

**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: «Основы кроссплатформенного программирования»

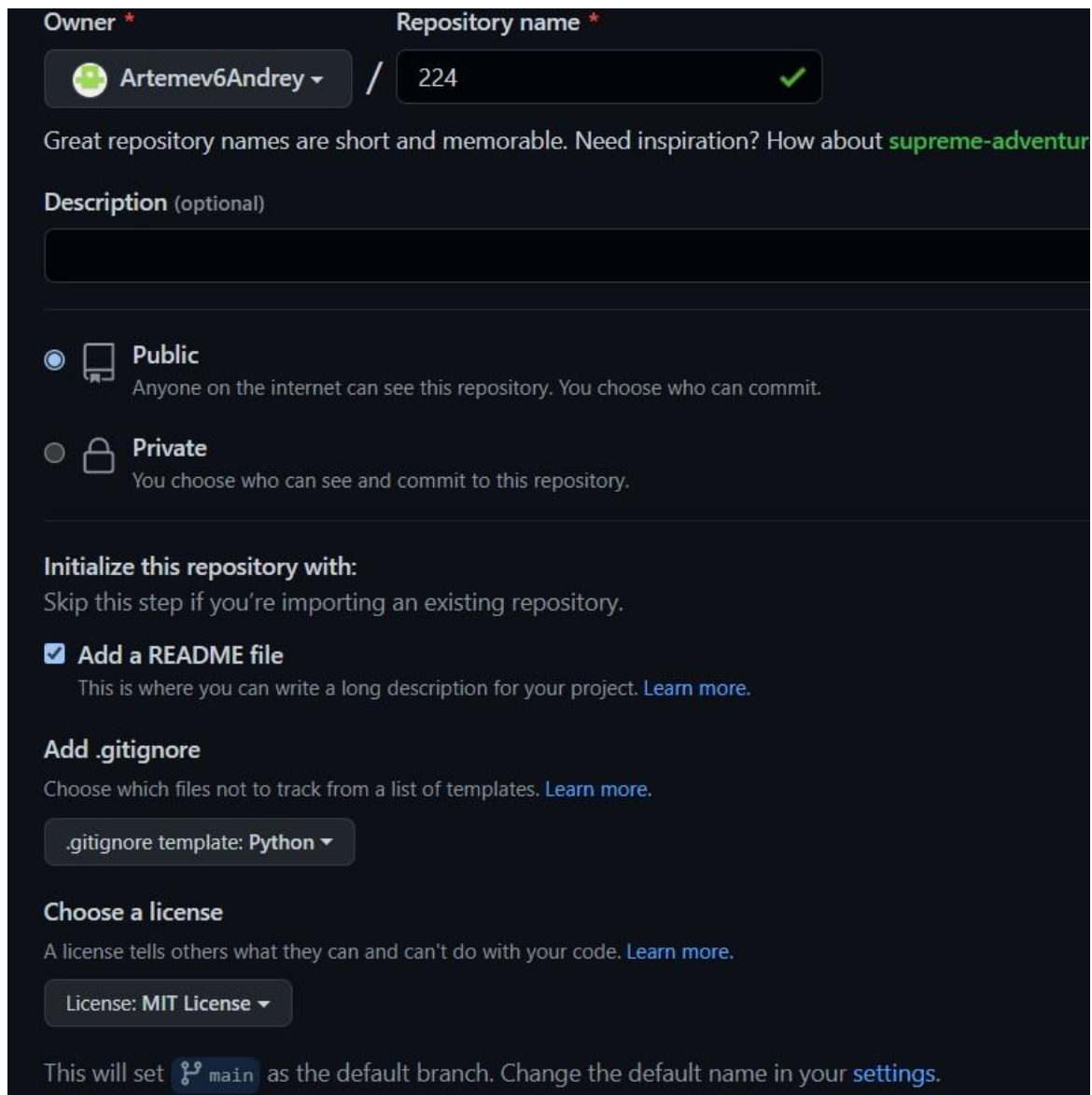
Тема: «Условные операторы и циклы в языке Python»

Выполнил: студент 1 курса
группы ИВТ-б-о-21-1
Артемьев Андрей Витальевич

Ставрополь 2022

Выполнение работы:

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



The screenshot shows the GitHub repository creation page. At the top, the 'Owner' is set to 'Artemev6Andrey' and the 'Repository name' is '224', which is marked as valid with a green checkmark. Below this, there is a hint about repository names. The 'Description' field is empty. Under the 'Visibility' section, 'Public' is selected, indicating that anyone on the internet can see the repository. The 'Initialize this repository with:' section has 'Add a README file' checked. The 'Add .gitignore' section shows the 'Python' template selected. The 'Choose a license' section has 'MIT License' selected. At the bottom, it states that the default branch will be set to 'main'.

Owner * / Repository name *

Artemev6Andrey / 224 ✓

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

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:
Skip this step if you're importing an existing repository.

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

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

.gitignore template: Python ▼

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

License: MIT License ▼

This will set `main` as the default branch. Change the default name in your [settings](#).

