

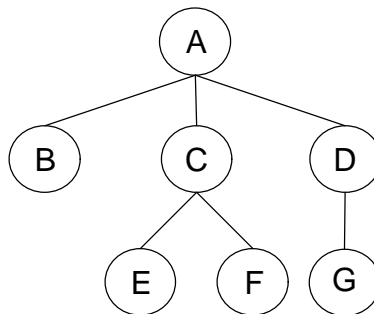
UFSC-CTC-INE  
INE5384 - Estruturas de Dados

## Árvores (2)

Prof. Ronaldo S. Mello  
2002/2

## Modelagem Física de Árvores

- Alternativas de implementação
  - vetor
  - encadeamento



## Árvore como Vetor

- Alternativa geralmente não adotada
  - vetor é bom para manter uma seqüência de dados, não uma hierarquia de dados...
    - operações de inserção, exclusão e certas consultas mais complexas que a alternativa por encadeamento
  - um vetor pode manter os dados de uma árvore para fins de armazenamento secundário

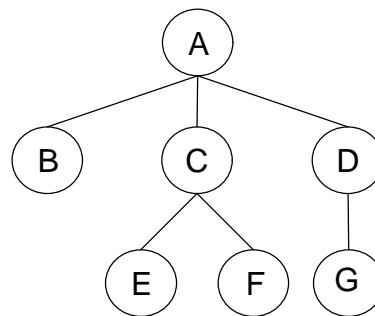
## Árvore como Vetor

- Alternativa 1: (nodo, número de filhos)

A	3	B	0	C	2	E	0	F	0	D	1	G	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---

### Algumas desvantagens:

- dado adicional (número filhos)
- inserção, exclusão:
  - achar a posição correta para I/E um nodo
  - deslocamento
- consulta
  - achar o pai de um nodo
  - achar os filhos de um nodo (usar pilha como ED auxiliar)



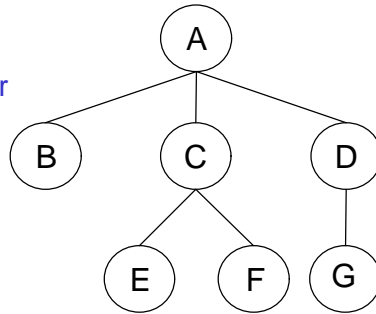
## Árvore como Vetor

- Alternativa 2: (nodo, posição do nodo pai)

0	1	2	3	4	5	6	7	8	9	10	11	12	13
A	-1	B	0	C	0	E	4	F	4	D	0	G	10

### Algumas desvantagens:

- dado adicional (posição pai)
- inserção de um filho:
  - buscar pai para descobrir o valor da sua posição
- exclusão:
  - deslocamento e atualização de referências a nodos pais no vetor
- consulta
  - achar os filhos de um nodo

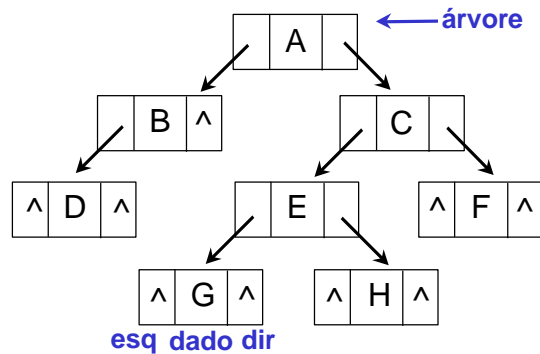


## Árvore como Encadeamento

- Maior flexibilidade de definição da estrutura
- Menor complexidade nas operações de manipulação dos dados

## Árvore como Encadeamento

- Alternativa de representação de árvores binárias



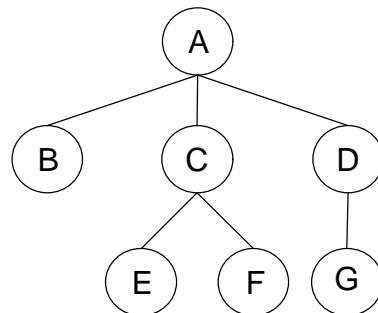
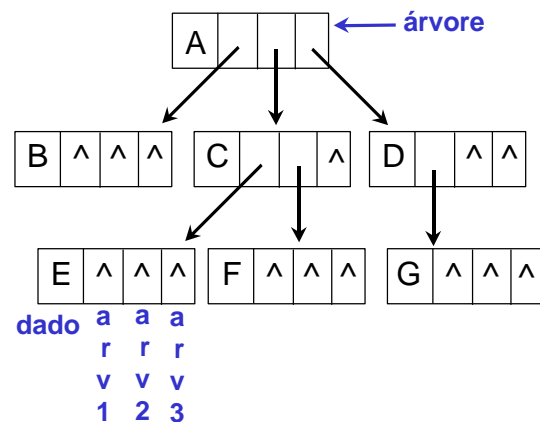
### Operações:

- inserção
- exclusão
- consulta
  - pai de um nodo
  - filhos de um nodo

(usa métodos recursivos de busca em profundidade - implementação simples)

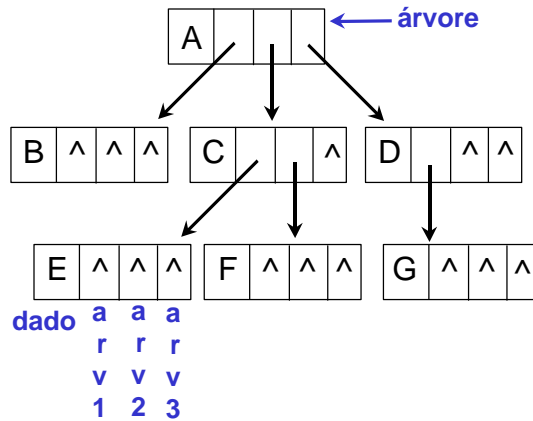
## Árvore como Encadeamento

- Alternativa de representação de árvores N-árias (exemplo para N = 3)



## Árvore como Encadeamento

- Alternativa de representação de árvores N-árias (exemplo para  $N = 3$ )

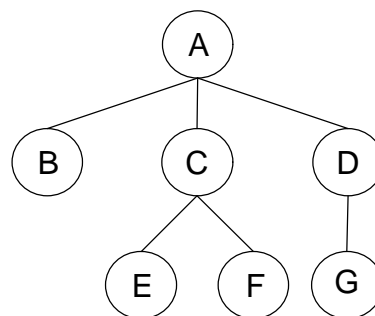
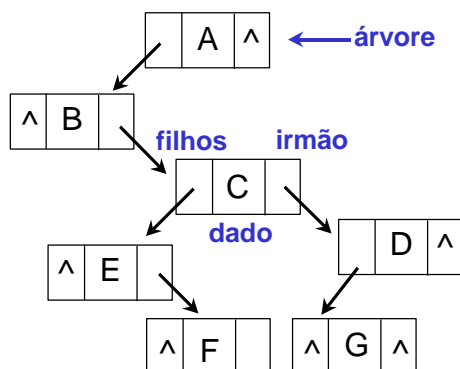


### Desvantagem:

- $N$  atributos de referência, sendo que vários deles podem não ter valor associado

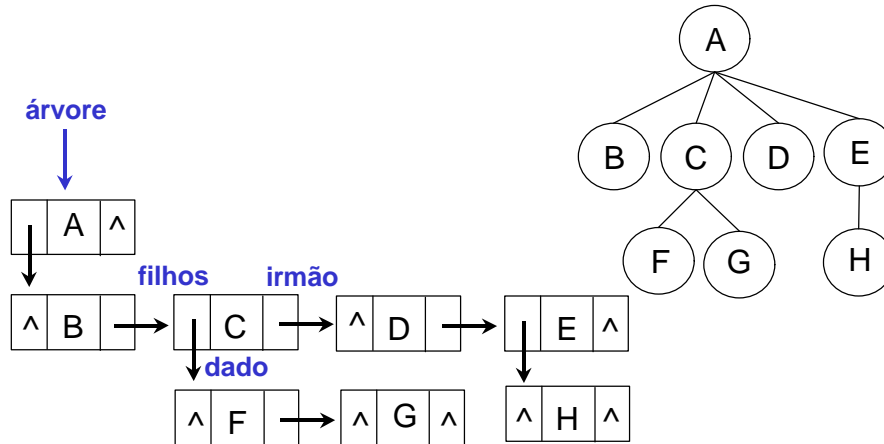
## Árvore como Encadeamento

- Alternativa melhor: representação **estilo árvore binária**



## Árvore como Encadeamento

- Alternativa de representação árvore binária pode ser aplicada a qualquer árvore



