PUC - Rio
INF1301 - Programação Modular
Prof. Flávio Bevilacqua
Grupo: Ian Albuquerque Raymundo da Silva ;
Lucas Ferraço de Freitas ;
Victor Augusto Lima Lins de Souza .

Lista de Exercícios para P1

1) Dê exemplo de um código com coesão procedural.
R.: Sequência de funções para atualizar uma coleção de filmes separados por tipo:
- tpFilme * filmesTipo ;
tpPredicadoBD predAcao ;
marcarComoSelecionado(butaoAcao) ;
predAcao = configPredicado(tipoAcao) ;
filmesTipo = recuperarBaseDados(predAcao) ;
inserirFilmesBaseMostrada(filmesTipo) ;
recarregarBDTela( ) ;
atualizarTela( ) ;

2) Uma baixa coesão resulta em um encapsulamento ruim. Certo ou errado, justifique.
R.: Certo, uma coesão baixa implica em baixo grau de interdependência entre os elementos do mesmo, assim os elementos provavelmente não tratam dos mesmos assuntos ou não pertencem às mesmas classes que os outros, logo, o encapsulamento vai ser ruim, pois a documentação vai tratar de diferentes assuntos e aspectos dentro de um mesmo módulo e os dados não terão como ser devidamente protegidos, porque eles não se relacionam corretamente, o que implica em não saber em quais contextos restringir a visibilidade dos dados.

3) Explique o que é callback e dê um exemplo (diferente do que foi dado em sala).
R.: Um *callback* ocorre quando em uma mesma transação entre um módulo servidor e um cliente, ambos trocam de função (cliente vira servidor e vice-versa). Exemplo: em um serviço de *streaming* de séries, o módulo de streaming (cliente) solicita ao banco de dados (servidor) uma série, logo após o módulo de banco de dados (cliente) solicita qual temporada o streaming (servidor) precisa, em seguida o módulo de streaming (cliente) pede a temporada específica ao banco de dados (servidor). A situação ainda se repete quando o módulo de banco de dados (cliente) solicita qual episódio o streaming (servidor) precisa, em seguida o módulo de streaming (cliente) pede um episódio específico ao banco de dados (servidor).

4) Existem construtos que não são artefatos. Certo ou errado, justifique com exemplos.
R.: Errado, artefatos são todas e quaisquer coisas criadas durante o processo de desenvolvimento e que sejam independentes entre si, logo, um construto é um artefato, tendo em vista que o mesmo é um resultado parcial (build) do desenvolvimento.

5) Trace um paralelo entre a programação modular e a orientação a objetos.
R.: A orientação a objetos trata cada classe de uma forma modularizada, pois cada classe tem seus atributos e seus métodos próprios, assim como os módulos da programação modular, onde cada um trabalha com suas funções e estruturas que tratam dos mesmos assuntos.

6) Quais das três arquiteturas abaixo apresentam TAD's de estruturas genéricas?
a)

b)

c)

R.: Levando em consideração que todas as estruturas tem que ser genéricas, a arquitetura da letra B é a resposta, pois o módulo PRINCIPAL é o que trabalha com os TAD's LISTA e ÁRVORE separadamente e tratando os dois sem nenhum tipo de dependência com nenhum outro módulo, logo, LISTA e ÁRVORE não conhecem os valores dos seus nós.

7) Complete as arquiteturas da questão 6 e apresente para cada uma:
- um exemplo de estrutura ;
- uma sequência das funções para criar esta estrutura .

R.:

Lista:

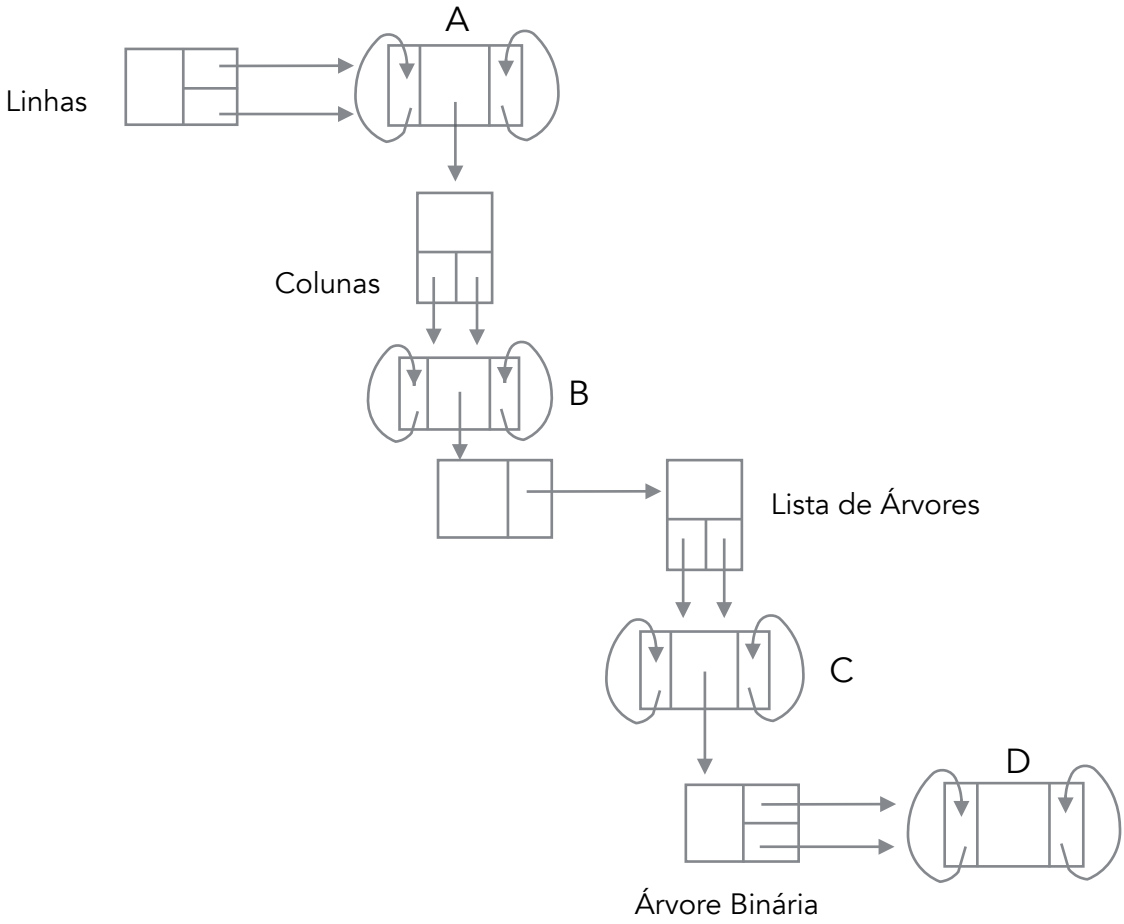
```
struct elemLista
{
        void * valor ;
        elemLista * proxNo ;
        elemLista * antNo ;
} ;
typedef struct elemLista noLista ;

noLista * criaNoLista( ) ;
int insereNoLista( noLista no , void valor ) ;
int retiraNoLista( noLista no ) ;
int obtemNoLista( void * valor ) ;
void destroiLista ( noLista * primNo ) ;
```

Árvore:

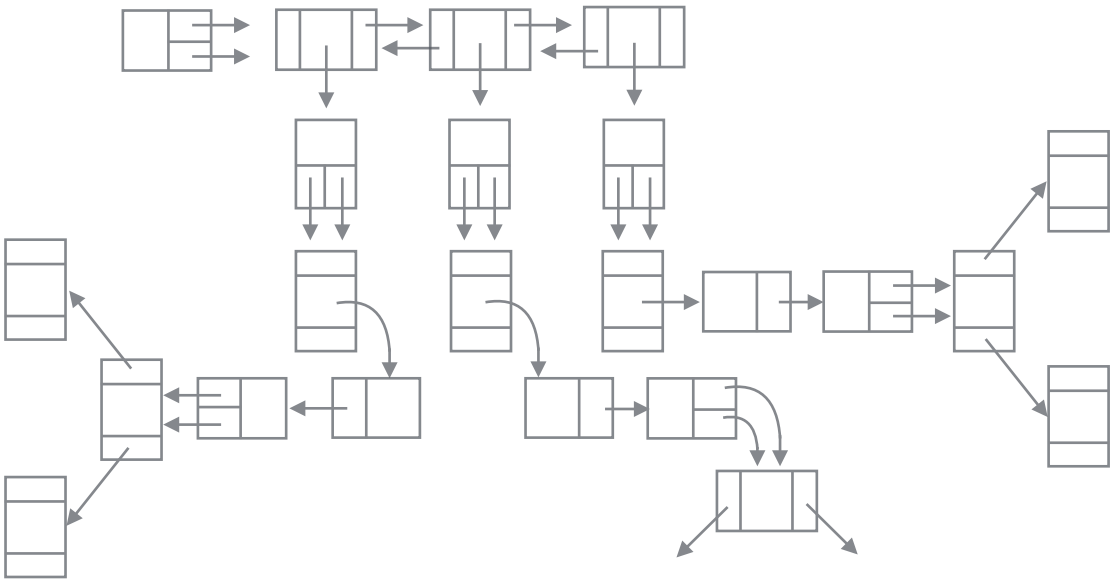
```
struct elemArvore
{
        void * valor ;
        elemArvore * raiz ;
        elemArvore * folhaEsq ;
        elemArvore * folhaDir ;
} ;
typedef struct elemArvore folhaArvore ;

folhaArvore * criaFolhaArvore( ) ;
int insereFolhaArv( folhaArvore folha , void valor ) ;
int retiraFolhaArv( folhaArvore folha ) ;
int obtemNoLista( void * valor ) ;
void destroiArvore ( folhaArvore * raiz ) ;
```

