10/02/2023

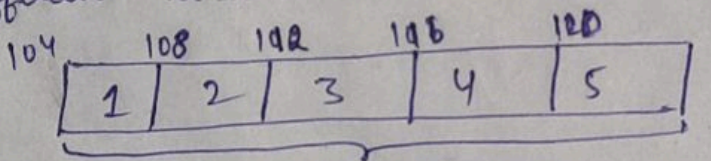
Array in C++

What is Array :→ • It is a Data structure collection of similar kinds of elements

- Arrays are stored in continous memory location.
- ex: ① an int array consists of integer type of value no char can be included in it.
  ② an char type array can only contain char type values in an array

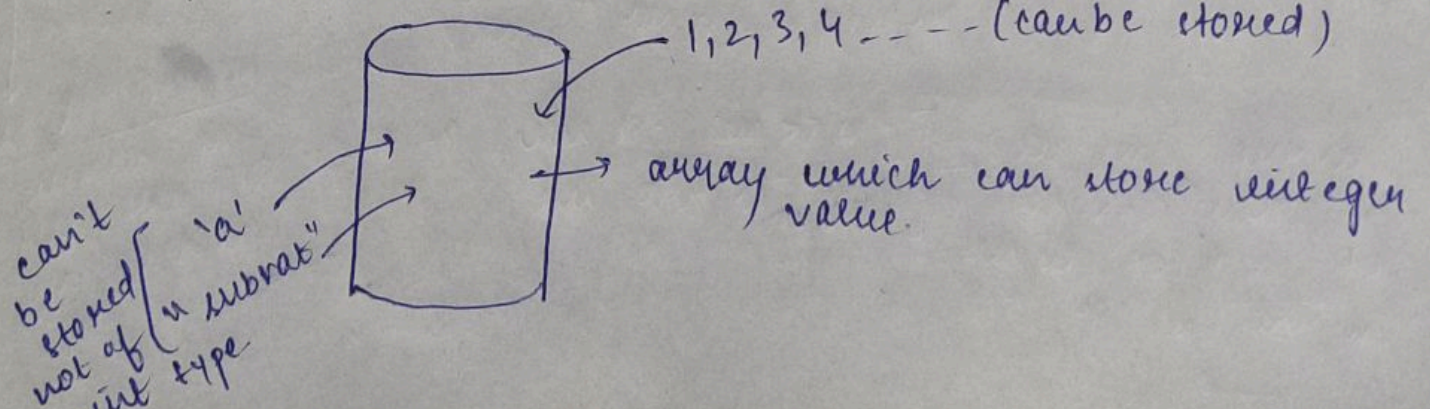
* **Why Array is stored at cnt memory location.**

It is said to be stored at continous memory location because suppose an array consists of 5 integer values
∴ size of an array will be $5 \times 4 = 20$ bytes (4 = datatype of int.)
would require 20 bytes to be stored inside the memory. but it can't be stored in the form of discrete i.e. 3 integers value at different location & 2 integer value at different location.



20 bytes continous memory location.

* We said array is a D.S consists of similar kinds of elements or values.



* Why Array is Needed?

We need array to avoid multiple declaration of variable while taking large amount of inputs.

i.e. if you want to create 2 variables it is possible to create by

```
int a;
int b;
```

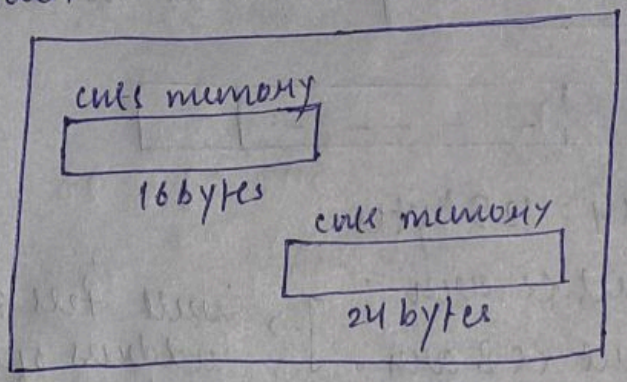
but you want to create or want 100's of inputs it become hectic to create multiple variables. So here arrays plays an important role.

i.e `int arr [300];`

- `int` → datatype
- `arr` → array name
- `[300]` → syntax; value inside it will tell to create continous memory for 300 integer type values on each int 4 byte $4 \times 300 = 1200$ byte

1200 byte acts memory allocation.

