S028 – Estrutura de Dados

Professor MSc. Eng. Márcio Lemos

E-mail: marcio.lemos@senairs.org.br

http://lattes.cnpq.br/4769158065464009

FIERGS SENAI

Apresentação do professor Márcio

- Mestre em Tecnologia Computacional pela Universidade Federal de Santa Catarina UFSC.
- Pós-Graduado em: MBA - Gestão Estratégica em TI;
- Pós-Graduado em: Gestão de Pessoas;
- Pós-Graduado em: Docência no Ensino Superior;
- Formação Android: Information Technology (IT). Pontifícia Universidade Católica do Rio Grande do Sul/PUCRS.

Apresentação do professor Márcio

- Graduação em Bacharelado Engenharia de Software,
- Graduado em Processos Gerenciais,
- Graduado em Análise e Desenvolvimento de Sistemas.

Apresentação do professor Márcio

- Diretor de TI na Auditec.
- Professor de Graduação na FATEC/SENAI-RS.
- Profissional com mais de 18 anos de experiência, nas áreas de Educação, Gestão e TI, em empresas como: **Dana Albarus**, **General Motors, Bradesco, IFRS**, **SEG, CNEC Brasil.**

Unidade Curricular

S028 – Estrutura de Dados

Carga Horária: 70 horas-aula

AULA 1

Apresentação da disciplina, conteúdo, metodologia de ensino, critérios de avaliação, cronograma, material de apoio (livros, e-books/Biblioteca Virtual).

Introdução à estrutura de dados (conceitos e aplicações).

Revisão de vetores; enumerações; definições, aplicações e exemplos de encapsulamento, coesão e acoplamento.

Exercícios.

Metodologias de Ensino

(X) Aulas expositivas dialogadas

(X) Aulas em laboratórios

(X) Situações problema

(X) Estudo de texto

OBJETIVO GERAL

- Utilizar de forma otimizada as diversas estruturas de dados apresentadas, levando em consideração o problema a ser resolvido ou otimizado, e também o contexto no qual ocorre esse problema.

OBJETIVOS ESPECÍFICOS (CAPACIDADES)

– Consolidar e aprofundar os conceitos fundamentais do paradigma de programação orientada a objetos.

– Definir as noções de projeto de programas e estruturas de dados orientados a objetos.

OBJETIVOS ESPECÍFICOS (CAPACIDADES)

– Definir e diferenciar as estruturas de dados genéricas fundamentais, tais como filas, pilhas, listas, dicionários e conjuntos.

– Manipular estruturas de dados através do emprego de algoritmos.

OBJETIVOS ESPECÍFICOS (CAPACIDADES)

– Selecionar e construir estruturas de dados adequadas para aplicações especificas, bem como modelar estas aplicações utilizando a noção de orientação a objetos.

– Comparar estruturas de dados através da adequação ao problema.

OBJETIVOS ESPECÍFICOS (CAPACIDADES)

– Implementar ferramentas de Software utilizando uma linguagem orientada a objetos.

– Introduzir conceitos básicos de complexidade e análise de algoritmos.

EMENTA

- Tipos abstratos de dados: Listas; Pilhas; Filas e enumerações.
- Algoritmos de ordenação: Algoritmos de inserção; Seleção e organização.

EMENTA

- Noções de complexidade e recursividade: Avaliação de desempenho; analítica e notação O; calcular o tempo de execução com base no número de operações primitivas; recursão direta e indireta.

EMENTA

- Persistência de dados com arquivos: Fluxo de E/S (streams); arquivos: conceitos; arquivos texto x arquivos binários; arquivos de acesso sequencial x arquivos de acesso direto; remoção de registros; arquivos de índices e serialização.

EMENTA

- Tratamento de exceções: Abordagens para tratamento de erros e exceções; tratamento de exceções em linguagem de programação orientada a objetos.

EMENTA

- Encapsulamento, coesão e acoplamento Definições, aplicações e exemplos.

Qualidades Pessoais

- As qualidades pessoais são atitudes, que serão desenvolvidas e observadas durante os processos de ensino e de aprendizagem, integradas às competências específicas da unidade curricular.
- Nesta unidade curricular serão desenvolvidas e observadas as seguintes qualidades pessoais:

Qualidades Pessoais

- Zelo com máquinas, equipamentos, instrumentos e ferramentas.
- Liderança de equipes de trabalho.
- Capacidade de planejamento.
- Consciência de qualidade técnica e ambiental.
- Consciência quanto à segurança e higiene no trabalho.
- Iniciativa na busca de novas tecnologias.

