Register in Hackerrank and practice

Example-1:

```
#include <stdio.h>

int main()
{
   printf("\n %d",printf("Hello World"));

   return 0;
}
```

Output:

```
Hello World
 11
```

Example-2:

```
#include <stdio.h>

int main()
{
   int i;
   printf("\n %d",scanf("%d",&i));

   return 0;
}
```

Output:

```
12

 1
```

An infinite loop is a looping construct that does not terminate the loop and executes the loop forever. It is also called an **indefinite** loop or an **endless** loop.

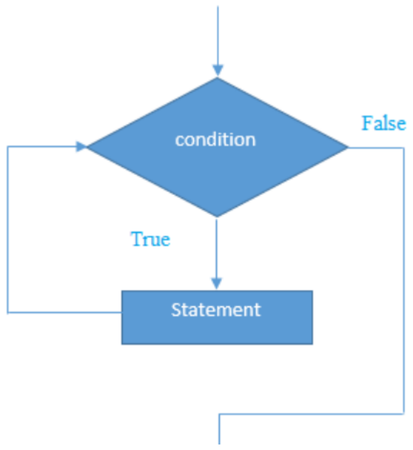
While Loop

The while loop is mostly used in the case where the number of iterations is not known in advance.

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

```
while(condition){
//code to be executed
}
```

Flowchart of while loop in C



- In while loop, the condition expression is compulsory.
- Running a while loop without a body is possible.

Example-1:

```
#include<stdio.h>
void main ()
{
   while()
   {
      printf("hello");
   }
}
```

compile time error: while loop can't be empty

Example-2:

```
#include<stdio.h>
void main ()
{
    int j = 1;
    while(j+=2,j<=10)
    {
        printf("%d ",j);
    }
    printf("%d",j);
}
```

Output

```
3 5 7 9 11
```

Example-3:

Infinite while loop

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

Syntax of for loop in C

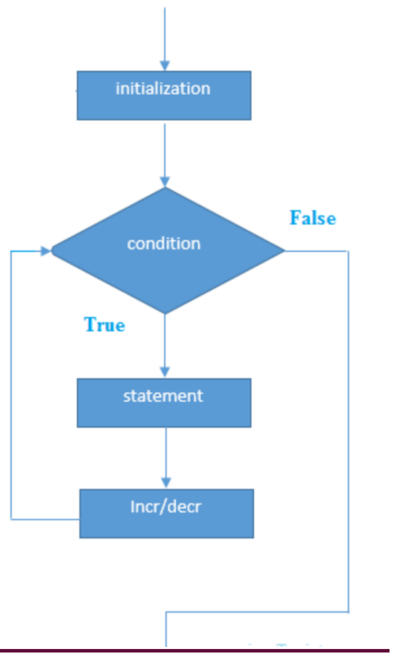
The syntax of for loop in c language is given below:

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

}

- We can initialize more than one variable in Expression 1.
- Expression 1 is optional.
- Expression 2 is optional.
- Expression 2 can have more than one condition, we can initialize the variable as well as update the loop variable in expression 2 itself.
- We can update more than one variable at the same time.
- Expression 3 is optional.

Flowchart of for loop (or can use the symbol discussed during flowcharts from textbook)



```
#include <stdio.h>

int main()
{
   int i;
   for(i=0;i<10;i++);
   {
      printf("%d",i);
   }

   return 0;
}
```

Output:

10

Example 1

```
#include <stdio.h>
int main()
{
    int a,b,c;
    for(a=0,b=12,c=23;a<2;a++)
    {
        printf("%d ",a+b+c);
    }
}
```

Output

```
35 36
```

Example 2

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

Output

```
1 2 3 4
```

Example 3

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

```
infinite loop
```

Example 4

```
#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
```

Infinite for loop

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.

```
#include<stdio.h>
void main ()
{
    for(;;)
    {
        printf("welcome ");
    }
}
```

do while loop

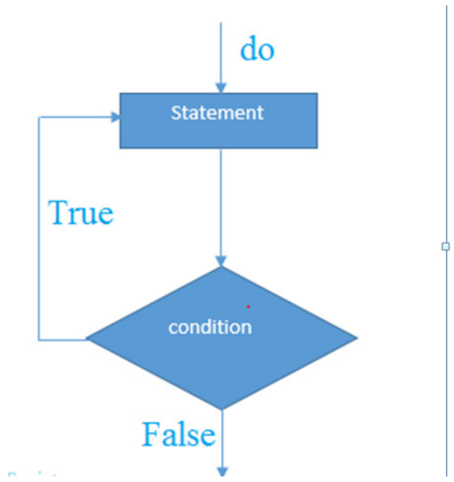
- The do while loop is a post tested loop.
- Using the do-while loop, we can repeat the execution of several parts of the statements.
- The do-while loop is mainly used in the case where we need to execute the loop at least once.
- The do-while loop is mostly used in menu-driven programs where the termination condition depends upon the end user.

do while loop syntax

The syntax of the C language do-while loop is given below:

```
do{
//code to be executed
}while(condition);
```

Flowchart of do while loop



Example1:

```
#include<stdio.h>
#include<stdlib.h>
void main ()
{
    char c;
    int no1, no2;
    int choice;
    printf(“Enter 2 numbers”);
    scanf(“%d %d ”, &no1,&no2);
    do{

    printf("\n1. Addition \n2. multiplication\n3. Exit\n");
   scanf("%d",&choice);
    switch(choice)
    {
        case 1 :
            printf("sum=%d", no1+no2);
                    break;
        case 2:
            printf("product = %d", no1*no2);
                    break;
      case 3:
            exit(0);
            break;
        default:
        printf("please enter valid choice");
    }
    printf("do you want to continue (y/n)?");
        scanf(" %c",&c);
    }while(c=='y');
}
```

Example-2:

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

Output

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

Example-3:

Program to print table for the given number using do while loop

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

Infinite do while loop

The do-while loop will run infinite times if we pass any non-zero value as the conditional expression.

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

Nested Loops in C

C supports nesting of loops. **Nesting of loops** is the feature in C that allows the looping of statements inside another loop.

Any number of loops can be defined inside another loop, i.e., there is no restriction for defining any number of loops. The nesting level can be defined at n times. You can define any type of loop inside another loop; for example, you can define '**while**' loop inside a '**for**' loop.

Syntax of Nested loop

```
Outer_loop
{
    Inner_loop
   {
        // inner loop statements.
   }
      // outer loop statements.
}
```

Outer_loop and **Inner_loop** are the valid loops that can be a 'for' loop, 'while' loop or 'do-while' loop.

Nested for loop

The nested for loop means any type of loop which is defined inside the 'for' loop.

```
for (initialization; condition; update)
{
    for(initialization; condition; update)
    {
          // inner loop statements.
    }
    // outer loop statements.
}
```

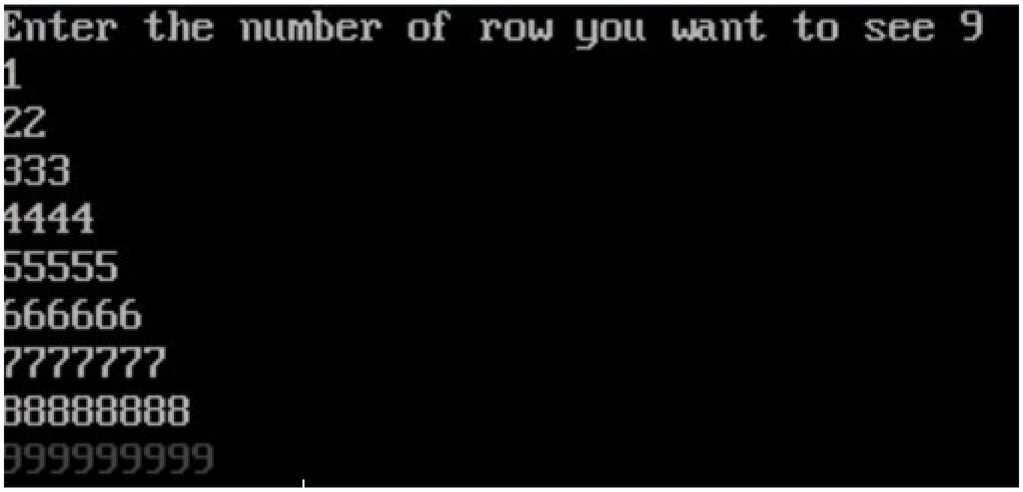
Nested while loop

The nested while loop means any type of loop which is defined inside the 'while' loop.

```
while(condition)
{
    while(condition)
    {
        // inner loop statements.
    }
// outer loop statements.
}
```

```
#include<stdio.h>
void main()
{
int i,j,term;
clrscr();
printf("Enter the number of row you want to see");
scanf("%d",&term);
for (i = 1 ; i <= term ; i++)
{
for(j = 1 ; j <= i ; j++)
{
printf("%d",i);
}
printf("\n");
}
getch();
}
```

Output #1:



```
Enter the number of row you want to see 9
1
22
333
4444
55555
666666
7777777
88888888
999999999
```

