

Algoritmos e Programação I

Estruturas de Dados (vetores, matrizes e registros)

Prof. Fernando Maia da Mota
mota.fernandomaia@gmail.com
CPCX/UFMS

Estruturas de Dados

- ❖ Tipos de dados elementares são caracterizados pelo fato que seus valores são atômicos, isto é, não admitem decomposição.
 - ❖ Inteiro, real, carácter, lógico.
- ❖ Entretanto, se os valores de um tipo de dados admitem decomposição em valores mais simples, então o tipo de dados é dito complexo ou estruturado.
- ❖ A organização de cada componente e as relações entre eles constitui o que chamamos de **estrutura de dados**.

Vetores

❖ Considere o seguinte problema:

Calcule a média aritmética das notas de 5 alunos de uma disciplina e determine o número de alunos que tiveram nota superior à média calculada.

Vetores

- ❖ O cálculo da média aritmética das notas de 5 alunos de uma disciplina pode ser resolvido através de uma estrutura de repetição como a que segue:

...

soma \leftarrow 0

PARA i DE 1 ATÉ 5 FAÇA

leia nota

soma \leftarrow soma + nota

FIMPARA

media \leftarrow soma/5

...

Vetores

- ❖ Entretanto, se seguirmos com este trecho de algoritmo, como determinaremos quantos alunos obtiveram nota superior à média calculada? Isto porque não temos as notas de cada um dos 5 alunos depois que o trecho anterior for executado. Logo, devemos optar por outro caminho:

Vetores

ALGORITMO "calcula média e notas superiores"

// declaração de constantes e variáveis

VAR

soma, media, nota1, nota2, nota3 : inteiro
nota4, nota5, num : inteiro

INICIO

// leitura das notas

LEIA nota1, nota2, nota3, nota4, nota5

// cálculo a soma das notas

soma \leftarrow nota1 + nota2 + nota3 + nota4 + nota5

// cálculo da média

media \leftarrow soma/5

Vetores

```
// cálculo das notas superiores à média
num ← 0
SE nota1 > media ENTÃO
    num ← num + 1
FIMSE
SE nota2 > media ENTÃO
    num ← num + 1
FIMSE
SE nota3 > media ENTÃO
    num ← num + 1
FIMSE
SE nota4 > media ENTÃO
    num ← num + 1
FIMSE
SE nota5 > media ENTÃO
    num ← num + 1
FIMSE
// escreve o número de alunos com nota superior à média
ESCREVA "o número de alunos com nota superior à média é: ", num
```

FIMALGORITMO

Vetores

- ❖ Como podemos constatar, não fomos capazes de utilizar uma estrutura de repetição para calcular quantas notas eram superiores à média.
- ❖ No caso do problema anterior, esta redundância não chega a ser um “fardo”, mas se tivéssemos 100, 1000, ou mesmo 1000000 de notas, esta solução seria inviável, uma vez que teríamos de escrever, respectivamente, 100, 1000 ou 1000000 de estruturas condicionais semelhantes, uma para cada nota.
- ❖ Felizmente, para problemas como este, temos uma forma eficaz de solução, que utiliza uma estrutura de dados denominada **vetor**.

Vetores

- ❖ A estrutura de dados vetor é uma estrutura de dados linear utilizada para armazenar uma lista de valores do mesmo tipo.
- ❖ Um dado vetor é definido como tendo algum número fixo de células idênticas.
- ❖ Cada célula armazena um, e somente um, dos valores de dados do vetor.
- ❖ Cada uma das células de um vetor possui seu próprio endereço, ou índice, através do qual podemos referenciá-la.

Vetores

- ❖ Ao definirmos um vetor, estamos na verdade especificando a estrutura de dados de um novo tipo de dados, o tipo de dados vetor, pois este tipo, ao contrário dos tipos primitivos, não está “pronto para uso” e, portanto, deve ser definido explicitamente dentro do algoritmo.
- ❖ Nós podemos definir um vetor através da especificação de um identificador para um tipo vetor, suas dimensões e o tipo de dados dos valores que ele pode armazenar.

Vetores

**DEFINATIPO VETOR[*li..ls*] DE <tipo dos elementos>
<nome do tipo>**

- ❖ Onde:
 - ❖ **DEFINATIPO** e **VETOR** são palavras-chave;
 - ❖ Tipo dos elementos é o nome do tipo dos elementos do vetor;
 - ❖ Nome do tipo é um identificador para o tipo sendo definido;
 - ❖ *li* e *ls* são respectivamente os limites inferior e superior do vetor.

