|  |  |  |
| --- | --- | --- |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное автономное  образовательное учреждение высшего образования  «Пермский государственный национальный  исследовательский университет» | | |
|  | Институт компьютерных наук и технологий | |
| **ОТЧЁТ**  по индивидуальной работе №2  по дисциплине «Языки программирования Python »  Вариант 10 | | |
|  | | Работу выполнил  студент группы ИТ-7,8-2025 1 курса  Латынцева Валерия Романовна  «12» июня 2025 г. |
| Работу проверил  Рубцова М. Б.  «16» июня 2025 г. |
| Пермь 2025 | | |

СОДЕРЖАНИЕ

[Постановка задачи 3](#_Toc192898948)

[Алгоритм решения 4](#_Toc792402246)

[Обоснование выбранных структур и типов данных](#_Toc2144919390) 7

[Тестирование 10](#_Toc1442624374)

[Тесты на работоспособность 10](#_Toc992866750)

[Полный список команд: 10](#_Toc669988185)

[Тесты на проверку “защиты от некорректного пользовательского ввода” 12](#_Toc978767114)

[Код программы 13](#_Toc1810785696)

# Постановка задачи

Очередь. Реализуйте структуру данных "очередь". Напишите программу, содержащую описание очереди и моделирующую работу очереди, реализовав все указанные здесь методы. Программа считывает последовательность команд и в зависимости от команды выполняет ту или иную операцию. После выполнения каждой команды программа должна вывести одну строчку.

Возможные команды для программы:

push n Добавить в очередь число n (значение n задается после команды). Программа должна вывести ok.

pop Удалить из очереди первый элемент. Программа должна вывести его значение.

front Программа должна вывести значение первого элемента, не удаляя его из очереди.

size Программа должна вывести количество элементов в очереди.

clear Программа должна очистить очередь и вывести ok.

exit Программа должна вывести bye и завершить работу.

Перед исполнением операций front и pop программа должна проверять, содержится ли в очереди хотя бы один элемент. Если во входных данных встречается операция front или pop, и при этом очередь пуста, то программа должна вместо числового значения вывести строку error.

Входные данные: вводятся команды управления очередью, по одной на строке

Выходные данные: вывести протокол работы очереди, по одному сообщению на строке

**Алгоритм решения**

1. Структура данных

Класс QueueNode:

Содержит:

* value - хранимое значение элемента
* next - ссылка на следующий узел (по умолчанию None)

Класс Queue:

Основные поля:

* head - указатель на первый элемент очереди
* tail - указатель на последний элемент очереди

Особенности:

* Размер вычисляется динамически
* Реализация на основе связного списка

2. Основные операции

push(n):

* Создает новый узел с значением n

Если очередь пуста (tail is None):

* Устанавливает head и tail на новый узел

Иначе:

* Присоединяет новый узел после tail
* Обновляет tail на новый узел
* Возвращает строку "ok"

pop():

Если очередь пуста (head is None):

* Возвращает "error"

Иначе:

* Сохраняет значение head.value
* Перемещает head на следующий узел

Если очередь стала пустой (head is None), обнуляет tail

* Возвращает сохраненное значение

front():

Если очередь пуста:

* Возвращает "error"

Иначе:

* Возвращает значение head.value

size():

* Инициализирует счетчик (0)
* Последовательно проходит все узлы от head до tail
* Возвращает количество узлов

clear():

* Обнуляет указатели head и tail
* Возвращает "ok"

print\_queue():

Если очередь пуста:

* Выводит "Очередь пуста"

Иначе:

* Выводит все значения элементов через пробел

3. Работа с пользователем

Основной цикл (функция main()):

* Выводит меню с доступными командами
* Ожидает ввод пользователя
* Обрабатывает команды:

Команды:

push <число>:

* Проверяет наличие числа после команды
* Проверяет, что введено целое число
* Добавляет число в очередь
* Выводит "ok" или сообщение об ошибке

pop:

* Удаляет первый элемент
* Выводит его значение или "error"
* При успехе выводит подтверждение удаления

front:

* Выводит первый элемент или "error"

size:

* Выводит текущий размер очереди

clear:

* Очищает очередь
* Выводит подтверждение очистки

exit:

* Завершает программу (выводит "bye")

После каждой команды:

* Автоматически выводит текущее состояние очереди

4. Обработка ошибок

Типы ошибок:

Ввод:

* Некорректный формат команды push
* Отсутствие числа после push
* Нечисловые значения
* Неизвестные команды

Операции:

* Попытки операций с пустой очередью (pop, front)

