

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

(подпись)

Руководитель практики:
Богданов С.С., ассистент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: Условные операторы и циклы в языке Python.

Цель работы: 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.

Ход выполнения работы:

1. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и .gitignore файл для языка программирования Python:

Рисунок 1 – Создание репозитория с заданными настройками

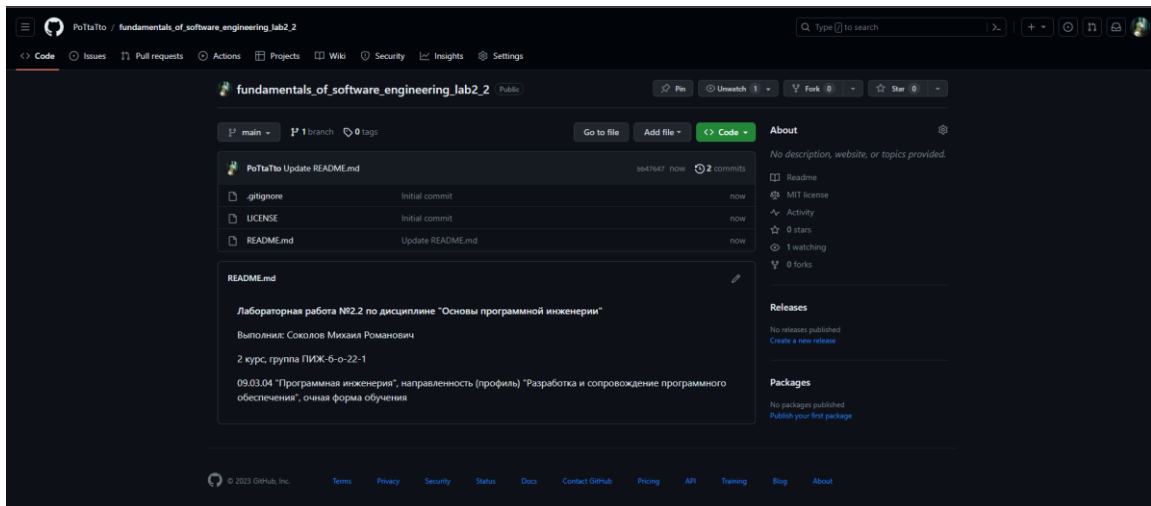


Рисунок 2 – Созданный репозиторий

```
PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.2> git clone https://github.com/PoTtaTto/fundamentals_of_software_engineering_lab2.2
Cloning into 'fundamentals_of_software_engineering_lab2.2'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.
Resolving deltas: 100% (1/1), done.
PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.2> |
```

Рисунок 3 – Клонирование репозитория

```
PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.2> git checkout -b develop
Switched to a new branch 'develop'
PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.2>
```

Рисунок 4 – Создание ветки develop, где будут происходить изменения проекта до его полного релиза

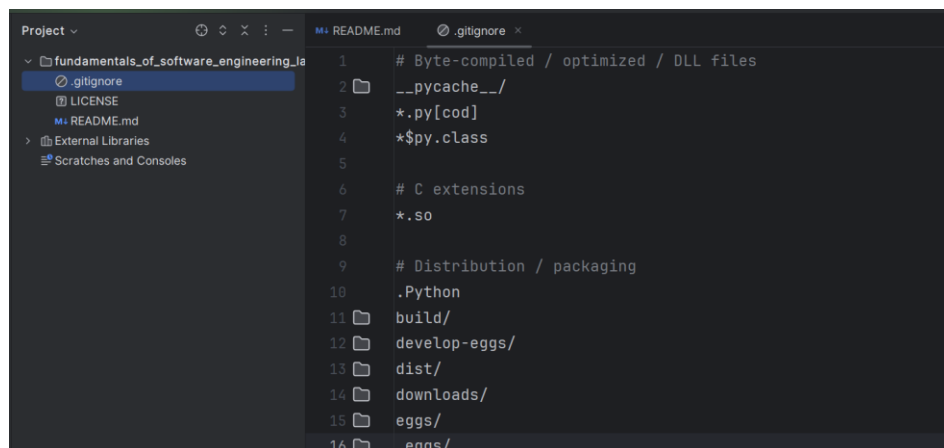
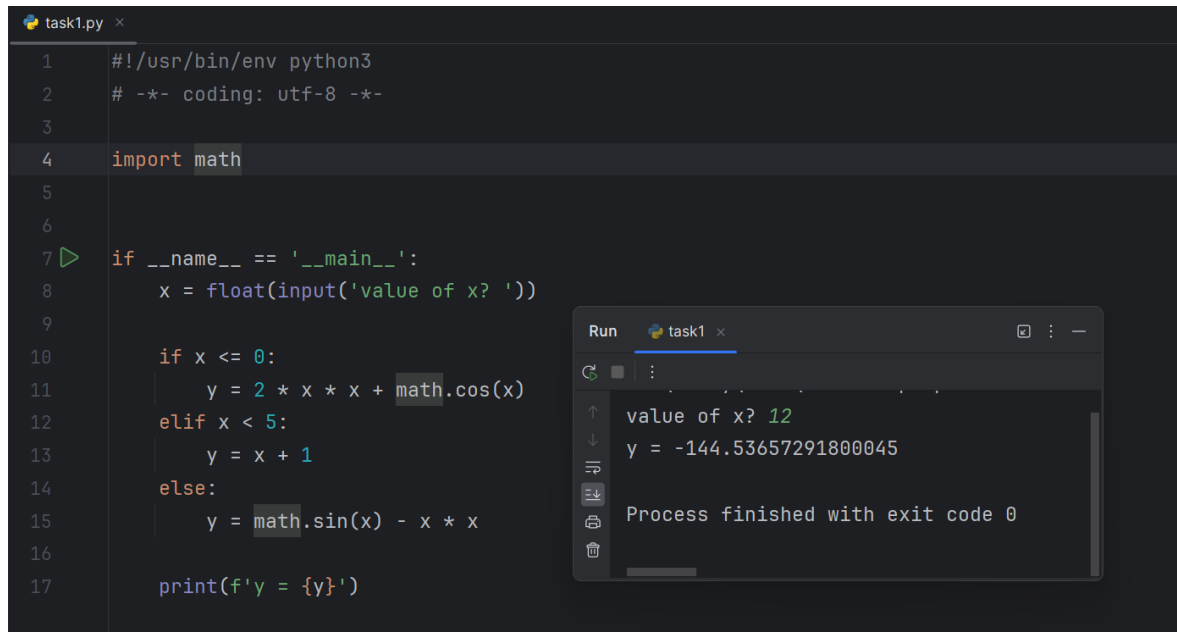


