Programação Modular

Lista de Exercícios para a P1

Grupo: Eric Grinstein, Jordana Mecler e Leonardo Wajnzstok

1) **Explique a relação entre o callback e os conceitos de módulo de implementação e definição.**

O módulo de implementação representa o servidor, pois possui códigos internos e estruturais encapsulados. O módulo de definição representa a interface com o cliente, pois possui as funções de acesso. Desse modo, o callback representa a inversão desses papéis quando a função de acesso requisita mais dados ao cliente para poder finalizar seu serviço (operar corretamente).

2) **Qual é a vantagem de um bom encapsulamento?**

Um bom encapsulamento garante a proteção de dados e de código, bem como protege a documentação interna e protege o uso indevido de código que não deve ser modificado.

3) **Dê um exemplo de diferença entre documentação interna e externa.**

Na documentação interna há informações técnicas quanto ao código, pois está relacionada ao servidor (módulo de implementação), enquanto que a externa possui informações sobre as funções exportadas pelo servidor, pois está relacionada a interface com o cliente (módulo de definição). A documentação externa deve ser simples, de modo que um outro programador possa entendê-la.

4) **Dê um exemplo de framework (que não seja o arcabouço). Explique com esse exemplo a parte genérica e específica (hotspot).**

O framework de programação web Python, Django. A parte genérica é composta por procedimentos comuns a aplicações web, como mecanismos de comunicação com bancos de dados. A parte específica corresponde as páginas web criadas, código de conexão com o banco de dados, etc.

5) **Explique o que torna um acoplamento de baixa qualidade. Apresente um exemplo de acoplamento de baixa qualidade e outro de qualidade alta.**

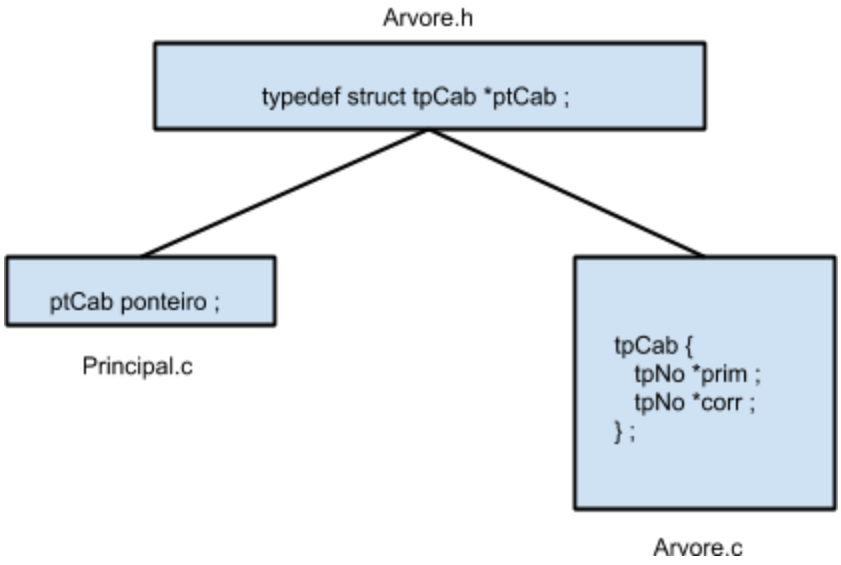
Um acoplamento possui baixa qualidade quando tem um conector muito grande, possui muitos conectores e possui alta complexidade dos conectores.

ex.: int Conta(tp** x, int tam) ;
int ContaFolhas(tp** arvore) ;

6) **Trace um paralelo entre as vantagens da programação modular e da programação orientada a objetos.**

Tanto a programação modular quanto a orientada a objetos tornam mais fácil a manutenção de código. Ambas trazem como vantagem também a reusabilidade do código. O encapsulamento é comum a ambas.

