UNIVERSITY
of IOANNINA

Μεταφραστές Εργασία

Το πρόγραμμα που θα εξηγήσουμε παρακάτω είναι ένας μεταφραστής για τη γλώσσα προγραμματισμού CPy σύμφωνα με τις προδιαγραφές που καθορίστηκαν στο μάθημα από τον καθηγητή και την εκφώνηση.

Εδώ περιγράφεται η αρχή του **Λεκτικού Αναλυτή**. Ο Λεκτικός Αναλυτής έχει τον ρόλο να διαβάζει ένα αρχείο που λαμβάνει ως είσοδο το πρόγραμμά μας και να εμφανίζει το αρχείο με τα αποτελέσματα του λεκτικού. Ξεκινάμε με το να δίνουμε αρχικές τιμές και ονόματα στα tokens και στις δεσμευμένες λέξεις.

```
class Lex():

    def __init__(self):
        self.inputChar = f.read(1)
        self.desmeumenes =["if","elif", "else","while","print","return","main","def","#def","#int","input","int","and","or","not","global" ]
        self.stringLine =''
        self.line_number =1


    def lex(self):

        initial_str = "start"
        i = 0
        desmeumenes = ["if","elif", "else", "while","print","return","main"]
        logicArray = ["and", "or", "not"]
        categories = ["number", "identifier", "keyword", "addOp", "mulOp", "relOp", "parenthesis",
                "logicalOp","punctuation","error","assigment"]
        recognized_string = ''
```

- Η μέθοδος **__init__** είναι ο constructor της κλάσης. Εκτελείται όταν δημιουργείται ένα νέο αντικείμενο της κλάσης.

- **self.inputChar**: Διαβάζει έναν έναν τους χαρακτήρες από το αρχείο f
- **self.desmeumenes**: Είναι η λίστα με τις δεσμευμένες λέξεις, δηλαδή τις λέξεις-κλειδιά της γλώσσας προγραμματισμού που υποστηρίζεται.
- **self.stringLine**: Αρχικοποιείται ως κενή συμβολοσειρά.
- **self.line_number**: Αρχικοποιείται στην τιμή 1, κρατά τον αριθμό της τρέχουσας γραμμής που αναλύεται.

- **initial_str**: Αρχικοποιείται με την τιμή "start". Θα χρησιμοποιηθεί για να καθορίσει την αρχική κατάσταση.

- **desmeumenes**: Η λίστα που περιέχει τις δεσμευμένες λέξεις. Αυτή η λίστα είναι πιο μικρή έκδοση της λίστας που χρησιμοποιείται στο **__init__.**
- **logicArray**: Η λίστα με τους λογικούς τελεστές (and, or, not) .

- **categories**: Η λίστα με κατηγορίες tokens που μπορούν να αναγνωριστούν από τον lexer. Οι κατηγορίες περιλαμβάνουν αριθμούς, αναγνωριστικά, λέξεις-κλειδιά, τελεστές πρόσθεσης, τελεστές πολλαπλασιασμού, τελεστές σχέσεων, παρενθέσεις, λογικούς τελεστές, σημεία στίξης, λάθη και αναθέσεις.
- **recognized_string**: Αρχικοποιείται ως κενή συμβολοσειρά. Αυτή η μεταβλητή θα χρησιμοποιηθεί για την αποθήκευση της τρέχουσας αναγνωρισμένης συμβολοσειράς κατά την ανάλυση του κώδικα.

Ο λεκτικός ξεκινά από μια αρχική κατάσταση (start) και διαβάζει έναν έναν τους χαρακτήρες από το αρχείο εισόδου. Για κάθε χαρακτήρα, αποφασίζει την κατηγορία στην οποία ανήκει και ενημερώνει την κατάσταση του Lex. Μόλις ολοκληρωθεί η αναγνώριση μιας ακολουθίας χαρακτήρων (δηλαδή ενός token), δημιουργεί ένα αντικείμενο Token και το επιστρέφει. Ο Lex επίσης καταγράφει τον αριθμό της γραμμής του αρχείου για κάθε token.

Αναλυτική Περιγραφή Ανά Κατηγορία

1. **Λευκοί Χαρακτήρες και Νέες Γραμμές**
   - Οι λευκοί χαρακτήρες και οι χαρακτήρες νέας γραμμής αγνοούνται. Αν συναντηθεί νέα γραμμή (\n), ο αριθμός γραμμής (self.line_number) αυξάνεται.

2. **Αναγνωριστικά και Λέξεις-Κλειδιά**
   - Εάν ο χαρακτήρας είναι γράμμα ή διπλό εισαγωγικό ("), ξεκινά η αναγνώριση αναγνωριστικών. Συνεχίζει να διαβάζει χαρακτήρες μέχρι να μην είναι πλέον γράμμα ή αριθμός.
   - Αν το αναγνωρισμένο string είναι στη λίστα των δεσμευμένων λέξεων (desmeumenes), κατηγοριοποιείται ως λέξη-κλειδί (keyword).
   - Αν είναι στον πίνακα λογικών τελεστών (logicArray), κατηγοριοποιείται ως λογικός τελεστής (logicalOp).
   - Αλλιώς κατηγοριοποιείται ως αναγνωριστικό (identifier).

