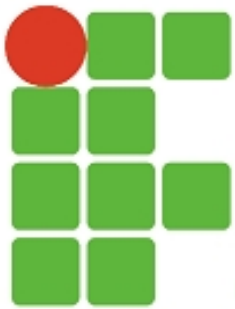


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARAÍBA
Campus Campina Grande

8

Curso Engenharia da Computação

Programação e Estruturas de Dados

Árvores – Fundamentos e Implementações

Prof. César Rocha
cesarocha@ifpb.edu.br

Objetivos

- *Explorar os conceitos fundamentais acerca do uso de **árvores** utilizando a linguagem C*
  - *Organização e implementação, características, vantagens, desvantagens, regras de utilização, operações básicas e algoritmos de implementação*
- *Será abordada ainda uma implementação de árvore bastante conhecida: **árvore binária***
- *Este módulo será utilizado como referência na entrega dos futuros projetos*
  - *Implementação das estruturas e algoritmos, criação das **bibliotecas** e práticas de laboratório*

Parte I : Fundamentos

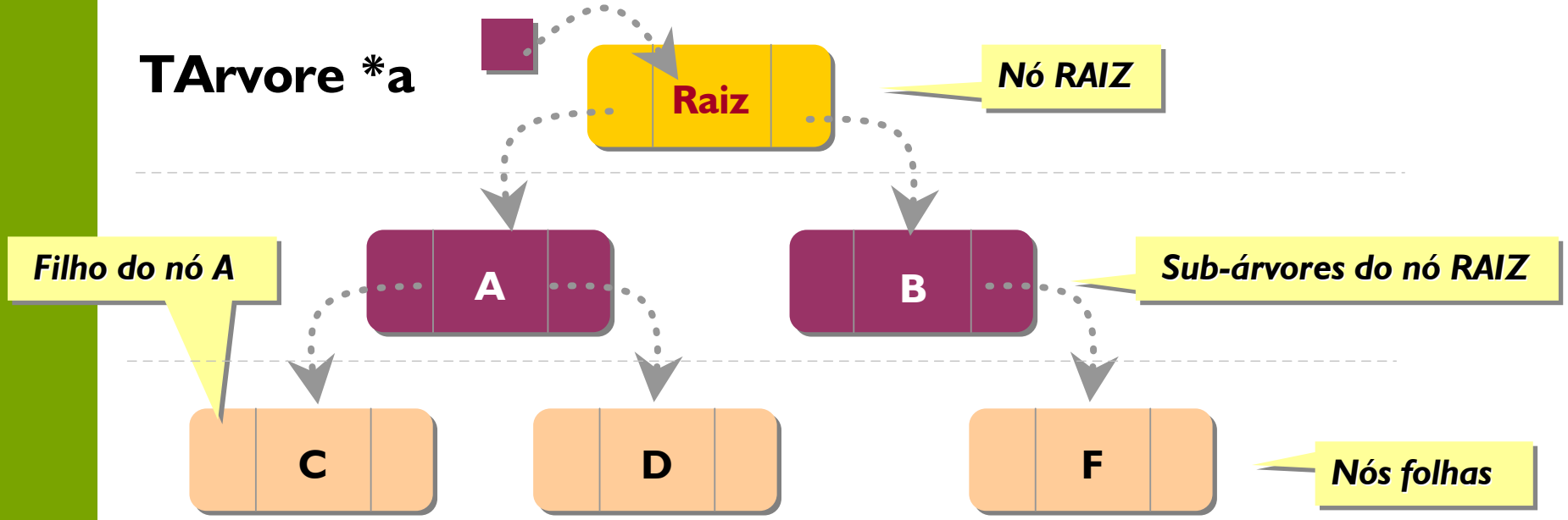
Motivação

- *Ao longo de todo o curso, temos examinado estruturas de dados que podem ser chamadas de unidimensionais ou lineares*
  - *Por exemplo: TADs que usam vetores e todas as listas.*
- *A importância destes tipos de estruturas é incontestável, porém...*
  - *elas não são adequadas para representarmos dados que devem ser dispostos de maneira hierárquica*
  - *não faz sentido, por exemplo, representar uma árvore de diretórios de um sistema operacional utilizando um vetor linear e pré-dimensionado*

