Vector -

19 February 2023 14:26

```
std::vector<int> v;                              creating vector
V = vector<int>();                               creating vector
std::vector<int> v(10);                          creating vector of capacity 10 with default value 0
std::vector<int> v(10,-1);                creating vector of capacity 10 with default value - 1
std::vector<int> v {1,2,3}                       creating vector with initial value 1,2,3
std::vector<int> v (v1.begin(),v1.end())      creating vector by coping value from another vector

v.insert(it+4,v1.begin(),v1.end())               inserting v1 vector into v vector from index 4

v.assign(7,100)                                  new vector with 7 element 100 value
v.at(i)                                          access location i
v.back()                                         returns the reference to the last element
v.data()                        returns a direct pointer to the memory array used internally by vector
                                to store its own element


v.clear()                                                delete all elements
v.empty()                                                returns 1 if vector is empty
v.size()                                                  no of element present in vector
v.capacity()                                             no of element vector can store
v.max_size()                                    returns the maximum number of element that the
                                                vector  can hold.


v.begin()                                                returns an iterator pointing to the first element
v.end()
v.cbegin()          returns a constant iterator pointing to the first element in the container
v.cend()            returns a constant iterator pointing to the past the last element in the container
v.rbegin()
v.rend()
v.crbegin()         returns a const_reverse_iterator pointing to the last element in the container
v.crend()   returns a const_reverse_iterator pointing to the reversed end of the sequence

all_of(v1.begin(),v1.end(),[](int i){return i>-1;})      return 1 if all value greater than –1 else 0
any_of(v1.begin(),v1.end(),[](int i){return i>-1;})      return 1 if any value greater than –1 else 0
binary_search(v.begin(),v.end(),val)                     return 1 if val in v else 0
count(v.begin,v.end(),val)                               count the number of val in v
count_if(v1.begin(),v1.end(),[](int i){return i>1;})     count with custom function
fill(v.begin(),v.begin()+3,val)                          fill first 3 values with val
find(v.begin(),v.end(),val)                 returns an iterator to location if find element else v.end()
lexicographical_compare(foo,foo+5,bar,bar+9)    lexicographical comparison
max_element(v.begin(),v.end())            return an iterator pointing to the max element
      min_element(v.begin(),v.end())          return an iterator pointing to the min element


v.erase(1)                                       delete element store at location 1
v.erase(v.begin()+2,v.begin()+4)                 delete element store at location 2 to 3


v1 == v2 ? true : false ;                                checking if two vectors are equal



for(int i=0; i<v.size(); i++){                           vector traversing
      cout<<v[i]<<" ";

for(auto el: v){                                         vector traversing
      cout<<el<<" ";
      }
```

I would recommend reading in the line into a string, then splitting it based on the spaces. For this, you can use the getline(...) function. The trick is having a dynamic sized data structure to hold the strings once it's split. Probably the easiest to use would be a vector.

```
#include<string>#include<vector>...
  string rawInput;
  vector<String> numbers;
  while( getline( cin, rawInput, ' ') )
  {
    numbers.push_back(rawInput);
  }
```

From <https://stackoverflow.com/questions/13096719/read-input-numbers-separated-by-spaces>

```
#include <bits/stdc++.h>
#include <string>
```

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

int main()
{
    string rawInput;
  vector<int> numbers;
  while( getline( cin, rawInput, ' ' ) )
  {
    numbers.push_back(atoi(rawInput.c_str()));
  }
    for(auto i:numbers){
      cout<<i<<endl;
    }
    cout << "Hello, World!";
    return 0;
}
```