7) **Apresente um exemplo de interface fornecida por terceiros em uma implementação de módulos em C.**



8) **Descreva as vantagens e desvantagens do uso de: bibliotecas estáticas e bibliotecas dinâmicas.**

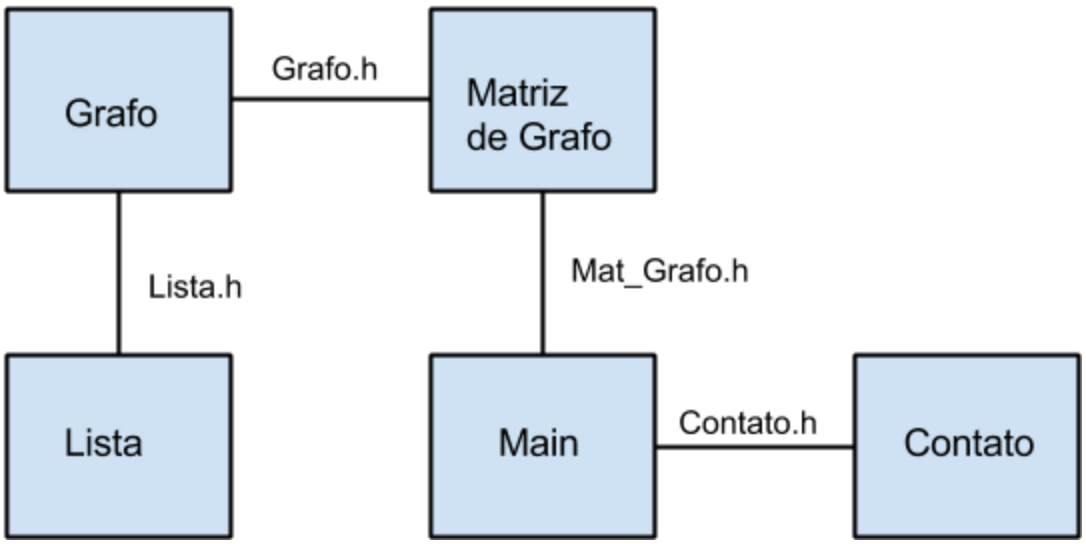
Bibliotecas estáticas: Vantagens: Não é necessária a instalação da biblioteca na máquina.

Desvantagens: É necessário sempre recompilar junto ao código. Programa mais pesado.

Bibliotecas dinâmicas: Vantagens: O código fonte pode ser omitido, pois necessita-se apenas na execução. Reuso de código sem necessidade de alteração.

Desvantagens: É necessária a instalação da biblioteca na máquina. Programa mais leve.

9) **Uma aplicação armazena em uma matriz de grafos várias redes de contatos. Cada grafo corresponde a uma rede de contatos de um país. Cada contato armazena nome e e-mail de uma pessoa. O grafo é construído apenas com listas duplamente encadeadas encapsuladas em um módulo específico. O grafo é encapsulado também em um módulo. Elabore o diagrama de arquitetura da aplicação e apresente a lista de módulos de definição e implementação, bem como a relação de nomes das funções de acesso disponibilizadas por cada módulo de definição.**



