

Chapter 7 - Arrays

An array is a collection of similar ^{imp} elements.

One variable \Rightarrow Capable of storing multiple values

Syntax

The syntax of declaring an Array looks like this:

int marks[90]; \Rightarrow Integer array

char name[20]; \Rightarrow Character array or String

float percentile[90]; \Rightarrow float array

The values can now be assigned to marks array like this:

marks[0] = 33;

marks[1] = 12;

Note: It is very important to note that the array index starts with 0.

Marks \rightarrow

7	6	21	3	91	3
---	---	----	---	----	---

 =

88	89
----	----

0 1 2 3 4 5 ... 88 89

Total = 90 elements

Accessing elements

Elements of an array can be accessed using:

`scanf ("%d", &marks[0]);` \Rightarrow Input first value

`printf ("%d", marks[0]);` \Rightarrow output first value of the array

Quick Quiz \rightarrow Write a program to accept marks of five students in an array and print them to the screen.

Initialization of an Array

There are many other ways in which an array can be initialized.

`int cgpa[3] = { 9, 8, 8 };` \Rightarrow Arrays can be initialized while declaration
`float marks[] = { 33, 40 };`

Arrays in memory

Consider this array:

`int arr[3] = { 1, 2, 3 };` \Rightarrow 1 integer = 4 bytes

This will reserve $4 \times 3 = 12$ bytes in memory
4 bytes for each integer.

1	2	3
62302	62306	62310

\Rightarrow arr in memory

Pointer Arithmetic

A pointer can be incremented to point to the next memory location of that type.

Consider this example

```
int i = 32;
```

```
int *a = &i;  $\Rightarrow$  a = 87994 address  $\rightarrow$  87994
```

```
a++;  $\Rightarrow$  Now a = 87998
```

```
char a = 'A';
```

```
char *b = &a;  $\Rightarrow$  b = 87994
```

```
b++;  $\Rightarrow$  Now b = 87995
```

```
float i = 1.7;
```

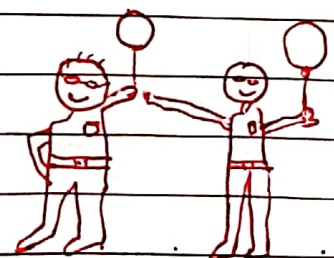
```
float *a = &i;  $\Rightarrow$  Address of i or a = 87994
```

```
a++;  $\Rightarrow$  Now a = 87998
```

Following operations can be performed on a pointers:

1. Addition of a number to a pointer
2. Subtraction of a number from a pointer
3. Subtraction of one pointer from another
4. Comparison of two pointer variables

Quick Quiz \rightarrow Try these operations on another variable by creating pointers in a separate program. Demonstrate all the four operations.



Yay! we understood
pointer arithmetic

Accessing Arrays using pointers

Consider this array

	7	9	2	8	1
index	0	1	2	3	4
	↑ ptr				

If ptr points to index 0, ptr++ will point to index 1 & so on...

This way we can have an integer pointer pointing to first element of the array like this:

```
int *ptr = &arr[0]; → or simply arr
ptr++;
*ptr ⇒ will have 9 as its value
```

Passing arrays to functions

Arrays can be passed to the functions like this

```
printArray(arr, n); ⇒ function call
```

```
Void printArray(int *i, int n); ⇒ function prototype
```

```
or
Void printArray(int i[], int n);
```


Multidimensional Arrays

An array can be of 2 dimension / 3 dimension / n dimensions

A 2 dimensional array can be defined as:

```
int arr[3][2] = { { 1, 4 }  
                 { 7, 9 }  
                 { 11, 22 } };
```

We can access the elements of this array as
arr[0][0], arr[0][1] & so on...

Value = 1

Value = 4

2-D arrays in Memory

A 2d array like a 1-d array is stored in contiguous memory blocks like this:

arr[0][0] arr[0][1] ...

1	4	7	9	11	22
---	---	---	---	----	----

87224 87228 ..

Quick Quiz: Create a 2-d array by taking input from the user. Write a display function to print the content of this 2-d array on the screen.