ARRAYS-LEVEL 1

07 September 2023 09:44

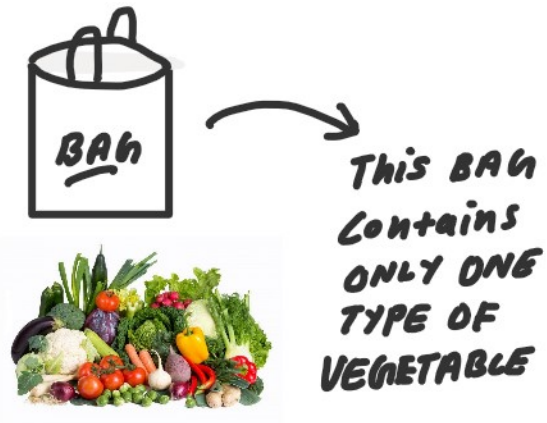
CLASS 07 — 6/9/2023

https://www.linkedin.com/in/manojoffcialmj/

ARRAY LEVEL-1

What is array :

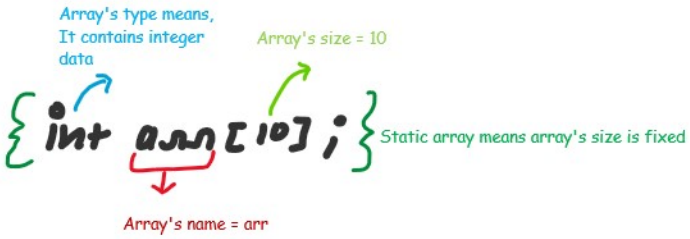
- List of similar items
- Collection of elements
- Contains data in continuous memory location
- It is a data structure
- It is also a single variable

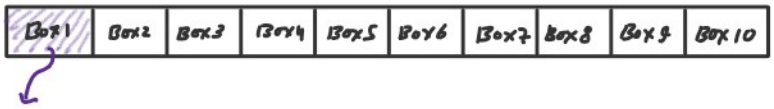


Why do we need of array :

- It helps to store large number of same data in single variable to avoid confusion that can occur when using several variables.
- If we have to store data in order list

Create an array :





Base or array name address

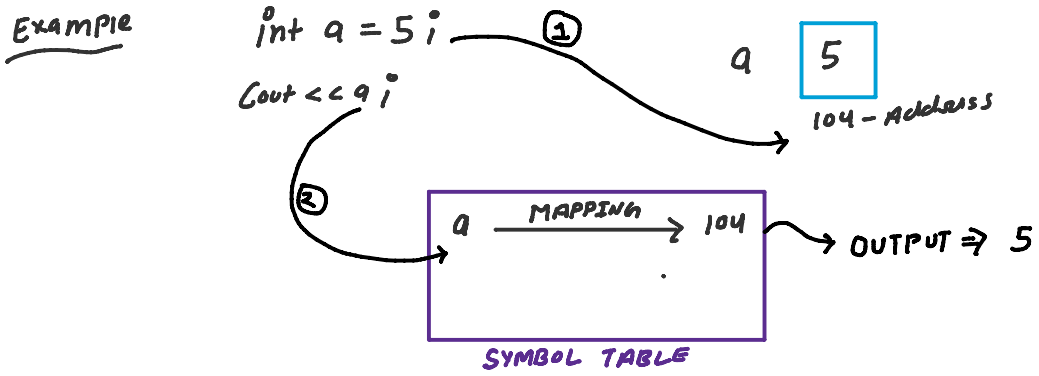
each box size = data type's size

Box1 = 4 bytes
Box10 = 4 bytes
} 4 x 10 => 40 bytes

Total size of array = data type's size * total box

Symbol table :

Example int a = 5; — (1)



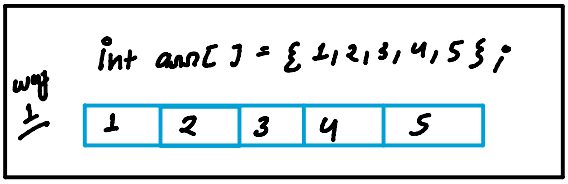
Address and sizeof operator :

