CHAPTER - 4 - Loop Control Instruction

Why loops

Sometimes we want our program to execute few set of instruction over and over again for Ex:-

printing 1 to 100 ; first 100 even numbers etc

Hence loops make it easy for a programmer to tell computer that a given set of instructions must be executed repeatedly

Types of loops

Primarily there are three types of loops in C language :-

(1) while loop
(2) do - while loop
(3) for loop

we will look into these one by one

While loop

```
while (Condition is true){

    // code
    // code
}
```

⇒ This block keeps executing as long as the condition is true.

An Example

```
int i = 0
while (i < 10){
printf("The value of is is %d", i); i++;
}
```

Note:- If the condition never becomes false, the while loop keeps getting executed. Such a loop is known as an infinite loop.

Quick Quiz:- Write a program to print natural number from 10 to 20 when initial loop counter is is initialized to 0.

```
# include <stdio.h>
int main (){
while (i<21){
if (i >= 10 ){
printf ("%d\n", i);
}
i++;
}
return 0;
}
```

Increment and decrement Operators

i++ → i is increased by 1
i-- → i is decreased by 1

printf("--i = %d", --i);

This first decrements i and then prints it

printf("i--=%d", i--);
This first prints i and the decrements it

(*) +++ Operator does not Exist => Important
(*) += is compound assignment Operator just like -=, *=, /= & %= => Also Important

do-while loops:-
The Syntax of do-while loop looks like this:-

```
do {
    // code;
    // code;
} while (condition);
```

do-while loop works very similar to while loop
while → checks the condition & then Executes the code
do-while → Executes the code & then checks the condition

do-while loop = while loop which Executes at least Once

Quick Quiz

Q⇒ Write a program to print first n natural numbers using do-while loop

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

For loop

The Syntax of for loop looks like this:

```
for (initialize; test; increment
                        or decrement)
{   // code;
    // code;
    // code; }
```

initialize → Setting a loop Counter to an initial value

Test → checking a condition

Increment → Updating the loop Counter.

An Example;

```
for (i = 0; i<3; i++){
    printf ("%d", i);
    printf ("\n");
}
```

Output :-

0
1
2

Quick Quiz:- write a program to print first n natural numbers Using for loop

```
#include <stdio.h>
int main(){
    int value;
    printf("Enter a number: ");
    scanf("%d", &value);

    for(int i=1; i<=value; i++){
        printf("%d\n", i);
    }
    return 0; }
```

A case of Decrementing for loop

```
for(i=5; i; i--)
    printf("%d\n", i);
```

This for loop will keep on running until i becomes 0.
The loop runs in following steps:

(1) i is initialized to 5
(2) The condition "i" (0 or non-0) is tested
(3) The code is Executed
(4) i is decremented
(5) Condition i is checked & code Executed if its not 0
(6) & So on until i is non 0.

Quick Quiz :- write a program to print n natural numbers in reverse Order

```
#include <stdio.h>
int main(){
    int value;
    printf("Enter an integer: ");
    scanf("%d", &value);

    for(int i=value; i; i--){
        printf("%d\n", i);
    }
return 0;
}
```

The Break Statement in C

The break Statement is used to Exit the loop irrespective of whether the Condition is true or false whenever a "break" is encountered inside the loop. the control is sent outside the loop

let us see this with the help of an Example

```
for(i=0; i<1000; i++){
    printf("%d\n", i);
    if(i==5){
        break;
    }
}
```

Output ⇒ 0
1
2
3
4
5

The Continue Statement in C

The Continue Statement is Used to immideately move to the next iteration of the loop

The control is taken to the next iteration thus skipping Everything below "continue" inside the loop for that iteration

Let us look at an Example

```
int skip = 5;
int i=0;

while (i<10){
  if (i != skip)
      continue;
  else
      printf("%d", i);
}
```

Output ⇒ 5 and not 0 - - - 9

Note

(1) Sometimes the name of the variable might not indicate the behaviour of the program
(2) break Statement Completely Exits the loop
(3) Continue Statement Skips the particular iteration of the loop.

Chapter 4 - Practice Set

Q1) write a program to print multiplication table a given number n.

```
#include <stdio.h>
int main(){
    int value;
    printf("Enter an integer: ");
    scanf("%d", &value);

    for(int i=1; i<=10; i++){
        for(int j=0; j<1; j++){
            printf("%d x %d = %d\n", value, i,
                                    value*i);
        }
    }
    return 0;
}
```

Q2) Write a program to print multiplication table of 10 in reversed order

```
#include <stdio.h>
int main(){
    int value;
    printf("Enter a value: ");
    scanf("%d", &value);

    for(int i=10; i>0; i--){
        for(int j=0; j<1; j++){
```

printf("%d x %d = %d\n", value, i, value*i);
}
}
return 0;
}

Q3 A do while loop is executed
(1) at least once
(2) at least twice
(3) at most once
⇒ (1) at least once

Q4 What can be done using one type of loop can also be done using the other two types of loops (True or False)

⇒ True

Q5 Write a program to sum first ten natural numbers using while loop.

⇒
```
#include<stdio.h>
int main(){
    int value, i=1, sum=0;
    printf("Enter a number: ");
    scanf("%d", &value);

    while(i<=value){
        sum+=i;
        i++;
    }
```

printf("Sum of natural numbers till %d is %d\n",
value, Sum);
return 0;
}

CS6 Write a program to implement program 5 using for and do-while loop

3) for loop

```
#include <stdio.h>
int main(){
    int value, Sum=0;
    printf("Enter a number: ");
    Scanf("%d", &value);

    for(int i=1; i<=value; i++){
        Sum+=i }

printf("Sum of natural numbers till %d is %d\n",
value, Sum);

return 0;
}
```

Do-while

```
do {
    Sum+=i }
while(i<=value)
```

=) the do-while Syntax.

Q7) write a program to calculate the sum of the numbers occuring in the multiplication table of a n (consider n×1 to n×10)

```
#include <stdio.h>
int main(){
    int value, i=1, sum=0;
    printf("Enter a number: ");
    scanf("%d", &value);

    for(i=1; i<=10; i++){
        sum+=value*i;
    }
    printf("The sum of the table of %d is %d\n",
                                value, sum);
    return 0;
}
```

Q8) writ a program to calculate the factorial of a given number using a for loop.

```
#include <stdio.h>
int main(){
    int value, mul=1;
    printf("Enter a number: ");
    scanf("%d", &value);

    for(int i=1; i<=value; i++){
        mul*=i; }
    printf("The factorial of %d is %d\n",
                                value, mul);
    return 0; }
```

Q89) Write a program to check whether a given number is prim or not using loops

```
#include <stdio.h>
int main() {
    int value;
    int is_prime = 1;
    printf("Enter a number: ");
    scanf("%d", &value);

    if (value <= 1) {
        printf("The number is not prime\n");
        return 0;
    }
    for (int i = 2; i < value; i++) {
        if (value % i == 0) {
            is_prime = 0;
            break;
        }
    }
    if (is_prime)
        printf("The number is a prime number\n");
    else {
        printf("The number is not a
                prime number\n");
    }
    return 0;
}
```