Unit II

Decision Making and Branching: simple if, if else, if else ladder, nested if, switch, nested switch-syntax, flowchart, and example programs.

Decision Making and Looping: while, do-while, for statements, nested loops-syntax, flowchart, and example programs.

Unconditional Branching Statements: break, continue and go to-syntax, flowchart, and example programs

Class 9: Iterative statements (Looping statements)

- In looping, a sequence of statements are executed until some conditions for termination of the loop are satisfied.
- A program loop consists of **two** parts.

➢ **Body of the loop**

➢ **Control statement**

- The control statement tests certain conditions and then directs the repeated execution of the statements contained in the body of the loop.

- Depending on the position of the control statement in the loop, a control structure may be classified as the **entry-controlled loop (pre test)** or as **exit-controlled loop (post-test).**
- The C-language provides three constructs for performing loop operations. They are

1. **while** loop (pre test)
2. **do-while** loop (post-test)
3. **for** loop (pre test)

While Loop

- In while loop, the condition is tested before any of the statements in the **statement block** is executed.
- If the condition is true, only then the statements will be executed otherwise the control will jump to the immediate statement outside the while loop block.

Syntax:

```
while (condition)
{
        statement_block;
}
statement x;
```

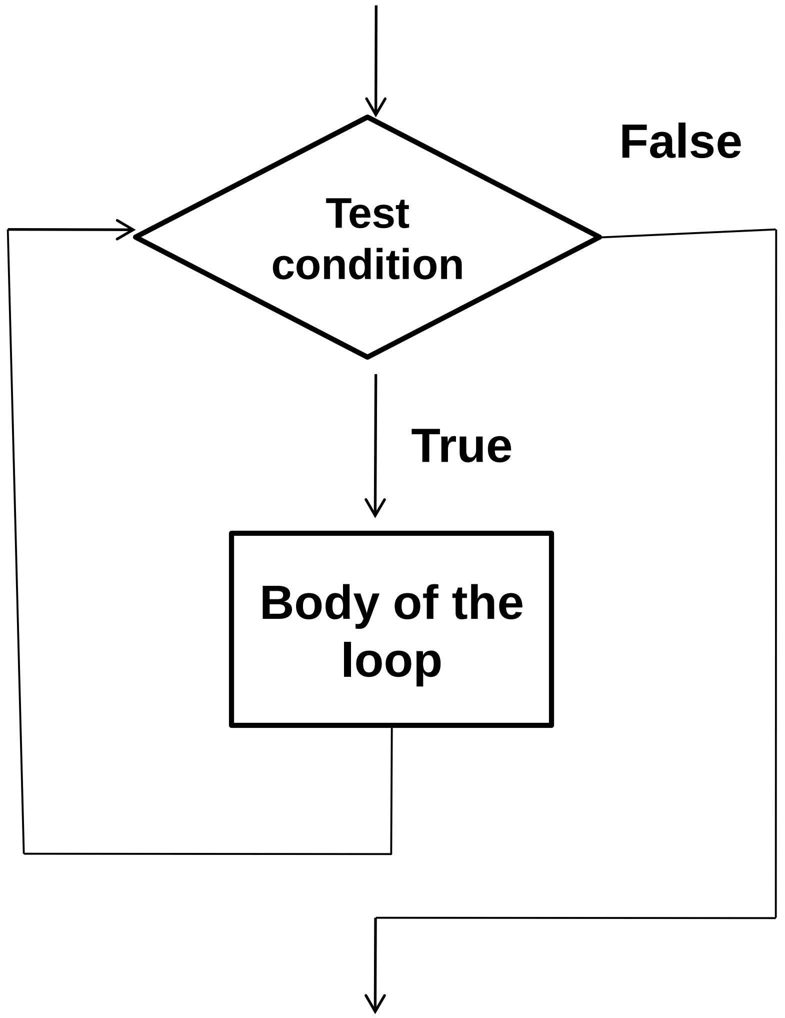

Fig: Entry controlled loop (while)

- **Program to print 1 to 10 numbers**

```
 main()
{
        int i=1;

            while(i<=10)
            {
                printf(“%d”, i);
                 i++;
            }
  }
```

