Συναρτησιακός Προγραμματισμός με το Mathematica

Στις "παραδοσιακές" γλώσσες προγραμματισμού, όπως οι FORTRAN, Pascal, C, κ.λπ. τα προγράμματα βασίζονται κυρίως στη μεταβολή των τιμών των μεταβλητών του προγράμματος. Οι τιμές των μεταβλητών καθορίζουν την κατάσταση του προγράμματος. Κατά την εκτέλεση του προγράμματος, το πρόγραμμα βρίσκεται σε μια αρχική κατάσταση η οποία καθορίζεται από τις αρχικές τιμές των μεταβλητών του. Στη συνέχεια, μέσω εντολών ανάθεσης της μορφής var = exp ή var:=expr οι μεταβλητές αλλάζουν τιμή μεταβάλλοντας την κατάσταση του προγράμματος. Οι εντολές εκτελούνται η μία μετά την άλλη. Επιπλέον, εντολές ελέγχου και επανάληψης, όπως οι If και While επιτρέπουν την αλλαγή της σειράς εκτέλεσης των εντολών ανάλογα με την κατάσταση του προγράμματος. Έτσι, ένα πρόγραμμα δεν είναι τίποτε άλλο από μια σειρά εντολών οι οποίες αλλάζουν την κατάσταση του προγράμματος. Επιπλέον, οι εντολές αυτές οργανώνονται σε μονάδες προγραμμάτων ή διαδικασίες (υπορουτίνες και συναρτήσεις). Έτσι, το παραδοσιακό υπόδειγμα προγραμματισμού, όπως εκφράζεται από τις γλώσσες που προαναφέρθηκαν, ονομάζεται διαδικασιακό ή προστακτικό.

Ο συναρτησιακός προγραμματισμός αποτελεί ένα διαφορετικό υπόδειγμα προγραμματισμού. Σε μια "καθαρά" συναρτησιακή γλώσσα, ένα πρόγραμμα είναι στην πραγματικότητα μια συνάρτηση η οποία δέχεται ως παράμετρο τις αρχικές τιμές των δεδομένων. Δεν υπαρχουν εντολές οι οποίες να αλλάζουν τις τιμές των μεταβλητών παρά μόνο συναρτήσεις οι οποίες χρησιμοποιούνται όπως τα βασικά αντικείμενα στις προστακτικές γλώσσες: Συμμετέχουν σε εκφράσεις υπολογισμού, περνιούνται ως παράμετροι επιστρέφονται ως αποτέλεσμα άλλων συναρτήσεων. Η επανάληψη στις συναρτησιακές γλώσσες υλοποιείται μέσω της αναδρομής, δηλαδή του ορισμού μιας συνάρτησης με τρόπο ώστε να περιλαμβάνει τον εαυτό της, όπως θα φανεί και στο παράδειγμα που ακολουθεί.

Ας σημειωθεί ότι η έννοια της συνάρτησης στο συναρτησιακό προγραμματισμό είναι διαφορετική από αυτή στον διαδικασιακό. Στον πρώτο, μια συνάρτηση ορίζει μια έκφραση και αντιστοιχεί στην μαθηματική έννοια του όρου. Στον δεύτερο, μια συνάρτηση είναι μια ακολουθία εντολών το οποίο ανταλλάσσει δεδομένα με τις υπόλοιπες μονάδες του προγράμματος.

Για λόγους που αφορούν στη δυνατότητα χειρισμού μαθηματικών εκφράσεων με συμβολικό όσο και αριθμητικό τρόπο, το Mathematica υποστηρίζει και τα δύο στυλ προγραμματισμού, συναρτησιακο και διαδικασιακό. Δίνουμε στη συνέχεια δύο παραδείγματα όπου υλοποιείται μια συνάρτηση για τον υπολογισμό του παραγοντικού ενός ακεραίου χρησιμοποιώντας τεχνικές του διαδικασιακού και του συναρτησιακού προγραμματισμού:

Σε διαδικαστικό στυλ προγραμματισμού, μια συνάρτηση για τον υπολογισμό του παραγοντικού είναι η ακόλουθη:

```
fact1[n_Integer] := (x = 1; i = n; While[i > 0, x = x * i; i = i - 1]; x);
fact1[5]
```

```
120
```