Qualidades Pessoais

- Capacidade de comunicação interpessoal.
- Respeito aos princípios da ética profissional.
- Responsabilidade com o trabalho executado.
- Capacidade de autoavaliação.
- Capacidade de resolução de problemas.
- Empreendedorismo.
- Outras

Metodologias de Ensino

(X) Aulas expositivas dialogadas

(X) Aulas em laboratórios

(X) Situações problema

(X) Estudo de texto

Critérios de Avaliação

- A avaliação tem caráter integrador, uma vez que existe um crescimento gradativo na mobilização dos conhecimentos, habilidades e atitudes desenvolvidos. O resultado é expresso sob a forma de nota que varia de 0 (zero) a 10 (dez), com intervalos de 0,1 (um décimo).
- É exigido, no mínimo, a **média 6,0 (seis)** para fins de aprovação na unidade curricular.
- O resultado final do aproveitamento é expresso através da média aritmética dos graus G1 e G2, conforme equação a seguir.

Critérios de Avaliação

$$Nf = (G1 + G2)/2$$

Estão aprovados alunos com Nf ≥ 6,0.

É exigida a frequência mínima de 75% para aprovação.

Os demais devem fazer a avaliação de substituição de grau, sendo sua nota final definida pela equação a seguir.

Critérios de Avaliação

Nf = (Nota do maior grau + Nota do grau substituído)/2

Critérios de Avaliação: Substituição de Grau

- Somente será permitida a substituição de um dos graus quando a média for inferior a 6,0. Nos casos em que a nota do G1 e do G2 forem iguais, o grau a ser substituído é definido pelo aluno e informado para a secretaria de controle e registro acadêmico mediante preenchimento de requerimento, no prazo estabelecido no calendário acadêmico.
- A substituição de grau deve contemplar especificamente os conteúdos avaliados no grau que será substituído e deve ser realizada na data estabelecida no Calendário Acadêmico.

Frequência Mínima

É exigida a frequência mínima de 75% para aprovação.

Bibliografia Básica

- FORBELLONE, André Luiz; EBERSPÄCHER, Henri Frederico. Lógica de programação: a construção de algoritmos e estruturas de dados com aplicações em Python. 4.ed. São Paulo: Grupo A Educação, 2022. E-book.
- GUEDES, Sergio (org.). Lógica de programação algorítmica. São Paulo: Pearson Education do Brasil, 2014. (Coleção Bibliografia Universitária Pearson). E-book.
- PUGA, Sandra; RISSETTI, Gerson. Lógica de programação e estruturas de dados: com aplicações em Java. 3. ed. São Paulo: Pearson Education do Brasil, 2016. E-book.

Bibliografia Complementar

- ARAÚJO, Sandro de. Lógica de programação e algoritmos. Curitiba: Contentus, 2020. E-book.
- ASCENCIO, Ana Fernanda Gomes; ARAÚJO, Graziela Santos de. Estruturas de dados: algoritmos, análise da complexidade e implementações em JAVA e C/C++. São Paulo: Pearson Education do Brasil, 2010. E-book.
- BORIN, Vinicius Pozzobon. Estrutura de dados. Curitiba: Contentus, 2020. E-book.
- FÉLIX, Rafael (org.). Programação orientada a objetos. São Paulo: Pearson Education do Brasil, 2016. E-book.
- RANGEL, Pablo; CARVALHO JR., José Gomes de. Sistemas orientados a objetos: teoria e prática com UML e Java. Rio de Janeiro: Brasport, 2021. E-book.

Informações Gerais

- O cronograma de atividades desta unidade curricular representa uma estimativa de distribuição de conteúdos, podendo haver ajustes ao longo do período letivo.

- As atividades relacionadas à unidade curricular vão além dos horários de aula, exigindo estudos complementares por parte dos estudantes. Caso haja necessidade o estudante pode procurar os serviços atendimento ao discente, tais como: monitoria, nivelamento ou aulas de reforço.

Informações Gerais

- O Ambiente Virtual de Aprendizagem (AVA) é utilizado como ferramenta de auxílio às aulas presenciais, assim é recomendado que você acesse periodicamente o T2K.

Núcleo de Apoio ao Estudante - NAE

Telefone: (51) 3904-2600 Ramal: 8468

Sala: 410

Introdução à estrutura de dados (conceitos e aplicações).

- Para o profissional da área da computação, é importante conhecer a forma como os dados são:

Introdução à estrutura de dados (conceitos e aplicações).

- **Armazenados**,
- **Organizados** e
- **Manipulados**, **pois essa atividade** é **uma das principais desenvolvidas pelos sistemas computacionais**.

