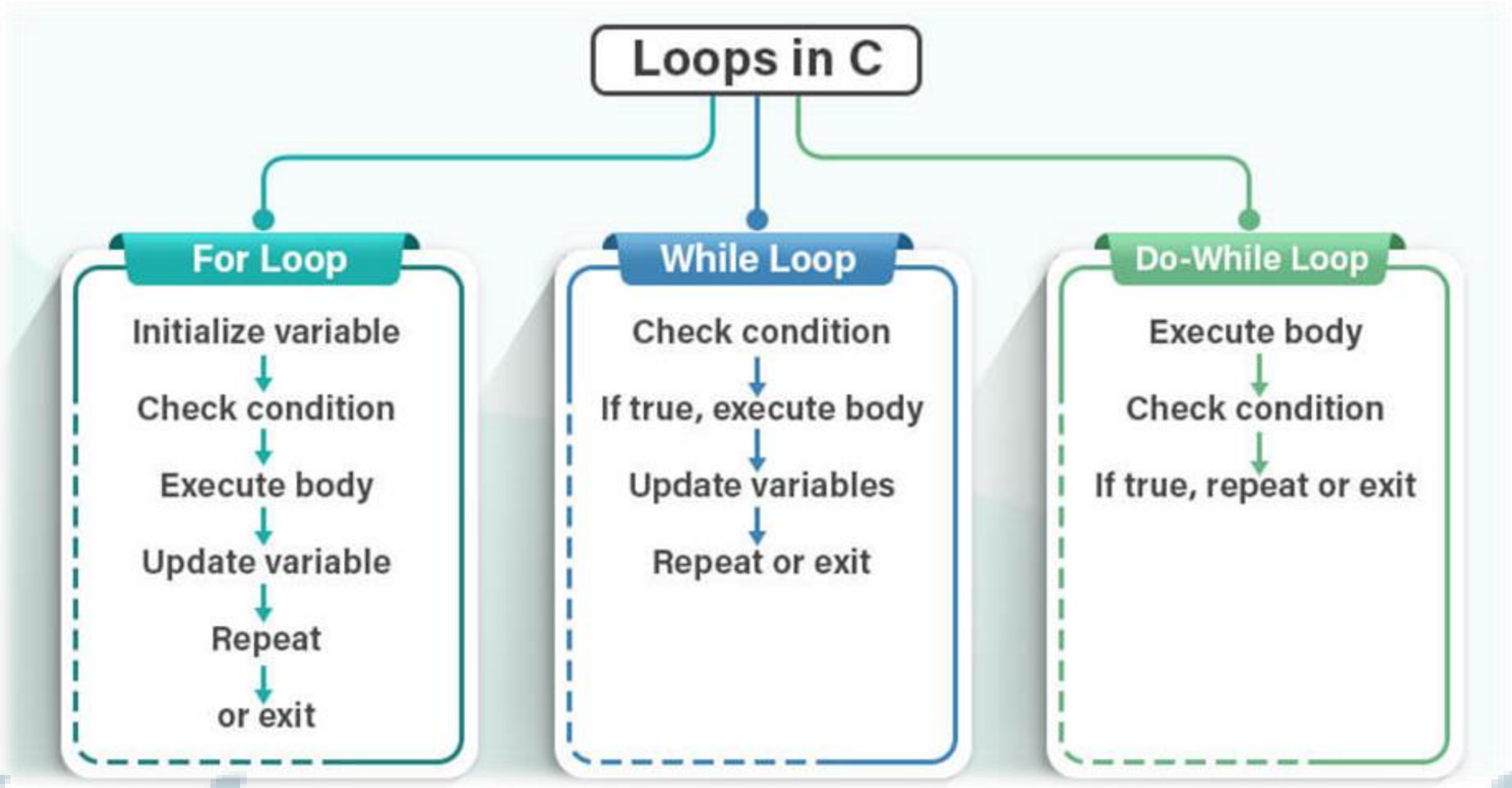
CHAPTER 08

Iteration Control Structures (loops)

Iteration control structure

- The third type of control structure is iteration control structure.
- Iteration is also referred as a loop, because the program will keep repeating the activity until the condition becomes false.
- C provides following three types of iteration structures:
  - for loop
  - while loop
  - do while loop

