Πανεπιστήμιο Ιωαννίνων
Πολυτεχνική Σχολή
Τμήμα Μηχ. Η/Υ & Πληροφορικής
Μάθημα: "Μεταφραστές ΙΙ"
Διδάσκων: Γεώργιος Μανής
Οκτώβριος 2025

Η γλώσσα προγραμματισμού oo2c

Να υλοποιήσετε μεταφραστή της γλώσσας **oo2c** (object oriented to C) ο οποίος θα μεταφράζει την αντικειμενοστραφή γλώσσα **oo2c** στη διαδικασιακή γλώσσα **C**. Η μετάφραση θα είναι από γλώσσα υψηλού επιπέδου σε γλώσσα υψηλού επιπέδου.

Ως εργαλείο ανάπτυξης να χρησιμοποιήσετε το μέτα-εργαλείο **ANTLR**. Συστηνόμενη γλώσσα υλοποίησης η Python.

Για τη γλώσσα **oo2c** ισχύουν τα εξής τα οποία πρέπει να λάβετε υπόψη σας στον σχεδιασμό της:

- ~~Η γλώσσα πρέπει να είναι υποσύνολο της Python, τα προγράμματα δηλαδή που είναι γραμμένα στη γλώσσα **oo2c** να μπορούν να εκτελεστούν από έναν διερμηνέα της Python~~
- κάθε πρόγραμμα αποτελείται από κλάσεις και κυρίως πρόγραμμα
- το κυρίως πρόγραμμα δεν δηλώνεται ή λαμβάνεται ως κλάση
- κάθε πρόγραμμα μπορεί να έχει μία έως περισσότερες κλάσεις
- κάθε κλάση δηλώνεται με τη λέξη κλειδί **class** και ακολουθεί το όνομα της κλάσης
- αν η κλάση κληρονομεί άλλη κλάση, τότε το όνομα της κλάσης που κληρονομείται τοποθετείται στη δήλωση της κλάσης, αμέσως μετά το όνομα της κλάσης που κληρονομεί και το σύμβολο ":"
- οι κλάσεις που κληρονομούνται πρέπει να προηγούνται στον κώδικα από τις κλάσεις που τις κληρονομούν
- στην αρχή κάθε κλάσης δηλώνονται οι μέθοδοι δημιουργοί, με το όνομα **___init___** και στη συνέχεια ακολουθούν οι παράμετροι της μεθόδου
- μία τουλάχιστον ή περισσότερες μέθοδοι δημιουργοί πρέπει να υλοποιηθούν στην αρχή της κλάσης
- μετά τις μεθόδους-δημιουργούς επιτρέπεται να δηλωθούν καμία ή περισσότερες μέθοδοι της κλάσης
- μία μέθοδος δηλώνεται με τη λέξη κλειδί **def**, ακολουθεί το όνομά της και στη συνέχεια, μέσα σε παρένθεση, οι παράμετροί της
- η πρώτη παράμετρος μιας μεθόδου είναι υποχρεωτικά η λέξη **self**, η οποία υποδηλώνει το αντικείμενο στο οποίο εφαρμόζεται
- τα πεδία των κλάσεων και οι τοπικές μεταβλητές των μεθόδων είναι όλα τύπου **int**
- οι παράμετροι των μεθόδων μπορεί να είναι αντικείμενα

- η πρόσβαση στα πεδία ενός αντικειμένου γίνεται με το όνομα του αντικειμένου, ακολουθούμενο από τελεία και στη συνέχεια το όνομα του πεδίου. Το όνομα του αντικειμένου μπορεί να είναι το **self** αν πρόκειται για αναφορά από ένα αντικείμενο στο εαυτό του
- οι εντολές που υποστηρίζει η γλώσσα είναι οι **if-else, while, print** και **return**, ενώ υποστηρίζεται επίσης και η εκχώρηση. Αν θέλετε να προσθέσετε εντολές, μπορείτε να το κάνετε. ~~Το αρχικό πρόγραμμα πρέπει να εξακολουθεί να μπορεί να εκτελείται σε διερμηνέα Python~~
- υποστηρίζονται λογικές συνθήκες και αριθμητικές εκφράσεις με τον συνήθη τρόπο και με τη συνήθη προτεραιότητα τελεστών
- η εφαρμογή μίας μεθόδου γίνεται με το όνομα του αντικειμένου στο οποίο εφαρμόζεται, ακολουθούμενο από μία τελεία, το όνομα της μεθόδου και τις παραμέτρους της σε παρένθεση

Καταληκτική ημερομηνία παράδοσης: 5 Δεκεμβρίου.