Introdução à estrutura de dados (conceitos e aplicações).

- Nesse sentido, a estrutura de dados é um elemento que vem, principalmente na **linguagem de programação C;**
- **JAVA, entre outras.**

Introdução à estrutura de dados (conceitos e aplicações).

- Para auxiliar na representação e na abstração de estruturas mais complexas, executando operações de **armazenamento** e **busca de dados na memória**, de maneira mais sofisticada e robusta.

Revisão de vetores;

- Os **vetores** e as **matrizes** são os exemplos mais comuns de estruturas de dados em **C** que são formadas por dados **homogêneos**, ou seja, são formadas por somente um tipo de dados.

Revisão de vetores;

- Para resolver a situação a seguir, considere a aplicação de dados homogêneos **(vetores e matrizes)** e ponteiros.

Desafio em conjunto. Revisão de vetores;

- Você está com a lista de preços de uma papelaria, que contém os seguintes **materiais** e preços normal e com **desconto:**

Desafio em conjunto. Revisão de vetores;

	PREÇO[0]	PREÇO[1]
MATERIAL[0]	15,00	12,50
MATERIAL[1]	13,00	7,50
MATERIAL[2]	100,00	97,00

Desafio em conjunto. Revisão de vetores;

- Você deverá elaborar um **código em linguagem C** para **apresentar um** vetor que contenha a média de preços por produto.

Desafio em conjunto. Revisão de vetores;

- O vetor deve ser apresentado de forma semelhante a este:

13,75	10,25	98,50

Desafio em conjunto. Revisão de vetores;

```
Insira o preco da posicao | 0|| 0|:15.00
Insira o preco da posicao | 0|| 1|:12.50
Insira o preco da posicao | 1|| 0|:13.00
Insira o preco da posicao | 1|| 1|:7.50
Insira o preco da posicao | 2|| 0|:100.00
Insira o preco da posicao | 2|| 1|:97.00

A tabela de precos contem os seguintes elementos:
|15.00|12.50|
|13.00|7.50|
|100.00|97.00|

A media de preco dos produtos e:
| 13.75 | 10.25 | 98.50 |

...Program finished with exit code 0
Press ENTER to exit console.
```


PROFISSIONAL:
VAI FICAR 1MIL$
CLIENTE:
DEIXA QUIETO, CONHEÇO
ALGUÉM Q FAZ MAIS BARATO

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define materiais 3
#define precos 2

main() { https://www.onlinegdb.com/online_c_compiler#tab-stdin
int posicao_material = 0, posicao_preco = 0;
float matriz[materiais][precos], vetor[materiais], valor_digitado = 0.0, media_preco = 0.0;

    for(posicao_material=0;posicao_material<materiais;posicao_material++){
        media_preco = 0.0;
            for(posicao_preco=0;posicao_preco<precos;posicao_preco++){
                printf("Insira o preco da posicao |%2d||%2d|:", posicao_material,posicao_preco);
                scanf("%f", &valor_digitado);
                 matriz[posicao_material][posicao_preco] = valor_digitado;
                 media_preco += matriz[posicao_material][posicao_preco];
            }
        vetor[posicao_material] = media_preco / precos;
    }

```

```
23      printf("\nA tabela de precos contem os seguintes elementos: \n");
24
25      for(posicao_material=0;posicao_material<materiais;posicao_material++){
26          for(posicao_preco=0;posicao_preco<precos;posicao_preco++){
27              printf("|%3.2f",matriz[posicao_material][posicao_preco]);
28          }
29          printf("|\n");
30      }
31
32      printf("\nA media de preco dos produtos e: \n| ");
33
34      for(posicao_material=0; posicao_material<materiais; posicao_material++)
35          printf("%3.2f | ",vetor[posicao_material]);
36  }
37
```

Desafio em conjunto. Revisão de vetores;

```
Insira o preco da posicao | 0|| 0|:15.00
Insira o preco da posicao | 0|| 1|:12.50
Insira o preco da posicao | 1|| 0|:13.00
Insira o preco da posicao | 1|| 1|:7.50
Insira o preco da posicao | 2|| 0|:100.00
Insira o preco da posicao | 2|| 1|:97.00

A tabela de precos contem os seguintes elementos:
|15.00|12.50|
|13.00|7.50|
|100.00|97.00|

