Βάρσος Βασίλειος	4323
Τζώρτζης Ραφαήλ	4503

MYY802 PROJECT REPORT

Εισαγωγή:

Στο παρακάτω report θα αναλύσουμε την λειτουργικότητα του μεταφραστή μας βάση των προσχεδιαστικών απαιτήσεων του μαθήματος. Τα επιμέρους κομμάτια του project που υλοποιήθηκαν είναι ο λεκτικός και συντακτικός αναλυτής, η παραγωγή ενδιάμεσου κώδικα καθώς και κάποια εισαγωγικά κομμάτια του πίνακα συμβόλων. Η δομή του report που θα ακολουθηθεί θα είναι μια γρήγορη εισαγωγή στον σχεδιασμό της κάθε επιμέρους φάσης στην οποία και θα επεξηγήσουμε την δομή του κώδικα καθώς και κάποιες σχεδιαστικές αποφάσεις οι οποίες πάρθηκαν, καθώς και πράγματα τα οποία δοκιμάσαμε κατα την διάρκεια της υλοποίησης αλλά δεν είχαν την επιθυμητή λειτουργικότητα. Έπειτα, θα παραθέσουμε τα test τα οποία τρέξαμε στον κώδικα μας και θα παραθέσουμε έναν μικρό σχολιασμό για το καθένα.

Λεκτικός Αναλυτής:

Ο λεκτικός αναλυτής μας καλείται ως συνάρτηση από τον συνακτικό αναλυτή και διαβάζει γράμμα-γράμμα το αρχικό πρόγραμμα, επιστρέφοντας σε κάθε κλήση του την επόμενη λεκτική μονάδα – token – σύμφωνα με τις προδιαγραφές τις εκφώνησης. Αν και η δομή ενός State Machine είναι στα μάτια μας πιο κομψή, επιλέξαμε να τον υλοποιήσουμε με χρήση πολλαπλών if statements λόγω της ευκολίας στην υλοποίηση του. Έτσι, η υλοποίηση μας ελέγχει τον χαρακτήρα που διαβάζεται και επιστρέφει το αντίστοιχο token.

Συντακτικός Αναλυτής:

Η δομή του συντακτικού αναλυτή είναι μια που μας απασχόλησε αρκετά κατά την διάρκεια της υλοποίησης της εργασίας. Σε πρώτο στάδιο, είχαμε υλοποιήσει μια βοηθητική συνάρτηση την match, η οποία έλεγχε εαν το input της συνάρητσης (το token δηλαδή που θέλαμε να κάνουμε match), αντιστοιχούσε σε αυτό που μας επέστρεφε ο λεκτικός αναλυτής. Έπειτα, η συνάρτηση μας καλούσε την get_next_token() και ενμημερωνόταν έτσι το token το οποίο ελέγχαμε κάθε φορα. Σε περίπτωση σφάλματος, η match θα γυρνούσε και το αντίστοιχο error message βάση του input που δεχόταν.

Η συνάρτηση αυτή όμως προκάλεσε πολλά προβλήματα κατά το testing του κώδικα και πολλές φορές δεν αναγνώριζε σωστά τα tokens, είτε δυσκόλευε την υλοποίηση παρά να την διευκολύνει, καθώς αρκετές φορές έπρεπε να ελέγξουμε το family του token αντί για το recognized_string. Στην προσπάθεια μας να κάνουμε adapt την βοηθητική μας συνάρτηση κατέληξε να περιπλέκει την υλοποίηση αρκετά, οπότε επιλέξαμε να την αφαιρέσουμε και αντί αυτού ο έλεγχος γίνεται για άλλη μια φορά με την μορφή πολλαπλών if statements, όπου βάση της γραμματικής της γλώσσας γίνεται η συνακτική ανάλυση.

Παραγωγή ενδιάμεσου κώδικα:

Για την παραγωγή ενδιάμεσου κώδικα υλοποιήσαμε την κλάση Quad βάση των υποδείξεων στις αντίστοιχες σημειώσεις. Έπειτα, στις επιμέρους συναρτήσεις του συνακτικού αναλυτή δημιουργούμε τις αντίστοιχες λίστες (επιμέρους και τελικές) για τα quads και τις επιστρέφουμε στις συναρτήσεις που χρειάζεται για να γίνουν τα αντίστοιχα backpatch. Δυστυχώς δεν έχει υλοποιηθεί η

εκτύπωση των quads στο τέλος, υπάρχουν όμως τα αντίστοιχα prints για τον έλεγχο της σωστής δημιουργίας τους.

