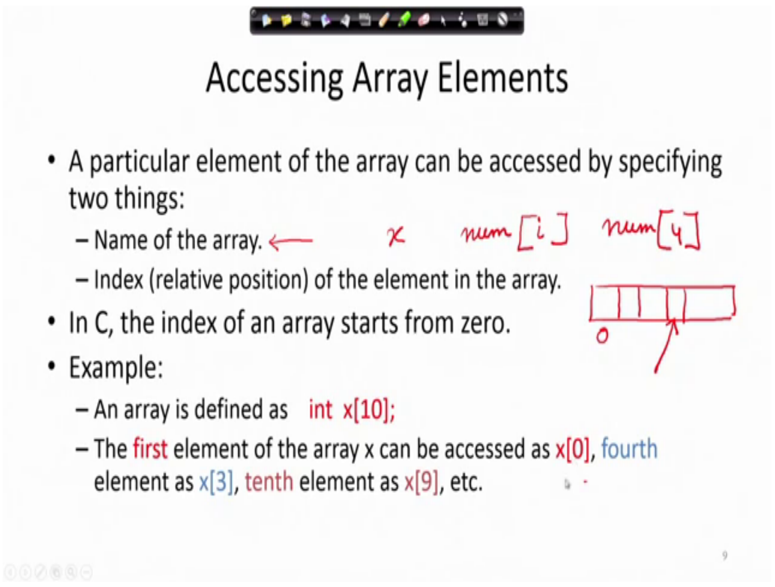
Problem Solving through Programming in C

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

Lecture – 28
Arrays (Contd.)

So, we now know that an array is identified by a name of the array. Some name like x, num, whatever and there is an index num i or say num 4, num 5 etcetera. So, by this index, by this index we identify which particular element of the array I am referring to.

(Refer Slide Time: 00:24)



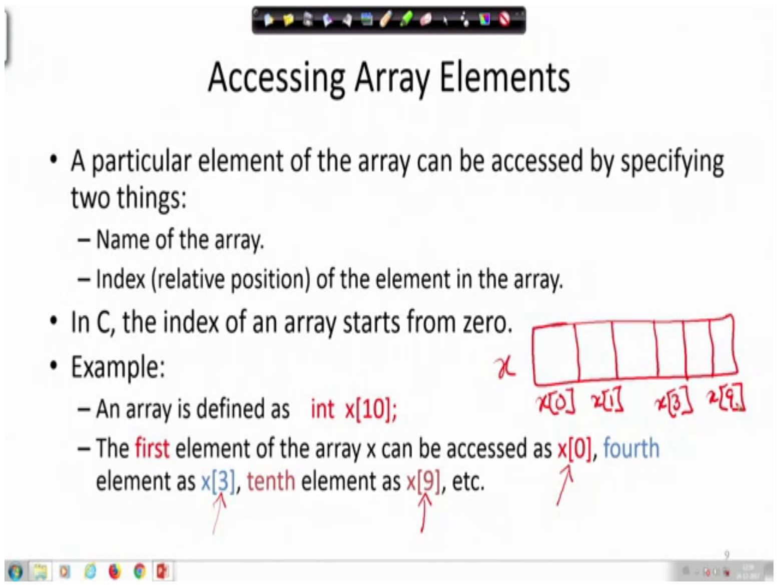
Accessing Array Elements

- A particular element of the array can be accessed by specifying two things:
  - Name of the array. ← x $num[i]$ $num[4]$
  - Index (relative position) of the element in the array.
- In C, the index of an array starts from zero.
- Example:
  - An array is defined as int x[10];
  - The first element of the array x can be accessed as x[0], fourth element as x[3], tenth element as x[9], etc.

Now, in c also you have said that the array starts from 0, the index of, the value of the index starts from 0 ok.

Next. So, when we have an area defined as int x 10, the index are from 0 x, x 0 to x 9 all right. The first element can be accessed as x 0 first element can be accessed as x 0 and the fourth element as x 3, the tenth element as x 9.

