Δομές Δεδομένων σε C

Μάθημα 6:

Δυαδικά Δένδρα

Δημήτρης Ψούνης

Περιεχόμενα Μαθήματος

Α. Θεωρία

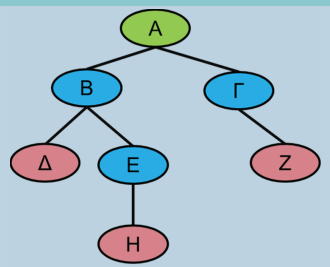
1. Δένδρο

1. Ορισμός Δένδρου

Το **«Δένδρο»** είναι μια μη γραμμική, ιεραρχική δομή δεδομένων στην οποία:

- Αποτελείται από κόμβους (κορυφές) και ακμές (πλευρές) που ομοιάζουν με ένα (ανάποδο) δένδρο. Οι κόμβοι είναι τριών ειδών:
  - Η **ρίζα** είναι ο μοναδικός κόμβος από τον οποίο μόνο ξεκινούν ακμές
  - Οι **εσωτερικοί κόμβοι** (ή μη τερματικοί) στις οποίες καταλήγουν και ξεκινούν ακμές.
  - Τα **φύλλα** (ή τερματικοί κόμβοι) στα οποία καταλήγει μόνο μία ακμή.

Παράδειγμα:
- Ρίζα: Α
- Εσωτερικοί Κόμβοι: Β,Γ,Ε
- Φύλλα: Δ,Η,Ζ



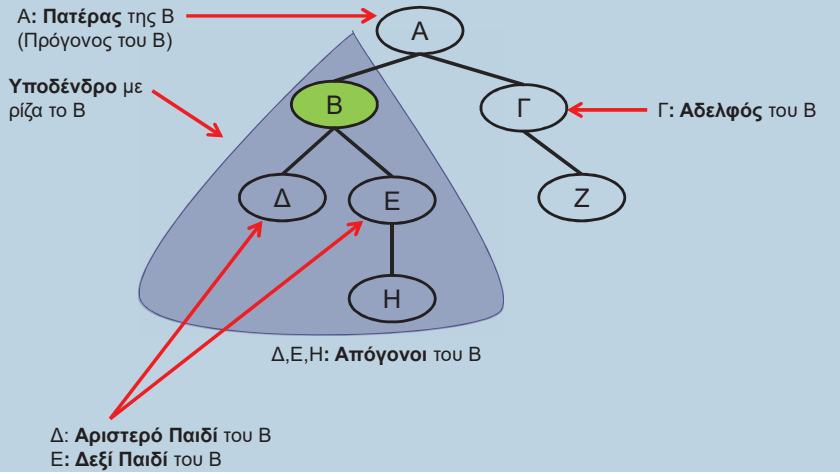
Παρατηρήσεις:
- Το δένδρο χωρίς κόμβους λέγεται «κενό δένδρο» (null tree).
- Ένας εσωτερικός κόμβος έχει τουλάχιστον ένα παιδί .
- Τα φύλλα δεν έχουν παιδιά.

Α. Θεωρία

1. Δένδρο

2. Οικογενειακές Σχέσεις στα Δένδρα

Σε ένα δένδρο ορίζονται **«οικογενειακές σχέσεις»** των κόμβων. Στο παράδειγμα μελετάμε την οικογένεια του κόμβου Β:

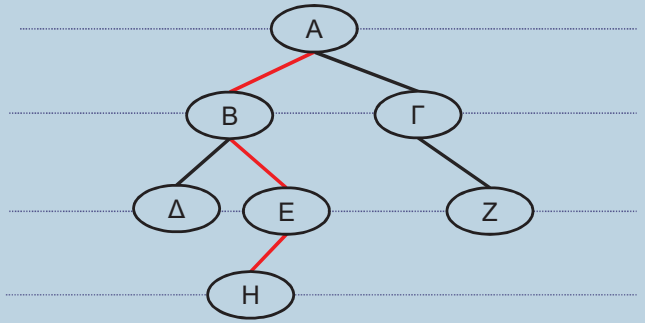


Α. Θεωρία

1. Δένδρο

3. Ύψος – Επίπεδα Δένδρου

- **Διαδρομή**: Ακολουθία κόμβων από μια αφετηρία προς κάποιο απόγονο
- **Μήκος Διαδρομής**: Πλήθος ακμών της διαδρομής
- **Επίπεδο Κόμβου**: Μήκος Διαδρομής από τη ρίζα
- **Ύψος Δένδρου**: Μέγιστο επίπεδο Κόμβου
- **Βαθμός Κόμβου**: Πλήθος Παιδιών Κόμβου



Παράδειγμα Διαδρομής: A,B,E,H
Μήκος Διαδρομής: 3

Ύψος Δένδρου = 3

Α. Θεωρία

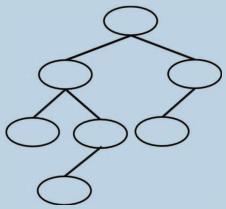
2. Δυαδικό Δένδρο

1. Ορισμός Δυαδικού Δένδρου

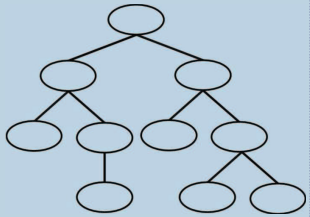
Επιπλέον ορισμοί :

