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

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №2.2
дисциплины «Основы программной инженерии»

Выполнил:
Баратов Семен Григорьевич
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

_____
(подпись)

Преподаватель:
Воронкин Р.А., канд. тех. наук, доцент,
доцент кафедры инфокоммуникаций

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Тема: Условные операторы и циклы в языке Python.

Цель: приобретение навыков программирования разветвляющихся алгоритмов и алгоритмов циклической структуры. Освоить операторы языка Python версии 3.x if, while, for, break и continue, позволяющих реализовывать разветвляющиеся алгоритмы и алгоритмы циклической структуры.

Результаты выполнения

1. Создали репозиторий с лицензией MIT, добавили в .gitignore необходимые правила для работы с IDE PyCharm, клонировали репозиторий, организовали репозиторий в соответствии с моделью git-flow.

```
Last login: Tue Oct 24 20:29:02 on ttys000
itssyoma@MacBook-Air-Sema Основы программной инженерии % git clone https://
github.com/itssyoma/megarepo_21.git
Cloning into 'megarepo_21'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
itssyoma@MacBook-Air-Sema Основы программной инженерии % cd megarepo_21
itssyoma@MacBook-Air-Sema megarepo_21 % git checkout -b develop
Switched to a new branch 'develop'
itssyoma@MacBook-Air-Sema megarepo_21 % git branch release
itssyoma@MacBook-Air-Sema megarepo_21 % git branch develop
fatal: a branch named 'develop' already exists
itssyoma@MacBook-Air-Sema megarepo_21 % git branch hotfix
itssyoma@MacBook-Air-Sema megarepo_21 % git branch feature
itssyoma@MacBook-Air-Sema megarepo_21 %
```

Рисунок 1 – Работа с репозиторием в командной строке.

2. Самостоятельно изучили рекомендации к оформлению исходного кода на языке Python PEP-8.

3. Проработали пример №1.

example1.py > ...

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import math


if __name__ == '__main__':
    x = float(input("Value of x? "))

    if x <= 0:
        y = 2 * x * x + math.cos(x)
    elif x < 5:
        y = x+1
    else:
        y = math.sin(x) - x * x

    print(f"y = {y}")

```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ

```
(.venv) itssyoma@MacBook-Air-Sema megarepo_22 % "/Users/itssyoma/Yandex.Disk.lo
dex.Disk.localized/Учеба/Основы программной инженерии/megarepo_22/example1.py"
Value of x? -3
y = 17.010007503399553
(.venv) itssyoma@MacBook-Air-Sema megarepo_22 % "/Users/itssyoma/Yandex.Disk.lo
dex.Disk.localized/Учеба/Основы программной инженерии/megarepo_22/example1.py"
Value of x? 4
y = 5.0
(.venv) itssyoma@MacBook-Air-Sema megarepo_22 % "/Users/itssyoma/Yandex.Disk.lo
dex.Disk.localized/Учеба/Основы программной инженерии/megarepo_22/example1.py"
Value of x? 19
y = -360.850122790337
```

Рисунок 2 – Пример №1. Код и результат выполнения программы.

4. Проработали пример №2.

example2.py > ...

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys


if __name__ == '__main__':
    n = int(input("Введите номер месяца: "))

    if n == 1 or n == 2 or n == 12:
        print("Зима")
    elif n == 3 or n == 4 or n == 5:
        print("Весна")
    elif n == 6 or n == 7 or n == 8:
        print("Лето")
    elif n == 9 or n == 10 or n == 11:
        print("Осень")
    else:
        print("Ошибка!", file=sys.stderr)
        exit(1)

```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ

```
(.venv) itssyoma@MacBook-Air-Sema megarepo_22 % "/Users/itssyoma/Yandex.Disk.loc
dex.Disk.localized/Учеба/Основы программной инженерии/megarepo_22/example2.py"
Введите номер месяца: 2
Зима
(.venv) itssyoma@MacBook-Air-Sema megarepo_22 % "/Users/itssyoma/Yandex.Disk.loc
dex.Disk.localized/Учеба/Основы программной инженерии/megarepo_22/example2.py"
Введите номер месяца: 11
Осень
(.venv) itssyoma@MacBook-Air-Sema megarepo_22 % "/Users/itssyoma/Yandex.Disk.loc
dex.Disk.localized/Учеба/Основы программной инженерии/megarepo_22/example2.py"
Введите номер месяца: -13
Ошибка!
```

Рисунок 3 – Пример №2. Код и результат выполнения программы.

