

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 практической работе №2.2
Дисциплины «Программирование на Python»

Выполнил:

Пустяков Андрей Сергеевич

2 курс, группа ИВТ-б-о-22-1,

09.03.01 «Информатика и
вычислительная техника (профиль)
«Программное обеспечение средств
вычислительной
техники и автоматизированных
систем», очная форма обучения

(подпись)

Руководитель практики:

Воронкин Р. А. кандидат технических
наук, доцент, доцент кафедры
инфокоммуникаций

(подпись)

Ставрополь, 2023 г.

Тема: Условные операторы и циклы в языке Python.

Цель: приобрести навыки программирования разветвляющихся алгоритмов и алгоритмов циклической структуры. Освоить операторы языка Python версии 3.x «if», «while», «for», «break», «continue», позволяющих реализовать разветвляющиеся алгоритмы и алгоритмы циклической структуры.

Ход работы:

Проработка примеров лабораторной работы в отдельных модулях в созданном репозитории.

Выполнение примера №1, вычисление значения функции по заданным параметрам «example_1» и результаты работы программы при различных исходных данных (рис. 1, 2).

```
#!/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}")
```

Рисунок 1 – Код программы примера №1.

```
C:\Users\Andrey\AppData\Local\Programs\Python\Py
Value of x? 1
y = 2.0

Process finished with exit code 0
```

```
C:\Users\Andrey\AppData\Local\Programs\Python
Value of x? -1
y = 2.5403023058681398

Process finished with exit code 0
```

```
C:\Users\Andrey\AppData\Local\Programs\Py
Value of x? 10
y = -100.54402111088937

Process finished with exit code 0
```

Рисунок 2 – Результаты работы программы примера №1.

Проработка примера №2, определение времени года по номеру месяца «example_2», результаты работы программы (рис. 3, 4).

```
#!/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)
```

Рисунок 3 – Код программы примера №2.

```
Введите номер месяца: 1
Зима
```

Process finished with **exit** code 0

```
Введите номер месяца: 3
Весна
```

Process finished with **exit** code 0

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39-64\python.exe
Введите номер месяца: 7
Лето
```

Process finished with **exit** code 0

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\python.exe
Введите номер месяца: 10
Осень
Process finished with exit code 0
```

Рисунок 4 – Результаты работы программы.

Проработка примера №3, необходимо вычислить конечную сумму «example_3», результаты работы программы (рис. 5, 6).

```
#!/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}")
```

Рисунок 5 – Код программы примера №3.

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\python.exe
Value of n? 100
Value of x? 2
S = 2.0150106074560328
```

Рисунок 6 – Результат работы программы примера №3.

Проработка примера №4, необходимо найти квадратный корень из вводимого числа заданной точности по рекуррентному соотношению «example_4», результаты работы программы (рис. 7, 8).

```

#!/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)}")

```

Рисунок 7 – Код программы примера №4.

```

C:\Users\Andrey\AppData\Local\Programs\Python\Python38\python.exe
Value of a? 30
Value of x? 5.477225575051661
S = 5.477225575051661

C:\Users\Andrey\AppData\Local\Programs\Python\Python38\python.exe
Value of a? 50
Value of x? 7.0710678118654755
S = 7.0710678118654755

```

Рисунок 8 – Результаты работы программы примера №4.

UML-диаграмма деятельности примера №4 (рис. 9).

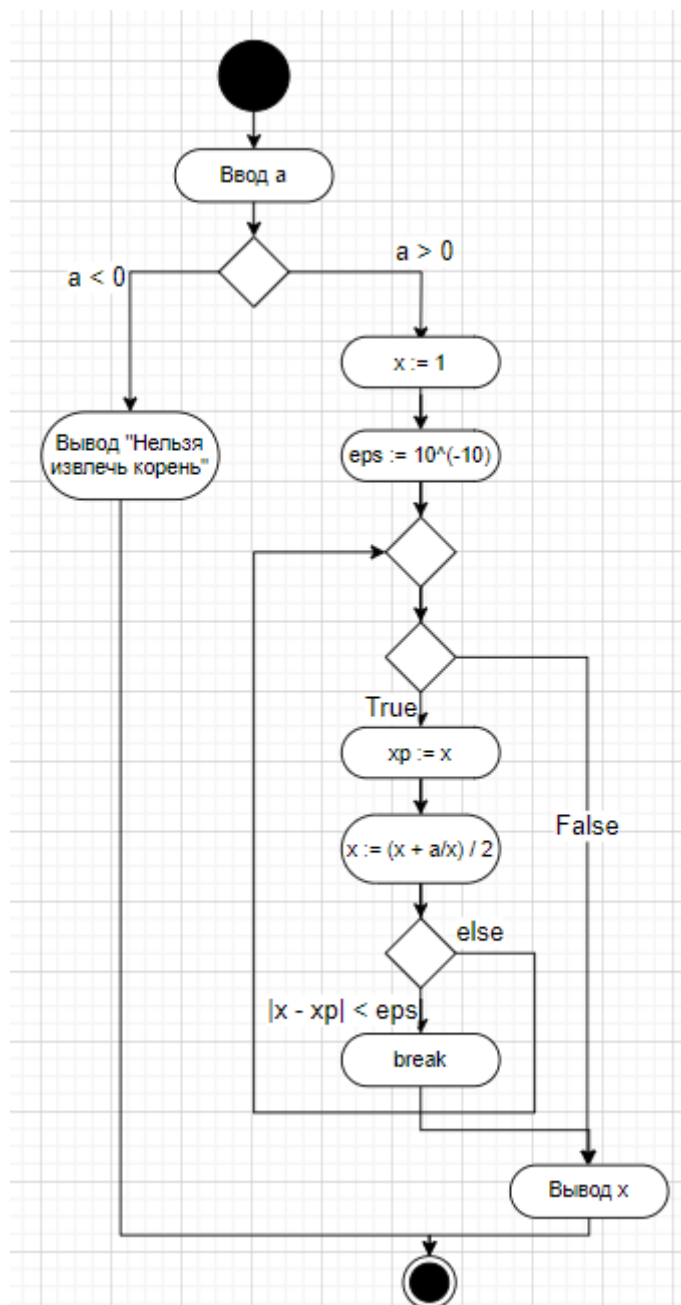


Рисунок 9 – UML-диаграмма примера №4.

Проработка примера №5, необходимо вычислить значение интегральной показательной функции «example_5», результаты работы программы (рис. 10, 11).

