Κεφάλαιο 1

Εισαγωγή σε Mathematica

Στο κεφάλαιο αυτό θα δώσουμε μια συνοπτική αναφορά στις δυνατότητες του Mathematica στους αριθμητικούς υπολογισμούς, στους συμβολικούς υπολογισμούς, στις γραφικές παραστάσεις και στα είδη προγραμματισμού που υποστηρίζει. Στα επιμέρους κεφάλαια της «Γραμμικής Άλγεβρας» και του «Λογισμού μιας Μεταβλητής» θα μελετήσουμε επιμέρους δυνατότητες του πακέτου.

1.1 Πότε δημιουργήθηκε το Mathematica ;

Ο Stephen Wolfram είναι ο επιστήμονας ο δημιουργός του Mathematica. Ο Wolfram γεννήθηκε το 1959 στο Λονδίνο, και πήρε το διδακτορικό του στην Θεωρητική Φυσική από το πανεπιστήμιο του Caltech σε ηλικία 20 χρονών. Ο Wolfram γρήγορα έγινε ένας από τους πρωταγωνιστές στον νέο ραγδαία αναπτυσσόμενο κλάδο των επιστημονικών υπολογισμών. Το 1979 δημιούργησε το SMP το πρώτο μοντέρνο υπολογιστικό σύστημα άλγεβρας το οποίο κυκλοφόρησε εμπορικά το 1981. Το κύριο ερευνητικό του ενδιαφέρον αφορούσε τις αρχές που διέπουν την πολυπλοκότητα που συναντάμε στην φύση. Το 1986 και έπειτα από μια επιτυχή ακαδημαϊκή καριέρα ο Wolfram, ίδρυσε την εταιρεία Wolfram η οποία διέθεσε εμπορικά την πρώτη έκδοση του Mathematica στις 23 Ιουνίου του 1988, η οποία σημείωσε σημαντική επιτυχία και καθιέρωσε την εταιρεία Wolfram ως μια από τις πρώτες εταιρείες σε παγκόσμια κατάταξη στην παραγωγή software. Το 1991, κυκλοφόρησε η $2^{η}$ έκδοση του Mathematica, ενώ ακολούθησαν οι εκδόσεις 3, 4 και 5 τις χρονιές 1996, 1999 και 2003 αντίστοιχα. Σήμερα υπάρχουν περίπου 2.000.000 χρήστες του Mathematica παγκοσμίως, οι οποίοι ανήκουν σε κλάδους όπου τα μαθηματικά είναι ένα απαραίτητο εργαλείο όπως μαθηματικοί, μηχανολόγοι, οικονομολόγοι και άλλες επιστήμες.

1.2 Αριθμητικοί υπολογισμοί

1.2.1. Τελεστές

Οι τελεστές είναι σύμβολα που δηλώνουν πράξεις μεταξύ τελεστών, δηλαδή αριθμών, αλυσίδων χαρακτήρων κ.λ.π.. Υπάρχουν 4 κατηγορίες τελεστών : αριθμητικοί, χαρακτήρων, σύγκρισης και λογικοί.

1.2.1.1 Τελεστές αριθμητικοί

Χρησιμοποιούνται για πράξεις μεταξύ αριθμών.

Τελεστής	Συνάρτηση	Λειτουργία	Σύνταξη
+	Plus[]	Πρόσθεση	x+y Plus[x,y]
–	Minus[]	Αφαίρεση	x-y Minus[x,y]
*	Times[]	Πολλαπλασιασμός	x*y Times[x,y]
/		Διαίρεση	x/y Times[x,Power[y,-1]
^	Power[]	Δύναμη (Ctrl+6)	x^y Power[x,y]

Συνεπώς παρατηρούμε από τον παραπάνω πίνακα αλλά και τους πίνακες των λογικών τελεστών και των τελεστών σύγκρισης που θα πούμε παρακάτω ότι κάθε έκφραση στο Mathematica μπορεί να γραφεί με την μορφή συνάρτησης. Την μορφή αυτή της συνάρτησης μπορούμε να την αναπαραστήσουμε με την εντολή FullForm ενώ την

κεφαλή της έκφρασης μπορείς να την δεις με την συνάρτηση Head. Η εκτέλεση κάθε εντολής γίνεται πατώντας Shift+Enter, ενώ το Enter το χρησιμοποιούμε αν θέλουμε να γράψουμε παραπάνω από μια εντολές στο ίδιο κελί για να τις εκτελέσουμε όλες μαζί στη συνέχεια με το Shift+Enter.

FullForm[έκφραση]	Αναπαριστά την έκφραση με μορφή συνάρτησης
Head[έκφραση]	Επιστρέφει το πρώτο όνομα της συνάρτησης που εμφανίζεται με την FullForm ή διαφορετικά το f στην συνάρτηση f[x,y,..].

Παράδειγμα 1.2.1.1.1

```
In[1]:= 3+4

Out[1]= 7

In[2]:= 78*84

Out[2]= 6552

In[3]:= 2^24

Out[3]= 16777216
```

Παρατήρηση. Το σύμβολο του πολλαπλασιασμού (*) μπορεί να αντικατασταθεί και με τον κενό χαρακτήρα.

Δραστηριότητα 1.2.1.1.2 Προσπάθησε να υπολογίσεις τις παρακάτω εκφράσεις :

$$31 \times \frac{4}{23} - 52$$

$$(32 + 24 - 23) + \frac{22}{2 - 34}$$

1.2.1.2 Τελεστές Σύγκρισης

Χρησιμοποιούνται για σύγκριση μεταξύ αριθμών ή αλυσίδων χαρακτήρων.

Τελεστής	Λειτουργία	Σύνταξη
== ή Equal[]	Ισότητα	x==y ή Equal[x,y]
!= ή UnEqual[]	Ανισότητα	x!=y ή UnEqual[x,y]
> ή Greater[]	Μεγαλύτερο	x>y ή Greater[x,y]
< ή Less[]	Μικρότερο	x<y ή Less[x,y]
>= ή GreaterEqual[]	Μεγαλύτερο ή ίσο	x>=y ή GreaterEqual[x,y]
<= ή LessEqual[]	Μικρότερο ή ίσο	x<=y ή LessEqual[x,y]

Παράδειγμα 1.2.1.2.1

```
In[10]:= 5==5

Out[10]= True
```

```
In[11]:= 5 ≠ 6

Out[11]= True
```

□

1.2.1.3 Τελεστές Λογικοί

Χρησιμοποιούνται για την εκτέλεση λογικών πράξεων.

Τελεστής	Λειτουργία
! ή Not[x]	Λογική άρνηση
&& ή And[x,y]	Λογική πρόσθεση
|| ή Or[x,y]	Διάζευξη
&&! ή Xor[x,y]	ή/και

Οι πίνακες αληθείας των τελεστών είναι :

X	Y	X&&Y	X || Y	!X	X&& !Y
T	T	T	T	F	F
T	F	F	T	F	T
F	T	F	T	T	T
F	F	F	F	T	F

Παράδειγμα 1.2.1.3.1

```
In[12]:= Not[3 >= 3]

Out[12]= False

In[13]:= (4 == 3) && (3 > 2)

Out[13]= False

In[14]:= True || True

Out[14]= True

In[15]:= (3 > 4) && ! (3 < 5)

Out[15]= False
```

Δραστηριότητα 1.2.1.3.2
Προσπάθησε να υπολογίσεις το αποτέλεσμα των λογικών εκφράσεων :
α) (1<2) και (4<2)
β) (2>5) ή (3>6)

1.2.2 Εκφράσεις στο Mathematica

Ανεξάρτητες μονάδες, όπως μεταβλητές, σταθερές, συναρτήσεις και τελεστές συνδυάζονται για να κατασκευάσουν εκφράσεις. Μια έκφραση είναι ένας τύπος υπολογισμού μιας τιμής. Οι τελεστές προσδιορίζουν την επεξεργασία που θα υποστούν οι τελεσταίοι.

Παράδειγμα 1.2.2.1

$$\underset{\text{τελεστέος}}{5} \quad \underset{\text{τελεστής}}{+} \quad \underset{\text{τελεστέος}}{x}$$

Ο υπολογισμός της τιμής μιας έκφρασης ακολουθεί την παρακάτω σειρά προτεραιότητας :

Πίνακας 1.2.2.1 Σειρά προτεραιότητας σε πράξεις τελεστών.

Τύπος	Τελεστής	Σειρά προτεραιότητας για τελεστές με την ίδια προτεραιότητα
Αριθμητικός	^	Δεξιά προς αριστερά
	* /	Αριστερά προς τα δεξιά
	+ –	Αριστερά προς τα δεξιά
Σύγκρισης	< <=	
	> >=	Αριστερά προς τα δεξιά
	== !=	
Λογικοί	!	Δεξιά προς αριστερά
	&&	Αριστερά προς τα δεξιά
	||	Αριστερά προς τα δεξιά

Σε μια έκφραση προηγούνται στην εκτέλεση οι πράξεις που είναι μέσα στις παρενθέσεις. Οι εκφράσεις διακρίνονται σε πραγματικές, ακέραιες κ.λ.π. αν το αποτέλεσμα της τιμής της έκφρασης είναι πραγματικός αριθμός, ακέραιος αριθμός κ.λ.π.

Παράδειγμα 1.2.2.2

Να υπολογιστούν τα αποτελέσματα των παρακάτω εκφράσεων :

3^2^3
2*(5+1)*3/4^2
(4>3)||(3<1)&&(!(3>1))

Απάντηση

1. Η σειρά προτεραιότητας στις δυνάμεις είναι από δεξιά προς τα αριστερά και επομένως θα έχουμε :

3^2^3=3^(2^3)=3^8=6561

2. $$2*\underbrace{(5+1)}_{6}*3/4\text{^}2 = 2*6*3/\underbrace{4\text{^}2}_{16} = \underbrace{2*6}_{12}*3/16 =$$
$$= \underbrace{12*3}_{36}/16 = 36/16$$

Επειδή έχουμε πράξεις μεταξύ ακεραίων το αποτέλεσμα θα είναι ένας ρητός αριθμός.

3. $$\underbrace{(4>3)}_{\text{True}} || \underbrace{(3<1)}_{\text{False}} \&\& \left(!\underbrace{(3>1)}_{\text{True}} \right) = True \,||\, False \,\&\&\, \underbrace{(!True)}_{False} =$$
$$= True \,||\, \underbrace{False \,\&\&\, False}_{False} = \underbrace{True \,||\, False}_{True} = True$$

Δραστηριότητα 1.2.2.3

Να υπολογίσεις το αποτέλεσμα των παρακάτω εκφράσεων :
α) (2>5) και (3>6) ή (4>3)

β) (5+4/2)^2^3

1.2.3 Ποιοι είναι οι τύποι αριθμών που υποστηρίζει το Mathematica ;

Το Mathematica υποστηρίζει : α) ακέραιους αριθμούς, β) ρητούς αριθμούς, γ) πραγματικούς αριθμούς και δ) μιγαδικούς αριθμούς. Στα παρακάτω παραδείγματα φαίνονται οι τρόποι με τον οποίο χειρίζεται το Mathematica τους διάφορους τύπους αριθμών.