8) Apresente o modelo físico, exemplo físico e assertivas (conforme necessidades) de uma matriz composta por lista de listas, em que cada célula possui um identificador com as coordenadas e um ponteiro para uma lista de árvores binárias genéricas. (OBS: Todas as estruturas possuem cabeças)

R.: - Modelo físico:



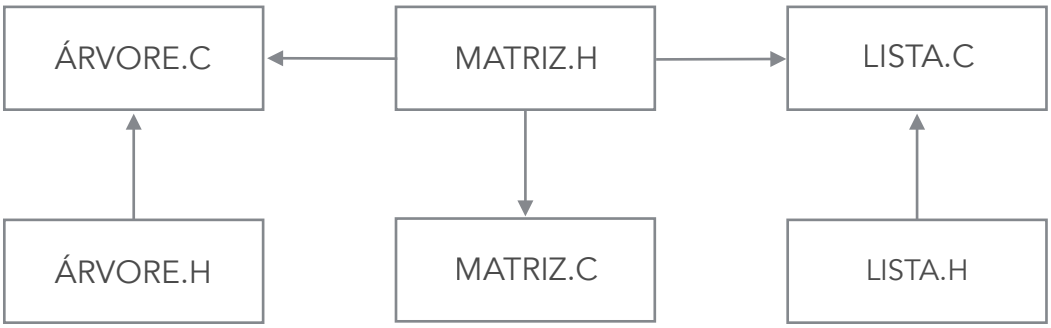
- Exemplo físico:



- Assertivas de A (Matriz):
    - Mesmas das de Lista (listadas a seguir);
    - Tamanho de todas as listas iguais.
- Assertivas de B, C (Lista):
    - Se pAnt ≠ NULL, então pCorr -> pAnt -> pProx = pCorr;
    - Se pProx ≠ NULL, então pCorr -> pProx -> pAnt = pCorr.
- Assertivas de D (Árvore):
    - Filho não aponta para o pai;
    - Elemento de subárvore a esquerda não aponta para elemento de subárvore a direita e vice-versa.

9) Apresente a arquitetura do programa que utiliza a estrutura da questão 8, considerando que matriz, lista e árvore são TADs específicos.

R.:



10) Para cada requisito abaixo, diga se é bem ou mal formulado e qual é o tipo dele.

a) A pesquisa deverá retornar rapidamente o resultado.

b) O programa possui uma interface bem intuitiva.

c) Ao informar o INPS, a funcionalidade apresenta a razão social.

d) O resultado da pesquisa lista de logradouros não ultrapassa 1000 linhas.

e) Não será apresentado na pesquisa lista de logradouros e quantidade pesquisado.

R.: a) Mal formulado, pois não especifica o quão rápido é necessário. Tipo: não funcional.

b) Mal formulado, pois não é definido o que é ser bem intuitivo. Tipo: não funcional.

c) Bem formulado, pois determina o que acontece exatamente ao pesquisar o campo específico. Tipo: funcional.

d) Bem formulado, desde que a pesquisa lista de lougradoros esteja bem definida. Tipo: funcional.

e) Bem formulado, desde que a pesquisa lista de lougradoros esteja bem definida. Tipo: inverso.

11) Apresente um trecho de código do trabalho que exemplifique uma declaração de variável sem definição.

R.: "typedef struct PCA_tagPeca * PCA_tppPeca" , presente na interface do módulo PECA do trabalho 2.

12) Qual é a relação entre os conceitos de baseline e construto?

R.: *Baseline*, por definição, é uma integração de todos os módulos (cada um com sua própria versão), que gera uma versão da aplicação pronta para ser utilizada ou executada, a qual constitui um construto.

13) Como é possível utilizar os printf's de auxílio ao desenvolvimento sem que os mesmos apareçam na versão entregue ao cliente e tenham que ser recolocados numa futura manutenção.

R.: Através de um *define* que é capaz de determinar a versão a ser executada (para o cliente ou para o desenvolvedor) colocando um *#ifdef* antes de cada *printf* para incluí-lo ou não na versão.