Πίνακας Συμβόλων:

Στο κομμάτι του πίνακα συμβόλων έχουν υλοποιηθεί μόνο οι κλάσεις που δίνονται στις σημειώσεις

Έλεγχος του Κώδικα:

Για τον έλεγχο της λειτουργικότητας του κώδικα χρησιμοποιήσαμε τις συναρτήσεις που δίνονταν μαζί με την επεξήγηση της γλώσσας cutePy. Για την καταγραφή της ροής του κώδικα όσο τρέχει χρησιμοποιήσαμε την βιβλιοθήκη logging για να τυπώνει μηνύματα στο τερματικό. Εφόσον ο κώδικας μας δεν τρέχει, μέσω της logging μπορούμε να δούμε μέχρι που λειτουργεί σωστά και να επαληθεύσουμε την λειτουργία του.

```
def main_factorial():
#{
    #$ declarations #$
    #declare x
    #declare i,fact
    #$ body of main_factorial #$
    x = int(input());
    fact = 1;
    i = 1;
    while (i<=x):
    #{
        fact = fact * i;
        i = i + 1;
    #}
    print(fact);
#}

if __name__ == "__main__":
    #$ call of main function #$
    main_factorial();
```

Το 1ο test είναι η main_fibonacci με την οποία ελέγχουμε αρκετές απο τις λειτουργικότητες του κώδικα.

Άρχικα, παρατηρούμε ότι με το που διαβάζεται η πρώτη λέξη – το “def” -, μπαίνουμε κατευθείαν στις αρχικές συναρτήσεις και δημιουργούμε το 1ο quad του begin block. Παρατηρούμε πως επιστρέφονται τα tokens κανονικά μέχρι και την εύρεση του σχολίου, όπου υπάρχει το πρόβλημα πως το recognized_string του comment που επιστρέφεται κρατάει και το «#» του GroupSymbol που ορίζει το τέλος του comment ενώ δεν θα έπρεπε. Ο λόγος που δημιουργείται το σφάλμα αυτό είναι πως η while που διαβάζει το comment χαρακτήρα χαρακτήρα καλείται μια τελευταία φορά που βρίσκει και το τέλος του comment, ενημερώνει όμως το string με το «#», δεν επηρρεάζεται όμως κάπως η λειτουργία του μεταφραστή καθώς θα δούμε παρακάτω κιόλας πως είναι μόνο στα σχόλια το σφάλμα αυτό.

```
DEBUG:root:...Found Family Keywords in .def.
DEBUG:root:...Went in on Start Rule with current token def
DEBUG:root:...Went in on def_main_part with current token def
DEBUG:root:...Went in on def_main_function with current token def
DEBUG:root:

DEBUG:root:...Found Family ID in .main_factorial.
DEBUG:root:...Matching ID with current token main_factorial
DEBUG:root:...nextQuad called. The next generated quad will have no. 2
DEBUG:root:Generating quad 1: begin_block, main_factorial, _, _
DEBUG:root:Found a GroupSymbol in: (
DEBUG:root:...Matching GroupSymbols with current token (
DEBUG:root:Found a GroupSymbol in: )
DEBUG:root:...Matching GroupSymbols with current token )
DEBUG:root:Found a Delimiter in: :
DEBUG:root:...Matching Delimiters with current token :
DEBUG:root:

DEBUG:root:...Found #
DEBUG:root:...Found a GroupSymbol in: #{
DEBUG:root:...Found start of block with current token #{
DEBUG:root:

DEBUG:root:

DEBUG:root:...Found #
DEBUG:root:...Found start of comment
DEBUG:root:...Found end of comment, the comment was:  declarations #
DEBUG:root:

DEBUG:root:

DEBUG:root:...Found #
DEBUG:root:...Found declaration with current string .declare.
DEBUG:root:...Went in on declarations
DEBUG:root:...Parsing through the declaration line
DEBUG:root:

DEBUG:root:...Found Family ID in .x.
DEBUG:root:...Went in on id_list
DEBUG:root:...Matching ID
DEBUG:root:

DEBUG:root:

DEBUG:root:...Found #
DEBUG:root:...Found declaration with current string .declare.
DEBUG:root:...Parsing through the declaration line
DEBUG:root:

DEBUG:root:...Found Family ID in .i.
DEBUG:root:...Went in on id_list
DEBUG:root:...Matching ID
DEBUG:root:Found a Delimiter in: ,
DEBUG:root:...Found Family ID in .fact.
```

