

Estrutura de Dados Árvore Binária (*Binary Tree*)

- ★ Estrutura de Dados onde :
 - ★ ou é vazia
 - ★ ou é uma raiz e 2 (sub)árvores binárias

Operações básicas:

- ★ criar uma árvore binária vazia;
- ★ verificar se a árvore binária está vazia
- ★ a partir de um novo elemento e duas árvores binárias, construir uma nova árvore binária;
- ★ devolver (ponteiro para) a subárvore esquerda;
- ★ devolver (ponteiro para) a subárvore direita;
- ★ consultar o elemento na raiz;

Representação dinâmica ou Representação estática

Representação Dinâmica para uma Árvore Binária

Lista duplamente ligada

type

```
TipoValor = . . . ;
```

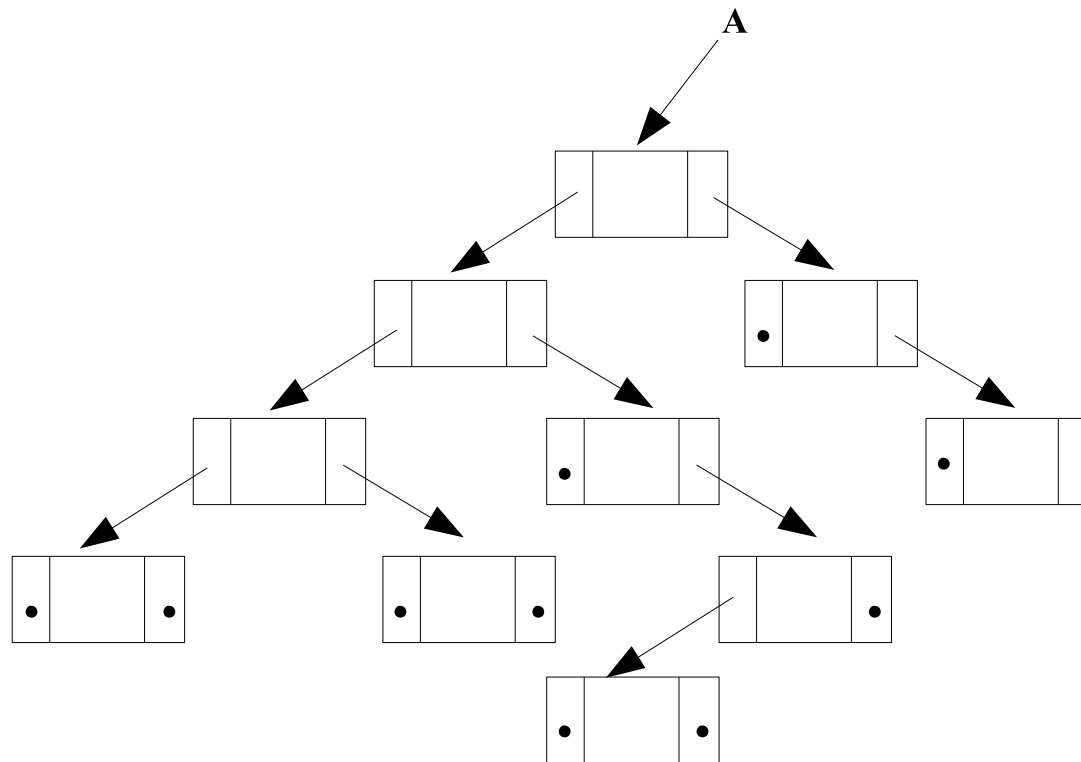
```
ArvBin = ^ Elemento;
```

```
Elemento = record
```

```
    valor : TipoValor;
```

```
    fesq, fdir : ArvBin;
```

```
end;
```



Operadores sobre Árvore Binária

- ★ `function criar : ArvBin;`
cria árvore binária vazia
- ★ `function vazia(A : ArvBin) : boolean;`
devolve verdade sse a árvore binária A estiver vazia
- ★ `function construir(E, D : ArvBin; x : TipoValor) : ArvBin`
dados novo valor e duas árvores binárias: constrói nova árvore binária com raiz no novo elemento
- ★ `function esquerda(A : ArvBin) : ArvBin;`
devolve um ponteiro para a subárvore esquerda de A
- ★ `function direita(A : ArvBin) : ArvBin;`
devolve um ponteiro para a subárvore direita de A
- ★ `function consulta(A : ArvBin) : ArvBin;`
devolve o valor na raiz de A

