Date:

- ARRAY -

① Defination -

Array is considered as linear data structure which holds homogeneous elements in index format.

② Data structures -

Data structures is way of storing & representing the data in perticular format.

There are too types of data structures as linear and non-linear.

③ Linear Data structure -

If the elements of data structure are stored sequentialy then it is considerd as linear data structure.

e.g - Array, linked list, stack, queue.

④ Non- linear Data structures -

If the elements of data structure

Date:

are stored in scattered format then it is a non-linear data structure.
e.g- Tree, Hashtable.

⑤ In case of array we can store only homogeneous elements, Homogeneous means every element is of same data type.

⑥ In case of array all the elements gets stored in a index format. The first index of array is always zero.
If array contains n elements then the index starts from zero to n-1

⑦ Practicle use of array -
If we want to store marks of five students then there are two ways in which we can write the program as.

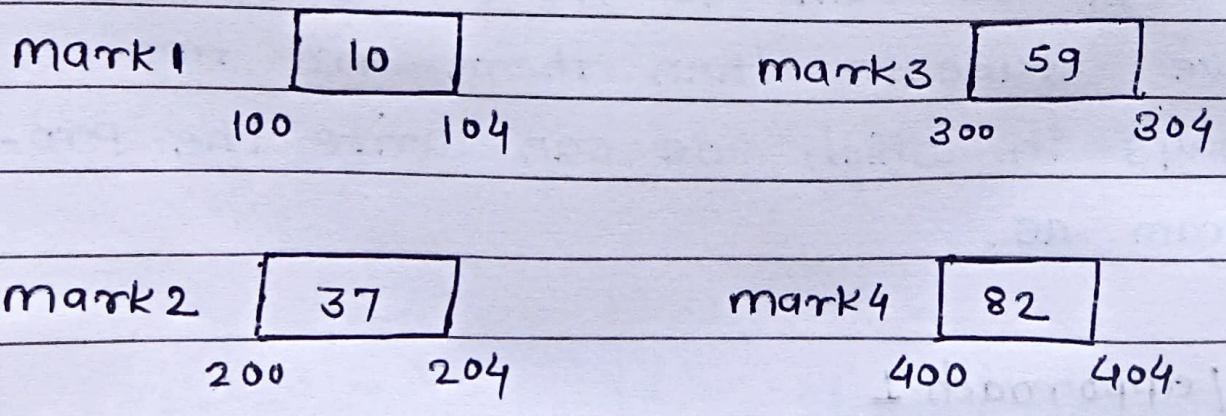
```
// approach 1
```

Date:

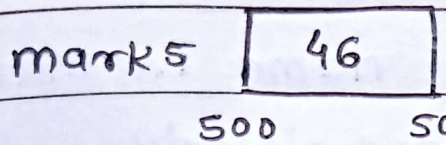
```
#include <stdio.h>
int main()
{
    int mark1 = 10;
    int mark2 = 37;
    int mark3 = 59;
    int mark4 = 82;
    int mark5 = 46;

    return 0
}
```

In above program we create 5 integer variable with different names for every variable seperate memory gets allocated as



Date:



According to above dig. the memory gets allocated for every variable in a scattered way.

As a programmer we have to remember the names of all variable's.

// approach 2

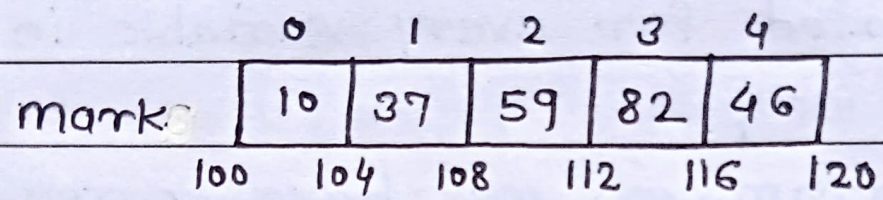
```
# include <stdio.h>
   int main()
{
   int mark [5] = {10,37,59,82,46};
   return 0
}
```

- In above program we create the array of integer.
- Due to this, there is no need to remember the names of every variable
- In this program we can acess as

Date:

the elements using single name i.e marks.

- The above array is diagramitically represented as.



⑧ There are two ways in which we can allocate the memory for array

1) static memory allocation
2) Dynamic memory allocation.

