Μεταγλωττιστές 2023-24

Προσθήκη συναρτήσεων στη γραμματική των αριθμητικών εκφράσεων

Στόχος

- Η υποστήριξη απλών συναρτήσεων από τον συντακτικό αναλυτή και AST interpreter
  - Δηλώνονται στην αρχή του προγράμματος
    - `function func-name (argument-list) { statements }`
  - Καλούνται είτε ως statement είτε ως expression
    - `func-name (expression-list)`
  - Μπορούν να επιστρέφουν μια float τιμή
    - `return expression`
    - Για απλοποίηση, το return θα επιτρέπεται ακόμα και στο «κυρίως πρόγραμμα» που ακολουθεί μετά τις δηλώσεις των συναρτήσεων

Συντακτική ανάλυση

- Προσθέστε στη γραμματική τους νέους κανόνες
  - Γραμματική για να αρχίσετε με το on-line εργαλείο στο http://smlweb.cpsc.ucalgary.ca/start.html

```
Stmt_list -> Stmt Stmt_list | .
Stmt -> id eq Expr | print Expr
     | if Expr Block_stmt | while Expr Block_stmt .
Block_stmt -> Stmt | lbr Stmt_list rbr .
Expr -> Term Term_tail .
Term_tail -> Addop Term Term_tail | .
Term -> Factor Factor_tail .
Factor_tail -> Multop Factor Factor_tail | .
Factor -> lpar Expr rpar | id | num .
Addop -> plus | minus .
Multop -> mult | div .
```

Η νέα γραμματική

```
Program → F_declarations Stmt_list
F_declarations → F_declaration F_declarations | ε
F_declaration → function id ( Param_list ) Block_stmt
Param_list → id Param_tail | ε
Param_tail → , id Param_tail | ε
Stmt_list → Stmt Stmt_list | ε
Stmt → id Assign_or_call | print Expr
     | if Expr Block_stmt (else Block_stmt)? | while Expr Block_stmt
     | return Expr
Assign_or_call → = Expr | ( Arg_list )
Arg_list → Expr Arg_tail | ε
Arg_tail → , Expr Arg_tail | ε
Block_stmt → Stmt | { Stmt_list }
Expr → Term (Addop Term)*
Term → Factor (Multop Factor)*
Factor → ( Expr ) | id Id_or_call | number
Id_or_call → ( Arg_list ) | ε
Addop → + | -
Multop → * | /
```

Μη τερματικό	Σύνολο FIRST	Σύνολο FOLLOW
Program	function id print if while return	EOT (end-of-text)
F_declarations	function	id print if while return EOT
F_declaration	function	
Param_list	id	)
Param_tail	,	)
Stmt_list	id print if while return	} EOT
Stmt	id print if while return	
Assign_or_call	= (	
Arg_list	(id num	)
Arg_tail	,	)
Block_stmt	{ id print if while return	
Expr	(id num	
Term	(id num	
Factor	(id num	
Id_or_call	(	* / + - { } ,) function id print if while return else EOT

Συντακτική ανάλυση (2)

- Δηλώστε τα νέα patterns του tokenizer
  - Keywords και σημεία στίξης
- Υλοποιήστε τις μεθόδους των νέων μη τερματικών συμβόλων
  - Και τροποποιήστε αν χρειαστεί τις υπάρχουσες μεθόδους
    - Αν έχουν αλλάξει τα FIRST/FOLLOW sets

Συντακτική ανάλυση (3)

- Δοκιμάστε τη λειτουργία του συντακτικού αναλυτή με τη διπλανή δοκιμαστική είσοδο
  - Χωρίς κατασκευή και διερμηνεία AST