```
#!/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}")
```

Рисунок 10 – Код программы примера №4.

```
C:\Users\Andrey\AppData\Local\Programs\Python
Value of x? 5
Ei(5.0) = 40.18527535579794

Process finished with exit code 0

C:\Users\Andrey\AppData\Local\Programs\Python
Value of x? -9
Ei(-9.0) = -1.2447337248921997e-05

Process finished with exit code 0
```

Рисунок 11 – Результаты работы промера №4.

UML-диаграмма деятельности примера №5 (рис. 12).

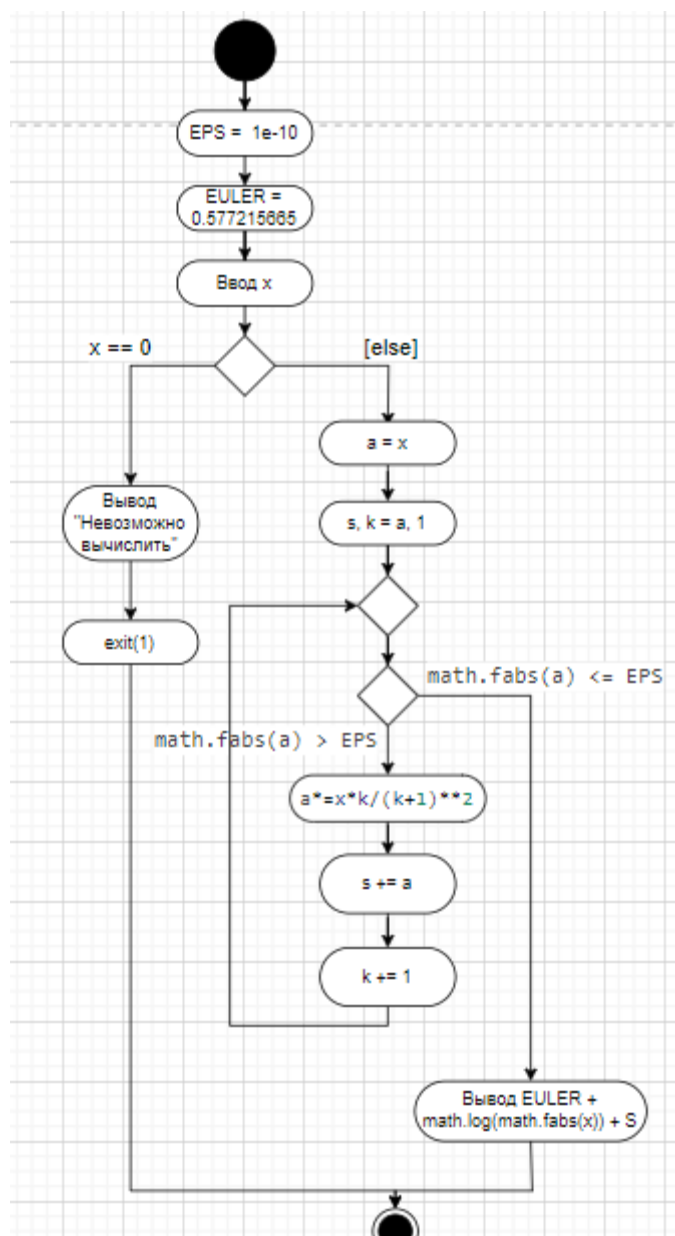


Рисунок 12 – UML-диаграмма примера №4.

Выполнение индивидуальных заданий:

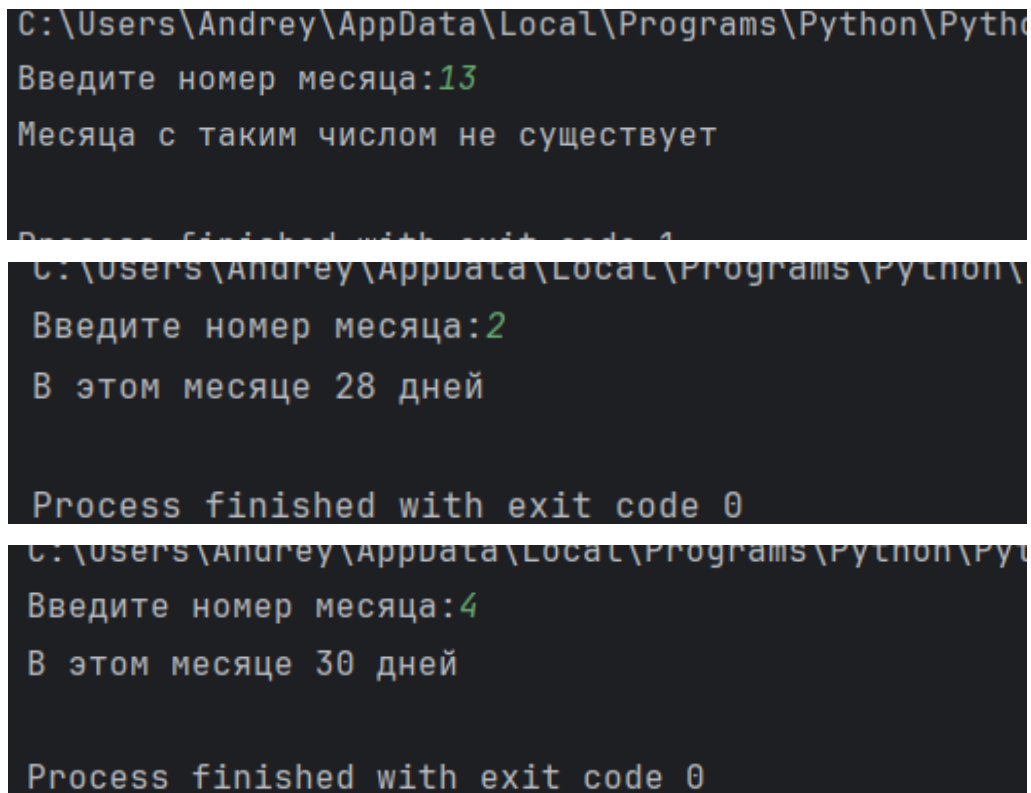
Задание 1.

Необходимо написать программу, которая выводит количество дней в месяце по заданному номеру месяца (Вариант 15 (2)). Код программы индивидуального задания №1 и результаты работы программы (рис. 13, 14).

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

if __name__ == '__main__':
    m = int(input("Введите номер месяца:"))
    if m < 1 or m > 12:
        print("Месяца с таким числом не существует")
        exit(1)
    if m == 1 or m == 3 or m == 5 or m == 7 or m == 8 or m == 10 or m == 12:
        days_number = 31
    elif m == 4 or m == 6 or m == 9 or m == 11:
        days_number = 30
    else:
        days_number = 28
    print("В этом месяце", days_number, "дней")
```

