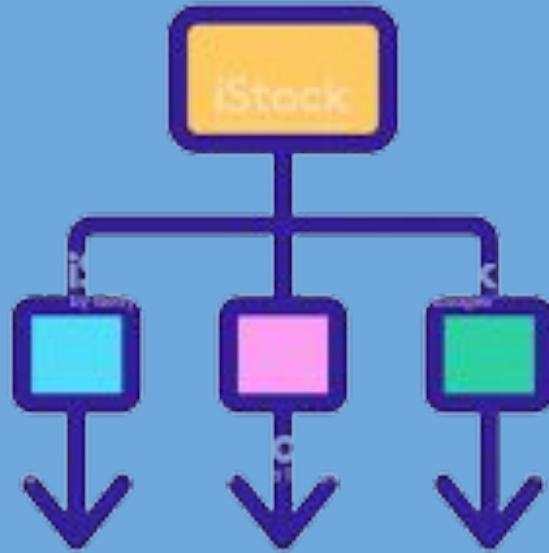
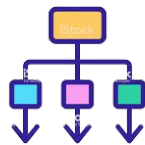


Algoritmo



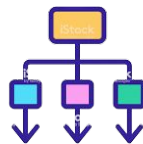
Conceito



“Algoritmo é uma sequência de passos que visa atingir um objetivo bem definido” (FORBELLONE, 1999).

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

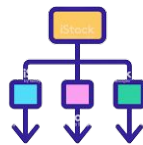
Conceito



“Algoritmo é uma sequência finita de instruções ou operações cuja execução, em tempo finito, resolve um problema computacional, qualquer que seja sua instância” (SALVETTI, 1999).

“Algoritmos são regras formais para a obtenção de um resultado ou da solução de um problema, englobando fórmulas de expressões aritméticas” (MANZANO, 1997).

Conceito



“Ação é um acontecimento que, a partir de um estado inicial, após um período de tempo finito, produz um estado final previsível e bem definido. Portanto, um algoritmo é a descrição de um conjunto de comandos que, obedecidos, resultam numa sucessão finita de ações” (FARRER, 1999).



Exemplo de Algoritmo

Vejam os algoritmos abaixo:

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: Multiplicar dois números

Passo 1 — Receber os dois números.

Passo 2 — Multiplicar os dois números.

Passo 3 — Mostrar o resultado obtido.



Exemplo de Algoritmo

Algoritmo 3 — 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 alface e tomate.

Passo 6 — Colocar alface e tomate.

Passo 7 — Pegar o hambúrguer.

Passo 8 — Fritar o hambúrguer.

Passo 9 — Colocar o hambúrguer.



"Mas eu realizo essas atividades de maneira diferente!"

Podem existir vários algoritmos para solucionar o mesmo problema.

Passos para construir algoritmo

Compreender completamente o problema a ser resolvido, destacando os pontos mais importantes e os objetos que o compõem.

Definir os dados de entrada, quais dados serão fornecidos.

Definir o processamento, quais cálculos serão efetuados.



Passos para construir algoritmo

Definir os dados de saída, ou seja, quais dados serão gerados depois do processamento.

Construir o algoritmo.

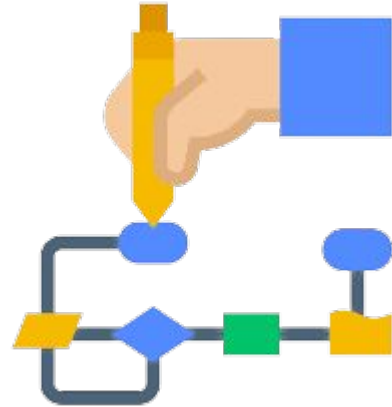
Testar o algoritmo.



