CL1002
INTRODUCTION TO COMPUTING

LAB 08

ARRAYS IN C

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};

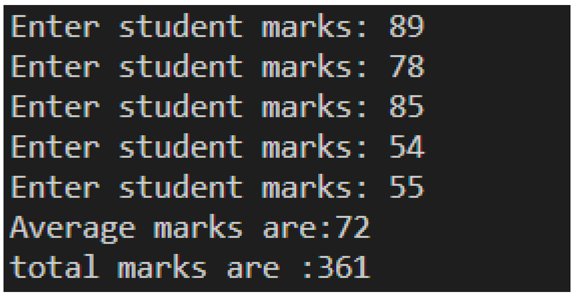
Array x

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

EXAMPLE (1D ARRAY)

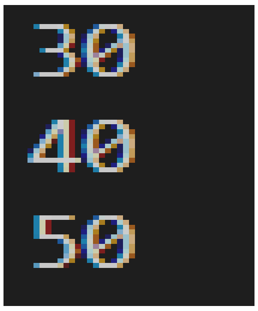
```
/* 1D Array 1st Example */

#include<stdio.h>
int main(){

        int avg, sum = 0;
        int i;
        int marks[5]; // array declaration

        for(i = 0; i<=4; i++){
                printf("Enter student marks: ");
                scanf("%d", &marks[i]); /* stores the data in array*/
                }
        for(i = 0; i<=4; i++)
                sum = sum + marks[i]; /* reading data from array */

                avg = sum/5;
                printf("Average marks are:%d\n", avg);
                printf("total marks are :%d\n", sum);
                return 0;

}
```



```
Enter student marks: 89
Enter student marks: 78
Enter student marks: 85
Enter student marks: 54
Enter student marks: 55
Average marks are:72
total marks are :361
```

Example 02:

```
#define size 5
#include <stdio.h>
int main(void)
{
int i;

/*
   int arrOfNumbers[5];
     arrOfNumbers[0] = 10;
     arrOfNumbers[1] = 20;
     arrOfNumbers[2] = 30;
     arrOfNumbers[3] = 40;
     arrOfNumbers[4] = 50;*/
     // 2nd alternattive
     //int arrOfNumbers[5] = {10,20,30,40,50};
     // 3rd alternative declaration
     //int arrOfNumbers[] = {10,20,30,40,50};
     // 4th alternative using macro
     int arrOfNumbers[size] = {10,20,30,40,50};
     for(i = 0; i < 5; i++)
     {
     /* The braces are not necessary; we use them to make the code
     clearer. */
     if(arrOfNumbers[i] > 20)
     printf("%d\n", arrOfNumbers[i]);
}
     return 0;}
```



```
30
40
50
```

Example 03
C does not check for index out of bound, it can be done by programmer itself.

```
#include <stdio.h>
int main(void)
{
    int i, j = 30, arrOfNumbers[3];
    for(i = 0; i < 4; i++)
            arrOfNumbers[i] = 100;
            printf("%d\n", j);
return 0;
} /* The program does not throw any error. It will print 30 once.
```

<u>Example 04</u>

```
#include <stdio.h>
int main(void)
{
    int i, arrOfNumbers[10] = {0};
    for(i = 0; i < 10; i++){
            arrOfNumbers[++i] = 20;
//          arrOfNumbers[i] = 20;
            printf("The elements of array are:\t%d\n",arrOfNumbers[i]);
    }


    return 0;
}
```

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'}};

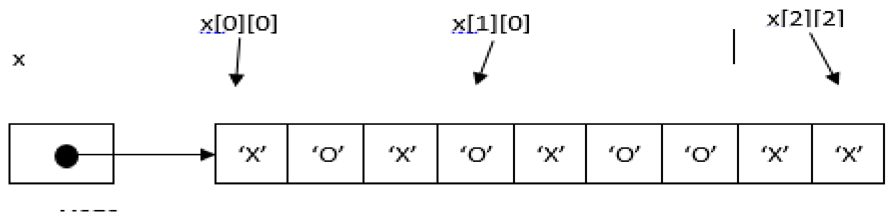
Array x

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:



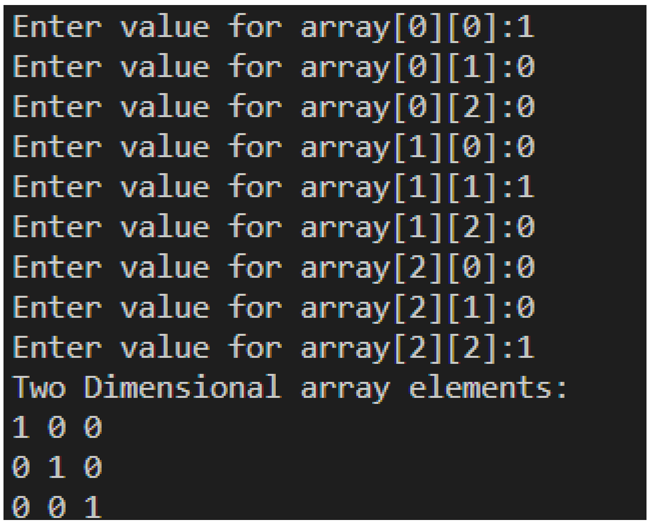
USES

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

Example 01:

```c
#include<stdio.h>
int main(){
   /* 2D array declaration*/
   int array[3][3];
   /*Counter variables for the loop*/
   int i, j;
   for(i=0; i<3; i++) {
      for(j=0;j<3;j++) {
         printf("Enter value for array[%d][%d]:", i, j);
         scanf("%d", &array[i][j]);
      }
   }
   //Displaying array elements
   printf("Two Dimensional array elements:\n");
   for(i=0; i<3; i++) {
      for(j=0;j<3;j++) {
         printf("%d ", array[i][j]);
         if(j==2){
            printf("\n");
         }
      }
   }
   return 0;
}
```

Output:



```
Enter value for array[0][0]:1
Enter value for array[0][1]:0
Enter value for array[0][2]:0
Enter value for array[1][0]:0
Enter value for array[1][1]:1
Enter value for array[1][2]:0
Enter value for array[2][0]:0
Enter value for array[2][1]:0
Enter value for array[2][2]:1
Two Dimensional array elements:
1 0 0
0 1 0
0 0 1
```

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.

Example:

```c
#include <stdio.h>

int main(void)
{
    // initializing the 3-dimensional array
    int x[2][3][2] = { { { 0, 1 }, { 2, 3 }, { 4, 5 } },
                       { { 6, 7 }, { 8, 9 }, { 10, 11 } } };

    // output each element's value
    for (int i = 0; i < 2; ++i) {
        for (int j = 0; j < 3; ++j) {
            for (int k = 0; k < 2; ++k) {
                printf("Element at x[%i][%i][%i] = %d\n", i, j, k, x[i][j][k]);
            }
        }
    }
    return (0);
}
```

Output:

```
Element at x[0][0][0] = 0
Element at x[0][0][1] = 1
Element at x[0][1][0] = 2
Element at x[0][1][1] = 3
Element at x[0][2][0] = 4
Element at x[0][2][1] = 5
Element at x[1][0][0] = 6
Element at x[1][0][1] = 7
Element at x[1][1][0] = 8
Element at x[1][1][1] = 9
Element at x[1][2][0] = 10
Element at x[1][2][1] = 11
```

Exercises

Task 01:

Create an array named alpha that is of char data type. User can input random characters into this array. Create a program that tells the user how many vowels and consonants has the user emtered. Also create two new arrays named vowels and consonant and store the vowels in vowels array, store the consonant in consonant array.

Task 02:

Write a program that takes a word from user and display it in reverse order.

Sample Output:

Enter name: Ali

Output: ilA

Task 03:

Suppose A, B, C are arrays of integers of size M, N, and (M + N) respectively. Your program should perform the following tasks:

- Take number of elements and elements from user input for array A and sort it in ascending order.
- Take number of elements and elements from user input for array A and sort it in ascending order.
- In array C, merge array A and array B in ascending order.
- Display array A and array be before sorting and after sorting. Display array C after merge.

Task 04:

Write a C program that declares an array Numbers of 50 components of type int. Initialize the array so that the first 25 components are divisible by 2, and the last 25 components are divisible by 3. Output the array so that 10 elements per line are printed.

Task 05:

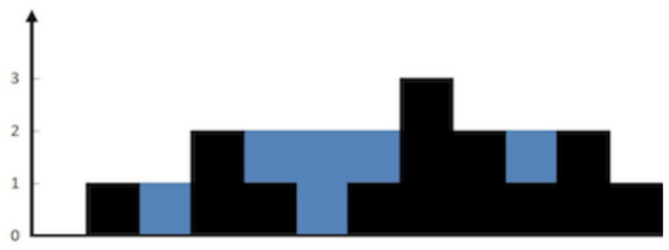
The teacher at a university needs help in grading a True/False test. The test contains the students' IDs and test answers in the form:
1022 TFTTF TFTT
where the student ID is 1022 and the answer to question 1 is True, the answer to question 2 is False, and so on. This student did not answer question 6 so a blank in written. The exam has 0 questions, and the class has 15 students. Each correct answer is awarded one points and no point are awarded for wrong answers and answers that are left blank. Write a program that processes the data. The output should be the student's ID, followed by the answers, followed by the test score, followed by the test grade. Assume the following grade scale:

- 90%–100% - A
- 80%–89.99% - B
- 70%–79.99% - C
- 60%–69.99% - D
- 0%–59.99% - F

Task 06:

Write a program that takes a non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it can trap after raining.

Example 1:



```
Input: height = [0,1,0,2,1,0,1,3,2,1,2,1]
Output: 6
Explanation: The above elevation map (black section) is represented by array
[0,1,0,2,1,0,1,3,2,1,2,1]. In this case, 6 units of rain water (blue section)
are being trapped.
```

Example 2:

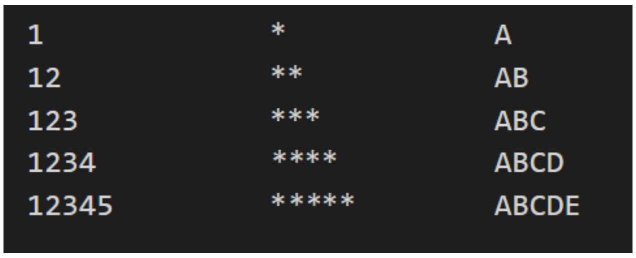
```
Input: height = [4,2,0,3,2,5]
Output: 9
```

Task 07:

Write a program that uses a two-dimensional array to store the highest and lowest temperatures for each month of the year. The program should output the average high, average low, and the highest and lowest temperatures for the year.

Task 08:

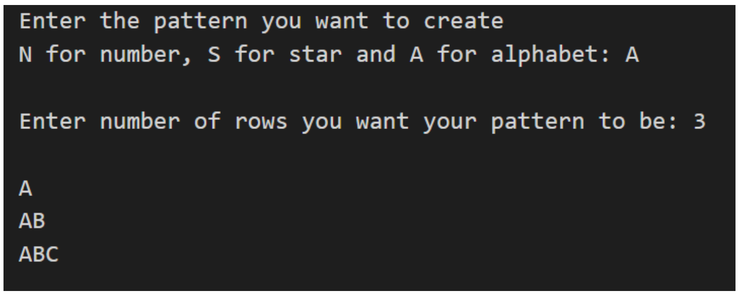
Create a program that take input from user (Number, Asterisk (*) and Alphabet) and design the following patterns based on user input.



```
1           *         A
12          **        AB
123         ***       ABC
1234        ****      ABCD
12345       *****     ABCDE
```

You should also take input for number of rows from user.

<u>Sample output:</u>

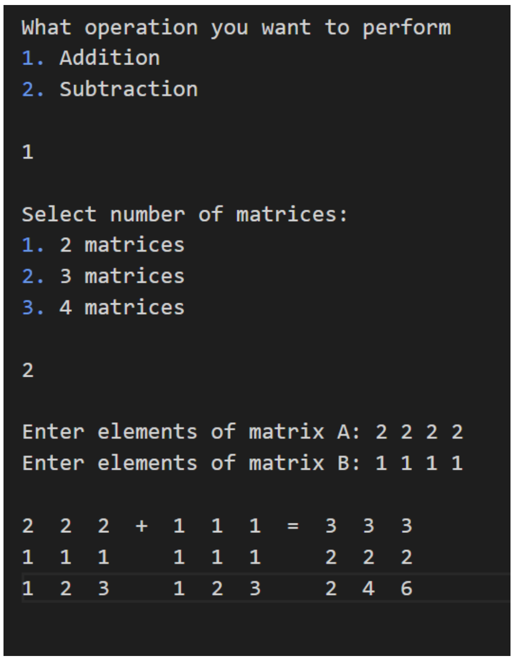


```
Enter the pattern you want to create
N for number, S for star and A for alphabet: A

Enter number of rows you want your pattern to be: 3

A
AB
ABC
```

Task 09:

Create a program which adds or subtracts matrices upon user desire. It can add/subtract 2, 3 and 4 3x3 matrices together. Write a program that takes user input:

- the operation the user wants to perform (add/subtract)
- how many matrices the user wants to add/subtract (2, 3 or 4)
- ask user to enter elements of all the matrices
- print the result in the following format:



```
What operation you want to perform
1. Addition
2. Subtraction

1

Select number of matrices:
1. 2 matrices
2. 3 matrices
3. 4 matrices

2

Enter elements of matrix A: 2 2 2 2
Enter elements of matrix B: 1 1 1 1

2  2  2  +  1  1  1  =  3  3  3
1  1  1     1  1  1     2  2  2
1  2  3     1  2  3     2  4  6
```

Also check for valid input. A user cannot enter other choices except for the given choices. If he selects any other option, it should print an error message.