(Refer Slide Time: 01:40)



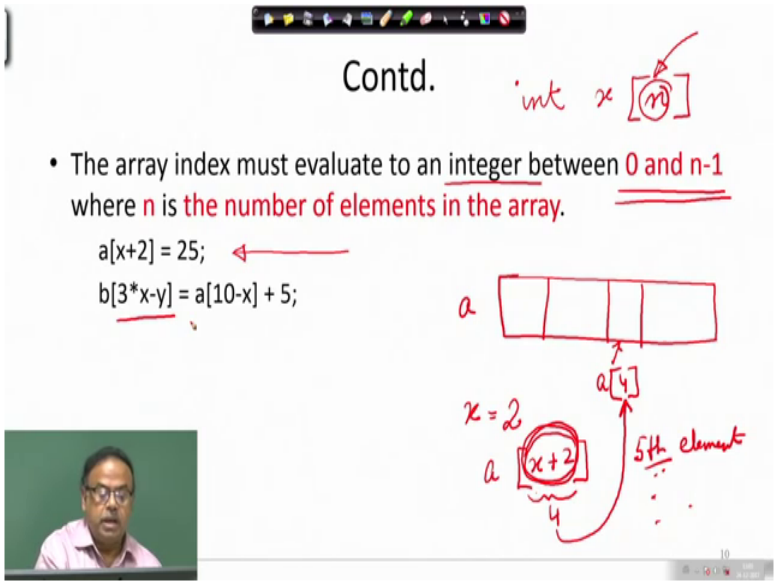
Accessing Array Elements

- A particular element of the array can be accessed by specifying two things:
  - Name of the array.
  - Index (relative position) of the element in the array.
- In C, the index of an array starts from zero.
- Example:
  - An array is defined as int x[10];
  - The first element of the array x can be accessed as x[0], fourth element as x[3], tenth element as x[9], etc.

I hope this is clear to you by now that this one is arrays x. So, this is x 0, this is x 1, the fourth element will be 3 and the last element will be x 9.

So, that is how the array elements are addressed, how they are referred to by the indices.

(Refer Slide Time: 02:14)



Contd.

- The array index must evaluate to an integer between 0 and n-1 where n is the number of elements in the array.

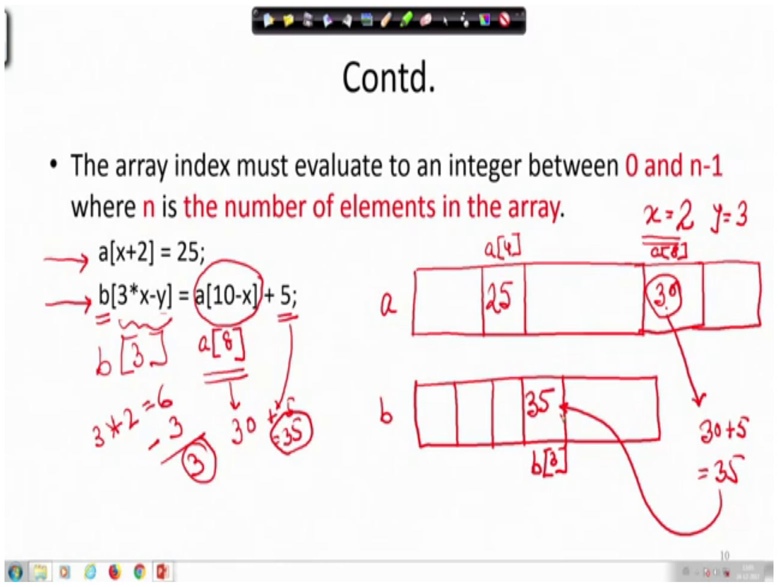
```
a[x+2] = 25;
b[3*x-y] = a[10-x] + 5;
```

Now, these indices must be integers. Now, the (Refer Time: 02:16) indices must be integers that is very important. So, if I have got an array defined as int x n, where n is the size of the array then the value of this n this index value of this index must lie between 0 to n minus 1. So, if it were twenty then the value must lie between 0 to 19 where n is the number of elements in the array.

Now, you can see here in this example that it is not the case that always I have to put a fixed integer as the index of course, I can write for an array, say an array is a I can always refer to some element of the array as say a 4 for the fifth element or I can have some other integer value x and suppose x is 2 now and if I refer I can refer to the same thing as a x plus 2; that means, this will also evaluate to 4 and this will also point to the fifth element of the array. Now, one thing that we have to be careful about that this value that is computed lies between in this range between 0 to n minus 1.