Рисунок 5 – часть .gitignore, созданного GitHub

2. Проработаем примеры лабораторной работы, фиксируя изменения и оформляя код согласно PEP-8. Построить UML-диаграммы деятельности для 4 и 5 заданий:

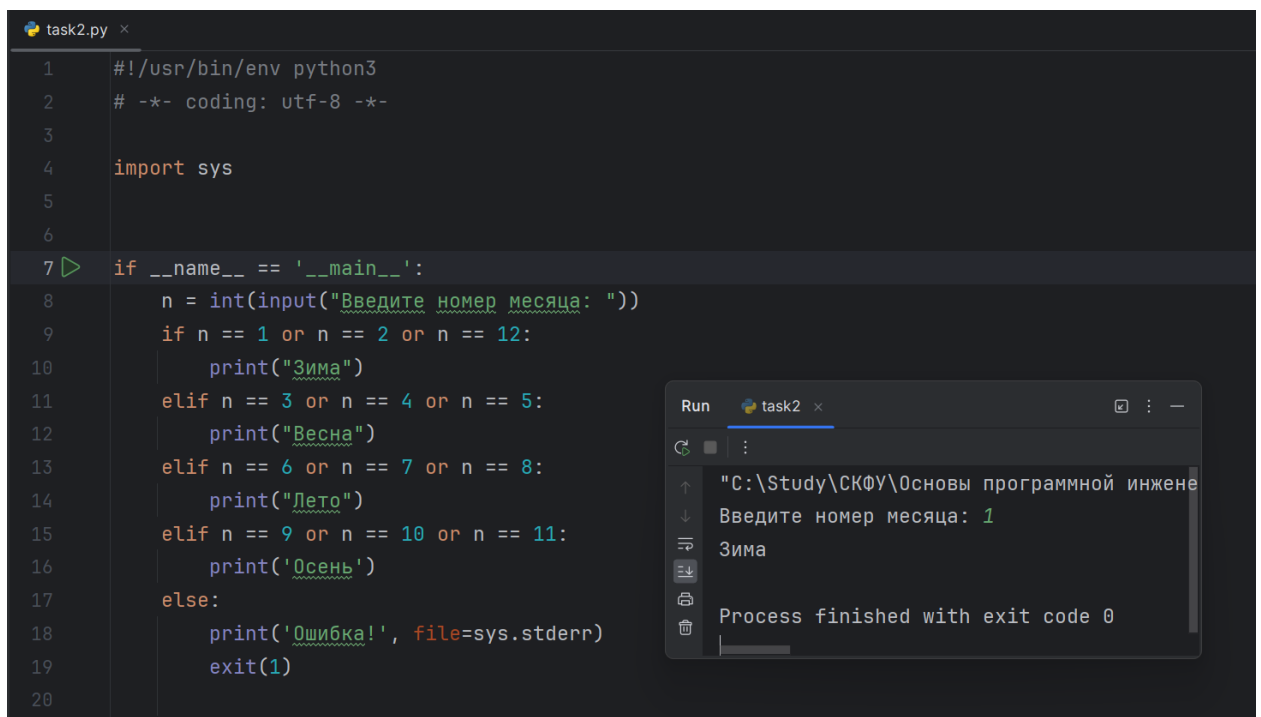


```
task1.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import math
5
6
7  if __name__ == '__main__':
8      x = float(input('value of x? '))
9
10
11     if x <= 0:
12         y = 2 * x * x + math.cos(x)
13     elif x < 5:
14         y = x + 1
15     else:
16         y = math.sin(x) - x * x
17
18     print(f'y = {y}')
```

Run task1 x

value of x? 12
y = -144.53657291800045
Process finished with exit code 0

Рисунок 6 – Нахождение значения функции и вывод программы (задание №1)



```
task2.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6
7  if __name__ == '__main__':
8      n = int(input("Введите номер месяца: "))
9      if n == 1 or n == 2 or n == 12:
10         print("Зима")
11     elif n == 3 or n == 4 or n == 5:
12         print("Весна")
13     elif n == 6 or n == 7 or n == 8:
14         print("Лето")
15     elif n == 9 or n == 10 or n == 11:
16         print('Осень')
17     else:
18         print('Ошибка!', file=sys.stderr)
19         exit(1)
20
```

Run task2 x

"C:\Study\СКФУ\Основы программной инженерии"
Введите номер месяца: 1
Зима
Process finished with exit code 0

