Lógica de Programação

Comandos Básicos Envolvendo Variáveis de Memória

Diretor Executivo
DAVID LIRA STEPHEN BARROS
Gerente Editorial
CRISTIANE SILVEIRA CESAR DE OLIVEIRA
Projeto Gráfico
TIAGO DA ROCHA
Autoria
IZABELLY MORAIS DE MORAIS
LEANDRO C. CARDOSO
MAX ANDRÉ DE AZEVÊDO SILVA

AUTORIA

Izabelly Morais de Morais

Sou licenciada em Ciência da Computação pela Universidade Federal da Paraíba (UFPB), e mestre em Ciência da Computação com ênfase em Engenharia de Software e Linguagens de Programação pela Universidade Federal de Pernambuco (UFPE). Leciono como professora formadora no Instituto Federal de Pernambuco (IFPE) e na Faculdade Pitágoras (João Pessoa, na Paraíba), onde tenho a oportunidade de transmitir minha experiência na área de Tecnologia e Educação. Por isso fui convidada pela Editora Telesapiens a integrar seu elenco de autores independentes. Estou muito feliz em poder ajudar você nesta fase de muito estudo e trabalho. Conte comigo!

Leandro C. Cardoso

Sou graduado em Comunicação Social com habilitação em Design Digital, e mestre em Tecnologias da Inteligência e Design Digital pela Pontifícia Universidade Católica de São Paulo (PUC-SP), com mais de 20 anos de experiência em direção de arte e criação. Passei por empresas como a Laureate International Universities – FMU/Fiam-Faam, a Universidade Anhembi Morumbi e o Centro Paula Souza (Fatec-Etec). Já atuei como analista de desenvolvimento pedagógico sênior, coordenador de curso técnico de Design Gráfico e revisor técnico e validador para curso EAD para clientes Laureate International Universities, DeVry Brasil, Unef, FAESF, Faculdade Positivo, Uninter e Platos Soluções Educacionais S.A. (Krotonn – Universidade Anhanguera). Além disso, sou autor de mais de 24 livros didáticos e um dos organizadores da Maratona de Criação e Design do Curso de Comunicação Visual da Etec Albert Einstein. Sou apaixonado pelo que faço e adoro transmitir minha experiência de vida àqueles que estão iniciando em suas profissões. Por isso fui convidado pela Editora Telesapiens a integrar seu elenco de autores independentes. Estou muito feliz em poder ajudar você nesta fase de muito estudo e trabalho. Conte comigo!

Max André de Azevêdo Silva

Sou formado em Ciência da Computação, pela Universidade Federal da Paraíba (UFPB), e mestre em Ciência da Computação com ênfase em Engenharia de Software pela Universidade Federal da Paraíba (UFPB). Tenho experiência na área de Desenvolvimento de Sistemas Web, Mobile e Jogos Eletrônicos. Atualmente, trabalho como analista de sistemas. Por isso fui convidado pela Editora Telesapiens a integrar seu elenco de autores independentes. Estou muito feliz em poder ajudar você nesta fase de muito estudo e trabalho. Conte comigo!

ICONOGRÁFICOS

Olá. Esses ícones irão aparecer em sua trilha de aprendizagem toda vez que:

OBJETIVO:
para o início do desenvolvimento de uma nova competência;

NOTA:
quando forem necessários observações ou complementações para o seu conhecimento;

EXPLICANDO MELHOR:
algo precisa ser melhor explicado ou detalhado;

SAIBA MAIS:
textos, referências bibliográficas e links para aprofundamento do seu conhecimento;

ACESSE:
se for preciso acessar um ou mais sites para fazer download, assistir vídeos, ler textos, ouvir podcast;

ATIVIDADES:
quando alguma atividade de autoaprendizagem for aplicada;

DEFINIÇÃO:
houver necessidade de se apresentar um novo conceito;

IMPORTANTE:
as observações escritas tiveram que ser priorizadas para você;

VOCÊ SABIA?
curiosidades e indagações lúdicas sobre o tema em estudo, se forem necessárias;

REFLITA:
se houver a necessidade de chamar a atenção sobre algo a ser refletido ou discutido sobre;

RESUMINDO:
quando for preciso se fazer um resumo acumulativo das últimas abordagens;

TESTANDO:
quando o desenvolvimento de uma competência for concluído e questões forem explicadas;

SUMÁRIO

UNIDADE 02

INTRODUÇÃO

Você sabia que a área de desenvolvimento de algoritmos estruturados envolvendo operações com variáveis e comandos condicionais simples é muito importante para lógica de programação? Isso mesmo. Para isso, é importante o conhecimento prévio do universo de hardware e software para o aprofundamento no mundo dos algoritmos computacionais, começando pela manipulação de constantes e variáveis de memória simples. Lembre-se que é fundamental você exercitar ao máximo e testar tudo que foi aprendido. Para tanto, é importante resolver os problemas de algoritmização. Entendeu? Ao longo desta unidade letiva você vai mergulhar neste universo!

OBJETIVOS

Olá. Seja muito bem-vindo à Unidade 2. Nosso objetivo é auxiliar você no desenvolvimento das seguintes competências profissionais até o término desta etapa de estudos:

1. Manipular constantes e variáveis de memória em uma solução algorítmica.
2. Utilizar expressões aritméticas envolvendo constantes e variáveis numéricas em soluções algorítmicas.
3. Utilizar expressões literais envolvendo constantes e variáveis alfanuméricas em soluções algorítmicas.
4. Aplicar estruturas condicionais “SE” em soluções algorítmicas.

Constantes e Variáveis de Memória

OBJETIVO:

Ao término deste Capítulo você será capaz de entender os conceitos de constantes e variáveis de memória, bem como a forma de manipulá-las em uma solução algorítmica. Isso será fundamental para o exercício de sua profissão. E então? Motivado para desenvolver esta competência? Então vamos lá. Avante!.

