Chapter – 6
Iteration in C

Course Code: CIS 115 & 115 L

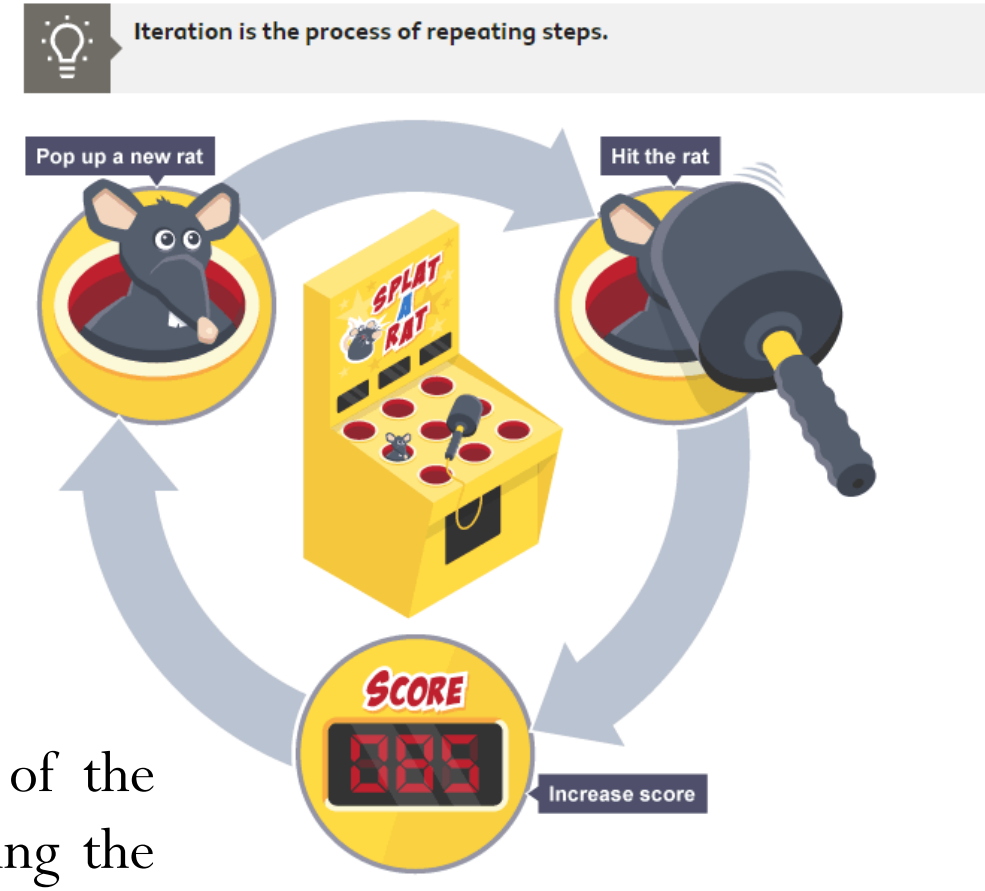
Course Title: Structured Programming

Course Teacher: Md. Faruk Hosen

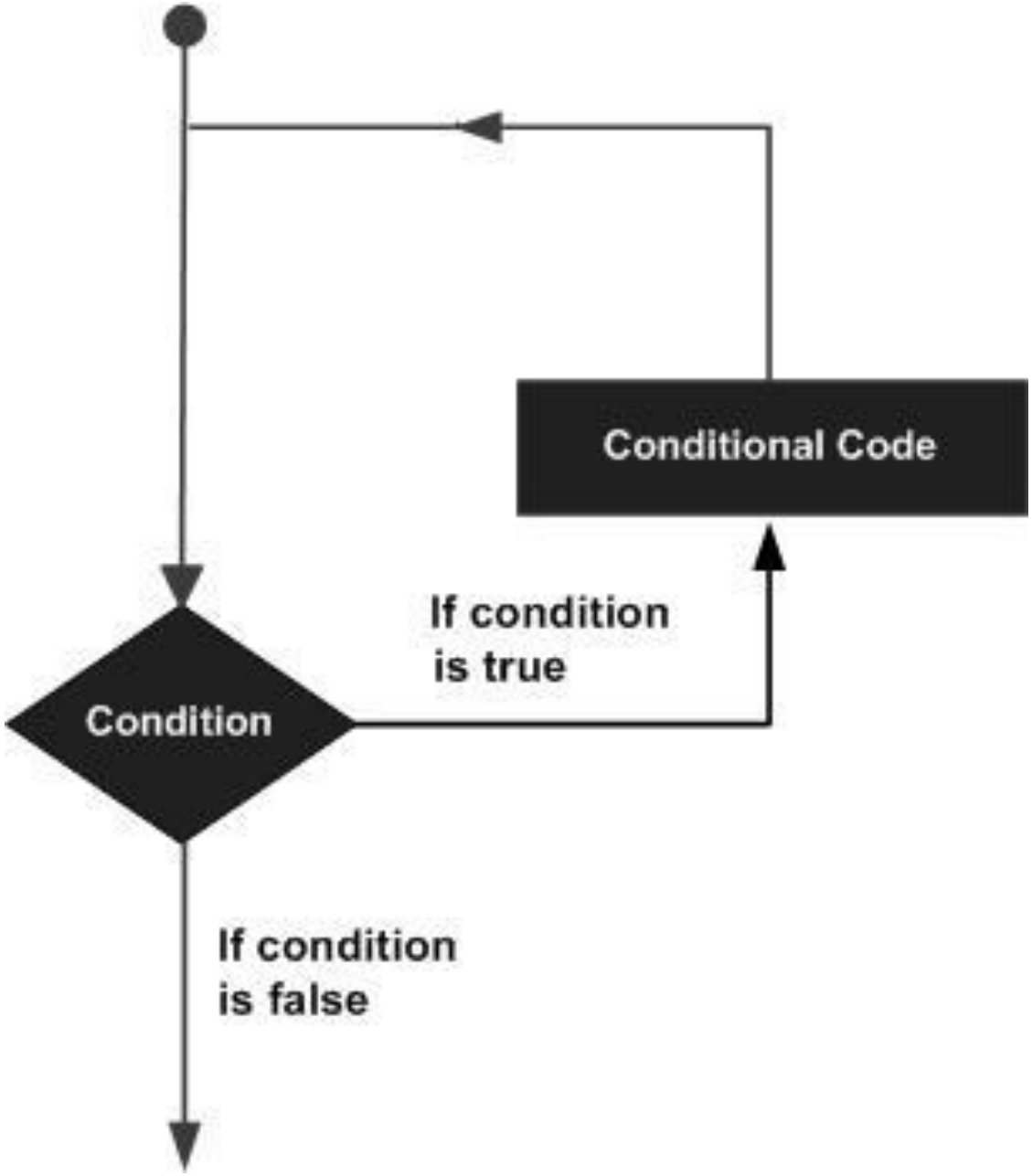
Prepared by: ABK Bhuiyan (Jehad)

Iterations

- *The repeated execution of some groups of code statements in a program is called* ***iteration.***



It enables us to alter the flow of the program so that instead of writing the same code again and again, we can repeat the same code for a finite number of times.


Conditional Code
If condition
is true
Condition
If condition
is false

Types of Iteration

- **1. Bounded or Finite Iteration** --- Bounded iteration is iteration where the number of times has a determined maximum when it is started. This is typically (but not necessarily) the case in a for loop.

- **2. Unbounded or Infinite Iteration** --- Unbounded iteration is iteration where the number of times has a determined maximum when it is started. This is typically (but not necessarily) the case in a while loop.

Ways to implement iteration in C (Loop)

- There are three different ways to implement iteration in c. these are
  - For loop
  - While loop
  - Do while loop

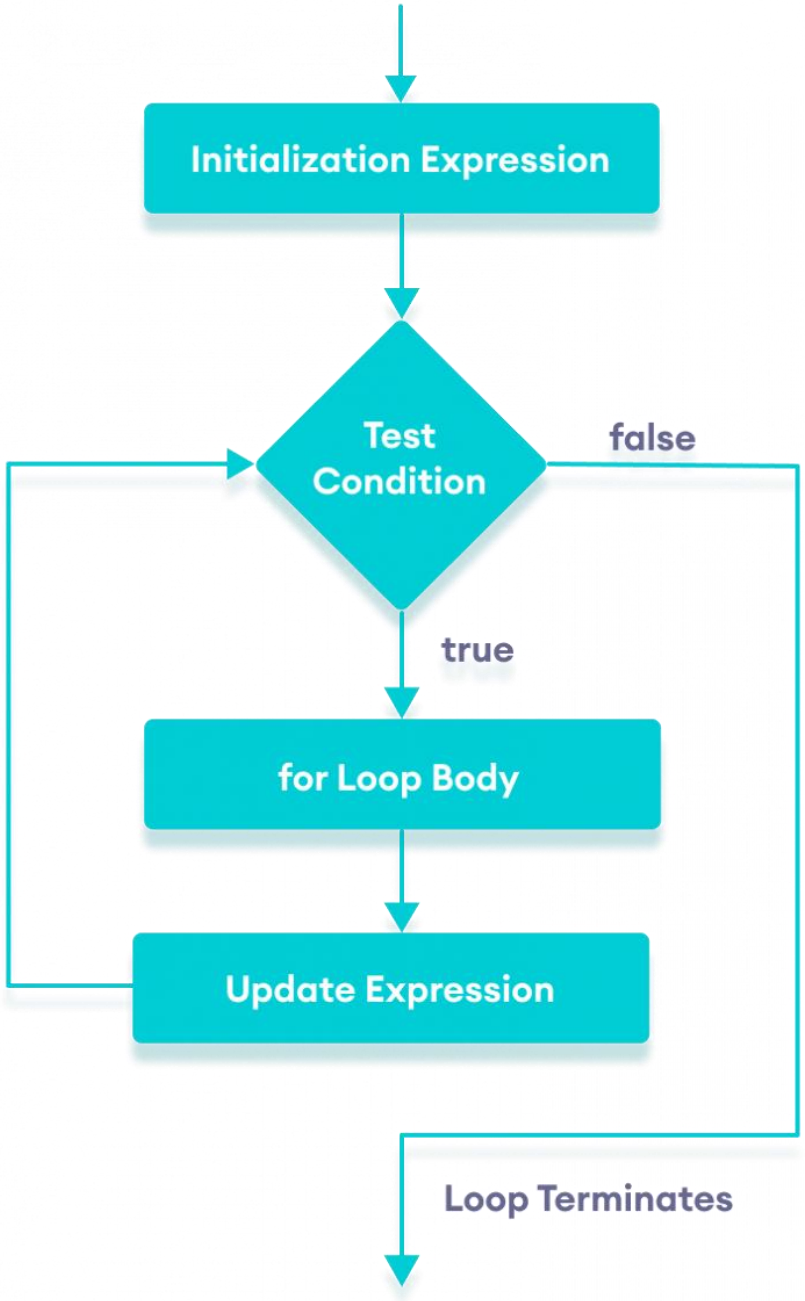
C For Loop

- C for loop is used to run a block of code for a certain number of times. The syntax of for loop is---

```
for (initialExpression; testExpression; updateExpression) {
        // body of the loop
}
```

- Here,
  - The initialExpression initializes and/or declares variables and executes only once.
  - The condition is evaluated. If the condition is true, the body of the for loop is executed.
  - The updateExpression updates the value of initialExpression.
  - The condition is evaluated again. The process continue until the condition is false.

C For Loop


Initialization Expression
Test Condition
false
true
for Loop Body
Update Expression
Loop Terminates

C For Loop Syntax

```
int n = 5; // for loop
for (int i = 1; i <= n; ++i)
{
        System.out.println("%d. C is fun", i);
 }
```

C While Loop

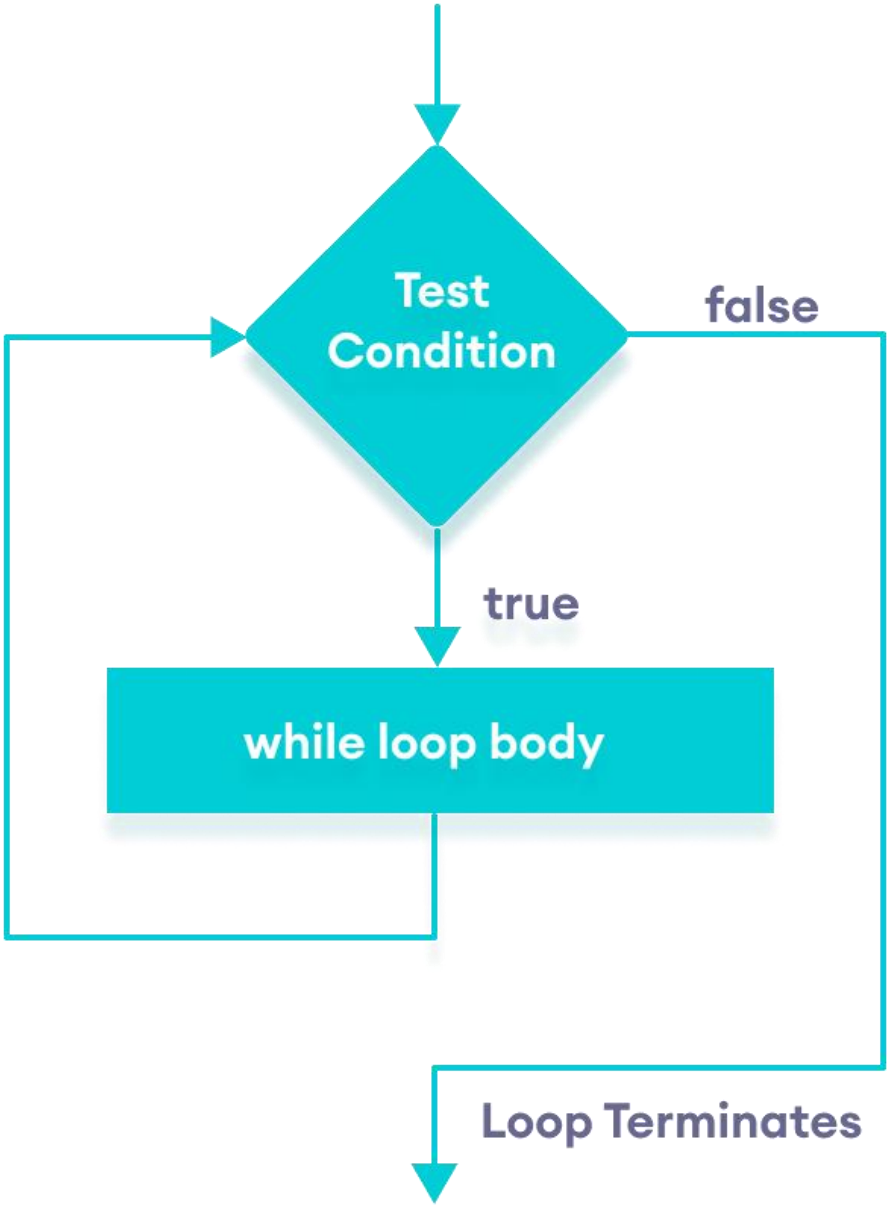
- C while loop is used to run a specific code until a certain condition is met. The syntax of the while loop is:

```
while (testExpression) {
  // body of loop
}
```

- Here,

- A while loop evaluates the textExpression inside the parenthesis ().
- If the textExpression evaluates to true, the code inside the while loop is executed.
- The textExpression is evaluated again.
- This process continues until the textExpression is false.
- When the textExpression evaluates to false, the loop stops.

C While Loop



C while loop syntax

```
int x = 1;

// Exit when x becomes greater than 4
while (x <= 4)
{
    System.out.println("Value of x:" + x);

    // Increment the value of x for
    // next iteration
    x++;
}
```

C do while loop

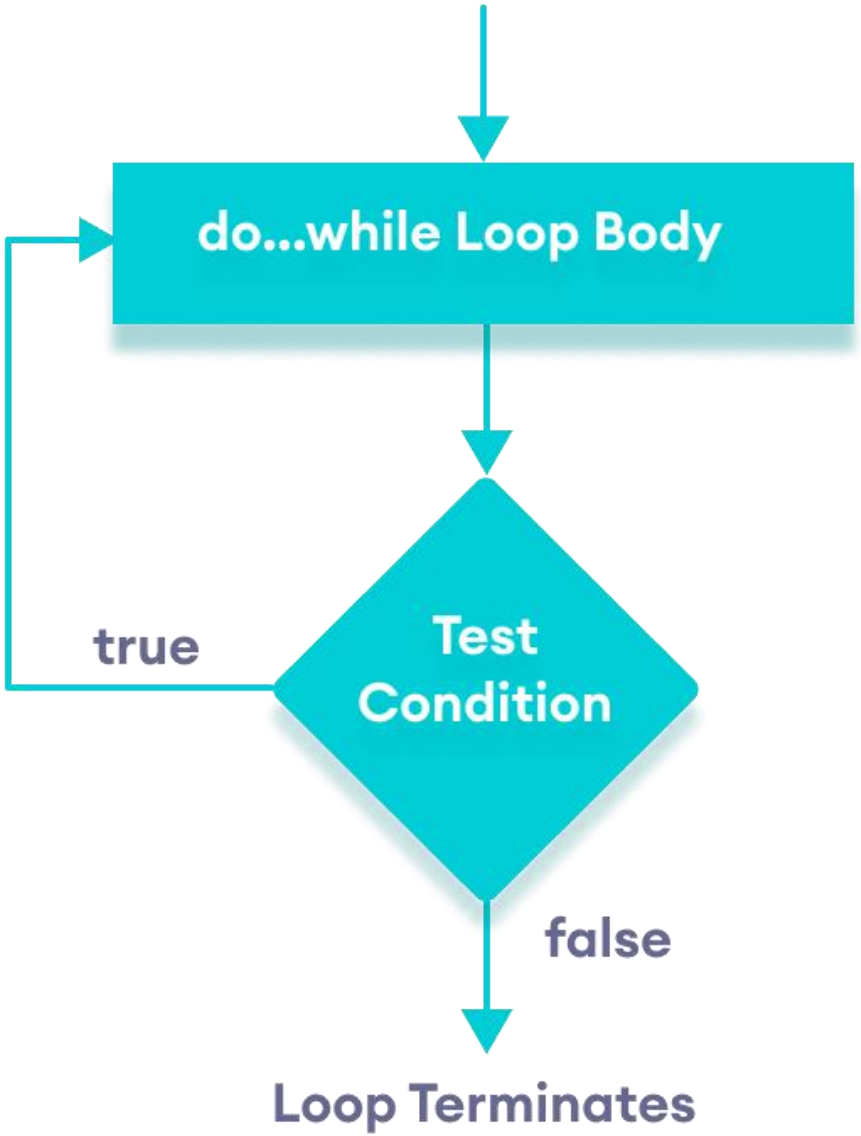
- The do…while loop is similar to while loop. However, the body of do…while loop is executed once before the test expression is checked. For example,

```
do {
    // body of loop
} while(textExpression)
```

- Here,

- The body of the loop is executed at first. Then the textExpression is evaluated.
- If the textExpression evaluates to true, the body of the loop inside the do statement is executed again.
- The textExpression is evaluated once again.
- If the textExpression evaluates to true, the body of the loop inside the do statement is executed again.
- This process continues until the textExpression evaluates to false. Then the loop stops.

Do while loop



C do while loop syntax

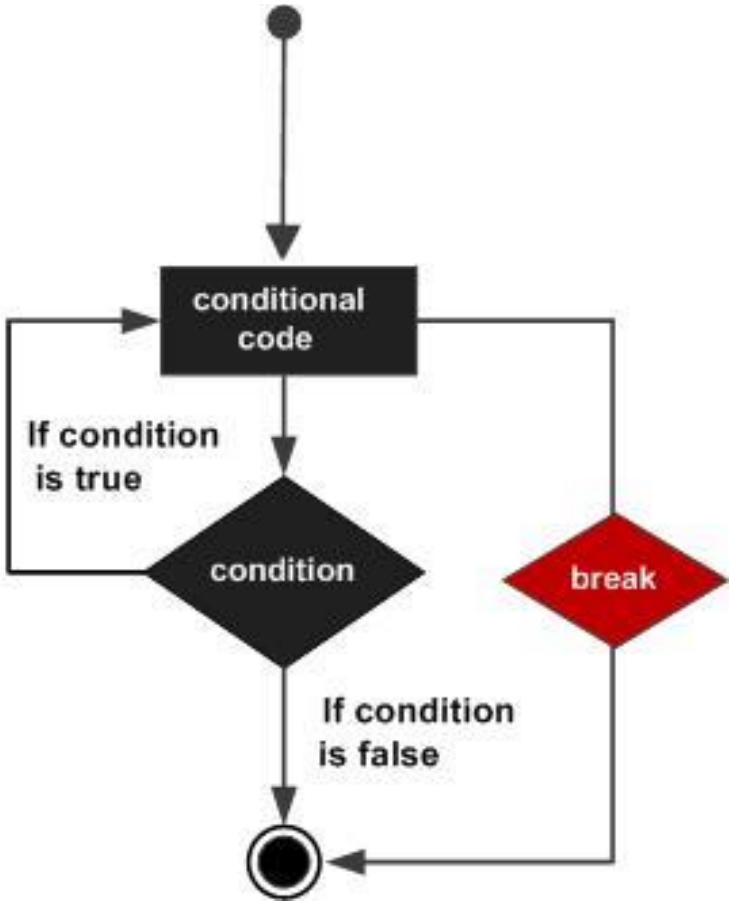
```
int x = 21;
do
{
    // The line will be printed even
    // if the condition is false
    System.out.println("Value of x:" + x);
    x++;
}
while (x < 20);
```

Loop Control Statements

- Loop control statements are used to change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.
- There are three different loop control statement in C
  - Break
  - Continue
  - Goto

The break statement

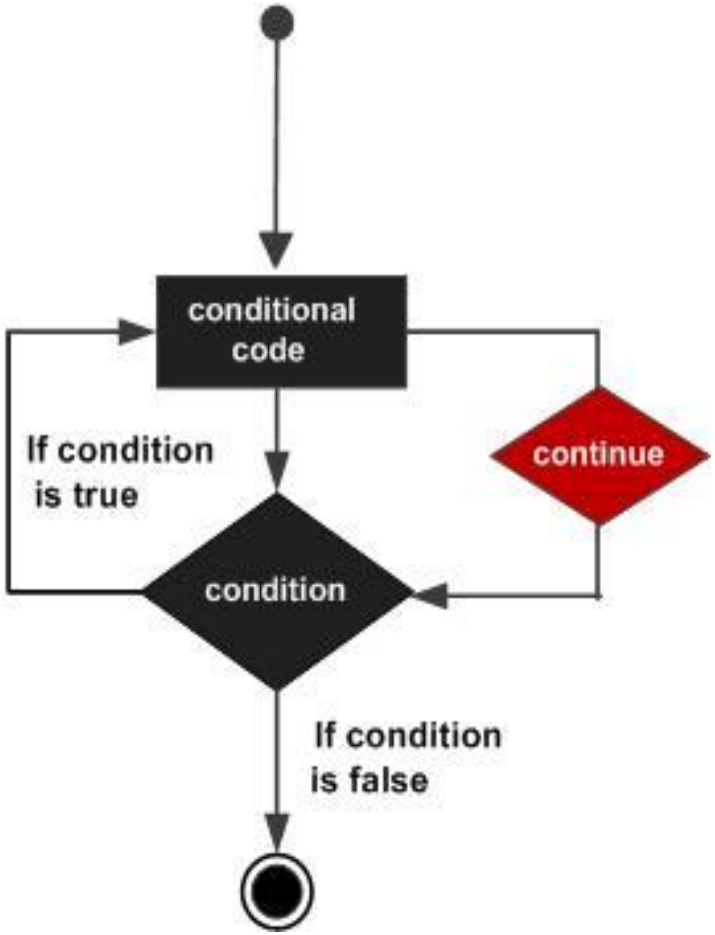
- It terminates the **loop** or **switch** statement and transfers execution to the statement immediately following the loop or switch.




```c
/* 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;
   }
}
```

The continue statement

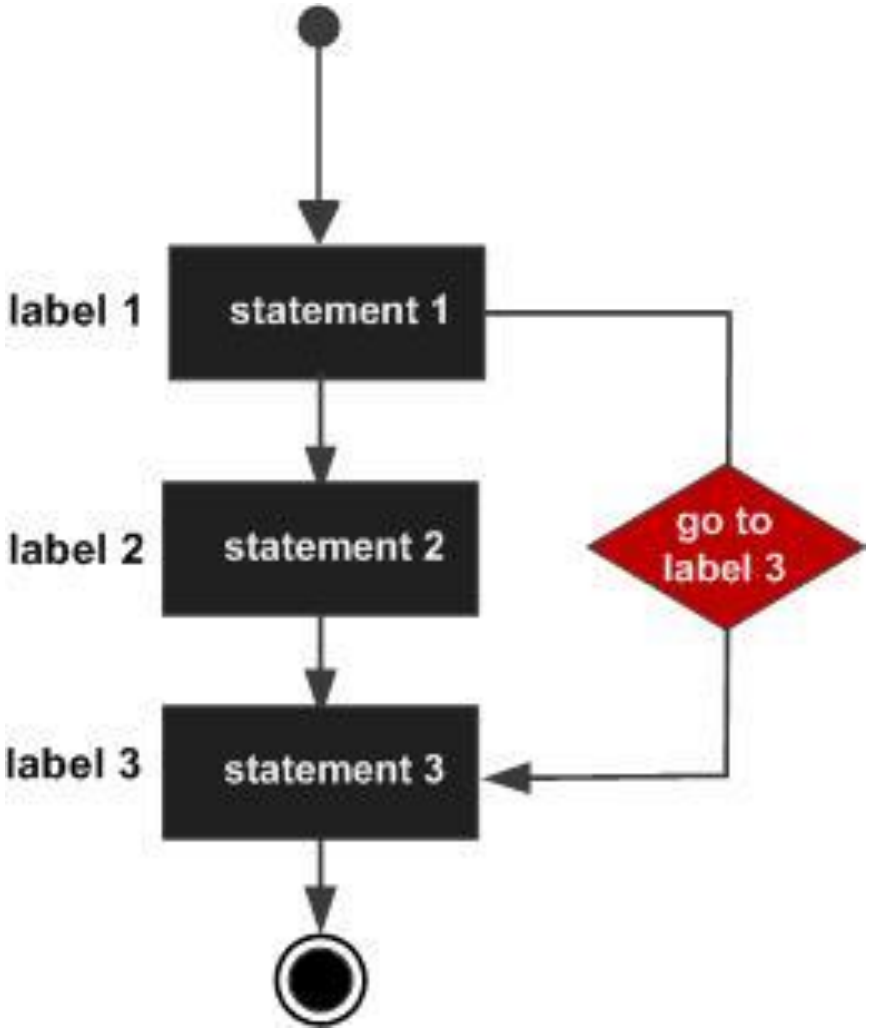
- Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating




```c
/* 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 );
```

The goto statement

- Transfers control to the labeled statement.




```
/* 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 );
```