Bismillahhir Rahmanir Rahim

Welcome

Hosted and prepared by

MD. SOROWAR MAHABUB RABBY

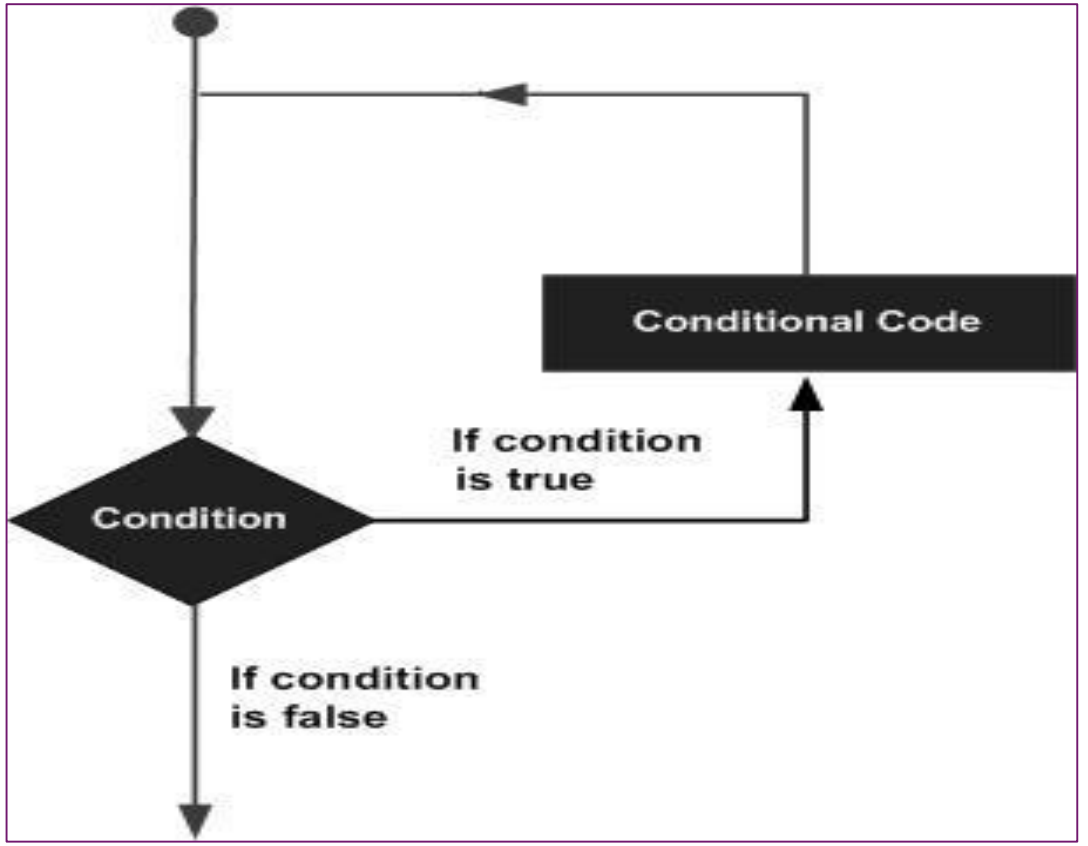
DEPARTMENT OF CSE (COMPUTER SCIENCE AND ENGINEERING) , IIUC

CELL: 01834756433, 01521564157

E-MAIL: sorowarmahabub1709vip@gmail.com

Loop
in C Language

The looping can be defined as repeating the same process multiple times until a specific condition satisfies. There are three types of loops used in the C language. In this part, we are going to learn all the aspects of C loops.



Advantage of loops in C

1) It provides code reusability.

2) Using loops, we do not need to write the same code again and again.

3) Using loops, we can traverse over the elements of data structures (array or linked lists).

