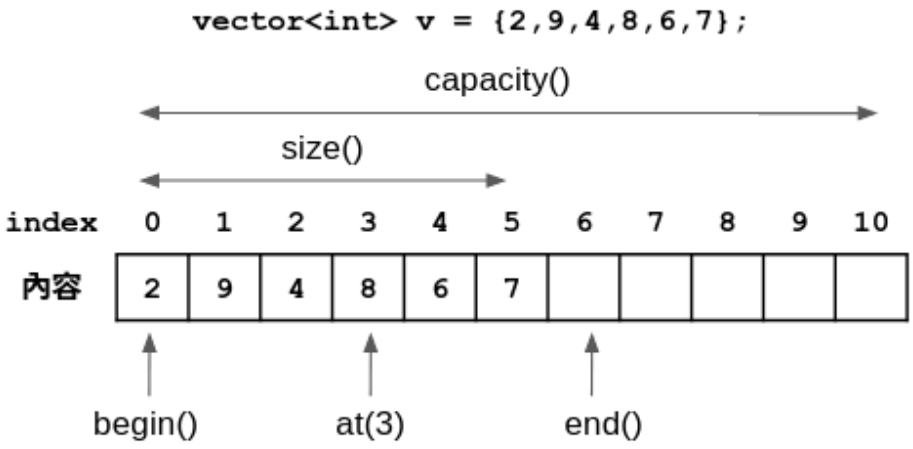
Tutorial of Using Vector in C++

A vector is a ***dynamic array*** that can resize itself if an element is inserted or deleted. The vector elements are contained in a contiguous storage and the container handles the storage automatically.



There are four types of functions you can use in vector: Modifier, Iterator, Capacity and Access.

Modifiers: As the name suggests, these are functions that are used to modify or change the vector. For example, assign() is used to clear the existing value and assigns a new value to the vector.

Iterators: Iterator functions are used to move across or iterate through the elements of the vector. For example, the end() function is used to point to the last element of the vector.

Capacity: The functions that lie under capacity have something to do with the size like, changing the size of the vector. For example, the resize(n) function is used to change the size of the vector.

Access: The functions are used to refer an element at a position. For example, the at(1) is used to refer an element at second position.

Common functions

Function Name	Function Type	Description
push_back():	Modifier	This function allows you to add a new element at the end of the vector.
pop_back():	Modifier	It is used to remove or delete the last element from the vector.
insert():	Modifier	This function is used to add a new element before a specified position inside the vector.
erase():	Modifier	It is used to remove the element from the container at a specified position or a range.

swap():	Modifier	It is used to swap the contents between the vectors, but it should be of the same type.
assign():	Modifier	It is used to assign a new value to the vector by substituting the old value.
clear():	Modifier	This function is used to remove all elements from the vectors.
begin():	Iterator	This function returns the iterator to the first element of the vector container.
end():	Iterator	This function returns the iterator to the last element of the vector container.
rbegin()	Iterator	Returns a reverse iterator pointing to the last element in the vector (reverse beginning). It moves from last to first element
rend()	Iterator	Returns a reverse iterator pointing to the theoretical element preceding the first element in the vector (considered as reverse end)
cbegin()	Iterator	Returns a constant iterator pointing to the first element in the vector.
cend()	Iterator	Returns a constant iterator pointing to the theoretical element that follows the last element in the vector.
crbegin()	Iterator	Returns a constant reverse iterator pointing to the last element in the vector (reverse beginning). It moves from last to first element
crend()	Iterator	Returns a constant reverse iterator pointing to the theoretical element preceding the first element in the vector (considered as reverse end)
size():	Capacity	This function is used to return the number of elements inside the vector.
max_size():	Capacity	It is used to return the maximum size of the vector.
resize(n):	Capacity	This function is used to resize the container, i.e., if the given size is greater than n, then the extra elements are removed. And if the size is less than n, then some extra elements are added.
capacity():	Capacity	This function returns the size that is currently allocated to the vector.
empty():	Capacity	It checks whether the vector is empty or not, and it returns true if a vector is empty else it returns false.
shrink_to_fit()	Capacity	Reduces the capacity of the container to fit its size and destroys all elements beyond the capacity.
reserve()	Capacity	Requests that the vector capacity be at least enough to contain n elements.
reference_operator(g)	Access	Returns a reference to the element at position ‘g’ in the vector ‘g’ in the vector
at(g)	Access	Returns a reference to the element at position ‘g’ in the vector
front()	Access	Returns a reference to the first element in the vector
back()	Access	Returns a reference to the last element in the vector
data()	Access	Returns a direct pointer to the memory array used internally by the vector to store its owned elements.

