

MIT WORLD PEACE UNIVERSITY

Object Oriented Programming with Java and C++
Second Year B. Tech, Semester 1

IMPLEMENTATION OF STL IN C++

PRACTICAL REPORT

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A2, PA 20

November 13, 2022

Contents

1 Aim and Objectives	1
2 Problem Statement	1
3 Theory	1
3.1 Concept of Standard Template Library	1
3.2 How is STL different from the C++ Standard Library?	1
3.3 Concept of Containers, Iterators and Algorithms	2
3.3.1 Containers	2
3.3.2 Iterators	3
3.3.3 Algorithms	4
4 Platform	5
5 Input	5
6 Output	5
7 Code	5
7.1 C++ Implementation of Problem A	5
7.1.1 C++ Input and Output	11
8 Conclusion	13
9 FAQs	14

1 Aim and Objectives

- To understand the use of Standard Template Library in C++
- To get familiar with list containers and iterators.

2 Problem Statement

A shop maintains the inventory of items. It stores information of items like ItemCode, ItemName, Quantity and Cost of it in a list of STL. Whenever Customer wants to buy an item, sales person inputs the ItemCode and or ItemName and the system searches in a file and displays whether it is available or not otherwise an appropriate message is displayed. If it is, then the system displays the item details and request for the quantity of items required. If the requested quantity of items are available, the total cost of items is displayed; otherwise the message is displayed as required items not in stock. After purchasing an item, system updates the list. Design a system using a class called Items with suitable data members and member functions. Implement a Menu Driven C++ program using STL concepts.

3 Theory

3.1 Concept of Standard Template Library

The Standard Template Library (STL) is a set of C++ template classes to provide common programming data structures and functions such as lists, stacks, arrays, etc. It is a library of container classes, algorithms, and iterators. It is a generalized library and so, its components are parameterized. It has a lot of useful things that we can use in our own code, without worrying to write lengthy implementations of basic data structure concepts.

STL has 4 components:

1. Algorithms : *They act on containers and provide means for various operations for the contents of the containers.*
2. Containers : *Containers or container classes store objects and data.*
3. Functions : *The STL includes classes that overload the function call operator. Instances of such classes are called function objects or functors.*
4. Iterators : *As the name suggests, iterators are used for working upon a sequence of values. They are the major feature that allows generality in STL.*

3.2 How is STL different from the C++ Standard Library?

The STL was written by Alexander Stepanov in the days long before C++ was standardised. The STL was already widely used as a library for C++, giving programmers access to containers, iterators and algorithms. When the standardisation happened, the language committee designed parts of the C++ Standard Library (which is part of the language standard) to very closely match the STL.

Over the years, many people — including prominent book authors, and various websites — have continued to refer to the C++ Standard Library as **The STL** despite the fact that the two entities are separate and that there are some differences. These differences are even more pronounced in the upcoming new C++ standard, which includes various features and significantly alters some classes.

So A lot of the functions and containers were written before the formation of various important libraries that we now use in C++. It is those libraries that are called the "STL". C++ standard libraries are ones written after it using those libraries in part.

3.3 Concept of Containers, Iterators and Algorithms

3.3.1 Containers

In C++, there are generally 3 kinds of STL containers:

- Sequential Containers : *In C++, sequential containers allow us to store elements that can be accessed in sequential order. Internally, sequential containers are implemented as arrays or linked lists data structures.*

Types of Sequential Containers

- Array
- Vector
- Deque
- List
- Forward List

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int main() {
6     // initialize a vector of int type
7     vector<int> numbers = {1, 100, 10, 70, 100};
8
9     // print the vector
10    cout << "Numbers are: ";
11    for(auto &num: numbers) {
12        cout << num << ", ";
13    }
14
15    return 0;
16 }
17
18 //Output
19 Numbers are: 1, 100, 10, 70, 100,
```

- Associative Containers: *In C++, associative containers allow us to store elements in sorted order. The order doesn't depend upon when the element is inserted. Internally, they are implemented as binary tree data structures.* Types of associative Containers.
 - Set
 - Map
 - Multiset
 - Multimaps

```
1  #include <iostream>
2  #include <set>
3  using namespace std;
4
5  int main() {
6
7      // initialize a set of int type
8      set<int> numbers = {1, 100, 10, 70, 100};
9
10     // print the set
11     cout << "Numbers are: ";
12     for(auto &num: numbers) {
13         cout << num << ", ";
14     }
15
16     return 0;
17 }
18 // Output:
19 Numbers are: 1, 10, 70, 100,
20
```

- **Unordered Associative Containers:** *In C++, STL Unordered Associative Containers provide the unsorted versions of the associative container. Internally, unordered associative containers are implemented as hash table data structures.* Types of Unordered Associated Containers

- Unordered Set
- Unordered Map
- Unordered Multiset
- Unordered Multimap

```
1  #include <iostream>
2  #include <unordered_set>
3  using namespace std;
4
5  int main() {
6
7      // initialize an unordered_set of int type
8      unordered_set<int> numbers = {1, 100, 10, 70, 100};
9
10     // print the set
11     cout << "Numbers are: ";
12     for(auto &num: numbers) {
13         cout << num << ", ";
14     }
15
16     return 0;
17 }
18 // Output
19 Numbers are: 70, 10, 100, 1,
20
```

3.3.2 Iterators

Iterators are one of the four pillars of the Standard Template Library or STL in C++. An iterator is used to point to the memory address of the STL container classes. For better understanding, you can

relate them with a pointer, to some extent. Iterators act as a bridge that connects algorithms to STL containers and allows the modifications of the data present inside the container. They allow you to iterate over the container, access and assign the values, and run different operators over them, to get the desired result.

Applications of Iterators:

1. The primary objective of an iterator is to access the STL container elements and perform certain operations on them.
2. The internal structure of a container does not matter, since the iterators provide common usage for all of them.
3. Iterator algorithms are not dependent on the container type.
4. An iterator can be used to iterate over the container elements. It can also provide access to those elements to modify their values.
5. Iterators follow a generic approach for STL container classes. This way, the programmers don't need to learn about different iterators for different containers.

Example to Demonstrate use of Iterators

```
1 #include<iostream>
2 #include<iterator> // for iterators
3 #include<vector> // for vectors
4 using namespace std;
5 int main()
6 {
7     vector<int> ar = { 1, 2, 3, 4, 5 };
8
9     // Declaring iterator to a vector
10    vector<int>::iterator ptr;
11
12    // Displaying vector elements using begin() and end()
13    cout << "The vector elements are : ";
14    for (ptr = ar.begin(); ptr < ar.end(); ptr++)
15        cout << *ptr << " ";
16
17    return 0;
18 }
19 //Output
20 The vector elements are : 1 2 3 4 5
```

3.3.3 Algorithms

STL provide different types of algorithms that can be implemented upon any of the container with the help of iterators. Thus now we don't have to define complex algorithm instead we just use the built in functions provided by the algorithm library in STL.

Algorithm functions provided by algorithm library works on the iterators, not on the containers. Thus one algorithm function can be used on any type of container. Use of algorithms from STL saves time, effort, code and are very reliable.

There are many types of algorithms already implemented reliably in the STL. Here we will use a simple Non Modifying Algorithm as an example.

```
1 #include <iostream>
2 #include <algorithm>
3 using namespace std;
4 int main ()
5 {
6     int values[] = {5,1,6,9,10,1,12,5,5,5,1,8,9,7,46};
7     int count_5 = count(values, values+15, 5);
8     cout<<"The number of times '5' appears in array= "<<count_5;
9     return 0;
10 }
11 // Output
12 The number of times 5 appears in an array= 4
```

4 Platform

Operating System: Arch Linux x86-64

IDEs or Text Editors Used: Visual Studio Code

Compilers : g++ and gcc on linux for C++

5 Input

For C++

1. Basic menu to add new elements, or purchase an item.
2. The Details of the item to add.
3. The quantity and code of the product that you wanna purchase.

6 Output

For C++

1. A list of all the elements in the Database.
2. Their codes and Quantities
3. Appropriate messages after each action is performed.

7 Code

7.1 C++ Implementation of Problem A

