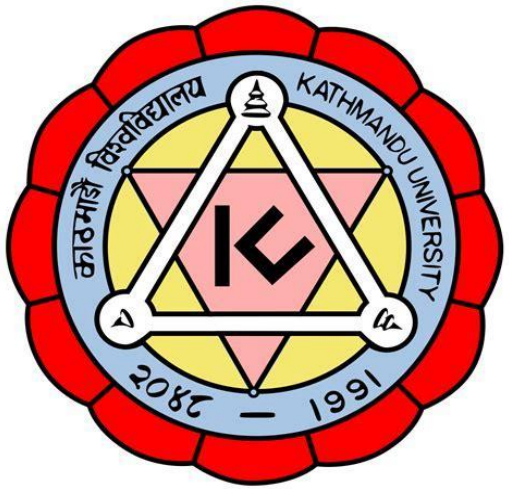
KATHMANDU UNIVERSITY SCHOOL OF MANAGEMENT

BBIS

COM 102 : 3 Credit Hours



8. Arrays

Outlines

- Introduction
- Defining an Array
- Processing an Array
- Passing Array to Function
- Multidimensional Array

- Quiz: Two sets
- Internal Exam

Why arrays???

Suppose, you want to declare 1 or at most 5 variable of same type.

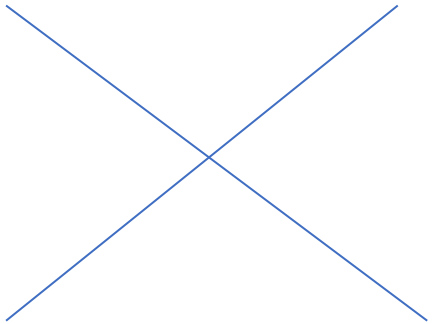
What will you do?
-> simply make 5 variable of same data type as usual.

NOT A BIG DEAL!!!!

But, you want store 100 or more variable of same type.

THEN???

```
int main()
{
Int marks1, marks2, … , marks100…..;
...
...
return 0;
}
```

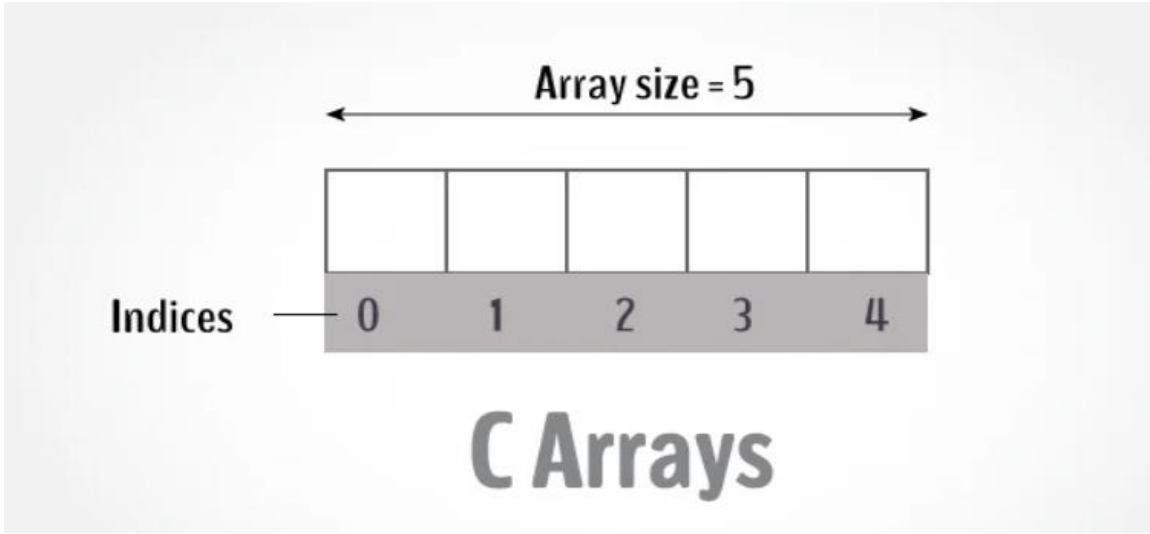


You will better do…

```
int main(void)
{
Int marks[100];   // Array that holds 100 integer data.
...
...
return 0;
}
```

Size = 100

OR




```
int main(void)
{
Int char[10];   // Array that holds 10 character data.
...
...
return 0;
}
```

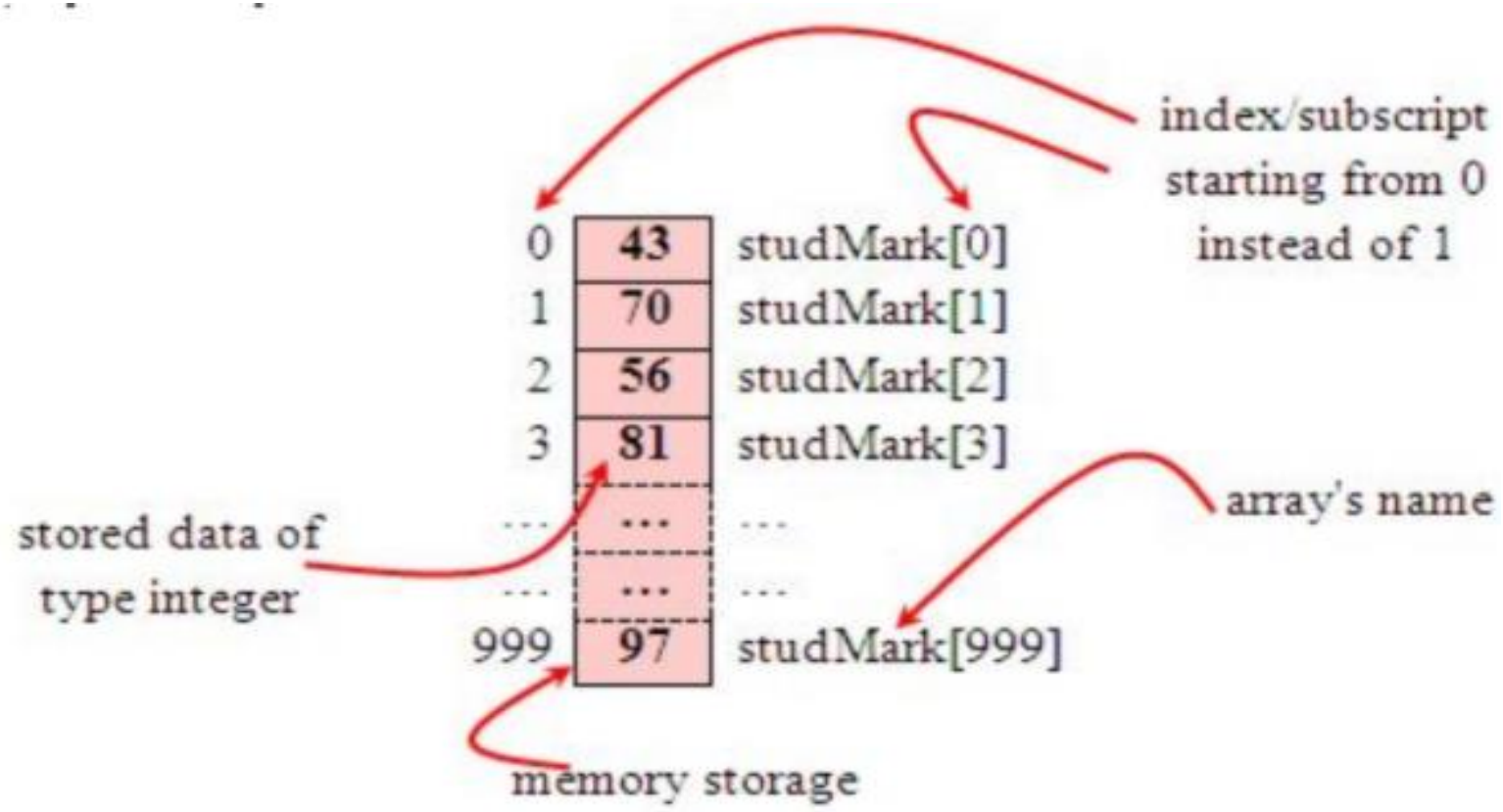
Size = 10

Arrays

- An array is a variable that can store multiple values. For example, if you want to store 100 integers, you can create an array for it.
- An array is a collection of elements of the same type that are referenced by a common name.
- It is an aggregate or derived data type. An array is a derived data type because it cannot be defined on its own, it is a collection of basic data types usually, such as integers, doubles, floats, Booleans, etc.
- All the elements of an array occupy a set of contiguous memory locations.

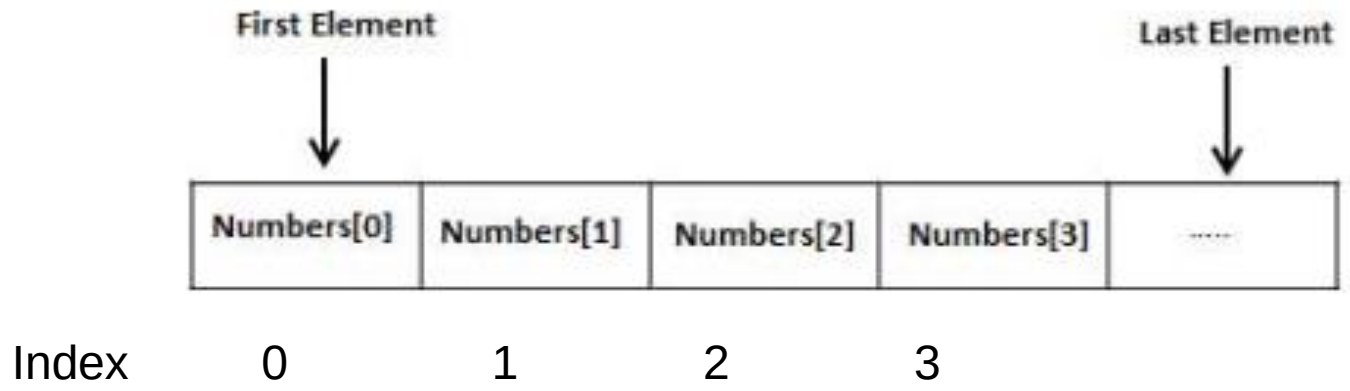
...

- Contiguous memory locations for storing 1000 students' marks.
- Graphically,



Arrays

- Instead of declaring individual variables, such as number0, number1, ..., and number99,
- declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables.
- A specific element in an array is accessed by an index.



ARRAYS

- In C language, array has a fixed size meaning once the size is given to it, it cannot be changed i.e. you can't shrink it neither can you expand it.

- if we change the size we can't be sure (it's not possible every time) that we get the next memory location to us as free.

- The shrinking will not work because the array, when declared, gets memory statically allocated, and thus compiler is the only one can destroy it.

One Dimensional Arrays

Dimension refers to the array's size, which is how big the array is.
A single or one dimensional array declaration has the following form,

```
array_element_data_type array_name[array_size];
```

Here,

- array_element_data_type define the base type of the array, which is the type of each element in the array.
- array_name is any valid C identifier name that obeys the same rule for the identifier naming.
- array_size defines how many elements the array will hold.

...

For example, to declare an array of 30 characters, that construct a people name, we could declare,

```
char cName[30];
```

- In this statement, the array character can store up to 30 characters with the first character occupying location cName[0] and the last character occupying cName[29].
- Note that the index runs from 0 to 29. In C, an index always starts from 0 and ends with array's (size-1).
- So, take note the difference between the array size and subscript/index terms.

Note: In C, each character occupies 1 byte of data.

How to declare an array?

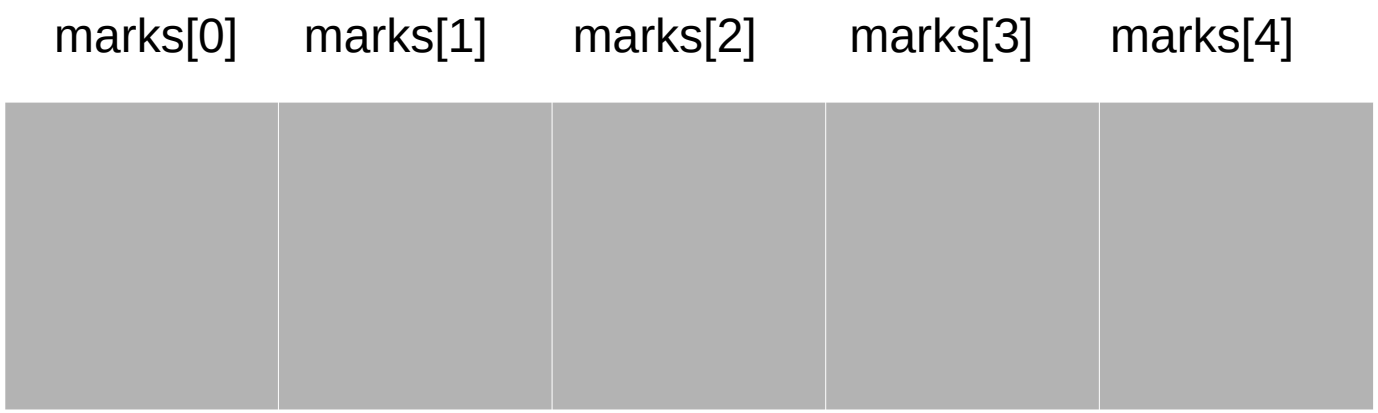
dataType arrayName[arraySize];
Eg: int marks[5], float data[20]
For example,

float mark[5];
Here, we declared an array, mark, of floating-point type. And its size is 5. Meaning, it can hold 5 floating-point values.

It's important to note that the size and type of an array cannot be changed once it is declared.

Access Array Elements

- You can access elements of an array by indices.
- Suppose you declared an array mark as above. The first element is mark[0], the second element is mark[1] and so on.



...

- Arrays have 0 as the first index, not 1. In this example, mark[0] is the first element.
- If the size of an array is n, to access the last element, the n-1 index is used. In this example, mark[4]
- Suppose the starting address of mark[0] is 2120d. Then, the address of the mark[1] will be 2124d. Similarly, the address of mark[2] will be 2128d and so on.
- This is because the size of a float is 4 bytes.

How to initialize an array?

It is possible to initialize an array during declaration. For example,

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

You can also initialize an array like this.

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

- Here, we haven't specified the size.
- However, the compiler knows its size is 5 as we are initializing it with 5 elements.

...

- mark[0] is equal to 60
- mark[1] is equal to 70
- mark[2] is equal to 80
- mark[3] is equal to 90
- mark[4] is equal to 100

marks[0]	marks[1]	marks[2]	marks[3]	marks[4]
60	70	80	90	100

Change Value of Array elements

```
int mark[5] = {50,60,70,80,90,100}

// make the value of the third element to 20
mark[2] = 20;

// make the value of the fifth element to 0
mark[4] = 0;
```

Input and Output Array Elements

Here's how you can take input from the user and store it in an array element.

```
// take input and store it in the 3rd element
scanf("%d", &mark[2]);

// take input and store it in the ith element
scanf("%d", &mark[i-1]);
```

Here's how you can print an individual element of an array.

```
// print the first element of the array
printf("%d", mark[0]);

// print the third element of the array
printf("%d", mark[2]);

// print ith element of the array
printf("%d", mark[i-1]);
```

// Program to find the average of n numbers using arrays

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

  int marks[10], i, n, sum = 0, 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];
  }

  average = sum / n;
  printf("Average = %d", average);

  return 0;
}
```

```
// C program to find the smallest and largest element in an array
#include<stdio.h>
int main()
{
int a[50],i,n,large,small;
printf("\nEnter the number of elements : ");
scanf("%d",&n);
printf("\nInput the array elements : ");
for(i=0;i<n;++i)
{
printf("\nelement [%d]:", i);
scanf("%d",&a[i]);
}
large=small=a[0];
for(i=1;i<n;++i)
{
if(a[i]>large)
large=a[i];
if(a[i]<small)
small=a[i];
}
printf("\nThe smallest element is %d\n",small);
printf("\nThe largest element is %d\n",large);
return 0;
}
```

```c
// Program to take 5 values from the user and store them in an array
// Print the elements stored in the array
#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]);
  }

  printf("Displaying integers: ");

  // printing elements of an array
  for(int i = 0; i < 5; ++i) {
    printf("%d\n", values[i]);
  }
  return 0;
}
```

```
Output:
Enter 5 integers: 1
-3
34
0
3
Displaying integers:
1
-3
34
0
3
```

Multidimensional arrays

In C programming, you can create an array of arrays.
These arrays are known as multidimensional arrays.
For example,

```
int x[3][4];
```

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

Contd...

	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]

2D-array

		Columns			
c[0] Array	Rows	c[0][0]	c[0][1]	c[0][2]	c[0][3]
		c[1][0]	c[1][1]	c[1][2]	c[1][3]
		c[2][0]	c[2][1]	c[2][2]	c[2][3]
		Columns			
c[1] Array	Rows	c[0][0]	c[0][1]	c[0][2]	c[0][3]
		c[1][0]	c[1][1]	c[1][2]	c[1][3]
		c[2][0]	c[2][1]	c[2][2]	c[2][3]

3D-array

Similarly, you can declare a three-dimensional (3d) array. For example,

```
int x[2][3][4];
```

Here, the array x can hold 24 elements.

1. The memory allocated to variable c is of data type int.
2. Total capacity array can hold is 2*3*4, which is equal to 24 elements.
3. The data is being represented in the form of 2 arrays with 3 rows and 4 columns each.

Initializing a multidimensional array

Initialization of a 2d array

```
// Different ways to initialize two-dimensional array

int x[2][3] = {{1, 3, 0}, {-1, 5, 9}};

int x[][3] = {{1, 3, 0}, {-1, 5, 9}};

int x[2][3] = {1, 3, 0, -1, 5, 9};
```

Accessing Two-Dimensional Array Elements

An element in a two-dimensional array is accessed by using the subscripts, i.e., row index and column index of the array.

For example

```
int val = a[2][3];
```

The above statement will take the 4th element from the 3rd row of the array.

Example: Inserting elements in 2D-array

```
#include <stdio.h>
 int main () {
   /* an array with 5 rows and 2 columns*/
   int a[5][2];
   int i, j;
 /* input each array element's value */
   for ( i = 0; i < 5; i++ ) {
      for ( j = 0; j < 2; j++ ) {
         printf("a[%d][%d] “, i+1,j+1 );
         scanf(“%d“, &a[i][j]);
      }
   }
return 0;
}
```

Example: Accessing elements in 2D-array

```
#include <stdio.h>
 int main () {
   /* an array with 5 rows and 2 columns*/
   int a[5][2] = { {0,0}, {1,2}, {2,4}, {3,6},{4,8}};
   int i, j;

    /* output each array element's value */
   for ( i = 0; i < 5; i++ ) {
      for ( j = 0; j < 2; j++ ) {
         printf("a[%d][%d] = %d\n", i,j, a[i][j] );
      }
   }
   return 0;
}
```

Change Value of 2D-Array elements

```
int a[5][2] = { {0,0}, {1,2}, {2,4}, {3,6},{4,8}};

// make the value of the 3,2 element to 20
mark[2][1] = 20;

// make the value of the 5,1 element to 0
mark[4][0] = 0;
```

Examples: 2d Arrays

```
// C program to store temperature of two cities of a week and display it.
#include <stdio.h>
const int CITY = 2;
const int WEEK = 7;
int main()
{
  int temperature[CITY][WEEK];

  // Using nested loop to store values in a 2d array
  for (int i = 0; i < CITY; ++i)
  {
    for (int j = 0; j < WEEK; ++j)
    {
      printf("City %d, Day %d: ", i + 1, j + 1);
      scanf("%d", &temperature[i][j]);
    }
  }
  printf("\nDisplaying values: \n\n");
```

```
// Using nested loop to display vlues of a 2d array
  for (int i = 0; i < CITY; ++i)
  {
    for (int j = 0; j < WEEK; ++j)
    {
      printf("City %d, Day %d = %d\n", i + 1, j + 1, temperature[i][j]);
    }
  }
  return 0;
}
```

// C program to find the sum of two matrices of order 2*2

```
#include <stdio.h>
int main()
{
  float a[2][2], b[2][2], result[2][2];
  // Taking input using nested for loop
  printf("Enter elements of 1st
matrix\n");
  for (int i = 0; i < 2; ++i)
    for (int j = 0; j < 2; ++j)
    {
      printf("Enter a%d%d: ", i + 1, j + 1);
      scanf("%f", &a[i][j]);
    }
  // Taking input using nested for loop
  printf("Enter elements of 2nd
matrix\n");
  for (int i = 0; i < 2; ++i)
    for (int j = 0; j < 2; ++j)
    {
      printf("Enter b%d%d: ", i + 1, j + 1);
      scanf("%f", &b[i][j]);
    }
```

```
// adding corresponding elements of two arrays
 for (int i = 0; i < 2; ++i)
   for (int j = 0; j < 2; ++j)
   {
     result[i][j] = a[i][j] + b[i][j];
   }

 // Displaying the sum
 printf("\nSum Of Matrix:");

 for (int i = 0; i < 2; ++i)
   for (int j = 0; j < 2; ++j)
   {
     printf("%.1f\t", result[i][j]);

     if (j == 1)
       printf("\n");
   }
 return 0;
}
```

Initialization of a 3d array

```
int test[2][3][4] = {
    {{3, 4, 2, 3}, {0, -3, 9, 11}, {23, 12, 23, 2}},
    {{13, 4, 56, 3}, {5, 9, 3, 5}, {3, 1, 4, 9}}};
```

		Columns			
c[0] Array	Rows	c[0][0]	c[0][1]	c[0][2]	c[0][3]
		c[1][0]	c[1][1]	c[1][2]	c[1][3]
		c[2][0]	c[2][1]	c[2][2]	c[2][3]
		Columns			
c[1] Array	Rows	c[0][0]	c[0][1]	c[0][2]	c[0][3]
		c[1][0]	c[1][1]	c[1][2]	c[1][3]
		c[2][0]	c[2][1]	c[2][2]	c[2][3]

Try it yourself

- Inserting elements in a 3D array.
- Accessing elements in a 3D array.
- Updating elements in a 3D array

Passing array to a function

In C programming, you can pass an entire array to functions.

Before that lets see, how we can pass the single/individual array elements to a function.

- Passing array elements to a function is similar to passing variables to a function.

```
#include <stdio.h>
void display(int age1, int age2) {
  printf("%d\n", age1);
  printf("%d\n", age2);
}

int main() {
  int age[] = {2, 8, 4, 12};

  // pass second and third elements to display()
  display(age[1], age[2]);
  return 0;
}
```

Three ways of passing an array

Formal parameters as a pointer:

```
void myFunction(int *param) {
        .
        .
        .
}
```

Formal parameters as a sized array:

```
void myFunction(int param[10]) {
        .
        .
        .
}
```

Formal parameters as an unsized array:

```
void myFunction(int param[]) {
        .
        .
        .
}
```

Example: Passing 1D array

```
#include <stdio.h>

/* function declaration */
double getAverage(int arr[], int
size);

int main () {

   /* an int array with 5 elements */
   int balance[5] = {1000, 2, 3, 17,
50};
   double avg;

   /* pass pointer to the array as an
argument */
   avg = getAverage( balance, 5 ) ;

   /* output the returned value */
   printf( "Average value is: %f ",
avg );
return 0;
}
```

```
double getAverage(int arr[], int size) {

   int i;
   double avg;
   double sum = 0;

   for (i = 0; i < size; ++i) {
      sum += arr[i];
   }

   avg = sum / size;

   return avg;
}
```

Example: Passing 2D array

```
#include <stdio.h>

void displayNumbers(int num[2][2]);

int main() {
  int num[2][2];
  printf("Enter 4 numbers:\n");
  for (int i = 0; i < 2; ++i) {
    for (int j = 0; j < 2; ++j) {
      scanf("%d", &num[i][j]);
    }
  }

  // pass multi-dimensional array to a function
  displayNumbers(num);

  return 0;
}
```

```
void displayNumbers(int num[2][2]) {
  printf("Displaying:\n");
  for (int i = 0; i < 2; ++i) {
    for (int j = 0; j < 2; ++j) {
      printf("%d\n", num[i][j]);
    }
  }
}
```

When passing two-dimensional arrays, it is not mandatory to specify the number of rows in the array. However, the number of columns should always be specified.

```
void displayNumbers(int num[][2]) {
  // code
}
```

Any Queries???