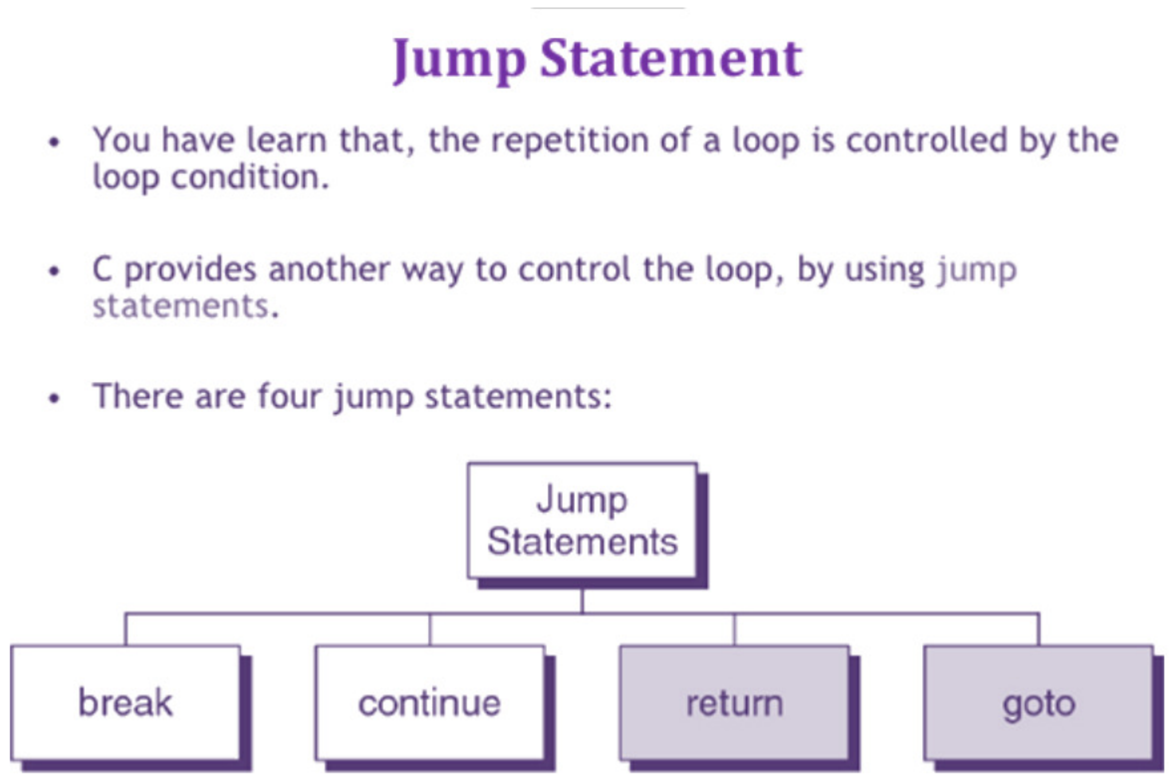
```
#include<stdio.h>
```

```
void main()
{
int i,j,term;
printf("Enter the number of row you want to see");
scanf("%d",&term);
for (i = term ; i >= 1 ; i--)
{
for(j = 1 ; j <= i ; j++)
{
printf("%d",i);
}
printf("\n");
}
}
```

```
55555

4444

333

22

1
```

PROGRAMS USING STARS

JUMP STATEMENTS



Jump Statement

- You have learn that, the repetition of a loop is controlled by the loop condition.
- C provides another way to control the loop, by using jump statements.
- There are four jump statements:

C break statement

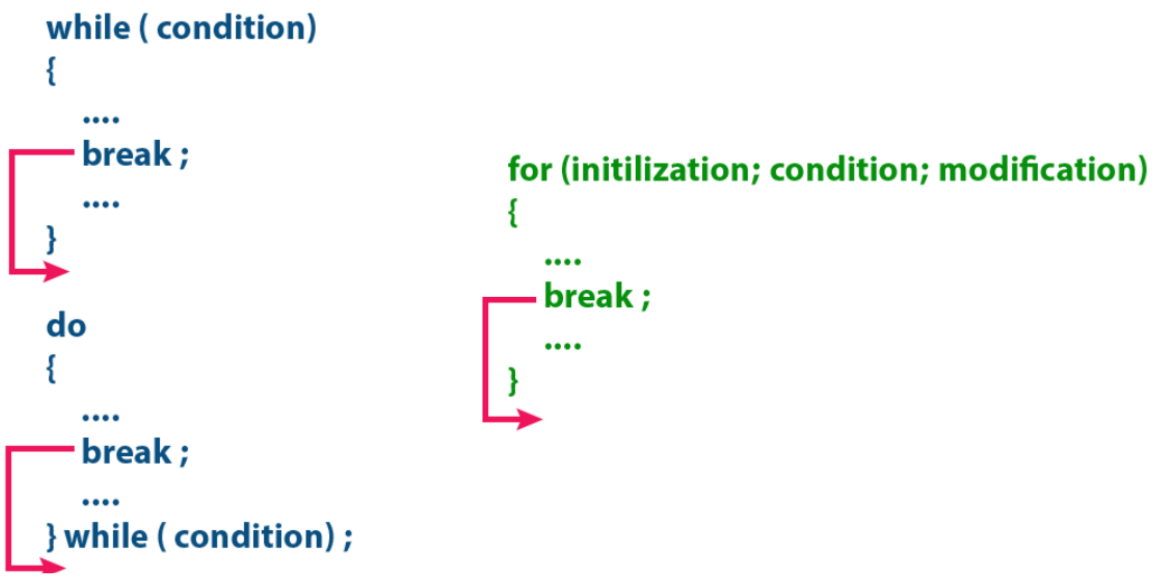
The break is a keyword in C which is used to bring the program control out of the loop. The break statement is used inside loops or switch statement. The break statement breaks the loop one by one, i.e., in the case of nested loops, it breaks the inner loop first and then proceeds to outer loops. The break statement in C can be used in the following two scenarios:

1. With switch case
2. With loop

Syntax:

```
//loop or switch case
break;
```

The floowing picture depictes the execution flow of the break statement.



Example

```
#include<stdio.h>
#include<stdlib.h>
void main ()
{
    int i;
    for(i = 0; i<10; i++)
    {
        printf("%d ",i);
        if(i == 5)
        break;
    }
    printf("came outside of loop i = %d",i);

}
```

Output

```
0 1 2 3 4 5 came outside of loop i = 5
```

```
#include<stdio.h>
void main ()
{
    int i = 0;
    while(1)
    {
        printf("%d  ",i);
        i++;
        if(i == 10)
        break;
    }
    printf("came out of while loop");
}
```

Output

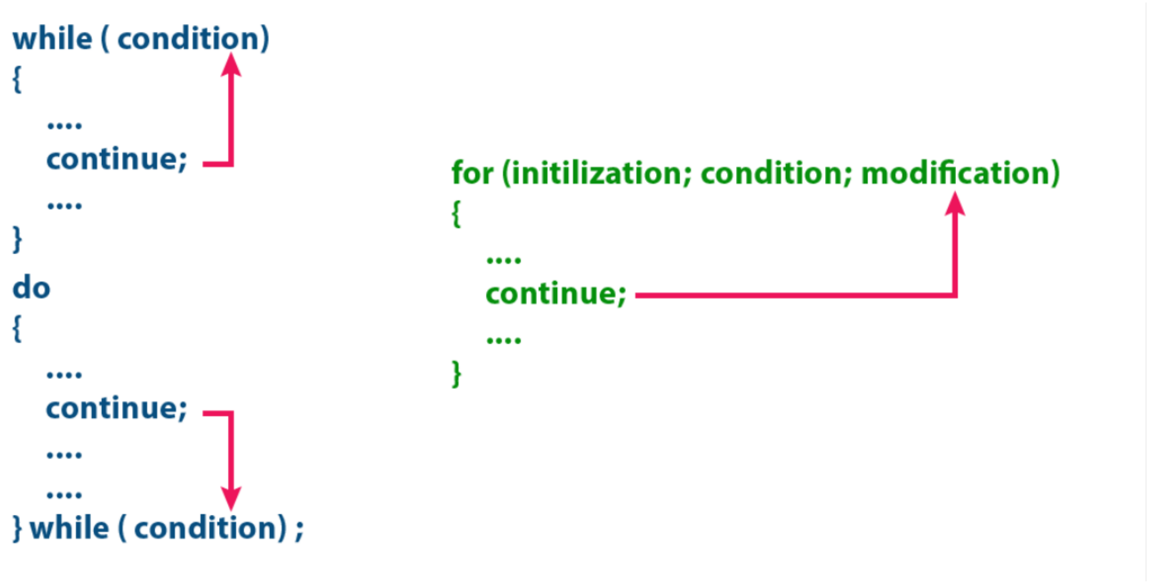
```
0  1  2  3  4  5  6  7  8  9  came out of while loop
```

Continue statement

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 a condition so that we can skip some code for a particular condition.

