Εργαστήριο 8: Αναδρομική διεργασία εισαγωγής καινούριου κόμβου σε ΔΔΑ

Στην ενότητα αυτή θα μελετηθούν τα εξής επιμέρους θέματα:

- Αναδρομική διεργασία εισαγωγής καινούριου κόμβου σε ΔΔΑ με αλφαβητική σειρά

Επανάληψη Αναδρομής

- Βασική έννοια στα Μαθηματικά και στην Πληροφορική.

- Στην πληροφορική η αναδρομή χρησιμοποιείται σαν *τεχνική προγραμματισμού* και σαν *μέθοδος σχεδιασμού αλγορίθμων*.

- Στον προγραμματισμό η αναδρομή εμφανίζεται με την **κλήση ενός υποπρογράμματος από τον εαυτό του**. Ένα αναδρομικό υποπρόγραμμα αποτελείται από:
  - ένα **βήμα τερματισμού**, όπου ορίζεται η εκτέλεση του υποπρογράμματος για κάποιες "μικρές "τιμές των παραμέτρων του, και
  - ένα **αναδρομικό βήμα**, κατά το οποίο η εκτέλεση του υποπρογράμματος ορίζεται ως συνδυασμός κλήσεων του υποπρογράμματος σε άλλες "μικρότερες" τιμές των παραμέτρων.

Παράδειγμα 1: Παραγοντικός Αριθμός με Αναδρομή

- Ας ορίσουμε τώρα τον αναδρομικό ορισμό της factorial.

  0! = 1 , **1!** = 1 **2!** = 1x2 = 2 **3!** = 1x2x3 = 6

  4! = 1x2x3x4=24 **5!** = 1x2x3x4x5 = 120

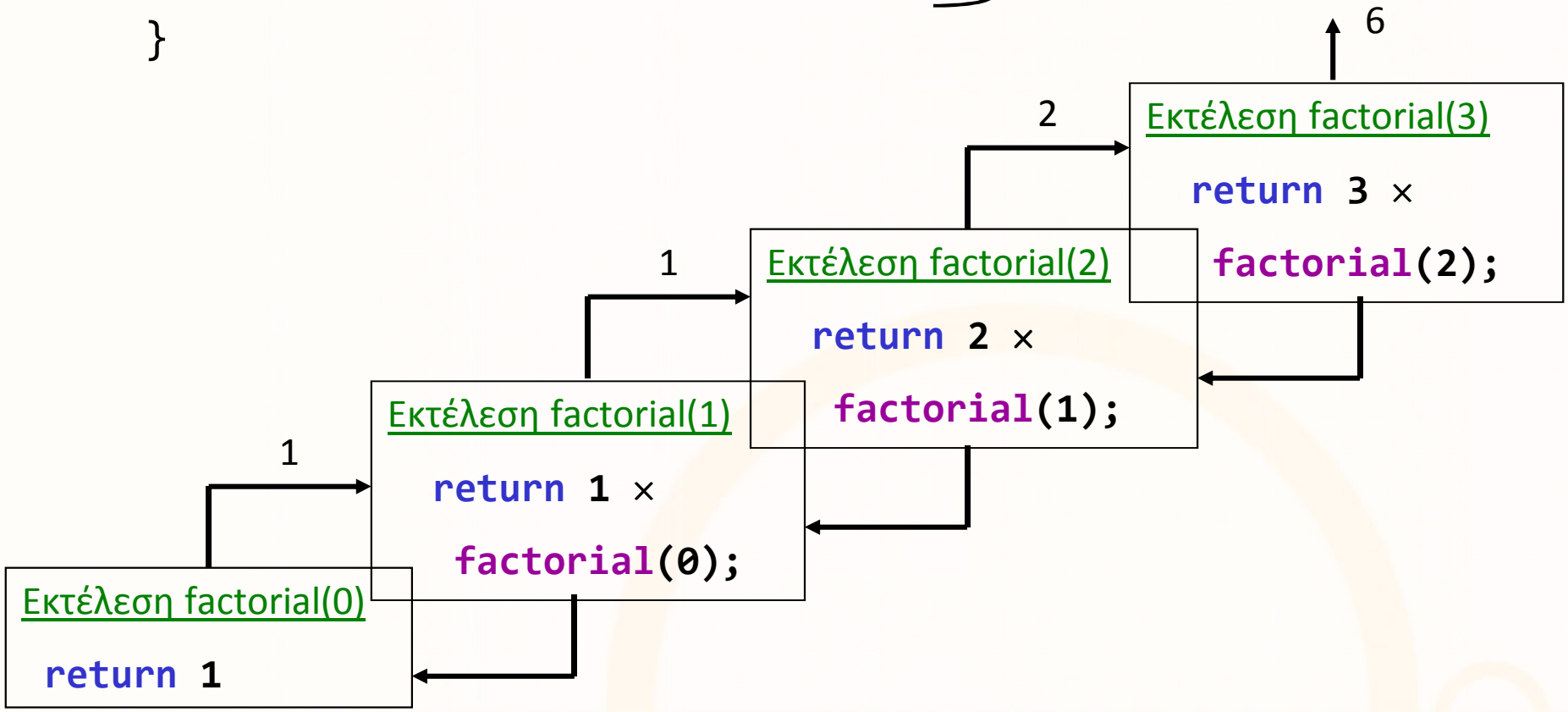
- Ας ορίσουμε τώρα τον αναδρομικό ορισμό της factorial.
- **Αναδρομικός Ορισμός συνάρτησης <factorial>**
  - Βήμα Τερματισμού: **0! = 1**
  - Αναδρομικό Βήμα: **n! = n x (n-1)!**