Conceito e Tipos de dados e Variáveis

Como já estudamos anteriormente, em uma solução algorítmica, sempre haverá um dado a ser processado e transformado em informação ou ação. Logo que esse dado entra no sistema computacional, ele precisa ser armazenado em uma variável de memória. A partir de então, inicia-se o processamento do dado propriamente dito, podendo este passar por uma série de outras variáveis de memória, a depender do objetivo do processamento e da lógica de programação algorítmica pensada pelo programador. Então, precisamos entender melhor o que é e como se comporta esse dado a ser armazenado em uma variável de memória.

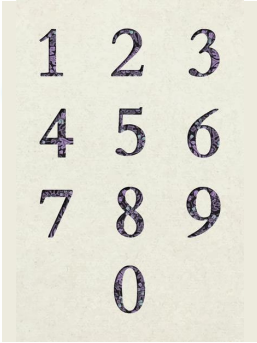
DEFINIÇÃO:

Dados são fragmentos de informações que precisam ser processados para se transformar em uma informação completa e inteligível.

Os dados, portanto, podem ser qualquer fragmento de informação que possa ser armazenado na memória do computador. Estamos falando de um nome, um número telefônico, um endereço, um salário, uma fotografia, uma música, enfim, tudo o que possa ser armazenado e venha a participar de um processamento. Embora um dado possa ser tudo isso, é classificado em três tipos fundamentais: numérico, alfanumérico e lógico.

O dado numérico pode ser um número inteiro ou fracionário, positivo ou negativo, de todo e qualquer tamanho, inclusive zero.

Figura 1 – O dado numérico pode ser um número inteiro ou fracionário, positivo ou negativo, inclusive zero



Fonte: Freepik

A seguir, vemos alguns exemplos de como esses dados são representados em um algoritmo:

- 4568 = número inteiro positivo; 35,5 = número fracionário positivo.
- -104 = número inteiro negativo.
- -0,458 = número fracionário negativo.
- 4,5678E+7 = número inteiro positivo em notação científica de base 10 (quatro vírgula cinco, seis, sete, oito, vezes dez elevado à sétima potência), equivalendo ao número inteiro 45678000.
- -4,5678E+7 = número inteiro negativo em notação científica de base 10 (menos quatro vírgula cinco, seis, sete, oito, vezes dez elevado à sétima potência), equivalendo ao número inteiro -45678000.
- 6,5E-3 = número fracionário positivo em notação científica de base 10 (seis vírgula cinco vezes dez elevado a menos três), equivalendo ao número 0,0065.
- -6,5E-3 = número fracionário negativo em notação científica de base 10 (menos seis vírgula cinco vezes dez elevado à menos três), equivalendo ao número 0,0065.

Um dado alfanumérico, também denominado "dado literal" ou "caractere", pode ser um texto contendo letras, algarismos e outros símbolos especiais; um dado alfanumérico tem seu tamanho medido em número de caracteres.

Figura 2 – Um dado alfanumérico também é chamado de "dado literal" ou "caractere"

Fonte: Freepik

Um caractere é a menor parte que compõe um dado desse tipo.

IMPORTANTE:

Em inglês, o dado alfanumérico é conhecido como string, que significa "cadeia" ou "corda". Esse termo remete ao sentido de cadeia de caracteres, ou seja, um conjunto de caracteres concatenados.

Esse tipo de dado é representado por quaisquer valores, símbolos, palavras ou frases entre aspas, como demonstrado a seguir:

- "Telesapiens" = palavra.
- "045223-X" = texto alfanumérico, contendo números e caracteres.
- "@#$%&*()" = texto alfanumérico contendo símbolos.

O que diferencia um dado numérico de um alfanumérico, quanto à sua representação em um algoritmo, é a presença ou não de um par de delimitadores tipo aspas ("..."). Assim sendo, podemos afirmar que os dados a seguir são completamente diferentes sob o ponto de vista do tipo de dado (MORAIS; AZEVEDO, 2017a):

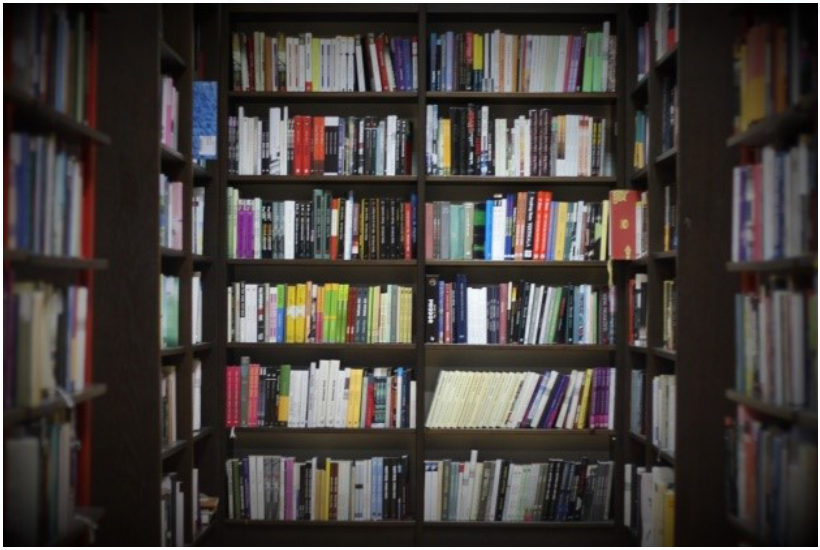
126 <> "1221"

O dado à esquerda se refere ao número 126 (cento e vinte e seis), já o da direita não representa um número, mas sim uma cadeia de caracteres composta pelos símbolos "1", "2", "2", e "1". Dessa maneira, não há como comparar os dois dados acima. Em relação a um dado lógico, também conhecido como "dado booleano", é um tipo de dado que só pode assumir um entre dois valores: verdadeiro ou falso. Dada essa característica, esse tipo de dado também é conhecido como "dado binário", sendo representado, em um algoritmo, como: verdadeiro ou falso. Em relação às variáveis, para Manzano e Oliveira (2008), podemos adotar a seguinte definição:

> Variável é tudo aquilo que é sujeito a variações, que é incerto, instável ou inconstante. E ao relacionarmos o termo com o contexto computacional, devemos ter em mente que o volume de informações a serem tratadas é imensurável, tendo em vista que não podemos definir valores e proporções, aspectos estes, que irão variar conforme o problema a ser solucionado. (MANZANO; OLIVEIRA, 2008, p. 25)

No contexto computacional, podemos entender uma variável como uma caixa vazia (variável de memória), que é colocada em uma estante com vários compartimentos (memória). Um exemplo prático pode ser de uma biblioteca, em que as prateleiras seriam a variável de memória e, as prateleiras, a memória na qual são inseridos os livros.

