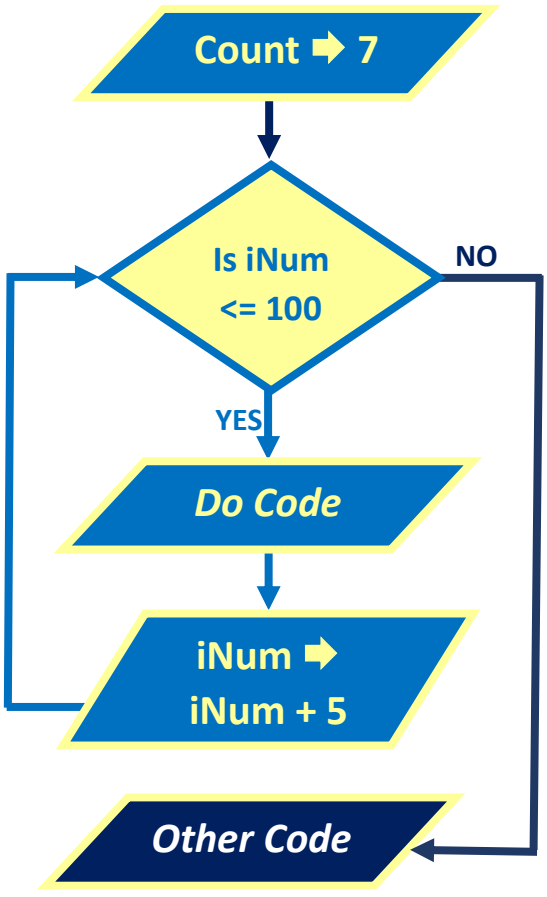
Mr Long Summary ON CONDITIONAL LOOPS

Iteration Programming

- THREE ways the code is executed:
  - Sequential – when each line is executed in order from first line to the last line.
  - Selection – when you select which code will be executed based on a condition.
  - *Iteration – repeatedly execute code based on a condition.*

- Example of Iteration using a flowchart:



RECAP: **For Loops**
for K := 1 **to** 10 **do**
- When you know how many times to execute the loop before it starts.
  - *Sum the first 100 numbers*
  - *Amount in a savings account after 2 years*

Conditional Loops

- When you DON'T know how many times to execute the loop before it starts.
- Reason to stop the loop occurs inside the loop
  - *Sum all the numbers until the sum is greater than 100.*
  - *How long it takes to save money in a savings account until you double your money.*

WHILE Loop Statement

General Structure – WHILE Loop Statement

```
while <condition(s)> do
  begin
    <statement> ;
    ....
    <statement> ;
  end ;
```

NOTE: *Although you don't need the **begin end** if only executing one statement, I suggest always using them so that you don't get confused if you decide to add more statements to the WHILE loop statement later.*

***WARNING:** Do NOT place a semi-colon (;) after the do operator.*

REPEAT Loop Statement

General Structure – REPEAT Loop Statement

```
repeat
    <statement> ;
    ....
    <statement> ;
 until <condition(s)> :
```

NOTE: *No **begin end** is needed.*

WHILE Loop	REPEAT Loop

while *<condition(s)>* **do**
begin
<statement(s)>
end ;
	repeat
	<statement(s)>
	until *<condition(s)>* ;

Pre-condition Loop *Condition is checked at START of loop*	**Post condition Loop** *Condition is checked at END of loop*
0 - ∞ *Loop can run from 0 times to infinity because condition is checked first.*	**1 - ∞** *Loop will run at least once to infinity because condition is only checked at the end.*
Condition(s) *TRUE means continue looping* *FALSE means stop loop*	**Condition(s)** *TRUE means stop loop* *FALSE means continue looping*
Begin..end *needed for multiple statements*	**NO begin..end** *needed because statements are between repeat and until operators.*

Condition(s)

- Condition must either result in a TRUE or a FALSE
  - Same as conditions used in IF statements
  - Use comparison operators { equal (**=**), greater than (**>**), less than (**<**), greater than or equal to (**>=**), less than or equal to (**<=**), not equal (**<>**) }
  - Multiple conditions use ***AND*** or ***OR*** operators
  - Can make use of ***NOT*** operator, Boolean functions and sets.