3. **Αριθμοί**
   - Αν ο χαρακτήρας είναι ψηφίο, ξεκινά η αναγνώριση αριθμών. Συνεχίζει να διαβάζει ψηφία μέχρι να συναντήσει μη ψηφιακό χαρακτήρα.
   - Το αναγνωρισμένο string κατηγοριοποιείται ως αριθμός (number).

4. **Τελεστές Πρόσθεσης και Αφαίρεσης**
   - Αν ο χαρακτήρας είναι + ή -, κατηγοριοποιείται ως τελεστής πρόσθεσης (addOp).

5. **Τελεστές Πολλαπλασιασμού και Υπολοίπου**
   - Αν ο χαρακτήρας είναι * ή %, κατηγοριοποιείται ως τελεστής πολλαπλασιασμού (mulOp).

6. **Σημεία Στίξης**
   - Αν ο χαρακτήρας είναι ; ή , ή :, κατηγοριοποιείται ως σημείο στίξης (punctuation).

7. **Παρενθέσεις**
   - Αν ο χαρακτήρας είναι (ή) ,{ή}, κατηγοριοποιείται ως παρένθεση (parenthesis).

8. **Τελεστής Διαίρεσης**
   - Αν ο χαρακτήρας είναι /, ελέγχει τον επόμενο. Αν ο επόμενος χαρακτήρας είναι /, κατηγοριοποιείται ως mulOp. Αν όχι, κατηγοριοποιείται ως σφάλμα (error).

9. **Τελεστής Ανάθεσης και Ισότητας**
   - Αν ο χαρακτήρας είναι =, ελέγχεται αν ο επόμενος χαρακτήρας είναι επίσης = για να κατηγοριοποιηθεί ως τελεστής σχέσης (relOp). Αν όχι, κατηγοριοποιείται ως ανάθεση (assigment).

10. **Τελεστής Ανισότητας**
   - Αν ο χαρακτήρας είναι !, ελέγχεται αν ο επόμενος χαρακτήρας είναι = για να κατηγοριοποιηθεί ως τελεστής σχέσης (relOp). Αν όχι, κατηγοριοποιείται ως σφάλμα (error).

11. **Τελεστές Σχέσης (Μικρότερο και Μεγαλύτερο)**
    - Αν ο χαρακτήρας είναι < ή >, ελέγχεται αν ο επόμενος χαρακτήρας είναι = ή > για να κατηγοριοποιηθεί ως τελεστής σχέσης (relOp).

12. **Σχόλια**
    - Αν ο χαρακτήρας είναι #, ελέγχεται ο επόμενος. Αν ο επόμενος χαρακτήρας είναι { ή }, κατηγοριοποιείται ως παρένθεση (parenthesis). Αν είναι #, κατηγοριοποιείται ως σχόλιο (comment). Συνεχίζει να διαβάζει μέχρι να βρει το κλείσιμο του σχολίου (#).

Στο τέλος της διαδικασίας, δημιουργείται ένα αντικείμενο Token με την αναγνωρισμένη συμβολοσειρά (recognized_string), την κατηγορία (initial_str) και τον αριθμό γραμμής (self.line_number). Το αντικείμενο αυτό επιστρέφεται και καταγράφεται στο αρχείο lex_syn.

Class QuadPointerList:

Αντιπροσωπεύει μια τετράδα (quadruple) που χρησιμοποιείται στην ενδιάμεση αναπαράσταση του κώδικα.

- **__init__(self, op, op1, op2, op3, label):** Αρχικοποιεί τα στοιχεία της τετράδας.
- **__str__(self):** Επιστρέφει μια συμβολοσειρά που αναπαριστά την τετράδα.

Class QuadList:

Διαχειρίζεται μια λίστα από τετράδες.

- **__init__(self):** Αρχικοποιεί τη λίστα τετράδων και άλλες μεταβλητές.
- **newTemp(self):** Δημιουργεί και επιστρέφει μία νέα προσωρινή μεταβλητή
- **emptyList():** Δημιουργεί μία κενή λίστα ετικετών τετράδων

- **makeList(self, label):** Δημιουργεί μια λίστα με ένα μοναδικό label.
- **mergeList(list1, list2):** Συνενώνει δύο λίστες.
- **genQuad(self, op, op1, op2, op3):** Δημιουργεί την επόμενη τετράδα και την προσθέτει στη λίστα.
- **nextQuad(self):** Επιστρέφει τον αριθμό της επόμενης τετράδας που
- πρόκειται να παραχθεί
- **backpatch(self, my_list, label):** Ενημερώνει την τετράδα στη λίστα με την αντίστοιχη ετικέτα.

Κλάσεις για τη Διαχείριση των Scopes

Class Table:

Διαχειρίζεται έναν πίνακα που περιέχει τα scopes.

- **__init__(self):** Αρχικοποιεί μια κενή λίστα scopes.
- **addScope(self, scope):** Προσθέτει ένα scope στη λίστα.
- **deleteScope(self):** Αφαιρεί το τελευταίο scope από τη λίστα.

Class Scope:

Αντιπροσωπεύει ένα συγκεκριμένο επίπεδο scope.