Conceitos

- *Uma **árvore** consiste de uma estrutura não-linear que representa uma relação de hierarquia*
- *Uma árvore é composta por um conjunto de **nós***
  - *Existe um nó R, chamado de nó **raiz***
    - *Este nó contém zero ou mais sub-árvores, cujas raízes são ligadas diretamente à R*
  - *Os nós raízes das sub-árvores são ditos **filhos** do nó R*
  - *Nós que não têm filhos são chamados de **folhas***
- *É bastante comum desenhar as árvores com o nó raiz para cima e os nós folhas para baixo*

Graficamente



- *Representamos explicitamente a direção dos ponteiros, na figura acima:*
  - *Eles apontam sempre do pai para os filhos*
- *Diferente do exemplo acima, podemos ter nós contendo mais de duas sub-árvores*

Grau

- *O número de sub-árvores presentes em um determinado nó indicará o* ***Grau*** *desse nó*
  - *Qual o grau de cada nó mostrado anteriormente?*
- *Conceito de Grau Máximo:*
  - *número máximo de sub-árvores que o nó pode ser raiz*
- *Nós que não tem grau, são chamados de nós folhas*
- *Para identificar os nós de uma estrutura, usamos a relação de hierarquia existente em uma árvore genealógica*
  - *Nó Filho, Nó pai, Nó neto, Nó irmão, ...*

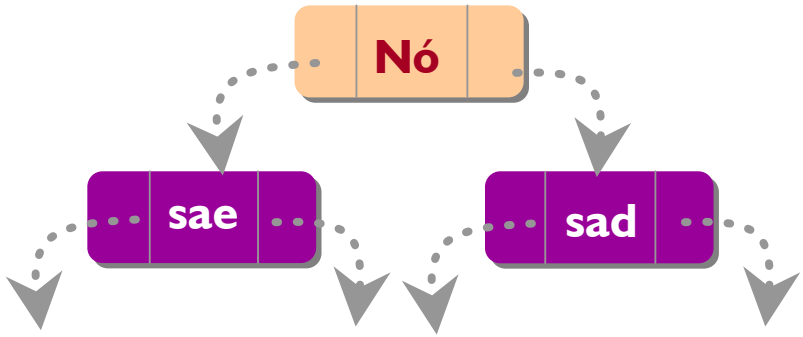
Nível e Altura

- Conceito de **Nível**:
  - Representa a distância de um nó até a raiz
    - Qual o nível do nó F (slide 6)?
- Importante: o nó que apresentar o maior nível fornecerá a **altura** de uma árvore
  - Só existe um caminho da raiz para qualquer nó
- O número de filhos permitido em cada nó e as informações armazenadas nestes nós e que diferenciam os diversos tipos de árvores existentes