* Consider



Now int arr [10] means creating cell memory for integer type array having 10 size

∴ 1 int = 4 byte = $4 \times 10 = 40$ byte

As it can't be created inside memory bcz memory consist of either 16 byte or 24 byte we require 40 byte but $16 + 24 = 40$ byte but array can't be stored at discrete memory location. So this results in memory loss due to unable to store.

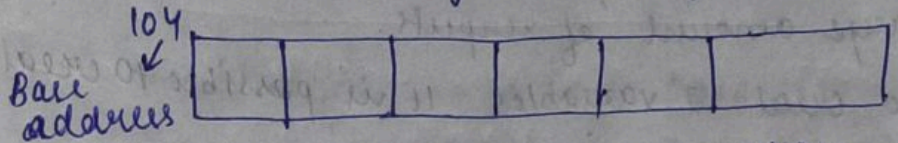
* Creating An Array:

`int arr [10];` → syntax

- `int` → data type
- `arr` → variable name
- `[10]` → size of an array.

① int arr[10]

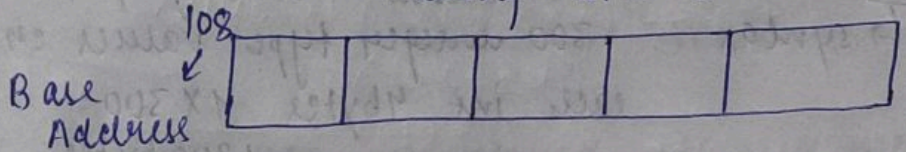
↳ creating array of integer type having size 10.



The initial value of an array is stored at what address said to be base address

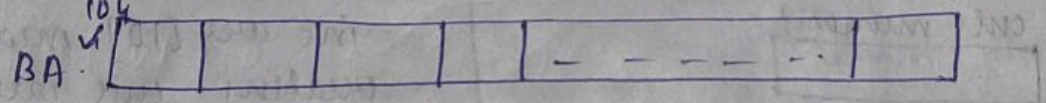
② int arr[6]

↳ creating enough memory for integer type array having size 6.



* How to check at what address array is stored.

To check at what address array has been stored we can write.



int arr[32] → 32 × 4 = 128 bytes

We can write. cout << arr;] → will tell the base address of an array,
or cout << &arr;

* Initialisation of an Array

int arr[] = {_____}

value

[int arr[6] = {1, 2, 3, 4, 5, 6}

value, comma seprator

static array when we tell the size of an array.

* int n; cin >> n; int arr[n];] → dynamic array asking user for input to give to size of an array

* int n; cin>>n; int arr[n]; → consider to be bad practice bcz for some cases / values it might work but may provide large value it might show runtime error bcz of array size limits.

* Array Initialisation:

→ int arr[] = {1,2,3,4,5}

→ int arr[5] = {1,2,3,4,5}

size of array ↓ ↳ total elements are 5

→ int arr[10] = {1,2,3,4,5}

size of array ↓ total elements are 5

what about remaining space.

