PROGRAMAÇÃO A

Conceitos Básicos

INTRODUÇÃO

Desde o início de sua existência, o homem procurou criar máquinas que o auxiliassem em seu trabalho, diminuindo o esforço e economizando tempo. Dentre essas máquinas, o computador vem se mostrando uma das mais versáteis, rápidas e seguras.

A finalidade de um computador é receber, manipular e armazenar dados.

O computador possui duas partes diferentes que trabalham juntas: o **hardware**, composto pelas partes físicas, e o **software**, composto pelos programas.

Uma **linguagem de programação** é uma linguagem que tanto o computador quanto o criador de software entendem.

As etapas para o desenvolvimento de um programa são:

- Análise
- Algoritmo
- Codificação

NOÇÕES DE LÓGICA

Lógica é a ciência que estuda as formas de pensamento.

A **Lógica** nos acompanha diariamente:

- Um bebê sabe que precisa chorar para receber atenção.
- Um casal com 3 filhos notou que um vaso estava quebrado, enquanto 2 das crianças estavam na escola. Quem é o culpado?
- Pegar um chiclete. Retirar o papel. Mastigar o chiclete. Jogar o papel no lixo.
- Se um carro está com a seta esquerda ligada. Significa que ele vai virar à direita ou à esquerda?

O pensamento (e a lógica) pode ser expresso por meio da linguagem oral ou escrita.

Um mesmo pensamento pode ser expresso em inúmeros idiomas, tanto oralmente quanto por escrito.

NOÇÕES DE LÓGICA

Existe lógica no nosso dia-a-dia?

Sempre que pensamos, a lógica ou a ilógica necessariamente nos acompanha. Quando falamos ou escrevemos, estamos expressando nosso pensamento, logo, precisamos usar a lógica nessas atividades.

Exemplos:

a. Todo mamífero é um animal.

Todo cavalo é um mamífero.

Portanto, todo cavalo é um animal.

b. Anacleto é mais velho que Felisberto.

Felisberto é mais velho que Marivaldo.

Portanto, Anacleto é mais velho que Marivaldo.

NOÇÕES DE LÓGICA - EXERCITANDO...

Vamos a um simples problema de lógica para aquecer!

Três senhoras - dona Branca, dona Rosa e dona Violeta – passeavam pelo parque quando dona Rosa disse:

- Não é curioso que estejamos usando vestidos de cores branca, rosa e violeta, embora nenhuma de nós esteja usando um vestido de cor igual ao seu próprio nome?

- Uma simples coincidência – respondeu a senhora com o vestido violeta.

Qual a cor do vestido de cada senhora?

ALGORITMOS - CONCEITOS

"**Algoritmo** é uma sequência de passos que visa atingir um objetivo bem definido."

"**Algoritmo** é a descrição de uma sequência de passos que deve ser seguida para a realização de uma tarefa."

"**Algoritmo** é uma sequência finita de instruções ou operações cuja execução, em tempo finito, resolve um problema computacional, qualquer que seja sua instância."

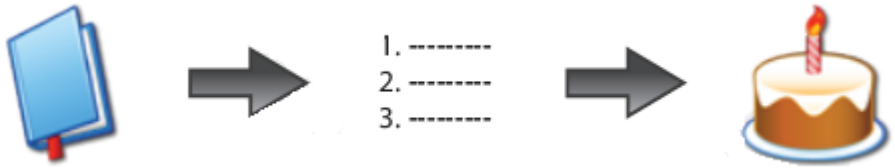
"**Algoritmos** são regras formais para a obtenção de um resultado ou da solução de um problema, englobando fórmulas de expressões aritméticas."

Algoritmos no nosso dia-a-dia: Receita de bolo, orientação para se chegar em algum endereço, realização de alguma tarefa rotineira.

Para exemplos de algoritmos abra o arquivo Exemplos de Algoritmos - Descrição Narrativa.pdf disponível no ambiente Moodle.

ALGORITMOS - O QUE É?

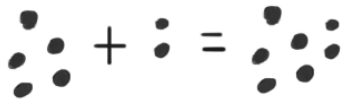
- Um algoritmo é uma sequência de instruções que resolve uma determinada tarefa. Essas instruções podem ser executadas por um computador ou até mesmo por um ser humano. Um algoritmo pode ser comparado a uma receita de bolo, onde cada passo da preparação do bolo corresponde a uma instrução do algoritmo.



- Normalmente, desenvolver algoritmos eficientes não é uma tarefa simples. No meio acadêmico, diversas técnicas para o desenvolvimento de algoritmos mais eficientes são estudadas pela Ciência da Computação.

ALGORITMOS - EXEMPLOS

Algoritmo 1 - Somar três números



- Passo 1 - Receber os três números.
- Passo 2 - Somar os três números.
- Passo 3 - Mostrar o resultado obtido.

Algoritmo 2 - Fazer um sanduíche

- Passo 1 - Pegar o pão.
- Passo 2 - Cortar o pão ao meio.
- Passo 3 - Pegar a maionese.
- Passo 4 - Passar a maionese no pão.
- Passo 5 - Pegar e cortar a alface e o tomate.
- Passo 6 - Colocar a alface e o tomate no pão.
- Passo 7 - Pegar o hambúrguer.
- Passo 8 - Fritar o hambúrguer.
- Passo 9 - Colocar o hambúrguer no pão.

ALGORITMOS - EXEMPLOS

Algoritmo 3 - Ir para a escola

- Passo 1 - Acordar cedo.
- Passo 2 - Ir ao banheiro.
- Passo 3 - Abrir o armário para escolher uma roupa.
- Passo 4 - Se o tempo estiver quente, pegar uma camiseta e uma calça jeans;
  Caso contrário, pegar um agasalho e uma calça jeans.
- Passo 5 - Vestir a roupa escolhida.
- Passo 6 - Tomar café.
- Passo 7 - Pegar uma condução.
- Passo 8 - Descer próximo à escola.

