ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ

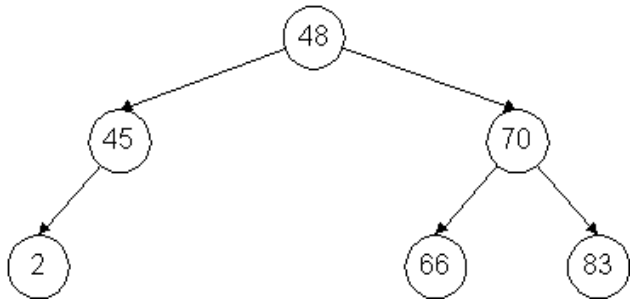
ΔΕΝΤΡΑ
5.4 Τα ΔΔΑ ως Αναδρομικές Δομές Δεδομένων

ΔΔΑ ως αναδρομική δομή δεδομένων

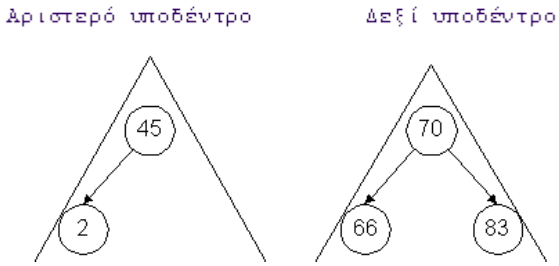
Ένα δυαδικό δέντρο μπορεί πολύ φυσικά να οριστεί ως μια **αναδρομική δομή δεδομένων** (**recursive data structure**).

ΔΔΑ ως αναδρομική δομή δεδομένων: *παράδειγμα*

Έστω ότι έχουμε το διπλανό δυαδικό δέντρο:



Στη ρίζα του δέντρου βρίσκεται ο αριθμός 48 και υπάρχουν δείκτες προς τους κόμβους με αριθμούς 45 και 70, καθένας από τους οποίους είναι ρίζα ενός δυαδικού υποδέντρου, όπως δείχνουν και τα ακόλουθα σχήματα:



Ας πάρουμε τώρα το αριστερό υποδέντρο, που έχει ρίζα τον κόμβο με τον αριθμό 45 και ένα αριστερό παιδί, αλλά δεν έχει δεξί.

Μπορούμε να θεωρήσουμε ότι αυτός ο κόμβος έχει δείκτες προς δυο δυαδικά υποδέντρα, ένα αριστερό υποδέντρο και ένα δεξί υποδέντρο (κενό), με την προϋπόθεση ότι επιτρέπονται τα κενά δέντρα:

Αναδρομικός ορισμός δυαδικού δέντρου

Επειδή το αριστερό υποδέντρο αποτελείται μόνο από έναν κόμβο, τον κόμβο με τον αριθμό 2, θα έχει κενό δεξί υποδέντρο και κενό αριστερό υποδέντρο.

Έτσι, λοιπόν, **ένα δυαδικό δέντρο μπορεί να οριστεί αναδρομικά** ως εξής:

Ένα δυαδικό δέντρο είτε

α. είναι κενό

είτε

β. αποτελείται από έναν κόμβο, που ονομάζεται ρίζα, ο οποίος έχει δείκτες προς δύο δυαδικά υποδέντρα, που ονομάζονται αριστερό υποδέντρο και δεξί υποδέντρο.

Αναδρομικός ορισμός δυαδικού δέντρου

Εξαιτίας της αναδρομικής φύσης των δυαδικών δέντρων, πολλές από τις βασικές λειτουργίες τους μπορούν να εκτελεστούν πολύ απλά χρησιμοποιώντας **αναδρομικούς αλγόριθμους**.

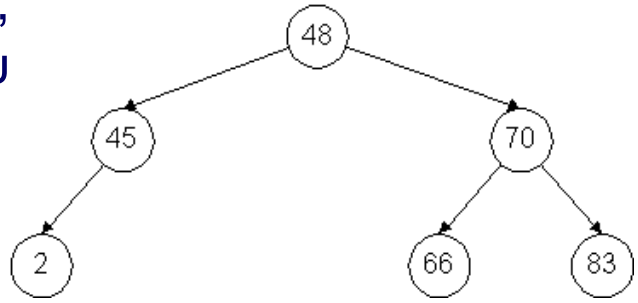
Έστω, για ***παράδειγμα***, η λειτουργία της διάσχισης, κατά την οποία μετακινούμαστε μέσα στο δυαδικό δέντρο και επισκεπτόμαστε κάθε κόμβο ακριβώς μια φορά.
Υποθέτουμε ότι η σειρά με την οποία επισκεπτόμαστε κάθε κόμβο δεν έχει σημασία, αλλά είναι σημαντικό να επισκεφτούμε όλους τους κόμβους και να επεξεργαστούμε τις πληροφορίες που περιέχονται σ' αυτούς ακριβώς μια φορά.

Ένα απλό αναδρομικό σχήμα είναι να διασχίσουμε το δυαδικό δέντρο ως εξής:

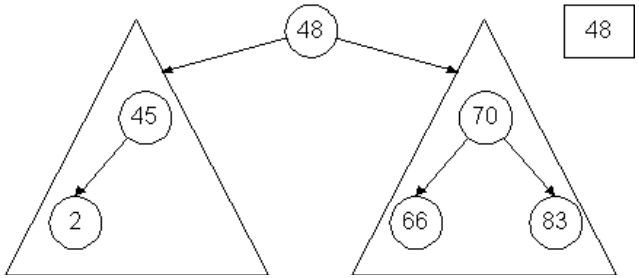
1. Επίσκεψη της ρίζας και επεξεργασία των περιεχομένων της.
2. Διάσχιση του αριστερού υποδέντρου.
3. Διάσχιση του δεξιού υποδέντρου.

Παράδειγμα αναδρομικής διάσχισης

Για να διασχίσουμε το διπλανό δέντρο, εμφανίζοντας τα περιεχόμενα κάθε κόμβου που επισκεπτόμαστε,



- ξεκινάμε από τη ρίζα και εμφανίζουμε τον αριθμό 48, όπως δείχνει και το διπλανό σχήμα:



Σχήμα Α

Εν συνεχεία πρέπει να διασχίσουμε πρώτα το αριστερό υποδέντρο και έπειτα το δεξί.

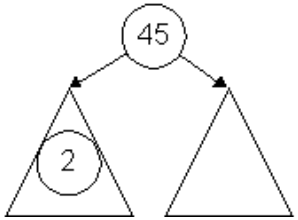
Όταν ολοκληρωθεί η διάσχιση και των δύο υποδέντρων θα έχουμε διασχίσει όλο το δυαδικό δέντρο.

Παράδειγμα αναδρομικής διάσχισης

Τώρα το πρόβλημα έχει περιοριστεί στο να διασχίσουμε δύο μικρότερα δυαδικά δέντρα.

- Παίρνουμε πρώτα το αριστερό υποδέντρο (Σχήμα Β) και επισκεπτόμαστε τη ρίζα του, οπότε εμφανίζεται ο αριθμός 45.

  Στη συνέχεια θα διασχίσουμε πρώτα το αριστερό υποδέντρο του και έπειτα το δεξί.



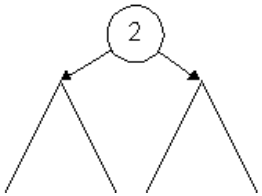
Σχήμα Β

48 | 45

- Για να διασχίσουμε το αριστερό υποδέντρο, επισκεπτόμαστε τη ρίζα του και εμφανίζουμε τον αριθμό 2.

  Τώρα πρέπει να διασχίσουμε πρώτα το αριστερό και στη συνέχεια το δεξί υποδέντρο.

  Όπως φαίνεται και από το σχήμα, και τα δύο υποδέντρα είναι κενά οπότε δεν χρειάζεται να κάνουμε τίποτα, δηλαδή η διάσχιση του δέντρου του Σχήματος Γ έχει ολοκληρωθεί.

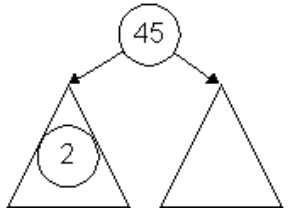


Σχήμα Γ

48 | 45 | 2

Παράδειγμα αναδρομικής διάσχισης

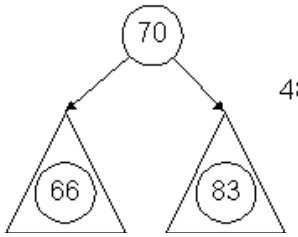
- Τώρα πρέπει να διασχίσουμε το δεξί υποδέντρο του δέντρου του Σχήματος Β. Το υποδέντρο αυτό είναι κενό, οπότε και πάλι δεν κάνουμε τίποτα. Επομένως, η διάσχιση του αριστερού υποδέντρου της ρίζας του αρχικού δέντρου έχει ολοκληρωθεί και



Σχήμα Β

- μένει να διασχίσουμε το δεξί υποδέντρο. Ξεκινάμε πάλι από τη ρίζα αυτού του υποδέντρου και εμφανίζουμε τον αριθμό 70, που είναι αποθηκευμένος εκεί, όπως φαίνεται και στο Σχήμα Δ.

  Στη συνέχεια θα διασχίσουμε το αριστερό και το δεξί υποδέντρο.

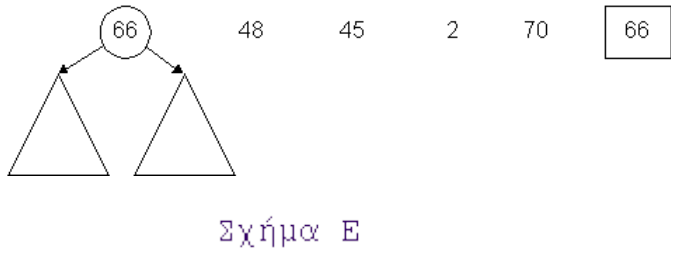


48 45 2 70

Σχήμα Δ

Παράδειγμα αναδρομικής διάσχισης

- Πρώτα θα διασχίσουμε το αριστερό υποδέντρο του παραπάνω σχήματος, οπότε εμφανίζουμε τον αριθμό 66, που βρίσκεται στη ρίζα του.
- Ακολουθεί η διάσχιση του αριστερού υποδέντρου και του δεξιού υποδέντρου. Όπως φαίνεται και από το Σχήμα Ε, τα υποδέντρα αυτά είναι κενά, πράγμα που σημαίνει ότι δεν έχουμε να κάνουμε τίποτα και η διάσχιση του δέντρου του Σχήματος Ε έχει ολοκληρωθεί.
- Τώρα μας μένει να διασχίσουμε το δεξί υποδέντρο του Σχήματος Δ.

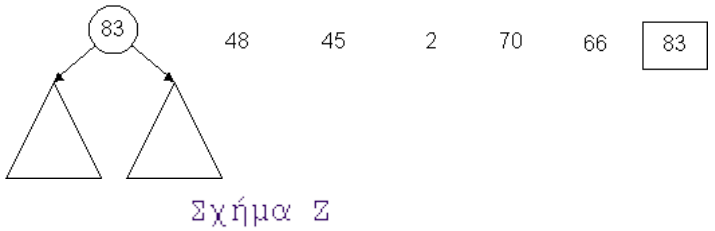


Σχήμα Ε

Εμφανίζουμε τον αριθμό 83 της ρίζας του και στη συνέχεια πρέπει να διασχίσουμε το αριστερό και το δεξί υποδέντρο, όπως φαίνονται και στο Σχήμα Ζ.

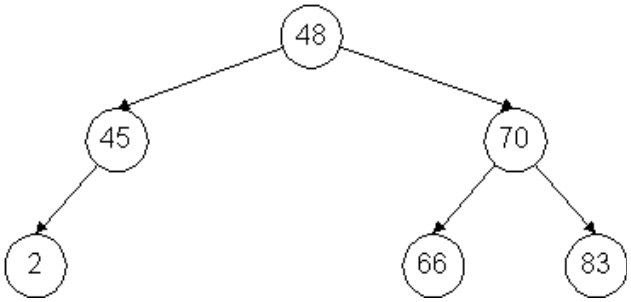
Επειδή και τα δύο αυτά δέντρα είναι κενά, δεν κάνουμε τίποτα.

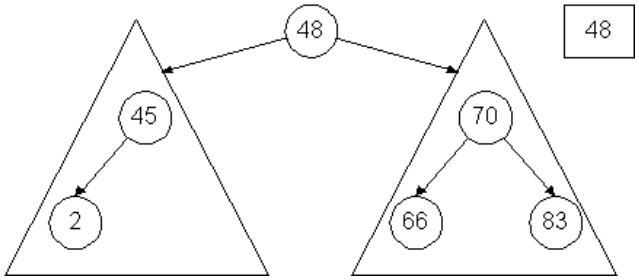
Η διάσχιση του δέντρου του Σχήματος Ζ έχει ολοκληρωθεί.



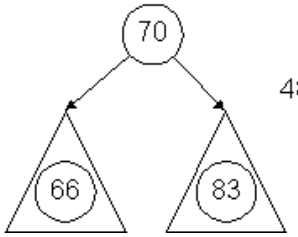
Σχήμα Ζ

Σύνοψη παραδείγματος διάσχισης ΔΔΑ
Ρίζα-Αριστερό υποδένδρο-Δεξί υποδένδρο



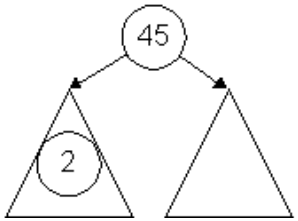


Σχήμα Α



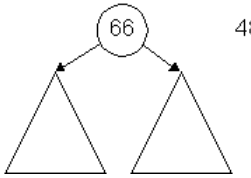
48 45 2 [70]

Σχήμα Δ



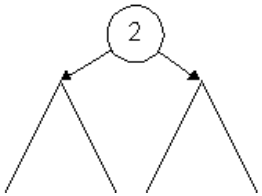
48 [45]

Σχήμα Β



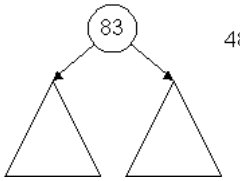
48 45 2 70 [66]

Σχήμα Ε



48 45 [2]

Σχήμα Γ



48 45 2 70 66 [83]

Σχήμα Ζ

Αναδρομική διάσχιση δέντρου

Όπως φαίνεται, από το προηγούμενο παράδειγμα, η διάσχιση ενός δυαδικού δέντρου αναδρομικά απαιτεί **τρία βασικά βήματα**, τα οποία συμβολίζουμε ως N, L και R:

N (**Node**): Επίσκεψη ενός κόμβου.
L (**Left**): Διάσχιση του αριστερού υποδέντρου ενός κόμβου.
R (**Right**): Διάσχιση του δεξιού υποδέντρου ενός κόμβου.

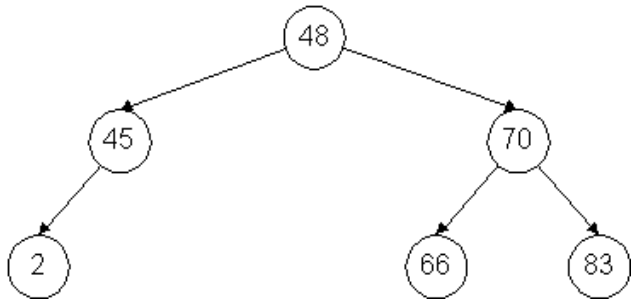
Παρόλο που στο παράδειγμά μας εκτελέσαμε τα βήματα με αυτή τη σειρά, στην πραγματικότητα έχουμε **έξι διατάξεις** όσον αφορά τη σειρά εκτέλεσης των βημάτων:

- LNR
- NLR
- LRN
- NRL
- RNL
- RLN

Αναδρομική διάσχιση δέντρου: *διάταξη LNR*

Αν, για παράδειγμα εφαρμοστεί η **διάταξη LNR**, τότε ο αντίστοιχος αλγόριθμος διάταξης είναι:

Αν το δυαδικό δέντρο είναι κενό **τότε**
  Μην κάνεις τίποτα
Αλλιώς
  L:Διάσχιση του αριστερού υποδέντρου
  N:Επίσκεψη της ρίζας
  R:Διάσχιση του δεξιού υποδέντρου
Τέλος_αν



Αν εφαρμόσουμε τον παραπάνω αλγόριθμο στο δυαδικό δέντρο που χρησιμοποιήσαμε και προηγουμένως, η διάσχισή του εμφανίζει τους κόμβους με τη σειρά 2, 45, 48, 66, 70, 83.

Αναδρομική διάσχιση δέντρου: διατάξεις

Από τις προαναφερθείσες διατάξεις οι πιο σημαντικές είναι οι τρεις πρώτες, όπου πρώτα διασχίζεται το αριστερό υποδέντρο και μετά το δεξί, και έχουν τις εξής συγκεκριμένες ονομασίες:

LNR: **ενδοδιατεταγμένη** (**inorder**)

NLR: **προδιατεταγμένη** (**preorder**)

LRN: **μεταδιατεταγμένη** (**postorder**)

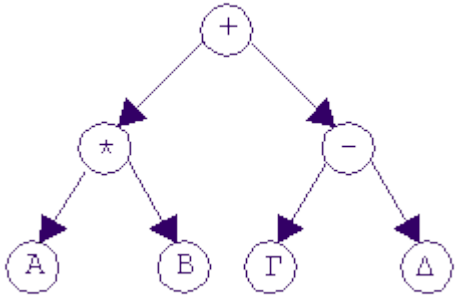
Δέντρα παράστασης

Μια αριθμητική έκφραση μπορεί να παρασταθεί με ένα δυαδικό δέντρο, στο οποίο τα φύλλα περιέχουν τελεστέους και οι εσωτερικοί κόμβοι τελεστές. Ένα τέτοιο δέντρο ονομάζεται **δέντρο παράστασης** (**expression tree**).

Παράδειγμα: η αριθμητική παράσταση

Α * Β + Γ - Δ

μπορεί να παρασταθεί με το διπλανό δυαδικό δέντρο:



όπου κάθε τελεστέος παριστάνεται σαν ένα παιδί ενός κόμβου πατέρα, που παριστάνει τον αντίστοιχο τελεστή.

Με μια ενδοδιατεταγμένη διάσχιση του παραπάνω δέντρου παράστασης προκύπτει η ενδοθεματική έκφραση: **Α * Β + Γ - Δ**

Μια προδιατεταγμένη διάσχιση του ίδιου δέντρου οδηγεί στην προθεματική έκφραση: **+ * Α Β - Γ Δ**

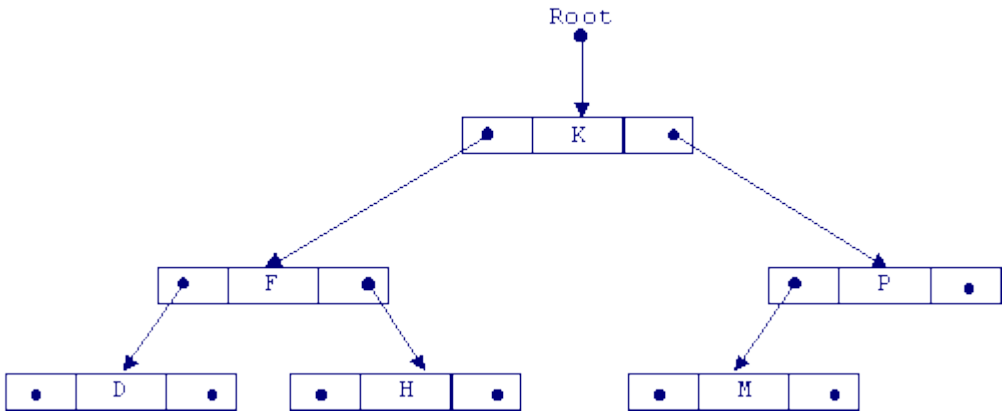
Τέλος, μια μεταδιατεταγμένη διάσχιση του δέντρου μας δίνει την μεταθεματική έκφραση: **Α Β * Γ Δ - +**

Ενδοδιατεταγμένη διάσχιση δέντρου

```
void RecBSTInorder(BinTreePointer Root)
/*Δέχεται:        Ένα δυαδικό δέντρο με το δείκτη Root να δείχνει στην ρίζα του.
  Λειτουργία:     Εκτελεί ενδοδιατεταγμένη διάσχιση του δυαδικού δέντρου και
                  επεξεργάζεται κάθε κόμβο ακριβώς μια φορά.
  Επιστρέφει:     Εξαρτάται από το είδος της επεξεργασίας.*/
{
    if (Root != NULL) {
        RecBSTInorder(Root->LChild);
        printf("%d ",Root->Data);          // επεξεργασία του κόμβου
        RecBSTInorder(Root->RChild);
    }
```

Παράδειγμα ενδοδιατεταγμένης διάσχισης δέντρου

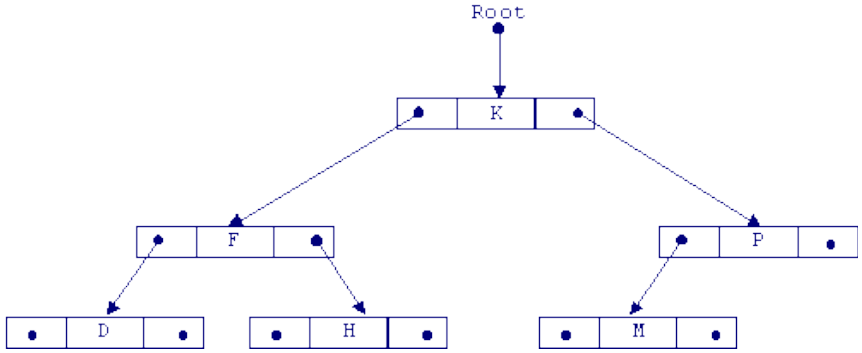
Για να δούμε στην πράξη πώς λειτουργεί η παραπάνω διαδικασία, έστω ότι έχουμε το παρακάτω δυαδικό δέντρο:



Ο πίνακας που ακολουθεί δείχνει αναλυτικά την ενέργεια της διαδικασίας RecBSTInorderγια το παραπάνω δυαδικό δέντρο:

Παράδειγμα ενδοδιατεταγμένης διάσχισης δέντρου

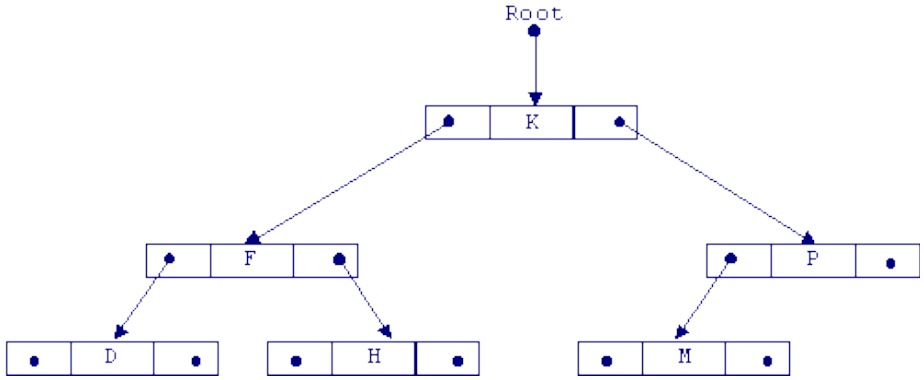
```
void RecBSTInorder(BinTreePointer Root)  {
   if (Root != NULL) {
      RecBSTInorder(Root->LChild);
      printf("%c ",Root->Data);
      RecBSTInorder(Root->RChild);
    }
}
```



