**Experiment No- 12 Date-**

**Aim – To study standard template library (STL)**

# Theory –

# Standard Template Library (STL)

The Standard Template Library (STL) in C++ is a powerful set of classes and functions designed for common data structures and algorithms. It provides flexible and efficient ways to manage and manipulate collections of data. STL includes components like containers, iterators, algorithms, and function objects, making it easier to write generic, reusable, and efficient code.

**Key Components of STL**

1. **Containers**: Collections of data that can hold multiple elements, such as arrays, linked lists, stacks, queues, and maps.
2. **Algorithms**: Functions that perform operations on data stored in containers, such as sorting, searching, counting, and transforming.
3. **Iterators**: Objects that provide a way to access elements in a container sequentially without exposing the underlying structure.
4. **Function Objects (Functors)**: Objects that can be used like functions and passed to algorithms to customize their behavior.

**Common STL Header Files**

Here is a list of commonly used STL header files and a brief description of the features they offer:

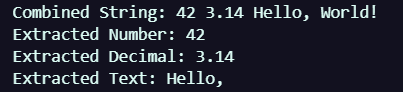
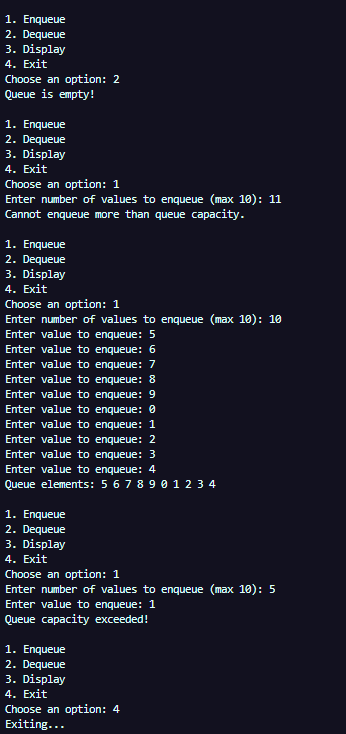
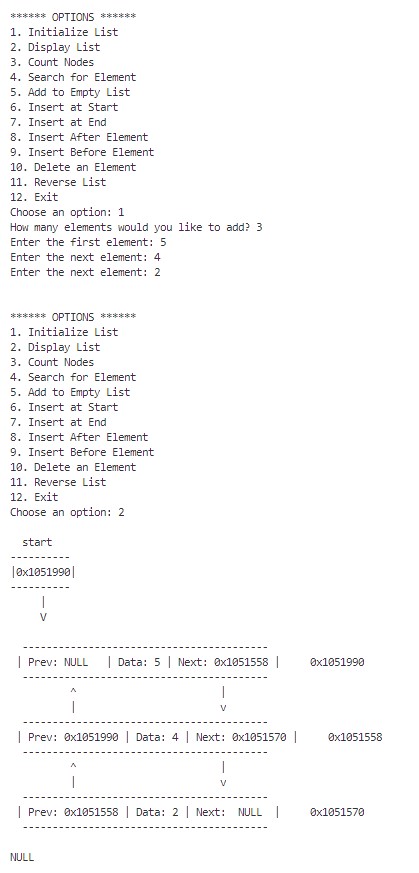
| **Header File** | **Description** |
| --- | --- |
| <vector> | Provides the vector container, a dynamic array that can resize automatically. |
| <list> | Implements a doubly linked list (list) for efficient insertion and deletion. |
| <deque> | Implements a double-ended queue (deque) that allows insertion and deletion at both ends. |
| <array> | Provides the array container, a fixed-size array with more functionality than a C array. |
| <stack> | Implements the stack container adapter for LIFO operations. |
| <queue> | Implements the queue container adapter for FIFO operations. |
| <priority\_queue> | Implements a priority\_queue, a queue where elements are arranged by priority. |
| <map> | Provides the map associative container, which stores key-value pairs sorted by keys. |
| <unordered\_map> | Implements unordered\_map, an associative container that stores key-value pairs with hash tables for fast access. |
| <set> | Provides the set associative container, which stores unique elements in sorted order. |
| <unordered\_set> | Provides unordered\_set, an associative container that stores unique elements in no particular order, using hash tables. |
| <algorithm> | Defines various algorithms such as sort, find, count, and accumulate. |
| <iterator> | Provides iterators, iterator traits, and iterator functions for working with STL containers. |
| <numeric> | Contains numeric algorithms such as accumulate, adjacent\_difference, and inner\_product. |
| <functional> | Contains function objects (functors) and other utilities for functional programming. |
| <utility> | Provides utility functions like pair, make\_pair, and other helper classes. |
| <memory> | Contains memory management utilities like unique\_ptr, shared\_ptr, and weak\_ptr. |
| <string> | Defines the string class for handling sequences of characters and string operations. |