- **Δυαδικό Δένδρο**: Δένδρο που κάθε κορυφή έχει το πολύ δύο παιδιά.
- **Πλήρες Δυαδικό Δένδρο:** Δυαδικό Δένδρο στο οποίο:
  - Όλα τα επίπεδα εκτός ίσως από το τελευταίο έχουν όλους τους δυνατούς κόμβους.
  - Στο τελευταίο επίπεδο οι κόμβοι είναι «όσο το δυνατόν αριστερά»

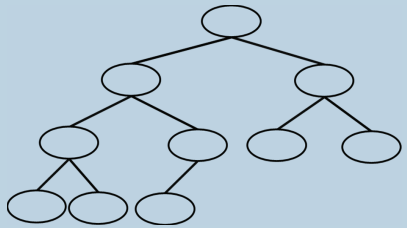
Δυαδικό Δένδρο
Όχι Πλήρες



Δυαδικό Δένδρο
Όχι Πλήρες



Πλήρες Δυαδικό Δένδρο



Α. Θεωρία

2. Δυαδικό Δένδρο

2. Ιδιότητες των Δυαδικών Δένδρων

Ιδιότητα 1:

- Ένα δυαδικό δένδρο ύψους H έχει το πολύ $n=2^{H+1}-1$ κόμβους

Απόδειξη:
Καταμετράμε τις κορυφές σε ένα δυαδικό δένδρο:

- Στο επίπεδο 0 έχουμε το πολύ 2^0 κορυφές
- Στο επίπεδο 1 έχουμε το πολύ 2^1 κορυφές
- …
- Στο επίπεδο H έχουμε το πολύ 2^H κορυφές (φύλλα)

Συνεπώς συνολικά οι κορυφές είναι το πολύ:

$$2^0 + 2^1 + \cdots + 2^H = \frac{2^{H+1}-1}{2-1} = 2^{H+1} - 1$$

Ιδιότητα 2:

- Σε ένα πλήρες δυαδικό δένδρο n κόμβων, ισχύει για το ύψος H ότι $\log(n+1)-1 \le H \le \log n$

Απόδειξη:
Ομοίως με την προηγούμενη απόδειξη, μόνο που στο τελευταίο επίπεδο έχουμε:

- Το πολύ 2^H κόμβους άρα συνολικά $n \le 2^{H+1}-1 \Rightarrow n+1 \le 2^{H+1} \Rightarrow \log(n+1) \le \log(2^{H+1}) \Rightarrow \log(n+1) \le H+1 \Rightarrow \log(n+1)-1 \le H$
- Τουλάχιστον ένα κόμβο: $n \ge 2^H - 1 + 1 \Rightarrow n \ge 2^H \Rightarrow \log(n) \ge \log(2^H) \Rightarrow \log(n) \ge H \Rightarrow H \le \log n$

Α. Θεωρία

2. Δυαδικό Δένδρο

3. Βασικές Πράξεις

Οι **βασικές πράξεις** σε ένα δένδρο είναι:

- **Αρχικοποίηση** του δένδρου (**init**)
- **Εισαγωγή** ενός στοιχείου στο δένδρο (**insert**)
- **Διαγραφή** ενός στοιχείου από το δένδρο (**delete**)
- **Έλεγχος** αν το δένδρο είναι άδειο (**empty**)
- **Περιεχόμενο** ενός κόμβου του δένδρου (**data**)
- **Διαπέραση** του δένδρου (**traverse**)

Υπάρχουν δύο υλοποιήσεις:

- Με δείκτες
  - Την ονομάζουμε «δυναμική αναπαράσταση»
- Με πίνακα
  - Την ονομάζουμε «συνεχόμενη αναπαράσταση»

Σημαντική Παρατήρηση: Υπάρχουν αρκετές δευτερεύουσες πράξεις ή ακόμη και πράξεις που διαμορφώνουμε κατά περίπτωση, μιας και το δένδρο είναι μια πολύπλοκη και εξαιρετικά χρήσιμη δομή δεδομένων. Οι δευτερεύουσες πράξεις καθορίζονται συνήθως από ειδικές περιπτώσεις δένδρων (τα οποία θα δούμε σε επόμενα μαθήματα) όπως τα Δυαδικά Δένδρα Αναζήτησης, τα Δένδρα-Σωροί, τα AVL δένδρα, ή τα B^+-δένδρα.

