CS6150 – Advanced Programming

Standard Template Library

September 22, 2025

A simple set of tasks

- Read strings from the user till the user enters “END”.
- Store the strings.
- Search a given value in the input strings.
- Modify the strings to be in upper case.

A simple set of tasks

- Read strings from the user till the user enters “END”.
  - How many strings will be given?
  - Not known in advance, so make an estimate.
- Store the strings.
  - Store strings in an array – size over-estimated.
  - Store strings in a list.
- Search a given value in the input strings.
  - Iterate over the stored strings.
- Modify the strings to be in upper case.
  - Iterate over the strings and keep modifying.

vector: a templatized container

vector: a dynamic array that can grow and shrink

- contiguous memory allocation
- memory management is taken care
- supports random access
- member functions include : size(), push_back(), pop_back()

vector: a templatized container

vector: a dynamic array that can grow and shrink

- contiguous memory allocation
- memory management is taken care
- supports random access
- member functions include : size(), push_back(), pop_back()

```
1  #include <vector>
2            ..
3            vector<string> myVec;
4            // a vector of strings
5
6            vector<int> myIntVec;
7            // a vector of integers
8            vector<Student> myStudVec;
9            // a vector of Students
10
11     ..
```

Using vector of strings

```
1  #include<iostream>
2  #include<vector>
3  using namespace std;
4  int main() {
5        vector<string> myVec;
6        string st;
7
8        while (1) {
9              cin >> st;
10             if (st == "END") break;
11             myVec.push_back(st);
12       }
13
14       int numOfInputs = myVec.size();
15       for (int i = 0; i < numOfInputs; i++) {
16            cout << myVec[i] << "   " ;
17            // array style access.
18       }
19       cout << endl;
20 }
```

iterator: an object that points to another object

vector<string>::iterator : an iterator for a vector of strings

iterator: an object that points to another object

vector<string>::iterator : an iterator for a vector of strings

- iterators are generalization of pointers
- used to iterate over a range of objects
- containers need to provide a way to access elements
- enables generic algorithms that work on different containers

iterator: an object that points to another object

vector<string>::iterator : an iterator for a vector of strings

- iterators are generalization of pointers
- used to iterate over a range of objects
- containers need to provide a way to access elements
- enables generic algorithms that work on different containers

```
1 #include <vector>
2
3 vector<string>::iterator it;
4       for (it=myVec.begin(); it!=myVec.end(); it++) {
5           cout << *it << "  " ;
6       }
```

Accessing vector using iterator

```
1  #include<iostream>
2  #include<vector>
3  using namespace std;
4  int main() {
5        vector<string> myVec;
6        string st;
7
8        while (1) {
9              cin >> st;
10             if (st == "END") break;
11             myVec.push_back(st);
12       }
13
14       vector<string>::iterator it;
15       for (it=myVec.begin(); it!=myVec.end(); it++) {
16            cout << *it << "  " ;
17       }
18       cout << endl;
19 }
```

Using STL algorithm for search

Let us use what STL provides : generic algorithm find

find:

- iterator to the first element in range
- iterator to the element just after the range
- value to be searched

Using STL algorithm for search

Let us use what STL provides : generic algorithm find

find:

- iterator to the first element in range
- iterator to the element just after the range
- value to be searched

```
1  #include<algorithm>
2  ..
3  int main() {
4      vector<string> myVec;
5
6      // same code as above to populate vector
7      vector<string>::iterator it;
8      it = find (myVec.begin(),myVec.end(),"CS6150");
9
10     if (it != myVec.end())
11        cout << "found CS6150 at "
12       << distance (myVec.begin(), it) << endl;
13 }
```

