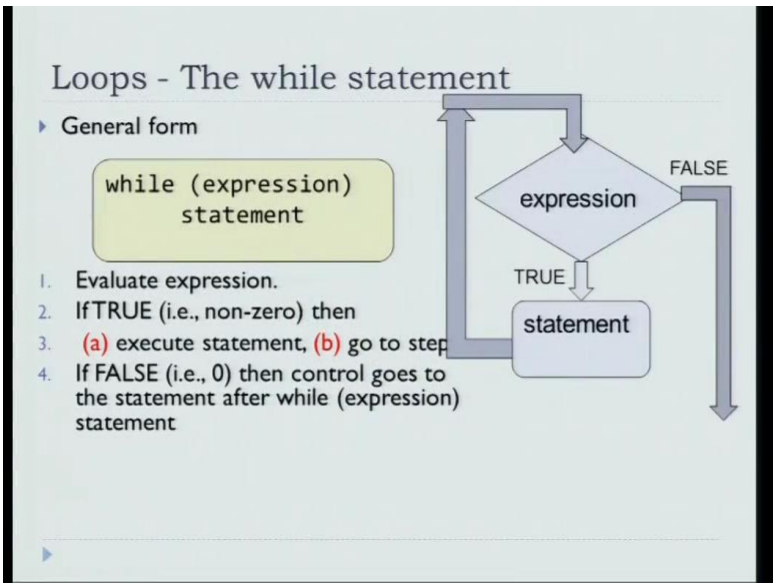
Introduction to Programming in C
Department of Computer Science and Engineering

In this session, we will look at loops in the C Programming language. And we will start with very basic kind of loop which is known as the while statement.

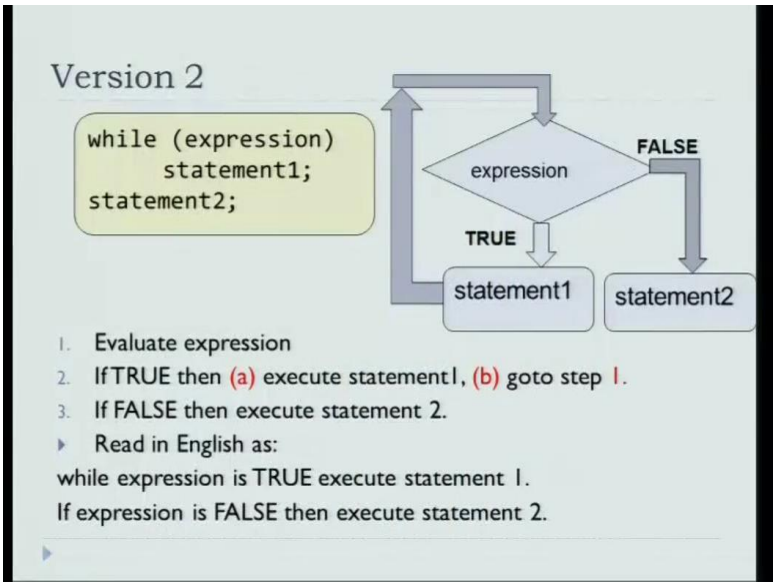
(Refer Slide Time: 00:04)



So, the general form of a while statement this similar to that of and if which is that there is an expression and you say while that expression, then do this statement. So, the flow chart corresponding to the while expression will be, you test whether the expression is true or false. If it is true you do this statement, if it is false you exit out of the loop and execute the next statement outside the loop.

So, if the expression is true in C that is the expression is non-zero, then execute the statement and go to the step outside the loop. If it is false then directly go outside the next statement after the while loop. This is similar if you recall to the, if block without the else. So, loops are a new thing that explicitly there was no loop construct in a flow chart, we just had this way of going back to the expression. But, in programming languages loops are such a basic programming need that in addition to the if block, you have loop construct as well.

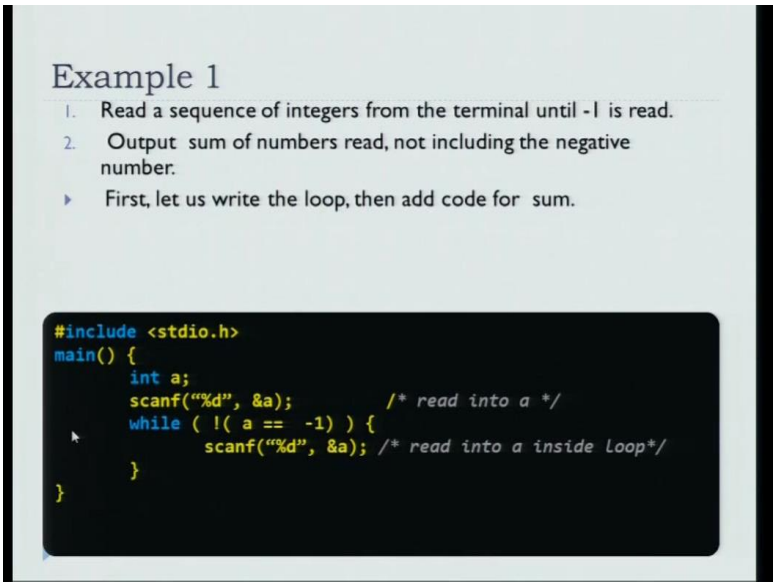
(Refer Slide Time: 01:39)



So, slightly different variant of the while expression will be that while expression statement 1 and then statement 2. So, the flow chart here is easy to follow if the... So, if first test whether the expression is true. If the expression is true then you execute statement 1. And then after you execute statement 1, then go to go back to the expression. If the expression is false then you go to statement 2. So, while the expression is true execute statement 1 and if the expression is false, then execute statement 2.

So, the difference in the, if condition will be that if this was an if block. Then, if the expression is true you do statement 1 and you exit out of the while loop. And that is not done in the case of a normal while loop. After you execute the statement you go back to the expression. So, as long as the expression is true you keep executing statement 1 and if the expression becomes false then you execute statement 2.

(Refer Slide Time: 02:47)



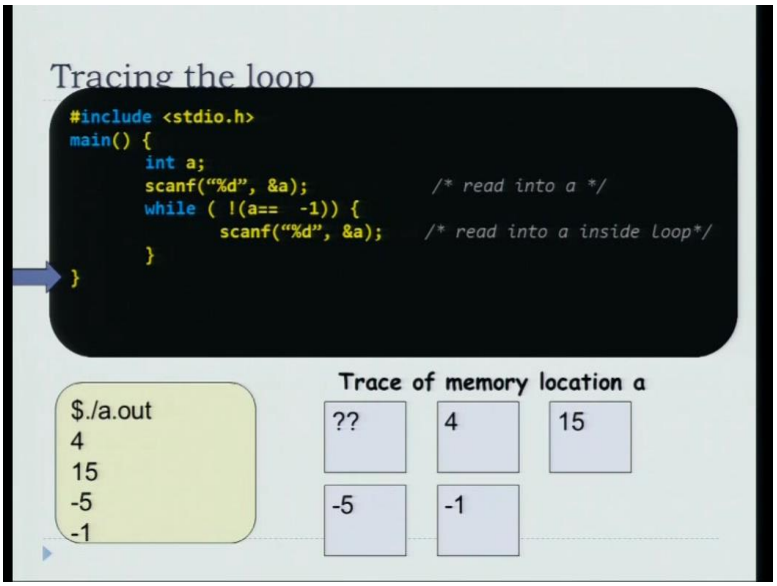
So, let us illustrate the use of a while loop with the help of a program. So, we will introduce a very simple problem which is, read a sequence of integers from the terminal until -1 is encountered. So, -1 signals that the input is at end. Now, what I want do is that sum up all the numbers until the -1 and output the sum. It is a very simple program. What you have to do is to read a sequence of numbers, until you hit the first -1 and then add this numbers and output their sum.

So, let us first introduce the very simple loop which will do only the basic thing of reading the numbers until a -1 is encounter. So, how do you write the loop you have stdio.h. And then you declare an integer variable and read the variable. So, this is supposed to be the first number. If that number is -1 then you do not have to read any more numbers. So, if the number is not -1. So, if a = -1 is false then you read one more number.

After you read one more number you do not finish the loop, you go back and test whether the loop condition is still true. So, you go back and check whether the second number you read was -1 or not. And then, you keep on reading it until you hit a -1. At some point when you hit a -1 you go back to the loop and the condition that a = -1 will be true. So, NOT of that will be false and you will exit the loop.

So, read the first number if it is a -1 do not enter the loop; otherwise, keep on reading numbers until you hit a -1. That is the meaning of the while loop.

(Refer Slide Time: 04:50)



So, let us just trace the execution of the loop on a sample input to understand how it works. So, in a box I will represent the memory location a and its current content. So, I run the program after compiling a.out and let us say that I enter the number 4. Now, you scan the number 4. So, memory location a becomes 4. Now, 4 is not -1. So, you enter the loop. So, then n let us say the next number is 50, you read the number into a. So, memory location a is now 15, 15 is not -1.

So, you again enter the loop, you enter -5, -5 is not -1. So, you enter the loop again. At this point you enter, you scan the number into a and a becomes -1. So, you go back to the loop again and now the test that so, a = -1, so, naught of that is false. So, the while condition becomes false at this point you exit the program. So, this is a very simple part of the program that we want to write, recall that we want to read a bunch of numbers and sum them. And the end of the numbers is represented by a -1. Until now we have just read those numbers.