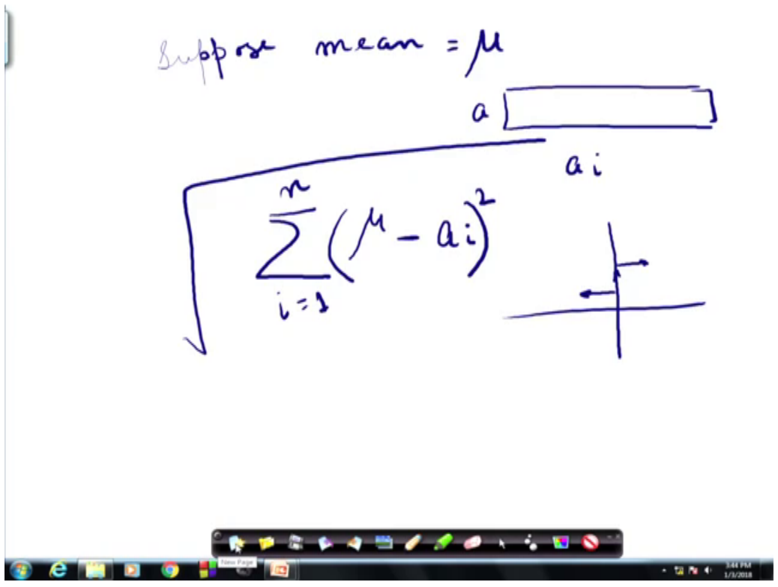
Problem Solving through Programming in C

Prof. Anupam Basu
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Lecture – 31
Linear Search

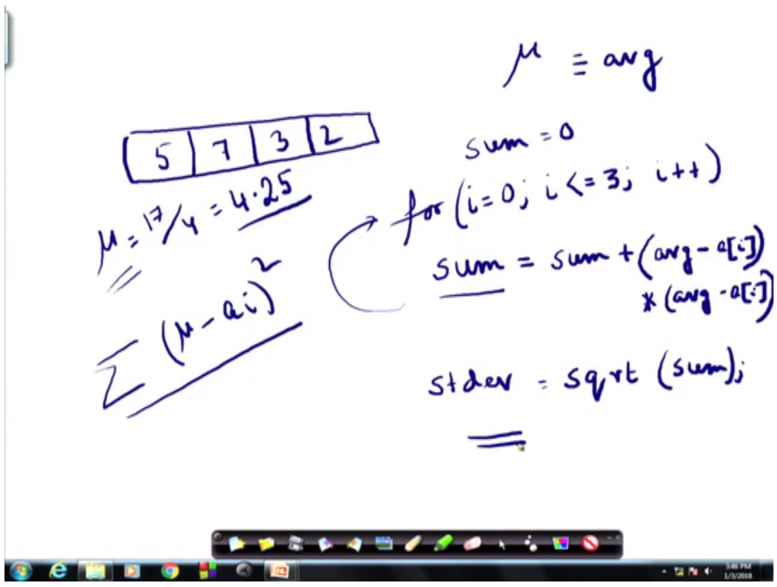
So, we were looking at computing standard deviation. So, standard deviation is mean; we have seen.

(Refer Slide Time: 00:28)



Now, standard deviation is if the mean is suppose the mean is represented by mu then I take the sum of the deviation from the mean for every element of the array. So, the array was a with every element being called a i.

(Refer Slide Time: 01:56)



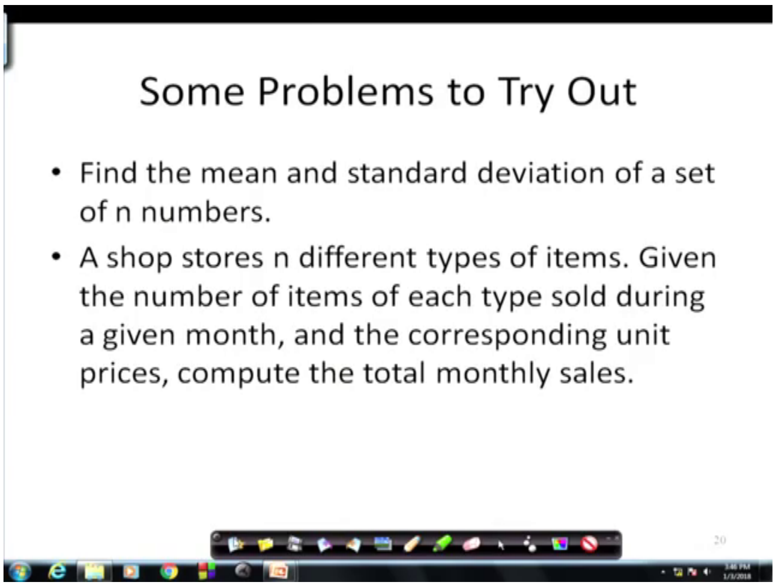
```
sum = 0
for (i=0; i<=3; i++)
sum = sum + (avg - a[i])
            * (avg - a[i])

stdev = sqrt (sum);
```

So, what I am trying to do is, I am taking the difference of a i from the mean and since its difference can be suppose there are some values and this is the mean and the value can be a little away from the mean on this side more than the mean or less than the mean. So, we take the square of the mean square of the difference and I do that for all the elements I equals 1 to n right and then I take the square root of the whole thing. That is my standard deviation other my variance. So, I can compute either this or this, whatever I like.

