Пермский национальный исследовательский политехнический университет ПНИПУ

Отчёт по лабораторной работе на тему “Cписки, стеки, очереди”

Выполнил студент группы РИС-23-3Б:

Смирнов Андрей Сергеевич

Проверила:

Доцент кафедры ИТАС

О.А. Полякова

2024

1. Постановка задачи:
2. Создание списка.
3. Добавление элемента в список
4. Удаление элемента из списка (в соответствии со своим вариантом).
5. Печать списка.
6. Запись списка в файл.
7. Уничтожение списка.
8. Восстановление списка из файла.
9. Анализ задачи:

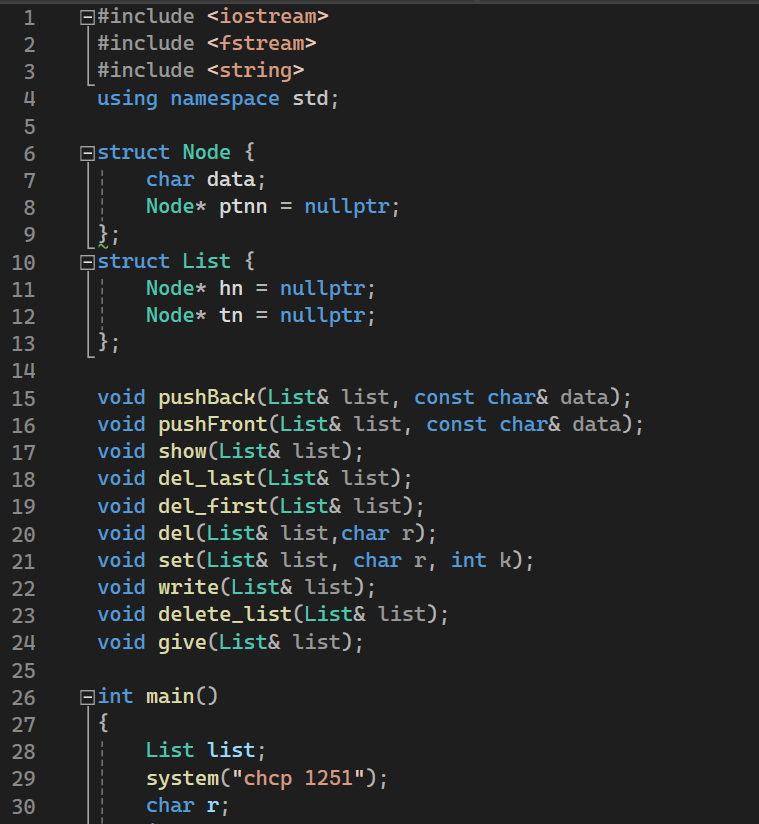
Необходимо создать список, затем заполнить его n количеством знаков, вводимых пользователем. Ввод будет обрабатываться через функцию.

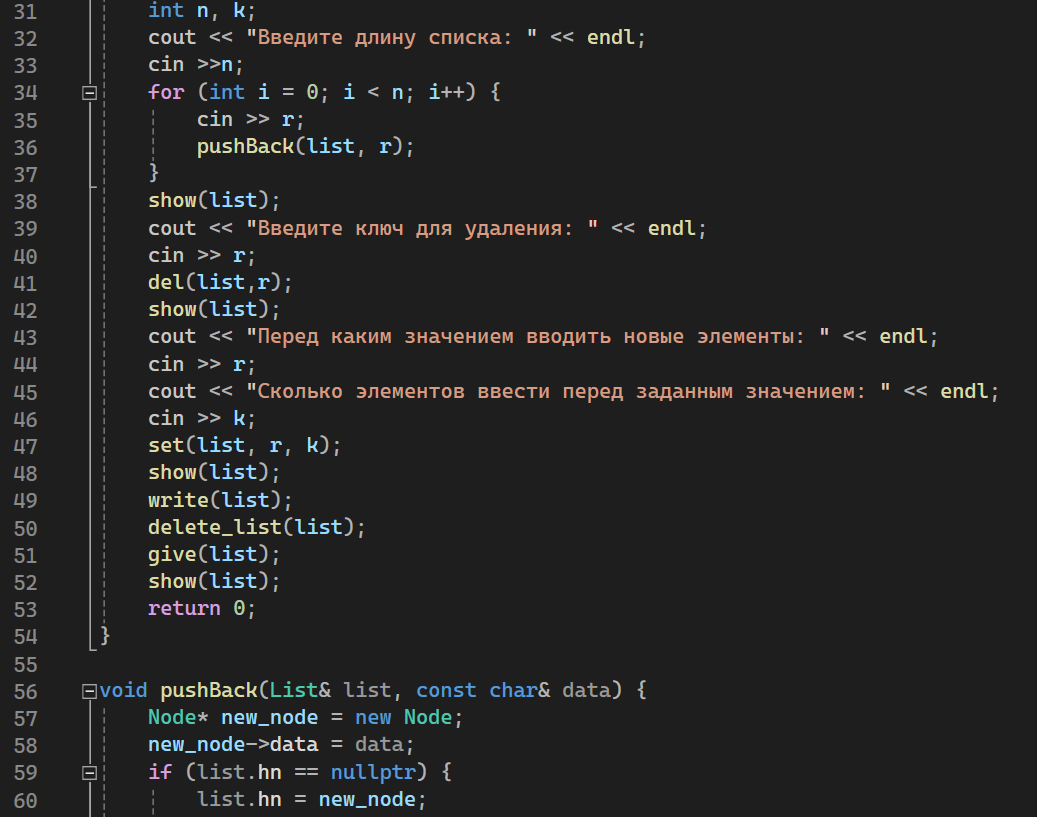
Для удаления блока по значению пройдём по значениям от головы списка до последнего блока, если блок с заданным ключом не будет найден, вывести что такого в списке нет. Удаление происходит так: голова доходит до нужного блока, указатель с головы перекидывается на следующий, блок удаляется. Запись файла происходит, как и вывод через функцию, с первого блока списка в файл записываются значения.

