while loop in C

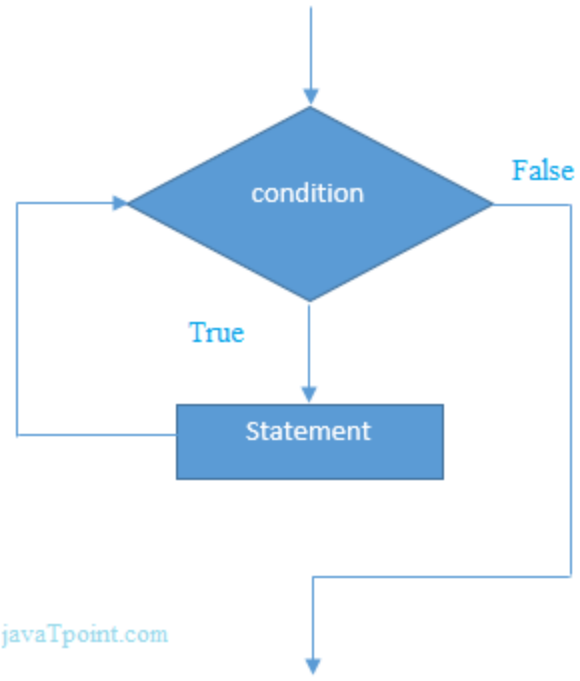
While loop is also known as a pre-tested loop. In general, a while loop allows a part of the code to be executed multiple times depending upon a given boolean condition. It can be viewed as a repeating if statement. The while loop is mostly used in the case where the number of iterations is not known in advance.

Syntax of while loop in C language

The syntax of while loop in c language is given below:

```
1. while(condition){
2. //code to be executed
3. }
```

Flowchart of while loop in C



Example of the while loop in C language

Let's see the simple program of while loop that prints table of 1.

```c
#include<stdio.h>
int main(){
int i=1;
while(i<=10){
printf("%d \n",i);
i++;
}
return 0;
}
```

Output

```
1
2
3
4
5
6
7
8
9
10
```

Program to print table for the given number using while loop in C

```c
#include<stdio.h>
int main(){
int i=1,number=0,b=9;
printf("Enter a number: ");
scanf("%d",&number);
while(i<=10){
printf("%d \n",(number*i));
i++;
}
return 0;
}
```

Output

```
Enter a number: 50
50
100
150
200
250
300
350
400
450
500
Enter a number: 100
100
200
300
400
500
600
700
800
900
1000
```

Properties of while loop

- A conditional expression is used to check the condition. The statements defined inside the while loop will repeatedly execute until the given condition fails.
- The condition will be true if it returns 0. The condition will be false if it returns any non-zero number.
- In while loop, the condition expression is compulsory.
- Running a while loop without a body is possible.
- We can have more than one conditional expression in while loop.
- If the loop body contains only one statement, then the braces are optional.

Example 1

```
1. #include<stdio.h>
2. void main ()
3. {
4.     int j = 1;
5.     while(j+=2,j<=10)
6.     {
```

```
7.          printf("%d ",j);
8.      }
9.      printf("%d",j);
10. }
```

Output

```
3 5 7 9 11
```

Example 2

```
1.  #include<stdio.h>
2.  void main ()
3.  {
4.      while()
5.      {
6.          printf("hello Javatpoint");
7.      }
8.  }
```

Output

```
compile time error: while loop can't be empty
```

Example 3

```
1.  #include<stdio.h>
2.  void main ()
3.  {
4.      int x = 10, y = 2;
5.      while(x+y-1)
6.      {
7.          printf("%d %d",x--,y--);
8.      }
9.  }
```

Output

```
infinite loop
```

Infinitive while loop in C

If the expression passed in while loop results in any non-zero value then the loop will run the infinite number of times.

```
while(1){
//statement
}
```

for loop in C

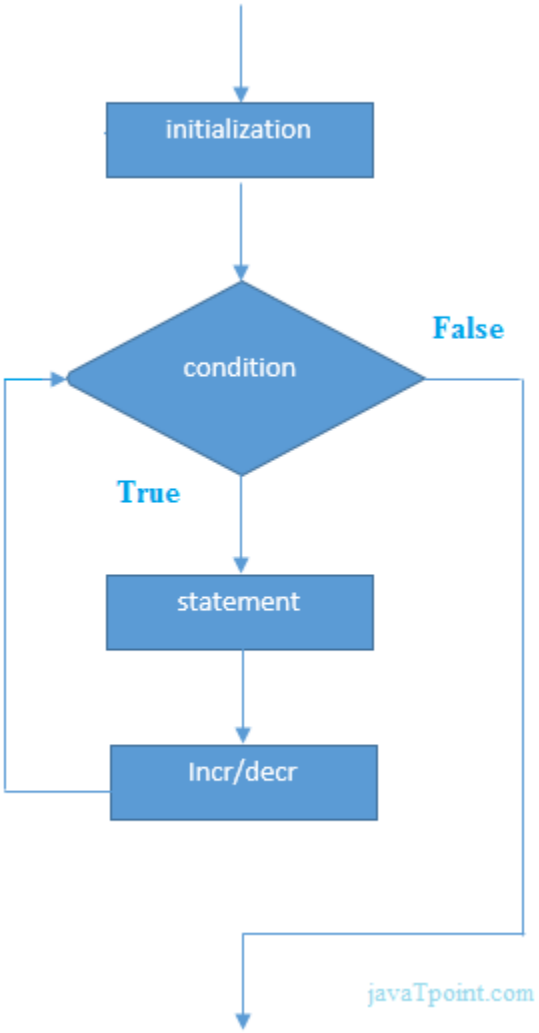
The **for loop in C language** is used to iterate the statements or a part of the program several times. It is frequently used to traverse the data structures like the array and linked list.