Σε συναρτησιακό στυλ προγραμματισμού, η συνάρτηση του παραγοντικού ορίζεται ως εξής:

```
fact2[n_Integer] := If[n > 1, n * fact2[n - 1], 1];
fact2[5]
```

```
120
```

Συναρτήσεις ανώτερης τάξης

Στο συναρτησιακό στυλ προγραμματισμού οι συναρτήσεις αποτελούν *πρώτης τάξης αντικείμενα*. Αυτό σημαίνει, όπως αναφέρθηκε προηγουμένως, ότι είναι δυνατόν να είναι παράμετροι, ακόμη και να επιστρέφονται από συναρτήσεις. Οι συναρτήσεις που χειρίζονται συναρτήσεις με τον παραπάνω τρόπο λέγονται *συναρτήσεις ανώτερης τάξης*. Ως παράδειγμα δίνουμε μια συνάρτηση Deriv η οποία δέχεται ως παράμετρο μια *συνάρτηση* **f** και επιστρέφει μια προσέγγιση της πρώτης παραγώγου της συνάρτησης:

```
Deriv[f_, x_] := ((f[x + 0.001] - f[x]) / 0.001)
```

Η **Deriv** δέχεται ως παράμετρο μια συνάρτηση **f** και μια τιμή, **x**, για την οποία θα υπολογίσει την πρώτη παράγωγο σύμφωνα με τη σχέση $\frac{\mathrm{df}(x)}{\mathrm{dx}} = \frac{f(x+\mathrm{dx})-f(x)}{\mathrm{dx}}$ για **dx=0.001**. Προσέξτε ότι η **Deriv** δέχεται ως παράμετρο την ίδια την συνάρτηση f, και ότι κάποια τιμή της. Στο παράδειγμα που ακολουθεί η Deriv υπολογίζει την παράγωγο της συνάρτησης sin στην τιμή 0.

```
Deriv[Sin, 0]
```

```
1.
```

Παρατηρήστε ότι η πρώτη παράμετρος της Deriv είναι η συνάρτηση Sin. Η Deriv είναι μια συνάρτηση ανώτερης τάξης, καθώς επεξεργάζεται άλλες συναρτήσεις. Από τη Μαθηματική σκοπιά, η παράγωγος είναι ένας τελεστής: Στο συναρτησιακό υπόδειγμα προγραμματισμού, *"τα πάντα είναι συναρτήσεις"* οπότε είναι φυσικό ο τελεστής της παραγώγου να υλοποιείται ως μια συνάρτηση ανώτερης τάξης, η οποία εφαρμόζεται σε άλλες συναρτήσεις.

Η *συνάρτηση* Function

Η *συνάρτηση* Function *επιτρέπει τη δημιουργία μιας συνάρτησης από μια παράσταση*. Οι *μορφές της* Function *είναι η ακόλουθη*:

```
Function[σώμα]
Function[x, σώμα]
Function[{x1, x2, ...}, σώμα]
```

όπου *σώμα είναι το σώμα της συνάρτησης η οποία δημιουργείται και* x, x_1, x_2, … *είναι παράμετροι της συνάρτησης*. Μια *συνάρτηση που δημιουργείται με αυτό τον τρόπο δεν έχει όνομα* και ονομάζεται *καθαρή* (pure) *συνάρτηση*.

Παραδείγματα *χρήσης της* Function *είναι τα ακόλουθα*:

```
Function[x, x^2][3]
Function[x, x^2][y]
```

```
9
```

```
y^2
```

Στο παράδειγμα, η έκφραση **Function[x,x^2]** παριστάνει τη συνάρτηση η οποία δέχεται ως παράμετρο έναν αριθμό και επιστρέφει το τετράγωνο αυτού του αριθμού. Η κλήση της x παραπάνω συνάρτησης με παράμετρο μια σταθερά επιστρέφει την τιμή της συνάρτησης για αυτή τη σταθερά, ενώ η κλήση με παράμετρο επιστρέφει μια παράσταση ως αποτέλεσμα.

```
Squares := Function[{x, y}, x^2 + y^2];
Squares[a, b]
Squares[3, 4]
```

```
a^2 + b^2
```

```
25
```

Στο παραπάνω παράδειγμα, Squares είναι η συνάρτηση με δύο παραμέτρους η οποία επιστρέφει το άθροισμα των τετραγώνων των παραμέτρων της.