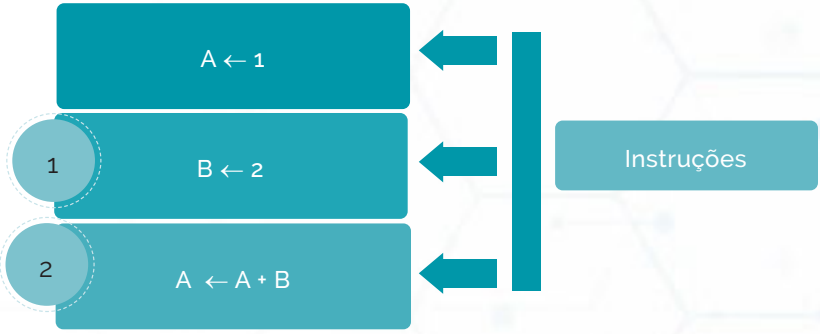
Figura 3 – Biblioteca



Fonte: Freepik

O algoritmo define quais dados serão armazenados em determinadas variáveis de memória, por quanto tempo e para que finalidade. Ele também estabelece que tipo de dado poderá ser armazenado em cada variável, bem como o nome que a identificará. Para entendermos melhor tudo isso, vamos visualizar o comportamento de algumas variáveis após estas serem processadas pelas instruções da Figura 4 (MORAIS; AZEVEDO, 2017a).

Figura 4 – Exemplo do comportamento de algumas variáveis após serem processadas pelas instruções



Fonte: Morais e Azevedo (2017a, p.18).

No trecho desse algoritmo, uma variável de nome "A" recebeu um dado numérico de valor "1". Em seguida, outra variável de nome "B" recebeu um outro dado, também numérico, de valor "2". Por fim, os dados contidos nas variáveis A e B foram somados e colocados na variável A, que teve seu valor original 1 substituído por um novo, 3. Ainda neste trecho de algoritmo, dizemos que A e B são variáveis, e os números 1 e 2 são constantes. Agora, vejamos algumas definições que, para atribuir uma constante ou o conteúdo de uma variável ou expressão a uma variável de memória, utiliza-se o operador "" na instrução. Além disso, que uma constante é um dado usado, normalmente, em uma expressão matemática, caractere ou lógica, que define um valor de equilíbrio que se mantém inalterado, independentemente das variáveis envolvidas. Vejamos alguns novos exemplos de variáveis e constantes envolvidas em outras operações e expressões.

```
NOME ← "JOÃO"
SOBRENOME ← "DA SILVA"
NOMECOMPLETO ← NOME & SOBRENOME
```

Dessa vez, uma variável de nome "NOME", recebeu uma constante alfanumérica igual a "JOÃO". Em seguida, outra variável de nome "SOBRENOME" recebeu outro dado, também alfanumérico, igual a "DA SILVA". Por fim, os dados contidos nas variáveis NOME e SOBRENOME foram concatenados e colocados em uma terceira variável, intitulada "NOMECOMPLETO". Depois do processamento das instruções de 1 a 3 acima, o conteúdo resultante da variável NOMECOMPLETO será "JOÃODA SILVA". Ao concatenarmos duas variáveis, como no exemplo acima, simplesmente, os conteúdos destes são justapostos um após o outro, da seguinte forma:

```
NOMECOMPLETO ← NOME & SOBRENOME
```

Os seguintes dados serão concatenados:

NOMECOMPLETO ← "JOÃO" & "DA SILVA"

O resultado da concatenação, portanto, é:

"JOÃODA SILVA"

Ou seja, sem o espaço em branco antes de "DA SILVA".

Operadores, Depuração de Algoritmos e VisuAlg

Acabamos de visualizar algumas instruções em dois exemplos de algoritmo. Vimos que, nessas instruções, existem alguns sinais que representam uma determinada operação, como a soma (A+B) e a concatenação (NOME&SOBRENOME). Foram utilizados, respectivamente, os seguintes operadores nessas instruções:

Quadro 1 – Exemplos de operadores

+ Operador Soma	Atua sobre variáveis e constantes numéricas
& Operador Concatenação	Atua sobre variáveis e constantes alfanuméricas.

Fonte: Morais e Azevedo (2017a, p.20).

A adoção de certos símbolos como "+" e "&" para servirem de operadores em uma instrução, faz parte da função sintática da linguagem de programação. A seguir, vamos apresentar outros operadores adotados pela linguagem algorítmica, segundo Manzano e Oliveira (2008).

Quadro 2 – Exemplos de operadores adotados pela linguagem algorítmica

- Operador Subtração	Atua sobre variáveis e constantes numéricas
* Operador Multiplicação	Atua sobre variáveis e constantes numéricas
/ Operador Divisão	Atua sobre variáveis e constantes numéricas
^ Operador Potenciação	Atua sobre variáveis e constantes numéricas

Fonte: Manzano e Oliveira (2008, p.23).

É importante considerar que o operador de potenciação "^" também pode ser representado por "**". Vejamos outros exemplos de instruções envolvendo operações com variáveis:

```
X ←300
Y ← 150
Z ← X/Y^2
```

Após serem executadas essas instruções, o valor da variável Z será igual a 4, uma vez que Z ← 300/150^2. Logo Z ← 2^2, ou seja, Z ←4. Analisando esse exemplo, caso após o processamento das instruções acrescentássemos o seguinte: CARRO ←Z & " POR 4" , resultaria como indefinido. O resultado de uma operação envolvendo variáveis de tipos diferentes é indefinido, ou seja, esse tipo de instrução provoca um erro de execução do algoritmo em tela. E esse é exatamente o caso, pois o conteúdo da variável Z é do tipo numérico, enquanto o operador utilizado na última instrução (& = concatenação) é um operador literal (alfanumérico).