Módulos de definição:
Lista.h
Grafo.h
Mat_Grafo.h
Contato.h

Módulos de implementação:
Lista.c
Grafo.c
Mat_Grafo.c
Contato.c
Main.c

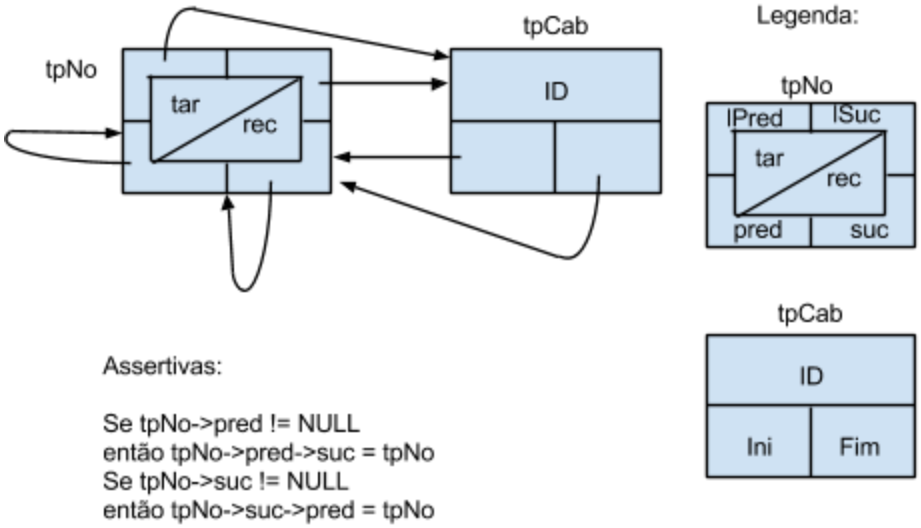
Funções de acesso:

Lista:	Cria_Lista	Grafo:	Cria_Grafo
	Destroi_Lista		Destroi_Grafo
	Insere_Lista		Insere_Grafo

Matriz de Grafo:	Cria_Matriz Destroi_Matriz Insere_Matriz	Contato:	Cria_contato Destroi_Contato

10) Gere o modelo estrutural com exemplo e assertivas para uma estrutura grafo genérico que não utilize a estrutura de lista encadeada (implementação diferente da mostrada em aula).

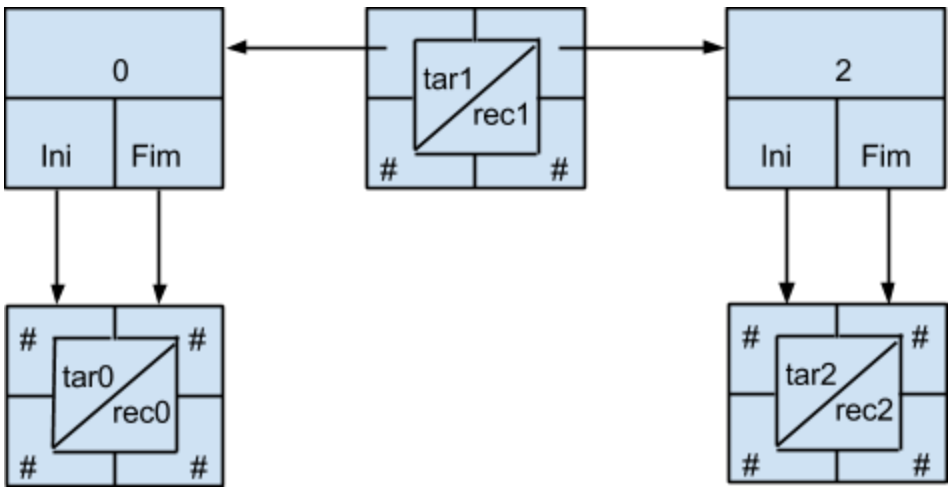
Modelo:



Assertivas:

Se tpNo->pred != NULL
então tpNo->pred->suc = tpNo
Se tpNo->suc != NULL
então tpNo->suc->pred = tpNo

Exemplo:



11) Qual é a diferença entre restrição e requisito inverso? Explique com exemplos.

Restrições são regras que limitam as alternativas de desenvolvimento, enquanto requisitos inversos são aqueles que não serão feitos na aplicação.

ex.: Restrição: O jogo só funcionará em computadores usando sistema operacional Windows.

Requisito inverso: O jogo não disporá de uma interface gráfica fora do prompt de comando do Windows.

12) Dê um exemplo de requisito funcional mal formulado (diferente do visto em aula).

