ST. XAVIER'S COLLEGE

MAITIGHAR, KATHMANDU, NEPAL
Phone: 01-5321365, 01-5344636
Email: ktm@sxc.edu.np

LAB/WRITTEN ASSIGNMENT NUMBER: 8

" Iterative Control structures in C "

Submitted By	Submitted To	Signature
Name : Pranjal Khatri Roll : 929 Class : 11 Section : I	Department Of Computer Science (+2) St.Xavier's College	

Submission Date: 28 November, 2024

Contents

Introduction to Iteration

Iteration is a fundamental concept in programming world in which a specific set of instructions are repeated in loop multiple times until a set condition is met. In C programming language, there are three types of iteration statements: for, while, and do-while

For Loop

The for loop is used to execute a set of statements repeatedly for a fixed number of times. It consists of three parts: initialization, condition, and increment/decrement.

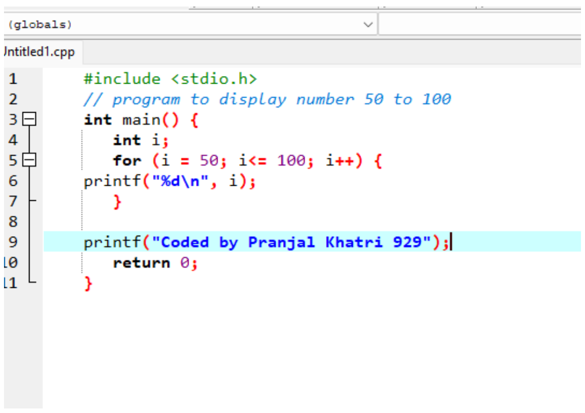
Working principle:

The initialization statement is executed only once at the beginning of the loop, and it is used to initialize the loop variable. The condition statement is evaluated at the beginning of each iteration, if it is true, the code inside the loop is executed. The increment/decrement statement is executed at the end of each iteration, and it is used to update the loop variable.

Syntax :

```
For(initialization ; condition ; increment or decrement )
  {
    Block of statements ;
   }
```

Example :

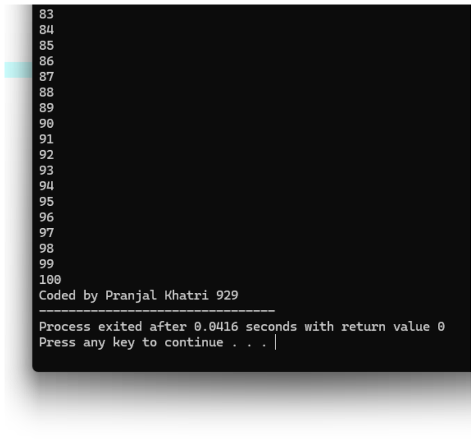


(globals)

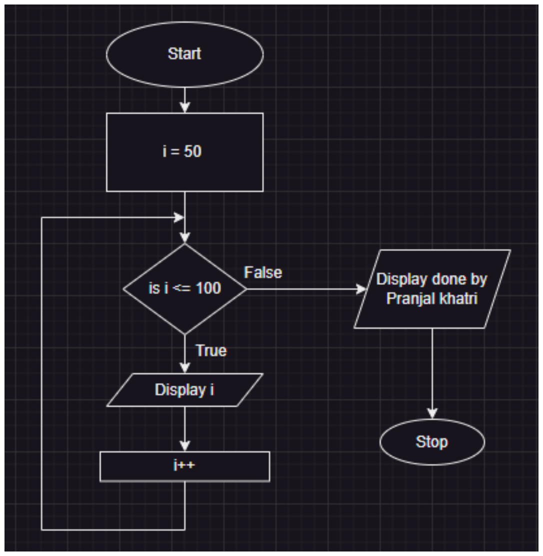
Untitled1.cpp

```
#include <stdio.h>
// program to display number 50 to 100
int main() {
   int i;
   for (i = 50; i<= 100; i++) {
printf("%d\n", i);
   }

printf("Coded by Pranjal Khatri 929");
   return 0;
}
```

