Estrutura de Dados

Prof. Dr. Eliseu LS

CONTEÚDO PELO PLANO DE ENSINO:

Alocação dinâmica e ponteiros, Arquivos, Tipos abstratos de dados: conceitos, operações, representações, manipulação, listas, pilhas e filas. Estruturas de representação de grafos. Estruturas para representação de árvores. Árvores binárias e suas aplicações.

LIVROS OBRIGATÓRIOS:

FORBELLONE,André Luiz Villar; EBERSPACHER, Henri Frederico, Lógica de Programação, construção de Algoritmos e Estrutura de Dados

MANZANO, José Augusto N. G., Estudo Dirigido: ALGORITMOS.

MIZRAHI, Victorine Viviane, Treinamento em Linguagem C++ - Módulo 1

Referência: www.cplusplus.com\reference

Notas de aulas da disciplina.

METODOLOGIA DAS AULAS

1. Recursos obrigatórios: Internet, computador e smartphone.
2. **Códigos de programas deverão ser digitados, compilados, executados, impressos digitalmente em um arquivo do ms-word ou PDF, a ser anexado somente pelo Teams, obrigatoriamente com os com os prints de execução e enunciado do código.**
3. Cada atividade valerá de 2 até 4 pontos e a clonagem de trabalhos de colegas é proibida, será descontado 1,0 de cada tópico não entregue na atividade.

DINÂMICA (Aulas práticas)

1. ***Aula interativa com exercícios de Aprendizagem:*** Nesta parte, o professor apresenta o princípio de funcionamento de cada comando e também os conceitos de lógica, em seguida exibe no datashow, de forma dinâmica ou estática, um exercício de aprendizagem contendo os conceitos de lógica e comandos que foram ensinados.

2. ***Aula dinâmica com exercícios de Fixação:*** Nesta parte da aula, caberá aos alunos a prática dos comandos e conceitos explicados pelo professor. Cabe ao aluno se esforçar para fazer os exercícios e poderá recorrer ao professor que estará disponível para tirar dúvidas via chat para sanar possíveis defeitos no código.

3. ***Forma de correção/entrega de exercícios/avaliações :*** O processo de correção destas atividades se dará de forma individualizada e pessoal pelo professor que apresentará a pontuação e os possíveis comentários de feedback pelo ms-teams. **Caberá ao aluno apresentar ao professor as suas dúvidas através do chat antes de enviar o código final, no prazo máximo de 7 dias contados a partir da publicação da atividade,** em caso de doença ou exceções, prazo poderá ser de no máximo 14 dias após a publicação inicial, com perda de 50% da pontuação da atividade. .

CORREÇÃO DOS PROGRAMAS/AVALIAÇÕES

a) A nota dos programas/avaliações será considerada nula (Zero), quando:

1. **o enunciado não for atendido na íntegra;**
2. **o código não contiver bibliotecas ou a diretiva namespace;**
3. **o código não contiver a sub rotina principal int main();**
4. **Número de sub rotinas (void ou não void) estiver diferente do que a o pedido no enunciado;**
5. **Técnica de programação (Linear ou Estruturada) estiver diferente do que a pedida no enunciado**

b) O código das SUB ROTINAS do tipo void ou não void, das avaliações, será considerado totalmente errado quando:

1. **A lista de parâmetros ou argumentos não estiver de acordo com o enunciado;**
2. **Os parâmetros e argumentos estiverem incorretos ou declarados sem necessidade;**
3. **contiver fórmulas erradas ou montadas de forma incorreta**

COMPOSIÇÃO DA NOTA

Avaliações CP, CC:

1. **CP**: nota prática, possui peso **0.50**, onde será avaliado em cada tarefa o conhecimento prático do aluno;

2. **CC**: nota cognitiva, possui peso **0.50**, onde será avaliado dentro de cada tarefa o conhecimento teórico sobre o assunto abordado na tarefa;

3. **MEDIA:** MEDIA = (NP + NC) / 2;