4. Проработали пример №3.

example3.py > ...

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import math


if __name__ == '__main__':
    n = int(input("Value of n? "))
    x = float(input("Value of x? "))

    S = 0.0

    for k in range(1, n + 1):
        a = math.log(k * x) / (k * k)
        S += a

    print(f"S = {S}")

```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ

```
(.venv) itssyoma@MacBook-Air-Sema megarepo_22 % "/Users/itssyoma/Yandex.Disk.lo
dex.Disk.localized/Учеба/Основы программной инженерии/megarepo_22/example3.py"
Value of n? 3
Value of x? 5
S = 2.485978652471746
(.venv) itssyoma@MacBook-Air-Sema megarepo_22 % "/Users/itssyoma/Yandex.Disk.lo
dex.Disk.localized/Учеба/Основы программной инженерии/megarepo_22/example3.py"
Value of n? 14
Value of x? 86
S = 7.704250854226356
(.venv) itssyoma@MacBook-Air-Sema megarepo_22 % "/Users/itssyoma/Yandex.Disk.lo
dex.Disk.localized/Учеба/Основы программной инженерии/megarepo_22/example3.py"
Value of n? 0
Value of x? 6
S = 0.0
```

Рисунок 4 – Пример №3. Код и результат выполнения программы.

5. Проработали пример №4 и составили UML-диаграмму деятельности.

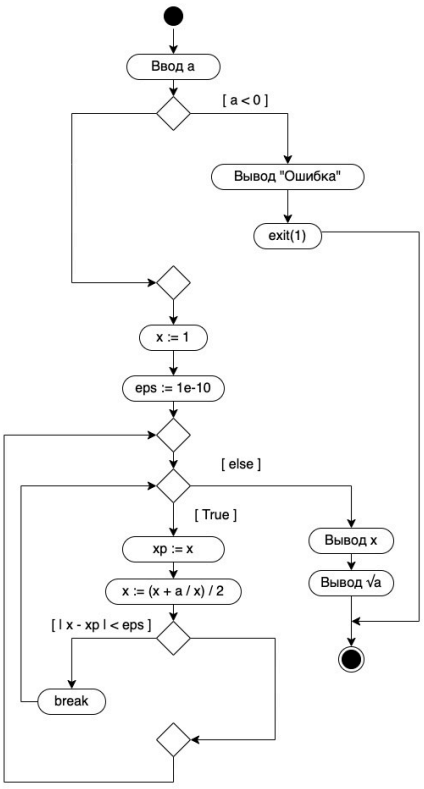

Рисунок 5 – Пример №4. UML-диаграмма деятельности.

example4.py > ...

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import math
import sys


if __name__ == '__main__':
    a = float(input("Value of a? "))
    if a < 0:
        print("Illegal value of a", file=sys.stderr)
        exit(1)

    x, eps = 1, 1e-10
    while True:
        xp = x
        x = (x + a / x) / 2
        if math.fabs(x - xp) < eps:
            break

    print(f"x = {x}\nX = {math.sqrt(a)}")

```

```
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ  ПОРТЫ

(.venv) itssyoma@MacBook-Air-Sema megarepo_22 % "/Users/itssyoma/Yandex.Disk.loc
dex.Disk.localized/Учеба/Основы программной инженерии/megarepo_22/example4.py"
Value of a? 3
x = 1.7320508075688772
X = 1.7320508075688772
(.venv) itssyoma@MacBook-Air-Sema megarepo_22 % "/Users/itssyoma/Yandex.Disk.loc
dex.Disk.localized/Учеба/Основы программной инженерии/megarepo_22/example4.py"
Value of a? 18
x = 4.242640687119286
X = 4.242640687119285
(.venv) itssyoma@MacBook-Air-Sema megarepo_22 % "/Users/itssyoma/Yandex.Disk.loc
dex.Disk.localized/Учеба/Основы программной инженерии/megarepo_22/example4.py"
Value of a? -6
Illegal value of a
```

Рисунок 6 – Пример №4. Код и результат выполнения программы.

