Ch. 4
Loop Control Instruction

Why Loops:

Sometimes we want our programs to execute few set of instructions over & over again for e.g. printing 1 to 100, first 100 even nos. 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 3 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
}

The blocks keep ⇒ repeating as long as the condition is true

For e.g.

```
#include<stdio.h>

int main(){
    int a;
    scanf("%d", &a);
    while(a<10){
        printf("%d\n", a);
        a++;
    }

    return 0;
}
```

Ans

(i) 0 1 2 3 4 5 6 7 8 9

For e.g (i) 0 → ∵ 0 is less than 10 & the condition is true & as we have a++ the compiler then checks 1, 2, 9

(ii) 80

(ii) prints nothing

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

The loop counter need not be int, it can be float as well

Increment & decrement operators

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

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

This first decrements i & then prints it

For e.g.

```
#include <stdio.h>

int main(){
    int i=5
    printf("The value ~~of~~ after i++ is %d\n",
                                        ++i);

// i++ ~~//~~ Pehle print fer increment
// ++i ~~//~~ Pehle increment fer print

printf("The value of i is %d\n", i);
    return 0;
}
```

Ans:
The value after i++ is 6
The value of i is 6

If it would have been i++ at place of ++i then: The value after i++ is 5

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

***) +++ operater does not exist

**) += is compound assignment operator just like -=, *=, /=, &%.=

do-While loop

```
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.

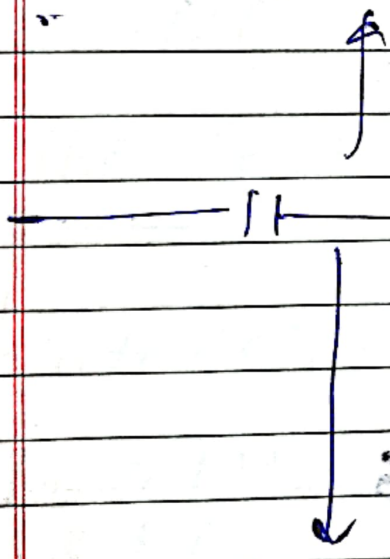
For-e-g. Basic

Assume basic structure

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

Ans

The value of i is 0
1
2
3
4
5
6
7
8
9



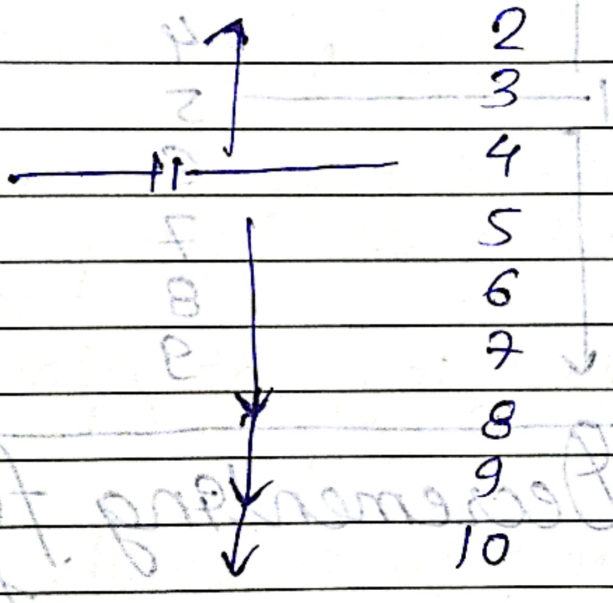
Quick Quiz = Write a program to print first n natural numbers using do-while loop

```
int i=0;
int n;
printf("Enter the value of n \n");
scanf("%d", &n);

do{
   printf("The value of
   printf("The number is %d \n", i+1);
     i++;
   }while(i<n);
```

Ans Enter the value of n → [ ] for. e.g. 10

The number is 1



for loop

The syntax of for loop looks like this:

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

  }
```

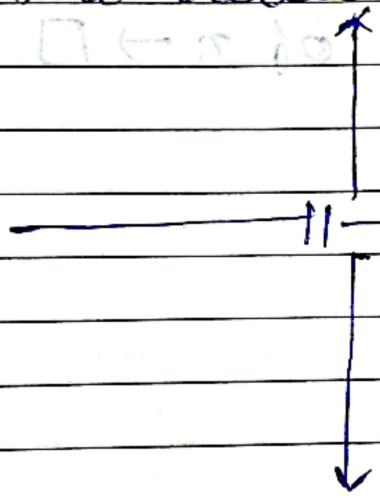
Initialize ⇒ Setting a loop counter to an initial value
Test → checking a condition
Increment → Updating the loop counter

For e.g.

```
for(int a=0; a<10; a++){
printf("The value of a is %d \n", a+1);
}
```

Ans:

The value of a is 1
2
3
4
5
6
7
8
9



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 on following steps:

1) i is initialized to 5
2) The condition "i" (0 or none) is tested
3) The code is executed
4) i is decremented
5) Condition i is checked & code is executed if it's not zero.
6) & so on until i is non-zero.