```
1 // Shop has item code, item name, quantity, and the cost.
2 // Input would be some item name, where we would first add things to the shop
  inventory
3 // Another option would be to check for the item in the database which is a list.
4 // If the item is found in the shop then you are supposed to ask them the number
  of items.
5 // if its available then you sell it, or else you tell them that that may items
  arent in stock.
```

```
6
7 #include <iostream>
8 #include <list>
9 #include <iomanip>
10 using namespace std;
11
12 // struct so we can put it in a single linked list.
13 struct Items
14 {
15     int item_code;
16     string item_name;
17     int item_quantity;
18     int item_cost;
19     Items(int a, string b, int q, int c) : item_code(a), item_name(b),
20     item_quantity(q), item_cost(c)
21     {
22     }
23 } currentItem(0, "", 0, 0), itemToAdd(0, "", 0, 0);
24
25 // linked list items so we can put objects of structures in it easily.
26 list<Items> items = {
27     Items(101, "Burger", 10, 200),
28     Items(102, "Fries", 10, 129),
29     Items(103, "Ice Cream", 10, 40),
30     Items(104, "Coke", 10, 50),
31 };
32
33 bool itemFound = false;
34 // returns true or false depending on whether the item was found
35 bool searchItem(int itemCode)
36 {
37     int i = 0;
38     for (list<Items>::iterator it = items.begin(); it != items.end(); it++, i++)
39     {
40         struct Items temp = *it;
41         if (temp.item_code == itemCode)
42         {
43             currentItem = temp;
44             return true;
45         }
46     }
47     return false;
48 }
49
50 // inserts items in the list
51 void insertItems()
52 {
53     cout << "Enter the details of the Item that you wanna enter to the database"
54     << endl;
55
56     i:
57     cout << "Enter the Code of the new Item: ";
58     cin >> itemToAdd.item_code;
59     if (searchItem(itemToAdd.item_code))
60     {
61         cout << "Item already exists, try another code!" << endl;
62         goto i;
63     }
64     cout << "Enter the Name of the new Item: ";
65     cin >> itemToAdd.item_name;
```



```
63
64     cout << "Enter the Quantity of the new Item: ";
65     cin >> itemToAdd.item_quantity;
66
67     cout << "Enter the Cost of the new Item: ";
68     cin >> itemToAdd.item_cost;
69
70     items.push_back(itemToAdd);
71     cout << "Item added successfully" << endl;
72 }
73
74 // just find the element at the element currentItemIndex, and replace it with the
    currentItem struct object.
75 void updateItems(int updatedItemcost, int updatedQuantity, int currentItemCode)
76 {
77     for (list<Items>::iterator it = items.begin(); it != items.end(); it++)
78     {
79         struct Items temp = *it;
80         if (temp.item_code == currentItemCode)
81         {
82             currentItem.item_quantity = updatedQuantity;
83             currentItem.item_cost = updatedItemcost;
84             items.erase(it);
85             items.push_back(currentItem);
86         }
87     }
88 }
89
90 // just displays the items in a table nicely
91 void displayItems()
92 {
93     struct Items tempItem(0, "", 0, 0);
94     cout << "The Items that you can buy are: " << endl;
95
96     cout << setw(20) << "ITEM CODE" << setw(20) << "ITEM NAME" << setw(20) << "
ITEM QUANTITY" << setw(20) << "ITEM COST" << endl;
97     for (list<Items>::iterator it = items.begin(); it != items.end(); it++)
98     {
99         tempItem = *it;
100         cout << setw(20) << tempItem.item_code << setw(20) << tempItem.item_name
<< setw(20) << tempItem.item_quantity << setw(20) << tempItem.item_cost << endl
;
101     }
102 }
103
104 // display the things, and check if the selected item code by the user exists in
    the thing by calling searchItem. Then input the item quantity, check for it,
    and update the currentItem object.
105 // then call the updateItem function.
106 bool PurchaseItems()
107 {
108     int selectedItemCode = -1;
109     int selectedQuantity = 0;
110 c:
111     try
112     {
113         cout << "Please enter the code of the item that you wanna buy" << endl;
114         cin >> selectedItemCode;
115         if (!searchItem(selectedItemCode))
```

```
116     {
117         throw selectedItemCode;
118     }
119     l:
120     cout << "Enter the Quantity of the Item that you wanna buy. "
121         << endl;
122     cout << "Max quantity is: " << currentItem.item_quantity << endl;
123     cin >> selectedQuantity;
124     try
125     {
126         if (selectedQuantity > currentItem.item_quantity || selectedQuantity
127         <= 0)
128         {
129             throw selectedQuantity;
130         }
131         else
132         {
133             cout << "That will be: " << currentItem.item_cost *
134             selectedQuantity << " Rupees" << endl;
135             cout << "Thank you for Purchasing!" << endl;
136             updateItems(currentItem.item_cost, currentItem.item_quantity -
137             selectedQuantity, selectedItemCode);
138         }
139     }
140     catch (int something)
141     {
142         cout << "Quantity you entered is not sensible! Try again!" << endl;
143         goto l;
144     }
145     return true;
146 }
147 catch (int something)
148 {
149     cout << "The code of the item that you entered doesnt exist. Try again. "
150     << endl;
151     goto c;
152 }
153 return true;
154 }
155 // function to change the price of the item.
156 bool changePrice()
157 {
158     int selectedItemCode = -1;
159     int updatedCost = 0;
160 c:
161     try
162     {
163         cout << "Please enter the code of the item that you wanna Change the Price
164         of" << endl;
165         cin >> selectedItemCode;
166         if (!searchItem(selectedItemCode))
167         {
168             throw selectedItemCode;
169         }
170     }
171     l:
172     cout << "Enter the Updated Price of the Item."
173         << endl;
174     cout << "Current Price is: " << currentItem.item_cost << endl;
```

```
170     cin >> updatedCost;
171     try
172     {
173         if (updatedCost <= 0)
174         {
175             throw updatedCost;
176         }
177         else
178         {
179             cout << "Done!" << endl;
180             updateItems(updatedCost, currentItem.item_quantity,
selectedItemCode);
181         }
182     }
183     catch (int something)
184     {
185         cout << "Cost you entered is not sensible! Try again!" << endl;
186         goto l;
187     }
188     return true;
189 }
190 catch (int something)
191 {
192     cout << "The code of the item that you entered doesnt exist. Try again. "
<< endl;
193     goto c;
194 }
195 return true;
196 }
197
198 // function to change the quantity of the item.
199 bool changeQuantity()
200 {
201     int selectedItemCode = -1;
202     int updatedQuantity = 0;
203 c:
204     try
205     {
206         cout << "Please enter the code of the item that you wanna update" << endl;
207         cin >> selectedItemCode;
208         if (!searchItem(selectedItemCode))
209         {
210             throw selectedItemCode;
211         }
212     l:
213         cout << "Enter the updated Quantity"
<< endl;
214         cout << "Current quantity is: " << currentItem.item_quantity << endl;
215         cin >> updatedQuantity;
216         try
217         {
218             if (updatedQuantity <= 0)
219             {
220                 throw updatedQuantity;
221             }
222             else
223             {
224                 cout << "Done!" << endl;
225                 updateItems(currentItem.item_cost, updatedQuantity,
```

```
selectedItemCode);
227     }
228 }
229 catch (int something)
230 {
231     cout << "Quantity you entered is not sensible! Try again!" << endl;
232     goto l;
233 }
234 return true;
235 }
236 catch (int something)
237 {
238     cout << "The code of the item that you entered doesnt exist. Try again. "
239 << endl;
240     goto c;
241 }
242 return true;
243 }
244 int main()
245 {
246     int choice = 0;
247     struct Items;
248     cout << "Welcome to McRonaldds" << endl;
249
250     do
251     {
252         cout << endl
253             << "What do you wanna do?\n\
254 1. Add new Items\n\
255 2. Purchase Item\n\
256 3. Change Price of Item\n\
257 4. Change Quantity of Item\n\
258 5. View Items\n\
259 6. Quit\n"
260             << endl;
261         cin >> choice;
262         switch (choice)
263         {
264             case 1:
265                 insertItems();
266                 break;
267             case 2:
268                 displayItems();
269                 PurchaseItems();
270                 break;
271             case 3:
272                 displayItems();
273                 changePrice();
274                 break;
275             case 4:
276                 displayItems();
277                 changeQuantity();
278                 break;
279             case 5:
280                 displayItems();
281                 break;
282             case 6:
283                 cout << "Thanks for Visiting our store!" << endl;
```

