МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Челябинский государственный университет»**

**(ФГБОУ ВО «ЧелГУ»)**

Институт информационных технологий

Кафедра информационных технологий и экономической информатики

ОТЧЕТ

по лабораторной работе №3

Авторы отчета: Постнов Алексей, ПрИ-202,

Ларюшкин Евгений, ПрИ-202,

Пеньков Фёдор, ПрИ-202.

Проверил: Николаев И.Е.

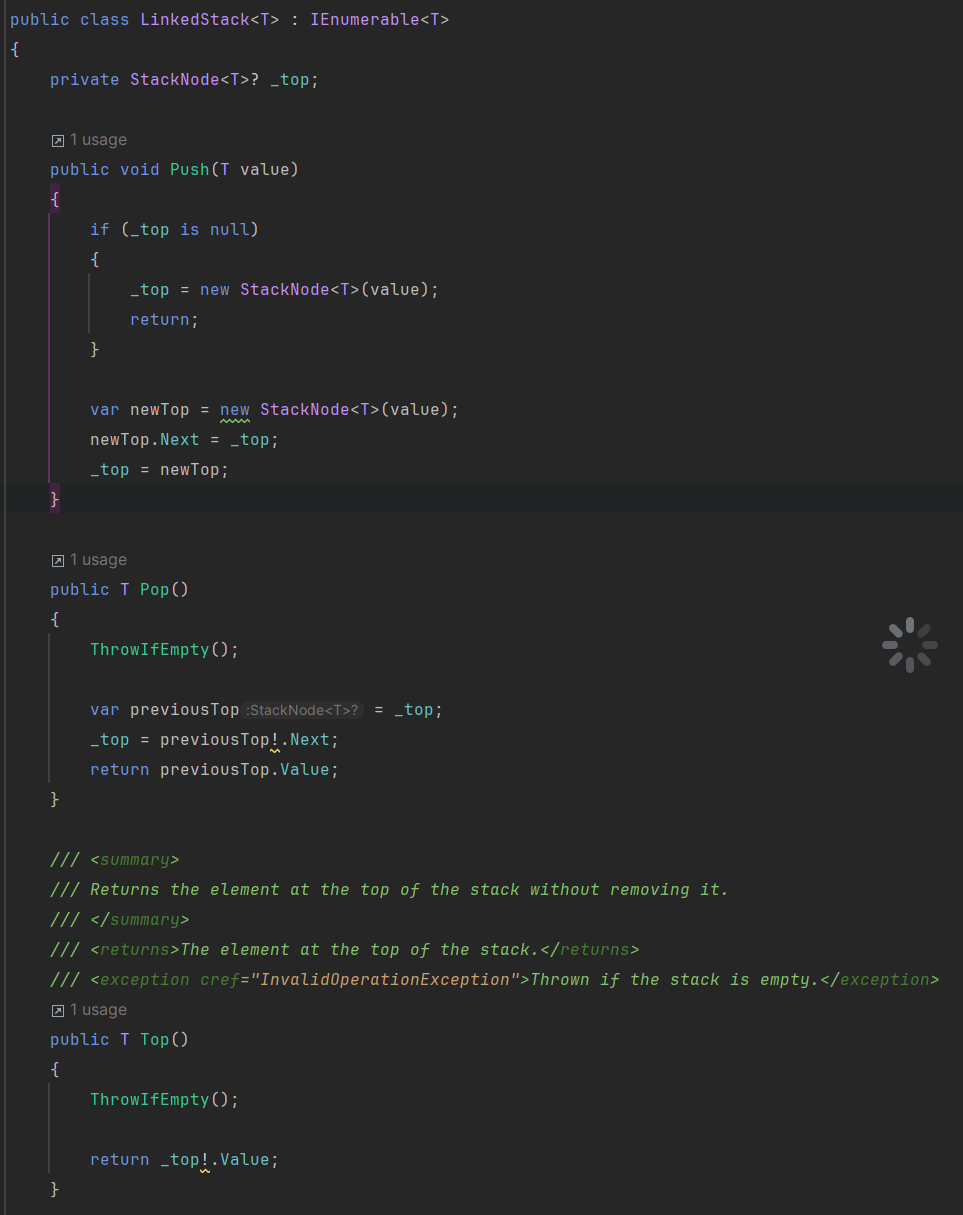
Отчет защищен \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата оценка

Челябинск 2024 г.

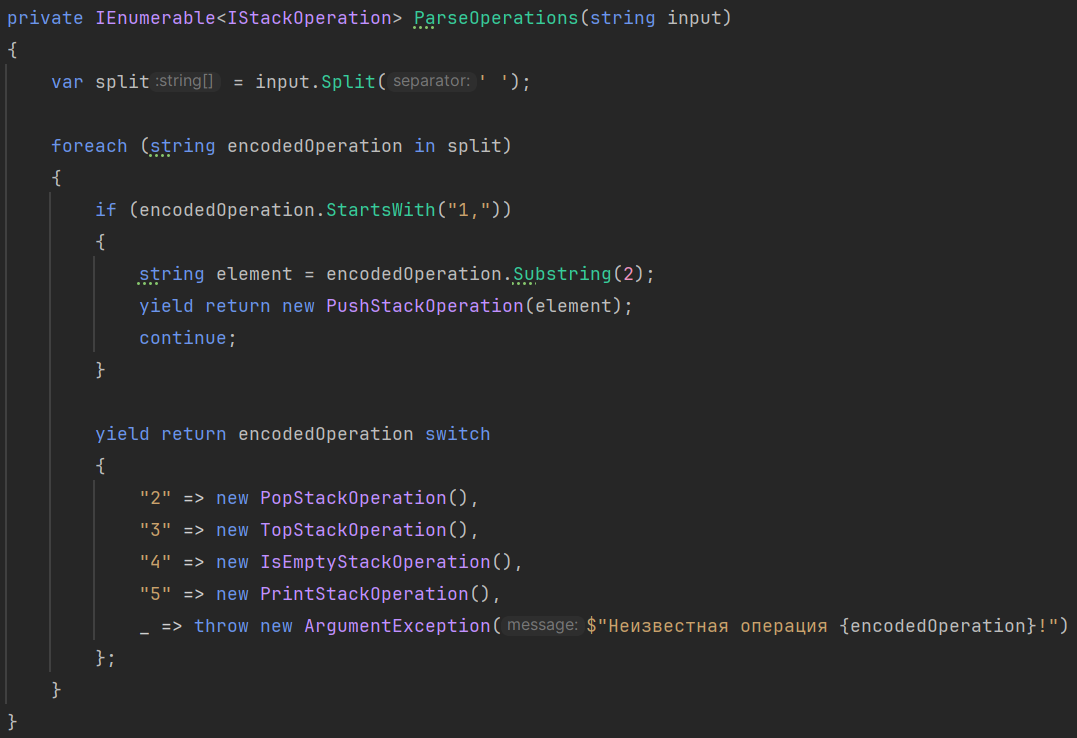
# **Часть 1**

1. Реализовать Стек с помощью СВОЕГО связного списка. Должны поддерживаться следующие операции работы со стеком: Push(elem), Pop(), Top(), isEmpty(), Print().