So, here is another example where the array index value has been computed through an expression 3 times x minus y is assigned. So, suppose here you can see that we are talking of two variable, two array variables.

(Refer Slide Time: 04:25)



Contd.

- The array index must evaluate to an integer between 0 and n-1 where n is the number of elements in the array.

```
a[x+2] = 25;
b[3*x-y] = a[10-x] + 5;
```

So, let us draw a picture. A picture always clarifies the things much better. So, let us have an array a and array b, a and b are two arrays right.

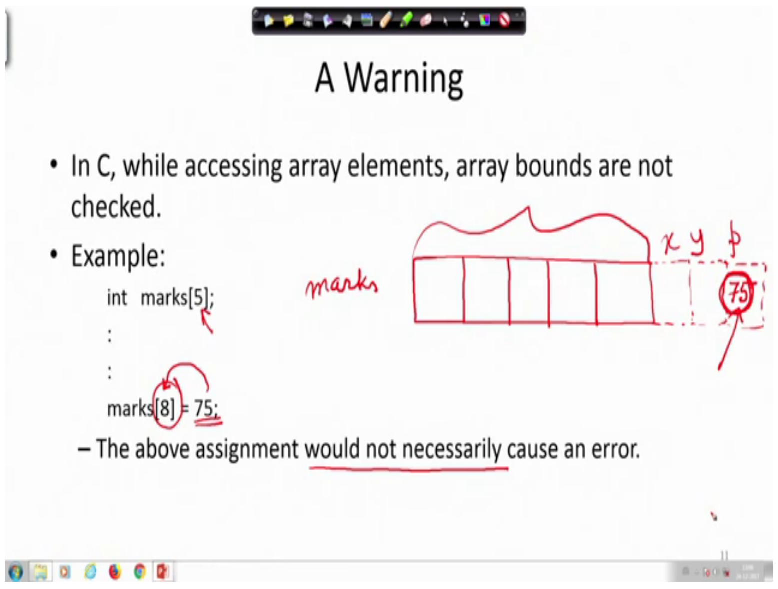
So, suppose x is 2 and y is 3 then this statement what does it do? A assigned x plus a x plus 2, x is 2 plus 2; that means, a 4 right. So, a 4 the fifth element is being assigned the value 25. And what is being done here? x is 2 a 10 minus x. So, that is a 8; that means, the ninth element, let us have the fifth let me make it a little bigger suppose this is the suppose this is the ninth element a 9, a 8 sorry a 8 this is 8. So, that was suppose 30 that is being taken plus 5 is being added to that. So, 30 plus 5 that is 35.

Now, note this, this part is being evaluated. What is being done in evaluating this part? I have got x to be 2. So, I am computing 10 minus x I get that to be 8; that means, I want a particular variable which is stored in stored as a 8; that means, in the nineth location of the array a and that is 30. I take that value 30 and add 5 to that, so I get 35. And then what do I do? I store it in the array b and wearing in the array b.

Now, I will compute this part 3 times x, x was 2. So, 3 times 2 is 6 minus y, y was 3. So, it will be 3. So, in b 3, b 0, b 1, b 2, b 3; that means, again in the 4th element of this array b 3 I will be storing the value 35. So, now, the array b will have the value 35 here. I hope this is clear. So, you have to be very clear about the distinction between the array index and the value of the array.

So, this 30 that I got after computing the index a 8 that was telling me where in the array I should look at to get that value. And then that was just a value and with that value I had another value 5 I added them together, I added them together and got the value 35. Now, I see where I am going to store that for that again I compute the index here and I compute the index the index must be within the range 0 to n minus 1, I compute that and I find that the index is 3; that means, in b 3 I have to store the value and go to b 3 and store the value. I hope this is clear right.