Рисунок 1. Создание репозитория

```

273 lines (218 sloc) | 5.31 KB

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
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

```

Рисунок 1.2 Добавление правил в .gitignore

```

MINGW64; c:/Users/aa715/Desktop/224
aa715@DESKTOP-70QJC9F MINGW64 ~/Desktop/224 (master)
$ git clone https://github.com/Artemev6Andrey/224.git
Cloning into '224'...
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.

aa715@DESKTOP-70QJC9F MINGW64 ~/Desktop/224 (master)
$ git flow init

No branches exist yet. Base branches must be created now.
Branch name for production releases: [master] Branch name for "next release" develop: [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/aa715/Desktop/.git/hooks] |

```

Рисунок 1.3 Клонирование и организация репозитория согласно модели ветвления git-flow

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

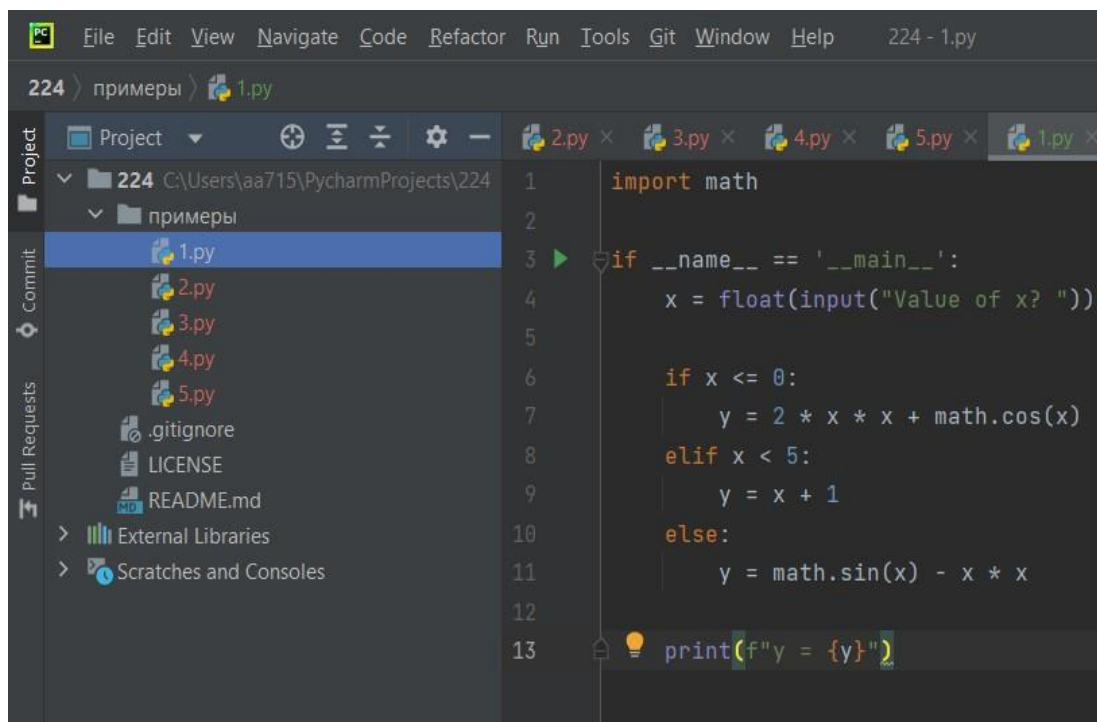


Рисунок 2.1 Примеры в проекте

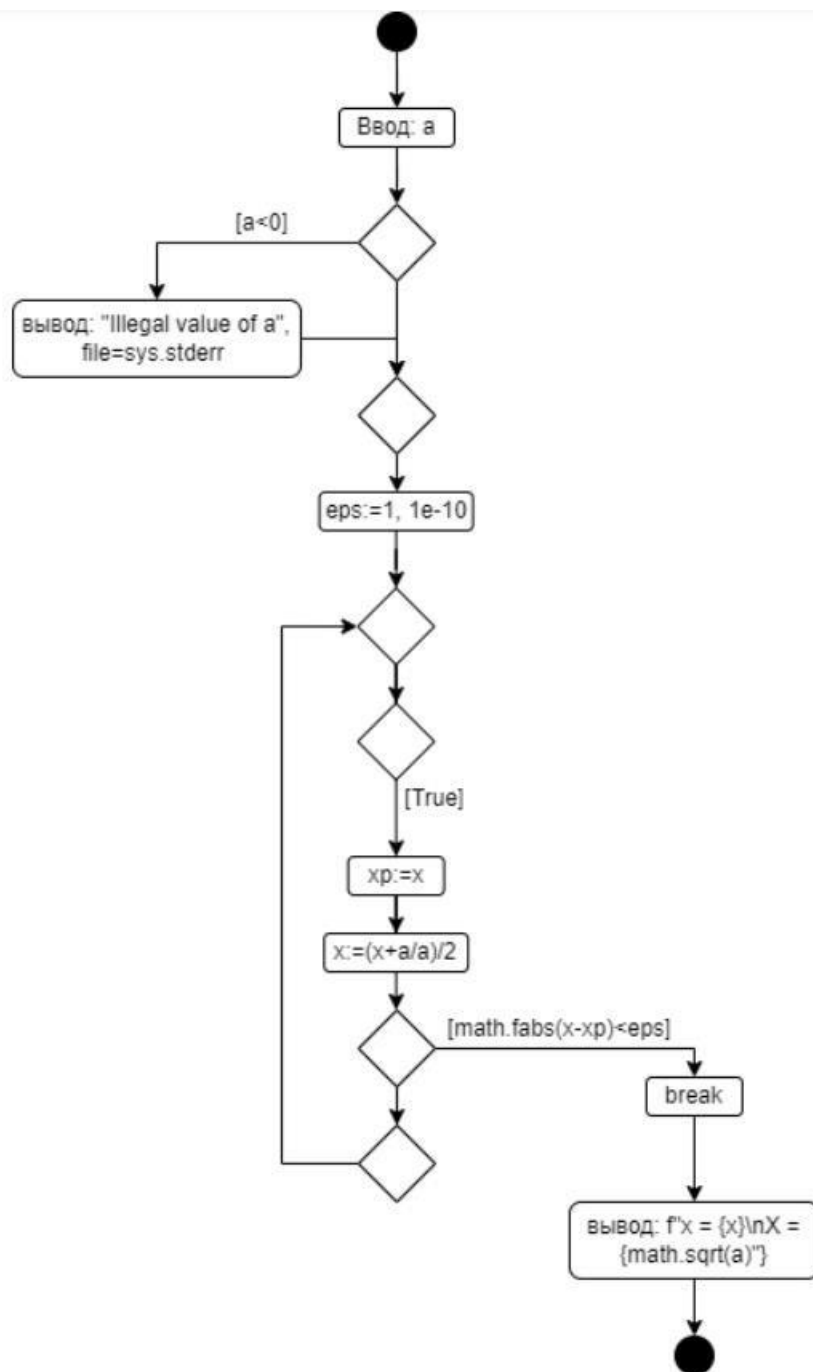


Рисунок 2.2 UML-диаграмма программы 4 примера

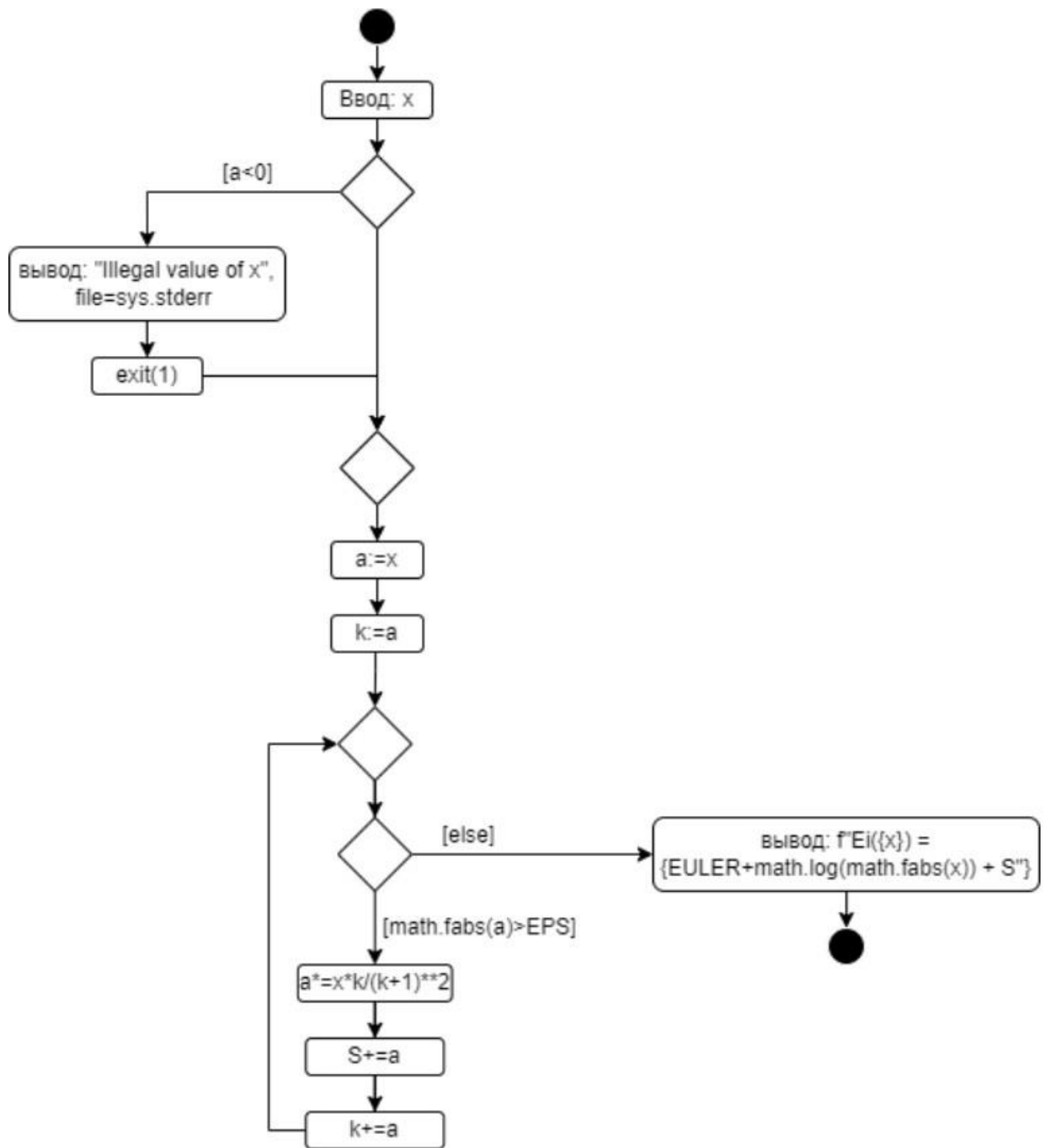


Рисунок 2.3 UML-диаграмма программы 5 примера

3. Выполнил индивидуальные задания и задание повышенной сложности.
Построил UML диаграммы программ.

Задание №1

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

```
4 """
5 3. Дано число m(1<=m<=7). Вывести на экран название дня недели, который соответствует
6 этому номеру.
7 """
8
9 import sys
10
11
12 if __name__ == '__main__':
13     m = int(input("Введите номер дня в недели: "))
14
15     if m == 1:
16         print("Понедельник")
17     elif m == 2:
18         print("Вторник")
19     elif m == 3:
20         print("Среда")
21     elif m == 4:
22
if __name__ == '__main__': > elif m == 2
1 (2) x
C:\Users\aa715\Desktop\venv\Scripts\python.exe "C:/Users/aa715/Desktop/1 (2).py"
Введите номер дня в недели: 3
Среда
Process finished with exit code 0
```

Рисунок 3.1 Программа к индивидуальному заданию №1

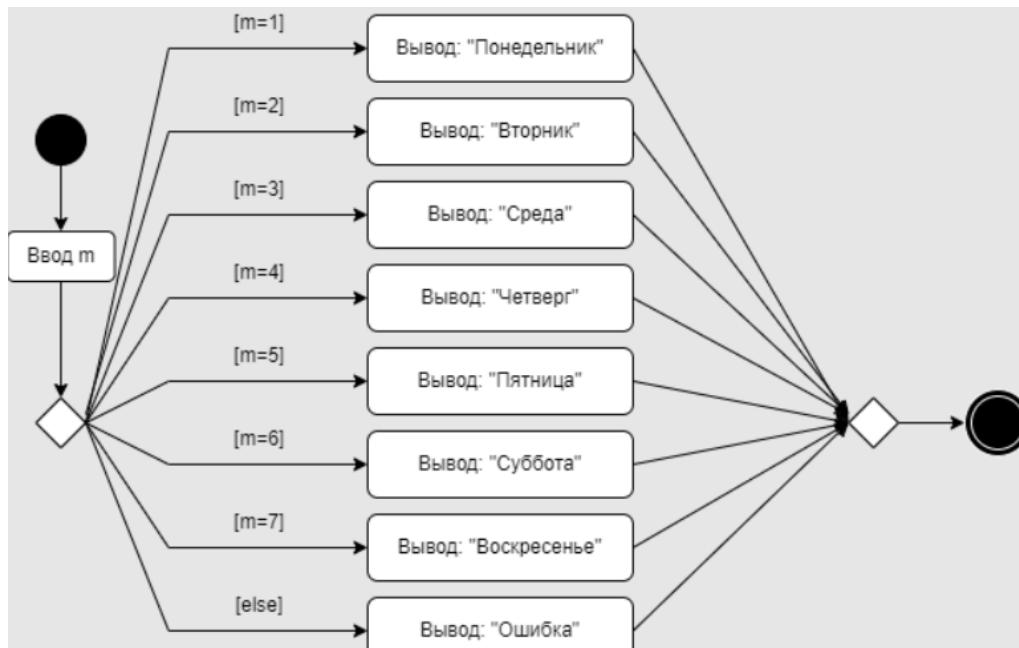
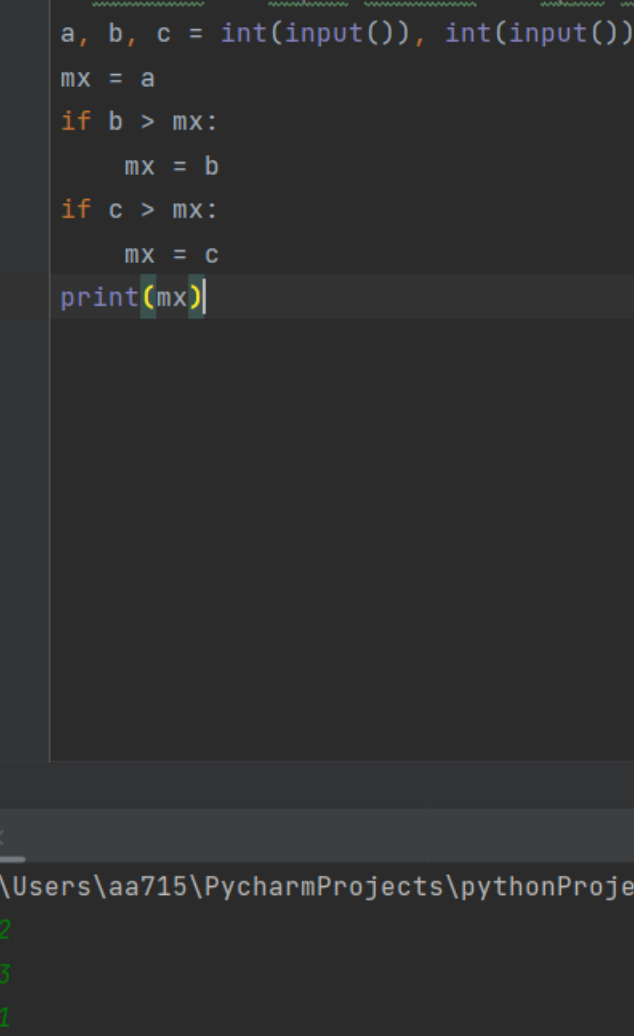


Рисунок 3.2 Блок схема к заданию №1

Задание №2

Вывести на экран большее из трёх заданных чисел.