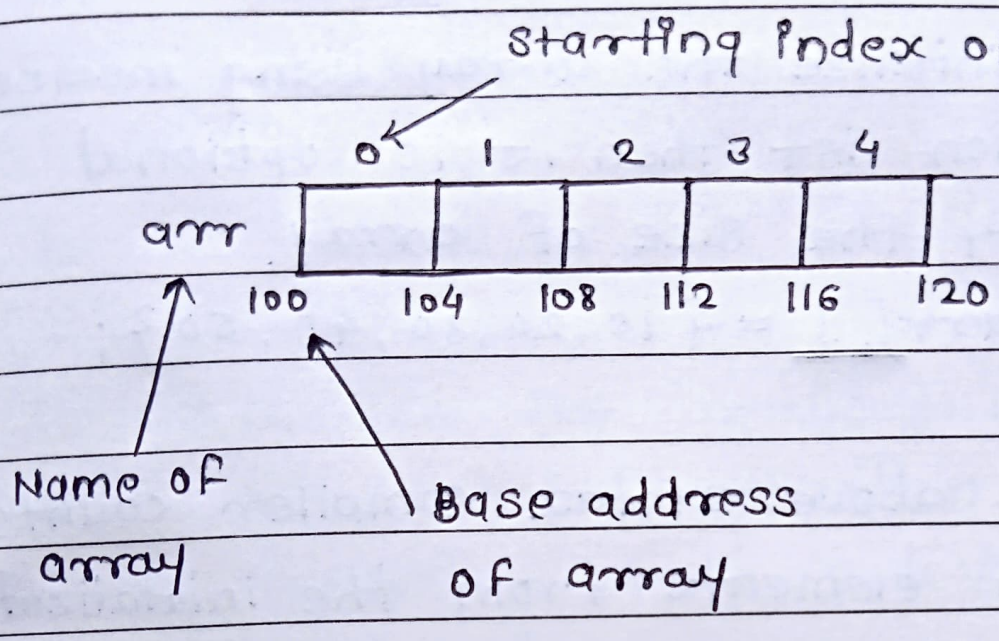
⑨ we can create the array statically by specifying the size of array in squre bracket [ ].

e.g- float arr [5];

↓ read.

arr is 1 dimentional array which contain 5 element's. each element is of type float.

Date:



⑩ we can initilize the members of array in two ways.

i) using member initialization list

eg- int arr[5] = {10, 20, 30, 40, 50};

member initialization list

ii) member by member initialization

eg- int arr[5];

```
arr[0] = 10;
arr[1] = 20;
arr[2] = 30;
arr[3] = 40;
arr[4] = 50;
```

member by member initialization

Date:

(11) If we initilize the array using member initalization list then it is optional to specify the size of array.

eg- int arr[] = {10,20,30,40,50};

In above syntax compiler counts the no of elements from the initialization list and allocate memory acordinaly.

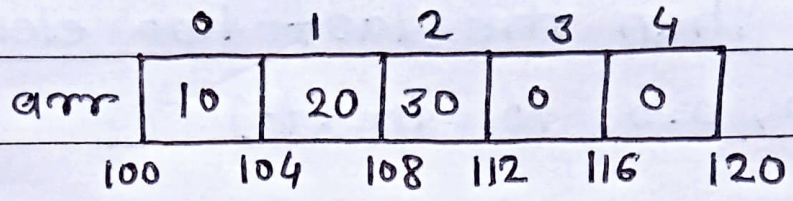
(12) If we are using member by member initialization technique then size of array is compulsary otherwise compiler will generate error.

e.g int arr[]; // error

(13) If we are using member initialization list to initilize some of the members of array then the reamaining elements gets initilizes with 0, 0.0, \0.

e.g- int arr[5] = {10,20,30};

Date:



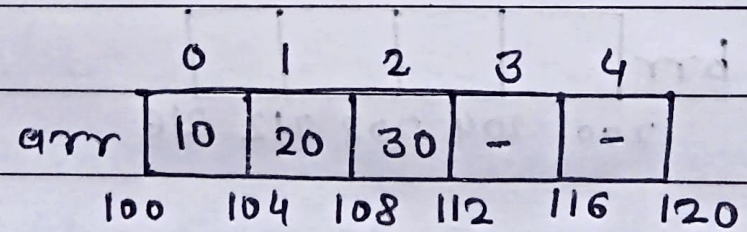
(14) If we initilize the members using member by member initilization technique. If the remaining count is less then the remaining elements gets inilize with it's default values which is depends on the storage class of that array.

eg- int arr[5];

arr[0] = 10;

arr[1] = 20;

arr[2] = 30;



If the above array is defined locally then the last two elements contains garbage in it.

