Fundamentals of Programming CCS1063/CSE1062

Lecture 9 –**Arrays**

Professor Noel Fernando

Fundamentals of Programming CCS1063/CSE1062

Lecture 9 –**Arrays**

Professor Noel Fernando

Arrays

- Consider a situation in which we have 20 students in a class, and we have been asked to write a program that reads and prints the marks of all the 20 students.
- In this program, we will need 20 integer variables with different names, as shown in Fig. 1.1

Arrays

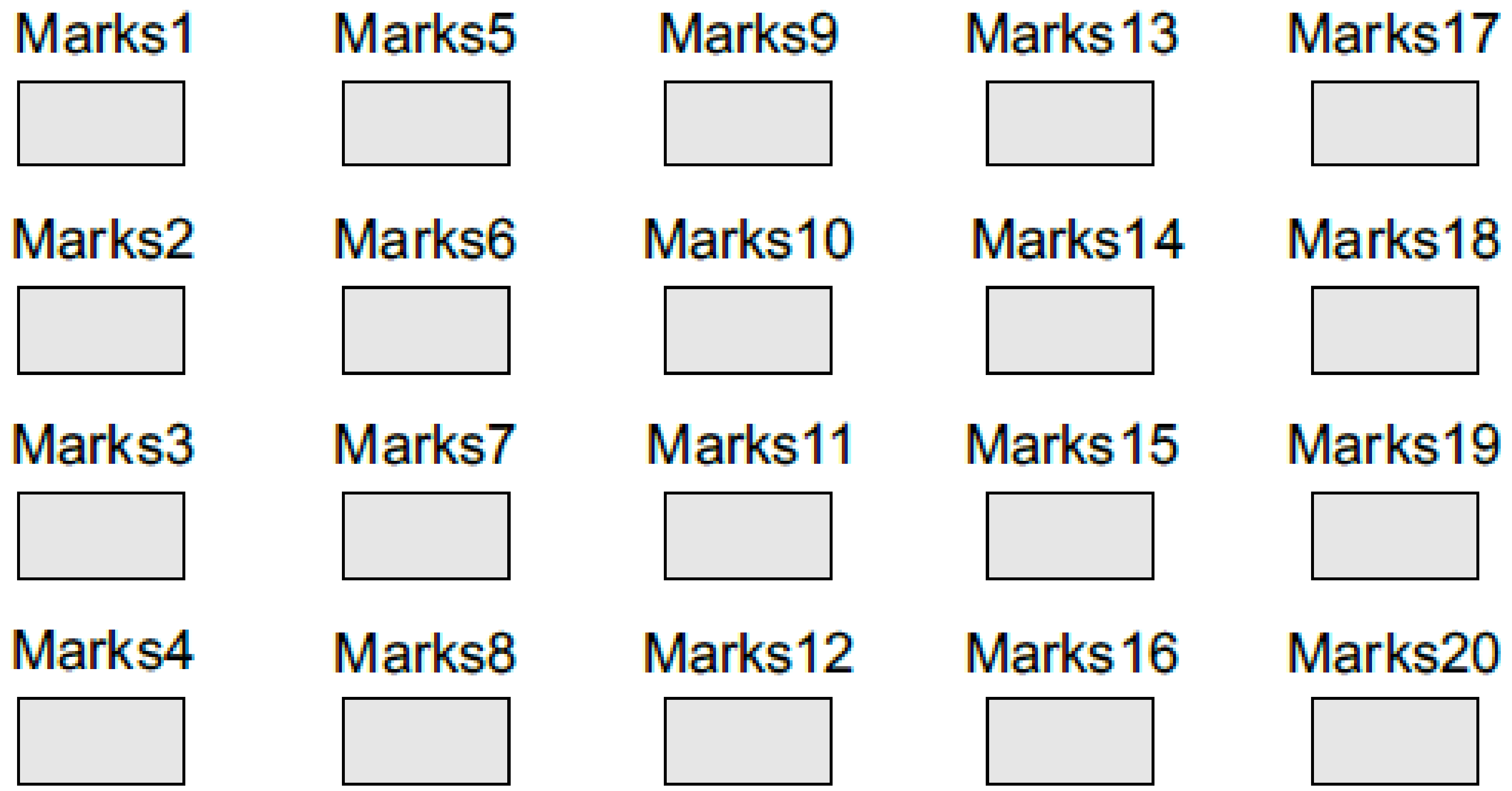


Fig 1.1

Arrays

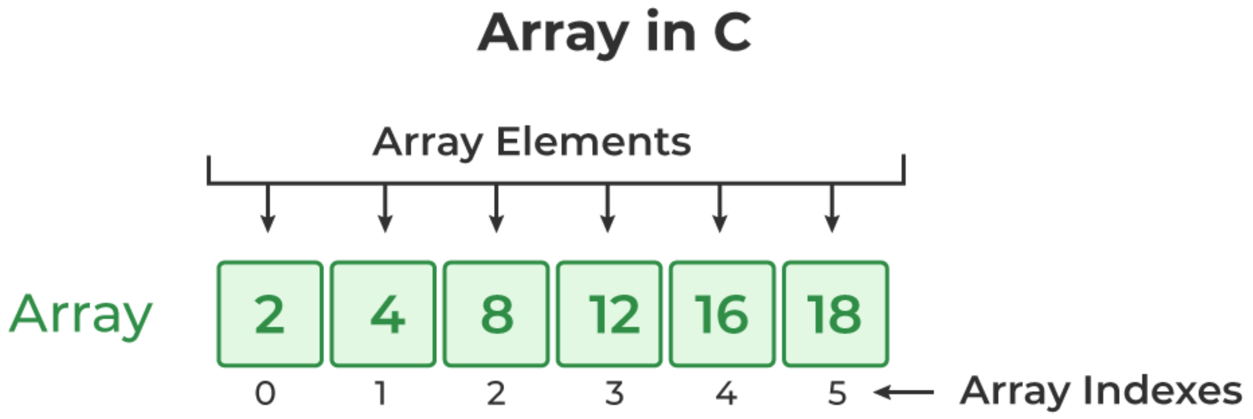
- If it is just a matter of 20 variables, then it might be acceptable for the user to follow this approach.
- But would it be possible to follow this approach if we have to read and print the marks of students,
- In the entire course (say 100 students)
- In the entire college (say 500 students)
- In the entire university (say 10,000 students)
- The answer is no, definitely not! To process a large amount of data,
- we need a data structure known as *array*.

Arrays

- An array is a collection of similar data elements.
- These data elements have the same data type.
- The elements of the array are stored in consecutive memory locations and are referenced by an index (also known as the *subscript*).
- The subscript is an ordinal number which is used to identify an element of the array

Arrays

- **Array in C** is one of the most used data structures in C programming.
- It is a simple and fast way of storing multiple values under a single name.
- **What is Array in C?**
- An array in C is a fixed-size collection of similar data items stored in contiguous memory locations.
- It can be used to store the collection of primitive data types such as int, char, float, etc.,
- and also derived and user-defined data types such as pointers, structures, etc.



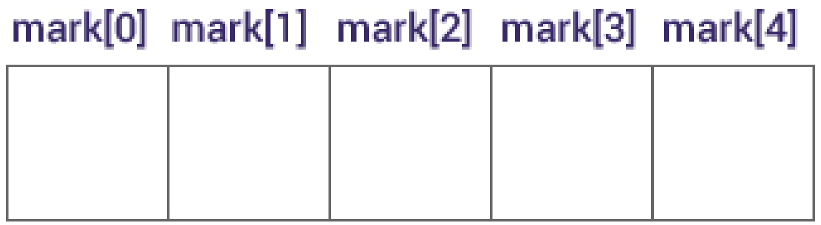
C Array Declaration

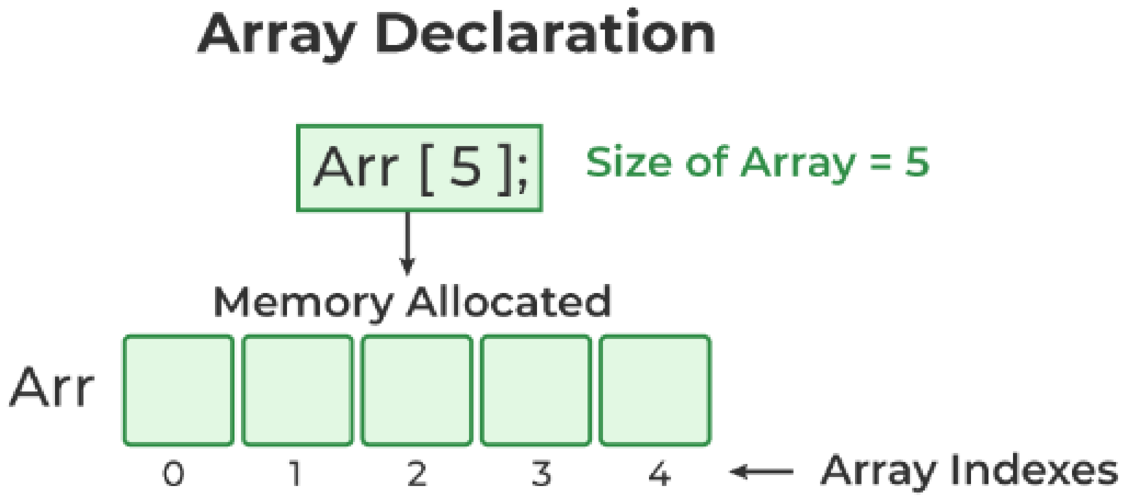
- **Syntax of Array Declaration**
- `data_type array_name [size];`
- or
- `data_type array_name [size1] [size2]...[sizeN];`
- where N is the number of dimensions.

Declaring an Array…

data_type array_name[array_size]

```
float Marks[5];
```





Write a C Program to illustrate the array declaration for 5 integer values and 5 Characters

Write a C Program to illustrate the array declaration for 5 integer values and 5 Characters

```
// C Program to illustrate the array declaration
#include <stdio.h>
int main()
{
	// declaring array of integers
	int arr_int[5];
	// declaring array of characters
	char arr_char[5];
	return 0;
}
```

Initializing Arrays

- It is possible to initialize an array during declaration

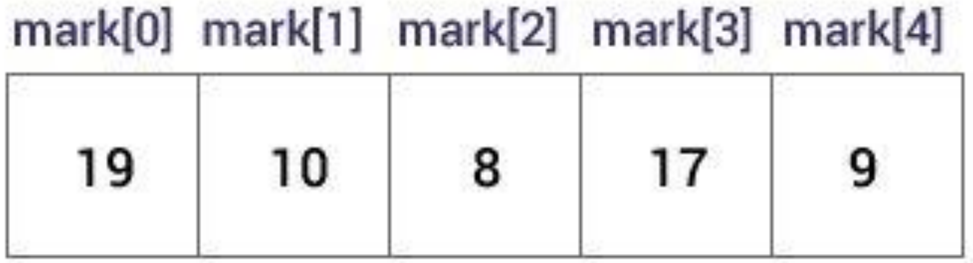
```
int mark[5] = {19, 10, 8, 17, 9};
```

- How to Change Value of Array elements after initialization
- Make the value of the third element to -1

```
mark[2] = -1;
```

- Make the value of the fifth element to 0

```
mark[4] = 0;
```



mark[0]	mark[1]	mark[2]	mark[3]	mark[4]
19	10	8	17	9

Write the contents of the array after changing the values ?

Question

1. Write a C Program to take 5 values from the user and store them in an array
2. Print the elements stored in the Array.

Answer

```
include <stdio.h>
 int main() {
int values[5];
printf("Enter 5 integers: ");
 // taking input and storing it in
an array
for(int i = 0; i < 5; ++i) {
 scanf("%d", &values[i]);
}
// printing elements of an array
for(int i = 0; i < 5; ++i)
 { printf("%d\n", values[i]);
 }
return 0;
 }
```

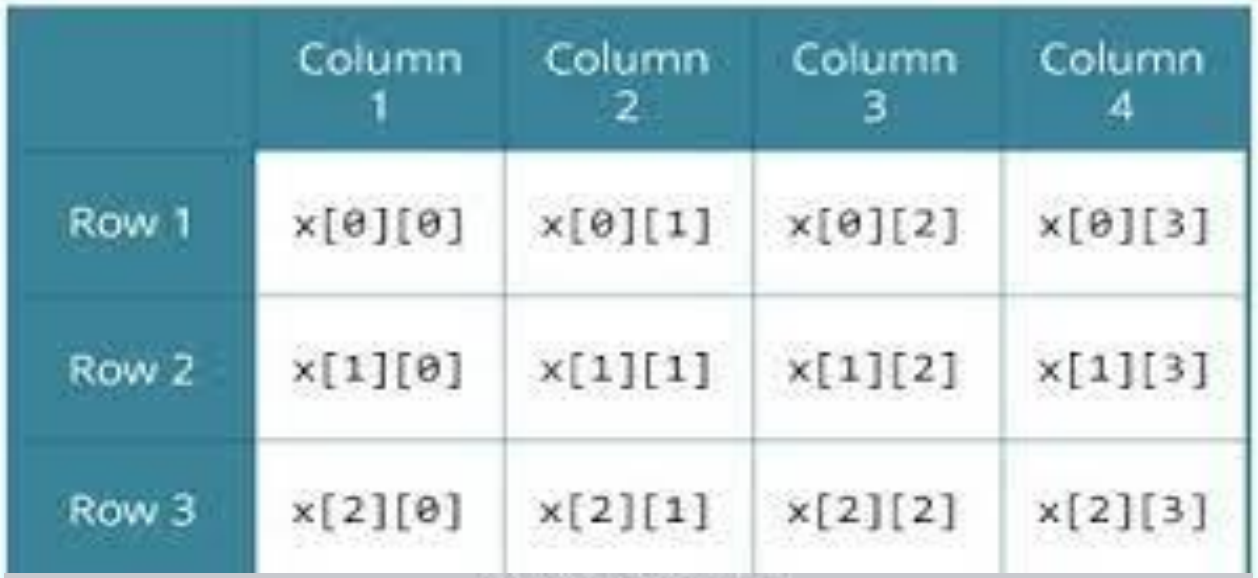
Question : Write a C Program to find the average of n numbers using arrays

Write a C Program to find the average of n numbers using arrays

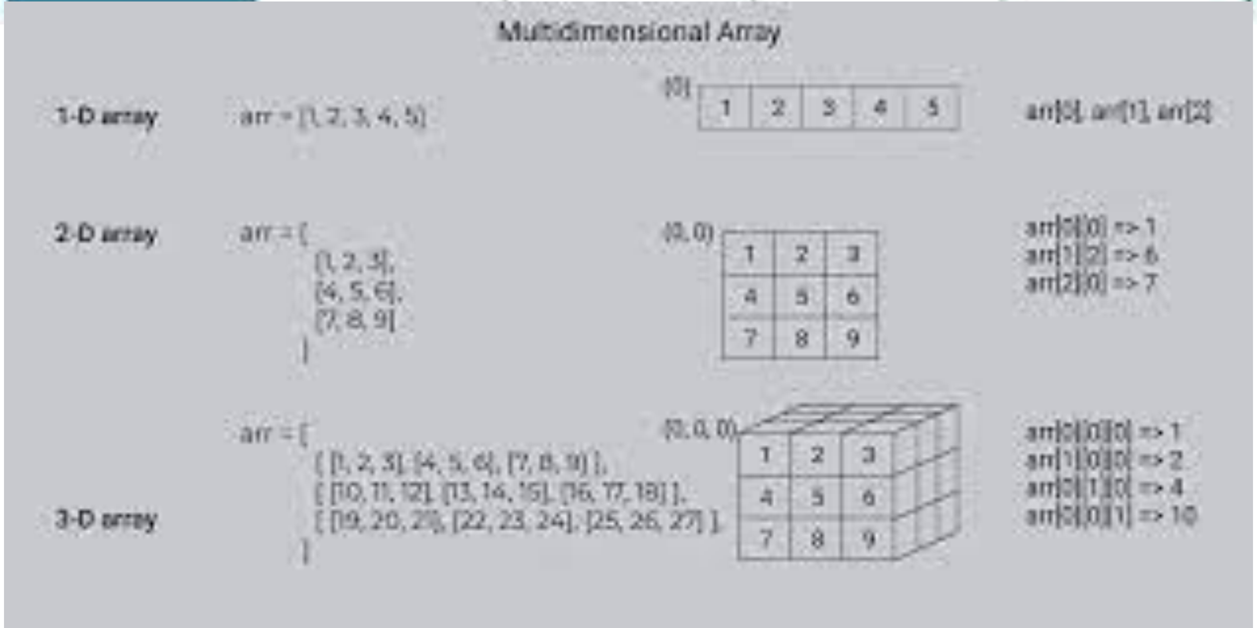
```
include <stdio.h>
int main() {
int marks[10], i, n, sum = 0;
 double average; printf("Enter
number of elements: ");
scanf("%d", &n);
for(i=0; i < n; ++i)
{ printf("Enter number%d: ",i+1);
scanf("%d", &marks[i]);
// adding integers entered by the
user to the sum variable
 sum += marks[i]; }
 // explicitly convert sum to
double
// then calculate average
 average = (double) sum / n;
 printf("Average = %.2lf",
average);
return 0;
}
```

Multidimensional Arrays

- if you want to store data as a tabular form, you need to use **multidimensional arrays**.
- A multidimensional array is basically an array of arrays.



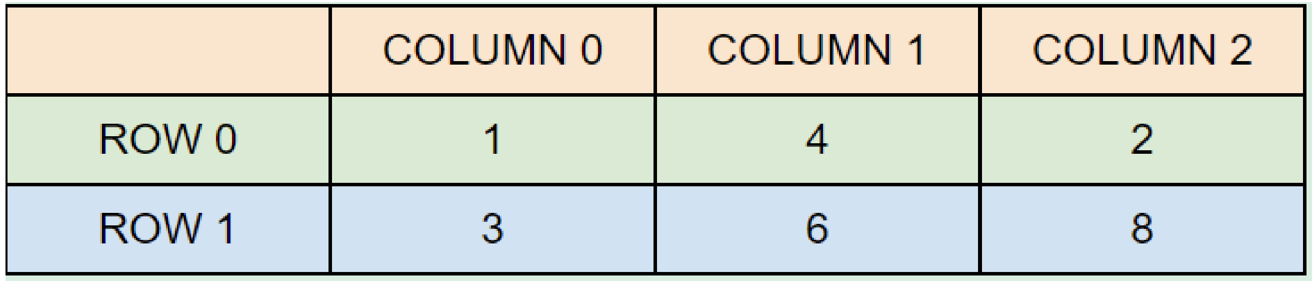
	Column 1	Column 2	Column 3	Column 4
Row 1	x[0][0]	x[0][1]	x[0][2]	x[0][3]
Row 2	x[1][0]	x[1][1]	x[1][2]	x[1][3]
Row 3	x[2][0]	x[2][1]	x[2][2]	x[2][3]



```
arr = [1, 2, 3, 4, 5]
arr[0], arr[1], arr[2]

arr = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
arr[0][0] => 1
arr[1][2] => 6
arr[2][0] => 7

arr = [
    [ [1, 2, 3], [4, 5, 6], [7, 8, 9] ],
    [ [10, 11, 12], [13, 14, 15], [16, 17, 18] ],
    [ [19, 20, 21], [22, 23, 24], [25, 26, 27] ]
]
arr[0][0][0] => 1
arr[1][0][0] => 2
arr[0][1][0] => 4
arr[0][0][1] => 10
```

Two-Dimensional Arrays

- A 2D array is also known as a matrix (a table of rows and columns).
- To create a 2D array of integers, take a look at the following example:
- int matrix[2][3] = { {1, 4, 2}, {3, 6, 8} };



	COLUMN 0	COLUMN 1	COLUMN 2
ROW 0	1	4	2
ROW 1	3	6	8

Questions

- Consider the following array
- matrix[2][3] = { {1, 4, 2}, {3, 6, 8} };
- Write a C code replace the first element “1” with “9”

- Consider the following array
- matrix[2][3] = { {1, 4, 2}, {3, 6, 8} };
- Write a C code print the matrix contents using arrays and for next loops.

Answers

- Change Elements in a 2D Array
- int matrix[2][3] = { {1, 4, 2}, {3, 6, 8} };
  matrix[0][0] = 9;
  printf("%d", matrix[0][0]);
- // Now outputs 9 instead of 1

- Loop Through a 2D Array

```
int matrix[2][3] = { {1, 4, 2},
{3, 6, 8} };

int i, j;
for (i = 0; i < 2; i++) {
  for (j = 0; j < 3; j++) {
    printf("%d\n", matrix[i][j]);
  }
}
```

Strings

- Strings are used for storing text/characters.
- For example, "Hello World" is a string of characters.
- Unlike many other programming languages, C does not have a **String type** to easily create string variables.
- you must use the char type and create an array of characters to make a string in C:

- E.g 1

```
char greetings[] = "Hello World!";
#include <stdio.h>
int main() {
  char greetings[] = "Hello World!";
  printf("%s", greetings);
   return 0;
}
```

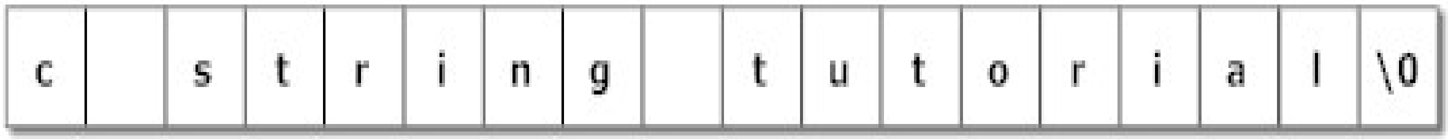
E.g. 2

```
#include <stdio.h>
int main() {
  char greetings[] = "Hello World!";
  printf("%c", greetings[0]);
   return 0;
}
```

Strings

- In C programming, a string is a sequence of characters terminated with a null character /0.
- For example:

```
char c[] = "c string tutorial";
```

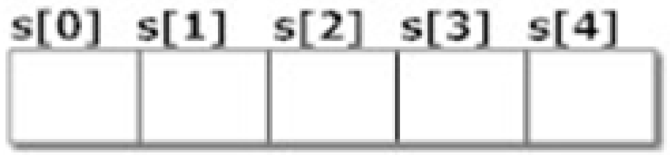


- When the compiler encounters a sequence of characters enclosed in the double quotation marks, it appends a null character \0 at the end by default.

Declaring strings…

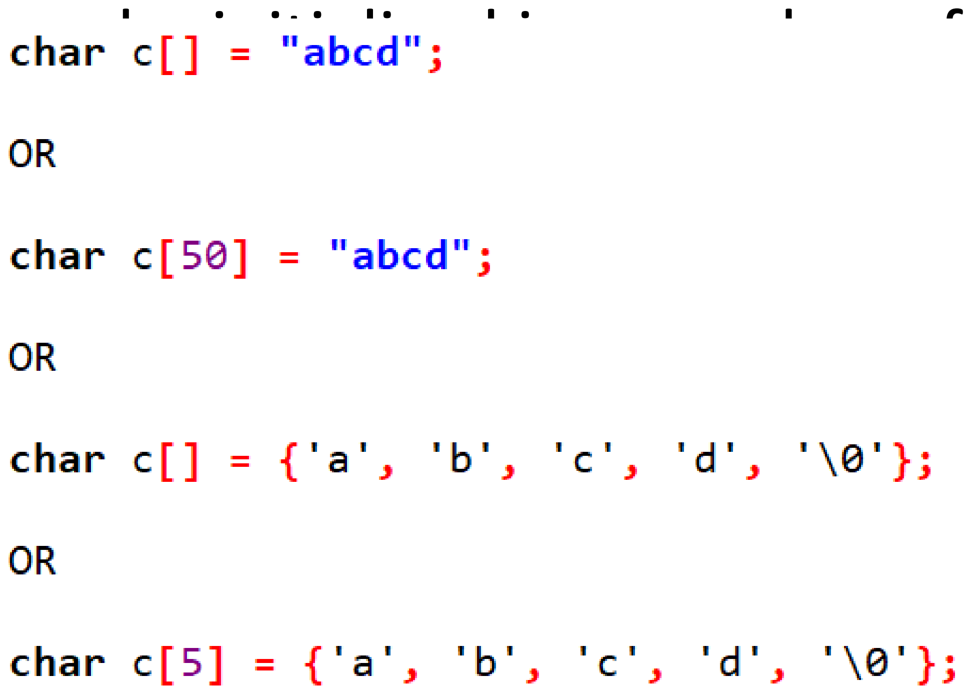
- Strings are declared in a similar manner as arrays.
- Only difference is that, strings are of char type.

```
char s[5];
```

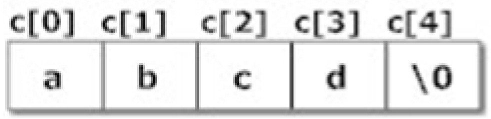


Initializing Strings...

- In C, string [illegible] lifferent ways.



```
char c[] = "abcd";

OR

char c[50] = "abcd";

OR

char c[] = {'a', 'b', 'c', 'd', '\0'};

OR

char c[5] = {'a', 'b', 'c', 'd', '\0'};
```



Question

- Consider the following set of characters
- “Volvo”
- Write a C program to do the followings:
- Store the above set of characters in an array
- Print all the characters in the array, one character at a Time.

Answer

```
#include <stdio.h>
int main() {
  char carName[] = "Volvo";
  int i;
    for (i = 0; i < 5; ++i) {
    printf("%c\n", carName[i]);
  }
  return 0;
}
```

```
V
o
l
v
o
```

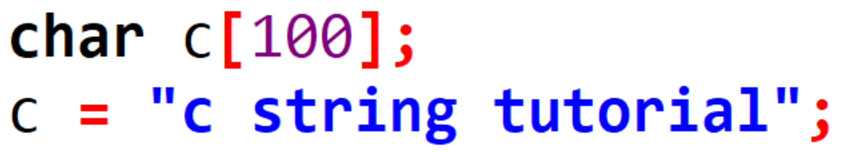
Assigning Values to Strings

- Arrays and strings are second-class citizens in C;
- So, they do not support the assignment operator once it is declared.
- Array type is not assignable.

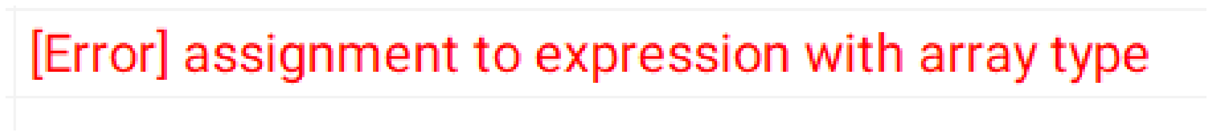
```
char c[100];
c = "c string tutorial";
```

Assigning Values to Strings

- Arrays and strings are second-class citizens in C;
- So, they do not support the assignment operator once it is declared.
- Array type is not assignable.



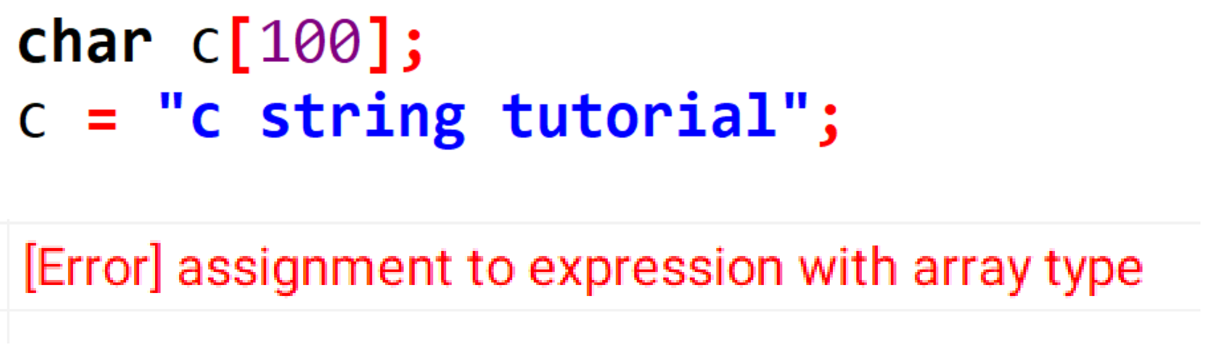
```c
char c[100];
c = "c string tutorial";
```



```
[Error] assignment to expression with array type
```

Assigning Values to Strings

- Arrays and strings are second-class citizens in C;
- So, they do not support the assignment operator once it is declared.
- Array type is not assignable.

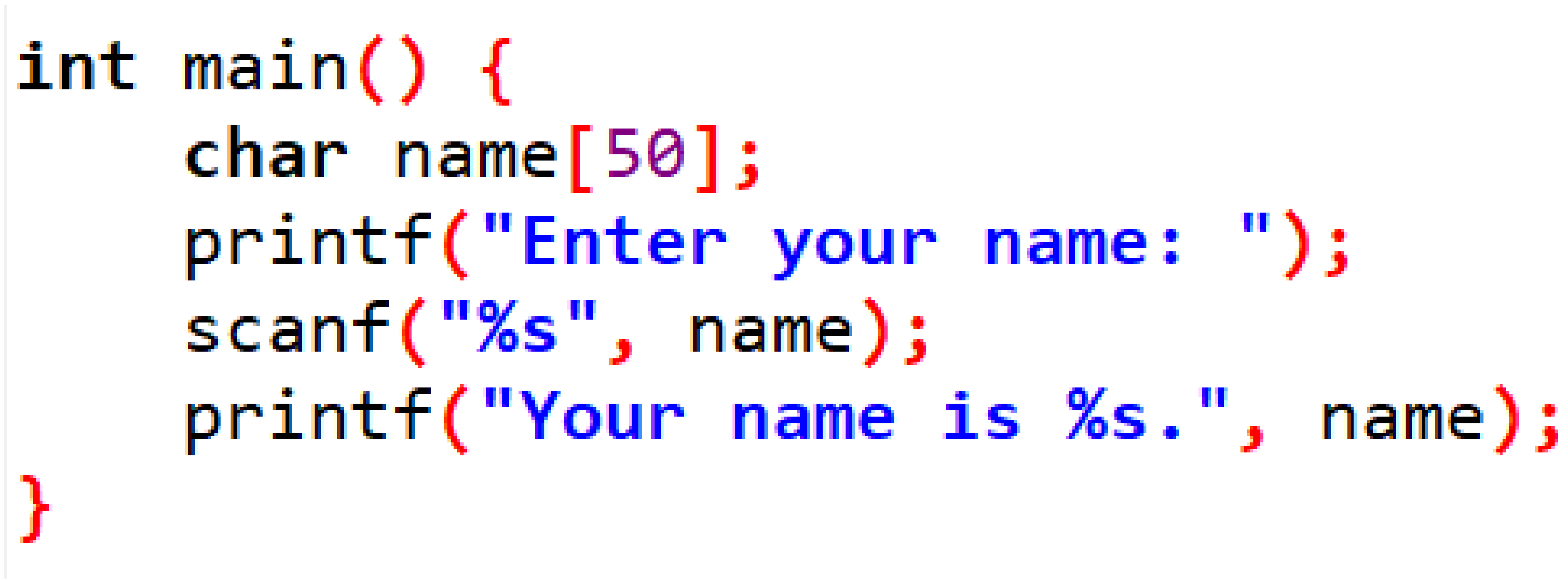


```
char c[100];
c = "c string tutorial";
```

```
[Error] assignment to expression with array type
```

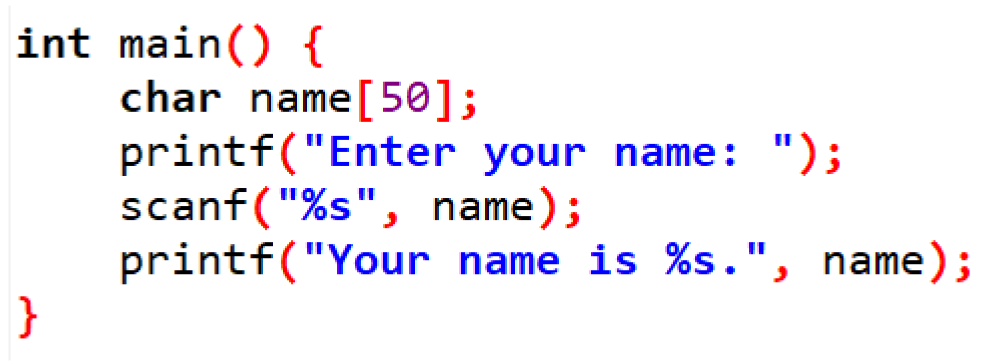
- We can use the strcpy() function to copy the string instead.

Reading Strings from user



```
int main() {
    char name[50];
    printf("Enter your name: ");
    scanf("%s", name);
    printf("Your name is %s.", name);
}
```

Reading Strings from user



```c
int main() {
    char name[50];
    printf("Enter your name: ");
    scanf("%s", name);
    printf("Your name is %s.", name);
}
```

```
Enter Your name  = “Sunil Perera”
Your name is :Sunil
```

Reading Strings from user

```c
int main() {
    char name[50];
    printf("Enter your name: ");
    scanf("%s", name);
    printf("Your name is %s.", name);
}
```

Enter our name : Sunil Perera

- Even though “Sunil Perera ” was entered in the above program, only " Sunil " was stored in the name string.
  - It's because there was a space after Sunil .
- Also notice that we have used the code name instead of &name with scanf().
  - Because name is an Array.

As we can observe from the above example.
`scanf()` stops scanning as soon as it encounters **whitespace** or newline

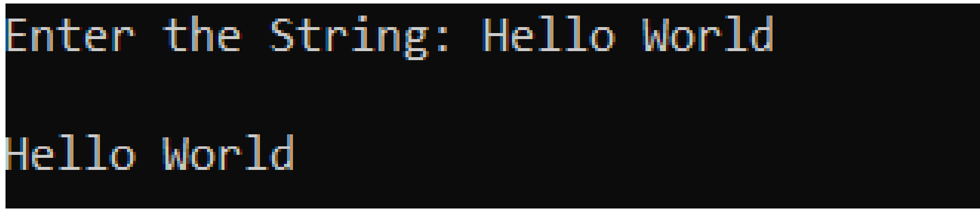
How to read a line of text?

- Gets() is a pre-defined function in C which is used to read a string or a text line.
- And store the input in a well-defined string variable.
- The function terminates its reading session as soon as it encounters a **newline character**.

- E.g

```
#include<stdio.h>
int main()
{
    char string[10];
    printf("Enter the String: ");
    gets(string);
    printf("\n%s",string);
    return 0;
}
```

Output is:



```
Enter the String: Hello World

Hello World
```

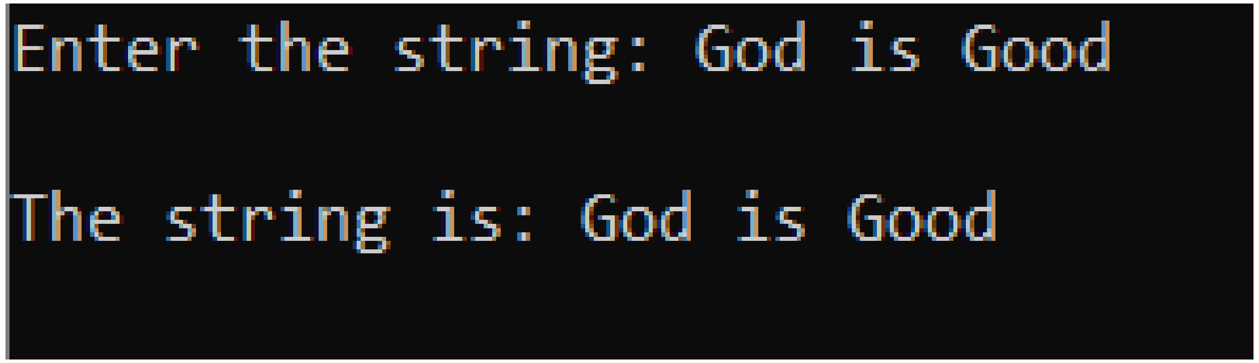
fgets() function in C

- The standard **C** library also provides us with yet another function, the fgets() function.
- The function reads a text line or a string from the specified file or console.
- And then stores it to the respective string variable.

```
• #include<stdio.h>
int main()
{
    char string[20];
    FILE *fp;
    fp=fopen("file.txt","r");
    fgets(string,20,fp);
    printf("The string is: %s",string);
    fclose(fp);
    return 0;
}
```

Read from stdin using fgets()

```
#include<stdio.h>
int main()
{
    char string[20];
    printf("Enter the string: ");
    fgets(string,20,stdin);        #input
from stdin stream
    printf("\nThe string is: %s",string);
    return 0;
}
```



```
Enter the string: God is Good

The string is: God is Good
```

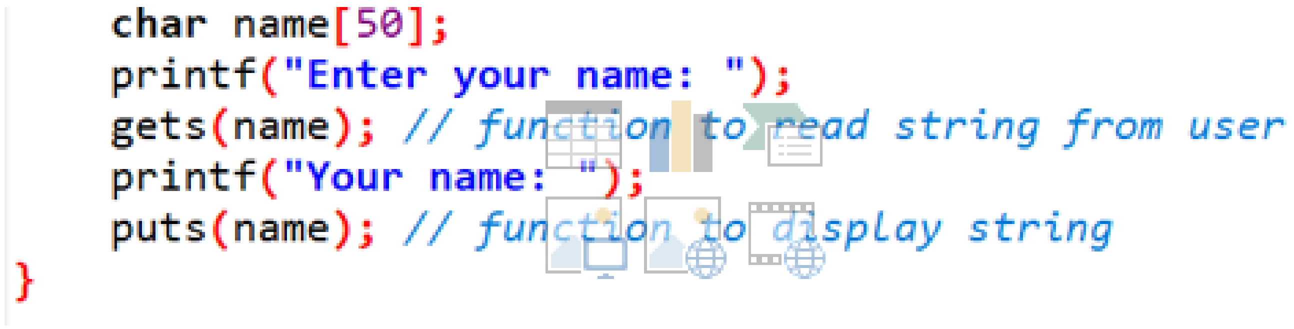
Note: stdin is an input stream where data is sent to and read by a program

Reading Line of Text

```
int main() {
	char name[50];
	printf("Enter your name: ");
	scanf("%[^\n]s", name);
	printf("Your name is %s.", name);
}
```

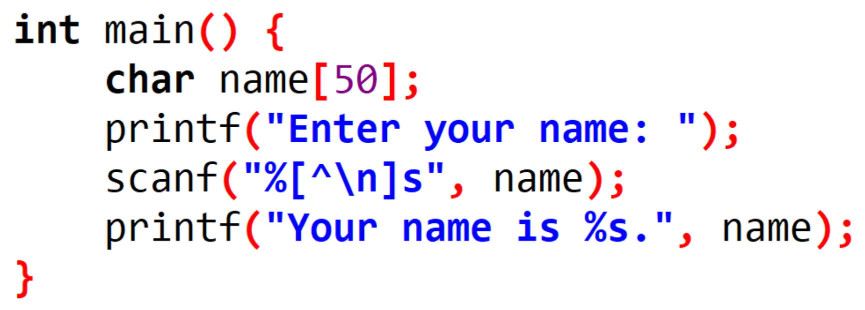
Reading Line of Text

```c
int main() {
    char name[50];
    printf("Enter your name: ");
    scanf("%[^\n]s", name);
    printf("Your name is %s.", name);
}
```

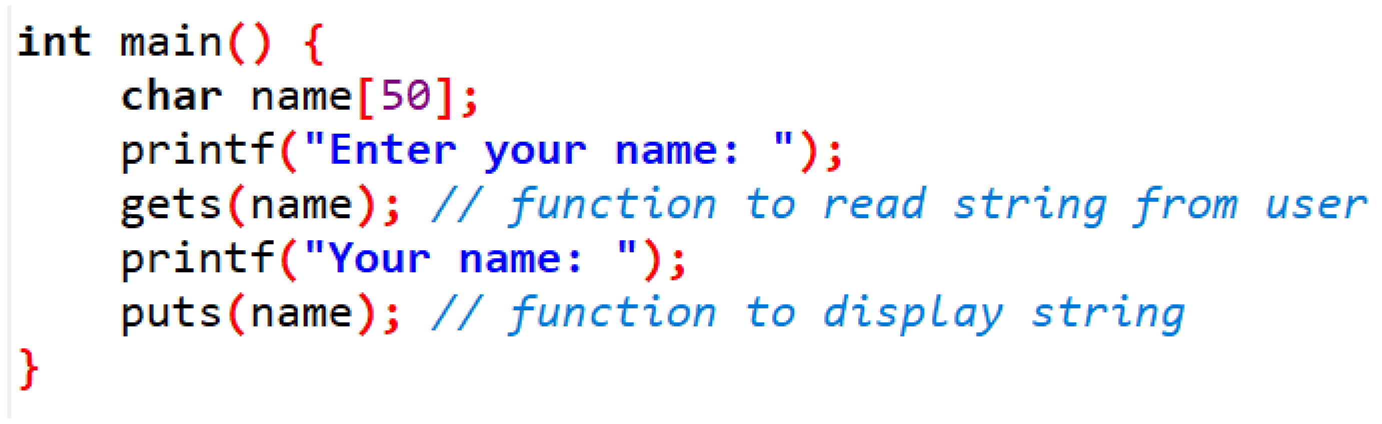


```c
    char name[50];
    printf("Enter your name: ");
    gets(name); // function to read string from user
    printf("Your name: ");
    puts(name); // function to display string
}
```

Reading Line of Text



```
int main() {
    char name[50];
    printf("Enter your name: ");
    scanf("%[^\n]s", name);
    printf("Your name is %s.", name);
}
```



```
int main() {
    char name[50];
    printf("Enter your name: ");
    gets(name); // function to read string from user
    printf("Your name: ");
    puts(name); // function to display string
}
```

- The gets() function can be to take input from the user, but it is removed from the C standard because gets() allows you to input any length of characters. Hence, there might be a buffer overflow.

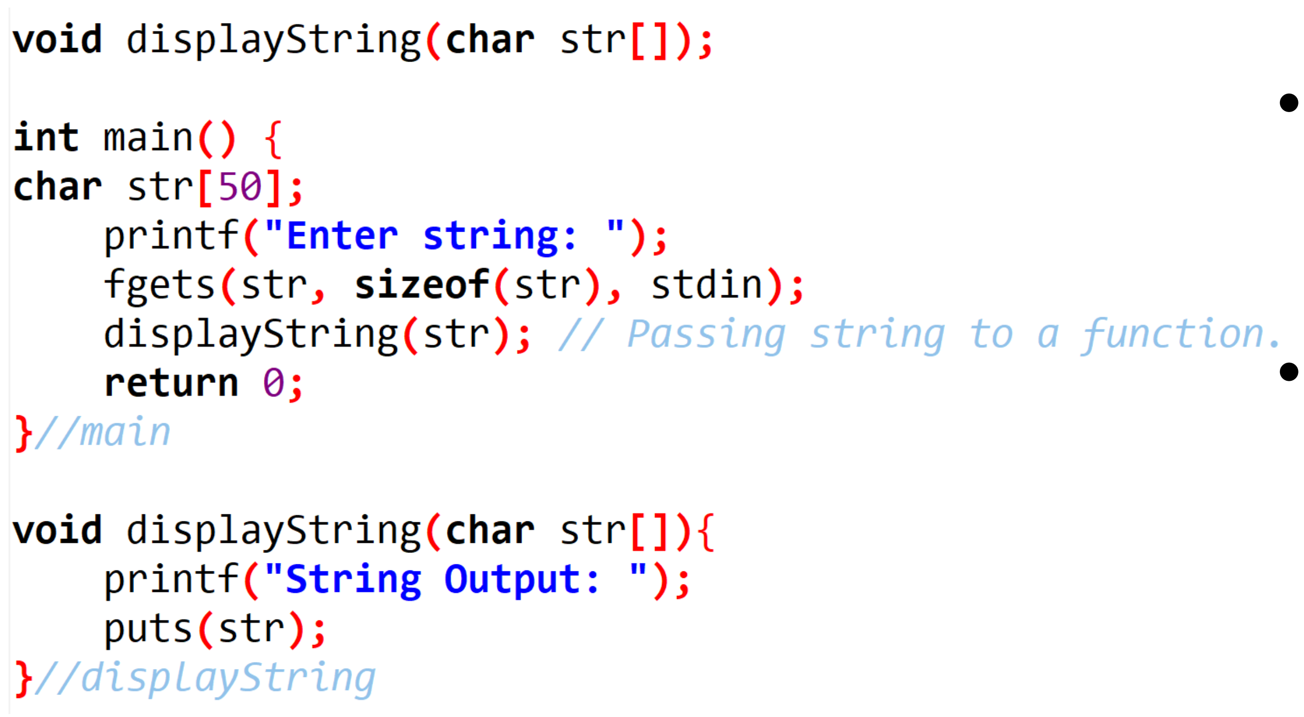
Reading Line of Text

- We can use the fgets() function to read a line of string, and, puts() to display the string.

```
int main() {
    char name[50];
    printf("Enter your name: ");
    fgets(name, sizeof(name), stdin); // ead a line of string
    printf("Your name: ");
    puts(name); // function to display string
}
```

- The sizeof(name) results to 50. Hence, we can take a maximum of 50 characters as input which is the size of the name string.

Passing string to a Function



```c
void displayString(char str[]);

int main() {
char str[50];
    printf("Enter string: ");
    fgets(str, sizeof(str), stdin);
    displayString(str); // Passing string to a function.
    return 0;
}//main

void displayString(char str[]){
    printf("String Output: ");
    puts(str);
}//displayString
```

- Even though both the functions, gets() and fgets() can be used for reading string inputs.
- The biggest difference between the two is the fact that the latter allows the user to specify the buffer size.

String Manipulation

- String manipulation can be done manually but, this makes programming complex and large.
- To solve this, C supports a large number of string handling functions, which are defined in the "string.h" header file.

Function	Work of Function
strlen()	computes string's length
strcpy()	copies a string to another
strcat()	concatenates(joins) two strings
strcmp()	compares two strings
strlwr()	converts string to lowercase
strupr()	converts string to uppercase

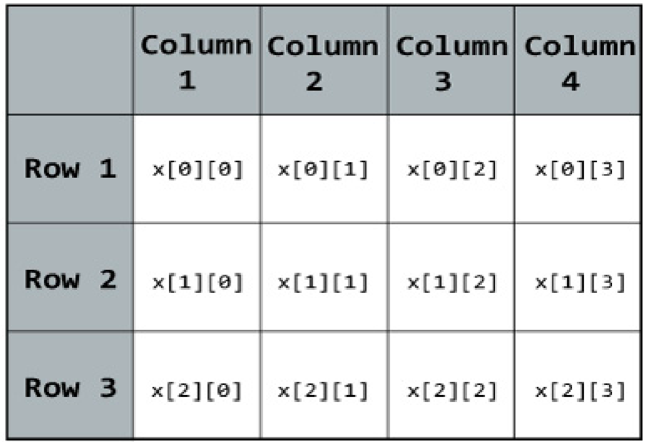
- More:
  - https://beginnersbook.com/2014/01/c-strings-string-functions/

Multidimensional Arrays

- In C programming, you can create an array of arrays known as multidimensional array.
- Example:

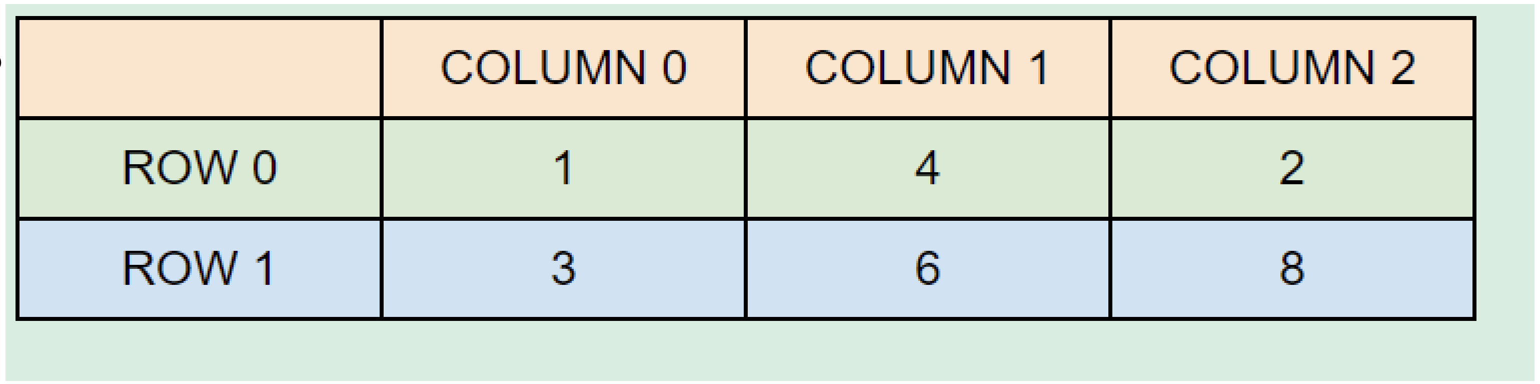
```c
float x[3][4];
```

- Here, x is a two-dimensional (2D) array. The array can hold 12 elements. You can think the array as table with 3 row and each row has 4 column.



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

Two-Dimensional Arrays

- A 2D array is also known as a matrix (a table of rows and columns).
- To create a 2D array of integers, take a look at the following example:
- int matrix[2][3] = { {1, 4, 2}, {3, 6, 8} };
- he first dimension represents the number of rows **[2]**, while the second dimension represents the number of columns **[3]**. The values are placed in row-order, and can be visualized like this:
- 

	COLUMN 0	COLUMN 1	COLUMN 2
ROW 0	1	4	2
ROW 1	3	6	8

Change Elements in a 2D Array

- Consider the following 2D Matrix
- int matrix[2][3] = { {1, 4, 2}, {3, 6, 8} };
- According to the above matrix,
- matix[0][0]=1
- How can above value as 9?

```
int matrix[2][3] = { {1, 4, 2}, {3, 6, 8} };
matrix[0][0] = 9;

printf("%d", matrix[0][0]);  // Now outputs 9
instead of 1
```

Loop Through a 2D Array

- Consider the array initiation
- int matrix[2][3] = { {1, 4, 2}, {3, 6, 8} };
- Write a C program to display the matrix contents as follows using two for next loops

```
1
4
2
3
6
8
```

How to Multiply Matrices

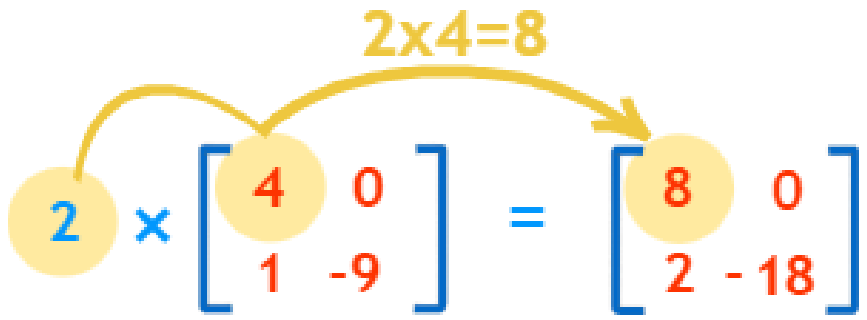
- A Matrix is an array of numbers:

$$\begin{bmatrix} 6 & 4 & 24 \\ 1 & -9 & 8 \end{bmatrix}$$

A Matrix

(This one has 2 Rows and 3 Columns)

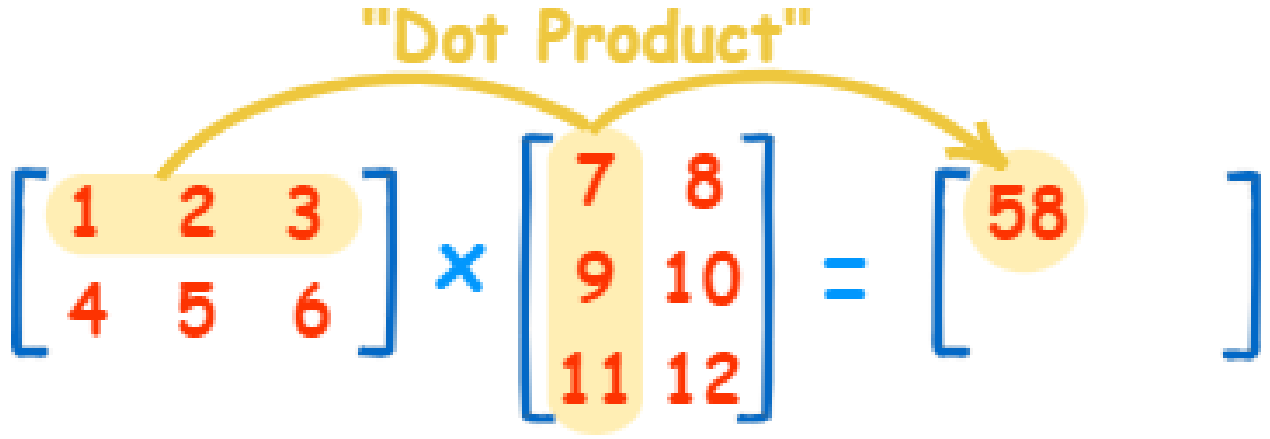
- To multiply a matrix by a single number is easy:



$$2 \times \begin{bmatrix} 4 & 0 \\ 1 & -9 \end{bmatrix} = \begin{bmatrix} 8 & 0 \\ 2 & -18 \end{bmatrix}$$

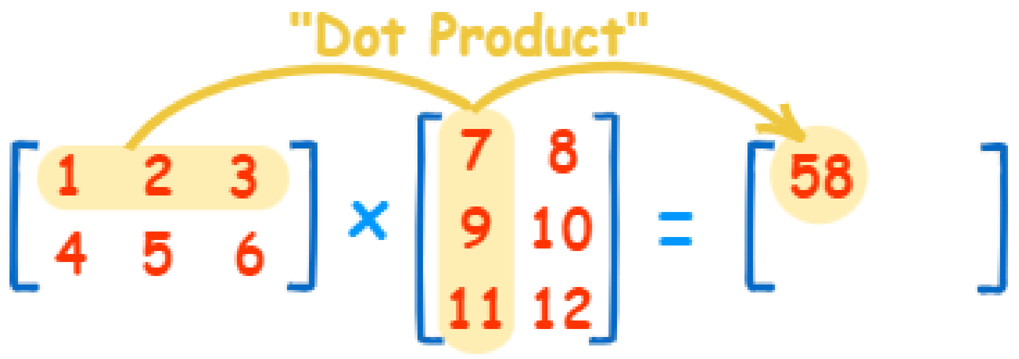
Multiplying a Matrix by Another Matrix

- But to multiply a matrix **by another matrix** we need to do the "dot product" of rows and columns ... what does that mean?
- Let us see with an example:
- To work out the answer for the **1st row** and **1st column**:



$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & \\ & \end{bmatrix}$$

Multiplying a Matrix by Another Matrix


"Dot Product"
$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & \\ & \end{bmatrix}$

Find the product of the above two matrices ?

Multiplying a Matrix by Another Matrix

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix}$$ ✓

Question : Write a C program to **Multipl**y two matrices

Multiplying a Matrix by Another Matrix using C

```
#include <stdio.h>
#include <stdlib.h>

// Edit MACROs here, according to your Matrix
Dimensions for
// mat1[R1][C1] and mat2[R2][C2]
#define R1 2 // number of rows in Matrix-1
#define C1 2 // number of columns in Matrix-1
#define R2 2 // number of rows in Matrix-2
#define C2 2 // number of columns in Matrix-2
```

Multiplying a Matrix by Another Matrix using C

```c
void mulMat(int mat1[][C1], int mat2[][C2])
{
    int rslt[R1][C2];

    printf("Multiplication of given two matrices is:\n");

    for (int i = 0; i < R1; i++) {
        for (int j = 0; j < C2; j++) {
            rslt[i][j] = 0;

            for (int k = 0; k < R2; k++) {
                rslt[i][j] += mat1[i][k] * mat2[k][j];
            }

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

        printf("\n");
    }
}
```

Multiplying a Matrix by Another Matrix using C

```c
// Driver code
int main()
{
    // R1 = 4, C1 = 4 and R2 = 4, C2 = 4 (Update these
    // values in MACROs)
    int mat1[R1][C1] = { { 1, 1 },
                         { 2, 2 } };

    int mat2[R2][C2] = { { 1, 1 },
                         { 2, 2 } };


    if (C1 != R2) {
        printf("The number of columns in Matrix-1  must be "
               "equal to the number of rows in "
               "Matrix-2\n");
        printf("Please update MACROs value according to "
               "your array dimension in "
               "#define section\n");
```

Multiplying a Matrix by Another Matrix using C

```
         exit(EXIT_FAILURE);
        }

          // Function call
        mulMat(mat1, mat2);

        return 0;
    }
```

What is the Transpose of a Matrix?

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}^{T} \longrightarrow \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

Question :Write a C program to get transpose of a matrix

Answer : Write a C program to get transpose of a matrix

```
#include <stdio.h>
void transpose(int p[3][3], int t[3][3]);
int main()
{ int i, j;
int p[3][3], t[3][3];
```

Answer : Write a C program to get transpose of a matrix

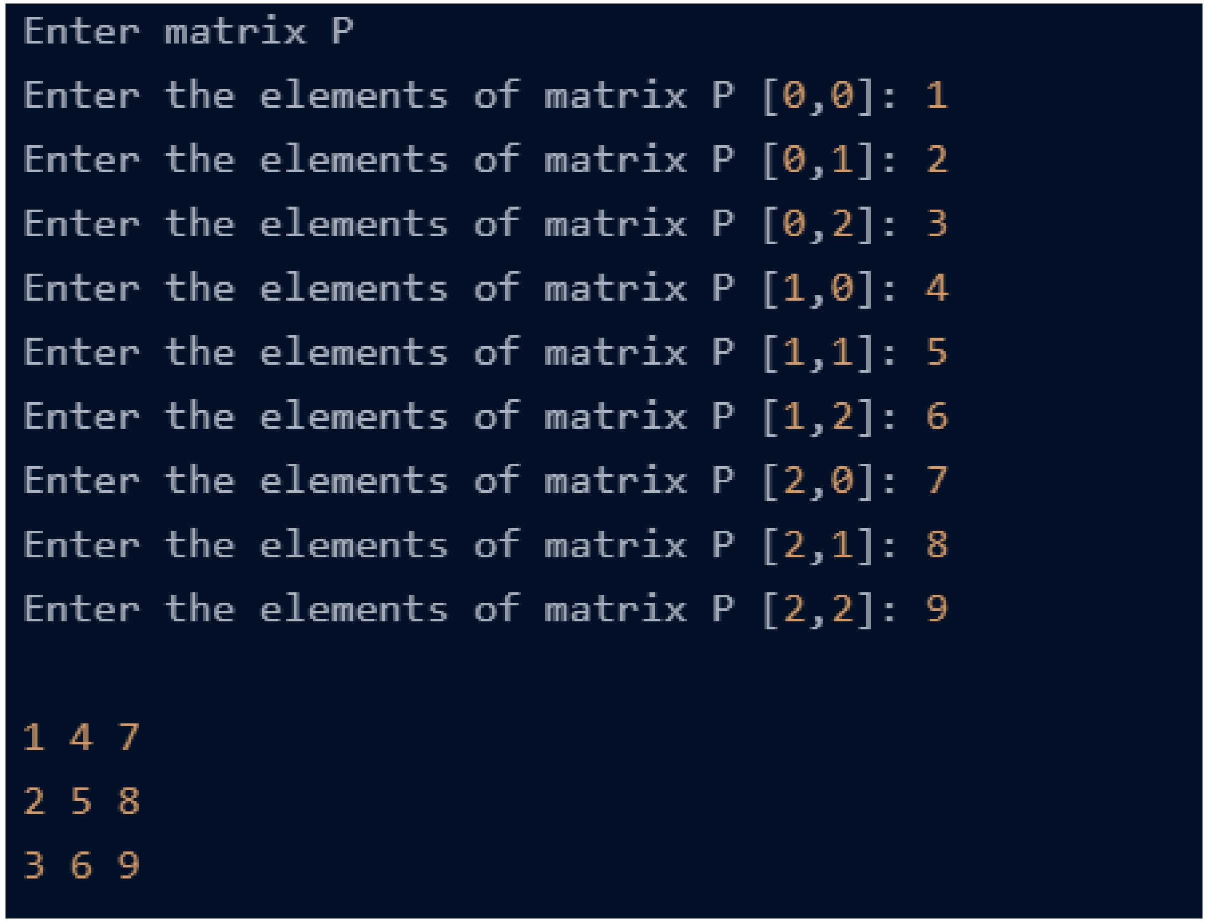
```
printf("Enter matrix P\n");
for (i = 0; i < 3; i++) {
for (j = 0; j < 3; j++) {
 printf("Enter the elements of matrix P [%d,%d]:
", i, j);
scanf("%d", & p[i][j]); } }
```

Write a C program to get transpose of a matrix

```
transpose(p, t);
printf("Transpose of matrix P is:\n\n");
for (i = 0; i < 3; i++) {
for (j = 0; j < 3; j++) {
 printf("%d ", t[i][j]); }
printf("\n");
}
}
```

Answer : Write a C program to get transpose of a matrix

```
void transpose(int p[3][3],
int t[3][3]) { int row,
col;
for (row = 0; row < 3;
row++) {
for (col = 0; col < 3;
col++) {
t[row][col] = p[col][row];
}
}
}
```



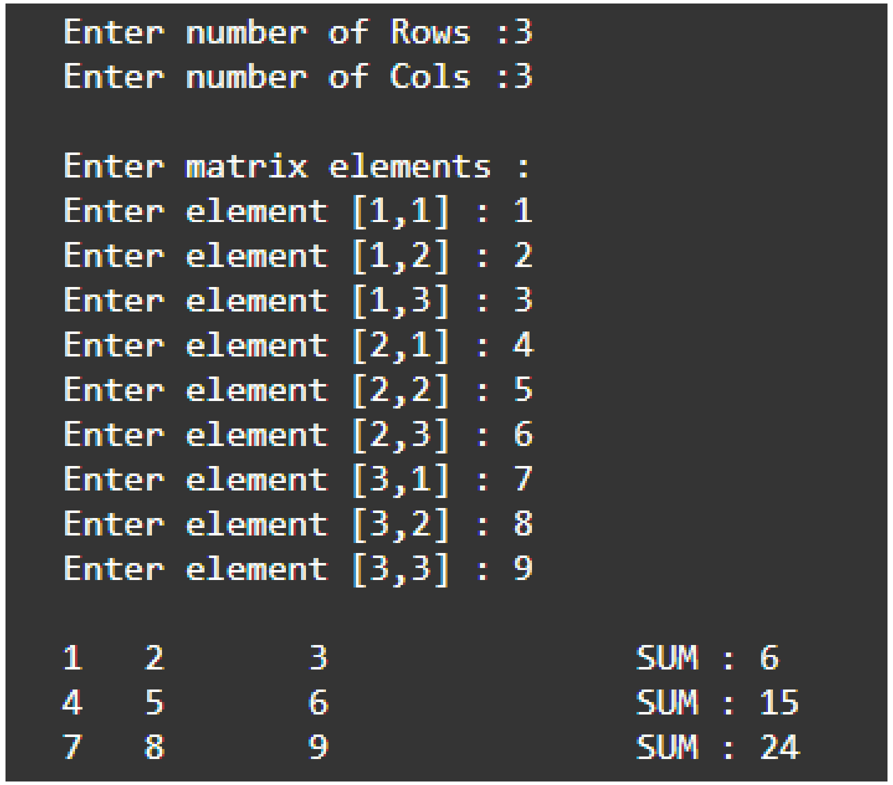
```
Enter matrix P
Enter the elements of matrix P [0,0]: 1
Enter the elements of matrix P [0,1]: 2
Enter the elements of matrix P [0,2]: 3
Enter the elements of matrix P [1,0]: 4
Enter the elements of matrix P [1,1]: 5
Enter the elements of matrix P [1,2]: 6
Enter the elements of matrix P [2,0]: 7
Enter the elements of matrix P [2,1]: 8
Enter the elements of matrix P [2,2]: 9


1 4 7
2 5 8
3 6 9
```

What will be the output of the following C code?

```
• #include <stdio.h>
• void f(int a[][])
• {
• a[0][1] = 3;
• int i = 0, j = 0;
• for (i = 0;i < 2; i++)
• for (j = 0;j < 3; j++)
• printf("%d", a[i][j]);
• }
• void main()
• {
• int a[2][3] = {0};
• f(a);
• }
```

Write a C Program to find sum of all elements of each row of a matrix

- Example



```
Enter number of Rows :3
Enter number of Cols :3

Enter matrix elements :
Enter element [1,1] : 1
Enter element [1,2] : 2
Enter element [1,3] : 3
Enter element [2,1] : 4
Enter element [2,2] : 5
Enter element [2,3] : 6
Enter element [3,1] : 7
Enter element [3,2] : 8
Enter element [3,3] : 9

1   2       3               SUM : 6
4   5       6               SUM : 15
7   8       9               SUM : 24
```

Write a C Program to find sum of all elements of each row of a matrix

- This C program will read a Matrix (two dimensional arrays) and print the sum of all elements of each row.

```
#include <stdio.h>
#define MAXROW          10
#define MAXCOL          10
int main()
{
        int matrix[MAXROW][MAXCOL];
        int i,j,r,c;
        int sum,product;
        printf("Enter number of Rows :");
        scanf("%d",&r);
        printf("Enter number of Cols :");
        scanf("%d",&c);
        printf("\nEnter matrix elements :\n");
```

Write a C Program to find sum of all elements of each row of a matrix

```
for(i=0;i< r;i++)
        {
                for(j=0;j< c;j++)
                {
                        printf("Enter element [%d,%d] : ",i+1,j+1);
                        scanf("%d",&matrix[i][j]);
                }
        }
```

Write a C Program to find sum of all elements of each row of a matrix

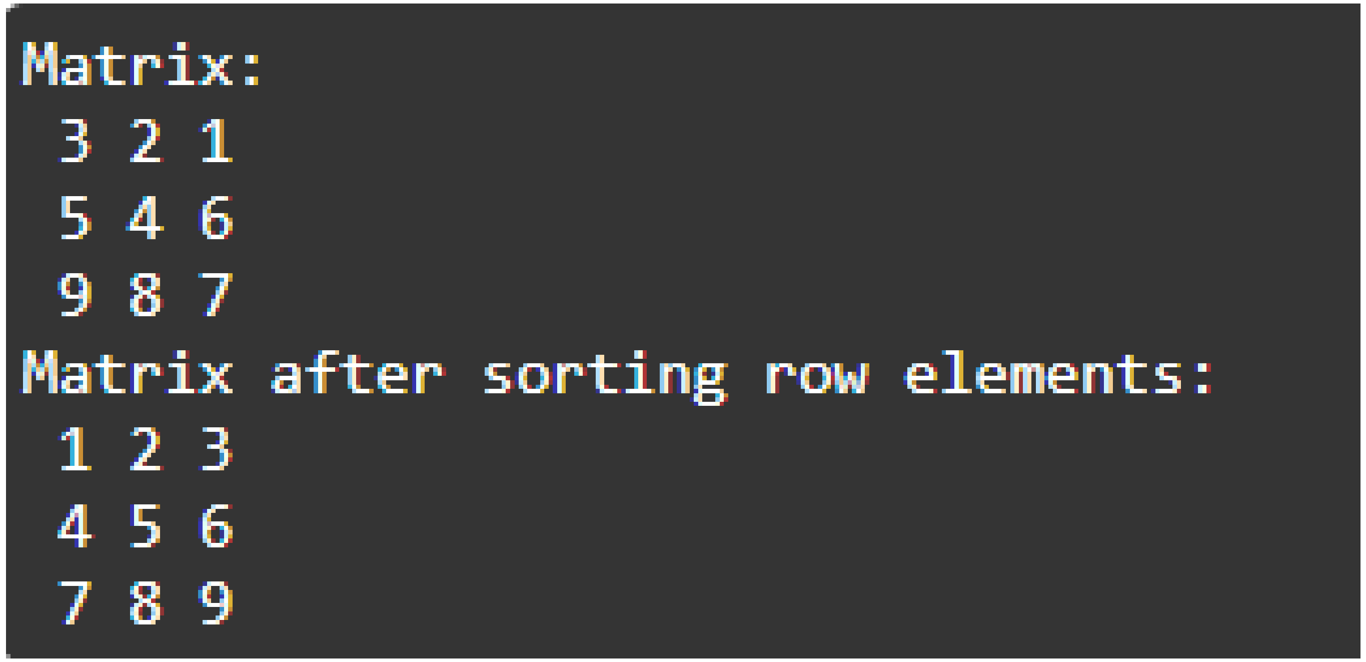
```
printf("\n");
        /*sum of all rows*/
        for(i=0;i< r;i++)
        {
                sum=0;                  /*initializing sum*/
                for(j=0;j< c;j++)
                {
                        printf("%d\t",matrix[i][j]);         /*print elements*/
```

Write a C Program to find sum of all elements of each row of a matrix

```
        sum             +=      matrix[i][j];
            }
            printf("\tSUM : %d",sum);
            printf("\n"); /*after each row print new line*/
        }

}
```

Write a C program to arrange row elements in ascending order

- **Output:**



```
Matrix:
 3 2 1
 5 4 6
 9 8 7
Matrix after sorting row elements:
 1 2 3
 4 5 6
 7 8 9
```

Write a C program to arrange row elements in ascending order

```
// C program to arrange row elements in ascending order
#include <stdio.h>
#define ROW 3
#define COL 3
int main()
{
    int Matrix[ROW][COL] = {
        { 3, 2, 1 },
        { 5, 4, 6 },
        { 9, 8, 7 }
    };
```

Write a C program to arrange row elements in ascending order

```
int i, j, k, temp;
    printf("Matrix:\n");
    for (i = 0; i < ROW; ++i) {
        for (j = 0; j < COL; ++j)
            printf(" %d", Matrix[i][j]);
        printf("\n");
    }
```

Write a C program to arrange row elements in ascending order

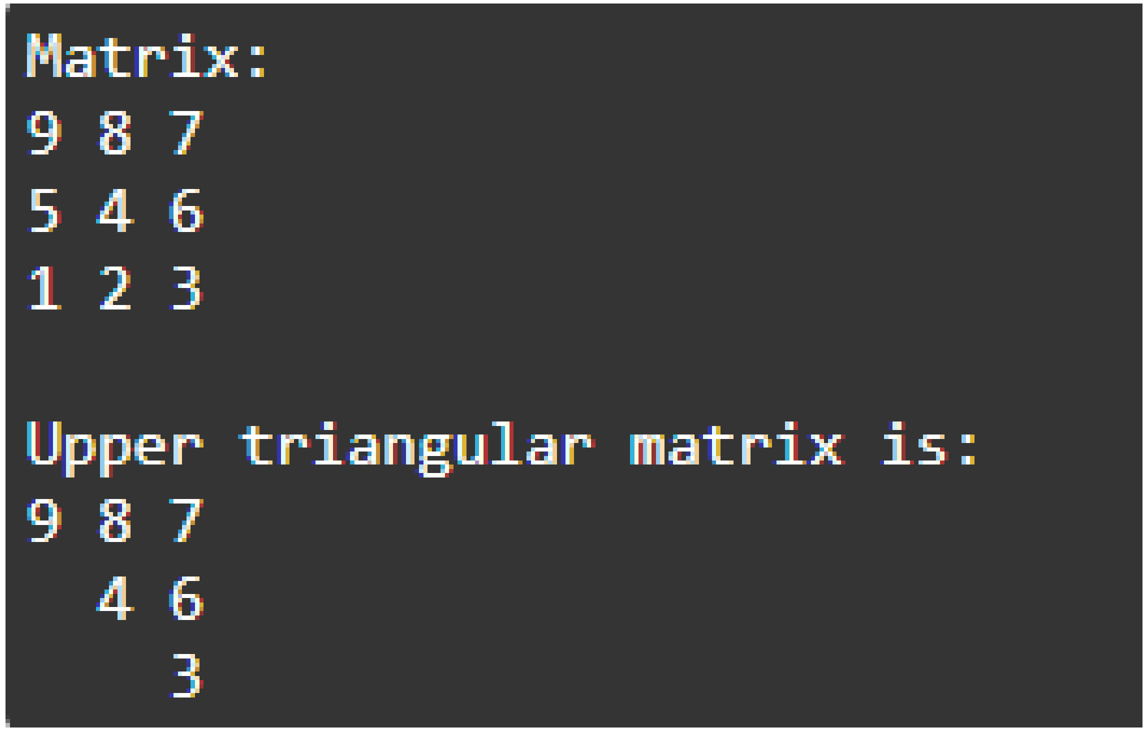
```
// Arrange rows elements in ascending order
   for (i = 0; i < ROW; ++i) {
      for (j = 0; j < COL; ++j) {
         for (k = (j + 1); k < COL; ++k) {
            if (Matrix[i][j] > Matrix[i][k]) {
               temp = Matrix[i][j];
               Matrix[i][j] = Matrix[i][k];
               Matrix[i][k] = temp;
            }
         }
      }
```

Write a C program to arrange row elements in ascending order

```
}

    printf("Matrix after sorting row elements:\n");
    for (i = 0; i < ROW; ++i) {
        for (j = 0; j < COL; ++j)
            printf(" %d", Matrix[i][j]);
        printf("\n");
    }
    return 0;
}
```

Write a C program to print the upper triangular matrix

- Expected output :



```
Matrix:
9 8 7
5 4 6
1 2 3

Upper triangular matrix is:
9 8 7
  4 6
    3
```

Write a C program to print the upper triangular matrix

```
// C program to print the upper triangular matrix
#include <stdio.h>
int main()
{
    int Matrix[3][3] = {
        { 9, 8, 7 },
        { 5, 4, 6 },
        { 1, 2, 3 }
    };
```

Write a C program to print the upper triangular matrix

```
int i, j;
   printf("Matrix:\n");
   for (i = 0; i < 3; ++i) {
       for (j = 0; j < 3; ++j) {
           printf("%d ", Matrix[i][j]);
       }
       printf("\n");
   }
```

Write a C program to print the upper triangular matrix

```
printf("\nUpper triangular matrix is: \n");
    for (i = 0; i < 3; i++) {
         for (j = 0; j < 3; j++) {
            if (j >= i)
                printf("%d ", Matrix[i][j]);
            else
                printf("  ");
        }
        printf("\n");
    }
    return 0;
}
```

Write a C Program to print diagonal elements of a Matrix

Write a C Program to print diagonal elements of a Matrix

```
Enter no. of rows :: 3
Enter no. of cols :: 3
Enter values to the matrix ::
Enter a[0][0] value :: 1
Enter a[0][1] value :: 2
Enter a[0][2] value :: 3
Enter a[1][0] value :: 4
Enter a[1][1] value :: 5
Enter a[1][2] value :: 6
Enter a[2][0] value :: 7
Enter a[2][1] value :: 8
Enter a[2][2] value :: 9
```

- Expected Output:
- The Diagonal elements of a matrix are ::

```
       1
         5
           9
```