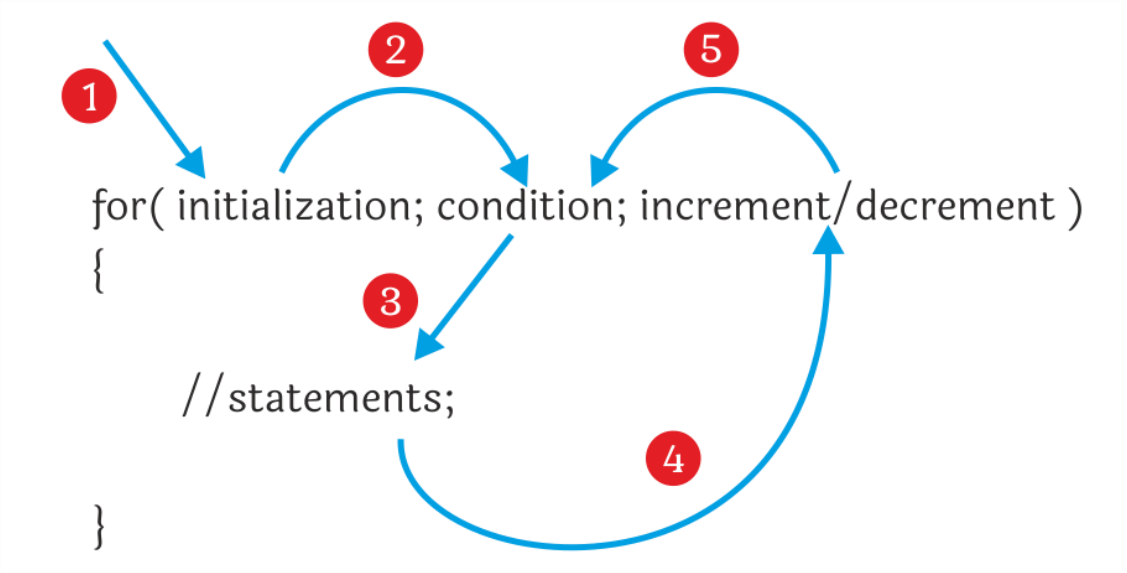
Iteration control structure



for Loop

The for loop is used to repeat a statement or a block of statements for a specified number of times.

Syntax



for Loop

Start/Initial value

End/Final Value

Increment of loop variable

Loop variable

keyword

```
for(int i = 1 ; i < 10; i++)
```

Loop condition

Body of the loop

```
{
    printf("i = %d", i);
}
```

for Loop

- **Initialization:**
  - Expression which specifies the starting value of counter variable
  - Executed only once
- **Condition:**
  - Relational Expression on the basis of which the statement(s) are executed or not
- **Increment/decrement:**
  - Part of loop that specifies change in counter variable after execution of the loop
- **Body of the loop:**
  - Statement or group of statements in braces to be repeated

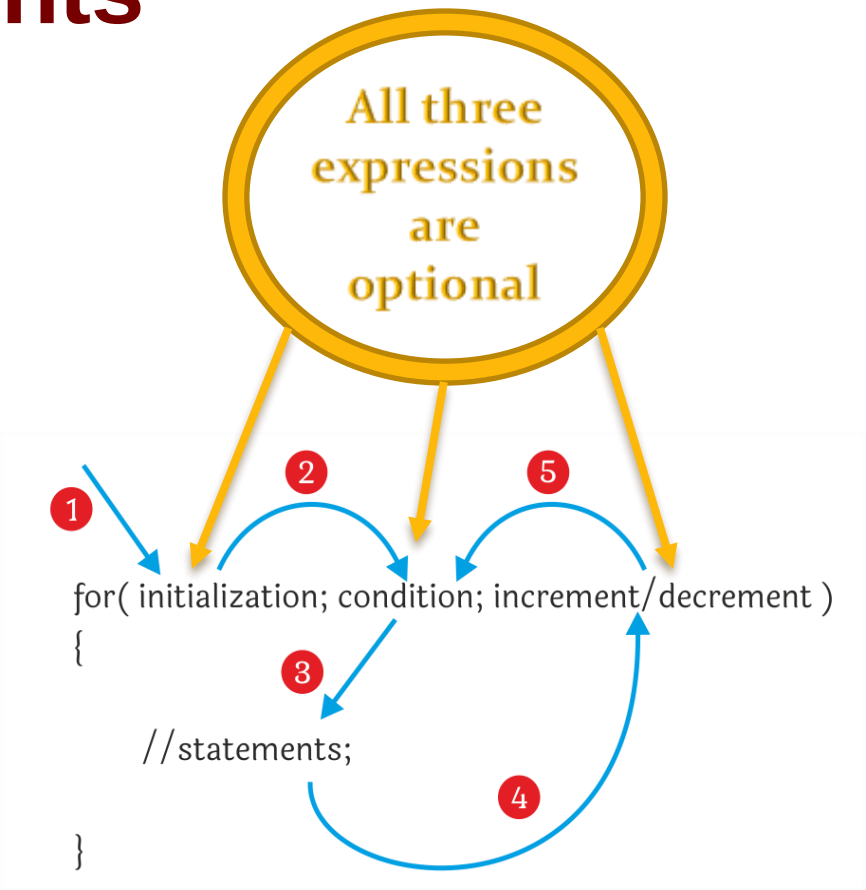
for Loop


```
for ( a = 1;        a < 5;        a ++ )
 {
  printf( "%d", a );
 }
```

