ΕΚΠΑ Σχολή Θετικών Επιστημών
Τμήμα Μαθηματικών

Αριθμητική Γραμμική Άλγεβρα

Αριθμητική Κινητής Υποδιαστολής στη Julia

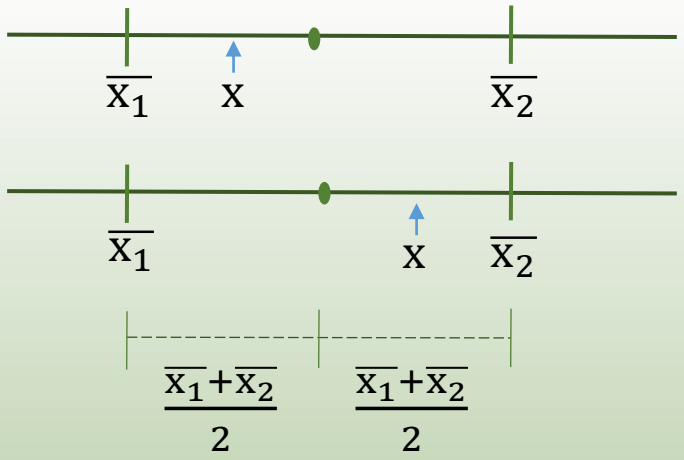
Οκτώβριος 2021

Ιορδάνης Ιωάννης

Καθηγήτρια: Μαριλένα Μητρούλη

Μετατροπή Πραγματικού Αριθμού x σε Αριθμό Κινητής Υποδιαστολής $\bar{x}$

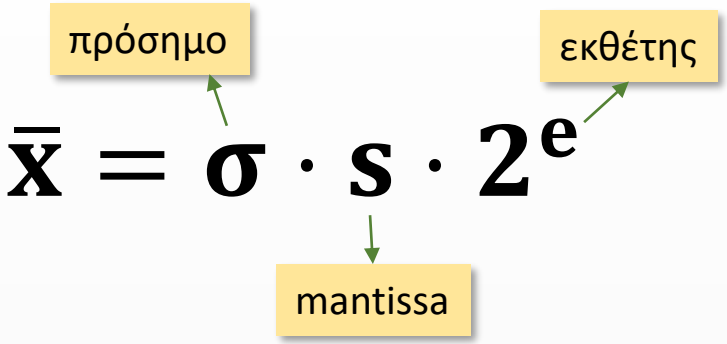
Στρογγύλευση



$x \rightarrow \overline{x_1}$

$x \rightarrow \overline{x_2}$

Μορφή Αριθμού Κινητής Υποδιαστολής



Παράδειγμα : $\bar{x} = 1 \cdot 1.57075 \cdot 2^{1} = 3.1415$

Μορφή Αριθμού Κινητής Υποδιαστολής

$$\bar{x} = \sigma \cdot s \cdot 2^{e}$$

πρόσημο : $\sigma = \begin{cases} 1, \text{θετικός} \\ -1, \text{αρνητικός} \end{cases}$

όρια εκθέτη : $m, M: m \le e < M$ (καθορίζουν το εύρος των αριθμών $\bar{x}$)

mantissa : $s = 1.(b_1 b_2 \dots b_d)_2 = 1 + b_1 \frac{1}{2^1} + b_2 \frac{1}{2^2} + \cdots + b_d \frac{1}{2^d}, \quad 1 \le s < 2$

