ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΡΓΑΣΤΗΡΙΟ ΜΕΤΑΓΛΩΤΤΙΣΤΩΝ

ΟΜΑΔΑ ΕΡΓΑΣΤΗΡΙΟΥ: Β3 1)

ΜΕΡΟΣ Α-3:Συμπλήρωση πρότυπου κώδικα FLEX

ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2022-2023

Σιδηρόπουλος Νικόλαος (ΠΑΔΑ-20390283)

Φράγκος Νικόλαος (ΠΑΔΑ-20390256)

Γουρδομιχάλης Δημήτριος (ΠΑΔΑ-20390043)

Τσελάνι Μαρίνο (ΠΑΔΑ-20390241)

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Εισαγωγή

Το μέρος Α3 της εργασίας ασχολείται με τη ***Λεκτική Ανάλυση***, δηλαδή την αναγνώριση των λεκτικών μονάδων σε μια ροή συμβόλων. Η αναγνώριση των διάφορων tokens πραγματοποιείται έπειτα από αίτημα του Συντακτικού Αναλυτή, κατά τη διάρκεια της Μεταγλώττισης.

Κεντρικό θέμα της εργασίας αποτελεί η ενασχόληση με το εργαλείο ***flex***. Είναι υπεύθυνο για τη δημιουργία λεκτικών αναλυτών. Περιλαμβάνει μια σειρά από κανόνες, οι οποίοι διαβάζουν τα σύμβολα που δίνονται ως είσοδος χαρακτήρα-χαρακτήρα και τα μετατρέπουν σε tokens. Το αρχείο που περιλαμβάνει τον κώδικα flex(.l) περνάει από έναν μεταγλωττιστή C. Έπειτα, «τρέχουμε» το αρχείο .c που παράγεται και παρατηρούμε τα αναγνωρισμένα tokens στο αρχείο εξόδου ή στην οθόνη του υπολογιστή μας.

Η δομή ενός προγράμματος flex είναι η εξής:

%{

Ορισμοί

}%

%%

Κανόνες

%%

Κώδικας χρήστη(Συνάρτηση main())

Ιδιαίτερη αναφορά πρέπει να γίνει στη συνάρτηση yylex, η οποία είναι υπεύθυνη για την λεκτική ανάλυση. Όταν δηλαδή καλείται η yylex διαβάζεται το επόμενο σύμβολο(χαρακτήρας) του αρχείου εισόδου και εφαρμόζονται οι αντίστοιχοι κανόνες. Η παραπάνω διαδικασία συνεχίζεται έως την αναγνώριση του EOF(το τέλος του αρχείου), δηλαδή του τερματικού συμβόλου.

Αναλυτικότερα, η ροή συμβόλων που χρησιμοποιήθηκε αποτελεί το αρχείο input.txt. Περιλαμβάνει ικανοποιητικό αριθμό έγκυρων αλλά και άκυρων λεκτικών μονάδων, με σκοπό να γίνεται κατανοητός ο τρόπος που χειρίζεται το πρόγραμμα τη ροή συμβόλων που του εκχωρούμε.

Ενώ, με βάση τις κανονικές τους εκφράσεις, αλλά και τη γραμματική της γλώσσας UniPython τα tokens(λεκτικές μονάδες) είναι τα εξής: Διαχωριστές, Αναγνωριστικά, Λέξεις κλειδιά, Λεκτικά κυριολεκτικά, Δεκαδικοί ακέραιοι, Δυαδικοί ακέραιοι, Αριθμοί κινούμενης υποδιαστολής, Φανταστικοί αριθμοί, Ειδικές λέξεις, Σχόλια, και

Unknown_Tokens, τα οποία αποτελούν τα σύμβολα που δεν περιγράφονται σε καμία από τις παραπάνω κατηγορίες.

Σημείωση: Τα σχόλια αναγνωρίζονται και έπειτα αγνοούνται. Επιπροσθέτως, αγνοούνται και οι white_spaces χαρακτήρες. Όταν εντοπιστεί χαρακτήρας αλλαγής γραμμής, ο μετρητής γραμμών(int line) αυξάνεται κατά ένα.

