YTO NIHON
EDUCAÇÃO LEVADA A SÉRIO

Lógica de Programação

Construindo Algoritmos com o VisuAlg

Sobre Este Curso

Público Alvo

Interessados em dar o primeiro passo rumo à formação na área de Desenvolvimento de Sistemas e Aplicativos.

Pré-Requisitos

Conhecimentos básicos do ambiente Windows.

Fevereiro | 2022

Índice

	Anotações

Anotações	

Anotações

Anotações

Copyright

As informações contidas neste material se referem à ***Lógica de Programação***, e estão sujeitas as alterações sem comunicação prévia, não representando um compromisso por parte do autor em atualização automática de futuras versões.

A ***Yto Nihon*** não será responsável por quaisquer erros ou por danos acidentais ou consequenciais relacionados com o fornecimento, desempenho, ou uso desta apostila ou os exemplos contidos aqui.

Os exemplos de empresas, organizações, produtos, nomes de domínio, endereços de e-mail, logotipos, pessoas, lugares e eventos aqui representados são fictícios. Nenhuma associação a empresas, organizações, produtos, nomes de domínio, endereços de e-mail, logotipos, pessoas, lugares ou eventos reais é intencional ou deve ser inferida.

A reprodução, adaptação, ou tradução deste manual mesmo que parcial, para qualquer finalidade é proibida sem autorização prévia por escrito da ***Yto Nihon***, exceto as permitidas sob as leis de direito autoral.

Texto original registrado no Escritório de Direitos Autorais da Fundação Biblioteca Nacional.

Equipe

Coordenação

- Jorge Barros

Conteúdos

- Jorge Barros

Revisão

- Jorge Barros

Histórico das edições

Edição	Idioma	Edição
1ª	Português	Agosto de 2013
2ª	Português	Setembro de 2013
3ª	Português	Novembro de 2013
4ª	Português	Fevereiro de 2014
5ª	Português	Julho de 2014

Anotações

Convenções

Fontes

As convenções a seguir são usadas nos materiais do curso para diferenciar os elementos do texto.

Convenção	Uso
Negrito	Representa comandos, opções de comando e sintaxe que devem ser digitados exatamente conforme mostrado. Também indica comandos em menus e botões, títulos e opções de caixas de diálogo e nomes de ícones e menus.
Itálico	Em instruções de sintaxe ou texto descritivo, indica nomes de argumentos ou espaços reservados para informações variáveis. O recurso de itálico também é usado para apresentar novos termos, para títulos de capítulo e para dar ênfase ao texto.
Maiúsculas	Indicam nomes de domínios, nomes de usuários, nomes de computadores, nomes de diretórios, nomes de pastas e de arquivos, exceto quando estiverem se referindo especificamente a nomes que fazem distinção entre maiúsculas e minúsculas. A menos que seja indicado de outra forma, você pode usar letras minúsculas ao digitar um nome de diretório ou arquivo em uma caixa de diálogo ou em um prompt de comando.
TODAS MAIÚSCULAS	Indica os nomes das teclas, sequências de teclas e combinações de teclas—por exemplo, ALT+BARRA DE ESPAÇOS.
`Monoespaçada`	Representa exemplos de códigos ou de textos de tela.
[ ]	Em elementos de sintaxe, engloba itens opcionais. Por exemplo, [nomedearquivo] na sintaxe de comando indica que você pode digitar um nome de arquivo com o comando. Digite apenas as informações entre colchetes, não os colchetes.
{ }	Em elementos de sintaxe, engloba itens obrigatórios. Digite apenas as informações entre chaves, não as chaves.
|	Em instruções de sintaxe, separa uma escolha do tipo ou/ou.
▸	Indica um procedimento com etapas sequenciais.
...	Em instruções de sintaxe, especifica que o item anterior pode ser repetido.
⋮	Representa uma parte omitida de um exemplo de código.

Configuração Regional

As configurações regionais e de idioma assumidas nesta apostila são: Português (Brasil).

Versão

A versão utilizada para este curso é o **VisuAlg 2.0**

Anotações	

Capítulo 01 - Introdução à Lógica de Programação

Objetivos

Neste capítulo você irá aprender:

- Algoritmo
- Programa
- Etapas de um Programa
- Formas de Representação de Algoritmos
- Descrição Narrativa Fluxograma
- Algoritmo
- Ciclo de Vida de um Programa
- Critérios de Qualidade de um Programa

Exercícios

Os exemplos e exercícios utilizados neste capítulo estão na pasta: **Capítulo 01 - Introdução à Lógica de Programação.**

Duração

Este capítulo levará aproximadamente 45 minutos

	Anotações

Algoritmo

al.go.rit.mo
sm (ár al-Huwârizmî)

1 Operação ou processo de cálculo.

2 Sequência de passos que visam atingir um objetivo bem definido.

Apesar de achar este nome estranho, algoritmos são comuns em nosso cotidiano, como, por exemplo, uma ***receita de bolo***. Nela está descrita uma série de ingredientes necessários, uma sequência de diversos passos - ações - a serem cumpridos para que se consiga fazer determinado tipo de bolo - que é um objetivo bem definido.

Em geral um algoritmo destina-se a resolver um problema: fixa um padrão de comportamento a ser seguido, uma norma de execução a ser trilhada, com o objetivo de alcançar a solução de um problema.

Programa

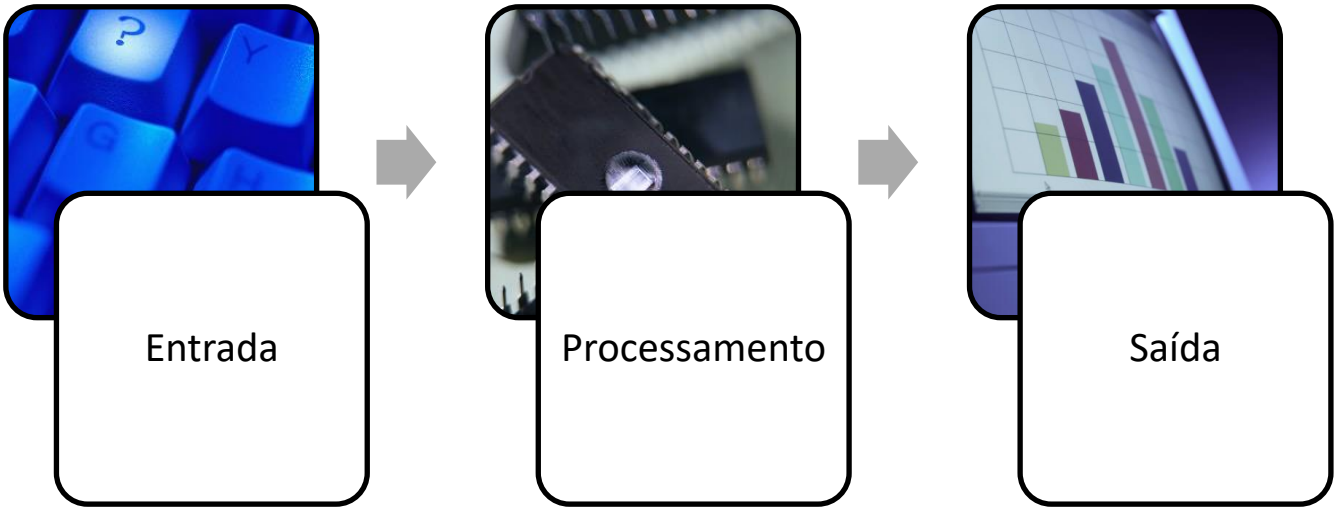
pro.gra.ma
sm (gr prógramma)

1 Processo de automação. Uma tarefa deixa de ser desempenhada pelo homem e passa a ser realizada por máquinas, sejam estes dispositivos mecânicos, eletrônicos (como os computadores) ou de natureza mista.

Para que a automação de uma tarefa seja bem-sucedida é necessário que a máquina que passará a realizá-la seja capaz de desempenhar cada uma das etapas constituintes do processo a ser automatizado com eficiência, de modo a garantir a repetibilidade do mesmo.

Anotações

Etapas de um Programa



Entrada (Input)

Refere-se a algum dado de entrada do processamento, são valores onde o processo irá atuar. Como por exemplo, um arquivo enviado para um compressor de dados.

Processamento

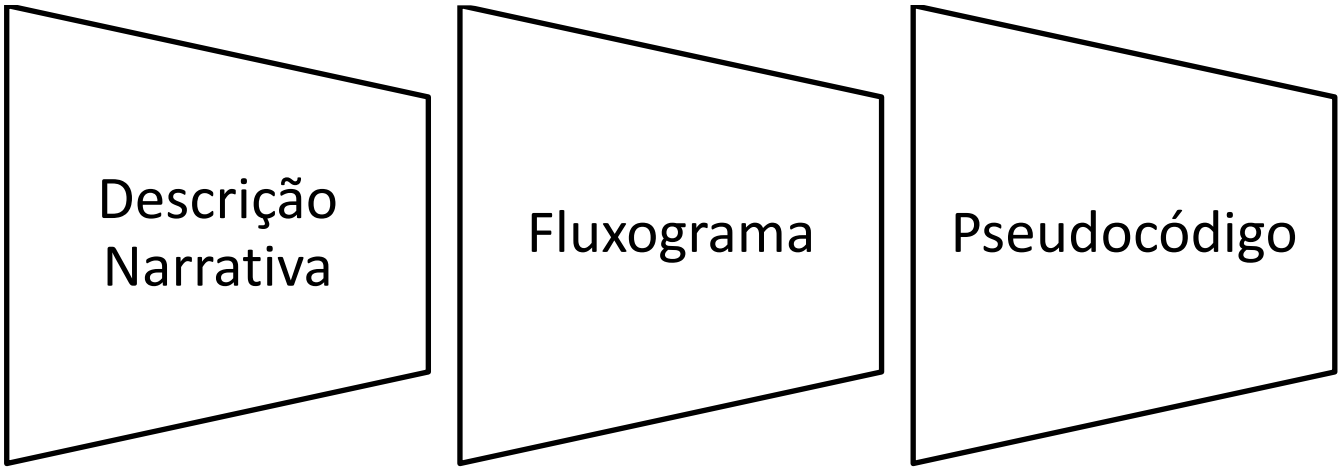
É onde os dados de entrada serão processados para gerar um determinado resultado. O computador executa o arquivo. (Outros exemplos: o cálculo salarial, uma complexa expressão matemática, ou até mesmo uma simples movimentação de dados ou comparação entre eles). No caso do processamento computadorizado esta tarefa é realizada por meio de um algoritmo escrito numa linguagem de programação que é compilado e gera o código de um programa responsável pelo processamento.

Saída (Output)

É simplesmente o resultado de todo o processamento, em todo processamento temos dados gerados como resultado, essas saídas, podem ser impressas na tela, em papel, armazenadas em um arquivo, ou até mesmo servir como entrada para outro processo. O computador exibe os resultados obtidos na tela.

	Anotações

Formas de Representação de Algoritmos



Descrição Narrativa

Troca de um Pneu Furado	Tomando um Banho
Afrouxar ligeiramente as porcas	Entrar no banheiro e tirar a roupa
Suspender o carro	Abrir a torneira do chuveiro
Retirar as porcas e o pneu	Entrar na água
Colocar o pneu reserva	Ensaboar-se
Apertar as porcas	Sair da água
Abaixar o carro	Fechar a torneira
Dar o aperto final nas porcas	Enxugar-se
	Vestir-se

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".

Vantagens

- O português é bastante conhecido por nós;

Desvantagens

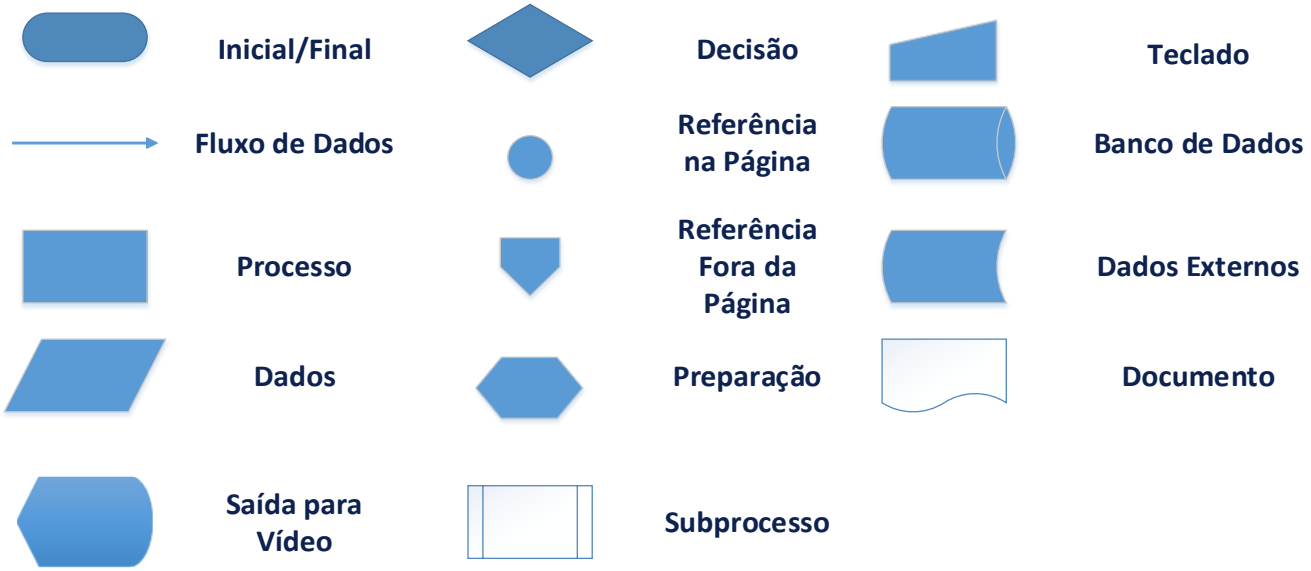
- Imprecisão;
- Pouca confiabilidade (a imprecisão acarreta a desconfiança);
- Extensão (normalmente, escreve-se muito para dizer pouca coisa).

Anotações

Fluxograma

É uma ferramenta usada e desenvolvida pelos profissionais de análise de sistemas de informação. Tem como finalidade descrever o fluxo, manual ou mecânico, especificando os suportes usados para os dados e as informações. Utiliza-se de símbolos convencionais (norma ISO 5807:1985), os quais indicarão símbolos de entrada de dados, do processamento de dados e da saída de dados.

Símbolos

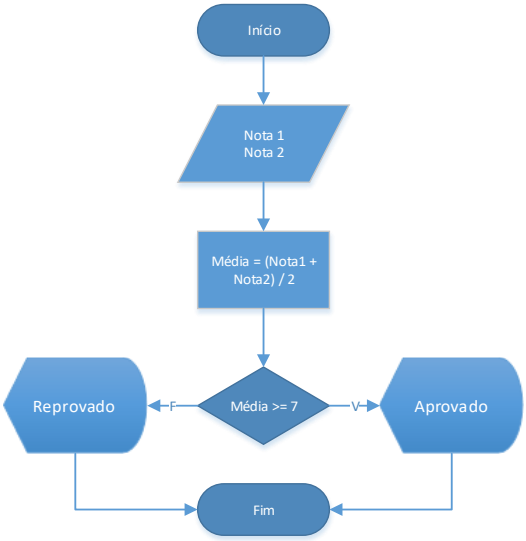


Esta forma é aproximadamente intermediária à descrição narrativa e ao pseudocódigo, pois é menos imprecisa que a primeira e, no entanto, não se preocupa com detalhes de implementação do programa, como o tipo das variáveis usadas.

Nota-se que os fluxogramas convencionais se preocupam com detalhes de nível físico da implementação do algoritmo. Por exemplo, figuras geométricas diferentes são adotadas para representar operações de saída de dados realizadas em dispositivos distintos, como uma fita magnética ou um monitor de vídeo.

Anotações

Exemplo



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.

Isto pode ser melhor visualizado pelo fato de que, apesar de vários caminhos poderem convergir para uma mesma figura do diagrama, há sempre um único caminho saindo desta. Exceções a esta regra são os símbolos finais, dos quais não há nenhum fluxo saindo, e os símbolos de decisão, de onde pode haver mais de um caminho de saída (usualmente dois caminhos), representando uma bifurcação no fluxo.

Vantagens

- Uma das ferramentas mais conhecidas;
- Figuras dizem muito mais que palavras;
- Padrão mundial

Desvantagens

- Faz com que a solução do problema já esteja amarrada a dispositivos físicos;
- Pouca atenção aos dados, não oferecendo recursos para descrevê-los ou representá-los;
- Complica-se à medida que o algoritmo cresce.

Anotações

Algoritmo

Estrutura de Algoritmo

```
ALGORITMO "SomaDeDoisValores"

VAR
Soma, A, B : INTEIRO

INICIO
ESCREVA("Digite o 1° Numero : ")
LEIA(A)

ESCREVA("Digite o 2° Numero : ")
LEIA(B)

Soma := a + B
ESCREVA(A, " +", B, " =", Soma)

FIMALGORITMO
```

De forma semelhante como os programas são escritos. Também chamado de Portugol ou Português Estruturado.

A linguagem de Programação mais próxima é o Pascal, com codificação quase idêntica ao Inglês Estruturado.

Esta forma de representação de algoritmos é rica em detalhes, como a definição dos tipos das variáveis usadas no algoritmo. Por assemelhar-se bastante à forma em que os programas são escritos, encontra muita aceitação.

Na verdade, está representação é suficientemente geral para permitir a tradução de um algoritmo nela representado para uma linguagem de programação específica seja praticamente direta.

Algoritmo é uma palavra que indica o início da definição de um algoritmo em Portugol ou Pseudocódigo.

Vantagens

- Independência física da solução (solução lógica apenas);
- Usa o português como base;
- Pode-se definir quais e como os dados vão estar estruturados;
- Passagem quase imediata do algoritmo para uma linguagem de programação qualquer.

	Anotações

Desvantagens

- Exige a definição de uma linguagem não real para trabalho;
- Não padronizado. Pode ter certas diferenças dependo do autor/programador.

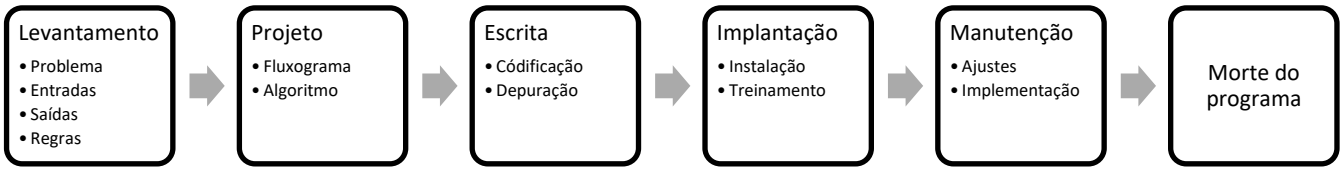
Teste de Mesa

A	B	Soma
3	5	8

O teste de mesa simula a execução de um algoritmo sem utilizar o computador, empregando apenas papel e caneta. Para isso, você deve orientar-se por meio dos passos descritos a seguir:

- Identifique as variáveis envolvidas em seu algoritmo;
- Crie uma tabela com linhas e colunas, onde corresponde ao número de instruções observadas pelo teste de mesa e é o número de variáveis envolvidas. Utilize a primeira coluna para identificar os números das linhas correspondentes às instruções observadas e identifique as demais colunas com o nome de uma variável;
- De cima para baixo, preencha cada uma das linhas da tabela com o número da linha que identifica cada instrução, seguido dos valores assumidos pelas variáveis do programa após a execução daquela instrução. Para indicar que o valor de uma variável foi lido, envolva-o entre parênteses: se o valor foi escrito pela instrução, envolva-o entre chaves: para valores indefinidos, isto é, aqueles que ainda não foram determinados até uma dada instrução, utilize a interrogação.

Ciclo de Vida de um Programa



Como tudo na terra, o programa tem um tempo de vida, chamado de ciclo de vida de um programa.

Levantamento

É a fase onde definimos o problema a ser resolvido utilizando um computador. Nesta fase relacionamos a entrada e a saída do futuro programa, assim como a definição dos arquivos auxiliares que ele venha a utilizar.

Problema

Determinar o que se quer resolver ou qual objetivo a ser atingido.

Entradas

Informações fornecidas, a partir das quais se desenvolverão os cálculos.

	Anotações

Saídas
As informações a serem geradas como resultado.

Regras
Mapear as regras e limitações do problema ou das limitações do agente executante (exemplo: se o agente fosse uma calculadora não-científica, iriam existir limitações no cálculo de funções, por exemplo).

Projeto
É a fase onde a resolução do problema é concebida. Neste ponto são definidos detalhes do algoritmo, estrutura de dados empregados pelo programa.

Fluxograma
Diagramação das rotinas.

Algoritmo
Redação em português estruturado das rotinas.

Escrita

Codificação
Consiste em codificar o programa em uma linguagem de programação apropriada.

Depuração
Ao final da escrita estaremos com o programa quase pronto; mas será que ele funciona? Esta é a fase onde se corrigem os erros, até que o objetivo seja alcançado.

Implantação
O programa é disponibilizado ao cliente para utilização.

Instalação
Instalação do sistema na estação e/ou no servidor.

Treinamento
Treinamento dos usuários do sistema instalado.

Manutenção
O programa poderá ser alterado, tanto para corrigir erros não encontrados na fase de depuração / implantação, como também para atender novas necessidades a serem solicitadas pelos usuários.

Ajustes
Ajustes solicitados pelos usuários.

	Anotações

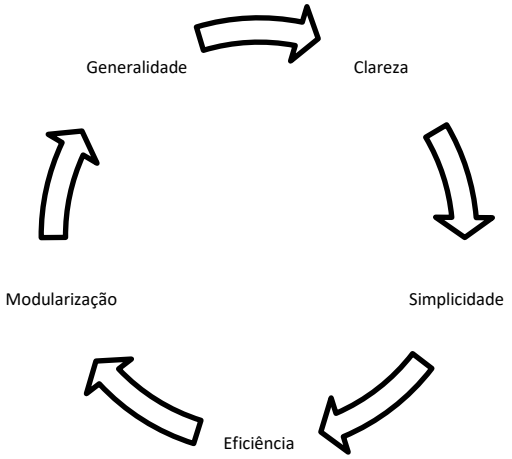
Implementação

Criação de novas funcionalidades.

Morte do programa

É quando o programa não atende mais as necessidades do usuário / cliente. Poderá ser substituído por outro ou nova versão.

Critérios de Qualidade de um Programa



Refere-se à precisão das informações manipuladas pelo programa, ou seja, os resultados gerados pelo processamento do programa devem estar corretos, caso contrário o programa simplesmente não tem sentido.

Clareza

Refere-se à facilidade de leitura do programa. Se um programa for escrito com clareza, deverá ser possível a outro programador seguir a lógica do programa sem muito esforço, assim como o próprio autor do programa entendê-lo após ter estado um longo período afastado dele.

Simplicidade

A clareza e precisão de um programa são normalmente melhoradas tornando as coisas o mais simples possível, consistentes com os objetivos do programa. Muitas vezes torna-se necessário sacrificar alguma eficiência de processamento, de forma a manter a estrutura do programa mais simples.

Eficiência

Refere-se à velocidade de processamento e a correta utilização da memória. Um programa deve ter performance suficiente para atender às necessidades do problema e do usuário, bem como deve utilizar os recursos de memória de forma moderada, dentro das limitações do problema.

Modularização

Durante a fase de projeto, a solução do problema total vai sendo fatorada em soluções de subproblemas, o que permite geralmente dividir o problema em forma natural em módulos com sub-rotinas claramente delimitadas, que podem ser implementados separadamente por diversos programadores de uma equipe, ou

	Anotações

seja, a modularização consiste no particionamento do programa em módulos menores bem identificáveis e com funções específicas, de forma que o conjunto desses módulos e a interação entre eles permite a resolução do problema de forma mais simples e clara.

Generalidade

É interessante que um programa seja tão genérico quanto possível de forma a permitir a reutilização de seus componentes em outros projetos.

Anotações

Exercícios

1) Montar em Descrição Narrativa uma sequência lógica para tomar banho. Verificar o que fazer se acabar a água.

2) Montar em Descrição Narrativa uma sequência lógica para trocar um pneu de um carro. Verificar se tem estepe.

3) Montar em Descrição Narrativa uma sequência lógica que crie um suco de acerola. Caso não tenha açúcar substituir por adoçante.

Anotações

Capítulo 02 - Lógica Proposicional

Objetivos

Neste capítulo você irá aprender:

- Introdução
- Princípios
- Proposições
- Tabela Verdade
- Operações Lógicas sobre Proposições
- Prioridade dos Conectivos

Exercícios

Os exemplos e exercícios utilizados neste capítulo estão na pasta: **Capítulo 02 - Lógica Proposicional.**

Duração

Este capítulo levará aproximadamente 60 minutos

Anotações

Introdução

Em lógica e matemática, uma lógica proposicional (ou cálculo sentencial) é um sistema formal no qual as fórmulas representam proposições que podem ser formadas pela combinação de proposições atômicas usando conectivos lógicos e um sistema de regras de derivação, que permite que certas fórmulas sejam estabelecidas como "teoremas" do sistema formal.

Em termos gerais, um cálculo é frequentemente apresentado como um sistema formal que consiste em um conjunto de expressões sintáticas (fórmulas bem formadas, ou FBFS), um subconjunto distinto dessas expressões, e um conjunto de regras formais que define uma relação binária específica, que se pretende interpretar como a noção de equivalência lógica, no espaço das expressões.

Quando o sistema formal tem o propósito de ser um sistema lógico, as expressões devem ser interpretadas como asserções matemáticas, e as regras, conhecidas como regras de inferência, normalmente são preservadoras da verdade. Nessa configuração, as regras (que podem incluir axiomas) podem então ser usadas para derivar "inferir" fórmulas representando asserções verdadeiras.

O conjunto de axiomas pode ser vazio, um conjunto finito não vazio, um conjunto finito enumerável, ou pode ser dado por axiomas esquemáticos. Uma gramática formal define recursivamente as expressões e fórmulas bem formadas (FBFS) da linguagem. Além disso, pode se apresentar uma semântica para definir verdade e valorações (ou interpretações).

A linguagem de um cálculo proposicional consiste em:

- Um conjunto de símbolos primitivos, definidos como fórmulas atômicas, proposições atômicas, ou variáveis;
- Um conjunto de operadores, interpretados como operadores lógicos ou conectivos lógicos.

Uma fórmula bem formada (FBF) é qualquer fórmula atômica ou qualquer fórmula que pode ser construída a partir de fórmulas atômicas, usando conectivos de acordo com as regras da gramática.

O que segue define um cálculo proposicional padrão. Existem muitas formulações diferentes as quais são todas mais ou menos equivalentes, mas que diferem nos detalhes:

- De sua linguagem, que é a coleção particular de símbolos primitivos e operadores;
- Do conjunto de axiomas, ou fórmulas distinguidas;
- Do conjunto de regras de inferência.

	Anotações

Princípios

Princípio da identidade	Princípio da não-contradição	Princípio do terceiro excluído
• Garante que uma proposição é igual a si mesma.	• Uma proposição não pode ser verdadeira e falsa.	• Uma proposição ou é verdadeira ou é falsa.

Proposições

As proposições são determinadas por sentenças declarativas pertencentes a certa linguagem que formam um conjunto de palavras ou símbolos e expressam uma ideia. As sentenças declarativas são afirmações que podem receber valores lógicos, Verdadeiro ou Falso apenas, e que um conjunto de palavras resultam em um pensamento completo. As proposições devem seguir os seguintes princípios:

- Princípio da identidade: garante que uma proposição é igual a si mesma;
- Princípio da não contradição: uma proposição não pode ser verdadeira e falsa;
- Princípio do terceiro excluído: uma proposição ou é verdadeira ou é falsa.

Exemplos

- O cachorro é um animal. - **Verdadeiro**
- 2 + 2 = 7 - **Falso**

Sentenças interrogativas, exclamativas e imperativas não são proposições, pois não é possível dizer se são verdadeiras ou falsas.

Exemplos

- Hoje está chovendo muito!
- Como foi a aula?
- Limpe a cozinha.
- Esta sentença não é verdadeira.

Anotações

Tabela Verdade

Proposição 1
V
F

P1	P2
V	V
V	F
F	V
F	F

P1	P2	P3
V	V	V
V	V	F
V	F	V
V	F	F
F	V	V
F	V	F
F	F	V
F	F	F

$F(x) : L = 2^n$

Onde

Elemento	Descrição
L	Número de linhas distintas
N	Número de proposições distintas
2	Número de valores lógicos possíveis

A tabela verdade é construída para determinar o valor lógico de uma proposição composta. Segue uma excelente estratégia para a construção da mesma.

Exemplo de construção da tabela verdade da proposição composta: p ^ q

Primeiramente verifique quantas “variáveis”, ou proposições simples que temos na proposição composta do exercício. Neste caso existem duas: p e q.

Anotações

Em seguida elevamos 2 ao número de variáveis, ou seja, 2^2. Nossa base do expoente é 2 pelo fato de possuir-se apenas 2 valores lógicos possíveis nas proposições (Verdadeiro ou Falso). O resultado de 2^2 é 4. Então nossa tabela terá 4 linhas, nessas linhas estarão todos os valores lógicos possíveis da nossa proposição composta.

Operações Lógicas sobre Proposições

Negação (Não) - ~

P	~P
Verdadeiro	**Falso**
Falso	**Verdadeiro**

A negação tem o valor inverso da fórmula negada.

Interpretações

"Não P", "Não é o caso de P", "A proposição 'P' é falsa".

Assim, em uma linguagem Lógica na qual P significa "Sócrates é mortal", ~P pode ser interpretada como "Sócrates não é mortal", e, se o primeiro é verdadeiro, o segundo é falso; e se o primeiro é falso, o segundo é verdadeiro.

Interpretar a negação por meio de antônimos também é uma alternativa, mas deve-se ter cautela, pois nem sempre é aplicável em todos os casos. No exemplo acima a interpretação por meio de antônimos é perfeitamente aplicável, ou seja, se P significa "Sócrates é mortal", ~P pode ser interpretada como "Sócrates é imortal". Por outro lado, em uma linguagem Lógica na qual Q significa "João é bom jogador", a proposição "João é mau jogador" não é a melhor interpretação para ~Q (João poderia ser apenas um jogador mediano).

Dica

O símbolo “¬” também pode ser utilizado como negação.

Conjunção (E) - ^

P	Q	P ^ Q
Verdadeiro	Verdadeiro	**Verdadeiro**
Verdadeiro	Falso	**Falso**
Falso	Verdadeiro	**Falso**
Falso	Falso	**Falso**

A conjunção entre duas fórmulas só é verdadeira quando ambas são verdadeiras.

Interpretação

"P ^ Q" pode ser interpretada como "P e Q", "Tanto P quanto Q", "Ambas as proposições 'P' e 'Q' são verdadeiras" etc.

	Anotações

Assim, em uma linguagem Lógica na qual P significa "Sou cidadão brasileiro" e Q significa "Sou estudante de filosofia", P ^ Q pode ser interpretada como "Sou cidadão brasileiro e estudante de filosofia"; o que só é verdade se P é verdadeira e Q é verdadeira.

Disjunção (Ou) - v

P	Q	P v Q
Verdadeiro	Verdadeiro	**Verdadeiro**
Verdadeiro	Falso	**Verdadeiro**
Falso	Verdadeiro	**Verdadeiro**
Falso	Falso	**Falso**

A disjunção entre duas fórmulas só é verdadeira quando ao menos uma delas é verdadeira.

Interpretação

"P v Q" pode ser interpretada como "P ou Q", "Entre as proposições P e Q, ao menos uma é verdadeira".

Assim, se P significa "Fulano estuda filosofia" e Q significa "Fulano estuda matemática", P v Q pode ser interpretada como "Fulano estuda filosofia ou matemática"; o que só é falso se nem P nem Q forem verdadeiras.

Com a disjunção é preciso tomar muito cuidado tanto na interpretação de fórmulas quanto na formalização de proposições, pois na linguagem natural muitas vezes os disjuntos são excludentes. Por exemplo: "Uma moeda ao ser lançada resulta em cara ou coroa", "Nestas férias eu vou viajar ou ficar em casa".

Disjunção Exclusiva (Ou Exclusivo) - v̲

P	Q	P v̲ Q
Verdadeiro	Verdadeiro	**Falso**
Verdadeiro	Falso	**Verdadeiro**
Falso	Verdadeiro	**Verdadeiro**
Falso	Falso	**Falso**

A disjunção exclusiva entre duas fórmulas só é verdadeira quando ao só uma delas é verdadeira, mas não ambos.

Exemplo

Ou o gato é macho ou o gato é fêmea, mas não ambos.

Interpretação

"P v̲ Q" pode ser interpretada como “Ou P ou Q", "Entre as proposições P e Q, ao só uma é verdadeira".

	Anotações

Assim, se P significa "Fulano estuda filosofia" e Q significa "Fulano estuda matemática", P v Q pode ser interpretada como “Ou Fulano estuda filosofia ou matemática"; o que só é verdadeiro se somente uma proposição for verdadeira.

Operação Condicional (Implicação) - →

P	Q	P → Q
Verdadeiro	Verdadeiro	**Verdadeiro**
Verdadeiro	Falso	**Falso**
Falso	Verdadeiro	**Verdadeiro**
Falso	Falso	**Verdadeiro**

A implicação entre duas fórmulas só é falsa se a da esquerda (antecedente) for verdadeira e da direita (consequente) for falsa.

