ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ - ΕΡΓΑΣΙΑ ΕΤΟΥΣ 2022-2023

Ξύδης Τριαντάφυλλος- Π21120

Κοντοπούλου Δέσποινα-Π21066

Ημερομηνία παράδοσης: 17/1/2023

Πίνακας Περιεχόμενων

1. 1η Άσκηση

1.1 Εκφώνηση Άσκησης

Σχεδιάστε και υλοποιήστε μια γεννήτρια συμβολοσειρών για την παρακάτω γραμματική:

<Z>::=(<K>)

<K>::=<G><M>

<G>::=ν|<Z>

<M>::= -<K>|+<K>|ε, όπου ε η κενή συμβολοσειρά

Λάβετε μέριμνα ώστε η διαδικασία να τερματίζεται οπωσδήποτε. Το πρόγραμμά σας θα πρέπει να τυπώνει τα βήματα της παραγωγής. Αποδεκτές γλώσσες προγραμματισμού:

C/C++

1.2 Τεκμηρίωση Επίλυσης

Από την δοθείσα γραμματική BNF συμπεραίνουμε για τα σύμβολα:
Μη-τερματικά: Z, K, G, M
Τερματικά: (,), ν, -, +, ε

Στις αρχικοποιήσεις του προγράμματος υπάρχουν:

- character arrays str και str2 μεγέθους 500 τα οποίο θα περιέχουν αντίστοιχα το πρώτο και το δεύτερο μέρος της συμβολοσειράς.
- Character arrays tmp1, tmp2, tmp3, tmp4, tmp5 μεγέθους 2 τα οποία περιέχουν τους τερματικούς χαρακτήρες. Θα χρησιμεύσουν με την συνάρτηση strcat.
- Void produceZ(), produceK(), produceG(), produceM() για την αντικατάσταση των μη τερματικών συμβόλων.
- Int main() η κύρια συνάρτηση του προγράμματος.

Όσον άφορα τις λειτουργίες των συναρτήσεων:

main()

Αρχικά δημιουργούμε μία γεννήτρια τυχαίων αριθμών με την βοήθεια της συνάρτησης srand(). Έπειτα εκτυπώνουμε στην κονσόλα το αρχικό σύμβολο Z και καλείται η συνάρτηση produceZ() ώστε να ξεκινήσει η διαδικασία της αντικατάστασης. Μόλις εκείνη η διαδικασία τελειώσει, εκτυπώνουμε την ζητούμενη συμβολοσειρά και το πρόγραμμα τελειώνει.

produceZ()

Εκτυπώνουμε τον κανόνα που θα χρησιμοποιηθεί. Προσθέτουμε στην τελική συμβολοσειρά το τερματικό σύμβολο «(» και εκτυπώνουμε την συμβολοσειρά όπως είναι μέχρι στιγμής. Καλούμε την produceK() η οποία θα συνεχίσει την ανάλυση και έπειτα προσθέτουμε στην τελική συμβολοσειρά το «)».

produceK()

Εκτυπώνουμε τον κανόνα που θα χρησιμοποιηθεί. Εκτυπώνουμε τους μη τερματικούς χαρακτήρες που έχουμε ήδη, το GM που αντικαθιστά το K, καθώς και τυχόν χαρακτήρες που περιέχει το str2. Έπειτα καλούμε τις produceG(), produceM().

produceG()

Επειδή έχουμε δύο ενδεχόμενες αναλύσεις του G, θα διαλέξουμε τυχαία μια εξ αυτών. Εκτυπώνουμε τον κανόνα που θα χρησιμοποιηθεί.
Αν επιλεγεί ο κανόνας G→ν τότε προσθέτω το ν στο str, εκτυπώνω την συμβολοσειρά που έχω ήδη, ένα «M)» στο τέλος (πάντα το ν συνοδεύεται από το M σε αυτό το στάδιο) και το str2.
Αν επιλεγεί το G→Z εκτυπώνω το str και τα σύμβολα «ZM)» (το M ακολουθεί πάντα το Z σε αυτό το στάδιο) .Τέλος εκτυπώνουμε την str2, προσθέτουμε σε αυτήν τα σύμβολα «M)» για τις υπόλοιπες εκτυπώσεις της ανάλυσης της συμβολοσειράς και καλούμε την produceZ().

produceM()