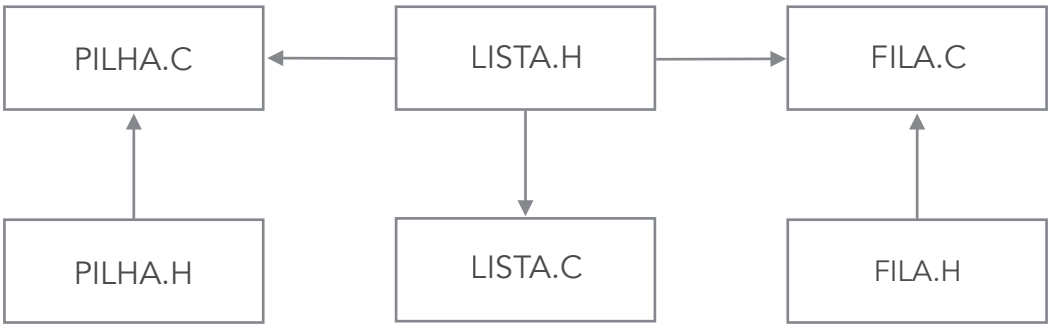
14) Dê um exemplo de requisito funcional originado por um não funcional.

R.: Requisito não funcional: todo registro de novo usuário deve ser protegido por uma conexão segura HTTPS com TLS 1.3.

Requisito funcional: sistema deve ser capaz de cadastrar novos usuários com a tecnologia especificada no requisito não funcional acima.

15) Dê um exemplo de reutilização de um módulo em duas arquiteturas de aplicações com objetivos distintos.

R.: Um módulo LISTA pode ser usado para implementar um módulo PILHA ou um módulo FILA.



16) Qual é a pior coesão existente? Exemplifique.

R.: A coesão incidental. Um exemplo é uma aplicação para *streaming* de filmes onde só houvesse um mesmo módulo, logo, em um mesmo *header file* teríamos:

- obterFilmes(tpTipoFilme tipoFilme);
- comecarVideo(tpFilme nomeFilme);
- solicitarLoginUsuario().

17) O encapsulamento dentro de uma aplicação pode ser percebido de várias maneiras. Explique três.

R.: O encapsulamento pode ser percebido por:

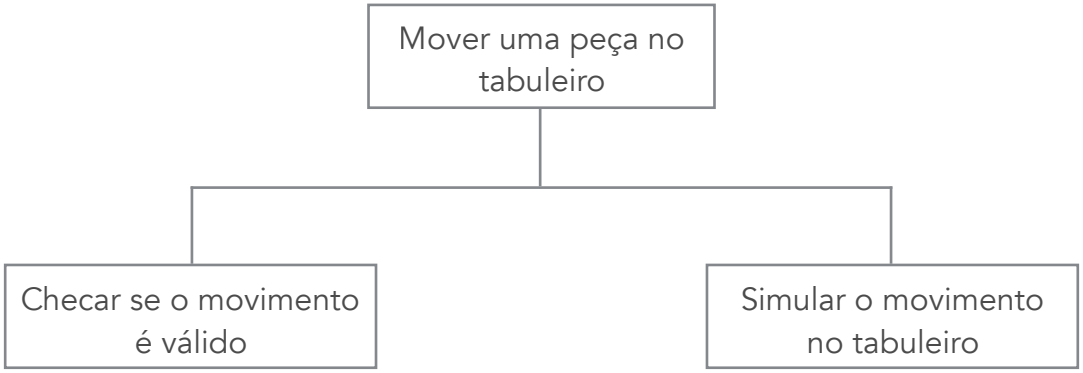
- documentação, da qual:
    - externa -> documentação voltada para o programador cliente;
    - interna -> documentação voltada para o programador servido;
    - de uso -> voltada para o usuário;
- dados, através de expressões como *private*/*protected*/entre outras que determinam o espaço de uso (quem a enxerga) da função ou variável em questão;
- diretivas de pré-processamento, que permitem definir quando certas partes do código devem aparecer/executar ou não.

18) Qual é a vantagem de se manter o menor número de header files em um programa modular?

R.: Não existem vantagens de se ter um menor número de *header files* em um programa. Quanto menor o número de *headers*, menos coeso é o programa, pois o módulo se torna menos específico.

19) Apresente o conjunto solução dentro de uma estrutura de decomposição de uma função qualquer dos seus trabalhos do período.

R.: Conjunto solução da função de mover uma peça pelo tabuleiro (TAB_MoverPecaTabuleiro) do módulo TABULEIRO.



20) Qual é o nome deste conjunto de componentes?

R.: O conjunto de componentes acima é o conjunto solução do problema apresentado.