**UNISINOS - UNIVERSIDADE DO VALE DO RIO DOS SINOS**
**CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS (C6/6) – Curso: Informática**

# *LABORATÓRIO II – AULA : Balanceamento de Árvore Binárias*

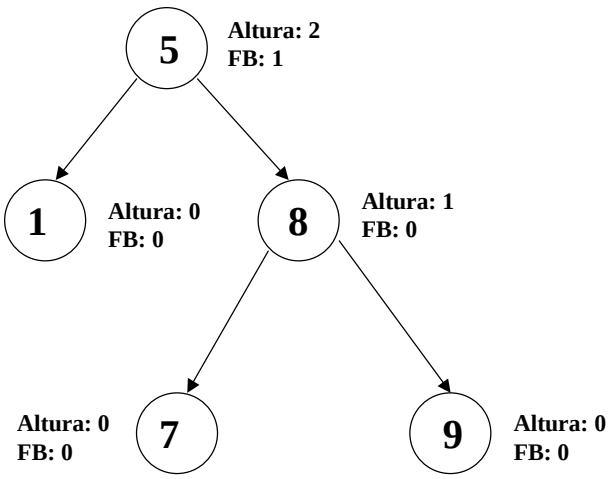
**Disciplina: Linguagem de Programação PASCAL**
**Professor responsável:** ***Fernando Santos Osório***
**Semestre: 99/2**
**Horário: 21 e 41**

**E-mail:** ***osorio@exatas.unisinos.br***
**Web:**
***http://www.inf.unisinos.br/~osorio/lab2.html***
**Xerox : *Pasta 54* – LAB. II (Xerox do C6/6)**

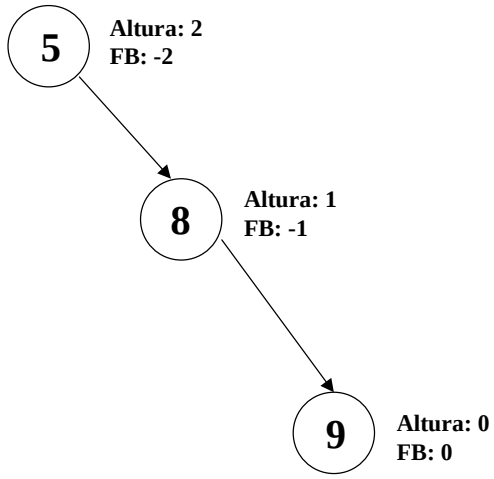
## ÁRVORES BINÁRIAS DE PESQUISA
## Balanceamento – Árvores AVL

Uma árvore binária balanceada é aquela em que dado um nodo qualquer da árvore a sua subárvore à esquerda possui uma altura igual (ou no máximo um nível superior ou inferior) em relação a sua subárvore à direita. Dois conceitos importantes são: altura de um certo nodo, e o fator de balanceamento (diferença entre a altura da subárvore esquerda em relação a subárvore direita). Exemplos:

**BALANCEADA**




**NÃO BALANCEADA / DEGENERADA**




**FB (Nodo) := Altura ( Nodo^.ArvEsq ) – Altura ( Nodo^.ArvDir);**

**Se FB = 0 ou FB = 1 ou FB = -1**
**Então este nodo está *BALANCEADO***
**Senão este nodo está *DESBALANCEADO***

Rotina recursiva de cálculo da altura de um nodo qualquer:

```
Function Altura (Nodo: Ptr_Nodo_AB):Integer;
Var
  Alt_Esq, Alt_Dir : Integer;
Begin
  If Nodo = NIL
  Then Altura := -1
  Else Begin
         Alt_Esq := Altura (Nodo^.ArvEsq);
         Alt_Dir := Altura (Nodo^.ArvDir);
         If Alt_Esq > Alt_Dir
         Then Altura := 1 + Alt_Esq
         Else Altura := 1 + Alt_Dir;
       End;
End;
```