Επειδή έχουμε τρεις ενδεχόμενες λύσεις, και επειδή θέλουμε το πρόγραμμα να τερματίζει, επιλέγουμε έναν random αριθμό μέχρι το 4.
Αν είναι το 0, επιλέγουμε τον κανόνα M→ -K, τον εκτυπώνουμε και προσθέτουμε το «-» στην συμβολοσειρά. Έπειτα εκτυπώνουμε την συμβολοσειρά που έχουμε ως τώρα και καλούμε την produceK().
Αν είναι το 1, επιλέγουμε τον κανόνα M→+K, και πράττουμε ομοίως.
Για αριθμούς από το 2 έως το 4 επιλέγουμε τον κανόνα M→ε και τον εκτυπώνουμε. Εκτυπώνουμε το str και έπειτα αν το str2 είναι κενό εκτυπώνουμε την τελική παρένθεση. Διαφορετικά διαγράφουμε το M που αναλύσαμε από το str2 και εκτυπώνουμε τα M που μένουν, εάν υπάρχουν.

1.3 Επίδειξη Λύσης

1ο παράδειγμα

```
C:\Users\desp\Documents\Despoina\UniPi\3rd_Semester\Compilers>a.exe

Z
Z-->(K) (K)
K-->GM  (GM)
G-->Z   (ZM)
Z-->(K) ((K)M)
K-->GM  ((GM)M)
G-->v   ((vM)M)
M-->-K  ((v-K)M)
K-->GM  ((v-GM)M)
G-->v   ((v-vM)M)
M-->e   ((v-v)M)
M-->e   ((v-v))
--------

((v-v))
```

2ο παράδειγμα

```
C:\Users\desp\Documents\Despoina\UniPi\3rd_Semester\Compilers>a.exe

Z
Z-->(K) (K)
K-->GM  (GM)
G-->Z   (ZM)
Z-->(K) ((K)M)
K-->GM  ((GM)M)
G-->v   ((vM)M)
M-->e   ((v)M)
M-->e   ((v))
--------

((v))
```

3ο παράδειγμα

```
C:\Users\desp\Documents\Despoina\UniPi\3rd_Semester\Compilers>a.exe

Z
Z-->(K) (K)
K-->GM  (GM)
G-->v   (vM)
M-->+K  (v+K)
K-->GM  (v+GM)
G-->Z   (v+ZM)
Z-->(K) (v+(K)M)
K-->GM  (v+(GM)M)
G-->Z   (v+(ZM)M)
Z-->(K) (v+((K)M)M)
K-->GM  (v+((GM)M)M)
G-->v   (v+((vM)M)M)
M-->+K  (v+((v+K)M)M)
K-->GM  (v+((v+GM)M)M)
G-->v   (v+((v+vM)M)M)
M-->e   (v+((v+v)M)M)
M-->e   (v+((v+v))M)
M-->e   (v+((v+v)))
--------

(v+((v+v)))
```

2. 2η Άσκηση

2.1 Εκφώνηση Άσκησης

Δίνεται η γραμματική:
G→ (M)
M→ YZ
Y→ a | b | G
Z→ *M | -M | +M| ε, όπου ε η κενή συμβολοσειρά
Σχεδιάστε και υλοποιήστε συντακτικό αναλυτή top-down που αναγνωρίζει την εκάστοτε συμβολοσειρά ή απαντά αρνητικά ως προς τη συντακτική της ορθότητα. Να επιστρέφεται το σχετικό δέντρο και να εκτυπώνεται. Να γίνει επίδειξη για την έκφραση ((a-b)*(a+b)).
Αποδεκτές γλώσσες προγραμματισμού: C/C++

2.2 Τεκμηρίωση επίλυσης

Αρχικοποιούμε τα εξής:

- Char array input[100]:
  Το char array στο οποίο ο θα εισαχθεί η συμβολοσειρά για αναγνώριση.
- Int i=0:
  Μια μεταβλητή int που θα χρησιμοποιήσουμε ως counter για την ανάγνωση του array input.
- Int G(), M(), Y(), Z():
  Μέθοδοι που επιβεβαιώνουν αν εφαρμόσθηκε η μετατροπή των μη-τερματικών συμβόλων G, M, Y, Z, εκτυπώνουν το δένδρο και ενημερώνουν για τυχών έλλειψη συμβόλων. Επιστρέφουν 0 σε περίπτωση αποδεκτής συμβολοσειράς, ειδάλλως επιστρέφουν 1.

  Όσον αφορά τις λειτουργίες των μεθόδων:

  G()

  Εκτυπώνει το G και τα παιδιά του. Αν ο χαρακτήρας i είναι «(» τότε καλείται η μέθοδος M για τον επόμενο χαρακτήρα. Έπειτα εφόσον υπάρχει το σύμβολο «)» επιστρέφεται 0. Σε κάθε άλλη περίπτωση εκτυπώνεται μήνυμα λάθους και επιστρέφεται 1.

