

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

Кафедра инфокоммуникаций

Институт цифрового развития

ОТЧЁТ

по лабораторной работе №2.2

Дисциплина: «Программирование на Python»

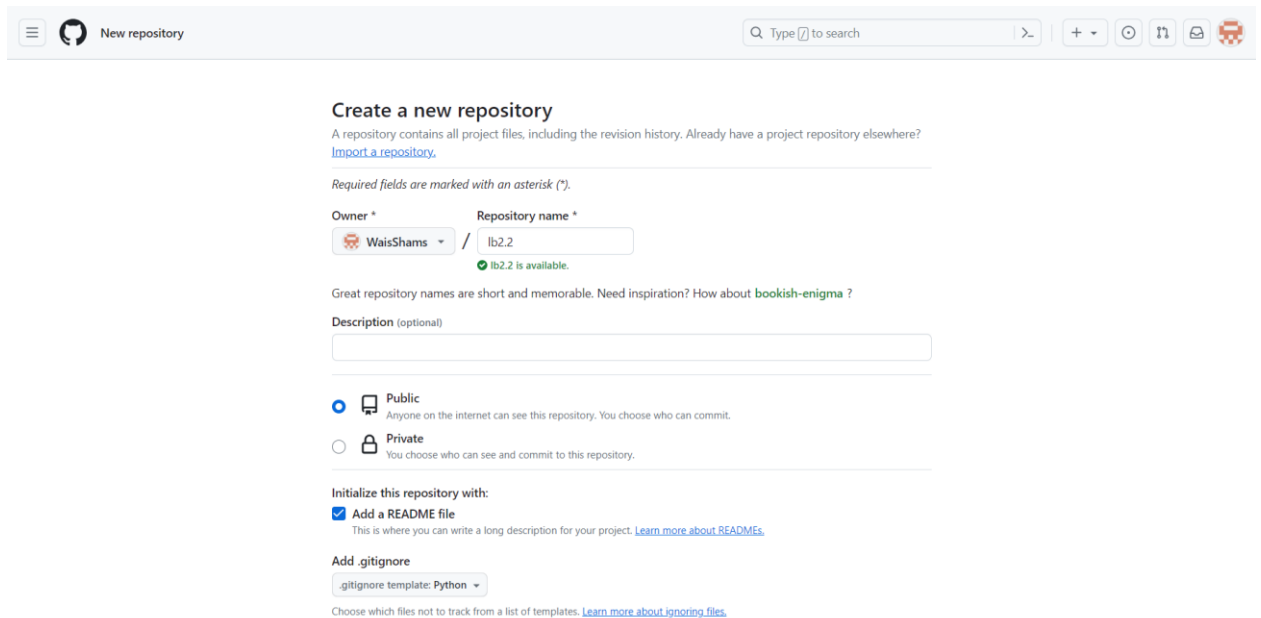
Тема: «Условные операторы и циклы в языке Python»

Выполнил: студент 2 курса
группы ИВТ-б-о-22-1
Шамс Вайсудин

Ставрополь 2022

Выполнение работы.

1. Создал репозиторий в GitHub «rep 2.2» в который добавил .gitignore, который дополнил правилами для работы с IDE PyCharm с ЯП Python, выбрал лицензию MIT, клонировал его на лок. сервер и организовал в соответствие с моделью ветвления git-flow.



New repository

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * Repository name *

WaisShams / lb2.2

lb2.2 is available.

Great repository names are short and memorable. Need inspiration? How about [bookish-enigma](#) ?

Description (optional)

☒ Public
Anyone on the internet can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

Initialize this repository with:

☒ Add a README file
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Python

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

```
C:\>git clone https://github.com/Arsen445/LB2.2.git
Cloning into 'LB2.2'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 7 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (7/7), 4.36 KiB | 744.00 KiB/s, done.
Resolving deltas: 100% (1/1), done.
```

```
C:\LB2.2>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:/LB2.2/.git/hooks]
```

2. Создал проект PyCharm в папке репозитория, проработал примеры ЛР.

