Arrays

What is an Array?

Arrays are referred to as structured data types. An array is defined as finite ordered collection of homogenous data, stored in continuous memory locations.

- **finite** means data range must be defined.
- **ordered** means data must be stored in continuous memory addresses.
- **homogenous** means data must be of similar data type.

An array is also called as **a method of clubbing multiple entities of similar type into a larger group**. These entities or elements can be of int, float, char, or double data type.

Uses of an Array

- to store list of Employee or Student names,
- to store marks of students,
- or to store list of numbers or characters etc.

Since arrays provide an easy way to represent data, it is classified amongst the data structures in C.

Advantages of Array

- In one go, we can initialize storage for more than one value. Because you can create an array of 10, 100 or 1000 values.
- They make accessing elements easier by providing random access. By random access we mean you can directly access any element in an array if you know its index.
- Sorting and searching operations are easy on arrays.

Disadvantages of Array

- Due to its fixed size, we cannot increase the size of an array during runtime. That means once you have created an array, then it's size cannot be changed.
- We cannot enter different types of inputs in a single arrays.
- Insertion and deletion of elements can be costly, in terms of time taken.

How to declare an array?

- Syntax – dataType arrayName[arraySize];

For example - int mark[5];

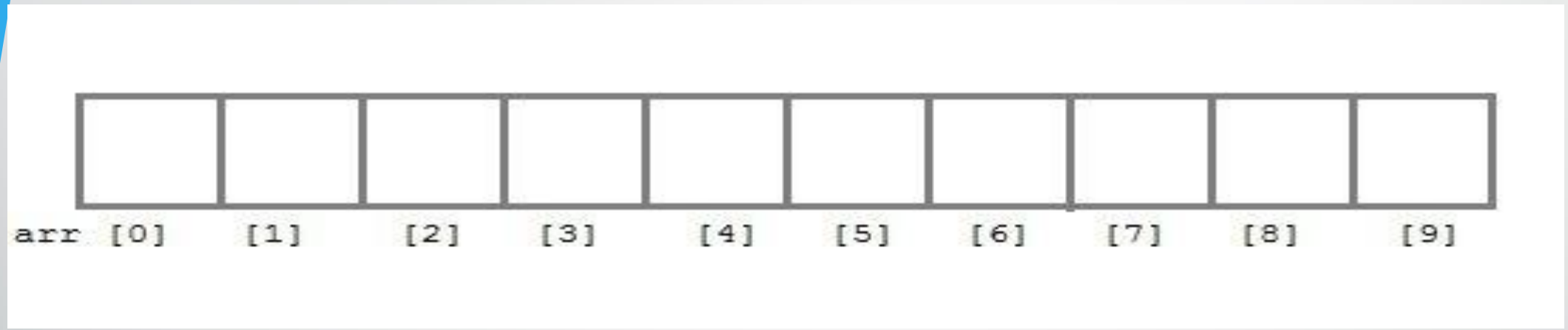
Here, we declared an array, **mark** of int type, And its size is 5. Meaning, it can hold 5 integer values.

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

Example of array declaration

- char a[5];
- float ar[9];
- int arr[10];

Indexing of array



- The index of an array starts from 0 to size-1 i.e first element of any array will be stored at arr[0] address and the last element will be at arr[size - 1].

Initialization of an array?

- An array can be initialized at either compile time or at runtime.
- That means, either we can provide values to the array in the code itself, or we can add user input value into the array.
- **Compile time** initialization of array means we provide the value for the array in the code, when we create the array,

Syntax - data-type array-name[size] = { list of values };

- **Runtime** initialization of an array can be done using scanf() function. This approach is usually used for initializing large arrays, or to initialize arrays with user specified values.

```
scanf("%d", &arr[3]);       // will insert element at index 3, i.e. 4th position
```

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.

Input Array Elements

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

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


// take input and store it as i th element
for(int i = 0; i < 10; i++)
    scanf("%d", &mark[i]);
```

Accessing Array Elements

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 all the elements of arrays
for(int i = 0; i < 10; i++)
    printf("%d", mark[i]);
```

Access array elements

- 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 it’s bound.

Example 1: Array Input/Output

```
// 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;
}
```

How to 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 of Array

```
// 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;
}
```

Passing array to a function

calculate the sum of array elements by passing to a function

```
#include <stdio.h>
float calculateSum(float num[]);
int main() {
  float result, num[] = {23.4, 55, 22.6, 3, 40.5, 18};
  // num array is passed to calculateSum()
  result = calculateSum(num);
  printf("Result = %.2f", result);
  return 0;
}
float calculateSum(float num[]) {
  float sum = 0.0;
  for (int i = 0; i < 6; ++i) {
    sum += num[i];
  }
  return sum;
}
```