Control Statement

Loops in C Programming

Loops in Programming [what is loop in programming?]

Looping Statement in C execute the Sequence of Statements many times until the stated condition becomes false. A loop in C consists of two parts, a body of a loop and a control Statement.

The control Statement is a combination of some conditions that direct the body of the loop to execute until the specified condition becomes false. The purpose of the C loop is to repeat the same code a number of times.

Types of Loops in C

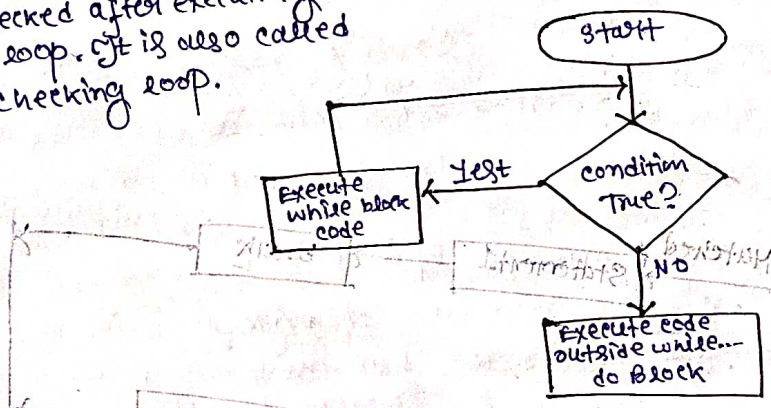
Looping Statement in C is classified into two types

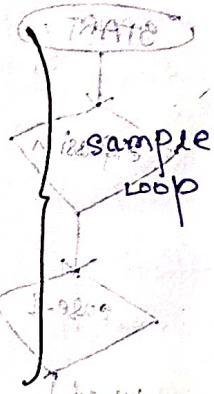
(1) Entry controlled Loop

in entry controlled loop in C, a condition is checked before executing the body of a loop. It is also called as a pre-checking loop.

(2) Exit controlled Loop

in a exit controlled loop, a condition is checked after executing the body of a loop. It is also called as a post-checking loop.





* What if Infinite Loop ?

The control condition Must be well defined and Specified otherwise the loop will execute an infinite number of times. The loop that does not stop executing and process the Statements number of times is called as an Infinite Loop.

* What if Endless Loop ?

An infinite loop is also called as an endless loop.

Characteristics of an infinite Loop

(1) No termination condition is specified
(2) The specified condition never meet

Types of Loop in C Language

→ The while Loop
→ The do-while Loop
→ The for Loop.

While Loop *

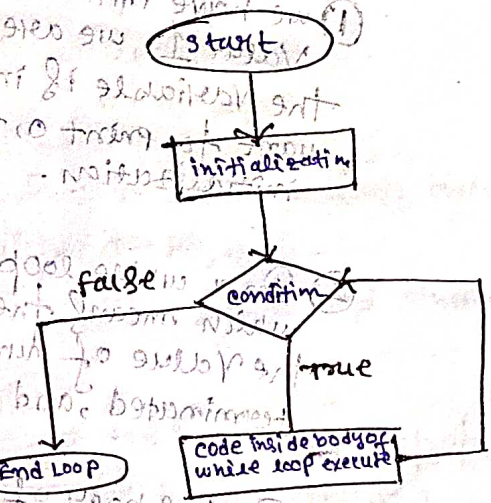
[what is / write down the syntax / draw the flowchart / Short note about while loop] *

A while Loop is the most straightforward looping structure. while loop syntax in C programming Language is as follows

Syntax of while Loop

```
while (condition) {
    statements;
}
```

Body of while Loop



It is a entry controlled loop. in while loop, a condition is evaluated before processing a body of the loop. If a condition is True then and only then the body of the loop is executed then control again goes back at the beginning, and the condition is checked if it is true, the same process is executed until the condition becomes ~~true~~ false. once the condition becomes false, the control goes out of the loop.

After exiting the loop, the control goes to the statements which are immediately after the loop. The body of a loop can contain more than one statement. if it contains only one statement, then the curly braces are not compulsory.

In, while loop if the condition is not true, then the body of a loop will not be executed, not even once.

Following program illustrates while loop in C

```
#include <stdio.h>
int main() {
    int num = 1;          // Initialization part
    while (num <= 10)     // conditional part
    {
        printf("%d\n", num);
        num++;            // Increment/decrement part
    }
    return 0;
}
```

OUTPUT

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

① we have initialized a variable called num with value 1. we are going to print from 1 to 10 hence the variable is initialized with value 1. if you want to print 0, then assign the value 0 during initialization.

② In a while loop, we have provide a condition (num<=10), which means the loop will execute the body until the value of num becomes 10. After that, the loop will be terminated, and control will fall outside the loop.

③ In the body of a loop, we have print function to print our number and an increment operation to increment the value per execution of a loop.

An initial value of num is 1, after the execution, it will become 2, and during the next execution, it will become 3. This process will continue until the value becomes 10 and then it will print the series on console and terminate the loop.