To print 1 to n numbers

```
 main()
{
  int i, n;
 printf(“Enter n”);
 scanf(“%d”, &n);
 i=1;
while(i<=n)
{
   printf(“%d”, i);
   i++;
}
```

```
Output: Enter n 5
     1 2 3 4 5
```

Loop programs

1. Write a C program to print 1 to n numbers
2. Write a C program to find sum and average of n numbers.
3. Write a C program to find factorial of given number
4. Write a C program to check whether given number is prime or not
5. Write a C program to find reverse of the given number
6. Write a C program to find sum of individual digits of number
7. Write a C program to check given number is palindrome or not
8. Write a C program to check given number is Armstrong or not

➢ sum of n numbers

```
sum=0,i=1;
while(i<=n)
   {
        sum=sum+i;
        i++;
      }
print sum
```

➢ Factorial of a number

```
  fact=1;
while(i<=n)
   {
      fact=fact*i;
      i++;
    }
print fact
```

➢ Reverse of given number

```
while(n>0)
     {
       x=n%10;
        printf("%d", x);
       n=n/10;
      }
```

➢ Reverse of given number

```
    rev=0;
while(n>0)
    {
      x=n%10;
      rev=rev*10+x;
      n=n/10;
     }
```

5. Sum of individual numbers in a numbers

```
sum=0;
while(n>0)
  {
      x=n%10;
      sum=sum + x;
      n=n/10;
   }
```

➢ ARMSTRONG number

```
arm=0;
m=n;
while(n>0)
  {
        x=n%10;
        arm=arm+(x*x*x);
        n=n/10;
   }
if(m= =n)
        print it is Armstrong
else
          print it is not an  Armstrong
```

➢ check given number is prime or not

```
count=0,i=1;
while(i<=n)
  {
        if(n%i==0)
        count++;
        i++;
  }
if (count==2)
        print it is prime;
  else
             it is not prime;
```

do-while Loop

- The do-while loop is similar to the while loop. The only difference is that in a do-while loop, the test condition is tested at the end of the loop and terminates with a **semicolon**.
- The body of the loop gets executed at least one time (even if the condition is false). The do while loop continues to execute while a condition is true.
- Do-while loops are widely used to print a list of options for a **menu driven** program.

Syntax:

```
do
{

statement_block;

} while (condition);

statement y;
```

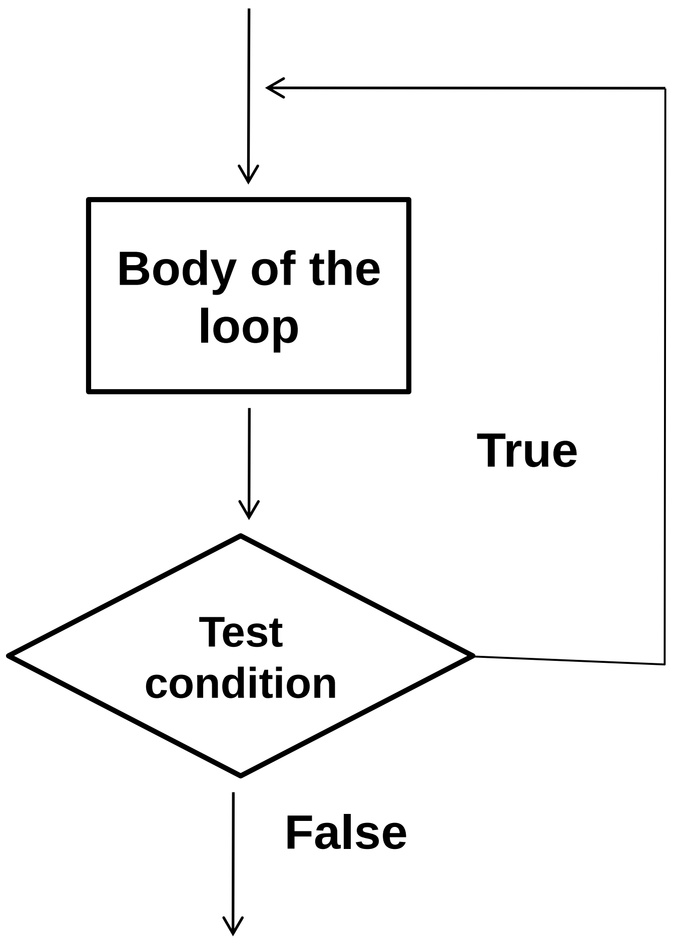

Fig: Exit controlled loop (do-while)

