Παρουσίαση Θέματος 3 (Γεωργιάδης Νικόλαος):

Στο αρχείο word “Check_if_grammar_is_LL(1).docx” βρίσκεται ο υπολογισμός των συνόλων FIRST,FOLLOW,EMPTY και LOOKAHEAD, η εύρεση των οποίων μας βοηθάει να καταλήξουμε σε ένα συμπέρασμα για το αν η γραμματική είναι LL(1).

Χρησιμοποιούνται δύο τρόποι για τον έλεγχο της γραμματικής.

Ο ένας αποτελεί τον έλεγχο ύπαρξης κοινών στοιχείων στα σύνολα LOOKAHEAD για τους κανόνες με ίδιο αριστερό μέλος. Καταλήγουμε στο ότι δεν υπάρχουν κοινά στοιχεία σε αυτά τα σύνολα, άρα η γραμματική είναι LL(1).

Ο άλλος τρόπος, αποτελεί την δημιουργία του συντακτικού πίνακα. Παρακάτω φαίνεται ο συντακτικός πίνακας που αντιστοιχεί στη γραμματική:

	(	)	α	β	*	-	+	$
S	S→(X)							
X	X→YZ		X→YZ	X→YZ				
Y	Y→S		Y→α	Y→β				
Z		Z→ε			Z→*X	Z→-X	Z→+X	

Παρατηρούμε πως δεν υπάρχει πάνω από ένας κανόνας σε κάποιο κελί, άρα η γραμματική είναι LL(1).

Στη συνέχεια, γίνεται επεξήγηση του αλγόριθμου και του σκεπτικού με τα οποία υλοποιήθηκε ο πηγαίος κώδικας c++ για την δημιουργία του συντακτικού αναλυτή topdown και του συντακτικού δέντρου. Προηγούνται αναφορές και στο εκτελέσιμο.

(το πρόγραμμα βρίσκεται στον προορισμό

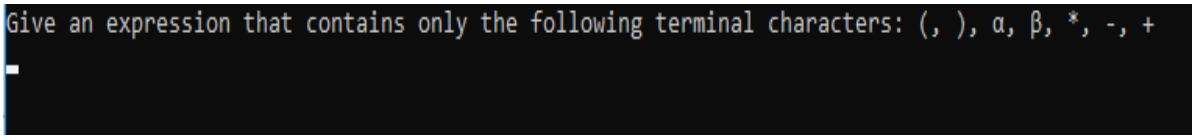
top_down_parser\top_down_parser.cpp

μαζί με το εκτελέσιμο και το header αρχείο που αναφέρεται παρακάτω)

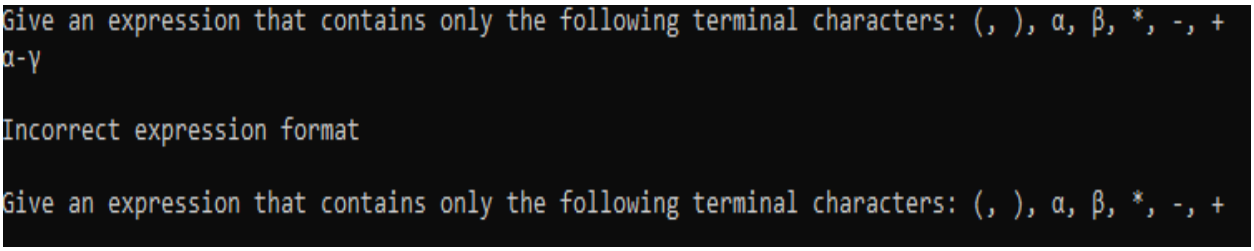
Να σημειωθεί πως έχουν χρησιμοποιηθεί τύποι μεταβλητών, συναρτήσεις, βιβλιοθήκες και εντολές όπως:

“wstring”,“wchar_t”,”wcout”,”wcin”,“<fcntl.h> “ “<io.h>“,_“setmode(_fileno(stdout/stdin), _O_WTEXT) “ κ.α. για την επίτευξη χρήσης ελληνικών γραμμάτων στη ροή εισόδου και εξόδου.