Interpretação

“P → Q” pode ser interpretada como “Se P, então Q”, “P implica Q”, “Se a proposição 'P ' é verdade, então a proposição ' Q ' também é verdade”, “A partir de 'P' inferimos 'Q'“, “P satisfaz Q”, “P é condição suficiente de Q”.

Assim, se, em uma linguagem Lógica, P significa "O botão vermelho foi apertado" e Q significa "O lugar inteiro explode", P → Q pode ser interpretada como "Se o botão vermelho foi apertado, o lugar inteiro explode", o que só é falso se o botão vermelho for apertado (verdade de P) e o lugar inteiro não explodir (falsidade de Q):

A interpretação da implicação é uma das mais complicadas. Talvez você tenha estranhado que a implicação seja verdadeira quando o antecedente é falso. Ou ainda, você poderia objetar "mas e se o botão for apertado, o lugar explodir, mas uma coisa não tiver nada a ver com a outra?".

Basicamente, o que se deve observar é que "O botão vermelho ser apertado" é condição suficiente para se deduzir que "O lugar inteiro explodiu", isto é, quando o botão é apertado, o lugar deve explodir. Se o botão for apertado e o lugar não explodir, algo está errado, ou seja, P não implica Q (P → Q é falso).

Quando temos na linguagem natural uma proposição que afirma que, a partir de um evento, outro segue inexoravelmente (por exemplo: "Se você sair na chuva sem guarda-chuva ou capa de chuva, então você vai se molhar") ou uma proposição que afirma que podemos deduzir um fato de outro (por exemplo: "Se todo número par é divisível por 2, então nenhum número par maior que 2 é primo"), podemos seguramente formalizar estas proposições por meio da implicação.

Mas o contrário, ou seja, interpretar uma implicação na linguagem natural, é problemático. Podemos estar lidando com uma implicação cujo antecedente e cujo consequente não têm relação alguma. Basta, contudo, que o antecedente seja falso ou o consequente seja verdadeiro para que a implicação seja verdadeira. Nestes casos, é bem difícil dar uma interpretação satisfatória para a implicação.

Anotações

Fica mais fácil lembrar a regra assim: copia o último ao menos que seja Falso, Falso.

Operação Bicondicional (Equivalência) - ↔

P	Q	P ↔ Q
Verdadeiro	Verdadeiro	**Verdadeiro**
Verdadeiro	Falso	**Falso**
Falso	Verdadeiro	**Falso**
Falso	Falso	**Verdadeiro**

A bi-implicação entre duas fórmulas é verdadeira quando ambas são verdadeiras ou ambas são falsas.

Interpretação

"P ↔ Q" pode ser interpretada como "P se e somente se Q", "P é equivalente a Q", "P e Q possuem o mesmo valor de verdade".

Assim, se P significa "As luzes estão acesas" e Q significa "O interruptor está voltado para cima", P ↔ Q pode ser interpretada como "As luzes estão acesas se e somente se o interruptor está voltado para cima", o que só é falso se as luzes estiverem acesas e o interruptor não estiver voltado para cima (verdade de P falsidade de Q), ou se as luzes não estiverem acesas e o interruptor estiver voltado para cima (falsidade de P e verdade de Q).

Prioridade dos Conectivos

Operador	Operação
~	Negação
^	Conjunção
v	Disjunção
v̲	Disjunção Excludente
→	Condicional
↔	Bicondicional

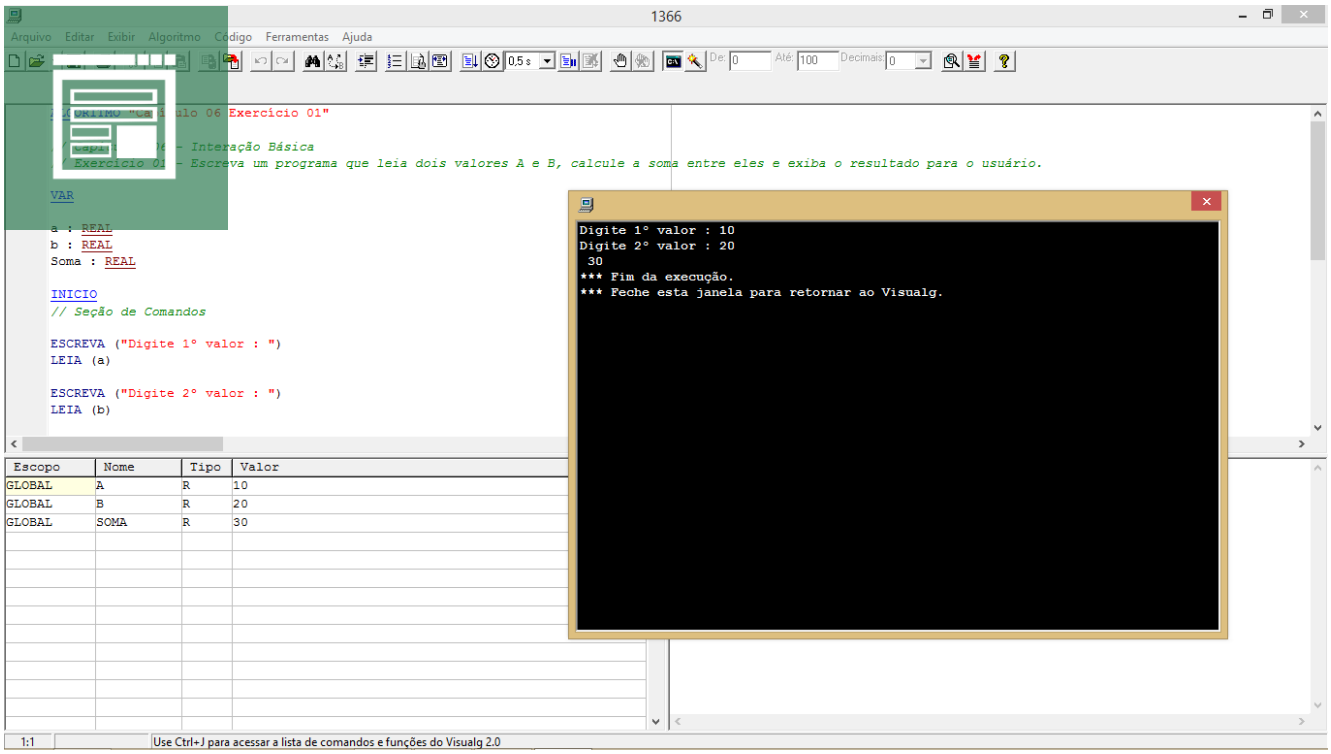
Anotações

Exercícios

1) (TRT 1ª Região/2008/CESPE) Utilizando as letras proposicionais adequadas na proposição composta “Nem Antônio é desembargador nem Jonas é juiz”, assinale a opção correspondente à simbolização correta dessa proposição.
   a) ¬(A ^ B)
   b) (¬A) v (¬B)
   c) (¬A) ^ (¬B)
   d) (¬A) → B
   e) ¬[A v (¬B)]
2) (UFB) Se p é uma proposição verdadeira, então:
   a) p ^ q é verdadeira, qualquer que seja q;
   b) p v q é verdadeira, qualquer que seja q;
   c) p ^ q é verdadeira só se q for falsa;
   d) p → q é falsa, qualquer que seja q
   e) n. d. a.
3) (UGF) A negação de x > -2 é:
   a) x > 2
   b) x <> -2
   c) x <= -2
   d) x < 2
   e) x <> 2
4) (ABC) A negação de todos os gatos são pardos é:
   a) Nenhum gato é pardo;
   b) Existe gato pardo;
   c) Existe gato não pardo;
   d) Existe um e um só gato pardo;
   e) Nenhum gato não é pardo.
5) (VUNESP) um jantar reúne 13 pessoas de uma mesma família. Das afirmações a seguir, referentes às pessoas reunidas, a única necessariamente verdadeira é:
   a) Pelo menos uma delas tem altura superior a 1,90m;
   b) Pelo menos duas delas são do sexo feminino;
   c) Pelo menos duas delas fazem aniversário no mesmo mês;
   d) Pelo menos uma delas nasceu num dia par;
   e) Pelo menos uma delas nasceu em janeiro ou fevereiro.

Anotações

Capítulo 03 - Trabalhando com o VisuAlg



Objetivos

Neste capítulo você irá aprender:

- A Tela Principal do VisuAlg
- A Barra de Tarefas
- Visualizador de Variáveis
- Simulador de Saída
- A Barra de Status
- O Menu do VisuAlg
- Lista de Funções
- Palavras Reservadas

Exercícios

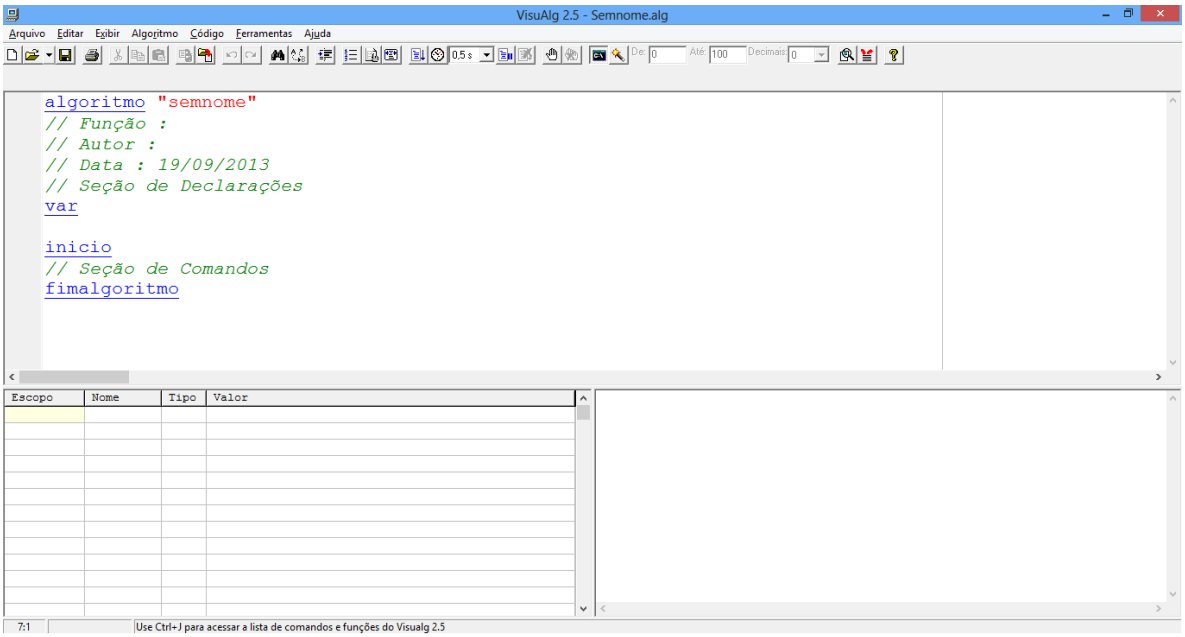
Os exemplos e exercícios utilizados neste capítulo estão na pasta: **Capítulo 03 - Trabalhando com o VisuAlg.**

Duração

Este capítulo levará aproximadamente 45 minutos

Anotações

A Tela Principal do VisuAlg



Quando o programa é carregado, já coloca no editor texto um "esqueleto" de algoritmo, cuja intenção é, além de poupar trabalho ao usuário, mostrar o formato básico de algoritmo que deve ser utilizado, bem como a forma dos comentários. A seguir explicamos cada componente da interface do VisuAlg.

A Barra de Tarefas

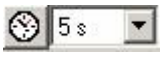
Contém os comandos mais utilizados no VisuAlg (estes comandos também podem ser acessados pelo menu ou por atalhos no teclado).

	Novo (Ctrl-N)	Cria um novo "esqueleto" de algoritmo, substituindo o texto existente no editor. Se este tiver sido modificado, o VisuAlg pedirá sua confirmação para salvá-lo antes que seja sobreposto.
	Abrir (Ctrl-A)	Abre um algoritmo anteriormente gravado, substituindo o texto existente no editor. Se este tiver sido modificado, o VisuAlg pedirá sua confirmação para salvá-lo antes que seja sobreposto.
	Salvar (Ctrl-S)	Salva imediatamente o texto do editor. Caso seja a primeira vez que um novo texto é gravado, o VisuAlg pedirá o seu nome e localização.

	Anotações

	Imprimir	Imprime imediatamente o texto existente no editor. Para configurar a impressão, use o comando Imprimir... do menu Arquivo (acessível também pelo atalho Ctrl-P).
	Recortar	Move um texto selecionado para a memória.
	Copiar	Copia um texto selecionado para a memória.
	Colar	Retira conteúdo da memória e coloca no local do cursor.
	Gravar Bloco	Abre uma janela para salvar o bloco selecionado. Objetivo é criar uma espécie de biblioteca de funções.
	Inserir um bloco	Insere um bloco arquivado no local do cursor.
	Desfazer e refazer	Desfaz e refaz ação criada no editor de texto.
	Localizar	Abre uma janela para você digitar a palavra que deseja localizar no Editor de textos.
	Substituir	Abre uma janela para você poder digitar a palavra que deseja localizar e substituir no Editor de textos.
	Corrigir Indentação (Ctrl-G)	Corrige automaticamente a indentação do "código-fonte", colocando os comandos dentro de uma estrutura de 3 colunas à direita da coluna inicial da estrutura conforme a configuração padrão.
	Numerar Linhas	Liga/desliga a exibição dos números das linhas na área à esquerda do editor. A linha e coluna do editor em que o cursor está em um determinado momento também são mostradas na primeira parte da barra de status, situada na parte inferior da tela. Os números de linhas, caso ligados, são desligados durante a execução do algoritmo por motivos técnicos, mas são ligados de volta ao fim do "programa".
	Mostra Variáveis Modificadas	Liga/desliga a exibição da variável que está sendo modificada. Como o número

Anotações

		de variáveis pode ser grande, muitas podem estar fora da janela de visualização; quando esta característica está ligada, o programa rola a grade de variáveis de modo que aquela que está sendo modificada no momento fique visível. Útil quando se está executando o algoritmo passo a passo. Por questões de performance, o valor padrão desta característica é desligada quando o algoritmo está sendo executado automaticamente, mas se você clicar este botão pode executar o algoritmo automaticamente com a exibição ligada. O valor volta automaticamente para desligada ao fim da execução.
		Restaura a teça inicial do Visual. Retorna a tela original do programa
	Executar (F9)	Inicia (ou continua) a execução automática do algoritmo.
	Tempo por linha (shift + F9)	Executa o algoritmo linha por linha automaticamente determinado por tempo escolhido pelo usuário.
	Passo (F8)	Inicia (ou continua) a execução do algoritmo linha por linha, dando ao usuário oportunidade de acompanhar o fluxo do programa, examinar variáveis, etc.
	Parar (Ctrl-F2)	Termina imediatamente a execução do algoritmo. Este botão fica desabilitado quando o algoritmo não está sendo executado.
	Marca e Desmarca um Breakpoints (F5)	Cria pontos de parada. Selecione a linha que deseja criar um ponto de parada na hora de execução do algoritmo e pressione o Breakpoints surgirá uma linha marrom e um marcador do lado esquerdo para indicar o ponto de parada. Para desmarcar o ponto de parada basta selecionar a linha que possui o breakpoint e clicar no mesmo. O breakpoint não funciona no modo passo a passo e para continuar o

Anotações

		algoritmo depois de uma parada pressione novamente o F9 ou o botão Executar.
	Desmarca todos os BreakPoints (CTRL+F5)	
	Executa em Modo DOS	Quando ativado durante a execução do algoritmo ele executa o algoritmo em uma janela em modo dos.
	Gerador valores	Substitui digitação do usuário por um sistema de geração aleatória de valores numéricos e caracteres (este comando não afeta a leitura de variáveis do tipo lógico com certeza uma coisa pouco usual.). Gera números e caracteres aleatoriamente, muito útil para não perder tempo pensando o que digitar, você escolhe o início e o fim dos valores e se for valores com casas decimais é só escolher quantas casas decimal você quer.
	Perfil (F7)	Mostra, após a execução de um algoritmo, quantas vezes cada linha foi executada. Útil para a análise de eficiência de um algoritmo, como por exemplo, nos métodos de classificação.
	Pilha (CTRL+F3)	Mostra a pilha de ativação do programa (call stack), com o nome dos procedimentos e funções chamados, nome, tipo e valor dos parâmetros.
	Ajuda on-line (F1)	Em construção.

Visualizador de Variáveis

Contém uma grade onde são mostrados: Escopo da variável (Global quando nome da variável for global ou o nome da função ou Procedimento quando for local). O Nome da variável (com índice ou índices, caso seja um elemento de um vetor), seu Tipo ("R" para Real, “I” para Inteiro, "C" para literal e "L" para lógico), e o seu Valor corrente. A versão atual do VisuAlg permite até 500 variáveis (cada elemento de um vetor conta como uma variável).

Também, de acordo com o tipo de parâmetro a cor no grid muda, e para os parâmetros passados por referência. Há uma seta que mostra o nome da variável que eles representam fora do subprograma. Isto tudo, naturalmente, só pode ser visto se executarmos o algoritmo passo a passo...

	Anotações

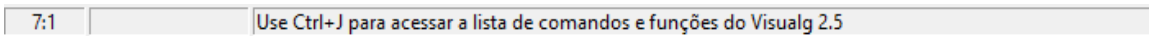
Escopo	Nome	Tipo	Valor
GLOBAL	BIM1	R	7
GLOBAL	BIM2	R	6
GLOBAL	BIM3	R	7
GLOBAL	BIM4	R	10
GLOBAL	MEDIAS	R	0
GLOBAL	MEDIAP	R	0
MEDIASIMPLES	(RETORNO)	R	0
MEDIASIMPLES	NOTA1	R	7
MEDIASIMPLES	NOTA2	R	6
MEDIASIMPLES	NOTA3	R	7
MEDIASIMPLES	NOTA4	R	10
MEDIASIMPLES	RESULTADO	R	7.5

Simulador de Saída

Mostra o resultado do algoritmo, invés de executar no modo DOS, você pode verificar a saída do algoritmo aqui no simulador de saída.

```
Início da execução
1° Bimestre: 7
2° Bimestre: 8
3° Bimestre: 9
4° Bimestre: 10
Média Simpes:  8.5
Média Ponderada:  9

Fim da execução.
```

A Barra de Status



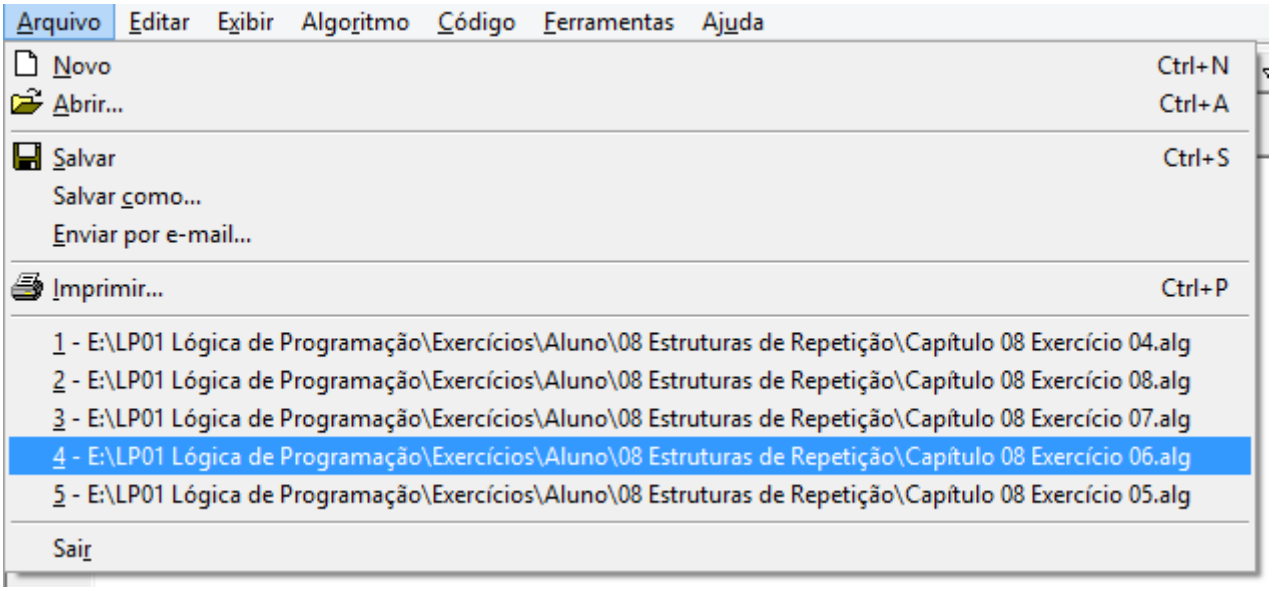
Situada na parte inferior da tela, contém três painéis. O primeiro mostra a linha e coluna onde o cursor está; o segundo mostra a palavra Modificado caso o algoritmo tenha sido alterado desde que foi carregado ou salvo pela última vez. O terceiro mostra o comando que mostra as funções predefinidas do VisuAlg.

Anotações

O Menu do VisuAlg

Arquivo

Possui os comandos para se abrir, salvar e imprimir algoritmos:



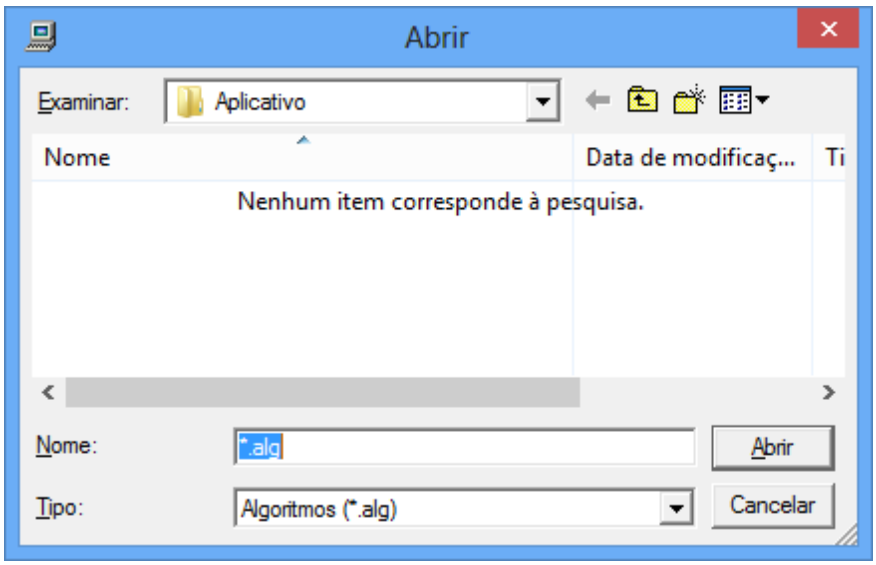
Novo

Cria um novo "esqueleto" de algoritmo, substituindo o texto existente no editor. Se este tiver sido modificado, o VisuAlg pedirá sua confirmação para salvá-lo antes que seja sobreposto.

Abrir

Abre um algoritmo anteriormente gravado, substituindo o texto existente no editor. Se este tiver sido modificado, o VisuAlg pedirá sua confirmação para salvá-lo antes que seja sobreposto.

	Anotações



Salvar

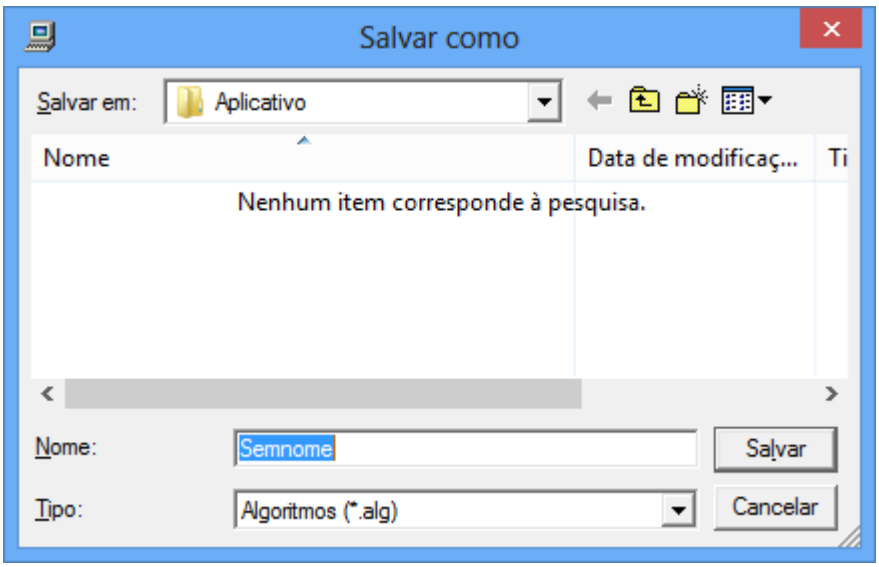
Salva imediatamente o texto do editor. Caso seja a primeira vez que um novo texto é gravado, o VisuAlg pedirá o seu nome e localização.

Salvar como ...

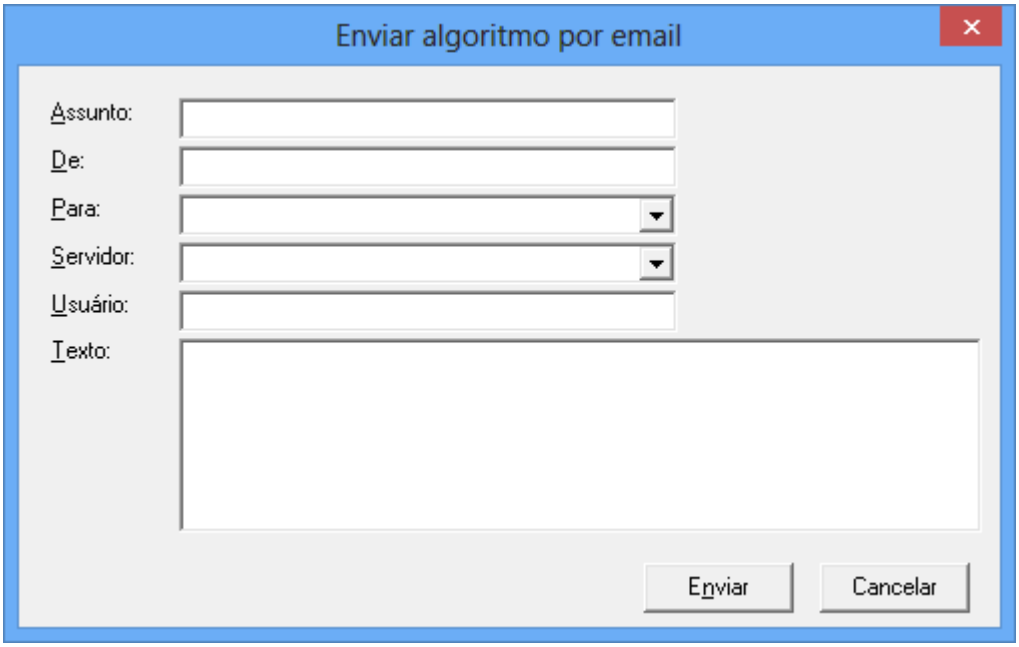
Permite salvar o texto do editor, exibindo antes a janela para se escolher o nome e localização.

Enviar por e-mail... Permite mandar o algoritmo por e-mail.

	Anotações



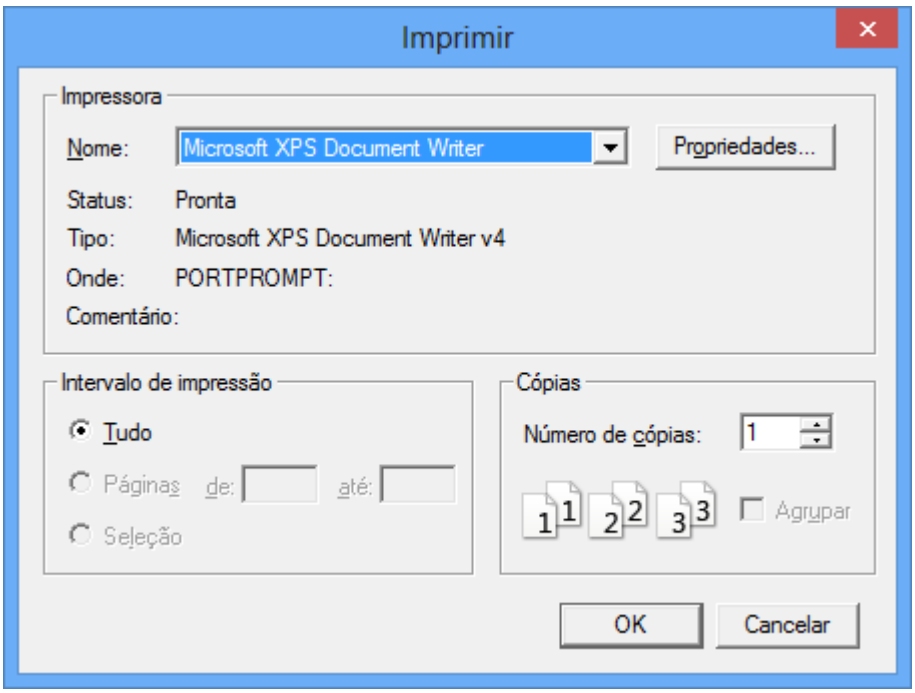
Enviar por e-mail



Imprimir...

Permite a impressão do algoritmo corrente, mostrando antes a janela de configuração de impressão (o botão Imprimir da barra de tarefas imprime o algoritmo imediatamente na impressora padrão).

	Anotações



"5 últimos"

Além destes comandos, há ainda a lista dos 5 últimos algoritmos utilizados, que podem ser abertos diretamente ao se escolher o seu nome.

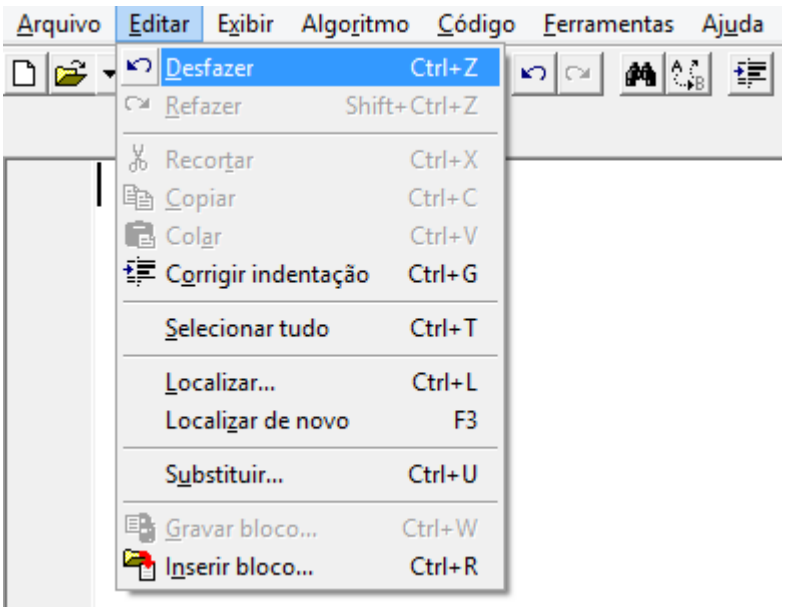
Sair

Abandona o VisuAlg.

Editar

Possui os comandos de manipulação de textos:

	Anotações



Desfazer

Desfaz a última ação feita no editor de texto.

Refazer

Refaz a última ação desfeita no editor de texto.

Recortar

Move um texto selecionado para a memória.

Copiar

Copia um texto selecionado para a memória.

Colar

Retira conteúdo da memória e coloca no local do cursor.

Corrigir Indentação (Ctrl-G)

Corrige automaticamente a indentação do "código-fonte", colocando os comandos dentro de uma estrutura 3 colunas à direita da coluna inicial da estrutura.

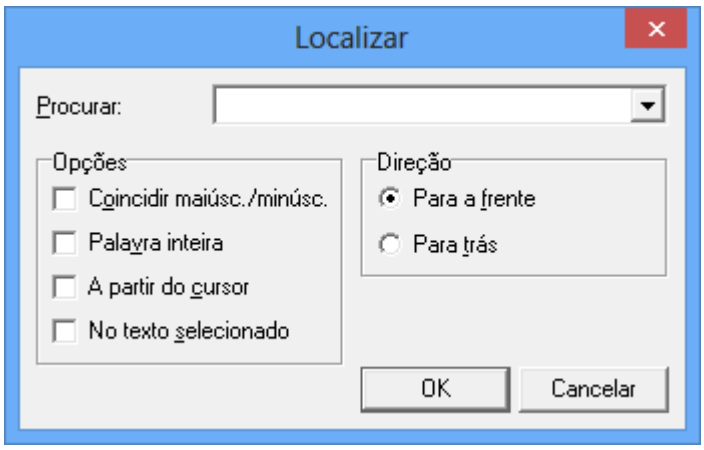
Seleciona tudo

Marca todo o texto no editor de texto.

Localizar

Abre uma janela para você digitar a palavra que deseja localizar no Editor de textos.

