Κεφάλαιο 10ο: Διαδικασιακός Προγραμματισμός

10.1 Ανάθεση τιμών σε μεταβλητές

Ο τελεστής ανάθεσης (=, :=) χρησιμοποιείται για να τοποθετήσουμε το αποτέλεσμα μιας έκφρασης (τιμή ή παράσταση) σε μια μεταβλητή. Η σύνταξή του έχει ως εξής:

var = expr	Κατά τη διάρκεια εκτέλεσης της εντολής ο Η/Υ υπολογίζει πρώτα το αποτέλεσμα της έκφρασης expr και στη συνέχεια, αυτό που θα βρει, το αναθέτει στη μεταβλητή var που υπάρχει αριστερά του =. Στη συνέχεια του προγράμματος κάθε φορά που θα βλέπει την μεταβλητή var θα την αντικαθιστά από την τιμή έκφρασης expr.
Set[var, expr]:	Ισοδύναμη με την προηγούμενη ανάθεση τιμών.
var := expr	Η ανάθεση της τιμής της έκφρασης expr στην μεταβλητή var γίνεται την στιγμή που καλούμε την μεταβλητή var.
SetDelayed[var, expr]:	Ισοδύναμη με την προηγούμενη ανάθεση τιμών.

Το παραπάνω ίσον (=) λοιπόν και στις δύο περιπτώσεις αναφέρεται σε ανάθεση τιμής και όχι σε ισότητα. Το περιεχόμενο της μεταβλητής που υπήρχε πριν από την εντολή της ανάθεσης τιμής χάνεται.

```
A = Random@Integer, 81, 15<D;
B := Random@Integer, 81, 15<D;
A1 = Table@A, 85<D
B1 = Table@B, 85<D

85, 5, 5, 5, 5<
```

```
810, 11, 7, 6, 15<
```

Η πρώτη εντολή υπολογίζει έναν τυχαίο ακέραιο αριθμό μεταξύ 1 και 15 (π.χ. τον 7) και τον τοποθετεί στη μεταβλητή Α. Αντίθετα στη δεύτερη εντολή ορίζουμε ότι η μεταβλητή Β θα δεχτεί έναν τυχαίο ακέραιο αριθμό μεταξύ 1 και 15 όταν την καλέσουμε. Συνεπώς στην τρίτη εντολή επειδή η μεταβλητή Α έχει ήδη την τιμή 7, όταν προσπαθούμε να πάρουμε έναν πίνακα με 5 αριθμούς ίσους με Α, παίρνει και τους 5 ίσους με 7. Αντίθετα όταν προσπαθούμε να δημιουργήσουμε έναν πίνακα με 5 αριθμούς της μορφής Β, κάθε φορά που καλούμε τη μεταβλητή Β για να την τοποθετήσουμε στο πίνακα, υπολογίζεται η έκφραση που βρίσκεται δεξιά του ίσον στην δεύτερη εντολή και το αποτέλεσμα τοποθετείται στον πίνακα Β1. Αυτός είναι και ο λόγος που ο πίνακας Β1 έχει διαφορετικές τυχαίες τιμές.

```
Set@z, Expand@Hx + yL^2DD
```

$$x^2 + 2\,x\,y + y^2$$

```
SetDelayed@w, Expand@Hx + yL^2DD
w
```

$$x^2 + 2\,x\,y + y^2$$

Παρατηρούμε ότι το z και το w έχουν τις ίδιες τιμές. Αν όμως στη συνέχεια θέσουμε όπου x = 1 + a τότε θα έχουμε:

```
x = 1 + a
```

$$1 + a$$

```
z
```

$$H1 + aL^2 + 2\,H1 + aL\,y + y^2$$

```
w
```

$$1 + 2\,a + a^2 + 2\,y + 2\,a\,y + y^2$$

Η διαφορά αυτή οφείλεται στο γεγονός ότι το z έχει ήδη την τιμή του αναπτύγματος $Hx + yL^2$ δηλαδή $x^2 + 2\,xy + y^2$ και απλώς αντικαθιστά το x με ισοδύναμή του έκφραση 1 + a στο ανάπτυγμα αυτό. Αντίθετα όταν καλέσουμε το w τότε τοποθετείται στην έκφραση $Hx + yL^2$ το x = 1 + a και στη συνέχεια υπολογίζεται το ανάπτυγμα του $Hx + yL^2$.

```
A = 1
```

$$1$$

```
A = A + 1
```

$$2$$

Η πρώτη εντολή δημιουργεί μια θέση στην μνήμη του υπολογιστή για τη μεταβλητή Α και τοποθετεί σε αυτή την τιμή 1. Η δεύτερη εντολή βρίσκει το αποτέλεσμα δεξιά του ίσον (=) που είναι 1 + 1 (αφού η τιμή της μεταβλητής Α από την πρώτη εντολή είναι 1) και το τοποθετεί στην ίδια θέση μνήμης που έχει δημιουργήσει στην πρώτη εντολή για να αποθηκεύει την τιμή της μεταβλητής Α. Παρατηρούμε, δηλαδή, ότι οτιδήποτε βρίσκεται στην μεταβλητή που βρίσκεται αριστερά του ίσον, χάνεται και στη θέση του τοποθετείται αυτό που βρίσκεται δεξιά του ίσον.

Η εντολή A = A + 1 μπορεί να αντικατασταθεί με άλλες εντολές πιο σύντομες και μερικές φορές περισσότερο λειτουργικές:

a ++	Αυξάνει την τιμή της μεταβλητής a κατά 1, και επιστρέφει την προηγούμενη τιμή του a.
a --	Μειώνει την τιμή της μεταβλητής a κατά 1, και επιστρέφει την προηγούμενη τιμή του a.
++ a	Αυξάνει την τιμή της μεταβλητής a κατά 1, και επιστρέφει την νέα τιμή του a.
-- a	Μιεώνει την τιμή της μεταβλητής a κατά 1, και επιστρέφει την νέα τιμή του a.
a += da	Αυξάνει την τιμή της μεταβλητής a κατά da (δηλαδή a = a + da), και επιστρέφει την νέα τιμή του a.
a -= da	Μειώνει την τιμή της μεταβλητής a κατά da (δηλαδή a = a - da), και επιστρέφει την νέα τιμή του a.
a *= c	Πολλαπλασιάζει την τιμή της μεταβλητής a με c, τοποθετεί το αποτέλεσμα στην μεταβλητή a και επιστρέφει την νέα τιμή του a.
a /= c	Διαιρεί την τιμή της μεταβλητής a με c, τοποθετεί το αποτέλεσμα στην μεταβλητή a και επιστρέφει την νέα τιμή του a.