(Refer Slide Time: 08:14)



A Warning

- In C, while accessing array elements, array bounds are not checked.
- Example:

```
int  marks[5];
:
:
marks[8] = 75;
```

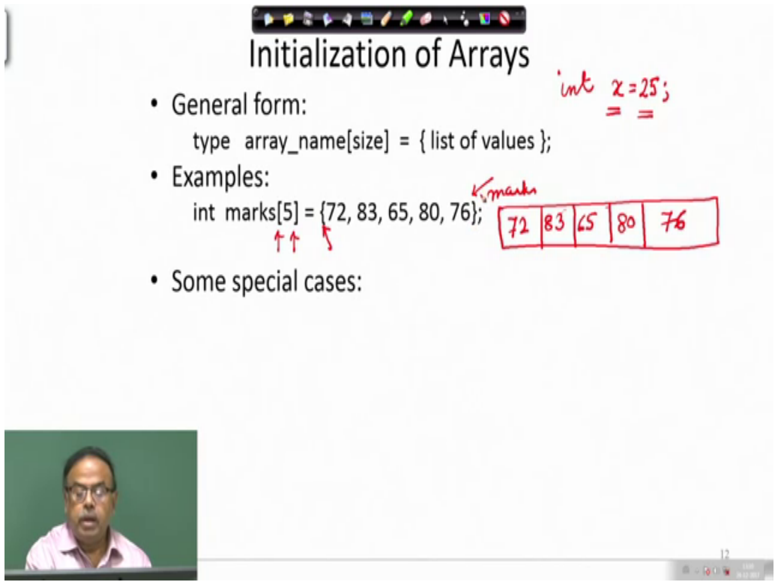
  - The above assignment would not necessarily cause an error.

Now, in C, there is a word of caution here that in the language C the array bounds are not checked for example, int I have got declared int marks 5; that means, I have got an array marks whose size is 5, a dimension is 5; that means, at least I can have 5 elements in that all right. Now, here what I am trying to do, here what I am trying to do is I am trying to write 75 in marks 8, but there is no space for marks 8, it was 5 6 7 8 this is the place where I am going to trying to write 75.

Now, it would be good if the compiler could give me any error that hey you had declared it that size to be 5 and you are going beyond the border all right, you are crossing the border. So, be careful warning syntax error or whatever syntax error and you are not allowed to do that, but unfortunately array bounds are often not checked. So, what can happen if you are not careful, then it will not necessarily cause an error, but will be writing something here all right, it will be writing something at this point which might be when I allocated the compiler allocated the memory this part was for the marks and this part were for other variables x y p and whatever value of p was there that is overwritten with 75. So, one must be very careful about this when writing the programs and running it.

Suppose it is a very common thing that often you will find that there are some funny errors occurring and the reason for that maybe you have crossed the array boundary.

(Refer Slide Time: 10:43)



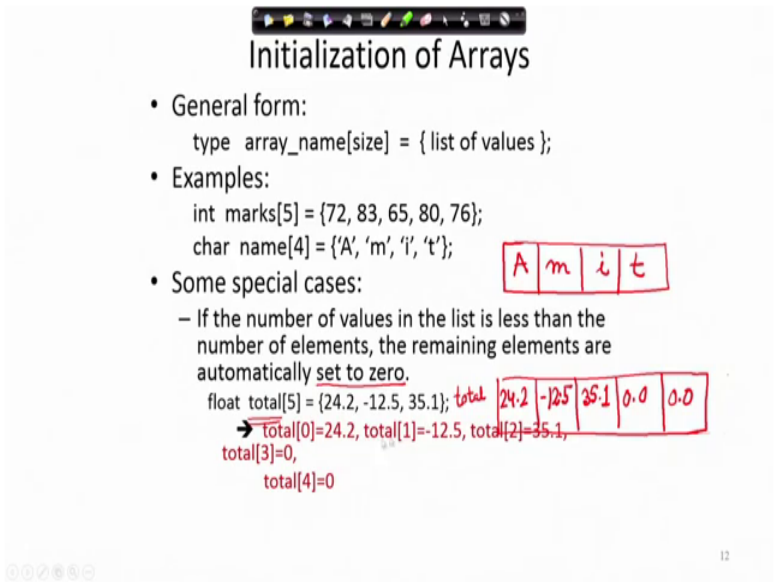
Initialization of Arrays

- General form:
  type array_name[size] = { list of values };
- Examples:
  int marks[5] = {72, 83, 65, 80, 76};
- Some special cases:

So, it can result in unpredictable results. So, general form of initialization of arrays whereas, the array name size with some list of values how do we. So, recall that we had done we can do initialization slight int x equal to 25 semi code, where I declare x to be an integer and also initialize it to a particular value.