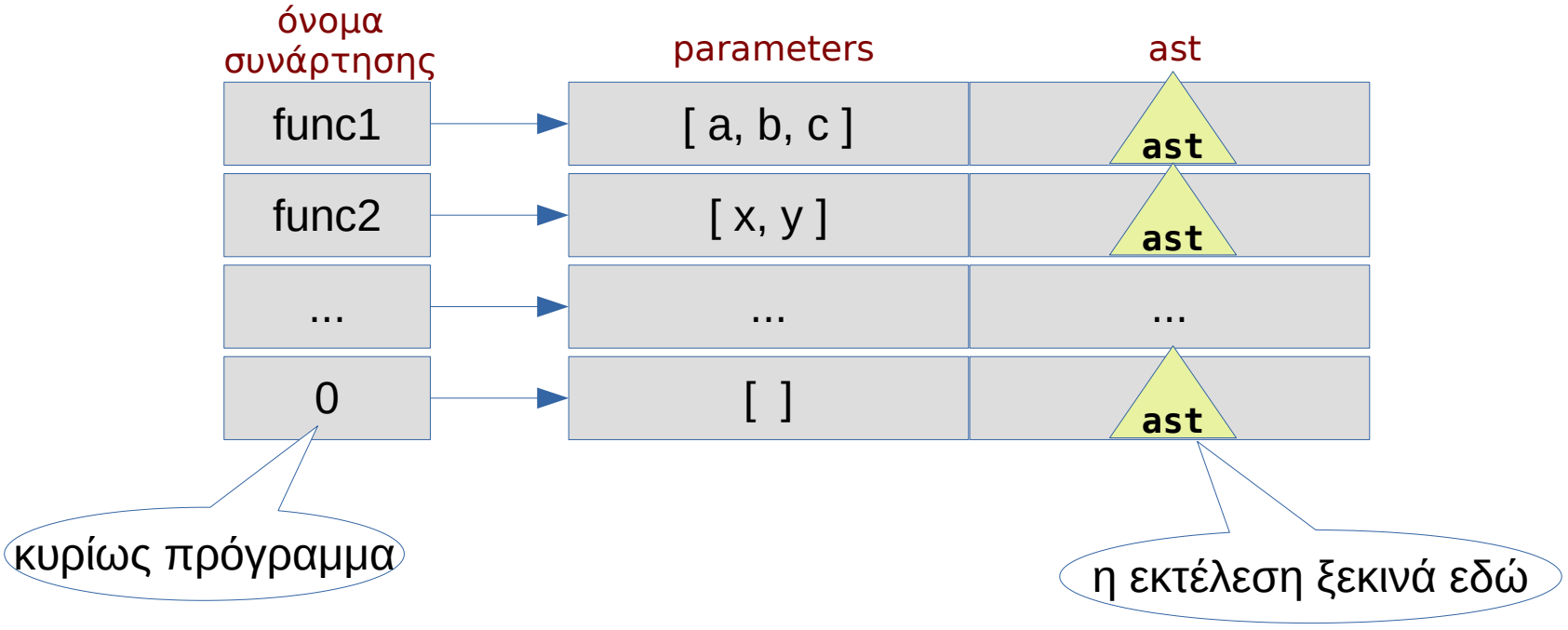
```
function um(x)
  return 0-x
function cube(x) {
  return x*x*x
}
a = 2 + 7.55*44
print a
if a-7 {
  b = 3*(a-99.01)
  it = 5
  while it {
    print it+b*0.23
    it = it - 1
  }
}
c = 5-3-2
if it+1 if c print c else print
cube(um(c+28))
```

Δηλώσεις συναρτήσεων

- Η δήλωση μιας συνάρτησης δημιουργεί ένα νέο AST εκτέλεσης
  - Θα πρέπει να αποθηκευτεί μαζί με τα AST των υπόλοιπων συναρτήσεων
  - Και το AST του κυρίως προγράμματος
  - Ο interpreter θα πρέπει να έχει πρόσβαση σε όλα τα AST εκτέλεσης
- Η πληροφορία κάθε συνάρτησης περιέχει επίσης μια λίστα με τα ονόματα των παραμέτρων της
- Ο συνακτικός αναλυτής θα πρέπει να προειδοποιεί μέσω σφάλματος για πολλαπλές δηλώσεις της ίδιας συνάρτησης

Πίνακας συναρτήσεων

- Ένα python dict με την πληροφορία για κάθε συνάρτηση
  - Το κυρίως πρόγραμμα αποθηκεύεται ως συνάρτηση "0"