K	Κλήση της RecBSTInorder με δείκτη στη ρίζα (F) του αριστερού υποδέντρου.	
F	Κλήση της RecBSTInorder με δείκτη στη ρίζα (D) του αριστερού υποδέντρου.	
D	Κλήση της RecBSTInorder με δείκτη (NULL) στη ρίζα του αριστερού υποδέντρου.	
null	null, επιστροφή στον κόμβο πατέρα.	
D	Εμφάνισε τα περιεχόμενα του κόμβου.	D
D	Κλήση της RecBSTInorder με δείκτη στη ρίζα (NULL) του δεξιού υποδέντρου.	
null	null, επιστροφή στον κόμβο πατέρα.	
D	Επιστροφή στον κόμβο πατέρα (*κλείσιμο αναδρομής με Root το D*).	
F	Εμφάνισε τα περιεχόμενα του κόμβου.	F

Παράδειγμα ενδοδιατεταγμένης διάσχισης δέντρου

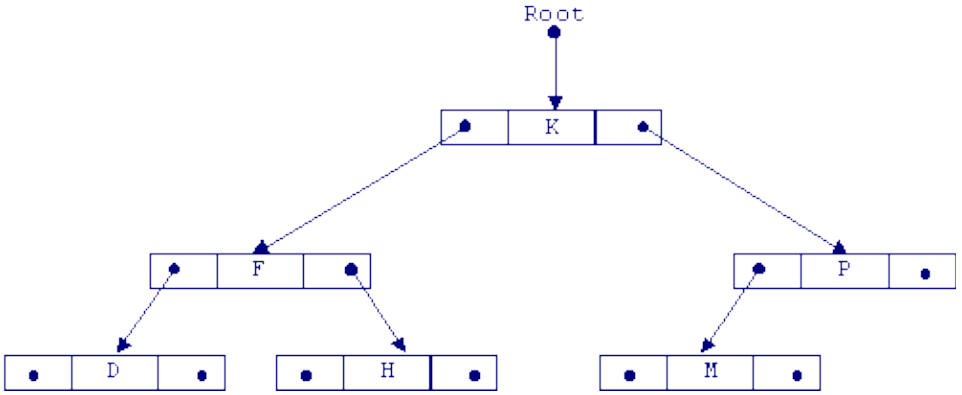
```
void RecBSTInorder(BinTreePointer Root)  {
   if (Root != NULL) {
       RecBSTInorder(Root->LChild);
       printf("%c ",Root->Data);
       RecBSTInorder(Root->RChild);
     }
}
```



F	Κλήση της RecBSTInorder με δείκτη στη ρίζα (H) του δεξιού υποδέντρου.	
H	Κλήση της RecBSTInorder με δείκτη στη ρίζα (NULL) του αριστερού υποδέντρου.	
null	null, επιστροφή στον κόμβο πατέρα.	
H	Εμφάνισε τα περιεχόμενα του κόμβου.	H
H	Κλήση της RecBSTInorder με δείκτη στη ρίζα (NULL) του δεξιού υποδέντρου.	
null	null, επιστροφή στον κόμβο πατέρα.	
H	Επιστροφή στον κόμβο πατέρα (*κλείσιμο αναδρομής με Root το H*).	
F	Επιστροφή στον κόμβο πατέρα (*κλείσιμο αναδρομής με Root το F*).	
K	Εμφάνισε τα περιεχόμενα του κόμβου.	K

Παράδειγμα ενδοδιατεταγμένης διάσχισης δέντρου

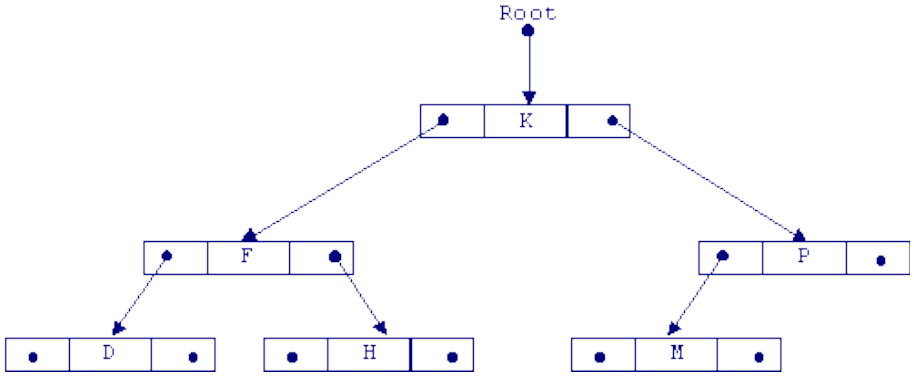
```
void RecBSTInorder(BinTreePointer Root) {
    if (Root != NULL) {
        RecBSTInorder(Root->LChild);
        printf("%c ",Root->Data);
        RecBSTInorder(Root->RChild);
      }
}
```



K	Κλήση της RecBSTInorder με δείκτη στη ρίζα (P) του δεξιού υποδέντρου.	
P	Κλήση της RecBSTInorder με δείκτη στη ρίζα (M) του αριστερού υποδέντρου.	
M	Κλήση της RecBSTInorder με δείκτη στη ρίζα (NULL) του αριστερού υποδέντρου.	
null	null, επιστροφή στον κόμβο πατέρα.	
M	Εμφάνισε τα περιεχόμενα του κόμβου.	M
M	Κλήση της RecBSTInorder με δείκτη στη ρίζα (NULL) του δεξιού υποδέντρου.	
null	null, επιστροφή στον κόμβο πατέρα.	
M	Επιστροφή στον κόμβο πατέρα (*κλείσιμο αναδρομής με Root το M*).	

Παράδειγμα ενδοδιατεταγμένης διάσχισης δέντρου

```
void RecBSTInorder(BinTreePointer Root)  {
    if (Root != NULL) {
        RecBSTInorder(Root->LChild);
        printf("%c ",Root->Data);
        RecBSTInorder(Root->RChild);
      }
}
```



P	Εμφάνισε τα περιεχόμενα του κόμβου.	P
P	Κλήση της RecBSTInorder με δείκτη στη ρίζα (NULL) του δεξιού υποδέντρου.	
null	null, επιστροφή στον κόμβο πατέρα.	
P	Επιστροφή στον κόμβο πατέρα (*κλείσιμο αναδρομής με Root το P*).	
K	Τέλος της διαδικασίας (*κλείσιμο αναδρομής με Root το K*). Η διάσχιση ολοκληρώθηκε.	

Παράδειγμα ενδοδιατεταγμένης διάσχισης δέντρου

Η ενδοθεματική διάσχιση που εκτελέσαμε για το παραπάνω δέντρο, εμφανίζει τα περιεχόμενα των κόμβων με αλφαβητική σειρά:

D, F, H, K, M, P

κι αυτό συμβαίνει γιατί, στην πραγματικότητα, πρόκειται όχι για ένα απλό δυαδικό δέντρο, αλλά για ένα ΔΔΑ.

Έτσι, λοιπόν, για κάθε κόμβο η τιμή που είναι αποθηκευμένη στο αριστερό παιδί του είναι μικρότερη από την τιμή του κόμβου αυτού και η τελευταία είναι μικρότερη από αυτήν που είναι αποθηκευμένη στο δεξί παιδί του κόμβου.

Επομένως, όλες οι τιμές που βρίσκονται σε κόμβους του αριστερού υποδέντρου ενός κόμβου είναι μικρότερες από την τιμή του κόμβου αυτού και η τιμή του κόμβου είναι με τη σειρά της μικρότερη από όλες τις τιμές των κόμβων του δεξιού υποδέντρου του.

Αφού, λοιπόν, μια ενδοθεματική διάσχιση είναι μια διάσχιση με διάταξη LNR, συμπεραίνουμε ότι η επίσκεψη των κόμβων γίνεται με αλφαβητική σειρά.

Μεταδιατεταγμένη & Προδιατεταγμένη διάσχιση

Για τα άλλα ήδη διατάξεων μπορούν να κατασκευαστούν συναρτήσεις απλά αλλάζοντας τη σειρά των λειτουργιών

L:Διάσχιση του αριστερού υποδέντρου

N:Επίσκεψη της ρίζας

R:Διάσχιση του δεξιού υποδέντρου.

```
//Προδιατεταγμένη διάσχιση
void RecBSTPreorder(BinTreePointer Root){
   if (Root!=NULL) {
      printf("%d ", Root->Data);
      RecBSTPreorder(Root->LChild);
      RecBSTPreorder(Root->RChild);
}
```

```
// Μεταδιατεταγμένη διάσχιση
void RecBSTPostorder(BinTreePointer Root){
   if (Root!=NULL) {
      RecBSTPostorder(Root->LChild);
      RecBSTPostorder (Root->RChild);
      printf ("%d ", Root->Data);
}
```

```
void RecBSTInorder(BinTreePointer Root)  {
   if (Root != NULL) {
      RecBSTInorder(Root->LChild);
      printf("%d ", Root->Data);
      RecBSTInorder(Root->RChild);
     }
 }
```

Αλγόριθμος αναδρομικής αναζήτησης σε ΔΔΑ

Στη συνέχεια θα κατασκευάσουμε και θα υλοποιήσουμε ένα **αλγόριθμο αναζήτησης ενός ΔΔΑ αναδρομικά**.

Ήδη κατασκευάσαμε ένα αλγόριθμο αναζήτησης ενός ΔΔΑ και υλοποιήσαμε την διαδικασία BSTSearch, σύμφωνα με την οποία:

- ξεκινάμε από την ρίζα, την συγκρίνουμε με το ζητούμενο στοιχείο και, αν δεν είναι ίσο με αυτήν, τότε
- συνεχίζουμε την αναζήτηση στο αριστερό υποδέντρο, στην περίπτωση που η ρίζα είναι μεγαλύτερη, ή με το δεξί υποδέντρο αν είναι μικρότερη.
- Αν το υποδέντρο με το οποίο συνεχίζουμε είναι κενό, τότε το στοιχείο δεν υπάρχει στο δέντρο, ενώ,
- αν δεν είναι κενό, εκτελούμε αναζήτηση στο υποδέντρο *με τον ίδιο ακριβώς τρόπο* που κάναμε και στο αρχικό δέντρο.

Αυτό δείχνει ότι στην πραγματικότητα σκεφτόμαστε αναδρομικά κι επομένως μπορούμε να χρησιμοποιήσουμε ένα αναδρομικό αλγόριθμο και την αντίστοιχη αναδρομική διαδικασία:

Διαδικασία αναδρομικής αναζήτησης σε ΔΔΑ

void RecBSTSearch(BinTreePointer Root, BinTreeElementType KeyValue,
boolean *Found, BinTreePointer *LocPtr)

/*Δέχεται: Ένα ΔΔΑ με το δείκτη *Root* να δείχνει στη ρίζα του και μια τιμή KeyValue.

Λειτουργία: Εκτελεί αναδρομική αναζήτηση στο ΔΔΑ για έναν κόμβο με τιμή *KeyValue* στο πεδίο κλειδί του.

Επιστρέφει: Η *Found* έχει τιμή TRUE και ο δείκτης *LocPtr* δείχνει στον κόμβο που περιέχει την τιμή *KeyValue*, αν η αναζήτηση είναι επιτυχής. Διαφορετικά η *Found* έχει τιμή FALSE.*/

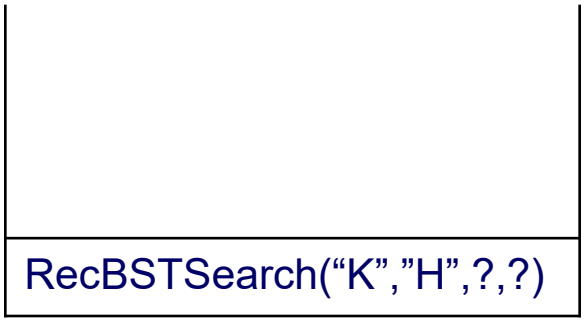
Διαδικασία αναδρομικής αναζήτησης σε ΔΔΑ

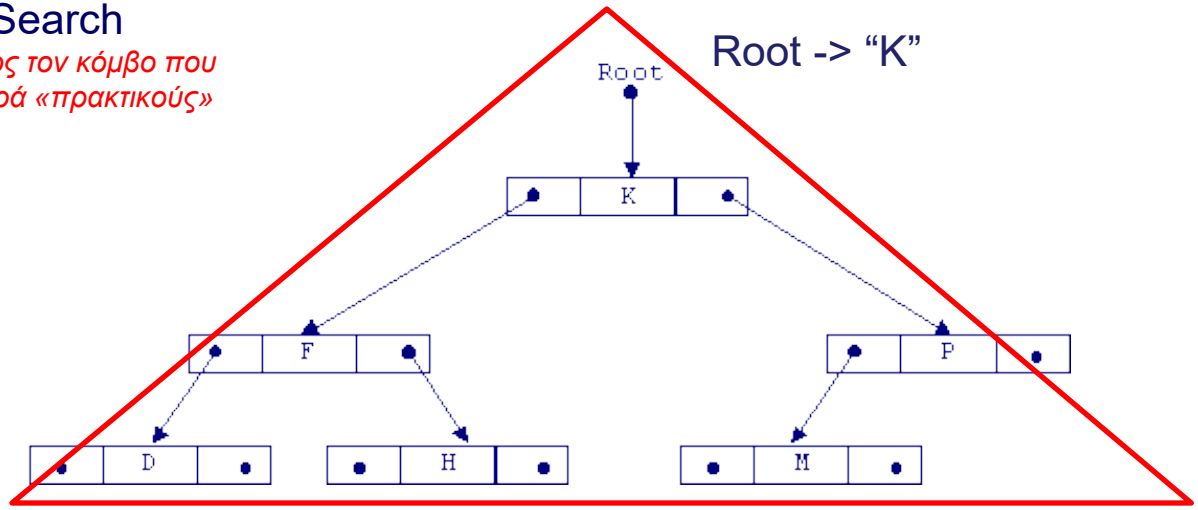
```
{
    if (BSTEmpty(Root))
        *Found = FALSE;
    else
        if (KeyValue < Root->Data)
            RecBSTSearch(Root->LChild, KeyValue, &(*Found), &(*LocPtr));
        else
            if (KeyValue > Root->Data)
                RecBSTSearch(Root->RChild, KeyValue, &(*Found), &(*LocPtr));
            else
                {
                    *Found = TRUE;
                    *LocPtr = Root;
                }
}
```

Αναζήτηση του “H”

Αναδρομικές κλήσεις της RecBSTSearch

Η 1η παράμετρος δεν είναι το “K” αλλά δείκτης προς τον κόμβο που περιέχει το “K”. Συμβολίζεται έτσι για λόγους καθαρά «πρακτικούς»

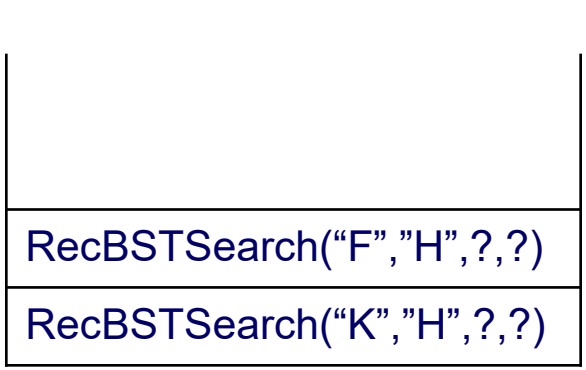


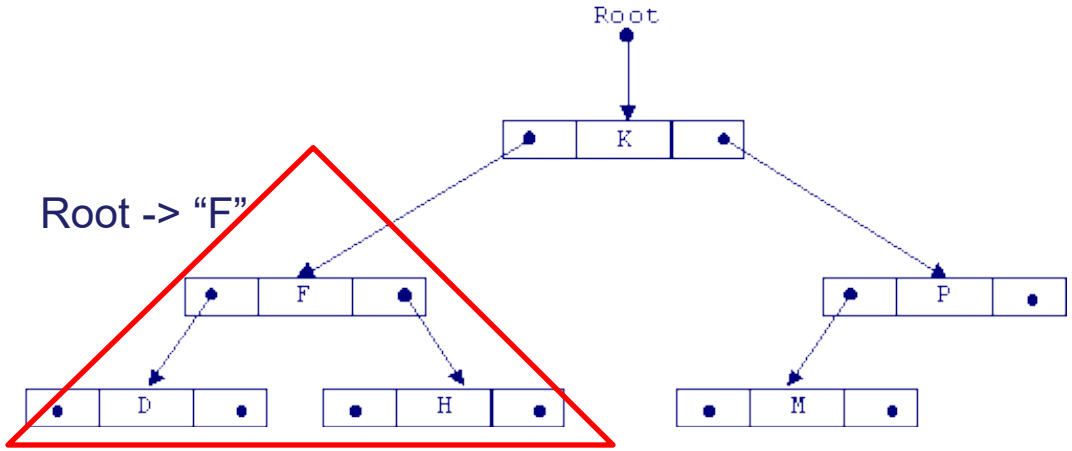



```
void RecBSTSearch(BinTreePointer Root, BinTreeElementType KeyValue,
                                boolean *Found, BinTreePointer *LocPtr)
{
   if (BSTEmpty(Root))
      *Found = FALSE;
   else
      if (KeyValue < Root->Data)
         RecBSTSearch(Root->LChild, KeyValue, &(*Found), &(*LocPtr));
      else
         if (KeyValue > Root->Data)
            RecBSTSearch(Root->RChild, KeyValue, &(*Found), &(*LocPtr));
         else
            {
               *Found = TRUE;
               *LocPtr = Root;
            }
}
```

Αναζήτηση του “H”

Αναδρομικές κλήσεις της RecBSTSearch



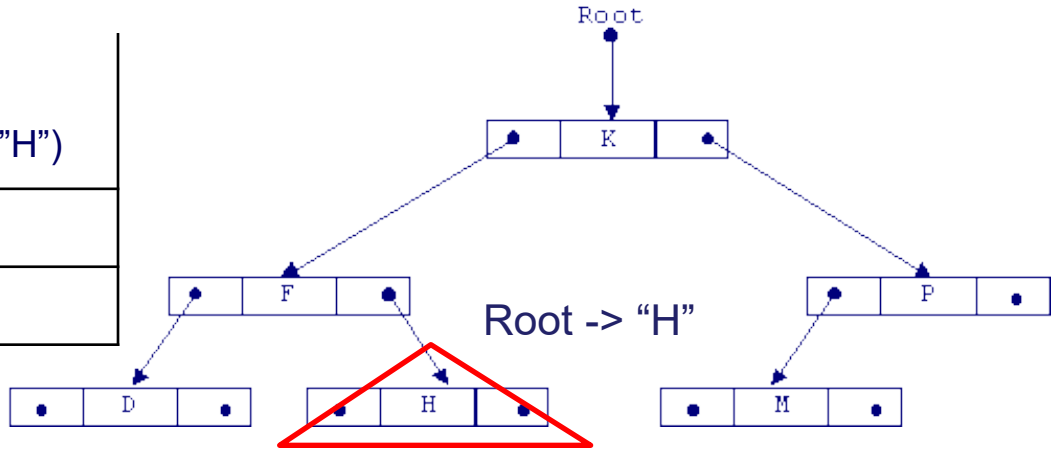



```
void RecBSTSearch(BinTreePointer Root, BinTreeElementType KeyValue,
                                boolean *Found, BinTreePointer *LocPtr)
{
   if (BSTEmpty(Root))
      *Found = FALSE;
   else
      if (KeyValue < Root->Data)
         RecBSTSearch(Root->LChild, KeyValue, &(*Found), &(*LocPtr));
      else
         if (KeyValue > Root->Data)
            RecBSTSearch(Root->RChild, KeyValue, &(*Found), &(*LocPtr));
         else
            {
               *Found = TRUE;
               *LocPtr = Root;
            }
}
```

Αναζήτηση του “H”

Αναδρομικές κλήσεις της RecBSTSearch

RecBSTSearch(“H”,”H”,TRUE,*LocPtr->”H”)
RecBSTSearch(“F”,”H”,?,?)
RecBSTSearch(“K”,”H”,?,?)

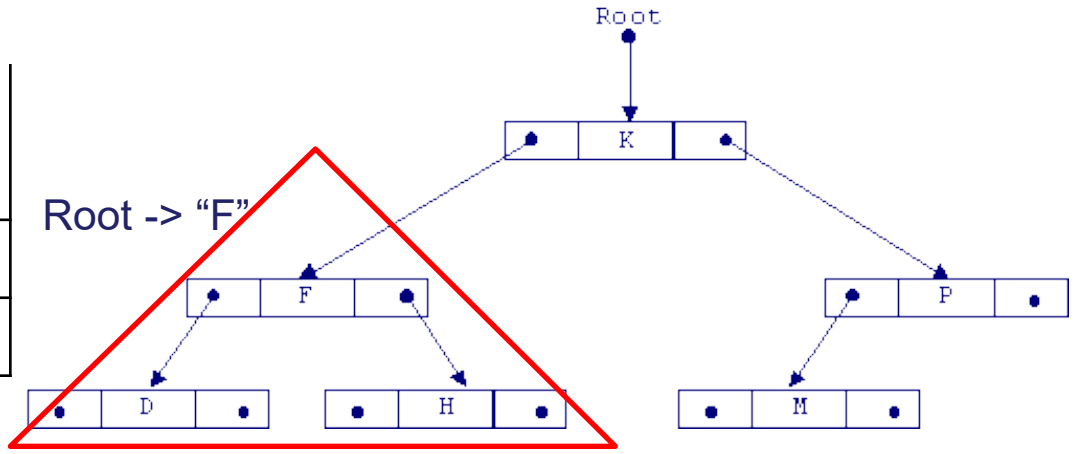



```
void RecBSTSearch(BinTreePointer Root, BinTreeElementType KeyValue,
                                  boolean *Found, BinTreePointer *LocPtr)
{
   if (BSTEmpty(Root))
      *Found = FALSE;
   else
      if (KeyValue < Root->Data)
         RecBSTSearch(Root->LChild, KeyValue, &(*Found), &(*LocPtr));
      else
         if (KeyValue > Root->Data)
            RecBSTSearch(Root->RChild, KeyValue, &(*Found), &(*LocPtr));
         else
            {
               *Found = TRUE;
               *LocPtr = Root;
            }
}
```

Αναζήτηση του “H”

Αναδρομικές κλήσεις της RecBSTSearch
(“κλείσιμο” αναδρομικών κλήσεων)

RecBSTSearch(“F”,”H”, TRUE,*LocPtr->”H”)
RecBSTSearch(“K”,”H”,?,?)

