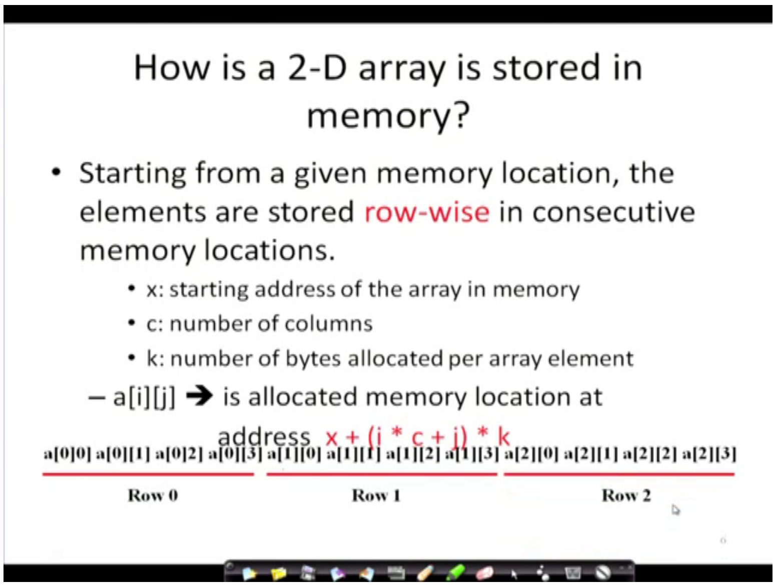
Problem Solving through Programming in C

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

Lecture – 34
2-D Array Operation

So, we were discussing that how a 2 dimensional array is stored in the memory.

(Refer Slide Time: 00:23)



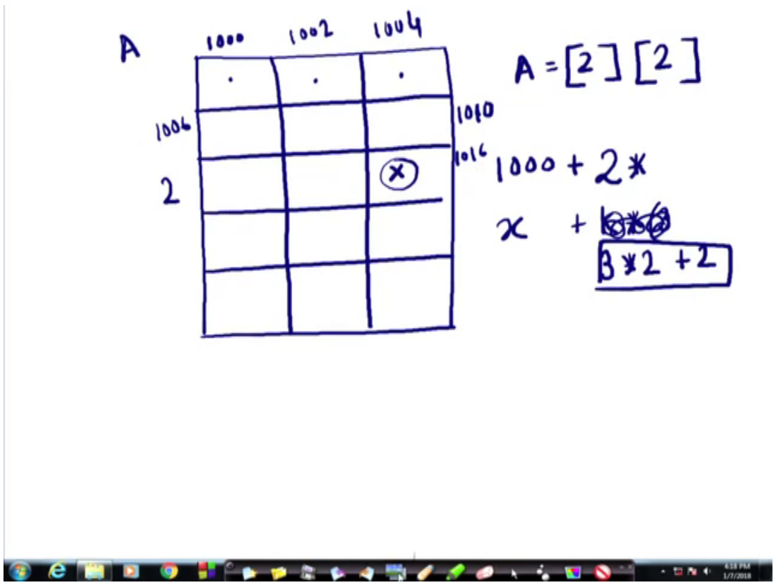
How is a 2-D array is stored in memory?

- 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 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]
Row 0
Row 1
Row 2

Now, we have already said that that is and 2 dimensional array is stored in a row wise form. So, we have to find out the address of a particular element in the memory, right.

(Refer Slide Time: 00:47)



If we have a 2 dimensional array and the row and the column is specified all right. So, say here there are 1 2 3 4 5 rows and 3 columns say a particular element here has got the address say the name of this array is A this particular element has got the address A 0 1 2, A 2 3, A 2 0 1 2, A 2 2 right this element. So, it is stored how it is stored this stored in this consecutive way first this one is stored then this one is stored, then this one is stored row wise right.