So, you will simply understand that in the code that we have given little earlier then there we had computed the mean; that means, mu has been computed which was the average that we computed last time. So, if I have read the elements in an array whatever the elements are 5 7 3 2 and I have computed the mean mu who has been computed mean is of 12 3 15; 17 divided by 4. So, it is 4 point something 4.1, 4.25 right that is my mean. So, now, for every element again in this array I find out for i assign 0, i less than equal to 3, i plus plus what do I do? For every element I have got the mu and let us call it average right.

So, sum was 0 what is it look like, sum was 0, sum will be now sum plus I am getting sigma of mu minus a i whole square. So, sum plus average minus a i times average minus a i, all right. So, that is the square average is mu minus a i that is. So, I am doing this square and I am repeatedly doing this and getting the new sum and at the end of this loop therefore, I have got this. So, I can see std, it is a dev let me call it standard deviation is square root of the sum all right, in that way I can find out the standard deviation also.

(Refer Slide Time: 04:24)



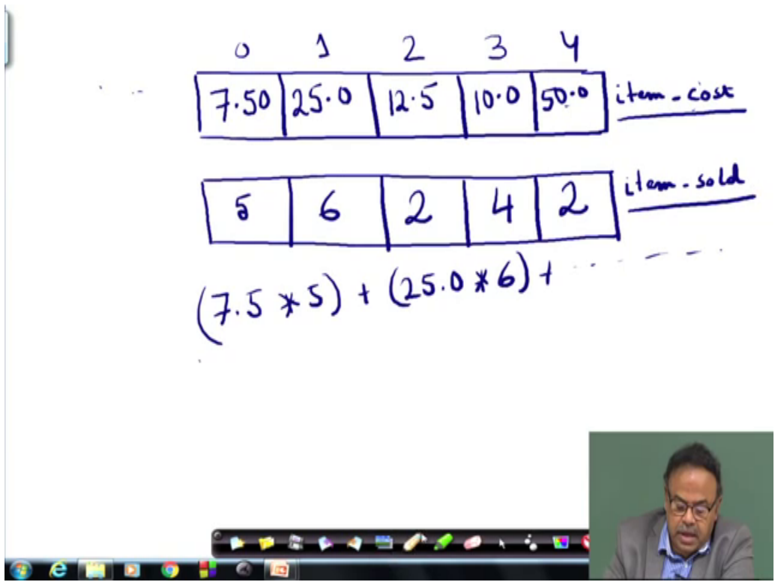
Some Problems to Try Out

- Find the mean and standard deviation of a set of n numbers.
- A shop stores n different types of items. Given the number of items of each type sold during a given month, and the corresponding unit prices, compute the total monthly sales.

Now, let us come to the now what is the application what is the meaning of this. So, with these say in a class you are you are supposed to write a program where you want to find out say in a class of physics. What is the average of the numbers of all the students? So, I will find out the mean and mean of the class marks right. Similarly I can find out that what is the standard deviation, how much did it vary, that also I can find out.

Now, let us look at this new another problem. Say a shop stores in different types of items, n different types of items. Now, given the number of items of each type sold during a given month and the corresponding unit price is compute the total monthly sell. So, what is the scenario? The scenario is this.

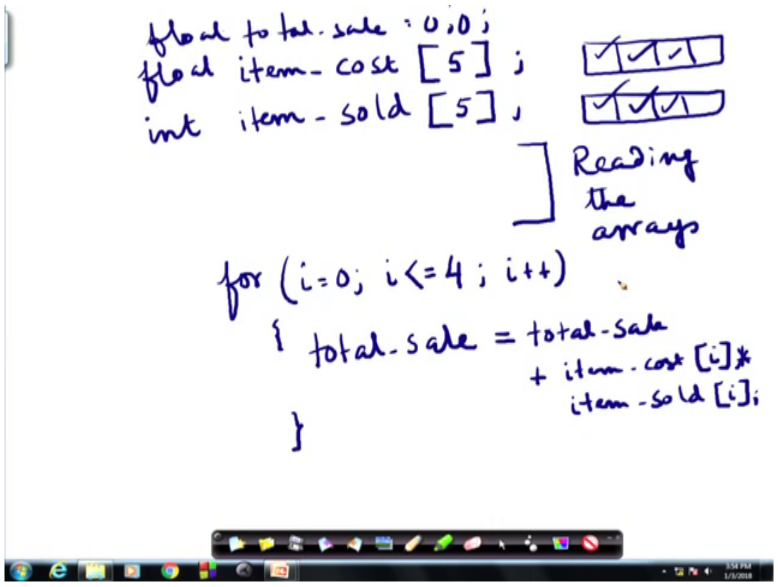
(Refer Slide Time: 05:24)