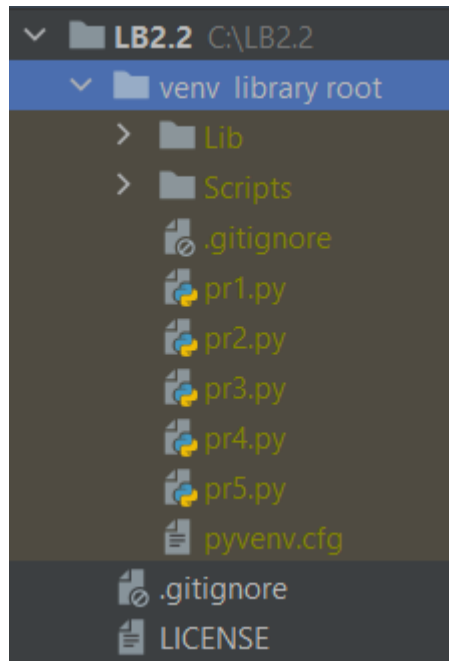


Рисунок 2.1 Примеры в проекте

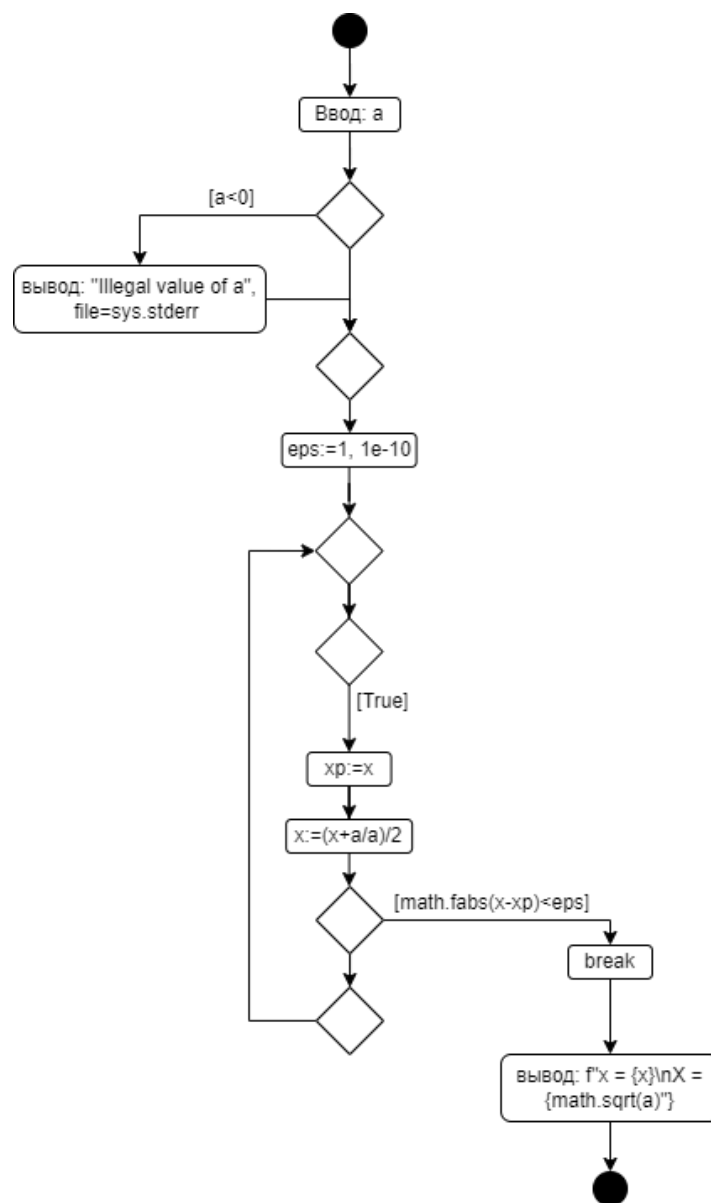


Рисунок 2.2 UML-диаграмма программы 4 примера

