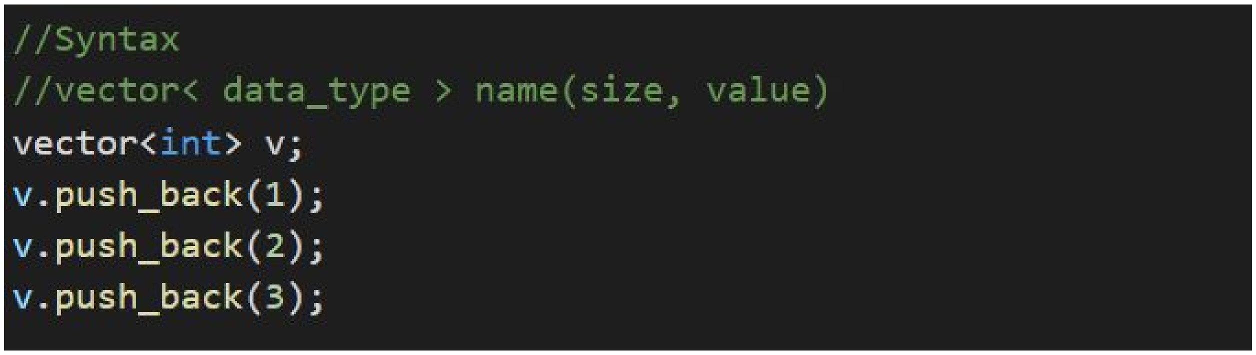
Vectors and Pairs in C++

Vectors in C++

Vectors are dynamic arrays. They have the ability to resize itself when it gets filled. The size of the vector gets doubled each time when they get filled.

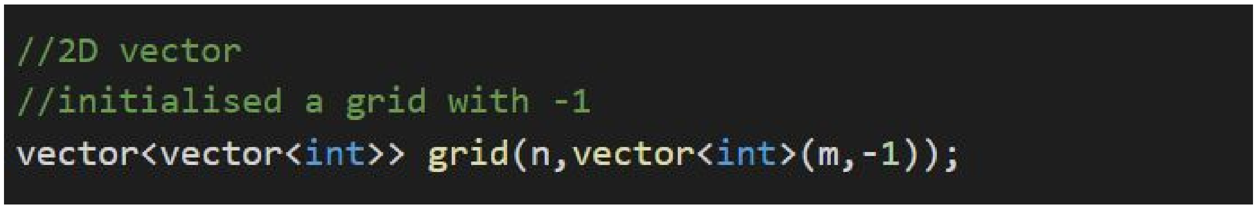
Syntax:



```
//Syntax
//vector< data_type > name(size, value)
vector<int> v;
v.push_back(1);
v.push_back(2);
v.push_back(3);
```

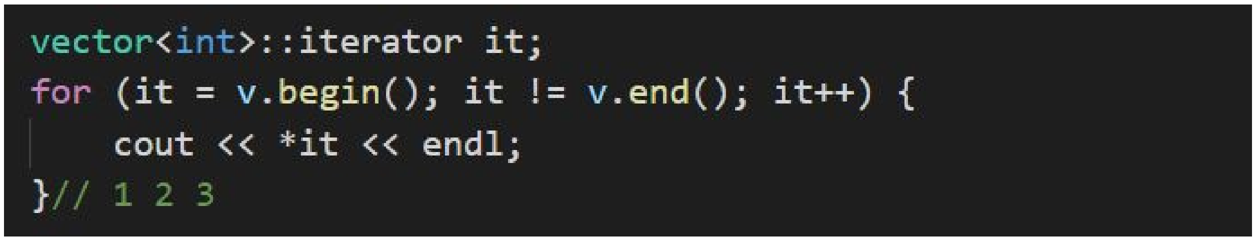
2D vector

Declared a vector of size nxm.



