Loop Control Structure/Decision making and Looping:

- We frequently need to perform an action over and over, often variations in the details each time. The mechanism which meets this need is a loop.
- So loop is a technique by using which we can repeat a part of a program.
- There are three types of loops:
  - While loop
  - Do-while loop
  - For loop.

While Loop:

- The while is an entry-controlled loop statement. The test-condition is evaluated and it the condition is true, then the body of the loop is executed.
- **Syntax:**

```
initialize loop counter
while(condition)
{
        statement(s);
        increment loop counter;
}
```

- **Example1:**

```
main()
{
        int i;
        i=0;
        while(i<10)
        {
                printf("Ram");
                i++;
        }
}
```

- **Output:** Ram Ram Ram Ram Ram Ram Ram Ram Ram Ram
- **Example2:**

```
main()
{
        int i;
        i=1;
        while(i<=32767)
        {
                printf("%d\n",i);
                i++;
        }
}
```

- **Output:** 1, 2, . . ., 32767, -32768, -32767, . . ., -2, -1, 0.
- **Example3:**

```
main()
{
        int i;
        i=1;
        while(i<=10);
```

```
        {
                printf("%d\n,i");
                i++;
        }
}
```

- **Output:** No output since there is a semi-colon after while.

Do-while Loop:

- It is an exit-controlled loop and therefore the body of the loop is executed at least once.
- **Syntax:**

```
initialize loop counter
do
{
        statement(s);
        increment loop counter;
}
while(condition);
```

- **Example:**

```
main()
{
        int i;
        i=0;
        do
        {
                printf("Ram");
                i++;
        }while(i<10);

}
```

- **Output:** Ram Ram Ram Ram Ram Ram Ram Ram Ram Ram
- **Difference between while and do-while loop**.
    - The while loop tests the condition before executing any of the statements within the while loop.
    - Do-while tests the condition after having executed the statements within the loop.
    - That means do-while would execute its statements at least once, even if the condition fails for the first time itself.
    - The while loop on the other hand will not execute its statements if the condition fails for the first time.
- **Example:**

```
main()
{
        int i;
        i=0;
        do
        {
                printf("Ram");
                i++;
        }while(i>10);
}
```

- **Output:** Ram
- **Example:**

```
main()
{
        int i;
        i=0;
        while (i>10);
        {
                printf("Ram");
                i++;
        }

}
```

- **Output:** Ram

For Loop:

- It is another entry-controlled loop.
- The for loop allows us to specify three things about a loop in a single line:
  - Setting a loop counter to an initial value.
  - Testing a loop counter to determine whether its value has reached the number of repetitions desired.
  - Increasing the value of the loop counter each time the program segment within the loop has been executed.
- **Syntax:**

```
for(initialise counter;test condition;increment counter)
{
        statement(s);
}
```

- **Example1:**

```
main()
{
        int i;
        i=0;
        for(i=1;i<=10;i++)
        {
                printf("%d\n",i);
        }
}
```

- **Output:** 1 to 10 each in one line separately.
- A for loop of the following type are allowed:
  -
```
i=1;
for( ;i<=10;i++)
        printf("%d\n",i);
```
  -
```
for(i=1;i<=10;)
{
        printf("%d\n",i);
        i++;
}
```
  -
```
i=1;
for(;i<=10;)
```

```
{
        printf("%d\n",i);
        i++;
}
```

- ```
  i=1;
  for(;i++<=10;)
  {
          printf("%d\n",i);
  }
  ```
- ```
  for( ; ;)
          printf("Hello\n");
  ```

- A for loop can contain multiple initialisations.

```
for(i=1,j=10;i<=10;i++)
```

- A for loop can contain multiple increment/decrement statements.

```
for(i=1,j=10;i<=10;i++,j+=2)
```

Odd Loop:

- In real life sometime it is not known how many times the statements of the loop are to be executed. This type of loop is known as the odd loop.
- **Example:**

```
main()
{
        char ch='y';
        while(ch=='y')
        {
                ……..
                ……..
                printf("Do you want to continue[y/n]:");
                scanf("%d",&ch);
        }
}
```

- **OR**

```
main()
{
        char ch='y';
        for(;ch=='y';)
        {
                ……..
                ……..
                printf("Do you want to continue[y/n]:");
                scanf("%d",&ch);
        }
}
```

Nested Loop:

- Like if statements can be nested, similarly while and fors can also be nested.
- **Example:**

```
main()
{
        int i;
```

```
        i=0;
        for(i=1;i<=5;i++)
        {
                for(j=1;j<=5;j++)
                {
                        printf("%d\t",i+j);
                }
                printf("\n");
        }
}
```

- **Output:**

2	3	4	5	6
3	5	7	9	11
4	7	10	13	16
5	9	13	17	21
6	11	16	21	26