Επίσης, στο αρχείο token.h γίνεται η δήλωση του συνόλου των token, πριν από τη συγγραφή του κώδικα: αρχείο simple-flex-code.l

Τα αποτελέσματα της Λεκτικής Ανάλυσης βρίσκονται στο αρχείο output.txt .

Οι εντολές για την δημιουργία ενός εκτελέσιμου προγράμματος το οποίο δέχεται ως είσοδο το input.txt και παράγει ένα output.txt ως έξοδο βρίσκονται στο αρχείο Makefile και εκτελούνται με την εντολή make.

Εντολές στο Makefile:

```
flex -o simple- flex -code.c simple-flex-code.l
gcc -o simple- flex -code simple-flex-code.c
./simple-flex-code input.txt output.txt
```

Είσοδος και έξοδος

Αρχεία: input.txt & output.txt

Δημιουργήσαμε το αρχείο input.txt το οποίο περιλαμβάνει ένα σύνολο έγκυρων και μη περιπτώσεων για την δοκιμή κάθε κατηγορίας token της γλώσσας uni-python.

input.txt:

#Valid Cases
#Operators
+
-
*
**
/
//
%
@
<<
>>
&

|
^
~
<
>
<=
>=
==
!=

#Keywords
False
class
finally
is
return
None
continue
for
lambda
try
True
def
from
nonlocal
while
and
del
global
not
with
as
elif
if
or
yield
assert
else
import
pass
break
except
in
raise

#Special words
(
)

```
[
]
{
}
,
:
.
;
@
=
->
+=
-=
*=
/=
//=
%=
@=
&=
|=
^=
>>=
<<=
**=

#Integer
0
1
+7
-10
+99999999999999

#Binary
0b1111
0B0000
0b010101001010101

#Float
1.2
+30.4
-0.01
1e10
1E10
1e-10
1E-10
1e+10
1E+10
-1.3e7
```

```
1.2E10
1.5e-10
1.2E-5
1.3e+10
1.5E+2
0e0

#Imaginary
1j
+2j
-3j
99999999999j
1.2j
1e10j
1E5j
1e-2j
1E-10j
-1e+10j
7E+4j
10.01e2j
1.2E10j
1.5e-10j
-1.2E-3j
1.3e+10j
3.2E+1j

#Indentifiers
nikos
flag_
sum3
nikos_123
_test

#Strings
"test 1"
'test 2'
"test 3 \n \\ \" \' \r "
"Hello world!"
'Mark said, \"Boo!\"\n'
"Dimitris, Marinos, Nikos, Nikos"
"min = %d, max = %d, sum = %d, MO = %.2f"

#Valid examples of general expressions
name = "nikos" ;
a = b + 1 ;
count += 1 ;
sum += 2 ;
pow2 = a ^ 2 ;
```

```
* i = & value ;

if ( a == 0 )
{
   flag = 0 ;
   }
   elif ( a == 1 )
   {
      flag = 1 ;
          }
          else
          {
             return 0 ;
             }

             x = 10 ;
             while ( x > 0 )
             {

                        if ( flag != 0 )
                              continue ;
                                    else
                                                x -= 1 ;
                                          }
```

#Invalid cases

```
#Integer
01
-+1
7+
10b

#Binary
0b
0B
1b1111
0B2

#Float
1.
.2
00.1
1e
1.5e
0.1.2

#Imaginary
0j
```

```
03j
0.j
10ej
2e-j
5.4ej

#Identifiers
9nikos
2_Sum

#Strings
"test 1'
'test 2"
"test 3
'test 4
"test 5 \"
"test 6 \ n"
"test 7 ' "
'test 8 " '

#Invalid examples of general expressions
a=5;

sum=sum+2;
count+=1;
name="nikos"
x<5
if(y!=0)
valBool=True;
```

Από το παραπάνω input αρχείο και τον κώδικα παράγεται το output.txt

output.txt:

```
Line=6, token=operator(+), value="+"
Line=7, token=operator(-), value="-"
Line=8, token=operator(*), value="*"
Line=9, token=operator(**), value="**"
Line=10, token=operator(/), value="/"
Line=11, token=operator(//), value="//"
Line=12, token=operator(%), value="%"
Line=13, token=operator(@), value="@"
Line=14, token=operator(<<), value="<<"
```

Line=15, token=operator(>>), value=">>"
Line=16, token=operator(&), value="&"
Line=17, token=operator(|), value="|"
Line=18, token=operator(^), value="^"
Line=19, token=operator(~), value="~"
Line=20, token=operator(<), value="<"
Line=21, token=operator(>), value=">"
Line=22, token=operator(<=), value="<="
Line=23, token=operator(>=), value=">="
Line=24, token=operator(==), value="=="
Line=25, token=operator(!=), value="!="
Line=29, token=keyword(False), value="False"
Line=30, token=keyword(class), value="class"
Line=31, token=keyword(finally), value="finally"
Line=32, token=keyword(is), value="is"
Line=33, token=keyword(return), value="return"
Line=34, token=keyword(None), value="None"
Line=35, token=keyword(continue), value="continue"
Line=36, token=keyword(for), value="for"
Line=37, token=keyword(lambda), value="lambda"
Line=38, token=keyword(try), value="try"
Line=39, token=keyword(True), value="True"
Line=40, token=keyword(def), value="def"
Line=41, token=keyword(from), value="from"
Line=42, token=keyword(nonlocal), value="nonlocal"
Line=43, token=keyword(while), value="while"
Line=44, token=keyword(and), value="and"
Line=45, token=keyword(del), value="del"
Line=46, token=keyword(global), value="global"
Line=47, token=keyword(not), value="not"
Line=48, token=keyword(with), value="with"
Line=49, token=keyword(as), value="as"
Line=50, token=keyword(elif), value="elif"
Line=51, token=keyword(if), value="if"
Line=52, token=keyword(or), value="or"
Line=53, token=keyword(yield), value="yield"
Line=54, token=keyword(assert), value="assert"
Line=55, token=keyword(else), value="else"
Line=56, token=keyword(import), value="import"
Line=57, token=keyword(pass), value="pass"
Line=58, token=keyword(break), value="break"
Line=59, token=keyword(except), value="except"
Line=60, token=keyword(in), value="in"
Line=61, token=keyword(raise), value="raise"
Line=65, token=s_word((), value="("
Line=66, token=s_word()), value=")"
Line=67, token=s_word([), value="["
Line=68, token=s_word(]), value="]"

```
Line=69, token=s_word({), value="{"
Line=70, token=s_word(}), value="}"
Line=71, token=s_word(,), value=","
Line=72, token=s_word(:), value=":"
Line=73, token=s_word(.), value="."
Line=74, token=s_word(;), value=";"
Line=75, token=operator(@), value="@"
Line=76, token=s_word(=), value="="
Line=77, token=s_word(->), value="->"
Line=78, token=s_word(+=), value="+="
Line=79, token=s_word(-=), value="-="
Line=80, token=s_word(*=), value="*="
Line=81, token=s_word(/=), value="/="
Line=82, token=s_word(//=), value="//="
Line=83, token=s_word(%=), value="%="
Line=84, token=s_word(@=), value="@="
Line=85, token=s_word(&=), value="&="
Line=86, token=s_word(|=), value="|="
Line=87, token=s_word(^=), value="^="
Line=88, token=s_word(>>=), value=">>="
Line=89, token=s_word(<<=), value="<<="
Line=90, token=s_word(**=), value="**="
Line=94, token=DECIMAL, value="0"
Line=95, token=DECIMAL, value="1"
Line=96, token=DECIMAL, value="+7"
Line=97, token=DECIMAL, value="-10"
Line=98, token=DECIMAL, value="+99999999999999"
Line=102, token=BINARY, value="0b1111"
Line=103, token=BINARY, value="0B0000"
Line=104, token=BINARY, value="0b010101001010101"
Line=108, token=FLOAT, value="1.2"
Line=109, token=FLOAT, value="+30.4"
Line=110, token=FLOAT, value="-0.01"
Line=111, token=FLOAT, value="1e10"
Line=112, token=FLOAT, value="1E10"
Line=113, token=FLOAT, value="1e-10"
Line=114, token=FLOAT, value="1E-10"
Line=115, token=FLOAT, value="1e+10"
Line=116, token=FLOAT, value="1E+10"
Line=117, token=FLOAT, value="-1.3e7"
Line=118, token=FLOAT, value="1.2E10"
Line=119, token=FLOAT, value="1.5e-10"
Line=120, token=FLOAT, value="1.2E-5"
Line=121, token=FLOAT, value="1.3e+10"
Line=122, token=FLOAT, value="1.5E+2"
Line=123, token=FLOAT, value="0e0"
Line=127, token=IMAGINARY, value="1j"
Line=128, token=IMAGINARY, value="+2j"
```