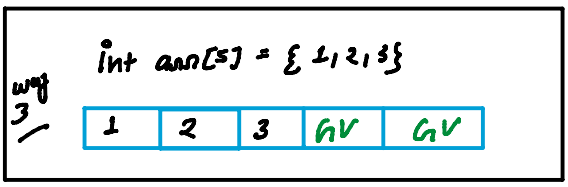
- Address operator can be used to find the address
- Size of operator can be used to find the size of a variable

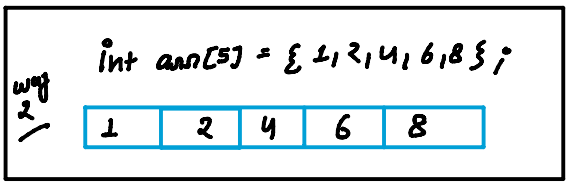
```cpp
#include<iostream>
using namespace std;

int main(){
    int a=5;
    int arr[10];

    cout<<"Address of a = "<< &a<<endl;
    cout<<"Size of a = "<< sizeof(a) <<" bytes"<<endl;

    cout<<"Address of arr = "<< &arr<<endl;
    cout<<"Size of arr = "<< sizeof(arr) <<" bytes"<<endl;

    return 0;
}

/*
OUTPUT:
Address of a = 0x7ffd434e624c
Size of a = 4 bytes
Address of arr = 0x7ffd434e6250
Size of arr = 40 bytes
*/
```

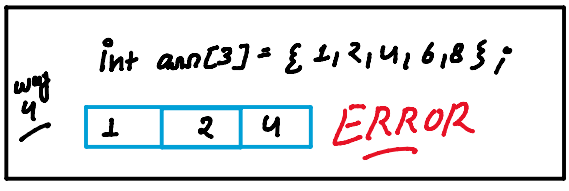
@manojofficialmj

Initialization of an array :



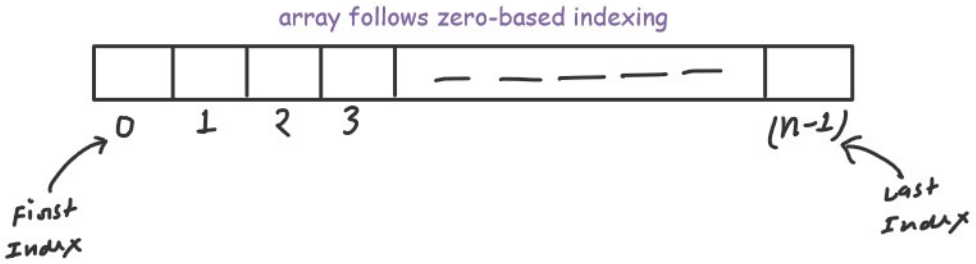


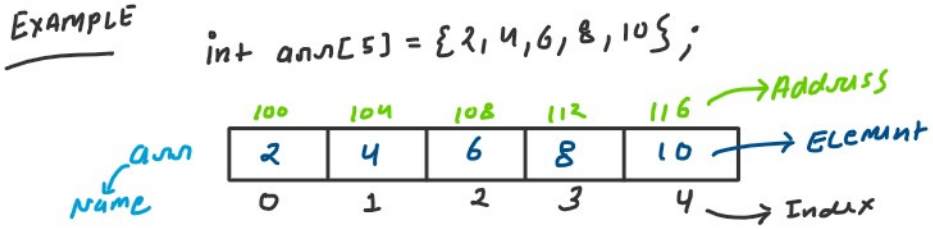




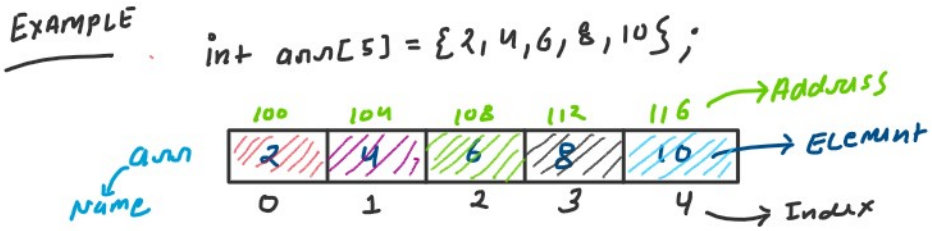
Bad practice with array size :