Рисунок 7 – Вывод времени года по номеру месяца (задание №2) (1)

```
task2.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6
7  if __name__ == '__main__':
8      n = int(input("Введите номер месяца: "))
9      if n == 1 or n == 2 or n == 12:
10         print("Зима")
11     elif n == 3 or n == 4 or n == 5:
12         print("Весна")
13     elif n == 6 or n == 7 or n == 8:
14         print("Лето")
15     elif n == 9 or n == 10 or n == 11:
16         print('Осень')
17     else:
18         print('Ошибка!', file=sys.stderr)
19         exit(1)
20
```

Run task2 x

"C:\Study\СКФУ\Основы программной инженерии
Введите номер месяца: 123
Ошибка!
Process finished with exit code 1

Рисунок 8 – Вывод времени года по номеру месяца (задание №2) (2)

```
task3.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import math
5
6
7  if __name__ == '__main__':
8      n = int(input('Value of n? '))
9      x = float(input('Value of x? '))
10
11     S = 0.0
12     for k in range(1, n + 1):
13         a = math.log(k * x) / (k * k)
14         S += a
15     print(f'S = {S}')
```

Run task2 x task3 x

"C:\Study\СКФУ\Основы программной инженерии
Value of n? 5
Value of x? 6
S = 3.068814808882306
Process finished with exit code 0

Рисунок 9 – Вычисление конечной суммы и вывод работы программы (задание №3)

```

task4.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import math
5  import sys
6
7
8  > if __name__ == '__main__':
9      a = float(input('Value of a? '))
10     if a < 0:
11         print('Illegal value of a', file=sys.stderr)
12         exit(1)
13
14     x, eps = 1, 1e-10
15     while True:
16         xp = x
17         x = (x + a / x) / 2
18         if math.fabs(x - xp) < eps:
19             break
20
21     print(f'x = {x}\nX = {math.sqrt(a)}')

```

Run task4 x

"C:\Study\СКФУ\Основы программной инженерии"

Value of a? 20

x = 4.47213595499958

X = 4.47213595499958

Process finished with exit code 0

Рисунок 10 – Нахождение квадратного корня числа и его сравнение с результатом метода sqrt() стандартной библиотеки Python (задание №4)