Рисунок 13 – Код программы индивидуального задания №1.



The figure consists of three vertically stacked screenshots of a terminal window showing the execution of the Python program. Each screenshot starts with the file path 'C:\Users\Andrey\AppData\Local\Programs\Python\Python...' and the prompt 'Введите номер месяца:'. The first screenshot shows input '13' and output 'Месяца с таким числом не существует', followed by 'Process finished with exit code 1'. The second screenshot shows input '2' and output 'В этом месяце 28 дней', followed by 'Process finished with exit code 0'. The third screenshot shows input '4' and output 'В этом месяце 30 дней', followed by 'Process finished with exit code 0'.

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python...
Введите номер месяца:13
Месяца с таким числом не существует
Process finished with exit code 1

C:\Users\Andrey\AppData\Local\Programs\Python\Python...
Введите номер месяца:2
В этом месяце 28 дней
Process finished with exit code 0

C:\Users\Andrey\AppData\Local\Programs\Python\Python...
Введите номер месяца:4
В этом месяце 30 дней
Process finished with exit code 0
```

Рисунок 14 – Результаты работы программы индивидуального задания №1.

UML-диаграмма индивидуального задания №1 (рис. 15).

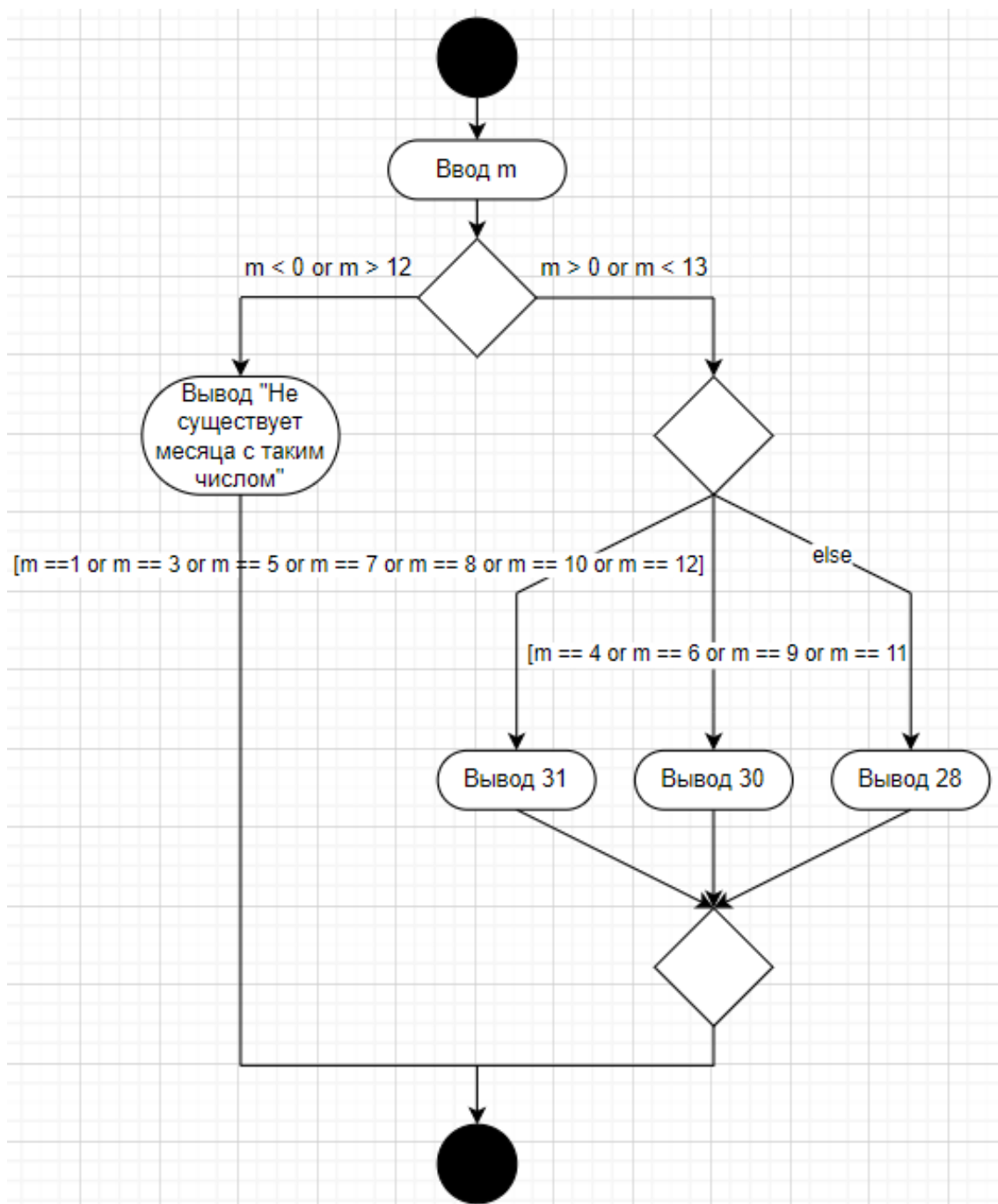


Рисунок 15 – UML-диаграмма индивидуального задания №1.

Задание 2.

Необходимо создать программу решения квадратного уравнения, которая выводит также комплексные решения (Вариант 15). Код программы индивидуального задания №2 и результаты работы программы для различных корней уравнения (рис. 16, 17).

```

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

# Импорт модуля, который позволяет вычислить корни отрицательных чисел
import cmath
import math

if __name__ == '__main__':
    coef_a = float(input("Введите коэффициент a: "))
    coef_b = float(input("Введите коэффициент b: "))
