Programming Fundamentals Lab

Lab # 11

Arrays in C

Instructor: Fariba Laiq

Email: fariba.laiq@nu.edu.pk

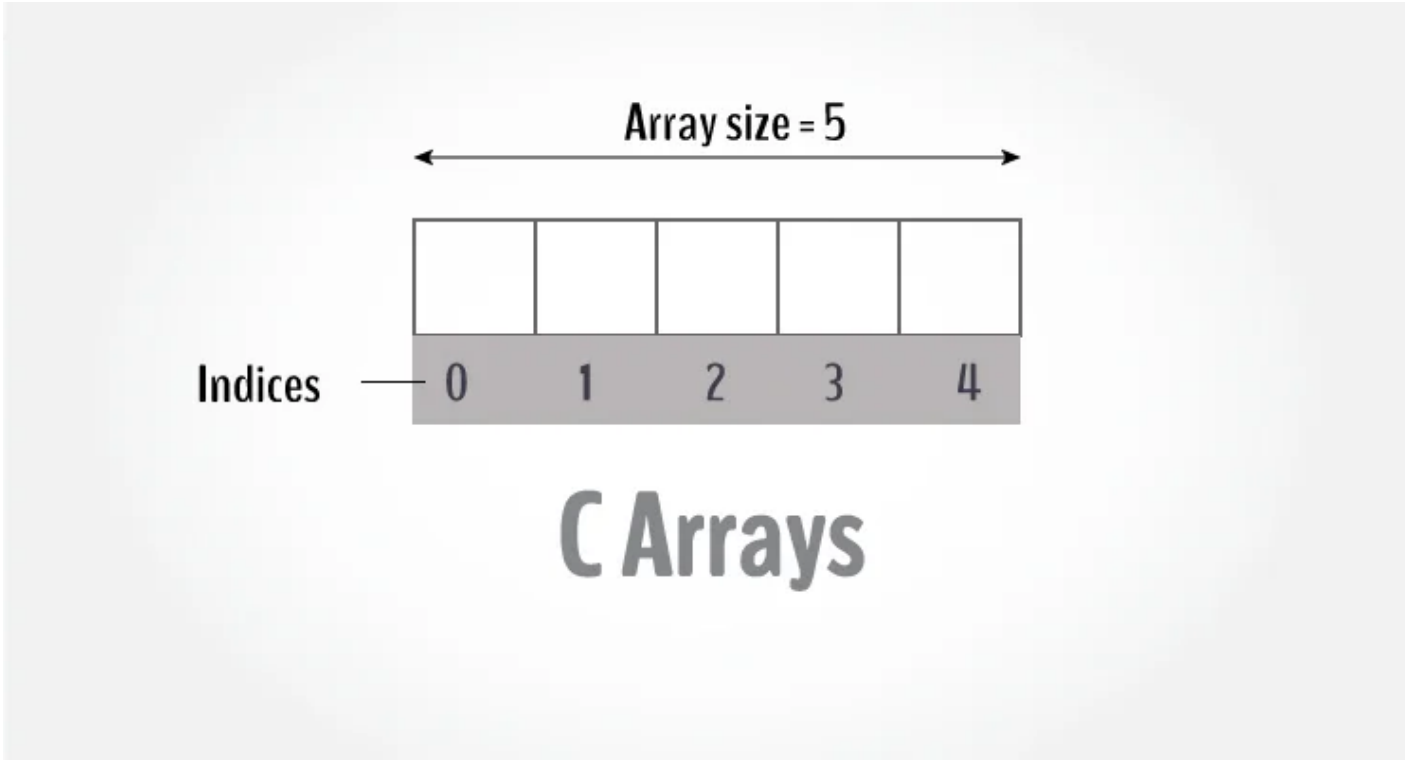
Course Code: CL1002

Semester Fall 2022

Department of Computer Science, National University of Computer and Emerging Sciences
FAST Peshawar Campus

Arrays in C/C++

1. It is a group of variables of similar data types referred to by a single element.
2. Its elements are stored in a contiguous memory location.
3. The size of the array should be mentioned while declaring it.
4. Array elements are always counted from zero (0) onward.
5. Array elements can be accessed using the position of the element in the array.
6. The array can have one or more dimensions.



An array in C/C++ or be it in any programming language is a collection of similar data items stored at contiguous memory locations and elements can be accessed randomly using indices of an array.

How to declare an array?

```
dataType arrayName[arraySize];
```

For example,

```
float mark[5];
```

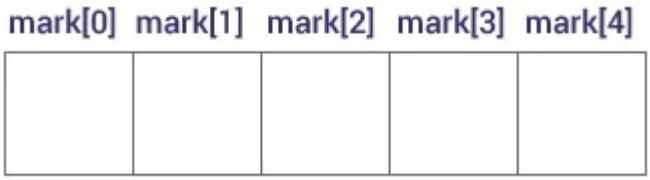
Here, we declared an array, mark, of floating-point type. And its size is 5. Meaning, it can hold 5 floating-point values.

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.



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]**

How to initialize an array?

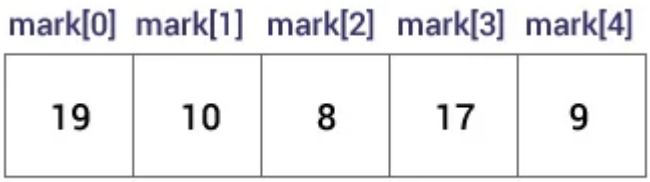
It is possible to initialize an array during declaration. For example,

```
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.



Here,

mark[0] is equal to 19

mark[1] is equal to 10

mark[2] is equal to 8

mark[3] is equal to 17

mark[4] is equal to 9

Change Value of Array elements

```
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;
```

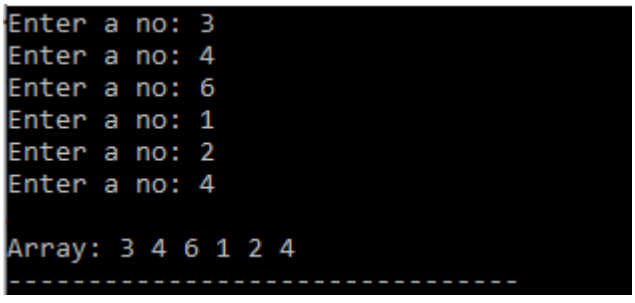
Example 1 | Array Input/Output

```
#include <stdio.h>
int main() {
	const int size=6;
	int arr[size];

	// array input
	for(int i=0; i<size; i++)
	{
		printf("Enter a no: ");
		scanf("%d", &arr[i]);
	}

	// array output
	printf("\nArray: ");
	for(int i=0; i<size; i++)
	{
		printf("%d ", arr[i]);
	}
	return 0;
}
```

Output:



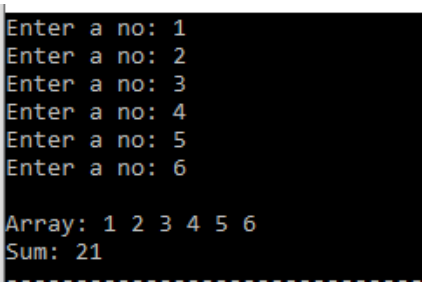
Example 2 | Calculate Sum

```
#include <stdio.h>
int main() {
	const int size=6;
	int arr[size];
	int sum=0;
	// array input
	for(int i=0; i<size; i++)
	{
		printf("Enter a no: ");
		scanf("%d", &arr[i]);
	}

	// array output
	printf("\nArray: ");
	for(int i=0; i<size; i++)
	{
		printf("%d ", arr[i]);
	}

	// calculating sum
	for(int i=0; i<size; i++)
	{
		sum=sum+arr[i];
	}
	printf("\nSum: %d", sum);
	return 0;
}
```

Output:

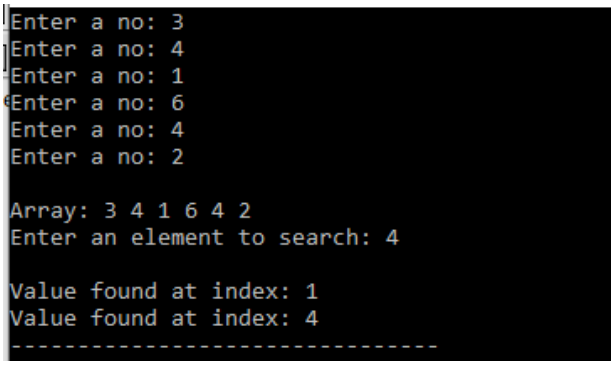


Search an Element in the Array

```
#include <stdio.h>
int main() {
	const int size=6;
	int arr[size];
	int sum=0;
	int value;
	// array input
	for(int i=0; i<size; i++)
	{
		printf("Enter a no: ");
		scanf("%d", &arr[i]);
	}
	// array output
	printf("\nArray: ");
	for(int i=0; i<size; i++)
	{
		printf("%d ", arr[i]);
	}

	// searching an element in array
	printf("\nEnter an element to search: ");
	scanf("%d", &value);
	for(int i=0; i<size; i++)
	{
		if(arr[i]==value)
		{
			printf("\nValue found at index: %d", i);
		}
	}
	return 0;
}
```

Output:



Passing an Array to a Function

To pass an entire array to a function, only the name of the array is passed as an argument. However, notice the use of [] in the function definition. This informs the compiler that you are passing a one-dimensional array to the function. As the array name contains the address of the first element. Therefore when passing array to a function, it is always passed by reference. Means any changes done to the array in that function will also be reflected in main or any part of the program. No copy is created in that case.

The findMin() function uses a for loop to iterate through the array and find the minimum value. It starts by assuming that the first element in the array is the minimum value, and then compares it with every subsequent element in the array. If it finds an element that is smaller than the current minimum, it updates the minimum value. Finally, the function returns the minimum value.

The findMax() function is similar to findMin(), but it finds the maximum value in the array. It starts by assuming that the first element in the array is the maximum value, and then compares it with every subsequent element in the array. If it finds an element that is larger than the current maximum, it updates the maximum value. Finally, the function returns the maximum value.

In the main() function, an integer array of size 6 is defined and initialized with some random integer values. The findMin() and findMax() functions are then called with this array as input, and the minimum and maximum values are stored in the min and max variables, respectively. Finally, the program prints out the minimum and maximum values using the printf() function.

Example Code:

```
#include <stdio.h>
int findMin(int arr[], int size)
{
	int min= arr[0];
	for(int i=0; i<size; i++)
	{
		if(arr[i]<min)
		{
			min=arr[i];
		}
	}
	return min;
}
int findMax(int arr[], int size)
{
	int max= arr[0];
	for(int i=0; i<size; i++)
	{
		if(arr[i]>max)
		{
			max=arr[i];
		}
	}
	return max;
}
int main() {
	const int size=6;
	int arr[]={2,4,1,6,7,9};
	int min=findMin(arr, size);
	int max=findMax(arr, size);
	printf("\nMin: %d", min);
	printf("\nMax: %d", max);
	return 0;
}
```

Output:

Min: 1

Max: 9