\n is used for formatting purpose which means the value will be printed on a new line.

Do-while loop

* [what is ? / write down the Syntax / draw the flowchart / Short note about do-while loop]

A do-while loop in c is similar to the while loop expt that the condition is always executed after the body of a loop. It is also good called an exit-controlled loop.

* what if exit controlled loop ? why it is called exit controlled loop ? Ans

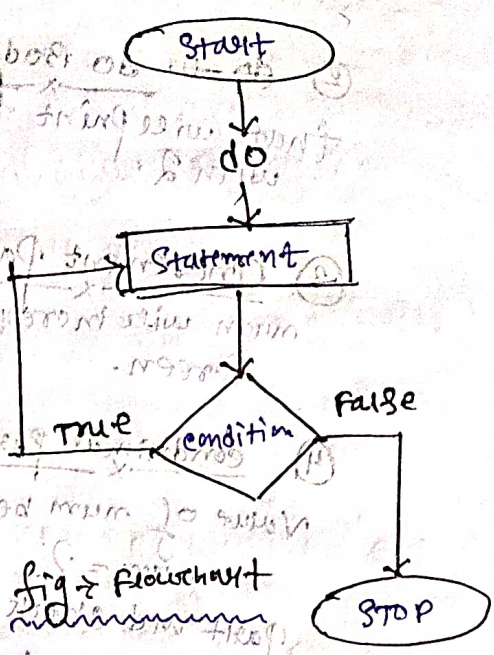


fig :- flowchart

Syntax of Do-while Loop in C

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

In do-while loop, the body of a loop is always executed at least once. After the body is executed, then it checks the condition. if the condition is true, then it will again execute the body of a loop otherwise control is transferred out of the loop.

Similar to the while loop, once the control goes out of the loop the statements which are immediately after the loop is executed.

The following loop program in c show the working of a do-while loop

```
#include <stdio.h>
int main() {
    ① int num=1; // initialization part
    do {
        printf("%d\n", 2*num); ②
        ③ num++; // increment / de-crement part
    } while (num<=10); // condition
        ④
}
```

Body of do

OUTPUT

```
2
4
6
8
10
12
14
16
18
20
```

(1) Initialization part :- first, we have initialized a Variable 'num' with value 1. Then we have written a do-while loop.

(2) do Body :- In a loop, we have a print function that will print the series by multiplying the value num with 2.

(3) Increment Part :- After each increment, the value of num will increase by 1, and it will printed on the screen.

(4) conditional part :- after printing the ~~rata~~ statements the value of num becomes num++. hence the value becomes num = 2. Now compiler will enter to the while part and check the condition, if the condition is true then repeat step 2 and step 3 and step 4 until the value of the ~~num becomes 10~~ condition of the num becomes false.

After that Loop will be terminated and a statements which is immediately after the loop will be executed.

For Loop

[what is ? / write down the syntax / draw the flowchart / Short note about for loop]

A for loop if more efficient loop Structure in 'c' programming. ~~the general~~ It is the most commonly used loops in c programming. It is an alternative / short cut of for while loop in c programming.

syntax of For Loop in C

```
for (initial value; condition; Increment/Decrement)
Body of for Loop {
    Statements;
}
```

Characteristics of for Loop in C

- The initial Value of the for loop is performed only once.
- The condition is a Boolean expression that tests and compares the counter to a fixed Value after each iteration, ~~stop~~ stopping the for loop when false is returned.
- The incrementation/decrementation increases (or decreases) the counter by set Value.

Program to illustrate the for Loop in C

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

① ② ③

OUTPUT

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

we have declared a Variable of an int, data type to store Values.

① Initialization part :- we have assigned Value 1 to the Variable number. ~~In the condition part~~

② Condition part :- In the condition part we have specified our condition.

③ Increment part :- after condition we increase the Value of ~~number~~ number by doing number++.

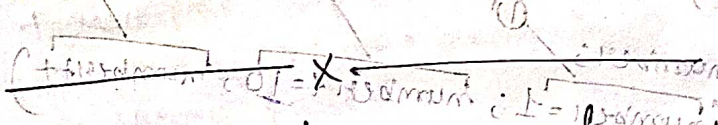
Next Page →

working principle of for loop

In the body of a loop, we have print function to print the numbers on a new line in the console. we have the value one stored in number, after the first iteration the value will be incremented, and it will become 2.

Now the variable number has the value 2. The condition will be re-checked and since the condition is true loop will be executed and it will print two on the screen.