Κατά την εκκίνηση εκτέλεσης του προγράμματος, ζητείται από τον χρήστη να δώσει μια έκφραση έγκυρης μορφής:



Στην περίπτωση λανθασμένης εισόδου εμφανίζεται το ανάλογο μήνυμα και ξαναζητείτε είσοδος από τον χρήστη:



Ας κάνουμε επίδειξη της ζητούμενης έκφρασης της εκφώνησης ((β-α)*(α+β)), για να εξηγήσουμε το σκεπτικό και στη συνέχεια τον αλγόριθμο:

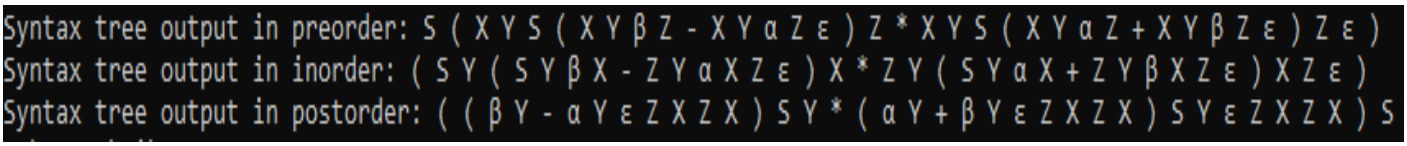
```
Give an expression that contains only the following terminal characters: (, ), α, β, *, -, +
((β-α)*(α+β))

Stack:                  Input:                  Table element:          Production:
$S                      ((β-α)*(α+β))$          M(S,(                   S-->(X)
$)X(                    ((β-α)*(α+β))$
$)X                     (β-α)*(α+β))$           M(X,(                   X-->YZ
$)ZY                    (β-α)*(α+β))$           M(Y,(                   Y-->S
$)ZS                    (β-α)*(α+β))$           M(S,(                   S-->(X)
$)Z)X(                  (β-α)*(α+β))$
$)Z)X                   β-α)*(α+β))$            M(X,β)                  X-->YZ
$)Z)ZY                  β-α)*(α+β))$            M(Y,β)                  Y-->β
$)Z)Zβ                  β-α)*(α+β))$
$)Z)Z                   -α)*(α+β))$             M(Z,-)                  Z-->-X
$)Z)X-                  -α)*(α+β))$
$)Z)X                   α)*(α+β))$              M(X,α)                  X-->YZ
$)Z)ZY                  α)*(α+β))$              M(Y,α)                  Y-->α
$)Z)Zα                  α)*(α+β))$
$)Z)Z                   )*(α+β))$               M(Z,))                  Z-->ε
$)Z)                    )*(α+β))$
$)Z                     *(α+β))$                M(Z,*)                  Z-->*X
$)X*                    *(α+β))$
$)X                     (α+β))$                 M(X,(                   X-->YZ
$)ZY                    (α+β))$                 M(Y,(                   Y-->S
$)ZS                    (α+β))$                 M(S,(                   S-->(X)
$)Z)X(                  (α+β))$
$)Z)X                   α+β))$                  M(X,α)                  X-->YZ
$)Z)ZY                  α+β))$                  M(Y,α)                  Y-->α
$)Z)Zα                  α+β))$
$)Z)Z                   +β))$                   M(Z,+)                  Z-->+X
$)Z)X+                  +β))$
$)Z)X                   β))$                    M(X,β)                  X-->YZ
$)Z)ZY                  β))$                    M(Y,β)                  Y-->β
$)Z)Zβ                  β))$
$)Z)Z                   ))$                     M(Z,))                  Z-->ε
$)Z)                    ))$
$)Z                     )$                      M(Z,))                  Z-->ε
$)                      )$
$                       $

Stack is empty, so the expression has been recognized
```

- Η πρώτη στήλη αντιστοιχεί στο περιεχόμενο της στοίβας σε κάθε βήμα.
- Η δεύτερη στήλη αντιστοιχεί στην είσοδο που διαβάζεται από το πρόγραμμα.

