Republic of the Philippines
Department of Education
REGION III
SCHOOLS DIVISION OFFICE OF NUEVA ECIJA

LEARNING ACTIVITY SHEET

SPECIAL PROGRAM IN ICT
ADVANCED PROGRAMMING 10
Second Quarter, Weeks1 - 2

NameofLearner:________________________________ Date:___________________________

Grade Level/Section:____________________________

Loop

BACKGROUND INFORMATION FOR LEARNERS

In computer science, a loop is a programming structure that repeats a sequence of instructions until a specific condition is met. Programmers use loops to cycle through values, add sums of numbers, repeat functions, and many other things.

Loops are supported by all modern programminglanguages, though their implementations and syntax may differ. Two of the most common types of loops are the **while loop** and the **for loop**.

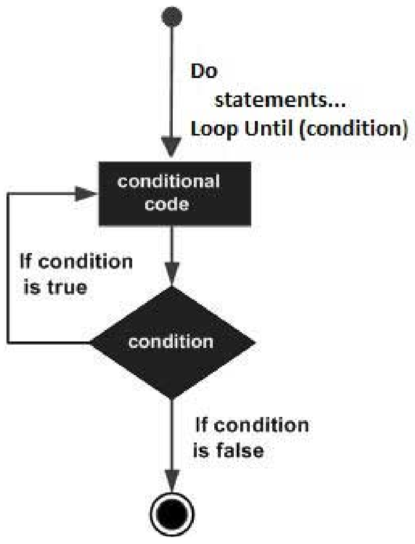
Do Loop

It repeats the enclosed block of statements while a Boolean condition is True or until the condition becomes True. It could be terminated at any time with the Exit Do statement.

The syntax for this loop construct is −

```
Do { While | Until } condition
   [ statements ]
   [ Continue Do ]
   [ statements ]
   [ Exit Do ]
   [ statements ]
Loop
-or-
Do
   [ statements ]
   [ Continue Do ]
   [ statements ]
   [ Exit Do ]
   [ statements ]
Loop { While | Until } condition
```

Flow Diagram



Example:

```
Module loops
SubMain()
' local variable definition
      Dim a As Integer = 10
      'do loop execution
Do
Console.WriteLine("value of a: {0}", a)
          a = a +1
LoopWhile(a <20)
Console.ReadLine()
EndSub
EndModule
```

When the above code is compiled and executed, it produces the following result −

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

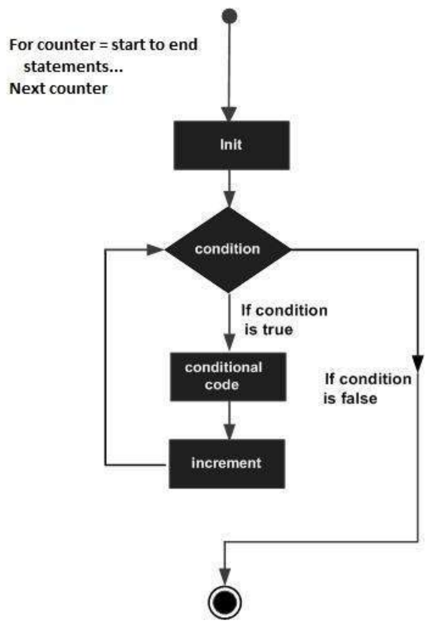
For...Next Loop

It repeats a group of statements a specified number of times and a loop index counts the number of loop iterations as the loop executes.

