

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

Институт цифрового развития
Кафедра инфокоммуникаций

«Условные операторы и циклы в языке Python»

ОТЧЕТ
по лабораторной работе №5
дисциплины
«Основы программной инженерии»

Выполнил:
Сотников Андрей Александрович
2 курс, группа ПИЖ-б-о-21-1,
09.03.04 «Программная
инженерия», направленность
(профиль) «Разработка и
сопровождение программного
обеспечения», очная форма
обучения

(подпись)

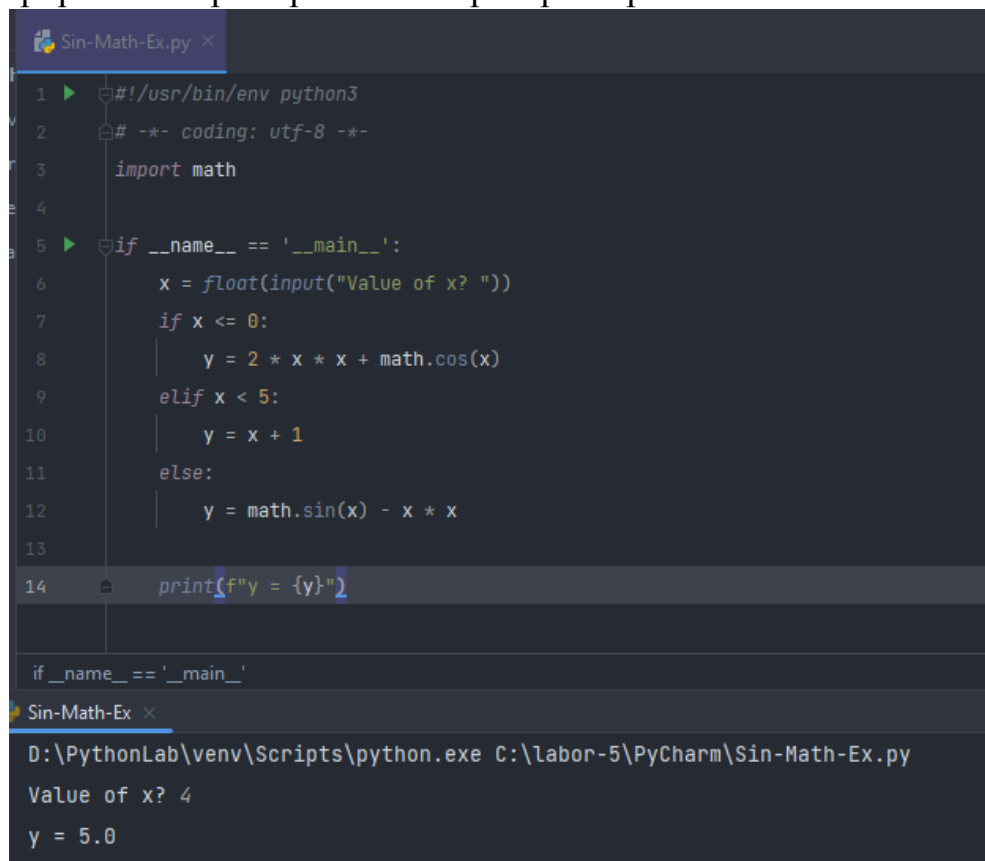
Проверил:

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2022 г.

Проработка примеров из лабораторной работы:



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

if __name__ == '__main__'

