Lecture - 4

Control and Decision Making Statements

if…else statements

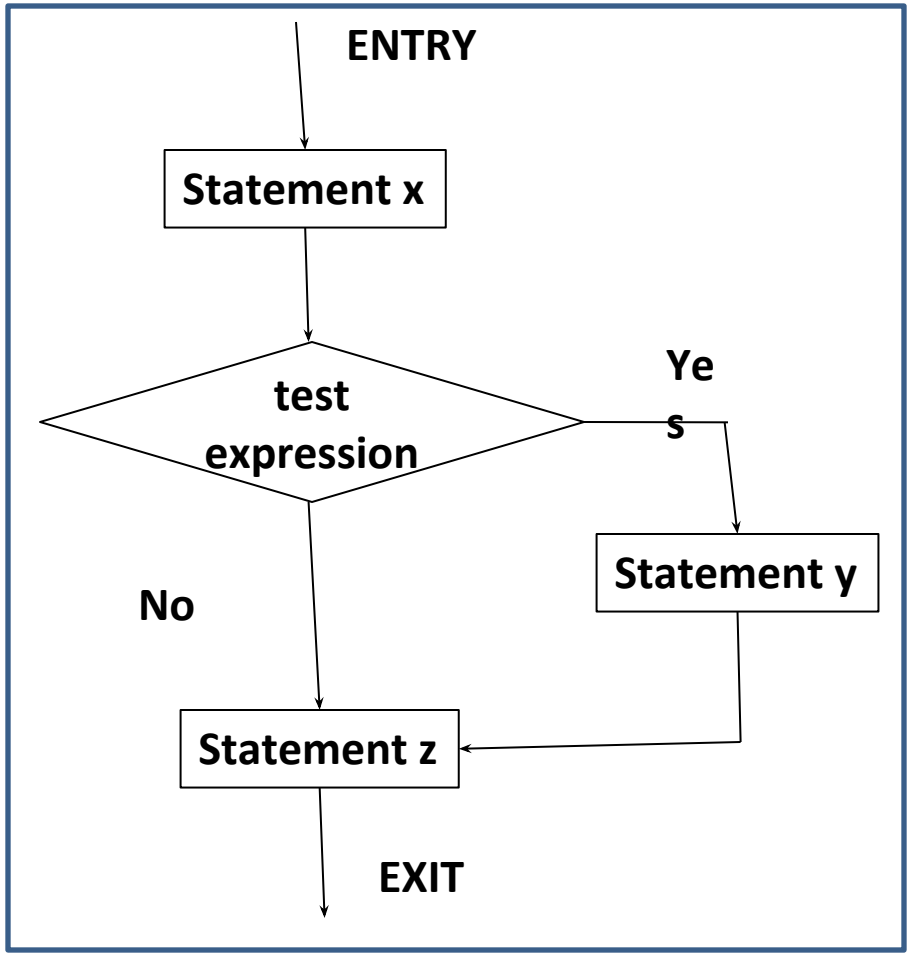
❑ Simple *if* statement

- Basic Structure

```
statement x;
if (test expression) {
      statement y;
}
statement z;
```

A set of **statement y** will be executed <u>only if</u> the **condition(or test expression)** is **true**

- Flow Chart



Simple *if* statement

- Example:

```
1  #include<stdio.h>
2  void main(){
3      int n;
4      scanf("%d",&n);
5      if (n%5==0) {
6          printf("%d is divisible by 5",n);
7      }
8      printf("\nHave a good day!");
9  }
```

- **Output ???**

Simple *if* statement

- Example:

```c
#include<stdio.h>
void main(){
    int n;
    scanf("%d",&n);
    if (n%5==0) {
        printf("%d is divisible by 5",n);
    }
    printf("\nHave a good day!");
}
```