Line=129, token=IMAGINARY, value="-3j"
Line=130, token=IMAGINARY, value="99999999999j"
Line=131, token=IMAGINARY, value="1.2j"
Line=132, token=IMAGINARY, value="1e10j"
Line=133, token=IMAGINARY, value="1E5j"
Line=134, token=IMAGINARY, value="1e-2j"
Line=135, token=IMAGINARY, value="1E-10j"
Line=136, token=IMAGINARY, value="-1e+10j"
Line=137, token=IMAGINARY, value="7E+4j"
Line=138, token=IMAGINARY, value="10.01e2j"
Line=139, token=IMAGINARY, value="1.2E10j"
Line=140, token=IMAGINARY, value="1.5e-10j"
Line=141, token=IMAGINARY, value="-1.2E-3j"
Line=142, token=IMAGINARY, value="1.3e+10j"
Line=143, token=IMAGINARY, value="3.2E+1j"
Line=147, token=IDENTIFIER, value="nikos"
Line=148, token=IDENTIFIER, value="flag_"
Line=149, token=IDENTIFIER, value="sum3"
Line=150, token=IDENTIFIER, value="nikos_123"
Line=151, token=IDENTIFIER, value="_test"
Line=155, token=STRING, value=""test 1""
Line=156, token=STRING, value="'test 2'"
Line=157, token=STRING, value=""test 3 \n \\ \" \' \r ""
Line=158, token=STRING, value=""Hello world!""
Line=159, token=STRING, value="'Mark said, \"Boo!\"\n'"
Line=160, token=STRING, value=""Dimitris, Marinos, Nikos, Nikos""
Line=161, token=STRING, value=""min = %d, max = %d, sum = %d, MO = %.2f""
Line=165, token=IDENTIFIER, value="name"
Line=165, token=s_word(=), value="="
Line=165, token=STRING, value=""nikos""
Line=165, token=s_word(;), value=";"
Line=166, token=IDENTIFIER, value="a"
Line=166, token=s_word(=), value="="
Line=166, token=IDENTIFIER, value="b"
Line=166, token=operator(+), value="+"
Line=166, token=DECIMAL, value="1"
Line=166, token=s_word(;), value=";"
Line=167, token=IDENTIFIER, value="count"
Line=167, token=s_word(+=), value="+="
Line=167, token=DECIMAL, value="1"
Line=167, token=s_word(;), value=";"
Line=168, token=IDENTIFIER, value="sum"
Line=168, token=s_word(+=), value="+="
Line=168, token=DECIMAL, value="2"
Line=168, token=s_word(;), value=";"
Line=169, token=IDENTIFIER, value="pow2"
Line=169, token=s_word(=), value="="
Line=169, token=IDENTIFIER, value="a"

