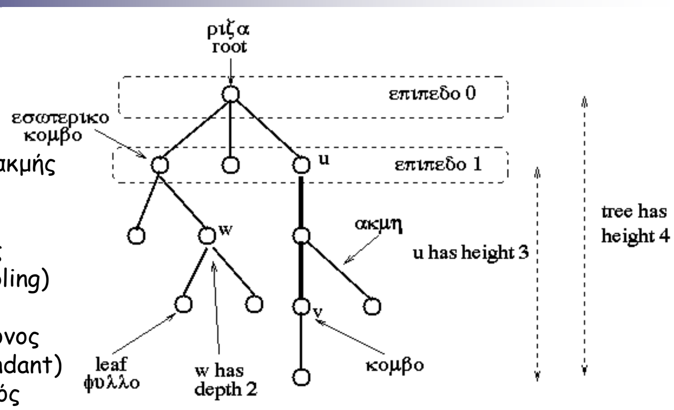
ΕΝΟΤΗΤΑ 3
ΔΕΝΔΡΑ

Δένδρα

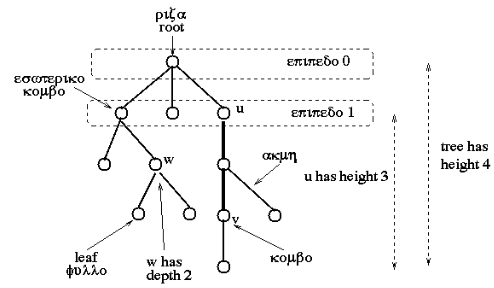
- Κόμβοι (nodes)
- Ακμές (edges)
- Ουρά και κεφαλή ακμής (tail, head)
- Γονέας – Παιδί – Αδελφικός κόμβος (parent, child, sibling)
- Μονοπάτι (path)
- Πρόγονος – απόγονος (ancestor, descendant)
- Φύλλο – Εσωτερικός κόμβος (leaf, non-leaf)



Ένα δένδρο T αποτελείται από ένα σύνολο από κόμβους μεταξύ των οποίων ορίζεται μια σχέση γονέα-παιδιού με τις εξής ιδιότητες:

- Αν το T δεν είναι το κενό δένδρο, περιέχει έναν ειδικό κόμβο που ονομάζεται ρίζα και δεν έχει γονέα.
- Για οποιοδήποτε άλλο κόμβο v του T υπάρχει ένας μοναδικός κόμβος w στο T που αποτελεί το γονέα του v.

Δένδρα



Βαθμός Κόμβου (node degree)
Ο αριθμός των παιδιών του κόμβου.

Βαθμός Δένδρου (tree degree)
Μέγιστος βαθμός μεταξύ των βαθμών των κόμβων του δένδρου.

Επίπεδο (level)
Η ρίζα βρίσκεται στο επίπεδο 0. Ένας κόμβος βρίσκεται στο επίπεδο k αν η απόσταση του από τη ρίζα είναι k. Το επίπεδο είναι επομένως ένα σύνολο από κόμβους.

Ύψος Κόμβου (node height)
Μήκος μακρύτερου μονοπατιού από τον κόμβο σε οποιοδήποτε φύλλο.

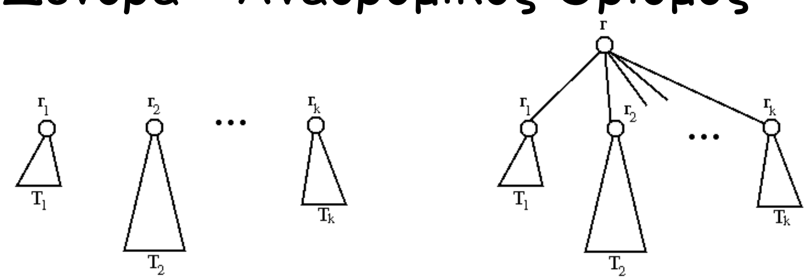
Ύψος Δένδρου (tree height)
Μέγιστο ύψος μεταξύ των υψών των κόμβων του δένδρου.

Βάθος Κόμβου (node depth)
Μήκος μονοπατιού από τη ρίζα στον κόμβο.