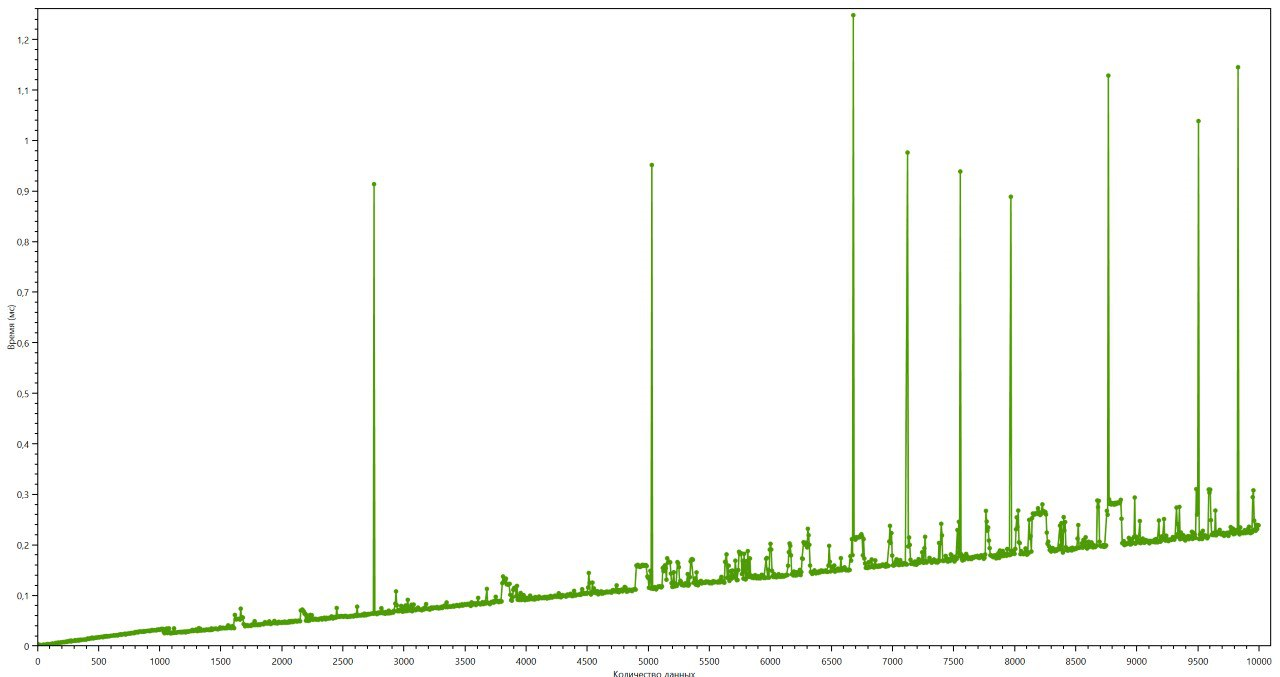
Реализация: Стек

1. В файле input.txt записаны числа от 1 до 5 через пробел. Каждому числу соответствует своя операция: 1 - Push(elem), 2 - Pop(), 3 - Top(), 4 - isEmpty(), 5 - Print(). Для операции push после единицы через запятую указывается значение помещаемого элемента (это может быть число либо слово). Результат выполнения каждой операции выводится на экран.

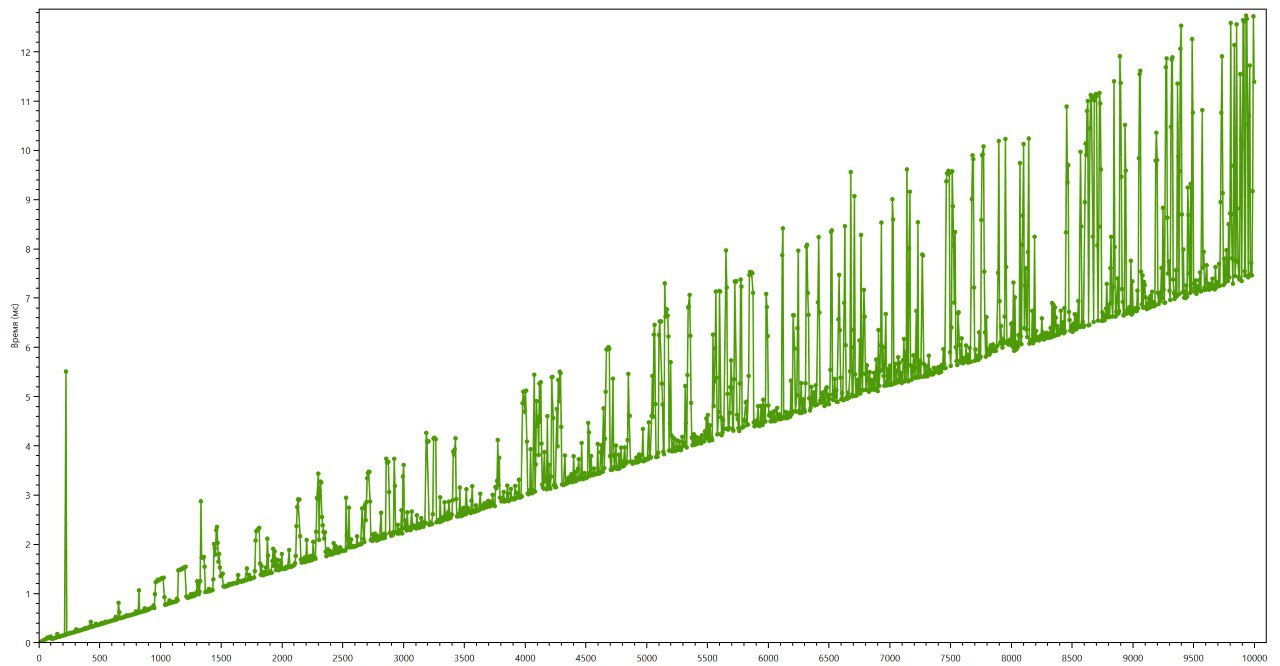


Реализация: Вывод каждой операции Стека

1. Произвести запуск с замером времени на считывание и выполнение операций стеком. В отчет занести результаты замеров времени для различных наборов операций в файле input.txt. Наборы должны быть различными по размеру и составу операций.