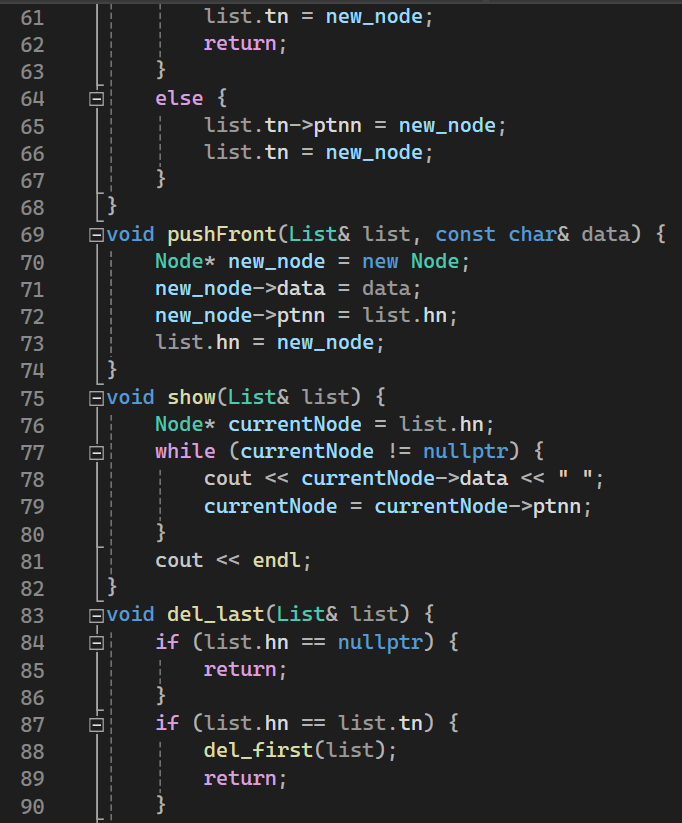
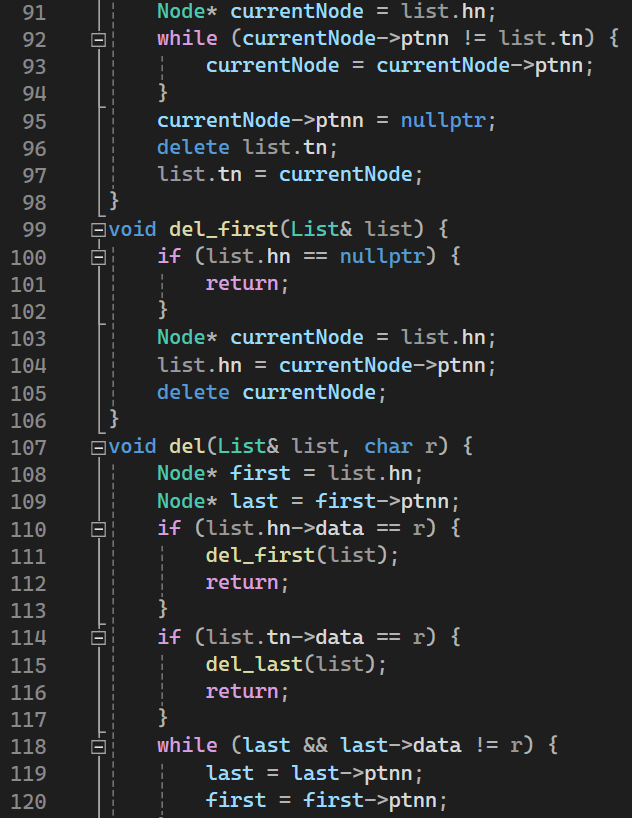
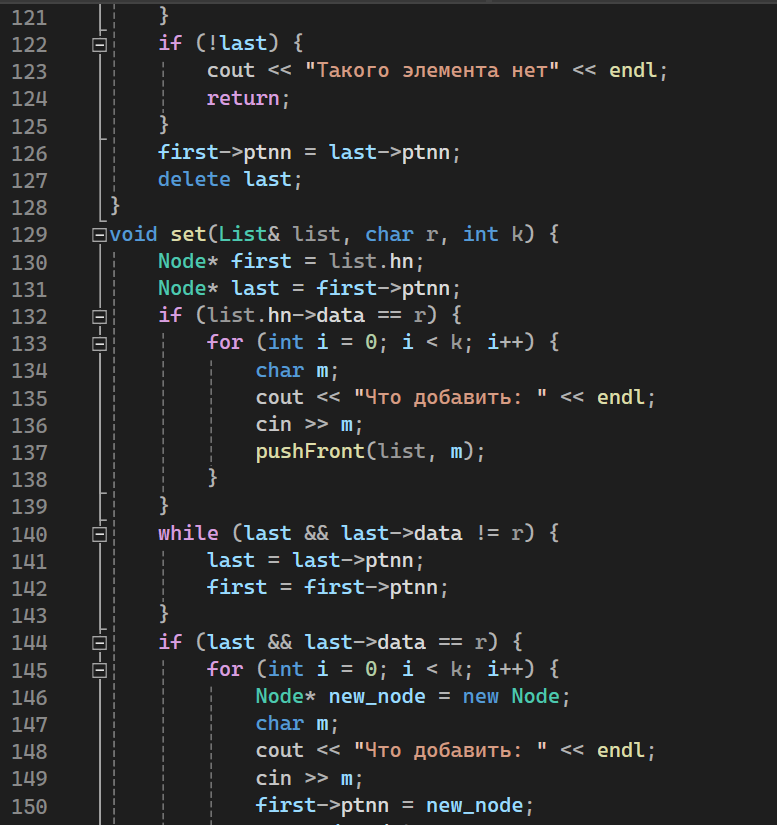
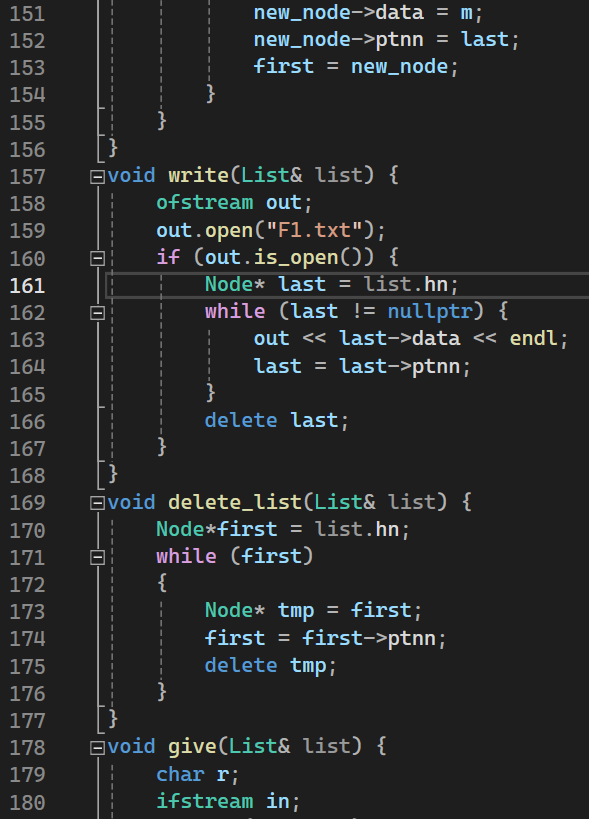
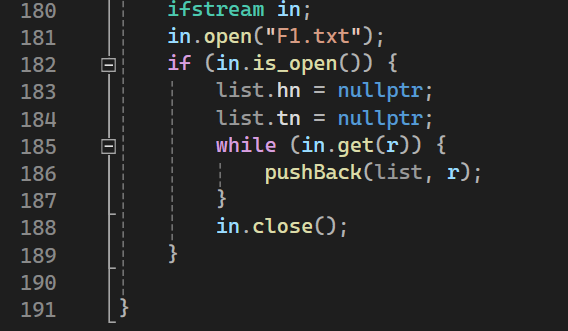
Удаление списка похоже на вывод: с первого блока удаляется блок, а следующий блок приравнивается к временной переменной, указывающей на следующий блок.

Восстановление списка происходит построчно: из файла построчно берётся значение и присваивается блоку списка.

