ALGORITMOS E LÓGICA DE PROGRAMAÇÃO

ANDREI LUIZ DEMETRIO E SILVA

andrei.silva@ifpa.edu.br

Introdução

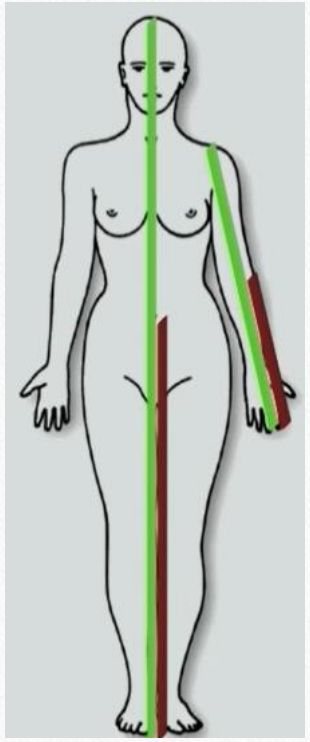
- Nome o qual as pessoas quase chegam a compreender logo de cara
- Se parece bastante com outra palavra que ouvimos desde criança: **Algarismo**
- Possuem a mesma origem, porém são coisas diferentes

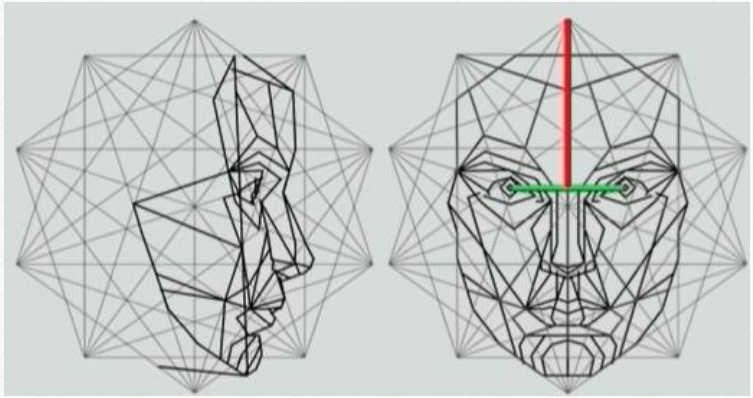
O que é Algoritmo?

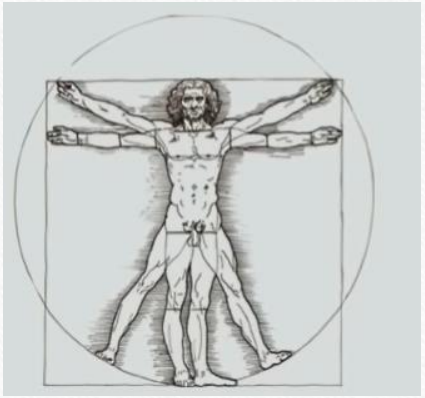
Algoritmo

"**Algoritmos** são conjuntos de passos **finitos** e **organizados** que, quando executados, resolvem um determinado **problema**" MANZANO, 2009

Proporção **Áurea**

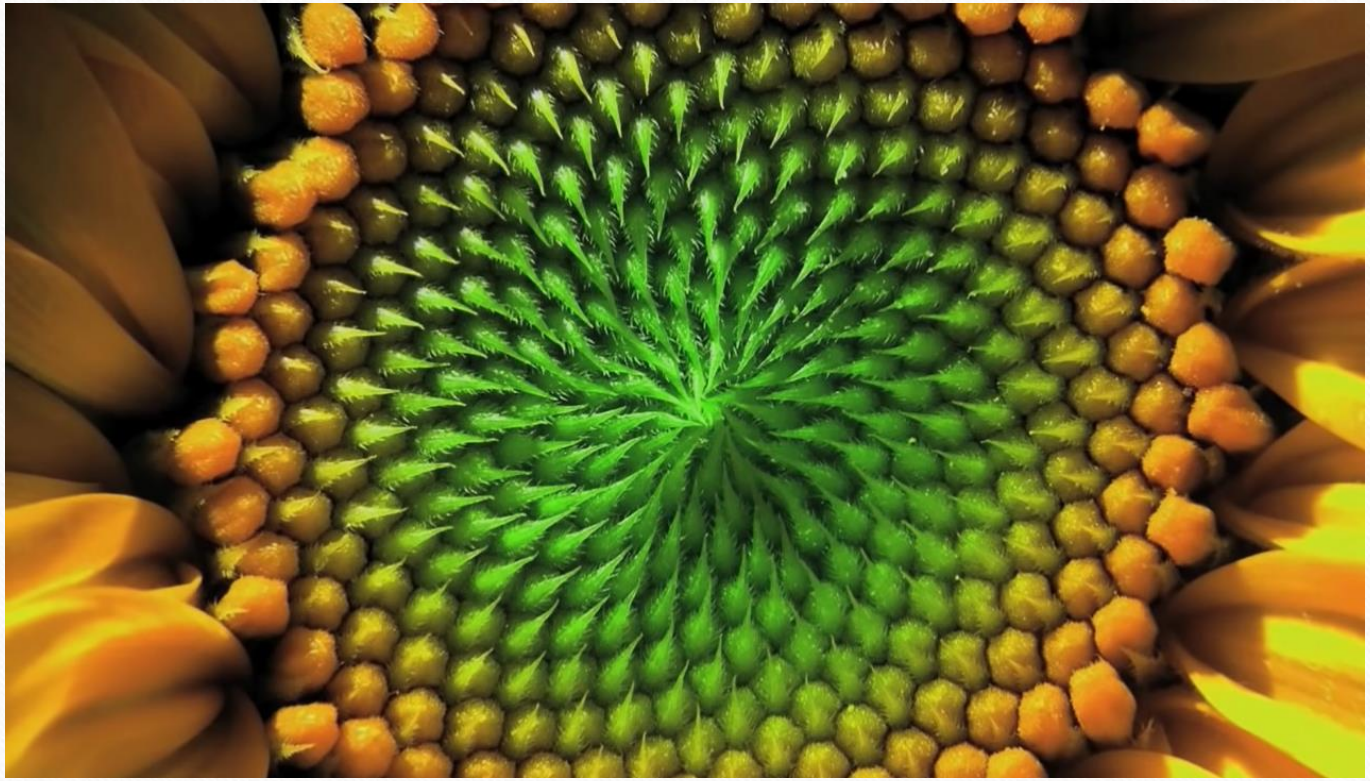




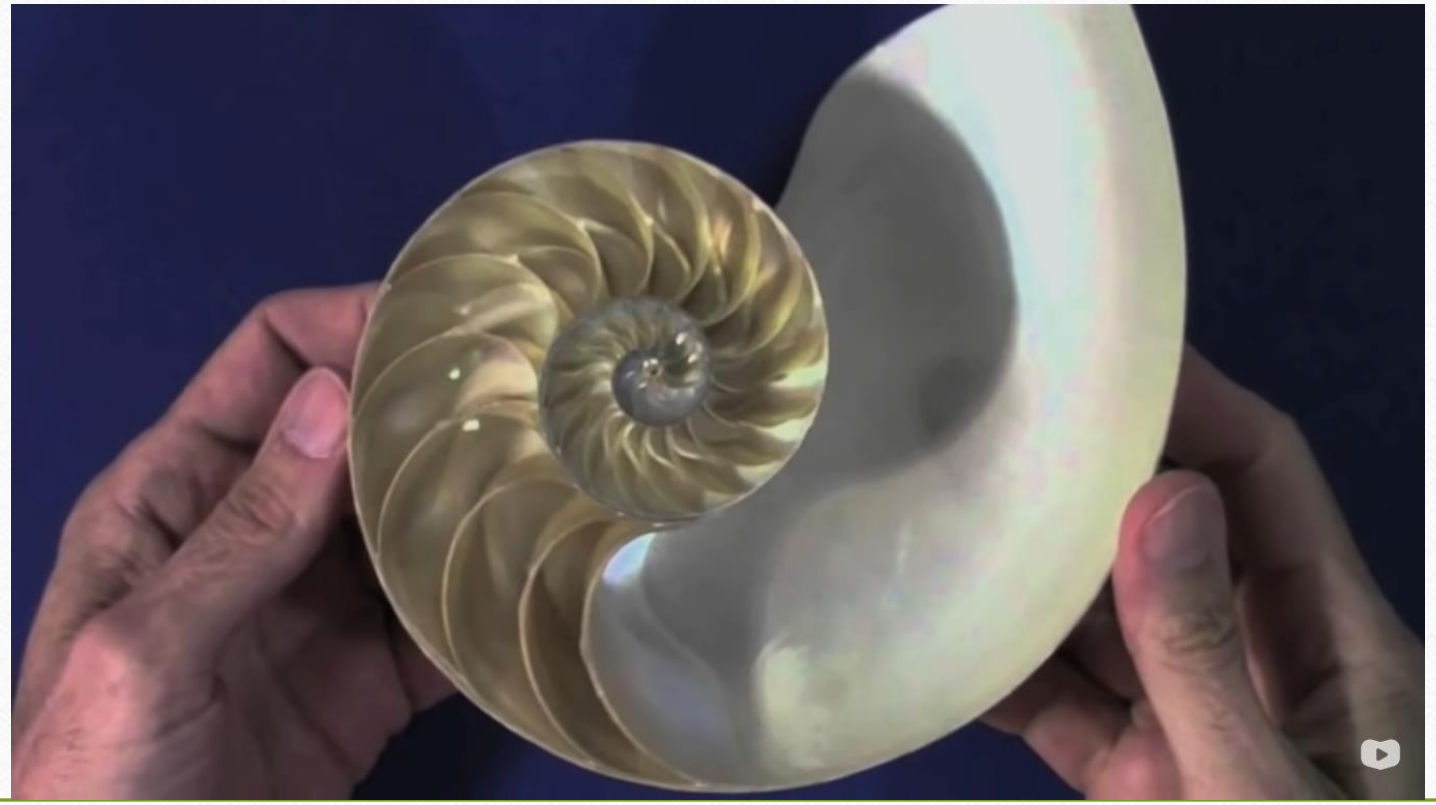


$$\varphi \text{ phi} = 1{,}618_{0339885}$$

Proporção **Áurea**



Proporção **Áurea**




Beethoven's FIFTH SYMPHONY
(First Movement Theme)


NATIONAL GEOGRAPHIC
$\frac{a}{b}$ = 1.61 !!! goldenratio
TOYOTA
$\frac{a}{b} = \frac{a'}{b'}$ = 1.618 !!!
goldenratio
bp
the concentric circles are in goldenratio
$\frac{a+b}{a} = \frac{a}{b} = \varphi \approx 1{,}61803$

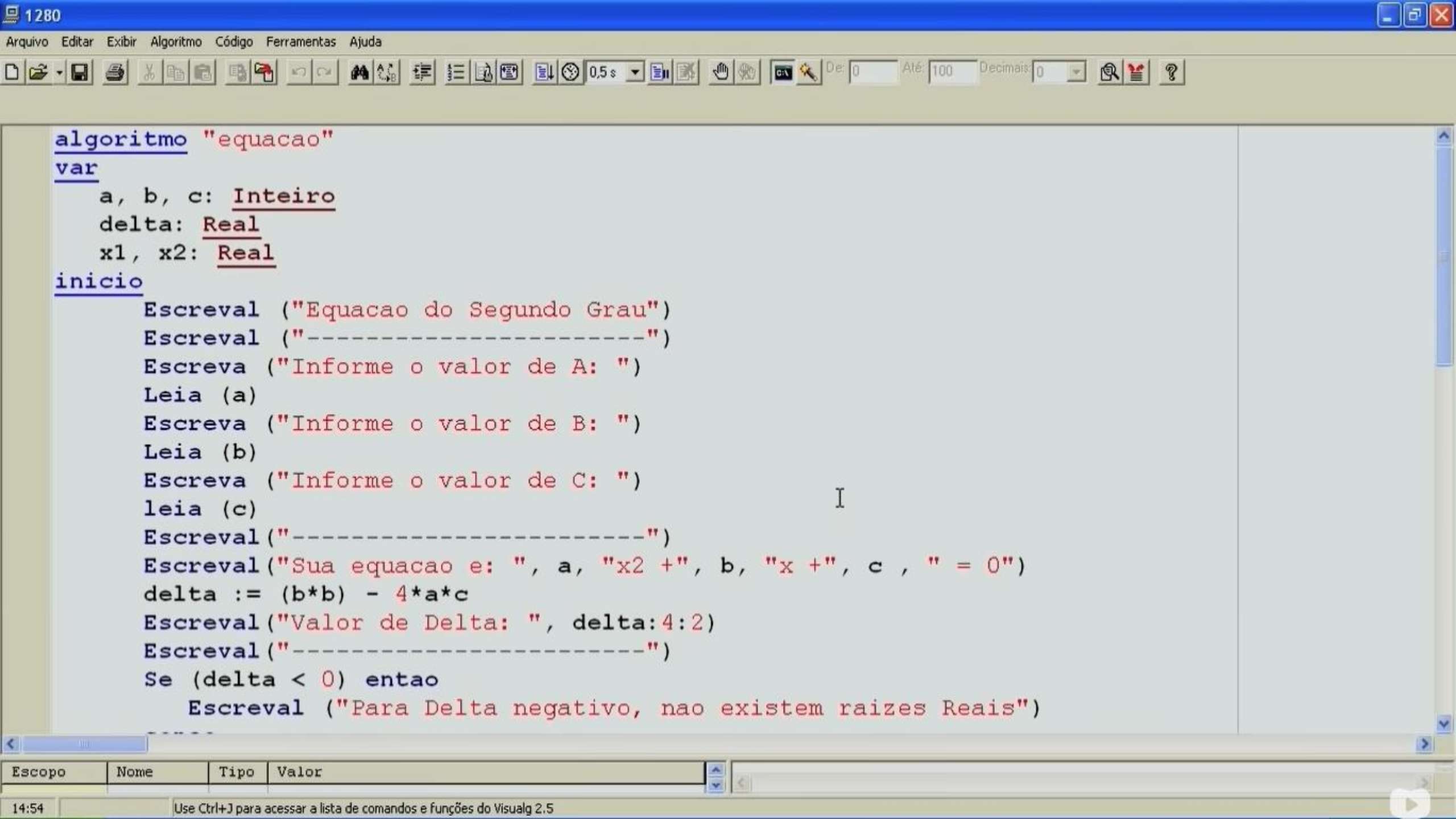
Algoritmo

- Toda reprodução de padrões é chamada de **rotina**.

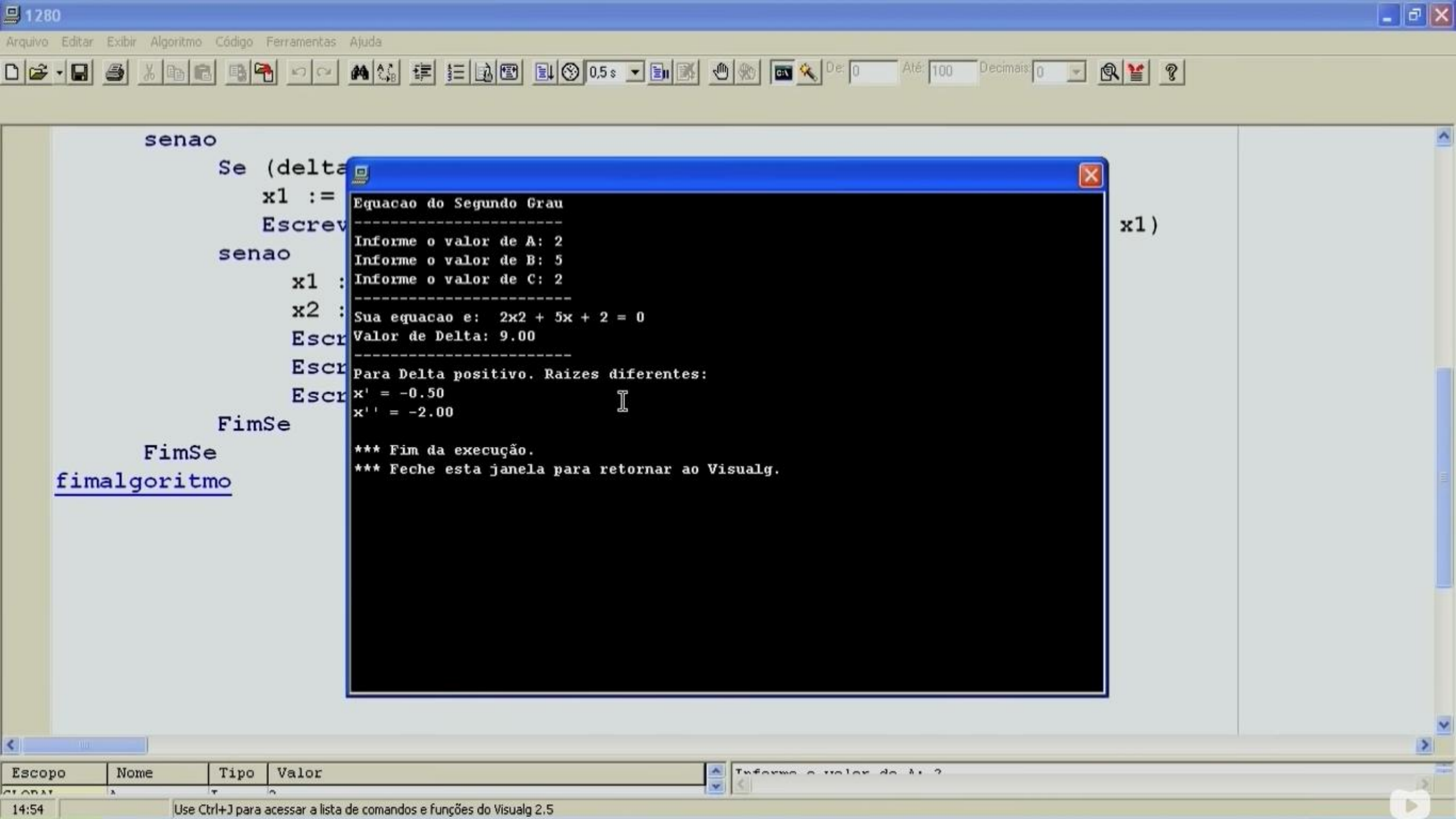
```
Algoritmo AtravessarRua
    Olhar para a direita
    Olhar para a esquerda
    Se estiver vindo carro
        Não Atravesse
    senão
        Atravesse
    Fim-Se
Fim-Algoritmo
```

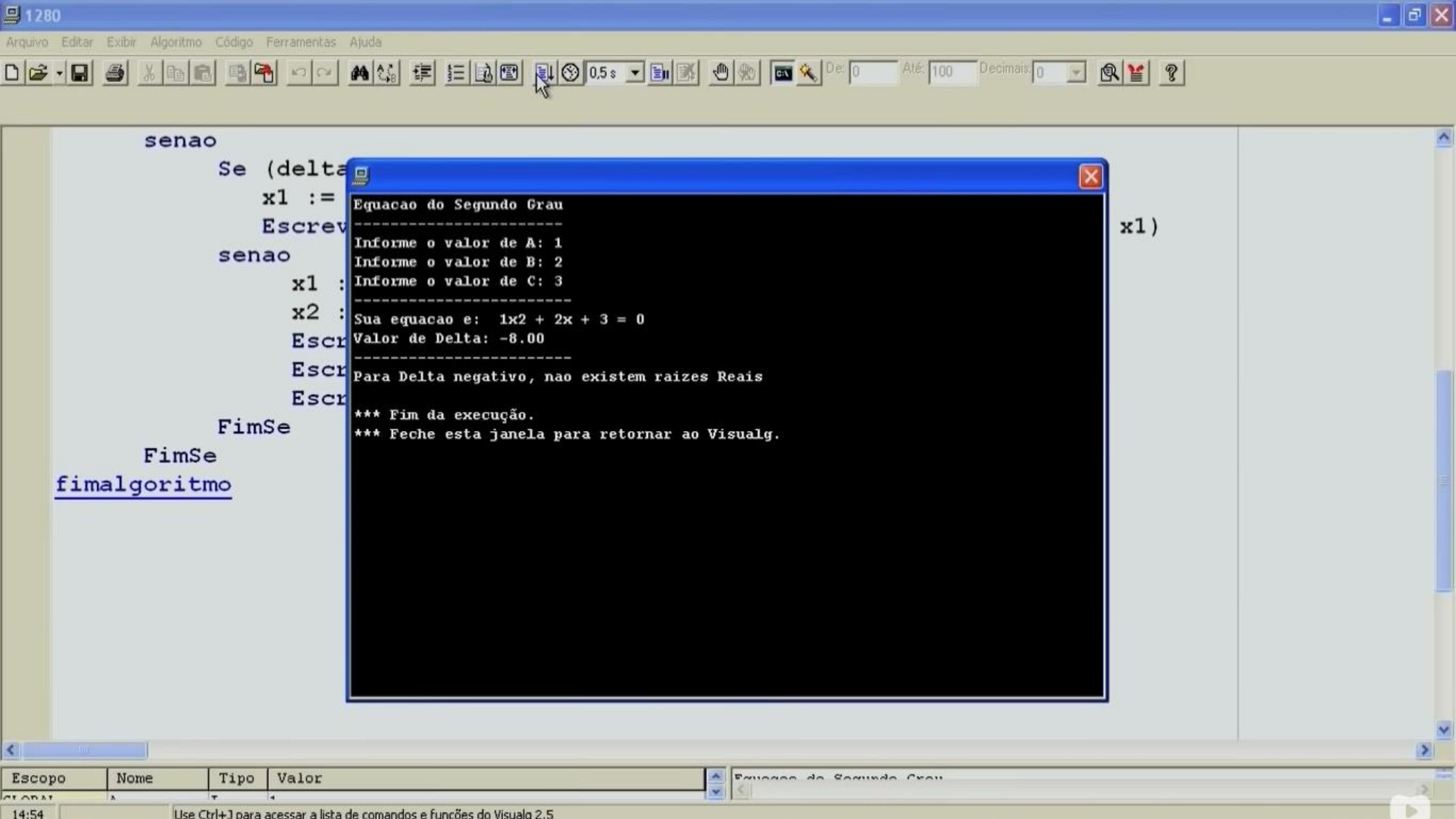
```
Algoritmo AtravessarRua
    Atravesse
    Se estiver vindo carro
        Olhar para a direita
    senão
        Olhar para a esquerda
    Fim-Se
    Não Atravesse
Fim-Algoritmo
```