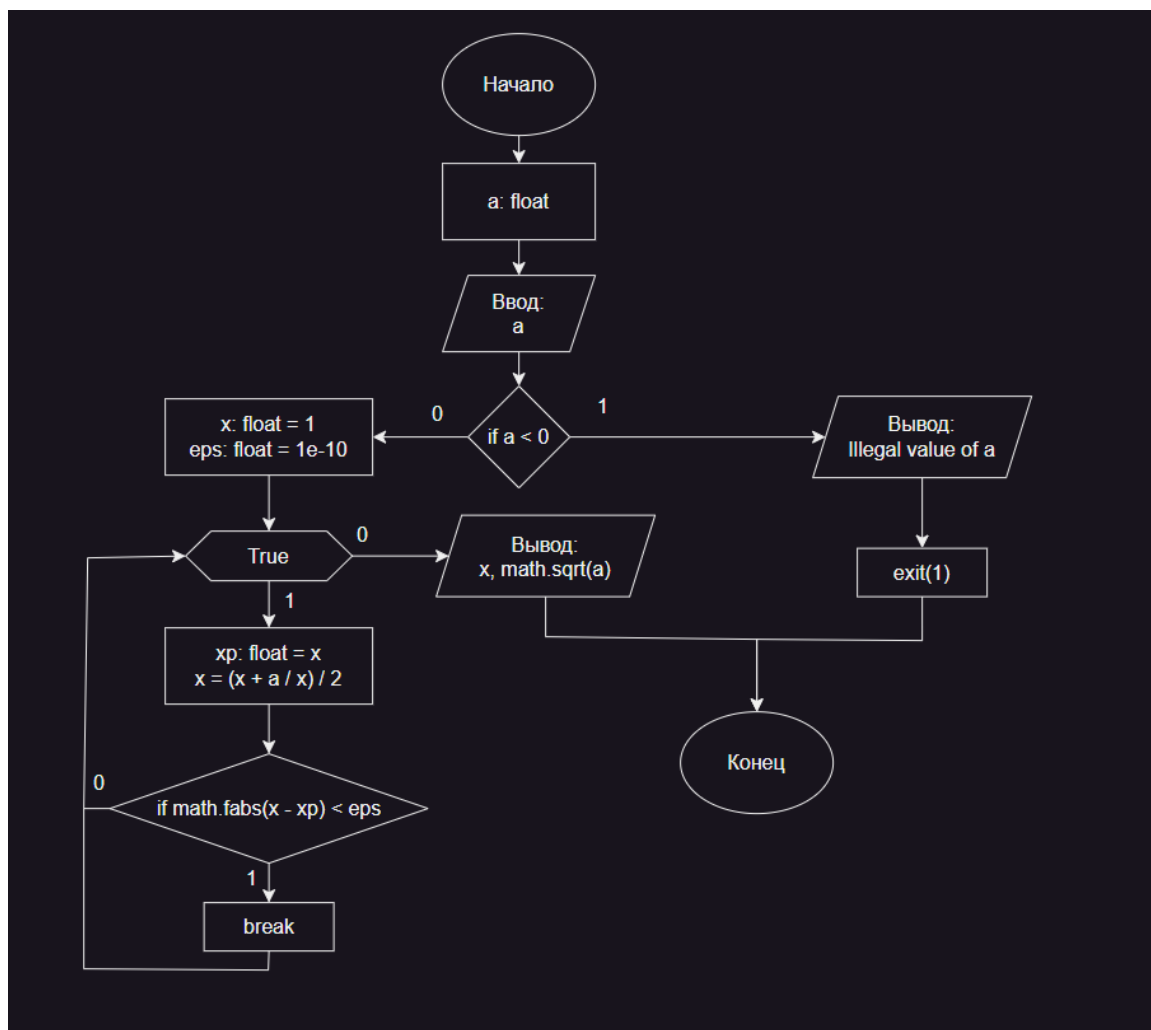


Рисунок 11 – Диаграмма для программы задания №4

```

task5.py x
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  # Точность вычислений.
10 EPS = 1e-10
11
12 if __name__ == '__main__':
13     x = float(input('Value of x? '))
14     if x == 0:
15         print('Illegal value of x', file=sys.stderr)
16         exit(1)
17
18     a = x
19     S, k = a, 1
20
21     # Найти сумму членов ряда.
22     while math.fabs(a) > EPS:
23         a *= x * k / (k + 1) ** 2
24         S += a
25         k += 1
26
27     # Вывести значение функции.
28     print(f"Ei({x}) = {EULER + math.log(math.fabs(x)) + S}")

```

Run task5 x

```

"C:\Study\СКФУ\Основы программной инженерии
Value of x? 3
Ei(3.0) = 9.93383257061422
Process finished with exit code 0

```

Рисунок 12 – Вычисление значения специальной (интегральной показательной) функции (задание №5)

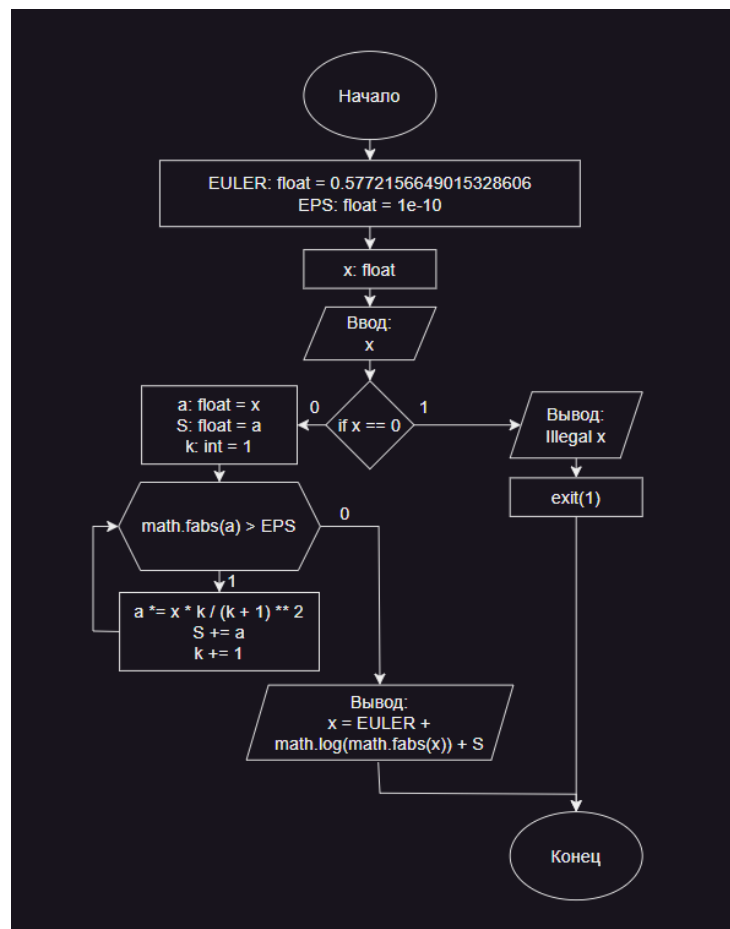
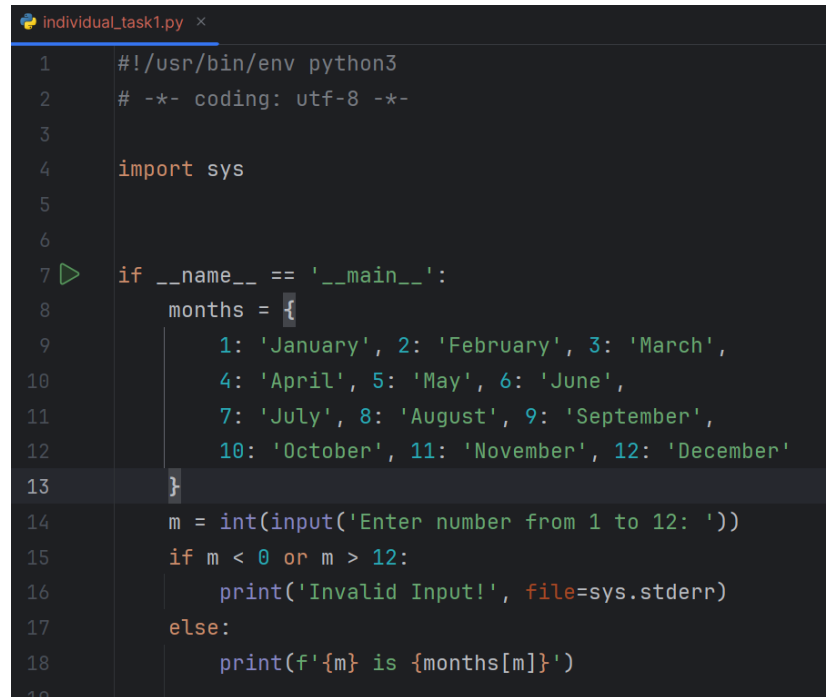


