# Loop Vocabulary

Variables and values can have multiple roles, but it is useful to mention three different roles in the context of loops:

Counter

Variable that is incremented every time a given event occurs.

int i = 0; // i is a counter  
while (i < 10){  
 Console.WriteLine($"{i}");  
 i++;  
}

Sentinel Value

A special value that signals that the loop needs to end.

Console.WriteLine("Give me a string.");  
string ans = Console.ReadLine();  
while (ans != "Quit") // The sentinel value is "Quit".  
{  
 Console.WriteLine("Hi!");  
 Console.WriteLine("Enter \"Quit\" to quit, or anything else to continue.");  
 ans = Console.ReadLine();  
}

Accumulator

Variable used to keep the total of several values.

int i = 0, total = 0;  
while (i < 10){  
 total += i; // total is the accumulator.  
 i++;  
}  
  
Console.WriteLine($"The sum from 0 to {i} is {total}.");

We can have an accumulator and a sentinel value at the same time:

Console.WriteLine("Enter a number to sum, or \"Done\" to stop and print the total.");  
string enter = Console.ReadLine();  
int sum = 0;  
while (enter != "Done")  
{  
 sum += int.Parse(enter);  
 Console.WriteLine("Enter a number to sum, or \"Done\" to stop and print the total.");  
 enter = Console.ReadLine();  
}  
Console.WriteLine($"Your total is {sum}.");

You can have counter, accumulator and sentinel values at the same time:

int a = 0;  
int sum = 0;  
int counter = 0;  
Console.WriteLine("Enter an integer, or N to quit.");  
string entered = Console.ReadLine();  
while (entered != "N") // Sentinel value  
{  
 a = int.Parse(entered);  
 sum += a; // Accumulator  
 Console.WriteLine("Enter an integer, or N to quit.");  
 entered = Console.ReadLine();  
 counter++; // counter  
}  
Console.WriteLine($"The average is {sum / (double)counter}");

We can distinguish between three “flavors” of loops (that are not mutually exclusive):

Sentinel controlled loop

The exit condition tests if a variable has (or is different from) a *specific value*.

User controlled loop

The number of iterations depends on the *user*.

Count controlled loop

The number of iterations depends on a *counter*.

Note that a user-controlled loop can be sentinel-controlled (that is the example we just saw), but also count-controlled (“Give me a value, and I will iterate a task that many times”).