- **Output**

```
25
25 is divisible by 5
Have a good day!

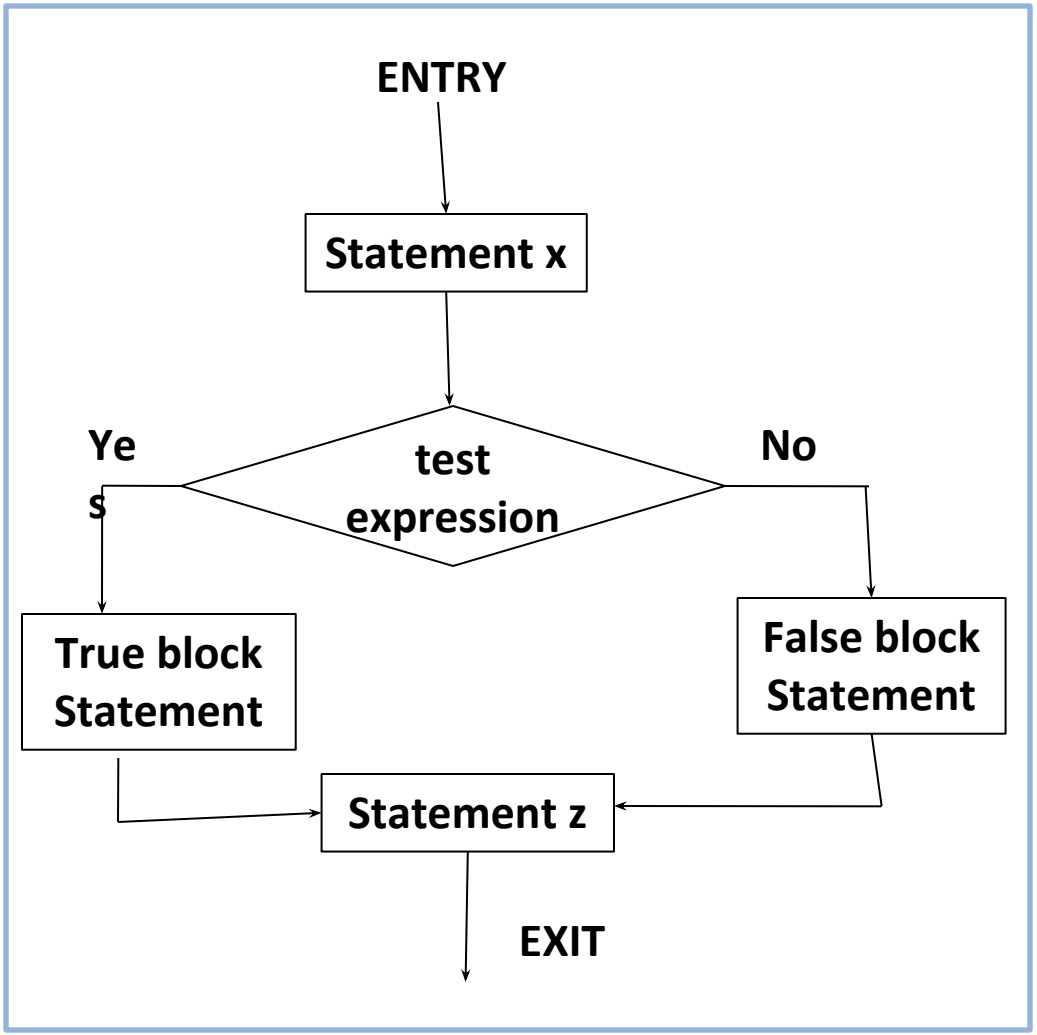
Process returned 0 (0x0)
Press any key to continue.
```

❑ *if…else* statement

- Basic structure

```
statement x;
if (test expression) {
        True block statement ;
}
else{
        False block statement;
}
statement z;
```

- Flow diagram ?



❑ Nested *if…else*

- Basic structure

```
statement x;
if (test expression) {
        if (test expression) {
                Statement 1 ;
        }
        else{
                Statement 2;
        }
}
else{
        Statement 3;
}
statement z;
```

- **Draw the Flow Chart**

- **Problem-1:** Write a C program to calculate the absolute value of an integer.
- **Problem-2:** Write a C program to determine whether a number is odd or even.
- **Problem-3:** Write a C program to determine the largest number from three given integers.