The syntax for this loop construct is –

```
For counter [ As datatype ] = start To end [ Step step ]
   [ statements ]
   [ Continue For ]
   [ statements ]
   [ Exit For ]
   [ statements ]
Next [ counter ]
```

Flow Diagram



Example:

```
Module loops
SubMain()
Dim a AsByte
' for loop execution
      For a = 10 To 20
Console.WriteLine("value of a: {0}", a)
      Next
Console.ReadLine()
   End Sub
End Module
```

When the above code is compiled and executed, it produces the following result −

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
value of a: 20
```

Each...Next Loop

It repeats a group of statements for each element in a collection. This loop is used for accessing and manipulating all elements in an array or a VB.Net collection.

The syntax for this loop construct is −

```
For Each element [ As datatype ] In group
   [ statements ]
   [ Continue For ]
   [ statements ]
   [ Exit For ]
   [ statements ]
Next [ element ]
```

Example

```
Module loops
SubMain()
DimanArray()AsInteger={1,3,5,7,9}
DimarrayItemAsInteger
'displaying the values

      For Each arrayItemInanArray
Console.WriteLine(arrayItem)
      Next
Console.ReadLine()
   End Sub
End Module
```

When the above code is compiled and executed, it produces the following result −

```
1
3
5
7
9
```

While... End While Loop

It executes a series of statements as long as a given condition is True.

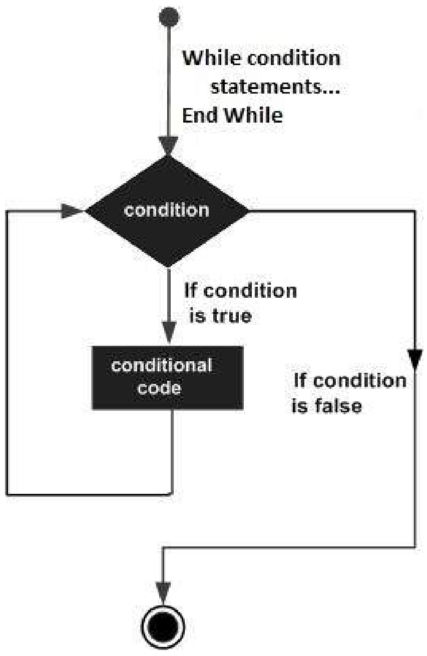
The syntax for this loop construct is –

```
While condition
   [ statements ]
   [ Continue While ]
   [ statements ]
   [ Exit While ]
   [ statements ]
End While
```

Here, statement(s) may be a single statement or a block of statements. The condition may be any expression, and true is logical true. The loop iterates while the condition is true.

When the condition becomes false, program control passes to the line immediately following the loop.

Flow Diagram



Here, key point of the *While* loop is that the loop might not ever run. When the condition is tested and the result is false, the loop body will be skipped and the first statement after the while loop will be executed.

Example

```
Module loops
SubMain()
Dim a AsInteger=10
' while loop execution '

While a <20
Console.WriteLine("value of a: {0}", a)
         a = a +1
EndWhile
Console.ReadLine()
EndSub
EndModule
```

When the above code is compiled and executed, it produces the following result −

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

With... End With Statement

It is **NOT** exactly a looping construct. It executes a series of statements that repeatedly refers to a single object or structure.

The syntax for this loop construct is −

```
With object
   [ statements ]
End With
```

Example

```
Module loops
PublicClassBook
PublicPropertyNameAsString
PublicPropertyAuthorAsString
PublicPropertySubjectAsString
EndClass
SubMain()
DimaBookAsNewBook
WithaBook
.Name="VB.Net Programming"
.Author="Zara Ali"
.Subject="Information Technology"
EndWith
WithaBook
Console.WriteLine(.Name)
Console.WriteLine(.Author)
Console.WriteLine(.Subject)
EndWith
Console.ReadLine()
EndSub
EndModule
```

When the above code is compiled and executed, it produces the following result −

```
VB.Net Programming
Zara Ali
Information Technology
```

Nested Loops

VB.Net allows using one loop inside another loop. Following section shows few examples to illustrate the concept.

Syntax

The syntax for a **nested For loop** statement in VB.Net is as follows −

```
For counter1 [ As datatype1 ] = start1 To end1 [ Step step1 ]
   For counter2 [ As datatype2 ] = start2 To end2 [ Step step2 ]
      ...
   Next [ counter2 ]
Next [ counter 1]
```

The syntax for a **nested While loop** statement in VB.Net is as follows −