Όπως αναφέρθηκε παραπάω, στις συναρτησιακές γλώσσες είναι δυνατόν μια συνάρτηση να επιστρέφει μια άλλη συνάρτηση ως αποτέλεσμα. Σε προηγούμενο παράδειγμα, η συνάρτηση Deriv δέχεται μια συνάρτηση ως παράγωγο και επιστρέφει ως αποτέλεσμα μια πραγματική τιμή. Στη συνέχεια φαίνεται πώς με χρήση της **Function** η Deriv μπορεί να τροποποιηθεί ώστε να επιστρέφει μια νέα συνάρτηση, την παράγωγο αυτής που δέχεται ως παράμετρο.

```
Deriv2[f_] := Function[{x}, (f[x + 0.001] - f[x]) / 0.001]
```

```
MyCos = Deriv2[Sin];
MyCos[0]
```

```
1.
```

Η **Deriv2** *δέχεται την παράμετρο* f *και επιστρέφει την παράγωγό της*. Η *συνάρτηση* MyCos *προκύπτει ως το αποτέλεσμα της* **Deriv2** *αν δεχτεί ως παράμετρο την συνάρτηση* Sin.

Η *συνάρτηση* Nest

Η *συνάρτηση* Nest *εφαρμόζει μια συνάρτηση* f *πάνω σε μια έκφραση* n *φορές*:

```
Nest[f, έκφραση, n]
```

Σ*το ακόλουθο παράδειγμα*

```
Nest[f, x, 3]
```

```
f[f[f[x]]]
```

η συνάρτηση f *εφαρμόζεται στη μεταβλητή* x *τρείς φορές*.

Σ*το παράδειγμα που ακολουθεί η* Nest *εφαρμόζεται για τον υπολογισμό του συνεχούς κλάσματος*:

```
Nest[Function[{q}, 1 / (1 + q)], x, 4]
```

$$\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+x}}}}$$

Εκφράσεις και λίστες

Ο *συναρτησιακός προγραμματισμός είτε με το Mathematica είτε με οποιαδήποτε άλλη γλώσσα προγραμματισμού που τον υποστηρίζει είναι ιδιαίτερα κατάλληλος για συμβολικούς υπολογισμούς* (symbolic computing). Μ*ε τον όρο αυτό αναφερόμαστε στην επεξεργασία μαθηματικών εκφράσεων όχι αριθμητικά*, *με βάση τις τιμές που παίρνουν οι μεταβλητές μιας έκφρασης*, *αλλά με το μετασχηματισμό των εκφράσεων αυτών από μια μορφή στην άλλη χρησιμοποιώντας κατάλληλους αλγορίθμους*. Η *συμβολική επίλυση εξισώσεων* (Solve), *οι εντολές απλοποίησης* (Simplify) *και ανάπτυξης* (Expand) *εκφράσεων αποτελούν παραδείγματα συμβολικού υπολογισμού από το Mathematica*. Οι *συμβολικοί υπολογισμοί στο Mathematica πραγματοποιούνται με τον κατάλληλο χειρισμό εκφράσεων και λιστών*. Μ*ια έκφραση στο Mathematica έχει μια δομή δένδρου*. Γ*ια μια έκφραση*, *η δομή αυτή εκτυπώνεται με χρήση της συνάρτησης* FullForm:

```
FullForm[a + b]
FullForm[Sqrt[(a + 1) ^ 2 - 2 (b ^ 2 + 3) ^ 2]]
```

```
Plus[a, b]
```

```
Power[Plus[Power[Plus[1, a], 2],
  Times[-2, Power[Plus[3, Power[b, 2]], 2]]], Rational[1, 2]]
```

Κάθε έκφραση έχει την κεφαλή της (head) και μια σειρά από μέλη, τα οποία με τη σειρά τους είναι δυνατόν να αποτελούν εκφράσεις. Έτσι η δομή μιας έκφρασης είναι δυνατόν να εκτείνεται σε πολλά επίπεδα. Στην έκφραση a+b φαίνεται παραπάνω ότι η κεφαλή είναι η Plus, ενώ τα μέλη της έκφρασης είναι τα a,b. Η κεφαλή μιας έκφρασης δίνεατι από τη συνάρτηση Head:

```
Head[a + b]
```

```
Plus
```

