Ορισμός Συναρτήσεων στην ML

Ονόματα και δεσμεύσεις: η συνάρτηση **val**

- Τα ονόματα σταθερών **δεσμεύονται** με τιμές σταθερών μέσω ορισμών της συνάρτησης **val**.

val codeof0 = ord “0”
val codeof9 = codeof0 + 9

- Τα ονόματα (codeof0, codeof9) δεσμεύονται στις ακέραιες τιμές 48 και 57 (η συνάρτηση ord παράγει τον κώδικα ASCII των χαρακτήρων στους οποίους εφαρμοζεται).

- Το = ορίζει τη δέσμευση συγκεκριμένου ονόματος σε τιμή, και επιπλέον λειτουργικά χρησιμοποιείται για την αντικατάσταση ονόματος/τιμής, όταν γίνεται υπολογισμός εκφράσεων.

- Ένα όνομα σταθεράς μπορεί να θεωρηθεί παρόμοιο με **μεταβλητή** (αλλά προσοχή, δεν είναι μεταβλητή μνήμης, όπως συμβαίνει στις διαδικασιακές γλ. προγ.).

Περιβάλλον και εμβέλεια

- **Περιβάλλον**: ένα σύνολο από δεσμεύσεις ονομάτων σε τιμές, δηλ. ορισμών της συνάρτησης **val**.
- Οι ορισμοί της **val** επεκτείνουν και τροποποιούν το περιβάλλον.
- Οι εκφράσεις υπολογίζονται μέσα σε συγκεκριμένο περιβάλλον.
- **Εμβέλεια ορισμού**: το περιβάλλον μέσα στο οποίο θεωρείται ισχύων ο ορισμός.

Ορισμός συναρτήσεων : **fun**

- Για να οριστεί μια νέα συνάρτηση πρέπει να της αποδοθεί όνομα, και να οριστεί δηλωτικά, ο κανόνας υπολογισμού της.
- Συνήθως συσχετίζεται με τη συνάρτηση μια **παράμετρος** (ένα προσωρινό όνομα για το όρισμά της).

fun double x = x + x
fun treble x = x * 3
fun sixtimes x = double (treble x)

- Η συνάρτηση **fun** χρησιμοποιείται για να οριστούν συναρτήσεις με παραμέτρους. Μπορεί να οριστεί μια συνάρτηση και μέσω της **val,** όταν δεν έχει παραμέτρους (σταθερές συναρτήσεις).

- Η εμβέλεια της παραμέτρου ενός ορισμού συνάρτησης περιορίζεται στον κανόνα υπολογισμού της συνάρτησης. Δηλαδή οι παράμετροι που εμφανίζονται σε διαφορετικούς ορισμούς συναρτήσεων δεν συσχετίζονται.

Παραδείγματα ορισμού συναρτήσεων (τύποι;)

```
fun even n = (n mod 2 = 0)
fun digit d = ord d >= codeof0 & ord d =< codeof9
fun uppercase s = ord s >= ord “A” & ord s =< ord “Z”
fun lowercase s = ord s >= ord “a” & ord s =< ord “z”
fun letter s = lowercase s or uppercase s
fun charofdigit n = chr (n + codeof0)
```

Συναρτήσεις με πλειάδες ορισμάτων

fun evernprod (x, y) = even (x * y)

fun eithereven (x, y) = even x or even y

fun sumbetween (m, n) = ((m+ n) * (abs(m-n) +1)) div 2

fun avepair (x, y) = (x + y) div 2

fun fst (x, y) = x

fun snd (x, y) = y

fun sumdiff (x, y) = (x+y, abs(x-y))

Υπό Συνθήκη Εκφράσεις: **if…then…else**

- Μαθηματικά συχνά ορίζουμε συναρτήσεις μέσω συνθηκών:

$$\min pair(x, y) = \begin{cases} x & if\ x < y \\ y & otherwise \end{cases}$$

- Η ML διαθέτει δομή **if…then…else**

fun minpair (x, y) = **if** x < y **then** x **else** y

Τύπος υπό συνθήκη εκφράσεων

$$\frac{E1 : bool \quad E2 : T \quad E3 : T}{if\ E1\ then\ E2\ else\ E3 : T}$$

- Προσοχή: E2 και E3 **ίδιου τύπου**, διαφορετικά ο τύπος της έκφρασης θα έπρεπε να εξαρτηθεί από την έκβαση του ελέγχου της συνθήκης.

- Ισχύει

if true **then** E2 **else** E3 = E2

if false **then** E2 **else** E3 = E3

Ενσωματωμένες υπό συνθήκη εκφράσεις

- Μπορούμε να ενσωματώσουμε υπό συνθήκη εκφράσεις σε άλλες για να αναπαραστήσουμε πολλαπλές διακλαδώσεις υπολογισμού.