Βάθος Δένδρου (tree depth)
Μέγιστο βάθος μεταξύ των βαθών των κόμβων του δένδρου.

Ύψος Δένδρου = Βάθος Δένδρου

Δένδρα – Αναδρομικός Ορισμός



Ένα κενό δένδρο T δεν περιέχει κόμβους και ακμές.

Ένα (μη-κενό) δένδρο T είναι ένα πεπερασμένο σύνολο από έναν ή περισσότερους κόμβους τ.ω.:

- Ένας μόνο κόμβος (χωρίς καμία ακμή) αποτελεί ένα δένδρο. Ο κόμβος αυτός είναι και ρίζα του δένδρου.
- Έστω ότι $T_1, \ldots, T_\kappa$ ($\kappa > 0$) είναι δένδρα που δεν μοιράζονται κόμβους και έστω $r_1, \ldots, r_k$ οι ρίζες τους. Έστω r ένας νέος κόμβος. Αν το T αποτελείται από τους κόμβους και τις ακμές των $T_1, \ldots, T_\kappa$, το νέο κόμβο r και τις νέες ακμές $<r,r_1>$, $<r,r_2>, \ldots, <r,r_k>$, τότε το T είναι δένδρο. Η ρίζα του T είναι το r. Τα $T_1, \ldots, T_\kappa$ είναι υποδένδρα του T.

Είδη Δένδρων

Διατεταγμένο Δένδρο
Δένδρο στο οποίο έχει οριστεί μια διάταξη στα παιδιά κάθε κόμβου.

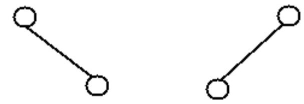
Δυαδικό δένδρο
Διατεταγμένο δένδρο του οποίου κάθε κόμβος έχει το πολύ δύο παιδιά (ένα αριστερό και ένα δεξί).
- λ (nill ή NULL): κενό δυαδικό δένδρο (που δεν περιέχει κανένα κόμβο και καμία ακμή)

Γεμάτο Δυαδικό Δένδρο (full binary tree)
Δεν υπάρχει κόμβος με μόνο ένα παιδί στο δένδρο.

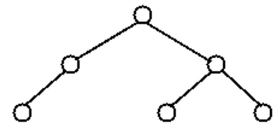
Τέλειο Δυαδικό Δένδρο (perfect binary tree)
Γεμάτο δυαδικό δένδρο στο οποίο όλα τα φύλλα έχουν το ίδιο βάθος.

Δάσος
Πεπερασμένο σύνολο από δένδρα.

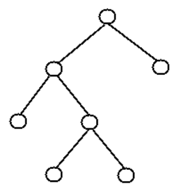
Διατεταγμένα Δένδρα



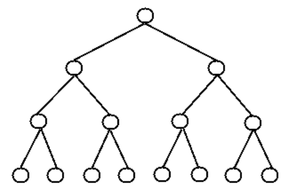
Δυαδικό Δένδρο



Γεμάτο Δυαδικό Δένδρο

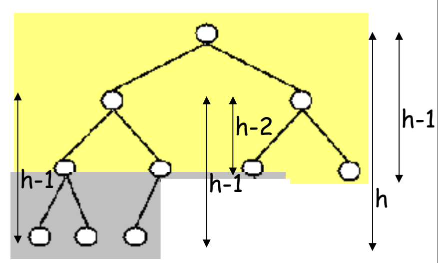


Τέλειο Δυαδικό Δένδρο

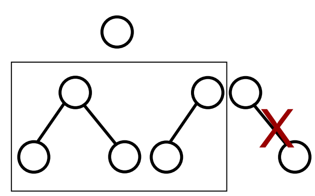


Είδη Δένδρων

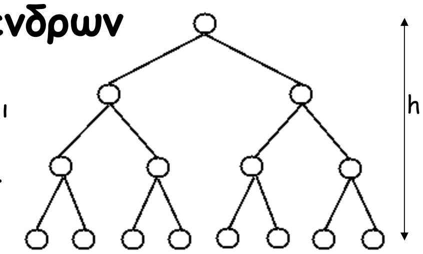
Πλήρες Δυαδικό Δένδρο Ύψους h (complete binary tree of height h)
Αποτελείται από ένα τέλειο δυαδικό δένδρο ύψους h-1 στο οποίο έχουν προστεθεί ένα ή περισσότερα φύλλα με ύψος h. Τα φύλλα αυτά έχουν τοποθετηθεί στις αριστερότερες θέσεις του δένδρου.