Παράδειγμα 1.2.3.1

Παρακάτω βλέπουμε ότι ο αριθμός ¾ θεωρείται από το Mathematica ως Rational με δύο ορίσματα : τον αριθμητή 3 και τον παρονομαστή 4, ενώ σε αντίθεση με άλλες γλώσσες προγραμματισμού το Mathematica αναπαριστά τον αριθμό αυτό αλλά και όλους τους ρητούς αριθμούς με την πλήρη μορφή τους.

```
In[50]:= 3/4
Out[50]= 3/4
In[51]:= FullForm[%]
Out[51]//FullForm=
Rational[3, 4]
In[52]:= Precision[%%]
Out[52]= ∞
```

Μπορούμε να ελέγξουμε αν ένας αριθμός είναι ακέραιος ή αν μια έκφραση είναι αριθμός κάνοντας χρήση των συναρτήσεων IntegerQ και NumberQ αντίστοιχα. Παρόμοιες συναρτήσεις μπορεί κάποιος να βρει με χρήση της βοήθειας (?*Q).

```
In[53]:= IntegerQ[3/4]
Out[53]= False
In[54]:= NumberQ[3/4]
Out[54]= True
In[55]:= NumberQ[Pi]
Out[55]= False
```

Παράδειγμα 1.2.3.2

Έστω ο μιγαδικός αριθμός

```
In[56]:= z = -√3 + I
Out[56]= i - √3
In[57]:= FullForm[z]
Out[57]//FullForm=
Plus[Complex[0, 1], Times[-1, Power[3, Rational[1, 2]]]]
```

Το πραγματικό του μέρος είναι

```
In[58]:= Re[z]
Out[58]= -√3
```

ενώ το φανταστικό του μέρος είναι

In[59]:= **Im[z]**

Out[59]= 1

Ο συζυγής μιγαδικός αριθμός είναι

In[60]:= **Conjugate[z]**

Out[60]= $-i-\sqrt{3}$

Το μέτρο του μιγαδικού αριθμού z είναι

In[61]:= **Abs[z]**

Out[61]= 2

και το όρισμα του είναι

In[62]:= **Arg[z]**

Out[62]= $\frac{5\pi}{6}$

και συνεπώς μπορούμε εύκολα να γράψουμε την τριγωνομετρική μορφή του αριθμού αυτού ως

In[63]:= **Simplify[2* (Cos[$\frac{5\,Pi}{6}$] + I Sin[$\frac{5\,Pi}{6}$])]**

Out[63]= $i-\sqrt{3}$

Η συνάρτηση *Simplify*[έκφραση] κάνει όλες τις δυνατές απλοποιήσεις που μπορούν να γίνουν στην «έκφραση». Μιγαδικοί αριθμοί μπορούμε να έχουμε ως το αποτέλεσμα της επίλυσης εξισώσεων :

In[64]:= **Clear[z]** (καθαρίζει το περιεχόμενο της μεταβλητής z)

In[65]:= **Solve[z^4 == 1, z]**

Out[65]= {{z → -1}, {z → -i}, {z → i}, {z → 1}}

Παράδειγμα 1.2.3.3 Ας υποθέσουμε ότι μας ζητούν να υπολογίσουμε τις ρίζες της εξίσωσης $z^6+1=0$ και να τις αναπαραστήσουμε στο μιγαδικό επίπεδο.
Οι λύσεις της παραπάνω εξίσωσης μπορούν να υπολογιστούν είτε από την συνάρτηση *Solve*[] :

In[66]:= **Solve[z^6 + 1 == 0, z] // N**

Out[66]= {{z → 0. - 1. i}, {z → 0. + 1. i},
{z → -0.866025 - 0.5 i}, {z → 0.866025 + 0.5 i},
{z → 0.866025 - 0.5 i}, {z → -0.866025 + 0.5 i}}

είτε υπολογίζοντας πρώτα το μέτρο και το όρισμα του μιγαδικού αριθμού :

In[67]:= **Abs[-1]**

Out[67]= 1

In[68]:= **Arg[-1]**

Out[68]= π

και εφόσον φέρουμε τον αριθμό στην τριγωνομετρική του μορφή :

In[69]:= **Simplify[1* (Cos[Pi] + I Sin[Pi])]**

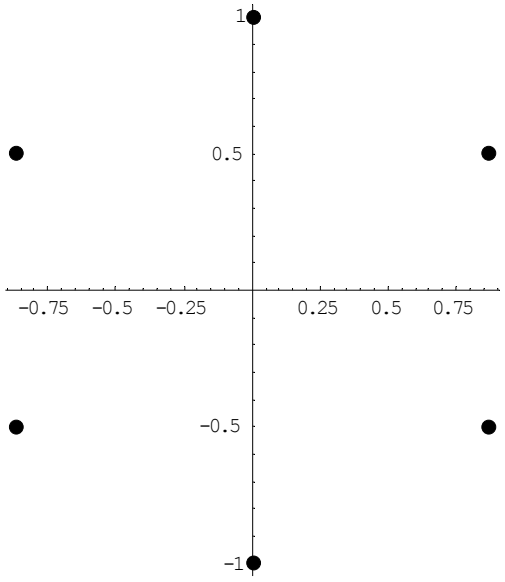
Out[69]= -1

να κάνουμε χρήση του τύπου του De'Moivre για τον υπολογισμό των ριζών

```
In[70]:= Q1 = Table[N[Sqrt[1, 6] (Cos[(2 n Pi + Pi) / 6] + Sin[(2 n Pi + Pi) / 6] I)], {n, 0, 5}]

Out[70]= {0.866025 + 0.5 i, 0. + 1. i, -0.866025 + 0.5 i,
          -0.866025 - 0.5 i, 0. - 1. i, 0.866025 - 0.5 i}
```

Μπορούμε μάλιστα να αναπαραστήσουμε τους αριθμούς αυτούς στο μιγαδικό επίπεδο

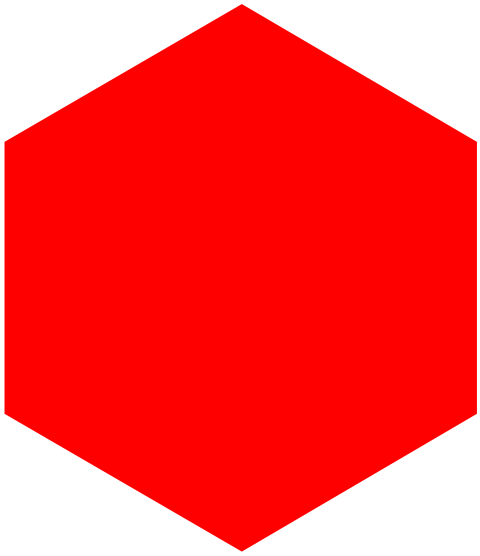
```
In[71]:= Q2 = Table[N[{Sqrt[1, 6] Cos[(2 n Pi + Pi) / 6], Sqrt[1, 6] Sin[(2 n Pi + Pi) / 6]}], {n, 0, 5}]

Out[71]= {{0.866025, 0.5}, {0., 1.}, {-0.866025, 0.5},
          {-0.866025, -0.5}, {0., -1.}, {0.866025, -0.5}}

In[72]:= ListPlot[Q2, PlotStyle → {PointSize[0.03]}, AspectRatio → Automatic]
```




```
Out[72]= - Graphics -
```

ενώ αν ενώσουμε τα παραπάνω σημεία θα δούμε ότι σχηματίζετε ένα κανονικό εξάγωνο

```
In[73]:= Show[Graphics[{RGBColor[1, 0, 0], Polygon[Q2]}], AspectRatio → Automatic]
```



```
Out[73]= - Graphics -
```

1.2.4 Προσεγγιστικοί υπολογισμοί

Οι πράξεις μεταξύ ακεραίων μας οδηγεί σε ακέραιο ή ρητό αποτέλεσμα. Αν θέλουμε να υπολογίσουμε όμως το αποτέλεσμα ως πραγματικό αριθμό θα πρέπει να κάνουμε χρήση της συνάρτησης N (με δύο τρόπους).

N[έκφραση]	Επιστρέφει την αριθμητική τιμή της έκφρασης.
N[έκφραση,p]	Επιστρέφει την αριθμητική τιμή της έκφρασης με προσέγγιση p σημαντικών ψηφίων.

$$\underbrace{d_1...d_r.d_{r+1}...d_p}_{p} \quad 0.0...\underbrace{d_1...d_p}_{p} \quad \underbrace{d_1...d_p}_{p}0...0.0$$

Σχήμα 1.2.4.1 Αναπαράσταση των σημαντικών ψηφίων p, σε αριθμούς κινητής υποδιαστολής ($d_p \neq 0$).

Οι αριθμοί κινητής υποδιαστολής στο Mathematica μπορούν να χωριστούν σε δύο κατηγορίες : α) τους αριθμούς ακρίβειας μηχανής (machine precision numbers), και β) τους αριθμούς με μεταβλητή ακρίβεια (arbitrary-precision numbers). Κάνοντας χρήση της δεύτερης κατηγορίας αριθμών μπορούμε να αυξήσουμε την ακρίβεια των αριθμών που αναπαριστούμε στον Η/Υ αλλά και έχουμε την δυνατότητα να εκτελέσουμε πράξεις υψηλής ακρίβειας.

Παράδειγμα 1.2.4.1

```
In[4]:= N[1/3]

Out[4]= 0.333333
```

ή

```
In[5]:= 1/3 // N

Out[5]= 0.333333

In[6]:= N[Pi, 100]

Out[6]= 3.14159265358979323846264338327950288419716939937510582
        97494459230781640628620899862803482534211706
```

Το Pi είναι η μαθηματική σταθερά π. Μπορούμε να υπολογίσουμε την ακρίβεια που χρησιμοποιεί το *Mathematica* για υπολογισμούς που κάνει μέσω της συνάρτησης *Precision*.

```
In[7]:= N[17^(1/2)]

Out[7]= 4.12311

In[8]:= Precision[%]

Out[8]= 16
```

□

Ένας πραγματικός αριθμός μπορεί να μετασχηματισθεί στον πλησιέστερο ρητό αριθμό μέσω της συνάρτησης *Rationalize*.

Rationalize[x]	Παίρνει τον πραγματικό αριθμό x και τον μετατρέπει στον πλησιέστερο ρητό αριθμό.
Rationalize[x,dx]	Παίρνει τον πραγματικό αριθμό x και τον μετατρέπει στον πλησιέστερο ρητό αριθμό q τέτοιο ώστε $\lvert x-q\rvert \le x$.

Παράδειγμα 1.2.4.2 Ας προσπαθήσουμε να προσεγγίσουμε το π με έναν ρητό αριθμό, με ακρίβεια 0.001

```
In[9]:= Rationalize[Pi, 0.001]

Out[9]= 201/64
```

□

Δραστηριότητα 1.2.4.3
α) Προσπάθησε να προσεγγίσεις το $\sqrt{2}$ με έναν ρητό αριθμό με ακρίβεια 10^{-6}.
β) Να υπολογίσεις την σταθερά e με ακρίβεια 10 σημαντικών ψηφίων.

