



Algoritmos e Lógica de Programação

Vetores e Matrizes

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.

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 4 Vetores e Matrizes | 5 |
| Seção 4.1 - Aplicações utilizando vetores e matrizes | 7 |
| Seção 4.2 - Operações sobre vetores e matrizes | 23 |
| Seção 4.3 - Os vetores como estruturas de dados | 37 |
| Seção 4.4 - As matrizes como estruturas de dados | 53 |

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.

Além dessa, temos 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 as 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, também, 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 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 estas 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 deste, 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 especificados os respectivos tipos de dados e variáveis utilizados nesse processo. No caso, o algoritmo proposto também incluía a opção de direcionar o usuário a 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 com essa foi aplicado o conceito em um algoritmo que visa coletar informações sobre a satisfação do usuário com relação ao uso do aplicativo. Através desta 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, que é desenvolvido com as estruturas de seleção e de repetição com teste no final "repita_até". Nesse sentido, você pode,

ainda, 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, para esta, o desenvolvimento de um algoritmo que calcule os juros sobre uma aplicação em fundo de renda fixa, 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 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 esta 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 esta 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 estes 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 4.1:

Figura 4.1 - 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 4.2 - 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 4.2 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 4.1 e 4.2, 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. Nesse sentido, 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 então 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 4.3 - 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 com o uso de vetores. Confira em: <<http://www.ime.usp.br/~macmulti/exercicios/vetores/>>.

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 se 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 := 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_Media_Final[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 := 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 este 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>>.

Assim como os vetores, unidimensionais, trabalhamos com uma estrutura de dado 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 ela com mais detalhes na seção 4.4 desta unidade, porém, é importante que você a conheça desde já.



Refleta

"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 := 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 := 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 e com o gabarito disponibilizado no apêndice do livro.

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 esta 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, também, em conseguir visualizar outras formas de se 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 4.1 – Dados dos alunosz

| 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 (2013)

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 (2013)

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.

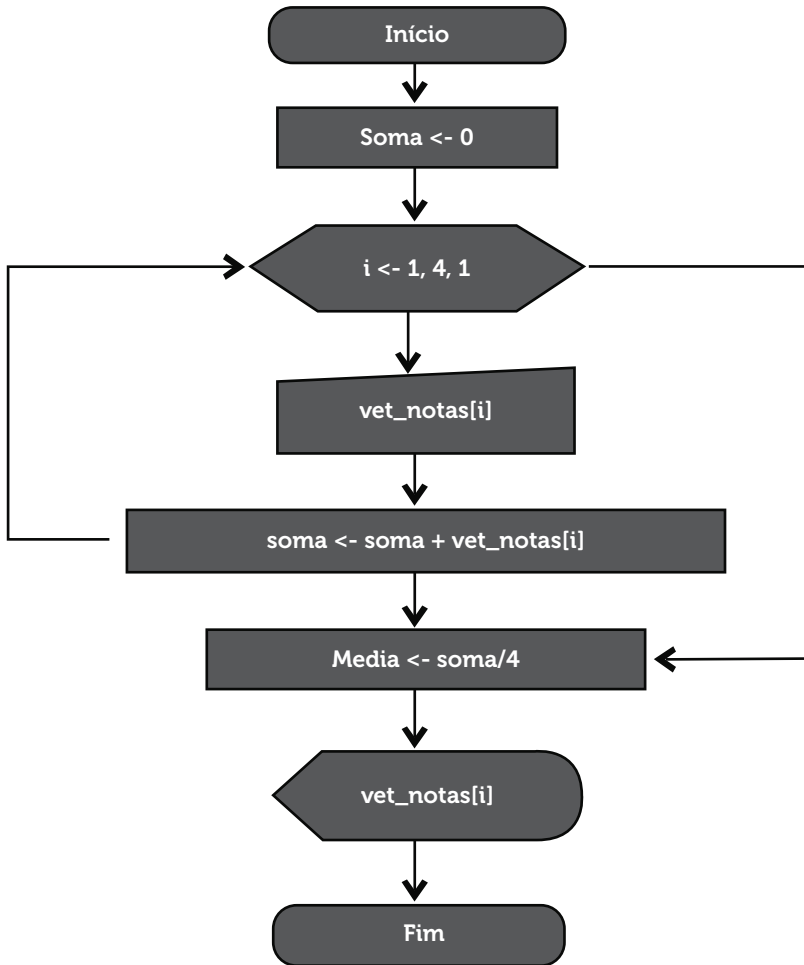


Refleta

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, 2013, 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 4.4 - Diagrama de blocos para leitura dos elementos de uma matriz do tipo vetor



Fonte: Manzano e Oliveira (2013, 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])

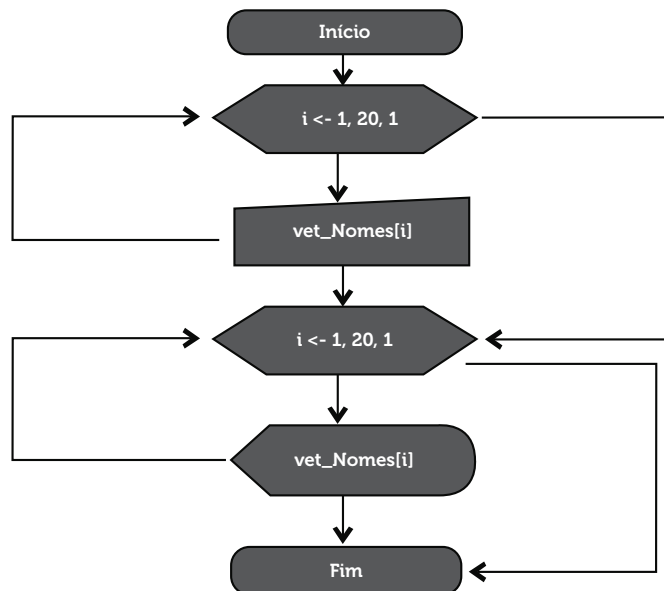
fimpara

fimalgoritmo

(Fonte: Adaptado de Manzano e Oliveira (2013, p. 119)).

Figura 4.5 - 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 exibê-los na tela de acordo com a ordem em que foram inseridos:



Fonte: Manzano e Oliveira (2013, 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 permitirá que um algoritmo armazene informações e as organiza. 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, 2013). 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, considerado 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, 2013).



Pesquise mais

Conheça todos os métodos indicados no link: <http://nicholasandre.com.br/sorting/>.

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 4.6 - 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 (2013)

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, 2013, p. 130):

Algoritmo "Pesquisa"

var

NOME: Vetor[1..10] de **cadeia**

I: **inteiro**

PESQ, RESP: **cadeia**

ACHA: **lógico**

início

para **I** **de** 1 **até** 10 **faça**

leia (NOME[I])

fimpara

//trecho de início de pesquisa sequencial

enquanto (RESP = "Sim") **faça**

escreva ("Insira o nome para pesquisa:")

leia (PESQ)

I <- 1

ACHA <- falso

enquanto (**I** <= 10) **e** (ACHA = falso) **faça**

se (PESQ = NOME[I]) **entao**

ACHA <- .verdadeiro.

senao

I <- **I** +1

fimse

fimenquanto

se (ACHA = verdadeiro) **então**

escreva (PESQ, "Foi localizado na posição:", **I**)

senao

escreva (PESQ, "Não foi localizado!")

fimse

escreva ("Deseja continuar? Responda (Sim/Não)")

leia (RESP)

fimenquanto

fimalgoritmo

Observe que a comparação será realizada até que todo o vetor seja percorrido ou, se encontre o valor correspondente que atenda a condição imposta na instrução de repetição enquanto.

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:= 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 do tipo vetor, 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, 2013, 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 e com o gabarito disponibilizado no apêndice do livro. | |
| 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 := 1 ate 20 faça Mat_Letra[IndLin,IndCol] := "L" FimPara FimPara Escreval ("Conteúdo da Matriz Letra") Escreval // Mostra o conteúdo da matriz Para IndLin := 1 ate 10 faça Para IndCol := 1 ate 20 faça Escreva(Mat_Letra[IndLin,IndCol]) // não salta linha FimPara Escreval // salta linh 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".