- **__init__(self, level):** Αρχικοποιεί το επίπεδο του scope και μια λίστα οντοτήτων.
- **addEntity(self, entity):** Προσθέτει μια οντότητα στο scope.
- __str__(self): Επιστρέφει μια συμβολοσειρά που αναπαριστά το scope και τις οντότητές του.

Κλάσεις για Οντότητες

Class Entity:

Η βασική κλάση για όλες τις οντότητες που μπορούν να υπάρχουν σε ένα scope.

- **__init__(self, name, level):** Αρχικοποιεί το όνομα και το επίπεδο της οντότητας.
- **__str__(self):** Επιστρέφει το όνομα της οντότητας.

Class Variable:

Κληρονομεί από την Entity και προσθέτει πληροφορίες για μεταβλητές.

- **__init__(self, name, level, datatype, offset):** Αρχικοποιεί τις πληροφορίες για μια μεταβλητή.
- **__str__(self):** Επιστρέφει μια συμβολοσειρά που αναπαριστά τη μεταβλητή και το offset

Class Procedure:

Κληρονομεί από την Entity και προσθέτει πληροφορίες για διαδικασίες.

- **__init__(self, name, level, startingQuad, framelength):** Αρχικοποιεί τις πληροφορίες για μια διαδικασία.
- **__str__(self):** Επιστρέφει μια συμβολοσειρά που αναπαριστά τη διαδικασία.
- **update(self, framelength, startingQuad):** Ενημερώνει τα στοιχεία της διαδικασίας.
- **addparameter(self, formalParameter):** Προσθέτει μια παράμετρο στη διαδικασία.

Class FormalParameter:

Κληρονομεί από την Procedure και την Entity, και προσθέτει πληροφορίες για τυπικές παραμέτρους.

- **__init__(self, name, level, datatype, mode):** Αρχικοποιεί τις πληροφορίες για μια τυπική παράμετρο.
- **__str__(self):** Επιστρέφει μια συμβολοσειρά που αναπαριστά την παράμετρο.

Class Parameter:

Κληρονομεί από την FormalParameter και την Variable.

- **__init__(self, name, level, datatype, mode, offset):** Αρχικοποιεί τις πληροφορίες για μια παράμετρο.

Class Function:

Κληρονομεί από την Procedure και προσθέτει πληροφορίες για συναρτήσεις.

- **__init__(self, name, level, datatype, startingQuad, framelength):** Αρχικοποιεί τις πληροφορίες για μια συνάρτηση.

Class TemporaryVariable:

Κληρονομεί από την Variable.

- **__init__(self, name, level, datatype, offset):** Αρχικοποιεί τις πληροφορίες για μια προσωρινή μεταβλητή.

Class SymbolicConstant:

Κληρονομεί από την Entity και προσθέτει πληροφορίες για συμβολικές σταθερές.

- **__init__(self, name, datatype, value):** Αρχικοποιεί τις πληροφορίες για μια συμβολική σταθερά.

```
class Syn():

    def __init__(self):
        self.lexical =Lex()
        self.errorText = ''
        self.Quad = QuadList()
        self.table = Table()
        self.scope = Scope(0)
        self.table.addScope(self.scope)

    def syn_analyzer(self):
        if self.program():
            print("---success---")
        else: print("---failed---")

    def error(self,string):
        print(string)
        return string
```

- **self.lexical = Lex():** Αρχικοποιεί ένα αντικείμενο της κλάσης Lex, το οποίο είναι υπεύθυνο για τη λεκτική ανάλυση. Αυτό το αντικείμενο χρησιμοποιείται για να πάρει τα tokens από τον πηγαίο κώδικα.

- **self.errorText = '':** Αρχικοποιεί μια κενή συμβολοσειρά που θα χρησιμοποιηθεί για την αποθήκευση μηνυμάτων σφάλματος κατά τη διάρκεια της συντακτικής ανάλυσης.
- **self.Quad = QuadList():** Αρχικοποιεί ένα αντικείμενο της κλάσης QuadList, το οποίο χρησιμοποιείται για τη δημιουργία τετράδων (quadruples) για την ενδιάμεση αναπαράσταση του κώδικα.
- **self.table = Table():** Αρχικοποιεί ένα αντικείμενο της κλάσης Table, το οποίο είναι ένας πίνακας συμβόλων που διατηρεί πληροφορίες για τις μεταβλητές και τις συναρτήσεις του προγράμματος.
- **self.scope = Scope(0):** Δημιουργεί ένα αντικείμενο της κλάσης Scope με αριθμό 0, το οποίο αντιπροσωπεύει το αρχικό επίπεδο του εύρους (scope).
- **self.table.addScope(self.scope):** Προσθέτει το αρχικό scope στον πίνακα συμβόλων.

Program:

Αυτή η συνάρτηση αποτελεί την κύρια συνάρτηση του συντακτικού αναλυτή. Ξεκινάει διαβάζοντας το πρώτο token από τον λεκτικό αναλυτή και στη συνέχεια καλεί τη συνάρτηση declarations για την ανάλυση των δηλώσεων. Στη συνέχεια, ελέγχει αν ορίζονται συναρτήσεις με τις συναρτήσεις def_function και def_main_function.