5. Проработали пример №5 и составили UML-диаграмму деятельности.

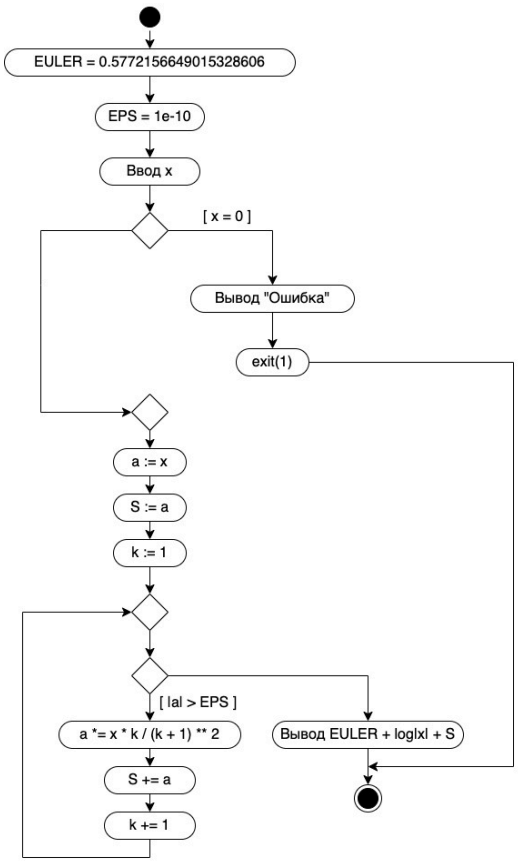

Рисунок 7 – Пример №5. UML-диаграмма деятельности.

example5.py > ...

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import math
import sys

# Постоянная Эйлера.
EULER = 0.5772156649015328606
# Точность вычислений.
EPS = 1e-10


if __name__ == '__main__':
    x = float(input("Value of x? "))
    if x == 0:
        print("Illegal value of x", file=sys.stderr)
        exit(1)

    a = x
    S, k = a, 1

    # Найти сумму членов ряда.
    while math.fabs(a) > EPS:
        a *= x * k / (k + 1) ** 2
        S += a
        k += 1

    # Вывести значение функции.
    print(f"Ei({x}) = {EULER + math.log(math.fabs(x)) + S}")
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ

```
(.venv) itssyoma@MacBook-Air-Sema megarepo_22 % "/Users/itssyoma/Yandex.D
n/python" "/Users/itssyoma/Yandex.Disk.localized/Учеба/Основы программно
Value of x? 4
Ei(4.0) = 19.63087447005282
(.venv) itssyoma@MacBook-Air-Sema megarepo_22 % "/Users/itssyoma/Yandex.D
n/python" "/Users/itssyoma/Yandex.Disk.localized/Учеба/Основы программно
Value of x? -3
Ei(-3.0) = -0.013048381085575267
(.venv) itssyoma@MacBook-Air-Sema megarepo_22 % "/Users/itssyoma/Yandex.D
n/python" "/Users/itssyoma/Yandex.Disk.localized/Учеба/Основы программно
Value of x? 0
Illegal value of x
```

Рисунок 8 – Пример №5. Код и результат выполнения программы.

6. Выполнили задание №1 (вариант №1). Дано натуральное число. Вывести на экран фразу Мне n лет, учитывая, что при некоторых значениях n слово «лет» надо заменить на слово «год» или «года».