	Anotações

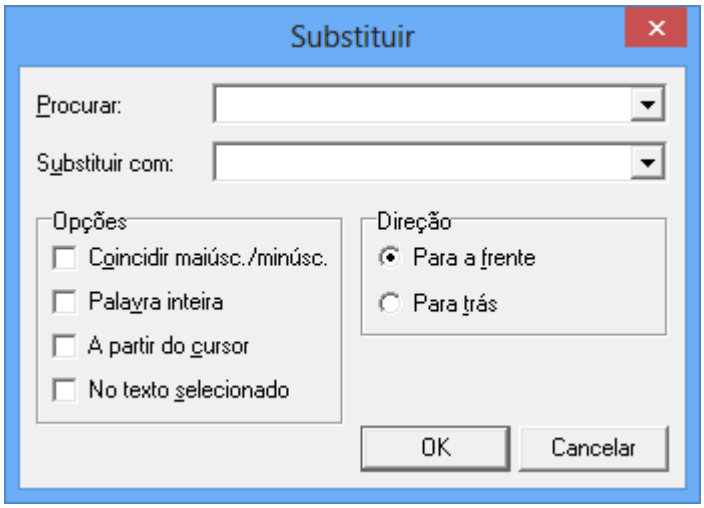


Localizar de novo (F3)

Procura pela a última palavra localizada.

Substituir

Abre uma janela para você poder digitar a palavra que deseja localizar e substituir no Editor de textos.



Gravar um bloco (Ctrl+W)

Salva um bloco selecionado. Objetivo é criar uma espécie de biblioteca de funções.

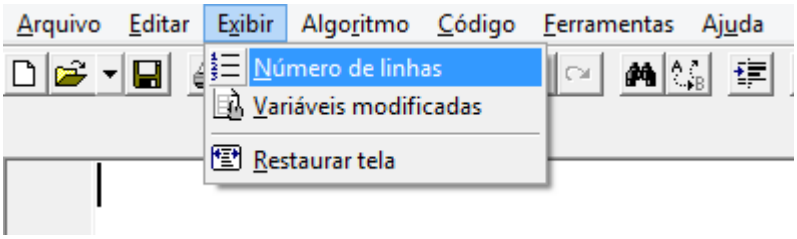
Inserir um bloco (Ctrl+R)

Insere um bloco salvo no local do cursor.

Exibir

Possui os comandos para ligar/desligar as seguintes características:

	Anotações



Número de linhas

Liga/desliga a exibição dos números das linhas na área à esquerda do editor. A linha e coluna do editor em que o cursor está em um determinado momento também são mostradas na primeira parte da barra de status, situada na parte inferior da tela. Os números de linhas, caso ligados, são desligados durante a execução do algoritmo por motivos técnicos, mas são ligados de volta ao fim do "programa".

Variáveis modificadas

Liga/desliga a exibição da variável que está sendo modificada. Como o número de variáveis pode ser grande, muitas podem estar fora da janela de visualização; quando esta característica está ligada, o programa rola a grade de variáveis de modo que aquela que está sendo modificada no momento fique visível. Útil quando se está executando o algoritmo passo a passo. Por questões de performance, o valor padrão deste item é desmarcado quando o algoritmo está sendo executado automaticamente, mas se você clicá-lo pode executar o algoritmo automaticamente com a exibição ligada. O item volta para desmarcado ao fim da execução (ele está relacionado ao botão correspondente da barra de tarefas).

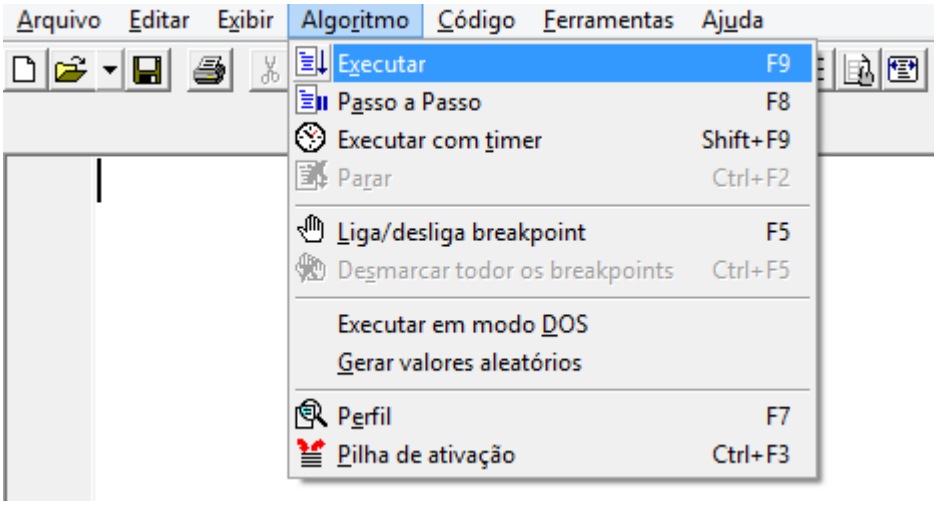
Restaura tela

Retorna a tela original do programa VisuAlg.

Algoritmo

Contém os comandos relativos à execução do algoritmo:

	Anotações



Executar

Inicia (ou continua) a execução automática do algoritmo.

Passo a passo

Inicia (ou continua) a execução do algoritmo linha por linha, dando ao usuário oportunidade de acompanhar o fluxo do programa, examinar variáveis, etc.

Executar com timer

Executa o algoritmo linha por linha automaticamente determinado por um tempo escolhido pelo o usuário.

Parar

Termina imediatamente a execução do algoritmo. Este item fica desabilitado quando o algoritmo não está sendo executado.

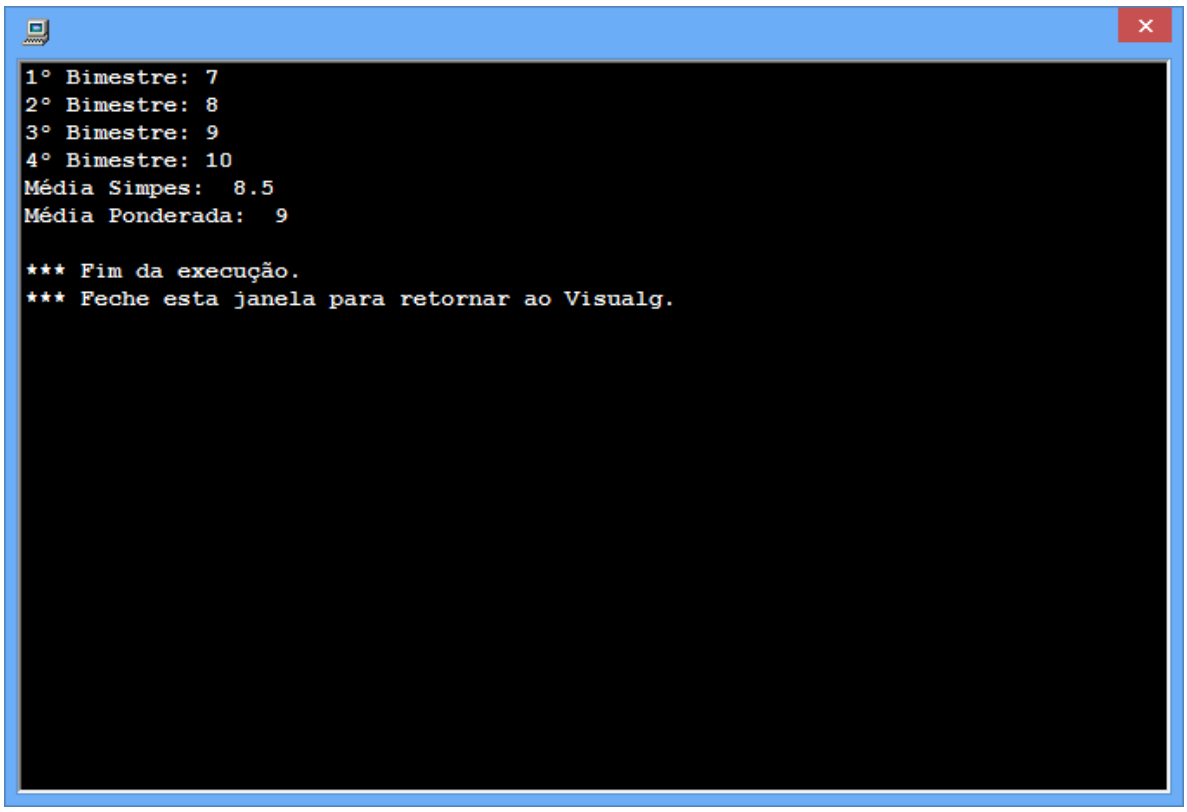
Marca e Desmarca um Breakpoints (F5)

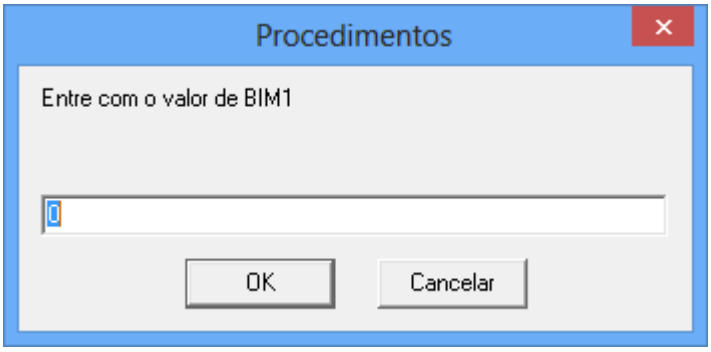
Cria pontos de parada. Selecione a linha que deseja criar um ponto de parada na hora de execução do algoritmo e pressione o Breakpoints surgirá uma linha marrom e um marcador do lado esquerdo para indicar o ponto de parada. Para desmarcar o ponto de parada basta selecionar a linha que possui o breakpoint e clicar no mesmo. O breakpoint não funciona no modo passo a passo e para continuar o algoritmo depois de uma parada pressione novamente o F9 ou o botão Executar. Desmarca todos os BreakPoints (CTRL+F5)

Executa em Modo DOS

Quando ativado durante a execução do algoritmo ele executa o algoritmo em uma janela em modo DOS

	Anotações





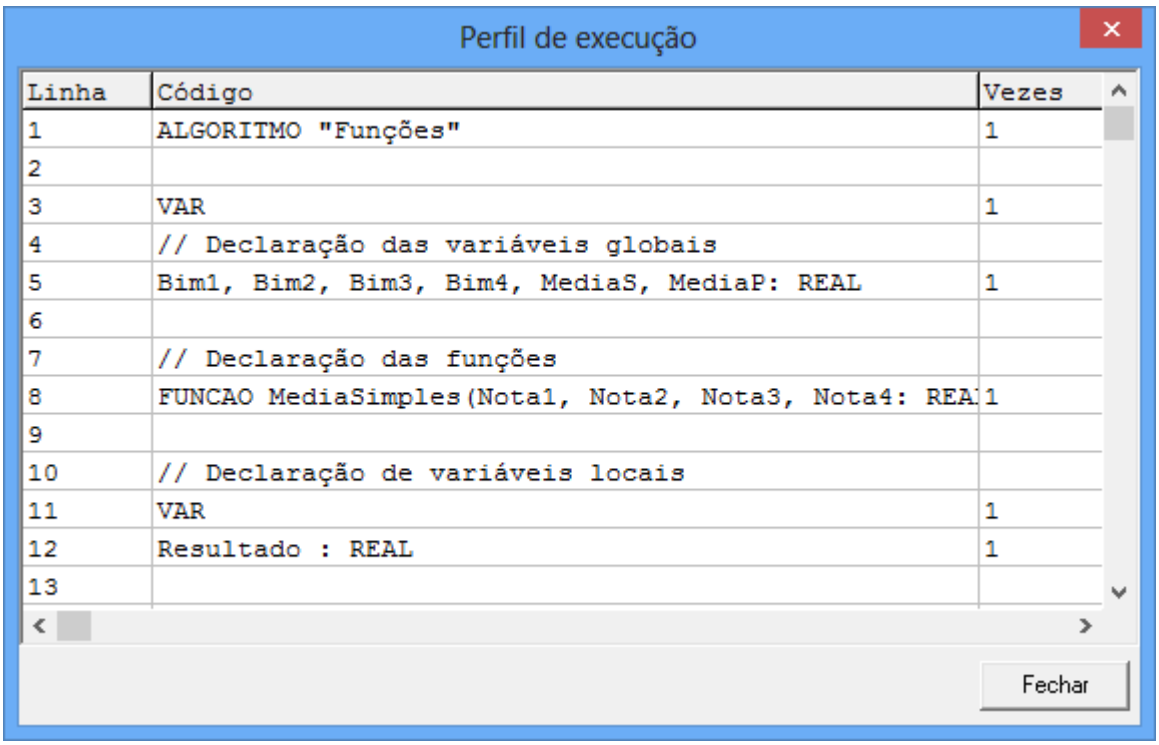
Gerar valores aleatórios

Substitui digitação do usuário por um sistema de geração aleatória de valores numéricos e caracteres (este comando não afeta a leitura de variáveis do tipo lógico com certeza uma coisa pouco usual.), Gera números e caracteres aleatoriamente, muito útil para não perder tempo pensando o que digitar, você escolhe o início e o fim dos valores e se for valores com casas decimais é só escolher quantas casas decimal você quer.

Perfil

Mostra, após a execução de um algoritmo, quantas vezes cada linha foi executada. Útil para a análise de eficiência de um algoritmo, como por exemplo, nos métodos de classificação.

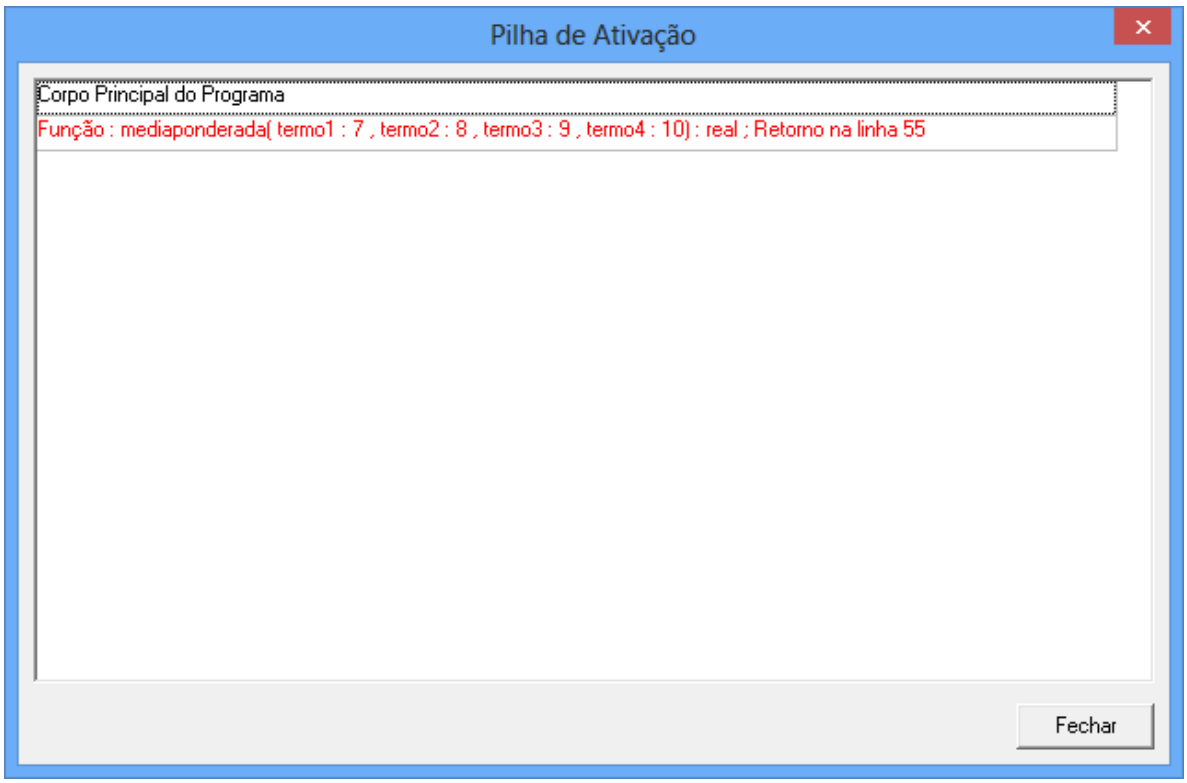
	Anotações



Pilha de ativação (CTRL+F3)

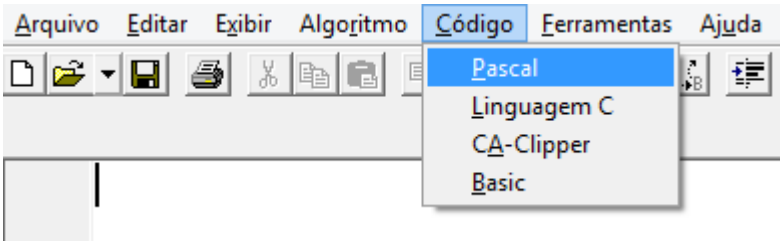
Mostra a pilha de ativação do programa (call stack), com o nome dos procedimentos e funções chamados, nome, tipo e valor dos parâmetros.

	Anotações



Código

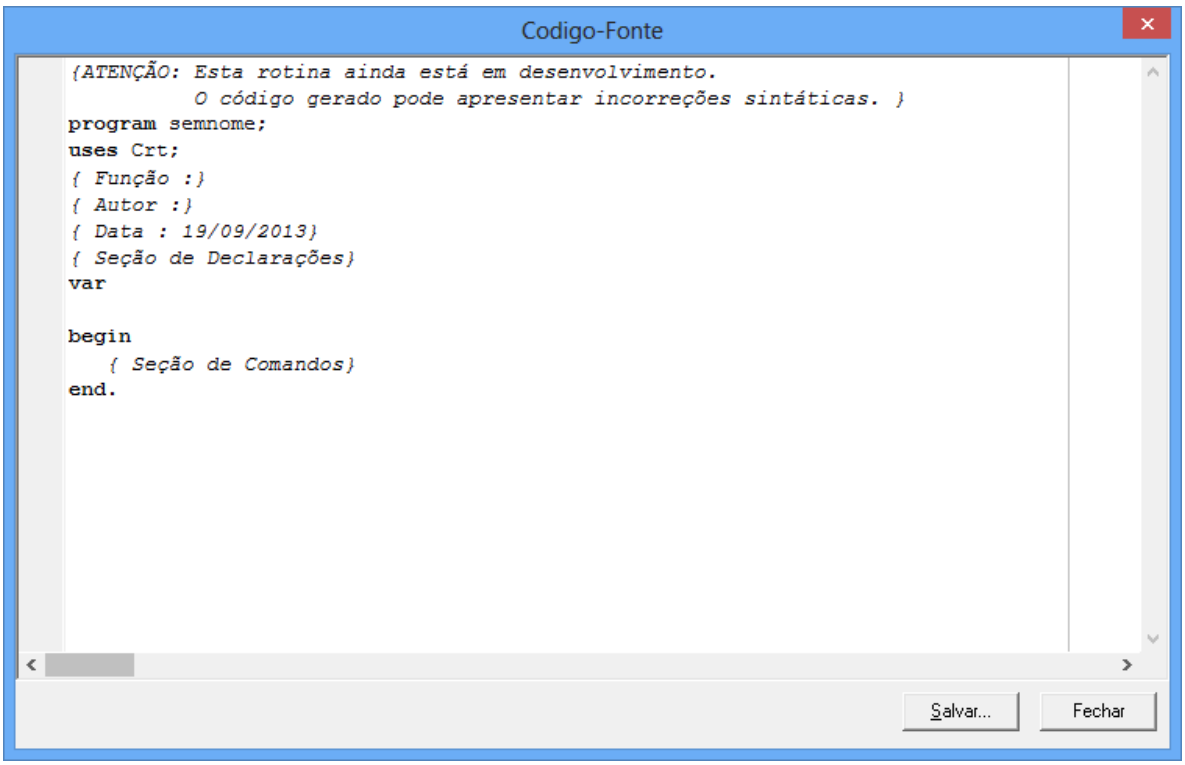
Contém os comandos relativos ao "código-fonte":



Pascal

Atualmente ele gera apenas o pascal.

	Anotações



Linguagem C

Em futuras versões do VisuAlg, este menu conterá também os comandos para geração de código-fonte nas linguagens especificadas, a partir do algoritmo corrente.

CA-Clipper

Em futuras versões do VisuAlg, este menu conterá também os comandos para geração de código-fonte nas linguagens especificadas, a partir do algoritmo corrente.

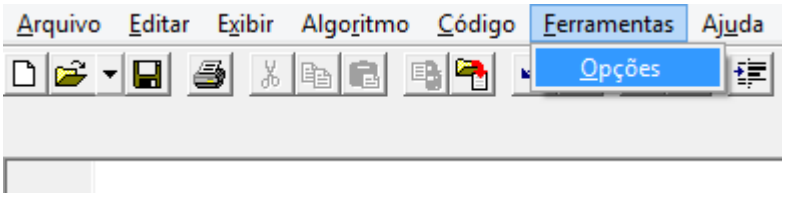
Visual Basic

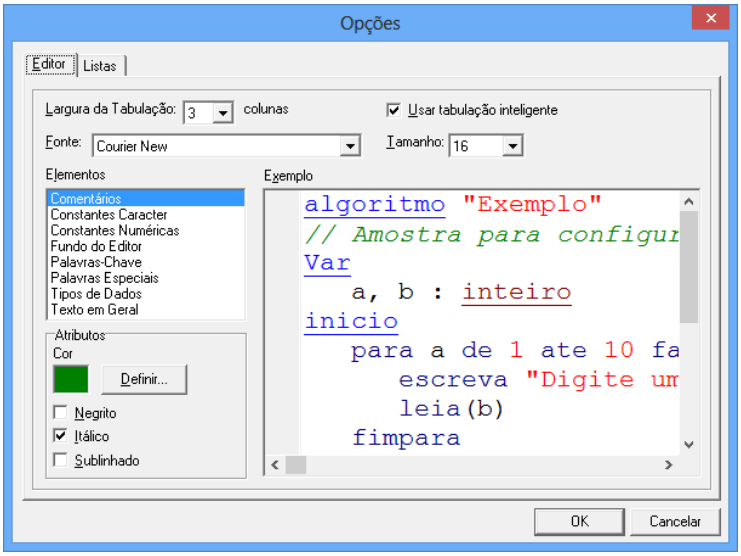
Em futuras versões do VisuAlg, este menu conterá também os comandos para geração de código-fonte nas linguagens especificadas, a partir do algoritmo corrente.

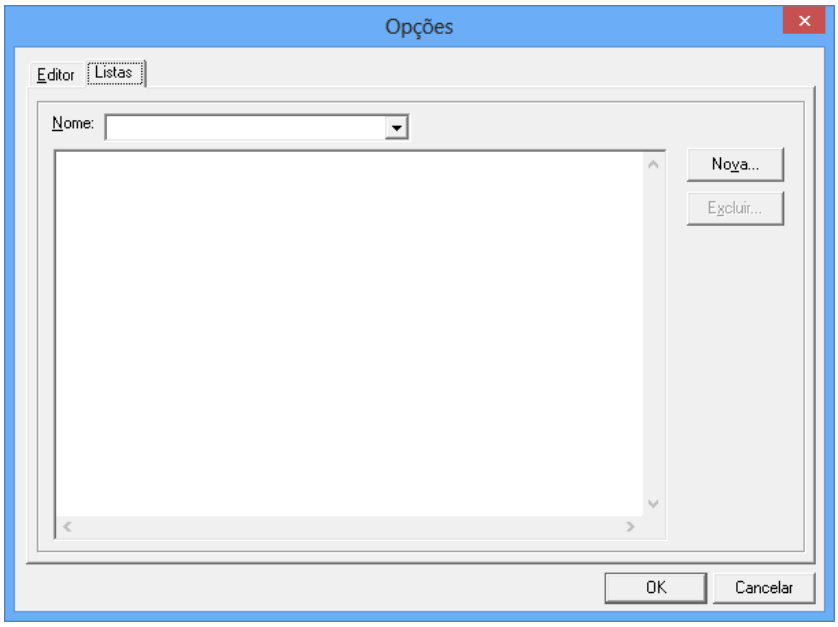
Ferramentas

Mostra duas abas, aba Editor onde você pode personalizar a fonte, cor, tabulação do VisuAlg e a aba Lista que você pode criar uma lista de dados para serem inseridas nas variáveis.

	Anotações



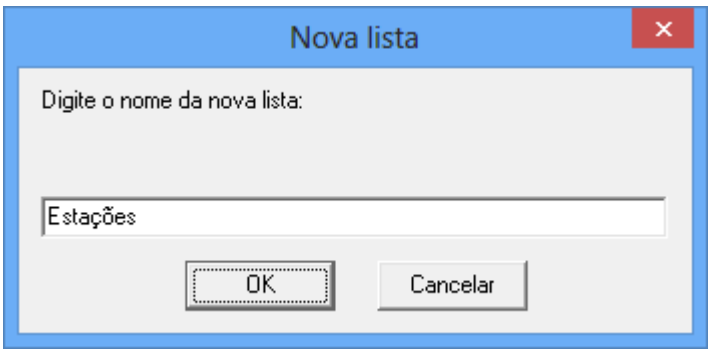




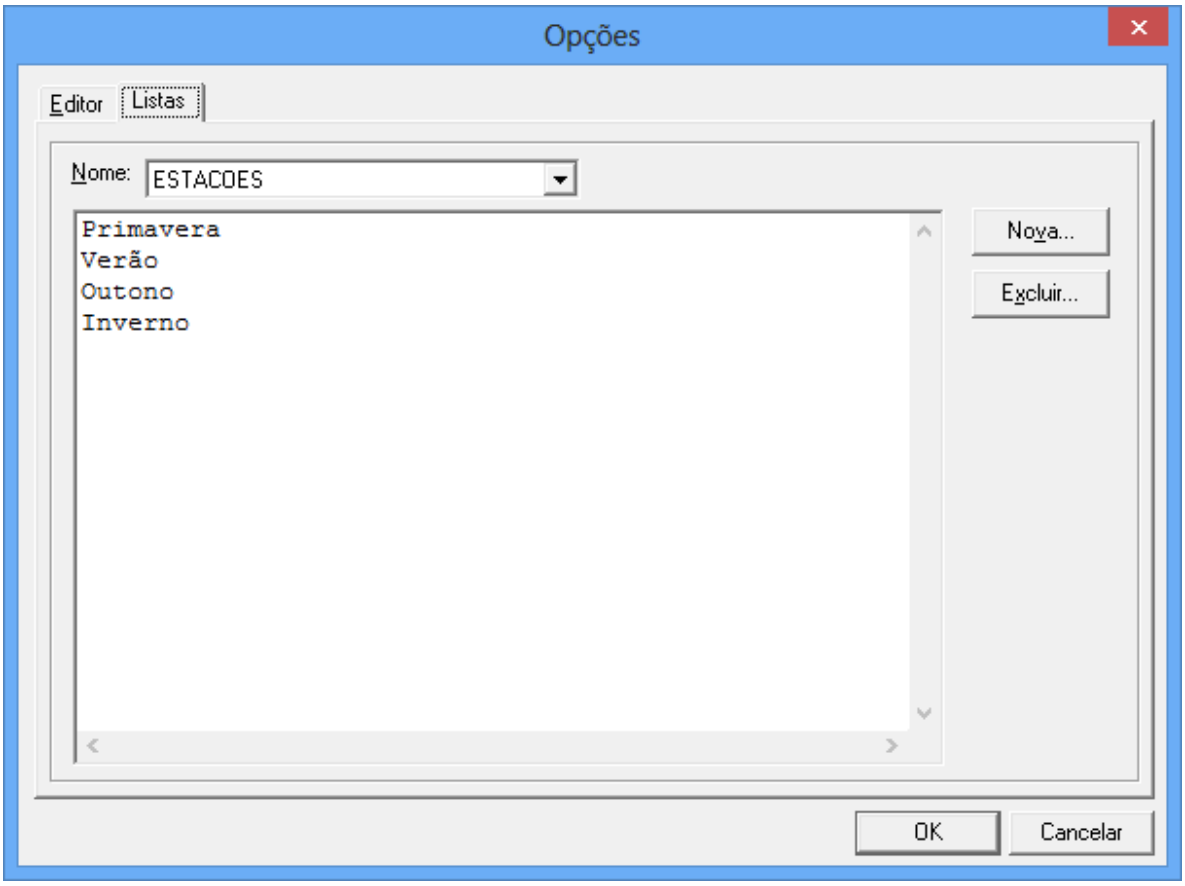
Como utilizar comando lista

Clicando no botão novo irá aparecer uma janela pedindo o nome da lista. Digite o nome desejado e de um OK.

	Anotações



Em seguida digite a lista de dados que deseja que apareça nas variáveis.



No algoritmo antes do comando leia coloque o seguinte comando: LISTA

"Computador" e deixe a opção  Gerar valores aleatórios. Veja um exemplo:

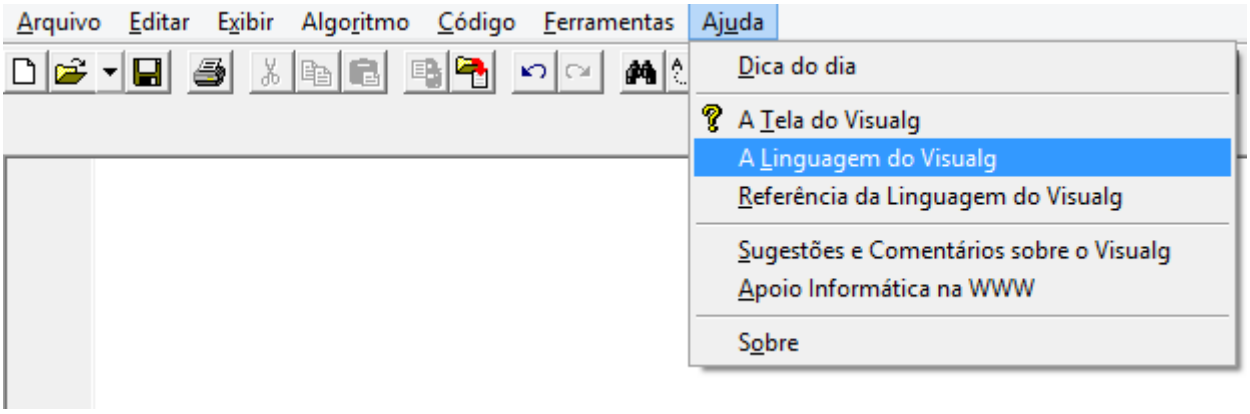
```
ALGORITMO "Comando Lista"
VAR
```

	Anotações

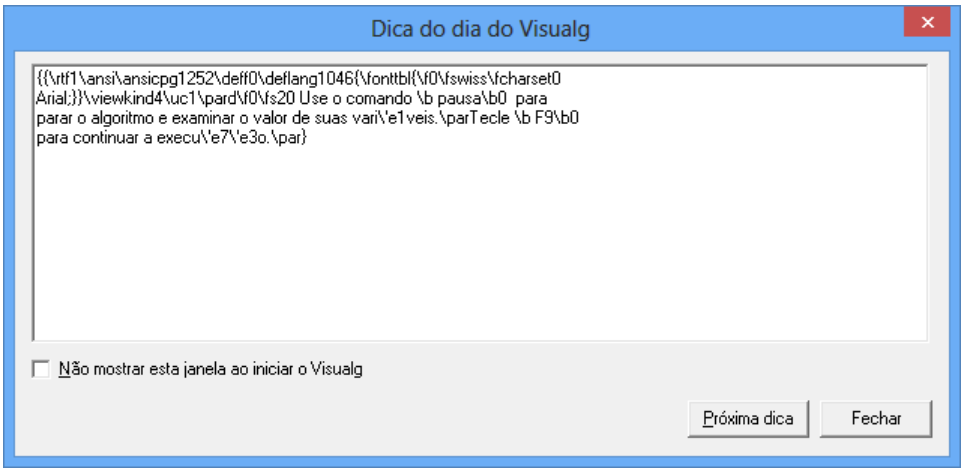
```
Estacoes :CARACTER
INICIO
LISTA "Estacoes"
LEIA (Estacoes)
ESCREVAL (Estacoes)
FIMALGORITMO
```

Ajuda

Contém o acesso às páginas de ajuda do VisuAlg e à janela Sobre.



Dica do dia



A Tela do VisuAlg

Idem não implementado.

A Linguagem do VisuAlg

Idem não implementado.

	Anotações

Referência da Linguagem do VisuAlg

Idem não implementado.

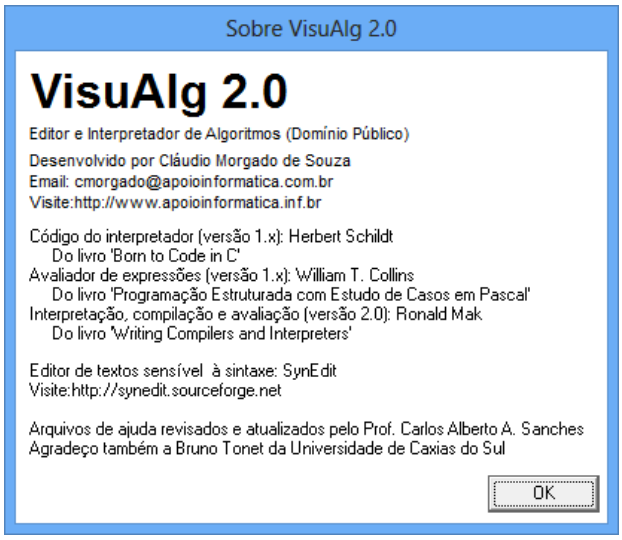
Sugestões e Comentários sobre o VisuAlg

Monta um e-mail para contato@apoioinformatica.inf.br com o assunto "VisuAlg" para que o usuário escreva suas sugestões e/ou comentário e os envie por e-mail.