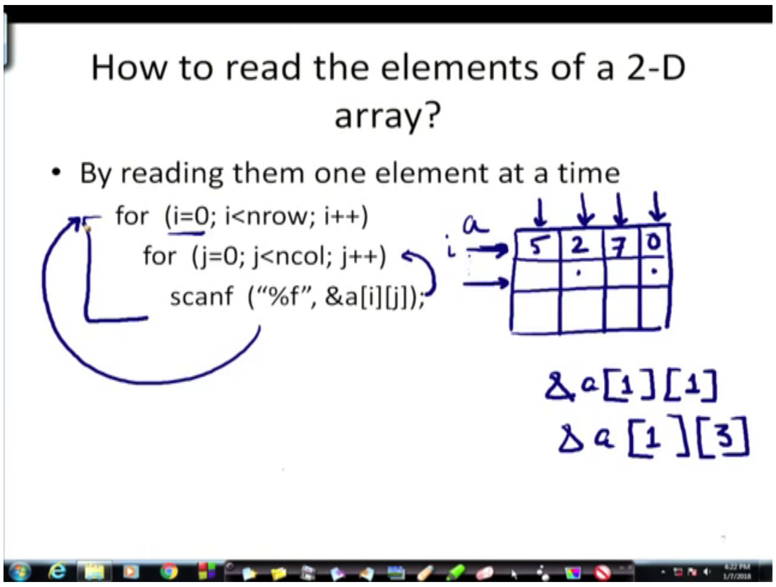
So, if in my each of them have got a memory location, so if the location of this is 1000 and if it be an integer array then this location will be 1002, this location will be 1004, this location will be 1006, 1008, 1010, right, in that way it will go on. So, a little thought will tell you that if the starting address. So, what will be the starting address? What will be the address of this? It will be 1000 plus how many columns have to come for each column shift for each column. What will be the address of this? 1010, 1012, 1014, 1016, right, actually this should be in the location 1016 of the memory.

So, we start with 1000 and then since each of the elements each of the elements are taking 2 bytes. So, I will have to go for each shift 2 bytes and how many such shifts. What is the distance from here to here element wise? We have got 8 such positions right and so with that I can find out the address of this x. So, if this was x and this was element was k then what would be my address; k into the offset is so many columns 2 3, 2 columns sorry 2 means 3 columns per row and 3 such rows. So, it will be 6 rows. So, in general we can say that if x be the starting address of the array in the memory and c be the number of columns and k be the number of bytes allocated then the address will be x plus i into c plus j c is the column number plus j. How many columns are there? So, if I go back to this if I go back to this I can say the how many columns c is each it has got 3 columns. So, k into 3 plus sorry 3 into k plus j right 3 into k means 2 plus j, j was 2.

So, that will be my result 1000 plus 6 plus two 8. So, that will be the location that I will be finding out. So, here that is the formulation; x plus I into c plus j times k, k is number of bytes allocated per element plus please look at this. So, the rows are the matrix is actually stored as a 0 0, a 0 1, a 0 2. So, this entire thing is a 0 3, so 1 2 3 4 so there 4 columns in one row. Then, the second row a 1, a 1 column 0, a 1 column 1, a 1 column 2, a 1 column 3, then, I proceed in this way. So, this is the arrangement of 3 rows and each row having 4 columns and using this formula which is intuitive we can find out the exact location of a particular element.

So, this part is row 1 this part is sorry this part is row 0, this part is row 1, this part is row 2, you can see that this is row 1 column 0, row 1 column 1, row 1 column 2 so and so forth.

(Refer Slide Time: 06:05)



How to read the elements of a 2-D array?

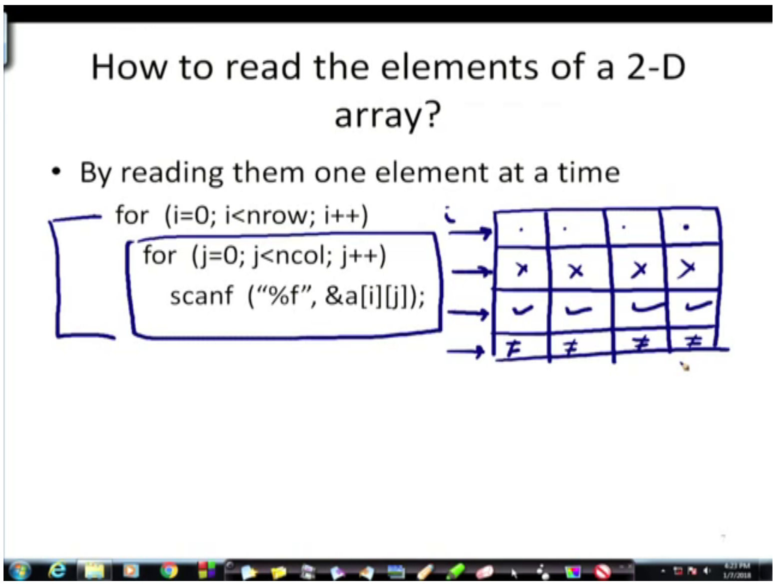
- By reading them one element at a time

```
for  (i=0; i<nrow; i++)
   for  (j=0; j<ncol; j++)
      scanf  ("%f", &a[i][j]);
```

