Programming in C

Nepal College of Information Technology

Course Instructor: Er. Rabina Chaudhary

Loop Control Statement :

- loop statement allows us to execute a statement or group of statements repeatedly till the given condition is true
- when the condition becomes, the loop terminates and control passes on to the statements following the loop
- A loop consists of two segments:

  a. control statement

  b. body of loop

Types of loop statements:

1. Entry Controlled loop
    - i. while loop
    - ii. for loop
2. Exit Controlled loop
    - i. do while loop

1. while loop

- It is an entry control loop where a condition is checked before executing the body of a loop
- The test condition is evaluated first and if the condition is true, the body of the loop is executed
- After the body of the loop is executed once, the test condition is evaluated again and if it is true, the body of loop is executed once again
- The process of repeated execution of the body of loop continues till the condition is true
- If the condition becomes false the program control is transferred to the statement immediately following the body of loop

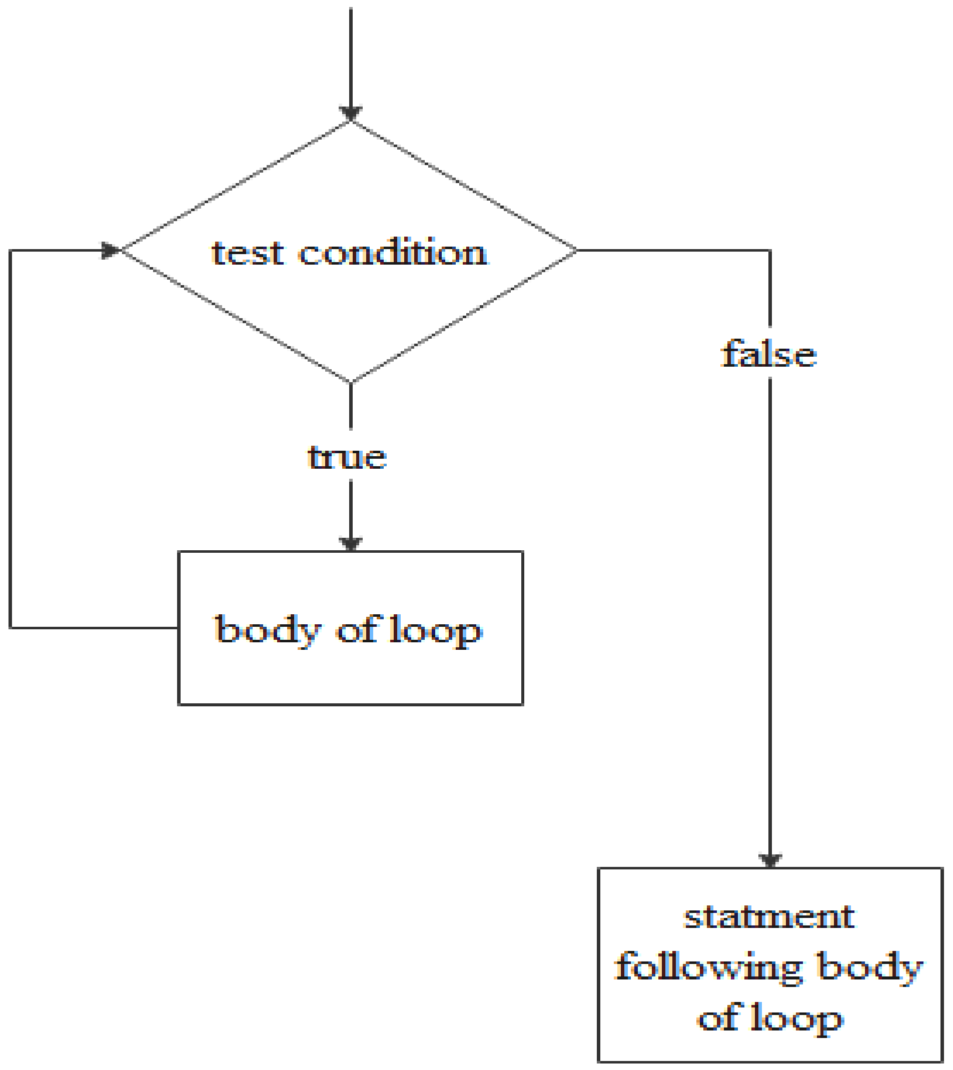
1. while loop

- Syntax:

```
while (test condition)
{
        body of loop
}
```

1. while loop

Flowchart:

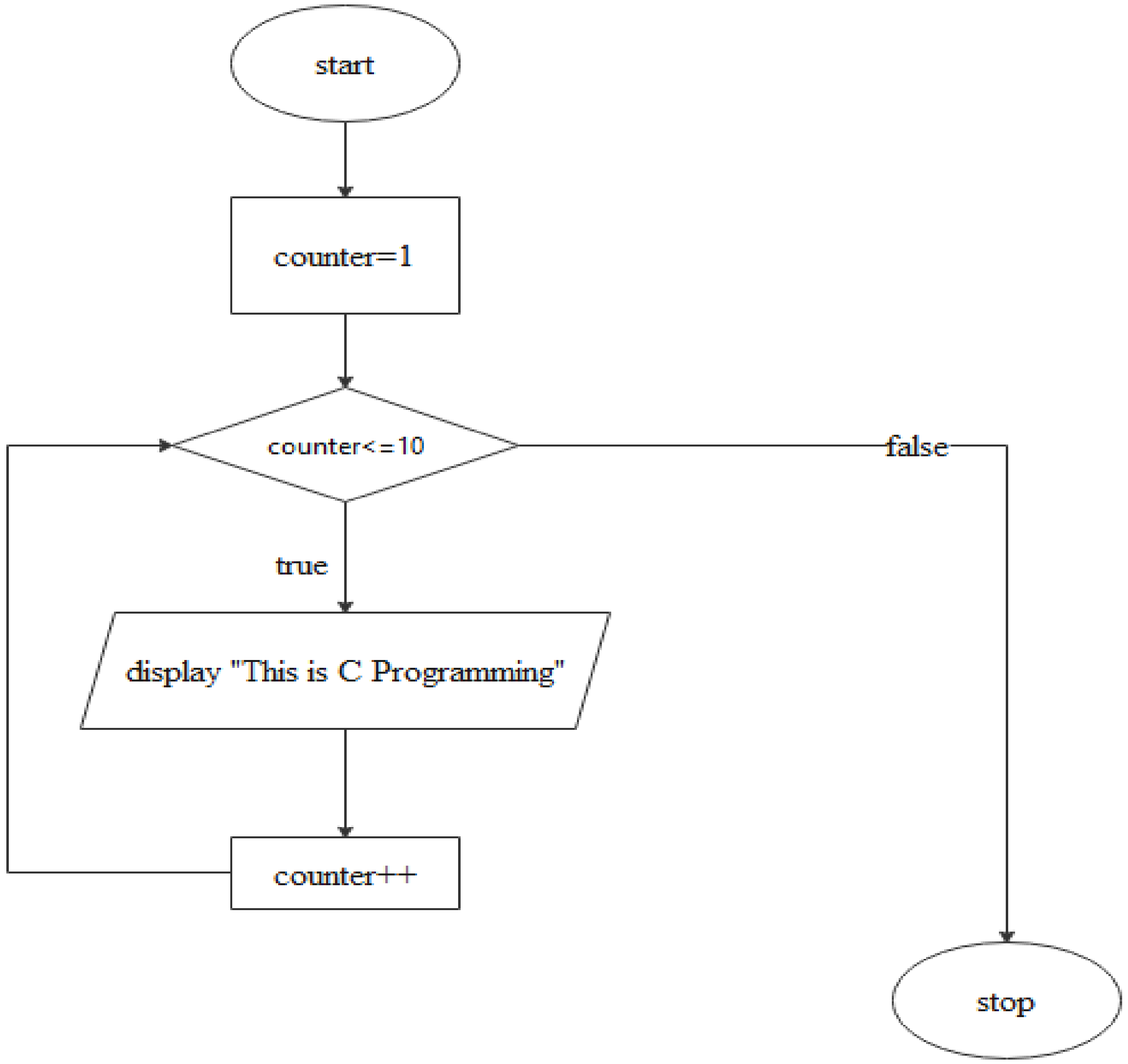


Example: Write a program to print this message “This is C programming” 10 times

Algorithm:

```
Step 1: start
Step 2: initialize counter variable to represent number of iteration to 1
Step 3: check if counter variable is less than 10
Step 4: if condition in step 3 is true, go to step 5, otherwise go to step 7
Step 5: display message “This is C Programming.” and increment the
value of counter variable by 1
Step 6: repeat step 3 to 5 until condition of step 3 is evaluated false
Step 7: stop
```

Example: flowchart to print this message “This is C programming” 10 times



Example: Write a program to print this message “This is C programming” 10 times.

```
#include<stdio.h>

int main()
{
        int counter;
        counter=1;
        while(counter<=10)
        {
                printf("This is C programming.\n");
                counter++;
        }
        return 0;
}
```

Class Work:

1. Write a program to display your name 50 times.
2. Write a program to display numbers from 50 to 500.
3. Write a program to generate multiplication table of input number.
4. Write a program to display numbers from 20 to 200 that is divisible by 5.
5. Write a program to display even numbers from 2000 to 5000.
6. Write a program to display odd numbers from n1 to n2 where n1 and n2 are numbers given by user where n2>n1.

Write a program to generate multiplication table of input number.

```
#include <stdio.h>
#include <conio.h>
int main()
{
        int number,n,counter;
        printf("Enter the number whose multiplication table is to be generated :");
        scanf("%d",&number);
        printf("Enter the range of multiplication table :");
        scanf("%d",&n);
        counter = 1;
        while(counter<=n)
        {
                printf("%d * %d = %d\n\n",number,counter,number*counter);
                counter++;
        }
        getch();
        return 0;
}
```

Tracing:

number = 9
n = 5

counter	counter<=n	num*counter=num*counter	counter++
1	1<=5 (T)	9* 1= 9	1+1=2
2	2<=5(T)	9*2=18	3
3	3<=5(T)	9*3= 27	4
4	4<=5(T)	9*4=36	5
5	5<=5(T)	9*5=45	6
6	6<=5 (F)	Loop Terminated	

Write a program to display numbers from 20 to 200 that is divisible by 5

```
#include <stdio.h>
#include <conio.h>
int main()
{
		int counter;
		counter = 20;
		while(counter<=200)
		{
			if(counter % 5 == 0)
			{
				printf("%d\t",counter);
			}
			counter ++;
		}
   getch();
   return 0;
}
```

Write a program to display odd numbers from n1 to n2 where n1 and n2 are numbers given by user where n2>n1

```
#include <stdio.h>
#include <conio.h>
int main()
{
            int n1,n2;
             printf("Enter the range to display the odd numbers");
            scanf("%d%d",&n1,&n2);
            if(n1>=n2)
            {
                        printf(" invalid input!!! ");
            }
            else
            {
                        while(n1<=n2)
                        {
                                    if(n1 % 2 != 0)
                                    {
                                                printf("%d\t",n1);
                                    }
                        n1++;
                        }
            }
   getch();
   return 0;
}
```

2. do while loop:

- It is <u>exit controlled loop</u>, the test condition is checked after executing the body of loop
- In do while loop, the body of loop is executed once and then the test condition in the while statement is evaluated
- The body of the loop is executed again if the test condition is true
- The process of executing the body of loop repeatedly continues as long as the condition is evaluated true
- When the condition becomes false, the loop is terminated and control is transferred to the statement following the loop
- <u>The body of the loop is executed at least once even if the condition is false</u>

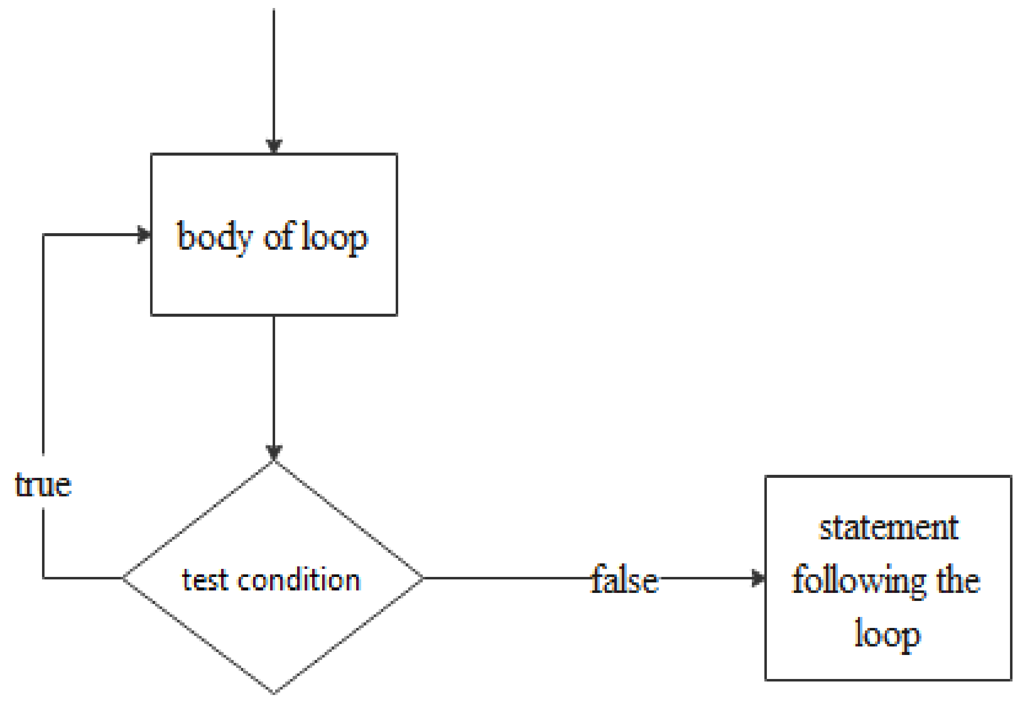
2. do while loop:

Syntax:

```
        do
        {
                body of loop
        }while (test condition);
```

do while loop:

Flowchart:



Example: Write a program to print this message “This is C programming” 10 times

Algorithm:

Step 1: start

Step 2: initialize counter variable to represent number of iteration to 1

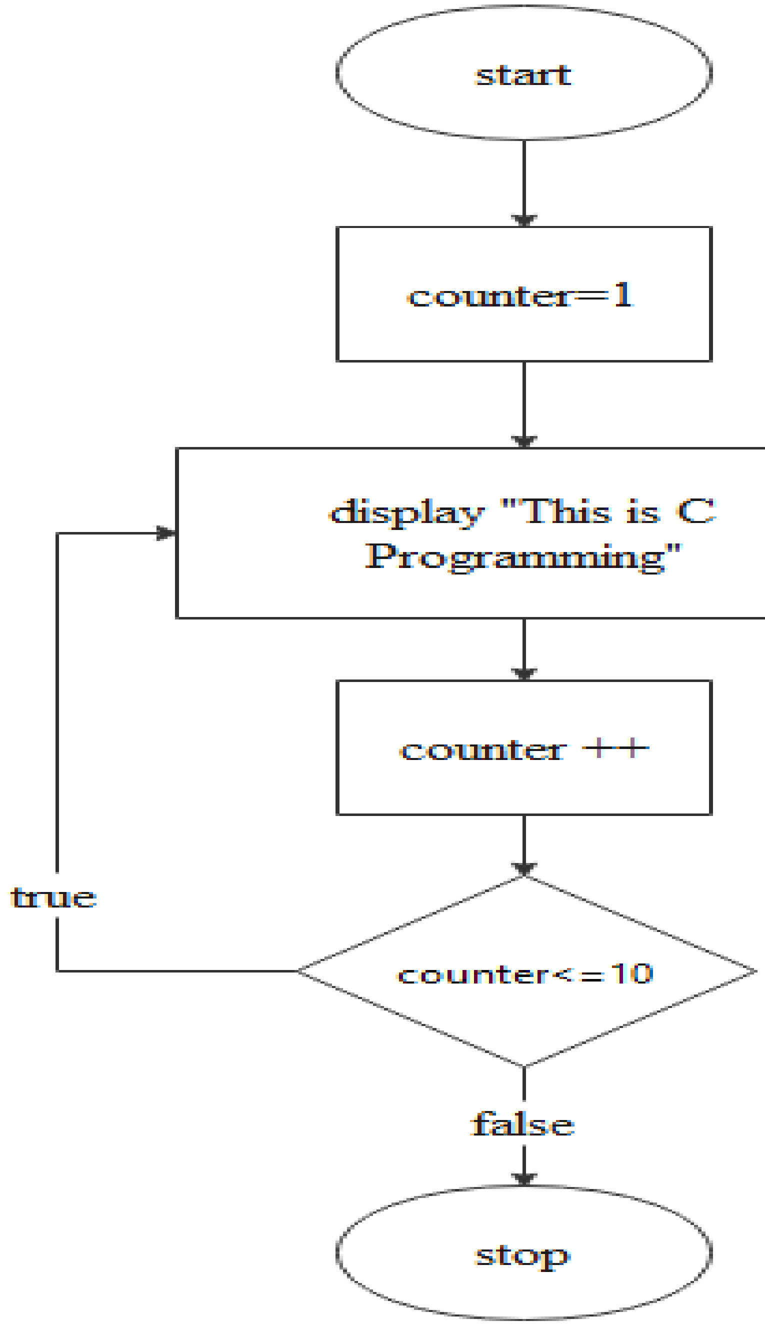
Step 3: display message “This is C Programming.” and increment the value of counter variable.

Step 4: check if counter variable is less than 10

Step 5: if condition in step 4 is true, go to step 3, otherwise go to step 6, repeat step 3 and 4 until condition of step 4 is evaluated false

Step 6: stop

Flowchart:


start
counter=1
display "This is C Programming"
counter ++
counter<=10
true
false
stop

Example: Write a program to print this message “This is C programming” 10 times

```
#include<stdio.h>

int main()
{
            int counter;
            counter=1;
            do
            {
                  printf("\nThis is C programming");
                  counter++;
            }while(counter<=10);

            return 0;
}
```

Class Work:

1. Write a program to read numbers from user until the user inputs zero. And calculate the average of the entered numbers.
2. Write a program to enter the numbers until the user wants and display the count of positive, negative and zeros entered.

Difference between while loop and do-while loop:

while loop

- It is entry controlled loop. i.e. test condition is evaluated first and body of loop is executed only if the condition is true
- Body of the loop may not be executed at all if the condition is not evaluated true at the first attempt only

do-while loop

- It is exit controlled loop. i.e. body of loop is executed once and the condition is evaluated for the repetition of loop for next time
- Body of the loop will be executed at least one time even of the test condition is not evaluated true in first attempt

Difference between while loop and do-while loop:

while loop

- Syntax:

```
while (test condition)
        {
                body of loop
        }
```

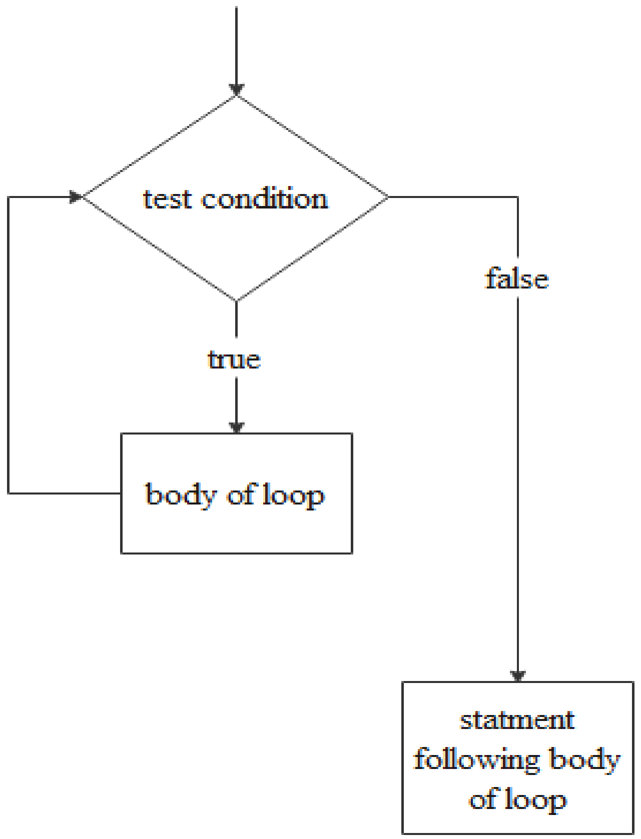
do-while loop

- Syntax:

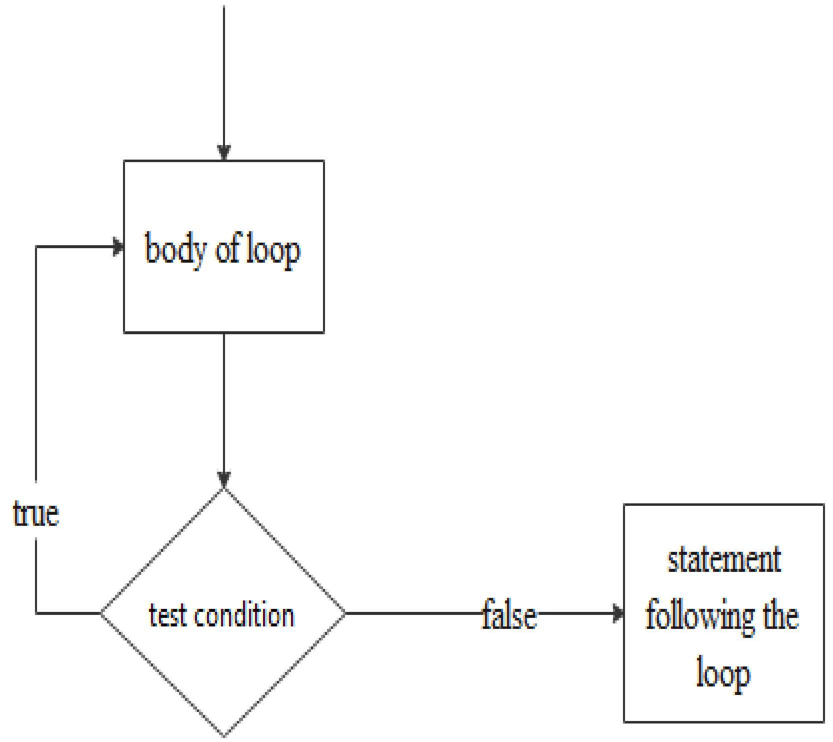
```
do
        {
                body of loop
        }
        while (test condition);
```

Difference between while loop and do-while loop:

while loop : flowchart



do-while loop : flowchart

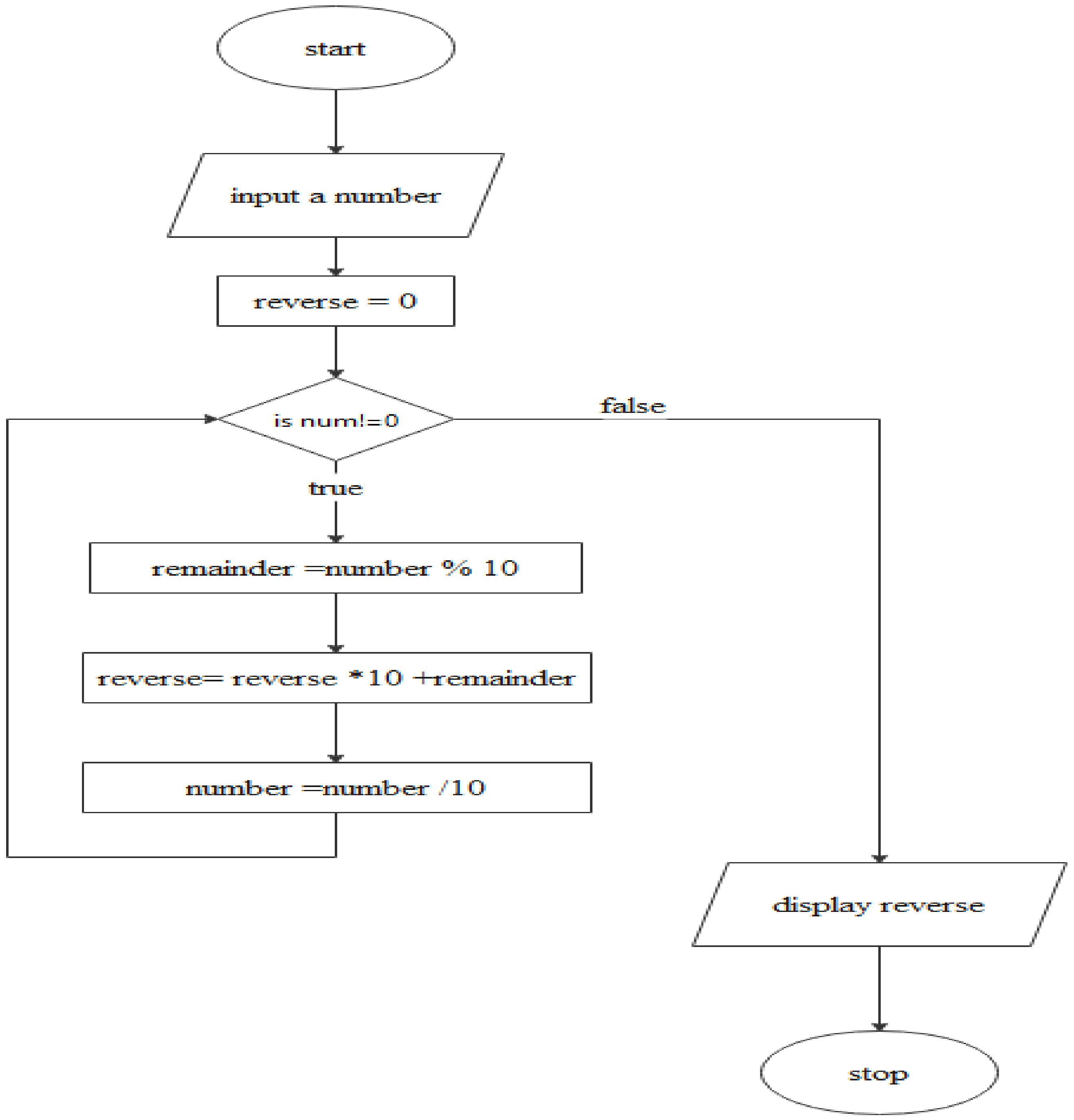


1. Write a program to reverse a given number.

<u>Algorithm:</u>

```
Step 1: start
Step 2: input a number
Step 3: initialize reverse to 0
Step 4: check if number is not equals to zero.
Step 5: repeat step 4 to step 8 till the condition of step 4 is evaluated true. Go to step 9 if
        condition is false
Step 6: isolate last digit of number using modulus operation
        last_digit=number%10
Step 7: append the last digit at the end of reverse
        reverse =reverse*10 + last_digit
Step 8: remove last digit from the number
        number =number/10
Step 9: display the reverse number
Step 10: stop
```

Flowchart:


start
input a number
reverse = 0
is num!=0
false
true
remainder =number % 10
reverse= reverse *10 +remainder
number =number /10
display reverse
stop

Write a program to reverse a given number

```
#include<stdio.h>
#include<conio.h>
int main()
{
            int number,remainder,reverse=0,num;
            printf("Enter a a number :");
            scanf("%d",&num);
             number=num;
            while(number!=0)
            {
                  remainder=number%10;
                  reverse=reverse*10+remainder;
                  number=number/10;
            }
            printf("\n\nThe reverse of %d is %d",num,reverse);
            getch();
            return 0;

}
```

Class Work:

1. Write a program to check if the given number is palindrome or not.

   (a number is palindrome if the reverse of the number is equals to the original number)

Class work

2. Write a program to find the sum of digits of a number.

(if the given number is 1234 then the sum of digits is 1+2+3+4)

Class Work:

3. Write a program to calculate sum of cube of digits of a given number.

example: number =145

sum = $1^3 + 4^3 + 5^3$

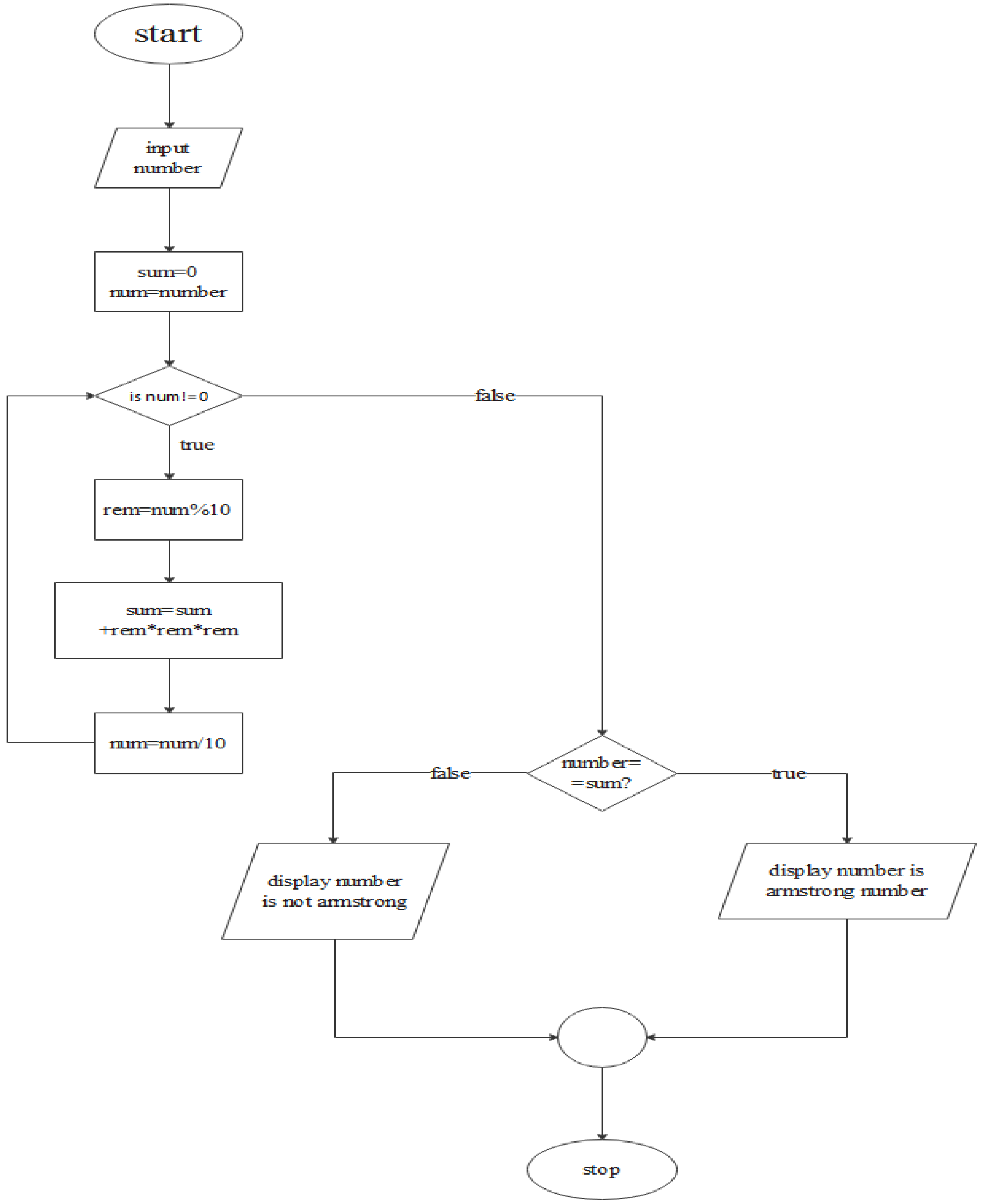
4. Write a program to check whether the given number is Armstrong or not.