Η παράδοση θα γίνει με ηλεκτρονικό ταχυδρομείο.

Καταληκτική ημερομηνία παρουσίασης της άσκησής (στο γραφείο ή σε teams): 12 Δεκεμβρίου

Ακολουθεί η γραμματική της γλώσσας **oo2c** η οποία μπορεί να χρησιμοποιηθεί στο antlt. Στη γραμματική ορίζονται όλες οι λεπτομέρειες της γλώσσας.

```
grammar oos;

startRule
	:	classes
	;

classes
	:	class_def*
		class_main_def
		EOF
	;

class_def
	:	'class' class_name ( '(' class_name (',' class_name  )* ')' )? ':'
		declarations
		class_body
	;

class_main_def
	:	'class' ('main' | 'Main') ':'
```

```
        declarations
        main_body
    ;

class_name
    : ID
    ;

declarations
    :   (decl_line (';' decl_line)* ';';')?
    ;

class_body
    :   (constructor_def ';';')+
        (method_def ';';')*
    ;

main_body
    :   method_main_def ';';'
    ;

decl_line
    :   types ID  (',' ID  )*
    ;

constructor_def
    :   'def' '___init___' parameters ':' class_name
        declarations
        method_body
    ;

method_def
    :   'def' ID parameters ':' ('int' | '-' | class_name)
        declarations
        method_body
    ;

method_main_def
    :   'def' 'main' '(' 'self' ')' ':' '-'
        declarations
        method_body
    ;

types
    :   class_name
```

```
        |   'int'
        ;

parameters
        :   '(' parlist ')'
        ;

method_body
        :   declarations
            (statements)?
        ;

return_type
        :   types
        |   '-'
        ;

parlist
        :   'self'
            (',' types ID )*
        ;

statements :   statement (';' statement )*
        ;


statement
        :   assignment_stat
        |   direct_call_stat
        |   if_stat
        |   while_stat
        |   return_stat
        |   input_stat
        |   print_stat
        ;

assignment_stat
        :   ('self.')? ID '=' expression
        |   constructor_call
        ;

direct_call_stat
        :   ('self.')? ID '.' func_call
        |   ('self.')? func_call
        ;
```

```
if_stat
      :  'if' '(' condition ')' ':'
         (statements ';' )?
         else_part
         'endif'
      ;

else_part
      :  'else' ':'
         ( statements ';' )?
      |
      ;

while_stat
      :  'while' '(' condition ')' ':'
         ( statements )?
         'endwhile'
      ;

return_stat
      :  'return' ('self' | 'self.'ID | expression)
      ;

input_stat
      :  'input' ('self.')? ID
      ;

print_stat
      :  'print' expression (',' expression)*
      ;

expression
      :  optional_sign term (add_oper term )*
      ;

arguments
      :  '(' arglist ')'
      ;

condition
      :  boolterm
         ('or' boolterm )*
      ;
```

```
optional_sign
        :   add_oper
        |
        ;

term
        :   factor  ( mul_oper factor )*
        ;

add_oper
        :   '+'
        |   '-'
        ;

arglist
        :   argitem (',' argitem )*
        |
        ;

boolterm
        :   boolfactor ( 'and' boolfactor )*
        ;

factor
        :   INTEGER
        |   '(' expression ')'
        |   ('self.')? ID
        |   ('self.')? ID '.' func_call
        |   ('self.')? func_call
        |   ID.ID
        |   ID
        ;

mul_oper
        :   '*'
        |   '/'
        ;

argitem
        :   expression
        ;

boolfactor
        :   'not' '[' condition ']'
        |   '[' condition ']'
```

```
        |   expression rel_oper expression
        ;

func_call
        :   ID arguments
        ;

constructor_call
        :   class_name
            arguments
        ;

rel_oper
        :  '=='
        |   '<='
        |   '>='
        |   '>'
        |   '<'
        |   '!='
        ;

//------------------------------------------------------------------------
//------------------------------------------------------------------------

WS: [ \t\r\n]+ -> skip;
COMMENTS: '#' ~[#]* '#' -> skip;
ID: ID_START (ID_CONTINUE)*;
INTEGER: NON_ZERO_DIGIT (DIGIT)* | '0'+;


fragment ID_START
        : [A-Z]
        | [a-z]
        ;

fragment ID_CONTINUE
        : '_'
        | [A-Z]
        | [a-z]
        | [0-9]
        ;


fragment NON_ZERO_DIGIT
        : [1-9]
```

```
    ;

fragment DIGIT
    : [0-9]
    ;
```