    coef_c = float(input("Введите коэффициент c: "))
    # Вычисление дискриминанта
    discriminant = (coef_b ** 2) - (4 * coef_a * coef_c)
    # Вычисление корней квадратного уравнения в зависимости от дискриминанта
    if discriminant > 0:
        root_1 = (-coef_b + math.sqrt(discriminant)) / (2 * coef_a)
        root_2 = (-coef_b - math.sqrt(discriminant)) / (2 * coef_a)
    elif discriminant == 0:
        root_1 = -coef_b / (2 * coef_a)
        root_2 = -coef_b / (2 * coef_a)
    else:
        root_1 = (-coef_b + cmath.sqrt(discriminant)) / (2 * coef_a)
        root_2 = (-coef_b - cmath.sqrt(discriminant)) / (2 * coef_a)
    # Вывод результатов в зависимости от видов корней
    if discriminant > 0:
        print("Уравнение имеет два корня:")
        print("Корень x1 =", root_1)
        print("Корень x2 =", root_2)
    elif discriminant == 0:
        print("Уравнение имеет один корень:")
        print("Корень x =", root_1)
    else:
        print("Уравнение имеет два комплексных корня:")
        print("Корень W1 =", root_1)
        print("Корень W2 =", root_2)

```

Рисунок 16 – Код программы индивидуального задания №2.

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\
Введите коэффициент a: 1
Введите коэффициент b: 5
Введите коэффициент c: 1
Уравнение имеет два корня:
Корень x1 = -0.20871215252208009
Корень x2 = -4.7912878474779195
```

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\
Введите коэффициент a: 1
Введите коэффициент b: 2
Введите коэффициент c: 1
Уравнение имеет один корень:
Корень x = -1.0
```

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\
Введите коэффициент a: 5
Введите коэффициент b: 1
Введите коэффициент c: 5
Уравнение имеет два комплексных корня:
Корень W1 = (-0.1+0.99498743710662j)
Корень W2 = (-0.1-0.99498743710662j)
```

Рисунок 17 – Результаты работы программы индивидуального задания №2.

UML-диаграмма деятельности индивидуального задания №2 (рис. 18).