I have got say 5 items, item 1, item 2, item 3, item 4 and item 5. And let me call it the item, let me call it on this side let me call it this array is item price, item cost, item cost. Suppose the cost here is 7 and half rupees per item of type 0, 25 for item of type 1, this is 0, this is 1, this is 2, this is 3, this is 4. So, the item cost for this for item of type 2 is 12.5, item of type 3 is 10, item of this is item of type 4 is 50 rupees.

Now, I want to store how many items of each type has been sold. So, I take another array and call it items sold. Suppose 5 items have been sold of type 0, 6 items here of type 1, 2 items of type 2, 4 items of type 3 and 2 items of type 1. Now, my question is what is the total sell? So, what should I do? You can easily understand that the here is the item cost. So, I have to multiply this with this, this with this and add all these costs. So, it will be 7.5 times 5 it will be 7.5 times 5 plus 25 times 6 plus so and so forth in that way I can find the total cost. So, what will the program look like? The program will look like I will be needing to arrays item cost and items sold and they should be of the same size, assuming that I know beforehand that there are 5 items.

(Refer Slide Time: 08:46)



```
float total.sale = 0.0;
float item-cost [5];
int item-sold [5];
                         ] Reading
                           the
                           arrays
    for (i=0; i<=4; i++)
      { total-sale = total-sale
                     + item-cost [i]*
                       item-sold [i];
      }
```

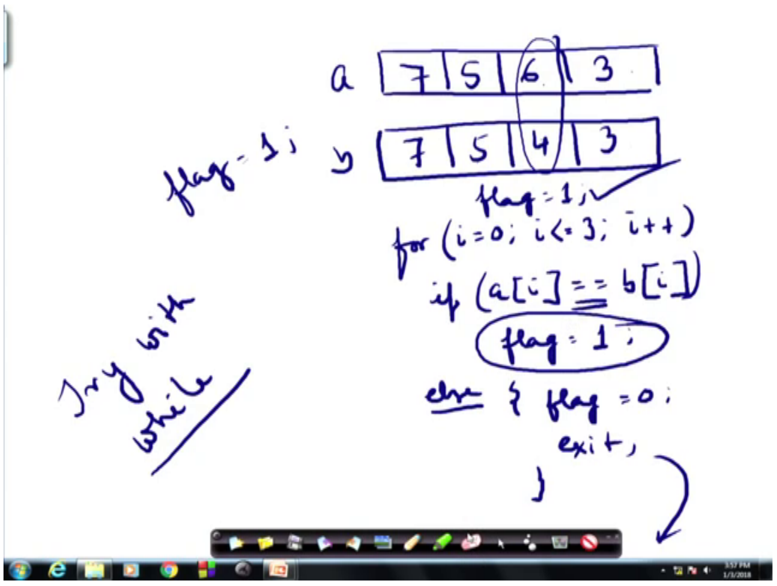
So, I can start with item cost of type 5, but this will be cost will be of type float and there will be another the number of items is, item sold the number of items sold is integer. So, I will have this, I am not showing the part that here I am reading the two arrays. So, after I read the arrays I will have two arrays like this, one is an integer array another is a floating point array this is the float and this is an integer.

Now, my actual body of the program will be in a loop for say I am doing it using for say i assign 0, i less than equal to 4 since the size is 5, i plus plus. And what do I do here? What do I do in the body of the for loop? I take ok, I write actually only one statement will do there is no harm in putting this bracket. Total sale which was a variable of type float total sale is, total sale was initialized to 0, total sale plus item cost i times multiplied by item sold i. So, this will be done in a loop.

And so I will take the first item, item cost 0 multiply with that with item cost item sold 0, add that with the total sale which was initialized to 0. So, here I can have float total sale initialized to 0, 0 point 0 all right, I can do that. So, now, I am doing it in a loop. So, first I multiply these two add it to total sales, next again in the next iteration i is implemented I take these two and multiply them and add it to the total sale then I do this two and multiply them add it to the total sale and I go on in this way. This is, this is in this way by using this array I will be able to add all these values and I will get the total sale at the end.