Apoio Informática na WWW

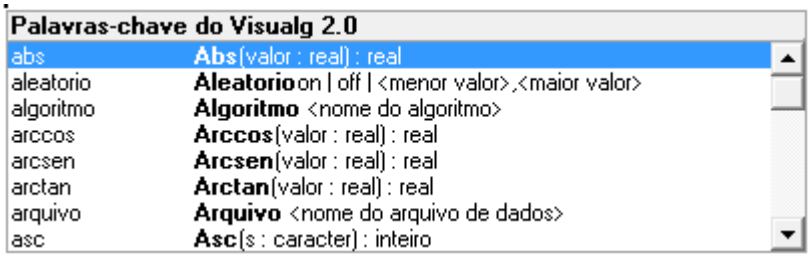
Abre no browser o site http://www.apoioinformatica.inf.br

Sobre



Lista de Funções

Lista de funções é uma maneira rápida de inserir funções predefinidas em seu algoritmo, a sua utilização é muito simples basta pressionar CTRL+J para mostrar a lista e selecionar a função desejada e dar um ENTER, depois é só passar os parâmetros desejados.



Anotações

Palavras Reservadas

ABS	ALEATORIO	ALGORITMO	ARCCOS	ARCSEN
ARCTAN	ARQUIVO	ASC	ATE	CARACTER
CASO	COMPR	COPIA	COS	COTAN
CRONOMETRO	DEBUG	DECLARE	E	ECO
ENQUANTO	ENTAO	ESCOLHA	ESCREVA	EXP
FACA	FALSO	FIMALGORITMO	FIMENQUANTO	FIMESCOLHA
FIMFUNCAO	FIMPARA	FIMPROCEDIMENTO	FIMREPITA	FIMSE
FUNÇÃO	GRAUPRAD	INICIO	INT	INTERROMPA
LEIA	LITERAL	LOG	LOGICO	LOGN
MAIUSC	MENSAGEM	MINUSC	NAO	NUMERICO
NUMPCARAC	OU	OUTROCASO	PARA	PASSO
PAUSA	PI	POS	PROCEDIMENTO	QUAD
RADPGRAU	RAIZQ	RAND	RANDI	REPITA
SE	SEN	SENAO	TAN	TIMER
VAR	VERDADEIRO	XOU		

Anotações

Exercícios

1) Qual a principal função do “Esqueleto” gerado quando abrimos o VisuAlg em nosso editor de texto?

2) Quais as possibilidades de apresentação de saída com o VisuAlg?

3) Quando e onde podemos visualizar os valores atribuídos a nossas variáveis?

Anotações

Capítulo 04 - Tipos de Dados, Constantes e Variáveis

Objetivos

Neste capítulo você irá aprender:

- Identificador
- Tipos de Identificadores
- Variável
- Constante
- Tipos de Primitivos de Dados
- Escopo e Visibilidade
- Regras de Nomenclatura

Exercícios

Os exemplos e exercícios utilizados neste capítulo estão na pasta: **Capítulo 04 - Tipos de Dados, Constantes e Variáveis.**

Duração

Este capítulo levará aproximadamente 45 minutos

	Anotações

Identificador

in.di.ca.dor
adj (baixo-lat indicatore)

1 que indica, ou serve de indicação.

2 *Informática*: É uma posição, localizada na memória capaz de reter e representar um valor.

Tipos de Identificadores

Variável	Constante

Variável

va.ri.á.vel
adj m+f (lat variabile)

1 Sujeito a variação

2 *Informática*: Corresponde a uma posição de memória cujo valor pode variar ao longo do tempo durante a execução do programa.

Constante

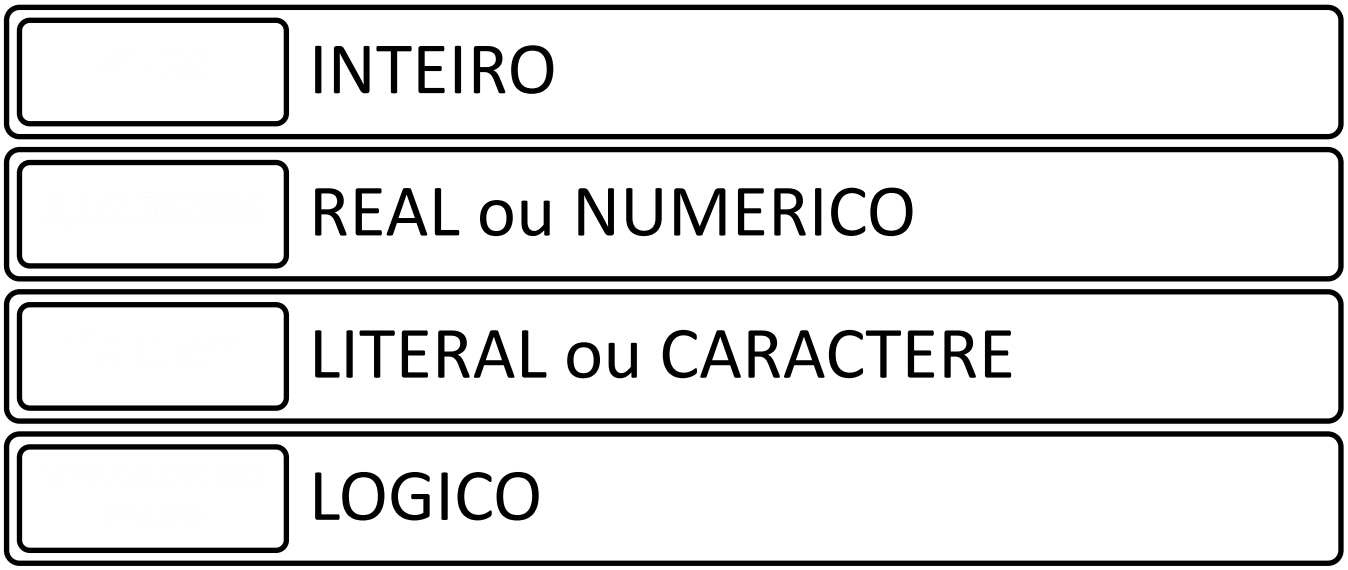
cons.tan.te
adj m+f (lat constante)

1 que não se desloca; firme, imutável.

2 *Informática*: Corresponde a uma posição de memória cujo valor é constante (não pode variar) ao longo do tempo durante a execução do programa.

	Anotações

Tipos de Primitivos de Dados



INTEIRO

São caracterizados como tipos inteiros, os dados numéricos positivos ou negativos. Excluindo-se destes quaisquer números fracionários.

Como exemplo deste tipo de dado, têm-se os valores: 6, -55, 72, 54, -74 entre outros.

REAL ou NUMERICO

São caracterizados como tipos reais, os dados numéricos positivos e negativos e números fracionários.

Como exemplo deste tipo de dado, têm-se os valores: 636.36, -86.75, 127.83, -649.34 entre outros.

LITERAL ou CARACTERE

São caracterizados como tipos caracteres, as sequências contendo letras, números e símbolos especiais. Uma sequência de caracteres deve ser indicada entre aspas (“”). Este tipo de dado também é conhecido como alfanumérico, string, literal ou cadeia.

Como exemplo deste tipo de dado, tem-se os valores: “Programação”, “av. Paulista, 171”, “Telefone (011) 3675 0033”, “01311-904”, “ ”, “42” entre outros.

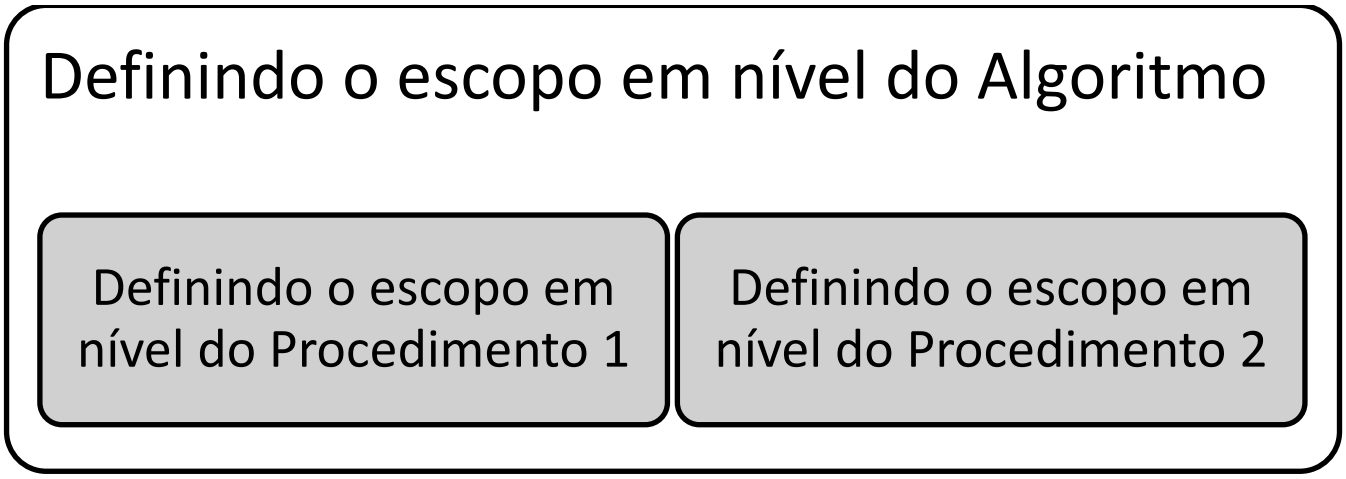
LOGICO

São caracterizados como tipos lógicos os dados com valor VERDADEIRO e FALSO, sendo que este tipo de dado poderá representar apenas um dos dois valores. Ele é chamado por alguns de **tipo booleano**, devido à contribuição do filósofo e matemático inglês *George Boole* na área da lógica matemática.

	Anotações

```
ALGORITMO "Tipos de Dados"

// Seção de Declarações
VAR

// Números inteiros
Idade : INTEIRO
Quantidade : INTEIRO

// Números com decimais
Salario : REAL
Comissao : NUMERICO

// Textos
Nome : CARACTER
Cidade : LITERAL

// Valores Lógicos (Verdadeiro / Falso)
VIP : LOGICO
Aprovado : LOGICO

INICIO
// Seção de Comandos

FIMALGORITMO
```

Anotações




```
ALGORITMO "Escopo e Visibilidade"

VAR
// Declaração das variáveis globais
Bim1, Bim2, Bim3, Bim4, MediaS, MediaP : REAL

// Declaração das funções
FUNCAO MediaSimples(Nota1, Nota2, Nota3, Nota4 : REAL) : REAL

// Declaração de variáveis locais
VAR
Resultado : REAL

INICIO

// Instruções
Resultado := (Nota1 + Nota2 + Nota3 + Nota4) / 4)

// Valor de retorno
RETORNE Resultado

FIMFUNCAO
```

Anotações

```
FUNCAO MediaPonderada(Termo1, Termo2, Termo3, Termo4 : REAL) : REAL

VAR
// Declaração de variáveis locais
Resultado : REAL

INICIO

// Instruções
Resultado := (Termo1 * 1 + Termo2 * 2 + Termo3 * 3 + Termo4 * 4) /
10

// Valor de retorno
RETORNE Resultado

FIMFUNCAO

INICIO
ESCREVA ("1° Bimestre : ")
LEIA (Bim1)

ESCREVA ("2° Bimestre : ")
LEIA (Bim2)

ESCREVA ("3° Bimestre : ")
LEIA (Bim3)

ESCREVA ("4° Bimestre : ")
LEIA (Bim4)

// Chamada da função
MediaS := MediaSimples(Bim1, Bim2, Bim3, Bim4)
MediaP := MediaPonderada(Bim1, Bim2, Bim3, Bim4)

ESCREVAL
```

	Anotações

```
ESCREVAL ("Média Simpes: ", MediaS)
ESCREVAL ("Média Ponderada: ", MediaP)

FIMALGORITMO
```

Regras de Nomenclatura

Não podem ser iguais a palavras reservadas;

Devem possuir como primeiro caractere uma letra ou sublinhado '_' (os outros caracteres podem ser letras, números e sublinhado);

Não podem conter espaços em branco;

Na sintaxe do português estruturado, não há diferença entre letras maiúsculas de minúsculas (NOME é o mesmo que nome).

Anotações

Exercícios

1) Associe:

Constante	Corresponde a uma posição de memória cujo valor pode variar ao longo do tempo durante a execução do programa.
Variável	Corresponde a uma posição de memória cujo valor é constante (não pode variar) ao longo do tempo durante a execução do programa.

2) Preencha as lacunas com **I** para **Inteiro**, **R** para **Real**, **C** para **Caractere** e **L** para **Lógico**.

( )	“São Paulo”
( )	FALSO
( )	1,982
( )	“007”
( )	10,333
( )	127
( )	549
( )	VERDADEIRO

3) Para os nomes de indicadores abaixo, preencha as lacunas com **V** para **Válido**, e **I** para **Inválido**.

( )	%Desconto
( )	_Salario
( )	1Trim
( )	ALGORITMO
( )	Bim1
( )	Código
( )	FIMPROCEDIMENTO
( )	MENSAGEM

Anotações

Capítulo 05 - Operadores

Objetivos

Neste capítulo você irá aprender:

- Atribuição
- Concatenação
- Aritméticos
- Comparação
- Lógicos
- Precedência de Operadores
- Linearização de Expressões

Exercícios

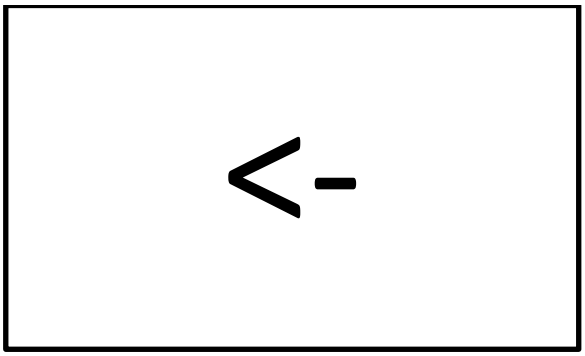
Os exemplos e exercícios utilizados neste capítulo estão na pasta: **Capítulo 05 - Operadores.**

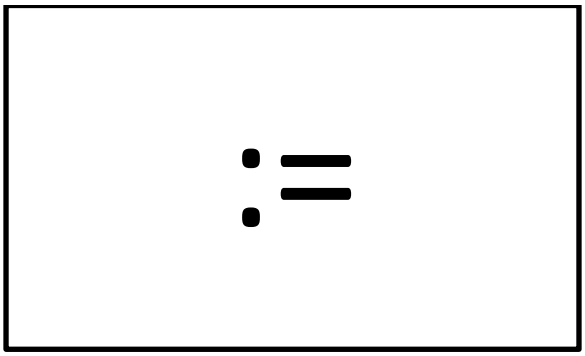
Duração

Este capítulo levará aproximadamente 45 minutos

	Anotações

Atribuição





Um comando de atribuição permite-nos fornecer um valor a certa variável, onde o tipo dessa informação deve ser compatível com o tipo da variável, isto é, somente podemos atribuir um valor lógico a uma variável capaz de comportá-lo, ou seja, uma variável declarada do tipo lógico. O comando de atribuição é uma seta apontando para a variável ou dois pontos e o sinal de igual.

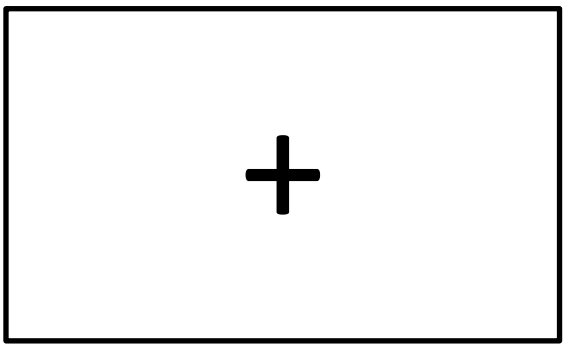
Exemplo

```
ALGORITMO "Atribuição"

VAR

// Números inteiros
Idade : INTEIRO
Quantidade : INTEIRO

// Números com decimais
Salario : REAL
Comissao : NUMERICO

// Textos
Nome : CARACTER
Cidade : LITERAL

// Valores Lógicos (Verdadeiro / Falso)
VIP : LOGICO
Aprovado : LOGICO
```

	Anotações

```
INICIO
// Seção de Comandos

Idade <- 28
Quantidade <- 3
Salario <- 15897.33
Comissao := 0.03
Nome := "José Luís Gomes"
Cidade := "São Paulo"
VIP := FALSO
Aprovado := VERDADEIRO

FIMALGORITMO
```

Concatenação



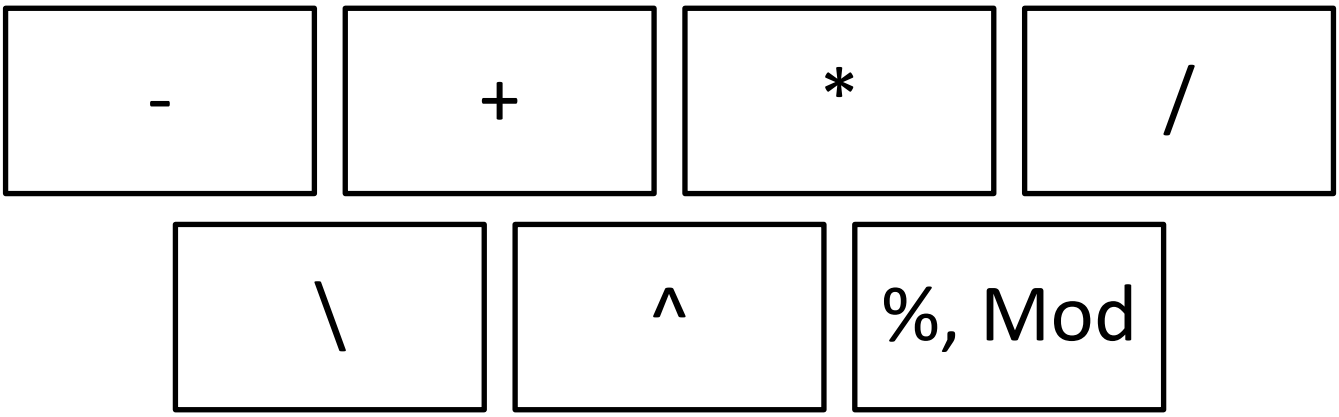
Exemplo

```
ALGORITMO "Concatenação"

// Seção de Declarações
VAR
PrimeiroNome : CARACTER
UltimoSobrenome : CARACTER
NomePassaporte : CARACTER

INICIO
// Seção de Comandos
```

	Anotações

```
PrimeiroNome := "José"
UltimoSobrenome := "Gomes"

NomePassaporte := UltimoSobrenome + ", " + PrimeiroNome

ESCREVA (NomePassaporte)

FIMALGORITMO
```

Aritméticos



O caractere (*) é adotado na maioria das linguagens de programação para representar a operação de multiplicação, ao invés do caractere (x), devido à possibilidade da ocorrência do mesmo no nome de variáveis. O símbolo (^) é adotado para representar a operação de Exponenciação, e em muitas literaturas será visto o símbolo (**) para esta finalidade, mas será adotado preferencialmente o símbolo (^).

Exemplo

```
ALGORITMO "Aritméticos"

// Seção de Declarações
VAR
Numero1, Numero2 : INTEIRO

INICIO
// Seção de Comandos

ESCREVA ("1° Número : ")
LEIA (Numero1)
```

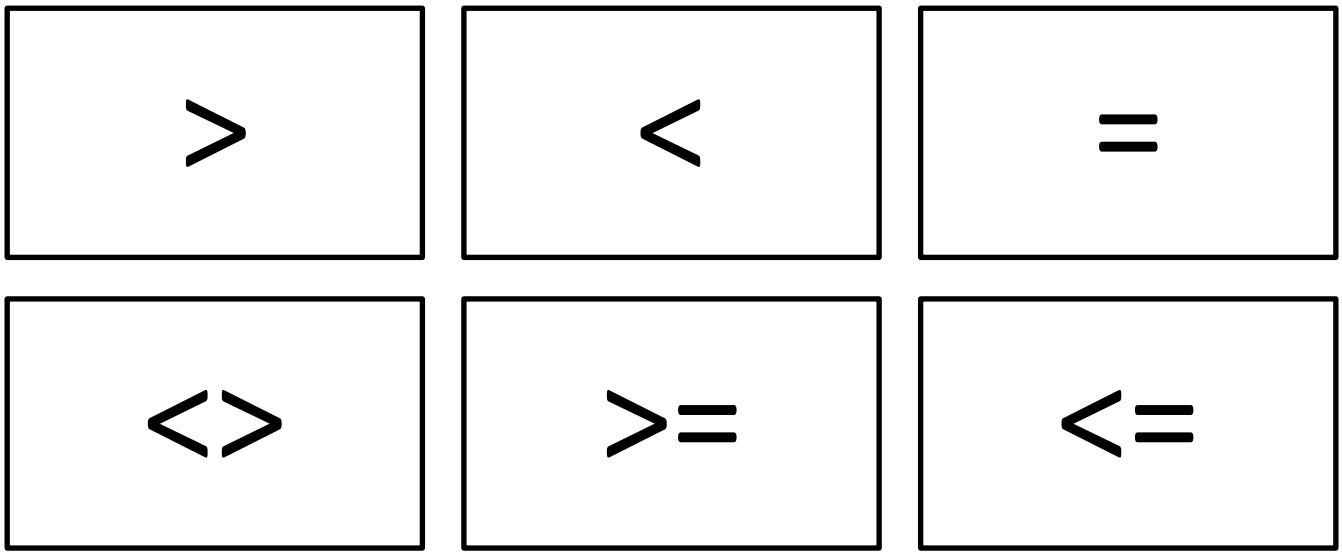
	Anotações

```
ESCREVA ("2° Número : ")
LEIA (Numero2)

ESCREVAL ( "Operadores Aritméticos" )

ESCREVAL

ESCREVAL ( "Soma" )
ESCREVAL ( Numero1, " +", Numero2, " =", Numero1 + Numero2 )

ESCREVAL

ESCREVAL ( "Subtração" )
ESCREVAL ( Numero1, " -", Numero2, " =", Numero1 - Numero2 )

ESCREVAL

ESCREVAL ( "Multiplicação" )
ESCREVAL ( Numero1, " *", Numero2, " =", Numero1 * Numero2 )

ESCREVAL

ESCREVAL ( "Divisão" )
ESCREVAL ( Numero1, " /", Numero2, " =", Numero1 / Numero2 )

ESCREVAL

ESCREVAL ( "Divisão de Inteiros" )
ESCREVAL ( Numero1, " \", Numero2, " =", Numero1 \ Numero2 )

ESCREVAL

ESCREVAL ( "Resto (Módulo Aritmético)" )
ESCREVAL ( Numero1, " MOD", Numero2, " =", Numero1 MOD Numero2)
ESCREVAL
```

	Anotações

```
ESCREVAL ( "Potência" )
ESCREVAL ( Numero1, " ^", Numero2, " =", Numero1 ^ Numero2 )

FIMALGORITMO
```

Comparação



Exemplo

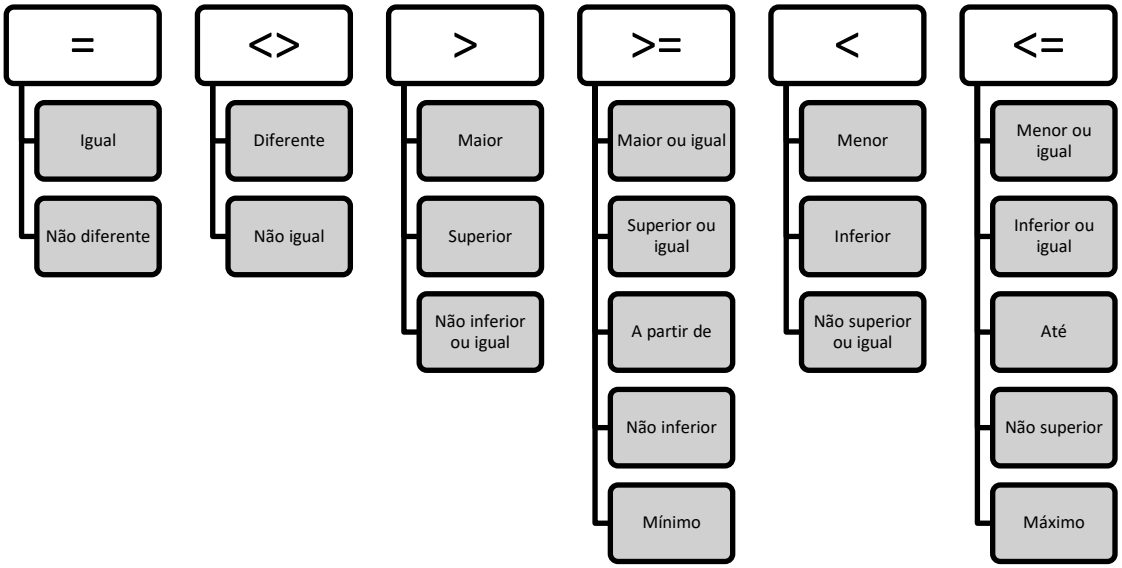
```
ALGORITMO "Comparação"

// Seção de Declarações
VAR
Numero1, Numero2 : REAL

INICIO
// Seção de Comandos

ESCREVA ("1° Número : ")
LEIA (Numero1)

ESCREVA ("2° Número : ")
LEIA (Numero2)
```

Anotações

```
ESCREVAL ( "Operadores de Comparação" )

ESCREVAL

ESCREVAL("Igual")
ESCREVAL(Numero1, " =", Numero2, " =", Numero1 = Numero2)
ESCREVAL

ESCREVAL("Diferente")
ESCREVAL(Numero1, " <>", Numero2, " =", Numero1 <> Numero2)
ESCREVAL

ESCREVAL("Maior")
ESCREVAL(Numero1, " >", Numero2, " =", Numero1 > Numero2)
ESCREVAL

ESCREVAL("Maior ou igual")
ESCREVAL(Numero1, " >=", Numero2, " =", Numero1 >= Numero2)
ESCREVAL

ESCREVAL("Menor")
ESCREVAL(Numero1, " <", Numero2, " =", Numero1 < Numero2)
ESCREVAL

ESCREVAL("Menor ou Igual")
ESCREVAL(Numero1, " <=", Numero2, " =", Numero1 <= Numero2)

FIMALGORITMO
```

Anotações

Termo *vs* Operador



Ao transcrever um texto para expressões usando os operadores fique atento para o diagrama acima. Uma vez que nem sempre o texto será explicito quanto ao uso dos operadores.

Lógicos

NAO	E	OU
XOU	EQV	IMP

Anotações

Exemplo

```
ALGORITMO "Lógicos"

// Seção de Declarações
VAR

INICIO
// Seção de Comandos

    ESCREVAL("Operadores Lógicos")
    ESCREVAL

    ESCREVAL("NAO")
    ESCREVAL("NAO FALSO", "=", NAO FALSO)
    ESCREVAL("NAO VERDADEIRO", "=", NAO VERDADEIRO)
    ESCREVAL

    ESCREVAL("E")
    ESCREVAL("VERDADEIRO e VERDADEIRO", "=", VERDADEIRO e
VERDADEIRO)
    ESCREVAL("VERDADEIRO e FALSO", "=", VERDADEIRO e FALSO)
    ESCREVAL("FALSO e VERDADEIRO", "=", FALSO e VERDADEIRO)
    ESCREVAL("FALSO e FALSO", "=", FALSO e FALSO)
    ESCREVAL

    ESCREVAL("OU")
    ESCREVAL("VERDADEIRO OU VERDADEIRO", "=", VERDADEIRO OU
VERDADEIRO)
    ESCREVAL("VERDADEIRO OU FALSO", "=", VERDADEIRO OU FALSO)
    ESCREVAL("FALSO OU VERDADEIRO", "=", FALSO OU VERDADEIRO)
    ESCREVAL("FALSO OU FALSO", "=", FALSO OU FALSO)
    ESCREVAL

    ESCREVAL("XOU")
    ESCREVAL("VERDADEIRO XOU VERDADEIRO", "=", VERDADEIRO XOU
VERDADEIRO)
```

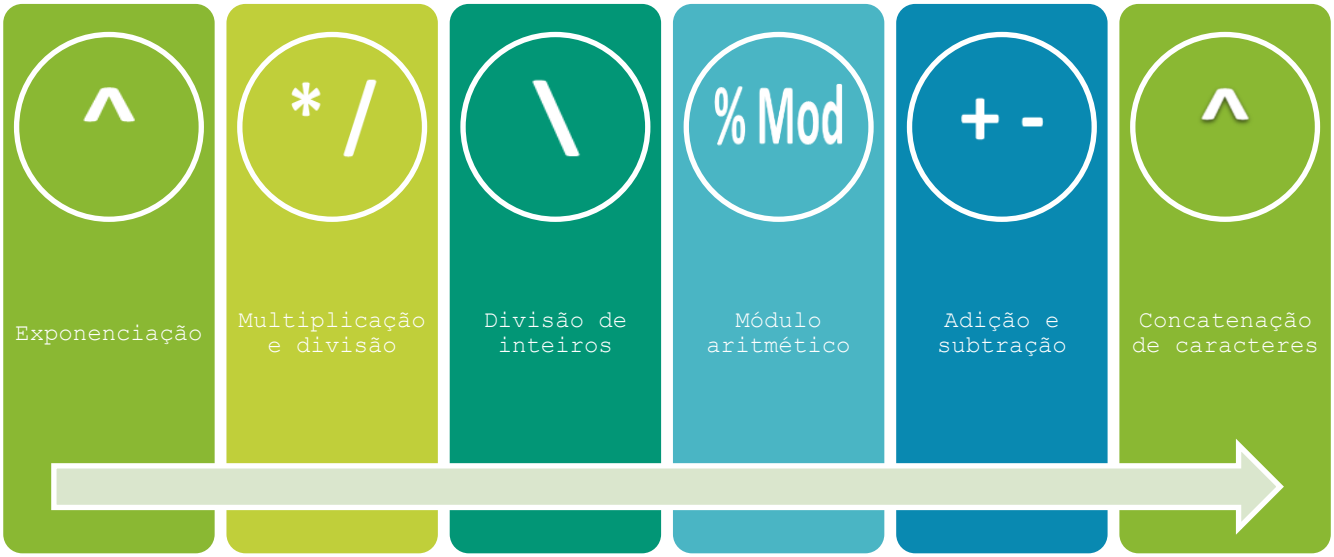
	Anotações

```
    ESCREVAL("VERDADEIRO XOU FALSO", "=", VERDADEIRO XOU FALSO)
    ESCREVAL("FALSO XOU VERDADEIRO", "=", FALSO XOU VERDADEIRO)
    ESCREVAL("FALSO XOU FALSO", "=", FALSO XOU FALSO)

FIMALGORITMO
```

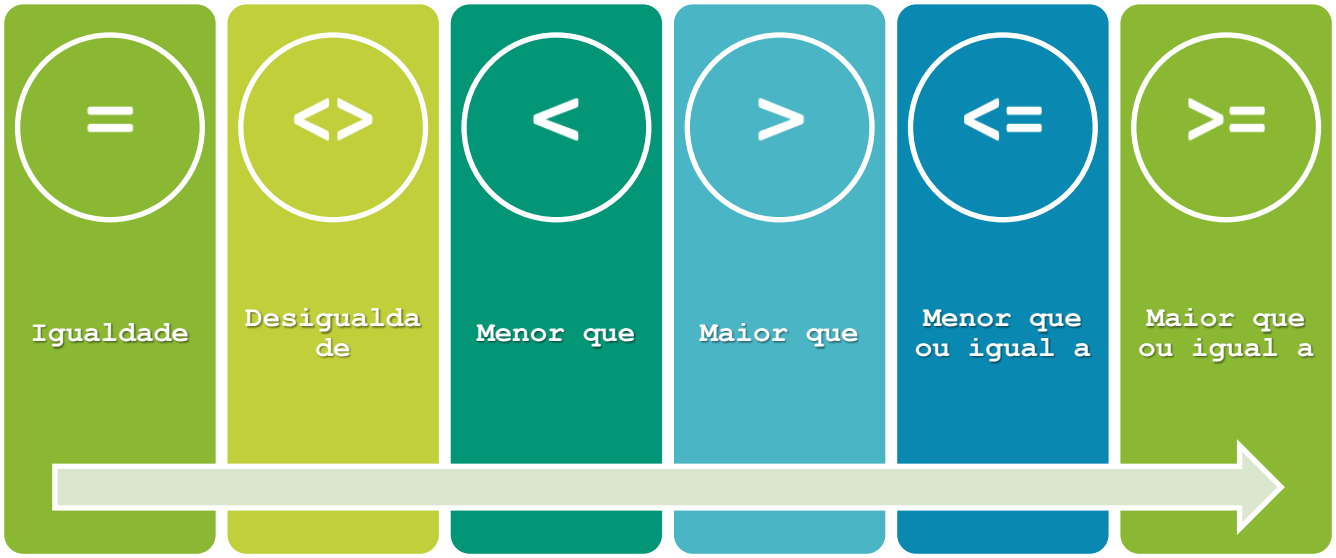
Anotações

Precedência de Operadores

Aritméticos



De comparação



Anotações

Lógicos

Linearização de Expressões

Para a construção de Algoritmos todas as expressões aritméticas devem ser linearizadas, ou seja, colocadas em linhas.

É importante também ressalvar o uso dos operadores correspondentes da aritmética tradicional para a computacional.