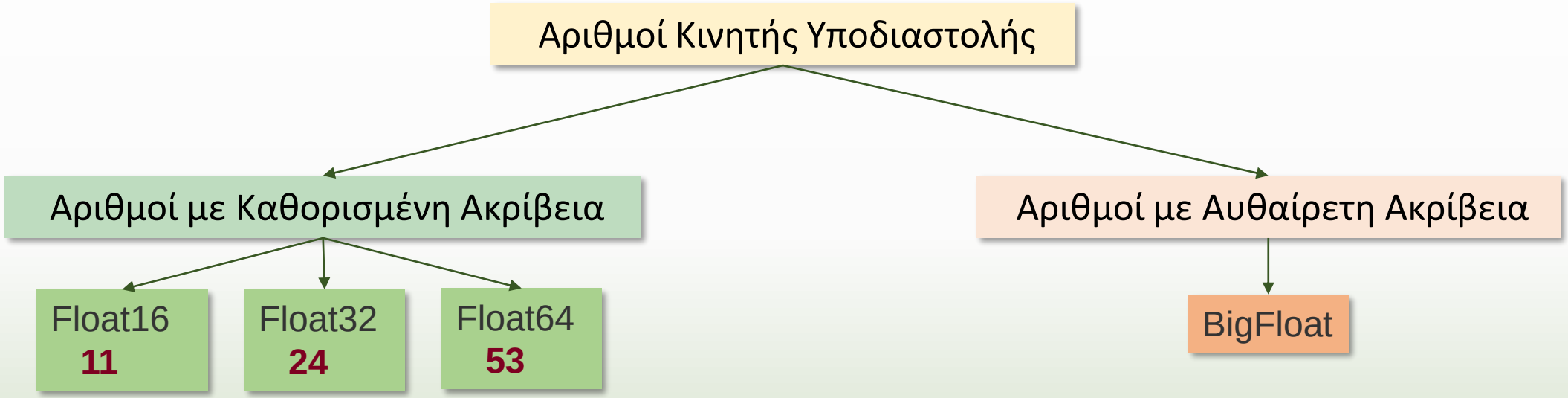
πλήθος ψηφίων σε mantissa: t=1+d (καθορίζουν την ακρίβεια των αριθμών $\bar{x}$)

Παράδειγμα

Είναι εφικτή η προσέγγιση του x = 3.1415 με τον $\bar{x} = 1 \cdot 1.57075 \cdot 2^1$;

Απώλεια στην ακρίβεια (t=11) $1.57075 \rightarrow 1.1001001000 \rightarrow 1.5703125$

Τύποι (Κατηγορίες) Αριθμών Κινητής Υποδιαστολής

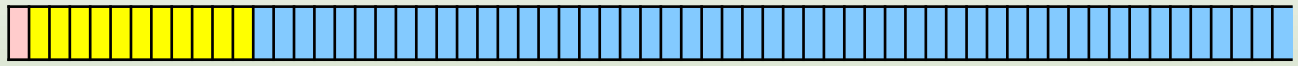


Τύποι Αριθμών Κινητής Υποδιαστολής με καθορισμένη ακρίβεια

Τύπος (julia)	Τίτλος	Σύνολο (bits)	Πρόσημο (bit)	Εκθέτης (bits)	Ελάχιστος εκθέτης m (τιμή)	Μέγιστος εκθέτης M (τιμή)	Bias (τιμή)	d=t-1 (bits)	Ακρίβεια t (bits)
Float16	Μισή	16	1	5	-14	15	15	10	11
Float32	Απλή	32	1	8	-126	127	127	23	24
Float64	Διπλή	64	1	11	-1022	1023	1023	52	53

Δυαδική μορφή Float64 Αριθμού

1 bit **11 bits** **52 bits**



Αλγόριθμος αποθήκευσης

Πρόσημο : Αν στο bit αποθηκεύεται η τιμή 0, ο $\bar{x}$ είναι θετικός, αλλιώς ο $\bar{x}$ είναι αρνητικός $(-1)^{\text{τιμή}}$

Εκθέτης : Αποθηκεύεται η τιμή e+bias, όπου bias σταθερά που εξαρτάται από τον τύπο του αριθμού

Mantissa : Αποθηκεύονται μόνο τα d bits. Δεν αποθηκεύεται το ακέραιο μέρος που είναι πάντα 1.

Μεγαλύτερο t , μεγαλύτερη ακρίβεια

Ακριβής αποθήκευση του x=3.1415 αν $\bar{x} = 1 \cdot 1.57075 \cdot 2^1$

Με τι ακρίβεια αποθηκεύεται η mantissa;

Ακρίβεια t=11 (Float16)

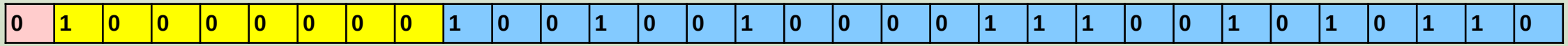
0 16 0.5703125

- 15 (bias) + 1 (το ακέραιο μέρος της mantissa)

--- ---

= 1 = 1.5703125

Ακρίβεια t=24 (Float32)



0 128 0.570749998

- 127 + 1

--- ---

= 1 = 1.570749998

Δυαδική Μορφή Αριθμού Κινητής Υποδιαστολής (Julia)