Ας δούμε μερικές εφαρμογές των παραπάνω εντολών:

```
A = 1;
A++
```

```
1
```

```
A
```

```
2
```

```
++A
```

```
3
```

```
A--
```

```
3
```

```
A
```

```
2
```

```
--A
```

```
1
```

```
A += 5
```

```
6
```

```
A -= 3
```

```
3
```

```
A  = 4
```

```
12
```

```
A ê= 6
```

```
2
```

Μπορούμε να αναθέσουμε την ίδια τιμή σε παραπάνω από μία μεταβλητές χρησιμοποιώντας την εντολή ανάθεσης:

var1 = var2 = ... = expr

```
x = y = 1
```

```
1
```

```
x + y
```

```
2
```

10.2 Εντολές Εισόδου - Εξόδου

Με τις εντολές εισόδου που διαθέτει το *Mathematica* μπορούμε να αναθέσουμε μία τιμή σε μια μεταβλητή πληκτρολογώντας την. Η συνάρτηση **Input** που διαθέτει το *Mathematica* μπορεί να χρησιμοποιηθεί ως μια εντολή εισόδου σε ένα πρόγραμμα.

var = **Input**[ ]	Ζητάει με διαλογική μορφή (μέσω παραθύρου) μια τιμή την οποία την αναθέτει στη μεταβλητή var.
var = **Input**["Μήνυμα"]	Ζητάει με διαλογική μορφή (μέσω παραθύρου) μια τιμή την οποία την αναθέτει στη μεταβλητή var, ενώ εμφανίζει οτ συγκεκριμένο μήνυμα για τον χρήστη στο παράθυρο διαλόγου.

Η εκτύπωση αποτελεσμάτων στο *Mathematica* γίνεται απλώς γράφοντας το όνομα της μεταβλητής που θέλουμε να εκτυπώσουμε είτε χρησιμοποιώντας την συνάρτηση **Print**[ ] που διαθέτει το *Mathematica.*

Print[έκφραση 1, έκφραση2 , ...]	Εκτυπώνει το σύνολο των εκφράσεων που υπάρχει μέσα στην Print και στη συνέχεια αλλάζει γραμμή.

Αν θέλουμε να εφαμόσουμε μια συγκεκριμένη μορφοποίηση στην εκτύπωσή μας μπορούμε να χρησιμοποιήσουμε τη συνάρτηση **StylePrint**[ ].

Στη συνέχεια θα δώσουμε δύο απλά παραδείγματα εισόδου - εξόδου:

Παράδειγμα 1: Να γράψετε πρόγραμμα που να διαβάζει από το πληκτολόγιο δύο ακεραίους και να υπολογίζει και να εκτυπώνει το άθροισμα τους.

```
a = Input@"Give one Integer"D;
b = Input@"Give one Integer"D;
c = a + b;
Print@"Το άθροισμα είναι=", cD
```

Παράδειγμα 2: Να γράψετε πρόγραμμα που να διαβάζει από το πληκτολόγιο μία λίστα από πραγματικούς αριθμούς και να υπολογίζει και να εκτυπώνει τον μέσο όρο των στοιχείων της λίστας.

```
H a=Λίστα L
H mo=Μέσος όρος L
a = Input@"Δώσε τη Λίστα="D;
mo = Sum@a@@iDD, 8i, 1, Length@aD<D ê Length@aD;
Print@"Μέσος όρος=", moD
```

Στα παραπάνω παραδείγματα έχουμε να κάνουμε τις εξής παρατηρήσεις:
α. Οι εντολές δόθηκαν στο ίδιο cell (κελί) με την σειρά, πατώντας Enter στο τέλος κάθε γραμμής.
β. Τα σχόλια τοποθετούνται ανάμεσα στα σύμβολα (* και *).
γ. Το ερωτηματικό στο τέλος της εντολής σημαίνει ότι ναι μεν η εντολή εκτελείται αλλά το αποτέλεσμα δεν εμφανίζεται στην οθόνη.

10.3 Εντολές Ελέγχου

10.3.1 Η εντολή If

If [συνθήκη, εντολή(-ές) 1, εντολή(-ές)2, εντολή(-ές)3]

Αν η συνθήκη είναι αληθής, τότε θα εκτελεστεί η εντολή(-ές)1, αν είναι ψευδής θα εκτελεστεί η εντολή(-ές)2, ενώ αν δεν μπορεί το *Mathematica* να εκφέρει γνώμη για τον αν η συνθήκη είναι αληθής ή ψευδής εκτελείται η εντολή(-ές)3. Οι εντολές 2 και 3 είναι προεραιτικές.

Για την διατύπωση της συνθήκης στη σύνταξη της εντολής **If** θα χρειαστούμε τους παρακάτω τελεστές σύγκρισης και λογικούς τελεστές:

Τελεστές Σύγκρισης

Τελεστής	**Λειτουργία**
==	Ισότητα
!=	Ανισότητα
>	Μεγαλύτερο
<	Μικρότερο
>=	Μεγαλύτερο ή ίσο
<=	Μικρότερο ή ίσο

Όταν συντάσουμε τους παραπάνω τελεστές σύγκρισης στο *Mathematica*, το *Mathematica* αναλαμβάνει να τους ξαναγράψει με την εξής μορφή: **ã**, **ƒ**, >, <, **¥**, **§**.

Λογικοί Τελεστές

Τελεστής	**Λειτουργία**
&&	Σύζευξη (και)
||	Διάζευξη (ή)

Στη συνέχεια θα δώσουμε μερικά παραδείγματα εφαρμογής της εντολής **If**.

Παράδειγμα 3: Να κατασκευάσετε τη συνάρτηση:

$$f(x)=\begin{cases} x & x \neq 0 \\ 1 & x = 0 \end{cases}$$

```
f@x_D := If@x ≠ 0,
   1 ê x,
   0,
"άγνωστο"D;
```

```
f@2D
```

$$\frac{1}{2}$$

```
f@0D
```

```
0
```

```
f@qD
```

```
άγνωστο
```

Παρατηρούμε ότι αν η τιμή του ορίσματος x είναι διαφορετική του μηδενός (π.χ. 2) εκτελείται η πρώτη εντολή, ενώ αν η τιμή του ορίσματος x είναι μηδέν τότε εκτελείται η δεύτερη εντολή. Αν όμως θέσουμε στο x μια τιμή (π.χ. q) που το *Mathematica* δεν γνωρίζει αν αυτή είναι διάφορη ή ίση του μηδενός τότε εκτελείται η τρίτη εντολή.

