CHƯƠNG 1: CÂY (TREE)

GV. Ngô Công Thắng
Bộ môn Công nghệ phần mềm
Khoa Công nghệ thông tin
Website: dse.hua.edu.vn/ncthang
Email: ncthang@vnua.edu.vn

Chương 1: Cây (Tree)

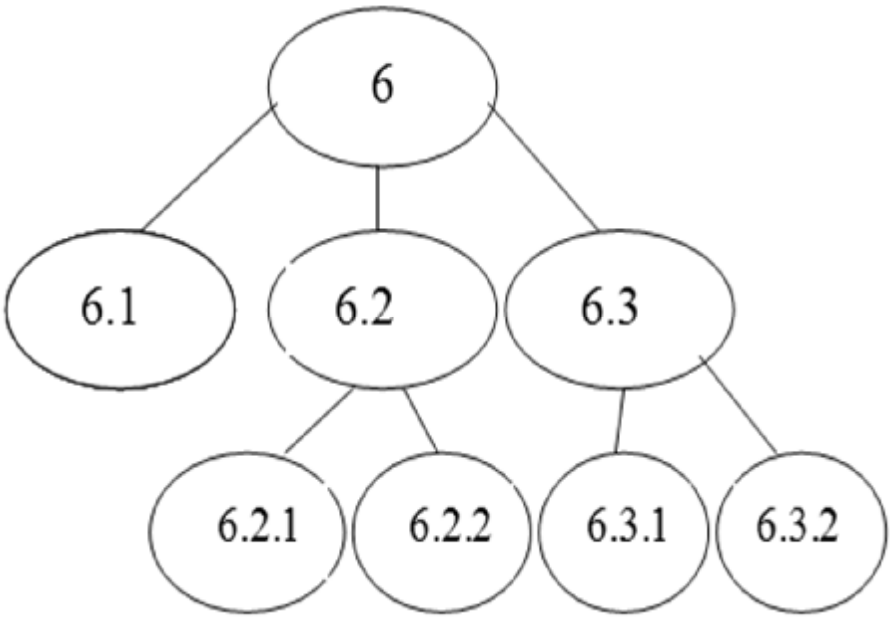
1. Định nghĩa và khái niệm

1.1. Định nghĩa cây (tree)

- Cây là một tập hợp hữu hạn các nút, trong đó có một nút đặc biệt gọi là gốc (root). Giữa các nút có một quan hệ phân cấp gọi là quan hệ cha con.
- Một cây không có nút nào gọi là cây rỗng (null tree).
- Các ví dụ về cây

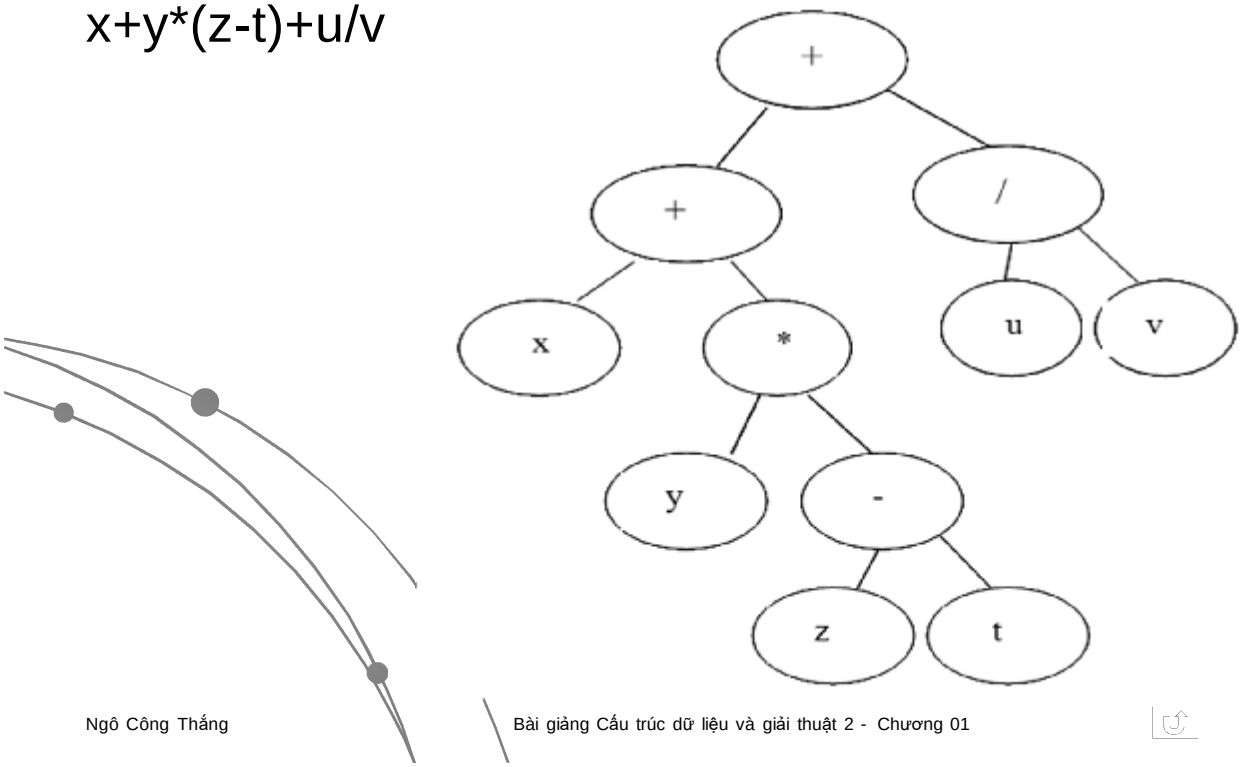
Ví dụ 1: Mục lục của một chương được biểu diễn dạng cây

