

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

Lecture – 33
String Operations

We have looked at arrays and their utility, we have in particular looked at one dimensional array and today we will introduce another variety of array, which is 2-dimensional array. In fact, we can have multi-dimensional arrays in general arrays can be; we can have in n dimensional array.

(Refer Slide Time: 00:35).

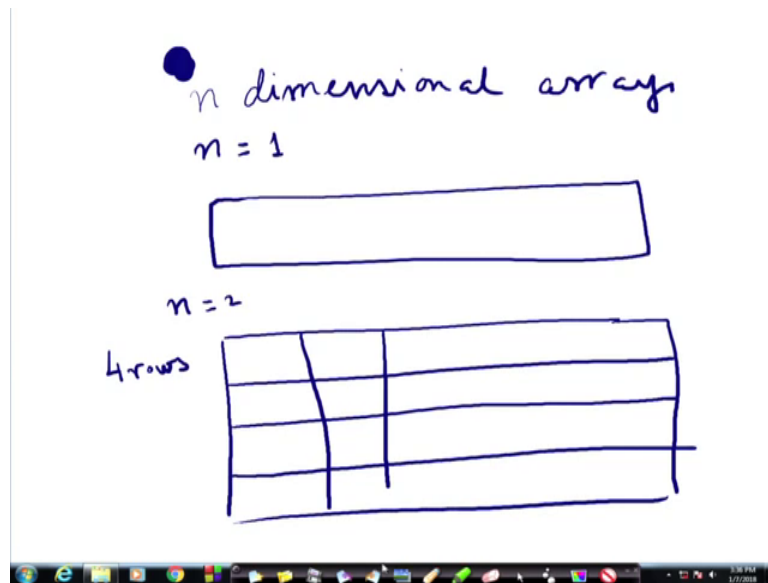
Two Dimensional Arrays

- We have seen that an array variable can store a list of values.
- Many applications require us to store a **table** of values.

	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5
Student 1	75	82	90	65	76
Student 2	68	75	80	70	72
Student 3	88	74	85	76	80
Student 4	50	65	68	40	70

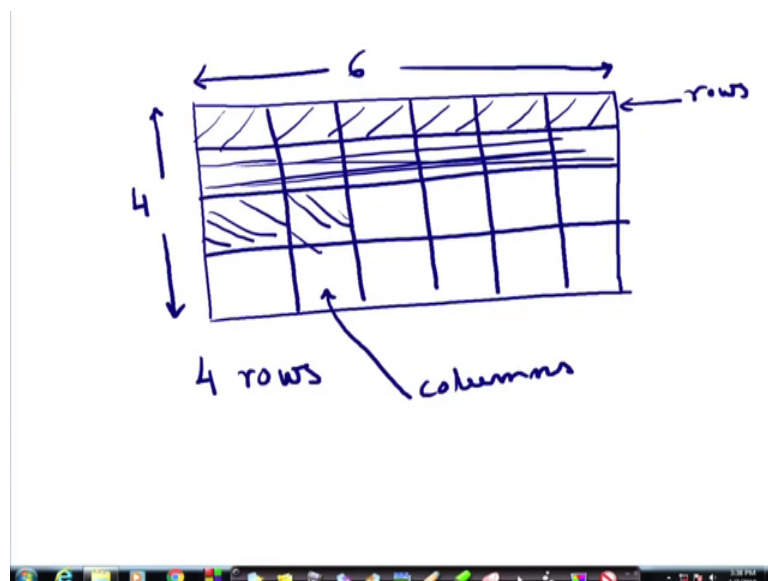
75

(Refer Slide Time: 00:51)



For a particular value of n equal to 1, we are having this one-dimensional array, right? Where, I have got some say 1 dimensional array of integers. So, for n equal to 2 we can have 2 dimensional arrays; that means, where there will length and there will be a breadth of the array. So, that array can be considered of consisting of a number of 1 dimensional arrays for example, as I am showing in this diagram I have got 2 dimensions on this dimension, I have got 4 rows on and on this dimension, I am sorry, this let me draw it a fresh.

