Dynamic Array→

```
# include <bits/stdc++.h>
using namespace std;
int main(){
        vector <int> v1;  // Empty vector
        vector <int> v2(5); // vector of size 5, default initialized
                                      to 0 .
        vector<int> v3(5,100); // Vector of size 5; initialized to 100
        vector <int> v4  = { 1,2,3,4,5 }; // Initialized with values.
        vector <int> v5 (v4.begin(), v4.end());
```

2. Common Operations

```
v. push_back(10); // Add element at the end
v. pop_back( ); // Remove last element
v. size();          // Get number of elements
v.empty();          // Check if vector is Empty
v. Clear ();        // Remove all elements
```

3. Accessing Element

```
v[i];   // access ith Element
v.at(i);  // access ith Element with bound checking
v. front(); // first Element
v. back (); // Last element
```

Iterators

```
for (auto it = v.begin(); it != v.end(); ++it){
    cout<< *it <<" ";
}
```

```
for ( int num: v){
    Cout<<num <<"  ";
}
```

Sorting and Searching

```
sort( v.begin(), v.end());
sort ( v.rbegin() , v.rend ());
reverse(v.begin(), v.end());

auto it = lower_bound (v.begin(),v.end(),x)
          ↳ first element ≥ x

auto it = upper_bound ( v.begin(),v.end(),x)
          ↳ First element >x

binary_search(v.begin() ,v.end())
                         ↳ returns true if found
```

Removing Duplicates(Two Methods)

```
Sort( v.begin(),v.end());
v.erase ( unique(v.begin() , v.end()) , v.end());

// using unordered_set
unordered_set<int> s = s(v.begin(), v.end());
v.assign( s.begin() , s.end());
```

Erasing Elements

```
v.erase (v.begin()+i) ;//erase element at index i
v.erase (v.begin(),v.begin()+3);
              ↳ erase with first 3 elements.
```

merge

```
vector<int> result (v1.size() + v2.size());
merge (v1.begin(), v1.end(),v2.begin(),v2.end(),result.begin());
```

↳ vec1 begin vec1 end ↳ vec2 begin vec2 end ↳ resulting vector begin

↳ merges sorted Arrays

Merge without sorting (something like concat)

```
vector<int> result = v1 ; // copy v1 into result.
result.insert (result.end(), v2.begin(), v2.end());
```

→ Dynamic Arrays

```
#include<bits/stdc++.h>
class Array{
public:
        int capacity = 2;
        int length = 0;
        int * arr = new int[2]; // arr of size 2;

        Array(){};
        // Insert element in the position of Array
        void pushback(int n){
            if (length == capacity){
                resize();
            }
            // insert at next empty position
            arr[length++] = n;
        }
        void resize(){
            capacity *= 2;
            int *newarr = new int[capacity];
            // Copy Element to newarr
            for( int i = 0; i < length; i++ ){
                newArr[i] = arr[i];
            }
            arr = newArr;
             ↳ usually we should free up old arr's memory.
        }
```

```
void popback(){
 if( length > 0){
     length--;
    }
}

int get(int i){
    if ( i< length){
        return arr[i]
    }
}
void insert ( int i , int n){
    if( i < length){
        arr[i] =n;
        return;
    }
}
void print(){
    for( int i=0; i ≤ length; i++){
        cout << arr[i] << ' ';
    }
}
```

→

```
# include <bits/stdc++.h>
using namespace std;
class Array {
public:
        int capacity = 2;
        int length = 0;

        int *arr = new int[capacity];
```

```
Array(){}

~Array(){
      delete[] arr;
}
void pushback(intn){
    if( length == capacity){
         resize();
    }
    arr[length++] = n;
}
void   resize() {
     capacity *= 2;
     int * newArr = new int [capacity];
     for(int i = 0; i< length; i++){
          newArr[i] = arr[i];
     }
     delete arr;
     arr = newArr;
}
void poplack(){
    if( length>0){
         arr[length--] = 0;
    }
}
void get(int i
    if( i< 0 || i >= length){
         Cout<<"index out of bounds";
          return -1;
    }
    return arr[i];
```

length++(post increment)
↳ will first use 0 then increment it to 1.

```
    }

   void insert( int i , int n){
       for( i <0 || i >= length ){
           Cout << "Error index out of bound";
           return -1;
       }
       arr[i] =n;
   }
  void print(){
    for(int i <0; i <length; i++){
       cout<<arr[i]<<' ';
     }
  }
```