Κλήσεις συναρτήσεων

- Ο έλεγχος ύπαρξης της καλούμενης συνάρτησης και του σωστού αριθμού ορισμάτων μετατίθεται για τη φάση της εκτέλεσης (στον interpreter)
  - Δεν υποστηρίζονται “forward declarations”
  - Κατά τη συντακτική ανάλυση μιας κλήσης συνάρτησης είναι πιθανό η καλούμενη συνάρτηση να μην έχει δηλωθεί ακόμα

“Procedure” ή “Function”;

- Στις πρώτες γλώσσες προγραμματισμού η διάκριση ήταν σαφής:
  - Procedure είναι μια συνάρτηση που καλείται για να εκτελέσει κάποιες εντολές (και δεν επιστρέφει κάτι)
  - Function είναι μια συνάρτηση που επιστρέφει κάποια τιμή, η οποία θα χρησιμοποιηθεί στα πλαίσια μιας έκφρασης
- Στις σύγχρονες γλώσσες προγραμματισμού η συνάρτηση καλείται να καλύψει και τους δύο ρόλους

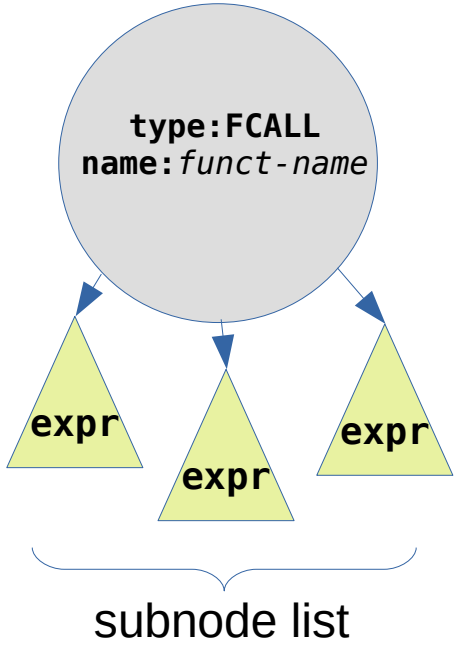
“Procedure” ή “Function”; (2)

- Ρόλοι στη γλώσσα μας:
  - Όταν η κλήση της συνάρτησης έχει δημιουργηθεί από το Stmt → … τότε η συνάρτηση χρησιμοποιείται ως procedure
    - Δεν αναμένεται να επιστρέψει κάτι (και αν επιστρέψει, αυτό δεν χρησιμοποιείται)
  - Όταν η κλήση της συνάρτησης έχει δημιουργηθεί από το Factor → … τότε η συνάρτηση χρησιμοποιείται ως function
    - Αναμένεται να επιστρέψει μια float τιμή

“Procedure” ή “Function”; (3)

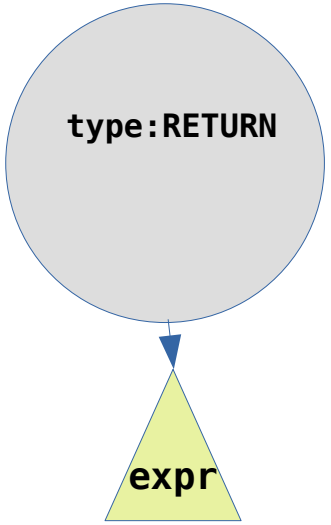
- Στη γλώσσα μας δεν υπάρχει δήλωση του τύπου της επιστρεφόμενης τιμής μιας συνάρτησης
  - Επίσης, δεν είναι υποχρεωτικό μια συνάρτηση να επιστρέφει τιμή από όλα τα σημεία επιστροφής
    - Αν εκτελεστεί return τότε επιστρέφεται μια τιμή
    - Αν η επιστροφή γίνει όταν απλά τελειώσει το block statement της συνάρτησης, δεν υπάρχει επιστρεφόμενη τιμή
- Συνεπώς ο έλεγχος αυτός θα πρέπει να μετατεθεί για τη στιγμή της εκτέλεσης
  - Εάν η κλήση γίνεται κατά την εκτέλεση εντολής (statement) → δεν απαιτείται επιστρεφόμενη τιμή
  - Εάν η κλήση γίνεται κατά τον υπολογισμό έκφρασης → απαιτείται επιστρεφόμενη τιμή

