Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Информатика»

Тема: Лабораторная работа 11

"Динамические структуры данных"

Семестр 2

Выполнил работу

Студент группы РИС-22-1Б

Бердышев Д.И.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

Г. Пермь-2023

# **Постановка задачи**

Написать программу, в которой создаются динамические структуры и выполнить их обработку в соответствии со своим вариантом.

Для каждого вариант разработать следующие функции:

1. Создание списка.

2. Добавление элемента в список (в соответствии со своим вариантом).

3. Удаление элемента из списка (в соответствии со своим вариантом).

4. Печать списка.

5. Запись списка в файл.

6. Уничтожение списка.

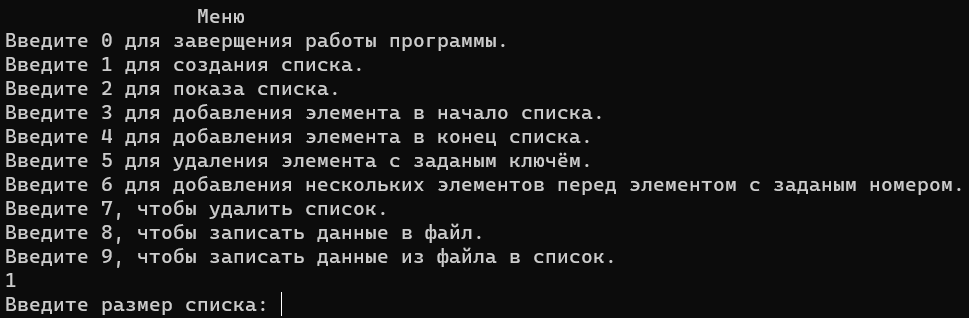
7. Восстановление списка из файла.

Вариант 25

Записи в линейном списке содержат ключевое поле типа \*char (строка символов). Сформировать двунаправленный (однонаправленный) список. Удалить элемент с заданным ключом. Добавить К элементов перед элементом с заданным номером.

# Анализ решения

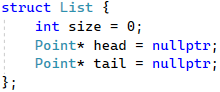
Для удобства работы с программой необходимо создать функцию menu и функцию ShowMenu, которая позволит работать непосредственно с меню программы и завершать работу тогда, когда это нужно. Также, после каждого совершённого действия необходимо очищать консоль с помощью функции system(“cls”).



*Рис. 1 - реализация меню программы в консоли*

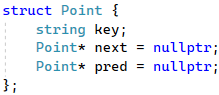
Также для удобства работы со списком создадим две структуры:

* Структура списка

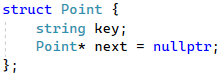


*Рис. 2 - структура List для работы с однонаправленным и двунаправленным списками*

* Структура элемента списка



*Рис. 3 - структура Point для работы с двунаправленным списком*

**

*Рис. 4 - структура Point для работы с однонаправленным списком*

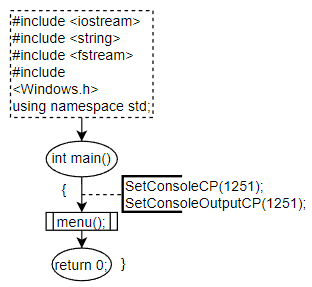
Следует заметить, что для работы с однонаправленным и двунаправленным списками структура List принимает одинаковый вид - она имеет поле int size, отвечающее за размерность списка, а также ключевые поля Point\* head и Point\* tail - указатели на первый и последний элементы списка.

Структура Point в свою очередь имеет различия:

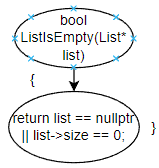
* Для однонаправленного списка она имеет ключевое поле string key, а также указатель на следующий элемент списка Point\* next
* Для двунаправленного списка она имеет ключевое поле string key, указатель на следующий элемент списка Point\* next и, кроме того, указатель на предыдущий элемент списка Point\* pred.

В этом и заключается различие между односвязным и двусвязным списком - в двунаправленном списке есть возможность перехода к предыдущему элементу, из-за чего повышается удобство работы с данной структурой. В однонаправленном списке мы не можем переходить к предыдущему элементу, из-за чего теряется удобство работы с данной структурой, однако нет необходимости при каждом добавлении или удалении элементов обновлять дополнительный указатель на предыдущий элемент.