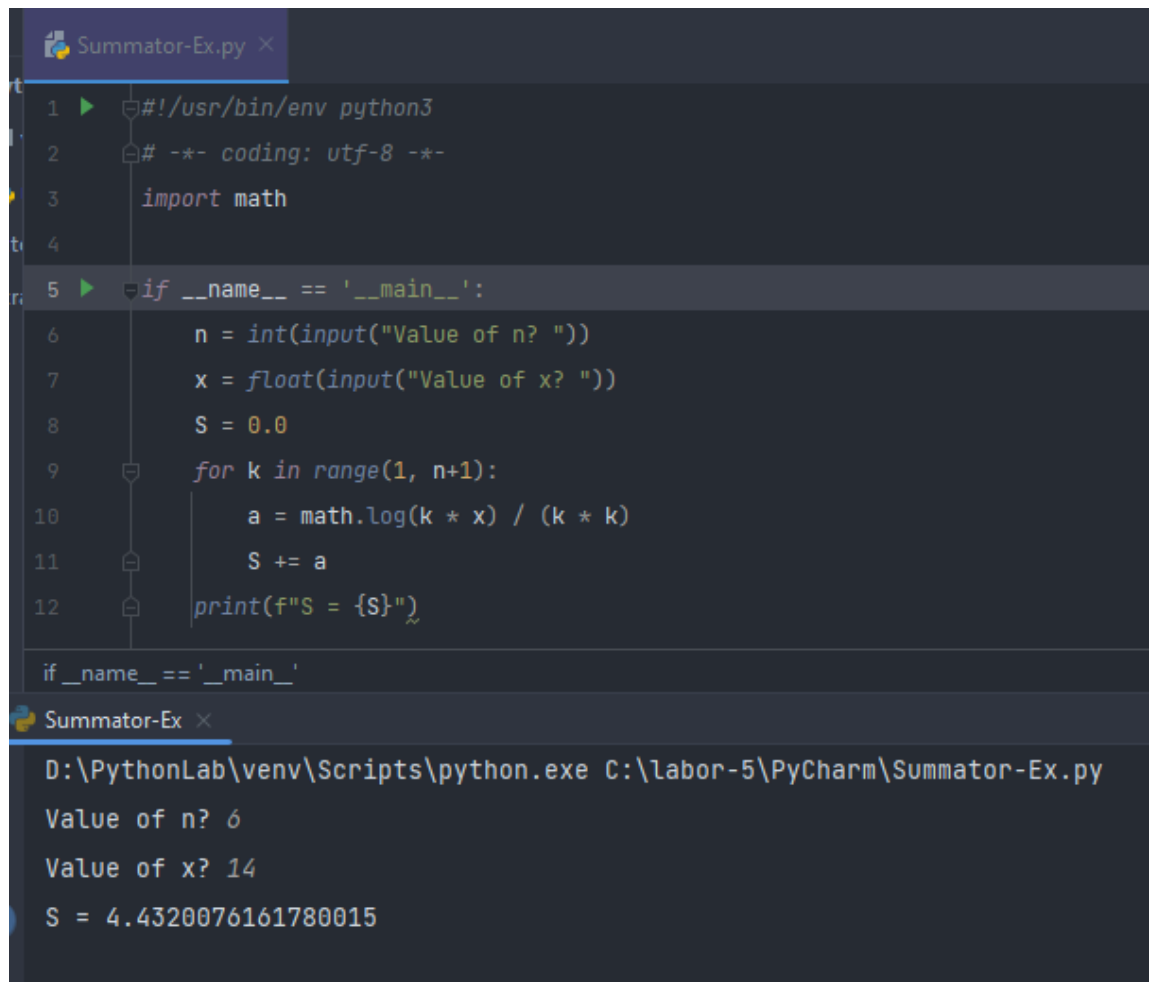
Sin-Math-Ex ×

D:\PythonLab\venv\Scripts\python.exe C:\labor-5\PyCharm\Sin-Math-Ex.py
Value of x? 4
y = 5.0

Рисунок 1 – Пример №1

```
Seasons-Ex.py ×
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      if n == 1 or n == 2 or n == 12:
8          print("Зима")
9      elif n == 3 or n == 4 or n == 5:
10         print("Весна")
11     elif n == 6 or n == 7 or n == 8:
12         print("Лето")
13     elif n == 9 or n == 10 or n == 11:
14         print("Осень")
15     else:
16         print("Ошибка!", file=sys.stderr)
17         exit(1)
18
19 if __name__ == '__main__' > else
Seasons-Ex ×
D:\PythonLab\venv\Scripts\python.exe C:\labor-5\PyCharm\Seasons-Ex.py
Введите номер месяца: 3
Весна
```

Рисунок 2 – Пример №2



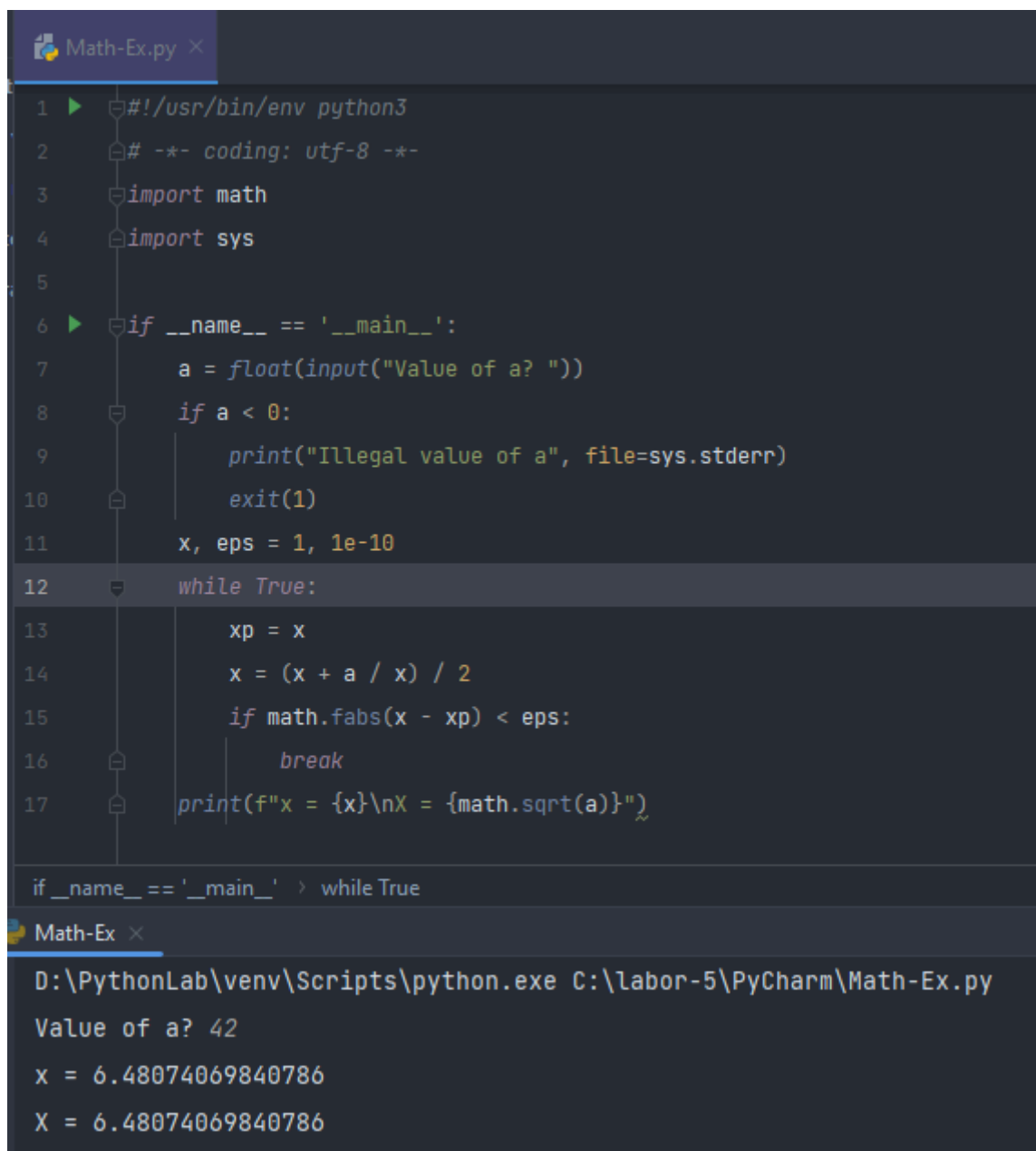
The image shows a screenshot of a Python IDE with two panels. The top panel displays the source code of a file named `Summator-Ex.py`. The code is as follows:

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

The bottom panel shows the execution of the script. The command prompt displays the path to the Python interpreter and the script file, followed by the user input and the resulting output:

```
D:\PythonLab\venv\Scripts\python.exe C:\labor-5\PyCharm\Summator-Ex.py
Value of n? 6
Value of x? 14
S = 4.4320076161780015
```

Рисунок 3 – Пример №3



The image shows a PyCharm IDE window with a Python script named `Math-Ex.py` and its execution output.

Script Content:

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

Execution Output:

```
D:\PythonLab\venv\Scripts\python.exe C:\labor-5\PyCharm\Math-Ex.py
Value of a? 42
x = 6.48074069840786
X = 6.48074069840786
```

Рисунок 4.1 – Пример №4

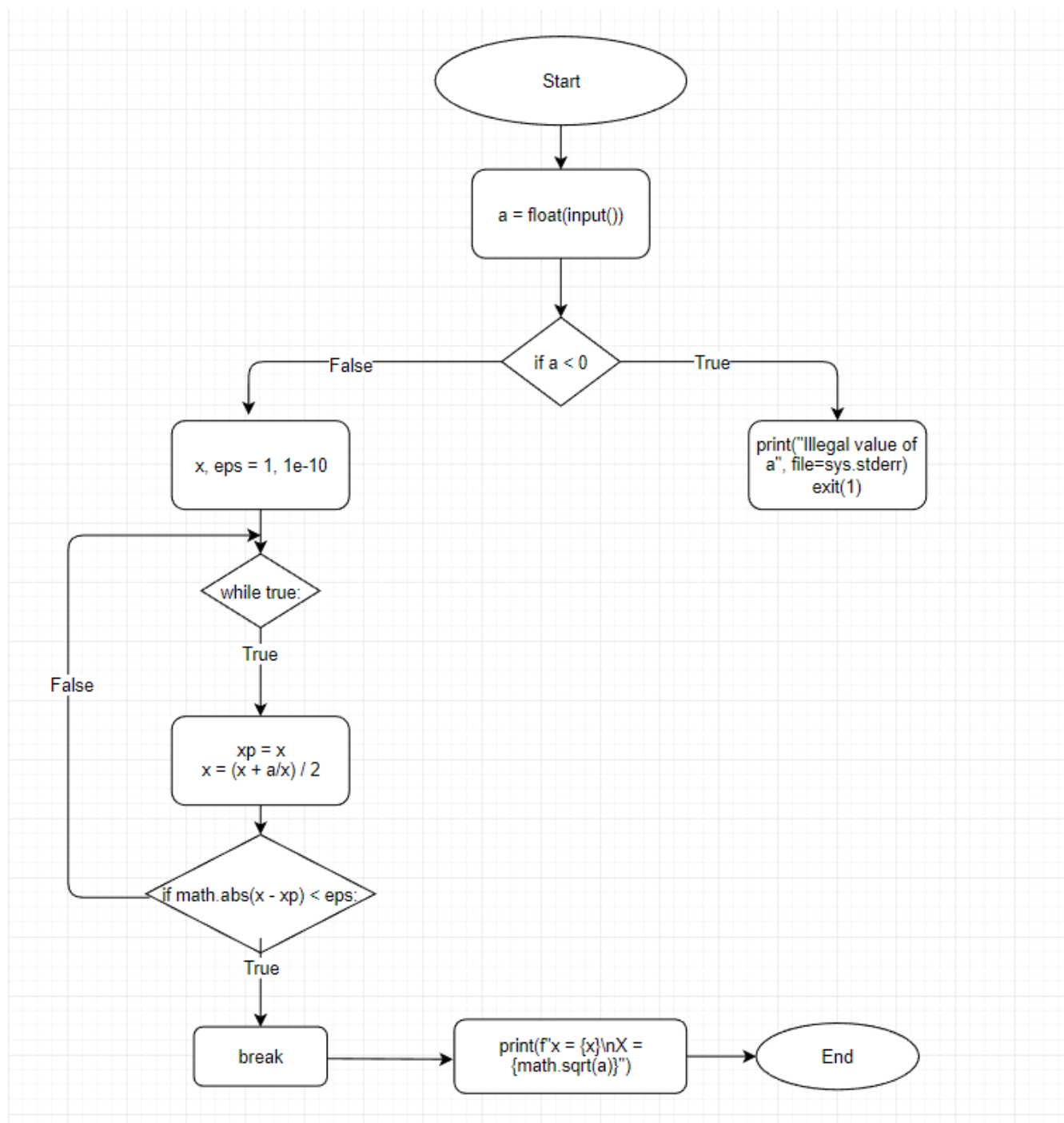


Рисунок 4.2 – UML диаграмма примера №4

```
Integral-Ex.py ×
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3      import math
4      import sys
5      # Постоянная Эйлера.
6      EULER = 0.5772156649015328606
7      # Точность вычислений.
8      EPS = 1e-10
9
10  ▶  if __name__ == '__main__':
11      x = float(input("Value of x? "))
12      if x == 0:
13          print("Illegal value of x", file=sys.stderr)
14          exit(1)
15      a = x
16      S, k = a, 1
17      # Найти сумму членов ряда.
18      while math.fabs(a) > EPS:
19          a *= x * k / (k + 1) ** 2
20          S += a
21          k += 1
22      # Вывести значение функции.
23      print(f"Ei({x}) = {EULER + math.log(math.fabs(x)) + S}")

if __name__ == '__main__':
Integral-Ex ×
D:\PythonLab\venv\Scripts\python.exe C:\labor-5\PyCharm\Integral-Ex.py
Value of x? 48
Ei(48.0) = 1.493630213112992e+19
```

