Arrays

Introduction:→

Need ?

- Consider a scenario where you have to input marks of 20 students & display them.
- → You will declare 20 integer variables
  - → 20 scanf / Read statements
  - → 20 print / write statements.

} 40 statement

mark 1 — mark 2 — marks 3 — marks 4
mark 5 — 6 — 7 — 8
— 9 — 10 — 11 — 12
— 13 — 14 — 15 — 16
— 17 — 18 — 19 — 20

} 20 Inputs (Marks)

- As you can observe above reading 20 inputs & display them will take total 40 statements, making code very large and complex, which is Not feasible actually. To overcome this problem we will use array.

Array:→ An array is a collection of similar type of data elements (homogeneous elements) stored in consecutive memory locations under the same name. These elements of array are accessed / refered by their

index value / subscript / offset.

Declaration:-

- Datatype – int, char, float, double etc.
- Name of Array variable – to identify array
- size – Max. No. of elements an array can hold.

Syntax:- datatype name [SIZE];

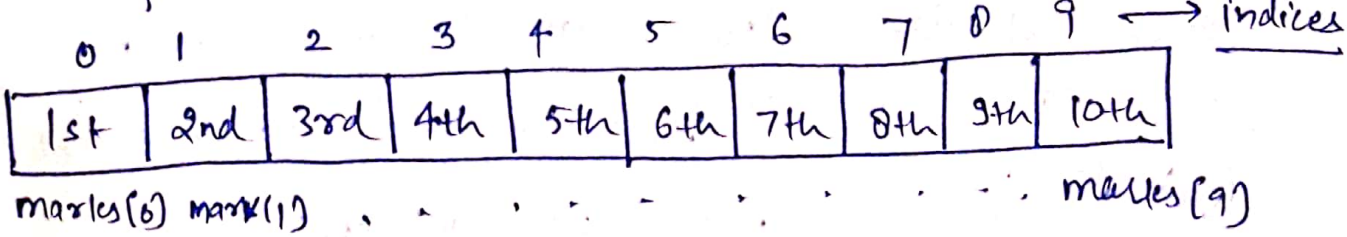
→ size of array is constant & must have a value at compile time.

Ex:- int marks [10];

→ marks is an array variable of 10 elements.

→ In C, the array index starts from 0 (Zero).

→ So first element at marks [0], the 2nd at marks [1], ... & last at marks [9]



Memory Calculation:-

1.) int b [6];

Total Memory Array elements will take	=	size of Array * size of each element

Hence it will take $6 * 4 = \boxed{24 \text{ Bytes}}$

(6 → size of array; 4 → size of integer)

2.) float a[5];

Total Memory = $5 * 4 = \boxed{20 \text{ Bytes}}$

(5 → size of Array; 4 → size of float)

3.) char c[6] = $1 * 6 = \boxed{6 \text{ Bytes}}$

Some important points :-

(1.) Illegal array declaration :-

- int arr[]; ✗ — Compile Time error
- int n, arr[n]; ✗ → Run time error as n have garbage value and hence garbage size array

(2.) ✓ using macros (#define)

```
#define N 100
void main()
{   int arr[N];
    ...
}
```

equivalent code ⇒

```
void main()
{
    int arr[100];
    ....
}
```

(iii) Size of array can be an expression.

```
#define N 100
Void main()
{
    int i=10;
    int arr[N+10], my_arr[i-5*10];
}
```

↓ Error

(iv) There is No boundary checking in Array at compile time or Run-time. (So it won't check validity of index)

Ex:- `int arr[100];`

`printf("%d", arr[105]);` → output will be Any garbage value

Accessing elements of the Array:-

Subscript / index / offset is used to access the array elements.

Syntax:- Array name [index].

Ex:- To Access 4th element of an Array arr

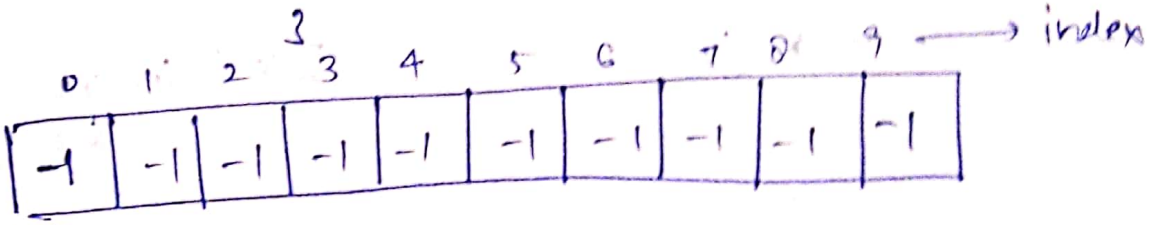
arr[3]

Storing values in Array

→ No single operation that can operate on all elements of array.

→ Hence we will use loops & can access elements of array by varying the value of index.

Ex:-
```
int i, marks[10];
for(i=0; i<10; i++)
                or
{               i<=9
    marks[i] = -1;
```

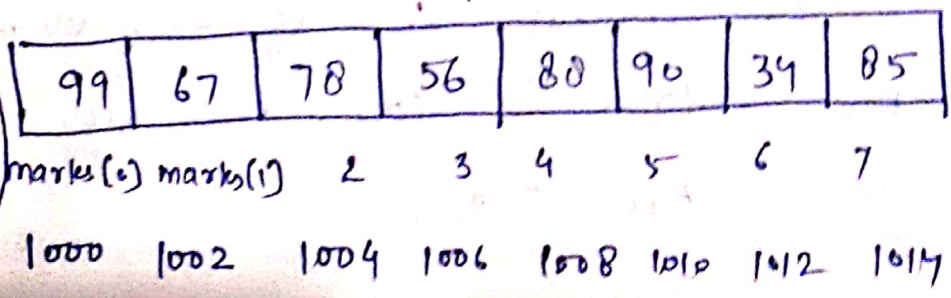


Name of array refers address of first element/value.

Calculating Address of the Array elements:-

Ex:- int marks[8] = {99, 67, 78, 56, 80, 90, 34, 85}

Calculate address of marks[4] if base address is 1000 (Address of first element)



As ~~size~~ Size of int is 2 or 4 Byte, hence we add +2 or +4 in base Address

if we check address of marks[4] we can find it as 1008 according to the memory map.

Using formula:-

Address of data element A[i] = Base Address + W * (i − Lower bound)

- A[i]: Array
- i: index for which Address Need to be found
- W: Size of each element

Hence Address of marks[4] = 1000 + 2 * (4 − 0)

↳ Lower bound / Starting index

Add. of marks[4] = 1008

→ use %u to print address (unsigned int) because address is always positive. & physically exists.

Ex(2). Suppose you have an array Arr[500] of 500 elements. Size of each element is 8 Bytes, Base address is 100. Find the Address of Arr[350].

Address of
Arr[350] = 100 + 8 * (350 - 0)

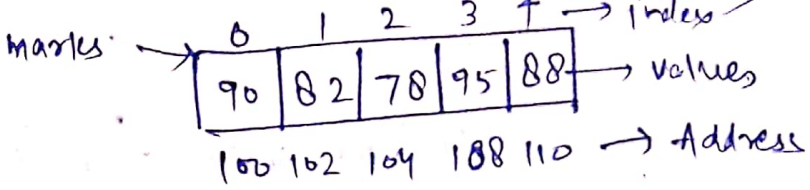
= 100 + 2800

Add. of Arr[350] = 2900

Initialization of Array :-

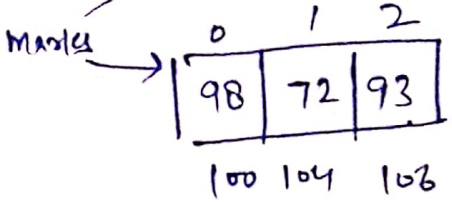
1.) Compile time Initialization :-

Ex:- (1.) int marks[5] = {90, 82, 78, 95, 88};



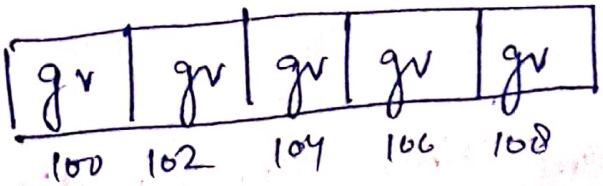
(2.) int marks[] = {98, 72, 93};

size is optional at compile time



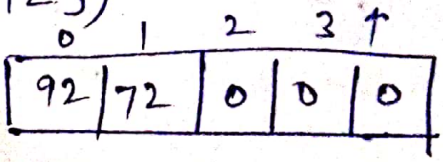
(3.) int marks[5];

initially all values would be garbage, if not initialized



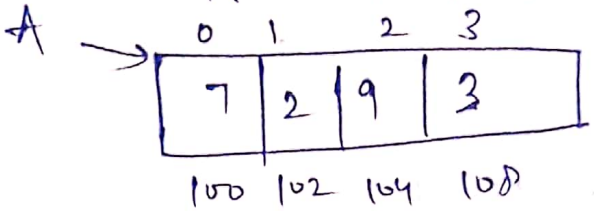
(4.) int marks[5] = {92, 72};

In partial initialization Remaining values becomes 0.



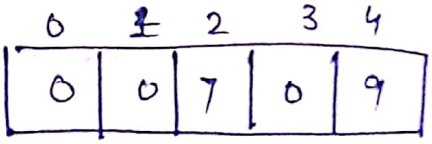
(5.) int A[4] = { 7, 2, 9, 3, 6, 8 };

here we have 6 initializers but only 4 size (warning) but it will store only first 4 values leaving rest 2 at last



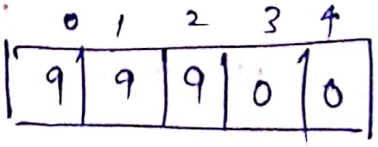
(6.) Indexed initialization:-

Ex:- (1) int arr[5] = { [2] = 7, [4] = 9 };



will set value 7 at index 2 and 9 at index 4 keeping rest as 0

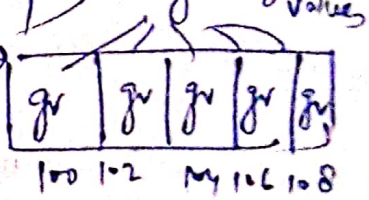
Ex (2) int arr[5] = { [0 ... 2] = 9 };



↳ for Range starting 0 to last index 2 set 9

2.) Run-Time Initialization:- (using loops)

Ex:- int a[5], i;

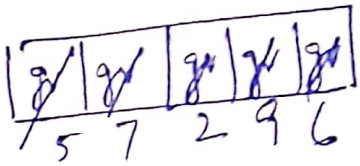


Input → for (i=0; i<5; i++)
scanf("%d", &a[i]);

i	&a[i]
0	&a[0] = 100
1	&a[1] = 102
2	&a[2] = 104
3	&a[3] = 106
4	&a[4] = 108

we will get these address using scanf in loop to insert values

Let values are 5, 7, 2, 9, 6



Output:- To print values

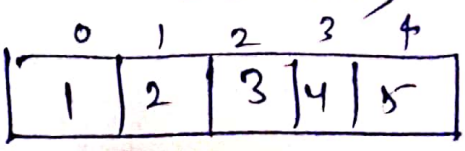
To print values
```
for (i=0; i<5; i++)
    printf("%d\n", a[i]);
```
→ 5
7
2
9
6

To print Addresses
```
for(i=0; i<5; i++)
    printf("%u\n", &a[i]);
```
100
102
104
106
108
} +2 or +4 in Add. due to int.

(3.) Initialization By Assignment

For Ex!.
```
int a[5], i;
for(i=0; i<5; i++)
    a[i]= i+1;
```

a →



Ques:- Suppose indexes of an array are given as (Lower Bound → -5, Upper Bound → 7)

A [-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7]

Address of first element is 50, size of each element is 2 Bytes. find Address of A[3].

Solution:

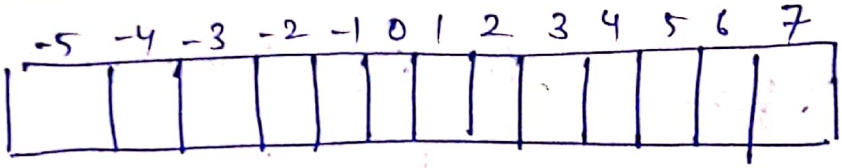
Add. of A[3] = Base Address + W * [i - Lower bound]

= 50 + 2 * [3 - (-5)]

(Lower Bound according to problem given)

= 50 + 16

Add. of A[3] = 66



size of Array will be / Length

Length = Upper Bound - Lower Bound + 1

= 7 - (-5) + 1

Size/Length = 13 ✓

Q(1) WAP to input an Array of 5 elements & display it.

Sol:

```
#include <stdio.h>
Void main()
{
    int A[5], i;
    printf("Enter Array elements");
    for(i=0; i<5; i++)
    {   printf("Enter element No. %d", i+1);
        scanf("%d", &A[i]);
    }
printf("Array is given by ");
for(i=0; i<5; i++)
    {
        printf("%d\n", i+1);
    }
}
```

Q(2) WAP to input an array and find the sum of all elements of that array. and average

Sol:

```
#include <stdio.h>
Void main()
{
    int A[100], n, i, s=0;
    float Avg;
```

→ declared array of size 100.

printf(" Enter Size "); → upto limit 100 as we have 100 max size

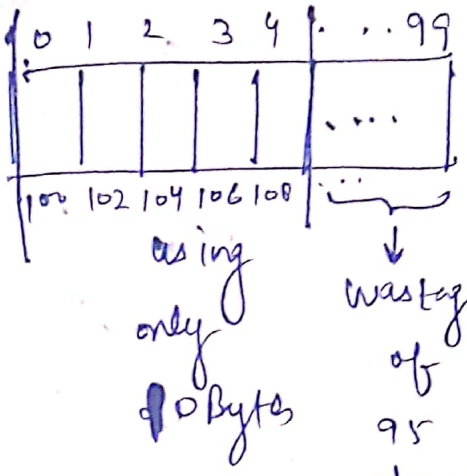
scanf("%d", &n); // Let n be 5

printf("Enter %d elements", n);

for(i=0; i<n; i++)
{
    scanf("%d", &A[i]);
    S = S + A[i];
}

avg = (float)S / n;

printf(" Sum = %d \n Average = %f", S, avg);



}

Q(3): WAP to find maximum and minimum element in an array.

Sol:

```
#include <stdio.h>
void main()
{
    int A[100], n, max, min, i;
    printf("Enter size");
    scanf("%d", &n);
    printf("Enter %d no. of elements", n);
    for(i=0; i<n; i++)
        scanf("%d", &A[i]);
```

```
max = A[0]; // Let first element is max.
min = A[0]; // ____________ is minimum

for (i=1; i<n; i++)   // checking Remaining elements.
{
    if ( A[i] > max)
        max = A[i];
    if ( A[i] < min)
        min = A[i];
}
printf(" Maximum = %d \n Minimum = %d ", max, min);
}
```