|  |  |  |
| --- | --- | --- |
| 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-5-2024 1 курса  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Захаров П.А.  «\_\_\_» \_\_\_\_\_\_\_\_2025 г. |
| Работу проверил  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Рубцова М.Б.  «\_\_\_» \_\_\_\_\_\_\_\_\_ 2025 г. |
| Пермь 2025 | | |

СОДЕРЖАНИЕ

[Постановка задачи 3](#_Toc153130027)

[Алгоритм решения 3](#_Toc153130028)

[Тестирование 3](#_Toc153130029)

Код программы...................................................................................................................5

# 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.

# Алгоритм решения

Для решения этой задачи я реализовал класс структуры данных очередь и вспомогательный класс узлов. В классе очереди реализованы методы для добавления элемента в конец очереди - push(), удаления и возврат элемента из начала очереди – pop(), возврат элемента из начала очереди без удаления – front(), возврат длины очереди – length(), удаления очереди – clear(), и метод выхода из программы – exit(). Функционал взаимодействия с очередью реализован через цикл while True с вводом команд через терминал.

# Тестирование

Тест 1:

Введите команду - push 1

ok

Введите команду - front

1

Введите команду – exit

Bye

Тест 2:

Введите команду - size

0

Введите команду - push 1

ok

Введите команду - size

1

Введите команду - push 2

ok

Введите команду - size

2

Введите команду - push 3

ok

Введите команду - size

3

Введите команду - exit

bye

Тест 3:

Введите команду - push 1

ok

Введите команду - pop

1

Введите команду - size

0

Введите команду - push 1

ok

Введите команду - clear

ok

Введите команду - front

error

Введите команду - size

0

Введите команду - exit

bye

# Код программы

Код - https://github.com/frorepelitta/IKM2

class Node: # узел для реализации очереди

    def \_\_init\_\_(self, data):

        self.data = data

        self.next = None

class Queue: # реализация очереди

    def \_\_init\_\_(self):

        self.head = None

        self.tail = None

        self.size = 0

    def push(self, data): # добавление элемента в конец очереди

        new\_node = Node(data)

        if self.head is None:

            self.head = new\_node

            self.tail = new\_node

        else:

            self.tail.next = new\_node

            self.tail = new\_node

        self.size += 1

        return 'ok'

    def pop(self): # удаления в возврат элемента из начала очереди

        if self.head is None:

            return 'error'

        node = self.head.data

        self.head = self.head.next

        self.size -= 1

        if self.head is None:

            self.tail = None

        return node

    def front(self): # возврат элемента из начала очереди без удаления

        if self.head == None:

            return 'error'

        return self.head.data

    def length(self): # возврат длины очереди

        return self.size

    def clear(self): # удаляет полностью очередь

        self.head = self.tail = None

        self.size = 0

        return 'ok'

    def exit(self): # выход из программы

        return 'bye'

queue = Queue()

while True:

    command = input('Введите команду - ').strip().split()

    if not command:

        continue

    if command[0] == "push":

        if len(command) != 2:

            print("Неправильный ввод, чтобы добавить элемент в очередь, нужно написать команду и число.")

            continue

        n = int(command[1])

        print(queue.push(n))

    elif command[0] == "pop":

        print(queue.pop())

    elif command[0] == "front":

        print(queue.front())

    elif command[0] == "size":

        print(queue.length())

    elif command[0] == "clear":

        print(queue.clear())

    elif command[0] == "exit":

        print(queue.exit())

        break

    else:

        print("Неправильная команда!")