ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Ευφυής Προγραμματισμός

Ενότητα 2: Συναρτήσεις-Δομές Ελέγχου

Ιωάννης Χατζηλυγερούδης
Πολυτεχνική Σχολή
Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Περιεχόμενα ενότητας

Συναρτήσεις-Δομές Ελέγχου :

1. Συναρτήσεις Χρήστη
2. Έλεγχος Ροής Προγράμματος
3. Αναδρομικές Συναρτήσεις
4. Συνάρτηση Let

Συναρτήσεις Χρήστη

Δημιουργία Συνάρτησης (1)

- Ορισμός

(**defun** <function-name> (<param$_1$> … <param$_n$>)
<form$_1$>
…
<form$_m$>)

Τυπικές Παράμετροι (Σύμβολα)

Πχ. (defun mesos-oros(num1 num2 num3)
(/ (+ num1 num2 num3) 3))

Δημιουργία Συνάρτησης (2)

- Κλήση Συνάρτησης:

  >(mesos-oros 3 2 4) → 3

  >(mesos-oros 8 2 4) → 14/3

Πχ. (defun mesos-oros(num1 num2 num3)
(coerce (/ (+ num1 num2 num3) 3) 'float))

>(mesos-oros 8 2 4) → 4.666666666666667

Έλεγχος Ροής Προγράμματος

Διατάξεις Επιλογής (1)

- **Διάταξη Συνθήκης if**

(if <condition> <then form> [<else form>])

- Εκτιμάται η συνθήκη <condition>. Αν το αποτέλεσμα είναι ≠ NIL, εκτιμάται η έκφραση <then form> και επιστρέφεται η τιμή της, αλλιώς η <else form>, αν υπάρχει, αλλιώς NIL.
- Μπορεί να γίνει χρήση των λογικών τελεστών **not**, **or** και **and** στη συνθήκη, αλλά και εν γένει σε μια σ-έκφραση

Διατάξεις Επιλογής (2)

Πχ.(defun mesos-oros-10 (num1 num2 num3)
(if (and (< num1 10)
(< num2 10))
(mesos-oros num1 num2 num3)
nil))

Κλήση:

> (mesos-oros-10 3 10 4) → NIL

> (mesos-oros-10 3 4 5) → 4

Διατάξεις Επιλογής (3)

- **Διάταξη Συνθήκης cond**

(cond (<condition$_1$> <action$_1$>)
(<condition$_2$> <action$_2$>)
. . .
(<condition$_m$> <action$_m$>))

- Εκτιμώνται οι συνθήκες <condition1>, …, <conditionm> με τη σειρά. Με το πρώτο <conditioni> που θα δώσει αποτέλεσμα ≠ NIL, εκτιμάται η αντίστοιχη έκφραση <actioni> και επιστρέφεται η τιμή της. Αν κανένα δεν δώσει τέτοιο αποτέλεσμα, τότε επιστρέφει NIL.

Διατάξεις Επιλογής (4)

Πχ. (defun mesos-oros-10 (num1 num2)
(cond ((and (< num1 10) (< num2 10))
(mesos-oros num1 num2))
(t nil)))

Πχ.(defun where-is (n)
(cond ((equal n 'athens) 'Greece)
((equal n 'New-York) 'USA)
((equal n 'London) 'England)
(t 'unknown)))

(where-is 'athens)→GREECE

Διατάξεις Επανάληψης (1)

- DOLIST

```
(dolist (<parameter>  <list form> <result form>)
                        <form1>
                          …
                        <formn>)
```

- Εκτιμάται το <list form>, που πρέπει να έχει σαν αποτέλεσμα λίστα
- Αποδίδονται τα στοιχεία της λίστας ένα-ένα στο <parameter>.

Διατάξεις Επανάληψης (2)

- Σε κάθε απόδοση εκτελούνται τα <form$_1$> … <form$_n$>.
- Εκτιμάται το <result form>.

