**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра САПР**

**ОТЧЁТ**

**по лабораторной работе №1 (вар.13)**

**по дисциплине «Алгоритмы и структуры данных»**

**Тема: Списки**

Студент гр. 0322 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Шаронин А.Д.

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Пестерев Д.О.

Санкт-Петербург

2022

# Задача

Реализовать объект в виде двусвязного списка со следующими методами:

1. void push\_back(T data); // добавление в конец списка
2. void push\_front(T data); // добавление в начало списка
3. void pop\_back(); // удаление последнего элемента
4. void pop\_front(); // удаление первого элемента
5. void insert(T data, int index); // добавление элемента по индексу
6. T& operator[](const int index); // получение элемента по индексу
7. void removeAt(int index); // удаление элемента по индексу
8. int getSize() { return Size; }; // получение размера списка
9. void clear(); // удаление всех элементов списка
10. void changeAt(T data, int index); // замена элемента по индексу на передаваемый элемент
11. void is\_clear(); // проверка на пустоту списка
12. void variant\_task(List& second\_list); // поиск первого вхождения другого списка в список

# Описание реализуемого класса и методов

Реализуемый класс List имеет public-методы, описанные выше, и поля указатели на голову и хвост списка, а также размер самого списка. Структура Node включает себя указатели на следующий элемент и на предыдущий, а также поле типа <T>(шаблон) для хранения данных.

void push\_back(T data): добавляем новый элемент в конец списка путем перенаправления указателей элементов.

void push\_front(T data): добавляем новый элемент в конец списка путем перенаправления указателей элементов.

void pop\_back(): удаляем хвост списка путём назначения нового и удаления старого.

void pop\_front(): удаляем голову списка путём назначения новой и удаления старой.

void insert(T data, int index): вставляем новый элемент списка путём его создания и перенаправления указателей соседних элементов.

T& operator[](const int index): возвращаем данный элемента из поля data путём пробега по списку до нужного элемента.

void removeAt(int index): удаляем элемент списка путём связывания соседних от него элементов с помощью указателей, а затем удаляя данный элемент.

int getSize() { return Size; }: возвращаем значения поля Size, которое отображает длину спсика.

void clear(): удаляем все элементы списка путём вызова метода pop\_front(), пока список не будет пуст.

void changeAt(T data, int index): меняем данные элемента списка путём пробега по списку до нужного элемента, а затем меняя его поле data на нужное нам значение.

void is\_clear(): проверяем пуст ли список используя поле Size.

void variant\_task(List& second\_list): ищем последние вхождение списка в список путём пробега по основному списку и сравнивания его со вторым списком.

# Пример работы

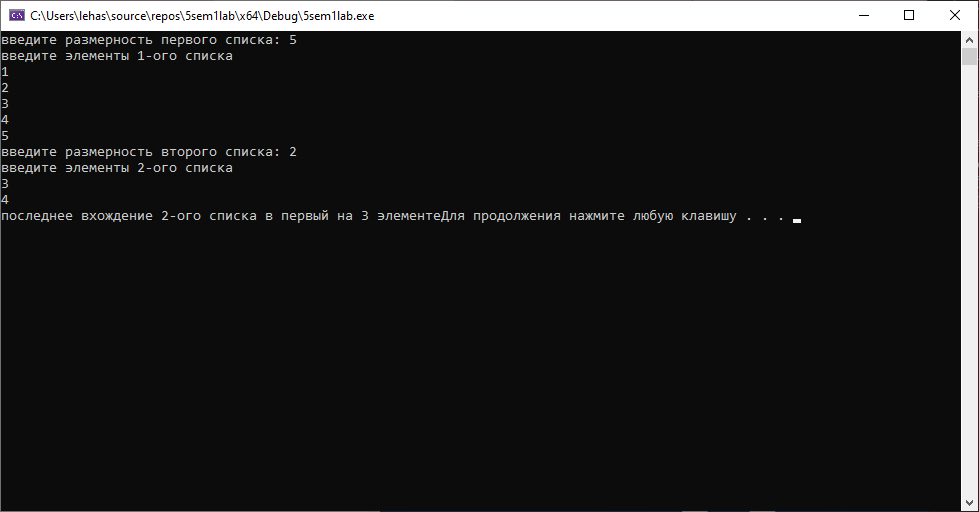
В данном примере показана работа задания по варианту (Рисунок 1).

Рисунок 1

# Листинг

Листинг программы представлен на гитхабе по ссылке: https://github.com/FOOZBY/5sem1lab.git