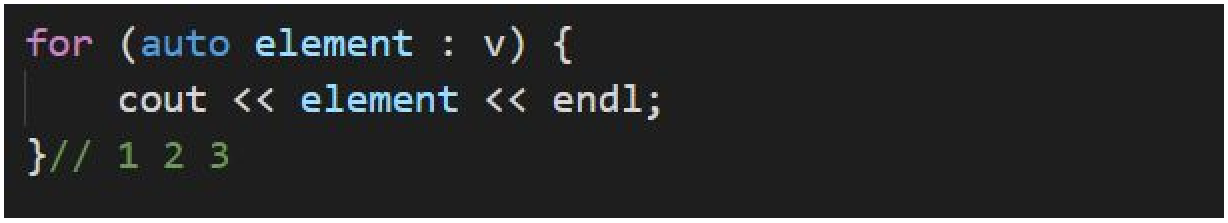
```
//2D vector
//initialised a grid with -1
vector<vector<int>> grid(n,vector<int>(m,-1));
```

Iterators



```
vector<int>::iterator it;
for (it = v.begin(); it != v.end(); it++) {
    cout << *it << endl;
}// 1 2 3
```

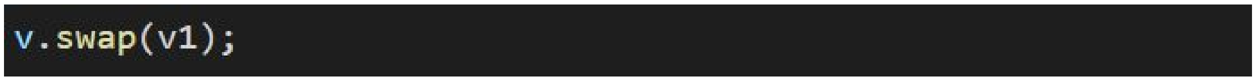
For each loop



```
for (auto element : v) {
    cout << element << endl;
}// 1 2 3
```

vector :: swap()

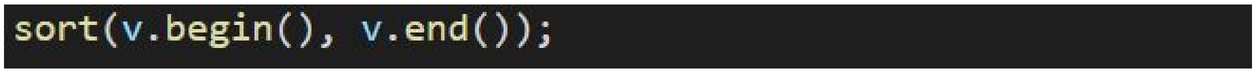
v.swap(v1)



```
v.swap(v1);
```

sort()

Sorting a vector, we can also add custom operators to the sorting functions.

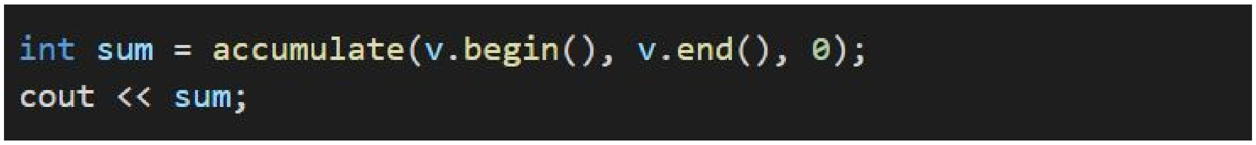


```
sort(v.begin(), v.end());
```

To get the sum of the vector

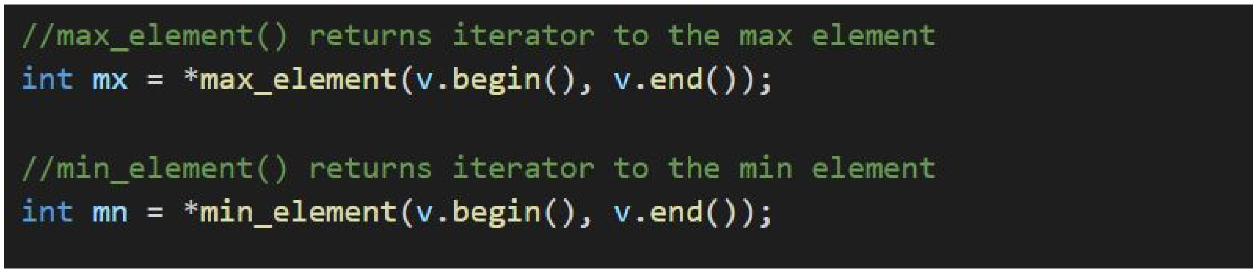
We use accumulate(), defined in the *<numeric>* library.