Παράδειγμα 4: Έστω η εξίσωση πρώτου βαθμού ax + b = 0. Να γράψετε πρόγραμμα που θα υπολογίζει και θα εκτυπώνει τις πιθανές λύσεις της εξίσωσης.

```
a = Input@"Δώσε το a="D;
b = Input@"Δώσε το b="D;
If@a ≠ 0,
 x = -b ê a;
 Print@"x=", xD,
 If@b « 0,
  Print@"Αόριστη"D,
  Print@"Αδύνατη"DDD
```

Παράδειγμα 5: Να γράψετε πρόγραμμα που να διαβάζει από το πληκτολόγιο μία λίστα από ακεραίους αριθμούς και να υπολογίζει και να εκτυπώνει τη διάμεσο των αριθμών αυτών.

Για να βρούμε τη διάμεσο πρέπει πρώτα να ταξινομήσουμε τα στοιχεία της λίστας, έστω a, σε αύξουσα σειρά και στη συνέχεια αν το πλήθος των στοιχείων της λίστας, έστω n, είναι άρτιος τότε η διάμεσος είναι ίση με την τιμή (a(n/2)+a(n/2+1))/2 διαφορετικά είναι ίση με a((n+1)/2).

```
a = Input@"Δώσε τη Λίστα="D;
n = Length@aD;
s = Sort@aD;
If@EvenQ@nD,
  d = Hs@@n ê 2DD + s@@n ê 2 + 1DDL ê 2,
  d = s@@Hn + 1L ê 2DDD;
Print@"Διάμεσος=", dD
```

Παράδειγμα 6: Να γράψετε πρόγραμμα το οποίο να διαβάζει από το πληκτρολόγιο δύο ακεραίους αριθμούς και αν το τελευταίο τους ψηφίο είναι το ίδιο να εκτυπώνει το άθροισμα τους ενώ στην αντίθετη περίπτωση να εκτυπώνει την απόλυτη τιμή της διαφοράς τους.

```
a = Input@"Δώσε έναν ακέραιο"D;
b = Input@"Δώσε έναν ακέραιο"D;
mona = Mod@a, 10D;
monb = Mod@b, 10D;
If@mona == monb,
 Print@a + bD,
 If@a > b,
  Print@a - bD,
  Print@b - aDDD
```

Παράδειγμα 7: Να κατασκευάσετε συνάρτηση, η οποία θα δέχεται ως ορίσματα δύο ακεραίους αριθμούς και αν το τελευταίο τους ψηφίο είναι το ίδιο να εκτυπώνει το άθροισμα τους ενώ στην αντίθετη περίπτωση να εκτυπώνει την απόλυτη τιμή της διαφοράς τους.

```
f@a_, b_D := Hmona = Mod@a, 10D;
  monb = Mod@b, 10D;
  If@mona == monb,
   Print@a + bD,
   If@a > b,
    Print@a - bD,
    Print@b - aDDDL
```

```
f@24, 244D
```

```
268
```

```
f@23, 54D
```

```
31
```

```
f@54, 23D
```

```
31
```

Παράδειγμα 8: Έστω η εξίσωση δευτέρου βαθμού $ax^2 + bx + c = 0$. Να γράψετε πρόγραμμα που θα υπολογίζει και θα εκτυπώνει τις πιθανές λύσεις της εξίσωσης.

```
a = Input@"Δώσε το a="D;
b = Input@"Δώσε το b="D;
c = Input@"Δώσε το c="D;
If@a ≠ 0,
 d = b^2 - 4 a c;
 If@d > 0,
  x1 = H-b + Sqrt@dDL ê H2 aL;
  x2 = H-b - Sqrt@dDL ê H2 aL;
  Print@"x1=", x1, "x2=", x2D,
  If@d « 0,
   x = -b ê H2 aL;
   Print@"x=", xD,
   Print@"Μιγαδικές Λύσεις"DDD,
 If@b ≠ 0,
  x = -c ê b;
  Print@"x=", xD,
  If@c « 0,
   Print@"Αόριστη"D,
   Print@"Αδύνατη"DDDD
```

Παράδειγμα 9: Να κατασκευάσετε συνάρτηση, η οποία θα δέχεται ως ορίσματα τους συντελεστές (a, b, c) μιας δευτεροβάθμιας εξίσωση και εκτυπώνει τις λύσεις της εξίσωσης.

```
f@a_, b_, c_D := HIf@a ≠ 0,
   d = b^2 - 4 a c;
   If@d > 0,
    x1 = H-b + Sqrt@dDL ê H2 aL;
    x2 = H-b - Sqrt@dDL ê H2 aL;
    Print@"x1=", x1, "x2=", x2D,
    If@d « 0,
     x = -b ê H2 aL;
     Print@"x=", xD,
     Print@"Μιγαδικές Λύσεις"DDD,
   If@b ≠ 0,
    x = -c ê b;
    Print@"x=", xD,
    If@c « 0,
     Print@"Αόριστη"D,
     Print@"Αδύνατη"DDDDL
```

```
f@1, 1, 1D
```

```
Μιγαδικές Λύσεις
```

```
f@1, 2, 1D
```

```
x=-1
```

```
f@1, 5, 1D
```

$$x1=\frac{1}{2}\left(-5+\sqrt{21}\right)\quad x2=\frac{1}{2}\left(-5-\sqrt{21}\right)$$

10.3.2 Η εντολή Which

Which[συνθήκη1, εντολή(-ές) 1, συνθήκη2, εντολή(-ές)2, συνθήκη3, εντολή(-ές)3, ...]

Η εντολή **Which** ελέγχει τις παραπάνω συνθήκες "συνθήκη1", "συνθήκη2", "συνθήκη3" κ.λ.π. και στην πρώτη αληθή συνθήκη που θα βρεί, έστω τη συνθήκηi, θα εκτελέσει την αντίστοιχη εντολή(-ές)i.

Στη συνέχεια θα δώσουμε μερικά παραδείγματα εφαρμογής της εντολής **Which**.

