МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ» ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет о лабораторной работе №5 по дисциплине «Основы программной инженерии»

Выполнил студент

2 курса, группы ПИЖ-б-о-20-1

Тотубалина С.С.

Проверил:

Доцент кафедры инфокоммуникаций,

Воронкин Р.А.

Ставрополь, 2021 г

Ход работы

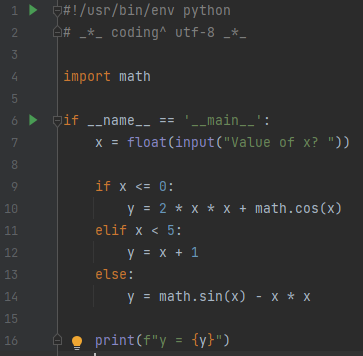


Рис. 1 – код программы lab.5\_ex.1.py

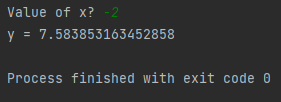


Рис. 2 – пример работы программы lab.5\_ex.1.py при х< =0

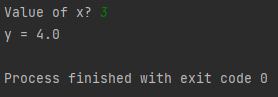


Рис. 3 – пример работы программы lab.5\_ex.1.py при х>0 и x<5

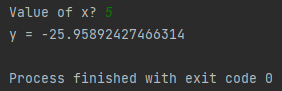


Рис. 4 – пример работы программы lab.5\_ex.1.py при x>=5

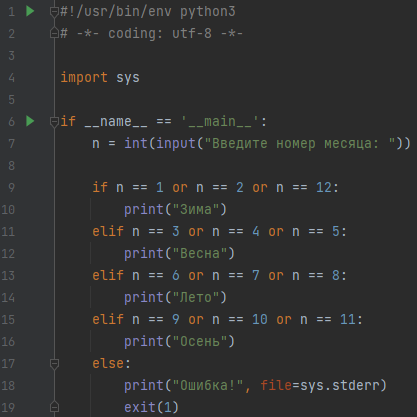


Рис. 5 – код программы lab.5\_ex.2.py

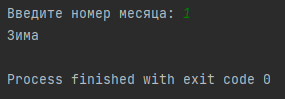


Рис. 6 – пример работы программы lab.5\_ex.2.py при n=1

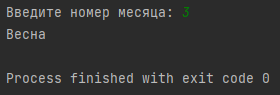


Рис. 7 – пример работы программы lab.5\_ex.2.py при n=3

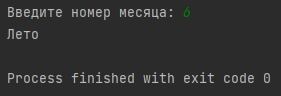


Рис. 8 – пример работы программы lab.5\_ex.2.py при n=6

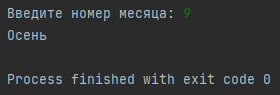


Рис. 9 – пример работы программы lab.5\_ex.2.py при n=9

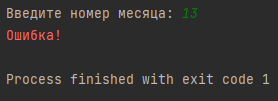


Рис. 10 – пример работы программы lab.5\_ex.2.py при n=13

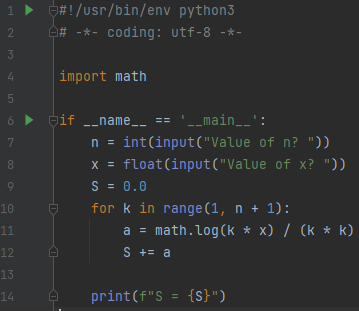


Рис. 11 – код программы lab.5\_ex.3.py

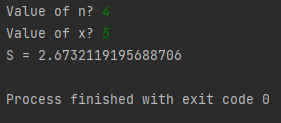


Рис. 12 – пример работы программы lab.5\_ex.3.py при n=4, x=5

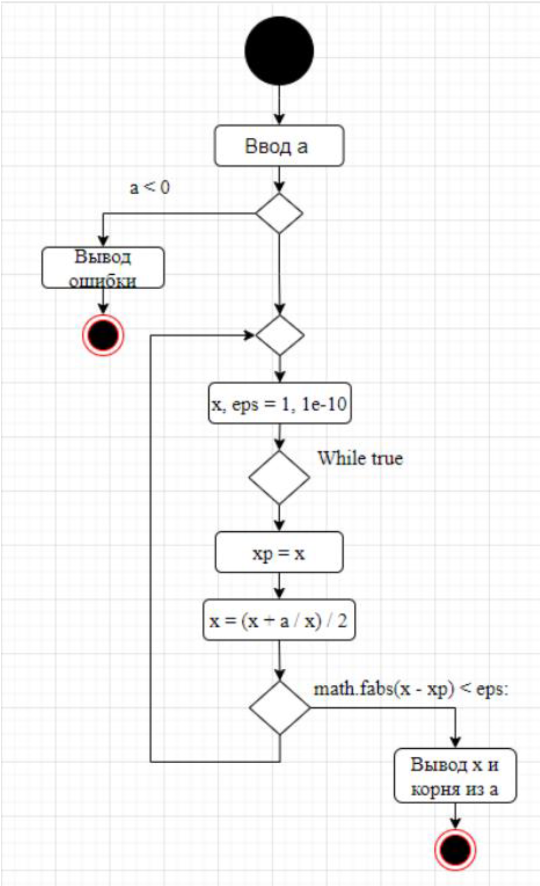


Рис. 13 – UML-диаграмма алгоритма

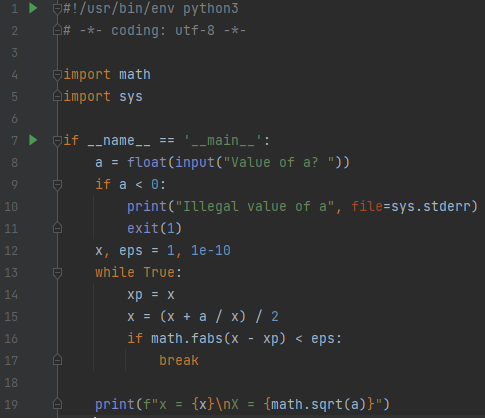


Рис. 14 – код программы

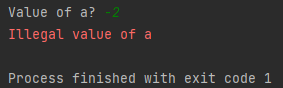


Рис. 15 – пример работы программы lab.5\_ex.4.py при a<0

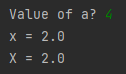


Рис. 16 – пример работы программы lab.5\_ex.4.py при a=4

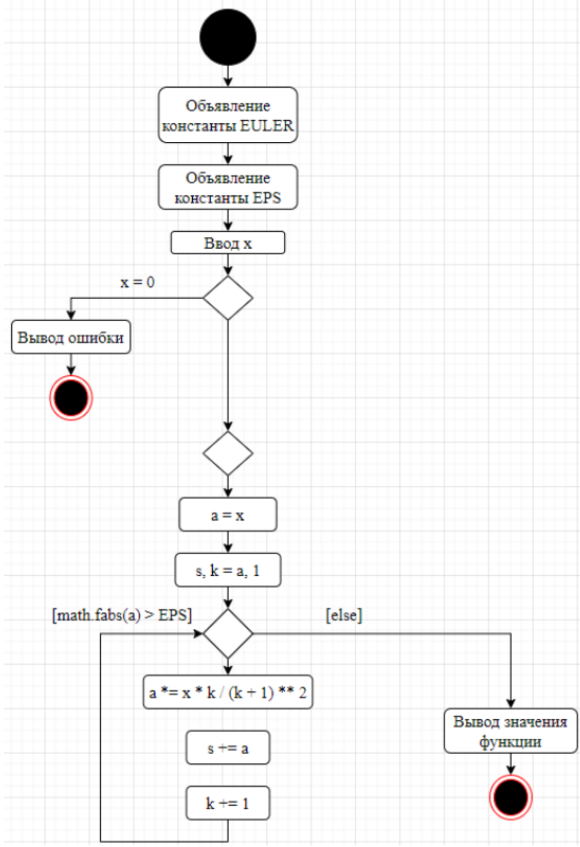


Рис. 17 – UML-диаграмма алгоритма



Рис. 18 – код программы

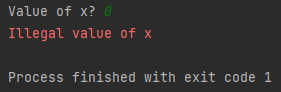


Рис. 19 – пример работы программы lab.5\_ex.5.py при x=0

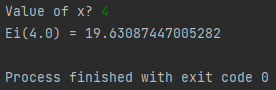


Рис. 20 – пример работы программы lab.5\_ex.5.py при x =4



