```
CSE1062 | CCS1063 ‘Practicals’ {

    [Fundamentals of Computer Programming]

        < Tutorial Session 07 - Arrays >

}
```

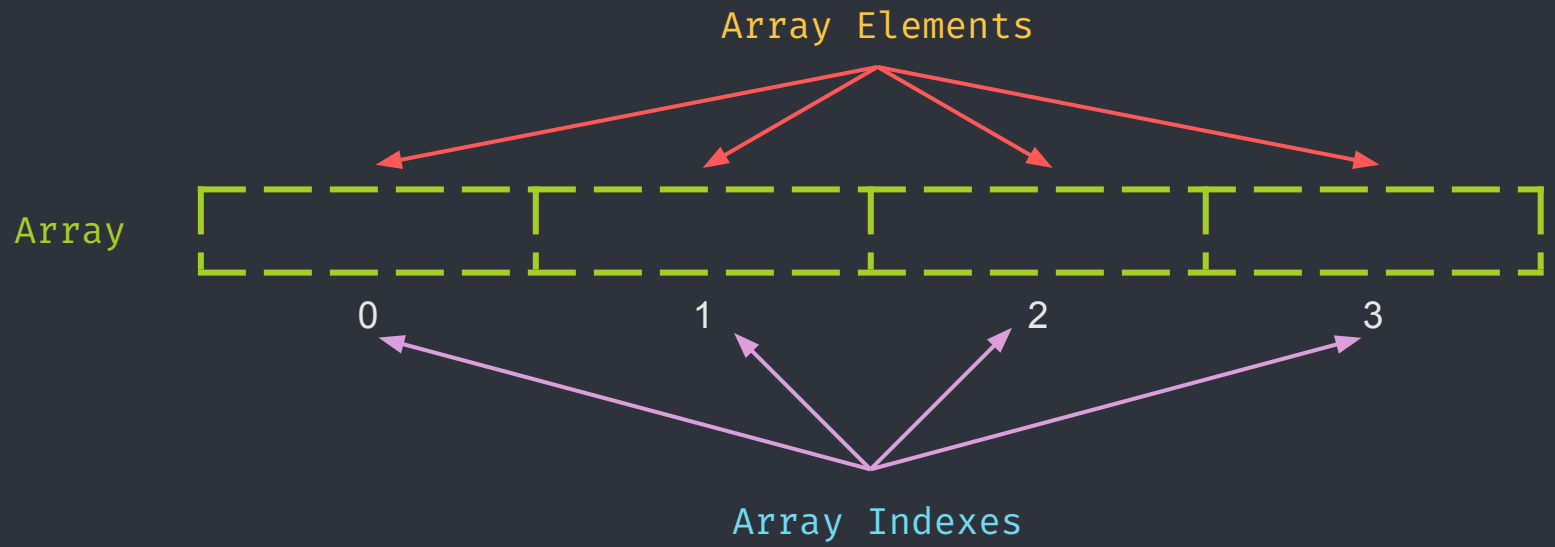
What is an array?

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.

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.

https://www.geeksforgeeks.org/c-arrays/

Arrays in C?



Properties of an Array

Most fundamental data structure.

* A fixed collection of same type data.
* Elements are stored in continuous blocks in computer memory.
* Elements of an array can be accessed by their indexes.
* Array types are:
    - Simple array (1D)
    - Multidimensional array (2D, 3D)

https://www.geeksforgeeks.org/c-arrays/

Advantage of C Array

1. **Code Optimization**: Less code to the access the data.
2. **Ease of traversing**: By using the for loop, we can retrieve the elements of an array easily.
3. **Ease of sorting**: To sort the elements of the array, we need a few lines of code only.
4. **Random Access**: We can access any element randomly using the array.

https://www.javatpoint.com/c-array

Disadvantage of C Array

Fixed Size:

Whatever size, we define at the time of declaration of the array, we can't exceed the limit. So, it doesn't grow the size dynamically like LinkedList which we will learn later.

https://www.javatpoint.com/c-array

Declaration of C Array

In C, we have to declare the array like any other variable before using it.

We can declare an array by specifying its name, the type of its elements, and the size of its dimensions.

When we declare an array in C, the compiler allocates the memory block of the specified size to the array name.

https://www.geeksforgeeks.org/c-arrays/

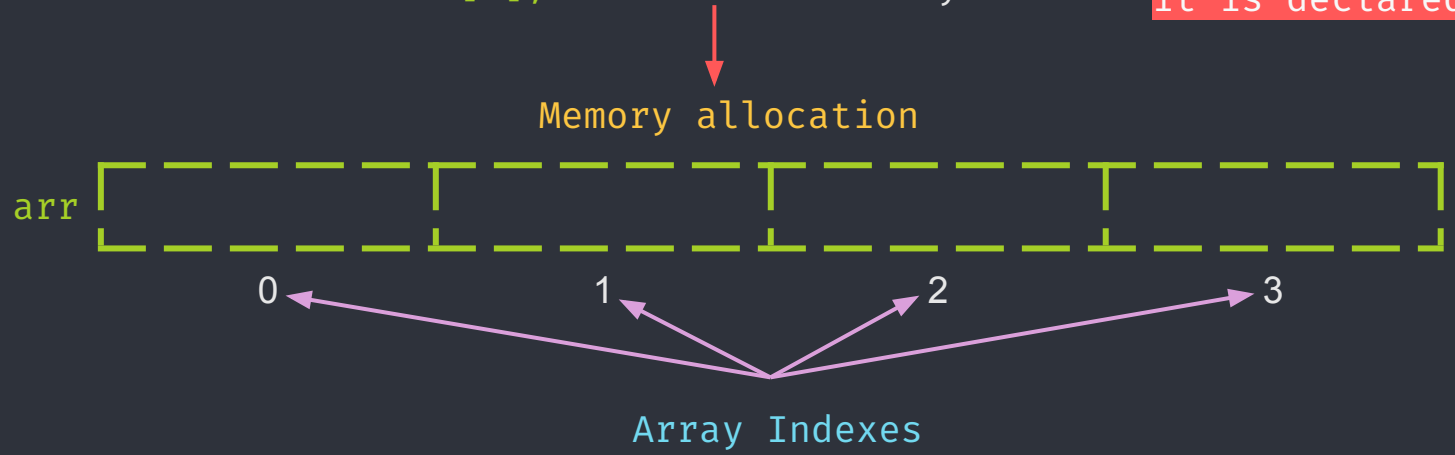
Syntax of Array Declaration

```
data_type array_name [size];

        or

data_type array_name [size1] [size2]...[sizeN];
```

```
arr[4];  Size of the array is 4
```



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

How to initialize an array?

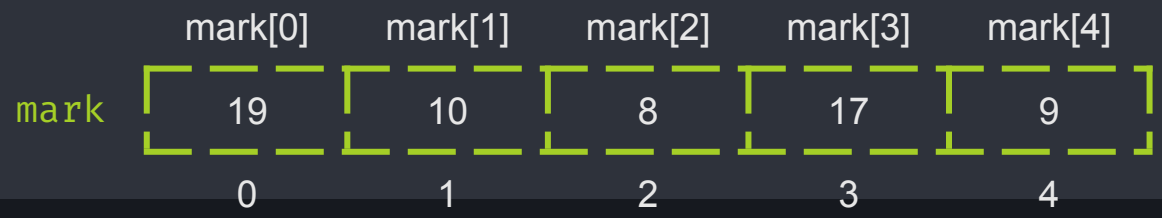
It is possible to initialize an array during declaration. For example,

```
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.



https://www.programiz.com/c-programming/c-arrays

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.

Few keynotes:

* 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 integer is 4 bytes.

https://www.programiz.com/c-programming/c-arrays

Change Value of Array elements

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


// make the value of the third element to -1
mark[2] = -1;


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

https://www.programiz.com/c-programming/c-arrays

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]);
```

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

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

https://www.programiz.com/c-programming/c-arrays

Example 1

Array Input/Output

```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;
}
```

Exercise

01. Write a program to find the average of n numbers using arrays

02. Write a program to sort an array element in ascending order. Ex: int num = {12,45,67,14,5,25};

https://www.programiz.com/c-programming/c-arrays

Access elements out of its bound!

Suppose you declared an array of 10 elements. Let's say,

```
int testArray[10];
```

You can access the array elements from testArray[0] to testArray[9].

Now let's say if you try to access testArray[12]. The element is not available. This may cause unexpected output (undefined behavior). Sometimes you might get an error and some other time your program may run correctly.

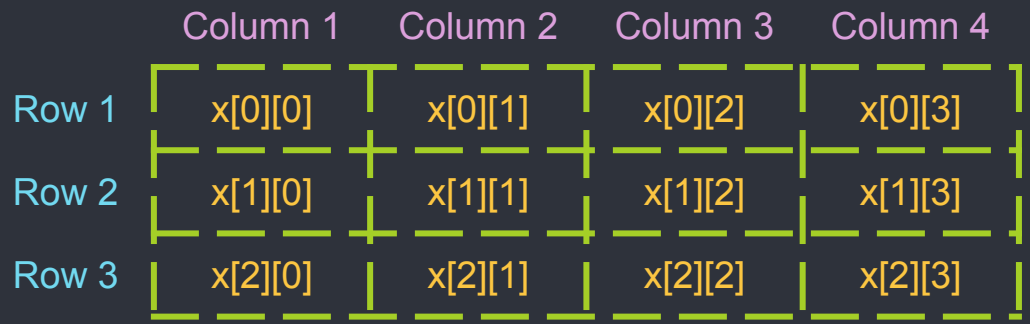
Hence, **you should never access elements of an array outside of its bound.**

Multidimensional arrays

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

```
float 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.



https://www.programiz.com/c-programming/c-multi-dimensional-arrays

Initializing a multidimensional array

Here is how you can initialize two-dimensional arrays:

Initialization of a 2d array

```
// Different ways to initialize two-dimensional array
int c[2][3] = {{1, 3, 0}, {-1, 5, 9}};
int c[][3] = {{1, 3, 0}, {-1, 5, 9}};
int c[2][3] = {1, 3, 0, -1, 5, 9};
```

https://www.programiz.com/c-programming/c-multi-dimensional-arrays

Example: Sum of two matrices

```c
// 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;
}
```

```
Thanks; {

    ‘Do you have any questions?’

        < bgamage@sjp.ac.lk >

}
```