

EDA - Estruturas de Dados

- Objetivos:

Capacitar o aluno a desenvolver soluções computacionais eficientes através da utilização de algoritmos e estrutura(s) de dados adequada(s). Desenvolver a lógica e o raciocínio abstrato.

- Motivação

“Programas = Algoritmos + Estrutura de Dados” (Niklaus Wirth)

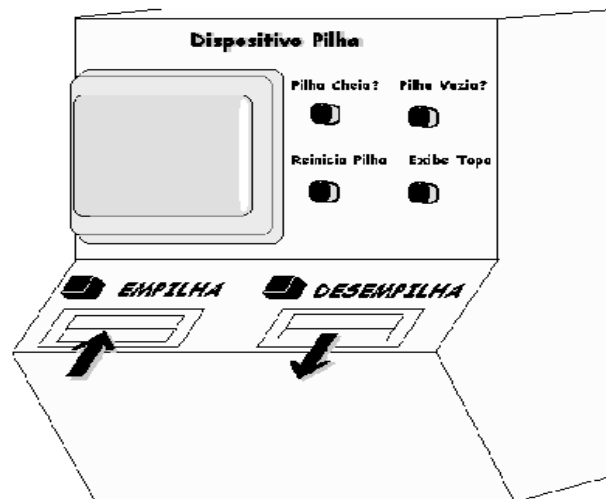
Problemas são resolvidos por programas contendo estruturas de dados

- Metodologia

Implementação (em linguagem C) de estruturas de dados (básicas) na forma de servidores de funcionalidades

• Como é esse “servidor”?

- Consiste do encapsulamento de funções e representações internas de dados;
- Uma ocultação da implementação de um conjunto de subprogramas e os dados que estes manipulam;
- Provedor de serviços para a solução de um problema (aplicações de estruturas de dados resolvem problemas);
- Comunicam-se com o mundo externo por meio de uma interface que “publica” os serviços que realizam;



- Qual é o objetivo desse encapsulamento?

O **encapsulamento** de dados e subprogramas **modulariza** o sistema construído;

O encapsulamento visa o desenvolvimento de módulos desacoplados (independentes) ou fracamente acoplados entre A aplicação (cliente da est de dados) e a própria estrutura de dados;

O encapsulamento permite a **separação de entidades lógicas** em blocos. Exemplo: módulo de aplicação da estrutura de dados e módulo de implementação da estrutura de dados;

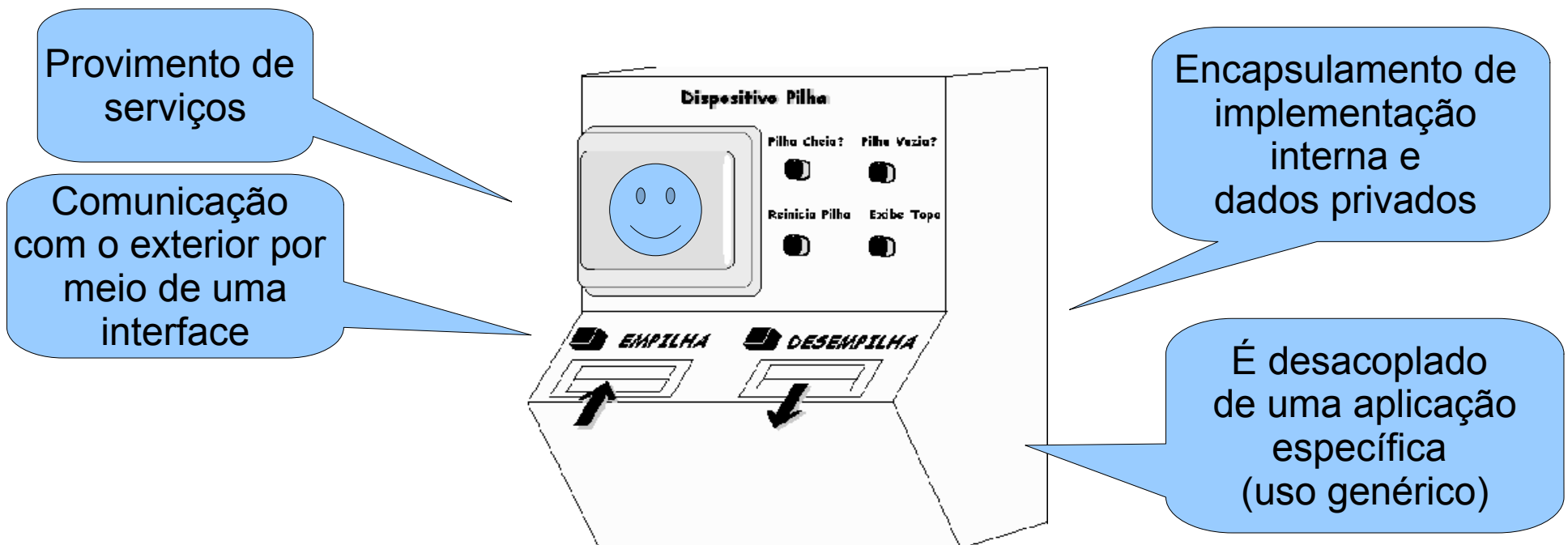
Essa independência entre blocos de código **facilita a manutenção e depuração do sistema**;