M()

Καλεί τις μεθόδους Y και Z για αναγνώριση της διακπεραίωσης του κάθε κανόνα. Εάν δεν έχουν εφαρμοσθεί επιστρέφει 1.

Y()

Εκτυπώνει την συνέχεια του δένδρου. Εάν ο χαρακτήρας i είναι a ή b εκτυπώνει το αντίστοιχο παιδί του Y, το i αυξάνεται κατά 1 για τον επόμενο κανόνα και επιστρέφεται 0. Αν ο χαρακτήρας i είναι «(» τότε καλείται η μέθοδος G για περαιτέρω έλεγχο. Σε κάθε άλλη περίπτωση εκτυπώνεται μήνυμα λάθους και επιστρέφεται 1.

Z()

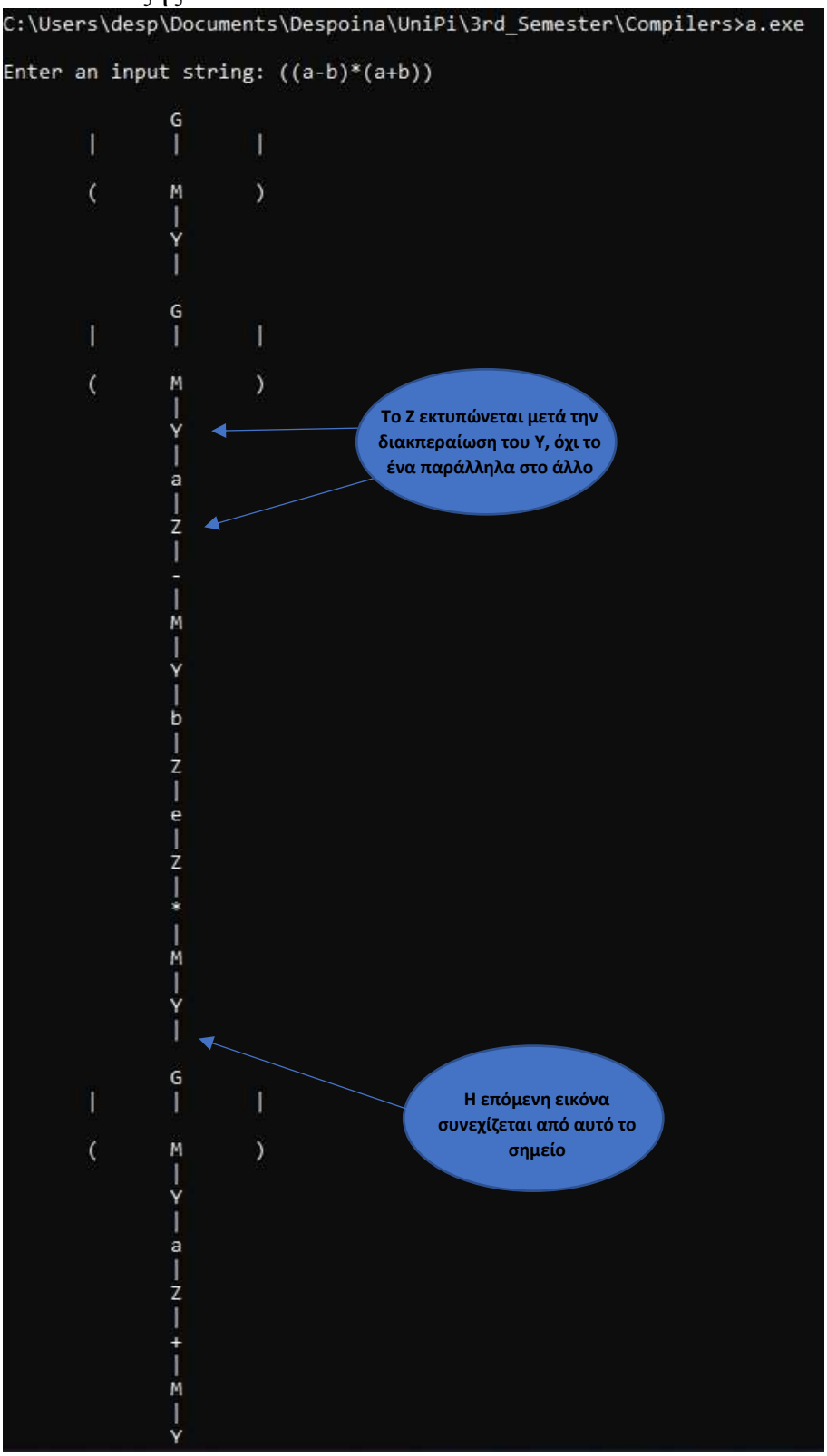
Εκτυπώνει την συνέχεια του δένδρου. Εάν ο χαρακτήρας i είναι *, - ή + εκτυπώνει το αντίστοιχο παιδί του Z,το i αυξάνεται κατά 1 για τον επόμενο κανόνα και καλείται η μέθοδος M για περαιτέρω έλεγχο. Διαφορετικά αναγνωρίζεται το κενό και εκτυπώνεται. Επιστρέφεται 0.