1.2.5 Ποια είναι τα δεδομένα που χειρίζεται ένα πρόγραμμα και σε ποιες κατηγορίες χωρίζονται;

Τα δεδομένα χωρίζονται σε δύο κατηγορίες :

Σταθερές. Αυτά που έχουν σταθερή τιμή κατά τη διάρκεια εκτέλεσης του προγράμματος. Οι σταθερές χωρίζονται σε δύο κατηγορίες :

α) στις σταθερές χωρίς όνομα πρδ. στην έκφραση 3.14*R^2 το 3.14 αποτελεί μια σταθερά χωρίς όνομα, και
β) στις συνηθισμένες αριθμητικές σταθερές για τις οποίες το Mathematica έχει κάποιο συνηθισμένο όνομα π.χ. για την σταθερά 3.14159 χρησιμοποιεί το όνομα Pi, για την σταθερά 2.71828 χρησιμοποιεί το όνομα E.

Μεταβλητές. Αυτά που η τιμή τους μεταβάλλεται κατά τη διάρκεια εκτέλεσης του προγράμματος. Πιο συγκεκριμένα, με τον όρο ***μεταβλητή*** εννοώ μια (ή παραπάνω) θέση η οποία δημιουργείται στη μνήμη του Η/Υ, για να δεχτεί ένα συγκεκριμένο τύπο δεδομένων, και η οποία έχει ένα χαρακτηριστικό όνομα που πληροί τους κανόνες που θέσαμε παραπάνω.

1.2.5.1 Ποιες είναι οι πιο συνηθισμένες συμβολικές μαθηματικές σταθερές ;

Όνομα σταθεράς	Παράδειγμα
Pi (π)	In[16]:= **N[Pi, 5]** Out[16]= 3.14159 In[17]:= **Sin[Pi / 2]** Out[17]= 1
E (εκθετική σταθερά)	In[18]:= **N[E, 5]** Out[18]= 2.71828
Degree (ακτίνιο)	In[19]:= **N[Degree, 5]** Out[19]= 0.0174533
I (η φανταστική μονάδα)	In[20]:= **I^2** Out[20]= -1
Infinity (άπειρο)	In[21]:= **1 / Infinity** Out[21]= 0
ComplexInfinity (άπειρη ποσότητα χωρίς καθορισμένη διεύθυνση)	In[22]:= **1 / 0** Power::infy : Infinite expression $\frac{1}{0}$ encountered. Out[22]= ComplexInfinity

1.2.5.2 Με ποιο τρόπο τοποθετούμε τιμές σε μεταβλητές ;

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

Μεταβλητή = Έκφραση Set[Μεταβλητή, Έκφραση]	Κατά τη διάρκεια εκτέλεσης της παραπάνω εντολής ο Η/Υ υπολογίζει πρώτα το αποτέλεσμα της «έκφρασης» και στη συνέχεια, αυτό που θα βρει, θα το αναθέσει στη «μεταβλητή» που υπάρχει αριστερά του =. Στη συνέχεια του προγράμματος κάθε φορά που θα βλέπει την «μεταβλητή» θα την αντικαθιστά από την τιμή

	«έκφραση». Ισοδύναμη της πρώτης έκφρασης είναι η δεύτερη.
Μεταβλητή := Έκφραση SetDelayed[Μεταβλητή, Έκφραση]	Η τοποθέτηση της «έκφρασης» στην «μεταβλητή» γίνεται μόνο την στιγμή που καλούμε την συγκεκριμένη «μεταβλητή». Ισοδύναμη της πρώτης έκφρασης είναι η δεύτερη.

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

Παράδειγμα 1.2.5.3

```
A = Random[Integer, {1, 10}];
B := Random[Integer, {1, 10}];
A1 = Table[A, {5}]
B1 = Table[B, {5}]
{10, 10, 10, 10, 10}
{2, 1, 6, 9, 1}
```

Η πρώτη εντολή υπολογίζει έναν τυχαίο ακέραιο αριθμό μεταξύ 1 και 10 (τον 10) και τον τοποθετεί στην θέση του A. Αντίθετα στην δεύτερη εντολή ορίζουμε ότι ο B θα δεχτεί έναν τυχαίο ακέραιο αριθμό μεταξύ 1 και 10 όταν τον καλέσουμε. Συνεπώς στην τρίτη εντολή επειδή το A έχει ήδη την τιμή 10, όταν προσπαθούμε να πάρουμε έναν πίνακα με 5 αριθμούς ίσους με A, παίρνει και τους 5 ίσους με 10. Αντίθετα όταν προσπαθούμε να δημιουργήσουμε έναν πίνακα με 5 αριθμούς της μορφής B, κάθε φορά που καλούμε τον B για να τον τοποθετήσουμε στον πίνακα, υπολογίζεται η έκφραση που βρίσκεται δεξιά του ίσο στην δεύτερη εντολή και τοποθετείται στον πίνακα B1. Αυτός είναι και ο λόγος που ο πίνακας B1 διαθέτει διαφορετικές τυχαίες τιμές. □

Παράδειγμα 1.2.5.4

```
z = Expand[(x + y)^2]
x^2 + 2 x y + y^2
w := Expand[(x + y)^2]
w
x^2 + 2 x y + y^2
```

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

```
x = 1 + a
1 + a
z
(1 + a)^2 + 2 (1 + a) y + y^2
w
1 + 2 a + a^2 + 2 y + 2 a y + y^2
```

Η διαφορά αυτή οφείλετε στο γεγονός ότι το z έχει ήδη την τιμή του αναπτύγματος του $(x+y)^2$ δηλαδή $x^2+2xy+y^2$ και απλώς αντικαθιστά το x με την ισοδύναμη έκφραση του 1+a στο ανάπτυγμα αυτό. Αντίθετα όταν καλέσουμε το w τότε

τοποθετείτε στην έκφραση $(x+y)^2$ το x=1+a και στη συνέχεια υπολογίζετε το ανάπτυγμα του $(x+y)^2$. □

Παράδειγμα 1.2.5.5

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

2

Η πρώτη εντολή δημιουργεί μια θέση στην μνήμη του Η/Υ για το Α και τοποθετεί την τιμή 1.

Η δεύτερη εντολή βρίσκει το αποτέλεσμα δεξιά του (=) που είναι 1+1 (εφόσον η τιμή του Α από την πρώτη εντολή ήταν 1) και το τοποθετεί στην ίδια θέση της μνήμης που είχε ανοίξει στην πρώτη εντολή για να στεγάσει την τιμή του Α. Παρατηρούμε, δηλαδή, ότι οτιδήποτε βρίσκεται στη μεταβλητή που βρίσκεται αριστερά του ίσον, χάνεται και στη θέση της 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. Οι εντολές αυτές εξηγούνται στη συνέχεια :

a++	Αυξάνει την τιμή του a κατά 1, και επιστρέφει την προηγούμενη τιμή του a
a--	Μειώνει την τιμή του a κατά 1, και επιστρέφει την προηγούμενη τιμή του a
++a	Αυξάνει την τιμή του a κατά 1, και επιστρέφει την νέα τιμή του a
--a	Μειώνει την τιμή του a κατά 1, και επιστρέφει την νέα τιμή του a
a+=da	Αυξάνει την τιμή του a κατά da π.χ. a=a+da, και επιστρέφει την νέα τιμή του a
a-=da	Μειώνει την τιμή του a κατά da π.χ. a=a-da, και επιστρέφει την νέα τιμή του a
a*=c	Πολλαπλασιάζει την τιμή του a με c και εφόσον τοποθετήσει το αποτέλεσμα στο a π.χ. a=a*c, επιστρέφει την νέα τιμή του a
a/=c	Διαιρεί την τιμή του a με c και εφόσον τοποθετήσει το αποτέλεσμα στο a π.χ. a=a/c, επιστρέφει την νέα τιμή του a

Παράδειγμα 1.2.5.6

```
A = 1;
A++
1
A
2
A += 3
5
```

□

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

$\text{Μεταβλητή}_1 = \text{Μεταβλητή}_2 = \ldots = \text{Έκφραση}$

Παράδειγμα 1.2.5.7

```
x = y = 1
1
x + y
2
```

□

Εάν δεν πρόκειται να χρησιμοποιήσουμε μια μεταβλητή στη συνέχεια του προγράμματος/notebook είναι καλύτερα να αποδεσμεύσουμε την μεταβλητή αυτή από τον ορισμό που τις είχαμε δώσει προκειμένου να μην επηρεάσει το υπόλοιπο πρόγραμμα/notebook. Η αποδέσμευση αυτή γίνεται με έναν από τους παρακάτω τρόπους :

x=.	Απομακρύνει όποιον ορισμό είχαμε δώσει στο x.
Unset[x]	
Clear[x]	Απομακρύνει τυχόν ορισμούς αλλά και τιμές που είχαμε προσδώσει στο x. Η Clear παρ'όλα αυτά δεν απομακρύνει χαρακτηριστικά που έχουν αποδοθεί στο x ή μηνύματα που συνοδεύουν το x ή ορισμένες ιδιότητες που του έχουν αποδοθεί. Αν επιπλέον θέλω να αφαιρέσω και τα χαρακτηριστικά αυτά μπορώ να χρησιμοποιήσω την ClearAll στην θέση της Clear.
Remove[x]	Η Remove απομακρύνει οριστικά το σύμβολο x έτσι ώστε να μην αναγνωρίζεται πια από το Mathematica.

Παράδειγμα 1.2.5.7

```
A = {1, 2, 3};
?A
Global`A
```

```
A = {1, 2, 3}
A=.
?A
Global`A
Remove[A]
?A
Information::notfound : Symbol A not found.
```

Στο παραπάνω παράδειγμα είναι φανερό ότι αν απομακρύνουμε την τιμή του A με την εντολή A=., παρατηρούμε ότι παραμένει η ιδιότητα του A ότι είναι μια global μεταβλητή, σε αντίθεση με την εντολή Remove[] που απομακρύνει πλήρως το A καθώς και τις ιδιότητες που το συνοδεύουν. □

Στην θέση της μεταβλητής x μπορούμε επιπλέον να χρησιμοποιήσουμε σύμβολα υποκατάστασης (* για κανέναν ή περισσότερους χαρακτήρες, @ για έναν ή περισσότερους χαρακτήρες εκτός από κεφαλαίους χαρακτήρες) για να δηλώσω ένα μεγαλύτερο σύνολο μεταβλητών π.χ. Clear[“x*”] ώστε να απομακρύνει τις τιμές από όλες τις μεταβλητές που ξεκινούν από το γράμμα x.