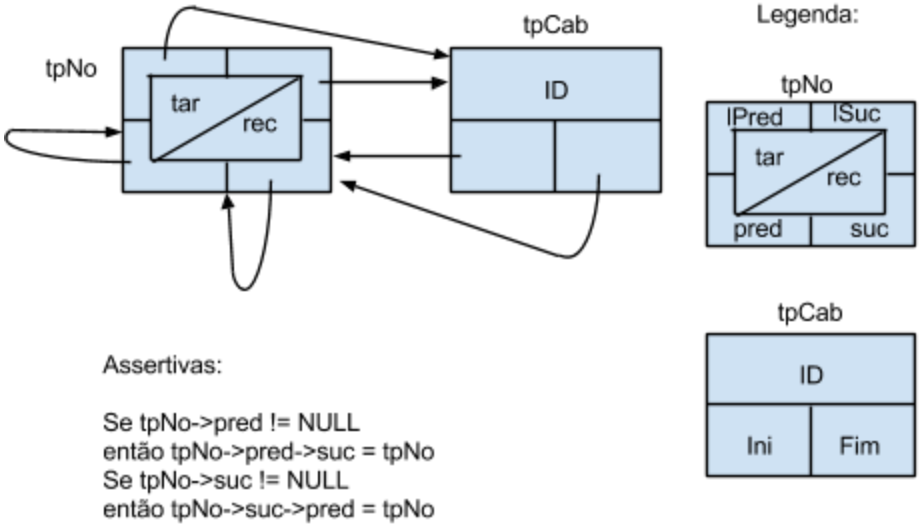
A aplicação deve mostrar as fotos mais bonitas primeiro.

13) Para que serve a etapa de elicitação de requisito.

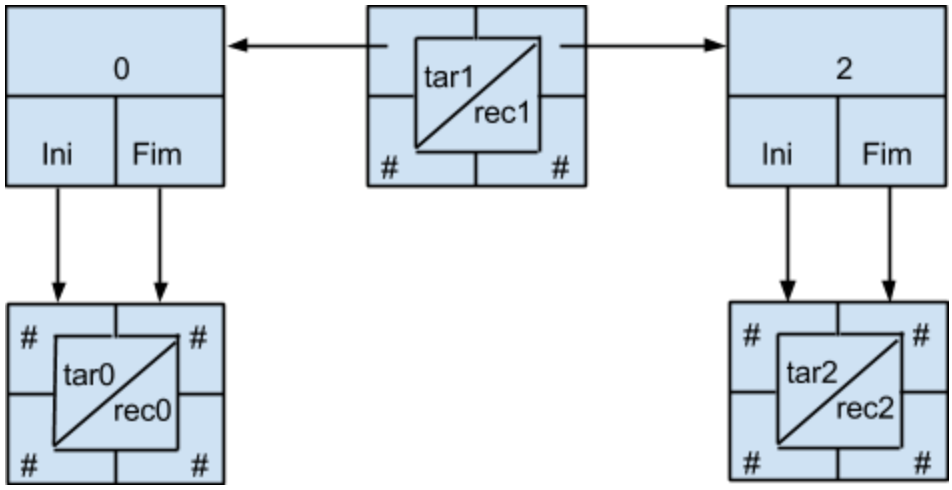
A elicitação de requisitos serve para listar todas as necessidades da aplicação junto ao cliente, e a partir disso criar uma documentação, cronograma e preço. Dessa forma evita-se problemas futuros com o cliente.

14) Uma aplicação utiliza uma estrutura para armazenar um cronograma completo de tarefas e recursos. Este cronograma armazena predecessoras e sucessoras. Apresente o modelo e o exemplo que você utilizaria para a estrutura que armazena este cronograma em memória.

Modelo:



Exemplo:



15) Dê exemplo de uma função morta existente em seu trabalho.

LIS_tpCondRet LIS_InserirElementoAntes(LIS_tppLista pLista , void * pValor) ;

16) Apresente um requisito funcional criado a partir de um não-funcional.

A aplicação terá um campo para login e senha.