Arrays.

I. Declaration of Arrays

Arrays ⇒ An array is a collection of elements of the same type that are reference by the common name.

Syntax: Data type Array name Array [size]; // Declaration of the array.

Ex: Int a[10].

Above Example is of 1-D array. 1-D array means 1 row to multiple columns.

Ex
Char c[10];
Float b[10];

1, 2, 3, 4, 5 ✓

10, 20, 0, -1 30 ✓

10, 20, 1.2, 2.2 ✗ // Datatypes are not same.

Array is also called Collection of homogeneous Elements in which are stored in contigeous memory Location.

Points to remember.

→ Int a[ ] : This would result with error. Always. Size should be specified.

→ Size should be of Integer data type. (Positive)

int a[5] ✗
int a[5] ✓
int a[2+2] ✓
int a[7*2] ✓

int a[]; ✗.
int a[1/2] ✗
int a[n] ✗ // we can use macros. before main() write #define n 5;
Now int a[n]; ✓

II Array Initialization. ⇒ Arays may be Initialised at time of declaration.

Arrays can be Initialised in two ways.

1) Compile time.
2) Run time.

a) Compile time :→ At the time of declaration itself. We specify data Elements

OR

Specifying data items at the time of declaration is known as Compile-time Initialisation.

Ex : ✓ int a[5] = {0, -1, 2, 3, 5} // Initialisation of array at the Compile time.

✓ int a[ ] = {0, 1, 2, 3, -1, 5}
Automatically size is Calculated.

✓ int a[5] = {0, 1, 2};
We have left with space for 2 data items by default its filled with zero in memory allocation.

✓ int a[5];
Now array will be filled with garbage value.

int a[5] = {1, 2, 3, 4, 5, 6, 7}
Elements Exceeds array size. This results with Error as we can store only 5 numbers.

✓ int a[5];

a[0] = 1
a[1] = 2
a[2] = 3
a[4] = 5
a[5] = 7

// Individual Items can be Initialised like this. This is not Preferable, ofcourse its correct.

✓ int a[5] = {0}; // All memory locations would be filled with zero.

✗ int a[5] = { }; // This gives error. Always need to specify size atleast one data item should not Leave blank.

✓ Char b[ ] = { 'S', 'K', I, T }

Here in Compile time we have fixed the size of the array. we can't Change it. also we have fixed Elements & cannot be Changed at Run time.

To avoid above disadvantage to ask the users to Enter the data. This is knowras Run-time Initialisation

b)

At Run-time : Initialising array make use of Iterative Statements with Scanf().

int a[5]; // Ask user to enter data.

Printf("Enter elements for array");

for(i=0; i<5; i++)

{ Scanf("%d", &a[i]);

}

to print we can use print data.

Note: when size of array is Large, its always better to go with Runtime Initialisation.

Ex:
```
Int a[100];
For (i=0; i<100; i++)
{
    if (i<30)
        a[i] = 1;
    if (i>30 && i<=100) //OR simply use else.
        a[i] = 0;.
}
```

iii]

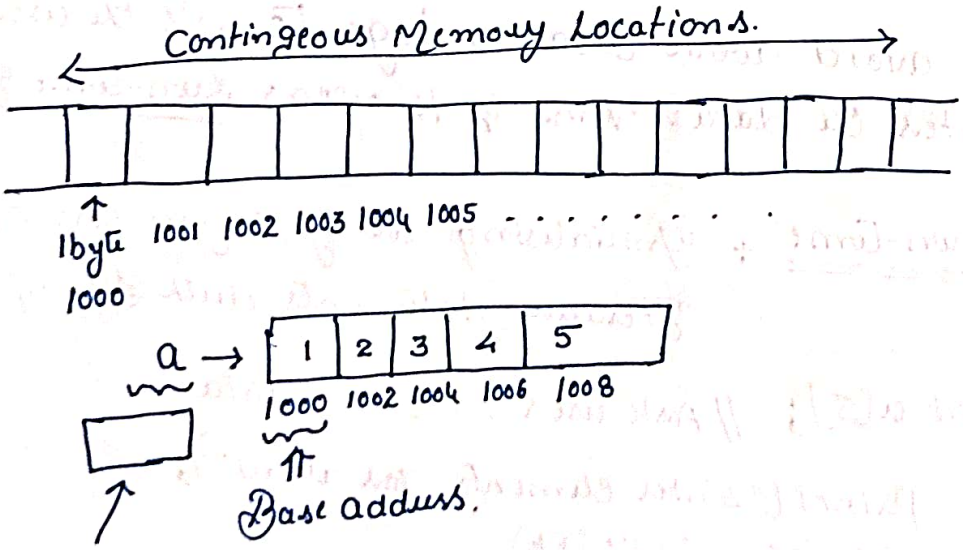
Memory Representation & accessing of arrays

int a[5] = {1, 2, 3, 4, 5}.

1 Integer takes 2-byte. Some m/c takes 4-bytes.

∴ 2 * 5 Elements ⇒ 10 bytes.

Note: always memory is allocated in contigeous memory Location.



Array name stores address of 1st Element also known as base address.

Any variable that stores address of other variable is known as pointer.

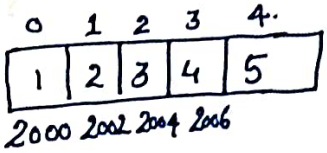
Hence array name is pointer variable or Internal Pointer.

ie array name is pointer variable.

Accessing Elements of array (1D)

Syntax ⇒ arrayname[index]

Ex: a[0] = 1
a[1] = 2
a[2] = 3.