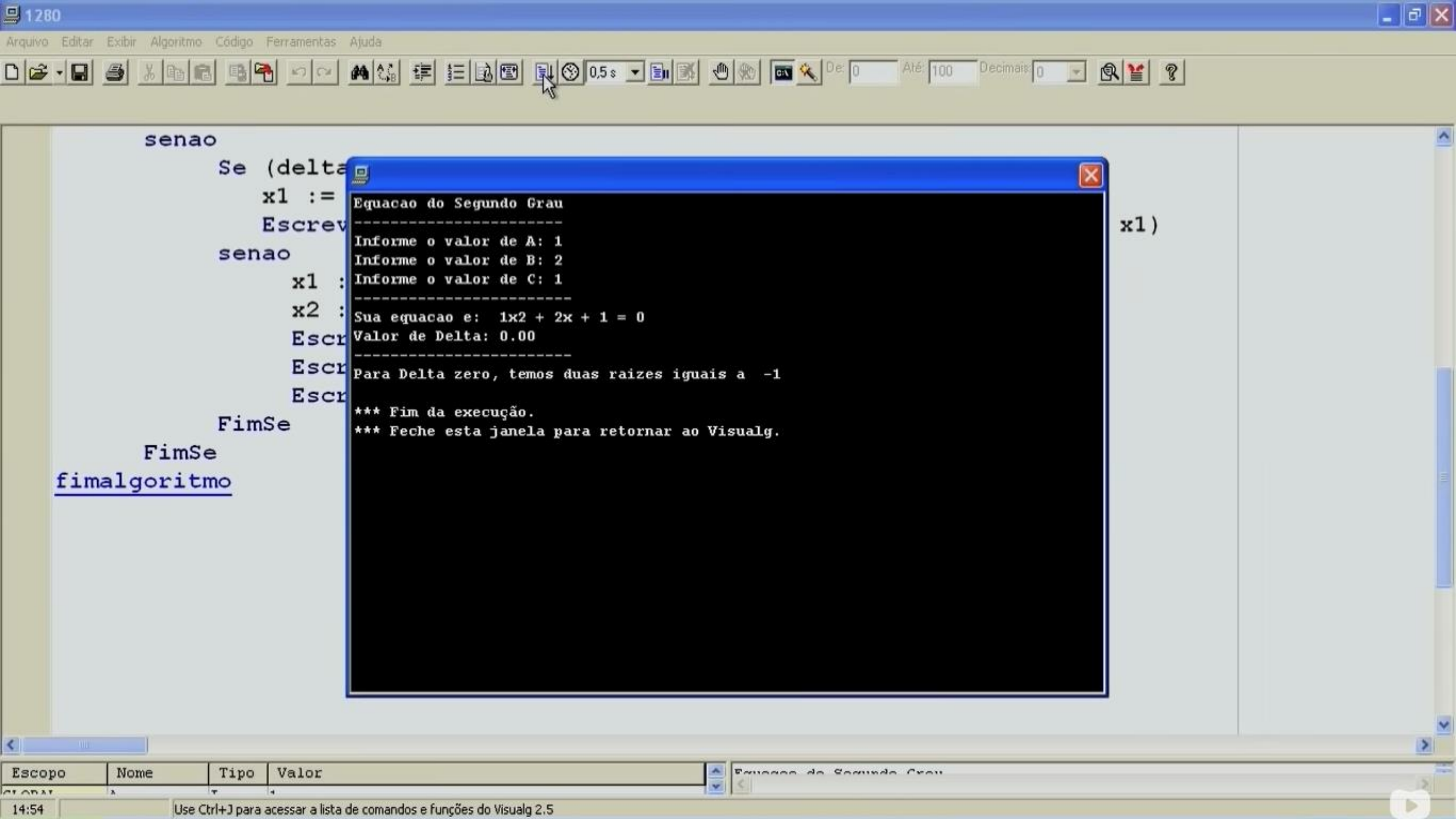

Keyboard or joystick (K or J) ?


1280
Arquivo Editar Exibir Algoritmo Código Ferramentas Ajuda
algoritmo "equacao"
var
a, b, c: Inteiro
delta: Real
x1, x2: Real
inicio
Escreval ("Equacao do Segundo Grau")
Escreval ("-----------------------")
Escreva ("Informe o valor de A: ")
Leia (a)
Escreva ("Informe o valor de B: ")
Leia (b)
Escreva ("Informe o valor de C: ")
leia (c)
Escreval("-----------------------")
Escreval("Sua equacao e: ", a, "x2 +", b, "x +", c , " = 0")
delta := (b*b) - 4*a*c
Escreval("Valor de Delta: ", delta:4:2)
Escreval("-----------------------")
Se (delta < 0) entao
Escreval ("Para Delta negativo, nao existem raizes Reais")
Escopo
Nome
Tipo
Valor
14:54
Use Ctrl+J para acessar a lista de comandos e funções do Visualg 2.5

```
        senao
              Se (delta = 0) entao
                 x1 := (-b + raizQ(delta))/(2*a)
                 Escreval ("Para Delta zero, temos duas raizes iguais a ", x1)
              senao
                    x1 := (-b + raizQ(delta))/(2*a)
                    x2 := (-b - raizQ(delta))/(2*a)
                    Escreval ("Para Delta positivo. Raizes diferentes: ")
                    Escreval ("x' = ", x1:4:2)
                    Escreval ("x'' = ", x2:4:2)
              FimSe
        FimSe
fimalgoritmo
```


1280
Arquivo Editar Exibir Algoritmo Código Ferramentas Ajuda
senao
Se (delta
x1 :=
Escrev
x1)
senao
x1 :
x2 :
Escr
Escr
Escr
FimSe
FimSe
fimalgoritmo
Equacao do Segundo Grau
Informe o valor de A: 2
Informe o valor de B: 5
Informe o valor de C: 2
Sua equacao e: 2x2 + 5x + 2 = 0
Valor de Delta: 9.00
Para Delta positivo. Raizes diferentes:
x' = -0.50
x'' = -2.00
*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
Escopo
Nome
Tipo
Valor
14:54
Use Ctrl+J para acessar a lista de comandos e funções do Visualg 2.5


1280
Arquivo Editar Exibir Algoritmo Código Ferramentas Ajuda
0,5 s
De: 0
Até: 100
Decimais: 0
senao
Se (delta
x1 :=
Escrev
x1)
senao
x1 :
x2 :
Escr
Escr
Escr
FimSe
FimSe
fimalgoritmo
Equacao do Segundo Grau
Informe o valor de A: 1
Informe o valor de B: 2
Informe o valor de C: 3
Sua equacao e: 1x2 + 2x + 3 = 0
Valor de Delta: -8.00
Para Delta negativo, nao existem raizes Reais
*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
Escopo
Nome
Tipo
Valor
14:54
Use Ctrl+J para acessar a lista de comandos e funções do Visualg 2.5


1280
Arquivo Editar Exibir Algoritmo Código Ferramentas Ajuda
0,5 s
De: 0
Até: 100
Decimais: 0
senao
Se (delta
x1 :=
Escrev
x1)
senao
x1 :
x2 :
Escr
Escr
Escr
FimSe
FimSe
fimalgoritmo
Equacao do Segundo Grau
Informe o valor de A: 1
Informe o valor de B: 2
Informe o valor de C: 1
Sua equacao e: 1x2 + 2x + 1 = 0
Valor de Delta: 0.00
Para Delta zero, temos duas raizes iguais a -1
*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
Escopo
Nome
Tipo
Valor
14:54
Use Ctrl+J para acessar a lista de comandos e funções do Visualg 2.5

Algoritmos Computacionais

Algoritmos Computacionais

- São passos a serem seguidos por um **módulo processador** e seus respectivos **usuários** que, quando executados na ordem correta, conseguem **realizar** determinada **tarefa**.

Internet Banking

Organização de Computadores

Organização de um Computador

- **Unidade de entrada**: em que ocorre a entrada de dados. Ex.: teclado, mouse.
- **Unidade de saída**: há a saída de informações. Ex.: monitor, impressora.
- **Unidade de Processamento Central**: responsável pelo processamento das informações e alocação de recursos.
- **Memória**: armazenamento de dados (RAM, HD, ROM, Cache).

Algoritmos

Propriedades

- Os algoritmos possuem cinco propriedades que os caracterizam:
  - **Finitude;**
  - **Definição;**
  - **Entrada;**
  - **Saída;**
  - **Eficácia.**

Algoritmo

- Antes de aprender a utilizar uma linguagem de programação específica, é importante que você compreenda que programar é:

“basicamente estruturar dados e construir algoritmos”

Formas de Representação de Algoritmos

- **Descrição Narrativa**;
- **Fluxograma Convencional**;
- **Pseudocódigo**, também conhecido como Linguagem Estruturada ou Portugol.

Descrição Narrativa

- Nesta forma de representação os algoritmos são expressos diretamente em linguagem natural;

Troca de um pneu furado:
Afrouxar ligeiramente as porcas
Suspender o carro
Retirar as porcas e o pneu
Colocar o pneu reserva
Apertar as porcas
Abaixar o carro
Dar o aperto final nas porcas

Receita de bolo:
Misture os ingredientes
Unte a forma com manteiga
Despeje a mistura na forma Se houver coco ralado então despeje sobre a mistura
Leve a forma ao forno
Enquanto não corar deixe a forma no forno
Retire do forno
Deixe esfriar

Tomando um banho:
Entrar no banheiro e tirar a roupa
Abrir a torneira do chuveiro
Entrar na água
Ensaboar-se
Sair da água
Fechar a torneira
Enxugar-se
Vestir-se

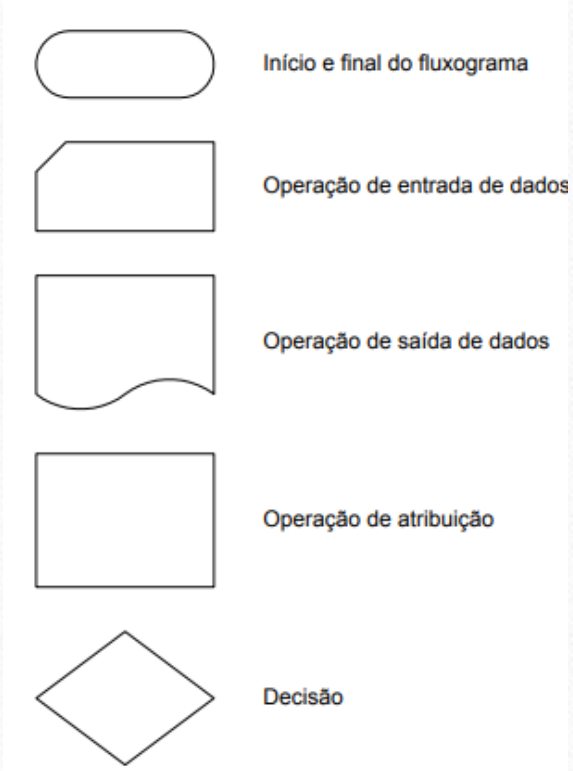
Descrição Narrativa

- Esta representação é pouco usada na prática porque o uso da linguagem natural muitas vezes dá oportunidade a más interpretações, ambiguidades e imprecisões.
- Por exemplo, a instrução "afrouxar ligeiramente as porcas" no algoritmo da troca de pneus está sujeita a interpretações diferentes por pessoas distintas. Uma instrução mais precisa seria: "afrouxar a porca, girando-a 30º no sentido anti-horário".

