December 21, 2015

Launchpad

Lecture - 7

Multi Dimension Arrays

Anushray Gupta

CODING BLOCKS

Status of Assignment?

Any doubts?

2 D Arrays

2 D Arrays Declaration/Initialization

I. int array1[2][3];
II. int array2[2][3] = {{1,2,3}, {4,5,6}};
III. Int array[][4] = {{1,2,3,4}, {4,5,6,7}, {8,9,10}};
IV. char array3[3][2] = {{'A','B'}, {'C','D'}, {'E','F'}};
V. char array4[][4] = {"abc", "def", "efg", "hig"};

Accessing an array

I. 2-D array can be visualized as a matrix with N rows and M Columns.
II. First element is 0,0 and last is N-1, M-1
III. To access jth element of ith row [considering i and j are 0 based] we can use arr[i][j] where arr is the name of the array.

Lets write some

I. Read a matrix and print transpose of it.
II. Read a matrix and find a number in it.

Time to try?

1. Write a program that determines which row or column in a 2d array of integers has the largest sum

How is it stored?

Depending on the architecture it could be either stored as:

I. Column Major Form
II. Row Major Form – Most common!

So what is arr[i][j]?

I. We know that name of the array is address of first element.
II. So when we are saying arr[i][j] its doing some calculation like *(arr + i * number of columns + j)
III. Conceptually this is correct but actually this is wrong.

Lets look at 1-D array again

int arr[3] = {1,2,3};

I. We know arr is an alias of address of first element i.e. arr == &arr[0]
II. But what is &arr ? Initially its value is same as arr but lets just try to increment it by 1 and see. cout << &arr +1 << endl;
III. This address is N*sizeof(data) far from the initial address where N is number of elements.
IV. So we can say &arr is also an address but its not address of one element but address of a complete row. We can say it's a pointer to array or a row pointer.

Lets see output of these statements

```
Int arr[][3] = {{1,2,3}, {4,5,6}, {7,8,9}};

cout <<arr << endl;
cout << arr+1 << endl;
cout << *(arr+1) << endl;
cout <<arr[0] << endl;
cout  <<*(arr[0]) << endl;
cout << &arr[0][0] << endl;
cout << arr[0] + 1 << endl;
cout << (&arr[0][0]) + 1 << endl;
cout << arr+1 << endl;
cout << &arr[1][0] << endl;
```

So what is actually arr[i][j]

I. For a 1-D array arr[i] is smilar to *(arr+i)
II. Similarly for 2-D array arr[i][j] is actually *(*(arr+i)+j)
III. Now name of the array is a row pointer or we can say it is pointer to an array pointing to first array of the 2D.
IV. Its value is same as &arr[0][0] but its behavior is not.
V. So &arr for a 2-D array is matrix pointer or we can say it is a pointer to array of arrays pointing to the complete matrix.

So finally we can say for 2-D array

Int arr[4][5];

I. arr is an alias of address of first row or we can say it is a pointer to array of 5 ints which is currently pointing to first array.
II. arr[0] is an alias of address of first element of first row (&arr[0][0]) or we can say it is a pointer to first element of 0^{th} row.
III. Similarly arr[i] is an alias of address of first element of ith row.(&arr[i][0])
IV. &arr is an alias of address of the complete matrix of size 4*5 elements or its is pointer to a 2D array

Declaring pointer to array

I. int (*p)[5] – This creates an pointer variable p which points to array of 5 integers.
II. int *p[5] is not the same as above. This means an array of integer pointers.
III. Round Brackets are important.

Passing 2-D arrays into a function.

I. Like in a 1-D array when we pass it to function we are passing pointer to an element.
II. Similarly for a 2-D array we are passing pointer to an array of size – number of columns.
III. So a function declaration could either look like
    I. void accept2D(int arr[][5])
    II. void accept2D(int (*arr)[5])

Array of strings!

I. We simulated a string by a 1-D character array.
II. Similarly we can simulate a list of strings by 2-D character array.
III. char stringlist[10][100];
IV. Above can store max 10 strings each of maxlength 100.
V. And each string can be accessed by strlinglist[i].

Initializing array of strings!

Lets see an example.

I. Given a list of strings and word S. Check if S exists in the list or not.

How about N-D array?

I. Declaring N-D array
II. Accessing N-D array
III. Storage of N-D array
IV. Initializing N-D array
V. Passing N-D array to a function

Time to try?

I. Write a program to create a matrix of alternate rectangles of O and X

For N = 5;

ooooo

oxxxo

oxoxo

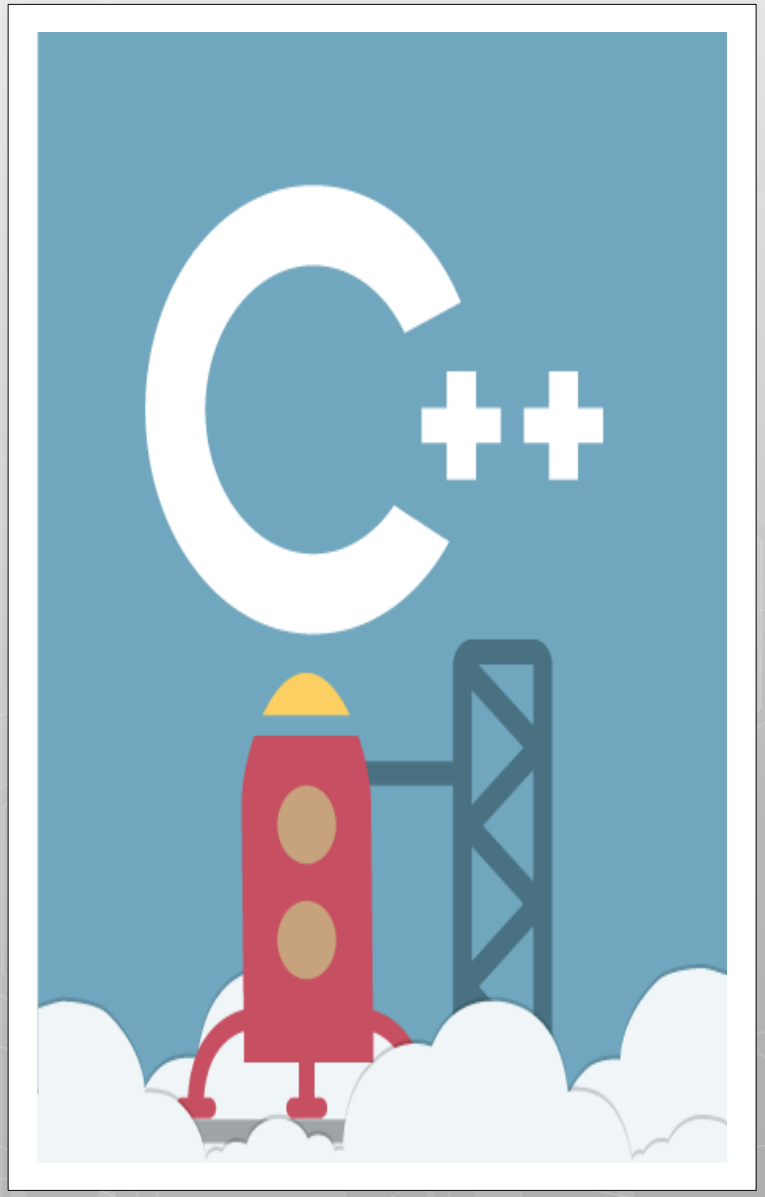
oxxxo

ooooo

II. Read N words and sort them lexicographically.

What is next class about?

I. Recursion



Thank You!

Anushray Gupta

anushray@codingblocks.com
+91-9555567876