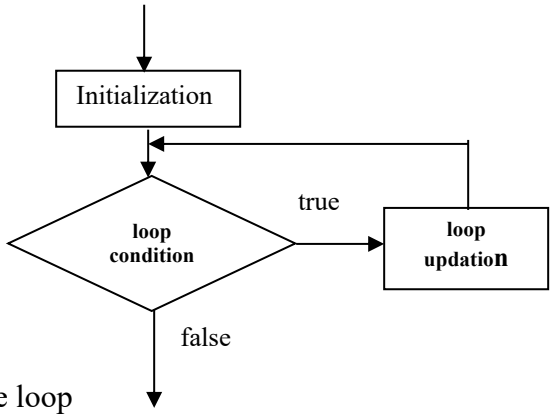
Repetition Statement (Loop)



Pretest loop

- initialization step → to start the loop
- test loop condition → to determine whether to continue the loop processing
- updation within loop body → to update the condition so that, at some point, the loop condition will evaluate to false

- **basic syntax with *while* statement**

```
while (condition)
    statement;          ← body of while statement
                          (Include condition updation statement)

while (condition)
{
    statement 1;
      ……                ← compound statement
    statement n;
 }
```

- **Counter controlled while loop: a loop control variable is used to control the number of times the loop is to be executed. The number of times the loop will be executed is known ahead of time.**

```
int counter = 0;                              ← counter variable
float sum = 0.0;   // accumulator ← set the initial value of an accumulator

while (counter < 5)     {                     ← condition based on the counter value
      cin >> inputData;
      sum = sum + inputData;

      counter = counter + 1;                  ← update the counter
}
cout << "The average of " << counter << " values is " << sum/count << endl;
```

How many times will the body of the statement be executed?
How many times will the logical expression be evaluated?
Trace the program execution.

- **Event driven while loop : loop terminates when a specific event occurs. Do not know how many times the loop will be executed ahead of time.**

```
int    inputData;
int    sum = 0;

cout << "Enter an integer: ";
cin >> inputData;            ← Priming read, initialization step

while (cin)      {           ← test (pre-test)
    sum = sum+inputData;

    cout << "Enter an integer:";
    cin >> inputData;        ← read in the next integer, loop updation
}
```

Change to a slightly different version:

```
int    inputData;
int    sum = 0;

cout << "Enter an integer: ";
while (cin>>inputData)       {          ← loop test (pre-test) and loop updation
    sum = sum+inputData;

    cout << "Enter an integer:";   // prompt for the next value
}
```

Questions:
1. What standard input value causes the standard input stream (cin) to have a value "false" ?
2. Body of the while loop may be executed 0, 1, or more times depending on the input values. What if the input values are : 9, 4, 6, -3, ^d
3. Because of the pretest, the loop may not be executed even once: What happens when the input value is : ^d

- **Sentinel controlled while loop: a special "sentinel" value is defined to indicate the end of input. The typical "sentinel" values are: -1, 9999, 0, … (application dependent)**

```
sum=0;
cout << "Enter an integer:";
cin>> inputData;

while (inputData  != -1)     {   ← -1 is sentinel value here
        sum=sum + inputData;

        cout << "Enter an integer:";
        cin>>inputData;
}
```

- **Infinite loop:** Happens when the logical condition always evaluates to true.

- **++, --, +=, -=, *=, /=, %= operators**

Practice question: Write a C++ program to
(1) read in a sequence of values from keyboard, count the number of positive, negative, and zeros in the sequence of values, the end of input values is signaled using ^d
(2) Reads in a sequence of values from the keyboard, find the largest and smallest values in the sequence, the end of input values is signaled using ^d.
(3) Continues to convert US dollar into Euro for as long as the user wants to perform more conversions.
(4) Find the number of even and odd digits in an integer entered from keyboard
(5) Find the number of values that is greater than its next value, given a sequence of values entered through keyboard

==========

Another while loop example:

Write a C++ program that prompts the user to enter a sequence of 10 integers, the program counts the number of values in the sequence that is greater than the value precedes it. For example, assuming the 10 values entered are : 12, 1, 20, 22, 24, 3, 5, 8, 7, 9, the program should output "there are 6 values that are greater than the value precede it.".

```
#include <iostream>
using namespace std;

int main() {

        int  numRead,  greaterThan;
        int  prevNum, currNum;

        cout << "Enter an integer" << endl;
        cin>> prevNum;

        numRead=0;
        greaterThan=0;
        while (numRead <9)
        {
                cout << "Enter the next integer" << endl;
                cin >> currNum;

                if (currNum > prevNum)
                        greaterThan ++;

                prevNum = currNum;
                numRead ++;
        }

        cout << "There are " << greaterThan
             << " values that are greater than the value precede it." << endl;

        return 0;
}
```

while loop with break and continue statements

- break statement will break out of the current loop

```
int count = 0;
while (count < 10) {

 if (count == 5) {
  break;
 }

 cout << count << endl;
 count++;
}
```

The output of this program segment is:
0
1
2
3
4

5 and other numbers after 6 will not be displayed, because the cout statement is after the break statement.

- continue statement will skip the rest of the statements in the current loop, but will continue with the next iteration of the loop

```
int count = 0;
while (count < 10) {

 count++;

 if (count == 5) {
  continue;
 }

 cout << count << endl;
}
```

The output of this program segment is:
1
2
3
4
6
7
8
9
10

Just number 5 is not displayed