The same thing can be done for arrays for example, I can write int marks 5 72, 83, 65, 80 16. Note that this initialization is within this two curly braces. That means, here I am creating an array whose name is marks and the values are 72 for mark 0, 83 for marks 1, 65 for marks 2, 80 for marks 3 and 76 for marks 4, 0 1 2 3 4 5 elements. I can declare in this way this is one form of initialization. Remember that here we are putting square brackets for declaring the dimension, but here we are putting curly braces.

(Refer Slide Time: 12:36)



Initialization of Arrays

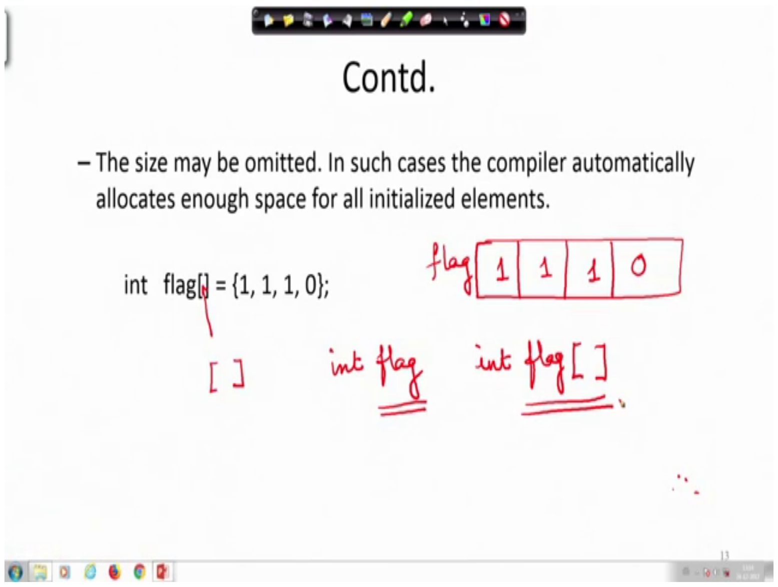
- General form:
  type array_name[size] = { list of values };
- Examples:
  int marks[5] = {72, 83, 65, 80, 76};
  char name[4] = {'A', 'm', 'i', 't'};
- Some special cases:
  - If the number of values in the list is less than the number of elements, the remaining elements are automatically set to zero.
    float total[5] = {24.2, -12.5, 35.1};
    → total[0]=24.2, total[1]=-12.5, total[2]=35.1, total[3]=0, total[4]=0

Similarly, this one char name 4 I have just typed in an array, so the array has got 4 elements and what is stored here A m i t, Amit. So, basically the name is Amit. So, that is coming as a string of 4 characters. Although I use the word string be careful about that because string has got a little more thing to it which I will describe later, but right now it is an array of characters, the characters array of characters of size 4.

Now, there are some special cases the number of the, if the number of values in the list is less than the number of elements the remaining elements are automatically set to 0 are automatically set to 0. For example, in this case I have got a variable what is the name of the array here the name of that is total and what type of array is it, it is an array of floating point numbers and so there are 5 elements, 1, 2, I have got 5 spaces, 2 3 4 and 5. There are 5 spaces yeah and I have loaded initialize it to this value.

So, here it is 24.2, the next one is minus 12.5, the next one is 35.1. The remaining elements which are not filled up are filled up with 0s automatically by the compiler if it is less. Now, be careful that sometimes some compilers behave in a are implemented without doing this bit and therefore, you must check whether your compiler usually the standard compilers do that, but you should be careful about it.

(Refer Slide Time: 15:05)