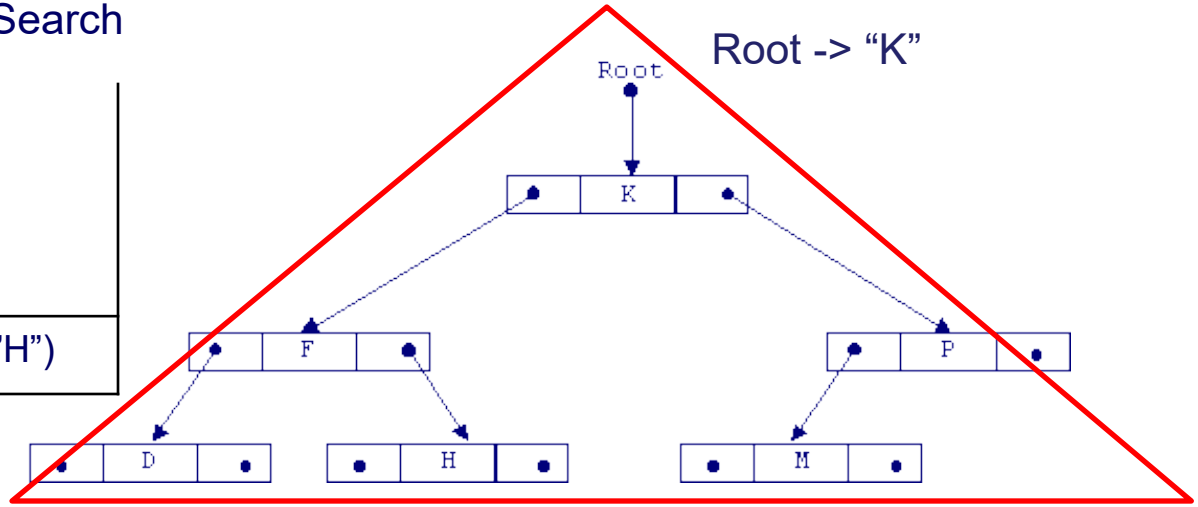



```
void RecBSTSearch(BinTreePointer Root, BinTreeElementType KeyValue,
                                boolean *Found, BinTreePointer *LocPtr)
{
   if (BSTEmpty(Root))
      *Found = FALSE;
   else
      if (KeyValue < Root->Data)
         RecBSTSearch(Root->LChild, KeyValue, &(*Found), &(*LocPtr));
      else
         if (KeyValue > Root->Data)
            RecBSTSearch(Root->RChild, KeyValue, &(*Found), &(*LocPtr));
         else
            {
               *Found = TRUE;
               *LocPtr = Root;
            }
}
```

Αναζήτηση του “H”

Αναδρομικές κλήσεις της RecBSTSearch
(“κλείσιμο” αναδρομικών κλήσεων)

RecBSTSearch(“K”,”H”, TRUE,*LocPtr->”H”)

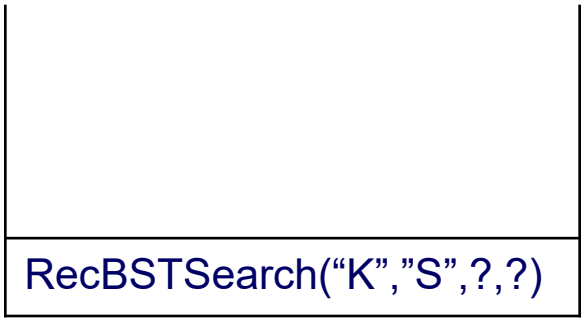


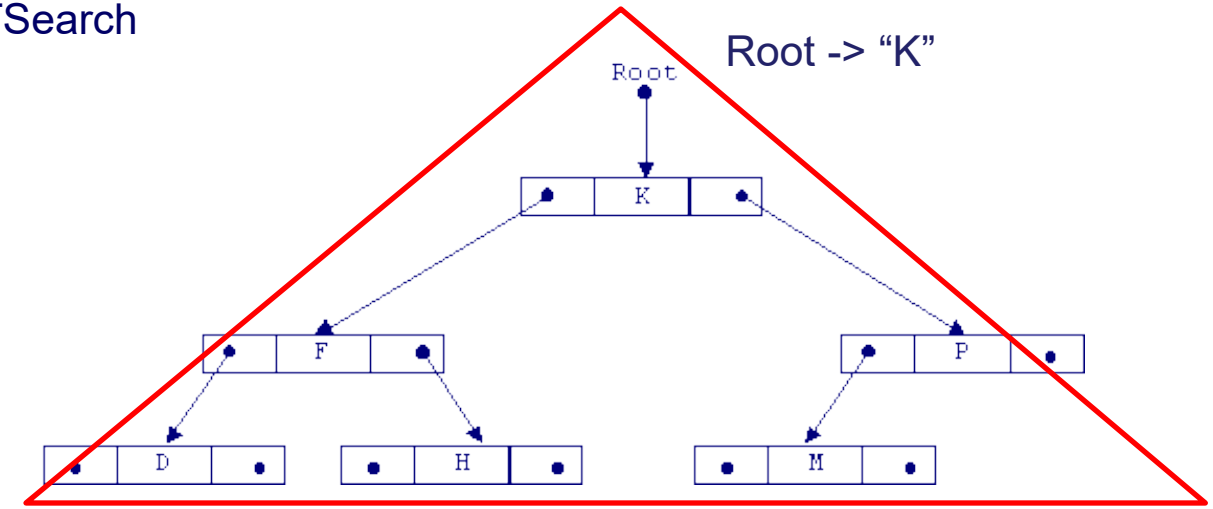

```
void RecBSTSearch(BinTreePointer Root, BinTreeElementType KeyValue,
                                boolean *Found, BinTreePointer *LocPtr)
{
   if (BSTEmpty(Root))
      *Found = FALSE;
   else
      if (KeyValue < Root->Data)
         RecBSTSearch(Root->LChild, KeyValue, &(*Found), &(*LocPtr));
      else
         if (KeyValue > Root->Data)
            RecBSTSearch(Root->RChild, KeyValue, &(*Found), &(*LocPtr));
         else
            {
               *Found = TRUE;
               *LocPtr = Root;
            }
}
```

RecBSTSearch θα επιστρέψει στην καλούσα συνάρτηση TRUE & τη διεύθυνση του κόμβου H

Αναζήτηση του “S”

Αναδρομικές κλήσεις της RecBSTSearch

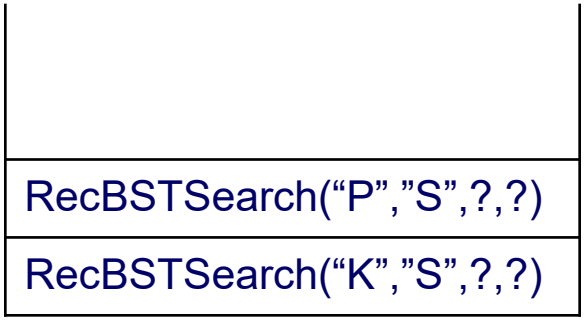


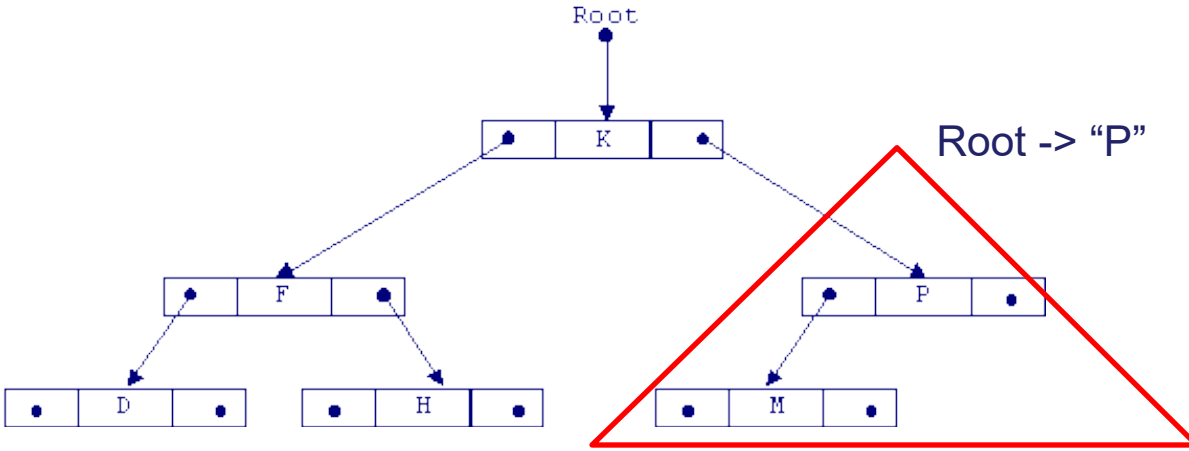



```
void RecBSTSearch(BinTreePointer Root, BinTreeElementType KeyValue,
                                boolean *Found, BinTreePointer *LocPtr)
{
   if (BSTEmpty(Root))
      *Found = FALSE;
   else
      if (KeyValue < Root->Data)
         RecBSTSearch(Root->LChild, KeyValue, &(*Found), &(*LocPtr));
      else
         if (KeyValue > Root->Data)
            RecBSTSearch(Root->RChild, KeyValue, &(*Found), &(*LocPtr));
         else
            {
               *Found = TRUE;
               *LocPtr = Root;
            }
}
```

Αναζήτηση του “S”

Αναδρομικές κλήσεις της RecBSTSearch



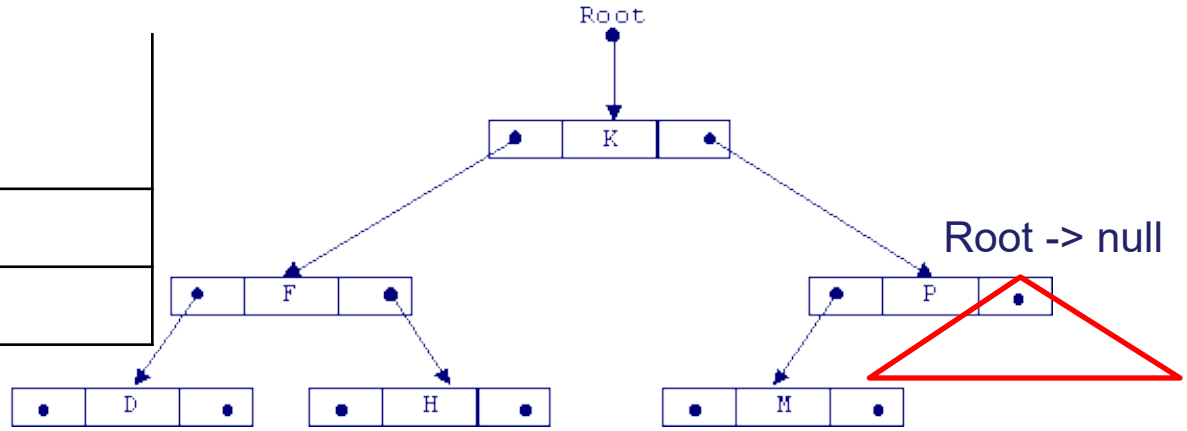



```
void RecBSTSearch(BinTreePointer Root, BinTreeElementType KeyValue,
                                boolean *Found, BinTreePointer *LocPtr)
{
   if (BSTEmpty(Root))
      *Found = FALSE;
   else
      if (KeyValue < Root->Data)
         RecBSTSearch(Root->LChild, KeyValue, &(*Found), &(*LocPtr));
      else
         if (KeyValue > Root->Data)
            RecBSTSearch(Root->RChild, KeyValue, &(*Found), &(*LocPtr));
         else
            {
               *Found = TRUE;
               *LocPtr = Root;
            }
}
```

Αναζήτηση του “S”

Αναδρομικές κλήσεις της RecBSTSearch

RecBSTSearch(“null”,”S”,FALSE, ?)
RecBSTSearch(“P”,”S”,?,?)
RecBSTSearch(“K”,”S”,?,?)



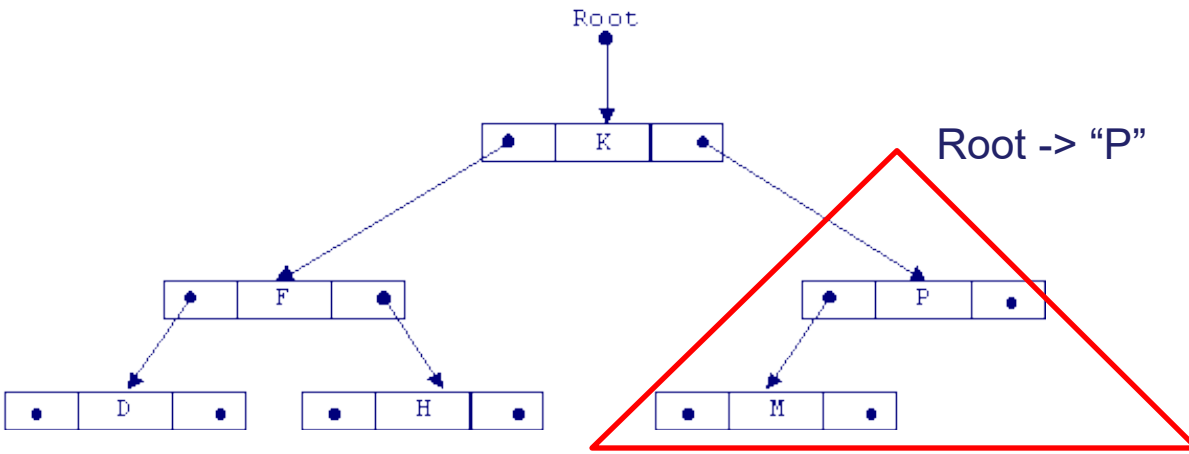

```
void RecBSTSearch(BinTreePointer Root, BinTreeElementType KeyValue,
                                  boolean *Found, BinTreePointer *LocPtr)
{
   if (BSTEmpty(Root))
      *Found = FALSE;
   else
      if (KeyValue < Root->Data)
         RecBSTSearch(Root->LChild, KeyValue, &(*Found), &(*LocPtr));
      else
         if (KeyValue > Root->Data)
            RecBSTSearch(Root->RChild, KeyValue, &(*Found), &(*LocPtr));
         else
            {
               *Found = TRUE;
               *LocPtr = Root;
            }
}
```

Αναζήτηση του “S”

Αναδρομικές κλήσεις της RecBSTSearch

(“κλείσιμο” αναδρομικών κλήσεων)

RecBSTSearch(“P”,”S”,FALSE,?)
RecBSTSearch(“K”,”S”,?,?)

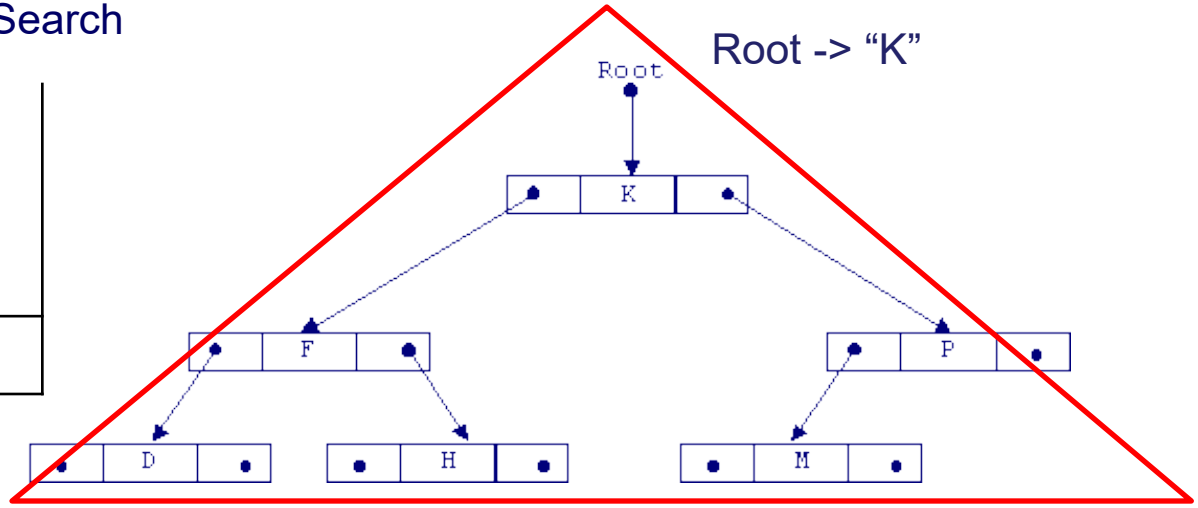



```
void RecBSTSearch(BinTreePointer Root, BinTreeElementType KeyValue,
                                boolean *Found, BinTreePointer *LocPtr)
{
   if (BSTEmpty(Root))
      *Found = FALSE;
   else
      if (KeyValue < Root->Data)
         RecBSTSearch(Root->LChild, KeyValue, &(*Found), &(*LocPtr));
      else
         if (KeyValue > Root->Data)
            RecBSTSearch(Root->RChild, KeyValue, &(*Found), &(*LocPtr));
         else
            {
               *Found = TRUE;
               *LocPtr = Root;
            }
}
```

Αναζήτηση του “S”

Αναδρομικές κλήσεις της RecBSTSearch
(“κλείσιμο” αναδρομικών κλήσεων)

RecBSTSearch(“K”,”S”, FALSE,?)




```
void RecBSTSearch(BinTreePointer Root, BinTreeElementType KeyValue,
                                boolean *Found, BinTreePointer *LocPtr)
{
   if (BSTEmpty(Root))
      *Found = FALSE;
   else
      if (KeyValue < Root->Data)
         RecBSTSearch(Root->LChild, KeyValue, &(*Found), &(*LocPtr));
      else
         if (KeyValue > Root->Data)
            RecBSTSearch(Root->RChild, KeyValue, &(*Found), &(*LocPtr));
         else
            {
               *Found = TRUE;
               *LocPtr = Root;
            }
}
```

RecBSTSearch θα επιστρέψει στην καλούσα συνάρτηση FALSE & «απροσδιόριστη» διεύθυνση για την 4η παράμετρο της συνάρτησης

Αναδρομική διαδικασία εισαγωγής στοιχείου

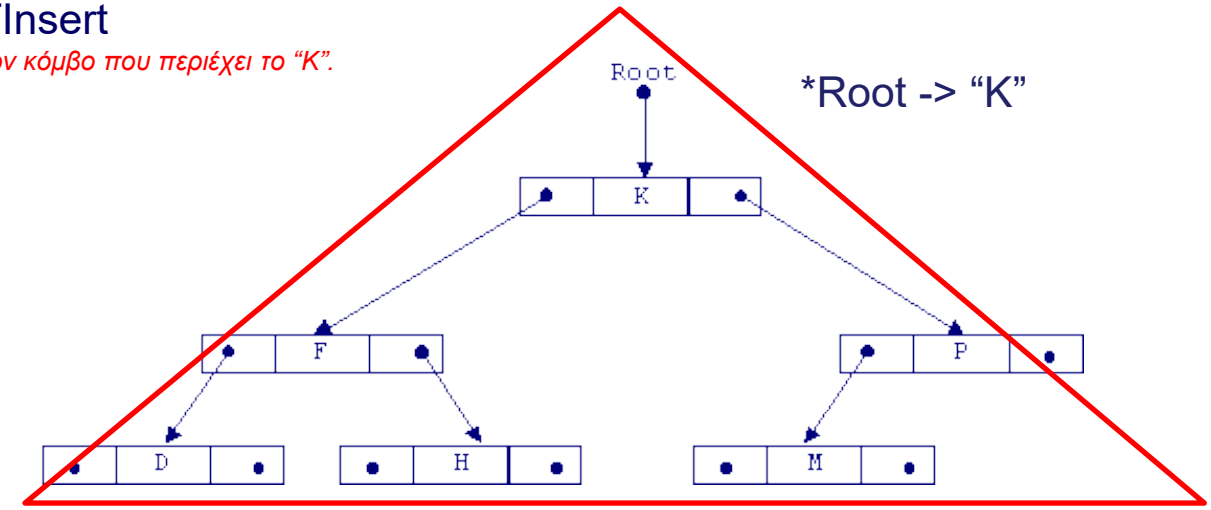
```
void RecBSTInsert(BinTreePointer *Root, BinTreeElementType Item)
/*Δέχεται:        Ένα ΔΔΑ με το δείκτη Root να δείχνει στη ρίζα του και ένα
                  στοιχείο Item.
 Λειτουργία:      Εισάγει αναδρομικά το στοιχείο Item στο ΔΔΑ.
 Επιστρέφει:      Το τροποποιημένο ΔΔΑ με τον δείκτη Root να δείχνει στη ρίζα του.*/
{
   if (BSTEmpty(*Root)) {
      (*Root) = (BinTreePointer)malloc(sizeof (struct BinTreeNode));
      (*Root) ->Data = Item;
      (*Root) ->LChild = NULL;
      (*Root) ->RChild = NULL;
   }
   else
         if (Item < (*Root) ->Data)
               RecBSTInsert(&(*Root) ->LChild,Item);
         else if (Item > (*Root) ->Data)
               RecBSTInsert(&(*Root) ->RChild,Item);
            else
               printf("ΤΟ ΣΤΟΙΧΕΙΟ ΕΙΝΑΙ ΗΔΗ ΣΤΟ ΔΔΑ\n");
}
```

Εισαγωγή του “L”

Αναδρομικές κλήσεις της RecBSTInsert

Η 1η παράμετρος δεν είναι το “K” αλλά δείκτης προς τον κόμβο που περιέχει το “K”. Συμβολίζεται έτσι για λόγους καθαρά «πρακτικούς»

RecBSTInsert(“K”,”L”)

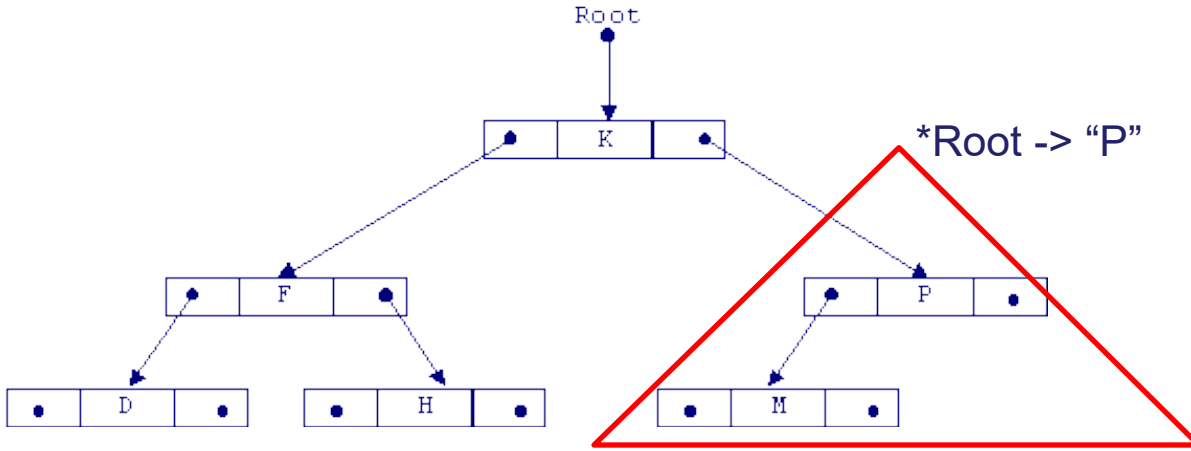



```
void RecBSTInsert(BinTreePointer *Root, BinTreeElementType Item) {
   if (BSTEmpty(*Root)) {
      (*Root) = (BinTreePointer)malloc(sizeof (struct BinTreeNode));
      (*Root) ->Data = Item;
      (*Root) ->LChild = NULL;
      (*Root) ->RChild = NULL;
   }
   else
            if (Item < (*Root) ->Data)
                     RecBSTInsert(&(*Root) ->LChild,Item);
            else if (Item > (*Root) ->Data)
                     RecBSTInsert(&(*Root) ->RChild,Item);
                else
                     printf("TO STOIXEIO EINAI HDH STO DDA\n");
}
```

