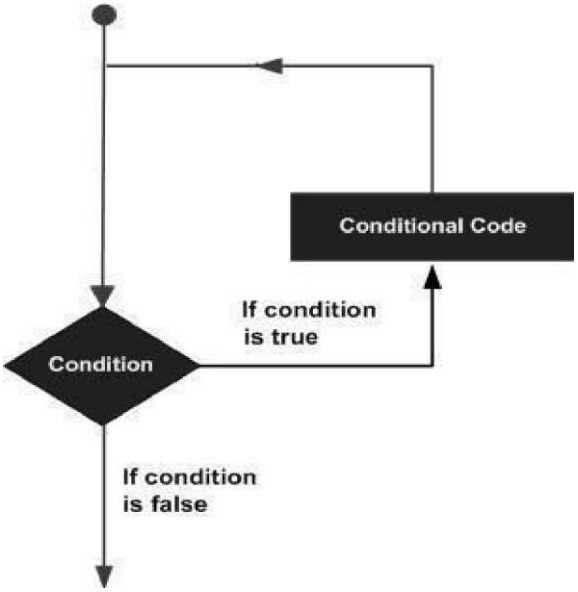
| CL-1002 Programming Fundamentals | LAB 04<br>ITERATIVE STATEMENTS IN C |
| --- | --- |

# Learning Objectives

1. Iteration & iterative statements in C
2. While loop
3. Do-while loop
4. For loop
5. Continue Statement
6. Break Statement
7. Nested Loops

## 1. Iteration and Types of Iterative C

By iteration or repetition, we mean doing the same task again and again. Usually, we want to repeat the task until some condition is met or for a predefined number of iterations. In computer science this is commonly referred as loop. Loops are used to repeat a statement, a block, a function or any combination of these. A typical looping workflow is shown in the figure.




Loops are very powerful and important tool in programming. Consider adding the salary of 1000 employees of a company. Or finding out how many people have been vaccinated from a list of 30 million people, that is just the population of Karachi what about whole Sindh or Pakistan? One way is to copy paste the code or repeating the lines of code. Well, the easier way is using a loop.

C programming language offers three kinds of iterative statements.

# Types of Loops

1. While Loop
2. Do-while Loop
3. For Loop

# 1. While Loop

The while loop is usually used when we don't know the number of iterations in advance.

**Application:** As a game developer you don't have any knowledge how many times user will play the game so you want to loop over the choice if they want to play again or not.

## a. C- Syntax

```
while ( loop repetition condition )
{

        Body of the loop.

 }
```

## b. Interpretation:

- The loop repetition condition (a condition to control the loop process) is tested; if it is true, the statement (loop body) is executed, and the loop repetition condition is retested.
- The statement is repeated as long as (while) the loop repetition condition is true. When this condition is tested and found to be false, the while loop is exited and the next program statement after the while statement is executed. If the condition is true forever the loop will run forever, we call such loop an infinite loop.
- It may not be executed at all if condition is false right from start.

## c. Example:

To count number of digits in a given integer.

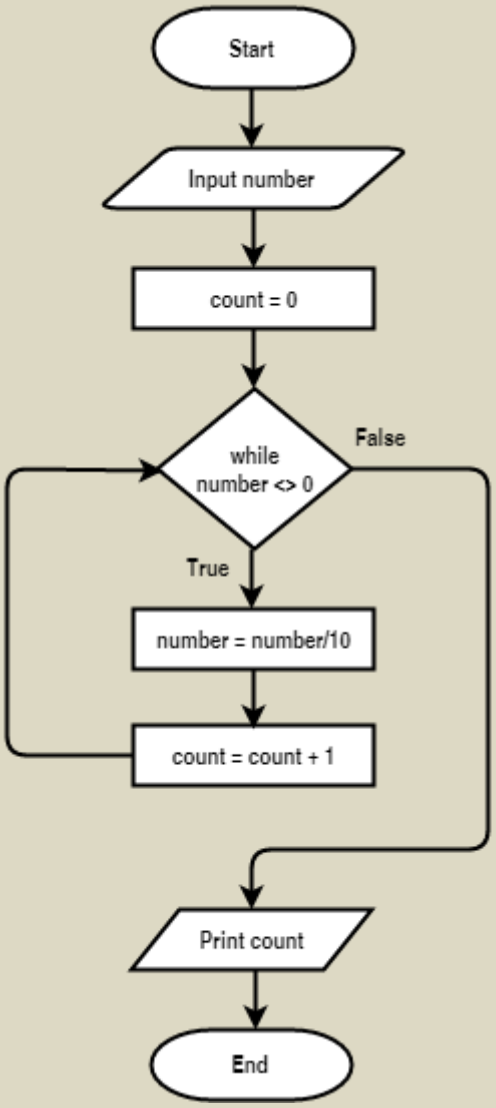
**Input:** 13542

**Output:** 5

## ALGORITHM

1. Start
2. Input number
3. count = 0
4. While number != 0 do
5. number=number/10
6. count = count + 1
7. While End
8. Print count
9. End

## FLOWCHART




## C-IMPLEMENTATION

```
#include <stdio.h>

    int main()
    {
        int number=0, count =0;
        scanf("%d",&number);
        while (number != 0)
        {       number = number/10;
                count = count + 1;
        }
        printf("The number of digits are: %d", count);
        return 0;
}
```

```
while ( loop repetition condition )
{
        //Statement_Block;
}
```

## 2. Do-while Loop

The do-while loop checks the loop repetition condition after running the body of the loop. This structure makes it most favorable in conditions where we are interested in running the body at least once.

**Application:** As a web-developer you are writing a registration application. You would like to make sure if the username is already taken or not. Therefore, you will keep asking for a unique username until provided, in such situations you would want the user to provide the username first and then perform the check.

### a. C- Syntax

```
do
{

        Body of the Loop;

} while ( loop repetition condition );
```

### b. Interpretation

- First, the body of the loop is executed.
- Then, the loop repetition condition is tested if it is true, the body is again and the condition is retested until it remains true the loop continues. When this condition is tested and found to be false, the loop is exited and the next statement after the do-while is executed.

### c. Example

Repeating the same example with do-while. Count number of digits in a given integer.

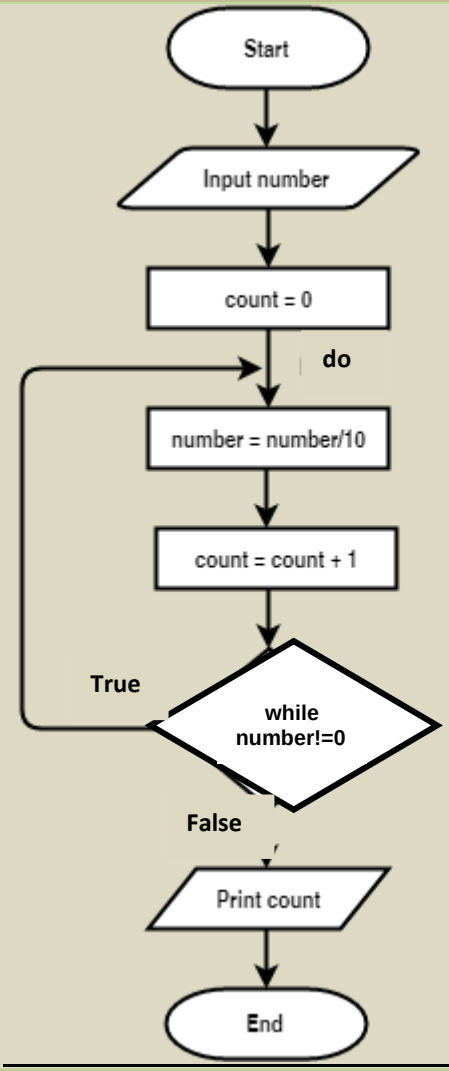
**Input:** 144452

**Output:** 6

## ALGORITHM

1. Start
2. Input number
3. count = 0
4. do
5. number=number/10
6. count = count + 1
7. while number != 0
8. Print count
9. End

## FLOWCHART




## C-IMPLEMENTATION

```
#include <stdio.h>

int main()
{
        int number=0, count =0;
        scanf("%d", &number);
        do
        {
                number = number/10;
                count = count + 1;
        } while (number != 0);

        printf("The number of digits are: %d", count);
        return 0;
}
```

```
do
{
        //Statement_Block;

} while ( loop repetition condition );
```

## 3. For Loop

The **For** loop is mostly used when we want to iterate for a specific number of times.

**Application:** You are developing a first-person shooter game in the beginning of the mission user is given 3 lives only, you would like the character to revive if killed for 3 times only.

### a. C- Syntax

```
for ( initialization expression ; loop repetition condition ; update expression )
{
        Body of the Loop;
}
```

### b. Interpretation

- The *initialization* is an assignment statement that is used to initialize the loop control variable with a value. This is the first statement to be executed in the loop and only run once.
- The *condition* is a relational expression that determines when the loop exits. This runs after initialization, and verified before every iteration.
- The update expression either *increment* or *decrement* the loop control variable on each iteration.

### c. Example

Print the numbers from 0 to desired value as shown below.

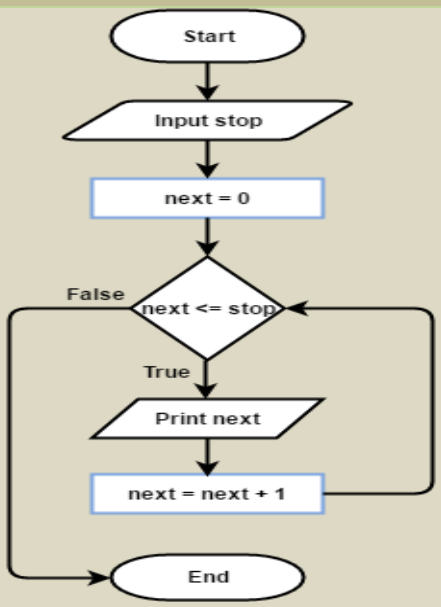
**Input:** 10

**Output:** 0 1 2 3 4 5 6 7 8 9 10

## ALGORITHM

1. Start
2. Input stop
3. FOR next = 0 to stop
4. Print next
5. next = next + 1
6. END FOR
7. End

## FLOWCHART




## C-IMPLEMENTATION

```
#include <stdio.h>

int main ()
{
int stop, next;
printf("Enter ending value:");
scanf("%d",&stop);

  for(next = 0 ; next <= stop ; next=next+1)


{
    printf("%d\t", next);
  }

  return 0;
}
```

```
for (initialization expression
;
loop repetition condition ;
update expression )
{
        //Statement_Block;
}
```

## OUTPUT

# Comparison of Loops

| | for | while | do-while |
|---|---|---|---|
| **Initialization of condition variable** | In the parenthesis of the loop. | Before the loop. | Before the loop or in the body of the loop. |
| **Test condition** | Before the body of the loop. | Before the body of the loop. | After the body of the loop. |
| **Updating the condition variable** | After the first execution. | After the first execution. | After the first execution. |
| **When to use** | for is generally used when there is a specific number of iterations | while is generally used when the number of iterations is not known in advance. | do-while is a loop with a post-condition. It is needed in cases, when the loop body is to be executed at least once. |

# 4. Continue Statement

The continue statement is very powerful in situations where we want to execute some portion of the loop body and skip a statement or block of statements. The control skips the loop body as it reaches the continue statement and starts the next iteration. Usually there is a condition for which we want to skip certain statements.

**Application:** As an AI developer you would work with a lot of datasets (e.g., thousands of images). Before training your AI model on these datasets you need to filter the corrupted or invalid images from the valid once. In such situations you can utilize continue on dependent situations instead of using multiple conditional statements.

## a. C Syntax

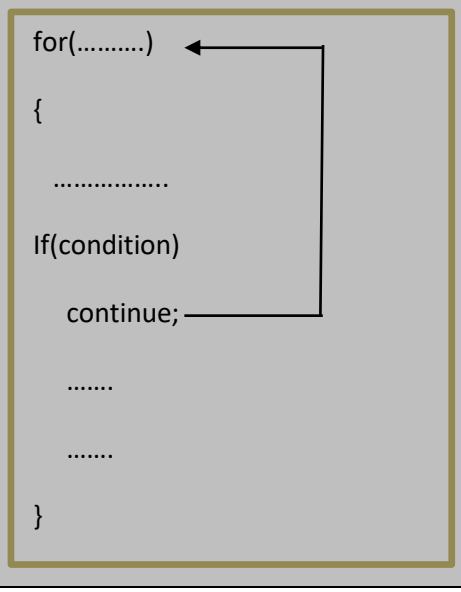
```
for (initialization; condition; increment/decrement) {

    block of statements;     // this block will execute always with each iteration of loop
    continue;
    block of statements;     // this block will be skipped.

}
```

**b. Example:**

```
#include<stdio.h>
int main()
{
   int i;
   for (int i=0; i<=10 ;i++)
   {
     if((i==3)||(i==7))
     {
          continue;
     }
      printf("The sum is %d", sum);
     }
   }
```




# 5. Break Statement

Break statement is used to exit the body of the loop without meeting the loop repetition condition. The statement after the break never gets executed and usually break is used to stop the loop before the loop termination condition is met. The controls execute the next statement after the loop body once it reaches the break statement in the loop body. Usually there is a condition upon which we want to exit the loop.

**Application:** You have written a fund-raising application for a cancer patient, after the required funds are collected, you would like to break out of the loop without knowing how many iterations have passed.

**a. C Syntax**

```
for (initialization; condition; increment/decrement) {

    block of statements;     // this block will execute always with each iteration of loop
    break;
    block of statements;     // this block never gets executed.

}
```

**b. Example**

```
#include<stdio.h>
int main()
{
   int a, sum=0;
   for(;  ;)
   {
     scanf("%d",&a);
     if(a==-999)
     {
     break;
     sum=sum+a;
     }
     }
printf("The sum is %d", sum);
```

```
for(…….)

{

  ……..

If(condition)

    break;

    …….

    …….

}
printf("…");
```

# Nested While Loop

Using While loop within while loops is said to be **Nested while loop**.

In nested while loop one or more statements are included in the body of the loop.

**IMPLEMENTATION IN C**

```
#include<stdio.h>

int main()

{

  int r,c,s;

  r=1;

  while(r<=5) /*outer loop*/

  {

   c=1;

   while(c<=2) /*inner loop*/

   {

    s=r+c;

    printf("r=%d c=%d sum=%d\n",r,c,s);

    c++;

   }

 printf("\n");

 r++;

}}
```

```
while (expr1)

        {

        :

                while (expr2)

                        {
                        :
                        Update expr2;
                }

                :

                        update expr1;
        }
```

## Nested Do While Loop

Using do While loop within do while loops is said to be **Nested while loop.**

In nested do while loop one or more statements are included in the body of the loop.

**IMPLEMENTATION IN C**

```
#include<stdio.h>

int main()

{

  int i=1,j=0,sum;

  do

  {

    sum=0;

    do

    {

      sum=sum+j;

      printf("%d",j);

      j++;

      if(j<=i)

      {

         printf("+");

       }

    }

     while(j<=i);

    printf("=%d\n",sum);

    j=1;

    i ++;

    }

    while(i<=10);}
```

```
do

    {

    :

        do

            {

            :

            Update expr;

        }while(expr);

        :

            update expr;

    }while(expr);
```

# Nested For Loop

- A for loop can contain any kind of statement in its body, including another for loop.
- The inner loop must have a different name for its loop counter variable so that it will not conflict with the outer loop.
- Nested loop: Loops placed inside one another, creating a loop of loops.

**IMPLEMENTATION IN C**

```
#include<stdio.h>

int main()

{



for (int i=1; i<=5;i++)

{

    for (int j=1;j<=i;j++)

    {

      printf("*");

      }

printf("\n");

 }
```

```
for ( expr1a; expr2a; expr3a )
{
                    :
  for ( expr1b; expr2b; expr3b )
  {
                              :
    }
                    :
}
```

# LAB 04 EXERCISES

**Problem 1:**
Write a program which will generate the Fibonacci series up to 10000. Also find the sum of the generated Fibonacci numbers divisible by 3, 5 or 7 only.
Example of Fibonacci series is: 1 1 2 3 5 8 13 25 …
Note: Do this task by using for loop DO NOT use arrays for this.

**Problem 2:**
In mathematics, the Least Common Multiple (LCM) and Greatest Common Divisor (GCD) are two important concepts used to find the smallest multiple of two or more numbers and the largest number that divides two or more numbers, respectively.

In this problem, your task is to write a program in C that takes two positive integers as input from the user, and then computes their LCM and GCD.

Input:
Two positive integers, let's say m and n (1 <= m, n <= 10^6)

Output:
The program should output two values, first being the LCM of the two numbers and second being the GCD of the two numbers.

Example:
Input: 15 25
Output: 75 5

In this example, the LCM of 15 and 25 is 75, and their GCD is 5.

**Problem 3:**
You are tasked to write a program that takes an integer value from the user and finds all the prime numbers till that range.

The program should use for loops to iterate from 2 till the given range and check for prime numbers. If the number is prime, then add it to the list of prime numbers.

Input:
The program takes a single integer as input, which represents the range till which the prime numbers are to be found.

Example 1:
Input: 10
Output: 2, 3, 5, 7

Example 2:
Input: 20
Output: 2, 3, 5, 7, 11, 13, 17, 19

Output:
The program should output a comma-separated list of all the prime numbers till the given range.

Constraints:
The input range should be an integer greater than or equal to 2.

**Problem 4:**
You are building a software program for a restaurant to keep track of its daily food waste. Write a program that inputs the food waste for each meal of the day (breakfast, lunch, and dinner) and then calculates and displays the total food waste for the day.

Input: 3 inputs representing the food waste for each meal of the day (breakfast, lunch, and dinner).

Output: The program should print the total food waste for the day.

Example:

Input:
Enter the food waste for breakfast: 10
Enter the food waste for lunch: 15
Enter the food waste for dinner: 20

Output:
Total food waste for the day: 45

**Problem 5:**
You are a software engineer at a financial company and you need to write a program in C to calculate the compound interest on an investment over several years. The program should ask the user for the initial investment amount, the interest rate, and the number of years for which the investment should grow. The program should then calculate and print out the total amount of the investment at the end of each year.

Input: The user inputs the initial investment amount (0 <= initial investment <= 10000), the interest rate (0 <= interest rate <= 1), and the number of years for which the investment should grow (1 <= number of years <= 30).

Output: The program should print the total amount of the investment at the end of each year.

Example:
Input:

Enter the initial investment amount: 1000
Enter the interest rate: 0.05
Enter the number of years: 5

Output:

Year 1: 1050.00
Year 2: 1102.50
Year 3: 1157.63
Year 4: 1215.51
Year 5: 1276.28

Explanation:
In the example, the user has invested $1000 with an interest rate of 5% for 5 years. The program calculates the total amount of the investment at the end of each year using the formula amount = initial investment * (1 + interest rate)^year.

**Problem 6:**
You are a software engineer at a weather station and you need to write a program in C to analyze the temperature data for a week. Your program should calculate the average temperature for each day, the highest temperature for the week, and the lowest temperature for the week.

Input: The user inputs the temperature readings for each day of the week (in degrees Fahrenheit). The temperature readings for each day are between -100 and 200 degrees Fahrenheit.

Output: The program should print the average temperature for each day of the week, the highest temperature for the week, and the lowest temperature for the week.

Example:

Input:

Enter the temperature for Monday: 75
Enter the temperature for Tuesday: 80
Enter the temperature for Wednesday: 70
Enter the temperature for Thursday: 85
Enter the temperature for Friday: 75
Enter the temperature for Saturday: 90
Enter the temperature for Sunday: 80

Output:
The average temperature for the week is: 78.5714
The highest temperature for the week is: 90
The lowest temperature for the week is: 70

**Problem 7:**
You are given a number n and you need to calculate the factorial of n. A factorial is the product of an integer and all the integers below it; e.g., factorial four ( 4! ) is equal to 4 * 3 * 2 * 1 = 24.

Input: The user inputs a positive integer n (1 <= n <= 12).

Output: The program should print the factorial of n.

Example:

Input:

Enter a positive integer: 4
Output:

The factorial of 4 is 24

**Problem 8:**
You have been hired by a company that specializes in creating fun and interactive games. Your task is to write a program for a matchstick game between the computer and a user. The rules of the game are as follows:

- There are 21 matchsticks placed in a pile.
- The computer asks the player to pick 1, 2, 3, or 4 matchsticks from the pile.
- After the player makes their pick, the computer makes its pick.
- The player who is forced to pick up the last matchstick loses the game.
- The program should ensure that the computer always wins the game by making the correct move in each turn.

Input: The user inputs the number of matchsticks they want to pick from the pile (1 <= number of matchsticks <= 4).

Output: The program should print the number of matchsticks left in the pile after each turn and who made the last pick.

**Problem 9:**
You are asked to write a program for a number guessing game. The game works as follows: the computer generates a random number between 1 and 100, and the user has to guess what the number is. The program should give the user hints about whether their guess is too high or too low, and the user has to keep guessing until they get the correct answer.

Input: The user inputs their guess for the number between 1 and 100.

Output: The program should print "Too low", "Too high", or "You won!" based on the user's guess and the generated number.

Example:
I have thought of a number between 1 and 100. Can you guess what it is? 50
Too low. Guess again: 75
Too high. Guess again: 60
Too low. Guess again: 65
You won!

Note: Use While Loop

**Problem 10**:

You are a teacher in a school and you have to grade the exams of your students. You need to write a program in C to calculate the average grade of the students based on their marks in 5 different subjects. The program should ask the user to input the marks for each student and then calculate the average grade for each student. Finally, the program should print out the average grade for each student.

Input: The user inputs the marks for each subject for each student (0 <= marks <= 100). The number of students can be between 1 and 50.

Output: The program should print the average grade for each student.

Example:
How many students would you like to grade? 2
Enter marks for student 1 in 5 subjects: 60 70 80 90 100
Enter marks for student 2 in 5 subjects: 80 90 70 60 50

Output:
The average grade for student 1 is 80.0
The average grade for student 2 is 70.0