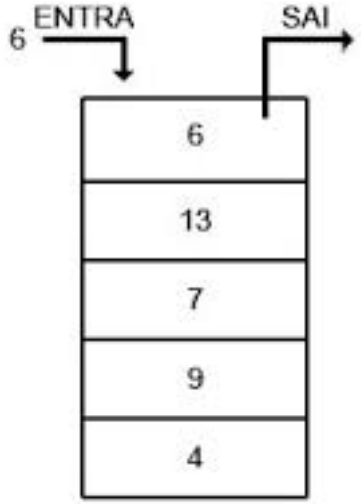
Universidade Federal da Paraíba
Centro de Informática
Departamento de Informática

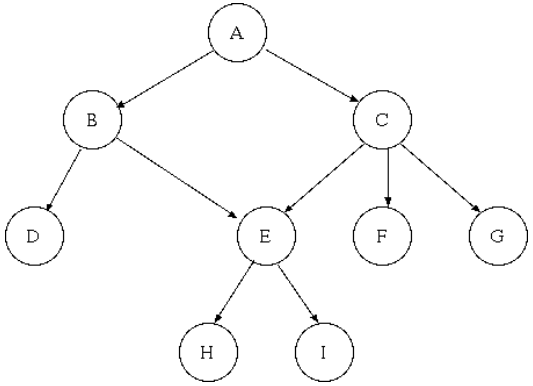
Estrutura de Dados
Introdução às ED

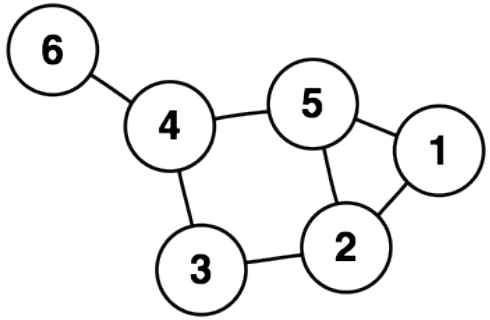
- Tiago Maritan
- tiago@ci.ufpb.br

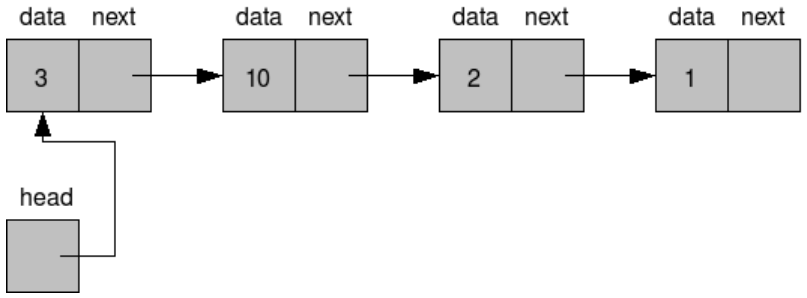
Estrutura de Dados: o que é?

- É o ramo da Ciências da Computação que estuda os diversos **mecanismos de organização de dados** para atender aos diferentes requisitos de processamento.