❑ *if…else if* ladder

Problem-4:

- Write a C program for grading the students in an academic institution. The grading is done according to the following rules:

Average Marks	Grade
80-100	A+
70-79	A
60-69	B
50-59	C
40-49	D
0-39	F

- Draw the corresponding flowchart.

if…else if ladder

Problem-5:

- Write a C program to evaluate a simple arithmetic expression as follows.

Sample I/O:

```
Enter a simple arithmetic expression: 4 – 5
Result is -1
```

```
Enter a simple arithmetic expression: 10 / 0
Warning – Divide by zero!!
```

Problem-6:

- Write a C program to determine the largest among three given integers. (hint: using *if… else if* ladder and nested *if…else*).

switch…case *statement*

❑ *switch…case statement*

- **General form/ Basic Structure/Syntax:**

```
switch ( expression )
{
  case value1:{
        block 1;
        break;
        }
  case value2:{
        block 2;
        break;
        }
        …….
        …….
  default:{
        default block;
        break;
        }
}
```

- **Problem-7:** Solve the Problem-4 using switch…case statement.
- **Problem-8:** Solve the Problem-5 using switch…case statement.

Iterative statements: Loops

❑ Loops

- The versatility of the computer lies in its ability to perform **a set of instructions repeatedly**. This involves repeating some portion of the program **either a specified number of times or until a particular condition is being satisfied**. This repetitive operation is done through a loop control instruction.

Iterative statements: LOOP

- Three types of looping statements:
  1. ***for*** loop
  2. ***while*** loop
  3. ***do…while*** loop

❑ *for* Loop: Syntax

```
for ( initialization;  condition;  increment/decrement){
        statements ;
}
```

How ***for*** Loop works?

1 2 4

for (initialization; condition; increment/decrement){

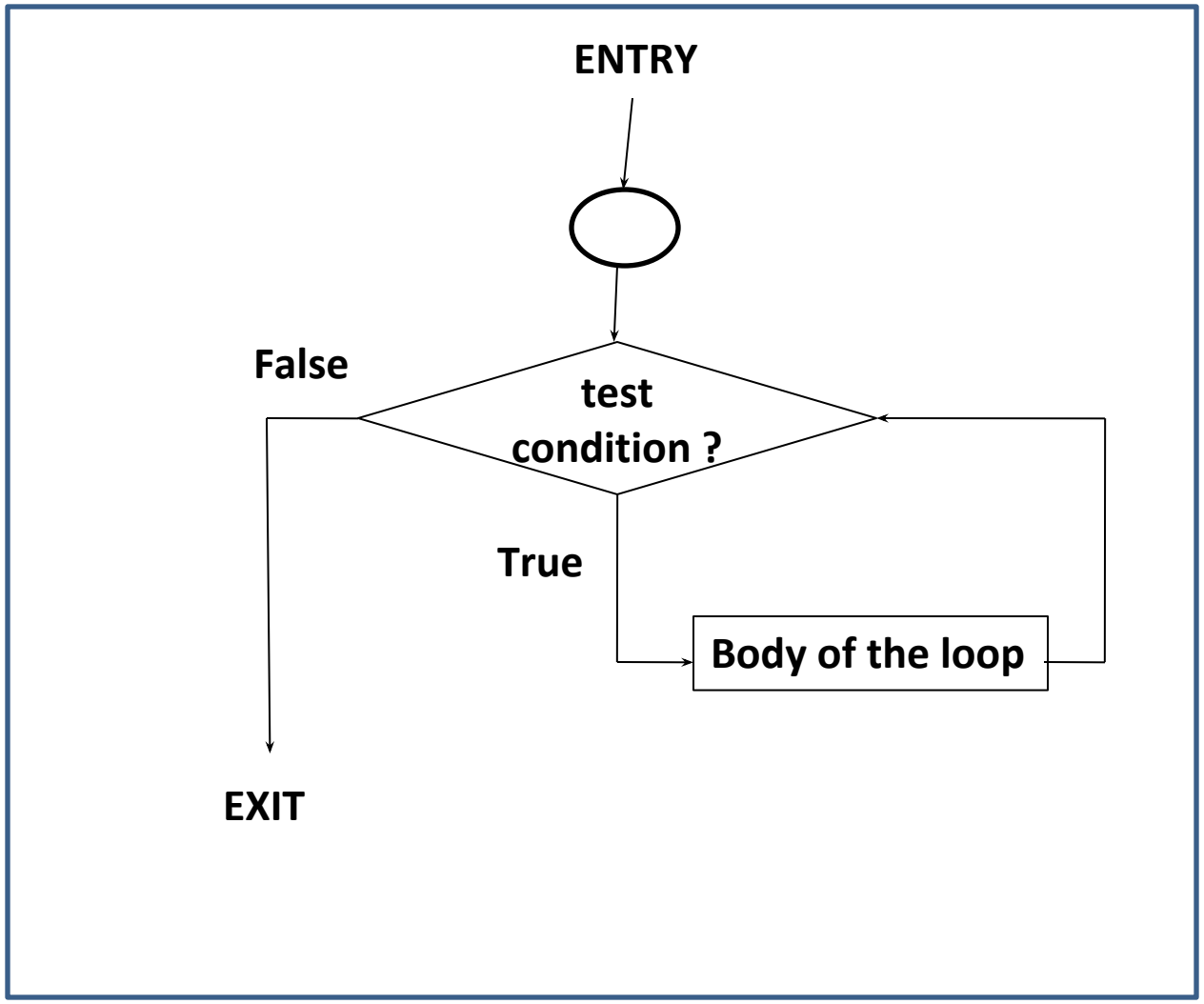
statements ;