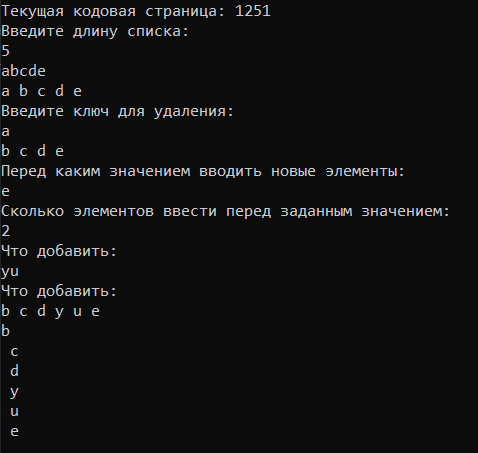
1. Блок-схема:
2. Код:

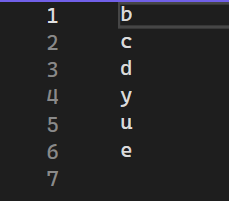




Результат работы:





6)Вывод: Списки удобны для изменения данных в середине списка, они быстрее обрабатываются по сравнению с массивами, но сложнее в создании.

7) GitHub:

<https://github.com/Andr0medA007/Labs>

Двумерный список:

1. Постановка задачи:
2. Создание списка.
3. Добавление элемента в список
4. Удаление элемента из списка (в соответствии со своим вариантом).
5. Печать списка.
6. Запись списка в файл.
7. Уничтожение списка.
8. Восстановление списка из файла.
9. Анализ задачи:

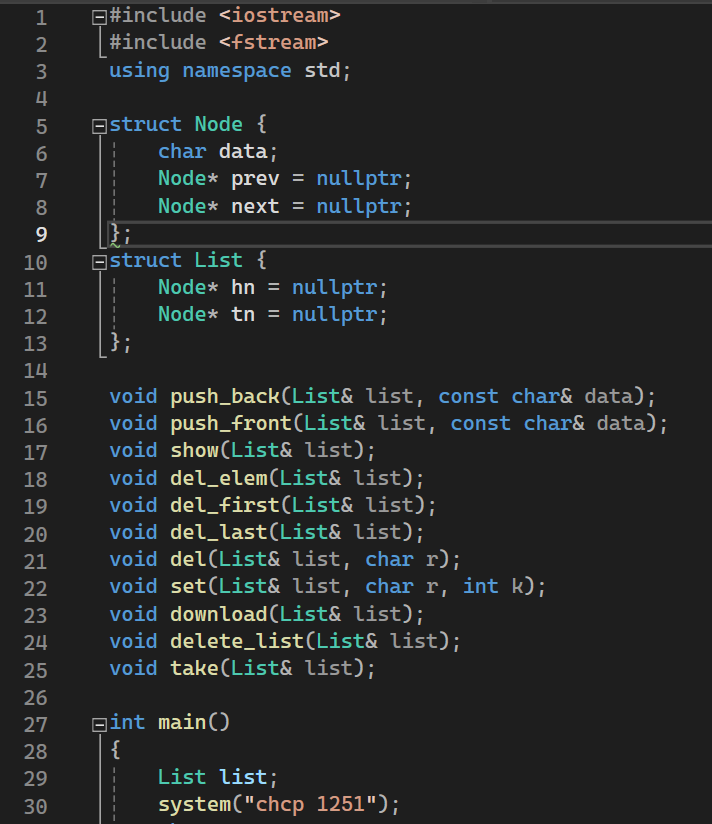
Необходимо создать список, затем заполнить его n количеством знаков, вводимых пользователем. Ввод будет обрабатываться через функцию.

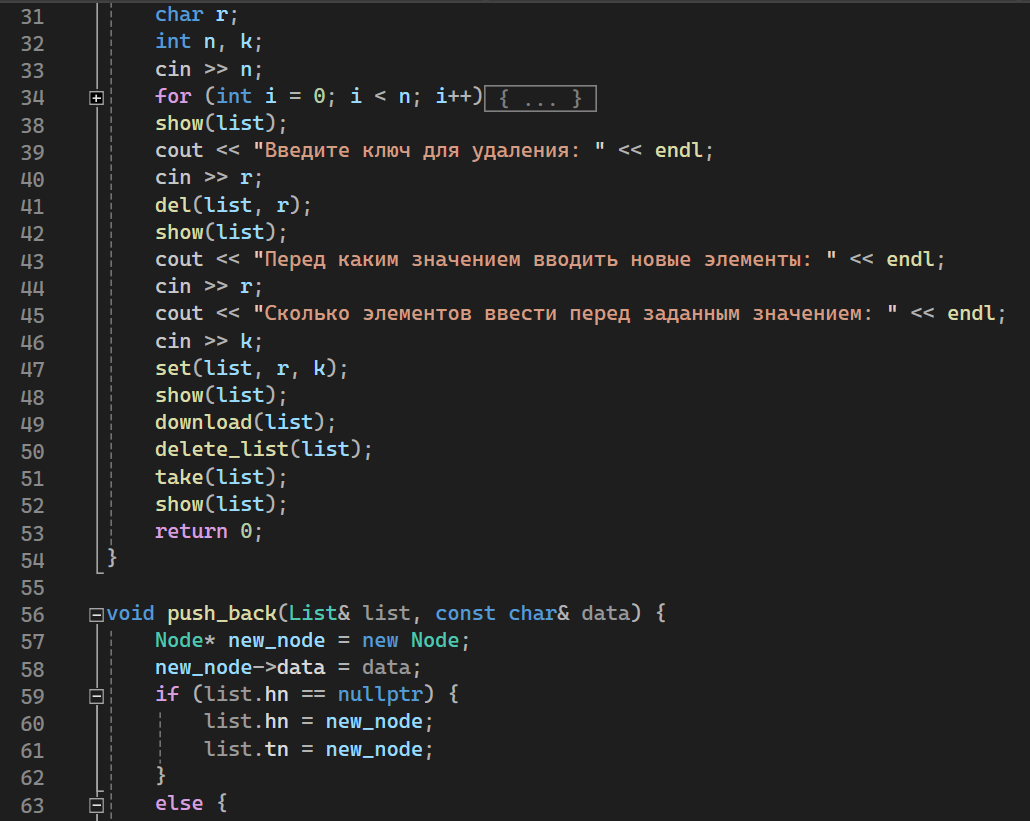
Для удаления блока по значению пройдём по значениям от головы списка до последнего блока, если блок с заданным ключом не будет найден, вывести что такого в списке нет. Удаление происходит так: голова доходит до нужного блока, указатель с головы перекидывается на следующий, блок удаляется. Запись файла происходит, как и вывод через функцию, с первого блока списка в файл записываются значения.