#define <numeric>



```
int sum = accumulate(v.begin(), v.end(), 0);
cout << sum;
```

To get Max_element/min_element of the vector



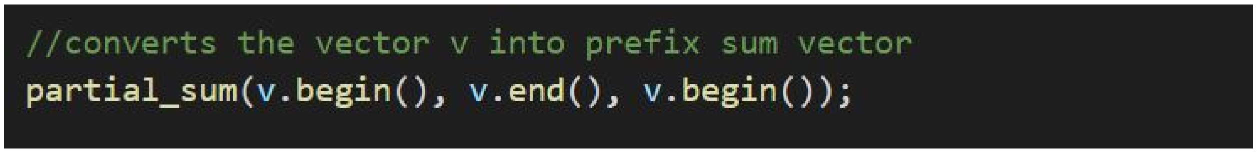
```
//max_element() returns iterator to the max element
int mx = *max_element(v.begin(), v.end());

//min_element() returns iterator to the min element
int mn = *min_element(v.begin(), v.end());
```

To convert the vector into a prefix sum vector(Quite useful)

Example: v = {1,2,3}

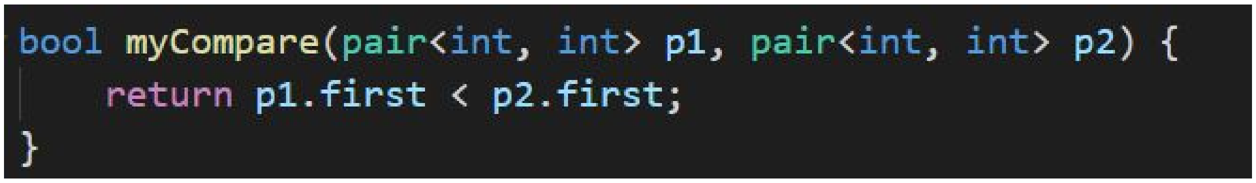
Prefix_sum v = {1,3,6}



```
//converts the vector v into prefix sum vector
partial_sum(v.begin(), v.end(), v.begin());
```

Custom comparators

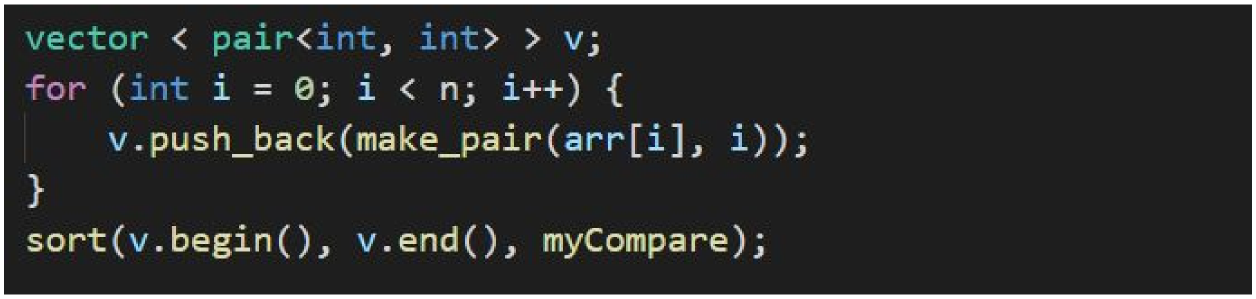
We can write custom comparators according to our needs.
They are very useful in sorting. It can be generic as well.



```
bool myCompare(pair<int, int> p1, pair<int, int> p2) {
    return p1.first < p2.first;
}
```

Pairs

Pair class couples together the pair of values, which may be of different types (T1 and T2). The individual values can be accessed through its public members first and second.
We can also form nested pairs as well, for example, pair<int, pair <int, int> >
make_pair() is used to couple the values. We can also do {a, b} to make pairs.



```
vector < pair<int, int> > v;
for (int i = 0; i < n; i++) {
    v.push_back(make_pair(arr[i], i));
}
sort(v.begin(), v.end(), myCompare);
```