Παραγοντικός Αριθμός με Αναδρομή

```
int factorial ( int n ) {
        if (n == 0)
                return 1;
        else
                return n x factorial(n-1);
        }
```

Βήμα Τερματισμού

Αναδρομικό Βήμα

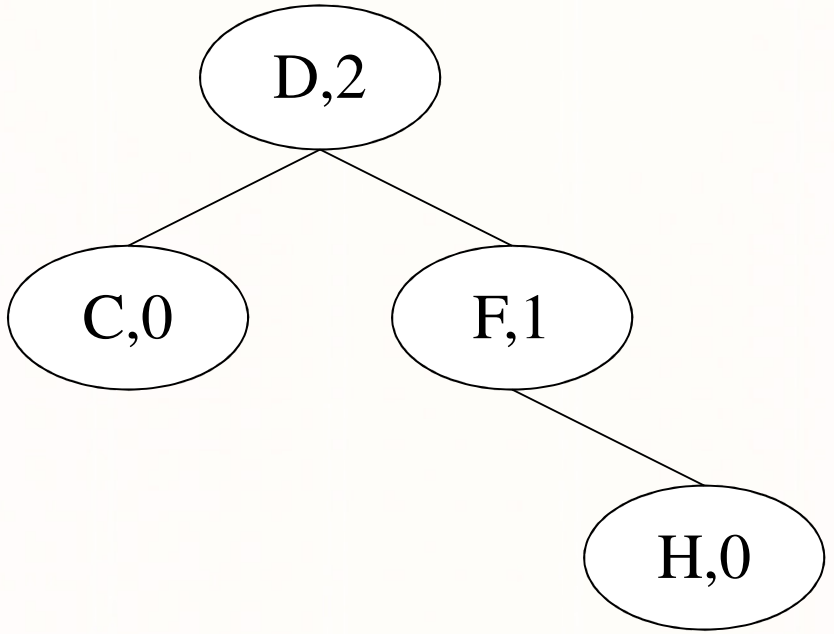


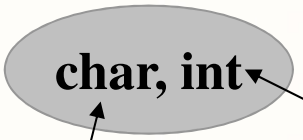
Δυαδικό Δέντρο Αναζήτησης με Αλφαβητική Σειρά

Μετά από εισαγωγή του D, C, F, H

Τι διαφέρει αν αρχίζαμε από την αρχή και η σειρά εισαγωγής ήταν:

1. D, F, C, H
2. D, C, H, F
3. C, D, F, H





Ακέραιος αριθμός που αντιπροσωπεύει το ύψος του κόμβο.