Parte 2 :
Árvores Binárias

Conceitos

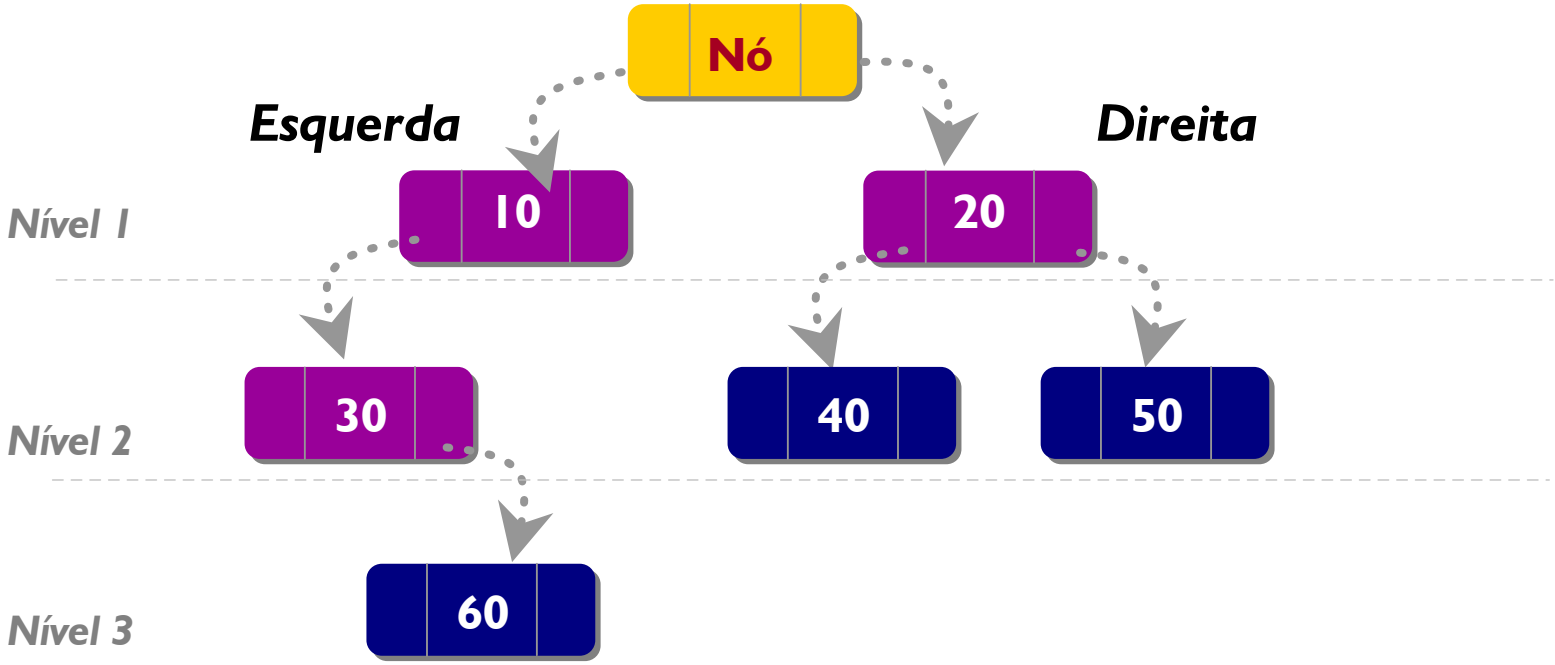
- *Em uma árvore binária, todos os seus nós possuem, no máximo,* ***duas sub-árvores***
  - *Cada nó pode ter zero, um ou dois filhos.*
  - *E ainda, é uma árvore de grau máximo igual a dois.*
- *As duas sub-árvores possíveis em cada nó são também denominadas de:*
  - *sub-árvore esquerda (sae).*
  - *sub-árvore direita (sad).*



- *Árvore binária completa*
  - *Árvore em que cada nó possui dois filhos (exceto os nós folhas, é claro).*

Graficamente

- *Observe:*



- *Onde:*

Grau máximo = 2

Raízes das sub-árvores

Nós folhas (grau = 0)

Pense um pouco...

- *O que você acha que seria necessário para implementar uma biblioteca de um novo TAD que representasse uma árvore binária?*
  1. *uma estrutura que guarde: dado e ponteiros*
  2. *dois apontadores indicando: esquerda e a direita do nó*

```
/* estruturação */
typedef struct arv {
   int info;
   struct arv *esq;
   struct arv *dir;
}no;
```

```
/* arvorebin.h */
Inicialização da árvore
Criar nó raiz
Árvore vazia
Imprimir a árvore
Inserir filho esquerdo
Inserir filho direito
Remover um determinado nó
```

Considerações

- **Importante!**
  - Uma árvore é representada pelo endereço do nó raiz
    - Uma árvore vazia é representada pelo valor NULL
  - O nível de uma árvore que contém apenas o nó raiz é 0
  - A altura de uma árvore vazia é negativa e vale -1
- Percurso:
  - Um percurso define a ordem em que os nós de uma árvore serão processados
  - Trabalharemos com três tipos de percursos: Pré-ordem, In-ordem e Pós-ordem

Tipos de percursos

- *Algoritmos de percursos em árvores:*
  1. *Pré-ordem*
     - *Utiliza a raiz*
     - *Percorre a sub-árvore esquerda*
     - *Percorre a sub-árvore direita*
  2. *In-ordem*
     - *Percorre a sub-árvore esquerda*
     - *Utiliza a raiz*
     - *Percorre a sub-árvore direita*
  3. *Pós-ordem*
     - *Percorre a sub-árvore esquerda*
     - *Percorre a sub-árvore direita*
     - *Utiliza a raiz*