Now, given so that is how it is stored and how we can find out what the address is, but how do we read a particular array elements into a 2 dimensional array. We have seen that I can read one dimensional array the elements of a 1 dimensional array into the array by repeatedly getting the character from the user and storing it in the proper indexes and proper positions by varying the index right. Here also we will be reading one element at a time and we will store them. So, my array is say the array that I am storing is a i j having some rows and columns and i is the row index and j is the column index. So, initially in this for loop look here, there are there is a nested for loop here. So, first I am keeping the i fixed for i equal to 0. Now, I come here inside this, inside this there is a nesting of for loop say I have got 4 columns. So, for j is equal to 0 to j less than 4? What do I do? I scanf and a i j now each element here each element here is identified by the row number and column number and so this one is and a 11, this one is and a 13 in that way we do.

So, now for j is equal to 0 I read the particular value and store it here suppose it is 5 next what is done, I am still inside this loop i increment j. So, j comes here note that i is fixed still how long will a i remain fixed as long as I am inside this inner loop and how long shall I be inside this inner loop till j reaches the end of the number of columns. So, next I read the other value. So, it is 2, next I read the other value it is 7, next I read another value may be it 0, then it comes to 4 comes to the end of the row because all the columns have been filled up then I come out of this loop and again go inside this. So, i is now incremented, i is now incremented to this one and I again do the same thing inside this loop. So, this inside this loop is filling up a row and this one is filling up all the rows.

(Refer Slide Time: 09:40)



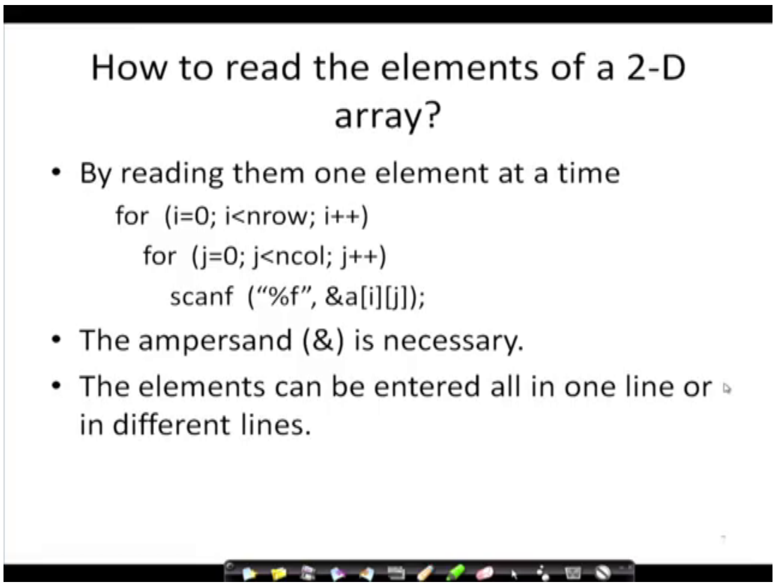
How to read the elements of a 2-D array?

- By reading them one element at a time

```
for (i=0; i<nrow; i++)
   for (j=0; j<ncol; j++)
      scanf ("%f", &a[i][j]);
```

Please take a little time to understand this. This loop, I am once again explaining this loop is filling up one row keeping i fixed, i is fixed and in a loop I am filling up the values here, here, here, here and then I increment this one. So, this part for filling up all the rows one by one, I come here again and again fill this up then I increment i and again fill this up again I increment this i, come here and fill this up all right that is how we read the elements of an array. So, that is being what is being shown here.

(Refer Slide Time: 10:32)



How to read the elements of a 2-D array?

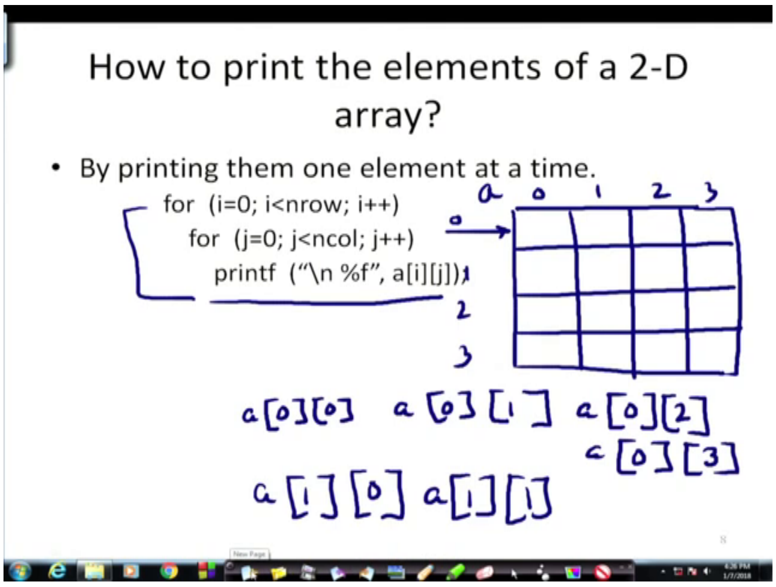
- By reading them one element at a time

```
for (i=0; i<nrow; i++)
   for (j=0; j<ncol; j++)
      scanf ("%f", &a[i][j]);
```