This loop will keep on executing until the value of the variable become 10. After that the loop will be terminated, and a series of 1-10 will be printed on the screen



Difference between While, do-while and for loop [5 Marks]

for loop	while loop	do-while loop
① In for loop statement is executed then after inc/dec the values.	①	
① It is known as entry controlled loop	① It is also called as entry controlled loop	① It is known as exit controlled loop
② The statement which you write down into a body of the for loop never executed if the condition is false	② Same like for loop	② Here the statements which you write down into the body of the loop execute atleast once. if the condition is false then also it will execute once.
③ for loop is used when you know the no of iteration	③ when you don't know the no of iteration that time we are using while loop	③ Same like while.
④ Syntax	④ Syntax	④ Syntax
⑤ Flowchart	⑤ Flowchart	⑤ Flowchart

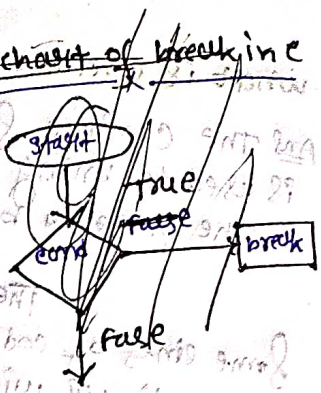
Break Statement

What is Break Statement ?

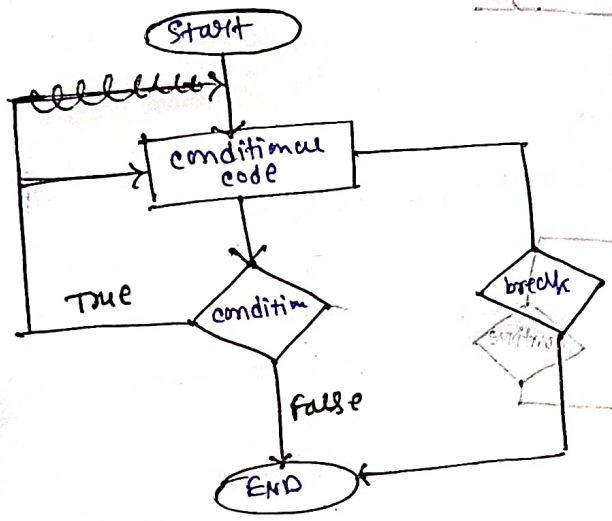
The Break is a keyword in c which is used to bring the program control out of the loop. The break statement is used inside loops or switch case.

The break statement breaks the loop one by one.

Flowchart of break in c



Flowchart of Break Statement



Characteristics / working Principle of Break Statement

1. Break Statement terminates the loop containing it
2. control of the program flows to the statement immediately after the body of the loop.
3. when the break statement is executed, the controll flow of the program comes out of the loop and starts executing the segment of code after the loop structure
4. if the break statement is inside a nested loop, the break will terminate the innermost loop.

Program to illustrates Break statement

```
# include <stdio.h>
int main(){
    int a, i, sum;
    for (i=1; i<=5; i++)
    { printf("Enter a no:\n");
      scanf("%d", &a);
        if (a<0){
            break;
        } else {
          sum = sum+a;
        }
}
printf("Sum: %d", sum);
}
```

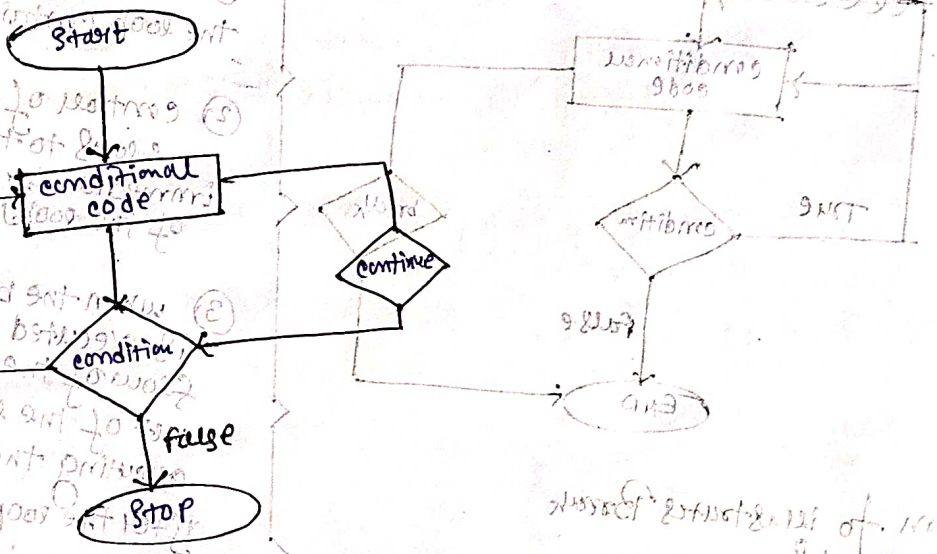
Continue Statement

what is continue Statement ?

Ans The continue Statement in c language is used to bring the program control to the beginning of the loop.

The continue Statement skips some lines of code inside the loop and continues with the next iteration. It is mainly used for condition so that we can skip some code for a particular condition.

Flowchart of continue Statement



Program to illustrate continue Statement

```
#include <stdio.h>
int main() {
    int i, a, sum=0;
    for(i=1; i<=5; i++)
    { printf("Enter an integer: \n");
      scanf("%d", &a);
        if (a<0){
            continue;
        } else {
            sum = sum+a;
    } }
    printf("sum=%d", sum);
}
```