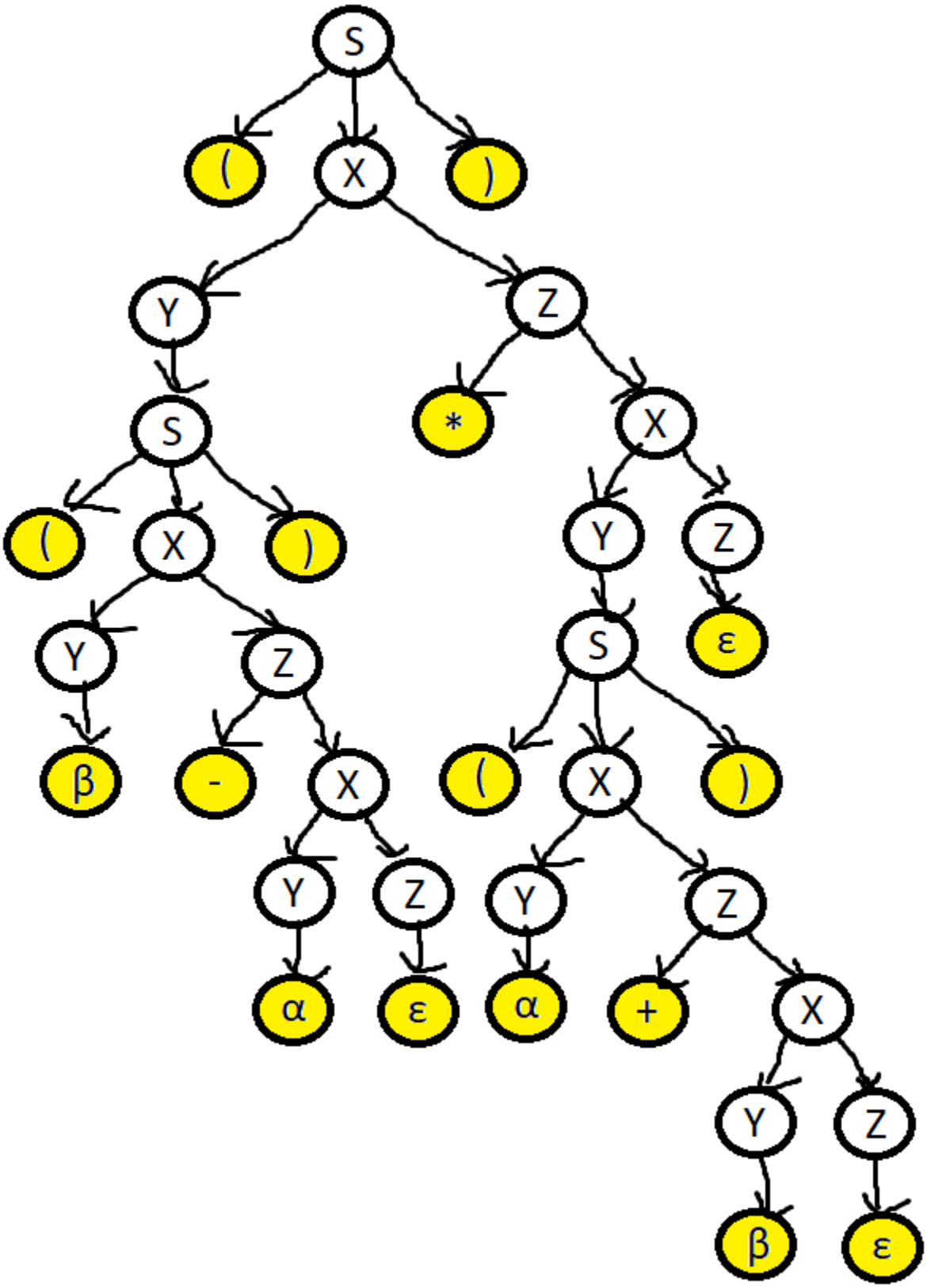
- Η τρίτη στήλη αντιστοιχεί στο κελί του συντακτικού πίνακα που ψάχνουμε έναν κανόνα.
- Η τέταρτη στήλη αντιστοιχεί στον κανόνα αντικατάστασης, εάν και εφόσον βρεθεί κάποιος στο κελί.