```
284         break;
285     default:
286         cout << "Sorry, we cant do that in this store" << endl;
287         break;
288     }
289     itemFound = false;
290 } while (choice != 6);
291 return 0;
292 }
```

Listing 1: Main.Cpp

7.1.1 C++ Input and Output

```
1 Welcome to McDonalds
2
3 What do you wanna do?
4     1. Add new Items
5     2. Purchase Item
6     3. Change Price of Item
7     4. Change Quantity of Item
8     5. View Items
9     6. Quit
10
11 5
12 The Items that you can buy are:
13     ITEM CODE      ITEM NAME      ITEM QUANTITY      ITEM COST
14         101          Burger           10             200
15         102          Fries            10             129
16         103        Ice Cream           10              40
17         104           Coke            10              50
18
19 What do you wanna do?
20     1. Add new Items
21     2. Purchase Item
22     3. Change Price of Item
23     4. Change Quantity of Item
24     5. View Items
25     6. Quit
26
27 1
28 Enter the details of the Item that you wanna enter to the database
29 Enter the Code of the new Item: 101
30 Item already exists, try another code!
31 Enter the Code of the new Item: 108
32 Enter the Name of the new Item: oreo
33 Enter the Quantity of the new Item: 50
34 Enter the Cost of the new Item: 40
35 Item added successfully
36
37 What do you wanna do?
38     1. Add new Items
39     2. Purchase Item
40     3. Change Price of Item
41     4. Change Quantity of Item
42     5. View Items
43     6. Quit
44
45 5
```

OOPJC Assignment 5

```
46 The Items that you can buy are:
47     ITEM CODE      ITEM NAME      ITEM QUANTITY      ITEM COST
48         101          Burger          10             200
49         102          Fries           10             129
50         103        Ice Cream          10              40
51         104           Coke           10              50
52         108          oreo            50              40
53
54 What do you wanna do?
55     1. Add new Items
56     2. Purchase Item
57     3. Change Price of Item
58     4. Change Quantity of Item
59     5. View Items
60     6. Quit
61
62 2
63 The Items that you can buy are:
64     ITEM CODE      ITEM NAME      ITEM QUANTITY      ITEM COST
65         101          Burger          10             200
66         102          Fries           10             129
67         103        Ice Cream          10              40
68         104           Coke           10              50
69         108          oreo            50              40
70 Please enter the code of the item that you wanna buy
71 108
72 Enter the Quantity of the Item that you wanna buy.
73 Max quantity is: 50
74 55
75 Quantity you entered is not sensible! Try again!
76 Enter the Quantity of the Item that you wanna buy.
77 Max quantity is: 50
78 4
79 That will be: 160 Rupees
80 Thank you for Purchasing!
81
82 What do you wanna do?
83     1. Add new Items
84     2. Purchase Item
85     3. Change Price of Item
86     4. Change Quantity of Item
87     5. View Items
88     6. Quit
89
90 3
91 The Items that you can buy are:
92     ITEM CODE      ITEM NAME      ITEM QUANTITY      ITEM COST
93         101          Burger          10             200
94         102          Fries           10             129
95         103        Ice Cream          10              40
96         104           Coke           10              50
97         108          oreo            46              40
98 Please enter the code of the item that you wanna Change the Price of
99 104
100 Enter the Updated Price of the Item.
101 Current Price is: 50
102 45
103 Done!
104
```