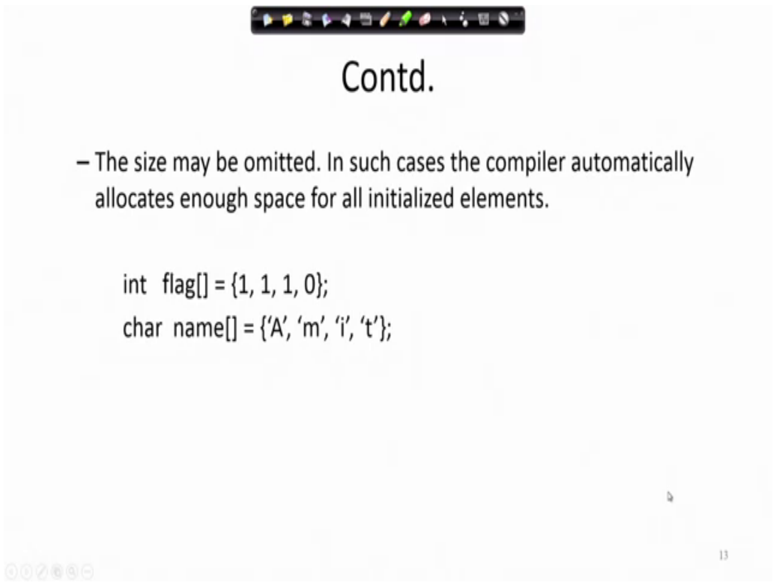
Contd.

- The size may be omitted. In such cases the compiler automatically allocates enough space for all initialized elements.

```
int  flag[] = {1, 1, 1, 0};
```

So, total will be this as I have said here. The size may be omitted in such cases the compiler automatically allocates enough space for the initialize variables. So, when I am initializing it then it is possible that I can write I can leave out this size because I am already writing it here in this form therefore, in this case what will happen flag and array will be created just for the initialized values. So, a 4 element space will be given to you which will be 1 1 and 0 because I have not mentioned anything here, but I must give this symbol because just to distinguish it between int flag and int flag this because this is just a variable integer variable this is an integer array. So, you must be careful about this too.

(Refer Slide Time: 16:24)



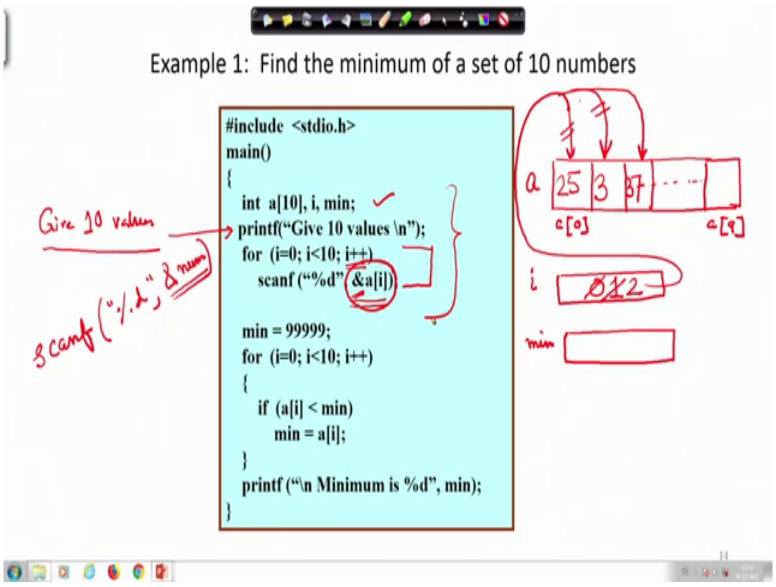
Contd.

– The size may be omitted. In such cases the compiler automatically allocates enough space for all initialized elements.

```
int  flag[] = {1, 1, 1, 0};
char  name[] = {'A', 'm', 'i', 't'};
```

So, similarly I could have done char name, I did not put anything here and just write A m i t.

(Refer Slide Time: 16:33)



Example 1: Find the minimum of a set of 10 numbers

```
#include <stdio.h>
main()
{
   int a[10], i, min;
   printf("Give 10 values \n");
   for (i=0; i<10; i++)
      scanf ("%d", &a[i]);

   min = 99999;
   for (i=0; i<10; i++)
   {
      if (a[i] < min)
         min = a[i];
   }
   printf ("\n Minimum is %d", min);
}
```

Now, let us come to this example finding the minimum of 10 numbers. Now, we are doing our first programming with arrays. We are doing the first programming using the arrays.

Now, here what we are doing is first we are reading the numbers I am reading the numbers all right. I am reading the numbers so and then I am storing them in an array.

So, let us look at this line by line. Let us look at the declarations; int a 10 what does it mean that I have got an array of size 10 and the name of the array is a all right, there is space for 10 elements a 0 to a 9.