Α. Θεωρία

2. Δυαδικό Δένδρο

4. Συνεχόμενη Αναπαράσταση

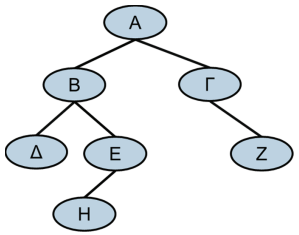
Στην **Συνεχόμενη Αναπαράσταση** Δυαδικού Δένδρου θα χρησιμοποιήσουμε:

- Έναν πίνακα από κόμβους μεγέθους $2^{H+1}-1$ (**data**), έναν ακέραιο Η (**height**)

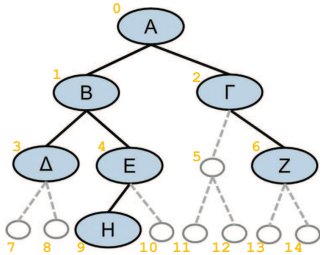
Θα ισχύσει η εξής σύμβαση:

- Το αριστερό παιδί του κόμβου i θα βρίσκεται στη θέση $2*i+1$
- Το δεξί παιδί του κόμβου i θα βρίσκεται στη θέση $2*i+2$
  - Άρα ο γονέας του κόμβου στη θέση i θα βρίσκεται στη θέση $(i-1)$ DIV 2

Π.χ. για το ακόλουθο δένδρο:



1. Υπολογίζουμε το ύψος
2. Νοητά Συμπληρώνουμε τους κενούς κόμβους
3. Συμπληρώνουμε τον πίνακα κατά επίπεδα
   (θα έχει 2^{3+1}-1=15 θέσεις)



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
data =	A	B	Γ	Δ	E		Z			H					

Α. Θεωρία

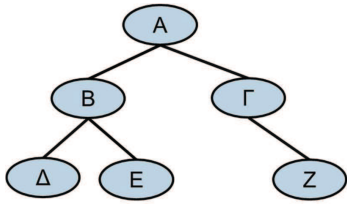
2. Δυαδικό Δένδρο

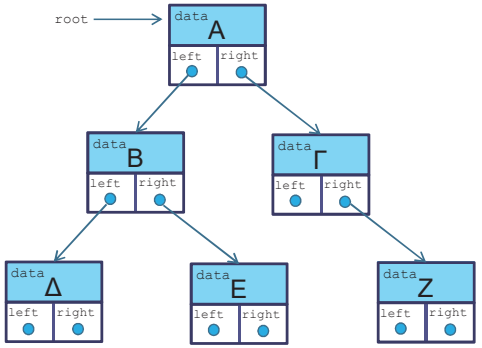
5. Δυναμική Αναπαράσταση

Στην δεύτερη υλοποίηση (**δυναμική αναπαράσταση**) θα χρησιμοποιήσουμε:

- Ο κόμβος θα αποτελείται από τα δεδομένα (**data**) και δύο δείκτες:
  - Έναν στο αριστερό παιδί (**left**)
  - Έναν στο δεξί παιδί (**right**)
- Ένας δείκτης θα είναι η ρίζα του δένδρου (συνηθίζεται να ονομάζεται **root**)

Π.χ. για το ακόλουθο δένδρο:





Παρατήρηση: Θέτουμε τον αντίστοιχο δείκτη (left ή right) ίσο με **NULL** για να δείξουμε ότι ο κόμβος δεν έχει αριστερό (ή δεξί παιδί).

Α. Θεωρία

3. Δυαδικά Δένδρα με Δυναμική Αναπαράσταση

1. Υλοποίηση σε C: Δηλώσεις

Οι **δηλώσεις** σε C είναι οι ακόλουθες:

- Ο κόμβος του δένδρου είναι μία δομή (struct) με τα εξής στοιχεία:
  - Το data μέρος του κόμβου (σε τύπο δεδομένων που ορίζουμε).
  - Τους δείκτες left και right που δείχνουν το αριστερό και το δεξί παιδί του κόμβου.

```
typedef int elem;           /* typos dedomenwn dendrou*/

struct node{                /* Typos komvou dendrou */
   elem data;               /* dedomena */
   struct node *left;       /* aristero paidi */
   struct node *right;      /* deksi paidi */
};

typedef struct node TREE_NODE; /* Sinwnimo tou komvou dendrou */
typedef struct node *TREE_PTR; /* Sinwnimo tou deikti komvou */
```

Το δένδρο θα είναι ένας δείκτης σε κόμβο δένδρου (θα δηλώνεται στη main).

Α. Θεωρία

3. Δυαδικά Δένδρα με Δυναμική Αναπαράσταση

2. Υλοποίηση σε C: Αρχικοποίηση Δένδρου

Η αρχικοποίηση γίνεται θέτοντας τον δείκτη δένδρου ίσο με NULL

```
/* TR_init(): arxikopoiei to dendro */
void TR_init(TREE_PTR *root)
{
   *root=NULL;
}
```

Προσοχή:

- Πάντα προτού ξεκινάμε την χρήση του δένδρου θα πρέπει να καλούμε μία φορά αυτήν τη συνάρτηση!