Vetores

- ❖ Por exemplo, para criarmos um vetor de 5 elementos inteiros chamado vetor notas, fazemos:

DEFINATIPO VETOR[1..5] DE INTEIRO vetor_notas

Vetores

- ❖ O número de elementos (células) de um vetor é dado por $ls - li + 1$.
- ❖ O índice do primeiro elemento (célula) é li , do segundo é $li + 1$, e assim por diante.
- ❖ Isto significa que dois vetores a e b com valores li e ls sejam 1 e 5 e 7 e 11, respectivamente, possuem o mesmo número de elementos: $5 = 5 - 1 + 1 = 11 - 7 + 1$.
- ❖ Entretanto, o primeiro elemento de a possui índice 1, enquanto o primeiro elemento de b possui índice 7.

Vetores

- ❖ Para declararmos uma variável de um tipo vetor, procedemos exatamente da mesma forma que para um tipo simples. Por exemplo, considere a criação de uma variável denominada `notas` do tipo vetor `notas`:

`notas : vetor_notas`

- ❖ Uma vez declarada a variável `notas`, podemos atribuir qualquer conjunto de 5 valores numéricos à variável. Entretanto, isto não é feito da forma mais natural, tal como:

`notas ← (-1, 0, 1, 33, -10)`

Vetores

- ❖ Mas sim individualmente, ou seja, um valor por vez a cada célula do vetor. Para tal usamos o nome da variável, o índice da célula e uma sintaxe própria.

notas[0] ← -1

- ❖ De forma geral, as células, e não a variável do tipo vetor como um todo, é que são utilizadas em qualquer operação envolvendo a variável:

- ❖ leia notas[1]

- ❖ escreva notas[1]

- ❖ notas[2] ← notas[1] + 1

Vetores

Problemas e soluções

Vetores

- ❖ Vamos iniciar esta seção resolvendo de forma eficaz o problema estabelecido anteriormente.

Calcule a média aritmética das notas de 5 alunos de uma disciplina e determine o número de alunos que tiveram nota superior à média calculada.

Vetores

```
// algoritmo para calcular a média de 5 notas e escrever
// quantos alunos obtiveram nota superior à média
ALGORITMO "calcula média e notas superiores"
```

VAR

```
// declaração de tipos
definatipo vetor[1..5] de real vetor_notas
```

```
// declaração de variáveis
notas : vetor_notas
soma, media : real
num, i : inteiro
```

INICIO

```
// lê e calcula a média das notas
soma ← 0
para i de 1 até 5 faça
    leia notas[i]
    soma ← soma + notas[i]
fimpara
```

Vetores

```
// cálculo da média
media ← soma/(5)
```

```
// cálculo das notas superiores à média
```

```
num ← 0
```

```
para i de 1 até 5 faça
```

```
    se notas[i] > media então
```

```
        num ← num+ 1
```

```
    fimse
```

```
fimpara
```

```
// escreve o número de alunos com nota superior à média
```

```
escreva "o número de alunos com nota superior à média é: ", num
```

FIMALGORITMO

Vetores

Escreva um algoritmo que declare uma variável de um tipo vetor de 10 elementos inteiros, leia 10 valores para esta variável e então escreva o maior e o menor valor do vetor e suas respectivas posições no vetor.

Vetores

```
// algoritmo para ler um vetor de 10 elementos e depois escrever
// o maior e o menor elemento e suas respectivas posições
algoritmo "encontra menor maior"
```

var

```
// cria um tipo vetor de números
definatipo vetor[1..10] de inteiro vetor10
// declaração de variáveis
n : vetor10
i, maior, menor, pmenor, pmaior : inteiro
```

inicio

```
// lê 10 números e armazena-os em um vetor
para i de 1 até 10 faça
    escreva "Entre com o elemento ", i, " do vetor: "
    leia n[i]
fimpara
```

Vetores

```
// determina menor e maior e suas posições
menor ← n[1]
maior ← n[1]
pmenor ← 1
pmaior ← 1
para i de 2 até 10 faça
    se menor > n[i] então
        menor ← n[i]
        pmenor ← i
    senão
        se maior < n[i] então
            maior ← n[i]
            pmaior ← i
    fimse
fimse

fimpara

// escreve o menor e o maior valor e suas posições
escreva "O menor valor é: ", menor
escreva "A posição do menor valor é: ", pmenor
escreva "O maior valor é: ", maior
escreva "A posição do maior valor é: ", pmaior
```

Vetores

Escreva um algoritmo que recebe um inteiro $0 < n \leq 100$ e um vetor de n números inteiros cuja primeira posição é 1 e inverte a ordem dos elementos do vetor sem usar outro vetor.