A media de preco dos produtos e:
| 13.75 | 10.25 | 98.50 |

...Program finished with exit code 0
Press ENTER to exit console.
```

Revisão de vetores;

- As estruturas de dados normalmente envolvem a **identificação** e o **desenvolvimento de entidades** e operações úteis.

Revisão de vetores;

- Determinam o tipo de problema que pode ser solucionado por meio da sua utilização.

Revisão de vetores;

- Elas também determinam a **representação de entidades abstratas** e a implementação de operações abstratas que podem ser utilizadas com representações concretas.

O QUE É UMA ESTRUTURA DE DADOS?

SINTAXE DA DECLARAÇÃO:

```
struct nome_da_estrutura {
    tipo_do_campo1 nome_do_campo1;
    tipo_do_campo2 nome_do_campo2;
    tipo_do_campo3 nome_do_campo3;
    ...
};
```

Serve para auxiliar o armazenamento, no computador, de dados que são vistos na vida real e, por isso, precisam de abstração.

Agrupa e manipula dados dentro de uma entidade identificada por meio de um único nome de variável.

É formada por elementos, membros ou campos.

Envolve o armazenamento de dados organizados na memória de forma mais sofisticada do que utilizando variáveis simples.

Faz referência aos dados utilizando índices que servem para otimizar a localização de um registro quando é feita uma consulta que envolve vários dados.

Permite executar operações e fazer manipulação de dados, indicando como os dados são representados e a forma possível de manipulá-los.

FIERGS SENAI

A estrutura de dados

- A **área da Ciência da Computação** trata diretamente com o armazenamento, a organização, a manipulação e a utilização de dados e informações, por isso, é importante entender como isso acontece. Estudar estruturas de dados, principalmente na **linguagem C**, envolve (TENENBAUM; LANGSAM; AUGENSTEIN, 1995):

A estrutura de dados

- **Um dado** é algo normalmente quantificável, o qual não tem um significado relevante de maneira isolada.

A estrutura de dados

- O dado é considerado como o fundamento da informação. Conhecendo-se somente o dado, dificilmente é possível chegar a um entendimento sobre o assunto tratado**, tomar decisões** ou chegar a conclusões.

A estrutura de dados

- Já a **informação consiste na ordenação** e na organização dos dados, de maneira que se torne possível **compreender o seu significado**, ou entendê-la dentro de um contexto.


DATA-01
DATA-01
DATA-03

A estrutura de dados

- Em outras palavras, a **informação é o fruto do processamento dos dados**, depois que estes são **analisados**, interpretados de uma maneira pré-definida e qualificados.

A estrutura de dados

- O **nosso conhecimento** é oriundo da consolidação dos dados em forma de informação.

A estrutura de dados

- **Quando o assunto é estrutura de dados**, um índice é uma referência que se associa a uma chave, a qual serve para otimizar a localização de um registro, quando é feita uma consulta que envolve vários dados.

