

Συναρτήσεις

Συνάρτηση : create

- ❖ Δεν φέρνει κάτι για να χρησιμοποιήσει
- ❖ Επιστροφή : pointer την διεύθυνση του κόμβου όπου θα δημιουργήσει τα παιδιά

```
struct node *create(){ //dimiourgia paidion tou k  
    struct node *komv ;  
    komv = (struct node *)malloc(sizeof(struct node));  
    int i ;  
    for (i = 0; i < 26; i++) { // dimiourgia 26 paidion  
        komv->pinakas[i] = NULL;  
    }  
    return komv;  
}
```

- Με δυναμική δέσμευση δημιουργεί χώρο , τύπου struct node για τον κόμβο
- Στην συνέχεια εντός του for δημιουργία / αρχικοποιεί τα 26 παιδιά (όσα τα γράμματα του αλφάβητου) με NULL (γιατί ακόμα δεν έχουμε επιλέξει ποια πρέπει να χρησιμοποιηθούν)
- Στο τέλος επιστρέφει την διεύθυνση του χώρου που μόλις έχω δημιουργήσει

Συνάρτηση : InsertNode

- ❖ .Φέρνει το δέντρο μέσα (root , από την κορυφή) και την συμβολοσειρά που θέλει να εισάγει στο δέντρο
- ❖ .Δεν επιστρέφει κάτι η συνάρτηση

```
void InsertNode(struct node *root , char* s){  
    int i ;  
    int arth ;  
    struct node* komv = root; // ksekinontas apo root  
    for(i=0;i<10;i++){  
        arth=s[i] - 'A' ; //arithmos gramatos ko  
        if (komv->pinakas[arth] == NULL) /  
            komv->pinakas[arth] = create();  
        komv = komv->pinakas[arth];  
    }  
    komv->used = false ; //afpu tora dimiour  
    komv->is_end = true; //telos simvolosira  
}
```

- για να μην επηρεάσω την κάθε διαδρομή / δέντρου με το ίσον αντιγράφω τα χαρακτηριστικά της δημιουργίας του (τους κόμβους)
- στο for κάθε φορά εντάσσω στο δέντρο τον χαρακτήρα που έχει σειρά
- πως;
- με το arth κρατάω τον αριθμό που του αναλογεί .Αν σε εκείνο το αριθμό το παιδί είναι ακόμη null (δλδ δεν έχει παιδιά) τότε καλώ την συνάρτηση create για να μου δημιουργήσει τα παιδιά του συγκεκριμένου κόμβου
- και στην συνέχεια προχωρώ στον κόμβο που δημιούργησα ή προϋπήρχε (komv=komp->pinakas[arth];)
- όταν τελειώσω με την εισαγωγή όλων των γραμμάτων τότε ενημερώνω ότι έφτασα στο τελευταίο γράμμα και ότι ακόμα δεν έχει χρησιμοποιηθεί .

Συνάρτηση : Search

- ❖ . Φέρνει το δέντρο με τις πληροφορίες και το κουπόνι που έχουμε διαβάσει .
- ❖ . Επιστρέφει true or false (το χρησιμοποιούμε κυρίως για να σταματά την συνάρτηση)

```
bool search(struct node *root, char kouponi[10] ){  
    struct node* komvos = root; // ksekinontas pao tin kefali g na min pirakso to root  
    int p ;  
    int i=0 ;  
    int arth ;  
    for(p=0 ; p<10 ; p++){  
        arth = kouponi[p] - 'A'; //arithos gramatos kouponiou  
        if (komvos->pinakas[arth]==NULL) { //an den exi mesa o komvos  
            printf("is not valid k ");  
            break;  
        }  
        komvos = komvos->pinakas[arth]; //metakinisi ston epomeno komvo t epipedou  
    }  
    if(komvos->is_end== false){ // an den ftasame sto telos tis lexis  
        printf("\nis nott valid ");  
        return false ;  
    }
```

- Στόχος συνάρτησης : να ψάξει αν υπάρχει το κουπόνι και αν έχει ξανά χρησιμοποιηθεί .
- Για να μην επηρεάσω την κορυφή της διαδρομής / δέντρου αντιγράφω τα χαρακτηριστικά σε άλλο κόμβους)
- Δημιουργώ ένα for που αναζητά με την σειρά κάθε γράμμα του κουπονιού αν υπάρχει στο δέντρο .
- στο κάθε επίπεδο αναζητά αν ο κόμβος που αντιστοιχεί στο γράμμα είναι δημιουργημένος και αν όχι τότε λέει οτι δεν υπάρχει το κουπόνι και βγαίνει από την συνάρτηση.