Line=169, token=operator(^), value="^"
Line=169, token=DECIMAL, value="2"
Line=169, token=s_word(;), value=";"
Line=170, token=operator(*), value="*"
Line=170, token=IDENTIFIER, value="i"
Line=170, token=s_word(=), value="="
Line=170, token=operator(&), value="&"
Line=170, token=IDENTIFIER, value="value"
Line=170, token=s_word(;), value=";"
Line=172, token=keyword(if), value="if"
Line=172, token=s_word((), value="("
Line=172, token=IDENTIFIER, value="a"
Line=172, token=operator(==), value="=="
Line=172, token=DECIMAL, value="0"
Line=172, token=s_word()), value=")"
Line=173, token=s_word({), value="{"
Line=174, token=IDENTIFIER, value="flag"
Line=174, token=s_word(=), value="="
Line=174, token=DECIMAL, value="0"
Line=174, token=s_word(;), value=";"
Line=175, token=s_word(}), value="}"
Line=176, token=keyword(elif), value="elif"
Line=176, token=s_word((), value="("
Line=176, token=IDENTIFIER, value="a"
Line=176, token=operator(==), value="=="
Line=176, token=DECIMAL, value="1"
Line=176, token=s_word()), value=")"
Line=177, token=s_word({), value="{"
Line=178, token=IDENTIFIER, value="flag"
Line=178, token=s_word(=), value="="
Line=178, token=DECIMAL, value="1"
Line=178, token=s_word(;), value=";"
Line=179, token=s_word(}), value="}"
Line=180, token=keyword(else), value="else"
Line=181, token=s_word({), value="{"
Line=182, token=keyword(return), value="return"
Line=182, token=DECIMAL, value="0"
Line=182, token=s_word(;), value=";"
Line=183, token=s_word(}), value="}"
Line=185, token=IDENTIFIER, value="x"
Line=185, token=s_word(=), value="="
Line=185, token=DECIMAL, value="10"
Line=185, token=s_word(;), value=";"
Line=186, token=keyword(while), value="while"
Line=186, token=s_word((), value="("
Line=186, token=IDENTIFIER, value="x"
Line=186, token=operator(>), value=">"
Line=186, token=DECIMAL, value="0"

Line=186, token=s_word()), value=")"
Line=187, token=s_word({), value="{"
Line=189, token=keyword(if), value="if"
Line=189, token=s_word((), value="("
Line=189, token=IDENTIFIER, value="flag"
Line=189, token=operator(!=), value="!="
Line=189, token=DECIMAL, value="0"
Line=189, token=s_word()), value=")"
Line=190, token=keyword(continue), value="continue"
Line=190, token=s_word(;), value=";"
Line=191, token=keyword(else), value="else"
Line=192, token=IDENTIFIER, value="x"
Line=192, token=s_word(-=), value="-="
Line=192, token=DECIMAL, value="1"
Line=192, token=s_word(;), value=";"
Line=193, token=s_word(}), value="}"

Line=200, UNKNOWN_TOKEN, value="01"
Line=201, UNKNOWN_TOKEN, value="-+1"
Line=202, UNKNOWN_TOKEN, value="7+"
Line=203, UNKNOWN_TOKEN, value="10b"
Line=207, UNKNOWN_TOKEN, value="0b"
Line=208, UNKNOWN_TOKEN, value="0B"
Line=209, UNKNOWN_TOKEN, value="1b1111"
Line=210, UNKNOWN_TOKEN, value="0B2"
Line=214, UNKNOWN_TOKEN, value="1."
Line=215, UNKNOWN_TOKEN, value=".2"
Line=216, UNKNOWN_TOKEN, value="00.1"
Line=217, UNKNOWN_TOKEN, value="1e"
Line=218, UNKNOWN_TOKEN, value="1.5e"
Line=219, UNKNOWN_TOKEN, value="0.1.2"
Line=223, UNKNOWN_TOKEN, value="0j"
Line=224, UNKNOWN_TOKEN, value="03j"
Line=225, UNKNOWN_TOKEN, value="0.j"
Line=226, UNKNOWN_TOKEN, value="10ej"
Line=227, UNKNOWN_TOKEN, value="2e-j"
Line=228, UNKNOWN_TOKEN, value="5.4ej"
Line=232, UNKNOWN_TOKEN, value="9nikos"
Line=233, UNKNOWN_TOKEN, value="2_Sum"
Line=237, UNKNOWN_TOKEN, value=""test"
Line=237, UNKNOWN_TOKEN, value="1'"
Line=238, UNKNOWN_TOKEN, value="'test"
Line=238, UNKNOWN_TOKEN, value="2""
Line=239, UNKNOWN_TOKEN, value=""test"
Line=239, token=DECIMAL, value="3"
Line=240, UNKNOWN_TOKEN, value="'test"
Line=240, token=DECIMAL, value="4"
Line=241, UNKNOWN_TOKEN, value=""test"