Έπειτα παρατηρούμε πως ο μεταφραστής μας περνάει κανονικά από τα declarations καθώς και τα assignments δημιουργώντας το quad για το input. Κάπου εδώ επιβεβαιώνουμε και την λειτουργία της genquad καθώς δημιουργεί ορθά τα quads ενημερώνοντας το quad number.

```
DEBUG:root:...Found #
DEBUG:root:...Found start of comment
DEBUG:root:...Found end of comment, the comment was:  body of main_factorial #
DEBUG:root:

DEBUG:root:

DEBUG:root:...Found Family ID in .x.
DEBUG:root:...Went in on statements
DEBUG:root:...Trying to understand if its a simple or structured statement
DEBUG:root:...Simple Statement found
DEBUG:root:...Found ID with current token x
DEBUG:root:...Assiginment function detected, matching ID with current token x
DEBUG:root:

DEBUG:root:Found a = in: =
DEBUG:root:...Matching Assignment with current token =
DEBUG:root:...Found Family ID in .int.
DEBUG:root:Found a GroupSymbol in: (
DEBUG:root:...Found Family Keywords in .input.
DEBUG:root:Found a GroupSymbol in: (
DEBUG:root:Found a GroupSymbol in: )
DEBUG:root:Found a GroupSymbol in: )
DEBUG:root:Found a Delimiter in: ;
DEBUG:root:...nextQuad called. The next generated quad will have no. 3
DEBUG:root:Generating quad 2: in, input, _, _
DEBUG:root:

DEBUG:root:

DEBUG:root:...Found Family ID in .fact.
DEBUG:root:...Trying to understand if its a simple or structured statement
DEBUG:root:...Simple Statement found
DEBUG:root:...Found ID with current token fact
DEBUG:root:...Assiginment function detected, matching ID with current token fact
DEBUG:root:

DEBUG:root:Found a = in: =
DEBUG:root:...Matching Assignment with current token =
DEBUG:root:Found an Integer in: .1.
DEBUG:root:Found a Delimiter in: ;
DEBUG:root:

DEBUG:root:

DEBUG:root:...Found Family ID in .i.
DEBUG:root:...Trying to understand if its a simple or structured statement
DEBUG:root:...Simple Statement found
DEBUG:root:...Found ID with current token i
DEBUG:root:...Assiginment function detected, matching ID with current token i
DEBUG:root:

DEBUG:root:Found a = in: =
DEBUG:root:...Matching Assignment with current token =
DEBUG:root:Found an Integer in: .1.
DEBUG:root:Found a Delimiter in: ;
DEBUG:root:

DEBUG:root:

DEBUG:root:...Found Family Keywords in .while.
DEBUG:root:...Trying to understand if its a simple or structured statement
```