Def_main_function:

Αυτή η συνάρτηση αναλαμβάνει την ανάλυση της κύριας συνάρτησης του προγράμματος. Ξεκινάει ελέγχοντας αν η πρώτη λέξη είναι #def. Στη συνέχεια, ελέγχει αν η επόμενη λέξη είναι main, που είναι η κύρια συνάρτηση του προγράμματος. Ακολουθούν διάφοροι έλεγχοι και αναλύσεις σχετικά με τη δομή της συνάρτησης, τη δήλωση μεταβλητών και τον κώδικα που ακολουθεί. Εάν όλα είναι σωστά, παράγεται η ανάλογη τετράδα μέσω της QuadList και επιστρέφεται True, διαφορετικά επιστρέφεται False.

Def_function:

Η μέθοδος εξετάζει τη δήλωση συνάρτησης, ελέγχοντας τη σωστή σύνταξη και την παρουσία όλων των απαραίτητων στοιχείων. Πιο συγκεκριμένα: ελέγχει εάν η επόμενη λέξη είναι η "def", υποδεικνύοντας την έναρξη ορισμού της συνάρτησης. Στη συνέχεια, ελέγχει αν ακολουθείται από ένα αναγνωριστικό (identifier), που είναι το όνομα της συνάρτησης. Έπειτα, ελέγχει εάν ακολουθείται από μια παρένθεση "(" και στη συνέχεια αναλύει τη λίστα παραμέτρων. Ελέγχει εάν η λίστα παραμέτρων ακολουθείται από κλείσιμο παρένθεσης ")", ελέγχει το σύμβολο ":" ,δημιουργεί ένα αντικείμενο συνάρτησης και το προσθέτει στον πίνακα συμβόλων. Τέλος, ελέγχει εάν ακολουθείται από το σύμβολο "#{" που υποδηλώνει το άνοιγμα block της συνάρτησης. Αν οι παραπάνω έλεγχοι αποτύχουν σε οποιοδήποτε στάδιο, η μέθοδος επιστρέφει False, δηλώνοντας ότι η ανάλυση της δήλωσης συνάρτησης απέτυχε.

Declarations:

Η μέθοδος χρησιμοποιείται για την ανάλυση δηλώσεων στο πρόγραμμα. Αυτή η μέθοδος επαναλαμβάνει την εκτέλεση της μεθόδου **declaration_line** μέχρι να μην επιστρέψει πλέον True, υποδεικνύοντας ότι δεν υπάρχουν περισσότερες δηλώσεις για ανάλυση. Η declaration_line ελέγχει αν η επόμενη λέξη μετά το #int ή τη λέξη global είναι έγκυρη για δήλωση μεταβλητών. Εάν ναι, συνεχίζει με την ανάλυση της λίστας αναγνωρίσεων (id_list) που ακολουθεί. Αν όλα αυτά είναι έγκυρα, επιστρέφει True, αλλιώς επιστρέφει False.

Id_list:

Η μέθοδος αναλύει λίστες από identifiers για τη δήλωση μεταβλητών ή των ορισμάτων μιας συνάρτησης. Αρχικά, ελέγχει εάν η πρώτη λέξη στη λίστα είναι ένα αναγνωριστικό (identifier). Αν ναι, τότε δημιουργεί μια μεταβλητή με το όνομα αυτό και τοποθετεί τη μεταβλητή στον πίνακα συμβόλων. Στη συνέχεια, εάν υπάρχουν περισσότερα αναγνωριστικά, τα προσθέτει επίσης στον πίνακα συμβόλων. Η μέθοδος επιστρέφει True αν όλα τα αναγνωριστικά αναλυθούν επιτυχώς και False αν αποτύχει η ανάλυση.

Statement:

Η συνάρτηση statement() αναλαμβάνει να εξετάσει τον τύπο της εντολής και να καλέσει τις αντίστοιχες υποσυναρτήσεις για την εκτέλεσή τους. Αν η εντολή είναι "identifier", "print" ή

"return", τότε καλείται η συνάρτηση simple_statement(). Αν η εντολή είναι "if" ή "while", τότε καλείται η συνάρτηση structured_statement().

Codeblock:

Η συνάρτηση codeblock() επαναλαμβάνει την εκτέλεση των εντολών. Καλεί την statement() για να εκτελέσει μια εντολή και στη συνέχεια ελέγχει εάν υπάρχουν περισσότερες εντολές.

Simple_statement:

Οι συναρτήσεις simple_statement() και structured_statement() αναλαμβάνουν να εκτελέσουν απλές και δομημένες εντολές αντίστοιχα. Στη simple_statement(), ανάλογα με τον τύπο της εντολής ("identifier", "print" ή "return"), καλούνται οι αντίστοιχες συναρτήσεις assignment_stat (), print_stat () και return_stat (). Η structured_statement() εκτελεί είτε μια εντολή "if" μέσω της συνάρτησης if_stat (), είτε μια εντολή "while" μέσω της συνάρτησης while_stat().Σε περίπτωση που η δομή του προγράμματος είναι εσφαλμένη, εκτυπώνεται ένα μήνυμα σφάλματος.