So, here we could see two very nice examples of application of arrays. Now, one another problem that I gave you I mean as I was while comparing the arrays you can do it in multiple ways that they were as two arrays you must have solved it by now, that they were two arrays like this 7, 7, 5, 5, here 6, here 4, but again 3, 3.

(Refer Slide Time: 12:24)

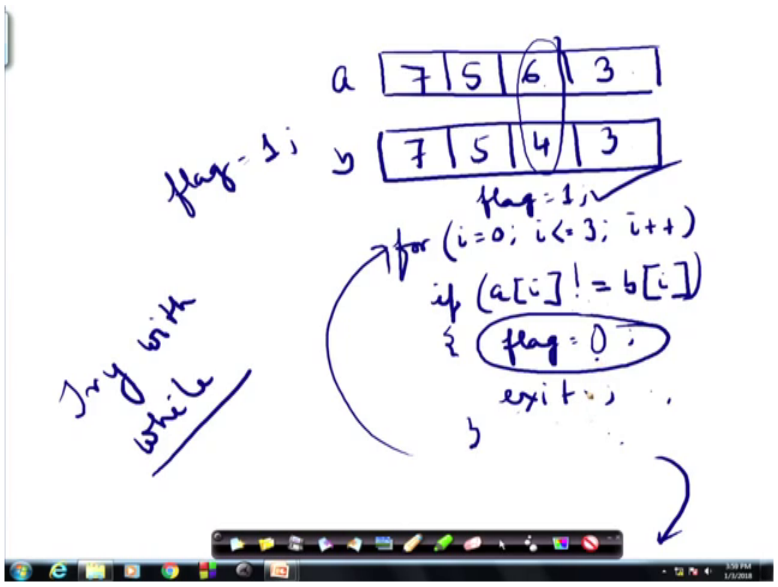


```
flag = 1;
for (i=0; i<=3; i++)
    if (a[i] == b[i])
        flag = 1;
    else { flag = 0;
        exit;
    }
```

So, everywhere they are matching, but intermediate they were affected and I had a flag value so flag was initialized to 1. And then in a for loop then in the for loop I was checking this was a and this was b, flag was initialized to 1, if a i is equal to b i this I can do because here I am doing it element wise then flag equal to 1 else flag equal to 0 and I can do exit and I come out of the loop straightway, I come out of the loop. Because it does not really matter at which position the mismatch occurs, as soon as the mismatch occurs I can say that flag is 0 and so I come out. So, when I was comparing this whenever I find a mismatch the flag will become 0 and it will not be reset to 1 again because of this match, because this part is not being computed I need not compute it. I need not compute it because my objective was to see if the two arrays are equal and here the violation has already occurred. So, they are not equal.

However so, that is one way you can try it with while loop also. You can try with while to solve the same problem. Another point is here the time and again here, here, everywhere I am setting the flag to 1, I could have changed that also here if what did I need to do I go on I haves in take flag to 1 and the condition I simply change I just change the condition if a i is not equal to b i, then make flag 0 exit.

(Refer Slide Time: 15:20)



```
flag = 1;
for (i=0; i<=3; i++)
  if (a[i] != b[i])
  { flag = 0;
    exit();
  }
```

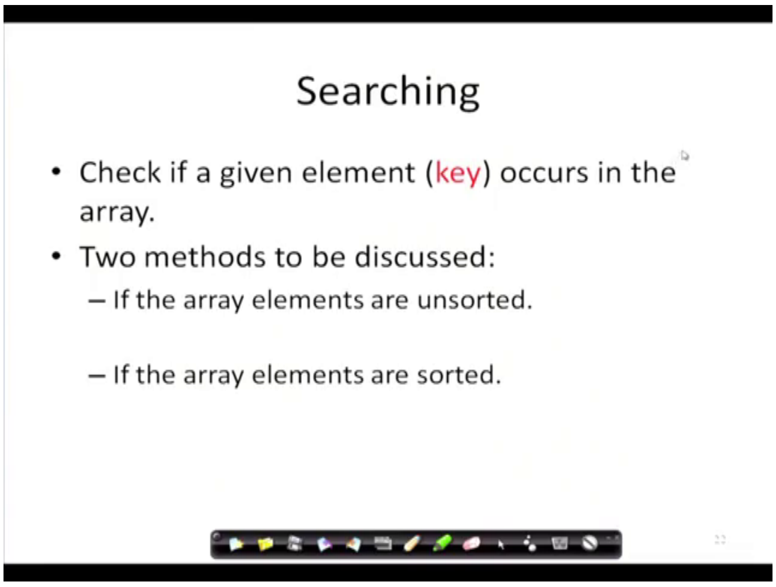
a	7	5	6	3
b	7	5	4	3

I could have done this. As long as this condition is not holding I am going on doing the loop is it clear, I will go on doing the loop as long as there is no mismatch, this condition means mismatch. As soon as there is a mismatch I will set the flag to 0 and exit. There is another way of solving the problem. So, you have to think logically what exactly you need to do and what exactly you are writing what is the flow and what is happening with the variables and I always suggest that you have small pictures of the different variables and see how they are changing in the course of running the program.

