

Algoritmos e Lógica de Programação

Juliana Schiavetto Dauricio

© 2015 por Editora e Distribuidora Educacional S.A

Todos os direitos reservados. Nenhuma parte desta publicação poderá ser reproduzida ou transmitida de qualquer modo ou por qualquer outro meio, eletrônico ou mecânico, incluindo fotocópia, gravação ou qualquer outro tipo de sistema de armazenamento e transmissão de informação, sem prévia autorização, por escrito, da Editora e Distribuidora Educacional S.A.

Presidente: Rodrigo Galindo

Vice-Presidente Acadêmico de Graduação: Rui Fava

Gerente Sênior de Editoração e Disponibilização de Material Didático:

Emanuel Santana

Gerente de Revisão: Cristiane Lisandra Danna

Coordenação de Produção: André Augusto de Andrade Ramos

Coordenação de Disponibilização: Daniel Roggeri Rosa

Editoração e Diagramação: eGTB Editora

Dados Internacionais de Catalogação na Publicação (CIP)

Dauricio, Juliana Schiavetto

D243a Algoritmos e lógica de programação / Juliana Schiavetto

Dauricio. – Londrina : Editora e Distribuidora Educacional S.A., 2015.

260 p.

ISBN 978-85-8482-227-0

1. Programação (Computadores). 2. Algoritmos.
3. Estruturas de dados (Computação). I. Título.

CDD 005.1

2015

Editora e Distribuidora Educacional S.A

Avenida Paris, 675 – Parque Residencial João Piza

CEP: 86041-100 – Londrina – PR

e-mail: editora.educacional@kroton.com.br

Homepage: <http://www.kroton.com.br/>

Sumário

| | |
|--|----------------|
| Unidade 1 Algoritmos e Programação Contextos e Práticas | 7 |
| Seção 1.1 - Histórico e definição de algoritmos: perspectivas de linguagem | 11 |
| Seção 1.2 - Tipos de dados e expressões: literais, lógicas e aritméticas | 25 |
| Seção 1.3 - Representação de algoritmos e o ambiente de programação | 39 |
| Seção 1.4 - Declaração de variáveis e constantes | 51 |
| Unidade 2 Estruturas de Decisão ou Seleção | 67 |
| Seção 2.1 - Instruções primitivas: entrada de dados, atribuição e saída | 71 |
| Seção 2.2 - Estrutura condicional simples | 87 |
| Seção 2.3 - Estrutura condicional composta | 103 |
| Seção 2.4 - Estrutura condicional sequencial e encadeada | 115 |
| Unidade 3 Estruturas de seleção (case) e repetição | 131 |
| Seção 3.1 - Estrutura de múltipla escolha (CASE) | 133 |
| Seção 3.2 - Repetição condicional com teste no início | 147 |
| Seção 3.3 - Repetição condicional com teste no final | 161 |
| Seção 3.4 - Repetição controlada por variável | 175 |
| Unidade 4 Vetores e Matrizes | 193 |
| Seção 4.1 - Aplicações utilizando vetores e matrizes | 195 |
| Seção 4.2 - Operações sobre vetores e matrizes | 211 |
| Seção 4.3 - Os vetores como estruturas de dados | 225 |
| Seção 4.4 - As matrizes como estruturas de dados | 241 |

Palavras do autor

Olá, aluno, seja bem-vindo.

Nesta unidade de ensino você conhecerá o que é algoritmo e a sua relação com as linguagens de programação, assim como o modo como esses auxiliam no desenvolvimento do raciocínio lógico. A partir de agora, você será apresentado aos conceitos e conteúdos componentes deste tema. Iniciamos com uma breve definição do que é algoritmo, seguido de um resgate de como ocorreu a evolução dos computadores e a importância dos algoritmos para os sistemas computacionais. Você também conhecerá como são os ambientes de programação, as formas de representação e como é que se desenvolve um algoritmo. Saber distinguir as expressões literais, lógicas e aritméticas também é importante no âmbito computacional e faz parte dos nossos estudos nesta primeira etapa. Além disso, conheceremos os tipos de dados, variáveis e constantes, no decorrer da leitura do seu material.

O foco da Unidade 2 são as operações e comandos que permitem a entrada e saída de dados, o ponto de partida para se aprender a manipular os dados e informações nos sistemas computacionais. Associados a esses conceitos, estão a atribuição de valores e a forma de realizar os procedimentos para a declaração das variáveis e constantes. Finalmente, iniciam-se os estudos em estruturas de decisão e seleção como “if/else” (se/então/senão e CASE), respectivamente. Com elas, você pode compreender a lógica envolvida nas estruturas de programação condicionais simples, compostas, encadeadas e também como é possível, em programação, oferecer opções ao usuário através da estrutura de seleção CASE ou CASO.

A Unidade 3 é responsável por promover a imersão em situações reais para aplicar os conhecimentos mais específicos, tais como: *Switch-CASE*, que permite ao usuário escolher dentre as opções predeterminadas e, ainda, estruturas de repetição como “for” (para), “do” (fazer) e “do/while” (faça enquanto). Na Unidade 4, a atenção está sobre o entendimento dos vetores e matrizes, sua estrutura, eficiência e aplicações.

Após todos os conceitos, o material de estudos ainda contribui para que você possa treinar os conhecimentos adquiridos, realizando exercícios orientados, que visam aproximar a situação real e os problemas enfrentados no dia a dia do desenvolvimento de *software*. Desejamos a você, desde já, bons estudos e dedicação para a conclusão desta etapa.

ALGORITMOS E PROGRAMAÇÃO: CONTEXTOS E PRÁTICAS

Convite ao estudo

Para iniciar os estudos em algoritmos e programação, é interessante refletir sobre o que são e quais algoritmos influenciam diretamente nas transações comerciais que se estabelecem na atualidade e que detêm uma relevância no desenvolvimento de soluções práticas e inovadoras. Esses trazem possibilidades de empreendimentos entre os mais diversos segmentos do mercado. A partir de tais informações e compreendendo a relevância do tema, conheça a competência de fundamento de área que se pretende desenvolver com este estudo e os objetivos específicos desta unidade.

Competência de fundamento de área

- Conhecer os princípios e conceitos que envolvem o aprendizado em construção de algoritmos e programação e a sua importância para o universo do desenvolvimento de sistemas.

Objetivos específicos

- Conhecer o que são, como se aplicam e a quem se destina a elaboração dos algoritmos;
- Conhecer como são desenvolvidas as estruturas computacionais com expressões literais, lógicas e aritméticas, além dos tipos de dados;
- Conhecer o ambiente de programação e como se dá o raciocínio lógico computacional, bem como os tipos e formas de representação de algoritmos através da elaboração do fluxograma, do pseudocódigo em VisuAlg e a sua aplicação em uma linguagem de programação conhecida como C.

- Conhecer e saber identificar o que são variáveis e constantes e como se faz a declaração de variáveis.

Importante trazer as definições pertinentes ao tema: algoritmo pode ser definido como **um conjunto de processos ou ações, que seguem uma sequência lógica, para executar uma tarefa**. Por exemplo, baixar um aplicativo em seu celular requer que você execute um passo a passo para realizar aquela instalação e o configure para poder utilizá-lo. A esse passo a passo dá-se o nome de algoritmo.

Para desenvolver algumas das atividades aqui apresentadas, você poderá considerar, entre outros, o cenário que envolve os comerciantes do Litoral Sul do país. Suponha que eles estejam se organizando em cooperativa para angariar recursos e melhorar as transações comerciais desse mercado e contrataram uma consultoria para desenvolver um aplicativo; você faz parte dessa consultoria. O grande desafio dos comerciantes e da empresa contratada para o desenvolvimento do aplicativo é encontrar uma solução que atenda à necessidade de divulgação, organização das informações, facilidade de uso e acesso para os seus clientes os encontrarem, agendarem ou mesmo tomarem conhecimento daquele determinado estabelecimento de hotelaria ou gastronomia. Um ponto de atenção é que será preciso realizar todos os processos, desde a determinação das ações do sistema e tarefas até a escolha das ferramentas de análise e desenvolvimento do aplicativo.

Uma das propostas da empresa é disponibilizar um protótipo do aplicativo para celular que informe aos usuários quais são os serviços de hotelaria e gastronomia disponíveis naquela região. A partir dessa situação, os comerciantes esperam obter soluções que apresentem uma forma de contabilizar o índice de satisfação do usuário quanto à facilidade de navegação e uso do aplicativo. Para tal, após realizar a consulta, o usuário é direcionado a um painel com os ícones que representam o seu índice de satisfação: insatisfeito, satisfeito ou plenamente satisfeito. O prazo para desenvolvimento desse protótipo é de seis meses.

O primeiro passo é compreender o que realmente será possível fazer e estipular as ações do sistema. Para tal, você precisa entender que é necessário identificar quais são as suas variáveis, bem como as estruturas que precisarão ser desenvolvidas para trabalhar com os dados, armazená-los, atribuir, excluir e alterar valores, verificar se há variáveis ou se serão declaradas como constantes, entre outros conceitos que precisam ser aplicados. Com o estudo de algoritmos e programação, a identificação desses passos será facilitada e você ainda poderá compreender as formas de representação, além de sua definição.

Atente-se também aos tipos de dados, às expressões literais, aritméticas e lógicas que serão necessárias para realizar um cálculo específico, como a média dos acessos, por exemplo, ou ainda, se deverá executar algum teste para informar os resultados aos comerciantes. Estabeleça um processo de verificação e controle de como possibilitar que o sistema contabilize os acessos, realize o cálculo da média através da indicação de sua expressão aritmética, indique uma forma de registrar essas informações e enviá-las aos interessados.

Muito importante, nesta primeira entrega, é que seja especificado o fluxograma dos processos que algoritmo deverá contemplar e os passos a seguir para o seu desenvolvimento. Essa é uma tarefa que requer a leitura dos materiais didáticos e dos artigos e vídeos que complementam os seus estudos. Fica a sugestão do desenvolvimento das estruturas do algoritmo em pseudocódigo, o que permite que você identifique os processos e a necessidade de alterações, antes do desenvolvimento efetivo do aplicativo. Encontre a lógica necessária para otimizar o processamento e uso dos recursos. Então, bons estudos e práticas a você.

Seção 1.1

Histórico e definição de algoritmos: perspectivas de linguagem

Diálogo aberto

Você já deve ter observado que todas as ações que realizamos obedecem a uma sequência, e que esta precisa ser executada para que se consiga atingir o objetivo predeterminado. Por exemplo, quando é necessário trocar o pneu do carro, o que você observa é que existem passos a cumprir. Veja o primeiro exemplo de algoritmo, neste caso em linguagem natural, que estabelece o passo a passo para trocar um pneu de carro:

1. Desparafusar a roda.
2. Suspende o carro com o macaco.
3. Retirar a roda com o pneu.
4. Colocar o estepe.
5. Abaixar o carro.
6. Parafusar a roda.



Refleta

"Para programarmos em um computador, devemos conhecer e entender o que são os algoritmos, e como utilizá-los para determinar a sequência de passos necessários para resolvermos determinados problemas ou, em outras palavras, encontrarmos a solução, ou a melhor solução, para a implementação em uma linguagem de programação." (PIVA JR. et al., 2012, p. 3)

Você sabe o que é uma linguagem de programação?

As linguagens de programação visam possibilitar a inserção dos passos ou ações que o *software* deverá executar, através de uma plataforma que permita, desde que respeite a sintaxe de cada uma, expressar a sequência lógica determinada no algoritmo, ou a sua construção propriamente dita.

Por exemplo, para passar um algoritmo de linguagem natural, que é a forma como falamos como precisa ser executada uma tarefa, para uma linguagem de programação

estruturada, como "C", ou mesmo orientada a objetos, como "Java" ou "C#", é preciso, além de estabelecer o passo a passo, seguir a forma como essa linguagem interpretará essas instruções de maneira que sejam compreendidas pelo computador.

Este, por sua vez, utilizará os processos de compilação (tradução e interpretação para a linguagem de máquina), ou seja, codifica e decodifica para permitir que o resultado seja visualizado em sua tela com o respectivo processamento da informação que foi solicitada.

Em algoritmos é comum utilizar os *softwares* VisuAlg (pseudocódigo) ou o Scilab (resolução principalmente de problemas matemáticos) para a execução dessas tarefas.

Sendo assim, a partir de agora é preciso definir, seja em linguagem natural, em pseudocódigo ou fluxograma, um algoritmo que deixe clara a sequência lógica que terá de ser cumprida para a solução do problema que os comerciantes do Litoral Sul têm que resolver.

Siga em frente.

Não pode faltar

A fim de compreender esse processo de descoberta e desenvolvimento dos algoritmos, vamos recapitular algumas informações, tais como: qual é a organização básica do computador; o que é linguagem de máquina; como ocorre a comunicação entre elas; o código ASCII; o que são os programas de computadores; como é a lógica de programação e a forma como acontece a resolução de problemas pelo computador, ou seja, como se dá o processamento da informação. Após compreender esses conceitos, você já poderá avançar em seus estudos sobre algoritmos e programação. Você se recorda de como é a organização dos computadores?

Estes são divididos em quatro unidades básicas: unidade de entrada, unidade de saída, unidade de processamento central e memória.



Assimile

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

Já o código ASCII (American Standard Code for Information Interchange ou Código Padrão Americano para Intercâmbio de Informações) é o responsável por converter os

caracteres para a sua respectiva combinação binária. Cada caractere é composto por oito bits (8 bits correspondem a 1 byte). Essa combinação binária se chama linguagem de máquina, pois possui apenas dois símbolos: 0 e 1, conhecidos por bit (binary digit).



Pesquise mais

Acesse também o *link* <<http://www.unit-conversion.info/texttools/convert-text-to-binary/#data>> (acesso em: 18 dez. 2015), que permite a conversão de uma entrada de dado (texto ou número) em binário.

Os computadores precisam de *softwares* que auxiliem no desenvolvimento de atividades, sejam elas pessoais ou profissionais. Existem *softwares* para realizar as mais variadas tarefas. O desenvolvimento de um *software* contempla algumas fases, como análise, quando se entende qual o produto de *software* que será gerado; o projeto, em que há as especificações de cada uma das tarefas que ele executará; a implementação, que será realizada com uma linguagem de programação; e os testes, que são realizados a fim de conferir se o *software* atende aos processos solicitados. Para que sejam passíveis de implementação, essas tarefas obedecem a uma sequência lógica chamada algoritmo.

Tal sequência facilita a transposição das ações que o *software* precisará executar, mas desenvolvidas de acordo com a sintaxe da linguagem de programação que se escolheu, seja orientada a objetos ou estruturada.



Refleta

“Como a comunicação entre as unidades do computador teria que ser obtida através de fenômenos físicos, os cientistas que conceberam os computadores atuais estabeleceram dois símbolos básicos para a linguagem. Esta quantidade de símbolos foi escolhida pelo fato de que através de fenômenos físicos é muito fácil obter dois estados distintos e não confundíveis, como passar corrente elétrica/não passar corrente elétrica, estar magnetizado/não estar magnetizado, etc., podendo cada um destes estados ser um dos símbolos.” (EVARISTO, 2002, p. 6)

Os algoritmos possuem cinco propriedades que os caracterizam. São elas (KNUTH, 1973 apud PIVA JR., 2012):

- Finitude: indica que o algoritmo deve encerrar após um número finito de execuções.
- Definição: evita definições que gerem ambiguidade.
- Entrada: atribui valores ao algoritmo, a partir de especificações de variáveis e funções antes de sua iniciação.

- Saída: representa os valores após o processamento, ou seja, a execução das ações.
- Eficácia: todas as ações que o algoritmo realiza precisam ser executáveis em um limite de tempo predeterminado, finito, mesmo que seja sem o auxílio do computador.

O esquema a seguir representa o mecanismo de funcionamento de um algoritmo.



Fonte: Adaptado de Piva Jr, 2012, p. 7.



Pesquise mais

Para auxiliá-lo, sugerimos um *link* em que poderá conhecer os mais diversos tipos de algoritmos e testar a sua eficiência. Disponível em: <<http://nicholasandre.com.br/sorting/>>. Acesso em: 02 mar. 2015.

Antes de aprender a utilizar uma linguagem de programação específica, é importante que você compreenda que programar, de acordo com Ziviane (2007, p. 1), é “basicamente estruturar dados e construir algoritmos”. A partir da análise de um determinado problema, você precisa, então, implementar o algoritmo que foi desenvolvido. Há dois tipos de problemas que envolvem os algoritmos: o primeiro trata da análise de um algoritmo único, parte a parte, e investiga a quantidade de execuções de um determinado processo, buscando o seu aprimoramento para poupar recursos da máquina; o outro é a análise de uma classe de algoritmos, esse estudo implica na análise de vários algoritmos a fim de identificar um que seja viável e compatível para apresentar como sendo a solução de um determinado problema.

O custo de um algoritmo também pode ser mensurado a partir do princípio da análise de sua complexidade, chamada de “f”, com relação a um algoritmo “n”, ou seja, $f(n)$ representa a função que mede o tempo gasto por um algoritmo para executar um dado problema de tamanho n . A esta função “f” dá-se o nome de complexidade de tempo do algoritmo. É importante observar que $f(n)$ também mede a quantidade de recursos de memória utilizada para a sua execução, é chamada de função de complexidade de espaço (ZIVIANE, 2007), ou seja, fatores como tempo e espaço devem ser levados em consideração quando se trata da eficiência de um algoritmo.

Agora veja um exemplo de como se dá a lógica referente à construção de algoritmos. Eles representam a realização das atividades seguindo passos, etapas. O algoritmo a seguir trabalha com o problema das Torres de Hanói.

Observe que há três hastes, serão aqui chamadas respectivamente de A, B e C. A primeira haste (A) tem anéis com diâmetros diferentes e que estão organizados em ordem decrescente. O desafio está em transferir os anéis da haste A para a B. Para resolver essa questão, você precisa respeitar algumas regras:

- a. Não pode mover o disco por mais de uma haste por vez, por jogada;
- b. É preciso respeitar a ordem decrescente dos diâmetros, ou seja, do menor para o maior.

Figura 1 | Torres de Hanói



Fonte: Criado pelo autor, adaptado de Ziviane, 2007, p. 6.

Confira abaixo uma das possíveis formas de se resolver esse problema, repetindo quantas vezes for necessário a sequência, de acordo com a quantidade de anéis existentes na torre.

Exemplo 1: Algoritmo Torres de Hanói com três anéis

Início

1. Mover o primeiro anel de A para B;
2. Mover o segundo anel de A para C;
3. Mover o anel de B para C;
4. Mover o terceiro anel de A para B;
5. Mover o anel de C para A;
6. Mover o anel de C para B;
7. Mover o anel de A para B.

Fim

**Assimile**

Conheça e tente implementar o pseudocódigo para o cálculo da quantidade de movimentos necessários para resolver o problema da Torre de Hanói:

Algoritmo: "Torre de Hanoi"

var

discos: inteiro

mov: real

inicio

// Seção de Comandos

escreva (" Informe a quantidade de discos: ")

leia (discos)

se (discos \geq 3) entao

 mov \leftarrow (2^{discos}-1)

 escreval (" O numero de movimentos será: ",mov)

senao

 escreva (" O numero de discos é insuficiente")

fimse

fimalgoritmo

Note que o algoritmo está escrito em linguagem natural, porém, é necessário que seja formalizado de acordo com um conjunto de regras que chamamos de sintaxe. Além desta, há as regras semânticas, ou seja, de lógica. Através de comandos, será possível cumprir o conjunto de regras que o computador interpretará.

A partir do uso dos comandos, será possível escrever as expressões de forma que o computador consiga interpretar essas informações. Essas expressões, que podem ser literais, aritméticas ou lógicas, realizam operações com os dados que são atribuídos. Os comandos recebem o nome de estruturas de programação e podem ser sequenciais, de decisão ou de repetição. Quando você for desenvolver um algoritmo, lembre-se de que, além das estruturas determinadas, também devem ser considerados os tipos de dados, que são as categorias que permitem a aplicação de um conjunto de comando que a máquina irá interpretar e traduzir para valores binários.

Para manipular os dados, é preciso usar as variáveis e os valores constantes. Há também a atribuição de valores para essas variáveis, que determinam quais são os operadores, as funções e os procedimentos necessários para a solução do problema.

Abaixo, um outro exemplo de algoritmo, porém, em linguagem não computacional, que representa uma tarefa do cotidiano: realizar a troca de uma lâmpada:

Início

- a. Verificar se o interruptor está desligado.;
- b. Pegar uma escada;
- c. Posicionar a escada no local;
- d. Subir a escada;
- e. Retirar a lâmpada queimada;
- f. Colocar a lâmpada nova;
- g. Descer da escada;
- h. Acender a lâmpada no interruptor;
- i. Se a lâmpada não acender, então:
- j. Retirar novamente a lâmpada queimada;
- k. Trocar por uma lâmpada nova.
- l. Senão:
- m. Descartar a lâmpada queimada;
- n. Guardar a escada;
- o. Encerrar a tarefa.

Fim



Refleta

Lógica é uma área da Matemática cujo objetivo é investigar a veracidade de suas proposições. (SOUZA et al., 2011, p. 19)

Agora, observe o exemplo que busca resolver uma situação real no ambiente computacional. Suponha que você precise desenvolver um algoritmo, representado pelo seu pseudocódigo e também por um fluxograma. Ao invés de utilizar a linguagem natural, será preciso iniciar o aprendizado de acordo com uma sintaxe e a lógica computacional. O problema em questão é: algoritmo para calcular a área de um triângulo.



Exemplificando

Exemplo em linguagem natural de um algoritmo para calcular a área de um triângulo:

Início

1. Solicitar ao usuário que digite os valores da base (b) e da altura (h).
2. Calcular a área (A) com a fórmula: $A = (b \cdot h) / 2$.
3. Exibir o valor da área (A).

Fim

Pseudocódigo

(Portugol)





Início

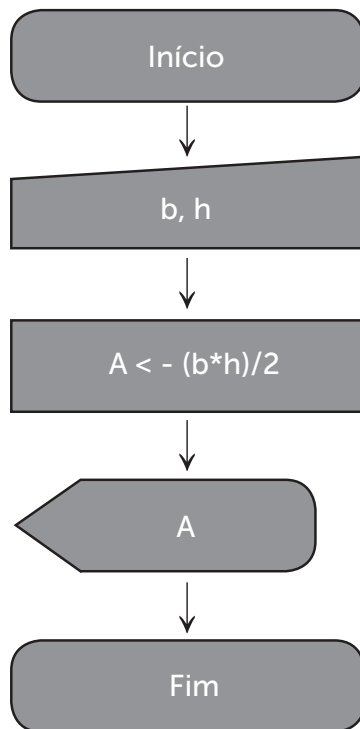
1. Leia (b, h).
2. $A \leftarrow (b \cdot h) / 2$.
3. Escreva (A).

Fim

Fluxograma

**Saiba mais**

-  Terminador
-  Processo
-  Entrada manual
-  Exibição





Faça você mesmo

Para especificar um algoritmo, você deve ter em mente que primeiro é preciso conhecer o problema a ser resolvido. Depois, você terá que identificar o problema e mapear o passo a passo para solucioná-lo. Isso é detalhar os processos. Essa sequência lógica permitirá chegar a uma solução que atenda às convenções para a sua elaboração. Vamos a mais um exemplo!

No bairro São João da Barra, na cidade de Mirandópolis, a companhia telefônica lançou uma promoção em que a cada 30 moradores que fizessem a adesão do seu plano de TV, internet e telefone, pagariam apenas o valor de R\$ 100,00 pelas assinaturas e, além disso, ganhariam um bônus de R\$ 67,00 no primeiro mês. Descubra qual o valor de cada assinatura.

| Algoritmo em linguagem natural | Algoritmo em lógica computacional |
|---|--|
| Início 1. Inserir valores 30, 67 e 100. 2. Subtrair de 67 de 100. 3. Dividir o resultado da subtração por 30. 4. Exibir o resultado. Fim | Início 1. Leia os valores A, B e C. 2. $s \leftarrow B - C$. 3. $res \leftarrow s/30$ 4. Escreva o resultado Fim |

A resolução matemática será uma equação do primeiro grau em que "x" é o custo de cada assinatura, então temos:

$$30x + 67 = 100$$

$$30x = 100 - 67$$

$$x = 33/30$$

$$x = 1.1$$

Porém, as regras matemáticas precisam ser consideradas, e como não há divisão por zero, é preciso inserir essa condição à operação agregando uma estrutura de decisão ou seleção que determine o teste que deverá ser realizado. Vamos lá. Veja o algoritmo correto:

Início

1. Leia A, B e C.
2. **Se** $C > B$ **e** $A > 0$ **e** $B \geq 0$ **e** $C > 0$ **então**.

1. $s \leftarrow C - B$.
2. $\text{valor_ass} \leftarrow s/A$.
3. Escreva o valor da assinatura, `valor_ass`.

3. Senão

1. Escreva que os valores não permitem a divisão.

Fim



Vocabulário

Sintaxe: ordem e disposição de apresentar as palavras para que sejam interpretadas.

Programação estruturada: paradigma de programação que segue a premissa de que contempla estruturas de seleção de informações, decisão e repetição.

Programação orientada a objetos: paradigma de programação que aproxima o mundo real do virtual através da abstração dos dados, vinculação das informações através de herança, encapsulamento e polimorfismo.



Assimile

[...] o papel da lógica na programação: provar que um algoritmo que foi elaborado está correto. Juntamente com a simulação do algoritmo, que consiste em executá-lo com dados reais, é possível saber se está correto e se leva a valores consistentes (SOUZA et al., 2011, p. 21).

Sem medo de errar

Agora, vamos iniciar o desenvolvimento da solução para o caso dos comerciantes do Litoral Sul. Considere que você realizou uma entrevista com os seus clientes para fazer um levantamento do que eles esperam que o protótipo apresente.

Assim, você precisa identificar os processos e escrever em estrutura de algoritmo, mas em linguagem natural, os passos que anotou. Verifique se o algoritmo a seguir pode ser considerado uma sugestão de resolução do problema e proponha também outras formas de resolvê-lo. Siga em frente!

A princípio, estabeleça quais serão os passos necessários para a sua conclusão. Veja a proposta.

Algoritmo de execução do projeto

1. Início
 2. Levantar requisitos do aplicativo junto aos comerciantes.
 3. Escrever o plano de ação do projeto.
 4. Obter a aprovação junto aos clientes.
 5. Elaborar o algoritmo do *software* em questão em linguagem natural e em pseudocódigo.
 6. Desenvolver o fluxograma das ações do aplicativo.
 7. Direcionar à equipe de desenvolvimento do projeto para a implantação.
 8. Desenvolver a solução utilizando uma linguagem de programação.
 9. Realizar os testes.
 10. Disponibilizar o aplicativo.
- Encerrar o projeto.



Atenção!

Busque a compreensão das ações que o sistema executará e que serão entregues nesta primeira fase.



Lembre-se

Você precisa identificar o passo a passo de todas as ações que o *software* deverá executar. Isto é válido tanto para a elaboração de um algoritmo simples, como trocar um pneu, como para a resolução de um problema complexo.

Avançando na prática

Pratique mais

Instrução

Desafiamos você a praticar o que aprendeu transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois as compare com a de seus colegas.

| Histórico e Definição de Algoritmos: perspectivas de linguagem | |
|--|---|
| 1. Competência de fundamento de área | Conhecer os princípios e conceitos que envolvem o aprendizado em construção de algoritmos e programação e a sua importância para o universo do desenvolvimento de sistemas. |
| 2. Objetivos de aprendizagem | Conhecer o que são, como se aplicam e a quem se destina a elaboração dos algoritmos; Conhecer como ocorre o raciocínio lógico computacional. |
| 3. Conteúdos relacionados | Definição de algoritmos; Histórico e perspectivas para a linguagem; Linguagem natural; Pseudocódigo (Português Estruturado) |
| 4. Descrição da SP | Em uma gincana escolar, o desafio era fazer com que dois grupos de três integrantes cada conseguissem atravessar uma ponte em que só é possível passar dois alunos por vez. No entanto, como são grupos diferentes, não é permitido que em qualquer dos lados da ponte fiquem mais alunos do grupo 2 (G2) que do grupo 3 (G3). Então, elabore o algoritmo que respeite essas regras e demonstre em linguagem natural e faça o teste conforme a indicação que segue. |
| 5. Resolução da SP | 1. Atravesse o aluno1 do G1 e aluno1 do G2. 2. Volte o aluno1 do G1. 3. Atravesse o aluno2 do G2 e aluno3 do G2. 4. Volte o aluno1 do G2. 5. Atravesse o aluno1 do G1 e aluno2 do G1. 6. Volte o aluno1 do G1 e aluno2 do G2. 7. Atravesse o aluno1 do G1 e aluno3 do G1. 8. Volte o aluno3 do G2. 9. Atravesse o aluno1 do G2 e o aluno2 do G2. 10. Volte o aluno2 do G2. 11. Atravesse o aluno2 do G2 2 aluno3 do G2. |



Lembre-se

Acesse o [link](http://rachacuca.com.br/jogos/missionarios-e-canibais/) e repita a sequência com o problema original entre os missionários e os canibais. Boa sorte! <<http://rachacuca.com.br/jogos/missionarios-e-canibais/>>. Acesso em: 02 abr. 2015.



Faça você mesmo

| Experimente fazer manualmente este passo a passo: Travessia | ida | volta | Ficam do lado A da ponte | Ficam do lado B da ponte |
|---|----------|-------|--------------------------|--------------------------|
| 1ª | 1G1, 1G2 | 1G1 | 1G2 | 1G1, 2G1, 3G1, 2G2, 3G2 |

| | | | | |
|----|----------|----------|------------------------------|--------------------|
| 2ª | 2G2, 3G2 | 1G2 | 2G2, 3G2 | 1G1, 2G1, 3G1, 1G2 |
| 3ª | 1G1, 2G1 | 1G1, 2G2 | 2G1, 3G2 | 1G1, 3G1, 1G2, 2G2 |
| 4ª | 1G1, 3G1 | 3G2 | 1G1, 2G1, 3G1 | 1G2, 2G2, 3G2 |
| 5ª | 1G2, 2G2 | 2G2 | 1G1, 2G1, 3G1, 1G2 | 2G2, 3G2 |
| 6ª | 2G2, 3G2 | - | 1G1, 2G1, 3G1, 1G2, 2G2, 3G2 | - |

Faça valer a pena

1. De acordo com o material de estudos, complete as lacunas com as palavras correspondentes assinalando a alternativa correta.

- i. _____: ordem e forma de apresentar as palavras para que sejam interpretadas.
- ii. _____: paradigma de programação que segue a premissa de que contempla estruturas de seleção de informações, decisão e repetição.
- iii. _____: paradigma de programação que aproxima o mundo real do virtual através da abstração dos dados, vinculação das informações através de herança, encapsulamento e polimorfismo.
 - a. programação estruturada, programação estruturada e algoritmos.
 - b. sintaxe, programação estruturada e programação orientada a objetos.
 - c. algoritmos, sintaxe e programação orientada a objetos.
 - d. sintaxe, algoritmos e programação estruturada.
 - e. variáveis, sintaxe e algoritmos.

2. Cite e explique os dois tipos de problemas relacionados aos algoritmos e a forma como esses podem ser analisados.

3. Assinale "V" para verdadeiro e "F" para falso. São características dos algoritmos:

- a. () Finitude: indica que o algoritmo deve encerrar após um número finito de execução.
- b. () Longitude: indica a capacidade de processamento do algoritmo desenvolvido.
- c. () Definição: evita definições que gerem ambiguidade.
- d. () Entrada: atribui valores ao algoritmo, a partir de especificações de variáveis e funções antes de sua iniciação.
- e. () Saída: representa os valores após o processamento, ou seja, a

execução das ações.

f. () Eficácia: todas as ações que o algoritmo realiza precisam ser executáveis em um limite de tempo predeterminado, finito, mesmo que seja sem o auxílio do computador.

4. A frase abaixo é referente a uma característica do desenvolvimento de qual conceito? Assinale a alternativa correta.

“No princípio designava a forma de resolver problemas matemáticos, depois princípios e teorias matemáticas, migrando finalmente para a área computacional, depois da invenção e consolidação dos computadores.” (PIVA Jr. et al., 2012)

- a. () declaração de variáveis.
- b. () entrada de dados.
- c. () sistemas de informação.
- d. () algoritmos.
- e. () programação orientada a objetos.

5. Quando e por quem os algoritmos foram introduzidos? Cite a lógica do algoritmo de Euclides.

Assinale a alternativa correspondente:

- a. () O termo surgiu com o astrônomo e matemático persa Abdullah Muhammad Bin Musa al- Khwarizmi, no século IX. Um dos algoritmos mais conhecidos, é o de Euclides, que determina o valor do MDC (máximo divisor comum).
- b. () Carlos Babbage desenvolveu o primeiro algoritmo aplicado para a resolução de cálculos aritméticos.
- c. () A máquina de Turing se mostra o algoritmo mais eficiente para a solução de problemas de qualquer natureza matemática até a atualidade.
- d. () O algoritmo de Euclides foi um dos primeiros que Charles Babbage desenvolveu no século XV.
- e. () Algoritmos não são considerados como uma boa prática de programação em função de os primeiros apresentados não considerarem soluções para problemas matemáticos, de acordo com Arquimedes.

6. Assinale a alternativa que representa o mecanismo de funcionamento de um algoritmo.

- a. () dados, processamento e informação.
- b. () dados, entrada e saída.

- c. () entrada, saída e banco de dados.
- d. () declaração, comandos e encerramento.
- e. () informação, processamento e saída.

7. Desenvolva o algoritmo para calcular a área de uma mesa, solicitando ao usuário que insira os valores da base e da altura. Com base nessas informações e no desenvolvimento do algoritmo, assinale a alternativa correta que representa a lógica que deverá ser implantada.

- a. () $A \leftarrow b * h$
- b. () escreva $b*h$.
- c. () leia (b,h)
- d. () digite altura (h)
- e. () leia (b)

Seção 1.2

Tipos de dados e expressões: literais, lógicas e aritméticas

Diálogo aberto

Antes de aprender como desenvolver programas de computador utilizando uma linguagem de programação, você será levado a conhecer as formas de apresentação dos elementos que devem ser considerados para efetuar as operações. Nessa área é muito comum que qualquer algoritmo elaborado apresente um contexto lógico-matemático, ou seja, necessite retornar o resultado de um cálculo, por exemplo. Porém, as expressões matemáticas não podem ser representadas nas linguagens computacionais com os mesmos símbolos a todo tempo e com a mesma forma. Precisam ser linearizadas e utilizar os operadores, que você provavelmente já conhece, mas que seguem uma padronização de acordo com a linguagem escolhida para o desenvolvimento.

Veja a Tabela 1 com os operadores matemáticos.

Tabela 1 | Principais operadores matemáticos em linguagem algorítmica

| Operações | Operador |
|---------------|----------|
| Adição | + |
| Subtração | - |
| Multiplicação | * |
| Divisão | / |

Fonte: Piva Jr. (2012, p. 59).

Diante deste contexto, veja um exemplo de expressão matemática que precisa ser linearizada para que o computador possa interpretar essa informação: $y = \frac{2x}{5x+7} + 4x$.

De acordo com a linearização da fórmula acima, obtemos: $y \leftarrow ((2*x)/(5*x+7))+(4*x)$. Observe como a representação dessa expressão foi modificada.

A fim de apresentar uma solução em algoritmos, para os comerciantes do Litoral Sul, a sua missão nesta fase é definir as expressões matemáticas, lógicas e literais que serão utilizadas. Ao conjunto de operações é dado o nome de função. Defina, de acordo com o algoritmo apresentado anteriormente, quais serão e como devem ser implementadas. Além das expressões, identifique também quais são os tipos de dados e as respectivas variáveis ou constantes que devem ser declaradas para a realização das operações: “assim como os números, que em algum momento da história tiveram sua forma e utilização padronizadas, aconteceu a mesma coisa com as expressões matemáticas quando utilizadas em algoritmos ou linguagens de programação computacionais.” (PIVA Jr., 2012, p. 58).

Você pode utilizar, para os cálculos mais complexos, operadores e funções em linguagem algorítmica para as mais diversas operações que o aplicativo deverá executar. As expressões, os tipos de dados e variáveis aqui apresentados servem como referência para o desenvolvimento de *software* independentemente da linguagem ou paradigma de programação, no entanto, precisa se atentar à sintaxe de cada uma delas.

Não pode faltar

Alguns operadores matemáticos são utilizados para cálculos considerados simples, mas e se você precisar calcular, por exemplo, uma equação de segundo grau ou ainda uma integral, derivadas, logaritmos? Enfim, há uma infinidade de funções e suas aplicações são importantes! Vamos conhecer outros operadores matemáticos que auxiliam na elaboração de funções e cálculos mais complexos. Veja as tabelas abaixo e as respectivas descrições:

Tabela 2 | Operações complexas com operador

| Operações | Operador | Exemplo |
|-------------------|--------------|---|
| Exponenciação | \wedge | $a \wedge b$ |
| Divisão inteira | \backslash | $a \backslash b$ retorna o valor inteiro da divisão |
| Módulo | $\%$ | Complementar: “resto de divisão de a por b” |
| Inversão de sinal | $-$ | $-a$ ($-(-a)$ resulta em a) |

Fonte: Piva Jr. (2012, p. 61).

Tabela 3 | Operações complexas com funções (para representação em pseudocódigo utilizando o VisuAlg).

| Operações | Funções | Explicação |
|-------------------------|-----------|---|
| Raiz Quadrada | Raizq(x) | Raiz quadrada |
| Exponenciação | Exp(x,y) | x elevado a y |
| Valor absoluto | Abs(x) | Valor absoluto de x. |
| Arco Cosseno | ArcCos(x) | Ângulo em radianos cujo cosseno é representado por x. |
| Arco Seno | ArcSen(x) | Ângulo em radianos cujo seno é representado por x. |
| Arco Tangente | ArcTan(x) | Ângulo em radianos. |
| Cosseno | Cos(x) | Cosseno em radianos do ângulo x. |
| Cotangente | CoTan(x) | Retorna o ângulo x em radianos. |
| Parte inteira do número | Int(x) | Retorna a parte inteira do número x. |
| Logaritmo | Log(x) | Retorna o logaritmo de x na base 10. |
| Logaritmo Neperiano | LogN(x) | Retorna o logaritmo neperiano de x (base e). |
| Valor Pi(π) | Pi | Retorna o valor 3.141592. |
| Quadrado | Quad(x) | Retorna o quadrado de x. |
| Valor aleatório | Rand | Randômico entre 0 e 1. |
| Seno | Sen(x) | Retorna o seno do ângulo x em radianos. |
| Tangente | Tan(x) | Retorna a tangente do ângulo x em radianos. |

Fonte: Piva Jr. (2012, p. 61- 62).



Assimile

Um bom exemplo de linearização de funções pode ser representado pela equação de 2º grau. Observe: $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

Após a linearização, a função ficará da seguinte forma: $x \leftarrow (-b + \text{raizq}(b^2 - (4 * a * c))) / (2 * a)$ e também podemos representar a linearização da fórmula, para calcular o delta e encontrar a segunda raiz da equação com a respectiva variação do sinal. Analise:

$$x \leftarrow (-b - \text{raizq}(b^2 - (4 * a * c))) / (2 * a).$$

Além dos operadores matemáticos, conheça também os operadores relacionais. Veja a Tabela 4.

Tabela 4 | Operadores relacionais (podem variar de acordo com a sintaxe)

| Operações | Operador | Exemplo |
|----------------|----------|------------------------------------|
| Igual | = | $a = b$ (a é igual a b?) |
| Diferente | < > | $a < > b$ (a é diferente de b?) |
| Maior que | > | $a > b$ (a é maior que b?) |
| Menor que | < | $a < b$ (a é menor que b?) |
| Maior ou igual | >= | $a >= b$ (a é maior ou igual a b?) |
| Menor ou igual | <= | $a <= b$ (a é menor ou igual a b?) |

Fonte: Piva Jr. (2012, p. 62-63).

Os operadores lógicos são muito importantes para a composição das expressões necessárias à execução dos programas. Observe na Tabela 5 quais são eles.

Tabela 5 | Operadores lógicos

| Operações | Significado |
|-----------|---|
| Não | É unário na negação. Tem a maior precedência entre os operadores lógicos. não (VERDADEIRO)= FALSO, e não (FALSO)= VERDADEIRO |
| Ou | Resulta em verdadeiro quando um dos seus operandos lógicos for verdadeiro. |
| E | Operador que resulta VERDADEIRO apenas se seus dois operandos lógicos forem verdadeiros. |
| Xou | Operador que resulta VERDADEIRO se seus dois operandos lógicos forem diferentes, e FALSO se forem iguais. |

Fonte: Piva Jr. (2012, p. 63).

Para verificar se as proposições são verdadeiras ou falsas de acordo com o operador lógico que é utilizado, é aconselhável que você saiba como realizar o teste. Para tal, a sugestão é o uso da tabela verdade. Então, a considerar os operadores lógicos, confira a seguir os testes que podem ser realizados. Para a expressão: $A = B \text{ e } D$, observe:

| B | D | $A = B \text{ e } D$ |
|------------|------------|----------------------|
| Falso | Falso | Falso |
| Falso | Verdadeiro | Falso |
| Verdadeiro | Falso | Falso |
| Verdadeiro | Verdadeiro | Verdadeiro |

Exemplo: para que a condição seja verdadeira e atenda à solicitação do operador lógico, suponha que uma concessionária tenha uma meta por vendedor de 18 carros por mês e o mínimo de R\$ 58.000,00 de valor bruto de vendas. Se esse vendedor atingir a meta, então, ele receberá 10% de comissão. Se não, o vendedor recebe

apenas 0,08% do total como participação nas vendas. Seja A o valor da comissão, B o total de carros vendidos e D o valor mínimo de vendas, elabore a expressão lógica que atenda a essa operação. **Resposta:** a expressão que representa essa operação é: $A = ((B \geq 18) \text{ e } (D \geq 58.000))$.

Agora, vamos fazer a tabela verdade para a expressão com o operador lógico "ou".

| B | D | A = B ou D |
|------------|------------|------------|
| Falso | Falso | Falso |
| Falso | Verdadeiro | Verdadeiro |
| Verdadeiro | Falso | Verdadeiro |
| Verdadeiro | Verdadeiro | Verdadeiro |

Exemplo 2: para o mesmo problema apresentado acima, considere que o vendedor, para ganhar a comissão total, precisa atingir ou o valor da meta de vendas que é de 18 carros por mês, ou o valor mínimo em reais que é de R\$ 58.000,00. Elabore a expressão que representa essa condição. **Resposta:** a expressão que representa essa operação é: $A = ((B \geq 18) \text{ ou } (D \geq 58.000))$.

Observe a ação do operador lógico **XOU** ou, operador lógico exclusivo, é derivado das conjunções, disjunções inclusivas e negação. É preciso que **ao menos uma** das condições seja verdadeira para que resulte em verdadeiro. Veja a sua tabela verdade.

| B | D | A = B xou D |
|------------|------------|-------------|
| Falso | Falso | Falso |
| Falso | Verdadeiro | Verdadeiro |
| Verdadeiro | Falso | Verdadeiro |
| Verdadeiro | Verdadeiro | Falso |

Exemplo 3: Considere ainda o problema sugerido no exemplo 1 e elabore a expressão que o representa. A diferença desse operador está em aceitar ao menos uma das condições para que resulte em verdadeiro. **Resposta:** $A = ((B \geq 18) \text{ xou } (D \geq 58.000))$.

Abaixo a tabela verdade do operador lógico "não".

| B | A = não B | A = B ou D |
|------------|------------|------------|
| Falso | Verdadeiro | Falso |
| Verdadeiro | Falso | Verdadeiro |

Exemplo 4: ainda considerando a situação-problema do exemplo 1, elabore a expressão que representa essa operação. **Resposta:** $A = (\text{nao}(B \geq 18) \text{ e } (\text{nao}(D \geq 58.000)))$. Os materiais de referência podem trazer o sinal "~" como indicação da notação para "não".



Pesquise mais

O link abaixo contém informações sobre algoritmos e aplicações. Acesse e teste os exemplos disponíveis. Disponível em: <<http://pichiliani.com.br/2013/04/visualizando-algoritmos/>>. Acesso em: 25 mar. 2015.

Os operadores lógicos auxiliam a execução das operações, porém, estas afetam diretamente o valor contido nos elementos que se chamam “variáveis”. Por exemplo, nas expressões acima nós temos as variáveis A, B e D. Os operadores relacionais “>” maior e “=” igual e os valores referentes a cada uma das variáveis. Note também que os valores das variáveis B e D não são alterados durante a execução, e por esse motivo são chamados de constantes. É importante que o valor da variável seja atribuído logo no momento de sua declaração.

Além desse aspecto, você precisa considerar como se faz para determinar o nome da variável, pois este será o seu identificador. São permitidos identificadores com letras maiúsculas, exemplo “COMISSAO_INTEGRAL”, no entanto, é recomendado que seja objetivo e transmita a informação que será ali armazenada ou manipulada, então, a sugestão é: “comissaoIntegral”.



Reflita

Ao espaço reservado previamente em memória, e devidamente rotulado com um nome (identificador), chamamos de variável. Caso o valor não venha a se alterar durante o programa, chamamos esse espaço de constante. (PIVA JR., 2012, p. 85)

- O espaço em branco também é um caractere especial.

As variáveis armazenam valores que são classificados quanto ao seu tipo e características. Veja na tabela abaixo os tipos de dados existentes.

| Tipos de dados | Características | Exemplos |
|----------------|---|-----------------------------------|
| Inteiro | Contempla todos os números inteiros relativos (positivos, negativos e nulos). | 12, -56, 852, 146698 |
| Real | Conjunto dos números reais nulos, negativos e positivos. | 23.5, 85.4, -354.8, -74, 3.141618 |

| | | |
|--------------------|---|--|
| Caracter | Composta por todos os caracteres alfanuméricos e especiais. | 0 - 9, A-Z, a-z, #,\$%,&,*@ |
| Lógico ou booleano | Assumem apenas a situação de verdadeiro ou falso. | 0 ou 1, verdadeiro ou falso, sim ou não. |

Fonte: Adaptado de Piva Jr. (2012, p. 86).

A seguir, há um exemplo para você praticar. Siga em frente.



Exemplificando

Observe o exemplo abaixo e as respectivas explicações.

Teste a transposição do algoritmo de soma de dois números (Scilab ou no VisuAlg-pseudocódigo) e observe como aparecem as entradas e saídas de dados e informação:

Algoritmo: "soma dois números"

// Função: algoritmo que exhibe o resultado da soma entre dois números.

// Autor : JJJ

// Data : 05/03/2015

// Seção de Declarações

var

x, y: inteiro

inicio

// Seção de Comandos

escreval("Digite o primeiro número: ")

leia(x)

escreval("Digite o segundo número: ")

leia(y)

escreva("A soma dos números é: ",x+y)

fimalgoritmo;

As variáveis x e y são do tipo de dado "inteiro".

Dados de entrada: o comando "escreval" indica que o dado é **solicitado ou exibido** ao usuário, e o "l" que há a **quebra de linha**.

O comando "leia" indica que o dado é verificado na variável indicada.