Tipos de Algoritmos

Os três tipos mais utilizados de algoritmos são:

1. Descrição narrativa;
2. Fluxograma;
3. 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.



Descrição Narrativa

Exemplo de Algoritmo em descrição narrativa:

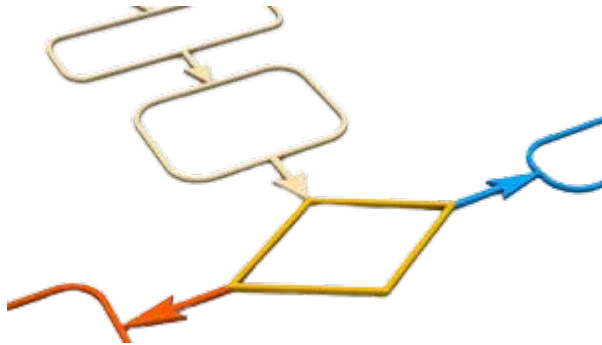
Passo 1 — Receber dois números que serão multiplicados.

Passo 2 — Multiplicar os números.





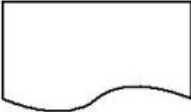
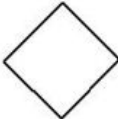
Passo 3 — Mostrar o resultado obtido na multiplicação.

Fluxograma

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

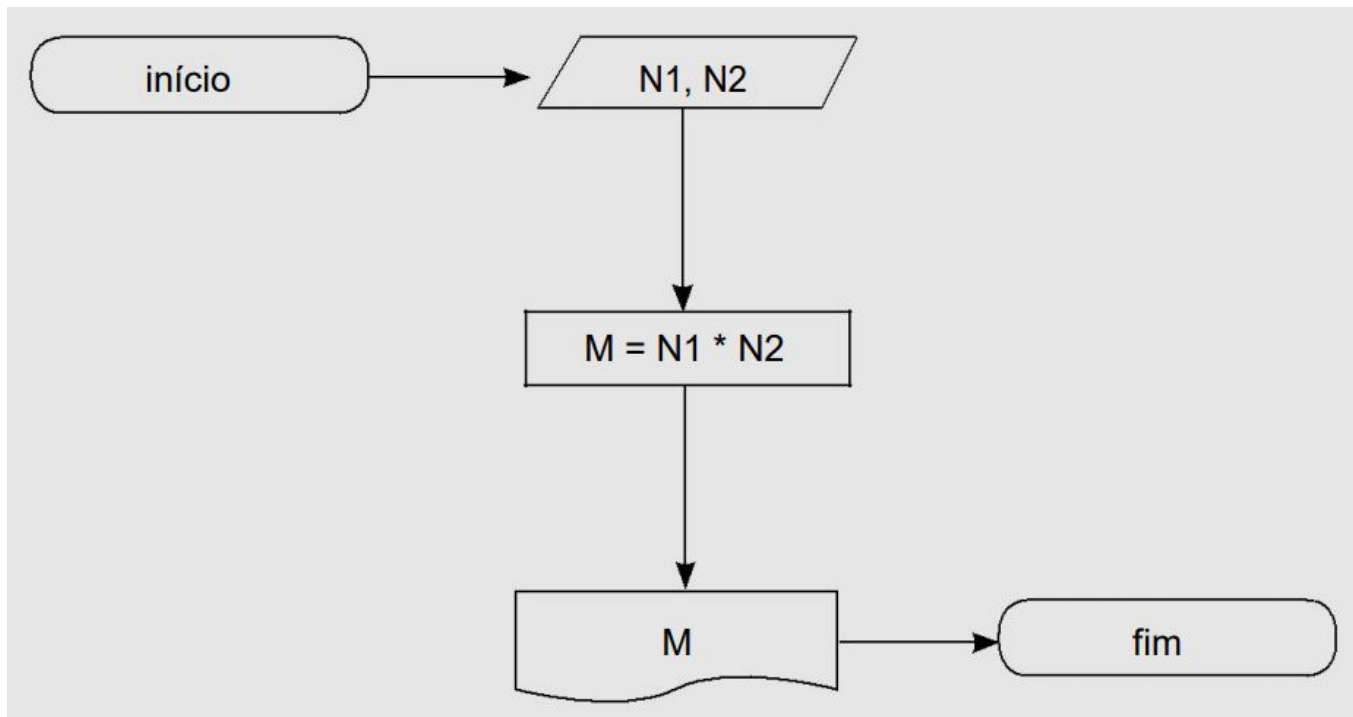


Símbolos Fluxograma

	Símbolo utilizado para indicar o início e o fim do algoritmo.
	Símbolo que permite indicar o sentido do fluxo de dados. Serve exclusivamente para conectar os símbolos ou blocos existentes.
	Símbolo utilizado para indicar cálculos e atribuições de valores.
	Símbolo utilizado para representar a entrada de dados.
	Símbolo utilizado para representar a saída de dados.
	Símbolo utilizado para indicar que deve ser tomada uma decisão, apontando a possibilidade de desvios.

Fluxograma

Exemplo:



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.



Pseudocódigo ou Portugol

Exemplo:

```
ALGORITMO  
DECLARE N1, N2, M NUMÉRICO  
ESCREVA "Digite dois números"  
LEIA N1, N2  
M ← N1 * N2  
  
ESCREVA "Multiplicação = ", M  
FIM_ALGORITMO.
```

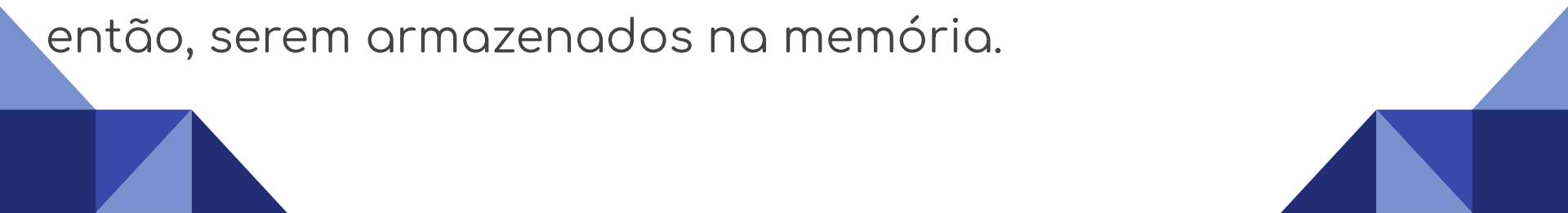

Variável

Um programa recebe 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.

Os dados são transformados em 0 e 1 ('zero' e 'um') para, então, serem armazenados na memória.



Variável

Cada dígito binário (0 ou 1) ocupa uma porção de memória chamada bit.

