# Arrays

(class 2)

Date 11-02-2023

* **Vector** :- vector are same as dynamic array with the ability to resize itself automatica-lly when an element is inserted or deleted.
vector elements are placed in contigous storage so that they can be accessed & traversed using iterators.
In vectors, data is inserted at the end.
Inserting at the end takes different-iable time, as sometimes the array may need to be extended.
Data is removed only from the end, & removing the last element takes only constant time because no resizing happens.
Storage is managed automatically so that on an attempt to insert an element into a full vector, a larger memory block is allocated for the vector, the vector elements are copied to the new block, & the old block is released.

$\rightarrow$ Name of vector

```
vector<int> arr;
```

$\downarrow$ Keyword    $\downarrow$ Datatype

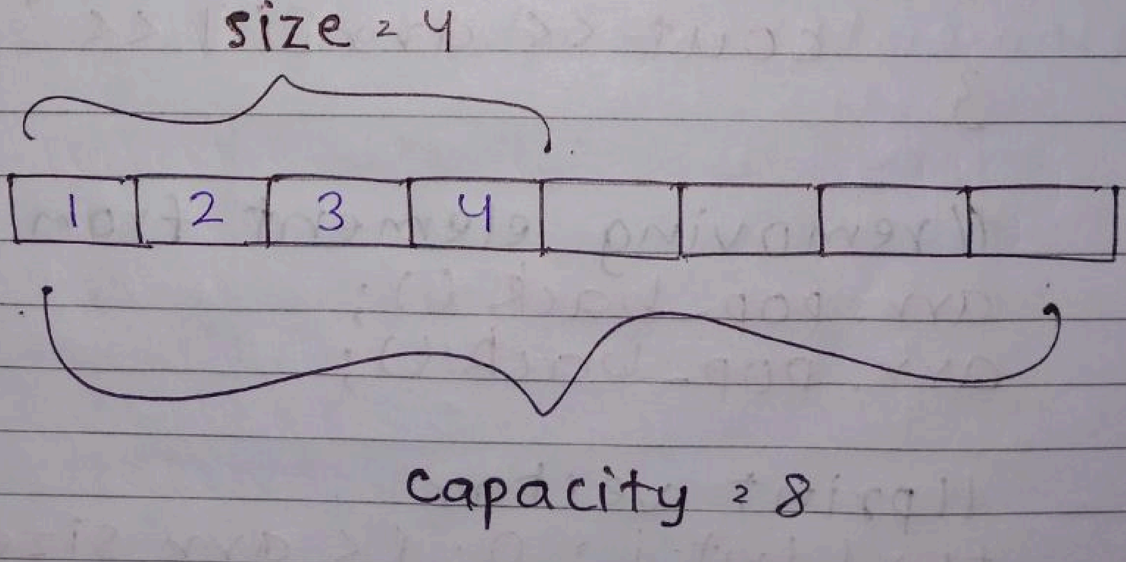
Date..................

## * vector initialization :-

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
   vector<int> arr;
   cout << arr.size() << endl;
   cout << arr.capacity();
   return 0;
}
```

size function return the no. of elements that are currently present in vector.

capacity function return the available space for elements in vector



* Insertion & Deletion to a vector :-

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

int main(){
    //declare vector
    vector<int> arr;

    //inserting element in vector
    arr.push_back(30);
    arr.push_back(40);
    arr.push_back(45);
    arr.push_back(59);

    //print vector
    for(int i= 0; i< arr.size();
                            i++)
    {
        cout << arr[i] << " ";
    }

    //removing element from vector
    arr.pop_back();
    arr.pop_back();

    //print vector
    for(int i = 0; i< arr.size();
                            i++)
    {
      cout<< arr[i] << " ";
    }
}
```

Date..................

* Explicitly allocate size to a vector :-

```
vector<int> arr(10);
cout << arr.size();
cout << arr.capacity();
```

↓

if we intially declare the size of vector, then all the values in the vector is by default 0.