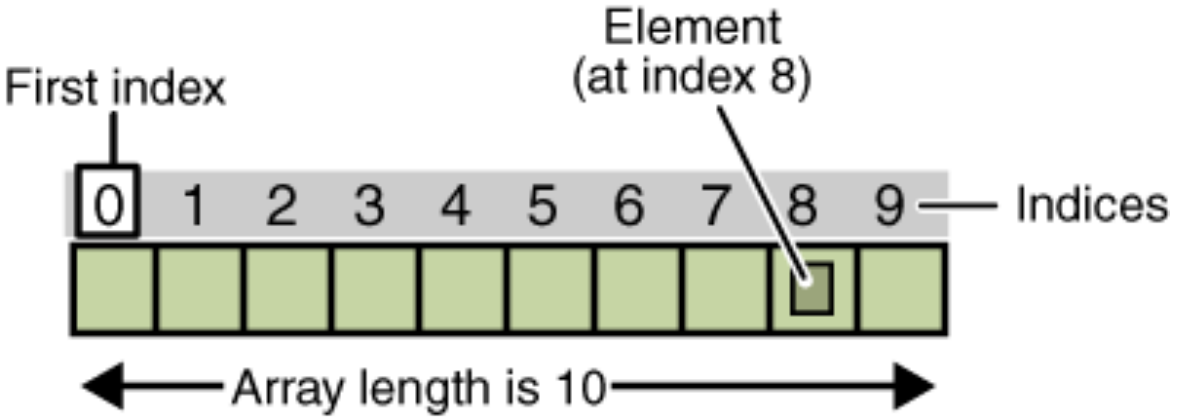


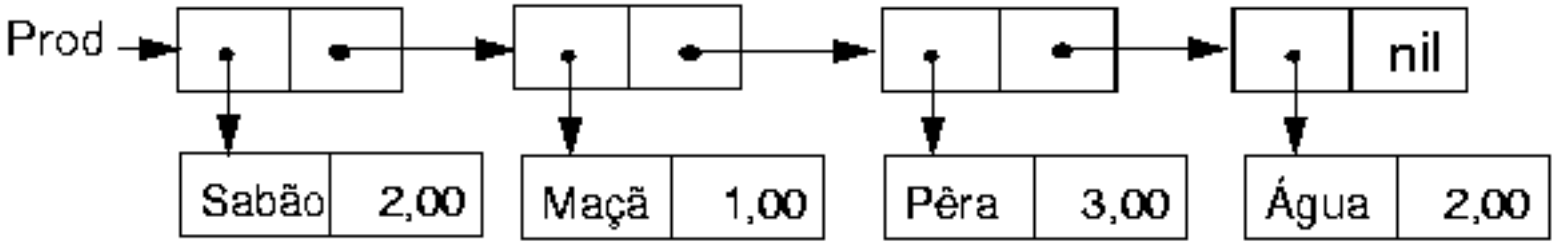




Ex 1: Como armazenar uma lista de clientes num sistema computacional ???

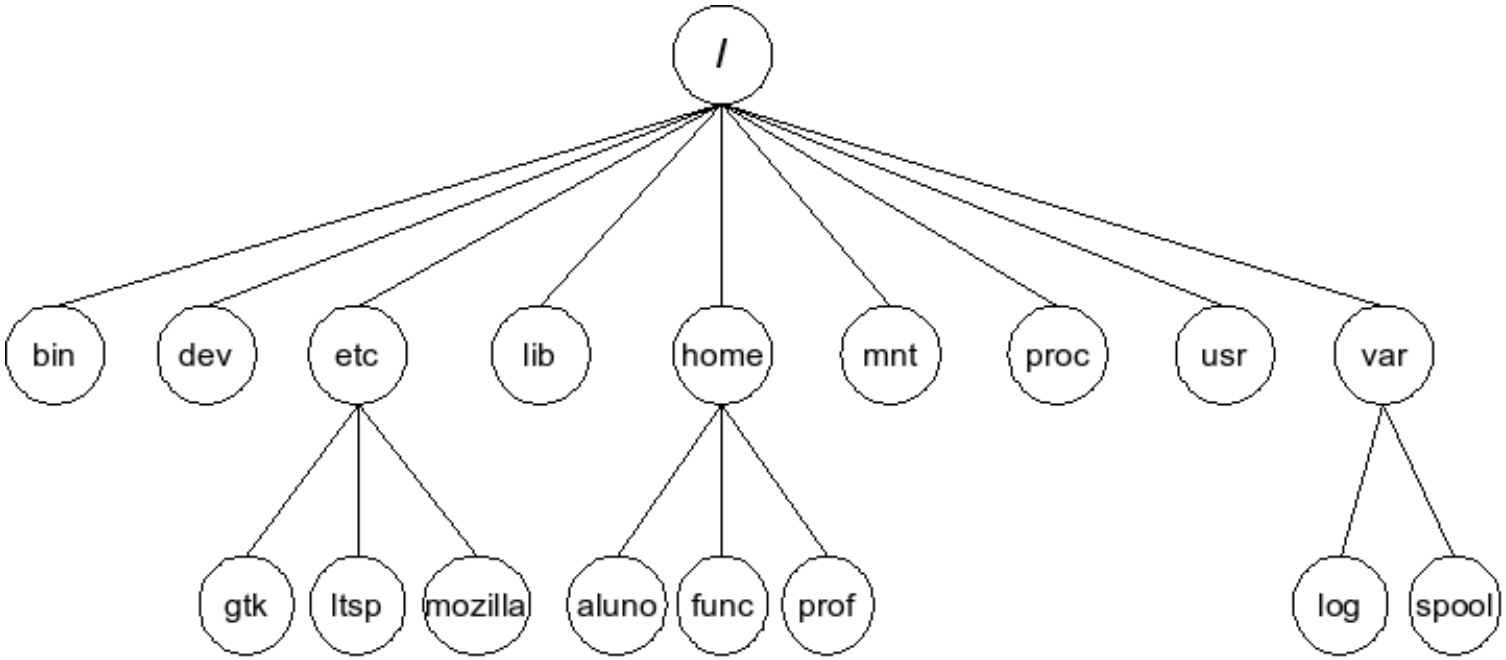
- Através de estruturas do tipo **Array (vetor) ou Lista.**