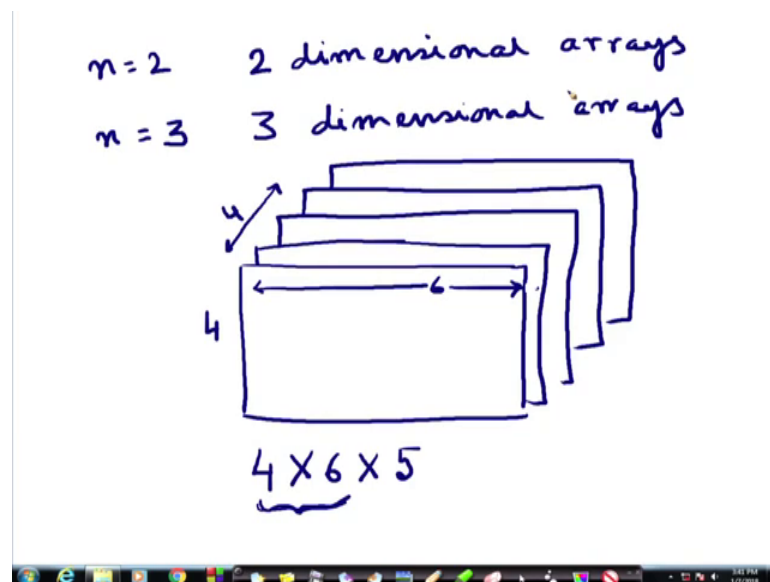
(Refer Slide Time: 01:54)



So, on this side I have got 4 rows and, on this side, I have got 1 2 3 4 5 6 positions. So, these are called columns and these are called rows, right? So, here I have got 2 dimensions one is the length, that is 6 or 6 columns and the breadth which is 4 rows since, these 2 dimensions. So, it is a 2-dimensional array.

So, a 2-dimensional array like this can be considered to be consisting of see in this case 4 1 dimensional arrays arrange one after another. So, this is one array, which we learnt till now this is another array this part is another array this part is another array. So, we have got 4 different arrays 1 2 3 4, 4 different arrays arranged one after another.

(Refer Slide Time: 03:54)



So, I can have for n equal to 2, I have got rows and columns. So, we get 2 dimensional arrays. Similarly, for n equal to 3 we can have 3 dimensional arrays, like so I have got one 2 dimensional arrays like this, and we can think of another 2 dimensional array line here, another 2 dimensional array line here, in that way I can have an arrangement of say may be 3 or 4, 2 dimensional arrays one after another, that is also an arrangement each of these may be 4 rows and 6 columns, sorry may be like this 4 rows and 6 columns.

So, this row this 3-dimensional array will be 4 rows 6 columns and 4 such, 2 dimensional arrays this is one 2-dimensional array this another 2-dimensional array, I am and this one 2-dimensional array and we have got 4 such 2-dimensional arrays arranged one after another, right? If I had a one more here, if I had one more here, same array then this would be changed to 5, in that way I can extend my concept of arranging data as we had

done for 1 dimensional array, we are extending it to 2 dimensions 3 dimensions in that way, it be difficult to concept I mean show it on the this screen we can go up to n.

So, n dimensional arrays, which will have different dimensions all together, but today we will be just discussing about 2 dimensional arrays and the thing is quite general and can be extended to other dimensional arrays also

So, we are now, talking about 2 dimensional arrays you can see the most common example of 2-dimensional array is a table means an arrangement of data just like this. This is list arrangement of information is known as a table, often we see table in other forms also for example, I can have a table of something like this.

(Refer Slide Time: 07:02)

	name	age	school	address
1				
2				
3				
4				

Table
organized in a tabular form

So, the first here I put name of the boy, name of a student then I store age of a student, school to which the student goes and address of the student may be, and I have got such scope of keeping it for say 4 students one student number 1, student number 2, student number 3, student number of 4 and I go on. So, this is this is known as a table because, I am organising them in a organised in a tabular form.