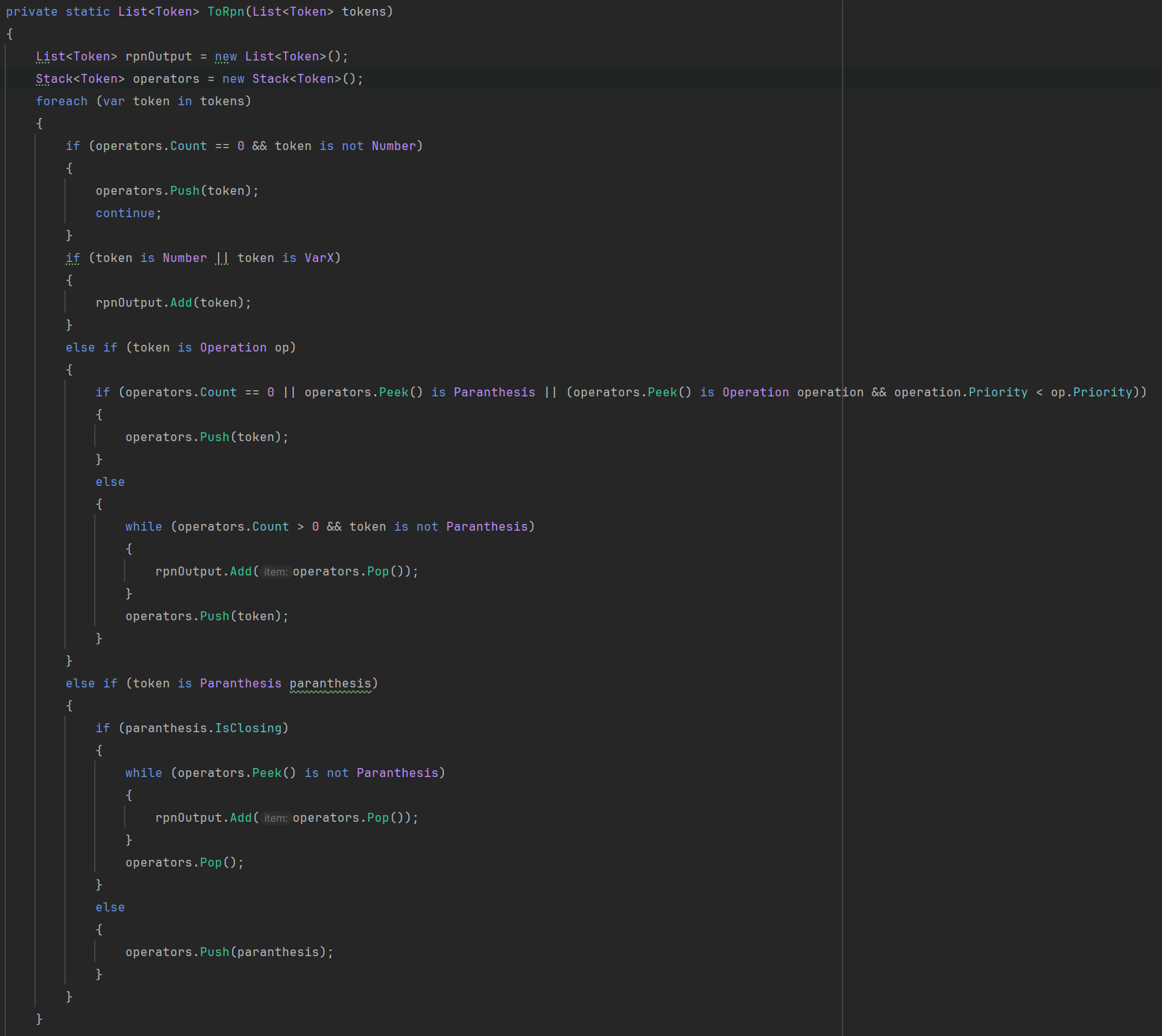
Оценка сложности: Наш стек



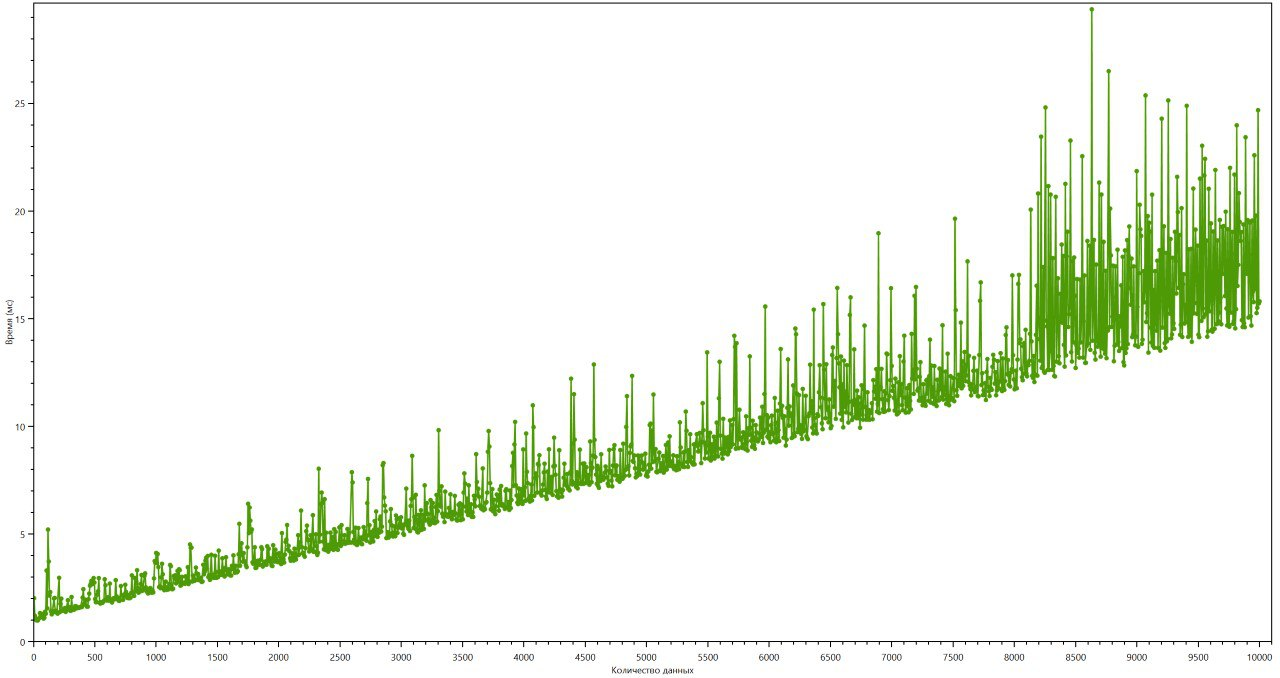
Оценка сложности: Стандартный стек

1. Реализовать алгоритм вычисления выражения, записанного в постфиксной записи (используя постфиксные вычисления). Постфиксная запись считывается из файла. В выражение входят только числа и знаки операций (+, -, \*, :, ^, ln, cos, sin, sqrt, «)».

После реализации алгоритма вычисления постфиксной записи, необходимо произвести расчет оценки сложности алгоритма.