Dados heterogêneos e homogêneos

- Quando uma **estrutura de dados** armazena elementos de tipos diferentes, é dito que esta é uma estrutura de dados heterogênea.

Dados heterogêneos e homogêneos

- Normalmente, esse **tipo de dado é chamado de registro**, então, o registro é uma estrutura de dados que faz o agrupamento de dados de tipos diferentes entre si.

Dados heterogêneos e homogêneos

- Ele é formado por uma quantidade determinada de campos, os quais são itens individuais de dados e relacionados de forma lógica (LAUREANO, 2008).

Dados heterogêneos e homogêneos

- Em outras palavras, um registro é algo que agrupa elementos que **não são do mesmo tipo de dado**, mas que têm vínculo lógico.

Dados heterogêneos e homogêneos

- Os **registros consistem em conjuntos de posições de memória**, os quais são identificados pelo mesmo nome e são individualizados por meio de identificadores que se associam a cada conjunto de posições **(Figura 1).**

Figura 1. Representação gráfica de um registro.

Fonte: Laureano (2008, p. 41).

Dados heterogêneos e homogêneos

- Para referenciar um único campo dentro de um registro, é preciso utilizar essa sintaxe:
  - **nome_da_estrutura[índice].nome_do_campo;**

Dados heterogêneos e homogêneos

- Os **dados homogêneos** são representados pelos vetores, ou estruturas de dados unidimensionais, e **pelas matrizes**, que são estruturas de dados bidimensionais (LAUREANO, 2008).

Vetores

- Os vetores são estruturas de dados **unidimensionais lineares** e, por isso, precisam de um **único índice** para fazer o endereçamento e para percorrer toda a estrutura.

Vetores

- Um vetor é uma estrutura de dados utilizada para fazer o armazenamento de **uma lista de valores do mesmo tipo**, ou seja, em uma mesma variável, vários valores de mesmo tipo são armazenados.

Vetores

- Uma estrutura de dados do tipo vetor tem as seguintes características (DEITEL; DEITEL, 2011):

Vetores

- É alocado de forma estática, ou seja, no momento da sua declaração,
- deve-se conhecer o tamanho ou a quantidade de posições que ele terá.
- É uma estrutura de dados homogênea, formada por elementos com o mesmo tipo de dado.

Vetores

- Cada posição do vetor contém somente um valor.
- Os tamanhos dos valores de cada posição, por serem do mesmo tipo, são iguais.
- Faz a alocação dos dados de forma sequencial **(Figura 2).**

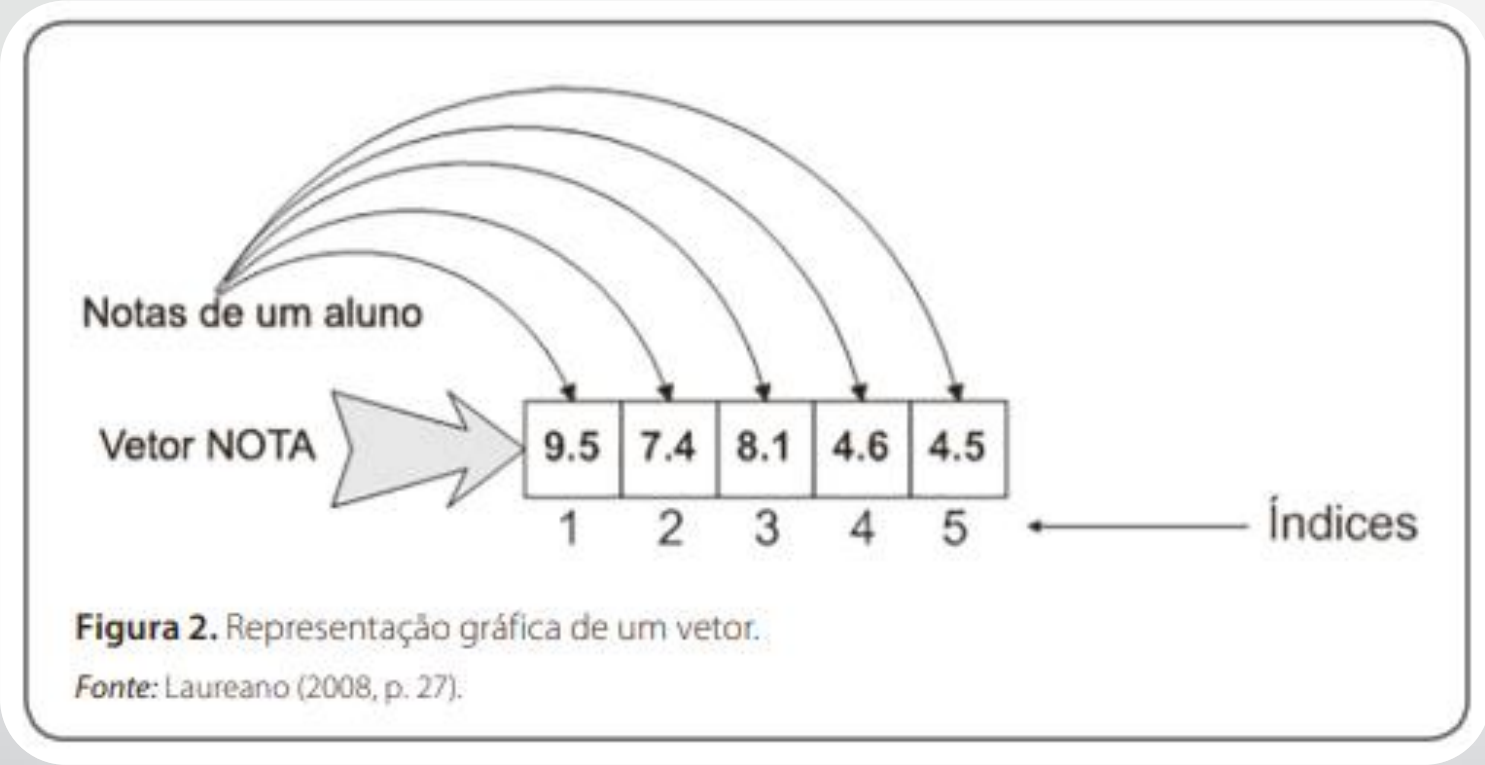


Figura 2. Representação gráfica de um vetor.

Fonte: Laureano (2008, p. 27).

Vetores

- **A sintaxe para a definição do vetor acima seria:**

  tipo_de_dado nome_do_vetor [quantidade_de_posicoes_do_vetor]

  Em C:

  - float nota[5];

Vetores

- **É importante lembrar que a primeira posição de um vetor** será sempre conhecida por **0** e nunca por 1.

