Programming Practices and Techniques

Session 6 – Iteration Constructs

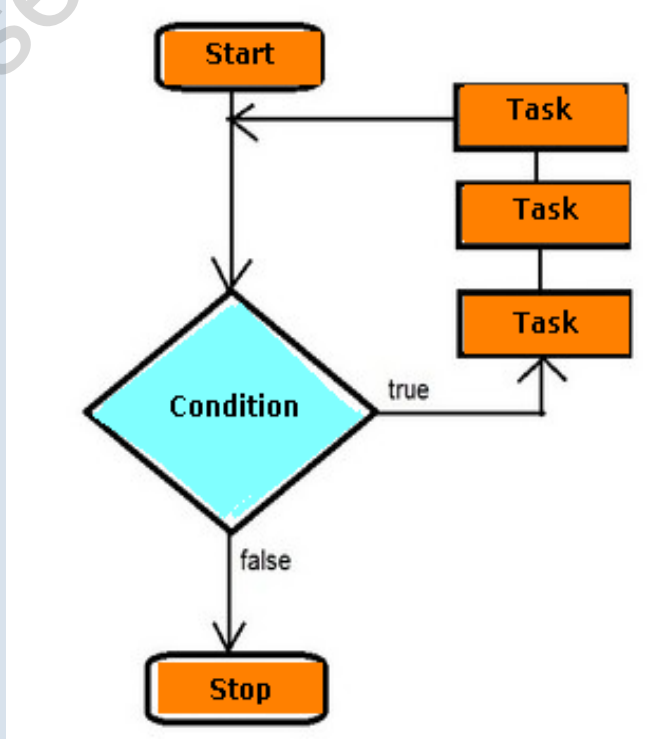
Objectives

- Explain looping constructs
- Describe the different types of loops
- Explain nested loops

Introduction

- Sometimes, a computer program needs to perform some tasks repetitively

- To perform these tasks, the iterative or looping construct is used



Loops 1-2

- Consider that a programmer has to write a program that displays a name 1000 times.
- The programmer cannot type the **`DISPLAY`** statement 1000 times.
- In such kind of scenarios, the iteration construct can be used.

Loops 2-2

- The example shows the pseudocode to add an iteration construct to the **DISPLAY** statement.

```
Do loop 1000 times
    DISPLAY "Mike"
End loop
```

Types of Loops

- Different types of loops supported by most programming languages are as follows:
  - The **WHILE** Loop
  - The **DO…WHILE** Loop
  - The **REPEAT…UNTIL** Loop
  - The **FOR** Loop

The WHILE Loop 1-5

- The **WHILE** loop repeats a statement or a set of statements until a certain specified condition is met or is true.

- The general form of the **WHILE** loop is as follows:

```
WHILE condition
DO
  statement set
END DO
```

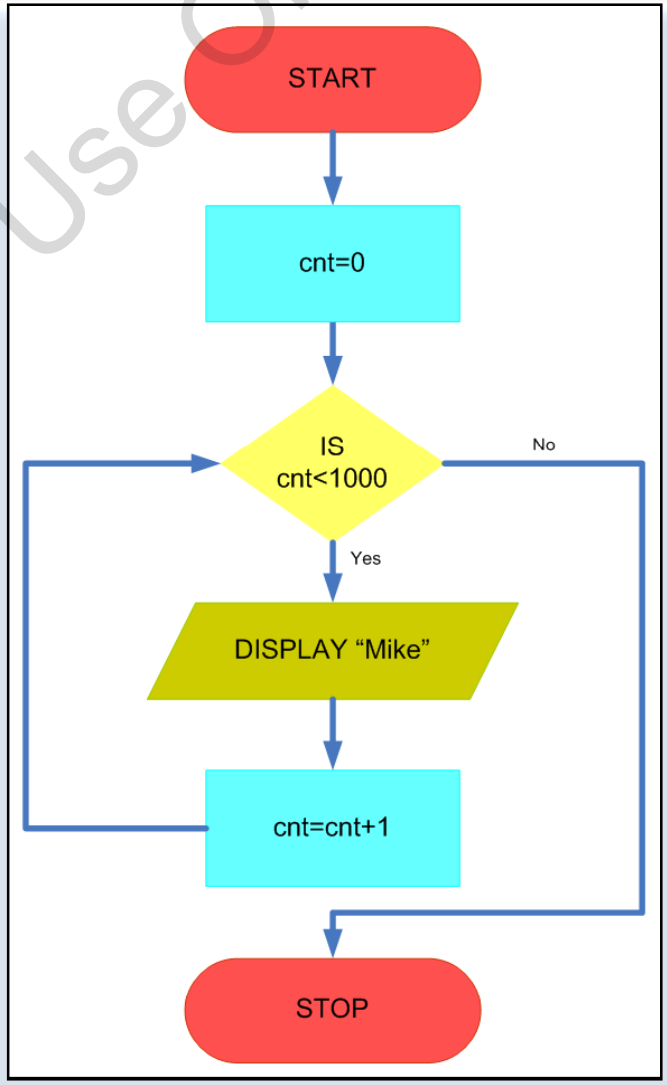
The WHILE Loop 2-5

- The modified pseudocode for the earlier example using the **WHILE** loop is as follows:

```
BEGIN
cnt=0
WHILE (cnt < 1000)
DO
  DISPLAY "Mike"
  cnt=cnt+1
END DO
END
```

The WHILE Loop 3-5

- The flowchart for the pseudocode is shown in the figure.