a	Output
1	

Valid for statements

- for(; ;)
- for(int i=1 ; ;)
- for(; k<10 ;)
- for(; ; k++)
- for(; x<12 ;)



The for loop variations

- The initialize expressional and increment expression of the for loop can also contain more than one statement separated buy a comma. For instance,

```
For (j=0, k=100; k-j>0; j++, k--)
```

- In this example, both j and k are initialized before the loop is entered,. After each iteration, j is incremented and k is decremented.

Example: Counting from 1 to 10

```
#include <stdio.h>
int main()
{
    int i;
    /* The loop iterates while i < 10, and i increases by one after every
       iteration*/
    for ( i = 0; i < 10; i++ ) {
        printf( "%d\n", i );
    }
}
```

The following program continues to iterate until characcter q is entered from the keyboard. Instead of testing the repeat condition, the test expression in this for loop checks the value of a character entered by the user.

```
#include<stdio.h>
#include<conio.h>
 void main(void)
{
        char ch;
        for ( ; ch != 'q';  )
        {
                puts("Enter any Character");
                ch = getch();
                printf("you entered: %c\n",ch);      }
}
```

Write a program which uses for loop to print out a table of ASCII codes from 32 to 127

```
#include<stdio.h>
#include<conio.h>
int main(void)
{
    int i;
    for(i=32; i<128 ; i++)                    /* code from 32 to 127 */
        printf("%3d = %c\t", i, i);           /* print as number and as char */
 }
```

Write a program to print the sum of odd numbers from 1 to 100;

i.e.,

Sum = 1 +3 + 5+ … + 99

```
#include<stdio.h>
int main(void)
{
        int i, sum=0;
        for(i=1; i<100; i=i+2)
        {
                sum = sum + i;
        }
        printf("\n\tThe sum is = %d", sum);
}
```

Write a program to calculate average of a list of numbers

```
#include <stdio.h>
int main()
{
    int i, n, sum = 0;
    float avg;
    printf("Input the 10 numbers : \n");
    for (i = 1; i <= 10; i++) {
        printf("Number-%d :", i);
        scanf("%d", &n);
        sum += n;
    }
    avg = sum / 10.0;
    printf("The sum of 10 no is : %d\nThe Average is : %f\n", sum, avg);
}
```

Write a program in c to print multiplication table

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

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

Practice

- Program that prints the given sequences

  0 5 10 15 20 25 30

  30 27 24 21 18 15 12

- Program to print odd numbers from 1 to 50
- Program to print even numbers ranging from n1 to n2 (n1 > n2)
- Program to print squares of all numbers from 1 to 10
- Program to print sum of squares of all numbers from 1 to 10

Problem: Find factorial of a given number

Planning the solution:

Input: Given number N

Output: Factorial of a given number

Processing: Find product of all numbers from 1 to the given number

Algorithm:

Step 1 Start

Step 2 Let NUM = 5
Initialize F = 1, k=1

Step 3 Repeat Step 4 and 5 WHILE k<=NUM

Step 4 Set F = F * k

Step 5 Increment k (k = k + 1)
End WHILE

Step 6 Output F

Step 7 Stop

Program to find factorial of a given number

```c
//Find factorial of a number
#include<stdio.h>
int main(void)
{
    int n,j,fact;
    printf("\nEnter number: ");
    scanf("%d", &n);
    fact=1;
    for(j=1;j<=n;j++)
    {
        //printf("%d  x %d = %d \n ",j,fact,fact*j);
        fact=fact*j;
    }
    printf("Factorial is %d", fact);
}
```

```
Enter number: 5
Factorial is 120
--------------------------
```

while loop

- while loop can be addressed as an **entry control** loop.
- Used to implement repetition when number of iterations is not known in advance and repetition continues until the test condition is true