- The ampersand (&) is necessary.
- The elements can be entered all in one line or in different lines.

The ampersand is necessary here just as in the case of an array it was necessary it is necessary here as well. The elements can be entered all in one line or in different lines it really does not matter whether you pet type 5, blank 6, blank 7, blank 8 or you type 5 enter 6 enter like that that really does not matter because we already know that every element will be stored in a row wise fashion.

(Refer Slide Time: 11:08)



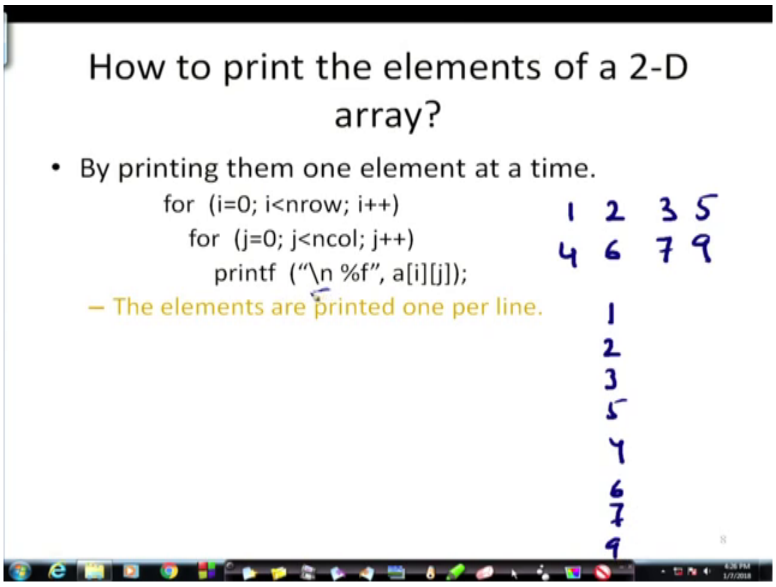
How to print the elements of a 2-D array?

- By printing them one element at a time.

```
for (i=0; i<nrow; i++)
   for (j=0; j<ncol; j++)
      printf ("\n %f", a[i][j]);
```

How to print the elements of an array? Again, suppose I have got an array, suppose I have got an array suppose I have got a 2 dimensional array and I will have to print the elements I cannot just print the array in one shot I cannot do that. So, again here I will be fixed and then I will print say this is a 0 1 2 3 0 1 2 3. So, first a 00 will be printed whatever was the value here I am printing it here, then j will be incremented. So, the next one that will be printed will be a 01, next j will be incremented. So, it will be a 02, a 03 so and so forth and then I will be incremented again in this way. So, it will be next printing will be a 1 and j will be reinitialised to 0 1, a 10, a 11 and like that it will be printed.

(Refer Slide Time: 12:44)



How to print the elements of a 2-D array?

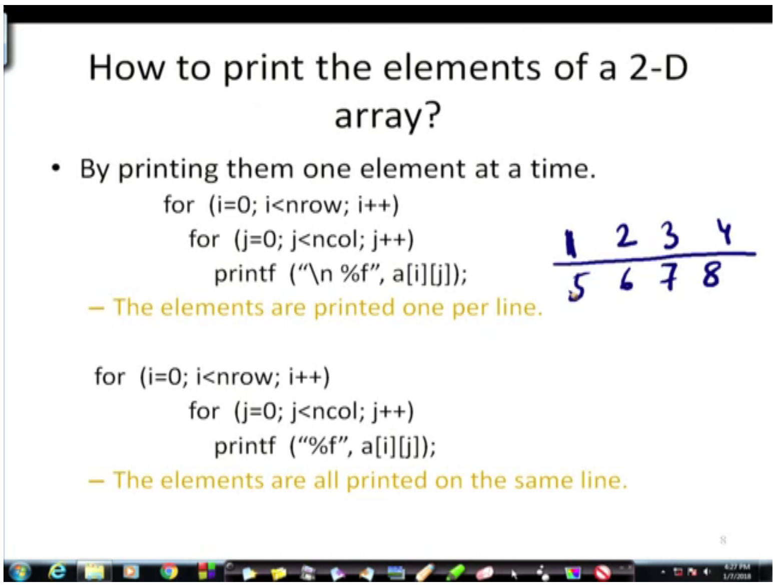
- By printing them one element at a time.

```
for  (i=0; i<nrow; i++)
  for  (j=0; j<ncol; j++)
    printf  ("\n %f", a[i][j]);
```

  - The elements are printed one per line.