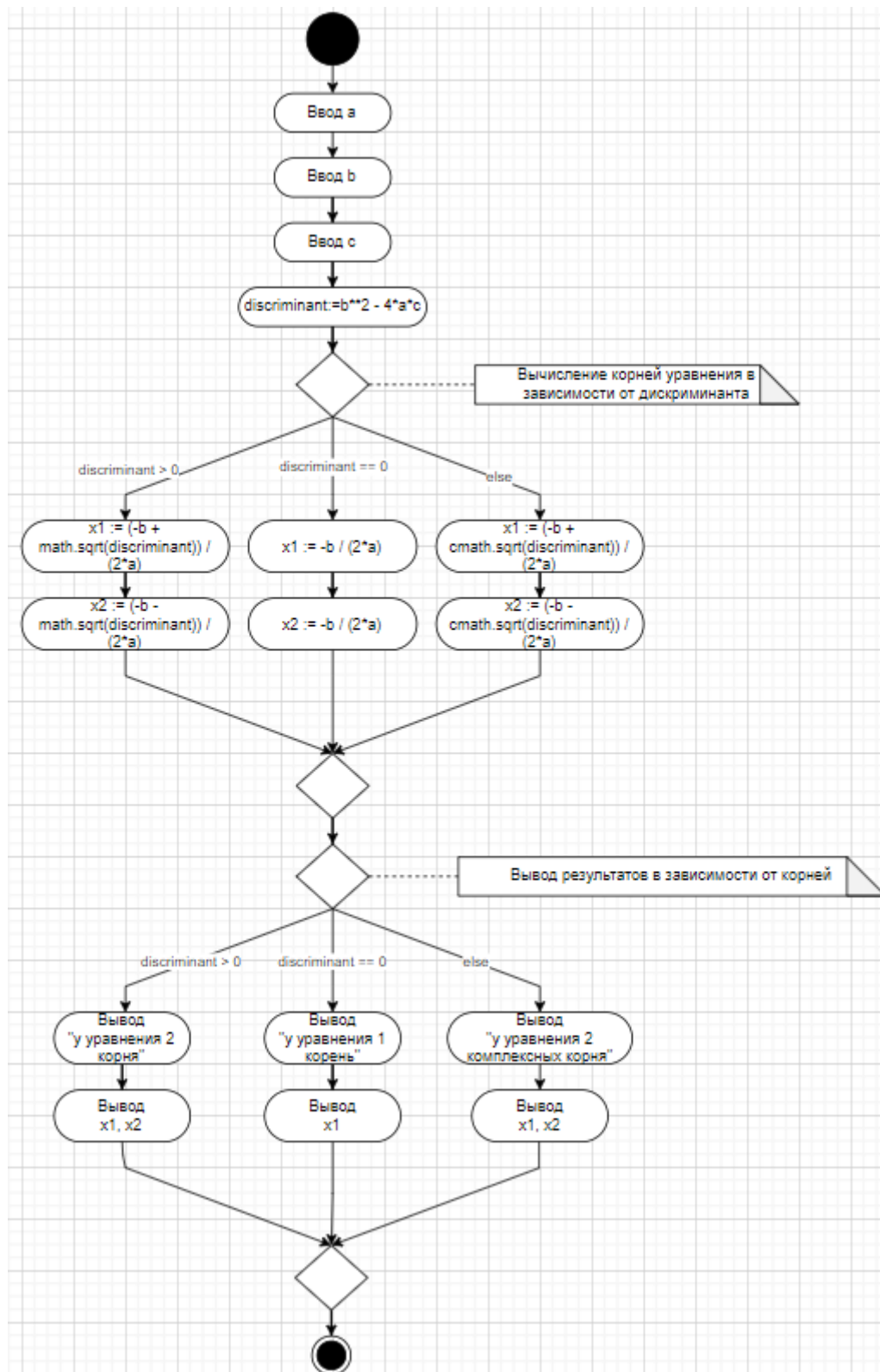


Рисунок 18 – UML-диаграмма индивидуального задания №2.

Задание 3.

Необходимо вычислить сумму всех n-значных чисел кратных k ($1 < n < 4$) (Вариант 15). Код программы индивидуального задания №3 и результаты работы программы при различных значениях n (рис. 19, 20).

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

if __name__ == '__main__':
    number_n = int(input("Введите число n:"))
    number_k = int(input("Введите число k:"))
    if number_n < 1 or number_n > 4:
        print("n не удовлетворяет условиям 1<n<4")
        exit(1)
    sum_k = 0
    for i in range(10**(number_n-1), 10**number_n):
        if i % number_k == 0:
            sum_k += i
        i += 1
    print("Сумма", number_n, "значных чисел =", sum_k)
```

Рисунок 19 – Код программы индивидуального задания №3.

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\
Введите число n:1
Введите число k:2
Сумма 1 значных чисел = 20
```

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\
Введите число n:2
Введите число k:30
Сумма 2 значных чисел = 180
```

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\  
Введите число n:3  
Введите число k:45  
Сумма 3 значных чисел = 11250
```

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\  
Введите число n:4  
Введите число k:20  
Сумма 4 значных чисел = 2470500
```

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\  
Введите число n:5  
Введите число k:1  
n не удовлетворяет условиям  $1 < n < 4$ 
```

Рисунок 20 – Результаты работы программы индивидуального задания №3.

UML-диаграмма деятельности индивидуального задания №3 (рис. 21).