As partes se combinam, mas não se “misturam”



● Porque utilizar encapsulamento na disciplina EDA?

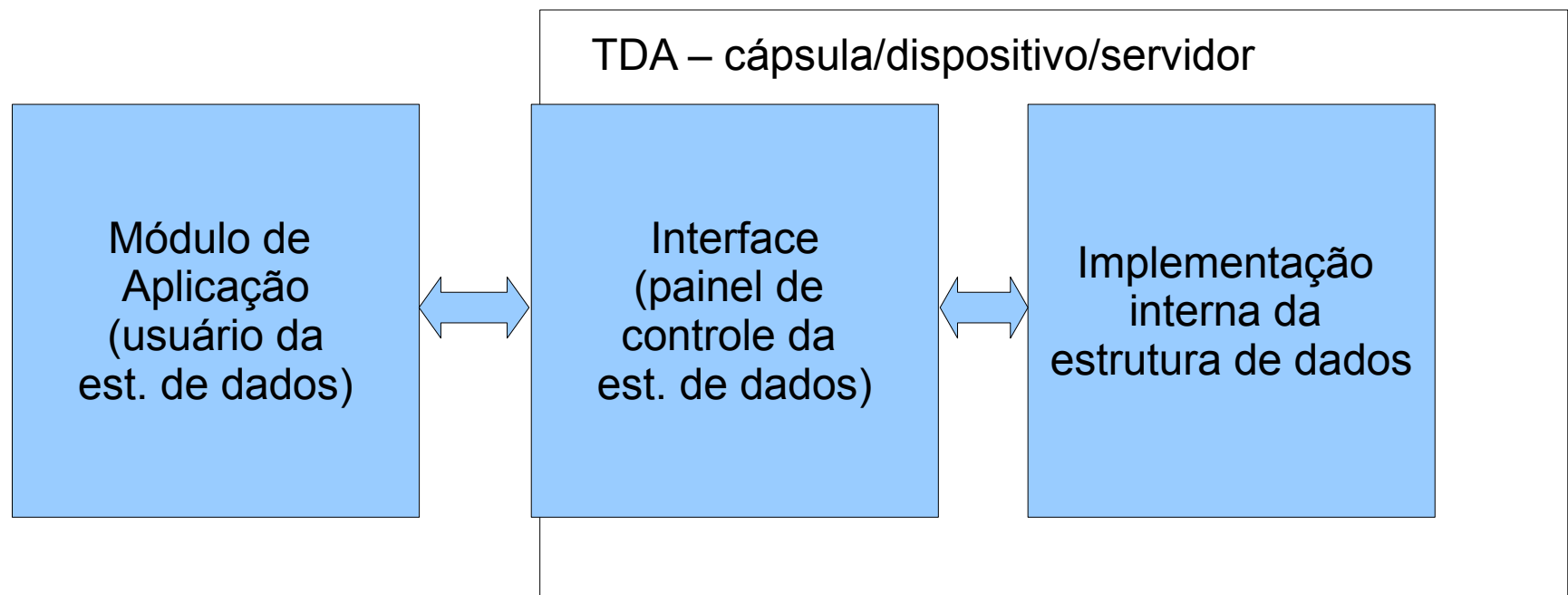
- Esse modelo vem de encontro aos objetivos da disciplina EDA:
 - É focado no desenvolvimento da estrutura de dados independente de aplicação específica;
 - É focado na oferta de variadas funcionalidades;



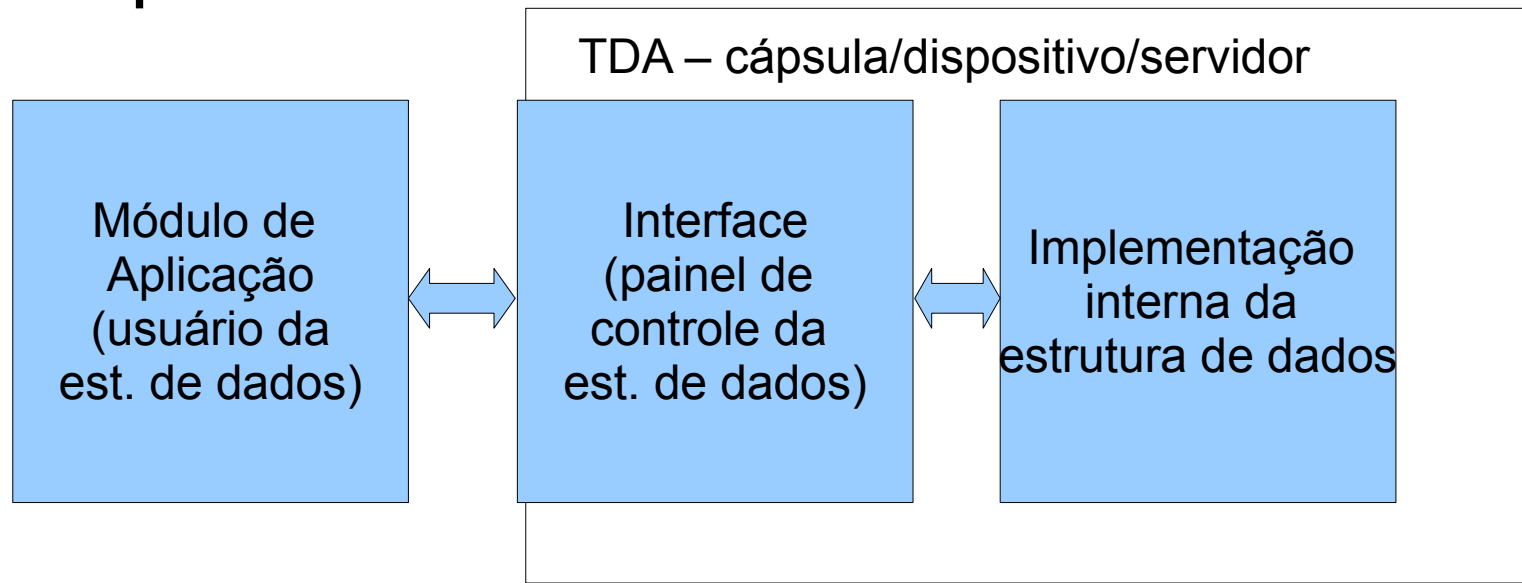
Como iremos implementar?

- A sintaxe de uma linguagem OO já incorpora comandos para a criação/instanciação de classes garantindo a abstração e encapsulamento;
- EDA utiliza a linguagem C, a qual não incorpora os comandos similares aos da OO, teremos que estruturar os nossos encapsulamentos partir de uma arquitetura de arquivos;

Para EDA, os encapsulamentos serão logicamente construídos em linguagem C;



Como iremos implementar?



Não há **restrições quanto a quantidade de serviços** (operações) providos pelo TDA. Utilizaremos a seguinte **classificação**:

- Operações **construtoras**: instanciam (criam na memória principal) um nova instância da estrutura de dados e retornam uma referência (endereço) para acesso.
- Operações de **acesso**: retornam informações sobre a estrutura de dados sem modificar o seu estado interno.
- Operações de **manipulação**: alteram o estado interno da estrutura de dados.
- Operações **destrutoras**: liberam todo o espaço de memória alocado para uso da estrutura de dados;

Documentação das operações?

