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 14

Работу выполнил
студент группы ИТ-6-2024 1 курса
Галиева Гулия Ралитовна
«12» июня 2025 г.

Работу проверил
_____ Фамилия И.О.
«___» _____ 2024 г.

Пермь 2024

СОДЕРЖАНИЕ

Постановка задачи

Реализуйте с помощью двусвязного циклического списка следующую игру: Ученики школы встают в круг. Один ученик пишет программу, генерирующую случайное целое число из промежутка [–10;10]. Если выпало положительное число, то отсчёт ведётся «по часовой стрелке», если отрицательное, то «против часовой». Ученик, на котором остановился счёт, не выбывает из круга, а делает «доброе дело» и уровень его рейтинга увеличивается на 1. Следующий отсчёт начинается с ученика, стоящего в круге рядом с тем, который только что делал «доброе дело», с правого (если отсчёт ведётся по часовой стрелке) или с левого (против часовой). После того, как игра всем надоела, нужно вывести список учеников в порядке невозрастания рейтинга. Если имеется несколько учеников с одинаковым рейтингом, то их нужно выводить в том порядке, в котором они поступили в список. Исходный список фамилий учеников для игры находится в текстовом файле. В начале игры рейтинг каждого ученика равен 0. Во время игры нужно выводить протокол: какое случайное число выпало, какой ученик делал доброе дело и рейтинг ученика после совершения доброго дела. Количество раундов игры вводится пользователем с клавиатуры

Алгоритм решения

1. Чтение данных из файла:

- Использовать класс StudentFileHandler для работы с файлом, содержащим фамилии учеников.
- Метод read_students класса StudentFileHandler читает построчно файл и создаёт кольцевой двусвязный список CircularDoublyLinkedList, где каждый элемент – узел StudentNode, содержащий фамилию ученика, рейтинг (изначально 0) и порядковый номер добавления.
- Метод add_students класса StudentFileHandler позволяет добавлять новые фамилии учеников в файл.

2. Представление данных:

- Для хранения списка учеников используется кольцевой двусвязный список (CircularDoublyLinkedList).

* Обоснование выбора: Кольцевая структура обеспечивает возможность циклического перемещения по списку, что необходимо для моделирования круга учеников. Двусвязность позволяет легко перемещаться как в прямом, так и в обратном направлении, что требуется для отсчёта в обе стороны.

- Каждый ученик представлен объектом класса StudentNode, содержащим:

  name: Фамилия ученика (строка).

  rating: Рейтинг ученика (целое число).

  prev: Ссылка на предыдущий узел в списке.

  next: Ссылка на следующий узел в списке.

  order: Порядковый номер добавления ученика в список (целое число). Это необходимо для обеспечения стабильной сортировки при одинаковом рейтинге.

* Обоснование выбора типов данных: Строки для фамилий и целые числа для рейтингов и порядкового номера добавления являются наиболее подходящими типами данных для представления этой информации.

3. Игровая логика:

- Класс Game инкапсулирует игровую логику.
- Метод play моделирует раунды игры.
- Для каждого раунда:
  * Генерируется случайное целое число в диапазоне [-10, 10] (исключая 0).
  * В зависимости от знака числа определяется направление отсчёта (по часовой или против часовой стрелки).
  * Происходит перемещение по списку на соответствующее количество позиций.
  * Увеличивается рейтинг ученика, на котором остановился отсчёт.
  * Выводится протокол раунда.
- После завершения всех раундов метод print_sorted_students сортирует список учеников по убыванию рейтинга (при равенстве рейтингов – по порядку добавления) и выводит отсортированный список.

4. Сортировка:

- Используется метод sort встроенного списка Python с функцией lambda в качестве ключа сортировки. Сортировка выполняется сначала по убыванию рейтинга (-s.rating), а затем по возрастанию порядка добавления (s.order). Это гарантирует стабильную сортировку при одинаковых рейтингах.

5. Обработка ввода пользователя:

• Класс InputHandler обеспечивает безопасный ввод положительного целого числа для количества раундов. Он обрабатывает исключения ValueError (некорректный ввод) и KeyboardInterrupt (прерывание ввода пользователем).