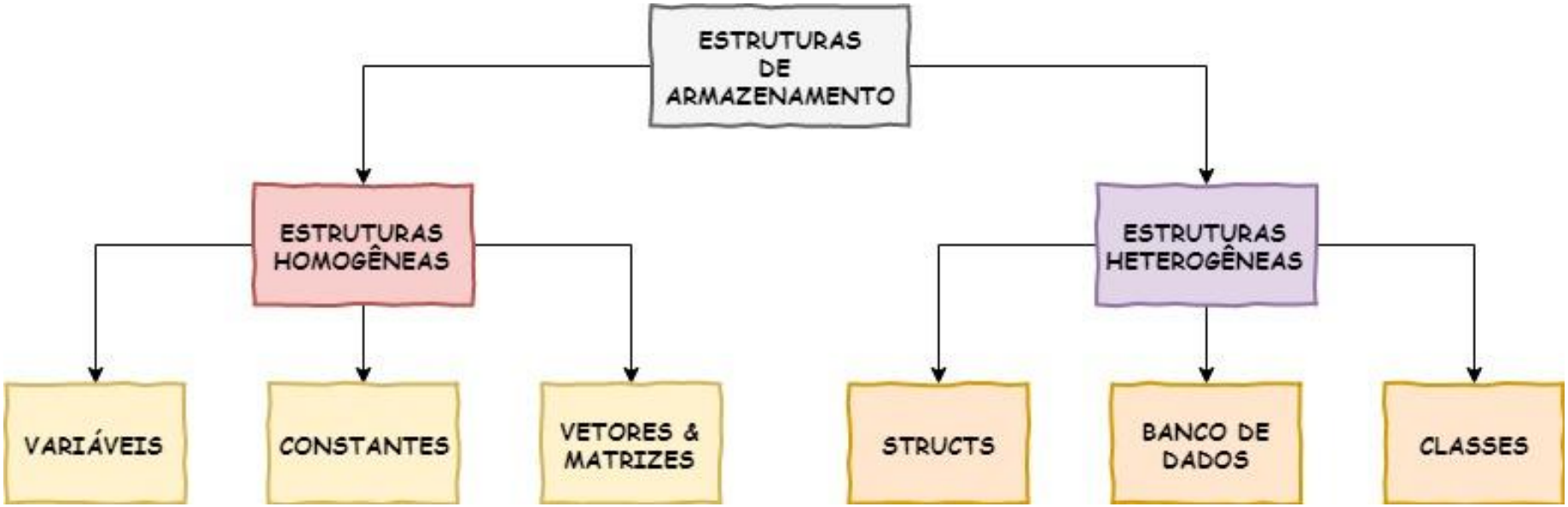
ATENÇÃO: Para a aprovação o aluno deverá acertar no mínimo 60% dos pontos propostos nas avaliações, caso hajam **atividades complementares**, as notas serão adicionadas na média final de modo que não ultrapasse o limite de 10.

1
Estruturas Homogêneas

Prof. Dr. Eliseu LS

ESTRUTURAS DE ARMAZENAMENTO

Estruturas **homogêneas** permitem o armazenamento de apenas um datatype por vez, como é o caso das matrizes, vetores, variáveis e constantes. Estruturas **heterogêneas** permitem o armazenamento de vários datatypes em uma única estrutura, como é o caso dos registros (structs), das tabelas em banco de dados, classes, etc.