Κόμβος AST για την κλήση συνάρτησης



- Ο τύπος του κόμβου είναι FCALL
- Το attribute “name” περιέχει το όνομα της καλούμενης συνάρτησης
- Τα παιδιά (subnodes) του κόμβου είναι μια σειρά από υποδέντρα αριθμητικής έκφρασης, ένα για κάθε όρισμα κλήσης
  - με τη σειρά που αντιστοιχούν στις δηλωμένες παραμέτρους της συνάρτησης

Κόμβος AST για το return



- Ο τύπος του κόμβου είναι RETURN
- Υπάρχει μοναδικό παιδί (subnode) με ένα υποδέντρο αριθμητικής έκφρασης
  - Υπολογίζει την επιστρεφόμενη τιμή

AST Interpreter

- Προσθήκη λειτουργικότητας για την κλήση συναρτήσεων (κόμβοι FCALL)
  - ως εκτέλεση εντολής
    - δεν απαιτείται επιστρεφόμενη τιμή
  - ως υπολογισμός μέρους έκφρασης
    - απαιτείται επιστρεφόμενη τιμή
- Προσθήκη λειτουργικότητας για την επιστροφή τιμών από συναρτήσεις (κόμβοι RETURN)
  - Και έλεγχος της ροής εκτέλεσης
    - Μετά από return δεν πρέπει να εκτελεστούν οι επόμενες εντολές

Εμβέλεια μεταβλητών

- Στη γλώσσα μας για λόγους απλότητας θα υπάρχουν μόνο τοπικές μεταβλητές
  - Κάθε συνάρτηση θα έχει τις δικές της μεταβλητές
    - Για την ακρίβεια: κάθε κλήση μιας συνάρτησης θα έχει τις δικές της μεταβλητές
  - Η μόνη επικοινωνία μεταξύ συναρτήσεων και του κυρίως προγράμματος θα γίνεται μέσω παραμέτρων και επιστρεφόμενων τιμών
  - Δεν θα υποστηρίζονται global μεταβλητές

Χώρος αποθήκευσης τοπικών μεταβλητών

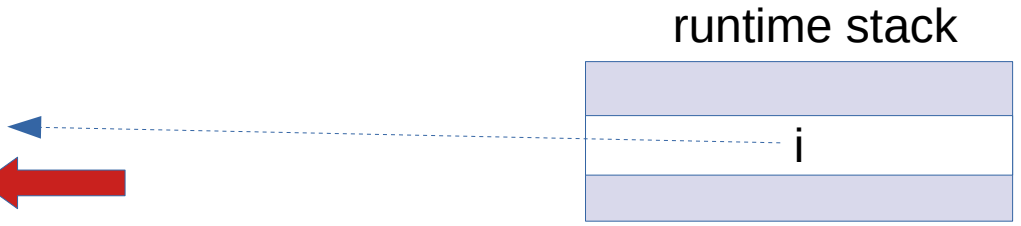
- Όλες οι «δομημένες» γλώσσες προγραμματισμού χρησιμοποιούν μια στοίβα στη μνήμη (runtime stack) για την αποθήκευση των τοπικών μεταβλητών τους
- Στη στοίβα δεσμεύεται χώρος για τις τοπικές μεταβλητές σε κάθε κλήση συνάρτησης (call frame ή activation record)
  - Η στοίβα μεγαλώνει
- Στη επιστροφή από μια κλήση συνάρτησης το τρέχον call frame αποδεσμεύεται
  - Η στοίβα επιστρέφει στην κατάσταση πριν την τελευταία κλήση
- Στα call frames αποθηκεύονται επίσης οι παράμετροι εισόδου στη συνάρτηση και η επιστρεφόμενη τιμή
  - Σημείωση: οι μοντέρνοι μεταγλωττιστές προσπαθούν μέσω βελτιστοποιήσεων να κρατούν μεγάλο μέρος από τα παραπάνω σε καταχωρητές αντί της στοίβας (κύρια μνήμη)

Παράδειγμα

```
f() {
  int i;
  …
  g();
}

g() {
  int i;
  …
  h();
}

h() {
  int i;
  …
  h();
}
```