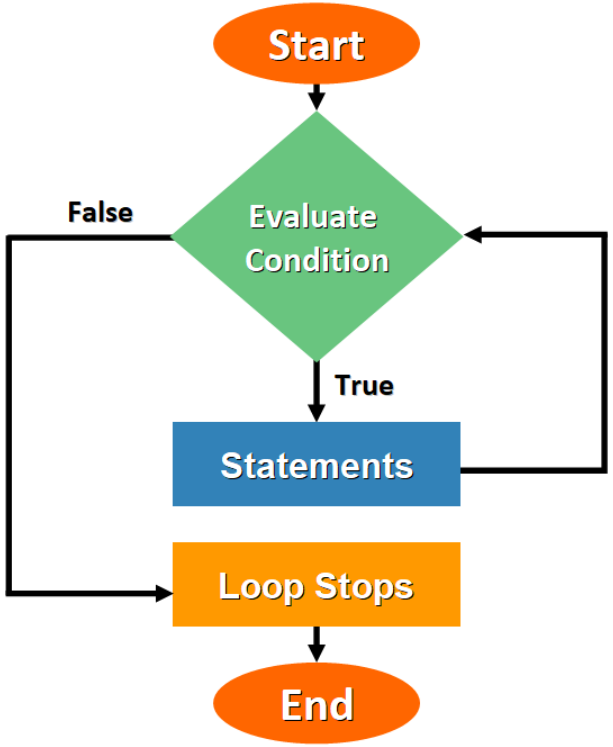
Condition

Any expression that evaluates to true or false

```
while (condition) {
    statement
    statement
    ...
}
following_statement
```

Loop body

It is executed as long as the condition is true



Example: Program to print first 10 natural numbers

```
#include<stdio.h>
void main()
{
    int x;
    x =1;
    while(x <=10)
    {
        printf("%d\t", x);  /* below statement means, do x = x+1, increment x by 1*/
        x++;
    }
}
```

The following program continues to iterate until character q is entered from the keyboard. Instead of testing the repeat condition, the test expression in this for loop checks the value of a character entered by the user.

```
#include<stdio.h>
#include<conio.h>
 void main(void)
{
    char ch;
  printf(“\nEnter a character: ");
  ch = getche();

  while(ch != 'q')
  {
    printf("\nInput character = %c", ch);
  ch = getche();
}
}
```

Program to print natural numbers from 1 to 10 in descending order

```c
#include<stdio.h>
int main()
{
    int n = 10;
    while(n ! = 0)
    {
        printf("%d\n" , n);
        n--;
    }
}
```

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

Practice

- Program that converts kg to pounds using while loop. Zero signals the end of input.
- Program to print the ASCII table (0-255)
- Program to print all lower case letters in reverse order on a single line using while loop
  (Lower case letters: 97-122)
- Program that reads a number and print its table using while loop

do while loop

General syntax is,

```
do
{
..........
}
while(condition);
```

do while loop

- In some situations it is necessary to execute body of the loop before testing the condition.
- Such situations can be handled with the help of do-while loop.
- do statement evaluates the body of the loop first and at the end, the condition is checked using while statement.
- It means that the body of the loop will be executed at least once, even though the starting condition inside while is initialized to be **false**.

Example: Program to print first 10 multiples of 5.

```
#include<stdio.h>
void main()
{
int a,i;
a =5;
i=1;
do
{
printf("%d\t", a*i);
i++;
}
while(i<=10);
}
```

output

5 10 15 20 25 30 35 40 45 50

Write a program to calculate average of a list of numbers using while loop:

```
#include<stdio.h>
#include<conio.h>
 void main(void)
{
int n, count = 1;
  float x, average, sum = 0;
        printf("How many numbers? ");
        scanf("%d", &n);
  do
    {
          printf("X = ");
          scanf("%f", &x);
     sum += x;
     ++count;
      }
  while(count <= n);
  average = sum/n;
  printf("\nThe average is %f", average);
  getch();
}
```

Jumping Out of Loops

- Sometimes, while executing a loop, it becomes necessary to skip a part of the loop or to leave the loop as soon as certain condition becomes **true**.

- This is known as jumping out of loop.

1) break statement

2) continue statement

1) break statement

- When break statement is encountered inside a loop, the loop is immediately exited and the program continues with the statement immediately following the loop.

```
while( condition check )
{
    statement-1;
    statement-2;
    if( some condition)
    {
        break;
    }
    statement-3;
    statement-4;
}
```

Jumps out of the loop, no matter how many cycles are left, loop is exited.

Example:

```
#include <stdio.h>
#include <conio.h>
void main(void)
{
int x;
        for(x=1;   ;x++)
        {
        printf("%d\n",x);
                if (x==10)
                break;
         }
        printf("End of Loop");
}
```

2) continue statement

It causes the control to go directly to the test-condition and then continue the loop process.