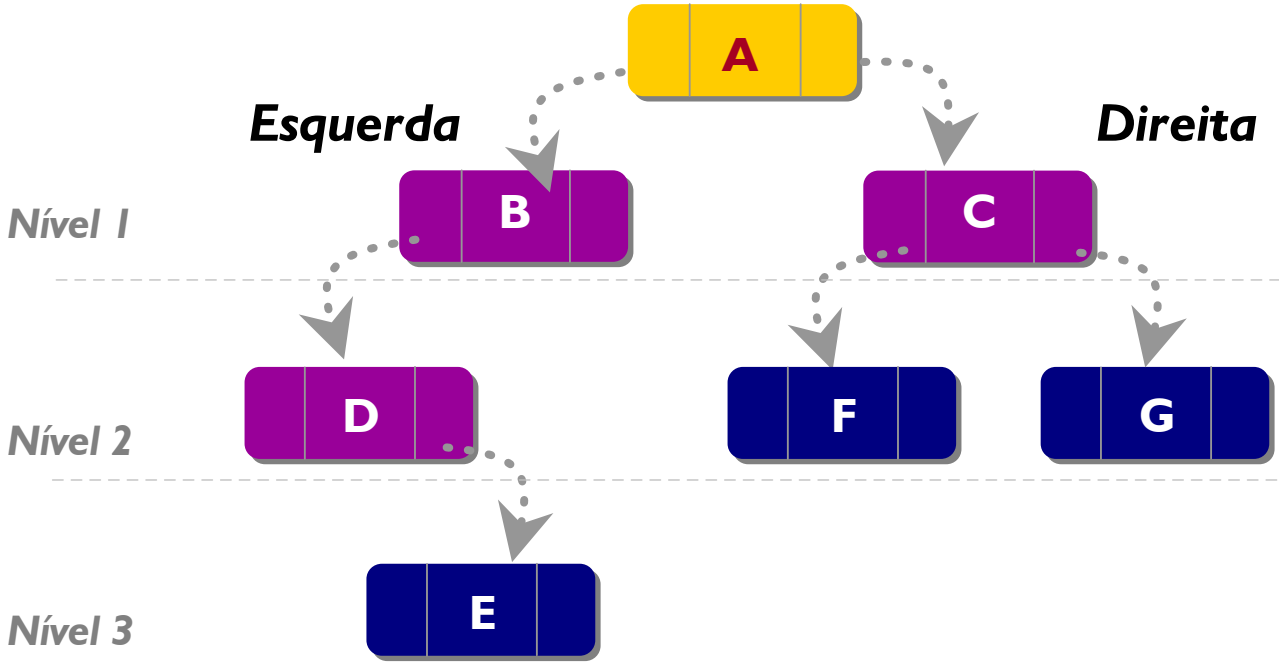
```
// Algoritmo recursivo pre-ordem
void preordem( arvore arv ) {
  if(!vazia( arv ){
    printf( "%d", arv->info );
    preordem( arv->esq );
    preordem( arv->dir );
  }
}
```

```
// Algoritmo recursivo in-ordem
void inordem( arvore arv ) {
  if(!vazia( arv ){
    inordem( arv->esq );
    printf( "%d", arv->info );
    inordem( arv->dir );
  }
}
```

```
// Algoritmo recursivo pos-ordem
void posordem( arvore arv ) {
  if(!vazia( arv ){
    posordem( arv->esq );
    posordem( arv->dir );
    printf( "%d", arv->info );
  }
}
```

Tipos de percursos

- *Observe:*



- *Percursos:*

Grau máximo = 2

Pré-ordem: ***A, B, D, E, C, F, G***
In-ordem: ***D, E, B, A, F, C, G***
Pós-ordem: ***E, D, B, F, G, C, A***

Algoritmos em C

- *O que deverá ser feito pelo aluno:*
  - *Escolha e instalação do ambiente a ser trabalhado no laboratório*
  - *Modelagem deste TAD (dados e operações)*
  - *Implementação dos algoritmos de operações básicas vistos em sala de aula na linguagem C*
  - *Utilização das regras de modelagem vistas no módulo anterior (criação de bibliotecas) e modularização*
  - *Implantação de código legível e bem documentado*
  - *Nomes de variáveis condizentes com o problema*
  - *Prática de laboratório*

Para um bom aproveitamento:

- *O aluno deve identificar a relação entre TAD (biblioteca e modularização) com a implementação da fila no código!*
- *Resolva todas as questões da* ***prática de laboratório de árvores***
- *Procure o professor ou monitor da disciplina e questione conceitos, listas, etc.*
- *Não deixe para codificar tudo e acumular assunto para a primeira avaliação.*
  - *Este é apenas um dos assuntos abordados na prova!*