Παράδειγμα 1.2.5.8

```
A1 = {1, 2, 3}
{1, 2, 3}
A2 = 2
2
Remove["`A*"]
?A1
Information::notfound : Symbol A1 not found.
?A2
Information::notfound : Symbol A2 not found.
```

□

Παρατηρήσεις

1. Προσέχουμε πολύ κατά την τοποθέτηση των παρενθέσεων/άγκιστρων. Αν δεν είμαστε σίγουροι για το σχεδιασμό των εκφράσεων, τοποθετούμε παρενθέσεις. Ένα κριτήριο που θα μας βοηθήσει στον έλεγχο είναι ότι :

   Αριστερές Παρενθέσεις = Δεξιές Παρενθέσεις
2. Ελέγχουμε αν χρησιμοποιούμε για κάθε μεταβλητή το ίδιο όνομα σε όλα τα μέρη του προγράμματος.
3. Ελέγχουμε αν οι μεταβλητές που υπάρχουν δεξιά του =, έχουν ήδη πάρει τιμές από πιο μπροστά, διαφορετικά το Mathematica τις χρησιμοποιεί σαν σύμβολα. Επίσης χρησιμοποιούμε το N[] ή //N όταν θέλουμε να δηλώσουμε ότι δεν θα δουλέψουμε με συμβολικές εκφράσεις.
4. Ελέγχουμε αν τα ορίσματα των συναρτήσεων έχουν πάρει σωστές τιμές, π.χ. υπόριζο θετικό, όρισμα λογαρίθμου θετικό κ.ο.κ.
5. Συνηθίζουμε να χρησιμοποιούμε ονόματα μεταβλητών που έχουν σχέση με την ποσότητα την οποία παριστούν π.χ. velocity για ταχύτητα, area για εμβαδόν κ.ο.κ. Για να ξεχωρίζουμε τα ονόματα των μεταβλητών μας από αυτά που χρησιμοποιεί το Mathematica συνηθίζουμε να γράφουμε το πρώτο γράμμα της

μεταβλητής με πεζούς χαρακτήρες π.χ. velocity αντί Velocity.

6. Όταν μια έκφραση είναι πολύπλοκη, τη σπάμε σε απλούστερες εκφράσεις π.χ.

$$z = \frac{-b + \sqrt{b^2 - 4ac}}{-b - \sqrt{b^2 - 4ac}}$$

Z1=B+Sqrt[B^2*A*C]
Z2=BSqrt[B^2*A*C]
Z=Z1/Z2

7. Χρησιμοποιούμε τις ενσωματωμένες συναρτήσεις του Mathematica όπου είναι δυνατό π.χ. $-B + Sqrt[B\,\hat{}\,2 - 4AC]$. Με τον τρόπο αυτό παίρνουμε ταχύτερα και καλύτερα αποτελέσματα όταν έχουμε να λύσουμε αριθμητικά προβλήματα.

1.2.6 Συναρτήσεις

Πολλές φορές για τον υπολογισμό εκφράσεων χρειαζόμαστε εκτός από τους γνωστούς αριθμητικούς τελεστές και συναρτήσεις όπως το ημίτονο, ο λογάριθμος κ.λ.π. Το *Mathematica* υποστηρίζει ένα σύνολο συναρτήσεων για τον υπολογισμό τέτοιων εκφράσεων. Τις πιο στοιχειώδεις συναρτήσεις τις δίνουμε παρακάτω.

Αριθμητικές συναρτήσεις

Όνομα	Περιγραφή	Παράδειγμα
Abs[x]	Απόλυτη τιμή του x	In[23]:= **Abs[-3.1]** Out[23]= 3.1 In[24]:= **Abs[3+4 I]** Out[24]= 5
Sqrt[x]	Τετραγωνική ρίζα του x	In[25]:= **Sqrt[4]** Out[25]= 2 In[26]:= **Sqrt[3+4 I]** Out[26]= 2 + i
Sin[x]	Ημίτονο του x σε ακτίνια	In[27]:= **Sin[Pi/2]** Out[27]= 1 In[28]:= **Sin[Pi I]** Out[28]= i Sinh[π]
ArcSin[x]	Τόξο ημιτόνου του x	In[29]:= **ArcSin[1]** Out[29]= $\frac{\pi}{2}$
Cos[x]	Συνημίτονο του x σε ακτίνια	In[30]:= **Cos[Pi/2]** Out[30]= 0

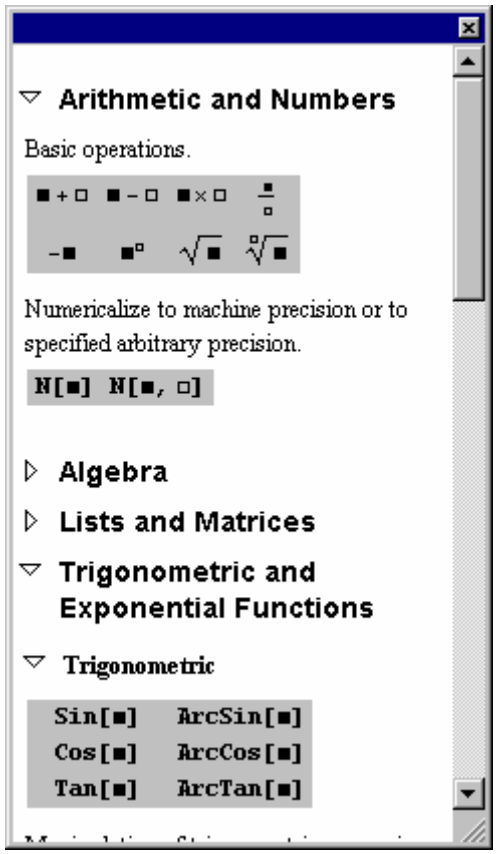
Όνομα	Περιγραφή	Παράδειγμα
ArcCos[x]	Τόξο συνημιτόνου του x	In[31]:= **ArcCos[0]** Out[31]= $\frac{\pi}{2}$
Tan[x]	Εφαπτομένη του x σε ακτίνια	In[32]:= **Tan[Pi/2]** Out[32]= ComplexInfinity
ArcTan[x]	Τόξο εφαπτομένης του x	In[33]:= **ArcTan[Infinity]** Out[33]= $\frac{\pi}{2}$
Exp[x]	e^x	In[34]:= **Exp[0]** Out[34]= 1 In[35]:= **Exp[1+2 I]** Out[35]= e^{1+2i}
Log[x]	Φυσικός λογάριθμος του x	In[36]:= **Log[Exp[2]]** Out[36]= 2 In[37]:= **Log[0]** Out[37]= $-\infty$
Log[b, x]	Λογάριθμος του x με βάση το b	In[38]:= **Log[10, 10^4]** Out[38]= 4
Max[X1, X2,...]	Ο μέγιστος των X1,X2,…	In[39]:= **Max[1, 5, 2, 9]** Out[39]= 9 In[40]:= **Max[{2, 3, 8, 1, -2}]** Out[40]= 8
Min[X1, X2, ...]	Ο ελάχιστος των X1,X2,…	In[41]:= **Min[1, 5, 2, 9]** Out[41]= 1
IntegerPart[x]	Ο πραγματικός αριθμός x χωρίς τα δεκαδικά ψηφία	In[42]:= **IntegerPart[2.3]** Out[42]= 2 In[43]:= **IntegerPart[2.6]** Out[43]= 2
FractionalPart[x]	Το δεκαδικό μέρος του αριθμού x	In[44]:= **FractionalPart[2.6]** Out[44]= 0.6

Όνομα	Περιγραφή	Παράδειγμα
`Round[x]`	Μετατροπή στον πλησιέστερο ακέραιο	In[45]:= **Round[2.3]** Out[45]= 2 In[46]:= **Round[2.6]** Out[46]= 3
`Floor[x]`	Μεγαλύτερος ακέραιος ≤x	In[47]:= **Floor[2.6]** Out[47]= 2

`Mod[x,y]`
ή
`x-IntegerPart[x/y] *y`
	Ακέραιο υπόλοιπο της διαίρεσης x με το y	In[48]:= **Mod[7, 3]**
		Out[48]= 1
		
		In[49]:= **7 - IntegerPart[7 / 3] * 3**
		Out[49]= 1

Παρατηρήσεις.

1. Τα ονόματα των συναρτήσεων στο Mathematica ξεκινούν πάντα με κεφαλαίο γράμμα.
2. Τα ορίσματα των συναρτήσεων δίνονται μέσα σε τετράγωνες αγκύλες [].
3. Εκτός από τα ονόματα των συναρτήσεων μπορούμε να χρησιμοποιήσουμε και τις παλέτες που διαθέτει το Mathematica π.χ. File->Palettes->BasicCalculations



Παράδειγμα 1.2.6.1 Να μετατραπούν οι παρακάτω αλγεβρικές εκφράσεις σε μορφή αποδεκτή από το Mathematica

α) $\dfrac{x+y}{x-y}$ β) $K\left(1+E\right)^{N}$

γ) $\sqrt[3]{x^2+y^2}$ δ) $\dfrac{-B+\sqrt{B^2-4AC}}{2A}$

ε) $\dfrac{\eta\mu\left(2\chi\right)}{\sigma\upsilon\nu^2\left(\chi-1\right)+1}$ στ) $\left|-x+\dfrac{y-1}{\ln(x)}\right|$

ζ) $x-1<y<x+1$ η) $\left|x\right|\le y$

Απάντηση

α) (X+Y)/(X–Y)

β) K*(1+E)^N

γ) (X^2+Y^2)^(1/3)

δ) (–B+Sqrt[B^2–4*A*C)]/(2*A)

ε) Sin[2*X]/((Cos[X–1])^2+1)

στ) Abs[–X+(Y–1)/Log[X]]

ζ) x-1<y<x+1

η) Abs[x]<=y

Δραστηριότητα 1.2.6.2. Υπολογίστε προσεγγιστικά τις εκφράσεις :

α) $\sin^2\left(\dfrac{\pi}{4}\right)+\cos^2\left(\dfrac{\pi}{4}\right)$

β) $e^{\pi}-\pi^{e}$

γ) $\left(1-i\right)\left(1+i\right)$

δ) $\tan^{-1}\left(\dfrac{1}{\pi}\right)$

1.2.7 Σύνθετοι αριθμητικοί υπολογισμοί.

Το Mathematica όπως θα δούμε και στα επιμέρους κεφάλαια διαθέτει ένα σύνολο συναρτήσεων που μπορεί να μας βοηθήσει στην επίλυση σύνθετων υπολογιστικών προβλημάτων. Παρακάτω δίνουμε ορισμένα χαρακτηριστικά παραδείγματα.

```
NSolve[x^2 - 5 x + 6 == 0, x]
{{x → 2.}, {x → 3.}}
```

Η συνάρτηση NSolve[f(x)==0,x] στο παραπάνω παράδειγμα δίνει μια αριθμητική προσέγγιση των λύσεων της εξίσωσης f(x)=0 ως προς x. Όταν μια εξίσωση δεν μπορεί να λυθεί αλγεβρικά, τότε μπορεί να λυθεί με προσεγγιστικές μεθόδους μέσω της συνάρτησης FindRoot[f(x)==0,{x,a}] όπου a είναι ένας αριθμός κοντά στην ρίζα

ο οποίος χρησιμοποιείται για την έναρξη της προσεγγιστικής μεθόδου που θα χρησιμοποιηθεί.

```
NSolve[Cos[x] == x + Log[x], x]
Solve::tdep : The equations appear to involve the variables
    to be solved for in an essentially non-algebraic way. More...
FindRoot[Cos[x] == x + Log[x], {x, 1}]
{x → 0.840619}
```

Μπορούμε να υπολογίσουμε ορισμένα ολοκληρώματα π.χ.

```
NIntegrate[Exp[x^2], {x, 0, 1}]
1.46265
```

Μέσω της συνάρτησης NIntegrate[f(x),{x,a,b}] που υπολογίζει το ορισμένο ολοκλήρωμα της συνάρτησης f(x) στο κλειστό διάστημα [a,b] με προσεγγιστικές μεθόδους.