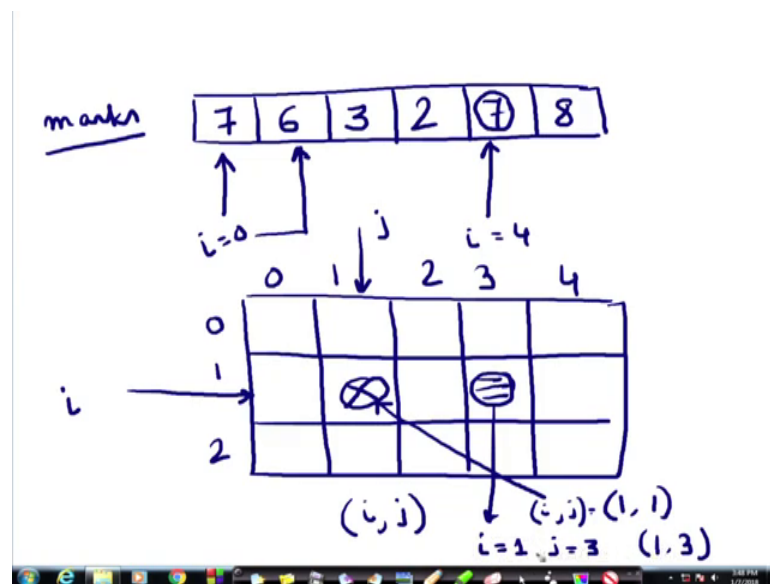
So, this is a table. So, what we were looking at now, is another table where here I am showing that, there are 4 students student 1, student 2, student 3, student 4 for each of the students I have got some marks in the different subjects, subject 1, subject 2, subject 3, subject 4 and subject 5. So, I can consider that all the marks of student 1 in the different

subjects are stored in a row, this row for the first student similarly, this row for the 2nd student, right? Similarly, this row for the 3rd student and this row for the 4th student.

So, each row wise I get the information of all students and column wise if I look in this direction, what do I get in subject 1 what are the different marks? And who have got what marks? And most interesting part is the intersection say for example, this element of the array what will this element tell me, this element has is identified by it is row and column, which row it is in and which column it is in, this element is 75 when what is the meaning of this 75 is the student 2s sub marks of subject 2, it is in whatever we tell you can tell, that is the subject to marks of student 2, this is the subject 3 marks of student 3.

So, in that way, I can represent any of these as identified by some row, name row number and column number recall that, in the case of 1 dimensional array.

(Refer Slide Time: 10:25)



When we are talking of one dimensional array there we had indices index, right? So, that index was telling us say for example, there are suppose these are marks again out of 10 alright, and I wanted to know what is the marks in subject 1. So, the i value i equal to 0 was pointing me to this, suppose this is marks i value was showing this, if I take the i value 1, then I get the marks of this subject like that now, in that case in the case of 1 if I wanted to have this element, that was identified by i value equal to 0 1 2 3 4, but in the case of a 2 dimensional array just as we are seeing here, we will be getting to a particular

element again in the same way with index, but here say for example, this element this element here.

How do I identify this element? This element will be identified by this row and this column. So, my index while this was for one dimensional array the index was i , in this case the index will be i and j . So, any pair $i j$ pair will identify which row and which column. So, this particular element is determined by the i value again just because, it is see I am saying that, it is starting from 0 1 2 and here 0 1 2 3 4. So, say this element can be identified by i is equal to 1 1, right? For example, this element this element is identified by sorry, index value $i j$ I am sorry here, the index is 1 1.

So, the $i j$ pair a pair is 1 1 for this i should be 1 and j should be 3. So, this pair is identified by 1, 3 whenever I get. So, suppose here this array was marks, similarly I will have an a name for this 2-dimensional array and I can access any element by this indices, but in this case since it is a 2-dimensional array, I have got 2 pointers 2 parts of this row part and the column part. So, here you can see that, the subjects are organised in this way subject marks are organised in this way and we can access any of this elements using the proper indexes student number and the subject number.

(Refer Slide Time: 14:07)

Contd.

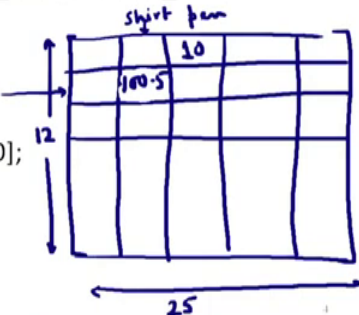
- The table contains a total of 20 values, five in each line.
 - The table can be regarded as a **matrix** consisting of **four rows** and **five columns**.
- C allows us to define such tables of items by using **two-dimensional** arrays.

