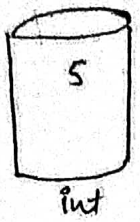
10/Feb/23

Lec-7

Array-1

⊙ What is Array →

- collection of similar type of elements.
- stored in contiguous memory location.
- linear data structure.



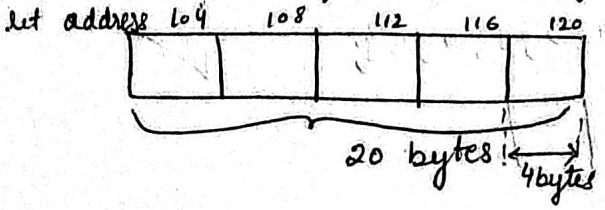
can we store 5 in this? Yes
"Hello" → NO
5.5 → NO
true/false → NO

} because it can only store integers.

⊙ Behind the scene (memory) -

An integer takes 4 bytes of space.

so an array of 5 integers takes a total memory of 5×4 = 20 bytes (contegeous memory).

eg, 

⊙ Why array is needed?

suppose we have to find greatest no. of two numbers. so we will make two variable, compare them and get the result.

Now suppose we want to find greatest element from 30/30000 elements. It's not convienient to create 30000 variables and compare.

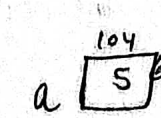
so that's why we use arrays to store element.

eg → int array[30000];

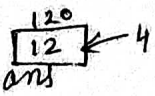
⊙ Creating an Array →

creating variable → int a = 5;

In memory →



int ans = 12;



Symbol table is a data structure maintained by compiler

Symbol Table
a → 104
ans → 120

Declaration of array →

```
int arr[10];
```

- `int` ↑ datatype
- `arr` ↓ name of the array
- `[10]` ← size of the array.

← Contiguous space of 10 integers (40 bytes) is allocated.

• Address of an array-

```
cout << arr ;
cout << &arr;
```

} both gives address of the array.

Questions-

Create an array of

- int → 53 size → `int arr[53];`
- char → 50 size → `char arr[50];`
- bool → 23 size → `bool arr[23];`

→ Initialization-

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

Yes, this size is not mandatory here. ✓

⇒

1	2	3	4	5

```
int brr[] = {1,2,3,4,5};   ✓
int crr[10] = {1,2,3,4,5};   ✓
```

Now the question is what is stored in the rest of 5 spaces? ⇒ Zero (0).

```
int drr[4] = {1, 2,3,4,5};   // Error.
```

Why Error?

Because the size of the array is 4 and we are storing 5 elements in it.

This is STATIC Array. (↳ fixed size.) Can we create an dynamic array?

dynamic array is creation of array at the ~~run~~ runtime. (We will discuss it later).

We have one more way →

```
int n;
cin >> n;
```

int arr[n];

But this is BAD practice and not recommended.

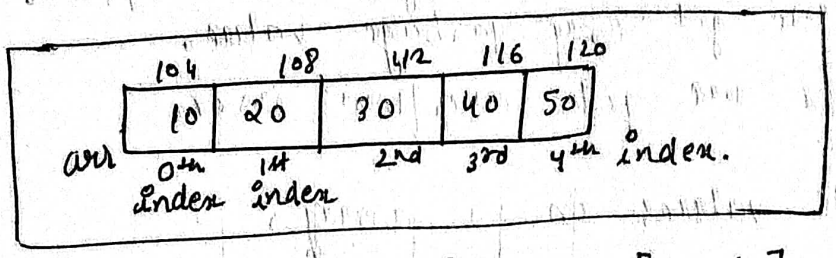
- **Questions-**
  - Create arrays
    - → int → 1, 2, 3, 5

      int arr[4] = {1, 2, 3, 5};
    - → char array → a, b, c, d

      char array[4] = {'a', 'b', 'c', 'd'};

⊙ Index and Access in array →

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



5 size array → index → [0 - 4]

so n size array → index → [0 to (n-1)]

To access first element →

arr[0] → 10

same way we can access other elements with the help of indexes.