Vetores

// algoritmo para inverter a ordem dos elementos de um vetor sem utilizar vetor auxiliar

algoritmo "inverte"

var

// cria um tipo vetor de números

definatipo vetor[1..100] de inteiro vet_int

// declaração de variáveis

v : vet_int

i, temp : inteiro

inicio

// entrada de dados

leia n

para i de 1 até n faça

leia v[i]

fimpara

Vetores

// troca os elementos com seus simétricos

para i de 1 até n DIV 2 faça

temp \leftarrow v[i]

v[i] \leftarrow v[n - i + 1]

v[n - i + 1] \leftarrow temp

fimpara

// saída dos resultados

para i de 1 até n faça

escreva v[i]

fimpara

fimalgoritmo

Matrizes

- ❖ Os vetores que estudamos até então são todos unidimensionais. Mas, podemos declarar e manipular com mais de uma dimensão, os quais denominamos **matrizes**.
- ❖ Por exemplo, podemos definir uma estrutura de dados matriz 4 por 5 (*uma tabela com 4 linhas e 5 colunas*), denominada **tabela**, escrevendo as seguintes linhas.

definatipo vetor[1..4] de inteiro coluna
definatipo vetor[1..5] de coluna tabela

Matrizes

- ❖ Neste exemplo, *coluna* é um tipo vetor de números, isto é, um tipo cuja estrutura de dados é um vetor unidimensional, como estamos acostumados a criar.
- ❖ Entretanto, *tabela* é um tipo vetor de coluna, ou seja, um tipo cuja estrutura de dados é um vetor bidimensional (*uma matriz*), pois cada um de seus elementos é do tipo vetor de números do tipo **coluna**!
- ❖ Uma forma alternativa para definir o tipo *tabela* acima (e preferida pelos desenvolvedores de algoritmo) é a seguinte:
defina tipo vetor[1..4,1..5] de inteiro *tabela*

Matrizes

- ❖ $Ls_1 - Li_1 + 1$, é o número de linhas da tabela.
- ❖ $Ls_2 - Li_2 + 1$, é o número de colunas da tabela.
- ❖ Uma vez definido o tipo tabela, podemos declarar uma variável deste tipo da mesma maneira que declaramos variáveis dos demais tipos.

mat : tabela

Matrizes

- ❖ A partir daí, podemos manipular a variável *mat* utilizando dois índices, em vez de apenas 1, para referenciar cada elemento desta matriz.
- ❖ O primeiro índice identifica a posição do elemento na primeira dimensão (linha) e o segundo identifica a posição do elemento na segunda dimensão (coluna).
- ❖ Suponha que desejemos atribuir o valor 0 a todos os elementos da matriz *mat*.

```
...  
para i de 1 até 4 faça  
    para j de 1 até 5 faça  
        mat[i, j] ← 0  
    fimpara  
fimpara  
..
```

Matrizes

Problemas e soluções

Matrizes

Escreva um algoritmo que declare uma variável de um tipo matriz de 4 por 5 elementos numéricos, leia valores para esta variável e escreva a soma dos elementos de cada linha da matriz, bem como a soma de todos os elementos.

Matrizes

algoritmo "soma por linha e de linhas de matriz"

var

// declaração de tipos

definatipo veter[1..4,1..5] de inteiro tabela

// declaração de variáveis

mat : tabela

i, j, somalin, somatot : inteiro

inicio

// leitura dos elementos da matriz

para i de 1 até 4 faça

para j de 1 até 5 faça

escreva "entre com o elemento ", i , " e ", j , "
matriz"

da

leia mat[i, j]

fimpara

fimpara

Matrizes

```
// soma elementos por linha e totaliza
somatot ← 0
para i de 1 até 4 faça
    // calcula a soma da linha i - LI_1 + 1
    somalin ← 0
    para j de 1 até 5 faça
        somalin ← somalin + mat[i, j]
    fimpara

    // exibe a soma da linha i - LI_1 + 1
    escreva "A soma dos elementos da linha ", i, ": ", somalin

    // acumula a soma das linhas para encontrar a soma total
    somatot ← somatot + somalin
fimpara

// exibe a soma total
escreva "A soma de todos os elementos é: ", somatot

fimalgoritmo
```

Matrizes

Dadas duas matrizes $A_{n \times m}$ e $B_{m \times p}$, com $n \leq 50$, $m \leq 50$ e $p \leq 50$. Obter a matriz matriz $C_{n \times p}$ onde $C = AB$.