3

}

Start 1 → 2 (if true) → 3
→ 4 → 2 (if true) → 3
→ 4 → 2 (if true) → 3
→ .....
→ 4 → 2 (if false) → **exit**

for Loop: Flow diagram



for Loop: Example

```
1   #include<stdio.h>
2   int main(){
3       int a;
4       for(a=1; a<=5; a++)
5       {
6           printf("value of a is %d\n",a);
7       }
8       return 0;
9   }
10
```

Output ? ? ?

for Loop: Example

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

Output

```
value of a is 1
value of a is 2
value of a is 3
value of a is 4
value of a is 5
```

Problem solving using ***for*** loop

- Write a C program –
  1. To find the sum of the series 1 + 2 + 3 + … + n.
  2. To find the factorial of an integer.
  3. To find the number of and sum of all integers greater than 100 and less than 200 that are divisible by 7.
  4. To find the sum of the following harmonic series for a given value *n.*

     [Harmonic series: 1 + 1/2 + 1/3 + … + 1/n]

❑ *continue* statement

- Used when we need to skip a part of loop under certain condition
- *"skip the following statement and continue with the next iteration"*

- **Solve the Problem:** Write a C program to print all the numbers from *m* to *n* **<u>except</u>** the numbers divisible by 5 (*m, n* are given values)

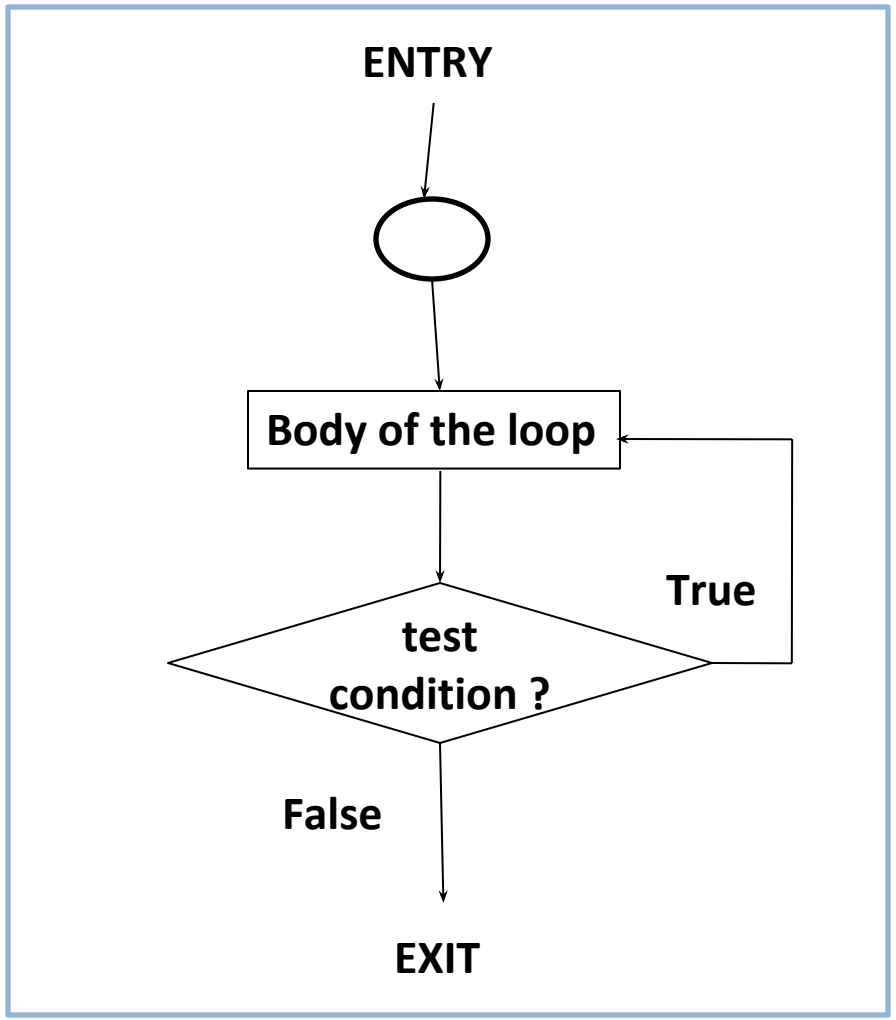
❑ *while* loop

- Syntax

```
While(condition){
        Statements;
 }
```

Start → Condition (if True) → Statements
→ Condition (if True) → Statements
→ ……
→ Condition (if False) → **Loop End**

while Loop: Flow Diagram



while Loop: Example

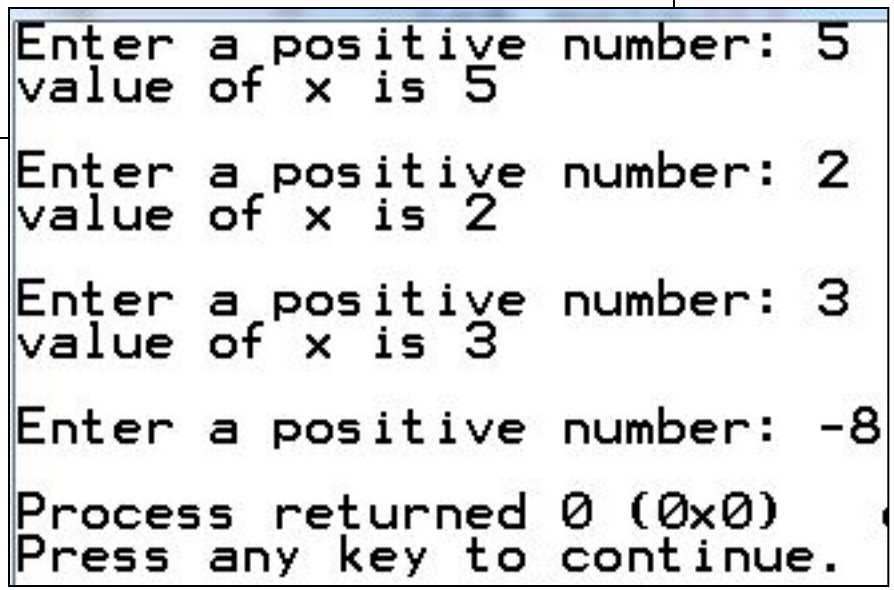
```
1   #include<stdio.h>
2   int main(){
3       int x;
4       printf("Enter a positive number: ");
5       scanf("%d",&x);
6       while(x > 0)
7       {
8           printf("value of x is %d\n",x);
9           printf("\nEnter a positive number: ");
10          scanf("%d",&x);
11      }
12      return 0;
13  }
```

Output ? ? ?

while Loop: Example

```c
#include<stdio.h>
int main(){
    int x;
    printf("Enter a positive number: ");
    scanf("%d",&x);
    while(x > 0)
    {
        printf("value of x is %d\n",x);
        printf("\nEnter a positive number: ");
        scanf("%d",&x);
    }
    return 0;
}
```

Output



Problem solving using ***while*** loop

- Write a C program –
  1. To display an integer in reverse order.
  2. To find the sum of the digits of a given integer.
  3. To convert an integer into the reverse integer.
  4. To find the first *n* Fibonacci numbers.
  5. To determine whether a number is prime or not.
  6. To convert an integer to its binary equivalent.

❑ *break* statement

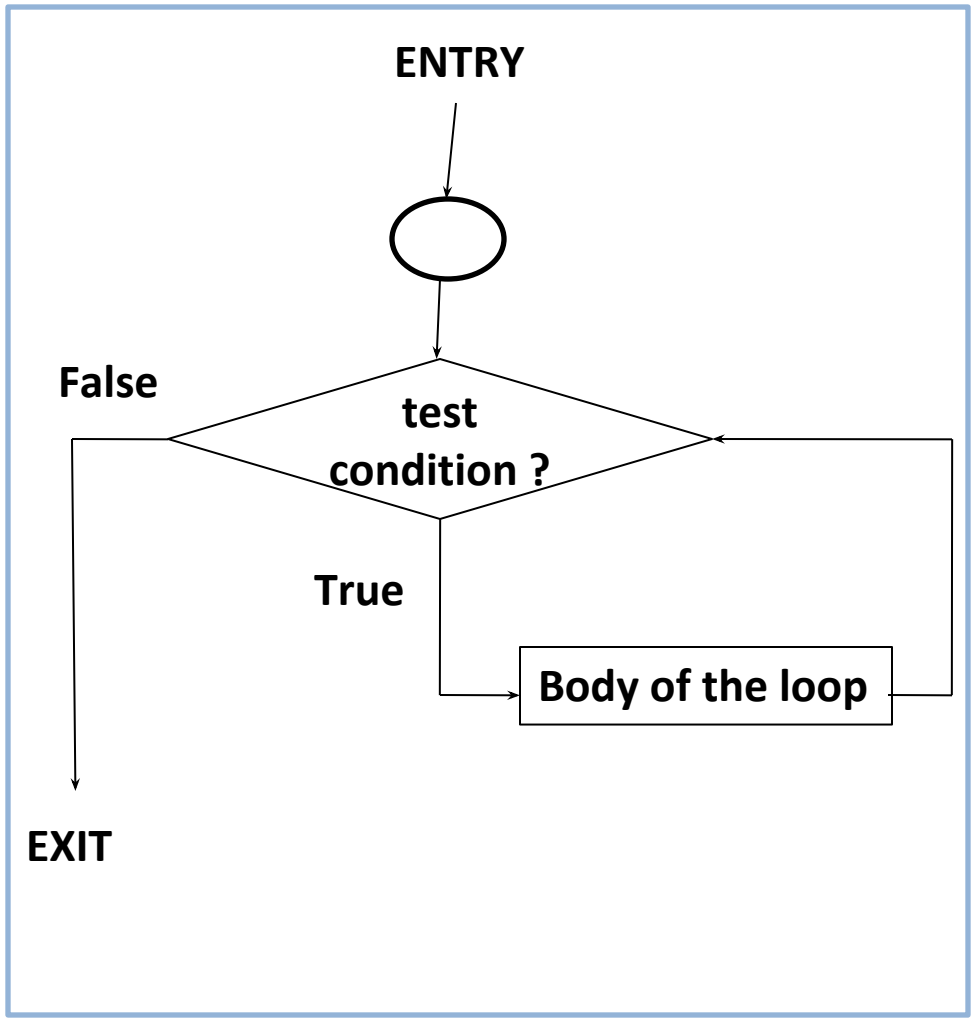
- When a *break* statement is encountered inside a loop, the loop is **immediately terminated**
- It can be used to terminate a *case* in the *switch* statement