Fluxograma Convencional

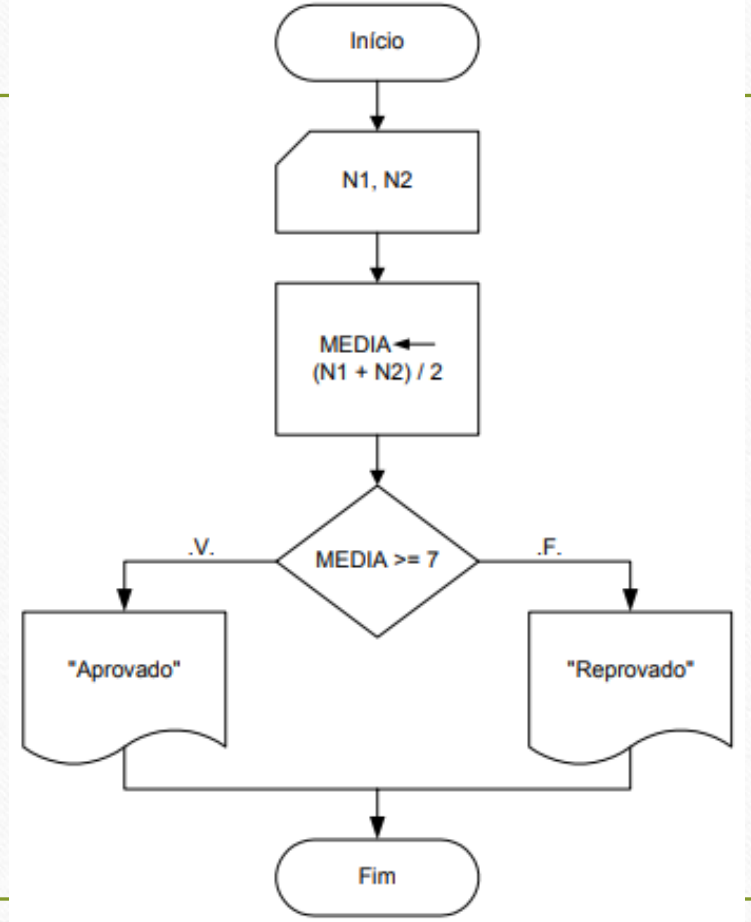
- É uma representação gráfica de algoritmos onde formas geométricas diferentes implicam ações (instruções, comandos) distintos.
- Tal propriedade facilita o entendimento das ideias contidas nos algoritmos e justifica sua popularidade.

Fluxograma Convencional



Fluxograma Convencional

De modo geral, um fluxograma se resume a um único símbolo inicial por onde a execução do algoritmo começa, e um ou mais símbolos finais, que são pontos onde a execução do algoritmo se encerra



Partindo do símbolo inicial, há sempre um único caminho orientado a ser seguido, representando a existência de uma única sequência de execução das instruções

Pseudocódigo

- Por assemelhar-se bastante à forma em que os programas são escritos, encontra muita aceitação;
- Esta representação é suficientemente geral para permitir a tradução de um algoritmo nela representado para uma linguagem de programação específica seja praticamente direta;

```
Algoritmo <nome_do_algoritmo>
        <declaração_de_variáveis>
        <subalgoritmos>
Início
        <corpo do algoritmo>
Fim
```

Pseudocódigo

- **Algoritmo** é uma palavra que indica o início da definição de um algoritmo em forma de pseudocódigo.
- **<nome_do_algoritmo>** é um nome simbólico dado ao algoritmo com a finalidade de distingui-los dos demais.
- **<declaração_de_variáveis>** consiste em uma porção opcional onde são declaradas as variáveis globais usadas no algoritmo principal e, eventualmente, nos subalgoritmos.
- **<subalgoritmos>** consiste de uma porção opcional do pseudocódigo onde são definidos os subalgoritmos.
- **Início** e **Fim** são respectivamente as palavras que delimitam o início e o término do conjunto de instruções do corpo do algoritmo.

Pseudocódigo

```
Algoritmo Calculo_Media
Var N1, N2, MEDIA: real
Inicio
Leia N1, N2
MEDIA ← (N1 + N2) / 2
Se MEDIA >= 7 então
    Escreva "Aprovado"
Senão
    Escreva "Reprovado"
Fim_se
Fim
```

Tipos de Dados

Introdução

- Todo o trabalho realizado por um computador é baseado na manipulação das informações contidas em sua memória;
  - Instruções;
  - Dados;

Dados Numéricos

- Antes de apresentar formalmente os tipos de dados numéricos, é conveniente recordar alguns conceitos básicos relacionados à teoria dos números e conjuntos.
  - O conjunto dos números **naturais** é representado por **N** e é dado por:
  - **N** = {1, 2, 3, 4, ...}

Dados Numéricos

- Na sequência, encontramos o conjunto dos números **inteiros**:
  - **Z** = {..., -3, -2, -1, 0, 1, 2, 3, ...}
- O conjunto **Z** contém todos os elementos de **N**, bem como alguns números que não pertencem a **N** (os números negativos e o zero). Portanto, dizemos que **N** está contido em **Z**, ou então, que **Z** contém **N**.

Dados Numéricos Inteiros

- Os números **inteiros** são aqueles que não possuem componentes decimais ou fracionários, podendo ser positivos ou negativos;
  - Como exemplos de números inteiros temos:
    - 24 - número inteiro positivo
    - 0 - número inteiro
    - -12 - número inteiro negativo

Dados Numéricos Reais

- Os dados de tipo real são aqueles que podem possuir componentes decimais ou fracionários, e podem também ser positivos ou negativos;
- Exemplos de dados do tipo real:
  - 24.01 - número real positivo com duas casas decimais
  - 144. - número real positivo com zero casas decimais
  - -13.3 - número real negativo com uma casa decimal
  - 0.0 - número real com uma casa decimal
  - 0. - número real com zero casas decimais

Observe que há uma diferença entre "0", que é um dado do tipo inteiro "0." (ou "0.0") que é um dado do tipo real.

Portanto, a simples existência do ponto decimal serve para diferenciar um dado numérico do tipo inteiro de um do tipo real.

Dados Literais

- O tipo de dados literal é constituído por uma sequência de caracteres contendo letras, dígitos e/ou símbolos especiais;
- Este tipo de dados é também muitas vezes chamado de **alfanumérico**, **cadeia** (ou **cordão**) de **caracteres**, ainda, do inglês, **string**.
- Usualmente, os dados literais são representados nos algoritmos pela coleção de caracteres, delimitada em seu início e término com o caractere aspas (").

Dados Literais

- Diz-se que o dado do tipo literal possui um comprimento dado pelo número de caracteres nele contido.
- Exemplos de dados do tipo literal:
  - "QUAL ?" - literal de comprimento 6
  - " " - literal de comprimento 1
  - "qUaL ?!$" - literal de comprimento 8’
  - " AbCdefGHi" - literal de comprimento 9
  - "1-2+3=" - literal de comprimento 6
  - “0” - literal de comprimento 1

Note que, por exemplo, "1.2" representa um dado do tipo literal de comprimento 3, constituído pelos caracteres "1", "." e "2", diferindo de 1.2 que é um dado do tipo real.

Dados Lógicos

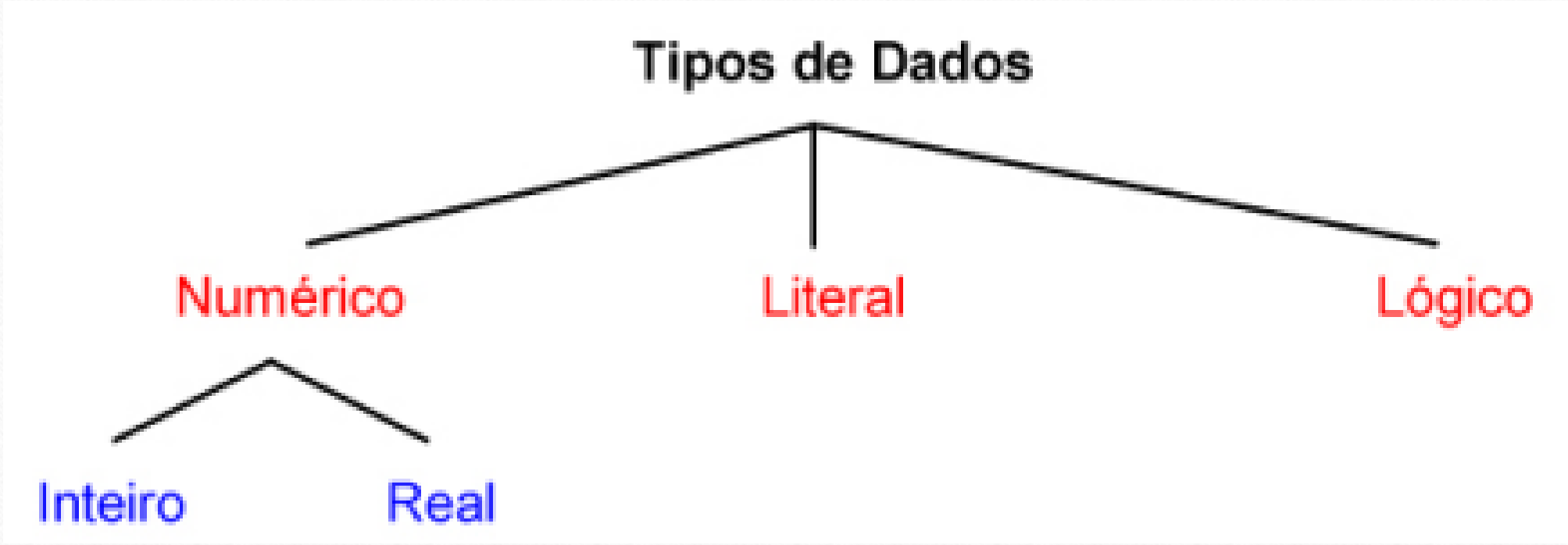
- A existência deste tipo de dado é, de certo modo, um reflexo da maneira como os computadores funcionam;
- Muitas vezes, estes tipos de dados são chamados de **booleanos**, devido à significativa contribuição de BOOLE à área da lógica matemática;
- O tipo de dados lógico é usado para representar dois únicos valores lógicos possíveis:
  - **verdadeiro** e **falso**

É comum encontrar-se em outras referências outros tipos de pares de valores lógicos como **sim/não**, **1/0**, **true/false**.