As **operações são documentadas** através da declaração da *pré* e *pós* condição junto ao respectivo protótipo:

- **Pré-condições:** são aquelas assumidas pela estrutura de dados como verdadeiras, é responsabilidade do usuário (cliente do TDA) estabelecer estas condições antes da chamada à respectiva operação;
- **Pós-condições:** descrevem os efeitos da chamada a uma determinada operação.

Exemplo:

```
/*Pré-condição: a existência de estrutura de dados instanciada cujo
endereço é passado para a função.
```

Pós-condição: se a estrutura de dados estiver cheio retorna a macro SIM, caso contrário retorna a macro NAO.* /

```
int testaCheia(pPE p);
```

As pré e pós-condições são declaradas no arquivo dos protótipos das funções (operações/serviços) disponíveis na interface do TDA.

Para estrutura de dados encapsulada é mandatório que:

Todas as operações (serviços) devem garantir: **atomicidade e consistência**

Atomicidade – ou tudo é feito ou nada é feito:

A operação é considerado um sucesso se e somente se todas as ações que a constituem tiverem sido executadas com sucesso completo. O fracasso parcial ou total de alguma ação leva ao desfazimento de todas as demais ações que constituem a operação, recuperando o TDA ao seu estado anterior a chamada da operação.

Exemplo: na transferência de valores entre contas é necessário que, da conta origem seja retirado um valor X e na conta destino seja somado o mesmo valor X. As duas operações devem ser completadas sem que qualquer erro aconteça, caso contrário, todas as alterações feitas nessa operação de transferência devem ser desfeitas;

Consistência:

A execução da operação deve levar o TDA de um estado consistente a um outro estado consistente.

Exemplo: operações bancárias não podem alterar a consistência dos dados pessoais do cliente de uma conta, por exemplo.

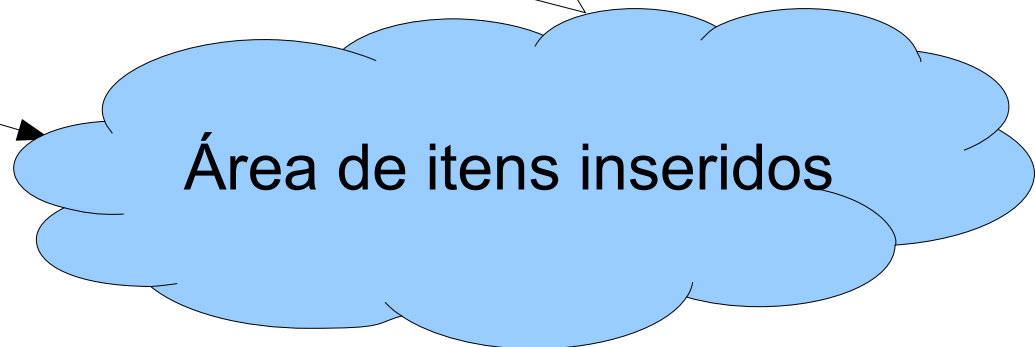
Como iremos implementar?

- As implementações das estruturas de dados em EDA sempre contarão com descritor (etiqueta contendo metadados) e área de itens inseridos

Descritor: contendo informações para gerenciamento da estrutura de dados



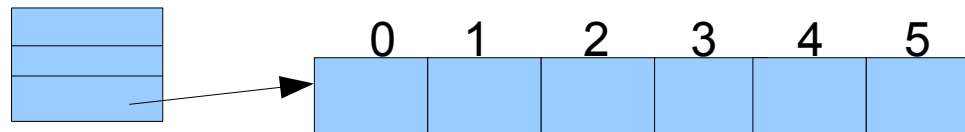
Área contendo informações inseridas pela aplicação da est. de dados



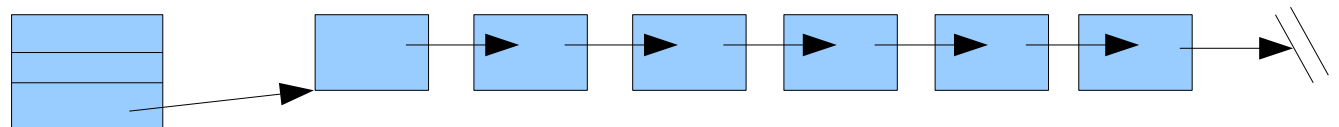
Área de itens inseridos

Como iremos implementar?