Αναδρομικός Ορισμός
- Ένα πλήρες δυαδικό δένδρο ύψους 0 αποτελείται από ένα μόνο κόμβο.
- Ένα πλήρες δυαδικό δένδρο ύψους 1 είναι ένα δένδρο ύψους 1 στο οποίο η ρίζα έχει είτε δύο παιδιά ή ένα μόνο **αριστερό** παιδί.
- Ένα πλήρες δυαδικό δένδρο ύψους h>1, αποτελείται από μια ρίζα και 2 υποδένδρα τ.ω:
  - είτε το αριστερό υποδένδρο είναι τέλειο ύψους h-1 και το δεξιό είναι πλήρες ύψους h-1, ή
  - το αριστερό υποδένδρο είναι πλήρες ύψους h-1 και το δεξιό είναι τέλειο ύψους h-2.



Ιδιότητες Δυαδικών Δένδρων



Πρόταση

Ένα τέλειο δυαδικό δένδρο ύψους h έχει $2^{h+1}-1$ κόμβους, εκ των οποίων 2^h είναι φύλλα και 2^h-1 είναι εσωτερικοί κόμβοι.

Απόδειξη

Με επαγωγή στο h.

<u>Βάση επαγωγής, h = 0</u>

Το τέλειο δυαδικό δένδρο ύψους 0 αποτελείται από έναν μόνο κόμβο-ρίζα και άρα έχει 1 κόμβο που είναι φύλλο και 0 εσωτερικούς κόμβους.

Πράγματι:

$2^{h+1}-1 = 2^{0+1}-1 = 2-1 = 1$ κόμβος

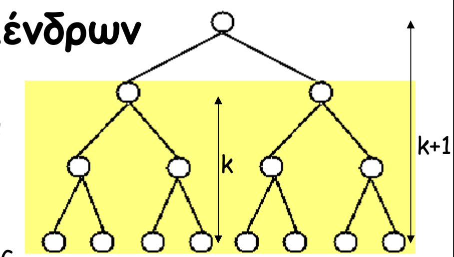
$2^h = 2^0 = 1$ φύλλο

$2^h-1 = 0$ εσωτερικοί κόμβοι

<u>Επαγωγική Υπόθεση</u>

Θεωρούμε έναν οποιονδήποτε ακέραιο k >=0. Έστω ότι οποιοδήποτε τέλειο δυαδικό δένδρο ύψους k έχει $2^{k+1}-1$ κόμβους, εκ των οποίων 2^k είναι φύλλα και 2^k-1 είναι εσωτερικοί κόμβοι.

Ιδιότητες Δυαδικών Δένδρων



<u>Επαγωγικό βήμα</u>

Θα αποδείξουμε ότι ο ισχυρισμός είναι σωστός για δένδρα ύψους k+1.

Ένα τέλειο δένδρο T ύψους k+1 αποτελείται από 2 τέλεια δένδρα ύψους k (έστω T_1, T_2) και τη ρίζα του. Από επαγωγική υπόθεση καθένα από τα T_1, T_2, έχει $2^{\kappa+1}-1$ κόμβους, εκ των οποίων 2^κ είναι φύλλα και $2^\kappa-1$ είναι εσωτερικοί κόμβοι.

Άρα το T έχει:

$2*(2^{\kappa+1}-1)+1$ κόμβους $= 2^{\kappa+2}-1$ κόμβους (όπως απαιτείται),

εκ των οποίων:

$2^\kappa + 2^\kappa = 2^{\kappa+1}$ είναι φύλλα (όπως απαιτείται), και

$2*(2^\kappa-1)+1 = 2^{\kappa+1}-1$ είναι εσωτερικοί κόμβοι (όπως απαιτείται).