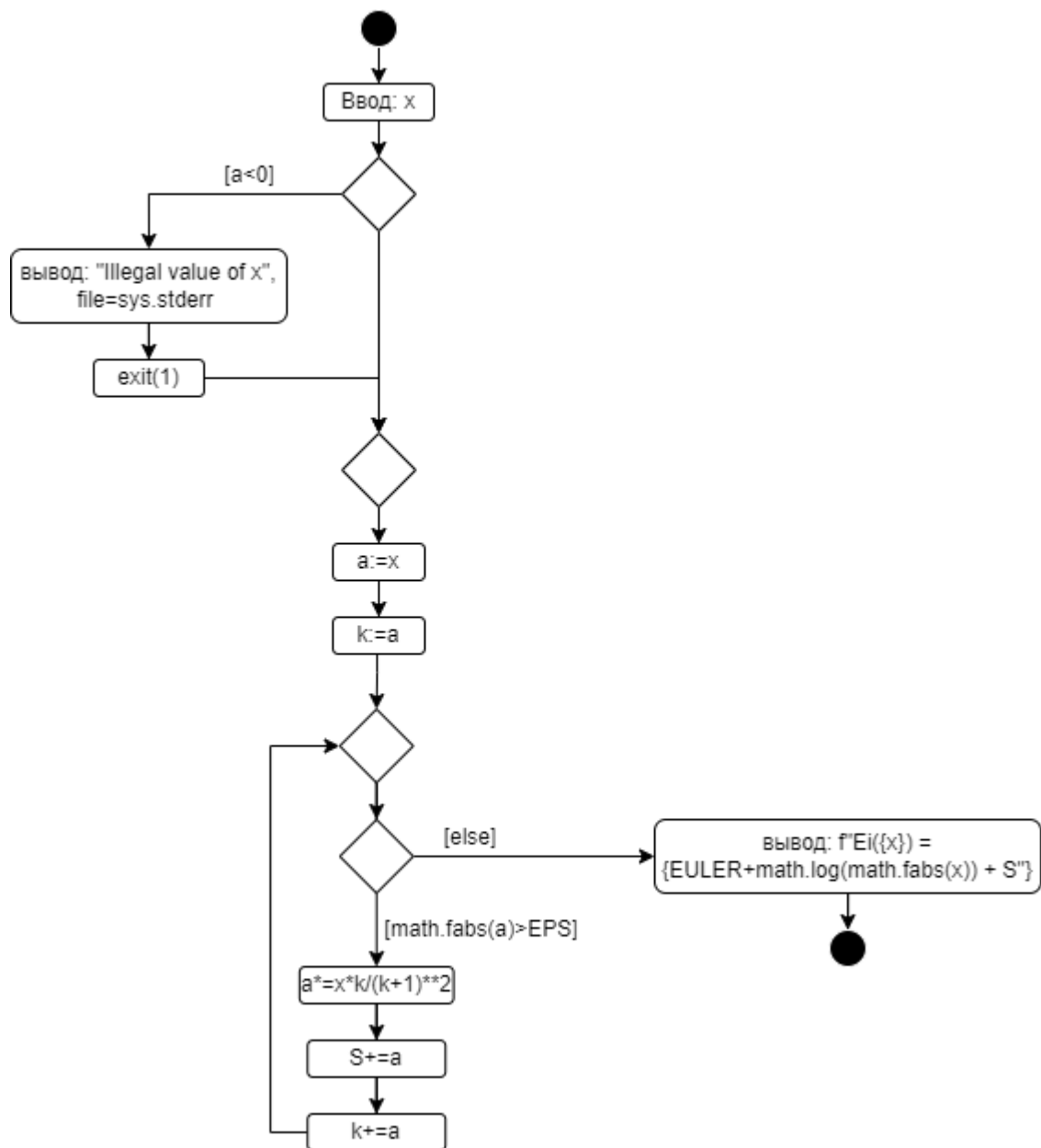


Рисунок 2.3 UML-диаграмма программы 5 примера

3. Выполнил индивидуальные задания согласно своему (24) варианту.

Построил UML диаграммы программ.

```

2.2 C:\LB2.2
venv library root
ind
ind1.py
Lib
pr
Scripts
gitignore
pyvenv.cfg

1 n = int(input('Сколько кВт/ч вы израсходовали?\n'))
2 if (n <= 250):
3     print('По тарифу 7 р. за кВт/ч вы должны ', n * 7, ' р. компании.')
4 elif (n > 250 and n <= 300):
5     print('По тарифу 17 р. за кВт/ч вы должны ', n * 17, ' р. компании.')
6 else:
7     print('По тарифу 20 р. за кВт/ч вы должны ', n * 20, ' р. компании.')
8 elif (n > 250 and n <= 300)

ind1 x
C:\LB2.2\venv\Scripts\python.exe C:/LB2.2/venv/ind/ind1.py
Сколько кВт/ч вы израсходовали?
265
По тарифу 17 р. за кВт/ч вы должны 4505 р. компании.
  
```

Рисунок 3.1 Программа к инд. заданию №1 и ее вывод

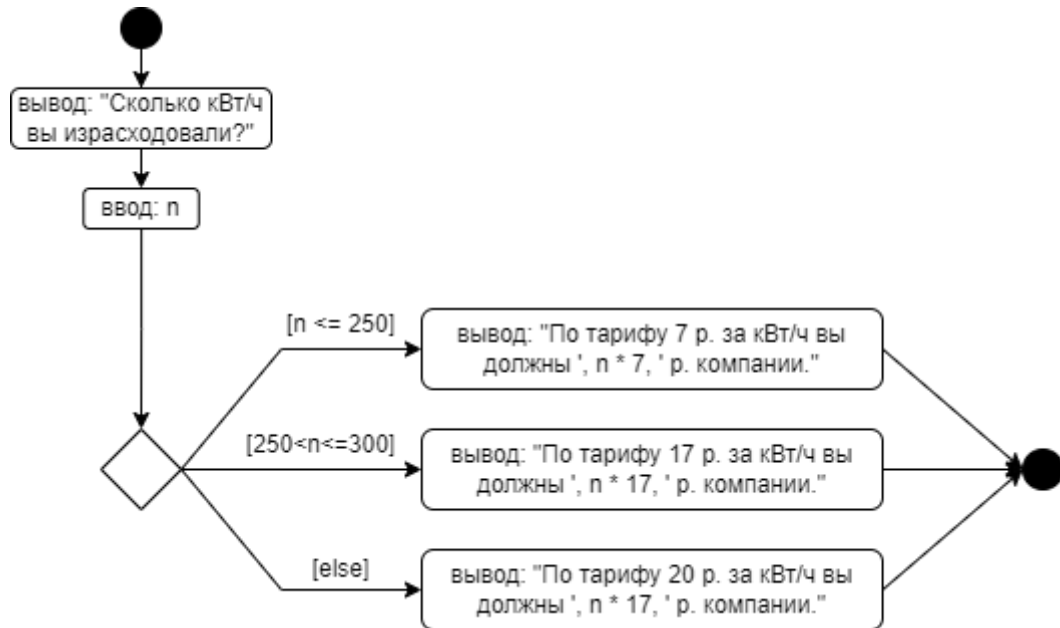


Рисунок 3.2 UML – диаграмма к программе инд. задания 1

```
2  Определить, есть ли среди трёх заданных чисел чётные.
3  """
4  a = int(input('add 1st num: '))
5  b = int(input('add 2nd num: '))
6  c = int(input('add 3rd num: '))
7  if (a % 2 == 0 or
8      b % 2 == 0 or
9      c % 2 == 0):
10     print('one or more of three added nums is even')
11 else:
12     print('no one of three added nums is even')
else
```

C:\LB2.2\venv\Scripts\python.exe C:/LB2.2/venv/ind/ind2.py
add 1st num: 5
add 2nd num: 3
add 3rd num: 1
no one of three added nums is even

Рисунок 3.3 Программа к инд. заданию №2

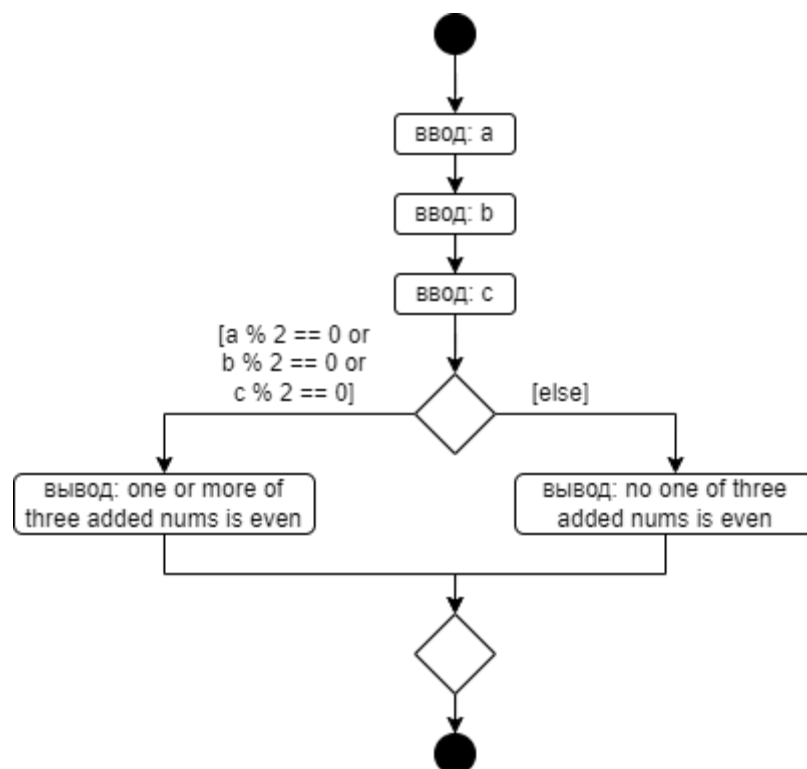


Рисунок 3.4 UML – диаграмма к программе инд. задания 2

```

1 """
2 Найти сумму целых положительных чисел, больших 20, меньших 100 и кратных 3.
3 """
4 sum = 0
5 for i in range(20, 100):
6     if i % 3 == 0:
7         sum += i
8 print(sum)
  
```

C:\LB2.2\venv\Scripts\python.exe C:/LB2.2/venv/ind/ind3.py
1620

Рисунок 3.5 Программа к инд. заданию №3

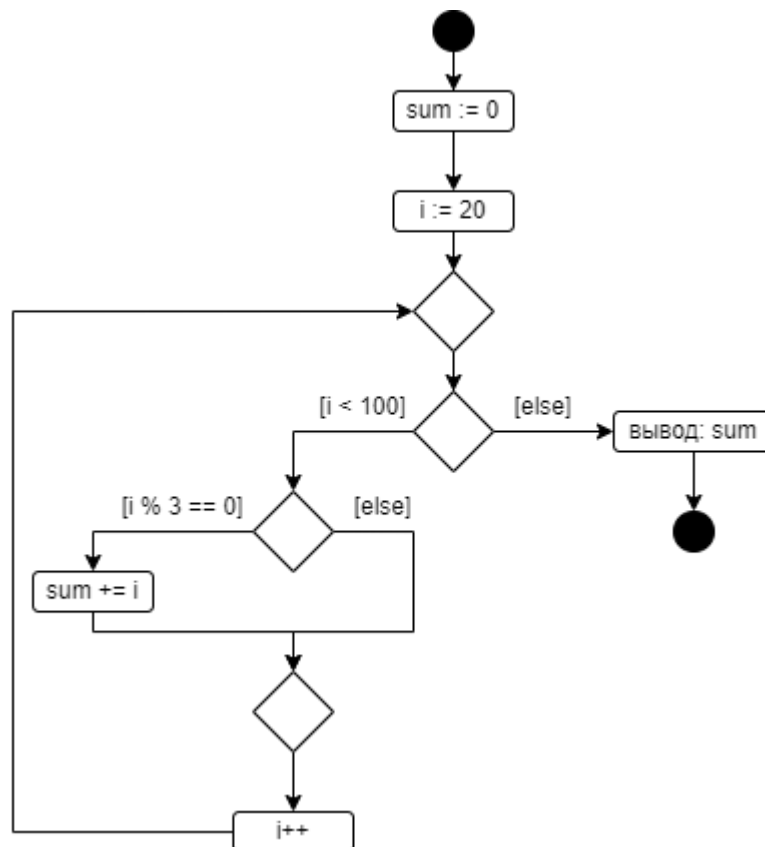


Рисунок 3.6 UML – диаграмма к программе инд. задания 3

4. Сделал коммит, выполнил слияние с веткой main, и запустил изменения в уд. репозиторий.

```

C:\LB2.2>git commit -m "added progs"
[develop 0e0206d] added progs
15 files changed, 43 insertions(+)
create mode 100644 .idea/.gitignore
create mode 100644 .idea/LB2.2.iml
create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/vcs.xml
create mode 100644 "\320\241\321\205\320\265\320\274\321\213\ind1.drawio"
create mode 100644 "\320\241\321\205\320\265\320\274\321\213\ind1.png"
create mode 100644 "\320\241\321\205\320\265\320\274\321\213\ind2.drawio"
create mode 100644 "\320\241\321\205\320\265\320\274\321\213\ind2.png"
create mode 100644 "\320\241\321\205\320\265\320\274\321\213\ind3.drawio"
create mode 100644 "\320\241\321\205\320\265\320\274\321\213\ind3.png"
create mode 100644 "\320\241\321\205\320\265\320\274\321\213/~$ind1.drawio.bkp"
create mode 100644 "\320\241\321\205\320\265\320\274\321\213/~$ind2.drawio.bkp"
create mode 100644 "\320\241\321\205\320\265\320\274\321\213/~$ind3.drawio.bkp"
  
```



```

C:\LB2.2>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\LB2.2>git merge develop
Updating 10225d6..0e0206d
Fast-forward
 .idea/.gitignore          | 3 +++
 .idea/LB2.2.iml           |10 ++++++++
 .idea/inspectionProfiles/profiles_settings.xml | 6 +++++
 .idea/misc.xml            | 4 +++++
 .idea/modules.xml         | 8 ++++++
 .idea/vcs.xml             | 6 +++++
 "\320\241\321\205\320\265\320\274\321\213\ind1.drawio" | 1 +
 "\320\241\321\205\320\265\320\274\321\213\ind1.png"   | Bin 0 -> 23003 bytes
 "\320\241\321\205\320\265\320\274\321\213\ind2.drawio" | 1 +
 "\320\241\321\205\320\265\320\274\321\213\ind2.png"   | Bin 0 -> 16363 bytes
 "\320\241\321\205\320\265\320\274\321\213\ind3.drawio" | 1 +
 "\320\241\321\205\320\265\320\274\321\213\ind3.png"   | Bin 0 -> 14193 bytes
 .../~$ind1.drawio.bkp" | 1 +
 .../~$ind2.drawio.bkp" | 1 +
 .../~$ind3.drawio.bkp" | 1 +
15 files changed, 43 insertions(+)
create mode 100644 .idea/.gitignore
create mode 100644 .idea/LB2.2.iml
create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/vcs.xml
create mode 100644 "\320\241\321\205\320\265\320\274\321\213\ind1.drawio"
create mode 100644 "\320\241\321\205\320\265\320\274\321\213\ind1.png"
create mode 100644 "\320\241\321\205\320\265\320\274\321\213\ind2.drawio"
create mode 100644 "\320\241\321\205\320\265\320\274\321\213\ind2.png"
create mode 100644 "\320\241\321\205\320\265\320\274\321\213\ind3.drawio"
create mode 100644 "\320\241\321\205\320\265\320\274\321\213\ind3.png"
create mode 100644 "\320\241\321\205\320\265\320\274\321\213/~$ind1.drawio.bkp"
create mode 100644 "\320\241\321\205\320\265\320\274\321\213/~$ind2.drawio.bkp"
create mode 100644 "\320\241\321\205\320\265\320\274\321\213/~$ind3.drawio.bkp"

C:\LB2.2>git push -u origin main
Enumerating objects: 21, done.
Counting objects: 100% (21/21), done.
Delta compression using up to 8 threads
Compressing objects: 100% (18/18), done.
Writing objects: 100% (20/20), 56.90 KiB | 9.48 MiB/s, done.
Total 20 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Arsen445/LB2.2.git
 10225d6..0e0206d  main -> main
branch 'main' set up to track 'origin/main'.

```

Рисунок 4.1 Работа в GIT CMD

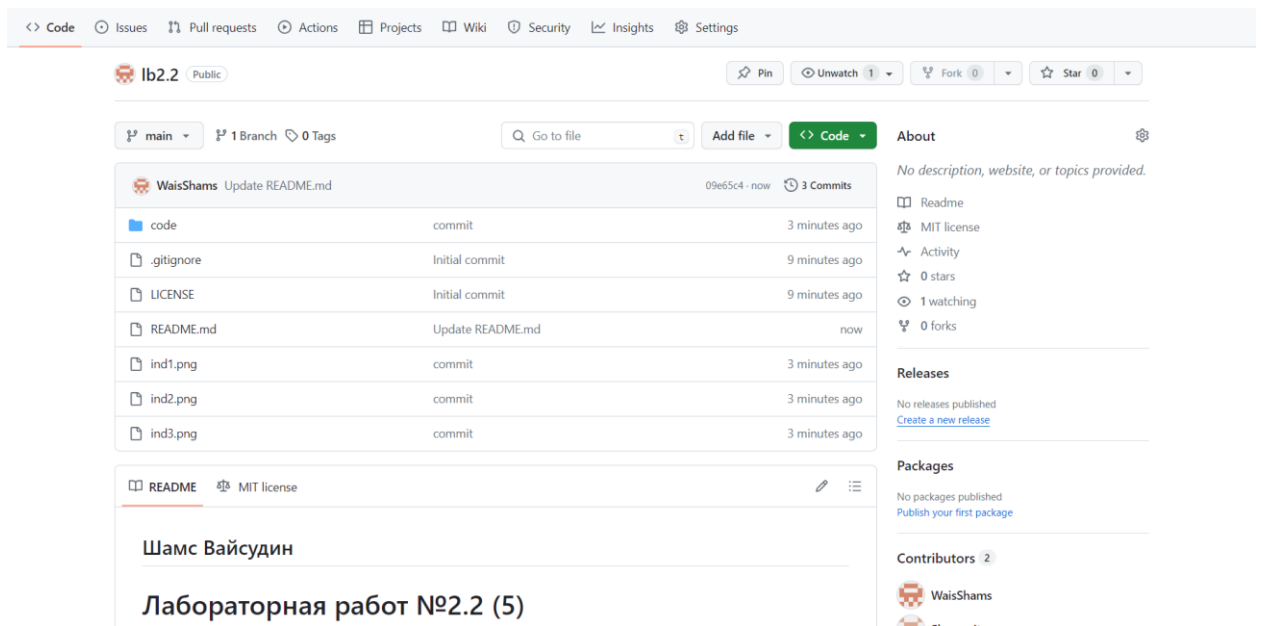


Рисунок 4.2 Изменения на уд. Сервере

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 объединённых 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. 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?

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.