As árvores AVL, cujo nome foi obtido a partir das iniciais de seus criadores russos – Adel'son-Vel'skii e Landis, são uma implementação de um algoritmo para a criação de árvores binárias balanceadas. A inserção de nodos nas árvores AVL é realizada através de uma rotina como a que segue:

```
Procedure Insere_AVL(Var Raiz: Ptr_Nodo_AB; Dado: Tipo_Dado);
Var
  Pivo, Novo : Ptr_Nodo_AB;
Begin
     Novo := Insere_ArvBin (Raiz, Dado);
     Pivo := Acha_Pivo (Novo);
     If Pivo <> NIL
     Then Balancear_AVL (Raiz, Pivo);
End;
```

**Insere_ArvBin** => Inserção em árvore binária ordenada. É mesma rotina que já foi estudada anteriormente, apenas com a diferença que esta rotina devolve um ponteiro para o novo nodo que foi inserido.

**Acha_Pivo** => Acha um nodo "pivô" da árvore que esteja desbalanceado, ou retorna NIL. Calcula a diferença das alturas, determinando o FB, se FB < -1 ou FB > +1 é ele!

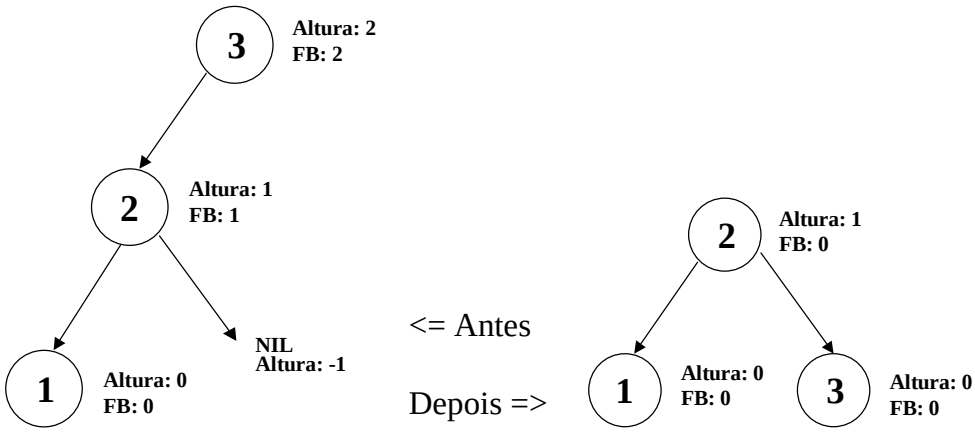
```
Function Acha_Pivo(Nodo: Ptr_Nodo_AB):Ptr_Nodo_AB;
Var
  Ptr : Ptr_Nodo_AB;
Begin
  Ptr := Nodo;
  While (Ptr <> NIL) And
        (abs(Altura(Ptr^.ArvDir) – Altura(Ptr^.ArvEsq)) < 2)
  Do Ptr := Ptr^.Pai;
  Acha_Pivo := Ptr;
End;
```

**Balancear_AVL** => Realiza o "balanceamento" da árvore que consiste basicamente em realizar rotações dos nodos da árvore (de acordo com o tipo de caso em que se encontra – vide mais abaixo).

## ➢ BALANCEAMENTO DE ÁRVORES AVL:

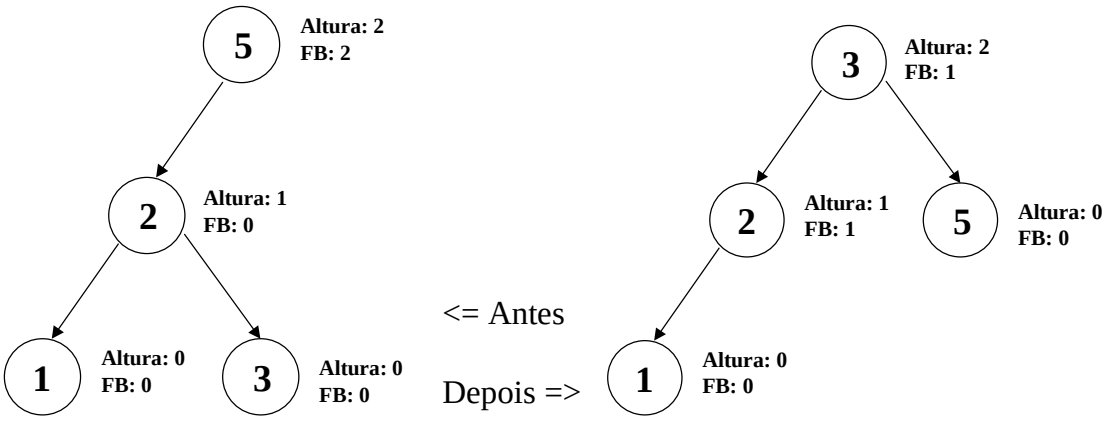
Uma vez que foi detectado o desbalanceamento da árvore após uma inserção, e que foi determinado o nodo pivô (onde o valor absoluto de FB é maior que 1), passamos para a realização do procedimento de balanceamento da árvore. Este procedimento consiste primeiramente de determinar em qual dos seguintes tipos de casos nos enquadramos:

**Tipo I** : Se a subárvore esquerda é maior que a subárvore direita (FB > 1)
E a subárvore esquerda desta subárvore esquerda é maior que a subárvore direita dela
Então realizar uma **rotação simples para a direita**;



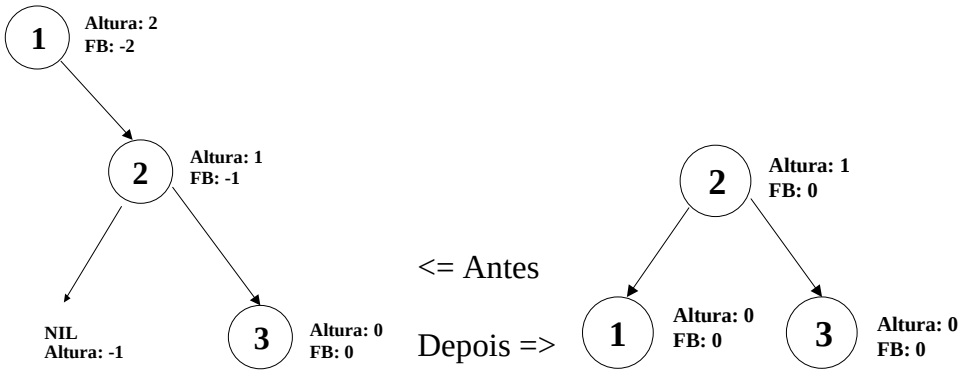

*Procedure Rot_Dir (Var Raiz:Ptr_Nodo_AB; Pivo: Ptr_Nodo_AB);*

**Tipo II** : Se a subárvore esquerda é maior que a subárvore direita (FB > 1)
E a subárvore esquerda desta subárvore esq. é menor ou igual que a subárvore direita
Então realizar uma **rotação dupla para a direita**;



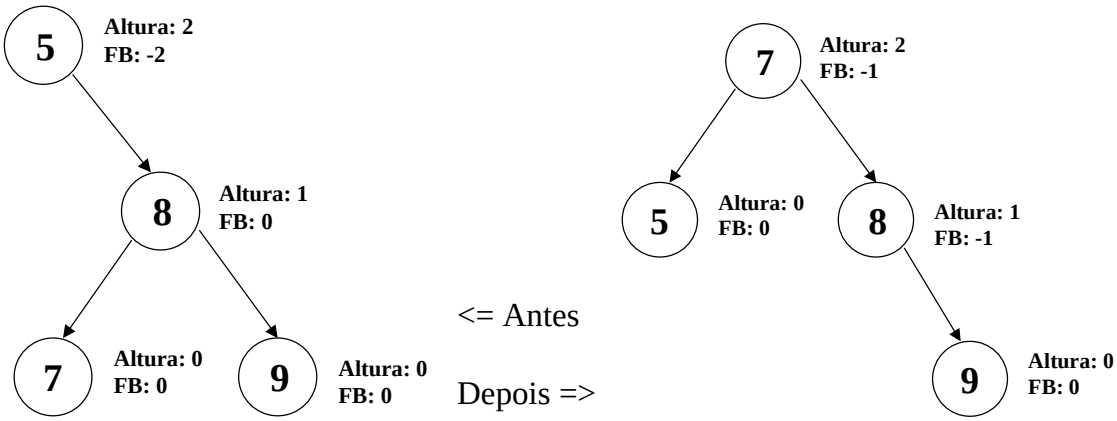

```
Procedure Rot_Dupla_Dir (Var Raiz: Ptr_Nodo_AB; Pivo: Ptr_Nodo_AB);
Begin
   Rot_Esq (Raiz, Pivo^.ArvEsq);
   Rot_Dir(Raiz, Pivo);
End;
```

