

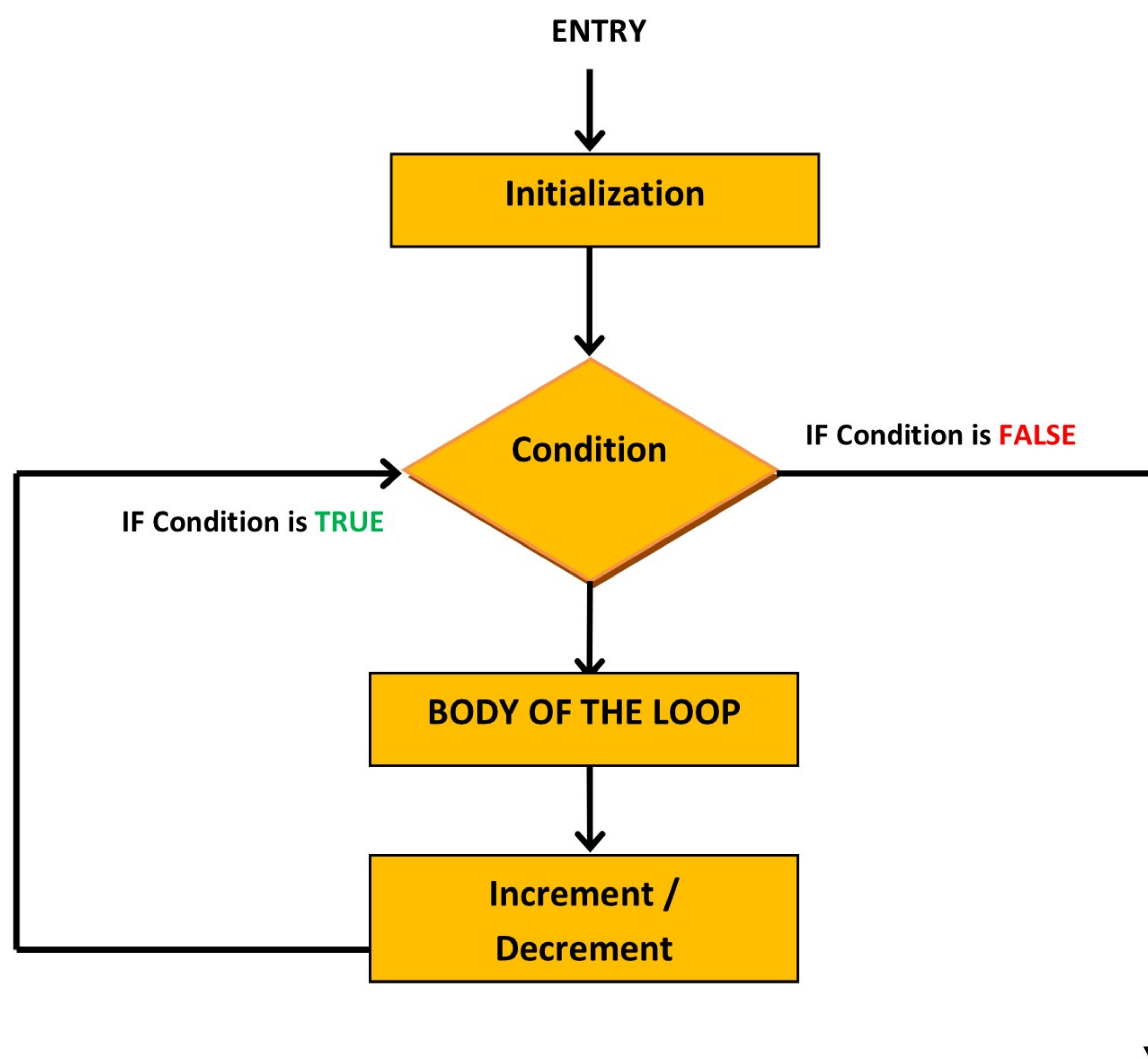
### 1. Explain loops available in C with example

- Loops are used to repeat execution of a block of code.
- During looping, a set of statements are executed until some condition for termination is encountered.
- Generally, looping process would include the following four steps
  - 1) Initialization of a counter
  - 2) Test for a termination condition
  - 3) Loop body statements
  - 4) Increment the counter
- C supports three types of looping

#### 1. while loop

- The simplest of all looping structure is while statement.
- The general format of the while statement is:

```
while (test condition)
{
    body of the loop ;
    increment or decrement;
}
```



- Condition is evaluated and if the condition is true then the body of the loop is executed.
- After the execution of the body, increment or decrement part will be executed, the condition is once again evaluated and if it is true, the body is executed once again.



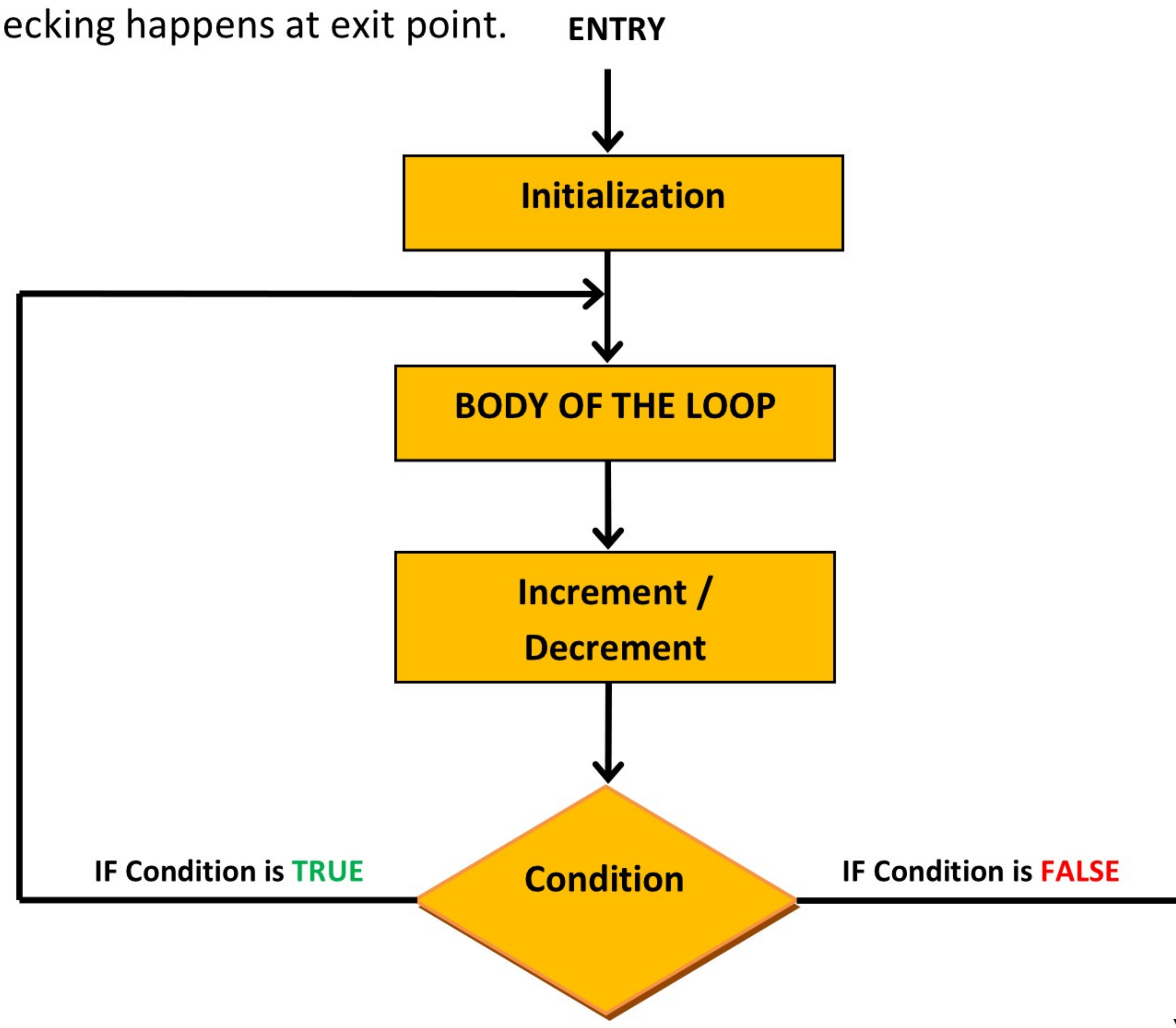
- This process is repeated till the test condition is true. When it becomes false, the control is transferred out of the loop.
- On exit, the program continues with the statements immediately after the body of the loop.
- While loop is also known as *entry control loop* because first condition is checked and if it is true then only body of the loop will be executed.

**Example:**

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

### 2. do...while loop

- In contrast to while loop, the body of the do...while loop is executed first and then the loop condition is checked.
- The body of the loop is executed at least once because do...while loop tests condition at the bottom of the loop after executing the body.
- Do...while loop is also known as *exit control loop* because first body of the loop is executed then increment or decrement statements will execute, then condition is executed, thus condition checking happens at exit point.





- The general format of the do...while statement is:

```
do
{
    statement;
    increment or decrement;
}
while(test-condition);
```

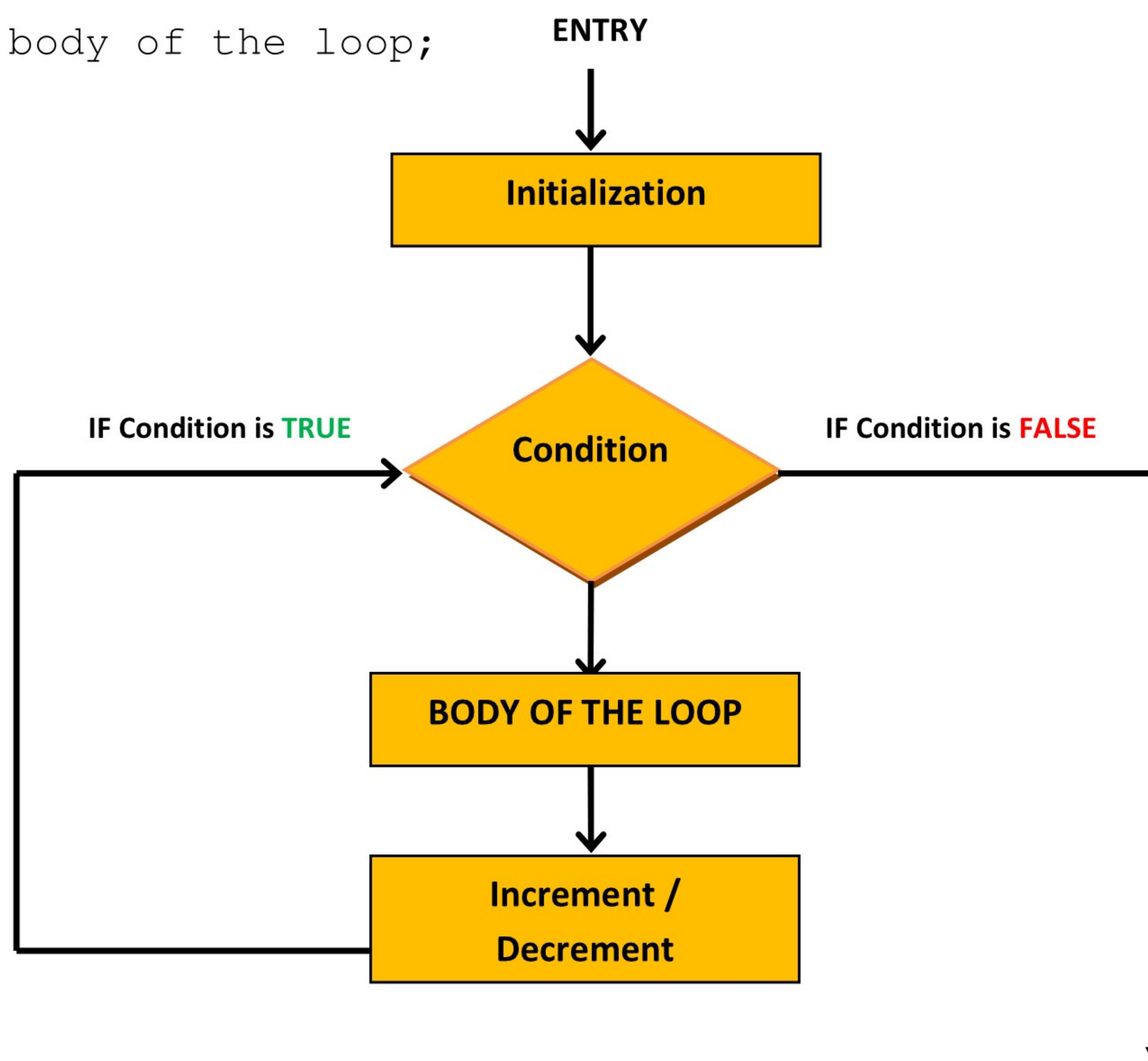
**Example:**

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

### 3. for Loop

- for* loop provides a more concise loop control structure.
- The general form of the for loop is:

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





- When the control enters for loop, the variables used in for loop is initialized with the starting value such as  $i=0$ ,  $count=0$ . Initialization part will be executed exactly one time.
- Then it is checked with the given test condition. If the given condition is satisfied, the control enters into the body of the loop. If condition is not satisfied then it will exit the loop.
- After the completion of the execution of the loop, the control is transferred back to the increment or decrement part of the loop. The control variable is incremented using an assignment statement such as  $i++$
- If new value of the control variable satisfies the loop condition then the body of the loop is again executed. This process goes on till the control variable fails to satisfy the condition.
- For loop is also *entry control loop* because first control-statement is executed and if it is true then only body of the loop will be executed.

**Example:**

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

<u><b>for Loop</b></u>	<u><b>while Loop</b></u>	<u><b>do...while Loop</b></u>
<pre>for( i=1; i &lt; = 10; i++) {     sum = sum + i ; }</pre>	<pre>i=1; while(i&lt;=10) {     sum = sum + i;     i ++; }</pre>	<pre>i=1; do {     sum = sum + i;     i ++; } while(i&lt;=10);</pre>

## 2. Nested loop

- When one for loop is contained inside another for loop then it is known as nested for loop.

**Syntax:**

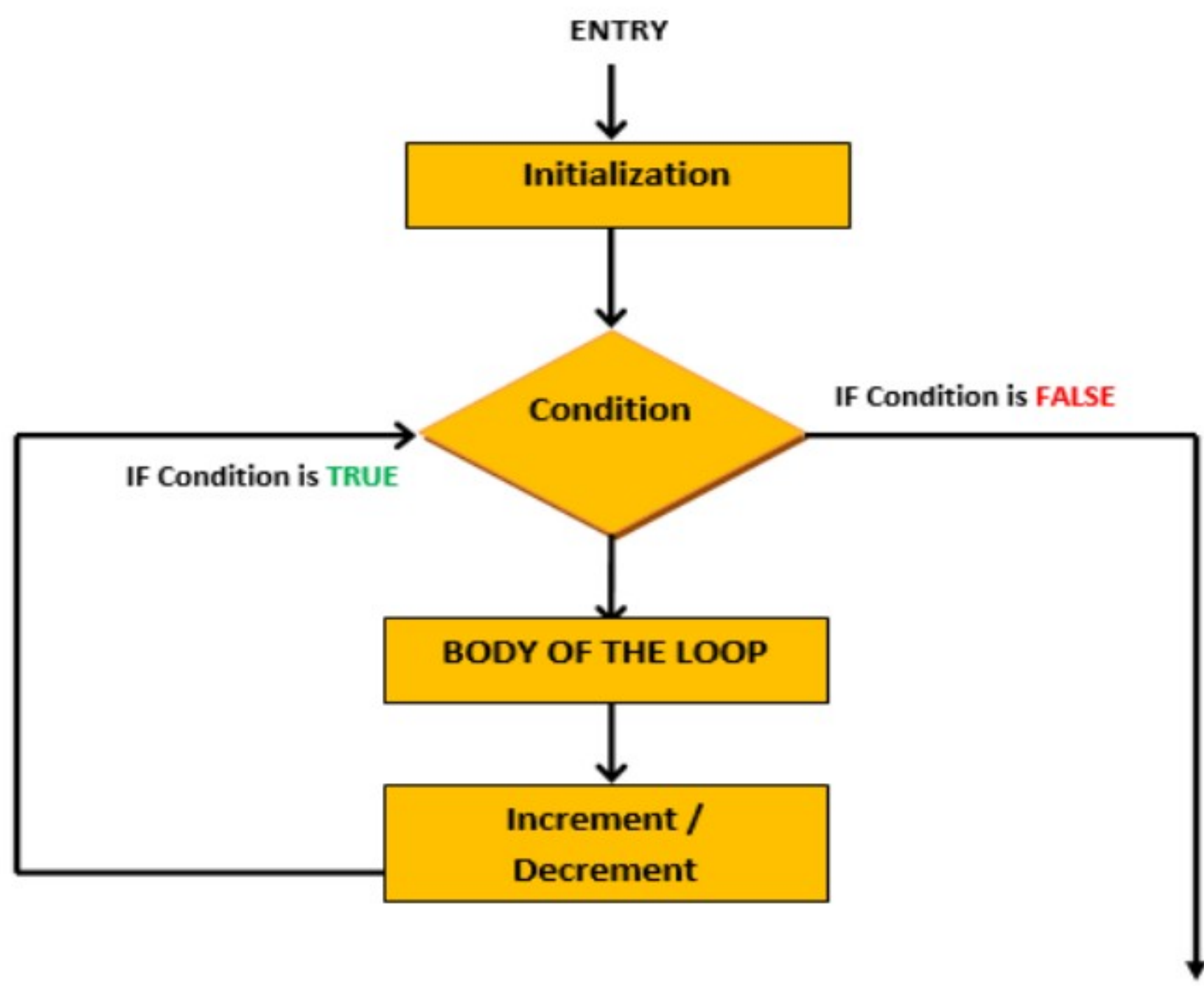
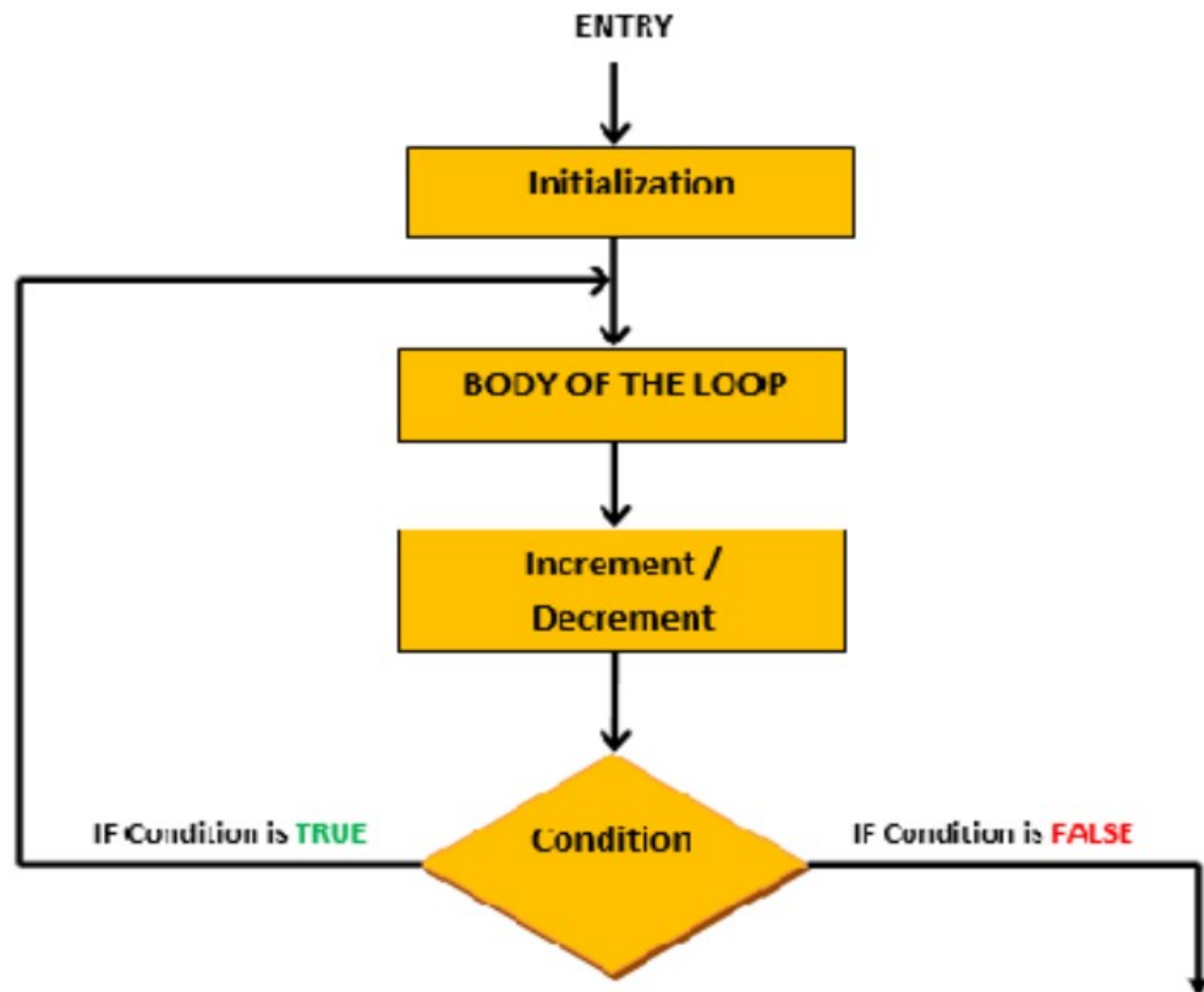
```
for (initialization; condition; increment/decrement)
{
    body of outer loop;
    for (initialization; condition; increment/decrement)
    {
        body of outer loop;
    }
}
```



```

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

### 3. State the difference between entry control loop and exit control loop.

Entry control loop(While Loop)	Exit control loop (do..while Loop)
Entry control loop checks condition first and then body of the loop will be executed.	Exit control loop first executes body of the loop and checks condition at last.
Body of loop may or may not be executed at all.	Body of loop will be executed at least once because condition is checked at last.
for, while are example of entry control loop.	do...while is example of exit control loop.
<b>Syntax:</b> <i>while(condition)</i> { <i>Statement....</i> <i>Statement....</i> } 	<b>Syntax:</b> do { <i>Statements....</i> <i>Statements..</i> }while( <i>condition</i> ); 
<b>Example:</b> <pre> void main() {     int i=1;     while(i &lt;= 10)     {         printf("\t%d",i);         i++;     } }           </pre>	<b>Example:</b> <pre> void main() {     int i=1;     do     {         printf("\t%d",i);         i++;     } while(i &lt;= 10); }           </pre>
 <pre> graph TD     ENTRY --&gt; Init[Initialization]     Init --&gt; Cond{Condition}     Cond -- "IF Condition is TRUE" --&gt; Body[BODY OF THE LOOP]     Body --&gt; IncDec[Increment / Decrement]     IncDec --&gt; Cond     Cond -- "IF Condition is FALSE" --&gt; Exit[ ]   </pre>	 <pre> graph TD     ENTRY --&gt; Init[Initialization]     Init --&gt; Body[BODY OF THE LOOP]     Body --&gt; IncDec[Increment / Decrement]     IncDec --&gt; Cond{Condition}     Cond -- "IF Condition is TRUE" --&gt; Body     Cond -- "IF Condition is FALSE" --&gt; Exit[ ]   </pre>