Saída: soma (informação)



Faça você mesmo

Agora, observe o mesmo exemplo na transposição para uma linguagem de programação (C, implementada no Dev C++):

| | |
|---|--|
| /* Programa deverá apresentar a soma entre dois números*/ | |
| #include <stdio.h> | Primeiramente é preciso estabelecer quais são as bibliotecas utilizadas por este programa em C. |
| #include <stdlib.h> | |
| main() | Em seguida, há a declaração das variáveis, especificando o tipo de dado e a solicitação da entrada de dados. |
| { int a, b, soma; | |
| printf("Informe o primeiro numero:\n"); | |
| scanf("%d", &a); | |
| printf("Informe o segundo numero:\n"); | |
| scanf("%d", &b); | |
| soma = a + b; | A variável "soma" recebe o valor da adição das variáveis "a" com "b". |
| printf("A soma dos numeros e: %d", soma); | |
| fflush (stdin); | "%" sucedido da letra "d" representa a exibição do valor do dado do tipo inteiro contido na variável "soma". |
| } | |
| | fflush: limpa o buffer |

Os comandos **"printf"** e **"scanf"** são, respectivamente, o "escreva" e "leia" do exemplo anterior, porém, escritos em pseudocódigo.

Sem medo de errar

Aplicação dos procedimentos de atuação convenientes à SP

1. Observe que no caso apresentado, para resolver a solicitação de desenvolvimento do aplicativo para os comerciantes do Litoral Sul será proposto o algoritmo a seguir. O pseudocódigo apresenta inicialmente o nome do algoritmo:

"Litoral Sul".

2. Em seguida, traz a especificação da fase do projeto e o que o algoritmo apresenta.

3. Indicar o autor do algoritmo também é recomendável, bem como a sua data de elaboração. Fatores que atribuem maior controle das alterações desse projeto, inclusive.

4. Observe que antes de iniciar efetivamente o bloco de declaração de variáveis, há a indicação desse com a palavra "var", sintaxe válida para o VisuAlg, que visa demonstrar a sequência em que as operações precisam ser apresentadas nos algoritmos e consequentemente nos programas.

5. São declaradas três variáveis do tipo caracter, que correspondem ao tipo de dado "char" em outras linguagens. A variável média foi declarada com o tipo de dado real, o que em linguagem de programação C e Java, é representado pelo tipo de dado "float". As demais variáveis receberão valores do tipo inteiro, equivalente ao tipo de dado "integer" nas demais linguagens.

6. Observe que em seguida à declaração das variáveis há efetivamente o início das operações que o algoritmo deverá executar para atender à demanda solicitada.

7. Os comandos "leia" e "escreva" representam, respectivamente, entrada e saída de dados. Houve a necessidade de uso de uma estrutura de seleção, que será melhor descrita a seguir. Nesse caso, ela serve para inserir o teste da condição que foi imposta e, a partir da escolha do usuário, será possível verificar e seguir adiante com as execuções previstas.

• Pseudocódigo

algoritmo "Litoral Sul"

// Função : representação do algoritmo de ações do aplicativo- Fase 1.

// Autor : JJJ

// Data : 12/03/2015

// Seção de Declarações

var

login, agenda, email: caractere

média: real

satisfação, satisfeito, insatisfeito, categoria, g, h: inteiro

inicio

// Seção de Comandos

escreva("Clique no ícone login para acessar o sistema.")

leia(login)

escreval("Digite 1 para escolher serviços de gastronomia ou 2 para hotelaria.")

leia(categoria)

se categoria = 1 então

g \leftarrow g + 1

senão

h \leftarrow h + 1

fimse

média \leftarrow (g + h)/2

escreval("A quantidade média de acessos é de:", média)

escreval("A quantidade de acessos para gastronomia é de:", g)

escreval("A quantidade de acessos para hotelaria é de:", h)

fimalgoritmo



Atenção!

Outro ponto de atenção aqui é para a atribuição de valores nas variáveis, que não têm característica de constante justamente porque podem sofrer alterações durante a execução do programa, de acordo com a opção escolhida.



Lembre-se

Você tem total autonomia para melhorar os processos e incrementar as funcionalidades, então, fique de olho no cronograma desse projeto e pratique!

Avançando na prática

| Pratique mais | |
|---|---|
| Instrução Desafiamos você a praticar o que aprendeu transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois as compare com a de seus colegas. | |
| Tipos de dados e expressões: literais, lógicas e aritméticas | |
| 1. Competência de fundamento de área | Conhecer os princípios e conceitos que envolvem o aprendizado em construção de algoritmos e programação e a sua importância para o universo do desenvolvimento de sistemas. |
| 2. Objetivos de aprendizagem | <ul style="list-style-type: none"> • Conhecer o que são, como se aplicam e a quem se destina a elaboração dos algoritmos; • Conhecer como ocorre o raciocínio lógico computacional. |
| 3. Conteúdos relacionados | <ul style="list-style-type: none"> • Expressões literais, lógicas e aritméticas; • Tipos de dados, variáveis e constantes. |
| 4. Descrição da SP | Nas turmas da 6ª série da escola "Estudar Faz Bem", a professora estava reforçando com os alunos as operações matemáticas. Para integrar as tecnologias de informação e comunicação nas aulas para essa turma, a professora solicitou o desenvolvimento de um aplicativo simples que permitisse aos alunos testar em uma calculadora amigável e interativa tais operações. O intuito é fazer com que os alunos, desde as primeiras etapas de seu ensino, compreendam a lógica computacional e, dessa forma, realizem a prova real manuscrita e também visualizem outras possibilidades de se realizar essas operações. Sua missão é desenvolver o algoritmo que apresente as operações adição, subtração, multiplicação e divisão, de forma que a interface apresentada seja fácil para esse público compreender. |
| 5. Resolução da SP | Faça você mesmo: algoritmo "Operações matemáticas" // Função : Elabore um algoritmo que receba dois números e a partir desses, exiba os respectivos resultados com as operações matemáticas adição, subtração, multiplicação e divisão. // Seção de Declarações var x, y: real início // Seção de Comandos escreva("Digite o primeiro número: ") leia(x) escreva("Digite o segundo número: ") leia(y) escreval("A soma é: ",x+y) escreval("A subtracao é: ",x-y) escreval("A multiplicacao é: ",x*y) escreval("A divisao é: ",x/y) fimalgoritmo |

**Lembre-se**

"A definição de um tipo de dado para uma determinada variável indica a reserva de um determinado espaço em memória. Caso esse espaço seja insuficiente, ocorrerá um erro de **time mismatch** ou **overflow**, levando à interrupção da execução do programa." (PIVA JR., 2012, p. 101)

Faça valer a pena

1. Elabore um algoritmo que permita ao lojista inserir o valor do produto e o valor do desconto. Em seguida, apresente o valor do produto e o seu valor com o desconto. Assinale a alternativa que indica corretamente a expressão para este cálculo.

- a. ☐ `desconto <- v_produto*pdesconto/100`
- b. ☐ `desconto <- desc/100 - valor_prod`
- c. ☐ `v_produto<- desconto * desconto/100`
- d. ☐ `desconto <- v_produto* desconto/0,1`
- e. ☐ `desconto <- v_produto* desconto/1,0`

2. Desenvolva um algoritmo que receba o valor de um ângulo em graus e exiba o seu valor em radianos: apresente o pseudocódigo e em linguagem C.

3. Elabore um algoritmo para calcular o consumo de combustível de um carro. Assinale a alternativa correta, após a análise da linha de comando abaixo.

Escreval ("O consumo de combustível do seu veículo é: ",distancia/combustivel)

- a. ☐ não é possível realizar esta operação.
- b. ☐ a sintaxe está incorreta.
- c. ☐ a sintaxe está correta, pois apresenta o valor médio de combustível que o tanque do carro comporta.
- d. ☐ o comando está correto, pois é possível efetuar uma operação matemática seguida da exibição da mensagem.
- e. ☐ está incorreta, pois não existe o comando escreval.

- 4.** Elabore um algoritmo que realize a cotação do dólar.
- 5.** Em um processo de precificação, o empreendedor precisa de uma ferramenta que o auxilie a estabelecer a sua margem de lucro e a calculá-la de forma mais rápida e facilitada. Para tal, desenvolva um algoritmo que receba o valor do produto, o valor da margem de lucro desejada em percentual e exiba o valor final. Assinale a alternativa que representa as variáveis que são necessárias para a realização desse algoritmo e o seu respectivo tipo de dado.
- a. () preço1, preçoTotal e preçoLíquido: caractere.
 - b. () valorProduto, margemLucro, valorFinal: real.
 - c. () valorProduto, margemLucro, valorFinal: inteiro.
 - d. () valorProduto, margemLucro, valorFinal: caractere.
 - e. () preço1, preçoTotal e preçoLíquido: inteiro.
- 6.** Dadas as sequências de tipos de dados abaixo, assinale a alternativa correta.
- a. () inteiro, real e caractere.
 - b. () real, strong e integer.
 - c. () char, íntegro e real.
 - d. () strong, íntegro e short.
- 7.** Assinale a alternativa que contém apenas os operadores relacionais.
- a. () and, or, xor
 - b. () >, <, >=, <=
 - c. () e, ou, xou
 - d. () *, /, -, +
 - e. () >, /, <=, *

Seção 1.3

Representação de algoritmos e o ambiente de programação

Diálogo aberto

No cenário descrito no *Convite ao Estudo*, a empresa de desenvolvimento “Think Now” precisa considerar inicialmente:

a. Os processos que o aplicativo deverá executar e como é o seu algoritmo. Além de buscar informações de soluções e algoritmos existentes que possam servir de parâmetro para esse desenvolvimento;

b. O ambiente de programação e como se dá a interpretação das ações para a linguagem de computador;

c. A lógica existente entre os processos que o aplicativo deverá executar e a sua respectiva ilustração com a elaboração do fluxograma;

d. A competência técnica da equipe de desenvolvimento para a identificação das expressões literais, lógicas e aritméticas, além de saber quando deve ser utilizada uma ou outra;

e. Além disso, será necessário que a equipe já identifique os tipos de dados que serão utilizados, o que será variável ou constante e inicie o seu processo de declaração, em ambiente computacional.

É preciso estabelecer qual será o passo a passo para realizar as consultas e reservas, e de forma que seja simples, rápido e fácil ao usuário final. A empresa também precisa identificar quais são os riscos que o projeto pode enfrentar para que seja concluído, ou seja, entregue.

Sendo assim, é possível inferir que devem ser apresentados:

- Para o passo a passo, além da observação do processo inicialmente informado, será preciso interpretar e identificar se todos os envolvidos compreendem e aprovam as etapas definidas;

- Elaborar o fluxograma de ações do *software*, ou seja, a representação do seu algoritmo é componente fundamental desse processo;

- A equipe responsável pelo desenvolvimento do projeto deverá alinhar as solicitações de funcionalidades do aplicativo que realmente são passíveis de realização dentro do prazo estipulado.

Além desses, você, que faz parte do quadro de colaboradores da empresa dedicados a esse projeto, deverá considerar, para o desenvolvimento de suas atividades, todos os tópicos de "a" a "e", descritos na situação acima, e auxiliar a sua equipe na resolução desta tarefa. Bom trabalho!

Não pode faltar

Considere que os algoritmos, de forma geral, visam estabelecer a sequência lógica existente entre as ações que devem ser executadas. Para tal, símbolos podem ser utilizados de forma que representem a informação desejada. Veja um exemplo de como pode ser representado um processo a partir do uso de operadores lógicos (conjunção "e", disjunção ou condicional "ou"), nesse caso, é descrito como exemplo o operador "e" (PIVA JR., 2012, p. 9):

p: $1 + 1 = 2$

r: $2 + 1 = 3$

q: $p \text{ e } r$

p é verdadeiro, pois a afirmação " $1+1=2$ " é verdadeira; r é verdadeiro, pois a afirmação " $2+1=3$ " é verdadeira. Se p é verdadeiro e r é verdadeiro, então q também é verdadeiro, pois " $V \text{ e } V = V$ ".

Na representação acima, o "q" é uma proposição verdadeira resultante de "p" e "r", que também o são. Essa lógica nada mais é do que uma representação lógica formal ou simbólica.

Observe o exemplo de um algoritmo que define o cálculo da média dos alunos, a considerar duas notas para obter um *status* de aprovação ou reprovação, caso a média seja inferior a 6,0:

Inserir nota 1.

Inserir nota 2.

Calcular a média sendo que: $(\text{nota } 1 + \text{nota } 2)/2$.

Exibir a média.

Verificar se a média é menor do que "6,0".

Se sim, indicar "aluno reprovado".

Se não, indicar "aluno aprovado".

Outro exemplo, um algoritmo de um programa que calcule a soma de dois números (mesma lógica do primeiro exemplo):

$a \leftarrow 1$

$b \leftarrow 2$

$c \leftarrow a + b$

$c = 3$



Refleta

"Saber as fases e as estratégias de resolução de um problema auxilia a construir novas ferramentas mentais para compreender e resolver novos problemas. Quanto mais praticar, mais ferramentas terá e mais problemas conseguirá resolver." (PIVA JR., 2012, p. 33)

A seguir, o exemplo de cálculo da média em C (Dev C++):

`/* Programa deverá apresentar a média de duas notas*/`

`#include <stdio.h>`

`#include <stdlib.h>`

`main()`

`{`

`float a, b, media;`

`printf("Informe duas notas:\n");`

`scanf("%f%f", &a, &b, &c);`

`media = (a + b + c)/3;`

`printf("A media e: %f, %f e %f igual a %f", a, b, c, media);`

`fflush (stdin);`

`}`

As prioridades de execução respeitam as propriedades matemáticas.

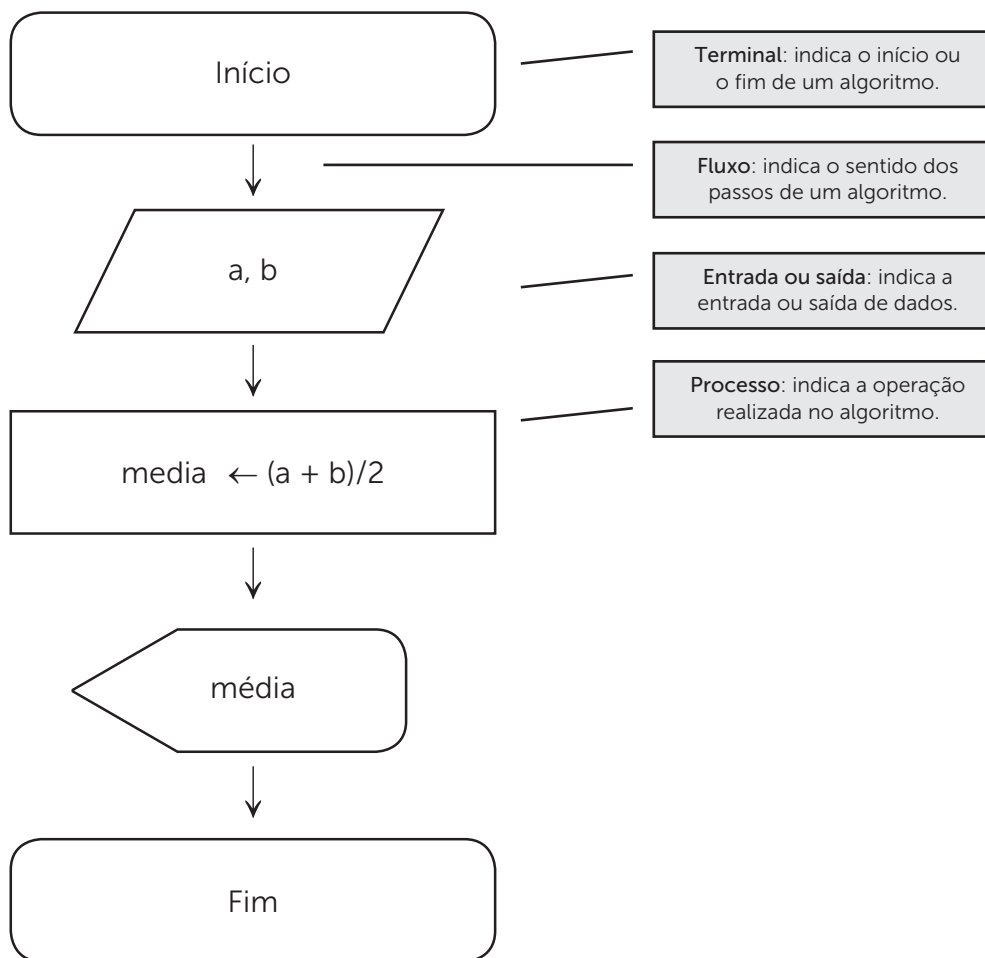


Pesquise mais

No site C Programming, você encontrará mais informações sobre a linguagem C. Disponível em: <<http://www.cprogramming.com/>>. Acesso em: 15 mai. 2015.

É possível a utilização de símbolos e também de uma linguagem não computacional, como no caso do algoritmo de cálculo da média. Nesse último caso, as entradas de dados são "nota 1" e "nota 2". Processamento é o cálculo efetivamente; e a média é a saída que se obtém do processamento.

Outra forma de representar um algoritmo é através do seu fluxograma. O exemplo será do algoritmo elaborado para realizar a soma de dois números. Veja abaixo como fazer e o que significa cada símbolo utilizado.



Analise o fluxograma e identifique que há a necessidade da declaração das variáveis, seguido da indicação da operação que será realizada, ou seja, o processamento e, logo após, há a apresentação do resultado desse processamento, uma saída, portanto, e o encerramento do processo do algoritmo, que representou, nesse caso, a soma de dois números.



Assimile

Representação do algoritmo pode ser:

- Algoritmo não computacional: é escrito em linguagem natural.
- Algoritmo computacional: é escrito em pseudocódigo ou português estruturado ou em uma linguagem de programação, ou seja, implementado.
- Fluxograma.

Para facilitar a sua compreensão acerca do que é algoritmo e como ele pode auxiliar na resolução de problemas, tenha em mente os passos:

1. Identifique o problema;
2. Imagine como pode resolver;
3. Analise as soluções e escolha a que for mais viável;
4. Reavalie se necessário.

Vamos exercitar mais um pouco o conteúdo. Suponha que em uma partida de vôlei o sistema registra os pontos a partir do lançamento manual, ou seja, realizado pelo juiz ou assistente da partida, que atualiza os valores ponto a ponto. Nesse caso, para estabelecer qual é o ganhador, o sistema faz a contagem dos pontos do jogador A e do jogador B, o que tiver a maior quantidade de pontos é o vencedor. Faça o algoritmo em linguagem natural, em pseudocódigo e o seu fluxograma.



Exemplificando

Algoritmo em linguagem natural:

1. Identifique os jogadores;
2. Lance os pontos do jogador A;

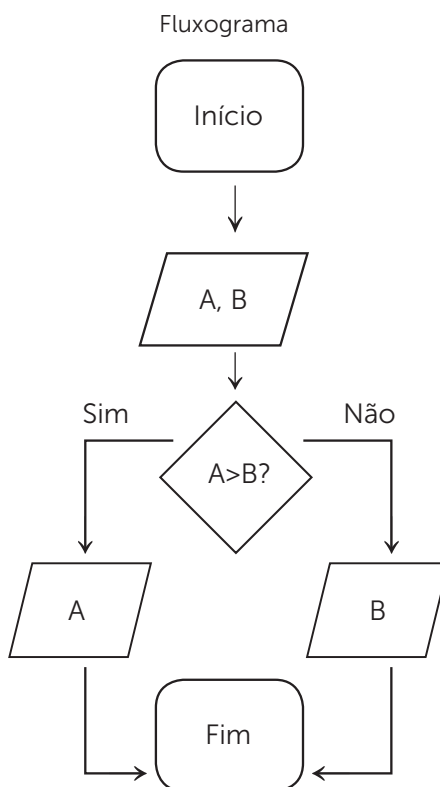
3. Lance os pontos do jogador B;
4. Verifique a maior quantidade de pontos;
5. Se jogador A com maior quantidade de pontos, então, Jogador A é o campeão, senão, Jogador B;
6. Encerra a partida.

Algoritmo em pseudocódigo (VisuAlg):

```
algoritmo "Verificar valor maior ou menor"
    // Função: Cálculo de pontos: maior ou menor
    // Autor: "Nós"
    // Data: 06/01/2015
    // Seção de Declarações
    var
        A, B: inteiro
    inicio
        // Seção de Comandos
        escreval("Informe os pontos do jogador A: ")
        leia (A)
        escreval ("Informe os pontos do jogador B: ")
        leia (B)
        se A > B entao
            escreva ("O campeão é o jogador A:", A )
        senao
            escreva ("O campeão é o jogador B:", B)
        fimse
    finalgoritmo
```

**Faça você mesmo**

Tente fazer o diagrama de bloco ou fluxograma do algoritmo acima.

**Sem medo de errar**

Agora é o momento de iniciar a resolução do projeto que a Think Now assumiu, e você está responsável por mapear os processos que o aplicativo deverá contemplar. Você e sua equipe, ao se reunirem com os comerciantes, identificaram que a maior preocupação deles estava em identificar de forma contínua, ou seja, desde a primeira fase da implantação, qual é a percepção do cliente em relação ao aplicativo, sua aceitação e usabilidade. Sugeriram um processo inicialmente mais simples, porém, que já retornasse a média de pessoas que acessaram. Então, a sua tarefa agora é mapear os processos para o desenvolvimento e apresentar o algoritmo desse aplicativo em linguagem natural, fluxograma e pseudocódigo. Então, mãos à obra.

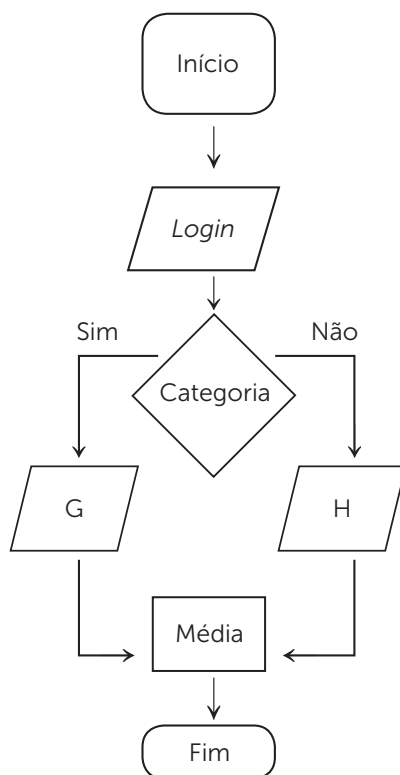
**Atenção!**

Para compreender os processos, é importante interpretar a solicitação e, com isso, escrever o passo a passo para a execução sem deixar de contemplar os processos-chave dessa operação.

A seguir, uma sugestão de transposição dos dados levantados, em **linguagem natural**.

1. Início
2. O aplicativo deverá permitir o login por cadastro ou por rede social.
3. Após logar, o usuário será direcionado a um processo para escolher a categoria: gastronomia ou hotel.
4. Aplicativo contabiliza acessos de gastronomia e hotelaria e gera a média.
5. Fim.

A definição dos processos inicialmente identificados, representados em **fluxograma**:





Lembre-se

Os fluxogramas precisam apresentar as ações do sistema, os processos, as entradas e saídas de dados. Você pode, inclusive, inserir as estruturas de seleção, repetição e decisão na representação do algoritmo. Pesquise mais na bibliografia básica e resolva os exercícios para praticar todas as formas de representação dos processos. Leia a Norma ISO 5807/1985, que estabelece os padrões que devem ser seguidos para a elaboração de diagramas e fluxogramas.

Avançando na prática

| Pratique mais | |
|---|--|
| Instrução Desafiamos você a praticar o que aprendeu transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois as compare com a de seus colegas. | |
| Representação de algoritmos e o ambiente de programação | |
| 1. Competência de fundamento de área | Conhecer os princípios e conceitos que envolvem o aprendizado em construção de algoritmos e programação, e a sua importância para o universo do desenvolvimento de sistemas. |
| 2. Objetivos de aprendizagem | <ul style="list-style-type: none"> • Conhecer o que são, como se aplicam e a quem se destina a elaboração dos algoritmos; • Conhecer o ambiente de programação e como se dá o raciocínio lógico computacional. |
| 3. Conteúdos relacionados | <ul style="list-style-type: none"> • O ambiente de programação; • Formas de representação de algoritmos; |
| 4. Descrição da SP | Suponha que no restaurante francês JacquesBistrô, além do valor do salário, os garçons não podem receber gorjetas dos clientes. Então, há uma calculadora especial que está programada para executar essa operação. Assim que o garçom encerra a conta no sistema, a calculadora solicita que ele digite dois números inteiros menores que 20. A comissão será calculada pela soma dos dois números e de duas formas distintas. Veja a fórmula $c1 = v1 + v2/100$ e a outra fórmula é: $c2 = (v1+v2)/100$. Para encerrar, o sistema calcula a soma das duas operações e exibe o valor que será pago ao garçom como comissão. Faça o algoritmo dessa calculadora em pseudocódigo e em linguagem C. |
| 5. Resolução da SP | <pre> algoritmo "JacquesBistrô" var n1,n2: inteiro c1,c2,comissaoFinal: real inicio // Seção de Comandos escreva("1º Número= ") leia(n1) escreva("2º Número= ") leia(n2) c1 <- n1 + n2 / 100 </pre> |

5. Resolução da SP

```
c2 <- (n1 + n2)/100
comissaoFinal<- c1 +c2
escreval("Comissão final= ", comissaoFinal:7:2)
finalgoritmo
```



Lembre-se

O título do algoritmo deve ser o mais objetivo possível, expressando exatamente a função realizada. O espaço reservado para declaração de variáveis vem antes do início dos processos de execução em VisuAlg.



Faça você mesmo

```
#include <stdio.h>
#include <conio.h>
main()
{ int n1, n2;
  float c1,c2, cf;
  printf("n1 = "); scanf ("%d", &n1);
  printf("n2 = "); scanf ("%d", &n2);
  c1 = n1 + n2 / 100;
  c2 = (n1 + n2)/100;
  cf= c1 + c2;
  printf("\n cf = %7.2f", cf);
  getch();
}
```

Faça valer a pena

1. Uma loja de artes e decoração, de porte pequeno, está precisando de um módulo de cálculo que indique, a partir do valor total da compra, o valor das prestações e parcela em até cinco vezes. Assinale a alternativa que representa o cálculo correto a ser inserido.

- a. () prestação <- compra/5
- b. () leia (compra)
- c. () compra = prestação/5
- d. () escreva(compra <- prestação)
- e. () leia (prestação = compra/5)

2. Seguindo o raciocínio lógico requerido, analise o algoritmo abaixo e assinale a alternativa que melhor representa o seu processamento.

```
var
distancia, combustivel: real
inicio
// Seção de Comandos
escreval("* Cálculo de consumo de combustível *")
escreva("Informe a distância a percorrer: ")
leia(distancia)
escreva("Informe a quantidade (em litros) de combustível gasto: ")
leia(combustivel)
escreval("O consumo de combustível será de: ",distancia/combustivel)
finalgoritmo
```

- a. () o algoritmo acima calcula o valor que se pagará ao completar o tanque de combustível.
- b. () o algoritmo acima calcula a quantidade de consumo de combustível do automóvel.
- c. () o algoritmo representa a seção de comandos para calcular o valor do combustível.
- d. () o algoritmo representa a situação real usada para estabelecer os preços de mercado do combustível.
- e. () o algoritmo informa simplesmente quais são as variáveis utilizadas para o cálculo do combustível.

3. Em uma empresa de automóveis, os vendedores, além do seu salário fixo, recebem uma comissão de 15% a cada venda realizada. A empresa precisa de um programa que, a partir da entrada do nome do vendedor, mostre o seu salário fixo e o valor total de vendas que ele fez no período. O valor da comissão se aplica ao valor total de vendas. Elabore o algoritmo equivalente a esta operação.

4. Considere duas variáveis, X e Y. Desenvolva um algoritmo que realize a troca dos valores das variáveis, ou seja, a variável X recebe o valor da variável Y e vice-versa e, ao final, o algoritmo deve apresentar também os valores trocados.

5. Analise a expressão: `v_real <- v_dolar/cotação` e assinale a alternativa que substituiria sem perdas o resultado se estivesse escrito na linguagem de programação C.

- a. ☐ `v_rs← v_us/cot;`
- b. ☐ `v_real == v_dolar/cotação`
- c. ☐ `v_real = v_dolar/cotação;`
- d. ☐ `real = dólar/5.`

6. Assinale a alternativa que indica o comando de entrada em C.

- a. ☐ `var`
- b. ☐ `get`
- c. ☐ `printf`
- d. ☐ `scanf`
- e. ☐ `fflush`

7. Assinale a alternativa que indica o comando de saída em C.

- a. ☐ `main()`
- b. ☐ `stdin.h`
- c. ☐ `set`
- d. ☐ `scanf`
- e. ☐ `printf`

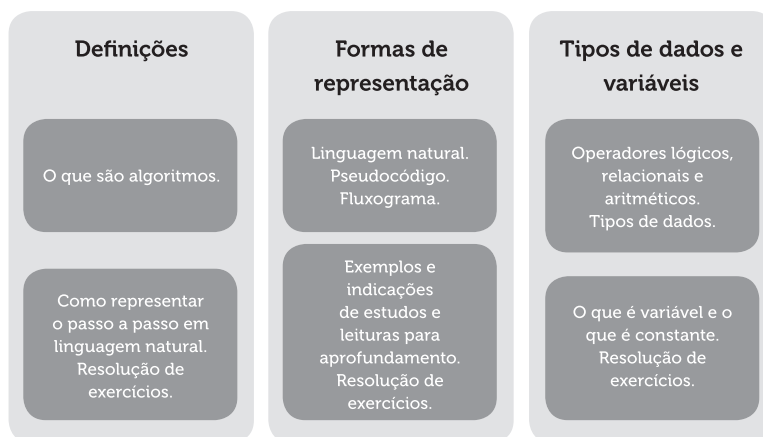
Seção 1.4

Declaração de variáveis e constantes

Diálogo aberto

A fim de sistematizar o conteúdo visto até o momento, nesta seção você pode identificar todos os pontos de atenção que precisam ser trabalhados para que seja possível a elaboração de algoritmos, ou seja, ampliar o seu contato com as informações dos tópicos anteriores e reforçar com os exercícios. Então, veja o Quadro 1, esquemático da abordagem conceitual realizada até o momento. Vamos lá.

Quadro 1 | Esquema de conteúdos estudados



Fonte: O autor (2015).

Leve em consideração que a primeira fase do projeto que os comerciantes do Litoral Sul solicitaram já está com as diretrizes e funcionalidades esclarecidas. A partir de agora, cabe à equipe da Think Now elaborar o algoritmo final que contemple a identificação dos processos e de todas as variáveis, constantes e os seus respectivos tipos de dados bem definidos e mapeados. Essa representação também deverá contemplar a elaboração do seu fluxograma.

Lembre-se de que a sequência a seguir retoma a importância da sua compreensão

acerca dos processos que o aplicativo executará para a elaboração do protótipo em algoritmos:

1. Entenda o processo através do levantamento de requisitos. Pergunte-se: o que os comerciantes precisam e o que solicitaram? Em que tempo?
2. Elabore um protótipo do aplicativo e apresente o seu algoritmo em linguagem natural; se preciso for, revise o material e utilize as dicas!
3. Identifique quais são as variáveis necessárias e o seu respectivo tipo de dado e elabore uma tabela com as especificações. Boas práticas.

Não pode faltar

Caro aluno, vamos em frente com a apresentação de situações que podem auxiliar no desenvolvimento da solução para a entrega do aplicativo. Vamos rever alguns conteúdos e ver o quanto conseguimos progredir com as práticas.

Primeiramente, vimos como se deu a concepção de algoritmos e citamos o de Euclides, como uma forma de manter as propriedades matemáticas da operação envolvida e ainda estabelecer a sequência de execução. Veja abaixo o exemplo em pseudocódigo.



Exemplificando

//seção de declarações

var

a, b, mdc, resto: inteiro;

//seção de comandos, lembre-se de que você pode solicitar ao usuário para inserir os dados de a e b (escreva e leia)

início

resto \leftarrow a mod b; //esta operação indica que o comando "mod" fará a divisão de "a" por "b" e apresentará o resto de divisão, que será atribuído à variável "resto".

enquanto resto \neq 0 faça //o comando de repetição enquanto, representado por while em linguagem c, indica que o programa fará uma sequência de

repetições enquanto a condição determinada for atendida.

$a \leftarrow b$;

$b \leftarrow \text{resto}$;

$\text{resto} \leftarrow a \bmod b$;

fimenquanto //toda estrutura iniciada, seja de seleção, repetição ou mesmo de decisão, deverá ser finalizada

$\text{mdc} \leftarrow b$;

fimalgoritmo.

Note que no algoritmo apresentado acima há uma estrutura de repetição sobre a qual você poderá aprofundar os estudos nas próximas unidades, porém, esta faz uma verificação do resto de divisão, e enquanto este for diferente de "0" (zero), implica na sequência de divisões imposta pelo algoritmo. Observe e tente você mesmo fazer a implementação do algoritmo em linguagem C.



Faça você mesmo

```
#include <stdio.h>

int main( void )

{int a,b,resto,mdc, dividendo, divisor;

printf( "Numero a: "); scanf( "%d", &a);
printf( "Numero b: "); scanf( "%d", &b);

dividendo = a;
divisor = b;

while ( resto !=0 ){

resto = dividendo % divisor;

dividendo = divisor;
divisor = resto; };

mdc = dividendo;

printf( "mdc [%d, %d] = %d ", a, b, mdc); return 0;}
```

Observe o uso dos operadores relacionais e sua importância na execução dos algoritmos, bem como dos operadores lógicos. As expressões matemáticas representadas pelas variáveis indicam claramente como acontece a alocação de valores e como o computador interpreta essas informações para que se possa imprimir, ou exibir, o resultado da operação.

Você também pode notar que, de acordo com o ambiente de programação em que se encontra, há uma forma específica de representação das variáveis necessárias ao programa. Por exemplo, em VisuAlg, os tipos de dados comuns são: caractere, lógico, inteiro e real. Veja a Tabela 6 com as características de cada um deles.

Tabela 6 | Tipos de dados em VisuAlg

| Tipos de dados em VisuAlg | Descrição | Tamanho |
|---------------------------|---|---------|
| Caractere | Corresponde a um caractere qualquer. | 1 byte |
| Lógico | Apresenta apenas valores para verdadeiro ou falso, sim ou não. | 1 byte |
| Inteiro | Compreende os números inteiros do intervalo, inclusive: -32768 a +32767. | 2 bytes |
| Real | Compreende todos os números de 2.9×10^{-39} a 1.7×10^{38} . | 6 bytes |

Fonte: Adaptado de Piva Jr. (2012, p. 90).

Uma característica interessante é que em VisuAlg a declaração das variáveis precede a seção de comandos e é indicada pela palavra “var”. Abaixo são apresentados os tipos de dados em linguagem C. A diferença, nesse sentido, é que na linguagem de programação C as variáveis podem ser declaradas em qualquer ponto do programa, e também é possível atribuir um valor a elas. No entanto, há duas recomendações que, por convenção, são sugeridas quanto à declaração das variáveis: antes das declarações de funções, ou seja, como variáveis globais; ou, logo no início do código da própria função, o que faz com que a variável seja local.

Tabela 7 | Tipos de dados em C

| Tipos de dados em C | Descrição | Tamanho |
|---------------------|--|---------|
| Char | Compreende de -127 a 126 caracteres. | 1 byte |
| int | Compreende de -32768 a +32767. | 2 bytes |
| Float | Compreende de $3.4\text{E}-38$ a $3.4\text{E}+38$ | 4 bytes |
| Double | Compreende de $1.7\text{E}-308$ a $1.7\text{E}308$ | 8 bytes |

Fonte: Adaptado de Piva Jr. (2012, p. 93).

O equivalente ao tipo de dado "real" em linguagem C são os tipos de dados "float" e também "double". O seu uso depende do tamanho do dado e de que forma deverá ser manipulado pelo programa. Outros tipos de dados em C também podem ser usados, tais como os que seguem na Tabela 8.

Tabela 8 | Tipos de dados em C e em linguagens comerciais

| Tipos de dados em C | Descrição | Tamanho |
|---------------------|--------------------------------|----------|
| signed char | -128 a 127 | 1 byte |
| unsigned char | 0 a 255 | 1 byte |
| signed int | -32768 a 32767 | 2 bytes |
| unsigned int | 0 a 65535 | 2 bytes |
| short int | -32768 a 32767 | 2 bytes |
| long int | -2.147.483.648 a 2.147.483.647 | 4 bytes |
| long double | 3.4xE-4932 a 1.1xE4932 | 10 bytes |

Fonte: Adaptado de Piva Jr. (2012, p. 93).

Retomando os conceitos, temos então a sequência apresentada para o desenvolvimento de algoritmos em que há, além da descrição do nome, seguido da declaração das variáveis, o início do algoritmo efetivamente. Veja mais um exemplo.

Em uma empresa de desenvolvimento de *software* há um módulo que realiza a cotação e autoriza a compra. O algoritmo dessa operação, que é realizada manualmente a partir do lançamento pelo usuário, considera dois orçamentos para a tomada de decisões. A regra é adotar o menor valor. Acompanhe a resolução abaixo:

```
//algoritmo: autorização para compra de software
```

```
//seção de declaração
```

```
var
```

```
software1, software2 : real
```

```
//seção de declaração
```

```
inicio
```

```
    escreval("Informe o valor do software 1:")
```

```
    leia (software1)
```

```
    escreval("Informe o valor do software 2:")
```

```
    leia (software2)
```

```
    se (software 1 > software 2) entao
```

escreval("Autorização de compra para o software 2 liberada!")

senao

escreval("Autorização de compra para o software 1 liberada!")

fimse

fimalgoritmo



Reflita

"Uma variável em um fluxograma representa uma área na memória onde se pode armazenar um valor". (SOUZA et al., 2011, p. 75)

Por enquanto, a abordagem focou na apresentação das formas de representação dos algoritmos, a linearização de expressões matemáticas, a importância dos operadores aritméticos, relacionais e lógicos. Porém, além desses, você também precisa utilizar as cadeias de caracteres, ou seja, aprender algumas funções de como realizar as atribuições de texto às variáveis e visualizar esse resultado.

A seguir, estão especificadas algumas funções utilizadas para que seja facilitada a manipulação de informações textuais, também chamadas de **expressões literais**. Também será comum ouvir a palavra de referência **string** para identificar o tipo de dado de que trata uma cadeia de caracteres.

Tabela 9 | Funções e procedimentos para as cadeias de caracteres

| Operação | Significado | Exemplos |
|----------------------------|---|--|
| <i>length(s)</i> | Fornecer como resultado o número de caracteres que compõem uma cadeia S. | $N \leftarrow \text{length}('Olá')$ O valor N é 3. |
| <i>concat(S1,S2)</i> | Une duas cadeias, a 2ª (S2) no final da 1ª (S1), formando uma nova cadeia. | $S \leftarrow \text{concat}('Bom', 'Dia')$ O valor de S é 'BomDia'. |
| <i>copy(S, ini, num)</i> | Retorna (copia) a uma nova cadeia de caracteres com os num elementos da cadeia S a partir da posição ini. | $S \leftarrow \text{copy}('Turbo Pascal', 7, 6)$ O valor de S é 'Pascal'. |
| <i>insert(S1, S2, ini)</i> | Insere uma nova cadeia (S1) na posição ini da S2, deslocando para a esquerda o resto da cadeia original (S2). | $S \leftarrow 'Turbo 7.0'$ $\text{insert}('Pascal', S, 7)$ O valor de S é 'Turbo Pascal 7.0' |

| | | |
|---------------|--|--|
| $pos(S1, S2)$ | Fornece como resultado a posição, na qual a cadeia S1 começa dentro da cadeia S2. Se a cadeia S1 não existir em S2, o resultado será zero. | $I \leftarrow pos('Pascal', 'Turbo Pascal')$ Aqui o valor de I é 7, mas $I \leftarrow pos('pascal', 'Turbo Pascal')$ O valor de I é 0. Existe diferença entre maiúsculas e minúsculas nessa função. |
|---------------|--|--|

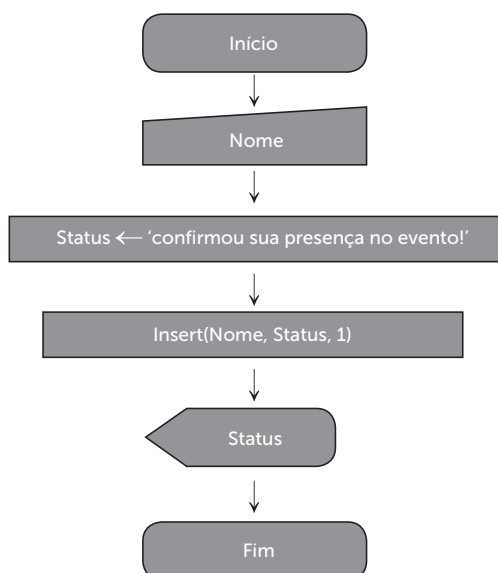
Fonte: Souza et al. (2011, p. 104).

Note que os caracteres em algoritmos, por convenção, são acompanhados de apóstrofes. Cada caractere corresponde a uma única letra ou símbolo. Essa forma de representação ('x') evita que um caractere seja confundido com o nome da variável x, por exemplo. Encontra-se nessa categoria de representação também o espaço em branco (' '). Por convenção, o tamanho da cadeia de caracteres é limitado a 255. Você sabia que um nome de variável pode ter no máximo 63 caracteres? Pois bem, identifique as variáveis e constantes com nomes objetivos que realmente representem ou remetam à informação que será atribuída.



Assimile

Observe a representação no fluxograma de uma sequência lógica para a manipulação de uma cadeia de caracteres que informe um Nome e exiba o Status: "Valter confirmou presença no evento".



Além da função *insert*, você pode testar os resultados obtidos com o uso das demais funções. Utilize a tabela acima e este fluxograma como

exemplos, e inicie os testes com outras variáveis, cenários e objetivos. Tais funções podem ser muito úteis no desenvolvimento do protótipo da aplicação que você apresentará junto à Think Now.



Pesquise mais

Leia o artigo do *site* Tecmundo e veja os sete principais algoritmos que foram selecionados e cabem no contexto descrito. Disponível em: <<http://www.tecmundo.com.br/tecnologia/56148-forca-invisivel-7-tipos-algoritmos-dominam-nosso-mundo.htm>>. Acesso em: 04 mar. 2015.

Sem medo de errar

A proposta oferecida pela Think Now, com a especificação das ações iniciais que o aplicativo deverá executar, está descrita no algoritmo em linguagem natural.

Algoritmo de ações iniciais do aplicativo

1. Início
2. Cadastrar estabelecimentos comerciais.
3. Logar usuário por cadastro em formulário próprio ou por redes sociais.
4. Solicitar que o usuário selecione: categoria (gastronomia ou hotelaria), cidade.
5. Exibir, de acordo com a seleção, os estabelecimentos cadastrados.
6. Disponibilizar a opção ao usuário para “conhecer” ou “agendar”.
7. Se o usuário simplesmente optar por conhecer, então,
 - 7a. Direcionar para o website do estabelecimento.
8. Senão
 - 8a. Exibir calendário e horários disponíveis.
 - 8b. Solicitar o preenchimento dos campos pelo usuário: local, data e hora.
 - 8c. Exibir a mensagem de confirmação: “Aguarde o contato de confirmação de reserva”.
 - 8d. Abrir a pesquisa de satisfação.

8e. Solicitar a resposta do usuário.

8f. Calcular a média de acessos e respostas à pesquisa por estabelecimento cadastrado.

8g. Enviar mensagem para o e-mail do estabelecimento com cópia ao usuário final com a solicitação de reserva e a resposta da pesquisa.

9. Encerra estrutura de seleção.

10. Retornar à página inicial.



Atenção!

A partir dessas definições, é possível identificar quais serão as variáveis necessárias para a sua implementação. Façamos o exercício de identificação das variáveis que deverão ser declaradas para que os processos abaixo identificados sejam realizados.



Lembre-se

| Variáveis | Tipo de dado em VisuAlg | Descrição |
|--|-------------------------|---|
| local, data, hora, G, H, msg_confirmacao (constante) | caractere | G – gastronomia H – hotelaria |
| satisfacao, conhecer, agenda | lógico | Variáveis aceitam 1 para sim e 0 para não |
| cont_satisfacao, qtd_sim, qtd_nao | inteiro | A cada indicação de satisfeito ('1') ou insatisfeito ('0') contar e armazenar |
| Media | real | Efetua o cálculo da média de acessos |

Avançando na prática

Pratique mais

Instrução

Desafiamos você a praticar o que aprendeu transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois as compare com a de seus colegas.

| Declaração de variáveis e constantes | |
|--------------------------------------|--|
| 1. Competência de fundamento de área | Conhecer os princípios e conceitos que envolvem o aprendizado em construção de algoritmos e programação, e a sua importância para o universo do desenvolvimento de sistemas. |
| 2. Objetivos de aprendizagem | <ul style="list-style-type: none"> • Conhecer o que são, como se aplicam e a quem se destina a elaboração dos algoritmos; • Conhecer como ocorre o raciocínio lógico computacional. |
| 3. Conteúdos relacionados | <ul style="list-style-type: none"> • Tipos de dados, variáveis e constantes; • Declaração de variáveis e constantes. |
| 4. Descrição da SP | Um triatleta, em fase de readaptação, está treinando para competir nas Olimpíadas. Sabendo que a prova pode ter no máximo 51,5 km, ele precisa atingir bom desempenho para realizar os 1,5 km de natação, 40 km de ciclismo e 10 km de corrida. O atleta busca um dispositivo que indique os tempos ideais para superar cada fase com tranquilidade e melhorar os seus tempos. Faça um algoritmo que receba um número, indique a quantidade de minutos mínima para cumprir a prova e também considere o intervalo, que seria a indicação do tempo ideal. O algoritmo deve verificar se esse número está no intervalo entre 1 e 20 min. |
| 5. Resolução da SP | <ul style="list-style-type: none"> • Pseudocódigo <p>algoritmo "Cálculo deTempo"</p> <p>// Função : Faça um algoritmo que receba um número que indica a quantidade de minutos mínima para cumprir a prova e, também, considere o intervalo, que seria a indicação do tempo ideal. O algoritmo deve verificar se esse número está no intervalo entre 1 e 20 min.// Autor :xxx</p> <p>// Data : xxx</p> <p>// Seção de Declarações</p> <p>var</p> <p>numero: inteiro</p> <p>inicio</p> <p>// Seção de Comandos</p> <p>escreva("Digite um número: ")</p> <p>leia(numero)</p> <p>se numero >= 1 entao</p> <p> se numero <= 20 entao</p> <p> escreval("O número está no intervalo entre 1 e 20")</p> <p> senao</p> <p> escreval("O número não está no intervalo entre 1 e 20")</p> <p> fimse</p> <p>senao</p> <p> escreval("O número não está no intervalo entre 1 e 20")</p> <p> fimse</p> <p>finalgoritmo</p> |



Lembre-se

Todo algoritmo precisa ser definido como exemplificado no exercício. Note que há as seções de declaração de variáveis; de inserção dos

comandos e funções. É preciso indicar o início e o fim do algoritmo. A palavra descrita como "var" indica as variáveis que serão utilizadas para realizar as operações e estas necessariamente são declaradas. Os comandos "leia" e "escreva" representam, respectivamente, as entradas e saídas de dados no algoritmo.



Faça você mesmo

- em linguagem C

```
#include<stdio.h>

main(){

    int num;

    printf("Informe um numero: ");

    scanf("%d",&num);

    if ((num > 1) & ( num < 20)){

        printf("O valor: %d, está no intervalo compreendido entre 1 e 20.", num);
    }

    else{

        printf("Informe valor que esteja no intervalo de 1 a 20.");  }

    fflush(stdin);getch(); }
```

Observe, no código-fonte acima, o uso do operador lógico "&&", que indica que a condição é inclusiva (e), ou seja, precisa atender às duas condições impostas na operação.

Você também já pode se familiarizar com os operadores relacionais: (>) maior, (<) menor, (>=) maior ou igual, (<=) menor ou igual e (=) igual.

Faça valer a pena

1. Elabore o fluxograma do algoritmo do protótipo a ser desenvolvido do aplicativo para os comerciantes do Litoral Sul.

2. Complete a tabela com os respectivos tipos de dados que precisam ser associados aos conceitos. Assinale a alternativa correta.

| Tipos de dados em VisuAlg | Descrição |
|---------------------------|---|
| | Corresponde a um caractere qualquer. |
| | Apresenta apenas valores para verdadeiro ou falso, sim ou não. |
| | Compreende os números inteiros do intervalo, inclusive: -32768 a +32767. |
| | Compreende todos os números de 2.9×10^{-39} a 1.7×10^{38} . |

- a. ☐ real, lógico, inteiro e caractere.
- b. ☐ lógico, real, integer e char.
- c. ☐ real, char, int e integer.
- d. ☐ string, booleano, inteiro e real.
- e. ☐ caractere, lógico, inteiro e real.

3. Escreva um algoritmo que armazene o valor informado pelo usuário em uma variável A e um outro valor em uma variável B. O algoritmo precisa apresentar a lógica necessária para realizar a inversão dos valores da variável A para B e de B para A. Por fim, exiba o valor final das variáveis.

4. Para a expressão "correcao <= 1.2 * deposito", identifique qual é o melhor tipo de dado a ser relacionado a estas variáveis. Assinale a alternativa correta.

- a. ☐ inteiro
- b. ☐ real
- c. ☐ long
- d. ☐ lógico
- e. ☐ caractere

5. Com base no exercício 3 desta lista, elabore o fluxograma do algoritmo proposto e assinale a alternativa que melhor representa as variáveis utilizadas nesse algoritmo.

- a. ☐ menos, mais e altera.
- b. ☐ qtd1, qtd2 e qtd3.

- c. () a, b e aux.
- d. () raiz1, raiz2 e delta.
- e. () num1, num2 e num3.

6. A considerar ano com 365 dias e mês com 30 dias, elabore um algoritmo que leia a idade de uma pessoa e escreva a sua idade em dias e em meses. A expressão "escreval("A sua idade em dias é de:", dias, " dias")" apresenta qual identificação direta de variável?

- a. () real.
- b. () dias.
- c. () dia.
- d. () inteiro.
- e. () idade.

7. Elabore um algoritmo que exiba as unidades, dezenas e centenas de um número qualquer fornecido pelo usuário. Analise o algoritmo e assinale a alternativa correta.

- a. () não é possível extrair o resto de divisão do número quando o tipo de dado for inteiro.
- b. () mod é um comando que exibe o resto de divisão.
- c. () div exibe resto de divisão.
- d. () mod e div não são utilizados em operações com tipos de dados inteiros.
- e. () não é possível exibir unidades, dezenas e centenas de um número.

Referências

EVARISTO, Jaime. **Aprendendo a programar programando na linguagem C:** para iniciantes. 3. ed. Maceió: Instituto de Computação Universidade Federal de Alagoas, 2002.

PIVA JR., Dilermando et al. **Algoritmos e programação de computadores.** Rio de Janeiro: Campus, 2012.

SOUZA, Marco Antonio Furlan de et al. **Algoritmos e lógica de programação.** 2. ed. São Paulo: Cengage Learning, 2011.

ZIVIANI, Nivio. **Projeto de Algoritmos:** com implementações em Java e C++. São Paulo: Cengage Learning, 2011.

Referências bibliográficas complementares:

1. EVARISTO, Jaime. **Aprendendo a Programar:** programando na linguagem C para iniciantes. Rio de Janeiro: Ed. Book Express, 2001.

2. SOUZA, Marco. **Algoritmos e lógica de programação.** 2. ed. São Paulo: Cengage Learning, 2011.

3. MANZANO, José A. N. G. **Algoritmos:** Lógica para Desenvolvimento de Programação de Computadores. 24. ed. São Paulo: Érica, 2010.

4. AGUILAR, Luis J. **Programação em C++.** 2. ed. Porto Alegre: McGraw Hill, 2008.

5. SZWARCFITER, Jayme Luiz; MARKEZON, Lilian. **Estruturas de dados e seus algoritmos.** 3. ed. Rio de Janeiro: LTC – Livros Técnicos e Científicos, 2010.

ESTRUTURAS DE DECISÃO E SELEÇÃO

Convite ao estudo

Ao iniciar esta unidade de ensino, você aluno pode até se questionar qual será mesmo o objetivo de aprender como fazer um algoritmo e além disso, como escolher uma estrutura de programação que atenda às necessidades do problema que terá de solucionar através do desenvolvimento de um algoritmo, um programa ou até mesmo um aplicativo? Pois é, agora você entenderá que para cada tipo de problema há uma estrutura específica que poderá ser aplicada. Neste contexto, entenda quais são e o que representam estas estruturas de programação que são classificadas em sequenciais, de decisão ou de repetição. Leia as respectivas descrições de cada uma delas:

- Instruções sequenciais: são aquelas estudadas com maior ênfase na Unidade de Ensino 1. Relembre que os problemas resolvidos apresentavam como solução, ações que dependiam fundamentalmente do passo anterior e, desde que a lógica estivesse correta, não havia a necessidade de tomada de decisão e sim, apenas a execução de ações sequenciais.
- Instruções de decisão: as instruções de decisão indicam que o programa deverá verificar se há alguma condição que valide ou invalide a operação que será realizada. Nesta unidade, você verá estas estruturas com maior ênfase. Estão inclusas as funções "se", "senão" e "se" encadeados.
- Instruções repetição: estas instruções de repetição, como o próprio nome diz, indicam que o programa deverá executar uma determinada operação por mais de uma vez e, por vezes, em escala.

De acordo com Böhm e Jacopini (1966, apud Souza, 2013, p. 126) as estruturas sequenciais, de decisão e de repetição são denominadas estruturas

primitivas de programação: “permitem a descrição de qualquer algoritmo que seja compatível, sendo implementável em um computador. Em resumo, qualquer programa de computador pode ser escrito combinando-se esses três tipos de estruturas.” É importante que você saiba inserir esse teste o algoritmo, utilizando uma estrutura de decisão, que, neste caso, a função “SE” ou “if” atende a este requisito. Para que você possa resolver esta situação, você precisa desenvolver algumas competências e habilidades, sendo assim, destaca-se a competência geral:

Competência geral: conhecer os princípios e conceitos que envolvem o aprendizado em construção de algoritmos e programação e, a sua importância para o universo do desenvolvimento de sistemas.

O que é preciso saber a partir de agora para que você esteja apto a desenvolver a solução de problemas que envolvam testes e verificação de condições? Conheça os objetivos que pretendemos atingir com o estudo desta unidade de ensino:

Objetivos Específicos

- Conhecer os conceitos e aplicações das estruturas de decisão “SE”.
- Conhecer como fazer uma estrutura condicional simples.
- Conhecer e aplicar as estruturas condicionais compostas.
- Conhecer e aplicar estruturas condicionais encadeadas.

Diante do conteúdo que você conhecerá, será possível retomar a implementação que iniciou incrementando elementos condicionais ao protótipo que será apresentado para os comerciantes do Litoral Sul. Relembrando, deverá ser elaborado um algoritmo que contemple as seguintes entregas:

- Após realizar a consulta, o usuário é direcionado a um painel com os ícones que representam o seu índice de satisfação: insatisfeito, satisfeito ou plenamente satisfeito. O prazo para desenvolvimento deste é de 6 meses.
- Um algoritmo que apresenta o índice de satisfação do usuário quanto a facilidade de navegação e uso do aplicativo a partir da seguinte regra: se ele estiver satisfeito, o sistema simplesmente exibe uma mensagem de agradecimento, senão, pede ao usuário que faça uma sugestão.

Para realizar esse procedimento, primeiramente o desenvolvedor identificará:

- a. variáveis e constantes necessárias ao desenvolvimento do algoritmo; os tipos de dados e estrutura de decisão que precisará contemplar para esta solução.
- b. Implementar esta solução através do VisuAlg.
- c. Implementar esta solução através do Dev C++ em linguagem de programação C.
- d. Incrementar o algoritmo com as estruturas estudadas.

Desde já, bons estudos!

Seção 2.1

Instruções primitivas: entrada de dados, atribuição e saída

Diálogo aberto

No cotidiano, muitas situações requerem uma tomada de decisão. Quando se traduz uma regra de negócio, ou seja, uma condição que precisa ser atendida, é preciso estabelecer um processo computacional que viabilize a realização desse teste. Então, como fazer com que essa regra seja mantida e, principalmente, cumprida, com a sua automatização?

Por exemplo, para o protótipo do aplicativo de divulgação de hotelaria e gastronomia solicitado pelos comerciantes do Litoral Sul, é preciso cumprir a regra de negócio que contabiliza o nível de satisfação do usuário. Para tal, você poderá resolver essa situação da seguinte forma:

- Escrever o algoritmo em linguagem natural com as respectivas necessidades de testes e verificações.
- Transcrever esse algoritmo em uma linguagem computacional. A princípio, pode ser elaborado no VisuAlg.

No entanto, alguns conceitos importantes terão de ser aplicados. De acordo com Souza (2013, p. 127), estruturas de decisão “são estruturas que permitem a tomada de decisão sobre qual o caminho a ser escolhido, de acordo com o resultado de uma expressão lógica”. Sendo assim, classifica-as em três formas fundamentais: SE-ENTÃO, SE-ENTÃO-SENÃO e CASO.

Você pode notar que as palavras acima (Se, Então, Senão) indicam que este é um comando condicional. Isso significa que as estruturas de decisão “Se-Então” e “Se-Então-Senão” podem representar rotinas que o sistema deverá executar para cada uma das expressões lógicas determinadas.

Já a palavra “Caso” indica que, a partir da escolha de uma opção pelo usuário, o programa executará comandos específicos para aquela determinada opção. Segundo Piva Junior (2012, p. 152), elas podem evitar erros de programação: “[...] o uso de

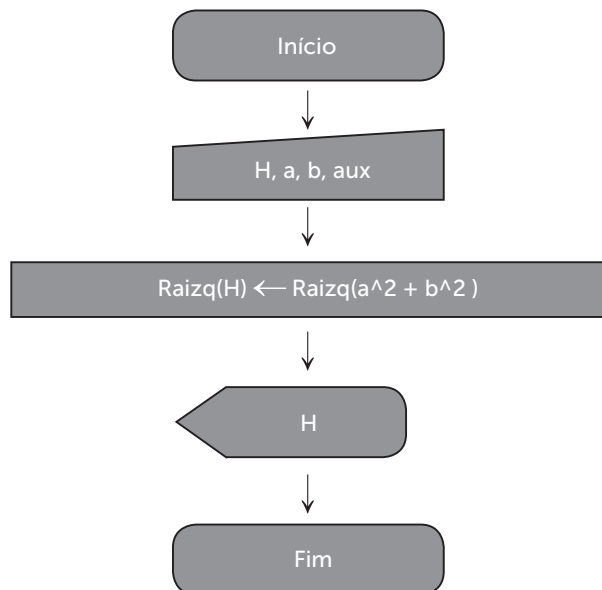
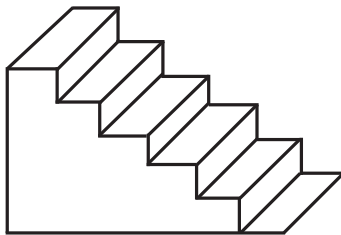
condições para permitir a escolha de executar ou não um trecho de programa é muito utilizado, principalmente quando precisamos incluir no programa condições de controle, para evitar situações não permitidas que podem resultar em erros”.

A seguir, as três formas fundamentais serão conceituadas e contextualizadas para que você exercite e pratique a resolução de exercícios com essas estruturas.

Siga em frente!

Não pode faltar

Suponha que você foi contratado para auxiliar um escritório de projetos de construção civil, e precisam determinar o padrão e o tamanho de uma escada residencial. Considere que a parede tem 8 metros de altura e a base poderá ocupar no máximo 6 metros, a considerar o espaço e a disposição dos demais cômodos, desenvolva o fluxograma que representa esse cálculo. A recomendação é que você utilize o Teorema de Pitágoras para realizar esse cálculo, pois neste caso, a regra é: “a soma dos quadrados dos **catetos** é igual ao quadrado da **hipotenusa**”. Sendo assim, temos a fórmula: $a^2 + b^2 = c^2$. Mas observe como pode ser estruturado sequencialmente:



Já quando se tratam de estruturas de decisão, à esta sequência é agregado um teste de verificação de uma condição, para somente após este teste, seguir com o processamento das operações.

Apresenta basicamente uma operação para cada resultado seja ele verdadeiro (true) ou falso (false).

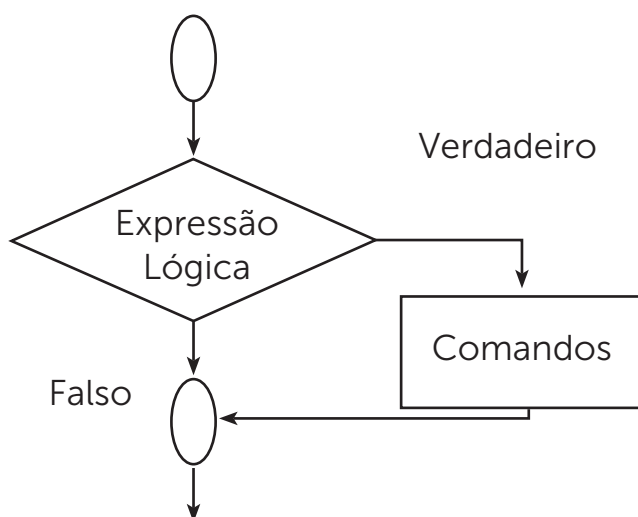
Observe o exemplo: Agora, vamos verificar primeiro se podemos utilizar o Teorema de Pitágoras. Para isso, a soma dos quadrados dos catetos "a" e "b" têm que resultar em um ângulo de 90°.

Mas afinal, como verificar se este é o caso de um triângulo reto e que, portanto, pode ser resolvido através do Teorema de Pitágoras?

Uma expressão lógica sempre resulta em um valor verdadeiro ou falso, como mencionado anteriormente. O que fica evidenciado nesse tipo de estrutura é a necessidade de se pensar nas respostas tanto para o caso de a operação resultar em verdadeira quanto para o caso de ser falsa. Isso implica o desenvolvimento de ações que contemplem as operações que o sistema deverá desenvolver para ambos casos.

Observe a figura que representa a lógica computacional envolvida em estruturas de decisão:

Figura 2 | Lógica de uma estrutura de decisão **Se-Então**



Fonte: Adaptado de Souza (2013, p. 127).

A Figura 2 mostra que a tomada de decisão em um sistema computacional depende de uma expressão lógica que indique o teste a realizar e, a partir do resultado, a decisão que será tomada caso seja verdadeiro ou falso o resultado.



Assimile

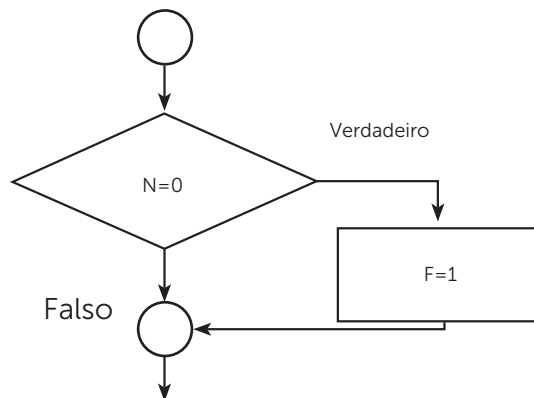
A lógica corresponde basicamente ao seguinte (PIVA JR. et al., 2012):

```

se condição então
    comando 1
fimse
  
```

Então, por exemplo, o cálculo do fatorial de um número seria:

Figura 3 | Fatorial de zero



Fonte: Adaptado de Piva Junior et al. (2012, p. 155).

Você sabia que o fatorial de zero é 1?



Refleta

Não há uma fórmula para o cálculo do fatorial. [...] A condição de teste para saber se atribui o valor 1 ao resultado e então passar para o cálculo do fatorial pela fórmula é dada pela questão: *n* é igual a 1 (um)? [...] se *n* for igual a 1, então, o comando fará a atribuição à variável fatorial. (PIVA JR. et al., 2012, p. 155)

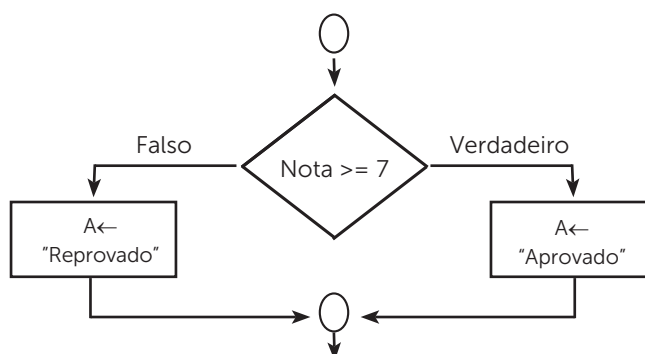


Pesquise mais

Fatorial: quando se pretende calcular o fatorial de um número, quer dizer que este apresentará o produto de todos os seus antecessores. Ex.: $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$. Disponível em: <<http://www.matematicadidatica.com.br/Fatorial.aspx>>. Acesso em: 11 maio 2015.

A Figura 4 indica que há a inserção de um valor para verificação. Neste caso, o valor da nota que é solicitada. Em seguida, se a nota for maior ou igual a 7,0 (sete), significa que a operação a realizar é a que contém a operação para o valor verdadeiro, pois a nota inserida obedece à restrição imposta. Senão, significa que a nota inserida é menor do que 7,0 (sete) e, portanto, o programa deve executar os comandos para a operação quando esta é falsa, ou seja, apresentará o resultado para o caso de o valor inserido não atender à condição descrita no processo de verificação, através da expressão lógica “Nota \geq 7”.

Figura 4 | Verifica nota para aprovação ou reprovação



Fonte: Adaptado de Piva Junior et al., (2012, p. 155).

Observe que, como no exemplo, a condição será sempre representada por uma expressão lógica. Veja também como deve ser representado o pseudocódigo da estrutura **se-então-senão**:

se condição **então**

comando 1

Este primeiro comando sempre será referente à operação para resultado do teste como "verdadeiro".

senão

comando 2

O segundo comando sempre será referente à operação para resultado do teste como "falso".

fimse

Note também a endentação. Isto significa que o comando 1 é executado para atender à condição expressamente representada na estrutura "se", assim como o comando 2 pertence à condição contrária (falso). É muito importante essa representação em função da organização do código e da identificação de referência do comando.



Reflita

As opções: então e senão são excludentes. Isto é, o algoritmo não executará as duas sequências de comandos para um mesmo valor da condição (PIVA JR. et al., 2012, p. 154).

Conceituando também a estrutura de decisão CASO, obtém-se para esta a seguinte representação sintática:

escolha <apresentação das opções>

caso 1 <constante 1>

<sequência de comandos a executar para esta opção>

caso 2 <constante 2>

<sequência de comandos a executar para esta opção>

caso n <constante n>

<sequência de comandos a executar para esta opção>

outrocaso: < demais sequências de comandos>

Vamos compreender melhor esta estrutura?

Trata-se basicamente de uma estrutura de decisão que permite a escolha de uma delas pelo usuário. Com isso, é necessário apresentá-las primeiro (**escolha** ou "**switch**" em C). A seguir, a partir da seleção da opção (caso 1, caso 2), serão executados os comandos inerentes àquela opção, e o encerramento da estrutura se dá através da inserção do comando "**outrocaso**", para que sejam apresentadas as instruções de comandos para o caso do usuário poder indicar, ou mesmo não aceitar, as opções que lhe foram apresentadas. Veja o exemplo em VisuAlg:

algoritmo "calculadora simples" //seção de declarações

var

x, y: **real**

opcao: **caractere**

inicio //seção de comandos

Escreval ("Digite dois números e informe se deseja (S)omar, (M)ultiplicar, (Sub)trair ou (D)ividir:");

Leia (x, y, opcao)

escolha opcao

caso "S"

Escreval (x, opcao, y, "=", x+y)

caso "M"

Escreval (x, opcao, y, "=", x * y)

caso "Sub"

Escreval (x, opcao, y, "=", x - y)

caso "D"

se y <= 0 **entao**

Escreval ("Informe número diferente de 0 (zero)!")

senao

Escreval (x, opcao, y, "=", x / y)

fimse

outrocaso

Escreval ("Opção inválida!")

fimescolha

fimalgoritmo

Agora que você já conhece as três principais formas das estruturas de decisão, vamos praticar um pouco!



Exemplificando

A turma "B" do curso de exatas "EPP" está fazendo aulas de revisão de matemática, no entanto, eles precisam implementar os conceitos aprendidos em um ambiente computacional. Foi lançado o seguinte desafio: elabore um algoritmo que identifique o tipo de triângulo a partir da inserção das medidas dos lados em VisuAlg:

algoritmo "classificação triângulo"

var

a, b, c: inteiro

inicio

escreval ("Digite os lados do triângulo que deseja classificar")

leia (a, b, c)

se (a < b+c) e (b < a+c) e (c < a+b) entao

 escreval ("Para as medidas:", a, " ", b, " ", c, " indicadas, é possível inferir que estas representam um triângulo:")

se (a = b) e (a = c) entao

 escreval ("Triângulo informado é: EQUILÁTERO.")

senao

se (a = b) ou (a = c) ou (b= c) entao

escreval ("Triângulo informado é: ISÓSCELES.")

senao

escreval ("Triângulo informado é: ESCALENO.")

fimse

fimse

senao

escreval ("As medidas informadas não representam um triângulo!")

fimse

fimalgoritmo

Observe que este exemplo apresenta inclusive o uso dos operadores lógicos "e" e "ou".

Com isso você conheceu o uso do "**se-entao**", "**se-entao-senao**"! Aproveite para treinar um pouco mais a linguagem de programação C e bons estudos!



Refleta

Quando a condição "se" indica o uso do operador lógico "e", significa que todas as condições impostas precisam ser atendidas para que seja verdadeiro o resultado.

Quando a condição "se" indica o uso do operador lógico "ou", significa que se apenas uma das condições impostas for atendida, é possível considerar o valor como verdadeiro.

Transcreva o algoritmo em uma linguagem de programação (preferencialmente C).

Orienta-se pelo exemplo a seguir para praticar um pouco a linguagem:



Faça você mesmo

```
#include <stdio.h>

void main ()

int a, b, c;

printf("Classificação do triângulo: informe a medida dos lados:");

scanf("%d %d %d", &a, &b, &c);

if (a< b + c && b< a +c && c < a + b)

    {
        printf("\n\n Dadas as medidas: %d, %d, %d, temos um
triângulo", a, b, c)

        if( a == b && a ==c)

            {

                printf("Este é um triângulo EQUILÁTERO! \n");

            else

                if ( a==b || a == c || b ==c)

                    printf("Este é um triângulo ISÓSCELES!
\n");

            else

                printf("Este é um triângulo ESCALENO! \n");

        }

    else

        printf("\n\n As medidas fornecidas, %d,%d,%d não formam
um triângulo", a, b, c);

    }

    fflush (stdin);

}
```



Pesquise mais

Material de aulas e tutoriais. Disponível em: <<http://www.ic.unicamp.br/~mc102/algoritmos.html>>. Acesso em: 12 maio 2015.



Lembre-se

Estamos trabalhando na solução para a proposta de algoritmo que deverá ser entregue aos comerciantes do Litoral Sul. Estes esperam que o aplicativo a partir de agora ofereça ao usuário a possibilidade de indicar efetivamente a sua avaliação de uso do aplicativo.



Atenção!

Para tal, ele deverá escolher o tipo de serviço utilizado e o sistema mostrará as opções e o índice de satisfação, conforme recomendação abaixo.

1. Inicia.
2. Opção para escolher tipo de serviço: hotelaria ou gastronomia.
3. Solicita para cada uma delas que o usuário responda ao índice de satisfação.
4. Se o usuário indicar satisfeito, exibe mensagem de agradecimento.
5. Senão, exibe mensagem de inserção de sugestão.
6. Caso o usuário informe uma opção inexistente, solicita que informe opção existente.
7. Encerra procedimento.

Agora tente você mesmo resolver em VisuAlg. Está fácil, vamos lá!

algoritmo "Litoral Sul Fase 2.1"

// Função : apresentar opções e contabilizar acessos. Gerar média de acessos.

// Autor : JJJ

```
// Data : 12/05/2015
// Seção de Declarações
var
    acessos, opcao2, G, H, satisfeito, insatisfeito: inteiro
    opcao1: caractere
inicio
// Seção de Comandos
    escreval("Olá, vamos avaliar este serviço de busca!")
    escreval(" Escolha o tipo de serviço pesquisado: G - gastronomia ou H- hotelaria")
    leia (opcao1)
    escolha opcao1
    caso "H"
        escreval ("**Índice de satisfação de usabilidade do aplicativo**")
        escreval (" Digite: 1 - satisfeito ou 2 - insatisfeito")
        leia (opcao2)
        se (opcao2 = 1) entao
            escreval ("Obrigado pela preferência!")
        senao
            escreval ("Faça a sua sugestão!")
        fimse
    caso "G"
        escreval ("**Índice de satisfação de usabilidade do aplicativo**")
        escreval (" Digite: 1 - satisfeito ou 2 - insatisfeito")
        leia (opcao2)
        se (opcao2 = 1) entao
            escreval ("Obrigado pela preferência!")
```

senao

escreval ("Faça a sua sugestão!")

fimse

outrocaso

escreval ("Informe opção existente!")

fimescolha

fimalgoritmo

Avançando na prática

| Pratique mais! | |
|--|--|
| Instrução Desafiamos você a praticar o que aprendeu transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois compare-as com as de seus colegas. | |
| Instruções primitivas: entrada de dados, atribuição e saída | |
| 1. Competência de fundamentos de área | Conhecer os princípios e conceitos que envolvem o aprendizado em construção de algoritmos e programação e a sua importância para o universo do desenvolvimento de sistemas. |
| 2. Objetivos de aprendizagem | <ul style="list-style-type: none"> • Conhecer os conceitos e aplicações das estruturas de decisão "SE". • Conhecer como fazer uma estrutura condicional simples. • Conhecer e aplicar as estruturas condicionais compostas. • Conhecer e aplicar estruturas condicionais encadeadas. |
| 3. Conteúdos relacionados | <ul style="list-style-type: none"> • Instruções primitivas: entrada de dados, atribuição e saída • Estrutura condicional simples |
| 4. Descrição da SP | Uma indústria de luminárias precisa mensurar, dentro de três horas, em qual dos setores houve maior nível de produção. Para tal, na troca de turno, cada líder deve lançar respectivamente os valores das suas últimas três horas de trabalho. Sendo assim, desenvolva um algoritmo que identifique e mostre o maior valor dentre os lançados. Este valor servirá de parâmetro para identificar a maior produtividade dos três turnos. |
| 5. Resolução da SP | Em VisuAlg testar a seguinte solução: <pre> var a, b, c, maior: inteiro inicio // Seção de Comandos escreval ("Informe o primeiro número") leia (a) escreval ("Informe o segundo número") </pre> |

| | |
|--|---|
| | leia (b) escreval ("Informe o terceiro número") leia (c) se (a >= b) entao maior <- a senao maior <- b fimse se (maior <= c) entao maior <- c fimse escreva ("O maior número é: ", maior) finalgoritmo. |
|--|---|



Lembre-se

O uso dos operadores lógicos e relacionais o auxiliará na descrição da expressão que melhor representará a condição que o sistema deverá executar. Quanto mais praticar o desenvolvimento de algoritmos, melhor!

Faça valer a pena!

1. Considerado relativamente de fácil implementação, desenvolva um algoritmo que apresente uma saudação ao usuário de acordo com o seu sexo: feminino ou masculino.

A partir deste breve enunciado, assinale a alternativa que representa a declaração das variáveis:

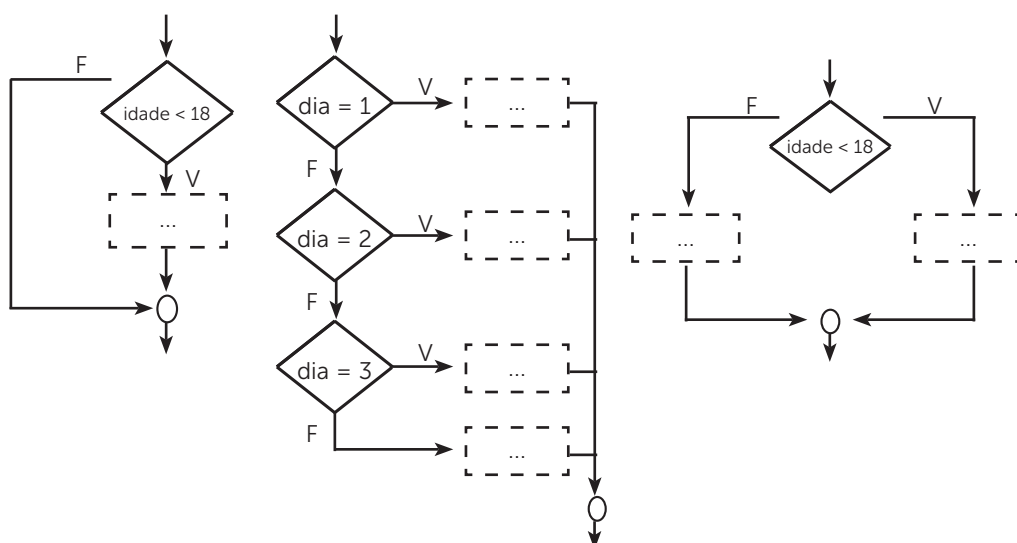
- a. () N, S: caracter
- b. () escreval ("Informe o seu nome")
- c. () escreva ("Seja Bem-Vinda ", N) fimse
- d. () inicio // Seção de Comandos
- e. () se (S = "masculino") entao

2. São consideradas estruturas de decisão ou seleção o que está indicado em qual das alternativas?

- a. () a, se, então.
- b. () caso, for, para.
- c. () se-então, se-então-senão, se-senão, caso.
- d. () while, do/while, if.
- e. () if/else, for, se.

3. Desenvolva o algoritmo para resolver a situação apresentada no exercício 1.

4. Assinale a alternativa que apresenta a sequência descrita nas figuras:



Disponível em: <<http://www.ramon.pro.br/comandos-portugol-vs-fluxograma/>>. Acesso em: 11 maio 2015.

- a. () se-senão, caso, se-então.
- b. () se-então, caso, se-então-senão.
- c. () caso, se, se-então.
- d. () se-senão, se-então-senão.
- e. () se, caso, então.

5. Leia as afirmações e assinale a alternativa correspondente:

- I. Estruturas de decisão são estruturas que permitem a tomada de decisão sobre qual o caminho a ser escolhido, de acordo com o resultado de uma expressão lógica.
- II. Classificam-se em duas formas fundamentais: SE-ENTÃO-SENÃO e CASO.
- III. Quando a condição "se" indica o uso do operador lógico "ou", significa

que se apenas uma das condições impostas for atendida, é possível considerar o valor como verdadeiro.

- a. () V, F e V.
- b. () F, F e F.
- c. () V, V e F.
- d. () V, F e F.
- e. () V, V e V.

6. Explique a diferença de uso dos operadores lógicos “e” e “ou”.

7. Conceitue as estruturas de decisão se-senão e se-então-senão.

Seção 2.2

Estrutura condicional simples

Diálogo aberto

Conhecer e saber aplicar as estruturas de decisão significa ir além de simplesmente ter contato com elas. Implica identificar qual delas utilizar em face da situação que precisará resolver. Lembre-se, estamos falando das estruturas de decisão ou seleção: se-então, se-então-senão ou caso. A partir disso, você já pode aprofundar os seus conhecimentos na estrutura de decisão se-então e, no decorrer desta unidade, você terá a oportunidade de ter mais contato com as demais.

Neste sentido, retome o contexto da estrutura se-então. Observe o exemplo abaixo que pretende inserir uma verificação da idade de uma pessoa para que apresente uma resposta padrão, de acordo com a condição inserida. O uso do se-então já atende à necessidade de resolução da situação proposta. Vamos então para as instruções deste teste:

se (idade >= 65) **entao**

escreval ("Idade permitida para solicitação de recursos previdenciários!")

fimse

Observe que a **semântica** é relativamente simples do ponto de vista lógico. Mas, e para inserir mais de uma verificação com o uso de um operador lógico, como fazer? Para este mesmo exemplo, suponha que, além de ter idade igual ou superior a 65 (sessenta e cinco) anos, a pessoa também precisa ser do sexo feminino. Veja a sintaxe como fica:

se (idade >= 65 **e** sexo = feminino) **entao**

escreval ("Idade permitida para solicitação de recursos previdenciários!")

fimse

Agora, veja outro exemplo com o operador lógico “ou”:

se (idade >= 65 **ou** tempoTrabalho>=30) **entao**

escreval (“Idade permitida para solicitação de recursos previdenciários!”)

fimse

Com isso, você pode notar que é importante saber trabalhar com os operadores lógicos e relacionais. Estes são elementos-chave para as estruturas de decisão. Agora que você já retomou alguns elementos básicos, pode aplicá-los para desenvolver o protótipo do aplicativo da Think Now. Bons estudos e práticas a você!

Não pode faltar

Também conhecida como estrutura condicional simples, se-então pode ser utilizada para desenvolver soluções para programas que necessitem, além de realizar cálculos, atribuições, receber valores e exibir resultados ou operações, indicar condições e testá-las para a tomada de decisão.



Assimile

Condição é uma expressão lógica que, sendo verdadeira, então o comando 1 é executado e, sendo falsa, o comando 2 é executado. Somente após executado um dos comandos o controle passará para o próximo comando, após o fim do comando condicional. (PIVA JR. et al., 2012, p. 163)

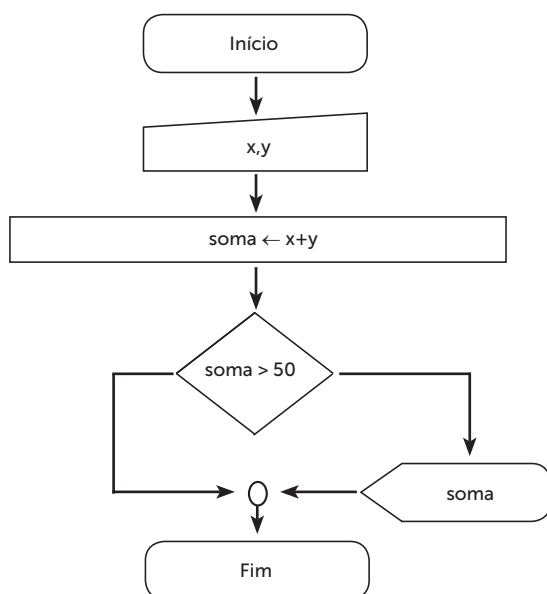
Como evidenciado acima, tenha sempre em mente que é preciso verificar se apenas uma condição será suficiente para atender à necessidade de solução do problema que foi apresentado. Também neste contexto é possível afirmar que o programa, após verificar a condição, se esta for falsa, não retornará valor algum referente ao teste, e sim, simplesmente, dará continuidade no processamento das informações.

Uma vez que este desvio para a realização do teste não resultou em verdadeiro, não há a necessidade de testar os comandos inerentes a essa operação. Contudo, não se esqueça também de que a melhor forma de gravar uma informação é realizando exercícios. Para tal, busque sempre identificar:

1. O cenário e o processo que deverá ser customizado.
2. Qual estrutura de decisão se adequa melhor à situação proposta.
3. Desenvolva a condição de forma a atender às regras de negócio que precisam ser testadas para se realizar a tomada de decisão.

Tendo isso em mente, você já pode iniciar essas práticas. Siga em frente! Vamos revisitar o exemplo da soma entre dois números, agora incrementando uma condição: o programa deve exibir o valor da soma caso esta seja superior a 50.

Primeiramente observe o exemplo de fluxograma que Souza (2013, p. 61) utiliza para explicar essa operação:



O desvio na operação indica, justamente, que após a verificação do teste, se o seu resultado for verdadeiro diante da condição imposta, o programa exibirá o resultado na tela.

E então, caso o resultado seja falso, o procedimento adotado é continuar a execução dos comandos descritos na sequência.



Exemplificando

Observe o exemplo que implementa esta situação com a inserção de uma condição na estrutura “se-então” em pseudocódigo:

algoritmo Exibe Soma

var

x, y, soma: inteiro

inicio

escreval ("Informe o valor de dois números inteiros:")

leia (x, y)

soma \leftarrow x + y

se (soma > 50) **então**

 escreval ("O valor da soma é:", soma)

fimse

finalgoritmo



Faça você mesmo

Agora, tente implementar essa estrutura inserindo a condição simples, utilizando a linguagem de programação C. Isso facilitará a sua compreensão da dinâmica do algoritmo descrito em pseudocódigo e deste já implementado em uma linguagem de programação. Vamos lá!

Observe e teste: seguindo praticamente o mesmo raciocínio utilizado no algoritmo anterior em pseudocódigo, desenvolva um programa que recebe dois números inteiros, os exibe na tela e ainda verifica qual deles é maior, exibindo, por fim, o que for maior entre eles. Veja a seguir uma sugestão:

```
/*Programa que recebe dois valores, verifica qual deles é maior e exibe
na tela*/

#include <stdio.h>

main(){

float x, y, maior;

printf("Informe dois números:");

scanf("%f%f", &x, &y);

maior = x;

if (y > x) {

    maior = y;

}
```

```
printf("Os valores informados foram %f%f", &x, &y);  
  
printf("O maior valor dentre os números digitados é: %f", &maior);  
  
}
```

Vamos entender o programa anteriormente descrito?

Primeiramente, você pode notar a presença de um comentário, pela indicação da frase que vem dentro dos símbolos `/* */`. Toda vez que tiver de inserir um comentário, seja para iniciar o programa, seja para explicar um comando ou outra estrutura, você poderá usar esta referência. Em seguida, há a descrição da biblioteca que deve ser chamada para buscar os comandos de entrada e saída de dados, atribuição, entre outros. Ela vem logo no início com `"#include"`, que justamente inclui neste programa, quando solicitado um comando que esteja nesta biblioteca, as funções que serão executadas.

O exemplo anterior retoma comandos de entrada e saída como `"printf"` e `"scanf"`. Observe ainda que a estrutura condicional simples é utilizada, de acordo com a sintaxe da linguagem de programação C. Por exemplo, a palavra `"se"` é escrita em inglês `"if"` e indica o início dessa estrutura de verificação da condição. Esta, inclusive, será comum a outras plataformas e linguagens de programação.

A condição é inserida entre parênteses `"()"` e, na sequência, vem o comando que deverá ser executado caso a condição seja verdadeira. Se a condição resultar em falso, o programa executará a expressão diretamente descrita após o encerramento do comando `"if"`. Para este algoritmo o que se tem é a exibição dos números digitados e também do maior deles.

Na estrutura condicional simples, como é chamada, é comum encontrar situações em que há necessidade de se implementar uma estrutura condicional composta, agregando à estrutura simples anteriormente exemplificada um outro comando a realizar dentro da mesma etapa, caso a resposta ao teste estabelecido seja falsa. Neste caso, é preciso utilizar o `"se-então-senão"`.



Reflita

- As palavras **se**, **então** e **senão** representam o comando condicional;
- A condição deve ser uma expressão lógica;
- O comando avalia a condição. Se o resultado da expressão for verdadeiro, então será executado o comando 1. Mas, se o resultado for falso, será executado o comando 2 (PIVA JR. et al., 2012, p. 153).

No mesmo raciocínio do exercício anterior, podemos praticar mais um pouco! Vamos agora seguir o exemplo que Evaristo (2001, p. 154) traz para este exercício de desenvolvimento do raciocínio lógico e compreensão da sintaxe em C. O programa consiste em solicitar a inserção de um número decimal qualquer, seja ele uma **dízima periódica** ou não. Observe no exemplo que os dados podem ser determinados pelo usuário. Se pensar em outra aplicação, os dados poderão ser atribuídos também na declaração das variáveis, sendo, assim, um valor constante.

Neste contexto, vamos verificar como fazer o algoritmo que arredonda um número. Siga em frente utilizando o Dev C++, de preferência, como plataforma de desenvolvimento, mas lembre-se, a linguagem utilizada é C.

```
/*Programa para arredondar valores*/

#include <stdio.h>

main()
{
    float num, parteFracionada;
    int Arredondamento;
    printf("Digite um número:");
    scanf("%f", &num);
    Arredondamento = num;
    parteFracionada = num - Arredondamento;
    if (parteFracionada >= 0.5)
```

```

        Arredondamento = Arredondamento +1;

        printf("O valor digitado %f quando arredondado obtém-se %d", num,
        Arredondamento);

    }

```

Esta operação é considerada de fácil compreensão e manipulação. Por esse motivo, além do VisuAlg, é importante que você refaça todos os exercícios sugeridos também no Dev C++, a fim de desenvolver maior autonomia, seja quanto ao uso e conhecimento da linguagem, seja quanto à construção do raciocínio lógico que advém das práticas.

Agora vamos praticar mais um pouco, com o desenvolvimento de um algoritmo que verifica um dado textual, ou seja, um tipo de dado caractere ou char. Suponha que você precise desenvolver um algoritmo que solicite ao usuário a inserção do nome, da idade e do gênero.

Esse algoritmo deverá verificar se o conteúdo informado pelo usuário referente à variável gênero é feminino ou masculino, e caso seja diferente de uma destas opções, o programa exibe uma mensagem ao usuário para que escolha apenas uma das opções apresentadas. Vamos lá!

```

#include <stdio.h>

#include <stdlib.h>

main()
{
    char gênero, nome;
    int  idade;
    system("cls");
        gênero = ' ';

    printf ("\nInforme o seu nome: \n");
    printf ("\nInforme a sua idade: \n");
    printf ("\nDigite F para o gênero feminino ou M para masculino:\n");
    scanf("%c%d%c", &nome, &idade, &gênero);
    if (gênero != 'F' && gênero != 'M')

```

Este comando limpa a tela de exibição dos dados que serão imediatamente apresentados.

```

{
    printf("Dado inválido.");
}

printf("\n\n");
system("pause");
return (0);
}

```

Este comando faz com que a tela de exibição dos resultados (DOS) permaneça aberta logo após o processamento das operações. Não feche automaticamente.

Muito bem, até aqui foram apresentados vários algoritmos que utilizam-se da estrutura de decisão condicional simples "se-então". Além de conhecer e poder aplicar em VisuAlg, você também pode empregar esta prática diretamente no ambiente de desenvolvimento Dev C++, conforme recomendado para os exercícios anteriores.

A partir de agora, recomenda-se também que você investigue um pouco mais sobre outros recursos computacionais que permitem o desenvolvimento de algoritmos. Um dos exemplos que passaremos a indicar é o Scilab. Abaixo, algumas recomendações de leituras para a sua familiarização com o ambiente.



Pesquise mais

Acesse os *links* que apresentam uma linguagem de programação também bastante utilizada para o ensino e desenvolvimento de algoritmos: Scilab.

- Apostila de Scilab da Universidade Federal do Rio Grande do Norte. Acaba de ser disponibilizada para consultas em: <<http://www.dca.ufrn.br/~estefane/academica/progsci.pdf>>. Acesso em: 19 maio 2015.
- Apostila de Scilab disponibilizada pela Universidade Estadual de Campinas: <http://www.ime.unicamp.br/~encpos/VIII_EnCPos/Apostila_Scilab.pdf>. Acesso em: 19 maio 2015.
- Vídeo: <<https://www.youtube.com/watch?v=Tdx3xErGAkw>>. Acesso em: 19 maio 2015.



Vocabulário

1. **Dízima periódica:** atribui-se esse nome ao número decimal ou fracionário que contém uma repetição periódica e infinita de algarismos.
2. **Semântica:** ação que resulta da execução do programa, porém, que apresenta um erro lógico. Diferentemente da sintaxe, que se não estiver correta o compilador identificará, um erro semântico, e por isso lógico, não é indicado na definição do comando por si, e sim no resultado da ação proveniente desse comando.
3. **Compilador:** interpretador da linguagem de programação para a linguagem de máquina. Com isso é possível realizar o processamento das operações realizadas. É composto por processador de texto para a codificação; um depurador que executará comando a comando (e este ainda facilita a busca por erros e os indica quando houver) um manual ou help para possíveis apresentações sintáticas e semânticas; e um linker, que faz com que um programa utilize recursos ou ainda outros programas (EVARISTO, 2001).

Sem medo de errar

Considere a solicitação de verificação de *login* que foi realizada para a Think Now inserir no protótipo do aplicativo, sendo que esta deverá solicitar ao usuário a indicação do tipo de usuário: se é comerciante ou se é cliente, uma única vez ao realizar o cadastro. Observe a seguir uma proposta para este algoritmo:

```
algoritmo "Teste cadastro usuário"

// Função : algoritmo verifica se o usuário pretende logar com uma rede social
//em seguida, verifica se o usuário é comerciante

// Autor :

// Data : 25/05/2015

// Seção de Declarações

var

nome,email: caracter

logarRedeSocial, comerciante: inteiro
```

```
inicio
// Seção de Comandos
escreval ("**Cadastro Usuário Gastronomia e Hotéis do Litoral Sul**")
escreval ("Informe o seu nome:")
leia(nome)
escreval("Logar com uma rede social?")
escreval("Responda 1 para Sim ou 2 para Nao.")
leia(logarRedeSocial)
se (logarRedeSocial=1) entao
    escreval("Aguarde enquanto o aplicativo configura o seu cadastro!")
senao
    escreval("Informe o seu email!")
    leia(email)
fimse
escreval("Você é comerciante?")
    escreval("Responda 1 para Sim ou 2 para Nao.")
    leia(comerciante)
    se (comerciante=1)entao
        escreval("Prezado Parceiro de Negócios, finalize o seu cadastro!")
    fimse
escreval("Cadastro realizado com sucesso!")
fimalgoritmo
```

Avançando na prática

Pratique mais!

Instrução

Desafiamos você a praticar o que aprendeu transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois compare-as com as de seus colegas.

Estrutura condicional simples

| | |
|--|--|
| 1. Competência de fundamentos de área | Conhecer os princípios e conceitos que envolvem o aprendizado em construção de algoritmos e programação e a sua importância para o universo do desenvolvimento de sistemas. |
| 2. Objetivos de aprendizagem | <ul style="list-style-type: none"> • Conhecer os conceitos e aplicações das estruturas de decisão "SE". • Conhecer como fazer uma estrutura condicional simples. • Conhecer e aplicar as estruturas condicionais compostas. • Conhecer e aplicar estruturas condicionais encadeadas. |
| 3. Conteúdos relacionados | <ul style="list-style-type: none"> • Instruções primitivas: entrada de dados, atribuição e saída. • Estrutura condicional simples. |
| 4. Descrição da SP | Para treinar os seus conhecimentos, propomos que você faça um algoritmo para resolver uma equação de segundo grau. |
| 5. Resolução da SP | <pre> /*Algoritmo Cálculo de raízes da equação de 2º grau. var A, B, C: real //indica a declaração das variáveis e o seu tipo de dado específico delta, x1, x2: real //indica a declaração das variáveis e o seu tipo de dado específico INICIO //indica o início do algoritmo Leia(A,B,C) //significa que o programa precisa ler estas variáveis, ou seja, verificar os seus valores delta ← B*B - 4.*A*C //expressão linear que indica o cálculo do "delta" componente da fórmula Se (delta >= 0) entao // estrutura de seleção que atribui uma condição para a realização das operações x1 ← (-B + sqrt(delta))/(2.*A) //expressão linear com o uso da função sqrt que calcula a raiz quadrada para a raiz 1 x2 ← (-B - sqrt(delta))/(2.*A) //expressão linear com o uso da função sqrt que calcula a raiz quadrada para a raiz 2 Escreva('As raízes reais são:', X1, X2) //comando que exhibe o resultado do cálculo das raízes. senao Escreva('Não existem raízes reais') //comando que será executado caso a condição predeterminada não seja atendida FIM //indica o encerramento das execuções do programa </pre> |



Lembre-se

Bibliotecas do C:

stdio.h: é responsável pelos comandos de entrada e saída de dados.

math.h: é responsável pelo acesso aos comandos para a declaração de funções matemáticas.

Stdlib.h: é responsável por converter os números, alocar ou realocar memória, entre outras funções.



Faça você mesmo

Implemente o algoritmo anterior usando a linguagem C. Siga o exemplo:

```
#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <math.h>

int main(void){

    // declaracao de variáveis

    float A, B, C, delta, x1,x2;

    printf("Digite com os coeficientes da equação\n"); // leitura dos dados

    scanf("%f %f %f",&A,&B,&C); // %d representa o tipo de dado inteiro; %f
    representa o tipo de dado real; %c representa o tipo de dado char; %s
    representa o tipo de dado string

    delta = B*B - 4.*A*C; // cálculo do delta

    // Teste do discriminante

    if (delta >= 0) {

        // calculo das raizes

        x1 = (-B + sqrt(delta))/(2.*A);

        x2 = (-B - sqrt(delta))/(2.*A);
```

```
printf("A equacao dada e: %f X^2 + %f X + %f\n", A,B,C);

printf("As raizes reais sao: raiz 1= %f 2a. raiz2= %f\n", x1, x2); }

else { printf("Nao existem raizes reais"); } getch(); } // fim do programa
```

Faça valer a pena!

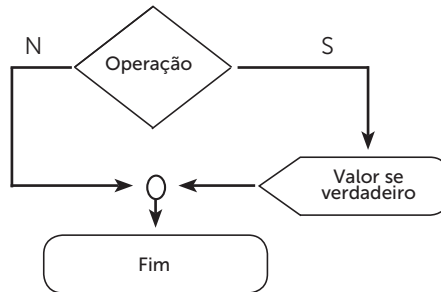
1. Dada a estrutura de seleção abaixo, assinale a alternativa que melhor descreve o processo:

```
se nivel = 1 entao
    escreval("O professor ganha"horas*12)
senao
    se nivel = 2 entao
        escreval("O professor ganha"horas*17)
    senao
        escreva("O professor ganha"horas*25)
fimse
fimse
fimalgoritmo
```

(Disponível em: <http://partilho.com.br/visualg/exercicios-visualg/visualg-lista-de-exercicios/#Exercicio_6>. Acesso em: 27 maio 2015).

- a. () Os comandos condicionais acima representam uma sucessão de testes com o uso de estruturas de seleção simples.
- b. () Os comandos indicam que há primeiro uma estrutura de seleção que verifica se o nível indicado é o 1, em seguida, através de uma composição com o senão, há o teste em uma condição encadeada que verifica se o dado inserido equivale ao nível 2 e, caso não seja, implica em aplicar uma terceira regra, finalizando o processo.
- c. () Os comandos de repetição acima representam uma sucessão de testes com o uso de estruturas de seleção simples.
- d. () Os comandos de decisão e repetição inseridos verificam e atribuem ao salário a somatória dos indicadores apresentados em cada uma das condições.
- e. () Os comandos se, então e senão, neste caso, representam uma agregação que poderá ser representada em fluxograma, e não na implementação.

2. O diagrama de blocos a seguir indica, respectivamente:



- a. () condição, resultado e encerramento do processo.
- b. () repetição, condição e operação.
- c. () condição, valor se verdadeiro e valor se falso.
- d. () repetição, valor se verdadeiro e encerramento.
- e. () repetição, valor se verdadeiro e operação.

3. Dada a sintaxe abaixo, é possível afirmar o que está descrito na alternativa:

```

if (sexo != 'F' && sexo != 'M')
{
    printf ("Dado inválido.");
}
  
```

- a. () o uso das chaves é de extrema importância nesse caso, pois indica que o comando é referente à condição imposta logo acima pelo comando condicional e não pode ser suprimida.
- b. () as chaves podem ser suprimidas independentemente da quantidade de argumentos que serão executados na estrutura de decisão.
- c. () os operadores relacionais e lógicos são aplicados de forma errônea e interferem no resultado da operação.
- d. () o uso das chaves pode ser suprimido apenas quando há um argumento que deverá ser executado após a verificação do comando.
- e. () a sintaxe do comando descrito acima está errada.

4. Para o comando abaixo, descrito em linguagem C, a palavra que completa o raciocínio lógico aplicado com o comando condicional é: Assinale a alternativa equivalente: `if (delta >= 0)`.

- a. `()` `senão`.
- b. `()` `faça`.
- c. `()` `else`.
- d. `()` `then`.
- e. `()` `if`.

5. O que representam as palavras `se`, `então` e `senão` para as estruturas de decisão ou seleção?

6. Explique as principais funções das bibliotecas:

- a. `stdio.h`:
- b. `math.h`:
- c. `stdlib.h`:

7. Analise a condição abaixo quanto à sintaxe e assinale a alternativa correspondente:

```
if (y > x) {  
    maior = y;  
}
```

- a. `()` a condição apresentada verifica, através do uso do operador relacional maior `>`, se o valor de `"y"` é maior que o valor da variável `"x"`.
- b. `()` o maior valor é armazenado na própria variável `"y"`.
- c. `()` a condição está semanticamente incorreta.
- d. `()` a sintaxe está incorreta.
- e. `()` o valor de `x` será sempre o maior valor de acordo com a condição inserida.

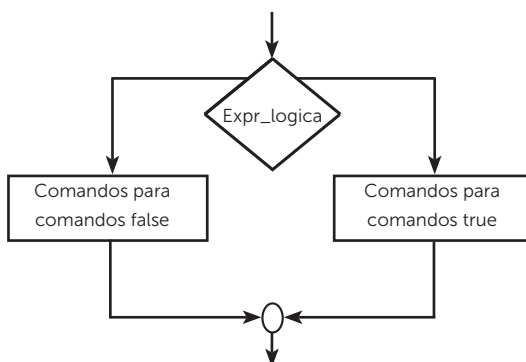
Seção 2.3

Estrutura condicional composta

Diálogo aberto

Olá, aluno. Primeiramente vamos recordar a estrutura de decisão composta: se-então-senão. Observe a estrutura que estudaremos: será com esse modelo lógico que os exercícios e soluções serão desenvolvidos.

Figura 5 | Estrutura de decisão composta



Fonte: Adaptado de Souza (2013, p. 128).

Vamos recapitular as entregas realizadas até o momento para o protótipo do aplicativo de divulgação dos locais de gastronomia e hotelaria do Litoral Sul: primeiro entregamos identificação das ações do aplicativo; depois, um algoritmo para contabilizar os acessos; em seguida, a entrega foi de um fluxograma das ações do sistema, integrando o procedimento de *login* ao processo de contabilização de acessos; identificamos outros processos que poderão ser acoplados ao protótipo e à sua respectiva necessidade de declaração de variáveis. Estamos agora na segunda etapa e as entregas realizadas foram: um algoritmo de medição de índice de satisfação do cliente e um algoritmo que verifica no momento do cadastro se o usuário é um comerciante ou não, para que ele seja direcionado à sua respectiva atividade no sistema. Agora, você precisará implementar a seguinte verificação quando o usuário for identificado como comerciante:

a. O usuário deverá informar alguns dados de cadastro para se manter como cooperado do sistema. Então ele deverá informar, além dos dados de cadastro advindos da rede social, caso tenha optado por esta forma de *login*, o seu endereço comercial: Rua, número, bairro, cidade, estado e CEP, para envio de boleto de contribuição mensal para o desenvolvimento e manutenção do *site* e aplicativo.

b. Se o usuário escolher a opção gastronomia, ele pagará o valor de R\$ 30,00 mensais para manter o vínculo com o *site*. Senão, ele será do setor de hotelaria e deverá pagar o valor de R\$ 20,00 para manter o vínculo com o *site*. Bons estudos!

Não pode faltar

As estruturas de decisão compostas, como o próprio nome diz, propõem as verificações de condições e a execução dos comandos pertinentes a elas. Isso significa que para uma verificação de condição que resulte em verdadeira, serão executados comandos pertinentes a essa resposta, senão, os comandos que serão executados pertencem ao processo referente ao resultado que seja negativo ou falso, do ponto de vista do teste realizado.



Refleta

*[...] representa alguma expressão lógica, que se resultar **true**, vai permitir a execução de um conjunto de um ou mais comandos quaisquer existentes no caminho true, os quais podem ser sequenciais, de decisão ou de repetição. [...] se o resultado for **false**, será executado um conjunto contendo um ou mais comandos quaisquer, existentes no caminho false, podendo, novamente, ter estruturas sequenciais, de decisão ou de repetição. Ambos os fluxos convergem para o final da estrutura (SOUZA, 2013, p. 128).*

É sabido, portanto, que em uma estrutura de decisão composta o sistema deverá apresentar uma resposta ou outra apenas. Recorda-se do exemplo de algoritmo para cálculo da média de alunos? A condição imediatamente expressa no comando "se" ou "if". Vejamos, por exemplo, uma verificação da nota. Se esta for maior que 7 (sete), o aluno deverá visualizar a mensagem de que está "Aprovado" na disciplina, senão, caso sua nota seja inferior ao valor estabelecido na condição, a mensagem exibida deverá ser "Reprovado". Então, seguindo este raciocínio de apresentação de uma resposta caso a condição seja verdadeira ou falsa, é possível aplicar tal estrutura para diversas situações em que há a necessidade de verificação de uma regra. Observe no exemplo como resolver essa simples verificação:



Exemplificando

Algoritmo "Cálculo da média"

// Função: Algoritmo calcula a média ponderada a partir da apresentação das notas e exibe mensagem de aprovação ou reprovação"

// média (ponderada), informar o nome e sua menção aprovado (media ≥ 7), senão, aluno reprovado.

// Autor : JJJ

// Data : 28/05/2015

// Seção de Declarações

var

nomeAluno: caractere

n1, n2, n3, media: real

inicio

 // Seção de Comandos

 escreval("***Cálculo médias***")

 escreva("Digite a primeira nota: ")

 leia(n1)

 escreva("Digite a segunda nota: ")

 leia(n2)

 escreva("Digite a terceira nota: ")

 leia(n3)

 media $\leftarrow (n1*4 + n2*2 + n3*4) / 10$

 se media ≥ 7 entao

 escreval("Aluno aprovado!")

 senao

 escreval("Aluno reprovado!")

```
fimse
finalgoritmo
```



Faça você mesmo

Agora você mesmo já pode implementar esse exemplo de cálculo da média no Dev C++. Tente com a média ponderada e também a aritmética. Bom trabalho!

```
/*Programa para verificar a aprovação de um aluno*/

#include <stdio.h>

main() {

float Bim1, Bim2, Bim3, Bim4, MedBim, PrFinal, MedFinal;

printf("Digite as quatro notas bimestrais:");

scanf("%f%f%f%f", &bim1, &Bim2, &Bim3, &Bim4);

MedBim= (Bim1+Bim2+Bim3+Bim4)/4;

MedFinal=MedBim;

if((MedBim<7) && (MedBim>=5))

{

    printf("Digite a nota da prova final");

    scanf("%f", &PrFinal);

    MedFinal=(MedBim*6 + PrFinal*4)/10;

}

if (MedFinal>5.5)

    printf("Aluno aprovado com media final %.2f \n", MedFinal);

else

    printf("Aluno reprovado com media final %.2f \n", MedFinal);

}
```

Fonte: Evaristo (2001, p. 48)



Vocabulário

True: verdadeiro

False: falso

Vamos agora falar de algumas particularidades dos algoritmos anteriormente apresentados. Primeiramente no pseudocódigo, quando o autor especificou que o aluno estará aprovado apenas se obtiver nota " \geq " (maior ou igual) a 7 (sete), limitou e definiu através de operadores relacionais que todas as notas inferiores a sete fariam com que o resultado fosse negativo para a condição imposta. Lembre-se quando você estudou limites, funções e inequações? Pois bem, estas podem fazer cada vez mais parte de sua rotina de desenvolvimento de soluções, então, a sugestão é que você saiba cada vez mais aplicar os conteúdos matemáticos às regras de negócios existentes nos mais variados segmentos de mercado.

Outro fator interessante de observar está no comando para calcular a média:

media $\leftarrow (n1*4 + n2*2 + n3*4) / 10$.

Observe que o cálculo obedecerá à representação da expressão e efetuará a operação de multiplicação, adição e divisão, respectivamente, em função da precedência matemática e prioridade de execução. Se a expressão não tivesse os parênteses, o compilador entenderia que apenas " $n3*4$ " seria dividido por 10, por exemplo. Então, não se esqueça de alguns fatores simples, porém, elementares no ato de desenvolvimento de *softwares*.

Agora, já no algoritmo em C, o que vamos destacar são os seguintes comandos:

- **scanf("%f%f%f%f", &Bim1, &Bim2, &Bim3, &Bim4);**

a. neste comando, os sinais de percentual (%) seguidos da letra f indicam ao compilador que este deverá atribuir valores do tipo de dado "float", ou seja, que permitem valores decimais, às variáveis em destaque "&Bim". O símbolo "&" indica a variável em que será alocado o valor informado.

- `if((MedBim<7) && (MedBim>=5))`

b. Já o destaque deste comando condicional está para o uso do operador lógico “e”, ou como na sintaxe acima, “&&”, que indica que as duas condições precisam ser atendidas para que a ação de aprovar o aluno seja realizada, no caso, o valor verdadeiro.

- `printf("Aluno reprovado com media final %0.2f \n", MedFinal);`

c. A expressão ‘.’ por ‘.’ indica que os valores decimais serão apresentados com até no máximo duas casas após a vírgula.

Muito bem, agora que já compreendeu outros fatores que podem influenciar no processamento e resultado final do seu algoritmo, conheça o operador ternário. A lógica proposta por este operador ternário poderá ser utilizada não apenas em aplicações desenvolvidas em C.

Costuma-se utilizá-lo quando a verificação tiver que retornar apenas o valor se falso ou o valor se verdadeiro. Uma das vantagens é que precisa de apenas uma única linha de comando, ao contrário de ter de usar a estrutura de decisão *if/else* (*se-então-senão*), o que pode otimizar o processamento dessa operação.

Observe como a sintaxe do operador ternário pode auxiliar no caso de condições compostas como *se-então-senão*. Então temos:

NomeDaVariável = (a expressão lógica que deverá ser testada) ? O comando para valores verdadeiros : O comando para valores se a condição for falsa.

Veja o exemplo:

`Maximo = (a >= b) ? a : b;`

Compreenda que a variável de nome “Máximo” receberá o valor de a ou b após a realização do teste, logo, é possível inferir que a operação representa a seguinte ação:

- **Se** o valor da variável a for maior ou igual que o valor contido na variável b, então, o resultado será apresentar o valor de a, *senão*, o resultado será apresentar o valor de b.

Outro exemplo de uso do operador ternário pode ser aplicado para o cálculo da média, por exemplo:

`Media = (n1 >= 7) ? "aprovado" : "reprovado";`

Todas as situações que envolvem regras de negócios, ou ainda, várias verificações antes de efetuar a execução dos comandos subsequentes, podem

ser mais ou menos complexas. Portanto, suponha que você tenha de executar o cálculo da média, oferecer opções para escolha e ainda apresentar outro resultado para esta operação, por exemplo, se o objetivo do programa fosse a criação de vários cenários dentro da mesma situação-problema.



Assimile

O comando condicional pode variar no seu formato, pode ser composto com outros comandos e pode conter um comando condicional dentro de outro comando condicional (PIVA JR. et al., 2013, p. 154).

Você pode, com as estruturas de decisão compostas, implementar tais verificações ou opções.

Sem medo de errar

algoritmo "Verifica dados cadastro"

// Função : permite a inserção de dados cadastrais e verifica a existência de

//campos vazios.

// Autor: JJJ

// Data : 29/05/2015

// Seção de Declarações

var

nome, end:caractere

op: inteiro

inicio

// Seção de Comandos

escreval("Prezado Cooperado, informe os seus dados:")

escreval("Nome:")

leia(nome)

```
escreval("Endereço:")  
  
leia(end)  
  
escreval("Informe 1 para Gastronomia e 2 para Hotelaria:")  
  
leia (op)  
  
se (op = 1) entao  
    escreval("Imprimir o boleto no valor de R$30,00")  
  
senao  
    escreval("Imprimir o boleto no valor de R$20,00")  
  
fimse  
  
finalgoritmo
```



Atenção!

Você ainda poderá implementar uma verificação de erros para essa rotina! Insira outros testes, ou seja, outras estruturas de decisão.



Lembre-se

Outras funções podem ser usadas no VisuAlg:

Abs(expressão) - Retorna o valor absoluto de uma expressão do tipo inteiro ou real.

ArcCos(expressão) - Retorna o ângulo (em radianos).

ArcSen(expressão) - Retorna o ângulo (em radianos).

ArcTan(expressão) - Retorna o ângulo (em radianos).

Cos(expressão) - Retorna o cosseno do ângulo (em radianos).

CoTan(expressão) - Retorna a cotangente do ângulo (em radianos).

Exp(base, expoente) - Retorna o valor de base elevado a expoente, sendo

ambos expressões do tipo real.

GraupRad(expressão) - Retorna o valor em radianos correspondente ao valor em graus representado por expressão.

Int(expressão) - Retorna a parte inteira do valor representado por expressão.

Log(expressão) - Retorna o logaritmo na base 10 do valor representado por expressão.

LogN(expressão) - Retorna o logaritmo neperiano (base e) do valor representado por expressão.

Pi - Retorna o valor 3.141592.

Quad(expressão) - Retorna o quadrado do valor representado por expressão.

RadpGrau(expressão) - Retorna o valor em graus correspondente ao valor em radianos representado por expressão.

RaizQ(expressão) - Retorna a raiz quadrada do valor representado por expressão.

Rand - Retorna um número real gerado aleatoriamente, maior ou igual a zero e menor que um.

RandI(limite) - Retorna um número inteiro gerado aleatoriamente, maior ou igual a zero e menor que limite.

Sen(expressão) - Retorna o seno do ângulo (em radianos) representado por expressão.

Tan(expressão) - Retorna a tangente do ângulo (em radianos) representado por expressão.

Disponível em: <<http://www.apoioinformatica.inf.br/produtos/visualg/linguagem/item/30-as-funcoes-do-visualg-versao-2-0>>. Acesso em: 29 maio 2015.

Avançando na prática

| Pratique mais! | |
|--|--|
| Instrução Desafiemos você a praticar o que aprendeu transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois compare-as com as de seus colegas. | |
| Estrutura condicional composta | |
| 1. Competência de fundamentos de área | Conhecer os princípios e conceitos que envolvem o aprendizado em construção de algoritmos e programação e a sua importância para o universo do desenvolvimento de sistemas. |
| 2. Objetivos de aprendizagem | <ul style="list-style-type: none"> • Conhecer os conceitos e aplicações das estruturas de decisão "SE". • Conhecer como fazer uma estrutura condicional simples. • Conhecer e aplicar as estruturas condicionais compostas. • Conhecer e aplicar estruturas condicionais encadeadas. |
| 3. Conteúdos relacionados | <ul style="list-style-type: none"> • Instruções primitivas: entrada de dados, atribuição e saída • Estrutura condicional simples |
| 4. Descrição da SP | Desenvolva um programa em linguagem de programação C que identifique se estamos em um ano bissexto. |
| 5. Resolução da SP | <pre>/*Programa que verifica se um dado ano é bissexto */ #include <stdio.h> main() { int Ano; printf("Digite o ano"); scanf("%d", &Ano); if (Ano % 4 == 0) printf("%d e' bissexto %d \n", Ano); else printf("%d nao e' bissexto %d \n", Ano); }</pre> <p>Fonte: Adaptado de Evaristo (2001, p. 41).</p> |



Lembre-se

Operações mais complexas poderão ser inseridas com o uso da estrutura de decisão encadeada. Continue os estudos e pratique!

Faça valer a pena!

1. Desenvolva um algoritmo na plataforma Dev C++ que ordene três números inseridos pelo usuário.
2. Assinale a alternativa em que há explicação correta para o comando `"scanf("%f%f%f%f", &Bim1, &Bim2, &Bim3, &Bim4);"`:
 - a. () Os sinais de percentual (%), seguidos da letra f, indicam ao compilador que este deverá atribuir valores do tipo de dado float. O símbolo "&" indica a variável em que será alocado o valor informado.
 - b. () Os sinais de percentual (%) seguidos da letra f indicam a variável em que será alocado o valor informado.
 - c. () O símbolo "&" indica ao compilador que este deverá atribuir valores do tipo de dado float.
 - d. () Indica que se o valor da variável Bim1 for maior ou igual ao valor contido na variável Bim2, então o resultado será apresentar o valor de Bim1, senão, o resultado será apresentar o valor de Bim2.
 - e. () Todas as situações que envolvem várias verificações antes de efetuar a execução dos comandos subsequentes utilizam o "%f" e o "&".
3. Leia as afirmações e assinale a alternativa que apresenta a sequência correta (Verdadeiro ou Falso):
 - I. O comando condicional pode variar no seu formato, pode ser composto com outros comandos e pode conter um comando condicional dentro de outro comando condicional.
 - II. A expressão `%0,2f` indica que os valores decimais serão apresentados em seu formato sem vírgulas.
 - III. O comando condicional não pode conter mais de um comando para execução.
 - a. () V, V, V.
 - b. () F, F, F.
 - c. () F, V, F.
 - d. () V, F, F.
 - e. () V, V, F.

4. Analise o comando "Maximo = (a >= b) ? a : b;" e assinale a alternativa que representa a sua estrutura:

- a. ☐ se-então.
- b. ☐ operador ternário.
- c. ☐ se-então-senão.
- d. ☐ se-senão.
- e. ☐ se-então-se.

5. Descreva a lógica envolvida na estrutura do operador ternário.

6. O que representa "if ((Mes == 4) || (Mes == 6) || (Mes == 9) || (Mes == 11))"?

- a. ☐ a expressão usa o operador lógico "e".
- b. ☐ a expressão usa o operador lógico "ou".
- c. ☐ a expressão usa o operador lógico "nao".
- d. ☐ a expressão usa o operador lógico "xou".
- e. ☐ a estrutura é de decisão simples.

7. O comando printf("O mes %d tem %d dias", Mes, NumDias); está:

- a. ☐ correto.
- b. ☐ incorreto.
- c. ☐ parcialmente correto.
- d. ☐ incompleto.
- e. ☐ semanticamente incorreto.

Seção 2.4

Estrutura condicional sequencial e encadeada

Diálogo aberto

Olá, aluno!

Vamos avançar um pouco mais e conhecer as estruturas condicionais compostas sequenciais e encadeadas?

A princípio, você identificará as suas formas de aplicação e em que situações estas são mais apropriadas. Então, quando se pensa em um aplicativo ou em um *software* que precisará verificar algumas condições, essas opções são bastante comuns.

No entanto, você estudará também outras estruturas que oferecem maneiras de otimizar tanto a codificação quanto o processamento.

Mas antes de conhecer tais estruturas, é importante saber diferenciar e aplicar as estruturas de decisão em suas diversas formas. Nesta seção, vamos conceituá-las e aplicá-las sequencialmente e encadeadas.

Neste ponto do desenvolvimento você incrementará o cadastro de usuário, porém, considerando um processo na visão do administrador desse aplicativo. Você deverá implementar uma opção que permita incluir, alterar e excluir informações de cadastro. Para tal, você ainda pode utilizar o VisuAlg.

Além de outras ferramentas que podem ser utilizadas, no entanto, se quiser transcrever em alguma linguagem computacional, a preferência para este momento de aprendizagem ainda repousa sobre a linguagem C, no Dev C++.

Por esse motivo, as estruturas condicionais ou de decisão, sequenciais e encadeadas, serão importantes para que você consiga organizar tais informações e implementar as opções.

Nesse sentido, você precisa pensar na aplicação de tais estruturas e em como podem auxiliar nesse processo.

Refleta sobre a situação proposta, qual será a melhor maneira de estruturar esse

algoritmo de forma que o seu custo, ou seja, o tempo de processamento e a necessidade de consumo de recursos de máquina não sejam muito elevados.

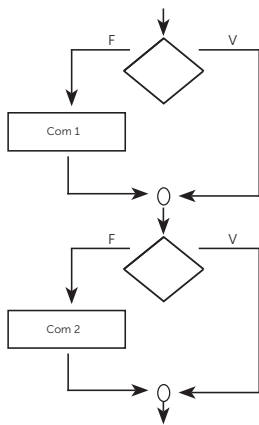
De acordo com essas novas implementações é que o seu estudo será orientado!

Desde já, bons estudos e práticas a você!

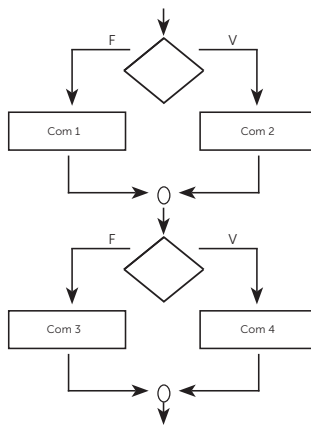
Não pode faltar

Você já aprendeu e aplicou o conceito da estrutura condicional simples e composta, porém, terá maior contato com a estrutura sequencialmente aplicada a partir deste estudo. O mesmo se aplica à estrutura encadeada. Então, quando ocorre o desvio sequencial ou encadeado em um algoritmo? É correto dizer que a estrutura sequencialmente aplicada é utilizada quando há uma estrutura condicional simples ou composta de forma que, após uma verificação, ou seja, uma estrutura condicional aplicada, vem logo em seguida outra aplicação de estrutura condicional, sendo, por essa razão, sequencial. Observe o exemplo no diagrama de bloco abaixo:

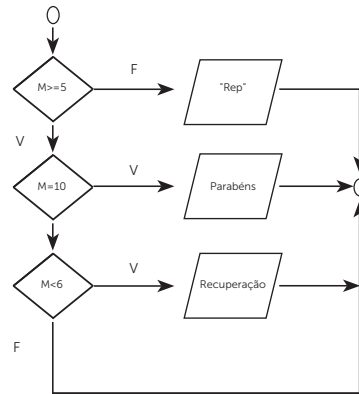
Figura 6 | Diagrama de bloco (fluxograma) estruturas condicionais simples e compostas sequenciais



Fonte: Adaptado de Souza (2013, p. 64).



Fonte: Adaptado de Souza (2013, p. 64).



Fonte: Adaptado de Piva Junior et al. (2012, p. 175).

A Figura 6 mostra três tipos de operações que podem ser realizadas com as estruturas de decisão sequencial simples e compostas. A primeira imagem representa a verificação da condição e a resposta caso esta seja verdadeira (Com 1); se for falsa, na sequência, pode ser inserida outra verificação que indicará outros comandos, do grupo de comandos 2 (Com 2) a executar caso a condição seja verdadeira, e assim sucessivamente.

A segunda imagem traz também a estrutura de decisão sequencial composta. Observe que a resposta para a verificação, tanto se verdadeira quanto se falsa, pertencem à mesma condição de verificação na estrutura. Em seguida, há outra verificação sendo executada e as respectivas respostas à condição, tanto se verdadeira quanto se falsa.

Na terceira imagem, a estrutura condicional composta apresenta a sequência de execução de atividades exibindo a resposta caso a condição verificada seja falsa, e então, se a condição for verdadeira, dá-se sequência às demais verificações de condições impostas nas estruturas de decisão. Também nesse caso, todas convergem para o encerramento da execução conforme a indicação.



Assimile

- Estrutura de decisão sequencial simples:

se (condição 1) **entao**

comandos 1 para condição verdadeira

fimse

se (condição 2) **entao**

comandos 2 para condição verdadeira

fimse

- Estrutura de decisão sequencial composta:

se (cond1) **entao**

se (cond2) **entao**

Conj, Comandos1

senao

Conj. Comandos2

fimse

senao

se (cond3) entao

Conj. Comandos3

senao

Conj. Comandos 4

fimse

fimse

Anteriormente foram apresentadas as estruturas de decisão sequencial simples e composta em pseudocódigo, evidenciando a forma como estas devem ser tratadas na codificação.



Reflita

As formas [...] apresentadas podem ser combinadas entre si, gerando outras possibilidades. Assim sendo, podem existir tomadas de decisão sequenciais com tomadas de decisão simples em conjunto com tomadas de decisão compostas. A codificação dessas estruturas segue as formas dos respectivos diagramas de blocos (MANZANO; OLIVEIRA, 2010, p. 66).

A fim de compreender melhor tais estruturas, veja no exemplo a seguir, mencionado pelos autores, a sua representação tanto em diagrama de blocos quanto em pseudocódigo.



Exemplificando

Suponha que, em uma cidade, a empresa que faz o controle e tratamento de água e esgoto precisa controlar o nível de água das caixas d'água dos bairros, para que os controladores possam identificar se a caixa d'água está cheia, vazia, transbordando ou sem vazão. Para tal, foram propostas as seguintes regras para verificação:

1. Verificar se o valor exibido na tela é igual a 1. Se sim, exibe a mensagem "Caixa d'água cheia." Senão, executar a condição que vem

expressamente na sequência.

2. Verificar se o valor informado é igual a 2. Se sim, exibe a mensagem "Caixa d'água vazia." Senão, executar a condição que vem expressamente na sequência.

3. Verificar se o valor informado é igual a 3. Se sim, exibe a mensagem "Caixa d'água transbordando." Senão, executar a condição que vem expressamente na sequência.

4. Verificar se o valor informado é menor ou igual a 0. Se sim, exibe a mensagem "Caixa d'água sem vazão." Senão, executar a condição que vem expressamente na sequência.

Agora, no VisuAlg, faça o seguinte pseudocódigo, que auxilia na resolução do problema proposto no exemplo:



Faça você mesmo

```
//algoritmo decisão_sequencial
```

```
var
```

```
N: inteiro
```

```
inicio
```

```
    leia (N)
```

```
    se (N =1) entao
```

```
        escreval ("Caixa d'agua cheia.")
```

```
    fimse
```

```
    se (N =2) entao
```

```
        escreval ("Caixa d'agua vazia.")
```

```
    fimse
```

```
    se (N = 3) entao
```

```
        escreval ("Caixa d'agua transbordando.")
```

```
    fimse
```

```
    se (N <= 0) entao
```

escreval ("Caixa d'água sem vazão.")

fimse

fimalgoritmo

Fonte: Adaptado de Manzano e Oliveira (2010, p. 67).



Pesquise mais

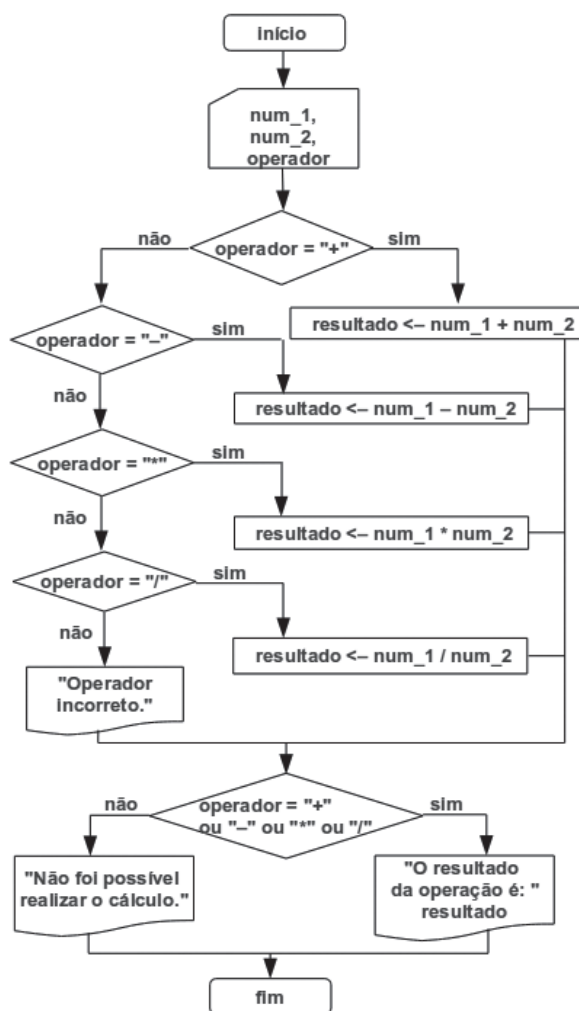
Material didático da Universidade Estadual de Maringá.

Disponível em: <<http://www.din.uem.br/~teclopes/FCaula5.pdf>>.

Acesso em: 02 jun. 2015.

A partir de agora, o foco está nas estruturas de decisão encadeadas. Estas são bastante utilizadas quando se deseja inserir, dentre várias opções, respostas para cada uma das verificações, tanto para o caso de esta condição ser verdadeira quanto para o caso de a condição resultar em falsa. Neste caso, a resposta para a condição quando esta é falsa é a inserção de um outro teste através da estrutura de decisão encadeada, sem a necessidade de finalizar a estrutura com "fimse" todas as vezes que se desejar inserir outra verificação. Observe a estrutura de decisão encadeada apresentada na Figura 7:

Figura 7 | Estrutura de decisão encadeada



Fonte: <http://www.activeinfo.com.br/curso_programacao/estruturas_de_decisao.html>. Acesso em: 03 jun. 2015.

Veja que a sequência de execução das instruções tem como consequência a outra condição ser verificada como resposta da estrutura anterior, tanto para o caso desta ser verdadeira quanto para o caso desta ser falsa.



Pesquise mais

Confira mais dicas de programação no *link*:

<<http://www.dicasdeprogramacao.com.br/estrutura-de-decisao-senao-senao/>>. Acesso em: 03 jun. 2015.

Os comandos condicionais encadeados podem ser substituídos em alguns casos por estruturas de seleção (CASO) que oferecem uma lista de opções ao usuário.

Sem medo de errar

```

algoritmo "Cadastro"
// Função :
// Autor : JJJ
// Data :
// Seção de Declarações
var
opcao,tipo:inteiro
nome, endereco:caracter
cpf,telefone:real
inicio
// Seção de Comandos
escreval("**Cadastro de cliente**")
escreval("1-Incluir")
escreval("2-Alterar")
escreval("3-Enviar")
leia(opcao)
se (opcao = 1) entao

```

```
escreval("Inclua o nome:")
leia(nome)
escreval("Inclua o endereço:")
leia(endereco)
escreval("Inclua o CPF:")
leia(cpf)
escreval("Inclua o Telefone:")
leia(telefone)
escreval("Nome:", nome)
escreval("Endereço:", endereco)
escreval("CPF:", cpf)
escreval("Telefone:", telefone)
senao
  se (opcao = 2) entao
    escreval("**Leve até o estabelecimento um comprovante de endereço**")
  senao
    se (opcao = 3) entao
      escreval (" Envie por Sedex no endereço de Caixa Postal 12098")
    fimse
  fimse
fimse
fimse
escreval ("**Agradecemos a preferência!**")
finalgoritmo
```



Lembre-se

Você pode desenvolver um algoritmo ótimo! Realize você mesmo uma possibilidade de implementação com estruturas de decisão encadeadas e pratique!



Faça você mesmo

Implemente este algoritmo em linguagem de programação C! Faça o download do Dev C++ e pratique!

Avançando na prática

Pratique mais!

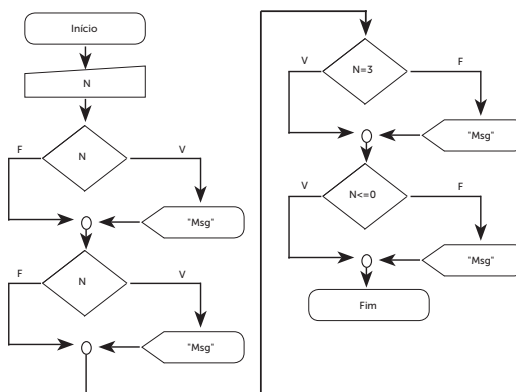
Instrução

Desafiamos você a praticar o que aprendeu transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois compare-as com as de seus colegas.

Estrutura condicional composta e encadeada

| | |
|--|---|
| 1. Competência de fundamentos de área | Conhecer os princípios e conceitos que envolvem o aprendizado em construção de algoritmos e programação e a sua importância para o universo do desenvolvimento de sistemas. |
| 2. Objetivos de aprendizagem | <ul style="list-style-type: none"> • Conhecer os conceitos e aplicações das estruturas de decisão "SE". • Conhecer como fazer uma estrutura condicional simples. • Conhecer e aplicar as estruturas condicionais compostas. • Conhecer e aplicar estruturas condicionais encadeadas. |
| 3. Conteúdos relacionados | <ul style="list-style-type: none"> • Estrutura condicional composta e encadeada. |
| 4. Descrição da SP | <p>Considerando o problema exemplificado anteriormente, que apresenta um algoritmo de controle de abastecimento de caixas d'água, desenvolva o seu diagrama de blocos (fluxograma):</p> <p>Retorne, na Seção 2.4 de autoestudo, em <i>Não Pode Faltar</i>, o exemplo:</p> <p>"Suponha que em uma cidade a empresa que faz o controle e tratamento de água e esgoto precisa controlar o nível de água das caixas d'água dos bairros, para que os controladores possam identificar se a caixa d'água está cheia, vazia, transbordando ou sem vazão. [...]".</p> |

5. Resolução da SP



Fonte: Manzano e Oliveira (2010, p. 67).



Lembre-se

Após a indicação da variável de tipo inteiro N , é solicitada a leitura de um valor para ela. Assim que a leitura é realizada e o valor é fornecido para a variável N , ocorre uma de quatro possibilidades. [...] O programa apresenta para o usuário uma mensagem informando a ocorrência, não importa o valor fornecido. (MANZANO; OLIVEIRA, 2010, p. 67).

Faça valer a pena!

1. Quanto às estruturas de decisão sequenciais e compostas, é correto afirmar o que está escrito na alternativa:
 - a. () estruturas de decisão sequenciais dependem exclusivamente de operadores relacionais.
 - b. () estruturas de decisão compostas podem ser encadeadas ou simples.
 - c. () estruturas de decisão não contemplam verificação de condições.
 - d. () estruturas de decisão sequenciais e encadeadas podem ser substituídas, em sua maioria, pela estrutura caso.
 - e. () estruturas de decisão encadeadas representam/são logicamente organizadas de forma igual às sequenciais.

2. Assinale a alternativa que apresenta o tipo de estrutura de decisão evidenciada abaixo:

```
se (N =1) entao
    escreval ("Caixa d'agua cheia.")
fimse
```

- a. () estrutura condicional simples.
- b. () estrutura condicional composta.
- c. () estrutura condicional sequencial.
- d. () estrutura condicional encadeada.
- e. () estrutura de seleção.

3. Assinale a alternativa que apresenta o tipo de estrutura de decisão evidenciada abaixo:

```
se (condição 1) entao
    comandos 1 para condição verdadeira
senao
    comandos 2 para condição falsa
fimse
```

- a. () estrutura condicional simples.
- b. () estrutura condicional composta.
- c. () estrutura condicional sequencial.
- d. () estrutura condicional encadeada.
- e. () estrutura de seleção.

4. Assinale a alternativa que apresenta o tipo de estrutura de decisão evidenciada abaixo:

```
se (tipo = 3) entao
    escreval("CPF:")
    leia(cpf)
senao
    se (tipo = 4)entao
```



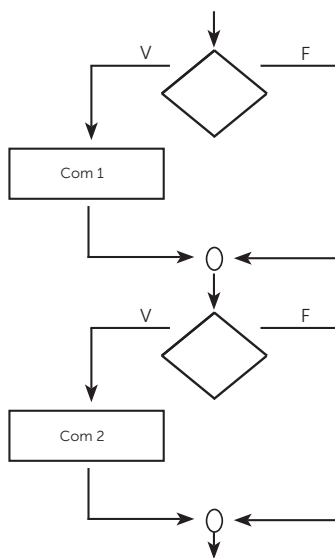
```

    escreval("Telefone:")
    leia (telefone)
  fimse
fimse

```

- a. () estrutura condicional simples.
- b. () estrutura condicional composta.
- c. () estrutura condicional sequencial.
- d. () estrutura condicional encadeada.
- e. () estrutura de seleção.

5. Analise a figura abaixo e assinale:



Refere-se a:

- a. () estrutura condicional simples sequencial.
- b. () estrutura condicional composta.
- c. () estrutura condicional sequencial.
- d. () estrutura condicional encadeada.
- e. () estrutura de seleção.

- 6.** Defina estruturas condicionais sequenciais.
- 7.** Explique a lógica da estrutura condicional encadeada.

Referências

SOUZA, Marco. **Algoritmos e lógica de programação**. 2. ed. São Paulo: Cengage Learning, 2013.

PIVA Jr. et. al. **Algoritmos e Programação de Computadores**. Rio de Janeiro: Elsevier, 2012.

Referências bibliográficas complementares:

1. EVARISTO, Jaime. **Aprendendo a Programar**: programando na linguagem C para iniciantes. Rio de Janeiro: Ed. Book Express, 2001.

2. ZIVIANI, Nivio. **Projeto de Algoritmos**: com implementações em Java e C++. São Paulo: Cengage Learning, 2011.

3. MANZANO, José A.N.G.. **Algoritmos**: Lógica para Desenvolvimento de Programação de Computadores. 24. ed. São Paulo: Érica, 2010.

4. AGUILAR, Luis J. **Programação em C++**. 2. ed. Porto Alegre. McGraw Hill, 2008. Disponível em: <<http://books.google.com.br/books?id=G4SaAgAAQBAJ&pg=PA16&dq=algoritmo&hl=pt-BR&sa=X&ei=nyeOU8XqMMHmsAT7jIGgBA&ved=0CDcQ6AEwAzgU#v=onepage&q=algoritmo&f=false>>. Acesso em: 03 jun. 2015.

5. SZWARCFITER, Jayme Luiz; MARKEZON, Lilian. **Estruturas de dados e seus algoritmos**. 3. ed. Rio de Janeiro: LTC – Livros Técnicos e Científicos, 2010.

ESTRUTURAS DE SELEÇÃO (CASE) E REPETIÇÃO

Convite ao estudo

Em algoritmos, você aprende que o sequenciamento das ações que o *software* deverá executar é o primeiro passo necessário para se transcrever as ações do sistema para a lógica computacional. A lógica computacional respeita as sintaxes de cada um dos ambientes de programação. Além disso, você também já deve ter observado que algumas estruturas não são suficientes ou não se apresentam como solução ótima a determinadas situações em que serão desenvolvidos produtos de *softwares*. Por esse motivo, é importante aprender como otimizar a programação, melhorando a codificação e empregando outras estruturas que permitem deixar o código mais enxuto, isto é, mais otimizado, com menos comandos encadeados, o que influencia, de modo geral, no processamento das informações.

A partir de agora, você será apresentado a mais algumas estruturas que permitirão desenvolver soluções mais enxutas ou ainda melhorar algumas propostas. Nesse sentido, e com o objetivo de apresentar outras técnicas de programação evidenciadas nos estudos de algoritmos, é que exemplos de aplicação das estruturas de seleção e de repetição serão abordados com mais ênfase. Então, retome a competência fundamental da área, que pretende desenvolver no decorrer dos seus estudos: “conhecer os princípios e conceitos que envolvem o aprendizado em construção de algoritmos e programação e a sua importância para o universo do desenvolvimento de sistemas”.

Além dessa, há os objetivos específicos desta unidade de ensino. Conheça-os:

- Saber identificar a necessidade ou a possibilidade de aplicar estruturas de seleção CASE/CASO, ou, como chamado em pseudocódigo em VisuAlg: "escolha".
- saber fazer a implementação de estruturas de repetição condicionais com teste no final;
- reconhecer, compreender e saber implementar as estruturas de repetição condicionais com teste no início;
- saber como aplicar as estruturas de repetição controladas por variáveis.

Lembre-se de que a situação que visa aproximar os conceitos com a prática profissional é a de desenvolvimento do aplicativo de busca de locais de gastronomia e hotelaria do Litoral Sul. É claro que esse é um exemplo, e, dadas as crescentes demandas pelo mercado de desenvolvimento de aplicativos, você pode transpor os conceitos às mais diversas situações da realidade profissional. Sendo assim, você pode pensar agora em principalmente implementar as estruturas de seleção e repetição para a escolha dos locais cadastrados previamente no sistema e disponibilizar essa opção ao usuário. É importante também que você desenvolva um algoritmo que contemple a venda de cupons para os locais indicados *on-line*. Desde já, bons estudos!

Seção 3.1

Estrutura de múltipla escolha (CASE)

Diálogo aberto

Ao desenvolver um algoritmo, um fator importante a considerar é que o profissional da área de análise e desenvolvimento de sistemas precisa ter empatia, ou seja, precisa se colocar no lugar do usuário para pensar na melhor solução para facilitar a usabilidade do software.

Pensando nisso você pode desenvolver soluções que apresentem, além de uma interface amigável, uma lógica de navegação que permita ao usuário compreender intuitivamente quais são as ações que ele encontrará, seja no sistema com que trabalha e realiza as suas rotinas diárias profissionais, ou, seja em um aplicativo que ele considera interessante para ter no seu celular, por exemplo. Contudo, as estruturas de seleção que serão aqui evidenciadas, representam mais uma alternativa dentre as existentes e aceitas pelas linguagens computacionais. Então, com o intuito de tornar a navegação e uso do aplicativo mais rápido, fácil e eficiente, sob o ponto de vista de processamento da informação inclusive, esta é uma boa opção de desenvolvimento. Nesse caso, você pode pensar em como implementar uma opção de reserva de data para o local escolhido. Considere o algoritmo desenvolvido na unidade 1, na seção 1.2, de escolha entre gastronomia e hotelaria como a ação inicial do aplicativo. Sendo assim, considere que na sequência as ações serão:

- Indicar um algoritmo que permita ao usuário, independente da opção escolhida entre gastronomia e hotelaria, realizar uma reserva em um local de sua escolha.