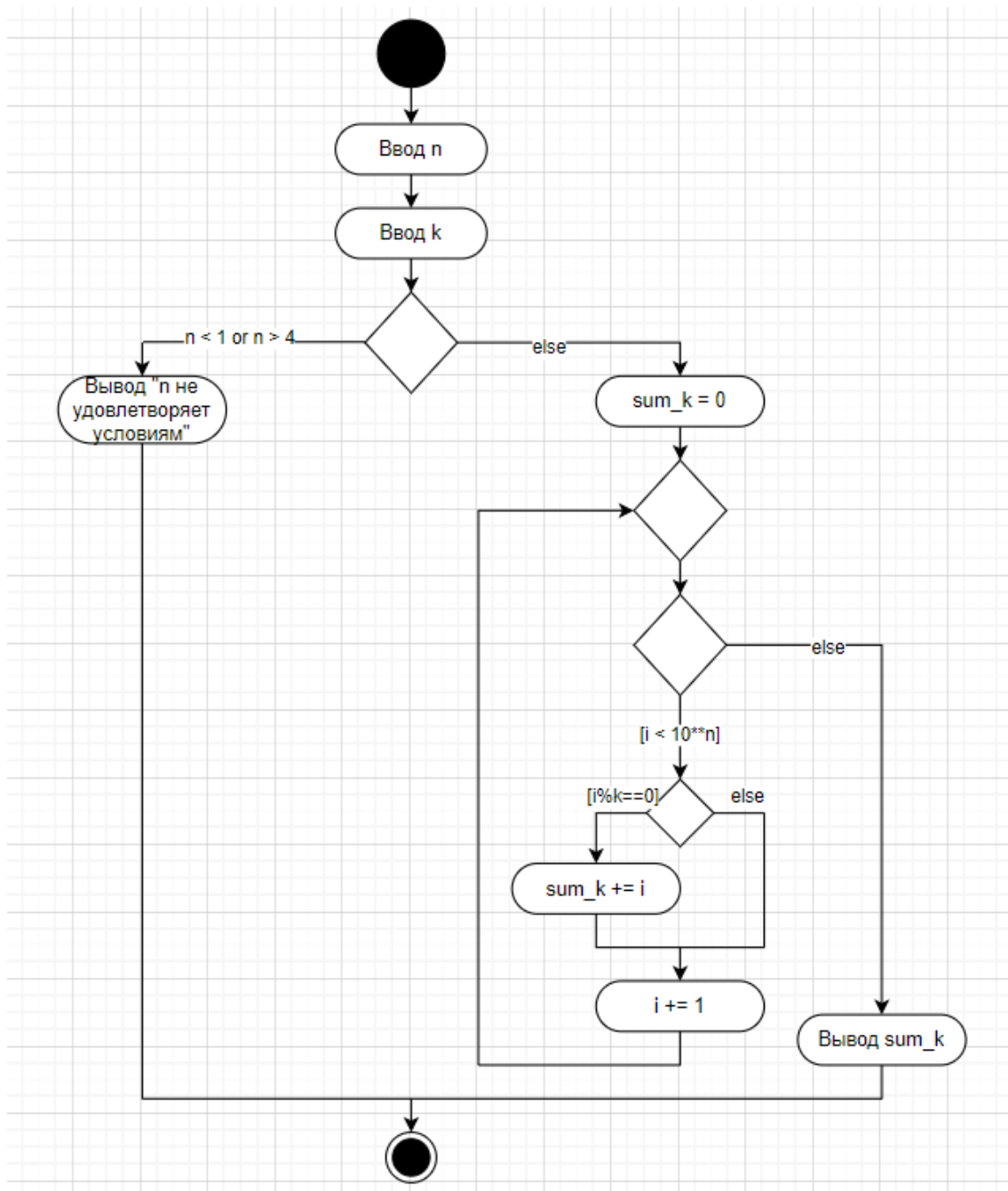


Рисунок 21 – UML-диаграмма индивидуального задания №3.

Задание повышенной сложности

Необходимо найти значение функции (дилогарифма, рис. 22) по ее разложению в ряд с заданной точностью $\varepsilon = 10^{-10}$, аргумент вводится с клавиатуры (Вариант 9):

$$f(x) = - \int_1^x \frac{\ln t}{t-1} dt = \sum_{k=1}^{\infty} \frac{(-1)^k (x-1)^k}{k^2}, \quad 0 \leq x \leq 2.$$

Рисунок 22 – Дилогарифм задания повышенной сложности.

Код программы задания повышенной сложности и результаты работы программы при различных значениях x (рис. 23, 24).

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

import math
import sys

# Точность вычислений.
EPS = 1e-10

if __name__ == '__main__':
    x = float(input("Введите аргумент x (0 <= x <= 2): "))
    if x < 0 or x > 2:
        print("Аргумент не подходит заданным условиям", file=sys.stderr)
        exit(1)

    a = 1 - x # Первый член ряда
    sum_row = a # Сумма членов ряда
    k = 1 # Счетчик
    # Найдем сумму членов ряда.
    while math.fabs(a) > EPS:
        a *= ((1 - x)*(k**2)) / ((k + 1)**2)
        sum_row += a
        k += 1
    # Вывести значение функции.
    print("Значение функции (дилогарифма) =", sum_row)
```

Рисунок 23 – Код программы задания повышенной сложности.

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\
Введите аргумент x ( $0 \leq x \leq 2$ ): 2
Значение функции (дилогарифма) = -0.822467033374095
```

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\
Введите аргумент x ( $0 \leq x \leq 2$ ): 1
Значение функции (дилогарифма) = 0.0
Process finished with exit code 0
```

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\
Введите аргумент x ( $0 \leq x \leq 2$ ): 0
Значение функции (дилогарифма) = 1.6449240668982423
Process finished with exit code 0
```