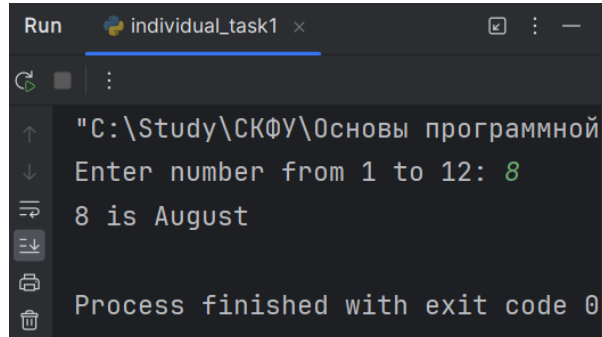
Рисунок 13 – Диаграмма для программы задания №5

3. Выполним индивидуальные задания и построим UML-диаграммы для них:



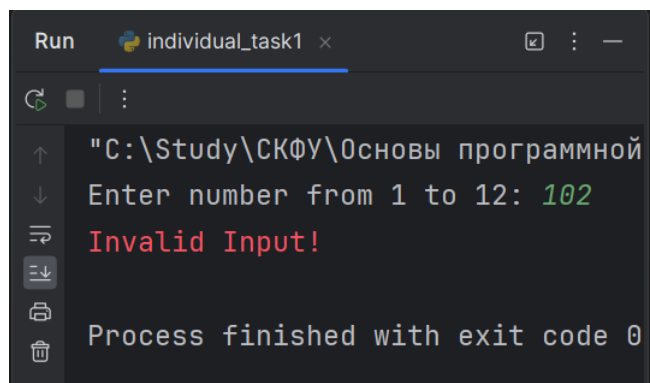
```
individual_task1.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6
7  if __name__ == '__main__':
8      months = {
9          1: 'January', 2: 'February', 3: 'March',
10         4: 'April', 5: 'May', 6: 'June',
11         7: 'July', 8: 'August', 9: 'September',
12         10: 'October', 11: 'November', 12: 'December'
13     }
14     m = int(input('Enter number from 1 to 12: '))
15     if m < 0 or m > 12:
16         print('Invalid Input!', file=sys.stderr)
17     else:
18         print(f'{m} is {months[m]}')
```

Рисунок 14 – Вывод названия месяца по его номеру (задание №1)



```
Run individual_task1 x
"C:\Study\СКФУ\Основы программной
Enter number from 1 to 12: 8
8 is August
Process finished with exit code 0
```

Рисунок 15 – Результат работы (1)



```
Run individual_task1 x
"C:\Study\СКФУ\Основы программной
Enter number from 1 to 12: 102
Invalid Input!
Process finished with exit code 0
```

Рисунок 16 – Результат работы (2)

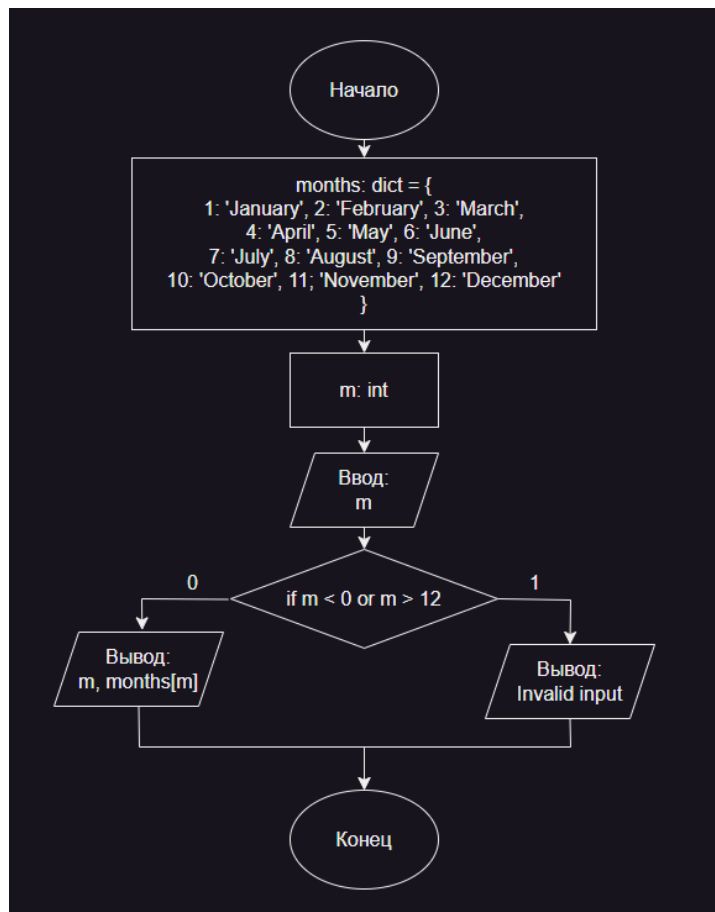


Рисунок 17 – Диаграмма программы индивидуального задания №1

```

individual_task2.py
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5  from math import sqrt
6
7
8  if __name__ == '__main__':
9      a, b, c = [float(num) for num in input('Enter a, b, c: ').split()]
10     if a == 0:
11         print('Error: a can\'t be null!', file=sys.stderr)
12     else:
13         discriminant = b**2 - 4*a*c
14         if discriminant >= 0:
15             x1, x2 = (-b + sqrt(discriminant)) / (2 * a), (-b - sqrt(discriminant)) / (2 * a)
16             x3, x4 = sqrt(x1), -1*sqrt(x2)
17             print(f'Real roots:\nx1: {x1}\nx2: {x2}\nx3: {x3}\nx4: {x4}')
18         else:
19             print('No real roots!')

```

Рисунок 18 – Исследование биквадратного уравнения $ax^4 + bx^2 + c = 0$
(задание №2)

```

"C:\Study\СКФУ\Основы программной инженерии\Лаб...
Enter a, b, c: 1 -3 2
Real roots:
x1: 2.0
x2: 1.0
x3: 1.4142135623730951
x4: -1.0

Process finished with exit code 0

```

Рисунок 19 – Результат работы (1)

```

"C:\Study\СКФУ\Основы программной инженерии\Л...
Enter a, b, c: 2 4 5
No real roots!

Process finished with exit code 0

```

Рисунок 20 – Результат работы (2)

```

"C:\Study\СКФУ\Основы программной инженерии\Л...
Enter a, b, c: 0 1 2
Error: a can't be null!

Process finished with exit code 0

```

Рисунок 21 – Результат работы (3)

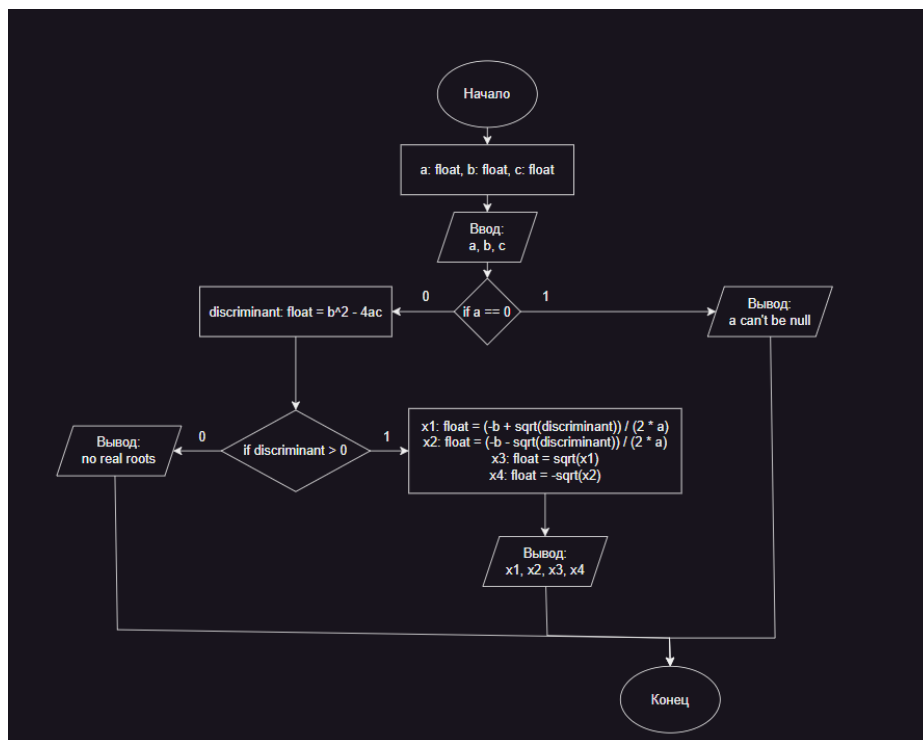


Рисунок 22 – Диаграмма программы индивидуального задания №2

```
Project Alt+1 sk3.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      result = []
6      for i in range(10, 100):
7          num_sum = sum([int(n) for n in str(i)])
8          if i % num_sum == 0:
9              result.append(i)
10     print(result)
```

Run individual_task3 x

"C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.2\fundamentals_of_software_engineer
[10, 12, 18, 20, 21, 24, 27, 30, 36, 40, 42, 45, 48, 50, 54, 60, 63, 70, 72, 80, 81, 84, 90]

Process finished with exit code 0

Рисунок 23 – Нахождение среди всех двухзначных чисел тех, которые делятся на сумму своих цифр (задание №3)