Στο τέλος, εμφανίζεται το σχετικό μήνυμα, ανάλογα με το αν αναγνωρίστηκε η έκφραση ή όχι.

Αν αναγνωριστεί, εκτυπώνεται το συντακτικό δέντρο σε προ-,ενδο- και μεταδιάταξη:



Το δέντρο παρουσιάζεται και γραφικά παρακάτω:


S
(
X
)
Y
Z
S
*
X
(
X
)
Y
Z
Y
Z
S
ε
β
-
X
(
X
)
Y
Z
Y
Z
α
ε
α
+
X
Y
Z
β
ε

Μη αναγνώριση της έκφρασης λόγω μη εύρεσης κανόνα:

```
Stack:                  Input:                  Table element:          Production:
$S                      α-β$                    M(S,α)

There is not any replacement rule on table index: [1][3] so the expression is not beeing recognized;
```

Μη αναγνώριση της έκφρασης λόγω μη κενής στοίβας στο τέλος:

```
Give an expression that contains only the following terminal characters: (, ), α, β, *, -, +
(

Stack:                  Input:                  Table element:          Production:
$S                      ($                      M(S,()                  S-->(X)
$)X(                    ($
$)X                     $

Stack is not empty, so the expression is not beeing recognized
```

Σκεπτικό

- Στην αρχή, γίνεται αρχικοποίηση ενός δυσδιάστατου πίνακα τύπου string, ο οποίος έχει την ίδια δομή με τον συντακτικό μας πίνακα:

	(	)	α	β	*	-	+	$
S	S→(X)							
X	X→YZ		X→YZ	X→YZ				
Y	Y→S		Y→α	Y→β				
Z		Z→ε			Z→*X	Z→-X	Z→+X	

```
wstring syntax_table[5][8] = {{L"",L"(",L")",L"α",L"β",L"*",L"-",L"+"}, /
                              {L"S",L")X("},                            /
                              {L"X",L"ZY",L"",L"ZY",L"ZY"},
                              {L"Y",L"S",L"",L"α",L"β"},
                              {L"Z",L"",L"ε",L"",L"",L"X*",L"X-",L"X+"}};
```

Τα κελιά του πίνακα που δεν έχουν κάποιο string, αντιστοιχούν στο κενό string “”.(Για τα περιεχόμενα γίνεται επεξήγηση παρακάτω)

- Χρησιμοποιούμε δύο βασικά string, τα οποία μεταβάλλονται κατά τη διάρκεια του προγράμματος(σε κάθε βήμα). Κάθε χαρακτήρας του string stack από τα αριστερά στα δεξιά, εκφράζει το περιεχόμενο της στοίβας. Συνεπώς, ο δεξιότερος χαρακτήρας εκφράζει το περιεχόμενο στην κορυφή της στοίβας:

```
wstring stack = L"$S"; //At the
beginning, stack has only $ and
S(starting character) characters
```

^^Αρχικοποίηση της string μεταβλητής stack κατά την εκκίνηση.

Το string input περιέχει σε κάθε βήμα το υπόλοιπο

Της εισόδου που πρέπει να διαβαστεί:

```
wstring input; //a string variable to
store user's input. We are going to edit
this string along the procedure, in order
to show the remaining input characters.
```

- Κατά την εκτέλεση του κύριου κομματιού του προγράμματος χρησιμοποιούνται επίσης οι μεταβλητές:

```
wstring stack_top, input_start;
//strings to store stack's top and
input's first characters
```

Οι οποίες σε κάθε βήμα, περιέχουν τον χαρακτήρα στην κορυφή της στοίβας και τον πρώτο χαρακτήρα εισόδου αντίστοιχα:

```
while (true) //a loop that breaks only when the expression has been recognized or not
{
    stack_top = stack[stack.length()-1]; //Every time we repeat, we are storing the la
    input_start = input[0]; //Every time we repeat, we are storing the first input cha
```

- Έχουμε επιλέξει στον συντακτικό πίνακα του προγράμματος να μην συμπεριλάβουμε την στήλη του ‘’$’’ επειδή είναι κενή και δεν θα είχε κάποια χρησιμότητα.

- Έχουμε επιλέξει τα περιεχόμενα του συντακτικού πίνακα του προγράμματος, να αποτελούν τα δεξιά μέλη των αντίστοιχων κανόνων αντικατάστασης. Οι χαρακτήρες τους βρίσκονται σε αντίστροφη σειρά, με στόχο να γίνεται άμεση αντικατάσταση του συμβόλου στην κορυφή της στοίβας με το string του πίνακα. Λόγω αυτού, γίνεται ανάποδη εκτύπωση των χαρακτήρων του κανόνα, όταν πρέπει να δείξουμε ποιος κανόνας αντικατάστασης εφαρμόστηκε σε ένα βήμα:

```
stack.erase(stack.end() - 1); //erase
string stack last not terminal symbol
```

^^Διαγραφή του κορυφαίου χαρακτήρα

```
if (syntax_table[i][j] != L"ε") //if the rule we got is not ε (if it is, we are not adding anything in string stack)

stack.append(syntax_table[i][j]); //append the new characters of the corresponding rule in string stack
```

^^Εάν και εφόσον δεν πέσουμε στον κανόνα αντικατάστασης με κενό, προσθέτουμε στο τέλος του string stack το string του πίνακα.

```
wcout << stack_top << "-->";

for (int k = syntax_table[i][j].length() - 1; k >= 0; k--)
    wcout << syntax_table[i][j][k]; //print in opposite order the characters of replacement rule
```

^^Εκτύπωση των χαρακτήρων του δεξιού μέλους του κανόνα που βρίσκεται στο κελί, με ανάποδη σειρά

- Έχει χρησιμοποιηθεί header αρχείο, το οποίο περιέχει μία δομή κλάσης για τριαδικά δέντρα, η οποία θα μας βοηθήσει να φτιάξουμε το συντακτικό δέντρο.

Ή κλάση περιέχει:

1.Την ιδιότητα τύπου `wchar_t` με όνομα key που αναπαριστά τον χαρακτήρα του κόμβου.

2.Τις ιδιότητες τύπου `ThreenodeTree*` με ονόματα `leftnode,middlenode,rightnode` που αναπαριστούν δείκτες στον αριστερό, μεσαίο και δεξί κόμβο παιδί αντίστοιχα, ενός κόμβου.

3.Τις μεθόδους τύπου `void` με ονόματα `Preorder_Output,Inorder_Output` και `Postorder_Output` που εκτυπώνουν τα περιεχόμενα των κόμβων του δέντρου σε προδιάταξη, ενδοδιάταξη και μεταδιάταξη αντίστοιχα.

4. Την μέθοδο τύπου `static ThreenodeTree*` με όνομα `searchNewrootInpreorder` που επιστρέφει έναν δείκτη στον κόμβο που θα αναλυθεί στη συνέχεια. Ο κόμβος που αναλύεται έχει σαν κλειδί το μη τερματικό σύμβολο στην κορυφή της στοίβας, και αποκτάει για παιδιά, τους χαρακτήρες του κανόνα αντικατάστασης.

Περεταίρω εξηγήσεις των μεθόδων γίνονται στο κομμάτι του αλγόριθμου πιο κάτω.

- Στο κύριο πρόγραμμα, για την δημιουργία του δέντρου, γίνεται αρχειοθέτηση ενός αρχικού pointer “tree” που δείχνει στον κόμβο ρίζα του δέντρου, ενός pointer “root” που δείχνει πάντα στον κόμβο που θα αναλυθεί σε ένα βήμα, και των pointer “rightnd”,”middlend” και “leftnd” που δείχνουν στα παιδιά του root σε κάθε βήμα. Ο tree παραμένει σταθερός κατά τη διάρκεια του προγράμματος, ενώ ο root και οι υπόλοιποι αλλάζουν σε κάθε βήμα:

```
ThreenodeTree* tree = new ThreenodeTree ('S');
//make the first tree node, that has as key the
starting character
```

^^Αρχειοθέτηση του tree

```
ThreenodeTree* root = tree; //root pointer is
being used in order to point at the node we are
going to expand
```

```
ThreenodeTree* rightnd = 0;
ThreenodeTree* middlend = 0;
ThreenodeTree* leftnd = 0; //Children node we are
going to add at each expansion of a root node
```

^^Αρχική αρχειοθέτηση του root και των υπόλοιπων

```
if (regex_match(stack_top, wregex(L"[SXYZ]")) && input_start!=L"$") //if stack_top
{
    root = ThreenodeTree::searchNewrootInpreorder(tree, stack[stack.length() - 1]);
```

^^Αν πρέπει να γίνει αντικατάσταση μη τερματικού συμβόλου, ψάχνουμε τον νέο κόμβο προς ανάλυση. Ο root αλλάζει.

```
leftnd = new ThreenodeTree(syntax_table[i][j][1]);
root->leftnode = leftnd; //Add a left child node in

rightnd = new ThreenodeTree(syntax_table[i][j][0]);
root->rightnode = rightnd; //Add a right child node
break;
```

^^Προσθήκη παιδιών στον root.Τα παιδιά από αριστερά προς τα δεξιά, έχουν για κλειδιά τους χαρακτήρες του κατάλληλου κανόνα από τα δεξιά προς τα αριστερά.

```
wcout << "Syntax tree output in preorder: ";
tree->Preorder_Output();
wcout << "\nSyntax tree output in inorder: ";
tree->Inorder_Output();
wcout << "\nSyntax tree output in postorder: ";
tree->Postorder_Output();
break; //end of procedure
```

^^Εκτύπωση του δέντρου ξεκινόντας από τον δείκτη της ρίζας, στο τέλος του προγράμματος

Αλγόριθμοι

- Αλγόριθμοι των μεθόδων του συντακτικού δέντρου:

i)Αλγόριθμος Αναγωγικής μεθόδου εκτύπωσης

```
Preorder_Output:
```

1.Ξεκίνα από την ρίζα

2.Αν υπάρχει ο κόμβος:

-Εκτύπωσε το κλειδί

-Εκτέλεσε τον αλγόριθμο για το αριστερό υποδέντρο.

-Εκτέλεσε τον αλγόριθμο για το μεσαίο υποδέντρο.

-Εκτέλεσε τον αλγόριθμο για το δεξί υποδέντρο.

Αλλιώς, μην κάνεις τίποτα.

ii) Αλγόριθμος Αναγωγικής μεθόδου εκτύπωσης `Inorder_Output:`

1.Ξεκίνα από την ρίζα

2.Αν υπάρχει ο κόμβος:

-Εκτέλεσε τον αλγόριθμο για το αριστερό υποδέντρο.

-Εκτύπωσε το κλειδί

-Εκτέλεσε τον αλγόριθμο για το μεσαίο υποδέντρο.

-Εκτέλεσε τον αλγόριθμο για το δεξί υποδέντρο.

Αλλιώς, μην κάνεις τίποτα.

iii)Αλγόριθμος Αναγωγικής μεθόδου εκτύπωσης

`Postorder_Output:`

1.Ξεκίνα από την ρίζα

2.Αν υπάρχει ο κόμβος:

-Εκτέλεσε τον αλγόριθμο για το αριστερό υποδέντρο.

-Εκτέλεσε τον αλγόριθμο για το μεσαίο υποδέντρο.

-Εκτέλεσε τον αλγόριθμο για το δεξί υποδέντρο.

-Εκτύπωσε το κλειδί

Αλλιώς, μην κάνεις τίποτα.

iv)Αλγόριθμος αναγωγικής μεθόδου αναζήτησης κόμβου root με κλειδί κ.

(υπενθυμίζεται ότι root είναι ο δείκτης στον κόμβο με κλειδί το μη τερματικό σύμβολο που αναλύεται):

1.Ξεκίνα από την ρίζα

2.Αν υπάρχει ο κόμβος:

-Αν ο κόμβος έχει το κλειδί κ προς αναζήτηση και δεν έχει παιδιά, τότε είναι ο κόμβος προς ανάλυση οπότε επέστρεψε αυτόν.

-Εκτέλεσε τον αλγόριθμο για το αριστερό υποδέντρο, και επέστρεψε το αποτέλεσμα στο p1. Αν ο p1 δεν είναι 0 επέστρεψε p1,αλλιώς συνέχισε.

-Εκτέλεσε τον αλγόριθμο για το μεσαίο υποδέντρο, και επέστρεψε το αποτέλεσμα στο p2. Αν ο p2 δεν είναι 0 επέστρεψε p2,αλλιώς συνέχισε.

-Εκτέλεσε τον αλγόριθμο για το δεξιό υποδέντρο, και επέστρεψε το αποτέλεσμα στο p3. Αν ο p3 δεν είναι 0 επέστρεψε p3,αλλιώς συνέχισε.

-Αν p1,p2,p3 επιστρέψουν 0, τότε επέστρεψε 0.

Αλλιώς, επέστρεψε 0.

- Αλγόριθμος για την δημιουργία του αναλυτή topdown και κατασκευής του συντακτικού δέντρου:

1.Θέσε `stack = "$S"` και πρόσθεσε στο τέλος του input το ‘’$’’.

2.Θέσε root=tree (υπενθυμίζεται ότι tree είναι ο δείκτης στον κόμβο ρίζα με κλειδί το ‘’S’’).

3.Θέσε `stack_top = stack[stack.length()-1]`

(ο τελευταίος χαρακτήρας του stack(κορυφή της στοίβας))

4.Θέσε `input_start = input[0]`

(ο πρώτος χαρακτήρας του input)

5.Τρέξε την αναγωγική μέθοδο αναζήτησης `κόμβου root με κλειδί stack_top` και θέσε στον root το αποτέλεσμα

6.

i)Αν το stack_top είναι μη τερματικό σύμβολο και το input_start δεν είναι ‘’$’’, θέσε i τη γραμμή του συντακτικού πίνακα που βρίσκεται ο χαρακτήρας stack_top και j τη στήλη που βρίσκεται ο χαρακτήρας input_start και:

-Αν υπάρχει κανόνας στο κελί ij:

-Ανάλογα με το πλήθος των χαρακτήρων του κανόνα, βάλε τόσα σε πλήθος κλαδιά στον root.Το κάθε κλαδί έχει έναν χαρακτήρα του κανόνα.

-Σβήσε τον τελευταίο χαρακτήρα του stack.

-Αν ο κανόνας δεν είναι ο ‘’ε’’ πρόσθεσε το string του κανόνα στο τέλος του stack. Πήγαινε πίσω στο βήμα 3.

-Αλλιώς, ο κανόνας είναι ο ‘’ε’’ οπότε μην προσθέσεις κάτι στο τέλος. Πήγαινε πίσω στο βήμα 3.

-Αλλιώς, δεν υπάρχει κανόνας στο κελί, οπότε η έκφραση δεν αναγνωρίζεται και τερματίζεται ο αλγόριθμος.

ii)Αλλιώς, αν το `stack_top` δεν είναι ‘’$’’ και το input_start δεν είναι ‘’$’’, τότε:

`stack_top` = input_start = κάποιο τερματικό σύμβολο, οπότε σβήσε τον τελευταίο και τον πρώτο χαρακτήρα των stack και input αντίστοιχα.

Πήγαινε πίσω στο βήμα 3.

iii)Αλλιώς, αν `stack_top!=''$''` και

input_start =''$'', το input διαβάστηκε και η στοίβα είναι μη κενή, άρα η έκφραση δεν αναγνωρίζεται και ο αλγόριθμος τερματίζεται.

iv)Αλλιώς, αν `stack_top=''$''` και

input_start !=''$'', το input δεν διαβάστηκε και η στοίβα είναι κενή, άρα η έκφραση δεν αναγνωρίζεται και ο αλγόριθμος τερματίζεται.

v) Αλλιώς, stack_top = input_start = ''$'', το input διαβάστηκε και η στοίβα είναι κενή, άρα η έκφραση αναγνωρίζεται και ο αλγόριθμος τερματίζεται.