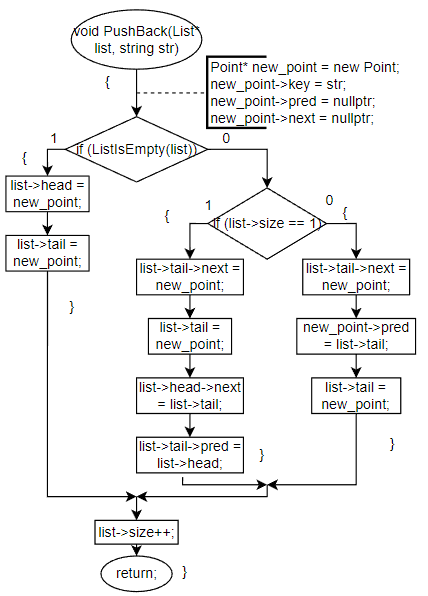
# Блок-схема



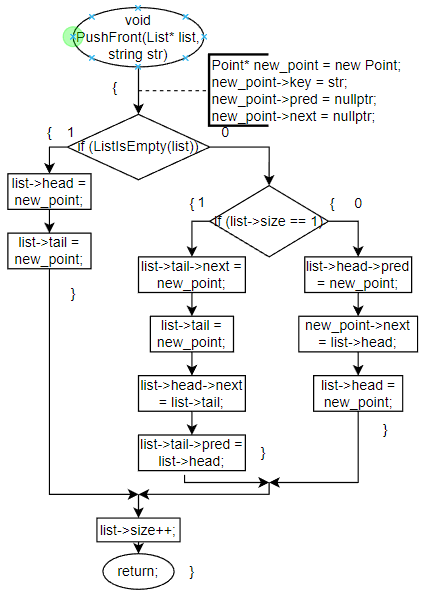
*Рис. 5 - функция main в блок-схеме*



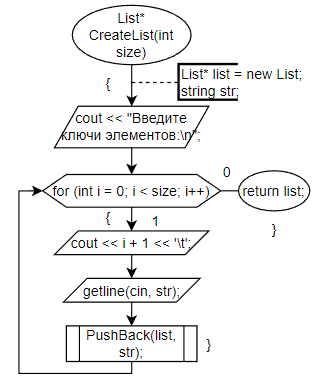
*Рис. 6 - функция ListIsEmpty в блок-схеме*



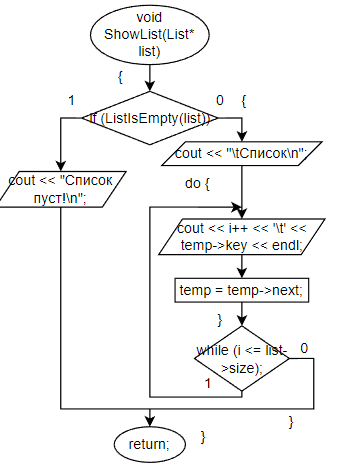
*Рис. 7 - функция PushBack в блок-схеме*



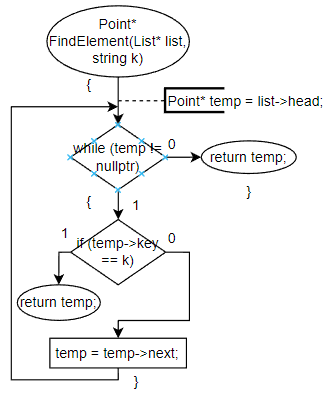
*Рис. 8 - функция PushFront в блок-схеме*



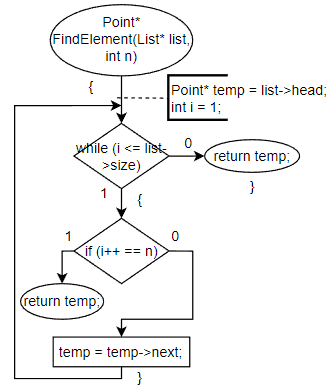
*Рис. 9 - функция CreateList в блок-схеме*



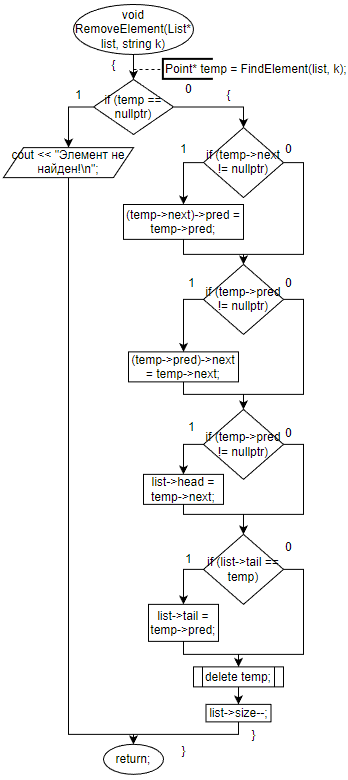
*Рис. 10 - функция ShowList в блок-схеме*