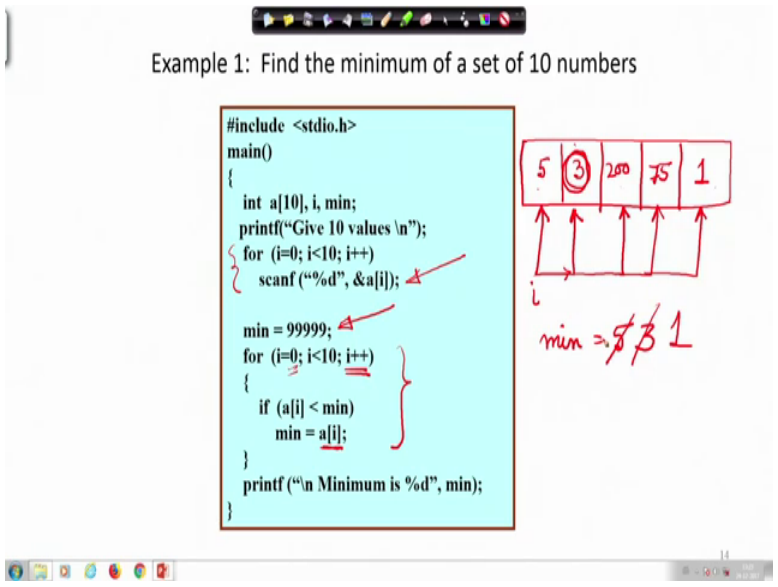
Next, I have got another variable I one variable another variable mean. So, that is my declaration part now I am printing here give 10 values. So, on the screen I will find that give 10 values and then backslash n.

So, now the user is entering 10 values and what am I doing? I am reading those 10 values. Now, compare when you till now when we read the different integers what did you do we wrote it like this right, scanf, percentage d meaning that is an integer ampersand, num something like that we did to read an integer called num. Compare that with what we have done here. Here what we have done is scanf percentage d remains the same because whatever I am reading is nothing, but an integer and ampersand a i; that means, this array is ith element I am reading, this means a I means I am reading the ith element of this array a. What is the value of i? i is 0 initially.

So, first time whenever I am I have asked 10 values. So, i is 0, first i is pointing here. So, i is 0. So, this; that means, this is pointing here. I read a particular value 25, I store it here all right. Then I go in the loop what happens? i plus plus, i is incremented to 1; please observe this carefully this variable is incremented to 1; that means, now the index shifts from here to here a 1 and I read and that is less than 10 typical for loop. So, I come here read the second value say 3. Go back increment I it becomes 2 less than 10 I come here let this brings 2, means again this is changed to the next element all right and I read the next value which may be 37. In that way I go on and read 10 values. Every time I am reading one particular integer and that is being, where am I storing it that that is the importance of this ampersand; that means, that I am reading this at the address of a i, so at this location.

So, this is the first part, first part of the program that has read the array. Now I want to find out the minimum of the array. Now, let us see.

(Refer Slide Time: 21:46)



Example 1: Find the minimum of a set of 10 numbers

```
#include <stdio.h>
main()
{
  int a[10], i, min;
  printf("Give 10 values \n");
  for (i=0; i<10; i++)
    scanf ("%d", &a[i]);

  min = 99999;
  for (i=0; i<10; i++)
  {
    if (a[i] < min)
      min = a[i];
  }
  printf ("\n Minimum is %d", min);
}
```

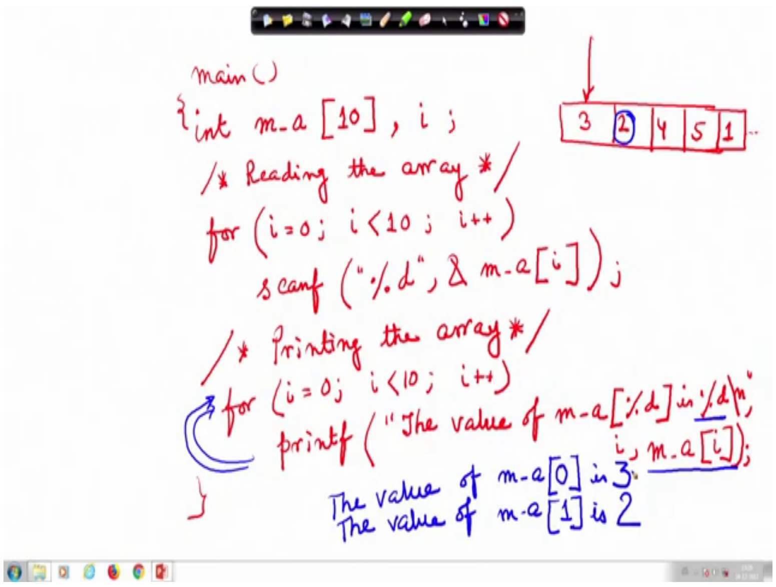
Let me now. So, is this part clear how we read the array now? So, in this way suppose I have read the array some array like say 5, 3, 200, 75, 1. Now, in this part of the program I am trying to find the minimum and that is typically what we did last time, done here in a little bit different way I am starting with a min mean which is very large. So, I am assuming that there will be no data given at this point, at this point no data will be given which is as large as 9 9 9 9 9, I am assuming that.