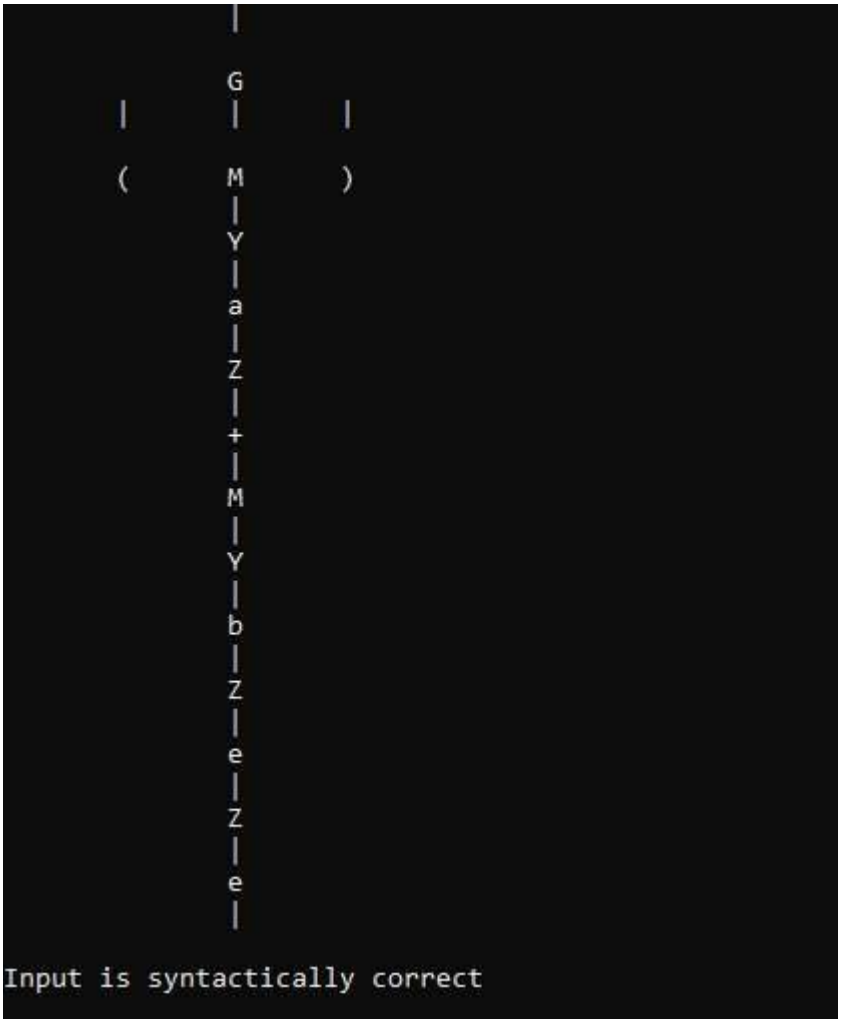
main()

Ζητείται από τον χρήστη να εισάγει την συμβολοσειρά για αναγνώριση, η οποία καταχωρείται στο input. Εάν η G και όλες οι συναρτήσεις που καλούνται με την σειρά τους δεν επιστρέψουν κάποιο λάθος, συγκρίνεται το μήκος του input με το counter i. Αν ταυτίζονται τότε έχουμε φτάσει στο τέλος του string και η συμβολοσειρά είναι αποδεκτή. Διαφορετικά, το i είναι μεγαλύτερο από το μήκος του input και υπάρχουν επιπλέον χαρακτήρες που δεν αναγνωρίστηκαν από κάποιον κανόνα.

2.3 Επίδειξη Λύσης

Η εκτέλεση του προγράμματος με είσοδο την έκφραση ((a-b)*(a+b)) εκτυπώνει τα εξής:




G
M
Y
a
Z
+
M
Y
b
Z
e
Z
e
Input is syntactically correct

3. 3η Άσκηση

3.1 Εκφώνηση Άσκησης

Σε ένα υποσύνολο φυσικής γλώσσας, τα ονόματα σημείων ορίζονται ως η παράθεση ενός μόνο συμβόλου, τα ονόματα ευθείων ορίζονται ως η παράθεση δύο συμβόλων, τα ονόματα τριγώνων ορίζονται ως η παράθεση τριών συμβόλων, κ.ο.κ, έως και την περίπτωση οκταγώνων. Δεν επιτρέπονται επαναλήψεις συμβόλων. Σχεδιάστε και υλοποιήστε πρόγραμμα Flex που θα αναλύει προτάσεις της μορφής «τρίγωνο BCD», «τετράγωνο BCDA», κ.ο.κ. και θα αποδέχεται μόνο τους σωστούς ορισμούς. Παραδείγματα εσφαλμένων ορισμών είναι «τετράγωνο AB», «τρίγωνο AAD», «γωνία BC». Αποδεκτές γλώσσες προγραμματισμού: FLEX (σε συνδυασμό με C/C++).

3.2 Τεκμηρίωση επίλυσης

Αρχικά εισάγουμε κώδικα σε C (definitions). Στις αρχικοποιήσεις έχουμε τα εξής:

- Char array str1[30] για την αποθήκευση του σχήματος (point, line, triangle κοκ)
- Char array str2[30] για την αποθήκευση των συμβόλων που ορίζουν το σχήμα
- Int check_unique(char* s) για τον έλεγχο της μοναδικότητας των συμβολών του str2
- Int check(char* s) για τον έλεγχο αντιστοιχίας μεταξύ των str1 και str2

Όσον αφορά τις μεθόδους:

check_unique(char* s)

Με την χρήση 2 for loops συγκρίνουμε όλους τους χαρακτήρες του array μεταξύ τους. Εάν βρεθούν **ίδιοι χαρακτήρες θα επιστραφεί 0**, ειδάλλως θα επιστραφεί 1

check(char* s)

Ορίζουμε ένα char pointer token ώστε με την βοηθεια της strtok να «σπάσουμε» σε tokens το string που εισάγει ο χρήστης και θα αρχικοποιήσουμε τα str1 και str2.

Με την βοήθεια της strcmp συγκρίνουμε το str1 με το string ‘point’. Εάν ταυτίζονται, τότε ελέγχουμε την μοναδικότητα του str2. Εκτυπώνουμε στον χρήστη το μήνυμα επιτυχίας ή αποτυχίας ανάλογα. Ομοίως πράττουμε και για τα υπόλοιπα σχήματα.

Έπειτα έχουμε τους κανόνες σε flex. Εάν διαβάσω το string «point» συνοδευόμενο από κενό και έπειτα 1 χαρακτήρα από το A έως το Z καλώ την συνάρτηση check για την είσοδο. Ομοίως ελέγχω για 2 χαρακτήρες μετρά την λέξη line, 3 χαρακτήρες μετρά την λέξη triangle κοκ. Εάν δοθεί οτιδήποτε άλλο εκτυπώνω μήνυμα λάθους.

Στην συνάρτηση main ζητώ από τον χρήστη να εισάγει την πρόταση του και την διαβάζω με την χρήση του yylex.

3.3 Επίδειξη Λύσης

```
C:\Users\desp\Documents\Despoina\UniPi\3rd_Semester\Compilers>a.exe

----

FOR FIRST WORD USE: point,line,triangle,square,pentagon,hexagon,heptagon,octagon.

FOR SECOND WORD USE: Any Character Based On First Word.

Press TAB AND ENTER TO TERMINATE.

----
point A
Accepted: point

point AB
Error.

octagon ABCDEFGH
Accepted: octagon

octagon ABCD
Error.

octagon ABCDEFGG
Rejected: repeated symbols
```

4. Βιβλιογραφικές Πηγές

1. "Μεταγλωττιστές" Σύγγραμμα, Μ.Κ. Βίρβου, 2014, Εκδόσεις Βαρβαρήγου, ISBN: 978-960-7996-15-1

2. https://gunet2.cs.unipi.gr/modules/document/file.php/TMB100/FLEX/flex-maual.pdf

3. https://isaaccomputerscience.org/concepts/dsa_toc_bnf?examBoard=all&stage=all

4. https://www.tutorialspoint.com/compiler_design/compiler_design_top_down_parser.htm