```
While condition1
   While condition2
      ...
   End While
End While
```

The syntax for a **nested Do...While loop** statement in VB.Net is as follows −

```
Do { While | Until } condition1
   Do { While | Until } condition2
      ...
   Loop
Loop
```

A final note on loop nesting is that you can put any type of loop inside of any other type of loop. For example, a for loop can be inside a while loop or vice versa.

Example

The following program uses a nested for loop to find the prime numbers from 2 to 100 −

```
Module loops
SubMain()
' local variable definition
      Dim i, j As Integer
      For i = 2 To 100
         For j = 2 To i
            'if factor found,not prime
If((iMod j)=0)Then
ExitFor
EndIf
Next j
If(j >(i \ j))Then
Console.WriteLine("{0} is prime",i)
EndIf
Nexti
Console.ReadLine()
EndSub
EndModule
```

When the above code is compiled and executed, it produces the following result −

```
2 is prime
3 is prime
5 is prime
7 is prime
11 is prime
13 is prime
17 is prime
19 is prime
23 is prime
29 is prime
31 is prime
37 is prime
41 is prime
43 is prime
47 is prime
53 is prime
59 is prime
61 is prime
67 is prime
71 is prime
73 is prime
79 is prime
83 is prime
89 is prime
97 is prime
```

Loop Control Statements

Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

VB.Net provides the following control statements.

Exit Statement

The Exit statement transfers the control from a procedure or block immediately to the statement following the procedure call or the block definition. It terminates the loop, procedure, try block or the select block from where it is called.

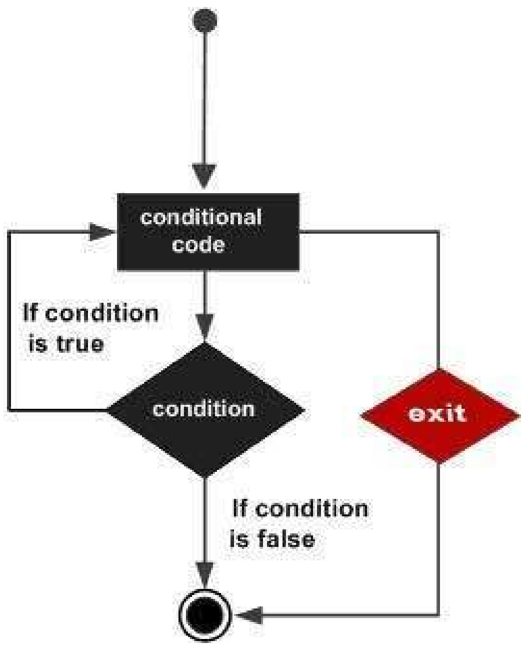
If you are using nested loops (i.e., one loop inside another loop), the Exit statement will stop the execution of the innermost loop and start executing the next line of code after the block.

Syntax

The syntax for the Exit statement is −

```
Exit { Do | For | Function | Property | Select | Sub | Try | While }
```

Flow Diagram



Example

```
Module loops
SubMain()
' local variable definition
      Dim a As Integer = 10
      'while loop execution '

      While (a < 20)
Console.WriteLine("value of a: {0}", a)
         a = a + 1
         If (a > 15) Then
            'terminate the loop usingexit statement
ExitWhile
EndIf
EndWhile
Console.ReadLine()
EndSub
EndModule
```

When the above code is compiled and executed, it produces the following result −

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
```

Continue Statement

The Continue statement causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating. It works somewhat like the Exit statement. Instead of forcing termination, it forces the next iteration of the loop to take place, skipping any code in between.