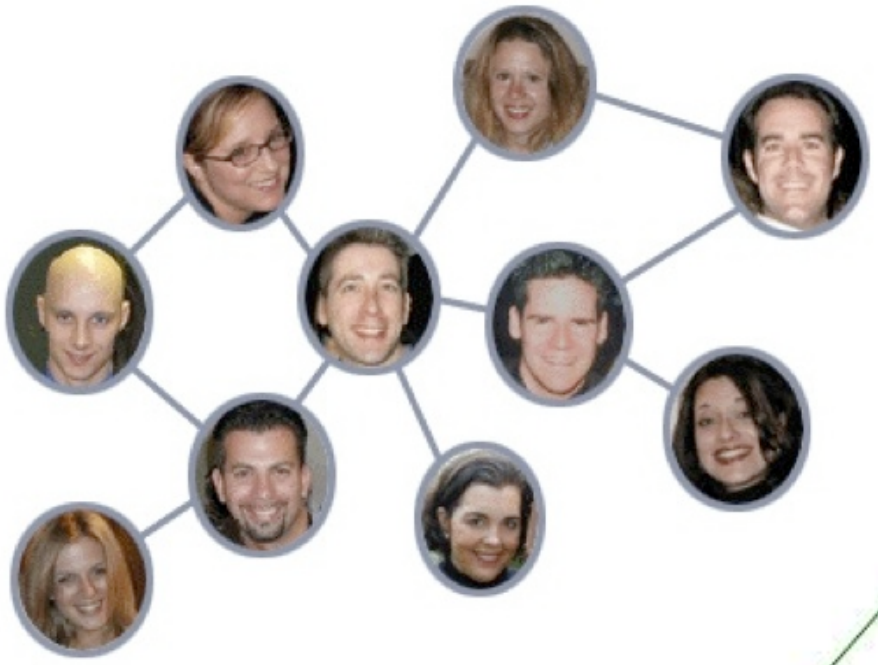
Ex 2: Como armazenar os dados de um sistema de arquivos num computador?

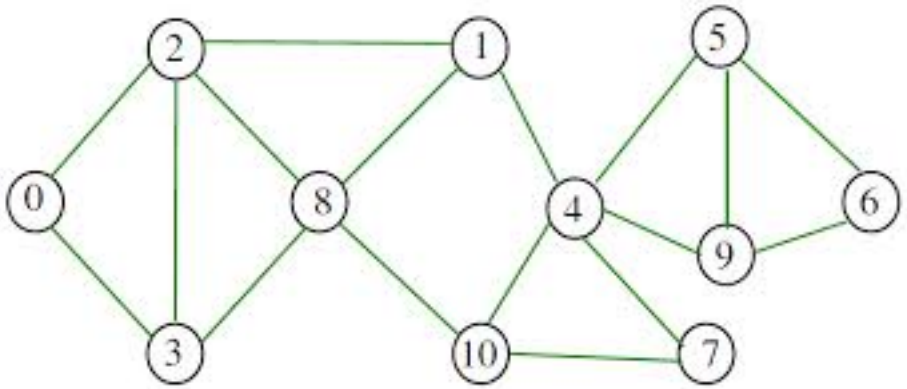
- Através de estruturas do tipo **Árvore**!



Ex 3: Como estruturar o relacionamentos entre dados dos usuários numa rede social?

- Através de estruturas do tipo **Grafo!**



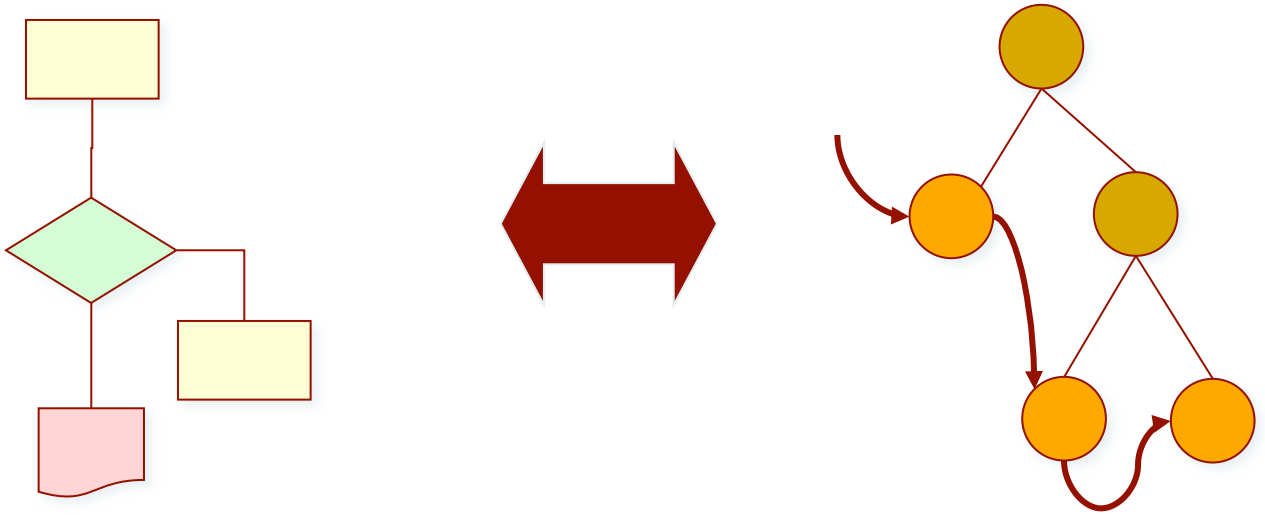


Relação entre Algoritmos e ED

- O objetivo da nossa disciplina será analisar as melhores alternativas para **manipular eficientemente os dados de um sistema computacional**.