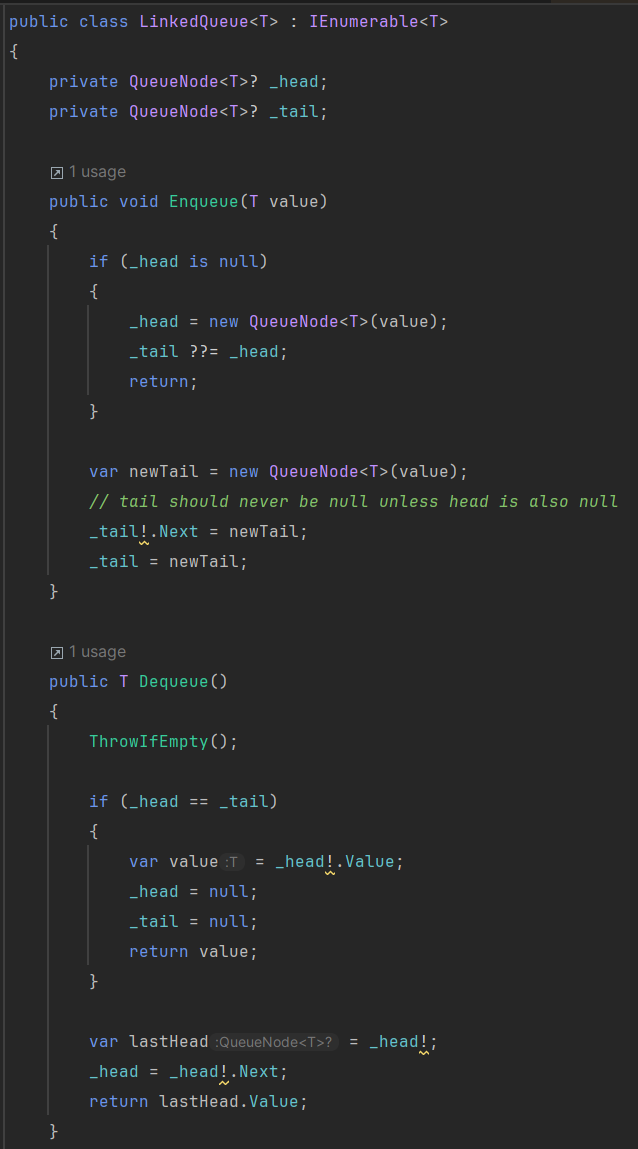
Реализация: Постфиксная запись



Оценка сложности: Постфиксная запись

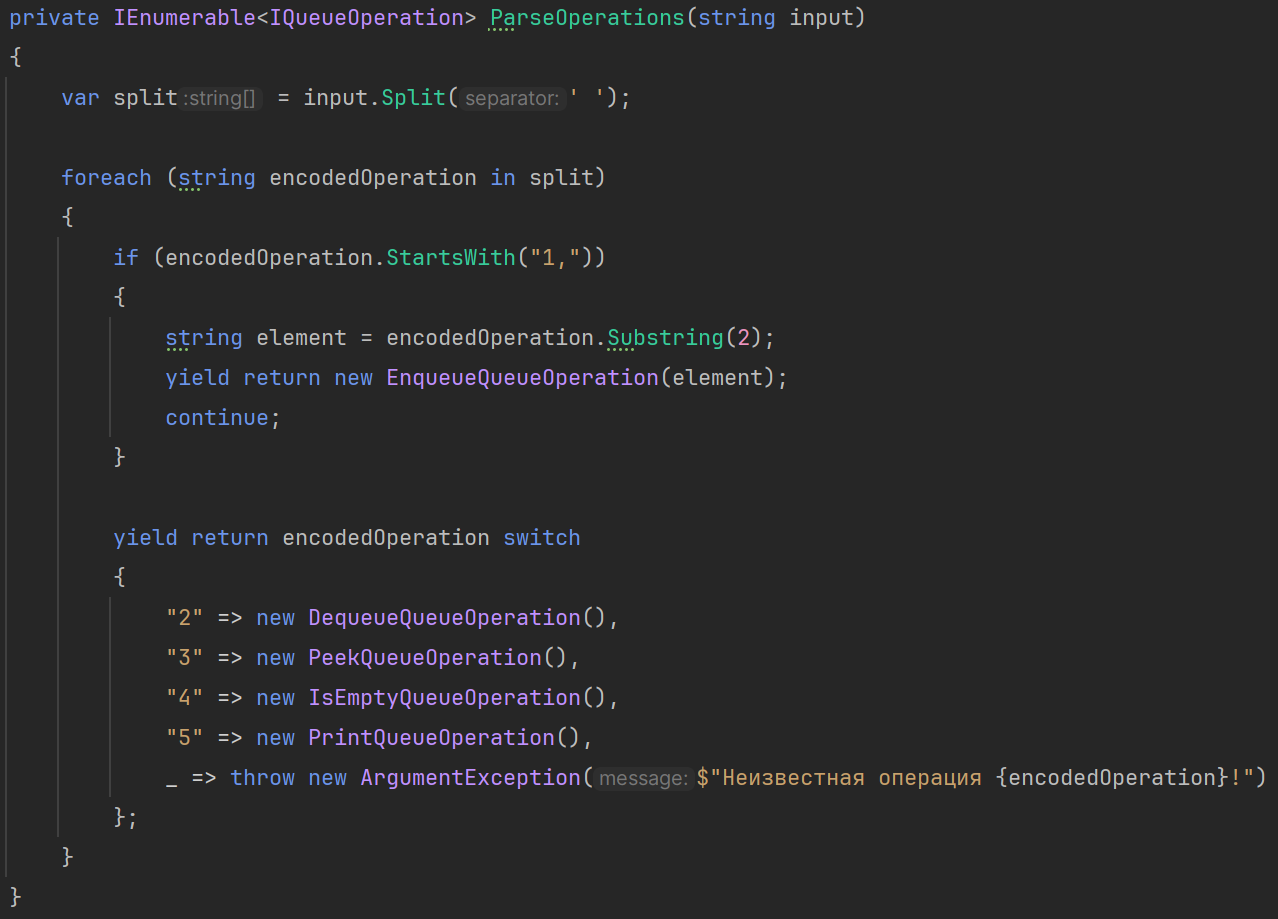
# **Часть 2**

1. Реализовать Очередь с помощью списка и с помощью стандартного класса Queue. Должны поддерживаться следующие операции работы с очередью: вставка/удаление элемента, проверка на пустоту, печать, вывод первого элемента.



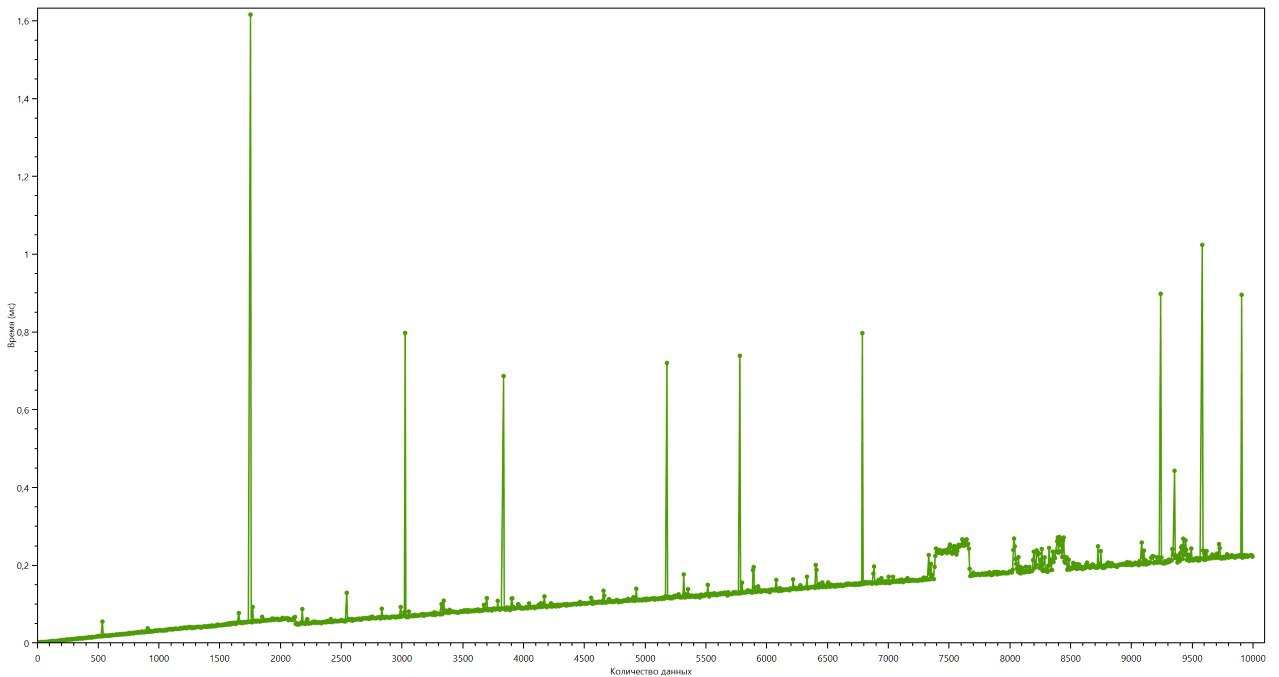
Реализация: Очередь

1. В файле input.txt записаны числа от 1 до 5 через пробел. Каждому числу соответствует своя операция: 1 - вставка, 2 - удаление, 3 – просмотр начала очереди, 4 – проверка на пустоту, 5 - печать. Для операции добавления в очередь после единицы через запятую указывается значение помещаемого элемента (это может быть число либо слово). Результат выполнения каждой операции выводится на экран.

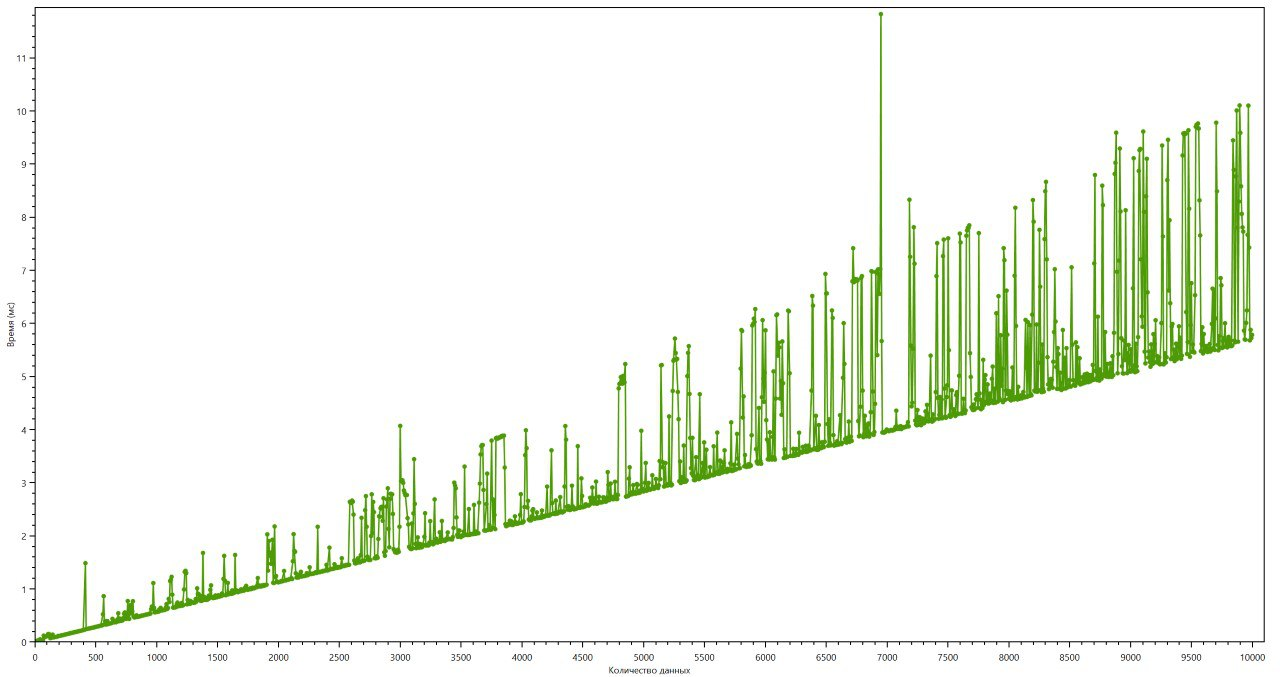


Реализация: Вывод каждой операции Стека

1. Произвести запуск с замером времени на считывание и выполнение операций очередью обоих реализация. В отчет занести результаты замеров времени для различных наборов операций в файле input.txt. Наборы операций должны быть следующими: 1) различными по длине, 2) одинаковыми по длине, но различными по составу операций (в этом случае вам необходимо добиться максимального расхождения по времени).



Оценка сложности: Наша Очередь



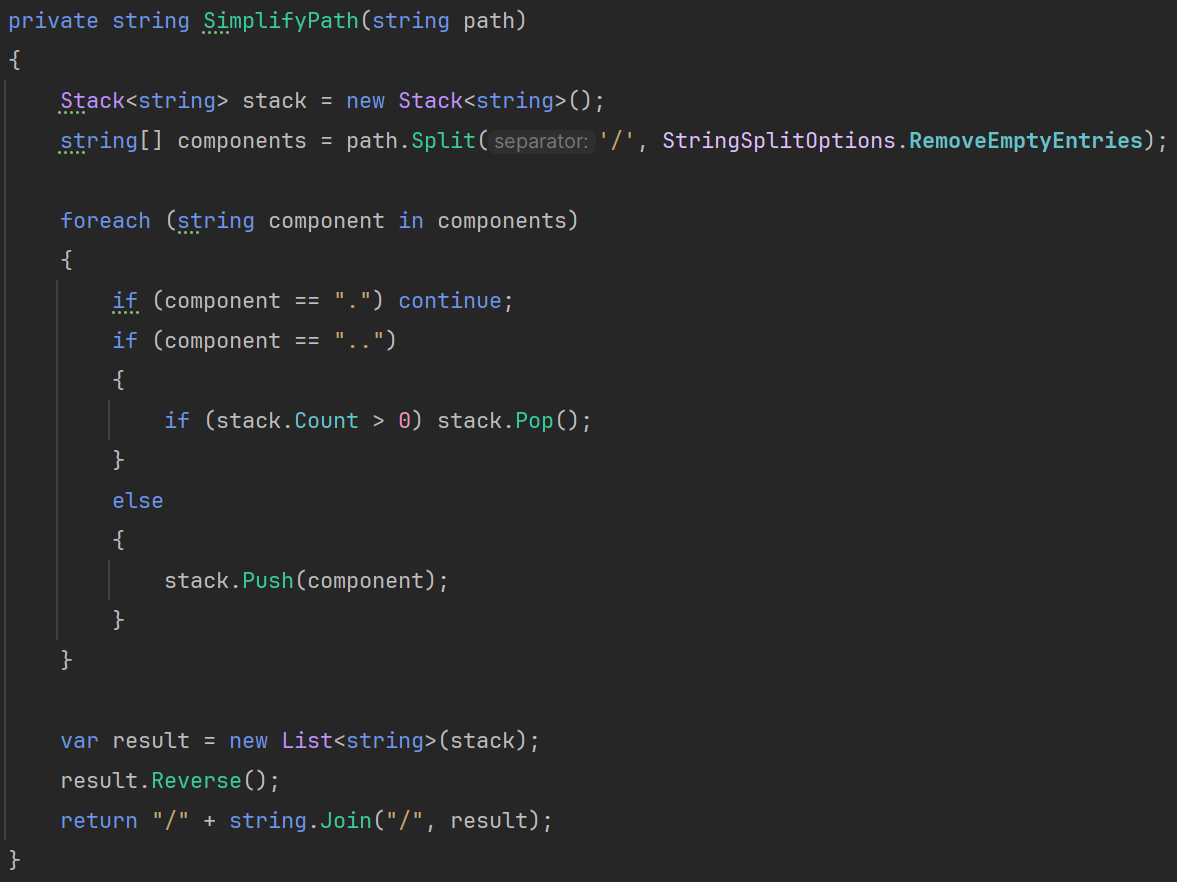
Оценка сложности: Стандартная Очередь

# **Часть 3**

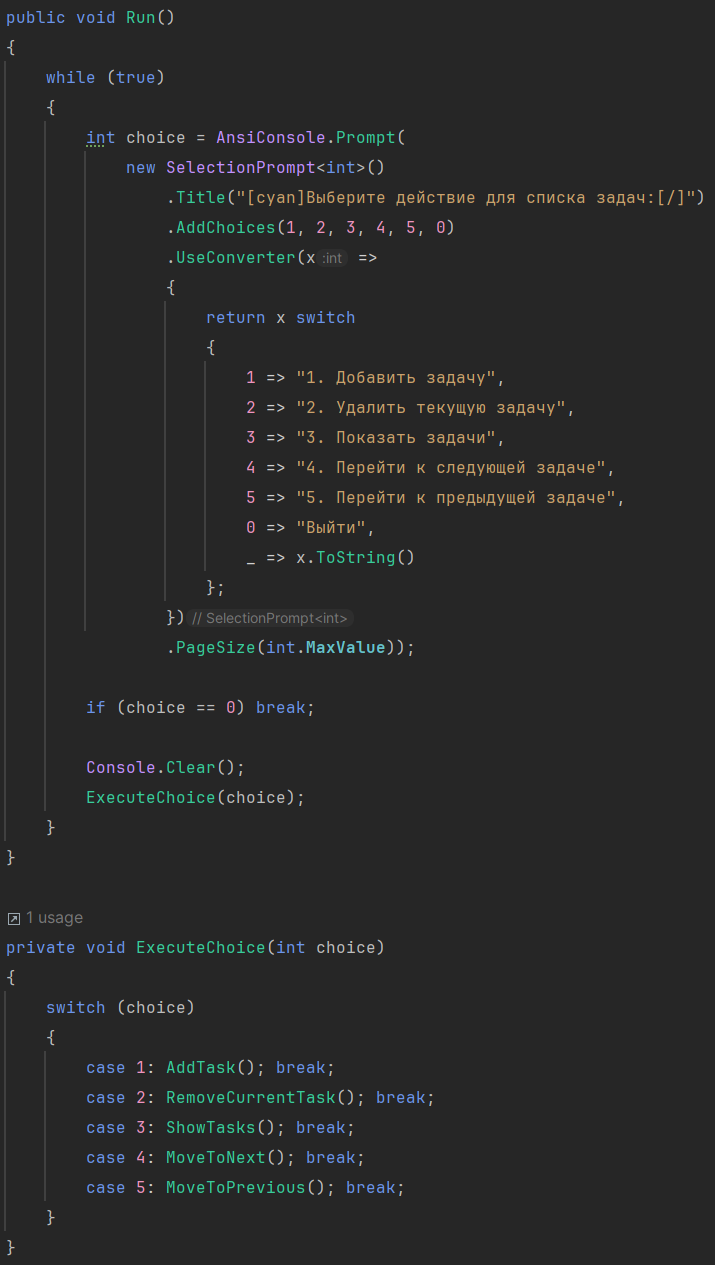
Каждая команда находит по 1 примеру применения структуры данных Список, Стек, Очередь, Деревья. Применение структуры данных может быть в рамках решения какой-либо стандартной/классической задачи/алгоритма, в рамках какой-либо технологии или метода. + реализация.

Каждая команда должна привести уникальный пример применимости (уникальный в рамках группы).

Упрощение пути (Стек)



To do list (Список):



Выполнение задач по очереди (Очередь):

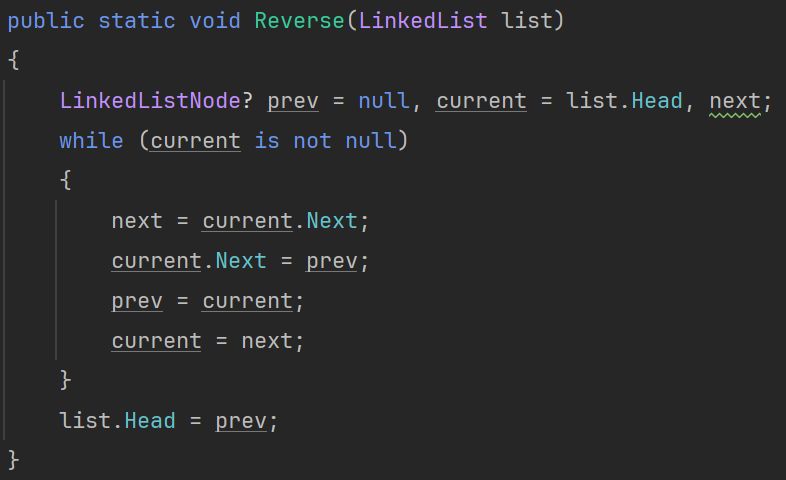


Access control list (Дерево):

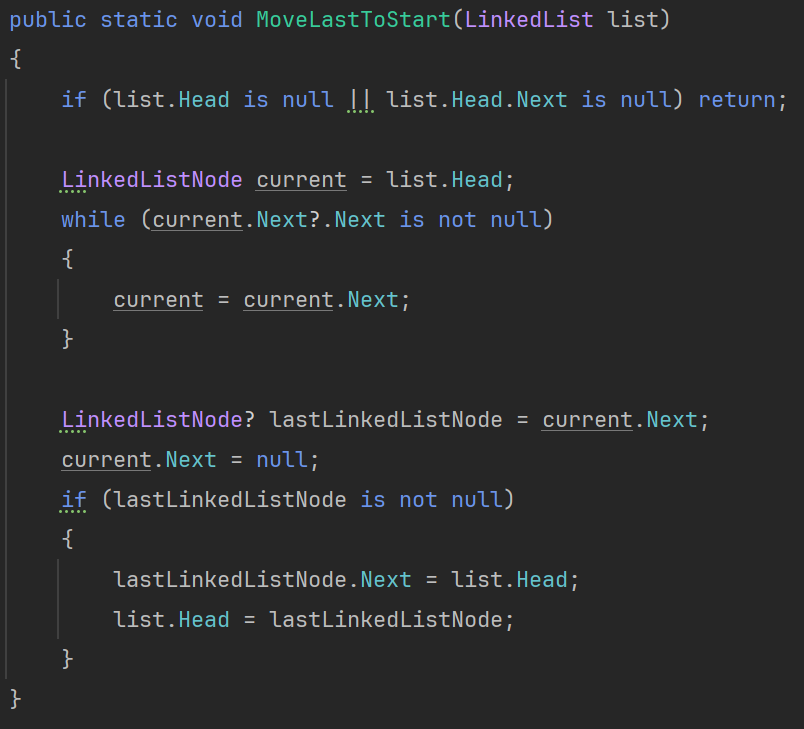


# **Часть 4**

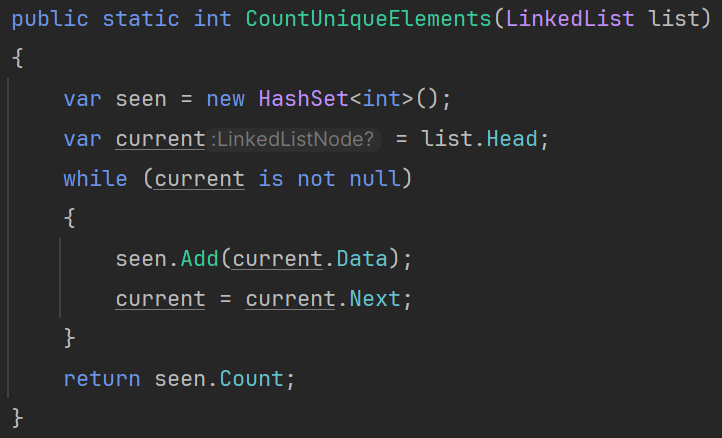
Написать функцию, которая переворачивает список L, т.е. изменяет ссылки в этом списке так, чтобы его элементы оказались расположенными в обратном порядке.



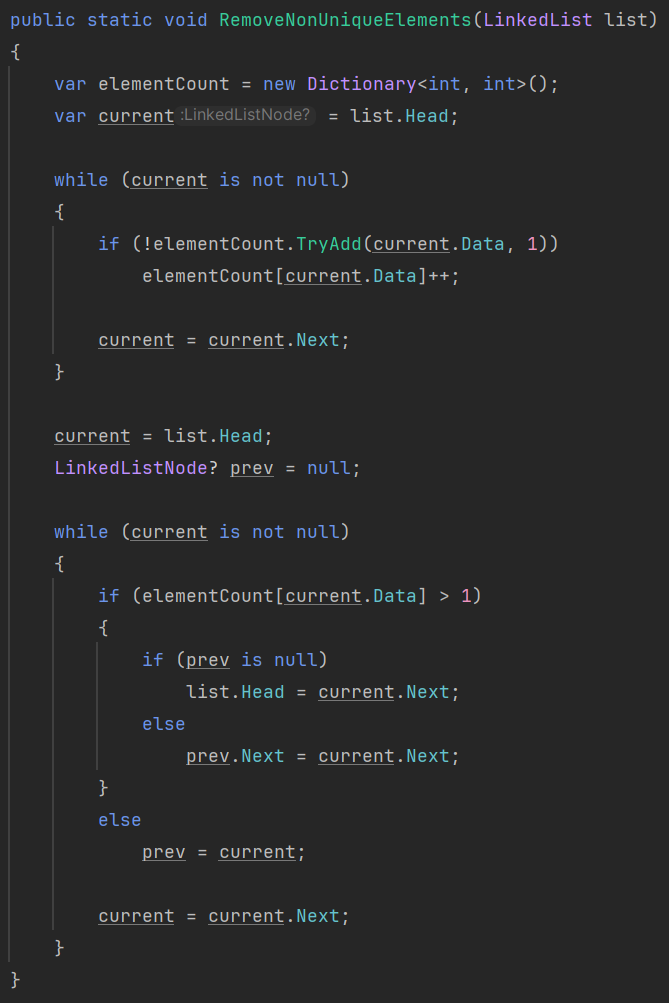
Написать функцию, которая переносит в начало (в конец) непустого списка L его последний (первый) элемент.



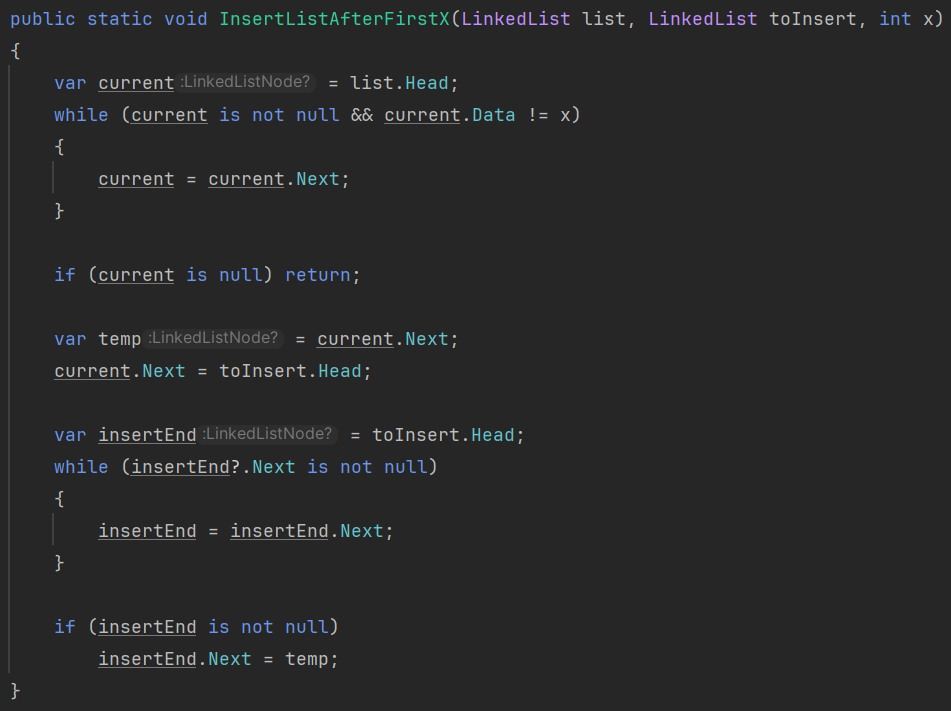
Написать функцию, которая определяет количество различных элементов списка, содержащего целые числа.



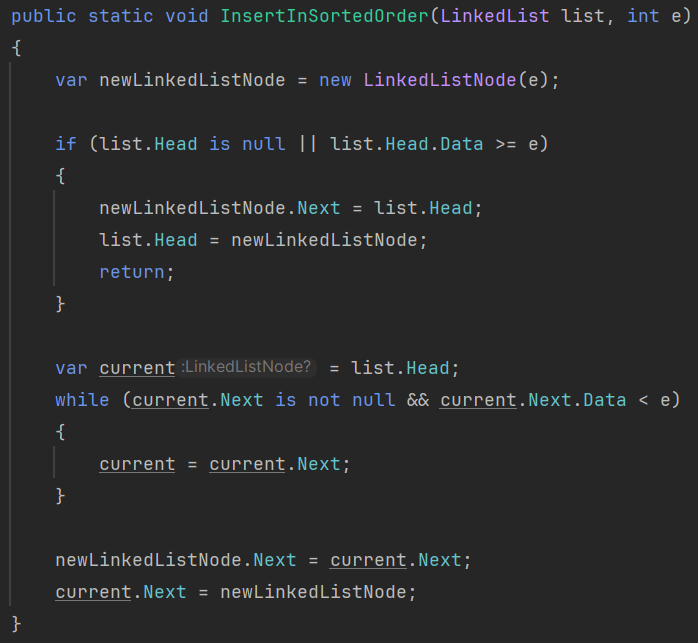
Написать функцию, которая удаляет из списка L неуникальные элементы.



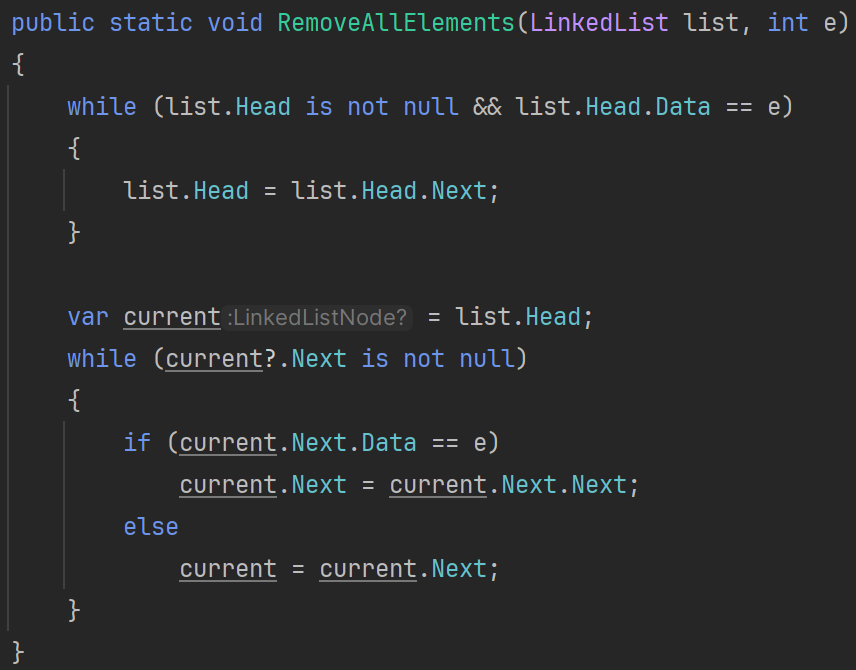
Написать функцию вставки списка самого в себя вслед за первым вхождением числа х.



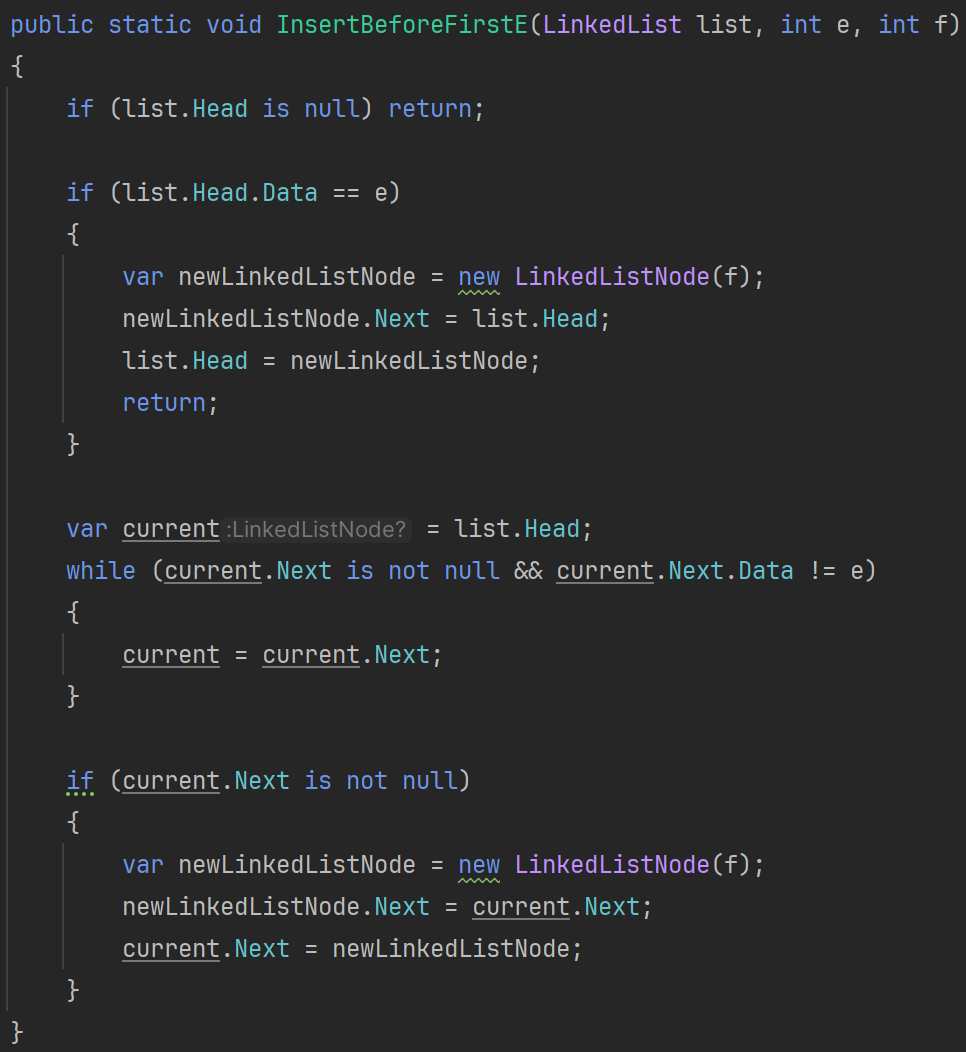
Написать функцию, которая вставляет в непустой список L, элементы которого упорядочены по не убыванию, новый элемент Е так, чтобы сохранилась упорядоченность.



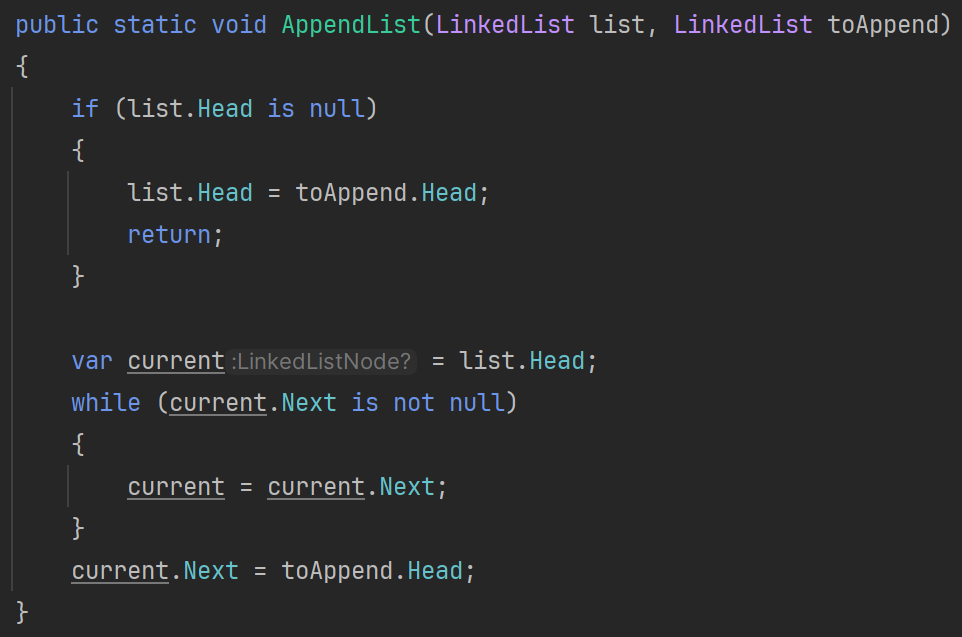
Написать функцию, которая удаляет из списка L все элементы Е, если таковые имеются.