Chương 6
- 6.1
- 6.2
  - 6.2.1
  - 6.2.2
- 6.3
  - 6.3.1
  - 6.3.2



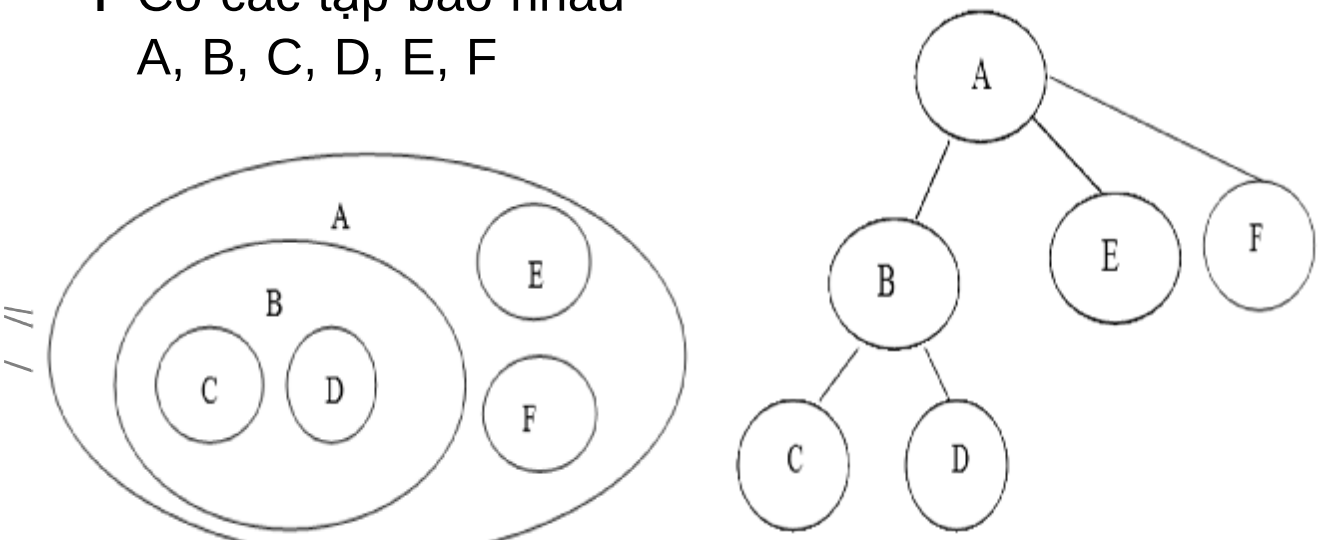
Ví dụ 2: Biểu thức số học được biểu diễn dạng cây

x+y*(z-t)+u/v



Ví dụ 3: Các tập bao nhau được biểu diễn dạng cây

- Có các tập bao nhau A, B, C, D, E, F



1.2. Các khái niệm

- Gốc (Root): Gốc là nút đặc biệt không có nút cha.

  Ví dụ 3: A là gốc. A là cha của B, E, F.

  B, E, F là con của A.

  B, E, F cũng là gốc của các cây con của A

- Cấp (Degree): Số con của một nút gọi là cấp của nút đó.

  Ví dụ 3: A có cấp là 3. E, F có cấp là 0.

  B có cấp là 2.

1.2. Các khái niệm (*tiếp*)

- Lá (Leaf): Nút có cấp bằng không gọi là lá hay nút tận cùng.

  Ví dụ 3: C,D,E,F là lá.

- Nút nhánh (Branch Node): Nút không là lá được gọi là nút nhánh hay nút trong.

  Ví dụ 3: B là nút nhánh.

- Mức (Level): Gốc cây có mức là 1. Nếu nút cha có mức là i thì nút con có mức là i+1.

  Ví dụ 3: A có mức là 1. B, E, F có mức là 2.

  C, D có mức là 3.

1.2. Các khái niệm (*tiếp*)

- Chiều cao của cây (Height) hay chiều sâu của cây (Depth): Là số mức lớn nhất của nút có trên cây.

  Ví dụ 1: Cây có chiều cao là 3

  Ví dụ 2: Cây có chiều cao là 5

  Ví dụ 3: Cây có chiều cao là 3

- Đường đi (Path): Nếu n_1, n_2, ..., n_k là các dãy nút mà n_i là cha của n_{i+1} ($1 \le i < k$) thì dãy đó gọi là đường đi từ n_1 đến n_k. Độ dài của đường đi bằng số nút trừ đi 1. .

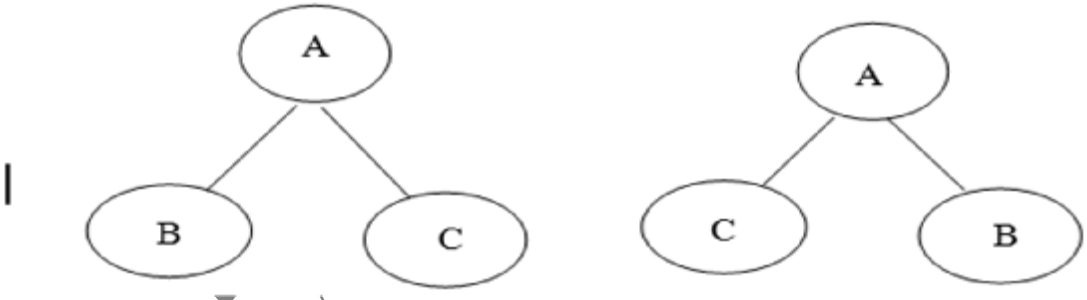
  Ví dụ 3: Đường đi từ A đến C cố độ dài là 3-1=2.

  Đường đi từ A đến E cố độ dài là 2-1=1.

1.2. Các khái niệm (*tiếp*)

- Nếu thứ tự các cây con của một nút được coi trọng thì cây đang xét là cây có thứ tự, ngược lại là cây không có thứ tự.
- Thường thì thứ tự các cây con của một nút được đặt từ trái sang phải.

Hai cây con sau đây là 2 cây con có thứ tự khác nhau.



1.2. Các khái niệm (*tiếp*)

- Đối với cây, ngoài quan hệ cha con, người ta còn mở rộng phỏng theo quan hệ trong gia tộc.
- Rừng (Forest): Nếu có một tập hữu hạn các cây phân biệt thì ta gọi tập đó là rừng.
- Nếu bỏ nút gốc của một cây thì ta sẽ có một rừng.

