Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №7
дисциплины
«Объектно-ориентированное программирование»
Вариант № 18

Выполнил:
Текеева Мадина Азрет-Алиевна
3 курс, группа ИВТ-б-о-23-2,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____
(подпись)

Проверил:
Доцент департамента цифровых, робототехнических систем и электроники института перспективной инженерии Воронкин Р.А

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2025 г.

Тема: Управление потоками в Python.

Цель работы: приобретение навыков написания многопоточных приложений на языке программирования python версии 3.x.

Репозиторий: https://github.com/bickrosss/TM_oop_lab7

Порядок выполнения работы:

1. Базовый пример создания потока через функцию.

```
from threading import Thread
from time import sleep

def func():
    for i in range(5):
        print(f"from child thread: {i}")
        sleep(0.5)

th = Thread(target=func)
th.start()

for i in range(5):
    print(f"from main thread: {i}")
    sleep(1)
```

2. Запуск двух потоков.

```
from threading import Thread
from time import sleep

def func():
    for i in range(5):
        print(f"from child thread: {i}")
        sleep(0.5)

th1 = Thread(target=func)
th2 = Thread(target=func)

th1.start()
th2.start()

th1.join()
th2.join()

print("--> stop")
```

3. Проверка состояния потока.

```
from threading import Thread
from time import sleep

def func():
    for i in range(5):
        print(f"from child thread: {i}")
        sleep(0.5)

th = Thread(target=func)
print(f"thread status: {th.is_alive()}")

th.start()
print(f"thread status: {th.is_alive()}")

sleep(5)
print(f"thread status: {th.is_alive()}")
```

4. Создание потока через наследование от Thread.

```
from threading import Thread
from time import sleep

class CustomThread(Thread):
    def __init__(self, limit):
        Thread.__init__(self)
        self.limit_ = limit

    def run(self):
        for i in range(self.limit_):
            print(f"from CustomThread: {i}")
            sleep(0.5)

cth = CustomThread(3)
cth.start()
```

5. Остановка потока с использованием Lock.

```
from threading import Thread, Lock
from time import sleep

lock = Lock()
stop_thread = False

def infinite_worker():
    print("Start infinite_worker()")
    while True:
        print("--> thread work")
```

```
        lock.acquire()
        if stop_thread is True:
            break
        lock.release()
        sleep(0.1)
    print("Stop infinite_worker()")

# Create and start thread
th = Thread(target=infinite_worker)
th.start()
sleep(2)

# Stop thread
lock.acquire()
stop_thread = True
lock.release()
```

6. Работа с демоническими потоками.

```
from threading import Thread
from time import sleep

def func():
    for i in range(5):
        print(f"from child thread: {i}")
        sleep(0.5)

th = Thread(target=func, daemon=True)  # или th.daemon = True
th.start()
sleep(0.2)
print("App stop")
```

7. Выполнили задание 1.

С использованием многопоточности для заданного значения x найти сумму ряда S с точностью члена ряда по абсолютному значению ε = 10^{-7} и произвести сравнение полученной суммы с контрольным значением функции y для двух бесконечных рядов.

18.

$$S = \sum_{n=0}^{\infty} \frac{x^{2n+1}}{2n+1} = x + \frac{x^3}{3} + \frac{x^5}{5} + \ldots;$$

$$x = 0,35;\ y = \ln \sqrt{\frac{1+x}{1-x}}.$$

Рисунок 1. Вариант 18

Листинг программы:

```
from threading import Thread
import math

class Series:
    """
    Класс для вычисления суммы ряда:
    S = Σ [x^(2n+1) / (2n+1)], n = 0 .. ∞
    Контрольное значение: y = 0.5 * ln((1+x)/(1-x))
    """

    def __init__(self, x: float = 0.35, eps: float = 1e-7) -> None:
        self.x = x
        self.eps = eps

    def _term(self, n: int) -> float:
        """
        Вычисление члена ряда для заданного n.
        """
        return (self.x ** (2 * n + 1)) / (2 * n + 1)

    def _worker(self, start: int, step: int, result_list: list, index: int) ->
None:
        """
        Потоковая функция для вычисления частичной суммы.
        """
        sum_local = 0.0
        n = start
        a_n = self._term(n)
        count = 0

        while abs(a_n) >= self.eps:
            sum_local += a_n
            n += step
            a_n = self._term(n)
            count += 1

        result_list[index] = sum_local
        # Для отладки можно раскомментировать:
        # print(f"[Thread {index}] Завершил. Посчитано членов ряда: {count}")

    def calculate(self, num_threads: int = 4) -> float:
        """
        Вычисление суммы ряда с использованием многопоточности.
        """
        threads = []
        results = [0.0] * num_threads

        # Создание и запуск потоков
```

```
        for i in range(num_threads):
            thread = Thread(target=self._worker, args=(i, num_threads, results,
i))
            threads.append(thread)
            thread.start()

        # Ожидание завершения всех потоков
        for thread in threads:
            thread.join()

        # Суммирование результатов всех потоков
        return sum(results)

    def __str__(self) -> str:
        """
        Строковое представление ряда и параметров.
        """
        return (
            "Исходный ряд:\n"
            "S = Σ [x^(2n+1) / (2n+1)],  n = 0 .. ∞\n\n"
            f"Параметры:\n"
            f"x = {self.x}\n"
            f"epsilon = {self.eps}\n\n"
            "Аналитическое выражение суммы:\n"
            "y = 0.5 * ln((1+x)/(1-x))"
        )

    def ex_value(self) -> float:
        """
        Вычисление контрольного значения функции.
        """
        return 0.5 * math.log((1 + self.x) / (1 - self.x))
```

Листинг вызова:

```
from time import time
from sum_series import Series

if __name__ == "__main__":
    # Создание объекта ряда с параметрами варианта 18
    series = Series(x=0.35, eps=1e-7)

    # Вывод информации о ряде
    print(series)

    # Многопоточное вычисление
    start_time = time()
    s_numeric = series.calculate(num_threads=4)
    end_time = time()
```

```python
# Контрольное значение
s_exact = series.ex_value()

# Вывод результатов
print("\nРезультаты вычислений:")
print(f"Сумма ряда (численно)      = {s_numeric:.10f}")
print(f"Контрольное значение y     = {s_exact:.10f}")
print(f"Абсолютная погрешность     = {abs(s_numeric - s_exact):.2e}")
print(f"Требуемая точность ε       = {series.eps:.1e}")
print(f"Время вычисления (4 потока) = {end_time - start_time:.6f} сек")

# Проверка точности
if abs(s_numeric - s_exact) < series.eps:
    print("\n✓ Точность достигнута: |S - y| < ε")
else:
    print("\n✗ Точность не достигнута: |S - y| ≥ ε")
```

```
Исходный ряд:
S = Σ [x^(2n+1) / (2n+1)],  n = 0 .. ∞

Параметры:
x = 0.35
epsilon = 1e-07

Аналитическое выражение суммы:
y = 0.5 * ln((1+x)/(1-x))

Результаты вычислений:
Сумма ряда (численно)      = 0.3654436525
Контрольное значение y     = 0.3654437543
Абсолютная погрешность     = 1.02e-07
Требуемая точность ε       = 1.0e-07
Время вычисления (4 потока) = 0.000502 сек

✗ Точность не достигнута: |S - y| ≥ ε
```

Рисунок 1. Результат выполнения программы

Вывод: в ходе выполнения лабораторной работы были приобретены навыки написания многопоточных приложений на языке программирования python версии 3.x.