Output :


```
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
Coded by Pranjal Khatri 929
--------------------------------
Process exited after 0.0416 seconds with return value 0
Press any key to continue . . .
```

Flowchart :



Algorithm :

Step 1: Start.
Step 2: Include the <stdio.h> library.
Step 3: Define the main function.
Step 4: Declare an integer variable i.
Step 5: Use a for loop to iterate i from 50 to 100.
Step 6: Print the value of i inside the loop.
Step 7: Print "Coded by Pranjal Khatri 929" after the loop.
Step 8: End the program with return 0.
Step 9: Stop.

While loop

The while loop is used to execute a set of statements repeatedly as long as a certain condition is true.

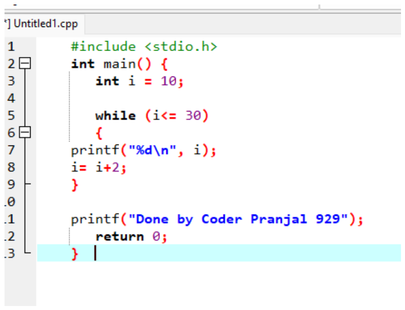
Syntax:

```
Initialization
While ( condition )
  {
   Block of statement
    Increment or decrement
   }
```

Principle:

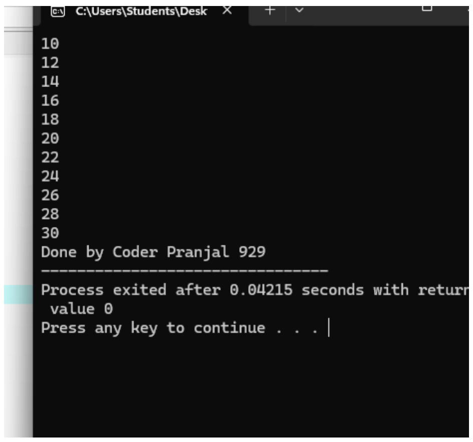
The condition is evaluated at the beginning of each iteration, and if it is true, the code inside the loop is executed. The loop continues until the condition becomes false.

Example:



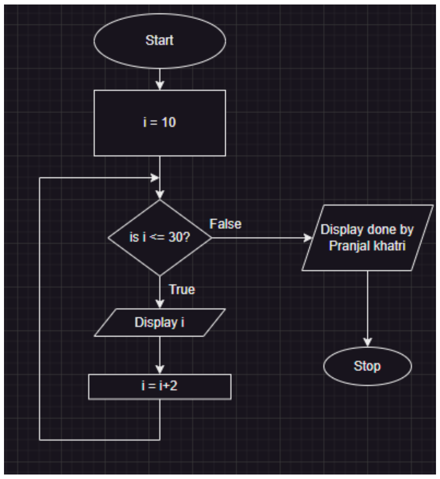
```
#include <stdio.h>
int main() {
    int i = 10;

    while (i<= 30)
    {
printf("%d\n", i);
i= i+2;
}

printf("Done by Coder Pranjal 929");
    return 0;
}
```

Output



```
10
12
14
16
18
20
22
24
26
28
30
Done by Coder Pranjal 929
--------------------------------
Process exited after 0.04215 seconds with retur
 value 0
Press any key to continue . . .
```

Flowchart:



Algorithm:

Step 1: Start.
Step 2: Include the <stdio.h> library.
Step 3: Define the main function.
Step 4: Declare an integer variable i and initialize it to 10.
Step 5: Use a while loop with the condition i <= 30.
Step 6: Inside the loop, print the value of i and increment i by 2.
Step 7: After the loop, print "Done by Coder Pranjal 929".
Step 8: End the program with return 0.
Step 9: Stop.