Рисунок 5.1 – Пример №5

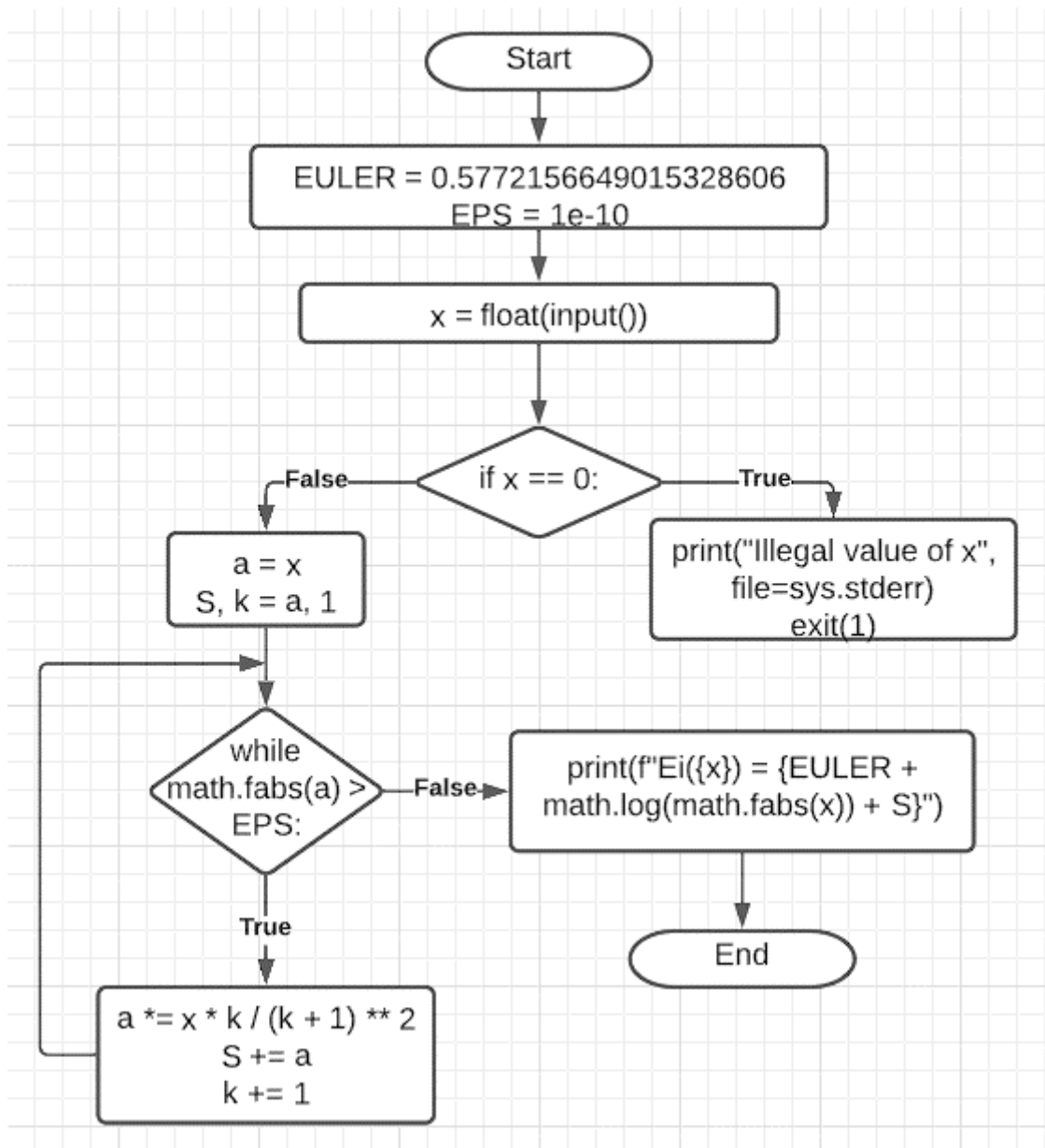


Рисунок 5.2 – UML диаграмма примера №5

Задание №1: Студенты убирают урожай помидоров. При сборе до 50 кг в день работа оплачивается из расчёта 30 коп. за 1 кг; при сборе от 50 до 75 кг в день - 50 коп. за 1 кг; при сборе от 75 до 90 кг в день - 65 коп. за 1 кг; при сборе свыше 90 кг в день - 70 коп. за 1 кг плюс 20 руб. премия. Студент собрал X кг за день. Определить его заработок.