Dados Lógicos

- Nos algoritmos apresentados nesta apostila os valores lógicos serão delimitados pelo caractere ponto (.).
- Exemplo:
  - .V. - valor lógico verdadeiro
  - .F. - valor lógico falso

Síntese



Exercício

- Classifique os dados especificados abaixo de acordo com seu tipo, assinalando com **I** os dados do tipo inteiro, com **R** os reais, com **L** os literais, com **B** os lógicos (booleanos), e com **N** aqueles para os quais não é possível definir a priori um tipo de dado.

( ) 0.21	( ) .T.	( ) .V.
( ) 1	( ) +3257	( ) .V
( ) V	( ) “a”	( ) “abc”
( ) “0.”	( ) “+3257”	( ) F
( ) 1%	( ) +3257.	( ) C
( ) “José”	( ) “-0.0”	( ) Maria
( ) 0,35	( ) “.F.”	( ) +36
( ) .F.	( ) ± 3	
( ) -0.001		

Variáveis

- A todo momento durante a execução de qualquer tipo de programa os computadores estão manipulando informações representadas pelos diferentes tipos de dados descritos;

Armazenamento de Dados na Memória

- Cada um dos diversos tipos de dados apresentados, necessita de uma certa quantidade de memória para armazenar a informação representada por eles;
- Esta quantidade é função do tipo de dado considerado, do tipo da máquina (computador) e do tipo de linguagem de programação

Armazenamento de Dados do Tipo Literal

- Devemos sempre ter em mente que um byte consegue representar 256 (2^8)possibilidades diferentes;
- Uma informação do tipo literal nada mais é do que um conjunto de caracteres que podem ser letras, dígitos ou símbolos especiais;
- ASCII;
- Exemplo: a informação do tipo literal "banana" possui seis caracteres e, portanto, seis bytes são necessários para reter a referida informação na memória

Armazenamento de Dados do Tipo Literal

Endereço	Informação
0	b (98)
1	a (97)
2	n (110)
3	a (97)
4	n (110)
5	a (97)

Armazenamento de Dados do Tipo Lógico

- Uma informação do tipo lógico só possui dois valores possíveis: .V. ou .F
- Assim, a princípio, um único bit seria suficiente para armazenar uma informação deste tipo;
- Contudo, deve-se lembrar que a menor porção de memória que se pode acessar é o byte

Armazenamento de Dados do Tipo Inteiro

- O conjunto dos números inteiros (Z) contém um número infinito de elementos:
- Z = { -∞, ..., -3, -2, -1, 0, 1, 2, 3, ..., +∞}
- Obviamente é inviável o armazenamento de todos os números deste conjunto num computador
- Se apenas um byte fosse utilizado para armazenar os dados do tipo inteiro, existiriam apenas 256 números diferentes neste conjunto:

{-127, -126, ..., -2, -1, 0, 1, 2, ..., 127, 128)

Armazenamento de Dados do Tipo Real

- O conjunto dos números reais (R) contém um número infinito de elementos e, pelas mesmas razões que o conjunto dos números inteiros, precisa ser limitado;
- Para dados deste tipo julgou-se apropriado adotar quatro bytes para sua representação interna nos computadores

Conceito e Utilidade de Variáveis

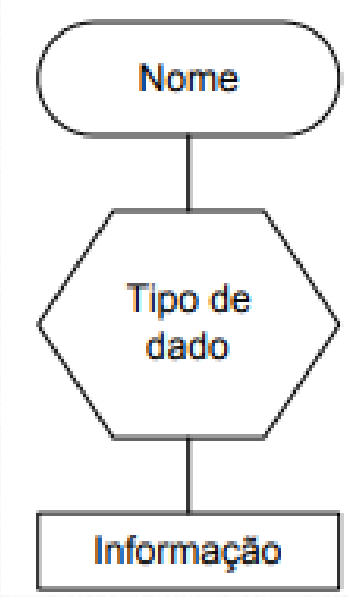
- Informações correspondentes a diversos tipos de dados são armazenadas na memória dos computadores;
- Para acessar individualmente cada uma destas informações, a princípio, seria necessário saber o tipo de dado desta informação
  - ou seja, o número de bytes de memória por ela ocupados e a posição inicial deste conjunto de bytes na memória;

Conceito e Utilidade de Variáveis

- Percebe-se que esta sistemática de acesso a informações na memória é bastante ilegível e difícil de se trabalhar.
- Para contornar esta situação criou-se o conceito de **variável**, que é uma entidade destinada a guardar uma informação.

Conceito e Utilidade de Variáveis

- Basicamente, uma variável possui três atributos: um **nome**, um **tipo de dado** associado à mesma e a **informação** por ela guardada.



Conceito e Utilidade de Variáveis

- Toda variável possui um nome que tem a função de diferenciá-la das demais.
- Cada linguagem de programação estabelece suas próprias regras de formação de nomes de variáveis

Conceito e Utilidade de Variáveis

- Adotaremos nesta apostila as seguintes regras:
  - um nome de variável deve necessariamente começar com uma letra;
  - um nome de variável não deve conter nenhum símbolo especial exceto a sublinha (_).

Conceito e Utilidade de Variáveis

- Exemplos:

SALARIO	= correto
1ANO	= errado (não começou com uma letra)
ANO1	= correto
A CASA	= errado (contém o caractere espaço em branco)
SAL/HORA	= errado (contém o caractere "/”)
SAL_HORA	= correto
_DESCONTO	= errado (não começou com uma letra)

Definição de Variáveis em Algoritmos

- Todas as variáveis utilizadas em algoritmos devem ser definidas antes de serem utilizadas;

Isto se faz necessário para permitir que o compilador reserve um espaço na memória para as mesmas!

Definição de Variáveis em Algoritmos

- Todas as variáveis utilizadas em algoritmos serão definidas no início do mesmo, por meio de um comando de uma das formas seguintes:
  - **VAR** <nome_da_variável> : <tipo_da_variável>
  - **VAR** <lista_de_variáveis> : <tipo_das_variáveis>

A palavra-chave VAR deverá estar presente sempre e será utilizada uma única vez na definição de um conjunto de uma ou mais variáveis

Numa mesma linha poderão ser definidas uma ou mais variáveis do mesmo tipo.

Para tal, deve-se separar os nomes das mesmas por vírgulas;

Variáveis de tipos diferentes devem ser declaradas em linhas diferentes

Definição de Variáveis em Algoritmos

- Exemplo de definição de variáveis:

```
VAR NOME : literal[10]
    IDADE : inteiro
    SALARIO : real
    TEM_FILHOS : lógico
```

A variável **NOME**, capaz de armazenar dados literais de comprimento 10 (dez caracteres);

A variável **IDADE**, capaz de armazenar um número inteiro;

A variável **SALARIO**, capaz de armazenar um número real;

A variável **TEM_FILHOS**, capaz de armazenar uma informação lógica.

Exercícios

1. Assinale com C os identificadores corretos e com I os incorretos. Explique o que está errado nos identificadores incorretos.

( ) valor
( ) _b248
( ) nota*do*aluno
( ) a1b2c3
( ) 3 x 4
( ) Maria

( ) km/h
( ) xyz
( ) nome empresa
( ) sala_215
( ) “nota”
( ) ah!

Exercícios

2. Supondo que as variáveis **NB**, **NA**, **NMAT** e **SX** sejam utilizadas para armazenar a nota do aluno, o nome do aluno, o número da matrícula e o sexo, declare-as corretamente, associando o tipo adequado ao dado que será armazenado.

Expressões

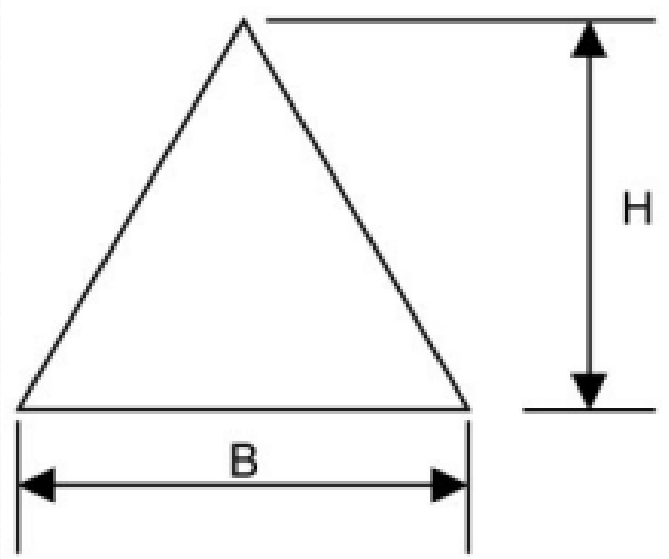
Conceito

- O conceito de expressão em termos computacionais está intimamente ligado ao conceito de expressão (ou fórmula) matemática;
- Onde um conjunto de variáveis e constantes numéricas relacionam-se por meio de operadores aritméticos compondo uma fórmula que, uma vez avaliada, resulta num valor

Conceito

- Por exemplo, a fórmula de cálculo da área do triângulo é dada por:
  - **AREA = 0.5 x B x H**

Esta fórmula utiliza três variáveis: **B** e **H**, que contêm as dimensões do triângulo, e **AREA**



Há, também, uma constante (**0.5**) e o operador de multiplicação (**x**), que aparece duas vezes na expressão

Uma **expressão** é uma combinação de variáveis, constantes e operadores, e que, uma vez avaliada, resulta num valor.

Operadores

- São elementos funcionais que atuam sobre **operandos** e produzem um determinado resultado;
- De acordo com o número de operandos sobre os quais os operadores atuam, os últimos podem ser classificados em:
  - **Binários;**
  - **Unários;**

Operadores

- Outra classificação dos operadores é feita considerando-se o tipo de dado de seus operandos e do valor resultante de sua avaliação;
- Os operadores dividem-se em **aritméticos**, **lógicos** e **literais**

Um caso especial é o dos operadores **relacionais**, que permitem comparar pares de operandos de tipos de dados iguais, resultando sempre num valor lógico.

Tipos de Expressões

- As expressões são classificadas de acordo com o tipo do valor resultante de sua avaliação.

Tipos de Expressões: Expressões Aritméticas

- São aquelas cujo resultado da avaliação é do tipo numérico, seja ele inteiro ou real;

Operador	Tipo	Operação	Prioridade
+	Binário	Adição	4
-	Binário	Subtração	4
*	Binário	Multiplicação	3
/	Binário	Divisão	3
**	Binário	Exponenciação	2
+	Unário	Manutenção de sinal	1
-	Unário	Inversão de sinal	1

Somente o uso de operadores aritméticos e variáveis numéricas é permitido em expressões deste tipo

Tipos de Expressões: Expressões Aritméticas

- Nos exemplos seguintes, assumiremos que:
  - **A**, **B** e **C** são variáveis do tipo inteiro;
  - **X**, **Y** e **Z** são variáveis do tipo real
- Exemplos:
  - A + B * C = expressão de resultado inteiro
  - A + B + Y = expressão de resultado real
  - A / B = expressão de resultado real
  - X / Y = expressão de resultado real

Tipos de Expressões: Expressões Lógicas

- São aquelas cujo resultado da avaliação é um valor lógico (.V. ou .F.).