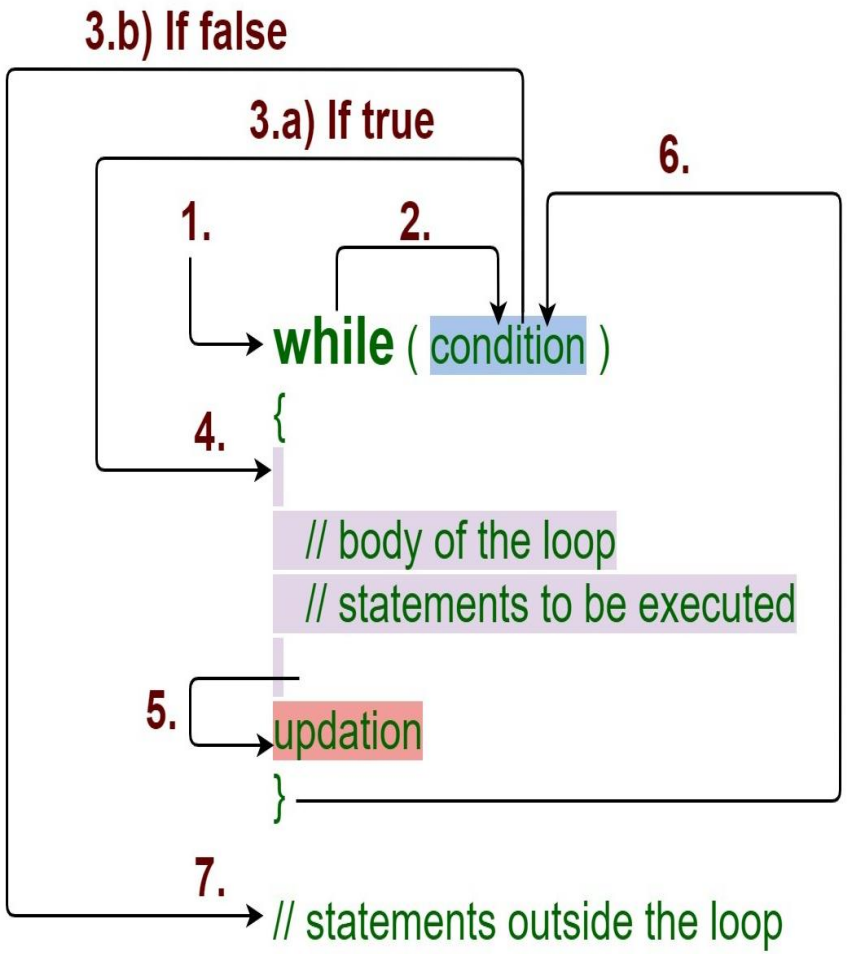
Types of loop

There are 3 types of Loop in C language, namely:

1.while loop

2.for loop

3.do while loop


3.b) If false
3.a) If true
6.
1.
2.
while (condition)
{
4.
// body of the loop
// statements to be executed
5.
updation
}
7.
// statements outside the loop

while

loop

while loop Intro

while loop can be addressed as an **entry control** loop. It is completed in 3 steps.

- Variable initialization. ***(e.g int x = 0;)***
- condition ***(e.g while(x <= 10))***
- Variable increment or decrement ***(x++ or x-- or x = x + 2)***

while loop Syntax

```
variable initialization;
while(condition) {
    statements;
    variable increment or decrement;
}
```