Now so, we have seen a useful commercial. So, called toy commercial problem that how I can find out the cost of a total sales or monthly sales. So, and here the number of items sold per month are given then you can do it.

Well, next let us look at a very important thing called searching. Searching is a fundamental task in any and in fact, in many computations. In many computations we need to search. What do we search? There are different types of searches, but we will be now talking about the simplest possible search that is we are trying to find out whether a particular element is there in an array right.

(Refer Slide Time: 17:24)

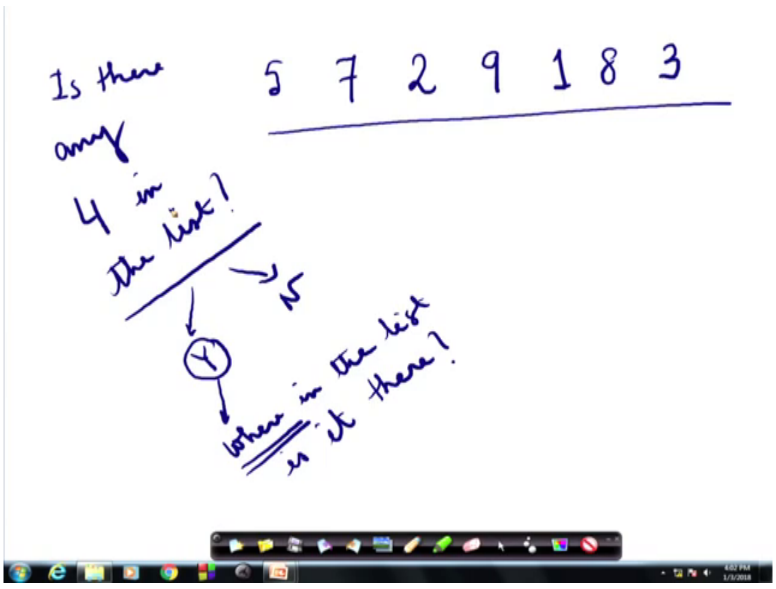


Searching

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

Say, so the purpose is to check if a given element which is known as the key is there in the array or not. We will first talk about the array is not arranged in any order and we will do that.

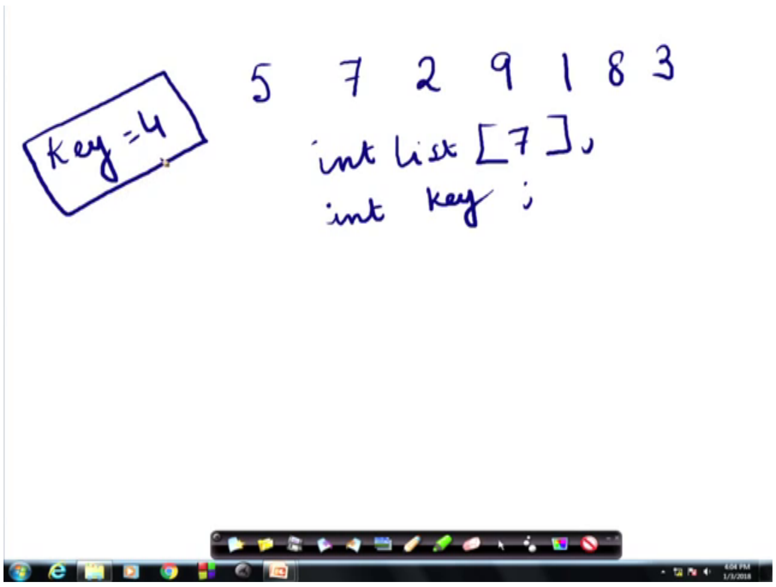
(Refer Slide Time: 17:56)



So, suppose I ask the question that is there any even number in the set of numbers given, suppose there are some numbers given 5 7 2 9 1 8 3 like that. I want to see and suppose this is a huge this is a list of hundred numbers. I want to find out whether there is any particular number forget about even number, for the time being for the timing let us assume that I want to see whether in this list any 4 is there, is there any 4 in the list that is the question that we are asking. The answer can be either yes or no.