Το Mathematica έχει μεγάλη εφαρμογή και στη Θεωρία Αριθμών όπως φαίνεται παρακάτω :

```
FactorInteger[2^105 - 1]
{{7, 2}, {31, 1}, {71, 1}, {127, 1}, {151, 1}, {337, 1},
 {29191, 1}, {106681, 1}, {122921, 1}, {152041, 1}}
```

Η συνάρτηση FactorInteger[] παραγοντοποιεί έναν ακέραιο αριθμό. Ο παραπάνω αριθμός είναι ο :

$$7^2 \times 31^1 \times 71^1 \times 127^1 \times 151^1 \times 337^1 \times 29191^1 \times 106681^1 \times 122921^1 \times 152041^1$$

Η συνάρτηση PrimeQ[] ελέγχει αν ένας αριθμός είναι πρώτος ενώ η συνάρτηση Prime[k] εμφανίζει τον k-οστο πρώτο πρώτο αριθμό.

```
PrimeQ[2^7 - 1]
True
Prime[1000]
7919
```

Δραστηριότητα 1.2.7.1 Έλεγξε αν ο $2^{17}-1$ είναι πρώτος αριθμός. Αν όχι παραγοντοποίησε τον. Κάνε το ίδιο για τον $2^{18}-1$.
Δραστηριότητα 1.2.7.2 Προσπάθησε να επιλύσεις την δευτεροβάθμια εξίσωση $x^2-5x+6=0$.

1.3 Συμβολικοί υπολογισμοί

Το Mathematica μπορεί να χειριστεί με την ίδια ευκολία που χειρίζεται τους αριθμούς και σύμβολα. Ας υποθέσουμε για παράδειγμα ότι θέλουμε να παραγοντοποιήσουμε την έκφραση $x^{39}+y^{39}$. Με την συνάρτηση Factor[] παίρνουμε το επιθυμητό αποτέλεσμα :

```
Factor[x^39 + y^39]
```

$$(x+y)\,(x^2-xy+y^2)\,(x^{12}-x^{11}y+x^{10}y^2-x^9y^3+x^8y^4-x^7y^5+x^6y^6-x^5y^7+x^4y^8-x^3y^9+x^2y^{10}-xy^{11}+y^{12})\,(x^{24}+x^{23}y-x^{21}y^3-x^{20}y^4+x^{18}y^6+x^{17}y^7-x^{15}y^9-x^{14}y^{10}+x^{12}y^{12}-x^{10}y^{14}-x^9y^{15}+x^7y^{17}+x^6y^{18}-x^4y^{20}-x^3y^{21}+xy^{23}+y^{24})$$

Αν πάλι θέλουμε να κάνουμε όλες τις δυνατές απλοποιήσεις στο παραπάνω αποτέλεσμα θα εφαρμόσουμε την συνάρτηση Simplify[] στο προηγούμενο αποτέλεσμα (%) και θα έχουμε :

```
Simplify[%]
```

$$x^{39}+y^{39}$$

Αν πάλι θέλουμε το ανάπτυγμα της πιο προηγούμενης έκφρασης θα εφαρμόσουμε την συνάρτηση Expand[] στο πιο προηγούμενο αποτέλεσμα (%%) και θα έχουμε :

```
Expand[%%]
```

$$x^{39}+y^{39}$$

Δραστηριότητα 1.3.1 Προσπάθησε να απλοποιήσεις την έκφραση :

$$x(x-2y)^3+y(2x-y)^3$$

Το Mathematica μπορεί να μας βοηθήσει στην ακριβή επίλυση εξισώσεων :

```
Solve[x^2 - 5 x + a == 0, x]
```

$$\left\{\left\{x\to\frac{1}{2}\left(5-\sqrt{25-4a}\right)\right\},\left\{x\to\frac{1}{2}\left(5+\sqrt{25-4a}\right)\right\}\right\}$$

Η Solve[f(x)==0,x] επιλύει την εξίσωση f(x)=0 ως προς x. Η συνάρτηση Reduce[f(x)==0,x] μάλιστα κάνει και διερεύνηση της εξίσωσης για τις διάφορες τιμές της παραμέτρου :

```
Reduce[a * x + b == 0, x]
```

$$b==0\;\&\&\;a==0\;||\;a\neq 0\;\&\&\;x==-\frac{b}{a}$$

Οι παραπάνω συναρτήσεις μπορούν να χρησιμοποιηθούν και για την επίλυση συστήματος εξισώσεων όπως παρακάτω :

```
Solve[{
  x1 - x2 + 2 * x3 + x4 == -2,
  -2 * x1 + x2 - 3 * x3 - 5 * x4 == 4,
  x1 - x2 + x3 + 6 * x4 == 0,
  2 * x1 + 3 * x2 + 5 * x3 - 7 * x4 == 1}, {x1, x2, x3, x4}]
```

$$\left\{\left\{x1\to-\frac{51}{2},\ x2\to\frac{11}{3},\ x3\to\frac{73}{6},\ x4\to\frac{17}{6}\right\}\right\}$$

Η σύνταξη τους στην περίπτωση αυτή είναι

Solve[{f1(x1,x2,…)==0,f2(x1,x2,..)==0,…},{x1,x2,…}]

και

Reduce[{f1(x1,x2,…)==0,f2(x1,x2,..)==0,…},{x1,x2,…}]

αντίστοιχα.

```
Solve[{
  x1 + x2 - x3 + x5 == 1,
  -x1 - x2 + 2 * x3 - x4 + x5 == 2,
  2 * x1 + 2 * x2 - 3 * x3 + 3 * x4 + x5 == 0,
  x1 + x2 - 3 * x4 + 2 * x5 == 3}, {x1, x2, x3, x4, x5}]
```

```
Solve :: svars :
 Equations  may  not  give  solutions  for  all  "solve "  variables  . More...
```

$$\left\{\left\{x1 \to \frac{9}{2} - x2 - \frac{7\,x5}{2},\ x3 \to \frac{7}{2} - \frac{5\,x5}{2},\ x4 \to \frac{1}{2} - \frac{x5}{2}\right\}\right\}$$

Στο Mathematica μπορούμε να ορίσουμε πίνακες όπως παρακάτω που να περιέχουν αριθμούς αλλά και σύμβολα μαζί :

```
A = {
    {1, 1, 1},
    {a, b, c},
    {a^2, b^2, c^2}}
```

$$\begin{pmatrix} 1 & 1 & 1 \\ a & b & c \\ a^2 & b^2 & c^2 \end{pmatrix}$$

Προκειμένου να έχουμε την παραπάνω μορφή του πίνακα στην απάντηση θα πρέπει να έχουμε επιλέξει από το μενού Cell->DefaultOutputFormatType->TraditionalForm. Στη συνέχεια μπορούμε να υπολογίσουμε την ορίζουσα του πίνακα A π.χ.

```
Det[A]
```

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

καθώς και να παραγοντοποιήσουμε το προηγούμενο αποτέλεσμα π.χ.

```
Factor[%]
```

$$-(a-b)(a-c)(b-c)$$

Μπορούμε να υπολογίσουμε τον αντίστροφο του πίνακα A π.χ.

```
Inverse[A] // Simplify
```

$$\begin{pmatrix} \frac{b\,c}{(a-b)(a-c)} & -\frac{b+c}{(a-b)(a-c)} & \frac{1}{(a-b)(a-c)} \\ \frac{a\,c}{b^2-a\,b-c\,b+a\,c} & \frac{a+c}{(a-b)(b-c)} & \frac{1}{(b-a)(b-c)} \\ \frac{a\,b}{(a-c)(b-c)} & -\frac{a+b}{(a-c)(b-c)} & \frac{1}{(a-c)(b-c)} \end{pmatrix}$$

Η συνάρτηση Inverse[A] υπολογίζει τον αντίστροφο πίνακα του A, ενώ το σύμβολο // σημαίνει ότι εφαρμόζουμε την συνάρτηση που υπάρχει δεξιά του // στο αποτέλεσμα που υπολογίσαμε. Αν a=2, b=2, c=3 τότε οι ιδιοτιμές και τα ιδιοδιανύσματα του πίνακα A είναι αντίστοιχα :

```
a = 2; b = 2; c = 3;
Eigenvalues[A]
```
$\{11, 1, 0\}$
```
Eigenvectors[A]
```
$$\begin{pmatrix} 3 & 8 & 22 \\ -1 & -1 & 1 \\ -1 & 1 & 0 \end{pmatrix}$$

ή και τα δύο μαζί :

```
Eigensystem[A]
```
$$\begin{pmatrix} 11 & 1 & 0 \\ \{3, 8, 22\} & \{-1, -1, 1\} & \{-1, 1, 0\} \end{pmatrix}$$

Το Mathematica μπορεί να χρησιμοποιηθεί στον υπολογισμό παραγώγων $1^{ης}$, $2^{ης}$ κ.ο.κ τάξης μέσω της συνάρτησης D[f(x),{x,τάξη}] ή μέσω της D[f(x),x] αν η τάξη είναι 1 π.χ.

```
D[x^2 - 5x + 6/(x - 1), x]
```
$$2x - \frac{6}{(x-1)^2} - 5$$
```
D[x^2 - 5x + 6/(x - 1), {x, 2}]
```
$$2 + \frac{12}{(x-1)^3}$$
```
D[x^2 - 5x + 6/(x - 1), {x, 3}]
```
$$-\frac{36}{(x-1)^4}$$

Παρόμοια μπορεί να χρησιμοποιηθεί για τον υπολογισμό αόριστων ολοκληρωμάτων :

```
Integrate[(x - 1)/(x^2 - 5x + 6), x]
```
$$2\log(x-3) - \log(x-2)$$
```
Integrate[(x^2 + 1) Exp[x], x]
```
$$e^x(x^2 - 2x + 3)$$

Συμβολικά μπορούμε να λύσουμε απλές διαφορικές εξισώσεις όπως παρακάτω :

```
DSolve[y''[x] == a y'[x] + y[x], y[x], x]
```
$$\left\{\left\{y(x) \to e^{\left(1-\sqrt{2}\right)x} c_1 + e^{\left(1+\sqrt{2}\right)x} c_2\right\}\right\}$$