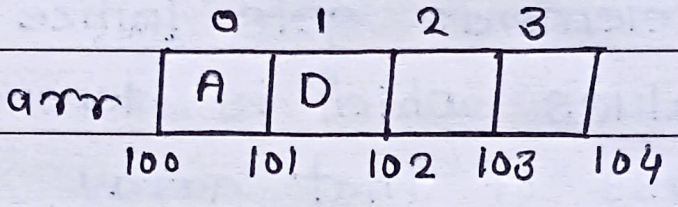
If array is defined globaly or

Date:

statically then the last two elements contains 0, 0.0, \0 in it.

(15) we can create array of any homogeneous data type which may be any primitive data type.

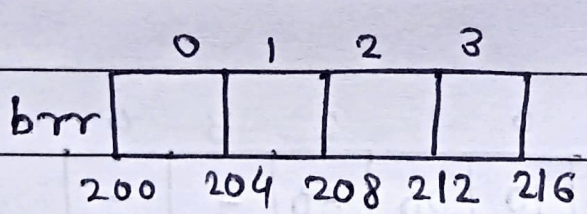
eg – 1) char arr [4];



arr [0] = 'A';

arr [1] = 'D';

2) int brr [4];



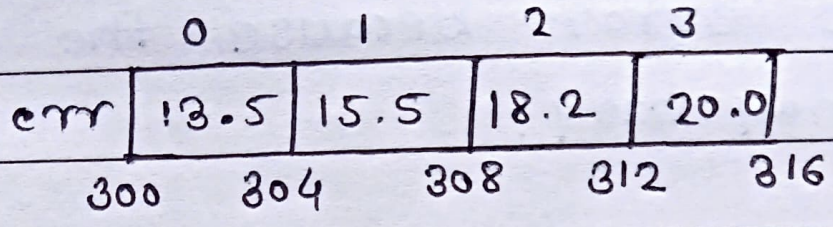
3) float crr [4];

crr [0] = 13.5;

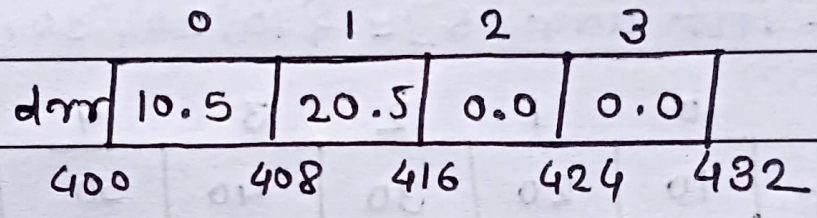
crr [1] = 15.2;

Date:

```
crr[2] = 18.3;
crr[3] = 20.0;
```



4) double drr[4] = {10.5, 20.5};



(16) while creating the array the size of array should be an integral constant.

We cannot use any variable to specify the size of array.

```
int arr[10];
.
.
int i=10;
int brr[i];          // error
```

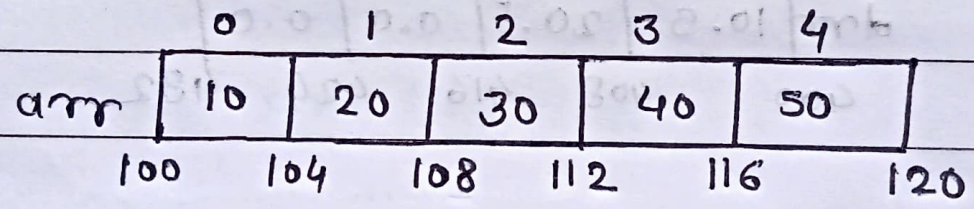
Date:

while creating the array size of array should be a number.

If it is a variable then compiler generates error because the size of array may vary.