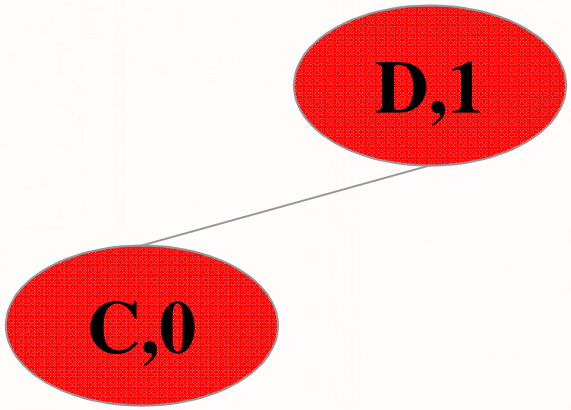
Ένας χαρακτήρας του αγγλικού αλφαβήτου

ΠΑΡΑΔΕΙΓΜΑ ΕΚΤΕΛΕΣΗΣ

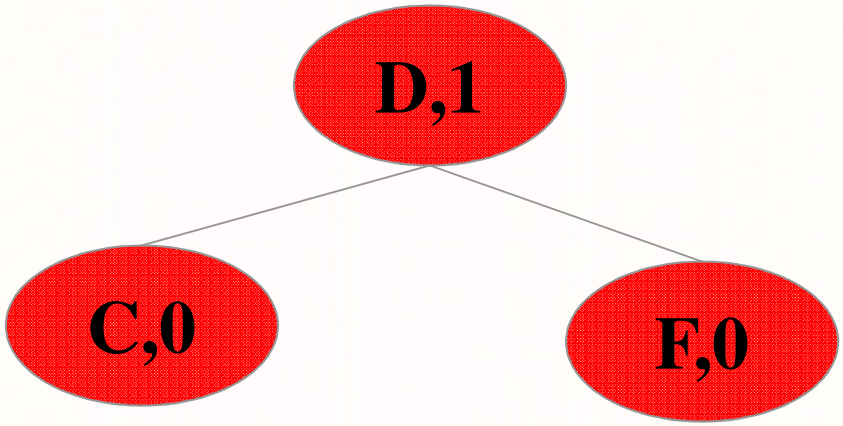
Εισαγωγή των κόμβων D,C,F,H,A,E,B,G

Εισαγωγή του κόμβου: D

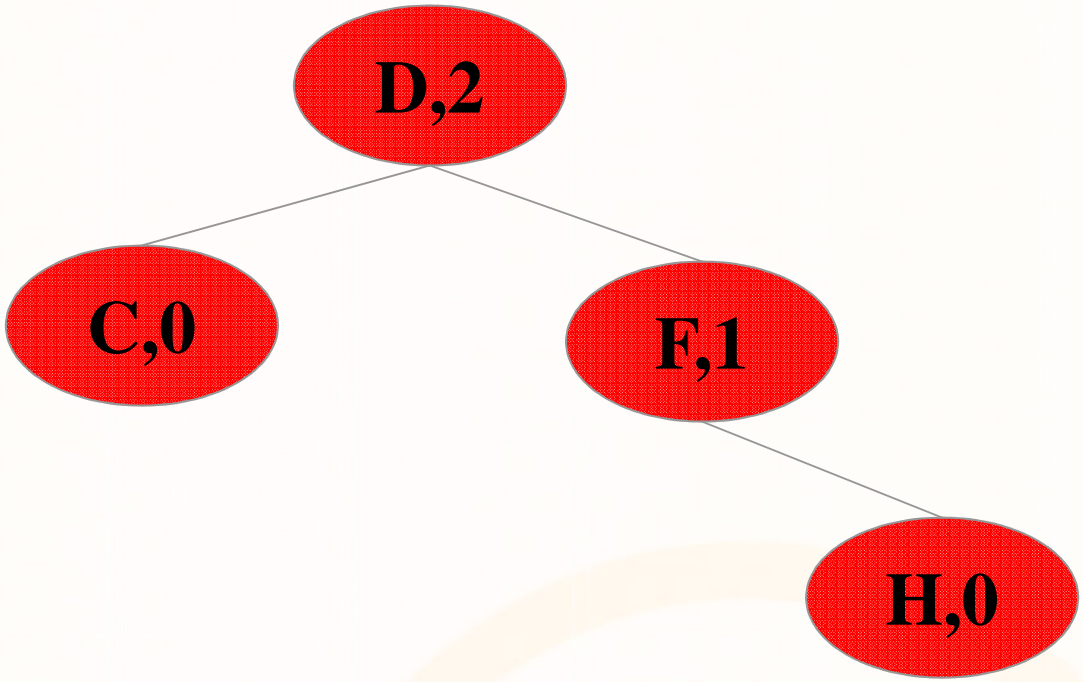
Εισαγωγή του κόμβου C



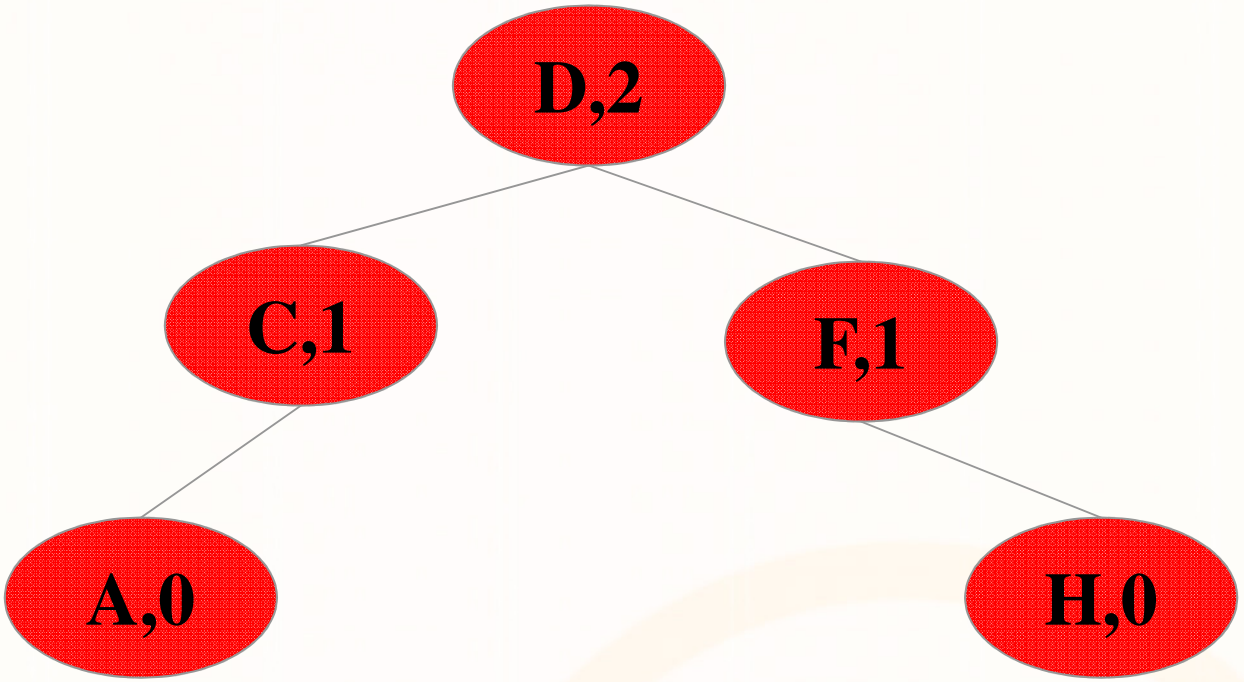
Εισαγωγή του κόμβου F



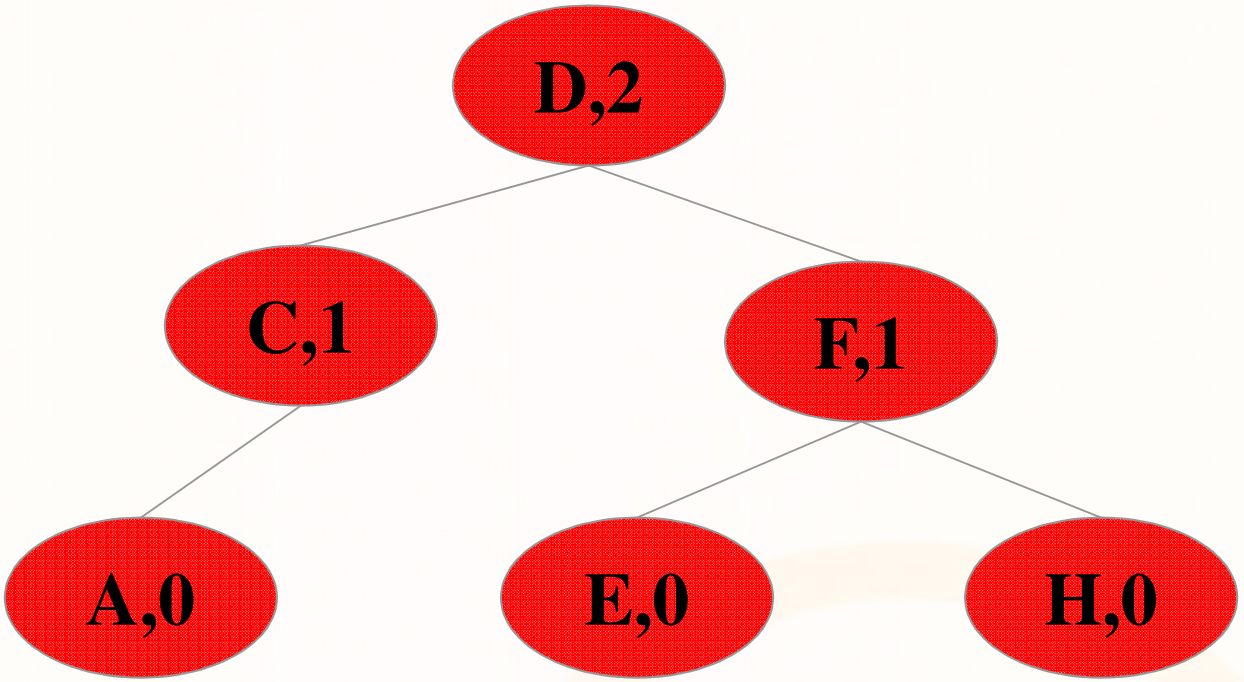
Εισαγωγή του κόμβου H



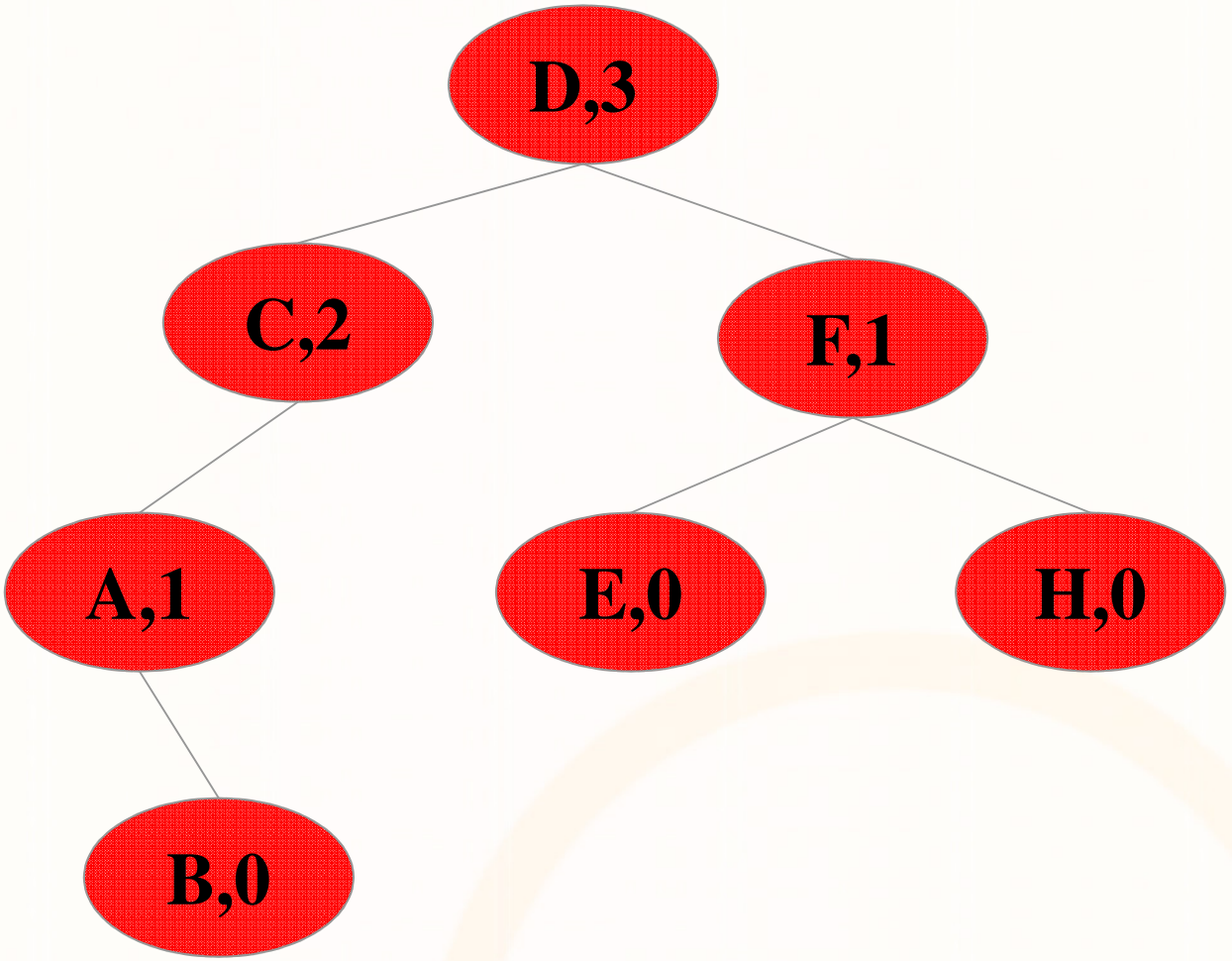
Εισαγωγή του κόμβου A



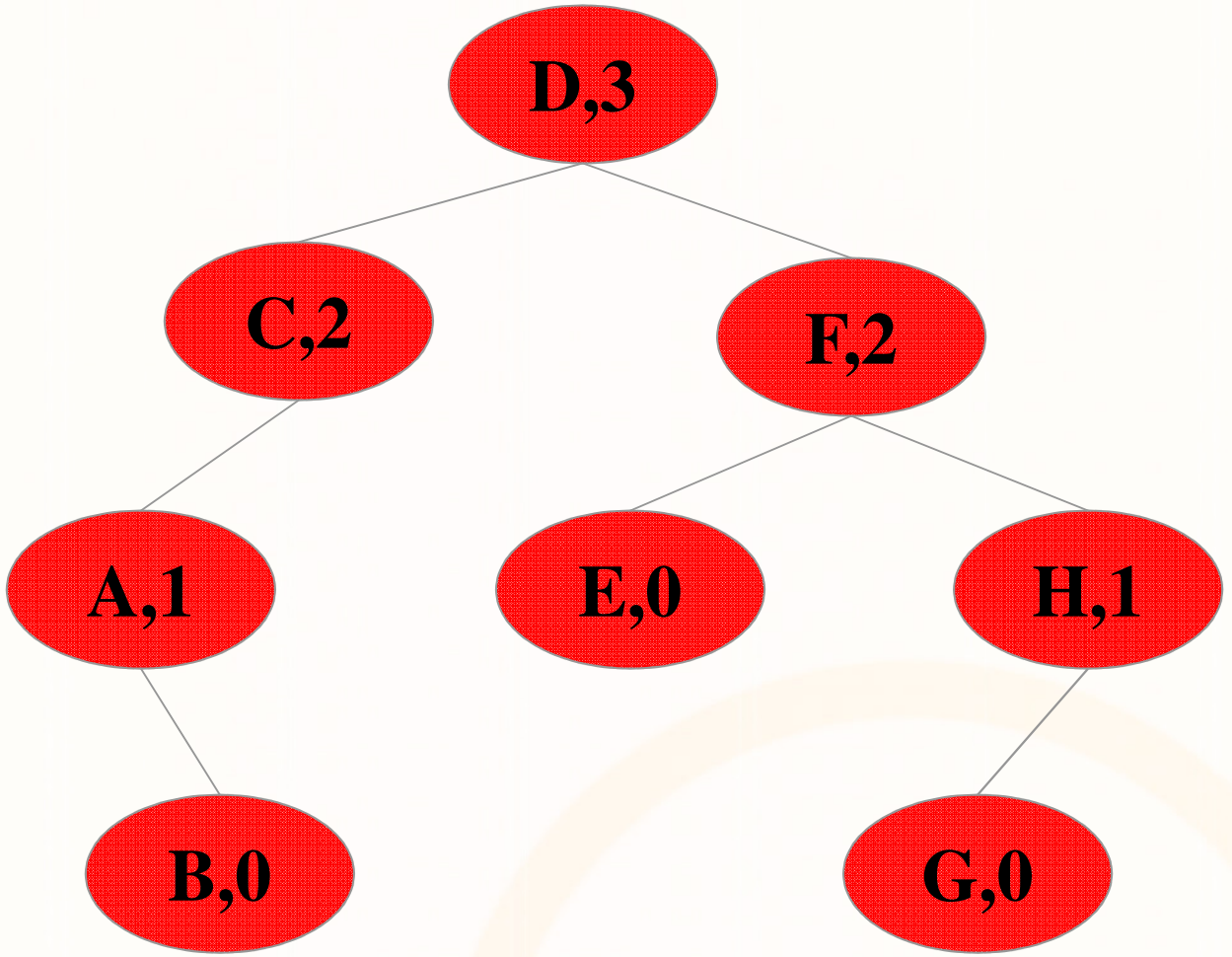
Εισαγωγή του κόμβου Ε



Εισαγωγή του κόμβου B



Εισαγωγή του κόμβου G



Υλοποίηση ΔΔΑ

Δομές που θα χρειαστείτε:

```
//The tnode data structure
typedef struct tnode{