In the table that was shown here, a showing how many elements 1 2 3 4 5 into 420 elements, right? So, the table contains 20 values 5 in each line and it can be arranged organises as a matrix 2-dimensional matrix consisting of 4 rows and 5 columns.

(Refer Slide Time: 14:40)

Declaring 2-D Arrays

- General form:
type array_name [row_size][column_size];
- Examples:
int marks[4][5];
float sales[12][25];
double matrix[100][100];



Now, C allows us to define such tables by using 2-dimensional array what does it mean, the general form is like this type, array name, row size, column size. Always you compare with what we learned for 1 dimensional array is just an extension of that, for example, marks 4 5. So, in this case which we saw just now, it is an array with and let us call this say, the name of this array is it also marks.

(Refer Slide Time: 15:12)

Two Dimensional Arrays

- We have seen that an array variable can store a list of values.
- Many applications require us to store a **table** of values.

	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5	marks
Student 1	75	82	90	65	76	
Student 2	68	75	80	70	72	
Student 3	88	74	85	76	80	
Student 4	50	65	68	40	70	

marks [4][5];

And marks has got. So, let me write here, because it is marks and it has got 4 rows and it has got 5 columns. So, that is how I can describe this array, and then we can go on filling

up the values of this array. So, array name, row size and column size. So, for example, int marks, similarly I can have cells of 25 items over 12 marks may be say, I can have something like, so let us try to see what this possibly can mean is that suppose I have got a table where, there are 12 rows alright, I will have 12 rows here and there will be 25 columns here alright, there will be 300 items so I am not showing that.

But say each of these rows may mean say this is January row, this is a February row, this is a march row etc, and suppose here there are different items or shop has got different items and in that way it has got 25 different items and here. We write how many units of a particular this row is shirts alright, this one says that, in the month of February how many shirts have been sold and may be this says in the month of say this is pen in the month of January or in the month of January how many pens have been sold?

So, in that way this entire matrix or this array can be named as the arrays cells, which will have 12 rows and 25 columns, similarly and another very important thing to note is this type, this type that we declared also in the case of 1 dimensional array is also applicable here, because each of these elements in this array will be of the same type. So, here cells I am saying and I have said cells is float.

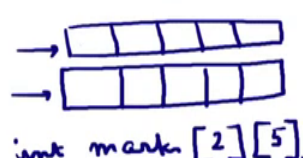
So, the way I wrote it is not correct I should write it as say, how many elements are there? What it can be that in the month of January? What is the worth of not how many shirts were sold? But, shirts worth how many rupees were sold. So, it can be here 100.5 say because, each of these elements have floating point numbers, alright? Similarly, matrix can be a 100 by 100 matrix where, each of the elements will there double precision number. So, we have got 2 places, 2 dimensions that we are specifying here.

(Refer Slide Time: 19:03).

Declaring 2-D Arrays

- General form:
type array_name [row_size][column_size];
- Examples:
int marks[4][5];
float sales[12][25];
double matrix[100][100];

int marks [5]



int marks [2][5];

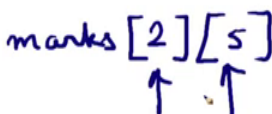
Unlike what we did in the case of a 1-dimensional array where, we did something like int marks say 5. So, int marks 5 is a 1-dimensional array with 5 elements, I can say that these are with 5 columns of this 1-dimensional array, which is 1 row had there been more rows like this, then immediately I have to identify which row and which column and if there be 2 such rows, then this array should be described as int marks say here, 2 5 because, they are rows.

(Refer Slide Time: 20:01)

Accessing Elements of a 2-D Array

- Similar to that for 1-D array, but use **two indices**.
 - First indicates **row**, **second** indicates **column**.
 - Both the indices should be expressions which evaluate to integer values.
- Examples:

marks [2][5]

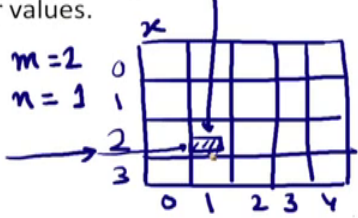


Similar to that of 1 dimensional array, but we use 2 indices, indices is the plural of index indices, right? So, the first one indicates the row and the second one indicates the column both the indices should be expressions which, evaluate to integer values and that was the case for 1 dimensional arrays also. So, if I have something like marks 2 5 these integer, these values must be integers, I can write expressions we will show there later, but those expressions must evaluate to integer values, alright?