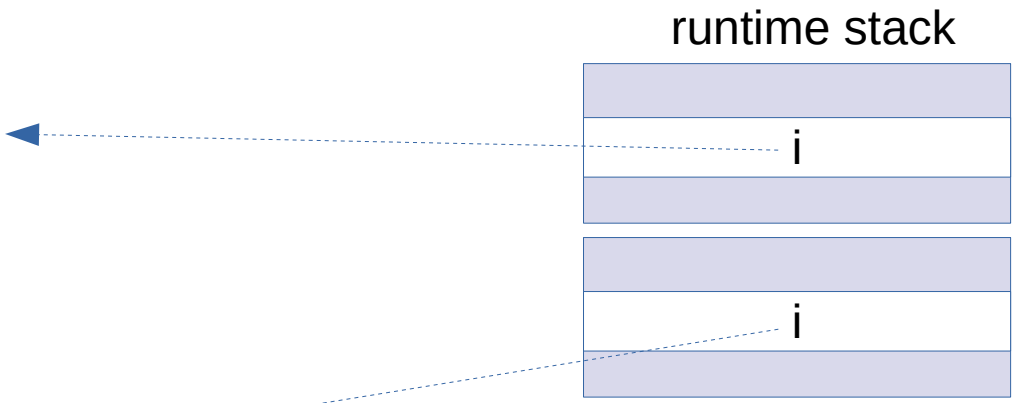
τρέχον call frame κλήσης f()

Παράδειγμα

```
f() {
  int i;
  …
  g();
}

g() {
  int i;
  …
  h();
}

h() {
  int i;
  …
  h();
}
```



call frame κλήσης f()

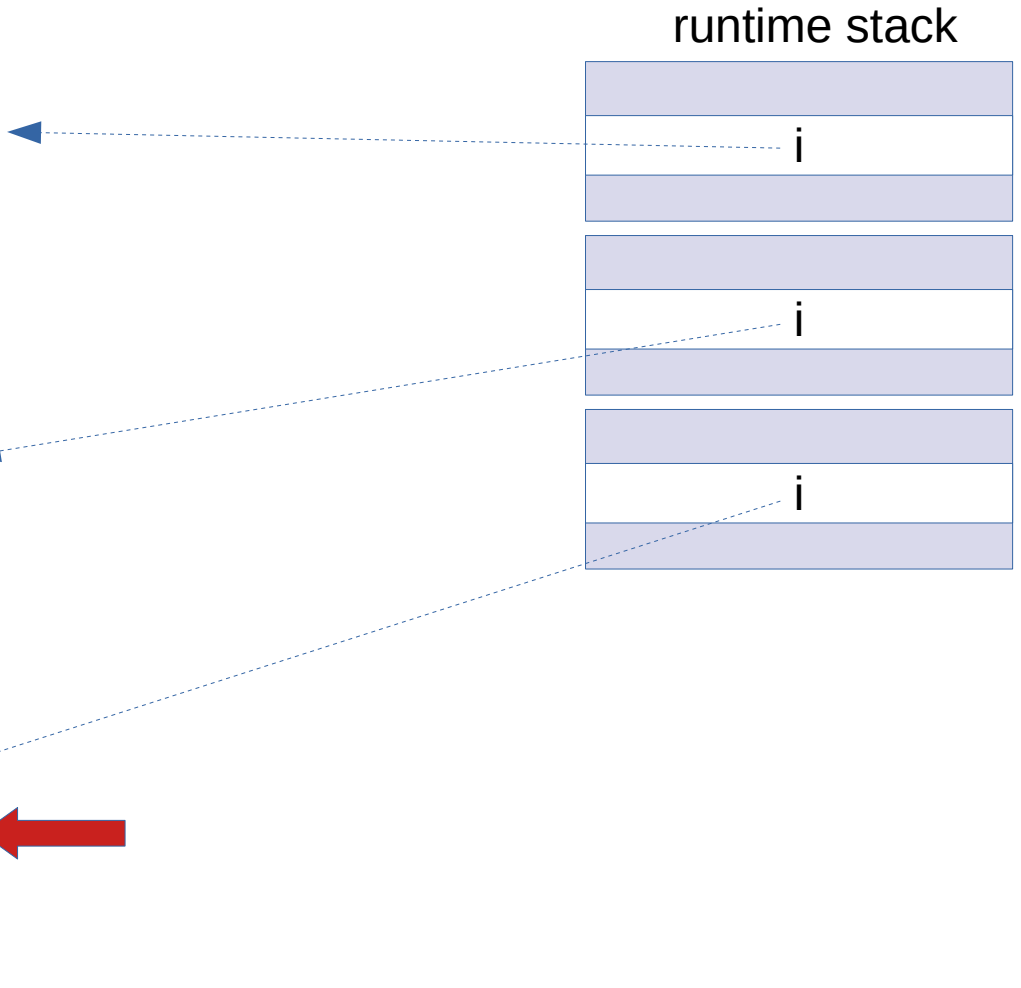
τρέχον call frame κλήσης g()