Algoritmos e ED – relação íntima

Algoritmos geralmente trabalham sobre **Estrutura de Dados**

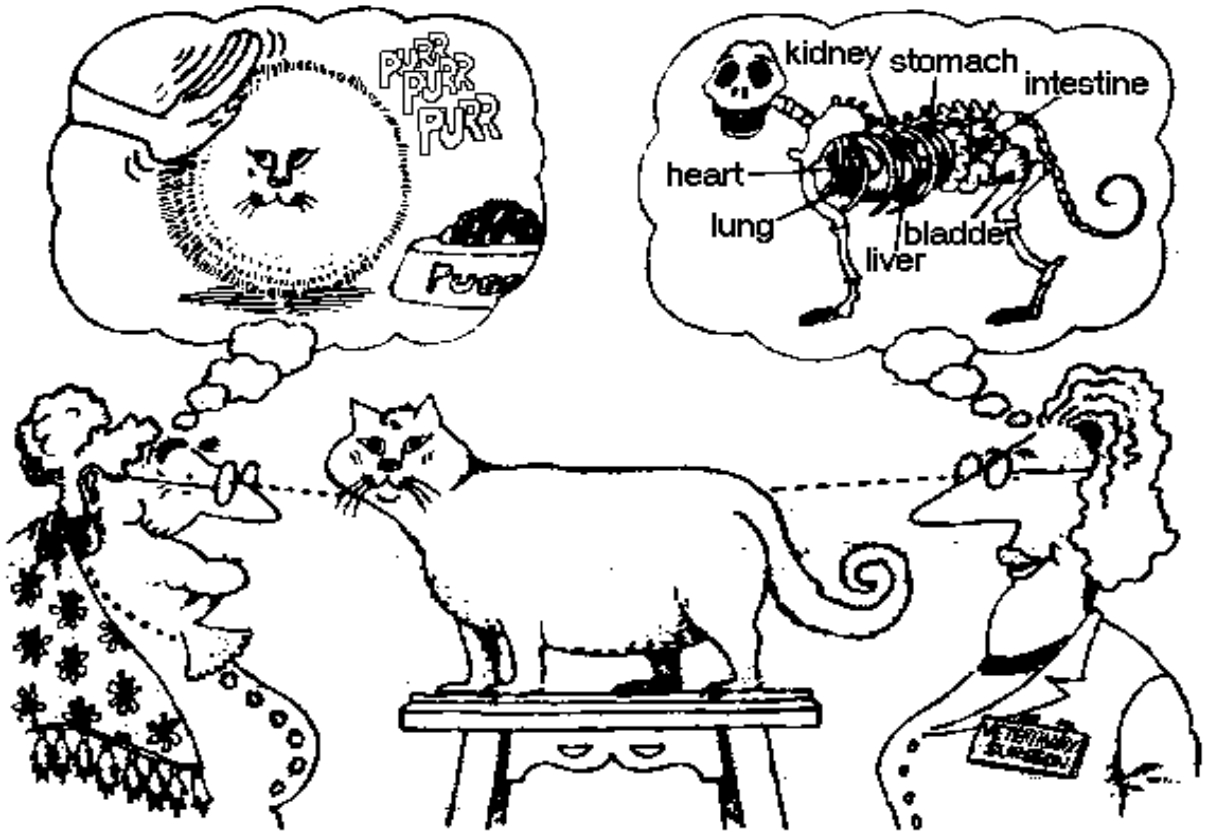


A escolha de uma **ED adequada** pode simplificar a implementação do algoritmo para um dado problema

Abstração e
Tipos Abstratos de Dados

Abstração e Tipos Abstratos de Dados

- Abstração:



Copyritght 1991 © Grady Booch

Abstração e Tipos Abstratos de Dados

- Abstração:
  - Permite-nos **dominar a complexidade do mundo real**
  - **Modela-se uma entidade** ou **conceito de forma simplificada... focando apenas no que é interessante para resolver o problema.**

Abstração e Tipos Abstratos de Dados

- Definição de Abstração:

> "*Uma abstração é uma visualização ou uma representação de uma entidade que* ***inclui somente os atributos de importância*** *em um* ***contexto particular.***"
>
> [Robert Sebesta]

Tipos Abstratos de Dados (TADs)

- **Tipo Abstrato de Dados (TAD)** é um **modelo** de **estruturação dos dados** que especifica:
  - O tipo dos dados armazenados;
  - As operações definidas sobre esses dados;
  - Os tipos de parâmetros dessas operações

TAD = encapsula dados + operações

TADs vs ED

- TAD:
  - **Conceito matemático básico** que define o tipo de dado;
  - Define **O QUE** cada operação faz, **não como faz**.
  - **Não** se relaciona com **detalhes de implementação**
    - Ex: Eficiência de espaço (uso memória) e tempo

- **Estrutura de Dados**
  - Método particular para **implementar um TAD** em uma LP.

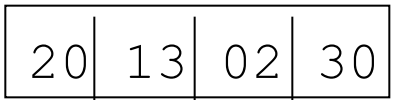
Operações em TADs

- Operações típicas de TADs:
  1. **Criação de um TAD;**
  2. **Inclusões e remoções de dados no TAD**;
  3. **Percurso no TAD** (varre todos os dados armazenados);
  4. **Busca** (busca algum dado dentro da estrutura).

Ex: Lista de números inteiros

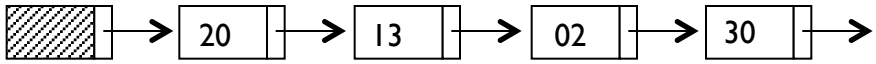
- Operações
  - Faz Lista Vazia
  - Insere número no começo da lista
  - Remove de uma posição i

Implementação por Vetores:




```
void Insere(int x, Lista L) {
   for(i=0;...) {...}
   L[0] = x;
}
```

Implementação por Listas Encadeadas




```
void Insere(int x, Lista L) {
    p = CriaNovaCelula(x);
    L.primeiro = p;
    ...
}
```

Programa usuário do TAD:

```
int main() {
   Lista L;
   int x;

   x = 20;

FazListaVazia(L);
   Insere(x,L);
   ...
}
```

Tipos Abstratos de Dados

- Um TAD **`pilha`** implementado em C como um array:

```
typedef struct st_pilha {
   int topo;
   int dados[MAX];
} pilha;

void cria_pilha (pilha *){...}
int empilha (pilha *, int) {...}
int desempilha (pilha *, int *) {...}
int ta_vazia (pilha) {...}
int topo (pilha, int *) {...}
int limpa (pilha *) {...}
```

Representação ou estrutura do TAD pilha

Operações do TAD pilha

Tipos Abstratos de Dados

- Um cliente para o TAD **`pilha`**:

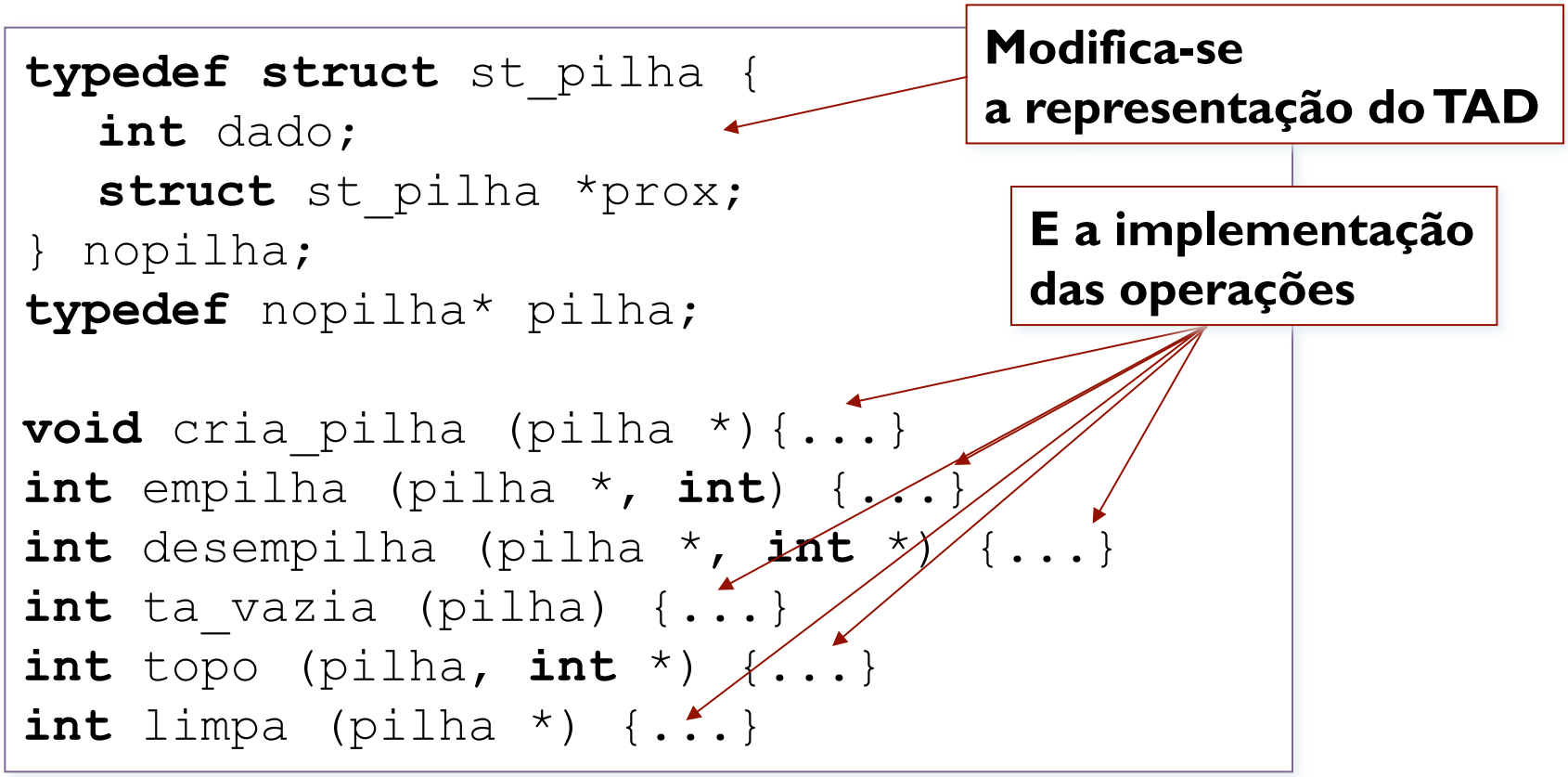
```
int main() {
   pilha p;
   int x;

   cria_pilha(&p);
   if (ta_vazia(p)) {
       empilha(&p, 1);
       empilha(&p, 3);
   }
   topo(&p, &x);
   if (x == 3)
       desempilha (&p, &y);
   limpe (&p);
}
```

Em respeito ao encapsulamento, não há acessos à representação da pilha nos clientes, mas apenas às suas operações.