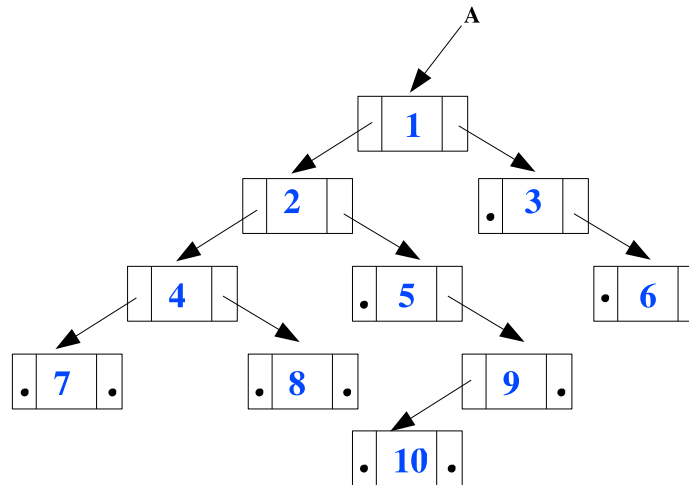
Árvore Binária: operação construtora principal

```
{* constroi uma raiz com valor val, filho esquerdo Te e filho direito Td *}  
{* ou seja, constroi um novo elemento para uma arvore binaria *}
```

```
function construir(Te : ArvBin; val : Item; Td : ArvBin)  
  var T : ArvBin;  
begin  
  new(T);  
  with T^ do  
  begin  
    valor := val;  
    fesq := Te;  
    fdir := Td;  
  end;  
  construir := T;  
end;
```

Visitar uma Árvore Binária

Travessias de uma Árvore Binária



★ Pré-ordem:

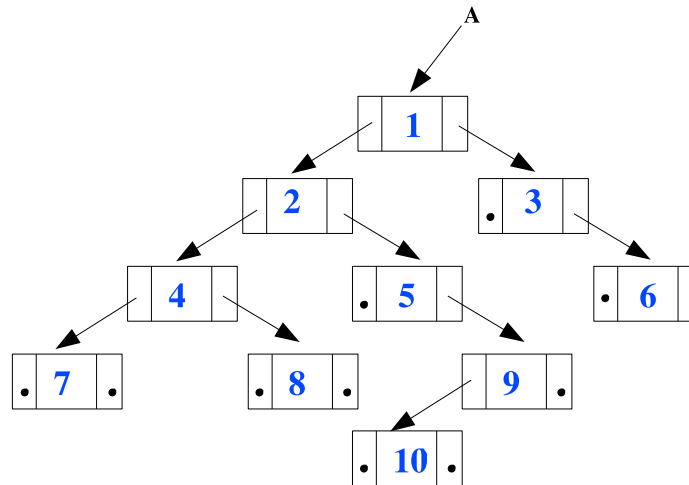
1. raiz;
2. sub-árvore esquerda;
3. sub-árvore direita;

★ Em-ordem:

1. sub-árvore esquerda;
2. raiz;
3. sub-árvore direita;

Visitar uma Árvore Binária

Travessias de uma Árvore Binária



★ Pós-ordem:

1. sub-árvore esquerda;
2. sub-árvore direita;
3. raiz;

★ Por Níveis:

Listar Recorrentemente uma Árvore Binária

Listar uma Árvore Binária Em-ordem

```
procedure listar_emordem(T : ArvBin);
begin
    write('{ ');
    if not vazia(T)
    then begin
        listar_emordem(esquerda(T)); { lista subarvore esquerda }
        escrever(consultar(T)); { escreve o valor desta raiz }
        listar_emordem(direita(T)); { lista subarvore direita }
    end;
    writeln('}'); writeln;
end;
```

Listar **Iterativamente** uma Árvore Binária Em-ordem

```
procedure lista_emordem2(T : ArvBin);
  var
    ptaux : ArvBin;
    P      : Pilha;    { pilha de ponteiros para no de arvore }
    continua, raiz : boolean;
begin
  write('{ ');
  if not vazia(T)
  then begin
    no slide seguinte . . .
    end;{ if not vazia(T)}
  writeln('}'); writeln;
end;
```

Listar **Iterativamente** uma Árvore Binária Em-ordem

```
ptaux := T; continua := true; P := criaP;
while continua do
begin
    while not vazia(ptaux) do { desce tudo para a esquerda }
    begin
        sobrepor(P, ptaux);
        ptaux := ptaux^.esquerda;
    end;
    raiz := true;
    while raiz and (not vaziaP(P)) do
    begin
        escrever(topo(P)); { escreve valor }
        if not vazia( direita(topo(P)) )
        then begin
            ptaux := direita(topo(P));
            raiz := false;
        end;
        remover(P);
    end;{ while raiz }
    if vaziaP(P) and vazia(ptaux) then continua := false;
end;{ while continua }
```

Visitar uma Árvore Binária por Nível

- ★ Utiliza-se uma estrutura de dados Fila, cujo item de informação é capaz de conter uma árvore binária (um ponteiro para) e um inteiro não-negativo (nível)
- ★ Como a visita é feita por níveis, o último nível visitado é o máximo dos níveis

Visitar uma Árvore Binária por Nível

Altura de uma Árvore Binária é o máximo dos seus níveis

```
function alturamax(T : ArvBin) : integer;
var
  nivel : integer;
  Q : FiladeNiveis;
begin
  if not vazia( T )
  then begin
    por(T, nivel, Q);
    while not vaziaF(Q) do
    begin
      frente(Q, T, nivel); {devolve a 1a.arvore e o seu nivel}
      tira(Q);
      if not vazia(T^.esq)
      then poe(T^.esq, nivel+1, Q);
      if not vazia(T^.dir)
      then poe(T->dir, nivel+1, Q);
    end;{ while }
  end;{ if not vazia T }
  alturamax := nivel; { o ultimo nivel = maximo nivel }
end;
```