[A] Write a C++ program to implement standard library vector sequence container

|  |  |
| --- | --- |
| **Program-**  #include <iostream>  #include <vector> *// Include the vector library*  using namespace std;  int *main*() {  *// Create a vector to store integer values*      vector<int> numbers;  *// Add elements to the vector using push\_back*      numbers.*push\_back*(10);      numbers.*push\_back*(20);      numbers.*push\_back*(30);      numbers.*push\_back*(40);    *// Display initial vector elements*      cout *<<* "Vector elements after push\_back operations: ";      for (int i = 0; i < numbers.*size*(); i++) {          cout *<<* numbers*[*i*]* *<<* " ";      }      cout *<<* *endl*;  *// Insert an element at a specific position*      numbers.*insert*(numbers.*begin*() *+* 2, 25); *// Insert 25 at index 2*  *// Display vector elements after insertion*      cout *<<* "Vector elements after insertion at index 2: ";      for (int i = 0; i < numbers.*size*(); i++) {          cout *<<* numbers*[*i*]* *<<* " ";      }      cout *<<* *endl*;  *// Access and modify an element*      numbers*[*3*]* = 35; *// Modify the element at index 3*  *// Display vector elements after modification*      cout *<<* "Vector elements after modifying index 3: ";      for (int i = 0; i < numbers.*size*(); i++) {          cout *<<* numbers*[*i*]* *<<* " ";      }      cout *<<* *endl*;  *// Remove the last element using pop\_back*      numbers.*pop\_back*();  *// Display vector elements after removing the last element*      cout *<<* "Vector elements after pop\_back: ";      for (int i = 0; i < numbers.*size*(); i++) {          cout *<<* numbers*[*i*]* *<<* " ";      }      cout *<<* *endl*;  *// Display the size and capacity of the vector* | OUTPUT –        cout *<<* "Current size of the vector: " *<<* numbers.*size*() *<<* *endl*;      cout *<<* "Current capacity of the vector: " *<<* numbers.*capacity*() *<<* *endl*;  *// Use range-based for loop to print vector elements*      cout *<<* "Vector elements using range-based for loop: ";      for (int num : numbers) {          cout *<<* num *<<* " ";      }      cout *<<* *endl*;  *// Clear all elements in the vector*      numbers.*clear*();      cout *<<* "Vector size after clear: " *<<* numbers.*size*() *<<* *endl*;      return 0;  } |

