ΣΤΟΙΧΕΙΑ ΠΟΥ ΣΥΜΠΛΗΡΩΝΕΙ Ο ΦΟΙΤΗΤΗΣ / Η ΦΟΙΤΗΤΡΙΑ

|  |  |
| --- | --- |
| Ονοματεπώνυμο φοιτητή | Ευάγγελος Μπάτσαλης |
| Αριθμός Μητρώου | ΑΜ119181 |
| Κωδικός Θ.Ε. | ΠΛΗ24 |
| Κωδικός Τμήματος | ΗΛΕ41 |
| Α/Α Γραπτής εργασίας | 4η ΕΡΓΑΣΙΑ |
| Ονοματεπώνυμο καθηγητή | ΦΙΤΣΙΛΗΣ ΠΑΝΑΓΙΩΤΗΣ |
| Σχόλια προς καθηγητή | Λόγω επαγγελματικών και οικογενειακών υποχρεώσεων αλλά και η έλλειψη γνώσεων της ΠΛΗ30.  Η μελέτη μου στην ύλη των μεταγλωττιστών έγινε σε πολύ σύντομο χρονικό διάστημα όπως και η εγγραφή της εργασίας.  Η επίδοση μου σε αυτή την εργασία όπως θα διαπιστώσετε είναι πολύ χαμηλή.  Ευελπιστώ ότι θα καλύψω τα κενά της ύλης ότι αφορά τους μεταγλωττιστές τον μήνα της επανάληψης του Μαΐου και εν δυνάμει θα ανασυγκροτήσω την βαθμολογία μου στην επόμενη εργασία. |

***Υπεύθυνη Δήλωση Φοιτητή****: Βεβαιώνω ότι είμαι συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία αυτής της εργασίας, είναι πλήρως αναγνωρισμένη και αναφέρεται, είτε στο σημείο «Σχόλια προς καθηγητή», είτε μέσα στην εργασία. Επίσης, έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς, είτε παραφρασμένες. Επίσης, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τη συγκεκριμένη Θεματική Ενότητα.*

Συμφωνώ και αποδέχομαι την ανωτέρω δήλωση

Δε συμφωνώ και δεν αποδέχομαι την ανωτέρω δήλωση (στην περίπτωση αυτή, ο Κ-Σ έχει δικαίωμα να μην αξιολογήσει την εργασία του φοιτητή)

|  |
| --- |
| Ημερομηνία ανακοίνωσης εργασίας: Τετάρτη 23/03/2022  Ημερομηνία υποβολής εργασίας: Τετάρτη 20/04/2022 11:59 μ.μ.  Ημερομηνία υποβολής εργασίας με ατομική παράταση: Τετάρτη 27/04/2022 11:59 μ.μ. |

ΣΤΟΙΧΕΙΑ ΠΟΥ ΣΥΜΠΛΗΡΩΝΕΙ Ο ΚΑΘΗΓΗΤΗΣ

|  |  |
| --- | --- |
| Ημερομηνία αξιολόγησης |  |
| Τελικός βαθμός |  |

Σχόλια προς φοιτητή / φοιτήτρια

##### Αναλυτική Αξιολόγηση

|  |  |  |  |
| --- | --- | --- | --- |
| Άσκηση | Περιγραφή | Ποσοστό | Βαθμός |
| **1** | **Υλοποίηση Τμηματος Μεταγλωττιστή** | **50** |  |
| 1.Α | Συντακτικός Αναλυτής | 10 |  |
| 1.Β | Πίνακας Συμβόλων |  |  |
| 1.Β.1 | Κλάσεις του Πίνακα Συμβόλων | 15 |  |
| 1.Β.2 | Κλήσεις Πίνακα Συμβόλων από Συντακτικό Αναλυτή | 15 |  |
| 1.Β.3 | Έλεγχος Ορθής Λειτουργίας | 10 |  |
| **2.** | **Διφορούμενες Γραμματικές** | **15** |  |
| 2.Α | Εντολή if-then-else | 5 |  |
| 2.Β | Εκφράσεις | 10 |  |
| **3.** | **Μετασχηματισμός Γραμματικής και LL(1) ανίχνευση** | **30** |  |
| 3.Α | Καταλληλότητα γραμματικής | 5 |  |
| 3.Β | Μετασχηματισμός γραμματικής | 10 |  |
| 3.Γ | Σύνολα FIRST-FOLLOW | 5 |  |
| 3.Δ | Υπολογισμός Πίνακα Ανίχνευσης | 5 |  |
| 3.Ε | Ανίχνευση συμβολοσειράς | 5 |  |
| **4.** | **Συμμόρφωση με τους Κανόνες Συγγραφής** | **5** |  |
|  | Σύνολο | **100** |  |

# Εκφώνηση της άσκησης

|  |
| --- |
| **Άσκηση 1 – Υλοποίηση Τμήματος Μεταγλωττιστή** |

Δίνεται η ακόλουθη γραμματική μιας μικρής γλώσσας προγραμματισμού:

<PROGRAM> ::= **program**

<DECLARATIONS>

<SUBPROGRAMS>

<STATEMENTS>

**endprogram**

<DATATYPE> ::= **integer**

| **real**

<DECLARATIONS> ::= ( **declare** <VARLIST> **enddeclare** )\*

<VARLIST> ::= <DATATYPE> ID ( **,** ID )\*

<SUBPROGRAMS> ::= ( <SUBPROGRAM> )\*

<SUBPROGRAM> ::= **function** ID **(** <FORMAL-ARGS> **) :** <DATATYPE>

<DECLARATIONS>

<STATEMENTS>

**endfunction**

| **procedure** ID **(** <FORMAL-ARGS> **)**

<DECLARATIONS>

<STATEMENTS>

**endprocedure**

<FORMAL-ARGS> ::= <PASSMODE> <DATATYPE> ID

( **,** <PASSMODE> <DATATYPE> ID )\*

| ε

<PASSMODE> ::= **in** | ε

<STATEMENTS ::= <STATEMENT> ( **;** <STATEMENT> )\*

<STATEMENT> ::= **input** <INPUT-TAIL>

| **print** <PRINT-TAIL>

| **return** <RETURN-TAIL>

| **call** <CALL-TAIL>

| ID <ASSIGNT-TAIL>

<INPUT-TAIL> ::= **(** ID **)**

<PRINT-TAIL> ::= **(** <EXPRESSION> **)**

<RETURN-TAIL> ::= **(** <EXPRESSION> **)**

<ASSIGN-TAIL> ::= **:=** <EXPRESSION>

<EXPRESSION> ::= < SIGN> <TERM> ( **+** <TERM> | **-** <TERM> )\*

<TERM> ::= <FACTOR> ( **\*** <FACTOR> | **/** <FACTOR> )\*

<FACTOR> ::= numerical\_constant

| ID ( <ID-TAIL> | ε )

| **(** <EXPRESSION> **)**

<ID-TAIL> ::= **(** <ACTUAL-ARGS> **)**

<ACTUAL-ARGS> ::= ~~<PASSMODE> <DATATYPE>~~ ID

( **,** <PASSMODE> <DATATYPE> ID )\*

<SIGN> ::= ε | **+** | **-**

Οι λέξεις κλειδιά, τα σύμβολα δηλαδή που ανήκουν στη γλώσσα, σημειώνονται με πράσινους χαρακτήρες. Για παράδειγμα, η παρένθεση μετά τη λέξη κλειδί print είναι με πράσινους χαρακτήρες, άρα αποτελεί μέρος της σύνταξης της print, ενώ η παρένθεση στον κανόνα EXPRESSION δεν είναι με πράσινα γράμματα, διότι αποτελεί σύμβολο ομαδοποίησης που χρησιμοποιεί η γραμματική για να δείξει την εμβέλεια του συμβόλου “\*”.