Do while loop

The do-while loop is used to execute a set of statements repeatedly as long as a certain condition is true. The difference between the while loop and the do-while loop is that the do-while loop executes the code inside the loop at least once before checking the condition.

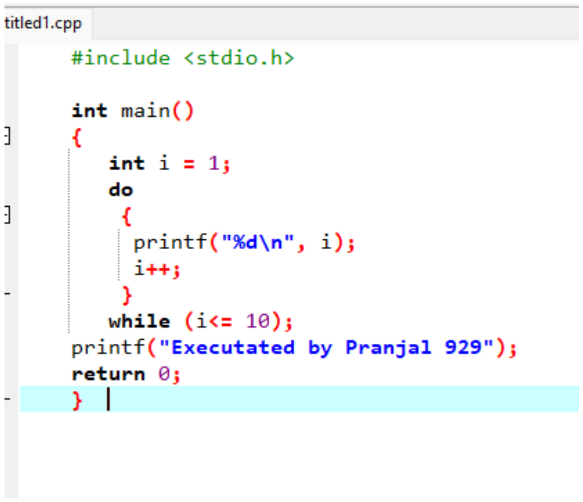
Syntax:

```
    Initialization
    Do
  {
    Block of statement
   Increment or decrement
  } while (condition);
```

Principle

The code inside the loop is executed first, and then the condition is checked. If the condition is true, the loop continues, otherwise, it terminates.

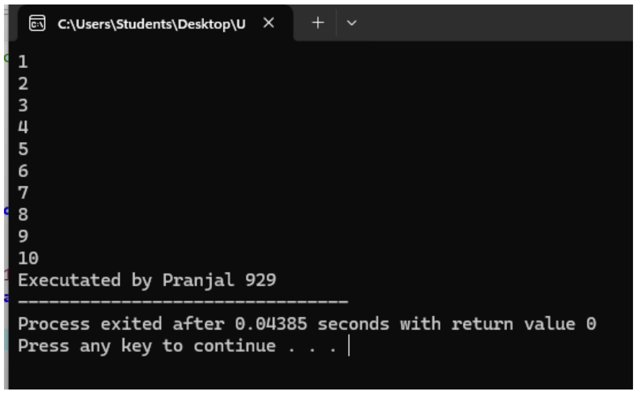
Example:



titled1.cpp

```
#include <stdio.h>

int main()
{
    int i = 1;
    do
     {
      printf("%d\n", i);
      i++;
     }
    while (i<= 10);
printf("Executated by Pranjal 929");
return 0;
}
```

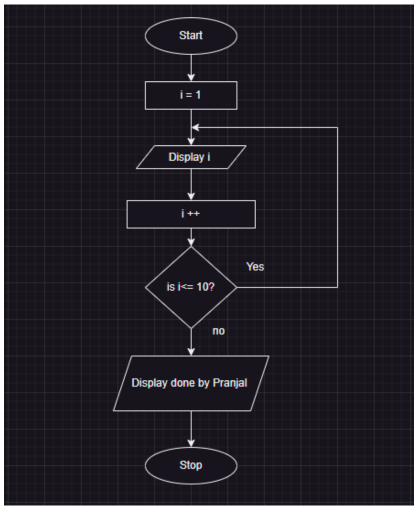
Output:



C:\Users\Students\Desktop\U

```
1
2
3
4
5
6
7
8
9
10
Executated by Pranjal 929
--------------------------------
Process exited after 0.04385 seconds with return value 0
Press any key to continue . . .
```

Flowchart:



Algorithm:

Step 1: Start.
Step 2: Include the <stdio.h> library.
Step 3: Define the main function.
Step 4: Set i = 1.
Step 5: Use a do-while loop to print i and increase i by 1 until i <= 10.
Step 6: Print "Executed by Pranjal 929".
Step 7: End.

Nested loops

A nested loop means a loop statement inside another loop statement. That is why nested loops are also called "loop inside loops". We can define any number of loops inside another loop.

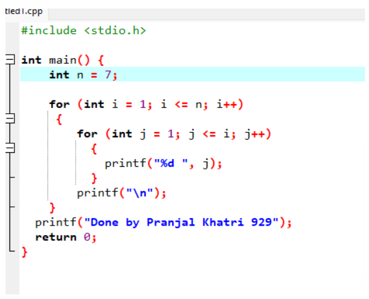
Nested for Loop

Any nested loop that is defined by using two for loop is nested for loop

Syntax

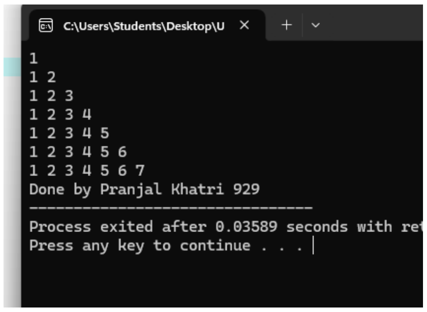
```
for ( initialization; condition; increment )
 {
      for ( initialization; condition; increment )
        {
          Block of statement for inner loop
        }

    Block of statement of outer loop
 }
```

Example



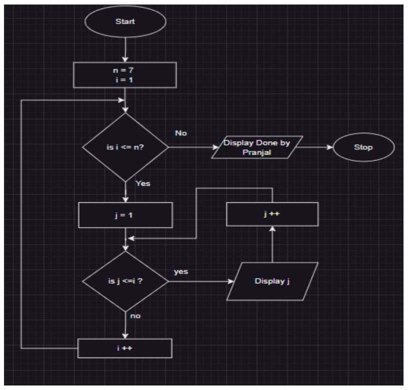
```c
#include <stdio.h>

int main() {
    int n = 7;

    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= i; j++)
        {
            printf("%d ", j);
        }
        printf("\n");
    }
  printf("Done by Pranjal Khatri 929");
  return 0;
}
```

Output



```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
Done by Pranjal Khatri 929
--------------------------------
Process exited after 0.03589 seconds with ret
Press any key to continue . . .
```

Flowchart:



Algorithm

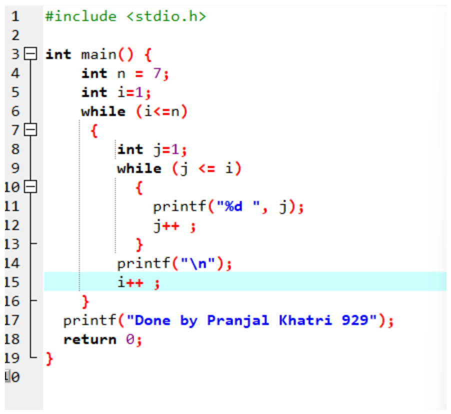
Step 1: Start.
Step 2: Initialize n as 7.
Step 3: Set i to 1 and repeat steps until i is less than or equal to n.
Step 4: Set j to 1 and repeat steps until j is less than or equal to i.
Step 5: Print j followed by a space.
Step 6: Increment j by 1.
Step 7: Print a newline after the inner loop ends.
Step 8: Increment i by 1.
Step 9: Print "Done by Pranjal Khatri 929".
Step 10: Stop

Nested while loop

If we use another while loop inside another while loop then it will be nested while loop.

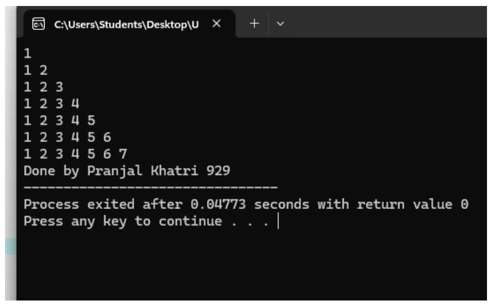
```
Syntax:
Initialization
while(condition)
  {
     while(condition)
        {
            Block of Statement of inside loop
            Increment or decrement

         }

    Block of statement of outside while loop
    Increment or decrement
  }
```

Example



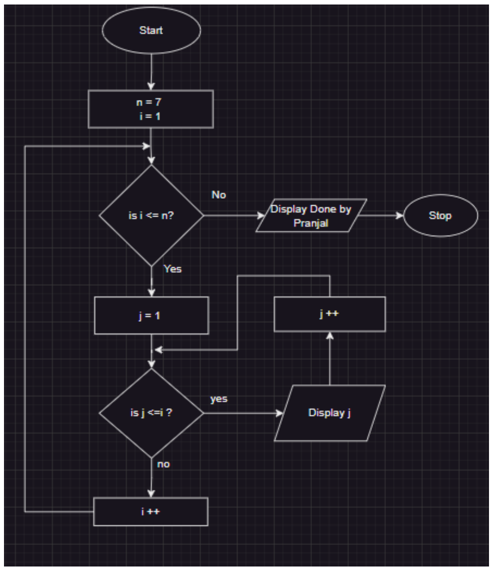
```c
#include <stdio.h>

int main() {
    int n = 7;
    int i=1;
    while (i<=n)
     {
        int j=1;
        while (j <= i)
          {
            printf("%d ", j);
            j++ ;
          }
        printf("\n");
        i++ ;
     }
  printf("Done by Pranjal Khatri 929");
  return 0;
}
```

Output



```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
Done by Pranjal Khatri 929
--------------------------------
Process exited after 0.04773 seconds with return value 0
Press any key to continue . . .
```

Flowchart



Algorithm:

Step 1: Start.

Step 2: Initialize n as 7.

Step 3: Set i to 1 and repeat steps until i is less than or equal to n.

Step 4: Set j to 1 and repeat steps until j is less than or equal to i.

Step 5: Print j followed by a space.

Step 6: Increment j by 1.

Step 7: Print a newline after the inner loop ends.

Step 8: Increment i by 1.

Step 9: Print "Done by Pranjal Khatri 929".

Step 10: Stop.

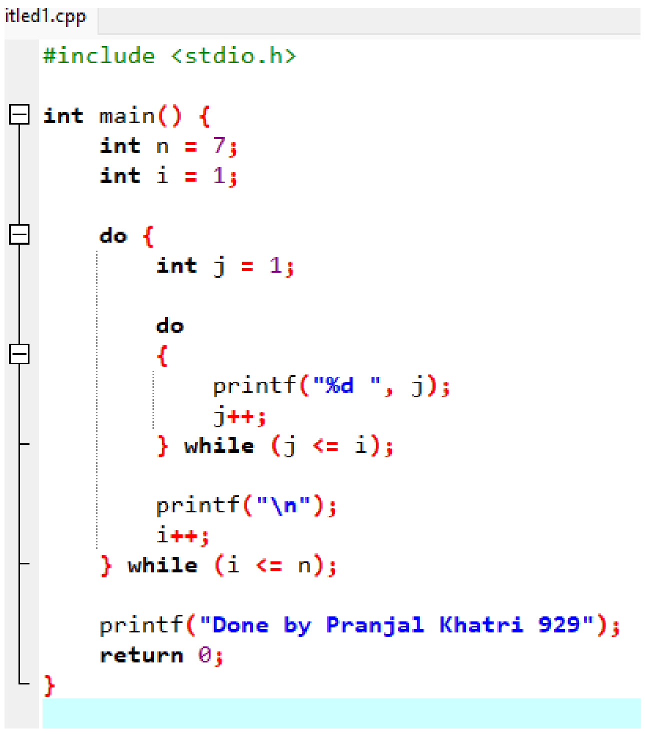
Do-while nested loop

Do while nested loop is a nested loop that is created by using a do while statement inside of another do while.