Em relação à depuração de algoritmos, no exemplo utilizado no tópico anterior, vivenciamos uma situação de erro de programa. A partir de agora, situações como essa serão rotineiras para você. Todo programador lida com erros de programa todos os dias e em todas as horas, por isso, ao concluir seu algoritmo ou programa, não hesite em testá-lo (ou depurá-lo) (FORBELLONE, 2005).

DEFINIÇÃO:

Depuração é a técnica utilizada por programadores e outros profissionais para testar um programa ou processo repetitivo qualquer. Trata-se de um ato contínuo que se repete até que seja atingido o ZERO erro.

Erros de programação (ou de algoritmização) acontecem por vários motivos. Estes podem ser classificados nas seguintes categorias:

- Erro de sintaxe – ocorre quando uma instrução não é escrita de acordo com as regras gramaticais da linguagem, ou seja, conforme a sintaxe da instrução.
- Erro de dados – acontece toda vez que constantes ou variáveis de diferentes tipos são envolvidas em uma mesma operação, como vimos no exemplo anterior.
- Overflow – do inglês, esse termo significa sobrecarga, e ocorre toda vez que o limite de tamanho de uma variável é excedido. Cada linguagem de programação estabelece o limite de cada tipo de variável.
- Looping – também do inglês, podemos traduzir esse termo como repetição sem fim. Não é propriamente um erro de programação, mas de lógica de programação, pois, por uma situação não prevista pelo programador, as instruções ficam sendo processadas em círculos para sempre ou até o programa ser cancelado.
- Indefinição – ocorre toda vez que um cálculo resulta em um valor indefinido ou inexistente, como uma divisão por zero, por exemplo.

Sobre o VisuAlg, trata-se de um aplicativo para PC (computadores desktop ou notebooks) com o sistema operacional Windows instalado, cuja finalidade é simular a execução de um algoritmo escrito na língua portuguesa.

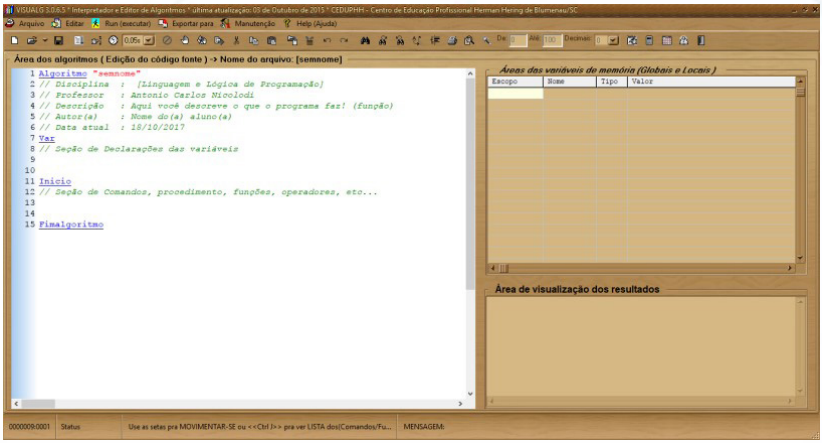
ACESSE:

Faça o download do VisuAlg no seu computador. Para acessar, clique aqui.

Na prática, o VisuAlg é um software interpretador de algoritmos escritos em pseudocódigos, sua interface é dialógica e extremamente intuitiva, mesmo para principiantes na área de programação, e sua tela

é dividida em três partes bem distintas: área dos algoritmos, área das variáveis de memória e área de visualização dos resultados..

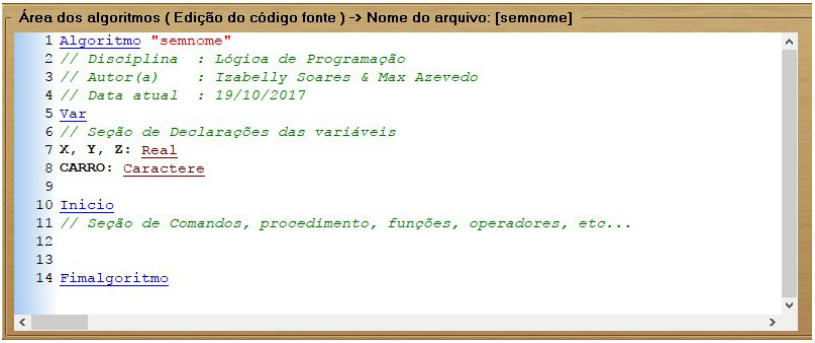
Figura 5 – Interface do VisuAlg versão 3.0



Fonte: Reprodução do software VisuAlg.

Com o VisuAlg é possível testar todos os algoritmos que iremos desenvolver ao longo desta disciplina. Vamos começar aprendendo como esse interpretador aceita a manipulação de variáveis e constantes. Para utilizar variáveis ao longo de um algoritmo VisuAlg, é necessário que você as declare logo no início do programa.

Figura 6 – Painel de edição e monitoramento do pseudocódigo em execução



Fonte: Reprodução do software VisuAlg.

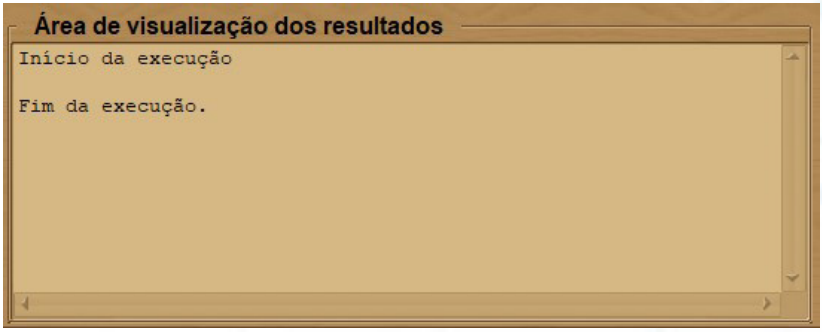
Perceba que as variáveis X, Y e Z foram declaradas como do tipo numérico e real; já a variável CARRO foi declarada como alfanumérica, que, no caso do VisuAlg, deve ser descrita como do tipo "caractere". Para executar esse algoritmo e testar o comportamento dessas variáveis, basta teclar <F9>, ou <F8> para executar o algoritmo passo a passo (linha a linha). Ao executar o algoritmo do começo ao fim, você deverá visualizar o quadro de variáveis.