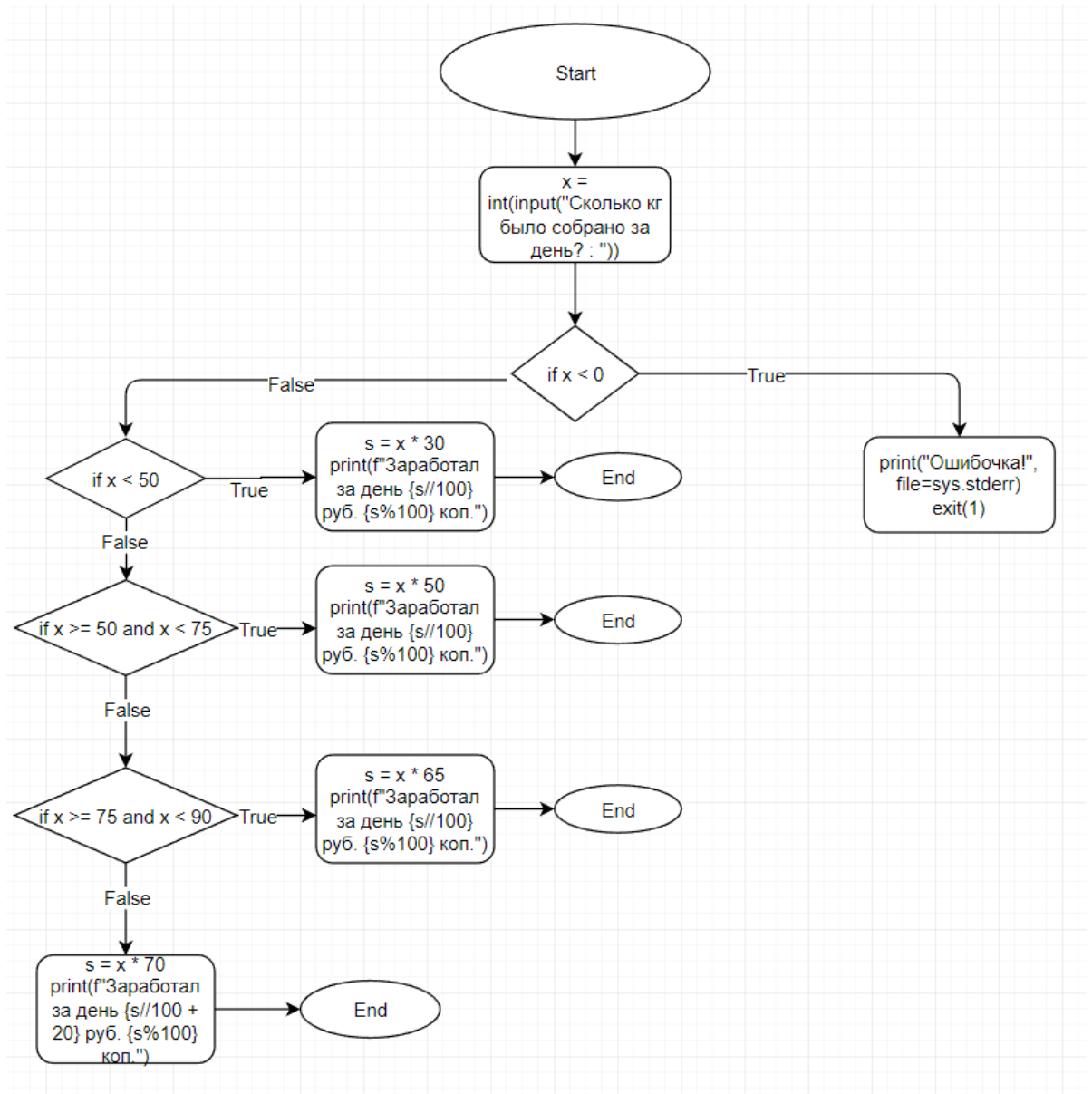


Рисунок 1 – UML диаграмма первого задания

```
Ind-Urojai.py ×
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3      import sys
4
5  ▶  if __name__ == '__main__':
6      x = int(input("Сколько кг было собрано за день? : "))
7
8      if x < 0:
9          print("Ошибочка!", file=sys.stderr)
10         exit(1)
11
12     if x < 50:
13         s = x * 30
14         print(f"Заработал за день: {s//100} руб. {s%100} коп.")
15
16     elif x >= 50 and x < 75:
17         s = x * 50
18         print(f"Заработал за день: {s // 100} руб. {s % 100} коп.")
19
20     elif x >= 75 and x < 90:
21         s = x * 65
22         print(f"Заработал за день: {s // 100} руб. {s % 100} коп.")
23
24     else:
25         s = x * 70
26         print(f"Заработал за день: {s // 100 + 20} руб. {s % 100} коп.")
27
28 if __name__ == '__main__':
```

Ind-Urojai ×

D:\PythonLab\venv\Scripts\python.exe C:\labor-5\PyCharm\Ind-Urojai.py
Сколько кг было собрано за день? : 91
Заработал за день: 83 руб. 70 коп.

Рисунок 2 – Код и результат работы программы №1

Задание №2: Составить программу, выясняющую делится ли натуральное число x нацело на натуральное число y .