Παράδειγμα

```
f() {
  int i;
  …
  g();
}

g() {
  int i;
  …
  h();
}

h() {
  int i;
  …
  h();
}
```



call frame κλήσης f()

call frame κλήσης g()

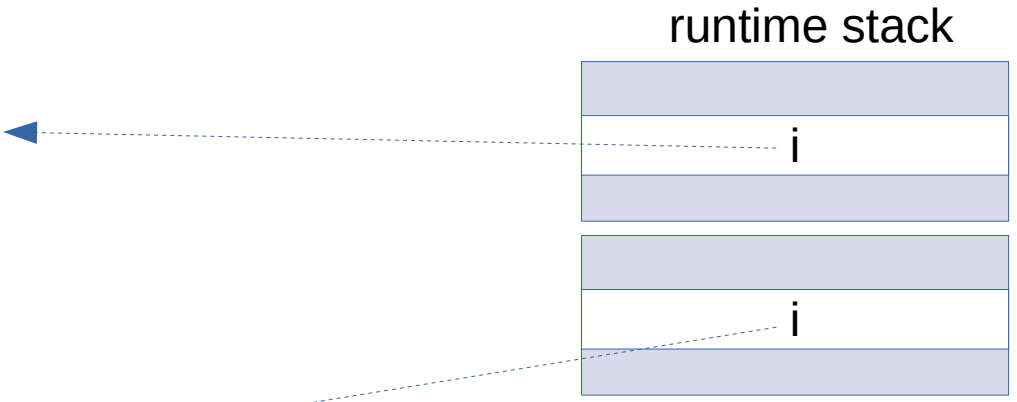
τρέχον call frame κλήσης h()

Παράδειγμα

```
f() {
  int i;
  …
  g();
}

g() {
  int i;
  …
  h();
}

h() {
  int i;
  …
  h();
}
```



call frame κλήσης f()

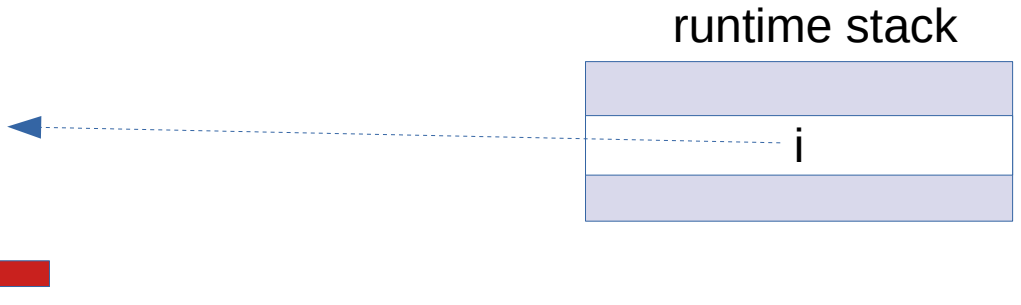
τρέχον call frame κλήσης g()

Παράδειγμα

```
f() {
  int i;
  …
  g();
}

g() {
  int i;
  …
  h();
}

h() {
  int i;
  …
  h();
}
```



τρέχον call frame κλήσης f()