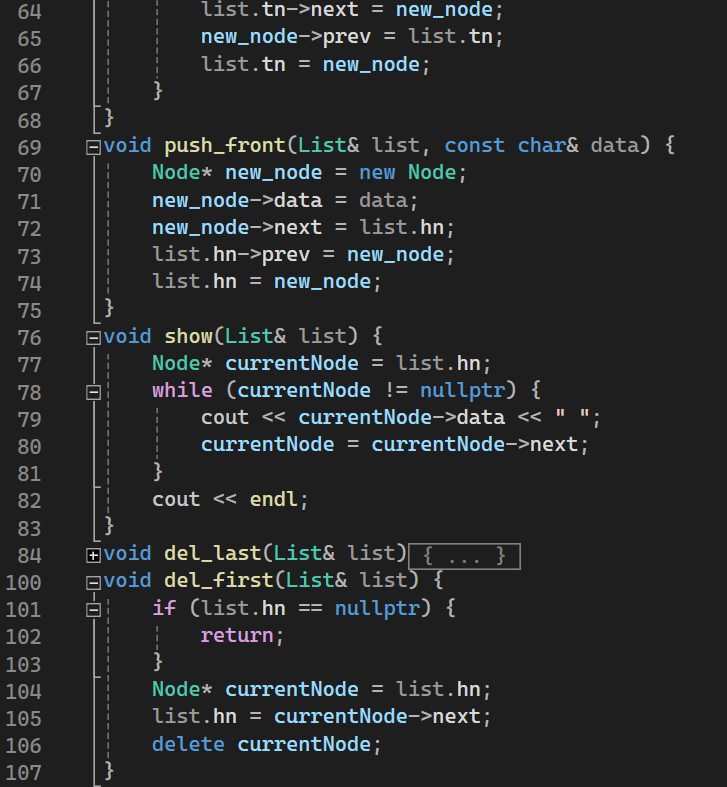
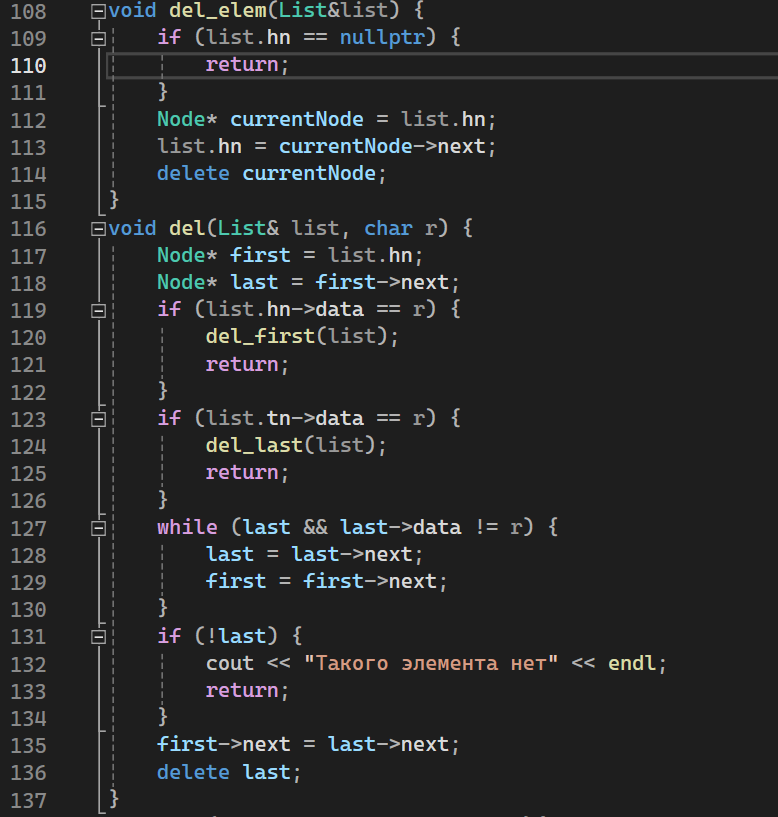
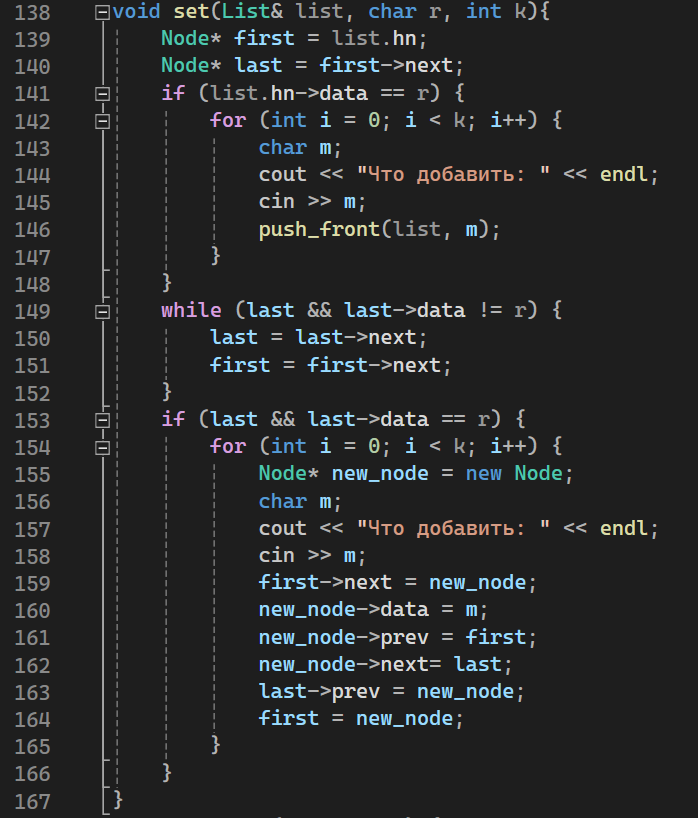
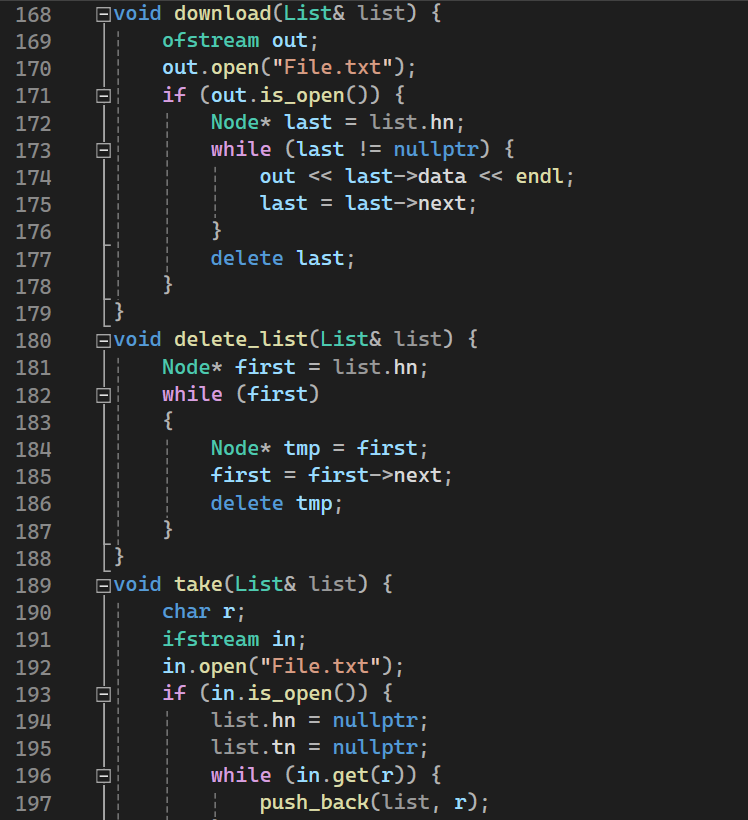
Удаление списка похоже на вывод: с первого блока удаляется блок, а следующий блок приравнивается к временной переменной, указывающей на следующий блок.

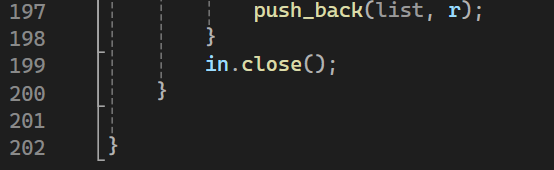
Восстановление списка происходит построчно: из файла построчно берётся значение и присваивается блоку списка.