Spiral

Teacher's Sign ...............

Date..................

Ques. Find unique element.

we've given an array of integers of size n, the task is to find the first non-repeating element in the arrays.

Example :-

{-1, 2, -1, 3, 0}

↓

the first number that does not repeat in the whole array is 2.

Approach: By using XOR operator.
XOR returns 0 when both elements are same & returns 1 when elements are different.

| a | b | c = a ^ b |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Example:- If we do XOR operation on elements in the array.

{1, 2, 4, 2, 1, 3, 6, 5, 5, 6, 4}

$\cancel{1} \wedge \cancel{1} = 0$
$\cancel{2} \wedge \cancel{2} = 0$
$\cancel{2} \wedge \cancel{2} \wedge 3 = 3$
$\cancel{2} \wedge \cancel{2} \wedge 3 \wedge \cancel{1} \wedge \cancel{1} = 3$

Like this, if we do XOR operation on all the elements in the array, then we find the answer.

$\cancel{1} \wedge \cancel{2} \wedge \cancel{4} \wedge \cancel{2} \wedge \cancel{1} \wedge 3 \wedge \cancel{6} \wedge \cancel{5} \wedge \cancel{5} \wedge \cancel{6} \wedge \cancel{4}$

↘ 3 is the output.

Code :-

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

void findUnique(vector<int> arr)
{
    //ans variable & initialize it with
       0 to perform XOR operation
    int ans = 0;
//perform XOR operation of ans
   variable & array elements
for(int i=0; i<arr.size();
                    i++)
{
    ans = ans ^ arr[i];
}

cout << "\n unique element
   in given array is " << ans;
}

int main()
{
  //taking size of vector
   int n;
  cout << "Enter the size of
     the vector: ";
  cin >> n;

  //declaring the vector of n size
  vector<int> arr(n);

  //taking input
  cout << "\n enter elements: ";
  for(int i=0; i< arr.size(); i++)
  {
     cin >> arr[i];
  }

  findUnique(arr);
}
```

Spiral                                   Teacher's Sign

Ques Union of two arrays.

$$\begin{tikzcd}[column sep=small]
\{1,3,5,7,9\} \arrow[dr] & \cup & \{2,4,6,8\} \arrow[dl] \\
& \{1,3,5,7,9,2,4,6,8\} &
\end{tikzcd}$$

Approach: we've given two arrays & we have to do union of these arrays.
Firstly, we create a new vector & copy the elements of both array in vector then print the vector.

code :-

```
int main()
{
    // array 1
    int arr1[] = {1, 3, 5, 7, 9};
    int n1 = 5;

    // array 2
    int arr2[] = {2, 4, 6, 8};
    int n2 = 4;

    vector<int> ans;
```

Date.................

```
//copying arr1 elements to ans
for(int i=0; i< n1; i++)
{
    ans.push_back(arr1[i]);
}

//copying arr2 elements to ans
for(int i=0; i< n2 ; i++)
{
    ans.push_back(arr2[i]);
}

// printing ans vector
for(int i = 0; i< ans.size(); i++)
{
    cout << ans[i] << " ";
}
return 0;
}
```

Spiral                                                Teacher's Sign ..............

Date...............

Ques Intersection of two arrays?

{1, 2, 3, 4, 6, 8} ∩ {3, 4, 9, 10}

↘ ↙

{3, 4}

Approach: we check each element of arr1 with each element of arr2, & if both elements are maching then we insert the element in ans vector. (using nested loops) At the end, print the ans vector.

Code:-

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
      int arr1[] = {1, 2, 3, 4, 6, 8};
      int n1 = 6;
      int arr2[] = {3, 4, 9, 10};
      int n2 = 4;

      vector<int> ans;
```

Date.................

```
for(int i=0; i<n1; i++)
{
  for(int j=0; j<n2; j++)
  {
    if(arr1[i] == arr2[j])
    {
      ans.push_back(arr1[i]);
    }
  }
}

cout << "Intersection are:";
for(int i=0; i< ans.size(); i++)
{
  cout << ans[i] << " ";
}

return 0;
}
```

If there are duplicate elements in any array then also Intersection will show duplicate elements.
To avoid this, we mark the element by any random value so that it will not compare at next itera-tion.