Тестирование

Для тестирования программы были использованы следующие подходы:

• Ручное тестирование: Программа запускалась с различными входными данными (разное количество раундов, разные фамилии учеников в файле) и проверялся протокол игры, а также отсортированный список учеников. Особое внимание уделялось случаям с отрицательными числами для отсчёта, а также случаям, когда у нескольких учеников одинаковый рейтинг.

• Тестирование крайних случаев:

• Пустой файл с фамилиями учеников. В этом случае программа должна корректно обработать ситуацию и предложить добавить новых учеников.

• Файл с одной фамилией ученика. Программа должна корректно работать в этом случае, обеспечивая циклический отсчёт.

• Большое количество раундов игры. Программа должна корректно обрабатывать большое количество раундов и не приводить к переполнению или другим ошибкам.

• Проверка стабильности сортировки: Создавались ситуации, когда у нескольких учеников оказывался одинаковый рейтинг, и проверялось, что их порядок в отсортированном списке соответствует порядку добавления в список.

• Проверка корректности обработки прерывания ввода пользователем: Проверялось, что программа корректно завершает работу при прерывании ввода пользователем с помощью Ctrl+C.

Примеры тестовых данных:

Введите новые фамилии учеников для добавления в файл.

Для завершения ввода оставьте строку пустой и нажмите Enter.

Фамилия: Галиева

Фамилия: Ершов

Фамилия:

Новые фамилии успешно добавлены в файл.

Введите количество раундов игры: 10

Начинаем игру!

Раунд 1: Выпало число -7 (против часовой стрелки). Ученик Гагарин сделал доброе дело. Рейтинг: 1

Раунд 2: Выпало число 10 (по часовой стрелке). Ученик Гагарин сделал доброе дело. Рейтинг: 2

Раунд 3: Выпало число -9 (против часовой стрелки). Ученик Петрова сделал доброе дело. Рейтинг: 1

Раунд 4: Выпало число -2 (против часовой стрелки). Ученик Иванов сделал доброе дело. Рейтинг: 1

Раунд 5: Выпало число 4 (по часовой стрелке). Ученик Петрова сделал доброе дело. Рейтинг: 2

Раунд 6: Выпало число 1 (по часовой стрелке). Ученик Волков сделал доброе дело. Рейтинг: 1

Раунд 7: Выпало число -6 (против часовой стрелки). Ученик Иванов сделал доброе дело. Рейтинг: 2

Раунд 8: Выпало число -7 (против часовой стрелки). Ученик Попов сделал доброе дело. Рейтинг: 1

Раунд 9: Выпало число 5 (по часовой стрелке). Ученик Волков сделал доброе дело. Рейтинг: 2

Раунд 10: Выпало число 8 (по часовой стрелке). Ученик Волков сделал доброе дело. Рейтинг: 3

Список учеников в порядке невозрастания рейтинга:

Волков: 3

Иванов: 2

Гагарин: 2

Петрова: 2

Попов: 1

Васильев: 0

Зайцев: 0

Галиева: 0

Ершов: 0

Введите новые фамилии учеников для добавления в файл.

Для завершения ввода оставьте строку пустой и нажмите Enter.

Фамилия:

Новые фамилии успешно добавлены в файл.

Введите количество раундов игры: 9

Начинаем игру!

Раунд 1: Выпало число 6 (по часовой стрелке). Ученик Зайцев сделал доброе дело. Рейтинг: 1

Раунд 2: Выпало число 6 (по часовой стрелке). Ученик Васильев сделал доброе дело. Рейтинг: 1

Раунд 3: Выпало число 9 (по часовой стрелке). Ученик Волков сделал доброе дело. Рейтинг: 1

Раунд 4: Выпало число 6 (по часовой стрелке). Ученик Петрова сделал доброе дело. Рейтинг: 1

Раунд 5: Выпало число -7 (против часовой стрелки). Ученик Зайцев сделал доброе дело. Рейтинг: 2

Раунд 6: Выпало число 3 (по часовой стрелке). Ученик Ершов сделал доброе дело. Рейтинг: 1

Раунд 7: Выпало число -2 (против часовой стрелки). Ученик Галиева сделал доброе дело. Рейтинг: 1

Раунд 8: Выпало число -4 (против часовой стрелки). Ученик Гагарин сделал доброе дело. Рейтинг: 1

Раунд 9: Выпало число -1 (против часовой стрелки). Ученик Иванов сделал доброе дело. Рейтинг: 1

Список учеников в порядке невозрастания рейтинга:

Зайцев: 2

Иванов: 1

Гагарин: 1

Петрова: 1

Васильев: 1

Волков: 1

Галиева: 1

Ершов: 1

Попов: 0

Тесты на проверку "защиты от дурака"

Введите новые фамилии учеников для добавления в файл.