Searching in a range

```
1  int main() {
2      vector<int> myVec;
3      for (int i = 1; i <= 100; i++)
4          myVec.push_back(i);
5
6      vector<int>::iterator it;
7
8      it = find (myVec.begin(), myVec.end(), 10);
9      if (it != myVec.end())
10         cout << "10 is at position " << distance(
               myVec.begin(), it) << endl;
11     else cout << "not found 10 in range\n";
12
13     it = find (myVec.begin()+20, myVec.begin()+25,
           10);
14     if (it != myVec.begin()+25)
15         cout << "10 is at position " << distance(
               myVec.begin(), it) << endl;
16     else cout << "not found 10 in range 20--24\n";
17 }
```

Replace X by Y

```
1  using namespace std;
2  int main() {
3        vector<string> myVec;
4
5        // same code as above to populate vector.
6        replace (myVec.begin(), myVec.end(), "CS6150",
             "CS6380");
7
8        for (auto it = myVec.begin(); it != myVec.end()
             ; it++) {
9          cout << *it << " " << endl;
10       }
11       cout << endl;
12 }
```

What have we learnt?

STL is a collection of common data structures and algorithms.

- **Container:** An object that stores a collection of other objects.
  - implemented as class templates.

What have we learnt?

STL is a collection of common data structures and algorithms.

- **Container:** An object that stores a collection of other objects.
  - implemented as class templates.
  - **vector** : a sequence container.

What have we learnt?

STL is a collection of common data structures and algorithms.

- **Container:** An object that stores a collection of other objects.
  - implemented as class templates.
  - **vector** : a sequence container.
- **Iterator:** A container class may an iterator used to iterate through the elements of the container.
  - vector has random access iterator

What have we learnt?

STL is a collection of common data structures and algorithms.

- **Container:** An object that stores a collection of other objects.
  - implemented as class templates.
  - **vector** : a sequence container.
- **Iterator:** A container class may an iterator used to iterate through the elements of the container.
  - vector has random access iterator
- **Operations for iterators:** A container iterator may support one or more operations.
  - begin iterator of vector supports ++, *, + amongst others.

What have we learnt?

STL is a collection of common data structures and algorithms.

- **Container:** An object that stores a collection of other objects.
  - implemented as class templates.
  - **vector** : a sequence container.
- **Iterator:** A container class may an iterator used to iterate through the elements of the container.
  - vector has random access iterator
- **Operations for iterators:** A container iterator may support one or more operations.
  - begin iterator of vector supports ++, *, + amongst others.
- **Algorithms:** A set of functions that can be used on range of elements in the container.
  - examples : find, replace, distance
  - work with iterators, can be used with different containers, work seamlessly over various ranges.

Containers, Iterators, Algorithms

Containers

- **Sequence Containers:** vector, deque (deck), list
- **Associative Containers:** set, multiset, map, multimap, unordered variants of these
- **Container Adaptors:** queue, priority_queue, stack

Containers, Iterators, Algorithms

Containers

- **Sequence Containers:** vector, deque (deck), list
- **Associative Containers:** set, multiset, map, multimap, unordered variants of these
- **Container Adaptors:** queue, priority_queue, stack

Iterators

- allow us to access objects in the container

Containers, Iterators, Algorithms

Containers

- **Sequence Containers:** vector, deque (deck), list
- **Associative Containers:** set, multiset, map, multimap, unordered variants of these
- **Container Adaptors:** queue, priority_queue, stack

Iterators

- allow us to access objects in the container

STL generic algorithms

- algorithms work on iterators rather than containers.
- compose algorithm with container : find on list, find on set, find on vector

Containers, Iterators, Algorithms

Containers

- **Sequence Containers:** vector, deque (deck), list
- **Associative Containers:** set, multiset, map, multimap, unordered variants of these
- **Container Adaptors:** queue, priority_queue, stack

Iterators

- allow us to access objects in the container

STL generic algorithms

- algorithms work on iterators rather than containers.
- compose algorithm with container : find on list, find on set, find on vector invoke algorithm with iterator for that container.
- templates provide compile time safety for combinations of containers, iterators and algorithms.