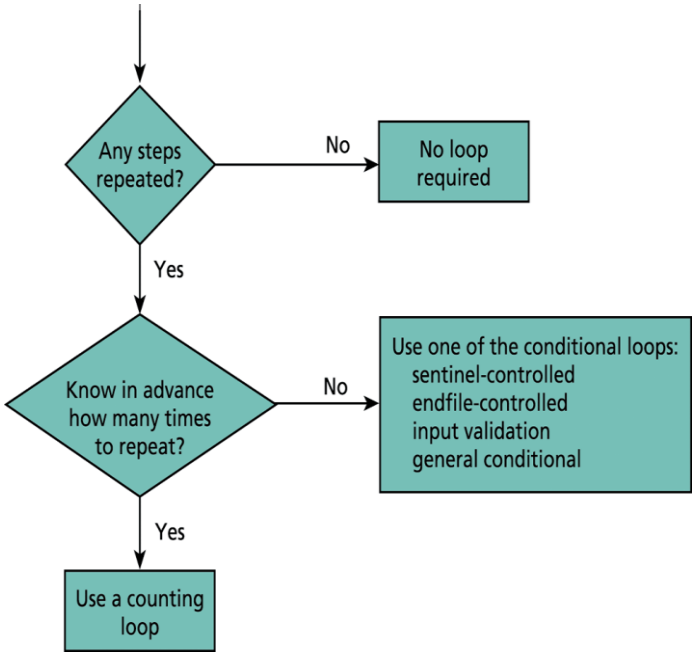
CONTROL STATEMENTS:LOOPING–REPETITION

- Iteration and repetitive execution (for, while, do-while), nested loops.

- ***Loop-****loop is a control structure that repeats a group of any kind of statements in a program.*
- ***Loop body-*** *The statements that are repeated in the loop.*

Looping Constructs in C:

- A set of statements may have to be executed number of times or till the condition is satisfied are called Looping Constructs.
- The statements that help us to execute a set of statements repeatedly are called Looping Constructs.
- It is implemented using the following statements.
  - ✓ for
  - ✓ while
  - ✓ do-while
- The loop will have to perform 3 tasks
  - ✓ Initialization ***: Loop Initialization.***
  - ✓ Condition checking ***: Loop Condition.***
  - ✓ Incrementation / decrementation ***: Loop Updating.***
- The difference between 3 looping control is the order in which they are represented.
- ***Three questions to determine whether loops will be required in the general algorithm:***
  1. Are there any steps repeated while solving the problem? If so, which ones?
  2. Do we know in advance how many times these steps are repeated?
  3. Do we know the condition until when we keep repeating these steps**?**



Types of looping constructs

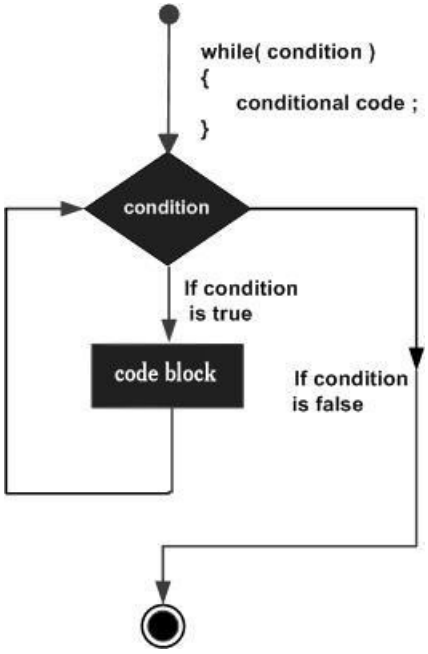
Kind	C Implementation Structure
Pre tested loops	while
Post tested loops	do-while
Count controlled loop	while , for , do- while
Event controlled loop: Sentinel-controlled loop	while , for
Event controlled loop: EndOfFile – controlled loop	while , for , do-while
Event controlled loop: Input validation loop	do-while
General conditional loop	while, for , do-while

The while loop

Syntax:

```
Statement1;
Statement2;
initialization;
while(Condition check)
{
Statement3;
Statement4;
incrementation;
}
Statement5;
```

Flowchart:



Explanation:

In case of while loop, the Initialization, Condition check and Incrementation is represented in the separate statements.

- First Initialization of the loop is performed.
- Next condition is checked. If condition is true then the control enters the body of the loop and statements are executed.
- Next Incrementation of the loop is performed, which is represented in loop body after which again the control goes back to the condition check, if condition is true, the body is executed again.

- This process repeats as long as the condition evaluates to be true. Once the condition evaluates to be false, the control flows to the statement outside the body of the loop.