Tradicional

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Computacional

```
x = (-b+ (b^2-4*a*c)^0, 5)/(2*a)
x = (-b- (b^2-4*a*c)^0, 5)/(2*a)
```

	Anotações

Exercícios

1) Escreva as seguintes as seguintes expressões “linearizadas” na forma convencional:
   a) A = pi * r ^ 2

   b) a ^ 2 + b ^ 2 = c ^ 2

   c) x’ = (-b + (b ^ 2 - 4 * a * c) ^(1/2)) / 2 * a

   d) (2/y -x) ^ 2 = (2/y) ^ 2 - 2 * (2/y) * x + x^2

	Anotações

2) Resolva as expressões lógicas abaixo, determinando o resultado da expressão, sendo ele, **VERDADEIRO** ou **FALSO**.

( )	((3 + 5) / 2) > 0)
( )	(3 * 3 > 10) ou (2 + 2 < 10)
( )	(10 <> 5) ou (1 + 1 = 0)
( )	(2 + 3 >= 5) e (18 / 3 < 7)
( )	(1 + 1 = 0) e (0 - 1 > 0)

3) Preencha com o operador adequado:

Não diferente	
Não igual	
Não inferior ou igual	
A partir de	
Mínimo	
Não superior ou igual	
Até	
Máximo	

4) Preencha com o operador adequado:

^	
v	
~	
→	
↔	
v	

	Anotações

Capítulo 06 - Interação Básica

Objetivos

Neste capítulo você irá aprender:

- Estrutura Sequencial
- Linhas de Comentário
- Comandos de Saída (Output)
- Comandos de Entrada (Input)

Exercícios

Os exemplos e exercícios utilizados neste capítulo estão na pasta: **Capítulo 06 - Interação Básica.**

Duração

Este capítulo levará aproximadamente 45 minutos

	Anotações

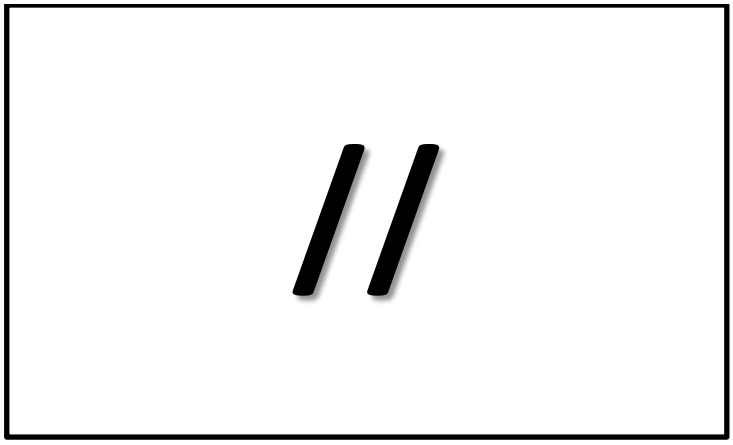
Estrutura Sequencial

```
ALGORITMO "Estrutura Sequencial"

// Seção de Declarações

VAR

INICIO

// Seção de Comandos

FIMALGORITMO
```

É o conjunto de ações primitivas que serão executadas numa sequência linear de cima para baixo e da esquerda para direita, isto é, na mesma ordem em que foram escritas.

Como podemos perceber, todas as ações devem pular uma linha, o que objetiva separar uma ação de outra.

Linhas de Comentário




```
ALGORITMO "Comentário"

// Função :
// Autor :
// Data : 12/08/2013
// Seção de Declarações

VAR
```

Anotações

```
INICIO

// Seção de Comandos

FIMALGORITMO
```

Comandos de Saída (Output)

ESCREVA ()	ESCREVAL ()

Da mesma forma que nosso algoritmo precisa de informações, o usuário precisa de respostas as suas perguntas, para darmos estas respostas usamos um comando de saída de dados (Output) para informar a resposta.

Sintaxe

```
ESCREVA (<expressão ou identificador ou constante>, <expressão ou identificador ou constante>, ... , <expressão ou identificador ou constante>)
ESCREVAL (<expressão ou identificador ou constante>, <expressão ou identificador ou constante>, ... , <expressão ou identificador ou constante>)
```

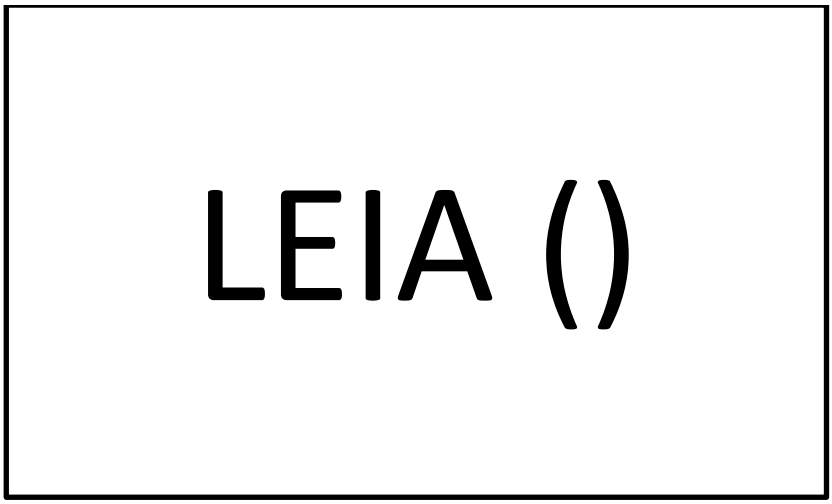
Exemplo

```
ALGORITMO "Comandos de Saída"

// Seção de Declarações

VAR

INICIO
```

	Anotações

```
// Seção de Comandos

// na mesma linha
ESCREVA("Texto 1")
ESCREVA("Texto 2")

// Pula uma linha
ESCREVAL

// na próxima linha
ESCREVAL("Texto 3")
ESCREVAL("Texto 4")

FIMALGORITMO
```

Comandos de Entrada (Input)



Para que nossos algoritmos funcionem, em quase todos os casos precisaremos de informações que serão fornecidas somente após o algoritmo pronto, e que sempre estarão mudando de valores, para que nossos algoritmos recebem estas informações, devemos então construir entradas de dados, pelas quais o usuário (pessoa que utilizar o programa) poderá fornecer todos os dados necessários.

Sintaxe

```
LEIA (<identificador>)
```

Exemplo

	Anotações

```
ALGORITMO "Comandos de Entrada"

// Seção de Declarações

VAR
Texto1, Texto2 : CARACTER

INICIO

// Seção de Comandos

// na mesma linha
LEIA(Texto1, Texto2)

FIMALGORITMO
```

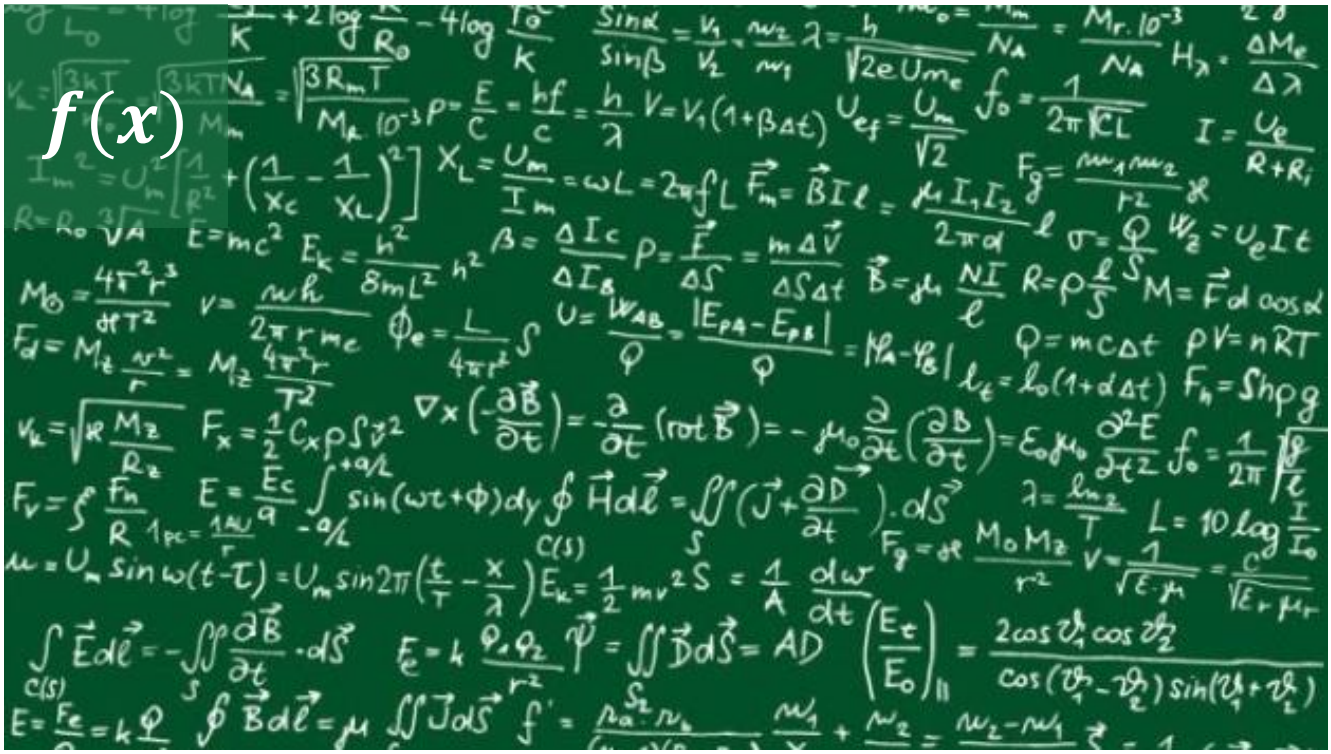
Anotações

Exercícios

1) Escreva um programa que leia dois valores A e B, calcule a soma entre eles e exiba o resultado para o usuário.
2) Faça um programa que leia um número inteiro informado pelo usuário, calcule o seu quadrado e exiba o resultado para o usuário.
3) Sabendo que a fórmula para calcular a área de uma circunferência é $\boldsymbol{A = \pi r^2}$, elabore um programa que leia o valor do raio da circunferência, calcule a área e exiba o resultado para o usuário.
4) Faça um algoritmo que leia um valor X e um valor N, e calcule o produto de X por 2 elevado a N.
5) Faça um algoritmo que leia a idade de uma pessoa expressa em anos, meses e dias e exiba apenas em dias.

Anotações

Capítulo 07 - Funções Predefinidas



Objetivos

Neste capítulo você irá aprender:

- Conceito de Funções
- Funções Texto
- Funções Matemáticas e Trigonométricas

Exercícios

Os exemplos e exercícios utilizados neste capítulo estão na pasta: **Capítulo 07 - Funções Predefinidas.**

Duração

Este capítulo levará aproximadamente 45 minutos

	Anotações

Conceito de Funções

Consideramos o termo “Função” em lógica de programação, como um relacionamento especial entre valores. Ou seja, dado um valor de entrada (Input) teremos um valor de saída (Output). Essa relação entre o valor de entrada e o valor de saída é onde está a resolução do problema.

Exemplos

Valor de Entrada (Input)	Relacionamento	Valor de Saída (Output)
1	x 5	**5**
3	x 5	**15**
5	x 5	**25**
7	x 5	**35**
13	x 5	**65**

Funções Texto

Função	Descrição
ASC (s: caracter)	Retorna um código ASCII para o primeiro caractere de uma cadeia de texto
COMPR (c: caracter)	Retorna o número de caracteres em uma cadeia de texto
COPIA (c: caracter, posini, posfin: inteiro)	Retorna um número específico de caracteres de uma cadeia de texto começando na posição especificada
MAIUSC (c: caracter)	Converte o texto em maiúsculas
MINUSC (c: caracter)	Converte o texto em minúsculas
NUMPCARAC (n: inteiro ou real)	Converte um número inteiro ou real para caractere
POS (subc, c: caracter)	Retorna aposição de texto dentro de outro

Exemplos

```
ALGORITMO "Funções Texto"

// Seção de Declarações
VAR

INICIO
// Seção de Comandos

ESCREVAL("ASC() - Retorna número ASCII de um o caractere")
ESCREVAL("Excel - CÓDIGO()")
ESCREVAL("ASC('@') = ", ASC("@"))
ESCREVAL
```

Anotações

```
ESCREVAL("COMPR() - Retorna o comprimento de um texto")
ESCREVAL("Excel - NÚM.CARACT()")
ESCREVAL("COMPR('   Bom   dia   ') = ", COMPR("   Bom   dia   "))
ESCREVAL

ESCREVAL("COPIA() - Retorna 'n' caracteres à partir de uma posição a
esquerda do texto")
ESCREVAL("Excel - EXT.TEXTO()")
ESCREVAL("COPIA('São Paulo', 3, 6) = ", COPIA("São Paulo", 3, 6))
ESCREVAL

ESCREVAL("MAIUSC() - Retorna o texto em Maiúscula")
ESCREVAL("Excel - Maiúscula()" )
ESCREVAL("MAIUSC('bOm dIa') = ", MAIUSC("bOm dIa"))
ESCREVAL

ESCREVAL("MINUSC() - Retorna o texto em Minúscula")
ESCREVAL("Excel - Minúscula()")
ESCREVAL("MINUSC('bOm dIa') = ", MINUSC("bOm dIa"))
ESCREVAL

ESCREVAL("NUMPCARAC() - Converte um número inteiro ou real para
caractere")
ESCREVAL("Excel - sem Correspondência")
ESCREVAL("NUMPCARAC(10.32) = ", NUMPCARAC(10.32))

ESCREVAL("POS() - Retorna a posição de um subtexto em um texto")
ESCREVAL("Excel - PROCURAR()")
ESCREVAL("POS('Paulo', 'São Paulo') = ", POS("Paulo", "São Paulo"))
ESCREVAL

FIMALGORITMO
```

	Anotações

Funções Matemáticas e Trigonométricas

Função	Descrição
ABS (valor: real)	Retorna o valor absoluto de um número
ARCCOS (valor: real)	Retorna o arco cosseno de um número
ARCSEN (valor: real)	Retorna o arco seno de um número
ARCTAN (valor: real)	Retorna o arco tangente de um número
COS (valor: real)	Retorna o cosseno de um número
COTAN (valor: real)	Retorna a cotangente de um número
EXP (<base>, <expoente>)	Retorna e elevado à potência de um número especificado
GRAUPRAD (valor: real)	Converte graus em radianos
INT (valor: real)	Retorna a parte inteira de um número
LOG (valor: real)	Retorna o logaritmo de base 10 de um número
LOGN (valor: real)	Retorna o logaritmo de um número de uma base especificada
PI: real	Retorna o valor de Pi
QUAD (valor: real)	Retorna o resultado de um número elevado a ao quadrado
RADPGRAU (valor: real)	Converte radianos em graus
RAIZQ (valor: real)	Retorna uma raiz quadrada
RAND: real	Retorna um número aleatório entre 0 e 1
RANDI (limite: inteiro)	Retorna um número aleatório entre os números especificados
SEN (valor: real)	Retorna o seno de um número
TAN (valor: real)	Retorna a tangente de um número

Exemplos

```
ALGORITMO "Funções Inteiro"

// Seção de Declarações
VAR

INICIO
// Seção de Comandos

ESCREVAL ("ABS() - Retorna o valor absoluto de um número.")
ESCREVAL ("Excel - ABS()")
ESCREVAL ("ABS(10) = ", ABS(10))
ESCREVAL ("ABS(0) = ", ABS(0))
ESCREVAL ("ABS(-10) = ", ABS(-10))
ESCREVAL
```

Anotações

```
ESCREVAL ("ARCCOS() - Retorna o arco cosseno de um número.")
ESCREVAL ("Excel - ATAN()")
ESCREVAL ("ARCCOS(1,3) = ", ARCCOS(1.3))
ESCREVAL

ESCREVAL ("ARCSEN() - Retorna o arco seno de um número.")
ESCREVAL ("Excel - ATAN()")
ESCREVAL ("ARCSEN(1,3) = ", ARCSEN(1.3))
ESCREVAL

ESCREVAL ("ARCTAN() - Retorna o arco tangente de um número.")
ESCREVAL ("Excel - ATAN()")
ESCREVAL ("ARCTAN(1,3) = ", ARCTAN(1.3))
ESCREVAL

ESCREVAL ("COS() - Retorna o cosseno de um ângulo.")
ESCREVAL ("Excel - COS()")
ESCREVAL ("COS(1,3) = ", Cos(1.3))
ESCREVAL

ESCREVAL ("COTAN() - Retorna a cotangente de um ângulo.")
ESCREVAL ("Excel - TAN()")
ESCREVAL ("COTAN(1,3) = ", COTAN(1.3))
ESCREVAL

ESCREVAL ("EXP() - Retorna a Potenciação.")
ESCREVAL ("Excel - EXP()")
ESCREVAL ("EXP(2,4) = ", Exp(2, 4))

ESCREVAL ("GRAUPRAD() - Converte grau para radiano.")
ESCREVAL ("Excel - ATAN()")
ESCREVAL ("GRAUPRAD(10) = ", GRAUPRAD(10))
ESCREVAL

ESCREVAL("INT() - Retorna o menor inteiro de um número")
```

Anotações

```
ESCREVAL("Excel - INT()")
ESCREVAL("INT(1.99) = ", INT(1.99))
ESCREVAL("INT(-1.99) = ", INT(-1.99))
ESCREVAL

ESCREVAL ("LOG() - Retorna o Logaritmo de base 10.")
ESCREVAL ("Excel - LOG()")
ESCREVAL ("LOG(10) = ", Log(10))
ESCREVAL

ESCREVAL ("LOGN() - Retorna o Logaritmo natural (ln).")
ESCREVAL ("Excel - LOG()")
ESCREVAL ("LOGN(10) = ", LOGN(10))
ESCREVAL

ESCREVAL ("PI() - Retorna o Valor Pi.")
ESCREVAL ("Excel - PI()")
ESCREVAL ("PI = ", Pi)
ESCREVAL

ESCREVAL ("QUAD() - Retorna o valor Elevado ao quadrado.")
ESCREVAL ("Excel - EXP()")
ESCREVAL ("QUAD(2) = ", QUAD(2))

ESCREVAL ("RADPGRAU() - Converte Radiano para grau.")
ESCREVAL ("Excel - ATAN()")
ESCREVAL ("RADPGRAU(10) = ", RADPGRAU(10))

ESCREVAL ("RAIZQ() - Retorna a raiz quadrada de um número.")
ESCREVAL ("Excel - RAIZ()")
ESCREVAL ("RAIZQ(81) = ", RAIZQ(81))
ESCREVAL ("RAIZQ(9) = ", RAIZQ(9))
ESCREVAL ("81 ^ 0.5 = ", 81 ^ 0.5)
ESCREVAL ("9 ^ 0.5 = ", 9 ^ 0.5)
ESCREVAL ("27 ^ (1/3) = ", 27 ^ (1 / 3))
ESCREVAL
```

	Anotações

```
ESCREVAL ("RAND() - Retorna número aleatório decimal entre 0 e 1.")
ESCREVAL ("Excel - ALEATÓRIO()")
ESCREVAL ("RAND() = ", RAND)
ESCREVAL ("RAND() = ", RAND)
ESCREVAL ("RAND() = ", RAND)
ESCREVAL

ESCREVAL("RANDI() - Gerador de números inteiros aleatórios com um
limite determinado")
ESCREVAL("Excel - ALEATÓRIOENTRE()")
ESCREVAL("RANDI() = ", RANDI(10))
ESCREVAL("RANDI() = ", RANDI(30))
ESCREVAL("RANDI() = ", RANDI(47))

ESCREVAL ("SEN() - Retorna o seno de um ângulo.")
ESCREVAL ("Excel - SEN()")
ESCREVAL ("SEN(1,3) = ", SEN(1.3))
ESCREVAL

ESCREVAL ("TAN() - Retorna a tangente de um ângulo.")
ESCREVAL ("Excel - TAN()")
ESCREVAL ("TAN(1,3) = ", Tan(1.3))
ESCREVAL

FIMALGORITMO
```

	Anotações

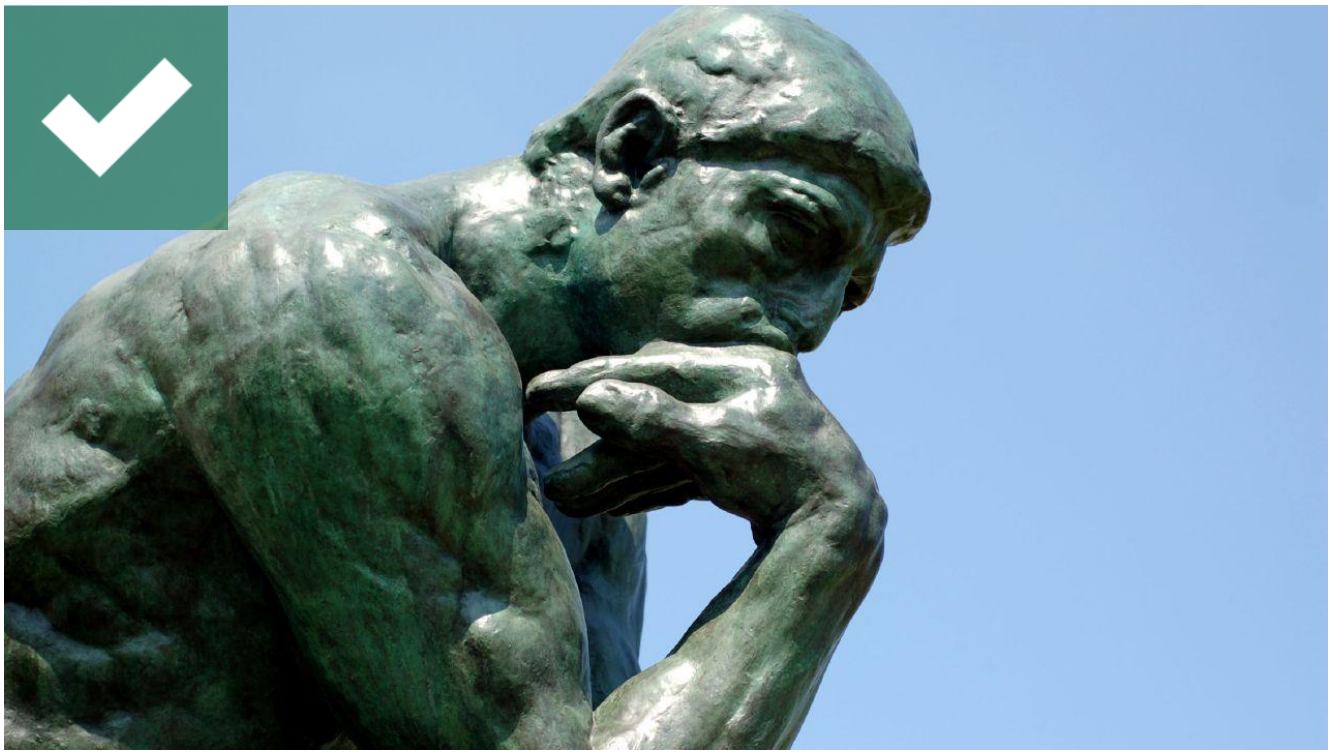
Exercícios

1) Associe:

COMPR (c : caracter)	Retorna o valor absoluto de um número
RANDI (limite : inteiro)	Retorna o número de caracteres em uma cadeia de texto
INT (valor : real)	Retorna e elevado à potência de um número especificado
ABS (valor : real)	Retorna a parte inteira de um número
EXP (<base>, <expoente>)	Retorna um número aleatório entre os números especificados

Anotações

Capítulo 08 - Laços de Validação



Objetivos

Neste capítulo você irá aprender:

- SE ... ENTAO ... SENAO
- SE ... ENTAO
- SE ... ENTAO ... SE ... ENTAO
- ESCOLHA... CASO

Exercícios

Os exemplos e exercícios utilizados neste capítulo estão na pasta: **Capítulo 08 - Laços de Validação.**

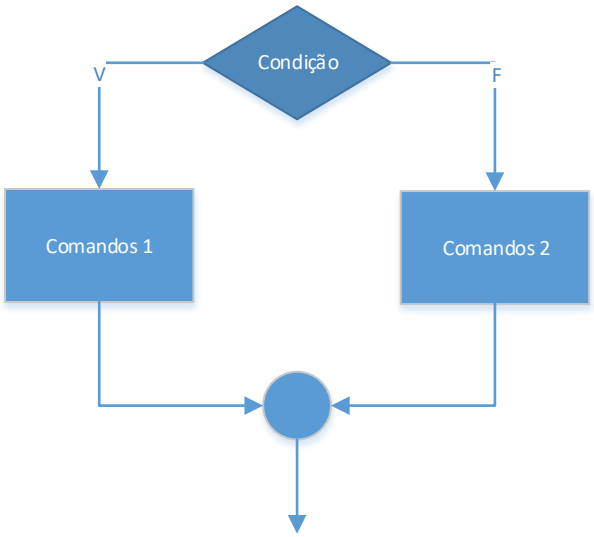
Duração

Este capítulo levará aproximadamente 100 minutos

	Anotações

SE ... ENTAO ... SENAO

Fluxograma



Sintaxe

```
SE <Condição> ENTAO
        <Comandos 1>
SENAO
        <Comandos 2>
FIMSE
```

Condição é uma expressão lógica (composta por uma ou mais proposições), e assim quando inspecionada, gera resultado falso ou verdadeiro.

Se Verdadeiro, a ação primitiva sob a cláusula será executada; caso contrário, isto é, se for Falso, a ação primitiva sob a cláusula será executada. Após isto, encerra a condição com o comando **FIMSE**.

Exemplo

```
ALGORITMO "Se Então Senão"

VAR
Nota1 : REAL
Nota2 : REAL
Media : REAL
```

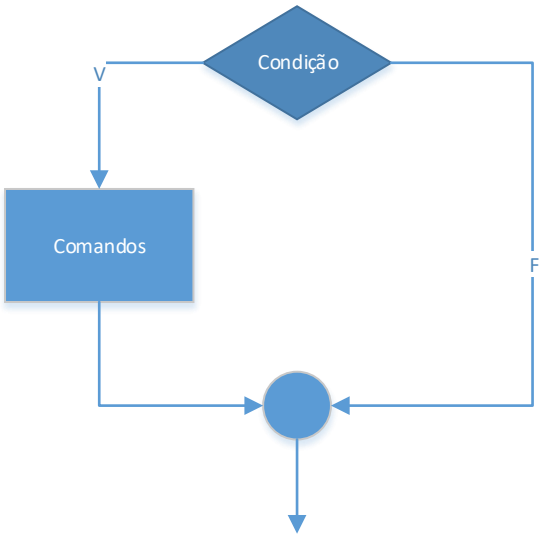
	Anotações

```
INICIO

ESCREVA("Digite o valor da primeira nota : ")
LEIA (Nota1)

ESCREVA("Digite o valor da segunda nota : ")
LEIA (Nota2)

Media := (Nota1 + Nota2) / 2

ESCREVAL ("A média é =", Media)

SE Media >= 7 ENTAO
   // Instrução com condição verdadeira
   ESCREVAL ("Aluno aprovado!")
SENAO
   // Instrução com condição falsa
   ESCREVAL ("Aluno não aprovado!")
FIMSE

FIMALGORITMO
```

	Anotações

SE ... ENTAO

Fluxograma



Sintaxe

```
SE <Condição> ENTAO
        <Comandos>
FIMSE
```

Se Verdadeiro, a ação primitiva sob a cláusula será executada; caso contrário, encerra o comando, neste caso, sem executar nenhum comando. Após isto, encerra a condição com o comando **FIMSE**.

Exemplo

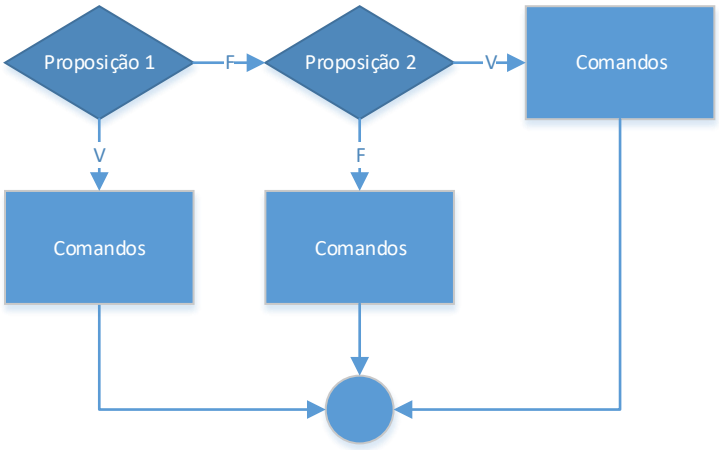
```
ALGORITMO "Se Então"

VAR
Nota1 : REAL
Nota2 : REAL
Media : REAL

INICIO
ESCREVA("Digite o valor da primeira nota : ")
LEIA (Nota1)
```

	Anotações

```
ESCREVA("Digite o valor da segunda nota : ")
LEIA (Nota2)

Media := (Nota1 + Nota2)/2

ESCREVAL ("A média é =", Media)

SE Media >= 7 ENTAO
   ESCREVAL ("Aluno aprovado!")
FIMSE

FIMALGORITMO
```

SE ... ENTAO ... SE ... ENTAO

Fluxograma



Sintaxe

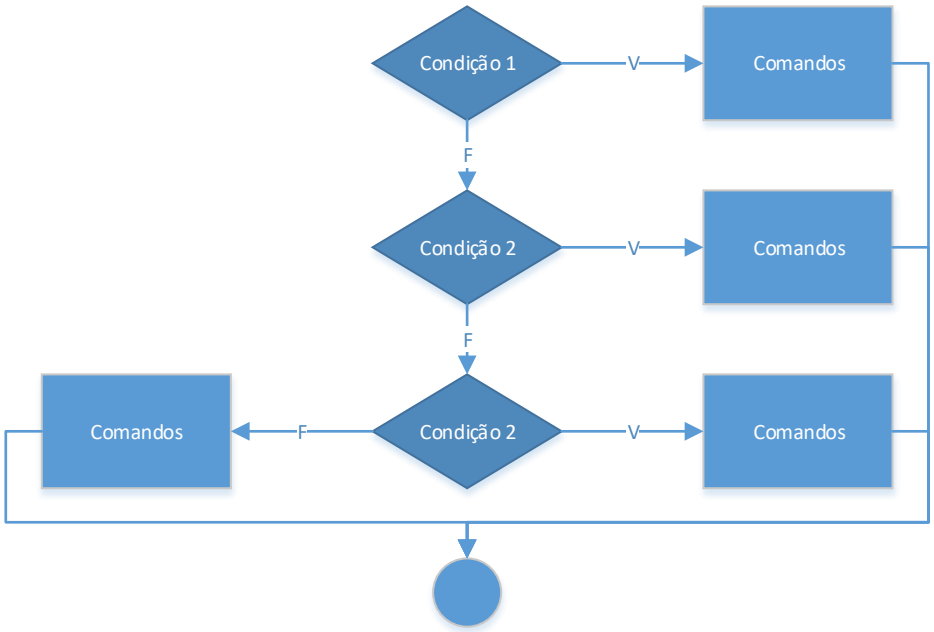
```
SE <Condição 1> ENTÃO
        <Comandos 1>
SENÃO
        SE <Condição 2> ENTÃO
           <Comandos 2>
        SENÃO
           <Comandos 3>
        FIMSE
```

	Anotações

```
FIMSE
```

ESCOLHA... CASO

Fluxograma



Sintaxe

```
ESCOLHA <Condição>
        CASO V1 : <Comandos>
        CASO V2 : <Comandos>
        CASO V3 : <Comandos>
FIM ESCOLHA
```

Esta estrutura evita que façamos muitos blocos se, quando o teste será sempre em cima da mesma variável.

Exemplo

```
ALGORITMO "Escolha Caso"

// Seção de Declarações
VAR
Opcao : INTEIRO

INICIO
```

	Anotações

```
ESCREVAL ("Digite '1', para praia")
ESCREVAL ("Digite '2', para cinema")
ESCREVAL ("Digite '3', para churrasco")
LEIA (Opcao)

ESCOLHA Opcao
CASO 1
   ESCREVA ("Sair de casa às 8 horas da manhã.")
CASO 2
   ESCREVA ("Sair de casa às 2 horas da tarde.")
CASO 3
   ESCREVA ("Sair de casa ao meio-dia.")
OUTROCASO //caso escolha opção diferente das anteriores
   ESCREVA ("Já que não optou, fique em casa mesmo e leia um
livro.")
FIMESCOLHA

FIMALGORITMO
```

Anotações

Exercícios

1) Elabore um algoritmo que calcule a multa paga por um pescador que ultrapassar a quantidade de quilos estabelecida por lei, a saber:
   - A quantidade de peixe por pessoa é de 50 Kg
   - A multa por quilo excedente é de R$ 4,00
2) Crie um programa que dada a idade de um jogador classifique-o em uma das seguintes categorias:

Infantil	5 a 10 anos
Juvenil	11 a 17 anos
Adulto	Maiores de 18 anos

3) Faça um programa em que o usuário informa à hora que inicia o seu turno de trabalho e é exibido na tela se é manhã, tarde ou noite:

Manhã	05h00min às 13h00min
Tarde	13h00min às 21h00min
Noite	21h00min às 05h00min

4) Elabore um programa que calcule quanto o cliente de um posto de gasolina irá pagar de acordo com a opção do combustível escolhido e a quantidade de litros comprados. Utilize os dados da tabela para desenvolver o algoritmo;

Código	Combustível	Preço
A	Álcool	1, 56/l
G	Gasolina	2, 56/l
D	Diesel	1, 81/l

5) Faça um algoritmo que calcule o valor da conta de luz de uma pessoa. Sabe-se que o cálculo da conta de luz segue a tabela abaixo:

Tipo de Cliente	Valor do KW/h
Residência	0, 83
Comércio	0, 62
Indústria	2, 03

Anotações

Capítulo 09 - Laços de Repetição

Objetivos

Neste capítulo você irá aprender:

- ENQUANTO ... FACA
- REPITA ... ATE
- PARA ... DE ... ATE ... PASSO ... FACA

Exercícios

Os exemplos e exercícios utilizados neste capítulo estão na pasta: **Capítulo 09 - Laços de Repetição.**

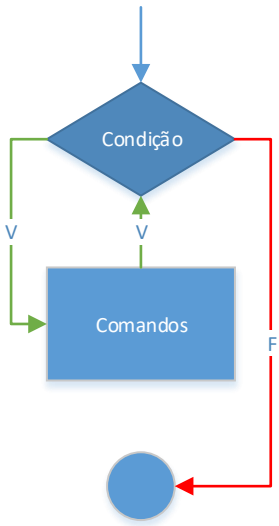
Duração

Este capítulo levará aproximadamente 100 minutos

	Anotações

ENQUANTO ... FACA

Fluxograma



Sintaxe

```
ENQUANTO <Condição> FACA
        <Comandos>
FIMENQUANTO
```

Esta estrutura faz seu teste de parada antes do bloco de comandos, isto é, o bloco de comandos será repetido, até que a condição seja **FALSO**. Os comandos de uma estrutura **ENQUANTO ... FACA** poderá ser executada uma vez, várias vezes ou nenhuma vez.

