

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ  
ФЕДЕРАЦИИ**  
**Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**

**«Основы ветвления Git»**

**Отчет по лабораторной работе № 2.2**

**по дисциплине «Основы программной инженерии»**

Выполнил студент группы ПИЖ-б-о-21-1

Образцова М. Д. .«25» октября 2022г.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

Проверил Воронкин Р.А. \_\_\_\_\_

(подпись)

Ставрополь 2022

## МЕТОДИКА И ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Дополнение файла .gitignore необходимыми правилами для работы с IDE PyCharm.

```
(use "git restore" to undo changes in the working directory)
modified:   .gitignore

no changes added to commit (use "git add" and/or "git commit -a")

M@DESKTOP-UVM9NOL MINGW64 ~/desktop/OPI_LR_5 (main)
$ git add .

M@DESKTOP-UVM9NOL MINGW64 ~/desktop/OPI_LR_5 (main)
$ git commit -m ".gitignore file"
[main fb55550] .gitignore file
1 file changed, 140 insertions(+), 2 deletions(-)
```

2. Организация репозитория в соответствии с моделью ветвления git-flow.

```
M@DESKTOP-UVM9NOL MINGW64 ~/desktop/OPI_LR_5 (main)
$ git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main] Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/] Hotfix branches? [hotfix/]
Support branches? [support/] Version tag prefix? []
Hooks and filters directory? [C:/Users/M/desktop/OPI_LR_5/.git/hooks]

M@DESKTOP-UVM9NOL MINGW64 ~/desktop/OPI_LR_5 (develop)
```

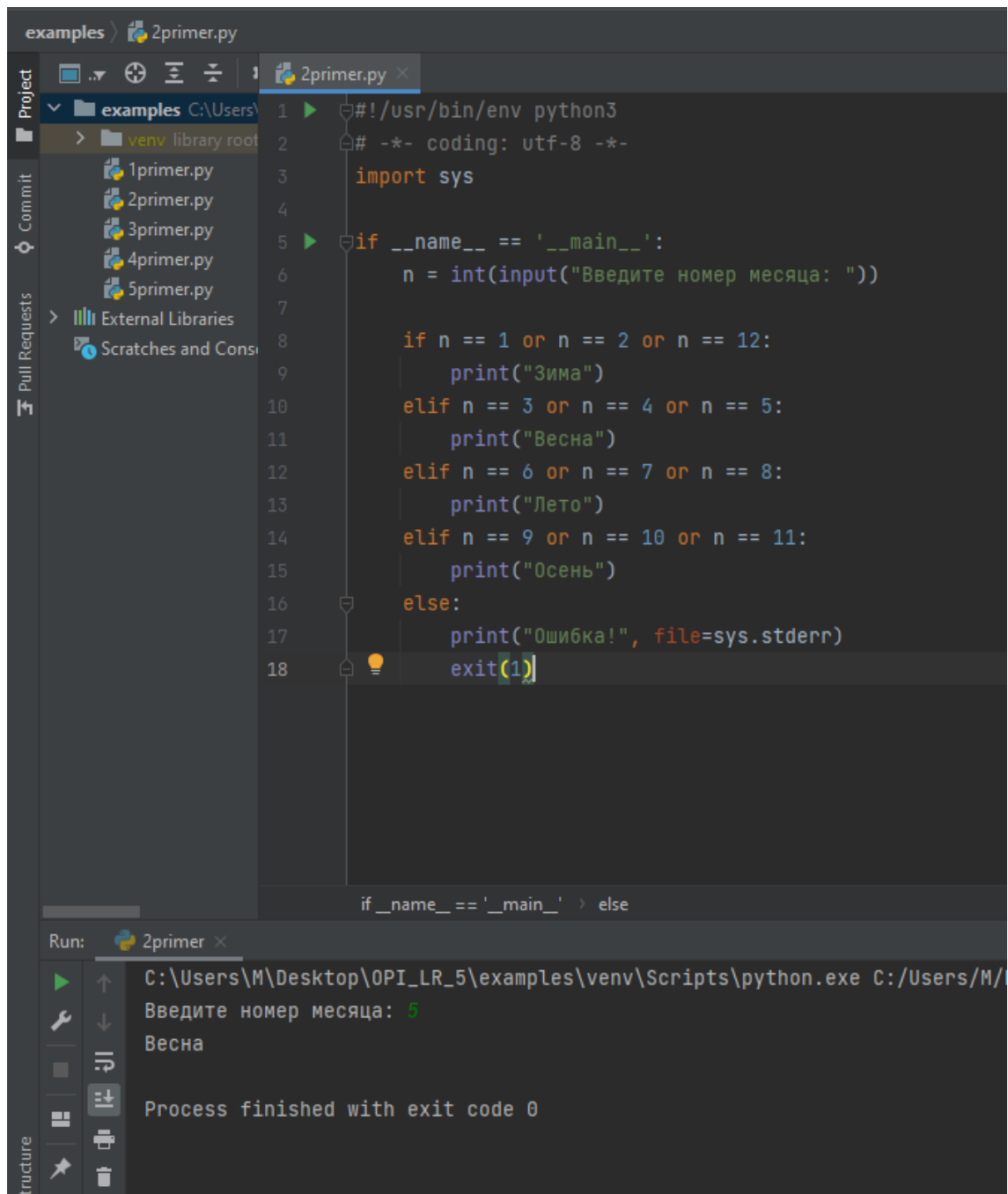
3. Отработка примеров лабораторной работы. Создание для каждого примера отдельного модуля языка Python. Для примеров 4 и 5 постройте UML-диаграмму деятельности.

