Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
“НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО”

Факультет программной инженерии и компьютерной техники

Направление подготовки (специальность) СППО

ОТЧЕТ

о практической работе №3

По дисциплине: программирование на C++

Обучающийся Розмирский Д.В. P4114
(Фамилия И.О.) (номер группы)

Санкт-Петербург
2024 г.

Отчет по лабораторной работе №3

Введение

Целью данной лабораторной работы было изучение использования стандартной библиотеки шаблонов C++, в частности контейнеров и алгоритмов STL. Задача заключалась в работе с различными контейнерами (vector, list) и их эффективной обработке с использованием алгоритмов из стандартной библиотеки, минимизируя использование циклов и ручного кода.

Задачи лабораторной работы

1. Создать вектор v1 размером от 500 до 1000 элементов. Элементами вектора являются экземпляры класса из ЛР №2, имеющие случайные значения.
2. Создать вектор v2, содержащий последние 200 элементов вектора v1.
3. Сформировать список list1, содержащий n наибольших элементов вектора v1 (в диапазоне от 20 до 50).
4. Сформировать список list2, содержащий n наименьших элементов вектора v2.
5. Удалить из векторов v1 и v2 перемещенные элементы.
6. Перегруппировать элементы в list1 так, чтобы в начале оказались все элементы, большие среднего значения.
7. Удалить из list2 все элементы, удовлетворяющие критерию.
8. Создать вектор v3 из элементов, которые присутствуют и в v1, и в v2.
9. Сформировать список list3 из пар элементов list1 и list2 так, чтобы оба списка имели одинаковый размер.
10. Решить предыдущую задачу для векторов v1 и v2 без приведения к одному размеру.

Описание реализации

Генерация данных

1. Создание вектора v1 - используется генератор случайных чисел для создания вектора v1 размером от 500 до 1000 элементов. Каждый элемент — это объект класса myClass, который был разработан в ЛР

№2. Класс myClass содержит поля для хранения значения (value) и приоритета (priority).

2. Создание вектора v2 - создаем вектор v2, копируя последние 200 элементов из вектора v1. Если размер вектора v1 меньше 200, копируем все элементы.

Работа с контейнерами

1. Формирование списка list1 - из вектора v1 выбираются n наибольших элементов с использованием алгоритма std::partial_sort_copy. Затем создается список list1, содержащий отсортированные элементы.
2. Формирование списка list2 - из вектора v2 выбираются n наименьших элементов с помощью алгоритма std::partial_sort_copy, и создается список list2. Порядок элементов в списке list2 не важен.
3. Удаление перемещенных элементов из векторов v1 и v2
   Для удаления элементов, которые были перенесены в list1 и list2, используется алгоритм std::remove_if и контейнер std::set. Это позволяет эффективно найти и удалить все элементы, которые были перемещены.
4. Перегруппировка элементов в list1 - для перегруппировки элементов используется среднее значение элементов списка, которое вычисляется с помощью алгоритма std::accumulate. Затем элементы перегруппировываются так, чтобы все элементы, большие среднего значения, оказались в начале списка.
5. Удаление нечётных элементов из list2 - из list2 удаляются все элементы, имеющие нечетное значение приоритета, с помощью метода remove_if.
6. Создание вектора v3 - вектор v3 создается с помощью алгоритма std::set_intersection, который позволяет найти пересечение элементов в векторах v1 и v2.
7. Формирование списка list3 из пар элементов list1 и list2
   Для создания списка list3 сначала выравниваются размеры list1 и list2. Затем с помощью алгоритма std::transform создаются пары элементов из двух списков и добавляются в list3.
8. Создание пар элементов из v1 и v2 без приведения к одинаковому размеру. Для создания пар элементов из v1 и v2 используется

минимальный размер из двух векторов, после чего пары формируются и добавляются в вектор пар.

Пример вывода

Программа выводит размеры всех контейнеров (v1, v2, list1, list2, list3, v_pairs) для анализа работы программы.

Выводы по результатам работы

1. Эффективное использование STL алгоритмов
   В ходе работы использовались такие алгоритмы как std::partial_sort_copy, std::set_intersection, std::remove_if, std::transform и другие. Это позволило минимизировать количество циклов и ручного кода, что сделало программу более читаемой и эффективной.
2. Перегруппировка и удаление элементов
   Было успешно реализовано удаление и перегруппировка элементов в списках с использованием методов стандартной библиотеки, что позволило избежать избыточных операций и повысить производительность программы.
3. Работа с контейнерами разного типа
   В программе были продемонстрированы различные операции с контейнерами vector и list. Это позволило лучше понять особенности работы с различными STL-контейнерами и их использование в зависимости от задачи.
4. Операторы класса myClass
   Для корректной работы с алгоритмами стандартной библиотеки были реализованы операторы сравнения (<, >, ==) для класса myClass. Это позволило использовать экземпляры класса в алгоритмах, таких как std::partial_sort_copy и std::set_intersection.
5. Анализ производительности
   Использование STL алгоритмов позволило избежать неэффективного кода и уменьшить количество итераций при выполнении задач, связанных с сортировкой, поиском и удалением элементов. Применение алгоритмов вместо циклов позволило добиться более оптимальной работы программы.

Основные недостатки

1. Обработка исключений - в текущей реализации отсутствует обработка исключений, что может привести к сбоям при работе с динамической памятью или некорректным данным.
2. Использование сырых указателей - в реализации класса myClass используются сырые указатели (Node*), что потенциально увеличивает риск утечек памяти. В будущем можно улучшить реализацию, применяя умные указатели (std::unique_ptr).
3. Избыточные операции копирования - в некоторых случаях копирование элементов между контейнерами может приводить к избыточным операциям, которые можно избежать. Оптимизация операций копирования и перемещения может улучшить производительность программы.

Вывод

В ходе лабораторной работы №3 были рассмотрены и реализованы различные операции с контейнерами и алгоритмами STL, такие как сортировка, поиск пересечения, удаление элементов и перегруппировка. Были использованы стандартные алгоритмы STL, что позволило минимизировать использование циклов и сделать код более читаемым и эффективным. Программа продемонстрировала возможности работы с контейнерами, а также подчеркнула важность правильного управления динамической памятью и использования операторов для работы с экземплярами класса.

Исходный код:

https://github.com/RozmiDan/cpp_itmo_labs/tree/main/thd_lab_cpp