Лабораторная работа 2. Ряд Фибоначчи с помощью итераторов

Задание 1

gen_fib.py

```python
import functools
import itertools


def fib_elem_gen():  3 usages
    """Генератор, возвращающий элементы ряда Фибоначчи"""
    a = 0
    b = 1

    while True:
        yield a
        res = a + b
        a = b
        b = res


def fib_coroutine(g):  1 usage
    @functools.wraps(g)
    def inner(*args, **kwargs):
        gen = g(*args, **kwargs)
        gen.send(None)
        return gen

    return inner


@fib_coroutine  11 usages
def my_genn():
    """Сопрограмма для генерации ряда Фибоначчи"""
    fib_gen = fib_elem_gen()

    while True:
        number_of_fib_elem = yield
        if number_of_fib_elem <= 0:
            result = []
        else:
            # Генерируем n элементов с помощью itertools.islice
            result = list(itertools.islice(fib_gen, number_of_fib_elem))
            # Сбрасываем генератор для следующего вызова
            fib_gen = fib_elem_gen()

        yield result


if __name__ == "__main__":
    g = fib_elem_gen()

    print("Первые 10 элементов ряда Фибоначчи:")
    for i in range(10):
        el = next(g)
        print(el, end=" ")
    print("\n")

    gen = my_genn()
    print("gen.send(3):", gen.send(3))

```

Результат

```
Первые 10 элементов ряда Фибоначчи:
0 1 1 2 3 5 8 13 21 34

gen.send(3): [0, 1, 1]

Process finished with exit code 0
```

test_fib.py

```python
import pytest
from gen_fib import my_genn


def test_fib_1():
    """Тривиальный случай n = 3, список [0, 1, 1]"""
    gen = my_genn()
    assert gen.send(3) == [0, 1, 1]


def test_fib_2():
    """Пять первых членов ряда"""
    gen = my_genn()
    assert gen.send(5) == [0, 1, 1, 2, 3]

def test_fib_zero():
    """Крайний случай: запрос 0 элементов"""
    gen = my_genn()
    assert gen.send(0) == []


def test_fib_one():
    """Крайний случай: запрос 1 элемента"""
    gen = my_genn()
    assert gen.send(1) == [0]
```

```python
def test_fib_two():
    """Крайний случай: запрос 2 элементов"""
    gen = my_genn()
    assert gen.send(2) == [0, 1]


def test_fib_negative():
    """Крайний случай: отрицательное количество элементов"""
    gen = my_genn()
    assert gen.send(-1) == []


def test_fib_sequence():
    """Последовательные вызовы с разными параметрами"""
    gen = my_genn()
    assert gen.send(3) == [0, 1, 1]
    assert gen.send(5) == [0, 1, 1, 2, 3]
    assert gen.send(2) == [0, 1]


def test_fib_large():
    """Запрос большего количества элементов"""
    gen = my_genn()
    result = gen.send(10)
    expected = [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
    assert result == expected
```

```python


def test_fib_empty_after_zero():
    """Проверка поведения после запроса 0 элементов"""
    gen = my_genn()
    assert gen.send(0) == []
    assert gen.send(3) == [0, 1, 1]


if __name__ == "__main__":
    pytest.main([__file__, "-v"])
```

Результат

```
Testing started at 8:53 ...
Launching pytest with arguments test_fib.py::test_fib_large --no-header --no-summary -q in D:\PycharmProjects\Prog5\LR2

============================== test session starts ==============================
collecting ... collected 1 item

test_fib.py::test_fib_large PASSED                                       [100%]

============================== 1 passed in 0.08s ===============================

Process finished with exit code 0
```

Задание 2

gen_fib.py

```python
import functools
import itertools
import math


def fib_elem_gen():  3 usages
    """Генератор, возвращающий элементы ряда Фибоначчи"""
    a = 0
    b = 1
    while True:
        yield a
        res = a + b
        a = b
        b = res

def fib_coroutine(g):  1 usage
    @functools.wraps(g)
    def inner(*args, **kwargs):
        gen = g(*args, **kwargs)
        gen.send(None)
        return gen
    return inner

@fib_coroutine  2 usages
def my_genn():
    """Сопрограмма для генерации ряда Фибоначчи"""
    fib_gen = fib_elem_gen()
    while True:
        number_of_fib_elem = yield
        if number_of_fib_elem <= 0:
            result = []
        else:
            # Генерируем n элементов с помощью itertools.islice
            result = list(itertools.islice(fib_gen, number_of_fib_elem))
            # Сбрасываем генератор для следующего вызова
            fib_gen = fib_elem_gen()
        yield result

class FibonacchiLst:  7 usages
    """Итератор, возвращающий элементы из списка, которые принадлежат ряду Фибоначчи"""
    def __init__(self, lst):
        self.lst = lst
        self.idx = 0  # инициализируем индекс для перебора элементов

    def __iter__(self):
        return self  # возвращает экземпляр класса, реализующего протокол итераторов

    def __next__(self):
        """Возвращает следующий элемент из списка, который является числом Фибоначчи"""
        while True:
            if self.idx >= len(self.lst):
                raise StopIteration
```

```
                current_element = self.lst[self.idx]
                self.idx += 1



# Демонстрация работы
if __name__ == "__main__":
    g = fib_elem_gen()

    print("Первые 10 элементов ряда Фибоначчи:")
    for i in range(10):
        el = next(g)
        print(el, end=" ")
    print("\n")

    # Тестирование сопрограммы
    gen = my_genn()
    print("gen.send(3):", gen.send(3))



    # Тестирование FibonacchiLst
    print("\nТестирование FibonacchiLst:")
    test_list = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1]
    fib_iterator = FibonacchiLst(test_list)
    result = list(fib_iterator)
    print(f"Для списка {test_list}")
    print(f"Числа Фибоначчи: {result}")
    print(f"Ожидаемый результат: [0, 1, 2, 3, 5, 8, 1]")
```

Результат

```
Первые 10 элементов ряда Фибоначчи:
0 1 1 2 3 5 8 13 21 34

gen.send(3): [0, 1, 1]

Тестирование FibonacchiLst:
Для списка [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1]
Числа Фибоначчи: []
Ожидаемый результат: [0, 1, 2, 3, 5, 8, 1]

Process finished with exit code 0
```

test_fib.py

```python
import pytest
from gen_fib import my_genn, FibonacchiLst

# Тесты для FibonacchiLst
def test_fibonacci_lst_basic():
    """Базовый тест для FibonacchiLst"""
    test_list = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1]
    result = list(FibonacchiLst(test_list))
    expected = [0, 1, 2, 3, 5, 8, 1]
    assert result == expected

def test_fibonacci_lst_empty():
    """Тест с пустым списком"""
    result = list(FibonacchiLst([]))
    assert result == []

def test_fibonacci_lst_single():
    """Тест с одним элементом"""
    result = list(FibonacchiLst([8]))
    assert result == [8]

def test_fibonacci_lst_negative():
    """Тест с отрицательными числами"""
    result = list(FibonacchiLst([-1, 0, 1, -2]))
    assert result == [0, 1]

def test_fibonacci_lst_large_numbers():
    """Тест с большими числами Фибоначчи"""
    test_list = [55, 89, 144, 200]
    result = list(FibonacchiLst(test_list))
    assert result == [55, 89, 144]

if __name__ == "__main__":
    pytest.main([__file__, "-v"])
```

Результат

```
============================= test session starts ==============================
collecting ... collected 1 item

test_fib.py::test_fibonacci_lst_large_numbers PASSED                     [100%]

============================== 1 passed in 0.05s ===============================

Process finished with exit code 0
```