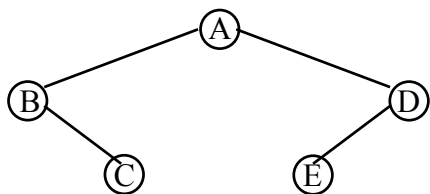


Instituto Luterano de Ensino Superior de Ji-Paraná
Curso Bacharelado em Informática
Estrutura de Dados I
Prof.: José Luiz A. Duizith



Pré: A B C D E

In : B C A E D

Pos: C B E D A

·
·
·
E
C
B
D
A

Usando Pilha

Procedimento Pré_Ordem (nodo) { s/ recursividade }

{ escreve o caminhamento em Pré-Ordem a partir de um nodo da árvore raiz }

Se (Nodo \diamond NIL) *{ raiz não vazia }*

Então Cria_pilha(Pilha)

Empilha (Pilha, Nodo)

Enquanto (Não Pilha_Vazia (Pilha))

Faça Aux \leftarrow Consulta_Pilha (Pilha)

Desempilha (Pilha)

Visita (Aux)

Se (Aux \uparrow .Dir \diamond NIL)

Entao Empilha (Pilha, Aux \uparrow .Dir)

Se (Aux \uparrow .Esq \diamond NIL)

Entao Empilha (Pilha, Aux \uparrow .Esq)

Destroi_Pilha (Pilha)

{ fazer Pós-Ordem sem recursividade. Poderá cair na prova !!! }

Procedimento Pré_Ordem (Nodo) { c/ recursividade }

Se (Nodo \diamond Nil)

Entao Visita (Nodo)

Pre_Ordem (Nodo \uparrow .Esq)

Pre_Ordem (Nodo \uparrow .Dir)

Procedimento In_Ordem (Nodo) { c/ recursividade }Se (Nodo \neq NIL)

Entao In_Ordem (Nodo↑.Esq)

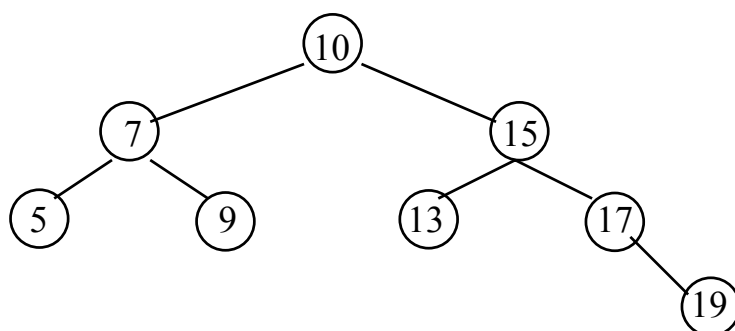
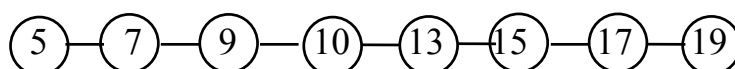
Visita (Nodo)

In_Ordem (Nodo↑.Dir)

TIPOS DE ÁRVORES

1º Árvore binária de busca (ou de pesquisa)

definição: Uma árvore binária é uma árvore binária de busca se para cada nó N da árvore for verdade que $SEM \leq N \leq SD_N$, isto é, todos os nodos da subárvore à esquerda precedem a raiz e a raiz precede todos os nodos da subárvore à direita.

SE_NSD_N: Subárvore à esquerda/direita de N.Nº acessos : $\log_2 N$ **Procedimento Cria_Arvore_Binaria (Raiz)**Raiz \leftarrow NIL**Procedimento Arvore_Vazia (Raiz) : Lógico**Logico \leftarrow (Raiz = NIL)**Procedimento Destroi_Arvore_Binaria (Raiz)**

Se (Não_Arvore_Vazia (Raiz))

Entao Pos_Ordem (Raiz)

Raiz \leftarrow NIL

{ visita (nodo) na Pos-Ordem seria
desaloque (nodo) }

Funcao Consulta (Nodo, Valor, Aux) : Lógico

{ Retorna verdadeiro, se “valor” existe na árvore de raiz nodo; caso contrário, retorna falso }

```

Se (Arvore_vazia (Nodo))
Entao Logico ← Falso
Senao Aux ← Nodo
    Logico ← Falso
    Enquanto (Não Lógico) E (Aux <> NIL)
        Faça
            Se (Valor = Aux↑.Dado)
                Entao Logico ← Verdadeiro
            Senao Se (Valor < Aux↑.Dado)
                Entao Aux ← Aux↑.Esq
            Senao Aux ← Aux↑.Dir