```
nota[0] = 9.5; (Atenção)!!
nota[1] = 7.4;
nota[2] = 8.1;
nota[3] = 4.6;
nota[4] = 4.5;
```

Matrizes

- As **matrizes são estruturas de dados bidimensionais**, os quais precisam de dois índices para fazer o endereçamento e para percorrer toda a estrutura:

Matrizes

- Um que servirá para **referenciar a linha** e outro que servirá para referenciar a coluna da matriz (TENENBAUM; LANGSAM; AUGENSTEIN, 1995).

Matrizes

- Assim como os vetores, as matrizes têm algumas características:

Matrizes

- São alocadas de forma estática, ou seja, no momento da sua declaração, deve-se conhecer o tamanho que elas terão;
- São uma estrutura de dados homogênea, a qual é formada por elementos com o mesmo tipo de dados;

Matrizes

- Cada posição da matriz contém somente um valor;
- Os tamanhos dos valores de cada posição, por serem do mesmo tipo, são iguais;
- Fazem a alocação dos dados de forma sequencial **(Figura 3).**

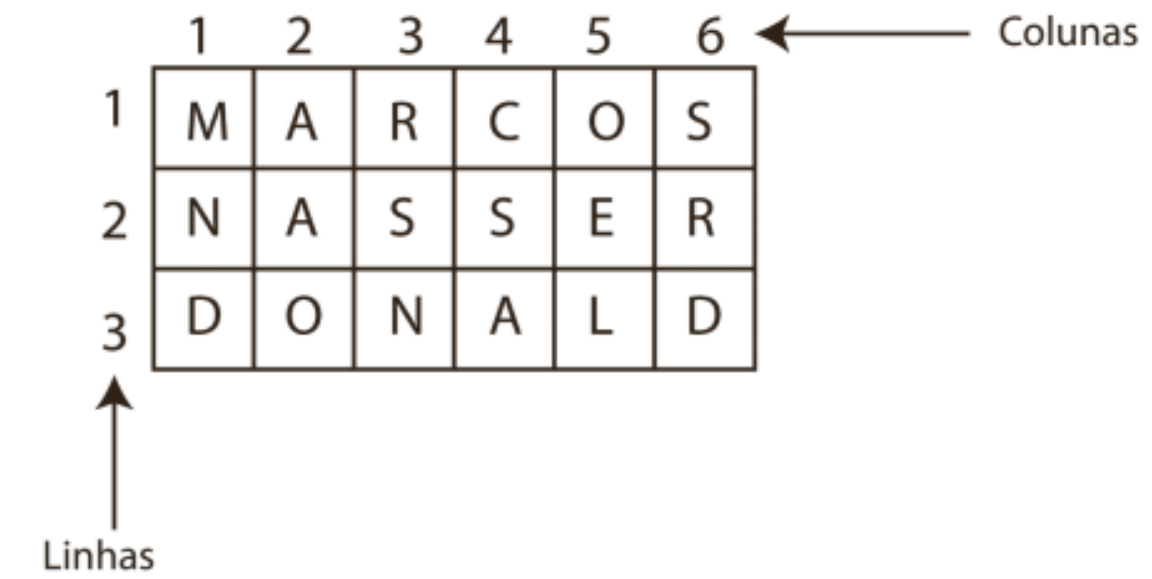


Figura 3. Representação gráfica de uma matriz.

Fonte: Laureano (2008, p. 31).

Matrizes

- Uma **matriz envolve dois** ou **mais vetores**, os quais são definidos por um conjunto de elementos, ou seja, cada dimensão da matriz é formada por um vetor.

Matrizes

- A **primeira dimensão de elementos** é considerada o primeiro vetor;
- A **segunda** é considerada o segundo vetor,
- A terceira dimensão de elementos é considerada o terceiro vetor, e assim por diante.

Matrizes

- A sintaxe para a definição da matriz acima seria:
  - **tipo_de_dado**

    nome_da_matriz [quantidade_de_linhas][quantidade_de_colunas]

  Em C:

  char letras[3][6]

Matrizes

- A alocação de valores para os elementos da matriz pode ser feita linha por linha ou coluna por coluna.
- É importante lembrar, da mesma forma que os vetores, que a primeira posição de cada dimensão da matriz será sempre conhecida por **0** **e** nunca por 1.

Matrizes

letras[0][0] = “M”; **(Atenção)**

letras[0][1] = “A”;

letras[0][2] = “R”;

letras[0][3] = “C”;

letras[0][4] = “O”;

letras[0][5] = “S”

Matrizes