Figura 7 – Painel de monitoramento das variáveis declaradas

Fonte: Reprodução do software VisuAlg.

Observe ainda que, ao executar esse código, o VisuAlg simula o que o usuário veria na tela de seu computador. Como o algoritmo não tinha comando de exibição de dados em tela, nada foi exibido, exceto as mensagens de início e término do algoritmo, como mostram os dois painéis de visualização de resultados.

Figura 8 – Área de visualização de resultados do VisuAlg



Fonte: Reprodução do software VisuAlg.

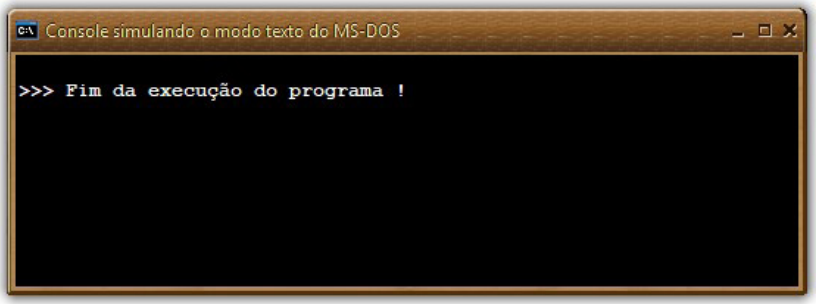
Além do painel de visualização de resultados, o VisuAlg ainda disponibiliza um simulador de console em interface textual, como em uma tela emulada do sistema operacional MS-DOS.

SAIBA MAIS:

Para se aprofundar no tema deste tópico, recomendamos o acesso ao vídeo a seguir. Para acessar, clique aqui.

Veja o resultado desse último algoritmo representado neste simulador:

Figura 9 – Área de visualização de resultados do VisuAlg



Fonte: Reprodução do software VisuAlg.

É importante salientar que as telas aqui apresentadas são meramente ilustrativas da versão 3. É recomendado sempre estar com as versões atualizadas, dessa maneira, alguns comandos podem ser alterados nas atualizações.

RESUMINDO:

E então? Gostou do que lhe mostramos? Aprendeu mesmo tudinho? Agora, só para termos certeza de que você realmente entendeu o tema de estudo deste Capítulo, vamos resumir tudo o que vimos. Você deve ter aprendido que um dos conceitos mais importantes no universo da programação de computadores é o da variável de memória. Sem as variáveis seria praticamente impossível submeter um dado a um processamento. Para processar um dado, é necessário guardá-lo em algum lugar, e a esse lugar que damos o nome de "variável de memória". Foram apresentadas algumas maneiras de manipular uma variável de memória em um algoritmo. E, por falar em algoritmo, é importante conhecer o método de representação no formato de pseudocódigo, utilizando a própria língua portuguesa como base para escrevermos todo e qualquer algoritmo.

Representação Gráfica e Textual de Algoritmos

OBJETIVO:

Ao término deste Capítulo você será capaz de utilizar expressões aritméticas envolvendo constantes e variáveis numéricas em soluções algorítmicas. Isso será fundamental para o exercício de sua profissão. E então? Motivado para desenvolver esta competência? Então vamos lá. Avante!.

Expressões Aritméticas

Chamamos de expressão aritmética a operação matemática que envolve dois ou mais operandos, podendo ser constantes e/ou variáveis. Vejamos, então, três exemplos de expressões matemáticas:

```
A ← (B + C / D)^2 + 140*B
N ← M*3 + N*2 - 1
X ← (X/Y + 1)^2
```

Note que em algumas expressões foram utilizados parênteses, esses elementos são, normalmente, utilizados para priorizar uma operação em detrimento de outras dentro de uma mesma expressão. Por exemplo: (B+C/D)^2+140*B resultaria em um dado completamente diferente não fossem os parênteses, como mostra a simulação a seguir. Imagine que B = 1, C = 4 e D = 2, mantendo-se os parênteses do primeiro exemplo , teremos o seguinte desenvolvimento:

```
A ← (1 + 4 / 2)^2 + 140*1
A ← (3)^2         + 140
A ← 9             + 140
A ← 149
```

Mas, como seria o resultado dessa expressão se tirássemos os parênteses?

```
A ← B + C / D^2 + 140*B
A ← 1 + 4 / 2^2 + 140*1
A ← 1 + 4 / 4     + 140
A ← 1 + 1         + 140
A ← 142
```

Com essas duas simulações, acabamos de recordar algo que aprendemos lá atrás, no ensino fundamental: a prioridade entre as operações, em outras palavras, em uma expressão envolvendo as quatro operações aritméticas, teremos sempre as seguintes prioridades, conforme apresentado no quadro 3.

Quadro 3 – Ordem de prioridade das operações aritméticas em uma expressão

Operação	Operador	Prioridade
Adição	+	3
Subtração	-	3
Multiplicação	*	2
Divisão	/	2
Exponenciação	^	1

Fonte: Morais e Azevedo (2017c, p.19).

Assim como as constantes e variáveis numéricas, as expressões aritméticas também podem ser classificadas em tipos distintos:

- Inteiras.
- Reais.

O tipo de uma expressão é dado pela classe do dado numérico resultante de sua execução, por exemplo: se uma expressão tem como resultado um número inteiro, ela será inteira; do contrário, teremos uma expressão do tipo real (ou fracionário) (MORAIS; AZEVEDO, 2017c).

IMPORTANTE:

Para que uma expressão seja inteira, todos os seus operandos devem ser inteiros; se pelo menos um desses operandos for uma constante ou variável do tipo real, a expressão será real.

Para comprovarmos o que acabamos de enunciar, imagine a seguinte expressão:

R← 10 + 100 + 399 + 4.059 + 1,5

R← 4.569,5

Observe: bastou somarmos o número 1,5 aos demais operandos para obtermos um resultado fracionário. A expressão aritmética tem utilidade na hora de programarmos um computador. Imagine que você precise criar um algoritmo que receba um valor salarial de alguém e apresente esse salário acrescido de um aumento de 50%. Veja a solução encontrada por um programador para esse problema:

```
ALGORITMO
LEIA SALARIO
NOVOSAL ← SALARIO + SALARIO * 0,5
ESCREVA NOVOSAL
FIM_ALGORITMO
```

Perceba que o algoritmo recebeu um salário digitado por um usuário qualquer, dentro de uma variável denominada "SALARIO". Em seguida, o programador atribuiu uma expressão matemática sobre a própria variável SALARIO, modificando o seu valor para o correspondente ao seu valor antigo, mais o percentual de 50% aplicado sobre ele mesmo. Esse percentual de 50% foi expresso na operação: SALARIO * 0,5. O valor 0,5 corresponde a 50 / 100, ou seja, cinquenta por cento, porém com

um pouco mais de conhecimento sobre aritmética. Esse programador poderia ter simplificado essa expressão da seguinte maneira:

```
ALGORITMO
LEIA SALARIO
NOVOSAL← SALARIO + SALARIO * 1,5
ESCREVA NOVOSAL
FIM_ALGORITMO
```

Note que alguma coisa mais a metade dela própria é o mesmo que multiplicá-la por uma vez e meia, logo, as duas expressões apresentadas como solução algorítmica do problema proposto resultam no mesmo valor.

SAIBA MAIS:

Para se aprofundar no tema dessa unidade, recomendamos o acesso ao menu "Help" do próprio VisuAlg, onde você poderá encontrar várias referências.

Vamos sofisticar ainda mais nosso algoritmo? E se, além do salário, ele permitisse que o usuário fornecesse o percentual de aumento a ser aplicado? Teríamos, então, o seguinte algoritmo:

```
ALGORITMO
LEIA SALARIO
LEIA PERCENTUAL
NOVOSAL ← SALARIO * (1 + PERCENTUAL)
ESCREVA NOVOSAL
FIM_ALGORITMO
```

Vamos utilizar o VisuAlg para simular esse último exemplo? Então, vamos declarar as variáveis SALARIO, PERCENTUAL e NOVOSAL como sendo numéricas e inteiras.

A lógica de programação é uma matéria comumente associada à matemática. Esse é um dos motivos pelo qual certas pessoas com pouca habilidade em matemática se afastam ou criam alguma resistência ao aprendizado desta disciplina. No entanto, conhecendo etapa por etapa, o aprendizado se torna mais fácil e intuitivo.

RESUMINDO:

E então? Gostou do que lhe mostramos? Aprendeu mesmo tudinho? Agora, só para termos certeza de que você realmente entendeu o tema de estudo deste Capítulo, vamos resumir tudo o que vimos. Você deve ter aprendido que a lógica de programação vai muito além de fazer contas de matemática. Ela tem mais a ver com a capacidade de abstração do que com a aritmética. Por isso é importante introduzir a matemática nos estudos na medida certa para resolvermos as questões mais básicas possíveis. Estamos falando das quatro operações: soma, subtração, multiplicação e divisão, além da potenciação e outras poucas funções matemáticas. É importante saber a fórmula certa para a sua aplicação nos exercícios e nas atividades de programação. Mais importante do que a matemática em si, é saber empregar os operadores certos nas variáveis corretas, de modo a resolver os problemas que podem aparecer no desenvolvimento dos projetos.

Operações e Expressões Alfanuméricas

OBJETIVO:

Ao término deste Capítulo você será capaz de utilizar expressões literais envolvendo constantes e variáveis alfanuméricas em soluções algoritmicas. Isso será fundamental para o exercício de sua profissão. E então? Motivado para desenvolver esta competência? Então vamos lá. Avante!.

Funções e Operações de Caracteres

Você sabe o que vem a ser uma função? Muitas vezes, quando desejamos submeter uma variável ou uma constante a um processamento, nem sempre encontramos um operador que resolva o problema. Por exemplo, se quisermos somar algo a uma variável, usamos o operador "+"; no entanto, se quisermos encontrar o valor inteiro correspondente a um número real, precisamos de uma função chamada INT. Vejamos como funciona a sintaxe dessa função (MORAIS; AZEVEDO, 2017c):

```
X ← 1,9
N ← Int ( X )
Escreva N
```

Essa função funciona como um pequeno algoritmo que recebe um parâmetro (ou argumento) de outro algoritmo, devolvendo-lhe um resultado processado daquele argumento.

DEFINIÇÃO:

Função é todo e qualquer algoritmo externo que pode ser chamado a partir de uma instrução, devolvendo um resultado processado a partir de um ou mais argumentos transmitidos pelo algoritmo que a chamou.

Uma função pode ser interna ou externa à linguagem de programação:

1. Funções internas: são aquelas disponibilizadas pela própria linguagem, assim como suas instruções e semântica.
2. Funções externas: são aquelas criadas por programadores e disponibilizadas para reuso em outros programas.

Ao contrário das operações aritméticas básicas, que são quatro (e mais a potenciação), as operações de caracteres se resumem a apenas uma: a concatenação. As linguagens de programação costumam utilizar dois operadores para simbolizar uma concatenação:

+ ou &

O VisuAlg, por exemplo, utiliza o mesmo operador da adição (+) para a concatenação de cadeias de caracteres (variáveis ou constantes). Por exemplo, se quisermos adicionar o pronome de tratamento "Sr." a uma variável que irá receber o nome de uma pessoa do teclado do computador, teremos o seguinte algoritmo (RODRIGUES, 2016):

```
ALGORITMO
LEIA NOME
NOME ← "Sr." + NOME
ESCREVA NOME
FIM_ALGORITMO
```

Para visualizar e compreender melhor o efeito da concatenação envolvendo a variável NOME, execute este algoritmo passo a passo, teclando <F8> em vez de <F9>.

PASSO 1: declaração da variável NOME (RODRIGUES, 2016).

PASSO 2: digitação do dado na variável NOME.

PASSO 3: concatenação da constante "Sr." com a variável NOME.

PASSO 4: exibição do dado contido na variável NOME.

Existe mais de uma dezena de funções do tipo caractere nas linguagens de programação (RODRIGUES, 2016). Elas têm a função de permitir a manipulação dos dados no interior de variáveis, ou a combinação desses dados com os de outras variáveis. Imagine que seja necessário identificar a posição de determinada letra em um nome, ou mesmo conhecer o comprimento dessa cadeia de caracteres. Para cada necessidade dessas existe uma função, conforme serão apresentados a seguir. Caracpnum(), Caracpnum (PALAVRA): retorna o número inteiro ou real equivalente à cadeia de caracteres ou expressão alfanumérica informada como argumento da função, no caso, PALAVRA. Analise o exemplo a seguir:

```
Var
PALAVRA: caractere
RESULTADO: inteiro
PALAVRA ← "1000"
RESULTADO ← Caracpnum(PALAVRA) + 2000
ESCREVA RESULTADO
```

No exemplo acima o valor numérico da variável RESULTADO será 3000. Caso não fosse a função Caracpnum() não haveria como somar a string NUMERO com o número 2000. Já Numpcarac(), Numpcarac (NUMERO): retorna uma cadeia de caracteres contendo a representação alfanumérica de NUMERO. Essa função faz exatamente o contrário da Caracpnum(), ou seja, converte um NUMERO em uma string (MORAIS; AZEVEDO, 2017c).

```
Var
NUMERO: inteiro
RESULTADO: caractere
NUMERO ← 1000
RESULTADO ← Numpcarac(NUMERO) + " reais"
ESCREVA RESULTADO
```

No exemplo acima a cadeia de caracteres gerada na variável RESULTADO será "1000 reais". Se não fosse a função Numpcarac(), não haveria como concatenar o número contido na variável NUMERO com a

constante "reais". E Compr(), Compr(FRASE): retorna um valor numérico inteiro correspondente ao tamanho da variável, constante ou expressão FRASE. Em outras palavras, essa função serve para contar quantos caracteres existem na expressão, constante ou variável passada como argumento da função, por exemplo:

```
Var
FRASE: caractere
RESULTADO: inteiro
FRASE ← "Há mais brilhos em nossos olhos que estrelas no céu"
RESULTADO ←  Compr(FRASE)
ESCREVA RESULTADO
```

No exemplo acima, o valor gerado na variável RESULTADO será 51, ou seja, existirão 51 caracteres dentro da variável RESULTADO após a execução desse algoritmo. Em relação à Cópia(), Copia(FRASE, INICIO, TAMANHO): retorna uma cadeia de caracteres contendo uma cópia parcial de FRASE, a partir do caractere situado na posição INICIO, incluindo TAMANHO de caracteres a partir daquela posição. Os caracteres são numerados da esquerda para à direita, começando de 1.

```
Var
FRASE: caractere
RESULTADO: caractere
FRASE ← "Há mais brilhos em nossos olhos que estrelas no céu"
RESULTADO  Copia(FRASE, 9, 7)
ESCREVA RESULTADO
```

No exemplo anterior, a cadeia de caracteres gerada na variável RESULTADO será "brilho", após a execução deste algoritmo.

SAIBA MAIS:

Para conhecer mais funções internas oferecidas pelo VisuAlg, veja o link a seguir. Para acessar, clique aqui.

O tipo de uma expressão é dado pelo tipo do dado numérico resultante de sua execução, por exemplo: se uma expressão tem como resultado um número inteiro, ela será inteira, do contrário, teremos uma expressão do tipo real (ou fracionário).

RESUMINDO:

E então? Gostou do que lhe mostramos? Aprendeu mesmo tudinho? Agora, só para termos certeza de que você realmente entendeu o tema de estudo deste Capítulo, vamos resumir tudo o que vimos. Você deve ter aprendido que nem só de matemática vive a lógica de programação. Na realidade, uma grande variedade de problemas informacionais envolve operações conhecidas como string, ou alfanuméricas. As cadeias de caracteres estão contidas na maior parte dos dados que são rotineiramente processados pelos computadores. Assim, é importante o conhecimento das operações de varredura e concatenação das cadeias de caracteres, fundamentais na formatação de exibição e impressão de dados. No entanto, antes de ir direto para o conhecimento sobre as operações string, precisamos conhecer conceito de função. Isso porque a maioria das operações envolvendo variáveis e constantes alfanuméricas são obtidas por meio de funções, e não por operações.

Estruturas Condicionais SE

OBJETIVO:

Ao término deste Capítulo você será capaz de aplicar estruturas condicionais SE em soluções algorítmicas. Isso será fundamental para o exercício de sua profissão. E então? Motivado para desenvolver esta competência? Então vamos lá. Avante!.

Operações Condicionais

Você sabe o que vem a ser uma operação condicional? Assim como uma operação aritmética submete seus operandos numéricos a um processamento matemático, por meio de operadores, o mesmo ocorre com as operações string: as operações condicionais se comportam mais ou menos da mesma forma. A diferença é que o resultado de uma operação condicional é sempre verdadeiro ou falso, por exemplo, em um sinal de pedestre a cor verde pode indicar como verdadeiro e a cor vermelha como falso. Outra diferença conceitual entre a operação condicional e as demais está nas instruções que as suportam.

Figura 10 – O resultado de uma operação condicional é sempre verdadeiro ou falso, e pode ser exemplificada com o sinal de trânsito relacionados aos pedestres

Fonte: Freepik

Normalmente, uma operação aritmética é utilizada em uma instrução de atribuição, ou seja, o resultado de uma expressão matemática é, geralmente, atribuído a uma variável de memória ou, ainda, pode ser utilizada diretamente em uma instrução de saída, como no comando ESCREVA, por exemplo. Uma operação condicional também pode ser atribuída a uma variável, porém o mais comum é que elas sejam utilizadas em comandos ou estruturas condicionais, como o comando SE, por exemplo. Para compreendermos melhor, imagine se fizéssemos um algoritmo que lesse dois números inteiros e, independentemente da ordem em que os digitássemos, eles sempre fossem mostrados em ordem crescente (do menor para o maior). Para exemplificar melhor vamos elaborar este algoritmo:

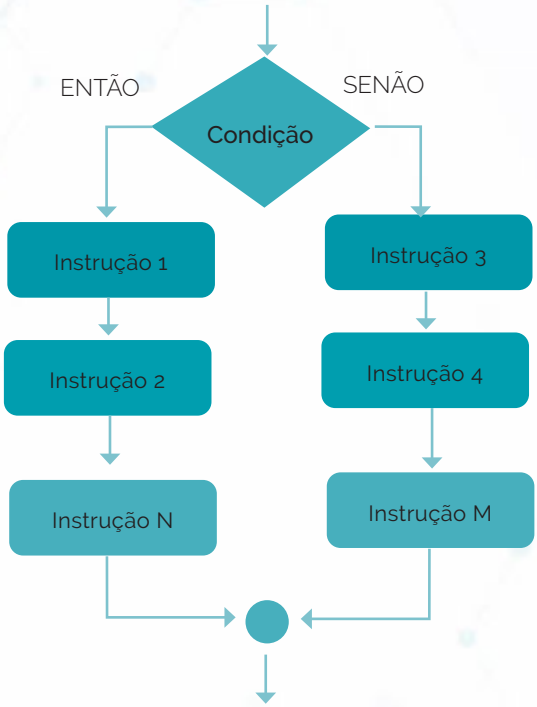
```
ALGORITMO
VAR
A, B: INTEIRO
LEIA (A, B)
SE A > B ENTÃO
        ESCREVA (B, A)
SENÃO
        ESCREVA (A, B)
FIMSE
FIMALGORITMO
```

Note que utilizamos uma estrutura condicional conhecida como SE. Essa estrutura tem a seguinte sintaxe:

```
SE <expressão condicional> ENTÃO
        <instrução-1>
        <instrução-2>
        ...
        <instrução-N>
SENÃO
        <instrução-3>
        <instrução-4>
        ...
        <instrução-M>
FIMSE
```

No caso do exemplo anteriormente ilustrado, a <expressão condicional> utilizada foi "A < B", ou seja, tivemos uma operação que submeteu duas variáveis a um operador relacional, no caso, o operador ">".

Figura 11 – Fluxograma da estrutura SE



Fonte: Morais e Azevedo (2017b).

Operadores relacionais comparam o valor de dois operandos, que podem ser expressões aritméticas ou caracteres, como duas variáveis entre si, uma constante com variável ou vice-versa.

SAIBA MAIS:

Para conhecer mais operadores oferecidos pelo VisuAlg, veja o link seguir. Para acessar, clique aqui.

São esses os operadores que podem ser usados em estruturas SE, entre outros.

Quadro 4 – Operadores relacionais

Operador	Significado	Exemplo de operação
>	Maior que	A > B
<	Menor que	A < B
>=	Maior ou igual a	A >= B
<=	Menor ou igual a	A <= B
=	Igual a	A = B
<>	Diferente de	A <> B

Fonte: Morais e Azevedo (2017b).

Já os operadores lógicos são aqueles que atuam sobre expressões condicionais, conectando-as ou invertendo seus sinais.

Quadro 5 – Operadores lógicos

Operador	Significado	Exemplo	É o mesmo que
NÃO	Negação	NÃO A > B	A <- B.
OU	Disjunção	A < B OU B > C	Verdadeiro se quaisquer das expressões forem verdadeiras, inclusive ambas.
E	Conjunção	A < B E B > C	Verdadeiro somente se ambas as expressões forem verdadeiras.
XOU	Exclusão	A < B XOU B > C	Verdadeiro se qualquer uma for verdadeira, mas falsa se ambas forem verdadeiras.

Fonte: Morais e Azevedo (2017b).

Vamos ver como ficaria o algoritmo-exemplo no interpretador do VisuAlg.

Figura 12 – Colocar dois números em ordem crescente

Área dos algoritmos (Edição de código fonte)

```
1 Algoritmo "Aula 08 – Exemplo 1"
2 Var
3 A, B: Inteiro
4 Início
5 Leia (A, B)
6 SE (A> B) então
7 Escreva (B, A)
8 Senão
9 Escreva (A, B)
10 FIMSE
11 Fimalgoritmo
```

Fonte: Adaptado da Reprodução do software VisuAlg.

Na lógica de programação é sempre importante executar testes. Dessa maneira, o uso de aplicativos como o VisuAlg é recomendado.

RESUMINDO:

E então? Gostou do que lhe mostramos? Aprendeu mesmo tudinho? Agora, só para termos certeza de que você realmente entendeu o tema de estudo deste Capítulo, vamos resumir tudo o que vimos. Você deve ter aprendido que é importante saber como manipular variáveis de memória, quer sejam numéricas ou caracteres. Para tanto, precisamos entender como se comportam as variáveis lógicas e suas expressões condicionais. Entender bem isso é preponderante para a programação de soluções computacionais, pois tomar decisões é algo que todo programador estará sempre fazendo ao longo da codificação de seus programas.

REFERÊNCIAS

FORBELLONE, A. L. **Lógica de programação**: a construção de algoritmos e estrutura de dados. São Paulo: Pearson, 2005.

MANZANO, J. A.; OLIVEIRA, J. F. **Algoritmos**: lógica para desenvolvimento de programação de computadores. 21. ed. São Paulo: Érica, 2008.

MORAIS, I.S.; AZEVEDO M. **Lógica de programação**: constantes e variáveis de memória. Recife: Unissau, 2017a.

MORAIS, I.S.; AZEVEDO M. **Lógica de programação**: estruturas condicionais SE. Recife: Unissau, 2017b.

MORAIS, I.S.; AZEVEDO M. **Lógica de programação**: operações e expressões alfanuméricas. Recife: Unissau, 2017c.

RODRIGUES, A. **Manual do VisuAlg.** Iguatu: IFCE, 2016.

SEBESTA, R. W. **Conceitos de linguagens de programação.** São Paulo: Bookman, 2011.