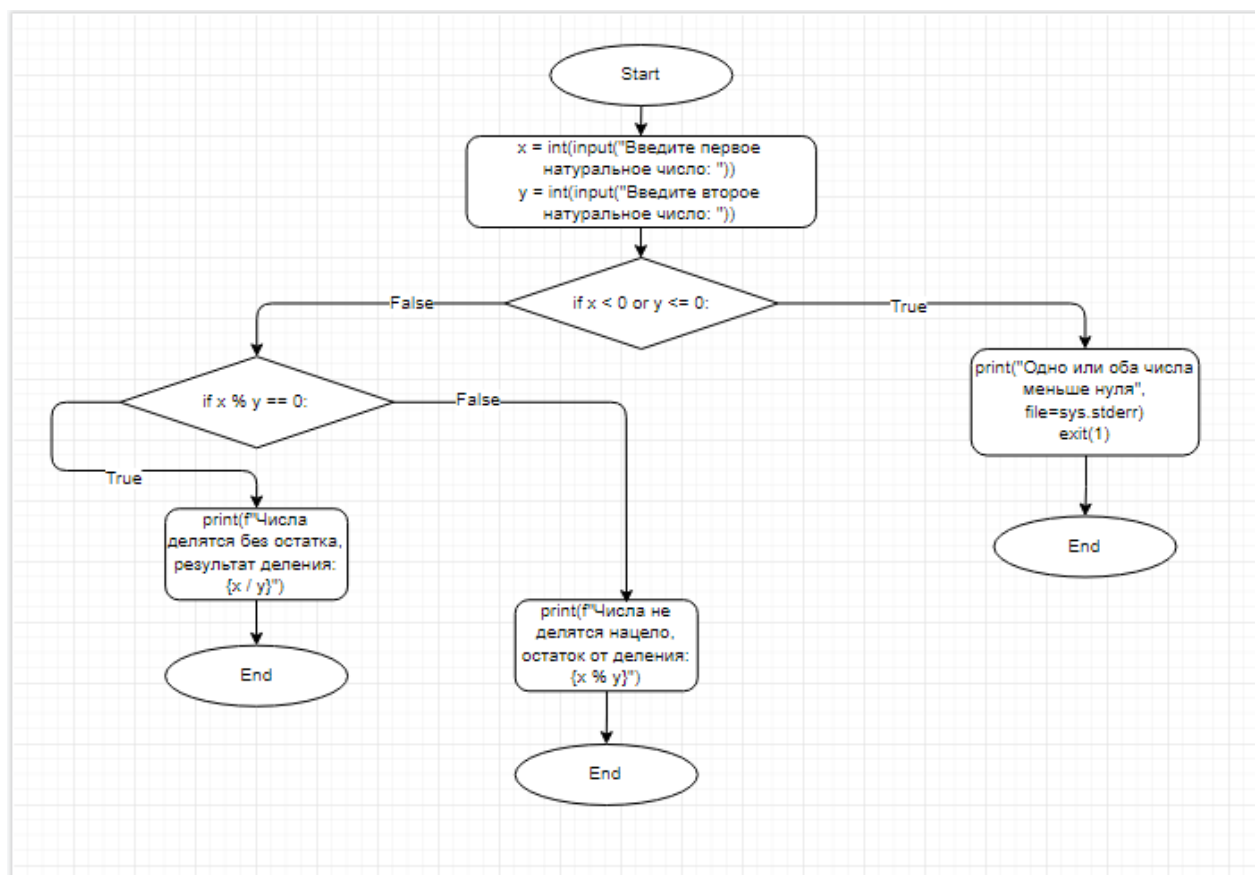
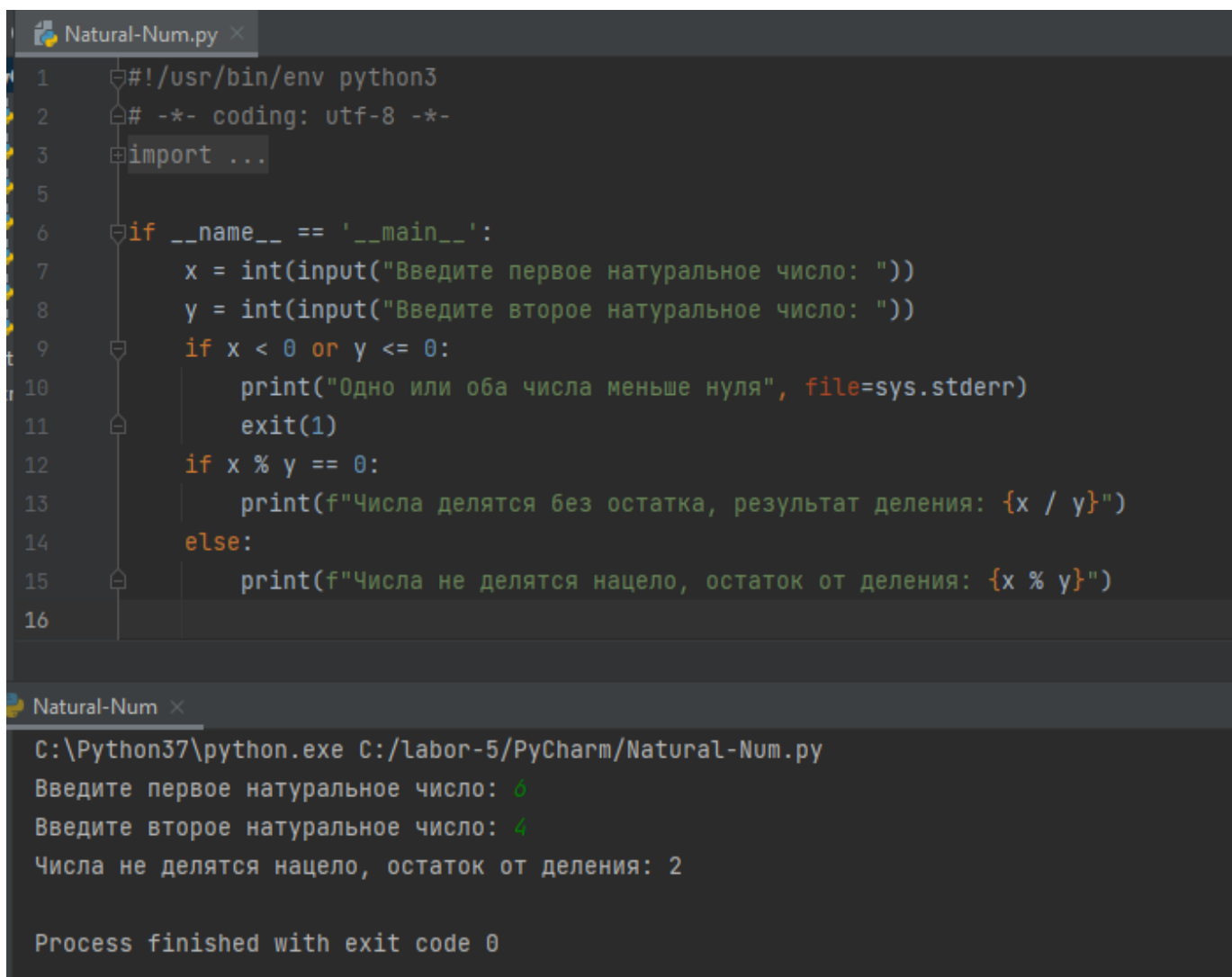


Рисунок 3 – UML диаграмма второго задания



The image shows a PyCharm IDE with two panels. The top panel displays a Python script named `Natural-Num.py`. The script prompts the user for two natural numbers, checks if they are non-negative, and then checks if they are divisible. The bottom panel shows the execution output, where the user entered 6 and 4, resulting in a remainder of 2.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  import ...
4
5
6  if __name__ == '__main__':
7      x = int(input("Введите первое натуральное число: "))
8      y = int(input("Введите второе натуральное число: "))
9      if x < 0 or y <= 0:
10         print("Одно или оба числа меньше нуля", file=sys.stderr)
11         exit(1)
12     if x % y == 0:
13         print(f"Числа делятся без остатка, результат деления: {x / y}")
14     else:
15         print(f"Числа не делятся нацело, остаток от деления: {x % y}")
16
```

C:\Python37\python.exe C:/labor-5/PyCharm/Natural-Num.py
Введите первое натуральное число: 6
Введите второе натуральное число: 4
Числа не делятся нацело, остаток от деления: 2

Process finished with exit code 0

Рисунок 4 – Код и результат работы программы №2

Задание №3: Дано натуральное число . Получить все его натуральные делители.

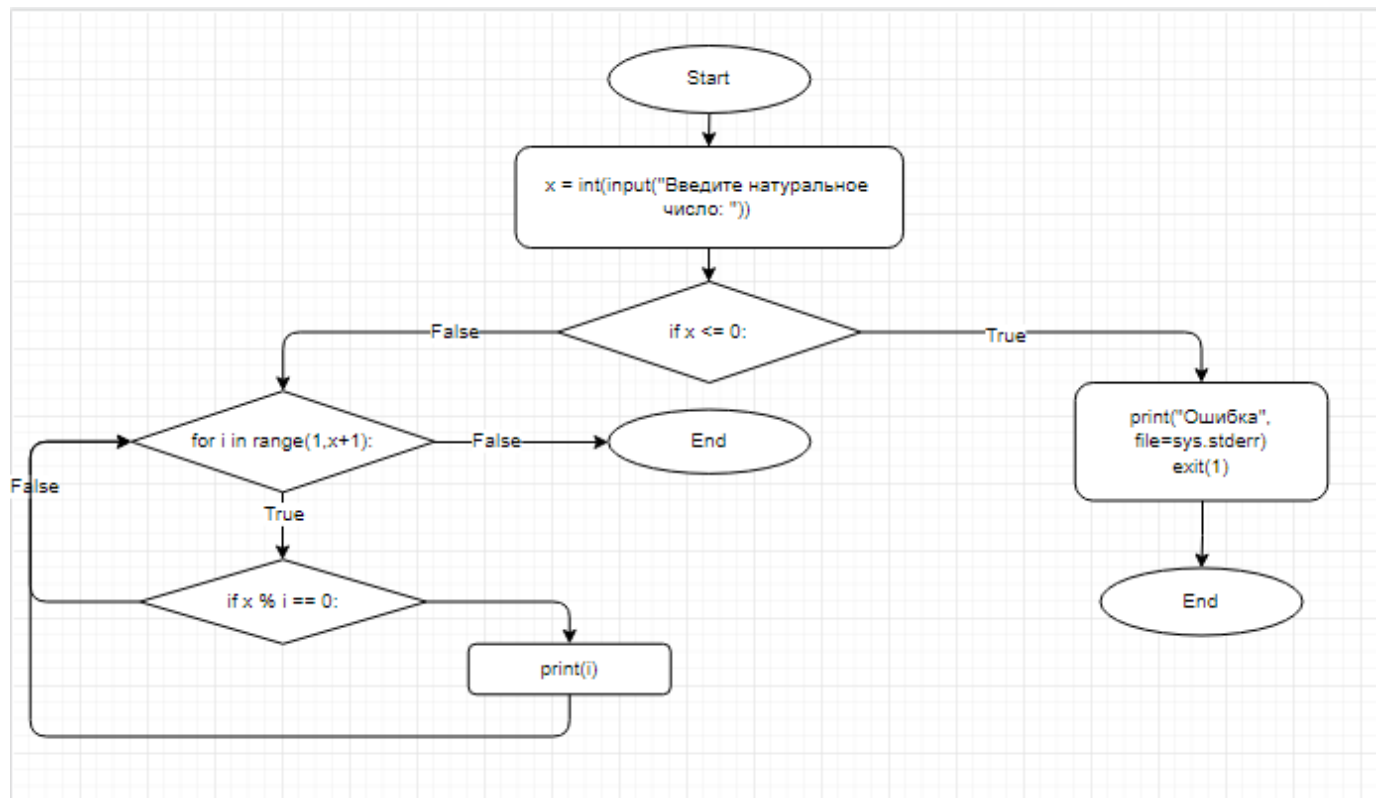


Рисунок 5 – UML диаграмма третьего задания

```

1  ▶  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  import math
4  import sys
5
6  ▶  if __name__ == '__main__':
7      x = int(input("Введите натуральное число: "))
8      if x <= 0:
9          print("Ошибка", file=sys.stderr)
10         exit(1)
11     for i in range(1, x+1):
12         if x % i == 0:
13             print(i)

```

Рисунок 6 – Код программы №3

```

C:\Python37\python.exe C:/labor-5/PyCharm/Natural-del.py
Введите натуральное число: 42
1
2
3
6
7
14
21
42

```

Рисунок 7.1 – Вывод программы №3

```
C:\Python37\python.exe C:/labor-  
Введите натуральное число: 7  
1  
7
```

Рисунок 7.2 – Вывод программы №3

```
C:\Python37\python.exe C:/labor-5  
Введите натуральное число: 165  
1  
3  
5  
11  
15  
33  
55  
165
```

Рисунок 7.3 – Вывод программы №3

Контрольные вопросы

1. Для чего нужны диаграммы деятельности UML?

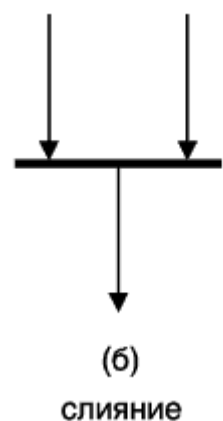
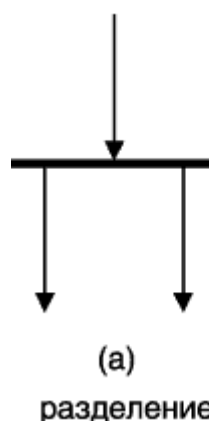
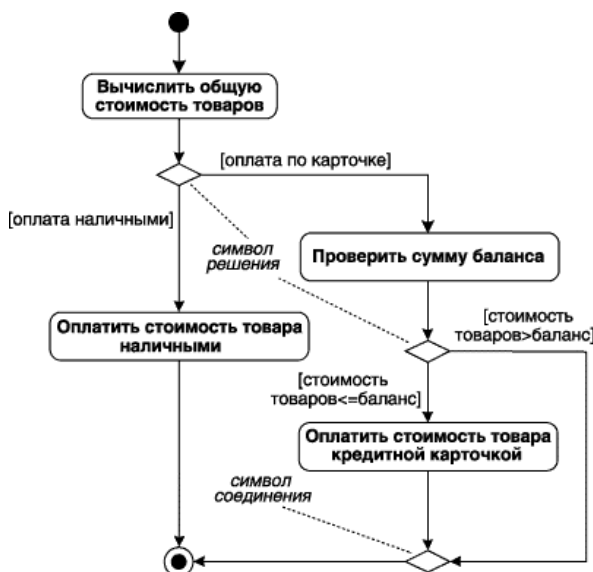
Диаграмма деятельности — это, по существу, блок-схема, которая показывает, как поток управления переходит от одной деятельности к другой, однако, по сравнению с последней, у ней есть явные преимущества: поддержка многопоточности и объектно-ориентированного проектирования.

2. Что такое состояние действия и состояние деятельности?

Состояния действия не могут быть подвергнуты декомпозиции. Кроме того, они атомарны. Это значит, что 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

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.

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



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?

Наличием условных операторов

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

Команда, которая выполняется только при каком-либо условии

7. Какие операторы сравнения используются в Python?

оператор `<` , «меньше»;
оператор `<=` , «меньше или равно»;
оператор `==` , «равно»;
оператор `!=` , «не равно»;
оператор `>` , «больше»;
оператор `>=` , «больше или равно».

8. Что называется простым условием? Приведите примеры.

Простым условием (отношением) называется выражение, составленное из двух арифметических выражений или двух текстовых величин (иначе их еще называют операндами), связанных одним из знаков приведенных в ответе на 7 вопрос

9. Что такое составное условие? Приведите примеры.

Составное условие – логическое выражение, содержащее несколько простых условий, объединенных логическими операциями.

10. Какие логические операторы допускаются при составлении сложных условий?

Логическое И, логическое ИЛИ, логическое отрицание

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

Да

12. Какой алгоритм является алгоритмом циклической структуры?

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

13. Типы циклов в языке Python.

В Python есть два вида циклов: for и while

14. Назовите назначение и способы применения функции range

Функция range возвращает неизменяемую последовательность чисел в виде объекта range. Синтаксис функции:

`range(stop)`

`range(start, stop[, step])`

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(0, 15, 2)`

16. Могут ли быть циклы вложенными?

Вложенный цикл - цикл который выполняется внутри другого цикла. Обычно вложенные циклы используются для работы с двумя измерениями.

Да могут

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

Пример бесконечного цикла:

```
a = 0  
while a == 0:  
    print("A")
```

Выйти из такого цикла можно при помощи оператора `break`

18. Для чего нужен оператор `break` ?

Оператор `break` предназначен для досрочного прерывания работы цикла `while`.

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

Оператор `continue` запускает цикл заново, при этом код, расположенный после данного оператора, не выполняется.

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

В операционной системе по умолчанию присутствуют стандартных потока вывода на консоль: буферизованный поток `stdout` для вывода данных и информационных сообщений, а также небуферизованный поток `stderr` для вывода сообщений об ошибках.

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

По умолчанию функция `print` использует поток `stdout`. Для того, чтобы использовать поток `stderr` необходимо передать его в параметре `file` функции `print`.

22. Каково назначение функции `exit` ?

Если в процессе выполнения программы произошли ошибки, программа должна передать операционной системе код возврата отличный от нуля. В Python завершить программу и передать операционной системе заданный код возврата можно посредством функции `exit`.