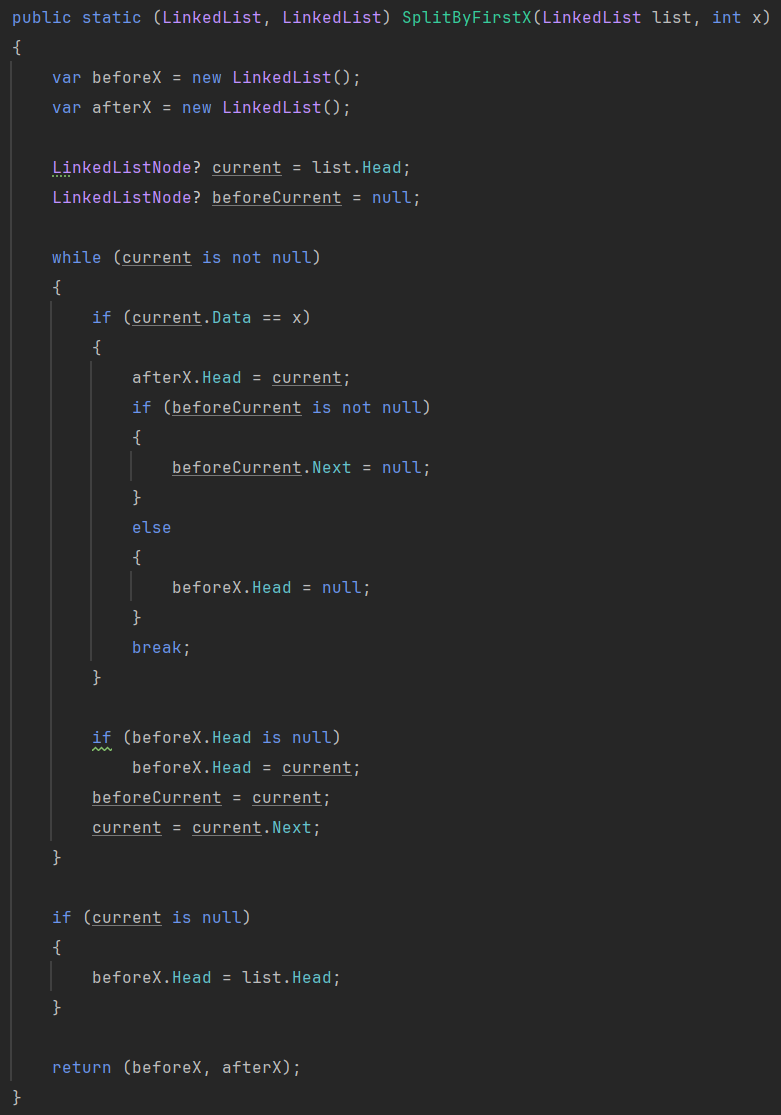
Написать функцию, которая вставляет в список L новый элемент F перед первым вхождением элемента Е, если Е входит в L.



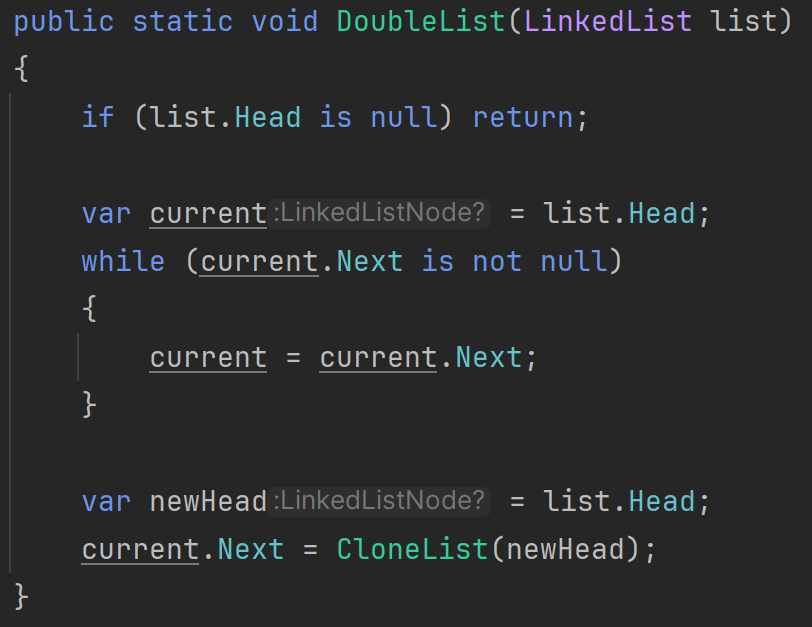
Функция дописывает к списку L список E. Оба списка содержат целые числа. В основной программе считать их из файла.



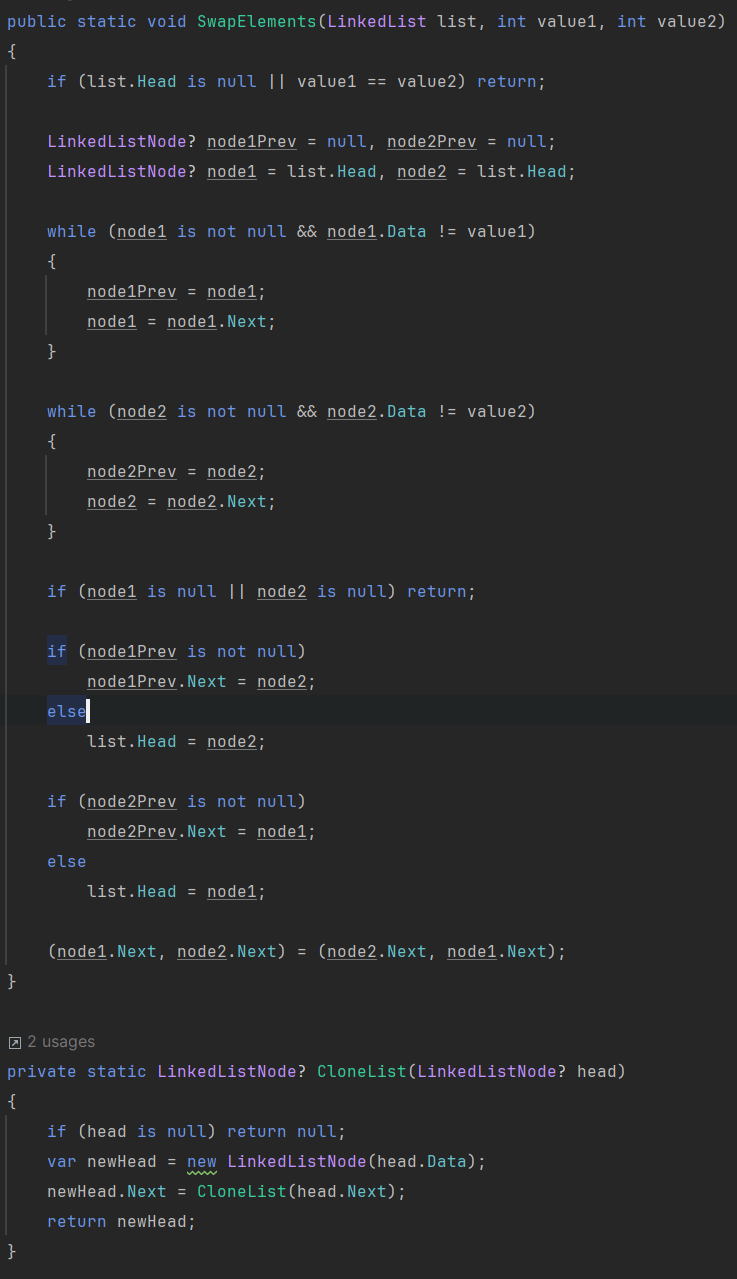
Функция разбивает список целых чисел на два списка по первому вхождению заданного числа. Если этого числа в списке нет, второй список будет пустым, а первый не изменится.



Функция удваивает список, т.е. приписывает в конец списка себя самого.



Функция меняет местами два элемента списка, заданные пользователем



Дерево Евгения:

