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.

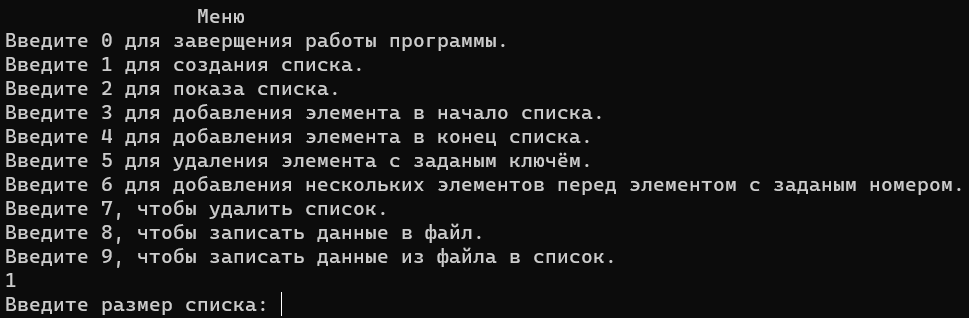
7. Восстановление списка из файла.

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

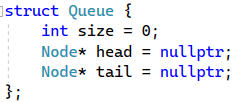
Для удобства работы с программой необходимо создать функцию menu и функцию ShowMenu, которая позволит работать непосредственно с меню программы и завершать работу тогда, когда это нужно. Также, после каждого совершённого действия необходимо очищать консоль с помощью функции system(“cls”).



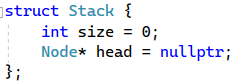
*Рис. 1 - реализация меню программы в консоли*

Также для удобства работы со списком создадим две структуры:

* Структуры очереди и стека

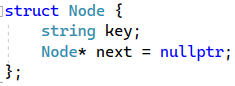


*Рис. 2 - структура Queue для работы с очередью*

**

*Рис. 3 - структура Stack для работы со стеком*

* Структура узла очереди и стека

**

*Рис. 4 - структура Node для работы с очередью и стеком*

Следует заметить, что для работы с однонаправленным и двунаправленным списками структура Node принимает одинаковый вид - она имеет поле string key, которое хранит ключ узла, а также поле Node\* next - указатели на следующий элемент очереди (стека).

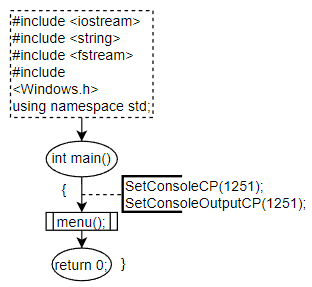
Структуры Queue и Stack в свою очередь имеют различия:

* У обеих структур имеется поле int size = 0 - размер очереди (стека)
* Структуры очереди имеет поля Node\* head и Node\* tail - указатели на первый и последний элементы очереди, а структура стека имеет поле Node\* head - указатель на “верхушку” стека.

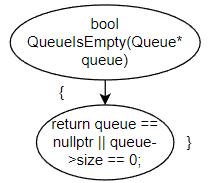
В этом и заключается различие между очередью и стеком:

* очередь представляет собой список элементов, организованных по принципу FIFO (англ. first in — first out, «первым пришёл — первым вышел»), поэтому для работы с ней необходимо добавлять элементы в конец очереди (tail) и удалять элементы из начала (head)
* стек представляет собой список элементов, организованных по принципу FILO (англ. first in — last out, «первым пришёл — последним вышел»), поэтому добавлять и удалять элементы мы будем только из “верхушки нашего стека.

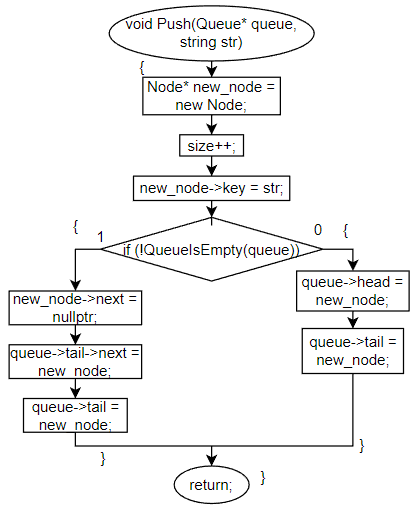
# Блок-схема



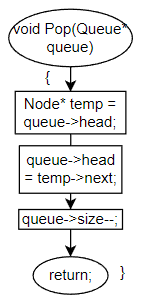
*Рис. 5 - функция main в блок-схеме*



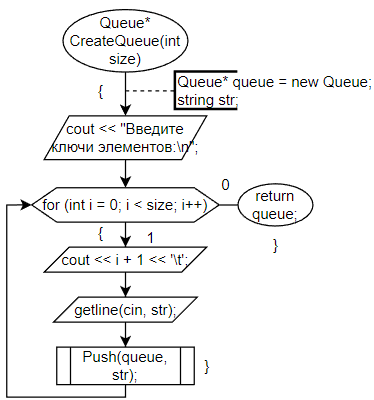
*Рис. 6 - функция QueueIsEmpty в блок-схеме*



