CS-1201 Object Oriented Programming

Standard Template Library

Arbish Akram

Department of Computer Science
Government College University

Standard Template Library

- The Standard Template Library contains many templates for useful algorithms and data structures.
- C++ STL is a set of data structures and algorithms that are commonly used during coding.
- For example, when solving a problem where a linked list is required, we can utilize the built-in `list` in the C++ STL library, instead of creating a linked list from scratch.
- The STL consists of three main components:
  - **Algorithms**
  - **Containers**
  - **Iterators**

Containers

A container is a generic class which implements a certain data structure.

1. Sequence container
   - Array
   - Vector
   - Queue
   - List
2. Associative containers
   - Set
   - MultiSet
   - Map
   - MultiMap

Vector

A `vector` has several advantages over an array:

- **No need to declare size**: Unlike an array, you do not need to declare the number of elements.
- **Dynamic resizing**: `Vectors` automatically increase their size when new elements are added.
- **Simpler syntax**: You can retrieve the number of elements using simpler syntax than with an array.

To use `vector`, include the header file:

```
#include <vector>
```

To declare a `vector` of integers:

```
vector<int> myVector;
```

Vectors

The following are some other useful methods for working with vectors:

- `at(int)`: Returns the value at a specific position in the vector. For example, given `vector<int> numbers = {4, 6, 8}`,
  - `numbers.at(0)` is 4
  - `numbers.at(1)` is 6
  - `numbers.at(2)` is 8
- `push_back(value)`: Adds a new value at the end of the vector. Example: `numbers.push_back(10)` will add 10 to the end of the vector, increasing its size.
- `pop_back()`: Removes the last element of the vector, reducing its size.
- `size()`: Returns the current number of elements in the vector.
- `clear()`: Empties the vector, changing the `size()` to 0.
- `empty()`: Returns `true` if the vector contains 0 elements, `false` otherwise.

Vector: Example

```
1  #include<iostream>
2  #include <vector>
3  using namespace std;
4  int main()
5  {
6      vector<int> myVector;
7      int enteredVal, position;
8      cout << "Enter an integer ";
9      cin >> enteredVal;
10     myVector.push_back(enteredVal);
11
12     cout << "Size of the list is " << myVector.size() << endl;
13     cout << "The list: " << endl;
14     for (int i = 0; i < myVector.size(); ++i)
15         cout << " " << myVector[i] << endl;
16     cout << "Enter a position to display ";
17     cin >> position;
18     cout << "The item at position " << position << " is: ";
19     cout << myVector.at(position) << endl;
20     return 0;
21 }
```