Example: Vector in C++

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

class Student
{
public:

	Student()
	{
		ID = 0; Name = ""; GPA = 0.00;
	}

	void setData(int _id, string _name, double _gpa)
	{
		ID = _id;
		Name = _name;
		GPA = _gpa;
	}

	string getName()
	{
		return Name;
	}

	double getGPA()
	{
		return GPA;
	}

private:
	int ID;
	string Name;
	double GPA;
};

int main()
{
	//Define vector for class student
	vector<Student> vStudent;
	Student student;

	//Add 3 students into vector
	student.setData(1, "Peter", 4.00);
	vStudent.push_back(student);

	student.setData(2, "Mary", 3.00);
	vStudent.push_back(student);

	student.setData(3, "Sam", 2.00);
	vStudent.push_back(student);


	//list out all student name and GPA
	cout << fixed << setprecision(2);
	//method 1 by iterator
	cout << "List by iterator:" << endl;
	for (auto it = vStudent.begin(); it != vStudent.end(); it++)
```

```
		cout << it->getName() << " has GPA = " << it->getGPA() << endl;
	cout << endl;

	//method 2 by element index
	cout << "List by element index:" << endl;
	for (int i=0; i<vStudent.size();i++)
		cout << vStudent[i].getName() << " has GPA = " << vStudent[i].getGPA() << endl;
	cout << endl;

	//Search Mary
	cout << "Search Mary:" << endl;
	for (int i = 0; i < vStudent.size(); i++)
		if (vStudent[i].getName() == "Mary")
			cout << "n-th element of " << vStudent[i].getName() << " is " << i << endl;
	cout << endl;

	//point to specific an element
	cout << "List specific elements:" << endl;
	cout << "The first element is " << vStudent.front().getName() << endl;
	cout << "The last element is " << vStudent.back().getName() << endl;
	cout << "The second element is " << vStudent.at(1).getName() << endl;
	cout << endl;

	//add new student at specific position
	cout << "Add Ricky at specific position 2:" << endl;
	student.setData(4, "Ricky", 3.50);
	vStudent.insert(vStudent.begin()+1, student);
	for (auto x : vStudent)
		cout << x.getName() << " has GPA = " << x.getGPA() << endl;
	cout << endl;

	//remove last student
	cout << "Remove last student Sam:" << endl;
	vStudent.pop_back();
	for (auto x : vStudent)
		cout << x.getName() << " has GPA = " << x.getGPA() << endl;
	cout << endl;


	//remove a student at specific position
	cout << "Remove the second element Mary:" << endl;
	vStudent.erase(vStudent.begin() + 1);
	for (auto x : vStudent)
		cout << x.getName() << " has GPA = " << x.getGPA() << endl;
	cout << endl;


