ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Πληροφορικής

Εργασία Μαθήματος ***Μεταγλωττιστές***

Αριθμός εργασίας – Τίτλος εργασίας	***Μεταγλωττιστές εργασία 6μήνου 2022-2023***
Ονόματα φοιτητών	Λεωνίδας Μάντζαρης, Αλέξιος Βασιλείου, Ιωάν Κρόιτορ Καταρτζίου, Αλέξιος Ρούτσης
Αρ. Μητρώου αντίστοιχα	Π21090, Π21009, Π21077,Π21145
Ημερομηνία παράδοσης	14/11/2023

Εκφώνηση εργασίας

(ΘΕΜΑ 1) – 3 μονάδες: 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++

(ΘΕΜΑ 2) – 4 μονάδες: 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++

(ΘΕΜΑ 3) – 3 μονάδες: Σε ένα υποσύνολο φυσικής γλώσσας, τα ονόματα σημείων ορίζονται ως η παράθεση ενός μόνο συμβόλου, τα ονόματα ευθειών 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++).

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1 ΘΕΜΑ 1ο ..... **Error! Bookmark not defined.**

1.1 Εισαγωγή ..... **Error! Bookmark not defined.**

1.2 Περιγραφή του προγράμματος ..... **Error! Bookmark not defined.**

2 Βιβλιογραφικές Πηγές 5

Εισαγωγή

Εδώ μπορείτε να περιελάβατε μία γενική εισαγωγή σχετικά με την εργασία, το πρόβλημα που καλείστε να επιλύσετε και τη μεθοδολογία που ακολουθήσατε (γλώσσα/ εργαλεία προγραμματισμού, αλγόριθμοι που υλοποιήσατε κτλ).

Περιγραφή του προγράμματος

Εδώ μπορείτε την περιγραφή της λύσης σας, τη δομή του προγράμματός σας, τον κώδικα κάθε διαδικασίας, μεθόδου ή συνάρτησης, μαζί με αναλυτικά σχόλια.

Σε κάθε ενότητα μπορείτε να περιελάβατε όσες υποενότητες κρίνετε σκόπιμο, (Ακολουθεί παράδειγμα επικεφαλίδων 2^{ου} και 3^{ου} επιπέδου). Π.χ. μία υποενότητα για κάθε διαδικασία ή για κάθε κλάση κτλ.

Επίδειξη της λύσης

Εδώ θα περιλάβετε ενδεικτικές εικόνες από την εκτέλεση του προγράμματος μαζί με ενδεικτικά screenshots και όποια ανάλυση κρίνετε απαραίτητη.

Κάθε εικόνα ή πίνακας θα πρέπει να έχει την ανάλογη επεξήγηση (λεζάντα) και αρίθμηση. Ακολουθούν ενδεικτικά παραδείγματα για την εισαγωγή εικόνας και την εισαγωγή πίνακα.