Механизм обработки:

* Проверка корректности ввода в блоке try-except
* Вывод понятных сообщений об ошибках
* Продолжение работы программы после ошибок

## Обоснование выбранных структур и типов данных

1. QueueNode (узел односвязного списка)

Содержит:

* value (число, int) - хранимое значение элемента очереди
* next (ссылка на следующий узел) - указатель на следующий элемент

Обоснование:

Минималистичная структура оптимальна:

* Требуется только ссылка на следующий элемент (не нужен двусвязный список)
* Экономия памяти ( По сравнению с двусвязным списком)

Простота управления:

* Легко добавлять в конец (tail.next)
* Легко удалять из начала (head = head.next)

2. Queue (очередь на связном списке)

Поддерживает операции:

* push - добавление в конец
* pop - удаление из начала
* front - просмотр головы
* size - получение размера
* clear - полная очистка
* print\_queue - вывод всех элементов

Обоснование:

Эффективность операций:

* Все ключевые операции выполняются за константное время
* В отличие от массива, нет дорогостоящих операций сдвига

Динамическое управление памятью:

* Автоматическое расширение/сжатие при операциях
* Нет ограничений на максимальный размер (в отличие от статических массивов)

Оптимальная структура:

* Поддержка двух указателей (head и tail) позволяет эффективно работать с обоими концами

3. Обработка команд (функция main())

Использует:

* Интерактивный ввод/вывод через консоль
* Простую обработку строковых команд
* Встроенные проверки ошибок

Обоснование:

Простота интерфейса:

* Понятные текстовые команды соответствуют стандартным операциям с очередью
* Минималистичный дизайн без избыточных функций

Отказоустойчивость:

* Проверка типов для команды push
* Обработка попыток операций с пустой очередью
* Универсальный обработчик исключений

Визуализация состояния:

* Автоматический вывод очереди после каждой операции
* Понятные сообщения об ошибках

# Тестирование

## Тесты на работоспособность

## Полный список:

**МЕНЮ ПРОГРАММЫ ОЧЕРЕДЬ**

Добро пожаловать в программу очередь! Доступные команды:

1. push <число> - Добавить число в очередь

2. pop - Удалить первый элемент

3. front - Показать первый элемент

4. size - Показать размер очереди

5. clear - Очистить очередь

6. exit - Выйти из программы

Введите команду полностью (например: 'push 3')!

Введите команду:

## Команда push <>:

Введите команду: push 5

ok

Содержимое очереди: 5

Введите команду: push 8

ok

Содержимое очереди: 5 8

Введите команду: push 3

ok

Содержимое очереди: 5 8 3

## Команда pop:

Введите команду: pop

5

Элемент удален

Содержимое очереди: 8 3

Введите команду: pop

8

Элемент удален

Содержимое очереди: 3

Введите команду: pop

3

Элемент удален

Содержимое очереди: Очередь пуста

Введите команду: pop

error

Содержимое очереди: Очередь пуста

## Команда front:

Введите команду: push 7

ok

Содержимое очереди: 7

Введите команду: push 2

ok

Содержимое очереди: 7 2

Введите команду: front

7

Содержимое очереди: 7 2

Введите команду: clear

ok

Очередь очищена

Содержимое очереди: Очередь пуста

Введите команду: front

error

Содержимое очереди: Очередь пуста

**Команда size:**

Введите команду: size

0

Содержимое очереди: Очередь пуста

Введите команду: push 1

ok

Содержимое очереди: 1

Введите команду: push 4

ok

Содержимое очереди: 1 4

Введите команду: size

2

Содержимое очереди: 1 4

**Команда clear:**

Введите команду: push 9

ok

Содержимое очереди: 9

Введите команду: push 6

ok

Содержимое очереди: 9 6

Введите команду: clear

ok

Очередь очищена

Содержимое очереди: Очередь пуста

Введите команду: size

0

Содержимое очереди: Очередь пуста

## Команда exit:

Введите команду: exit

bye

## Тесты на проверку “защиты от некорректного пользовательского ввода”

**Некорректный push:**

Введите команду: push

Ошибка: после 'push' укажите число

Содержимое очереди: Очередь пуста

Введите команду: push abc

Ошибка: необходимо ввести целое число

Содержимое очереди: Очередь пуста

**Неизвестная команда:**

Введите команду: peek

Неизвестная команда: peek

Содержимое очереди: Очередь пуста

# Код программы

[Ссылка на гитхаб](https://github.com/ValeriLat/LatyntsevaVIT8.git)