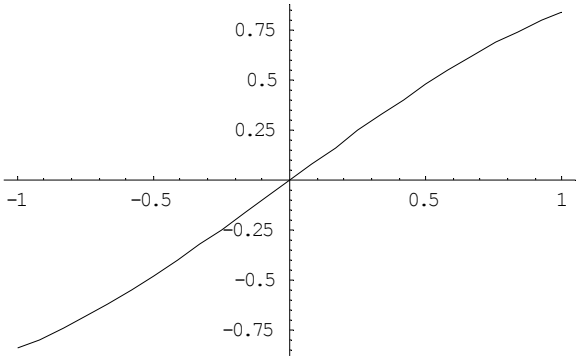
ή συστήματα διαφορικών εξισώσεων :

```
DSolve[{y[x] == -z'[x], z[x] == -y'[x]}, {y[x], z[x]}, x]
```
$$\left\{\left\{z(x) \to \frac{1}{2} e^{-x}\left(1 + e^{2x}\right) c_1 - \frac{1}{2} e^{-x}\left(-1 + e^{2x}\right) c_2,\ y(x) \to \frac{1}{2} e^{-x}\left(1 + e^{2x}\right) c_2 - \frac{1}{2} e^{-x}\left(-1 + e^{2x}\right) c_1\right\}\right\}$$

1.4. Γραφικές παραστάσεις

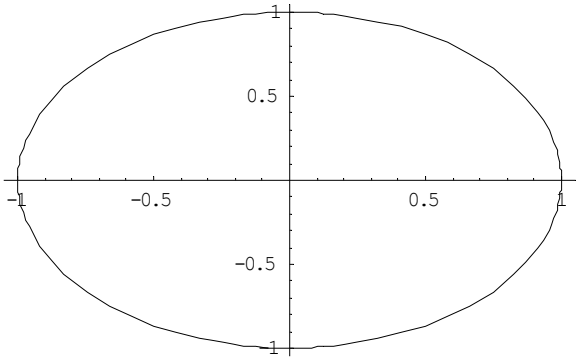
Το *Mathematica* σου δίνει την δυνατότητα να δημιουργείς την γραφική παράσταση μιας συνάρτησης :

```
Plot[Sin[x], {x, -1, 1}]
```



ή και παραπάνω από μια συναρτήσεων μαζί

```
Plot[{Sqrt[1-x^2], -Sqrt[1-x^2]}, {x, -1, 1}]
```



Περισσότερες πληροφορίες για τα ορίσματα της παραπάνω συνάρτησης μπορείς να πάρεις γράφοντας

```
?Plot
Plot[f, {x, xmin, xmax}] generates a plot of f as a function of x from xmin to xmax.
   Plot[{f1, f2, ... }, {x, xmin, xmax}] plots several functions fi. More...
```

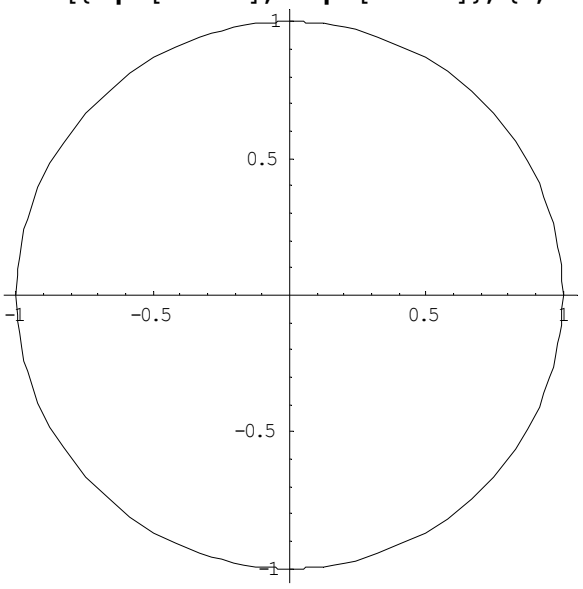
ή

```
??Plot
Plot[f, {x, xmin, xmax}] generates a plot of f as a function of x from xmin to xmax.
   Plot[{f1, f2, ... }, {x, xmin, xmax}] plots several functions fi. More...
```

```
Attributes[Plot] = {HoldAll, Protected}

Options[Plot] = {AspectRatio → 1/GoldenRatio, Axes → Automatic, AxesLabel → None,
  AxesOrigin → Automatic, AxesStyle → Automatic, Background → Automatic,
  ColorOutput → Automatic, Compiled → True, DefaultColor → Automatic,
  DefaultFont :→ $DefaultFont, DisplayFunction :→ $DisplayFunction, Epilog → {},
  FormatType :→ $FormatType, Frame → False, FrameLabel → None, FrameStyle → Automatic,
  FrameTicks → Automatic, GridLines → None, ImageSize → Automatic,
  MaxBend → 10., PlotDivision → 30., PlotLabel → None, PlotPoints → 25,
  PlotRange → Automatic, PlotRegion → Automatic, PlotStyle → Automatic,
  Prolog → {}, RotateLabel → True, TextStyle :→ $TextStyle, Ticks → Automatic}
```

Συνεπώς μπορούμε να αλλάξουμε το AspectRatio από 1/φ σε Automatic και να έχουμε

```
Plot[{Sqrt[1 - x^2], -Sqrt[1 - x^2]}, {x, -1, 1}, AspectRatio → Automatic]
```



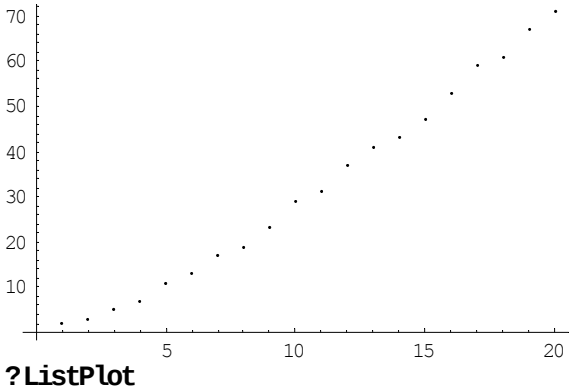
Μπορούμε και να δημιουργήσουμε ένα σύνολο σημείων (ή να καλέσουμε ένα σύνολο σημείων από ένα αρχείο)

```
Table[{i, Prime[i]}, {i, 1, 20}]
```

1	2
2	3
3	5
4	7
5	11
6	13
7	17
8	19
9	23
10	29
11	31
12	37
13	41
14	43
15	47
16	53
17	59
18	61
19	67
20	71

και να σχεδιάσουμε την γραφική παράσταση των σημείων αυτών

```
ListPlot[%]
```

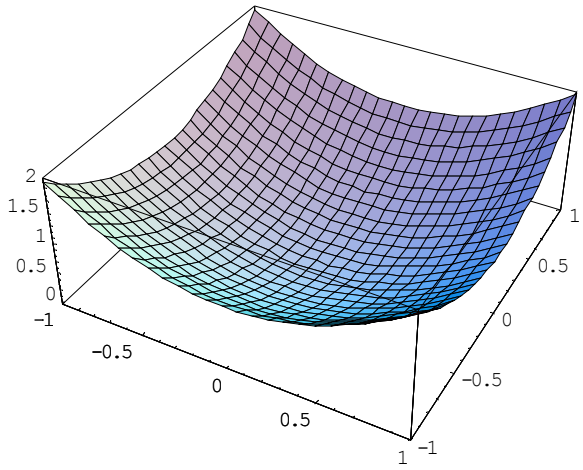



```
?ListPlot
```

```
ListPlot[{y1, y2, ... }] plots a list of values. The x coordinates
   for each point are taken to be 1, 2, ... . ListPlot[{{x1, y1}, {x2,
   y2}, ... }] plots a list of values with specified x and y coordinates. More...
```

Μπορούμε να σχεδιάσουμε και γραφικές παραστάσεις συναρτήσεων δύο μεταβλητών

```
Plot3D[x^2 + y^2, {x, -1, 1}, {y, -1, 1}]
```

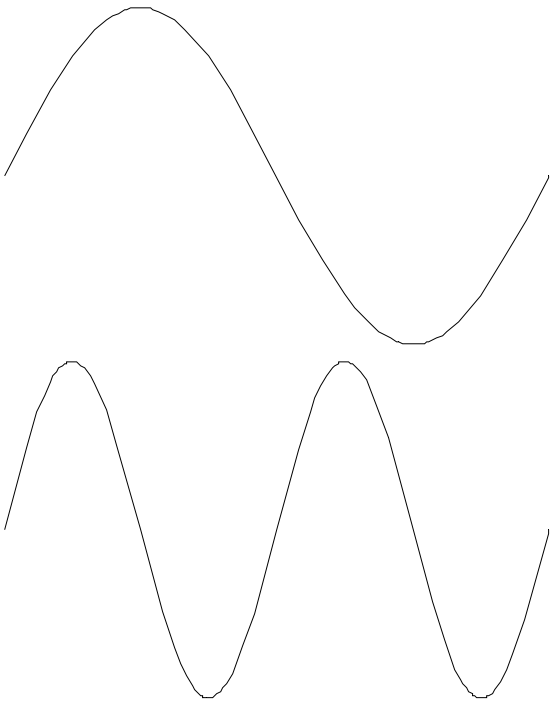


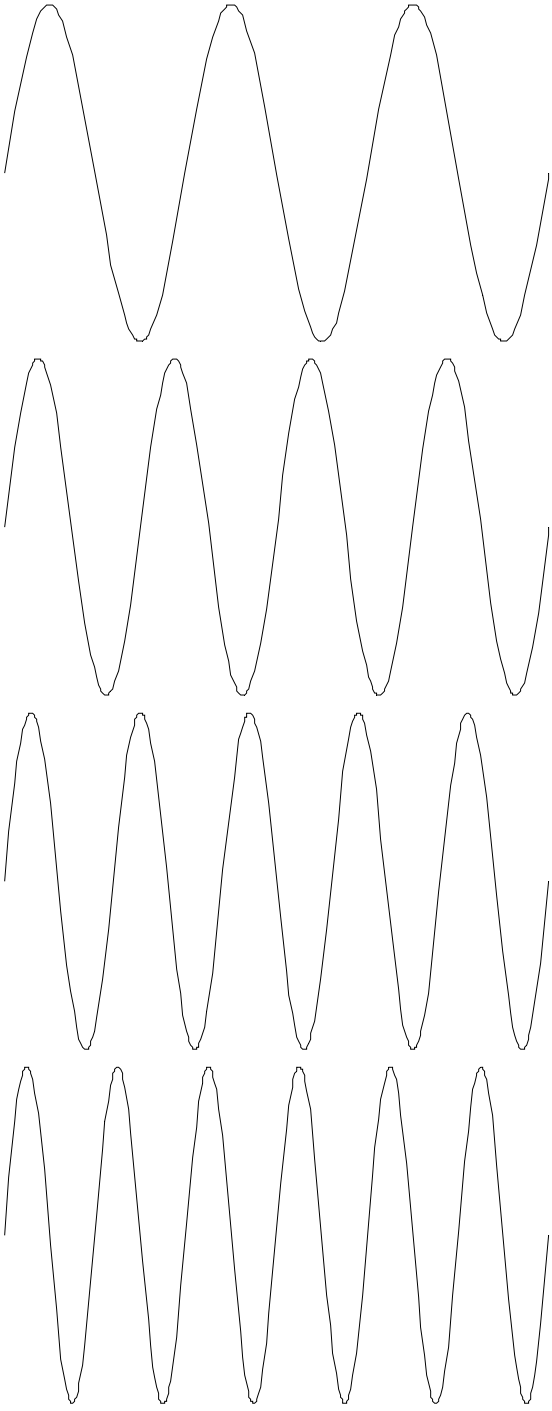

```
?Plot3D
```

Plot3D[f, {x, xmin, xmax}, {y, ymin, ymax}] generates a three-dimensional plot of f as a function of x and y. Plot3D[{f, s}, {x, xmin, xmax}, {y, ymin, ymax}] generates a three-dimensional plot in which the height of the surface is specified by f, and the shading is specified by s. More...