The image shows a Python IDE with a file named `main.py` open. The code in the file is as follows:

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import math
5
6 if __name__ == '__main__':
7     x = float(input("Value of x? "))
8
9     if x <= 0:
10         y = 2 * x * x + math.cos(x)
11     elif x < 5:
12         y = x + 1
13     else:
14         y = math.sin(x) - x * x
15
16 print(f"y = {y}")
```

Below the code editor, the execution output is shown in a terminal window. The command prompt is `C:\Users\M\Desktop\OPI_LR_5\examples\venv\Scripts\python.exe`. The user input is `Value of x? 4`, and the output is `y = 5.0`. The process finished with exit code 0.

Первый пример



The image shows a code editor with a dark theme. The left sidebar displays a project structure with a folder named 'examples' containing files '1primer.py' through '5primer.py'. The main editor window shows the code for '2primer.py'. The code is a Python script that takes a month number as input and prints the corresponding season. The code is as follows:

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 import sys
4
5 if __name__ == '__main__':
6     n = int(input("Введите номер месяца: "))
7
8     if n == 1 or n == 2 or n == 12:
9         print("Зима")
10    elif n == 3 or n == 4 or n == 5:
11        print("Весна")
12    elif n == 6 or n == 7 or n == 8:
13        print("Лето")
14    elif n == 9 or n == 10 or n == 11:
15        print("Осень")
16    else:
17        print("Ошибка!", file=sys.stderr)
18    exit(1)
```

Below the code editor, the 'Run' panel shows the execution command and output:

```
Run: 2primer x
C:\Users\M\Desktop\OPI_LR_5\examples\venv\Scripts\python.exe C:/Users/M/...
Введите номер месяца: 5
Весна
Process finished with exit code 0
```

Второй пример

```
1  ▶  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import math
5
6  ▶  if __name__ == '__main__':
7      n = int(input("Value of n? "))
8      x = float(input("Value of x? "))
9
10     S = 0.0
11     for k in range(1, n + 1):
12         a = math.log(k * x) / (k * k)
13         S += a
14
15     print(f"S = {S}")
```

if \_\_name\_\_ == '\_\_main\_\_'