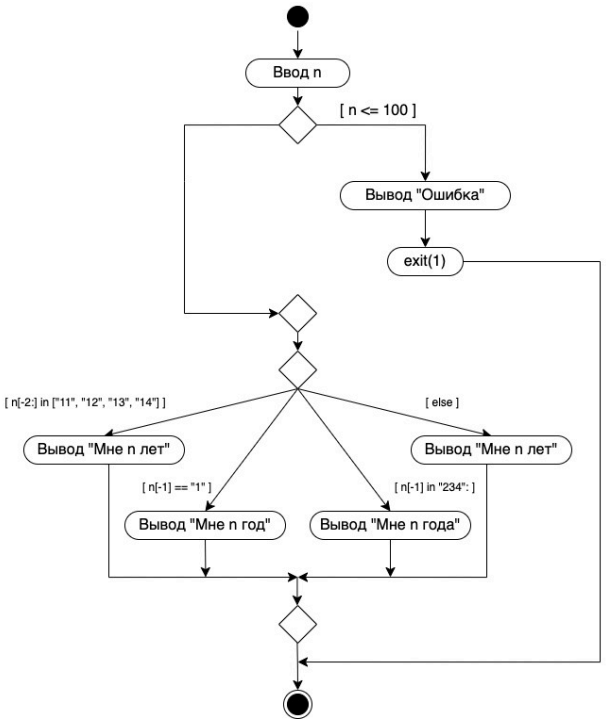

Рисунок 9 – Задание №1. UML-диаграмма деятельности.

task1.py > ...

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys


if __name__ == "__main__":
    n = str(input("Введите возраст: "))

    if int(n) <= 100:
        print("Недопустимое значение возраста!", file=sys.stderr)
        exit(1)


    if n[-2:] in ["11", "12", "13", "14"]:
        print(f"Мне {n} лет")
    elif n[-1] == "1":
        print(f"Мне {n} год")
    elif n[-1] in "234":
        print(f"Мне {n} года")
    else:
        print(f"Мне {n} лет")

```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ

```
(.venv) itssyoma@MacBook-Air-Sema megarepo_22 % "/Users/itssyoma/Yandex.Disk
n/python" "/Users/itssyoma/Yandex.Disk.localized/Учеба/Основы программной ин
Введите возраст: 19
Недопустимое значение возраста!
(.venv) itssyoma@MacBook-Air-Sema megarepo_22 % "/Users/itssyoma/Yandex.Disk
n/python" "/Users/itssyoma/Yandex.Disk.localized/Учеба/Основы программной ин
Введите возраст: 312
Мне 312 лет
(.venv) itssyoma@MacBook-Air-Sema megarepo_22 % "/Users/itssyoma/Yandex.Disk
n/python" "/Users/itssyoma/Yandex.Disk.localized/Учеба/Основы программной ин
Введите возраст: 541
Мне 541 год
(.venv) itssyoma@MacBook-Air-Sema megarepo_22 % "/Users/itssyoma/Yandex.Disk
n/python" "/Users/itssyoma/Yandex.Disk.localized/Учеба/Основы программной ин
Введите возраст: 763
Мне 763 года
(.venv) itssyoma@MacBook-Air-Sema megarepo_22 % "/Users/itssyoma/Yandex.Disk
n/python" "/Users/itssyoma/Yandex.Disk.localized/Учеба/Основы программной ин
Введите возраст: 718
Мне 718 лет
```

Рисунок 10 – Задание №1. Код и результат выполнения программы.

7. Выполнили задание №2 (вариант №1). В японском календаре был принят 60-летний цикл, состоящий из пяти 12-летних подциклов. Внутри подцикла года носили названия животных мыши, коровы, тигра, зайца, дракона, змеи, лошади, овцы, обезьяны, курицы, собаки и свиньи. Попарно года в подцикле обозначались названиями цвета: зеленый, красный, желтый, белый и черный. По номеру года определить его название по японскому календарю, считая за начало очередного цикла 1984 год - год зеленой мыши (1985 - год зеленой коровы, 1986 - год красного тигра, 1987 - год красного зайца и т. д.).

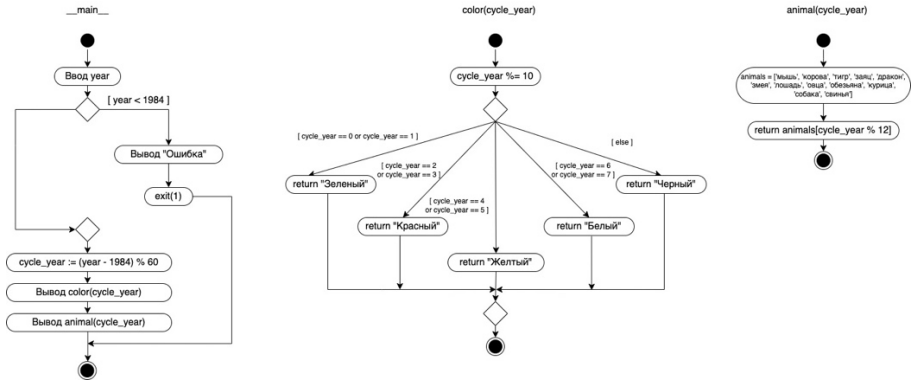

Рисунок 11 – Задание №2. UML-диаграмма деятельности.

task2.py > ...

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys


def color(cycle_year):
    cycle_year %= 10
    if cycle_year == 0 or cycle_year == 1:
        return "Зеленый"
    if cycle_year == 2 or cycle_year == 3:
        return "Красный"
    if cycle_year == 4 or cycle_year == 5:
        return "Желтый"
    if cycle_year == 6 or cycle_year == 7:
        return "Белый"
    else:
        return "Черный"

def animal(cycle_year):
    animals = ['мышь', 'корова', 'тигр', 'заяц', 'дракон', 'змея', 'лошадь', 'овца', 'обезьяна', 'курица', 'собака', 'свинья']
    return animals[cycle_year % 12]

if __name__ == "__main__":
    year = int(input("Введите год: "))
    if year < 1984:
        print("Недопустимое значение года!", file=sys.stderr)
        exit(1)

    cycle_year = (year - 1984) % 60
    print(f"{year} год по японскому календарю\nГод в цикле: {cycle_year+1}\nЦвет: {color(cycle_year)}\nЖивотное: {animal(cycle_year)}")
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ

```
Введите год: 1984
1984 год по японскому календарю
Год в цикле: 1
Цвет: Зеленый
Животное: мышь
(.venv) itssyoma@MacBook-Air-Sema megarepo_22 % "/Users/itssyoma/Yandex.Disk.localized/Учеба/Основы программной инженерии/megarepo_22/.venv/bin/python" "/Users/itssyoma/Yandex.Disk.localized/Учеба/Основы программной инженерии/megarepo_22/task2.py"
Введите год: 2023
2023 год по японскому календарю
Год в цикле: 40
Цвет: Черный
Животное: заяц
```

Рисунок 12 – Задание №2. Код и результат выполнения программы.

8. Выполнили задание №3 (вариант №1). Найти все трехзначные натуральные числа, сумма цифр которых равна их произведению.

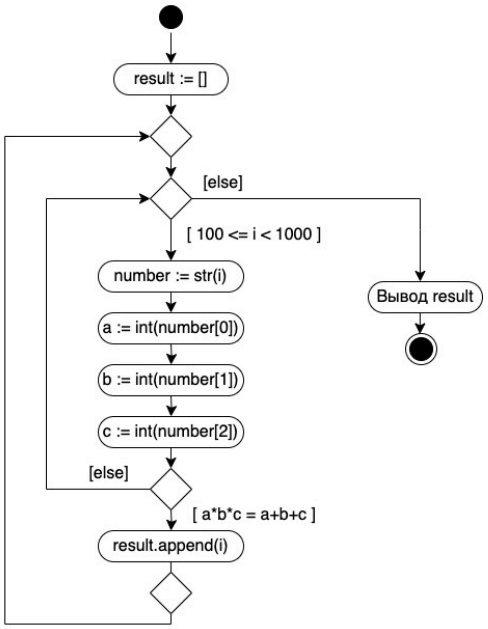

Рисунок 13 – Задание №3. UML-диаграмма деятельности.

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


if __name__ == "__main__":
    result = []

    for i in range(100, 1000):
        number = str(i)
        a = int(number[0])
        b = int(number[1])
        c = int(number[2])
        if a*b*c == a+b+c:
            result.append(i)

    print(result)
```

```
ПРОБЛЕМЫ   ВЫХОДНЫЕ ДАННЫЕ   КОНСОЛЬ ОТЛАДКИ

(.venv) itssyoma@MacBook-Air-Sema megarepo_22 % "/Us
n/python" "/Users/itssyoma/Yandex.Disk.localized/Уче
[123, 132, 213, 231, 312, 321]
(.venv) itssyoma@MacBook-Air-Sema megarepo_22 %
```

Рисунок 14 – Задание №3. Код и результат выполнения программы.

9. Решили задание повышенной сложности (вариант №1 – интегральный синус). Составить UML-диаграмму деятельности, программу и произвести вычисления вычисление значения специальной функции по ее разложению в ряд с точностью, аргумент функции вводится с клавиатуры.

$$\mathrm{Si}(x) = \int_0^x \frac{\sin t}{t}\,dt = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)(2n+1)!}.$$

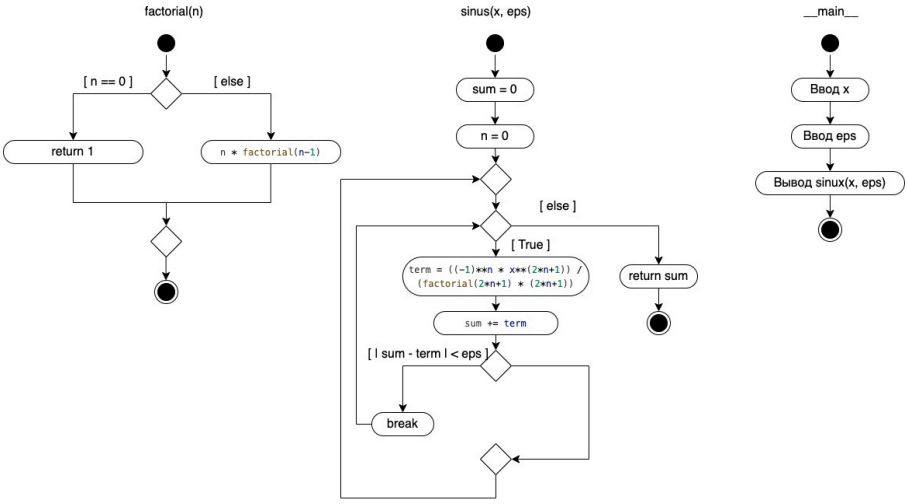

Рисунок 15 – Задание повышенной сложности.

UML-диаграмма деятельности.

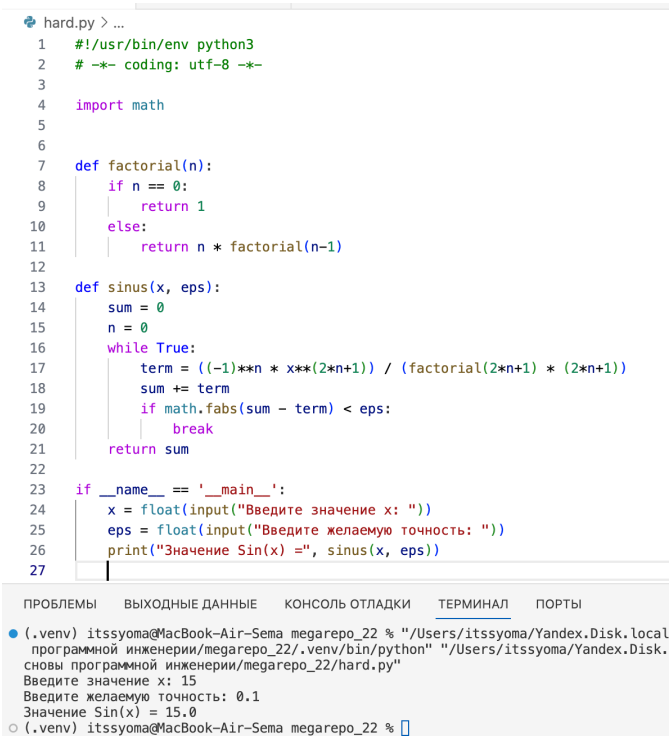

Рисунок 16 – Задание повышенной сложности.

Код и результат выполнения программы.

Ответы на контрольные вопросы

1. Для чего нужны диаграммы деятельности UML?

Диаграммы деятельности UML используются для визуализации и описания последовательности действий или процессов в системе. Они помогают понять, как система работает и какие операции выполняются в различных ситуациях.

2. Что такое состояние действия и состояние деятельности?

Состояние действия представляет выполнение определенного действия или операции в процессе работы системы. Состояние деятельности представляет выполнение набора связанных действий или операций.

3. Какие нотации существуют для обозначения переходов и ветвлений в диаграммах деятельности?

