# Lesson 12 – Loops

* What are Loops?
* Do While
* For – Next
* Algorithms with loops

What students should know

**4h**

Οι επαναληπτικές δομές είναι από τους πιο βασικές λειτουργίες μια γλώσσας προγραμματισμού. Μία επανάληψη (βρόχος) σε ένα πρόγραμμα υπολογιστή είναι μια οδηγία που επαναλαμβάνεται μέχρι να επιτευχθεί μια καθορισμένη συνθήκη. Σε μια δομή επανάληψης, ο βρόχος θέτει μια ερώτηση. Εάν η απάντηση απαιτεί ενέργεια, εκτελείται. Το ίδιο ερώτημα τίθεται ξανά και ξανά μέχρι να μην απαιτηθεί περαιτέρω δράση. Κάθε φορά που τίθεται η ερώτηση ονομάζεται επανάληψη.

Ένας προγραμματιστής υπολογιστή που πρέπει να χρησιμοποιήσει τις ίδιες γραμμές κώδικα πολλές φορές σε ένα πρόγραμμα μπορεί να χρησιμοποιήσει ένα βρόχο για να εξοικονομήσει χρόνο, χώρο, ευκολία προγραμματισμού.

**Remember**

Οι επαναλήψεις πρέπει να τελειώνουν κάποια στιγμή διαφορετικά πρόκειται για προγραμματιστικό λάθος. Μια επανάληψη που δεν τελειώνει ποτέ λέγεται και ατέρμων βρόχος. Στην περίπτωση αυτή ο υπολογιστής εγκλωβίζεται σε μία αέναη επανάληψη των ίδιων λειτουργιών χωρίς «συναίσθηση» της ματαιότητας!



Στην B4X υπάρχουν 4 επαναληπτικές εντολές. Στο παρόν μάθημα θα παρουσιαστούν οι εντολές **for**, **Do** **while,** **Do Until** και σε επόμενο μάθημα η τέταρτη εντολή **for each**.

## Η εντολή Do While

Υποθέστε ότι θα θέλατε να εμφανίσετε στην οθόνη με την εντολή log τους αριθμούς από το 1 έως το 5 τον ένα κάτω από τον άλλο. Ο κώδικας που θα έπρεπε να γράψετε σε B4J θα ήταν ο παρακάτω:

Log(1)

Log(2)

Log(3)

Log(4)

Log(5)

Αντιλαμβάνεστε πως αν έπρεπε να συνεχίσετε να τυπώνετε αριθμούς τότε θα έπρεπε να συνεχίσετε να γράφετε τις εντολές Log για όσους αριθμούς χρειάζεστε. Το γεγονός αυτό δυσκολεύει κατά πολύ τη συγγραφή του κώδικα και τον γεμίζει επαναλαμβανόμενες εντολές ενώ θα μπορούσατε να χρησιμοποιήσετε μια εντολή επανάληψης για τον σκοπό αυτό.

Η εντολή Do While δηλώνεται ως εξής:

Picture Εντολή Do While

**Do While** *συνθήκη*

Εντολές

…

**Loop**

**Όσο η**  *συνθήκη* είναι αληθής

Εκτέλεσε τις Εντολές

…

**Επέστρεψε στη συνθήκη**

Στο παράδειγμα με τους πέντε αριθμούς και με χρήση Do While ο κώδικας γράφεται:

**Private** i as Int = 1

**Do Whil**e *i <= 5*

**Log**(i)

i = i + 1

**Loop**

Η **μεταβλητή i** δημιουργήθηκε προκειμένου να μετράει πόσες φορές θα εκτελεστεί η επανάληψη. Για αυτό ονομάζεται και **μετρητής** της επανάληψης.

Επίσης, η Συνθήκη εξετάζει αν το περιεχόμενο του μετρητή ξεπέρασε το όριο που έχει θέσει ο προγραμματιστής δηλαδή την τιμή 5 και εφόσον συμβεί **σταματά** η επανάληψη.

Πριν από το τέλος επανάληψης με την εντολή Loop ο μετρητής επαναλήψεων αυξάνεται κατά +1. Η τιμή αυτή ονομάζεται και **Βήμα** της επανάληψης

Το βήμα μπορεί να είναι θετικό ή αρνητικό αλλά ποτέ δεν μπορεί να είναι 0 γιατί τότε η επανάληψη θα εκτελούνταν για πάντα.