****[ B] Write a C++ program to implement standard library list sequence container

|  |  |
| --- | --- |
| **Program –**    **#include <iostream>**  **#include <list> *// Include the list library***  **using namespace std;**  **int *main*() {**  ***// Create a list to store integer values***  **list<int> numbers;**  ***// Add elements to the list using push\_back and push\_front***  **numbers.*push\_back*(10); *// Add 10 at the end***  **numbers.*push\_back*(20); *// Add 20 at the end***  **numbers.*push\_front*(5); *// Add 5 at the beginning***  **numbers.*push\_back*(30); *// Add 30 at the end***    ***// Display initial list elements***  **cout *<<* "List elements after push operations: ";**  **for (int num : numbers) {**  **cout *<<* num *<<* " ";**  **}**  **cout *<<* *endl*;**  ***// Insert an element at a specific position***  **auto it = numbers.*begin*();**  ***advance*(it, 2); *// Move iterator to the third position***  **numbers.*insert*(it, 15); *// Insert 15 at the third position***  ***// Display list elements after insertion***  **cout *<<* "List elements after insertion at position 3: ";**  **for (int num : numbers) {**  **cout *<<* num *<<* " ";**  **}**  **cout *<<* *endl*;**  ***// Remove an element from the list***  **numbers.*remove*(20); *// Removes all occurrences of 20***  ***// Display list elements after removal***  **cout *<<* "List elements after removing 20: ";**  **for (int num : numbers) {**  **cout *<<* num *<<* " ";**  **}**  **cout *<<* *endl*;**  ***// Accessing and modifying the first and last elements***  **numbers.*front*() = 1; *// Modify the first element***  **numbers.*back*() = 25; *// Modify the last element***  ***// Display list elements after modification***  **cout *<<* "List elements after modifying front and back: ";**  **for (int num : numbers) {**  **cout *<<* num *<<* " ";**  **}**  **cout *<<* *endl*;**  ***// Pop elements from front and back***  **numbers.*pop\_front*(); *// Remove the first element***  **numbers.*pop\_back*(); *// Remove the last element***  ***// Display list elements after popping front and back***  **cout *<<* "List elements after popping front and back: ";**  **for (int num : numbers) {**  **cout *<<* num *<<* " ";**  **}**  **cout *<<* *endl*;**  ***// Clear all elements in the list***  **numbers.*clear*();**  **cout *<<* "List size after clear: " *<<* numbers.*size*() *<<* *endl*;**  **return 0;**  **}** | **Output –** |

**** **[C] Write a C++ program to implement standard library deque sequence container**

|  |  |
| --- | --- |
| **Program –**  **#include <iostream>**  **#include <deque> *// Include the deque library***  **using namespace std;**  **int *main*() {**  ***// Create a deque to store integer values***  **deque<int> numbers;**  ***// Add elements to the deque using push\_back and push\_front***  **numbers.*push\_back*(10); *// Add 10 at the end***  **numbers.*push\_back*(20); *// Add 20 at the end***  **numbers.*push\_front*(5); *// Add 5 at the beginning***  **numbers.*push\_back*(30); *// Add 30 at the end***  ***// Display initial deque elements***  **cout *<<* "Deque elements after push operations: ";**  **for (int num : numbers) {**  **cout *<<* num *<<* " ";**  **}**  **cout *<<* *endl*;**  ***// Insert an element at a specific position***  **auto it = numbers.*begin*() *+* 2; *// Iterator pointing to the third position***  **numbers.*insert*(it, 15); *// Insert 15 at the third position***  ***// Display deque elements after insertion***  **cout *<<* "Deque elements after insertion at position 3: ";**  **for (int num : numbers) {**  **cout *<<* num *<<* " ";**  **}**  **cout *<<* *endl*;**  ***// Remove elements from the front and back***  **numbers.*pop\_front*(); *// Remove the first element***  **numbers.*pop\_back*(); *// Remove the last element***  ***// Display deque elements after popping front and back***  **cout *<<* "Deque elements after popping front and back: ";**  **for (int num : numbers) {**  **cout *<<* num *<<* " ";**  **}**  **cout *<<* *endl*;**  ***// Accessing and modifying the first and last elements***  **numbers.*front*() = 1; *// Modify the first element***  **numbers.*back*() = 25; *// Modify the last element***  ***// Display deque elements after modification***  **cout *<<* "Deque elements after modifying front and back: ";**  **for (int num : numbers) {**  **cout *<<* num *<<* " ";**  **}**  **cout *<<* *endl*;**  ***// Clear all elements in the deque***  **numbers.*clear*();**  **cout *<<* "Deque size after clear: " *<<* numbers.*size*() *<<* *endl*;**  **return 0;**  **}** | **Output –** |