**Tipo III**: Se a subárvore esquerda é menor que a subárvore direita (FB < -1)
E a subárvore direita desta subárvore direita é maior que a subárvore esquerda dela
Então realizar uma **rotação simples para a esquerda**;




*Procedure Rot_Esq (Var Raiz:Ptr_Nodo_AB; Pivo: Ptr_Nodo_AB);*

**Tipo IV**: Se a subárvore esquerda é menor que a subárvore direita (FB < -1)
E a subárvore direita desta subárvore direita é menor que a subárvore esquerda dela
Então realizar uma **rotação dupla para a esquerda**;




```
Procedure Rot_Dupla_Esq (Var Raiz: Ptr_Nodo_AB; Pivo: Ptr_Nodo_AB);
Begin
    Rot_Dir (Raiz, Pivo^.ArvDir);
    Rot_Esq(Raiz, Pivo);
End;
```

MATERIAIS COMPLEMENTARES:

http://www.inf.unisinos.br/~marcelow/ensino.html (Lab. II)
http://www.inf.unisinos.br/~anibal/ (Prog. II)

# EXERCÍCIOS – *Árvores AVL*

1. Faça um programa para a criação de árvores AVL (inserção ordenada/balanceada e exibição dos dados). Crie as seguintes rotinas genéricas de manipulação de dados:

   Definições elementares:

```
Type
    Tipo_Dado       = Integer;
    Ptr_Nodo_AB     = ^Nodo_Arv_Bin;
    Nodo_Arv_Bin    = Record
                        Dado: Tipo_Dado;
                        Pai: Ptr_Nodo_AB;
                        ArvEsq, ArvDir: Ptr_Nodo_AB;
                        Altura: Integer;  { Opcional }
                      End;
```

   Rotinas:

```
{ Inicializa a árvore binária ordenada do tipo AVL }
Procedure Inicializa_AVL (Var Raiz: Ptr_Nodo_AB);

{ Calcula a altura dos nodos de uma árvore }
Function Altura (Raiz: Ptr_Nodo_AB): Integer;

{ Acha o pivô – nodo desbalanceado que vai sofrer o balanceamento }
Function Acha_Pivo (Nodo: Ptr_Nodo_AB): Ptr_Nodo_AB;

{ Insere um dado na árvore binária ordenada, retorna o ponteiro para o nodo inserido }
Function Insere_ArvBin (Var Raiz: Ptr_Nodo_AB; Dado: Tipo_Dado): Ptr_Nodo_AB;

{ Rotação simples a esquerda em um nodo pivô de uma árvore tipo AVL }
Procedure Rot_Esq (Var Raiz, Pivo: Ptr_Nodo_AB);

{ Rotação simples a direita em um nodo pivô de uma árvore tipo AVL }
Procedure Rot_Dir (Var Raiz, Pivo: Ptr_Nodo_AB);

{ Rotação dupla a esquerda em um nodo pivô de uma árvore tipo AVL }
Procedure Rot_Dupla_Esq (Var Raiz, Pivo: Ptr_Nodo_AB);

{ Rotação dupla a direita em um nodo pivô de uma árvore tipo AVL }
Procedure Rot_Dupla_Dir (Var Raiz, Pivo: Ptr_Nodo_AB);

{ Insere um dado na árvore AVL, retorna o ponteiro para o nodo inserido }
Procedure Balancear_AVL (Var Raiz, Pivo: Ptr_Nodo_AB);

{ Insere um dado na árvore AVL, realiza o balanceamento se necessário }
Function Insere_AVL (Var Raiz: Ptr_Nodo_AB; Dado: Tipo_Dado);

{ Exibe o conteúdo (dados) da árvore AVL de modo hierárquico na tela }
Procedure Exibe_AVL (Raiz: Ptr_Nodo_AB);
```