Εισαγωγή του “L”

Αναδρομικές κλήσεις της RecBSTInsert

RecBSTInsert(“P”,”L”) RecBSTInsert(“K”->RChild, “L”)
RecBSTInsert(“K”,”L”)




```
void RecBSTInsert(BinTreePointer *Root, BinTreeElementType Item) {
    if (BSTEmpty(*Root)) {
        (*Root) = (BinTreePointer)malloc(sizeof (struct BinTreeNode));
        (*Root) ->Data = Item;
        (*Root) ->LChild = NULL;
        (*Root) ->RChild = NULL;
    }
    else
            if (Item < (*Root) ->Data)
                    RecBSTInsert(&(*Root) ->LChild,Item);
            else if (Item > (*Root) ->Data)
                    RecBSTInsert(&(*Root) ->RChild,Item);
                else
                    printf(“ΤΟ ΣΤΟΙΧΕΙΟ ΕΙΝΑΙ ΗΔΗ ΣΤΟ ΔΔΑ\n");
}
```

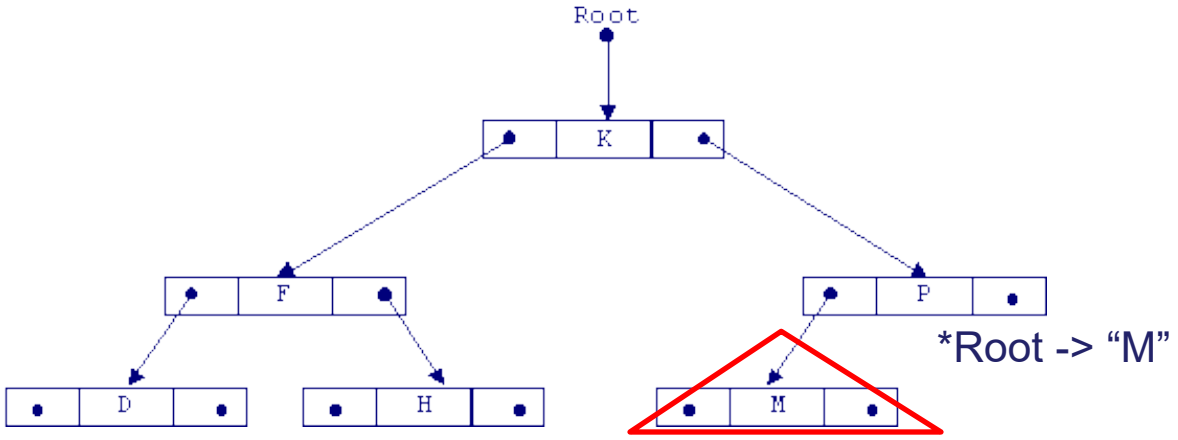
Εισαγωγή του “L”

Αναδρομικές κλήσεις της RecBSTInsert

RecBSTInsert(“M”,”L”)
RecBSTInsert(“P”->LChild, “L”)

RecBSTInsert(“P”,”L”)
RecBSTInsert(“K”->RChild, “L”)

RecBSTInsert(“K”,”L”)

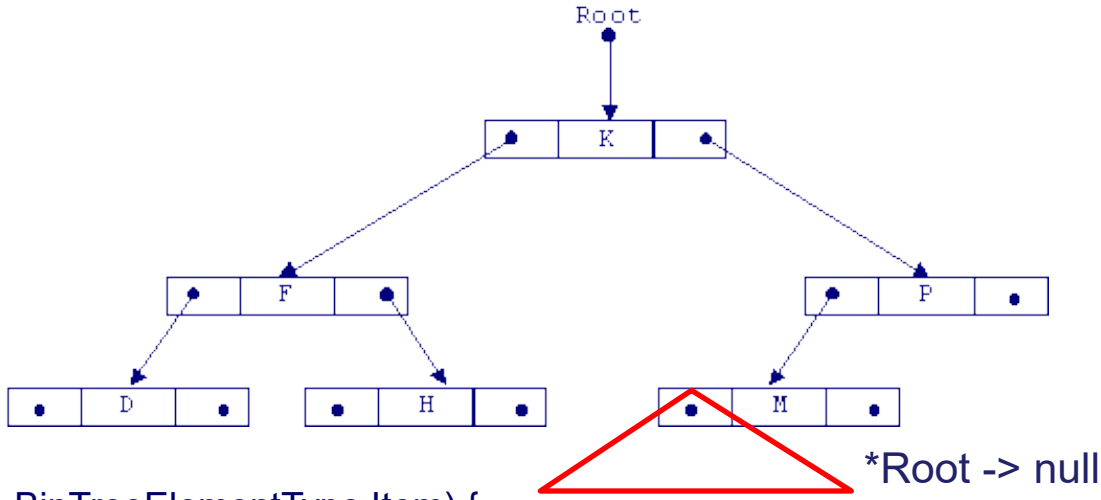



```
void RecBSTInsert(BinTreePointer *Root, BinTreeElementType Item) {
   if (BSTEmpty(*Root)) {
      (*Root) = (BinTreePointer)malloc(sizeof (struct BinTreeNode));
      (*Root) ->Data = Item;
      (*Root) ->LChild = NULL;
      (*Root) ->RChild = NULL;
   }
   else
            if (Item < (*Root) ->Data)
                     RecBSTInsert(&(*Root) ->LChild,Item);
            else if (Item > (*Root) ->Data)
                     RecBSTInsert(&(*Root) ->RChild,Item);
                else
                     printf(“ΤΟ ΣΤΟΙΧΕΙΟ ΕΙΝΑΙ ΗΔΗ ΣΤΟ ΔΔΑ\n");
}
```

Εισαγωγή του “L”

Αναδρομικές κλήσεις της RecBSTInsert

RecBSTInsert(null,”L”) RecBSTInsert(“M”->LChild, “L”)
RecBSTInsert(“M”,”L”) RecBSTInsert(“P”->LChild, “L”)
RecBSTInsert(“P”,”L”) RecBSTInsert(“K”->RChild, “L”)
RecBSTInsert(“K”,”L”)




```
void RecBSTInsert(BinTreePointer *Root, BinTreeElementType Item) {
   if (BSTEmpty(*Root)) {
      (*Root) = (BinTreePointer)malloc(sizeof (struct BinTreeNode));
      (*Root) ->Data = Item;
      (*Root) ->LChild = NULL;
      (*Root) ->RChild = NULL;
   }
   else
            if (Item < (*Root) ->Data)
                     RecBSTInsert(&(*Root) ->LChild,Item);
            else if (Item > (*Root) ->Data)
                     RecBSTInsert(&(*Root) ->RChild,Item);
                else
                     printf(“ΤΟ ΣΤΟΙΧΕΙΟ ΕΙΝΑΙ ΗΔΗ ΣΤΟ ΔΔΑ\n");
}
```

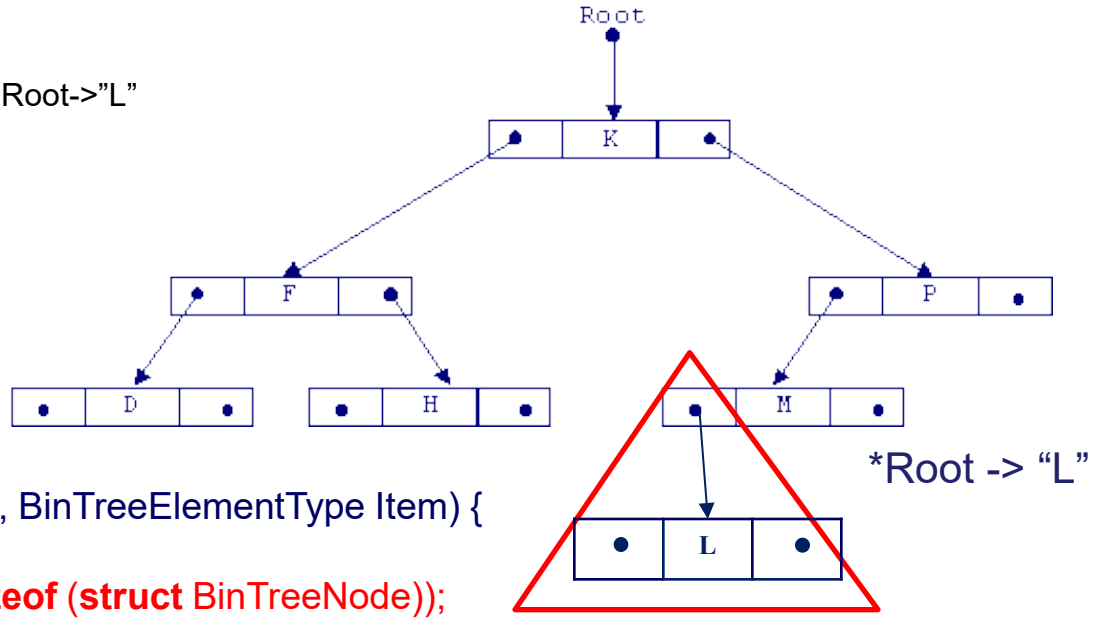
Εισαγωγή του “L”

Αναδρομικές κλήσεις της RecBSTInsert

(“κλείσιμο” αναδρομικών κλήσεων)

RecBSTInsert(“L”,”L”) RecBSTInsert(“M”->LChild, “L”)
RecBSTInsert(“M”,”L”) RecBSTInsert(“P”->LChild, “L”)
RecBSTInsert(“P”,”L”) RecBSTInsert(“K”->RChild, “L”)
RecBSTInsert(“K”,”L”)

Return *Root->”L”




```
void RecBSTInsert(BinTreePointer *Root, BinTreeElementType Item) {
   if (BSTEmpty(*Root)) {
      (*Root) = (BinTreePointer)malloc(sizeof (struct BinTreeNode));
      (*Root) ->Data = Item;
      (*Root) ->LChild = NULL;
      (*Root) ->RChild = NULL;
   }
   else
            if (Item < (*Root) ->Data)
                     RecBSTInsert(&(*Root) ->LChild,Item);
            else if (Item > (*Root) ->Data)
                     RecBSTInsert(&(*Root) ->RChild,Item);
                else
                     printf(“ΤΟ ΣΤΟΙΧΕΙΟ ΕΙΝΑΙ ΗΔΗ ΣΤΟ ΔΔΑ\n");
}
```

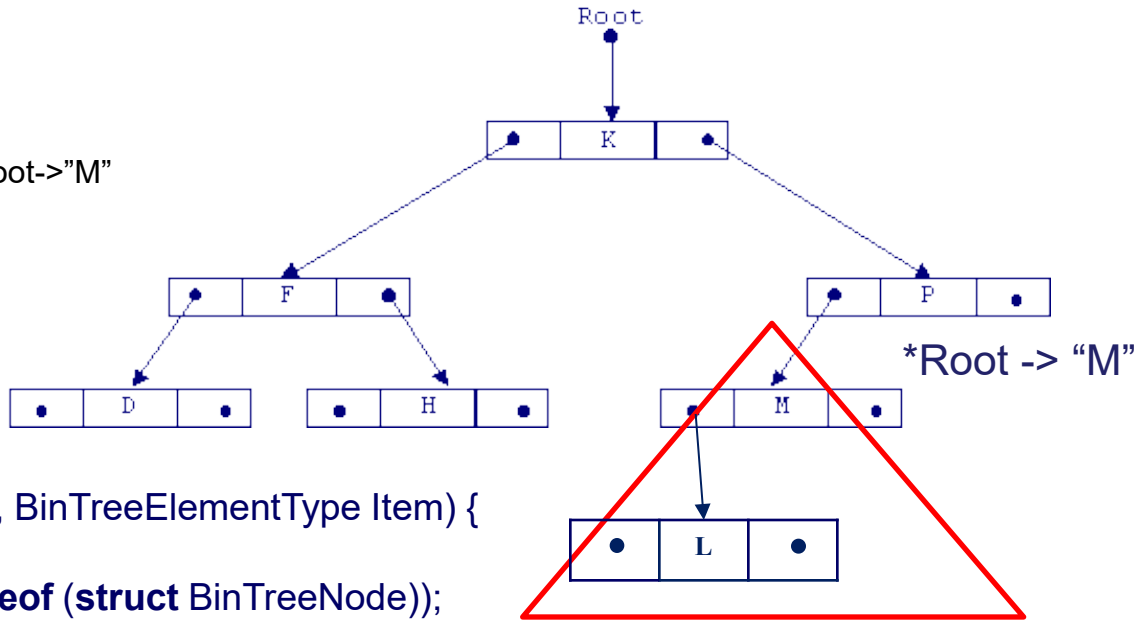
Εισαγωγή του “L”

Αναδρομικές κλήσεις της RecBSTInsert

(“κλείσιμο” αναδρομικών κλήσεων)

RecBSTInsert(“M”,”L”) RecBSTInsert(“P”->LChild, “L”)
RecBSTInsert(“P”,”L”) RecBSTInsert(“K”->RChild, “L”)
RecBSTInsert(“K”,”L”)

Return *Root->”M”




```
void RecBSTInsert(BinTreePointer *Root, BinTreeElementType Item) {
   if (BSTEmpty(*Root)) {
      (*Root) = (BinTreePointer)malloc(sizeof (struct BinTreeNode));
      (*Root) ->Data = Item;
      (*Root) ->LChild = NULL;
      (*Root) ->RChild = NULL;
   }
   else
            if (Item < (*Root) ->Data)
                     RecBSTInsert(&(*Root) ->LChild,Item);
            else if (Item > (*Root) ->Data)
                     RecBSTInsert(&(*Root) ->RChild,Item);
                else
                     printf(“ΤΟ ΣΤΟΙΧΕΙΟ ΕΙΝΑΙ ΗΔΗ ΣΤΟ ΔΔΑ\n");
}
```

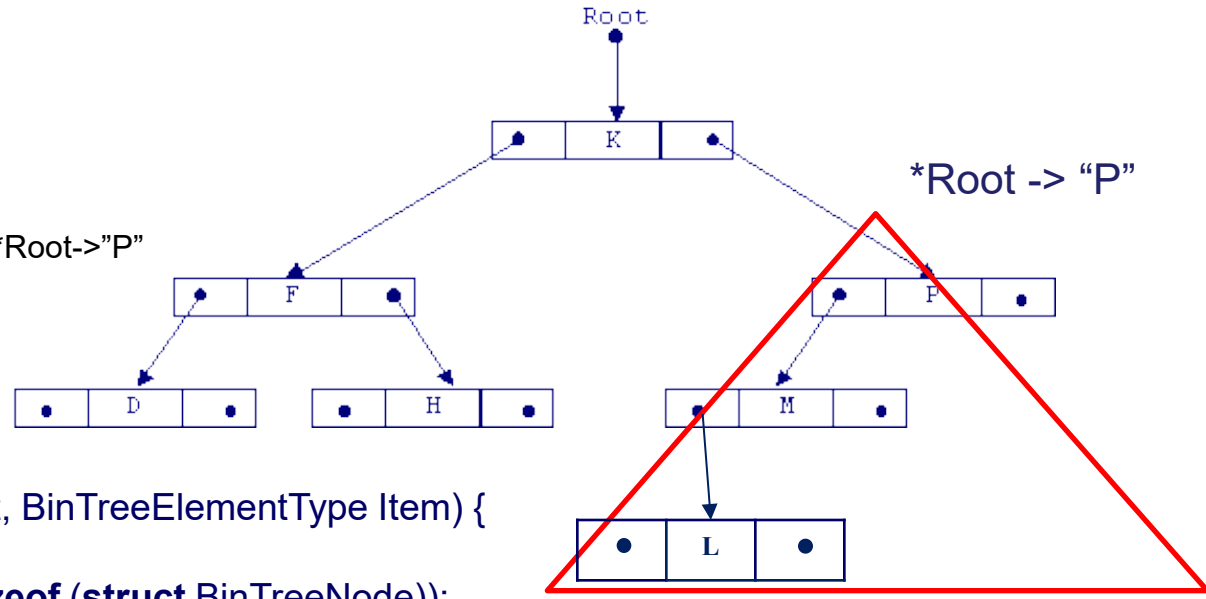
Εισαγωγή του “L”

Αναδρομικές κλήσεις της RecBSTInsert

(“κλείσιμο” αναδρομικών κλήσεων)

RecBSTInsert(“P”,”L”) RecBSTInsert(“K”->RChild, “L”)
RecBSTInsert(“K”,”L”)

Return *Root->”P”



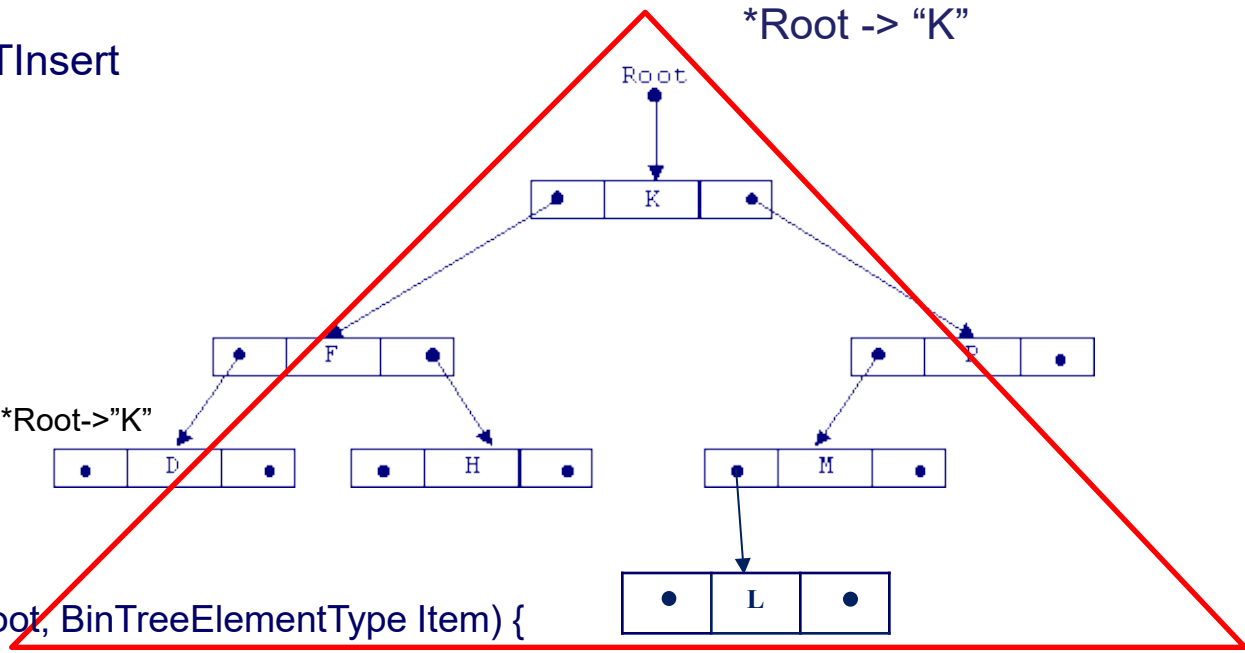

```
void RecBSTInsert(BinTreePointer *Root, BinTreeElementType Item) {
   if (BSTEmpty(*Root)) {
      (*Root) = (BinTreePointer)malloc(sizeof (struct BinTreeNode));
      (*Root) ->Data = Item;
      (*Root) ->LChild = NULL;
      (*Root) ->RChild = NULL;
   }
   else
            if (Item < (*Root) ->Data)
                     RecBSTInsert(&(*Root) ->LChild,Item);
            else if (Item > (*Root) ->Data)
                     RecBSTInsert(&(*Root) ->RChild,Item);
                else
                     printf(“ΤΟ ΣΤΟΙΧΕΙΟ ΕΙΝΑΙ ΗΔΗ ΣΤΟ ΔΔΑ\n");
}
```

Εισαγωγή του “L”

Αναδρομικές κλήσεις της RecBSTInsert
(“κλείσιμο” αναδρομικών κλήσεων)

RecBSTInsert(“K”,”L”)

Return *Root->”K”




```
void RecBSTInsert(BinTreePointer *Root, BinTreeElementType Item) {
   if (BSTEmpty(*Root)) {
      (*Root) = (BinTreePointer)malloc(sizeof (struct BinTreeNode));
      (*Root) ->Data = Item;
      (*Root) ->LChild = NULL;
      (*Root) ->RChild = NULL;
   }
   else
            if (Item < (*Root) ->Data)
                     RecBSTInsert(&(*Root) ->LChild,Item);
            else if (Item > (*Root) ->Data)
                     RecBSTInsert(&(*Root) ->RChild,Item);
                else
                     printf(“ΤΟ ΣΤΟΙΧΕΙΟ ΕΙΝΑΙ ΗΔΗ ΣΤΟ ΔΔΑ\n");
}
```

Αναδρομική διαδικασία διαγραφής στοιχείου

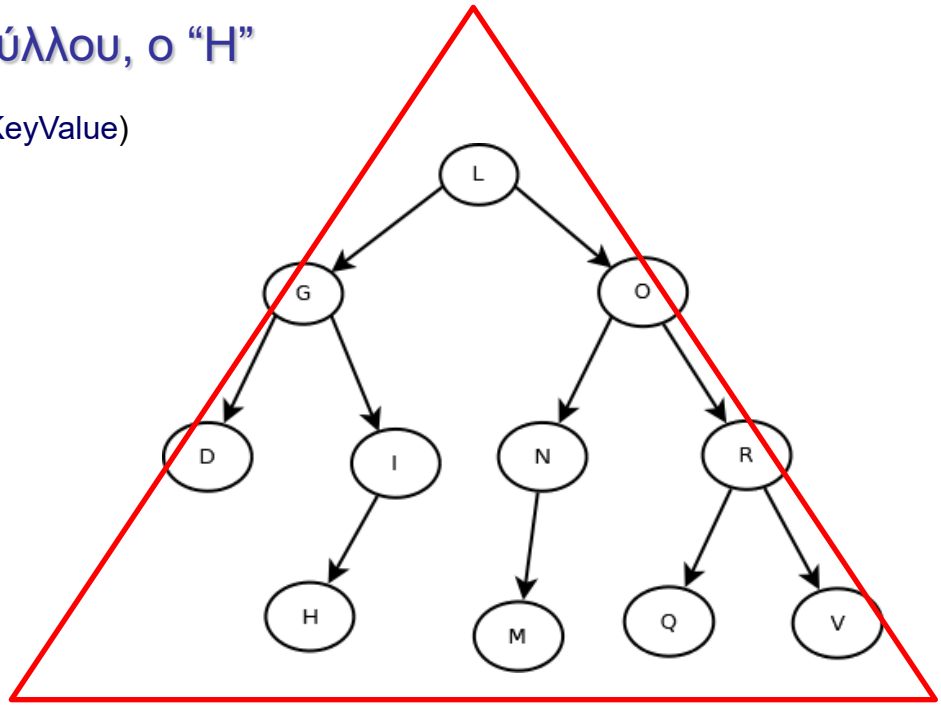
```
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
/*Δέχεται:        Ένα ΔΔΑ με το δείκτη Root να δείχνει στη ρίζα του και μια τιμή
                  KeyValue.
 Λειτουργία:      Προσπαθεί αναδρομικά να βρει έναν κόμβο στο ΔΔΑ που να περιέχει
                  την τιμή KeyValue στο πεδίο κλειδί του τμήματος δεδομένων του και,
                  αν τον βρει, τον διαγράφει από το ΔΔΑ.
 Επιστρέφει:      Το τροποποιημένο ΔΔΑ με τον δείκτη Root να δείχνει στη ρίζα του/
{

   BinTreePointer TempPtr;

    if (BSTEmpty(*Root)) printf("to %d DeN BRE8HKe STO DDA\n", KeyValue);
    else
          if (KeyValue < (*Root)->Data)
                RecBSTDelete(&((*Root)->LChild), KeyValue);
          else if (KeyValue > (*Root)->Data)
                RecBSTDelete(&((*Root)->RChild), KeyValue);
              else
```

Αναδρομική διαδικασία διαγραφής στοιχείου

```
if ((*Root)->LChild == NULL)
{
    TempPtr = *Root;
    *Root = (*Root)->RChild;
    free(TempPtr );
}
else if ((*Root)->RChild == NULL)
  {
        TempPtr = *Root;
        *Root = (*Root)-> LChild;
        free(TempPtr );
  }
  else {
          TempPtr = (*Root)->RChild;
          while (TempPtr->LChild != NULL)
                TempPtr = TempPtr->LChild;
          (*Root)->Data = TempPtr->Data;
          RecBSTDelete(&((*Root)->RChild), (*Root)->Data);
  }
}
```

Διαγραφή φύλλου, ο "Η"

```
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
 {

  BinTreePointer TempPtr;

   if (BSTEmpty(*Root))
       printf("TO STOIXEIO DEN BRE8HKE STO DDA\n");
   else
       if (KeyValue < (*Root)->Data)
           RecBSTDelete(&((*Root)->LChild), KeyValue);
       else if (KeyValue > (*Root)->Data)
           RecBSTDelete(&((*Root)->RChild), KeyValue);
           else
               if ((*Root)->LChild == NULL) {
                  TempPtr = *Root;
                  *Root = (*Root)->RChild;
                  free(TempPtr );
               }
               else if ((*Root)->RChild == NULL) {
                       TempPtr = *Root;
                       *Root = (*Root)-> LChild;
                      free(TempPtr );
               }
               else {
                  TempPtr = (*Root)->RChild;
                  while (TempPtr->LChild != NULL)
                           TempPtr = TempPtr->LChild;
                  (*Root)->Data = TempPtr->Data;
                  RecBSTDelete(&((*Root)->RChild),
                           (*Root)->Data);
               }
}
```

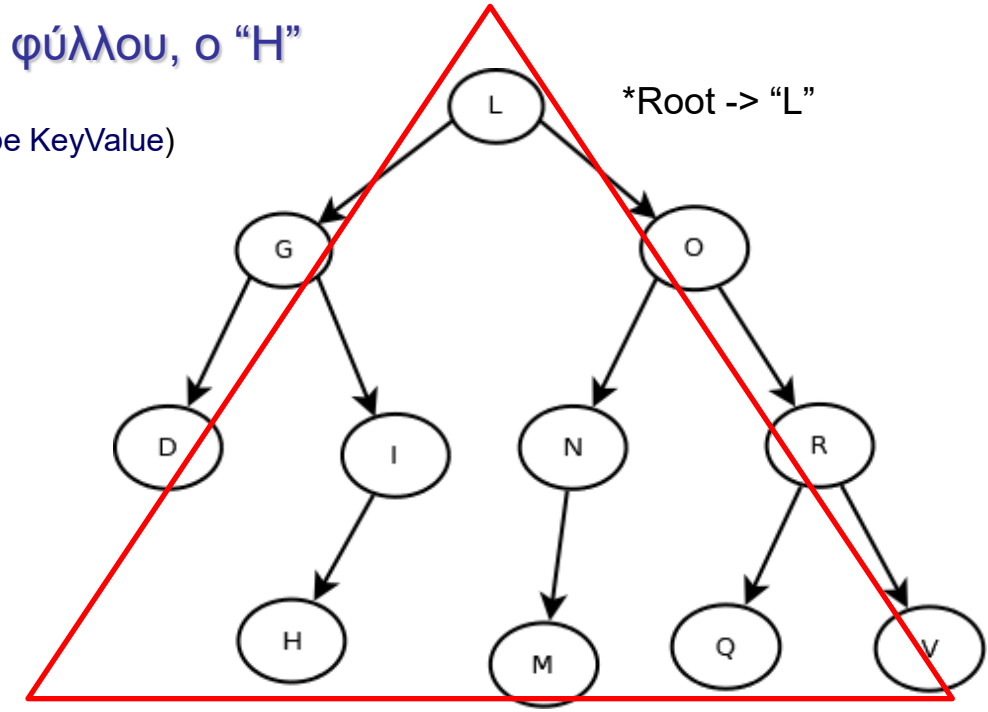


Αναδρομικές κλήσεις της RecBSTDelete

Η 1η παράμετρος δεν είναι το "L" αλλά δείκτης προς τον κόμβο που περιέχει το "L". Συμβολίζεται έτσι για λόγους καθαρά «πρακτικούς»

Διαγραφή φύλλου, ο "Η"

```
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
 {

  BinTreePointer TempPtr;

 if (BSTEmpty(*Root))
     printf("TO STOIXEIO DEN BRE8HKE STO DDA\n");
else
        if (KeyValue < (*Root)->Data)
              RecBSTDelete(&((*Root)->LChild), KeyValue);
        else if (KeyValue > (*Root)->Data)
              RecBSTDelete(&((*Root)->RChild), KeyValue);
              else
                   if ((*Root)->LChild == NULL) {
                      TempPtr = *Root;
                      *Root = (*Root)->RChild;
                     free(TempPtr );
                   }
                   else if ((*Root)->RChild == NULL) {
                             TempPtr = *Root;
                              *Root = (*Root)-> LChild;
                            free(TempPtr );
                   }
                   else {
                      TempPtr = (*Root)->RChild;
                     while (TempPtr->LChild != NULL)
                               TempPtr = TempPtr->LChild;
                      (*Root)->Data = TempPtr->Data;
                      RecBSTDelete(&((*Root)->RChild),
                               (*Root)->Data);
                   }
}
```



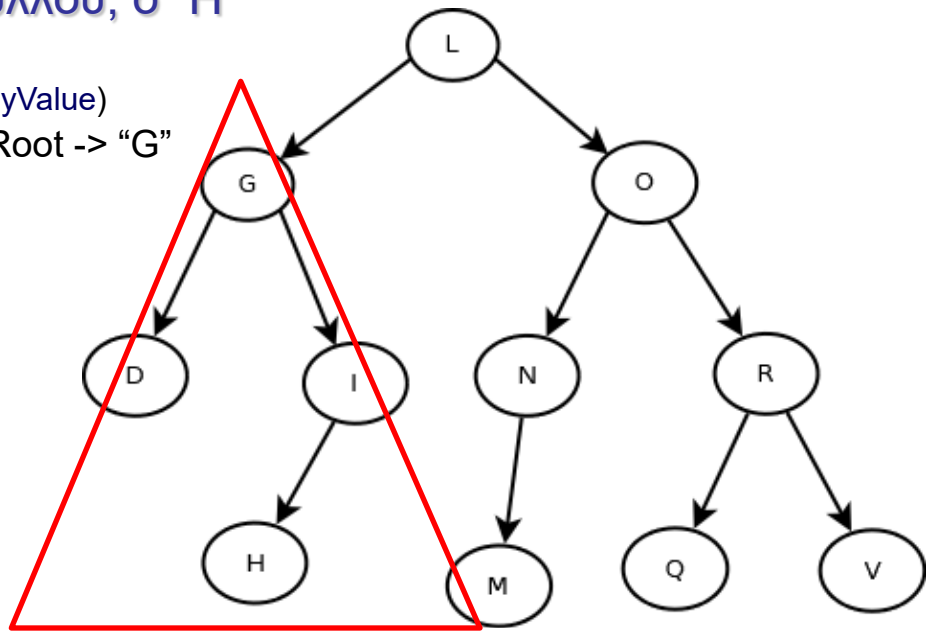
Αναδρομικές κλήσεις της RecBSTDelete

Η 1η παράμετρος δεν είναι το "L" αλλά δείκτης προς τον κόμβο που περιέχει το "L". Συμβολίζεται έτσι για λόγους καθαρά «πρακτικούς»

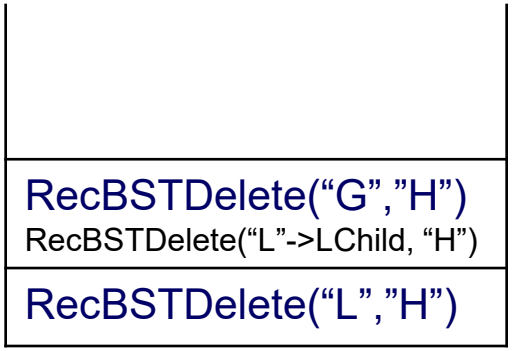
Διαγραφή φύλλου, ο "Η"

```
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
 {

  BinTreePointer TempPtr;

    if (BSTEmpty(*Root))
        printf("TO STOIXEIO DEN BRE8HKE STO DDA\n");
    else
        if (KeyValue < (*Root)->Data)
            RecBSTDelete(&((*Root)->LChild), KeyValue);
        else if (KeyValue > (*Root)->Data)
            RecBSTDelete(&((*Root)->RChild), KeyValue);
            else
                if ((*Root)->LChild == NULL) {
                    TempPtr = *Root;
                    *Root = (*Root)->RChild;
                   free(TempPtr );
                }
                else if ((*Root)->RChild == NULL) {
                        TempPtr = *Root;
                         *Root = (*Root)-> LChild;
                       free(TempPtr );
                }
                else {
                    TempPtr = (*Root)->RChild;
                   while (TempPtr->LChild != NULL)
                            TempPtr = TempPtr->LChild;
                    (*Root)->Data = TempPtr->Data;
                    RecBSTDelete(&((*Root)->RChild),
                            (*Root)->Data);
                }
}
```

*Root -> "G"

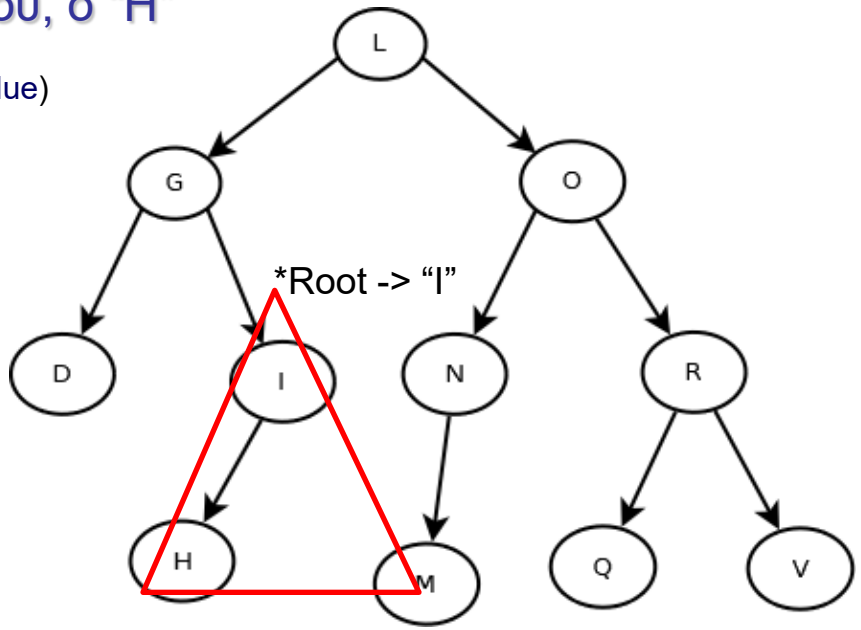


Αναδρομικές κλήσεις της RecBSTDelete



Διαγραφή φύλλου, ο "Η"

```
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
 {

  BinTreePointer TempPtr;

   if (BSTEmpty(*Root))
       printf("TO STOIXEIO DEN BRE8HKE STO DDA\n");
   else
       if (KeyValue < (*Root)->Data)
           RecBSTDelete(&((*Root)->LChild), KeyValue);
       else if (KeyValue > (*Root)->Data)
           RecBSTDelete(&((*Root)->RChild), KeyValue);
           else
               if ((*Root)->LChild == NULL) {
                  TempPtr = *Root;
                  *Root = (*Root)->RChild;
                  free(TempPtr );
               }
               else if ((*Root)->RChild == NULL) {
                       TempPtr = *Root;
                       *Root = (*Root)-> LChild;
                      free(TempPtr );
               }
               else {
                  TempPtr = (*Root)->RChild;
                  while (TempPtr->LChild != NULL)
                           TempPtr = TempPtr->LChild;
                  (*Root)->Data = TempPtr->Data;
                  RecBSTDelete(&((*Root)->RChild),
                           (*Root)->Data);
               }
}
```

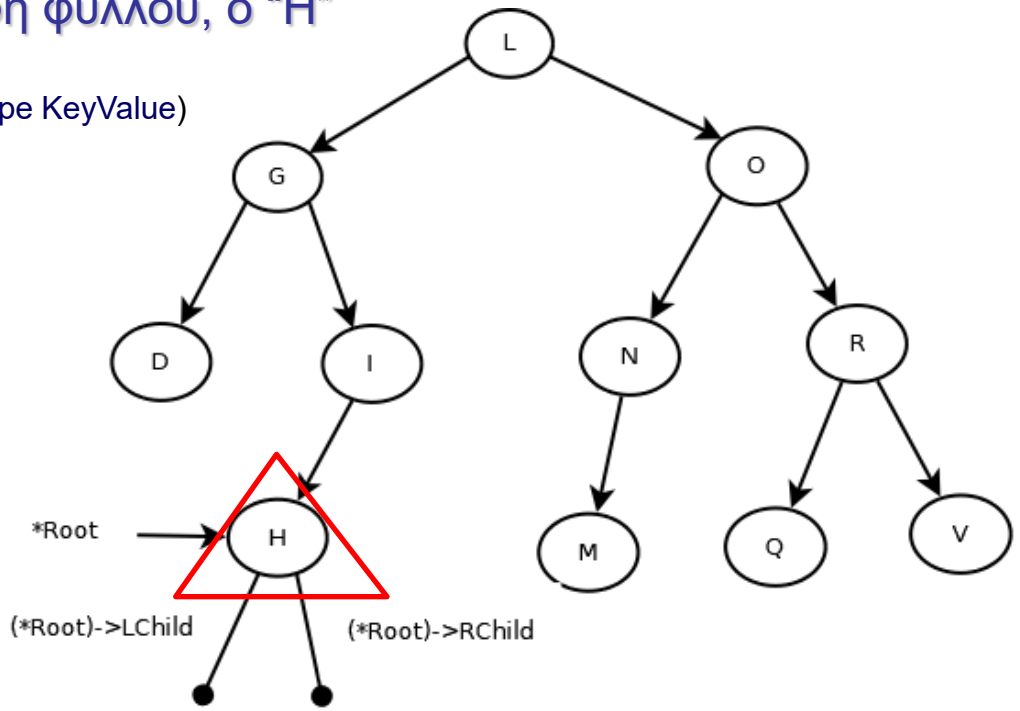


Αναδρομικές κλήσεις της RecBSTDelete

RecBSTDelete("I","H") RecBSTDelete("G"->RChild, "H")
RecBSTDelete("G","H") RecBSTDelete("L"->LChild, "H")
RecBSTDelete("L","H")

Διαγραφή φύλλου, ο "Η"

```
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
 {

  BinTreePointer TempPtr;

   if (BSTEmpty(*Root))
       printf("TO STOIXEIO DEN BRE8HKE STO DDA\n");
   else
        if (KeyValue < (*Root)->Data)
              RecBSTDelete(&((*Root)->LChild), KeyValue);
        else if (KeyValue > (*Root)->Data)
              RecBSTDelete(&((*Root)->RChild), KeyValue);
              else
                   if ((*Root)->LChild == NULL) {
                      TempPtr = *Root;
                      *Root = (*Root)->RChild;
                     free(TempPtr );
                    }
                   else if ((*Root)->RChild == NULL) {
                            TempPtr = *Root;
                             *Root = (*Root)-> LChild;
                           free(TempPtr );
                   }
                   else {
                       TempPtr = (*Root)->RChild;
                      while (TempPtr->LChild != NULL)
                               TempPtr = TempPtr->LChild;
                       (*Root)->Data = TempPtr->Data;
                       RecBSTDelete(&((*Root)->RChild),
                               (*Root)->Data);
                   }
}
```

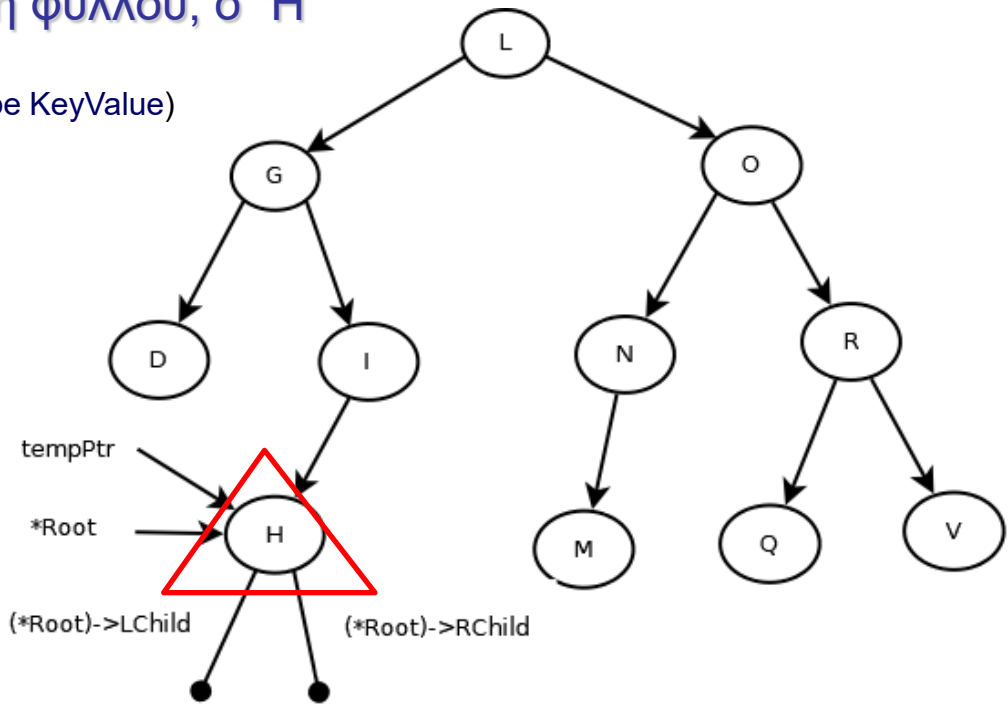


Αναδρομικές κλήσεις της RecBSTDelete

RecBSTDelete("H","H") RecBSTDelete("I"->LChild, "H")
RecBSTDelete("I","H") RecBSTDelete("G"->RChild, "H")
RecBSTDelete("G","H") RecBSTDelete("L"->LChild, "H")
RecBSTDelete("L","H")

Διαγραφή φύλλου, ο "Η"

```
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
 {

  BinTreePointer TempPtr;

   if (BSTEmpty(*Root))
       printf("TO STOIXEIO DEN BRE8HKE STO DDA\n");
   else
       if (KeyValue < (*Root)->Data)
           RecBSTDelete(&((*Root)->LChild), KeyValue);
       else if (KeyValue > (*Root)->Data)
           RecBSTDelete(&((*Root)->RChild), KeyValue);
           else
               if ((*Root)->LChild == NULL) {
                  TempPtr = *Root;
                  *Root = (*Root)->RChild;
                 free(TempPtr );
                }
               else if ((*Root)->RChild == NULL) {
                        TempPtr = *Root;
                         *Root = (*Root)-> LChild;
                       free(TempPtr );
                }
               else {
                   TempPtr = (*Root)->RChild;
                  while (TempPtr->LChild != NULL)
                           TempPtr = TempPtr->LChild;
                   (*Root)->Data = TempPtr->Data;
                   RecBSTDelete(&((*Root)->RChild),
                           (*Root)->Data);
                }
}
```

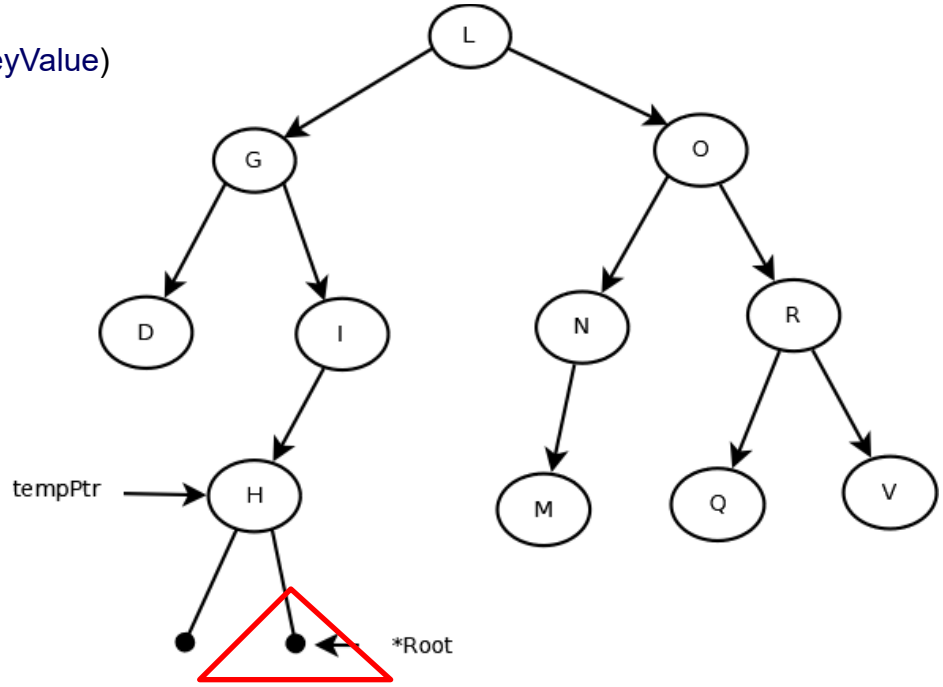


Αναδρομικές κλήσεις της RecBSTDelete

RecBSTDelete("H","H") RecBSTDelete("I"->LChild, "H")
RecBSTDelete("I","H") RecBSTDelete("G"->RChild, "H")
RecBSTDelete("G","H") RecBSTDelete("L"->LChild, "H")
RecBSTDelete("L","H")

Διαγραφή φύλλου, ο "Η"

```
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
 {

  BinTreePointer TempPtr;

   if (BSTEmpty(*Root))
       printf("TO STOIXEIO DEN BRE8HKE STO DDA\n");
   else
        if (KeyValue < (*Root)->Data)
              RecBSTDelete(&((*Root)->LChild), KeyValue);
        else if (KeyValue > (*Root)->Data)
              RecBSTDelete(&((*Root)->RChild), KeyValue);
              else
                   if ((*Root)->LChild == NULL) {
                      TempPtr = *Root;
                      *Root = (*Root)->RChild;
                     free(TempPtr );
                   }
                  else if ((*Root)->RChild == NULL) {
                            TempPtr = *Root;
                             *Root = (*Root)-> LChild;
                           free(TempPtr );
                  }
                  else {
                      TempPtr = (*Root)->RChild;
                     while (TempPtr->LChild != NULL)
                               TempPtr = TempPtr->LChild;
                      (*Root)->Data = TempPtr->Data;
                      RecBSTDelete(&((*Root)->RChild),
                               (*Root)->Data);
                  }
}
```

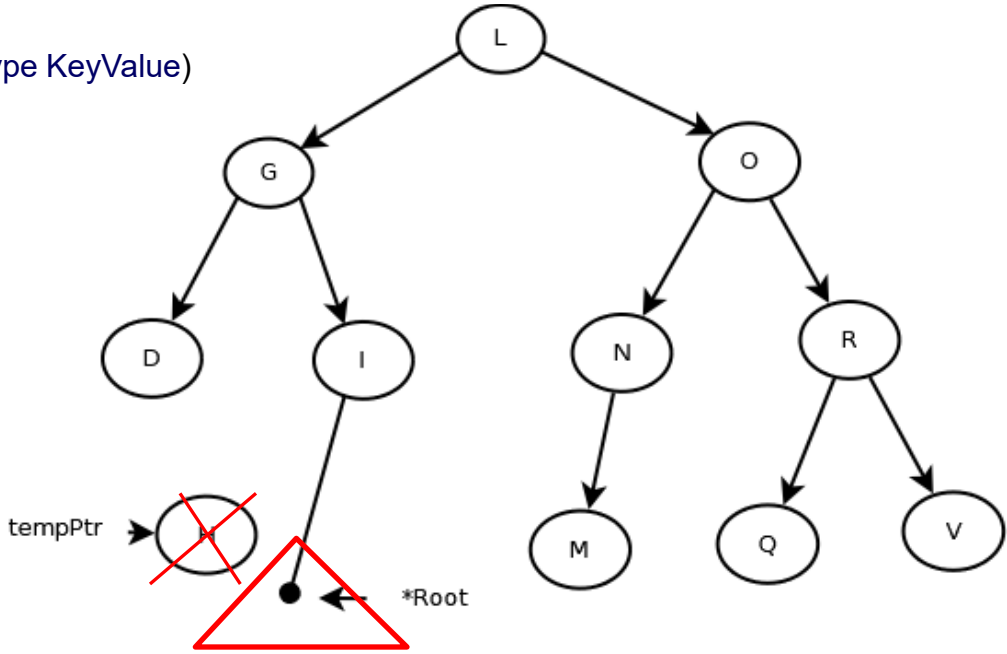


Αναδρομικές κλήσεις της RecBSTDelete

RecBSTDelete("H","H") RecBSTDelete("I"->LChild, "H")
RecBSTDelete("I","H") RecBSTDelete("G"->RChild, "H")
RecBSTDelete("G","H") RecBSTDelete("L"->LChild, "H")
RecBSTDelete("L","H")

Διαγραφή φύλλου, ο "Η"

```
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
 {

  BinTreePointer TempPtr;

   if (BSTEmpty(*Root))
       printf("TO STOIXEIO DEN BRE8HKE STO DDA\n");
   else
        if (KeyValue < (*Root)->Data)
              RecBSTDelete(&((*Root)->LChild), KeyValue);
        else if (KeyValue > (*Root)->Data)
              RecBSTDelete(&((*Root)->RChild), KeyValue);
              else
                   if ((*Root)->LChild == NULL) {
                      TempPtr = *Root;
                      *Root = (*Root)->RChild;
                     free(TempPtr );
                   }
                   else if ((*Root)->RChild == NULL) {
                            TempPtr = *Root;
                             *Root = (*Root)-> LChild;
                           free(TempPtr );
                   }
                   else {
                      TempPtr = (*Root)->RChild;
                     while (TempPtr->LChild != NULL)
                              TempPtr = TempPtr->LChild;
                      (*Root)->Data = TempPtr->Data;
                      RecBSTDelete(&((*Root)->RChild),
                              (*Root)->Data);
                   }
}
```

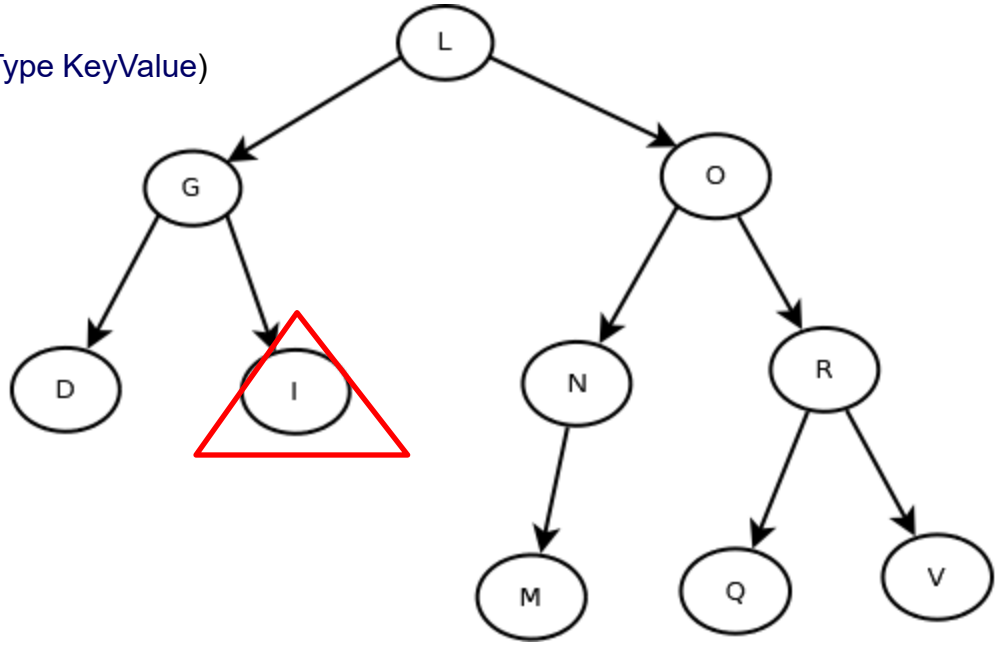


Αναδρομικές κλήσεις της RecBSTDelete

RecBSTDelete(null,"H") RecBSTDelete("I"->LChild, "H")	Return *Root-> NULL
RecBSTDelete("I","H") RecBSTDelete("G"->RChild, "H")	
RecBSTDelete("G","H") RecBSTDelete("L"->LChild, "H")	
RecBSTDelete("L","H")	

Διαγραφή φύλλου, ο "Η"

```
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
 {

  BinTreePointer TempPtr;

   if (BSTEmpty(*Root))
       printf("TO STOIXEIO DEN BRE8HKE STO DDA\n");
   else
        if (KeyValue < (*Root)->Data)
              RecBSTDelete(&((*Root)->LChild), KeyValue);
        else if (KeyValue > (*Root)->Data)
              RecBSTDelete(&((*Root)->RChild), KeyValue);
              else
                   if ((*Root)->LChild == NULL) {
                      TempPtr = *Root;
                      *Root = (*Root)->RChild;
                     free(TempPtr );
                   }
                   else if ((*Root)->RChild == NULL) {
                             TempPtr = *Root;
                              *Root = (*Root)-> LChild;
                            free(TempPtr );
                   }
                   else {
                      TempPtr = (*Root)->RChild;
                     while (TempPtr->LChild != NULL)
                               TempPtr = TempPtr->LChild;
                      (*Root)->Data = TempPtr->Data;
                      RecBSTDelete(&((*Root)->RChild),
                               (*Root)->Data);
                   }
}
```



Αναδρομικές κλήσεις της RecBSTDelete

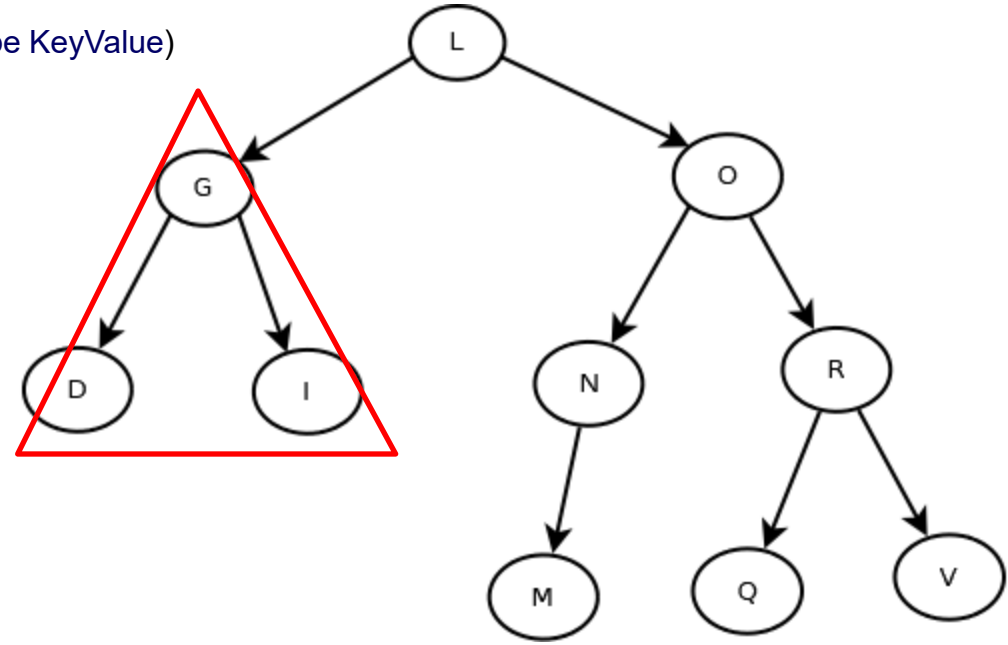
(κλείσιμο των αναδρομικών κλήσεων)

RecBSTDelete("I","H") RecBSTDelete("G"->RChild, "H")
RecBSTDelete("G","H") RecBSTDelete("L"->LChild, "H")
RecBSTDelete("L","H")

Return *Root->"I"

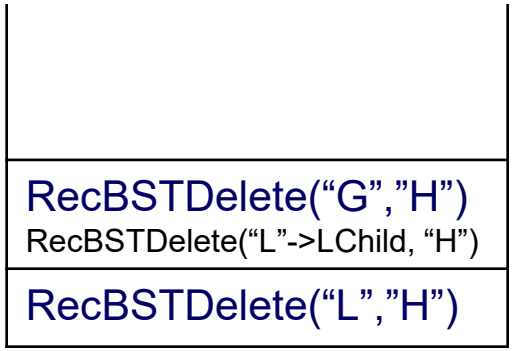
Διαγραφή φύλλου, ο "Η"

```
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
 {

  BinTreePointer TempPtr;

   if (BSTEmpty(*Root))
       printf("TO STOIXEIO DEN BRE8HKE STO DDA\n");
   else
        if (KeyValue < (*Root)->Data)
              RecBSTDelete(&((*Root)->LChild), KeyValue);
        else if (KeyValue > (*Root)->Data)
              RecBSTDelete(&((*Root)->RChild), KeyValue);
              else
                   if ((*Root)->LChild == NULL) {
                      TempPtr = *Root;
                      *Root = (*Root)->RChild;
                     free(TempPtr );
                   }
                  else if ((*Root)->RChild == NULL) {
                           TempPtr = *Root;
                            *Root = (*Root)-> LChild;
                          free(TempPtr );
                  }
                  else {
                      TempPtr = (*Root)->RChild;
                     while (TempPtr->LChild != NULL)
                              TempPtr = TempPtr->LChild;
                      (*Root)->Data = TempPtr->Data;
                      RecBSTDelete(&((*Root)->RChild),
                              (*Root)->Data);
                  }
}
```



Αναδρομικές κλήσεις της RecBSTDelete

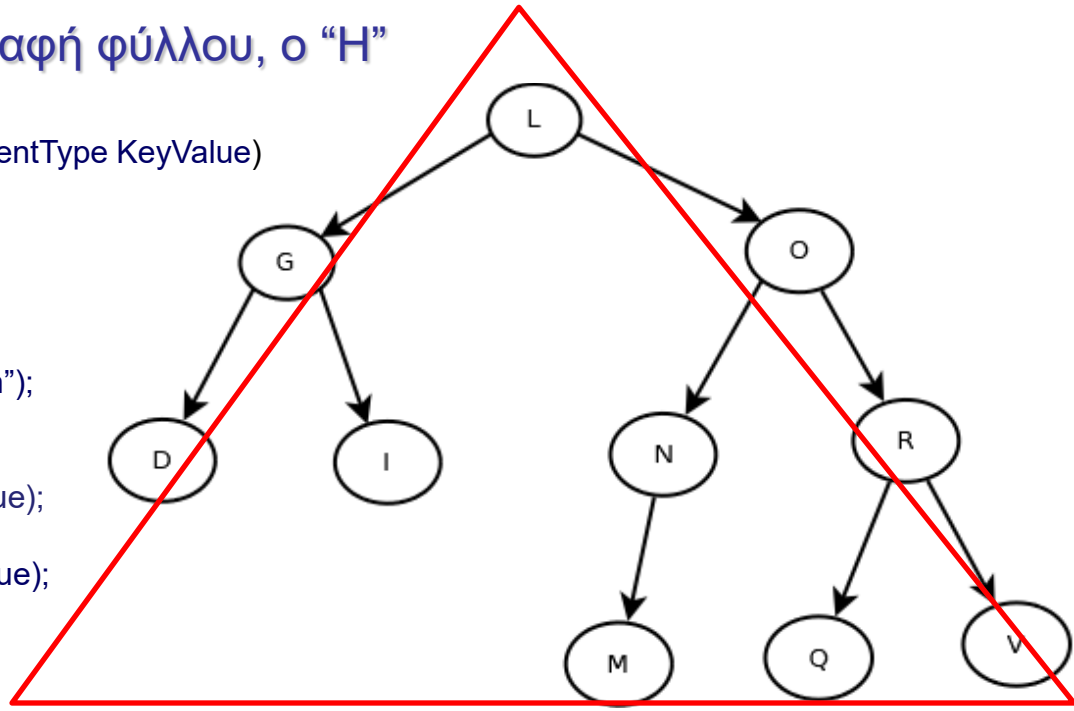
(κλείσιμο των αναδρομικών κλήσεων)



Return *Root->"G"

Διαγραφή φύλλου, ο "Η"

```
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
 {

  BinTreePointer TempPtr;

   if (BSTEmpty(*Root))
        printf("TO STOIXEIO DEN BRE8HKE STO DDA\n");
   else
        if (KeyValue < (*Root)->Data)
              RecBSTDelete(&((*Root)->LChild), KeyValue);
        else if (KeyValue > (*Root)->Data)
              RecBSTDelete(&((*Root)->RChild), KeyValue);
              else
                   if ((*Root)->LChild == NULL) {
                      TempPtr = *Root;
                      *Root = (*Root)->RChild;
                     free(TempPtr );
                   }
                   else if ((*Root)->RChild == NULL) {
                            TempPtr = *Root;
                             *Root = (*Root)-> LChild;
                           free(TempPtr );
                   }
                   else {
                       TempPtr = (*Root)->RChild;
                      while (TempPtr->LChild != NULL)
                               TempPtr = TempPtr->LChild;
                       (*Root)->Data = TempPtr->Data;
                       RecBSTDelete(&((*Root)->RChild),
                               (*Root)->Data);
                   }
}
```



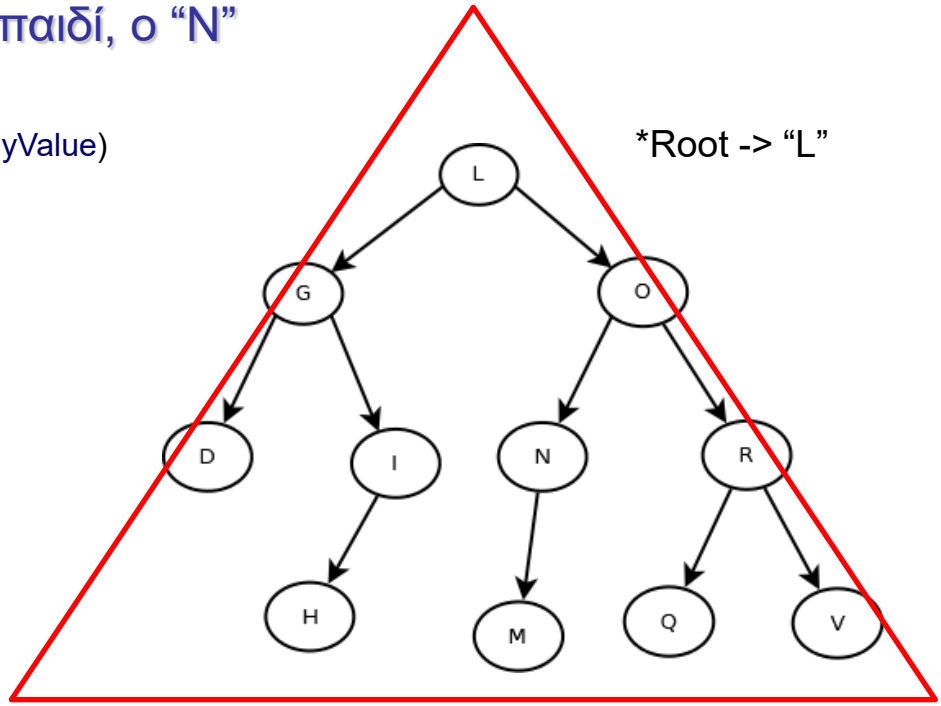
Αναδρομικές κλήσεις της RecBSTDelete

(κλείσιμο των αναδρομικών κλήσεων)

Return *Root->"L"

Διαγραφή κόμβου με 1 παιδί, ο “N”

```
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
 {

  BinTreePointer TempPtr;

   if (BSTEmpty(*Root))
       printf("TO STOIXEIO DEN BRE8HKE STO DDA\n”);
   else
        if (KeyValue < (*Root)->Data)
              RecBSTDelete(&((*Root)->LChild), KeyValue);
        else if (KeyValue > (*Root)->Data)
              RecBSTDelete(&((*Root)->RChild), KeyValue);
              else
                   if ((*Root)->LChild == NULL) {
                      TempPtr = *Root;
                      *Root = (*Root)->RChild;
                     free(TempPtr );
                    }
                   else if ((*Root)->RChild == NULL) {
                            TempPtr = *Root;
                             *Root = (*Root)-> LChild;
                           free(TempPtr );
                   }
                   else {
                      TempPtr = (*Root)->RChild;
                     while (TempPtr->LChild != NULL)
                              TempPtr = TempPtr->LChild;
                      (*Root)->Data = TempPtr->Data;
                      RecBSTDelete(&((*Root)->RChild),
                              (*Root)->Data);
                   }
}
```

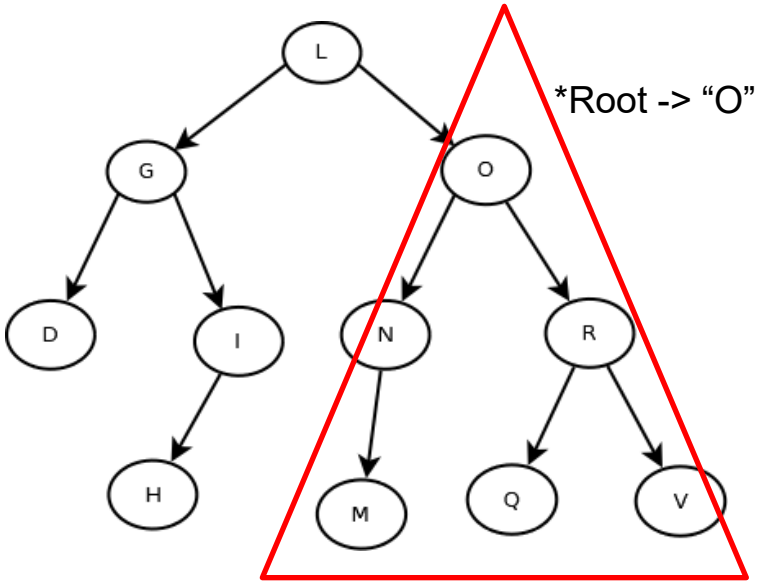


Αναδρομικές κλήσεις της RecBSTDelete

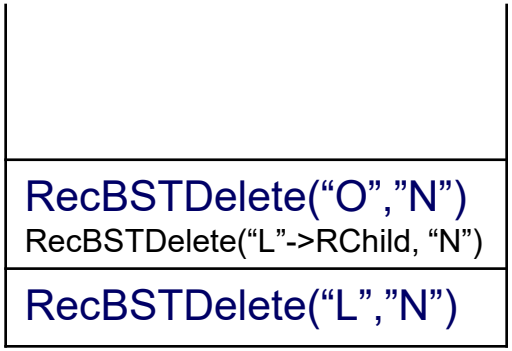
Η 1η παράμετρος δεν είναι το “L” αλλά δείκτης προς τον κόμβο που περιέχει το “L”. Συμβολίζεται έτσι για λόγους καθαρά «πρακτικούς»

Διαγραφή κόμβου με 1 παιδί, ο "N"

```
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
 {

  BinTreePointer TempPtr;

   if (BSTEmpty(*Root))
       printf("TO STOIXEIO DEN BRE8HKE STO DDA\n");
   else
       if (KeyValue < (*Root)->Data)
           RecBSTDelete(&((*Root)->LChild), KeyValue);
       else if (KeyValue > (*Root)->Data)
           RecBSTDelete(&((*Root)->RChild), KeyValue);
           else
             if ((*Root)->LChild == NULL) {
               TempPtr = *Root;
               *Root = (*Root)->RChild;
              free(TempPtr );
             }
             else if ((*Root)->RChild == NULL) {
                 TempPtr = *Root;
                  *Root = (*Root)-> LChild;
                free(TempPtr );
             }
             else {
               TempPtr = (*Root)->RChild;
              while (TempPtr->LChild != NULL)
                     TempPtr = TempPtr->LChild;
               (*Root)->Data = TempPtr->Data;
               RecBSTDelete(&((*Root)->RChild),
                     (*Root)->Data);
             }
}
```

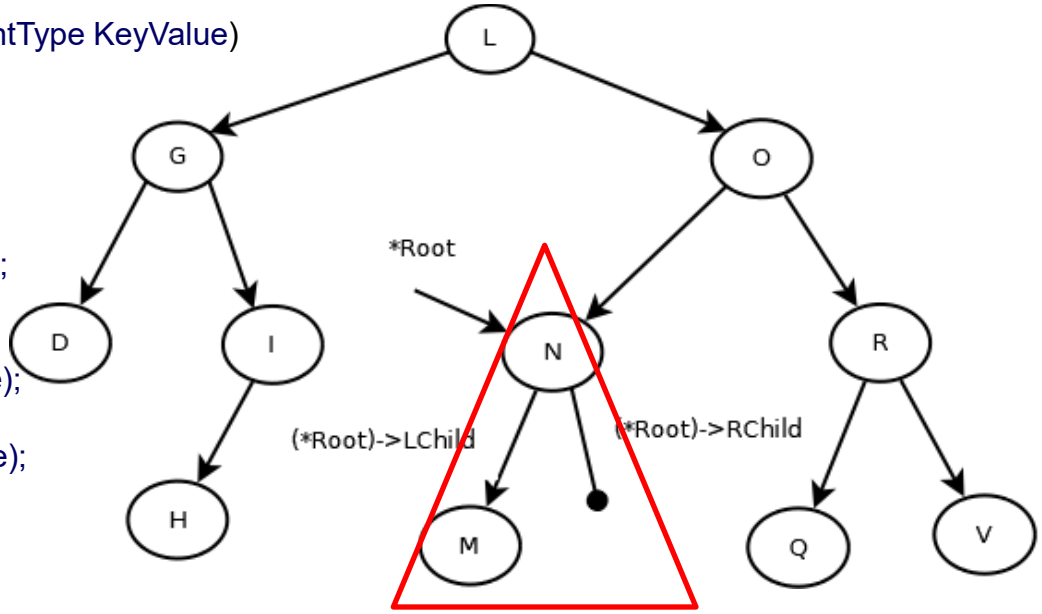


Αναδρομικές κλήσεις της RecBSTDelete



Διαγραφή κόμβου με 1 παιδί, ο “N”

```
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
 {

  BinTreePointer TempPtr;

   if (BSTEmpty(*Root))
       printf” TO STOIXEIO DEN BRE8HKE STO DDA\n”);
   else
        if (KeyValue < (*Root)->Data)
              RecBSTDelete(&((*Root)->LChild), KeyValue);
        else if (KeyValue > (*Root)->Data)
              RecBSTDelete(&((*Root)->RChild), KeyValue);
              else
                   if ((*Root)->LChild == NULL) {
                      TempPtr = *Root;
                      *Root = (*Root)->RChild;
                     free(TempPtr );
                   }
                  else if ((*Root)->RChild == NULL) {
                            TempPtr = *Root;
                             *Root = (*Root)-> LChild;
                           free(TempPtr );
                  }
                  else {
                      TempPtr = (*Root)->RChild;
                     while (TempPtr->LChild != NULL)
                              TempPtr = TempPtr->LChild;
                      (*Root)->Data = TempPtr->Data;
                      RecBSTDelete(&((*Root)->RChild),
                              (*Root)->Data);
                  }
}
```



Αναδρομικές κλήσεις της RecBSTDelete

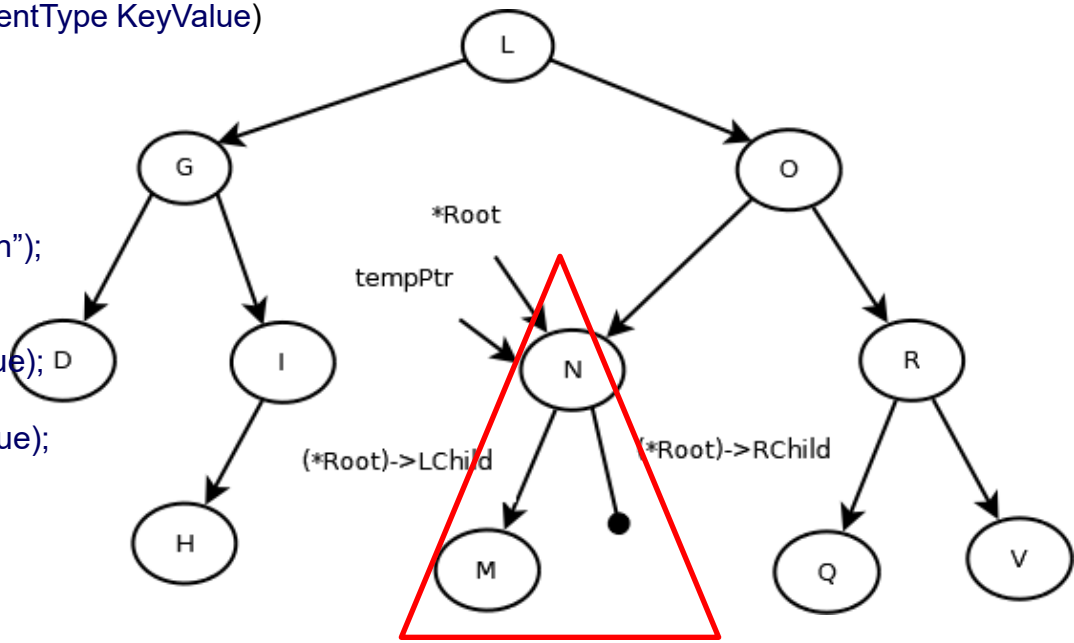
RecBSTDelete(“N”,”N”)
RecBSTDelete(“O”->LChild, “N”)

RecBSTDelete(“O”,”N”)
RecBSTDelete(“L”->RChild, “N”)

RecBSTDelete(“L”,”N”)

Διαγραφή κόμβου με 1 παιδί, ο “N”

```
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
 {

  BinTreePointer TempPtr;

   if (BSTEmpty(*Root))
        printf(“TO STOIXEIO DEN BRE8HKE STO DDA\n”);
   else
        if (KeyValue < (*Root)->Data)
              RecBSTDelete(&((*Root)->LChild), KeyValue);
        else if (KeyValue > (*Root)->Data)
              RecBSTDelete(&((*Root)->RChild), KeyValue);
              else
                   if ((*Root)->LChild == NULL) {
                      TempPtr = *Root;
                      *Root = (*Root)->RChild;
                     free(TempPtr );
                   }
                  else if ((*Root)->RChild == NULL) {
                            TempPtr = *Root;
                             *Root = (*Root)-> LChild;
                           free(TempPtr );
                  }
                  else {
                      TempPtr = (*Root)->RChild;
                     while (TempPtr->LChild != NULL)
                              TempPtr = TempPtr->LChild;
                      (*Root)->Data = TempPtr->Data;
                      RecBSTDelete(&((*Root)->RChild),
                              (*Root)->Data);
                  }
}
```



Αναδρομικές κλήσεις της RecBSTDelete

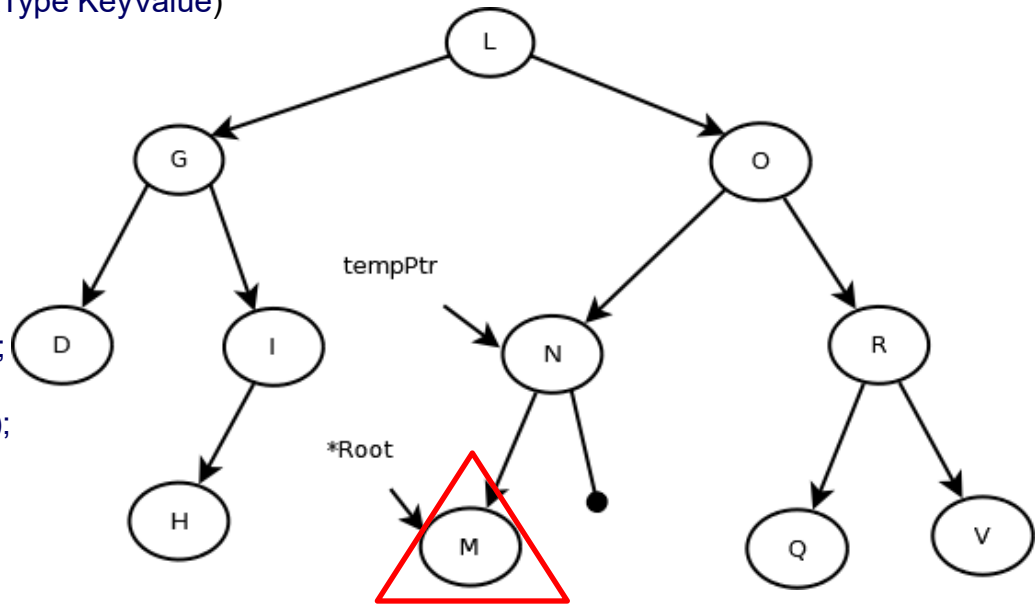
RecBSTDelete(“N”,”N”)
RecBSTDelete(“O”->LChild, “N”)

RecBSTDelete(“O”,”N”)
RecBSTDelete(“L”->RChild, “N”)

RecBSTDelete(“L”,”N”)

Διαγραφή κόμβου με 1 παιδί, ο "N"

```
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
 {

  BinTreePointer TempPtr;

   if (BSTEmpty(*Root))
       printf("TO STOIXEIO DEN BRE8HKE STO DDA\n");
   else
        if (KeyValue < (*Root)->Data)
              RecBSTDelete(&((*Root)->LChild), KeyValue);
        else if (KeyValue > (*Root)->Data)
              RecBSTDelete(&((*Root)->RChild), KeyValue);
              else
                   if ((*Root)->LChild == NULL) {
                      TempPtr = *Root;
                      *Root = (*Root)->RChild;
                     free(TempPtr );
                   }
                  else if ((*Root)->RChild == NULL) {
                            TempPtr = *Root;
                             *Root = (*Root)-> LChild;
                           free(TempPtr );
                  }
                  else {
                      TempPtr = (*Root)->RChild;
                     while (TempPtr->LChild != NULL)
                              TempPtr = TempPtr->LChild;
                      (*Root)->Data = TempPtr->Data;
                      RecBSTDelete(&((*Root)->RChild),
                              (*Root)->Data);
                  }
}
```



Αναδρομικές κλήσεις της RecBSTDelete

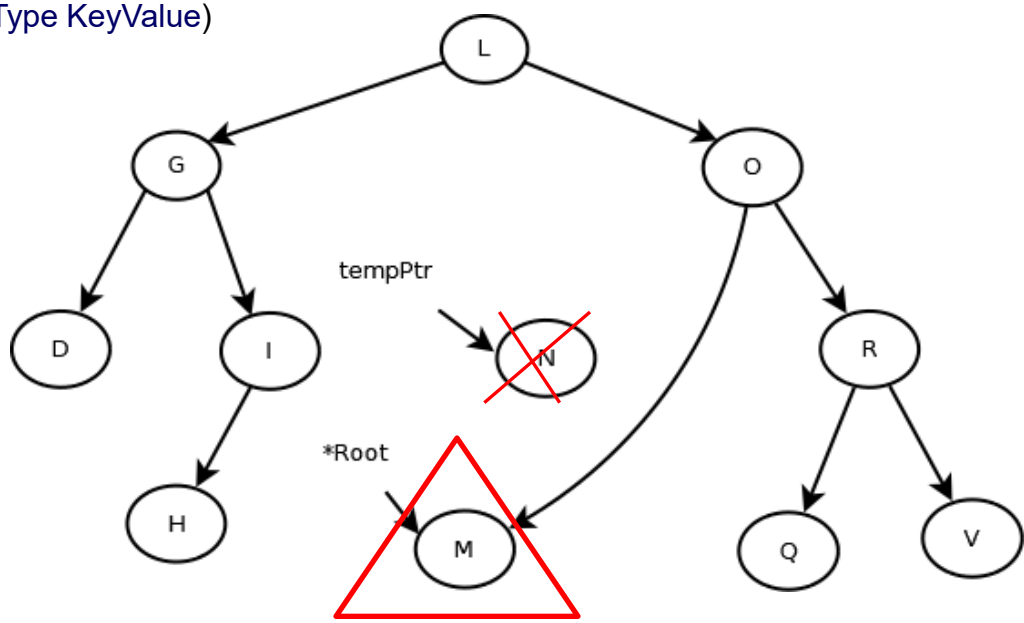
RecBSTDelete("N","N")
RecBSTDelete("O"->LChild, "N")

RecBSTDelete("O","N")
RecBSTDelete("L"->RChild, "N")

RecBSTDelete("L","N")

Διαγραφή κόμβου με 1 παιδί, ο "Ν"

```
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
 {

  BinTreePointer TempPtr;

   if (BSTEmpty(*Root))
       printf("TO STOIXEIO DEN BRE8HKE STO DDA\n");
   else
       if (KeyValue < (*Root)->Data)
           RecBSTDelete(&((*Root)->LChild), KeyValue);
       else if (KeyValue > (*Root)->Data)
           RecBSTDelete(&((*Root)->RChild), KeyValue);
           else
               if ((*Root)->LChild == NULL) {
                   TempPtr = *Root;
                   *Root = (*Root)->RChild;
                   free(TempPtr );
               }
               else if ((*Root)->RChild == NULL) {
                       TempPtr = *Root;
                       *Root = (*Root)-> LChild;
                       free(TempPtr );
               }
               else {
                   TempPtr = (*Root)->RChild;
                   while (TempPtr->LChild != NULL)
                           TempPtr = TempPtr->LChild;
                   (*Root)->Data = TempPtr->Data;
                   RecBSTDelete(&((*Root)->RChild),
                           (*Root)->Data);
               }
}
```



Αναδρομικές κλήσεις της RecBSTDelete

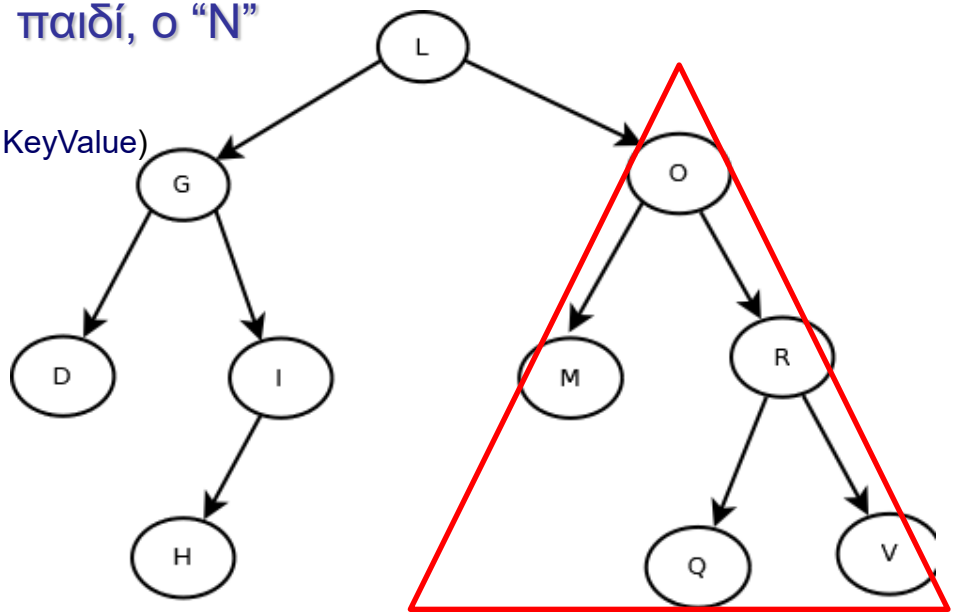
RecBSTDelete("M","N")
RecBSTDelete("O"->LChild, "N")

RecBSTDelete("O","N")
RecBSTDelete("L"->RChild, "N")

RecBSTDelete("L","N")

Return *Root->"M"

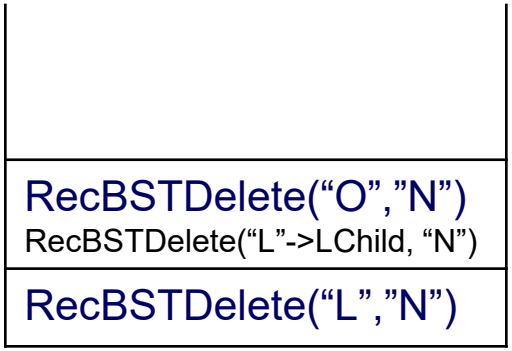
Διαγραφή κόμβου με 1 παιδί, ο "N"




```
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
 {

  BinTreePointer TempPtr;

   if (BSTEmpty(*Root))
       printf("TO STOIXEIO DEN BRE8HKE STO DDA\n");
   else
        if (KeyValue < (*Root)->Data)
              RecBSTDelete(&((*Root)->LChild), KeyValue);
        else if (KeyValue > (*Root)->Data)
              RecBSTDelete(&((*Root)->RChild), KeyValue);
              else
                   if ((*Root)->LChild == NULL) {
                      TempPtr = *Root;
                      *Root = (*Root)->RChild;
                     free(TempPtr );
                   }
                   else if ((*Root)->RChild == NULL) {
                             TempPtr = *Root;
                              *Root = (*Root)-> LChild;
                             free(TempPtr );
                   }
                   else {
                      TempPtr = (*Root)->RChild;
                     while (TempPtr->LChild != NULL)
                               TempPtr = TempPtr->LChild;
                      (*Root)->Data = TempPtr->Data;
                      RecBSTDelete(&((*Root)->RChild),
                               (*Root)->Data);
                   }
}
```

Αναδρομικές κλήσεις της RecBSTDelete

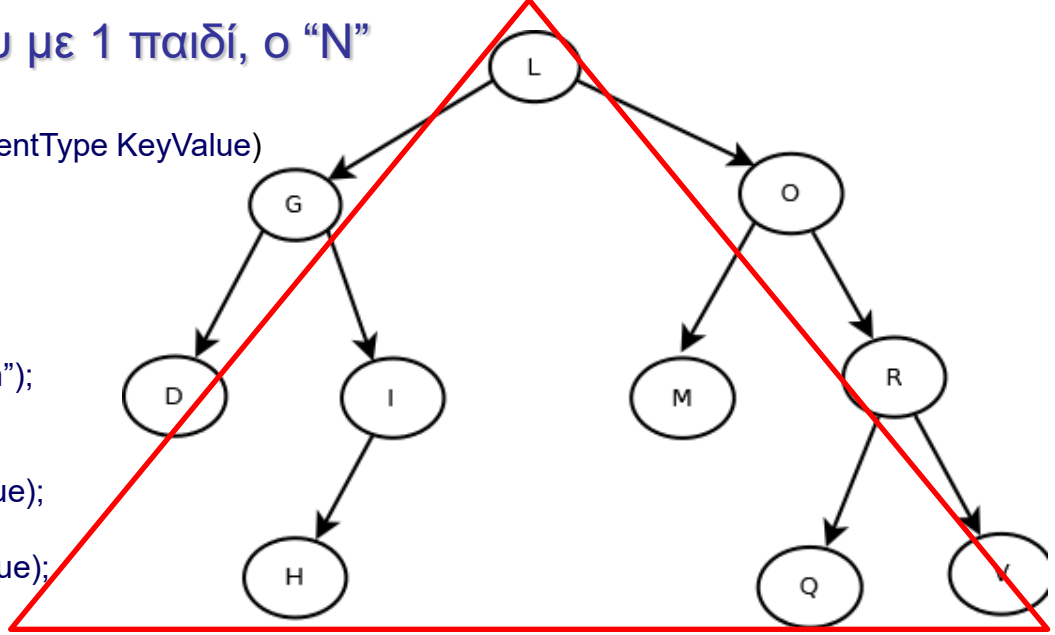
(κλείσιμο των αναδρομικών κλήσεων)



Return *Root->"O"

Διαγραφή κόμβου με 1 παιδί, ο "Ν"

```
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
 {

  BinTreePointer TempPtr;

   if (BSTEmpty(*Root))
       printf("TO STOIXEIO DEN BRE8HKE STO DDA\n");
   else
       if (KeyValue < (*Root)->Data)
           RecBSTDelete(&((*Root)->LChild), KeyValue);
       else if (KeyValue > (*Root)->Data)
           RecBSTDelete(&((*Root)->RChild), KeyValue);
           else
               if ((*Root)->LChild == NULL) {
                   TempPtr = *Root;
                   *Root = (*Root)->RChild;
                   free(TempPtr );
               }
               else if ((*Root)->RChild == NULL) {
                       TempPtr = *Root;
                       *Root = (*Root)-> LChild;
                       free(TempPtr );
               }
               else {
                   TempPtr = (*Root)->RChild;
                   while (TempPtr->LChild != NULL)
                           TempPtr = TempPtr->LChild;
                   (*Root)->Data = TempPtr->Data;
                   RecBSTDelete(&((*Root)->RChild),
                           (*Root)->Data);
               }
}
```



Αναδρομικές κλήσεις της RecBSTDelete

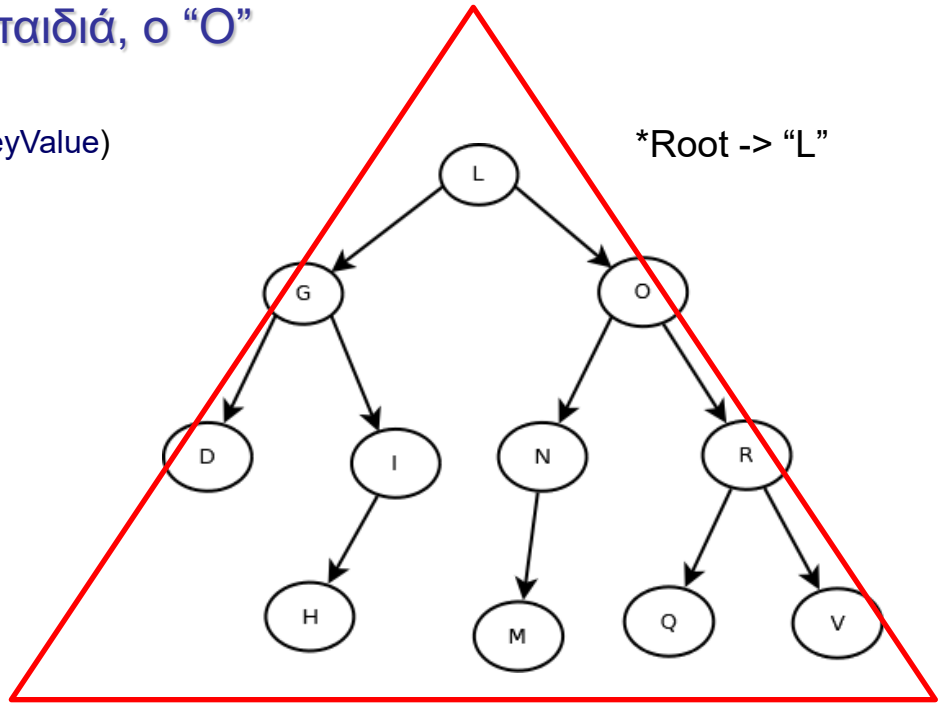
(κλείσιμο των αναδρομικών κλήσεων)