Έπειτα, στην while που διαβάζεται ο μεταφραστής μας αναγνωρίζει ορθά το condition δημιουργώντας το αντίστοιχο quad, δημιουργείται επίσης και το jump quad σε περίπτωση που δεν ισχύει η συνθήκη. Εδώ έχει γίνει το σφάλμα πως το jump quad έχει λανθασμένο quad number καθώς δεν καλούνται σωστά τα ενδιάμεσα quads της while.

```
DEBUG:root:...Found Family Keywords in .while.
DEBUG:root:...Trying to understand if its a simple or structured statement
DEBUG:root:...Structured Statement found
DEBUG:root:...Found while function with current token while
DEBUG:root:...Went in on while_stat with current token while
DEBUG:root:...nextQuad called. The next generated quad will have no. 4
DEBUG:root:

DEBUG:root:Found a GroupSymbol in: (
DEBUG:root:...Found Family ID in .i.
DEBUG:root:...Went in on condition with current token i
DEBUG:root:...Went in on bool_term with current token i
DEBUG:root:...Went in on bool_factor with current token i
DEBUG:root:...Went in on expression with current token i
DEBUG:root:...Went in on optional_sign with current token i
DEBUG:root:...No sign found, returning
DEBUG:root:...Went in on term with current token i
DEBUG:root:...Went in on factor with current token i
DEBUG:root:...Went in on idtail with current token i
DEBUG:root:...Returning term i
DEBUG:root:...Found term i
DEBUG:root:...Returning expression i
DEBUG:root:Found a RelOperator: <
DEBUG:root:...Found a RelOperator: <= , getting next token and finding next expression
DEBUG:root:...Found Family ID in .x.
DEBUG:root:...Went in on expression with current token x
DEBUG:root:...Went in on optional_sign with current token x
DEBUG:root:...No sign found, returning
DEBUG:root:...Went in on term with current token x
DEBUG:root:...Went in on factor with current token x
DEBUG:root:...Went in on idtail with current token x
DEBUG:root:...Returning term x
DEBUG:root:...Found term x
DEBUG:root:...Returning expression x
DEBUG:root:...nextQuad called. The next generated quad will have no. 5
DEBUG:root:...nextQuad called. The next generated quad will have no. 6
DEBUG:root:Generating quad 5: <=, i, x, _
DEBUG:root:...nextQuad called. The next generated quad will have no. 7
DEBUG:root:...nextQuad called. The next generated quad will have no. 8
DEBUG:root:Generating quad 7: jump, _, _, _
DEBUG:root:Found a GroupSymbol in: )
DEBUG:root:...nextQuad called. The next generated quad will have no. 9
Traceback (most recent call last):
```

Και κάπου εδώ το πρόγραμμα μας σταματάει να τρέχει επιστρέφοντας αυτό το error. Το πρόβλημα που υποθέτουμε πως έχει δημιουργήθει είναι πως η BTrue περιέχει 2 στοιχεία τα οποία περνάνε ως ορίσματα, το οποίο σημαίνει πως λογικά δεν επεξεργάζονται καλά οι ενδιάμεσοι πίνακες που επιστρέφονται απο τις συναρτήσεις που υλοποιούν τα booleans.

```
  File "cutePy_4323_4503.py", line 665, in while_stat
    self.quad.backpatch(BTrue,self.quad.nextQuad())
TypeError: backpatch() takes 2 positional arguments but 3 were given
```

Ας πάμε στο 2ο test της main_fibonacci.

```
def main_fibonacci():
#{
#declare x
def fibonacci(x):
#{
if (x<=1):
return(x);
else:
return (fibonacci(x-1)+fibonacci(x-2));
#}
x = int(input());
print(fibonacci(x));
#}

if __name__ == "__main__":
    #$ call of main function #$
    main_fibonacci();
```

Παρατηρούμε απο το 1ο screenshot του τερματικού πως και σε αυτό το test έχουμε ορθή υλοποίηση της συντακτικής ανάλυσης καθ'ολη τη διάρκεια της λειτουργίας του προγράμματος και δημιουργούνται τα απαραίτητα quads μέχρι και την εύρεση του if statement.

```
DEBUG:root:...Found Family Keywords in .def.
DEBUG:root:...Went in on Start Rule with current token def
DEBUG:root:...Went in on def_main_part with current token def
DEBUG:root:...Went in on def_main_function with current token def
DEBUG:root:

DEBUG:root:...Found Family ID in .main_fibonacci.
DEBUG:root:...Matching ID with current token main_fibonacci
DEBUG:root:...nextQuad called. The next generated quad will have no. 2
DEBUG:root:Generating quad 1: begin_block, main_fibonacci, _, _
DEBUG:root:Found a GroupSymbol in: (
DEBUG:root:...Matching GroupSymbols with current token (
DEBUG:root:Found a GroupSymbol in: )
DEBUG:root:...Matching GroupSymbols with current token )
DEBUG:root:Found a Delimiter in: :
DEBUG:root:...Matching Delimiters with current token :
DEBUG:root:

DEBUG:root:...Found #
DEBUG:root:...Found a GroupSymbol in: #{
DEBUG:root:...Found start of block with current token #{
DEBUG:root:

DEBUG:root:...Found #
DEBUG:root:...Found declaration with current string .declare.
DEBUG:root:...Went in on declarations
DEBUG:root:...Parsing through the declaration line
DEBUG:root:

DEBUG:root:...Found Family ID in .x.
DEBUG:root:...Went in on id_list
DEBUG:root:...Matching ID
DEBUG:root:

DEBUG:root:...Found Family Keywords in .def.
DEBUG:root:...Found new function with current token  def
DEBUG:root:...Matching Keywords
DEBUG:root:

DEBUG:root:...Found Family ID in .fibonacci.
DEBUG:root:...Matching ID
DEBUG:root:...nextQuad called. The next generated quad will have no. 3
DEBUG:root:Generating quad 2: begin_block, fibonacci, _, _
DEBUG:root:Found a GroupSymbol in: (
DEBUG:root:...Matching GroupSymbols
DEBUG:root:...Found Family ID in .x.
DEBUG:root:...Went in on id_list
DEBUG:root:...Matching ID
DEBUG:root:Found a GroupSymbol in: )
DEBUG:root:...Matching GroupSymbols
DEBUG:root:Found a Delimiter in: :
DEBUG:root:...Matching Delimiters
DEBUG:root:

DEBUG:root:...Found #
DEBUG:root:...Found a GroupSymbol in: #{
DEBUG:root:...Matching GroupSymbols
DEBUG:root:

DEBUG:root:...Found Family Keywords in .if.
DEBUG:root:...Went in on declarations
```

Το error που παρατηρούμε εδώ όμως έρχεται παρακάτω, όπου διαβάζοντας την if μπαίνουμε στην statements, και ενώ στην while αναγνωριζόταν κανονικά πως είχαμε structured statement, δεν αναγνωρίζει την if, κι ας διαβάζει σωστά το keyword. Έτσι, δεν έχουμε εικόνα για το τι συμβαίνει μετέπειτα. Με το test αυτό όμως θα μπορούσαμε να ελέγξουμε τις περιπτώσεις της return και της print, οι οποίες και δεν έχουν ελεγχθει.

```
DEBUG:root:...Found Family Keywords in .if.
DEBUG:root:...Went in on declarations
DEBUG:root:...Went in on statements
DEBUG:root:...Trying to understand if its a simple or structured statement
Traceback (most recent call last):
  File "cutePy_4323_4503.py", line 1004, in <module>
    main()
  File "cutePy_4323_4503.py", line 996, in main
    result = parser.parse()
  File "cutePy_4323_4503.py", line 295, in parse
    self.start_rule()
  File "cutePy_4323_4503.py", line 302, in start_rule
    self.def_main_part()
  File "cutePy_4323_4503.py", line 308, in def_main_part
    self.def_main_function()
  File "cutePy_4323_4503.py", line 339, in def_main_function
    self.def_function()
  File "cutePy_4323_4503.py", line 392, in def_function
    self.statements()
  File "cutePy_4323_4503.py", line 427, in statements
    self.statement()
  File "cutePy_4323_4503.py", line 433, in statement
    if self.current_token.recognized_string in ["print", "return"] or self.current_token.family == "ID":
AttributeError: 'NoneType' object has no attribute 'recognized_string'

C:\Users\varso\source\repos\cutePy_4323_4503>
```

Έπειτα πηγαίνουμε στο test no.3 της count_digits, η οποία επι το πλείστον ελέγχει τα ίδια με τα προηγούμενα tests, οπότε θα πάμε κατευθείαν στο κομμάτι που επιβεβαιώσαμε επιπλέον λειτουργικότητες και σφάλματα.

```
def main_countdigits():
#{
#declare x, count
x = int(input());
count = 0;
while (x>0):
#{
x = x // 10;
count = count + 1;
#}
print(count);
#}

if __name__ == "__main__":
    #$ call of main function #$
    main_countdigits();
```