Tipos Abstratos de Dados

- A implementação muda para lista encadeada:



Tipos Abstratos de Dados

- O cliente, no entanto, não se altera:

```
int main() {
   pilha p;
   int x;

   cria_pilha(&p);
   if (ta_vazia(p)) {
       empilha(&p, 1.2f);
       empilha(&p, 3.0f);
   }
   topo(&p, &x);
   if (x == 3.0f)
       desempilha (&p, &y);
   limpe (&p);
}
```

Tipos Abstratos de Dados

- Entretanto, se o encapsulamento fosse violado...

```
int main() {
   pilha p;
   float x;

   cria_pilha(&p);
   if (p.topo == -1) {
      empilha(&p, 1.2f);
      empilha(&p, 3.0f);
   }
   x = p.dados[p.topo];
   if (x == 3.0f)
      desempilha (&p, &y);
   limpe (&p);
}
```

Acessos à estrutura do TAD violam o encapsulamento e aumentam o acoplamento entre o TAD e seus clientes

Estes acessos à representação do TAD são específicos para a implementação com arranjos. Não servem para a nova implementação com lista!

Tipos Abstratos de Dados

- O conjunto de operações públicas que um TAD oferece é chamado de **Interface do TAD**.

- Mudanças nos clientes só são necessárias se interface for modificada

Tipos Abstratos de Dados

- Suporte para TADs em C:
  - C permite que se implementem TADs em **módulos**.
  - Arquivo de cabecalho (extensão .h):
    - Define tipo e os protótipos das funções.
  - Arquivo de programa (extensão .c):
    - Implementa as operações

- **O encapsulamento de C é limitado**
  - Clientes podem acessar os campos da estrutura do tipo.

Exemplo de um módulo pilha em C

- Arquivo de cabeçallho (pilha.h):

```
#define MAX 100

typedef struct st_pilha {
   int topo;
   float dados[MAX];
} pilha;

// protótipos das funções
void cria_pilha (pilha *);
int empilha (pilha *, float);
int desempilha (pilha *, float *);
int ta_vazia (pilha);
int topo (pilha, float*);
int limpa (pilha *);
```

Exemplo de um módulo pilha em C

- Arquivo de programa (pilha.c):

```
#include "pilha.h"

void cria_pilha (pilha *p) {
   ...
}

int empilha (pilha *p, float e) {
   ...
}

... //outras funções
```

Tipos Abstratos de Dados

- Suporte para TADs em Java e C++:
  - Implementa TADs usando **classes** (arquivo .java ou .cpp).
  - Encapsula dados + operações
  - **Pode-se controlar o nível de acesso aos atributos da estrutrura de dados**

Exemplo de classe Pilha em Java

```
public class Pilha {
   private int topo;
   private float[] dados;

   public void empilha (float e) {
       ...
   }

   public float desempilha() {
       ...
   }
   ... // outros métodos

} //fim da classe
```

Exemplo de classe Pilha em C++

```
class Pilha {
   private:
      int topo;
      float dados[MAX];
   public:
      void empilha (float e) {
         ...
      }

      float desempilha() {
         ...
      }
      ... // outros métodos
} //fim da classe
```

Tipos Abstratos de Dados

- Benefícios da programação com TADs:
  - **Organiza programa em unidades lógicas**
    - Podem ser compiladas e mantidas em separado.
  - **Encapsulamento**
    - Promove a independência entre o TAD e seus clientes.
  - **Aumento da confiabilidade**
    - Clientes não têm acesso direto à representação do TAD, a não ser por suas operações.

Universidade Federal da Paraíba
Centro de Informática
Departamento de Informática

Estrutura de Dados
Introdução às ED

- Tiago Maritan
- tiago@ci.ufpb.br