RecBSTDelete("L","N")

Return *Root->"L"

Διαγραφή κόμβου με 2 παιδιά, ο "Ο"

```
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
 {

  BinTreePointer TempPtr;

   if (BSTEmpty(*Root))
       printf("TO STOIXEIO DEN BRE8HKE STO DDA\n");
  else
       if (KeyValue < (*Root)->Data)
            RecBSTDelete(&((*Root)->LChild), KeyValue);
       else if (KeyValue > (*Root)->Data)
            RecBSTDelete(&((*Root)->RChild), KeyValue);
            else
                 if ((*Root)->LChild == NULL) {
                    TempPtr = *Root;
                    *Root = (*Root)->RChild;
                   free(TempPtr );
                 }
                 else if ((*Root)->RChild == NULL) {
                           TempPtr = *Root;
                            *Root = (*Root)-> LChild;
                          free(TempPtr );
                 }
                 else {
                    TempPtr = (*Root)->RChild;
                   while (TempPtr->LChild != NULL)
                            TempPtr = TempPtr->LChild;
                    (*Root)->Data = TempPtr->Data;
                    RecBSTDelete(&((*Root)->RChild),
                            (*Root)->Data);
                 }
}
```



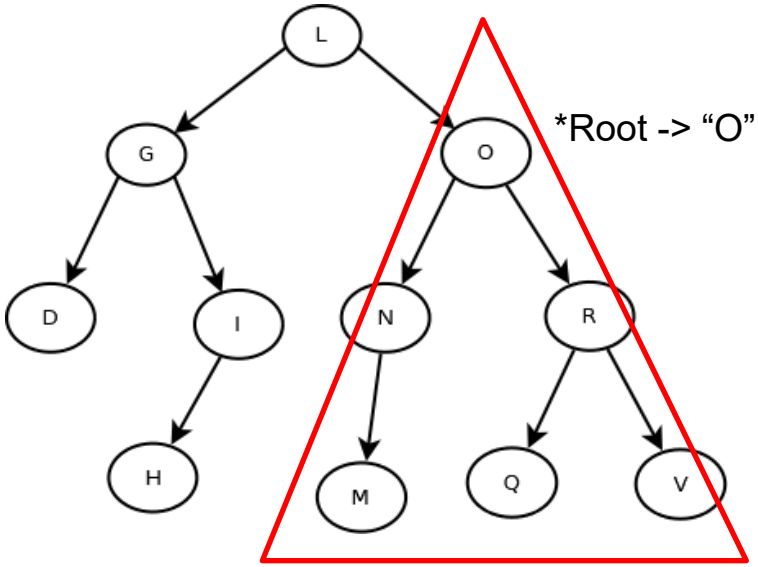
*Root -> "L"

Αναδρομικές κλήσεις της RecBSTDelete

Η 1η παράμετρος δεν είναι το "L" αλλά δείκτης προς τον κόμβο που περιέχει το "L". Συμβολίζεται έτσι για λόγους καθαρά «πρακτικούς»

Διαγραφή κόμβου με 2 παιδιά, ο "Ο"

```
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
 {

  BinTreePointer TempPtr;

   if (BSTEmpty(*Root))
       printf("TO STOIXEIO DEN BRE8HKE STO DDA\n");
   else
       if (KeyValue < (*Root)->Data)
           RecBSTDelete(&((*Root)->LChild), KeyValue);
       else if (KeyValue > (*Root)->Data)
           RecBSTDelete(&((*Root)->RChild), KeyValue);
           else
               if ((*Root)->LChild == NULL) {
                  TempPtr = *Root;
                  *Root = (*Root)->RChild;
                  free(TempPtr );
               }
               else if ((*Root)->RChild == NULL) {
                       TempPtr = *Root;
                       *Root = (*Root)-> LChild;
                      free(TempPtr );
               }
               else {
                  TempPtr = (*Root)->RChild;
                  while (TempPtr->LChild != NULL)
                           TempPtr = TempPtr->LChild;
                  (*Root)->Data = TempPtr->Data;
                  RecBSTDelete(&((*Root)->RChild),
                           (*Root)->Data);
               }
}
```



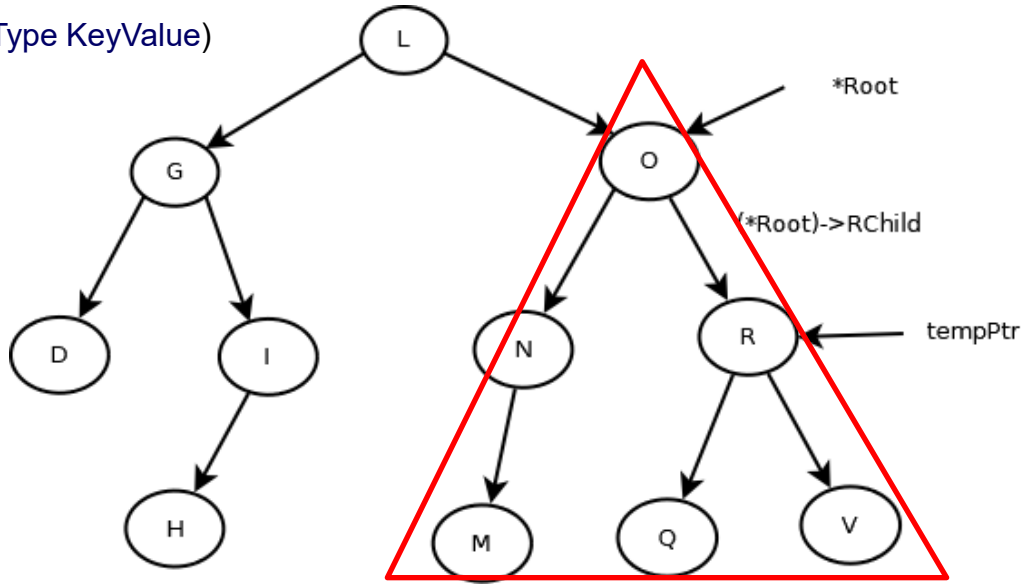
Αναδρομικές κλήσεις της RecBSTDelete

RecBSTDelete("O","O")
RecBSTDelete("L"->RChild, "O")

RecBSTDelete("L","O")

Διαγραφή κόμβου με 2 παιδιά, ο "Ο"

```
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
 {

  BinTreePointer TempPtr;

   if (BSTEmpty(*Root))
       printf("TO STOIXEIO DEN BRE8HKE STO DDA\n");
   else
        if (KeyValue < (*Root)->Data)
              RecBSTDelete(&((*Root)->LChild), KeyValue);
        else if (KeyValue > (*Root)->Data)
              RecBSTDelete(&((*Root)->RChild), KeyValue);
              else
                   if ((*Root)->LChild == NULL) {
                      TempPtr = *Root;
                      *Root = (*Root)->RChild;
                     free(TempPtr );
                   }
                  else if ((*Root)->RChild == NULL) {
                            TempPtr = *Root;
                             *Root = (*Root)-> LChild;
                           free(TempPtr );
                  }
                  else {
                      TempPtr = (*Root)->RChild;
                     while (TempPtr->LChild != NULL)
                              TempPtr = TempPtr->LChild;
                      (*Root)->Data = TempPtr->Data;
                      RecBSTDelete(&((*Root)->RChild),
                              (*Root)->Data);
                  }
}
```



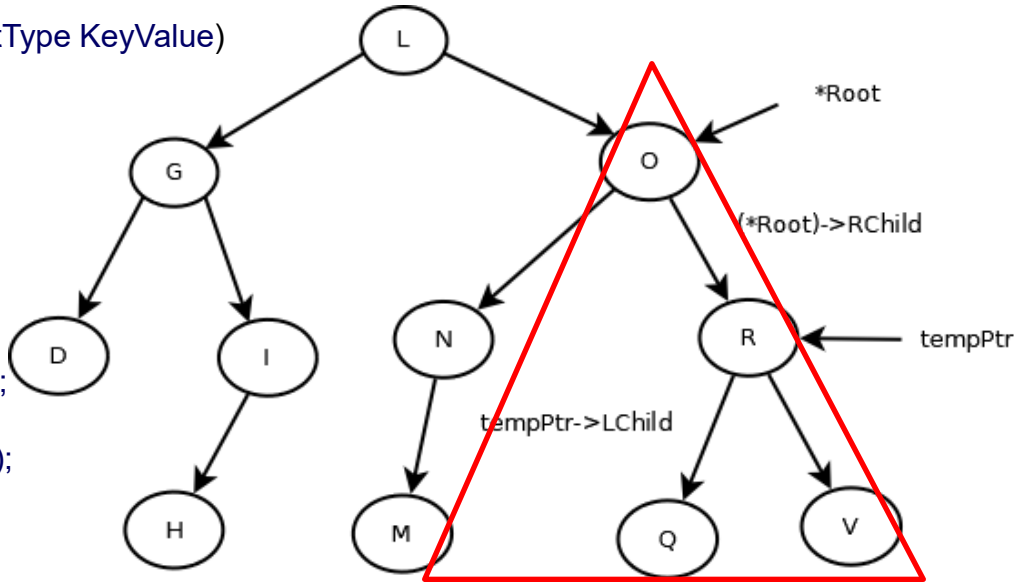
Αναδρομικές κλήσεις της RecBSTDelete

RecBSTDelete("O","O")
RecBSTDelete("L"->RChild, "O")

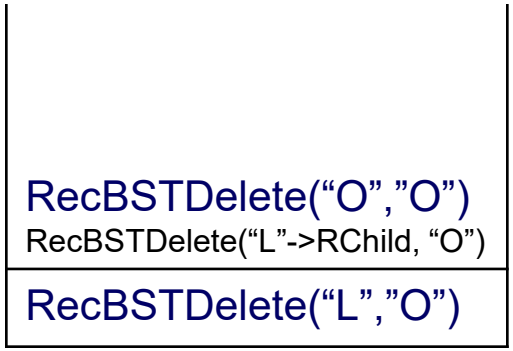
RecBSTDelete("L","O")

Διαγραφή κόμβου με 2 παιδιά, ο "Ο"

```
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
 {

  BinTreePointer TempPtr;

   if (BSTEmpty(*Root))
      printf("TO STOIXEIO DEN BRE8HKE STO DDA\n");
   else
        if (KeyValue < (*Root)->Data)
              RecBSTDelete(&((*Root)->LChild), KeyValue);
        else if (KeyValue > (*Root)->Data)
              RecBSTDelete(&((*Root)->RChild), KeyValue);
              else
                   if ((*Root)->LChild == NULL) {
                      TempPtr = *Root;
                      *Root = (*Root)->RChild;
                     free(TempPtr );
                   }
                   else if ((*Root)->RChild == NULL) {
                            TempPtr = *Root;
                             *Root = (*Root)-> LChild;
                           free(TempPtr );
                   }
                   else {
                       TempPtr = (*Root)->RChild;
                      while (TempPtr->LChild != NULL)
                               TempPtr = TempPtr->LChild;
                       (*Root)->Data = TempPtr->Data;
                       RecBSTDelete(&((*Root)->RChild),
                               (*Root)->Data);
                   }
}
```

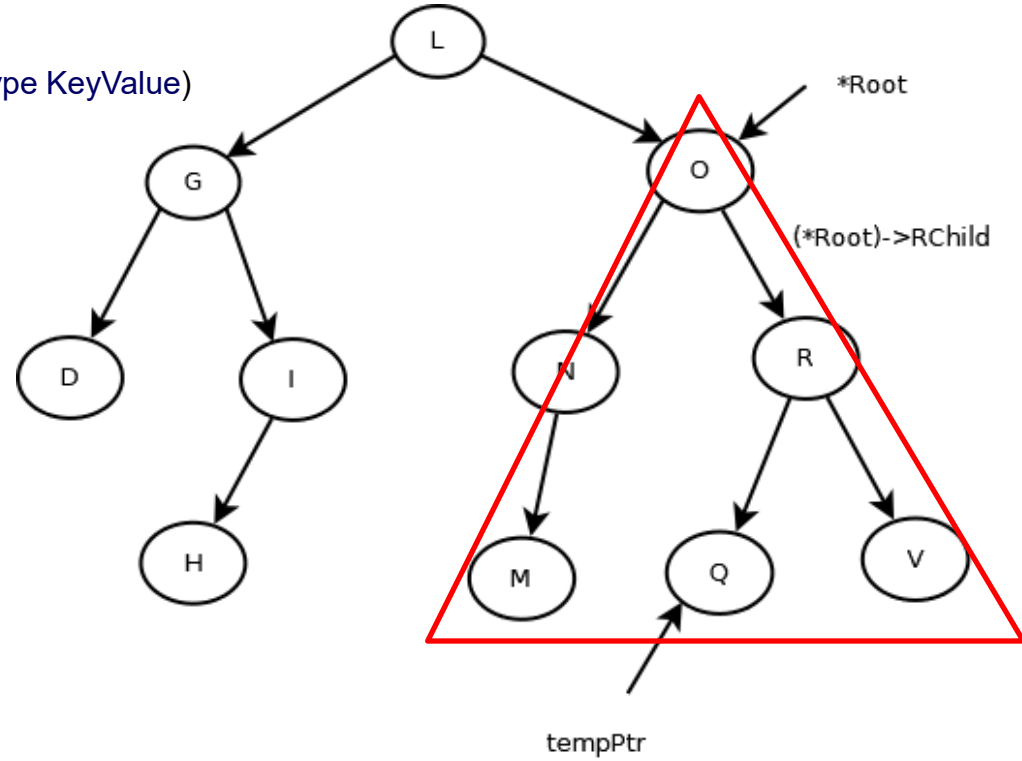


Αναδρομικές κλήσεις της RecBSTDelete



Διαγραφή κόμβου με 2 παιδιά, ο "Ο"

```
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
 {

  BinTreePointer TempPtr;

   if (BSTEmpty(*Root))
       printf("TO STOIXEIO DEN BRE8HKE STO DDA\n");
   else
        if (KeyValue < (*Root)->Data)
              RecBSTDelete(&((*Root)->LChild), KeyValue);
        else if (KeyValue > (*Root)->Data)
              RecBSTDelete(&((*Root)->RChild), KeyValue);
              else
                   if ((*Root)->LChild == NULL) {
                      TempPtr = *Root;
                      *Root = (*Root)->RChild;
                     free(TempPtr );
                    }
                   else if ((*Root)->RChild == NULL) {
                             TempPtr = *Root;
                              *Root = (*Root)-> LChild;
                            free(TempPtr );
                   }
                   else {
                       TempPtr = (*Root)->RChild;
                      while (TempPtr->LChild != NULL)
                               TempPtr = TempPtr->LChild;
                       (*Root)->Data = TempPtr->Data;
                       RecBSTDelete(&((*Root)->RChild),
                               (*Root)->Data);
                   }
}
```



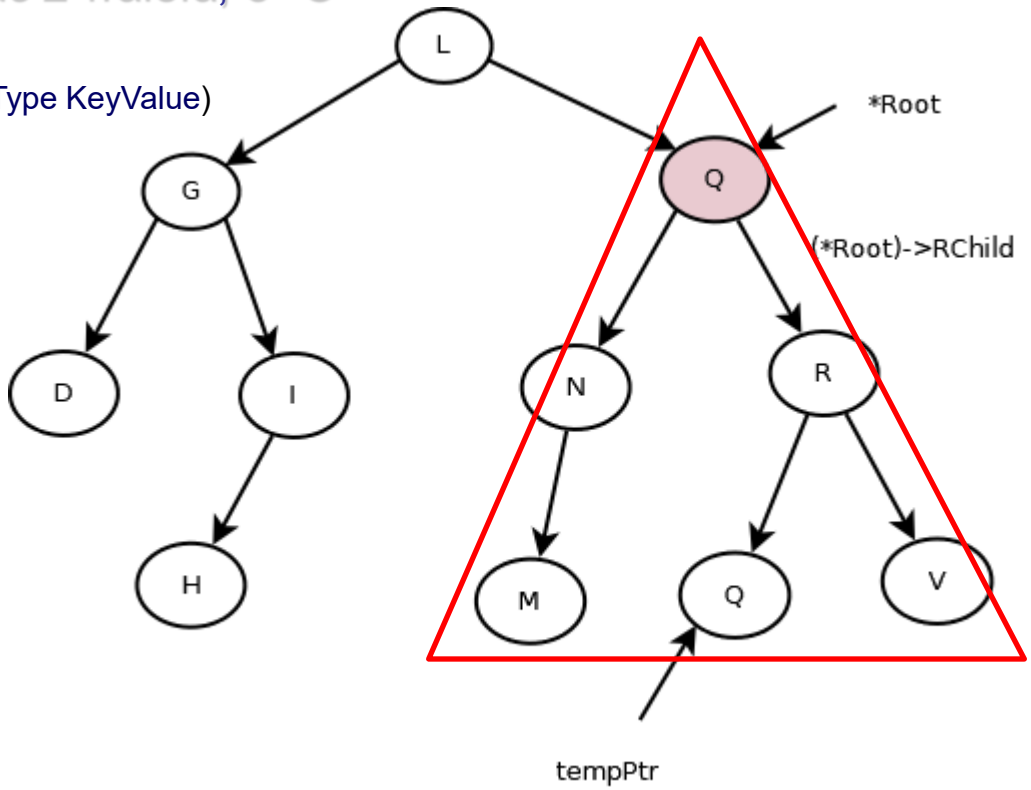
Αναδρομικές κλήσεις της RecBSTDelete

RecBSTDelete("O","O")
RecBSTDelete("L"->RChild, "O")

RecBSTDelete("L","O")

Διαγραφή κόμβου με 2 παιδιά, ο "Ο"

```
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
 {

  BinTreePointer TempPtr;

    if (BSTEmpty(*Root))
        printf("TO STOIXEIO DEN BRE8HKE STO DDA\n");
    else
        if (KeyValue < (*Root)->Data)
            RecBSTDelete(&((*Root)->LChild), KeyValue);
        else if (KeyValue > (*Root)->Data)
            RecBSTDelete(&((*Root)->RChild), KeyValue);
            else
                if ((*Root)->LChild == NULL) {
                    TempPtr = *Root;
                    *Root = (*Root)->RChild;
                   free(TempPtr );
                }
                else if ((*Root)->RChild == NULL) {
                        TempPtr = *Root;
                         *Root = (*Root)-> LChild;
                       free(TempPtr );
                }
                else {
                    TempPtr = (*Root)->RChild;
                   while (TempPtr->LChild != NULL)
                            TempPtr = TempPtr->LChild;
                    (*Root)->Data = TempPtr->Data;
                    RecBSTDelete(&((*Root)->RChild),
                            (*Root)->Data);
                }
}
```



Αναδρομικές κλήσεις της RecBSTDelete

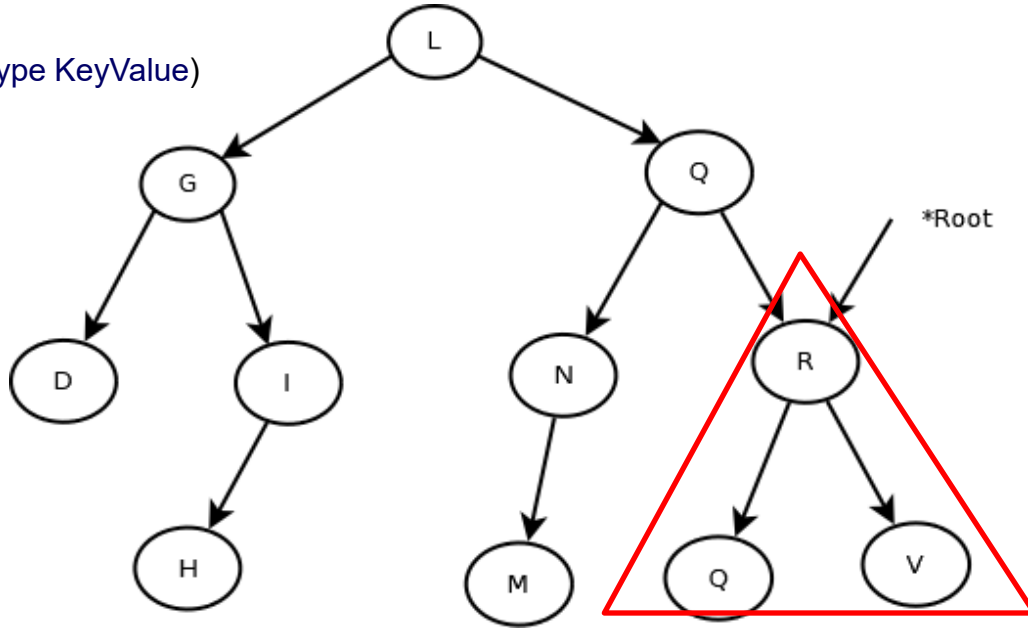
RecBSTDelete("Ο","Ο")
RecBSTDelete("L"->RChild, "Ο")

RecBSTDelete("L","Ο")

Διαγραφή κόμβου φύλλου, ο "Q"

```
void RecBSTDelete(BinTreePointer *Root, BinTreeElementType KeyValue)
 {

  BinTreePointer TempPtr;

   if (BSTEmpty(*Root))
       printf("TO STOIXEIO DEN BRE8HKE STO DDA\n");
   else
        if (KeyValue < (*Root)->Data)
              RecBSTDelete(&((*Root)->LChild), KeyValue);
        else if (KeyValue > (*Root)->Data)
              RecBSTDelete(&((*Root)->RChild), KeyValue);
              else
                   if ((*Root)->LChild == NULL) {
                      TempPtr = *Root;
                      *Root = (*Root)->RChild;
                     free(TempPtr );
                   }
                   else if ((*Root)->RChild == NULL) {
                            TempPtr = *Root;
                             *Root = (*Root)-> LChild;
                           free(TempPtr );
                   }
                   else {
                       TempPtr = (*Root)->RChild;
                      while (TempPtr->LChild != NULL)
                               TempPtr = TempPtr->LChild;
                       (*Root)->Data = TempPtr->Data;
                       RecBSTDelete(&((*Root)->RChild),
                               (*Root)->Data);
                   }
 }
```

Περίπτωση διαγραφή κόμβου φύλλου



Θα συνεχίσει τη διαγραφή του Q στο υποδένδρο με ρίζα το R, ως περίπτωση διαγραφή κόμβου φύλλου

Αναδρομικές κλήσεις της RecBSTDelete

RecBSTDelete("R","Q") RecBSTDelete("Q"->RChild, "Q")
RecBSTDelete("O","O") RecBSTDelete("L"->RChild, "O")
RecBSTDelete("L","O")

Αναδρομική υλοποίηση διαδικασιών ΔΔΑ

TestBSTRec.c

BstRecADT.c

BstRecADT.h