An Armstrong number is an integer such that the sum of the cubes of its digits is equal to the number itself. example 153 is Armstrong number.

- Algorithm:

1. Start
2. Input number
3. Assign num=number and sum =0
4. Check if num is not equals to 0
5. Isolate last digit of number using modulus operator , remainder= num%10
6. Calculate cube of remainder and add them, sum=sum + remainder ^3
7. Remove last digit from the number, num = num/10
8. Repeat step 4 to step 8 until the condition is true else go to step 9
9. Check if (number == sum)
10. If condition in step 9 is true, display Armstrong number
11. If condition in step 9 is false , display not Armstrong number
12. stop

Armstrong number: flowchart



Class work:

5. Write a program to calculate factorial of a given number using while loop.

3. for loop:

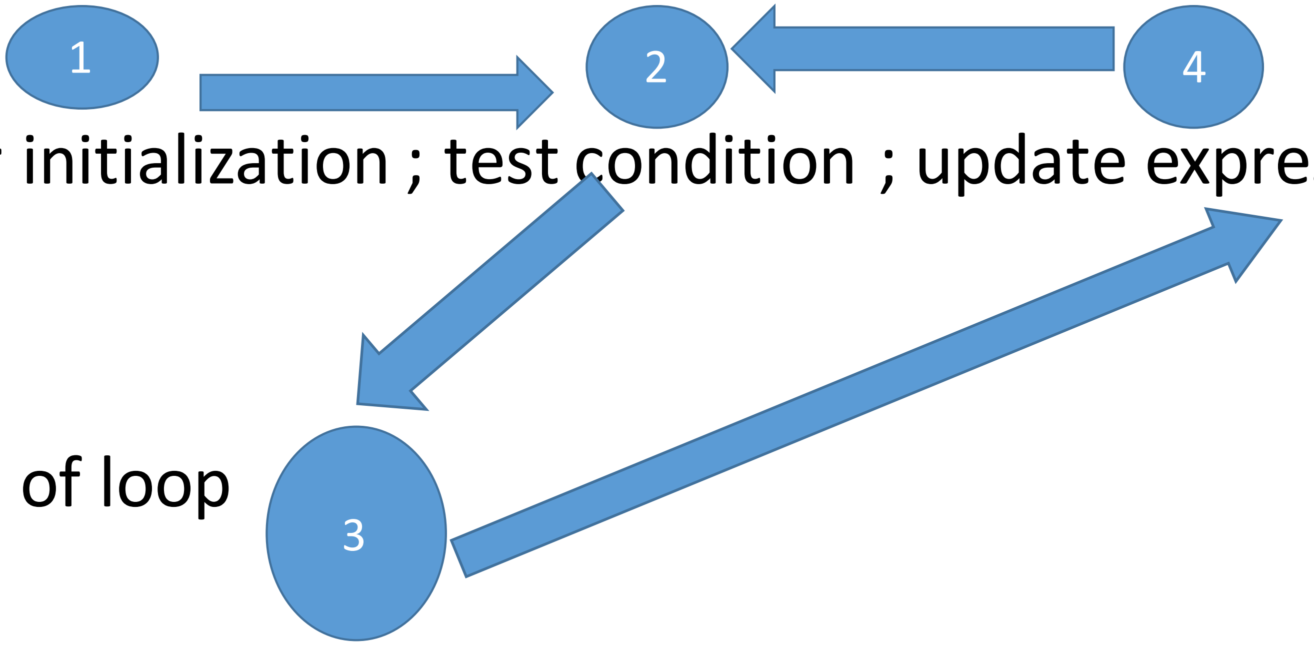
Syntax:

```
    for( counter initialization ; test condition ; update expression)
    {
        body of loop
    }
```

3. for loop:

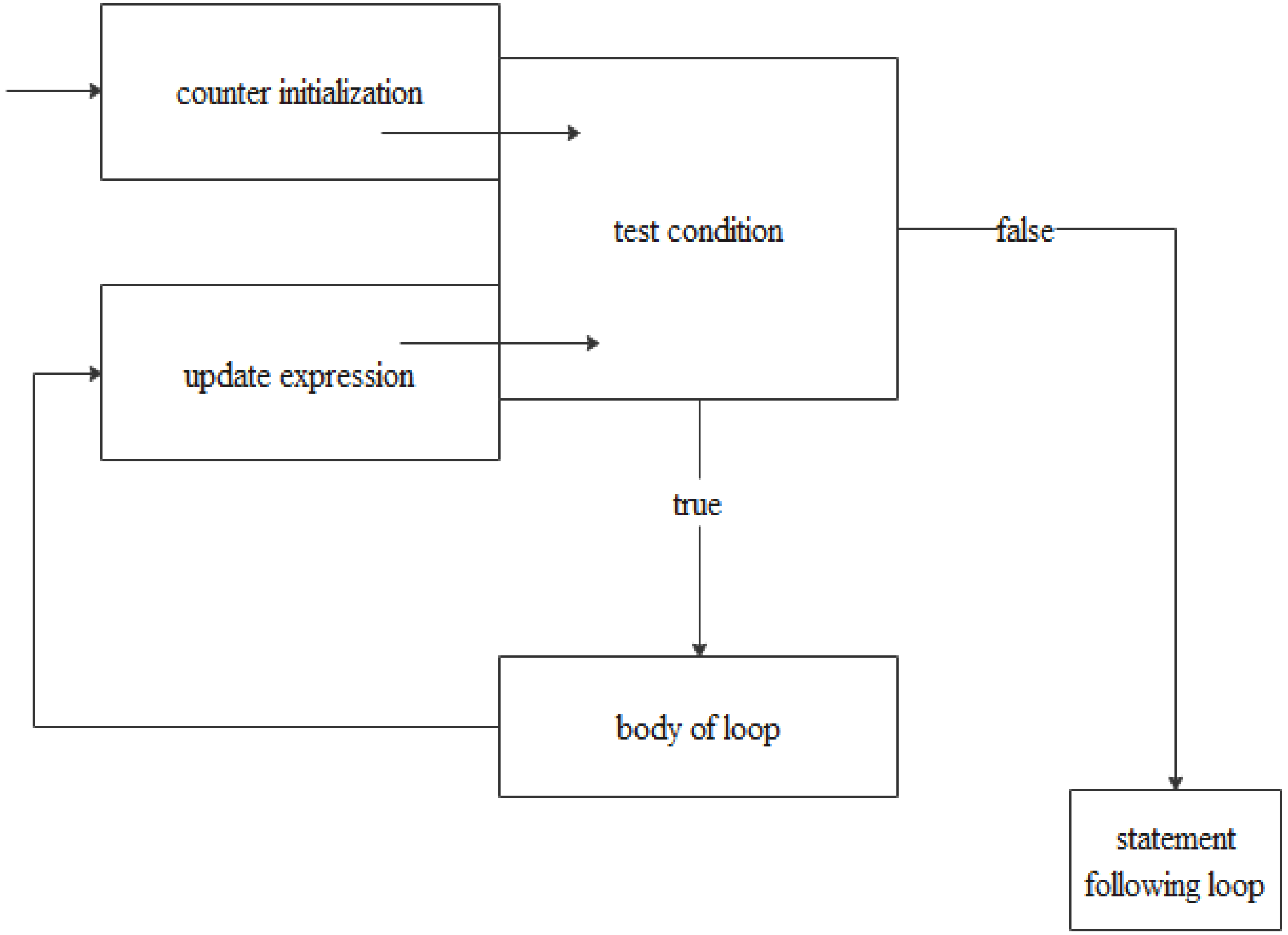
Syntax:

```
        for(counter initialization ; test condition ; update expression)
        {

                body of loop

        }
```