1primer x

C:\Users\M\Desktop\OPI\_LR\_5\examples\venv\Scripts\python.e

Value of n? 5

Value of x? 3

S = 2.054316893779431

Process finished with exit code 0

Третий пример

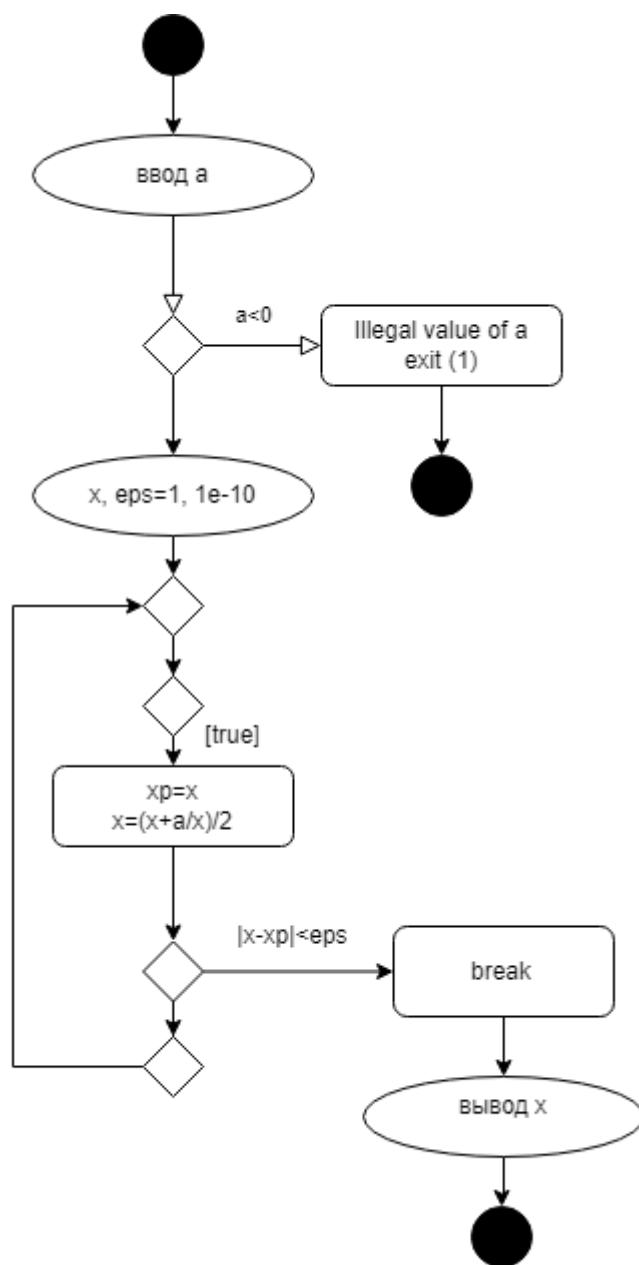
The image shows a Python IDE with a file named `4primer.py` open. The code is a script that calculates the square root of a user-input number `a` using the Newton-Raphson method. It includes error handling for negative values and a loop that continues until the result is accurate within a specified epsilon.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import math
5  import sys
6
7  if __name__ == '__main__':
8      a = float(input("Value of a? "))
9      if a < 0:
10         print("Illegal value of a", file=sys.stderr)
11         exit(1)
12
13         x, eps = 1, 1e-10
14         while True:
15             xp = x
16             x = (x + a / x) / 2
17             if math.fabs(x - xp) < eps:
18                 break
19
20         print(f"x = {x}\nX = {math.sqrt(a)}")
```

Below the code editor, the output of the script is shown in a terminal window. The user has entered the value 13, and the program has calculated the square root to a high precision, matching the built-in `math.sqrt` function.

```
1primer x
C:\Users\M\Desktop\OPI_LR_5\examples\venv\Scripts\python.exe C:/Us
Value of a? 13
x = 3.6055512754639896
X = 3.605551275463989
Process finished with exit code 0
```

Четвертый пример



```
1  ▶  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import math
5  import sys
6
7  # Постоянная Эйлера.
8  EULER = 0.5772156649015328606
9
10 # Точность вычислений.
11 EPS = 1e-10
12
13 ▶  if __name__ == '__main__':
14      x = float(input("Value of x? "))
15      if x == 0:
16          print("Illegal value of x", file=sys.stderr)
17          exit(1)
18
19      a = x
20      S, k = a, 1
21
22      # Найти сумму членов ряда.
23      while math.fabs(a) > EPS:
24          a *= x * k / (k + 1) ** 2
25          S += a
26          k += 1
27
28      # Вывести значение функции.
29      print(f"Ei({x}) = {EULER + math.log(math.fabs(x)) + S}")
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