Operador	Tipo	Operação	Prioridade
.OU.	Binário	Disjunção	3
.E.	Binário	Conjunção	2
.NÃO.	Unário	Negação	1

Existem outros operadores lógicos, como por exemplo o **OU_EXCLUSIVO**., mas suas funções podem ser exercidas por combinações dos três tipos de operadores

Tipos de Expressões: Expressões Lógicas

- Para exemplificar o uso de operadores lógicos:

A	B	.NÃO. A	.NÃO. B	A .OU. B	A .E. B
.F.	.F.	.V.	.V.	.F.	.F.
.F.	.V.	.V.	.F.	.V.	.F.
.V.	.F.	.F.	.V.	.V.	.F.
.V.	.V.	.F.	.F.	.V.	.V.

Tipos de Expressões: Expressões Lógicas

- Há, ainda, outro tipo de operador que pode aparecer em operações lógicas: os operadores **relacionais**:

Operador	Operação
=	Igual
<>	Diferente
<	Menor
<=	Menor ou igual
>	Maior
>=	Maior ou igual

Tipos de Expressões: Expressões Literais

- São aquelas cujo resultado da avaliação é um valor literal;
- Bem menos frequente que os anteriores;
- Os tipos de operadores existentes variam de uma linguagem de programação para outra, não havendo uma padronização.

Exercício

1. Dada a declaração de variáveis:

VAR A, B, C : **inteiro**
X, Y, Z : **real**
NOME, RUA : **literal**[20]
L1, L2 : **lógico**

Classifique as expressões seguintes de acordo com o tipo de dado do resultado de sua avaliação, em **I** (inteiro), **R** (real), **L** (literal), **B** (lógico) ou **N** (quando não for possível defini-lo):

Exercício

( ) A + B + C	( ) L1 .OU. L2	( ) (A = B)
( ) A + B + Z	( ) RUA <> NOME	( ) X + Y / Z
( ) NOME + RUA	( ) A + B / C	( ) X = Z / A
( ) A B	( ) A + X / Z	( ) L1 ** L2
( ) A Y	( ) A + Z / A	( ) A + B / L2
( ) NOME RUA	( ) A B = L1	( ) X < L1 / RUA

Exercício

2. Para as mesmas variáveis declaradas no exercício 1, às quais são dados os valores seguintes:

A = 1	**Z** = -1.0
B = 2	**L1** = .V.
C = 3	**NOME** = "PEDRO"
X = 2.0	**RUA** = "PEDRINHO"
Y = 10.0	**L2** = .F.

Determine o resultado da avaliação das expressões

Exercício

A + C / B → _____

A + B + C →_____

C / B / A → _____

X ** B →_____

- (X ** B) →_____

(-X) ** B →_____

NOME + RUA →_____

NOME = RUA →_____

L1 .OU. L2 →_____

(L1 .E.(.NÃO. L2)) →_____

(L2 .E. (.NÃO. L1)) →_____

(L1 .E. (.NÃO. L2)) .OU. (L2 .E. (.NÃO. L1)) →_____

Exercício

X Y .E. C = B →_____

(C - 3 * A) (X + 2 * Z) →_____

Instruções Primitivas

Instruções Primitivas

- São os comandos básicos que efetuam tarefas essenciais para a operação dos computadores;
  - Entrada e saída de dados
  - Movimentação dos mesmos na memória
- Estão presentes na absoluta maioria das linguagens de programação

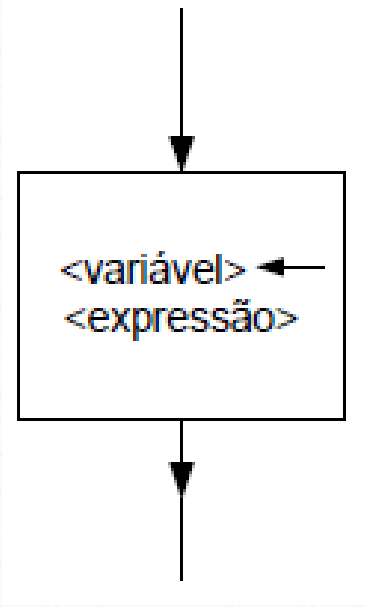
Programa que não utiliza nenhuma instrução primitiva é incapaz de se comunicar com o mundo exterior e, portanto, não tem utilidade alguma

Instruções Primitivas

- Algumas definições:
  - **Dispositivo de entrada;**
  - **Dispositivo de saída;**
  - **Sintaxe;**
  - **Semântica;**

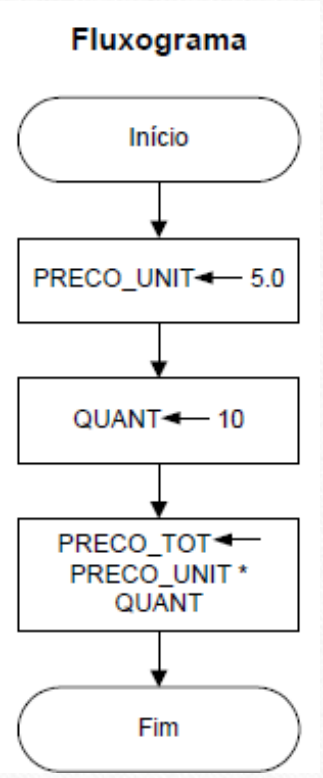
Instrução Primitiva de Atribuição

- **Instrução primitiva** de atribuição, ou simplesmente **atribuição**, é a principal maneira de se armazenar uma informação numa variável. Sua sintaxe é:
  - **<nome_de_variável> ← <expressão>**



Instrução Primitiva de Atribuição

Fluxograma



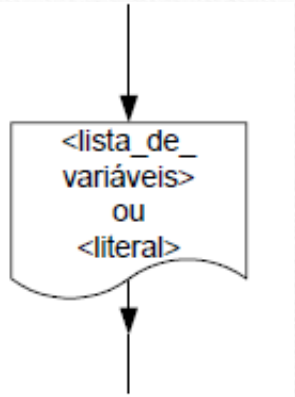
Pseudocódigo

```
Algoritmo EXEMPLO_6.1
Var  PRECO_UNIT, PRECO_TOT : real
     QUANT : inteiro
Início
     PRECO_UNIT ← 5.0
     QUANT ← 10
     PRECO_TOT ← PRECO_UNIT * QUANT
Fim.
```

Instrução Primitiva de Saída de Dados

- São o meio pelo qual informações contidas na memória dos computadores são colocadas nos dispositivos de saída
  - Visualização do Usuário;
- Há duas sintaxes possíveis para esta instrução:
  - **Escreva** <lista de variáveis>

  ou
  - **Escreva** <literal>

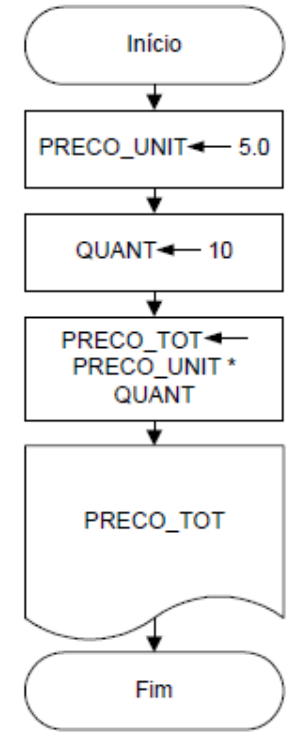


Daqui por diante, **Escreva** será considerada uma palavra reservada e não mais poderá ser utilizada como nome de variável.

Uma **lista_de_variáveis** é um conjunto de nomes de variáveis separados por vírgulas. Um **literal** é simplesmente um dado do tipo literal delimitado por aspas.

Instrução Primitiva de Saída de Dados

Fluxograma



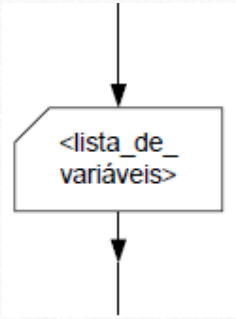
Pseudocódigo

```
Algoritmo EXEMPLO_6.4
Var  PRECO_UNIT, PRECO_TOT : real
     QUANT : inteiro
Início
     PRECO_UNIT ← 5.0
     QUANT ← 10
     PRECO_TOT ← PRECO_UNIT * QUANT
     Escreva PRECO_TOT
Fim.
```

A semântica da instrução primitiva de saída de dados é muito simples: os argumentos do comando são enviados para o dispositivo de saída;

Instrução Primitiva de Entrada de Dados

- O algoritmo anterior ainda necessita de uma melhoria essencial
- Toda vez que ele é executado, o mesmo valor é calculado
- Seria interessante que estes valores pudessem ser fornecidos ao computador pelo usuário do programa toda vez que o programa fosse executado
- A instrução primitiva de entrada de dados foi criada para suprir esta necessidade.
  - Sua sintaxe é:
  - Leia <lista_de_variáveis>

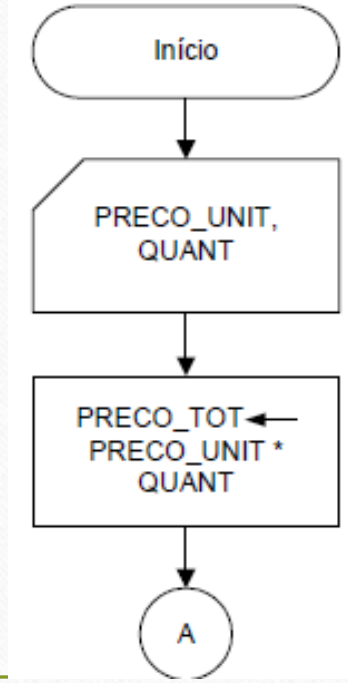


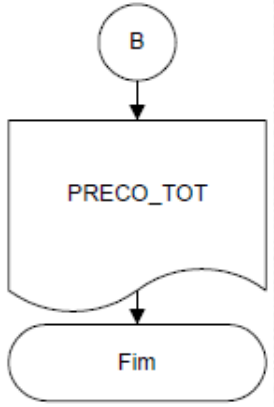
Daqui em diante **Leia** será tratada como uma palavra reservada e não mais poderá ser usada como nome de variável em algoritmos