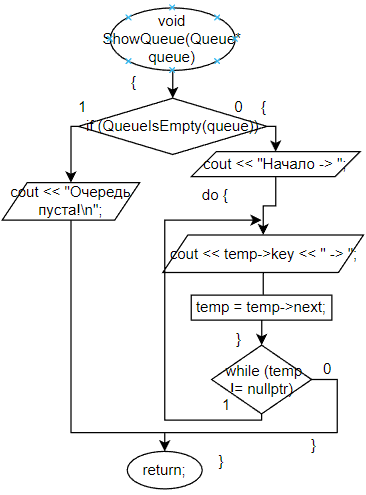
*Рис. 7 - функция Push в блок-схеме*

**

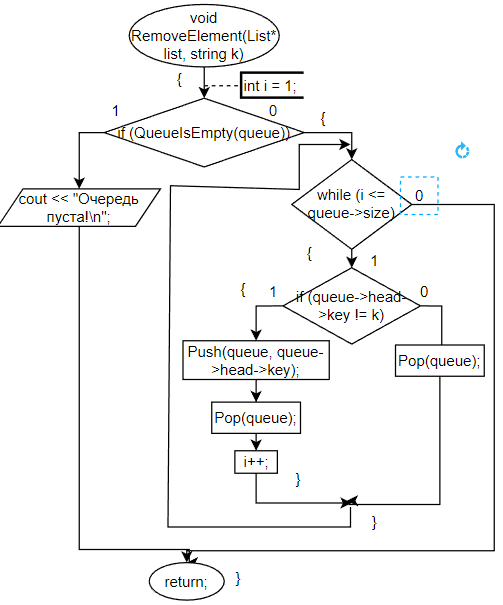
*Рис. 8 - функция Pop в блок-схеме*



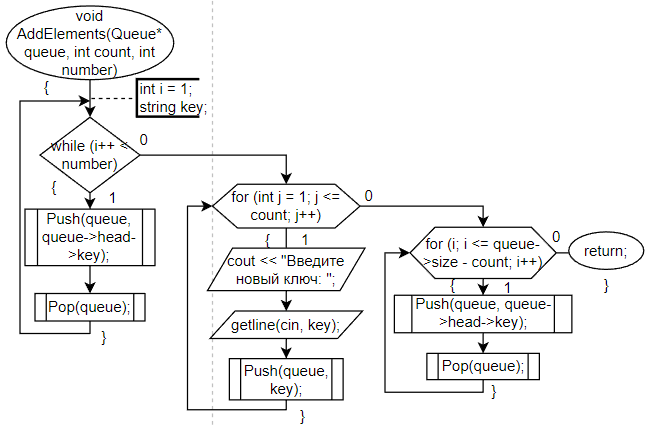
*Рис. 9 - функция CreateQueue в блок-схеме*



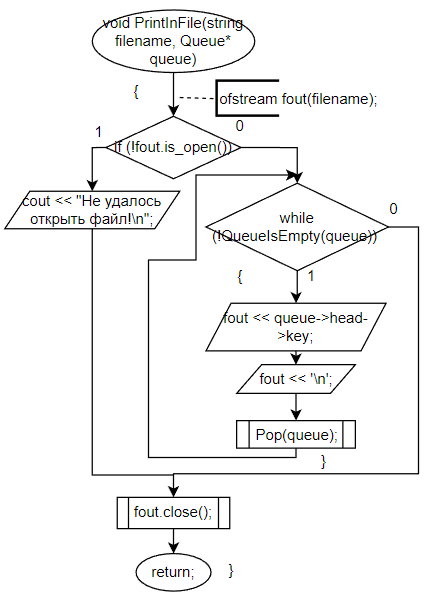
*Рис. 10 - функция ShowQueue в блок-схеме*



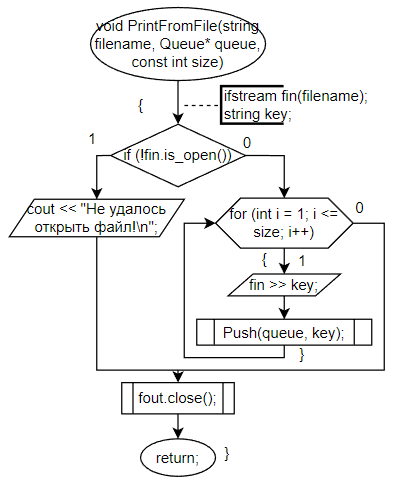
*Рис. 11 - функция RemoveElement в блок-схеме*



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

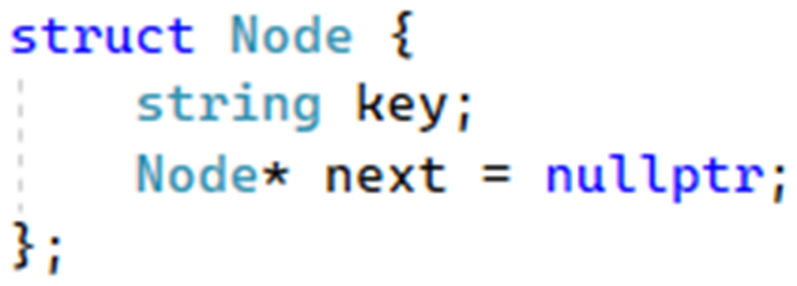
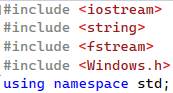


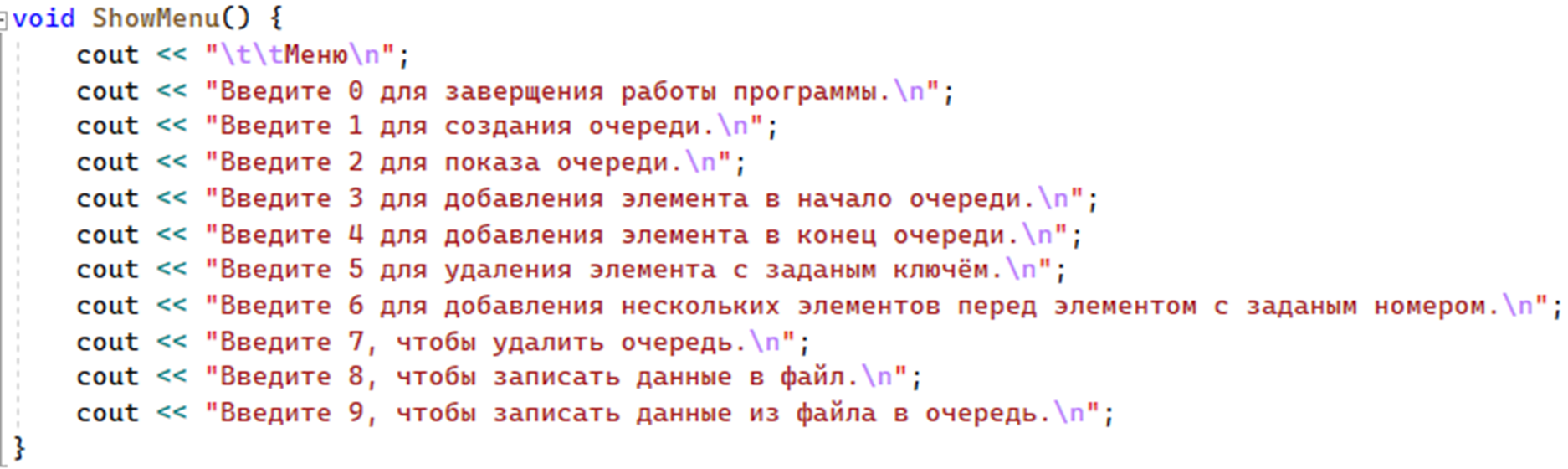
*Рис. 13 - функция PrintInFile в блок-схеме*

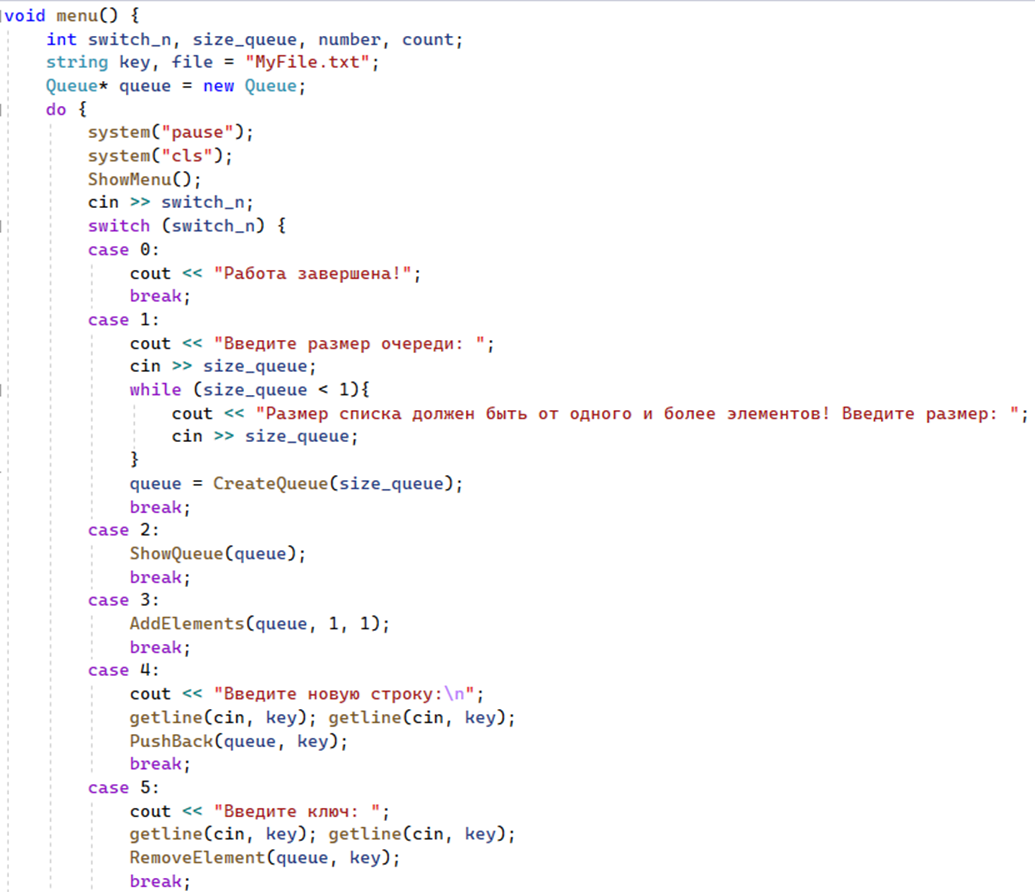


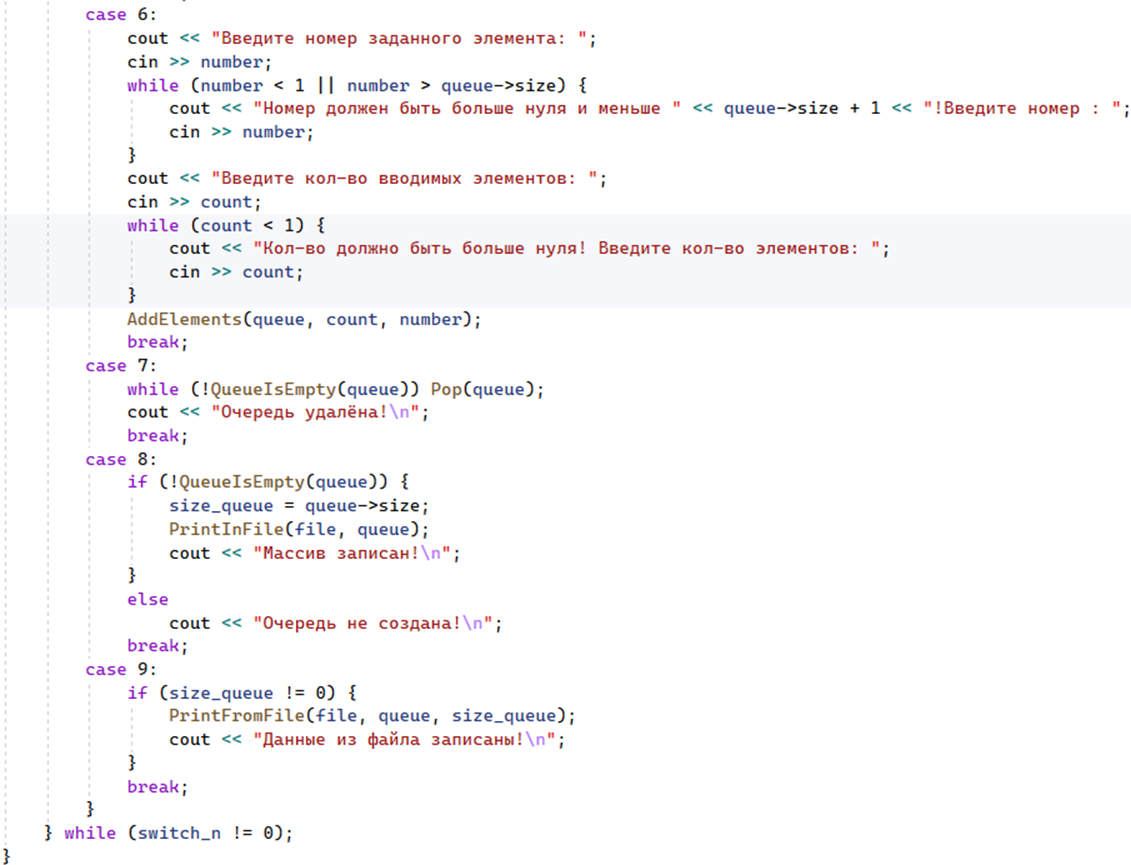
*Рис. 14 - функция PrintFromFile в блок-схеме*

# Программный код

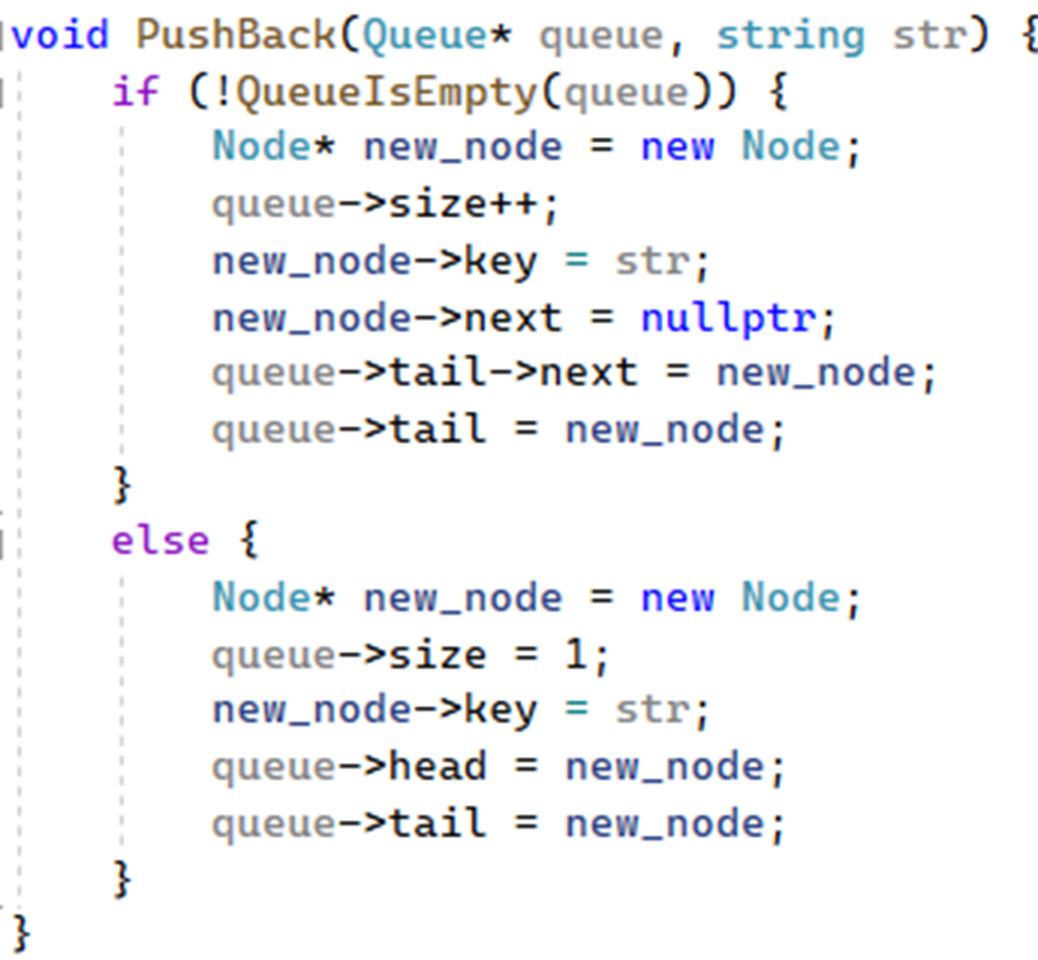


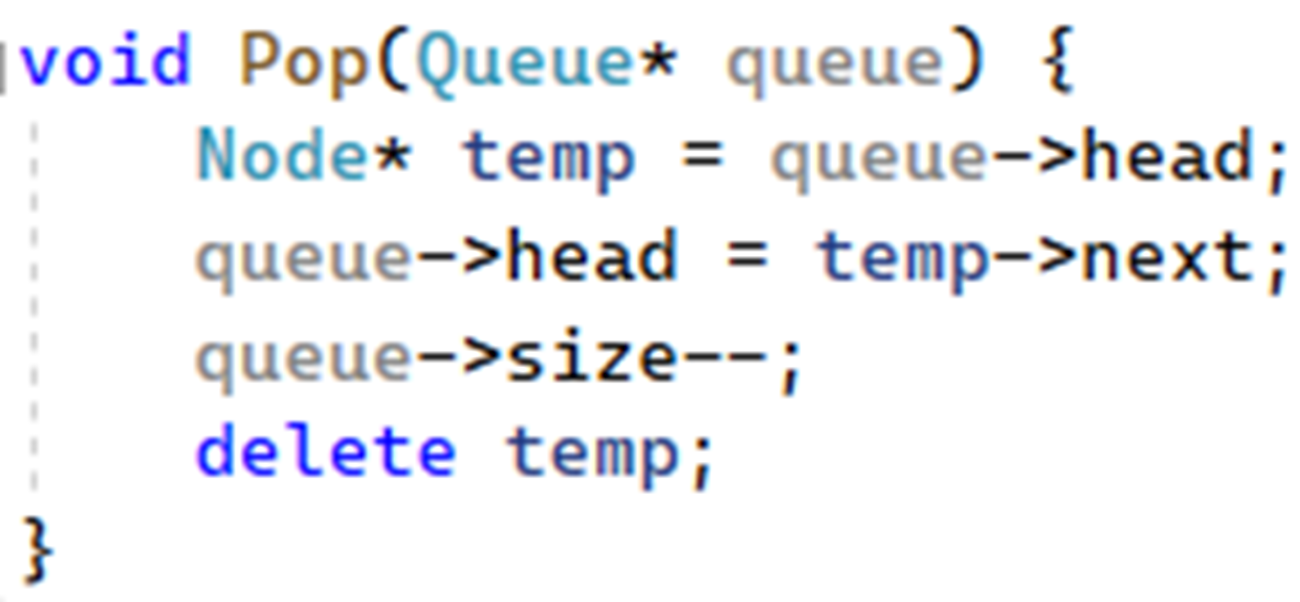
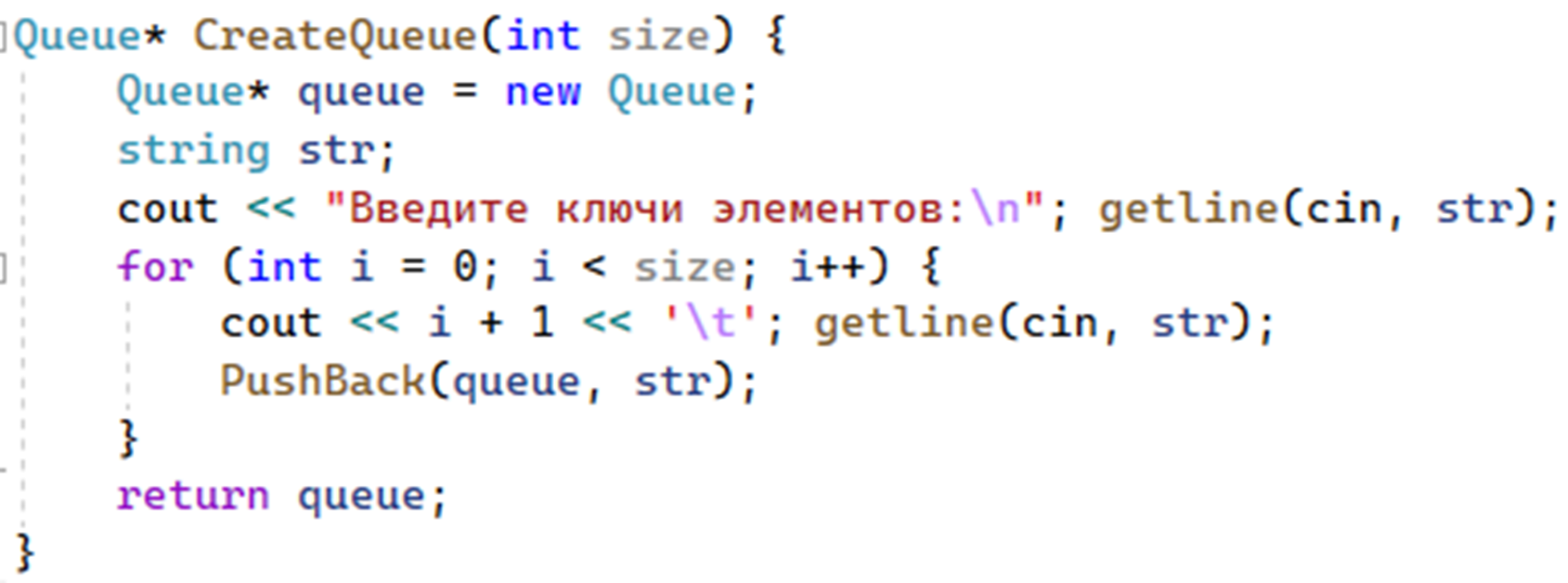


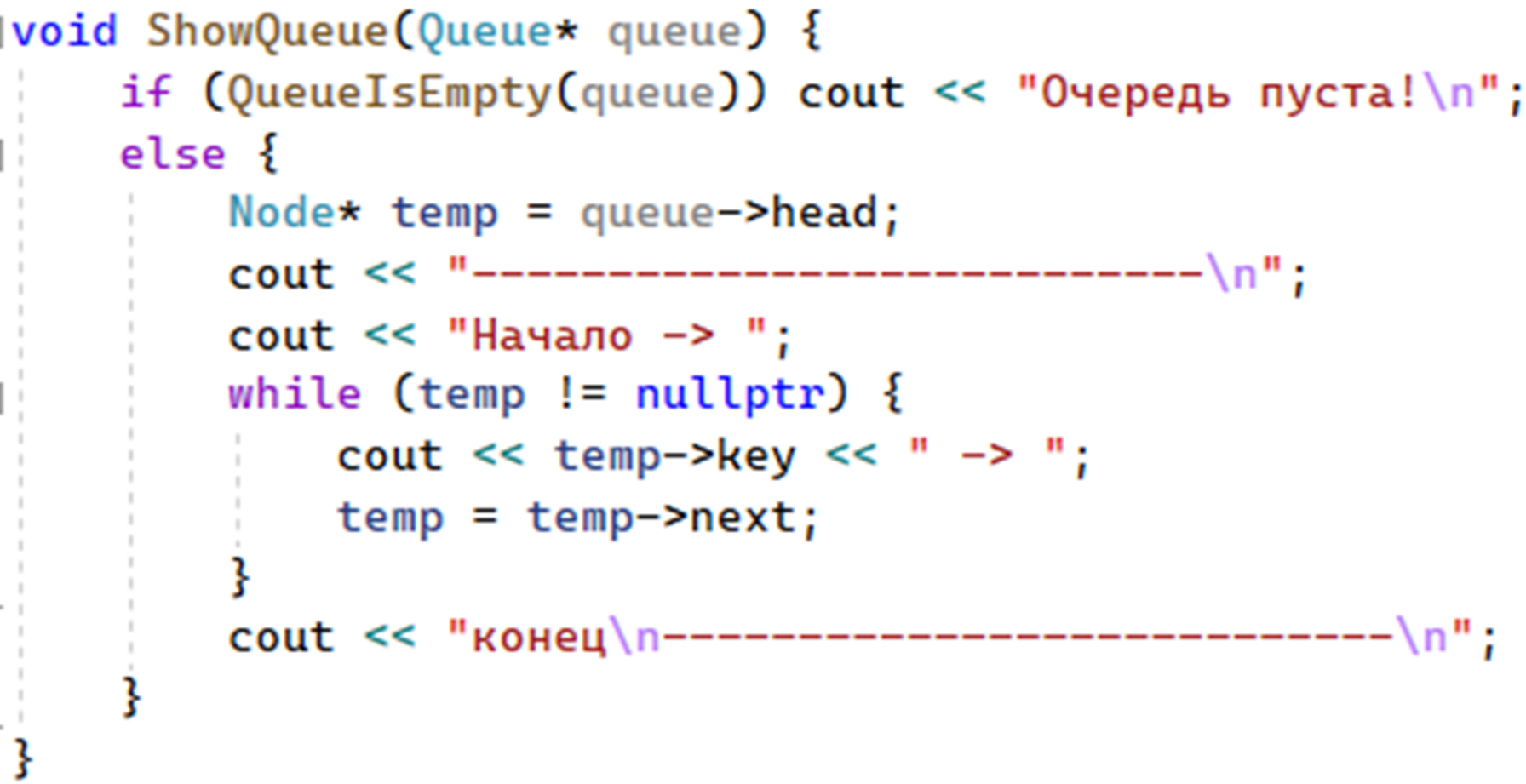


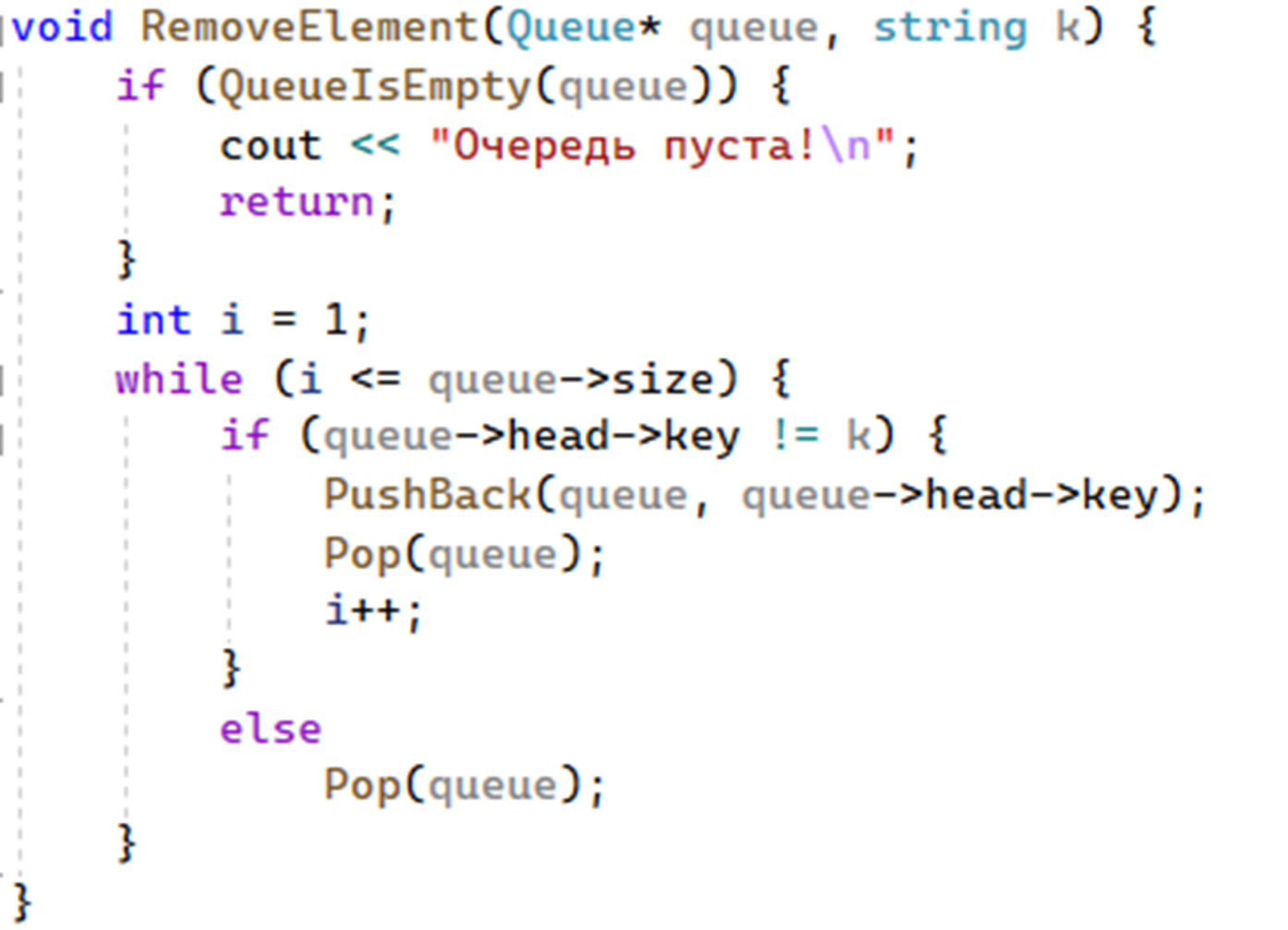