(a, b, c) = (1, 1.0, -1.0)
τύποι αριθμών
typeof(a) = Int64
typeof(b) = Float64
typeof(c) = Float64

---

bitstring(a)= 0000000000000000000000000000000000000000000000000000000000000001

bitstring(b)= **0**01111111111**0000000000000000000000000000000000000000000000000000** $1.0 = 1 \cdot 1.0 \cdot 2^0$

bitstring(c)= **1**01111111111**0000000000000000000000000000000000000000000000000000** $-1.0 = -1 \cdot 1.0 \cdot 2^0$
(1023-1023=0)

(b, sign(b), exponent(b), significand(b)) = (1.0, 1.0, 0, 1.0)
(c, sign(c), exponent(c), significand(c)) = (1.0, -1.0, 0, 1.0)

Μετατροπή σε τύπο Αριθμού Κινητής Υποδιαστολής (Julia)

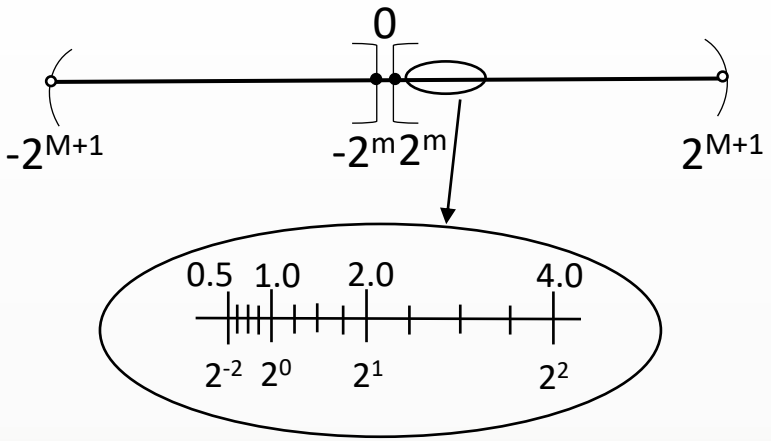
```
x = 3.1415
x1 = Float16(x) = Float16(3.14)
(Float64(x1),          typeof(x1),    Float64(sign(x1)),     exponent(x1),      Float64(significand(x1))) =
(3.140625   ,          Float16    ,   1 .0                   ,    1             ,      1.5703125               )
-------------------------------------------------------------------------------------------------------------------
x2 = Float32(x) = 3.1415f0
(Float64(x2),          typeof(x2),   Float64(sign(x2)),      exponent(x2),    Float64(significand(x2))) =
 (3.14149999618,   Float32    ,   1.0                   ,     1             ,   1.570749980926514   )
-------------------------------------------------------------------------------------------------------------------
x3 = Float64(x) = 3.1415
(Float64(x3),          typeof(x3),     sign(x3)),                exponent(x3),              significand(x3)) =
(3.1415      ,          Float64    ,    1.0       ,               1             ,               1.57075       )
-------------------------------------------------------------------------------------------------------------------
```

ακρίβεια στο Float16: **precision(x1)** = 11
ακρίβεια στο Float32: **precision(x2)** = 24
ακρίβεια στο Float64: **precision(x3)** = 53

precision: επιστροφή των t ψηφίων της mantissa

Τυπολόγιο Αριθμών Κινητής Υποδιαστολής

Το σύνολο των αριθμών x χωρίζεται σε διαστήματα της μορφής $[2^e, 2^{e+1})$



1) Για κάθε διάστημα $[2^e, 2^{e+1})$:

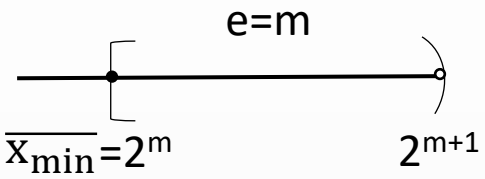
- Εύρος διαστήματος=$2^{e+1} - 2^e = 2^e$ **(1)** (κάθε διάστημα είναι διπλάσιο από το προηγούμενο)
- Πλήθος των $\bar{x}$ στο διάστημα= $2^d = 2^{t-1}$ **(2)** (το πλήθος σε κάθε διάστημα είναι σταθερό)
- Απόσταση μεταξύ 2 διαδοχικών αριθμών $\bar{x}$ στο ίδιο διάστημα= $\frac{2^e}{2^d} = 2^{e-d}$ **(3)**