Assignment_stat:

Η συνάρτηση assignment_stat () είναι υπεύθυνη για την εκχώρηση τιμής σε μια μεταβλητή. Αρχικά ελέγχει εάν η εντολή ξεκινά με ένα αναγνωριστικό "identifier", υποδεικνύοντας τη μεταβλητή στην οποία θα εκχωρηθεί η τιμή. Στη συνέχεια ελέγχει τον τελεστή εκχώρησης "=" και τον τύπο της τιμής που θα εκχωρηθεί. Ανάλογα με τον τύπο της τιμής, δημιουργούνται αντίστοιχες τετράδες (genQuad) για την εκχώρηση της τιμής στην μεταβλητή, ενώ ελέγχονται και οι απαραίτητες παρενθέσεις για την ορθότητα της σύνταξης.

Print_stat:

Η συνάρτηση print_stat() υλοποιεί την εντολή εκτύπωσης. Ελέγχει αν η εντολή ξεκινά με τη λέξη-κλειδί "print" και συνεχίζει ελέγχοντας τις απαραίτητες παρενθέσεις και εκφράσεις. Ανάλογα με τη σύνταξη, δημιουργούνται αντίστοιχες τετράδες (quads).

Return_stat:

Η συνάρτηση return_stat () αναλαμβάνει να εκτελέσει την εντολή επιστροφής αποτελέσματος από μια συνάρτηση. Ελέγχει εάν η εντολή ξεκινά με τη λέξη "return". Ανάλογα με τη σύνταξη που ακολουθεί, δημιουργούνται τετράδες (quads) για την αντίστοιχη επιστροφή του αποτελέσματος. Σε περίπτωση που η δομή του προγράμματος είναι εσφαλμένη, εκτυπώνεται μήνυμα σφάλματος.

Expression:

Η συνάρτηση αναλύει μια έκφραση. Αρχικά, ελέγχει εάν υπάρχει προαιρετικό πρόσημο (+/-) μέσω της συνάρτησης optional_sign(). Στη συνέχεια, καλεί τη συνάρτηση term() για την ανάλυση του πρώτου όρου της έκφρασης. Μετά, μέσω μιας επανάληψης, ελέγχει εάν υπάρχει πρόσθεση ή αφαίρεση και αναλύει τους επόμενους όρους μέσω της συνάρτησης term(). Οι όροι προστίθενται ή αφαιρούνται μεταξύ τους και αποθηκεύονται σε μια νέα θέση w, η οποία χρησιμεύει ως τρέχουσα θέση της έκφρασης. Τέλος, επιστρέφεται η θέση της έκφρασης.

Term:

Η συνάρτηση αναλύει έναν όρο της έκφρασης. Αρχικά, καλεί τη συνάρτηση factor() για την ανάλυση του πρώτου παράγοντα του όρου. Στη συνέχεια, μέσω μιας επανάληψης, ελέγχει εάν υπάρχει πολλαπλασιασμός ή διαίρεση ή υπόλοιπο διαίρεσης (mod) και αναλύει τους επόμενους παράγοντες μέσω της συνάρτησης factor(). Οι παράγοντες αποθηκεύονται σε μια νέα θέση w, η οποία χρησιμεύει ως τρέχουσα θέση του όρου. Τέλος, επιστρέφεται η θέση του όρου.

If_stat:

Η συνάρτηση υλοποιεί τη λειτουργία της εντολής if-else .Πρέπει να διαχειριστούμε τις δύο λίστες που επιστρέφει η condition. Οι τετράδες που έχουν αποθηκευτεί στη λίστα true πρέπει να οδηγηθούν στο code_block(), άρα το backpatch() στο nextQuad() πρέπει να γίνει πριν την παραγωγή της πρώτης εντολής των code_block(). Το backpatch() του false πρέπει να γίνει πριν το else, αν υπάρχει else ή μετά το τέλος της if, μετά και από την τελευταία τετράδα που θα παράγεται για το if. Τοποθετούμε το backpatch() στον κανόνα ifStat, ακριβώς πριν το else, τότε αν δεν υπάρχει else, οι τελευταίες τετράδες που θα παραχθούν θα είναι από το code_block άρα το backpatch() πράγματι γίνεται μετά την τελευταία τετράδα που θα παραχθεί το if, αν υπάρχει else, τότε η επόμενη τετράδα που θα παραχθεί θα είναι από το code_block.