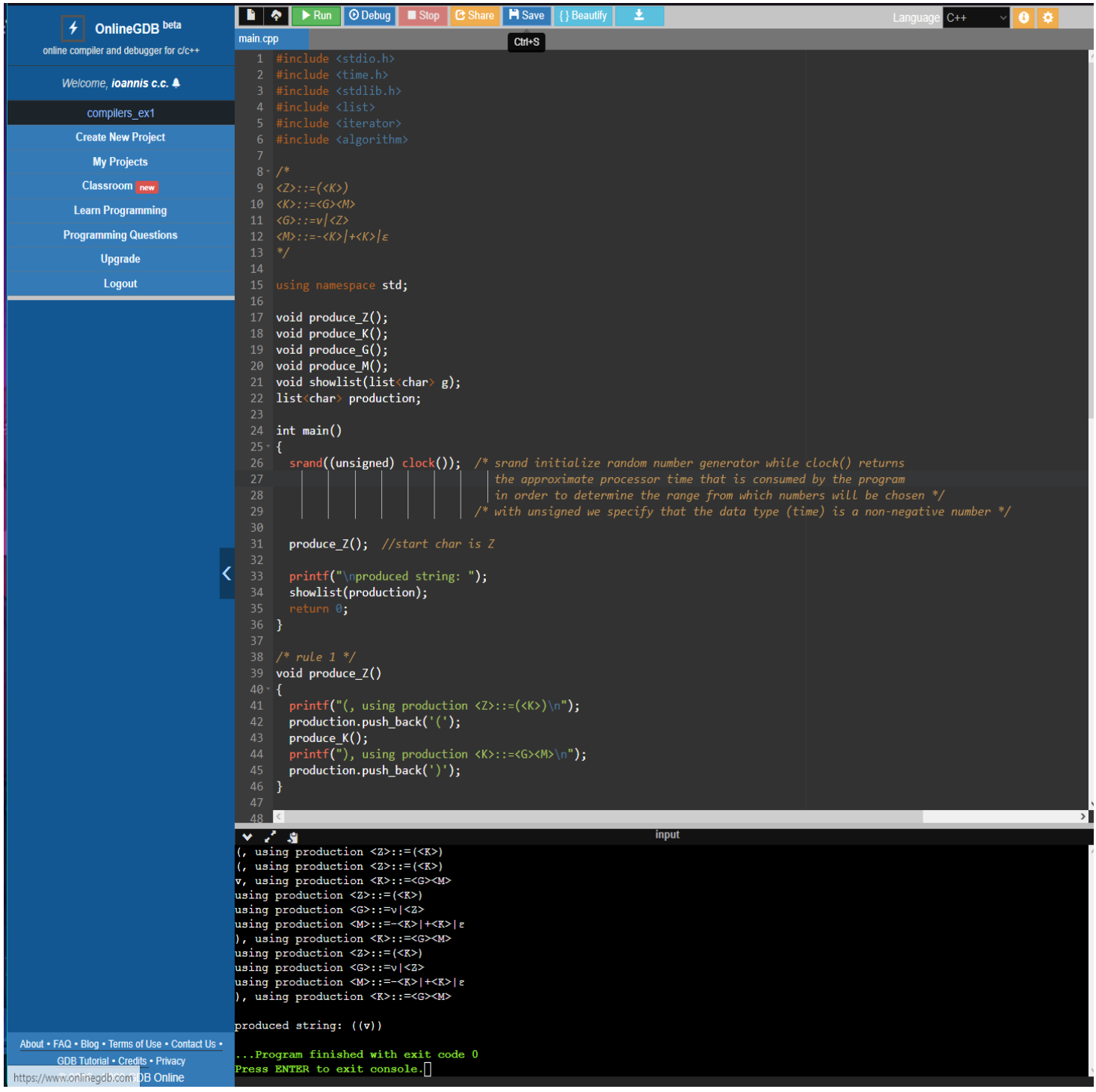
ΕΙΣΑΓΩΓΗ(1)

Παρατίθενται οι κανόνες γραμματικής:
Z→ (K) (1)
K→ GM (2)
G→ v|Z (3)
M→ -K|+K| (4)

ΠΕΡΙΓΡΑΦΗ ΠΡΟΓΡΑΜΜΑΤΟΣ-ΛΥΣΗΣ(2)

Για να απαντήσουμε στην άσκηση πρέπει να κατασκευάσουμε μία γεννήτρια συμβολοσειρών. Έτσι σκεφτήκαμε να κατασκευάσουμε 4 αναδρομικές συναρτήσεις, για κάθε κανόνα παραγωγής αντίστοιχα και ξεκινώντας από το αρχικό σύμβολο (Z) να κατασκευάσουμε μία συμβολοσειρά. Στις περιπτώσεις που είχαμε περισσότερες επιλογές (κανόνες 3 & 4),η επιλογή του κανόνα που θα εφαρμοστεί γίνεται με τυχαίο τρόπο (έχοντας ορίσει μία γεννήτρια τυχαίων θετικών αριθμών με βάση τη clock() συνάρτηση και χρησιμοποιώντας το υπόλοιπο της διαίρεσης με το 2 ή 3 ως κριτήριο επιλογής). Για να τερματίσουμε το πρόγραμμα, λάβαμε υπόψιν πως αυτό θα συμβεί εάν η main() κάνει return, πράγμα το οποίο συμβαίνει όταν όλες η αναδρομικές συναρτήσεις εκτελέσουν τον κώδικά τους (τελειώσει η αναδρομή). Γι’ αυτό στην produce_M() θέσαμε μεγαλύτερη πιθανότητα στην εμφάνιση του ε (κενό) ώστε να επιστρέψει στη main και να τερματίσει. Τέλος, σε κάθε βήμα εμφανίζεται ο κανόνας που χρησιμοποιήθηκε και ο χαρακτήρας που παράγεται (αν υπάρχει).