Η γλώσσα αυτή υποστηρίζει μόνο ακέραιους και πραγματικούς αριθμούς, ενώ οι μεταβλητές της ξεκινούν από γράμμα και στη συνέχεια μπορούν να αποτελούνται από γράμματα ή ψηφία. Στη γραμματική, με το σύμβολο numerical\_constant συμβολίζουμε τις ακέραιες και πραγματικές σταθερές, ενώ με ID τις μεταβλητές, τις συναρτήσεις και τις διαδικασίες του προγράμματος.

Το παρακάτω πρόγραμμα είναι γραμμένο στη γλώσσα αυτή και ζητάει την ακτίνα ενός κύκλου και στη συνέχεια υπολογίζει και τυπώνει στην οθόνη τη διάμετρο, την περιφέρεια και το εμβαδόν του κύκλου.

program

declare real r, d, a, c enddeclare

declare integer useless enddeclare

declare real dummy enddeclare

function area(in integer x): real

declare real t enddeclare

declare integer dummy enddeclare

declare real useless enddeclare

t:=3.14\*x\*x;

return(t)

endfunction

procedure printx(real a, real b)

declare integer dummy enddeclare

declare real useless enddeclare

print(a);

print(b)

endprocedure

input(r);

d:=2\*r;

c:=3.14\*d;

a:=area(r);

call printx(area,circ)

endprogram

Για την γλώσσα αυτή σας δίνεται ο λεκτικός αναλυτής υλοποιημένος ως μία κλάση της Java. Κατανοήστε τον κώδικα του λεκτικού αναλυτή στον βαθμό που αυτό σας είναι απαραίτητο, ώστε να μπορέσετε να ανταποκριθείτε στα ακόλουθα ερωτήματα.

**Ερώτημα 1.Α – Συντακτικός Αναλυτής**

Ο συντακτικός αναλυτής κατασκευάζεται από τη γραμματική με τρόπο μηχανιστικό. Για κάθε έναν από τους κανόνες της γραμματικής φτιάχνουμε μία μέθοδο. Όταν συναντάμε μη τερματικό σύμβολο καλούμε την αντίστοιχη μέθοδο. Όταν συναντάμε τερματικό σύμβολο, εάν και ο λεκτικός αναλυτής επιστρέφει λεκτική μονάδα που αντιστοιχεί στο τερματικό αυτό σύμβολο, τότε έχουμε αναγνωρίσει επιτυχώς τη λεκτική μονάδα. Αντίθετα, εάν ο λεκτικός αναλυτής δεν επιστρέψει τη λεκτική μονάδα που περιμένει ο συντακτικός αναλυτής, τότε έχουμε λάθος και καλείται ο διαχειριστής σφαλμάτων. Όταν αναγνωριστεί και η τελευταία λέξη του υπό μεταγλώττιση προγράμματος, τότε η συντακτική ανάλυση έχει στεφτεί με επιτυχία.

Σας δίνεται μέρος του κώδικα που υλοποιεί τον συντακτικό αναλυτή. Συμπληρώστε τον κώδικα για τους κανόνες που λείπουν: declarations(), formal\_arguments(), print\_tail(), term().

|  |
| --- |
| ***Μαθησιακά Αποτελέσματα***  Στην άσκηση 1.Α. θα σας δοθεί η δυνατότητα να κατανοήσετε:   * την περιγραφή μίας γλώσσας μέσα από μία γραμματική * την έννοια της συντακτικής ανάλυσης * την κύρια ιδιότητα μιας γραμματικής LL(1) * την υλοποίηση ενός συντακτικού αναλυτή με τη μέθοδο της αναδρομικής κατάβασης * τη λειτουργία ενός συντακτικού αναλυτή * τα σφάλματα μεταγλώττισης που μπορούν να προκύψουν κατά τη φάση της συντακτικής ανάλυσης και τη διαχείρισή τους |

**Απάντηση**

*Να δώσετε τον κώδικα που γράψατε εσείς και υλοποιεί μέρος του συντακτικού αναλυτή επαρκώς σχολιασμένο. Αυτό σημαίνει ότι πρέπει να συμπληρώσετε τον κώδικα που αντιστοιχεί στις μεθόδους που δεν έχουν υλοποιηθεί και να τον παραθέσετε στην απάντησή σας. Μην παραθέσετε ολόκληρο τον κώδικα, παρά μόνο αυτόν που υλοποιήσατε εσείς.*

*Προτιμήστε να εισάγετε τον κώδικα πριν τον τροποποιήσετε για το ερώτημα Β.*

*Εάν δεν έχετε δώσει απάντηση, γράψτε με κεφαλαία γράμματα:* ***ΔΕΝ ΑΠΑΝΤΗΘΗΚΕ.***

*Εάν εν γνώση σας δίνετε ελλιπή απάντηση, γράψτε με κεφαλαία γράμματα:* ***ΕΛΛΙΠΗΣ ΑΠΑΝΤΗΣΗ.*** *Εξηγείστε σε ποιο σημείο θεωρείτε την απάντησή σας ελλιπή και γιατί****.***

private void declarations(){ // global and local vraiables declaration

// add code here

//<DECLARATIONS> ::= ( declare <VARLIST> enddeclare )\*

//μέθοδος Declarations επίσης θα χρησιμοποιήσω while λόγω της επανάληψης της μεθόδου στην παρένθεση

//η οποία διαβάζει το token στη κλάση enum και εφόσον το διαβάσει θα προχωρίσει στο επόμενο token από τη κλάση token της λίστας token

//εφόσον διαβάσει το token enddeclare τότε θα προχωρήσει στο token end declare διαφορετικά θα καλέσει τον διαχειριστή σφαλμάτων με το μηνυμα λάθους ότι περιμένει enddeclare

while("declareTK".equals(token.type.name())){

token = lex.nextToken();

varlist();

if("enddeclareTK".equals(token.type.name())){

token = lex.nextToken();

}else{

error("enddeclare expected");

}

}

}

private void formal\_arguments() // describes the formal parameters of a function or procedure

{

// add code here

//υλοποίηση μεθόδου formal\_arguments()

//<FORMAL-ARGS> ::= <PASSMODE> <DATATYPE> ID

// ( , <PASSMODE> <DATATYPE> ID )\*

// | ε

passmode();

datatype();

if("idTK".equals(token.type.name())){//εαν βρεθεί το τόκεν ID

token = lex.nextToken();

//υλοποίηση με wilhe λόγω της επανάληψης του αστερίσκου με την επαναλαμβανόμενη επανάληψη κάθε φορά που βρίσκει το ,

while("commaTK".equals(token.type.name())){

passmode();

datatype();

if(("idTK".equals(token.type.name()))){

token = lex.nextToken();

}else{

error("ID expected");

}

}

}else if("whitespaceTK".equals(token.type.name())){ //else if για το white space |ε

token = lex.nextToken();

}else {

error("ID expected");//else μύνημα λάθος error αν δεν εισαχθεί ID

}

}

private void print\_tail(){ // follows a print command

// add code here

//υλοποίηση:

//<PRINT-TAIL> ::= ( <EXPRESSION> )

//πρώτα θα εισαχθεί από το token η αριστερή παρένθεση διαφορετικά θα βγάλει μήνυμα λάθους

if("leftpTK".equals(token.type.name())){

token = lex.nextToken();

expression();//έπειτα θα καλέσει τη μέθοδο expression

//τέλος θα εισαχθεί από το token η δεξιά παρένθεση διαφορετικά θα βγάλει μήνυμα λάθους

if("rightpTK".equals(token.type.name())){

token = lex.nextToken();