Now, the elements are since it is a back slash n you can see the elements have printed one per line because every time I am printing I am giving a back slash n. So, although the; my array could be 1 2 3 5 4 6 7 9, it will be printed as 1 2 3 5 4 6 7 9 because at every point I have given a back slash n.

(Refer Slide Time: 13:23)



How to print the elements of a 2-D array?

- By printing them one element at a time.

```
for  (i=0; i<nrow; i++)
  for  (j=0; j<ncol; j++)
    printf  ("\n %f", a[i][j]);
```

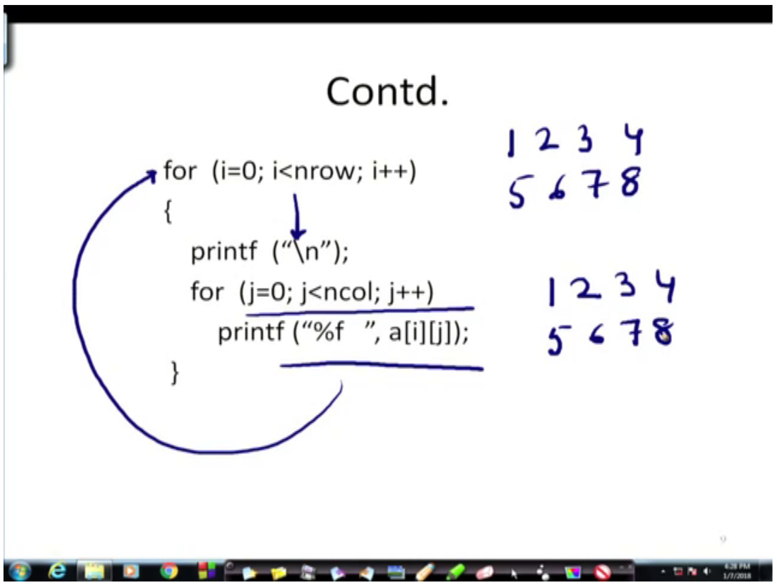
  - The elements are printed one per line.

```
for  (i=0; i<nrow; i++)
  for  (j=0; j<ncol; j++)
    printf  ("%f", a[i][j]);
```

  - The elements are all printed on the same line.

Now, here what are we doing here? In this case the elements will all be printed in the same line because I have not put the back slash n. So, it will be printed all the elements say sorry all the elements say 1 2 3 4 5 6 7 8, it will be printed in this way if that was the matrix then this part will be printed in one line and then I go up there is no back slash n this one will be followed and the entire thing will be printed in one line, 1 2 3 4 5 6 7 8.

(Refer Slide Time: 14:14)

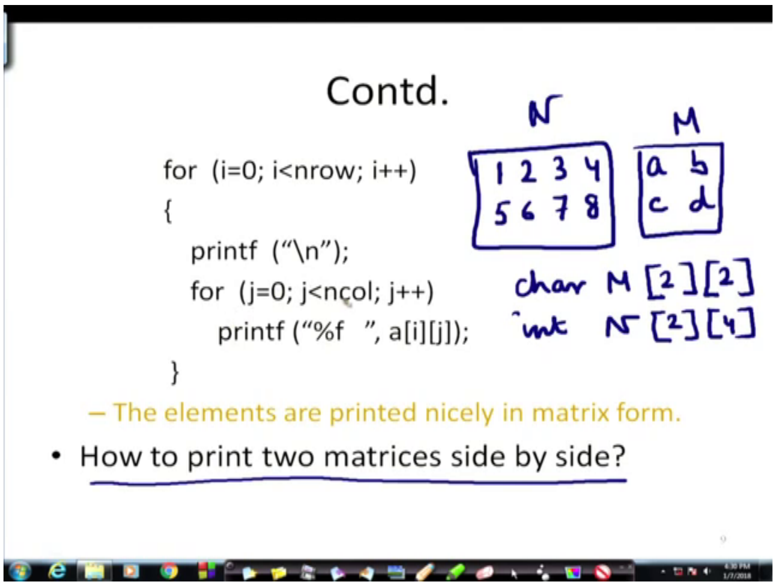


Contd.

```
for  (i=0; i<nrow; i++)
{
   printf  ("\n");
   for  (j=0; j<ncol; j++)
      printf ("%f   ", a[i][j]);
}
```

But if I had done this that first I had this again 1 2 3 4 5 6 7 8 and I want it to be printed in this way because here I have not given any back slash n it will be printed one two 3 4 and as it comes out of this loop it goes here and comes to the next row first thing that it does is printf back slash n. So, the control will come here and then 5 6 7 8 will be printed therefore, this is a better way of printing a 2 dimensional array. So, that it looks like a 2 dimensional array.