2) Έψιλον της μηχανής $\epsilon_{mach} = 2^{-d}$ **(4)**
Απόσταση των στοιχείων στο $[2^0, 2^{0+1}) = [1,2)$ (e=0)
Απόσταση του 1 με τον επόμενο διαδοχικό του

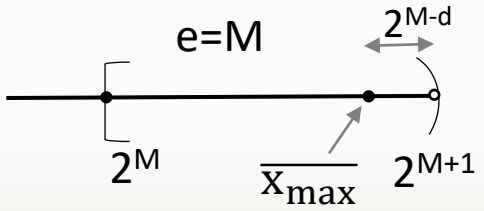
Τυπολόγιο Αριθμών Κινητής Υποδιαστολής

3) Όρια εκθέτη : $m, M: m \le e \le M$

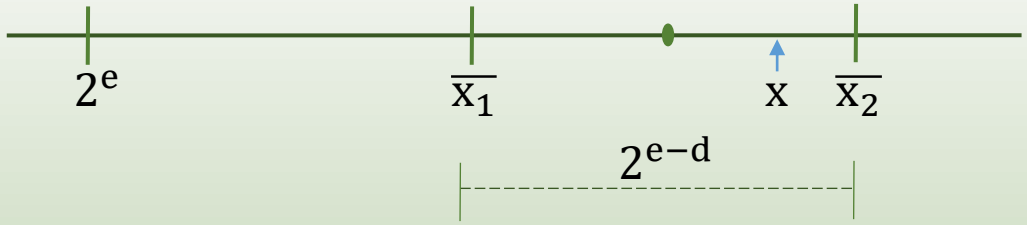
- Ελάχιστος θετικός αριθμός : $\overline{x_{min}} = 2^m$ **(5)**



- Μέγιστος θετικός αριθμός : $\overline{x_{max}} = 2^{M+1} - 2^{M-d} = 2^M - 2^{M-d} + 2^M$ **(6)**
  (αφαίρεση απόστασης 2^{M-d} από πάνω όριο)



4) Στρογγύλευση του πραγματικού αριθμού x στον $\bar{x}$
όπου ο $\bar{x}$ είναι ο πλησιέστερος αριθμός ως προς x



Στο διάστημα $[2^e, 2^{e+1})$

- απόσταση μεταξύ x και $\bar{x}$: $|\bar{x} - x| \le \frac{2^{e-d}}{2}$ **(7)**
- $\bar{x} \ge 2^e$ **(8)**

Σχετικό σφάλμα: $\frac{|\bar{x}-x|}{|x|} \le \frac{2^{e-d}}{2 \cdot 2^e} = \frac{2^{-d}}{2} = \frac{\epsilon_{mach}}{2} = 2^{-t}$ **(9)**

μοναδιαίο σφάλμα στρογγύλευσης

Απόσταση μεταξύ 2 διαδοχικών αριθμών, έψιλον μηχανής (Julia) και διαστήματα

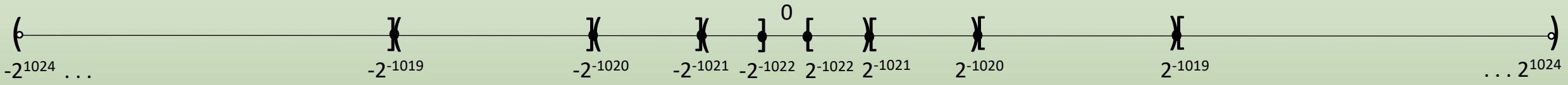
Απόσταση μεταξύ 2 διαδοχικών αριθμών κινητής υποδιαστολής στο [2^e , 2^{e+1})

απόσταση : 2.0 ^ (e – d) = 4.440892098500626e-16 (e=1, d=52) διάστημα [2^1, 2^2)=[2,4)

έψιλον μηχανής : **2.0 ^ (e – d)** = 2.220446049250313e-16 (e=0, d=52) διάστημα [2^0, 2^1)=[1,2)

απόσταση στο διάστημα [2,4) **eps(Float64(3))**= 4.440892098500626e-16
έψιλον της μηχανής **eps(Float64))** = 2.220446049250313e-16
έψιλον της μηχανής **eps()** = 2.220446049250313e-16