Επίδειξη προγράμματος



ΕΙΣΑΓΩΓΗ(2)

Παρατίθενται οι κανόνες γραμματικής σε αναλυμένη μορφή:
G→ (M) (1)
M→ YZ (2)
Y→ a (3)
Y→ b (4)
Y→ G (5)
Z→ *M (6)
Z→ -M (7)
Z→+M(8)
Z→ ε (8)

Για να απαντήσουμε στην άσκηση πρέπει να βρω τι είδους LL(k) γραμματική είναι. Θα δοκιμάσω αν είναι LL(1).Για να γίνει αυτό πρέπει :
 πρώτα να κατασκευάσουμε τα σύνολα FIRST και FOLLOW
 να βρούμε τη συνάρτηση LOOKAHEAD και
 να εξετάσουμε αν ισχύει ο κανόνας : (A→α , A→β στη γραμματική που έχουν το ίδιο αριστερό μέλος ισχύει ότι LOOKAHEAD(A→a) ∩ LOOKAHEAD(B→a) ≠ κενού)
 ένας επιπλέον τρόπος να εξετάσουμε αν η γραμματική είναι LL(1), είναι μέσω του συντακτικού πίνακα (Μεταγλωττιστές Μ.Κ.Βίρβου σελ.198-201) ώστε να κατασκευάσουμε συντακτικό αναλυτή που δεν απαιτεί οπισθοδρόμηση.

ΕΠΙΔΕΙΞΗ ΛΥΣΗΣ(2)

Σύνολα FIRST:

Αν α σύμβολο γραμματικής τότε γνωρίζουμε ότι:

1.

1. Αν α τερματικό FIRST(a) = {a}.
2. Αν α = ε τότε FIRST(a) = FIRST(ε) = {ε}.
3. Αν α = μη τερματικό X και X→ B1|B2|...|Bn τότε :

   FIRST(a) = FIRST(X) = FIRST(X) = FIRST(B1) ∪ FIRST(B2) ∪ … ∪ FIRST(Bn)

2. Επίσης γνωρίζουμε ότι αν α = X1 X2 ...Xn τότε :

1. FIRST (α) = FIRST(X1) αν ε ∉ FIRST(X1)
2. αν ε ∈ FIRST(X1) τότε FIRST(α) = FIRST(X1) – {ε} ∈ FIRST(X2...Xn)
3. αν ε ∈ FIRST(X1) & ε ∈ FIRST(X2) & … & ε ∈ FIRST(Xn) τότε :

FIRST (α) = FIRST(X1 X2 ...Xn) ∪ {ε}.

Άρα :

FIRST(G) = { (}
FIRST(M) = { a, b, (}
FIRST(Y) = { a, b, (}
FIRST(Z) = { +, - , * , ε }

Σύνολα FOLLOW:

Έστω γραμματική G = (S, N, T, P) με A E N και $ συμβολίζει τέλος συμβολοσειράς.

Τότε έχουμε:

1. Αν A = S τότε $ ∈ FOLLOW (A)

2. Για κάθε p ∈ P και A να περιλαμβάνεται στο δεξιό μέλος ενός p της μορφής B→ γAδ τότε:

i. αν δ = c όπου c τερματικό τότε FOLLOW (A) = FOLLOW (A) + { c }
ii. αν δ μη-τερματικό τότε FOLLOW (A) = FOLLOW (A) + FIRST { δ } - { ε }
iii. αν δ = ε ή ε προκύπτει από το δ ύστερα από μία σειρά παραγωγών, τότε:
FOLLOW (A) = FOLLOW (A) + FOLLOW (B).