}else{

error("right parenthesis exprected");

}

}else{

error("left parenthesis exprected");

}

}

private void term(){ // a term consists of factors

// add code here

//<TERM> ::= <FACTOR> ( \* <FACTOR> | / <FACTOR> )\*

//πρώτα θα καλέσω τη μέθοδο factor

factor();

//έπειτα λόγω της επανάληψης του αστερίσκου θα τρέξω την επόμενη επιλογή μέ while για όταν εισάγεται \* ή / θα εκτελείται

while(("multTK".equals(token.type.name()) || ("divTK".equals(token.type.name())))){

token = lex.nextToken();

factor();

}

}

**Ερώτημα 1.Β. Πίνακας Συμβόλων**

Ο πίνακας συμβόλων είναι μία δομή η οποία αποθηκεύει πληροφορίες για οτιδήποτε έχει όνομα μέσα σε ένα πρόγραμμα. Στη γλώσσα προγραμματισμού της άσκησης, ονομασία έχει οτιδήποτε έχει συμβολιστεί με ID, δηλαδή τα ονόματα των μεταβλητών, τα ονόματα των συναρτήσεων και διαδικασιών και τα ονόματα των τυπικών παραμέτρων.

Ο πίνακας συμβόλων είναι μία δυναμική δομή στην οποία κατά τη διάρκεια της μετάφρασης ενός προγράμματος προστίθεται και αφαιρείται πληροφορία, έτσι ώστε σε κάθε στιγμή να περιέχονται σε αυτόν μόνο οι μεταβλητές, συναρτήσεις, διαδικασίες και παράμετροι που εκείνη τη στιγμή της μετάφρασης έχει δικαίωμα να προσπελάσει το πρόγραμμα. Ο πίνακας συμβόλων που θα υλοποιήσουμε είναι απλοποιημένος σε πολλά σημεία, αλλά εκπαιδευτικά ικανός να μας εισαγάγει στον τρόπο ανάπτυξης και λειτουργίας ενός πίνακα συμβόλων.

Στον πίνακα συμβόλων αποθηκεύονται οντότητες. Κάθε οντότητα αντιστοιχεί σε μία μεταβλητή, συνάρτηση, διαδικασία ή τυπική παράμετρο. Για τη γλώσσα της άσκησης οι οντότητες μπορούν να χωριστούν σε καθολικές (*global*) και τοπικές (*local*). Έτσι, ο πίνακας συμβόλων θα αποτελείται από δύο λίστες οντοτήτων, μία για τις καθολικές και μία για τις τοπικές οντότητες. Οι καθολικές οντότητες προστίθενται στην αντίστοιχη λίστα, μόλις συναντάμε τη δήλωσή τους στο κυρίως πρόγραμμα. Οι τοπικές οντότητες προστίθενται στην λίστα με τις τοπικές οντότητες όταν συναντάμε τη δήλωσή τους και αφαιρούνται όταν τερματιστεί η μετάφραση της συνάρτησης ή της διαδικασίας στην οποία ανήκουν. Πρακτικά, με την ολοκλήρωση της μετάφρασης μίας συνάρτησης ή διαδικασίας η λίστα με τις τοπικές οντότητες μηδενίζεται.

Πέρα από τις πράξεις της πρόσθεσης μίας οντότητας και της αφαίρεσης όλων των τοπικών οντοτήτων, όταν τελειώσει η εκτέλεση μιας συνάρτησης ή διαδικασίας, υπάρχει και η αναζήτηση οντότητας η οποία γίνεται ως εξής: αρχικά αναζητούμε την οντότητα που μας ζητήθηκε στη λίστα με τις τοπικές οντότητες. Αν η οντότητα βρεθεί, τότε την επιστέφουμε ως αποτέλεσμα. Αν η οντότητα δεν βρεθεί στις τοπικές οντότητες, τότε την αναζητούμε στις καθολικές οντότητες. Στην περίπτωση που βρεθεί εκεί, τότε πάλι την επιστρέφουμε σαν αποτέλεσμα. Αν δεν βρεθεί και στη λίστα με τις καθολικές οντότητες, τότε επιστρέφουμε κατάλληλο μήνυμα σφάλματος και τερματίζουμε την εκτέλεση. Με τον μηχανισμό αυτόν υλοποιείται η υπερκάλυψη των καθολικών οντοτήτων από τοπικές. Φυσικά κατά την εισαγωγή μίας νέας οντότητας πρέπει να γίνεται έλεγχος αν οντότητα με το ίδιο όνομα υπάρχει στη λίστα που η νέα οντότητα επιχειρείται να τοποθετηθεί. Σε μια τέτοια περίπτωση πρέπει να εμφανιστεί μήνυμα σφάλματος και να τερματιστεί η μετάφραση.

Οι οντότητες του πίνακα συμβόλων για τη γλώσσα της άσκησης είναι ο ακόλουθες:

Variable:

name : το όνομα της οντότητας

level: η εμβέλεια: local/global

type: ο τύπος δεδομένων - integer/real

location : η σχετική θέση της στη μνήμη

Παρατηρήστε ότι δεν υπάρχει πεδίο για την τιμή της μεταβλητής. Θυμηθείτε ότι το πρόγραμμα μεταφράζεται και δεν εκτελείται.

FormalParameter:

name : το όνομα της οντότητας

level: η εμβέλεια: local/global

type: ο τύπος δεδομένων - integer/real

mode: ο τρόπος περάσματος - call\_by\_value/call\_by\_reference

location : η σχετική θέση της στη μνήμη

Function:

name : το όνομα της οντότητας

type: ο τύπος δεδομένων – integer/real

level: η εμβέλεια: για τη γλώσσα μας είναι πάντοτε global

arguments: λίστα με τις τυπικές παραμέτρους. Κάθε στοιχείο της λίστας είναι αντικείμενο της κλάσης FormalParameter

Procedure:

name : το όνομα της οντότητας

level: η εμβέλεια: για τη γλώσσα μας είναι πάντοτε global

arguments: λίστα με τις τυπικές παραμέτρους. Κάθε στοιχείο της λίστας είναι αντικείμενο της κλάσης FormalParameter

Το πεδίο “location” δείχνει τη σχετική θέση της οντότητας από την αρχή της λίστας (τοπική ή καθολική) στην οποία βρίσκεται. Χώρο στη μνήμη χρειαζόμαστε μόνο για τις μεταβλητές και παραμέτρους και όχι για τις συναρτήσεις και διαδικασίες. Αν η μεταβλητή ή η τυπική παράμετρος είναι ακέραια θεωρούμε ότι χρειαζόμαστε 4 bytes για να αποθηκευτεί. Αν η μεταβλητή ή η τυπική παράμετρος είναι πραγματικός αριθμός, τότε θεωρούμε ότι χρειαζόμαστε 8 bytes για να αποθηκευτεί. Άρα για την παρακάτω δήλωση:

declare real x,y enddeclare

declare integer a enddeclare

declare real z enddeclare

Το x θα βρίσκεται στη θέση 0, το y στην θέση 8, to a στη θέση 16, το z στη θέση 20.

Ας δούμε πως μεταβάλλεται δυναμικά ο πίνακας συμβόλων κατά τη διάρκεια της μετάφρασης του παραδείγματος υπολογισμού εμβαδού και περιφέρειας του κύκλου.

Μόλις η μετάφραση ολοκληρώσει την ανάλυση της γραμμής 2, ο πίνακας συμβόλων θα είναι ο ακόλουθος:

========================

Symbol Table

--------------------------

Global level:

Variable{name=r, type=real, location=0, level=global}

Variable{name=d, type=real, location=8, level=global}

Variable{name=a, type=real, location=16, level=global}

Variable{name=c, type=real, location=24, level=global}

--------------------------

Local level:

========================

Μετά τις γραμμές 3 και 4 θα έχουν προστεθεί μερικές ακόμα καθολικές μεταβλητές:

========================

Symbol Table

--------------------------

Global level:

Variable{name=r, type=real, location=0, level=global}

Variable{name=d, type=real, location=8, level=global}

Variable{name=a, type=real, location=16, level=global}

Variable{name=c, type=real, location=24, level=global}

Variable{name=useless, type=integer, location=32, level=global}

Variable{name=dummy, type=real, location=36, level=global}

--------------------------

Local level:

========================

Μετά τη γραμμή 6 θα έχει προστεθεί η συνάρτηση area στο καθολικό επίπεδο και οι τυπικές της παράμετροι στο τοπικό επίπεδο:

========================

Symbol Table

--------------------------

Global level:

Variable{name=r, type=real, location=0, level=global}

Variable{name=d, type=real, location=8, level=global}

Variable{name=a, type=real, location=16, level=global}

Variable{name=c, type=real, location=24, level=global}

Variable{name=useless, type=integer, location=32, level=global}

Variable{name=dummy, type=real, location=36, level=global}

Function{name=area, type=real, argumemts=FormalParameter{name=x, mode=by\_value,type=integer, location=0, level=local}}

--------------------------

Local level:

FormalParameter{name=x, mode=by\_value,type=integer, location=0, level=local}

========================

Μετά τις γραμμές 7, 8 και 9 θα έχουν προστεθεί και οι τοπικές μεταβλητές:

========================

Symbol Table

--------------------------

Global level:

Variable{name=r, type=real, location=0, level=global}

Variable{name=d, type=real, location=8, level=global}

Variable{name=a, type=real, location=16, level=global}

Variable{name=c, type=real, location=24, level=global}

Variable{name=useless, type=integer, location=32, level=global}

Variable{name=dummy, type=real, location=36, level=global}

Function{name=area, type=real, argumemts=FormalParameter{name=x, mode=by\_value,type=integer, location=0, level=local}}

--------------------------

Local level:

FormalParameter{name=x, mode=by\_value,type=integer, location=0, level=local}

Variable{name=t, type=real, location=**4**, level=local}

Variable{name=dummy, type=integer, location=**12**, level=local}

Variable{name=useless, type=real, location=**16**, level=local}

========================

Η μορφή του πίνακα συμβόλων δεν μεταβάλλεται στις γραμμές 10 και 11 (αν θέλουμε να είμαστε πιο αυστηροί, σε έναν πίνακα συμβόλων στη φάση αυτή προστίθενται προσωρινές μεταβλητές, αλλά για απλούστευση της άσκησης δεν τις έχουμε συμπεριλάβει στις απαιτήσεις της).

Στην γραμμή 12 αφαιρούνται όλες οι μεταβλητές από το τοπικό επίπεδο:

========================

Symbol Table

--------------------------

Global level:

Variable{name=r, type=real, location=0, level=global}

Variable{name=d, type=real, location=8, level=global}

Variable{name=a, type=real, location=16, level=global}

Variable{name=c, type=real, location=24, level=global}

Variable{name=useless, type=integer, location=32, level=global}

Variable{name=dummy, type=real, location=36, level=global}

Function{name=area, type=real, argumemts=FormalParameter{name=x, mode=by\_value,type=integer, location=0, level=local}}

--------------------------

Local level:

========================

Τη θέση τους θα πάρουν οι αντίστοιχες μεταβλητές από τη διαδικασία printx.

Μετά τη γραμμή 14, ο πίνακας συμβόλων θα γίνει:

========================

Symbol Table

--------------------------

Global level:

Variable{name=r, type=real, location=0, level=global}

Variable{name=d, type=real, location=8, level=global}

Variable{name=a, type=real, location=16, level=global}

Variable{name=c, type=real, location=24, level=global}

Variable{name=useless, type=integer, location=32, level=global}

Variable{name=dummy, type=real, location=36, level=global}

Function{name=area, type=real, argumemts=FormalParameter{name=x, mode=by\_value,type=integer, location=0, level=local}}

Procedure{name=printx, argumemts=FormalParameter{name=a, mode=by\_reference,type=real, location=0,

level=local},FormalParameter{name=b, mode=by\_reference,type=real, location=**8**, level=local}}

--------------------------

Local level:

FormalParameter{name=a, mode=by\_reference,type=real, location=0, level=local}

FormalParameter{name=b, mode=by\_reference,type=real, location=16, level=local}

========================

Ενώ μετά την προσθήκη των τοπικών μεταβλητών:

========================

Symbol Table

--------------------------

Global level:

Variable{name=r, type=real, location=0, level=global}

Variable{name=d, type=real, location=8, level=global}

Variable{name=a, type=real, location=16, level=global}

Variable{name=c, type=real, location=24, level=global}

Variable{name=useless, type=integer, location=32, level=global}

Variable{name=dummy, type=real, location=36, level=global}

Function{name=area, type=real, argumemts=FormalParameter{name=x, mode=by\_value,type=integer, location=0, level=local}}

Procedure{name=printx, argumemts=FormalParameter{name=a, mode=by\_reference,type=real, location=0,

level=local},FormalParameter{name=b, mode=by\_reference,type=real, location=**8**, level=local}}

--------------------------

Local level:

FormalParameter{name=a, mode=by\_reference,type=real, location=0, level=local}

FormalParameter{name=b, mode=by\_reference,type=real, location=**8**, level=local}

Variable{name=dummy, type=integer, location=**16**, level=local}

Variable{name=useless, type=real, location=**20**, level=local}

========================

Με το τέλος της printx και για όσο διαρκεί η μετάφραση του κυρίως προγράμματος, θα υπάρχει στον πίνακα συμβόλων μόνο η καθολική λίστα:

========================

Symbol Table

--------------------------

Global level:

Variable{name=r, type=real, location=0, level=global}

Variable{name=d, type=real, location=8, level=global}

Variable{name=a, type=real, location=16, level=global}

Variable{name=c, type=real, location=24, level=global}

Variable{name=useless, type=integer, location=32, level=global}

Variable{name=dummy, type=real, location=36, level=global}

Function{name=area, type=real, argumemts=FormalParameter{name=x, mode=by\_value,type=integer, location=0, level=local}}

Procedure{name=printx, argumemts=FormalParameter{name=a, mode=by\_reference,type=real, location=0, level=local}

,FormalParameter{name=b, mode=by\_reference,type=real, location=**8**, level=local}}

--------------------------

Local level:

========================

|  |
| --- |
| ***Μαθησιακά Αποτελέσματα***  Στην άσκηση 1.Β. θα σας δοθεί η δυνατότητα να κατανοήσετε:   * ποια η χρησιμότητας ενός πίνακα συμβόλων * πως θα σχεδιάσουμε έναν πίνακα συμβόλων * πως εισάγουμε και αναζητούμε πληροφορία μέσα σε έναν πίνακα συμβόλων * πως εντοπίζουμε σφάλματα με τη βοήθεια του πίνακα συμβόλων * πως θα αξιοποιηθεί η τεχνολογία του αντικειμενοστραφούς προγραμματισμού σε έναν πίνακα συμβόλων |

**Ερώτημα 1.Β.1 – Κλάσεις του Πίνακα Συμβόλων**

Για τον πίνακα συμβόλων σας δίνονται υλοποιημένες οι κλάσεις EntityList και SymbolTable. Οι κλάσεις Entity, Variable, FormalParameter, Function και Procedure ζητείται να υλοποιηθούν από εσάς. Υλοποιήστε τις κλάσεις αυτές, εκμεταλλευόμενοι τη φιλοσοφία του αντικειμενοστρεφούς προγραμματισμού. Πέρα από τους constructors, τους setters και getters που απαιτούνται, πρέπει να υλοποιήσετε κατάλληλα την *toString()*, ώστε ο πίνακας συμβόλων να εμφανίζεται όπως φαίνεται στο παράδειγμα ανάπτυξης του πίνακα συμβόλων που δόθηκε παραπάνω.