Η ίδια δομή ισχύει όπως φαίνεται στα παραπάνω παραδείγματα και για τις λίστες. FullForm αναδεικνύει τη δομή τους, όπως φαίνεται στα παραδείγματα που ακολουθούν:

```
FullForm[{1, 3, 6}]
FullForm[{a, {b, c}, {d, e, f}}]
```

```
List[1, 3, 6]
```

```
List[a, List[b, c], List[d, e, f]]
```

Προσέξτε ότι μια λίστα είναι δυνατόν να περιέχει άλλες λίστες, δίνοντας πολλά επίπεδα διάρθρωσης. Η Head εφαρμόζεται και σε μια λίστα:

```
Head[{a, b, c}]
```

```
List
```

Η συνάρτηση Map

Η Map δέχεται ως παράμετρο μια συνάρτηση και μια έκφραση (ή λίστα) και εφαρμόζει τη συνάρτηση στα στοιχεία αυτής της έκφρασης. Η πιο απλή μορφή της Map είναι η ακόλουθη:

```
Map[f, έκφραση]
```

όπου εφαρμόζεται η συνάρτηση f στα στοιχεία της έκφρασης (ή λίστας) στο πρώτο επίπεδο διάρθωσης της έκφρασης. Στα παραδείγματα που ακολουθούν η συνάρτηση Sqrt εφαρμόζεται σε κάθε στοιχείο μιας λίστας.

```
Map[Sqrt, {1, 4, 9, 16}]
Map[Sqrt, {a, b, c}]
```

```
{1, 2, 3, 4}
```

$$\{\sqrt{a}, \sqrt{b}, \sqrt{c}\}$$

Η Map *εφαρμόζεται στο πρώτο επίπεδο διάρθωσης αλγεβρικών εκφράσεων*:

```
Map[Sqrt, a + b + c + d]
Map[Cos,  a + (b + c) ^ 2  +  Pi ^ 3]
```

$$\sqrt{a} + \sqrt{b} + \sqrt{c} + \sqrt{d}$$

```
Cos[a] + Cos[(b + c)^2] + Cos[π^3]
```

Μι*α συνηθισμένη χρήση της συνάρτησης* Function *είναι ο ορισμός μιας ανώνυμης συνάρτησης ως παραμέτρου στη* Map:

```
Map[Function[{x}, x ^ 2], a + b + c]
```

$$a^2 + b^2 + c^2$$

Η *παραπάνω χρήση της* Map *έχει την εξής σημασία*: Εφάρμοσε *τη συνάρτηση η οποία υπολογίζει το τετράγωνο ενός αριθμού σε κάθε όρο του αθροίσματος* a+b+c.

```
Map[f, έκφραση, επίπεδο]
```

όπου το επίπεδο είναι ένας αριθμός ο οποίος ορίζει το "βάθος" στη δομή μιας έκφρασης στην οποία θα εφαρμοστεί η συνάρτηση:

```
Map[Function[{x}, x ^ 3],  Cos[a + b] + Sin[a + c], 2]
```

$$\text{Cos}[(a + b)^3]^3 + \text{Sin}[(a + c)^3]^3$$

Α*ν θέλουμε να εφαρμοστεί η συνάρτηση σε κάθε στοιχείο μιας έκφρασης χρησιμοοιούμε την* MapAll:

```
MapAll[Function[{x}, x^2], (a + b) * c + Sqrt[d^2 + e^2]]
```

$$\left((a^2+b^2)^4\, c^4+\sqrt{(d^{16}+e^{16})^2}\right)^2$$

Η *συνάρτηση* Apply

Η *συνάρτηση* Apply *αντικαθιστά το* Head *μιας έκφρασης* ή *λίστας με μια συνάρτηση*:

```
Apply[f, έκφραση]
```

Σ*το παράδειγμα*, *το* head *της λίστας* (List) *αντικαθίσταται από τη συνάρτηση* Plus.

```
Apply[Plus, {a, b, c}]
```

```
a + b + c
```