Remaining space are filled by 0.

```
for(i=0; i<10; i++){
    cout << arr[i];
}
```

output

1 2 3 4 5 0 0 0 0 0

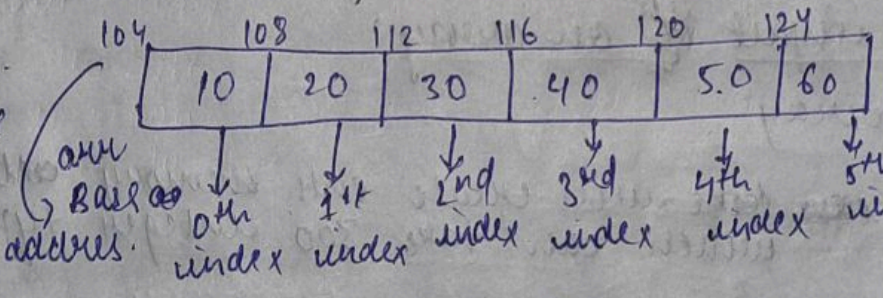
5 elements contain in array ↓ rest space are filled by 0.

→ int arr[4] = {1,2,4,5,6}

//error as size is 4 and providing inputs values more than size.

* Index is Access in Array:

0 based index

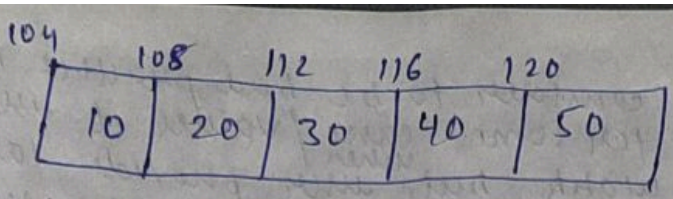


6 size array 0-4 (index)

n size array [0 → (n-1)]

* How to check the element is at what address

value at (arr/B.A + index x datatype size)



→ value at (BA + index * datatypesize)

value at (104 + 0 x 4) → 0th → value at (104 + 3 x 4) → 3rd

value at (104) = 10 " " (104 + 12)

→ value at (104 + 1 x 4) → 1st value at (116) = 40

(104 + 4)

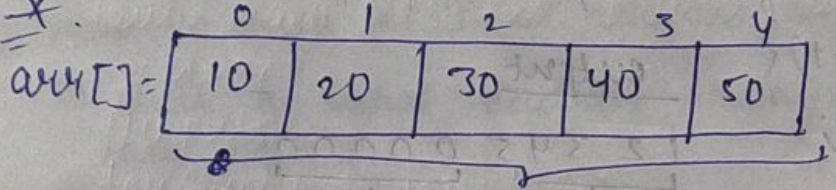
value at (108) = 20

→ value at (104 + 6 x 4) → 6th

(104 + 24)

(128) → garbage value / error] compiler dependent.

* .




```
for (int i=0; i<5; i++){
    cout << arr[i];
}
```

0 < 5 yes
→ arr[0] = 20
1 < 5 yes
→ arr[1] = 30
→ 2 < 5 yes
arr[2] = 40

output

10 20 30 40 50

* How to take input in an array:

create an array

int arr[300];
↳ will create arr memory allocation which can take 300 integer type value

```
for (int i=0; i<10; i++){
    cin >> arr[i];
}
```
// taking inputs

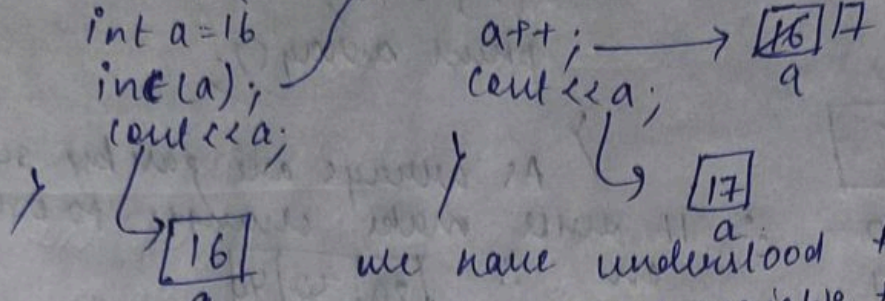
```
for (int i=0; i<10; i++){
    cout << arr[i] << " ";
}
```
// printing array

* Arrays and functions

Consider

```
int main() {
    int a = 16
    inc(a);
    cout << a;
}
```

```
int inc (int a) {
    a++;
    cout << a;
}
```



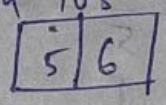
We have understood the concept of pass by value i.e it create the copy of variable to the function without effecting the original value.