Estruturas Homogêneas - Vetor (ARRAY)

VETOR é uma MATRIZ do tipo LISTA homogênea de dados, isto é, possui apenas uma dimensão apenas, onde cada elemento pode ser acessado pelo seu índice ou sua posição. Vetores podem ser declarados de forma explícita onde os dados já são inseridos na declaração ou de forma não explícita onde os dados serão lidos posteriormente:

[FORMA EXPLÍCITA]

define-se o nome e os dados,

o tamanho dependerá dos dados.

int x[ ] = { 1, 0 , -1 };

n = total de elementos = 3

1 está na posição 0

0 está na posição 1

-1 está na posição (n - 1) que é 2

Vetor X [ ]	
posição	**elemento**
0	**1**
1	**0**
2	**-1**

[NÃO EXPLÍCITA]

define-se o nome e o tamanho somente. Os dados são inseridos depois.

```
int x[3];
x[0] = 5;
x[1] = 10;
x[2]=3;

string nome[3];
int i = 1;
nome[i] = “Maria”;
i = 2;
nome[i] = “João”;
```

Vetores Explícitos (Listas) & Sizeof()

Os vetores explícitos são declarados sem tamanho, já que seu tamanho será definido pelo quantidade de elementos inseridos. Para descobrir o tamanho desse vetor, usa-se a função sizeof (), você também pode fazer cálculos e outras operações com elementos do vetor.

PROGRAMA 1

Este exemplo cria o vetor **x [ ]** com **n** elementos, em seguida descobre o número de elementos (n), sizeof(x) é o tamanho total do vetor **x**, sizeof(int) é o tamanho de um elemento do tipo int. cria uma variável aux para armazenar a soma do quadrado do elemento do vetor com ele mesmo. O loop for é utilizado para armazenar e exibir o valor de aux.

```
#include "iostream"
#include "math.h"
using namespace std;
int main ()  {
int x[ ] = { 1, 0 , -1 };
int n = sizeof(x)/sizeof(int); // total de
elementos
int aux = 0;
for (int i = 0; i < n ; i ++)
{   aux = pow( x [ i ],  2 )  +  x [ i ] ;
     cout << aux << endl;
}
return 0; }
```

Simulando o programa 1 - vetor explícito

n	i	x[i]	aux
3	0	x[0] = 1	$1^2 + 1 = 2$
3	1	x[1] = 0	$0^2 + 0 = 0$
3	2	x[2] = -1	$-1^2 + -1 = 0$

Os valores calculados de aux são respectivamente: 2 , 0, 0

Vetores não Explícitos (Necessário leitura)

Os vetores não explícitos são declarados com tamanho pré-definido, através do tamanho é utilizado um loop for para fazer a leitura dos elementos de cada posição.

PROGRAMA 2

Este exemplo cria vetores do tipo double e int, os vetores salário e idade são lidos diretamente no for. No último loop for os vetores poderão ser exibidos todos juntos, o salário é exibido com 10% de aumento.

***ATENÇÃO:** Antes de digitar o programa, faça a simulação no caderno com os seguintes salarios e idades:*

SAL= 1000, 1500, 2000

IDA= 18 , 20, 22

```
#include "iostream"
#include "math.h"
using namespace std;
int const n=3;
double sal[n];
int ida[n];
int main ()  { setlocale(LC_ALL,
"Portuguese-brasilian");
double x;
for (int i = 0; i < n ; i ++) {
    cout << "\nDigite salário e idade separados
por espaço:";
     cin >> sal[i];
   cin >> ida[i]; }
for ( int i=0; i<n; i++)
{
x = sal[i] * 1.1;
cout << x  << " - " << ida[i] << endl;
}
 return 0; }
```

TAREFA DE FIXAÇÃO / AVALIAÇÃO

a) Fazer o programa fonte de um algoritmo para exibir os vetores: string nome [] = {“Sara Lemes”,”Ricardo Jafé”} , double salario [] = {12000, 10243.20} e int idade[]={30, 45}; O salário deverá ser exibido com 10% de aumento. Implemente um menu de execução.

b) Fazer um programa que leia através do teclado dados de clientes para um cadastro de um site qualquer, os dados deverão ser armazenados em vetores: nome, cpf, endereço, email e telefone. Permitir a leitura de no máximo 4 registros apenas

NOTA: Entrega obrigatória individual pelo Teams, em um arquivo DOCX ou PDF, juntamente com enunciado, programas e prints da tela de execução. A regra é descontar um ponto, no mínimo, para cada tópico não entregue.