1. Блок-схема:
2. Код:

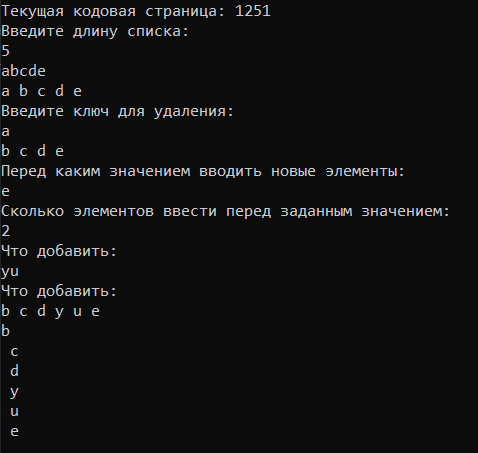


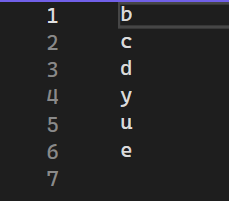




Результат работы:





6)Вывод: Списки удобны для изменения данных в середине списка, они быстрее обрабатываются по сравнению с массивами, но сложнее в создании.

7) GitHub:

<https://github.com/Andr0medA007/Labs>

Стеки:

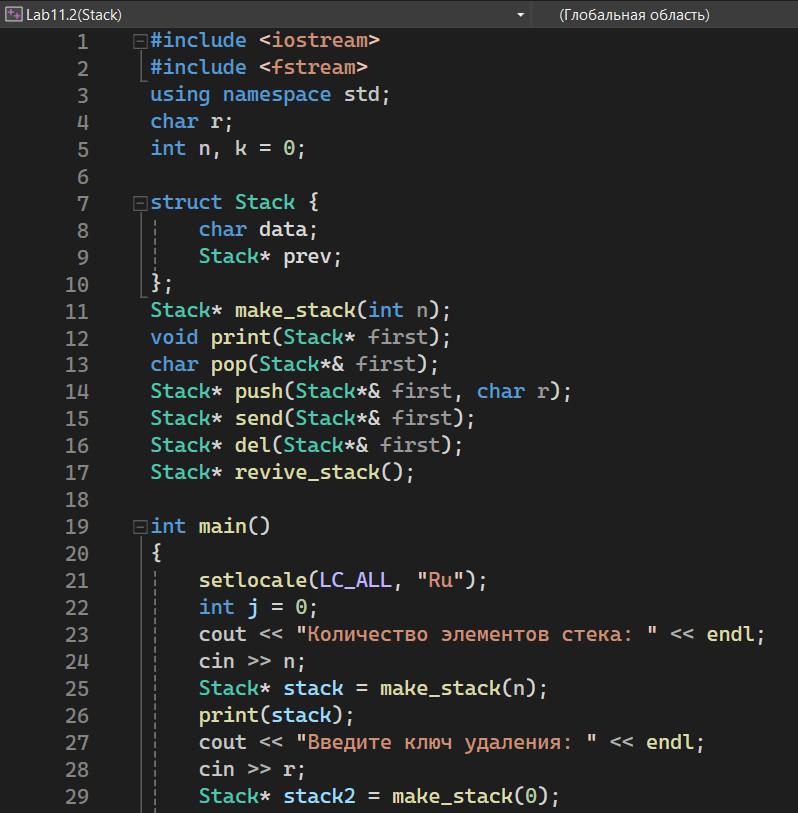
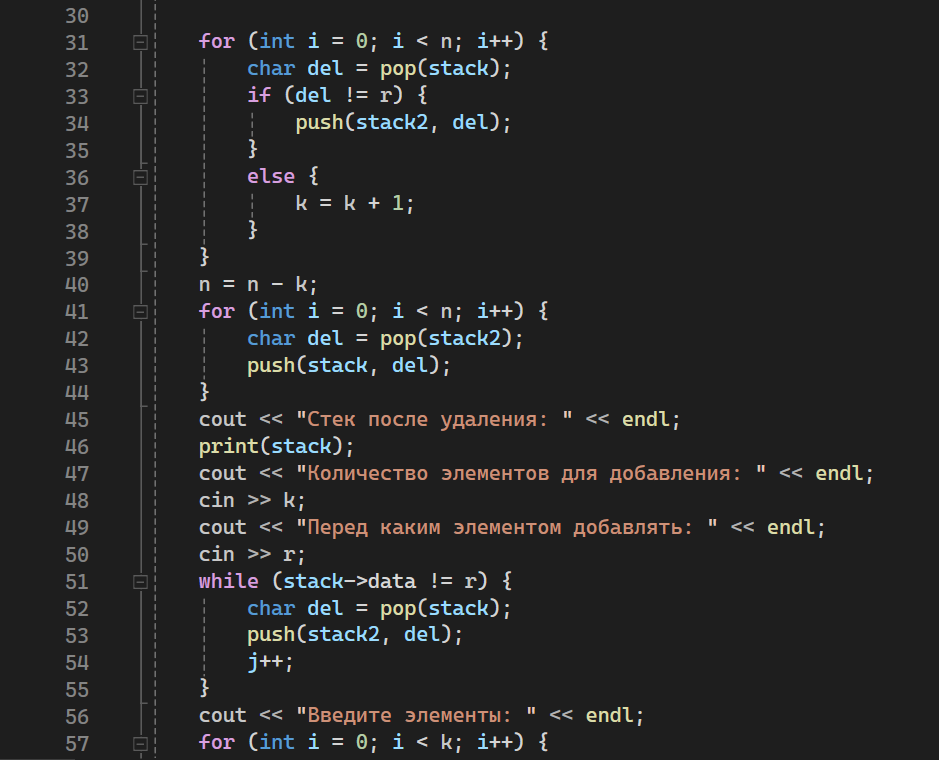
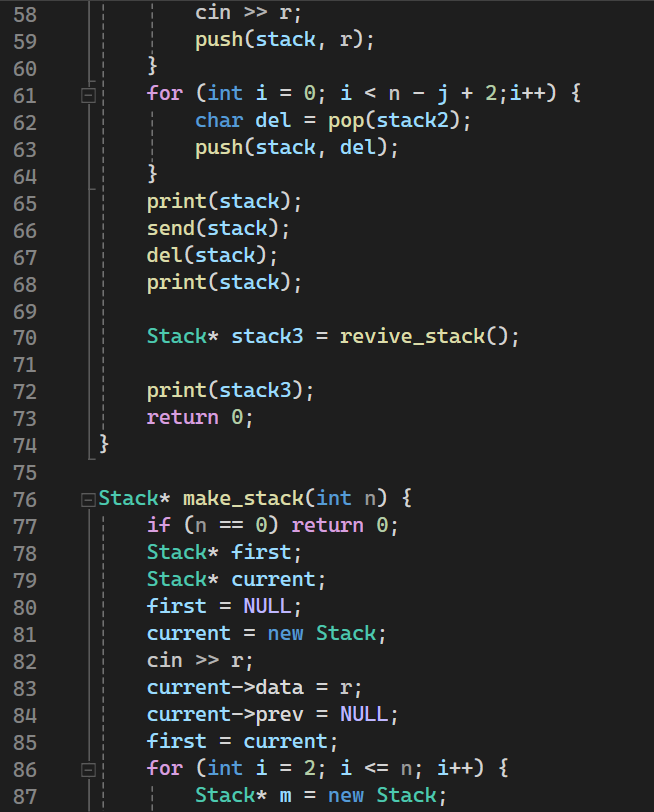
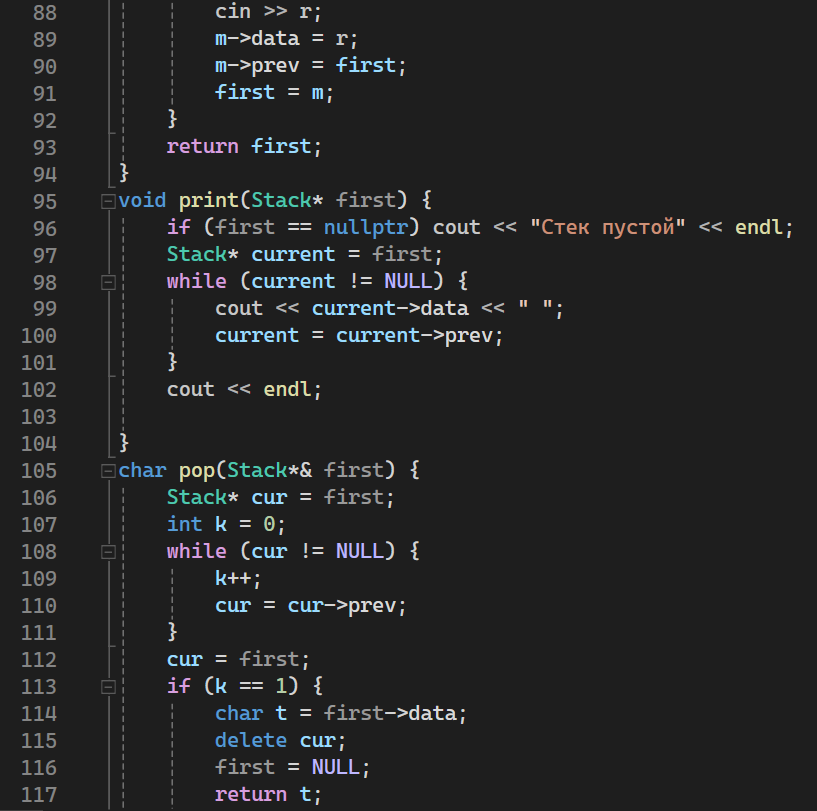
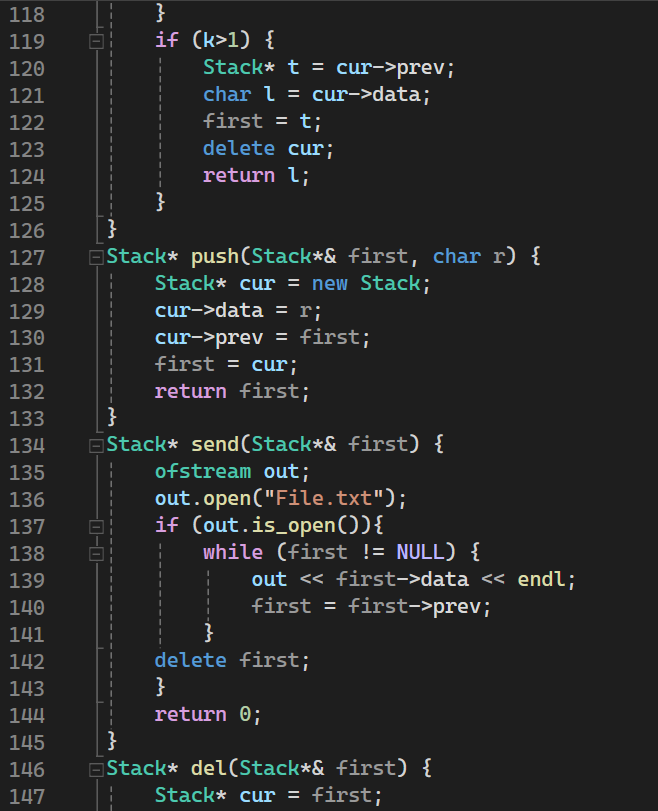
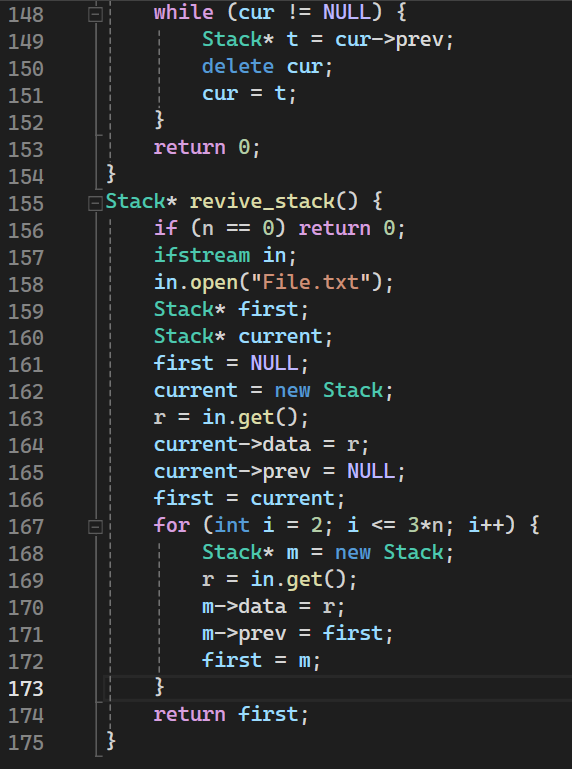
1. Постановка Задачи:

* Создание стека
* Добавление элементов в стек
* Удаление элемента из стека (в соответствии со своим вариантом).
* Печать стека
* Запись списка в стек
* Уничтожение стека
* Восстановление стека из файла.

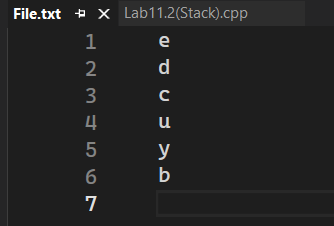
1. Анализ задачи:

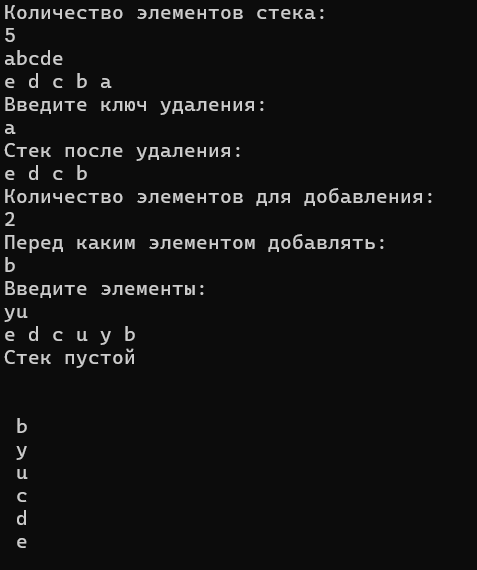
* Создание стека, основанного на односвязном списке
* Обработка стека по принципу первый пришёл, последний уйдёшь
* Постепенное заполнение стека значениями
* Удаление элемента, путём доставания из одного стека в другой всех верхних элементов над нужным, удаление нужного и возврат оставшихся элементов.
* Запись в файл производится по одному с верхнего
* Запись из файла происходит построчно

1. Блок-Схема
2. Код:

1. Результат работы:





1. Вывод: стеки могут использоваться в тех системах, где важен принцип первый зашёл, последний вышел.
2. GitHub: <https://github.com/Andr0medA007/Labs/tree/main/Labs/11%20%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0>

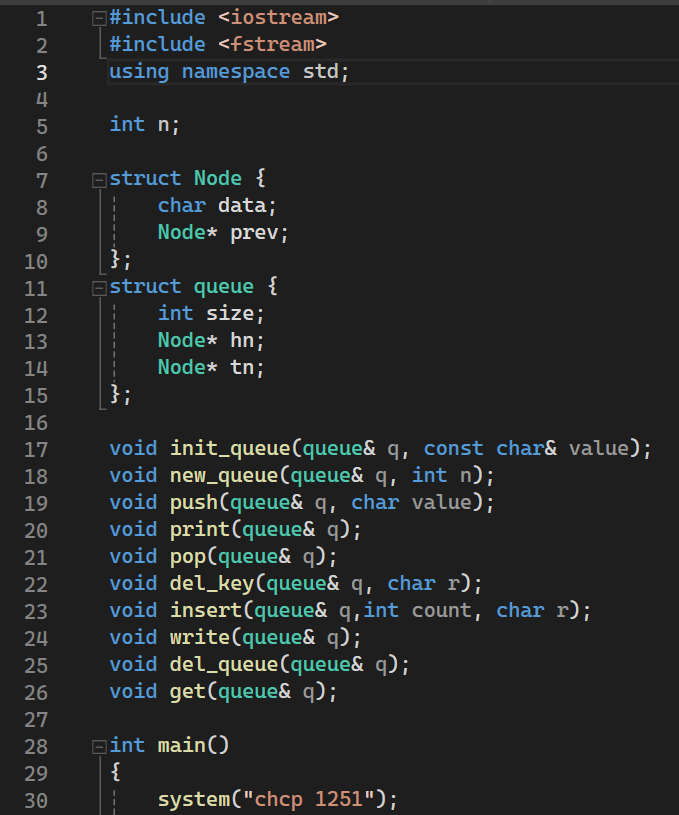
Очередь:

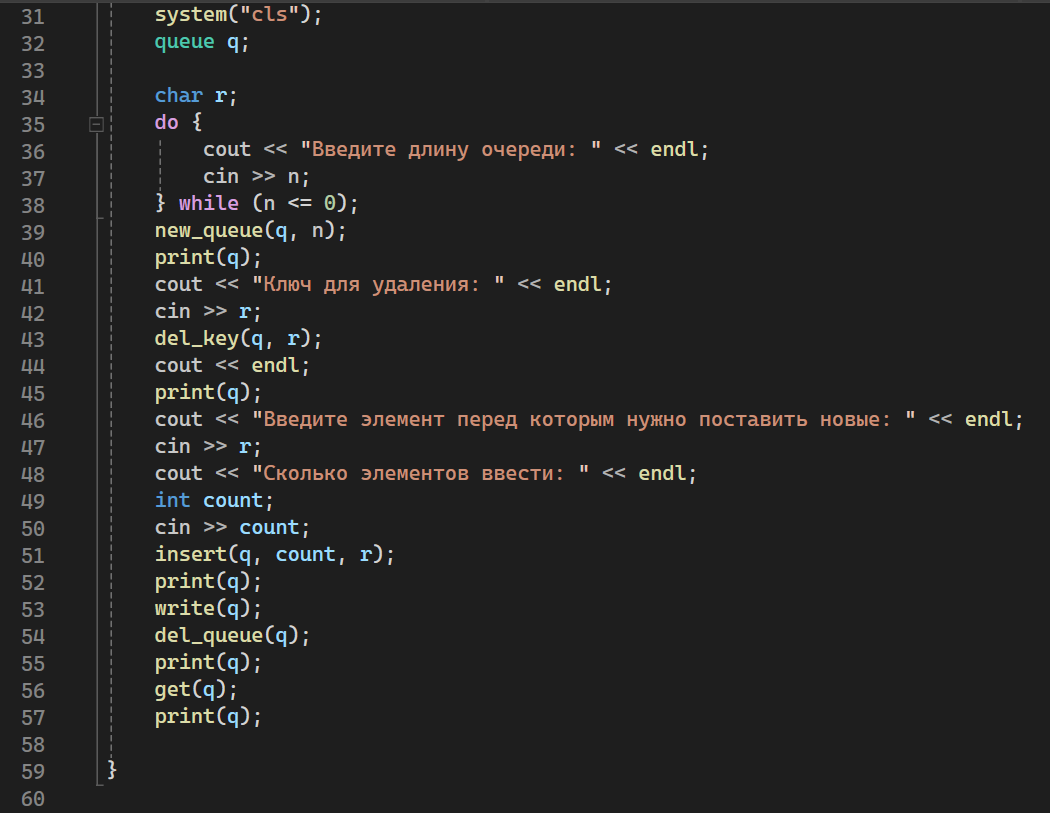
1. Постановка Задачи:

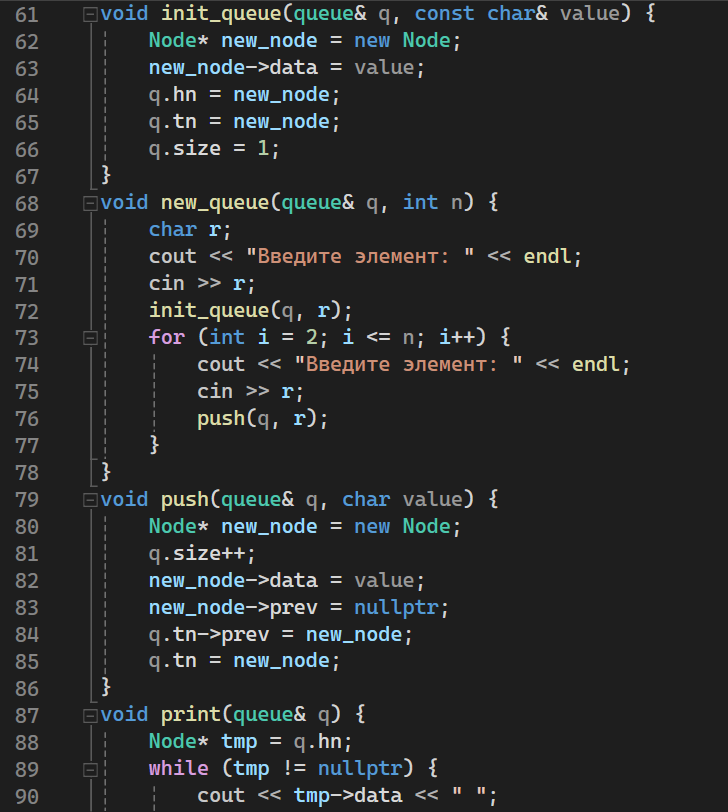
* Создание очереди
* Добавление элементов в очередь
* Удаление элемента из очереди (в соответствии со своим вариантом).
* Печать очереди
* Запись очереди в файл
* Уничтожение очереди
* Восстановление очереди из файла.