1primer ×

C:\Users\M\Desktop\OPI\_LR\_5\examples\venv\Scripts\python.exe C:/Use

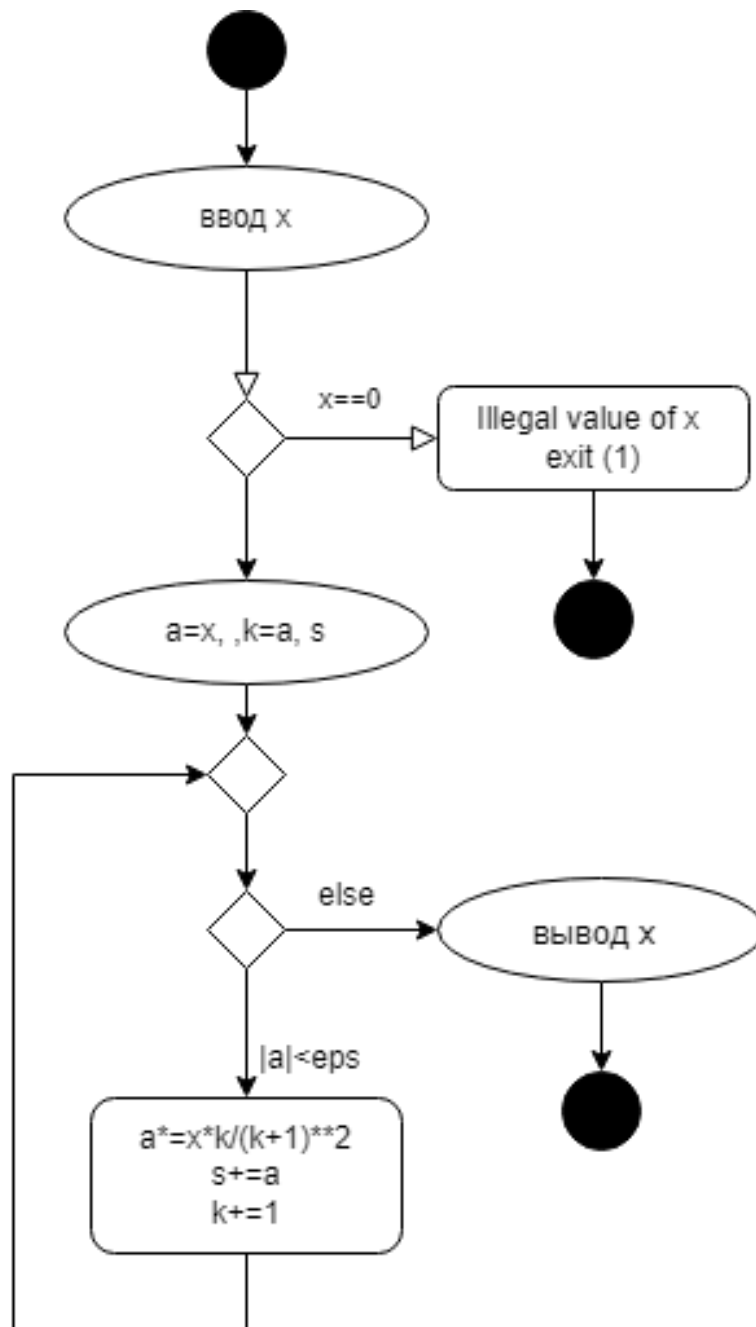
Value of x? 21

Ei(21.0) = 66127186.355484925

Process finished with exit code 0

Пятый пример



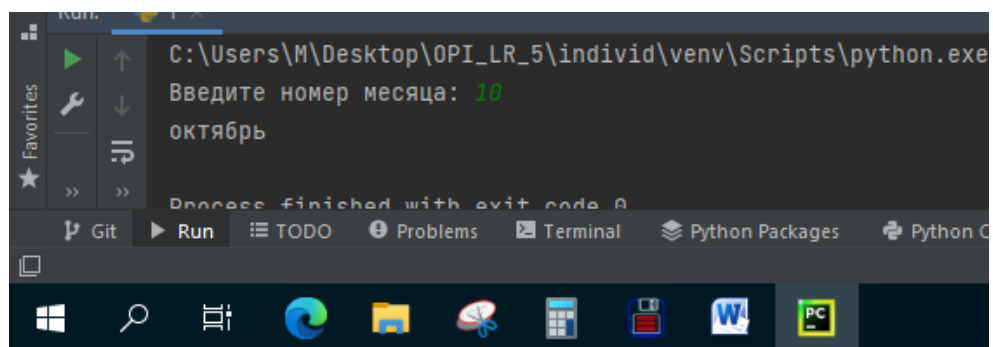
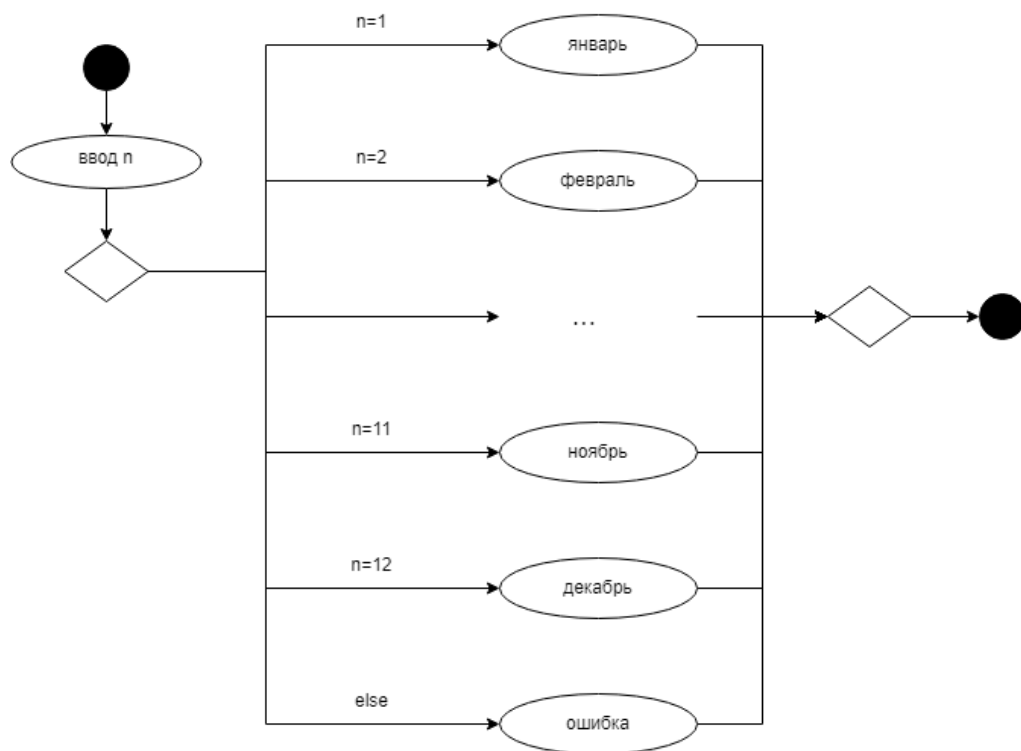


4. Выполнение индивидуальных заданий.

### Индивидуальные задания

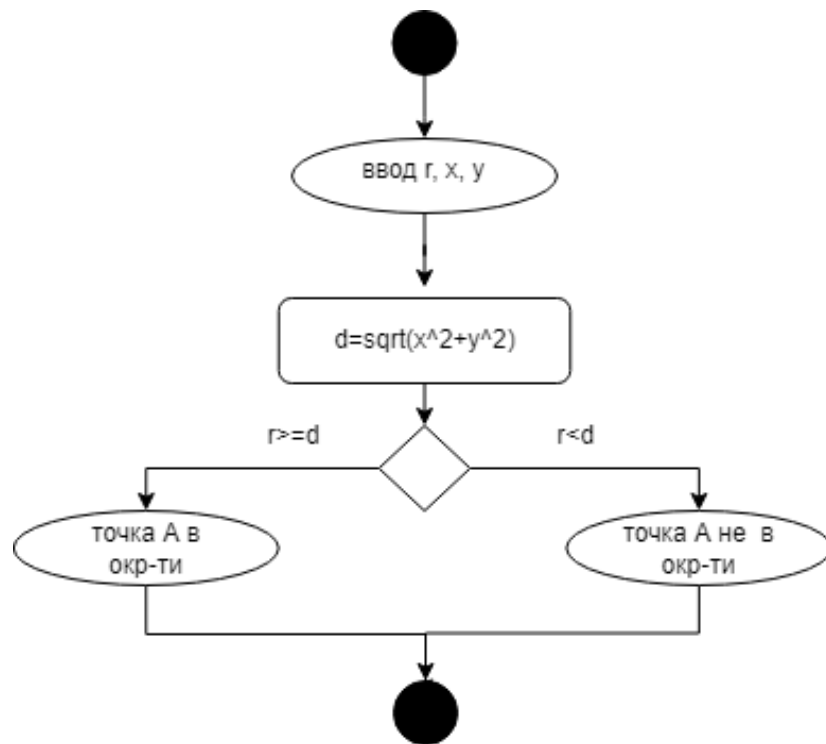
Задание 1 (7 вариант)

С клавиатуры вводится цифра (от 1 до 12). Вывести на экран название месяца, соответствующего цифре.



## Задание 2 (20 вариант)

Попадёт ли точка A в окружность заданного радиуса с центром в начале координат?