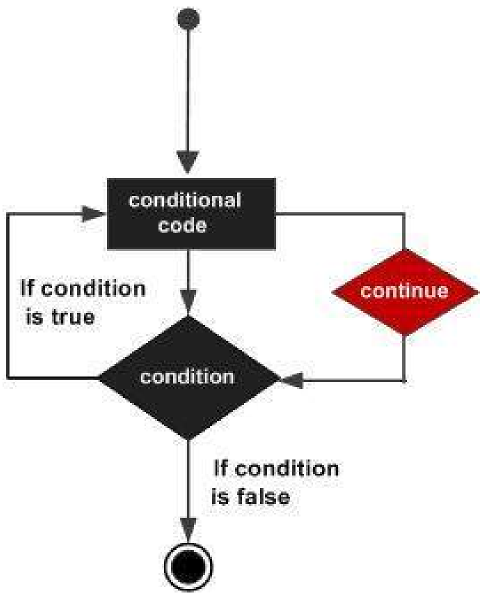
For the For...Next loop, Continue statement causes the conditional test and increment portions of the loop to execute. For the While and Do...While loops, continue statement causes the program control to pass to the conditional tests.

Syntax

The syntax for a Continue statement is as follows −

```
Continue { Do | For | While }
```

Flow Diagram



Example

Live Demo

```
Module loops
SubMain()
' local variable definition
      Dim a As Integer = 10
      Do
         If (a = 15) Then
            ' skip the iteration '
            a = a + 1
            Continue Do
         End If
Console.WriteLine("value of a: {0}", a)
         a = a + 1
      Loop While (a < 20)
Console.ReadLine()
   End Sub
End Module
```

When the above code is compiled and executed, it produces the following result −

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

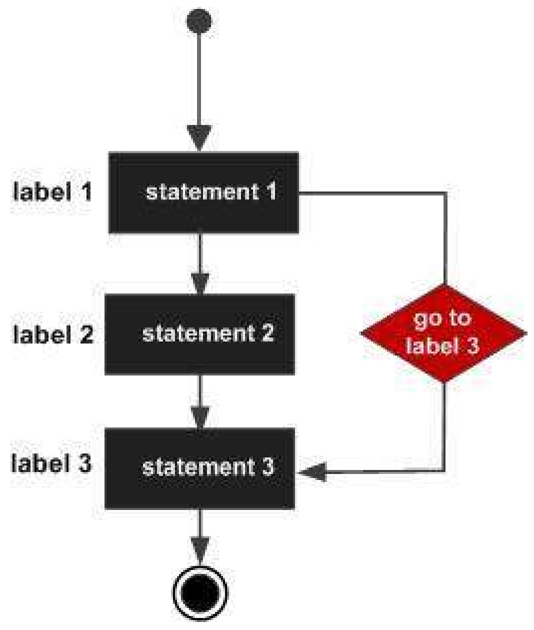
GoTo Statement

The GoTo statement transfers control unconditionally to a specified line in a procedure.

The syntax for the GoTo statement is −

```
GoTo label
```

Flow Diagram



Example

```
Module loops
SubMain()
' local variable definition
      Dim a As Integer = 10
Line1:
      Do
         If (a = 15) Then
            ' skip the iteration '
            a = a + 1
GoTo Line1
         End If
Console.WriteLine("value of a: {0}", a)
         a = a + 1
      Loop While (a < 20)
Console.ReadLine()
   End Sub
End Module
```

