 10.
   (f) 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!