***Chapter 7 - Arrays***

An array is a collection of similar elements.

One variable => Capable of storing multiple values

**Syntax,**

The syntax of declaring an array looks like this:

int marks[90]; => integer array

char name[20]; => character array or string

float percentile[90]; => float array

Copy

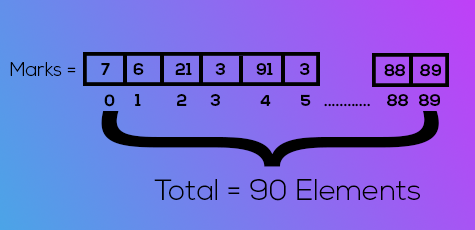
The values can now be assigned to marks array like this:

marks[0]=33;

marks[1]=12;

Copy

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



**Accessing Elements**

Elements of an array can be accessed using:

scanf(“%d”,&marks[0]); => Input first value

printf(“%d”, marks[0]); => Output first value of the array

Copy

**Quick Quiz:**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} => Arrays can be initialised while declaration

float marks[]={33,40}

Copy

**Arrays in memory**

Consider this array:

Int arr[3]={1,2,3} => 1 integer = 4 bytes

Copy

This will reserve 4x3=12 bytes in memory. 4 bytes for each integer.

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |

62302      62306    62310                             {Array 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; ==> a =87994 (address = 87994)

a++; ==> now a = 87998

char a = 'A';

char \*b = &a; ==> b = 87994

b++; ==> now b = 87995

float i = 1.7;

float\*a = &i; ==> Address of i or a = 87994

a++; ==> Now a = 87998

Copy

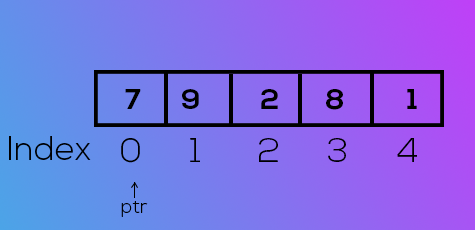
**Following operations can be performed on pointers:**

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:**Try these operations on another variable by creating pointers in a separate program. Demonstrate all four operations.

**Accessing arrays using pointers**

Consider this array,



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

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

int \* ptr = & arr[0]; => or simply arr

ptr++;

\*ptr => will have 9 as it’s value

Copy

**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);

Copy

**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}

};

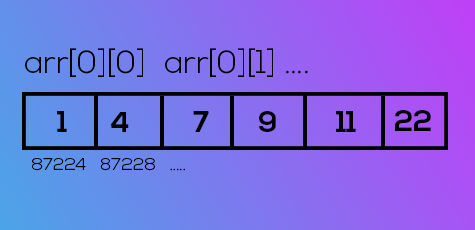
Copy

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

At arr [0] [0] value would be 1 and at arr [0] [1] value would be 4.

**2-D arrays in Memory**

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



**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.

Chapter 7- Practice Set

1. Create an array of 10 numbers. Verify using pointer arithmetic that (ptr+2) points to the third element where ptr is a pointer pointing to the first element of the array.
2. If S[3] is a 1-D array of integers, then \*(S+3) refers to the third element:

* True
* False
* Depends

1. Write a program to create an array of 10 integers and store a multiplication table of 5 in it.
2. Repeat problem 3 for a general input provided by the user using scanf()
3. Write a program containing a function that reverses the array passed to it.
4. Write a program containing functions that counts the number of positive integers in an array.
5. Create an array of size 3x10 containing multiplication tables of the numbers 2,7 and 9, respectively.
6. Repeat problem 7 for a custom input given by the user.
7. Create a three-dimensional array and print the address of its elements in increasing order.