```
105 What do you wanna do?
106     1. Add new Items
107     2. Purchase Item
108     3. Change Price of Item
109     4. Change Quantity of Item
110     5. View Items
111     6. Quit
112
113 4
114 The Items that you can buy are:
115     ITEM CODE    ITEM NAME    ITEM QUANTITY    ITEM COST
116         101         Burger         10         200
117         102         Fries          10        129
118         103       Ice Cream         10         40
119         108           oreo         46         40
120         104          Coke          10         45
121 Please enter the code of the item that you wanna update
122 108
123 Enter the updated Quantity
124 Current quantity is: 46
125 50
126 Done!
127
128 What do you wanna do?
129     1. Add new Items
130     2. Purchase Item
131     3. Change Price of Item
132     4. Change Quantity of Item
133     5. View Items
134     6. Quit
135
136 5
137 The Items that you can buy are:
138     ITEM CODE    ITEM NAME    ITEM QUANTITY    ITEM COST
139         101         Burger         10         200
140         102         Fries          10        129
141         103       Ice Cream         10         40
142         104          Coke          10         45
143         108           oreo         50         40
144
145 What do you wanna do?
146     1. Add new Items
147     2. Purchase Item
148     3. Change Price of Item
149     4. Change Quantity of Item
150     5. View Items
151     6. Quit
152
153 6
154 Thanks for Visiting our store!
```

Listing 2: Output for Problem 1

8 Conclusion

Thus, the purpose of the STL libraries in C++ was understood, and implemented successfully. Containers like lists and iterators for them were also used and understood.

9 FAQs

1. What are class templates? How are they created? What is the need for class templates?

Templates are powerful features of C++ which allows us to write generic programs. There are two ways we can implement templates:

- Function Templates
- Class Templates

Similar to function templates, we can use class templates to create a single class to work with different data types. There are few, but important reasons to use class templates:

- Class templates come in handy as they can make our code shorter and more manageable.
- If you have various functions that you wanna implement on a set of data, but you aren't sure which data type will be given as input, then you can use class templates.
- Example: If you are creating a calculator, a class template to include basic functions like addition, subtraction etc would be perfect as you can get an integer or a floating point value as your input for your calculator.

2. Create a template for bubble sort functions.

```
1 template <class T>
2 bubbleSort(T arr[], int n)
3 {
4     int i, j;
5     for (i = 0; i < n - 1; i++)
6         for (j = 0; j < n - i - 1; j++)
7             if (arr[j] > arr[j + 1])
8                 swap(arr[j], arr[j + 1]);
9 }
10
11
```

3. Explain with example, how Function Templates are implemented?

```
1 #include <iostream>
2 using namespace std;
3 template<class T> T add(T &a, T &b)
4 {
5     T result = a+b;
6     return result;
7 }
8 int main()
9 {
10     int i =2;
11     int j =3;
12     float m = 2.3;
13     float n = 1.2;
14     cout<<"Addition of i and j is :"<<add(i,j);
15     cout<<'\\n';
16     cout<<"Addition of m and n is :"<<add(m,n);
17     return 0;

```



```
18 }  
19
```

4. Explain with example how can a class template be created.

```
1 // Class template  
2 template <class T>  
3 class Number {  
4     private:  
5         // Variable of type Tf  
6         T num;  
7  
8     public:  
9         Number(T n) : num(n) {}    // constructor  
10  
11     T getNum() {  
12         return num;  
13     }  
14 };  
15  
16 int main() {  
17  
18     // create object with int type  
19     Number<int> numberInt(7);  
20  
21     // create object with double type  
22     Number<double> numberDouble(7.7);  
23  
24     cout << "int Number = " << numberInt.getNum() << endl;  
25     cout << "double Number = " << numberDouble.getNum() << endl;  
26  
27     return 0;  
28 }  
29 // Output  
30 int Number = 7  
31 double Number = 7.7  
32
```

5. Explain Generic functions and Generic class.

- Generic functions use the concept of a function template. Generic functions define a set of operations that can be applied to the various types of data.
- The type of the data that the function will operate on depends on the type of the data passed as a parameter.
- For example, Quick sorting algorithm is implemented using a generic function, it can be implemented to an array of integers or array of floats.
- A Generic function is created by using the keyword template. The template defines what function will do.
- Just like a class is a collection of a bunch of functions, that can be inherited and instantiated at once, generic functions are similar in that manner, except in a generic class, you could use a generic data type, and this would be applicable and useful for implementing each function in the Class. So each function in the class can be called and can manipulate variables of the generic data type for which the class is defined.