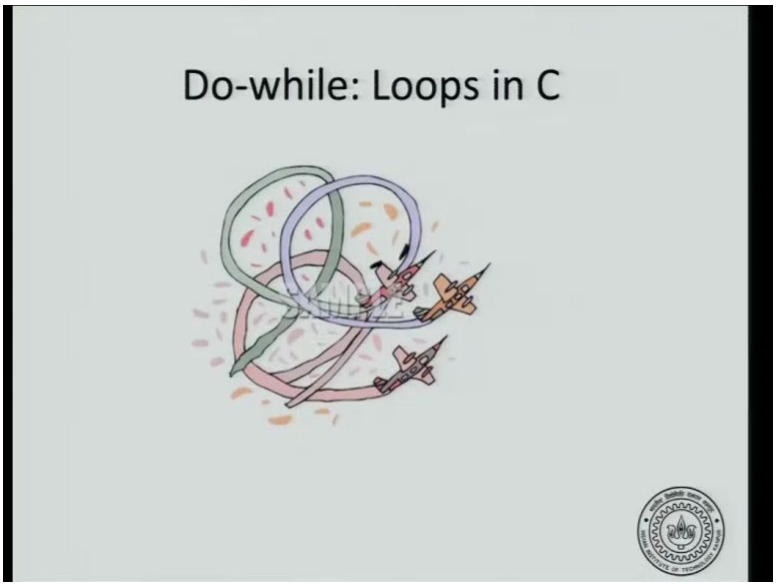
Introduction to Programming in C
Department of Computer Science and Engineering

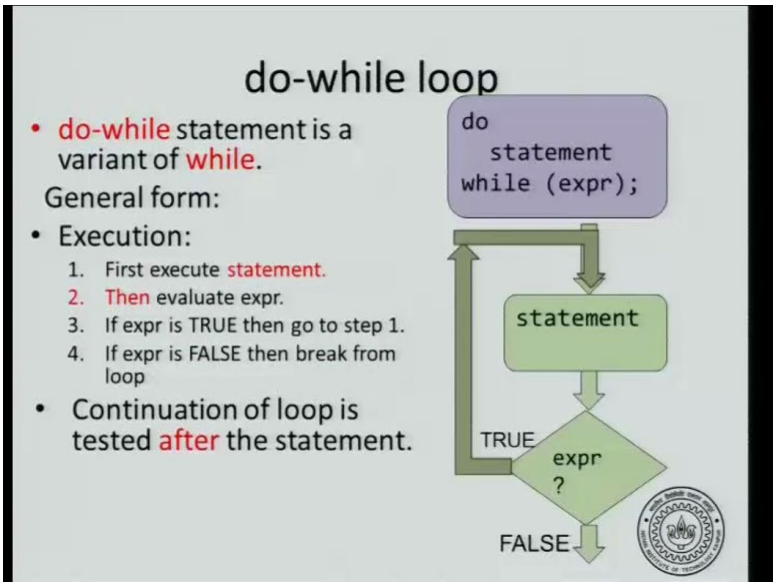
So far we have been using while loops in C, now C programming language also provides you other kinds of loops.

(Refer Slide Time: 00:12)



Let's look at some of them. The first alternative loop mechanism in C that we will look at is what is known as a do-while loop?

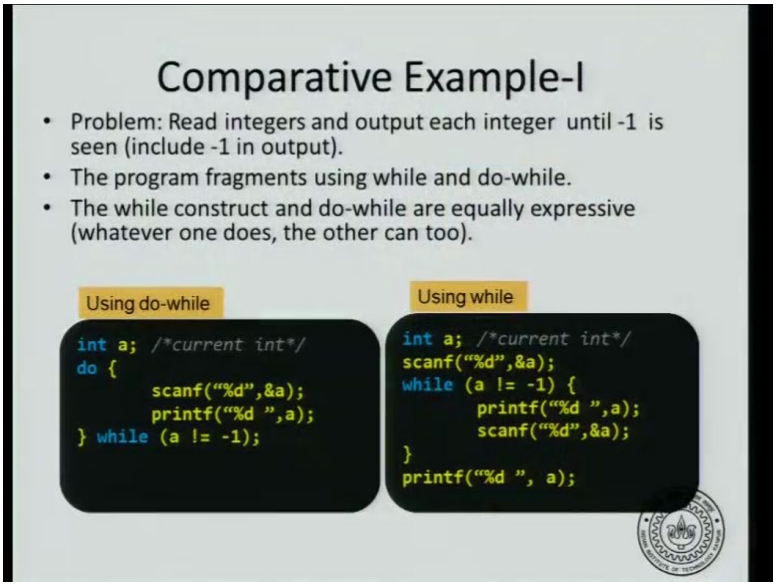
(Refer Slide Time: 00:20)



And so it is a variant of a while loop, I am the general form is what you see here, you have do statement followed by while expression. And here is an important syntactic difference which causes some syntax errors, when you code. The do while terminates within semicolon, where is the while loop does? So, the while loop has the following a form which is while expression, and then statement; the difference is that here the statement is occurring before the while the test expression. So, the way it execute is the following. We first execute the statement, then evaluate the expression. If the expression is true, you go back to step 1; that is execute this statement. If the expression is false, then you get out.

So, you execute the x statement then test whether the expression is true or not, if it is true you go back and the execute the statement again, so you loop. If the statement is false, you get out of the loop. The difference from while loop and do while loop is the following, you have statement that will be executed without testing the expression even once. So, when you start executing the loop, you will first execute the statement without testing the expression, and after testing the expression you will go back and test the loop expression, if it is true and you start executing the loop again. So, the first execution of the statement there is no test done for that.

(Refer Slide Time: 02:13)



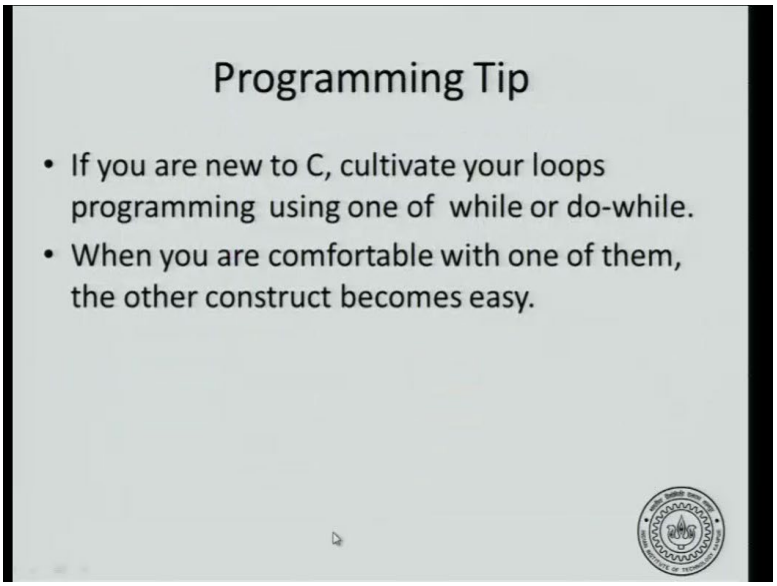
So, let us see the comparison between a loop while loop and do while loop. So, we will look at the following problem, you have to read numbers and output in integer until a -1

is seen. Now the difference is that in this problem you have to include the -1. So, read all the numbers up to an including -1, and print all the numbers. So, we will have the following programs using while loop and do while loop. Now the important thing to notice is that the while construct and the do while construct are equally expressive. So, you cannot right any more new programs using the do while construct, then you could using the while construct, but certain kinds of programs or easier using or shorted using the do while construct. For example let us all this problem using the while construct. So, what you do initially is, you declare a variable then scan the variable; if the variable is -1, you immediately exit out of the loop, and print -1 and finish the program.

If the number is not -1, you print the value and scan the next number. If the number you scan this not -1, you just print it and repeat the loop. If it is -1, you exit out of the loop and print the -1 that you show. So, here is the logic using the do while loop, in using the while loop. And notice that when we existed out of the loop we needed a printf statement, and before you yes, enter the loop you needed a scanf statement. So, this was the structure of the program. This problem can be elegantly solved using the do while loop. What you initially need to do is to declare a variable, then scan the variable and print it any way. Either the number is -1 or it is not. In any case we need to print it.

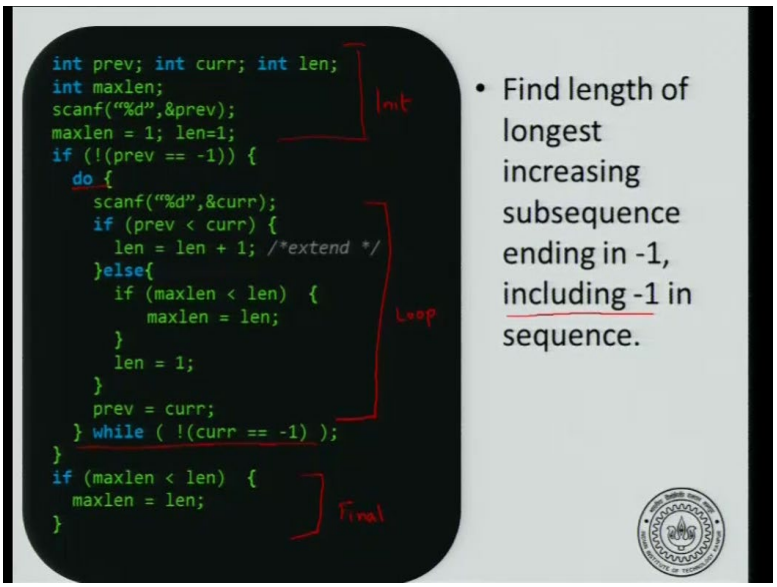
So, go ahead and print it then test whether the number was -1. If it is -1, your done and you exit out of the program. If it is not -1, you go back and scan the next number and print it. So, this is a program that we have seen where you could do this same think with the while loop. The only difference is that the do while program is shorter. And please be careful about the syntactic difference between the while loop and the do while loop, notice the semicolon at the end this causes a lot of confusion when you compile the program it is easy to miss this.

(Refer Slide Time: 05:05)



If you are new to C programming, you can strict to one particular loop. As I said before you cannot write any new programs that you can do is using the do while loop, then you could previously do using the while loop. So, you can write the same logic, you can write the same number of programs using the while loop, and the do while loop it gives you no further power. So, it is recommended that you stick to one loop pick while or pick do while whatever you do, but stick to that loop in when you right the program. When you are comfortable with one of the loops programming using the other loop becomes easy.

(Refer Slide time: 05:51)



So, let us try to solve a problem that we have already seen, which is to find the length of the longest contiguous increasing subsequence ending in -1. The difference that we have is that earlier we did not include -1 in the sequence when you computed the length of the sequence, now we will include -1. So, here is the program to do that and the logic - the core logic, so here is the initialization, and here is the loop logic, and the final check.

So, if you recall from the lecture which covered the problem solving the longest increasing subsequence, then you will see that the main structures in the code. The main lines of logic in the code are pretty much the same. All I have done is to change the while logic to the do while logic. And let see what that is accomplish for us. So, what this does is that you will scan a particular number, if the particular number is bigger than the previous number, then you extend the sequence. If it is less than or equal to the previous number, then you stop this sequence and started new sequence, this was the logic.

And when you start a new sequence the length is new start with 1. Then you say current equal to the next number, and previous equal to the number that was just red. So, the logic here is that the testing for whether the currently rate number is -1 is done at the end of the loop. So, is the first number is -1, you just do all this and then say that the length of the increasing subsequence is 1, then you test if the currently read number is -1 or not. If the currently read number is -1, then you are already done and you exit out of the loop. Then you check whether max length is less then length as before. ((Refer Time: 08:26)) difference between this logic, and the logic that we have seen before is that we do this execution without testing whether the currently read number is -1. So, automatically what happens is that if the number is -1, all these steps will be performed before we test that the sequence has ended. So, automatically we ensure that -1 is also included when we calculate the increasing subsequence.