Παραδείγματα της εντολής Do While

**Private** i as **Int** = 100

**Do Whil**e *i >= 1*

**Log**(i)

i = i - 1

**Loop**

**Example 1**

Να εμφανιστούν όλοι οι αριθμοί από το 100 έως 1



Στο παράδειγμα ο μετρητής επαναλήψεων i ξεκινά από μία μεγάλη τιμή (την 100) και καταλήγει σε μία μικρή (την 0) πριν ολοκληρωθεί η επανάληψη. Προσοχή σε αυτήν την περίπτωση το **βήμα** της επανάληψης είναι **αρνητικό** (-1).

**Private** i as **Int** = 1

**Do Whil**e *i <= 100*

**If** i mod 2 = 0 **then**

**Log**(i)

**End if**

i = i + 1

**Loop**

**Private** i as **Int** = 2

**Do Whil**e *I <= 100*

**Log**(i)

i = i + 2

**Loop**

**Example 2**

Να εμφανιστούν όλοι οι άρτιοι αριθμοί από το 1 έως το 100



Εδώ παρουσιάζονται δύο λύσεις η πρώτη χρησιμοποιεί μία εντολή if μέσα στην επανάληψη προκειμένου να ελέγξει αν ο αριθμός i είναι άρτιος και μόνο τότε εκτελεί την εντολή Log. Στον δεύτερο αλγόριθμο έχει αλλάξει η **αρχική τιμή** του i σε 2 και το **βήμα** αυξάνεται κατά 2 γεγονός που δημιουργεί πιο γρήγορο αλγόριθμο στην εκτέλεση.

**Remember**

Όταν ο μετρητής ξεκινά από τιμή μικρότερη από την τελική τιμή τότε πρέπει πάντα το **βήμα** να είναι **θετικό**.

Όταν ο μετρητής ξεκινά από τιμή μεγαλύτερη από την τελική τιμή τότε πρέπει πάντα το **βήμα** να είναι **αρνητικό**



**Example 3 – Αλγόριθμος Άθροισμα**

Να γίνει πρόγραμμα που για δέκα αριθμούς που εισάγονται στο πρόγραμμα να υπολογίζεται το άθροισμά τους. Οι αριθμοί αυτοί θεωρείστε ότι κυμαίνονται μεταξύ -100 και 100



Για τις ανάγκες του παραδείγματος θα χρησιμοποιηθεί η συνάρτηση. Η χρήση της γίνεται όπως παρακάτω:

**Private** i **as** **Int** = 1

**Private** A **as** **Int**

**Private** intSum **as** **Int** = 0

**Do Whil**e *i <= 10*

A = **Rnd**(-100, 100)

intSum = intSum + Α

i = i + 1

**Loop**

**Remember**

Η εντολή **Rnd(ΑρχικήΤιμή, ΤελικήΤιμή)** επιστρέφειέναν αριθμό μεταξύ της αρχικής και τελικής τιμής.

π.χ. Α = Rnd(1, 10) επιστρέφει ένα αριθμό μεταξύ 1 και 10 στη μεταβλητή Α



**Private** i **as** **Int** = 1

**Private** A **as** **Int**

**Private** intCounter **as** **Int** = 0

**Do Whil**e *i <= 10*

A = **Rnd**(-100, 100)

**If** A < 0 **then**

intCounter = intCounter + 1

**End If**

i = i + 1

**Loop**

**Example 4 – Αλγόριθμος Πλήθος**

Να γίνει πρόγραμμα που για δέκα αριθμούς που εισάγονται στο πρόγραμμα να υπολογίζεται το πλήθος των αρνητικών. Οι αριθμοί αυτοί θεωρείστε ότι κυμαίνονται μεταξύ -1000 και 1000



**Example 5 – Αλγόριθμος Μέγιστο - Ελάχιστο**

Να γίνει πρόγραμμα που για δέκα αριθμούς που εισάγονται στο πρόγραμμα να υπολογίζεται ο μεγαλύτερος και ο μικρότερος αριθμός. Οι αριθμοί αυτοί θεωρείστε ότι κυμαίνονται μεταξύ -1000 και 1000