The screenshot shows the PyCharm IDE interface. The top toolbar contains icons for running (a green play button), debugging (a blue bug icon), and other development tools. Below the toolbar, the 'Run' tab is active, displaying the execution output of the script. The output shows the program's logic for finding the maximum of three numbers: it prompts for input, reads three integers (222, 223, 221), and prints the maximum value, 223. The process finished with exit code 0.

```
# -*- coding: utf-8 -*-  
# Вывести на экран большее из трёх заданных чисел  
a, b, c = int(input()), int(input()), int(input())  
mx = a  
if b > mx:  
    mx = b  
if c > mx:  
    mx = c  
print(mx)
```

222
223
221
223

Process finished with exit code 0

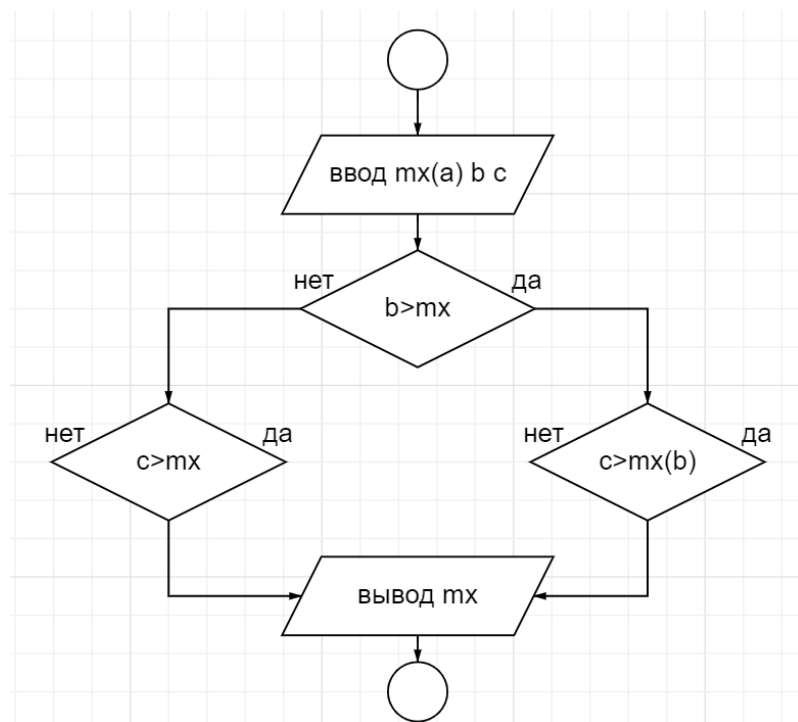
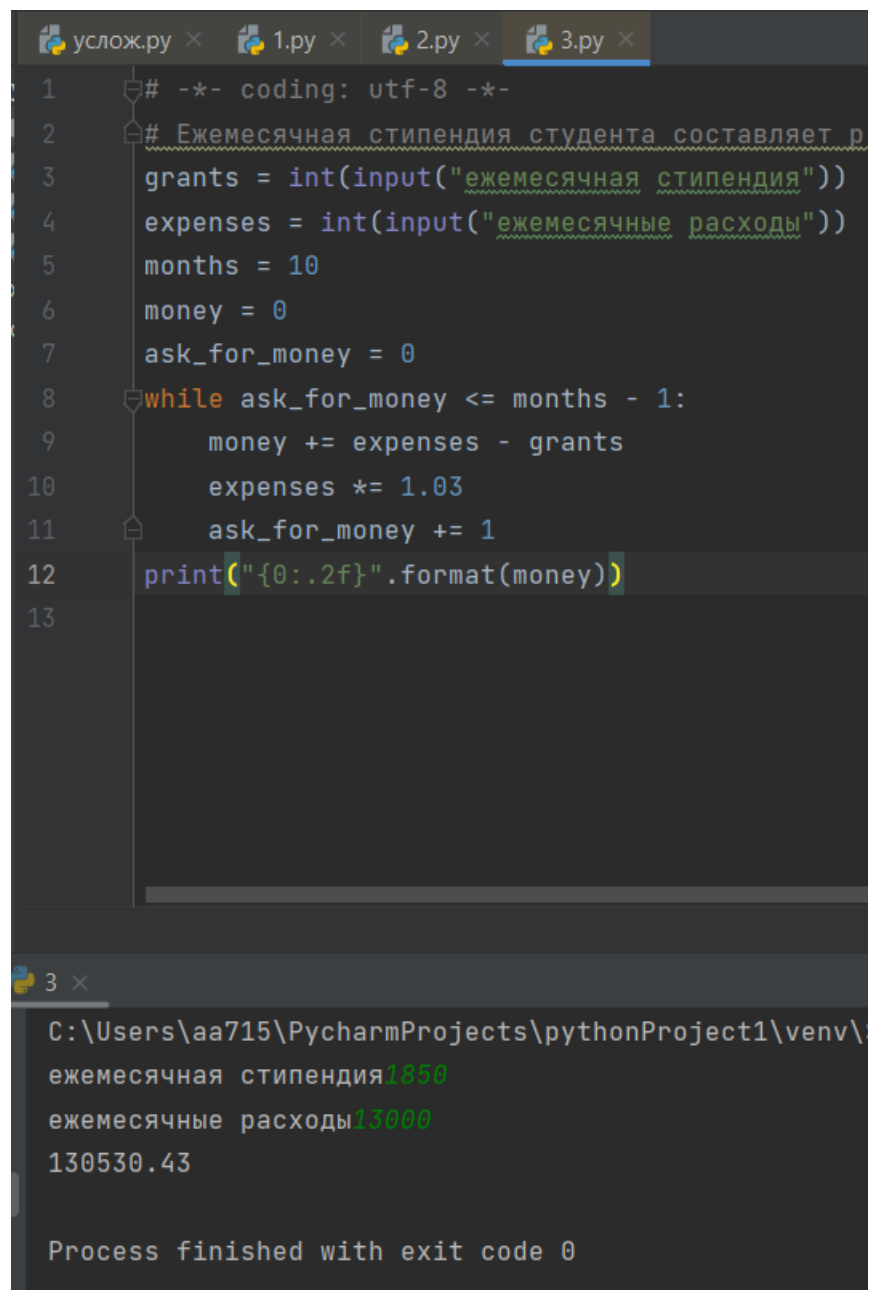


Рисунок 3.5 Блок схема к заданию №2

Задание №3

Ежемесячная стипендия студента составляет A р., а расходы на проживание превышают стипендию и составляют B р. в месяц. Рост цен ежемесячно увеличивает расходы на 3%. Составьте программу расчета необходимой суммы денег, которую надо одновременно просить у родителей, чтобы можно было прожить учебный год (10 месяцев), используя только эти деньги и стипендию.



The image shows a PyCharm IDE with a Python script in a file named 3.py. The script calculates the final amount of money after 10 months, considering monthly grants and expenses with a 3% increase in expenses each month. The execution output shows the user inputting 1850 for grants and 13000 for expenses, resulting in a final amount of 130530.43.

```
1  # -*- coding: utf-8 -*-
2  # Ежемесячная стипендия студента составляет р
3  grants = int(input("ежемесячная стипендия"))
4  expenses = int(input("ежемесячные расходы"))
5  months = 10
6  money = 0
7  ask_for_money = 0
8  while ask_for_money <= months - 1:
9      money += expenses - grants
10     expenses *= 1.03
11     ask_for_money += 1
12     print("{0:.2f}".format(money))
13
```

3 x

C:\Users\aa715\PycharmProjects\pythonProject1\venv\
ежемесячная стипендия1850
ежемесячные расходы13000
130530.43

Process finished with exit code 0

Рисунок 3.6 Задание №3

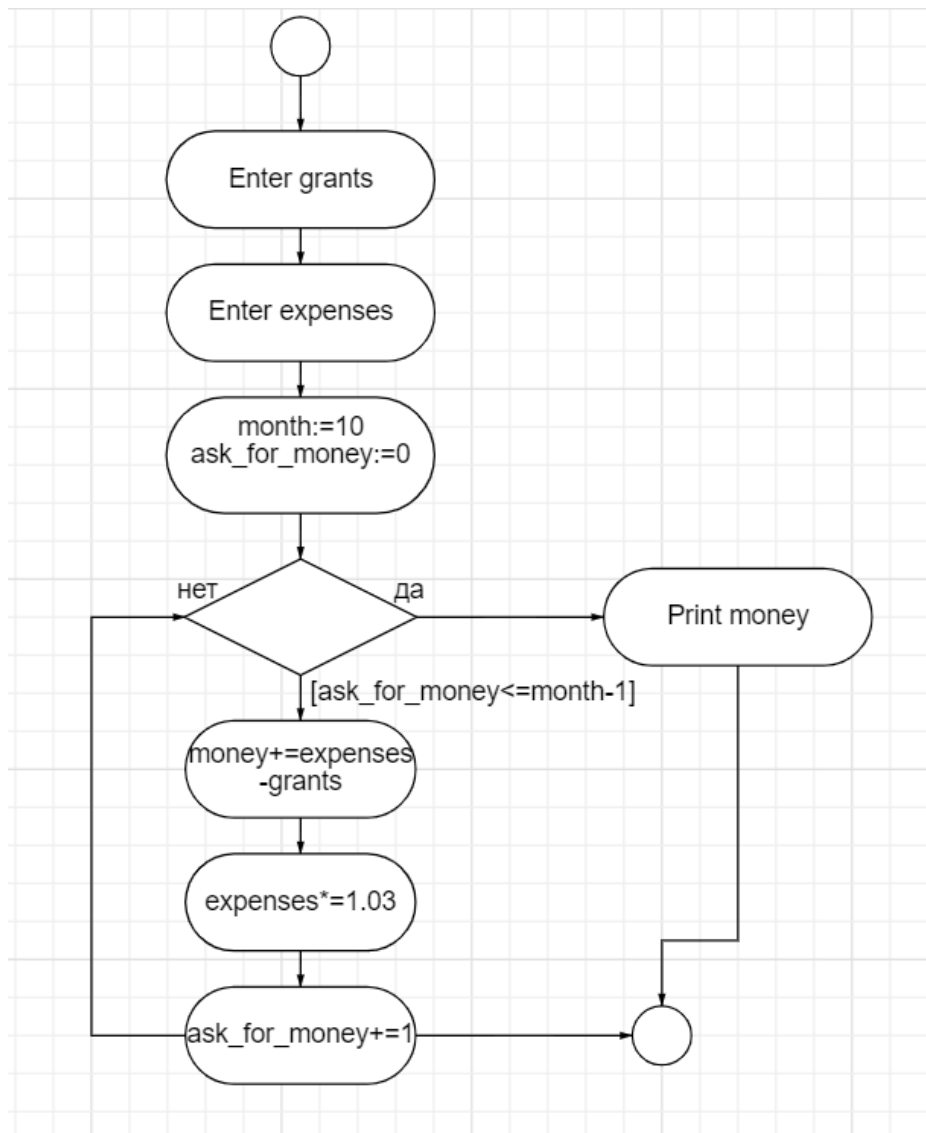


Рисунок 3.7 Блок схема к заданию №3

4. Задача повышенной сложности

Интегральный косинус:

$$\text{Ci}(x) = \gamma + \ln x + \int_0^x \frac{\cos t - 1}{t} dt = \gamma + \ln x + \sum_{n=1}^{\infty} \frac{(-1)^n x^{2n}}{(2n)(2n)!}.$$

```
услож.ру × 1.py × 2.py × 3.py ×
1  # -*- coding: utf-8 -*-
2  # интегральный косинус
3  import math
4  import sys
5
6  EULER = 0.5772156649015328606
7  EPS = 1e-10
8  if __name__ == '__main__':
9      x = float(input("x = "))
10     if x == 0:
11         print("Error", file=sys.stderr)
12         exit(1)
13     a = -x ** 2 / 4
14     S, n = a, 1
15     while math.fabs(a) > EPS:
16         a *= (-1 * x ** 2 * 2 * n) / (2 * (n + 1)) ** 2
17         S += a
18         n += 1
19     print(f"Ci({x}) = {EULER + math.log(math.fabs(x)) + S}")
    if __name__ == '__main__':

услож ×
C:\Users\aa715\PycharmProjects\pythonProject1\venv\Scripts\python.exe C
x = 32
Ci(32.0) = 1.070004389939887e+202

Process finished with exit code 0
```

Рисунок 4.1 Решение задачи

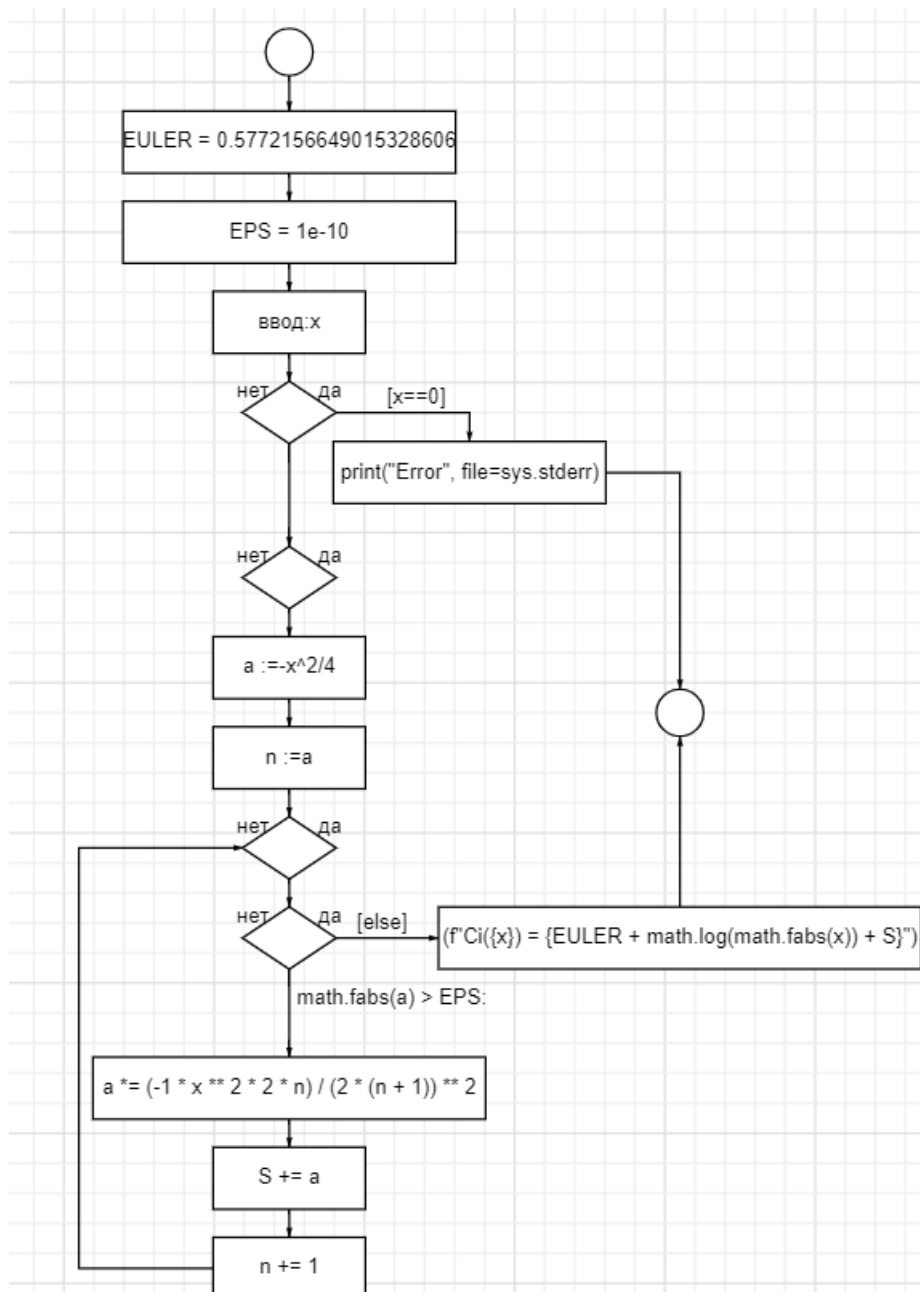


Рисунок 4.2 Блок схема к заданию №4

```

aa715@DESKTOP-70QJC9F MINGW64 ~/Desktop/224/224 (develop)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

aa715@DESKTOP-70QJC9F MINGW64 ~/Desktop/224/224 (main)
$ git merge develop
Updating 6c22c60..1cc7c07
Fast-forward
 .../1.py" | 12 ++++++++
 .../2.py" | 9 ++++++
 .../3.py" | 12 ++++++++
 .../1.py" | 13 ++++++++
 .../2.py" | 16 ++++++++
 .../3.py" | 12 ++++++++
 .../4.py" | 17 ++++++++
 .../5.py" | 25 ++++++++
 .../\321\203\321\201\320\273\320\276\320\266.py" | 19 ++++++++
9 files changed, 135 insertions(+)
 create mode 100644 "\320\270\320\275\320\264\320\270\320\262\320\270\320\264\321\2
03\320\260\320\273\321\214\320\275\321\213\320\265\1.py"
 create mode 100644 "\320\270\320\275\320\264\320\270\320\262\320\270\320\264\321\2
03\320\260\320\273\321\214\320\275\321\213\320\265\2.py"
 create mode 100644 "\320\270\320\275\320\264\320\270\320\262\320\270\320\264\321\2
03\320\260\320\273\321\214\320\275\321\213\320\265\3.py"
 create mode 100644 "\320\277\321\200\320\270\320\274\320\265\321\200\321\213\1.py"
 create mode 100644 "\320\277\321\200\320\270\320\274\320\265\321\200\321\213\2.py"
 create mode 100644 "\320\277\321\200\320\270\320\274\320\265\321\200\321\213\3.py"
 create mode 100644 "\320\277\321\200\320\270\320\274\320\265\321\200\321\213\4.py"
 create mode 100644 "\320\277\321\200\320\270\320\274\320\265\321\200\321\213\5.py"
 create mode 100644 "\321\203\321\201\320\273\320\276\320\266\320\275\320\265\320\2
75\320\275\320\276\320\265\321\203\321\201\320\273\320\276\320\266.py"

aa715@DESKTOP-70QJC9F MINGW64 ~/Desktop/224/224 (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean

aa715@DESKTOP-70QJC9F MINGW64 ~/Desktop/224/224 (main)
$ git push
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 8 threads
Compressing objects: 100% (13/13), done.
Writing objects: 100% (14/14), 3.18 KiB | 3.18 MiB/s, done.
Total 14 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Artemev6Andrey/224.git
6c22c60..1cc7c07 main -> main

```

Рисунок 4.4 Работа в консоли

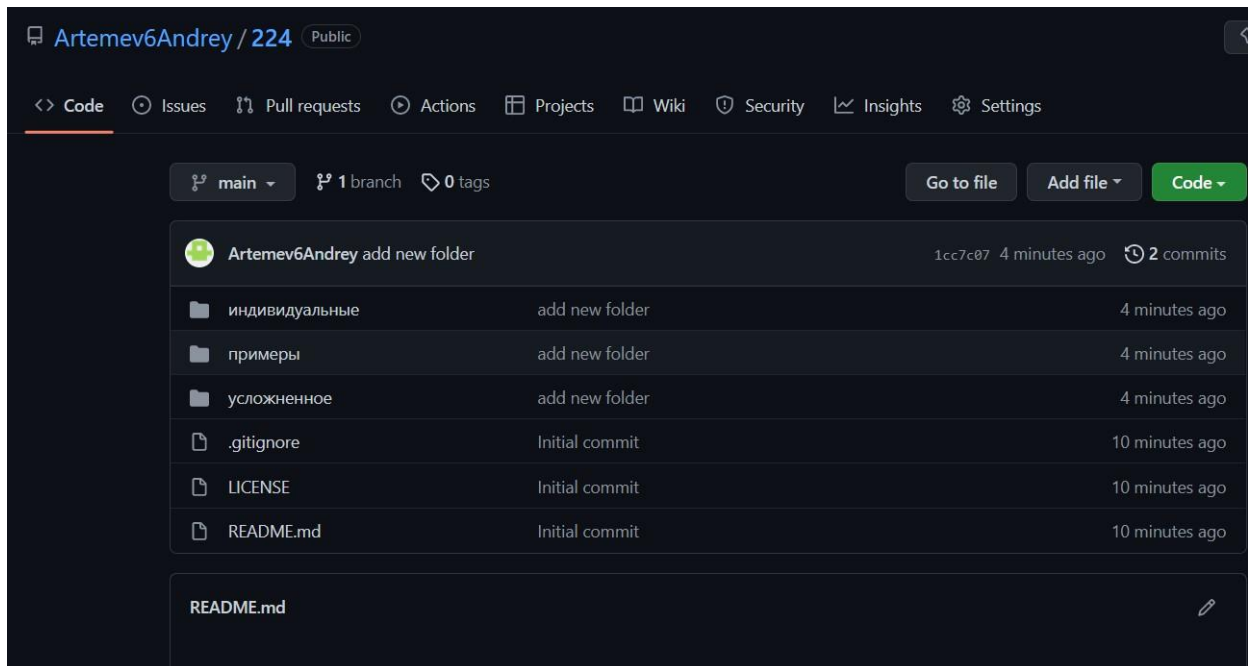


Рисунок 4.5 Изменения на удаленном репозитории

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.