Lesson 12 – Loops

⏱ 4h

What students should know
• What are Loops? • Do While • Do Until • For – Next • Algorithms with loops

Repetitive structures are one of the most basic functions of a programming language. A loop in a computer program is a directive that repeats until a specified condition is reached. In a repeat structure, the loop poses a question. If the answer requires action, it runs. The same question is asked over and over again until no further action is required. Every time the question is asked it is called repetition.

A computer programmer who has to use the same lines of code multiple times in a program can use a loop to save time, space, ease of programming.

Remember

Repeats have to end at some point or it's a programming error. A repetition that never ends is called an endless loop. In this case the computer is trapped in a perpetual repetition of the same functions without "awareness" of vanity!

In B4X there are four repeat **commands**: For, Do While, Do Until, for each.

The Do While command

Suppose you would like to display on the screen with the command log numbers from 1 to 5 below each other. The code you should write to B4J would be as follows:

```
Log(1)
Log(2)
Log(3)
Log(4)
Log(5)
```

You realize that if you had to keep printing numbers then you would have to continue writing log commands for as many numbers as you need.

The command Do While is stated as follows:

Do While *Condition* Commands ... **Loop**	**As long as** the condition is true Execute the Commands ... **Go back to condition.**

Picture 1 Command Do While

In the example with the five numbers and using Do While the code is written:

```
Private i as Int = 1

Do While i <= 5
   Log(i)
   i = i + 1
Loop
```

Variable **i** counts how many times the loop will run.

The Condition also examines whether the contents of the counter exceeded the limit set by the developer, i.e. 5, and if this happens, the repetition stops.

Before the loop iteration end, the repeat counter is increased by +1. This value is also called Step.

The step can be positive or negative, but it can never be 0 because then the repetition would run forever.

Remember

When the counter starts at a value less than the final value then the **step must** be **positive.**

When the counter starts from a value greater than the final value then the **step must** be **negative.**

Examples of the command Do While.

Example 1

Show all integers from 100 to 1

```
Private i as  Int  = 100

Do While i >= 1
   Log(i)
   i = i -  1
Loop
```

In the example the repeat counter it starts at a large value (100) and ends at a small (0) before the repetition is complete. Attention in this case to the **Step** is **Negative** (-1).

Example 2

Show all even numbers from 1 to 100

```
Private i as  Int  = 1

Do While  i  <= 100
   If I  mod 2 = 0
then
     Log(i)
   End if
   i = i +  1
Loop
```

```
Private i as  Int  =  2

Do While  I <= 100
   Log(i)
   i = i + 2
Loop
```

Here are two solutions presented the first uses a command if in loop to check whether the number is even and only then executes the command Log. In the second algorithm has changed the **starting value** of the i in 2 and the **Step** increases by 2 which creates a faster algorithm to run.

Example 3 – Sum Algorithm

Make a program that for ten numbers entered, the program calculates their sum. These numbers are considered to range between -100 and 100

The function will be used for the purposes of the example. It is used as follows:

Remember

The **Rnd command (First Value, Last Value)** returns a number between the First and Last value.

e.g. A = Rnd(1, 10) returns a number between 1 and 10 in variable A

```
Private I  As  Int  =  1
Private A  as Int
Private intSum  as Int  = 0

Do While  i  <= 10
   A = Rnd(-100, 100)
   intSum =  intSum  +  A

   i = i +  1
Loop
```

Example 4 – Algorithm Count.

Make a program that for ten numbers entered, the program end calculates the number of negatives. These numbers are considered to range between -1000 and 1000

```
Private I  As  Int  =  1
Private A  as Int
Private intCounter  as Int  = 0

Do While  i  <= 10
   A = Rnd(-100, 100)
   If A < 0  then
     intCounter = intCounter + 1
   End If
   i = i +  1
Loop
```

Example 5 - Maximum Algorithm - Minimum

Make a program that for ten numbers entered the program counts the largest and smallest numbers. These numbers are considered to range between -1000 and 1000

```
Private I  As  Int  =  1
Private A  as Int
Private intMax,  intMin  as Int

A = Rnd(-100, 100)
intMax = A
intMin = A
Do While  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. First read a number out-side the while.
2. Set as the starting value in the intMax end intMin variable the first number since there is no one else to compare.
3. For each new number it is checked
   if it is less than intMin then intMin replaced by A if the new A is greater than intMax the intMax is replaced with the new A.

Loops with unknow repeats.

Often in programming we do not know from the beginning the number of repetitions. This usually happens when the repeat depends on the value taken from user or on values calculated during the repeat.

Example 6 – Non-specific number of repetitions

Create a program that constantly reads numbers and calculates the Averages of the even. The program stops when a number less than 0 is entered.