**Απάντηση**

*Να δώσετε τον κώδικα που απαιτείται ώστε να υλοποιηθούν οι παραπάνω κλάσεις. Να παραθέσετε ολόκληρο τον για κάθε μία από τις κλάσεις. Εκμεταλλευτείτε τη φιλοσοφία της αντικειμενοστραφούς ανάπτυξης.*

*Εάν δεν έχετε δώσει απάντηση, γράψτε με κεφαλαία γράμματα:* ***ΔΕΝ ΑΠΑΝΤΗΘΗΚΕ.***

*Εάν εν γνώση σας δίνετε ελλιπή απάντηση, γράψτε με κεφαλαία γράμματα:* ***ΕΛΛΙΠΗΣ ΑΠΑΝΤΗΣΗ.*** *Εξηγείστε σε ποιο σημείο θεωρείτε την απάντησή σας ελλιπή και γιατί****.***

//Entity Class

package parser\_ex4;

// add code

//τα χαρακτηριστικά της οντότητας βάση εκφώνησης της άσκησης και των χαρακτηριστικών που εντοπίζω είναι:

/\*

    name : το όνομα της οντότητας

    level: η εμβέλεια: local/global

\*/

//Δημιουργία κλάσης με το όνομα Entity

public class Entity{

    String name;

    String level;

    //δημιουργια full constructor με όλα τα χαρακτηριστικά

    public Entity(String name, String level){

        this.name = name;

        this.level = level;

}

    //δημιουργία των getter και setter της κλάσης

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public String getLevel() {

        return level;

    }

    public void setLevel(String level) {

        this.level = level;

    }

    //το string Method

    @Override

    public String toString(){

        return ", level="+level;

    }

}

//Variable Class

package parser\_ex4;

// add code

//Variable:

//  name : το όνομα της οντότητας

//  level: η εμβέλεια: local/global

//  type: ο τύπος δεδομένων - integer/real

//  location : η σχετική θέση της στη μνήμη

//δημιουργία κλάσης και κληρονομεί από την οντότητα τα βασικά χαρακτηριστικά τα οποία επεκτείνει επιπλέον χαρακτηριστικά τον τύπο και το location

public class Variable extends Entity{

    String type;

    int location;

    //δημιουργία extended constructor

    public Variable(String name, String level) {

        super(name, level);

    }

    //δημιουργία extended constructor με επιπλέον ψαρακτηριστικά

    public Variable(String name, String level, String type, int location) {

        super(name, level);

        this.type = type;

        this.location = location;

    }

    //δημιουργία getter και setter

    public String getType() {

        return type;

    }

    public void setType(String type) {

        this.type = type;

    }

    public int getLocation() {

        return location;

    }

    public void setLocation(int location) {

        this.location = location;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public String getLevel() {

        return level;

    }

    public void setLevel(String level) {

        this.level = level;

    }

    //δημιουργία μέθοδος toString

    @Override

    public String toString(){

        return "Variable{name="+name+", type="+type+", location="+location+super.level+"}";

    }

}

//FormalParameter Class

package parser\_ex4;

// add code

/\*

FormalParameter:

    name : το όνομα της οντότητας

    level: η εμβέλεια: local/global

    type: ο τύπος δεδομένων - integer/real

    mode: ο τρόπος περάσματος - call\_by\_value/call\_by\_reference

    location : η σχετική θέση της στη μνήμη

\*/

//δημιουργία κλάσης και κληρονομεί από την οντότητα τα βασικά χαρακτηριστικά τα οποία επεκτείνει επιπλέον χαρακτηριστικά τον τύπο και το location

public class FormalParameter extends Entity{

    String type;

    String mode;

    int location;

//δημιουργία extended constructor

    public FormalParameter(String name, String level) {

        super(name, level);

    }

//δημιουργία extended constructor με επιπλέον ψαρακτηριστικά

    public FormalParameter(String name, String level, String type, String mode, int location) {

        super(name, level);

        this.type = type;

        this.mode = mode;

        this.location = location;

    }

//δημιουργία getter και setter

    public String getType() {

        return type;

    }

    public void setType(String type) {

        this.type = type;

    }

    public String getMode() {

        return mode;

    }

    public void setMode(String mode) {

        this.mode = mode;

    }

    public int getLocation() {

        return location;

    }

    public void setLocation(int location) {

        this.location = location;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public String getLevel() {

        return level;

    }

    public void setLevel(String level) {

        this.level = level;

    }

     //δημιουργία μέθοδος toString

    @Override

    public String toString(){

        return "FormalParameter{name="+name+", mode="+mode+", type="+type+", location="+location+super.level+"}";

    }

}

//Function Class

package parser\_ex4;

// add code

import java.util.ArrayList;

/\*

Function:

    name : το όνομα της οντότητας

    type: ο τύπος δεδομένων – integer/real

    level: η εμβέλεια: για τη γλώσσα μας είναι πάντοτε global

arguments: λίστα με τις τυπικές παραμέτρους. Κάθε στοιχείο της λίστας είναι αντικείμενο της κλάσης FormalParameter

\*/

//δημιουργία κλάσης και κληρονομεί από την οντότητα τα βασικά χαρακτηριστικά τα οποία επεκτείνει επιπλέον χαρακτηριστικά τον τύπο και το location

public class Function extends Entity{

    String type;

    //δημιουργία δυναμικής συνδεδεμένης λίστας για ευκολία χρήσης της βάση της εκφώνησης

    //arguments: λίστα με τις τυπικές παραμέτρους. Κάθε στοιχείο της λίστας είναι αντικείμενο της κλάσης FormalParameter

    ArrayList<FormalParameter> arguments;

//δημιουργία extended constructor

    public Function(String name, String level) {

        super(name, level);

    }

//δημιουργία extended constructor με επιπλέον ψαρακτηριστικά

    public Function(String name, String level, String type, ArrayList<FormalParameter> arguments) {

        super(name, level);

        this.type = type;

        this.arguments = new ArrayList<>();

    }

    //δημιουργία getter και setter

    public String getType() {

        return type;

    }

    public void setType(String type) {

        this.type = type;

    }

    public ArrayList<FormalParameter> getArguments() {

        return arguments;

    }

    public void setArguments(ArrayList<FormalParameter> arguments) {

        this.arguments = arguments;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public String getLevel() {

        return level;

    }

    public void setLevel(String level) {

        this.level = level;

    }

    //add formal parameter into arguments list

    public void setFormalParameterIntoArguments(FormalParameter e){

        arguments.add(e);

    }

    //to string method

    @Override

    public String toString(){

        return "Function{name="+name+", type="+type+", arguments="+arguments+"}";

    }

}

//Procedure Class

package parser\_ex4;

// add code

import java.util.ArrayList;

/\*

Procedure:

    name : το όνομα της οντότητας

    level: η εμβέλεια: για τη γλώσσα μας είναι πάντοτε global

arguments: λίστα με τις τυπικές παραμέτρους. Κάθε στοιχείο της λίστας είναι αντικείμενο της κλάσης FormalParameter

\*/

//δημιουργία κλάσης και κληρονομεί από την οντότητα τα βασικά χαρακτηριστικά τα οποία επεκτείνει επιπλέον χαρακτηριστικά τον τύπο και το location

public class Procedure extends Entity{

    //επιπλέον ζητούμενα χαρακτηριστικά

    String type;