Instrução Primitiva de Entrada de Dados

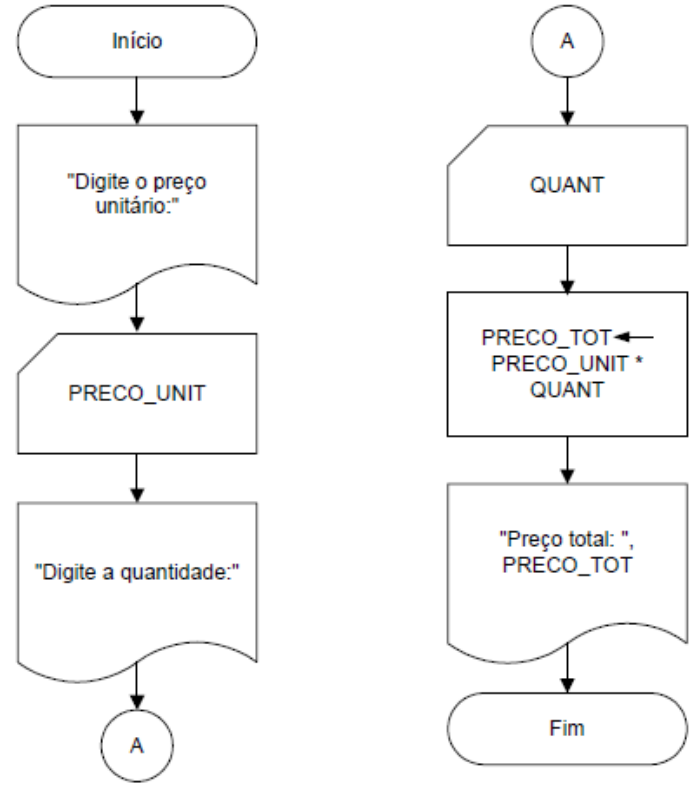
- A semântica da instrução de entrada (ou leitura) de dados é, de certa forma, inversa à da instrução de escrita

```
Algoritmo EXEMPLO_6.6
Var  PRECO_UNIT, PRECO_TOT : real
     QUANT : inteiro
Início
     Leia PRECO_UNIT, QUANT
     PRECO_TOT ← PRECO_UNIT * QUANT
     Escreva PRECO_TOT
Fim.
```





Instrução Primitiva de Entrada de Dados




```
Algoritmo EXEMPLO_6.7
Var PRECO_UNIT, PRECO_TOT : real
    QUANT : inteiro
Início
    Escreva "Digite o preço unitário: "
    Leia PRECO_UNIT
    Escreva "Digite a quantidade: "
    Leia QUANT
    PRECO_TOT ← PRECO_UNIT * QUANT
    Escreva "Preço total: ", PRECO_TOT
Fim.
```

Exercícios Avaliativos

Controle de Fluxo de Execução

- Até o momento os algoritmos estudados utilizam apenas instruções primitivas de atribuição, e de entrada e saída de dados;
- No entanto, na prática muitas vezes é necessário executar ações diversas em função dos dados fornecidos ao algoritmo

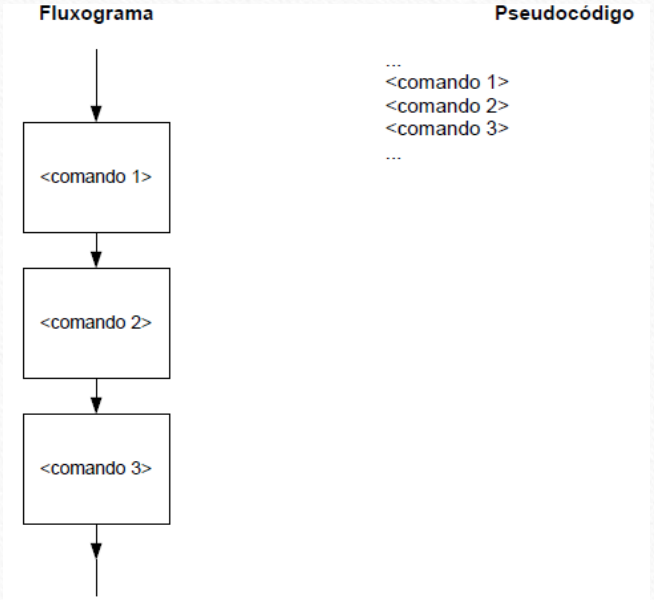
- De acordo com o modo como este controle é feito, estas estruturas são classificadas em:
  - estruturas sequenciais;
  - estruturas de decisão;
  - estruturas de repetição

Comandos Compostos

- Um **comando composto** é um conjunto de zero ou mais comandos (ou instruções) simples, como atribuições e instruções primitivas de entrada ou saída de dados;

Estrutura Sequencial

- Na estrutura sequencial os comandos de um algoritmo são executados numa sequência pré-estabelecida. Cada comando é executado somente após o término do comando anterior.



Estruturas de Decisão

- Neste tipo de estrutura o fluxo de instruções a ser seguido é escolhido em função do resultado da avaliação de uma ou mais condições.
- Uma **condição** é uma expressão lógica

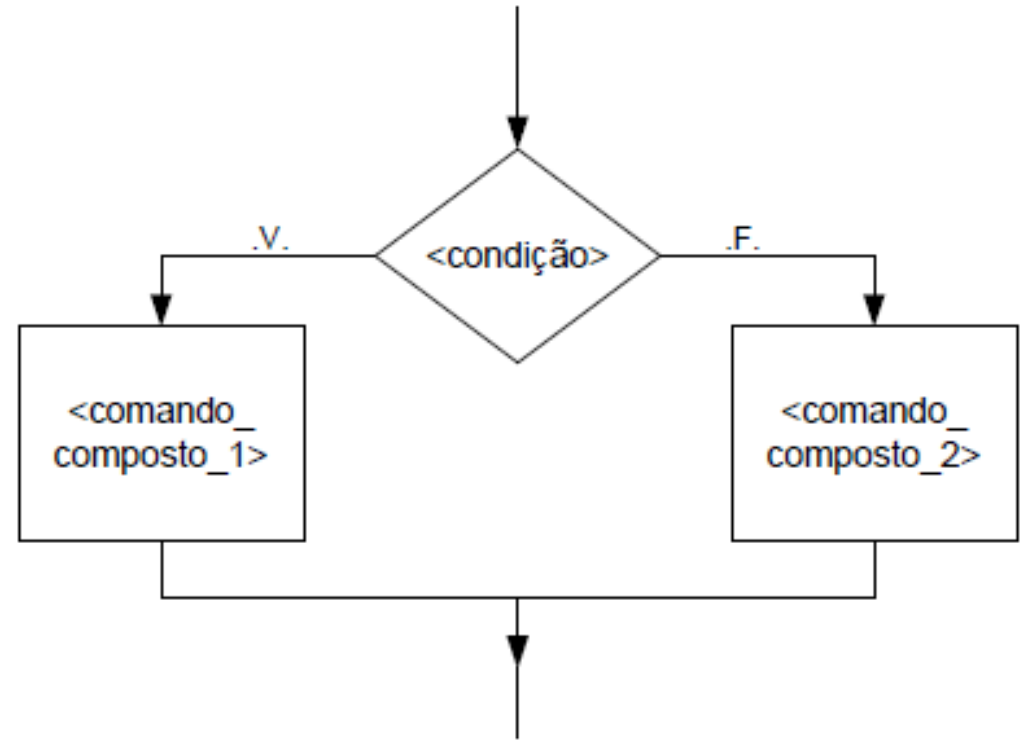
Estruturas de Decisão

- A classificação das estruturas de decisão é feita de acordo com o número de condições que devem ser testadas para que se decida qual o caminho a ser seguido. Segundo esta classificação, têm-se dois tipos de estruturas de decisão:
  - Se
  - Escolha

Estruturas de Decisão: Tipo Se

- Nesta estrutura uma única condição (expressão lógica) é avaliada. Se o resultado desta avaliação for verdadeiro (V), então um determinado conjunto de instruções (comando composto) é executado.
- Caso contrário, ou seja, quando o resultado da avaliação for falso (F), um comando diferente é executado.

Estruturas de Decisão: Tipo Se



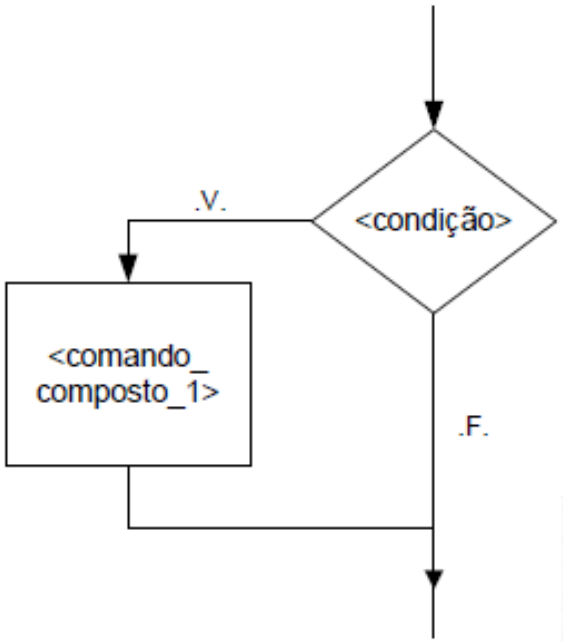
Pseudocódigo

Se <condição>
    Então
        <comando_composto_1>
    Senão
        <comando_composto_2>
Fim_se

Note-se o aparecimento de novas palavras-reservadas **Se**, **Então**, **Senão** e **Fim_se**.

Estruturas de Decisão: Tipo Se

Fluxograma



Pseudocódigo

```
Se <condição>
    Então
        <comando_composto_1>
Fim_se
```

Há casos particulares e muito comuns desta construção, onde o comando composto 2 é um conjunto vazio de instruções.

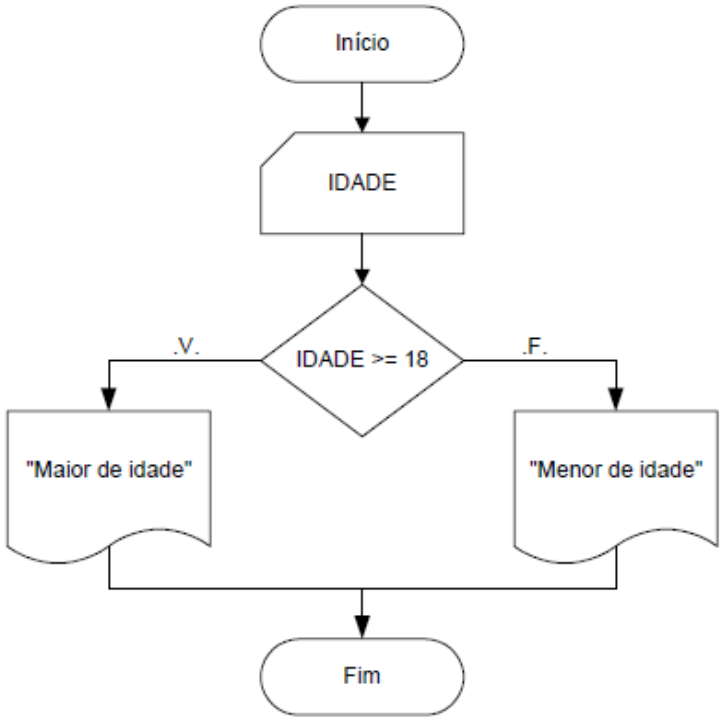
Neste caso, a porção relativa ao **Senão** pode ser omitida

Estruturas de Decisão: Tipo Se

- A semântica desta construção é a seguinte: no caso da condição ser verdadeira, o **Comando_composto_1** é executado e, após seu término, o fluxo de execução prossegue pela próxima instrução após o **Fim_se**.
- Quando a condição é falsa, o fluxo de execução prossegue normalmente pela primeira instrução após o **Fim_se**.

