Лабораторная работа №6.

Универсальные контейнеры и фундаментальные алгоритмы библиотеки STL.

Цель работы: изучение средств стандартной библиотеки шаблонов STL для поддержки динамических структур данных и фундаментальных алгоритмов.

Теоретические сведения

Алгоритмы STL предназначены для работы с контейнерами и другими последовательностями. Каждый алгоритм реализован в виде шаблона или набора шаблонов функции, поэтому может работать с различными видами последовательностей и данными различных типов. Объявления фундаментальных алгоритмов находятся в заголовочном файле <algorithm>.

Все алгоритмы STL можно разделить на 4 группы

1) немодифицирующие операции с последовательностями;
2) модифицирующие операции с последовательностями;
3) алгоритмы, связанные с сортировкой;
4) алгоритмы для работы с множествами и пирамидами.

В качестве параметров алгоритму, как правило, передаются итераторы, определяющие начало и конец обрабатываемо последовательности. Вид итераторов определяет типы контейнеров, для которых может использоваться данный алгоритм.

Немодифицирующие операции

Функция	Описание
`for_each()`	выполняет заданную операцию с каждым элементом последовательности
`find()`	находит 1-е вхождение заданного значения в последовательность
`find_if()`	находит 1-е соответствие заданному условию в последовательность
`count()`	подсчитывает количество вхождений заданного значения в последовательность
`count_if()`	подсчитывает количество соответствий заданному условию в последовательности
`search()`	находит 1-е вхождение заданной подпоследовательности

Модифицирующие операции

Функция	Описание
`copy()`	копирует последовательность, начиная с 1-го элемента
`swap()`	меняет местами два элемента
`replace()`	заменяет элементы указанным значением
`replace_if()`	заменяет элементы при выполнении условия
`replace_copy()`	копирует последовательность, заменяя элементы указанным значением
`replace_copy_if()`	копирует последовательность, заменяя элементы при выполнении условия
`fill()`	заполняет все элементы указанным значением
`remove()`	удаляет элементы с заданным значением
`remove_if()`	удаляет элементы при выполнении условия
`remove_copy()`	копирует последовательность, удаляя элементы с заданным значением
`remove_copy_if()`	копирует последовательность, удаляя элементы при выполнении условия
`reverse()`	меняет порядок следования элементов на обратный
`transform()`	выполняет заданную (модифицирующую) операцию с каждым элементом последовательности
`unique()`	удаляет соседние элементы, имеющие одинаковые значения

Алгоритмы, связанные с сортировкой

Функция	Описание
`sort()`	сортирует последовательность с высокой средней эффективностью
`partial_sort()`	сортирует часть последовательности
`stable_sort()`	сортирует последовательность, сохраняя порядок следования одинаковых элементов
`lower_bound()`	находит 1-е вхождение заданного значения в отсортированную последовательность
`upper_bound()`	находит в отсортированной последовательности 1-й элемент, больший заданного значения
`binary_search()`	находит заданный элемент в отсортированной

	последовательности, используя алгоритм двоичного поиска
`merge()`	объединяет две отсортированные последовательности в одну отсортированную
`min_element()`	поиск наименьшего элемента в последовательности
`max_element()`	поиск наибольшего элемента в последовательности

Алгоритмы для работы с множествами

Функция	Описание
includes()	проверка вхождения одного множества в другое
set_union()	объединение множеств
set_intersection()	пересечение множеств
set_difference()	разность множеств
make_heap()	преобразование последовательности с произвольным доступом в пирамиду
pop_heap()	извлечение элемента из пирамиды
push_heap()	добавление элемента в пирамиду
sort_heap()	сортировка пирамиды

Задание

1) Создайте контейнер последовательности (вектор, связный список или дек – см. свой вариант задания) и заполните его элементами пользовательского типа. Перегрузите операции сравнения и выгрузки в поток для пользовательского типа.
2) Замените в контейнере элементы в соответствии со своим вариантом задания. При необходимости используйте алгоритмы `replace()`, `replace_if()`, `replace_copy()`, `fill()`.
3) Удалите из контейнера элементы в соответствии с заданием. При необходимости используйте алгоритмы `remove()`, `remove_if()`, `remove_copy()`.
4) Отсортируйте элементы в контейнере по возрастанию или убыванию с помощью алгоритма `sort()`.

5) Найдите заданный элемент (элементы) последовательности в соответствии с заданием. При необходимости используйте алгоритмы `find()`, `find_if()`, `count()`, `count_if()`.
6) С помощью алгоритмов `for_each()`, `transform()` для каждого элемента последовательности выполните заданное действие в соответствии с заданием.

Варианты заданий

Вариант 1.

1. Контейнер – вектор.
2. Заменить элементы, превосходящие по значению среднее арифметическое для всего контейнера, на элементы с минимальным значением по контейнеру.
3. Удалить два самых больших элемента.
4. Отсортировать по возрастанию.
5. Найти элемент, заданный с клавиатуры.
6. Каждый элемент разделить на максимальное значение в контейнере.

Вариант 2.

1. Контейнер – связный список.
2. Заменить все элементы с нулевым значением на элементы со значением 1.
3. Удалить по два элемента из начала и конца списка.
4. Отсортировать по возрастанию.
5. Найти количество отрицательных элементов.
6. К каждому элементу в контейнере добавить 1.

Вариант 3.

1. Контейнер – дек.
2. Заменить максимальный элемент минимальным.
3. Удалить все элементы, по значению меньшие среднего арифметического.
4. Отсортировать по убыванию.
5. Найти медиану (элемент в центре отсортированной последовательности).
6. Изменить знак каждого элемента в контейнере.

Вариант 4.

1. Контейнер – вектор.
2. Заменить значения трех первых элементов значениями трех последних.
3. Удалить максимальный элемент.
4. Отсортировать по убыванию.
5. Найти первый элемент, по значению превосходящий своих соседей.
6. Каждый элемент в контейнере возвести в квадрат.

Вариант 5.

1. Контейнер – связный список.
2. Заменить минимальный элемент максимальным.
3. Удалить элемент, заданный с клавиатуры.
4. Отсортировать по возрастанию.
5. Найти 1-й элемент, меньший заданного значения.
6. Каждый элемент заменить на среднее арифметическое двух соседей.