    //δημιουργία δυναμικής συνδεδεμένης λίστας για ευκολία χρήσης της βάση της εκφώνησης

    //arguments: λίστα με τις τυπικές παραμέτρους. Κάθε στοιχείο της λίστας είναι αντικείμενο της κλάσης FormalParameter

    ArrayList<FormalParameter> arguments;

    //δημιουργία extended constructor

    public Procedure(String name, String level){

        super(name,level);

    }

    //δημιουργία extended constructor με επιπλέον ψαρακτηριστικά

    public Procedure(String name, String level, String type, ArrayList<FormalParameter> arguments) {

        super(name, level);

        this.type = type;

        this.arguments = new ArrayList<>();

    }

    //δημιουργία getter και setter

    public String getType() {

        return type;

    }

    public void setType(String type) {

        this.type = type;

    }

    public ArrayList<FormalParameter> getArguments() {

        return arguments;

    }

    public void setArguments(ArrayList<FormalParameter> arguments) {

        this.arguments = arguments;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public String getLevel() {

        return level;

    }

    public void setLevel(String level) {

        this.level = level;

    }

        //add formal parameter into arguments list

    public void setFormalParameterIntoArguments(FormalParameter e){

        arguments.add(e);

    }

    //to string method

    public String toString(){

        return "Procedure{name="+name+", arguments="+arguments+","+arguments+"}";

    }

}

**Ερώτημα 1.Β.2 – Κλήσεις Πίνακα Συμβόλων από τον Συντακτικό Αναλυτή**

Μελετήστε τον κώδικα του συντακτικού αναλυτή και συμπληρώστε τον κατάλληλο κώδικα ώστε να υλοποιείται πίνακας συμβόλων για την γλώσσα και τις προδιαγραφές που περιγράφηκαν παραπάνω.

Μπορείτε να κάνετε ό,τι αλλαγές/προσθήκες απαιτούνται να γίνουν, για παράδειγμα να αλλάξετε τον τύπο δεδομένων που επιστρέφει μία συνάρτηση ή να προσθέσετε παραμέτρους σε αυτήν.

**Απάντηση**

*Να δώσετε τον κώδικα που απαιτείται ώστε να υλοποιηθούν οι απαιτήσεις/προδιαγραφές για τον πίνακα συμβόλων. Να παραθέσετε ολόκληρο τον για κάθε μία από τις μεθόδους του συντακτικού αναλυτή την οποία χρειάστηκε να τροποποιήσετε.*

*Εάν δεν έχετε δώσει απάντηση, γράψτε με κεφαλαία γράμματα:* ***ΔΕΝ ΑΠΑΝΤΗΘΗΚΕ.***

*Εάν εν γνώση σας δίνετε ελλιπή απάντηση, γράψτε με κεφαλαία γράμματα:* ***ΕΛΛΙΠΗΣ ΑΠΑΝΤΗΣΗ.*** *Εξηγείστε σε ποιο σημείο θεωρείτε την απάντησή σας ελλιπή και γιατί****.***

***ΔΕΝ ΑΠΑΝΤΗΘΗΚΕ***

**Ερώτημα 1.Β3. – Έλεγχος Ορθής Λειτουργίας**

Να ελέγξετε την ορθή λειτουργία του μεταγλωττιστή με προγράμματα τα οποία θα συντάξετε (δύο με τρία προγράμματα) και θα είναι κατάλληλα επιλεγμένα ώστε να πείθουν για την ορθή λειτουργία του πίνακα συμβόλων.

**Απάντηση**

*Να ελέγξετε την ορθή λειτουργία του πίνακα συμβόλων χρησιμοποιώντας και παραδίδοντας προγράμματα που έχετε φτιάξει για το σκοπό αυτό. Η κατάλληλη επιλογή των προγραμμάτων βαθμολογείται.*

*Να συμπεριλάβετε στην απάντησή σας*

*Α) τα προγράμματα αυτά*

*Β) screenshots από την έξοδο που εμφανίζουν*

*Εάν δεν έχετε δώσει απάντηση, γράψτε με κεφαλαία γράμματα:* ***ΔΕΝ ΑΠΑΝΤΗΘΗΚΕ.***

*Εάν εν γνώση σας δίνετε ελλιπή απάντηση, γράψτε με κεφαλαία γράμματα:* ***ΕΛΛΙΠΗΣ ΑΠΑΝΤΗΣΗ.*** *Εξηγείστε σε ποιο σημείο θεωρείτε την απάντησή σας ελλιπή και γιατί****.***

***ΔΕΝ ΑΠΑΝΤΗΘΗΚΕ***

**Άσκηση 2 – Διφορούμενες Γραμματικές**

|  |
| --- |
| ***Μαθησιακά Αποτελέσματα***  Στην άσκηση 2. θα σας δοθεί η δυνατότητα να κατανοήσετε:   * Τις γραμματικές * Ποιες γραμματικές είναι διφορούμενες * Πως ανιχνεύουμε ότι αν μία γραμματική είναι διφορούμενη * Ποια γραμματική είναι κατάλληλη για LL(1) ανίχνευση |

**Ερώτημα 2.A – Εντολή if-then-else**

Η παρακάτω γραμματική περιγράφει συντακτικά την εντολή if-then-else σε μια τυπική γλώσσα προγραμματισμού:

stmt  **if** expr **then** stmt

| matched\_Stmt

matched\_Stmt  **if** expr **then** matched\_Stmt **else** stmt

| **other**

Όπου **other** υποδηλώνει οποιαδήποτε άλλη εντολή.

Δείξτε ότι η παραπάνω γραμματική είναι διφορούμενη. Δώστε ένα παράδειγμα που να το αποδεικνύει μαζί με τα σχετικά δέντρα ανίχνευσης.

**Απάντηση**

*Εάν δεν έχετε δώσει απάντηση, γράψτε με κεφαλαία γράμματα:* ***ΔΕΝ ΑΠΑΝΤΗΘΗΚΕ.***

*Εάν εν γνώση σας δίνετε ελλιπή απάντηση, γράψτε με κεφαλαία γράμματα:* ***ΕΛΛΙΠΗΣ ΑΠΑΝΤΗΣΗ.*** *Εξηγείστε σε ποιο σημείο θεωρείτε την απάντησή σας ελλιπή και γιατί****.***

***ΔΕΝ ΑΠΑΝΤΗΘΗΚΕ***

**Ερώτημα 2.Β – Εκφράσεις**

Να συμπληρώσετε την παρακάτω γραμματική ώστε να είναι κατάλληλη για LL(1) ανίχνευση εκφράσεων που περιλαμβάνουν μεταβλητές, τις δυαδικές πράξεις της πρόσθεσης και του πολλαπλασιασμού σε infix μορφή με τη συνήθη προτεραιότητα/προσεταιριστικότητα και παρενθέσεις. Να χρησιμοποιήσετε αποκλειστικά τα δοθέντα μη-τερματικά σύμβολα και να συμπληρώσετε όπου χρειάζεται το δεξί σκέλος των κανόνων. Θα πρέπει να εξηγήσετε πώς προέκυψε η τελική γραμματική.

|  |  |  |  |
| --- | --- | --- | --- |
|  | S |  |  |
|  | Ε |  | Χ Υ |
|  | Υ |  |  |
|  | X |  | Ζ W |
|  | W |  |  |
|  | Z |  |  |
|  |  |  |  |

**Απάντηση**

*Εάν δεν έχετε δώσει απάντηση, γράψτε με κεφαλαία γράμματα:* ***ΔΕΝ ΑΠΑΝΤΗΘΗΚΕ.***

