

**Introduction to Programming in C**  
**Prof. Satyadev Nandakumar**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kanpur**

**Lecture - 12**

(Refer Slide Time: 00:27)

```
int prev; int curr; int len=0;
int maxlen = 0;
scanf("%d",&prev);
if(prev != -1) {
    len = 1; maxlen =1;
    scanf("%d",&curr);
    while(curr != -1) {
        if(prev < curr) {
            len = len + 1;
        }else{
            if(maxlen < len){
                /* Longer subseq. found*/
                maxlen = len;
            }
            len = 1;
        }
        prev = curr;
        scanf("%d",&curr);
    }
    /*when the Last subseq is Longest*/
    if(maxlen < len){
        maxlen = len;
    }
}
```

- Keep a variable **maxlen**, initialized to 0.
- When a new increasing subsequence is found, compare its length (len) with maxlen.
- If **maxlen < len**, we have a new larger incr. subseq.

So, when we extend the sequence we do not have to do anything special when we break a sequence, and we start a new sequence, then all we have to do is you check whether the currently say the sequence that you just saw was longer than the previously known longest sequence. If that is the case then the sequence that just ended is becoming the longer sequence we have seen so far. Otherwise you maintain the max length. So, forget about the currently stop sequence. Now there is a... So, that this loop, and at the end we have to do slight tricky logic, it could so happen that the sequence ends with a longest sequence increasing sub sequence. In that case, we will never reset the max length. So, if the last sequence is the longest, you also have to handle the case separately. So, we will see an example where, if you exit out of the loop that is you have already seen a minus 1, you just have to check whether the last increasing sequence that you show was in fact the longest. So, there is a small if block at the end to do that.

(Refer Slide Time: 01:25)

The image shows a C program on the left and its execution state on the right. The code is as follows:

```
int prev; int curr; int len=0;
int maxlen = 0;
scanf("%d",&prev);
if(prev != -1) {
    len = 1; maxlen =1;
    scanf("%d",&curr);
    while(curr != -1) {
        if(prev < curr) {
            len = len + 1;
        }else{
            if(maxlen < len){
                /* Longer subseq. found*/
                maxlen = len;
            }
            len = 1;
        }
        prev = curr;
        scanf("%d",&curr);
    }
    /*when the Last subseq is Longest*/
    if(maxlen < len){
        maxlen = len;
    }
}
```

The execution state on the right shows the input sequence 3 2 1 3 5 -1. The variables are tracked in a table:

prev	curr	len	maxlen
3	2	0	0
2	1	1	1
1	3	2	3
3	5	3	
5	-1		

In this part we will just see, small tracing of this program on a sample input. So, that the logic of the program become slightly more clear. So, I have pit a particular input 3 2 1 3 5 minus 1, and you will see that the longest increasing sequences are 3. So, the increasing sequences are 3, then 2, then 1 3 5. So, 1 3 5 is going to with a longest increasing sub sequence, and let us see how are program will find that out. So, initially you have a bunch of variables which should be declare. So, length is 0, max length is 0, and previous and current are undefined. Then you first read previous. So, previous becomes 3, it is not minus 1, so you enter the if condition, at which point you set length and max length to 1.

Now you scan the current number. So, current becomes 2. So, remember that previous is now 3, and current is 2. So, current is not minus 1, therefore you enter the while loop. Previous is less than current is false, because previous is 3 and current is 2. Therefore, you enter the else part, max length less than length is false; both are 1. Therefore, you start a new sequence with length equal to 1. Now you continue the loop with previous becoming current.

So, previous is now 2 and current you read the next number which is 1. So, previous and current have both more 1 step. So, current is not minus 1, previous less than current is again false, because 2 is greater than 1. So, you enter the else part. Max length and length

there is no change. So, you reset the length to 1, previous is current. So, current previous becomes 1, and you scan the next number which is 3. Now at this point previous is 1, and current is 3. So, the if condition is true. So, you extent the length; length increases by 1. Again you advance previous and current. So, previous becomes 3, current becomes 5. Again 3 is less than 5, so increase the length we are extending the sequence. So, the length becomes 3. Advance, so previous becomes 5, and current becomes minus 1 at this point you exit the loop.

And now you encounter the situation that max length, which is the length that we have seen so far, recall that it is one, but there length of the sequence that we just top the see the input with is 3; that is that happen, because the longest increasing ((Refer Time: 05:09)) sub sequence, well was at the end of the input. So, it happen right of the end. So, when we exit the loop we have to do 1 additional check, we cannot simply say that the maximum length that we have seen in the sequence is 1, because max length is the length of the longest sequence we have seen before the current 1. The current 1 was the 1 that we just stop to with it had a length of 3. So, we just check, if max length equal to length is less than the length, then we said max length to be the length. So, once you do that max length becomes 3. So, you just a handle the case when the longest increasing sub sequence is the last. Now you can exit out of the if condition, and then print that the maximum length that you have seen is 3.