⑰ Name of array is internally considered as the base address of first element of array.

eg- int arr[5] = {10, 20, 30, 40, 50};




```
printf("%d", arr);          // 100
printf("%d", arr[0]);       // 10
printf("%d", arr[4]);       // 50
printf("%d", &arr[0]);      // 100
printf("%d", &arr[4]);      // 116
printf("%d", &arr);         // 100
```

Date:

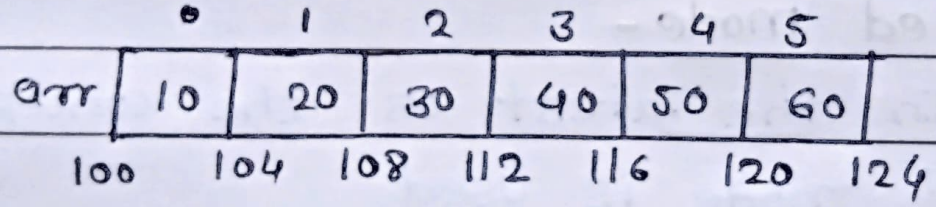
name of array is the address of first element and address of array & arr is internally considered as the base address of whole array.

(18) when we specify the name of array it gives the address of first element

when we use ← operator with the name of array it will give the address of whole array.

e.g -

```
int arr[6] = {10, 20, 30, 40, 50, 60};
```



arr ⇒ 100

&arr ⇒ 100

arr+1 ⇒ 104

(&arr)+1 ⇒ 124

arr+2 ⇒ 108

(&arr)+2 ⇒ 148

Date:

(19) segmentation fault :-

when we run the program it is considered as process for every process os allocates a seperate memory which is called as the address space of process.

when process tries to access the contents which are the outside of the address space of the process then os terminate that process abnormally by segmentation fault.

(20) Protected mode -

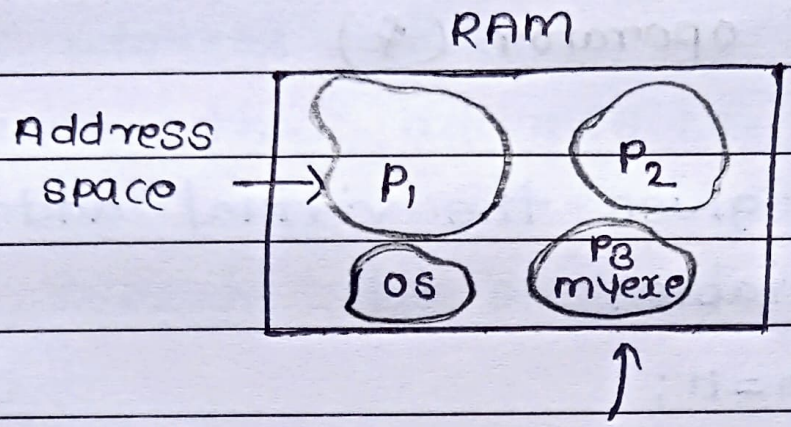
In the latest os the concept of protected mode is used.

In case of protected mode every process (running program) gets specific ammount of memory in RAM.

In case of protected mode every process is isolated from the every another process.

Date:

This isolation provides the protection to every running process.



.c → .i → .obj → .exe

(21) If we initilize some more members a compare to the size of array then it may lead to generate segmentation fault.

In c programming strict array boundary checking is not performed but it gets performed in c++, Java; Due to above point there is no compile time error.

eg-

int arr[5] = {10, 20, 30, 40, 50, 60, 70};

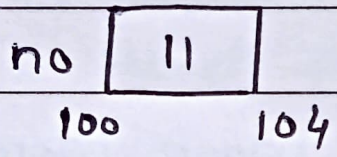
Date:

The above syntax may generate runtime failure (segmentation fault)

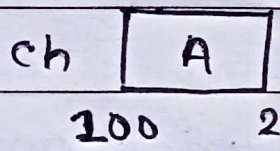
(22) Address of operator (&)

- & operator gives the virtual address of any variable.

eg - 1) int no = 11;



2) char ch = 'A';




```
printf ("%d", no);          // 11
printf ("%d", &no);         // 100
printf ("%c", ch);          // A
printf ("%d", &ch);         // 200
```

(23) virtual Address & physical address -

There are two types of address

Date:

that we can use in case of any type of programming.

When we compile the program the address which is generated by the because that address is not in reality

When we run the program loader will allocate the physical memory inside RAM.

After allocating the physical memory the address is considered as the physical address.

& operator is considered I always gives the virtual address.

It is not possible to fetch the physical address programmaticaly.