Matrizes

algoritmo "produto de matrizes"

var

// definição de tipos

definatipo vetor[1..50,1..50] de inteiro matriz

// declaração de variáveis

A, B, C : matriz

i, j, k, n, m, p : inteiro

inicio

// entrada de dados

leia n, m, p

para i de 1 até n faça

para j de 1 até m faça

leia A[i, j]

fimpara

fimpara

para i de 1 até m faça

para j de 1 até p faça

leia B[i, j]

fimpara

fimpara

Matrizes

```
// Cálculo da matriz produto
para i de 1 até n faça
    para j de 1 até p faça
        C[i, j] ← 0
        para k de 1 até m faça
            C[i, j] ← A[i, k] * B[k, j] + C[i, j]
        fimpara
    fimpara
fimpara

// escrita da matriz C
para i de 1 até n faça
    para j de 1 até p faça
        escreva C[i, j]
    fimpara
fimpara

fimalgoritmo
```

Registros

- ❖ Um registro é uma estrutura de dados que agrupa dados de tipos distintos ou, mais raramente, do mesmo tipo.
- ❖ Um registro de dados é composto por um certo número de campos de dado, que são itens de dados individuais.
- ❖ Por exemplo, suponha que desejemos criar um algoritmo para manter um cadastro de empregados de uma dada empresa.

Registros

- ❖ Neste cadastro, temos os seguintes dados para cada empregado:
 - ❖ Nome do empregado.
 - ❖ CPF do empregado.
 - ❖ Salário do empregado.
 - ❖ Se o empregado possui ou não dependentes.
- ❖ Então, cada ficha deste cadastro pode ser representada por um registro.
- ❖ Neste caso, a natureza dos campos é bem diversificada, pois nome pode ser representado por uma cadeia, CPF e salário por valores numéricos e existência de dependentes por um valor lógico.

Registros

- ❖ A sintaxe para definição de um novo tipo registro é a seguinte:

definatipo registro

< campo1 > : < tipo do campo1 >

< campo2 > : < tipo do campo2 >

...

< campon > : < tipo do campon >

fimregistro <nome do tipo>

Registros

- ❖ Como exemplo, vamos criar um registro para representar uma ficha do cadastro de empregados dado como exemplo anteriormente:

definatipo registro
nome : cadeia
CPF : inteiro
salario : real
temdep : lógico
fimregistro regficha

Registros

- ❖ Uma vez que um tipo registro tenha sido definido podemos criar tantas variáveis daquele tipo quanto quisermos, assim como fazemos com qualquer outro tipo complexo visto até então:

ficha1, ficha2, ficha3 : regficha

- ❖ Cada uma dessas três variáveis é um registro do tipo *regficha* e, portanto, cada uma delas possui quatro campos de dado: *nome*, *CPF*, *salario* e *temdep*.

Registros

- ❖ Assim como variáveis de vetores e matrizes, variáveis registros são manipuladas através de suas partes constituintes. Uma variável registro é manipulada através de seus campos.
- ❖ Então, se desejarmos atribuir um valor a uma variável registro, temos de efetuar a atribuição de valores para seus campos constituintes, um de cada vez. Como exemplo, considere a seguinte atribuição do conjunto de valores “Beltrano de tal”, 123456789, 1800000 e falso à variável *ficha1*:

ficha1.nome ← “Beltrano de Tal”

ficha1.CPF ← 123456789

ficha1.salario ← 180.00

ficha1.temdep ← falso

Registros

- ❖ A partir deste exemplo, podemos verificar que o acesso a cada campo de uma variável é realizado através da escrita do nome da variável registro seguido de um ponto (.) que, por sua vez, é seguido pelo nome do campo.

Vetores de Registros

- ❖ Registros nos fornecem uma forma de agrupar dados de natureza distinta. Entretanto, criarmos uma única variável de um registro complexo não parece ser muito diferente de criarmos uma variável para cada campo do registro e tratá-las individualmente.
- ❖ A grande força dos registros reside no uso deles combinado com vetores. Por exemplo:

Um grupo de 100 empregados iria requerer um conjunto de 100 variáveis registros, o que pode ser conseguido através da criação de um vetor de 100 elementos do tipo registro em questão!