Now Question is how address is Calculated.

Formula ⇒ Base address + Index * Size of int

Ex: To access 4th Element ⇒ 2000 + 3 * 2.
⇒ 2000 + 6.
⇒ 2006

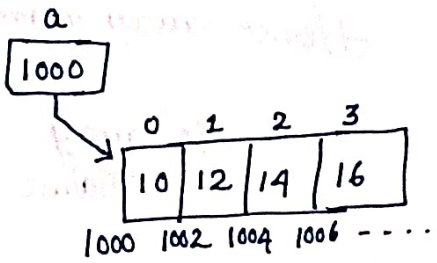
Points to remember.

1) Array is a collection of of more than one data item of same type.
2) All data items are stored in contigeous memory Location.
3) No of items array holds is size of array.
4) Once size has declared, cannot be changed at run time [fixed size]
5) Index starts from zero.
6) Known as derived data type.
7) Accessing an element is faster in arrays (Make use of Index)
8) Allow to store data in multidimensional form.
9) Inserting & deleting elements from array is costly.
10) No bound checking in C.

IV Operations on arrays.

a) Inserting an Element.

Ex:
```
void main()
{
  int a[5], i;
  for(i=0; i<5; i++)
  { scanf("%d", &a[i]);
  }
}
```



b) Printing an Element.

Ex:
```
void main()
{
  int a[5], i;
  for(i=0; i<5; i++)
  { printf("%d", a[i]);
  }
}
```

a[0]
Base add^{n} + i * Size of int.

a[2] → 1000 + 2*2.
1004

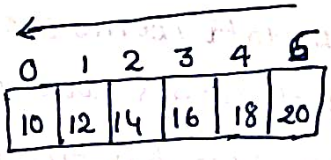
c) Traversing an array

Accessing arrays can be done in 3 ways.

→ From right to Left. (as shown above).

→ From Left to right

Ex:
```
for(i=5; i>=0; i--)
{ printf("%d", a[i]);
}
```



if i = -1 then stop traversing.

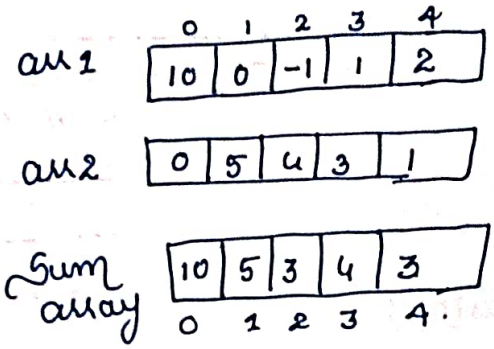
Ex: To calculate sum & average of marks.

```
#include <stdio.h>
void main()
{
    int marks[5], i;
    float sum=0, avg;
    for(i=0; i<5; i++)
    {
        scanf("%d", &marks[i]);
    }
    for(i=0; i<5; i++)
    {
        sum = sum + marks[i];
    }
    avg = sum/5;
    printf("sum = %d", sum);
    printf("\n average = %f", avg);
}
```

Ex: To read array of 10 Integers & count total number of even & odd elements.

```
void main()
{
    int a[10], even, odd=0;
    for(i=0; i<10; i++)
    {
        scanf("%d", &a[i]);
    }
    for(i=0; i<10; i++)
    {
        if(a[i]%2==0)
            even = even++;
        else
            odd = odd++;
    }
    printf("Even elements are
            %d", even);
    printf("Odd elements are
            %d", odd);
}
```

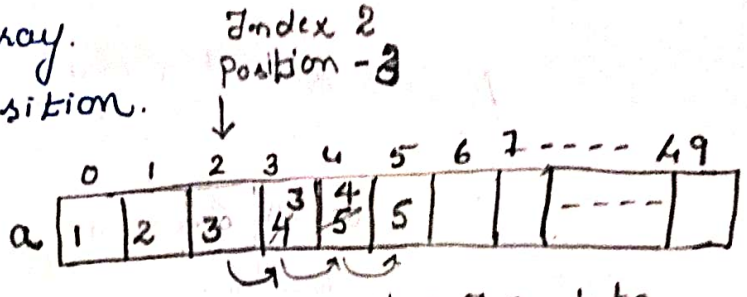
Prog to read two arrays of size 5 & store sum of these array in third array.




```
#include <stdio.h>
void main()
{
    int arr1[5], arr2[5], sumarr[5], i;
    for(i=0; i<5; i++)
        scanf("%d", &arr1[i]);
    for(i=0; i<5; i++)
        scanf("%d", &arr2[i]);
    for(i=0; i<5; i++)
    { sumarr[i] =
        arr1[i] + arr2[i];
      Printf("%d", sumarr[i]);
          OR
      Printf("Sum array element at index %d is = %d\n",
                                  i, sumarr[i]);
    }
}
```

IV] Inserting Element in an array.

a] Inserting at specific position.



Now I want to insert an element at position 2 then we need to shift array elements towards right.

```
Void main()
{
    Int a[50], size, i, num, pos;
    Printf("Enter size of array");
    Scanf("%d", &size);
    Printf("Enter elements of array");
    for(i=0; i<size; i++)
    {
        Scanf("%d", &a[i]);
    }
    Printf("Enter data U want to Insert");
    Scanf("%d", &num);
    Printf("Enter position");
    Scanf("%d", &pos);
    for(i = size-1; i>=pos-1; i--)
    {
        a[i+1] = a[i];
    }
    a[pos-1] = num;
    size++;
```

```
if(pos<0 || pos>size+1)
{ Printf("Invalid position");
}
```