➢ Program to print 1 to 10 numbers using do-while loop

```
 main()
{
        int i=1;
             do
            {
               printf(“%d”, i);
                i++;
           } while(i<=10);
  }
```

- **do-while loop**

```
i=11;
do
{         printf("%d", i);
          i++;
}while(i<=10);
```

o/p: 11

while loop

```
i=11;
 while(i<=10)
{
     printf("%d", i);

          i++;
}
```

o/p: no output

for Loop

- Like the while and do-while loop, the for loop is used to repeat a task until a particular condition is true.
- The syntax of a for loop

```
for (initialization; condition; updation)
{
    body of the loop
}
```

- The execution of the for statement is as follows:

1. Initialization of the control variable is done first and is initialized **only once** using assignment statement such as i=1.

2. The condition such as i>10 determines when the loop will exit. If the condition is true, the body of the loop is executed; otherwise the loop is terminated

3. When the body of the loop is executed, the control is transferred back to the for statement after evaluating the last statement in the loop. Now, the control variable is incremented /decremented such as i++,i-=2 and the new value is again tested to see whether it satisfies the loop condition. If the condition is satisfied, the body of the loop is again executed. This process continues till the value of the control variable fails to satisfy the test-condition.

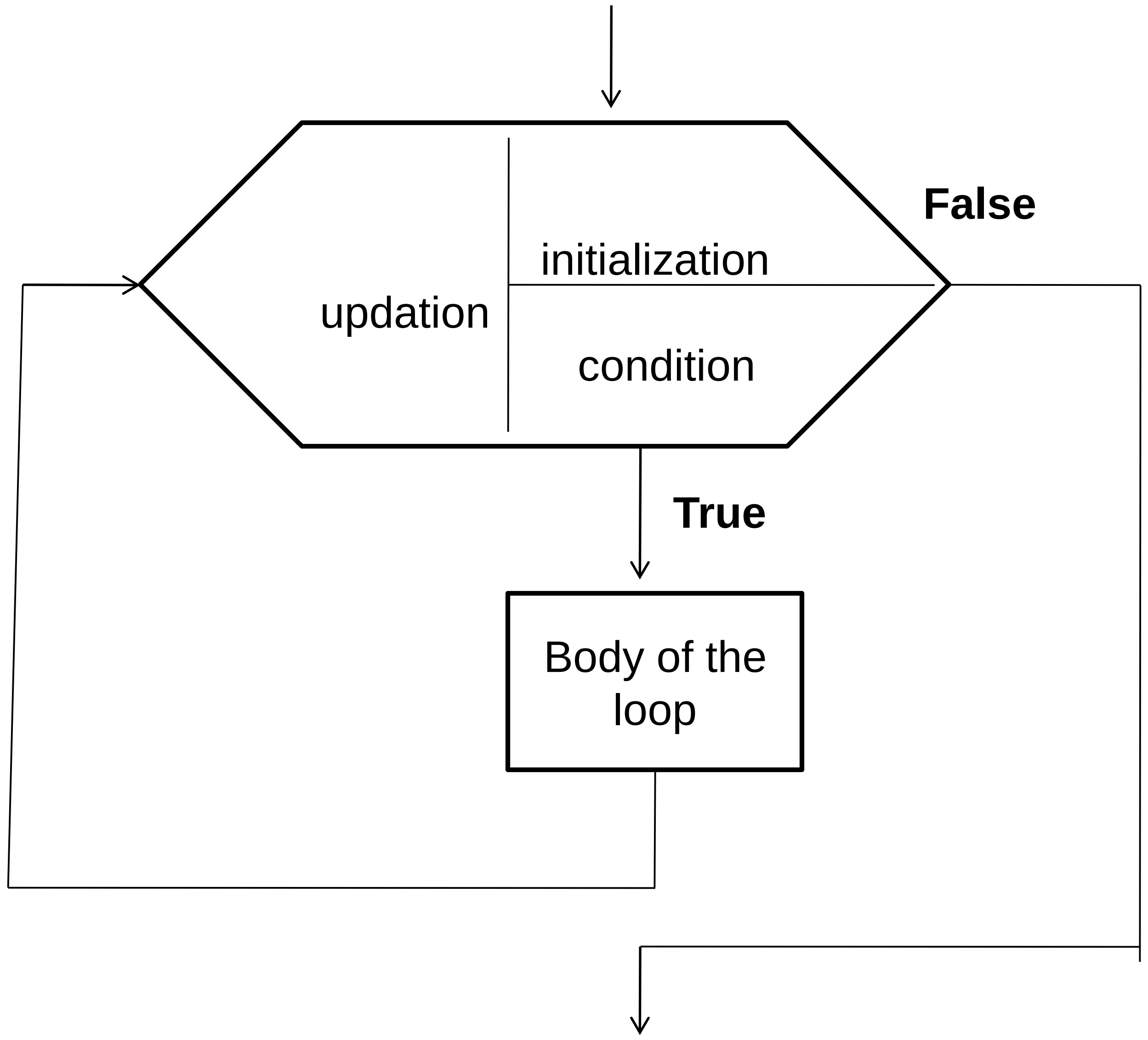
False
initialization
updation
condition
True
Body of the loop

Fig: for loop

- **Program to print 1 to 10 numbers using for loop**

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

1. Write a C program to print fibonacci series (0112358..)
2. Write a C program to find sum of even numbers and odd numbers in a given range
3. Write a C program to find sum of the $1+ 2^2+3^3+\ldots\ldots\ldots n^n$ series
4. Write a C program to find sum of the $1+1/2+1/3+\ldots\ldots\ldots 1/n$ series
5. Write a C program to find sum of the $1+1/1!+1/2!+1/3!+\ldots\ldots\ldots 1/n!$ series

6. Write a C program to find sum of $x + x^2/2 + x^3/3 + \ldots\ldots\ldots$ series
7. Write a C program to find sum of $x - x^3/3! + x^5/5! + \ldots\ldots\ldots$ series
8. Write a C program to print Pascal triangle
9. Write a C program to print prime numbers between 1 and n
10. Write a C program to print Armstrong numbers between 1 and n
11. Write a program to find gcd of two numbers
12. Write a program to find lcm of two numbers
13. **Write a program to find factorial of a number**
14. **Write a program to check whether given number is prime or not**

➢ Fibonacci series

```
int i, f1 = 0, f2 = 1, t = 0, n;
read n;
print f1,f2;
for(i=3; i<=n; i++)
{
        t=f1+f2;
        f1=f2;
        f2=t;
}
print t;
```

➢ Sum of odd and even numbers in a range

```
i=1,esum=0,osum=0;
while(i < = n)
  {
        if(i%2= =0)
              esum+=i;
        else
              osum+=i;
        i++;
  }
print esum, osum
```

- Sum of series like 1+(1/2)+(1/3)+(1/4)+...

```
n= last number in series
for(i=1; i<=n ; i++)
{
sum=sum+(1.0/i);
}
print   sum
```

- Sum of series like x +x^2/2+x^3/3+……….

```
int i, n, x;
float product=1.0,sum=0;
for(i=1;i<=n;i++)
{
product=product*x;
sum=sum+(product/i);
}
print sum
```

Sum of series like $x - x^3/3!+x^5/5!+\ldots\ldots\ldots$

```
 Read n, x;
 float sum=0,k=1;
 k=x, sum=x;
 for(i=3;i<=n; i+=2)
{
   k=(-k*x*x)/(i*(i-1));
 sum=sum + k;
}
Print sum;
```

➢ Prime numbers in a range…

```
printf("\n The List of prime numbers between %d-%d",j,k);

for(;j<=k;j++,count=0)
{
        for(i=1;i<=j;i++)
        {
                if(j%i==0)
                        count++;
        }
        if(count==2)
        {
                printf("\t%d",j);
        }
}
```

➢ Armstrong numbers in a range

```
 read range j,k
for(  ; j<=k ;  j++,arm=0)
{
        for(i = j; i>0 ; i = i/10)
        {
                x=i%10;
                arm=arm+x*x*x;
        }
        if (arm = = j)
                print arm
}
```

Gcd of two numbers

```
while(y!=0)
{
   r=x%y;
   y=r;
   x=y
}
printf(“gcd=%d”,x);
```

lcm of two numbers

$$x * y = gcd * lcm$$

so $lcm = x * y / gcd$