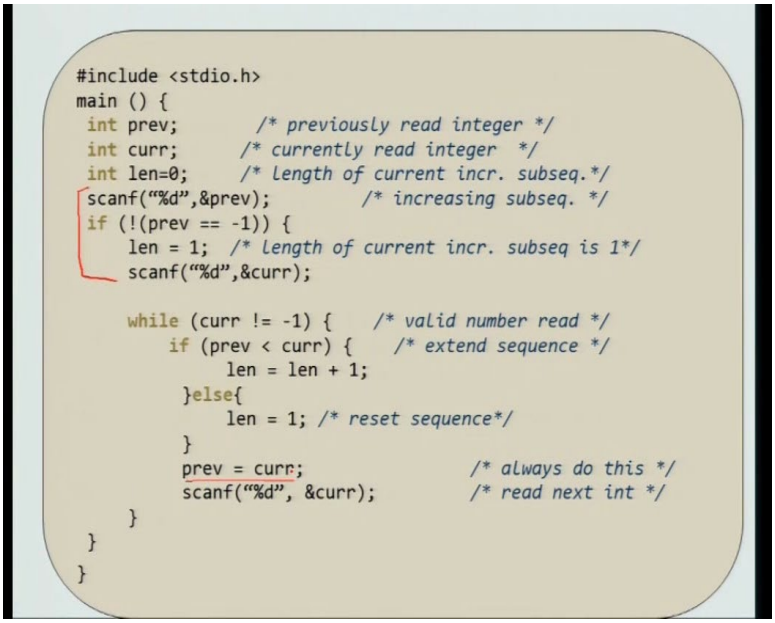
Introduction to Programming in C
Department of Computer Science and Engineering

(Refer Slide Time: 00:31)



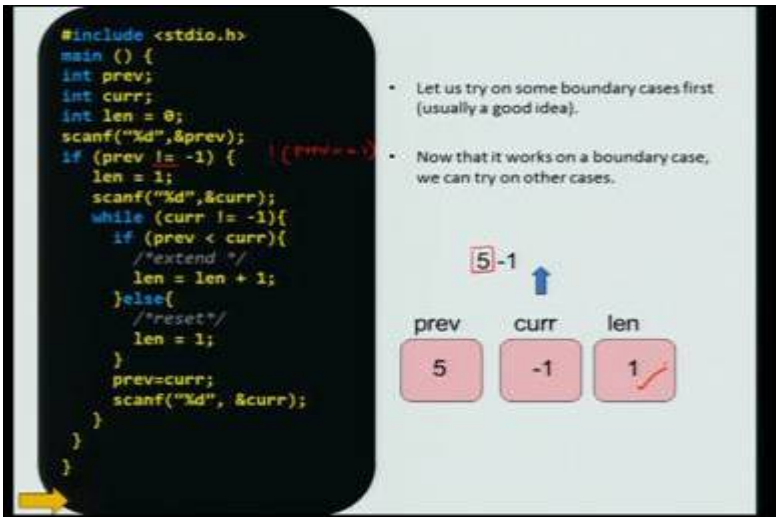
In this session we will try to code up the C code for finding the length of the longest increasing contiguous sub sequence. So, let us first examine what we need to do, we will write a code, and from the previous discussion we saw that we need at least three variables; one for a storing the previous number, one for storing the current number, and the third for storing the length of the current decreasing sub sequence. So, we start by declaring all those three variables and initializing length to 0. So, here is a new construct that we are seeing for the first time, which is that when you declare a variable, you can also initialize it immediately by saying len equal to 0. So, this is a very intuitive notation. So, this will declare a variable and immediately initialized it to 0. Once we declare these three variables, let say that we scan the first variable into previous.

Now, let us focus on, on the main body of the program. If the currently read number is if the currently read number is ! = -1, then you say that you start with length 1. So, the length of the current increasing sub sequences 1, and then you scan the next number into curr. So, here is current number. So, this part of the code is just to initialize. So, if the current, if the first number is -1, then there is no point in getting into the program, because the its equivalent to the empty inputs. So, there is no increasing sub sequence to be found. So, you just exit out of the program ((Refer Time: 01:58)). So, initially we just check to see

whether the first number is -1 or not. If the first number is ! = -1, you scan the next number, so current will be the second number. And if current is ! = -1 while the currently read number is ! = -1. What you do is exactly the logic that we were discussing before. If the previous number is less than the current number then you extend the length by 1.

So, length equal to length plus 1 says that I am continuing the current increasing sub sequence by increasing its length. Otherwise that is current is less than or equal to previous, you break the sequence and say length equal to 1. Then we have this step previous equal to current, which is the advancing both variables by 1. So, previous becomes the currently read number, and current becomes the next number to be ((Refer Slide: 03:02)). So, recall from the diagram that previous and current were at some position, and we will advance both of them by 1. And when the loop condition is check the next time we will check whether the currently seen number is -1. So, so far we have coded up part of the logic, which is the part of the logic dealing with when the current when the next number is read do we extend the sequence or do we break the sequence and start a new sequence. So, this is just part of the work that we need to do to solve the problem. So, lets continue with the logic.

(Refer Slide Time: 03:40)

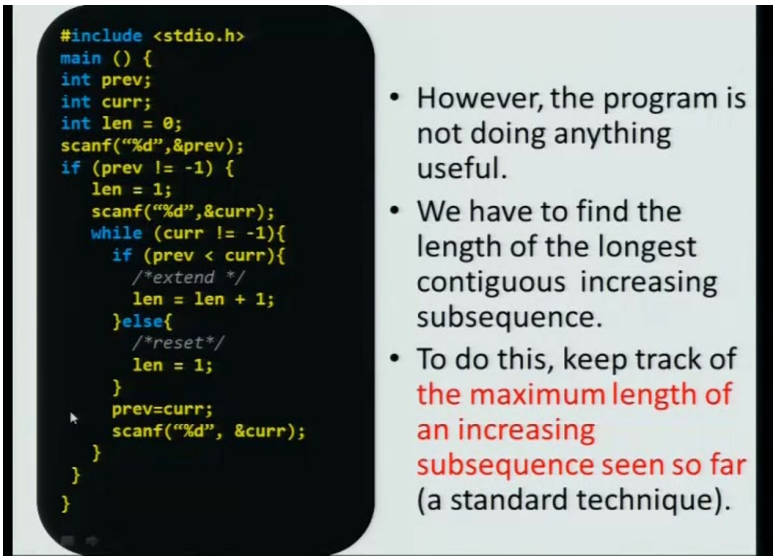


So, let us start with a few boundary cases, and let see that whether these works. If it works we can try or logic on other cases. So, let us say that by boundary cases I mean may be very long inputs or very short inputs. So, these are cases where your code normally breaks. So, when you test your code it is always a good idea to check boundary cases. And one thing that makes programming difficult is that in or when we do things by hand, we know

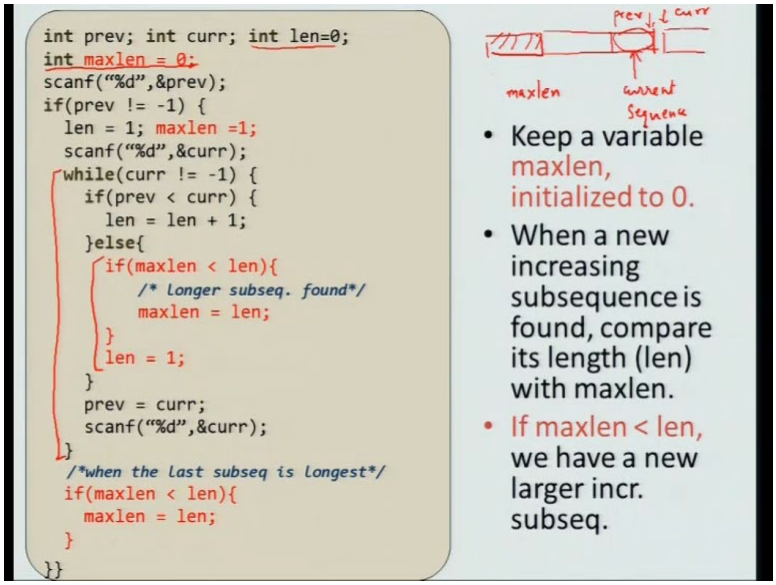
how to handle the boundary cases elegantly, but in a program unless you say how to handle the boundary cases, the program might break. And the lot of testing and the lot of errors come from incorrectly handling the boundary cases. So, it is always good to handle the boundary cases, let us try test our code on very small inputs.

So, let us say that I enter a sequence 5 - 1. So, previous becomes, so length is 0; previous becomes 5 and then since prev ! = -1, I have used an abbreviation here which is the not equal to operator, this is the same as saying not of previous equal to equal to 1. So, previous not equal to 1 is the same as saying not of previous equal to equal to 1, its an operator in C, then you say that the len equal to 1, then you say that the length is 1, because you have seen 1, 1 number and scan the next number. The next number is -1. So, you scan the next number, and the next number is -1, so you break the sequence. So, immediately exit out of the sequence, and the length of the increasing sub sequence that we saw, so 5 is the only increasing sub sequence. And when we exited the length was 1. So, we handle the boundary case of an extremely small sequence, a sequence with exactly 1 element correctly. This gives us confidence that the code could be correct, of course we have handle only the boundary case. Now, we see need to test it for other cases as well.

(Refer Slide Time: 06:17)



So far the program is not doing any anything useful, because we are just extending the sequence and breaking the sequence. What recall that the what we was suppose to do was to fine the length of the longest increasing sub sequence. So, this is the main, this the main



thing that we have to do in the logic. So, to do this what we do is something simple. We keep track of the maximum length sequence that we have seen so far, keep track of the length of the longest sequence that we have seen so far. Also we have the current sequence. Now all we need to do is whether to check whether the current sequence is longer than the previously known longest one. So, for this what we do is keep track of the maximum length that we have seen so far. So, this is a standard technique in program. And how do I do that.

So, let us modify the program a little bit. So, earlier we resend out that we need at least three variables. Now in order to keep track of the length the maximum length that we have seen so far, I need a new variable. So, this part we have already done before. And here is the maxlen = 0. So, that is the new variable which is the maximum length that we have seen so far. When we start the program we have not seen any increasing sub sequence, and therefore the length of the longest increasing sequence, the current increasing sequence is 0; that is len equal to 0. And the length of the maximum length that we have seen so far is also 0.

Then you scan the new number; if the new number is not -1 you continue. So, length equal to 1, now max length equal to 1, because currently the longest sequence that we have seen so far is 1 1 long. You scan the num next number. If you… So, here is the main body of the loop, and what we need to do is the following, if the currently read number is greater than the previous number, we extend the sequence. So, this logic is the same as before. Otherwise which means that current number is that less than or equal to previous. So, we are starting the new sequence. So, the situation is the following we have some maxlen

sequence somewhere in the past. So, maxlen is the length of the sequence that we have seen somewhere in the past.

Now we are scanning in the sequence we have a current sequence. And we have decided to break this sequence. So, we have we are now starting a new sequence starting at current. So, we are a this part of the logic. So, we have decided to start a new sequence, that is because the current sequences last number is greater than or equal to the current number. So, here is previous, and this is current. So, we are deciding to start a new sequence what we need to see is whether this sequence is longer than the previously known maximum length. If the sequence that we just stopped is longer than the previously known maximum length sequence. So, if maxlen < len notice that length is then sequence that length of the sequence that we just stopped. Then we say that maxlen = len. So, if the current sequence is longer than the previously known maxlen, what we do is that maxlen becomes the length of this sequence. Otherwise if the current sequence was shorter than the previously known maximum length, we do not do anything, so maximum length is the same…