When the above code is compiled and executed, it produces the following result −

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

LEARNING COMPETENCY

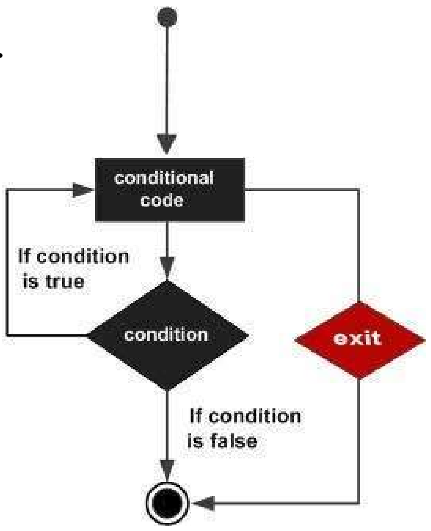
Apply looping structure in a program

ACTIVITIES

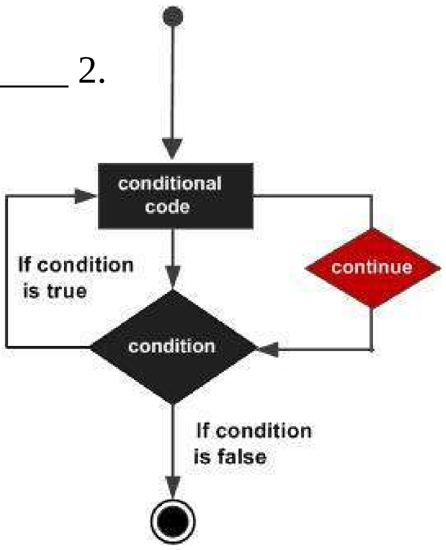
ACTIVITY 1:

Identify what kind of Loop or Statement are the images below based on their Flow Diagram. Write your answer on the space provided before each number.

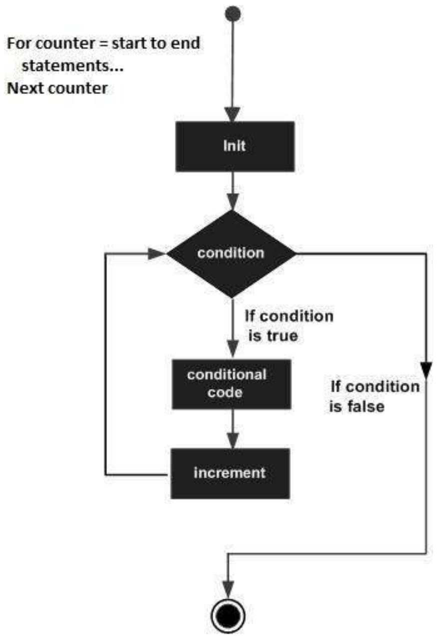
________________1.



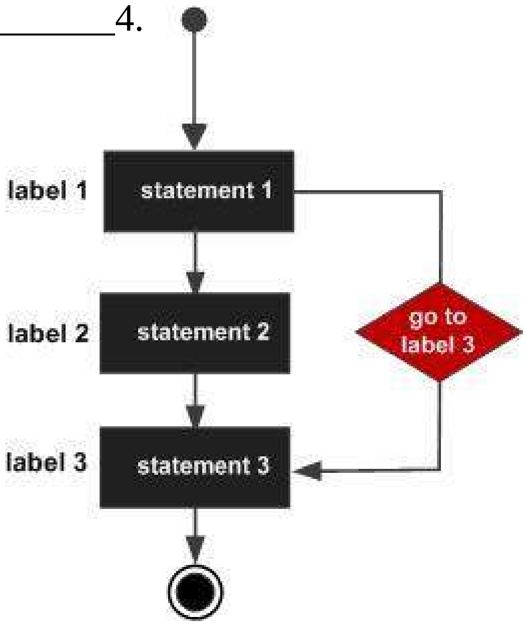
______________ 2.



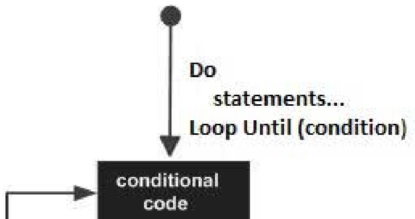
________________3.



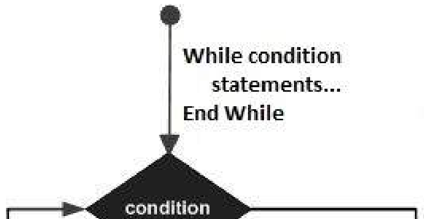
_______________4.



________________5.



________________6.



REFLECTION

How would you relate Loops and Statements in you daily activities as a student?

REFERENCES

https://techterms.com/definition/loop#:~:text=In%20computer%20science%2C%20a%20loop,functions%2C%20and%20many%20other%20things.

https://www.tutorialspoint.com/vb.net/vb.net_loops.htm

Prepared by: **MAXIMO A LUNA JR**