Επίσης μπορούμε να δώσουμε κίνηση στα γραφικά μας.

```
<< Graphics`Animation`
Animate[Plot[Sin[n x], {x, 0, 2 Pi}, Axes → False], {n, 1, 6, 1}]
```





? Animate

```
Animate[command, iterator, options...] uses the iterator to
   run the specified graphics command, and animates the results. More...
```

Αρκεί να κάνεις διπλό κλίκ σε κάποια από τις παραπάνω γραφικές παραστάσεις.

1.5 Προγραμματισμός

Το *Mathematica* διαθέτει μια υψηλού επιπέδου γλώσσα προγραμματισμού η οποία μας δίνει την δυνατότητα να δημιουργούμε τις δικές μας συναρτήσεις επεκτείνοντας με τον τρόπο αυτό τις δυνατότητες της γλώσσας αυτής. Η γλώσσα προγραμματισμού του Mathematica εκτός από την δυνατότητα να διαχειρίζεται σύμβολα και νούμερα, διαθέτει επίσης και τρεις διαφορετικές μεθοδολογίες προγραμματισμού : τον διαδικασιακό προγραμματισμό (procedural programming), τον συναρτησιακό προγραμματισμό (functional programming), και τον κανονοκεντρικό προγραμματισμό (rule-based programming).

1.5.1 Διαδικασιακός προγραμματισμός (procedural programming)

Η μεθοδολογία του διαδικασιακού προγραμματισμού, μέσω γνωστών γλωσσών διαδικασιακού προγραμματισμού, είναι αυτή που διδάσκεται συνήθως πρώτα σε σχολεία αλλά και Πανεπιστήμια. Γλώσσες διαδικασιακού προγραμματισμού είναι η C++, η Fortran 90/95, η Pascal κ.λ.π..

Στον διαδικασιακό προγραμματισμό πρέπει εμείς να περιγράψουμε, με την μορφή εντολών, τα βήματα που θα πρέπει να ακολουθήσει ο υπολογιστής για να επιλύσει ένα πρόβλημα. Ένα τυπικό παράδειγμα διαδικασιακού προγραμματισμού είδαμε προηγουμένως αλλά και στην ενότητα 1.1. Στον διαδικασιακό προγραμματισμό οι μεταβλητές δεν θεωρούνται πλέον συναρτήσεις αλλά θέσεις της μνήμης του Η/Υ οι οποίες μπορούν να αλλάξουν τιμή. Οι συνθήκες συνήθως πετυχαίνονται με εντολές όπως η If, Which, Switch ενώ οι επαναλήψεις με εντολές όπως οι While, For, Do.

Στον διαδικασιακό προγραμματισμό συνήθως διαχωρίζουμε το πρόβλημα μας σε επιμέρους ανεξάρτητα προβλήματα (διαδικασίες, γνωστές σε διάφορες γλώσσες προγραμματισμού και ως functions, procedures, subroutines) τα οποία αρχικά επιλύουμε και ελέγχουμε για την ορθότητα τους και στη συνέχεια τα συνθέτουμε ώστε να πάρουμε την επίλυση του αρχικού μας προβλήματος, μια μέθοδος γνωστή ως σχεδίαση από πάνω προς τα κάτω (top-down design). Εναλλακτικές μεθόδους σχεδίασης μπορεί να βρει ο ενδιαφέρων αναγνώστης στο βιβλίο (Α. Καμέας, 2000, Εισαγωγή στη Πληροφορική : Τεχνικές Προγραμματισμού, Τόμος Β, ΕΑΠ). Παρόλο που η χρήση του διαδικαστικού προγραμματισμού είναι δελεαστική για τους γνώστες αυτής της τεχνικής λόγω εμπειρίας από άλλες διαδικαστικές γλώσσες προγραμματισμού, θα θέλαμε να επισημάνουμε ότι συνήθως οδηγεί σε πολύπλοκες μεθοδολογίες επίλυσης με αρνητικές συνέπειες στον χρόνο εκτέλεσης των προγραμμάτων σε σχέση με την μεθοδολογία του συναρτησιακού προγραμματισμού.

Παράδειγμα 1.5.1 Στο παρακάτω πρόγραμμα χρησιμοποιούμε την μεθοδολογία του διαδικασιακού προγραμματισμού για τον υπολογισμό της αναδρομικής ακολουθίας

$$a_n = \frac{1}{2}\left(a_{n-1} + \frac{2}{a_{n-1}}\right), a_0 = 1$$

για n=100000 :

```
t = 1.0;
Do[t = (1/2) (t + 2/t), {100000}];
t
```

1.41421

■

Η εντολή Do είναι εντολή επανάληψης και η σύνταξη της δίνεται παρακάτω :

```
?Do
Do[expr, {imax}] evaluates expr imax times. Do[expr, {i, imax}] evaluates expr with
   the variable i successively taking on the values 1 through imax (in steps
   of 1). Do[expr, {i, imin, imax}] starts with i = imin. Do[expr, {i, imin,
   imax, di}] uses steps di. Do[expr, {i, imin, imax}, {j, jmin, jmax}, ... ]
   evaluates expr looping over different values of j, etc. for each i. More...
```

Άλλες εντολές επανάληψης είναι οι While και For

```
?While
While[test, body] evaluates test, then
   body, repetitively, until test first fails to give True. More...
```

```
s = 0;
i = 1;
While[i ≤ 1001, (s = s + i; i = i + 2)];
s
```

```
?For
For[start, test, incr, body] executes start, then
   repeatedly evaluates body and incr until test fails to give True. More...
```

```
For[i = 1, i ≤ 3, ++i, Print[Prime[i]]]
```

2

3

5

Εντολές συνθήκης που χρησιμοποιούμε είναι οι If, Which και Switch

```
?If
If[condition, t, f] gives t if condition evaluates
   to True, and f if it evaluates to False. If[condition, t, f, u]
   gives u if condition evaluates to neither True nor False. More...
```

```
f[x_] := If[x > 0, x^2 - 3, x^2 + 3]
f[1]
```

−2

```
f[-1]
```

4

```
?Which
Which[test1, value1, test2, value2, ... ]
   evaluates each of the testi in turn, returning the value of the
   valuei corresponding to the first one that yields True. More...
```

```
Which[2 > 3, 1, 1 > 4, 2, 2 == 2, 3]
```

3

```
?Switch
Switch[expr, form1, value1, form2, value2, ... ] evaluates
   expr, then compares it with each of the formi in turn, evaluating and
   returning the valuei corresponding to the first match found. More...
```

```
f[x_] := Switch[Head[x],
  Plus, x,
  Times, x^2,
  List, Reverse[x]]
```

```
f[a * b]
```

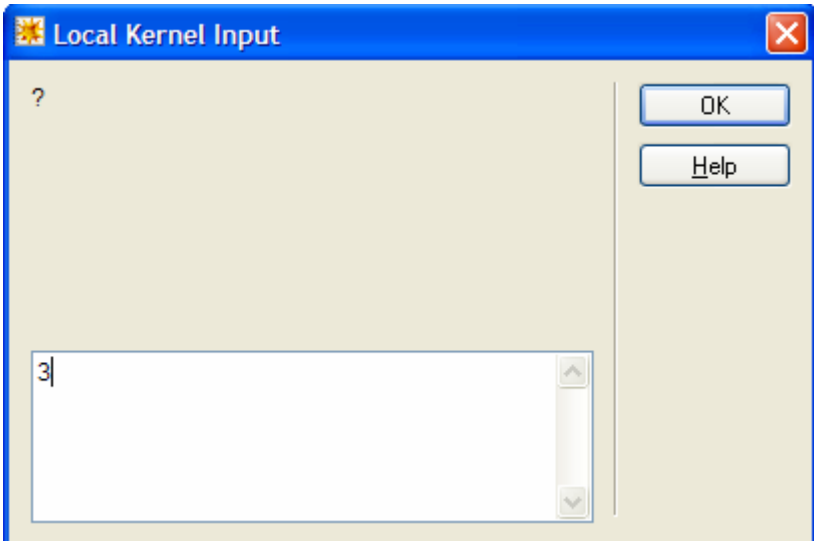
$a^2 b^2$

Για την ανάθεση τιμών σε μεταβλητές εκτός από τον τελεστή ανάθεσης (=) χρησιμοποιούμε και την συνάρτηση Input[]

```
?Input
```

```
Input[ ] interactively reads in one Mathematica expression.
   Input["prompt"] requests input, using the specified string as a prompt. More...
```

```
a = Input[]
```



3

1.5.2 Συναρτησιακός προγραμματισμός (functional programming)

Το επίκεντρο του συναρτησιακού προγραμματισμού είναι ο υπολογισμός εκφράσεων και όχι η εκτέλεση εντολών (ανάθεση τιμών σε μεταβλητές όπως στον διαδικασιακό προγραμματισμό). Οι εκφράσεις στην γλώσσα του συναρτησιακού προγραμματισμού δημιουργούνται με την χρήση συναρτήσεων οι οποίες δέχονται ως ορίσματα απλές τιμές ή άλλες συναρτήσεις. Στον συναρτησιακό προγραμματισμό από την στιγμή που μια μεταβλητή δεχθεί μια τιμή, δεν μπορεί να θεωρηθεί ως μεταβλητή η οποία δέχεται αλλαγές, αλλά αντίθετα θεωρείται ως μια συνάρτηση που έχει καθορισμένη τιμή στο υπόλοιπο πρόγραμμα. Οι επαναλήψεις στον συναρτησιακό προγραμματισμό πραγματοποιούνται συνήθως με αναδρομή ενώ οι συνθήκες που χρειαζόμαστε υλοποιούνται με την εντολή if. Παρακάτω δίνουμε ένα παράδειγμα όπου φαίνεται η διαφορά του συναρτησιακού με τον διαδικασιακό προγραμματισμό.

Παράδειγμα 1.5.2.1 Να υπολογισθεί το άθροισμα των αριθμών 1,2,3,…,10.

α) μέσω συναρτησιακού προγραμματισμού :

```
Sum[i, {i, 1, 10}]
55
```

Η μέθοδος υπολογισμού που χρησιμοποιείται είναι η εφαρμογή συναρτήσεως.