[D] Write a C++ program to implement standard library stack adapter class

|  |  |
| --- | --- |
| **Program –**  #include <iostream>  #include <stack> *// Include the stack library*  using namespace std;  int *main*() {  *// Create a stack to store integer values*      stack<int> myStack;  *// Push elements onto the stack*      myStack.*push*(10); *// Push 10*      myStack.*push*(20); *// Push 20*      myStack.*push*(30); *// Push 30*  *// Display the top element*      cout *<<* "Top element: " *<<* myStack.*top*() *<<* *endl*;  *// Pop elements from the stack*      cout *<<* "Popping elements from the stack: ";      while (!myStack.*empty*()) {          cout *<<* myStack.*top*() *<<* " "; *// Access the top element*          myStack.*pop*(); *// Remove the top element*      }      cout *<<* *endl*;  *// Check if the stack is empty*      if (myStack.*empty*()) {          cout *<<* "The stack is now empty." *<<* *endl*;      }      return 0;  } | **Output –** |

**[E] Write a C++ program to implement standard library queue adapter class template**

|  |  |
| --- | --- |
| **Program –**  #include <iostream>  #include <queue> *// Include the queue library*  using namespace std;  int *main*() {  *// Create a queue to store integer values*      queue<int> myQueue;  *// Enqueue elements onto the queue*      myQueue.*push*(10); *// Add 10 to the back*      myQueue.*push*(20); *// Add 20 to the back*      myQueue.*push*(30); *// Add 30 to the back*  *// Display the front and back elements*      cout *<<* "Front element: " *<<* myQueue.*front*() *<<* *endl*;      cout *<<* "Back element: " *<<* myQueue.*back*() *<<* *endl*;  *// Dequeue elements from the queue*      cout *<<* "Dequeuing elements from the queue: ";      while (!myQueue.*empty*()) {          cout *<<* myQueue.*front*() *<<* " "; *// Access the front element*          myQueue.*pop*(); *// Remove the front element*      }      cout *<<* *endl*;  *// Check if the queue is empty*      if (myQueue.*empty*()) {          cout *<<* "The queue is now empty." *<<* *endl*;      }      return 0;  } | **Output –** |

**[F] Write a C++ program to implement standard library list sequence container and do bidirectional iteration and sorting**

|  |  |
| --- | --- |
| **Program –**  #include <iostream>  #include <list> *// list datatype*  #include <algorithm>  using namespace std;  *//itarators are used to iterate(list, vector, map, set, etc) through the elements of the container*  void *iterateForward*(const list<int>& lst) {      cout *<<* "Forward iteration: ";      for (const int& val : lst) {          cout *<<* val *<<* " ";      } *// for each loop is used to iterate through the elements of the list*      cout *<<* *endl*;  }  void *iterateBackward*(const list<int>& lst) {      cout *<<* "Backward iteration: ";      for (auto it = lst.*rbegin*(); it *!=* lst.*rend*(); *++*it) { *// rbegin() and rend() are used to iterate in reverse order*          cout *<<* *\**it *<<* " "; *// \*it is used to access the value at the iterator*      }      cout *<<* *endl*;  }  void *sortList*(list<int>& lst) {      lst.*sort*(); *// .sort() function sorts the list in ascending order*  }  void *displayMenu*() {      cout *<<* "Menu:" *<<* *endl*;      cout *<<* "1. Display list forward" *<<* *endl*;      cout *<<* "2. Display list backward" *<<* *endl*;      cout *<<* "3. Sort list" *<<* *endl*;      cout *<<* "4. Exit" *<<* *endl*;      cout *<<* "Enter your choice: ";  }  int *main*() {      list<int> myList; *// list datatype*      int choice, value;      cout *<<* "Enter elements of the list (enter -1 to stop): ";      while (cin *>>* value && value != -1) {          myList.*push\_back*(value); *// .push\_back does the role of enqueue by adding elements to the end of the list*      }      do {  *displayMenu*();          cin *>>* choice;          switch (choice) {              case 1:  *iterateForward*(myList);                  break;              case 2:  *iterateBackward*(myList);                  break;              case 3:  *sortList*(myList);                  cout *<<* "List sorted." *<<* *endl*;                  break;              case 4:                  cout *<<* "Exiting..." *<<* *endl*;                  break;              default:                  cout *<<* "Invalid choice. Please try again." *<<* *endl*;          }      } while (choice != 4);      return 0;  } | **Output –** |

**Conclusion – All the codes were successfully executed using the concepts of *Standard Tem*plate Library .**