Syntax:

```
//loop statements
continue;
//some lines of the code which is to be skipped
```



Continue statement example 1

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

```
infinite loop
```

```
#include<stdio.h>
int main(){
int i=1;//initializing a local variable
//starting a loop from 1 to 10
for(i=1;i<=10;i++){
if(i==5){//if value of i is equal to 5, it will continue the loop
continue;
}
printf("%d \n",i);
}//end of for loop
return 0;
}
```

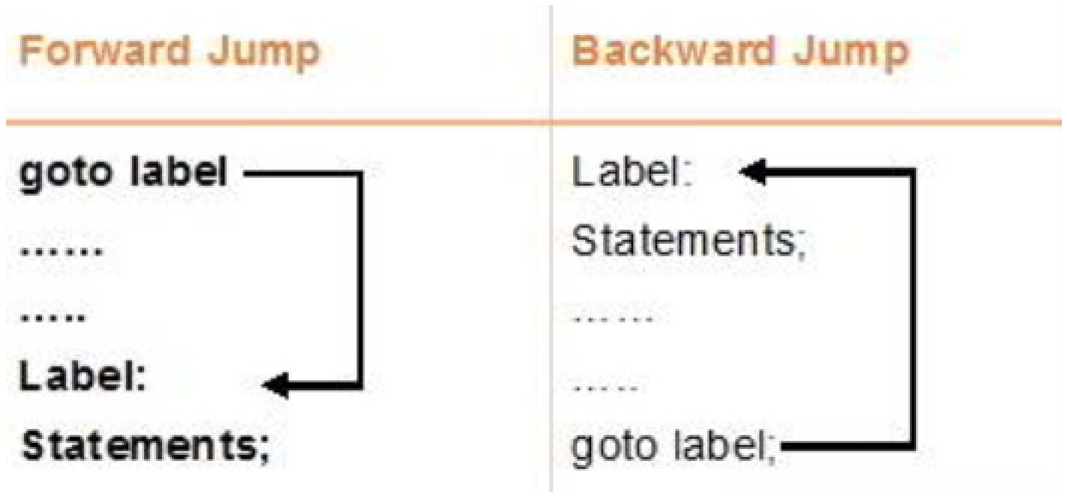
```
1
2
3
4
6
7
8
9
10
```

C goto statement

The goto statement is known as jump statement in C. As the name suggests, goto is used to transfer the program control to a predefined label. The goto statment can be used to repeat some part of the code for a particular condition. It can also be used to break the multiple loops which can't be done by using a single break statement. However, using goto is avoided these days since it makes the program less readable and complicated.

Syntax:

```
label:
//some part of the code;
goto label;
```



goto example

```
#include <stdio.h>
int main()
{
  int num,i=1;
  printf("Enter the number whose table you want to print?");
  scanf("%d",&num);
  table:
  printf("%d x %d = %d\n",num,i,num*i);
  i++;
  if(i<=10)
  goto table;
}
```

Output:

```
Enter the number whose table you want to print?10
10 x 1 = 10
10 x 2 = 20
10 x 3 = 30
10 x 4 = 40
10 x 5 = 50
10 x 6 = 60
10 x 7 = 70
10 x 8 = 80
10 x 9 = 90
10 x 10 = 100
```

The only condition in which using goto is preferable is when we need to break the multiple loops using a single statement at the same time. Consider the following example.

```
#include <stdio.h>
int main()
{
  int i, j, k;
  for(i=0;i<10;i++)
  {
    for(j=0;j<5;j++)
    {
      for(k=0;k<3;k++)
      {
        printf("%d %d %d\n",i,j,k);
        if(j == 3)
        {
          goto out;
        }
      }
    }
  }
  out:
  printf("came out of the loop");
}
```

```
0 0 0
0 0 1
0 0 2
0 1 0
0 1 1
```

```
0 1 2
0 2 0
0 2 1
0 2 2
0 3 0
came out of the loop
```

TO CHECK THE GIVEN NUMBER IS PRIME NO

```c
#include <stdio.h>
int main() {
    int n, i, flag = 0;
    printf("Enter a positive integer: ");
    scanf("%d", &n);

    for (i = 2; i <= n / 2; ++i) {

        // condition for non-prime
        if (n % i == 0) {
            flag = 1;
            break;
        }
    }

    if (n == 1) {
        printf("1 is neither prime nor composite.");
    }
    else {
        if (flag == 0)
            printf("%d is a prime number.", n);
        else
            printf("%d is not a prime number.", n);
    }

    return 0;
}
```