Рис. 21 – индивидуальное задание №1

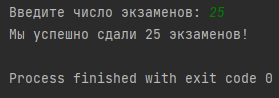


Рис. 22 – пример работы индивидуального задания №1

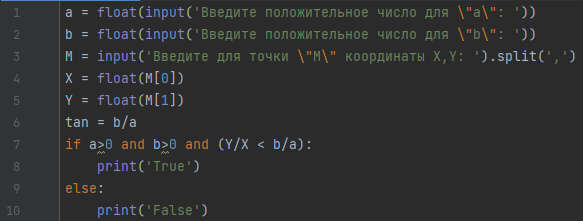


Рис. 23 – индивидуальное задание №2

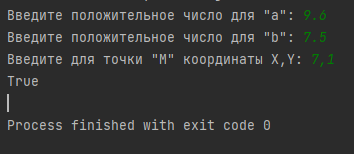


Рис. 24 – пример работы индивидуального задания №2

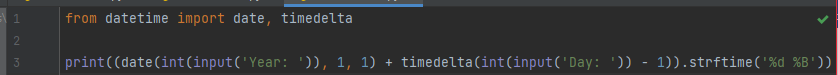


Рис. 23 – индивидуальное задание №3

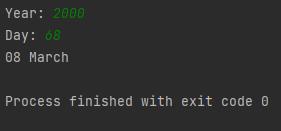


Рис. 24 – пример работы индивидуального задания №3

Ответы на вопросы:

1. Для чего нужны диаграммы деятельности UML?

Диаграммы деятельности - это один из пяти видов диаграмм, применяемых в UML для моделирования динамических аспектов поведения системы. Диаграмма деятельности - это, по существу, блок-схема, которая показывает, как поток управления переходит от одной деятельности к другой, однако, по сравнению с последней, у ней есть явные преимущества: поддержка многопоточности и объектно-ориентированного проектирования. Диаграмма деятельности (Activity diagram) показывает поток переходов от одной деятельности к другой.

2. Что такое состояние действия и состояние деятельности?

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?

В UML переход представляется простой линией со стрелкой. Точка ветвления представляется ромбом. В точку ветвления может входить ровно один переход, а выходить – два или более.

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 выполняется последовательно независимо от чего-либо, а алгоритм ветвления выполняется определенные действия в зависимости от выполнения условия или условий.

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

Оператор ветвления «if» позволяет выполнить определенный набор инструкций в зависимости от некоторого условия. Возможны следующие варианты использования:

• Конструкция «if»

• Конструкция «if» - «else»

• Конструкция «if» - «elif» - «else»

7. Какие операторы сравнения используются в Python?

<, >, <=, >=, ==, !=.

8. Что называется простым условием? Приведите примеры.

Логические выражения являются простыми, если в них выполняется только одна логическая операция. «x > 15» или «a != b»

9. Что такое составное условие? Приведите примеры.