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\
Введите аргумент x ( $0 \leq x \leq 2$ ): 10
Аргумент не подходит заданным условиям
Process finished with exit code 1
```

Рисунок 24 – Результаты работы программы задания повышенной сложности.

UML-диаграмма деятельности задания повышенной сложности (рис. 25).

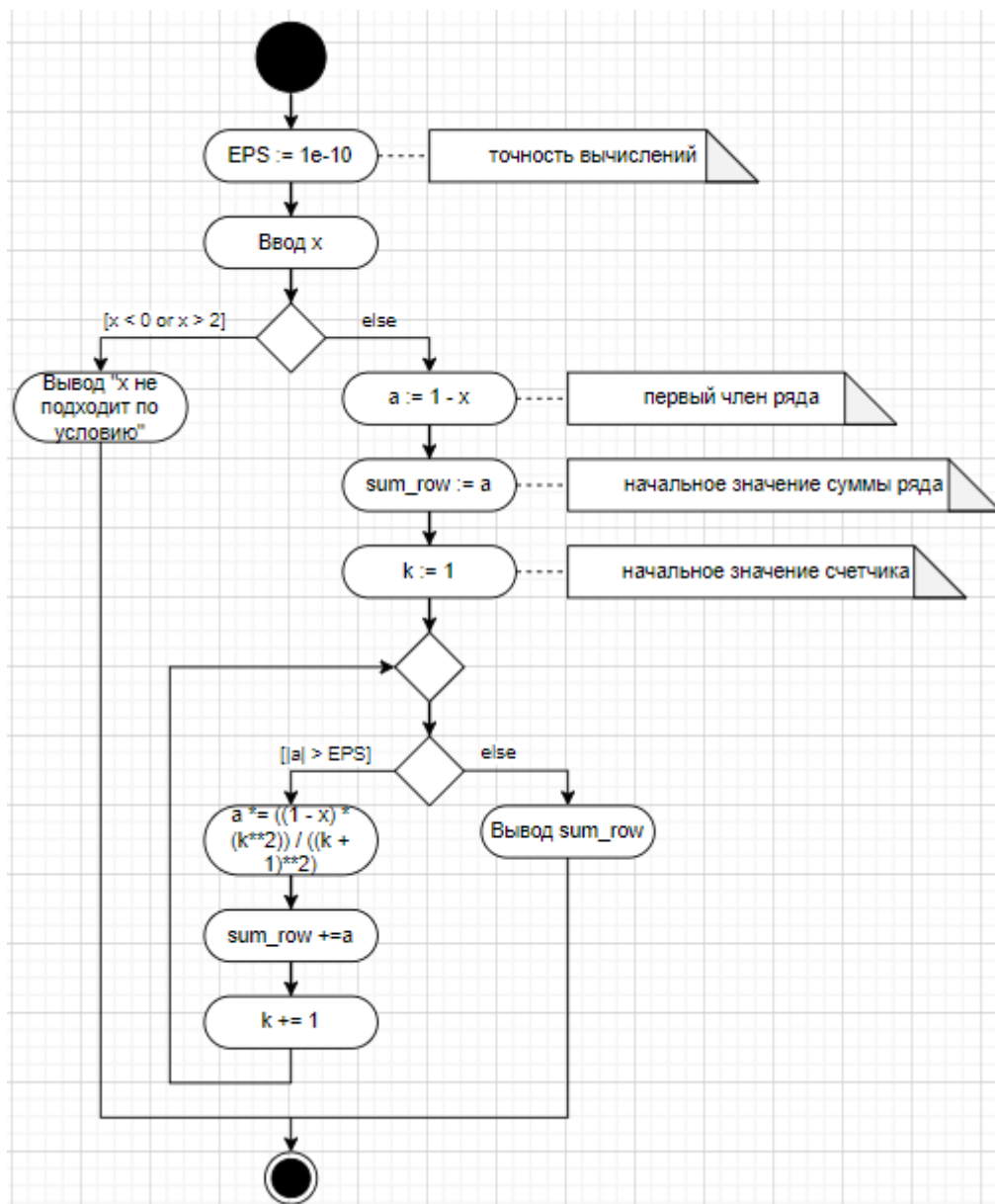


Рисунок 25 – UML-диаграмма задания повышенной сложности.

Ответы на контрольные вопросы:

1. Диаграммы деятельности — это один из пяти видов диаграмм, применяемых в UML для моделирования динамических аспектов поведения системы. Диаграмма деятельности — это блок-схема, которая показывает, как поток управления переходит от одной деятельности к другой, однако, по сравнению с последней, у ней есть явные преимущества: поддержка многопоточности и объектно-ориентированного проектирования. Эта диаграмма показывает поток переходов от одной деятельности к другой (Деятельность (Activity) — это продолжающийся во времени неатомарный

шаг вычислений в автомате). Деятельности в конечном счете приводят к выполнению некоего действия (Action), составленного из выполняемых атомарных вычислений, каждое из которых либо изменяет состояние системы, либо возвращает какое-то значение. Действие может заключаться в вызове другой операции, послке сигнала, создании или уничтожении объекта либо в простом вычислении. Графически диаграмма деятельности представляется в виде графа, имеющего вершины и ребра.

2. Все выполняемые атомарные вычисления (Вычислить выражение, в результате чего изменяется значение некоторого атрибута или возвращается некоторое значение; выполнить операцию над объектом - послать ему сигнал или даже создать его или уничтожить) называются состояниями действия, поскольку каждое из них есть состояние системы, представляющее собой выполнение некоторого действия. Внутри можно записывать произвольное выражение. Состояния действия не могут быть подвергнуты декомпозиции. Кроме того, они атомарны (Внутри них могут происходить различные события, но выполняемая в состоянии действия работа не может быть прервана). Обычно предполагается, что длительность одного состояния действия занимает неощутимо малое время. В противоположность этому состояния деятельности могут быть подвергнуты дальнейшей декомпозиции, вследствие чего выполняемую деятельность можно представить с помощью других диаграмм деятельности. Состояния деятельности не являются атомарными, то есть могут быть прерваны. Предполагается, что для их завершения требуется заметное время. Можно считать, что состояние действия — это частный вид состояния деятельности, а конкретнее — такое состояние, которое не может быть подвергнуто дальнейшей декомпозиции. А состояние деятельности можно представлять себе как составное состояние, поток управления которого включает только другие состояния деятельности и действий.

3. Для описания потока используются переходы, показывающие путь из одного состояния действия или деятельности в другое. В UML переход

представляется простой линией со стрелкой. Поток управления должен где-то начинаться и заканчиваться. Начальное состояние – закрашенный круг, в то время как конечное состояние – закрашенный круг внутри окружности. Для обозначения ветвлений используется структура, состоящая из ромба и переходов с установленными условиями.

4. Ветвление описывает различные пути выполнения в зависимости от значения некоторого булевского выражения. Точка ветвления представляется ромбом, в неё может входить ровно один переход, а выходить - два или более. Для каждого исходящего перехода задается булевское выражение, которое вычисляется только один раз при входе в точку ветвления. Ни для каких двух исходящих переходов эти сторожевые условия не должны одновременно принимать значение «истина», иначе поток управления окажется неоднозначным. Но эти условия должны покрывать все возможные варианты, иначе поток остановится. Для удобства разрешается использовать ключевое слово `else` для пометки того из исходящих переходов, который должен быть выбран в случае, если условия, заданные для всех остальных переходов, не выполнены. Алгоритм называется алгоритмом ветвящейся структуры в случае, если в нем присутствует ветвление (структура, в которой действия выполняются или не выполняются в зависимости от входных данных или выполнения условий).