```
Private A  as Int
Private intCount  as Int  = 0

A = Rnd(-100, 100)                1
Do While A  >  0
                                  2
   If A mod 2 = 0  then
     intCount =  intCount  + 1
   End If

   A = Rnd(-100, 100)             3
Loop
```

1. First read a number outside the replay.
2. The DoWhile condition checks whether the number A is within the limits.
3. A new number is read before the end of the iteration.

Example 7 – Non-specific number of repetitions

Create a program that reads numbers and calculates their sum. The program stops when the sum exceeds 200.

```
Private A  as Int
Private intSum  as Int  = 0

                                  1
Do While intSum <= 200
   A = Rnd(-100, 100)
   intSum = intSum + A            2
Loop
```

1. The condition checks whether the total has not reached the limit of 200.
2. A number is read and then program calculates the sum. Condition checked again by Do While.

The Do Until command.

The operation of the Do Until is like the Do While. The only difference is the condition which controls when a repetition will end as opposed to the Do While which controls how long it will work. In both cases the check shall be carried out at the beginning, in which case the code shall not be executed if the Condition is False.

Remember

The condition of Do Until is the denial of Do While.

Example 1

Show all numbers from 100 to 1

```
Private i as  Int  = 100

Do Until  i  <  1
  Log(i)
  i = i -  1
Loop
```

Example 2

Show all even numbers from 1 to 100

```
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 3 – Sum Algorithm

Make a program that for ten numbers entered, the program calculates their sum. These numbers are considered to range between -100 and 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  +  A

  i = i +  1
Loop
```

Example 4 – Algorithm Count

Make a program that for ten numbers entered, the program calculates the number of negatives. These numbers are considered to range between -1000 and 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
```

Example 5 - Maximum Algorithm - Minimum

Make a program that for ten numbers entered, the program counts the largest and smallest numbers. These numbers are considered to range between -1000 and 1000.

```
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 6 – Non-specific number of repetitions

Create a program that constantly reads numbers and calculates the Averages of the even. The program stops when a number less than 0 is entered.

```
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 7 – Non-specific number of repetitions

Create a program that reads numbers and calculates their sum. The program stops when the sum exceeds 200.

```
Private A  as Int
Private intSum  as Int  = 0

Do Until  intSum  >  200
  A = Rnd(-100, 100)
  intSum = intSum + A
Loop
```

For Loop

The for loop is probably the simplest repeat command.

```
for i  = n1  to  n2  Step  n3
        Commands
        ...
Next
```

Where:

i the Counter variable

n1 the initial value of the Counter

n2 the final value

n3 step of repetition

- At the beginning of repetition, variable 'i' accepts n1.
- Executes commands within the repetition.
- At the end of the repetition the value of 'i' increases or decreases by n3
- Check if 'i' has exceeded the final value n2.
    - if the step is positive and 'i' is less than or equal to the final value then the repeat runs again
    - If the step is negative and 'i' is greater than or equal to the final value, then the repeat runs again.

Remember

In relation to **Do While** and **Do Until** the **For** command:

- Do not needs to initialize the counter.
- Do not needs to set the step change operation.
- It is always for measurable repeats (however, you can use the exit command to get out of it at any time).

Examples for

Example 1

Show all numbers from 100 to 1

```
Private I As Int

For i = 100 to 1 step -1
  Log(i)
Next
```

Example 2

Show all even numbers from 1 to 100

```
Private I  As  Int

For i = 1  to  100
  If i  mod 2 = 0 then
    Log(i)
  End If
Next
```

Exercises

1. Write a program that reads an integer K between 1 and 200 and calculates the sum of 1+2+3+...+K.
2. Write a program that the user will give numbers and calculate the average of the above numbers. The program ends by entering zero.
3. A consumer wants to buy toys for gifts to 10 children. It also wants their total price not to exceed 200€. Make a program that:
    a. Enter the cost of the game (random number between 10-50 ,command Rnd(10, 50))
    b. Calculate the total cost of games purchased so far,
    c. Check if the money has run out.
    d. Finally show the total cost of the games and their number.
4. 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)
5. Build a program with the help of the turtle that creates polygons with a number of angles that the user enter in an appropriate field. The program will include a design start button and a screen cleaning button that when pressed cleans the screen and the turtle will placed in center of screen.
6. Draw the following shapes with the turtle: