**Fundamental of Computer Programing**

ASSIGNMENT#1

CONCEPT OF LOOP AND ITS TYPES

By: ABDUL GHAFFAR KALHORO

Class: BSCS-6C

Reg: No: 194699

# Concept of Loops in Programing:

Generally in programing, statements are executed sequentially. Different control structures are provided by Programming languages that leads us for more complicated execution paths. Among such those control structures, Loops have a lot of importance in programing languages. Loops are basic and useful in programming, that helps a programmer to execute any code repeatedly and can be controlled by adding condition.

In programing, loops are very significant to execute a statement or group of statements multiple times. Loops execute a number of dictations inside its body until some condition is true, but when the condition becomes false, it cause to end the loop.

# Types of Loops:

In programing we have three types of loops:

1. While loop
2. for loop
3. do…while loop

Loop Types

## 1) while loop:

It is a repetition statement in C programming which repeatedly executes a target statement as long as a given condition is true. It allows the programmer to specify that a program should repeat an action while some condition remains true. Basically while loop is alike to if selection in the case that their set of statements are executed whenever the condition is true. The main difference between them is that the while loop repeats the statement execution as long as the condition is true and if selection does not have the repetition feature.

The statement contained in the loop establish the body of the loop, which may be a single statement or a block within {curly braces}. When the condition become false, the repetition terminates and the first statement after the loop executes.

A while loop in C-Language is mainly expressed as:

while (condition test){

//repeating statement or group of statements

//Increment / decrement

}

More clearly the concept of while loop can be expressed in following diagram.



This given diagram is basically a flowchart, through which the concept of loop can be shown clearly.

### Structure of the while loop:

while loop can be used in programing by following these three steps:

step1:

Initialize counter variable with some value and then it has been tested for the condition.

step2:

Whenever the condition becomes true then the body of the while loop would executed otherwise control come out of the loop.

step3:

Counter is incremented or decremented then it has been tested again and again for the loop condition. It keeps on going until the condition returns false.

#### Example:

Program to print first 10 natural numbers using while loop in c language.

.

#include<stdio.h>

int main(void)

{

int number; //declaring the variable as integer.

number = 1; //initialization of variable

while (number <= 10) //loop condition

{

printf("%d\t", number);

number++; //increment

}

getchar();

}

Output

1 2 3 4 5 6 7 8 9 10

In this loops, we have two means of repetition:

1. Counter-controlled repetition
2. Sentinel-controlled repetition

### 1) Counter-Controlled Repetition:

In this case the number of repetitions needed for the loop are known before the beginning of loop. Counter controlled repetitions requires initialized control variable (loop counter), an increment or decrement statement and a condition used to terminate the loop (continuation condition). Every counter-controlled repetition has to exit after running a certain number of times. The count is hold in a variable called an index or counter. When the counter attains a certain value in which continuation condition becomes false then the loop will end.

* The number of repetitions is known before the loop begins executing, therefore the counter controlled repetition is usually called as definite repetition.

EXAMPLE: - In C Language, a definite while loop can be considered as an example of counter controlled loop as described below:

1. int sum;

2. sum = 0;

3. counter = 1;

4. while ( counter<= 10){

5.     sum = sum + n\*n;

6.      counter = counter+ 1;

7.     }

### 2) Sentinel-Controlled Repetition:

In this case the number of execution of the loop is unknown. It is a repetition procedure for solving a problem by using a sentinel value (also called a signal value, a dummy value or a flag value) that indicates "end of data". The signal value itself is not a part of the processed data.

* Sentinel-controlled repetition is often called indefinite repetition because it is not known in advance how many times the loop will be executed.

Example: - An example of when we would use sentinel-controlled repetition is when we are writing a program in which we do not know in advance, where it would end.

## 2) for loop:

As that of the while loop, for loop is used to perform some operations repeatedly. The while loops are meant to be used when we do not know how many times we shall need to repeat an action, some condition will change in the loop to terminate it. If you know exactly that how many iterations you will be making then the for loop is the better choice for that program, for that reason for loop is considered as a counter controlled repetition.

Basic layout of the for loop in c language is shown below:

for (initialization; condition; increment/decrement)

{

//body

}

for loop can be used as an infinite loop by defining the condition always true or by simply writing the loop as empty. ( i-e, for(;;) )

#### Structure of the for loop:

In for loop, two braces are used in its statement, and there are three parts inside the parenthesis of the for loop statement which are discussed below:

* Parrt\_1: The first part this loop contains initialization in which loop counter variable is initialized to starting value or previously initialized variable is used.
* Part\_2: Second part of loop is its middle part in which loop condition is used which determines when the loop terminates.
* Part\_3: In third part counter variable is incremented or decremented.

The use of for loop can be shown by following given example:

#### Example:

Program to print even numbers from 4 to 16 using for loop in c language.