Υπάρχει μια συγκεκριμένη αλληλουχία κλήσεων των επιμέρους συναρτήσεων η οποία παρατηρείται μέσα στην while, και δεν ανενώνεται ορθά το token στην expression, επιστρέφοντας έτσι την παρένθεση αντί του 0, επιστρέφοντας μας έτσι αυτό το error

```
raise SyntaxError(f"Expected ), but found " +self.current_token.family+ "in line" + self.current_token.current_line)
```

Το ίδιο πρόβλημα παρατηρήσαμε και με το πρώτο test, το οποίο πριν υλοποιηθούν οι συναρτήσεις του ενδιάμεσου κώδικα υλοποιούσε την συντακτική ανάλυση μέχρι τέλους σχεδόν, δημιουργούσε όμως λανθασμένα το quad στον πολλαπλασιασμό fact = fact * 1, λόγω λανθασμένης ενημέρωσης του token.

```
DEBUG:root:Found a GroupSymbol in: (
DEBUG:root:...Found Family ID in .x.
DEBUG:root:...Went in on condition with current token x
DEBUG:root:...Went in on bool_term with current token x
DEBUG:root:...Went in on bool_factor with current token x
DEBUG:root:...Went in on expression with current token x
DEBUG:root:...Went in on optional_sign with current token x
DEBUG:root:...No sign found, returning
DEBUG:root:...Went in on term with current token x
DEBUG:root:...Went in on factor with current token x
DEBUG:root:...Went in on idtail with current token x
DEBUG:root:...Returning term x
DEBUG:root:...Found term x
DEBUG:root:...Returning expression x
DEBUG:root:Found a RelOperator: >
DEBUG:root:...Found a RelOperator: > , getting next token and finding next expression
DEBUG:root:Found a GroupSymbol in: )
DEBUG:root:...Went in on expression with current token )
DEBUG:root:...Went in on optional_sign with current token )
DEBUG:root:...No sign found, returning
DEBUG:root:...Went in on term with current token )
DEBUG:root:...Went in on factor with current token )
DEBUG:root:...Returning term )
DEBUG:root:...Found term )
DEBUG:root:...Returning expression )
DEBUG:root:...nextQuad called. The next generated quad will have no. 5
DEBUG:root:...nextQuad called. The next generated quad will have no. 6
DEBUG:root:Generating quad 5: >, x, ), _
DEBUG:root:...nextQuad called. The next generated quad will have no. 7
DEBUG:root:...nextQuad called. The next generated quad will have no. 8
DEBUG:root:Generating quad 7: jump, _, _, _
DEBUG:root:Found a Delimiter in: :
```

Περάσαμε αρκετό χρόνο να συζητάμε οτιδήποτε δεν λειτουργούσε στο project μας, ας κάνουμε μια νίξη όμως σε ότι λειτουργεί. Παρατηρούμε πως σε όλα τα test καλείται σωστά ο λεκτικός αναλυτής, διαβάζονται ορθά όλες οι λεκτικές μονάδες ανεξαρτήτως οικογένειας και δημιουργούνται τα quads του begin block. Υπάρχει ένα μικρό πρόβλημα με τα expressions όπου μερικές φορες δεν ενημερώνονται σωστά τα tokens, αλλά επι το πλείστον ελέγχονται σωστα κι απο τον συντακτικό αναλυτή. Εαν πηγαίναμε παρακάτω θα βλέπαμε πως δημιουργούνται και οι πίνακες που κρατάνε τα quads του ενδιάμεσου κώδικα, όπου και γίνονται return στις αντίστοιχες συναρτήσεις για να γίνουν backpatch. Εφόσον το πρόγραμμα μας δεν τρέχει μέχρι τέλους δεν βάλαμε να εκτυπώνει στο τερματικό την λίστα με όλα τα quads αλλά θα παρατηρούσαμε επι το πλείστον σωστή απαρίθμηση, δεν έχει γίνει όμως έλεγχος εαν γίνονται σωστά όλα τα backpatch, εάν και πιθανώς θα είχε σφάλματα καθώς καλούνται κάποια nextQuad τα οποία χαλάνε την απαρίθμηση. Έπειτα, έχουν υλοποιηθεί και οι κλάσεις του πίνακα συμβόλων, χωρίς να έχει γίνει όμως κάποια χρήση τους.