      char name;              //το όνομα του κόμβου
      int height;             //το ύψος του κόμβου
      struct tnode *left;     //Το αριστερό παιδί
      struct tnode *right;    //Το δεξί παιδί
}tnode;

typedef struct tnode *Tnode;
```

Συναρτήσεις που πρέπει να υλοποιήσετε

- `int max(Tnode left, Tnode right);`
  επιστρέφει το μέγιστο ύψος 2 υποδένδρων

- `Tnode Insert(Tnode t, char c);`
  εισάγει ένα καινούριο κόμβο με το όνομα c στο δέντρο t στην σωστή θέση και επιστρέφει την ανάθεση αυτή

- `void printTree(Tnode t);`
  τυπώνει το δέντρο t.

Λύσεις

```
int max(Tree left, Tree right){
      if(left==NULL && right==NULL)
            return 0;
      if(left==NULL)
            return right->height;
      if(right==NULL)
            return left->height;
      if(left->height < right->height)
            return right->height;
      else
            return left->height;
}
```

Λύσεις (συν.)

```
Tree Insert(Tree t, char c){
      if (t==NULL) {
            //Initialize a new node
            printf("Inserting new node (%c:0).\n", c);
            t = (tree*) malloc ( sizeof( tree ) );
            t->height=0;
            t->name = c;
            t->left = NULL;
            t->right = NULL;
            return t;
      }
…
```

Λύσεις (συν.)

```
…
else{
        printf("Current Node\nName: %c\nHeight: %d\n",
                t->name, t->height);
        if(c < t->name){ //Move to the left