The WHILE Loop 4-5

- In C, the syntax for the **WHILE** loop is as follows:

```
while(condition)
{
  statement set
}
```

The WHILE Loop 5-5

- In C, the example would be written as follows:

```
#include <stdio.h>
void main()
{
  int cnt;
  cnt=0;
  while(cnt<1000)
  {
     printf("Mike");
     cnt++;
  }
}
```

The DO…WHILE Loop 1-4

- The **WHILE** loop tests the condition before the body of the loop is executed.
- Therefore, the body of the loop may not be executed at all if the condition is not satisfied in the first attempt.
- On some occasions, it might be necessary to execute the body of the loop before the test is performed.
- Such situations can be handled with the help of the **DO…WHILE** loop.

The DO…WHILE Loop 2-4

- The general form of the **DO…WHILE** loop is as follows:

```
DO
Statements
WHILE condition
```

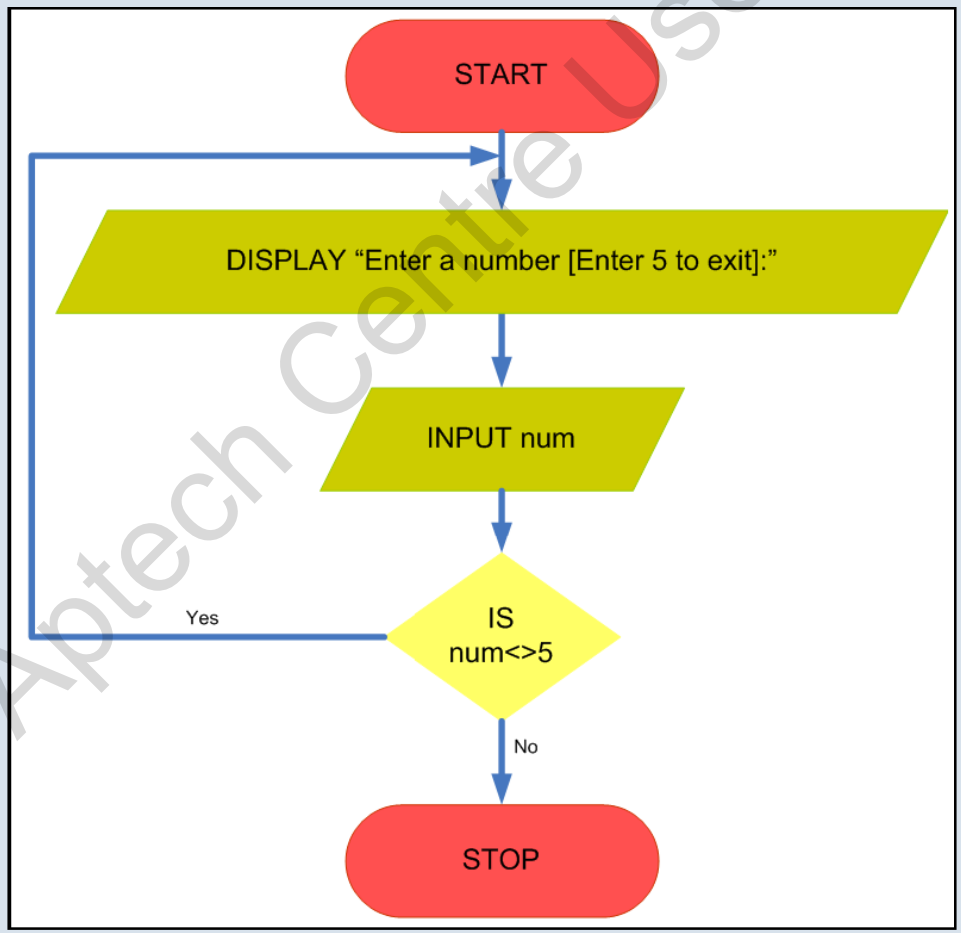
The DO…WHILE Loop 3-4

- Consider an example where the loop displays a message and accepts a number from the user.
- The loop continues doing this until the user enters the number 5 and then, exits the program.
- The pseudocode for this is as follows:

```
BEGIN
DO
    DISPLAY "Enter a number [enter 5 to exit];"
    INPUT num
WHILE (num <> 5)
END
```

The DO…WHILE Loop 4-4

- The flowchart for the example is shown in the figure.



