

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 РАБОТЕ №2.2
дисциплины «Основы кроссплатформенного программирования»

Выполнил:
Кондратенко Даниил Витальевич
1 курс, группа ИТС-б-о-22-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи»,
направленность (профиль)
«Инфокоммуникационные системы и
сети», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Р.А., канд. тех. наук, доцент,
доцент 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.

Порядок выполнения работы:

Задание 1.

Создал новый репозиторий и клонировал его на свой компьютер.

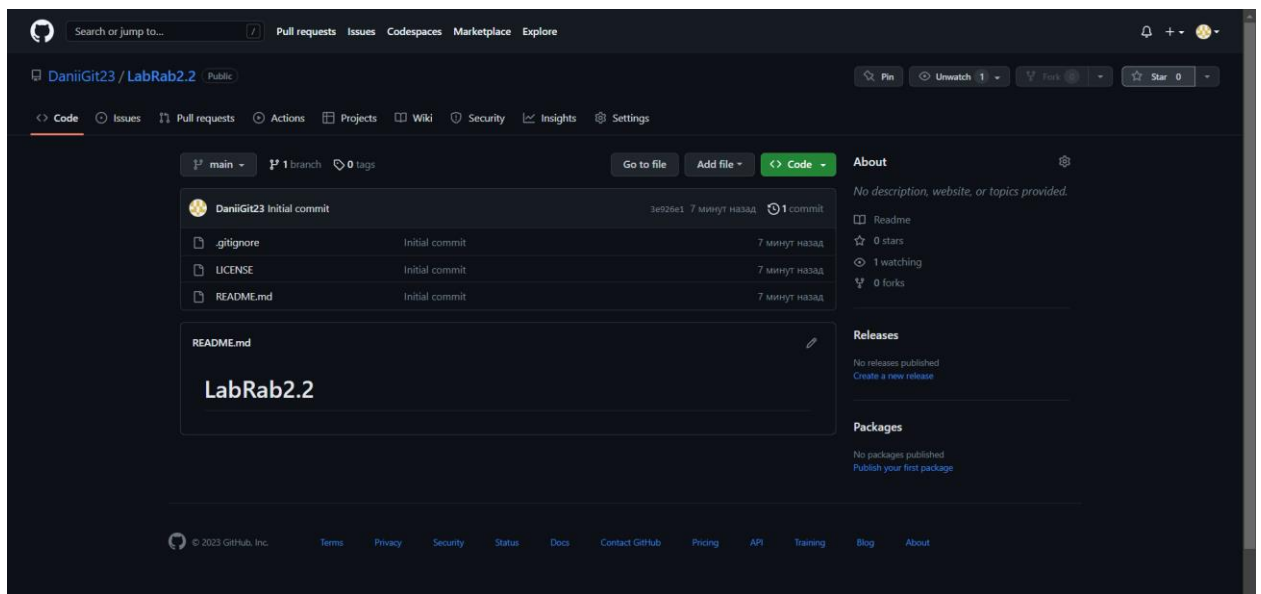


Рисунок 1. Новый репозиторий

```
C:\Users\HUAWEI>git config --global user.name "Daniil"

C:\Users\HUAWEI>git config --global user.email "kondratenko_danil.23@mail.ru"

C:\Users\HUAWEI>git clone https://github.com/DaniilGit23/LabRab2.2.git
Cloning into 'LabRab2.2'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.

C:\Users\HUAWEI>cd C:\Users\HUAWEI\LabRab2.2

C:\Users\HUAWEI\LabRab2.2>
```

Рисунок 2. Клонирование репозитория

Задание 2.

Организовал свой репозиторий в соответствии с моделью ветвления git-flow.

```
C:\Users\HUAWEI\LabRab2.2>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? [] v
Hooks and filters directory? [C:/Users/HUAWEI/LabRab2.2/.git/hooks]

C:\Users\HUAWEI\LabRab2.2>git branch
* develop
  main

C:\Users\HUAWEI\LabRab2.2>
```

Рисунок 2. Модель ветвления git-flow

Задание 3.

Создал проект PyCharm в папке репозитория.

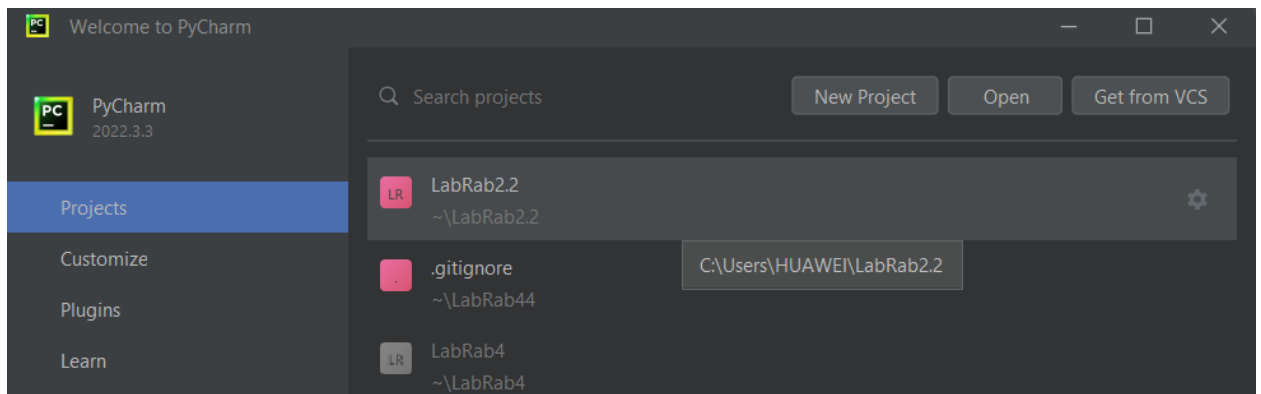


Рисунок 3. Проект в PyCharm

Задание 4.

Работа с примером 1.

Создал новый файл *primer1.py* в PyCharm и написал код соответствующий примеру 1. Реализацию примера осуществлял в новой ветке *feature/primer1*.

Условие примера:

Пример 1. Составить UML-диаграмму деятельности и программу с использованием конструкции ветвления и вычислить значение функции

$$y = \begin{cases} 2x^2 + \cos x, & x \leq 3.5, \\ x + 1, & 0 < x < 5, \\ \sin 2x - x^2, & x \geq 5. \end{cases}$$

```
C:\Users\HUAWEI\LabRab2.2>git flow feature start primer1
Switched to a new branch 'feature/primer1'

Summary of actions:
- A new branch 'feature/primer1' was created, based on 'develop'
- You are now on branch 'feature/primer1'

Now, start committing on your feature. When done, use:

git flow feature finish primer1
```

Рисунок 4. Создание ветки для примера 1

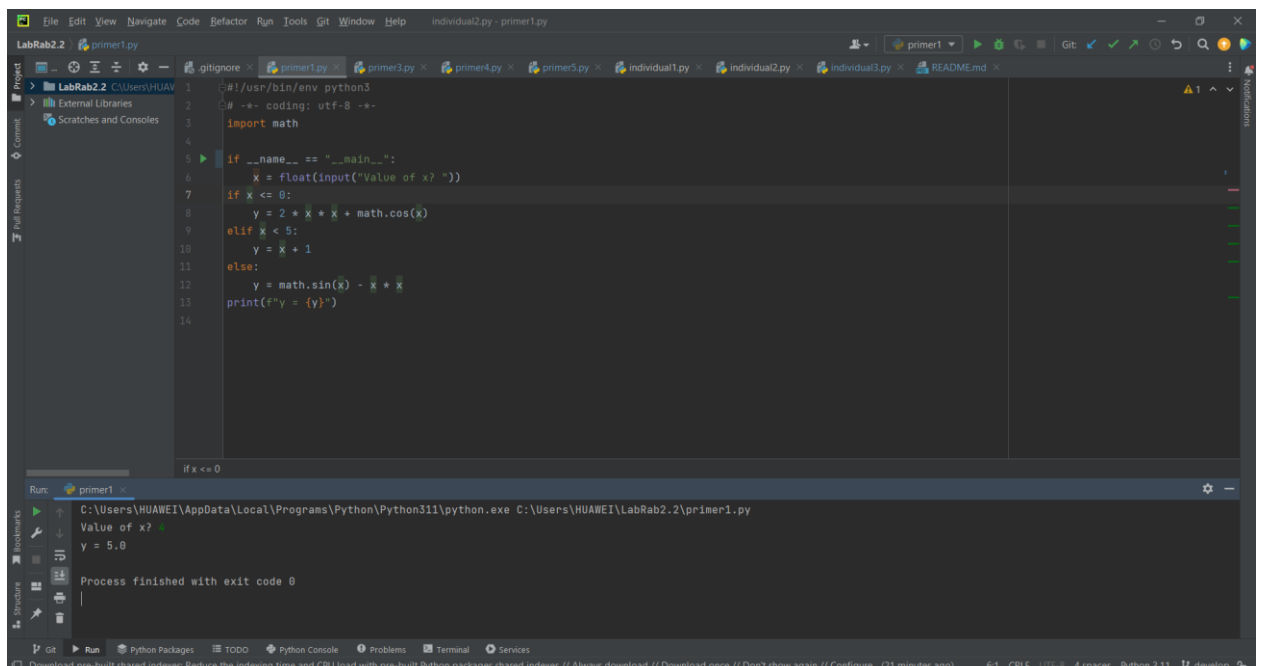


Рисунок 5. Код примера и его выполнение.

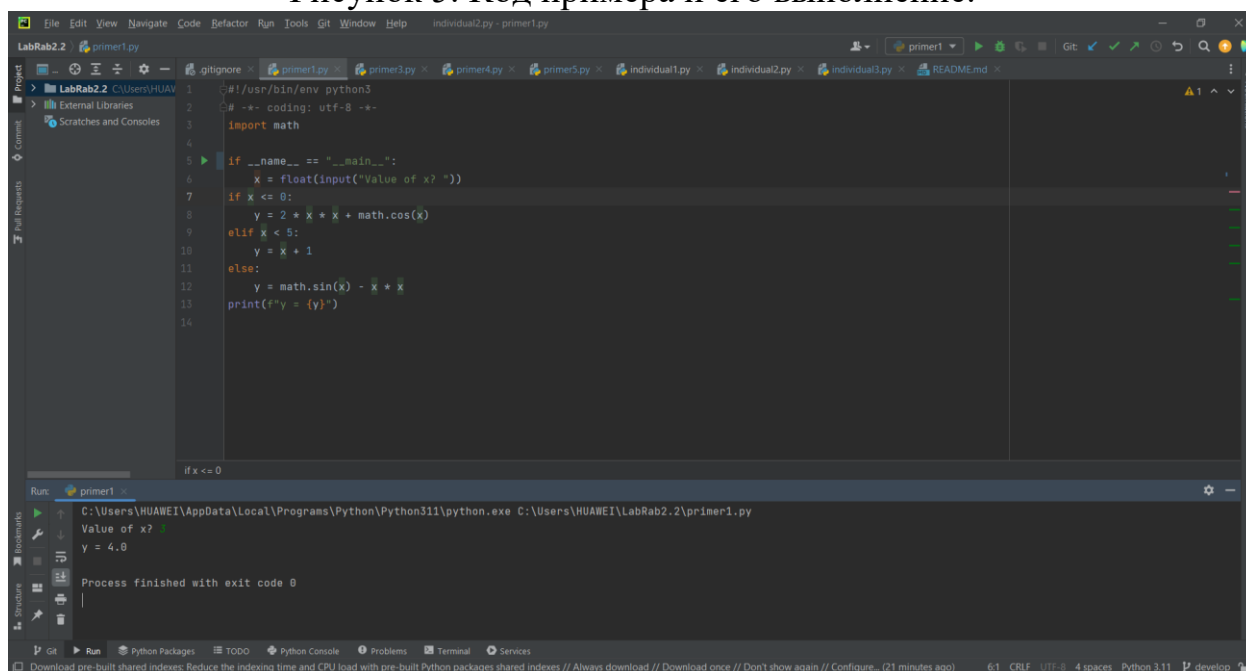


Рисунок 6. Код примера и его выполнение (с другими введенными с клавиатуры данными)

Так как использовалась модель ветвления *git-flow* далее со следующими примерами проделываю в *Git* следующие операции – добавление изменений, коммит, завершение (слияние ее с веткой *develop*) с веткой *feature/*(название ветки), создание ветки *release/*(название ветки) – для того чтобы внести некоторые изменения, если они необходимы, завершение с этой веткой и слияние с веткой *main*).

Задание 5.

Работа с примером 2.

Создал новый файл *primer2.py* в *PyCharm* и написал код соответствующий примеру 2. Реализацию примера осуществлял в новой ветке *feature/primer2*.

Условие примера:

Пример 2. Составить UML-диаграмму деятельности и программу для решения задачи: с клавиатуры вводится номер месяца от 1 до 12, необходимо для этого номера месяца вывести наименование времени года.

```
C:\Users\HUAWEI\LabRab2.2>git flow feature start primer2
Switched to a new branch 'feature/primer2'

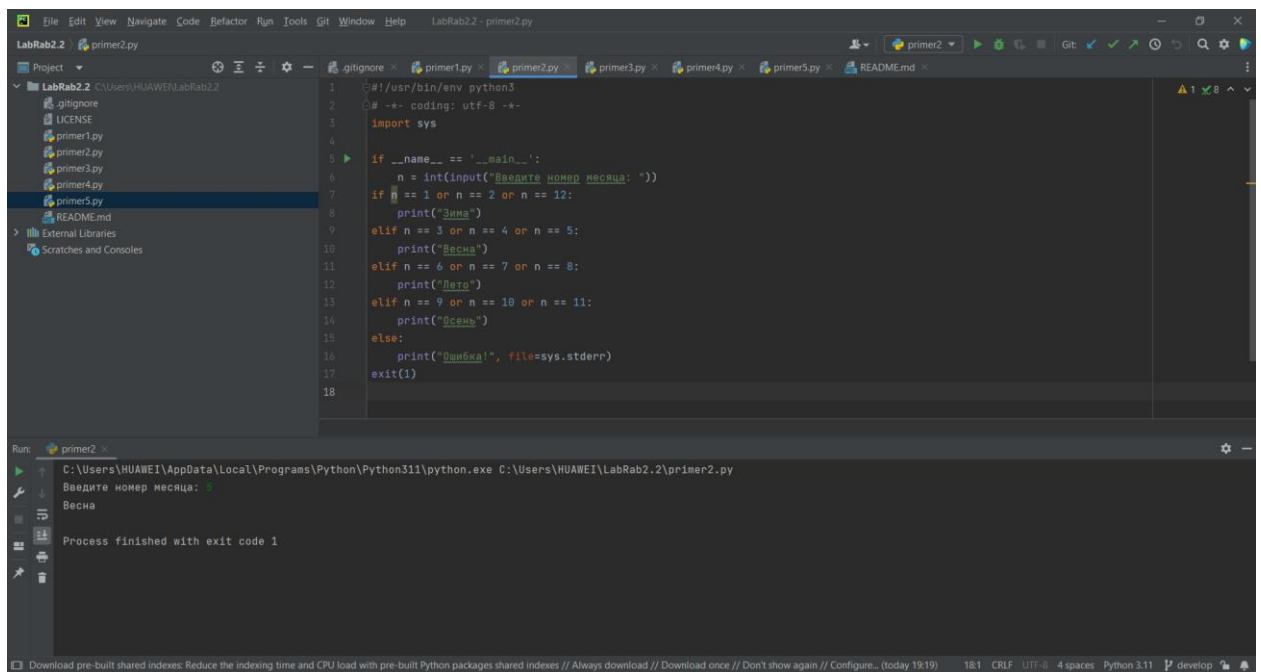
Summary of actions:
- A new branch 'feature/primer2' was created, based on 'develop'
- You are now on branch 'feature/primer2'

Now, start committing on your feature. When done, use:

    git flow feature finish primer2

C:\Users\HUAWEI\LabRab2.2>|
```

Рисунок 7. Создание ветки для примера 2



The screenshot shows an IDE window titled 'LabRab2.2 - primer2.py'. The left sidebar displays a project tree with files: .gitignore, LICENSE, primer1.py, primer2.py (selected), primer3.py, primer4.py, primer5.py, and README.md. The main editor area shows the code for primer2.py, which 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      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
```

Below the editor, the 'Run' panel shows the execution command: `C:\Users\HUAWEI\AppData\Local\Programs\Python\Python311\python.exe C:\Users\HUAWEI\LabRab2.2\primer2.py`. The output shows the prompt 'Введите номер месяца: ' followed by the input '5', and the result 'Весна'. The process finished with exit code 1.

Рисунок 8. Код примера и его выполнение

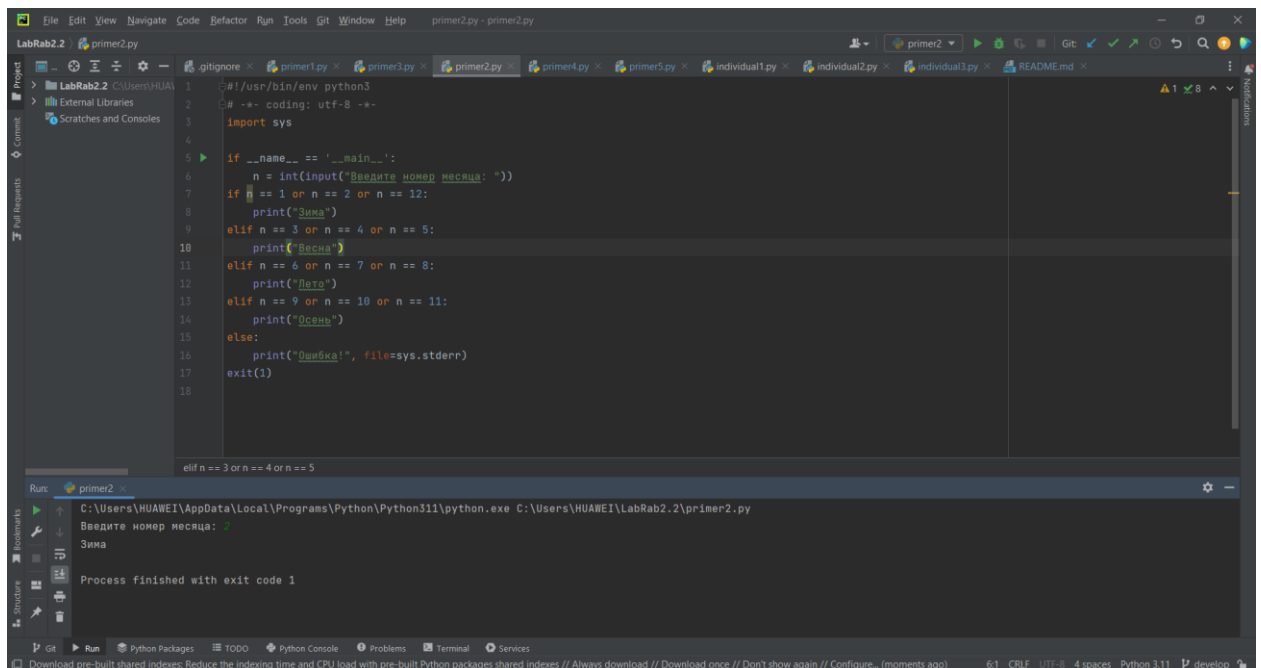


Рисунок 9. Код примера и его выполнение (с другими введенными с клавиатуры данными)

Так как использовалась модель ветвления git-flow далее со следующими примерами проделываю в Git следующие операции – добавление изменений, коммит, завершение (слияние ее с веткой develop) с веткой feature/(название ветки), создание ветки release/(название ветки) – для того чтобы внести некоторые изменения, если они необходимы, завершение с этой веткой и слияние с веткой main).

Задание 5.

Работа с примером 3.

Создал новый файл *primer3.py* в PyCharm и написал код соответствующий примеру 3. Реализацию примера осуществлял в новой ветке feature/primer3.

Условие примера:

Пример 3. Составить UML-диаграмму деятельности и написать программу, позволяющую вычислить конечную сумму:

$$S = \sum_{k=1}^n \frac{\ln kx}{k^2},$$

где n и k вводятся с клавиатуры.

```

C:\Users\HUAWEI\LabRab2.2>git flow feature start primer3
Switched to a new branch 'feature/primer3'

Summary of actions:
- A new branch 'feature/primer3' was created, based on 'develop'
- You are now on branch 'feature/primer3'

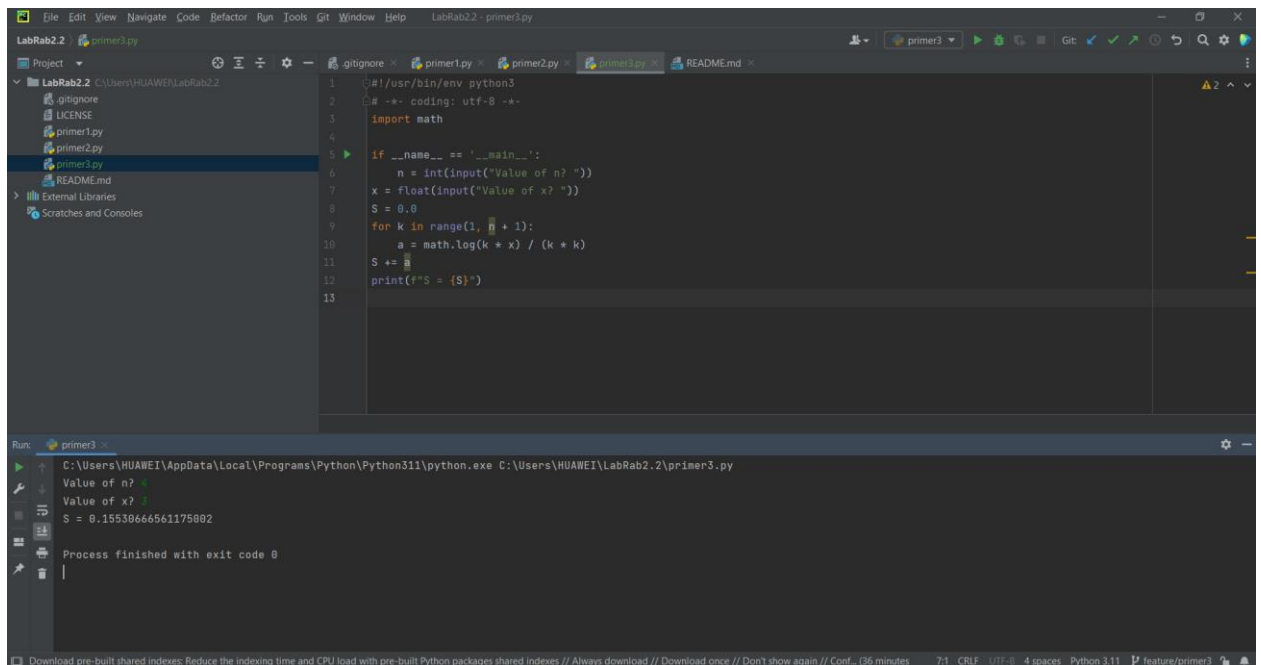
Now, start committing on your feature. When done, use:

    git flow feature finish primer3

C:\Users\HUAWEI\LabRab2.2>git status
On branch feature/primer3
nothing to commit, working tree clean

```

Рисунок 10. Создание ветки для примера 3



The screenshot shows an IDE with a project named 'LabRab2.2'. The file explorer on the left shows the project structure, including 'primer3.py'. The main editor displays the code for 'primer3.py', which is a Python script that calculates the harmonic series sum for a given n. 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}")
13

```

The Run window at the bottom shows the execution of the script. The output is as follows:

```

C:\Users\HUAWEI\AppData\Local\Programs\Python\Python311\python.exe C:\Users\HUAWEI\LabRab2.2\primer3.py
Value of n? 10
Value of x? 1.5
S = 0.15530666561175002
Process finished with exit code 0

```

Рисунок 11. Код примера и его выполнение

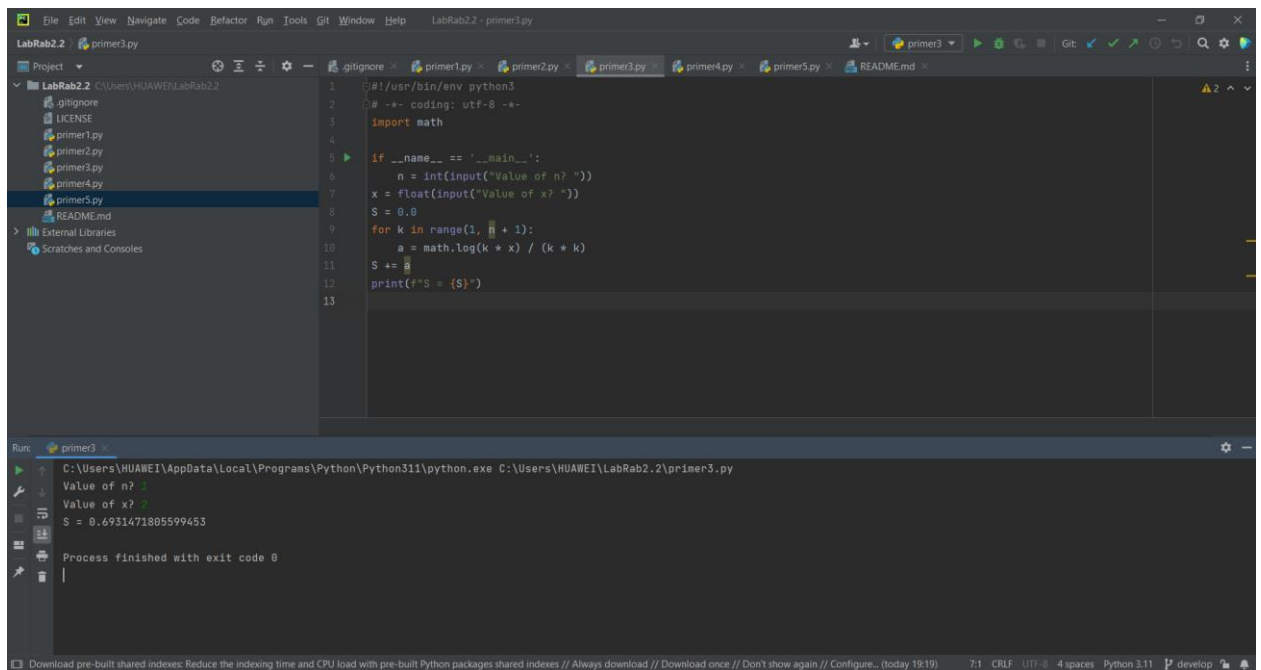


Рисунок 12. Код примера и его выполнение (с другими введенными с клавиатуры данными)

Так как использовалась модель ветвления *git-flow* далее со следующими примерами проделываю в *Git* следующие операции – добавление изменений, коммит, завершение (слияние ее с веткой *develop*) с веткой *feature/*(название ветки), создание ветки *release/*(название ветки) – для того чтобы внести некоторые изменения, если они необходимы, завершение с этой веткой и слияние с веткой *main*).

Задание 6.

Работа с примером 4.

Создал новый файл *primer4.py* в *PyCharm* и написал код соответствующий примеру 4. Реализацию примера осуществлял в новой ветке *feature/primer4*.

Условие примера:

Пример 4. Найти значение квадратного корня $x = \sqrt{a}$ из положительного числа a вводимого с клавиатуры, с некоторой заданной точностью ε с помощью рекуррентного соотношения:

$$x_{n+1} = \frac{1}{2} \cdot \left(x_n + \frac{a}{x_n} \right). \quad (3)$$

В качестве начального значения примем $x_0 = 1$. Цикл должен выполняться до тех пор, пока не будет выполнено условие $|x_{n+1} - x_n| \leq \varepsilon$. Сравните со значением квадратного корня, полученным с использованием функций стандартной библиотеки. Значение $\varepsilon = 10^{-10}$.

```
C:\Users\HUAWEI\LabRab2.2>git flow feature start primer4
Switched to a new branch 'feature/primer4'

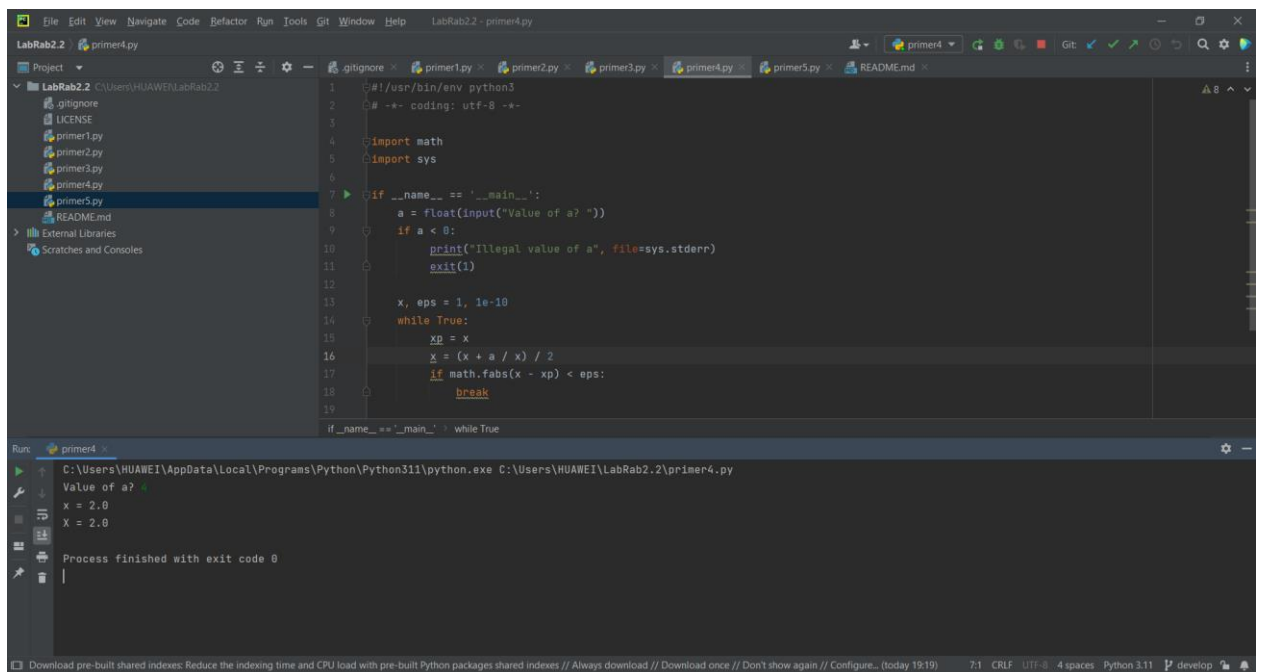
Summary of actions:
- A new branch 'feature/primer4' was created, based on 'develop'
- You are now on branch 'feature/primer4'

Now, start committing on your feature. When done, use:

    git flow feature finish primer4

C:\Users\HUAWEI\LabRab2.2>|
```

Рисунок 13. Создание ветки для примера 4



The screenshot shows an IDE window titled 'LabRab2.2 - primer4.py'. The left sidebar displays the project structure with files like 'gitignore', 'LICENSE', 'primer1.py', 'primer2.py', 'primer3.py', 'primer4.py', 'primer5.py', 'README.md', 'External Libraries', and 'Scratches and Consoles'. The main editor area shows the code for 'primer4.py':

```
1  C:\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 if __name__ == '__main__': while True
```

The bottom panel shows the 'Run' output for 'primer4.py':

```
C:\Users\HUAWEI\AppData\Local\Programs\Python\Python311\python.exe C:\Users\HUAWEI\LabRab2.2\primer4.py
Value of a? 2.0
x = 2.0
X = 2.0
Process finished with exit code 0
```

Рисунок 14. Код примера и его выполнение

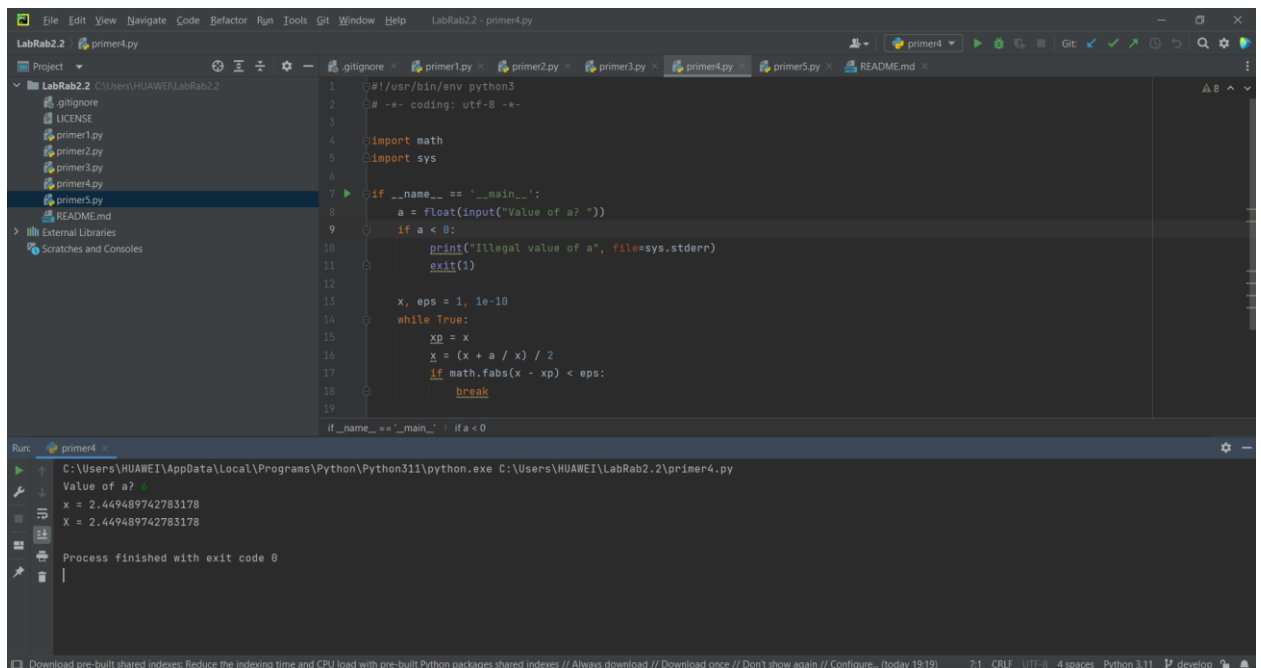


Рисунок 15. Код примера и его выполнение (с другими введенными с клавиатуры данными)

Задание 7.

Работа с примером 5.

Создал новый файл *primer5.py* в PyCharm и написал код соответствующий примеру 5. Реализацию примера осуществлял в новой ветке feature/primer5.

Условие примера:

Пример 5. Вычислить значение специальной (интегральной показательной) функции

$$\text{Ei}(x) = \int_{-\infty}^x \frac{\exp t}{t} dt = \gamma + \ln x + \sum_{k=1}^{\infty} \frac{x^k}{k \cdot k!}, \quad (4)$$

где $\gamma = 0.5772156649 \dots$ - постоянная Эйлера, по ее разложению в ряд с точностью $\varepsilon = 10^{-10}$, аргумент x вводится с клавиатуры.

```

C:\Users\HUAWEI\LabRab2.2>git flow feature start primer5
Switched to a new branch 'feature/primer5'

Summary of actions:
- A new branch 'feature/primer5' was created, based on 'develop'
- You are now on branch 'feature/primer5'

Now, start committing on your feature. When done, use:

    git flow feature finish primer5

C:\Users\HUAWEI\LabRab2.2>

```

Рисунок 16. Создание ветки для примера 4

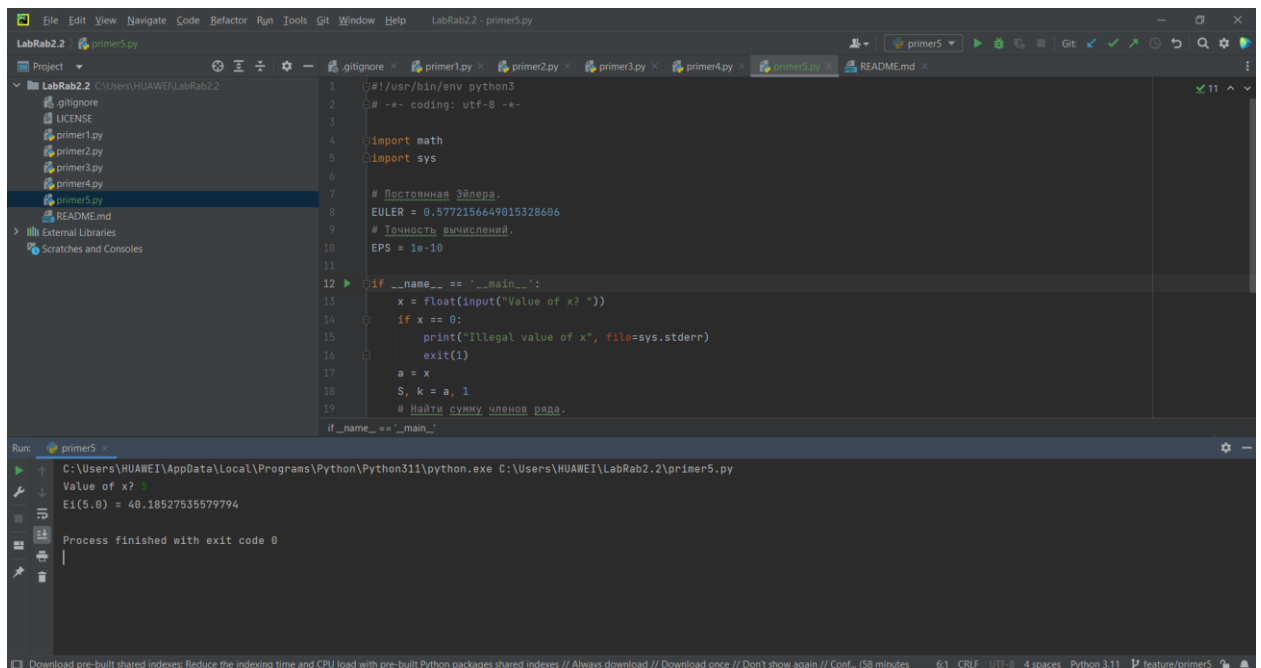


Рисунок 17. Код примера и его выполнение

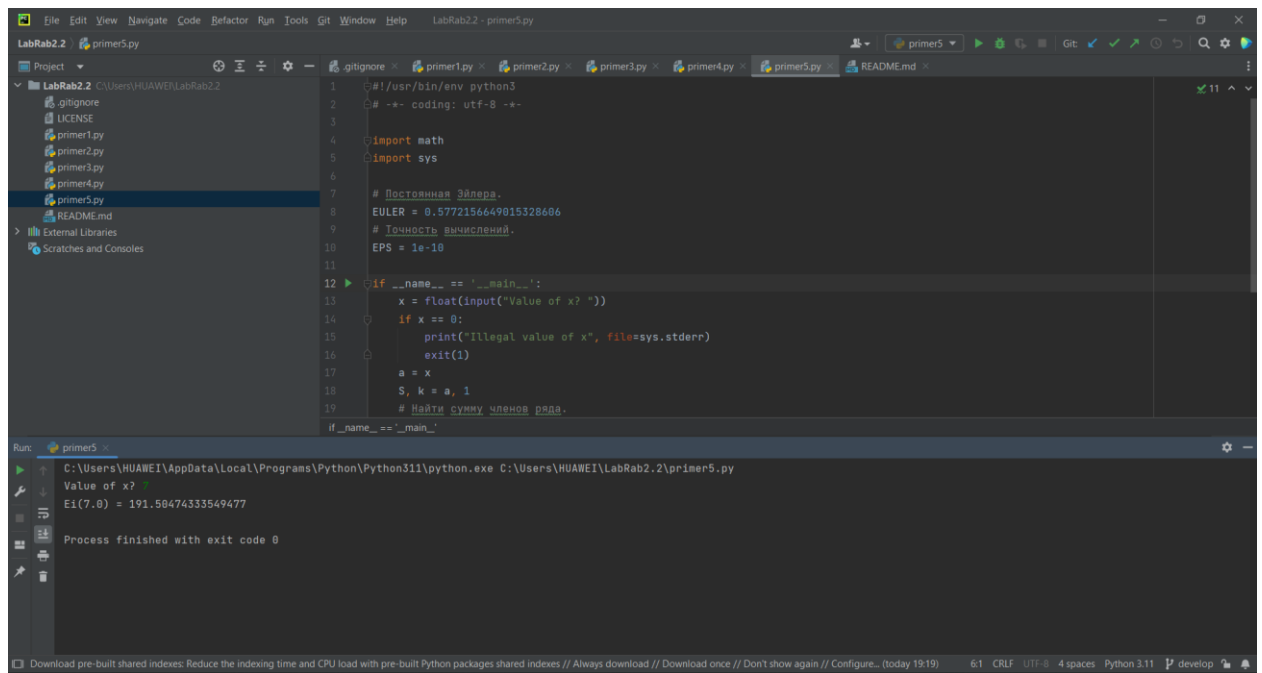
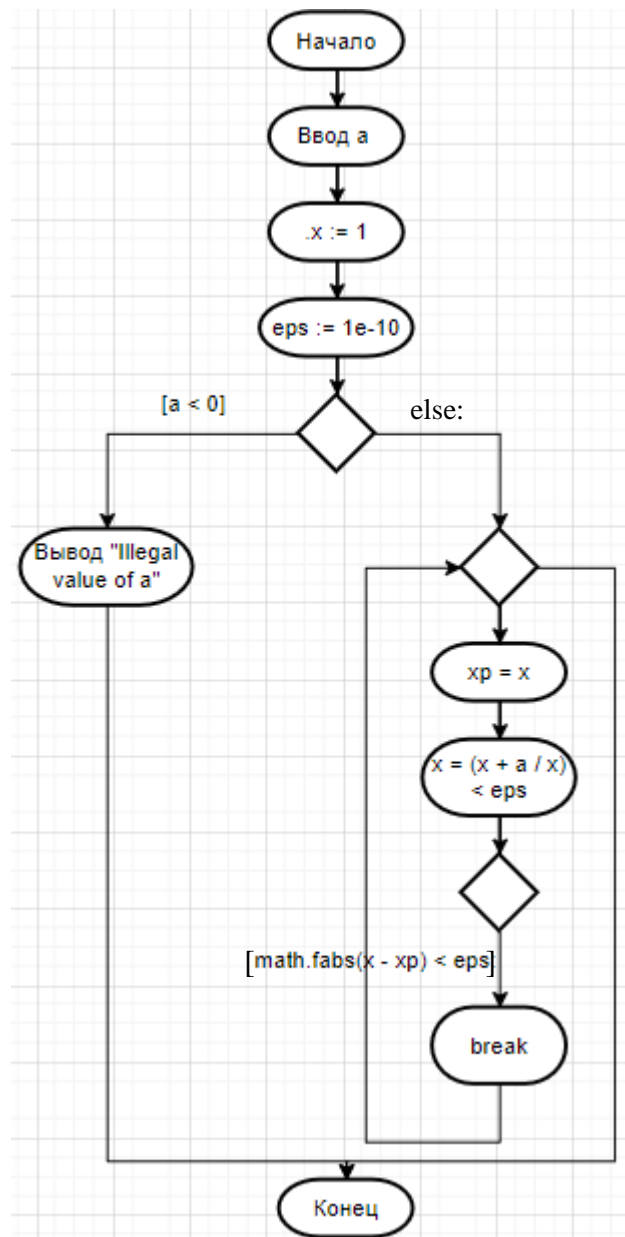


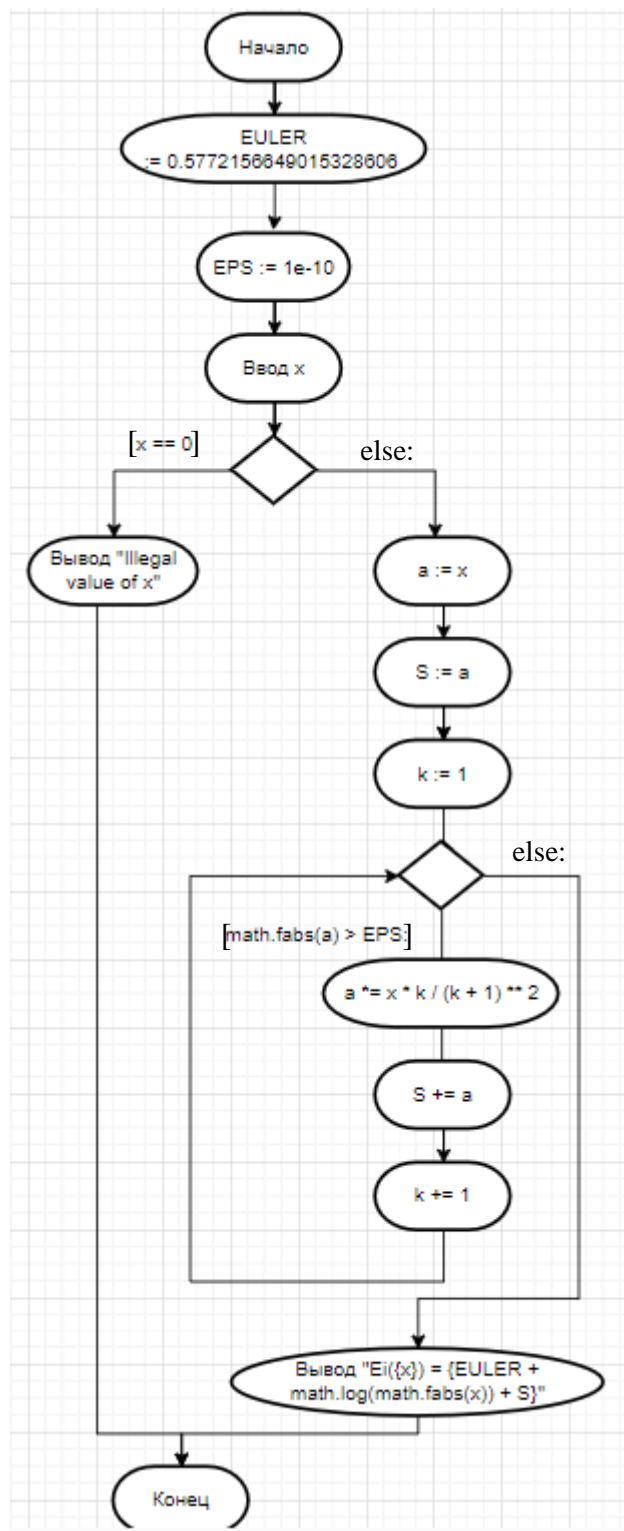
Рисунок 18. Код примера и его выполнение (с другими введенными с клавиатуры данными)

Задание 8.

UML-диаграмма деятельности для примера 4.



UML-диаграмма для примера 5.



Задание 11.

Индивидуальное задание 1.

Вариант 3. (т.к. по списку в группе 16-ый, а всего заданий 13)

Условие задания:

3. Дано число m ($1 \leq m \leq 7$). Вывести на экран название дня недели, который соответствует этому номеру.

Выполнял задание также с моделью git-flow.

Для этого создал новую ветку `feature/individual1` и в ней создал новый файл `individual1.py` и написал в нем код программы.

```
C:\Users\HUAWEI\LabRab2.2>git flow feature start individual1
Switched to a new branch 'feature/individual1'

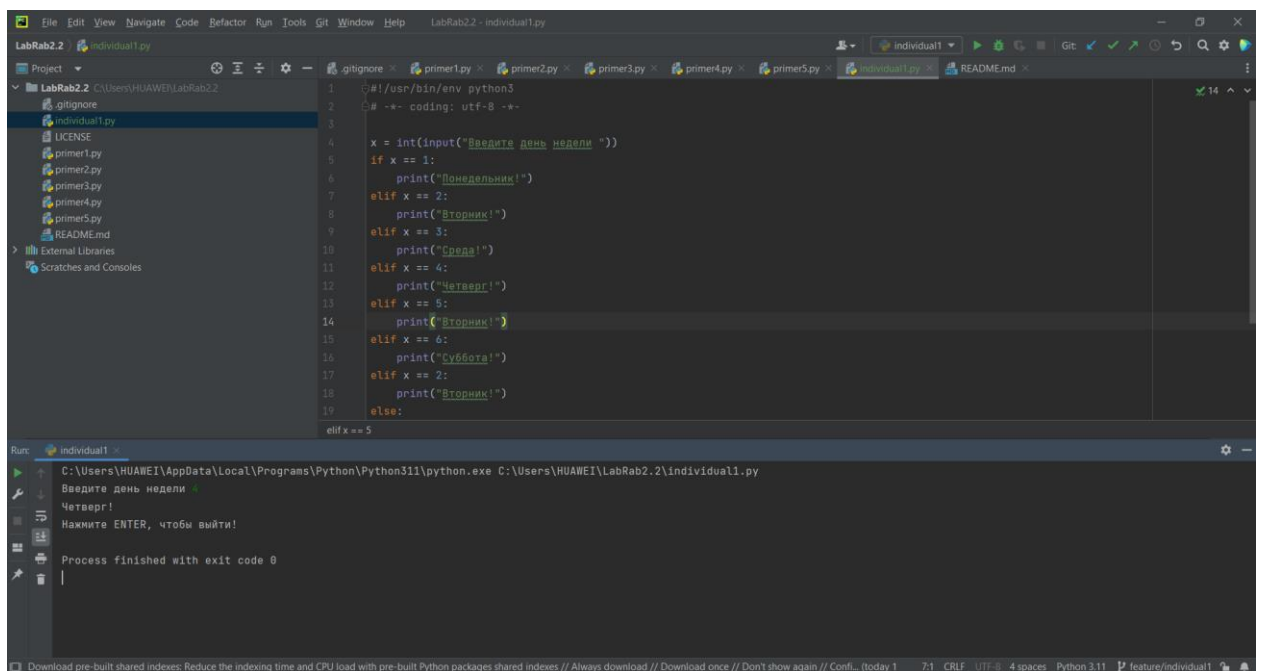
Summary of actions:
- A new branch 'feature/individual1' was created, based on 'develop'
- You are now on branch 'feature/individual1'

Now, start committing on your feature. When done, use:

    git flow feature finish individual1

C:\Users\HUAWEI\LabRab2.2>|
```

Рисунок 19. Новая ветка для индивидуального задания 1



The screenshot shows a code editor with a file named `individual1.py`. The code is a Python script that prompts the user to enter a day of the week (1-7) and prints the corresponding day name. The code is as follows:

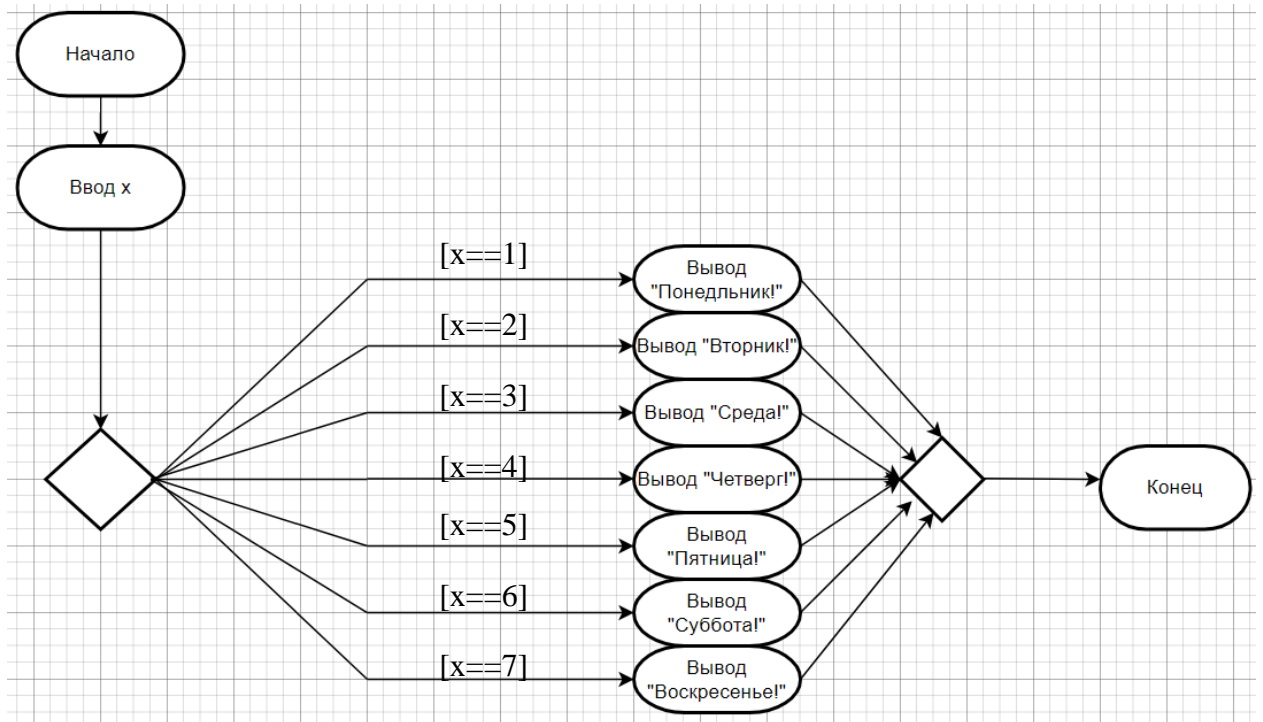
```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 x = int(input("Введите день недели "))
5 if x == 1:
6     print("Понедельник!")
7 elif x == 2:
8     print("Вторник!")
9 elif x == 3:
10    print("Среда!")
11 elif x == 4:
12    print("Четверг!")
13 elif x == 5:
14    print("Пятница!")
15 elif x == 6:
16    print("Суббота!")
17 elif x == 7:
18    print("Воскресенье!")
19 else:
20    print("Неверный ввод!")
```

The output of the program is shown in the console window below the code editor:

```
Введите день недели 4
Четверг!
Нажмите ENTER, чтобы выйти!
Process finished with exit code 0
```

Рисунок 20. Код задания и его выполнение

UML-диаграмма для индивидуального задания 1.



Задание 12.

Индивидуальное задание 2.

Вариант 16. (т.к. по списку в группе 16-ый, а всего заданий 23)

Условие задания:

16. Даны три действительных числа. Составить программу, выбирающую из них те, которые принадлежат интервалу (0, 1).

```

C:\Users\HUAWEI\LabRab2.2>git flow feature start individual2
Switched to a new branch 'feature/individual2'

Summary of actions:
- A new branch 'feature/individual2' was created, based on 'develop'
- You are now on branch 'feature/individual2'

Now, start committing on your feature. When done, use:

git flow feature finish individual2
  
```

Рисунок 21. Новая ветка для индивидуального задания 2

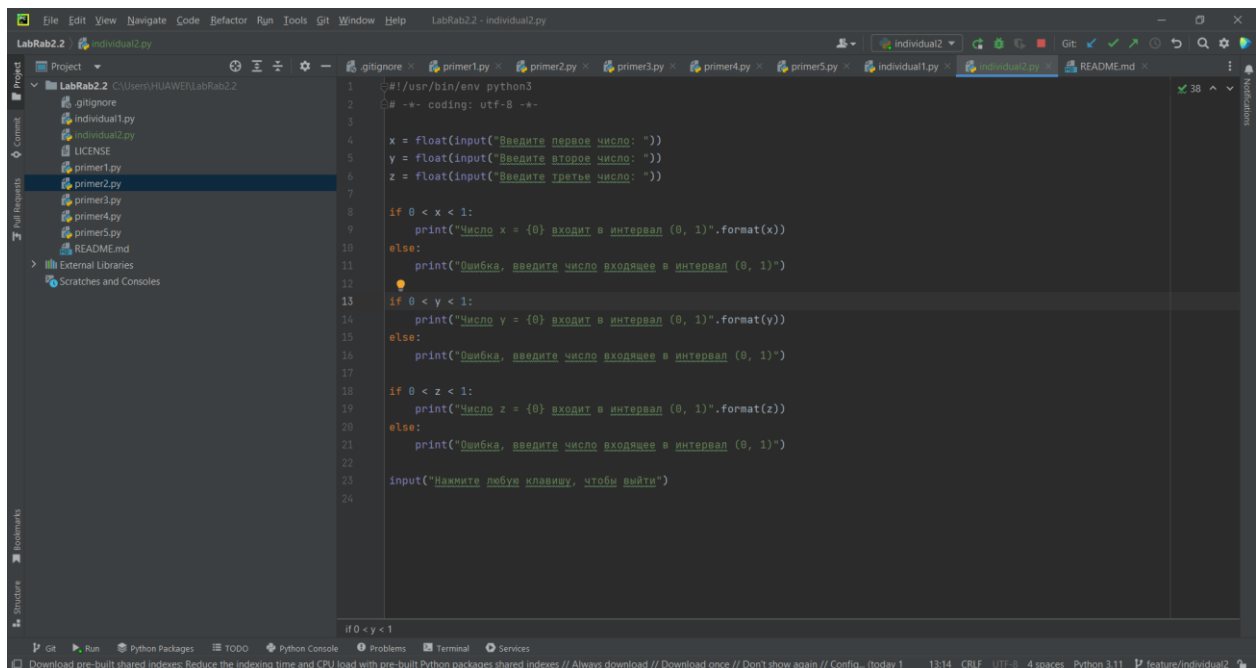


Рисунок 22. Программа для задания

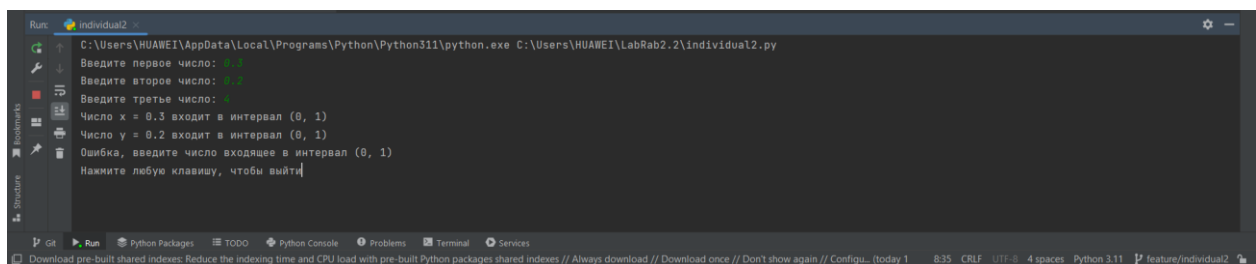
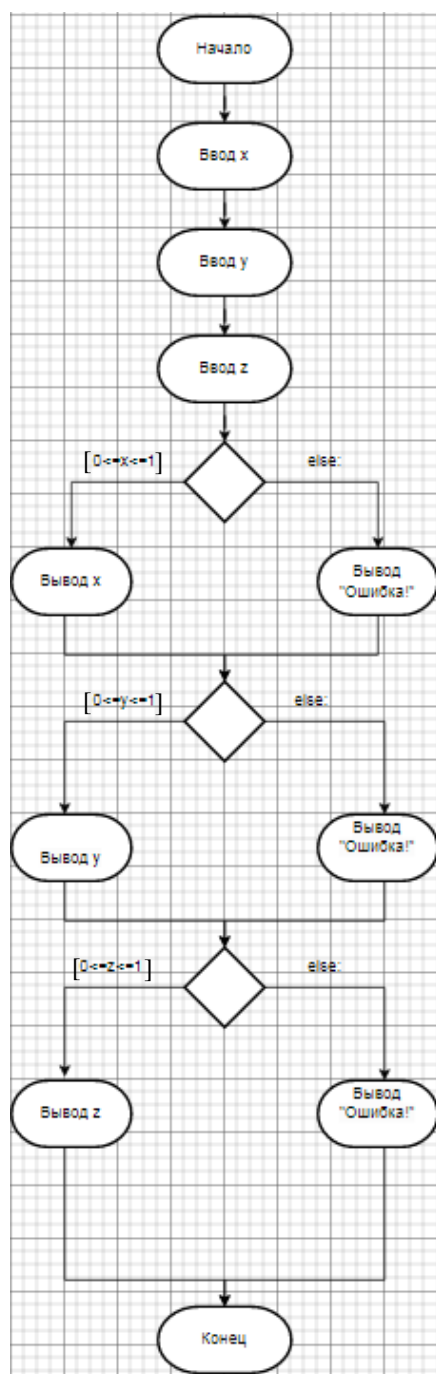


Рисунок 23. Выполнение задания

UML-диаграмма для индивидуального задания 2.



Задание 13.

Индивидуальное задание 3.

Вариант 16. (т.к. по списку в группе 16-ый, а всего заданий 22)

Условие задания:

16. Ученик выучил в первый день 5 английских слов. В каждый следующий день он выучивал на 2 слова больше, чем в предыдущий. Сколько английских слов выучит ученик в 10-ый день занятий.

```
C:\Users\HUAWEI\LabRab2.2>git flow feature start individual3
Switched to a new branch 'feature/individual3'

Summary of actions:
- A new branch 'feature/individual3' was created, based on 'develop'
- You are now on branch 'feature/individual3'

Now, start committing on your feature. When done, use:

    git flow feature finish individual3

C:\Users\HUAWEI\LabRab2.2>|
```

Рисунок 24. Новая ветка для индивидуального задания 3

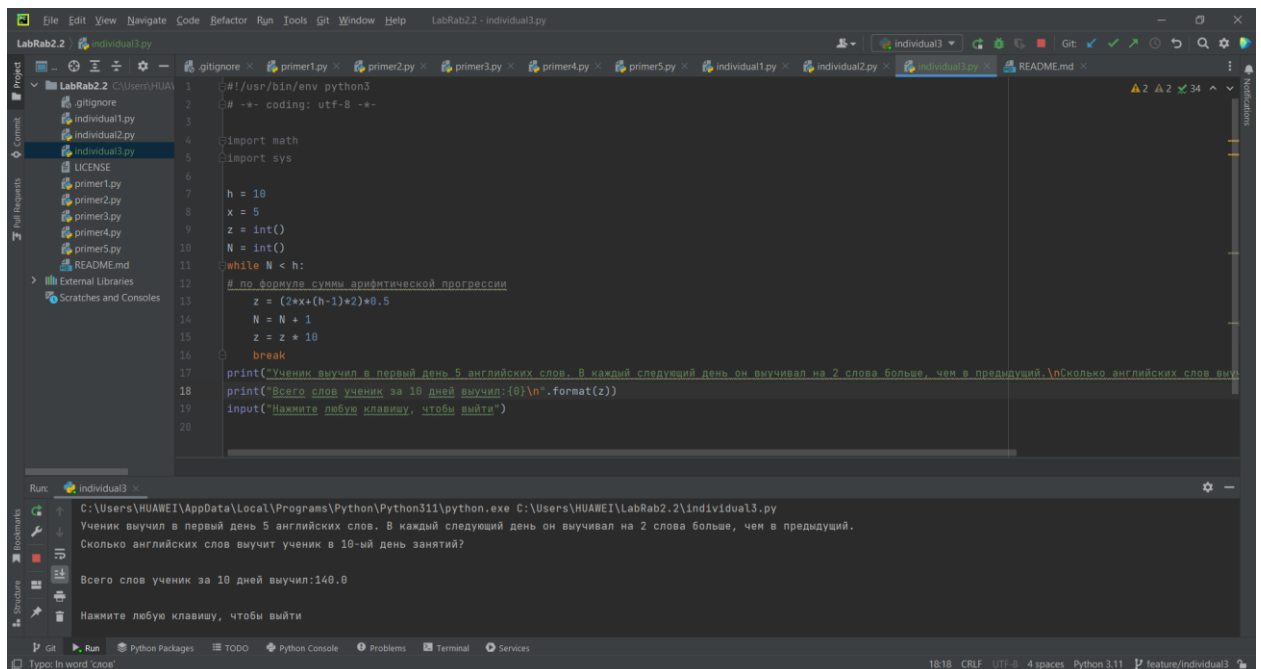
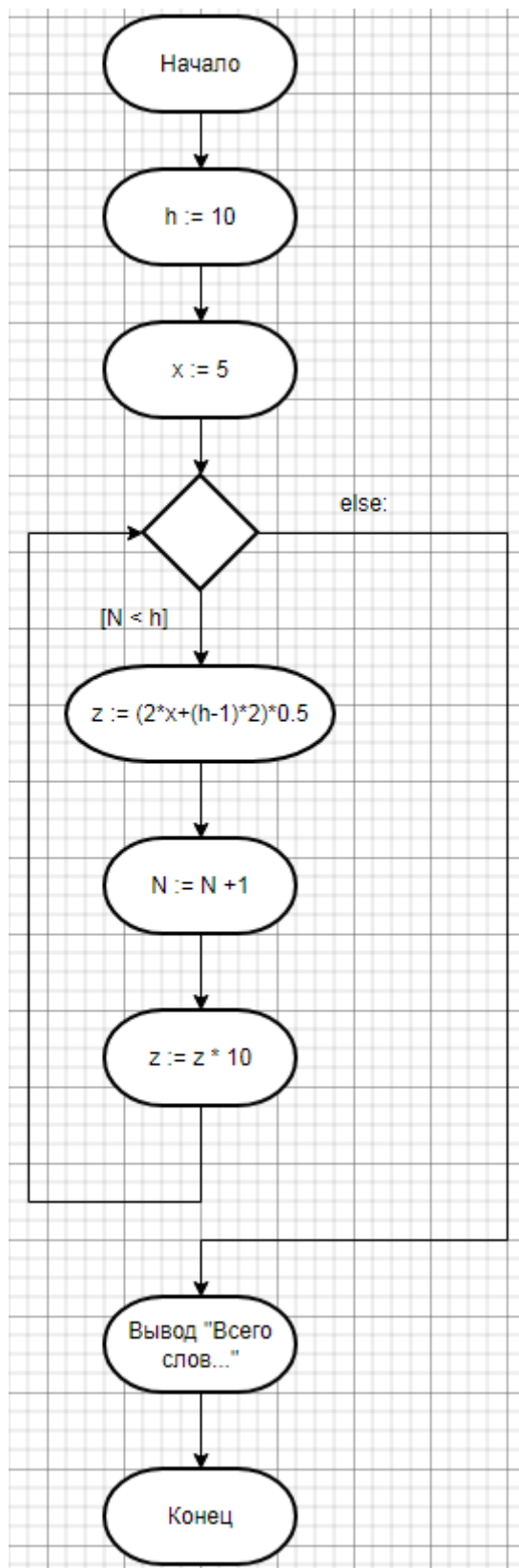


Рисунок 25. Программа и выполнение индивидуального задания 3.



Ссылка на репозиторий: <https://github.com/DaniiGit23/LabRab2.2.git>

Ответы на контрольные вопросы:

1) Для чего нужны диаграммы деятельности UML?

Диаграммы деятельности. Унифицированный язык моделирования (UML) является стандартным инструментом для создания «чертежей» программного обеспечения. С помощью UML можно визуализировать, специфицировать, конструировать и документировать артефакты программных систем. UML пригоден для моделирования любых систем: от информационных систем масштаба предприятия до распределенных Web-приложений и даже встроенных систем реального времени. Это очень выразительный язык, позволяющий рассмотреть систему со всех точек зрения, имеющих отношение к ее разработке и последующему развертыванию. Несмотря на обилие выразительных возможностей, этот язык прост для понимания и использования.

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 изображаются прямоугольниками с закругленными краями. Внутри такого символа можно записывать произвольное выражение.

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.

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

Переходы. 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. Поток управления должен где-то начинаться и заканчиваться (разумеется, если это не бесконечный поток, у которого есть начало, но нет конца). Как показано на рисунке, вы можете задать как начальное состояние (закрашенный кружок), так и конечное (закрашенный кружок внутри окружности).

Ветвление. Простые последовательные переходы встречаются наиболее часто, но их одних недостаточно для моделирования любого потока управления. Как и в блок-схеме, вы можете включить в модель ветвление, которое описывает различные пути выполнения в зависимости от значения некоторого булевского выражения. Точка ветвления представляется ромбом. В точку ветвления может входить ровно один переход, а выходить – два или более. Для каждого исходящего перехода задается булевское выражение, которое вычисляется только один раз при входе в точку ветвления. Ни для каких двух исходящих переходов эти сторожевые условия не должны одновременно принимать значение «истина», иначе поток управления окажется неоднозначным. Но эти условия должны покрывать все возможные варианты, иначе поток остановится.

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. 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?

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 двух возможных шагов.

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

Условный оператор состоит из трёх слов IF ELSE THEN. Здесь <условие> просто помещает значение на вершину стека, IF анализирует флаг, и если: он не равен нулю, то выполняются выражения до ELSE или THEN ; если он равен нулю, то выполняется выражения между ELSE и THEN .

Оператор цикла while выполняет указанный набор инструкций до тех пор, пока условие цикла истинно. Истинность условия определяется также как и в операторе if.

Оператор break предназначен для досрочного прерывания работы цикла while.

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

Оператор for выполняет указанный набор инструкций заданное количество раз, которое определяется количеством элементов в наборе.

Функция range возвращает неизменяемую последовательность чисел в виде объекта range.

7) Какие операторы сравнения используются в Python?

В Python есть шесть операций сравнения. Все они имеют одинаковый приоритет, который выше, чем у логических операций.

Разрешенные операции сравнения:

- $x < y$ – строго x меньше y ,
- $x \leq y$ – x меньше или равно y ,
- $x > y$ – строго x больше y ,
- $x \geq y$ – x больше или равно y ,
- $x == y$ – x равно y ,
- $x != y$ – x не равно y .

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: $<$ - меньше, чем... Например, простыми отношениями являются следующие: $x > 10$; $k \leq \text{sqr}(c) + \text{abs}(a+b)$; $9 \neq 11$; 'мама' \neq 'папа'.

9) Что такое составное условие? Приведите примеры.

Составные условия — это условия, состоящие из двух или более простых условий, соединенных с помощью логических операций: **and** , **or** , **not**. Простые условия при этом заключаются в скобки.

Примеры составных условий:

$(a < 5) \text{ and } (b > 8)$

$(x \geq 0) \text{ or } (x < -3)$

$\text{not } (a = 0) \text{ or } (b = 0)$

10) Какие логические операторы допускаются при составлении сложных условий?

Используются специальные операторы, объединяющие два и более простых логических выражения. Широко используются два оператора – так называемые логические **И** (**and**) и **ИЛИ** (**or**).

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

Да. Такой оператор ветвления называется вложенным. В этом случае важно не перепутать какая ветвь кода к какому оператору относится. Поэтому

рекомендуется соблюдать отступы в исходном коде программы, чтобы не запутаться. `if <условие> then if< условие> then< оператор 1>;` Вложенный условный оператор.

12) Какой алгоритм является алгоритмом циклической структуры?

Например, перевод текста с иностранного языка (прочитать первое предложение, перевести, записать и т.д.)

Построение графика функции по точкам (взять первый аргумент, вычислить значение функции, построить точку и т.д.)

13) Типы циклов в языке Python.

Цикл — это блок, элементы которого повторяют своё действие заданное количество раз. Бывают. Практически все современные алгоритмы содержат в себе циклы. В языке Python существуют следующие типы циклов: 1. `While()` — Цикл с предусловием 2. `for()` — Цикл с чётким количеством проходов.

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

Функция `range` является одной из встроенных функций, доступных в Python. Он 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 ()` принимает один обязательный и два необязательных параметра. Это работает по-разному с различными комбинациями аргументов.

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

```
range(15, 0, -2)
```

16) Могут ли циклы быть вложенными?

Чисто технически во вложенных циклах нет ничего особенного. Их можно вкладывать внутрь любого блока и друг в друга сколько угодно раз. Но прямой связи между внешним и вложенным циклами нет. Внутренний цикл может использовать результаты внешнего, а может и работать по своей собственной логике независимо. Вложенные циклы коварны. Их наличие

может резко увеличить сложность кода, так как появляется множество постоянно изменяющихся переменных.

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. О программе, вошедшей в бесконечный цикл, иногда говорят, что она зацклилась.

18) Для чего нужен оператор break?

Выражение break заставляет программу выйти из цикла.

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

Выражение Continue. Выражение continue дает возможность пропустить часть цикла, где активируется внешнее условие, но при этом выполнить остальную часть цикла. При этом прерывается текущая итерация цикла, но программа возвращается к началу цикла. Выражение continue размещается в блоке кода под выражением цикла, обычно после условного выражения if.

20) Для чего нужны стандартные потоки stdout и stderr?

STDOUT - стандартный вывод - то, куда выводят данные команды echo/print, консоль или сокет, отправляющий данные браузеру. STDERR – поток сообщений об ошибках.

21) Каково назначение функции exit?

Функция exit() вызывает немедленное нормальное завершение программы.

Вывод: В ходе выполнения лабораторной работы были приобретены навыки программирования разветвляющихся алгоритмов и алгоритмов циклической структуры. Также изучены операторы языка Python.3 if, while, for, break, continue, позволяющих реализовывать разветвляющиеся алгоритмы и алгоритмы циклической структуры.