Παράδειγμα 10: Να κατασκευάσετε συνάρτηση, η οποία θα δέχεται ως ορίσματα τις συντεταγμένες ενός σημείου και θα επιστρέφει τις τιμές 1, 2, 3 και 4 αν το σημείο βρίσκεται στο 1ο, 2ο, 3ο και 4ο τεταρτημόριο αντίστοιχα. Αν το σημείο είναι το (0, 0) να επιστρέφει την τιμή 0. Τέλος, αν το σημείο ανήκει στον άξονα των x να επιστρέφει την τιμή -1 ενώ αν το σημείο ανήκει στον άξονα των y να επιστρέφει την τιμή -2.

```
f@x_, y_D := Which@x « 0 && y « 0, 0,
   y « 0, -1,
   x « 0, -2,
   x > 0 && y > 0, 1,
   x < 0 && y > 0, 2,
   x < 0 && y < 0, 3,
   x > 0 && y < 0, 4D;
```

```
f@0, 0D
```

```
0
```

```
f@3, 0D
```

```
-1
```

```
f@0, 4D
```

```
-2
```

```
f@4, 5D
```

```
1
```

```
f@-2, 5D
```

```
2
```

```
f@-3, -2D
```

```
3
```

```
f@3, -2D
```

```
4
```

Παράδειγμα 11: Να γράψετε πρόγραμμα το οποίο να διαβάζει τον βαθμό πτυχίου ενός φοιτητή και να εκτυπώνει τον λεκτικό χαρακτηρισμό του.

```
a = Input@"Δωσε τον βαθμό πτυχίου"D;
Which@a ≤ 10 && a > 8.5, Print@"Άριστα"D,
 a ≤ 8.5 && a > 6.5, Print@"Πολύ καλά"D,
 a ≤ 6.5 && a ≥ 5, Print@"Καλώς"D,
 a > 10 »» a < 5, Print@"Λάθος ο βαθμός πτυχίου"DD
```

Παράδειγμα 12: Να κατασκευάσετε συνάρτηση, η οποία θα δέχεται ως όρισμα τον βαθμό πτυχίου ενός φοιτητή και να εκτυπώνει τον λεκτικό χαρακτηρισμό του.

```
f@a_D := Which@a ≤ 10 && a > 8.5, Print@"Άριστα"D,
  a ≤ 8.5 && a > 6.5, Print@"Πολύ καλά"D,
  a ≤ 6.5 && a ≥ 5, Print@"Καλώς"D,
  a > 10 »» a < 5, Print@"Λάθος ο βαθμός πτυχίου"DD
```

```
f@8.51D
```

```
Άριστα
```

```
f@6.51D
```

```
Πολύ καλά
```

```
f@6.5D
```

```
Καλώς
```

```
f@12D
```

```
Λάθος ο βαθμός πτυχίου
```

Παράδειγμα 13: Έστω η εξίσωση δευτέρου βαθμού $ax^2 + bx + c = 0$. Να γράψετε πρόγραμμα που θα υπολογίζει και θα εκτυπώνει τις πιθανές λύσεις της εξίσωσης με χρήση της εντολής **Which.**

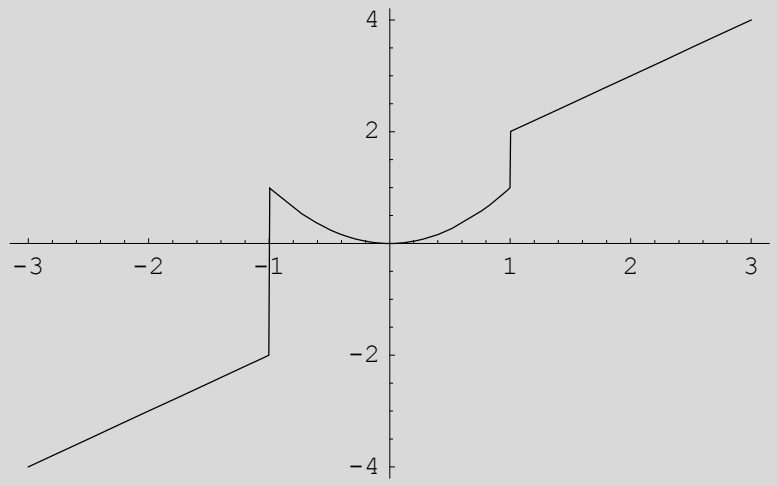
```
a = Input@"Δώσε το a="D;
b = Input@"Δώσε το b="D;
c = Input@"Δώσε το c="D;
If@a ≠ 0,
 d = b^2 - 4 a c;
 Which@d > 0,
  x1 = H-b + Sqrt@dDL ê H2 aL;
  x2 = H-b - Sqrt@dDL ê H2 aL;
  Print@"x1=", x1, "x2=", x2D,
  d « 0,
  x = -b ê H2 aL;
  Print@"x=", xD,
  d < 0,
  Print@"Μιγαδικές Λύσεις"DD,
 If@b ≠ 0,
  x = -c ê b;
  Print@"x=", xD,
  If@c « 0,
   Print@"Αόριστη"D,
   Print@"Αδύνατη"DDDD
```

Παράδειγμα 14: Να ορίσετε τη συνάρτηση

$$f(x) = \begin{cases} x - 1 & x \le 1 \\ x^2 - 1 & -1 < x < 1 \\ x + 1 & x \ge 1 \end{cases}$$

και να σχεδιάσετε τη γραφική της παράσταση για x∈[-3, 3].

```
f@x_D := Which@x ≤ -1, x - 1,
  x < 1, x^2,
  x ≥ 1, x + 1D
Plot@f@xD, 8x, -3, 3<D
```




```
- Graphics -
```

10.3.3 Η εντολή Switch

Κατά την εκτέλεση της εντολής **If** ο έλεγχος του προγράμματος μεταφέρεται σε ένα σύνολο εντολών ανάλογα με την τιμή μιας συνθήκης, η οποία μπορεί να πάρει δύο πιθανές τιμές: αληθής ή ψευδής.

Αν παρ' όλα αυτά θέλουμε να έχουμε παραπάνω από δύο επιλογές, τότε μπορούμε να χρησιμοποιήσουμε την εντολή Switch η οποία συντάσσεται ως εξής:

Switch[έκφραση, παράσταση1, εντολή(-ές) 1, παράσταση2, εντολή(-ές)2, παράσταση3, εντολή(-ές)3, ...]

Η εντολή **Switch** υπολογίζει αρχικά την έκφραση και στη συνέχεια ελέγχει με ποια από τις παραστάσεις "παράσταση1", "παράσταση2", παράσταση3" κ.λ.π. είναι ίση. Για την πρώτη ίση παράσταση που θα βρει, έστω τη παράστασηi θα εκτελέσει την αντίστοιχη εντολή(-ές).

