

# Looping

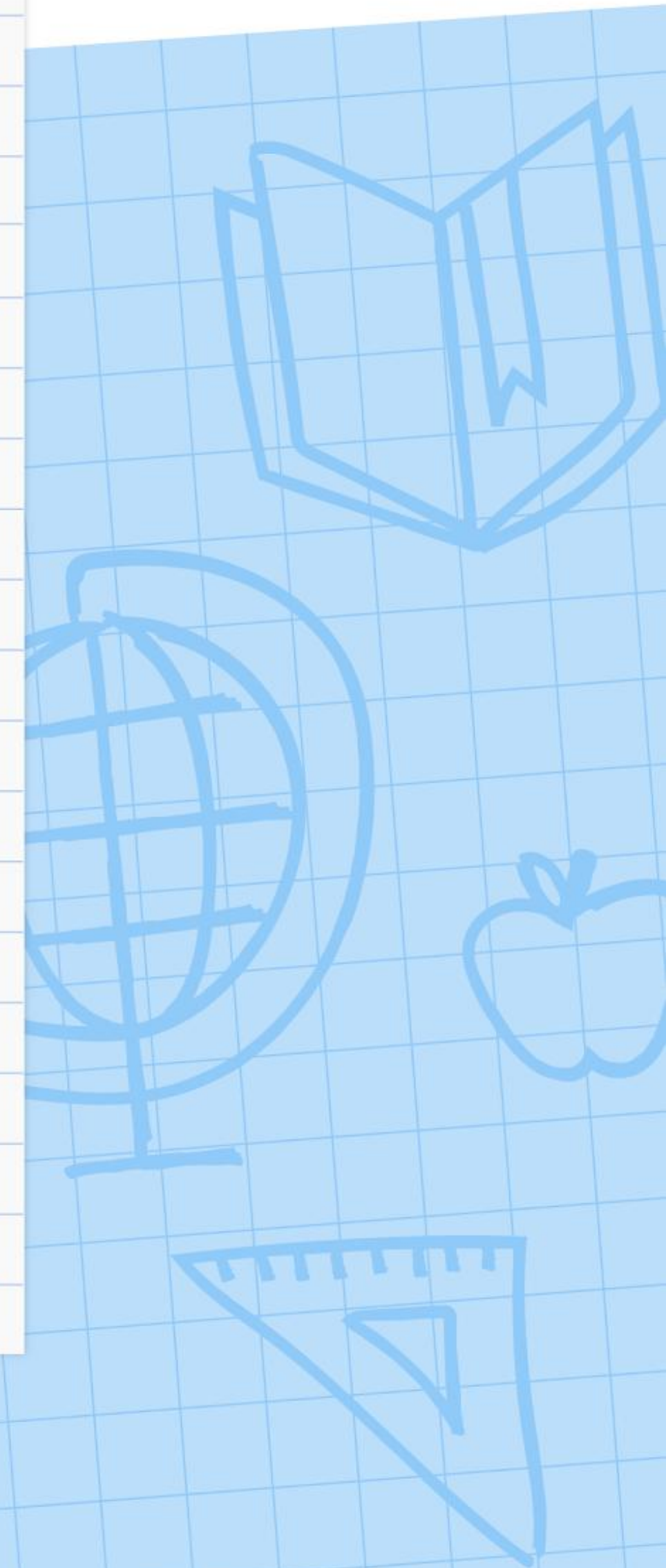
Iteration/Looping in C Programming

Mina Maher

@menamosadef5@gmail.com

```
filterByOrg = filterByOrg ? study.lead_organization === filterByOrg : true  
filterByStatus = filterByStatus ? study.status === filterByStatus : true  
return (filterByOrg || filterByStatus) ? study : null  
} catch (err) {  
    console.log(err)  
}
```

```
function filterStudies({ studies, filterByOrg, filterByStatus }) {  
    return studies.filter(study => {  
        filterByOrg = filterByOrg ? study.lead_organization === filterByOrg : true  
        filterByStatus = filterByStatus ? study.status === filterByStatus : true  
        return (filterByOrg || filterByStatus) ? study : null  
    })  
}
```





# What is Looping ?

- The looping can be defined as repeating the same process multiple times until a specific condition satisfies. It is known as iteration also. There are three types of loops used in the C language language. In this part of the tutorial, we are going to learn all the aspects of C loops.



# Why looping?

- The looping simplifies the complex problems into the easy ones. It enables to alter the flow of the program so that instead of writing the same code again and again, we can execute the same code for a finite number of times.
- For example, if we need to (printf) 'UNIVERSITY OF CALCUTTA' 10-times then, instead of using the (printf) statement 10 times, we can use (printf ) once inside a loop which runs up to 10 iteration.



# What are the advantages of Looping?

- It provides code reusability.
- Using loops, we do not need to write the same code again and again.
- Using loops, we can traverse over the elements of data structures (array or linked lists).

# Types of C Loops

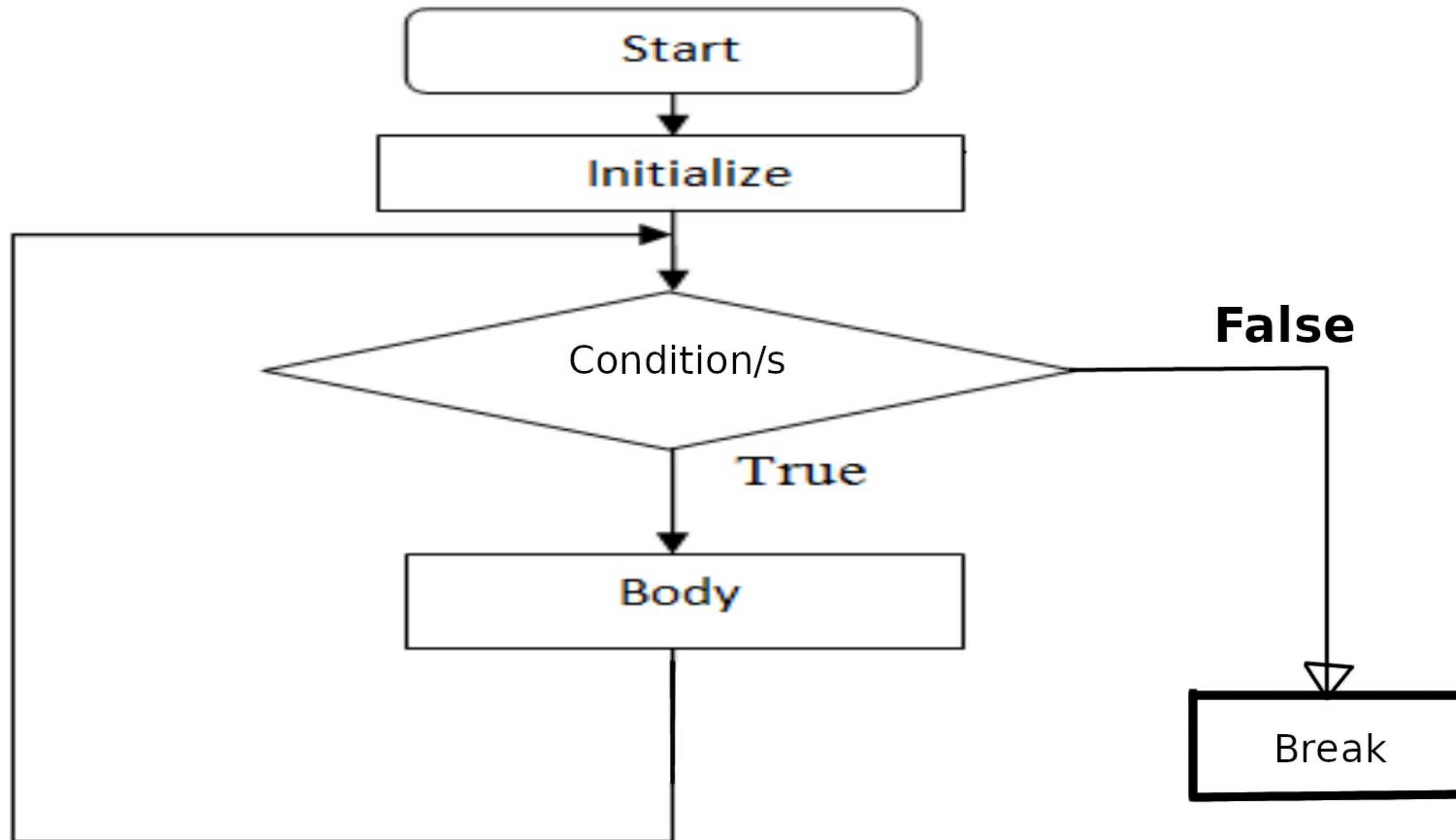
There are three types of loops in C language those are given below:

- While
- do while
- for

# Essential components of a loop

- Counter
- Initialisation of the counter with initial value
- Condition to check with the optimum value of the counter .
- Statement(s) to be executed by iteration
- Increment/decrement

# Flowchart for a loop





# “while” loop in C

- The “while” loop executes the code block “block” as long as the conditional statement “condition” is true, in this case the loop is denoted as a pre-tested loop (the condition encircles the code block).
- The syntax of while loop in c language is given below :

```
variable-initialisation;  
while (condition) {  
    block;  
}
```



Write a C-program to print 10 natural numbers

```
#include <stdio.h>
int main() {
    int i=1;
    while(i <= 10)
    {
        printf("%d \n",i);
        i=i+1;
    }
}
```

output

1

2

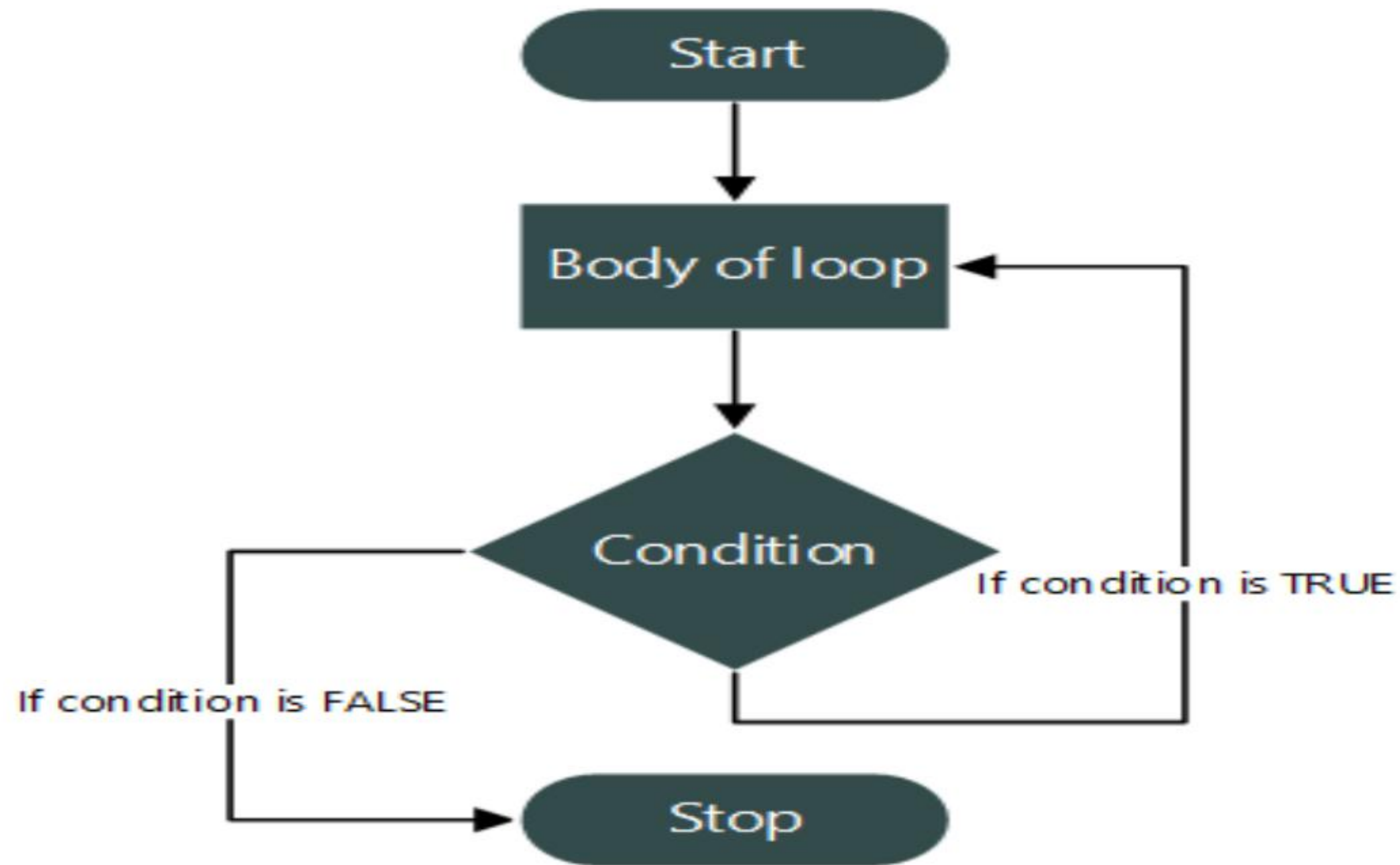
10

# do-while loop in C

- The “do-while” loop continues until a given condition satisfies. It is also called **post-tested loop**. It is used when it is necessary to **execute the loop at least once** (mostly menu driven programs).
- The syntax of The syntax of do-while loop in c language loop in c language is given below: is given below:

```
int index = 0;
do
{
    block;
} while (condition);
```

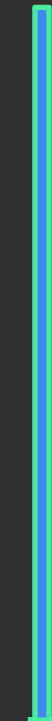
# Flowchart for the “do-while” loop





```
/* same above project */  
#include <stdio.h>  
int main(void)  
{  
    int i = 0;  
    do  
    {  
        printf("%d \n",i);  
        i=i+1;  
    } while(i<=10);  
    return 0;  
}
```

output



1  
10

# “for” loops in C

- A “for” loop in C language is used to iterate the statements or a part of the program several times. It is frequently used to traverse the data structures like the array and linked list.

The syntax of for loop in c language is given below: .

```
for (initialization; condition/s; statement/s)
{
    block;
}
```

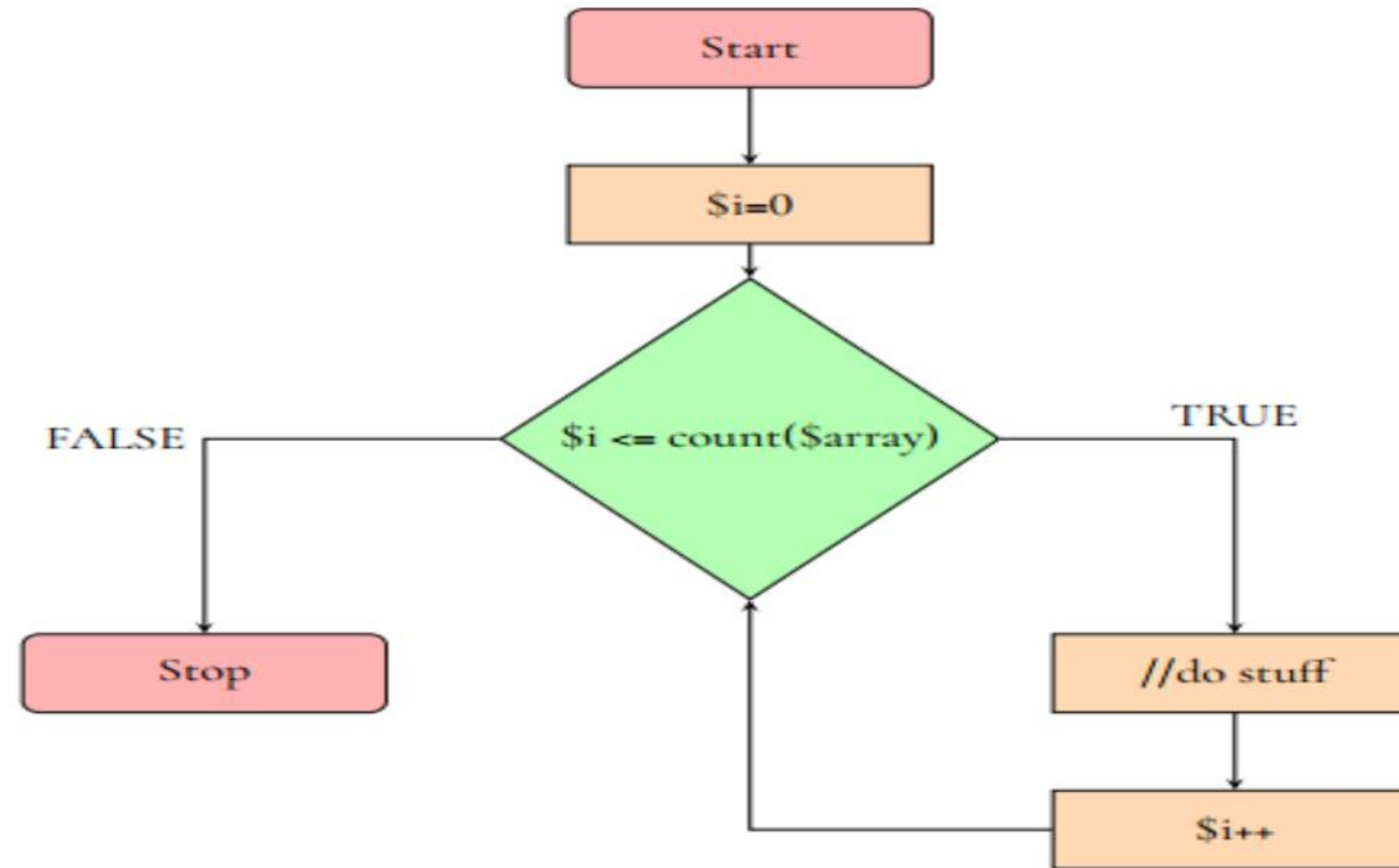
- **Initialization:** an initialization of the loop variable and more than one variable can be initialised.
- **Condition/s:** a conditional expression, it checks for a specific condition to be satisfied and if it is not, the loop is terminated, it also can have more than one condition. However, the loop will iterate until the last condition becomes false. Other conditions will be treated as statements.
- **Statement/s:** can be used as a statement to update the value of the loop variable (re-assignment statement).

The syntax of for loop in c language is given below: .

```
for (initialization; condition/s; statement/s)
{
    block;
}
```



# Flowchart : "for" loop



program to print natural numbers 1 to 15

```
#include <stdio.h>
int main()
{
    int i;
    for (i = 1; i <= 15; i = i+1) {
        printf("%d, ", i);
    }
    return 0;
}
```

output 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,

# Nested loops in C

C programming language allows to use one loop inside another loop.

```
for (init/s; condition/s; statement/s)
{
    for (init/s; condition/s; statement/s)
    {
        statement(s);
    }
    statement(s);
}
```





- The syntax for a nested while loop statement in C programming language is as follows :

```
while (condition)
{
    while (condition)
    {
        statement(s);
    }
    statement(s);
}
```



- The syntax for a nested do...while loop statement in C programming language is as follows:

### NOTE :

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

```
do
{
    statement(s);
    do
    {
        statement(s);
    } while (condition);
} while (condition);
```



# break-statement in C

The break statement in C programming language has the following two usages :

1. When the break statement is encountered inside a loop, the loop is immediately terminated and program control resumes at the next statement following the loop.
2. It can be used to terminate a case in the switch statement (covered in the next chapter).

## NOTE :

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

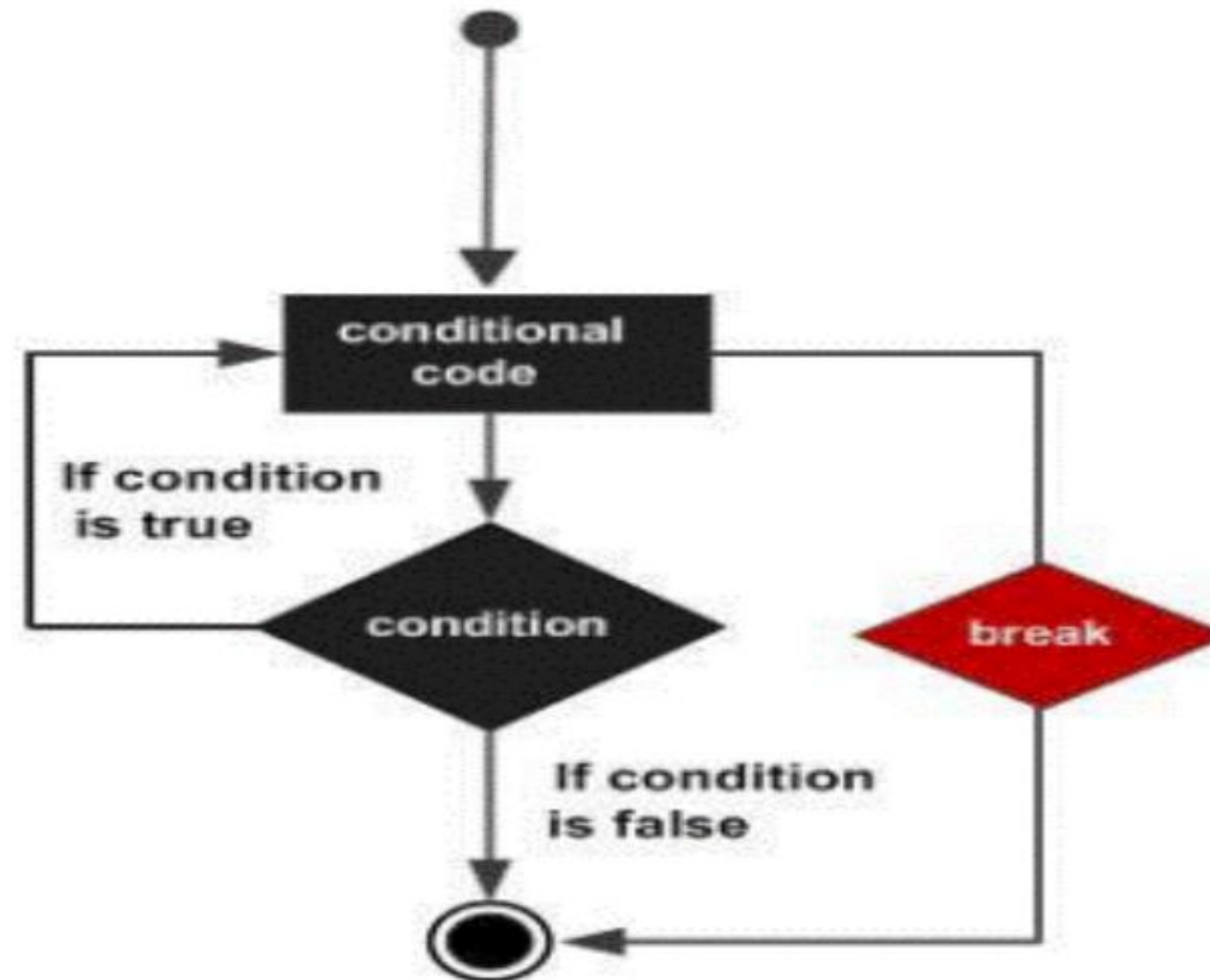


# SYNTAX and FLOWCHART

The syntax for a break statement in C is as follows:

```
break ;
```

● Flowchart of break :



```
/* project about break statement */
#include <stdio.h>
int main ()
{
    /* local variable definition */
    int a = 10;
    /* while loop execution */
    while (a < 20)
    {
        printf("value of a: %d\n", a);
        a++;
        if (a > 15)
        {
            /* terminate the loop using break statement */
            break;
        }
    }
    return 0;
}
```

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

## Output :

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 in C

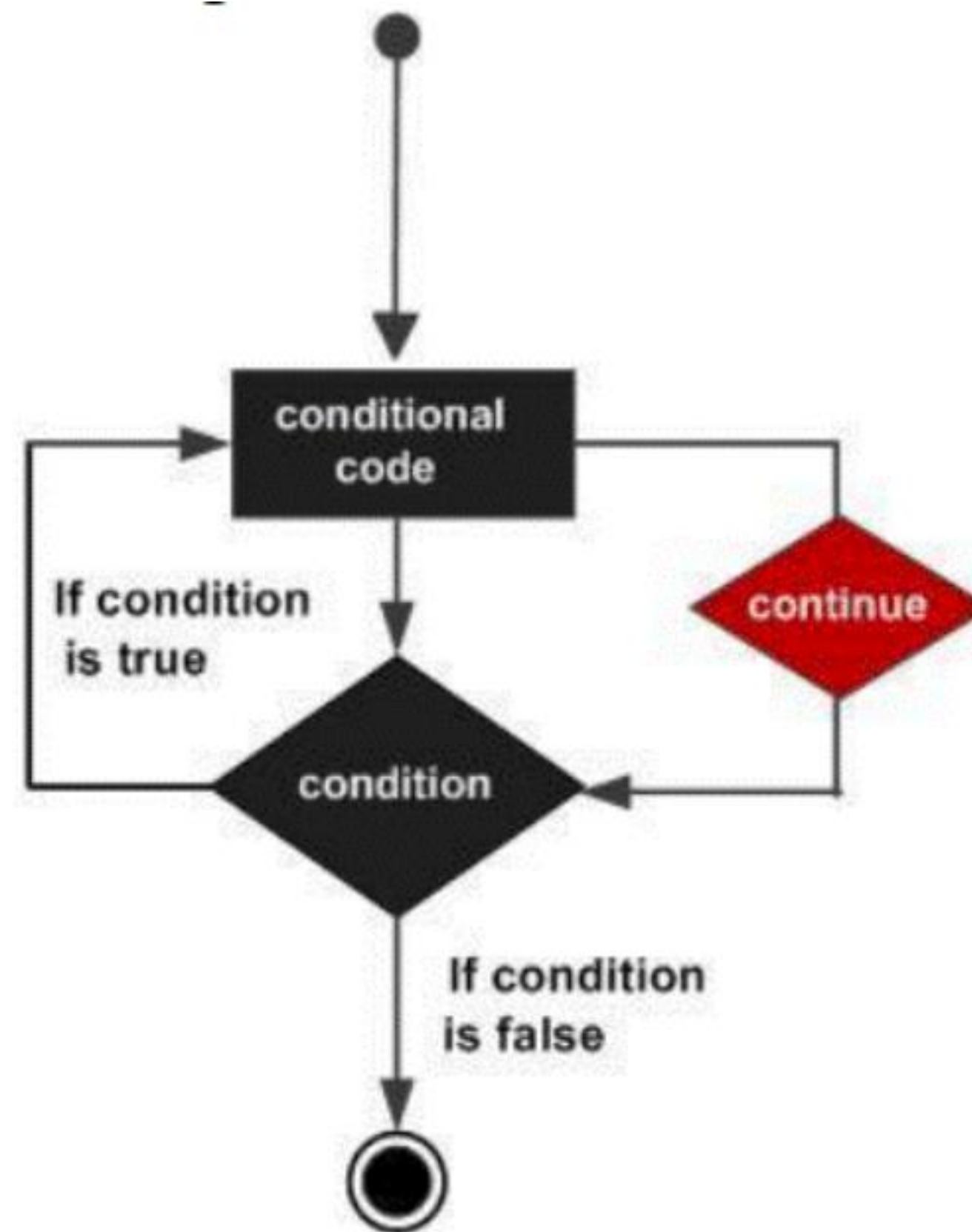
- The continue statement in C programming language works somewhat like the break statement. Instead of forcing termination, however, continue forces the next iteration of the loop to take place, skipping any code in between.
- For the “for” loop, the continue statement causes the conditional test and increment portions of the loop to be executed. For the “while” and “do...while” loops, the continue statement causes the program control to pass the conditional tests.

# Syntax and Flowchart

The syntax for a continue statement in C is as follows:

```
continue;
```

Flowchart of Continue



```
#include <stdio.h>
int main ()
{
    /* local variable definition */
    int a = 10;
    /* do loop execution */
    do {
        if (a == 15) {
            /* skip the iteration */
            a = a + 1;
            continue;
        }
        printf("value of a: %d\n", a);
        a++;
    } while( a < 20 );
    return 0;
}
```



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 in C

A goto statement in C programming language provides an unconditional jump from the goto to a labeled statement in the same function.

## NOTE:

Use of goto statement is highly discouraged in any programming language because it makes difficult to trace the control flow of a program, making the program hard to understand and hard to modify. Any program that uses a goto can be rewritten so that it doesn't need the goto.