Άρα :

$ ∈ FOLLOW{ G }

FIRST{) } – { ε } ⊆ FOLLOW{ M } => {) } ⊆ FOLLOW{ M }

FIRST{ Z } – { ε } ⊆ FOLLOW(Y) => { +, - , * } ⊆ FOLLOW{ Y }

FOLLOW{ M } ⊆ FOLLOW { Z }

FOLLOW{ Y } ⊆ FOLLOW { G }

FOLLOW{ Z } ⊆ FOLLOW { M }

FOLLOW{ M } ⊆ FOLLOW { Y }

Έτσι θα έχουμε :

FOLLOW{ G } = { $, + , - , * ,) }

FOLLOW{ M } = {) }

FOLLOW{ Y } = { * , + , - ,) }

FOLLOW{ Z } = {) }

Συνάρτηση LOOKAHEAD:

LOOKAHEAD (G→ (M)) = FIRST{ (} = { (}
LOOKAHEAD (M→ YZ) = FIRST{ Y } = { a, b , c }
LOOKAHEAD (Y→ a) = FIRST{ a } = { a } (1)
LOOKAHEAD (Y→ b) = FIRST{ b } = { b } (2)
LOOKAHEAD (Y→ c) = FIRST{ c } = { c } (3)
LOOKAHEAD (Z→ *M) = FIRST{ * } = { * } (4)
LOOKAHEAD (Z→ -M) = FIRST{ - } = { - } (5)
LOOKAHEAD (Z→ +M) = FIRST{ + } = { + } (6)
LOOKAHEAD (Z→ ε) = FOLLOW{ ε } = {) } (7)

(1) ⋂ (2) ⋂ (3) = Ø

(4) ⋂ (5) ⋂ (6)⋂(7) = Ø

Άρα είναι LL(1)