- Όπου n, m τύπου int και s1, s2 τύπου string:
  - m + (**if** m = 0 **then** n **else** n – m)
  - **if** m = n **then** m **else if** m < n **then** avepair (m, n) **else** (m + n) div 2
  - size (**if** s1 = s2 **then** s1^s2 **else** s1^S1)

Εναλλακτικός τρόπος ορισμού συναρτήσεων με πολλαπλά ορίσματα

fun minmaxpair (x, y) = **if** x < y **then** (x, y) **else** (y, x)

fun maxpair anypair = snd(minmaxpair anypair)

Τι είναι το anypair στον ορισμό αυτό;

Υπολογίστε maxpair (3, 6)

Εναλλακτικός τρόπος ορισμού συναρτήσεων με περιπτώσεις

```
fun     not true = false
    |   not false = true
fun     true or b = true
    |   false or b = b
fun     true & b = b
    |   false & b = false
```

- Αντί για υποθετικές εκφράσεις, μπορούμε να ορίσουμε μια συνάρτηση παραθέτοντας διαφορετικές περιπτώσεις, ανάλογα με συγκεκριμένες σταθερές τιμές που μπορεί να δεχτούν ως όρισμα.
- Οι περιπτώσεις διαχωρίζονται με |

Infix και Prefix μορφές

```
fun   not true = false
  |   not false = true
fun   true or b = true
  |   false or b = b
fun   true & b = b
  |   false & b = false
```

```
fun or (true, b) = true
  | or (false, b) = b
fun & (true, b) = b
  | & (false, b) = false
```

Υπολογισμός εκφράσεων με «ταίριασμα» παραμέτρων

- Όταν υπολογίζονται εκφράσεις, οι σταθερές τιμές για τις οποίες γίνεται ο υπολογισμός «ταιριάζουν»/ενοποιούνται με τις παραμέτρους που περιέχονται στον ορισμό της συνάρτησης.

(not true) or false =

false or false (1η περίπτωση not)

false (2η περίπτωση or, b / false)

Ενοποίηση παραμέτρων

- Μπορούμε να χρησιμοποιήσουμε όσες περιπτώσεις θέλουμε για τον ορισμό συναρτήσεων, και όποιο συνδυασμό παραμέτρων επιθυμούμε (σταθερές, πλειάδες, μεταβλητές), αρκεί:
  - οι παράμετροι να έχουν τύπο συμβατό με τον τύπο του πεδίου ορισμού της συνάρτησης
  - οι περιπτώσεις να παρατίθενται όλες, δηλ. οι παράμετροι να καλύπτουν όλα τα δυνατά ορίσματα
  - να μην υπάρχει ασάφεια ως προς το ποια περίπτωση ισχύει – σε περίπτωση που περισσότερες περιπτώσεις καλύπτουν τα ορίσματα της συνάρτησης, υποθέτουμε ότι υπάρχει από πάνω προς τα κάτω προτεραιότητα στις περιπτώσεις.

Παράδειγμα
Υπολογισμός σε περιβάλλον, ορισμός με περιπτώσεις

```
val first = “A”
val second = “B”
val third =“C”
val coffee = “coffee”
val tea = “tea”
val water = “water”
fun chose first = coffee
      |  chose second = tea
      |  chose third = water
fun value coffee = 1
      | value tea = 2
      | value water =0
      |value coffee = 3
fun atp (choice, quantity) = quantity * value chose choice
```

- Υπολογίστε atp (A, 2) + atp (B, 1)

Αναδρομικοί ορισμοί: αυτοαναφορά

stringcopy : int x string → string

- Για κάθε n >=0 και string s, stringcopy (n, s) παράγει string από τη συνένωση n αντιγράφων του s.
- stringcopy(3, “xyz”)= “xyzxyzxyz”.

```
fun stringcopy (n, s) =    if n=0
                           then “”
                           else stringcopy(n-1, s) ^ s
```

Υπολογισμός με αναδρομικό ορισμό: επανεγγραφή με αντικατάσταση, εμβέλεια

stringcopy (2, “ab”)
= **if** 2 = 0 **then** “” **else** stringcopy (2-1, “ab”) ^ “ab”
= **if** false **then** “” **else** stringcopy (1, “ab”) ^ “ab”
=stringcopy (1, “ab”) ^ “ab”
= (**if** 1=0 **then** “” **else** stringcopy (1-0, “ab”) ^ “ab”) ^ “ab”
= (**if** false **then** “” **else** stringcopy (0, “ab”) ^ “ab”) ^ “ab”
=(stringcopy(0, “ab”) ^ “ab”) ^ “ab”
= ((**if** 0 = 0 **then** “” **else** stringcopy (0-1, “ab”) ^ “ab”) ^ “ab”) ^ “ab”
= ((**if** true **then** “” **else** stringcopy (0-1, “ab”) ^ “ab”) ^ “ab”) ^ “ab”
= (“” ^ “ab”) ^ “ab”
=“ab” ^ “ab”
=“abab”