#include <stdio.h>

#include<conio.h>

int main() {

// variable declaration.

int myNumber;

printf("Even numbers from 4 to 16 are:\n");

// for loop with repeating condition.

for (myNumber = 4; myNumber < 17; myNumber +=2)

printf("%d\t", myNumber);

\_getch();

return 0;

}

Output

Even numbers from 4 to 16 are:

4 6 8 10 12 14 16

## 3) do…while loop:

There may exist some situations in which it is necessary to execute body of the loop before testing the condition. Such situations can be handled with the help of do-while loop. In ‘do’ statements the body of the loop evaluate first, then the condition is checked for repetition of statements using ‘while’ statements.

It is often used as definite controlled repetition, however it can be used as infinite loop when the condition is always true. do…while loop is quite different from the other two loops (i.e. while and for loops), because this statement runs the code first and then checks the condition, so there is always a guarantee that the code runs at least once.

The general representation of do…while loop in c-language is:

do

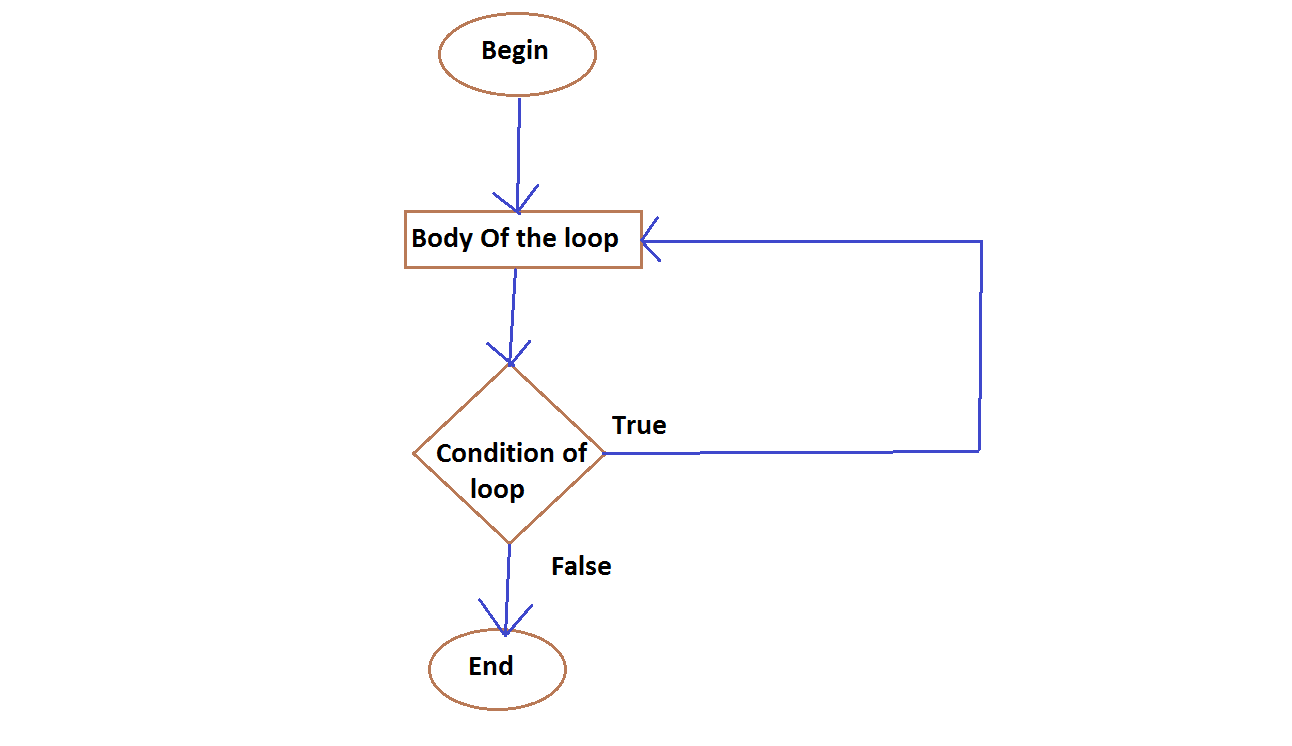
{

//statements to be executed 1st and then to be repeated.

}

while(condition)

It can be better expressed in terms of flowchart as under:



do…while loop

#### Structure:

In C language, do…while loop is used in same manner as that of while loop and has the structure almost same as that of while loop, but in do…while loop the syntax ‘do’ is used before parenthesis. In braces the block of statements is used to be repeated after 1st execution. And while loop is used after middle braces for repeating purpose whose counter controlled variable is incremented or decremented within while statement or in the body of the while loop. Which is shown in following example.

#### Example:

A program in C language to print first 10 multiples of 3 using do…while loop.

#include<stdio.h>

#include<conio.h>

void main()

{

int digit\_1, digit\_2;

digit\_1 = 3;

digit\_2 = 1;

do

{

printf("%d \t", digit\_1\*digit\_2);

digit\_2++;

} while (digit\_2 <= 10);

\_getch();

}

Output

3 6 9 12 15 18 21 24 27 30