ALGORITMOS - EXEMPLOS

Algoritmo 4 - Sacar dinheiro no banco 24 horas

- Passo 1 - Ir até um banco 24 horas.
- Passo 2 - Colocar o cartão.
- Passo 3 - Digitar a senha.
- Passo 4 - Solicitar a quantia desejada.
- Passo 5 - Se o saldo for maior ou igual à quantia desejada, sacar;
  Caso contrário, mostrar mensagem de impossibilidade de saque.
- Passo 6 - Retirar o cartão.
- Passo 7 - Sair do banco 24 horas.

ALGORITMOS - EXEMPLOS

Algoritmo 5 - Trocar uma lâmpada

- Passo 1 - Acionar o interruptor.

  Se a lâmpada não acender, então:

  - Passo 2 - Pegar uma escada.
  - Passo 3 - Posicionar a escada embaixo da lâmpada.
  - Passo 4 - Buscar uma lâmpada nova.
  - Passo 5 - Subir na escada.
  - Passo 6 - Retirar a lâmpada queimada.
  - Passo 7 - Colocar a lâmpada nova.
  - Passo 8 - Descer da escada
  - Passo 9 - Acionar o interruptor.

    Enquanto a lâmpada não acender faça

    - Passo 10 - Retirar a lâmpada queimada
    - Passo 11 - Colocar a lâmpada nova.
    - Passo 12 - Acionar o interruptor.

ALGORITMOS - TORRES DE HANÓI 

As **Torres de Hanói** é um quebra-cabeça composto por uma base contendo três torres (A, B e C) e três discos de diâmetros distintos (1, 2 e 3). Neste quebra-cabeça, o objetivo é encontrar uma forma de mover todos os discos da torre A para a torre C, usando a torre B como espaço auxiliar, de modo que:

- apenas um disco seja movido de cada vez;
- nenhum disco seja posicionado sobre outro disco de diâmetro menor;
- os discos sejam imediatamente transferidos de uma torre para outra.

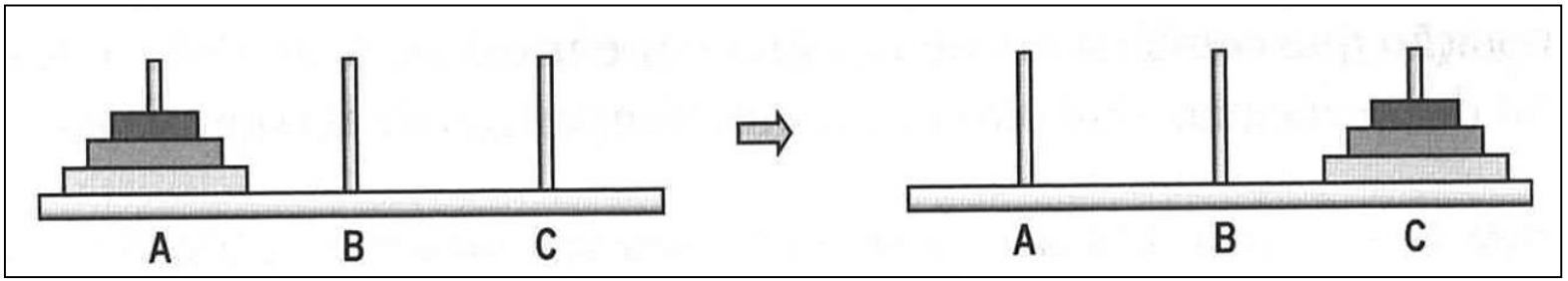


Figura 1 - As Torres de Hanói.

ALGORITMOS - TORRES DE HANÓI (SOLUÇÃO)

Há várias soluções possíveis para este problema. Uma delas é a seguinte:

1° passo: mova o disco do topo da torre A para o topo da torre C.

2° passo: mova o disco do topo da torre A para o topo da torre B.

3° passo: mova o disco do topo da torre C para o topo da torre B.

4° passo: mova o disco do topo da torre A para o topo da torre C.

5° passo: mova o disco do topo da torre B para o topo da torre A.

6° passo: mova o disco do topo da torre B para o topo da torre C.

7° passo: mova o disco do topo da torre A para o topo da torre C.

Uma vez definida esta sequencia de passos, qualquer pessoa capaz de executá-la é também capaz de solucionar automaticamente o problema. Nenhum raciocínio adicional é necessário, pois a sequencia já descreve um procedimento correto para resolver o problema. Basta que a pessoa execute estes passos na ordem indicada.

Um procedimento para um problema, definido por uma sequencia finita e ordenada de passos executáveis, é denominado **algoritmo**.

MÉTODO PARA A CONSTRUÇÃO DE ALGORITMOS

Para a construção de qualquer tipo de algoritmo, é necessário seguir estes passos:

- Compreender completamente o problema a ser resolvido, destacando os pontos mais importantes e os objetos que o compõem.
- Definir os dados de entrada, ou seja, quais dados serão fornecidos e quais objetos fazem parte desse cenário problema.
- Definir o processamento, ou seja, quais cálculos serão efetuados e quais as restrições para esses cálculos. O processamento é responsável pela transformação dos dados de entrada em dados de saída. Além disso, deve-se verificar quais objetos são responsáveis pelas atividades.
- Definir os dados de saída, ou seja, quais dados serão gerados depois do processamento.
- Construir o algoritmo utilizando um dos tipos descritos nos próximos slides.
- Testar o algoritmo realizando simulações.

ALGORITMOS - CONCLUSÃO

Algoritmos são muito comuns no nosso dia a dia. Alguns exemplos são:

- O manual de instalação de um aparelho de DVD, que descreve passo a passo como devemos proceder para conectar esse aparelho a um televisor.
- Uma receita culinária, que descreve os passos para preparação de um prato.
- Um mapa, que descreve como proceder para chegar a uma localização.
- Uma escala de avaliação clínica da saúde de uma pessoa.

No contexto da **Computação**, é interessante que algoritmos sejam executados por computadores e não por pessoas. Ao definir um algoritmo computacional, é preciso restringir o mesmo a um conjunto bastante limitado de passos (ou operações) que um computador é capaz de executar. Também é necessário uma notação que permita descrever precisamente estes passos, sem nenhuma ambiguidade.

ALGORITMOS - EXERCÍCIOS

1. Elabore um algoritmo que descreva os passos para elaborar uma receita culinária que você saiba fazer sem precisar consultar as instruções de um livro ou site.
2. Descreva os passos que você executa diariamente para chegar até a Universidade.
3. Elabore um algoritmo que descreva os passos necessários para que uma pessoa consiga obter a sua primeira habilitação.
4. Elabore um algoritmo que descreva os passos necessários para que uma pessoa consiga tirar seu título de eleitor.

TIPOS DE ALGORITMOS

Nós podemos representar um algoritmo da maneira que acharmos melhor, desde que tal representação seja bem estruturada e organizada. Porém, as representações mais utilizadas são a da Descrição Narrativa, Fluxograma e de Pseudocódigo ou Portugol.

Descrição narrativa

A descrição narrativa consiste em analisar o enunciado do problema e escrever, utilizando uma linguagem natural (por exemplo, a língua portuguesa), os passos a serem seguidos para sua resolução.

Vantagem: não é necessário aprender nenhum conceito novo, pois uma língua natural, neste ponto, já é bem conhecida.

Desvantagem: a língua natural abre espaço para várias interpretações, o que posteriormente dificultará a transcrição desse algoritmo para programa.

DESCRIÇÃO NARRATIVA

A frase **"O pregador foi grampeado durante o conserto"** possui até *oito sentidos diferentes*!

Quantos sentidos você consegue tirar desta frase???

Para evitar esse e outros problemas, um conjunto de regras é utilizado para restringir e estruturar o uso do português na representação de algoritmos e que, intencionalmente, se aproximam da maneira pela qual o fazem linguagens de programação reais como C, como a finalidade de facilitar a futura codificação dos algoritmos.

FLUXOGRAMA

O fluxograma é um dos métodos mais utilizados para se representar um algoritmo. Trata-se de uma espécie de diagrama e é utilizado para documentar processos (simples ou complexos). Tal tipo de diagrama ajuda o leitor a visualizar um processo, compreendê-lo mais facilmente e encontrar falhas ou problemas de eficiência.

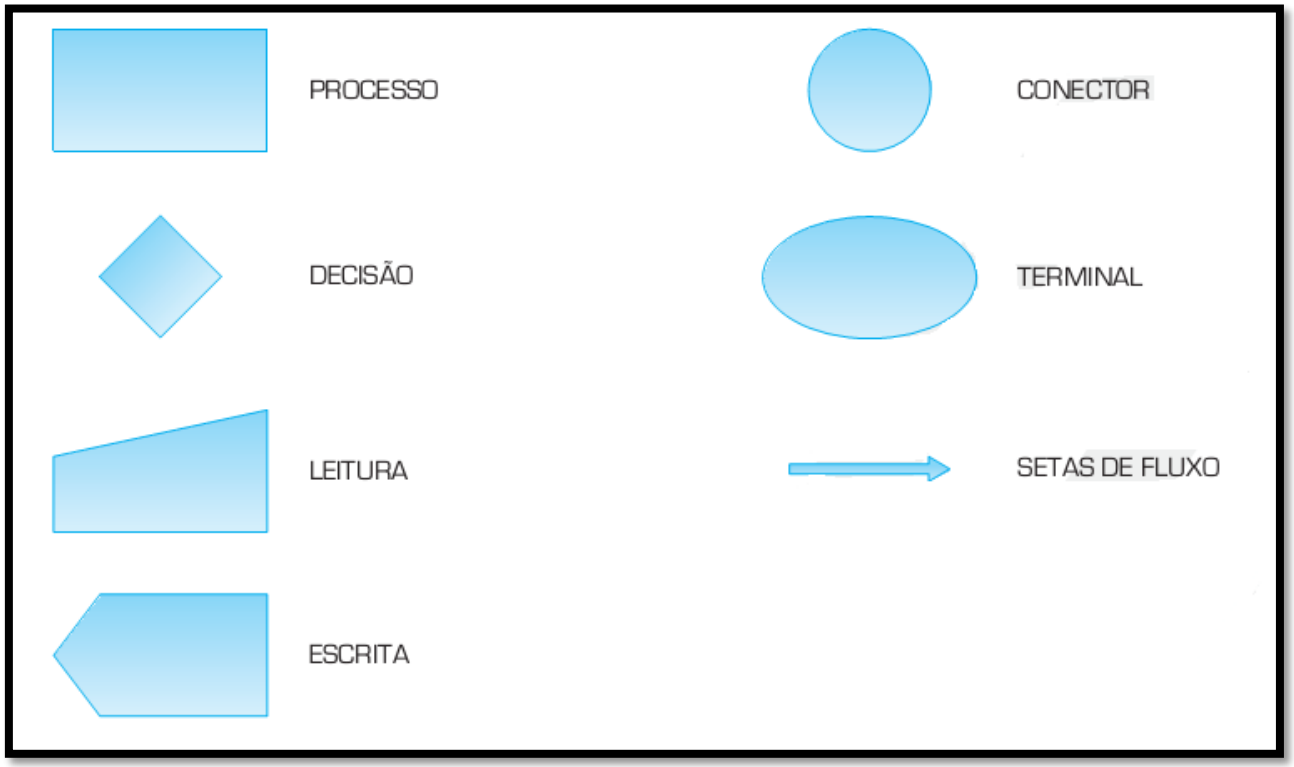
É uma descrição precisa e detalhada de um algoritmo, feita em uma notação que combina elementos gráficos e textuais.

O fluxograma consiste em analisar o enunciado do problema e escrever, utilizando símbolos gráficos predefinidos, os passos a serem seguidos para sua resolução.

Vantagem: o entendimento de elementos gráficos é mais simples que o entendimento de textos.

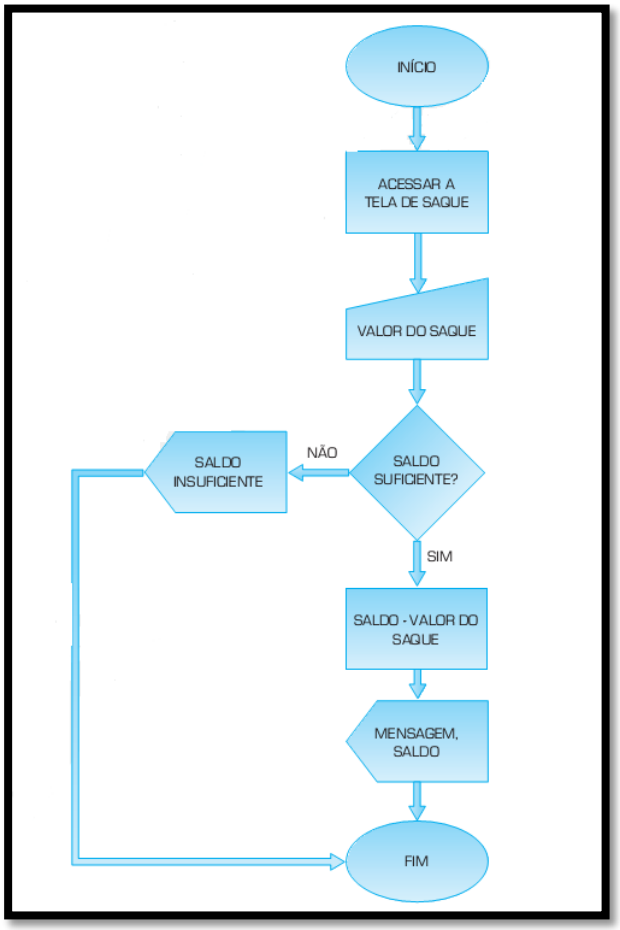
Desvantagem: é necessário aprender a simbologia dos fluxogramas e, além disso, o algoritmo resultante não apresenta muitos detalhes, dificultando sua transcrição para um programa.

CONJUNTO DE SÍMBOLOS UTILIZADOS NO FLUXOGRAMA



Vamos supor que seja necessário criar um algoritmo para sacar uma determinada quantia de dinheiro de um caixa eletrônico de um banco. Como ficaria o fluxograma desse algoritmo?

FLUXOGRAMA - EXEMPLO



Importante!

Para entender o algoritmo que um fluxograma representa, é necessário conhecer o significado de cada símbolo.

PSEUDOCÓDIGO OU PORTUGOL

O pseudocódigo ou portugol consiste em analisar o enunciado do problema e escrever, por meio de regras predefinidas, os passos a serem seguidos para sua resolução.

Escrever um algoritmo em pseudocódigo é outra forma muito utilizada por autores de livros que tratam de algoritmos, pois dessa forma o leitor não precisa ter o conhecimento prévio de nenhuma linguagem de programação. Nos países cujo idioma principal é o português, muitos se referem ao pseudocódigo como portugol.

Vantagem: a passagem do algoritmo para qualquer linguagem de programação é quase imediata, bastando conhecer as palavras reservadas da linguagem que será utilizada.

Desvantagem: é necessário aprender as regras do pseudocódigo.

Para mais exemplos de algoritmos dos tipos mostrados até aqui abra o arquivo Exemplos de Algoritmos - DN, Fluxogramas e Pseudocódigo.pdf disponível no ambiente Moodle.

PSEUDOCÓDIGO OU PORTUGOL

Vamos ver como ficaria o exemplo anterior escrito em pseudocódigo:

```
1  INICIO
2     LER(ValorDoSaque)
3     SE ValorDoSaque > 0 E ValorDoSaque <= Saldo ENTÃO
4        Saldo = Saldo - ValorDoSaque;
5        ESCREVER("Saque efetuado com sucesso. Saldo atual: ", Saldo);
6     SENÃO
7        ESCREVER("Saldo Insuficiente.");
8     FIM SE
9  FIM
```

A representação em pseudocódigo é bem simples e na maioria dos casos é suficiente para se explicar um algoritmo. Existem alguns interpretadores de portugol como o VisuAlg e, no caso da língua inglesa, temos algumas linguagens como Pascal e BASIC cuja sintaxe se assemelha muito com a sintaxe de um pseudocódigo em inglês.

FAÇA: Tente fazer em casa um fluxograma de um algoritmo para a operação de depósito em um caixa eletrônico de um banco.

CONCEITO DE VARIÁVEL

Um algoritmo e, posteriormente, um programa, recebem **dados** que precisam ser armazenados no computador para serem utilizados no **processamento.**

Esse armazenamento é feito na **memória.** Todos os computadores trabalham com **sistema numérico binário** e, nesse sistema, os dados são transformados em **0** e **1** ('zeros' e 'uns') para, então, serem armazenados na memória.

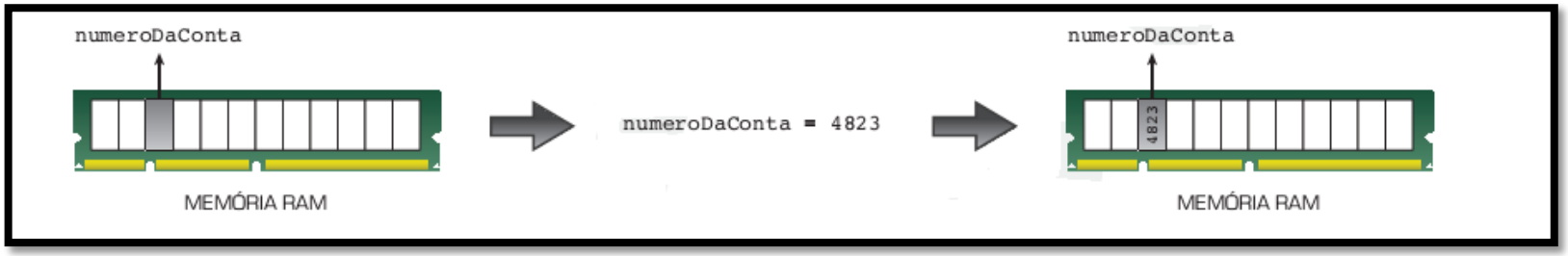
Cada dígito binário (0 ou 1) ocupa uma porção de memória chamada **bit,** e um conjunto de **8 bits** é denominado **byte.** Cada **byte** é identificado e acessado por meio de um **endereço.**

Uma **variável** representa uma posição de **memória,** que possui **nome, tipo, tamanho** e **seu conteúdo pode variar ao longo do tempo, durante a execução de um programa.** Embora uma variável possa assumir diferentes valores, **ela só pode armazenar um valor a cada instante.**

CONCEITO DE VARIÁVEL

- Os dados manipulados por um programa são armazenados em variáveis. Normalmente, uma variável é associada a uma posição da memória RAM. Nas variáveis é possível armazenar dados de vários tipos: numéricos, *strings* (texto), booleanos (verdadeiro ou falso), referências, entre outros.

- Toda variável possui um nome (um identificador). Os nomes das variáveis são utilizados para manipular os dados contidos nelas. Como, normalmente, as variáveis são associadas à posições da memória RAM, os identificadores das variáveis funcionam como nomes simbólicos dos endereços da memória RAM.



CONCEITO DE VARIÁVEL

Todos os caracteres existentes possuem um correspondente numérico na **tabela ASCII (American Standard Code for Information Interchange)**, transformado em caractere binário pelo método de divisão para, então, ser armazenado na memória.

Caractere	Valor decimal na tabela ASCII	Valor binário
A	65	01000001
B	66	01000010
C	67	01000011

Tabela 1 - Uma amostra da tabela ASCII.

Todo computador possui uma tabela de alocação que contém o nome da variável, seu tipo (para saber quantos bytes ocupará) e seu endereço inicial de armazenamento. Dessa maneira, quando queremos buscar algum dado na memória, basta sabermos o nome da variável, que o computador, por meio da tabela de alocação, busca automaticamente.

Para saber como converter um número decimal em binário e vice-versa acesse o arquivo Exemplo de transformação de número decimal em binário.xlsx no ambiente Moodle.

UNIDADES DE INFORMAÇÃO

Complete a pilha de dados sabendo que o valor de cada quadrado é igual a soma dos dois quadrados abaixo dele.

Saiba que:

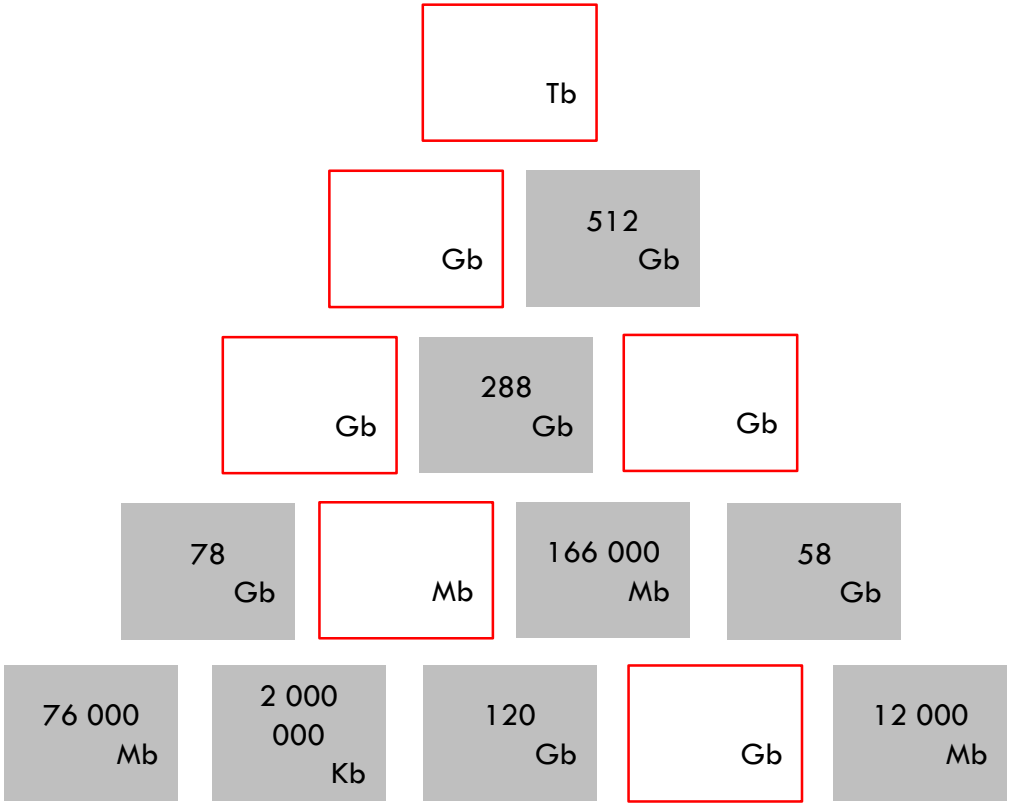
1 Terabyte -> 1Tb -> 1000 Gb

1 Gigabyte -> 1 Gb -> 1000 Mb

1 Megabyte -> 1 Mb -> 1000 Kb

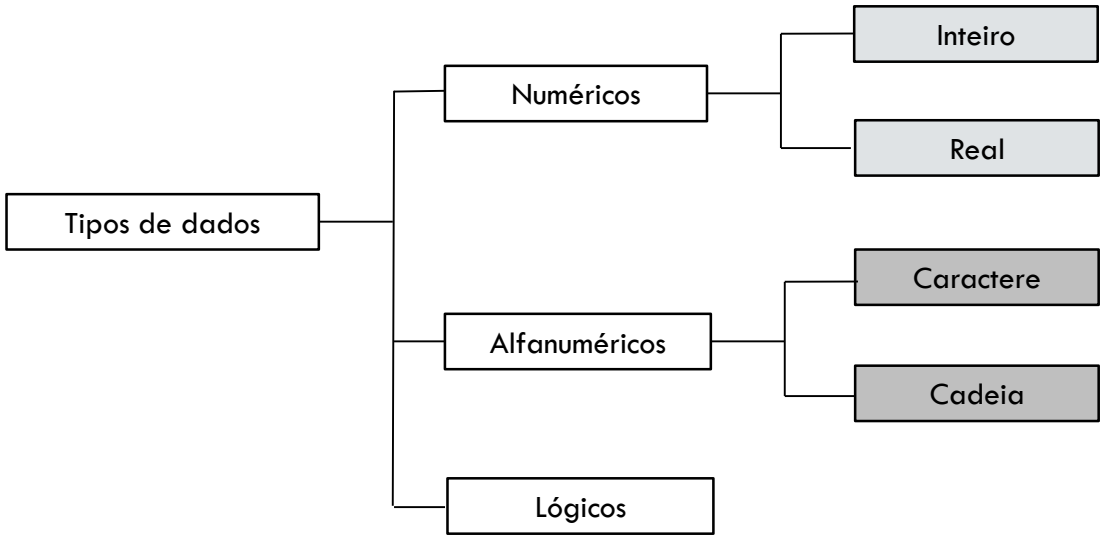
1 Kilobyte -> 1 Kb -> 1000 bytes

1 byte -> 8 bits



TIPOS DE DADOS

A principal finalidade dos computadores é o processamento de dados e, apesar de os computadores trabalharem internamente apenas com números binários, a maioria das linguagens de programação permite o uso de tipos de dados mais intuitivos. Os **tipos de dados** mais comuns podem ser classificados em:



Classificação dos tipos de dados mais comuns em programação

TIPOS DE DADOS

Os tipos de dados mais utilizados em programação são: *numéricos*, *lógicos* e *literais* ou *caracteres*.

Numéricos

Os dados numéricos dividem-se em dois grupos: *inteiros* e *reais*.

Os **números inteiros** podem ser positivos ou negativos e *não* possuem parte fracionária. Exemplos de dados numéricos inteiros:

-23	-357
98	237
0	-2

Os **números reais** podem ser positivos ou negativos e possuem parte fracionária. Exemplos de dados numéricos reais:

23.45	0.0
346.89	-247.0
-34.88	3.1415

Observação: Os números reais seguem a notação da língua inglesa, ou seja, a parte decimal é separada da parte inteira por um ponto, e não por uma vírgula.

TIPOS DE DADOS

Lógicos

São também chamados dados booleanos (oriundos da álgebra de Boole) e podem assumir valores *verdadeiro* ou *falso*.

Literais ou caracteres

São dados formados por um único caractere ou por uma cadeia de caracteres. Esses caracteres podem ser letras maiúsculas, as letras minúsculas, os números (não podem ser usados para cálculos) e os caracteres especiais (&, #, @, ?, +). Exemplos de dados literais:

"aluno"
"1234"
"@ internet"
"0.34"
"1 + 2"
'A'
'3'

Observação: Um caractere é representado entre aspas simples e um conjunto de caracteres é representado entre aspas duplas.

TIPOS DE DADOS

Determine qual é o tipo primitivo (inteiro, real, lógico ou caractere) de informação presente nas sentenças a seguir:

a) A placa "*Pare!*" tinha *2* furos de bala.

b) Josefina subiu *5* degraus para pegar uma maçã *boa*.

c) Alberta levou *3,5* horas para chegar ao hospital onde concebeu uma *garota*.

d) Astrogilda pintou em sua camisa: "*Preserve* o *meio ambiente*", e ficou devendo R$ *100,59* ao vendedor de tintas.

e) Felisberto recebeu sua *18ª* medalha por ter alcançado a marca de *57,3* segundos nos *100* metros rasos.

f) Danilo tem *32* anos de idade, *1,71* metros de altura e pesa *78,8* quilos.

FORMAÇÃO DE IDENTIFICADORES

Os **identificadores** são os **nomes das variáveis**, dos **programas**, das **constantes**, das **rotinas**, das **unidades** etc. As regras básicas para a formação dos identificadores são:

- Os caracteres permitidos são: números, letras maiúsculas, letras minúsculas e o caractere sublinhado (_).
- O primeiro caractere deve ser sempre uma letra ou o caractere sublinhado.
- Não são permitidos espaços em branco e caracteres especiais (@, $, +, -, %, !).
- Não podemos usar as palavras reservadas nos identificadores, ou seja, palavras que pertençam à linguagem de programação.

