|  |  |  |
| --- | --- | --- |
| Министерство науки и высшего образования РФ  Федеральное государственное автономное образовательное учреждение высшего образования  «Пермский государственный национальный исследовательский университет» | | |
|  | Институт компьютерных наук и технологий | |
| **ОТЧЁТ**  по лабораторной работе №1  по дисциплине «Языки программирования»  Вариант 5 | | |
|  | | Работу выполнил  студент группы ПМИ-9,10-2023 2 курса  Казаков Н.С.  «14» Октября 2024 г. |
| Работу проверил  Ракина В.Д. преподавателя  «\_\_\_» \_\_\_\_\_\_\_\_\_ 2024 г. |
| Пермь 2024 | | |

Оглавление

[Задание 1 4](#_Toc180049634)

[Текст задания 4](#_Toc180049635)

[Алгоритм решения 4](#_Toc180049636)

[Тестирование 5](#_Toc180049637)

[Код программы 5](#_Toc180049638)

[Задание 2 6](#_Toc180049639)

[Текст задания 6](#_Toc180049640)

[Алгоритм решения 6](#_Toc180049641)

[Тестирование 7](#_Toc180049642)

[Код программы 7](#_Toc180049644)

[Задание 3 8](#_Toc180049645)

[Текст задания 8](#_Toc180049646)

[Алгоритм решения 8](#_Toc180049647)

[Тестирование 9](#_Toc180049648)

[Код программы 9](#_Toc180049650)

[Задание 4 10](#_Toc180049651)

[Текст задания 10](#_Toc180049652)

[Алгоритм решения 10](#_Toc180049653)

[Тестирование 11](#_Toc180049654)

[Код программы 11](#_Toc180049656)

[Задание 5 12](#_Toc180049657)

[Текст задания 12](#_Toc180049658)

[Алгоритм решения 12](#_Toc180049659)

[Тестирование 13](#_Toc180049660)

[Код программы 13](#_Toc180049662)

[Задание 6 14](#_Toc180049663)

[Текст задания 14](#_Toc180049664)

[Алгоритм решения 14](#_Toc180049665)

[Тестирование 14](#_Toc180049666)

# Задание 1

## Текст задания

Даны вектор V, дек D и список L. Каждый исходный контейнер содержит не менее трех элементов, количество элементов является нечетным. Удвоить значения первого, среднего и последнего элемента каждого из исходных контейнеров.

## Алгоритм решения

1.Инициализация контейнеров:

-Даны три контейнера: вектор V, дек D и список L.

-Проверяется, что каждый контейнер содержит не менее 3 элементов, и их количество является нечетным.

2.Поиск элементов для изменения: Для каждого контейнера выполняется одинаковый набор операций:

-Найти первый элемент: это элемент, находящийся в начале контейнера.

-Найти средний элемент: его индекс или позиция равна половине размера контейнера (индекс = size / 2).

-Найти последний элемент: это элемент, находящийся в конце контейнера.

3.Удвоение значений:

-Для каждого из найденных элементов (первого, среднего и последнего) выполняется -операция удвоения. То есть значение каждого из них умножается на 2.

-Повторение операции для каждого контейнера:

-Повторить шаги 2 и 3 для всех контейнеров: вектора V, дека D и списка L.

4.Вывод результатов:

-После выполнения всех операций, вывести содержимое контейнеров для проверки результатов.

# Тестирование

## Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение Автоматически созданное описаниеКод программы

[KNs4/LAB-c-2 (github.com)](https://github.com/KNs4/LAB-c-2)

# Задание 2

## Текст задания

Даны дек D и список L. Каждый исходный контейнер содержит не менее 5 элементов. Вставить перед пятым с конца элементом списка последние 5 элементов дека в обратном порядке. Использовать один вызов функции-члена insert.

## Алгоритм решения

1.Проверка количества элементов:

-Убедиться, что как дек D, так и список L содержат по крайней мере 5 элементов.

2.Определение пятого элемента с конца в списке L:

-Для списка L найти итератор, указывающий на пятый элемент с конца:

-Использовать итератор end() для обозначения конца списка.

-Сдвинуть этот итератор на 5 позиций назад (используя advance()), чтобы получить итератор на пятый с конца элемент.

3.Получение последних 5 элементов из дека D:

-Из дека D выбрать последние 5 элементов.

-Использовать реверсированные итераторы rbegin() и rend() для обращения к этим элементам в обратном порядке.

4.Вставка элементов:

-Использовать один вызов функции-члена insert() для вставки последних 5 элементов дека (в обратном порядке) в список L перед пятым с конца элементом, найденным на шаге 2.

5.Вывод результатов:

-После вставки вывести список L для проверки результата.

## Тестирование

## Изображение выглядит как текст, снимок экрана, Шрифт, линия Автоматически созданное описание

## Код программы

[KNs4/LAB-c-2 (github.com)](https://github.com/KNs4/LAB-c-2)

# Задание 3

## Текст задания

Даны два списка L1 и L2 с одинаковым количеством элементов N. Получить в списке L2 комбинированный набор элементов исходных списков вида A1, B1, A2, B2, A3, B3, …, AN, BN, где Ai обозначают элементы исходного списка L1, а Bi — элементы исходного списка L2. Для этого использовать итераторы i1 (для списка L1) и i2 (для списка L2), связав их с первым элементом соответствующего списка. В цикле, который должен повторяться N раз, вызывать функциючлен splice для списка L2 с первым параметром ++i2 и третьим параметром i1++

## Алгоритм решения

Инициализация:

-Проверить, что списки L1 и L2 имеют одинаковое количество элементов.

-Если это условие не выполняется, выдать ошибку.

-Инициализировать итераторы i1 для списка L1 и i2 для списка L2, установив их на первые элементы соответствующих списков.

Цикл для обработки элементов:

-Цикл должен выполняться N раз, где N — количество элементов в списке.

-На каждой итерации:

--Переместить текущий элемент, на который указывает итератор i1 из списка L1 в список L2 после позиции, на которую указывает итератор i2.

-Для этого использовать функцию splice:

--Первый параметр: ++i2 (следующая позиция в списке L2).

--Второй параметр — список, из которого перемещаем элемент (L1).

--Третий параметр — итератор i1, который указывает на текущий элемент в L1.

-После перемещения элемента из L1 инкрементировать итераторы i1 и i2.

Завершение работы:

-После выполнения цикла все элементы из L1 будут перемещены в L2 в чередующемся порядке.

-Список L1 станет пустым, а список L2 будет содержать элементы из обоих списков в виде: A1, B1, A2, B2, …, AN, BN.

Вывод результата:

-Вывести итоговый список L2, чтобы убедиться, что чередование выполнено правильно.

## Тестирование

## 

## Код программы

[KNs4/LAB-c-2 (github.com)](https://github.com/KNs4/LAB-c-2)

# Задание 4

## Текст задания

Дан набор символов. Вывести все символы из исходного набора в том же порядке, заменяя цифровые символы на символ подчеркивания. Использовать алгоритм replace\_copy\_if.

## Алгоритм решения

Ввод набора символов:

-Считать строку символов, введенную пользователем.

Лямбда-функция для проверки цифровых символов:

-Определить условие для замены символов. Использовать функцию isdigit, которая проверяет, является ли символ цифрой.

Использование алгоритма replace\_copy\_if:

-Алгоритм replace\_copy\_if заменяет все символы, удовлетворяющие заданному условию, на другой символ.

-В качестве условия передать лямбда-функцию, которая возвращает true, если символ является цифрой.

-Вместо цифр будет использоваться символ подчеркивания \_.

-Результат будет выводиться с использованием итераторов, таких как ostream\_iterator для непосредственного вывода результата на экран.

Вывод результата:

-Алгоритм работает по принципу копирования: он копирует все символы из исходного набора в вывод, заменяя только цифры на подчеркивания.

## Тестирование

## Изображение выглядит как снимок экрана, текст, линия, Шрифт Автоматически созданное описание

## Код программы

[KNs4/LAB-c-2 (github.com)](https://github.com/KNs4/LAB-c-2)

# Задание 5

## Текст задания

Дан список L, содержащий как отрицательные, так и положительные элементы. Вставить нулевой элемент после первого отрицательного элемента и перед последним положительным элементом. Использовать два вызова алгоритма find\_if и два вызова функции-члена insert. Указание.Алгоритм find может возвращать обратный итератор, однако функция-член erase не позволяет его использовать для удаления элемента. Необходимо перейти от обратного итератора r к связанному с ним обычному итератору, используя функцию-член обратного итератора r.base(). При этом следует учитывать, что функция r.base() возвращает итератор, связанный с элементом, следующим за тем, с которым связан обратный итератор r. Поэтому в функции-члене erase следует указать одно из следующих выражений (предполагается, что r — это обратный итератор, который вернул алгоритм find, и этот итератор отличен от rend): -- r.base() или (++r).base().

## Алгоритм решения

Поиск первого отрицательного элемента:

-Использовать алгоритм find\_if для поиска первого отрицательного элемента в списке. Для этого передать лямбда-функцию, которая проверяет, меньше ли элемент нуля.

Вставка нуля после первого отрицательного элемента:

-Если первый отрицательный элемент найден, использовать функцию-член insert для вставки нулевого элемента после этого элемента. Для этого использовать std::next с итератором, указывающим на первый отрицательный элемент.

Поиск последнего положительного элемента:

-Использовать алгоритм find\_if в обратном направлении (через rbegin) для поиска последнего положительного элемента. Лямбда-функция будет проверять, больше ли элемент нуля.

Вставка нуля перед последним положительным элементом:

-Алгоритм find\_if вернёт обратный итератор. Чтобы использовать его для вставки элемента, нужно преобразовать его в обычный итератор с помощью base(). Вставить 0 перед этим элементом, корректно управляя смещением итератора.

## Тестирование

## Изображение выглядит как снимок экрана, линия, Шрифт Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Шрифт, линия Автоматически созданное описание

## Код программы

[KNs4/LAB-c-2 (github.com)](https://github.com/KNs4/LAB-c-2)

# Задание 6

## Текст задания

Даны списки L1 и L2, каждый из которых содержит четное количество элементов. Поменять местами первую и вторую половину каждого списка (например, список 1, 2, 3, 4 должен быть преобразован следующим образом: 3, 4, 1, 2). Для первого списка использовать алгоритм swap\_ranges, для второго — алгоритм rotate. Использовать также функцию advance

## Алгоритм решения

Проверка четности количества элементов:

-Перед началом решения убедиться, что каждый список содержит четное количество элементов. Это важно, так как задача требует разделения списка на две равные части.

Обработка списка L1 с помощью swap\_ranges:

-Найти середину списка L1 с помощью функции advance. Для этого создаем итератор, указывающий на начало списка, и сдвигаем его на половину длины списка.

-Использовать swap\_ranges для обмена элементов первой половины с элементами второй половины списка.

Обработка списка L2 с помощью rotate:

-Найти середину списка L2 с помощью функции advance.

-Использовать алгоритм rotate, чтобы переместить вторую половину списка в начало, а первую половину — в конец.

## Тестирование

## Изображение выглядит как текст, снимок экрана, Шрифт Автоматически созданное описание Изображение выглядит как снимок экрана, линия, Шрифт, текст Автоматически созданное описаниеКод программы

[KNs4/LAB-c-2 (github.com)](https://github.com/KNs4/LAB-c-2)