Line=241, token=DECIMAL, value="5"
Line=241, UNKNOWN_TOKEN, value="\""
Line=242, UNKNOWN_TOKEN, value=""test"
Line=242, token=DECIMAL, value="6"
Line=242, UNKNOWN_TOKEN, value="\"
Line=242, UNKNOWN_TOKEN, value="n""
Line=243, UNKNOWN_TOKEN, value=""test"
Line=243, token=DECIMAL, value="7"
Line=243, UNKNOWN_TOKEN, value="'"
Line=243, UNKNOWN_TOKEN, value="'"
Line=244, UNKNOWN_TOKEN, value="'test"
Line=244, token=DECIMAL, value="8"
Line=244, UNKNOWN_TOKEN, value="'"
Line=244, UNKNOWN_TOKEN, value="'"
Line=248, UNKNOWN_TOKEN, value="a=5;"
Line=249, UNKNOWN_TOKEN, value="sum=sum+2;"
Line=250, UNKNOWN_TOKEN, value="count+=1;"
Line=251, UNKNOWN_TOKEN, value="name="nikos""
Line=252, UNKNOWN_TOKEN, value="x<5"
Line=253, UNKNOWN_TOKEN, value="if(y!=0)"
Line=254, UNKNOWN_TOKEN, value="valBool=True;"

Γενικές παρατηρήσεις/Σχολιασμός αποτελεσμάτων :

- Σύμφωνα με την δήλωση των tokens , το σύμβολο @ θα αναγνωρίζεται πάντα ως OPERATOR και όχι ως S_WORD.
- Όταν έχουμε, παραδείγματος χάριν, α+β με κενά ενδιάμεσα, αναγνωρίζονται ως IDENTIFIER OPERATOR IDENTIFIER ενώ όταν δεν υπάρχουν κενά αναμεσά τους, αναγνωρίζονται όλα μαζί UNKNOWN_TOKEN.
- Η μεταγλώττιση των αρχείων με κώδικα flex και των ενδιάμεσων με κώδικα c, που παράχθηκαν, έγινε με επιτυχία και χωρίς την ανάγκη αντιμετώπισης τυχών προβλημάτων.
- Οι μη έγκυρες περιπτώσεις που φαίνονται στο αρχείο output.txt διακρίνονται σε δύο κατηγορίες σφάλματος. Η πρώτη περίπτωση αφορά αυτές που δεν αναγνωρίζονται απο τους δηλωμένους κανόνες και η δεύτερη αφορά αυτές που αποτελούνται απο ένα συνδυασμό διαφορετικών κανόνων χωρίς whitespace μεταξύ τους.

Αρμοδιότητες μελών:

Για την εκπόνηση της συγκεκριμένης εργασίας υπήρξε δια ζώσης και εξ' αποστάσεως συνεργασία μεταξύ των μελών της ομάδας, που αναγράφονται παρακάτω. Αναλυτικότερα, τα εξής μέλη ασχολήθηκαν με το σύνολο των τμημάτων της εργασίας:

- **Σιδηρόπουλος Νικόλαος: Συγγραφή αρχείου εισόδου, συγγραφή κώδικα, σύνταξη τεκμηρίωσης, σχολιασμός κώδικα και αποτελεσμάτων.**
- **Φράγκος Νικόλαος: Συγγραφή αρχείου εισόδου, συγγραφή κώδικα, σύνταξη τεκμηρίωσης, σχολιασμός κώδικα και αποτελεσμάτων.**
- **Γουρδομιχάλης Δημήτριος: Συγγραφή αρχείου εισόδου, συγγραφή κώδικα, σύνταξη τεκμηρίωσης, σχολιασμός κώδικα και αποτελεσμάτων.**
- **Τσελάνι Μαρίνο: Συγγραφή αρχείου εισόδου, συγγραφή κώδικα, σύνταξη τεκμηρίωσης, σχολιασμός κώδικα και αποτελεσμάτων.**