So, again in a for loop here i is starting from 0, that is i is pointing here. I am comparing a i; that means, a 0 a 0 we with min a 0 is less than min therefore, I can assign a 0 to min. So, my min becomes 5. I go back to the loop i is incremented. So, i is incremented and i comes here; less than 10, so I come here and again compare a i. What is a i? a i is this value 3, is 3 less than min yes. So, the min becomes 3. I again go back increment the index.

So, you see in this for loop I am incrementing the index. We will see how we can write it in different ways. Now, this is a very important exercise and you will have to program it yourself meet with errors at several times, but then ultimately we will find it that is not that difficult. So, i is now 3, I again compare 200 with min now 200 is not less than the min. I go back to the loop I is incremented, 75 is not less than the min I come here, 1 is less than the mean. So, this will be like this in this way it goes on.

So, here we could find two distinct applications of this for loop, one for reading the array and storing it in a set of locations and the other one is looking through that array for every element I am looking through this area and I am finding the minimum. Now, let us do a little bit of exercise before we conclude this lecture today. Say I want to read an array of integers and I want to print the array. So, what should I do? I am leaving out the standard include stdio dot h declaration, declarations let me do; int let me call it, an array m underscore a meaning my array and let the size be 10 all right.

(Refer Slide Time: 25:24)



```
main()
{ int m_a [10], i;
  /* Reading the array */
  for (i=0; i<10; i++)
      scanf ("%d", & m_a[i]);
  /* Printing the array */
  for (i=0; i<10; i++)
      printf ("The value of m_a[%d] is %d\n",
                                    i, m_a[i]);
}
```

```
The value of m_a[0] is 3
The value of m_a[1] is 2
```

Also i, and then index i index need not be i, but it must be an integer variable and then I am starting to read the array, I am trying to read the array. So, my program started here. For reading the array, I am doing for i assign 0, i less than less than 10, i plus plus scanf percentage d. And what am I reading? m a i. So, in a loop I will be reading m a 0, m a 1, m a 2 like that and then I want to print the array.

So, suppose I have read an array which is something like 3 2 4 5 1. Now, I want to print the array. So, what shall I do? I will just simply again I will take one elements one by one and will be printing them. So, I can write for i assign 0, i less than 10 i plus plus; that means, I will be doing this. So, i is starting with 0 and will go up to this here I am assuming that there are 10 elements here although I have shown only 5, then printf I can say that, so I am printing the value of m a this percentage d. So, these things will be as it is printed except for this is percentage d backslash n and sorry backslash n and then it is continuing m underscore a i. Let us see what will happen and then I conclude this.

So, I have stored this now I am printing this. What will be printed? Some printing will be something like this let me use another color for this. What will be printed is the value first iteration. First iteration what will be printed. The value of m a; what is the value of i? I am sorry here, there should be another one I have written wrong they are two percentage d is, so here should be i comma this all right. So, I will repeat again. The value of m a. Then this, then this part will come and this will correspond to the first variable that is i. What is the value of i? Value of i is 0 is, then the second with, second place holder that is m a 0s value; that means, it is 3 backslash n. Again it will go in the loop and what will it again print, the value of m underscore a will be printed as it is percentage d the value of the index of m a 1 is then the value m a 1 is 2, is 2 like that it will be printed. So, these are two fundamental operations of reading an array and printing an array. We will continue further with the arrays.