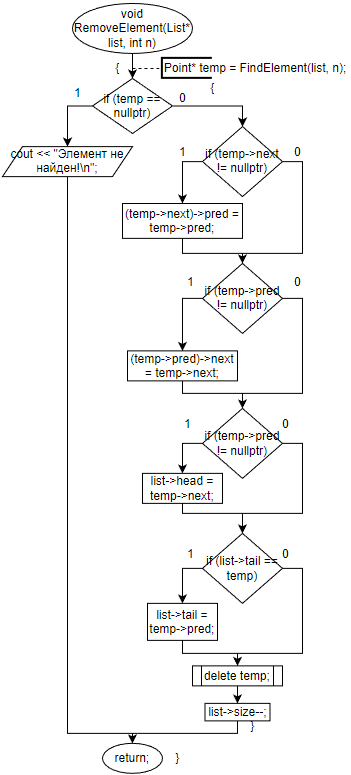
*Рис. 11 - функция FindElement в блок-схеме*



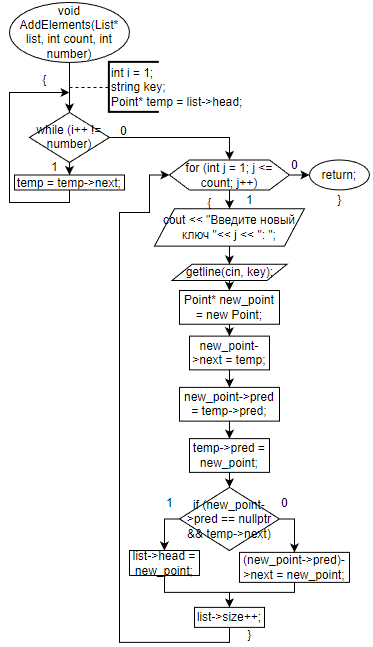
*Рис. 12 - функция FindElement в блок-схеме*



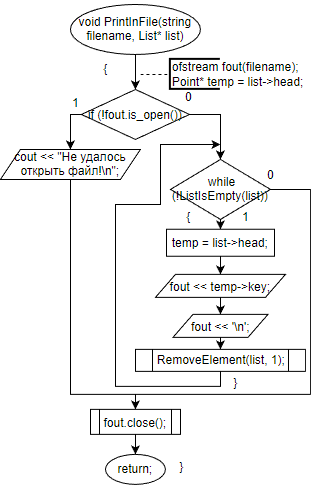
*Рис. 13 - функция RemoveElement в блок-схеме*



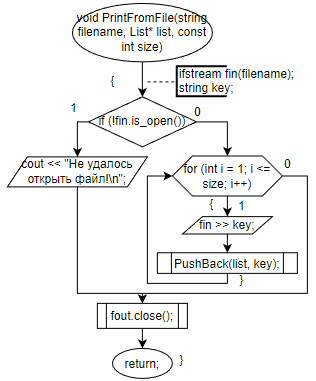
*Рис. 14 - функция RemoveElement в блок-схеме*



*Рис. 15 - функция AddElements в блок-схеме*

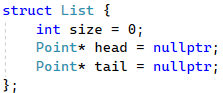
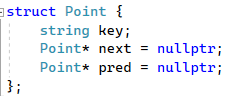
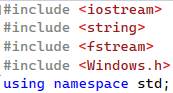


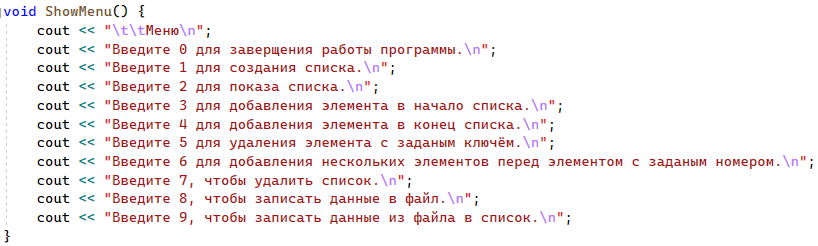
*Рис. 16 - функция PrintInFile в блок-схеме*

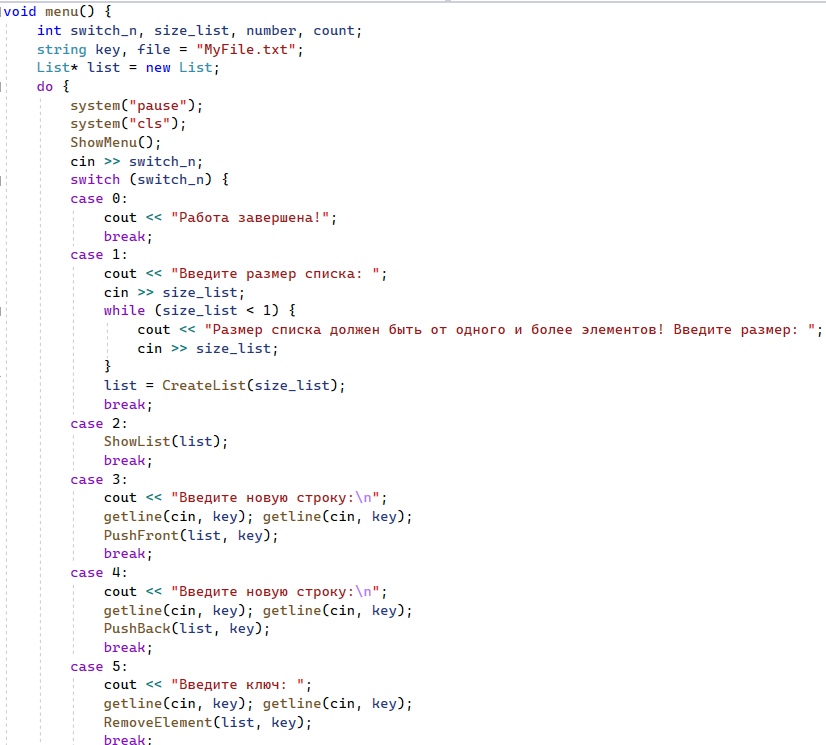


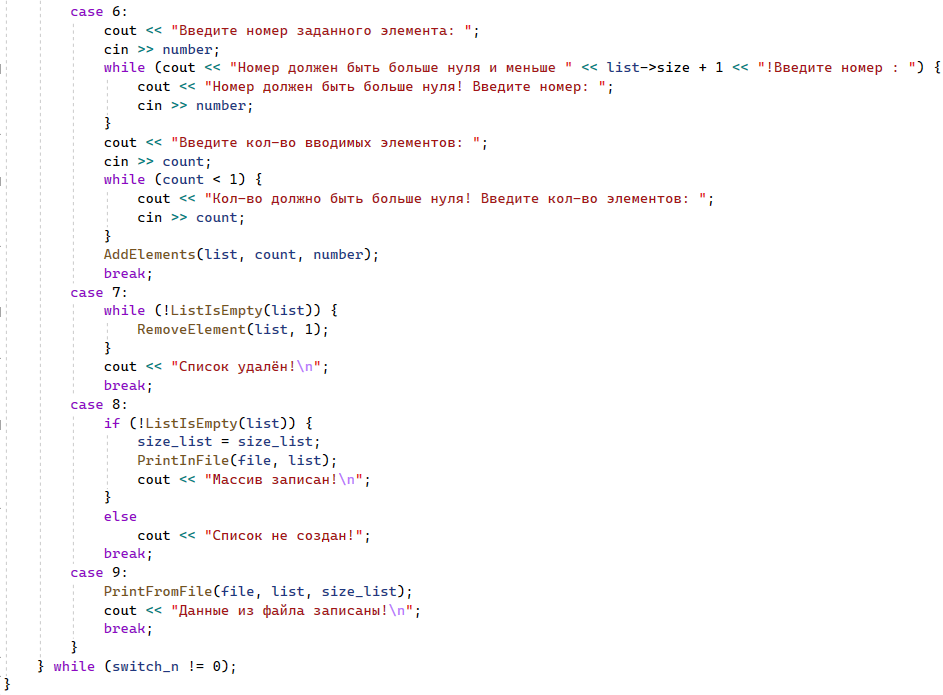
*Рис. 17 - функция PrintFromFile в блок-схеме*

# Программный код

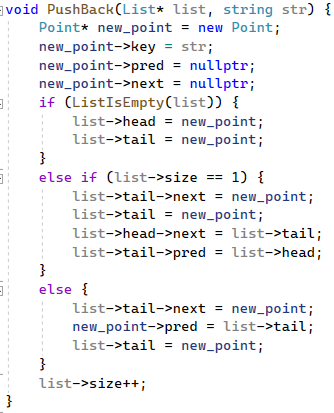


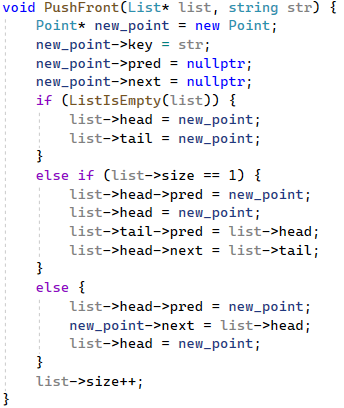


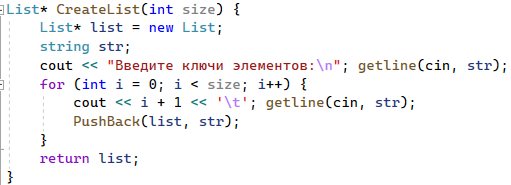


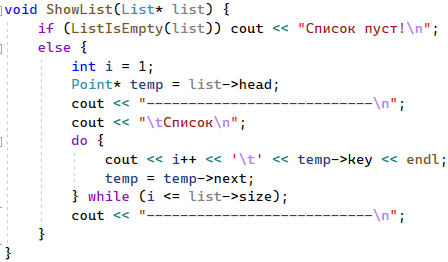


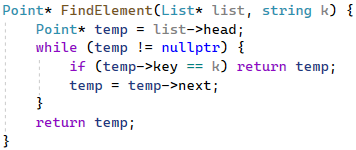


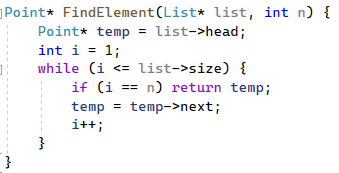


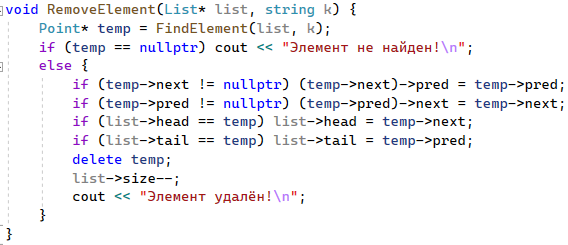


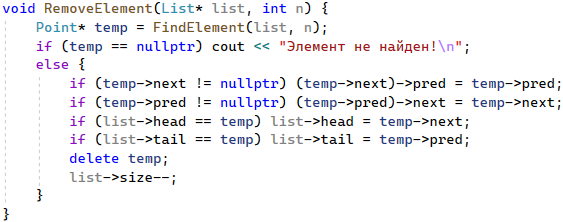


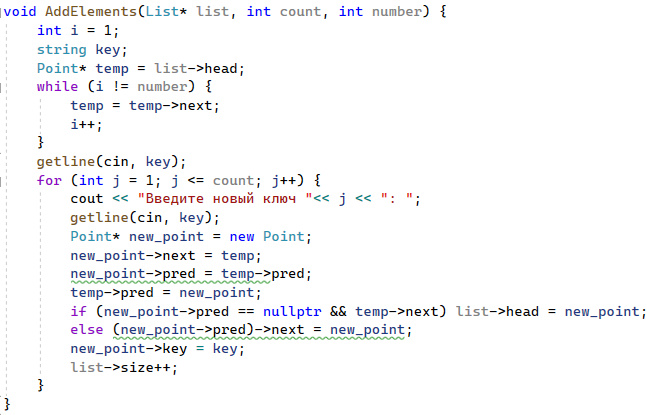


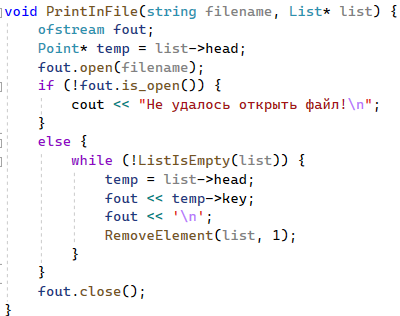


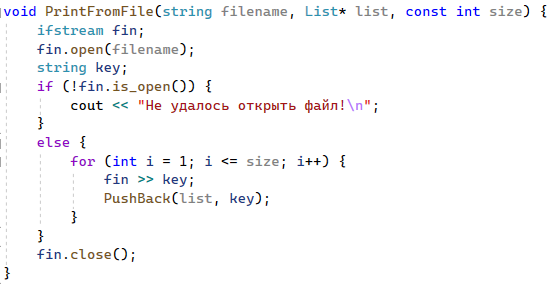


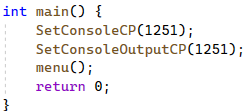












# Работа программы