Consider the following program:

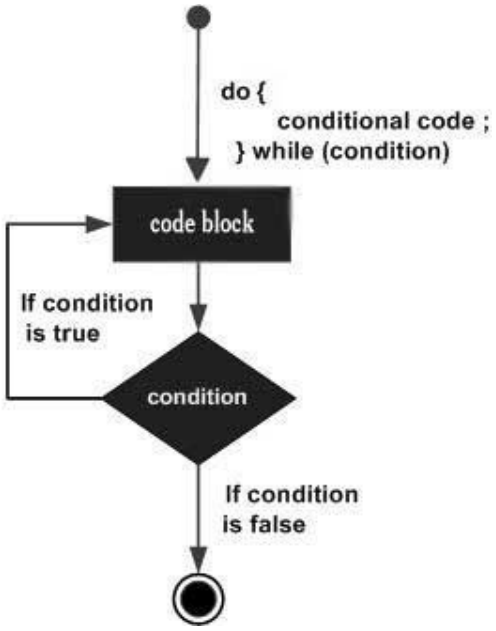
Statement1;
Statement2;
initialization;
while(**Condition check**)
{
Statement3;
Statement4;
incrementation;
}
Statement5;
	#include<stdio.h>
	void main()
	{
	int i;
	i=1;
	while(i<=5)
	{
	printf("INDIA\n");
	i++;
	}
	printf("is our country\n");
	getch();
	}
		OUTPUT:
		INDIA
		INDIA
		INDIA
		INDIA
		INDIA
		is our country

The do *while* loop

Syntax:

```
Statement1;
Statement2;
initialization;
do
{
Statement3;
Statement4;
incrementation;
}
while(condition check);
Statement5;
```

Flowchart:



Explanation:

In case of for do-while, the Initialization, Condition check and Incrementation is done in the separate statement.

- First Initialization of the loop is performed.
- Next condition is not checked. Rather body of the loop is entered and statements are executed.

- Next Incrementation is performed after which condition is checked, if condition is true, the control re-enters the body and executes the statements again.
- This process repeats as long as the condition evaluates to be true. Once the condition evaluates to be false, the control flows to the statement outside the body of the loop.

Consider the following program:

Statement1;
Statement2;
initialization;
do
{
Statement3;
Statement4;
incrementation;
}
while(**Condition check**);
Statement5;
	#include<stdio.h>
	void main()
	{
	int i;
	do
	{
	printf("INDIA\n");
	i++;
	}
	while(i<=5);
	printf("is our country\n");
	getch();
	}
		OUTPUT:
		INDIA
		INDIA
		INDIA
		INDIA
		INDIA
		is our country

"What are the Entry controlled and Event/ Exit controlled loops in C?"

Sl.No.	Entry controlled loops	Exit controlled loops
1	Entry Controlled will check the Condition at First and doesn't execute if it is False	Exit Controlled will check the Condition at Last and at least once the statement will execute though it is False
2	Entry control is otherwise called as WHILE loop, because the while loop checks the condition at first, and then only execute the following instructions.	Exit control is also called as DO WHILE loop, because the do while loop checks the condition at last
3	In Entry controlled loop the test condition is checked first and if that condition is true than the block of statement in the loop body will be executed	Exit controlled loop the body of loop will be executed first and at the end the test condition is checked, if condition is satisfied than body of loop will be executed again.
4	Entry control is otherwise called as WHILE loop, because the while loop checks the condition at first, and then only execute the following instructions	Exit control is also called as DO WHILE loop, because the do while loop checks the condition at last

Programs based on while loop -

1. **Write a program to reverse a given number**

```
#include<stdio.h>
void main()
{
   int num=1234,rnum=0,rem;
   while (num!=0)
   {
      rem = num%10;
      rnum = rnum*10+rem;
      num = num/10;
   }
  printf("reversed num = %d",rnum);
}
```

2. **Write a program to find Gcd and Lcm**

```
#include<stdio.h>
void main()
{
   int n=15,m=25,gcd;
   while (n!=m)
   {
    if (n>m)
        n = n-m;
    else
        m = m –n;
   }
  gcd = n;
   printf("gcd = %d",gcd);
}
```

3. **Design a program that calculates the average exam grade for a class of students using Sentinel controlled while loop.**

```
#include <stdio.h>
        int main ( )
        {
                int  counter, grade, total, average ;
                total = 0 ;
                counter = 1 ;
                printf("Enter a grade: ") ;
                scanf("%d", &grade) ;
                        while (grade != -1)
                        {
                            total = total + grade ;
                            counter = counter + 1 ;
                            printf("Enter another grade: ") ;
                            scanf("%d", &grade) ;
                        }
                average = total / counter ;
                printf ("Class average is: %d\n", average) ;
                return 0 ;
        }
```

4. Input validation: do – while loop

```
#include <stdio.h>
 void main ( )
{
        int  empno ;
        float  bookamt, folderamt, comm;
        char  choice;
    do
    {
    printf("enter employee number:");
    scanf("%d", &empno );
    printf(" \nEnter total sales  amount of  books and folders in Rs.\n ");
    scanf("%f %f",&bookamt,&folderamt);
    comm = (bookamt*0.1 + folderamt * 0.15);
    printf("\n Commission earned =  Rs.%.2f",comm);
    printf("\n Do you want to continue  Y/N?");
    ch = getchar();
    }
        while (choice == 'y' || choice == 'Y');
}
```

Output:

```
Enter employee number   :   1500
Enter total sales amount of books and folders in Rs. 2100   1700
Commission earned = Rs. 465
Do you want to continue Y/N?  N
```

Note:

Loop Type	Description
while loop	Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
for loop	Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.
do while loop	It is more like a while statement, except that it tests the condition at the end of the loop body.
Nested Loops	You can use one or more loops inside any other while, for, or do..while loop.