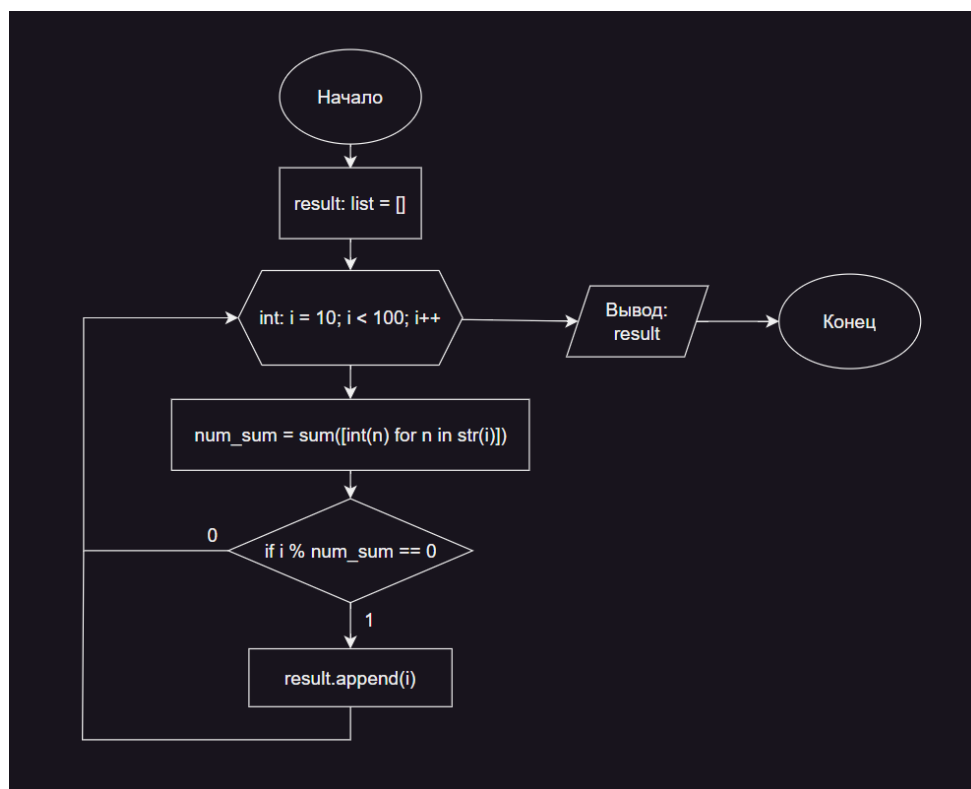


Рисунок 24 – Диаграмма программы индивидуального задания №3

```
individual_task4.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import math
5
6
7  if __name__ == '__main__':
8      x, n = float(input('Enter x: ')), int(input('Enter n: '))
9      eps = 1e-10
10     k, j = 0, 0
11     while True:
12         jp = j
13         j += (((-x**2 / 4) ** k) / (math.factorial(k) * math.factorial(k + n)))
14         k += 1
15         if math.fabs(j - jp) < eps:
16             break
17
18     print(f'J(x) = {{{(x / 2)**n} * j}}')
```

Run individual_task4 x

"C:\Study\СКФУ\Основы программной инженерии"


Enter x: 12

Enter n: 4

J(x) = 0.18249896538440707

Process finished with exit code 0

Рисунок 25 – Вычисление функции Бесселя первого рода $J_n(x)$ (задание повышенной сложности)

 **MedCalc®**
easy-to-use statistical software

Search

HOME FEATURES DOWNLOAD ORDER CONTACT FAQ MANUAL

Manual » Spreadsheet overview » Engineering functions

BesselJ function

Description

BESSELJ(**x**,**n**) returns the Bessel function of first kind. **x** is the value at which to evaluate the function, and **n** is the order of the Bessel function. If **n** is not an integer, it is truncated.

Calculator

BesselJ(,) =

Рисунок 26 – Использование стороннего калькулятора для подтверждения правильности вычисления

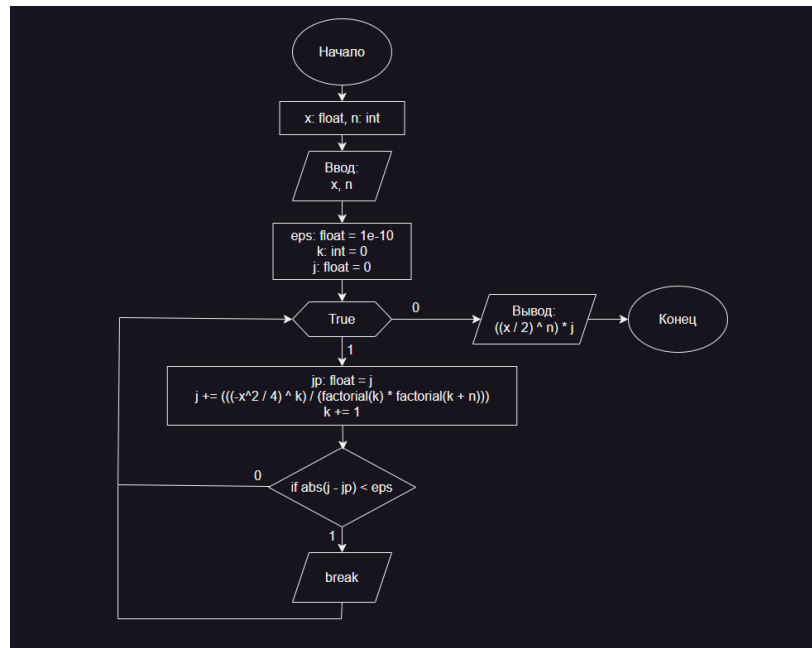


Рисунок 27 – Диаграмма программы для задания повышенной сложности

4. Зафиксируем проделанные изменения, сольем ветки и отправим на удаленный репозиторий:

```

[END]
5ab73bc (HEAD -> develop) Individual tasks are added
691c98a task5.py is added
2924adb task4.py is added
6b2ee19 task3.py is added
f9a77c0 task2.py is added
57f4999 task1.py module is added
bb47647 (origin/main, origin/HEAD, main) Update README.md
12cac97 Initial commit
~
  
```

Рисунок 28 – Коммиты ветки develop

```

(venv) PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.2\fundamentals_of_software_engineering_lab2_2> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
(venv) PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.2\fundamentals_of_software_engineering_lab2_2> git merge develop
Updating bb47647..5ab73bc
task2.py      | 19 ++++++
task3.py      | 15 ++++++
task4.py      | 21 ++++++
task5.py      | 28 ++++++
10 files changed, 167 insertions(+), 1 deletion(-)
create mode 100644 individual_task1.py
create mode 100644 individual_task2.py
create mode 100644 individual_task3.py
create mode 100644 individual_task4.py
create mode 100644 task1.py
create mode 100644 task2.py
create mode 100644 task3.py
  
```

Рисунок 29 – Слияние веток main и develop

```

Updating bb47647..5ab73bc
(venv) PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.2\fundamentals_of_software_engineering_lab2_2> git push origin main
Enumerating objects: 25, done.
Counting objects: 100% (25/25), done.
Delta compression using up to 12 threads
Compressing objects: 100% (23/23), done.
Writing objects: 100% (23/23), 3.89 KiB | 1.95 MiB/s, done.
Total 23 (delta 7), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (7/7), completed with 1 local object.
To https://github.com/PoTaTto/fundamentals_of_software_engineering_lab2_2
   bb47647..5ab73bc  main -> main
(venv) PS C:\Study\СКФУ\Основы программной инженерии\Лабораторная работа 2.2\fundamentals_of_software_engineering_lab2_2>