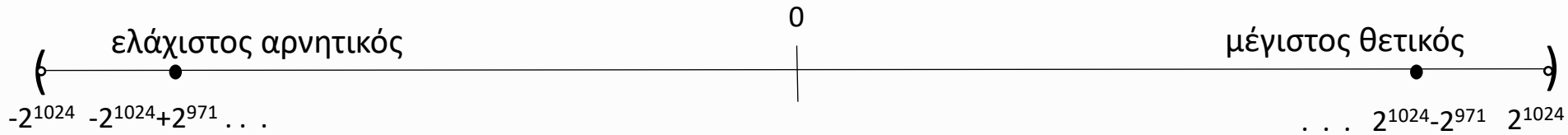
Διαστήματα ευθείας αριθμών μηχανής (Float64)



Πλήθος αριθμών ανά διάστημα (Float64) : Int(2.0 ^(precision(Float64)- 1))=4.503.599.627.370.496

Διάταξη και όρια (Julia)

ελάχιστος θετικός: **2.0 ^ -1022** = 2.2250738585072014e-308
μέγιστος θετικός : **2.0 ^ 1023 - 2.0 ^ (1023 - 52)) + 2.0 ^ 1023** = 1.7976931348623157e308



Άμεσος υπολογισμός ελάχιστου θετικού και μέγιστου θετικού αριθμού
[**floatmin(Float64)**, **floatmax(Float64)**] =
[2.2250738585072014e-308, 1.7976931348623157e308]

---

x = 5.0
επόμενος του 5.0: **x + eps(x)** = 5.000000000000001
επόμενος του 5.0: **nextfloat(Float64(x))** = 5.000000000000001
προηγούμενος του 5.0: **prevfloat(x)** = 4.999999999999999

5.0 : **bitstring(x)** = "0100000000010100000000000000000000000000000000000000000000000000"
επόμενος του 5.0 : bitstring(nextfloat(x))= "0100000000010100000000000000000000000000000000000000000000000001"

όρια εκθέτη: **exponent(floatmin(Float64))** = -1022
exponent(floatmax(Float64)) = 1023

Αριθμοί Κινητής Υποδιαστολής με αυθαίρετη ακρίβεια (BigFloat)

Διερεύνηση προβλημάτων με: - δεδομένα που είναι εκτός των ορίων των Float64 αριθμών
- απαιτήσεις μέγιστης ακρίβειας

- Αριθμοί με αυθαίρετη ακρίβεια
- Προεπιλεγμένη ακρίβεια :256 bits
- Ελάχιστη ακρίβεια :2 bits
- Μέγιστη ακρίβεια :$2^{31} - 1 = 2{,}147{,}483{,}647$ bits
- Απαιτούμενα bits του εκθέτη :31 bits

ty=BigFloat
m = exponent(floatmin(ty)) = -1,073,741,824
M = exponent(floatmax(ty)) = 1,073,741,822
Int(round(log(2, -m + M + 1))) = 31

Εντολή για αυθαίρετη ακρίβεια: **setprecision(BigFloat, t)**
Δεν ισχύει η εντολή **bitstring**

Μετατροπή και ακρίβεια με BigFloat

pistring =
“3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664 7"

---

aF32 = Float32(pi) = 3.1415927f0
aF64 = Float64(pi) = 3.14159265358979

---

aBF = **BigFloat(pi)**= 3.14159265358979323846264338327950288419716939937510582097494459230781640628620

---

setprecision(BigFloat, 300) = 300

aBF300 = **aBF *BigFloat(1.0)** =3.14159265358979323846264338327950288419716939937510582097494459230781640628619802945362503 18

aBF300=**BigFloat(pi)** = 3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482 48

Μετατροπή και ακρίβεια με BigFloat

```
typeof(aF32)        = Float32
typeof(aF64)        = Float64
typeof(aBF)         = BigFloat
typeof(aBF300)      = BigFloat
precision(aBF)      = 256
precision(aBF300) = 300

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
exponent(aF32)      = 1
exponent(aF64)      = 1
exponent(aBF)       = 1
exponent(aBF300) = 1
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
significand(aF32) =    1.5707964f0
significand(aF64) =    1.5707963267948966

significand(aBF) =     1.570796326794896619231321691639751442098584699687552910487472296153908203143099

significand(aBF300) =1.570796326794896619231321691639751442098584699687552910487472296153908203143104993140174124
```

Σωστή και Λανθασμένη χρήση με BigFloat

a1 = 1.2
typeof(a1) = Float64
a2 = **BigFloat(a1)** = 1.19999999999999**55591079014993738383054733276367187 5**
typeof(a2) = BigFloat

---

b1 = 2.4
typeof(b1) = Float64
b2 = **BigFloat(b1)** = 2.39999999999999**9111821580299874767661094665527343 75**
typeof(b2) = BigFloat

---

c2 = a2 + b2 = 3.59999999999999**8667732370449812151491641998291015625**

---

a22 = **BigFloat("3.2")** = 1.200000000000000000000000000000000000000000000000000000000000000000000000000000000000007
a22 = **BigFloat(string(a1))** = 1.200000000000000000000000000000000000000000000000000000000000000000000000000000000000007
b22 = **BigFloat(string(b1))** = 2.400000000000000000000000000000000000000000000000000000000000000000000000000000000000014
c22 = a22 + b22 = 3.600000000000000000000000000000000000000000000000000000000000000000000000000000000000021

Σωστή και λανθασμένη χρήση με BigFloat

BigFloat(1.0/3.0) =0.33333333333333314829616256247390992939472198486328125

BigFloat(1) / 3 =0.3333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333348

1/ BigFloat(3) =0.3333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333348

BigFloat(1) / BigFloat(3) =0.3333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333348

BigFloat(1.1) / BigFloat(3.3) =0.33333333333333337819082927778410505801298818429958983874622989760710256222576 15

BigFloat(1.1) / BigFloat("3.3") =0.33333333333333336024783090000379491936076771129261363636363636363636363636383

BigFloat("1.1") / BigFloat(3.3) =0.33333333333333335127633171111364202320519527371983593549849195904284102489030 63

BigFloat("1.1") / BigFloat("3.3")=0.3333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333348

Σωστή και λανθασμένη χρήση με BigFloat

a = sqrt(BigFloat(3.2)) = 1.78885438199983180677802199993048024085625692827989141450637756927645314807830 3

a ^ 2 = 3.20000000000000017763568394002504646778106689453125

a = sqrt(BigFloat("3.2")) = 1.78885438199983175712733893498502098835249468768 9 220579416717796328416740510238

a ^ 2 = 3.19999999999999999999999999999999999999999999999999999999999999999999999999999972

BigFloat(e)= 2.71828182845904523536028747135266249775724709369995957496696762772407663035355 5
BigFloat(π) = 3.14159265358979323846264338327950288419716939937510582097494459230781640628619 8

---

r=rand(BigFloat,3)
0.60120918402873597787980652241245749985816225236285316385936922838252828986223 63
0.14537880991397423193227343148018634761508024508814713406641561328973334120794 47
0.78275093771919197551416278478482833660071361701246498501887319036811045199637 4

BigFloat.(randn(3))
0.93198522558347551836277489201165735721588134765625
-0.44519560479357045412740490064607001841068267822265625
-1.24347299446289238922247477603377774357795715332031 25

Υπολογισμός του επόμενου του μέγιστου θετικού

m = 2 ^ 1024.0 = **Inf**
m1 = floatmax(Float64) = 1.7976931348623157e308 **Εκτός ορίων σε Float64**
m2 = eps(Float64(m1)) = 1.99584030953472e292
m1 + m2 = **Inf**

---

setprecision(BigFloat, 53) = 53
m3 = BigFloat(0) = 0.0
nextmax1 = m1 + m2 + m3 = **Inf**
nextmax1 = m1 + m3 + m2 = 1.7976931348623159e+308 **1η λύση με Bigfloat**
nextmax1 = m3 + m1 + m2 = 1.7976931348623159e+308
typeof(nextmax1) = BigFloat

---

nextmax2 = 2 ^ BigFloat(1024) = 1.7976931348623159e+308 **2η λύση με BigFloat**

Έλεγχος

2 ^ BigInt(1024) = 179769313486231590772930519078902473361797697894230657273430081157732675805500963132708477322
407536021120113879871393357658789768814416622492847430639474124377767893424865485276302219601246094119453082952
085005768838150682342462881473913110540827237163350510684586298239947245938479716304835356329624224137216

---

q1 = 2 ^ 1023.0 = 8.98846567431158e307
mantissa του 2 ^ 1023.0: q2 = q1 / 10 ^ 307.0 = 8.98846567431158 **Λύση με Float64**
q3 = 2.q2 = 17.97693134862316
mantissa του 2 ^ 1024.0 : nextmaxmant64 = q3 / 10 = 1.7976931348623162

-