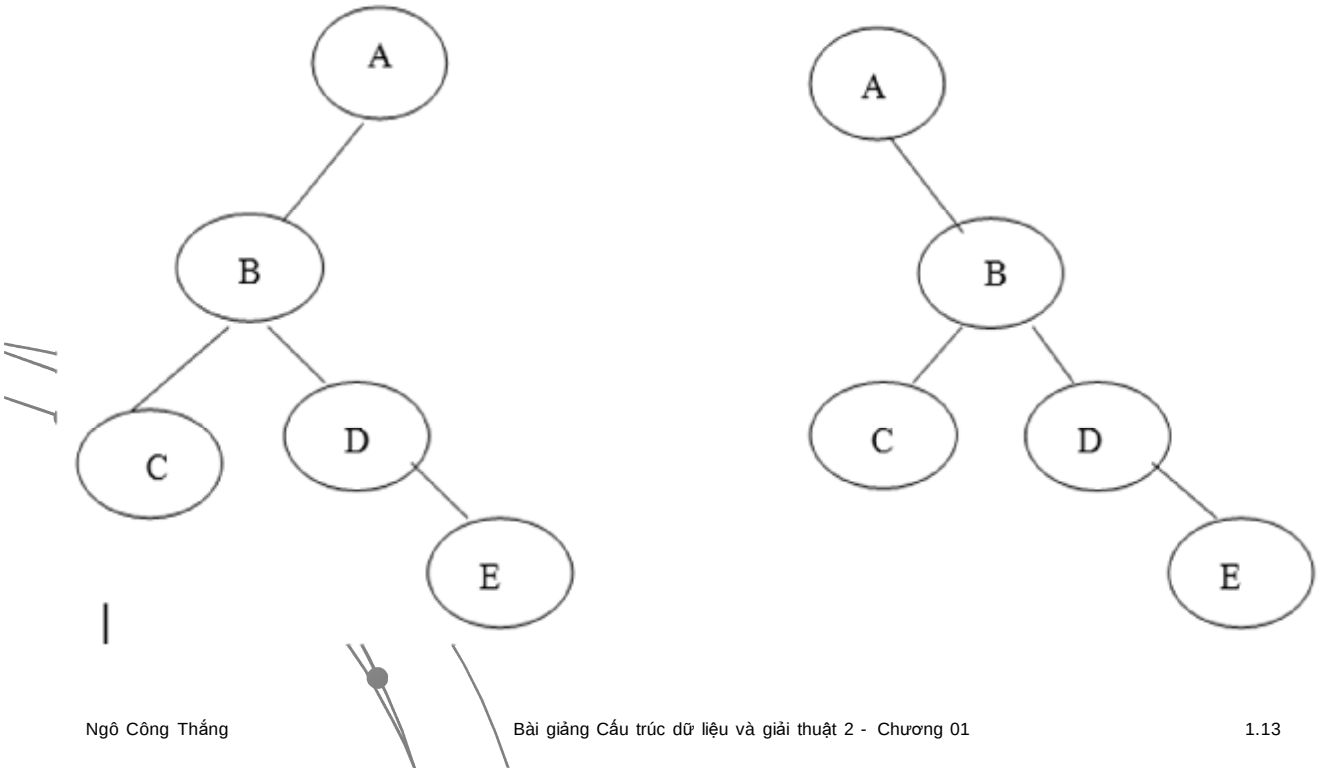
2. Cây nhị phân

2.1. Định nghĩa và tính chất

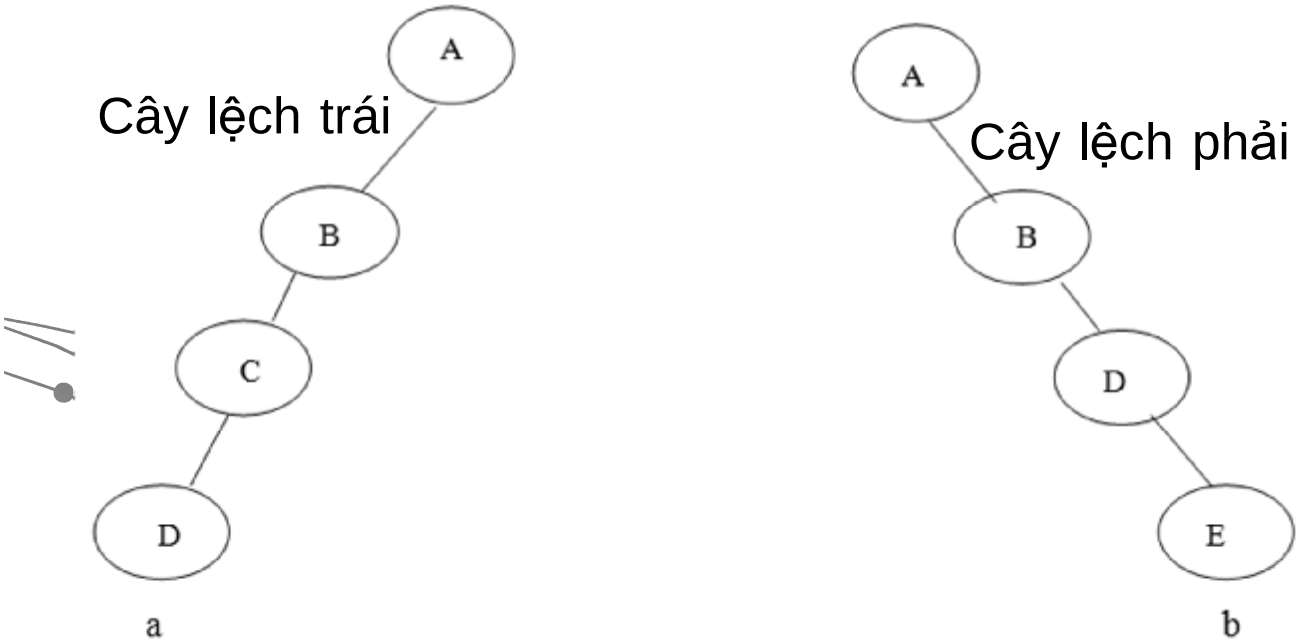
2.1.1. Định nghĩa cây nhị phân

- Cây nhị phân là dạng đặc biệt của cấu trúc cây, mọi nút trên cây chỉ có tối đa là 2 con.
- Đối với cây con của một nút người ta phân biệt cây con trái và cây con phải. Như vậy cây nhị phân là cây có thứ tự.

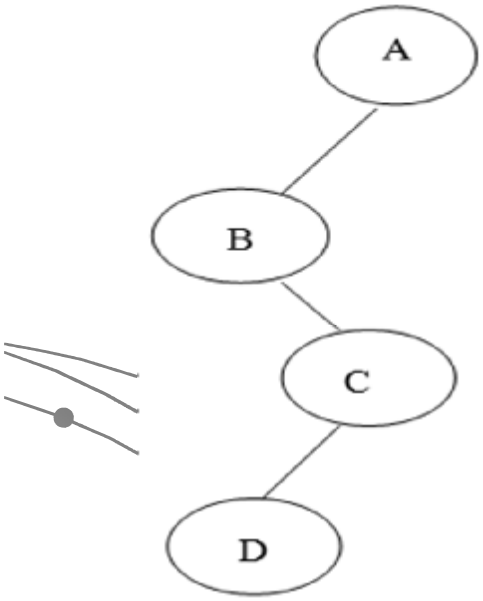
Ví dụ 1: Hai cây sau đây là khác nhau



Ví dụ 2: Cây nhị phân suy biến có dạng một danh sách tuyến tính

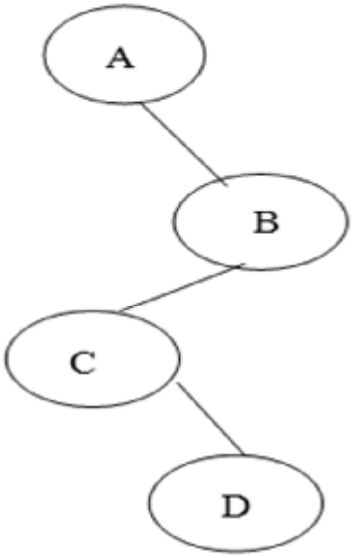


Ví dụ 2: Cây nhị phân suy biến có dạng một danh sách tuyến tính (*tiếp*)