*Εάν εν γνώση σας δίνετε ελλιπή απάντηση, γράψτε με κεφαλαία γράμματα:* ***ΕΛΛΙΠΗΣ ΑΠΑΝΤΗΣΗ.*** *Εξηγείστε σε ποιο σημείο θεωρείτε την απάντησή σας ελλιπή και γιατί****.***

***ΔΕΝ ΑΠΑΝΤΗΘΗΚΕ***

**Άσκηση 3 – Μετασχηματισμός Γραμματικής και LL(1) ανίχνευση**

Έστω η ακόλουθη γραμματική χωρίς συμφραζόμενα με αρχικό σύμβολο το S και τερματικά σύμβολα τα **x**, **y**, **z** και **w**.

|  |  |  |  |
| --- | --- | --- | --- |
| (1) | S |  | **x** |
| (2) |  | | | **y** S **z** S **w** |
| (3) |  | | | **y** T **w** |
| (4) | T |  | S |
| (5) |  | | | T S |

|  |
| --- |
| ***Μαθησιακά Αποτελέσματα***  Στην άσκηση 4. θα σας δοθεί η δυνατότητα να κατανοήσετε:   * Πως διαπιστώνουμε αν μία γραμματική είναι κατάλληλη για LL(1) ανίχνευση * Πως μετασχηματίζουμε μια γραμματική ώστε να γίνει κατάλληλη για LL(1) ανίχνευση * Τι είναι τα σύνολα FIRST και FOLLOW * Τον τρόπο υπολογισμού τους * Τι είναι και πως κατασκευάζουμε έναν πίνακα ανίχνευσης |

**Ερώτημα 3.Α – Καταλληλότητα γραμματικής**

Να εξηγήσετε γιατί η δοθείσα γραμματική δεν είναι κατάλληλη για LL(1) ανίχνευση.

**Απάντηση**

*Εάν δεν έχετε δώσει απάντηση, γράψτε με κεφαλαία γράμματα:* ***ΔΕΝ ΑΠΑΝΤΗΘΗΚΕ.***

*Εάν εν γνώση σας δίνετε ελλιπή απάντηση, γράψτε με κεφαλαία γράμματα:* ***ΕΛΛΙΠΗΣ ΑΠΑΝΤΗΣΗ.*** *Εξηγείστε σε ποιο σημείο θεωρείτε την απάντησή σας ελλιπή και γιατί****.***

Η δοθείσα γραμματική δεν είναι κατάλληλη για LL(1) διότι όπως αναφέρει και η θεωρία διακρίνω τρία σφάλματα στους όρους καταλληλόλητας:

* Διακρίνω το y και το w ως κοινό πρόθεμα στον κανόνα (2) και (3) δηλαδή:

S -> x | y S z S w | y T w

* Υπάρχει άμεση αριστερή αναδρομή στο μη τερματικό σύμβολο Τ όπου το Τ -> Τ S
* Επιπλέον η θεωρία αναφέρει: Κάθε κανόνας που περιέχει Ν εναλλακτικά δεξιά μέρη να ξαναγραφεί έτσι ώστε τουλάχιστον τα Ν-1 να αρχίζουν με τερματικό σύμβολο

Στο (4), (5) παράδειγμα:

(4) Τ -> Τ

(5) | Τ S

Κανένα από τα δεξιά μέρη δεν αρχίζει με τερματικό σύμβολο

**Ερώτημα 3.Β – Μετασχηματισμός γραμματικής**

Να μετασχηματίσετε την γραμματική ώστε να είναι κατάλληλη για LL(1) ανίχνευση. Να κάνετε τις κατάλληλες αντικαταστάσεις ώστε να προκύψει ως τελική γραμματική αυτή με τους λιγότερους κανόνες.

**Απάντηση**

*Εάν δεν έχετε δώσει απάντηση, γράψτε με κεφαλαία γράμματα:* ***ΔΕΝ ΑΠΑΝΤΗΘΗΚΕ.***

*Εάν εν γνώση σας δίνετε ελλιπή απάντηση, γράψτε με κεφαλαία γράμματα:* ***ΕΛΛΙΠΗΣ ΑΠΑΝΤΗΣΗ.*** *Εξηγείστε σε ποιο σημείο θεωρείτε την απάντησή σας ελλιπή και γιατί****.***

(1) S  x

(2) | y S z S w

(3) | y T w

(4) T  S

(5) | T S

Θα ξεκινήσω με τον μετασχηματισμό κοινού παράγοντα διότι διακρίνω κοινό πρόθεμα στον κανόνα (2) και (3)

S -> x | y S z S w | y T w

άρα βάση κανόνα αντικατάσταση γίνεται με τον νέο κανόνα C έχουμε:

1. S -> x | y C
2. C -> S z S w | T w
3. T -> S | T S

έπειτα θα μετασχηματίσω βάση της άμεσης αριστερής αναδρομής T  S | T S όπου θα χρησιμοποιήσω τον νέο βοηθητικό κανόνα Κ

Τελικώς έχουμε την ακόλουθη γραμματική:

1. S -> x | y C
2. C -> S z S w | T w
3. T -> SK
4. K -> TK | ε

Σε επέκταση των κανόνων έχουμε

1. S -> x
2. S -> y C
3. C -> S z S w
4. C -> T w
5. T - > SK
6. K -> TK
7. K -> ε

Αντικατάσταση του κανόνα (5) στον (4) και επίσης σβήνω τον κανόνα (5), άρα έχουμε:

1. S -> x
2. S -> y C
3. C -> S z S w
4. C -> S K w
5. K -> T K
6. K -> ε

Διακρίνω κοινό πρόθεμα στον κανόνα (3) και (4) άρα:

1. S -> x
2. S -> y C
3. C -> S D
4. D -> z S w
5. D -> K w
6. K -> T K
7. K -> ε

Μπορώ να κάνω αντικατάσταση από τον κανόνα (6) στον κανόνα (5), άρα:

1. S -> x
2. S -> y C
3. C -> S D
4. D -> z S w
5. D -> Τ Κ w
6. K -> ε

**Ερώτημα 3.Γ – Σύνολα FIRST-FOLLOW**

Για την γραμματική που θα προκύψει από το προηγούμενο ερώτημα να παρουσιαστεί αναλυτικά ο υπολογισμός των συνόλων *First* και *Follow* για τα μη-τερματικά σύμβολα της γραμματικής και στη συνέχεια να συμπληρώσετε τον παρακάτω πίνακα.

|  |  |  |
| --- | --- | --- |
|  | **First** | **Follow** |
| **S** |  |  |
| **…** |  |  |

**Απάντηση**

*Εάν δεν έχετε δώσει απάντηση, γράψτε με κεφαλαία γράμματα:* ***ΔΕΝ ΑΠΑΝΤΗΘΗΚΕ.***

*Εάν εν γνώση σας δίνετε ελλιπή απάντηση, γράψτε με κεφαλαία γράμματα:* ***ΕΛΛΙΠΗΣ ΑΠΑΝΤΗΣΗ.*** *Εξηγείστε σε ποιο σημείο θεωρείτε την απάντησή σας ελλιπή και γιατί****.***

|  |  |  |
| --- | --- | --- |
|  | **First** | **Follow** |
| **S** | {x,y} | {$,z,w} |
| **C** | {x,y,z,w} | {$,z,w} |
| **D** | {z,w} | {$,z,w} |
| **K** | ε |  |

**Υπολογισμός σύνολα First**

First(S) = First(x) First(y C) = {x,y}

First(C) = First(S) First(D) = First(x)First(y C) First(z S w) First(T K w) = {x,y,z,w}

First(D) = First(z S w) First(T K w) = {z,w}

First(K) = First(ε) = {ε}

**Υπολογισμός σύνολα Follow**

Follow(S) = $

Follow(S) =Follow(D)={w}

Follow(S) = First(D) = {z,w}

Follow(C) = Follow(S) = {$,z,w}

Follow(D) = Follow(C) = Follow(S) = {$,z,w}

**Ερώτημα 3.Δ – Υπολογισμός Πίνακα Ανίχνευσης**

Να υπολογίσετε τον πίνακα ανίχνευσης του μοντέλου LL(1) ανίχνευσης για την γραμματική που προέκυψε ως κατάλληλη για LL(1) ανίχνευση.