While_stat:

Η μέθοδος υλοποιεί την αναγνώριση και εκτέλεση της εντολής while. Αρχικά, ελέγχει εάν η τρέχουσα λέξη είναι η λέξη-κλειδί "while" .Η condition είναι μία λογική συνθήκη η οποία θα επιστρέψει τις δύο γνωστές λίστες με τα q.true και q.false. Οι ασυμπλήρωτες τετράδες της q.true πρέπει να συμπληρωθούν με την πρώτη τετράδα των code_block(), διότι εκεί πρέπει να μεταβεί ο έλεγχος όταν η συνθήκη ισχύει. Οι ασυμπλήρωτες τετράδες της q.false πρέπει να συμπληρωθούν με την πρώτη τετράδα της επόμενης εντολής, δηλαδή έξω από τον while, διότι εκεί πρέπει να μεταβεί ο έλεγχος όταν η συνθήκη δεν ισχύει. Άρα ακριβώς πριν την code_block() πρέπει να τοποθετηθεί η backpatch(), με παράμετρο το nextQuad() για τη λίστα q.true. Επίσης, ακριβώς στο τέλος, πρέπει να τοποθετηθεί η backpatch() για τη λίστα q.false. Πέρα από τη διαχείριση των δύο λιστών, για να λειτουργήσει το while χρειάζεται κάτι ακόμα. Όταν εκτελούνται οι εντολές των code_block πρέπει ο έλεγχος να μεταβαίνει στην αρχή της λογικής συνθήκης, ώστε αυτή να επανεξετάζεται. Για να μπορέσουμε να γίνει αυτό, πρέπει να γίνουν δύο πράγματα. Το πρώτο είναι, την ώρα που δημιουργείται η πρώτη τετράδα της συνθήκης, η ετικέτα να σημειώνεται. Επειδή η δημιουργία της τετράδας θα γίνει μέσα στο condition και επειδή εμείς υλοποιούμε τώρα τον κανόνα για το while, θα σημειώσουμε την τετράδα αυτή ακριβώς έξω και πριν το condition, εκτελούμενο την nextQuad(). Τέλος πρέπει να φροντίσουμε να δημιουργηθεί και η τετράδα η οποία θα κάνει το άλμα από το τέλος του code_block στην αρχή της condition, δηλαδή στο condQuad.

Idtail:

Η συνάρτηση αναλύει το τμήμα που ακολουθεί ένα αναγνωριστικό. Αρχικά, ελέγχει εάν υπάρχουν παρενθέσεις μετά το αναγνωριστικό. Αν υπάρχουν παρενθέσεις, τότε γίνεται ανάλυση τους με τη χρήση της συνάρτησης
actual_par_list(). Στη συνέχεια, δημιουργείται μια νέα προσωρινή μεταβλητή variable και δημιουργούνται τετράδες κώδικα για την αποθήκευση των παραμέτρων και την κλήση της συνάρτησης με το συγκεκριμένο όνομα. Τέλος, γίνεται έλεγχος για την παρουσία της δεξιάς παρένθεσης.

Actual_par_list:

Η συνάρτηση επεξεργάζεται μια λίστα πραγματικών παραμέτρων . Αρχικά, αναλύει μια έκφραση με τη χρήση της συνάρτησης expression και δημιουργεί μια τετράδα μέσω της genQuad. Στη συνέχεια, επεξεργάζεται διαδοχικά τις επόμενες παραμέτρους εάν υπάρχουν, διαβάζοντας τα επόμενα σύμβολα και ακολουθώντας την ίδια διαδικασία επεξεργασίας της έκφρασης . Επιστρέφει True αν η διαδικασία ολοκληρωθεί με επιτυχία, ή False εάν κάποια παράμετρος δεν μπορεί να αναλυθεί σωστά.

Optional_sign:

Η συνάρτηση ελέγχει εάν υπάρχει προαιρετικό πρόσημο σε μια έκφραση. Εάν το τρέχον σύμβολο είναι τελεστής πρόσθεσης ή αφαίρεσης, αναγνωρίζει το σύμβολο και επιστρέφει True. Διαφορετικά, επιστρέφει επίσης True, χωρίς να αναμένει τον τελεστή.

Condition:

Η συνάρτηση αναλαμβάνει την αξιολόγηση λογικών προτάσεων που περιλαμβάνουν λογικούς τελεστές OR. Για κάθε όρο στην πρόταση, καλεί τη συνάρτηση bool_term() για να τον αξιολογήσει και στη συνέχεια συγχωνεύει τις λίστες true και false που προκύπτουν .

Bool_term:

Η συνάρτηση χειρίζεται την αξιολόγηση λογικών προτάσεων που περιλαμβάνουν τελεστές AND. Για κάθε παράγοντα στην πρόταση, καλεί τη συνάρτηση bool_factor() για να τον αξιολογήσει και στη συνέχεια συγχωνεύει τις λίστες true και false.

Bool_factor:

Η συνάρτηση είναι υπεύθυνη για την αξιολόγηση ενός λογικού παράγοντα, ο οποίος μπορεί να αποτελείται από μια λογική πρόταση, έναν αριθμό ή έναν συνδυασμό τους. Κατά την αξιολόγηση, αρχικά ελέγχει εάν ο παράγοντας ξεκινά με τον τελεστή not. Αν ναι, τότε αντιστρέφει το αποτέλεσμα της εσωτερικής λογικής πρότασης που βρίσκεται εντός παρενθέσεων.
Αξιολογεί τις εκφράσεις, δημιουργεί τετράδες για την εκτέλεση της σύγκρισης και επιστρέφει δύο λίστες: μία με τις τιμές true και μία με τις τιμές false που αποτελούνται από τους αντίστοιχους δείκτες τετράδων.