On encountering continue, cursor lea2ve the current cycle of loop, and starts with the next cycle.

```
while( condition check )
{
    statement-1;
    statement-2;
    if( some condition)
    {
        continue;
    }
    statement-3;
    statement-4;
}
```

Jumps to the next cycle directly.

Not executed for the cycle of loop in which continue is executed.

Example: Calculating some of all even numbers between 1 to 10

```
#include <stdio.h>
#include <conio.h>
void main(void)
{
 int x,sum=0;
        for(x=1; x<=10 ;x++)

{
        if (x%2 != 0)
        continue; // skips rest part of loop
        printf("%d\n",x);
        sum+=x;
        }

        printf("Sum of all even numbers between 1-10=%d",sum);
} // end of main()
```

Nested for loops

- When one for command is performed within another, they are said to be nested.
- The inside loop is completely repeated for each repetition of the outside loop.

Nested for loops

```
for(i=1;i<=3;i++)
{
    for(j=1;j<=3;j++)
    {
        -----
    }
}
```

Outer Loop

Inner Loop

9

Outer Loop Variable - i

Inner Loop Variable - j

For each run of the outer loop, the inner loop runs maximum number of times.

i	j
1 -->	1 to 3
2 -->	1 to 3
3 -->	1 to 3

Nested for loops

- Write a program to print product of numbers as given below:

1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12

Nested for loops

```c
#include<stdio.h>
int main()
{
    int j, k, prod=1;
    for(j=1; j<=3; j++)
        for(k=1; k<=4; k++)
        {
            prod = j * k;
            printf("\n\t%d x %d = %d",j,k,prod);
        }
}
```

```
        1 x 1 = 1
        1 x 2 = 2
        1 x 3 = 3
        1 x 4 = 4
        2 x 1 = 2
        2 x 2 = 4
        2 x 3 = 6
        2 x 4 = 8
        3 x 1 = 3
        3 x 2 = 6
        3 x 3 = 9
        3 x 4 = 12
--------------------------------
```

Nested for loops

- Write a program that prints a table of squares of all numbers from 1 to 10 as shown below

Number	Square
1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
10	100

Half Pyramid of *

```
*
* *
* * *
* * * *
* * * * *
```

```
#include <stdio.h>
int main() {
for (i = 1; i <= 5; ++i)
 {
      for (j = 1; j <= i; ++j)
 {
          printf("* ");
      }
      printf("\n");
   }
   return 0;
}
```

Half Pyramid of Numbers

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

```
#include <stdio.h>
int main() {
   int i,j;
for (i = 1; i <= 5; ++i) {
      for (j = 1; j <= i; ++j) {
         printf("%d ", j);
      }
      printf("\n");
   }
   return 0;
}
```

Inverted half pyramid of *

```
* * * * *
* * * *
* * *
* *
*
```

```
#include <stdio.h>
int main() {
   int i, j;
for (i = 5; i >= 1; --i) {
      for (j = 1; j <= i; ++j) {
         printf("* ");
      }
      printf("\n");
   }
   return 0;
}
```

inverted half pyramid of numbers

```
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

```
#include <stdio.h>
int main()
 {
   int i, j;
for (i = 5; i >= 1; --i)
 {
      for (j = 1; j <= i; ++j)
 {
         printf("%d ", j);
      }
      printf("\n");
   }
   return 0;
}
```

Practice these patterns

```
12345
1234
123
12
1
```

```
1
12
123
1234
12345
```

```
54321
4321
321
21
1
```

```
55555
4444
333
22
1
```

```
*****
****
***
**
*
```

```
1
22
333
4444
55555
```

```
*
**
***
****
*****
```

```c
#include <stdio.h>
  int main(void) {
      int n;
  printf("Enter the number of columns");
  scanf("%d",&n);
  //printing the upper part of the pattern..
 for(int i=0;i<n;i++)
 {
   for(int j=0;j<i;j++)
   {
      printf(" ");
   }
   for(int k=1;k<=n-i;k++)
   {
    printf("*");
   }
   printf("\n");
 }
   //printing the lower part of the pattern..
for(int i=1;i<n;i++)
{
  for(int j=1;j<n-i;j++)
  {
    printf(" ");
  }
  for(int k=1;k<=i+1;k++)
  {
    printf("*");
  }
  printf("\n");
}
  return 0;
}
```

```
Enter the number of columns5
*****
 ****
  ***
   **
    *
   **
  ***
 ****
*****
```

Nested while loops:

Similar to for loop, you can use one while loop within another while loop.