Κλάση RuntimeStack

- Θα τη βρείτε στις βιβλιοθήκες του εργαστηρίου
  - Βεβαιωθείτε ότι έχετε την πιο πρόσφατη έκδοση
- Παρέχει ένα απλό API για τη διαχείριση ξεχωριστών χώρων συμβόλων (τοπικών μεταβλητών και των τιμών τους) ανά κλήση συνάρτησης

```
from compilerlabs import RuntimeStack
```

RuntimeStack API

- Δημιουργία νέου αντικειμένου RuntimeStack

```
rs = RuntimeStack()
```

  - Το νέο runtime stack θα αποτελείται από ένα μοναδικό call frame (του “main”)

RuntimeStack API (2)

- Προσθήκη νέου call frame

```
rs.push_frame(zip(param_list,arg_values))
```

  - Το νέο call frame τοποθετείται στην κορυφή του runtime stack (γίνεται το τρέχον frame)
  - Το προαιρετικό όρισμα του push_frame() αρχικοποιεί το νέο frame με τα ονόματα μεταβλητών του param_list και τις αντίστοιχες τιμές τους (arg_values)
    - Χρησιμοποιείται για να αρχικοποιήσουμε τις παραμέτρους των συναρτήσεων

RuntimeStack API (3)

- Διαγραφή τρέχοντος call frame

```
rs.pop_frame()
```

- Επιστροφή τιμής μεταβλητής στο τρέχον call frame

```
value = rs.dereference(varname)
```

  - Εάν δεν υπάρχει το varname επιστρέφεται None

- Ανάθεση τιμής μεταβλητής στο τρέχον call frame

```
rs.assign(varname,value)
```

RuntimeStack API (4)