**Private** i **as** **Int** = 1

**Private** A **as** **Int**

**Private** intMax, intMin **as** **Int**

A = **Rnd**(-100, 100)

intMax = A

intMin = A

**Do Whil**e *i <= 9*

A = **Rnd**(-100, 100)

**If** intMax < A **then**

intMax = A

**End If**

**If** intMin > A **then**

intMax = A

**End If**

i = i + 1

**Loop**

1. Πρώτα διαβάζεται ένα αριθμός έξω από την επανάληψη.
2. Ορίζεται ως αρχική τιμή στη μεταβλητή intMax και intMin ο πρώτος αριθμός αφού δεν υπάρχει άλλος να συγκριθεί.
3. Για κάθε νέο αριθμό ελέγχεται αν είναι μικρότερος από το intMin και γίνεται αντικατάσταση ή αν ο νέος Α είναι μεγαλύτερος από τον intMax οπότε και αντικαθίσταται με τον νέο Α.

## Επαναλήψεις που δεν γνωρίζουμε το πλήθος τους

Συχνά στον προγραμματισμό δεν γνωρίζουμε από την αρχή το πλήθος των επαναλήψεων. Αυτό συνήθως συμβαίνει όταν η επανάληψη εξαρτάται από την τιμή που δέχεται ως δεδομένο ή από τιμές που υπολογίζονται κατά την διάρκεια της επανάληψης.

**Example 6 – Μη συγκεκριμένο πλήθος επαναλήψεων**

Να γίνει πρόγραμμα που να διαβάζει συνεχώς αριθμούς και να υπολογίζει τον ΜΟ των αρτίων. Το πρόγραμμα σταματά όταν εισαχθεί αριθμός μικρότερος από το 0.



1. Η συνθήκη στην επανάληψη ελέγχει αν το σύνολο δεν έχει φτάσει στο όριο 200.
2. Μέσα στην επανάληψη διαβάζεται ένας αριθμός αθροίζεται και στην συνέχεια ελέγχεται ξανά η τιμή της συνθήκης από τη Do While.

**Private** A **as** **Int**

**Private** intSum **as** **Int = 0**

**Do Whil**e *intSum <= 200*

*A = Rnd(-100, 100)*

intSum = intSum + A

**Loop**

**Example 7 – Μη συγκεκριμένο πλήθος επαναλήψεων**

Να γίνει πρόγραμμα που να διαβάζει συνεχώς αριθμούς και να υπολογίζει το άθροισμα τους. Το πρόγραμμα να σταματά όταν το άθροισμα ξεπεράσει την τιμή 200.



1. Πρώτα διαβάζεται ένα αριθμός έξω από την επανάληψη.
2. Η συνθήκη της Do While ελέγχει αν ο αριθμός Α είναι εντός των ορίων.
3. Πριν το τέλος της επανάληψης διαβάζεται νέος αριθμός.

**Private** A **as** **Int**

**Private** intCount **as** **Int = 0**

A = **Rnd**(-100, 100)

**Do Whil**e *A > 0*

**If** A mod 2 = 0 **then**

intCount = intCount + 1

**End If**

A = Rnd(-100, 100)

**Loop**

## Η εντολή Do Until

Η λειτουργία της Do Until είναι παρόμοια με της Do While. Η μόνη διαφορά είναι η συνθήκη η οποία ελέγχει πότε θα τελειώσει μια επανάληψη σε αντίθεση με τη Do While που ελέγχει για πόσο θα λειτουργεί. Και στις δύο περιπτώσεις ο έλεγχος πραγματοποιείται στην αρχή οπότε σε καμία περίπτωση δεν εκτελείται ο κώδικας αν ισχύσει η συνθήκη της Until.

**Remember**

Η συνθήκη της Until είναι η άρνηση της While.