```
letras[1][0] = "N";
letras[1][1] = "A";
letras[1][2] = "S";
letras[1][3] = "S";
letras[1][4] = "E";
letras[1][5] = "R";
```

Matrizes

letras[2][0] = “D”;

letras[2][1] = “O”;

letras[2][2] = “N”;

letras[2][3] = “A”;

letras[2][4] = “L”;

letras[2][5] = “D”

Aplicação de dados homogêneos (vetores e matrizes)

- Para que se possa compreender a aplicação de um vetor, pode-se criar um arquivo em **C** com o nome **vetor.cpp**, cujo código será o seguinte:

Aplicação de dados homogêneos (vetores e matrizes)

- Para que se possa compreender a aplicação de um vetor, pode-se criar um arquivo em **C** com o nome **vetor.cpp**, cujo código será o seguinte:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define elementos 10
main(){
    int posicao, vetor[elementos], maior = 0;
    for(posicao=0;posicao<elementos;posicao++) {
         vetor[posicao] = posicao;
        }
    printf("O vetor criado contem os seguintes elementos: \n\n|");
    for(posicao=0; posicao<elementos; posicao++)

        printf("%d | ",vetor[posicao]);
    printf("\n\nExibido ao contrario, os elementos do vetor sao: \n\n| ");
     for (posicao=9;posicao>=0;posicao--)
        printf("%d | ",vetor[posicao]);
```

```
17
18      printf ("\n\nOs elementos pares do vetor sao: \n\n| ");
19          for(posicao=0; posicao<elementos; posicao++)
20          if((vetor[posicao] % 2) ==0)
21           printf("%d | ",vetor[posicao]);
22
23      printf ("\n\nO maior elemento do vetor e: | ");
24          for (posicao=0; posicao<elementos; posicao++)
25           if (vetor[posicao] >maior)
26              maior = vetor[posicao];
27              printf("%d | ",maior);
28  }
29
```

```
O vetor criado contem os seguintes elementos:

|0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Exibido ao contrario, os elementos do vetor sao:

| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Os elementos pares do vetor sao:

| 0 | 2 | 4 | 6 | 8 |

O maior elemento do vetor e: | 9 |

...Program finished with exit code 0
Press ENTER to exit console.
```

Aplicação de dados homogêneos (vetores e matrizes)

- A execução do código acima exibirá uma tela semelhante à da Figura.

Aplicação de dados homogêneos (vetores e matrizes)

- Uma das maneiras mais fáceis de incluir elementos em um vetor é utilizando o laço for, o que é feito entre as linhas 11 e 15 do código acima.

Aplicação de dados homogêneos (vetores e matrizes)

- Depois disso, o código do programa exibe os valores contidos no vetor...

Aplicação de dados homogêneos (vetores e matrizes)

- Ainda utilizando o laço for, mas agora fazendo um teste com cada posição do vetor, são exibidos os elementos pares.


DELL
www.dell.com

Aplicação de dados homogêneos (vetores e matrizes)

- Para que se possa compreender a aplicação de uma matriz, pode-se criar um arquivo em **C** com o nome **matriz.cpp**, cujo código será o seguinte:

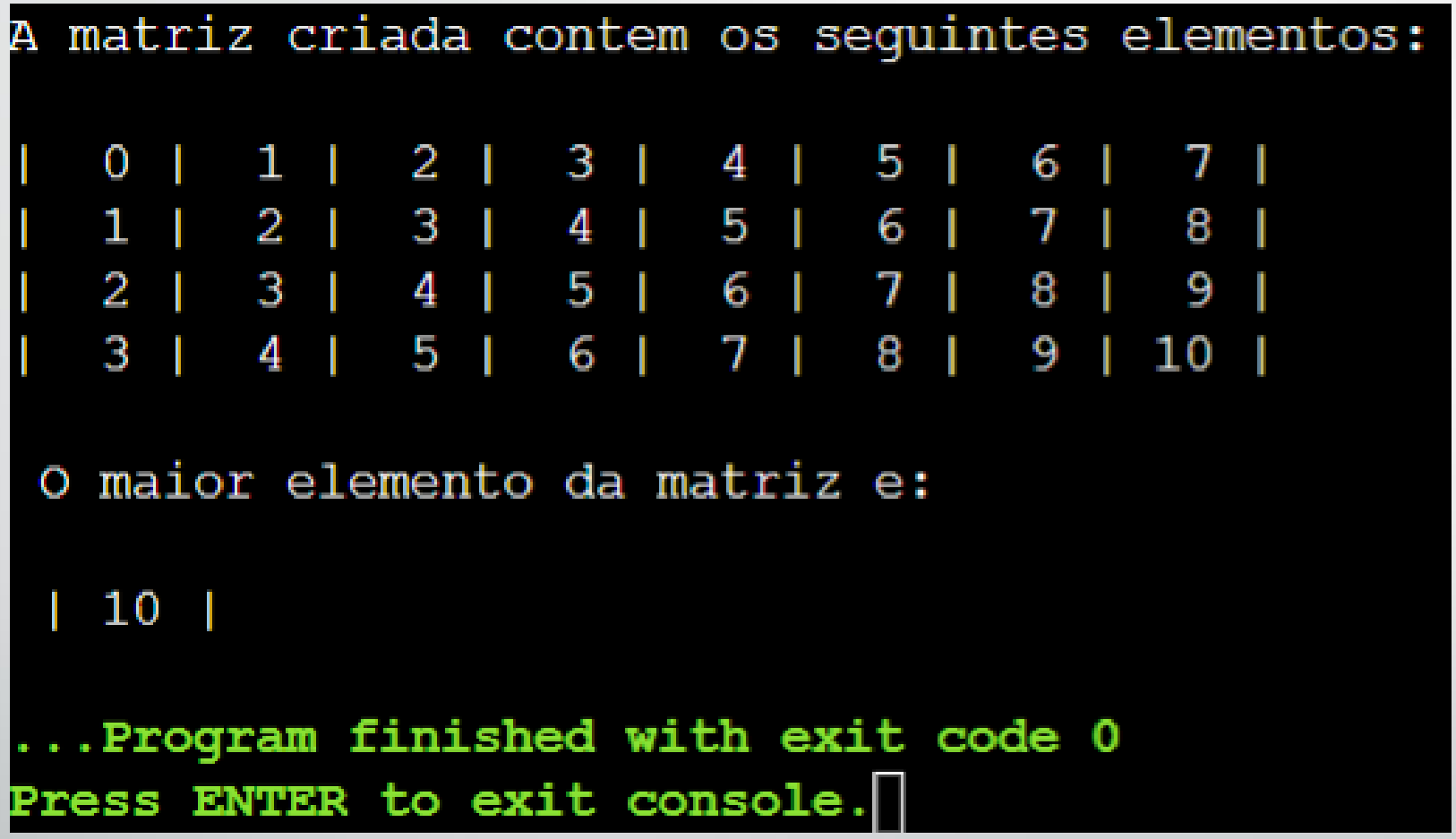
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define linhas 4
#define colunas 8
main() {
    int posicao_linha = 0, posicao_coluna = 0,
    matriz[linhas] [colunas], valor = 0, maior = 0;

    for (posicao_linha=0;posicao_linha<linhas;posicao_linha ++) {
        for (posicao_coluna=0;posicao_coluna<colunas;posicao_coluna++){

            valor= posicao_linha + posicao_coluna;
            matriz[posicao_linha] [posicao_coluna]= valor;
        }
    }
    printf("A matriz criada contem os seguintes elementos: \n\n");
```

```
19
20      for(posicao_linha=0; posicao_linha<linhas; posicao_linha++){
21              for (posicao_coluna=0;posicao_coluna<colunas;posicao_coluna++){
22                      printf("| %2d ",matriz[posicao_linha] [posicao_coluna]);
23                  }
24              printf("|\n" );
25      }
26      printf("\n O maior elemento da matriz e: \n\n ");
27
28    for(posicao_linha=0; posicao_linha<linhas; posicao_linha++){
29          for (posicao_coluna=0;posicao_coluna<colunas;posicao_coluna++){
30
31                  if(matriz[posicao_linha] [posicao_coluna] >maior)
32                      maior =matriz[posicao_linha] [posicao_coluna];
33              }
34      }
35      printf("| %d | ",maior);
36  }
37
```



Aplicação de dados homogêneos (vetores e matrizes)

- Como a **matriz** é formada de vários vetores, mas tem mais de uma dimensão, uma das melhores maneiras de inserir elementos em uma matriz é utilizar **laços de for encadeados**, nos quais um faz a varredura nas posições da linha e outro faz a varredura nas posições da coluna.

Obrigado pela atenção

Sigo à disposição pelo e-mail:

marcio.lemos@senairs.org.br