Λειτουργίες σε Δένδρα

- ❑ Parent(v): επιστρέφει τον κόμβο γονέα του v ή nill αν ο v είναι η ρίζα
- ❑ Children(v): επιστρέφει το σύνολο των παιδιών του v ή το άδειο σύνολο αν ο v είναι φύλλο
- ❑ FirstChild(v): επιστρέφει το πρώτο παιδί του v ή nill αν ο v είναι φύλλο (σε διατεταγμένα δένδρα)
- ❑ RightSibling(v): επιστρέφει το δεξιό αδελφικό κόμβο του v ή nill αν ο v είναι η ρίζα ή το δεξιότερο παιδί του γονικού του κόμβου
- ❑ LeftSibling(v): επιστρέφει τον αριστερό αδελφικό κόμβο του v ή nill αν ο v είναι η ρίζα ή το αριστερότερο παιδί του γονικού του κόμβου
- ❑ IsLeaf(v): επιστρέφει true αν ο v είναι φύλλο, false διαφορετικά
- ❑ Depth(v): επιστρέφει το βάθος του v στο δένδρο
- ❑ Height(v): επιστρέφει το ύψος του v στο δένδρο

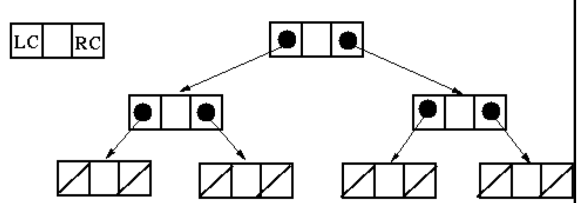
Σε δυαδικά δένδρα

- ❑ LeftChild(v) (RightChild(v)): επιστρέφει το αριστερό (δεξιό, αντίστοιχα) παιδί του v (ή nill)

Υλοποίηση Δένδρων

Υλοποίηση Δυαδικών Δένδρων

Κάθε κόμβος έχει ένα πεδίο data και δύο δείκτες LC (Left Child) και RC (Right Child) που δείχνουν στο αριστερό και στο δεξιό παιδί του κόμβου αντίστοιχα.

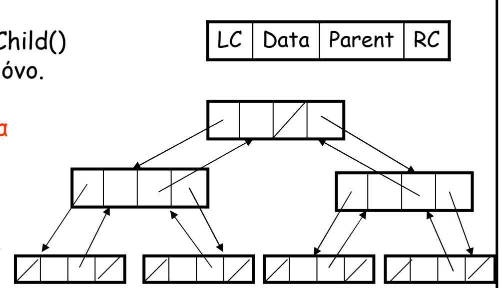


Οι λειτουργίες LeftChild() και RightChild() υλοποιούνται πολύ εύκολα σε Θ(1) χρόνο.

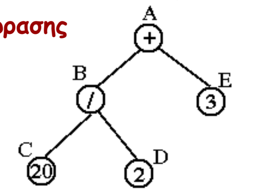
Είναι το ίδιο αλήθεια για τη λειτουργία Parent()?

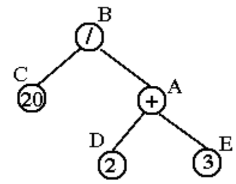
Αποδοτική Υλοποίηση της Parent()

Κρατάμε και ένα τρίτο δείκτη σε κάθε κόμβο που δείχνει στον κόμβο γονέα («Διπλά Συνδεδεμένο» δένδρο).



Δένδρα Αριθμητικών Εκφράσεων





Υπολογισμός Αριθμητικής Έκφρασης

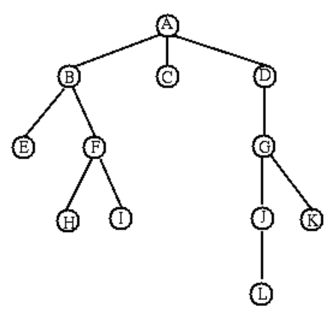
- *Label(v):* ο αριθμός ή η πράξη που αποτελεί τα data του v
- *ApplyOp(op: operation, x,y: numbers):* υπολογίζει την έκφραση x <op> y, ανάλογα με το ποια πράξη είναι το op.

function *Evaluate(pointer P):* **integer**
/* Return value of the expression represented by the tree with root P */

```
integer x_l, x_r, res;
if IsLeaf(P) then return Label(P)
else {
    x_l = Evaluate(LeftChild(P))
    x_r = Evaluate(RightChild(P))
    op = Label(P)
    res =  ApplyOp(op, x_l, x_r);
    return res;
}
```

```
if IsLeaf(P->A)              -> FALSE
x_l = Evaluate(P->B);        (<- 10)
   if IsLeaf(P->B)           -> FALSE
   x_l = Evaluate(P->C);     (<- 20)
       if IsLeaf(P->C) return 20;
   x_r = Evaluate(P->D);     (<- 2)
       if IsLeaf(P->D) return 2;
   res = ApplyOp('/', 20, 2) = 20 / 2 = 10;
   return 10;
x_r = Evaluate(P->E);        (<- 3)
   if IsLeaf(P->E) return 3;
res = ApplyOp('+', 10, 3) = 10 + 3 = 13;
return 13;
```


Διάσχιση Δένδρων



Διάσχιση ή διέλευση δένδρου (tree traversal) χαρακτηρίζεται κάθε διαδικασία επισκέψεως-επεξεργασίας όλων των κόμβων του δένδρου με συστηματικό τρόπο.

Ισοδύναμα, θα μπορούσε να ορισθεί ως επιβολή ολικής διατάξεως επί των κόμβων του δένδρου μέσω συστηματικής ανακτήσεως των δεικτών και επεξεργασία των δεδομένων των κόμβων βάσει αυτής της διατάξεως.

Η ύπαρξη πολλαπλών δυνατοτήτων για τη διάταξη ενός κόμβου ως προς τους απογόνους του, δημιουργεί διάφορα είδη διασχίσεων (προδιατεταγμένη διάσχιση, μεταδιατεταγμένη διάσχιση, ενδοδιατεγμένη διάσχιση).

Visit(pointer p): αυθαίρετη λειτουργία που εφαρμόζεται στον κόμβο στον οποίο δείχνει ο δείκτης P

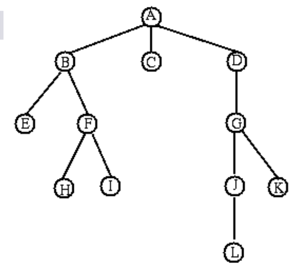
Παράδειγμα – Τύπωση των δεδομένων του κόμβου

```
Visit(pointer p) {
   print(p->data);
}
```

Προδιατεταγμένη Διάσχιση

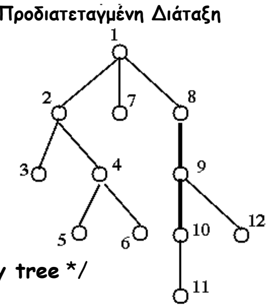
Για κάθε κόμβο v, η προδιατεταγμένη διάσχιση κάνει τα εξής με τη σειρά που αναφέρονται:

- Επίσκεψη του v
- Επίσκεψη των υποδένδρων του v ξεκινώντας από το αριστερότερο προς το δεξιότερο υποδένδρο του.



Η διαδικασία διάσχισης ξεκινά από τη ρίζα.
Κάθε κόμβος προηγείται των παιδιών του στην διάταξη που προκύπτει.

Προδιατεταγμένη Διάταξη



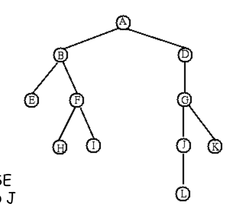

```
Procedure PreOrder(pointer p) {
/* P is a pointer to the root of a general tree */
   if (p == NULL) return;
   Visit(p);
   foreach child q of p, in order (from left to right)
        PreOrder(q);
}
```

```
Procedure PreOrder(pointer p) {
/* p is a pointer to the root of a binary tree */
    if (p == NULL) return;
    Visit(p);
    PreOrder(p->LC);
    PreOrder(p->RC);
}
```