В составных условиях используется 2 и более логические операции. «x > 8 and y <= 3»

10. Какие логические операторы допускаются при составлении сложных условий?

and и or

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

Да, может.

12. Какой алгоритм является алгоритмом циклической структуры?

Алгоритм 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 «while» и цикл «for».

14. Назовите назначение и способы применения функции range.

Функция range возвращает неизменяемую последовательность чисел в виде объекта range. Параметры функции:

• start - с какого числа начинается последовательность. По умолчанию - 0

• stop - до какого числа продолжается последовательность чисел. Указанное число не включается в диапазон.

• step - с каким шагом растут числа. По умолчанию 1

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.

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

range(15, 0, -2).

16. Могул ли быть циклы вложенными?

Да, могут.

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

a = 0 while a >= 0:

if a == 7:

break a += 1

print("A")

Оператор «break» предназначен для досрочного прерывания работы цикла «while».

18. Для чего нужен оператор break?

Оператор «break» предназначен для досрочного прерывания работы цикла «while».

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 стандартные потоки вывода на консоль: буферизованный поток stdout для вывода данных и информационных сообщений, а также небуферизованный поток stderr для вывода сообщений об ошибках. По умолчанию функция print использует поток stdout. 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.

21. Как в Python организовать вывод в стандартный поток stderr?

Для того, чтобы использовать поток stderr необходимо передать его в параметре file функции print.

22. Каково назначение функции exit?

В Python завершить программу и передать операционной системе заданный код возврата можно посредством функции exit. В данном примере выполняется вызов exit(1) , что приводит к немедленному завершению программы и операционной системе передается 1 в качестве кода возврата, что говорит о том, что в процессе выполнения программы произошли ошибки.