- Πχ. (defun add-two (n) (let ((new-list nil))

  (dolist (elem n (reverse new-list))

  (push (+ 2 elem) new-list))))

> (add-two ' (1 2 3)) → (3 4 5)

Διατάξεις Επανάληψης (3)

- DO

```
(do ((<param1> [<init-val1> [<update-form1>]])
                    . . .
     (<paramn> [<init-valn> [<update-formn>]]))
    (<term-test> [<inter-form>] <result-form>)
   <form1>
      …
   <formn>)
```

Διατάξεις Επανάληψης (4)

- Εκτιμώνται τα <init-val$_i$> και καταχωρούνται στις αντίστοιχες παραμέτρους παράλληλα.
- Εξετάζεται το <term-test>. Αν είναι NIL, εκτελείται το σώμα (<form$_1$> … <form$_n$>), αλλιώς επιστρέφεται το <result-form>.
- Εκτιμώνται τα <update-form$_i$> και αποδίδονται στις αντίστοιχες παραμέτρους παράλληλα. Στη συνέχεια εξετάζεται το <term-test> κ.ο.κ.

Διατάξεις Επανάληψης (5)

Πχ. (defun add-two (n)
(do ((new-list n (cdr new-list))
(result nil))
((null new-list) (reverse result))
(push (+ 2 (car new-list)) result)))

Διατάξεις Επανάληψης (6)

DO*

Ό,τι και το do, μόνο που τώρα οι αποδόσεις των αρχικών τιμών και οι ενημερώσεις γίνονται ακολουθιακά.

Διατάξεις Επανάληψης (7)

- LOOP

```
(loop  <form1>  …  <formn>)
```

- Εκτελούνται τα <form1> … <formn> με τη σειρά αναγραφής έως ότου εκτελεστεί κάποια πρόταση return.

Διατάξεις Επανάληψης (8)

- DOTIMES

```
(dotimes (<parameter>  <up-bound form>
<result form>)
   <form1>
      …
   <formn>)
```

Διατάξεις Επανάληψης (9)

- Εκτιμάται το <up-bound form>, που πρέπει να έχει σαν αποτέλεσμα ακέραιο, έστω n.
- Αποδίδονται οι ακέραιες τιμές 0 … n-1 στο <parameter> διαδοχικά.
- Για κάθε τιμή εκτελούνται τα $<form_1>$ … $<form_n>$.
- Εκτιμάται το <result form>.

Διατάξεις Επανάληψης (10)

- Πχ. (defun power (m n)

  (let ((result 1))

  (dotimes (count n result)

  (setf result (* m result)))))

>power(2 3) →8

Αναδρομικές Συναρτήσεις

Αναδρομικές Συναρτήσεις (1)

- Μια συνάρτηση ονομάζεται αναδρομική (recursive) όταν ορίζεται μέσω του εαυτού της αναδρομικός ορισμός.

Αναδρομικές Συναρτήσεις (2)

- Πχ. (defun **filter-pos** (num-list)

```
(cond ((null num-list) nil)
      ((minusp (car num-list))
       (cons (car num-list)
             (filter-pos (cdr num-list))))
      (t (filter-pos (cdr num-list)))))
```

Αναδρομική Κλήση

- > (filter-pos '(1 4 -8 -2)) →(-8 -2)

Αναδρομικές Συναρτήσεις (3)

- Πχ. (defun length-list (list)

  (cond ((null list) 0)

   (t (1+ (length-list (rest list))))))

> (length-list '(a b c d e)) →5

Συνάρτηση Let

Συνάρτηση Let (1)

- **Let:** Ανάθεση αρχικών τιμών σε μεταβλητές

```
(let ((<variable1>  <init-value1>)
  (<variable2>  <init-value2>)
                . . .
  (<variablem>  <init-valuem>))
   <form1>
      …
   <formn>)
```

- Χρησιμοποιείται για δημιουργία και απόδοση τιμών σε τοπικές μεταβλητές

Συνάρτηση Let (2)

- (defun filter-pos (num-list)
  (if (null num-list) nil
  (let ((first-elem (car num-list))
  (rest-list (cdr num-list)))
  (if (minusp first-elem)
  (cons first-elem (filter-pos rest-list))
  (filter-pos rest-list)))))

- >(filter-pos '(-1 3 -4 2))→(-1 -4)

Συνάρτηση Let (3)

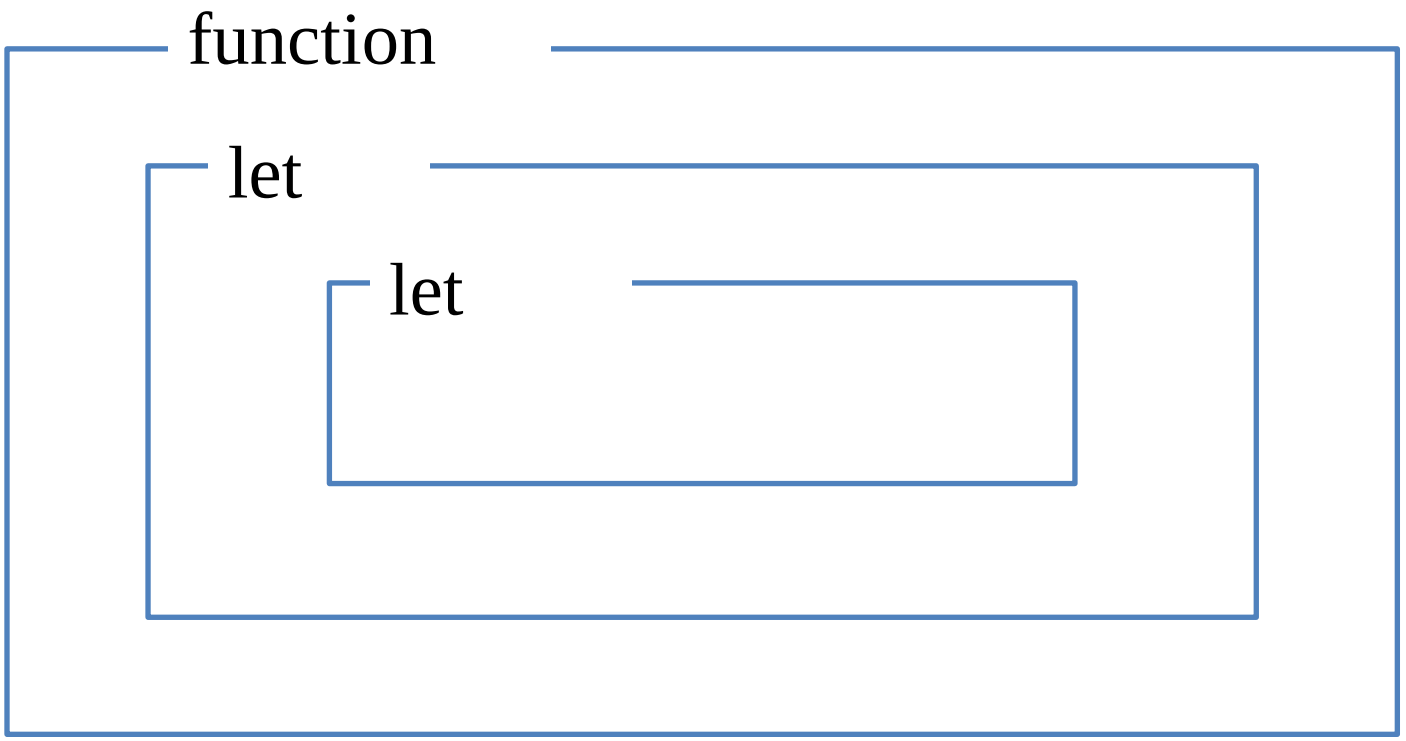
- Το let δρα παράλληλα (πρώτα εκτιμώνται οι αρχικές τιμές και μετά αποδίδονται στις μεταβλητές)
- Το let* δρα ακολουθιακά (κάθε αρχική τιμή εκτιμάται και αποδίδεται στην αντίστοιχη μεταβλητή με τη σειρά αναγραφής)

```
> (setf x 'outside)

> (let  ((x 'inside) (y x)) (list x y)) ↵
>  (INSIDE OUTSIDE)

> (let*  ((x 'inside) (y x)) (list x y)) ↵
>  (INSIDE INSIDE)
```

Συνάρτηση Let (4)

- Τα let, let* δημιουργούν νοητούς φράχτες μέσα σε μια συνάρτηση.
- Επίσης η ίδια η συνάρτηση δημιουργεί ένα νοητό φράχτη γύρω της.
- Μεταβλητές σε ένα φράχτη έχουν νόημα στους εσωτερικούς του φράχτες, όχι όμως αντίστροφα (**τοπικές μεταβλητές**).

Συνάρτηση Let (5)



Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.0.

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.