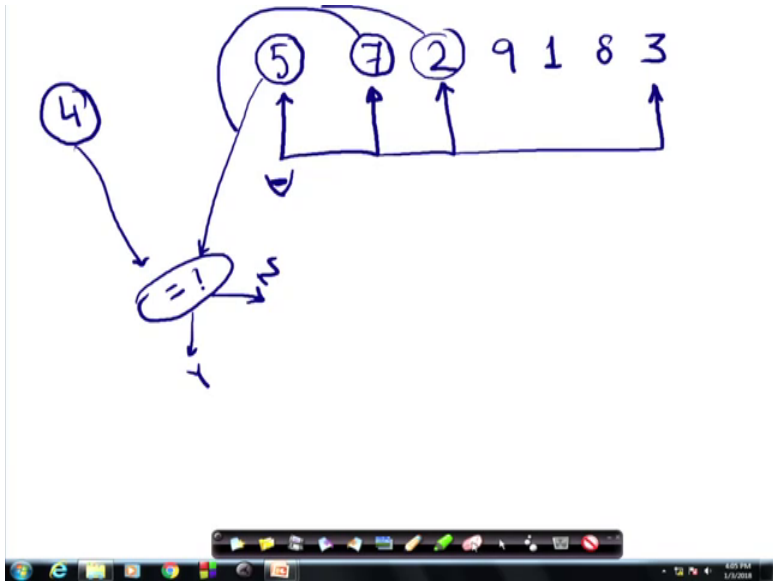
If it is yes then the next question comes where in the list is it there, is it where means in which position it is there. I may ask like to know the position or I may not like to know the position I would be satisfied to know whether this list contains any 4 or not all right; so 5 7 2 9 1 8 3.

(Refer Slide Time: 19:28)



So, again 5 7 2 9 1 8 3 and my key is 4 because I am interested in the existence of 4. So, instead of writing the c which you will be writing I will be discussing how to approach this problem what would the pseudo code be and I am sure in the assignments in your practice you can write the program. So, I know I need to know beforehand this list. So, so I need to know a list which may be an integer list of might be here 7 numbers, I need to know that also I need to know which key I am searching for.

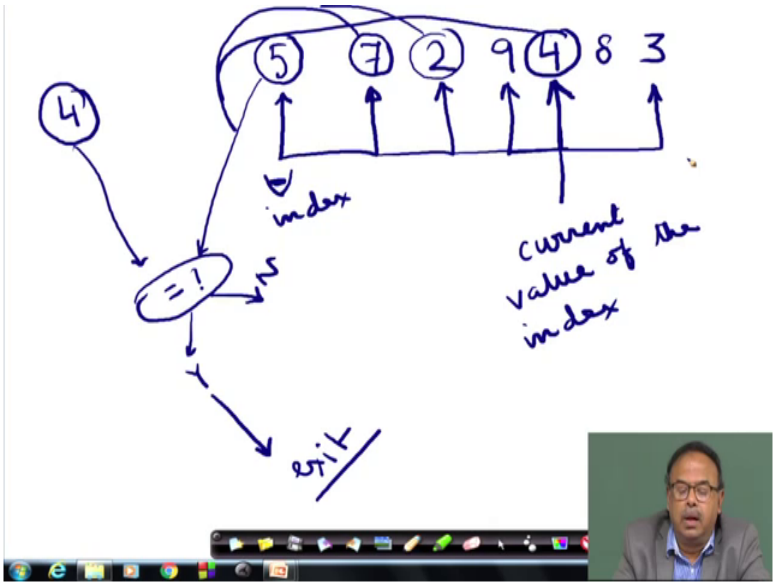
(Refer Slide Time: 20:46)



Once I know these two, I know 5 7 2 9 1 8 3 is my list and I know that 4 is my key then how should I go about it.

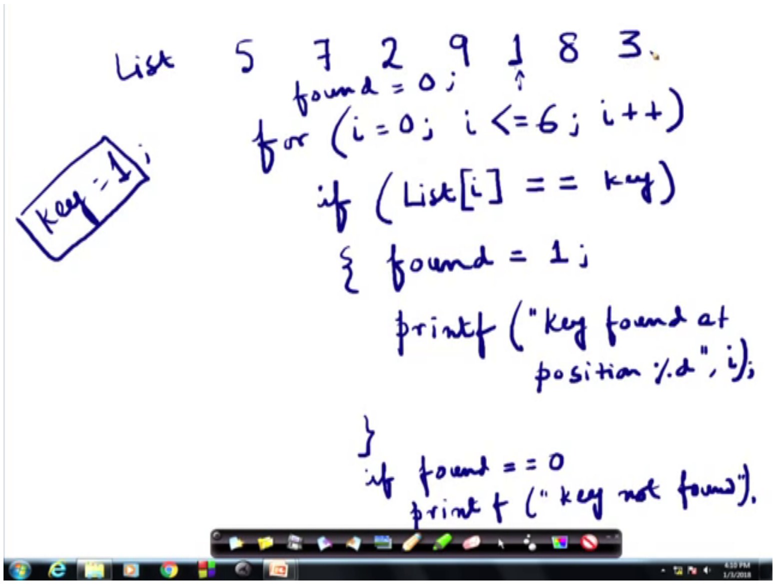
I have got 4 in mind. I start to look at as if I am looking at through some means at different positions I look at this position and check this element and compare this element with the key and I ask are they equal. If the answer is yes, then obviously I can say that 4 is in the list. But as you can see it is not true, so in case of no I will shift my focus from here to here and I will now compare with 4 this element 7, are they equal, no then I will again shift the focus and I will go on shift the focus and I will compare with this element with the key all right. In that I will go on. How long shall I go on till the end of the list.