c

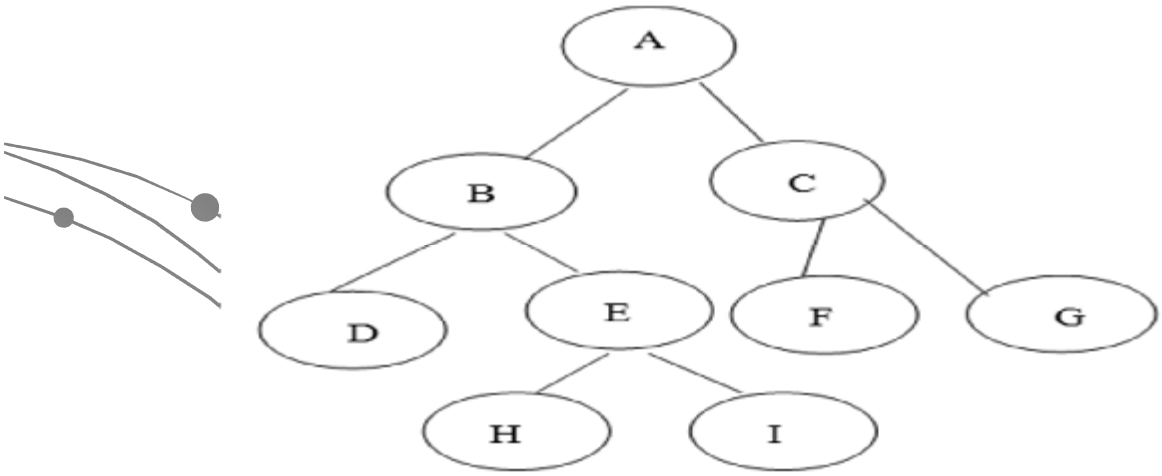
Cây zíc zắc



d

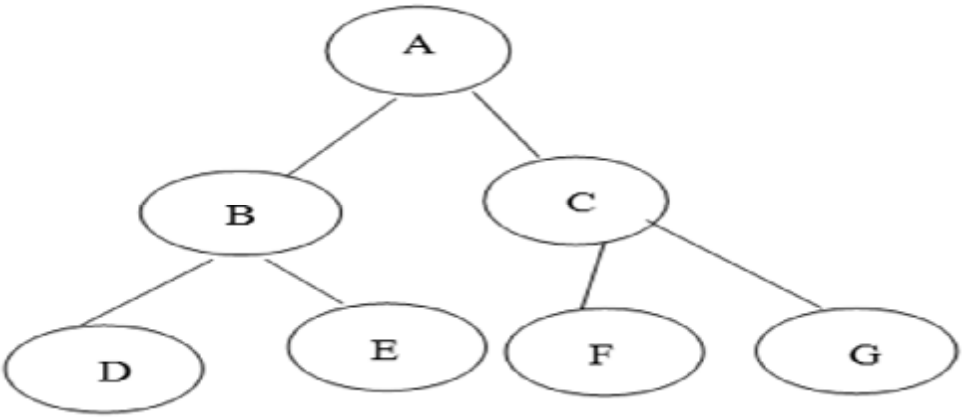
2.1.1. Định nghĩa cây nhị phân (*tiếp*)

- Cây nhị phân hoàn chỉnh: Là cây nhị phân mà các nút nhánh ở các mức đều có hai nút con.



2.1.1. Định nghĩa cây nhị phân (*tiếp*)

- Cây nhị phân đầy đủ: Là cây nhị phân mà các nút ở mọi mức của nút nhánh đều có hai con. Cây nhị phân đầy đủ là trường hợp đặc biệt của cây nhị phân hoàn chỉnh.



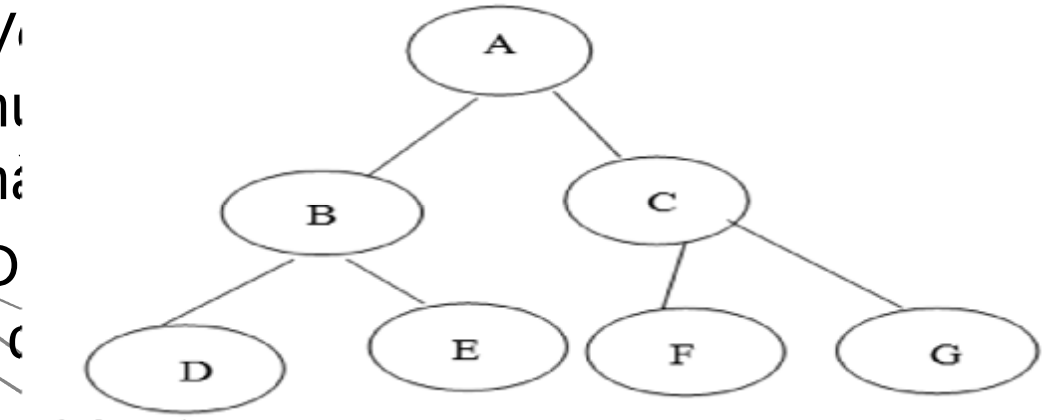
2.1.2. Tính chất

- Số lượng tối đa các nút ở mức i trên 1 cây nhị phân là $2^{(i-1)}$ ($i \geq 1$).
- Số lượng tối đa các nút trên 1 cây nhị phân có chiều cao h là $2^h - 1$.

2.2. Lưu trữ cây nhị phân

2.2.1. Lưu trữ kế tiếp

- V
  nu
  na
- D
  i(



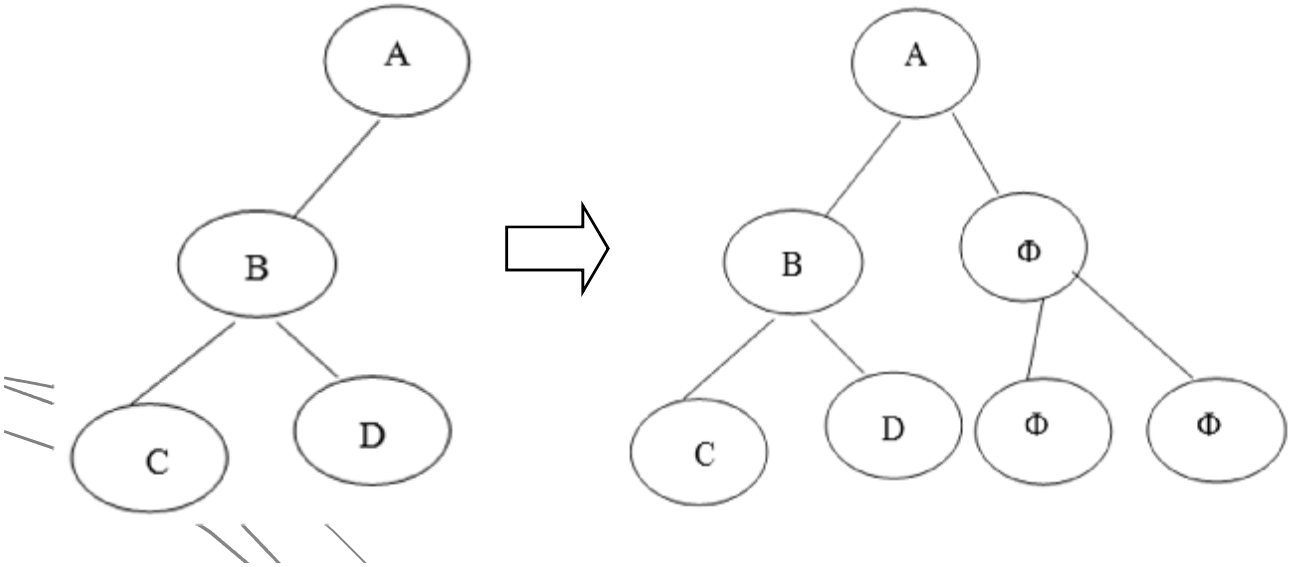
thứ
u:

V(1)	V(2)	V(3)	V(4)	V(5)	V(6)	V(7)
A	B	C	D	E	F	G

2.2.1. Lưu trữ kế tiếp (*tiếp*)

- Với cách lưu trữ bằng mảng, khi biết địa chỉ của nút cha sẽ tính được địa chỉ của nút con và ngược lại. Nếu nút cha là i thì con trái là 2i và con phải là 2i+1. Nếu nút con là i thì nút cha là [i/2].
- Nếu cây không đầy đủ ta phải thêm các nút trống vào để đươc cây nhị phân đầy đủ, sau đó lưu trữ cây đầy đủ đã tạo ra.