arr[1] → 20, arr[2] → 30, arr[3] → 40, arr[4] → 50

What is the significance of 0?

arr[0]

arr + 0×4

(arr + 0)

value at this address

value at (104) address = 10

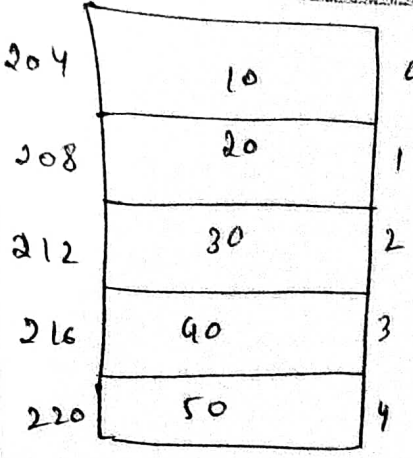
arr[1]

value at (arr + 1×4)

value at (104 + 1×4)

value at (104+4)

value at 108 = 20



arr[0] → value at (arr + index * size of data type)
(base address)

= value at (200+0)
= value at (200) = 10

arr[4] → value at (arr + 4x4)
→ value at 216
= 40

arr[10] → value at (arr + 10x4)
= value at 244.

↳ (but this memory is not allocated for this array.

So in this case two things can happen.

1. we get any garbage value.
2. or we get overflow error.

⊙ Printing all values in a array →

```
int arr[] = {1, 3, 5, 7, 9};
for (int index = 0; index < 5; index++){
    cout << arr[i];
}
```

//o/p → 1 3 5 7 9.

dry run –

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

i = 0, 0 < 5 → T arr[0] print 1
i = 1, 1 < 5 → T arr[1] print 3
i = 2, 2 < 5 → T arr[2] print 5
i = 3, 3 < 5 → T arr[3] print 7
i = 4, 4 < 5 → T arr[4] print 9
i = 5, 5 < 5 → F

⊙ Taking array's element as an input →

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

```
int n;
cin >> n;
arr[i] = n;
```

⊙ Instead of using int arr[n];
we can use int arr[10000];
↑ a big number.
even if we are using/storing only 10 elements in it.

Q1→ Take 5 element i/p in array and print their doubles.

Sol.
```
int arr[5];
for (int i = 0; i<5; i++){          // Taking 5 element in
        cin>>arr[i];                   input.
}
for (int i = 0; i < 5; i++){
        arr[i] *= 2;                // printing their doubles.
        cout << arr[i];
}
```
↳ or simply remove these two lines with a single line → cout << 2*arr[i];

Q2. We have an array →
arr[] = {1, 3, 5, 7, 9};
change it's all elements to 1 like this → {1, 1, 1, 1, 1}

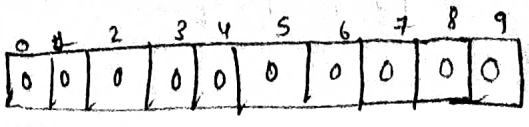
Sol-
```
int arr[] = {1, 3, 5, 7, 9};
for (int i = 0; i<5; i++){
        arr[i] = 1;
}
```

H.W → Find out what doest memset func do and how to use it.

⊙
```
int arr[10] = {1, 2};
for (int i = 0; i<10; i++){
        cout << arr[i];
}
```
o/p → 1 2 0 0 0 0 0 0 0 0 remaining elements are 0's.

⊙
```
Int arr[10];
for (int i = 0; i<10; i++){
        cout << arr[i];
}
```
o/p → all 10 garbage values.

⊙ To initialize all elements with 0:→
int arr[10] = {0};



⊙ How to initialize all elements with 1?

1 ⇒
```
int arr[10];
for (int i=0; i<10; i++){
    arr[i] = 1;
}
```

This will not work
`arr[10] = {1};`

1	0	0	0	0	0	0	0	0	0

2 ⇒ memset, Explore it.

⊙ Arrays & functions →

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

a [16] 104

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

pass by value. ← copy

a [16 | 17] 412

17 is printed.

16 is printed.