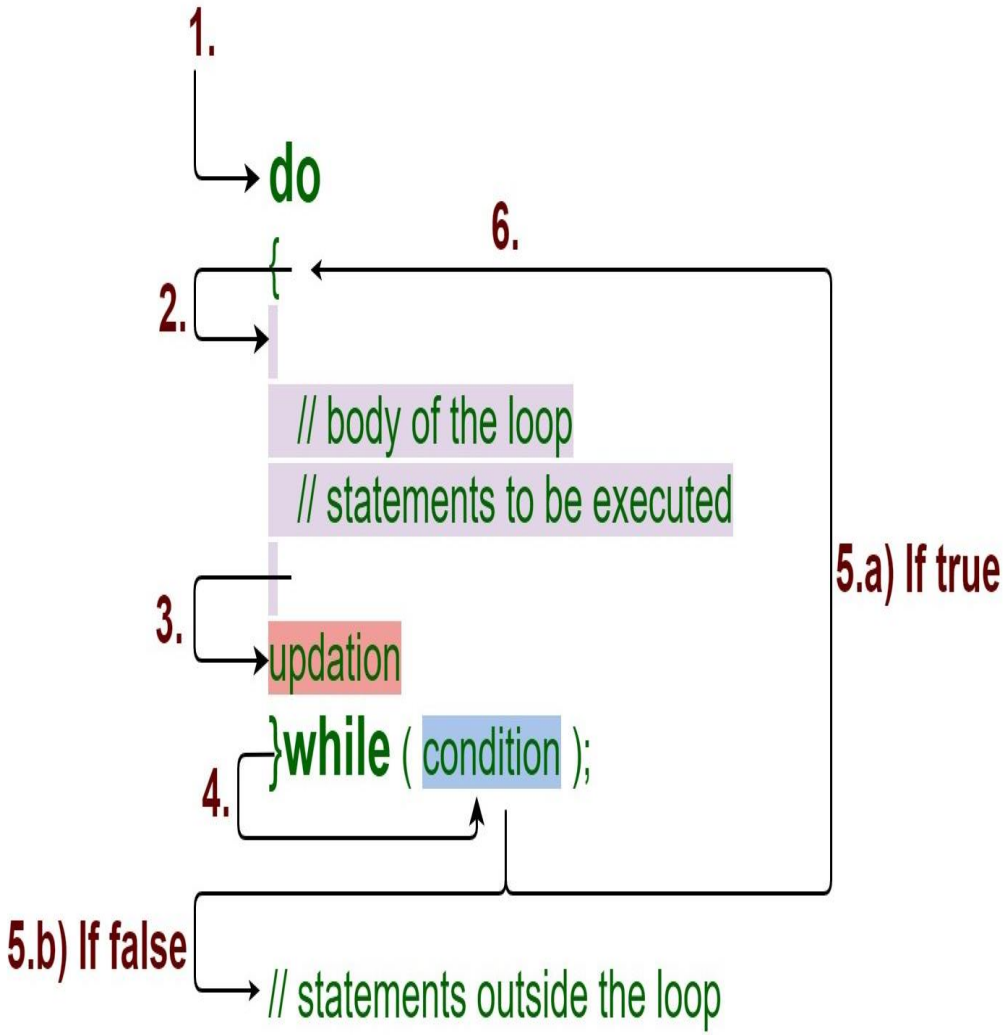
while loop Example 1

```c
#include<stdio.h>
#include<conio.h>
int main()
{
    int num=1;    //initializing the variab
    while(num<=10) //while loop with con
    {
        printf("%d\n",num);
        num++;          //incrementing operat
    }
    return 0;
}
```

1
2
3

Output:

1
2
3
4
5
6
7
8
9
10



do while loop

While & do while loop

While

```
int i = 0;
while(i > 0)
{
    printf("%d", i);
    i--;
}
```

do-While

```
int i = 0;
do
{
    printf("%d", i);
    i--;
} while(i > 0);
```

do while loop Intro

In some situations 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. do statement evaluates the body of the loop first and at the end, the condition is checked using while statement.

do while loop Syntax

It means that the body of the loop will be executed at least once, even though the starting condition inside **while is initialized to be false. General syntax is,**

```
do {
     ..... .....
} while(condition)
```

do while loop Example 1

```c
#include<stdio.h>
#include<conio.h>
int main()
{
    int num=1;
    do //do-while loop
    {
        printf("%d\n", 2*num);
        num++;
    }while(num<=10);
    return 0;
}
```

Output:

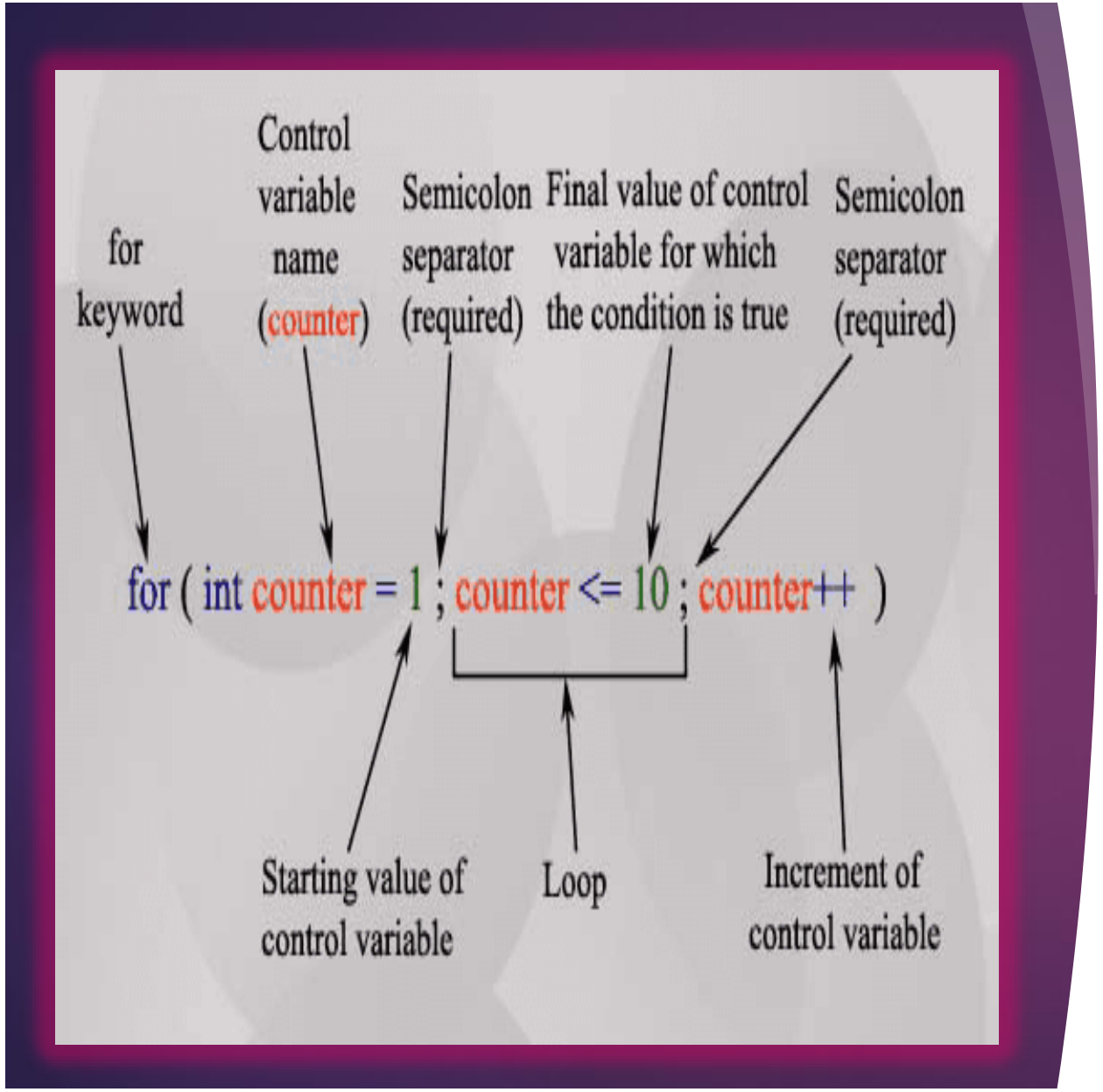
2
4
6
8
10
12
14
16
18
20

do while loop Example 2

```
//Example: Program to print first 10 multiples of 5.
#include<stdio.h>
void main() {
int a, i;
a = 5; i = 1;
do {
      printf("%d\t", a*i);
      i++;
} while(i <= 10);
}
```

Output:

5 10 15 20 25 30 35
40 45 50


for keyword
Control variable name (counter)
Semicolon separator (required)
Final value of control variable for which the condition is true
Semicolon separator (required)
for (int counter = 1 ; counter <= 10 ; counter++)
Starting value of control variable
Loop
Increment of control variable

for

loop

for loop Intro

In for loop we have exactly two semicolons, one after initialization and second after the condition. In this loop we can have more than one initialization or increment/decrement, separated using comma operator. But it can have only one **condition**.

The for loop is executed as follows:

1.It first evaluates the initialization code.
2.Then it checks the condition expression.
3.If it is **true**, it executes the for-loop body.
4.Then it evaluate the increment/decrement condition and again follows from step 2.
5.When the condition expression becomes **false**, it exits the loop.

for loop Syntax

for loop is used to execute a set of statements repeatedly until a particular condition is satisfied. We can say it is an **open ended loop.**. General format is,

```
for(initialization; condition; increment/decrement)
{
        statement-block;
}
```

for loop Example 1

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

Output:

1
2
3
4
5
6
7
8
9
10

for loop Example 2

```
// Example: Program to print first 10 natural numbers

#include<stdio.h>
void main( ) {
int x;
for(x = 1; x <= 10; x++) {
     printf("%d\t", x);
}
 return 0:
}
```

Output:

1 2 3 4 5 6 7 8 9 10

Nested for loop Syntax

We can also have nested for loops, i.e one for loop inside another for loop. Basic syntax is,

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

nested for loop Example

```
//Example: Program to print half Pyramid of numbers

#include<stdio.h>
void main( ) {
int i, j;
/* first for loop */
for(i = 1; i <= 5; i++) {
        printf("\n");
        /* second for loop inside the first */
        for(j = i; j > 0; j--) {
                printf("%d", j);
        }
}
}
```

Output:

1
21
321
4321
54321

The Infinite Loop

A loop becomes an infinite loop if a condition never becomes false.