Στη συνέχεια θα δώσουμε μερικά παραδείγματα εφαρμογής της εντολής **Switch**.

Παράδειγμα 15: Να κατασκευάσετε συνάρτηση, η οποία θα δέχεται ως όρισμα έναν ακέραιο αριθμό και θα ελέγχει αν ο αριθμός αυτός είναι άρτιος ή περιττός.

```
f@n_Integer?PositiveD := Switch@Mod@n, 2D,
  1, "Odd",
  0, "Even"D
```

```
f@2D
```

```
Even
```

```
f@1D

Odd
```

Η επικεφαλίδα Integer στο όρισμα της συνάρτησης αναγκάζει το n να είναι ακέραιος, ενώ ο έλεγχος ?Positive αναγκάζει το n να είναι και θετικός, αν θέλουμε η συνάρτηση f να δώσει απάντησει:

```
f@-10D

f@-10D
```

```
f@2.5D

f@2.5D
```

Παράδειγμα 16: Σε ένα παιχνίδι μεταξύ δύο ατόμων, η βαθμολογία έχει ως εξής:
α) Αν το άτομο Α παίξει την κίνηση 1 όταν το άτομο Β έπαιξε την κίνηση 1 τότε το άτομο Α κερδίζει 1 βαθμό.
β) Αν το άτομο Α παίξει την κίνηση 2 όταν το άτομο Β έπαιξε την κίνηση 1 τότε το άτομο Α χάνει 1 βαθμό.
γ) Αν το άτομο Α παίξει την κίνηση 1 όταν το άτομο Β έπαιξε την κίνηση 2 τότε το άτομο Α κερδίζει 2 βαθμούς.
δ) Αν το άτομο Α παίξει την κίνηση 2 όταν το άτομο Β έπαιξε την κίνηση 2 τότε το άτομο Α χάνει 2 βαθμούς.
Να κατασκευάσετε συνάρτηση που θα υπολογίζει το κέρδος του ατόμου Α ξέροντας τις κινήσεις των Α και Β.

```
f@x_, y_D := Switch@8x, y<,
  81, 1<, 1,
  82, 1<, -1,
  81, 2<, 2,
  82, 2<, -2D
```

Στην παραπάνω συνάρτηση αντιστοιχίσαμε τις κινήσεις x, y των παικτών Α και Β σε μία λίστα {x, y} και ανάλογα με τη τιμή της λίστα αυτής επιστρέφουμε και το αντίστοιχο κέρδος του παίκτη Α. Ας διαλέξουμε τώρα ένα τυχαίο παίξιμο για τους δύο παίκτες και στη συνέχεια ας υπολογίσουμε το κέρδος του παίκτη Α.

```
f@Random@Integer, 81, 2<D, Random@Integer, 81, 2<DD

2
```

10.4 Εντολές Επανάληψης

Σε πολλά προβλήματα απαιτείται η επανάληψη ενός συνόλου ενεργειών. Η επανάληψη μιας διαδικασίας παραπάνω από μία φορές ονομάζεται ανακύκλωση.

Στην ενότητα αυτή θα περιγράψουμε τις εντολές **Do**, **For** και **While** οι οποίες χρησιμοποιούνται σε προβλήματα που απαιτούνται ανακυκλώσεις. Συνήθως χρησιμοποιούμε τις εντολές **Do** και **For** όταν θέλουμε να επαναλάβουμε ένα σύνολο εντολών για συγκεκριμένο αριθμό επαναλήψεων ενώ την εντολή **While** την χρησιμοποιούμε όταν δεν είναι γνωστός εκ των προτέρων ο αριθμός των επαναλήψεων

10.4.1 Η εντολή Do

Do[εντολή, {imax}]	Εκτελεί την "εντολή" imax φορές.
Do[εντολή, {i, imax}]	Εκτελεί την "εντολή" καθώς η μεταβλητή i παίρνει τιμές από 1 έως imax με βήμα 1.
Do[εντολή, {i, imin, imax}]	Εκτελεί την "εντολή" καθώς η μεταβλητή i παίρνει τιμές από imin έως imax με βήμα 1.
Do[εντολή, {i, imin, imax, di}]	Εκτελεί την "εντολή" καθώς η μεταβλητή i παίρνει τιμές από imin έως imax με βήμα di.

Με τον όρο "εντολή" εννοούμε μία ή περισσότερες εντολές οι οποίες είναι χωρισμένες με ερωτηματικό.

Στην τελευταία από τις παραπάνω εκφράσεις της εντολής **Do** που είναι η πιο γενική μορφή διακρίνουμε τρεις περιπτώσεις, όσον αφορά το βήμα:

Περίπτωση 1η: di > 0.
Η μεταβλητή i παίρνει την i = imin.
Αν i **§** imax, τότε εκτελείται η "εντολή". Στη συνέχεια, προσθέτουμε το βήμα di στην τιμή της μεταβλητής i. Πηγαίνουμε πίσω και επαναλαμβάνουμε τον έλεγχο.
Αν i > imax, τότε μεταφερόμαστε εκτός της ανακύκλωσης.

Περίπτωση 2η: di < 0.
Η μεταβλητή i παίρνει την i = imin.
Αν i **¥** imax, τότε εκτελείται η "εντολή". Στη συνέχεια, προσθέτουμε το βήμα di στην τιμή της μεταβλητής i. Πηγαίνουμε πίσω και επαναλαμβάνουμε τον έλεγχο.
Αν i < imax, τότε μεταφερόμαστε εκτός της ανακύκλωσης.

Περίπτωση 3η: di = 0.
Δεν μπορεί το βήμα di να είναι μηδέν.