5. Разветвляющийся алгоритм – алгоритм, в котором вычислительный процесс осуществляется либо по одной, либо по другой ветви, в зависимости от выполнения некоторого условия. Программа разветвляющейся структуры реализует такой алгоритм. В программе разветвляющейся структуры имеется хотя бы один условный оператор. Линейный алгоритм - алгоритм, все этапы которого выполняются однократно и строго последовательно.

6. Условные операторы – это специальные конструкции, благодаря которым в программе происходит ветвление в зависимости от условий. Могут быть представлены в полной и краткой формах.

7. В Python присутствуют такие операторы сравнения: `if`, `else`, `elif` для проверки истинности условия, для обработки данных при не выполнении предыдущих условий или для проверки истинности выражения при ложности предыдущего соответственно.

8. Простое условие - выражение, которое анализирует только одно условие. Оно используется для принятия решений в коде на основе значения переменных или других условий. Примеры: «`if True`», «`if s == 0`», «`if bool_thing`».

9. Составное условие – логическое выражение, содержащее несколько простых условий, объединенных логическими операциями (`not`, `and`, `or`). Пример: `(m == s or m != 0)`.

10. В сложных условиях можно использовать логические операторы `and`, `or`, `not` для обозначений логического «и», «или» и «не» соответственно.

11. Оператор ветвления может быть вложен в другой оператор ветвления, в таком случае вложенный оператор ветвления будет проверять истинность выражения в случае истинности первого.

12. Алгоритм циклической структуры — алгоритм, в котором происходит многократное повторение одного и того же участка программы. Такие повторяемые участки вычислительного процесса называются циклами.

13. В Python существует цикл «`for`» (выполняет указанный набор инструкций заданное количество раз) и цикл «`while`» (выполняет указанный набор инструкций до тех пор, пока условие цикла истинно).

14. Функция `range(“начало”, “конец”, “шаг”)` возвращает неизменяемую последовательность чисел в виде объекта `range`, где начало - с какого числа начинается последовательность (по умолчанию = 0); конец - до какого числа (не включительно) продолжается последовательность чисел; шаг - с каким шагом растут числа (по умолчанию 1).

15. Команда будет выглядеть: `for smth in range(15, 0, -2)`, первым значением мы указываем начало, вторым конец, а третьим – шаг перебора.

16. Цикл может быть вложен в другой цикл (например, для поиска элементов в двумерном списке).

17. Бесконечный цикл в программировании — цикл, написанный таким образом, что условие выхода из него никогда не выполняется (например, неправильно заданно условие выхода из цикла). Чтобы выйти из цикла нужно использовать оператор «break».

18. Оператор «break» предназначен для досрочного прерывания работы цикла.

19. Оператор «continue» завершает текущий круг цикла, переходя к следующему кругу этого же цикла, при этом не выполняя следующий после этого оператора код.

20. В Python есть стандартные потоки: «stdout» — для стандартного вывода (на экран), «stderr» — для стандартного вывода ошибок (вывод ошибок на экран).

21. Необходимо наличие модуля sys (import sys) и использовать команды print(“сообщение”, file=sys.stderr).

22. Функция exit с кодом «1» завершает выполнение программы в Python.

Вывод: в ходе выполнения лабораторной работы были исследованы операторы языка Python версии 3.x: if, while, for, break и continue, позволяющие реализовывать разветвляющиеся алгоритмы и алгоритмы циклической структуры. Также, приобретены навыки программирования разветвляющихся алгоритмов и алгоритмов циклической структуры. Были изучены основные методы составления UML-диаграмм деятельности для наглядной документации работы программы. Были изучены понятия простых и составных операторов, а также логические операторы. Были изучены алгоритмы циклической структуры, ветвящейся структуры.