Α. Θεωρία

3. Δυαδικά Δένδρα με Δυναμική Αναπαράσταση

3. Υλοποίηση σε C: Έλεγχος – Άδειο Δένδρο

Ο **έλεγχος** αν το δένδρο είναι **κενό**, γίνεται βλέποντας αν ο δείκτης δένδρου είναι ίσος με NULL.

```
/* TR_empty(): epistrefei TRUE/FALSE
 *          analoga me to an to dendro einai adeio */
int TR_empty(TREE_PTR root)
{
   return root == NULL;
}
```


Α. Θεωρία

3. Δυαδικά Δένδρα με Δυναμική Αναπαράσταση

4. Υλοποίηση σε C: Περιεχόμενο Κόμβου

Η συνάρτηση επιστρέφει το περιεχόμενο (τα δεδομένα) ενός κόμβου.

```
/* TR_data(): epistrefei ta dedomena tou komvou
              pou deixnei o deiktis p */
elem TR_data(TREE_PTR p)
{
   return p->data;
}
```


Α. Θεωρία

3. Δυαδικά Δένδρα με Δυναμική Αναπαράσταση

5. Υλοποίηση σε C: Εισαγωγή ως Ρίζα

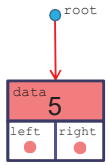
Η συνάρτηση «εισαγωγή ως ρίζα» εισάγει έναν νέο κόμβο ως ρίζα του δένδρου:

1. Δημιουργεί τον νέο κόμβο και θέτει τα δεδομένα σε αυτόν
2. Θέτει τον δείκτη ρίζας να δείχνει στον κόμβο αυτόν

Δημιουργία της ρίζας με δεδομένα «5»

ΠΡΙΝ

ΜΕΤΑ



Α. Θεωρία

3. Δυαδικά Δένδρα με Δυναμική Αναπαράσταση

5. Υλοποίηση σε C: Εισαγωγή ως Ρίζα

```
/* TR_insert_root(): Eisagei to stoixeio x
                     sti riza tou dendrou */
int TR_insert_root(TREE_PTR *root,elem x)
{
   TREE_PTR newnode;

   if (*root!=NULL)
      return FALSE;

   newnode=(TREE_NODE *)malloc(sizeof(TREE_NODE));
   if (!newnode)
   {
      printf("Adynamia desmeusis mnimis");
      return FALSE;
   }
   newnode->data=x;
   newnode->left=NULL;
   newnode->right=NULL;

   *root=newnode;
   return TRUE;
}
```

Α. Θεωρία

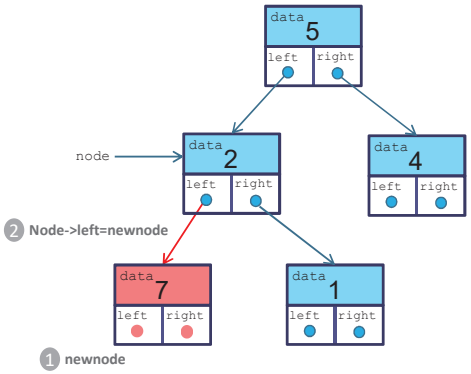
3. Δυαδικά Δένδρα με Δυναμική Αναπαράσταση

6. Υλοποίηση σε C: Εισαγωγή ως Αριστερό Παιδί

Η συνάρτηση «εισαγωγή ως αριστερό παιδί» εισάγει έναν νέο κόμβο ως αριστερό παιδί:

1. Δημιουργεί τον νέο κόμβο και θέτει τα δεδομένα σε αυτόν
2. Θέτει αριστερό παιδί του τρέχοντος κόμβου ίσο με τον νέο κόμβο

ΕΙΣΑΓΩΓΗ του «7» ως αριστερό παιδί του κόμβου 2»



Α. Θεωρία

3. Δυαδικά Δένδρα με Δυναμική Αναπαράσταση

6. Υλοποίηση σε C: Εισαγωγή ως Αριστερό Παιδί

```
/* TR_insert_left(): Eisagei to stoixeio x
                            ws aristero paidi tou node */
int TR_insert_left(TREE_PTR node,elem x)
{
   TREE_PTR newnode;

   if (node->left!=NULL)
      return FALSE;

   newnode=(TREE_NODE *)malloc(sizeof(TREE_NODE));
   if (!newnode)
   {
      printf("Adynamia desmeusis mnimis");
      return FALSE;
   }
   newnode->data=x;
   newnode->left=NULL;
   newnode->right=NULL;

   node->left=newnode;
   return TRUE;
}
```


Α. Θεωρία

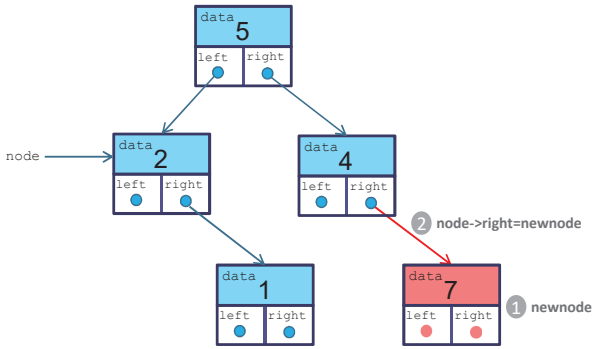
3. Δυαδικά Δένδρα με Δυναμική Αναπαράσταση

7. Υλοποίηση σε C: Εισαγωγή ως Δεξί Παιδί

Η συνάρτηση «εισαγωγή ως δεξί παιδί» εισάγει έναν νέο κόμβο ως δεξί παιδί:

1. Δημιουργεί τον νέο κόμβο και θέτει τα δεδομένα σε αυτόν
2. Θέτει δεξί παιδί του τρέχοντος κόμβου ίσο με τον νέο κόμβο

ΕΙΣΑΓΩΓΗ του «7» ως αριστερό παιδί του κόμβου 4»



Α. Θεωρία

3. Δυαδικά Δένδρα με Δυναμική Αναπαράσταση

7. Υλοποίηση σε C: Εισαγωγή ως Δεξί Παιδί

```
/* TR_insert_right(): Eisagei to stoixeio x
                            ws deksi paidi tou node */
int TR_insert_right(TREE_PTR node,elem x)
{
   TREE_PTR newnode;

   if (node->right!=NULL)
      return FALSE;

   newnode=(TREE_NODE *)malloc(sizeof(TREE_NODE));
   if (!newnode)
   {
      printf("Adynamia desmeusis mnimis");
      return FALSE;
   }
   newnode->data=x;
   newnode->left=NULL;
   newnode->right=NULL;

   node->right=newnode;
   return TRUE;
}
```

Α. Θεωρία

3. Δυαδικά Δένδρα με Δυναμική Αναπαράσταση

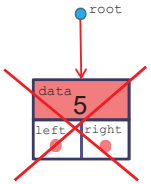
8. Υλοποίηση σε C: Διαγραφή της ρίζας

Η συνάρτηση δέχεται ως όρισμα τη ρίζα ενός δένδρου:

1. Ελέγχει ότι δεν έχει παιδία.
2. Διαγράφει τον κόμβο.
3. Θέτει τη ρίζα ίση με NULL

ΔΙΑΓΡΑΦΗ της ρίζας του δένδρου

ΠΡΙΝ



ΜΕΤΑ

Α. Θεωρία

3. Δυαδικά Δένδρα με Δυναμική Αναπαράσταση

8. Υλοποίηση σε C: Διαγραφή της ρίζας

```
/* TR_delete_root(): Diagrafei ti riza enos
        dentrou efoson den exei paidia */
int TR_delete_root(TREE_PTR *root, elem *x)
{
   if ((*root)->left!=NULL || (*root)->right!=NULL)
      return FALSE;

   *x=(*root)->data;
   free(*root);
   *root=NULL;
   return TRUE;
}
```


Α. Θεωρία

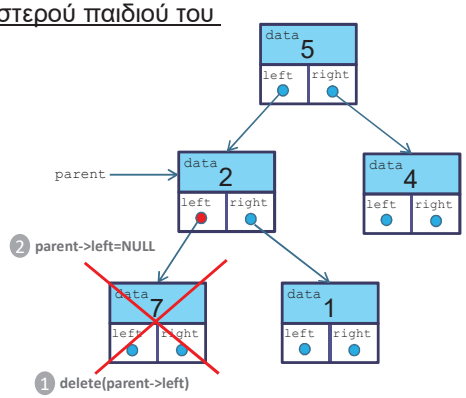
3. Δυαδικά Δένδρα με Δυναμική Αναπαράσταση

9. Υλοποίηση σε C: Διαγραφή αριστερού παιδιού κόμβου

Η συνάρτηση δέχεται ως όρισμα έναν κόμβο (στον οποίο δείχνει ο parent) και διαγράφει το αριστερό παιδί του:

1. Ελέγχει ότι το αριστερό παιδί δεν έχει παιδιά.
2. Διαγράφει το αριστερό παιδί.
3. Θέτει τον δείκτη left του κόμβου parent ίσο με NULL.

ΔΙΑΓΡΑΦΗ του αριστερού παιδιού του κόμβου «2»



Α. Θεωρία

3. Δυαδικά Δένδρα με Δυναμική Αναπαράσταση

9. Υλοποίηση σε C: Διαγραφή αριστερού παιδιού κόμβου

```
/* TR_delete_left(): Diagrafei to aristero paidi
        tou komvou parent (efoson den exei paidia) */
int TR_delete_left(TREE_PTR parent, elem *x)
{
   TREE_PTR current;

   if (parent->left==NULL)
      return FALSE;

   current=parent->left;
   if (current->left!=NULL || current->right!=NULL)
      return FALSE;

   *x=current->data;
   free(current);
   parent->left=NULL;
   return TRUE;
}
```

Α. Θεωρία

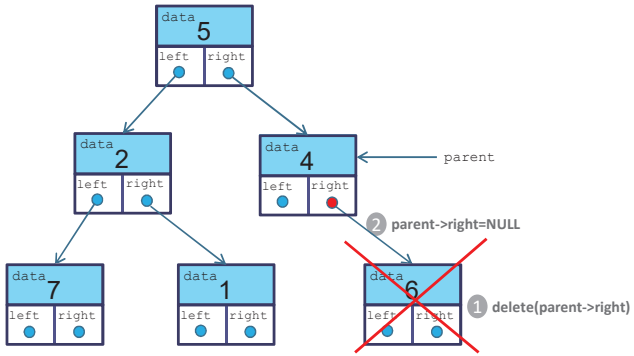
3. Δυαδικά Δένδρα με Δυναμική Αναπαράσταση

10. Υλοποίηση σε C: Διαγραφή δεξιού παιδιού κόμβου

Η συνάρτηση δέχεται ως όρισμα έναν κόμβο (στον οποίο δείχνει ο parent) και διαγράφει το δεξί παιδί του:

1. Ελέγχει ότι το δεξί παιδί δεν έχει παιδιά.
2. Διαγράφει το δεξί παιδί.
3. Θέτει τον δείκτη right του κόμβου parent ίσο με NULL.