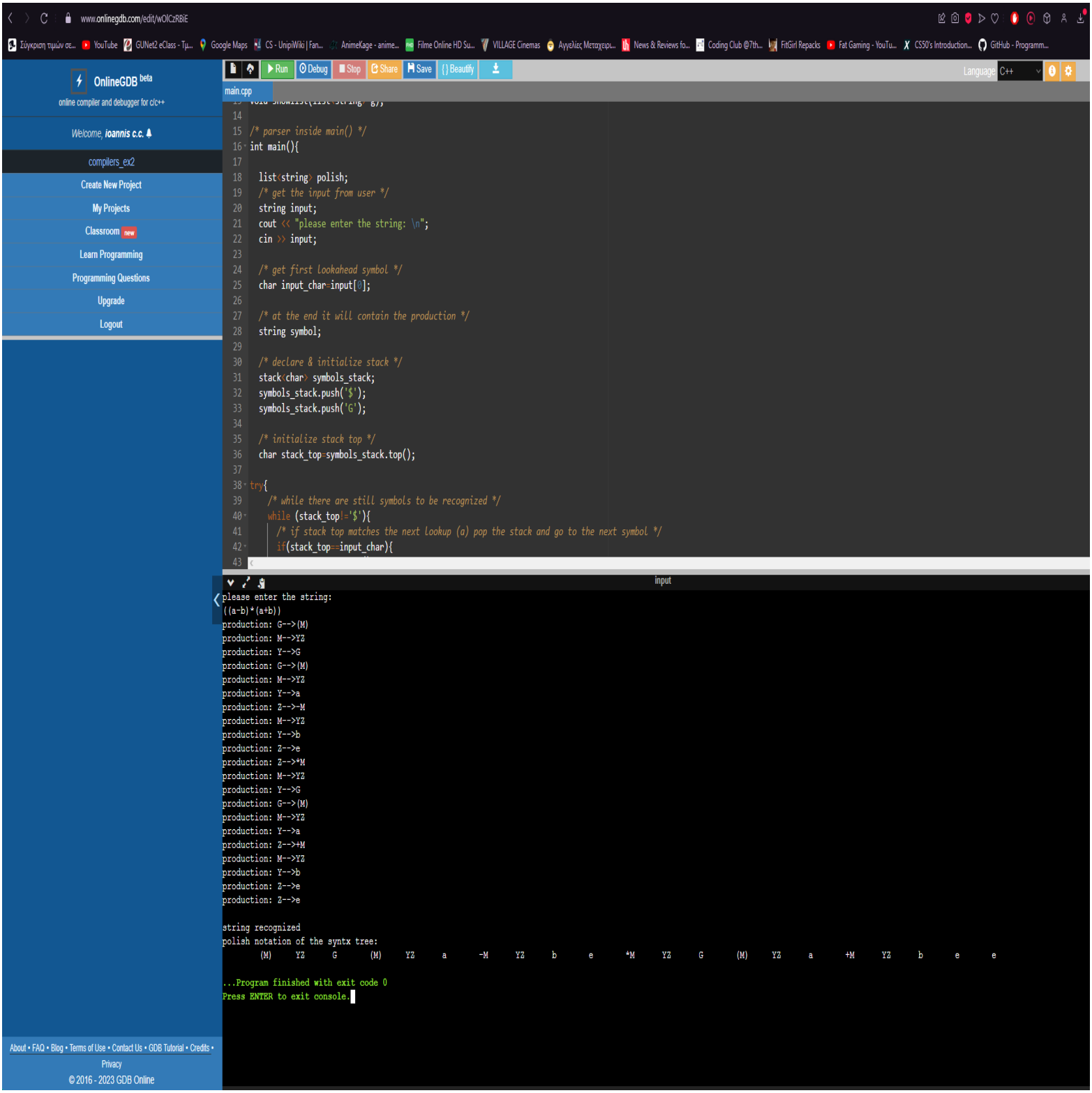
Συντακτικός Πίνακας

V\T	(	)	a	b	c	*	-	+
G	G→(M)							
M	M→YZ		M→YZ	M→YZ				
Y	Y→G		Y→a	Y→b				
Z		Z→ε				Z→*M	Z→-M	Z→+M

ΠΕΡΙΓΡΑΦΗ ΠΡΟΓΡΑΜΜΑΤΟΣ(2)

Προσπαθήσαμε να υλοποιήσουμε έναν top down LL(1) συντακτικό αναλυτή σε C++ και γι' αυτό τον λόγο χρησιμοποιήσαμε στοίβα και συντακτικό πίνακα τον οποίο παραθέτουμε παρακάτω. Αρχικά, ορίζουμε μία συνάρτηση return_substitute_symbol(char stack_char, char lookahead_char) η οποία προσομοιώνει τον συντακτικό πίνακα ελέγχοντας κάθε φορά ποιο σύμβολο θα είναι το επόμενο σύμβολο προς αντικατάσταση. Στη συνέχεια, ορίζουμε εντός της main() μία στοίβα (που αρχικοποιείται με $ και G) μέσω της οποίας εκτελούμε τη συντακτική ανάλυση αφού παίρνοντας κάθε φορά το πρώτο lookahead σύμβολο και εξετάζοντας το stack_top, πραγματοποιούμε την κατάλληλη αλλαγή (pop ή push με βάση τον συντακτικό πίνακα). Τέλος, ο έλεγχος ορθότητας επιτυγχάνεται μέσω ενός try-catch block το οποίο ανιχνεύει ακολουθίες συμβολοσειρών που δεν ανήκουν στην γραμματική χωρισμένο υποπεριπτώσεις είτε επειδή οδηγούμαστε σε μία περίπτωση η οποία δεν ανήκει στον συντακτικό πίνακα (κενό κελί), είτε διότι η συμβολοσειρά που εισαγάγαμε περιέχει ακολουθίες χαρακτήρων που δεν ανήκουν στη γραμματική. Όσον αφορά την εκτύπωση του συντακτικού δέντρου, σκεφτήκαμε να εκτυπώνουμε κάθε φορά την παραγωγή που χρησιμοποιεί στην αναγνώριση και επιπλέον να εκτυπώνει την πολωνική γραφή του (γνώση από μάθημα επιλογής Εφαρμογές Θεωρίας Γραφημάτων σημειώσεις μαθήματος σελ.112) η οποία εμφανίζει σε προδιάταξη τη διάσχιση του δένδρου και με τη βοήθεια των κανόνων παραγωγής της γραμματικής είναι πολύ εύκολο να κατασκευαστεί.

Επίδειξη προγράμματος για τη δοσμένη συμβολοσειρά ((a-b)*(a+b))



ΠΕΡΙΓΡΑΦΗ ΠΡΟΓΡΑΜΜΑΤΟΣ(3)

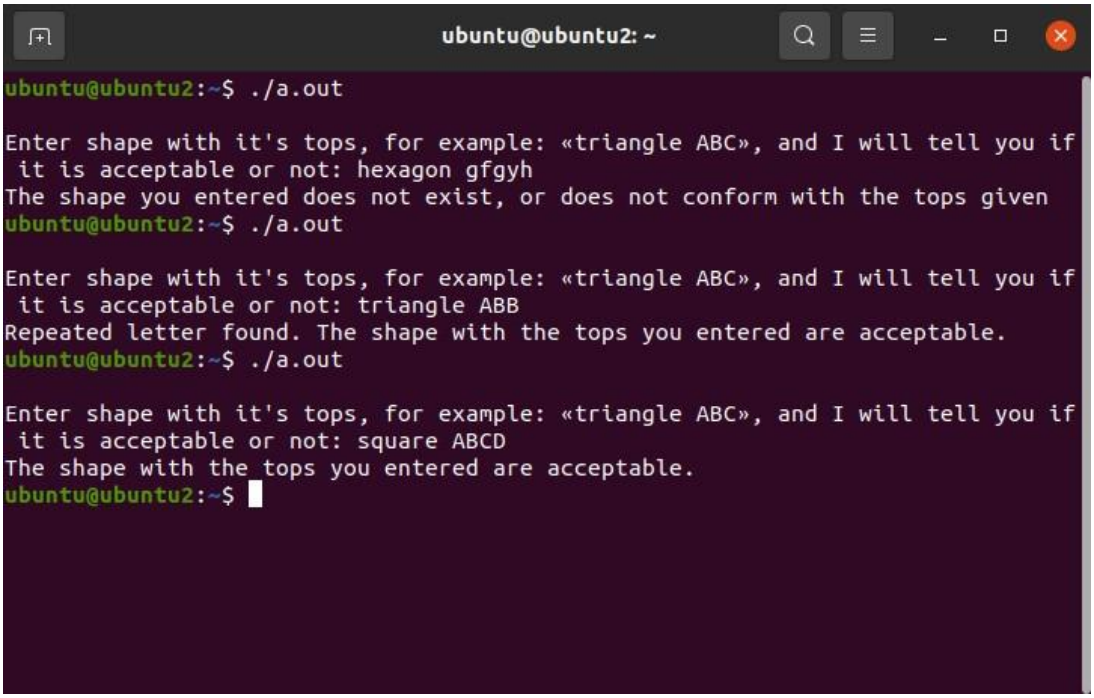
Όλες οι αποδεκτές προτάσεις αποτελούνται από 2 μέρη: γεωμετρική σχήμα (geometric shape: point,line, triangle , square, pentagon, hexagon, heptagon, octagon) και το όνομα που δίνει ο χρήστης στο εκάστοτε γεωμετρικό σχήμα ([A-Z][A-Z][A-Z]…).

• Στο τμήμα των ορισμών έχουν οριστεί οι απαραίτητες βιβλιοθήκες, η μεταβλητή bool acceptableRule η οποία ελέγχει αν το γεωμετρικό σχήμα είναι σωστό και έχει το σωστό αριθμό κορυφών και τέλος υπάρχει ο ορισμός της συνάρτησης findRepeatedTop(char* s) η οποία ψάχνει αν υπάρχουν δύο κορυφές με το ίδιο όνομα

• Στο τμήμα κανόνων το πρόγραμμα αναγνωρίζει τις λέξεις point,line, triangle , square, pentagon, hexagon, heptagon, octagon ως γεωμετρικά σχήματα , ένα κενό χαρακτήρα και ορισμένες επαναλήψεις γραμμάτων ως ονόματα των γεωμετρικών σχημάτων. Επίσης με την (.) αναγνωρίζει όλες τις υπόλοιπες λέξεις. Τέλος με το \n δεν λαμβάνεται input μετά από το enter

• Στο τμήμα κώδικα εμφανίζεται ένα μήνυμα στο χρήστη με ένα παράδειγμα χρήσης και στη συνέχεια τρέχει το πρόγραμμα.

ΕΠΙΔΕΙΞΗ ΛΥΣΗΣ



Βιβλιογραφικές Πηγές

Στο τέλος της εργασίας θα πρέπει ***να περιλάβετε οπωσδήποτε****, όλες τις αναφορές των βιβλιογραφικών πηγών που χρησιμοποιήσατε για τη λύση του προβλήματος (βιβλία, ιστοσελίδες κτλ). Ακολουθεί ενδεικτικό παράδειγμα.*

1. **Menezes, Vandstone.** *Handbook of Applied Cryptography.* New York : CRC PRess, 1996.
2. Dragon Book (2nd ed. pages: 222-231)
3. Μεταγλωττιστές (Μ.Κ. Βίρβου)
4. https://www.geeksforgeeks.org/list-cpp-stl/
5. https://linux.die.net/man/3/srand
6. https://www.geeksforgeeks.org/stack-in-cpp-stl/
7. https://cplusplus.com/reference/stack/stack/