When the conditional expression is absent, it is assumed to be true. You may have an initialization and increment expression, but C programmers more commonly use the for(;;) construct to signify an infinite loop.

Obukk, then what can we do to stop a loop….?

Loop Control Statements

Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

C supports the following control statements.

Sr.No.	Control Statement & Description
1	**break statement** Terminates the loop or switch statement and transfers execution to the statement immediately following the loop or switch.
2	**Continue statement** Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.
3	**auto statement** Transfers control to the labeled statement.

Loop Control Statements (break)

The **break** statement in C programming has the following two usages −

•When a **break** statement is encountered inside a loop, the loop is immediately terminated and the program control resumes at the next statement following the loop.
•It can be used to terminate a case in the **switch** statement (covered in the next chapter).
If you are using nested loops, the break statement will stop the execution of the innermost loop and start executing the next line of code after the block.

The syntax for a **break** statement in C is as follows −

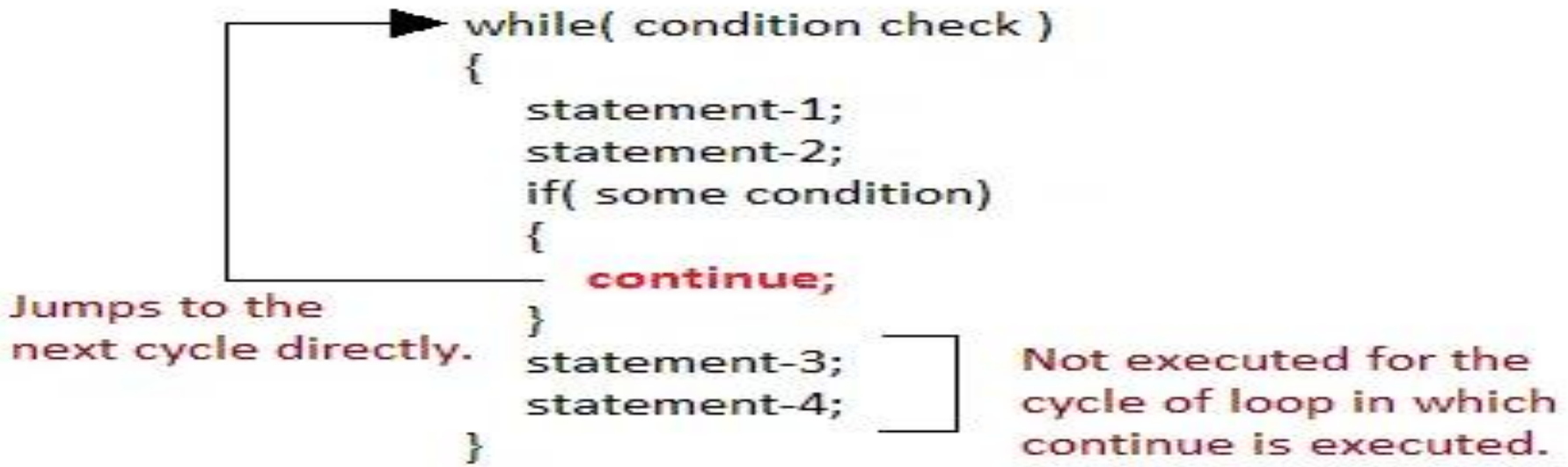
```
break;
```

Loop Control Statements (break)

```
while( condition check )
{
    statement-1;
    statement-2;
    if( some condition)
    {
        break;
    }
    statement-3;
    statement-4;
}
```

Jumps out of the loop, no matter how many cycles are left, loop is exited.

Loop Control Statements (continue)



Loop Control Statements (continue, goto)

Will be available in pdf in Google Classroom, Okay…?

More about nested loop

PDF in Google Classroom

&

https://www.programtopia.net/c-programming/docs/nested-loop

Obuk, ajk sesh hbe!?

Any questions?

Please,

Anyone answer/ explain.

Chat এ লিখে পেল দ্রুত!

Kn loop ta brtter mne hlo.....?

Thanks
(Jajakallah Khairan)

Hosted and prepared by

MD. SOROWAR MAHABUB RABBY

DEPARTMENT OF CSE (COMPUTER SCIENCE AND ENGINEERING) , IIUC

CELL: 01834756433, 01521564157

E-MAIL: sorowarmahabub1709vip@gmail.com