Computer CIA – 1

Register Number: 1740256

**Date:** 5/12/2017 Name: Jeevan Koshy

--------------------------------------------------------------------------------------------

**Question 1: Write a short note on 2-dimensional arrays**An array is a collection of homogenous elements stored in continuous location with the same name.

**Syntax** – datatype arrayname[size]; (Range is till n-1)

Where n is the number of elements in the array.

A two-dimensional array is a logical data structure that is useful in describing an object that is physically two dimensional. It is generally a matrix – and also is useful in organizing a set of values that are dependent on 2 inputs. This array clearly illustrates the differences between a logical and a physical view of data.

The 2-dimensional array has finite number of rows and finite number of columns. The declaration form of 2-d array is –

**Syntax** - datatype arrayname[rowsize][columnsize];

For example –

int a[3][5];

Here, in the above example an array ‘a’ is defined with an integer data type having 3 rows and 5 columns.

To implement a two-dimensional array, it is necessary to develop a method of ordering it’s elements in a linear fashion and of transforming a two-dimensional reference to the linear representation. Therefore, an array is a linear data structure.

With 2-d arrays, we can do perform operations on matrices such as-

* Addition & subtraction of matrices.
* To check whether a matrix is identity or not
* Multiplication of matrices
* Transpose of a matrix

Every element in an array matrix is identified by an element name of the form matrix[i][j]; where ‘i’ and ‘j’ are the subscripts that uniquely identify each element in the matrix.

In case of a two – dimensional array, during declaration – the maximum number of column should be specified for processing all array elements. The implementation of the array stores all the elements in a single contiguous block of memory. In memory, the array is arranged with the elements of the rightmost index next to each other. In other words, matrix[1][1] comes right before matrix[1][2] in memory.  
**For scanning a 2-d array** -

for(i=0; i<row; i++)

{

for(j=0;j<col;j++)

{

scanf("%d",&arr[i][j]);

}

}  
**For Printing a 2-d array** -  
  
for(i=0;i<row; i++)

{

for(j=0;j<col;j++)

{

printf("%d \t",arr[i][j]);

}printf("\n"); }

--------------------------------------------------------------------------------------------------------------------------------------

**Question 2: Write a short note on merging of arrays**

Merging of arrays are basically bringing together 2 separate arrays sorted arrays to form one array. If the arrays are unsorted, we have to sort them and then merge them to finally form one array.

To merge the second array to the first array, the programmer has to first accept elements in the 1st and 2nd array. He then has to shift each element from the second array to the first array by increasing the length and then taking each element.

Since taking each element would be lengthy, we use a for loop to shift the elements and the complier will do the rest. Merge sort is based on Divide and conquer method. It takes the list to be sorted and divide it in half to create two unsorted lists. The two unsorted lists are then sorted and merged to get a sorted list.

1. Divide the input which we have to sort into two parts in the middle. Call it the left part and right part.

Example: Say the input is -10 32 45 -78 91 1 0 -16 then the left part will be -10 32 45 -78,

and the right part will be 91 1 0 -16.

2. Sort each of them separately. Note that here sort does not mean to sort it using some other method. We use the same function recursively.

3. Then merge the two sorted parts.

--------------------------------------------------------------------------------------------

**Question 3: C Program to use 2-d arrays**

#include<stdio.h> /\* header files \*/

#include<conio.h>

main()

{ /\* beginning of main \*/

int i,row,j,col,arr[20][20],rowch,colch; /\* variables declared \*/

char opt,ch;

printf("Program to extract an element from a matrix \n");

printf("===============================================");

do

{

BEG:

printf("\n Enter the order of rows: ");

scanf("%d",&row);

if(row<=0 || row>20)

{

printf("\n The matrix can have only 20 rows and cannot be negative");

goto BEG;

}

HI:

printf("\n Enter the order of columns: ");

scanf("%d",&col);

if(col<=0 || col>20)

{

printf("\n The matrix can have only 20 columns and cannot be negative");

goto HI;

}

printf("\n ----------------------------------------------");

printf("\n Enter the matrix: ");

printf("\n ---------------------------------------------- \n"); /\* User Entering the matrix \*/

for(i=0;i<row;i++)

{

for(j=0;j<col;j++)

{

scanf("%d",&arr[i][j]);

}

}

printf("\n ----------------------------------------------"); /\* printing the matrix \*/

START:

printf("\n The matrix is: \n");

for(i=0;i<row;i++)

{

for(j=0;j<col;j++)

{

printf("%d \t",arr[i][j]);

}

printf("\n");

}

printf("\n ----------------------------------------------");

ABC:

printf("\n Enter the row number of the element you want to extract: ");

scanf("\n %d",&rowch);

if(rowch==0 || rowch<0)

{

printf("\n The number of rows start from 1 and can only be positive");

goto ABC;

}

if(rowch<=row)

{

DEF:

printf("\n Enter the column number of the element you want to extract: ");

scanf("\n %d",&colch);

if(colch==0 || colch<0)

{

printf("\n The number of columns start from 1 and can only be ");

goto DEF;

}

if(colch<=col)

{

printf("\n ----------------------------------------------");

printf("\n The element in the %d row and the %d column is",rowch,colch); /\* asking which element user wants \*/

printf("\n %d",arr[rowch-1][colch-1]); /\* -1 as index starts from 0 \*/

printf("\n Do you want to continue?(Y/N): ");

scanf(" %c",&opt);

if(opt=='N'||opt=='n')

{

goto END;

}

printf("\n Do you want to work on the same matrix?(Y/N): ");

scanf(" %c",&ch);

if(ch=='Y'||ch=='y')

{

goto START;

}

else

{

goto BEG;

}

}

else

{

printf("\n Please enter a valid column number!");

goto DEF;

}

}

else

{

printf("\n Please enter a valid row number!");

goto ABC;

}

}while(opt=='Y'||opt=='y');

END:

printf("===============================================\n");

printf("\n\n END OF PROGRAM");

getch();

return 0;

}