Ας δούμε στη συνέχεια μερικά παραδείγματα απλής εφαρμογής των παραπάνω εκφράσεων της εντολής **Do**:

```
Do@Print@"test"D, 85<D
```

```
test
test
test
test
test
```

```
Do@Print@iD, 8i, 1, 5<D
```

```
1
2
3
4
5
```

```
Do@a = i + 1; Print@aD, 8i, 5, 1, −1<D
```

```
6
5
4
3
2
```

Στη συνέχεια θα δώσουμε μερικά ακόμη παραδείγματα εφαρμογής της εντολής **Do**:

Παράδειγμα 17: Να δημιουργήσετε μία λίστα με 5 τυχαίους αριθμούς από το 1 έως το 20 και με 4 τυχαίους αριθμούς από το -20 έως το -1.

```
s = 8<;
Do@AppendTo@s, Random@Integer, 81, 20<DD, 85<D;
Do@AppendTo@s, Random@Integer, 8-20, -1<DD, 84<D;
s

810, 1, 3, 2, 14, -19, -1, -11, -13<
```

Παράδειγμα 18: Να κατασκευάσετε συνάρτηση, η οποία θα δημιουργεί μία λίστα με n τυχαίους αριθμούς από το 1 έως το 20 και με m τυχαίους αριθμούς από το -20 έως το -1.

```
tyxaioi@n_Integer?Positive, m_Integer?PositiveD := Hs = 8<;
  Do@AppendTo@s, Random@Integer, 81, 20<DD, 8n<D;
  Do@AppendTo@s, Random@Integer, 8-20, -1<DD, 8m<D;
  sL
```

```
tyxaioi@3, 4D

81, 19, 14, -18, -17, -6, -4<
```

Παράδειγμα 19: Να υπολογιστεί ο 10ος όρος της ακολουθίας

$$\alpha_n=\frac{1}{2}(\alpha_{n-1}+\frac{2}{\alpha_{n-1}}),\quad \alpha_1=1,$$

η οποία συγκλίνει στην τιμή $\sqrt{2}$.

```
aold = 1; DoAanew = 1/2 Haold + 2/aold L; aold = anew, 89<E; anew

4787633501779563550338751478164352626393810985192405254654229276259¥
  1925362787770306352384325384596398594331240032637710299217577668299¥
  6313024689222179880942725517434844559710363478381403509044255129799¥
 3385368114944226131160489088412764413184597197600430424080424489299¥
  1745564033552086544609184704239228339511303049317527075732707720¥
  4943610618917073241080260452775055121081948254768847591544963982¥
  848
```

Παρατηρούμε, ότι το αποτέλεσμα είναι ένας ρητός αριθμός. Πράγματι, αφού δώσαμε ως αρχική τιμή την ακέραια τιμή 1, το *Mathematica* κάνει πράξεις μεταξύ ακεραίων και προσπαθεί να δώσει αποτέλεσμα ακέραιο αριθμό. Επειδή όμως το πηλίκο δεν είναι ακέραιος αριθμός μας επιστρέφει ως αποτέλεσμα το ρητό αριθμό που αντιστοιχεί στο πηλίκο. Για να αποφύγουμε το παραπάνω αποτέλεσμα, μπορούμε είτε να χρησιμοποιήσουμε τη συνάρτηση **N** του *Mathematica* που επιστρέφει την αριθμητική τιμή της μεταβλητής anew είτε να δώσουμε ως αρχική τιμή τη πραγματική τιμή 1.0, οπότε το *Mathematica* κάνει πράξεις μεταξύ πραγματικών και δίνει αποτέλεσμα πραγματικό αριθμό.

```
aold = 1; DoAanew = 1/2 Jaold + 2/aold Ny; aold = anew, 89<E; N@anewD
```

```
1.41421
```

```
aold = 1.0; DoAanew = 1/2 Jaold + 2/aold Ny; aold = anew, 89<E; anew
```

```
1.41421
```

Παράδειγμα 20: Να κατασκευάσετε συνάρτηση, η οποία θα υπολογίζει τον n-στό όρο της ακολουθίας

$$\alpha_n=\frac{1}{2}(\alpha_{n-1}+\frac{2}{\alpha_{n-1}}),\quad \alpha_1=1,$$

η οποία συγκλίνει στην τιμή $\sqrt{2}$.

```
akolouthia@n_Integer?PositiveD :=
  Jaold = 1; DoAanew = 1/2 Jaold + 2/aold Ny; aold = anew, 8n - 1<E; N@anewDNy
```

```
akolouthia@10D
```

```
1.41421
```

Παράδειγμα 21: Να γράψετε πρόγραμμα που θα υπολογίζει τη σειρά: $S=1+2^2+3^3+4^4$.

```
s = 0;
Do@s = s + i^i, 8i, 1, 4<D;
s
```

```
288
```

Παράδειγμα 20: Να κατασκευάσετε συνάρτηση, η οποία θα υπολογίζει τη σειρα $S=1+2^2+3^3+4^4+...+n^n$ για δοθέντα ακέραιο n.

```
seira@n_Integer?PositiveD := Hs = 0;
  Do@s = s + i^i, 8i, 1, n<D;
  sL
```

```
seira@4D
```

```
288
```

Παράδειγμα 21: Να γράψετε πρόγραμμα που θα διαβάζει έναν μονοδιάστατο πίνακα ακεραίων αριθμών και θα υπολογίζει και θα εκτυπώνει το πλήθος των θετικών και των αρνητικών αριθμών του πίνακα.

```
a = Input@"Δώσε τον πίνακα"D;
negcounter = poscounter = 0;
Do@If@a@@iDD ≥ 0,
   poscounter = poscounter + 1,
   negcounter = negcounter + 1D,
  8i, 1, Length@aD<D;
Print@"poscounter=", poscounterD;
Print@"negcounter=", negcounterD;
```

10.4.1.1 Εμφωλευμένες επαναληπτικές διαδικασίες (ανακυκλώσεις)

Είναι δυνατό να τοποθετήσουμε μια **Do** μέσα σε μία άλλη, ή αλλιώς να έχουμε δύο μεταβλητές αντί για μία μέσα στην **Do** όπως φαίνεται παρακάτω:

Do[εντολή, {i, imin, imax, di}, {j, jmin, jmax, dj}] ή **Do**[**Do**[εντολή, {j, jmin, jmax, dj}], {i, imin, imax, di}]

Αρχικά θέτει i=imin και εκτελεί την "εντολή" καθώς η μεταβλητή j παίρνει τιμές από jmin έως jmax με βήμα dj. Εφόσον τελειώσουν οι πρώτες επαναλήψεις, προσθέτει στη μεταβλητή i το βήμα di και επαναλαμβάνει την εκτέλεση της "εντολής" καθώς η μεταβλητή j παίρνει τιμές από jmin έως jmax με βήμα dj. Η διαδικασία συνεχίζεται έως ότου η τιμή της μεταβλητής i ξεπεράσει το imax.

Με τα παρακάτω παραδείγματα, μπορούμε εύκολα να καταλάβουμε πως λειτουργεί η εντολή **Do** όταν έχουμε διπλή επανάληψη (ανακύκλωση):

```
Do@Print@8i, j<D, 8i, 1, 3<, 8j, 1, 2<D
```

```
81, 1<
```

```
81, 2<
```

```
82, 1<
```

```
82, 2<
```

```
83, 1<
```

```
83, 2<
```

```
Do@
 Do@Print@8i, j<D, 8j, 1, 2<D,
 8i, 1, 3<D
```

```
81, 1<
```

```
81, 2<
```

```
82, 1<
```

```
82, 2<
```

```
83, 1<
```

```
83, 2<
```

Στα παραπάνω δύο παραδείγματα, για i=1 εκτελείται η εντολή Print[{i,j}] για j=1, 2, 3. Στη συνέχεια αυξάνεται το i κατά ένα, i=2 και εκτελείται η "εντολή" Print[{i,j}] για j=1, 2, 3.

Παράδειγμα 22: Να γράψετε πρόγραμμα που θα διαβάζει τα στοιχεία ενός δισδιάστατου πίνακα ακεραίων αριθμών και:
i) θα διαβάζει έναν αριθμό που θα αντιστοιχεί σε μια στήλη και θα υπολογίζει το ελάχιστο της στήλης αυτής και
ii) θα διαβάζει έναν αριθμό που θα αντιστοιχεί σε μια γραμμή και θα υπολογίζει το μέγιστο της γραμμής αυτής.

```
a = Input@"Δώσε πίνακα="D;
r = Input@"Δώσε τη γραμμή="D;
c = Input@"Δώσε τη στήλη="D;
max = a@@r, 1DD;
Do@If@max < a@@r, jDD,
   max = a@@r, jDDD,
  8j, 1, Length@a@@rDDD<D;
Print@"Το max της ", r, " γραμμής είναι:", maxD;
min = a@@1, cDD;
Do@If@min > a@@i, cDD,
   min = a@@i, cDDD,
  8i, 1, Length@aD<D;
Print@"Το min της ", c, " στήλης είναι:", minD;
```

Παράδειγμα 23: Να γράψετε πρόγραμμα που θα διαβάζει τα στοιχεία ενός δισδιάστατου πίνακα ακεραίων αριθμών και θα εξετάζει αν ο πίνακας είναι αραιός. Θεωρούμε ότι ένας πίνακας είναι αραιός αν πάνω από το 80% του πλήθους των στοιχείων του είναι μηδέν.

```
a = Input@"Δώσε πίνακα="D;
counter = 0;
Do@
  Do@If@a@@i, jDD « 0,
    counter = counter + 1D,
   8j, 1, Length@a@@1DDD<D,
  8i, 1, Length@aD<D;
If@counter > 0.8  Length@aD  Length@a@@1DDD,
  Print@"Αραιός"D,
  Print@"Όχι αραιός"DD;
```

10.4.2 Η εντολή For

Η σύνταξη της εντολής **For** μοιάζει πολύ με τη σύνταξη της εντολής **For** που συναντάμε στην γλώσσα προγραμματισμού C με τη μόνη διαφορά ότι αντί για το σύμβολο ";" χρησιμοποιούμε το σύμβολο ",".

For[έναρξη, έλεγχος, αύξηση, εντολή]

Το πρώτο όρισμα "έναρξη" είναι αυτό που θα εκτελεστεί πρίν αρχίσει η επάναληψη. Το δεύτερο όρισμα "έλεγχος" περιέχει έναν έλεγχο ο οποίος θα γίνεται πριν την κάθε επανάληψη. Αν ο έλεγχος επιστρέψει True τότε θα ακολουθήσει η επανάληψη διαφορετικά η επανάληψη θα σταματήσει. Το τρίτο όρισμα "αύξηση" εκτελείται μετά την εντολή και έχει σκοπό την ενημέρωση κάποιας μεταβλητής που περιέχεται στο όρισμα "έλεγχος". Συνήθως πρόκειται για προσαύξηση του μετρητή επανάληψης κατά ένα βήμα. Το τέταρτο όρισμα "εντολή" αποτελείται από μία ή περισσότερες εντολές οι οποίες θα εκτελεστούν κατά την επανάληψη. Οι εντολές αυτές θα πρέπει να είναι χωρισμένες με ερωτηματικό.

```
For@i = 1, i ≤ 5, i++, Print@iDD
```

```
1
```

```
2
```

```
3
```

```
4
```

```
5
```

Παράδειγμα 24: Να δημιουργήσετε μία λίστα με 5 τυχαίους αριθμούς από το 1 έως το 20 και με 4 τυχαίους αριθμούς από το -20 έως το -1. (Να κάνετε χρήση της εντολής For).

```
s = 8<;
For@i = 1, i ≤ 5, i++, AppendTo@s, Random@Integer, 81, 20<DDD;
For@i = 1, i ≤ 4, i++, AppendTo@s, Random@Integer, 8−20, −1<DDD;
s

811, 16, 3, 7, 10, −5, −1, −5, −8<
```

Παράδειγμα 25: Μια πολύ γνωστή ακολουθία φυσικών αριθμών στα μαθηματικά είναι η ακολουθία Fibonacci (ορισμένη από τον Ιταλό μαθηματικό Leonardo Fibonacci). Οι όροι της ακολουθίας αυτής, ας τη συμβολήσουμε f, ορίζονται ως εξής:

$$f_1 = 1,\ f_2 = 1,\ f_3 = 2,\ f_4 = 3,\ f_5 = 5 \ldots$$

και γενικώς $f_1 = f_{n-1} + f_{n-2}$ (δηλαδή κάθε όρος εκτός από τους δύο πρώτους, δίνεται ως άθροισμα των δύο προηγούμενων όρων). Γράψτε πρόγραμμα που θα ζητά ένα φυσικό αριθμό n και στη συνέχεια θα υπολογίζει και θα εκτυπώνει τον n-στό όρο της ακολουθίας Fibonacci. (Να κάνετε χρήση της εντολής For).

```
n = Input@"Δώσε έναν φυσικό αριθμό"D;
f1 = 1;
f2 = 1;
For@i = 3, i ≤ n, i++,
  f = f2 + f1;
  f1 = f2;
  f2 = fD;
Print@"O ", n, "−στός όρος της ακολουθίας Fibonnaci είναι ο:"D
Print@fD
```

```
O 150−στός όρος της ακολουθίας Fibonnaci είναι ο:
```

```
9969216677189303386214405760200
```

Παράδειγμα 26: Να κατασκευάσετε συνάρτηση, η οποία θα υπολογίζει τον n-οστό όρο της ακολουθίας fibonacci. (Να κάνετε χρήση της εντολής For).

```
fibo@n_Integer?PositiveD := Hf1 = 1;
  f2 = 1;
   For@i = 3, i ≤ n, i++,
    f = f2 + f1;
    f1 = f2;
    f2 = fD;
  fL
```

```
fibo@10D

55
```