ΔΙΑΓΡΑΦΗ του δεξιού παιδιού του κόμβου «4»



Α. Θεωρία

3. Δυαδικά Δένδρα με Δυναμική Αναπαράσταση

10. Υλοποίηση σε C: Διαγραφή δεξιού παιδιού κόμβου

```
/* TR_delete_right(): Diagrafei to deksi paidi
        tou komvou node (efoson den exei paidia) */
int TR_delete_right(TREE_PTR parent, elem *x)
{
   TREE_PTR current;

   if (parent->right==NULL)
      return FALSE;

   current=parent->right;
   if (current->left!=NULL || current->right!=NULL)
      return FALSE;

   *x=current->data;
   free(current);
   parent->right=NULL;
   return TRUE;
}
```


Α. Θεωρία

4. Διαπέραση Δυαδικών Δένδρων

1. Γενικά

Οι διαπεράσεις είναι μεθοδολογίες για να επισκεφθούμε όλες τις κορυφές ενός Δυαδικού Δένδρου:

- Εξετάζουμε:
  - Την προδιατεταγμένη διαπέραση (preorder). Που εκτελεί τη σειρά επίσκεψης:
    - Τρέχουσα Κορυφή
    - Αριστερό Υποδένδρο
    - Δεξί Υποδένδρο
  - Την ενδοδιατεταγμένη διαπέραση (inorder). Που εκτελεί τη σειρά επίσκεψης:
    - Αριστερό Υποδένδρο
    - Τρέχουσα Κορυφή
    - Δεξί Υποδένδρο
  - Την μεταδιατεταγμένη Διαπέραση (postorder). Που εκτελεί τη σειρά επίσκεψης:
    - Αριστερό Υποδένδρο
    - Δεξί Υποδένδρο
    - Τρέχουσα Κορυφή

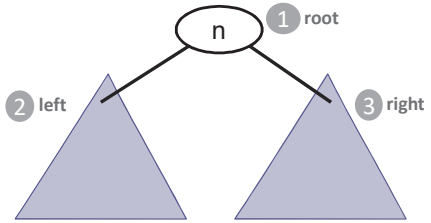
Α. Θεωρία

4. Διαπέραση Δυαδικών Δένδρων

2. Υλοποίηση σε C: Προδιατεταγμένη Διαδρομή

```
-------------------------------------------------------------
Αλγόριθμος Προδιατεταγμένης Διαδρομής (PRE-ORDER)
Είσοδος: Δυαδικό Δένδρο Τ
Έξοδος: Προδιατεταγμένη Διάσχιση των Κορυφών του Τ
-------------------------------------------------------------
procedure PRE-ORDER(n)
  Αν (n≠ΚΕΝΟ)
     Εκτύπωση του n
     PRE-ORDER(Αριστερό Παιδί της n)
     PRE-ORDER(Δεξί Παιδί της n)
  Τέλος-Αν
end procedure
```

ΠΡΟΔΙΑΤΕΤΑΓΜΕΝΗ ΔΙΑΠΕΡΑΣΗ

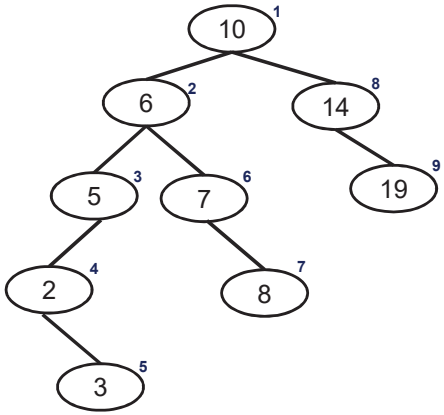


Α. Θεωρία

4. Διαπέραση Δυαδικών Δένδρων

2. Υλοποίηση σε C: Προδιατεταγμένη Διαδρομή

Παράδειγμα Εκτέλεσης της Προδιατεταγμένης Διαδρομής στο ακόλουθο δένδρο:



- Παράγεται η ακολουθία (σειρά επίσκεψης): 10,6,5,2,3,7,8,14,19

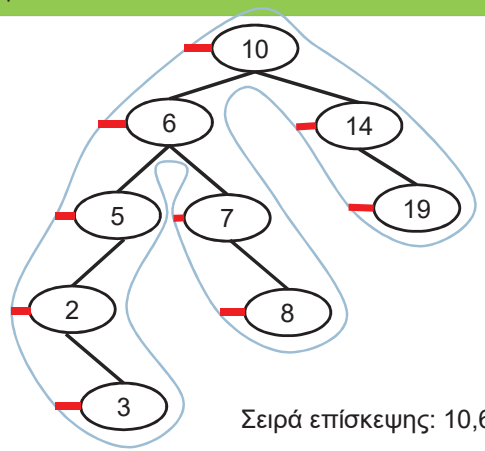
Α. Θεωρία

4. Διαπέραση Δυαδικών Δένδρων

2. Υλοποίηση σε C: Προδιατεταγμένη Διαδρομή

Εμπειρικός Τρόπος (εκτέλεσης της προδιατεταγμένης διαπέρασης):