## Classe Árvore Binária

Classe ÁrvoreBinária

início

dado     Object;  
 esq     ÁrvoreBinária;  
 dir     ÁrvoreBinária;

construtor ÁrvoreBinária (obj Object);

início

dado ← obj;  
 esq ← NULL;  
 dir ← NULL;

fim;

fim;

## Classe Árvore Binária

Método insereEsq(obj Object)

início

esq ← NEW ÁrvoreBinária(obj);

fim;

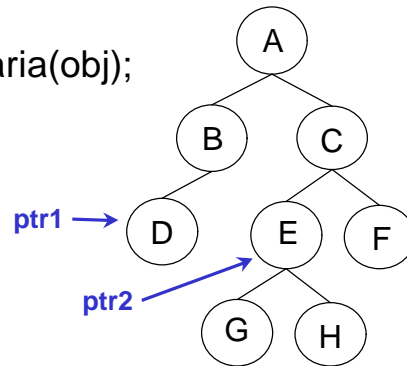
Método removeDir();

início

dir ← NULL;

fim

Exemplos: ptr1.insereEsq(l);  
ptr2.removeDir();



## Classe Árvore Binária

Método buscaPreOrdem();

início

se this ≠ NULL então

início

visita(dado);

esq.buscaPreOrdem();

dir.buscaPreOrdem();

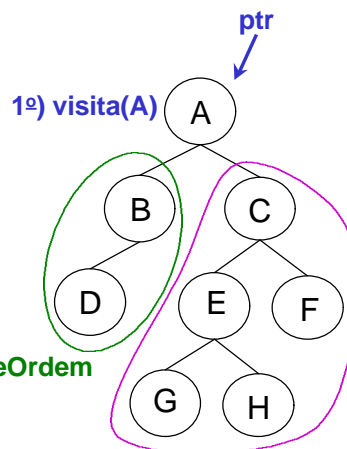
fim;

fim;

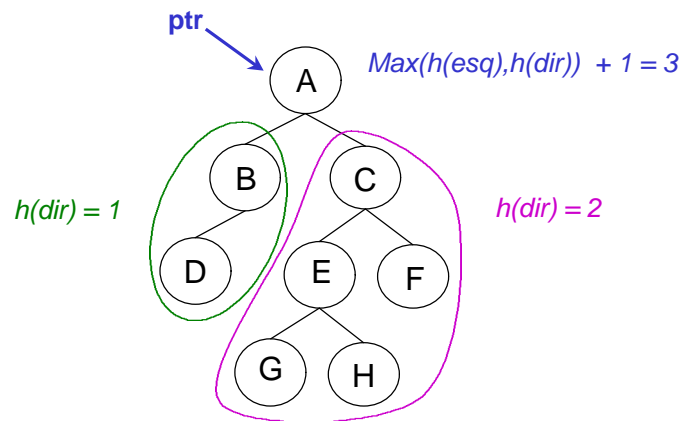
2º) buscaPreOrdem

3º) buscaPreOrdem

Exemplo: ptr.buscaPreOrdem();



## Altura de uma Árvore



Exemplo: `ptr.altura();`

## Classe Árvore Binária

Método `altura()` retorna inteiro;

início

`altEsq, altDir` inteiro;

    se `this = NULL` então retorna -1;

`altEsq` ← `esq.altura()`;

`altDir` ← `dir.altura()`;

    se `altEsq > altDir` então retorna `altEsq + 1`

    senão retorna `altDir + 1`;

fim;



## Métodos Principais

Classe ÁrvoreBinária

início

```
insereEsq(obj Object);  
insereDir(obj Object);  
removeEsq();  
removeDir();  
altura() retorna inteiro;  
ehFolha() retorna booleano;  
buscaLargura();
```

fim;

## Métodos Principais

Classe ÁrvoreBinária

início

```
buscaPreOrdem();  
buscaPosOrdem();  
buscaInOrdem();  
retNumDados() retorna inteiro;  
visita(obj Object);  
encontra(raiz Object) retorna ÁrvoreBinária;  
retFilhos(pai Object) retorna ListaEncadeada;  
retPai(filho Object) retorna Object;
```

fim;

## Métodos Principais

Classe ÁrvoreBinária

início

profundidade(obj Object) retorna inteiro;

insereFilhoEsq(pai Object, filho Object);

insereFilhoDir(pai Object, filho Object);

removeFilhoEsq(pai Object);

removeFilhoEsq(pai Object);

fim;