Write a program to print n natural numbers in reverse order

Code: int n;
printf("Enter the value of n \n");
scanf("%d", &n);

for(int i=n; i; i--){
printf("The value of i is %d \n", i);
}

Ans:
Enter the value of n
3 for 3

The value of i is 3
" " 2
" " 1

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.

```
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 immediately move to next iteration of the loop.
The control is taken to the next iteration thus skipping everything below "continue" inside the loop for that iteration

```
int skip=5;        → Here skip does not mean skip
int i=0;

while(i<10){  i++;
    if(i!=skip)
        continue;        Output ⇒ 5 & not 0-to 9
    else{
        printf("%d", i);
    }
```

Notes:

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.

Practice Set → 4

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

```
for(i=10; i; i--){
    printf("10 x %d = %d", i, 10*i);
#include <stdio.h>

int main(){
    printf("Multiplication table of 10\n");
    for(int i=10; i; i++){
    printf("10x%d=%d\n", i, 10*i);
    }

return 0;
}
```

Ans: Multiplication table of 10

10 x 10 = 100
10 x 9 = 90
10 x 8 = 80
⋮
10 x 1 = 10

Q2) A do while loop is executed atleast once.

Q3) What can be done using one type of loop can also be done using the other two types of loops - True or False?
→ True

Q4 Write a program to sum first ten natural numbers using while loop:

```
#include <stdio.h>

int main() {
    int i, sum=0, n=10;

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

    return 0;
}
```

Ans: The value of sum 1 to 10 is 55

Q5 Write a program to calculate the factorial of a given number using a for loop

```
# include <stdio.h>

int main() {
    int i=0, n=3, factorial=1;
    for (i=1; i<=n; i++) {
        factorial *= i;
    }
    printf("The value of factorial %d is %d", n, factorial);

    return 0;
}
// You can change the value of n to get factorial of n
```

for e.g. If n=3 → Output → The value of factorial of 3 is 6

n=7 → The value of factorial of 7 is 5040

Q6 Write a program to check whether a given number is prime or not using loops

Code:

```
# include <stdio.h>
int main() {
    int n=5, prime=1;
    for (int i=2; i<n; i++) {
```

```
        if (n % i == 0) {
            prime = 0;
            break;
        }
    }
    if (prime == 0 && n != 2) {
        printf("This is not a prime no.");
    }
    else {
        printf("This is a prime no.");
    }
    return 0;
}
```

// You can change value of n & check whether the number is prime or not

Ans:

// As n = 5

This is a prime no..

for n = 4

This is not a prime no..

PROJECT 1: No. GUESSING GAME

We will write a program that generates a random number & asks the player to guess it. If the players guess is higher than the actual number the program displays "Lower number please!

Similarly if the users guess is too low, the program prints "Higher number please".
When the user guesses the correct no., the program displays the number of guesses the player took to arrive at the no.
Hint: Use loops
Use a random number generator

```
# Include <stdio.h>

int main(){
    int number;
    number = rand();
```

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

int main(){
  int number, guess, nguesses=1;
  srand(time(0));
number = rand()%100+1;/* Generates a
                    random number bet^n 1&100*/

// printf("The number is %d\n", number);

/* Keep running the loop until the no.
                              is guessed */
```

```
do{
  printf("Guess the number betn 1 & 100\n");
  scanf("%d", &guess);

if(guess>number){
  printf("Lower number please!\n");
  }

else if(guess<number){
  printf("Higher number please!\n");
  }

else{
  printf("You guessed it in %d attempts\n",
                                nguesses);
  }
  nguesses++;
}while(guess!=number);

  return 0;
}
```

Ans:

Guess the number betn 1 to 100
45
Higher ng. please!
Guess the number betn 1 to 100
86
You guessed it in 2 attempts