Exemplo

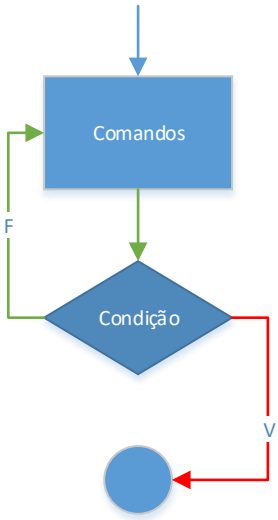
```
ALGORITMO "Enquanto Faça"

VAR

// Declaração das variáveis do algoritmo
Opcao: CARACTER

INICIO

// Início da estrutura de repetição
```

	Anotações

```
Opcao := "S"

ENQUANTO Opcao = "S" FACA

   ESCREVA("Continuar: ")
   LEIA (Opcao)

   // Loop
FIMENQUANTO

FIMALGORITMO
```

REPITA ... ATE

Fluxograma



Sintaxe

```
REPITA
        <Comandos>
ATE <Condição>
```

Exemplo

```
ALGORITMO "Repita Até"

VAR
```

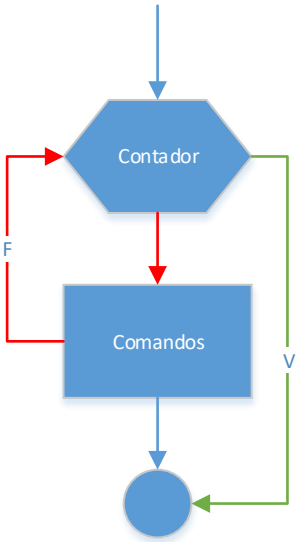
	Anotações

```
// Declaração das variáveis do algoritmo
Opcao: CARACTER

INICIO

// Início da estrutura de repetição

Opcao := "S"

REPITA

   ESCREVA("Continuar: ")
   LEIA (Opcao)

   // Loop
ATE Opcao = "N"

FIMALGORITMO
```

PARA ... DE ... ATE ... PASSO ... FACA

Fluxograma

	Anotações



Sintaxe

```
PARA <variável> := <valor> ATE <valor> PASSO <N> FACA
        < Bloco de comandos >
FIMPARA
```

Nas estruturas de repetição vistas até agora, acorrem casos em que se torna difícil determinar quantas vezes o bloco será executado. Sabemos que ele será executado enquanto uma condição for satisfeita - **ENQUANTO ... FACA**, ou até que uma condição seja satisfeita - **REPITA ... ATE**. A estrutura **PARA ... PASSO** repete a execução do bloco um número definido de vezes, pois ela possui limites fixos.

Exemplo

```
ALGORITMO "Para de até Passo Faça"

VAR

// Declaração do contador
Contador : INTEIRO

INICIO

PARA Contador DE 1 ATE 10 PASSO 1 FACA
   ESCREVAL (Contador)
FIMPARA
```

	Anotações

```
ESCREVAL

PARA Contador DE 1 ATE 10 FACA
   ESCREVAL (Contador)
FIMPARA
ESCREVAL

PARA Contador DE 10 ATE 1 PASSO -1 FACA
   ESCREVAL (Contador)
FIMPARA
ESCREVAL

FIMALGORITMO
```

Anotações

Exercícios

1) Desenvolva um programa que calcule e exiba a soma dos mil primeiros números inteiros.
2) Elaborar um programa que converta valores em graus Celsius para Fahrenheit de 1 a 100.
3) Escreva um algoritmo que calcule o fatorial (N!) de um determinado número fornecido pelo usuário.
4) Sabendo que a séria de Fibonacci é formada pela sequência 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ... e que seus termos, a partir do terceiro, é igual à soma dos dois anteriores, desenvolva um programa que exiba a série até o 50º termo.
5) A eleição do grêmio estudantil de uma escola possui três chapas candidatas. Os códigos de cada chapa são 11, 12, 13, os votos nulos é 14, os brancos 15. Elabore um algoritmo que calcule e escreva:
   O total de votos para cada chapa e o percentual de cada sobre uma sobre o total
   - O total de nulos e seu percentual sobre o total
   - O total de brancos e seu percentual sobre o total

Anotações

Capítulo 10 - Variáveis Indexadas

Objetivos

Neste capítulo você irá aprender:

- Variáveis Indexadas Unidimensionais (Vetores)
- Variáveis Indexadas Bidimensionais (Matrizes)

Exercícios

Os exemplos e exercícios utilizados neste capítulo estão na pasta: **Capítulo 10 - Variáveis Indexadas.**

Duração

Este capítulo levará aproximadamente 100 minutos

	Anotações

Variáveis Indexadas Unidimensionais (Vetores)

Sintaxe

```
<Identificador> : VETOR [<tamanho>] de < tipo >
```

Tamanho [VI ... VF]=> Vi= Valor inicial do índice e VF valor Final do índice.

Definição

Variáveis indexadas com uma única dimensão, também conhecidas como vetores, são referenciadas por um único índice.

Exemplo

```
ALGORITMO "Vetores"

// Seção de Declarações
VAR
NomeMes : VETOR [1..12] de CARACTER
NumeroMes : INTEIRO

INICIO
// Seção de Comandos

NomeMes[1] := "Janeiro"
NomeMes[2] := "Fevereiro"
NomeMes[3] := "Março"
NomeMes[4] := "Abril"
NomeMes[5] := "Maio"
NomeMes[6] := "Junho"
NomeMes[7] := "Julho"
NomeMes[8] := "Agosto"
NomeMes[9] := "Setembro"
NomeMes[10] := "Outubro"
NomeMes[11] := "Novembro"
NomeMes[12] := "Dezembro"

ESCREVA("Mês : ")
LEIA (NumeroMes)
```

	Anotações

```
ESCREVAL( NomeMes[NumeroMes] )

FIMALGORITMO
```

Para se atribuir um valor a um elemento do vetor devemos utilizar o seguinte padrão:

```
< Identificador>[<posição>] := <valor>
```

Variáveis Indexadas Bidimensionais (Matrizes)

Tamanho [VI ... VF]=> Vi= Valor inicial do índice e VF valor Final do índice.

Definição

Variáveis indexadas com duas dimensões, também conhecida como matrizes, são referenciadas por dois índices, cada qual começando por 1.

Sintaxe

```
<Identificador> : VETOR [<tamanho1>, <tamanho2>] de < TIPO >
```

Exemplo

```
ALGORITMO "Matrizes"
VAR
NomeEstacao : VETOR [1..4, 1..2] de CARACTER
NumeroEstacao : INTEIRO
NumIdioma : INTEIRO

INICIO
// Seção de Comandos

NomeEstacao[1, 1] :=  "Primavera"
NomeEstacao[2, 1] :=  "Verão"
NomeEstacao[3, 1] :=  "Outono"
NomeEstacao[4, 1] :=  "Inverno"

NomeEstacao[1, 2] :=  "Spring"
NomeEstacao[2, 2] :=  "Summer"
NomeEstacao[3, 2] :=  "Fall"
NomeEstacao[4, 2] :=  "Winter"
```

	Anotações

```
ESCREVA("Estação [1-4] : ")
LEIA (NumeroEstacao)

ESCREVAL( "Português : ", NomeEstacao[NumeroEstacao, 1] )
ESCREVAL( "Inglês : ", NomeEstacao[NumeroEstacao, 2] )

FIMALGORITMO
```

Anotações

Exercícios

1) Escreva um algoritmo que leia os nomes de 15 funcionários de uma fábrica e os seus respectivos salários e os armazene em dois vetores diferentes. Em seguida, exiba uma lista com o nome e o salário de cada um.
2) Escreva um algoritmo que leia os elementos (números inteiros) de uma matriz de 5 linhas e 5 colunas e os exiba.
3) Multiplique os elementos da matriz que você construiu no exercício anterior por 10 e os exiba.
4) Escreva um algoritmo que armazene e mostre os nomes dos 11 jogadores titulares de 5 times de futebol.
5) Escreva um algoritmo que armazene valores inteiros em uma matriz 3 x 4 e calcule e mostre a média aritmética dos valores digitados.

Anotações

Capítulo 11 - Depurando Códigos

Objetivos

Neste capítulo você irá aprender:

- Depuração de Códigos
- Erros de Compilação
- Erros em Tempo de Execução
- Depurar Códigos no VisuAlg

Exercícios

Os exemplos e exercícios utilizados neste capítulo estão na pasta: **Capítulo 11 - Depurando Códigos.**

Duração

Este capítulo levará aproximadamente 45 minutos

Anotações

Depuração de Códigos

Definição

Existem infinitas possibilidades de erro dentro de nosso código. Para encontrar e corrigir esses erros antes de seu produto ser vendido, foi criada a técnica de depuração.

Depuração de códigos ou **debug** é uma técnica de prevenção de problemas genéricos de aplicações. O ato de depurar um determinado bloco de código tem como objetivo localizar problemas que poderiam causar saídas inesperadas em nosso programa. Trata-se de uma tarefa as vezes muito trabalhosa, porém que evitará um desperdício de tempo e retrabalho no futuro.

Neste capítulo iremos entender os dois principais causadores de erro em programas simples, os Erros de Compilação e os Erros em Tempo de Execução.

Erros de Compilação

Definição

O Compilador de uma linguagem de programação é o responsável em traduzir o código escrito em uma determinada linguagem para um "código de máquina", tornando possível para o computador realizar a determinada instrução desejada.

Erros de compilação ocorrem quando existe algo errado com o código escrito no programa, fazendo o compilador não conseguir traduzir seu programa para uma linguagem que o computador entenderá.

Podem ocorrer por diversos motivos, sendo dois deles mais comuns, como descrito abaixo:

1. Erro na sintaxe do código: Ocorre quando deixamos de escrever algum caractere que não pode deixar de existir, ou quando escrevemos algo na sequencia errada.

   Exemplo

```
//Maneira correta:
ESCREVA("Estação [1-4] : ")
//Erro de compilação:
ESCRVA(“Estação [1-4] : ”)

//Maneira correta:
ESCREVA("Estação [1-4] : ")
//Erro de compilação:
ESCREVA(Estação [1-4] : ”)
```

	Anotações

2. Erro de referência (ou assembly): Ocorre quando esquecemos de referenciar explicitamente no nosso código de onde vem o determinado componente / estrutura que estamos utilizando. Até o momento, não foi utilizado nenhum tipo de componente / estrutura com o VisuAlg, porém para os cursos sequenciais será necessário o entendimento.

Erros em Tempo de Execução

Definição

Todo programa contém a lógica do que deve ser executado em suas linhas de código. O computador somente traduz essas linhas de código e os executa. Uma vez que o código foi feito para realizar uma tarefa específica, não se pode esperar que ele realize outra tarefa.

Quando desenvolvemos um código para que faça uma validação da idade de uma pessoa, se esquecermos de validar uma opção, possivelmente nosso código não fará o que gostaríamos que ele fizesse, o que é totalmente compreensível, pois um computador não tem a capacidade de realizar nada além do que foi definido pelo nosso código.

Exemplo

```
ALGORITMO "Cursos na Yto Nihon"

// Seção de Declarações
VAR
Nota : INTEIRO

INICIO
ESCREVAL ("Dê uma nota de 0 a 5 para o curso de Lógica de
Programação.")
LEIA (Nota)

ESCOLHA Nota
CASO 0
   ESCREVA ("Péssimo.")
CASO 1
   ESCREVA ("Ruim.")
CASO 2
   ESCREVA ("Regular.")
CASO 3
   ESCREVA ("Bom.")
```

	Anotações

```
CASO 5
   ESCREVA ("Exelente")
FIMESCOLHA

FIMALGORITMO
```

Nesse caso, existe uma lógica para saber o valor descritivo dada uma nota de 0 a 5. Caso nós rodássemos esse código, ele não indicaria nenhum problema de compilação, pois não está nada escrito errado e não temos referências externas com problemas. Porém, caso reparemos bem, se o aluno digitar a nota 4, que está entre 0 e 5, não teremos nenhuma resposta, o que é um erro do programa.

Depurar códigos no VisuAlg

Conceito

Depurar códigos em qualquer tipo de programa que eventualmente você venha a desenvolver é algo necessário e imprescindível para a qualidade do seu produto. Pode ser entendido como a primeira fase dos testes de um programa, onde o próprio programador será o responsável por ver se o que deveria acontecer está realmente acontecendo.

Entre as diversas possibilidades de testar seu programa, existem as mais simples, como:

a. Rodar seu programa e ver se o que acontece é o esperado.
b. Realizar “Testes de mesa” enquanto formula a lógica de seu programa, antes mesmo de escrever qualquer linha de código.
c. Inserir breakpoints em pontos chave do seu programa e verificar se até aquele momento tudo está como deveria estar.

a. Programação da Emoção

A opção “a” não pode ser nem considerada uma possibilidade. Considerando que um programa vem para resolver um problema, caso não tenhamos certeza do que está por vir, poderemos causar outros. Devemos tomar muito cuidado com essa possibilidade. Ela existe, porém não devemos usufruir dela. Gaste mais tempo com o produto e faça dele melhor.

b. Teste de Mesa

Já na opção “b” temos um conceito que é extremamente necessário e útil ao programador. O “Teste de mesa” é um teste do qual desenvolvemos a nossa lógica e simulamos o que irá acontecer sem testar o programa diretamente, possível de se fazer apenas com papel e caneta.

Essa técnica consiste em separarmos as variáveis que utilizaremos em nosso programa em uma tabela e acompanharmos linha a linha do programa qual será o valor de cada uma das variáveis.

Exemplo de Teste de Mesa

	Anotações

Devemos desenvolver um programa que leia dois números (A e B) e inverta seus valores. Após a inversão exibir seus valores.

Supondo que desenvolvamos o programa da seguinte maneira:

```
ALGORITMO "Programa 1 - Teste de Mesa"

// Seção de Declarações
VAR
A : INTEIRO
B : INTEIRO

INICIO
ESCREVA ("Digite o valor de A:")
LEIA (A)
ESCREVA ("Digite o valor de B.")
LEIA (B)

A <- B
B <- A

ESCREVAL ("Valor de A.", A)
ESCREVAL ("Valor de B.", B)
FIMALGORITMO
```

Utilizando o Teste de Mesa para esse algoritmo faríamos uma tabela da seguinte maneira:

Linha do Código	Valor de A	Valor de B
11	10	5
12	5	5
13	5	5
14	5	5
15	5	5
16	5	5

Como podemos ver, o resultado do programa **não** é o solicitado pelo enunciado, onde passados os valores “10” para a variável “A” e o valor “5” para a variável “B” deveríamos obter ao final de nosso algoritmo o valor de “10”

	Anotações

para “B” e o valor de “5” para “A”. Dessa maneira comprovamos que nossa lógica não está correta sem ao menos usar o computador.

Conseguimos resolver nosso problema proposto utilizando uma **variável auxiliar** conforme o exemplo:

```
ALGORITMO "Programa 2 - Teste de Mesa"

// Seção de Declarações
VAR
  A : INTEIRO
  B : INTEIRO
  AUX : INTEIRO

INICIO
ESCREVA ("Digite o valor de A:")
LEIA (A)
ESCREVA ("Digite o valor de B:")
LEIA (B)

AUX <- A
A <- B
B <- AUX

ESCREVAL ("Valor de A:", A)
ESCREVAL ("Valor de B:", B)
FIMALGORITMO
```

Utilizando o Teste de Mesa para esse algoritmo faríamos uma tabela da seguinte maneira:

Linha do Código	Valor de A	Valor de B	Valor de AUX
14	10	5	-
15	10	5	10
16	5	5	10
17	5	10	10
18	5	10	10
19	5	10	10

Anotações

Dessa maneira conseguimos verificar que o código escrito tem a lógica que o exercício solicitou, o valor passado em “A” agora está em “B” e o de “B” está em “A”.

c. Breakpoints

Nas linguagens de programação mais modernas é possível inserir pontos de parada em tempo de execução. Isso significa que podemos descobrir como está o programa no pondo que desejarmos, sendo assim possível detectar falhas de lógica em pontos específicos de nosso programa sem precisar passar por todos os pontos anteriores.

Supondo que temos um programa muito extenso, do qual não seja necessário visualizar ponto a ponto de nosso código para saber se uma determinada parte que estamos desenvolvendo agora está certa. Para esse caso o Breakpoint é fundamental. Rodamos nosso código e pedimos para que ele pare somente onde estamos precisando testar. Todo o restante do código presumimos que está funcionando como deveria e não nos preocupamos.

Exemplo de depuração com comando PAUSA do VisuAlg

Para exemplificar o uso de pontos de parada com o VisuAlg utilizaremos um algoritmo que já utilizamos anteriormente no capítulo de “Laços de Validação” porém adicionando o comando “PAUSA”.

```
ALGORITMO "Se Então Senão"

VAR
Nota1 : REAL
Nota2 : REAL
Media : REAL

INICIO

ESCREVA("Digite o valor da primeira nota : ")
LEIA (Nota1)

ESCREVA("Digite o valor da segunda nota : ")
LEIA (Nota2)

Media := (Nota1 + Nota2) / 2

PAUSA
```

	Anotações

```
ESCREVAL ("A média é =", Media)

SE Media >= 7 ENTAO
   // Instrução com condição verdadeira
   ESCREVAL ("Aluno aprovado!")
SENAO
   // Instrução com condição falsa
   ESCREVAL ("Aluno não aprovado!")
FIMSE

FIMALGORITMO
```

Dessa maneira pudemos visualizar que nosso código funcionou exatamente como deveria, porém, antes de exibir a média calculada, o código foi pausado. Com essa pausa, pudemos visualizar os valores das variáveis antes mesmo de ser exibido na tela.

O comando para continuarmos o código linha por linha é o **F8**. Caso quisermos continuar o código até o final sem parar em local algum, o comando é **F9**.

Conseguimos obter o mesmo resultado do comando **PAUSA** clicando ao lado esquerdo da linha de código que gostaríamos que pausasse nossa execução. Repare que a linha inteira ficará com uma cor avermelhada e um ponto circular ficará onde clicou, o que não é nada mais que uma indicação de onde seu código irá parar, ou seja, um **breakpoint**. Para retirar o **breakpoint** clique em cima do mesmo.

Lembrando que somente pausará o programa se a linha escolhida conter algo escrito, do contrário não pausará. Se precisar parar o código antes ou depois de alguma linha específica escreva o comando **PAUSA**.

Outra possibilidade de depuração com o VisuAlg é o comando **DEBUG** que consiste em somente pausar o código caso uma expressão lógica for verdadeira.

Exemplo de breakpoint com comando DEBUG do VisuAlg

Utilizaremos o mesmo algoritmo do exercício anterior, adicionando o comando uma expressão lógica no local onde teríamos o comando PAUSA e o comando DEBUG para decidir se pausamos ou não dado o resultado da expressão lógica.

```
ALGORITMO "Se Então Senão"
VAR
Nota1 : REAL
```

	Anotações

```
Nota2 : REAL
Media : REAL
logic : logico

INICIO

ESCREVA("Digite o valor da primeira nota : ")
LEIA (Nota1)
ESCREVA("Digite o valor da segunda nota : ")
LEIA (Nota2)

Media := (Nota1 + Nota2) / 2

SE Nota1 >= 5  ENTAO
   logic <- verdadeiro
SENAO
     logic <- falso
FIMSE

DEBUG logic

ESCREVAL ("A média é =", Media)

SE Media >= 7 ENTAO
   // Instrução com condição verdadeira
   ESCREVAL ("Aluno aprovado!")
SENAO
   // Instrução com condição falsa
   ESCREVAL ("Aluno não aprovado!")
FIMSE

FIMALGORITMO
```

Conforme visualizamos ao rodar o programa, somente foi pausado para possível avaliação quando obtivemos um valor VERDADEIRO para a expressão lógica. Posteriormente funcionou igualmente ao comando **PAUSA**.

	Anotações

Exercícios

1) Quais são as principais vantagens de aplicar técnicas de depuração de códigos?

2) Quando desenvolvemos uma aplicação, existem ao menos três maneiras de verificar se nosso código realizará ou não o esperado. Cite quais são e classifique o tipo de erro qual se resolve utilizando essa técnica (Execução, Compilação ou Não se Aplica).

3) Quais os comandos que adicionando ao código do VisuAlg auxiliam a depuração de código? Qual a diferença entre eles?

4) A quem cabe a responsabilidade de depurar o código existente? Assinale a alternativa correta.
   a) Analista de Testes
   b) Cliente
   c) Programador
   d) Gerente do Produto
   e) Analista de Sistemas

Anotações

Capítulo 12 - Subalgoritmos

Objetivos

Neste capítulo você irá aprender:

- Subalgoritmo
- Semelhanças e Diferenças
- Funções
- Procedimentos

Exercícios

Os exemplos e exercícios utilizados neste capítulo estão na pasta: **Capítulo 12 - Subalgoritmos.**

Duração

Este capítulo levará aproximadamente 120 minutos

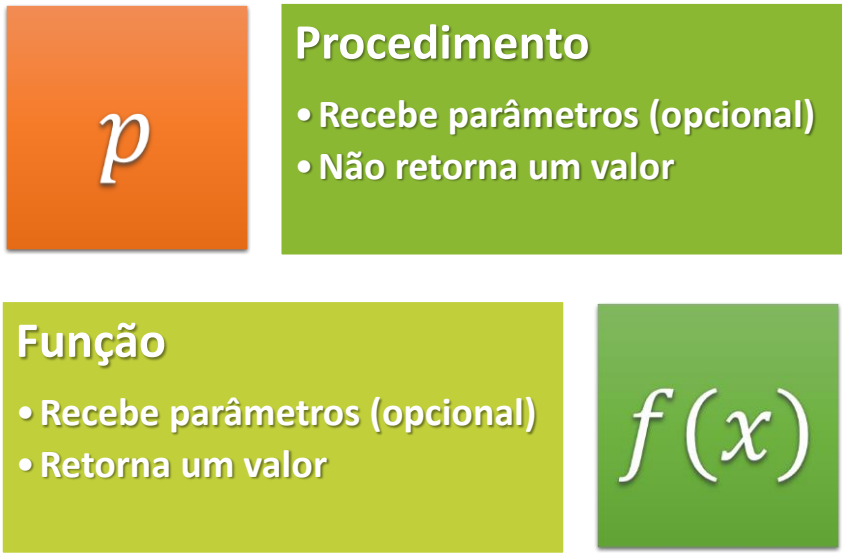
	Anotações

Subalgoritmo

sub.al.go.rit.mo
ant (ár al-Huwârizmî)

1 ***Informática*****:** Porção de código que resolve um problema muito específico, parte de um problema maior (a aplicação final).

Semelhanças e Diferenças



Funções

Criando Funções

```
ALGORITMO "<nome do algoritmo>"
VAR
<Declaração de variáveis globais>
<Definição da função>
INICIO
<Lista de comandos>
FIMALGORITMO
```

Sintaxe da Função

```
FUNCAO <identificador> ([var]<parâmetros>) <tipo de retorno>
VAR
<Declaração de variáveis locais>
INICIO
```

Anotações

```
<Lista de comandos>
RETORNE <variável de retorno>
FIMFUNCAO
```

A criação de uma Função deve ser declarada, com os demais objetos, no início do programa. Este tipo de subalgoritmo sempre retorna um e apenas um valor ao algoritmo que lhe chamou. Cada função tem associada ao seu valor de retorno um tipo explícito. Da mesma maneira com que os parâmetros são fixos para toda a chamada o retorno também é fixo.

Cuidados

- Sempre declare as variáveis globais antes da função.
- A função sempre fica dentro do escopo Algoritmo e Fim Algoritmo.
- Procure não declarar variáveis globais com o mesmo nome das variáveis da função.

Exemplo

```
ALGORITMO "Funções"

VAR
// Declaração das variáveis globais
Bim1, Bim2, Bim3, Bim4, MediaS, MediaP : REAL

// Declaração das funções
FUNCAO MediaSimples(Nota1, Nota2, Nota3, Nota4 : REAL) : REAL

// Declaração de variáveis locais
VAR
Resultado : REAL

INICIO

// Instruções
Resultado := (Nota1 + Nota2 + Nota3 + Nota4) / 4)

// Valor de retorno
RETORNE Resultado

FIMFUNCAO
```

	Anotações

```
FUNCAO MediaPonderada(Termo1, Termo2, Termo3, Termo4 : REAL) : REAL

VAR
// Declaração de variáveis locais
Resultado : REAL

INICIO

// Instruções
Resultado := (Termo1 * 1 + Termo2 * 2 + Termo3 * 3 + Termo4 * 4) /
10

// Valor de retorno
RETORNE Resultado

FIMFUNCAO

INICIO
ESCREVA ("1° Bimestre : ")
LEIA (Bim1)

ESCREVA ("2° Bimestre : ")
LEIA (Bim2)

ESCREVA ("3° Bimestre : ")
LEIA (Bim3)

ESCREVA ("4° Bimestre : ")
LEIA (Bim4)

// Chamada da função
MEDIAS := MediaSimples(Bim1, Bim2, Bim3, Bim4)
MEDIAP := MediaPonderada(Bim1, Bim2, Bim3, Bim4)

ESCREVAL
```

	Anotações

```
ESCREVAL ("Média Simpes : ", MediaS)
ESCREVAL ("Média Ponderada : ", MediaP)

FIMALGORITMO
```

Procedimentos

Criando Procedimentos

```
ALGORITMO "<nome do algoritmo>"
VAR
<Declaração de variáveis globais>
<Definição do procedimento>
INICIO
<Lista de comandos>
FIMALGORITMO
```

Sintaxe do Procedimento

```
PROCEDIMENTO <identificador> ([VAR]<parâmetros>)
VAR
<Declaração de variáveis locais>
INICIO
<Lista de comandos>
FIMPROCEDIMENTO
```

Cuidados

- Sempre declare as variáveis globais antes da função.
- A função sempre fica dentro do escopo Algoritmo e Fim Algoritmo.
- Procure não declarar variáveis globais com o mesmo nome das variáveis da função.

Exemplo

```
ALGORITMO "Procedimentos"

VAR
// Declaração das variáveis globais
Bim1, Bim2, Bim3, Bim4 : REAL

// Declaração dos procedimentos
```

	Anotações

```
PROCEDIMENTO MediaSimples (Nota1, Nota2, Nota3, Nota4 : REAL)

// Declaração de variáveis locais
VAR
Resultado : REAL

INICIO

// Instruções
Resultado := (Nota1 + Nota2 + Nota3 + Nota4) / 4)

ESCREVAL ("Média Simpes : ", Resultado)

FIMPROCEDIMENTO

PROCEDIMENTO MediaPonderada(Termo1, Termo2, Termo3, Termo4 : REAL)

VAR
// Declaração de variáveis locais
Resultado : REAL

INICIO

// Instruções
Resultado := (Termo1 * 1 + Termo2 * 2 + Termo3 * 3 + Termo4 * 4) /
10

ESCREVAL ("Média Ponderada : ", Resultado)

FIMPROCEDIMENTO

INICIO
ESCREVA ("1° Bimestre : ")
LEIA (Bim1)

ESCREVA ("2° Bimestre : ")
```

	Anotações

```
LEIA (Bim2)

ESCREVA ("3° Bimestre : ")
LEIA (Bim3)

ESCREVA ("4° Bimestre : ")
LEIA (Bim4)

// Chamada dos procedimentos
MediaSimples(Bim1, Bim2, Bim3, Bim4)
MediaPonderada(Bim1, Bim2, Bim3, Bim4)

FIMALGORITMO
```

	Anotações

Exercícios

1) Escreva um algoritmo e crie, nele, uma função que receba um número e retorne a soma dos números inteiros positivos e menores que o número digitado. Se o número digitado for negativo ou nulo, não chama a função e mostra uma mensagem de erro.
2) Escreva um algoritmo que leia duas notas de um aluno e chame uma função que calcula a média aritmética desse aluno.
3) Escreva um algoritmo que receba um número inteiro e chame um procedimento que calcule o quadrado desse número. Crie esse procedimento.
4) Escreva um algoritmo que receba um número inteiro e chame um procedimento que receba um número e mostre a raiz quadrada do mesmo.
5) Faça um algoritmo que leia três números e crie uma função que calcule o quadrado desses números e retorne o valor.

Anotações

Capítulo 13 - Introdução ao Banco de Dados

Objetivos

Neste capítulo você irá aprender:

- Banco de Dados
- Tabela
- Registro
- Atributo
- Chave
- Relacionamento

Exercícios

Os exemplos e exercícios utilizados neste capítulo estão na pasta: **Capítulo 13 - Introdução ao Banco de Dados.**

Duração

Este capítulo levará aproximadamente 30 minutos

	Anotações

Banco de Dados

ban.co
sm (germ bank)

da.dos
sm pl (lat data)

1 ***Banco de dados, Informática***: Coleção organizada de dados, armazenados num meio físico, para tratamento posterior.

Tabela

ta.be.la
sf (lat tabella)

1 Quadro ou agrupamento coerente de dados.

2 Informática: uma simples estrutura de linhas e colunas em uma tabela, cada linha contém um mesmo conjunto de colunas.

Registro

re.gis.tro
sm (baixo-lat registru)

1 Ficha individual para determinada finalidade.

2 Cada linha formada por uma lista ordenada de colunas representa um registro. Os registros não precisam conter informações em todas as colunas, podendo assumir valores nulos quando assim se fizer necessário.

Atributo

a.tri.bu.to
sm (lat attributu)

1. Aquilo que é próprio ou peculiar de alguém ou de alguma coisa.
2. Condição, propriedade, qualidade.
3. ***Informática*:** Entrada de um campo num arquivo.
4. ***Informática*:** Informação referente à exibição ou apresentação da informação.

Chave

cha.ve
sf (lat clave)

Anotações

1 Instrumento, comumente de metal, apropriado para manobrar a lingueta da fechadura ou cadeado a que pertence.

2 ***Informática*****:** Conjunto de um ou mais atributos que determinam a unicidade de cada registro.

Chave

cha.ve
sf (lat clave)

1 Instrumento, comumente de metal, apropriado para manobrar a lingueta da fechadura ou cadeado a que pertence.

2 ***Informática*****:** Conjunto de um ou mais atributos que determinam a unicidade de cada registro.

Tipos de Chave

Chave Primária

Chave Estrangeira

Chave primária - *Primary Key*

É a chave que identifica cada registro dando-lhe unicidade. A chave primária nunca se repetirá.

Chave Estrangeira - *Foreign Key*

É a chave formada através de um relacionamento com a chave primária de outra tabela. Define um relacionamento entre as tabelas e pode ocorrer repetidas vezes. Caso a chave primária seja composta na origem, a chave estrangeira também o será.

Anotações

Relacionamento

re.la.ci.o.na.men.to
sm (relacionar+mento)

1 Ação ou efeito de relacionar; relacionação.

2 ***Informática*: um** relacionamento do Modelo de Entidades e Relacionamentos é uma associação entre entidades distintas.