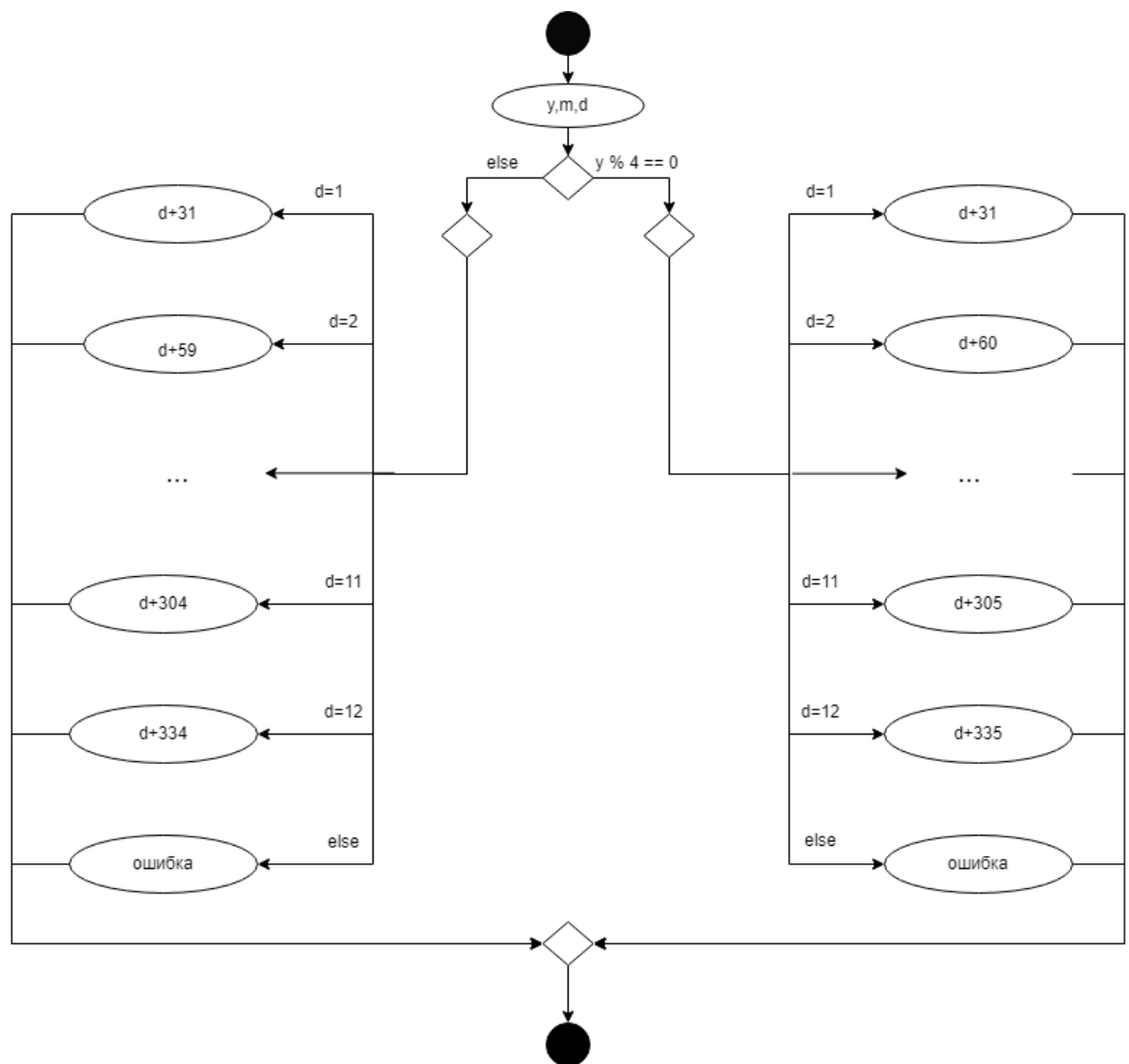


```
C:\Users\M\Desktop\OPI_LR_5\individ\venv\Scripts\python.exe C:/Users/M/Desktop/
введите радиус окружности: 5
введите первую координату точки A:
x=1
введите вторую координату точки A:
y=4
точка A входит в окружность
d=4.123105625617661

Process finished with exit code 0
```