Υλοποιήση Test

Το παρακάτω τεστ περιλαμβάνει εμφωλευμένες κλήσεις def, δομή if, κλήση global και τη δεσμευμένη #int.

```
def quad(x):
#{

    #int g
    global t
    ## nested function sqr ##
    def sqr(x):
    #{
        ## body of sqr ##

        if (x != 0):
            return (-1);

        else:
            return (fib(x-1)+fib(x-2));
    #}

    ## body of quad ##

    counterFunctionCalls = counterFunctionCalls + 1;
    y = sqr(x)*sqr(x);
    return (y);


#}
```

Παρακάτω φαίνονται τα αποτελέσματα του λεκτικού και του συντακτικού αναλυτή:

```
Recognize : [def], Type : [identifier], Line : [1]
Recognize : [quad], Type : [identifier], Line : [1]
Recognize : [(], Type : [parenthesis], Line : [1]
Recognize : [x], Type : [identifier], Line : [1]
Recognize : [)], Type : [parenthesis], Line : [1]
Recognize : [:], Type : [punctuation], Line : [1]
Recognize : [#{], Type : [parenthesis], Line : [2]
Recognize : [#int], Type : [error], Line : [4]
Recognize : [g], Type : [identifier], Line : [4]
Recognize : [global], Type : [identifier], Line : [5]
Recognize : [t], Type : [identifier], Line : [5]
Recognize : [def], Type : [identifier], Line : [7]
Recognize : [sqr], Type : [identifier], Line : [7]
Recognize : [(], Type : [parenthesis], Line : [7]
Recognize : [x], Type : [identifier], Line : [7]
Recognize : [)], Type : [parenthesis], Line : [7]
Recognize : [:], Type : [punctuation], Line : [7]
Recognize : [#{], Type : [parenthesis], Line : [8]
Recognize : [if], Type : [keyword], Line : [11]
Recognize : [(], Type : [parenthesis], Line : [11]
Recognize : [x], Type : [identifier], Line : [11]
Recognize : [!=], Type : [relOp], Line : [11]
Recognize : [0], Type : [number], Line : [11]
Recognize : [)], Type : [parenthesis], Line : [11]
Recognize : [:], Type : [punctuation], Line : [11]
Recognize : [return], Type : [keyword], Line : [12]
Recognize : [(], Type : [parenthesis], Line : [12]
Recognize : [-], Type : [addOp], Line : [12]
Recognize : [1], Type : [number], Line : [12]
Recognize : [)], Type : [parenthesis], Line : [12]
Recognize : [;], Type : [punctuation], Line : [12]
Recognize : [else], Type : [keyword], Line : [14]
Recognize : [:], Type : [punctuation], Line : [14]
Recognize : [return], Type : [keyword], Line : [15]
Recognize : [(], Type : [parenthesis], Line : [15]
Recognize : [fib], Type : [identifier], Line : [15]
Recognize : [(], Type : [parenthesis], Line : [15]
Recognize : [x], Type : [identifier], Line : [15]
Recognize : [-], Type : [addOp], Line : [15]
Recognize : [1], Type : [number], Line : [15]
Recognize : [)], Type : [parenthesis], Line : [15]
Recognize : [+], Type : [addOp], Line : [15]
Recognize : [fib], Type : [identifier], Line : [15]
```

```
Recognize : [(], Type : [parenthesis], Line : [15]
Recognize : [x], Type : [identifier], Line : [15]
Recognize : [-], Type : [addOp], Line : [15]
Recognize : [2], Type : [number], Line : [15]
Recognize : [)], Type : [parenthesis], Line : [15]
Recognize : [)], Type : [parenthesis], Line : [15]
Recognize : [;], Type : [punctuation], Line : [15]
Recognize : [#}], Type : [parenthesis], Line : [16]
Recognize : [counterFunctionCalls], Type : [identifier], Line : [20]
Recognize : [=], Type : [assigment], Line : [20]
Recognize : [counterFunctionCalls], Type : [identifier], Line : [20]
Recognize : [+], Type : [addOp], Line : [20]
Recognize : [1], Type : [number], Line : [20]
Recognize : [;], Type : [punctuation], Line : [20]
Recognize : [y], Type : [identifier], Line : [21]
Recognize : [=], Type : [assigment], Line : [21]
Recognize : [sqr], Type : [identifier], Line : [21]
Recognize : [(], Type : [parenthesis], Line : [21]
Recognize : [x], Type : [identifier], Line : [21]
Recognize : [)], Type : [parenthesis], Line : [21]
Recognize : [*], Type : [mulOp], Line : [21]
Recognize : [sqr], Type : [identifier], Line : [21]
Recognize : [(], Type : [parenthesis], Line : [21]
Recognize : [x], Type : [identifier], Line : [21]
Recognize : [)], Type : [parenthesis], Line : [21]
Recognize : [;], Type : [punctuation], Line : [21]
Recognize : [return], Type : [keyword], Line : [22]
Recognize : [(], Type : [parenthesis], Line : [22]
Recognize : [y], Type : [identifier], Line : [22]
Recognize : [)], Type : [parenthesis], Line : [22]
Recognize : [;], Type : [punctuation], Line : [22]
Recognize : [#}], Type : [parenthesis], Line : [25]
Recognize : [], Type : [error], Line : [26]
```

Δημιουργείται ένα αντικείμενο Token με την αναγνωρισμένη συμβολοσειρά, την κατηγορία και τον αριθμό γραμμής.

```
0 : begin_block ,sqr , _ ,_
1 : != ,x , 0 ,_
2 : jump ,_ , _ ,_
3 : retv ,1 , _ ,_
4 : jump ,_ , _ ,_
5 : - ,x , 1 ,T_0
6 : par ,T_0 , CV ,_
7 : par ,T_1 , ret ,_
8 : call ,_ , _ ,fib
9 : - ,x , 2 ,T_2
10 : par ,T_2 , CV ,_
11 : par ,T_3 , ret ,_
12 : call ,_ , _ ,fib
13 : + ,T_1 , T_3 ,T_4
14 : retv ,T_4 , _ ,_
15 : end_block ,sqr , _ ,_
16 : begin_block ,quad , _ ,_
17 : + ,counterFunctionCalls , 1 ,T_5
18 : = ,T_5 , _ ,counterFunctionCalls
19 : par ,x , CV ,_
20 : par ,T_6 , ret ,_
21 : call ,_ , _ ,sqr
22 : par ,x , CV ,_
23 : par ,T_7 , ret ,_
24 : call ,_ , _ ,sqr
25 : * ,sqr , sqr ,T_8
26 : = ,T_8 , _ ,y
27 : retv ,y , _ ,_
28 : end_block ,quad , _ ,_
```

Ο **ενδιάμεσος κώδικας** που παρέχεται αναπαριστά τις λειτουργίες των συναρτήσεων sqr και quad του αρχικού κώδικα. Ξεκινώντας με τη συνάρτηση sqr, ο κώδικας αρχίζει με την εντολή begin_block που δηλώνει την αρχή του μπλοκ της συνάρτησης. Ακολουθεί ο έλεγχος αν x != 0, και αν η συνθήκη είναι αληθής, εκτελείται άλμα με την εντολή jump που οδηγεί στην εντολή retv 1 για την επιστροφή της τιμής -1. Αν η συνθήκη δεν είναι αληθής, υπολογίζεται x - 1 και το αποτέλεσμα αποθηκεύεται στην προσωρινή μεταβλητή T_0. Η T_0 περνάει ως παράμετρος στη συνάρτηση fib, και η επιστροφή της τιμής της αποθηκεύεται στην T_1. Στη συνέχεια, υπολογίζεται x - 2 και το αποτέλεσμα αποθηκεύεται στην T_2, η οποία περνάει επίσης ως παράμετρος στη fib, και η επιστροφή της τιμής της αποθηκεύεται στην T_3. Οι τιμές T_1 και T_3 προστίθενται και το αποτέλεσμα αποθηκεύεται στην T_4, η οποία επιστρέφεται ως η τελική τιμή της sqr. Η εντολή end_block δηλώνει το τέλος του μπλοκ της συνάρτησης. Στη συνάρτηση quad, το μπλοκ αρχίζει με την εντολή begin_block, και αυξάνεται η τιμή του counterFunctionCalls κατά 1, με το αποτέλεσμα να αποθηκεύεται στην T_5 και να ανατίθεται πάλι στο counterFunctionCalls. Η συνάρτηση sqr καλείται δύο φορές με το x ως παράμετρο, και οι τιμές επιστροφής αποθηκεύονται στις T_6 και T_7. Το τετράγωνο της τιμής επιστροφής της sqr υπολογίζεται και αποθηκεύεται στην T_8, η οποία ανατίθεται στη μεταβλητή y και επιστρέφεται ως η τελική τιμή της quad. Η εντολή end_block δηλώνει το τέλος του μπλοκ της συνάρτησης quad.

```
scope 0:  x/12  name = quad/0
scope 1:  g/12  t/16  x/20  name = sqr/0
scope 2:  T_0/12  T_1/16  T_2/20  T_3/24  T_4/28
scope 0:  x/12  name = quad/0
scope 1:  g/12  t/16  x/20  name = sqr/32 T_5/24  T_6/28  T_7/32  T_8/36
```

Ο πίνακας συμβόλων του κώδικα περιλαμβάνει τις μεταβλητές και τις συναρτήσεις με τις αντίστοιχα offset . **Στο Scope 0**, η μεταβλητή x έχει **offset** 12 και η συνάρτηση quad είναι δηλωμένη. **Στο Scope 1**, που είναι μέσα στη quad, η g έχει **offset** 12, η global μεταβλητή t έχει **offset** 16, η παράμετρος x της sqr έχει **offset** 20, και η συνάρτηση sqr είναι δηλωμένη. **Στο Scope 2**, που είναι μέσα στη sqr, οι προσωρινές μεταβλητές T_0, T_1, T_2, T_3, και T_4 έχουν **offset** 12, 16, 20, 24, και 28 αντίστοιχα. **Στο Scope 1** ξανά, η g έχει **offset** 12, η t έχει **offset** 16, η x της sqr έχει **offset** 20, η sqr είναι δηλωμένη με **framelength** 32, και οι προσωρινές μεταβλητές T_5, T_6, T_7, και T_8 έχουν **offset** 24, 28, 32, και 36 αντίστοιχα.