**Example 1**

Να εμφανιστούν όλοι οι αριθμοί από το 100 έως 1



**Private** i as **Int** = 100

**Do Until** *i < 1*

**Log**(i)

i = i - 1

**Loop**

**Private** i as **Int** = 1

**Do Until** *i >100*

**If** i mod 2 = 0 **then**

**Log**(i)

**End if**

i = i + 1

**Loop**

**Private** i as **Int** = 2

**Do Until** i *> 100*

**Log**(i)

i = i + 2

**Loop**

**Example 2**

Να εμφανιστούν όλοι οι άρτιοι αριθμοί από το 1 έως το 100



**Private** i **as** **Int** = 1

**Private** A **as** **Int**

**Private** intSum **as** **Int** = 0

**Do Until** *i > 10*

A = **Rnd**(-100, 100)

intSum = intSum + Α

i = i + 1

**Loop**

**Example 3 – Αλγόριθμος Άθροισμα**

Να γίνει πρόγραμμα που για δέκα αριθμούς που εισάγονται στο πρόγραμμα να υπολογίζεται το άθροισμά τους. Οι αριθμοί αυτοί θεωρείστε ότι κυμαίνονται μεταξύ -100 και 100



**Example 4 – Αλγόριθμος Πλήθος**

Να γίνει πρόγραμμα που για δέκα αριθμούς που εισάγονται στο πρόγραμμα να υπολογίζεται το πλήθος των αρνητικών. Οι αριθμοί αυτοί θεωρείστε ότι κυμαίνονται μεταξύ -1000 και 1000



**Private** i **as** **Int** = 1

**Private** A **as** **Int**

**Private** intCounter **as** **Int** = 0

**Do Until** *i > 10*

A = **Rnd**(-100, 100)

**If** A < 0 **then**

intCounter = intCounter + 1

**End If**

i = i + 1

**Loop**

**Private** i **as** **Int** = 1

**Private** A **as** **Int**

**Private** intMax, intMin **as** **Int**

A = **Rnd**(-100, 100)

intMax = A

intMin = A

**Do Until** *i > 9*

A = **Rnd**(-100, 100)

**If** intMax < A **then**

intMax = A

**End If**

**If** intMin > A **then**

intMax = A

**End If**

i = i + 1

**Loop**

**Example 5 – Αλγόριθμος Μέγιστο - Ελάχιστο**

Να γίνει πρόγραμμα που για δέκα αριθμούς που εισάγονται στο πρόγραμμα να υπολογίζεται ο μεγαλύτερος και ο μικρότερος αριθμός. Οι αριθμοί αυτοί θεωρείστε ότι κυμαίνονται μεταξύ -1000 και 1000



**Private** A **as** **Int**

**Private** intCount **as** **Int = 0**

A = **Rnd**(-100, 100)

**Do Until** *A < 0*

**If** A mod 2 = 0 **then**

intCount = intCount + 1

**End If**

A = Rnd(-100, 100)

**Loop**

**Example 6 – Μη συγκεκριμένο πλήθος επαναλήψεων**

Να γίνει πρόγραμμα που να διαβάζει συνεχώς αριθμούς και να υπολογίζει τον ΜΟ των αρτίων. Το πρόγραμμα σταματά όταν εισαχθεί αριθμός μικρότερος από το 0.



**Private** A **as** **Int**

**Private** intSum **as** **Int = 0**

**Do Until** *intSum > 200*

*A = Rnd(-100, 100)*

intSum = intSum + A

**Loop**

**Example 7 – Μη συγκεκριμένο πλήθος επαναλήψεων**

Να γίνει πρόγραμμα που να διαβάζει συνεχώς αριθμούς και να υπολογίζει το άθροισμα τους. Το πρόγραμμα να σταματά όταν το άθροισμα ξεπεράσει την τιμή 200.



## Εντολή for

Η εντολή for είναι ίσως η πιο απλή εντολή επανάληψης. Χρησιμοποιείται για καθορισμένο αριθμό επαναλήψεων και απαιτεί την ύπαρξη μιας μεταβλητής μετρητή. Δηλώνεται ως εξής:

**for** i = n1 **to** n2 **Step** n3

Εντολές

…

**Next**

Όπου:

i η μεταβλητή του μετρητή

n1 η αρχική τιμή του μετρητή

n2 η τελική τιμή

n3 το βήμα της επανάληψης

* Στην αρχή της επανάληψης η μεταβλητή ι δέχεται την τιμή n1.
* Εκτελεί τις εντολές που βρίσκονται μέσα στην επανάληψη
* Στο τέλος της επανάληψης αυξάνεται ή μειώνεται η τιμή του I κατά n3
* Ελέγχεται αν το i έχει ξεπεράσει την τελική τιμή n2
  + Αν το βήμα είναι θετικό και το i είναι μικρότερο ή ίσο της τελικής τιμής τότε η επανάληψη εκτελείται ξανά
  + Αν το βήμα είναι αρνητικό και το i είναι μεγαλύτερο ή ίσο της τελικής τιμής τότε η επανάληψη εκτελείται ξανά.

Σε σχέση με την εντολή Do While και Do Until η for:

* Δεν χρειάζεται αρχικοποίηση του μετρητή
* Δεν χρειάζεται να ορίσετε την πράξη αλλαγής του βήματος
* Αφορά αποκλειστικά μετρήσιμες επαναλήψεις (Παρόλα αυτά μπορείτε να χρησιμοποιήσετε τη εντολή exit προκειμένου να βγείτε από αυτή όποια στιγμή θέλετε).

### Παραδείγματα for

**Private** i **as** **Int**

**For** i = 100 **to** 1 **step** -1

**Log**(i)

**Next**

**Example 1**

Να εμφανιστούν όλοι οι αριθμοί από το 100 έως 1



**Private** i **as** **Int**

**For** i = 1 **to** 100

**If i mod 2 = 0 then**

**Log**(i)

**End If**

**Next**

**Example 2**

Να εμφανιστούν όλοι οι άρτιοι αριθμοί από το 1 έως το 100



## Ασκήσεις

1. Να γραφτεί πρόγραμμα που να δέχεται έναν ακέραιο αριθμό και να υπολογίζει το άθροισμα των περιττών από το 1 μέχρι τον αριθμό που δέχτηκε.
2. Να δοθεί ένας αλγόριθμος ο οποίος να διαβάζει έναν ακέραιο Κ μεταξύ 1 και 200 και να υπολογίζει το άθροισμα 1+2+3+...+Κ.
3. Να γραφτεί πρόγραμμα που ο χρήστης θα δίνει αριθμούς και θα υπολογίζεται ο μέσος όρος των παραπάνω αριθμών. Το πρόγραμμα τερματίζει με την εισαγωγή του μηδενός.
4. Ένας καταναλωτής θέλει να αγοράσει παιχνίδια για δώρα σε 10 παιδιά. Θέλει επίσης η συνολική τιμή τους να μην ξεπερνά τα 200€. Να γίνει πρόγραμμα που:
   1. Να εισάγεται το κόστος του παιχνιδιού (εντολή Rnd(10, 50) )
   2. Να υπολογίζει το συνολικό κόστος των παιχνιδιών που αγοράστηκαν μέχρι τώρα,
   3. Να ελέγχει αν έχουν εξαντληθεί τα χρήματα
   4. Να εμφανιστεί τελικά το συνολικό κόστος των παιχνιδιών και ο αριθμός τους.
5. A leap year (also known as an intercalary year or bissextile year) is a calendar year that contains an additional day (or, in the case of a lunisolar calendar, a month) added to keep the calendar year synchronized with the astronomical year or seasonal year. A year is a leap year when:

if (year is not divisible by 4) then (it is a common year)

else if (year is not divisible by 100) then (it is a leap year)

else if (year is not divisible by 400) then (it is a common year)

else (it is a leap year)

Make a program that shows the leap years between 1900 and 2100

(<https://en.wikipedia.org/wiki/Leap_year>)

1. Κατασκευάστε πρόγραμμα με τη βοήθεια της χελώνας που να δημιουργεί κανονικά πολύγωνα με πλήθος γωνιών που θα εισάγει ο χρήστης σε κατάλληλο πεδίο. Το πρόγραμμα θα περιλαμβάνει κουμπί εκκίνησης της σχεδίασης και κουμπί καθαρισμού της οθόνης που όταν πατηθεί να καθαρίζει η οθόνη και η χελώνα θα τοποθετείται στο κέντρο της.
2. Σχεδιάστε με την χελώνα τα παρακάτω σχήματα:







