

Problem Solving through Programming In C
Prof. Anupam Basu
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Lecture - 42
“search” as a function

We have seen functions in the earlier lecture. And we have also seen we tried to solve a problem with where there were names of the students in one array, and the marks in another array. And we tried to find out try to print out the name of the student who got the highest marks so, for that we had 2 different arrays. Now the same thing I can do today say I have 10 students.

(Refer Slide Time: 00:51)

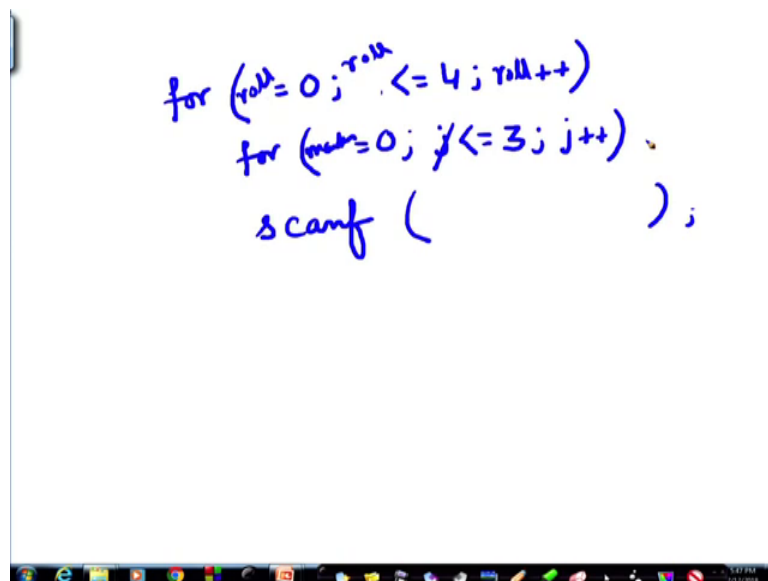
Roll No	L	S	M	H	T
1	50	25	70	35	✓
2	70	20	90	15	✓
3	35	75	90	80	✓
4	20	25	60	30	✓
5	10	20	30	40	✓

And each of them have appeared for, I am so sorry, each of them has appeared for 4 subjects, all right. Or let us say there are 5 students, and each of them has appeared for 4 subjects. So, how do I, what would be a convenient way of representing them? One convenient way of representing them would be to have a 2-dimensional matrix, 2-dimensional array, where on this side I will have the roll number of the students, which is an integer and I have got 5 students. So, 1, 2, 3, 4 and 5, 5 students, and the marks as the columns right, for any problem solving we will have to think of how can we represent the data so that our computation is made easy. So, at our computation is facilitated.

So, suppose there are 5 students say roll number one, roll number 2, roll number 3, roll number 4, roll number 5, and for each of them we have got 4 columns for the 4 subjects, right. Suppose the subjects are say language science, mathematics and say history, 4 subjects right. And so, how would I read the marks? How would I first of all store the marks of the students, you can very easily understand now, now I am not writing the program I am leaving the writing of the program to you.

But let us try to understand how it can possibly be solved. So, I will have a function, I can write a function of acquiring data. What will that do? For on this side, I can have an index i , that will talk about the row that I am talking about; that means, which student I am talking about and another index maybe j or instead of i j you can say roll and marks, that is also fine ok. So, keeping i fixed, and then you can fill up the row of the marks of the student. So, how would that program segment look like? That program segment will look like for i assigned 0, i less than equal to 4, because there are 5 students, i plus plus.

(Refer Slide Time: 04:04)



```
for (roll = 0; roll <= 4; roll++)  
for (marks = 0; marks <= 3; marks++)  
scanf ( );
```