Syntax of for loop in C

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

```
for(Expression 1; Expression 2; Expression 3){
//code to be executed
}
```

Flowchart of for loop in C



C for loop Examples

Let's see the simple program of for loop that prints table of 1.

```
1. #include<stdio.h>
2. int main(){
3. int i=0;
4. for(i=1;i<=10;i++){
5. printf("%d \n",i);
6. }
7. return 0;
```

```
}
```

Output

```
1
2
3
4
5
6
7
8
9
10
```

C Program: Print table for the given number using C for loop

```
#include<stdio.h>
int main(){
int i=1,number=0;
printf("Enter a number: ");
scanf("%d",&number);
for(i=1;i<=10;i++){
printf("%d \n",(number*i));
}
return 0;
}
```

Output

```
Enter a number: 2
2
4
6
8
10
12
14
16
18
20
Enter a number: 1000
1000
2000
3000
4000
5000
6000
```

```
7000
8000
9000
10000
```

Properties of Expression 1

- The expression represents the initialization of the loop variable.
- We can initialize more than one variable in Expression 1.
- Expression 1 is optional.
- In C, we can not declare the variables in Expression 1. However, It can be an exception in some compilers.

Example 1

```
1. #include <stdio.h>
2. int main()
3. {
4.     int a,b,c;
5.     for(a=0,b=12,c=23;a<2;a++)
6.     {
7.         printf("%d ",a+b+c);
8.     }
9. }
```

Output

```
35 36
```

Example 2

```
1. #include <stdio.h>
2. int main()
3. {
4.     int i=1;
5.     for(;i<5;i++)
6.     {
7.         printf("%d ",i);
8.     }
```

```
}
```

Output

```
1 2 3 4
```

Properties of Expression 2

- Expression 2 is a conditional expression. It checks for a specific condition to be satisfied. If it is not, the loop is terminated.
- Expression 2 can have more than one condition. However, the loop will iterate until the last condition becomes false. Other conditions will be treated as statements.
- Expression 2 is optional.
- Expression 2 can perform the task of expression 1 and expression 3. That is, we can initialize the variable as well as update the loop variable in expression 2 itself.
- We can pass zero or non-zero value in expression 2. However, in C, any non-zero value is true, and zero is false by default.

Example 1

```
#include <stdio.h>
int main()
{
    int i;
    for(i=0;i<=4;i++)
    {
        printf("%d ",i);
    }
}
```

output

```
0 1 2 3 4
```

Example 2

```
#include <stdio.h>
int main()
```

```
{
    int i,j,k;
    for(i=0,j=0,k=0;i<4,k<8,j<10;i++)
    {
        printf("%d %d %d\n",i,j,k);
        j+=2;
        k+=3;
    }
}
```

Output

```
0 0 0
1 2 3
2 4 6
3 6 9
4 8 12
```

Example 3

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

Output

```
infinite loop
```

Properties of Expression 3

- Expression 3 is used to update the loop variable.
- We can update more than one variable at the same time.
- Expression 3 is optional.

Example 1

```c
#include<stdio.h>
void main ()
{
    int i=0,j=2;
    for(i = 0;i<5;i++,j=j+2)
    {
        printf("%d %d\n",i,j);
    }
}
```

Output

```
0 2
1 4
2 6
3 8
4 10
```

Loop body

The braces {} are used to define the scope of the loop. However, if the loop contains only one statement, then we don't need to use braces. A loop without a body is possible. The braces work as a block separator, i.e., the value variable declared inside for loop is valid only for that block and not outside. Consider the following example.

```c
#include<stdio.h>
void main ()
{
    int i;
    for(i=0;i<10;i++)
    {
        int i = 20;
        printf("%d ",i);
    }
}
```

Output

```
20 20 20 20 20 20 20 20 20 20
```

Infinitive for loop in C

To make a for loop infinite, we need not give any expression in the syntax. Instead of that, we need to provide two semicolons to validate the syntax of the for loop. This will work as an infinite for loop.

```
1. #include<stdio.h>
2. void main ()
3. {
4.     for(;;)
5.     {
6.         printf("welcome to javatpoint");
7.     }
8. }
```