Στην συνέχεια διενεργώ τους πιο κάτω ελέγχους και στο τέλος κάθε ελέγχου βγαίνω από την συνάρτηση

1. αν δεν έχουμε φτάσει μέχρι το τέλος της διαδρομής τότε ο κόμβος δεν είναι διαθέσιμος .

```
if(komvos->is_end== false){ // an den ftasame sto telos tis l  
    printf("\nis nott valid ");  
    return false ;  
}  
else if(komvos->is_end == true && komvos->used== false ){  
    printf("\nis valid");  
    komvos->used=true ;  
    return true;  
}  
else if (komvos->is_end==true && komvos->used == true ){ // te  
    printf("\nis not valid ");  
    return false ;  
}  
return false;
```

2. αν είμαστε στο τέλος της διαδρομής και δεν έχει χρησιμοποιηθεί τότε ο κόμβος είναι διαθέσιμος .


3. αν είμαστε στο τέλος της διαδρομής και ο κόμβος έχει χρησιμοποιηθεί τότε ο κόμβος δεν είναι διαθέσιμος .

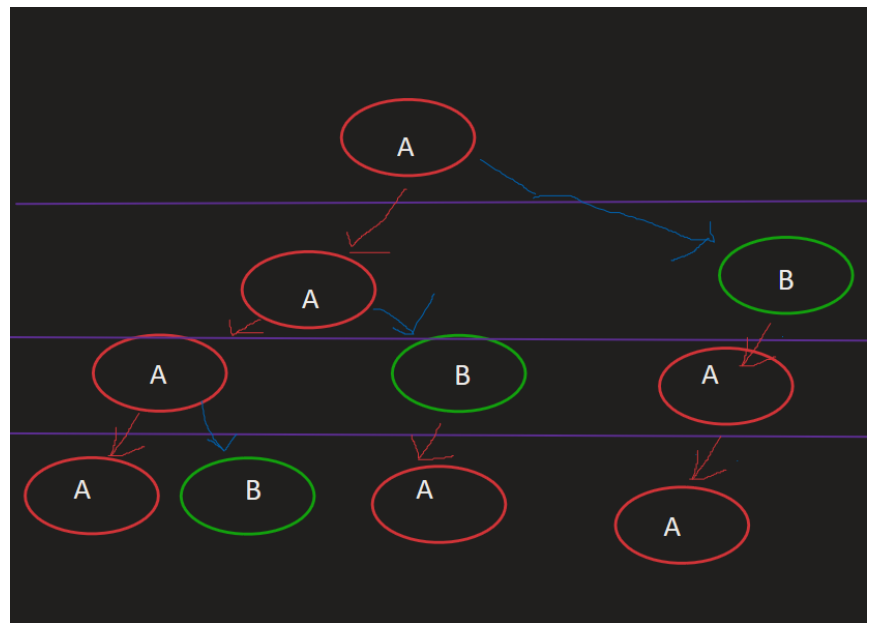
Διαδικασία

1.AAAA

- υπάρχει ο κόμβος A ; όχι δημιουργώ A
- υπάρχει παιδί A ; όχι δημιουργώ A αριστερά πάω αριστερά
- υπάρχει παιδί A ; όχι δημιουργώ A αριστερά πάω αριστερά
- υπάρχει παιδί A ; όχι δημιουργώ A αριστερά

2.AAAB

- υπάρχει ο κόμβος A ; ναι χρησιμοποιώ A
 - υπάρχει παιδί A ; ναι ,χρησιμοποιώ A , πάω αριστερά για το επόμενο
 - υπάρχει παιδί A ; ναι ,χρησιμοποιώ A , πάω δεξιά για το επόμενο
 - υπάρχει παιδί B ; όχι δημιουργώ B δεξιά
- 



3. AABA

- υπάρχει ο κόμβος A ;
ναι χρησιμοποιώ A
- υπάρχει παιδί A ; ναι
, χρησιμοποιώ A , πάω
δεξιά για το επόμενο
- υπάρχει παιδί B ; όχι
δημιουργώ B δεξιά , πάω αριστερά για το επόμενο
- υπάρχει παιδί A ; όχι δημιουργώ A αριστερά

4. ABAA

- υπάρχει ο κόμβος A ; ναι χρησιμοποιώ A
- υπάρχει παιδί B ; όχι δημιουργώ B δεξιά , πάω αριστερά για το επόμενο
- υπάρχει παιδί A ; όχι δημιουργώ A αριστερά πάω αριστερά
- υπάρχει παιδί A ; όχι δημιουργώ A αριστερά