                printf("Moving to the left of node
                        (%c:%d)\n", t->name, t->height);
                t->left = Insert(t->left, c);}
        else{ //Move to the right
                printf("Moving to the right of node
                        (%c:%d)\n", t->name, t->height);
                t->right = Insert(t->right, c);
        }
        //Set the height
        t->height = 1 + max(t->left, t->right);
        return t; } } }
```

Λύσεις (συν.)

```
void printTree(Tree t, int h){
        int i;
        if(t!=NULL){
                printTree(t->left, h-1);
                for(i=0; i<h*10; i++) printf(" ");
                        printf("(%c:%d)\n", t->name, t->height);
                printTree(t->right, h-1);
        }
}
```

Λύσεις (συν.)

```
void InsertNewNode(Tree t, char c){
        printf("Trying to insert new node with name [%c]\n", c);
        root = Insert(root, c);
        printf("Done\n\n");
}
```

Εναλλακτική Λύση Insert με void

```
void Insert(tnode **t, char c){
  if(*t==NULL){
    (*t)=(tnode *) malloc( sizeof( tnode ) );
    (*t)->height=0;
    (*t)->left=NULL;
    (*t)->right=NULL;
    (*t)->Name=c;
  }else if((*t)->Name > c){
    Insert(&((*t)->left),c);
    (*t)->height= max((*t)->left,(*t)->right)+1;
  }else if((*t)->Name < c){
    Insert(&((*t)->right),c);
    (*t)->height= max((*t)->left,(*t)->right)+1;
  }
}
```