№ 9-10 Стандартная библиотека шаблонов (STL). Алгоритмы

Создать три проекта демонстрации: использования контейнерных классов для хранения встроенных типов данных; использования контейнерных классов для хранения пользовательских типов данных; использования алгоритмов STL.

<u>В проекте № 1</u> выполнить следующее:

1. Создать объект-контейнер в соответствии с вариантом задания и заполнить его данными, тип которых определяется вариантом задания. Просмотреть контейнер. (*Дополнительно: также создать конструктор типа initializer_list***)**

2. Изменить контейнер, удалив из него одни элементы и заменить их на другие. (Почему нельзя удалять элементы в цикле через итераторы на изменяемый обьект?)

3. Просмотреть контейнер, используя для доступа к элементам итераторы. Использовать revers и const итераторы.

4. Создать второй контейнер (тип определить по варианту) и заполнить его данными того же типа, что и первый контейнер.

5. Изменить первый контейнер, удалив из него n элементов после заданного и добавив затем в него все элементы из второго контейнера. Просмотреть первый и второй контейнеры.

<u>В проекте № 2</u> выполнить то же самое, но для данных пользовательского типа. Причём должны использоваться умные указатели (исходный контейнер — unique_ptr, принимающий — shared_ptr). В качестве пользовательского типа данных использовать пользовательский класс практикума № 3 или 6. Для вставки и удаления элементов контейнера использовать соответствующие операции, определенные в классе контейнера. Для ввода-вывода объектов пользовательского класса следует перегрузить операции >> и <<. (*Дополнительно: продемонстрировать работу с weak_ptr и объяснить его назначение. Объяснить принципы работы всех типов умных указателей***)**

<u>В проекте № 3</u> выполнить следующее:

1. Создать контейнер, содержащий объекты пользовательского типа. Тип контейнера выбирается в соответствии с вариантом задания – последний столбец (третий контейнер). Отсортировать его по убыванию элементов. Если алгоритмы не поддерживают заданный тип контейнера, написать свой алгоритм. Просмотреть контейнер. (*Дополнительно: использовать 3 разных алгоритма сортировки из STL***)**

2. Используя подходящий алгоритм, найти в контейнере элемент, удовлетворяющий заданному условию. Использовать поиск с конца. (*Дополнительно: используйте алгоритмы lower_bound и upper_bound***)**

3. Переместить элементы, удовлетворяющие заданному условию в другой (предварительно пустой) контейнер. Тип второго контейнера определяется вариантом задания (при перемещении элементов ассоциативного контейнера в не ассоциативный перемещаются только данные, ключи не перемещаются и наоборот, при перемещении элементов не ассоциативного контейнера в ассоциативный должен быть сформирован ключ). Просмотреть второй контейнер с конца через итераторы.

4. Отсортировать первый и второй контейнеры по возрастанию элементов. Просмотреть их. Найти k-тую порядковую статистику.

5. Получить третий контейнер путем слияния первых двух. Просмотреть третий контейнер.

6. Подсчитать, сколько элементов, удовлетворяющих заданному условию, содержит третий контейнер. Определить, есть ли в третьем контейнере элемент, удовлетворяющий заданному условию. (*Дополнительно: используйте лямбды; используйте на контейнерах различные алгоритмы для работы с множествами(не менее двух)***)**

7. Продемонстрировать работу со std::string — использовать не менее 3 разных конструкторов, а также конструктор копирования, перемещения. Использовать не менее трех специализированных методов, которые есть у string(вроде substr, split)/

8. Работа с функциональными объектами. Написать свой функтор, который принимает стандартный (из STL) функциональный объект. Применить внутри функтора библиотечный функциональный объект. Написать второй функциональный объект с отличным от первого кол-вом параметров. Использовать как минимум 2 различных биндера для их связки. Продемонстрировать их использование.

№	Первый контейнер	Второй контейнер	Встроенный тип данных	Третий контейнер
1	stack	set	char	list
2	vector	list	int	map
3	list	deque	long	vector
4	deque	stack	float	multiset
5	array	queue	double	list
6	priority_queue	vector	char	stack
7	vector	stack	string	map
8	map	forward_list	long	list
9	multimap	deque	float	list
10	unordered_set	stack	int	multiset
11	multiset	queue	char	multiset

12	vector	unordered_map	double	set
13	list	set	int	map
14	deque	multiset	long	set
15	vector	set	long	map
16	queue	unordered_map	double	list
17	forward_list	queue	double	set
18	map	stack	char	set
19	queue	set	int	
20	map	list	int	queue
21	unordered_multimap	vector	int	list
22	unordered_multiset	vector	long double	list
23	multimap	unordered_set	float	vector
24	multiset	queue	char	vector
25	vector	set	bool	list
26	bitset	map	bool	list