What are Vectors

Vectors in C++ are the dynamic arrays that are used to store data. Unlike arrays, which are used to store sequential data and are static in nature, Vectors provide more flexibility to the program. Vectors can resize itself automatically when an element is inserted or deleted depending on the need of the task to be executed. It is not the same in an array where only a given number of values can be stored under a single variable name. In vectors, data is inserted at the end of it.

Note 1: Vector elements can be accessed just like array elements: vector[m];

Note 2: You should never assign vector elements like its an array because it will not increase the vector size. For example, this: vector[m] = n; will not work, instead learn how to use the modifiers.

How to create a Vector:

vector< object\_type > variable\_name;

Example:

#include <vector>

Using namespace std;

Int main() {

vector<int> intVector;

vector<string> stringVector;

}

Vector Iterators:

**begin():** This function returns an iterator pointing to the **first** element in the vector.

**end():** The end() function returns an iterator pointing to the **last** element in the vector.

Vector Modifiers:

**assign(n, m)**: erases everything in an array (if there is anything) and assigns the element m, n times

n: number of times to assign a value

m: the value to assign

Example:

vector<int> aHundredTens;

//adds 100 tens to the vector:

aHundredTens.assign(100, 10); //10 10 10 10 10 10 10 10 …

//removes all tens and assigns 3 fives:

aHundredTens.assign(3, 5); // 5 5 5

**push\_back(m)**:  adds the element m into a vector from the back. Increases vector size by one element.

Example:

vector<int> vec;

//adds a single 10 in the vector

vec.push\_back(10); //10

**pop\_back()**: removes one element from a vector from the back. Reduces the size of the vector by one element.

Example:

vector<int> vec;

vec.assign(3, 10); //10 10 10

vec.pop\_back(); //10 10

**insert(m, k, n)**: This function inserts an element n in the position of m (m can only be vector.begin() or vector.end()). The second argument is optional. It determines how many times to add the element n in the position m.

Example:

vector<int> vec;

vec.assign(3, 10); //10 10 10

vec.insert(vec.end(), 5); // 10 10 10 5

vec.insert(vec.begin() + 1, 2, 3); //10 3 3 10 10 5

**erase(n, m)**: removes elements ranging from position n to position m. m is optional, and if not used, this function will only remove the element in the position n. Reduces the size of the vector by range or if not used, by one. (n and m must include vector.begin() or vector.end())

Example:

vector<int> vec;

vec.assign(5, 10); //10 10 10 10 10

vec.insert(vec.begin() + 1, 5); 10 5 10 10 10 10

vec.remove(vector.begin() + 1); //10 10 10 10 10

vec.remove(vector.begin(), vector.end() - 1) //10

**swap()**: swap() function is used to swap the contents of one vector with another vector of the same type. Sizes may differ.

**clear()**: clear() function is used to remove all the elements of the vector container

After ChatGPT answer:

In C++, a vector is a dynamic array that can grow or shrink in size as needed during runtime. Here is the basic syntax for declaring and using a vector in C++:

1. Declaration:
2. #include <vector> // include the vector library
3. // syntax for declaring a vector of integers
4. std::vector<int> myVector;
5. // syntax for declaring a vector of strings
6. std::vector<std::string> myStringVector;
7. Initialization:
8. // initialize vector with default values
9. std::vector<int> myVector{1, 2, 3, 4, 5};
10. // initialize vector with a specified size and default value
11. std::vector<int> myVector(10, 0); // initializes vector of size 10 with all elements set to 0
12. // initialize vector from another vector
13. std::vector<int> myOtherVector{6, 7, 8, 9, 10};
14. std::vector<int> myVector(myOtherVector);
15. Accessing elements:
16. // access element at index i
17. int element = myVector[i];
18. // access first element
19. int firstElement = myVector.front();
20. // access last element
21. int lastElement = myVector.back();
22. // get number of elements in the vector
23. int vectorSize = myVector.size();
24. Adding and removing elements:

1. // add element to end of vector
2. myVector.push\_back(6);
3. // remove last element
4. myVector.pop\_back();
5. // insert element at index i
6. myVector.insert(myVector.begin() + i, value);
7. // remove element at index i
8. myVector.erase(myVector.begin() + i);
9. // clear all elements in the vector
10. myVector.clear();
11. Iterating through the vector:
12. // iterate through the vector using a for loop
13. for (int i = 0; i < myVector.size(); i++)
14. {
15. std::cout << myVector[i] << " ";
16. }
17. // iterate through the vector using a range-based for loop
18. for (int element : myVector)
19. {
20. std::cout << element << " ";
21. }
22. // iterate through the vector using an iterator
23. for (auto it = myVector.begin(); it != myVector.end(); it++)
24. {
25. std::cout << \*it << " ";
26. }

Note: the **auto** keyword in the iterator loop allows the compiler to automatically determine the data type of the iterator.