### Задание 3 (20 вариант)

Заданы три натуральных числа  $d$ ,  $m$ ,  $y$ , которые обозначают число, месяц и год. Найти порядковый номер даты, начиная отсчет с начала года.



```

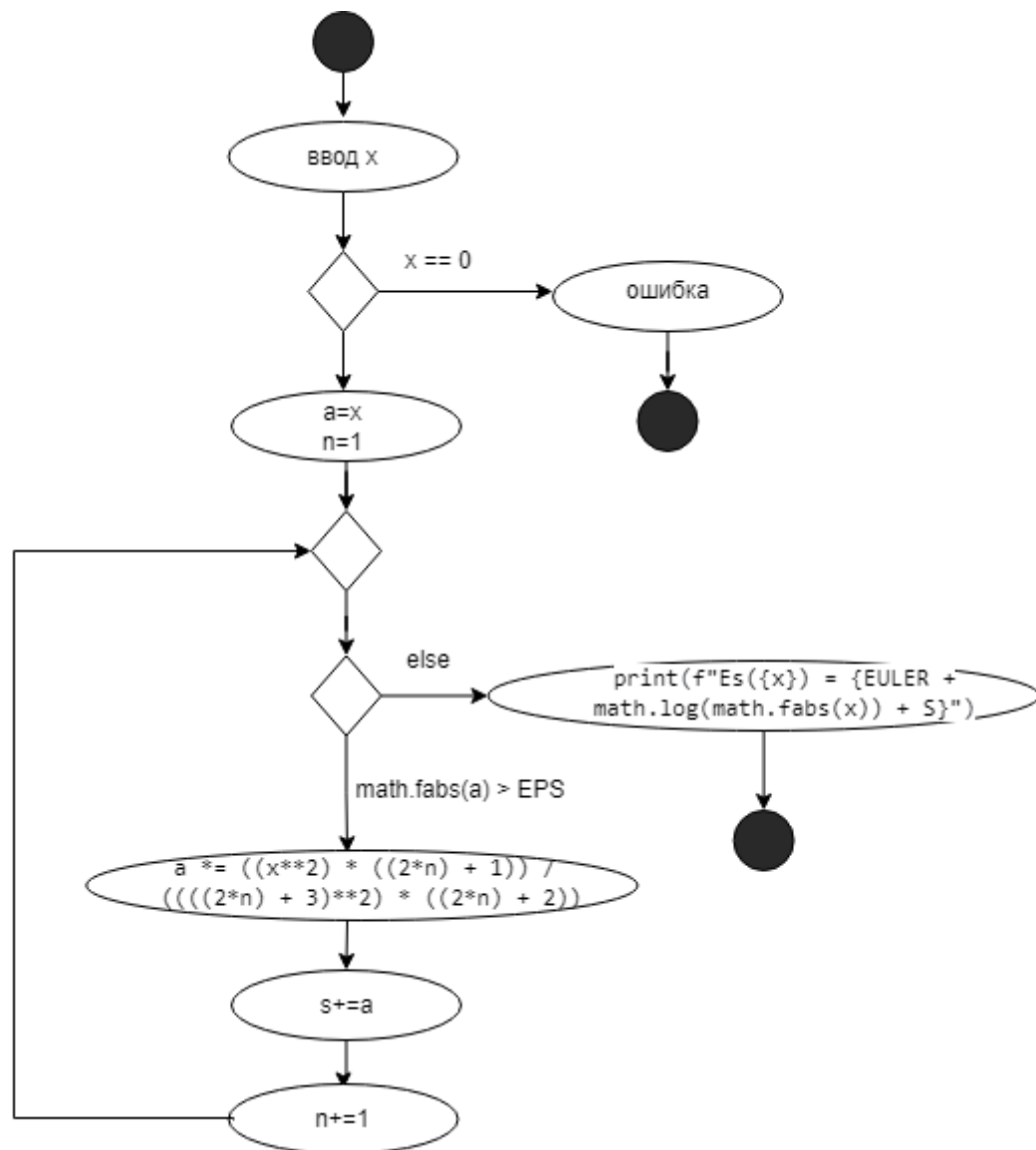
C:\Users\M\Desktop\OPI_LR_5\individ\venv\Scripts\python.exe C:/Users/M/D
введите число= 15
введите месяц= 2
введите год= 2022
порядковый номр даты=46
  
```

### Задание повышенной сложности

Составить UML-диаграмму деятельности, программу и произвести вычисления вычисление значения специальной функции по ее разложению в ряд с точностью , аргумент функции вводится с клавиатуры.

### 3. Интегральный гиперболический синус:

$$\text{Shi}(x) = \int_0^x \frac{\text{sh } t}{t} dt = \sum_{n=0}^{\infty} \frac{x^{2n+1}}{(2n+1)(2n+1)!}.$$



```
C:\Users\M\Desktop\OPI_LR_5\individ>python 4.py
Value of x? 5
Es(5.0) = 13.053766091837101
```

Вопросы для защиты работы

1. Для чего нужны диаграммы деятельности UML?

Диаграммы деятельности представляют собой графическое представление рабочих процессов поэтапных действий и действий с поддержкой выбора, итерации и параллелизма. Они описывают поток управления целевой системой, такой как исследование сложных бизнес-правил и операций, а также описание прецедентов и бизнес-процессов. В UML диаграммы деятельности предназначены для моделирования как вычислительных, так и организационных процессов.

2. Что такое состояние действия и состояние деятельности?

Состояние действия — это состояние, внутренняя активность которого является действием.

Состояние деятельности — это состояние, внутренняя активность которого является деятельностью.

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

Переходы, ветвление, алгоритм разветвляющейся структуры, алгоритм циклической структуры.

4. Какой алгоритм является алгоритмом разветвляющейся структуры?

Алгоритм разветвляющейся структуры - это алгоритм, в котором вычислительный процесс осуществляется либо по одной, либо по другой ветви, в зависимости от выполнения некоторого условия.

5. Чем отличается разветвляющийся алгоритм от линейного?

Линейный алгоритм - алгоритм, все этапы которого выполняются однократно и строго последовательно.

Разветвляющийся алгоритм - алгоритм, содержащий хотя бы одно условие, в результате проверки которого ЭВМ обеспечивает переход на один из нескольких возможных шагов.

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

Оператор, конструкция языка программирования, обеспечивающая выполнение определённой команды (набора команд) только при условии

истинности некоторого логического выражения, либо выполнение одной из нескольких команд.

Условный оператор имеет полную и краткую формы.

7. Какие операторы сравнения используются в Python?

If, elif, else

8. Что называется простым условием? Приведите примеры.

Простым условием называется выражение, составленное из двух арифметических выражений или двух текстовых величин.

Пример: `x == y`

9. Что такое составное условие? Приведите примеры.

Составное условие – логическое выражение, содержащее несколько простых условий объединенных логическими операциями. Это операции `not`, `and`, `or`.

Пример: `(a == b or c == d)`

10. Какие логические операторы допускаются при составлении сложных условий?

`not`, `and`, `or`.

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

Может.

12. Какой алгоритм является алгоритмом циклической структуры?

Циклический алгоритм — алгоритм, предусматривающий многократное повторение одного и того же действия (одних и тех же операций) над новыми исходными данными.

13. Типы циклов в языке Python.

В Python есть 2 типа циклов: - цикл `while`, - цикл `for`.

14. Назовите назначение и способы применения функции `range`.

Функция `range` генерирует серию целых чисел, от значения `start` до `stop`, указанного пользователем. Мы можем использовать его для цикла `for` и обходить весь диапазон как список.

Функция `range` возвращает неизменяемую последовательность чисел в виде объекта `range`.

Параметры функции:

`start` - с какого числа начинается последовательность. По умолчанию - 0

`stop` - до какого числа продолжается последовательность чисел.

Указанное число не включается в диапазон

`step` - с каким шагом растут числа. По умолчанию 1

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

`range(15, 0, -2)`

16. Могут ли быть циклы вложенными?

Могут.

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

Бесконечный цикл в программировании — цикл, написанный таким образом, что условие выхода из него никогда не выполняется.

18. Для чего нужен оператор `break`?

Оператор `break` предназначен для досрочного прерывания работы цикла.

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

Оператор `continue` используется только в циклах. В операторах `for`, `while` оператор `continue` запускает цикл заново, при этом код, расположенный после данного оператора, не выполняется.

20. Для чего нужны стандартные потоки `stdout` и `stderr`?

Ввод и вывод распределяется между тремя стандартными потоками:  
`stdin` – стандартный ввод (клавиатура), `stdout` – стандартный вывод (экран),  
`stderr` – стандартная ошибка (вывод ошибок на экран)

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

Указать в `print(..., file=sys.stderr)`.

22. Каково назначение функции `exit`?



Функция `exit()` модуля `sys` - выход из Python.