```

Funcao Encontra_Pai (Nodo, Valor) : Aux_Pai

{ Caso valor exista na árvore de raiz “nodo”, retorna o pai do nodo; caso o nodo encontrado for a raiz, então retorna NIL, se não existir, retorna o possível pai do nodo }

```

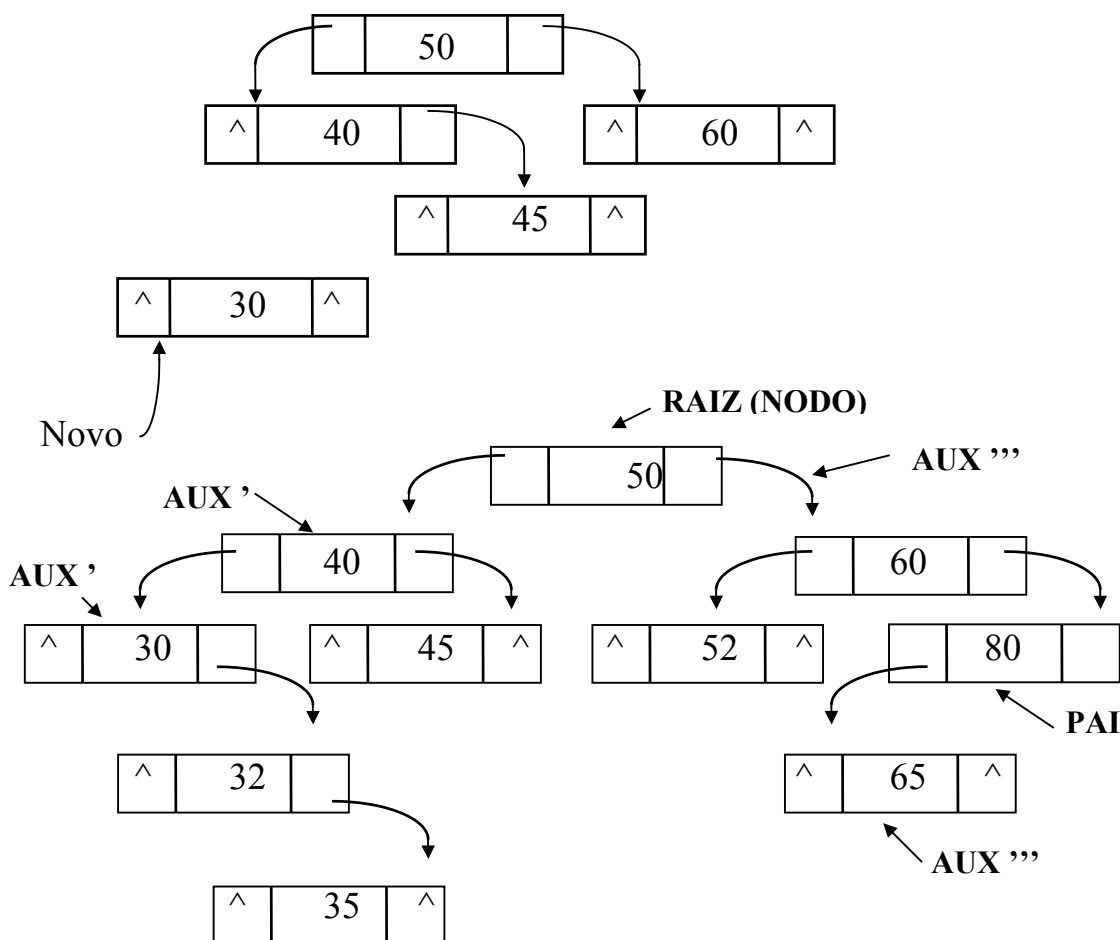
Se (Arvore_Vazia (Nodo))
Entao Aux_Pai ← NIL
Senao Se (Valor = Nodo↑.Dado)      {nodo que possui o valor é a raiz}
    Entao Aux_Pai ← NIL
    Senao Aux ← Nodo
        Achou ← Falso
        Enquanto ((Não Achou) E (Aux <> NIL))
            Faça
                Aux.Pai ← Aux
                Se (Valor < Aux↑.Dado)
                    Entao Aux ← Aux↑.Esq
                Senao Aux ← Aux↑.Dir
                Se (Aux <> NIL) E (Aux↑.Dado = Valor)
                    Entao Achou ← Verdadeiro

