Vectors in C++

Introduction

Vectors are very similar to dynamic arrays for which we don't have to determine the size as done in case of arrays.

Vectors are sequence containers representing arrays that can change in size.

They can be used in the same way as the arrays and use the contiguous storage locations for storing data as in case of arrays. Just the difference is that these containers can change their size as per the requirements.

Dynamic implementation of arrays is used by vectors to store the elements with some initial size. In case of arrays, we need to reallocate the memory and then copy back these elements which is a time-expensive task. This is easily avoidable in case of vectors as these directly allocates some extra storage to accommodate the elements. There is a possibility that a vector's capacity is more than the number of elements present in it as it increases it's space using square function.

This is achieved by *amortized constant time* complexity which is internally achieved in C++.

Syntax for vector declaration:

```
vector<data_type> vector_name;
```

For example: To declare a vector of integer with a name v:

```
vector<int> v;
```

To dynamically allocate the same:

```
vector<int> *v = new vector<int>();
```

Header file used is : **#include<vector>**

Some important functions of vector:

push_back	Add element at the end
pop_back	Delete last element
insert	Insert elements
size	Return size
capacity	Return size of allocated storage capacity

Practice to implement a vector in C++

Statement: Perform the following tasks:

1. Create a vector of integer.
2. Insert the elements from 1 to 10 and print size along with capacity at each step.

Solution Code:

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

int main() {
        vector<int> v;                          // Part 1

        for(int i = 1; i <= 10; i++) {          // Part 2
                cout << v.capacity() << " " << v.size() << endl;
                v.push_back(i);
        }
        return 0;
}
```

Output:

```
0 0
1 1
2 2
4 3
4 4
8 5
8 6
8 7
8 8
16 9
```

Observation:

You can see that capacity doubles each time as the vector gets an extra element. But size represents the total number of elements present in the vector irrespective of its capacity.