Array indexing :





Access array's elements :



- access array's elements individually	- iterate array's elements using loop

arr[0] = 2
arr[1] = 4
arr[2] = 6
arr[3] = 8
arr[4] = 10
	for(int i=0; i<5 ; i++){
	cout << arr[i] << endl;
	}

Taking input in an array :

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

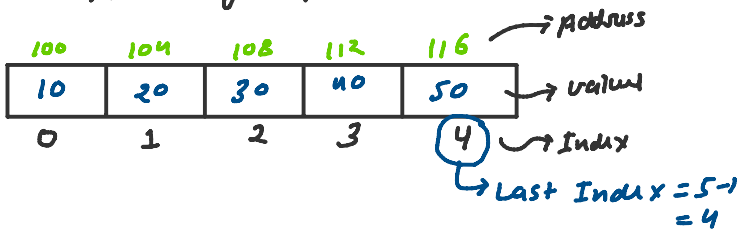
```
cin>>arr[0]
cin>>arr[1]
cin>>arr[2]
cin>>arr[3]
cin>>arr[4]
```

5
10
15
20
25

Meaning of array[i] (FORMULA) :

EXAMPLE

```
int array[5];
```



FORMULA

array[i] --> value at {Base Address + (Data Type Size * Index)}

10 at ↳ 100 + (4 * 0) = 100

20 at ↳ 100 + (4 * 1) = 104

30 at ↳ 100 + (4 * 2) = 108

40 at ↳ 100 + (4 * 3) = 112

50 at ↳ 100 + (4 * 4) = 116

Updating array with example :

Problem Statement 01:

1. 10 size of array
2. Take input in that array
3. Double-up each value of that array

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

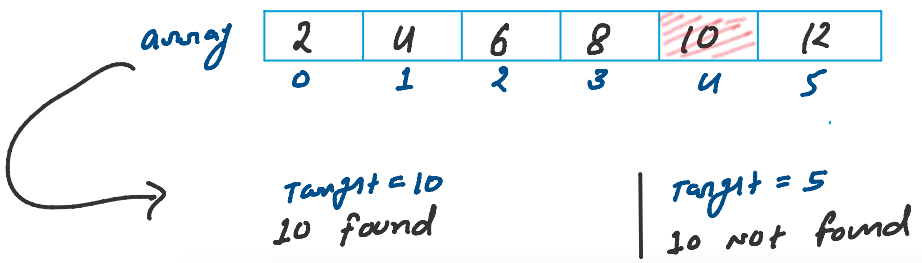
```
int n;
for(int i=0; i<n; i++){
    cin>> arr[i];
}
```

```
for(int i=0; i<n; i++){
    arr[i] = 2*arr[i];
}
```

```cpp
#include<iostream>
using namespace std;

int main(){
    int n=10;
    int arr[10];

    // Taking input from user
    for(int i = 0; i < n; i++){
        cin>>arr[i];
    }

    // Double-up each value of that array
    for (int i = 0; i < n; i++)
    {
        // Updating array's element
        arr[i]=2*arr[i];
    }

    // Printing array
    for (int i = 0; i < n; i++)
    {
        cout<<arr[i]<<" value at index "<<i<<endl;
    }
    return 0;

}

/*
INPUT: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
OUTPUT: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
*/
```

@manojofficialmj

Linear search in an array :



```cpp
#include<iostream>
using namespace std;

int main(){
    int n=6;
    int target=10;
    int arr[]={2,4,6,8,10,12};

    bool flag=false;

    for(int i=0; i<n; i++){
        if(target==arr[i]){
            flag=true;
            break;
        }
    }

    // Apply condition to print target is found or not
    if(flag==true){
        cout<<target<<" found";
    }else{
        cout<<target<<" not found";
    }
    return 0;
}

/*
OUTPUT:
10 found
*/
```

@manojofficialmj

Array and function :

```
int main() {
    ① int arr[100];
    ② int size = 5;
    ③ PrintArray(arr, size);
}
```

```
PrintArray(int arr[], int size){
    for(int i=0; i<size; i++){
        cout << arr[i] << " ";
    }
}
```

1. Array can store elements up to 100.
2. We want to store only 5 elements.
3. When we want to pass array in function so we should also pass the array's size in it.

PROGRAM1

Count 0's an 1's in an array

arr

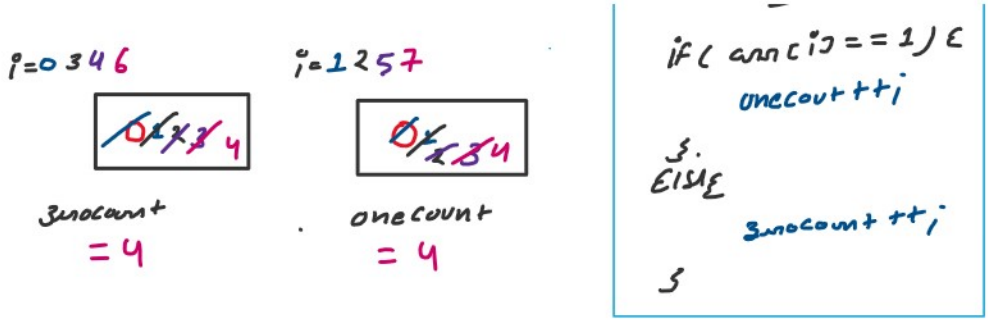
0	1	2	3	4	5	6	7
0	1	0	1	0	1	0	1

i=0 3 4 6

i=1 2 5 7

Logic

```
if(arr[i]==1){
    mecount++;
```

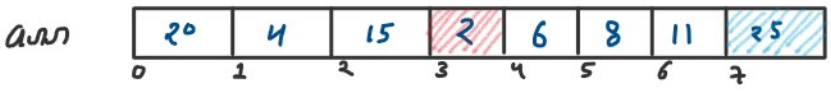



```cpp
// Count 0's and 1's in an array
#include<iostream>
using namespace std;

// Counting zero's and one's
void countZeroOne(int arr[],int size){
    int zeroCount=0;
    int oneCount=0;

    for(int i=0;i<size;i++){
        if(arr[i]==0){
            zeroCount++;
        }
        else{
            oneCount++;
        }
    }
    cout<<"Zero Counts: "<<zeroCount<<endl;
    cout<<"One Counts: "<<oneCount<<endl;
}

// Main method
int main(){
    int arr[100];
    int size;

    // Entering size of array
    cout<<"Enter size: ";
    cin>>size;

    // Taking input in array
    cout<<"Enter element: ";
    for(int i=0;i<size;i++){
        cin>>arr[i];
    }

    // Calling method
    countZeroOne(arr,size);
    return 0;
}

/*
OUTPUT:
Enter size: 8
Enter element: 0 1 0 1 0 1 0 1
Zero Counts: 4
One Counts: 4
*/
```

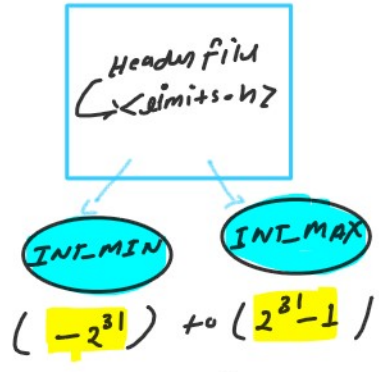
@manojofficialmj

PROGRAM2

Minimum and maximum number in an array



size = 8



minimum

① int minVal = INT_MAX;

② for(int i=0; i<size; i++){
  if(minval > arr[i]){
    minval = arr[i];
  }
}

o/p ②

MAXIMUM

① int maxVal = INT_MIN;

② for(int i=0; i<size; i++){
  if(maxval < arr[i]){
    maxval = arr[i];
  }
}

o/p 25

a = 5 , b = 10

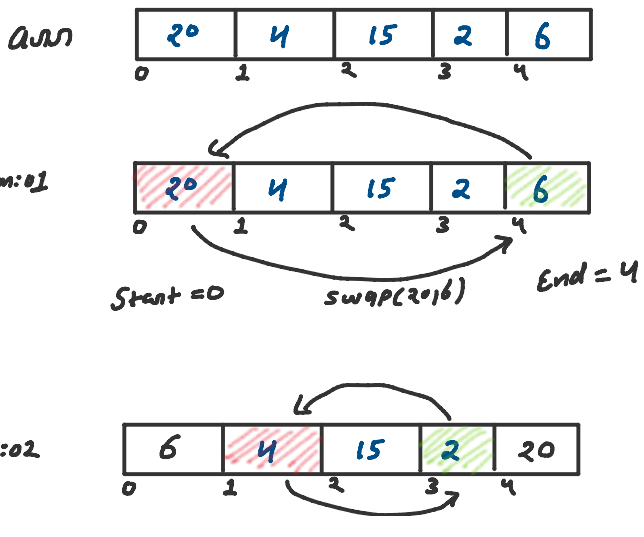
min(a,b) → minimum value 5
max(a,b) → maximum value 10

```cpp
// Minimum in an array
void findMinimumInArray(int arr[],int size){
    int minVal=INT_MAX;

    for(int i=0;i<size;i++){
        if(minVal>arr[i]){
            minVal=arr[i];
        }
    }
    cout<<"Minimum Element: "<<minVal<<endl; // 2
}
                                    @manojofficialmj
```

```cpp
// Maximum in an array
void findMaximumInArray(int arr[],int size){
    int maxVal=INT_MIN;

    for(int i=0;i<size;i++){
        if(maxVal<arr[i]){
            maxVal=arr[i];
        }
    }
    cout<<"Maximum Element: "<<maxVal<<endl; // 25
}
                                    @manojofficialmj
```

PROGRAM3

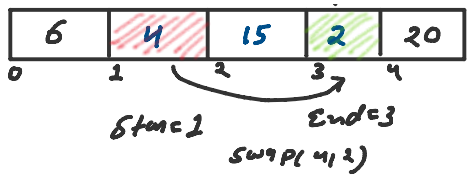
Reverse an array (Two Pointers Approach)



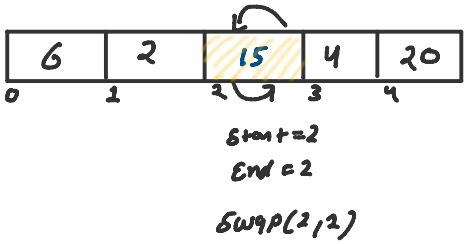
start <= End → Terminating condition

swap(arr[start], arr[end]);
start ++
End --

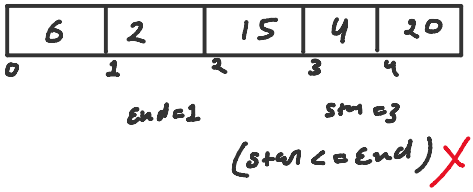
Iteration:02



Iteration:03



Iteration:04



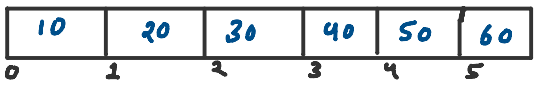

```
// Reverse an array (Two pointer approach)
void reverseArray(int arr[],int size){
    int start=0;
    int end=size-1;

    while(start<=end){
        swap(arr[start],arr[end]);
        start++;
        end--;
    }
}
                                        @manojofficialmj
```

PROGRAM

Extreme print an array (Two Pointers Approach)

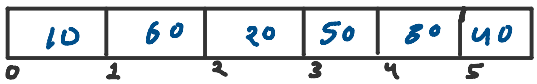
Even Last

arr



size = 6

O/P



(start < End) → may be

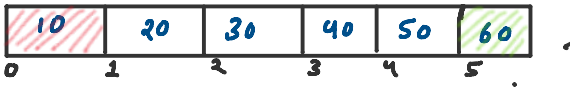
① Start=0, End=size-1

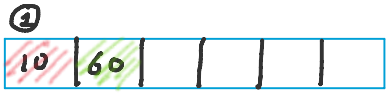
② print arr[start]

③ Print arr[End]

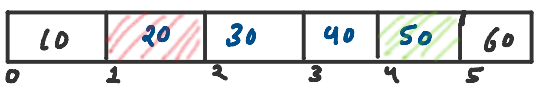
④ Start++
End--

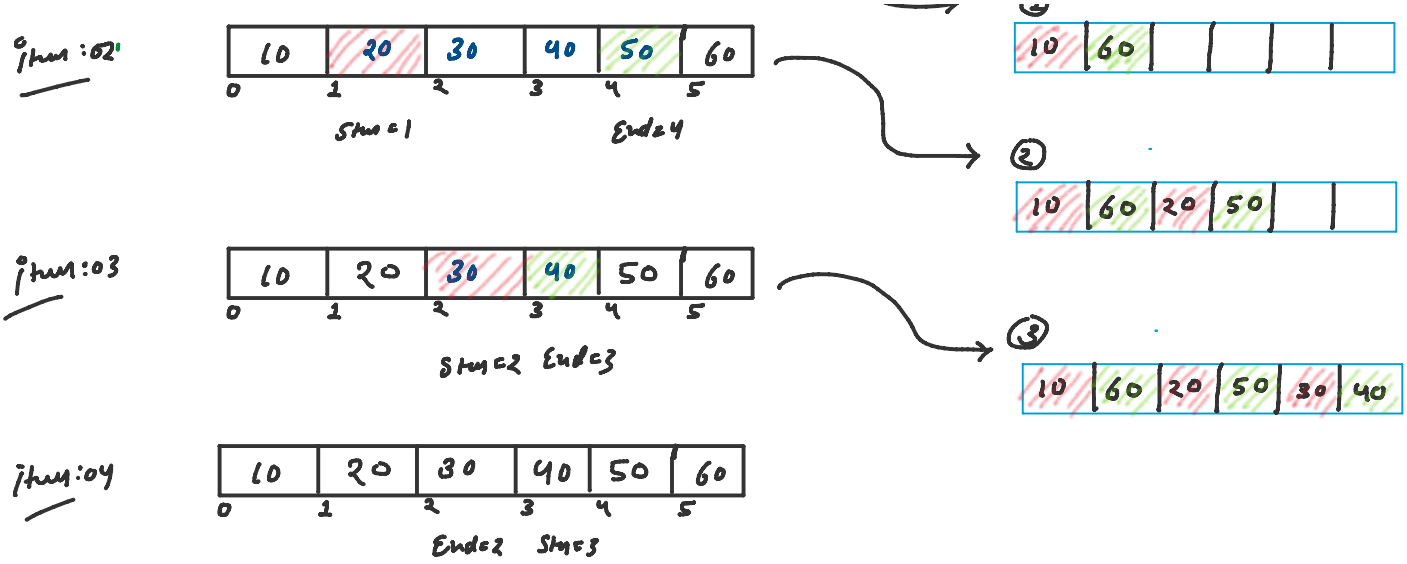
Iteration:01





Iteration:02

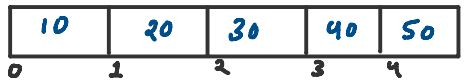




(Start < End) ✗ End

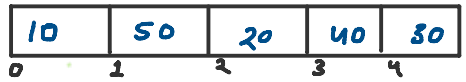
ODD CASE

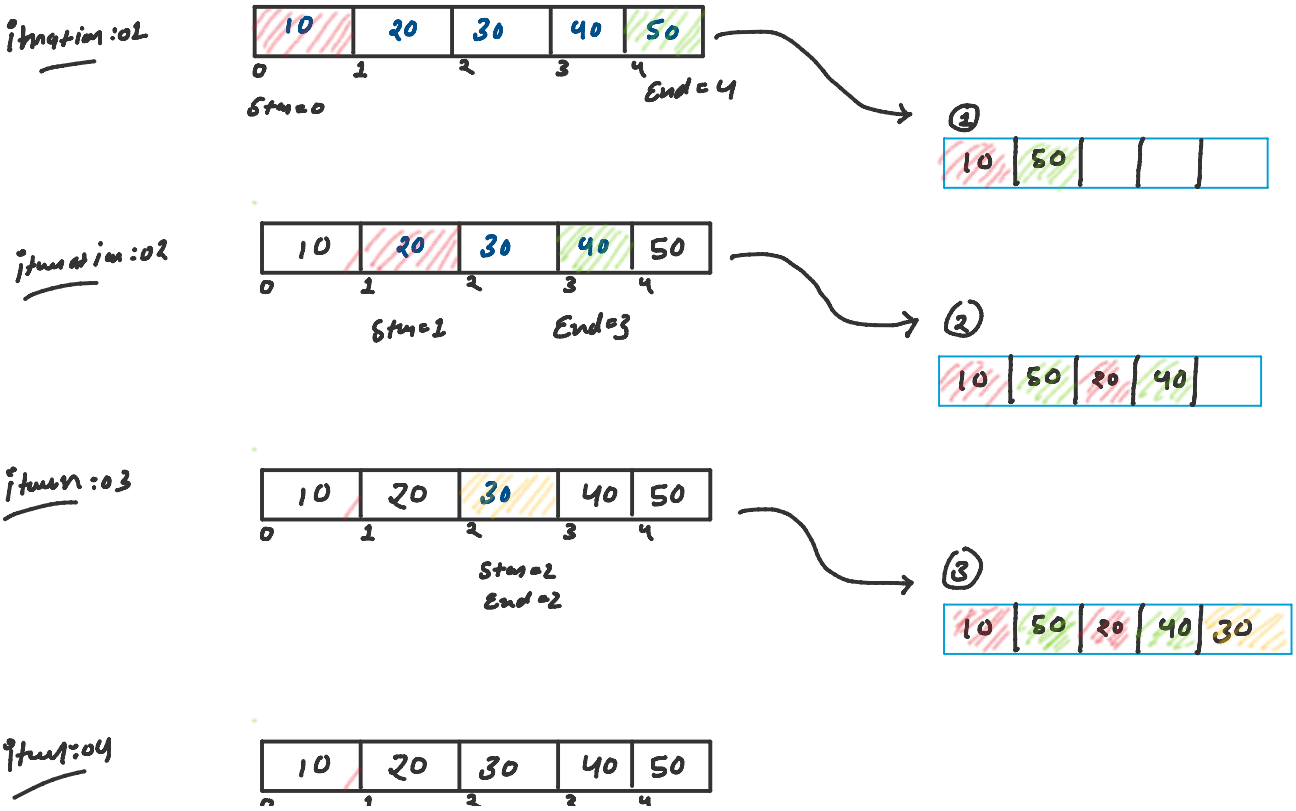
arr



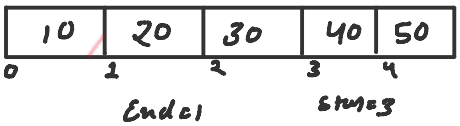
Size = 5

output





9th1:04



End=1 Start=3

(Start <= End) X END

```cpp
// Extreme print an array (Two pointer approach)
void extremeArray(int arr[],int size){
    int start=0;
    int end=size-1;

    while(start<=end){
        if(start==end){
            cout<<arr[start]<<" ";
        }
        else{
            cout<<arr[start]<<" ";
            cout<<arr[end]<<" ";
        }

        start++;
        end--;
    }
}
```

@manojofficialmj