Um conjunto de 8 bits é denominado byte.

Todos os caracteres existentes são transformado em caractere binário para ser armazenado na memória.

ASCII Alphabet			
A	1000001	N	1001110
B	1000010	O	1001111
C	1000011	P	1010000
D	1000100	Q	1010001
E	1000101	R	1010010
F	1000110	S	1010011
G	1000111	T	1010100
H	1001000	U	1010101
I	1001001	V	1010110
J	1001010	W	1010111
K	1001011	X	1011000
L	1001100	Y	1011001
M	1001101	Z	1011010

Variável

Uma variável representa uma posição de memória, que possui nome e tipo 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.

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

Tipos de Dados

Os tipos de dados mais utilizados são:

1. Numéricos (inteiros e reais).
2. Lógicos (boolean).
3. Literais ou caracteres.



Tipos de Dados: 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 98 0 -357

Tipos de Dados: Numéricos

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

Os números reais podem ser positivos ou negativos e possuem parte fracionária.

Exemplos de dados numéricos reais:

23.45

346.89

-34.88

0.0

-247.0

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 os valores verdadeiro ou falso.



Verdadeiro / True



Falso/ False

Tipos de Dados: Caracteres

Literais ou Caracteres são dados formados por um único caractere ou por uma cadeia de caracteres.

Esses caracteres podem ser as 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'

Formação dos 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: os números, as letras maiúsculas, as letras minúsculas e o caractere sublinhado.
- O primeiro caractere deve ser sempre uma letra ou o caractere sublinhado.