(Refer Slide Time: 15:14)



Contd.

```
for  (i=0; i<nrow; i++)
{
   printf  ("\n");
   for  (j=0; j<ncol; j++)
      printf ("%f   ", a[i][j]);
}
```

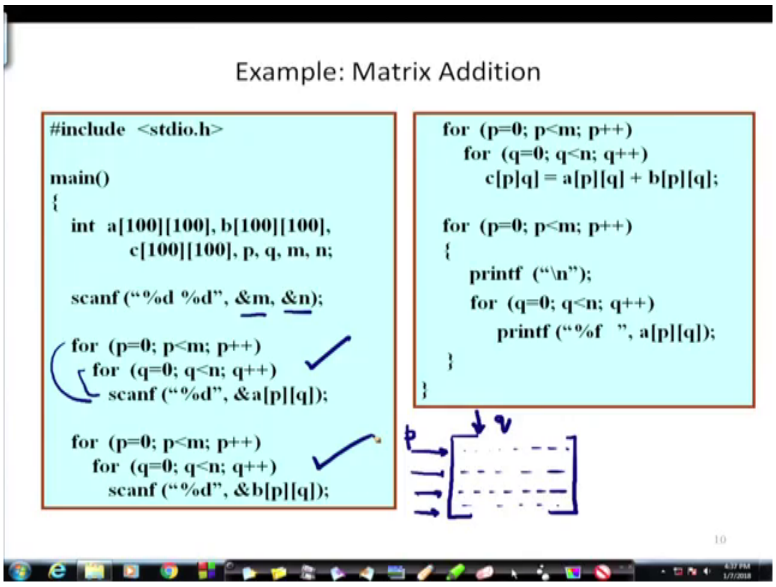
– The elements are printed nicely in matrix form.

- How to print two matrices side by side?

So, that this will make the elements now the question is you just think how can we print two matrices side by side. So, suppose there is one matrix 1 2 3 4 another is character matrix a b c d how can you print them side by side what should we do? Can you think of that? I mean if I say I have got two matrices one is 1 2 3 4 5 6 7 8. Another matrix is say a b c d. So, I they have different dimensions.

So, first of all, first of all how can you represent such a metrics? Say this matrix how will you declare that simple this will be a character matrix character let us call the this matrix let it be M, M is 2 by 2; that means, each elements of this matrix will be a character. And this is int, let it be N, 2 by 4, right. These are the two matrices now you can think of how you can print two matrices side by side. If I do look at this by this row by this I will first print 1 2 3 4 and then what should I do? I should go to here I should again for j is equal to j to M call might be j plus plus, I will be doing I will be printing M i j. So, this will come immediately side by side if necessary I could have given another blank in between and I could have printed this as a complete row next I go up and print this and this. So, I suggest that you try to write this program and run it and see whether you get a nice output or not.

(Refer Slide Time: 17:46)



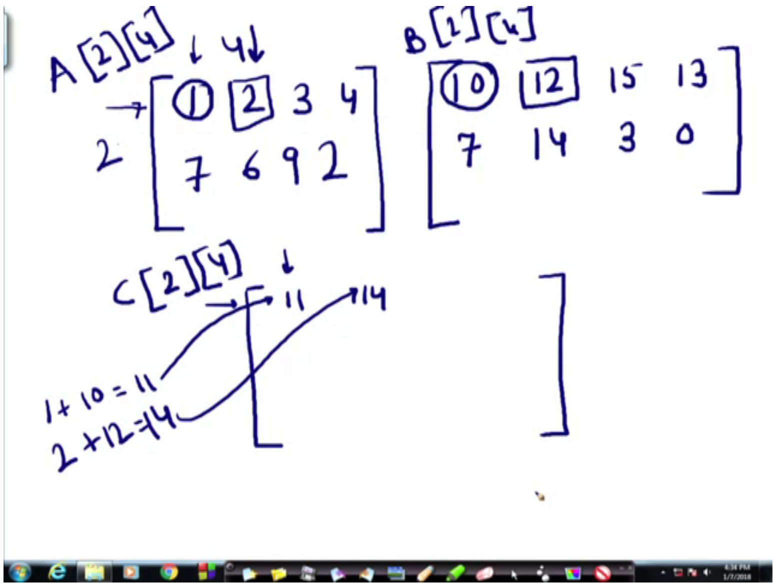
Example: Matrix Addition

```
#include <stdio.h>

main()
{
  int a[100][100], b[100][100],
        c[100][100], p, q, m, n;

  scanf ("%d %d", &m, &n);

  for (p=0; p<m; p++)
    for (q=0; q<n; q++)
      scanf ("%d", &a[p][q]);

  for (p=0; p<m; p++)
    for (q=0; q<n; q++)
      scanf ("%d", &b[p][q]);
```

```
  for (p=0; p<m; p++)
    for (q=0; q<n; q++)
      c[p][q] = a[p][q] + b[p][q];

  for (p=0; p<m; p++)
  {
    printf ("\n");
    for (q=0; q<n; q++)
      printf ("%f  ", a[p][q]);
  }
}
```

Now, we come to a very well known problem that we often encounter matrix addition. I am going to add two matrices. This is a very simple problem. So, all of you know how this can be done.

(Refer Slide Time: 18:04)



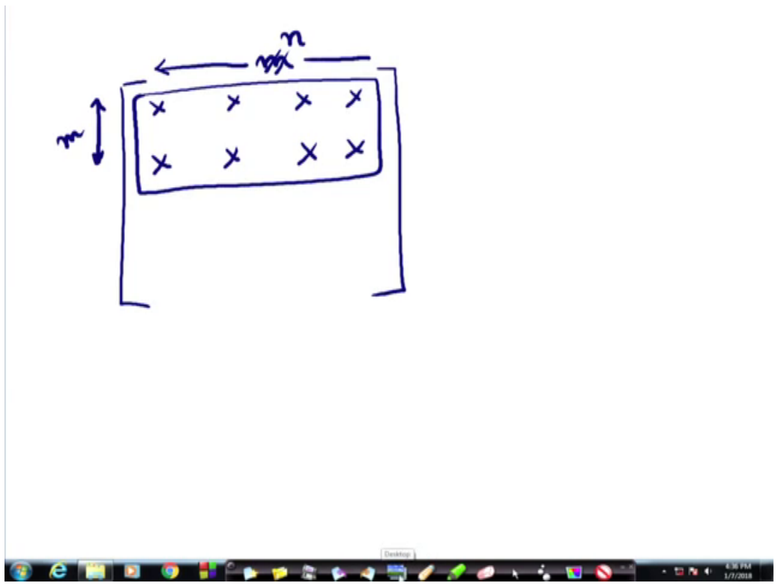
So, I have got a matrix of the same size two matrices. So, it is 1 2 3 4 7 6 9 2 and there is another matrix 10 12 15 13 7 14 3 0. Now, I want to add them and have another matrix. First of all this is a matrix of 2 rows and 4 columns 2 rows and 4 columns. So, we can say that this A is a 2 by 4 matrix and this one B is also a 2 by 4 matrix. Now, the sum I will store in another matrix whose dimension will also be two by 4 has to be. And how is matrix addition done? I will first take this element.

What is this element? I am saying this element what is this element. This element means a particular i j value 0 0 element i 0 j 0 from these array. So, I take this element 1 and then take b i j of the same i j value 0 0 1 add them together 1 plus 10 I get 11, I put that 11 in the i j value remaining same 0 0 of the c matrix. So, here it is 11.

Next, I am increasing the index i here j remains sorry I am incrementing whatever I do I keep the I fixed and increment the j. So, i is 1, j is 1 is 0, j is 1. So, with that I take this value 2 and take this value with a same i and j value, i is 0 and j is 1 add them. So, that is 4teen and I store this value here in that way I go on for this entire matrix that is how I do the matrix addition. So, let us see how the code looks like for this and let us try to understand the code this much is include s t d i o dot h then my main program is continuing. What is being done?

Here I am trying to add 2, I am writing the program for adding 2 matrices of size 100 by 100 let us look at this 100 by 100, a matrix of size 100 rows and 100 columns, b of 100 rows and 100 columns and c of 100 rows and 100 columns. And I have got other integer values p q m n. I am reading the matrices, I am reading sorry I am reading m and n alright two values. Now, what is m and n? m and n although I have here recall the difference between I mention about the size and the dimension is the maximum size the maximum size that it can have 100 and 100 are the maximum sizes of the a matrix and b matrix, but actually I can have I may not need so many locations. So, what I can do?

(Refer Slide Time: 22:01)

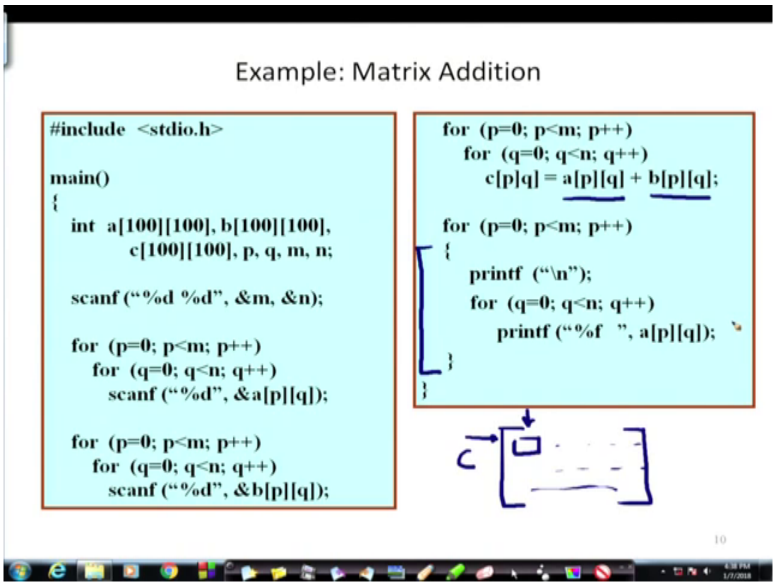


I have got a has got so much size, but actually I am filling up only these say this much. So, this is my m, I am sorry this is my m and this is my n ok.

So, my actual matrix is small dimension is a largest size that I can accommodate all right. So, I need that actual value m and n and then for p is an index p equals to 0 less than m p plus plus q. So, I am reading the matrix a, and here you see I am not using the conventional i j indices I am saying p is keeping account of my row and q is keeping account of my column. So, for p 0 to m and q 0 to n I am filling up the a matrix, then I come here fill up this matrix out of this outer loop.

This inner loop, with this inner loop what am I doing I am filling out particular row and the outer loop I am going to the next column, repeatedly I am saying that. So, in this way I read matrix a here, here I read matrix b then I am doing the addition here. Again my the dimensions of the two matrices will be the same when I add and the result matrix will also be the same according to matrix algebra, that is I am adding two matrices they will result in the same dimension matrix. So, again for p equal to 0 to m and q equal to 0 to m I take c p q. So, I have taken two matrices.

(Refer Slide Time: 2 4:13)



Example: Matrix Addition

```
#include <stdio.h>

main()
{
   int a[100][100], b[100][100],
           c[100][100], p, q, m, n;

   scanf ("%d %d", &m, &n);

   for (p=0; p<m; p++)
      for (q=0; q<n; q++)
        scanf ("%d", &a[p][q]);

   for (p=0; p<m; p++)
      for (q=0; q<n; q++)
        scanf ("%d", &b[p][q]);
```

```
for (p=0; p<m; p++)
   for (q=0; q<n; q++)
      c[p][q] = a[p][q] + b[p][q];

for (p=0; p<m; p++)
{
   printf ("\n");
   for (q=0; q<n; q++)
      printf ("%f  ", a[p][q]);
}
}
```

And now I am generating c matrix where this value this particular element is being computed by the corresponding element from a and the corresponding element from b being added I am filling it up and this is going on in a loop all right and so I fill this entire thing up. What is a last for loop? This last for loop is nothing, but printing the same array, printing the array in the proper way.

Now, here you please note that it is back slash n. So, it will come nicely after one row there will be a we will go to the new line and we will print it this way. So, this is a matrix addition which is a very useful very common and simple application of our knowledge of two dimensional array to start with.

We will other applications of this to in a future. Now, the other thing that will be discussing in the consequence lectures are next. So, we now have got an idea of 1 dimensional array and we have got an idea of 2 dimensional array. We have also seen strings and character arrays. Next, we will move to functions in the next lecture. We have already mentioned about the term functions while moving and in the course of other discussion, and we have already seen one particular function that is main function that is always there is a main body of any C program and we have also seen some library functions. But next lecture onwards we will look at user defined functions.

We have seen some stored functions in the earlier lecture like string copy, string length, finding string length, similarly a user can himself or herself write a particular function of his or her own need. We will have to see why we need to write functions and how a function can be written from the next lecture onwards.

Thank you.