Estruturas de Decisão: Tipo Se

Fluxograma



Pseudocódigo

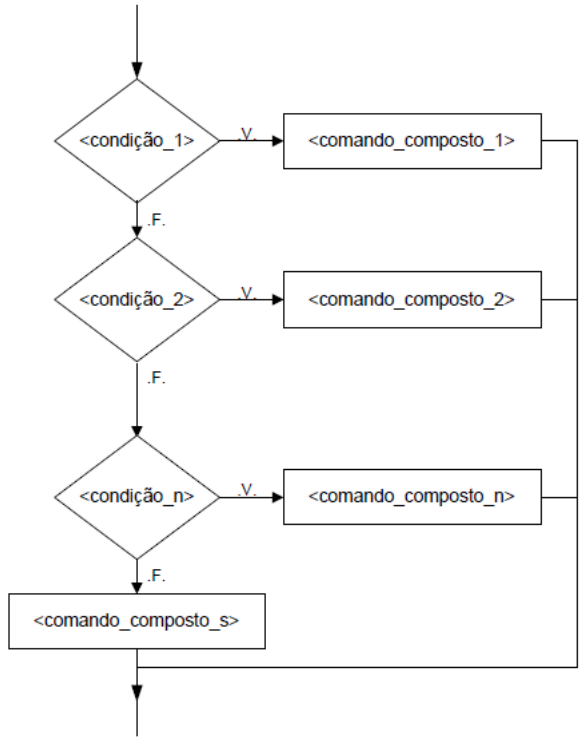
```
Algoritmo Exemplo_7.4
Var  IDADE : inteiro
Início
      Leia IDADE
      Se IDADE >= 18
            Então
               Escreva "Maior de idade"
            Senão
               Escreva "Menor de idade"
      Fim_se
Fim.
```

Estruturas de Decisão: Tipo Escolha

- Este tipo de estrutura é uma generalização da estrutura Se, onde somente uma condição era avaliada e dois caminhos podiam ser seguidos.
- Na estrutura de decisão do tipo Escolha pode haver uma ou mais condições a serem testadas e um comando composto diferente associado a cada uma destas.

Estruturas de Decisão: Tipo Escolha

Fluxograma

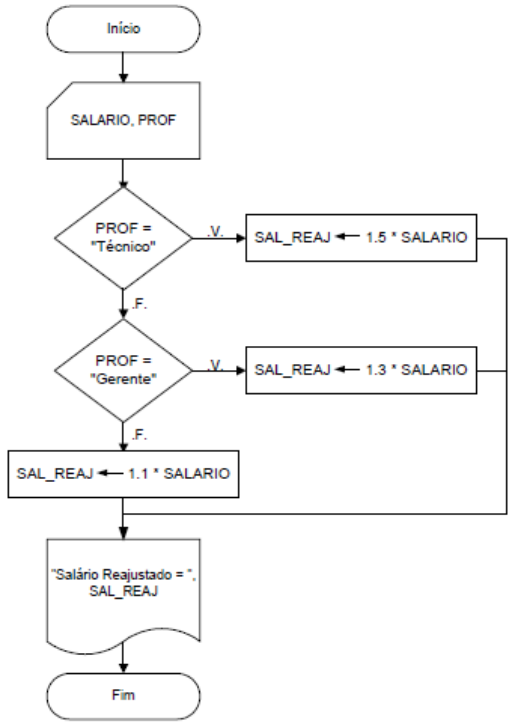


Pseudocódigo

```
Escolha
      Caso <condição_1>
         <comando_composto_1>
      Caso <condição_2>
         <comando_composto_2>
      Caso <condição_n>
         <comando_composto_n>
      Senão
         <comando_composto_s>
Fim_escolha
```

Estruturas de Decisão: Tipo Escolha

Fluxograma



Pseudocódigo

```
Algoritmo Exemplo_7.6
Var  SALARIO, SAL_REAJ : real
     PROF : literal[20]
Início
     Leia SALARIO, PROF
     Escolha
          Caso PROF = "Técnico"
            SAL_REAJ ← 1.5 * SALARIO
          Caso PROF = "Gerente"
            SAL_REAJ ← 1.3 * SALARIO
          Senão
            SAL_REAJ ← 1.1 * SALARIO
     Fim_escolha
     Escreva "Salário Reajustado = ", SAL_REAJ
Fim.
```

Exercícios

- Calcule a média aritmética das 3 notas de um aluno e mostre, além do valor da média, uma mensagem de "Aprovado", caso a média seja igual ou superior a 6, ou a mensagem "reprovado", caso contrário;
- Elaborar um algoritmo que lê 3 valores a,b,c e os escreve. A seguir, encontre o maior dos 3 valores e o escreva com a mensagem : "É o maior ".
- Elaborar um algoritmo que lê 2 valores a e b e os escreve com a mensagem: "São múltiplos" ou "Não são múltiplos".

Estruturas de Repetição

- São muito comuns as situações em que se deseja repetir um determinado trecho de um programa certo número de vezes;
- Por exemplo, pode-se citar o caso em que se deseja realizar um mesmo processamento para conjuntos de dados diferentes;
- Exemplo: processamento de folha de pagamentos de uma empresa em que o mesmo cálculo é efetuado para cada um dos funcionários.

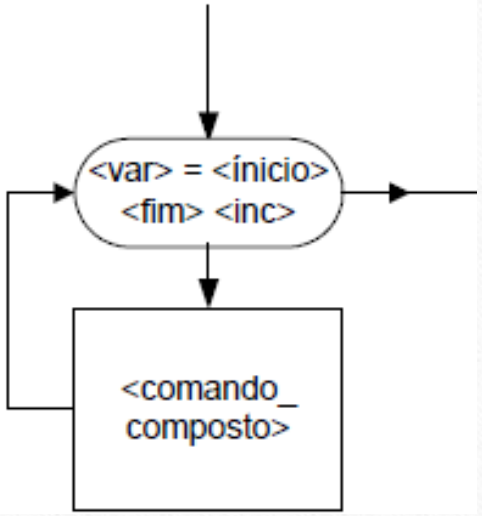
As estruturas de repetição são muitas vezes chamadas de **Laços** ou, também, de **Loops**.

- A classificação das estruturas de repetição é feita de acordo com o conhecimento prévio do número de vezes que o conjunto de comandos será executado. Assim, os laços dividem-se em:
  - **laços contados**, quando se conhece previamente quantas vezes o comando composto no interior da construção será executado;
  - **laços condicionais**, quando não se conhece de antemão o número de vezes que o conjunto de comandos no interior do laço será repetido, pelo fato de o mesmo estar amarrado a uma condição sujeita à modificação pelas instruções do interior do laço.

Laços Contados

- São úteis quando se conhece previamente o número de vezes que se deseja executar um determinado conjunto de comandos

Fluxograma



Pseudocódigo

Para <var> **de** <início> **até** <final> **incr de** <inc> **faça**
    <comando_composto>
Fim_para

Laços Condicionais

- São aqueles cujo conjunto de comandos em seu interior é executado até que uma determinada condição seja satisfeita;
- Ao contrário do que acontece nos laços contados, nos laços condicionais não se sabe de antemão quantas vezes o corpo do laço será executado.

Laços Condicionais

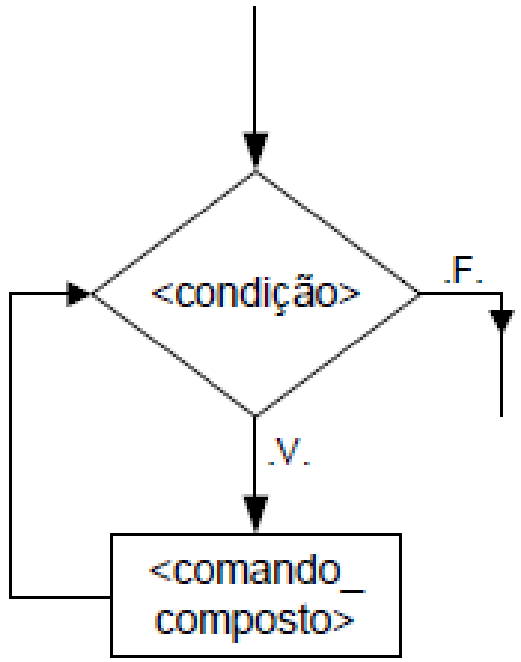
- As construções que implementam laços condicionais mais comuns nas linguagens de programação modernas são:
  - Enquanto
  - Repita

Laços Condicionais: Construção Enquanto

- Sua semântica é a seguinte: ao início da construção Enquanto a condição é testada.
- Se seu resultado for falso, então o comando composto no seu interior não é executado e a execução prossegue normalmente pela instrução seguinte ao Fim_enquanto.

Laços Condicionais: Construção Enquanto

Fluxograma



Pseudocódigo

Enquanto <condição> **faça**
    <comando_composto>
Fim_enquanto

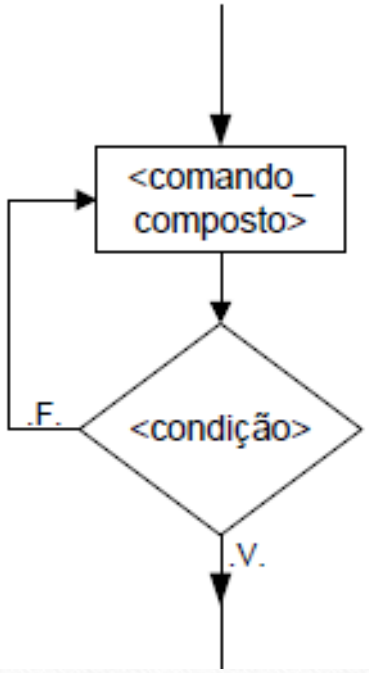
Uma vez dentro do corpo do laço, a execução somente abandonará o mesmo quando a condição for falsa

Laços Condicionais: Construção Repita

- Seu funcionamento é bastante parecido ao da construção **Enquanto**;
- O comando é executado uma vez
- A condição é testada: se ela for falsa, o comando composto é executado novamente e este processo é repetido até que a condição seja verdadeira

Laços Condicionais: Construção Repita

Fluxograma



Pseudocódigo

```
Repita
        <comando_composto>
Até que <condição>
```

Esta construção difere da construção Enquanto pelo fato de o comando composto ser executado uma ou mais vezes (pelo menos uma vez), ao passo que na construção Repita o comando composto é executado zero ou mais vezes (possivelmente nenhuma)

Exercício

- Escrever um algoritmo que lê 5 valores para a, um de cada vez, e conta quantos destes valores são negativos, escrevendo esta informação.

- Desenvolver um algoritmo que efetue a soma de todos os números ímpares que são múltiplos de três e que se encontram no conjunto dos números de 1 até 500