The FOR Loop 1-4

- The **FOR** loop provides a more concise loop control structure.
- The general form for a **FOR** loop is as follows:

```
FOR counterVariable IN RANGE startValue
to endValue [STEP value]
DO
  statement set
END DO
```

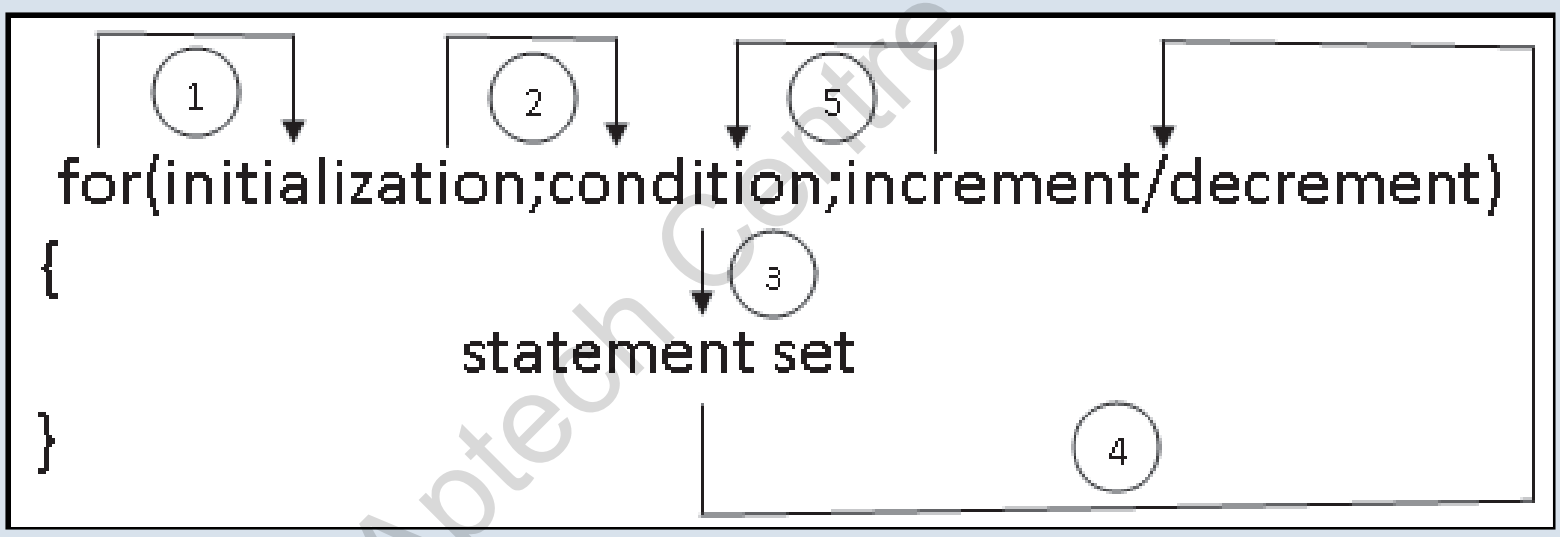
The FOR Loop 2-4

- Consider the earlier example to display a name 1000 times.

```
BEGIN
FOR cnt IN RANGE 1 to 1000
DO
  DISPLAY "Mike"
END DO
END
```

The FOR Loop 3-4

- The C syntax for the **FOR** loop along with its execution path is shown in the figure.



The FOR Loop 4-4

- The **FOR** loop construct for displaying the name Mike 1000 times in C language is shown in the example.

```
…
for(cnt=0;cnt<1000;cnt++)
{
  printf("Mike");
}
…
```

Nested Loops 1-2

- Nested loop:
  - Is a loop within another loop.
  - Is used in almost all programming languages.
  - Can have upto 15 levels in C language to be nested within one another.
  - Can have varied levels of nesting for other programming languages.

Nested Loops 2-2

- An example of a nested **WHILE** loop where the program accepts five numbers from the user and displays them twice on the screen is shown.

```
BEGIN
cnt = 1
cnt2 = 1
WHILE cnt < 6
INPUT num
   cnt2 = 1
   WHILE cnt2 < 3
   DO
      PRINT num
   cnt2 = cnt2 + 1
   END DO
cnt = cnt + 1
END DO
END
```

Summary

- The iterative or looping construct is used to repeat certain steps a specific number of times or till some specified condition is met.
- The different types of loops are the **WHILE** loop, the **DO…WHILE** loop, the **REPEAT…UNTIL** loop, and the **FOR** loop.
- The **WHILE** loop repeats a statement or a set of statements while a certain specified condition is True.
- The **DO…WHILE** loop executes the body of the loop before the condition test is performed.
- The **REPEAT…UNTIL** loop executes as long as the condition is False.
- The **FOR** loop provides a more concise loop control structure, which includes counter variables, range, and step value.
- A nested loop means a loop within another loop.