Você pode buscar outras possibilidades de aplicação das estruturas de seleção CASE, , por exemplo, pense em como é possível desenvolver uma plataforma que permita aos clientes realizar compras de ingressos online, ou mesmo, registrar a compra deste em um computador local, apresentando opção seja ao usuário final, seja em uma interface para o colaborador efetuar a venda do ingresso. Então, como observado, os conceitos aqui apresentados, podem ser aplicados às mais variadas atividades que envolvam soluções em produtos de software. Como nós estamos avançando nos estudos das estruturas possíveis para implementar os algoritmos trabalhados,

podemos também, transpor esses conhecimentos para pensar na automatização e sistematização de algumas ações que podem ser facilitadas ou mesmo, agilizadas se apoiadas a um sistema que processe, armazene e controle informações e dados. Agora que você já sabe o que precisa ser feito, inicie as atividades de estudos apoiado pelo seu material didático e as instruções dos demais materiais. No livro didático, você tem os conteúdos abordados de forma simples e prática. Com isso você pode ter o contato com o conteúdo conceitual e praticar um pouco com os exemplos e exercícios propostos. Então, siga em frente e bons estudos!

Não pode faltar

Você se recorda do exemplo abordado na unidade de ensino anterior, em que é apresentada uma sugestão que contém a estrutura de seleção CASE juntamente com as estruturas de decisão? Agora, você pode aprofundar mais os seus conhecimentos sobre essa estrutura. Reveja abaixo e acompanhe o procedimento:

```

algoritmo "calculadora simples"
//seção de declarações
var
x, y: real
opcao: caractere
inicio //seção de comandos
  Escreval ("Digite dois números e informe se deseja (A)dição, (M)ultiplicação, (Sub)tração ou (D)ivisão:")
  Leia (x, y, opcao)
  escolha opcao
    caso "S"
      Escreval (x, opcao, y, "=", x+y)
    caso "M"
      Escreval (x, opcao, y, "=", x * y)
    caso "Sub"
      Escreval (x, opção, y, "=", x - y)
    caso "D"
      se (y <= 0) entao
        Escreval ("Informe número diferente de o (zero)!")
      senao
        Escreval (x, opcao, y, "=", x / y)
      fimse
    outrocaso
      Escreval ("Opção inválida!")
  fimescolha
fimalgoritmo

```

Primeiramente, há, como convenção, a apresentação do nome do algoritmo e, em seguida, inicia-se a declaração das variáveis. Na sequência, temos o início do algoritmo efetivamente, com a exibição de uma mensagem ao usuário, solicitando

que ele informe uma das opções apresentadas. A leitura da variável "opcao" será essencial para que o sistema compreenda qual dos comandos CASE ou CASO, como evidenciado no pseudocódigo, deverá ser executado.

Para o caso de a opção escolhida ser a indicação "A", o sistema exibirá apenas a mensagem e o valor da soma. Para a segunda opção "M", o sistema entende que será exibida a multiplicação dos números solicitados. No caso da opção ser "SUB", o sistema apresentará o resultado da subtração e, para o caso de o usuário inserir a letra "D" como opção de ação que deseja executar, o sistema fará exatamente a sequência de comandos descrita na respectiva estrutura de seleção, que compreende a verificação de uma condição, realizada por uma estrutura de decisão composta "se-então-senão".

O comando "outrocaso" indica que se o usuário digitar uma informação que não está prevista dentre as opções apresentadas, o sistema exibirá uma mensagem informativa. Não se pode esquecer de finalizar as estruturas iniciadas, como indicam os comandos: "fimse", "fimescolha" e "fimalgoritmo".



Refleta

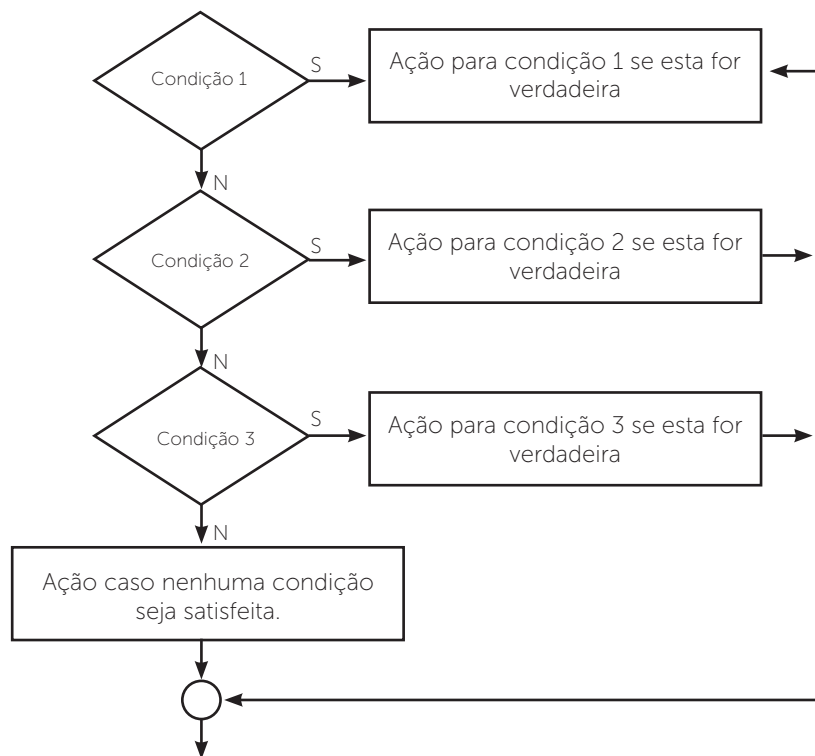
A tomada de decisão por seleção é uma alternativa mais rápida ao uso de tomadas de decisão sequenciais ou mesmo encadeadas. Essa estrutura lógica de condição é útil e pode ser usada em situações em que se possui um grande número de verificações. Essa estrutura é um tanto limitada [...] (MANZANO; OLIVEIRA, 2010, p. 71).

Observe que a forma como foi abordada a estrutura de seleção CASO pode ser também ilustrada pelo diagrama de blocos ou fluxograma abaixo:



Assimile

Figura 8 – Fluxograma da estrutura de seleção



Fonte: Adaptado de Manzano e Oliveira (2010, p. 71).

- Observe como é que ocorre o processamento da informação, a partir da escolha do usuário;
- não há a execução sequencial, e sim a declaração dos comandos de forma sequencial, porém esse comando de seleção permite que o compilador compreenda que deverá executar apenas as ações pertinentes àquele bloco de comandos;
- fato esse que acelera o processamento e apresentação do resultado.

Conheça agora a estrutura sintática das estruturas de seleção CASO:

caso <nome da variável>

seja <opção 1> **faça**

[neste bloco vêm os comandos a executar para a condição 1 se esta for verdadeira]

seja <opção 2> **faça**

[neste bloco vêm os comandos a executar para a condição 2 se esta for verdadeira]

seja <opção 3> **faça**

[neste bloco vêm os comandos a executar para a condição 3 se esta for verdadeira]

outrocaso

[ação para o caso de nenhuma condição ser satisfeita]

fimcaso



Pesquise mais

Este *link* traz uma apostila que sintetiza as práticas e como aplicar os conceitos de lógica de programação, estruturas e comandos que são usados no Scilab. Disponível em: <<http://www.dca.ufrn.br/~estefane/academica/progsci.pdf>>. Acesso em: 9 jun. 2015.

Observe que a estrutura sintática determina exatamente como será a declaração de tal estrutura de seleção. Antes de iniciar o desenvolvimento e prática dos exemplos desta seção, a sugestão é que você realize o acesso a outros materiais didáticos que contêm informações de como esta estrutura é utilizada nas plataformas computacionais.

O *link* indica uma apostila em que há a aplicação de todas as estruturas vistas até o momento em Scilab. Investigue também informações de como representá-las em VisuAlg, em C, em C++, em Java, entre outras.



Exemplificando

Vamos seguir um exemplo sugerido por Manzano e Oliveira (2010, p. 72-73), em que há a apresentação do português estruturado, o diagrama de blocos e a estrutura do algoritmo em linguagem natural.

Então, vamos ao exercício:

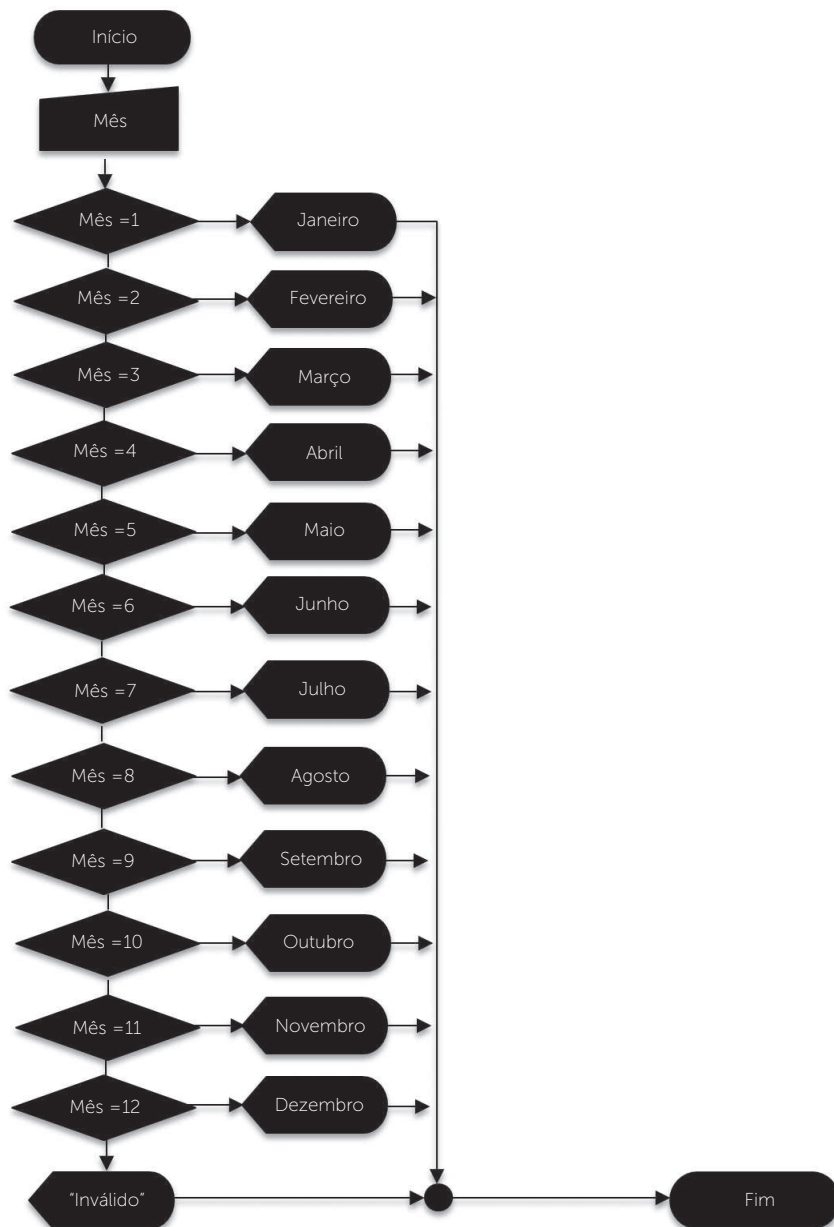
Desenvolver um programa de computador que leia um valor numérico inteiro entre os valores 1 e 12 e apresente, por extenso, o nome do mês correspondente ao valor inserido pelo usuário. Caso sejam fornecidos valores menores do que 1 (um) e maiores que 12 (doze), o programa deve apresentar a mensagem de valor inválido.

Veja a solução:

1. Efetuar a leitura de um valor numérico inteiro (variável MÊS).
2. Se a variável MÊS for igual a 1, apresentar a mensagem "janeiro".
3. Se a variável MÊS for igual a 2, apresentar a mensagem "fevereiro".
4. Se a variável MÊS for igual a 3, apresentar a mensagem "março".
5. Se a variável MÊS for igual a 4, apresentar a mensagem "abril".
6. Se a variável MÊS for igual a 5, apresentar a mensagem "maio".
7. Se a variável MÊS for igual a 6, apresentar a mensagem "junho".
8. Se a variável MÊS for igual a 7, apresentar a mensagem "julho".
9. Se a variável MÊS for igual a 8, apresentar a mensagem "agosto".
10. Se a variável MÊS for igual a 9, apresentar a mensagem "setembro".
11. Se a variável MÊS for igual a 10, apresentar a mensagem "outubro".
12. Se a variável MÊS for igual a 11, apresentar a mensagem "novembro".
13. Se a variável MÊS for igual a 12, apresentar a mensagem "dezembro".
14. Se a variável MÊS for menor do que 1 ou maior do que 12, apresentar a mensagem "Valor inválido".

Veja agora a representação desse algoritmo em diagrama de blocos:

Figura 9 – Fluxograma da estrutura de repetição II



Fonte: Adaptado de Manzano e Oliveira (2010, p. 71).



Faça você mesmo

```

Português Estruturado
algoritmo Verifica Mês
var
    Mês: inteiro
início
    leia (Mês)
    caso (Mês)
        seja 1 faça
            escreva "Janeiro"
        seja 2 faça
            escreva "Fevereiro"
        seja 3 faça
            escreva "Março"
        seja 4 faça
            escreva "Abril"
        seja 5 faça
            escreva "Maio"
        seja 6 faça
            escreva "Junho"
        seja 7 faça
            escreva "Julho"
        seja 8 faça
            escreva "Agosto"
        seja 9 faça
            escreva "Setembro"
        seja 10 faça
            escreva "Outubro"
        seja 11 faça
            escreva "Novembro"
        seja 12 faça
            escreva "Dezembro"
    senão
        escreva ("Valor inválido!")
    fim_caso
fim
  
```

Lembre-se de que o comando **senão** é opcional nas estruturas de tomada de decisão por seleção. Observe também que a instrução **caso/seja...faça/senão/fim_caso** é recomendada apenas para soluções em que seja necessário implementar tomadas de decisão encadeadas ou sequenciais. Nesse caso, as ações do algoritmo são previstas a partir do uso do operador relacional "igual a"

(MANZANO; OLIVEIRA, 2010).

Sem medo de errar



Atenção!

Veja abaixo uma proposta de solução em VisuAlg para implementar a estrutura de seleção para o procedimento de solicitação de reserva:

```
algoritmo "Reserva"
var
    Dia,Mes,Ano, dia_aniversario, mes_aniversario: inteiro
inicio
    // Seção de Comandos

    escreva("Digite a data da reserva (dia, mês, ano): ")
    leia(Dia, Mes, Ano)

    escolha Mes

        caso 1
            escreval("Data: ", Dia, " de janeiro de ", Ano)
        caso 2
            escreval("Data: ", Dia, " de fevereiro de ", Ano)
        caso 3
            escreval("Data: ", Dia, " de marco de ", Ano)
        caso 4
            escreval("Data: ", Dia, " de abril de ", Ano)
        caso 5
            escreval("Data: ", Dia, " de maio de ", Ano)
        caso 6
            escreval("Data: ", Dia, " de junho de ", Ano)
        caso 7
            escreval("Data: ", Dia, " de julho de ", Ano)
        caso 8
            escreval("Data: ", Dia, " de agosto de ", Ano)
        caso 9
            escreval("Data: ", Dia, " de setembro de ", Ano)
        caso 10
            escreval("Data: ", Dia, " de outubro de ", Ano)
        caso 11
            escreval("Data: ", Dia, " de novembro de ", Ano)
```

```

        caso 12
        escreval("Data: ", Dia, " de dezembro de ", Ano)
    outrocaso
        escreval("Mes invalido!")
    fimsecolha
se (dia_aniversario == Dia) e (mes_aniversario == Mes) entao
    escreval("Pelo seu aniversário, Nós te presentamos com a sua
solicitação!")
    escreval("Para maiores informações entre em contato.")
fimse
finalgoritmo

```



Lembre-se

*O mesmo raciocínio aplicado à primeira condição se aplica às demais condições previstas [...]. Se uma das condições gerar resultado lógico falso, o fluxo do programa é desviado pela linha sinalizada pelo rótulo **N** para a próxima condição de avaliação. Se nenhuma das condições for satisfeita, executa-se a última ação antes e em cima do símbolo conector (MANZANO; OLIVEIRA, 2010, p. 71).*

Avançando na prática

Pratique mais

Instrução

Desafiamos você a praticar o que aprendeu, transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois as compare com a de seus colegas.

Título da situação-problema:

| | |
|---------------------------------------|--|
| 1. Competência de fundamentos de área | Conhecer os princípios e conceitos que envolvem o aprendizado em construção de algoritmos e programação e a sua importância para o universo do desenvolvimento de sistemas. |
| 2. Objetivos de aprendizagem | <ul style="list-style-type: none"> Saber identificar a necessidade ou a possibilidade de aplicar estruturas de seleção CASE; saber fazer a implementação de estruturas de repetição condicionais com teste no final; reconhecer, compreender e saber implementar as estruturas de repetição condicionais com teste no início; saber como aplicar as estruturas de repetição controladas por variáveis. |

| | |
|---------------------------|--|
| 3. Conteúdos relacionados | Estrutura de múltipla escolha (CASE) |
| 4. Descrição da SP | A partir da proposta desenvolvida no VisualG, que permite a inclusão de uma data de solicitação de reserva, implemente o algoritmo usando preferencialmente o Dev C++. Veja o exemplo de como transpor para a sintaxe da linguagem C. |
| 5. Resolução da SP | <pre> #include <stdio.h> #include <conio.h> void main() { int Dia,Mes,Ano; printf("Digite uma data (dia mes ano - separados por um espaço):"); scanf("%d%d%d", &Dia, &Mes, &Ano); switch(Mes) { case 1: printf("\nData: %d de janeiro de %d", Dia, Ano);break; case 2: printf("\nData: %d de fevereiro de %d", Dia, Ano);break; case 3: printf("\nData: %d de março de %d", Dia, Ano);break; case 4: printf("\nData: %d de abril de %d", Dia, Ano);break; case 5: printf("\nData: %d de maio de %d", Dia, Ano);break; case 6: printf("\nData: %d de junho de %d", Dia, Ano);break; case 7: printf("\nData: %d de julho de %d", Dia, Ano);break; case 8: printf("\nData: %d de agosto de %d", Dia, Ano);break; case 9: printf("\nData: %d de setembro de %d", Dia, Ano);break; case 10: printf("\nData: %d de outubro de %d", Dia, Ano);break; case 11: printf("\nData: %d de novembro de %d", Dia, Ano);break; case 12: printf("\nData: %d de dezembro de %d", Dia, Ano);break; default: printf("\nMes invalido!"); } getch(); } </pre> <p>Fonte: (PIVA JR., 2013, p. 27)</p> |



Lembre-se

Quando implementar em linguagem C, se usar o Dev C++, você precisará salvar o seu arquivo com a extensão ".c" para que não tenha problemas ao compilar esse programa futuramente.



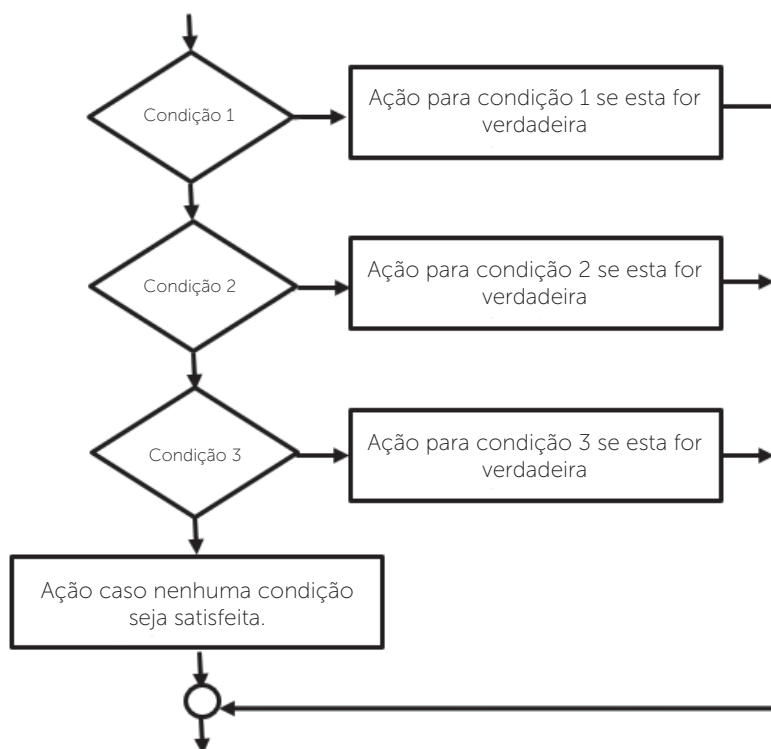
Faça você mesmo

Leia o material de apoio disponível no *link* do Scilab: <<http://www.scilab.org/fr/resources/documentation/tutorials>>. Acesso em: 15 jun. 2015. E tente desenvolver o seu algoritmo nesta ferramenta!

Assista também à videoaula: <<https://www.youtube.com/watch?v=wWJ9FVZ7l3M>>. Acesso em: 15 jun. 2015.

Faça valer a pena

1. A figura abaixo refere-se à qual estrutura de programação?



- a) ☐ Estrutura primitiva.
- b) ☐ Estrutura condicional composta.
- c) ☐ Estrutura de seleção.
- d) ☐ Estrutura de decisão simples.

e) () Estrutura de decisão sequencial com seleção.

2. Escreva a forma sintática da estrutura de seleção que deverá ser seguida para implementação.

3. Analise o algoritmo abaixo e assinale a alternativa correta.

```

Leia (x, y, opcao)
escolha opcao
    caso "S"
        Escreval (x, opcao, y, "=", x+y)
    caso "M"
        Escreval (x, opcao, y, "=", x * y)
    caso "Sub"
        Escreval (x, opção, y, "=", x - y)
    caso "D"
        se y <= 0 entao
            Escreval ("Informe número
                        diferente de o (zero)!")
        senao
            Escreval (x, opcao, y, "=", x / y)
        fimse
fimescolha

```

a) () A estrutura de seleção permite, além da escolha de opções, inserir verificações que pertencem ao processo escolhido. Por esse motivo, o processamento pode ser otimizado.

b) () A estrutura de seleção não comporta uma nova verificação através de estrutura de decisão.

c) () Não há a necessidade de definição de uma variável padrão para leitura e validação da opção escolhida.

d) () O encerramento da estrutura de decisão "fimse" deve ser inserido, no caso do exemplo, após o encerramento da estrutura de seleção.

e) () É possível inserir apenas 8 situações de escolha em estruturas de seleção.

4. Qual é a relação existente entre as estruturas de seleção, decisão e os operadores relacionais?

5. Desenvolva um algoritmo que, com uma estrutura de seleção, indique a realização de uma verificação da quantidade de homens e mulheres que visitaram uma feira de tecnologia realizada no Estado de SP.

6. Dado o algoritmo abaixo, assinale se as afirmações são verdadeiras ou falsas.

```

escolha sexo
  caso "H"
    h <- h + 1
  caso "M"
    m <- m + 1
  outrocaso
    escreval("Sexo só pode ser H ou M!")
fimescolha

```

- a) () A estrutura de seleção não pode conter atribuição de valores ou cálculos nos blocos de comandos.
- b) () Podem conter apenas a exibição de mensagens.
- c) () Pode ainda ter um “senão” que complete a verificação que será realizada, sendo este não obrigatório.
- d) () Cabe às estruturas de seleção contabilizar, nesse caso, de acordo com uma quantidade limitada de inserções.
- e) () Estruturas de seleção são viáveis apenas para situações em que há tomada de decisão.

7. Analise a frase abaixo e assinale a alternativa que melhor completa as lacunas.

O comando “_____” indica que se o usuário digitar uma informação que não está prevista dentre as opções apresentadas, o sistema exibirá uma mensagem informativa. Não se pode esquecer de finalizar as estruturas iniciadas, como indicam os comandos: “_____”.

- a) fimcaso/fimescolha
- b) fimescolha/outrocaso
- c) outrocaso/fimescolha
- d) fimse/fimescolha
- e) fimalgoritmo/outrocaso

Seção 3.2

Repetição condicional com teste no início

Diálogo aberto

Estamos iniciando uma etapa muito importante dos estudos e que permitirá o desenvolvimento de melhores soluções em algoritmos. Por esse motivo, você precisa ter compreendido qual a importância de cada uma das etapas anteriores, principalmente para darmos continuidade à lógica computacional das estruturas de decisão e de seleção.

Então, já que estamos evoluindo, teremos de apresentar mais uma etapa do projeto idealizado. Nesse sentido, você já deve estar apto a identificar melhorias, processos mais ágeis em desenvolvimento, integrando a lógica computacional com a necessidade dos negócios e transações comerciais que a cada dia são renovados a partir do auxílio de um produto de *software*, como, por exemplo, um aplicativo. Para melhorar esse processo, recomenda-se o uso das estruturas de repetição, também conhecidas como laços ou *loops*. Essa associação a laços se dá justamente porque é inserido um processo que determina a quantidade de vezes em que será necessário realizar um bloco de comandos.

Pensando em desenvolver para o protótipo uma melhoria nas estruturas já apresentadas, de forma a otimizar e reduzir o tamanho deste código, podemos dizer que, se inserirmos uma estrutura de repetição, será possível atingir um nível de processamento mais elevado, além de deixar o código fonte mais organizado. Para tal, é preciso conhecer quais são as estruturas de repetição e como se dá a execução dos comandos a partir da inserção de repetições.

Estudaremos os seguintes laços de repetição:

- os que acontecem no início do bloco de comandos e determinam a sua execução, apenas **enquanto** determinada restrição ou condição for verdadeira. Contempla o conjunto de instruções **enquanto, faça e fim enquanto**;
- o laço de repetição que é conhecido como **repita**, que executará um bloco de instruções ao menos uma vez antes da verificação da condição. Esse comando contempla o conjunto **repita... até que**;

- o terceiro laço estudado será o **para**, que poderá agregar às outras estruturas uma limitação de repetições, ou seja, insere um número finito de repetições. O conjunto de comandos da estrutura de repetição para é: **para... de... até... passo... faça... fim_para**.

Conheça, a seguir, algumas definições e exemplos com a estrutura de repetição **enquanto, faça e fim enquanto**.

Não pode faltar

Olá, aluno. Vamos, a partir de agora, conhecer e trabalhar com a sintaxe da estrutura de repetição **enquanto**. Observe abaixo como é a sintaxe em português estruturado:

```

enquanto (condição) faça
    <conjunto de instruções a executar caso a condição seja
verdadeira>
fim_enquanto
  
```



Assimile

Siga as dicas de Piva Jr. (2012, p. 204) para a elaboração de seu algoritmo usando a estrutura de repetição enquanto:

- *o bloco de comando se inicia com a palavra-chave **enquanto** e termina com o **fimenquanto**;*
- *o comando utiliza outra palavra-chave: **faça**, sem a cedilha, sem a expressão lógica;*
- *a expressão-lógica não precisa estar entre parênteses.*

Já na sintaxe dessa estrutura na linguagem de programação que estamos trabalhando, C, na plataforma de desenvolvimento Dev C++, o que se altera é justamente a forma de escrever. Em C, a palavra enquanto será substituída por **while**. Veja a seguir a sua estrutura sintática em C:

while (expressão lógica)

```
{
    comando;
    .
    .
    comando;}
```



Assimile

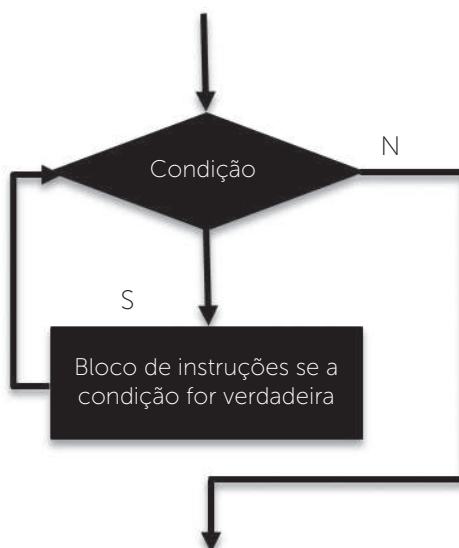
Note que não há a necessidade de inserção de indicação de um comando que substitua o fim_ enquanto para esse caso. No entanto, há a necessidade de abrir e fechar o bloco com o uso das chaves "{ }".

Esse tipo de laço pode ser identificado como "laço de teste lógico no início". Isso ocorre, pois há, primeiramente, a inserção do comando que indica o tipo de laço que será executado, seguido de uma condição. Apenas após essa verificação será iniciado o comando **faça**, a sequência de execução dos comandos do bloco de instruções da estrutura de repetição.

No diagrama, é possível notar que as instruções do bloco de comandos subsequentes apenas serão executadas caso a condição ou expressão lógica verificada seja verdadeira. E, se essa for falsa, a execução será dos comandos que seguem imediatamente fora do laço.

O seu diagrama de blocos é representado da seguinte forma:

Figura 10 – Diagrama de blocos do laço de repetição enquanto



Fonte: Manzano e Oliveira (2010, p. 89).



Refleta

A estrutura de repetição **enquanto...faça...fim_enquanto** tem o funcionamento controlado por decisão, e pode executar certo conjunto de instruções enquanto a condição verificada for Verdadeira. No momento em que essa condição se torna Falsa, o processamento da rotina é desviado para fora do laço. Se a condição for Falsa logo de início, as instruções do laço são ignoradas (MANZANO; OLIVEIRA, 2010, p. 89).

Com as estruturas de repetição, de modo geral, é melhorada a dinâmica de processamento e execução do programa, pois não é necessário inserir um conjunto de instruções para a mesma verificação, validação ou controle mais do que uma vez. Há a necessidade de desenvolvimento dessa estrutura apenas uma vez e será, conseqüentemente, executada sempre que a condição imposta for verdadeira.

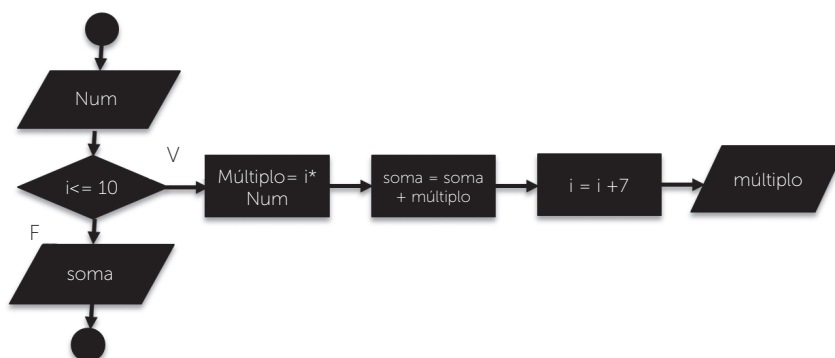


Exemplificando

Escreva um programa que imprima na tela os dez primeiros múltiplos de um número inteiro qualquer fornecido pelo usuário (lido). No final, imprima também a soma desses dez números.

Veja abaixo como o autor desenvolveu o diagrama de blocos para esse programa e a sua solução, tanto em VisuAlg quanto em C:

Figura 11 – Fluxograma estrutura de repetição enquanto



Fonte: Piva Jr. (2012, p. 215).

Agora, vamos fazer o pseudocódigo dessa operação. Siga em frente!

algoritmo "Múltiplos de número lido".

//Seção de Declarações

var

numero, soma, múltiplo, i: **inteiro**.

inicio

//seção de comandos

escreva ("Digite um número:")

leia (numero)

escreval ("Valor lido:", numero)

escreva ('Lista de Múltiplos:')

i <- 1 // i = 1

soma <- 0

enquanto (i <= 10) **faca**

 múltiplo <- i * numero

escreva (múltiplo, " ")

 soma <- soma + múltiplo

 i <- i+1

fimenquanto

escreval ()

escreval ("Soma = ", soma)

finalgoritmo

O algoritmo apresentado pode ser útil para que se aprimore o algoritmo da Unidade 2, na Seção de autoestudo 2.1, em que você precisou contabilizar os acessos das categorias gastronomia e hotelaria. Utilize um contador e siga os passos para esse desenvolvimento:

a) para esse caso, você pode ainda inserir uma estrutura de seleção para que o usuário informe a opção "G" ou "H". Já está pronta no material em 2.1;

b) mas, antes, é necessário verificar se este usuário não é um comerciante ou cooperado, o que está sugerido na Seção de autoestudo 2.2;

c) para melhorar o algoritmo proposto, você pode inserir uma verificação com a estrutura de repetição enquanto, sendo que pode realizar o procedimento de contabilizar os acessos para cada categoria, de acordo com o tipo de usuário. **Enquanto** ele for o usuário final, e, sem permissões de controle, apenas de leitura e algumas verificações, o sistema contabilizará os acessos para cada categoria.

d) e, em seguida, abrirá apenas uma tela em que seja possível procurar o local de acordo com a categoria escolhida.



Faça você mesmo

Faça você mesmo essa integração. Tente implementar essa melhoria no protótipo que está sendo proposto de algoritmo de busca. Lembre-se também de que essa é apenas uma das ações deste programa. Primeiramente em VisuAlg para testar a lógica escolhida, depois transcreva na sintaxe da linguagem C.

É possível testar a entrada de dados e também solicitar ao usuário que insira um valor válido. Enquanto esse não digitar um valor que seja coerente para a sequência de execução dos comandos, e que atenda à condição, uma mensagem será exibida ao usuário.

Nesse caso, é melhor iniciarmos a exemplificação aplicando esses conceitos. Então, considere o exercício a seguir, primeiramente desenvolvido o seu pseudocódigo em VisuAlg.

Vamos seguir um exemplo de Piva Jr., 2012, p. 215-218:



Pesquise mais

Assista à videoaula: <<https://www.youtube.com/watch?v=8-JWuzb-gIE>>. Acesso em: 15 jun. 2015 e aprimore os seus conhecimentos.

Agora, você pode treinar também a estrutura de repetição de acordo com o exemplo de Piva Jr. (2012) implementado em linguagem C. Vamos lá para mais uma prática importante de fixação.

```
#include <stdio.h>
void main()
{
    int numero, soma, múltiplo, i;
    printf ("Digite um número:");
    scanf ("%d", numero);
```

```

printf("Valor lido: %d\n", numero)
printf("Lista de Múltiplos:");
soma=0;
i = 1;
while (i <= 10)
{
    múltiplo = i * numero;
    printf ("%d", múltiplo);
    soma += múltiplo;
    i++;
}
printf ("\nSoma =%d", soma);
}

```

Vamos à interpretação do algoritmo acima: a função **void main()** indica que essa é a principal e que os demais procedimentos ou funções implementados serão chamados à execução a partir dessa função. O uso da expressão **void** indica que a função não retornará valores, no entanto, haverá apenas a exibição de uma mensagem na tela.

Na sequência, há a declaração das variáveis com o seu respectivo tipo de dado. Depois, iniciam-se os comandos. O primeiro deles (`printf`) solicita ao usuário que informe um número. O sistema faz a leitura da variável indicada para armazenamento e exibe na tela o valor que foi inserido pelo usuário para simples conferência e uma mensagem que informa que será relacionada uma lista de números que serão identificados como múltiplos do valor digitado.

Depois dessas execuções, são inicializadas as variáveis "soma" e "i" para que não resulte em um erro de execução por não conterem valores. Em seguida, o algoritmo apresenta a estrutura de repetição que, a partir da condição, verifica e indica a sequência de comandos que serão executados enquanto a quantidade de repetições for menor ou igual a 10.

O comando **while**, nesse caso, indica que a repetição ocorrerá dez (10) vezes. Os comandos pertinentes a essa estrutura serão executados enquanto essa quantidade não ultrapasse o limite estabelecido. Para que a variável "múltiplo" possa realizar a operação e apresentar o seu respectivo resultado, a cada execução dessa estrutura a variável "i" sempre será atualizada pelo contador e, com isso, será exibido o múltiplo correspondente. Ao término das execuções da estrutura de repetição, será apresentada a soma dos números.

A condição (`i<=10`) precisa ser estabelecida por uma questão de convenção da

estrutura de repetição **enquanto/ while**. Pois, caso ela não exista, o compilador não saberá identificar quantas vezes deverá executar aquela operação. Por esse motivo, o uso dessa estrutura de repetição **enquanto** é viável para determinar essa limitação. Além disso, apresenta-se como uma boa opção para quem deseja otimizar o processamento das informações.



Assimile

Se você implementar o algoritmo acima em C, siga as recomendações de Piva Jr. (2012, p. 205):

- *O comando se inicia com a palavra-chave while.*
- *A <expressão – lógica> deve estar entre parênteses.*
- *Para a repetição de um conjunto de comandos, é necessário colocá-los dentro de um bloco de comandos, isto é, entre chaves {}.*

Sem medo de errar

Incremente uma função ao sistema que permita ao usuário do aplicativo proposto:

a) efetuar a reserva em um hotel, quantas vezes desejar dentro de um mesmo mês e no mesmo acesso. Para tal, crie uma estrutura de repetição que solicite a cada 3 reservas uma confirmação de que o cliente deseja continuar com o procedimento de reserva enquanto a resposta for "Sim";

b) Se a resposta for diferente de "Sim", encerra o procedimento.

Português Estruturado

```
algoritmo "Pré-reserva"
// Função : estabelecer rotina de pré- reserva
// Autor : JJJ
// Data : 16/06/2015
// Seção de Declarações
var
telefone, CPF, reserva: caractere
ano, dia, mes,numReserva: inteiro
inicio
```

```
// Seção de Comandos
escreval ("**Sistema de Pré-reserva Hotéis**")
escreval ("Informe o seu CPF:")
leia(CPF)
escreval ("Confirme o número do seu telefone celular:")
leia (telefone)
numReserva <- 0
reserva <- "sim"
enquanto (numReserva <=3) e (reserva = "sim") faça
    escreval("Informe o mês: 1 a 12")
    leia (mes)
    escreval("Informe o dia:")
    leia (dia)
    escreval("Deseja realizar outra reserva?")
    leia (reserva)
    numReserva <- numReserva +1
fimenquanto
escreval ("Solicitação de Reservas concluída, aguarde contato.")
escreval ("Quantidade de reservas realizadas:", numReserva)
finalgoritmo
```



Atenção!

- a) Você pode elaborar uma rotina de verificação de ano, se ele é bissexto;
- b) se for, o mês de fevereiro aceitará até o dia 29 a solicitação de reservas.



Lembre-se

Essa estrutura de repetição servirá para repetir um comando ou um conjunto de comandos enquanto a condição determinada for verdadeira.

Avançando na prática

| Pratique mais | |
|--|--|
| <p align="center">Instrução</p> <p>Desafiamos você a praticar o que aprendeu, transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois as compare com a de seus colegas.</p> | |
| "Repetição condicional com teste no início" | |
| 1. Competência de fundamentos de área | Conhecer os princípios e conceitos que envolvem o aprendizado em construção de algoritmos e programação e a sua importância para o universo do desenvolvimento de sistemas. |
| 2. Objetivos de aprendizagem | <ul style="list-style-type: none"> • Saber identificar a necessidade ou a possibilidade de aplicar estruturas de seleção CASE; • reconhecer, compreender e saber implementar as estruturas de repetição condicionais com teste no início; • saber fazer a implementação de estruturas de repetição condicionais com teste no final; • saber como aplicar as estruturas de repetição controladas por variáveis. |
| 3. Conteúdos relacionados | Repetição condicional com teste no início. |
| 4. Descrição da SP | Elabore o algoritmo do programa dado abaixo em linguagem de programação C, em pseudocódigo em VisuAlg. |
| 5. Resolução da SP | <pre> /*Programa que determina o menor divisor próprio de um inteiro*/ #include <stdio.h> #include <math.h> main() { float r; int num, divisor, i; num =1; while (num != 0) { printf("Digite um numero (0 para encerrar):"); scanf("%d", &num); divisor = 0; r = sqrt (num); i =2; while (i<=r) { </pre> |

| | |
|--------------------|--|
| 5. Resolução da SP | <pre> if (num % i == 0) { divisor = i; i = num; } else i = i + 1; if (divisor != 0) printf ("%d e' divisor próprio de %d \n", divisor, num); } } } </pre> |
|--------------------|--|



Lembre-se

"Outra aplicação importante do comando **while** diz respeito às aplicações sucessivas de um programa. [...] Se quisermos a sua execução para outra entrada, precisamos executar o programa de novo" (EVARISTO, 2001, p. 62)



Faça você mesmo

Elabore o diagrama de blocos desse algoritmo.

Faça valer a pena

1. Assinale a alternativa que apresenta a sintaxe correta para a estrutura de repetição enquanto:

- a) faça <comandos> enquanto
- b) enquanto <comandos> faça
- c) enquanto <comandos> faça <comandos> fim_enquanto
- d) enquanto <condição> faça <comandos> fim_enquanto.
- e) while/do<comandos>end while.

2. Dadas as afirmações abaixo, assinale a alternativa correspondente.

I – O bloco de comando se inicia com a palavra-chave enquanto e termina com o fim enquanto.

II – O comando utiliza outra palavra-chave: faça, sem a cedilha, sem a expressão lógica.

III – A expressão-lógica não precisa estar entre parênteses.

a) V – V – V.

b) F – F – V.

c) V – F – V.

d) F – V – F.

e) F – F – F.

3. Dado o comando while ($i \leq 10$), assinale a alternativa correspondente.

a) Há uma atribuição do valor 10 à variável i.

b) i é um contador que será executado apenas nessa instrução.

c) O comando enquanto (while) representa a repetição a partir da validação da condição determinada que, no caso, incide em executar um conjunto de comandos subsequentes, se esta for verdadeira.

d) O comando while indica que haverá uma verificação ao encerrar o bloco de comandos.

e) O comando while retorna o valor 0 após a execução dos programas.

4. Explique a estrutura de repetição enquanto.

5. Leia o trecho abaixo e assinale a alternativa que melhor descreve a explicação dada.

“Uma outra aplicação importante do comando while diz respeito às aplicações sucessivas de um programa. [...] Se quisermos a sua execução para outra entrada, precisamos executar o programa de novo” (EVARISTO, 2001, p. 62).

a) Indica que o comando while será executado apenas durante aquele

respectivo processamento.

b) Significa que o comando executará a sequência de comandos para todas as entradas de dados do programa.

c) Representa a função do comando de executar repetidas vezes um conjunto de instruções dadas, desde que a entrada de dados seja diferente de 0.

d) Representa o comando de atribuição para a condição ser validada.

e) O comando while requer uma entrada de dados inserida manualmente pelo usuário para todas as verificações do programa.

6. Dado o trecho de algoritmo abaixo, assinale a alternativa que contém os mesmos comandos utilizados, porém, em linguagem C.

```

"enquanto programa <> "N" faça
    escreva("Digite o valor do carro: ")
    leia(valor)
    escreva("Digite o ano do carro: ")
    leia(ano)
    se ano <= 2000 entao
        desconto <- 0.12
        total2000 <- total2000 + 1
        total <- total + 1
    senao
        desconto <- 0.07
        total <- total + 1
fimse
escreva("Deseja continuar calculando? (S) Sim - (N) Não - ")

```

```
leia(opc)
escolha opc
    caso "S"
        programa <- "S"
    caso "N"
        programa <- "N"
    outrocaso
        programa <- "S"
    limpatela
    escreval("As opções disponíveis são apenas S ou N!!!")
    leia(opc)
fimescolha
fimenquanto
```

- a) if/else, switch, while.
- b) else, if, for.
- c) for, if/else, while.
- d) while, if/else, switch.
- e) switc, if, else.

7. Elabore um algoritmo que receba uma "X" quantidade de números e mostre se estes são: positivos, negativos ou zero.

Seção 3.3

Repetição condicional com teste no final

Diálogo aberto

Temos construído, ao longo dos estudos, abordagens que nos levam à compreensão de quais são as estruturas necessárias em desenvolvimento de *software* para que estes sejam projetados.

Nesse sentido, cabe salientar que, para todo projeto, há a necessidade de, além do desenvolvimento de algoritmos, que é parte do todo, que seja realizada a análise do projeto e todo ele respaldado nas premissas que a engenharia de *software* recomenda como boas práticas. Então, não se esqueça de que aqui estão concentradas, modularmente, apenas a teoria e a apresentação de sua aplicação para que seja possível transpor para situações do cotidiano profissional, e ainda que estejam em conformidade com as necessidades de implantação de sistemas que foi contratada.

Pensando nisso, como temos trabalhado no desenvolvimento de um protótipo, precisamos inserir algumas funcionalidades. Nesse caso, um aplicativo como o que está sendo proposto pode conter várias, porém, diante do tempo disponível para desenvolvimento, como você pensaria na oferta deste produto de *software* ao seu cliente?

Vamos supor que, diante de tantas possibilidades de entretenimento existentes, o usuário do aplicativo, quando pensa em viajar, precisa fazer o planejamento da viagem e do que pretende despende de recursos. Isso é natural, mas como auxiliar comerciantes e usuários finais? Com o aplicativo proposto será possível que o usuário acesse as principais promoções que ocorrerão na semana e você precisa desenvolver um algoritmo que permita a venda de cupons *on-line*, imprima o nome do cliente no cupom que foi adquirido e ainda exiba o valor total, utilizando as estruturas de repetição com teste no final.

Outras estruturas que já aprendemos, como de seleção ou decisão, também poderão ser utilizadas. Para que você consiga realizar essa tarefa, há várias maneiras. Aqui, fica a recomendação de implementação com uso da estrutura de repetição com teste no final.

O programa deve permitir que o usuário realize essa operação até que deseje finalizar o pedido. O importante é que você compreenda a lógica de programação e desenvolva ainda outras soluções que visem sempre à otimização e eficiência do código.

Então, boas práticas e siga em frente!

Não pode faltar

A partir do momento em que é preciso executar a mesma operação por diversas vezes em um programa, imagine se você tivesse de fazer o mesmo bloco de comandos para realizar essa operação por quantas vezes fosse necessário. Totalmente inviável desenvolver uma estrutura várias vezes e deixar o código-fonte, além de extenso, ineficaz.

Nesse contexto, uma forma de resolver problemas como esse são as estruturas de repetição, que vêm facilitar essas implementações. Sendo assim, além de já ter aprendido sobre estruturas com *looping* ou repetição inseridos no início, veremos, agora, o comando **repita_até_que**, a fim de explicar as estruturas em que ocorre o teste no final do bloco de comandos.



Pesquise mais

Um uso bastante importante para as estruturas de repetição com teste no final que precisamos ressaltar é o caso em que os comandos repetidos devem, obrigatoriamente, ser executados pelo menos uma vez.