	return 0;
}
```

Output

List by iterator:

Peter has GPA = 4.00

Mary has GPA = 3.00

Sam has GPA = 2.00

List by element index:

Peter has GPA = 4.00

Mary has GPA = 3.00

Sam has GPA = 2.00

Search Mary:

n-th element of Mary is 1

List specific elements:

The first element is Peter

The last element is Sam

The second element is Mary

Add Ricky at specific position 2:

Peter has GPA = 4.00

Ricky has GPA = 3.50

Mary has GPA = 3.00

Sam has GPA = 2.00

Remove last student Sam:

Peter has GPA = 4.00

Ricky has GPA = 3.50

Mary has GPA = 3.00

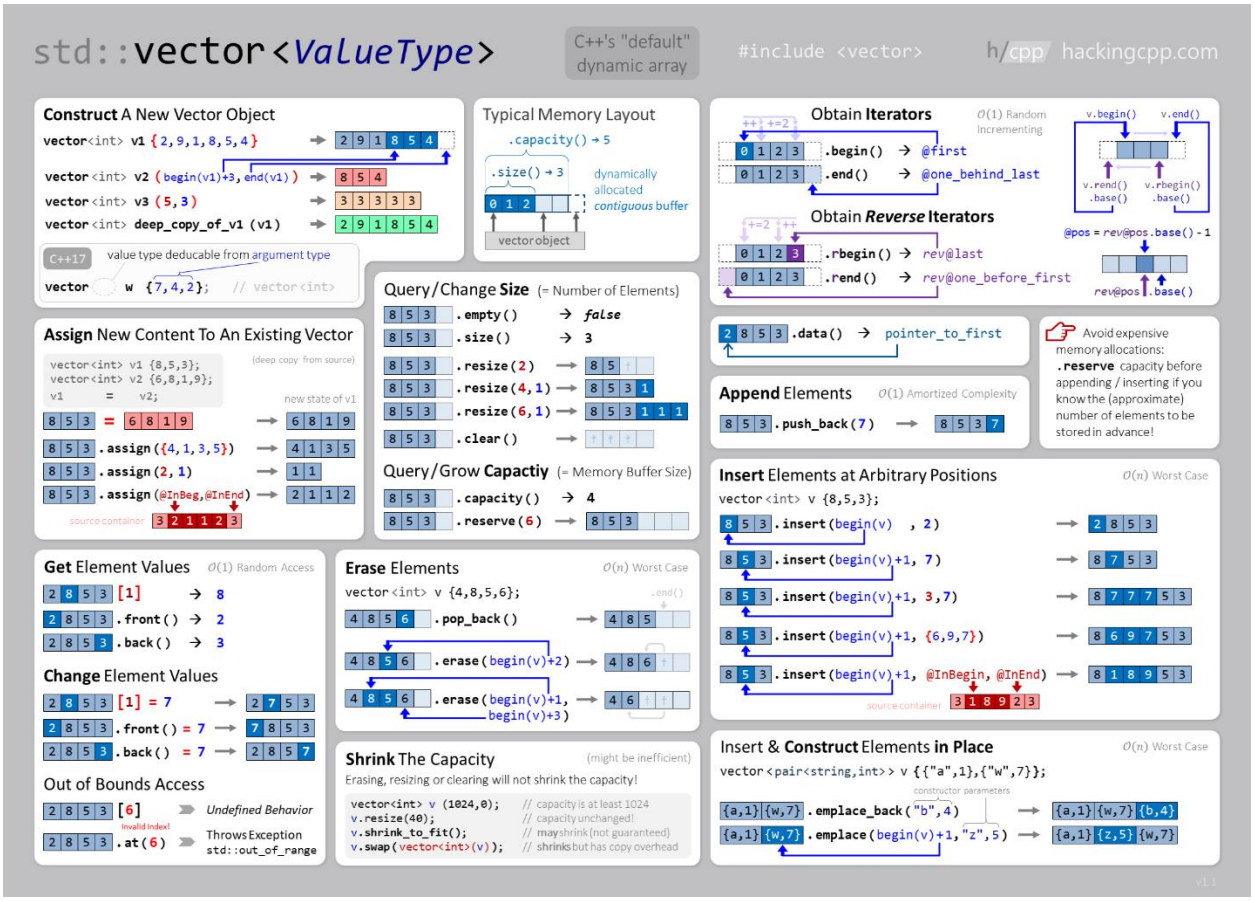
Remove the second element Mary:

Peter has GPA = 4.00

Mary has GPA = 3.00

Reference

1. https://hackingcpp.com/cpp/std/vector.html



2. https://www.geeksforgeeks.org/creating-a-vector-of-class-objects-in-cpp/

3. https://www.tutorialspoint.com/cpp_standard_library/vector.htm

4. https://www.programiz.com/cpp-programming/vectors

- End -