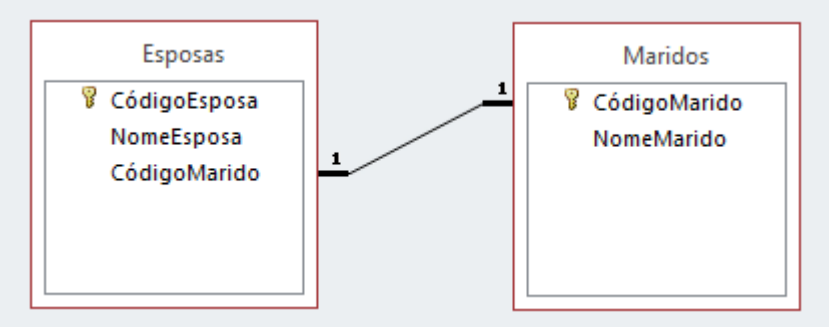
Tipos de Relacionamento

Um para um (1 para 1)

Um para muitos (1 para ∞)

Muitos para muitos (∞ para ∞)

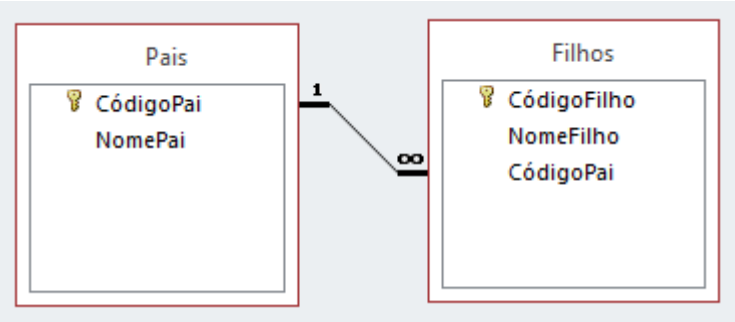
Um para um (1 para 1)



Indica que as tabelas têm relação unívoca entre si. Você escolhe qual tabela vai receber a chave estrangeira;

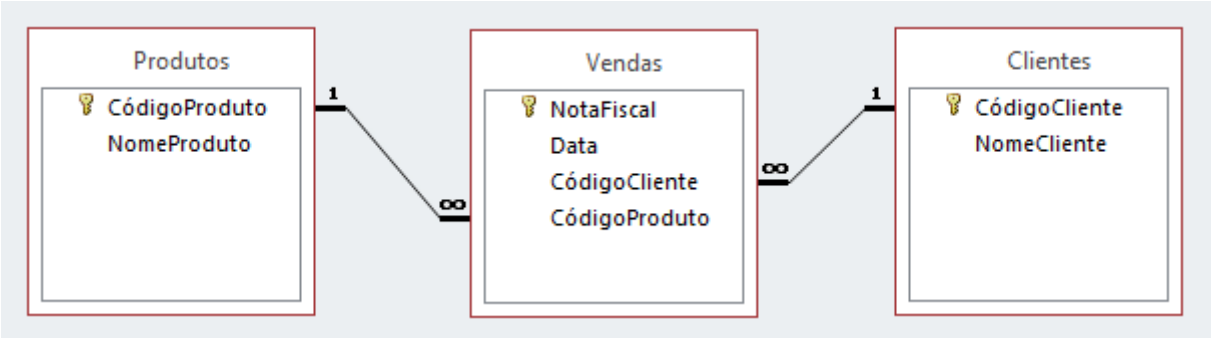
Anotações

- **Um para muitos (1 para ∞)**



A chave primária da tabela que tem o lado 1 está para ir para a tabela do lado ∞. No lado ∞ ela é chamada de chave estrangeira;

Muitos para muitos (∞ para ∞)



Quando tabelas têm entre si relação **∞... ∞**, é necessário criar uma nova tabela com as chaves primárias das tabelas envolvidas, ficando assim uma chave composta, ou seja, formada por diversos campos-chave de outras tabelas. A relação então se reduz para uma relação 1 ... ∞, sendo que o lado n ficará com a nova tabela criada.

	Anotações

Exercícios

1) Palavras Cruzadas

Horizontal

3. Uma simples estrutura de linhas e colunas. Em uma tabela, cada linha contém um mesmo conjunto de colunas.

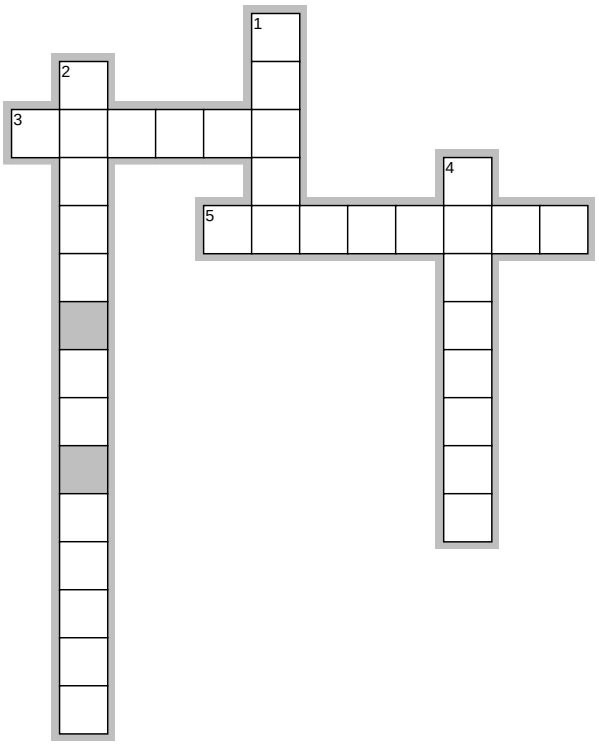
5. Cada linha formada por uma lista ordenada de colunas representa um registro. Os registros não precisam conter informações em todas as colunas, podendo assumir valores nulos quando assim se fizer necessário.

Vertical

1. Conjunto de um ou mais atributos que determinam a unicidade de cada registro.

2. Coleção organizada de dados, armazenados num meio físico, para tratamento posterior

4. Informação referente à exibição ou apresentação da informação



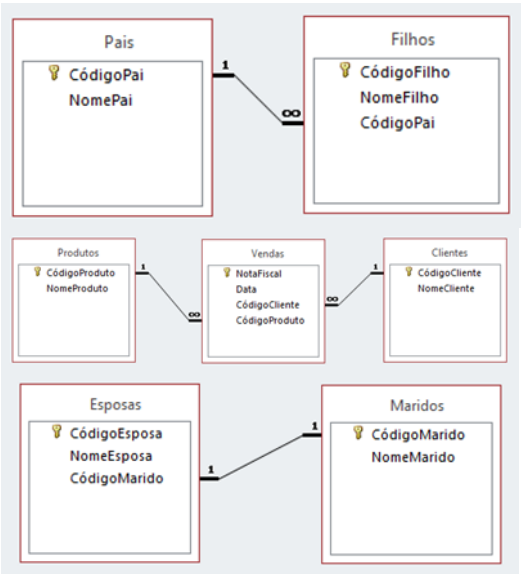
Anotações

2) Associe:

Um para um (1 para 1)

Um para muitos (1 para ∞)

Muitos para muitos (∞ para ∞)



Anotações

Capítulo 14 - Introdução à Orientação a Objetos

Objetivos

Neste capítulo você irá aprender:

- Programação Orientada-a-Objetos
- Objetos
- Classes
- Mensageria
- Herança
- Porque Programar Orientado-a-Objetos
- Linguagens de Programação

Exercícios

Os exemplos e exercícios utilizados neste capítulo estão na pasta: **Capítulo 14 - Introdução à Orientação a Objetos.**

Duração

Este capítulo levará aproximadamente 45 minutos

	Anotações

Programação Orientada-a-Objetos

A programação Orientada-a-Objetos foi criada para tentar aproximar o mundo real do mundo computacional. Sendo assim, programar Orientado-a-Objetos é tentar simular o mundo real dentro do computador. Para isso, esse paradigma utiliza de Objetos para troca de mensagens, cabendo ao programador dizer qual será a estrutura de cada objeto, o que cada um deles pode fazer e como será a comunicação entre esses objetos.

Objetos

Definição

Qualquer coisa no nosso mundo é capaz de ser traduzida em Objeto. Isso significa que uma pessoa, um cachorro, um carro, um barco ou qualquer outra coisa do nosso mundo pode ser entendida como um objeto.

Objetos no mundo real se assemelham de duas maneiras, ambos têm "**propriedades**" e "**comportamentos**". Suas propriedades definem **quem / o que** esse objeto é e seus comportamentos dizem o que ele é capaz de fazer - suas **ações**.

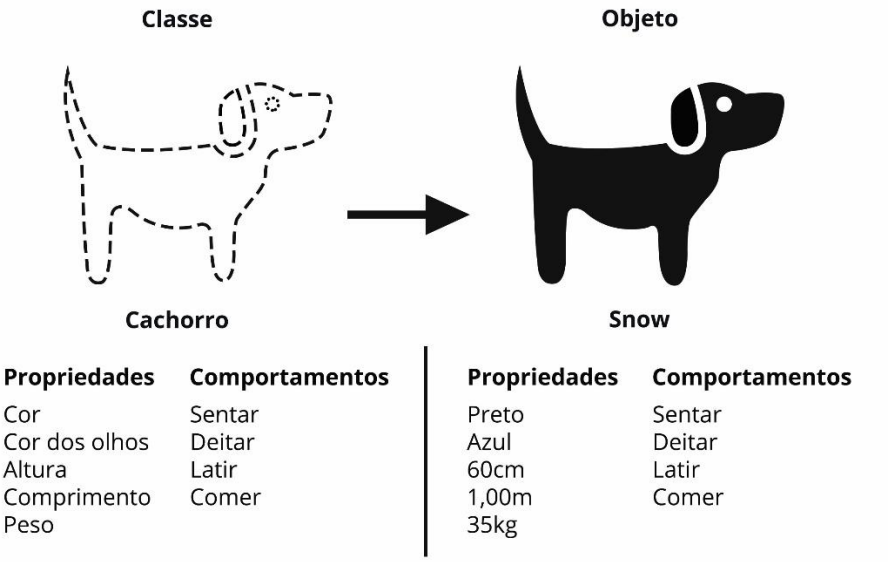
Pensando em alguns objetos de nosso cotidiano, percebemos quão complexos eles podem se tornar. Pense em quais estados físicos - propriedades, e quais comportamentos ele pode ter. Perceberá que em alguns casos, objetos tem propriedades que podem ser outros objetos.

Quando buscamos um objeto em específico, ou seja, um objeto com propriedades e comportamentos já estabelecidos formam o que chamamos de uma **Instancia de Uma Classe**.

Exemplo

Um cachorro tem: nome, cor da pelagem, cor dos olhos e raça. Ao mesmo tempo ele tem a capacidade de: latir, andar, deitar, rolar e etc.

	Anotações



Snow é um cachorro, ela tem: nome, cor da pelagem, cor dos olhos e raça. Ou seja, ela é um Objeto cachorro, porém um específico objeto cachorro, o Snow. Chamamos Snow de uma **Instância de uma Classe**, pois ele tem tudo que um cachorro tem, porém ele é único, nenhum outro cachorro é igual a ele.

Classes

Definição

Classes são a definição da estrutura de propriedades e comportamentos que definem um determinado tipo de Objeto. Quais são as propriedades que este objeto terá e quais suas ações possíveis.

A Classe é somente a estrutura, ela não armazena informações de objetos. Serve para que seja possível distinguir cada uma das instancias de cada objeto, armazenando as informações passadas em sua criação.

Anotações

Exemplo

Clase Pessoa	Objeto Augusto (Instância da Classe Pessoa)

• Propriedades
  • Nome
  • Endereço
  • Telefone
• Comportamentos
  • Beber
  • Comer
  • Andar
	• Propriedades
	  • Augusto
	  • rua Dr. Ferreira Lopes, 317
	  • 99292-9292
	• Comportamentos
	  • Beber
	  • Comer
	  • Andar

Augusto é uma pessoa, contem suas propriedades e se comporta conforme uma pessoa, porém diferente de outras devido a suas propriedades.

Mensageria

Definição

Tentando aproximar a programação ao mundo real, devemos pensar em como as coisas se relacionam no mundo real e como faremos isso programaticamente.

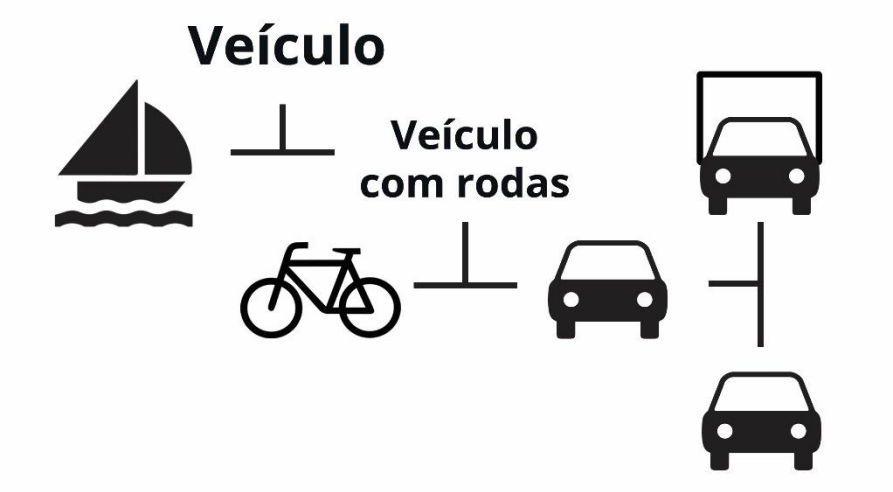
Dessa maneira foi desenvolvido para programação orientada a objetos o conceito de **Mensagem** onde cada objeto se relaciona com os outros objetos através de mensagens.

Exemplo

Caso um objeto Professor envie uma mensagem para um objeto do tipo Aluno dizendo “Copie a matéria”, o objeto Aluno irá interpretar essa mensagem e consequentemente irá executar a instrução dada dentro de seu comportamento de nome CopiarMatéria.

	Anotações

Herança



Definição

Herança é o processo de formação de uma classe baseando-se em uma classe já existente, adicionando novas propriedades e comportamentos.

A Classe já existente chamamos de Super Classe ou Classe Base e a nova classe chamamos de Classe Derivada uma vez que ela é uma derivação / especialização da classe já existente, podendo chamar de Classe Filha ou Sub Classe também.

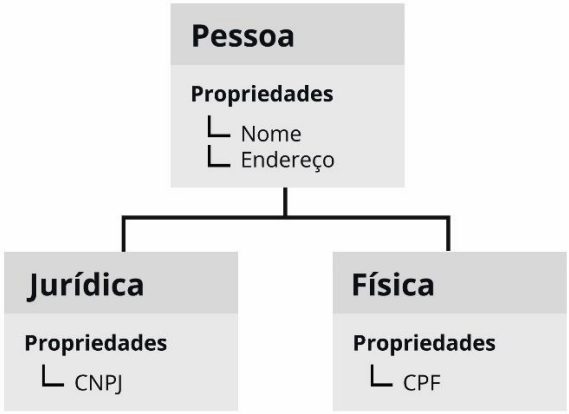
Com a Herança podemos reutilizar um código existente e tornar a aplicação algo mais próximo da realidade, facilitando o entendimento de outros programadores quando forem dar manutenção em seu código ou desenvolver novas funcionalidades.

Exemplo

Supondo que estejamos desenvolvendo um software para vendas, sendo possível vender para pessoas físicas e pessoas jurídicas.

Ambos tipos, pessoa física e pessoa jurídica contém algumas propriedades em comum, que dizem respeito a uma “Pessoa”. Utilizando o conceito de Herança, conseguimos criar uma Classe Base da qual será possível derivar duas classes, a de Pessoa Física e a de Pessoa Jurídica, conforme a imagem abaixo:

Anotações



Porque programar Orientado-a-Objetos

A programação Orientada-a-Objetos vem se tornando cada dia mais famosa e utilizada devido ao nível de proximidade com a realidade.

A programação anteriormente a POO era algo complexo de se aprender, com uma curva de aprendizado muito lenta e desfavorável, causando um desconforto muito grande para qualquer interessado.

Com a POO o relacionamento entre a área de negócio e a área de desenvolvimento de software ficou mais simples e compreensível para ambas as partes. A orientação a objetos permite que exista uma documentação legível tanto pelo desenvolvedor do sistema quanto pelo entendedor do negócio em si.

Normalmente para documentar a POO utiliza-se UML (Unified Modeling Language), que é uma ferramenta/linguagem para permitir esse entendimento entre negócio e programação. Algumas tecnologias como o Microsoft Visual Studio, fornecem uma estrutura para se documentar o código que deverá existir em UML e já prepara o próprio para o programador em C#.NET ou VB.NET.

Linguagens de Programação

Dentre as linguagens criadas com o padrão de orientação-a-objetos, as mais utilizadas no Brasil e no mundo são C#.NET e VB.NET (Microsoft), Java (Sun/Oracle) e Objective-C (Apple).

Quando pensamos em C#.NET e VB.NET estamos focados no mundo Microsoft, onde temos desenvolvimento de aplicações para Windows (XP / Vista / 7 / 8), Mobile (Windows Phone) e Web. A linguagem VB.NET visava trazer todos os desenvolvedores de VB6 e VBA para a nova plataforma Microsoft, porém com o crescimento de linguagens com uma sintaxe diferenciada, como o Java, o foco da Microsoft hoje é o C#.NET, que utiliza das mesmas funções e ferramentas disponíveis para VB.NET com o .NET Framework, porém contempla o necessário para desenvolvedores de outras linguagens se ambientarem mais rapidamente.

Java é a principal linguagem OpenSource que utiliza o paradigma de orientação-a-objetos. Nela é possível desenvolvimento para Windows e Linux, Mobile (Android) e Web. Contém a maior das comunidades de

	Anotações

programação, o que permite um esclarecimento de dúvidas extremamente eficiente. Atualmente existem mais de 3 bilhões de aplicações que utilizam Java - segundo a Oracle.

Objective-C é a linguagem utilizada dentro dos Sistemas Operacionais Apple, seja Mobile com Iphone, Ipod, Ipad, ou Desktop com o MAC OS (Mavericks, Lion). Somente a linguagem Swift (nova linguagem Apple) pode ser usada para o desenvolvimento dentro dos ambientes Apple nativamente, porém Objective-C é usada em 99,9% das aplicações Apple e não existe tendência de que isso seja alterado brevemente.

Existem diversas tecnologias para utilizar linguagens de ambientes diferentes, porém quanto mais natividade entre ambiente e linguagem de programação melhor. É possível escrever um programa em C#.NET para funcionar em um ambiente Android ou Apple, porém é mais provável uma perda de performance e limitações de possibilidades, tendo em vista que seu melhor ambiente é a linguagem feita para aquele determinado ambiente.

Exercícios

1) No mundo real e no computacional, objetos se assemelham de duas maneiras. Cite quais são.

2) Como é feito o relacionamento entre objetos no mundo computacional quando desenvolvemos nosso código utilizando uma linguagem orientada-a-objetos?

3) Considerando as Classes Professor, Aluno e Pessoa, ilustre o conceito de Classe Base e Classe Derivada. Desenvolva uma forma que ao menos 3 propriedades e 2 comportamentos sejam compartilhados.

4) Indique as alternativas **verdadeiras** (V) e as **falsas** (F) sobre Programação Orientada-a-Objetos:

	O desenvolvimento orientado a objetos é uma maneira de trazer a realidade para o computador, porém tem uma curva de aprendizado longa e complexa.
	O código desenvolvido por uma linguagem orientada-a-objetos é fácil de compreender, contempla a realidade e aproxima a área de negócios com a de programação.

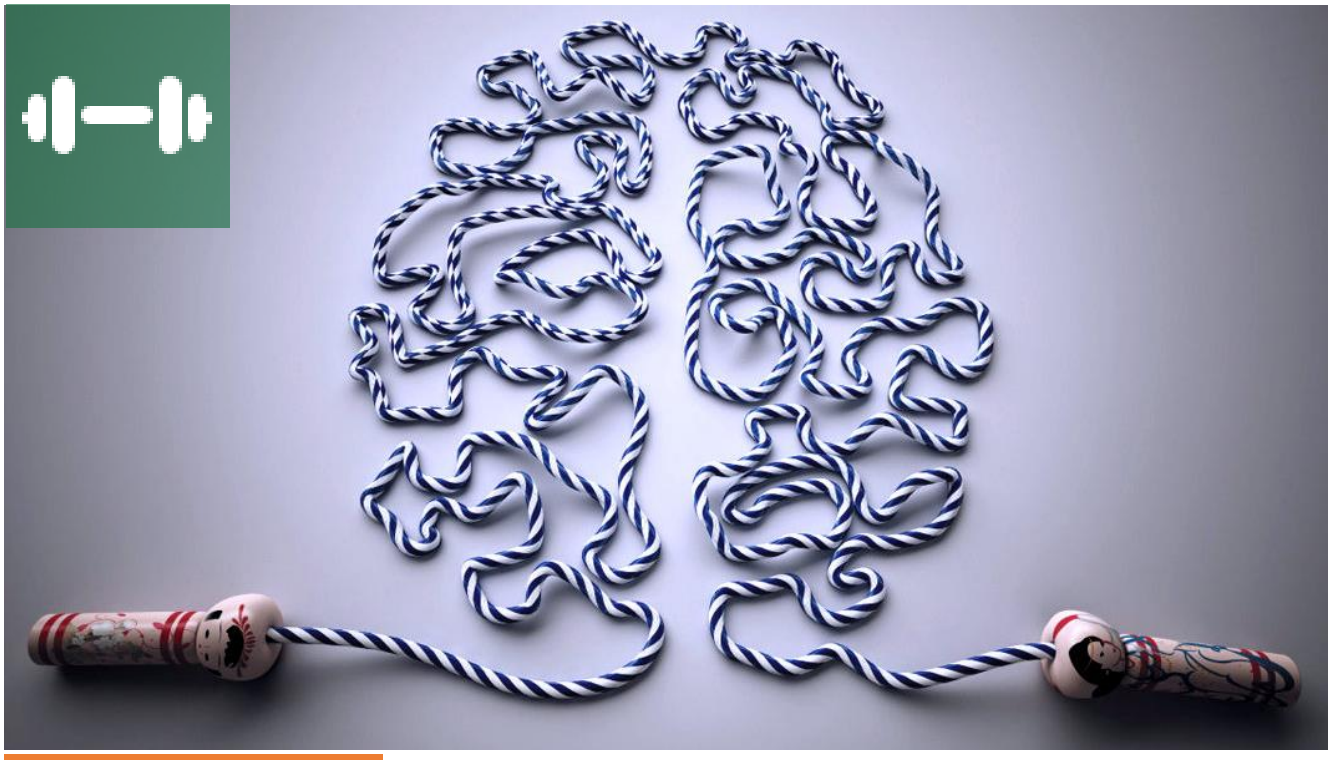
Anotações

	Herança é um conceito dentro de linguagens orientadas-a-objetos da qual uma classe derivada é capaz de especializar uma superclasse.
	Cada objeto é único. Objetos se assemelham em relação a sua estrutura, através de sua Classe.
	O relacionamento entre objetos é entendido como Instancia de uma Classe.

5) Dentre as principais linguagens orientadas-a-objetos, podemos dizer que:
   a) Apple é a linguagem utilizada para desenvolver no ambiente Iphone, Ipod e Ipad.
   b) C#.NET e VB.NET tem como princípio o desenvolvimento para ambientes Windows, porém não compartilham de ferramentas semelhantes.
   c) Java é uma das linguagens mais utilizadas no mundo e contém uma das maiores comunidades de programação.
   d) C#.NET é a linguagem de programação foco da Microsoft. Utiliza de ferramentas compartilhadas com VB.NET e é escrita em uma sintaxe semelhante ao Java.

Anotações

Apêndice - Respostas dos Exercícios



Objetivos

Finalizado o capítulo o aluno conhecerá as respostas dos exercícios:

- Capítulo 01 - Introdução à Lógica de Programação
- Capítulo 02 - Lógica Proposicional
- Capítulo 03 - Trabalhando com o Visual
- Capítulo 04 - Tipos de Dados, Constantes e Variáveis
- Capítulo 05 - Operadores
- Capítulo 06 - Interação Básica
- Capítulo 07 - Funções Predefinidas
- Capítulo 08 - Laços de Validação
- Capítulo 09 - Laços de Repetição
- Capítulo 10 - Variáveis Indexadas
- Capítulo 11 - Depurando Códigos
- Capítulo 12 - Subalgoritmos
- Capítulo 13 - Introdução ao Banco de Dados
- Capítulo 14 - Introdução à Orientação a Objetos

Exercícios

Os exemplos e exercícios utilizados neste capítulo estão na pasta **Instrutor**.

Duração

Este capítulo levará aproximadamente 150 minutos

	Anotações

Capítulo 01 - Introdução à Lógica de Programação

1) Montar em Descrição Narrativa uma sequência lógica para tomar banho. Verificar o que fazer se acabar a água.
   a) Entrar no banheiro e tirar a roupa
   b) Abrir a torneira do chuveiro
   c) Se acabou a água, fechar a torneira e vá para item **i)**
   d) Entrar na água
   e) Ensaboar-se
   f) Sair da água
   g) Fechar a torneira
   h) Enxugar-se
   i) Vestir-se
   j) Fim
2) Montar em Descrição Narrativa uma sequência lógica para trocar um pneu de um carro. Verificar se tem estepe.
   a) Verificar se tem estepe, se não vá para i)
   b) Afrouxar ligeiramente as porcas
   c) Suspender o carro
   d) Retirar as porcas e o pneu
   e) Colocar o pneu reserva
   f) Apertar as porcas
   g) Abaixar o carro
   h) Dar o aperto final nas porcas
   i) Fim
3) Montar em Descrição Narrativa uma sequência lógica que crie um suco de acerola. Caso não tenha açúcar substituir por adoçante.
   a) Aprontar liquidificador
   b) Colocar acerola no liquidificador
   c) Se tiver açúcar, colocar açúcar, se não colocar adoçante
   d) Completar com água
   e) Ligar liquidificador
   f) Aguardar 15 segundos
   g) Desligar liquidificador

Anotações

Capítulo 02 - Lógica Proposicional

1) (TRT 1ª Região/2008/CESPE) Utilizando as letras proposicionais adequadas na proposição composta “Nem Antônio é desembargador nem Jonas é juiz”, assinale a opção correspondente à simbolização correta dessa proposição.
   a) ¬(A ^ B)
   b) (¬A) v (¬B)
   c) (¬A) ^ (¬B)
   d) (¬A) → B
   e) ¬[A v (¬B)]
2) (UFB) se p é uma proposição verdadeira, então:
   a) p ^ q é verdadeira, qualquer que seja q;
   b) p v q é verdadeira, qualquer que seja q;
   c) p ^ q é verdadeira só se q for falsa;
   d) p → q é falsa, qualquer que seja q
   e) n. d. a.
3) (UGF) a negação de x > -2 é:
   a) x > 2
   b) x <> -2
   c) x <= -2
   d) x < 2
   e) x <> 2
4) (ABC) a negação de todos os gatos são pardos é:
   a) Nenhum gato é pardo;
   b) Existe gato pardo;
   c) Existe gato não pardo;
   d) Existe um e um só gato pardo;
   e) Nenhum gato não é pardo.
5) (VUNESP) um jantar reúne 13 pessoas de uma mesma família. das afirmações a seguir, referentes às pessoas reunidas, a única necessariamente verdadeira é:
   a) Pelo menos uma delas tem altura superior a 1,90m;
   b) Pelo menos duas delas são do sexo feminino;
   c) Pelo menos duas delas fazem aniversário no mesmo mês;
   d) Pelo menos uma delas nasceu num dia par;
   e) Pelo menos uma delas nasceu em janeiro ou fevereiro.

Anotações

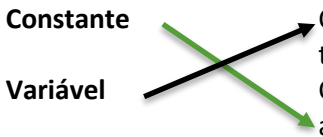
Capítulo 03 - Trabalhando com o VisualAlg

1) Qual a principal função do “Esqueleto” gerado quando abrimos o VisuAlg em nosso editor de texto?
   Além de poupar trabalho ao usuário, mostrar o formato básico de algoritmo que deve ser utilizado, bem como a forma dos comentários.
2) Quais as possibilidades de apresentação de saída com o VisuAlg?
   Podemos utilizar o Simulador de Saídas e o modo DOS. Ambos nos permitem visualizar as saídas de nosso programa.
3) Quando e onde podemos visualizar os valores atribuídos às nossas variáveis?
   Através do Visualizador de Variáveis. Somente sendo possível quando nosso código já estiver em execução.

Anotações

Capítulo 04 - Tipos de Dados, Constantes e Variáveis

1) Associe:



Corresponde a uma posição de memória cujo valor pode variar ao longo do tempo durante a execução do programa.

Corresponde a uma posição de memória cujo valor é constante (não pode variar) ao longo do tempo durante a execução do programa.

2) Preencha as lacunas com **I** para **Inteiro**, **R** para **Real**, **C** para **Caractere** e **L** para **Lógico**.

(C)	“São Paulo”
(L)	FALSO
(R)	1,982
(C)	“007”
(R)	10,333
(I)	127
(I)	549
(L)	VERDADEIRO

3) Para os nomes de indicadores abaixo, preencha as lacunas com **V** para **Válido**, e **I** para **Inválido**.

(I)	%Desconto
(V)	_Salario
(I)	1Trim
(I)	ALGORITMO
(V)	Bim1
(I)	Código
(I)	FIMPROCEDIMENTO
(I)	MENSAGEM

Anotações

Capítulo 05 - Operadores

1) Escreva as seguintes as seguintes expressões “linearizadas” na forma convencional:
   a) A = pi * r ^ 2

$$A = \pi r^2$$

   b) a ^ 2 + b ^ 2 = c ^ 2

$$a^2 + b^2 = c^2$$

   c) x’ = (-b + (b ^ 2 - 4 * a * c) ^(1/2)) / 2 * a

$$x' = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

   d) (2/y -x) ^ 2 = (2/y) ^ 2 - 2 * (2/y) * x + x^2

$$\left(\frac{2}{y} - x\right)^2 = \left(\frac{2}{y}\right)^2 - 2\left(\frac{2}{y}\right)x + x^2$$

2) Resolva as expressões lógicas abaixo, determinando o resultado da expressão, sendo ele, **VERDADEIRO** ou **FALSO**.