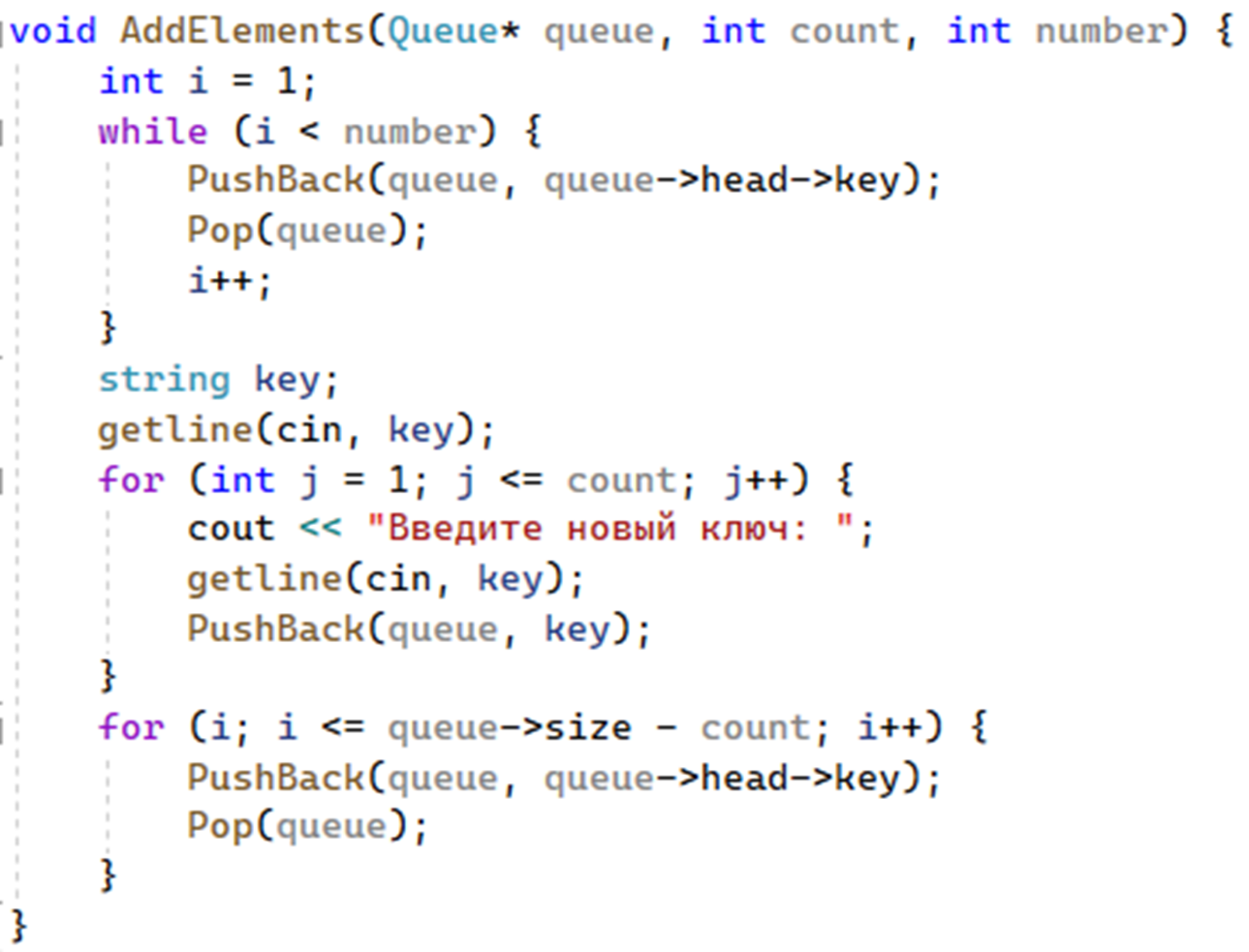




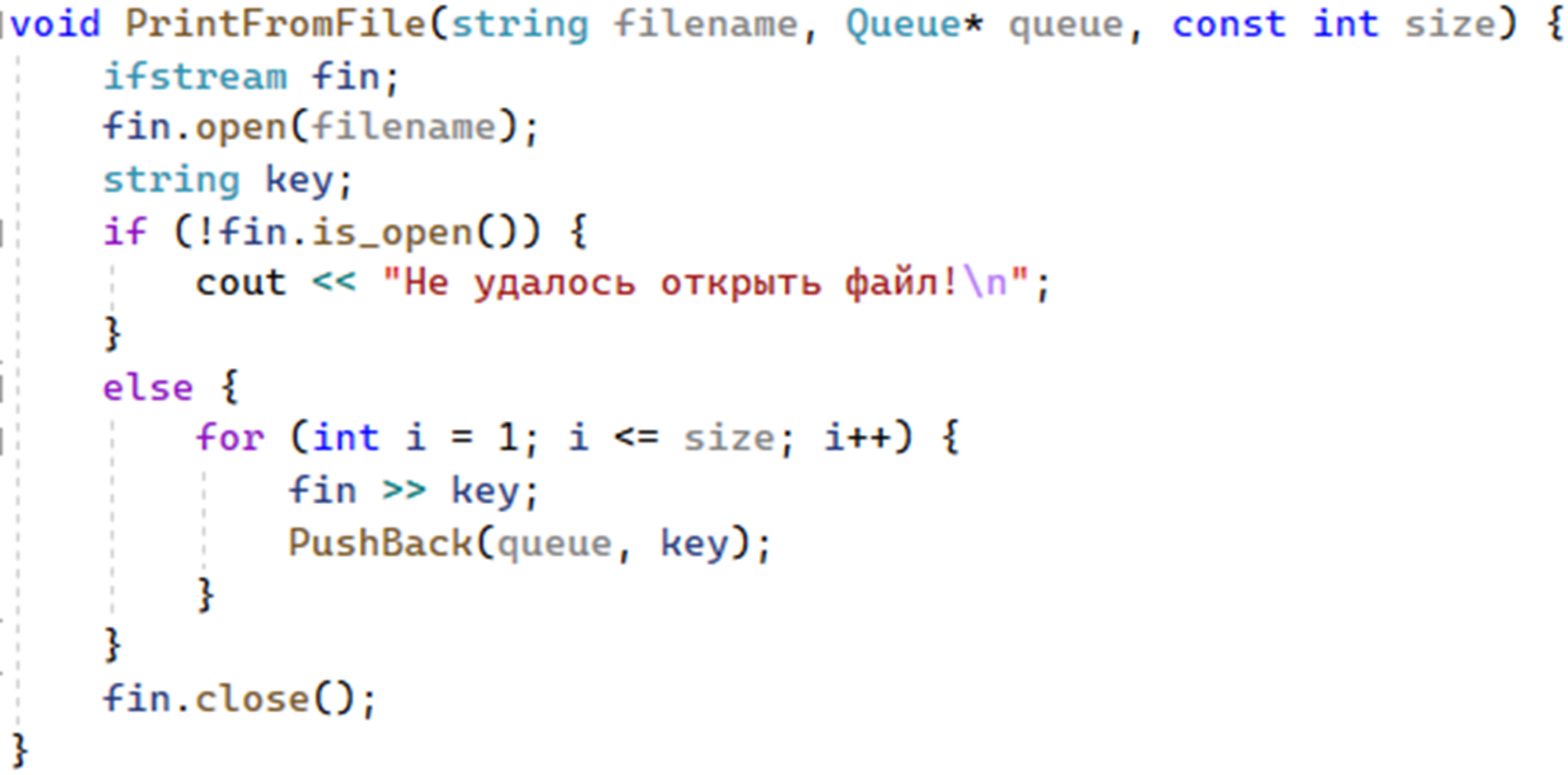


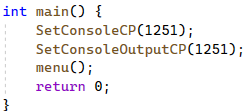












# Работа программы