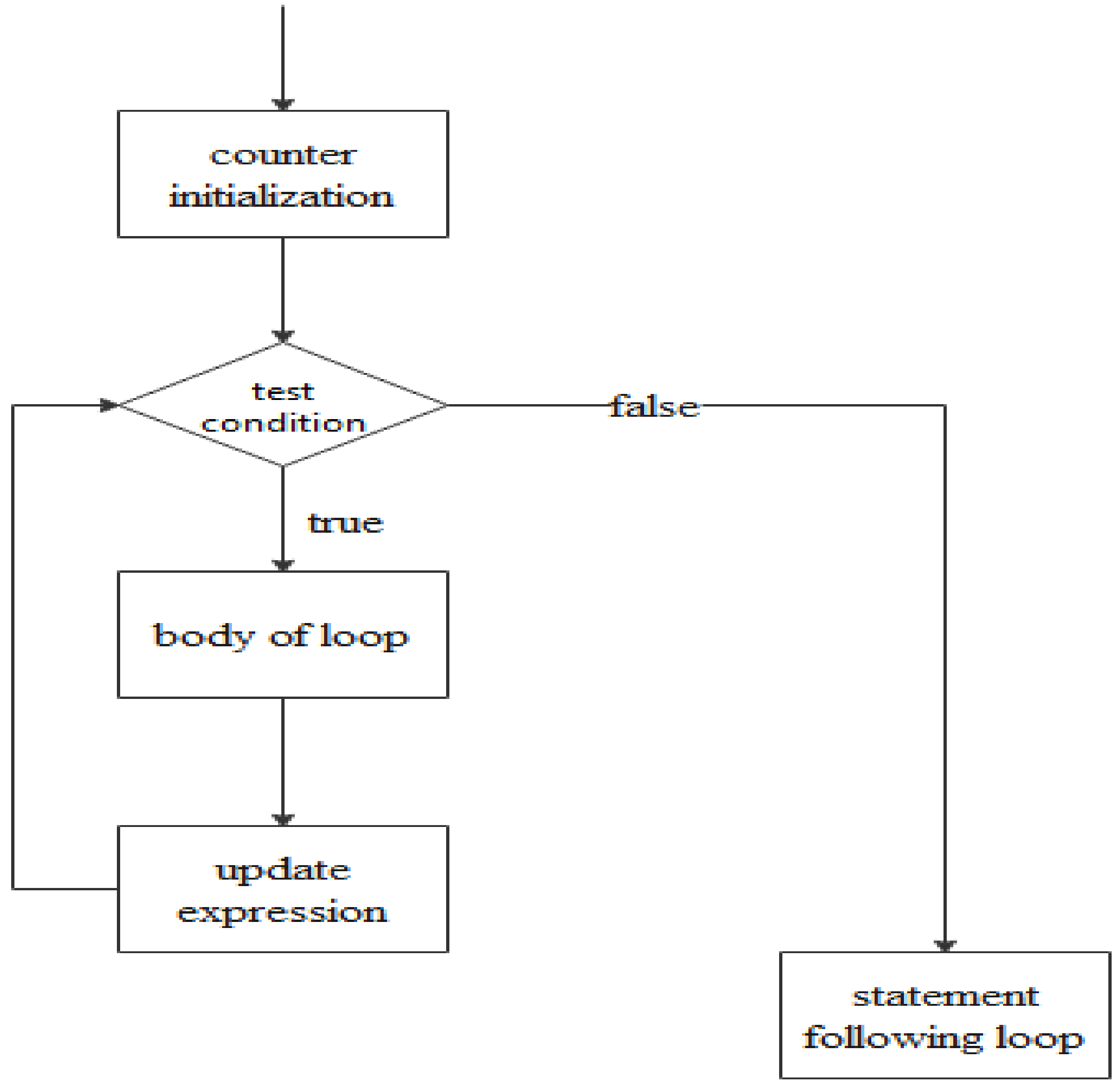


for loop:

Flowchart:



for loop:


counter initialization
test condition
false
true
body of loop
update expression
statement following loop

1. Write a program to calculate factorial of a given number using for loop.

```
#include<stdio.h>
int main()
{
        int number,factorial,i;
        printf("Enter the number :");
        scanf("%d",&number);
        factorial = 1;
        for(i=1;i<=number;i++)
        {
                factorial=factorial*i;
        }
        printf("\n%d!=%d",number,factorial);
        return 0;
}
```

Class work:

1. Write a program to calculate the sum of n natural numbers. i.e sum of natural numbers from 1 to n.
2. Write a program to find sum of first n odd integers (i.e. 1+3+5+… +2n-1)
3. Write a program to find sum of 5+10+15+20+…up to nth term.
4. Write a program raise number X to power n i.e. X^n

5. Write a program to print the Fibonacci sequence up to nth terms.

6. Write a program to print nth term of Fibonacci sequence.

7. Write a program to print the Fibonacci sequence up to nth terms where initial terms are input by the user.

8. Write a program to print first 10 terms of the following series using for loop

1 5 9 13 …

9. Write a program to find sum of the series

$1/1 + 1/2 + 1/3 + \ldots + 1/n$

Write a program to print the Fibonacci sequence up to nth terms.

```
#include<stdio.h>
int main()
{
		int t1,t2,next_term,i,n;
		t1=0;
		t2=1;
		printf("Enter the number of terms of fibonacci series to be generated :");
		scanf("%d",&n);
		for(i=1;i<=n;i++)
		{
				if(i==1)
					{
						printf("%d\t",t1);
						next_term=t1;
					}
				else if(i==2)
					{
						printf("%d\t",t2);
						next_term=t2;
					}
```

```
                        else
                              {
                                    next_term=t1+t2;
                                    t1=t2;
                                    t2=next_term;
                                    printf("%d\t",next_term);
                              }
            }
            printf("\n\n%dth term of fibonacci series is %d",n,next_term);
return 0;
}
```

Nested loop:

- A loop statement inside another loop statement is called nested loop

Syntax:

```
for (counter initialization ; test condition ; update expression)
{
        statement/s ;
        for( counter initialization ; test condition ; update expression)
        {
                body of inner loop

        }
        statement/s;
}
```

Outer loop

inner loop

Write a program to generate Armstrong numbers from 100 to 1000.

```
#include <stdio.h>
int main()
{
        int number,  num,  sum,  rem;
        printf("Armstrong  numbers  from 100 to 1000 are :\n\n");
        for(number=100  ;number<=1000  ;number++)
        {

                num=number;
                sum=0;
                while(num!=0)
                {
                        rem=num%10;
                        sum=sum+(rem*rem*rem);
                        num=num/10;
                }
                if(sum==number)
                {
                        printf("%d  \t",number);
                }
        }
        return  0;
}
```

Class Work:

1. Write a program to add first seven terms of the following series using for loop.

   $$\text{sum} = 1/1! + 2/2! + 3/3! + \ldots$$

2. Write a program to find sum of first n terms

   $$\text{sum} = x - x^3/3! + x^5/5! - x^7/7! + \ldots$$

3. Write a program to find sum of first n terms

   $$\text{sum} = 1 - x^2/2! + x^4/4! - x^6/6! + \ldots$$

Class work:

4. The sine of x can be calculated approximately by using the first n terms of the infinite series. (x is expressed in radian, note ∏ radian = 180^o)

$$\sin(x) = x - x^3/3! + x^5/5! - x^7/7! + \dots$$

5. Write a program to calculate the sum of cosine series

$$\cos(x) = 1 - x^2/2! + x^4/4! - x^6/6! + \dots$$

Class Work:

6. Write a program to check whether given number is prime or not.
7. Write a program to display all prime numbers from n1 to n2 where n1<n2.
8. Write a program to generate multiplication table of numbers from 1 to n, where n is given by user

Q. Write a program to find sum of first n terms

$$sum = x – x^3/3! + x^5/5! – x^7/7! + \ldots$$

```
#include<stdio.h>
#include<math.h>

int main()
{
        int i, n , x ,odd_term,factorial,j,numerator,sign;
        float term,sum=0;
        printf("Enter the value of x : ");
        scanf("%d",&x);
        printf("Enter the number of terms :");
        scanf("%d",&n);
        //convert x in degree to x in radian in case of sin series
        //x in radian = (x in degree * 3.14159)/180
for(i=1;i<=n;i++)
          {
                    odd_term=2*i-1;
                    factorial =1;
                    for(j=1;j<=odd_term;j++)
                    {
                              factorial=factorial*j;
                    }
                    numerator=pow(x,odd_term);
                    sign=pow(-1,i+1);
                    term=1.0*sign*numerator/factorial;
                    printf("\n");
                    printf("%f \t " , term);
                    sum = sum + term;
          }
          printf("\n\nSum=%f",sum);

return 0;
}
```