- a. () matriz/ simples/ bidirecional/ composta/ bidirecional
- b. () vetor/ simples/ bidirecional/ composta/ bidirecional
- c. () vetor/composta/bidirecional/simples/bidirecional
- d. () vetor/ composta/ unidirecional/ composta/ bidirecional
- e. () 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 a 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 esta aplicação pode ser transposta e aproximada da 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 e faça 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 esta lógica, as quantidades de convites vendidos para os jantares, de acordo com 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 implementá-las interagindo com a estrutura de dado em vetor, ou, como também 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 que segue 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".

```
inicio
// 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
escreval( 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 são 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 que Souza et al. (2011) 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.

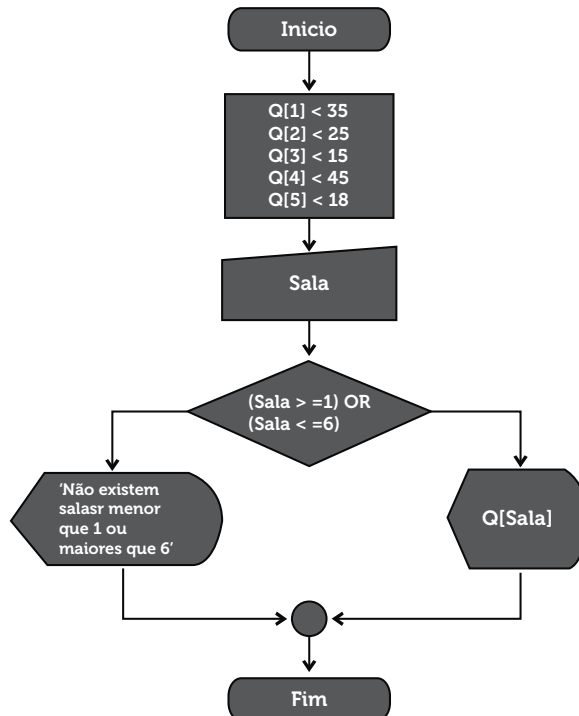
Este vetor deve armazenar a quantidade de alunos de cada sala. Não se esqueça de tentar elaborar e praticar o desenvolvimento das representações de acordo com cada um dos conceitos estudados. Os diagramas de blocos ou fluxogramas podem ajudar no desenvolvimento e implementação das estruturas que serão utilizadas para as ações que o algoritmo deve executar.

Observe como é a representação desta operação em diagrama de blocos:



Assimile

Figura 4.6 - 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, também, 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, então 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).

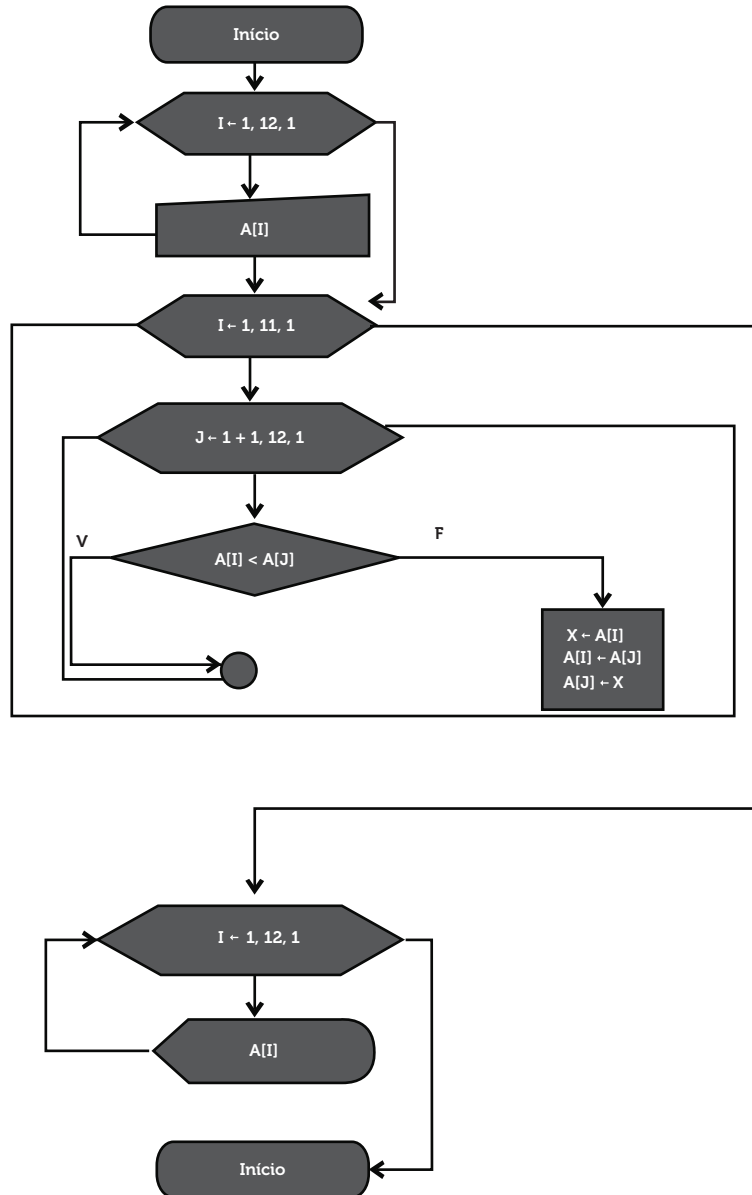
Agora, vamos trabalhar com mais um exemplo de aplicação de algoritmos com as estruturas de dados de vetor. Considere que uma empresa comercializa seus produtos pela internet e precisa iniciar um controle de vendas por cliente que deve ordenar as compras que ele realizou em um vetor. Veja no exemplo a seguir uma maneira de se 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 (2011) sugerem para realizar essa ordenação. Vamos lá! Confira o fluxograma, ou diagrama de blocos, desta operação.

Figura 4.7 - Fluxograma ordenação de elementos em um vetor



Fonte: Manzano e Oliveira (2011, 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
  escreva A[I]
fimpara
finalgoritmo