(V)	((3 + 5) / 2) > 0)
(V)	(3 * 3 > 10) ou (2 + 2 < 10)
(V)	(10 <> 5) ou (1 + 1 = 0)
(F)	(2 + 3 >= 5) e (18 / 3 < 7)
(F)	(1 + 1 = 0) e (0 - 1 > 0)

4) Preencha com o operador adequado:

Não diferente	=
Não igual	<>
Não inferior ou igual	>
A partir de	>=
Mínimo	>=
Não superior ou igual	<
Até	<=
Máximo	<=

5) Preencha com o operador adequado:

^	**NAO**
v	**E**
~	**OU**
→	**XOU**
↔	**EQV**

	Anotações

v	IMP

Capítulo 06 - Interação Básica

1) Escreva um programa que leia dois valores a e B, calcule a soma entre eles e exiba o resultado para o usuário.

```
ALGORITMO "Capítulo 06 Exercício 01"

// Capítulo  06 - Interação Básica
// Exercício 01 - Escreva um programa que leia dois valores a e B,
calcule a soma entre eles e exiba o resultado para o usuário.

VAR

a : REAL
b : REAL
Soma : REAL

INICIO
// Seção de Comandos

ESCREVA ("Digite 1° valor : ")
LEIA (a)

ESCREVA ("Digite 2° valor : ")
LEIA (b)

Soma := a + b
ESCREVA (Soma)

FIMALGORITMO
```

2) Faça um programa que leia um número inteiro informado pelo usuário, calcule o seu quadrado e exiba o resultado para o usuário.

```
ALGORITMO "Capítulo 06 Exercício 02"

// Capítulo  06 - Interação Básica
```

	Anotações

```
// Exercício 02 - Faça um programa que leia um número inteiro
informado pelo usuário, calcule o seu quadrado e exiba o resultado
para o usuário
VAR

Numero : INTEIRO
Quadrado : INTEIRO

INICIO
// Seção de Comandos

ESCREVA ("Número : ")
LEIA (Numero)

Quadrado := Numero * Numero

ESCREVA (Numero, "² = ", Quadrado)

FIMALGORITMO
```

3) Sabendo que a fórmula para calcular a área de uma circunferência é $\boldsymbol{A = \pi r^2}$, elabore um programa que leia o valor do raio da circunferência, calcule a área e exiba o resultado para o usuário.

```
ALGORITMO "Capítulo 06 Exercício 03"

// Capítulo  06 - Interação Básica
// Exercício 03 - Sabendo que a fórmula para calcular a área de uma
circunferência é a = pi * r ^ 2, elabore um programa que leia o
valor do raio da circunferência, calcule a área e exiba o resultado
para o usuário.

VAR

Area : REAL
Raio : REAL

INICIO
// Seção de Comandos
```

	Anotações

```
ESCREVA ("Raio : ")
LEIA (Raio)

Area := (PI * Raio ^ 2)

ESCREVA ("Área de ", Raio, " é", Area)

FIMALGORITMO
```

4) Faça um algoritmo que leia um valor X e um valor N, e calcule o produto de X por 2 elevado a N.

```
ALGORITMO "Capítulo 06 Exercício 04"

// Capítulo  06 - Interação Básica
// Exercício 04 - Faça um algoritmo que leia um valor X e um valor
N, e calcule o produto de X por 2 elevado a N
VAR
x, n, Total : REAL

INICIO
// Seção de Comandos
ESCREVA ("Digite o valor de X : ")
LEIA (x)

ESCREVAL

ESCREVA ("Digite o valor de N : ")
LEIA (n)

Total := (2 * x) ^ n

ESCREVAL ("(", x, " * 2) ^ ", n, " =", Total)

FIMALGORITMO
```

5) Faça um algoritmo que leia a idade de uma pessoa expressa em anos, meses e dias e exiba apenas em dias.

```
ALGORITMO "Capítulo 06 Exercício 05"
```

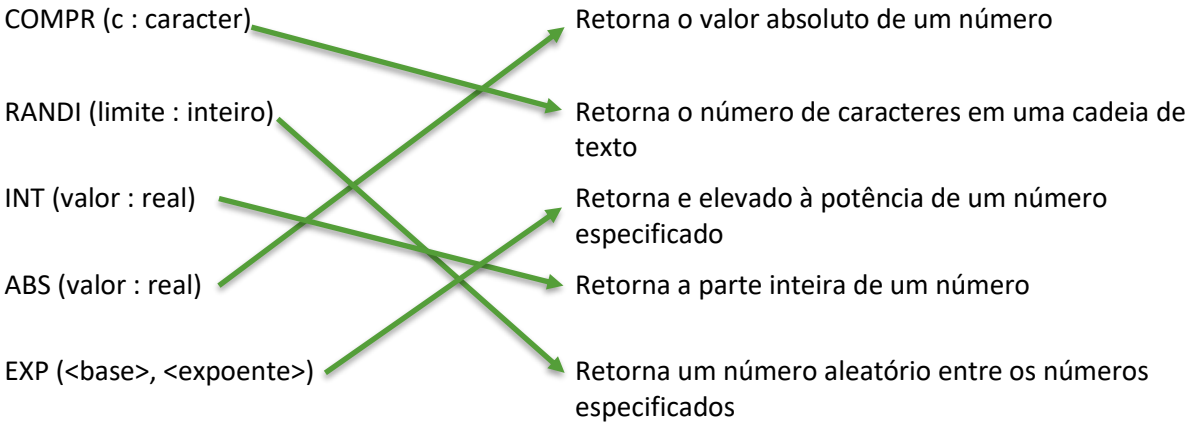
	Anotações

```
// Capítulo  06 - Interação Básica
// Exercício 05 - Faça um algoritmo que leia a idade de uma pessoa
expressa em anos, meses e dias e exiba apenas em dias.

VAR
Dia, Mes, Ano, Total : INTEIRO

INICIO
// Seção de Comandos

LEIA (Ano, Mes, Dia)

Total := (Ano * 365) + (Mes * 30) + Dia

ESCREVAL (Total)

FIMALGORITMO
```

Anotações

Capítulo 07 - Funções Predefinidas

1) Associe:



Anotações

Capítulo 08 - Laços de Validação

1) Elabore um algoritmo que calcule a multa paga por um pescador que ultrapassar a quantidade de quilos estabelecida por lei, a saber:
   - A quantidade de peixe por pessoa é de 50 Kg
   - A multa por quilo excedente é de $ 4,00

```
ALGORITMO "Capítulo 08 Exercício 01"

// Capítulo  08 - Estruturas de Decisão
// Exercício 01 - Elabore um algoritmo que calcule a multa paga por
um pescador que ultrapassar a quantidade de quilos estabelecida por
lei, a saber :
// * a quantidade de peixe por pessoa é de 50 Kg
// * a multa por quilo excedente é de R$ 4,00

VAR
Quilos, Multa : REAL

INICIO

ESCREVA ("Informe o peso : ")
LEIA (quilos)

SE (Quilos > 50) ENTAO

   Multa := (Quilos - 50) * 4

   ESCREVAL ("Multa por excesso de peso : R$ ", Multa)
SENAO
   ESCREVAL ("Liberado")
FIMSE

FIMALGORITMO
```

2) Crie um programa que dada a idade de um jogador classifique-o em uma das seguintes categorias:

Infantil	5 a 10 anos
Juvenil	11 a 17 anos
Adulto	Maiores de 18 anos

Anotações

```
ALGORITMO "Capítulo 08 Exercício 02"

// Capítulo  08 - Estruturas de Decisão
// Exercício 02 - Crie um programa que dada a idade de um jogador
classifique-o em uma das seguintes categorias :

// Infantil    5 a 10 anos
// Juvenil     11 a 17 anos
// Adulto Maiores de 18 anos

VAR

Idade : INTEIRO

INICIO
// Seção de Comandos
ESCREVA("Digite a idade do jogador : ")
LEIA(Idade)
ESCOLHA Idade
CASO 0, 1, 2, 3, 4
   ESCREVAL("Muito Novo")
CASO 5, 6, 7, 8, 9, 10
   ESCREVAL("Infantil")
CASO 11, 12, 13, 14, 15, 16, 17
   ESCREVAL("Juvenil")
OUTROCASO
   ESCREVAL("Adulto")
FIMESCOLHA

FIMALGORITMO
```

3) Faça um programa em que o usuário informa à hora que inicia o seu turno de trabalho e é exibido na tela se é manhã, tarde ou noite:

Manhã	05h00min às 13h00min
Tarde	13h00min às 21h00min
Noite	21h00min às 05h00min

```
ALGORITMO "Capítulo 08 Exercício 03"
```

	Anotações

```
// Capítulo  08 - Estruturas de Decisão
// Exercício 03 - Faça um programa em que o usuário informa a hora
que inicia o seu turno de trabalho e é exibido na tela se é manhã,
tarde ou noite :
//   Manhã = 05h00min às 13h00min
//   Tarde = 13h00min às 21h00min
//   Noite = 21h00min às 05h00min

var
Horario : INTEIRO

INICIO

// Seção de Comandos
ESCREVA ("Digite a horário : ")
LEIA (horario)

SE (Horario >= 5) e (Horario < 13) ENTAO
   ESCREVAL ("Manhã")
SENAO
   SE (Horario >= 13) e (Horario < 17) ENTAO
      ESCREVAL ("Tarde")
   SENAO
      ESCREVAL ("Noite")
   FIMSE
FIMSE

FIMALGORITMO
```

4) Elabore um programa que calcule quanto o cliente de um posto de gasolina irá pagar de acordo com a opção do combustível escolhido e a quantidade de litros comprados. Utilize os dados da tabela para desenvolver o algoritmo;

Código	Combustível	Preço
A	Álcool	1, 56/l
G	Gasolina	2, 56/l
D	Diesel	1, 81/l

Anotações

```
ALGORITMO "Capítulo 08 Exercício 04"

// Capítulo  08 - Estruturas de Decisão
// Exercício 04 - Elabore um programa que calcule quanto o cliente
de um posto de gasolina irá pagar de acordo com a opção do
combustível escolhido e a quantidade de litros comprados. Utilize os
dados da tabela para desenvolver o algoritmo;

// Código     Combustível Preço
// a          Álcool      R$ 1,56/l
// G          Gasolina    R$ 2,56/l
// D          Diesel      R$ 1,81/l

VAR
Litros, Valor : REAL
Combustivel : CARACTER

INICIO

ESCREVA ("Quantidade de litros : ")
LEIA (Litros)

ESCREVA ("Escolha a = álcool, G = gasolina, D = diesel : ")
LEIA (Combustivel)

ESCOLHA Combustivel
CASO "A"
   Valor := Litros * 1.56
CASO "G"
   Valor := Litros * 2.56
CASO "D"
   Valor := Litros * 1.81
OUTROCASO
   ESCREVAL ("Opção errada")
FIMESCOLHA
```

Anotações

```
ESCREVAL ("Total a pagar : R$ ", Valor)

FIMALGORITMO
```

5) Faça um algoritmo que calcule o valor da conta de luz de uma pessoa. Sabe-se que o cálculo da conta de luz segue a tabela abaixo:

Tipo de Cliente	Valor do KW/h
Residência	R$ 0, 83
Comércio	R$ 0, 62
Indústria	R$ 2, 03

```
ALGORITMO "Capítulo 08 Exercício 05"

// Capítulo  08 - Estruturas de Decisão
// Exercício 05 - Faça um algoritmo que calcule o valor da conta de
luz de uma pessoa. Sabe-se que o cálculo da conta de luz segue a
tabela abaixo :

// Tipo de Cliente  Valor do KW/h
// Residência  R$ 0,83
// Comércio    R$ 0,62
// Indústria   R$ 2,03

// Seção de Declarações
VAR
Residencia, KW : INTEIRO
Consumo : REAL

INICIO
// Seção de Comandos
ESCREVAL("Informe o tipo de residência : ")
ESCREVAL("1 - Residência")
ESCREVAL("2 - Comércio")
ESCREVAL("3 - Indústria")
LEIA(Residencia)
ESCOLHA Residencia
CASO 1
   ESCREVA("Informe a quantidade de KW gastos no mês : ")
```

Anotações

```
   LEIA(KW)
   Consumo := kw * 0.83
   ESCREVAL("Valor da conta : ", Consumo)
CASO 2
   ESCREVA("Informe a quantidade de KW gastos no mês : ")
   LEIA(KW)
   Consumo := kw * 0.62
   ESCREVAL("Valor da conta : ", Consumo)
CASO 3
   ESCREVA("Informe a quantidade de KW gastos no mês : ")
   LEIA(KW)
   Consumo := kw * 2.03
   ESCREVAL("Valor da conta : ", Consumo)
OUTROCASO
   ESCREVAL("Tipo de residência inválido!")
FIMESCOLHA

FIMALGORITMO
```

Anotações

Capítulo 09 - Laços de Repetição

1) Desenvolva um programa que calcule e exiba a soma dos mil primeiros números inteiros.

```
ALGORITMO "Capítulo 09 Exercício 01"

// Capítulo  09 - Estruturas de Repetição
// Exercício 01 - Desenvolva um programa que calcule e exiba a soma
dos mil primeiros números inteiros.

VAR
Numero, Soma : INTEIRO

INICIO
// Seção de Comandos

PARA Numero DE 1 ATE 1000 FACA

   Soma := Soma + Numero

FIMPARA

ESCREVA ("A soma dos primeiros mil números é ", Soma)

FIMALGORITMO
```

2) Elaborar um programa que converta valores em graus Celsius para Fahrenheit de 1 a 100.

```
ALGORITMO "Capítulo 09 Exercício 02"

// Capítulo  09 - Estruturas de Repetição
// Exercício 02 - Elaborar um programa que converta valores em graus
Celsius para Fahrenheit de 1 a 100.

VAR
Celsius : INTEIRO
Fahrenheit : REAL

INICIO
```

	Anotações

```
// Seção de Comandos

PARA Celsius DE 1 ATE 100 FACA

   Fahrenheit := (Celsius * 1.8 + 32)

   ESCREVAL ("°C ", Celsius, " = ", "°F", Fahrenheit)

FIMPARA

FIMALGORITMO
```

3) Escreva um algoritmo que calcule o fatorial (N!) de um determinado número fornecido pelo usuário.

```
ALGORITMO "Capítulo 09 Exercício 03"

// Capítulo  09 - Estruturas de Repetição
// Exercício 03 - Escreva um algoritmo que calcule o fatorial (N!)
de um determinado número fornecido pelo usuário.

VAR
Contador : INTEIRO
N : INTEIRO
Resultado : INTEIRO

INICIO
// Seção de Comandos

ESCREVA("Fatorial : ")
LEIA (N)

Resultado := 1

PARA Contador DE 1 ATE N FACA

   Resultado := Resultado * Contador

FIMPARA
```

Anotações

```
ESCREVAL (N, "! =", Resultado)

FIMALGORITMO
```

4) Sabendo que a séria de Fibonacci é formada pela sequência 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ... e que seus termos, a partir do terceiro, é igual à soma dos dois anteriores, desenvolva um programa que exiba a série até o 50º termo.

```
ALGORITMO "Capítulo 09 Exercício 04"

// Capítulo  09 - Estruturas de Repetição
// Exercício 04 - Sabendo que a séria de Fibonacci é formada pela
sequência 1, 1, 2, 3, 5, 8, 13, 21,34, 55, ...  e que seus termos, a
partir do terceiro, é igual à soma dos dois anteriores, desenvolva
um programa que exiba a série até o 50° termo.

VAR
Proximo : REAL
Atual : REAL
Anterior : REAL
Contador : INTEIRO

INICIO
// Seção de Comandos

Proximo := 0
Atual := 0
Anterior := 1

Para Contador de 1 ate 50 Faca

   Proximo := Atual + Anterior

   ESCREVAL (Proximo)

   Anterior := Atual
   Atual := Proximo
```

	Anotações

```
FIMPARA

FIMALGORITMO
```

5) A eleição do grêmio estudantil de uma escola possui três chapas candidatas. Os códigos de cada chapa são 11, 12, 13, os votos nulos é 14, os brancos 15. Elabore um algoritmo que calcule e escreva:

   O total de votos para cada chapa e o percentual de cada sobre uma sobre o total

   - O total de nulos e seu percentual sobre o total
   - O total de brancos e seu percentual sobre o total

```
ALGORITMO "Capítulo 09 Exercício 05"

// Capítulo  09 - Estruturas de Repetição
// Exercício 05 - a eleição do grêmio estudantil de uma escola
possui três chapas candidatas. os códigos de cada chapa são 11, 12,
13, os votos nulos é 14. os brancos 15. Elabore um algoritmo que
calcule e escreva :
//    a) o total de votos para cada chapa e o percentual de cada
sobre uma sobre o total
//    b) o total de nulos e seu percentual sobre o total
//    c) o total de brancos e seu percentual sobre o total

VAR
Chapa1 : REAL
Chapa2 : REAL
Chapa3 : REAL
Branco : REAL
Nulo : REAL
Voto : REAL

TotalChapa1 : REAL
TotalChapa2 : REAL
TotalChapa3 : REAL
TotalBranco : REAL
TotalNulo : REAL
TotalVoto : REAL
```

	Anotações

```
PercentualChapa1 : REAL
PercentualChapa2 : REAL
PercentualChapa3 : REAL
PercentualBranco : REAL
PercentualNulo : REAL

Inicio
// Seção de Comandos

ESCREVAL ("Registre seu voto : ")
ESCREVAL ("11 = Chapa 1")
ESCREVAL ("12 = Chapa 2")
ESCREVAL ("13 = Chapa 3")
ESCREVAL ("14 = Branco")
ESCREVAL ("15 = Nulo")
ESCREVAL ("99 = Sair")

ENQUANTO Voto <> 99 FACA

   ESCREVA ("Qual o seu voto? ")
   LEIA (Voto)

   TotalVoto := TotalVoto + 1

   ESCOLHA Voto
   CASO 11
      TotalChapa1 := TotalChapa1 + 1
   CASO 12
      TotalChapa2 := TotalChapa2 + 1
   CASO 13
      TotalChapa3 := TotalChapa3 + 1
   CASO 14
      TotalBranco := TotalBranco + 1
   CASO 15
      TotalNulo := TotalNulo + 1
   FIMESCOLHA
```

	Anotações

```
FIMENQUANTO

TotalVoto := TotalVoto - 1

PercentualChapa1 := TotalChapa1 / TotalVoto
PercentualChapa2 := TotalChapa2 / TotalVoto
PercentualChapa3 := TotalChapa3 / TotalVoto
PercentualBranco := TotalBranco / TotalVoto
PercentualNulo := TotalNulo / TotalVoto

ESCREVAL ("Apuração de voto:")
ESCREVAL ("Total", TotalVoto)
ESCREVAL ("Chapa 1", TotalChapa1, " votos e um percentual de ",
PercentualChapa1)
ESCREVAL ("Chapa 2", TotalChapa2, " votos e um percentual de ",
PercentualChapa2)
ESCREVAL ("Chapa 3", TotalChapa3, " votos e um percentual de ",
PercentualChapa3)
ESCREVAL ("Branco", TotalBranco, " votos e um percentual de ",
PercentualBranco)
ESCREVAL ("Nulo", TotalNulo, " votos e um percentual de ",
PercentualNulo)

FIMALGORITMO
```

Anotações

Capítulo 10 - Variáveis Indexadas

1) Escreva um algoritmo que leia os nomes de 15 funcionários de uma fábrica e os seus respectivos salários e os armazene em dois vetores diferentes. Em seguida, exiba uma lista com o nome e o salário de cada um.

```
ALGORITMO "Capítulo 10 Exercício 01"

// Capítulo  10 - Variáveis Indexadas
// Exercício 01 - Escreva um algoritmo que leia os nomes de 15
funcionários de uma fábrica e os seus respectivos salários e os
armazene em dois vetores diferentes. em seguida, exiba uma lista com
o nome e o salário de cada um.

// Seção de Declarações
VAR

Nome : VETOR [1..15] de CARACTER
Salario : VETOR [1..15] de REAL
Contador : INTEIRO

INICIO

PARA Contador DE 1 ATE 15 FACA

   ESCREVAL(Contador, "° Funcionário")
   ESCREVA("Nome: ")
   LEIA(Nome[Contador])

   ESCREVA("Salário: ")
   LEIA(Salario[Contador])
   ESCREVAL

FIMPARA

PARA Contador DE 1 ATE 15 FACA

   ESCREVAL(Contador, "° Funcionário: ", Nome[Contador],
Salario[Contador])
```

	Anotações

```
FIMPARA

FIMALGORITMO
```

2) Escreva um algoritmo que leia os elementos (números inteiros) de uma matriz de 5 linhas e 5 colunas e os exiba.

```
ALGORITMO "Capítulo 10 Exercício 02"

// Capítulo  10 - Variáveis Indexadas
// Exercício 02 - Escreva um algoritmo que leia os elementos
(números inteiros) de uma matriz de 5 linhas e 5 colunas e os exiba.

// Seção de Declarações
VAR

Elementos : VETOR [1..5, 1..5] de INTEIRO
Linha : INTEIRO
Coluna : INTEIRO

INICIO

PARA Linha DE 1 ATE 5 FACA

   PARA Coluna DE 1 ATE 5 FACA

      ESCREVA(Linha, "ª Linha, ", Coluna, "ª Coluna: ")
      LEIA(Elementos[Linha, Coluna])

   FIMPARA

FIMPARA

PARA Linha DE 1 ATE 5 FACA

   PARA Coluna DE 1 ATE 5 FACA
```

	Anotações

```
      ESCREVAL(Linha, "ª Linha, ", Coluna, "ª Coluna: ",
Elementos[Linha, Coluna])

   FIMPARA

FIMPARA

FIMALGORITMO
```

3) Multiplique os elementos da matriz que você construiu no exercício anterior por 10 e os exiba.

```
ALGORITMO "Capítulo 10 Exercício 03"

// Capítulo  10 - Variáveis Indexadas
// Exercício 03 - Multiplique os elementos da matriz que você
construiu no exercício anterior por 10 e os exiba.

// Seção de Declarações
VAR

Elementos : VETOR [1..5, 1..5] de INTEIRO
Linha : INTEIRO
Coluna : INTEIRO

INICIO

PARA Linha DE 1 ATE 5 FACA

   PARA Coluna DE 1 ATE 5 FACA

      ESCREVA(Linha, "ª Linha, ", Coluna, "ª Coluna: ")
      LEIA(Elementos[Linha, Coluna])

   FIMPARA

FIMPARA

LIMPATELA
```

	Anotações

```
PARA Linha DE 1 ATE 5 FACA

   PARA Coluna DE 1 ATE 5 FACA

      ESCREVAL(Linha, "ª Linha, ", Coluna, "ª Coluna: ",
Elementos[Linha, Coluna] * 10)

   FIMPARA

FIMPARA

FIMALGORITMO
```

4) Escreva um algoritmo que armazene e mostre os nomes dos 11 jogadores titulares de 5 times de futebol.

```
ALGORITMO "Capítulo 10 Exercício 04"

// Capítulo  10 - Variáveis Indexadas
// Exercício 04 - Escreva um algoritmo que armazene e mostre os
nomes dos 11 jogadores titulares de 5 times de futebol.

// Seção de Declarações
VAR

Jogadores : VETOR [1..11, 1..5] de CARACTER
Linha : INTEIRO
Coluna : INTEIRO

INICIO

PARA Coluna DE 1 ATE 5 FACA

   PARA Linha DE 1 ATE 11 FACA

      ESCREVA(Coluna, "° Time, ", Linha, "° Jogador: ")
      LEIA(Jogadores[Linha, Coluna])
```

Anotações

```
   FIMPARA

FIMPARA

LIMPATELA

PARA Coluna DE 1 ATE 5 FACA

   PARA Linha DE 1 ATE 11 FACA

      ESCREVAL(Coluna, "° Time, ", Linha, "° Jogador: ",
Jogadores[Linha, Coluna])

   FIMPARA

FIMPARA

FIMALGORITMO
```

5) Escreva um algoritmo que armazene valores inteiros em uma matriz 3 x 4 e calcule e mostre a média aritmética dos valores digitados.

```
ALGORITMO "Capítulo 10 Exercício 05"

// Capítulo  10 - Variáveis Indexadas
// Exercício 05 - Escreva um algoritmo que armazene valores inteiros
em uma matriz 3 x 4 e calcule e mostre a média aritmética dos
valores digitados.

// Seção de Declarações
VAR

Elementos : VETOR [1..3, 1..4] de INTEIRO
Linha : INTEIRO
Coluna : INTEIRO
Media: REAL

INICIO
```

	Anotações

```
PARA Linha DE 1 ATE 3 FACA

   PARA Coluna DE 1 ATE 4 FACA

      ESCREVA(Linha, "ª Linha, ", Coluna, "ª Coluna: ")
      LEIA(Elementos[Linha, Coluna])

      Media := Media + Elementos[Linha, Coluna]

   FIMPARA

FIMPARA

Media := Media / (3 * 4)

ESCREVAL("Média: ", Media)


FIMPARA

FIMALGORITMO
```

Capítulo 11 - Depuração de Códigos

1) Quais são as principais vantagens de aplicar técnicas de depuração de códigos?

   Depurar um código previne erros dos quais poderiam causar problemas no futuro. Previnem possíveis erros que ocorreriam em tempo de execução, não trazendo o objetivo que era esperado.

2) Quando desenvolvemos uma aplicação, existem ao menos três maneiras de verificar se nosso código realizará ou não o esperado. Cite quais são e classifique o tipo de erro qual se resolve utilizando essa técnica (Execução, Compilação).

   Teste de Mesa - Erros em Tempo de execução.
   Breakpoints - Erros em Tempo de execução.
   Programação da Emoção - Não previne nada.

3) Quais os comandos que adicionando ao código do VisuAlg auxiliam a depuração de código? Qual a diferença entre eles?

	Anotações

Existem dois comandos para depuração de códigos com o VisuAlg, sendo eles PAUSA e DEBUG.

O comando PAUSA faz com que nosso programa pare em Tempo de execução para que possamos averiguar os valores de variáveis até aquele momento.

O comando DEBUG funciona semelhantemente ao comando PAUSA porém somente quando um determinado operador lógico for VERDADEIRO. Portanto, somente será pausado se a expressão lógica for verdade.

4) A quem cabe a responsabilidade de depurar o código existente? Assinale a alternativa correta.
   a. Analista de Testes
   b. Cliente
   c. Programador
   d. Gerente do Produto
   e. Analista de Sistemas

Anotações

Capítulo 12 - Subalgoritmos

1) Escreva um algoritmo e crie, nele, uma função que receba um número e retorne a soma dos números inteiros positivos e menores que o número digitado. Se o número digitado for negativo ou nulo, não chama a função e mostra uma mensagem de erro.

```
ALGORITMO "Capítulo 11 Exercício 01"

// Capítulo  11 - Subalgoritmos
// Exercício 01 - Escreva um algoritmo e crie, nele, uma função que
receba um número e retorne a soma dos números inteiros positivos e
menores que o número digitado. se o número digitado for negativo ou
nulo, não chama a função e mostra uma mensagem de erro.

VAR
// Declaração das variáveis globais
Numero: INTEIRO

// Declaração das funções
FUNCAO SomaNumeros(NumeroFinal:INTEIRO): INTEIRO

// Declaração de variáveis locais
VAR
Contador, Soma : INTEIRO

INICIO

// Instruções
PARA Contador DE 1 ATE NumeroFinal FACA

   Soma := Soma + Contador

FIMPARA

// Valor de retorno
RETORNE Soma

FIMFUNCAO
```

Anotações

```
INICIO
ESCREVA ("Número: ")
LEIA (Numero)

// Chamada da função

SE Numero <= 0 ENTAO
   ESCREVAL("Número inválido!")
SENAO
   ESCREVAL (SomaNumeros(Numero))
FIMSE

FIMALGORITMO
```

2) Escreva um algoritmo que leia duas notas de um aluno e chame uma função que calcula a média aritmética desse aluno.

```
ALGORITMO "Capítulo 11 Exercício 02"

// Capítulo 11 - Subalgoritmos
// Exercício 02 - Escreva um algoritmo que leia duas notas de um
aluno e chame uma função que calcula a média aritmética desse aluno.

VAR
// Declaração das variáveis globais
Nota1, Nota2, Media: REAL

// Declaração das funções
FUNCAO MediaSimples(NotaA, NotaB: REAL):REAL

// Declaração de variáveis locais
VAR
Resultado : REAL

INICIO

// Instruções
```

	Anotações

```
Resultado := (NotaA + NotaB) / 2)

// Valor de retorno
RETORNE Resultado

FIMFUNCAO

INICIO
ESCREVA("1ª Nota: ")
LEIA (Nota1)

ESCREVA("2ª Nota: ")
LEIA (Nota2)


// Chamada da função
Media := MediaSimples(Nota1, Nota2)

ESCREVAL

ESCREVAL ("Média: ", Media)

FIMALGORITMO
```

3) Escreva um algoritmo que receba um número inteiro e chame um procedimento que calcule o quadrado desse número. Crie esse procedimento.

```
ALGORITMO "Capítulo 11 Exercício 03"

// Capítulo  11 - Subalgoritmos
// Exercício 03 - Escreva um algoritmo que receba um número inteiro
e chame um procedimento que calcule o quadrado desse número. Crie
esse procedimento.

VAR
// Declaração das variáveis globais
Numero: REAL
```

	Anotações

```
// Declaração dos Procedimentos
PROCEDIMENTO Quadrado(Valor: REAL)

// Declaração de variáveis locais
VAR
Resultado : REAL

INICIO

// Instruções
Resultado := Valor ^ 2

// Exibe o Resultado
ESCREVAL(Valor, "² = ", Resultado)

FIMPROCEDIMENTO

INICIO
ESCREVA("Número: ")
LEIA (Numero)

// Chamada do Procedimento
Quadrado(Numero)

FIMALGORITMO
```

4) Escreva um algoritmo que receba um número inteiro e chame um procedimento que receba um número e mostre a raiz quadrada do mesmo.

```
ALGORITMO "Capítulo 11 Exercício 04"

// Capítulo  11 - Subalgoritmos
// Exercício 04 - Escreva um algoritmo que receba um número inteiro
e chame um procedimento que receba um número e mostre a raiz
quadrada do mesmo.

VAR
// Declaração das variáveis globais
```

	Anotações

```
Numero: REAL

// Declaração das funções
PROCEDIMENTO RaizQuadrada(Valor: REAL)

// Declaração de variáveis locais
VAR
Resultado : REAL

INICIO

// Instruções
Resultado := Valor ^ 0.5

// Exibe o Resultado
ESCREVAL("Raiz Quadrada de ", Valor, "= ", Resultado)

FIMPROCEDIMENTO

INICIO
ESCREVA("Número: ")
LEIA (Numero)

// Chamada do Procedimento
RaizQuadrada(Numero)

FIMALGORITMO
```

5) Faça um algoritmo que leia três números e crie uma função que calcule o quadrado desses números e retorne o valor.

```
ALGORITMO "Capítulo 11 Exercício 05"

// Capítulo  11 - Subalgoritmos
// Exercício 05 - Faça um algoritmo que leia três números e crie uma
função que calcule o quadrado desses números e retorne o valor.

VAR
```

	Anotações

```
// Declaração das variáveis globais
Numero : VETOR [1..3] de REAL
Contador: INTEIRO


// Declaração das funções
FUNCAO Quadrado(Valor: REAL):REAL

// Declaração de variáveis locais
VAR
Resultado : REAL

INICIO

// Instruções
Resultado := Valor ^ 2

// Valor de retorno
RETORNE Resultado

FIMFUNCAO

INICIO

PARA Contador DE 1 ATE 3 FACA

   ESCREVA(Contador, "° Número: ")
   LEIA(Numero[Contador])

FIMPARA

ESCREVAL

PARA Contador DE 1 ATE 3 FACA

   // Chamada da Função
```

	Anotações

```
   ESCREVAL(Contador, "° Número: ", Numero[Contador]," ² = ",
Quadrado(Numero[Contador]))

FIMPARA

FIMALGORITMO
```

Anotações

Capítulo 13 - Introdução ao Banco de Dados

1) Palavras Cruzadas

Horizontal

3. **TABELA**—Uma simples estrutura de linhas e colunas. Em uma tabela, cada linha contém um mesmo conjunto de colunas.

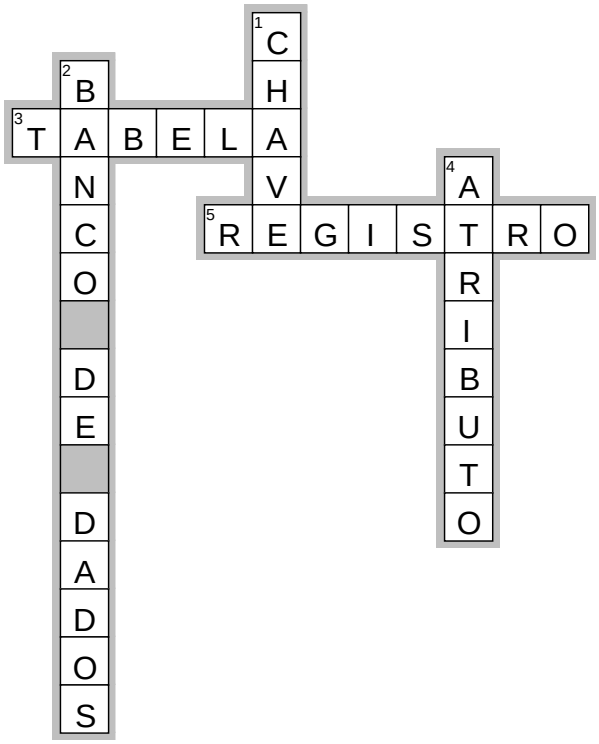
5. **REGISTRO**—Cada linha formada por uma lista ordenada de colunas representa um registro. Os registros não precisam conter informações em todas as colunas, podendo assumir valores nulos quando assim se fizer necessário.

Vertical

1. **CHAVE**—Conjunto de um ou mais atributos que determinam a unicidade de cada registro.

2. **BANCO DE DADOS**—Coleção organizada de dados, armazenados num meio físico, para tratamento posterior

4. **ATRIBUTO**—Informação referente à exibição ou apresentação da informação



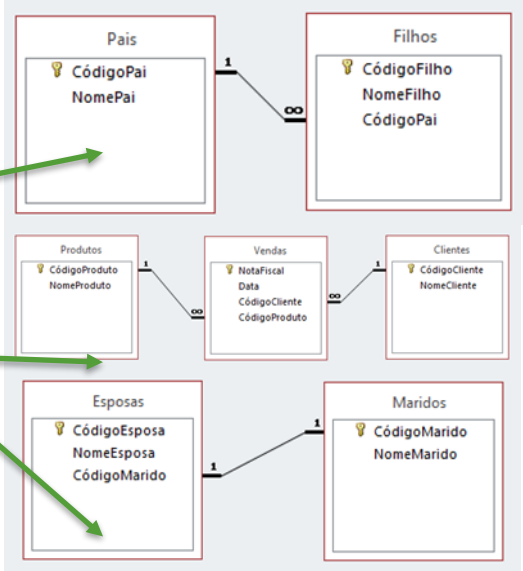
Anotações

2) Associe:

Um para um (1 para 1)

Um para muitos (1 para ∞)

Muitos para muitos (∞ para ∞)



Anotações

Capítulo 14 - Introdução à Orientação-a-Objetos

1) No mundo real e no computacional, objetos se assemelham de duas maneiras. Cite quais são.

   Objetos se assemelham por terem propriedades que definem seu estado físico e por seus comportamentos que definem o que o objeto é capaz de fazer, suas ações.

2) Como é feito o relacionamento entre objetos no mundo computacional quando desenvolvemos nosso código utilizando uma linguagem orientada-a-objetos?

   O relacionamento entre os objetos é feito através de mensagens.

3) Considerando as Classes Professor, Aluno e Pessoa, ilustre o conceito de Classe Base e Classe Derivada. Desenvolva uma forma que ao menos 3 propriedades e 2 comportamentos sejam compartilhados.

   Super Classe Pessoa

   - Propriedades:
     - Nome
     - Endereço
     - Telefone
     - E-mail pessoal
   - Comportamentos:
     - Andar
     - Falar
     - Comer
     - Compreender

   Classe Filha Professor

   - Propriedades:
     - E-mail corporativo
     - Número do crachá
     - Matéria que aplica
   - Comportamentos:
     - Aplicar prova
     - Corrigir prova
     - Dar nota
     - Tirar duvidas

   Classe Filha Aluno

   - Propriedades:

	Anotações

    - Curso inscrito
    - Matrícula
    - Situação (aprovado/reprovado)
- Comportamentos:
    - Aprender
    - Realizar trabalhos acadêmicos
    - Fazer provas
    - Copiar matérias

4) Indique as alternativas corretas (**V**) e as falsas (**F**) sobre Programação Orientada-a-Objetos:
    a. (**F**) O desenvolvimento orientado a objetos é uma maneira de trazer a realidade para o computador, porém tem uma curva de aprendizado longa e complexa.
    b. (**V**) O código desenvolvido por uma linguagem orientada-a-objetos é fácil de compreender, contempla a realidade e aproxima a área de negócios com a de programação.
    c. (**V**) Herança é um conceito dentro de linguagens orientadas-a-objetos da qual uma classe derivada é capaz de especializar uma superclasse.
    d. (**V**) Cada objeto é único. Objetos se assemelham em relação a sua estrutura, através de sua Classe.
    e. (**F**) O relacionamento entre objetos é entendido como Instancia de uma Classe.
5) Dentre as principais linguagens orientadas-a-objetos, podemos dizer que:
    a. Apple é a linguagem utilizada para desenvolver no ambiente Iphone, Ipod e Ipad.
    b. C#.NET e VB.NET tem como princípio o desenvolvimento para ambientes Windows, porém não compartilham de ferramentas semelhantes.
    c. Java é uma das linguagens mais utilizadas no mundo e contém uma das maiores comunidades de programação.
    d. **C#.NET é a linguagem de programação foco da Microsoft. Utiliza de ferramentas compartilhadas com VB.NET e é escrita em uma sintaxe semelhante ao Java.**

Anotações