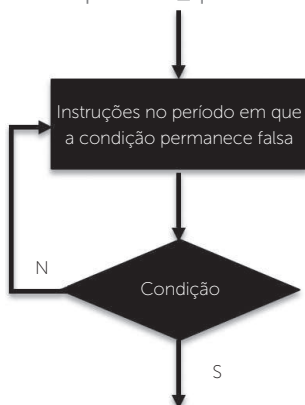
Esse *loop* de teste no final pode ser aplicado às mais diversas situações, como, por exemplo, a verificação de quantidade de extratos que podem ser tirados no mês de uma conta corrente, ou ainda, controle da quantidade de vezes em que é possível repetir uma senha caso o usuário não se recorde e, até mesmo, quantos exames sem pagar a mais por eles um usuário de convênio médico pode fazer no mês.

Essas são apenas algumas situações em que é possível aplicar estruturas de repetição com teste no final.



Assimile

Figura 12 – Diagrama de blocos estrutura repita...até_que.



Fonte: Machado e Maia (2013, p. 97).

O diagrama de bloco ao lado permite identificar como acontece o processamento das instruções do programa.

Essa é conhecida como de estrutura de repetição com teste no final. Os comandos apenas serão executados se a condição for verdadeira. Se a condição é atendida, o processamento passará para as instruções que vêm imediatamente após a verificação da condição, ou teste lógico, e segue com a execução do programa.

Além disso, você também pode implementá-la nas mais diversas demandas em que há cálculos e operações matemáticas, estatísticas e que necessitam de repetição para que se obtenha um resultado mais viável tanto sob o ponto de vista do negócio quanto do ponto de vista do desenvolvimento de *softwares*.



Pesquise mais

Aproveite os estudos e leia a apostila disponível em: <<http://olimpiada.ic.unicamp.br/extras/cursoC/Cap06-RepeticaoControle-texto.pdf>>. Acesso em: 13 jul. 2015.

Segundo as definições dos autores pesquisados, todos convergem para a definição apresentada em nossa referência básica de Machado e Maia. Confira abaixo:



Reflita

"A estrutura **repita...até_que** tem o seu funcionamento controlado também por decisão, porém executa um conjunto de instruções pelo menos uma vez antes de verificar a validade da condição estabelecida. Diferente da enquanto, que executa somente um conjunto de instruções, enquanto a condição é verdadeira" (MACHADO; MAIA, 2013, p. 97).

Com isso, ainda é possível chamar a sua atenção para questões de aplicação de tais estruturas. Lembre-se que, além do projeto de acordo com a Engenharia de *Software* do produto contratado, as empresas da atualidade precisam não apenas do *software* funcionando, mas do grau de confiabilidade e integridade das informações que esse irá disponibilizar.

Então, você pode se deparar com situações em que será preciso implementar estruturas simples e complexas. Nas estruturas de repetição com teste no final, um exemplo que pode ser citado que é considerado de baixa complexidade é a inserção de um contador para que determinada operação seja interrompida quando ele chegar ao limite estabelecido.

Sendo assim, veremos um exemplo para melhorar a sua compreensão sobre essa estrutura. Logo abaixo, é contextualizada em um diagrama de blocos a forma como acontece o processamento da informação e também a ênfase no modo de execução que se apresenta.



Exemplificando

Nesse tipo de estrutura, você pode notar que será preciso utilizar uma variável conhecida como contador.

Suponha que você precise desenvolver um algoritmo que considere o valor de uma variável, que podemos chamar de N, e fazer uma operação de multiplicação. Além de exibir o resultado e repetir essa operação por quantas vezes precisar, vamos estabelecer um valor padrão, como mencionado pelos autores, de cinco repetições.

Veja no exemplo de Machado e Maia (2013) como é que se faz para implementar essa estrutura. Está evidenciado o seu português estruturado:

Português Estruturado:

programa Estrutura_repita_até_que

var

N, R: inteiro

Resp: caractere

inicio

Resp ← "Sim"

repita

leia (N)

R ← N *5

escreva("Resultado:", R))

escreva("Deseja continuar?")

leia (Resp)

ate_que (Resp <> "Sim")

fim

Observe que a estrutura de repetição acima apresenta dentro dela

os comandos que precisarão ser executados até que a condição imposta seja atendida. Nesse caso, se a resposta for “sim”, o programa continuará a execução dos demais comandos.

Se a resposta for “não”, ele volta a solicitar a inserção de um valor para que aconteça a operação de multiplicação que vem na sequência.

Esses comandos serão executados até que a resposta seja “sim”, ou seja, que a condição imposta seja verdadeira.

Na condição, quando se declara um contador, a operação que pertence à estrutura de repetição com teste no final seguirá o mesmo raciocínio lógico.



Faça você mesmo

Agora, é a sua vez de testar os seus conhecimentos implementando esse algoritmo em linguagem de programação C. Siga em frente!

O exemplo de diagrama de blocos ou fluxograma apresentado traz a indicação da repetição com teste no final, com o direcionamento do fluxo para retornar ao início do bloco de comandos indicado, para que repita as instruções quantas vezes forem necessárias até que a condição imposta na decisão seja atendida. Além disso, para essa operação, você notou que há a presença de um contador. Mas você sabe para que serve, como e quando utilizá-lo?

Normalmente, esse é implementado com o nome de variável “i”, que será responsável pelo comando de contabilização de um determinado processo e, portanto, também pode ser denominada de “contador”. Mas o conceito pode ser aplicado sempre que necessário, independentemente do nome da variável. Então, veja a seguir uma estrutura de repetição com teste no final e contador:

$i \leftarrow 1$

repita

escreva (“I =”, i)

$i \leftarrow i + 1$

ate $i > 3$

Na estrutura apresentada, você já deve ter em mente que a variável “i”, em algoritmos, foi declarada corretamente na seção de declaração de variáveis. Para evitar erro, essa deverá ser inicializada antes da execução do bloco de comandos. Então, podemos interpretar que o algoritmo acima repetirá a instrução de incremento na variável “i” até que o valor dessa seja maior do que três (3). Quando “i” atender à condição, o processo será continuado com as demais instruções, quando houver. Sendo assim, veja mais um exemplo de aplicação da estrutura de repetição com teste no final com contador:

Português estruturado

programa loop_2

var

N, R, i: inteiro

inicio

$i \leftarrow 1$

repita

leia N

$R \leftarrow N * 3$

escreva R

$i \leftarrow i + 1$

até_que ($i > 5$)

fim

Fonte: Machado e Maia, 2013, p. 98.

Novamente, há a inserção do contador para o controle da quantidade de vezes em que as instruções serão repetidas. Há a inicialização da variável para iniciar a contagem. Além disso, temos a possibilidade de agregar à estrutura outras instruções, como exibição dos valores de “R”, manipulação das demais variáveis, e ainda, outros comandos e estruturas, de acordo com o tipo situação que o algoritmo precisa atender. Não se esqueça de que o caráter da variável de contabilização permite que o seu tipo de dado seja apenas inteiro em VisuAlg ou int em linguagem C.

Nesse sentido, sempre que for preciso executar um bloco de instruções por mais de duas vezes, e que exista uma condição a ser atendida, é recomendável o uso

da estrutura de repetição com teste no final (`repita_até`), ou ainda, no início (`faça_enquanto`) de forma a evitar repetições de instruções desnecessárias no código-fonte. Isso pode acarretar um tempo de processamento maior que o necessário, no caso de o código ser desenvolvido de acordo com as boas práticas de programação.



Atenção!

Na Unidade 2, na qual você estudou as estruturas de decisão (`se-então-senão`), há um exercício em que você pode iniciar as melhorias de código. Teste implementar o algoritmo com essas estruturas e aguarde. Na próxima unidade, você aprenderá outras estruturas que poderão resolver a situação proposta com um código mais enxuto e eficiente.

Outro ponto importante a mencionar é que você se sairá melhor em algoritmos se praticar. Quanto mais praticar, melhor será o desenvolvimento de seu raciocínio lógico e, conseqüentemente, do algoritmo que você elaborar.



Pesquise mais

Abaixo, há uma recomendação de leitura e exercícios para que você pratique e se torne um *expert* em estruturas de programação. Siga em frente! Disponível em: <http://www.ufpa.br/sampaio/curso_de_icc/icc/aula%2011/repita_ate.htm>. Acesso em: 17 jul. 2015.

Você deve ter concluído que a expressão “`repita`” indica o início do bloco de comandos que deverá ser executado repetidas vezes, ou seja, é o início do laço. Já a expressão “`até (expressão lógica)`” indica o fim da sequência de comandos que foi especificada para o bloco. Então, a cada vez que é executada a expressão lógica, esta é avaliada. Se a condição for falsa, os comandos do bloco serão executados; se for verdadeira, o programa dará sequência à execução das instruções subseqüentes.



Reflita

Em estruturas de repetição com teste no final, a repetição será executada se a condição imposta for falsa, o que, em lógica, implica em inverter o raciocínio e a repetição acontecer até que a condição imposta seja verdadeira (SOUZA, 2013).

Sem medo de errar

Agora, a fim de abordar o modo como pode ser implementada a estrutura de repetição com teste no final, você precisa desenvolver o algoritmo em português estruturado para a venda de cupons.

```
algoritmo "Venda de cupons"

// Função : Estrutura de repetição com teste no final

// Autor : JJJ

// Data : 13/07/2015

// Seção de Declarações

var

qtdCupons, evento: inteiro

a_pagar_JantarEsp, a_pagar_JantarIta, valorTotal, valorFinal: real

nomeUsuario, resp: caractere

inicio

// Seção de Comandos

escreval("Informe o seu nome para impressão no cupom.")

leia(nomeUsuario)

repita

    escreval("Selecione o local:")

    escreval ("1- Jantar Espanhol")

    escreval ("2- Noite Italiana")

    leia (evento)

    escreval("Informe a quantidade de cupons:")
```

```

leia(qtdCupons)

escolha evento

    caso 1

        escreval("Valor Cupom: R$110,00 por pessoa.")

        a_pagar_JantarEsp <- 110 * qtdCupons

        escreval("Valor a pagar por", qtdcupons," para o jantar Espanhol: R$",a_
pagar_JantarEsp)

    caso 2

        escreval ("Valor Cupom: R$120,00 por pessoa.")

        a_pagar_JantarIta <- 120 * qtdCupons

        escreval("Valor a pagar por", qtdcupons," para o jantar Espanhol: R$",a_
pagar_JantarIta)

    outrocaso

        escreval ("Informe opção 1 ou 2.")

fimescolha

valorTotal <- a_pagar_JantarEsp + a_pagar_JantarIta

escreval ("Deseja adquirir mais cupons? S para sim ou N para nao.")

leia(resp)

ate resp = "n"

escreval(nomeUsuario," , o valor total a pagar é: R$: ", valorTotal,".00.")

escreval("Obrigado pela preferência e divirta-se!")

finalgoritmo

```



Atenção!

O cálculo apresentado acima não considera a possibilidade de o cliente realizar mais do que duas compras. Para que isso seja possível, é necessário implementar uma estrutura que permita a contabilização de todos os cupons vendidos.



Lembre-se

Os comandos de repetição são chamados de iterativos, nome que vem da álgebra, do método das aproximações sucessivas ou método de iteração linear. Em sua definição básica, a variável de controle, em cada repetição, sofre uma variação em seu valor anterior e o resultado obtido deve convergir para o valor-limite. Caso a variação que a variável de controle sofre não convergir para o valor-limite, o processo se repetirá infinitamente. Isso significa um erro grave de programação (PIVA JR., 2012, p. 203).

Avançando na prática

Pratique mais

Instrução

Desafiamos você a praticar o que aprendeu, transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois as compare com a de seus colegas.

Repetição condicional com teste no final

| | |
|---------------------------------------|---|
| 1. Competência de fundamentos de área | Conhecer os princípios e conceitos que envolvem o aprendizado em construção de algoritmos e programação e a sua importância para o universo do desenvolvimento de sistemas. |
|---------------------------------------|---|

| | |
|------------------------------|--|
| 2. Objetivos de aprendizagem | <ul style="list-style-type: none"> • Saber identificar a necessidade ou a possibilidade de aplicar estruturas de seleção CASE; • reconhecer, compreender e saber implementar as estruturas de repetição condicionais com teste no início; • saber fazer a implementação de estruturas de repetição condicionais com teste no final; • saber como aplicar as estruturas de repetição controladas por variáveis. |
| 3. Conteúdos relacionados | Repetição condicional com teste no final. |
| 4. Descrição da SP | Dado o algoritmo apresentado para a venda de cupons, desenvolva o seu diagrama de blocos. |
| 5. Resolução da SP | <p>Figura 14 – Fluxograma estrutura de repetição repita_até</p> <pre> graph TD Inicio([Início]) --> nome[/nome/] nome --> Local_qtd{{Local, qtd}} Local_qtd --> Qtd_Cupons[/Qtd Cupons/] Qtd_Cupons --> Valor_Total_1[Valor Total] Valor_Total_1 --> Evento{Evento} Evento -- False --> Evento Evento -- True --> Local_1[Local 1] Evento -- True --> Local_2[Local 2] Evento -- True --> Local_3[Local 3] Local_1 --> Merge(()) Local_2 --> Merge Local_3 --> Merge Merge --> Valor_Total_2[Valor Total] Valor_Total_2 --> S_N[/S, N/] S_N --> Resp_n{Resp = "n"} Resp_n -- False --> Evento Resp_n -- True --> Valor_Total_3([Valor Total]) Valor_Total_3 --> Valor_Total_4([Valor Total]) Valor_Total_4 --> Fim([Fim]) </pre> <p>Fonte: Machado e Maia (2013, p. 99).</p> |



Lembre-se

"[...] o bloco do comando começa com a palavra-chave **repita** e termina com a palavra-chave **até** (sem acento). Expressão- lógica: enquanto ela for falsa, a sequência de comandos é repetida. Quando ela tornar verdadeira o controle passa para o próximo comando que se segue" (PIVA Jr., 2012, p. 212).



Faça você mesmo

Agora que você já conhece a estrutura de repetição com teste no final e já fez o pseudocódigo e o seu diagrama de blocos, você pode desenvolver a solução em linguagem de programação C. Bons estudos!

Faça valer a pena!

1. Descreva a estrutura de repetição com teste no final.

2. Dadas as afirmações, assinale a alternativa correta.

I – Nas estruturas de repetição com teste no final, não há tomada de decisão.

II – As estruturas de repetição com teste no final são controladas por decisão.

III – Não é necessário uso de variável de incremento.

- a) V – V – V.
- b) F – F – F.
- c) F – V – F.
- d) F – V – V.
- e) V – V – F.

3. Complete as lacunas da frase com as palavras da alternativa correta.

"Você deve ter concluído que a expressão '_____' indica o

_____ do bloco de comandos que deverá ser executado repetidas vezes, ou seja, é o início do laço. Já a expressão 'até (_____)' indica o fim da sequência de comandos que foi especificada para o bloco."

- a) for/true/false.
- b) repita/início/expressão lógica.
- c) case/for/while.
- d) enquanto/faça/repita
- e) enquanto/início/até

4. Na frase abaixo, há algumas descrições. Associe os respectivos conceitos e assinale a alternativa correta.

"A estrutura _____ tem o seu funcionamento controlado também por decisão, porém executa um conjunto de instruções pelo menos uma vez antes de verificar a validade da condição estabelecida. Diferente da _____, que executa somente um conjunto de instruções, enquanto a condição é verdadeira" (MACHADO; MAIA, 2013, p. 97).

- a) enquanto/fimenquanto.
- b) do/while.
- c) *enquanto/faça.*
- d) *repita...até_que/enquanto.*
- e) enquanto/repita_até.

5. Explique quais são as diferenças entre as estruturas de repetição com teste no final e com teste no início.

6. Dadas as afirmações, assinale a alternativa que contém a sequência correta quanto à sua veracidade.

I – As estruturas de repetição com teste no final não permitem que outras estruturas como as de seleção ou decisão sejam usadas no mesmo bloco de comandos.

II – As estruturas de repetição com teste no final permitem que outras estruturas, como as de seleção ou decisão, sejam usadas no mesmo bloco de comandos.

III – É possível usar incremento ou simplesmente analisar a condição determinada na estrutura de repetição com teste no final.

- a) F – V – V.
- b) F – F – V.
- c) F – F – F.
- d) V – V – V.
- e) V – F – F.

7. Dado o trecho de código abaixo, identifique a qual estrutura de repetição se refere.

```
i ← 1  
repita  
  escreva ("I =", i)  
  i ← i + 1  
ate i > 3
```

- a) Com teste no início.
- b) Com teste no final.
- c) Com decisão no início.
- d) Com decisão no final.
- e) Com seleção de teste.

Seção 3.4

Repetição controlada por variável

Diálogo aberto

Vamos iniciar os estudos que envolvem o desenvolvimento de *software* com estruturas de repetição com variável de controle. Diferentemente das já estudadas, as quais nos ensinaram como inserir a repetição de um bloco de comandos a partir da realização de testes, seja no final do bloco com o comando “enquanto” ou no início com o comando “repita”, essa alternativa de repetição por variável de controle vem nos mostrar uma forma de facilitar a codificação em estruturas que requerem um número finito de repetições. Estamos falando da estrutura “para” ou “for”, como é conhecida nas linguagens de programação.

Ao encontro de nosso objetivo de entender e saber aplicar tais estruturas, vamos focar no desenvolvimento de uma solução que seja contemplada no protótipo de aplicativo para divulgação e vendas de cupons de gastronomia e hotelaria dos comerciantes do Litoral Sul cuja evolução estamos acompanhando.

Nesse contexto, você deverá implementar a função de controle por variável “para”, com a inserção de uma funcionalidade que acrescente ao algoritmo anterior de compra de bilhetes uma nova funcionalidade que permite ao administrador do sistema realizar uma pesquisa do público-alvo que tem realizado as compras.

A princípio, o algoritmo deverá retornar apenas a quantidade de aquisições realizadas por homens ou por mulheres, pois o próximo passo será desenvolver uma política de fidelização de clientes, que também deverá ser implementada no aplicativo. Sendo assim, uma pesquisa que mostra de cada dez aquisições de cupons de um determinado estabelecimento quantos deles são para homens e quantos são para mulheres é o desafio deste exercício.

Recapitulando, vimos até aqui as funcionalidades básicas de algoritmos, definimos algumas verificações utilizando as estruturas de decisão e de seleção, e agora estamos incluindo outras funções com as estruturas de repetição. Mas, para melhorar as rotinas propostas, teremos de estudar, além dessas, os vetores, as matrizes e entender de que forma essas contribuirão com o desenvolvimento do protótipo.

Então, fica a recomendação de implementação da repetição por variável de controle e prepare-se para os outros desafios que envolvem a lógica computacional e suas habilidades em desenvolvimento para atender às regras de negócios.

Por ora, bons estudos e práticas!

Não pode faltar

O estudo de algoritmos visa apresentar as estruturas que podemos utilizar para o desenvolvimento de *softwares* e ainda evidenciar como é a lógica computacional, ou seja, a forma que devemos apresentar as informações para que o computador possa interpretá-las.

Compreender as estruturas de programação existentes, bem como saber aplicá-las, é fundamental para quem pretende se dedicar a um mercado cada vez mais em expansão e difundido entre vários tipos de aplicações, desde tradicionais, para *desktops*, até programação para facilitar o dia a dia das pessoas com eletrodomésticos inteligentes, que estão customizados para auxiliar em tarefas como verificar quais alimentos estão faltando nos compartimentos de uma geladeira, por exemplo, ou ainda lembrar de compromissos da semana. Enfim, há inúmeras formas de se implementar as estruturas de programação. Sem contar as aplicações que são desenvolvidas para as plataformas *mobile*.

Isso tudo faz com que esses estudos sejam intensificados e que você, ao estudar, consiga fazer o exercício de aplicação de tais mecanismos de controle através de programação que venha atender às situações e regras de negócios de que as empresas necessitam, assim como trabalhar em novidades que agreguem à sociedade uma utilidade às suas principais áreas de preocupação, tais como: saúde, educação, segurança e bem-estar. De forma que consiga transcender as práticas de sala de aula, para a identificação de possibilidades de aplicação de tais conceitos no cotidiano das pessoas e das empresas, aproximando cada vez mais a teoria, que é de extrema importância, da sua aplicação, a prática.

Devido a tal motivação, vamos continuar os estudos nas estruturas computacionais que envolvem o desenvolvimento de algoritmos e a lógica de programação. Com isso, o foco desta seção repousa sobre a apresentação da estrutura de repetição controlada por variável, conhecida como **"para (for)"**. Esta é recomendada a fim de facilitar a programação de estruturas em que se conhece a quantidade de repetições que o programa deverá realizar. Então, as outras estruturas com teste no início (enquanto) e com teste no final (repita) são sugestões para o caso de não ter uma quantidade definida de repetições que serão realizadas.



Assimile

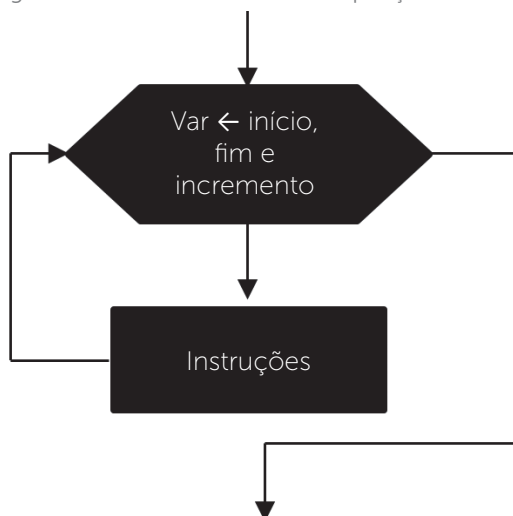
Veja como ficará o português estruturado, seguindo as recomendações de estudos de Manzano e Oliveira (2010):

para <nome da variável> **de** <parâmetro de início> **até** <parâmetro de fim> **passo** <valor do incremento da operação> **faça**

<bloco de comandos que deverão ser executados na repetição por variável de controle> **fim_para**

Agora, veja, também, como é feita a representação dessa estrutura no diagrama de blocos:

Figura 15 – Diagrama de blocos estrutura de repetição controlada por variável



Fonte: Manzano e Oliveira (2010, p. 102).

Observe que nessa estrutura as condições para que as instruções sejam repetidas se encontram logo no início do bloco. Isso permite que se estabeleça um critério finito, ou seja, que a repetição aconteça apenas por um determinado número de vezes e dá-se sequência à execução dos comandos.

O símbolo utilizado para representar o comando chama-se “processamento predefinido ou preparação” (MANZANO; OLIVEIRA, 2010, p. 101).

Agora que você já tem essas informações, conheça os comandos que compõem a estrutura **“para (for)”**. A sintaxe de cada ambiente de programação pode variar, no entanto a lógica dessa estrutura implementa basicamente uma verificação com a utilização de uma variável, que tem por função estabelecer um critério como condição de repetição. Por exemplo, em VisuAlg (PIVA JR., 2012, p. 207):

```
para i de 1 ate 4 faca
    escreval ("I = ", i)
fimpara
```

Como apresentado no exemplo, todos os comandos elencados (**para...de...até...faça...fimpara**) foram utilizados, e cada um com uma função complementar ao comando “para”, de forma que seja estabelecida a quantidade de repetições que o bloco de comandos deverá executar, através da inserção da variável de controle, nesse caso atuando como contador.

A lógica computacional para o processamento desses comandos acontece da seguinte forma (PIVA JR., 2012):

1. A variável de controle, no caso, chamamos de “i” no exemplo, sendo iniciada com o valor mínimo pretendido antes da linha de comando “para” ($i = 1$).
2. Na linha de comando “para”, há a comparação do valor da variável de controle com o limite estabelecido (até).
3. Se o valor comparado for menor ou igual ao valor do limite, então as instruções que estão dentro do bloco de comandos do “para” serão executadas repetidamente até que se atinja o valor limite.
4. A cada execução das instruções dentro do comando “para”, há o incremento da variável de controle, e encerram-se esses procedimentos quando o valor ultrapassar o limite predefinido.

A atribuição de valores à variável de controle acontece apenas uma vez. Isso significa que à variável de controle deve ser atribuído valor antes de iniciar o laço. Apenas quando não se é determinado o incremento com o comando passo, o compilador interpreta que é o valor padrão 1 (um), mais especificamente a função de incremento. Quando se trata ainda de incremento, pode ser realizado também o decremento na contagem. Não se esqueça!



Reflita

"As instruções **para...de...até...passo...faça...fim_para** têm o funcionamento controlado por uma variável denominada contador. Sendo assim, pode executar um conjunto de instruções determinado número de vezes" (MANZANO; OLIVEIRA, 2010, p. 101).

Veja no exemplo abaixo como é possível implementar os comandos.



Exemplificando

O comando **para** será utilizado em um algoritmo que recebe a idade de 100 pessoas e mostra uma mensagem que informa se a pessoa é maior ou menor de idade, sendo a maioridade a partir de 18 anos. Vamos ao algoritmo:

```
// Autor : JJJ
// Data : 27/07/2015
// Seção de Declarações
var
x, idade: inteiro
inicio
// Seção de Comandos
para x de 1 ate 100 faca
    escreva("Qual é a sua idade? ")
    leia(idade)
    se idade >= 18 entao
        escreva("Você é maior de idade!")
    fimse
fimpara
fimalgoritmo
```

**Faça você mesmo**

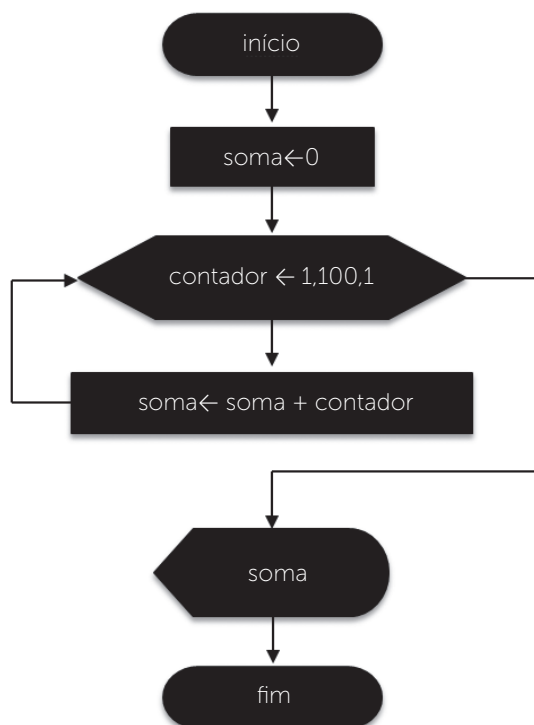
Tente implementar o algoritmo sugerido em linguagem de programação C. Utilize, de preferência, o Dev C++.

**Pesquise mais**

Pesquise mais sobre as estruturas de repetição e implemente os algoritmos na controlada por variável, conforme as recomendações de seu material. Disponível em: <<https://www.youtube.com/watch?v=WJQz20i7Cyl>>. Acesso em: 27 jul. 2015.

Vamos fazer a representação em diagrama de blocos de um algoritmo que, a partir de 100 números inteiros dados, efetue a sua soma. Veja como fica:

Figura 16 – Diagrama de blocos algoritmo soma de números inteiros



Fonte: Manzano e Oliveira (2010, p. 215).

Veja também como fica o português estruturado desse algoritmo (MANZANO; OLIVEIRA, 2010, p. 102):

```

programa soma_inteiros
var
    soma, contador: inteiro
inicio
    soma  $\leftarrow$  0
    para contador de 1 ate 100 passo 1
faca
    soma  $\leftarrow$  soma + contador
fim_para
escreva soma
fim

```

Observe que inicialmente há as instruções de declaração do programa e as respectivas variáveis utilizadas. Em seguida, inicia-se o algoritmo, a seção de comandos. Então, a variável soma é inicializada com o valor zero (0). A inserção do comando “para” utiliza a variável contador seguido do comando “de...até” para indicar a quantidade de vezes que o bloco de comando será repetido. O comando “passo” serve como incremento, então, até que a quantidade de repetições seja menor ou igual ao limite de 100 preestabelecido, esse será incrementado de mais 1, indicando a quantidade de vezes que os comandos foram executados. Quando temos apenas a estrutura **para...de...até...faça...fimpara**, conforme sugerido nos primeiros exemplos,

```

para i de 1 ate 4 faca
    escreval ("I = ", i)
fimpara ,

```

vemos que, para uma quantidade de repetições relativamente pequena, não há a necessidade de inserção do comando **passo** conforme sugerido no último exemplo, pois não é um elemento obrigatório. Exceto se há a necessidade de decrementar conforme há a repetição. O que ele faz é basicamente substituir a necessidade de declaração da instrução que realiza o incremento “i ++” ou “i = i + 1”, sendo essas expressões substituídas pelo comando “passo”, conforme indicação do exemplo. Recapitule a lógica inserida com outro exemplo citado por Manzano e Oliveira (2010, p. 102):

Português estruturado:

programa Looping_2

var

N, R, I: inteiro

inicio

para I de 1 ate 5 passo 1 faça

leia N

$R \leftarrow N * 3$

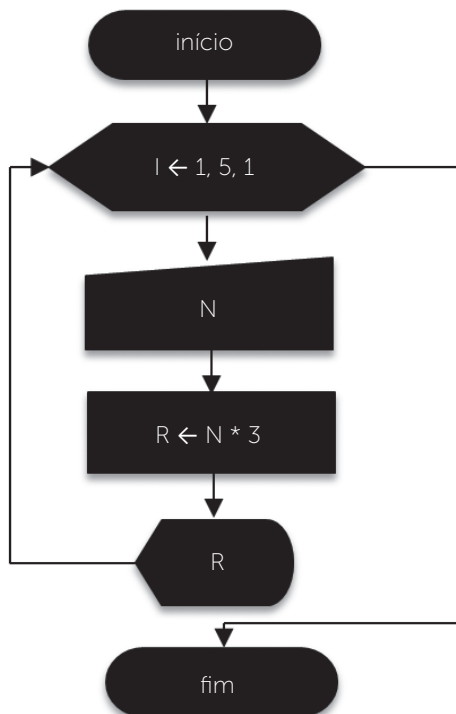
escreva R

fim_para

fim

Observe no exemplo a função de incremento à variável de controle que o comando **passo** exerce.

Figura 17 – Diagrama de bloco com comando passo



Fonte: Adaptado de Manzano e Oliveira (2010).



Refleta

Será executado o conjunto de instruções entre os comandos **para** e **fim_para**, sendo a variável 'i' (variável de controle) inicializada com o valor 1 e incrementada de mais 1 pelo comando **passo** até o valor 5. Esse tipo de repetição pode ser utilizado todas as vezes que se tiver a necessidade de repetir trechos finitos, quando se conhece o valor inicial e o valor final (MANZANO; OLIVEIRA, 2010. p. 103).

Com isso, finalizamos os estudos em estruturas de repetição e fica a recomendação de resolução dos exercícios e práticas no VisuAlg utilizando os exemplos e exercícios propostos. Bons estudos!

Sem medo de errar

Como a entrega dessa fase prevê a inserção de uma funcionalidade que conta utilizando a estrutura de repetição por variável de controle “para” ou “for”, você precisa elaborar o algoritmo para esta nova ação do sistema. Siga em frente e veja uma sugestão proposta para esse algoritmo:

```
programa Genero_cupom

var

nome, sexo, h, m: caractere

inicio

// Seção de Comandos

para x de 1 ate 10 faca

    escreva("Qual o seu nome? ")

    leia(nome)

    escreva("Informe as seguintes siglas para indicar H - Homem ou M - Mulher: ")

    leia(sexo)

    escolha sexo

        caso "H"

            h <- h + 1

        caso "M"

            m <- m + 1

    outrocaso

        escreval("Informe apenas H para o caso de gênero masculino ou M para o caso de gênero feminino!")

    fimescolha

fimpara
```

limpatela

escreval(h," Homens adquiriram cupons de seu estabelecimento.")

escreval(m," Mulheres adquiriram cupons de seu estabelecimento ")

finalgoritmo



Atenção!

Um algoritmo pode ser implementado de diversas formas, desde que a sintaxe esteja correta e a semântica, ou seja, a lógica esteja coerente com a regra de negócio que se deseja atender. Verifique as possibilidades de melhorias e pratique mais.



Lembre-se

Na representação da estrutura PARA-ATÉ-FAÇA (**DESDE-PARA-FAÇA**) está subentendido que o contador é automaticamente incrementado e controlado (a partir do valor inicial, alcança-se o valor final-inclusive em incrementos unitários). É dessa forma que o comando PARA-ATÉ-FAÇA (**DESDE-PARA-FAÇA**) é implementado na maioria das linguagens de programação (SOUZA et al., 2011, p. 144).

Avançando na prática

| Pratique mais | |
|---|--|
| <p>Instrução</p> <p>Desafiamos você a praticar o que aprendeu, transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois as compare com a de seus colegas.</p> | |
| <p>Repetição controlada por variável</p> | |
| <p>1. Competência de fundamentos de área</p> | <p>Conhecer os princípios e conceitos que envolvem o aprendizado em construção de algoritmos e programação e a sua importância para o universo do desenvolvimento de sistemas.</p> |

| | |
|------------------------------|---|
| 2. Objetivos de aprendizagem | <ul style="list-style-type: none"> • Saber identificar a necessidade ou a possibilidade de aplicar estruturas de seleção CASE; • reconhecer, compreender e saber implementar as estruturas de repetição condicionais com teste no início; • saber fazer a implementação de estruturas de repetição condicionais com teste no final; • saber como aplicar as estruturas de repetição controladas por variáveis. |
| 3. Conteúdos relacionados | Repetição controlada por variável. |
| 4. Descrição da SP | Um cliente solicita a inserção de uma funcionalidade de controle de preço de venda e preço de custo de produtos em sua loja de suplementos alimentares. No entanto, os produtos são sempre comprados em lotes de cinquenta unidades e o sistema precisa verificar o lote e mudar o preço de acordo com essa quantidade no sistema. Um parâmetro pode ser a data de compra, para a validação dos lotes. O sistema deve retornar o preço de custo e de venda, se houve ganho, perda ou se não houve lucros para aquele produto. Pratique! |
| 5. Resolução da SP | <pre>// Seção de Declarações var x: inteiro precoVenda, precoCusto: real mediaCusto, mediaVenda: real inicio // Seção de Comandos para x de 1 ate 50 faca limpatela escreva("Informe o preço de custo do produto: ") leia(precoCusto)</pre> |

```

mediaCusto <- mediaCusto + precoCusto
escreva("Informe o preço de venda: ")
leia(precoVenda)

mediaVenda <- mediaVenda + precoVenda
se precoVenda < precoCusto entao
    escreval("Para este produto houve perdas
em lucratividade.")
senao
    se precoVenda = precoCusto entao
        escreval("Para este produto não houve
perdas e nem lucros.")
    fimse
    se precoVenda > precoCusto entao
        escreval("Para este produto houve
ganho e boa lucratividade.")
    fimse
fimse
fimpara
escreval("A média de preço de custo foi:
",mediaCusto/40)
escreval("A média de preço de venda foi:
",mediaVenda/40)
finalgoritmo

```



Faça você mesmo

Implemente o algoritmo em linguagem de programação C. Boas práticas!



Lembre-se

para, de, ate, faca e fimpara são palavras-chave do comando; variável: é a variável de controle do comando e deve ser do tipo inteiro; valor inicial: é o valor de início da variável de controle. Pode ser uma constante ou uma expressão aritmética, desde que o valor seja do tipo inteiro. A atribuição desse valor à variável de controle é feita apenas uma vez antes de iniciar a primeira repetição do comando (PIVA JR., 2012, p. 207).

Faça valer a pena

1. Complete as lacunas da frase com as palavras da alternativa que apresenta a sequência correta de acordo com as definições apresentadas.

"A inserção do comando _____ utiliza a variável contador seguido dos comandos _____ para indicar a quantidade de vezes que o bloco de comando será repetido. O comando _____ serve como incremento, então, até que a quantidade de repetições seja menor ou igual ao limite de 100 preestabelecido, esse será incrementado de mais 1, indicando a quantidade de vezes que os comandos foram executados."

- a) para/de...até/passo.
- b) ate..de/para/passo.
- c) passo/para/de...até.
- d) de..até/passo/para.
- e) para/passo/de..até.

2. Analise as afirmações e assinale a alternativa referente à sequência correta.

I – As estruturas com teste no início (enquanto) e com teste no final (repita) são sugestões para o caso de não existir uma quantidade definida de repetições que serão realizadas.

II – O comando "for" é equivalente ao comando enquanto.

III – O comando "passo" realiza o incremento da variável de controle.

São verdadeiras as afirmações:

- a) I, II e III.
- b) I e II.
- b) I e III.
- d) II e III
- e) Apenas III.

3. Assinale a alternativa que contém o significado correto do símbolo

utilizado para elaborar o diagrama de blocos da estrutura de repetição controlada por variável.

- a) Exibir.
- b) Preempção.
- c) Datagrama.
- d) Preparação.
- e) Processo.

4. Descreva qual é a lógica computacional da estrutura de repetição controlada por variável.

5. Desenvolva um diagrama de blocos e o respectivo português estruturado que utilize o comando “para” com o incremento “passo”.

6. São verdadeiras as afirmações:

I – Esse tipo de repetição pode ser utilizado todas as vezes em que se tiver a necessidade de repetir trechos finitos, quando se conhece o valor inicial e o valor final.

II – Variável: é a variável de controle do comando e deve ser do tipo inteiro.

III – Valor inicial: é o valor de início da variável de controle. Pode ser uma constante ou uma expressão aritmética, desde que o valor seja do tipo inteiro.

- a) I e II.
- b) I, II e III.
- c) II e III.
- d) III e I.
- e) Apenas I.

7. De acordo com a estrutura abaixo, assinale a alternativa que contém a informação coerente ao tipo de comando apresentado.

```
para i de 1 ate 10 faca  
    escreval ("I = ", i)  
fimpara
```

- a) i é uma variável indefinida.
- b) Não é obrigatória a inicialização da variável de controle nem antes de iniciar o comando de repetição, nem depois.
- c) A variável de controle é o "i".
- d) O contador pode ser suprimido da estrutura controlada por variável.
- e) A variável de controle não será incrementada na repetição.

Referências

EVARISTO, Jaime. **Aprendendo a programar**: programando na linguagem C para iniciantes. Rio de Janeiro: Book Express, 2001.

MANZANO, José A. N. G.; OLIVEIRA, Jayr Figueiredo de. **Algoritmos**: lógica para desenvolvimento de programação de computadores. 24. ed. São Paulo: Érica, 2010.

PIVA JR. et al. **Algoritmos e programação de computadores**. Rio de Janeiro: Elsevier, 2012.

SOUZA, Marco. **Algoritmos e lógica de programação**. 2. ed. São Paulo: Cengage Learning, 2013.

ZIVIANI, Nivio. **Projeto de Algoritmos**: com implementações em Java e C++. São Paulo: Cengage Learning, 2011.

VETORES E MATRIZES

Convite ao estudo

Prezado aluno, bem-vindo a mais esta unidade de ensino. Vamos aprender o que são os vetores e matrizes e como se aplicam tais estruturas, de forma a elaborar algoritmos mais eficientes e que permitam a manipulação ordenada e otimizada dos registros. Para que isso aconteça, vamos focar no desenvolvimento da competência fundamental de área que estamos trabalhando, que é a seguinte:

- Conhecer os princípios e conceitos que envolvem o aprendizado em construção de algoritmos e programação e a sua importância para o universo do desenvolvimento de sistemas.

Temos também os seguintes objetivos de aprendizagem para esta unidade de ensino:

- Conhecer quais são as aplicações a partir do uso de vetores e matrizes.
- Conhecer quais são as operações realizadas por vetores e matrizes.
- Saber e conhecer quais são as operações em que se aplicam às estruturas de dados em vetores.
- Saber e conhecer quais são as operações em que se aplicam as estruturas de dados em matrizes.

Estamos finalizando este material de estudos, e com ele encerramos a apresentação das principais estruturas necessárias para o desenvolvimento de algoritmos e a lógica de programação.

Ao encontro dos objetivos propostos, aprendemos também no decorrer dos nossos trabalhos os seguintes elementos: na Unidade 1, trabalhamos as

definições, o histórico e os principais algoritmos estudados que servem como referência até os dias de hoje. Na Unidade 2, trabalhamos os tipos de dados e os operadores que podem ser utilizados para o desenvolvimento de algoritmos e as estruturas de decisão e seleção. Já na Unidade 3, vimos as estruturas de repetição com teste no início e com teste no final, além de reforçar com exercícios a estrutura de seleção. Com isso, na Unidade 4, temos a ênfase na apresentação de conceitos como vetores e matrizes.

Com esta forma de organizar e manipular as informações é possível melhorar algumas estruturas dos algoritmos que estamos trabalhando para a proposta da *Think Now*. Esta deve apresentar um protótipo de aplicativo que ofereça locais e ofertas de gastronomia e hotelaria para os comerciantes do Litoral Sul e o desenvolvimento de algumas funcionalidades. Neste sentido, a partir de agora, vamos propor melhorias nos algoritmos anteriormente apresentados com o uso das estruturas de vetor e matrizes, e ainda, novas funcionalidades que busquem maior controle dos investimentos e recursos para o desenvolvimento do aplicativo, além de funcionalidades de armazenamento e registro dos dados que são trabalhados, como os de cadastro. Além disso, para encerrar esta unidade, vamos propor transpor essas estruturas para a linguagem de programação C utilizando a plataforma de desenvolvimento Dev C++. Desde já, bons estudos e práticas a você!

Seção 4.1

Aplicações utilizando vetores e matrizes

Diálogo aberto

Estamos finalizando os estudos em algoritmos e lógica de programação e precisamos resgatar algumas das ações que já realizamos para o desenvolvimento do protótipo de aplicativo que foi sugerido. Nesse sentido, na Unidade 1, fizemos um levantamento de todas as ações do projeto e as transformamos em diagrama de blocos de forma a apresentar também em linguagem natural o passo a passo necessário para a realização e condução do projeto. Além disso, ainda foi proposto um algoritmo que apresentasse a possibilidade de se obter a média de acessos que cada categoria recebeu, para tal, foi proposto um algoritmo em linguagem natural e foram especificados os respectivos tipos de dados e variáveis utilizados nesse processo. No caso, o algoritmo proposto também incluía a opção de direcionar o usuário para conhecer o estabelecimento comercial gastronômico ou do setor de hotelaria, ou ainda, direcionar a um módulo de agenda e reservas.

Na Unidade 2, trabalhamos, inicialmente, com um algoritmo que implementa estruturas de decisão e foi aplicado o conceito em um algoritmo que visa coletar informações sobre a satisfação do usuário com relação ao uso do aplicativo. Através dessa mesma estrutura, apresentamos os conceitos com testes, como a inserção de uma verificação de cadastro. Outra ação do aplicativo que foi proposta utilizando as estruturas de decisão foi a implantação da verificação de categoria, de acordo com o cadastro realizado pelo cliente, para a emissão do boleto de coparticipação de pagamento. Já a Seção de autoestudo 2.4 trouxe a possibilidade de criar, através do conceito da estrutura de decisão encadeada, uma interface que realiza a inclusão, alteração e exclusão dos dados do cliente.

Na Unidade 3, fizemos a apresentação do algoritmo de realização de reserva a partir do uso do conceito da estrutura de seleção CASO. Com isso, na sequência, fizemos a inclusão de mais uma funcionalidade, agora com a utilização da estrutura de repetição com teste no início "faça, enquanto", que realiza uma pré-reserva. A fim de promover o uso de mais de uma estrutura em um único algoritmo, também foi proposto um que realiza a venda de cupons, desenvolvido com as estruturas de seleção e de repetição com teste no final "repita_até". Nesse sentido, você pode

acompanhar mais uma implementação que faz uso da estrutura de repetição “para” ou “for” como conhecida, em que se classifica a venda do cupom, a partir do gênero do usuário que está realizando a compra, tendo esta funcionalidade o objetivo de permitir a implantação de um módulo de fidelização e de ações de acordo com as preferências desse usuário. Agora, na Unidade 4, o desafio consiste em implementar algumas operações que visam aprimorar os algoritmos apresentados anteriormente e ainda sugerir novas funcionalidades, como o desenvolvimento de um algoritmo que calcule os juros sobre uma aplicação em fundo de renda fixa, em que os comerciantes ingressaram a fim de angariar recursos para desenvolver o aplicativo. Essa é uma situação que aproxima o conteúdo teórico com a sua aplicação prática. Desde já, bons estudos e práticas a você!

Não pode faltar

Com os estudos desta unidade de ensino em algoritmos e lógica de programação, vimos, principalmente, quais são as estruturas que podem compor ações em um sistema computacional. Vimos até aqui a definição de algoritmos, variáveis e constantes, tipos de dados, operações básicas e operadores lógicos, além disso, conhecemos também as estruturas de decisão e seleção, as estruturas de repetição, e agora vamos estudar outras que chamamos de vetores e matrizes.

Mas, afinal, para que servem tais estruturas?

Você percebeu que, em algumas situações, é preciso armazenar dados, ou mesmo criar estruturas em que seja possível organizá-los para gerar históricos, criar cenários, ou acessá-los mais facilmente. Para resolver essas questões, vamos apresentar, a partir de agora, os vetores e as matrizes.

Quando estamos desenvolvendo um algoritmo, ao declarar uma variável, estamos informando ao computador que é preciso separar um espaço em memória, que acaba de receber o nome da variável declarada, e este interpreta essa informação de modo a permitir a alocação de um valor nesse espaço que foi determinado. No entanto, ao informar também o seu tipo de dados, estabelece-se uma regra em que naquele espaço de memória será alocado o respectivo tipo de dado especificado.

Nesse contexto, é possível afirmar, de acordo com Piva Jr. et al. (2012), que em uma variável não é permitido armazenar vários valores, ainda que estes sejam do mesmo tipo. Para resolver essa questão, são utilizados os vetores.



Refleta

“O vetor é uma variável composta, pois é formado por um número finito de variáveis, e homogêneo porque essas variáveis são de um mesmo tipo

de dado. Além disso, o vetor é unidirecional, pois possui somente uma dimensão, representado em linha ou em coluna” (PIVA JR. et al., 2012, p. 229).

Como mencionado, o vetor representa uma **variável composta**. Isso significa que ele é um conjunto de variáveis. O computador, no momento em que são declarados o vetor e a quantidade de posições ou variáveis agrupadas que ele terá, direcionará as informações para os espaços do vetor e a alocação acontecerá todas as vezes que ele for indicado/apontado, estes são termos comuns no ambiente computacional. Os vetores também respeitam a regra do tipo de dado (primitivo: inteiro, real, caractere ou lógico), você se lembra qual é ela? É simples: vetores só podem armazenar dados que sejam do mesmo tipo. Lembre-se disso.

Também conhecido como VCHU (Variável Composta Homogênea Unidimensional), o vetor identifica cada um dos seus elementos, ou espaços que foram alocados, através de um índice, ou seja, um número de referência que permita visualizar em uma determinada posição de memória, pertencente ao vetor em questão, o valor que armazena.



Assimile

Observe a estrutura unidimensional que é apresentada:

O vetor a seguir possui cinco posições, conforme ilustrado na Figura 18:

Figura 18 - Vetor unidimensional

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|

Fonte: Adaptado de Piva Jr. et al. (2012, p. 237)

Agora, a este vetor, neste caso, estamos definindo que cada posição contém um dado de um cliente. Esse dado pode ser a quantidade de cupons que ele adquiriu para um evento. Então, temos a seguinte representação:

Figura 19 - Vetor unidimensional cupons

| | | | | |
|---|---|---|---|---|
| 3 | 5 | 2 | 1 | 7 |
| 1 | 2 | 3 | 4 | 5 |

Quantidade de cupons por cliente

índice de posição do vetor

Fonte: Adaptado de Piva Jr. et al. (2012, p. 237)

A Figura 19 representa um vetor com cinco posições. Cada uma delas armazena dados de um cliente, no caso, a quantidade de cupons que cada um adquiriu.

O apontador, ou índice, conforme indicado nas Figuras 18 e 19, pode ser uma variável simples, uma constante, ou ainda um cálculo que resulte em um número inteiro (PIVA JR. et al., 2012).

Há outras formas de aplicação de vetores. Como no exemplo, estamos alocando em cada um dos elementos do vetor a quantidade de cupons que os clientes adquirem, no entanto, talvez seja preciso exibir outras informações, como o nome do cliente. Como é possível trabalhar com vetores se quantidade e nome são tipos de dados diferentes?

Nesse caso, podemos criar mais um vetor, mas agora que contenha o tipo de dado caractere e armazene o nome dos clientes. O próximo passo é fazer com que o vetor “cupons” se relacione com o vetor “nomes”. Isso é possível em função do índice do vetor que deverá apontar para a mesma posição nos dois vetores. O que acontece é que o índice aponta para os dois vetores de forma que a informação seja relacionada, por exemplo, posição 2 do vetor cupom com posição 2 do vetor nomes resulta em:

Figura 20 - Apontador dos vetores cupons e nomes

| | | | | | | | | | |
|---|---|---|---|---|----------|--------|-------|------|--------|
| 3 | 5 | 2 | 1 | 7 | Katarina | Júnior | Lúcia | Tony | Andrey |
| 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 |




Fonte: Adaptado de Piva Jr. et al. (2012)



Faça você mesmo

O *link* apresenta informações sobre os principais modos e aplicações para o uso de vetores. Confira em: <<http://www.ime.usp.br/~macmulti/exercicios/vetores/>>. Acesso em: 27 jul. 2015.

Um vetor possui os seguintes atributos: identificador, tamanho, tipo de dado e conteúdo. A atribuição de dados a um vetor pode acontecer das seguintes formas, de acordo com Piva Jr. et al. (2012, p. 241):

- Declaração de um vetor para registro das vendas mensais:

"Var

Vet_Vendas_Anual: VETOR [1..12] de REAL

IndVet: Byte"

- Formas de atribuição de dados em vetores:

a. Atribuir valor de venda referente a um mês:

Vet_Vendas_Anual[10] := 200,00

b. Consulta do valor de venda de um determinado mês com inserção pelo usuário:

Leia(Vet_Vendas_Anual[8])

c. Cálculo do acréscimo de 10% sobre o valor de vendas em um dezembro:

Vet_Vendas_Anual[12] := Vet_Vendas_Anual[12] x 1,10

- Acesso com valores constantes como índices em vetores:

a. Atribuição de valor de venda para uma posição do vetor que seja correspondente ao valor da variável IndVet:

Vet_Vendas_Anual[IndVet] := 600

Para esta atribuição, o autor supramencionado indica que, se a posição do vetor que foi apontada pela variável IndVet for igual a 5 (IndVet =5), então o valor "600" será atribuído à quinta posição do vetor. E se quiséssemos incrementar um outro valor na posição "6" do vetor, como seria esse comando? Analise a seguir:

Vet_Vendas_Anual[IndVet +1] :=800

Além dessa forma de manipulação do dado do vetor, através da utilização de uma variável, no caso a IndVet, também é possível inserir acréscimo, ou outras operações, com a seguinte declaração (PIVA JR. et al., 2012, p. 242):

Vet_Vendas_Anual[IndVet] := Vet_Vendas_Anual[IndVet] x 1,05

- Consultas a vetores podem ser realizadas da seguinte forma:

a. Exibir valores de um determinado mês:

Escreva (Vet_Vendas_Anual[6]:8:2)

b. Exibir valores da variável IndVet:

Escreva(Vet_Vendas_Anual[IndVet]:8:2)

As codificações inseridas “:8:2” correspondem a posições decimais, no caso duas (:2), e “:8” é utilizado quando se deseja manter alinhada a parte inteira da operação para variáveis do tipo de dado real.

Observe no exemplo uma forma de aplicar a estrutura de dado vetor:



Exemplificando

O exemplo a seguir é de um algoritmo que lê a nota de 10 alunos e exibe sua média final e nome. Correlaciona a média ao nome do aluno, nos respectivos vetores. Analise o exemplo em VisuAlg (PIVA JR. et al., 2012, p. 248):

Algoritmo “MédiaFinal_Alunos”

//Seção de declaração de variáveis:

Var

Vet_Nome_Aluno: Vetor [1..10] de caractere

Vet_MediaFinal: Vetor [1..10] de real

//declaração do índice do vetor

Contador_Alunos: inteiro

Acum_Media, MediaTurma: real

//Entrada de dados- solicita, lê nome e média final de 10 alunos e determina uma situação

Inicio

para Contador_Alunos de 1 ate 10 faca

limpatela

Escreval (“Informe Nome do Aluno(a):”, Contador_Alunos, “ :”)

Leia (Vet_Nome_Aluno[Contador_Alunos])

Escreval (“Informe sua media final:”)

Leia (Vet_MediaFinal[Contador_Alunos])

Se (Vet_MediaFinal[Contador_Alunos]>=7) entao

Escreval (Vet_Nome_Aluno[Contador_Alunos], “Você está aprovado(a)!”)

Senao

Se (Vet_MediaFinal[Contador_Alunos]>=5) entao

Escreval (Vet_Nome_Aluno[Contador_Alunos], “Você está em exame!”)

Senao

```

        Escreval (Vet_Nome_Aluno[Contador_Alunos], "Você está reprovado!")
    Fimse
Fimse
Acum_Media <= Acum_Media + Vet_MediaFinal[Contador_Alunos]
FimPara
//Calcula e mostra a média final da turma
MediaTurma:= Acum_Media/ Contador_Alunos -1
Limpatela
Escreval ("Média da turma:", MediaTurma:2:1)
Escreval
//Mostra o desempenho do aluno relacionando sua média final, com a
média da turma
Para Contador_Alunos de 1 ate 10 faca
    Se Vet_MediaFinal[Contador_Alunos]> MediaTurma entao
        Escreval      (Vet_Nome_Aluno[Contador_Alunos], "-      Media:",
            Vet_MediaFinal[Contador_Alunos]:2:1, "- Bom Aluno(a).")
    Senao
        Se (Vet_MediaFinal[Contador_Alunos] < MediaTurma) entao
            Escreval ("Vet_Nome_Aluno[Contador_Alunos], "Media:",
                Vet_MediaFinal[Contador_Alunos]:2:1, "- Aluno(a) com baixo
                desempenho")
        Senao
            Escreval (Vet_Nome_Aluno[Contador_Alunos], "Media:",
                Vet_MediaFinal[Contador_Alunos]:2:1, "-Aluno(a) Médio(a).")
    FimSe
FimSe
FimPara
Fimalgoritmo
(PIVA JR. et al., 2012, p. 250-251)

```

O algoritmo apresentado no exemplo, além de trabalhar diretamente com vetores, utiliza índices para que seja possível estabelecer uma correlação entre eles. Podemos realizar as operações de tomadas de decisão, operações matemáticas e lógicas a partir da manipulação dos valores contidos nos elementos do vetor. Isso em função do apontamento realizado pelo índice, que permite a identificação pela posição do vetor.

Então, na seção de declaração de variáveis, temos de considerar os vetores do algoritmo. Quando há mais de um vetor e estes precisarão fornecer dados que se

correlacionam, como no exemplo, a média por aluno, há, também, a necessidade de definição na seção de variáveis, do índice, no caso do exemplo "Contador_Alunos", e análise das estruturas de seleção encadeadas que foram utilizadas, além da lógica computacional para que esse processamento seja realizado e traga a informação desejada de forma ótima, sob o ponto de vista de eficiência.



Pesquise mais

Assista ao vídeo do canal Games Parati, que ensina a desenvolver algoritmos com vetores utilizando o VisuAlg. Disponível em: <<https://www.youtube.com/watch?v=gmtZSoyy0UI>>. Acesso em: 27 jul. 2015.

Assim como os vetores unidimensionais, trabalhamos com uma estrutura de dados que permite correlacionar mais dados e ordenar as buscas com algoritmos mais eficientes. Chamamos esse conjunto de vetores de matriz. Uma matriz pode ser bidimensional ou multidimensional. Estudaremos com mais detalhes na Seção 4.4 desta unidade, porém, é importante que você a conheça desde já.



Reflita

"A matriz é uma variável composta, pois é formada por um número finito de variáveis, e homogênea porque essas variáveis são de um mesmo tipo de dado" (PIVA JR. et al., 2012, p. 276).

SEM MEDO DE ERRAR

A fim de angariar recursos para o desenvolvimento do aplicativo, os comerciantes do Litoral Sul, organizados em cooperativa, ingressaram em uma carteira de aplicações de um fundo de renda fixa, para investir os valores das contribuições. Sua tarefa será desenvolver um algoritmo em que possam, a partir dos valores aplicados, identificar os valores de juros que serão pagos sobre a aplicação.

Algoritmo "Juros_Aplicacao"

Var

VL_Aplicacao : Real

Vet_Tx_Juros, Vet_VL_Juros, Vet_VL_Corrigido : Vetor [1..12] De Real

Ind_Vet : Inteiro

```

Inicio
  LimpaTela

  Escreval("Informe Valor da Aplicacao Inicial R$:")
  Leia(VL_Aplicacao)
  Vet_VL_Corrigido[1] := VL_Aplicacao

  Escreval("Informe Taxa de Juros Inicial %:")
  Leia(Vet_Tx_Juros[1])

  Para Ind_Vet de 1 ate 12 faca
    Vet_Tx_Juros[Ind_vet] <= Vet_Tx_Juros[Ind_Vet] * 1.025
    Vet_VL_Juros[Ind_Vet] <= Vet_VL_Corrigido[Ind_Vet] *
      (Vet_Tx_Juros[Ind_vet] / 100)
    Vet_VL_Corrigido[Ind_Vet] <= Vet_VL_Corrigido[Ind_Vet] +
      Vet_VL_Juros[Ind_Vet]

    Vet_Tx_Juros[Ind_vet+1] <= Vet_Tx_Juros[Ind_Vet]
    Vet_VL_Corrigido[Ind_Vet+1] <= Vet_VL_Corrigido[Ind_Vet]
  FimPara

// Mostra o conteúdo dos vetores (Taxa de Juros Val. Juros Val. Corrigido
  LimpaTela
  Escreval("Valor da Aplicacao Inicial R$ ",VL_Aplicacao:0:2)
  Escreval
  Escreval("% Tx. Juros Val. Juros Val.Aplic.Corrigida")
  Escreval
  Para Ind_Vet de 1 ate 12 faca
    Escreval(Vet_Tx_Juros[Ind_vet]:2:4,"      ",
      Vet_VL_Juros[Ind_Vet]:6:2,"      ",
      Vet_VL_Corrigido[Ind_Vet]:6:2)
  FimPara
  Escreval
FimAlgoritmo

```

Fonte: Piva Jr. et al. (2012)



Faça você mesmo

Implemente esse exercício também em linguagem C. Boas práticas!



Atenção!

Somente números ou variáveis inteiras podem ser utilizados como índices de um vetor. Por exemplo, se i for uma variável inteira contendo um número que está dentro do intervalo de índices de um vetor A , $A[i]$ será uma expressão válida (SOUZA et al., 2011, p. 171).



Lembre-se

Apesar de não ser necessário declarar explicitamente o tamanho do vetor em um fluxograma, ele possui um tamanho máximo definido pelo problema. Utilizar índices que ultrapassem o maior índice do vetor ou que sejam menores que o menor índice de um vetor é uma operação ilegal e constitui um erro no algoritmo (SOUZA et al., 2011, p. 171).

Avançando na prática

Pratique mais

Instrução

Desafiamos você a praticar o que aprendeu transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois as compare com as de seus colegas.

Aplicações utilizando vetores e matrizes

| | |
|--|--|
| 1. Competência de fundamentos de área | Conhecer os princípios e conceitos que envolvem o aprendizado em construção de algoritmos e programação e a sua importância para o universo do desenvolvimento de sistemas. |
| 2. Objetivos de aprendizagem | Conhecer quais são as aplicações a partir do uso de vetores e matrizes. |
| 3. Conteúdos relacionados | Aplicações utilizando vetores e matrizes. |
| 4. Descrição da SP | Dado o exemplo de aplicação em uma escola do algoritmo de cálculo de média e exibição do respectivo nome do aluno e sua média final, elabore agora o algoritmo em linguagem de programação C. Siga as recomendações do autor de referência desta aula, Piva Jr. et al. (2012, p. 259-260). |

5. Resolução da SP

```

#include <stdio.h>
#include <stdlib.h>
int main()
{
    char Vet_Nome_Aluno[10];
    float Vet_MediaFinal[10];
    int Contador_Alunos;
    float Acum_Media=0, MediaTurma;

    for (Contador_Alunos=0; Contador_Alunos < 10; Contador_
Alunos++)
    {
        system("cls");

        printf("Informe Nome do Aluno(a) - %d : ",Contador_
Alunos+1);
        fflush(stdin);gets(Vet_Nome_Aluno[Contador_Alunos]);

        printf("Informe sua Media Final: ");
        scanf("%f", &Vet_MediaFinal[Contador_Alunos]);
// -----
// SOLUÇÃO PROPOSTA - Estrutura de repetição para
validação da Média Final
//          informada.
//          Considerando que a Média Final é variável do
tipo Float e,
//          portanto, aceita números NEGATIVOS, então
essa condição
//          deve, também, ser verificada.

        while (Vet_MediaFinal[Contador_Alunos] < 0 ||
Vet_MediaFinal[Contador_Alunos] > 10.0)
        {
            printf("Erro: Media Final deve estar entre 0 e
10.");
            printf("\n");
            printf("Informe sua Media Final: ");
            scanf("%f", &Vet_MediaFinal[Contador_Alunos]);
            continue;
        }

// FIM DA SOLUÇÃO PROPOSTA
// -----
// -----
        if (Vet_MediaFinal[Contador_Alunos] >= 7.0)
            printf("%s Voce esta APROVADO(A).",
Vet_Nome_Aluno[Contador_Alunos]);
        else
            if (Vet_MediaFinal[Contador_Alunos] >= 5.0)
                printf("%s Voce esta EM EXAME.",
Vet_Nome_Aluno[Contador_Alunos]);
            else
                printf("%s Voce esta REPROVADO (A).",
Vet_Nome_Aluno[Contador_Alunos]);
        Acum_Media = Acum_Media + Vet_MediaFinal[Contador_
Alunos];

```

5. Resolução da SP

```

Vet_MediaFinal[Contador_Alunos];

    _sleep(1500);
}
MediaTurma = Acum_Media / Contador_Alunos;

system("cls");
printf("\nMédia da turma: %.2f ",MediaTurma);

for (Contador_Alunos=0; Contador_Alunos < 10; Contador_
Alunos++)
{
    if (Vet_MediaFinal[Contador_Alunos] > MediaTurma)
        printf("\n%s - Media: %.2f - Bom Aluno(a).",
            Vet_Nome_Aluno[Contador_Alunos],
            Vet_MediaFinal[Contador_Alunos]);
    else
        if (Vet_MediaFinal[Contador_Alunos] < MediaTurma)
            printf("\n%s - Media: %.2f - Aluno(a) com baixo
desempenho.",
                Vet_Nome_Aluno[Contador_Alunos],
                Vet_MediaFinal[Contador_Alunos]);
        else
            printf("\n%s - Media: %.2f - Aluno(a) Medio(a).",
                Vet_Nome_Aluno[Contador_Alunos],
                Vet_MediaFinal[Contador_Alunos]);
    }

printf("\n");
system("pause");

return 0;
}

(Fonte: Piva Jr. et al., 2012).

```

**Lembre-se**

Os índices que acessam os elementos de um vetor de tamanho n não precisam ser necessariamente enumerados no intervalo $[1, n]$. Esta, no entanto, é a numeração mais óbvia. Nada impede que se defina um intervalo de índices como $[0, n-1]$ ou $[-3, n-3]$ e assim por diante (SOUZA et al., 2011, p. 171).

**Faça você mesmo**

Desenvolva um algoritmo que receba e leia valores em dois vetores com dez posições. Realize uma tarefa simples, como a multiplicação dos elementos, e exiba o resultado em um terceiro vetor.

Faça valer a pena

1. Assinale a alternativa que apresenta uma informação verdadeira acerca das aplicações com o uso de vetores.

- a) Quando estamos desenvolvendo um algoritmo, ao declarar uma variável, estamos informando ao computador que é preciso separar um espaço em memória, que acaba de receber o nome da variável declarada, e este interpreta essa informação de modo a permitir a alocação de um valor nesse espaço que foi determinado.
- b) Os índices que acessam os elementos de um vetor de tamanho n precisam ser necessariamente enumerados no intervalo $[1, n]$.
- c) Não é necessário que somente números ou variáveis inteiras sejam utilizados como índices de um vetor. Permite outros tipos de dados para o índice.
- d) É obrigatória a declaração do tamanho do vetor.
- e) A matriz é uma variável simples, pois é formada por um número finito de variáveis, e homogênea porque essas variáveis são de um mesmo tipo de dado.

2. Complete:

"Somente _____ ou _____ podem ser utilizados como índices de um vetor. Por exemplo, se i for uma variável inteira contendo um número que está dentro do intervalo de índices de um vetor A , $A[i]$ será uma expressão válida" (SOUZA et al., 2011, p. 171).

Assinale a alternativa que melhor completa as lacunas da frase.

- a) tipos de dados/matrizes.
- b) matrizes/vetores.
- c) inteiros/variáveis.
- d) tamanho/vetores.
- e) números/variáveis inteiras.

3. Descreva quais são as formas de atribuição de valores em vetores utilizando uma constante.

4. São verdadeiras:

I. O apontador, ou índice, pode ser uma variável simples, uma constante, ou ainda um cálculo que resulte em um número inteiro.

II. Apesar de não ser necessário declarar explicitamente o tamanho do vetor em um fluxograma, ele possui um tamanho máximo definido pelo problema.

III. Utilizar índices que ultrapassem o maior índice do vetor ou que sejam menores que o menor índice de um vetor é uma operação ilegal e constitui um erro no algoritmo.

- a) I, II e III.
- b) I e II.
- c) II e III.
- d) I e III.
- e) Apenas III.

5. Explique como acontece a alocação dos espaços de memória para os vetores.**6.** Analise a declaração a seguir:

Vl_Aplicacao : Real

Vet_Tx_Juros, Vet_Vl_Juros, Vet_Vl_Corrigido : Vetor [1..12] De Real

Ind_Vet : Inteiro

São, respectivamente, as instruções acima as declarações de:

- a) índice, variável e vetores.
- b) vetores, índice e variável.
- c) variável, vetores e índice.
- d) variável, controle e vetores.
- e) controle, índice e matriz.

7. Complete as lacunas da frase com as palavras de uma das alternativas a seguir.

"Vetores só podem armazenar dados que sejam _____".

- a) de tipos de dados diferentes.
- b) do mesmo tipo de dados.
- c) binários e de texto.
- d) binários.
- e) texto.

Seção 4.2

Operações sobre vetores e matrizes

Diálogo aberto

Olá, aluno, bem-vindo a mais uma seção de estudos. Vamos trabalhar com as operações realizadas com os vetores e as matrizes. Dentro desse contexto, faremos a aproximação do conteúdo teórico e a prática profissional com o seguinte foco: desenvolver um algoritmo que leia os dados de cadastro dos comerciantes e os respectivos valores pagos para manter seus estabelecimentos disponíveis no aplicativo e informe a média de valores pagos anualmente. Lembrando sempre que estamos trabalhando os conceitos e as operações sobre vetores e matrizes, e teremos uma aula inteira dedicada às matrizes. Por esse motivo, você pode ficar à vontade para inserir na solução apresentada uma matriz que auxilie também no armazenamento dos dados de cadastro que foram solicitados.

Nesse sentido, também podemos inserir algumas estruturas que já estudamos, como uma estrutura de repetição controlada por variável. Estamos falando da estrutura "Para" ou "for" em linguagens de programação, pois apenas trabalhamos com a expressão "Para" quando estamos elaborando o algoritmo em pseudocódigo em VisuAlg ou em linguagem natural.

Como o nosso foco agora está em conseguir visualizar outras formas de desenvolver algoritmos ótimos, pense em outras possibilidades de aplicação e construção de soluções em situações do dia a dia. Tenha sempre um olhar empreendedor e visualize nas situações a possibilidade de facilitá-las através da implementação de algoritmos eficientes que permitam trazer a informação ao usuário mais rapidamente, e ainda, a mais correta de acordo com as suas preferências. Pense na evolução das estruturas tecnológicas computacionais e como estas podem te ajudar a aplicar as competências desenvolvidas com os seus estudos.

Sendo assim, considere que os algoritmos, a compreensão das estruturas lógicas de decisão, seleção, repetição e agora as estruturas de dados que permitem organizá-los em vetores e matrizes são apenas o primeiro passo para que você consiga vislumbrar as mais diversas aplicações e potencialidades trazidas com as linguagens de programação e os bancos de dados, sem contar as redes de computadores, como

a internet, que permitem levar esses conceitos e suas aplicações aos mais variados segmentos do mercado.

Não se esqueça: tem dúvidas? Investigue, busque outras fontes de informações. Habitue-se a não ficar apenas com as informações e ensinamentos de uma única origem, e sim, tenha a consciência de que o seu aprendizado será potencializado se agregar às leituras do seu material didático a visita aos *links* sugeridos, o tempo dedicado a assistir aos vídeos recomendados, além de buscar e aprender a aprender com as situações em que é possível desenvolver ainda mais os conhecimentos adquiridos. Desde já, bons estudos e práticas a você!

Não pode faltar

Você já trabalhou com alguns exemplos na aula passada, mas com certeza ainda precisa praticar mais para aprender efetivamente a desenvolver e aplicar as estruturas de dados em vetores e matrizes. Por esse motivo, vamos treinar um pouco mais.

Lembre-se de que vetor é uma variável composta e unidirecional. Matriz também é uma variável composta, porém, pode ser bidirecional ou multidirecional. Suponha que estamos trabalhando com o exemplo de cálculo e apresentação da média dos alunos de uma classe. Vamos trabalhar com a seguinte tabela, como fonte de dados para nossos estudos:

Tabela 10 – Dados dos alunos

| Identificação do Aluno | Nota1 | Nota2 | Nota3 | Média |
|------------------------|-------|-------|-------|-------|
| 1 | 5,0 | 10,0 | 3,0 | 6,0 |
| 2 | 4,0 | 9,0 | 5,0 | 6,0 |
| 3 | 8,0 | 10,0 | 6,0 | 8,0 |
| 4 | 7,0 | 5,0 | 9,0 | 7,0 |

Fonte: Adaptado de Manzano e Oliveira (2010)

O raciocínio para esta operação é o seguinte: ao invés de declarar quatro (4) variáveis para guardar as respectivas médias, nós vamos criar um vetor com 4 posições para alocar as médias dos alunos. Observe como é o português estruturado dessa operação:



Assimile

Algoritmo "Media"

var

```

vet_notas: Vetor [1..4] de real
soma, media: real
i: inteiro
inicio
soma <- 0
para i de 1 ate 4 passo 1 faca
    leia (vet_notas [i])
    soma <- soma + vet_notas[i]
fimpara
media <- soma /4
escreva ("A média do alunos é:", media)
fimalgoritmo

```

Fonte: Adaptado de Manzano e Oliveira (2010)

O algoritmo traz o vetor “vet_notas” como uma estrutura para armazenar as notas do aluno. Em seguida, para realizar o cálculo da média, exibe-o após o procedimento. Podemos notar que a estrutura de um vetor é a seguinte:

Nome_da_variável: nome_da estrutura_de_dado [<tamanho_do_vetor>] de <tipo de dado>

Temos, então, respectivamente, logo a seguir, representada a sintaxe de declaração da estrutura de dado do tipo vetor:

vet_notas: Vetor [1..4] de real

O mesmo procedimento se aplica à declaração das matrizes. É correto afirmar que um vetor é uma matriz unidimensional.

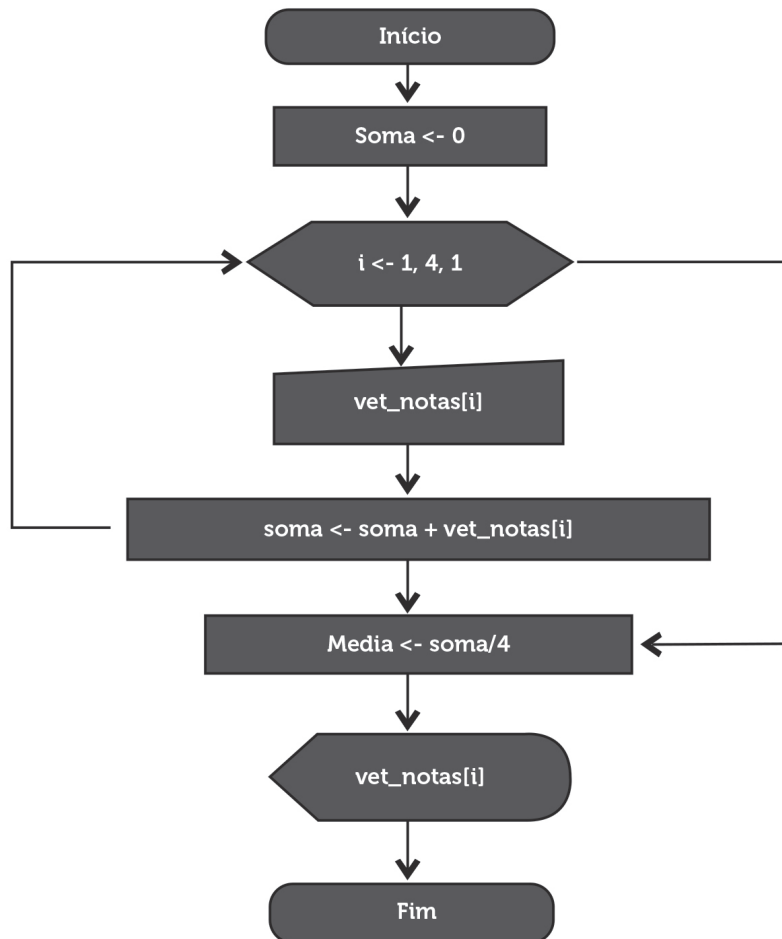


Refleta

A leitura de uma matriz é processada passo a passo, um elemento por vez. A instrução de leitura é leia seguida da variável mais o índice (MANZANO; OLIVEIRA, 2010, p. 109).

Conheça também como se representa um algoritmo que trabalha com o conceito de leitura de dados de uma matriz, pode ser apresentado em diagrama de blocos:

Figura 21 - Diagrama de blocos para leitura dos elementos de uma matriz do tipo vetor



Fonte: Manzano e Oliveira (2010, p. 109)

Agora que você já conhece como se faz a representação por diagrama de blocos, a leitura dos elementos de um vetor, ou matriz, você já pode praticar desenvolvendo um algoritmo que tenha por função armazenar o nome de pessoas. Observe o exemplo de como se faz para implementar essa prática. Siga em frente!



Exemplificando

O pseudocódigo desta operação é trabalhado da seguinte forma em VisuAlg:

Algoritmo "Vetor Nomes"**var**

vet_Nomes: Vetor [1..20] de caractere

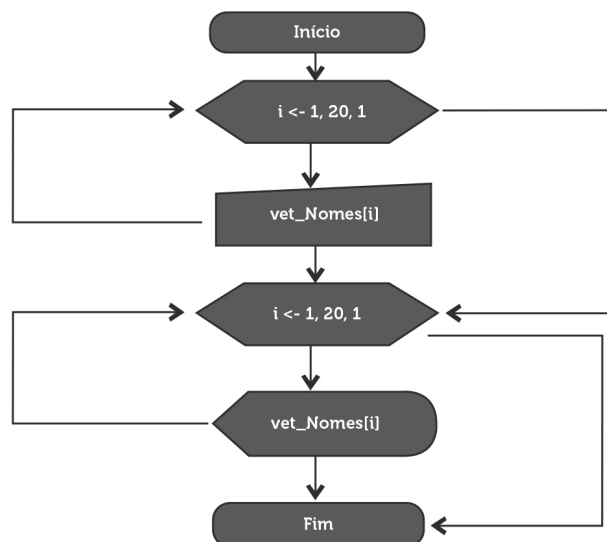
i: inteiro

inicio**para** i **de** 1 **ate** 20 **passo** 1 **faca** **leia** (vet_Nomes [i])**fimpara****para** i **de** 1 **ate** 20 **passo** 1 **faca** **escreva** (vet_Nomes [i])**fimpara****fimalgoritmo**

(Fonte: Adaptado de Manzano e Oliveira (2010, p. 119).

Figura 22 - Diagrama de blocos para ler e escrever 20 nomes

Observe agora, como faz para elaborar o diagrama de blocos do algoritmo apresentado que tem por função ler o nome de 20 pessoas e exibi-los na tela de acordo com a ordem em que foram inseridos:



Fonte: Manzano e Oliveira (2010, p. 118).



Faça você mesmo

Faça você mesmo a transposição deste algoritmo para a linguagem de programação C. Boas práticas!

Mas por que é tão importante aprender a trabalhar com essas estruturas?

A resposta é simples, você precisa saber como o computador armazena e organiza informações por meio de um algoritmo. Para que isso possa acontecer, há uma classificação dos vetores e matrizes que buscam ordenar as informações dessas estruturas de dados. É possível, portanto, organizar os dados contidos nas matrizes por: ordem numérica, alfabética ou alfanumérica, sempre de acordo com a tabela de referência ASCII.

Quando falamos em classificação por ordem numérica, esta pode ser tanto crescente quanto decrescente. Já a classificação por ordem alfabética pode ser de "A – Z" ou de "Z – A", sempre será realizada primeiro a ordenação dos caracteres em letra maiúscula para depois acontecer a ordenação dos caracteres que estão em letra minúscula. A classificação alfanumérica pode ser realizada por ordem ascendente ou descendente como na ordenação alfabética. São válidos todos os símbolos da tabela ASCII (MANZANO; OLIVEIRA, 2010). Nesse sentido, foram desenvolvidos para aumentar a eficiência de algoritmos, alguns métodos de classificação conhecidos como:

- **Inserção:** esta classificação pode ser direta; por busca binária ou por incrementos decrescentes, também conhecidos como *shellsort*.

- **Troca:** também chamado de *bubblesort*, ou método bolha; outro método citado pode ser o da agitação, ou *shakesort*; o método do pente, chamado de *combsort*, método conhecido como *quicksort*, ou de partição e troca. Este método é considerado relativamente simples, eficaz, porém, lento e recomendado para a ordenação de pequena quantidade de dados.

- **Seleção:** direta; em árvore também chamado de *heapsort* e o método da árvore amarrada conhecido como *threadedheapsort*.

- **Distribuição de chaves:** também conhecido como método da indexação direta *radixsort*.

- **Intercalação:** simples que é o *mergesort*, ou o método da intercalação de sequências naturais.

- **Cálculo de endereços:** este evidencia o uso das listas de colisão e o método da solução postergada das colisões (MANZANO; OLIVEIRA, 2010).



Pesquise mais

Conheça todos os métodos indicados no *link*: <<http://nicholasandre.com.br/sorting/>>. Acesso em: 27 jul. 2015.

A lógica computacional dos métodos de ordenação elencados acima tem o seguinte pressuposto, observe nas linhas a seguir como são realizadas essas ações. Vamos considerar que é necessário ordenar os elementos de um vetor que está a princípio organizado da seguinte forma:

Figura 23 - Representação da ordenação inicial de um vetor A

| | | | | | |
|---|---|---|---|---|--------------------|
| 7 | 6 | 4 | 3 | 2 | Elementos do vetor |
| 1 | 2 | 3 | 4 | 5 | Índice |

Fonte: Adaptado de Manzano e Oliveira (2010)

Para ordenar os elementos do vetor acima, em ordem crescente e não decrescente como apresentado, a princípio, vamos representá-los da seguinte forma, o chamaremos de Vetor A: $A[1] = 7$, $A[2] = 6$, $A[3] = 4$, $A[4] = 3$, $A[5] = 2$. Sendo assim, a fim de inserir o método de ordenação por troca, é preciso, primeiramente, que você compreenda o processo todo. Para que a troca aconteça, é necessário aplicar o método de propriedade distributiva. Confira a lógica envolvida:

1. Primeiramente, há a comparação do elemento de $A[1]$ com os demais do vetor. A mesma comparação acontece com os elementos subsequentes, ou seja, o elemento de $A[2]$ será comparado aos seguintes $A[3]$, $A[4]$ e $A[5]$. Assim, acontecerá sucessivamente com os demais elementos do vetor.
2. Quando a comparação de $A[1]$ e $A[2]$ resulta em o elemento do primeiro ser maior que o da segunda posição, será realizada a troca.
3. Para aplicar a propriedade distributiva, o elemento atualizado de $A[1]$ será agora comparado com o próximo elemento da sequência, no caso $A[3]$. O mesmo raciocínio será aplicado, ou seja, se o elemento de $A[1]$ for maior que o de $A[3]$, então, será efetuada a troca.
4. O mesmo procedimento de comparação, para verificar qual elemento das posições subsequentes será realizado e, se o elemento for maior que o de $A[1]$, haverá a troca. Essa comparação acontece até a última posição do vetor, no caso do exemplo, $A[5]$. Depois de realizar a comparação de $A[1]$ com os elementos das demais posições, inicia-se o mesmo processo para comparar o elemento de $A[2]$ com os demais do vetor. A composição do vetor neste caso passa a ser a seguinte: $A[1] = 2$, $A[2] = 3$, $A[3] = 4$, $A[4] = 6$, $A[5] = 7$.

Para dados do tipo caractere, acontece o mesmo procedimento, porém, o valor a comparar é o que está estabelecido pela tabela ASCII, com as devidas distinções entre caracteres maiúsculos e minúsculos.

Alguns questionamentos podem surgir, por exemplo, como realizar tal ordenação ou mesmo a busca de elementos de um vetor que seja grande, ou seja, que contenha uma quantidade de dados armazenados superior a 1000 elementos? Algumas buscas ou pesquisas podem ser citadas, uma delas é a pesquisa simples, também conhecida como sequencial. A sua desvantagem é que pode ser um pouco lenta por ter de percorrer sequencialmente cada um dos elementos da matriz ou vetor. Observe a seguir um exemplo de algoritmo que realiza a pesquisa simples em um vetor com 10 nomes (MANZANO; OLIVEIRA, 2010, p. 130):

```

algoritmo "Pesquisa"
var
  NOME: Vetor[1..10] de caracter
  I: inteiro
  PESQ, RESP: caracter
  ACHA: logico
inicio
  escreval ("Informe 10 nomes:")
  para I de 1 ate 10 faca
    leia (NOME[I])
  fimpara
  //trecho de início de pesquisa sequencial
  RESP <- "sim"
  enquanto (RESP = "sim") faca
    I <- 1
    ACHA <- falso
    escreval ("Insira o nome para pesquisa:")
    leia (PESQ)
    enquanto (I <= 10) e (ACHA = falso) faca
      se (PESQ = NOME[I]) entao
        ACHA <- verdadeiro
        escreval (PESQ, " Foi localizado na posição:", I)
      senao
        I <- I + 1
      escreval (PESQ, " Não foi localizado!")
    fimse
  fimenquanto
  escreva ("Deseja continuar? Responda (sim/não)")
  leia (RESP)
fimenquanto
fimalgoritmo

```

Sem medo de errar

A solução apresentada a seguir visa permitir a leitura dos dados de cadastro dos comerciantes, bem como os respectivos valores pagos, para que possam permanecer com os anúncios no aplicativo. Sua tarefa é desenvolver um algoritmo que permita essa ação. Acompanhe a proposta, crie e desenvolva seu próprio algoritmo com base nos estudos realizados até aqui e boas práticas!

algoritmo "Cadastro_Comerciantes"

// Função: Le o Nome dos comerciantes e os Valores pagos.

// Calcula a média de pagamentos realizados e determina a situação de cada comerciante

// (Em dia, Em Atraso ou Renovação).

// Os dados de cada comerciante são carregados em vetores.

Var

Vet_Nome_Comerciante : Vetor [1..10] de **Caractere**

Vet_Pagto: Vetor [1..10] de **Real**

Vet_MediaPgto : Vetor [1..10] de **Real**

Contador_Comerciantes : **Inteiro**

inicio

para Contador_Comerciantes de 1 **ate** 10 **faca**

Escreval ("Informe Nome do Comerciante - ", Contador_ Comerciante, " :")

Leia (Vet_Nome_Comerciante[Contador_Comerciante])

Escreval ("Informe o pagamento realizado:")

Leia (Vet_Pagto[Contador_Comerciante])

// Calcula e mostra a Média de pagamentos do comerciante

Vet_MediaPgto[Contador_ Comerciante]:= Vet_Pagto[Contador_ Comerciante]

se (Vet_MediaPgto[Contador_Comerciante] >= 100.0) **entao**

Escreval(Vet_Nome_Comerciante[Contador_ Comerciante], " sua Média de pagamentos realizados é ", Vet_MediaPgto[Contador_ Comerciante]:2:2 " - Você está Em dia.")

senao

Se (Vet_MediaPgto[Contador_ Comerciante] >= 50.0) **entao**

Escreval(Vet_Nome_Comerciante[Contador_Alunos], " Sua Média de pagamentos realizados é ", Vet_MediaPgto[Contador_ Comerciante]:2:2 " - Você está Em atraso.")

senao

Escreval(Vet_Nome_Comerciantes[Contador_ Comerciante], " Sua Média de pagamentos realizados é ", Vet_MediaPgto[Contador_ Comerciante]:2:2 " - Você está em período de Renovação.")

fimse

fimse

Escreval // saltar uma linha

FimPara

Fimalgoritmo

(Fonte: Adaptado de Piva Jr. (2013, p.280))



Atenção!

O algoritmo visa evidenciar a situação em que se encontra o cadastro do comerciante e exibe o seu *status*, além da média de pagamentos que foram realizados.



Lembre-se

Um importante aspecto a ser considerado é que, na manipulação de uma matriz unidimensional, utiliza-se uma única instrução de laço (enquanto, para ou repita). No caso de matrizes com mais dimensões, deve ser utilizado o número de laço relativo ao tamanho de sua dimensão. Dessa forma, uma matriz de duas dimensões deve ser controlada com dois laços, a de três dimensões com três laços e assim por diante (MANZANO; OLIVEIRA, 2010, p. 139).

Avançando na prática

Pratique mais

Instrução

Desafiamos você a praticar o que aprendeu transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois as compare com as de seus colegas.

Operações sobre vetores e matrizes

| | |
|---------------------------------------|---|
| 1. Competência de Fundamentos de Área | Conhecer os princípios e conceitos que envolvem o aprendizado em construção de algoritmos e programação e a sua importância para o universo do desenvolvimento de sistemas. |
| 2. Objetivos de aprendizagem | Conhecer quais são as operações realizadas por vetores e matrizes. |
| 3. Conteúdos relacionados | Operações sobre vetores e matrizes |
| 4. Descrição da SP | Desenvolver um algoritmo que permita cadastrar 20 comerciantes e, para cada um, até 8 anúncios. Isso quer dizer que você poderá implementar uma matriz que comporte essa estrutura. Há uma recomendação a seguir de acordo com Piva Jr., siga o exemplo e implemente outras estruturas. |
| 5. Resolução da SP | Algoritmo "Preenche_Matriz" // Função: Preencher matriz, contendo 20 e 8 colunas por linha, com o //caractere "L" e, em seguida mostra seu conteúdo em tela Var Mat_Letra : Vetor [1..10, 1..20] de Caractere IndLin, IndCol : Inteiro |

| | |
|--|---|
| | <pre> Início // Carregar a matriz Para IndLin := 1 ate 10 faça Para IndCol de 1 ate 20 faça Mat_Letra[IndLin,IndCol] := "L" FimPara FimPara Escreval ("Conteúdo da Matriz Letra") Escreval // Mostra o conteúdo da matriz Para IndLin de 1 ate 10 faça Para IndCol := 1 ate 20 faça Escreva(Mat_Letra[IndLin,IndCol]) // não salta linha FimPara Escreval // salta linha FimPara Fimalgoritmo (Fonte: Piva Jr. et al. (2013, p. 303) </pre> |
|--|---|



Lembre-se

"A declaração de uma matriz segue as mesmas regras sintáticas para a definição de vetores, alterando somente a dimensão dessa variável, ou seja, o seu tamanho. Da mesma forma que em vetor, a declaração da dimensão da matriz requer reserva de memória, mesmo que não utilizemos na sua totalidade" (PIVA JR. et al., 2012, p. 287).



Faça você mesmo

Desenvolva a solução apresentada também em linguagem de programação C.

Faça valer a pena!

1. Complete as lacunas da frase.

"_____ é uma variável _____ e _____. Matriz também é uma variável _____, porém, pode ser _____ ou multidirecional".

- matriz/simples/bidirecional/composta/bidirecional.
- vetor/simples/bidirecional/composta/bidirecional.
- vetor/composta/bidirecional/simples/bidirecional.
- vetor/composta/unidirecional/composta/bidirecional.
- matriz/composta/unidirecional/composta/bidirecional.

2. Analise as afirmações e assinale a alternativa correspondente.

I. A estrutura de um vetor é a seguinte: **Nome_da_variável: nome_da estrutura_de_dado** [<tamanho_do_vetor>] **de** <tipo de dado>.

II. É correto afirmar que um vetor é uma matriz unidimensional.

III. É possível, portanto, organizar os dados contidos nas matrizes por: ordem numérica, alfabética ou alfanumérica

a) V – V – V.

b) F – F – F.

c) F – V – V.

d) V – F – F.

e) V – V – F.

3. Das afirmações a seguir, quais delas são verdadeiras? Assinale a alternativa correspondente.

I. Inserção: esta classificação pode ser direta; por busca binária ou por incrementos decrescentes, também conhecidos como *shellsort*.

II. Troca: apenas será possível realizar a troca de elementos entre as posições do vetor se este for do tipo real.

III. Seleção: direta; em árvore também chamado de *heapsort* e o método da árvore amarrada conhecido como *threadedheapsort*.

a) V – V – V.

b) F – F – F.

c) V – F – V.

d) F – F – V.

e) V – F – F.

4. Descreva quais são os algoritmos que implementam ordenação dos elementos de uma matriz ou vetor.

5. Explique o procedimento de troca de elementos de uma matriz para a ordenação simples sequencial.

6. O comando que representa a declaração de um vetor com 10 posições é:

- a) Mat_Letra : Vetor [1..10, 1..20] de Caractere.
- b) Vet_Pagto: Vetor [1..10] de Real.
- c) IndLin, IndCol : Inteiro.
- d) Contador_Comerciantes : Inteiro.
- e) A[1]= 2, A[2]=3, A[3]= 4, A[4]= 6, A[5]= 7.

7. Analise a frase a seguir e assinale a alternativa que contém o conceito que foi apresentado. "A comparação será realizada até que todo o vetor seja percorrido ou se encontre o valor correspondente que atenda à condição imposta na instrução de repetição".

- a) vetor.
- b) matriz.
- c) pesquisa simples sequencial.
- d) descrição de bubblesort.
- e) descrição de quicksort.

Seção 4.3

Os vetores como estruturas de dados

Diálogo aberto

Estamos estudando nesta unidade o que são e como se comportam as estruturas de dados em vetor ou matriz. Nesse contexto, vamos aproximar os conteúdos teóricos e práticos desta unidade com os trabalhados anteriormente. Podemos, nesta aula, trabalhar da seguinte forma para que essa aproximação aconteça: vamos desenvolver um algoritmo que permita somar os conteúdos de um vetor. Para tal, os elementos serão separados em números ímpares e pares e o algoritmo deverá ainda efetuar a soma deles de acordo com essa classificação. Mas afinal, como essa aplicação pode ser transposta para a realidade do mercado de trabalho?

Como estamos trabalhando a ideia de desenvolvimento de um protótipo de aplicativo que deve apresentar as suas principais funcionalidades e representá-las em forma de algoritmos, vamos implementá-lo com o olhar voltado à ação de alocar, assim que o usuário finaliza a compra, os valores finais da compra realizada em vetores. Sua função é elaborar um algoritmo que mostre os conteúdos desse vetor e, ainda, some os números pares encontrados, fazendo a mesma operação de soma para os números ímpares. Para elaborar este algoritmo, é preciso que você considere, a princípio, uma limitação do tamanho do vetor, na ordem de 258 posições. Você pode supor que este será aplicado para ordenar, de acordo com essa lógica, as quantidades de convites vendidos para os jantares, considerando o que foi sugerido na unidade anterior de nossos estudos.

A fim de resgatar os conteúdos básicos estudados até o momento, que podem auxiliar no desenvolvimento do algoritmo proposto, você precisa saber como declarar o vetor e de que forma podem ser armazenadas tais informações.

Além disso, você também precisa refletir sobre o algoritmo de ordenação que poderá usar para realizar a busca dos valores desse vetor. No entanto, lembre-se também de que sempre há uma estrutura de repetição para que se possa estabelecer as regras de ordenação e consulta aos elementos do vetor.

Nesse sentido, você pode utilizar, para resolver a situação proposta, a estrutura de repetição controlada por variável "para" ou "for", como na sintaxe das linguagens de programação.

Agora que você já estudou as principais estruturas de decisão, seleção e repetição, conheça e pratique sua implementação, interagindo com a estrutura de dado em vetor, ou, como também é conhecido, matriz unidimensional.

Ficam as recomendações de estudos e leituras complementares para que você possa evoluir ainda mais com as implementações sugeridas ao longo do seu material, além da resolução dos exercícios, que ajudam a fixar as informações trabalhadas para as aulas. Bons estudos e práticas!

Não pode faltar

Estudamos até aqui as estruturas de dados chamadas vetores e conhecemos alguns conceitos e representações de matrizes bidirecionais e multidirecionais. Nesse contexto, vamos retomar brevemente algumas informações básicas sobre vetores. Siga em frente!

O algoritmo a seguir evidencia algumas informações importantes que vamos reforçar:

```
var
aluno: caractere
notas: vetor[1..3] de real
x: inteiro
media: real
```

Temos na declaração do vetor, como vimos, primeiramente, o nome do vetor, no caso "notas". seguido da identificação da estrutura com o comando "vetor" e a delimitação do seu tamanho entre colchetes. E, em seguida, há a declaração do tipo de dado, que foi especificado para este, "real".

```
início
// seção de Comandos
escreval("Média de alunos")
escreva("Digite o nome do aluno: ")
leia(aluno)
```

Aqui é inserida a estrutura de repetição controlada por variável, que realizará a alocação até que o valor do índice seja igual a 3. Nesse caso, será executado o bloco de comandos até o índice atingir o valor especificado.

```
para x de 1 ate 3 faça
    escreval("informe a nota: ")
    leia(notas[x])
fimpara
media <- (notas[1] + notas[2] + notas [3]) / 3
limpatela
escrava( aluno.)
escreval( "Média: ". media)
finalgoritmo
```

Neste bloco, temos a variável média recebendo os valores contidos no vetor notas, nas posições delimitadas pelo índice entre os colchetes.