```

Procedimento Insere (Nodo, Valor)

```

Se (Arvore_Vazia (Nodo))
Entao Aloque (Novo)
    Nodo↑.Esq ← NIL
    Nodo↑.Dado ← Valor
    Nodo↑.Dir ← NIL
    Nodo ← Novo
Senao Se (Consulta (Nodo, Valor, Aux))
    Entao ERRO { valor já existe }
Senao Aloque (Novo)
    Novo↑.Esq ← NIL
    Novo↑.Dado ← Valor
    Novo↑.Dir ← NIL
    Pai ← Encontra_Pai (Nodo, Valor)
    Se (Valor < Pai↑.Dado)
        Entao Pai↑.Esq ← Novo
    Senao Pai↑.Dir ← Novo
  
```



REMOÇÃO:

- **Nodo Folha:**
 - **acerta subárvore do pai**
 - **remove a folha**

- **Nodo com uma subárvore**
 - **acerta o pai com a subárvore restante**
 - **remove o nodo**

- **Nodo com duas subárvores:**
 - **encontrar o predecessor / sucessor**
 - **substituir o nodo pelo predecessor / sucessor**
 - **remove o predecessor / sucessor**

Procedimento Remove (Nodo, Valor)

Se (Arvore_Vazia (Nodo))

Entao ERRO1

Senao Se (Não Consulta (Nodo, Valor, Aux))

Entao ERRO2

Senao Se ((Aux↑.Esq = NIL) E (Aux↑.Dir = NIL)) *{nodo folha}*

Entao Pai ← Encontra_Pai (Nodo,Valor) *{achou o pai}*

Se (Pai = NIL) *{nodo a remover é a raiz}*

Entao Nodo ← NIL

Senao Se (Valor < Pai↑.Dado)

Entao Pai↑.Esq ← NIL

Senao Pai↑.Dir ← NIL

Senao Se (aux↑.Esq = NIL) *{nodo c/ subárvore à direita}*

Entao Pai ← Encontra_Pai (Nodo,Valor)

Se (Pai = NIL) *{nodo a remover é a raiz}*

Entao Nodo ← Aux↑.Dir

Senao Se (Valor < Pai↑.Dado)

Entao Pai↑.Esq ← Aux↑.Dir

Senao Pai↑.Dir ← Aux↑.Dir

Senao Se (Aux↑.Dir = NIL) *{nodo c/ subárvore a esquerda}*

Entao Pai ← Encontra_Pai (nodo,valor)

Se (Pai = NIL) *{nodo a remover é a raiz}*

Entao Raiz ← Aux↑.Esq

Senao Se (Valor < Pai↑.Dado)

Entao Pai↑.Esq ← Aux↑.Esq

Senao Pai↑.Dir ← Aux↑.Esq

Senao Pred ← Predecessor (Aux,Valor)

{nodo com 2 subárvores}

Temp ← Pred↑.Dado

Remove (Aux,Temp)

Aux↑.Dado ← Temp

Aux ← NIL

Se (Aux <> NIL)

Entao DESALOQUE (Aux)

Função Predecessor (Nodo) : Aux_Pred

```

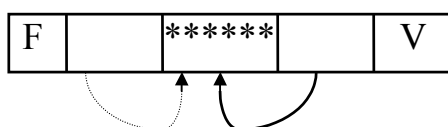
Se (Nodo = NIL)
Entao  Aux ← NIL
Senao  Aux ← Nodo↑.Esq
      Enquanto (Aux↑.Dir <> NIL)
      Faça Aux ← Aux↑.Dir
      Aux_Pred ← Aux

```

Árvore Amarrada (ou costurada - Alinhavadas (“Threaded Trees”))

Amarras → referências a outros nós da árvore.

Se subárvore da esq. De um nó é vazia, então a referência da esq. Deste nó aponta p/ o nó predecessor a este, com respeito a uma dada ordem de caminhamento. De forma similar, se a subárvore da direita de um nó é vazia, então a referência da direita do nó aponta p/ o nó que lhe sucede na mesma ordem de caminhamento.



Amarrada Ref. Estruturada

Quando a árvore não for vazia, a ref. Da esq. Do nó descritor apontará p/ a raiz da árvore. Se a ordem de caminhamento p/ a qual a árvore for amarrada é a ordem central, então o nó descritor será o predecessor e sucessor do 1º e do último nó da árvore, respectivamente. Isto impõem, na realidade, uma característica circular à árvore.

→ Muitos nodos possuem referências nulas, tais referências podem servir para indicar uma determinada ordem de caminhamento.

