# LOOPS IN C

Prepared By
**Mahantesh S. Devoor**

# Content

- Objective
- Introduction
- Types of looping
  - while
  - do-while
  - for
- Assessment metric
- Conclusion
- References

# OBJECTIVES

## Course Objective

- Understand the basic terminology used in computer programming
- It stresses the strengths of C, which provide students with the means of writing efficient, maintainable, and portable code.
- write, compile and debug programs in C language.
- Increase the ability to learn new programming languages

# Topic Objective

- Understand the basics of looping.
- To use the while, do-while and for repetition statement to execute statements in a program repeatedly.

# INTRODUCTION

- Statements in a program are executed one after the other

```
ex:    statement 1;
       statement 2;
           :
       statement n;
```

- Sometimes, the user want to execute a set of statements repeatedly.

- Loop statements are used to repeat the execution of statement or blocks.

- Iteration of a loop: the number of times the body of loop is executed.

- Two types of loop structure are:

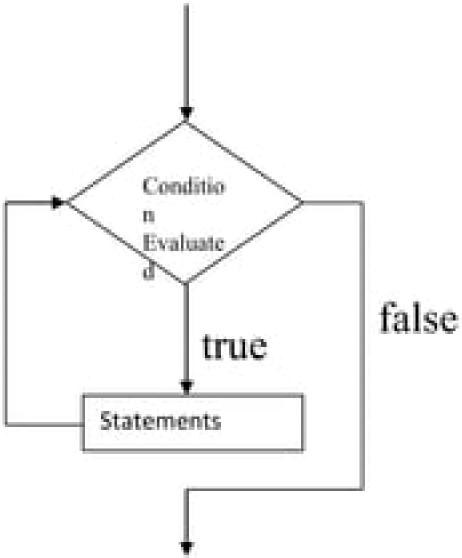
  Pretest : Entry - controlled loop

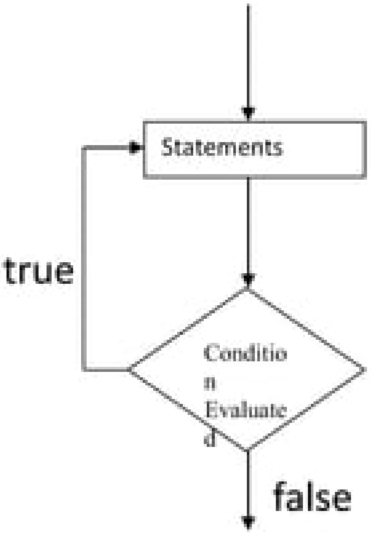
  Posttest : Exit – controlled loop

# Pretest Vs. Posttest

***Pretest :*** Condition is tested before each iteration to check if loops should occur.

***Posttest :*** Condition is tested after each iteration to check if loop should continue (at least a single iteration occurs).





# TYPES OF LOOP

- while loop
- do-while loop
- for loop

# while Loop

- It has a loop condition only that is tested before each iteration to decide whether to continue or terminate the loop.
- The body of a while loop will execute zero or more times

**Syntax:**

```
    while (<condition>){
        <statement/block>;
       }
```

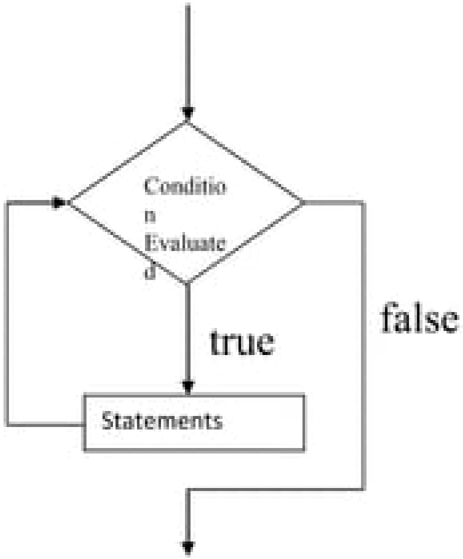
**Example :**

```
int i=0;
while(i<3){
     printf(“Hello\n”);
    i++;
    }
```

**Output:**

```
Hello
Hello
Hello
```

## Flow diagram



# do…while Loop

❖ Do while has a loop condition only that is tested after each iteration to decide whether to continue with next iteration or terminate the loop.

**Syntax:**

```
    do{
      <statement/block>;
         }while(condition);
```

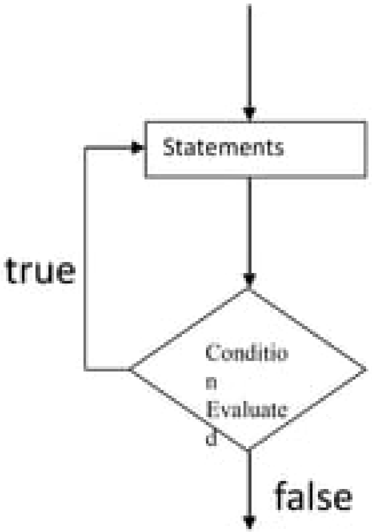
**Example:**

```
int i=0;
do{
        Printf (“Hello\n”);
            i++;
      } while (i<3);
```

**Output:**

```
Hello
Hello
Hello
```

## Flow diagram



# for Loop

for loop has three parts:

- Initializer is executed at start of loop.

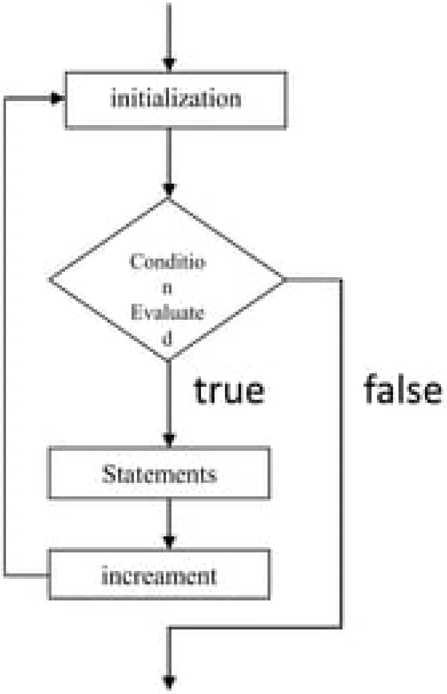
- Loop condition is tested before iteration to decide whether to continue or terminate the loop.

- Increment is executed at the end of each loop iteration.

**Syntax:**

```
for( [initialize]; [condition]; [incrementor] )
{
        <statement/block>;
}
```

## Flow diagram



**Example:**

```
for(i=0; i<3; i++)
{
        printf(“Hello\n”);
}
```

**Output:**

```
        Hello
        Hello
        Hello
```

# ASSESSMENT METRIC

- What is looping? List the types of looping.
- Explain the while loop with an example.
- Give the difference between while and do-while loops
- Explain the syntax of for loop with an example
- List out the difference between while and for loop. And also explain the do-while loop.

# CONCLUSION

❖ Importance of loops in any programming language is immense, they allow us to reduce the number of lines in a code, making our code more readable and efficient.

# REFERENCES

[1]. E. Balaguruswamy, **“Programming in ANSI C”,** Third edition, Tata McGraw Hill Publications, 2002.

[2]. P. B. Kotur, **“Computer Concepts and C programming”,** Kindle Edition, Sapna Book House, Bangalore, 2009.

# Thank You