Date..................

```
for(int i = 0; i < n1; i++)
{
  for(int j = 0; j < n2; j++)
  {
     if(arr1[i] == arr2[j])
     {
         arr2[j] = INT_MIN;
         ans.push_back(arr1[i]);
     }
  }
}
```

Date..................

Ques. Pair sum.

```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
    vector<int> arr{10, 20, 30, 40};
    int sum = 60;
    //outer loop for picking up 1
        element at a time.
    for(int i = 0; i < arr.size(); i++)
    {
        // inner loop for making first
            element pair with rest of the
            right side elements.
        for(int j = i+1; j < arr.size();
                                   j++)
        {
            if(arr[i] + arr[j] == sum)
            {
                cout << "Pair : (" << arr[i]
                << ", " << arr[j] << ")" <<
                 " makes " << sum;
            }
        }
    }
    return 0;
}
```

Date.................

* logic to find & print every pair in an array.

```
int k = 1;
for(int i = 0; i < arr.size(); i++)
{
   for(int j = i+1; j < arr.size(); j++)
   {
      cout << "Pair " << k << ":( " <<
        arr[i] << ", " << arr[j] << ")";
      k++;
   }
}
```

↓

if we have to find & print every pair in array then the logic is above, using 2 loops

↓

For finding Triplet, we use 3 loops.

Date..................

Ques Triplet sum.

```
#include <iostream>
#include <vector>
using namespace std;
int main(){
    vector<int> arr{10,20,30,40};
    int sum = 60;
    for(int i=0; i<arr.size(); i++)
    {
        for(int j=i+1; j < arr.size();
                                j++)
        {
            for(int k = j+1 ; k < arr.
                        size(); k++)
            {
                if(arr[i] + arr[j] +
                        arr[k] == sum)
                {
                    cout << "Pair:(" <<
                    arr[i] << arr[j] <<
                    arr[k] << ")" <<
                    "makes" << sum;
                }
            }
        }
    }
    return 0;
}
```

Date..................

Ques. Find four numbers that is equal to the sum.

```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
    vector<int> arr {10,20,30,40,50};
    int sum = 100;
    for(int i = 0; i < arr.size(); i++){
      for(int j = i+1; j < arr.size(); j++){
        for(int k = j+1; k < arr.size(); k++)
        {
          for(int l = k+1; l < arr.size(); l++)
          {
            if(arr[i] + arr[j] + arr[k]
                + arr[l] == sum)
            {
              cout << arr[i] << arr[j]
               << arr[k] << arr[l];
            }
          }
        }
      }
    }
    return 0;
}
```

Date.................

Ques: Sort 0's & 1's.

we've given an array of 0's and 1's & we have to sort them, means, all 0's at left side & all 1's at right side.

Approach :- Here we use "Two Pointer Approach".

"start" variable will point on 0th index.

"end" variable will point on n-1 index.

"i" variable will also point on 0th index.

If arr[i] is 0, then swap it with arr[start] and increment i & start both.

If arr[i] is 1, then swap it with arr[end] and decrement only end, i will not incremented because after swapping with arr[end], arr[i] got a new value & we have to process that value again.

This loop will run until i <= end. when "i == end", loop will terminate.

Date.................

code :-

```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
   vector<int> arr{1,0,1,1,0,0,1,0};
   int start = 0;
   int end = arr.size()-1;
   int i = 0;

   while(i<=end){
      if(arr[i]==0)
      {
         swap(arr[i], arr[start]);
         i++;
         start++;
      }
      if(arr[i]==1)
      {
         swap(arr[i], arr[end]);
         end--;
      }
   }

   for(int a=0; a<arr.size(); a++)
   {
      cout << arr[a] << " ";
   }
   return 0;
}
```