RECAP: **For Loops**

```
iSum := 0 ;
for K := 1 to 100 do
  begin
    iSum := iSum + K ;
  end ;
```

- Adds ALL numbers from 1 to 100
- For loops have a built-in looping variable (K in example)
  - *Looping variable always has a starting value.*
  - *Looping variable increases by 1*

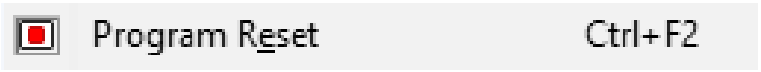
WHILE Loop Example	REPEAT Loop Example

iSum := 0 ;
K := 1 ;
while *iSum < 100* **do**
begin
iSum := iSum + K ;
inc (K) ;
end ;
	iSum := 0 ;
	K := 1 ;
	repeat
	iSum := iSum + K ;
	inc (K) ;
	until *iSum >= 100* ;

Explanation: *Add all the numbers from 1 onwards UNTIL the sum of those numbers is 100 or more.* *Note: The changes of the Sum equalling 100 exactly is highly unlikely so that is why you use 100 or more.*	
NOTE: Need a looping variable but ***While*** and ***Repeat*** loops do not have a built-in looping variable. Therefore, you need to program one into the code: ○ *Looping variable always has a starting value. (**K := 1**)* ○ *Looping variable increases by 1 (**inc(K) ;**)*	
Condition: *iSum < 100* *While iSum isn't 100 or more, keep loop.* *When iSum is 100 or more, stop the loop.*	**Condition: *iSum >= 100*** *Repeat the loop until iSum is 100 or more.* *Note the condition is the OPPOSITE of the while loop.*

Infinite Loop

- Infinite loop is when the condition to stop the loop is never reached and so the loop will continue to loop and never stop.
- To stop an infinite loop in Delphi, click on ***Run*** tab and then ***Program Reset***.



WHILE Loop Example	REPEAT Loop Example

iSum := 0 ;
K := 1 ;
while *iSum* **=** *100* **do**
begin
iSum := iSum + K ;
inc (K) ;
end ;
	iSum := 0 ;
	K := 1 ;
	repeat
	iSum := iSum **– K** ;
	inc (K) ;
	until *iSum >= 100* ;

Note: Condition is *iSum = 100* *The sum of all incremental numbers will never equal 100 exactly. The iSum variable will eventually exceed 100 it will continue to add new values onto iSum and never decrease, therefore never get to 100. This results in an **Infinite Loop**.*	**NOTE: *iSum – K ;*** *iSum variable starts at 0 and is continuously decreasing in the loop. It will never reach the value of 100 or more as it is always decreasing. This results in an **Infinite Loop**.*

ITC or ICT Principle

- When using a ***While*** loop, you must decide on a value(s) or variable(s) that will determine if the loop is executed and when it stops.
- Referred to as the ITC principle:

Initialise	The variable or variables must be assigned default values before the While loop condition is checked.
Test	The variable or variables are tested in the loop. If the condition is TRUE then the code in the loop is executed.
Change	Somewhere in the loop, the variable or variables, that are tested, should change.

Example 1

```
K := 1 ;
iCount := 0 ;
while iCount < 100 do
  begin
    if ( K MOD 7 = 0 ) AND ( K MOD 3 = 0) then
    begin
     memDisplay.lines.add( IntToStr( K ) ) ;
     inc ( iCount ) ;
   end ; //end of if
  inc ( K ) ;
end ; //end of while
```

Explanation:
Displays the first 100 numbers that are divisible by BOTH 7 and 3.

ITC Principle:
Variable that determines when loop must stop: ***iCount***

Initialise: iCount := 0 ;
Test: iCount < 100
Change: **inc(iCount) ;**

- When using a ***Repeat*** loop, because the test occurs at the end (post condition loop), rather use the ICT principle:

Initialise	The variable or variables must be assigned default values before the *Repeat* loop condition is checked.
Change	Somewhere in the loop, the variable or variables, that are tested, should change.
Test	The variable or variables are tested in the loop. If the condition is FALSE then the code in the loop is executed.

Example 2

```
K := 1 ;
iSum := 0 ;
repeat

 if ( K MOD 7 = 0 ) AND ( K MOD 3 = 0) then
  begin
   iSum := iSum + K ;
 end ; //end of if
 inc ( K ) ;

until iSum >= 300 ;
```

Explanation:
Total ALL numbers that are divisible by BOTH 7 and 3 until the total is 300 or more.

ITC Principle:
Variable that determines when loop must stop: ***iSum***

Initialise: iSum := 0 ;
Change: iSum := iSum + K ;
Test: iSum >= 300

Loop Control

Loops are controlled in different ways:

- ***Counter*** controlled loops: looping variable is initialised and its value is changed inside the loop

```
iSum := 0 ;
for iCounter := 1 to 10 do
 begin
   iSum := iSum + iCounter ;
 end ;
```

- ***Sentinel*** controlled loops: variable controlling the loop is tested for a sentinel or signal value.

```
repeat
   //do code
   sAns := Inputbox( '' , 'Execute code again [Y/N]:' , '' ) ;
until sAns = 'N' ;
```

- ***Result*** controlled loops: loop continues until a result has been met.

```
iSum := 1 ;
while iSum <= 100 do
 begin
   iSum := iSum * 2 ;
 end ;
```

Additional Links:

- Youtube video playlist:
  https://www.youtube.com/watch?v=NBurr5mJ1Ys&list=PLxAS51iVMjv-rGgtTMpzUzVEFMqj2Zcjb
- Google drive resource activities:
  https://tinyurl.com/MLE-G10IT-ConditionalProgramming