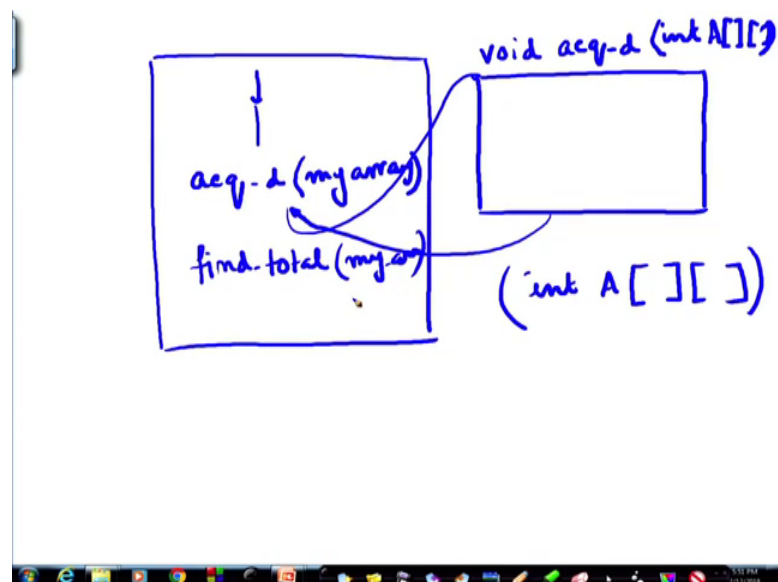
That is for one row for j equals 0, j less than equal to 3, because there are 4 columns j plus plus, scanf now you know what to write within the scanf ok.

So, in that way I can read the marks. What I was saying is that instead of this I can see for, I can be very clear about it, for roll equal to 0 roll less than equal to 4 roll plus plus, sorry, roll plus plus. And for j similarly I can make marks. So, marks 0, marks less than 3, marks plus plus it is a train that way I can fill up the table ok. So now, in that way, I

can fill up the table and say, I get the marks as 50, 25, 70, 35 like that some numbers. Now I can have another function that will find out the total marks of each student. So, right now I do not have a space for that.

So, I can keep that in mind when I am preparing my table, then I can have the fifth column for the total marks, right. Now I will pass on I have read this. So, first of all, I in from my main program, I went to a particular function I called a particular function, which acquired the data and filled up all these. Say, 70, 20, 90, 15, 35, 75, 90, 80, 20, 25, 60, 30, 10, 20, 30, 40. Like that that it has been prepared. So, quickly let us see what we did.

(Refer Slide Time: 07:14)



We had we entered into a main function. And from the main function, I called invoked a function, which is acquired data, and what should the type of that function be acquired data, is it returning anything? No.

So, it should be something like void acquire data d. And where is it acquiring the data? In the integer array, int say let us call that array A and there should be it is a 2-dimensional array. So, I should have something like this. This time writing clearly int a if that array was a, that was the parameter. Now so, here that was array A, and here it has been called like acquire data, say, my array. And also the size has been specified. The size is suppose globally specified.

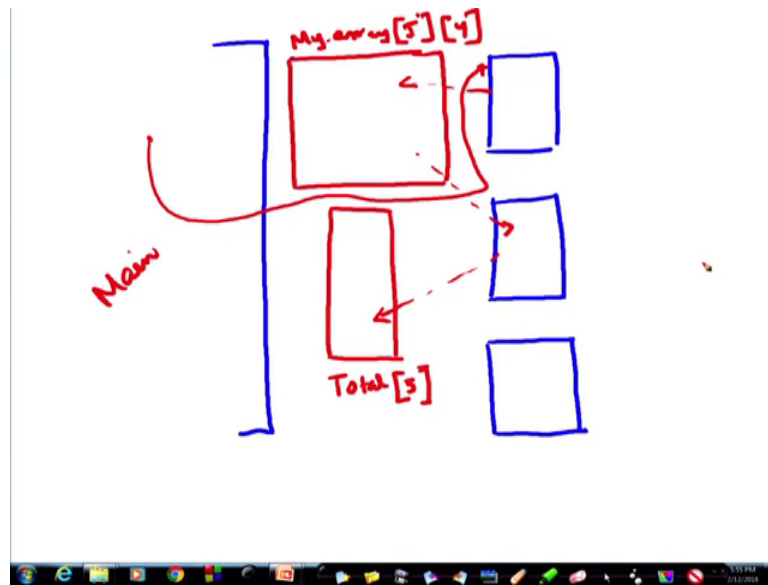
So, I get that array size. Suppose it is defined that row roll size is 5 and subject size is 4. So, I know, but this says subject size plus 1. So, I needed one more. So, I can I can go here and from there, I returned here. And then I can call another fine total of my array. Then what will you do? Essentially, I come to this, and what does that function do that piece of function; that is, finding out the total of each student.

So, that will again be a for loop like the one that we had done earlier, and I will add keep I to be fixed, and add these and write the total here. Now I could have kept it here, or I can have another array total let us do that, suppose I am not having I am just having the marks here. And the total array is not here, but a separate array all together, all right.

So, I have another array which is total. So, I can have the sum of each of them being passed to this. So, in that case when I am calling, when I am calling fine total then I am passing on this array as well as the total array 2 arrays because I want to put the sum on the total array ok. With that I can find the sum of this sum of this, sum of this, sum of this, sum of this. And then I can make another function, where it finds the max or whatever I want to do.

So, accordingly the point is that for each of the activities, I can make small small functions. One is for acquiring data; one is for finding the total. One is for finding the maximum number, or maybe some other things also sorting and all those. I can do, I can find out the failure list another function. So, I can have a complete set of small small functions.

(Refer Slide Time: 11:36)



Each independently done, say did acquiring data, finding total, finding max, all these and they are communicating through a main program, and the main program my data is one array. So, let me let me draw the data using red.

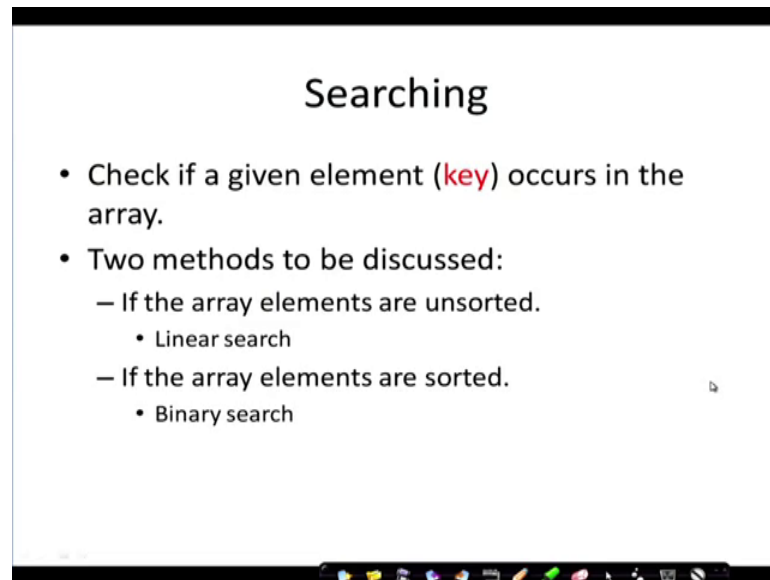
So, here I have got my array, which was called my array, array which had the size 5, 5 and 4. And I had another array total, total which is a one-dimensional array how many? 5 students so these 2 so, the main function is invoking from different points for different tasks these functions. And this function is working on this array, shared array. This function is working on this array and generating the total, maybe this function is doing something else.

So, that these arrays are being shared by the main function and the small functions. That is how using functions I can divide a task into smaller sub tasks and can do it very nicely. Now with this, now let us move to another problem one a version of which we had encountered earlier; that is, searching an array. When we have got an array for example, here when he found say suppose, I found the total here. I found the total, and from there I find out which one is a maximum or is it that if there is any student whose total is less than 100, I want to test that.

So, I have to search this array. Similarly, I can search if there is any student, who has got in science more than 90. I can search that along this array, all right? I find if there is any 40 is the pass mark, if there is any student. Who has not passed in maths? I find here

there is a failure case. So, for that I need to do a search. So, search is very fundamental, search is absolutely fundamental to computation. And so, first we have already seen linear search, but I once again have a journey through that ok.

(Refer Slide Time: 14:46)



The slide is titled "Searching" and contains the following text:

- Check if a given element (**key**) occurs in the array.
- Two methods to be discussed:
 - If the array elements are unsorted.
 - Linear search
 - If the array elements are sorted.
 - Binary search

So, we know that in a search we are given a key which we are searching for a particular key that we want to find out, if the key is present in the array, then we have to say that the key is present and also the position where the key is present, all right. Now we will consider 2 cases. One is that the array is unsorted.

What do I mean by unsorted? By unsorted I mean that the elements are not in any particular order.

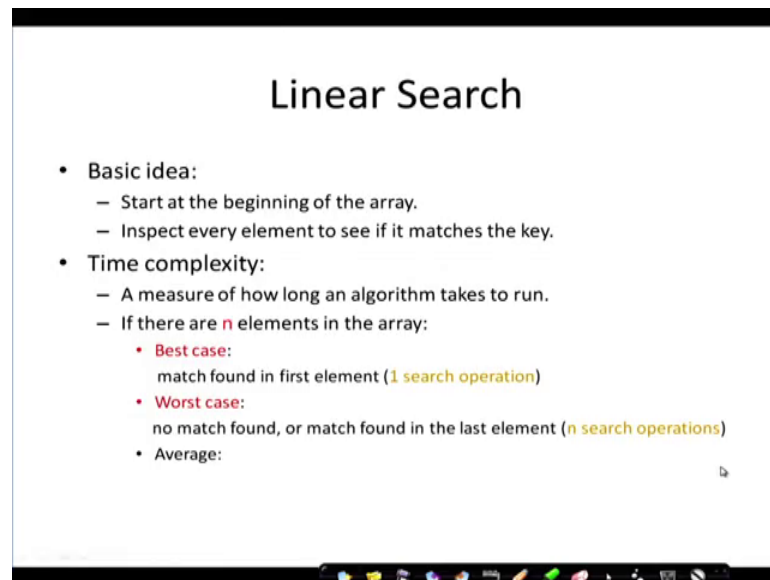
(Refer Slide Time: 15:22)



For example, think of this array where the elements are 5, 3, 2, 9, 8, 17, 6. Now this array is not in any increasing order, not in some decreasing order. It is decreasing here decreasing here. Again, increased here, decreased here, again increased here, decreased here. On the other hand, if the array was something like this; say, 2, 3, 5, 6, sorry, 2, 3, 5, 5, 6, 8, 9, 17. Then this array is sorted in an increasing order. That as I go from top to down it is in an increasing order, it is sorted in increasing order. It could be sorted, in the decreasing order also. Like, say for example, I start with 17, then 9, then 8, then 6, then 5, then 3, then 2.

So, this is as we go down the numbers are decreasing. So, these are ordered or sorted. And this is unsorted. These also sorted, but in a different order. So, I may have to search from an array that is either unsorted or sorted. So, first let us think of dealing with unsorted arrays, all right. And we will look into the linear search algorithm for doing that. And if the elements are sorted we have got a more efficient such algorithm called the binary search.

(Refer Slide Time: 17:46)



Linear Search

- Basic idea:
 - Start at the beginning of the array.
 - Inspect every element to see if it matches the key.
- Time complexity:
 - A measure of how long an algorithm takes to run.
 - If there are n elements in the array:
 - Best case:
match found in first element (1 search operation)
 - Worst case:
no match found, or match found in the last element (n search operations)
 - Average:

So, linear search we have already seen in an earlier lecture. So, the basic idea is we start at the beginning of the array. So, if we come here, if we come here, sorry, we start say I want to search this. I start at the beginning of the array. I start here. And I have got a key. A key is there, say key is 8. So, I start from the top.

Look for the availability of a look for the element matching the key, 5 is not matching 8, then I go on increasing this index, and go on comparing till I find either if 8 is not there, I will come to the end of the array all the elements have been checked I have not found the match, or when I get the match I will say that at this position, I have got the key matching, all right. So, we will start at the beginning of the adding, and inspect every element to see if it matches the key. Now if I want to do it in this way, we often talk about time complexity, it is a measure I am not going into the formal way of the measure, the measure of how long an algorithm takes to run.

So, as you can understand, if there are n elements in the array, then the best case would be let us you will find out what it will be say, I start with this. It may be supposed, the key was 5, suppose the key was 5. In that case, that is the best case with one match one comparison I find it how many comparisons how many inspections I needed only one. So, that is the best case, the worst case could be if my key was 6. If my key was 6, then I would have to inspect every element. And since there are 7 elements here, I had to carry out 7 comparisons, 7 inspections till I found the match right.

So, the best case would be first element one inspection one search operation. And the worst case would be no match found. Either the last element or not at all found even after that. In that case I need n search operations. So, the best case is one, the worst case is if the size of the array is n, then n. So, the average would be n plus 1 by 2 search operations, n plus 1 by 2 that is a cost how much time that gives a measure of the time the algorithm will take to run.

(Refer Slide Time: 21:23)

Contd.

```

int linear_search (int a[], int size, int key)
{
    int pos = 0;
    while ((pos < size) && (a[pos] != key))
        pos++;
    if (pos < size)
        return pos; /* Return the position of match */
    return -1; /* No match found */
}

```

So now we are trying to write it we had explained this algorithm also earlier. But now that we have done functions, let us write a function for linear search. Why am I writing the function? Say again if I go to this case. Now I may like to say my task is to find out whether I any student who has got more than 80, all right. So, I will need to write a function, I have to search this in a linear way in a sequential way one after another and for that I have to write a function.

So, how is that function, how will that function look like? Let us see what are the things I need I have got an array. So, I need to know which array I will be working with, and that is this array A. And what is the size of the array n whatever that is 7 maybe in our case the example that I was showing was 7. So, that is so, after I compare 7 elements, if I do not find the key, then I am unsuccessful in finding the search is not yielding any result. The other thing is key the element for example, 8 which I am finding out in this array of numbers, right. Say something like this I am trying to find out 8.

So now let us look at the algorithm. Int pos equal to 0 pos means position. So, initially I am in this position. That is this array is a so, a 0, while pos is less than size it is, right now 0 it is less than size; that means, the pos is not has not exceeded the last element; that means, I have not yet checked the last element. That is why this is said, and the this pos the element a pos a pos, pos is the index is 5, and suppose my key is 8, if a pos is not equal to key.

Then I will increment pos, I will check for the next element. Now look at this. This is an end here. If either of these conditions fail; that means, if I have not found it fine, but I have exceeded the key exceeded the array size, I mean limit then I will stop. Or if I have found the key 8 here, although I have not reached the end I will stop ok. So, that is the condition, then I go on increasing pos; if pos is less than n; that means, when I come out of this while loop, when any of these conditions are not satisfied.

So, if pos is less than n; that means, actually it should have been pos is less than sighs ok. If I have come, if still pos is let us it should be pos is less than size. If pos is less than size; that means, I have not come to the end of the array then; obviously, why did I come out? Because this condition was false this condition was false means what? I have already found the key, right then I will return the position.

So, pos increments from 0 to 1 to 2, to 3, then to 4, when I do that the size was 5, I have not exceeded 5 is still less than 5. And so, pos could be in 2, but I have come out I have come out because I found this. So, at pos pos position 4, I am getting the key. So, pos is returned, otherwise if this is not true, then I will return minus 1, minus 1. If it is returned; that means, that I have not found the key ok. So, this is the look of the linear search function, very simple. Now so, the key appears in if the key appears within this limit, then I will return the pointer pos otherwise I shall return the key.

(Refer Slide Time: 27:04)

Contd.

```
int x[] = {12, -3, 78, 67, 6, 50, 19, 10};
```

↑ ↑ ↑ ↑ pos = 4

- Trace the following calls:
 - `linear_search(x, 8, 6);` → Returns 4
 - `search(x, 8, 5);` → Returns -1

So, here is an example say, this is the array x, 12 minus 3, 78, 67, 6, 50 etcetera. I want to trace the following calls. I call search so, if you go up the linear search, let me call this is linear search. It should be written down as linear search. I am calling with the array x all right the size of the array 8, and the key is 6 ok. Here what will happen? Size of the array is 8.

So, I will start from here 12, no minus 3, no my key is 6 right, I increment pos from here to here, no, it is not matching pos is not the key, a pos is not the key here a pos the key. So, I come out with 0, 1, 2, 3, 4, pos value is 4 here. What will happen for this? I the same array x size 8, but my key is 5 you can see that I will go up to this and after that I will increment pos plus plus. So, it will be more than 8. And so, the size is 8. So, I will not be able to I will say that I have not got the key, all right.

So, that is so, this one will return 4, and this one will return minus 1. That is how the linear search algorithm works. Now let us stop here for today. Next, we will look at another more efficient search, but it require that demands something more from us that the array must be sorted in that case we can apply the binary search algorithm to make it more efficient ok. So, today what we saw is how we can write as small small functions to solve a big problem after a break down that big problem into smaller parts. And also, we found how we can the concept of linear search which we learnt earlier, how we can write a function for that ok.