Formação dos 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:

- 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.

Formação dos Identificadores

Exemplos de identificadores válidos:

```
NOTA  
X5  
A32  
NOTA1  
MATRICULA  
nota_1  
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 ();  
case — por ser palavra reservada;  
SET — por ser palavra reservada.
```

Paradigma de Programação



Paradigmas de Programação

Um paradigma de programação está intimamente relacionado à forma de pensar do programador e como ele busca a solução para os problemas.

É o paradigma que permite ou proíbe a utilização de algumas técnicas de programação.

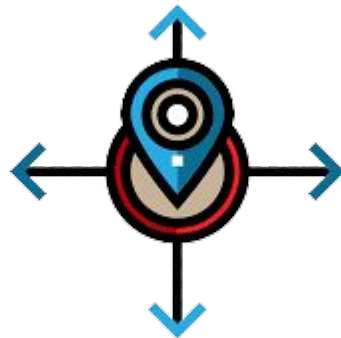
Ele é capaz, ainda, de mostrar como o programador analisou e abstraiu o problema a resolver.

Existem vários paradigmas de programação: estruturado, orientado a objetos, lógico, funcional, dentre outros.

Paradigmas de Programação

Pelo paradigma estruturado (também conhecido como imperativo ou procedural), qualquer problema pode ser quebrado em problemas menores, de mais fácil solução, chamados de sub-rotinas ou funções.

Todo processamento pode ser realizado pelo uso de três tipos de estrutura: sequencial, condicional e iterativa (de repetição).



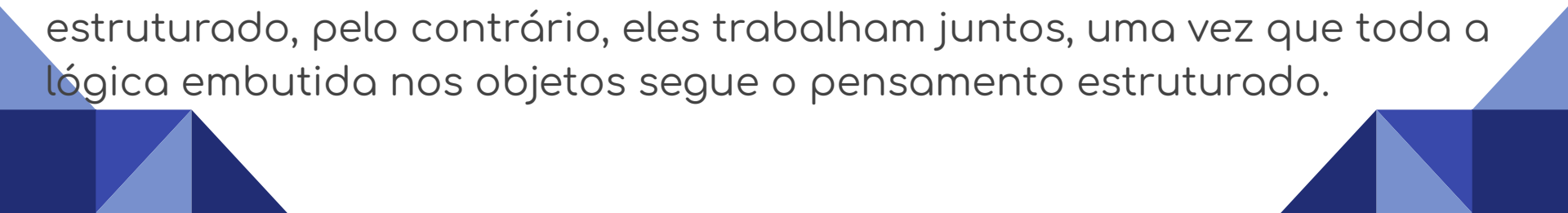
Paradigmas de Programação

Já o paradigma orientado a objetos compreende o problema como uma coleção de objetos interagindo por meio de trocas de mensagem.

Os objetos são estruturas de dados contendo estado (dados) e comportamento (lógica).

Um conjunto de objetos com informações comuns e com o mesmo comportamento dá origem a uma classe.

Deve-se observar que o paradigma orientado a objetos não exclui o estruturado, pelo contrário, eles trabalham juntos, uma vez que toda a lógica embutida nos objetos segue o pensamento estruturado.



Paradigmas de Programação

```
1 import java.util.Scanner;
2
3 class Suma {
4
5     public static void main() {
6         int a, b, c = 0;
7         Scanner s = new Scanner(System.in);
8         System.out.println("Digite dois números:");
9         a = s.nextInt();
10        b = s.nextInt();
11        c = a + b;
12        System.out.println("A soma de a+b = " + c);
13    }
14 }
15 }
```

```
1 #include <iostream>
2 using namespace std;
```

```
1 Program somar ;
2 var
3 a,b: integer;
4 Begin
5     writeln('Digite dois números:');
6     readln (a);
7     readln(b);
8     writeln ('A soma de a+b = ', a+b);
9
10 End.
```