Όπως και στην εντολή **Do** έτσι και στην εντολή **For** (όπως και σε κάθε εντολή επανάληψης) μπορούμε να τοποθετήσουμε μια **For** μέσα σε μία άλλη:

```
For@i = 1,
 i ≤ 2,
 i++,
 For@j = 1,
  j ≤ 3,
  j++,
  Print@8i, j<DDD
```

```
81, 1<
```

```
81, 2<
```

```
81, 3<
```

```
82, 1<
```

```
82, 2<
```

```
82, 3<
```

Παράδειγμα 27: Να γράψετε πρόγραμμα που θα διαβάζει τα στοιχεία ενός δισδιάστατου πίνακα ακεραίων αριθμών και θα εξετάζει αν ο πίνακας είναι αραιός. Θεωρούμε ότι ένας πίνακας είναι αραιός αν πάνω από το 80% του πλήθους των στοιχείων του είναι μηδέν. (Να κάνετε χρήση της εντολής For).

```
a = Input@"Δώσε πίνακα="D;
counter = 0;
For@i = 1, i ≤ Length@aD, i++,
  For@j = 1, j ≤ Length@a@@1DDD, j++, If@a@@i, jDD « 0,
    counter = counter + 1DDD;
If@counter > 0.8  Length@aD  Length@a@@1DDD,
  Print@"Αραιός"D,
  Print@"Οχι αραιός"DD;
```

Παράδειγμα 28: Να υπολογίσετε το άθροισμα των αρτίων αριθμών από το 1 έως το 1000. (Να κάνετε χρήση της εντολής For).

```
s = 0;
For@i = 2, i ≤ 1000, i = i + 2, s = s + iD;
s

250500
```

Παράδειγμα 29: Να υπολογίσετε το άθροισμα των περιττών αριθμών από το 1 έως το 1000. (Να κάνετε χρήση της εντολής For).

```
s = 0;
For@i = 1, i ≤ 1000, i = i + 2, s = s + iD;
s

250000
```

10.4.3 Η εντολή While

Η εντολή **While** χρησιμοποιείται συνήθως όταν θέλουμε να επαναλάβουμε ένα σύνολο εντολών για έναν αριθμό φορών που δεν γνωρίζουμε εκτ ων προτέρων.

While[έλεγχος, εντολή]

Πρώτα γίνεται η εκτέλεση του "ελέγχου". Αν επιστραφεί False η επαναληπτική διαδικασία σταματάει. Αν επιστραφεί True τότε εκτελείται η "εντολή" και επαναλαμβάνεται ο "έλεγχος". Το δεύτερο όρισμα "εντολή" αποτελείται από μία ή περισσότερες εντολές οι οποίες θα εκτελεστούν κατά την επανάληψη. Οι εντολές αυτές θα πρέπει να είναι χωρισμένες με ερωτηματικό.

Θα πρέπει να δωθεί μεγάλη προσοχή ώστε οι μεταβλητές που περιέχονται στον "έλεγχο" να έχουν πάρει αρχική τιμή πριν τον "έλεγχο" ώστε να μπορέσει να εκτελεστεί η "εντολή", αλλά και να αλλάζουν τιμή μέσα στην "εντολή" ώστε η επαναληπτική διαδικασία να σταματήσει σε κάποιο πεπερασμένο βήμα.

Παράδειγμα 30: Να υπολογίσετε το άθροισμα των αρτίων αριθμών από το 1 έως το 1000. (Να κάνετε χρήση της εντολής While).

```
s = 0;
i = 2;
While@i ≤ 1000, s = s + i; i = i + 2D
s

250500
```

Παράδειγμ 31: Να υπολογίσετε τη ρίζα της συνάρτησης $f[x] = \cos[x] - x^2$ με ακρίβεια $e=10^{-6}$κάνοντας χρήση της επαναληπτικής μεθόδου Newton-Rapshon:

$$\alpha_{n+1}=\alpha_n-\frac{f[\alpha_n]}{f'[\alpha_n]},\quad \alpha_0 = c \in R.$$

```
f@x_D = Cos@xD - x^2;
x0 = Input@"Δώσε αρχική τιμή="D;
If@f'@x0D « 0,
  Print@"Error"D,
  xold = N@x0D; xnew = N@xold - f@xoldD ê f'@xoldDD;
  While@Abs@xnew - xoldD > 10^-6,
   xold = xnew; xnew = N@xold - f@xoldD ê f'@xoldDDDD;
xnew

0.824132
```

Παράδειγμ 32: Να γράψετε πρόγραμμα το οποίο θα υπολογίζει τη σειρά S=$\frac{1}{2}+\frac{1}{4}+\frac{1}{6}+\frac{1}{8}$+... με τελευταίο όρο αυτόν που δεν ξεπερνάει την τιμή 0.000001.

```
s = 0;
i = 1;
While@H1. ê H2  iL > 0.00001L, s = s + 1. ê H2  iL; i = i + 1D;
s

5.69849
```

Παράδειγμ 33: Να γράψετε πρόγραμμα το οποίο θα υπολογίζει τον μέγιστο κοινό διαιρέτη δύο αριθμών σύμφωνα με τον αλγόριθμό του Ευκλείδη.

```
a = Input@"Δώσε έναν ακέραιο"D;
b = Input@"Δώσε έναν ακέραιο"D;
If@a ≥ b, diaireteos = a;
  diairetis = b, diaireteos = b; diairetis = aD;
r = Mod@diaireteos, diairetisD;
While@r ≠ 0, diaireteos = diairetis;
  diairetis = r; r = Mod@diaireteos, diairetisDD;
diairetis
```

Παράδειγμ 34: Να κατασκευάσετε συνάρτηση η οποία θα υπολογίζει τον μέγιστο κοινό διαιρέτη δύο αριθμών σύμφωνα με τον αλγόριθμό του Ευκλείδη.

```
mkd@a_Integer, b_IntegerD :=
 HIf@a ≥ b, diaireteos = a;
   diairetis = b, diaireteos = b; diairetis = aD;
  r = Mod@diaireteos, diairetisD;
  While@r ≠ 0, diaireteos = diairetis;
   diairetis = r; r = Mod@diaireteos, diairetisDD;
  diairetisL
```

```
mkd@320, 180D

20
```

Για επαλήθευση μπορούμε να χρησιμοποιήσουμε και τη συνάρτηση GCD που διαθέτει το *Mathematica*.

```
GCD@320, 180D

20
```