Παράδειγμα
Εκτύπωση Κόμβων

A,B,E,F,H,I,C,D,G,J,L,K

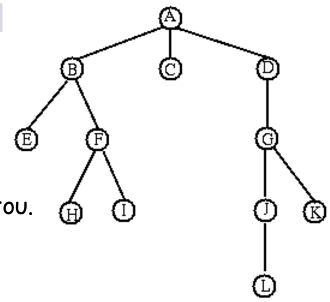
Ιχνηλατώντας την PreOrder




```
PreOrder(p->A)
  if (p == NULL) // αποτιμάται σε FALSE
  print(p->A);     // τυπώνει το A
  PreOrder(p->B);
     if (p == NULL) // -> FALSE
     print(p->B); // τυπώνει το B
     PreOrder(p->E);
         if (p == NULL) // -> FALSE
         print(p->E); // τυπώνει το E
         PreOrder(NULL); // LC του E
             if (p == NULL) return;
         PreOrder(NULL) // RC του E
             if (p == NULL) return;
     PreOrder(p->F);
         if (p == NULL) // -> FALSE
         print(p->F); // τυπώνει το F
         PreOrder(p->H);
              if (p == NULL) // -> FALSE
              print(p->H); // τυπώνει το H
              PreOrder(NULL); // LC του H
                 if (p == NULL) return;
               PreORder(NULL) // RC του H
                 if (p == NULL) return;
         PreOrder(p->I);
             if (p == NULL) // -> FALSE
             print(p->I); // τυπώνει το I
             PreOrder(NULL); // LC του I
                if (p == NULL) return;
             PreOrder(NULL) // RC του I
                if (p == NULL) return;
   // τέλος PreORder(p->I), PreOrder(p->F) & PreOrder(p->B)
```

```
PreOrder(p->D);
   if (p == NULL) // -> FALSE
   print(p->D); // τυπώνει το D
   PreOrder(p->G);
       if (p == NULL) // -> FALSE
       print(p->G); // τυπώνει το G
       PreOrder(p->J); // LC του G
          if (p == NULL) // -> FALSE
          print(p->J); // τυπώνει το J
          PreOrder(p->L);
              if (p == NULL) // -> FALSE
              print(p->L); // τυπώνει το L
              PreOrder(NULL); // LC του L
                if (p == NULL) return;
              PreORder(NULL) // RC του L
                if (p == NULL) return;
          PreOrder(NULL);  // RC του J
              if (p == NULL) return;
       PreOrder(P->K);
          if (p == NULL) // -> FALSE
          print(p->K); // τυπώνει το K
          PreOrder(NULL); // LC του K
              if (p == NULL) return;
          PreOrder(NULL) // RC του K
              if (p == NULL) return;
   PreOrder(NULL); // RC του D
      if (p == NULL) return;
// τέλος PreOrder(p->D) & PreORder(p->A)
```

Μεταδιατεταγμένη Διάσχιση



Για κάθε κόμβο v, η μεταδιατεταγμένη διάσχιση κάνει τα εξής με τη σειρά που αναφέρονται:

- Επίσκεψη των υποδένδρων του v ξεκινώντας από το αριστερότερο προς το δεξιότερο υποδένδρο του.
- Επίσκεψη του v

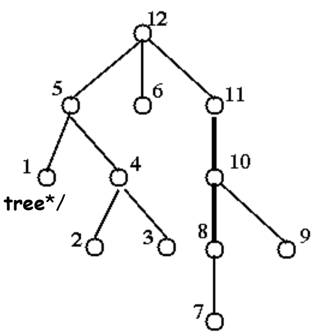
Η διαδικασία διάσχισης ξεκινά από τη ρίζα.
Κάθε κόμβος έπεται των παιδιών του στην διάταξη.

```
Procedure PostOrder(pointer p) {
/* p is a pointer to the root of a general tree */
     if (p == NULL) return;
    foreach child q of p, in order {
         Postorder(q);
    }
    Visit(p);
}
```

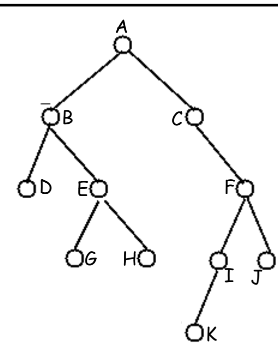
Παράδειγμα Εκτύπωση Κόμβων

E,H,I,F,B,C,L,J,K,G,D,A

```
Procedure PostOrder(pointer p) {
/*p is a pointer to the root of a binary tree*/
    if (p == NULL) return;
    PostOrder(p->LC);
    PostOrder(p->RC);
    Visit(p);
}
```



Ενδοδιατεγμένη Διάσχιση



Για κάθε κόμβο v, η ενδοδιατεταγμένη διάσχιση ενός δυαδικού δένδρου κάνει τα εξής με τη σειρά που αναφέρονται:

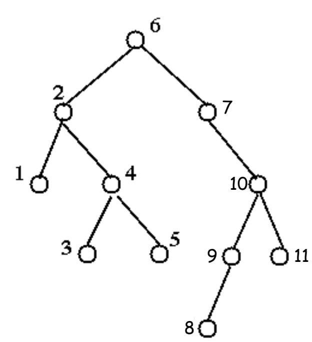
- Επίσκεψη του αριστερού υποδένδρου του v
- Επίσκεψη του v
- Επίσκεψη του δεξιού υποδένδρου του v

Η διαδικασία διάσχισης ξεκινά από τη ρίζα.
Το αριστερό παιδί ενός κόμβου v προηγείται του v, ενώ το δεξιό παιδί του έπεται του v στην διάταξη.

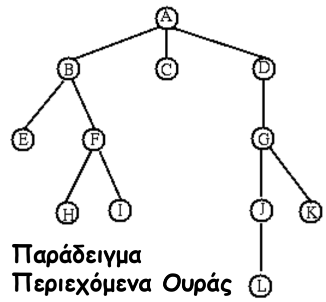
```
Procedure InOrder(pointer p) {
/* p is a pointer to the root of a binary tree */
    if (p == NULL) return;
    InOrder(p->LC);
    Visit(p);
    InOrder(p->RC);
}
```

Παράδειγμα Εκτύπωσης

D,B,G,E,H,A,C,K,I,F,J



Διασχίσεις Δένδρων



Διάσχιση Δένδρου κατά Επίπεδα (κατά πλάτος)
Επισκέπτεται τους κόμβους κατά αύξον βάθος και τους κόμβους του ίδιου επιπέδου από τα αριστερά προς τα δεξιά.

Χρήση Ουράς
❑ Αρχικά η ουρά περιέχει μόνο τη ρίζα.
❑ Στη συνέχεια επαναληπτικά: κάνουμε Deque ένα στοιχείο της ουράς και προσθέτουμε τα παιδιά από αριστερά προς τα δεξιά του στοιχείου αυτού.

```
Procedure LevelOrder(pointer r) {
   Queue Q;  pointer P;
   MakeEmptyQueue(Q); Enqueue(Q,r);
   while (! IsEmptyQueue(Q)) {
        P = Dequeue(Q);
        Visit(P);
        foreach child c of P, in order, do
                Enqueue(c);
   }
}
```

Παράδειγμα
Περιεχόμενα Ουράς
A
B, C, D
C, D, E, F
D, E, F
E, F, G
F, G
G, H, I
H, I, J, K
I, J, K
J, K
K, L
L
<empty>

Διασχίσεις Δένδρων

Με Χρήση Στοίβας
Αναδρομικές λύσεις έχουν ήδη συζητηθεί.
Χρονική Πολυπλοκότητα Διάσχισης;
O(ng(n)), όπου n το πλήθος των κόμβων και g(n) η χρονική πολυπλοκότητα της Visit()

Χωρική Πολυπλοκότητα;
Μέγεθος στοίβας ανάλογο του ύψους του δένδρου.

Διάσχιση κατά Επίπεδα
Χρονική Πολυπλοκότητα Διάσχισης; O(n)
Χωρική Πολυπλοκότητα;
Πόσους κόμβους μπορεί να περιέχει η ουρά στη χειρότερη περίπτωση;

Υλοποίηση Διατεταγμένων Δένδρων

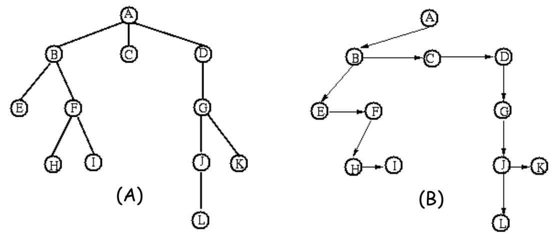
Αν είναι γνωστό το μέγιστο πλήθος παιδιών ενός κόμβου (έστω MC), ο κόμβος περιέχει έναν πίνακα με MC δείκτες έναν για κάθε δυνητικό παιδί του (κάποιοι από τους δείκτες μπορεί να είναι NULL αν τα αντίστοιχα παιδιά δεν υπάρχουν).

Τι γίνεται αν δεν γνωρίζουμε τον αριθμό των παιδιών που μπορεί να έχει κάποιος κόμβος;

Απεικόνιση Διατεταγμένου Δένδρου ως Δυαδικό

Αν κάθε κόμβος διασυνδέεται με το αριστερότερο παιδί του και με τον πρώτο στα δεξιά αδελφικό του κόμβο, τότε αρκούν δύο δείκτες σε κάθε κόμβο.

Το αρχικό δένδρο μετασχηματίζεται σε δυαδικό!



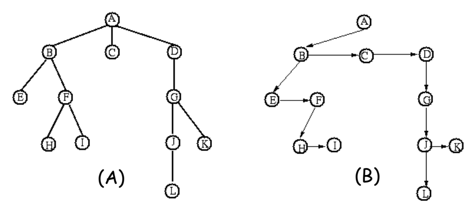
Μορφή Κόμβου

LC	data	RS

- ❑ LC: Left Child
- ❑ RS: Right Sibling

Υλοποίηση Διατεταγμένων Δένδρων

Μπορούμε να τυπώσουμε το διατεταγμένο δένδρο (A) βασιζόμενοι στο δυαδικό δένδρο (B);



- ❑ Επίσκεψη σε κάθε κόμβο v.
- ❑ Η Visit τυπώνει το πεδίο data των κόμβων που αποτελούν παιδιά του κόμβου v στο αρχικό δένδρο.

```
Procedure PrintTree(pointer R) {
    if (R == NULL) return;
    Visit(R);
    PrintTree(R->LC);
    PrintTree(R->RS);
}
```

```
Procedure Visit(pointer R) {
    pointer P;
    print(R->data); print(" Children: ");
    P = R->LC;
    while (P != NULL) {
        print(P->data);
        P = P->RS;
    }
    print("\n");
}
```

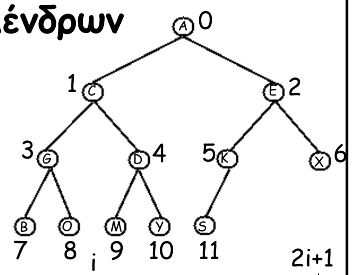
Ύψος δυαδικού ως προς το αρχικό δένδρο;

Πολυπλοκότητες

- ❑ FirstChild(), RightSibling(): Θ(1)
- ❑ kth-child(k, v), εύρεση του k-οστού παιδιού του v: Θ(k).
- ❑ Parent(): δεν υποστηρίζεται αποδοτικά.

Υλοποίηση Πλήρων Δυαδικών Δένδρων

Υπάρχει μόνο ένα πλήρες δυαδικό δένδρο με n κόμβους και το υλοποιούμε με ένα πίνακα N στοιχείων.
Αριθμούμε τους κόμβους με αριθμούς στο διάστημα {0,…,n-1} και αποθηκεύουμε τον κόμβο i στο στοιχείο T[i] του πίνακα.
Θέλουμε να κάνουμε την αρίθμηση έτσι ώστε να πετύχουμε την εκτέλεση χρήσιμων λειτουργιών στο δένδρο σε σταθερό χρόνο.



i					2i+1						
0	1	2	3	4	5	6	7	8	9	10	11
A	C	E	G	D	K	X	B	O	M	Y	s

Αρίθμηση

Η ρίζα είναι ο κόμβος 0.
Το αριστερό παιδί του κόμβου i αριθμείται ως κόμβος 2i+1, ενώ το δεξί παιδί του ως κόμβος 2i+2.

Υλοποίηση Λειτουργιών

- IsLeaf(i): return (2i+1 $\geq$ n);
- LeftChild(i): if (2i+1 < n) return (2i+1) else return nill;
- RightChild(i): if (2i+2 < n) return(2i+2); else return nill;
- LeftSibling(i): if (i != 0 and i not odd) return (i-1);
- RightSibling(i): if (i != n-1 and i not even) return(i+1);
- Parent(i): if (i != 0) return($\lfloor (i-1)/2 \rfloor$);

Χρονική πολυπλοκότητα κάθε λειτουργίας: $\Theta(1)$