EXEMPLOS DE IDENTIFICADORES

Exemplos de identificadores válidos:

A	a
nota	NOTA
X5	A32
Nota1	MATRICULA
nota_1	_valor
dia	IDADE

Exemplos de identificadores inválidos:

5b	*por começar com número;*
e 12	*por conter espaço em branco;*
x-y	*por conter o caractere especial -;*
prova 2n	*por conter espaço em branco;*
nota(2)	*por conter os caracteres especiais ();*
if	*por ser palavra reservada;*

FORMAÇÃO DE IDENTIFICADORES - EXERCÍCIOS

Para cada identificador abaixo marque V se o mesmo for válido ou I se este for inválido.

( )	nome
( )	51
( )	você
( )	valor!
( )	6_nota
( )	preco_carro_novo
( )	printf
( )	?nome
( )	somadetodososnumerospares
( )	-x
( )	Aluno 1
( )	eu_amo_a_aula_de_programação
( )	onde eu vou aplicar este conhecimento ?

FORMAÇÃO DE IDENTIFICADORES - EXERCÍCIOS

Assinale os identificadores válidos:

a) (X)

b) U2

c) AH!

d) “ALUNO”

e) #55

f) KM/L

g) UYT

h) ASDRUBAL

i) AB*C

j) 0&0

k) P{0}

l) B52

m) Rua

n) CEP

o) dia/mês

p) Programa1

q) LoVe

r) 4 EVER

LINGUAGEM C/C++ - HISTÓRIA

Segundo alguns autores, Dennis Ritchie inventou a linguagem C e foi o primeiro a implementá-la usando um computador DEC PDP-11, que utilizava o sistema operacional Unix. Essa linguagem é resultante de um processo evolutivo de linguagens, cujo marco inicial foi uma linguagem chamada BCPL, desenvolvida por Martin Richards, que teve forte influência em uma linguagem denominada B, inventada por Ken Thompson. Na década de 1970, B levou ao desenvolvimento de C.

Durante alguns anos, o padrão da linguagem C foi aquele fornecido com a versão 5 do sistema operacional Unix, mas, com a popularização dos microcomputadores, várias implementações de C foram criadas, gerando, assim, muitas discrepâncias. Para resolver tal situação, o American National Standards Institute (ANSI) estabeleceu, em 1983, um comitê para definir um padrão que guiasse todas as implementações da linguagem C.

A linguagem C++ é uma extensão da linguagem C, e as instruções que fazem parte desta última representam um subconjunto da primeira. Os incrementos encontrados na linguagem C++ foram feitos para dar suporte à programação orientada a objetos, e a sintaxe dessa linguagem é basicamente a mesma da linguagem C.

LINGUAGEM C/C++ - CURIOSIDADES

- ✓ É a linguagem mais usada para criar softwares;
- ✓ É a linguagem mais usada para criar jogos;
- ✓ É a linguagem que está por trás da Internet;
- ✓ É a linguagem usada para escrever Sistemas Operacionais;
- ✓ É usada para *escrever quase todas as outras linguagens*;
- ✓ Pode ser usada para qualquer processador existente, de relógios e telefones a aviões e satélites;
- ✓ Pode ser usada embutida em pequenos dispositivos como o Arduino; e
- ✓ É a linguagem preferida pelos hackers.

CONCEITO DE PROGRAMA

Um **programa** é um algoritmo descrito em uma **linguagem de programação,** ou seja, em uma linguagem que o computador seja capaz de interpretar e executar.

Exemplo de programa escrito na linguagem de programação C

```c
/* media.c - calcula a média de um aluno */
#include <stdio.h>

int main(void)
{
    float a, b, c;

    printf("Notas? ");
    scanf("%f %f", &a, &b);

    c = (a+b)/2;

    printf("Media = %.1f\n", c);

    if(c >= 6.0)
        printf("Aprovado\n");
    else
        printf("Reprovado\n");

    return 0;
}
```

Em linguagem C temos:

- Comentários
- Diretivas
- Função principal
- Variáveis (declaração e atribuição)
- Exibição de dados
- Leitura de dados
- Operadores aritméticos, relacionais e lógicos
- Palavras reservadas
- Estruturas sequenciais, condicionais e de repetição

Para mais detalhes de como é um programa escrito em linguagem C, acesse o arquivo Elementos da linguagem de programação C.pdf no ambiente Moodle.

CRIAÇÃO DE PROGRAMAS E COMPILADORES (01/02)

As etapas básicas para a criação de um programa são as seguintes:

- **Análise:** nessa etapa, precisamos compreender o problema em questão, definindo que dados são fornecidos como **entrada**, que **processamento** deve ser efetuado e que informações devem ser apresentadas como **saída.**
- **Projeto:** nessa etapa, precisamos elaborar um **algoritmo** que descreva, passo a passo, como o computador deve proceder para obter os dados de entrada, processá-los e exibir as informações de saída, de acordo com o foi definido na etapa de análise (esta é a etapa em que construímos o **fluxograma** ou o **pseudocódigo**).
- **Implementação:** nessa etapa, precisamos codificar um **programa** correspondente ao algoritmo elaborado na etapa de projeto (esta é a etapa em que usamos uma **linguagem de programação**).
- **Teste:** nessa etapa, precisamos **executar** o programa em um computador e verificar como ele se comporta para diversos dados de entrada (esta é a etapa em que usamos um **compilador**).

CRIAÇÃO DE PROGRAMAS E COMPILADORES (02/02)

Um **compilador** é um programa que interpreta os comandos escritos em uma linguagem de programação e os converte em uma forma que o computador seja capaz de executar. Em geral, há vários compiladores diferentes para uma mesma linguagem de programação. Na ferramenta **Code::Blocks** é possível escolher qual compilador será usado para compilar os programas. Por padrão, a ferramenta utiliza o compilador chamado **GNU GCC Compiler**.