- Κατασκευάζω το περίγραμμα και τραβάω γραμμή **αριστερά** από κάθε κόμβο.
- Έπειτα σαρώνω το περίγραμμα αριστερόστροφα από τη ρίζα και όπου συναντάω γραμμή καταγράφω τον κόμβο



Σειρά επίσκεψης: 10,6,5,2,3,7,8,14,19

Α. Θεωρία

4. Διαπέραση Δυαδικών Δένδρων

2. Υλοποίηση σε C: Προδιατεταγμένη Διαδρομή

```
/* TR_preorder(): Ektypwsi kata tin
        prodiatetagmeni diadromi   */
void TR_preorder(TREE_PTR v)
{
   if(v!=NULL)
   {
      TR_print_node(v);
      TR_preorder(v->left);
      TR_preorder(v->right);
   }
}
```

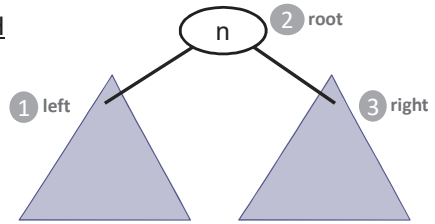

Α. Θεωρία

4. Διαπέραση Δυαδικών Δένδρων

3. Υλοποίηση σε C: ΕνδοΔιατεταγμένη Διαπέραση

```
-------------------------------------------------------------
Αλγόριθμος Ενδοδιατεταγμένης Διαδρομής (IN-ORDER)
Είσοδος: Δυαδικό Δένδρο Τ
Έξοδος: Ενδοδιατεταγμένη Διάσχιση των Κορυφών του Τ
-------------------------------------------------------------
procedure IN-ORDER(v)
  Αν (v≠ΚΕΝΟ)
     IN-ORDER(Αριστερό Παιδί της v)
     Εκτύπωση του v
     IN-ORDER(Δεξί Παιδί της v)
  Τέλος-Αν
end procedure
```

ΕΝΔΟΔΙΑΤΕΤΑΓΜΕΝΗ ΔΙΑΠΕΡΑΣΗ

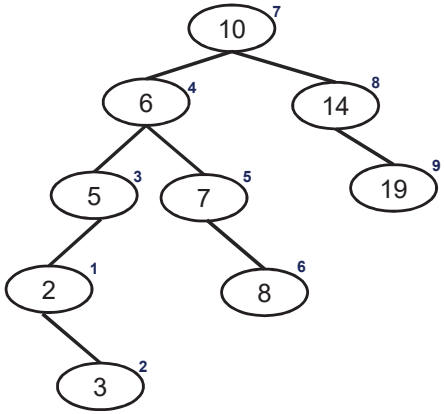


Α. Θεωρία

4. Διαπέραση Δυαδικών Δένδρων

3. Υλοποίηση σε C: ΕνδοΔιατεταγμένη Διαπέραση

Παράδειγμα Εκτέλεσης της Ενδοδιατεταγμένης Διαδρομής στο ακόλουθο δένδρο:



- Παράγεται η ακολουθία (σειρά επίσκεψης): 2,3,5,6,7,8,10,14,19

Σημείωση: Η ακολουθία εμφανίζεται σε αύξουσα σειρά γιατί το δένδρο έχει ειδική μορφή (Δυαδικό Δένδρο Αναζήτησης => Το μελετάμε στο επόμενο μάθημα)

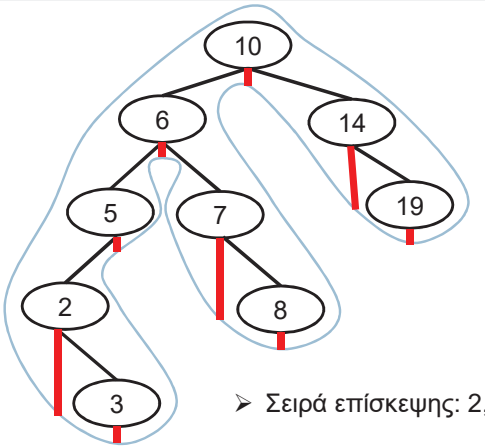
Α. Θεωρία

4. Διαπέραση Δυαδικών Δένδρων

3. Υλοποίηση σε C: ΕνδοΔιατεταγμένη Διαπέραση

Εμπειρικός Τρόπος (εκτέλεσης της ενδοδιατεταγμένης διαπέρασης):

- Κατασκευάζω το περίγραμμα και τραβάω γραμμή **κάτω** από κάθε κόμβο.
- Έπειτα σαρώνω το περίγραμμα αριστερόστροφα από τη ρίζα και όπου συναντάω γραμμή καταγράφω τον κόμβο