Ví dụ



V(1)	V(2)	V(3)	V(4)	V(5)	V(6)	V(7)
A	B	Φ	C	D	Φ	Φ

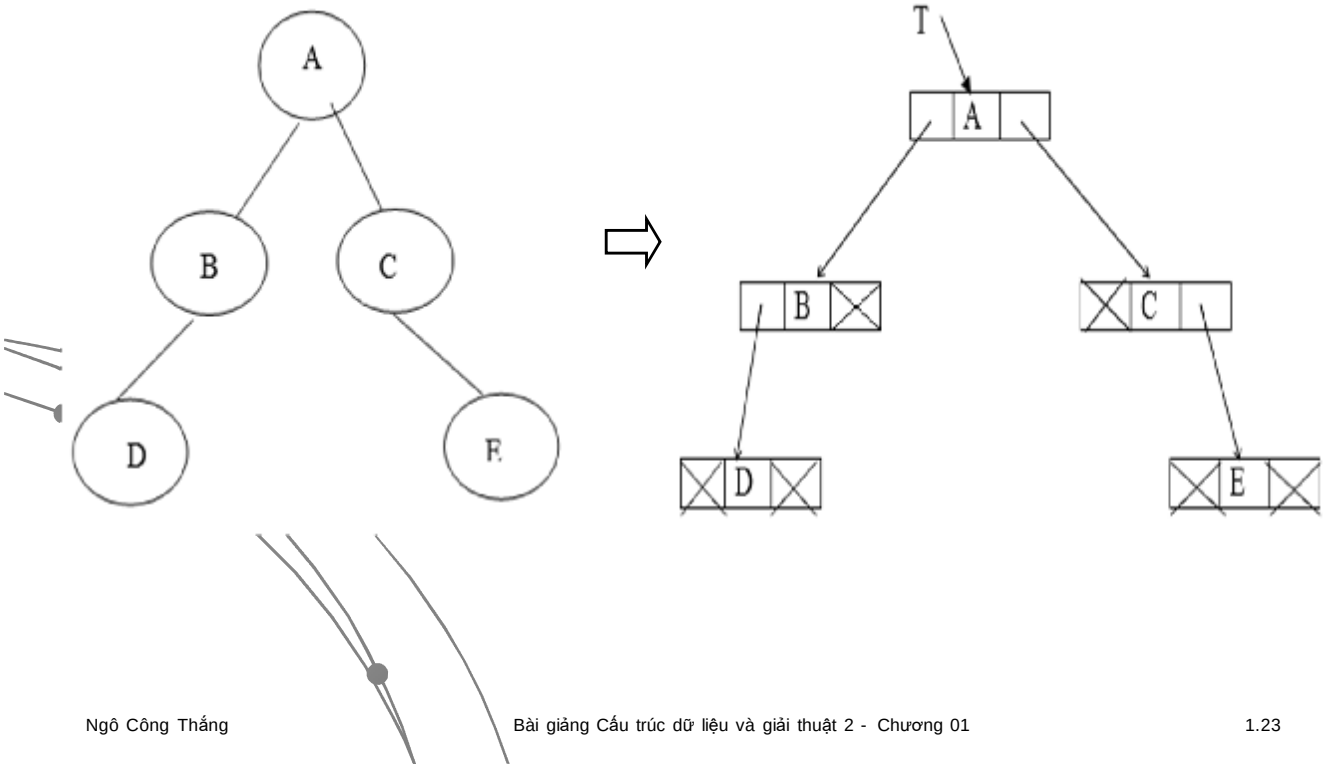
2.2.2. Lưu trữ phần tán

- Trong cách lưu trữ này, mỗi nút ứng với một phần tử nhớ có quy cách dưới đây.
- Để truy nhập vào các nút trong cây nhị phân cần có một con trỏ T trỏ vào nút gốc của cây đó.
- Người ta quy ước: Nếu cây rỗng thì T = Φ

LPTR	INFOR	RPTR

LPTR : Con trỏ trỏ tới cây con trái của nút đó
RPTR : Con trỏ trỏ tới cây con phải của nút đó
INFOR : Trường thông tin.

Ví dụ



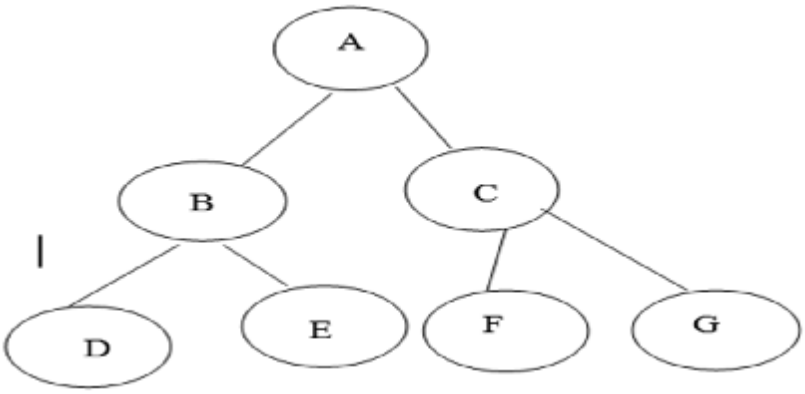
2.3. Các phép toán duyệt cây nhị phân

- Phép xử lý các nút trên cây (gọi chung là phép thăm - visit) là cách thăm tất cả các nút của cây một cách hệ thống, sao cho mỗi nút chỉ được thăm một lần.
- Một nút có 2 con, ta có 3 cách duyệt, các cách duyệt được định nghĩa đệ quy như sau:
  - Cách 1: Duyệt theo thứ tự trước (preorder traversal)
    - Thăm gốc
    - Duyệt cây con trái theo thứ tự trước
    - Duyệt cây con phải theo thứ tự trước