- Ανάθεση return value στο τρέχον call frame

```
rs.return_value = value
```

- Ανάκτηση return value από το τρέχον call frame

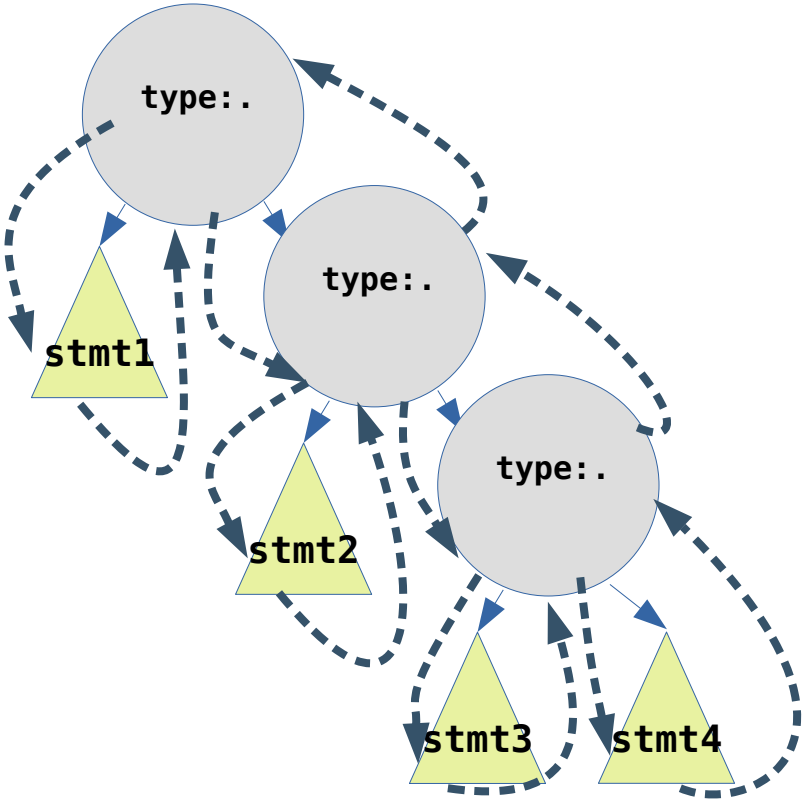
```
rv = rs.return_value
```

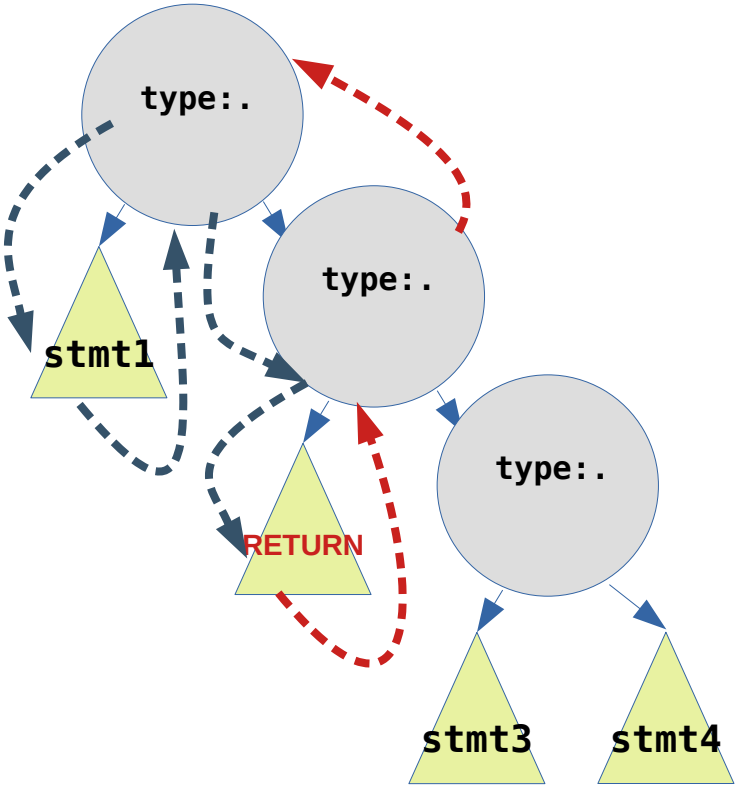
  - Εάν δεν έχει τεθεί ως τώρα, επιστρέφεται None
  - Προσοχή: η ανάκτηση του return_value πρέπει να γίνει πρίν τη διαγραφή του τρέχοντος call frame
    - δηλ. πριν την κλήση του pop_frame()

Διαδικασία κλήσης συνάρτησης

- Έλεγχος: υπάρχει το όνομα της συνάρτησης στον πίνακα συναρτήσεων;
- Έλεγχος: ο αριθμός των παραμέτρων στη δήλωση της συνάρτησης είναι ίσος με τον αριθμό των ορισμάτων κλήσης;
- Υπολογισμός κάθε ορίσματος κλήσης (expression subnodes στο FCALL)
- Προσθήκη (push) ενός νέου call frame, αρχικοποιημένου με τα ονόματα των παραμέτρων και τις αντίστοιχες τιμές των ορισμάτων κλήσης
- Εκτέλεση του AST της καλούμενης συνάρτησης
- Αποθήκευση της επιστρεφόμενης τιμής (return value)
- Αφαίρεση (pop) του call frame
- Επιστροφή του return value

Έλεγχος ροής εκτέλεσης μετά από return





ControlFlow enumeration

- Μέχρι τώρα η εκτέλεση των εντολών δεν επέστρεφε κάποια τιμή
- Τώρα θα χρησιμοποιήσουμε την επιστρεφόμενη τιμή για να ελέγξουμε τη ροή εκτέλεσης μετά το return

```
from enum import Enum
ControlFlow = Enum('ControlFlow','NORMAL RETURN')
```

- Τιμές enumeration
  - ControlFlow.NORMAL → η εκτέλεση συνεχίζεται κανονικά
  - ControlFlow.RETURN → η εκτέλεση διακόπτεται, επιστροφή προς τα πίσω

Εντολές και επιστρεφόμενες τιμές

ASSIGN PRINT FCALL	return ControlFlow.NORMAL
IF	if expr return execute_statement(subnode[1]) else return ControlFlow.NORMAL
IFELSE	if expr return execute_statement(subnode[1]) else return execute_statement(subnode[2])

WHILE	while expr {
	  cf = execute_statement(subnode[1])
	  if (cf==ControlFlow.RETURN) return cf
	}
	return ControlFlow.NORMAL

RETURN	return ControlFlow.RETURN
CONCAT (.)	cf = execute_statement(subnode[0]) if (cf==ControlFlow.RETURN) return cf return execute_statement(subnode[1])