# Syntax and Flowchart

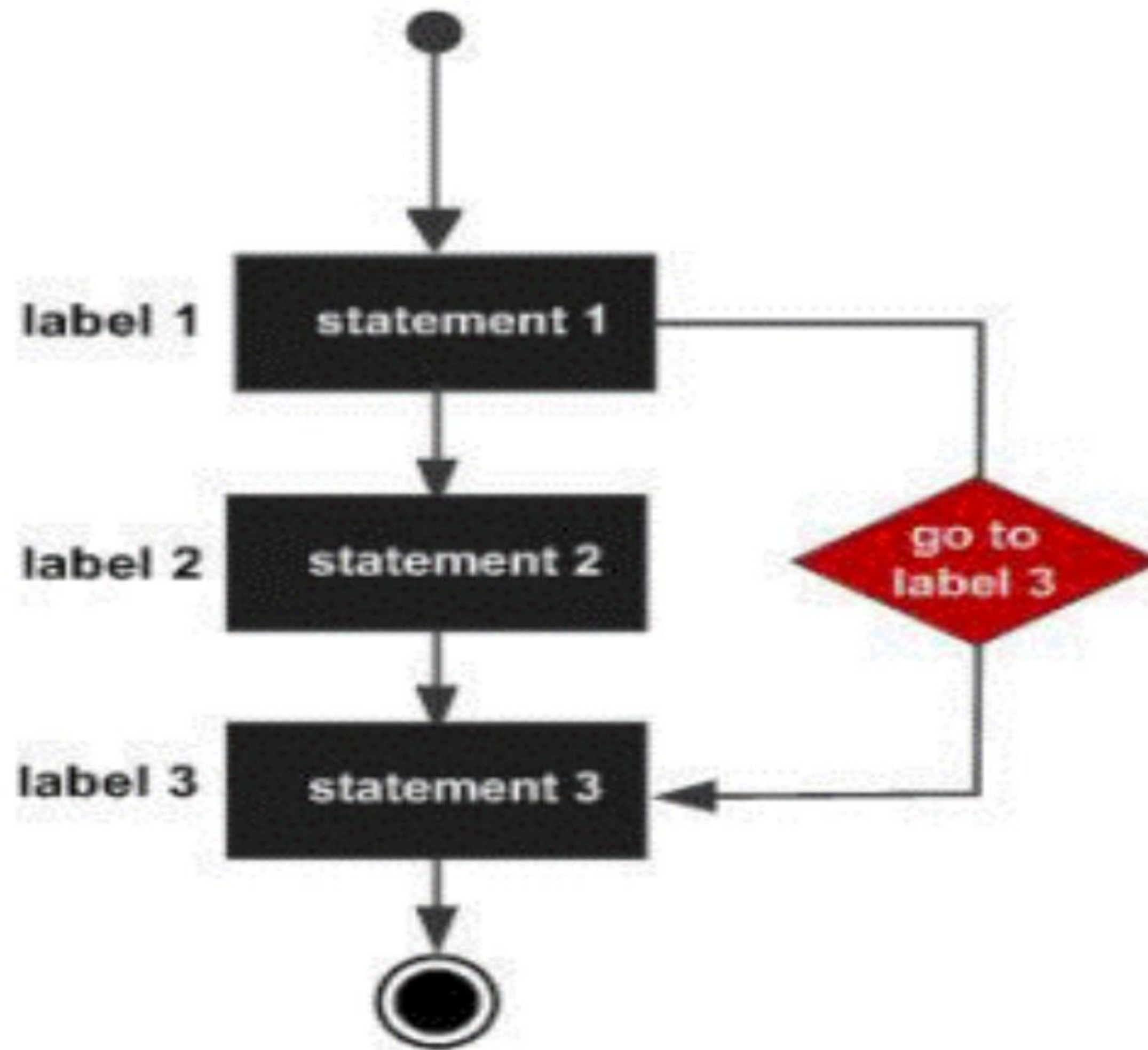
The syntax for a goto statement in C is as follows:

```
goto label;  
..  
.  
label: statement;
```

## Note :

Here label can be any plain text except C keyword and it can be set anywhere in the C program above or below to goto statement.

## Flowchart of go to :





## Example

```
#include <stdio.h>
int main ()
{
    /* local variable definition */
    int a = 10;
    /* do loop execution */
    LOOP:
        do
        {
            if (a == 15) {
                /* skip the iteration */
                a = a + 1;
                goto LOOP;
            }
            printf ("value of a: %d\n", a);
                a++;
        } while (a < 20);
    return 0;
}
```

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

# The Infinite Loop

- A loop becomes infinite loop if a condition never becomes false.
- The for loop is traditionally used for this purpose. Since none of the three expressions that form the for loop are required, you can make an endless loop by leaving the conditional expression empty.

```
#include<stdio.h>
int main () {
    for( ; ; )
    {
        printf("This loop will run forever.\n");
    }
    return 0;
}
```

- When the conditional expression is absent, it is assumed to be true. You may have an initialization and increment expression, but C programmers more commonly use the `for(;;)` construct to signify an infinite loop.

## NOTE:

You can terminate an infinite loop by pressing Ctrl + C keys.

Thank you

At the end please JUST SMILE