2.3. Duyệt cây nhị phân (*tiếp*)

- Cách 2: Duyệt theo thứ tự giữa (inorder traversal)
  - Duyệt cây con trái theo thứ tự giữa
  - Thăm gốc
  - Duyệt cây con phải theo thứ tự giữa
- Cách 3: Duyệt theo thứ tự sau (postorder traversal)
  - Duyệt cây con trái theo thứ tự sau
  - Duyệt cây con phải theo thứ tự sau
  - Thăm gốc

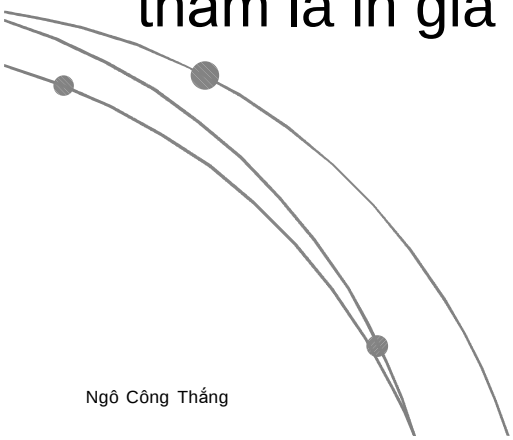
Ví dụ với cây nhị phân sau:



- Duyệt theo thứ tự trước: A B D E C F G
- Duyệt theo thứ tự giữa: D B E A F C G
- Duyệt theo thứ tự sau: D E B F G C A

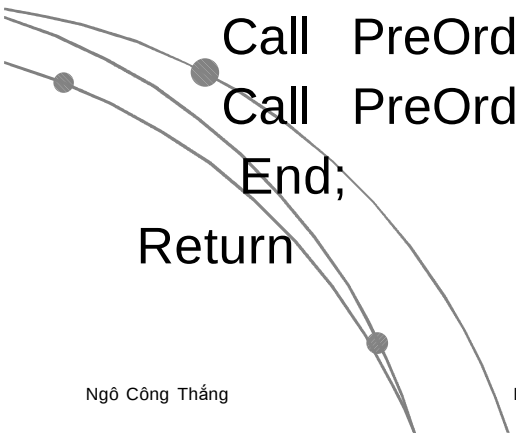
2.3. Duyệt cây nhị phân (*tiếp*)

- Các thủ tục duyệt cây nhị phân đều được viết ở dạng đệ qui.
- Giả sử cây nhị phân lưu trữ bằng danh sách liên kết, T là con trỏ trỏ tới gốc, phép thăm là in giá trị trường Infor của nút đó.



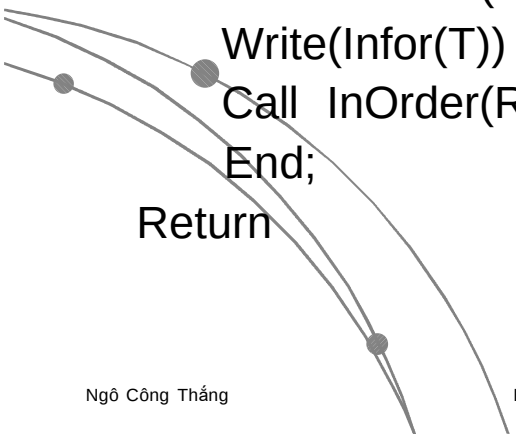
Duyệt cây theo thứ tự trước:

```
Procedure PreOrder(T)
 If T = ϕ then
    Return
 Else Begin
    Write(Infor(T))
    Call PreOrder(Lptr(T))
    Call PreOrder(Rptr(T))
     End;
Return
```



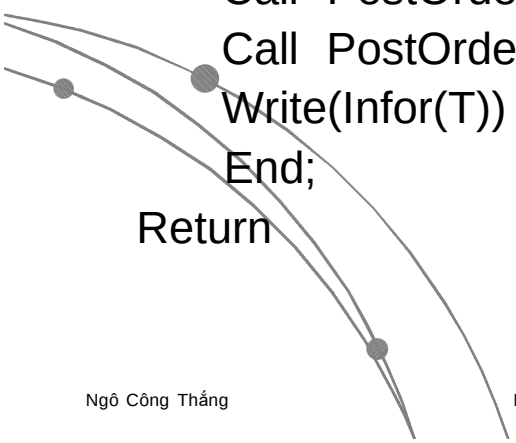
Duyệt cây theo thứ tự giữa:

```
Procedure InOrder(T)
  If  T = ϕ  then Begin
       Return
      End
  Else Begin
       Call  InOrder(Lptr(T))
       Write(Infor(T))
       Call  InOrder(Rptr(T))
       End;
Return
```



Duyệt cây theo thứ tự sau:

```
Procedure PostOrder(T)
  If  T = ϕ  then Begin
       Return
      End
  Else Begin
       Call  PostOrder(Lptr(T))
       Call  PostOrder(Rptr(T))
       Write(Infor(T))
       End;
Return
```



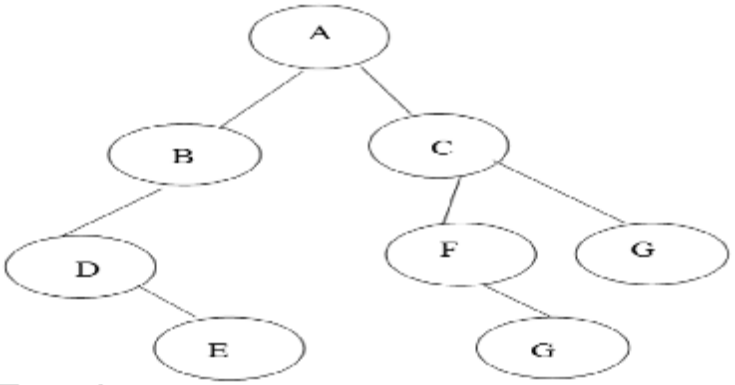
Bài tập

- Bài 1:
  - Xây dựng cây nhị phân biểu diễn biểu thức: (a+b/c)*(d-e*f)
  - Vẽ sơ đồ lưu trữ cây nhị phân biểu diễn biểu thức ở dạng lưu trữ kế tiếp, lưu trữ liên kết.
  - Cho biết thứ tự các nút khi duyệt cây nhị phân đó theo 3 cách.

Bài tập (tiếp)

Bài 2. Cho cây nhị phân dưới đây. Hãy

- Vẽ sơ đồ lưu trữ cây nhị phân ở dạng lưu trữ kế tiếp và lưu trữ liên kết
- Cho biết thứ tự các nút khi duyệt cây nhị phân đó theo 3 cách.