**Απάντηση**

*Εάν δεν έχετε δώσει απάντηση, γράψτε με κεφαλαία γράμματα:* ***ΔΕΝ ΑΠΑΝΤΗΘΗΚΕ.***

*Εάν εν γνώση σας δίνετε ελλιπή απάντηση, γράψτε με κεφαλαία γράμματα:* ***ΕΛΛΙΠΗΣ ΑΠΑΝΤΗΣΗ.*** *Εξηγείστε σε ποιο σημείο θεωρείτε την απάντησή σας ελλιπή και γιατί****.***

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Non terminal | x | y | z | w | T | K | $ |
| S | S -> x | S -> y C |  |  |  |  |  |
| C | C -> S D | C -> S D | C -> S D | C -> S D |  |  |  |
| D |  |  | D -> z S w | D -> T K w |  |  |  |
| K |  |  |  | Κ -> ε |  |  |  |

**Ερώτημα 3.E – Ανίχνευση συμβολοσειράς**

Με χρήση του πίνακα ανίχνευσης που θα προκύψει από το ερώτημα Δ και του αλγόριθμου LL(1) ανίχνευσης να δείξετε μια top-down ανίχνευση της συμβολοσειράς **y y x z x w x y x w w** συμπληρώνοντας έναν πίνακα όπως ο παρακάτω:

| **Στοίβα (top στοιχείο δεξιά)** | **Είσοδος/lookahead token** | **Ενέργεια** |
| --- | --- | --- |
| $ S |  y y x z x w x y x w w $ | Push/Pop/Match/Error/Accept |
| ... |  |  |

**Απάντηση**

*Εάν δεν έχετε δώσει απάντηση, γράψτε με κεφαλαία γράμματα:* ***ΔΕΝ ΑΠΑΝΤΗΘΗΚΕ.***

*Εάν εν γνώση σας δίνετε ελλιπή απάντηση, γράψτε με κεφαλαία γράμματα:* ***ΕΛΛΙΠΗΣ ΑΠΑΝΤΗΣΗ.*** *Εξηγείστε σε ποιο σημείο θεωρείτε την απάντησή σας ελλιπή και γιατί****.***

| **Στοίβα (top στοιχείο δεξιά)** | **Είσοδος/lookahead token** | **Ενέργεια** |
| --- | --- | --- |
| $ S |  y y x z x w x y x w w $ | Push/Pop/Match/Error/Accept |
| $ C y | y y x z x w x y x w w $ | S -> y C |
| $ C | y x z x w x y x w w $ |  |
| $ D S | y x z x w x y x w w $ | C -> S D |
| $ D C y | y x z x w x y x w w $ | S -> y C |
| $ D C | x z x w x y x w w $ |  |
| $ D D S | x z x w x y x w w $ | C -> S D |
| $ D D x | x z x w x y x w w $ | S -> x |
| $ D D | z x w x y x w w $ |  |
| $ D w S z | z x w x y x w w $ | D -> z S w |
| $ D w S | x w x y x w w $ |  |
| $ D w x | x w x y x w w $ | S -> x |
| $ D w | w x y x w w $ |  |
| $ D | x y x w w $ |  |

**Υποδείξεις για τη συγγραφή της εργασίας**

1. Για την απάντηση της εργασίας θα πρέπει να χρησιμοποιηθεί το υπόδειγμα της εργασίας. Στο υπόδειγμα:

* Συμπληρώστε όλα τα στοιχεία με κίτρινο.
* Μην ξεχάσετε να δηλώσετε εάν η εργασία αποτελεί προϊόν αποκλειστικά δικής σας εργασίας.
* Ενσωματώστε στις απαντήσεις τα διαγράμματα και τον κώδικα που είναι αναγκαία σε κάθε ερώτημα. Δεν θα πρέπει να κάνετε παραπομπές της μορφής «βλέπε αρχείο…».
* Αν δεν έχετε απαντήσει σε ένα ερώτημα γράψτε: «**ΔΕΝ ΑΠΑΝΤΗΘΗΚΕ**».
* Αν απαντήσατε με ελλείψεις σε ένα ερώτημα γράψτε: «**ΕΛΛΙΠΗΣ ΑΠΑΝΤΗΣΗ**».

1. Η συνεργασία στην ανάλυση της εργασίας επιτρέπεται, αλλά καλό είναι να αναφερθεί στον ειδικό χώρο στην πρώτη σελίδα της εργασίας. Η συνεργασία δεν πρέπει να οδηγεί σε από κοινού επίλυση και συγγραφή της εργασίας. Η υποβολή κοινών απαντήσεων από διαφορετικούς φοιτητές που συνεργάστηκαν δεν επιτρέπεται και θεωρείται ως **ΑΝΤΙΓΡΑΦΗ**. Οι απαντήσεις ελέγχονται, τόσο μεταξύ των φοιτητών του ιδίου τμήματος, όσο και μεταξύ φοιτητών διαφορετικών τμημάτων χρησιμοποιώντας το αυτόματο σύστημα εντοπισμού λογοκλοπής Turnitin. Η αντιγραφή έχει ως αποτέλεσμα το **ΜΗΔΕΝΙΣΜΟ ΤΗΣ ΕΡΓΑΣΙΑΣ ΣΥΝΟΛΙΚΑ** (βαθμός -2) και την παραπομπή των παραβατών στην Κοσμητεία της Σχολής Θετικών Επιστημών & Τεχνολογίας, σύμφωνα με τον εσωτερικό κανονισμό του ΕΑΠ.
2. Η κατασκευή του κώδικα Java θα πρέπει να γίνει αποκλειστικά με το εργαλείο Netbeans.
3. Ο φοιτητής θα πρέπει να ανεβάσει στο study.eap.gr δύο αρχεία:
   * Ένα αρχείο που θα περιέχει την απάντηση της 4η εργασία σε αρχείο κειμένου με όνομα PLH24\_4ERG\_EPITHETO\_ONOMA.doc.
   * Ένα συμπιεσμένο αρχείου σε μορφή zip ή rar που θα περιέχει το project Netbeans. To όνομα του αρχείου θα είναι:
   * PLH24\_4ERG\_EPITHETO\_ONOMA.<rar|zip>.
     1. Να γίνει χρήση λατινικών χαρακτήρων για την αποφυγή προβλημάτων.
     2. Ο κατάλογος με τον κώδικα Java θα πρέπει να μπορεί να εκτελείται **χωρίς αλλαγές** από τον καθηγητή.
     3. O κώδικας java πρέπει να είναι **ΣΧΟΛΙΑΣΜΕΝΟΣ ΑΝΑΛΥΤΙΚΑ.**
4. Στις απαντήσεις σας μην χρησιμοποιείτε κόκκινο χρώμα.

H εφαρμογή των παραπάνω κανόνων είναι **ΥΠΟΧΡΕΩΤΙΚΗ** και βαθμολογείται σύμφωνα με το αντίστοιχο κριτήριο αξιολόγησης. Η μη εφαρμογή του πρώτου κανόνα μπορεί να οδηγήσει σε συνολική απόρριψη της εργασίας.

**ΚΑΛΗ ΕΠΙΤΥΧΙΑ!**