Will it will be work same for arrays?

Consider

```
int main() {
    int arr[] = {5,6}
    inc(arr);
    print array()
}
```

```
int inc (int arr[]) {
    arr[0] = arr[0] + 10;
    print the array:
}
```



So in arrays instead of pass by value pass by reference take place i.e means passing the address of original array. The changes made are made inside the actual array since it is known as pass by reference. ∴ Any changes made by the function will change the original array.

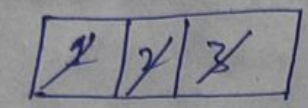
* why we need to pass size along with array to func

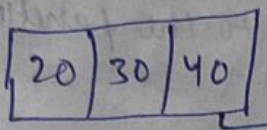
Consider: int arr[30] = {1,2,3,4,5}

↳ it tells to create ints memory for 30 integer type value. but the actual total no. of value an arr contain are 5 ∴ The size of arr contain is 5. So explicitly we need to pass no. of elements the size of any array i.e how many values it contains.

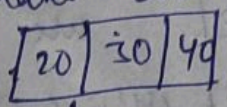
```
int main(){
    int a[] = {1,2,3};
    func(a, size);
    print(a);
}
```

```
func(a[], size){
    a[0] = 20;
    a[1] = 30;
    a[2] = 40;
    print array();
}
```





∴ As arrays are pass by reference it will make changes to original one ∴



will be printed two times

*To create doubles:

```
int arr[30];
int n;
cout<<"enter the no. of inputs an array can take :";
  cin>>n;
//inputs
for(int i=0; i<n; i++){
    cin>>arr[i];
}
//print doubles
for(int i=0; i<n; i++){
    cout<<arr[i]*2<<" ";
}
return 0;
}
```

output

```
5
1
2
3
4
5
```

1 2 3 4 5 → array

① Linear search (Traversal)

```
#include<iostream>
using namespace std;
bool findnum(int arr[],
             int n, int num)
```

steps to linear search

① Initialize the array
② find the no. of element you want in array.
③ inputs of array
④ print array

(5) take input to find which no.
(6) call function to check number found or not.

(2) Count's 0's and 1's in Array

(1) initalize array
(2) Take input how much no. element you want in all array.
(3) take array in input
(4) print array
(5) call function to find no. of zero's and one's

(3) To find the maxno. in an array

(1) initialize array
(2) Take input how much element you want initial array.
(3) take array in input
(4) print array.
(5) call function to print maxNO

Note

To find maxNo. always take a variable = INT_MIN
↓
will take minimum value of int.
maxy → -2^{31} $-2^{-31}-1$
↓
MINVALUE

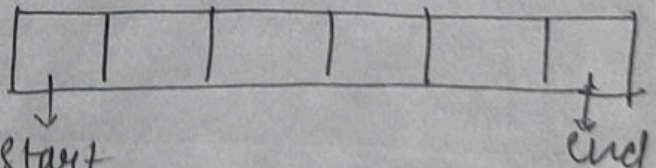
(4) Extreme Print in Array:

(1) initialize array
(2) Take input how much element you want initial array.
(3) take array an ~~array~~ input
(4) print array
(5) call function

```
start = 0;
end = n-1;
if (start == end){
    cout << arr[start];}
else {cout << arr[start];
      cout << arr[end];}
```

2 pointer approach

int start = 0
int end = n-1



start++
end--

(8) reverse a ~~number~~ array

(1) initialize array
(2) Take input how much element you need in an array.
(3) take array an input.
(4) print array
(5) call function to reverse

```
start = 0;
end = n-1;
swap(arr[start], arr[end])
start++;
end--;
```

(6) again print array.