Для завершения ввода оставьте строку пустой и нажмите Enter.

Фамилия:

Новые фамилии успешно добавлены в файл.

Введите количество раундов игры: 0

Пожалуйста, введите положительное число.

Введите количество раундов игры: -1

Пожалуйста, введите положительное число.

Введите количество раундов игры: p

Ошибка: введите целое число.

Введите количество раундов игры: 0,5

Ошибка: введите целое число.

Введите количество раундов игры: 1

Начинаем игру!

Раунд 1: Выпало число -10 (против часовой стрелки). Ученик Ершов сделал доброе дело. Рейтинг: 1

Список учеников в порядке невозрастания рейтинга:

Ершов: 1

Иванов: 0

Попов: 0

Гагарин: 0

Петрова: 0

Васильев: 0

Волков: 0

Зайцев: 0

Галиева: 0

Код программы

```
import random

# Класс, представляющий одного ученика в списке
class StudentNode:
    def __init__(self, name, order):
        self.name = name
        self.rating = 0
        self.prev = None
        self.next = None
        self.order = order # Порядок добавления для стабильной
сортировки

# Класс кольцевого двусвязного списка для хранения учеников
class CircularDoublyLinkedList:
    def __init__(self):
        self.head = None # Начало списка
        self.size = 0  # Количество учеников в списке

    # Метод добавления нового ученика в конец списка
    def append(self, name):
        new_node = StudentNode(name, self.size)  # Создаём новый
узел с порядковым номером
        if self.head is None:
            # Если список пуст, новый узел указывает сам на себя
(кольцо из одного элемента)
            self.head = new_node
            new_node.next = new_node
            new_node.prev = new_node
        else:
            # Иначе вставляем новый узел в конец списка
            tail = self.head.prev
            tail.next = new_node
            new_node.prev = tail
            new_node.next = self.head
            self.head.prev = new_node
        self.size += 1
```

```
    # Итератор для прохода по всем узлам списка
    def __iter__(self):
        if not self.head:
            return
        current = self.head
        for _ in range(self.size):
            yield current
            current = current.next

    # Метод для получения списка всех узлов
    def to_list(self):
        return list(self.__iter__())

# Класс для работы с файлом учеников: чтение и добавление новых фамилий
class StudentFileHandler:
    def __init__(self, filename):
        self.filename = filename    # Имя файла для хранения фамилий

    # Чтение учеников из файла и создание списка
    def read_students(self):
        students = CircularDoublyLinkedList()
        try:
            with open(self.filename, 'r', encoding='utf-8') as f:
                for line in f:
                    name = line.strip()
                    if name:
                        students.append(name)
        except FileNotFoundError:
            print(f"Файл '{self.filename}' не найден. Будет создан новый файл при добавлении учеников.")
        except IOError as e:
            print(f"Ошибка при чтении файла '{self.filename}': {e}")
        return students

    # Добавление новых фамилий в файл с клавиатуры
    def add_students(self):
        print("Введите новые фамилии учеников для добавления в файл.")
        print("Для завершения ввода оставьте строку пустой и нажмите Enter.")
        try:
            with open(self.filename, 'a', encoding='utf-8') as f:
                while True:
                    try:
                        surname = input("Фамилия: ").strip()
                    except KeyboardInterrupt:
                        #     Обработка     прерывания     ввода пользователем
                        print("\nВвод прерван пользователем.")
```

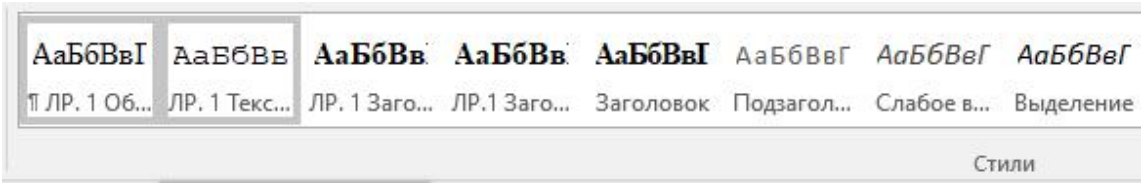
```
                    break
                if surname == '':
                    # Пустая строка — выход из цикла ввода
                    break
                f.write(surname + '\n')     # Записываем
фамилию в файл
            print("Новые фамилии успешно добавлены в файл.\n")
        except IOError as e:
            print(f"Ошибка при записи в файл '{self.filename}':
{e}")

# Класс, управляющий игрой
class Game:
    def __init__(self, students):
        self.students = students  # Кольцевой список учеников

    # Метод запуска игры на заданное количество раундов
    def play(self, rounds):
        if self.students.size == 0:
            print("Список учеников пуст.")
            return

        current = self.students.head  # Начинаем с головы списка

        for round_num in range(1, rounds + 1):
            # Генерируем случайный шаг от -10 до 10, исключая 0
            step = random.randint(-10, 10)
            while step == 0:
                step = random.randint(-10, 10)

            direction = "по часовой стрелке" if step > 0 else
"против часовой стрелки"
            steps_count = abs(step)

            # Двигаемся по кругу на заданное количество шагов
            for _ in range(steps_count):
                current = current.next if step > 0 else
current.prev

            # Ученик, на котором остановились, делает доброе
дело — увеличиваем рейтинг
            current.rating += 1

            # Выводим информацию о раунде
            print(f"Раунд {round_num}: Выпало число {step}
({direction}). "
                  f"Ученик {current.name} сделал доброе дело.
Рейтинг: {current.rating}")

            # Следующий отсчёт начинается с ученика рядом с
current
            current = current.next if step > 0 else current.prev
```

```
    # Метод вывода отсортированного списка учеников по рейтингу
    def print_sorted_students(self):
        student_list = self.students.to_list()
        # Сортируем по рейтингу по убыванию, при равенстве — по
порядку добавления
        student_list.sort(key=lambda s: (-s.rating, s.order))

        print("\nСписок учеников в порядке невозрастания
рейтинга:")
        for s in student_list:
            print(f"{s.name}: {s.rating}")

# Класс для обработки ввода пользователя
class InputHandler:
    def get_positive_int(self, prompt):
        # Метод для безопасного ввода положительного целого
числа
        while True:
            try:
                value = input(prompt)
                num = int(value)
                if num <= 0:
                    print("Пожалуйста, введите положительное
число.")
                    continue
                return num
            except ValueError:
                print("Ошибка: введите целое число.")
            except KeyboardInterrupt:
                print("\nВвод прерван пользователем.")
                return None

# Главная функция программы
def main():
    filename = "students.txt"  # Имя файла с фамилиями учеников
    file_handler = StudentFileHandler(filename)

    # Сначала даём возможность добавить новых учеников в файл
    file_handler.add_students()

    # Читаем учеников из файла в кольцевой список
    students = file_handler.read_students()

    input_handler = InputHandler()
    rounds = input_handler.get_positive_int("Введите количество
раундов игры: ")
    if rounds is None:
        # Если ввод прерван, завершаем программу
        return

    game = Game(students)
```

```
    print("\nНачинаем игру!\n")
    game.play(rounds)                # Запускаем игру
    game.print_sorted_students()  # Выводим результаты

if __name__ == "__main__":
    main()
```

Инструкция по применению стилей и оформлению работы

Для оформления частей отчёта следует использовать заранее созданные стили. Все стили, которые могут пригодиться, начинаются с «ЛР. 1 …».



ЛР. 1 Обычный – для оформления текста задания и алгоритма решения.

ЛР. 1 Текст программы – для оформления кода программы.

ЛР. 1 Заголовок 1 – заголовок первого уровня (для того, чтобы озаглавить основные разделы отчета).

ЛР. 1 Заголовок 2 – заголовок второго уровня (для того, чтобы озаглавить подразделы).

Для того, чтобы перенести текст следующего блока на другую страницу, необходимо воспользоваться инструментов «Разрыв страницы» в разделе «Вставка».