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 «Язык программирования Python»
Вариант 10

Работу выполнил
студент группы ИТ-5,6-2025 1 курса
Лесникова П. С.
«12» июня 2025 г.

Работу проверил
Рубцова М. Б.
«18» июня 2025 г.

Пермь 2025

СОДЕРЖАНИЕ

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. Основные операции
    - push(n)-Добавление элемента:

        Создаем новый узел new_node с данными n.

        Если очередь пуста:

        Устанавливаем head = new_node и tail = new_node.

        Иначе:

        Присоединяем new_node к концу.

        Перемещаем tail на новый узел.

        Увеличиваем length на 1.

        Возвращаем "ok".
    - pop()-Удаление элемента

        Если очередь пуста:

        Возвращаем "error".

        Сохраняем значение head.data.

        Перемещаем head на следующий узел.

        Если head стал None:

        Обнуляем tail.

        Уменьшаем length на 1.

        Возвращаем сохраненное значение.
    - front()-Просмотр первого элемента

        Если очередь пуста:

        Возвращаем "error".

        Иначе:

        Возвращаем head.data.
    - size()-Получение размера

        Возвращаем length.
    - clear()-Очистка очереди

        Обнуляем head и tail.

        Сбрасываем length в 0.

        Возвращаем "ok".
2. Работа с пользователем

    Основной цикл:

- Выводит меню с доступными командами
- Ожидает ввод пользователя

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 удаленный элемент

- front:

  Выводит первый элемент или "error"

- size:

  Выводит текущий размер очереди

- clear:

  Очищает очередь

  Выводит "ok"

- exit:

  Завершает программу

  Выводит "bye"

3. Обработка ошибок

Ввод:

- Некорректный формат команды
- 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)

   Обоснование:

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

2. Queue (очередь)

   Обоснование:

   - Все ключевые операции выполняются за константное время
   - Нет ограничений на максимальный размер
   - Поддержка двух указателей (head и tail) позволяет эффективно работать с обоими концами

3. Обработка команд

   Обоснование:

   - Понятные текстовые команды соответствуют стандартным операциям с очередью

Тестирование

Добро пожаловать в программу очередь! Доступные команды:
push <число> - Добавить число в очередь
pop - Удалить первый элемент
front - Показать первый элемент
size - Показать размер очереди
clear - Очистить очередь
exit - Выйти из программы
Введите команду полностью (например: 'push 1')
Введите команду:

Команда push <число>:
Введите команду: push
Ошибка: после 'push' укажите число
Введите команду: push 6
ok
Введите команду: push +
Ошибка: необходимо ввести целое число
Введите команду: push 5/6
Ошибка: необходимо ввести целое число

Команда pop:
Введите команду: pop
6
Введите команду: pop 8
error
Введите команду: pop
error

Команда front:
Введите команду: push 3
ok
Введите команду: front
3

Введите команду: push 4
ok
Введите команду: push 5
ok
Введите команду: front
3
Введите команду: clear
ok
Введите команду: front
error

Команда size:

Введите команду: clear
ok
Введите команду: front
error
Введите команду: size
0
Введите команду: push 6
ok
Введите команду: push 4
ok
Введите команду: size
2

Команда clear:

Введите команду: push 6
ok
Введите команду: push 4
ok
Введите команду: size
2
Введите команду: clear
ok

Введите команду: size

0

Введите команду: front

error

Команда exit:

Введите команду: exit

bye

Тесты на проверку “защиты от некорректного пользовательского ввода”:

Введите команду: push 5l5

Ошибка: необходимо ввести целое число

Введите команду: push

Ошибка: после 'push' укажите число

Введите команду: pash 5

Неизвестная команда: pash

Введите команду: pop 6

error

Введите команду: frant

Неизвестная команда: frant

Код программы

Ссылка на гитхаб