As demais instruções apresentadas no algoritmo acima, com exceção do “limpatela”, que como o próprio nome diz tem por função deixar a tela de exibição dos resultados sem outras informações que não sejam pertinentes àquele processamento, são instruções de entrada e saída de dados, que, neste momento, já foram bastante trabalhadas e conhecidas no decorrer das aulas e estrutura de seu material didático.

Dessa forma, como podemos armazenar e localizar os elementos de um vetor?

Para este caso, seguiremos um exemplo de Souza et al. (2011) que retrata bem a forma como podemos trabalhar com um vetor, que permite armazenar “Q” elementos, a definir o seu tamanho de acordo com o problema que for especificado.

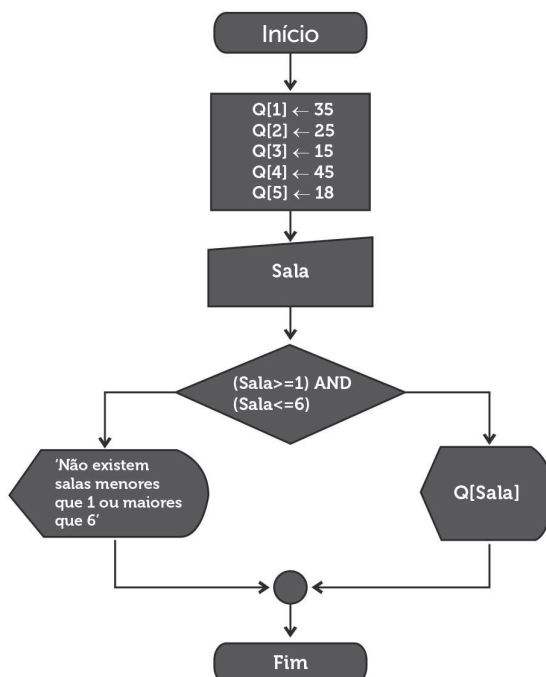
Esse vetor deve armazenar a quantidade de alunos de cada sala. Não se esqueça de tentar elaborar e praticar o desenvolvimento das representações de acordo com cada um dos conceitos estudados. Os diagramas de blocos ou fluxogramas podem ajudar no desenvolvimento e implementação das estruturas que serão utilizadas para as ações que o algoritmo deve executar.

Observe como é a representação dessa operação em diagrama de blocos:



Assimile

Figura 23 - Fluxograma seguro para armazenar e localizar elementos de um vetor



Fonte: Souza et al. (2011, p. 173)



Faça você mesmo

De acordo com o fluxograma acima, tente elaborar o algoritmo em pseudocódigo no VisuAlg e implemente-o em linguagem de programação C, utilizando o Dev C++.

O algoritmo representado no fluxograma acima exibe o elemento do vetor "Q" de acordo com o índice que é referente à sala. Dessa forma, temos para cada elemento do vetor a quantidade de alunos de cada sala respectivamente. Se solicitarmos que seja exibida a quantidade de alunos da sala em que se encontra, por exemplo, o elemento 4 do vetor, teremos a exibição do valor 45, equivalente à quantidade de alunos. A estrutura de decisão implementada tem por função verificar se o índice que foi solicitado realmente está de acordo com o tamanho do vetor, então, realiza-se o teste e, se o valor inserido for inferior a 1 ou superior a 6, será exibida a mensagem de que não existem salas correspondentes àquela solicitação.



Refleta

Embora uma variável tipo vetor armazene um conjunto de elementos simultaneamente, a manipulação desses elementos é individual, como se fosse um conjunto de variáveis de mesmo nome, identificadas por números individuais (SOUZA et al., 2011, p. 169).

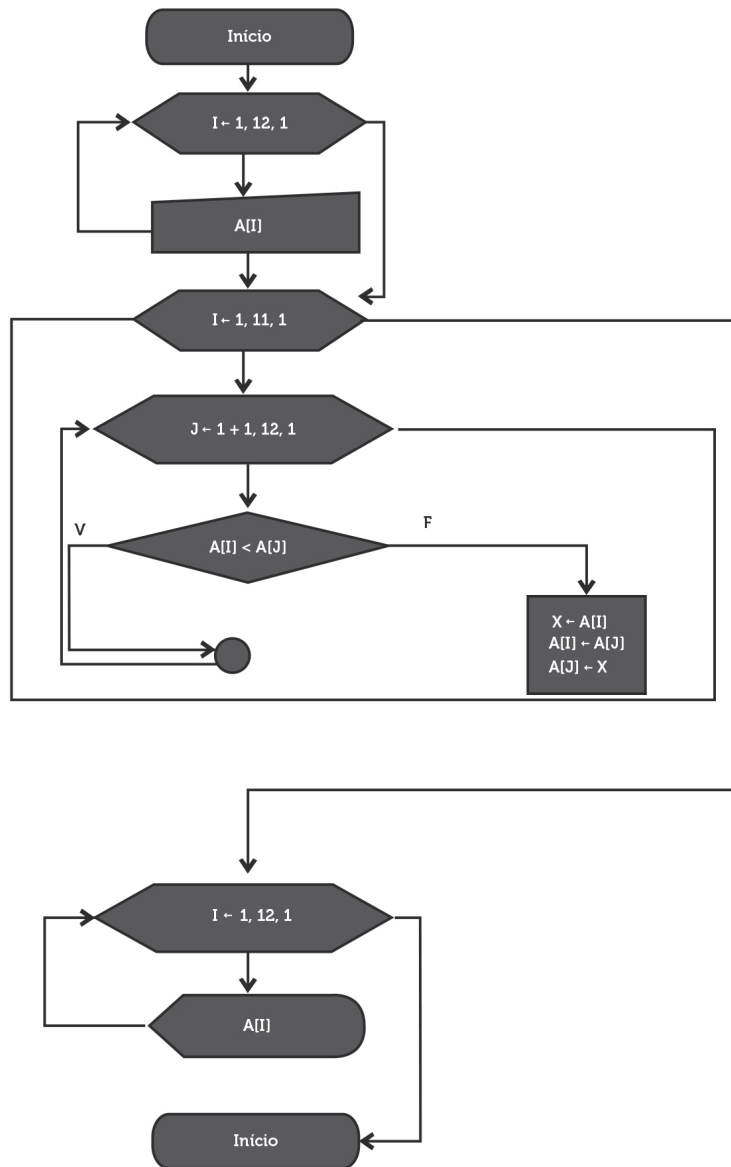
Agora, vamos trabalhar com mais um exemplo de aplicação de algoritmos com as estruturas de dados de vetor. Considere que uma empresa comercializa seus produtos pela internet e precisa iniciar um controle de vendas por cliente e, ainda, o sistema de vendas on-line deve ordenar as compras que ele realizou em um vetor. Veja no exemplo a seguir uma maneira de implementar um algoritmo que precisa realizar essa tarefa.



Exemplificando

Primeiramente, vamos estabelecer qual será o tamanho do vetor. Para essa representação faremos um vetor com 12 posições, ou seja, que armazene até doze elementos e, em ordem decrescente, apresente todos os elementos armazenados no vetor. Siga o exemplo que Manzano e Oliveira (2010) sugerem para realizar essa ordenação. Vamos lá! Confira o fluxograma, ou diagrama de blocos, desta operação.

Figura 24 - Fluxograma ordenação de elementos em um vetor



Fonte: Manzano e Oliveira (2010, p. 216)

Veja, também, o algoritmo em pseudocódigo desta operação em VisuAlg:

algoritmo "Classificação"

var

A: vetor [1..12] de inteiro

```

I, X, J: inteiro
início
  para I de 1 até 12 passo 1 faça
    leia A[I]
  fimpara
  para I de 1 até 11 passo 1 faça
    para J de I + 1 até 12 passo 1 faça
      se (A[I] < A[J]) então
        X ← A[I]
        A[I] ← A[J]
        A[J] ← X
      fimse
    fimpara
  fimpara
  para I de 1 até 12 passo 1 faça
    escreva A[I]
  fimpara
finalgoritmo

```

Fonte: Manzano e Oliveira (2010, p. 216)



Pesquise mais

O artigo *O Varal de Roupas* traz de forma simples e bastante didática uma aplicação de vetores em linguagem de programação C, em que, seguindo um determinado contexto, é desenvolvido o algoritmo e implementado na plataforma de programação. Disponível em: <[http://www.iiisci.org/Journal/CV\\$/risci/pdfs/IXA179PU.pdf](http://www.iiisci.org/Journal/CV$/risci/pdfs/IXA179PU.pdf)>. Acesso em: 27 jul. 2015.

Agora que você já viu alguns exemplos, pratique mais este exercício proposto por Piva Jr. et al. (2012, p. 270), em que o algoritmo precisa ler 20 números do tipo de dado inteiro, inseridos pelo usuário e armazenados em um vetor. O algoritmo deverá ainda ler e mostrar o conteúdo do vetor em ordem inversa à da leitura; exibir o primeiro e o último número carregado no vetor; a soma dos elementos do vetor; e, por fim, o algoritmo deverá exibir a média aritmética dos elementos do vetor.

Em VisuAlg

Algoritmo "Vet_Numeros_Inteiros"

Var

Vet_Num_Int : Vetor [1..20] De Inteiro

Ind_Vet, Soma_Inteiros : Inteiro

Parar : Caractere

Inicio

Para Ind_Vet de 1 Ate 20 faca

 LimpaTela

 Escreval("Informe o ",Ind_Vet,"o. Numero Inteiro")

 Leia(Vet_Num_Int[Ind_Vet])

 Soma_Inteiros := Soma_Inteiros + Vet_Num_Int[Ind_Vet]

FimPara

// Mostrar conteudo do Vetor, somatorio e media dos numeros contidos nele

LimpaTela

Escreval("Conteudo do Vetor na ordem Inversa do carregamento")

Escreval

Para Ind_Vet de 20 ate 1 passo -1 faca

 Escreval(Vet_Num_Int[Ind_Vet])

FimPara

Escreval

Escreval("<< Pressione uma tecla qualquer >>")

Leia(Parar)

Escreval

Escreval("Primeiro Número Carregado no Vetor : ",Vet_Num_Int[1])

Escreval

Escreval("Ultimo Número Carregado no Vetor : ",Vet_Num_Int[20])

Escreval

Escreval("Soma dos Números contidos no Vetor : ",Soma_Inteiros)

Escreval

Escreval("Média dos Números contidos no Vetor : ",Soma_Inteiros Div 20)

FimAlgoritmo

Fonte: Piva Jr. et al. (2012, p. 26).

Em Linguagem de Programação C

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
int main()
```

```
{
```

```
    int Vet_Num_Int[20];
```

```
    int Ind_Vet;
```

```
    int Soma_Inteiros = 0;
```

```
    for (Ind_Vet = 0; Ind_Vet < 20; Ind_Vet ++)
```

```
    {
```

```
        system("cls");
```

```
        printf("Informe o Numero Inteiro do elemento --> %i%s",
```

```
            Ind_Vet, " - do Vetor: ");
```

```
        scanf("%i", &Vet_Num_Int[Ind_Vet]);
```

```
        Soma_Inteiros = Soma_Inteiros + Vet_Num_Int[Ind_Vet];
```

```
    }
```

```
    system("cls");
```

```
    printf("\nConteudo do Vetor na ordem Inversa do carregamento\n\n");
```

```
    for (Ind_Vet = 19; Ind_Vet >= 0; Ind_Vet --)
```

```
    {
```

```
        printf("%i%s",Vet_Num_Int[Ind_Vet], " ");
```

```
    }
```

```

printf("\n\nPrimeiro  Número Carregado no Vetor : %i",Vet_Num_Int[0]);
printf("\n\nUltimo   Número Carregado no Vetor : %i",Vet_Num_Int[19]);
printf("\n\nSoma   dos Números contidos no Vetor : %i",Soma_Inteiros);
printf("\n\nMédia dos Números contidos no Vetor : %i",Soma_Inteiros / 20);
printf("\n\n");
system("pause");

return 0;
}

```

Fonte: Piva Jr. et al. (2012, p. 26).

Tente resolver, primeiramente, no VisuAlg e, depois, em linguagem de programação C. Siga em frente!

SEM MEDO DE ERRAR

Para resolver a situação-problema proposta, vamos utilizar mais um exemplo de Piva Jr. et al. (2012). Observe que o algoritmo proposto armazena em vetores diferentes números pares e ímpares, e ainda estabelece valores de referência, no caso, que ordene elementos de 348 a 863. Além disso, o algoritmo ainda precisa exibir esses valores e efetuar a soma dos elementos pares e ímpares dos respectivos vetores. Mas como aplicar esse exemplo à situação proposta? Precisamos elencar, nesse intervalo, valores que sejam pertinentes à regra de negócios estabelecida e, com isso, podemos realizar essa operação. Siga o exemplo e implemente em VisuAlg.

Algoritmo "Numeros_Pares_e_Impares"

Var

Vet_Num_Par, Vet_Num_Impar : Vetor [1..258] De Inteiro

Numero, Ind_Vet, Ind_VetP, Ind_VetI, Cont_lin : Inteiro

Soma_Pares, Soma_Impares : Inteiro

Parar : Caractere

Inicio

LimpaTela

Ind_VetP := 1

```

Ind_VetI := 1
Para Numero de 348 Ate 863 faca
  Se (Numero Mod 2 = 0) entao
    Vet_Num_Par[Ind_VetP] := Numero
    Soma_Pares := Soma_Pares + Numero
    Ind_VetP := Ind_VetP + 1
  Senao
    Vet_Num_Impar[Ind_VetI] := Numero
    Soma_Impares := Soma_Impares + Numero
    Ind_VetI := Ind_VetI + 1
  FimSe
FimPara
// Mostra o conteudo dos vetores (Vet_Num_Par e Vet_Num_Impar) e
// a Soma dos numeros pares e impares contidos neles
  LimpaTela
  Escreval("Vetor Num.Pares Vet Num.Impares")
  Escreval
  Cont_Lin := 1
  Para Ind_Vet de 1 Ate 258 Faca
    Se (Cont_Lin < 20) entao
      Escreval(Vet_Num_Par[Ind_Vet], " ", Vet_Num_Impar[Ind_Vet])
      Cont_Lin := Cont_Lin + 1
    senao
      Escreval
      Escreval("<< Pressione uma tecla qualquer >>")
      Leia(Parar)
      LimpaTela
      Escreval("Vetor Num.Pares Vet Num.Impares")
      Escreval
      Cont_Lin := 1

```

```

        Escreval(Vet_Num_Par[Ind_Vet], " ", Vet_Num_ImPar[Ind_Vet])
    FimSe
FimPara
Escreval
    Escreval("Soma dos Pares : ", Soma_Pares)
    Escreval
    Escreval("Soma dos Impares: ", Soma_Impares)
FimAlgoritmo

```

Fonte: Piva Jr. et al. (2012, p. 20)



Atenção!

Verifique que, para facilitar a operação, foi necessário trabalhar com dois vetores, um para cada categoria que deve ser avaliada!



Lembre-se

O uso de vetores para solucionar problemas, através da construção de algoritmos, dependerá sempre da análise criteriosa do problema. Todavia, é uma escolha que implicará a organização do programa e, consequentemente, no seu desempenho quando em execução (PIVA JR. et al., 2012, p. 269).

Avançando na prática

Pratique mais

Instrução

Desafiamos você a praticar o que aprendeu transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois as compare com as de seus colegas.

Os vetores como estruturas de dados I

1. Competência de Fundamentos de Área

Conhecer os princípios e conceitos que envolvem o aprendizado em construção de algoritmos e programação e a sua importância para o universo do desenvolvimento de sistemas.

| | |
|------------------------------|--|
| 2. Objetivos de aprendizagem | Saber e conhecer quais são as operações em que se aplicam as estruturas de dados em vetores. |
| 3. Conteúdos relacionados | Os vetores como estruturas de dados I |
| 4. Descrição da SP | Resolução do exercício proposto no <i>Sem Medo de Errar</i> em linguagem de programação C. |
| 5. Resolução da SP | <pre> #include <stdio.h> #include <stdlib.h> int main() { int Vet_Num_Par[258], Vet_Num_Impar[258]; int Numero, Ind_Vet, Ind_VetP, Ind_VetI, Cont_Lin; int Soma_Pares = 0; int Soma_Impares = 0; system("cls"); Ind_VetP = 0; Ind_VetI = 0; for (Numero = 348; Numero < 864; Numero++) { if ((Numero % 2) == 0) { Vet_Num_Par[Ind_VetP] = Numero; Soma_Pares = Soma_Pares + Numero; Ind_VetP = Ind_VetP + 1; } else { Vet_Num_Impar[Ind_VetI] = Numero; Soma_Impares = Soma_Impares + Numero; Ind_VetI = Ind_VetI + 1; } } system("cls"); printf("\n\nVetor Num. Pares Vet Num. Impares\n"); Cont_Lin = 1; for (Ind_Vet=0; Ind_Vet < 258; Ind_Vet++) { if (Cont_Lin < 20) { printf("\n%5s%i%15s%i", "Vet_Num_Par[Ind_Vet]", "Vet_Num_Impar[Ind_Vet]"); Cont_Lin = Cont_Lin + 1; } else { printf("\n\n>>>> "); system("pause"); system("cls"); printf("\nVetor Num. Pares Vet Num. Impares\n"); Cont_Lin = 1; printf("\n%5s%i%15s%i", "Vet_Num_Par[Ind_Vet]", "Vet_Num_Impar[Ind_Vet]"); } } printf("\n\nSoma dos Pares : %i", Soma_Pares); printf("\n\nSoma dos Impares: %i", Soma_Impares); </pre> |

| | |
|--|--|
| | <pre>printf("\n\n>>> "); system("pause"); return 0; }</pre> <p>Fonte: Piva Jr. et al. (2012, p. 20)</p> |
|--|--|



Lembre-se

Quando se define um vetor, na realidade, se está requisitando ao sistema operacional do computador para que reserve uma área contínua de memória, a fim de armazenar os valores de um mesmo tipo de dado (SOUZA et. al., 2011, p. 168).



Faça você mesmo

Investigue outras funcionalidades além do uso de vetores e matrizes. Verifique de que forma é possível criar arquivos e outras estruturas de dados em ambientes computacionais.

Faça valer a pena!

1. Complete a frase com os conceitos apresentados em uma das alternativas a seguir.





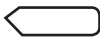
"Embora uma variável tipo _____ armazene um conjunto de elementos _____, a manipulação desses elementos é _____, como se fosse "um conjunto de variáveis de mesmo nome, identificadas por números individuais" (SOUZA et al., 2011, p. 169).

- a) vetor/simultaneamente/individual.
- b) individual/vetor/simultaneamente.
- c) matriz/individual/vetor.
- d) conjunto/variáveis/simultaneamente.
- e) simultaneamente/vetores/matrizes.

2. Analise a instrução e assinale a alternativa que corresponde à sua respectiva descrição em algoritmos: $Q[1] \leftarrow 35$.

- a) Matriz bidirecional "Q"; índice que aponta para o elemento 1 do vetor; elemento da posição "1" do vetor recebe o valor 35.
- b) Vetor "Q"; elemento 35 do vetor que aponta para o elemento 1.
- c) Matriz multidirecional "Q"; elemento 35 do vetor que aponta para o elemento 1.
- d) Vetor bidirecional "Q"; índice que aponta para o elemento 1 do vetor; elemento da posição "1" do vetor recebe o valor 35.
- e) Vetor "Q"; índice que aponta para o elemento 1 do vetor; elemento da posição "1" do vetor recebe o valor 35.

3. Assinale a alternativa que contém o elemento do fluxograma apropriado para representar a operação: $(Sala \geq 1) \text{ OR } (Sala \leq 6)$.

- a) Seta indicativa de fluxo 
- b) Decisão 
- c) Terminação 
- d) Entrada de dados 
- e) Exibição de dados 

4. Descreva o raciocínio empregado para realizar a seguinte operação: notas: vetor[1..3] de real.

5. Explique a expressão: $media \leftarrow (notas[1] + notas[2] + notas[3]) / 3$.

6. Assinale a alternativa que contém a expressão de realização de operações utilizando vetores, de acordo com a sintaxe e lógica corretas.

- a) $Vet_Num_Par[Ind_VetP] = Numero$.
- b) $Vet_Num_Par[Ind_VetP] = Numero$.
- c) $Vet_Num_Par[""] = Numero$.
- d) $Vet_Num_Par[Ind_VetP] = Numero$.
- e) $Vet_Num_Par["_"] = Numero$.

7. A instrução “se $A[I] < A[J]$ entao” representa:

- a) Estrutura de decisão com teste lógico entre os índices utilizados para manipular a informação na matriz bidirecional.
- b) Estrutura de repetição com teste lógico no início entre os índices utilizados para manipular a informação no vetor.
- c) Estrutura de repetição com teste lógico no final entre os índices utilizados para manipular a informação no vetor.
- d) Estrutura de seleção com teste lógico entre os índices utilizados para manipular a informação no vetor.
- e) Estrutura de decisão com teste lógico entre os índices utilizados para manipular a informação no vetor.

Seção 4.4

As matrizes como estruturas de dados

Diálogo aberto

Olá, aluno, bem-vindo à última aula desta unidade. Vamos trabalhar com matrizes e para isso você já deve estar adaptado às estruturas que temos estudado, como os vetores e as demais estruturas que vimos até o momento, que são de extrema importância para a compreensão desta aula.

Como estamos aproximando a teoria da prática profissional através da elaboração dos algoritmos que representam as funcionalidades do aplicativo para os comerciantes do Litoral Sul, partimos sempre da premissa de que as situações apresentadas como exemplos são solicitações de funcionalidades do aplicativo e estas devem ser implementadas de acordo com as estruturas que estamos estudando.

Nesse contexto, para que o aplicativo possa colaborar com a captação de clientes e fidelização deles, vamos continuar a elaboração dos nossos algoritmos, nesta aula, com o foco em desenvolver uma matriz que tem por função armazenar, por cliente, as quantidades de cupons adquiridos através do aplicativo, porém, nosso algoritmo se limitará a uma implementação utilizando uma matriz quadrada 4x4. Com isso, será proposto um algoritmo em pseudocódigo no qual você poderá desenvolver uma solução também em linguagem de programação C.

O objetivo de aprendizagem desta unidade é apresentar como se faz para utilizar a estrutura de dados conhecida como matriz, desde a definição até as possibilidades de aplicações e operações em que é possível implementar essas estruturas. Sendo assim, conseguimos desenvolver, com o apoio desses conceitos, os exercícios que são propostos na unidade, em cada uma das aulas, principalmente desenvolvendo a visão analítica necessária para saber quais são os problemas em que é possível aplicar esses conceitos, melhorar algoritmos e os programas existentes, além de criar novos algoritmos que proporcionem a eficiência desejada em processamento e alocação dos dados em memória, a partir das estruturas de dados definidas e das operações que podem ser estabelecidas entre elas.

Com isso, fica a recomendação de estudos de acordo com as leituras do material, desenvolvimento das atividades de pré-aula, as atividades de aprofundamento e

a prática dos exercícios propostos. Além disso, os *links* sugeridos podem auxiliar no processo de ensino-aprendizagem que se propõe.

Desde já, bons estudos e práticas a você!

Não pode faltar

Estudamos nas aulas anteriores que um vetor é uma variável composta por n variáveis simples e precisam ser do mesmo tipo de dado. Na mesma proporção, temos as matrizes que são variáveis compostas, porém, de n vetores. A matriz também pode aceitar em seus elementos apenas um tipo de dado e pode ser acessada por mais de um índice. Mas será que podemos utilizar mais de uma matriz em um algoritmo? A resposta é sim. A fim de resolver um determinado problema, é possível que um mesmo algoritmo contenha mais de uma matriz para que possa atender ao objetivo proposto.

Uma matriz, também conhecida como “Variável Composta Homogênea Multidimensional - VCHM”, determina as posições de memória que serão alocadas sob a identificação do nome da matriz. Por esse motivo, os elementos da matriz correspondem a endereços que só podem ser acessados com o uso de índices. As matrizes podem ser bidirecionais (linha e coluna) ou multidirecionais. Seguindo esse raciocínio, você já consegue responder o nome que se dá a uma matriz com a mesma quantidade de linhas e colunas? O nome desse tipo de matriz é “matriz quadrada”.

Para entender melhor as estruturas aqui trabalhadas, podemos resgatar o exemplo do cálculo das médias que temos utilizado. Vejamos a seguir como podemos evoluir com a complexidade do algoritmo e aumentando a sua eficiência quando temos mais de um dado a apresentar. Por exemplo, para cada aluno, precisamos solicitar e ler suas notas bimestrais e com isso calcular a média (ponderada, sendo primeiro bimestre com peso 4 e segundo bimestre com peso 6) e mostrar se o aluno está “Aprovado” ou “Reprovado”.

Diante do exemplo apresentado, quantas matrizes serão necessárias para desenvolver uma solução que atenda a essa solicitação?

Nesse caso, temos que considerar duas matrizes, pois o nome do aluno é de um tipo de dado diferente das notas, temos, respectivamente, os tipos de dados caractere e real. Então, de acordo com Piva Jr. et al. (2012), será preciso utilizar os apontadores para fazer referência do nome do aluno que pode estar em uma matriz chamada Vet_Nome_Aluno, com a sua respectiva nota em uma outra matriz chamada Mat_Notas. Com isso, serão necessários outros dois apontadores para indicar as informações trabalhadas em cada uma delas. Mas, e a busca da informação, como funciona?

Suponha que eu tenha uma matriz 10x3, ou seja, que contenha 10 linhas e três colunas. Essa matriz também pode ser considerada, portanto, uma matriz com três dimensões, sendo que temos, respectivamente: nota primeiro bimestre, nota segundo bimestre e média final.

Nesse sentido, vamos agora desenvolver essa ideia passo a passo, visando ampliar a compreensão desde a declaração da matriz até a manipulação dos dados. Observe o exemplo em VisuAlg a seguir:



Exemplificando

Algoritmo "MediaFinal_Alunos"

```
// Função: Le o Nome e as Notas Bimestrais 1 e 2 de 10
alunos.
// Calcula a média final, e determina a situação de cada aluno
// (Aprovado, Em Exame ou Reprovado), e estabelece uma relação
// entre sua média final e a média da turma.
```

Var

```
Vet_Nome_Aluno : Vetor [1..10] de Caractere
Mat_Notas : Vetor [1..10,1..3] de Real
// Coluna 1 armazena a Nota do Bimestre 1
// Coluna 2 armazena a Nota do Bimestre 2
// Coluna 3 armazena a Média Final calculada
Contador_Alunos : Inteiro
Acum_Media, MediaTurma : Real
inicio
  Para Contador_Alunos de 1 ate 10 faca
    Escreval ("Informe Nome do Aluno(a) - ",Contador_Alunos," :")
    Leia(Vet_Nome_Aluno[Contador_Alunos])
    Escreval ("Informe sua Nota Bimestral 1 :")
    Leia(Mat_Notas[Contador_Alunos,1])
    Escreval ("Informe sua Nota Bimestral 2 :")
    Leia(Mat_Notas[Contador_Alunos,2])
    Mat_Notas[Contador_Alunos,3] := Mat_Notas[Contador_Alunos,1]*0.4
```

```

+   Mat_Notas[Contador_Alunos,2]*0.6
Acum_Media <= Acum_Media + Mat_Notas[Contador_Alunos,3]
Se (Mat_Notas[Contador_Alunos,3] >= 7.0) entao
    Escreval(Vet_Nome_Aluno[Contador_Alunos],", sua Média Final é ",
    Mat_Notas[Contador_Alunos,3]:2:2, " - Você está APROVADO(A).")
senao
    Se (Mat_Notas[Contador_Alunos,3] >= 5.0) entao
        Escreval(Vet_Nome_Aluno[Contador_Alunos],", sua Média
        Final é ",
        Mat_Notas[Contador_Alunos,3]:2:2, " - Você está EM
        EXAME.")
senao
    Escreval(Vet_Nome_Aluno[Contador_Alunos],", sua Média
    Final é ",
    Mat_Notas[Contador_Alunos,3]:2:2, " - Você está
    REPROVADO(A).")
FimSe
FimSe
    Escreval // saltar uma linha
FimPara
MediaTurma <= Acum_Media / Contador_Alunos
LimpaTela
Escreval("Média da turma: ",MediaTurma:2:1)
Escreval
Para Contador_Alunos de 1 ate 10 faca
    Se (Mat_Notas[Contador_Alunos,3] > MediaTurma) entao
        Escreval(Vet_Nome_Aluno[Contador_Alunos]," - Média: ",Mat_
        Notas[Contador_Alunos,3]:2:1, " - Bom Aluno(a).")
senao
    Se (Mat_Notas[Contador_Alunos,3] < MediaTurma) entao
        Escreval(Vet_Nome_Aluno[Contador_Alunos], " - Média: ",Mat_
        Notas[Contador_Alunos,3]:2:1, " - Aluno(a) com baixo desempenho")
senao

```

```

        Escreval(Mat_Notas[Contador_Alunos,3], " - Média: ", Mat_
        Notas[Contador_Alunos,3]:2:1, " - Aluno(a)
        Médio(a).")
    FimSe
FimSe
FimPara
Fimalgoritmo

```

Fonte: Piva Jr. et al. (2012, p. 299-301)



Pesquise mais

Assista ao vídeo que ensina como encontrar a diagonal principal de uma matriz, linguagem de programação C, disponível no canal *Códigos Eficientes*. Disponível em: <<https://www.youtube.com/watch?v=RZXNqc2dxdY>>. Acesso em: 27 jul. 2015.

Podemos elencar no algoritmo que utilizamos como exemplo os seguintes pontos de atenção quanto à matriz, que você deve observar em todos os algoritmos que vir a desenvolver. São eles:

a. Declaração: a diferença da declaração de uma matriz para um vetor consiste na dimensão da variável. Por exemplo:

- **Vet_Nome_Aluno: Vetor [1..10] de Caractere**.....aqui temos um vetor do tipo de dado caractere, com 10 elementos.
- **Mat_Notas: Vetor [1..10,1..3] de Real**.....aqui temos uma matriz do tipo de dado real, com 10 linhas e três colunas.

b. Sintaxe: a sintaxe de definição de uma matriz é representada da seguinte forma:

- **<identificador do vetor>: VETOR [Li1..Lf1, LiN..LfN] de <tipo de dado>**

O identificador do vetor é o nome que atribuímos a ele. A palavra "VETOR" é a instrução que determina a declaração tanto de um vetor quanto de uma matriz, seguido imediatamente da definição de dimensão da matriz, sendo que é preciso delimitar a quantidade de linhas e de

colunas, respectivamente. Por fim, a definição do tipo de dado da matriz é estabelecida.

Quando trabalhamos com o conceito conhecido como instância, muito utilizado em linguagens de programação, podemos criar um tipo de dado que seja do tipo matriz. Observe o exemplo (PIVA JR. et al., 2012):

- Var

Mat_Exemplo: Mat4x2;

Mas, além de variáveis, tipo de dado, o conceito também se aplica para que uma matriz seja declarada e compreendida de acordo com a sintaxe, como uma constante. Nesse caso, há a atribuição dos valores dos elementos da matriz em sua declaração. Observe o exemplo:

- Const

Mat_Exemplo2: Array [1..2, 1..5] de real \leftarrow ((1.10, 1.5, 1.3, 1.2, 1.1) , (2.5, 2.4, 2.3, 2.2, 2.1));

indLinha, indCol: inteiro

Para uma matriz com duas linhas e cinco colunas, temos a atribuição dos valores da primeira linha e da segunda, respectivamente, sendo que o número antes do sinal “.” (ponto) refere-se diretamente à coluna e o que vem depois do sinal é o valor atribuído ao elemento.

c. Manipulação de dados na matriz: os índices devem ser declarados para que se possa apontar os elementos de acordo com as suas respectivas posições na matriz. Ex.: indLinha, indCol: inteiro.

d. Atribuição: um elemento de uma matriz pode ser acessado de várias maneiras, vamos considerar o exemplo de Mat_Notas_Aluno para, em VisuAlg:

- Atribuir valor:

Vet_Notas_Aluno[1,3]: = 10

Atribuição realizada para o elemento que está na linha 1, coluna três com o valor para nota “10”.

- Atribuição inserida pelo usuário (teclado):

Leia(Vet_Notas_Aluno[1,3])

Utilizamos o comando de entrada de dados “Leia”; indicamos a respectiva posição no vetor “[1,3]” e, com isso, a respectiva atribuição por inserção do dado.

- Operações com os elementos da matriz, como atribuir um peso para a nota lançada:

`Vet_Notas_Aluno[1,3] := Vet_Notas_Aluno[1,3] x 0,6.`

- Atribuição de valor para a posição seguinte ao elemento apontado pelo índice:

`Vet_Notas_Aluno[indLinha, indCol+1] := 7,5.`

e. Consulta de valores: as consultas podem ser realizadas da seguinte forma:

- Escreva ("`Vet_Notas_Aluno[indLinha,indCol]`")

Agora que você já conheceu as formas de declarar, manipular, atribuir valores e consultá-los, podemos praticar um pouco mais. A seguir está um exemplo que transpõe a versão que fizemos acima no VisulaG para a linguagem de programação C.



Faça você mesmo

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    char Vet_Nome_Aluno[10][30];
    float Mat_Notas[10][3];
    // Mat_Notas - Coluna 0 = Nota do Bimestre 1
    // 1 = Nota do Bimestre 2
    // 2 = Média Final calculada
    int Contador_Alunos;
    float Acum_Media=0, MediaTurma;

    for (Contador_Alunos=0; Contador_Alunos < 10; Contador_Alunos++)
    {
        system("cls");
        printf("Informe Nome do Aluno(a) - %d :", Contador_Alunos+1);
        fflush(stdin); gets(Vet_Nome_Aluno[Contador_Alunos]);
        printf("\nInforme sua Nota Bimestral 1 : ");
        scanf("%f", &Mat_Notas[Contador_Alunos][0]);
        printf("\nInforme sua Nota Bimestral 2 : ");
        scanf("%f", &Mat_Notas[Contador_Alunos][1]);
```

```

        Mat_Notas[Contador_Alunos][2] = Mat_Notas[Contador_Alunos]
        [0] * 0.4 + Mat_Notas[Contador_Alunos] [1] * 0.6;
        Acum_Media = Acum_Media + Mat_Notas[Contador_Alunos][2];
    if (Mat_Notas[Contador_Alunos][2] >= 7.0)
        Vet_Nome_Aluno[Contador_Alunos]);
    else
        if (Mat_Notas[Contador_Alunos][2] >= 5.0)
            printf("\n%s Voce esta EM EXAME.", Vet_Nome_Aluno[Contador_
            Alunos]);
        else
            printf("\n%s Voce esta REPROVADO (A).", Vet_Nome_
            Aluno[Contador_Alunos]);
            _sleep(2000); // Para execuão por 2 segundos
    }
    MediaTurma = Acum_Media / Contador_Alunos;
    system("cls");
    printf("\n\nMedia da turma: %.1f ",MediaTurma);
    for (Contador_Alunos=0; Contador_Alunos < 10; Contador_Alunos++)
    {
        if (Mat_Notas[Contador_Alunos][2] > MediaTurma)
            printf("\n\n%s - Media: %.1f – Bom Aluno(a).",
                Vet_Nome_Aluno[Contador_Alunos], Mat_Notas[Contador_Alunos][2]);
        else if (Mat_Notas[Contador_Alunos][2] < MediaTurma)
        {
            printf("\n\n%s - Media: %.1f - Aluno(a) com baixo desempenho",
                Vet_Nome_Aluno[Contador_Alunos],
                Mat_Notas[Contador_Alunos][2]);
        }
        else
        {
            printf("\n\n%s - Media: %.1f - Aluno(a)
            Medio(a)", Vet_Nome_Aluno[Contador_Alunos],
            Mat_Notas[Contador_Alunos][2]);
        }
    }
    printf("\n>>>> ");
    system("pause");
    return 0;
}

```

Fonte: Piva Jr. et al. (2012, p. 308-309)



Reflita

Portanto, ao contrário do vetor que precisa somente de um apontador ou

índice para acessar um elemento contido nele, uma matriz bidimensional precisa de dois índices, um que aponte para a linha e outro para a coluna; na interseção da linha com a coluna tem-se o elemento de dado cujo conteúdo se quer acessar (PIVA JR. et al., 2010, p. 286).



Assimile

Com relação à ordenação de elementos de uma matriz de duas dimensões, o processo é o mesmo utilizado para ordenar matrizes de uma dimensão. Se você sabe fazer a ordenação de um estilo de matriz, sabe fazer a ordenação de qualquer estilo, seja ela da dimensão que for (MANZANO; OLIVEIRA, 2012, p. 143).

SEM MEDO DE ERRAR

Podemos propor um algoritmo para ler as quantidades de cupons que o usuário adquiriu, mas, a título de regra de negócio e, em contrapartida, de conhecimento de manipulação dos dados em matrizes, você poderá implementar também um complemento em que há a troca dos elementos de uma matriz para a oposta em outra matriz. Além disso, o algoritmo deverá apresentar a mensagem de acordo com a soma dos elementos da nova matriz, se o cliente ganhou mais um cupom ou se precisa acumular mais pontos. Siga em frente e implemente, além dessa, outras formas de resolver e melhorar as operações solicitadas!

```
algoritmo "Ganhe cupons"
// Função : xxxx
// Autor : xxxx
// Data : xxx
// Seção de Declarações
var
mat1: vetor[1..4,1..4] de inteiro
mat2: vetor[1..4,1..4] de inteiro
x, y: inteiro
soma, soma1, soma2: inteiro
inicio
```

```

// Seção de Comandos
// Recebimento dos valores da matriz
para x de 1 ate 4 faca
    para y de 1 ate 4 faca
        escreva("Digite o valor",x,"-",y,": ")
        leia(mat1[x,y])
    fimpara
fimpara
// Troca entre as posições entre duas matrizes
para x de 1 ate 4 faca
    mat2[1,x] <- mat1[4,x]
    mat2[4,x] <- mat1[1,x]
fimpara
para x de 1 ate 4 faca
    mat2[2,x] <- mat1[2,x]
    mat2[3,x] <- mat1[3,x]
fimpara
// Impressão dos valores das matrizes
limpatela
escreval("Matriz original: ")
para x de 1 ate 4 faca
    escreval(mat1[x,1],mat1[x,2],mat1[x,3],mat1[x,4])
fimpara
escreval("Matriz modificada: ")
para x de 1 ate 4 faca
    escreval(mat2[x,1],mat2[x,2],mat2[x,3],mat2[x,4])
fimpara
escreval("")
// Soma entre os termos das matrizes
para x de 1 ate 4 faca
    para y de 1 ate 4 faca
        soma1 <- soma1 + mat1[x,y]
    fimpara
fimpara
para x de 1 ate 4 faca

```

```

para y de 1 ate 4 faca
    soma2 <- soma2 + mat2[x,y]
fimpara
fimpara
// Resultado final
escreval("")
escreval("Soma da matriz 1 + matriz 2 = ",soma1+soma2)
se (soma > 10) então
    Escreval (" Parabéns, você ganhou mais um convite. Escolha o seu!")
senão
    Escreval("Acumule mais pontos! ",soma1-soma2)
fimse
fimalgoritmo

```



Atenção!

O link <<https://www.youtube.com/watch?v=gmtZSoyy0UI>> (acesso em: 27 jul. 2015) contém um vídeo que demonstra o desenvolvimento de um algoritmo no VisuAlg utilizando a estrutura de dados matriz.



Lembre-se

Quando o número de linhas de uma matriz é igual ao número de colunas, a matriz é dita matriz quadrada. Neste caso, os elementos de índices iguais constituem a diagonal principal (EVARISTO, 2001, p. 107).

Avançando na prática

| Pratique mais | |
|---|--|
| <p>Instrução</p> <p>Desafiamos você a praticar o que aprendeu transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois as compare com as de seus colegas.</p> | |
| Os vetores como estruturas de dados II | |
| <p>1. Competência de Fundamentos de Área</p> | <p>Conhecer os princípios e conceitos que envolvem o aprendizado em construção de algoritmos e programação e a sua importância para o universo do desenvolvimento de sistemas.</p> |

| | |
|------------------------------|---|
| 2. Objetivos de aprendizagem | Saber e conhecer quais são as operações em que se aplicam as estruturas de dados em matrizes. |
| 3. Conteúdos relacionados | Os vetores como estruturas de dados II. |
| 4. Descrição da SP | Já que estamos encerrando nossas atividades nesta aula, vamos praticar alguns exemplos. Suponha que você tenha de elaborar um algoritmo para receber e exibir os elementos de uma matriz 3x3. Considere o exercício a seguir para desenvolver o seu próprio algoritmo que exiba a diagonal principal de uma matriz quadrada. |
| 5. Resolução da SP | <p>algoritmo "Matriz 3x3 Somar Diagonal"</p> <p>// Função Faça um algoritmo para ler uma matriz 3X3 real e imprimir</p> <p>// a soma dos elementos da Diagonal principal. Generaliza para uma matriz NXN;</p> <p>// Seção de Declarações</p> <p>var</p> <p>matrizA:vetor[1..3,1..3] de real</p> <p>somaDiag1:real</p> <p>ij:inteiro</p> <p>inicio</p> <p>para i de 1 ate 3 faca</p> <p>para j de 1 ate 3 faca</p> <p>escreva("Digite os numeros: [i, " + ", j, "] ")</p> <p>leia(matrizA[i,j])</p> <p>fimpara</p> <p>fimpara</p> <p>para i de 1 ate 3 faca</p> <p>para j de 1 ate 3 faca</p> <p>escreva(matrizA[i,j])</p> <p>fimpara</p> <p>escreval("")</p> <p>fimpara</p> <p>para i de 1 ate 3 faca</p> <p>para j de 1 ate 3 faca</p> <p>somaDiag1<-(matrizA[1,1] + matrizA[2,2] + matrizA[3,3])</p> <p>fimpara</p> <p>fimpara</p> <p>escreval("-----")</p> <p>escreval("Soma da Diagonal 1 é = ", somaDiag1)</p> <p>escreval("-----")</p> <p>fimalgoritmo</p> <p>Fonte: <http://www.adaobraga.com.br/exercicios-de-matrizes-no-VisuAlg/comment-page-1/>. Acesso em: 21 set. 2015</p> |



Lembre-se

Se o número de linhas e o número de colunas de uma tabela não são conhecidos, pode-se usar um duplo *while* aninhado, definindo-se um *flag* para encerramento da digitação dos elementos de cada linha e um outro *flag* para encerramento da digitação da matriz (EVARISTO, 2001, p. 105).



Faça você mesmo

Pratique mais! A seguir, mais um exemplo que podemos aproveitar para praticar, também em linguagem de programação C, o desenvolvimento e a aplicação de matrizes.

```
/*
Programa que lê uma matriz de inteiros 3X3 e mostra
os valores que estão na diagonal principal
*/
main() {
int minhaMatriz[ 3 ][ 3 ], i, j;
for ( i = 0; i < 3; i++ ) {
    for ( j = 0; j < 3; j++ ) {
        printf( "[ %d ][ %d ]: ", i + 1, j + 1 );
        scanf( "%d", &minhaMatriz[ i ][ j ] );
    }
}
printf( "\nRESULTADO\n" );
for ( i = 0; i < 3; i++ ) {
    for ( j = 0; j < 3; j++ ) {
        if ( j == i ) {
            printf( "\n[ %d ][ %d ]: %d.", i, j, minhaMatriz[ i ][ j ] );
        }
    }
}
getch();
}
```

Fonte: <<http://codigoseficientes.blogspot.com.br/2014/01/linguagem-c-como-usar-vetores-de-uma.html>>. Acesso em: 21 set. 2015.

Faça valer a pena

1. Assinale a alternativa que demonstra corretamente a declaração de uma matriz.

- a) matrizA: vetor[1..3,1..3] de real.
- b) somaDiag1: real.
- c) matriz := vetor.

- d) `printf("[%d][%d]: ", i + 1, j + 1).`
- e) escreva("Digite os numeros: [", i, " + ", j, "] ").

2. Complete as lacunas da frase com as palavras de uma das alternativas a seguir.

"Quando o número de _____ de uma matriz é igual ao número de _____ a matriz é dita matriz _____. Neste caso, os elementos de índices iguais constituem a diagonal principal". (EVARISTO, 2001, p. 107).

- a) vetor/matriz/quadrada.
- b) colunas/matriz/quadrada.
- c) linhas/colunas/quadrada.
- d) declaração/vetor/quadrada.
- e) vetor/matriz/diagonal.

3. Está correto o que se afirma em:

I. Com relação à ordenação de elementos de uma matriz de duas dimensões, o processo é o mesmo utilizado para ordenar matrizes de uma dimensão.

II. Uma matriz bidimensional precisa de dois índices, um que aponte para a linha e outro para a coluna, na interseção da linha com a coluna tem-se o elemento de dado, cujo conteúdo se quer acessar.

III. Os elementos da matriz correspondem a endereços que só podem ser acessados com o uso de índices.

- a) II e III.
- b) I, II e III.
- c) Apenas I.
- d) I e II.
- e) I e III.

4. Explique como é a sintaxe que distingue uma variável de tipo de dado de constante quando estamos trabalhando com matriz.

- 5.** Cite as possíveis formas de se atribuir valores em matrizes.
- 6.** Assinale a alternativa que contém uma informação verdadeira sobre operações com matrizes.
- a) É aconselhável declarar apenas um índice para manipular os dados em uma matriz.
 - b) A atribuição dos valores em uma matriz é possível apenas através de inserção pelo usuário.
 - c) A atribuição de valores em uma matriz é possível apenas no momento da declaração.
 - d) Os índices devem ser declarados para que se possa apontar os elementos de acordo com as suas respectivas posições na matriz.
 - e) Não é necessário utilizar índices em matriz.
- 7.** Assinale a alternativa que contém uma das formas de se atribuir valores em matrizes.
- a) `Int minhaMatriz[3][3], i, j.`
 - b) `Leia(Vet_Notas_Aluno[1,3]).`
 - c) `Vet_Notas_Aluno[1,3]: = 10.`
 - d) `Escreva(matrizA[i,j]).`
 - e) `Para Mat_Vet[1,3] := faça.`

Referências

Básicas

PIVA JR., Dilermando et al. **Algoritmos e programação de computadores**. Rio de Janeiro: Elsevier, 2012.

SOUZA, Marco et al. **Algoritmos e lógica de programação**. 2. ed. São Paulo: Cengage Learning, 2013.

Complementares

EVARISTO, Jaime. **Aprendendo a programar**: programando na linguagem C para iniciantes. Rio de Janeiro: Book Express, 2001.

MANZANO, José A. N. G.; OLIVEIRA, Jayr Figueiredo de. **Algoritmos**: lógica para desenvolvimento de programação de computadores. 24. ed. São Paulo: Érica, 2010.

ZIVIANI, Nivio. **Projeto de algoritmos**: com implementações em Java e C++. São Paulo: Cengage Learning, 2011.

Anotações

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

Anotações

This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

Anotações

[illegible]