```

Fonte: Manzano e Oliveira (2011, 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)>

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 := 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 := 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 := 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 := 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 | |
|---|--|
| <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 e com o gabarito disponibilizado no apêndice do livro.</p> | |
| Os vetores como estruturas de dados I | |
| <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 vetores. |
| 3. Conteúdos relacionados | Os vetores como estruturas de dados I |
| 4. Descrição da SP | Resolução do exercício proposto no Sem Medo de Errar 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 ainda outras estruturas de dados em ambiente 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 representara 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: $\text{media} \leftarrow (\text{notas}[1] + \text{notas}[2] + \text{notas}[3]) / 3$.

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_ParInd_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 com a 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 destes 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,

ainda, a prática dos exercícios propostos. Além desses, 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 peso 6) e mostrar se o aluno está “Aprovado” ou “Reprovado”.

Diante do exemplo apresentado, quantas matrizes serão necessárias para desenvolver a 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 de 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 := 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 := 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
```

Finalgoritmo

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>>.

Podemos elencar no algoritmo que utilizamos como exemplo os seguintes pontos de atenção quanto a matriz, que servem para que você observe 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:

$\text{Vet_Notas_Aluno}[1,3] := \text{Vet_Notas_Aluno}[1,3] \times 0,6.$

- Atribuição de valor para a posição seguinte ao elemento apontado pelo índice:

$\text{Vet_Notas_Aluno}[\text{indLinha}, \text{indCol}+1] := 7,5.$

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)



Refleta

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., 2012, 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 deste, 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
finalgoritmo
```



Atenção!

O *link* <<https://www.youtube.com/watch?v=gmtZSoyy0UI>> contém um vídeo que demonstra como desenvolver um algoritmo no VisualG utilizando a estrutura de dado 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 | |
|--|---|
| 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 e com o gabarito disponibilizado no apêndice do livro. | |
| Os vetores como estruturas de dados II | |
| 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 matrizes. |
| 3. Conteúdos relacionados | Os vetores como estruturas de dados II |
| 4. Descrição da SP | Já que estamos encerrando nossas atividades aqui 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: Disponível em: <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 seguir para praticar também em linguagem de programação C, o desenvolvimento e 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: Disponível em: <<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 a 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. () `Escreva ("Vet_Notas_Aluno[indLinha,indCol]")`
- 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, 2011.

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