- **Solve the Problem:** Write a C program to print all the numbers from *m* to *n* **<u>until</u>** the number is divisible by 5 (*m, n* are given values)

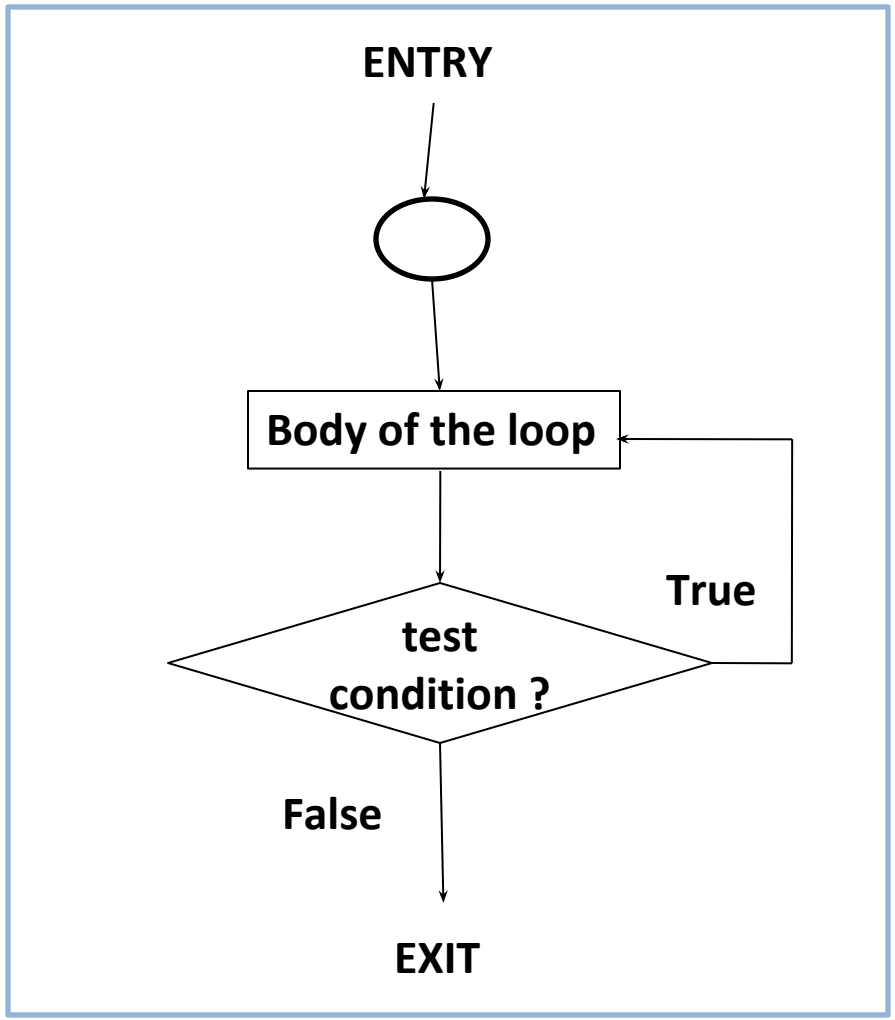
❑ Entry controlled loop and Exit controlled loop

- **Entry control loop**
  - Test condition is evaluated before the start of the loop execution
  - Pre-test loop
- **Exit control loop**
  - Test condition is evaluated at the end of the loop execution
  - Loop will be unconditionally executed for the first time only
  - Post-test loop

Flow chart for Entry controlled loop and Exit controlled loop



Entry controlled loop



Exit controlled loop

❑ Counter controlled loop

- Use when we know exactly how many times the loop will be executed
- Use a control variable known as counter
- Counter must be initialized, tested and updated properly
- The number of times the loop will execute may be a constant or variable
- Also known as ‘definite repetition loop’
- Example : ???

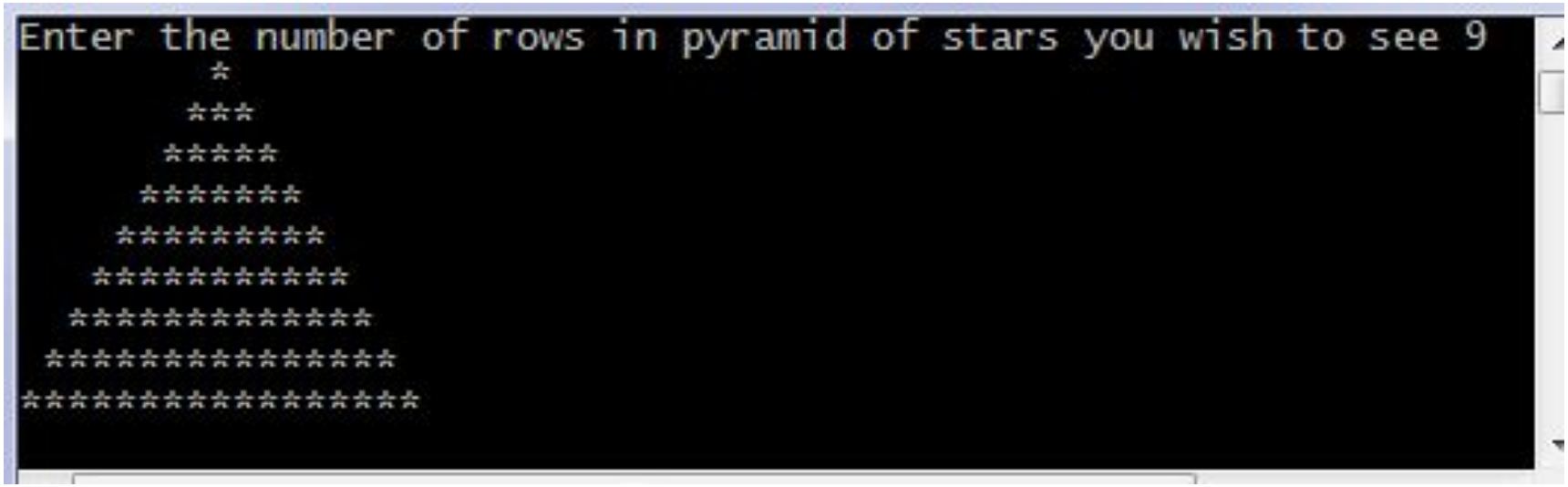
❑ Sentinel controlled loop

- Use when we don’t know exactly how many times the loop will be executed
- The control variable known as sentinel variable
- Also known as ‘indefinite repetition loop’
- Example: ???

Infinite Loop

❑ Pattern programs in C

- how to create various patterns of numbers and stars
- require nested loops (a loop inside another loop).
- a way of arranging these (numerals, stars, or characters) in some logical manner, or they may form a sequence
- Some of these are triangles, pyramid, diamond or symmetrical shape/pattern


Enter the number of rows in pyramid of stars you wish to see 9
*
***
*****
****
****
****
****
****
****

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







```
12345
1234
123
12
1
```

```
    1
   123
  12345
 1234567
123456789
```

```
A
B C
D E F
G H I J
K L M N O
```

```
 A
 A B
 A B C
 A B C D
 A B C D E
```

```
      A
     ABA
    ABCBA
   ABCDCBA
  ABCDEDCBA
```