Vetores de Registros

- ❖ Um vetor de registros é criado da mesma forma que criamos vetores de qualquer dos tipos que aprendemos até então.
- ❖ Suponha, por exemplo, que desejemos criar um vetor de 100 elementos do tipo *regficha*. Isto é feito da seguinte forma:

definatipo registro

nome : cadeia

CPF : inteiro

salario : real

temdep : lógico

fimregistro regficha

definatipo vetor[1..100] de regficha vetcad

Vetores de Registros

- ❖ Agora, considere a declaração de uma variável do tipo *vetcad*:

cadastro : vetcad

- ❖ Para manipular um registro individual do vetor cadastro, utilizamos o nome da variável vetor e o índice do elemento correspondente ao registro que queremos acessar.

cadastro[1].nome ← "Sicrano de Tal"
cadastro[1].CPF ← 987654321
cadastro[1].salario ← 540.00
cadastro[1].temdep ← verdadeiro

Registro de Tipos Complexos

- ❖ Assim como combinamos vetores e registros para criar vetores de registro, podemos ter um registro de cujo um ou mais campos são de um outro tipo de registro ou vetores.
- ❖ Suponha, por exemplo, que queremos adicionar um campo ao nosso registro *regficha* para representar a conta bancária do empregado. Este campo pode ser um outro registro contendo os campos nome do banco, número da agência e número da conta bancária.
- ❖ Portanto, a nova definição de *regficha* seria:

Registro de Tipos Complexos

definatipo registro

banco : cadeia

agencia : inteiro

numcc : inteiro

fimregistro regbanco

definatipo registro

nome : cadeia

CPF : inteiro

salario : real

temdep : lógico

conta : regbanco

fimregistro regficha

Registro de Tipos Complexos

- ❖ O fato do campo *conta* de *regbanco* ser um outro registro faz com que a manipulação deste campo seja realizada, também, através de suas partes constituintes.
- ❖ Então, se criarmos uma variável *ficha* do tipo *regficha*, temos de manipular o campo *conta* de *ficha* como segue:

ficha : regficha

ficha.conta.banco ← “Banco de Praça”

ficha.conta.agencia ← 666

ficha.conta.numcc ← 4555

Registros

Problemas e soluções

Registros

- ❖ Suponha que você tenha sido contratado para construir um programa de cadastro de boletins, neste programa você deve ler os dados de n alunos, onde $0 > n \leq 50$ e imprimir a média final de cada aluno por disciplina e a média geral.
- ❖ Considere para o cadastro de aluno os seguintes campos: nome do aluno, idade, série.
- ❖ Para cada aluno a seguinte matriz de notas deve ser preenchida:

Disciplina	Nota 1	Nota 2	Nota 3
1			
2			
3			

Registros

// algoritmo para cadastrar e calcular boletim de aluno

algoritmo "Boletim"

var

// tipo matriz que armazena as notas dos alunos

definatype vetor[1..3, 1..3] de real vet_notas

definatype registro

nome : cadeia

idade : inteiro

serie : inteiro

matriz : vet_notas

fimregistro aluno

definatype vetor[1..50] de aluno vet_aluno

alunos : vet_aluno // vetor de registros contendo os dados dos alunos

i, j, n : inteiro

soma, somaDisc : real

Registros

inicio

LEIA n

//leitura dos alunos e notas

para i de 1 até n faça

LEIA alunos[i].nome, alunos[i].idade, alunos[i].serie

para j de 1 até 3 faça

leia alunos[i].matriz[j][1]

leia alunos[i].matriz[j][2]

leia alunos[i].matriz[j][3]

fimpara

fimpara

Registros

```
//impressão boletim
soma ← 0
para i de 1 até n faça
    ESCREVA alunos[i].nome

    para j de 1 até 3 faça
        somaDisc ← alunos[i].matriz[j][1] + alunos[i].
            matriz[j][2] + alunos[i].matriz[j][3]
        escreva "Disciplina ", i, " : ", (somaDisc/3)
        soma ← soma + somaDisc
    fimpara

    escreva "Nota Média Geral: ", (soma / 9)
fimpara
```

FIMALGORITMO

Referências

- ❖ **SIQUEIRA, Marcelo F. Algoritmos e Estrutura de Dados. Mato Grosso do Sul: CCET/CPCX - UFMS, 2007.**
- ❖ **Farrer, H. et. al. Algoritmos Estruturados. Editora Guanabara, 1989.**
- ❖ **Shackelford, R.L. Introduction to Computing and Algorithms. Addison- Wesley Longman, Inc, 1998.**