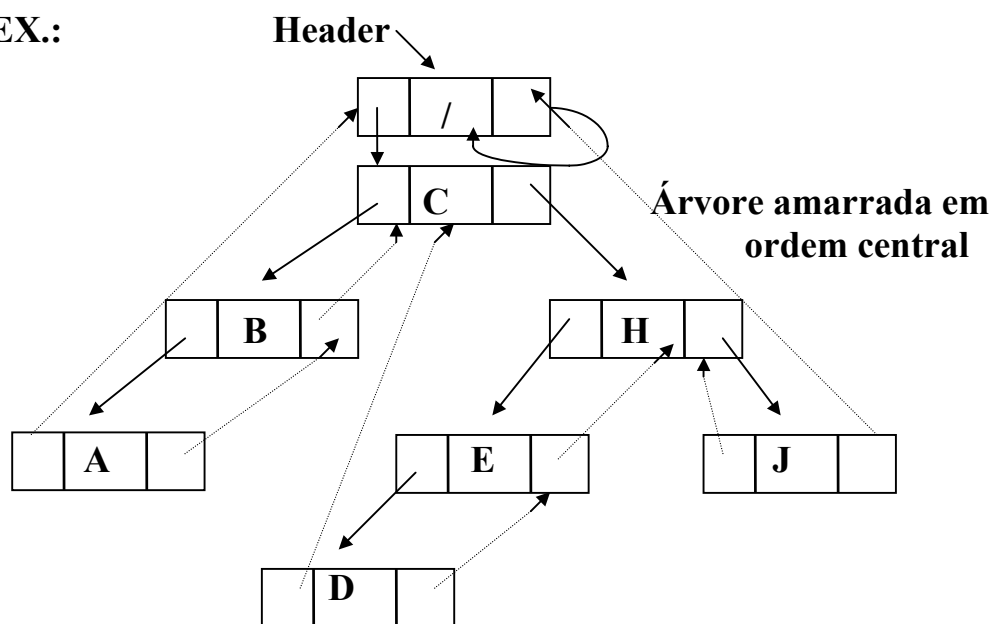
→ Para distinguir uma ligação estrutural de uma amarrada acrescentam-se 2 campos lógicos adicionais.



Se ‘verdadeiro’ indicam que a ligação correspondente é estrutural, senão é amarrada.

→ Para facilitar operações de novos nodos é criado um descritor que possui a mesma estrutura que os nodos da árvore.

EX.:



→ Esquerda de **C** é cheia, então a ligação é estrutural, então o **ELIG** é Verdadeiro.

→ A Direita de **B**, é pontilhada, então a ligação é amarrada e o **DLIG** é Falso.

Procedimento Cria_Arvore_Amarrada (Header)

ALOQUE (Header)

Se (Header = NIL)

Entao ERRO

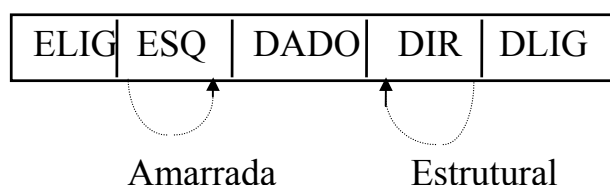
Senao Header↑.Dir ← Header

Header↑.DLIG ← Verdadeiro { *ligação estrutural*}

Header↑.Esq ← Header

Header↑.ELIG ← Falso

HEADER

**Funcao Sucessor_Central (Nodo) : Sucessor**

```

Sucessor ← Nodo↑.Dir
Se (Nodo↑.DLIG)
Entao
    Enquanto (Sucessor↑.ELIG)
        Faça Sucessor ← Sucessor↑.Esq
Se (Sucessor↑.Dir = Sucessor)    {achou o header}
Entao Sucessor ← NIL

```

Funcao Predecessor_Central (Nodo) : Predecessor

```

Predecessor ← Nodo↑.Esq
Se (Predecessor↑.Dir = Predecessor) {achou o header}
Entao Predecessor ← NIL

```

Procedimento Insere_Esq (Nodo, Aux)

{ insere o nodo AUX a esquerda do nodo; se a subarvore esquerda do nodo for vazia, AUX torna-se sua subarvore esquerda, caso contrário AUX é inserido entre NODO e sera subarvore esquerda }

```

Aux↑.Esq ← Nodo↑.Esq
Aux↑.ELIG ← Nodo↑.ELIG
Nodo↑.Esq ← Aux
Nodo↑.ELIG ← Verdadeiro
Aux↑.Dir ← Nodo
Aux↑.DLIG ← Falso
Se (Aux↑.ELIG)    {nodo possui subarvore esquerda}
Entao
    Pred ← Predecessor_central (Aux)
    Pred↑.Dir ← Aux
    Pred↑.DLIG ← Falso

```

→ Ao inserir um nodo a esquerda, ele deve ser estruturada.

→ ELIG de Aux verdadeiro → entao é estrutural e então possui nodos abaixo.