β) μέσω διαδικασιακού προγραμματισμού :

```
s = 0; For[i = 1, i ≤ 10, ++i, s = s + i]; s
55
```

Η μέθοδος υπολογισμού που χρησιμοποιείται είναι η ανάθεση μεταβλητών. □

Κύρια χαρακτηριστικά του συναρτησιακού προγραμματισμού είναι : α) δυνατότητα συγγραφής προγραμμάτων δομημένων με σαφήνεια και λιτότητα τα οποία διαθέτουν ένα υψηλό επίπεδο αφαιρετικότητας, β) μας παρέχουν πολύ ισχυρά εργαλεία για την επίλυση προβλημάτων, γ) βοηθούν στην ελάττωση του χρόνου και του κόστους παραγωγής ενός προγράμματος (δες παραπάνω παράδειγμα), δ) αντιμετωπίζουν με επιτυχία το πρόβλημα του μεγέθους και της πολυπλοκότητας που έχουν τα σύγχρονα προβλήματα. Το κύριο μειονέκτημα του συναρτησιακού προγραμματισμού αφορά την μεγάλη απαίτηση σε ταχύτητα επεξεργασίας από τον Η/Υ, σε σχέση με τα προγράμματα τα οποία έχουν γραφεί με διαδικασιακό προγραμματισμό. Οι αρχές του συναρτησιακού προγραμματισμού βασίστηκαν στην ανάλυση λάμδα από τους Alonzo Church και Haskell Curry κατά τη δεκαετία 1920-1940 και στη συνέχεια με τη δημιουργία γλωσσών συναρτησιακού προγραμματισμού όπως η Lisp από τον John McCarthy (δεκαετία 1960), FP από τον John Backus (1978), ML από τον Roobin Milner (μέσα δεκαετίας 1970), Miranda από τον David Turner (τέλη δεκαετίας 1970 – δεκαετία 1980), γλώσσα Haskell και Haskell 98 (1988 και 1999 αντίστοιχα) κ.λ.π. .

1.5.3 Κανονοκεντρικός προγραμματισμός (rule-based programming)

Ο όρος κανόνας (rule) στον κανονοκεντρικό προγραμματισμό αναφέρετε στον ορισμό της συνάρτησης. Ένα κανονοκεντρικό πρόγραμμα αποτελείται από ένα σύνολο ορισμών συναρτήσεων τα οποία όμως αφορούν πάντα την ίδια συνάρτηση. Κάθε ορισμός αναφέρεται σε διαφορετική μορφή που μπορούν να έχουν τα ορίσματα της συνάρτησης. Όταν λοιπόν ζητήσουμε να εκτελεστεί η συνάρτηση το πρόγραμμα θα αποφασίσει το πώς θα ενεργήσει ανάλογα με την μορφή που έχουν τα ορίσματα της συγκεκριμένης συνάρτησης. Στον διαδικασιακό προγραμματισμό, αντίθετα, θα πρέπει ο προγραμματιστής να ορίσει μέσω εντολών συνθήκης και ενός πολύπλοκου προγράμματος (spaghetti programming) την συμπεριφορά της συνάρτησης για διαφορετικά ορίσματα. Η δε διόρθωση και συντήρηση τέτοιου είδους διαδικασιακών προγραμμάτων αποτελεί ένα δύσκολο εγχείρημα διότι θα πρέπει να καταλάβεις την δύσκολη λογική που εμπεριέχει το πρόγραμμα. Στον κανονοκεντρικό προγραμματισμό η εκτέλεση των εντολών δεν είναι ακολουθιακή όπως στον διαδικασιακό προγραμματισμό, αλλά βασίζεται στην ενεργοποίηση συγκεκριμένων συνθηκών. Μια χαρακτηριστική γλώσσα κανονοκεντρικού προγραμματισμού εκτός του Mathematica είναι η Prolog.

Στο επόμενο παράδειγμα δείχνουμε πως θα μπορούσε να ορισθεί μαθηματικά η πρόσθεση δύο ακεραίων αριθμών μέσω κανόνων.

Παράδειγμα 1.5.3.1 Έστω η συνάρτηση succ(n) (succ(n)=n+1) που δίνει τον επόμενο αριθμό του n και με την συνάρτηση f(n,m) ορίζουμε την συνάρτηση που μας δίνει το αποτέλεσμα της πρόσθεσης των ακεραίων n,m. Η συνάρτηση f(n,m) μπορεί να ορισθεί μέσω των 2 κανόνων :

f(0,m)=m ($1^{ος}$ κανόνας)
f(n,m)=succ(f(n-1,m)) ($2^{ος}$ κανόνας)

Ας δούμε ένα παράδειγμα υπολογισμού της συνάρτησης f. Έστω ότι θέλουμε να υπολογίσουμε το άθροισμα 2+2, τότε θα έχουμε :

f(2,2) = f(succ(1),2) = (από $2^{ο}$ κανόνα)
= succ(f(1,2)) = succ(f(succ(0),2)) = (από $2^{ο}$ κανόνα)
= succ(succ(f(0,2))) = (από $1^{ο}$ κανόνα)
= succ(succ(2)) = succ(3) = 4

Το παραπάνω παράδειγμα μπορεί εύκολα να υλοποιηθεί στο περιβάλλον του Mathematica, ως εξής :

Βήμα 1. Ορίζουμε την συνάρτηση που μας δίνει τον επόμενο ακέραιο του n.

```
succ[n_Integer] := n + 1
```

Βήμα 2. Δίνουμε τους δύο κανόνες της συνάρτησης f που υπολογίζει το άθροισμα δύο ακεραίων.

```
f[0, m_Integer] := m
f[n_Integer, m_Integer] := succ[f[n - 1, m]]
```

Βήμα 3. Υπολογίζουμε τα βήματα που θα κάνει το Mathematica για τον υπολογισμό της τιμής f[2,2].

```
Trace[f[2, 2]]
{f[2, 2], succ[f[2 - 1, 2]], {{2 - 1, 1},
  f[1, 2], succ[f[1 - 1, 2]], {{1 - 1, 0}, f[0, 2], 2},
  succ[2], 2 + 1, 3}, succ[3], 3 + 1, 4}
```

□

Παράδειγμα 1.5.3.2 Να υπολογισθεί ο $20^{ος}$ όρος της ακολουθίας Fibonacci.

$$F_1 = 1, F_2 = 1, F_n = F_{n-1} + F_{n-2}$$

Στο Mathematica ορίζουμε τους παραπάνω κανόνες :

```
fib[1] = fib[2] = 1;
fib[n_Integer] := fib[n - 1] + fib[n - 2]
```

και υπολογίζουμε το αποτέλεσμα

```
fib[20]
```

6765

Αν θέλουμε να δούμε με ποιον τρόπο υπολογίσθηκε ο $3^{ος}$ όρος της ακολουθίας γράφουμε :

```
Trace[fib[3]]
```

{fib(3), fib(3 – 2) + fib(3 – 1), {{3 – 1, 2}, fib(2), 1}, {{3 – 2, 1}, fib(1), 1}, 1 + 1, 2}

■

1.6 Σύνοψη

Στην ενότητα αυτή, προσπαθήσαμε να δώσουμε μια πολύ συνοπτική περιγραφή των δυνατοτήτων του Mathematica. Σε επιμέρους κεφάλαια θα δούμε τις δυνατότητες του Mathematica σε ειδικά προβλήματα της Γραμμικής Άλγεβρας και του Λογισμού μιας μεταβλητής.

1.7 Απαντήσεις στις δραστηριότητες

Δραστηριότητα 1.2.1.1.2

α)

```
N[31*4/23-52]
```

−46.6087

β)

```
N[(32+24-23)+22/(2-34)]
```

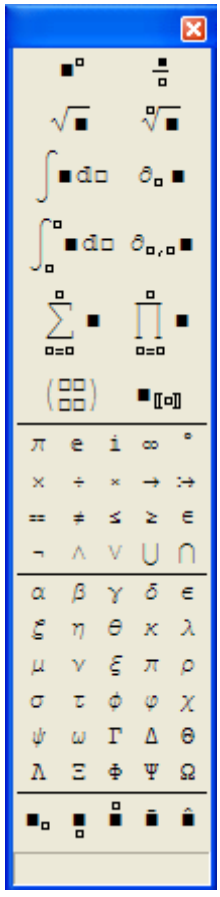
32.3125

Αν έχοντας πατημένο το πλήκτρο Ctrl πατήσεις το πλήκτρο / θα έχεις την μορφή του κλάσματος στο Mathematica όπου μπορείς να προσθέσεις τον αριθμητή και παρονομαστή που θέλεις. Η μετακίνηση από τον αριθμητή στον παρονομαστή γίνεται με το πλήκτρο TAB ή κάνοντας κλίκ στον αριθμητή/παρονομαστή.

$$N\left[(32+24-23)+\frac{22}{2-34}\right]$$

32.3125

Μπορείς να βοηθηθείς και με το εργαλείο ($\frac{\blacksquare}{\blacksquare}$) από την παλέτα BasicInput



Δραστηριότητα 1.2.1.3.2

```
(1 < 2) && (4 < 2)
```
False
```
(2 > 5) || (3 > 6)
```
False

Δραστηριότητα 1.2.2.3

(α)

$$\underbrace{(2>5)}_{\text{False}} \&\& \underbrace{(3>1)}_{\text{True}} \| \underbrace{(4>3)}_{\text{True}} = \text{False}\&\&\text{True}\|\text{True} =$$

$$= \underbrace{False \,\&\, \&\text{True}}_{False} \| True = \underbrace{False \| True}_{True} = True$$

```
(2 > 5) && (3 > 1) || (4 > 3)
```
True

(β)

(5+4/2)^2^3=(5+2)^2^3=7^2^3=7^8=5764801

```
(5 + 4 / 2) ^2^3 // Simplify
```
5764801

Δραστηριότητα 1.2.4.3

(α)

```
Rationalize[Sqrt[2], 10^(-6)]
```
$\frac{1393}{985}$

(β)

```
N[E, 10]
```
2.718281828

Δραστηριότητα 1.2.6.2.

α)

```
Sin[Pi / 4]^2 + Cos[Pi / 4]^2
```
1

ή

```
Sin[Pi/4]^2 + Cos[Pi/4]^2
```
1

β)

```
N[E^π - π^E]
```
0.681535

Η αναπαράσταση του π γίνεται με το συνδυασμό των πλήκτρων [ESC]pi[ESC], ενώ της δύναμης με το Ctrl+^.

γ)

```
(1 - I) (1 + I) // ComplexExpand
```
2

Η ComplexExpand[] αναπτύσει το αποτέλεσμα της μιγαδικής πράξης.

δ)

```
N[ArcTan[1/π]]
```

0.308169

Δραστηριότητα 1.2.7.1

```
PrimeQ[2^17 - 1]
```

```
True
```

```
PrimeQ[2^18 - 1]
```

```
False
```

```
FactorInteger[2^18 - 1]
```

```
{{3, 3}, {7, 1}, {19, 1}, {73, 1}}
```

Δραστηριότητα 1.2.7.2

```
NSolve[x^2 - 5 x + 6 == 0, x]
```

```
{{x → 2.}, {x → 3.}}
```

Δραστηριότητα 1.3.1

```
Simplify[x (x - 2 y) ^3 + y (2 x - y) ^3]
```

```
(x - y) (x + y)^3
```