(Refer Slide Time: 22:27)



If suppose here there was a 4, instead of 1 there was a 4 here, suppose instead of 1 there was a 4 here. Then when my focus changes to this point and then again comes to this point and then I find that this element is matching the key value, then I can exit and say yes, 4 is there in the list. Then if there is other question has to be answered that 4 is there, but where is 4, where is 4, in that case what would be an answer be. Your answer would be this position. And what is this position? This position is nothing, but the current index current value of the index. Here the index started this is my index which was shifting.

(Refer Slide Time: 23:48)

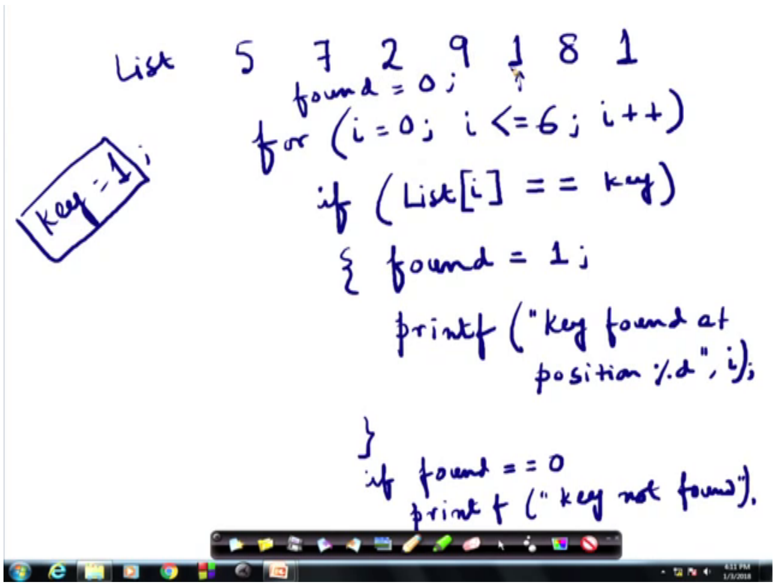


```
List   5  7  2  9  1  8  3.

key = 1;
      found = 0;
      for (i = 0; i <= 6; i++)
         if (List[i] == key)
         { found = 1;
             printf("key found at
                     position %d", i);
         }
         if found == 0
           printf("key not found").
```

So, now we can think of the algorithm I have got 5 7 2 9 1 8 3 and my key is 4. So, I will be doing. So, this is a loop which is a list for i equals 0; that means, the focus or the index i less than equal to 1 2 3 4 5 6 7 6, i plus plus, i plus plus. If least i, that means, the ith element of the list is equal to the key then I can say found assigned 1. And what is found? Found is some variable which I have initialized at this point initially nothing is found it is not found. So, found is 0 initially I have not found the key here I am comparing as soon as I compare I put found equal to 1 and then I can exit or this automatically this loop will go on.

Now, if I do it in this way what is the problem? Suppose my key was, suppose my key was 1 then it goes on i 0, this is never happening found is still 0, it goes on it comes to here and I count come to found equal to 1 and then I can say if found equal to 1, I can printf, printf key found at position percentage d i. So, at that point I can also print that it has been found here right. In the worst case what can happen? Found will remain 0, found will remain 0 and I will come to the end of this point when I come here I can check if found is 0, printf key not found; if I do not find it. There can be different flavors of the same problem.

(Refer Slide Time: 27:46)



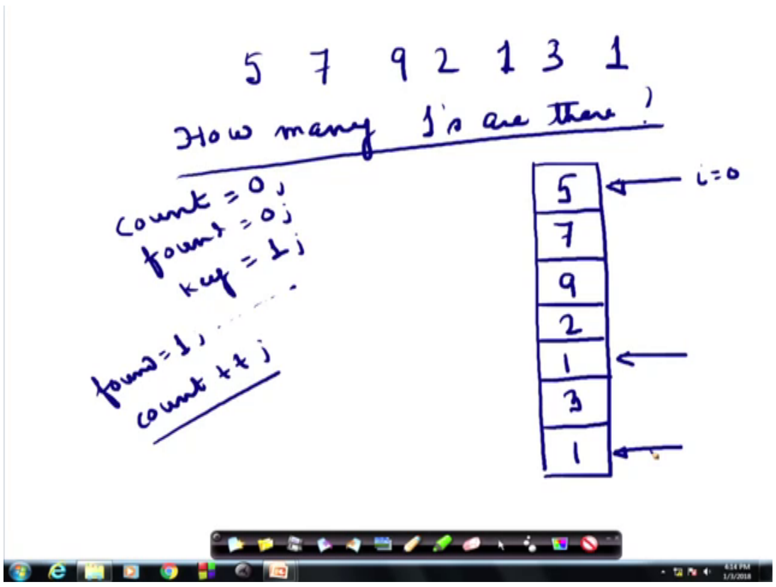
```
List   5   7   2   9   1   8   1
          found = 0;
      for (i = 0;  i <= 6;  i++)
           if (List[i] == key)
           { found = 1;
               printf ("key found at
                        position %d", i);

           }
            if found == 0
             printf ("key not found");
```

