CL101 INTRODUCTION TO COMPUTING

LAB 08

ARRAYS IN C

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

ARRAY

An array is a collection of data items of the same type.

SIGNIFICANCE OF ARRAY

Programming problems can be solved efficiently by grouping the data items together in main memory than allocating an individual memory cell for each variable.
For Example: A program that processes exam scores for a class, would be easier to write if all the scores were stored in one area of memory and were able to be accessed as a group. C allows a programmer to group such related data items together into a single composite data structure called array.

ONE-DIMENSIONAL ARRAYS

In one-dimensional array, the components are arranged in the form of a list.

SYNTAX:

```
element-type aname [ size ];  /* uninitialized */
element-type aname [ size ] = { initialization list }; /* initialized */
```

INTERPRETATION:

- The general uninitialized array declaration allocates storage space for array aname consisting of size memory cells.
- Each memory cell can store one data item whose data type is specified by element-type (i.e., double, int , or char).
- The individual array elements are referenced by the subscripted variables aname [0] , aname [1] , . . , aname [size −1] .
- A constant expression of type int is used to specify an array's size . In the initialized array declaration shown, the size shown in brackets is optional since the array's size can also be indicated by the length of the initialization list .
- The initialization list consists of constant expressions of the appropriate element-type separated by commas.
- Element 0 of the array being initialized is set to the first entry in the initialization list , element 1 to the second, and so forth.

MEMORY REPRESENTATION

All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.

double x[5] = { 5.0, 2.0, 3.0, 1.0, -4.5};

Array x

x[0]	x[1]	x[2]	x[3]	x[4]

x[0]	x[1]	x[2]	x[3]	x[4]
5.0	2.0	3.0	1.0	-4.5

TWO-DIMENSIONAL ARRAYS

A two dimensional array is a collection of a fixed number of components arranged in rows and columns (that is, in two dimensions), wherein all components are of the same type.
Two-dimensional arrays are used to represent tables of data, matrices, and other two-dimensional objects.

SYNTAX:

element-type aname [$size_1$] [$size_2$]; /* uninitialized */

INTERPRETETION

- Allocates storage for a two-dimensional array (aname) with size1 rows and size2 columns.
- This array has size1*size2 elements, each of which must be referenced by specifying a row subscript (0 , 1 ,… size1-1) and a column subscript (0 , 1 ,…size2-1).
- Each array element contains a character value.

MEMORY REPRESENTATION

char x[3][3] = {{'X', '0', 'X'}, {'0', 'X', '0'}, {'0', 'X', 'X'}};

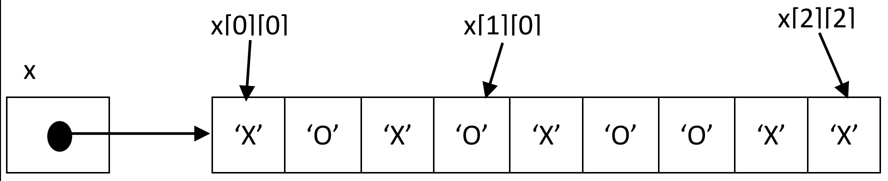
Array x

Row \ Column	0	1	2
0	x[0][0]	x[0][1]	x[0][2]
1	x[1][0]	x[1][1]	x[1][2]
2	x[2][0]	x[2][1]	x[2][2]

Row \ Column	0	1	2
0	X	O	X
1	O	X	O
2	O	X	X

x[1][2]

Because memory is addressed linearly, a better representation is like:



USES

- Storing a table of data (not the only way).
- Any kind of matrix processing, as a 2D array really is a matrix.

MULTIDIMENSIONAL ARRAYS

Multidimensional array is a collection of a fixed number of elements (called components) arranged in n dimensions (n>=1).

SYNTAX:

element-type aname [size 1] [size 2] ... [size n]; /* storage allocation */

INTERPRETATION:

- Allocates storage space for an array aname consisting of size 1 × size 2 × ... × size n memory cells.
- Each memory cell can store one data item whose data type is specified by element-type . The individual array elements are referenced by the subscripted variables aname [0][0] ... [0] through aname [size 1 −1][size 2 −1] ...[size n −1] .
- An integer constant expression is used to specify each size i .

USES

With input data on temperatures referenced by day, city, county, and state, day would be the first dimension, city would be the second dimension, county would be the third dimension, and state would be the fourth dimension of the array. In any case, any temperature could be found as long as the day, the city, the county, and the state are known. A multidimensional array allows the programmer to use one array for all the data.

STORING A STRING IN AN ARRAY OF CHARACTERS

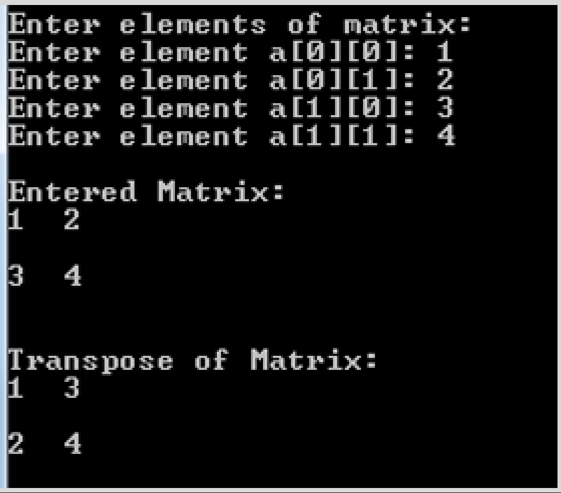
The previous syntax display shows that individual characters can be stored in an array by writing each character in the initialization list. If the list is long, this can be done more easily by using a string instead of an initialization list.

char vowels[] = "Hello World";

EXAMPLE (TWO-DIMENSIONAL ARRAY)

```
//Program to display the transpose of given 2x2 matrix(2D array)

#include <stdio.h>

int main()
{

          int matrix[2][2], transpose[2][2], row, col;

          // Storing elements of the matrix
          printf("\nEnter elements of matrix:\n");
          for(row=0; row<2; row++)
            for(col=0; col<2; col++)
            {
              printf("Enter element a[%d][%d]: ",row,col);
              scanf("%d", &matrix[row][col]);
            }

          // Displaying the matrix[][]
          printf("\nEntered Matrix: \n");
          for(row=0; row<2; row++)
```

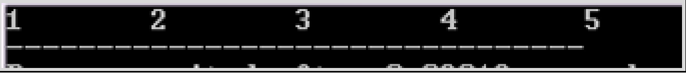
```
            for(col=0; col<2; col++)
            {
               printf("%d  ", matrix[row][col]);
               if (col == 1)
                  printf("\n\n");
            }

         // Finding the transpose of matrix
         for(row=0; row<2; row++)
            for(col=0; col<2; col++)
            {
               transpose[col][row] = matrix[row][col];
            }

         // Displaying the transpose of matrix
         printf("\nTranspose of Matrix:\n");
         for(row=0; row<2; row++)
            for(col=0; col<2; col++)
            {
               printf("%d  ",transpose[row][col]);
               if(col==1)
                  printf("\n\n");
            }

   return 0;
}
```



```
Enter elements of matrix:
Enter element a[0][0]: 1
Enter element a[0][1]: 2
Enter element a[1][0]: 3
Enter element a[1][1]: 4

Entered Matrix:
1  2

3  4


Transpose of Matrix:
1  3

2  4
```

EXAMPLE (PASSING 1D ARRAY AS ARGUMENT)

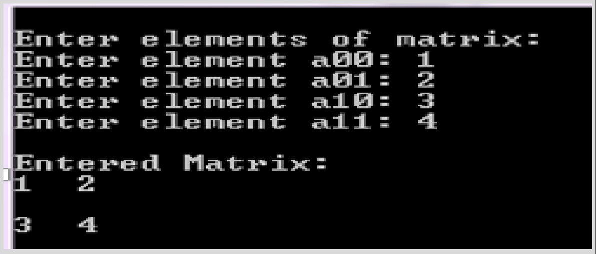
```
#include<stdio.h>

void func(int a[],int size)
{
        int i;
        for(i=0;i<size;i++)
        {
                printf("%d\t",a[i]);
        }
}

int main()
{
        int a[5]= {1,2,3,4,5};
        func(a,5);
}
```



```
1       2       3       4       5
--------------------------------
```

EXAMPLE (PASSING 2D ARRAY AS ARGUMENT)

```c
#include <stdio.h>

void Display_Matrix(int a[][2], int m, int n)
{
   int row ,col;

   printf("\nEntered Matrix: \n");
   for(row=0; row<m; row++)
      for(col=0; col<n; col++)
      {
         printf("%d  ", a[row][col]);
         if (col == 1)
         printf("\n\n");
      }
}

int main()
{
   int a[2][2],row, col;

   // Storing elements of the matrix
   printf("\nEnter elements of matrix:\n");
   for(row=0; row<2; row++)
      for(col=0; col<2; col++)
      {
         printf("Enter element a[%d][%d]:",row,col);
         scanf("%d", &a[row][col]);
      }

   Display_Matrix(a,2,2);
   return 0;
}
```



```
Enter elements of matrix:
Enter element a00: 1
Enter element a01: 2
Enter element a10: 3
Enter element a11: 4

Entered Matrix:
1   2

3   4
```

LAB 08 EXERCISES

INSTRUCTIONS:

NOTE: Violation of any of the following instructions may lead to the cancellation of your submission.

1) Create a folder and name it by your student id (k16-1234).
2) Paste the .c file for each question with the names such as Q1.c, Q2.c and so on into that folder.
3) Submit the zipped folder on slate.

NOTE: USE MODULAR PROGRAMMING APPROACH FOR SOLVING ALL OF THE FOLLOWING PROBLEMS.

QUESTION#1

Write a program containing a function named 'ArraySum' that computes and returns the sum of all elements in an array, where the array and its size are given as parameters. Use appropriate parameters and return type.

QUESTION#2

Write a program containing three functions named 'Input_Matrix', 'Transpose_Matrix' and 'Display_Transpose' to enter, calculate and display the transpose of a two dimensional matrix. The dimensions of the matrix must be taken as input. Use appropriate parameters and return type. The main function must call only one function to start the execution.
HINT: Call the next function inside the previous one.

QUESTION#3

Use a single-subscripted array passed to a function to solve the following problem. Read in 20 numbers, each of which is between 10 and 100, inclusive. As each number is read, print it only if it's not a duplicate of a number already read. Provide for the "worst case" in which all 20 numbers are different.

QUESTION#4

An instructor has 30 students in her class. Each student is identified by a number from 1 to 30. Grades are stored in a one-dimensional array. The instructor would like to enter a student number and have the student's test score printed on the monitor. Develop a program to output the needed information.

QUESTION#5

An instructor has a class of 25 students. Each student is identified by a number from 1 to 25. All tests are stored in a two-dimensional array, with each column containing the grades for each test. The instructor would like to enter the student number and the test number and have the grade for that test printed on the monitor. Develop a program to output the needed information.

QUESTION#6

N numbers are entered from the keyboard into an array. The number to be searched is entered through the keyboard by the user. Write a program to find if the number to be searched is present in the array and if it is present, display the number of times it appears in the array along with the position.

QUESTION#7

Write a program to print the reverse of an entered array. Your program must contain only one array.

QUESTION#8

Write a program to insert an element into an array.

QUESTION#9
Write a program to sort all the elements of an array (in ascending order and descending order).

QUESTION#10
Write a program to copy the content of one array into another in reverse order.

QUESTION#11
Write a program that takes input in two 2-D array and add these two matrices and give addition of these two matrices in 3rd matrix

QUESTION#12
Write a program to delete a given number from array.

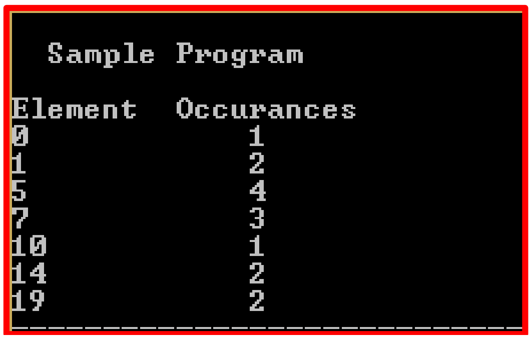
QUESTION#13
Use a double-subscripted array to solve the following problem. A company has four salespeople (1 to 4) who sell five different products (1 to 5). Once a day, each salesperson passes in a slip for each different type of product sold. Each slip contains:
a) The salesperson number
b) The product number
c) The total dollar value of that product sold that day
Thus, each salesperson passes in between 0 and 5 sales slips per day. Assume that the information from all of the slips for last month is available. Write a program that will read all this information for last month's sales and summarize the total sales by salesperson by product. All totals should be stored in the double-subscripted array `sales`. After processing all the information for last month, print the results in tabular format with each column representing a particular salesperson and each row representing a particular product. Cross total each row to get the total sales of each product for last month; cross total each column to get the total sales by salesperson for last month. Your tabular printout should include these cross totals to the right of the totaled rows and to the bottom of the totaled columns.

QUESTION#14
A small airline has just purchased a computer for its new automated reservations system. The president has asked you to program the new system. You'll write a program to assign seats on each flight of the airline's only plane (capacity: 10 seats). Your program should display the following menu of alternatives:
Please type 1 for "first class"
Please type 2 for "economy"
If the person types 1, then your program should assign a seat in the first class section (seats 1–5).
If the person types 2, then your program should assign a seat in the economy section (seats 6–10).
Your program should then print a boarding pass indicating the person's seat number and whether it's in the first class or economy section of the plane.
Use a single-subscripted array to represent the seating chart of the plane. Initialize all the elements of the array to 0 to indicate that all seats are empty. As each seat is assigned, set the corresponding element of the array to 1 to indicate that the seat is no longer available.
Your program should, of course, never assign a seat that has already been assigned. When the first class section is full, your program should ask the person if it's acceptable to be placed in the economy section (and vice versa). If yes, then make the appropriate seat assignment. If no, then print the message "Next flight leaves in 3 hours."

QUESTION#15

Write a program to take an array and find the number of occurrences each number had. The output should be something like this:

NOTE: Sort the array elements first and then calculate the frequency of each element. Use single for-loop to calculate the frequency of each element.



```
  Sample Program

Element  Occurances
0          1
1          2
5          4
7          3
10         1
14         2
19         2
-----------------------------
```

QUESTION#16

Suppose you have the following matrices:

$$M1 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix} \text{ and } M2 = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix}$$

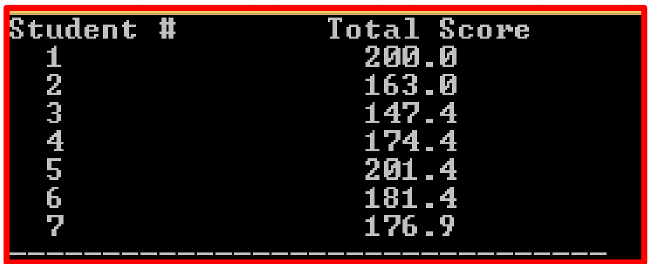
Write a C program that will calculate its product and print the resultant matrix.

$$M1 \times M2 = \begin{pmatrix} 70 & 80 & 90 \\ 158 & 184 & 210 \end{pmatrix}$$

QUESTION#17

Write a program that calculates the total score for students in a class. Suppose the scores are stored in a three-dimensional array named scores. The first index in scores refers to a student, the second refers to an exam, and the third refers to the part of the exam. Suppose there are 7 students, 5 exams, and each exam has two parts--the multiple-choice part and the programming part. So, scores[i][j][0] represents the score on the multiple-choice part for the i's student on the j's exam. Your program displays the total score for each student.

{{{7.5, 20.5}, {12, 22.5}, {22, 33.5}, {43, 21.5}, {15, 2.5}},
{{4.5, 21.5}, {12, 22.5}, {12, 34.5}, {12, 20.5}, {14, 9.5}},
{{5.5, 30.5}, {9.4, 2.5}, {13, 33.5}, {11, 23.5}, {16, 2.5}},
{{6.5, 23.5}, {9.4, 32.5}, {13, 34.5}, {11, 20.5}, {16, 7.5}},
{{8.5, 25.5}, {9.4, 52.5}, {13, 36.5}, {13, 24.5}, {16, 2.5}},
{{9.5, 20.5}, {9.4, 42.5}, {13, 31.5}, {12, 20.5}, {16, 6.5}},
{{1.5, 29.5}, {9.4, 22.5}, {19, 30.5}, {10, 30.5}, {19, 5.0}}};



```
Student #        Total Score
  1                200.0
  2                163.0
  3                147.4
  4                174.4
  5                201.4
  6                181.4
  7                176.9
-----------------------------------
```