- Σειρά επίσκεψης: 2,3,5,6,7,8,10,14,19

Α. Θεωρία

4. Διαπέραση Δυαδικών Δένδρων

3. Υλοποίηση σε C: ΕνδοΔιατεταγμένη Διαπέραση

```
/* TR_inorder(): Ektypwsi kata tin
        endodiatetagmeni diadromi  */
void TR_inorder(TREE_PTR v)
{
   if(v!=NULL)
   {
      TR_inorder(v->left);
      TR_print_node(v);
      TR_inorder(v->right);
   }
}
```

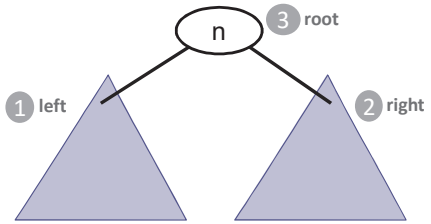

Α. Θεωρία

4. Διαπέραση Δυαδικών Δένδρων

5. Υλοποίηση σε C: ΜεταΔιατεταγμένη Διαπέραση

```
-------------------------------------------------------------
Αλγόριθμος Μεταδιατεταγμένης Διαδρομής (POST-ORDER)
Είσοδος: Δυαδικό Δένδρο Τ
Εξοδος: Μεταδιατεταγμένη Διάσχιση των Κορυφών του Τ
-------------------------------------------------------------
procedure POST-ORDER(v)
  Αν (v≠ΚΕΝΟ)
     POST-ORDER(Αριστερό Παιδί της v)
     POST-ORDER(Δεξί Παιδί της v)
     Εκτύπωση του v
  Τέλος-Αν
end procedure
```

ΜΕΤΑΔΙΑΤΕΤΑΓΜΕΝΗ ΔΙΑΠΕΡΑΣΗ

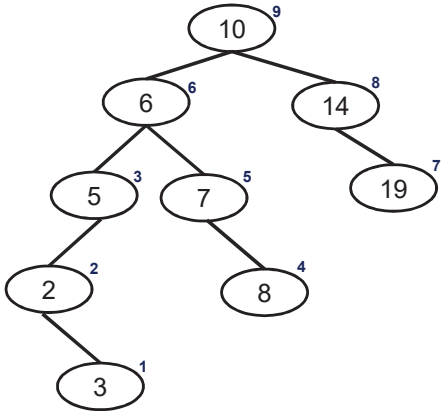


Α. Θεωρία

4. Διαπέραση Δυαδικών Δένδρων

4. Υλοποίηση σε C: ΜεταΔιατεταγμένη Διαπέραση

Παράδειγμα Εκτέλεσης της Μεταδιατεταγμένης Διαδρομής στο ακόλουθο δένδρο:



- Παράγεται η ακολουθία (σειρά επίσκεψης): 3,2,5,8,7,6,19,14,10

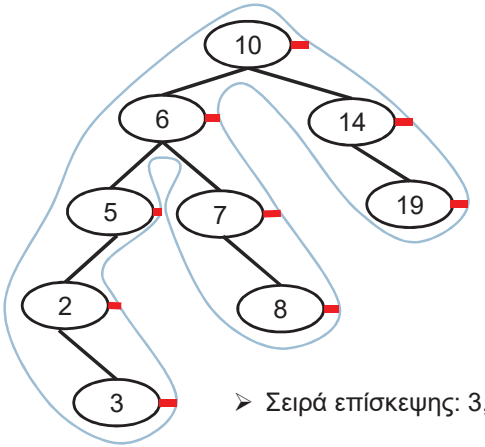
Α. Θεωρία

4. Διαπέραση Δυαδικών Δένδρων

4. Υλοποίηση σε C: ΜεταΔιατεταγμένη Διαπέραση

Εμπειρικός Τρόπος (εκτέλεσης της ενδοδιατεταγμένης διαπέρασης):

- Κατασκευάζω το περίγραμμα και τραβάω γραμμή **δεξιά** από κάθε κόμβο.
- Έπειτα σαρώνω το περίγραμμα αριστερόστροφα από τη ρίζα και όπου συναντάω γραμμή καταγράφω τον κόμβο



- Σειρά επίσκεψης: 3,2,5,8,7,6,19,14,10

Α. Θεωρία

4. Διαπέραση Δυαδικών Δένδρων

4. Υλοποίηση σε C: ΜεταΔιατεταγμένη Διαπέραση

```
/* TR_postorder(): Ektypwsi kata tin
        metadiatetagmeni diadromi  */
void TR_postorder(TREE_PTR v)
{
        if(v!=NULL)
        {
                TR_postorder(v->left);
                TR_postorder(v->right);
                TR_print_node(v);
        }
}
```


Β. Ασκήσεις

Εφαρμογή 1: Μελέτη Προγράμματος

- Μελετήστε το project tree.dev στο οποίο υλοποιούνται οι βασικές πράξεις των δένδρων που μελετήσαμε στο μάθημα. Σε αυτό θα «πατήσουμε» στο επόμενο μάθημα για να κατασκευάσουμε τα δυαδικά δένδρα αναζήτησης.