(Refer Slide Time: 21:14)

Accessing Elements of a 2-D Array

- Similar to that for 1-D array, but use **two indices**.
 - First indicates **row**, **second** indicates **column**.
 - Both the indices should be expressions which evaluate to integer values.
- Examples:
 $x[m][n] = 0;$



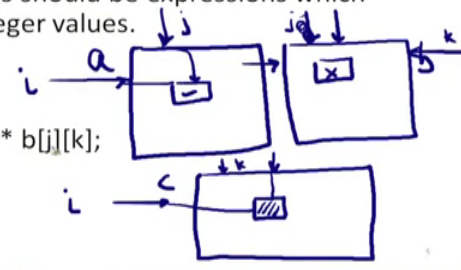
$m=2$
 $n=1$

Examples are $x[m][n]$. So, whatever m value is m has to be an integer, suppose m here is 2 and n is 1, then in this array whose name is x we have got a number of rows and a number of columns, which element am I referring to here, I am referring to row to 0 1 2 3. So, I am referring to this row and which column I am referring to 1. So, this was 0 1 2 3 4. So, I am referring to this column therefore, I am actually pointing at this particular element, alright? This particular element.

(Refer Slide Time: 22:25).

Accessing Elements of a 2-D Array

- Similar to that for 1-D array, but use **two indices**.
 - First indicates **row**, second indicates **column**.
 - Both the indices should be expressions which evaluate to integer values.
- Examples:
 $x[m][n] = 0;$
 $c[i][k] += a[i][j] * b[j][k];$



Similarly, I can do operations like this what does this mean let us try to understand this means, I have I am referring to 3 different 2-dimensional arrays one is a, which has got some rows and columns and these another array this is a this is b and I have got another array c..

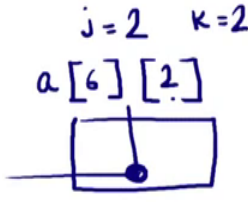
Now, I am taking what is being done here, this means c i k. So, this c is being designated by an index i, another index k this one is being done by referred by i and some j this 1 by j and k what is being done here, that for different values of i j and k, I am selecting this has got a special application, but I am not going to that I am looking at taking at the taking a particular value from here, that is c i and c k k is pointing here.

So, this one I am taking and adding that, with the product of a i and j, j might be pointing somewhere here, I am taking this element and I am taking may be b j for b array j row I am sorry, j row j should be here, j row and k column. So, I am taking another element from here. So, I am taking this element multiplied by this element here, and adding that with this element that is taken from c, that is what I am doing. So, this simple example, but this has got for mode implication you will realise that soon.

(Refer Slide Time: 25:02)

Accessing Elements of a 2-D Array

- Similar to that for 1-D array, but use **two indices**.
 - First indicates **row**, second indicates **column**.
 - Both the indices should be expressions which evaluate to integer values.
- Examples:
 - $x[m][n] = 0;$
 - $c[i][k] += a[i][j] * b[j][k];$
 - $b = \text{sqrt}(a[j*3][k]);$



Here you see, what we have done here why is this example shown here, you see we are specifying the index in the form of an expression, suppose j was 2 and here, which one I am referring to a j into 2 so; that means, a 6 followed by whatever value of k was might be 6 2; that means, I am referring to the element of the array a , a is an array whose 7th row and 3rd column element is being taken.

So, if this be like this I am taking this 7th row element and 2nd or 3rd column element, I am taking that value and finding this square root of that, and that is again being put to the corresponding value of that, is coming to some other variable name actually this should be since, this an array this should have been kept as some other variable in b , alright?

(Refer Slide Time: 26:19).

How is a 2-D array is stored in memory?