PARADIGMAS DE PROGRAMAÇÃO

Paradigma é um termo com origem no grego "paradeigma" que significa modelo, padrão. No sentido lato corresponde a algo que vai servir de modelo ou exemplo a ser seguido em determinada situação. São as normas orientadoras de um grupo que estabelecem limites e que determinam como um indivíduo deve agir dentro desses limites.

No paradigma estruturado de programação (também conhecido como *imperativo* ou *procedural*), qualquer problema pode ser quebrado em problemas menores, de mais fácil solução, chamados de sub-rotinas ou funções. Ele preconiza também que todo processamento pode ser realizado pelo uso de três tipos de **estruturas de controle: sequencial, condicional** e **iterativa (de repetição)**.

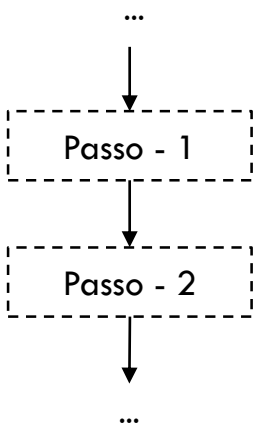
PROGRAMAÇÃO ESTRUTURADA

Em meados da década de 1960, Böhm e Jacopini provaram que todo algoritmo computacional pode ser descrito em termos de apenas três padrões de agrupamento de passos. Esses padrões, denominados **estruturas de controle**, são também os componentes básicos a partir dos quais os programas são construídos.

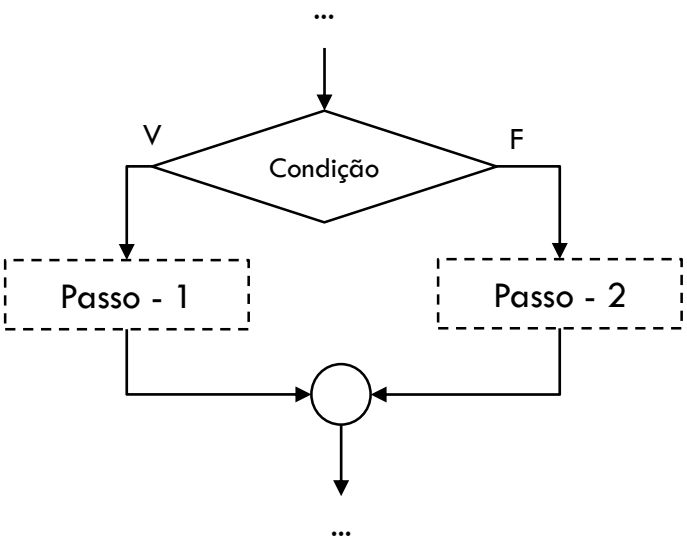
As três estruturas de controle básicas são:

- **Sequencia:** essa estrutura permite indicar dois ou mais passos que devem ser executados sequencialmente, na ordem em que são especificados.
- **Seleção:** permite indicar dois passos que devem ser executados de forma mutuamente exclusiva, dependendo de uma determinada condição.
- **Repetição:** permite indicar um ou mais passos que devem ser executados repetidamente, dependendo de uma determinada condição.

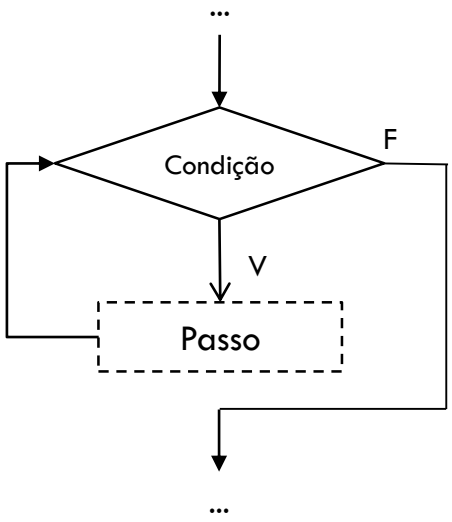
ESTRUTURAS DE CONTROLE



(a) Sequência

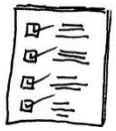


(b) Seleção



(c) Repetição

As três estruturas de controle básicas da programação estruturada

RESUMO 

- ✓A **Lógica** se relaciona com a “ordem da razão”, com a “correção do pensamento”, e que é necessário utilizar processos lógicos de programação para construir algoritmos;
- ✓**Algoritmo** é uma sequencia de passos bem definidos que têm por objetivo solucionar um determinado problema;
- ✓A **estrutura sequencial** significa que o algoritmo é executado passo a passo, sequencialmente, da primeira a última ação;
- ✓A **estrutura de seleção** permite que uma ação seja ou não executada, dependendo do valor resultante da inspeção de uma condição; e
- ✓A **estrutura de repetição** permite que trechos de algoritmos sejam repetidos até que uma condição seja satisfeita ou enquanto uma condição não estiver satisfeita.

REFERÊNCIAS BIBLIOGRÁFICAS

- ASCENCIO, A. F. G.; CAMPOS, E. A. V. D. **Fundamentos da Programação de Computadores: Algoritmos, Pascal, C/C++ (Padrão ANSI) e Java.** 3. ed. São Paulo: Pearson Education do Brasil, 2012. 569 p.
- FORBELLONE, A. L. V.; EBERSPACHER, H. F. **Lógica de Programação: A construção de algoritmos e estruturas de dados.** 3. ed. São Paulo: Prentice Hall, 2005. 218p.
- PEREIRA, S. D. L. **Algoritmos e Lógica de Programação em C: Uma abordagem didática.** 1. ed. São Paulo: Érica, 2010. 190 p.