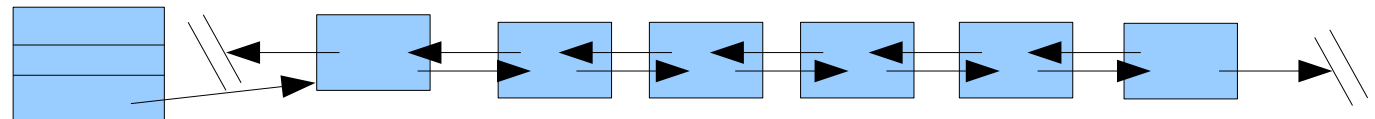
- Utilizaremos duas estratégias de implementação da área de itens inseridos:
 - Tamanho máximo predeterminado e estático: a área de dados fica contida normalmente em um vetor cujo tamanho máximo é alocado na criação da estrutura de dados;



- Tamanho máximo não predeterminado: não há um vetor construído para conter a área de dados;
 - A estrutura se desenvolve em função de inserções (alocando memória) ou remoções (liberando memória);
 - Cada elemento inserido tem que ser encadeado na estrutura por meio de apontadores;



Encadeamento
Simple e Duplo

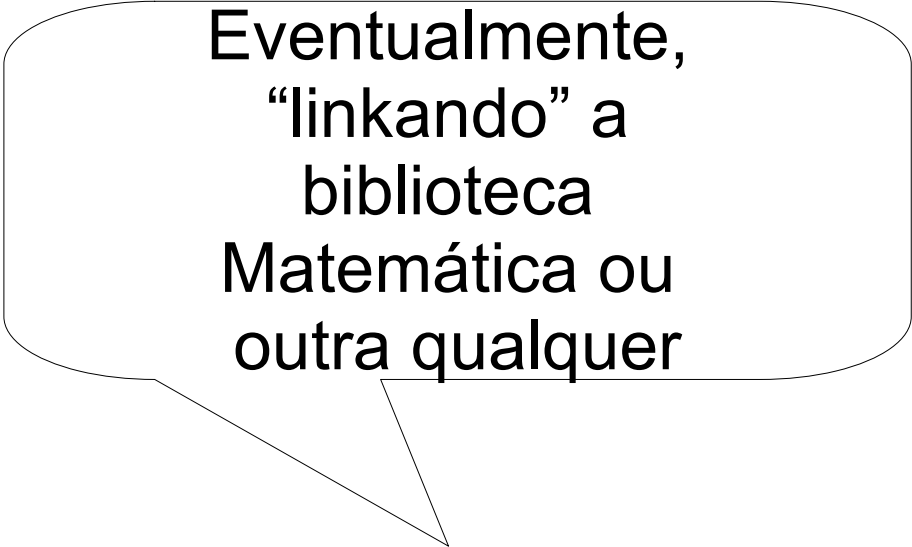


Exemplos de possíveis designações para as implementações de Pilhas:

- Utilizaremos designações tais como:
 - PE: pilha estática
 - Tamanho máximo pré-determinado, contida em um vetor, sem encadeamento;
 - PDSE: Pilha Dinâmica Simplesmente Encadeada
 - Tamanho máximo pré-determinado (estático), não é contida em vetor e é encadeada;
 - PDDE: Pilha Dinâmica Duplamente Encadeada
 - Tamanho máximo pré-determinado (estático), não é contida em vetor e é encadeada;
 - PESE: Pilha Estática Simplesmente Encadeada
 - Tamanho máximo não pré-determinado (dinâmico), contida em um vetor e encadeada...

Faremos algo similar para as demais estruturas Fila, Lista, Árvore, etc, por aí vai... é meio exótico mas funciona...

Como compilar?



Eventualmente,
“linkando” a
biblioteca
Matemática ou
outra qualquer

```
>> gcc -Wall -o progSaida aplica.c pdse.c -lm
```

Ou pela sua IDE favorita (Codeblocks, Geany, DevCpp, etc...) utilizando um arquivo de projeto.