Для обозначения переходов и ветвлений в диаграммах деятельности используются стрелки и условные обозначения. Например, стрелка с заливкой обозначает успешный переход, а стрелка с пунктирной линией -

неуспешный переход. Условные обозначения могут использоваться для указания условий, при которых происходят переходы или ветвления.

4. Какой алгоритм является алгоритмом разветвляющейся структуры?

Алгоритм разветвляющейся структуры является алгоритмом, в котором выполняется проверка определенного условия, и в зависимости от результата проверки происходит разветвление выполнения программ.

5. Чем отличается разветвляющийся алгоритм от линейного?

Разветвляющийся алгоритм отличается от линейного тем, что в разветвляющемся алгоритме есть возможность выбора различных путей выполнения в зависимости от условий, в то время как в линейном алгоритме выполнение происходит последовательно без разветвлений.

6. Что такое условный оператор? Какие существуют его формы?

Условный оператор - это конструкция, которая позволяет выполнять определенные действия в зависимости от условий. Существуют две формы условного оператора: if-else (если-иначе) и if-elif-else (если-иначе-если).

7. Какие операторы сравнения используются в Python?

В Python используются следующие операторы сравнения: равно (==), не равно (!=), больше (>), меньше (<), больше или равно (>=), меньше или равно (<=).

8. Что называется простым условием? Приведите примеры.

Простое условие - это условие, которое содержит только одно сравнение. Например, x > 5 или y == "hello".

9. Что такое составное условие? Приведите примеры.

Составное условие - это условие, которое содержит несколько сравнений, объединенных логическими операторами. Например, x > 5 and y < 10 или x == 0 or y != "hello".

10. Какие логические операторы допускаются при составлении сложных условий?

При составлении сложных условий в Python можно использовать логические операторы and (и), or (или) и not (не).

11. Может ли оператор ветвления содержать внутри себя другие ветвления?

Да, оператор ветвления может содержать внутри себя другие ветвления. Это называется вложенными ветвлениями.

12. Какой алгоритм является алгоритмом циклической структуры?

Алгоритм циклической структуры является алгоритмом, в котором определенный блок кода выполняется несколько раз до тех пор, пока выполняется определенное условие.

13. Типы циклов в языке Python.

В языке Python существуют следующие типы циклов: цикл while (пока), цикл for (для) и цикл do-while (делать-пока).

14. Назовите назначение и способы применения функции range.

Функция range используется для создания последовательности чисел. Она может принимать один, два или три аргумента. Первый аргумент указывает начало последовательности, второй - конец последовательности (не включая его), а третий - шаг, с которым будут генерироваться числа.

15. Как с помощью функции range организовать перебор значений от 15 до 0 с шагом 2?

Для организации перебора значений от 15 до 0 с шагом 2 с помощью функции range можно использовать следующий код: for i in range(15, -1, -2).

16. Могут ли быть циклы вложенными?

Да, циклы могут быть вложенными, то есть один цикл может находиться внутри другого цикла.

17. Как образуется бесконечный цикл и как выйти из него?

Бесконечный цикл образуется, когда условие выхода из цикла никогда не становится ложным. Чтобы выйти из бесконечного цикла, можно использовать оператор break, который прерывает выполнение цикла.

18. Для чего нужен оператор break ?

Оператор break используется для прерывания выполнения цикла и выхода из него. Когда оператор break выполняется, управление передается за пределы цикла.

19. Где употребляется оператор continue и для чего он используется?

Оператор continue используется для пропуска текущей итерации цикла и перехода к следующей итерации. Он позволяет пропустить выполнение оставшейся части кода в текущей итерации и перейти к следующей итерации.

20. Для чего нужны стандартные потоки stdout и stderr?

Стандартные потоки stdout (стандартный вывод) и stderr (стандартный поток ошибок) используются для вывода информации в консоль. stdout используется для вывода обычной информации, а stderr - для вывода сообщений об ошибках или предупреждений.

21. Как в Python организовать вывод в стандартный поток stderr?

В Python вывод в стандартный поток stderr можно организовать с помощью модуля sys. Например, можно использовать следующий код: import sys; print("Ошибка", file=sys.stderr).

22. Каково назначение функции exit?

Функция exit используется для завершения работы программы. Она принимает необязательный аргумент - код завершения, который может быть использован для передачи информации об ошибке или успешном выполнении программы.