- Starting from a given memory location, the elements are stored **row-wise** in consecutive memory locations.
 - x: starting address of the array in memory
 - c: number of columns
 - k: number of bytes allocated per array element

– $a[i][j]$ → is allocated memory location at

$$\text{address } x + (i * c + j) * k$$

$a[0][0]$ $a[0][1]$ $a[0][2]$ $a[0][3]$ $a[1][0]$ $a[1][1]$ $a[1][2]$ $a[1][3]$ $a[2][0]$ $a[2][1]$ $a[2][2]$ $a[2][3]$

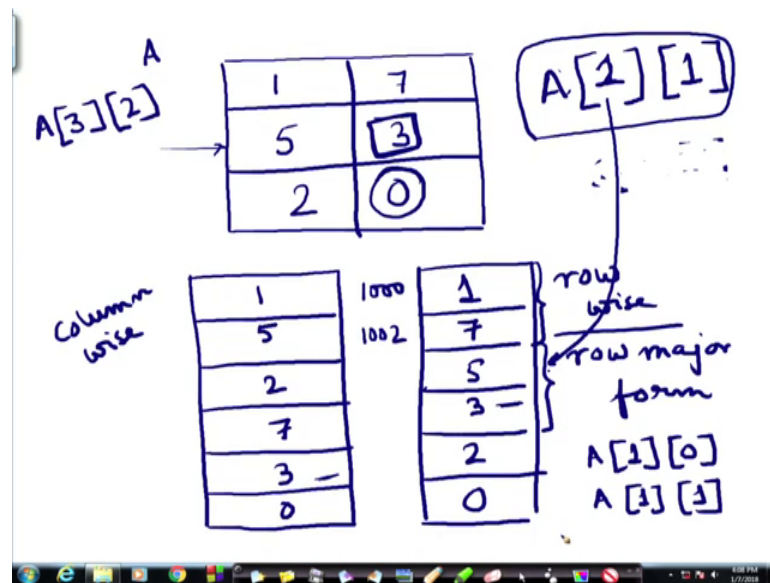
So, how is 2 dimensional arrays stored in memory we know that, a 1-dimensional array is stored in a memory row I mean in the contiguous memory locations.

(Refer Slide Time: 26:46)

3
5
4
7

So, just for the sake of recalling if we had, if I have a 1 dimensional array say 3 5 4 7, they were being mapped to the memory actual memory, here is the memory location the different memory locations and I am storing them here, 3 5 4 7 in contiguous locations, but what happens to the case of 2 dimensional array for example, if I have the array that I am drawing here, with 3 rows and 2 columns, alright?

(Refer Slide Time: 27:24)



So, what is the dimension of this array a, a is 3 2 3 rows and 2 columns now, suppose it is 1 7 5 3 2 0 say those are the values in that, case if I stored it in the memory I can stored it now, I am drawing how it can be stored in the memory I need 1 2 3 4 5 6 locations in the memory. So, I will have 6 locations. So, I can store them 1 5 2 7 3 0 or I can store them as 1 7 5 3 2 0. Now, what is a difference between these 2 storages this one is I am storing them column wise and, in this case, I am storing them row wise row after row, right?

In c the compiler stores the arrays row wise, which is often known as in the row major form and this is the column major form, but in c we are doing it in row major form there is a specific significance of this. Because, whenever I want to access an element say a array, I want to find the particular element say 2 3, which element shall I find out here for example, here let us take this case and let me try to find out the element say A this is A capital A I want to find out the element 2 1; that means, I am actually trying to find out the row number 2 and element number 1. So, I am trying to find this one out. Now, where will it be if I do not know. So, say this is accidentally I came coming to the same let us say, let us make this 1 1 1; that means, which one row 1 column 1. So, I am actually trying to find this element 3.

Now, what is the address of this element, where is it stored you can see that 3 has been stored here if I store in the column major form, but if I store in the row major form it is

here. So, whenever I am trying to find this element out of this memory representation I will actually have to fetch it from the memory, right? So, therefore, I have to find out the address of this and for that, I need to know where it is stored row wise or column wise since, I know it is row wise.

So, how should I go about it since, there are 2 columns, so for the first row I will say that, if this been my starting address 1000 in the first row we will take 1000 and 1000 this is starting from 1002, alright? This is the first row, this is the second row, and this means the second-row part and second rows this is one what is this element 5, this is second row and 0th element and this one is second row and then the next element.

So, it is 1 1. So, using knowing that, whether it is stored in the row major form or column major form is very essential in order that, I can find out the address of the element in a 2-dimensional array, we will continue this further more explanations are required about this.