In array →

```
main(){
    int arr = {5, 6};
    inc(arr);
    printarray(arr, 2);
}
```

104 | 108

5	6

```
inc(int brr[]){
    brr[0] = brr[0] + 10;
    printarray(brr, 2);
}
```

Name can be different. but the location is same.

15	6

suppose printarray is a func which prints all element of the array.

10	6

so this is pass by reference.

Whenever an array is passed in a function it is always passed by reference.

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

1	2	3

0 1 2

20	30	40

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

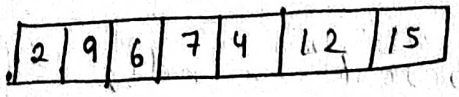
20	30	40

- **sizeof** → gives the size of an array.
  But what happens when only some elements are there in array (not all).

```
int arr[10] = { 1, 2 };
```

in this case sizeof operator does not give us the value filled with element but it gives the size allocated. (In this case 40).

⑥ Linear Search in Array:



→ 6 is present in this array or not?
⇒ Yes, it's present.

How do we know?
⇒ Go to each element and compare.
arr[i] == 6 ?

Let's code this -

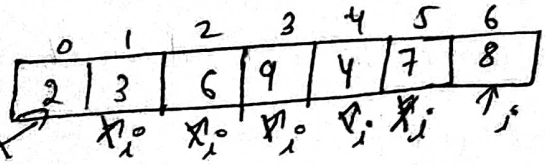
```
main(){
    int arr[5] = {1,3,5,7,8};
    int size = 5;
    cout << "Enter the key to find";
    int key;
    cin >> key;
    if (linearSearch (arr, size, key)){
        cout << "Found" << endl;
    }
    else{
        cout << "Not Found" << endl;
    }
}
bool linearSearch (int arr[], int size, int key){
    for (int i = 0; i < size ; i++){
        if (arr[i] == key)
            return true;
    }
    return false;
}
```

• Alternate way - (w/o function)

```
main(){
    int arr[] = {1, 2, 3, 4, 5};
    int size = 5;
    int key = 3;
    bool flag = 0;
    for (int i = 0; i < size; i++){
        if (arr[i] == key){
            flag = 1;
            break;
        }
    }
    if (flag)
        cout << "Present" << endl;
    else
        cout << "Not present" << endl;
}
```

flag → 1 (found the element)

flag → 0 (not found)



i = 0
arr[0] == key
2 == 8 → F
i++

i = 1
arr[1] == key → F
i++

i = 2
arr[2] == key → F
i++

i = 3
arr[3] == key → F
i++

i = 4
arr[4] == key → F
i++

i = 5
arr[5] == key → F
i++

i = 6
arr[6] == key → T

found.

⊙ Count 0's and 1's in an array.

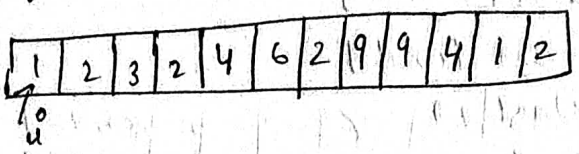
```
int arr[] = {0,1,1,1,1,0,0,0,1,1,0,1,0};
int size = 13;
int numZero = 0;
int numOne = 0;
for(int i = 0; i < size; i++){
```

```
        if (arr[i] == 0)
            numZero++;
        if (arr[i] == 1)
            numOne++;
    }
}
```

We could have used else here but in case of array contains other than 0 and 1 than using else is a problem.

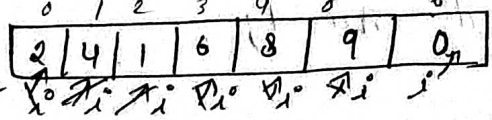
```
cout << "Number of Zeroes: " << numZero << endl;
cout << "Number of Ones: " << numOne << endl;
```

⑥ Count no. of 2's →




```
int cnt = 0;
for (int i=0; i< size ; i++){
    if (arr[i] == 2)
        cnt++;
}
cout << "No of 2's : " << cnt << endl;
```

⑥ Maximum no. in an array →



Best practice (when you have to find max no., use INT_MIN to initialize and when you have to find min, use INT_MAX as initialization.)

```
int maxNum = INT_MIN;
```

