# Vectors

Vectors are the same as dynamic arrays with the **ability to resize themselves automatically** when an element is inserted or deleted, with their storage being handled automatically by the container.

Vector elements are placed in **contiguous storage** so that they can be accessed and traversed using iterators.

In vectors, data is inserted at the end. Inserting at the end takes differential time, as sometimes the array may need to be extended. Removing the last element takes only constant time because no resizing happens. Inserting and erasing at the beginning or in the middle is linear in time.

1. To use vectors, you have to include the vector header file:

```
#include <vector>
```

2. The syntax to declare a vector is :

```
std::vector<dataType> vectorName;
```

3. The different ways to initialize a vector are :
    - Initializing using list

    ```
    std::vector<dataType> name({value1, value2, value3, ....});
    ```

    - Initializing with a single value

    ```
    vector<dataType> name(size, value);
    ```

    - Initializing from another vector

    ```
    vector<dataType> name(other_vec);
    ```

4. We can access the values inside a vector using square brackets just like arrays.

```
vector<int> v = {1, 2, 3};
cout<<v[2]; // Output - 3
```

## Iterators

Iterators are used to get the address of specific position of a container. Some iterators used in vectors are:

1. **begin()** - Returns an iterator pointing to the first element in the vector.
2. **end()** - Returns an iterator pointing to the theoretical element that follows the last element in the vector.
3. **rbegin()** - Returns a reverse iterator pointing to the last element in the vector (reverse beginning). It moves from last to first element.
4. **rend()** - Returns a reverse iterator pointing to the theoretical element preceding the first element in the vector (considered as reverse end).
   You can also use **cbegin()** and **cend()** if you need constant iterators or combine with r adapter to get **crbegin()** and **crend()**.

```
#include <iostream>
#include <vector>
using namespace std;
int main(){
    vector<int> v = {1, 2, 3, 4, 5};
    for(vector<int>::iterator it = v.begin(); it != v.end(); it++){
        cout << *(it) << " ";
    }

    cout<<endl;

    for(auto it = v.end()-1; it != v.begin()-1 ; it--){
        cout << *(it) << " ";
    }

    cout<<endl;

    for(auto it: v){
        cout << it << " ";
    }
}
```

## Functions

1. **size()** - Returns the number of elements in the vector.
2. **empty()** - Returns whether the container is empty.
3. **front()** - Returns a reference to the first element in the vector.
4. **back()** - Returns a reference to the last element in the vector.
5. **push_back()** - It push the elements into a vector from the back.
6. **pop_back()** - It is used to pop or remove elements from a vector from the back.
7. **emplace()** - It extends the container by inserting new element at position.
8. **emplace_back()** - It is used to insert a new element into the vector container, the new element is added to the end of the vector.
9. **erase()** - It is used to remove elements from a container at the specified position.
10. **swap()** - It is used to swap contents of one vector with another vector of same type. Sizes may differ.
11. **insert()** - It inserts new elements before the element at the specified position.

```cpp
#include <iostream>
#include <vector>
using namespace std;
int main(){
    vector<int> v = {-1, 0};
    cout << v.size() << "\n"; // Output : 2
    cout << v.empty() << "\n"; // Output : 0
    cout << v.front() << "\n"; // Output : -1
    cout << v.back() << "\n"; // Output : 0

    v.push_back(1);
    int n = v.size();
    cout << v[n - 1] << "\n"; // Output : 1

    v.pop_back();
    cout << v[v.size()-1] << "\n"; // Output : 0

    v.emplace(v.begin(), -2);
    cout << v[0] << "\n"; // Output : -2

    v.emplace_back(1);
    cout << v[v.size() - 1] << "\n"; // Output : 1

    vector<int> v2 = {9, 8, 7};
    v.swap(v2);
    cout << "After Swap \nVector 1: ";
```

```
    for (int i = 0; i < v.size(); i++)
        cout << v[i] << " "; // Output : 9 8 7
    cout << "\nVector 2: ";
    for (int i = 0; i < v2.size(); i++)
        cout << v2[i] << " "; // Output : -2 -1 0 1

    v2.erase(v2.begin());
    cout << v2[0] << "\n"; // Output : -1
    v2.erase(v2.begin()+1 , v2.end());
    for (int i = 0; i < v2.size(); i++)
        cout << v2[i] << " "; // Output : -1

    v.insert(v.begin(), 5);
    cout << v[0] << "\n"; // Output : 5
    v.insert(v.begin() + 2, 3, 100);
    for (int i = 0; i < v.size(); i++)
        cout << v[i] << " "; // Output : 5 9 100 100 100 8 7
}
```

The time complexity for doing various operations on vectors is-

- Random access – constant O(1)
- Insertion or removal of elements at the end – constant O(1)
- Insertion or removal of elements – linear in the distance to the end of the vector O(N)
- Knowing the size – constant O(1)
- Resizing the vector- Linear O(N)