Procedimento Insere_Dir (Nodo, Aux)

```

Aux↑.Dir ← Nodo↑.Dir
Aux↑.DLIG ← Nodo↑.DLIG
Nodo↑.Dir ← Aux
Nodo↑.DLIG ← Verdadeiro
Aux↑.Esq ← Nodo
Aux↑.ELIG ← Falso
Se (Aux↑.DLIG)           {nodo possui Sub-direita}
Entao
    | Suc ← Aux↑.Dir
    | Suc↑.Esq ← Aux
    | Suc↑.ELIG ← Falso

```

Procedimento In_Ordem (Header)

{caminhamento em In_Ordem s/ recursividade, sem estruturas adicionais}

```

Se (Header↑.Esq = Header)  {árvore vazia}
Entao ERRO
Senao Aux ← Sucessor_Central (Header)
    Enquanto (Aux <> NIL)
        Faça
            | Visita (AUX)
            | Aux ← Sucessor_Central (Aux)

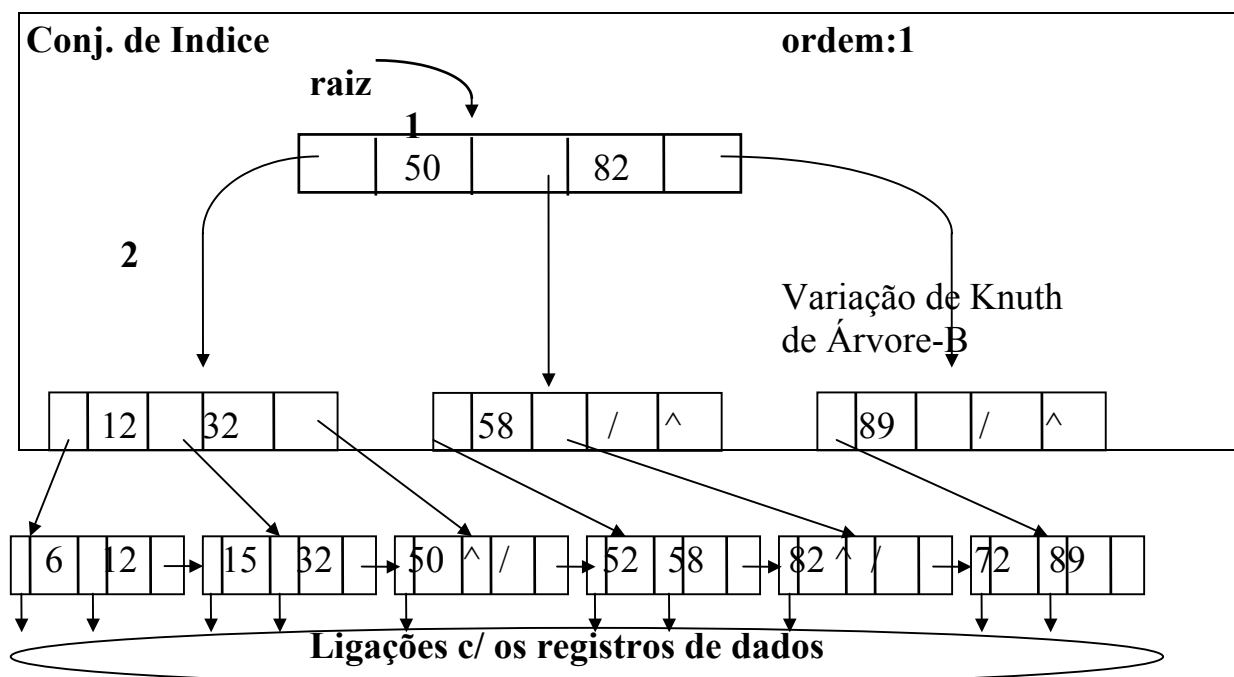
```

Árvore Tipo-B (“B Tree”)

Def.: é uma árvore que possui as seguintes características:

- a) todos os nodos folhas estão no mesmo nível
- b) um nodo que possua “k” filhos, possui “k-1” chaves
- c) todo nodo, exceto a raiz, possui no máximo $k / 2$ filhos

Aplicação: manipulação de índices de arquivos



→ Achar onde está o reg do arq. 50 (1) → percorre a esquerda (2) → 50 é maior que 32? Então percorre a direita (3)

Uma árvore-B de ordem n tem no mínimo n , mas não mais do que $2n$ valores de dados (chaves) em cada nodo → se um nodo possui k valores possui $k + 1$ ligações (apontadores)

Ex.:

ESQ	DADO1	MEIO	DADO2	DIR
-----	-------	------	-------	-----