Include a header file, limits.h

```
int size = 7;
int maxi = INT_MIN;
for (int i=0; i< size; i++){
    if (arr[i] > maxi)
        maxi = arr[i];
}
cout << "Maximum no. is " << maxi << endl;
```

⑥ Minimum no. in an array →

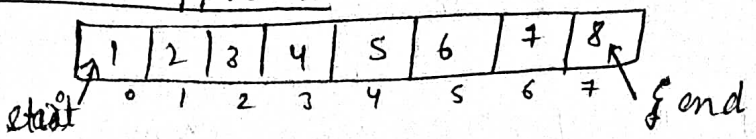
```
int arr[] = {2,4,6,1,3,7,9,11,56,43,21};
int size = 11;
int mini = INT_MAX;
for (int i=0; i< size; i++){
    if (arr[i] < mini)
        mini = arr[i];
} cout << "Minimum no. is: " << mini << endl;
```

Extreme print in Array -

i/p → {1, 2, 3, 4, 5, 6, 7, 8};

o/p → 1, 8, 2, 7, 3, 6, 4, 5

Two pointer's approach -



start → starting index → 0

end → last index → (size-1)

loop.
- print arr[start]
- print arr[end]
- start++ ← increment start
- end-- ← decrement end.

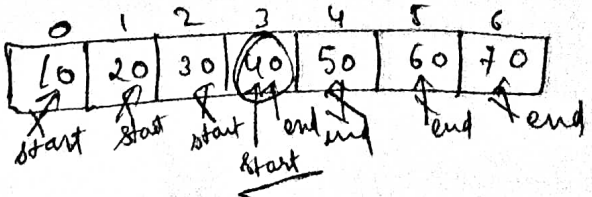
Rukna kab hai?

↳ jaise hi start end se aage nikal jayega.

```
int arr[] = {10, 20, 30, 40, 50, 60, 70, 80};
int size = 8;
int start = 0, end = size-1;
while (true){
    if (start >= end)
        break;
    cout << arr[start] << " ";
    cout << arr[end] << " ";
    start++;
    end--;
}
```

o/p → 10 80 20 70 30 60 40 50

In case of odd element one element will be printed 2 times.



o/p → 10 20 60 30 50 40 40

because start and end pointing at the same element.

~~so with~~ so we will first add an simple condition.

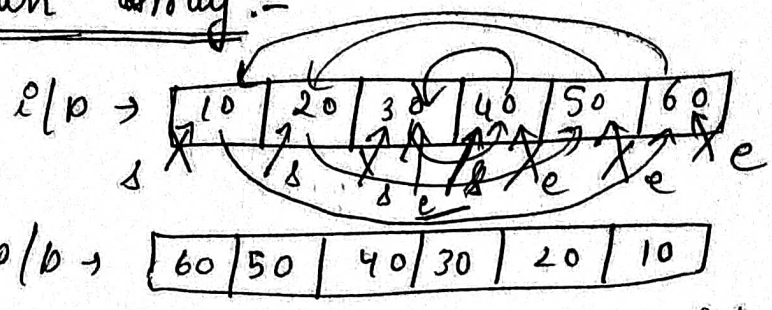
```
if (start == end)
    cout << arr[start] << " ";
```

→ or here we can also write arr[end]. (either of them).

Now while condition

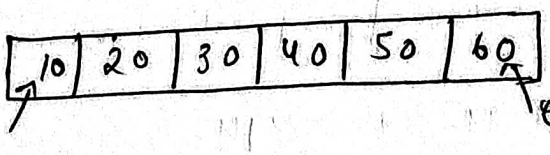
↳ while (start <= end)

⑥ Reverse an Array :-



⇒ we only swapped extreme points.

we have a predefined function to swap.



s = 0
e = n-1

Steps →

loop → ① swap(arr[s], arr[e])
② s++;
③ e--;

let's code this →

```
int arr[] = {10, 20, 30, 40, 50, 60};
int size = 6;
int start = 0, end = size - 1;
while (start <= end){
    swap(arr[start], arr[end]);
    start++;
    end--;
}
```