```

Рисунок 30 – Отправка изменений на удаленный репозиторий

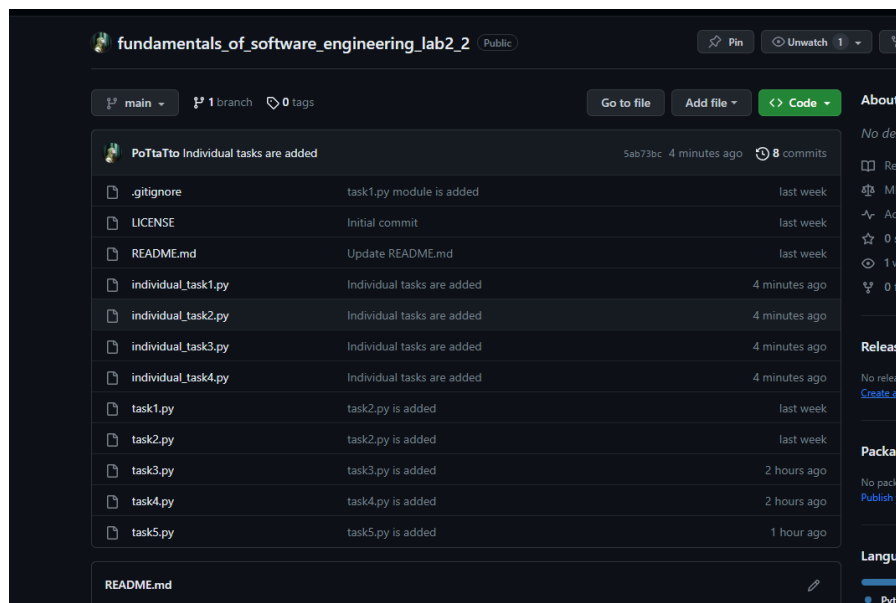


Рисунок 31 – Изменения удаленного репозитория

Ответы на контрольные вопросы:

1. Для чего нужны диаграммы деятельности UML?

Диаграмма деятельности (Activity diagram) 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, послыке сигнала, создании или уничтожении объекта либо в простом вычислении — скажем, значения выражения. Графически диаграмма деятельности представляется в виде графа, имеющего вершины и ребра.

2. Что такое состояние действия и состояние деятельности?

В потоке управления, моделируемом диаграммой деятельности, происходят различные события. Вы можете вычислить выражение, в результате чего изменяется значение некоторого атрибута или возвращается некоторое значение. Также, например, можно выполнить операцию над объектом, послать ему сигнал или даже создать его или уничтожить. Все эти выполняемые атомарные вычисления называются состояниями действия, поскольку каждое из них есть состояние системы, представляющее собой выполнение некоторого действия.

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

Когда действие или деятельность в некотором состоянии завершается, поток управления сразу переходит в следующее состояние действия или деятельности. Для описания этого потока используются переходы, показывающие путь из одного состояния действия или деятельности в другое. В UML переход представляется простой линией со стрелкой.

4. Какой алгоритм является алгоритмом разветвляющейся структуры?

Это 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 один или несколько условных операторов. Для программной реализации условия используется логическое выражение. В сложных структурах с большим числом ветвей применяют оператор выбора.

5. Чем отличается разветвляющийся алгоритм от линейного?

Разветвляющийся алгоритм отличается от линейного тем, что он позволяет программе принимать решения на основе различных условий.

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

Оператор ветвления `if` позволяет выполнить определенный набор инструкций в зависимости от некоторого условия. Существуют: `if`, `if-else`, `if-elif-else`.

7. Какие операторы сравнения используются в Python?

`>`, `<`, `<=`, `>=`, `!=`, `==`

8. Что называется простым условием? Приведите примеры:

Простые условия – это те, в которых выполняется только одна логическая операция, например: `var >= 1023`, `var == 0`, `var != 3` и т. д.

8. Что такое составное условие? Приведите примеры:

Может понадобиться получить ответа `True` или `False` в зависимости от результата выполнения двух и более простых выражений. Для этого используется составное условие из операторов `or`, `and` и `not`. Например: `var < 100 and var > 0`, `not var < 10 or var == 100` и т. д.

10. Какие логические операторы допускаются при составлении сложных условий?

В Python допускаются три основных логических оператора для составления сложных условий: `and`, `or` и `not`.

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

Да, может.

12. Какой алгоритм является алгоритмом 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:

Цикл со счетчиком (`for`), цикл с предусловием (`while`), цикл `for-each` (`for in`). Цикла с постусловием нет, но его можно сделать, используя другие алгоритмические конструкции и операторы.

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

Функция `range` возвращает неизменяемую последовательность чисел в виде объекта `range`. Функция `range` хранит только информацию о значениях `start`, `stop` и `step` и вычисляет значения по мере необходимости. Это значит, что независимо от размера диапазона, который описывает функция `range`, она всегда будет занимать фиксированный объем памяти. Обычно применяется для цикла с счетчиком.

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

```
range(15, -1, -2)
```

16. Могут ли быть циклы вложенными?

Могут.

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

Например, можно воспользоваться конструкцией `while True`. Чтобы выйти необходимо воспользоваться оператором `break`.

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` 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`. Само же определение потоков `stdout` и `stderr` находится в стандартном пакете Python `sys`. Хорошим стилем программирования является наличие вывода ошибок в стандартный поток `stderr` поскольку вывод в потоки `stdout` и `stderr` может обрабатываться как операционной системой, так и сценариями пользователя по-разному.

22. Каково назначение функции `exit`?

В Python завершить программу и передать операционной системе заданный код возврата можно посредством функции `exit`.