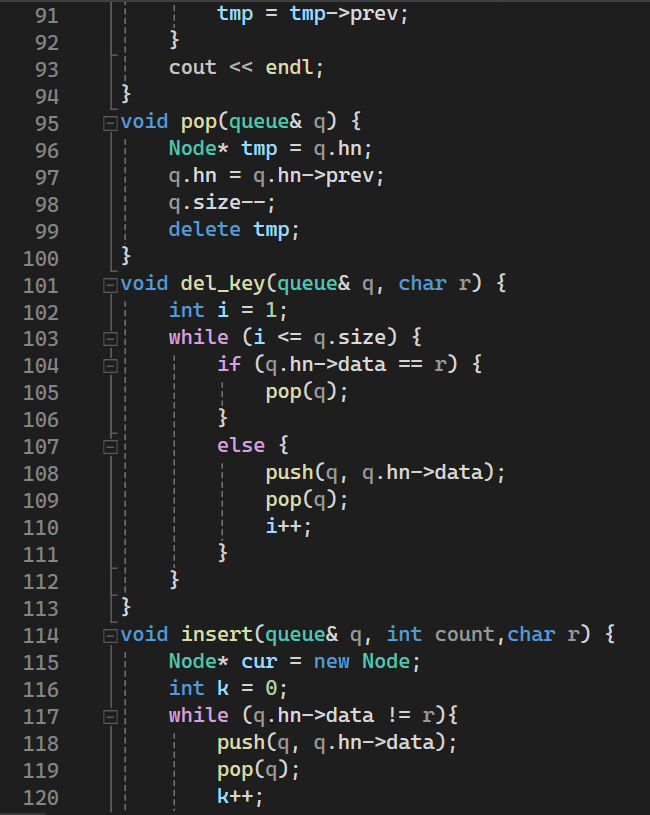
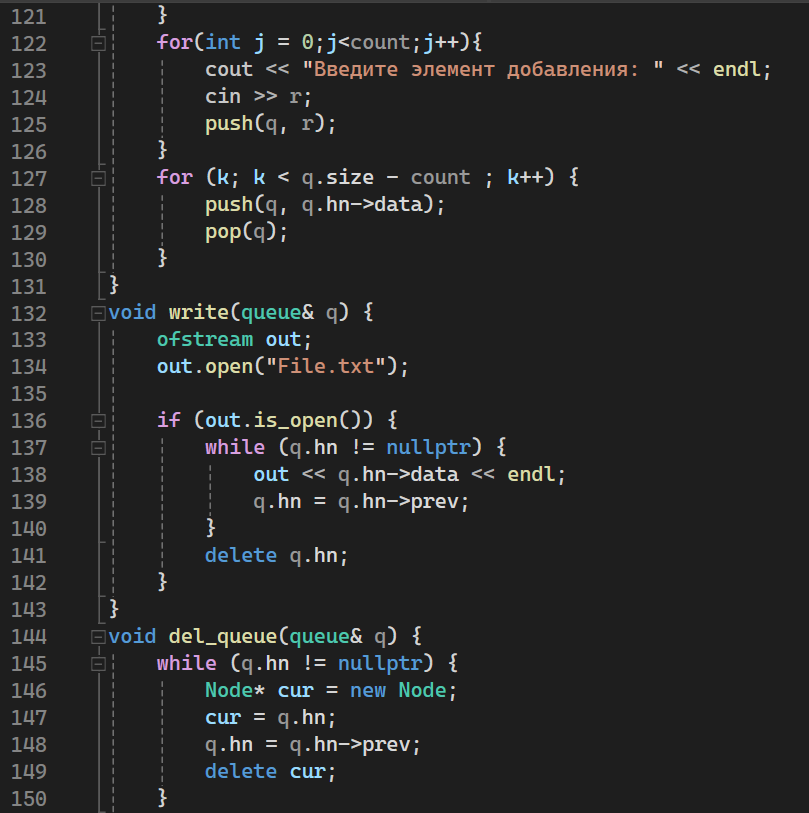
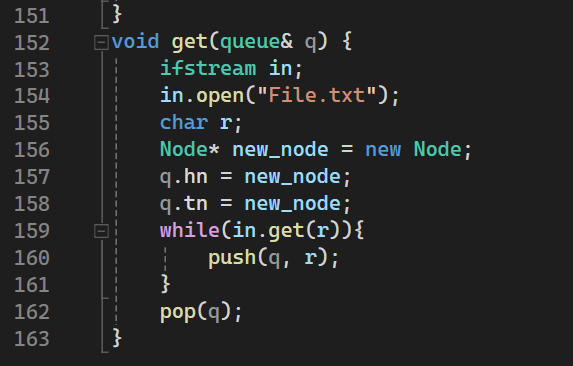
1. Анализ задачи:

* Создание очереди, основанного на односвязном списке
* Обработка стека по принципу первый пришёл, первый уйдёшь
* Постепенное заполнение очереди значениями
* Удаление элемента, путём перемещения в начало всех элементов перед нужным, затем удаление нужного и перемещение оставшихся элементов дальше в начало очереди.
* Запись в файл производится по одному с начала очереди
* Запись из файла происходит построчно

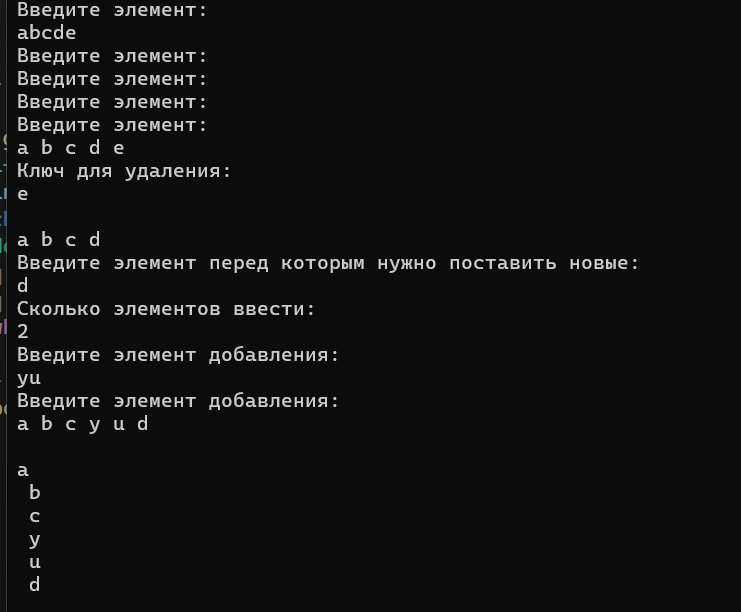
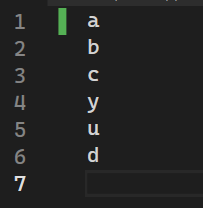
1. Блок-Схема
2. Код: 





1. Результат работы:

1. Вывод: очереди могут использоваться в тех системах, где важен принцип первый зашёл, первый вышел.
2. GitHub: <https://github.com/Andr0medA007/Labs/tree/main/Labs/11%20%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0/Lab11.1(queue)>