

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**

Выполнила:
Кузнецова Алена Валерьевна
1 курс, группа ИВТ-б-о-21-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», очная
форма обучения

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2022 г.

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.

Выполнение работы:

1. Создали общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.

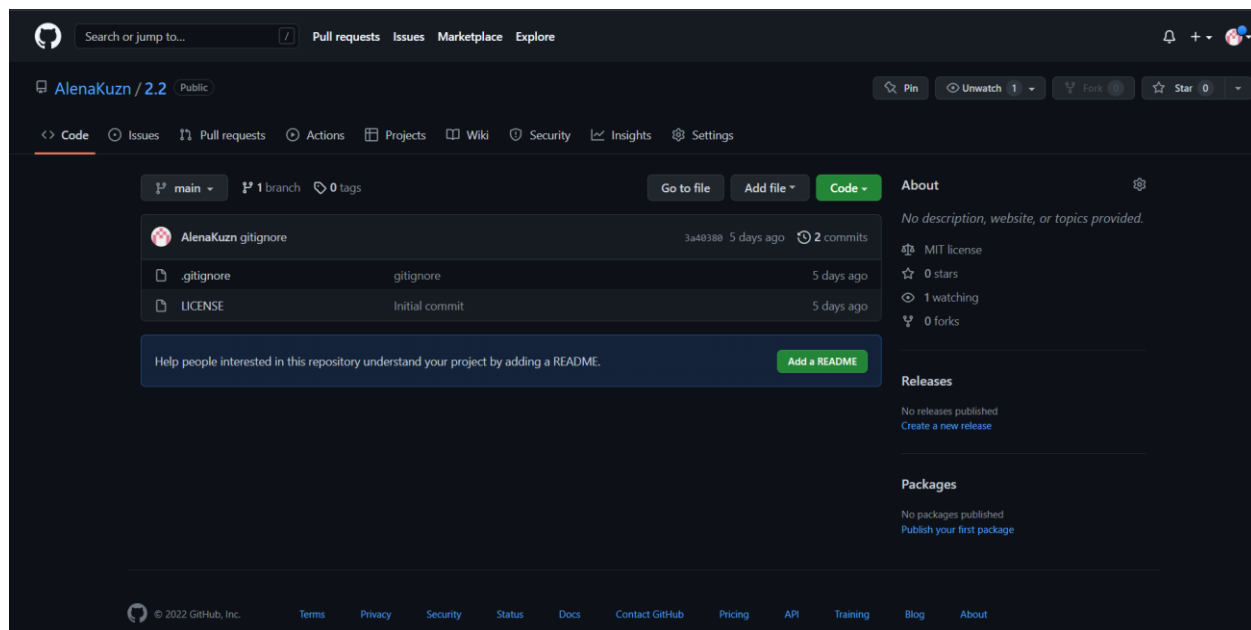


Рисунок 1 – Создание репозитория

2. Выполнили клонирование созданного репозитория.

```
C:\Users\akuzn> cd /d C:\Users\akuzn

C:\Users\akuzn>git clone https://github.com/AlenaKuzn/2.2.git
Cloning into '2.2'...
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.
```

Рисунок 2 – Клонирование репозитория

3. Дополнили файл .gitignore необходимыми правилами для работы с IDE PyCharm.

```
1 # Created by https://www.toptal.com/developers/gitignore/api/python,pycharm
2 # Edit at https://www.toptal.com/developers/gitignore?templates=python,pycharm
3
4 ### PyCharm ###
5 # Covers JetBrains IDEs: IntelliJ, RubyMine, PhpStorm, AppCode, PyCharm, CLion, Android Studio, WebStorm and Rider
6 # Reference: https://intellij-support.jetbrains.com/hc/en-us/articles/206544839
7
8 # User-specific stuff
9 .idea/**/workspace.xml
10 .idea/**/tasks.xml
11 .idea/**/usage.statistics.xml
12 .idea/**/dictionaries
13 .idea/**/shelf
14
15 # AWS User-specific
16 .idea/**/aws.xml
17
18 # Generated files
19 .idea/**/contentModel.xml
20
21 # Sensitive or high-churn files
22 .idea/**/dataSources/
23 .idea/**/dataSources.ids
24 .idea/**/dataSources.local.xml
25 .idea/**/sqlDataSources.xml
26 .idea/**/dynamic.xml
27 .idea/**/uiDesigner.xml
28 .idea/**/dbnavigator.xml
29
30 # Gradle
31 .idea/**/gradle.xml
32 .idea/**/libraries
33
34 # Gradle and Maven with auto-import
35 # When using Gradle or Maven with auto-import, you should exclude module files,
36 # since they will be recreated, and may cause churn. Uncomment if using
37 # auto-import.
38 # .idea/artifacts
39 # .idea/compiler.xml
```

Рисунок 3 – Дополнение файла .gitignore

4. Организуйте свой репозиторий в соответствии с моделью git-flow.

```
C:\Users\akuzn\2.2>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/akuzn/2.2/.git/hooks]
```

Рисунок 4 - Моделью ветвления git-flow

5. Проработали примеры лабораторной работы.







	AlenaKuzn 5
..	
 example1.py	1'
 example2.py	2
 example3.py	3
 example4.py	4
 examply5.py	5

Рисунок 5 – Примеры из лабораторной работы

6. Для примеров 4 и 5 построили UML-диаграмму деятельности.

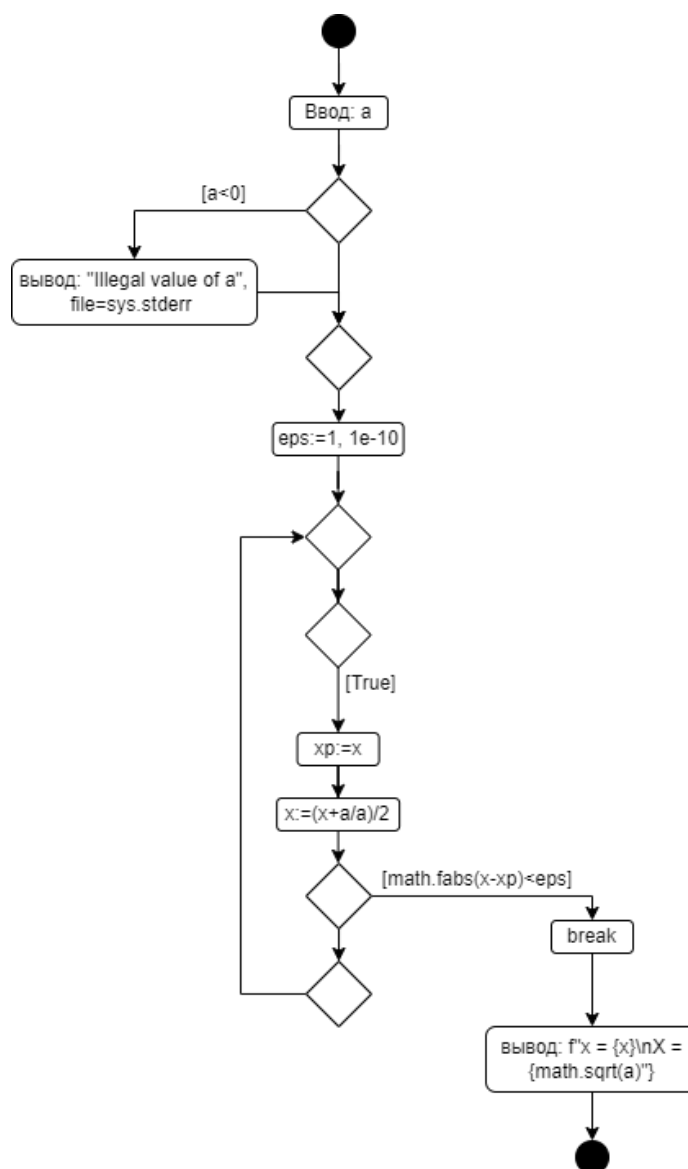


Рисунок 6 - UML-диаграмма программы 4 примера

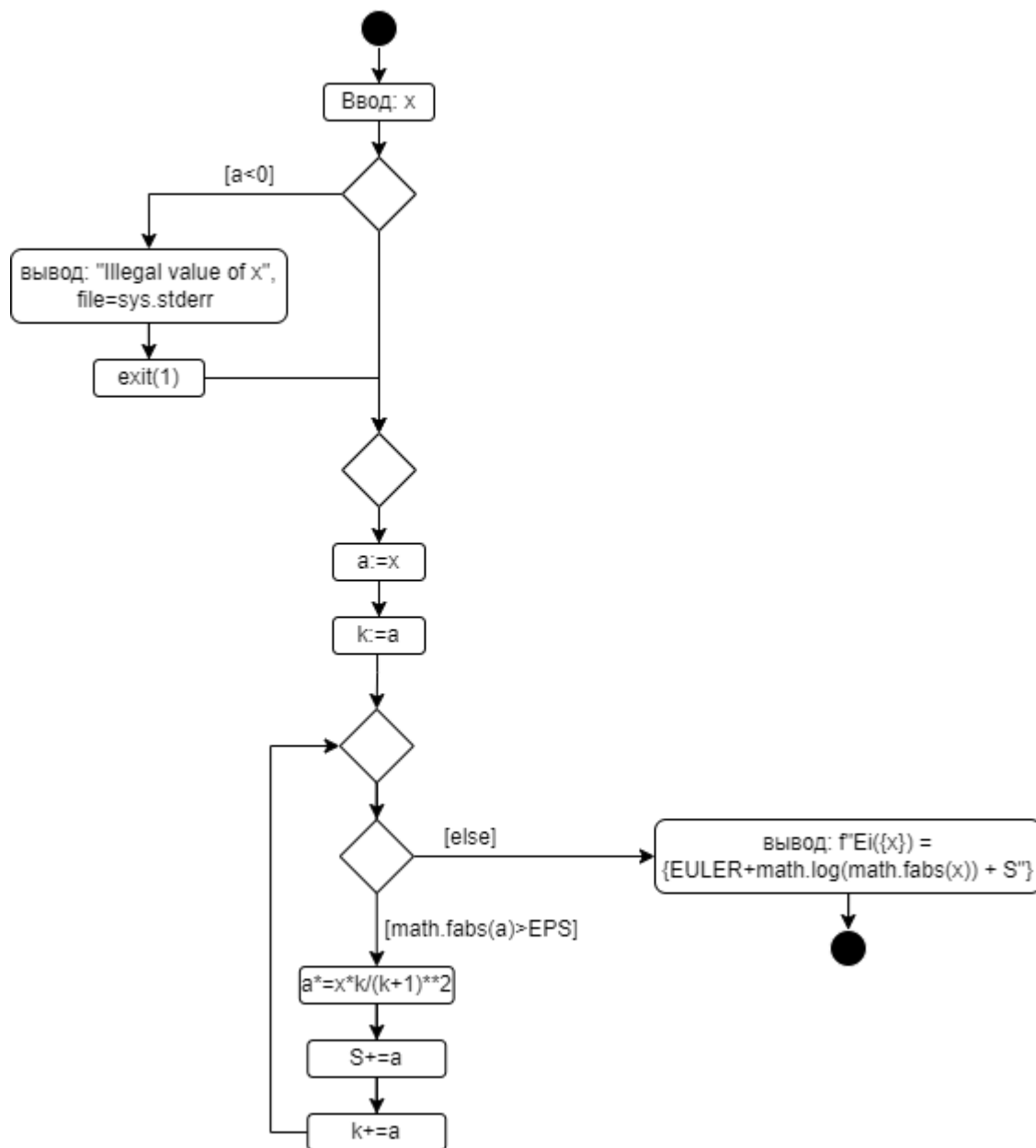


Рисунок 7 - UML-диаграмма программы 5 примера

7. Выполнили индивидуальные задания.

Задание 1

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

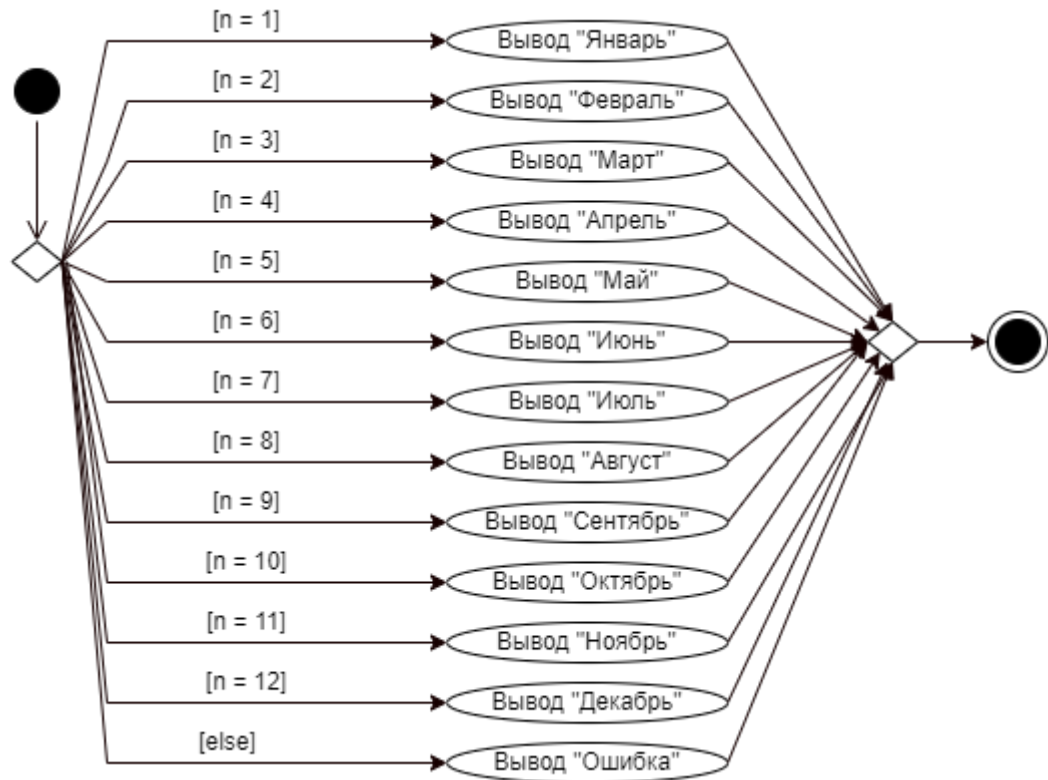


Рисунок 8 - UML-диаграмма индивидуального задания 1

Задание 2

Провести исследование биквадратного уравнения, где, и действительные числа. Если действительных корней нет, то об этом должно быть выдано сообщение, иначе должны быть выданы 2 или 4 действительных корня.

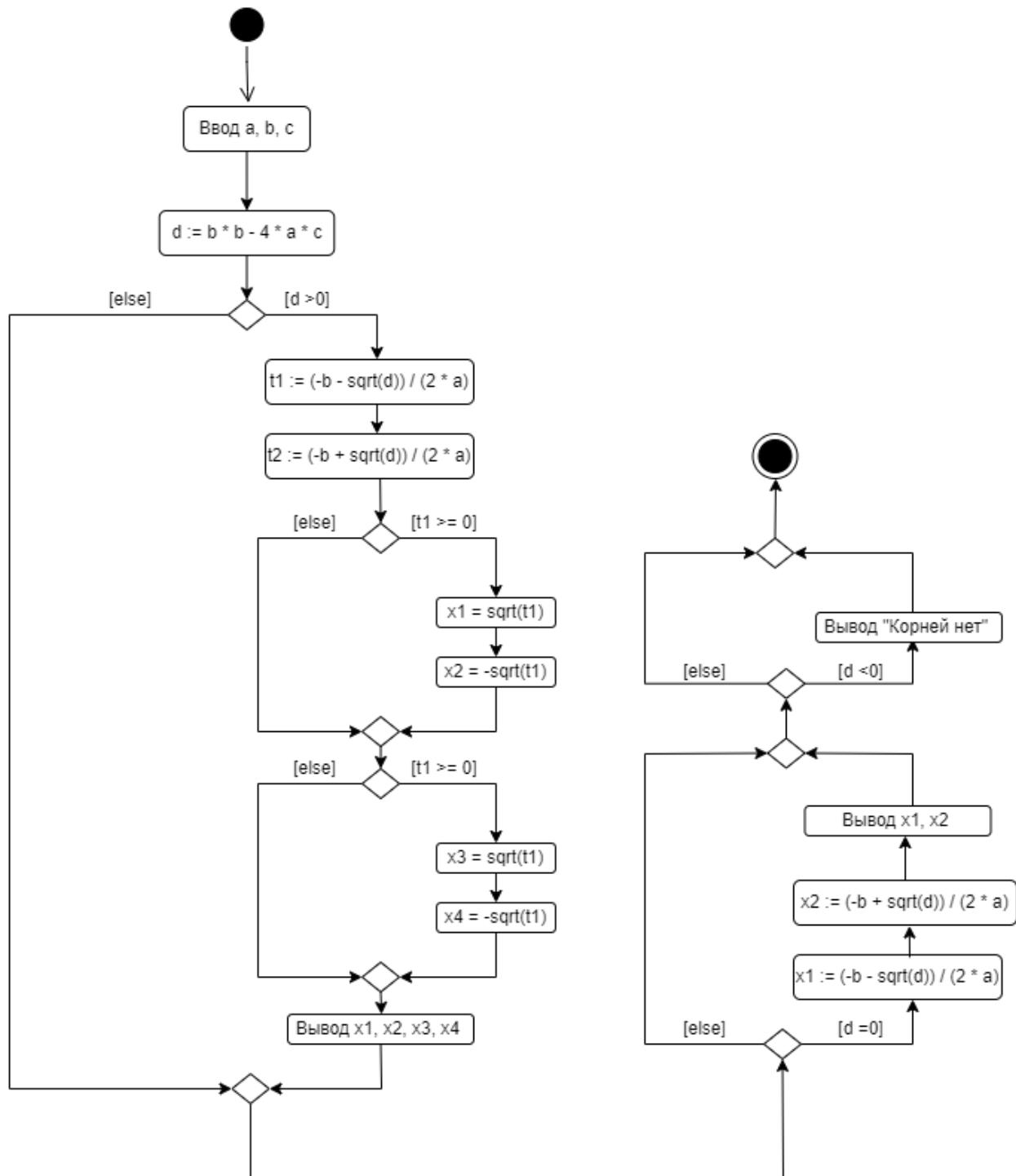


Рисунок 9 - UML-диаграмма индивидуального задания 2

Задание 3

Определить среди всех двузначных чисел те, которые делятся на сумму своих цифр.

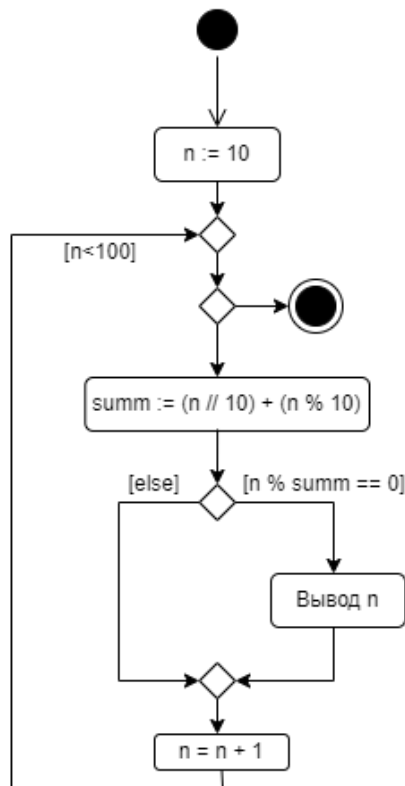


Рисунок 10 - UML-диаграмма индивидуального задания 3

8. Выполнили слияние ветки для разработки с веткой main / master.

```
C:\Users\akuzn\2.2\individ>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\Users\akuzn\2.2\individ>git merge develop
Updating 3a40380..da2085c
Fast-forward
 example/example1.py | 11 ++++++++
 example/example2.py | 15 ++++++++
 example/example3.py | 10 ++++++++
 example/example4.py | 15 ++++++++
 example/examply5.py | 21 ++++++++
 individ/indyvid1.py | 31 ++++++++
 individ/indyvid2.py | 33 ++++++++
 individ/indyvid3.py | 10 ++++++++
 uuml/individ1.png   | Bin 0 -> 52622 bytes
 uuml/individ2.png   | Bin 0 -> 49372 bytes
 uuml/individ3.png   | Bin 0 -> 13891 bytes
11 files changed, 146 insertions(+)
 create mode 100644 example/example1.py
 create mode 100644 example/example2.py
 create mode 100644 example/example3.py
 create mode 100644 example/example4.py
 create mode 100644 example/examply5.py
 create mode 100644 individ/indyvid1.py
 create mode 100644 individ/indyvid2.py
 create mode 100644 individ/indyvid3.py
 create mode 100644 uuml/individ1.png
 create mode 100644 uuml/individ2.png
 create mode 100644 uuml/individ3.png

C:\Users\akuzn\2.2\individ>git push
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
```

Рисунок 11 – Выполнили слияние ветки

Контрольные вопросы:

1. Для чего нужны диаграммы деятельности UML?

Позволяет наглядно визуализировать алгоритм программы.

2. Что такое состояние действия и состояние деятельности?

Состояние действия - частный вид состояния деятельности, а конкретнее – такое состояние, которое не может быть подвергнуто дальнейшей декомпозиции.

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?

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: `a == b`

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 a == c)`

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 — это вид алгоритма, в процессе выполнения которого одно или несколько действий нужно повторить.

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.

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

`range(15, 0, 2)`

16. Могу 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?

Используется для выхода из цикла.

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

Оператор continue используется только в циклах. В операторах for , while , do while , оператор continue выполняет пропуск оставшейся части кода тела цикла и переходит к следующей итерации цикла.

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.