Syntax :

```
  initialization
   do
     {
        do
          {
             Block of statement of inside loop
           }  while(condition);
        Block of  statement of outer loop
     } while(condition);
```

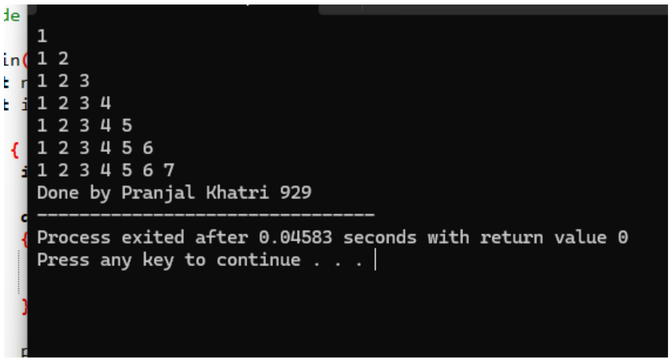
Example :



itled1.cpp

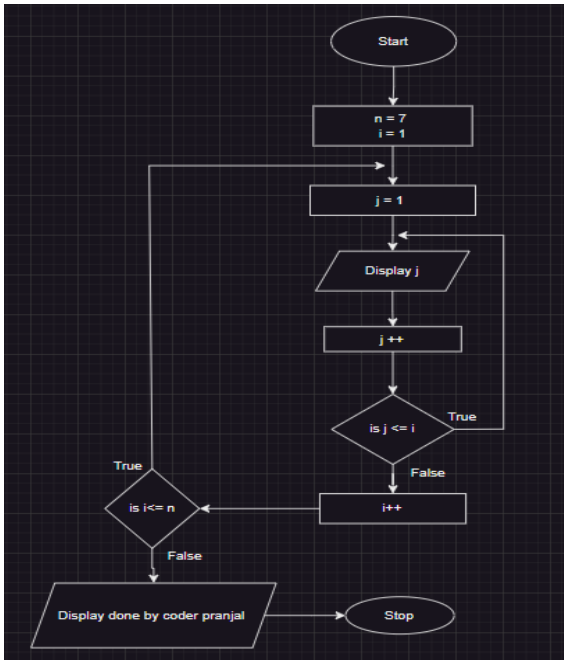
```
#include <stdio.h>

int main() {
    int n = 7;
    int i = 1;

    do {
        int j = 1;

        do
        {
            printf("%d ", j);
            j++;
        } while (j <= i);

        printf("\n");
        i++;
    } while (i <= n);

    printf("Done by Pranjal Khatri 929");
    return 0;
}
```

Output:



```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
Done by Pranjal Khatri 929
--------------------------------
Process exited after 0.04583 seconds with return value 0
Press any key to continue . . .
```

Flowchart:



Algorithm:

Step 1: Start.

Step 2: Initialize n as 7.

Step 3: Set i to 1 and repeat steps until i is less than or equal to n.

Step 4: Set j to 1 and repeat steps until j is less than or equal to i.

Step 5: Print j followed by a space.

Step 6: Increment j by 1.

Step 7: Print a newline after the inner loop ends.

Step 8: Increment i by 1.

Step 9: Print "Done by Pranjal Khatri 929".

Step 10: Stop.

Conclusion

We looked at some of the fundamental C control structures in this lab, including the for, while, and do-while looping constructs, as well as the if-else and switch decision-making constructs. With the help of the structures, programmers can design logically flowing programs that repeat or not repeat tasks according to predetermined criteria. For instance, the while and do-while loops apply to conditional iterations, but the for loop pertains to constant iterations.

The power of these constructs is increased by nested loops and control structures, which allow for complicated decision-making and repeating activities within of repetitive chores. For instance, nested if-else statements manage multi-level decision-making processes, whereas a nested for loop is best suited for multi-dimensional activities like matrix computations. With practice, we can write code structures that are effective, readable, and reusable.