3. Cây tổng quát

- Cây tổng quát là cây có cấp m nào đó.
- Nếu biểu diễn cây tổng quát bằng danh sách liên kết thì một nút có bao nhiêu nhánh sẽ có bấy nhiêu trường liên kết, cách biểu diễn này phức tạp. Nếu biểu diễn cây bằng mảng thì quá trình xử lý cũng rất phức tạp.
- Để đơn giản ta biểu diễn cây tổng quát bằng cây nhị phân. Ta nhận thấy với bất kỳ nút nào trên cây tổng quát nếu có thì chỉ có:
  - Một nút con cực trái (con cả)
  - Một nút em kề cận phải

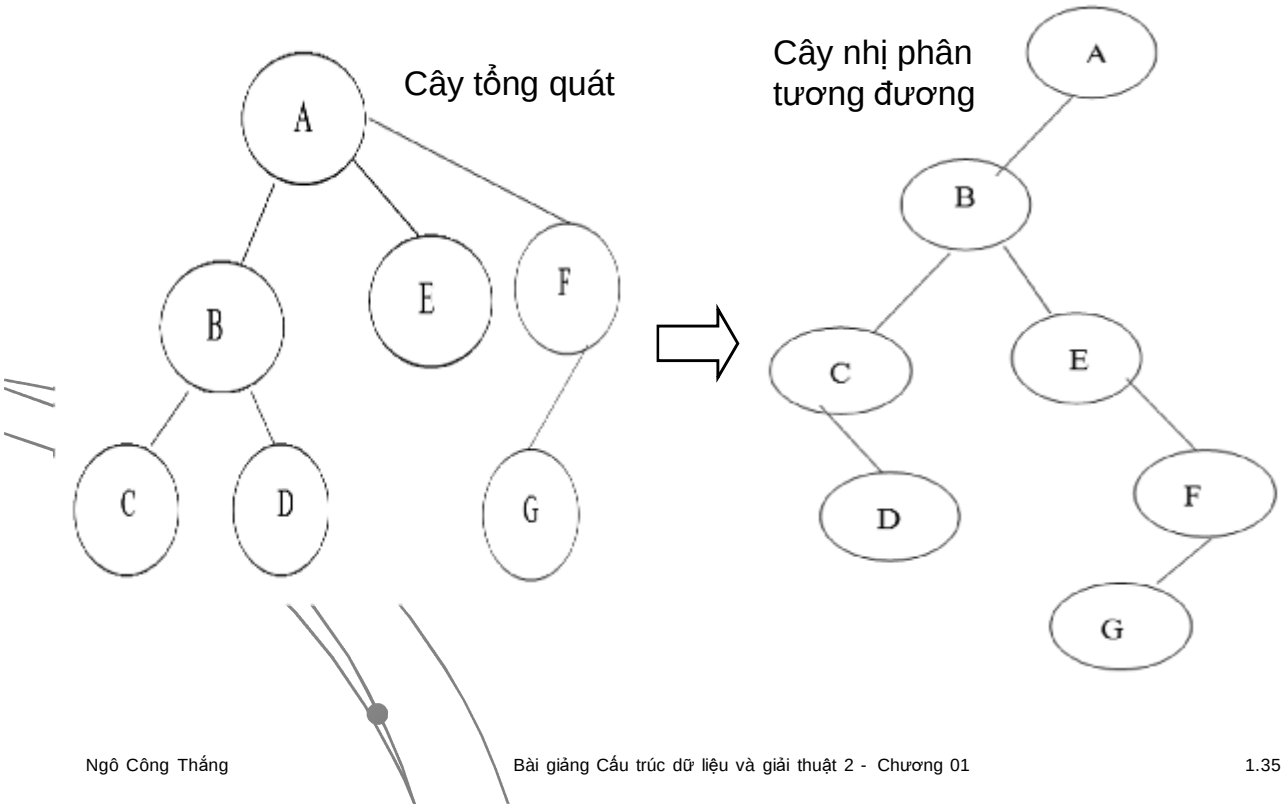
3. Cây tổng quát

- Khi chuyển sang cây nhị phân tương đương, mỗi nút có con trái là con cực trái, con phải là em kề cận phải.
- Mỗi nút của cây tổng quát có có qui cách như sau:

CHILD	INFOR	SIBLING

- CHILD : con trỏ trỏ tới nút con cực trái.
- SIBLING : con trỏ trỏ tới nút em kề cận.
- INFOR : trường thông tin.

Ví dụ:



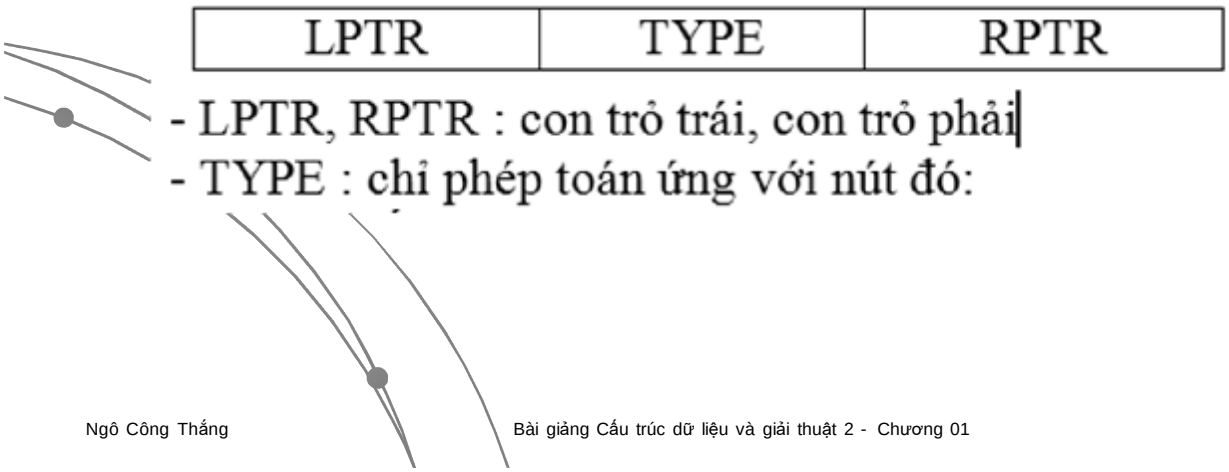
3. Cây tổng quát

- Sau khi chuyển thành cây nhị phân tương đương ta có thể lưu trữ cây tổng quát bằng danh sách liên kết.
- Duyệt cây tổng quát sử dụng các phép duyệt cây nhị phân tương đương.

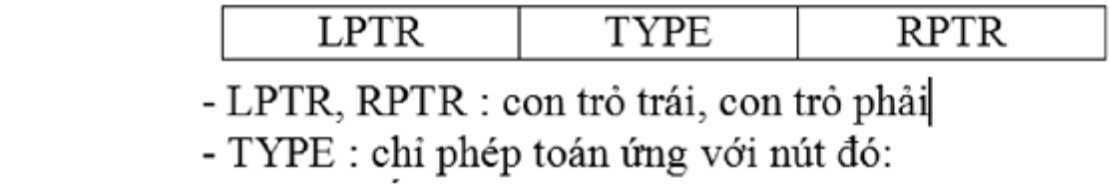
4. Ứng dụng

4.1. Cây biểu diễn biểu thức

- Biểu thức số học với các phép toán 2 ngôi như + - * / có thể biểu diễn bởi cây nhị phân có các nút với quy cách như sau:

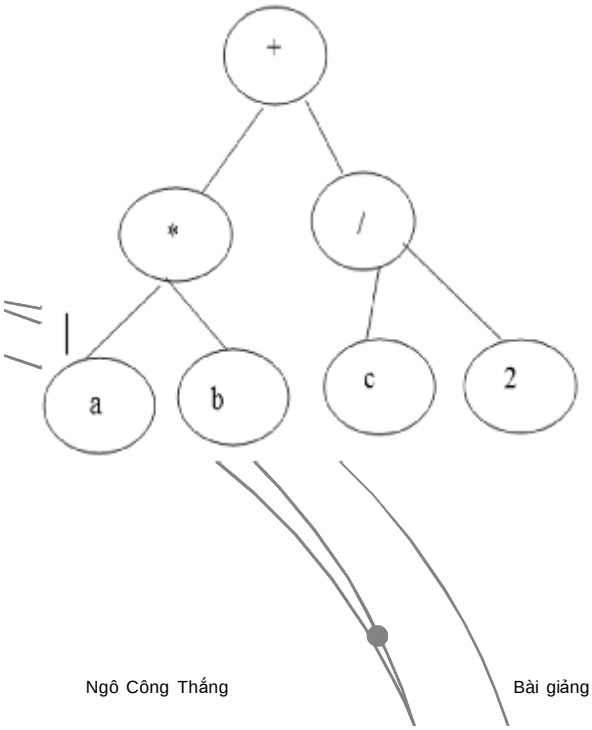


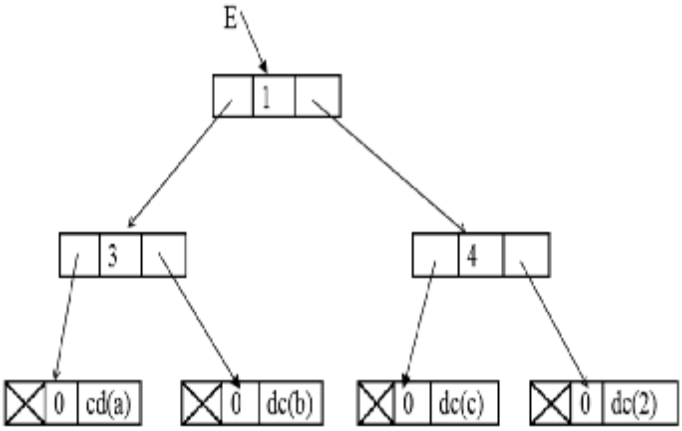
4.1. Cây biểu diễn biểu thức (*tiếp*)



- Nếu không phải nút lá thì giá trị của TYPE sẽ là 1, 2, 3, 4, 5 ứng với các phép +, - , *, /, θ (đổi dấu).
- Nếu là nút lá thì TYPE có giá trị là 0 để chỉ biến hoặc hằng tương ứng với nút đó, còn RPTR trỏ tới địa chỉ trong bảng ký hiệu của biến hoặc hằng và LPTL = Null.
- Ta kí hiệu Value(F) là giá trị ô F
- E là con trỏ trỏ tới gốc cây.
- F là con trỏ phụ.

Ví dụ: Biểu diễn biểu thức a*b+c/2 bằng cây nhị phân sau:





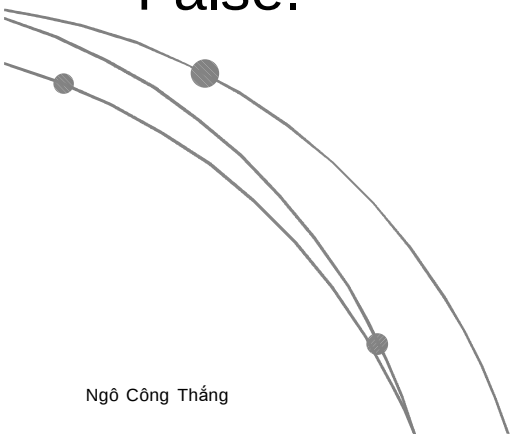
4.2. Định giá trị biểu thức

- Thuật giải định giá trị biểu thức biểu diễn bởi cây nhị phân có gốc E. Thuật giải này được viết dưới dạng đệ quy:

```
Function  EVAL(E)
    Case
        TYPE(E)=0:  Begin F:=RPTR(E)
                          Return(Value(F))
                    End
        TYPE(E)=1:  Return ( EVAL(LPTR(E))+EVAL(RPTR(E)))
        TYPE(E)=2:  Return ( EVAL(LPTR(E))-EVAL(RPTR(E)))
        TYPE(E)=3:  Return ( EVAL(LPTR(E))*EVAL(RPTR(E)))
        TYPE(E)=4:  Return ( EVAL(LPTR(E))/EVAL(RPTR(E)))
        TYPE(E)=5:  Return ( - EVAL(RPTR(E)))
        Else  Return(00)
    End case
Return
```

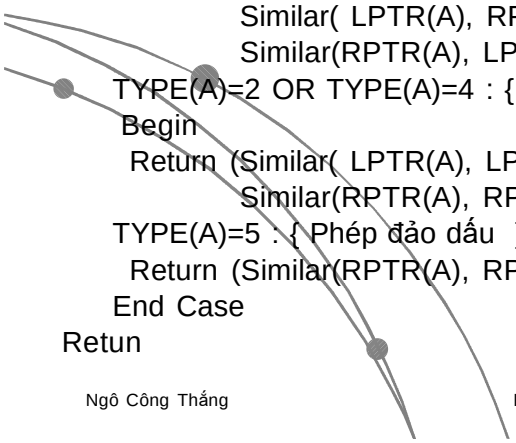
4.3. Xác định 2 biểu thức tương đương

- Cho 2 cây nhị phân biểu diễn biểu thức trỏ bởi A, B. Hàm xác định 2 biểu thức tương đương Similar cho giá trị True nếu 2 biểu thức tương đương, ngược lại cho giá trị False.



Hàm Similar

```
Function   Similar(A,B)
    Bước 1 { Kiểm tra loại gốc cây}
           If  TYPE(A)# TYPE(B)  then  Return(False)
    Bước 2 { Kiểm tra tính tương đương }
    Case
    TYPE(A)=0 : If  Value(RPTR(A)) # Value(RPTR(B)) then  Return(False)
           Else Return(True)
    TYPE(A)=1 OR TYPE(A)=3 : { Phép + hoặc * }
     Begin
      Return (Similar( LPTR(A), LPTR(B)) AND
                Similar(RPTR(A), RPTR(B))  OR
                Similar( LPTR(A), RPTR(B)) AND
                Similar(RPTR(A), LPTR(B)))
    TYPE(A)=2 OR TYPE(A)=4 : { Phép - hoặc / }
     Begin
      Return (Similar( LPTR(A), LPTR(B)) AND
                Similar(RPTR(A), RPTR(B)))
    TYPE(A)=5 : { Phép đảo dấu  }
      Return (Similar(RPTR(A), RPTR(B)))
    End Case
Retun
```



BTVN

- Xây dựng cây nhị phân biểu diễn biểu thức sau: a/b - c*d
- Viết giả mã tính giá trị của biểu thức trên.