key = 1;

The other flavor could be that suppose this element 1 is there at multiple places suppose it is also here. Then what will this program result in? What would be its output? I will go on checking here I will check the for loop is extending up to which part; the for loop is extending up to this, this position and this is separate I should not have I should have written it on the on this side.

So, I go on checking this. So, key is found at position number 0 1 2 3 4, key is found at position 4. The list is not exhausted ultimately it will come to this point and when it and again it will say the key is found at position 6, twice it will it is found it will be told like that keys it will print twice. If at the end it comes and still the key the value of the variable found is 0 then it will say print is not found.

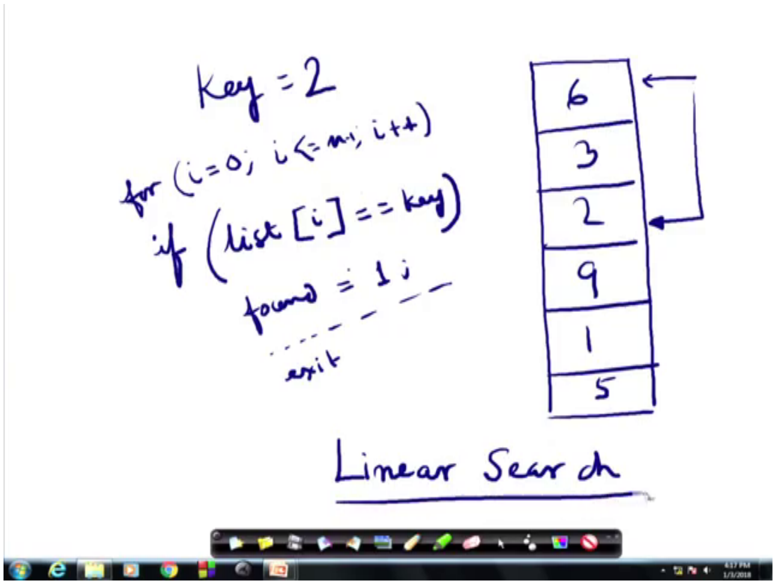
(Refer Slide Time: 29:05)



I could have also say the same thing same problem I do, say suppose I have got a list 5 7 9 2 1 3 1 the question is how many ones are there. The same algorithm will do, the same algorithm will work right. What is the same the algorithm? The algorithm is I start from one point from this beginning I let me draw it in this way if this be an array where all my elements are there, 5 7 9 2 1 3 1 I start from the beginning i equals 0 and for every element I compare with the key and go on till the end.

Now, if I want to do this what is the additional variable that I will require? I will require another variable count which is initially 0 other than found which is also false. So, whenever I find a 1 and my key is 1. So, whenever I find a 1 I will say found is equal to 1 and also I will do count plus plus and I will continue. Here I come and I will find I will have the value of count to be 2. So, I can also, so I can say that here I could print where found is becoming 1. So, I can also, I can say at which position it is a found and how many times it is found.

(Refer Slide Time: 31:27)



Now, it can another flavor could be that I have got this say this array; I have got this array and some array 6 3 2 9 1 5 and whenever I have been given a key say the key is 2, as soon as I find 2 that is enough for me. I just want to know whether 2 is there in the list, I am not interested to know how many times it is there or in which position it is there or at best I may like to know at which position I you found it first. So, what I can do, I will go on searching like this and whenever you find 2 then you print that I have found 2 here and exit.

So, what will you do? The loop will be, the loop inside the loop you will have you can do it like this that if a i or list i, let us I was writing list i is not equal to key all right for you could have done it by while also n minus 1 i plus plus you go on doing this. If list i is not is sorry is if it is equal to the key what I have done is equal to key say found equal to 1 and printf the position and exit and you need not go through the entire loop. Now, there are, so that is this sort of search which I am doing in a linear way from one side to the other is known as linear search. It is the simple, very simple search for a particular element. We will see a little bit more on this in the next lecture.