Standard Template Library (STL)

Standard Template Library is a software library for the C++ programming language that influenced several parts of the C++ Standard Library. It provides four components called algorithms, containers, functions, and iterators. Note that the term "STL" or "Standard Template Library" does not show up anywhere in the ISO 14882 C++ standard. So referring to the C++ standard library as STL is wrong, ie, STL and C++ Standard Library are 2 different things with the former being the subset of the latter.

1. Containers
2. Iterators
3. Algorithms

Containers

The STL contains sequence containers and associative containers. The Containers are objects that store data. The standard sequence containers include vector, deque, and list. The standard associative containers are set, multiset, map, multimap, hash_set, hash_map, hash_multiset, and hash_multimap. There are also container adaptors queue, priority_queue, and stack, that are containers with the specific interface, using other containers as implementation.

Iterators

An iterator is an object that enables a programmer to traverse a container. The STL implements five different types of iterators: input (used to read a sequence of values), output (used to write a sequence of values), forward (that can be read, written to, and move forward), bidirectional (like forward iterators, but can also move backwards) and random access (move freely any number of steps in one operation). Iterators are the major feature that allows the generality of the STL.

Algorithms

Algorithms in STL is a collection of functions specially designed to be used on ranges of elements. A range is any sequence of objects that can be accessed through iterators or pointers, such as an array or an instance of some of the STL containers. Examples of algorithms in STL: sort (Sort elements in range), binary_search (Test if a value exists in sorted sequence), min_element (Return smallest element in range), etc. Note that all these algorithms can be applied to any data type accepted as a template. More functions can be found at link: `https://en.cppreference.com/w/cpp/algorithm`

Lab exercises

Exercise 1

You are given an array of integers within range -10^9 to 10^9. Your task is to write a function that sorts the array in non-decreasing order. The input array can contain both positive and negative integers. Apply the following operations on given array:

```
sort: Sorts the elements in the array in non-decreasing order.
reverse: Reverses the order of the elements in the array.
rotate: Rotates the order of the elements in the array.
min_element: Finds the minimum element in the array.
max_element: Finds the maximum element in the array.
count: Counts the number of occurrences of a specific value in the array.
accumulate: Calculates the sum of all elements in the array.
```

Exercise 2

You are given a vector of strings. Your task is to write a function that sorts the vector of strings in lexicographical order.

```
Input: ["banana", "apple", "orange", "grape"]

Output: ["apple", "banana", "grape", "orange"]
```

Apply the following operations on given array:

```
sort: Sorts the elements in the array in non-decreasing order.
reverse: Reverses the order of the elements in the array.
rotate: Rotates the order of the elements in the array.
min_element: Finds the minimum element in the array.
max_element: Finds the maximum element in the array.
count: Counts the number of occurrences of a specific value in the array.
accumulate: Calculates the sum of all elements in the array.
```

Exercise 3

Write a program that simulates a shopping cart using a vector container. The program should allow users to add items to the cart, remove items from the cart, and view the items currently in the cart.

- Create a class called ShoppingCart that contains a vector of strings to store the items in the cart. Implement the following member functions for the ShoppingCart class:
- void addItem(const std::string& item): Adds the specified item to the cart.
- void removeItem(const std::string& item): Removes the specified item from the cart.
- void displayCart() const: Displays the items currently in the cart.
- The program should provide a menu-driven interface to allow users to interact with the shopping cart (e.g., add items, remove items, display cart).
- The program should continue running until the user chooses to exit.

```
Welcome to the Shopping Cart program!

1. Add item to cart
```

```
2. Remove item from cart
3. View cart
4. Exit

Enter your choice: 1
Enter item to add: Apple
Apple added to cart.

1. Add item to cart
2. Remove item from cart
3. View cart
4. Exit

Enter your choice: 1
Enter item to add: Banana
Banana added to cart.

1. Add item to cart
2. Remove item from cart
3. View cart
4. Exit

Enter your choice: 3
Items in cart:
1. Apple
2. Banana

1. Add item to cart
2. Remove item from cart
3. View cart
4. Exit

Enter your choice: 2
Enter item to remove: Apple
Apple removed from cart.

1. Add item to cart
2. Remove item from cart
3. View cart
4. Exit

Enter your choice: 3
Items in cart:
1. Banana

1. Add item to cart
2. Remove item from cart
3. View cart
4. Exit

Enter your choice: 4
Goodbye!
```

Exercise 4

You are given a list of student names and their corresponding scores in a contest. Your task is to store this information in a map container, where the keys are the student names (strings) and the values are their scores (integers). Additionally, you need to implement functions to add a new student to the map, remove a student from the map, and display the scores of all students in ascending order of their names.

- Use a std::map or std::unordered_map to store the student names and scores.
- Implement the following member functions for the StudentScores class:
  - void addStudent(const std::string& name, int score): Adds a new student with the given name and score to the map.
  - void removeStudent(const std::string& name): Removes the student with the given name from the map.
  - void displayScores() const: Displays the scores of all
  - students in ascending order of their names.
- Assume that each student name is unique.

```
Student Scores:

Add student:
Enter student name: Ali
Enter student score: 85
Student added successfully.

Add student:
Enter student name: Babar
Enter student score: 92
Student added successfully.

Add student:
Enter student name: Raza
Enter student score: 78
Student added successfully.

Student Scores:
Ali: 85
Babar: 92
Charlie: 78

Remove student:
Enter student name to remove: Babar
Student removed successfully.

Student Scores:
Ali: 85
Charlie: 78
```

Exercise 5

You are given a list of integers. Your task is to store these integers in a set container, where duplicates

are not allowed. Additionally, you need to implement functions to add a new integer to the set, remove an integer from the set, and display all integers in the set in ascending order.

- Use a std::set or std::unordered_set to store the integers.
- Implement the following member functions for the IntegerSet class:
  -
  - void addInteger(int num): Adds a new integer to the set.
  - void removeInteger(int num): Removes the integer from the set.
  - void displayIntegers() const: Displays all integers in the set in ascending order.

  Assume that each integer appears at most once in the input list.

****