

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
дисциплины «Программирование на Python»

Выполнил:
Дзуев Альберт Мухаметович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», очная
форма обучения

(подпись)

Руководитель практики:
Воронкин Р.А., доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____


Ставрополь, 2023 г.

Тема: Условные операторы и циклы в языке Python

Цель: 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:

1. Создал новый репозиторий, клонировал его, в нем создал ветку developer и перешел на нее.
2. Проработал примеры лабораторной работы:



```
ythion\2\prim1.py
Value of x? 3
y = 4.0
```

Рисунок 1. Вывод программы примера 1



```
ythion\2\prim2.py
Введите номер месяца: 3
Весна
```

Рисунок 2. Вывод программы примера 2

```

python\2\prim3.py"
Value of n? 4
Value of x? 3
S = 1.9459948857353426

```

Рисунок 3. Вывод программы примера 3

```

Value of a? -1
Illegal value of a

```

Рисунок 4. Вывод программы примера 4

```

python\2\prim5.py"
Value of x? 0
Illegal value of x

```

Рисунок 5. Вывод программы примера 5

3. Сделал UML диаграммы для примеров 4 и 5:

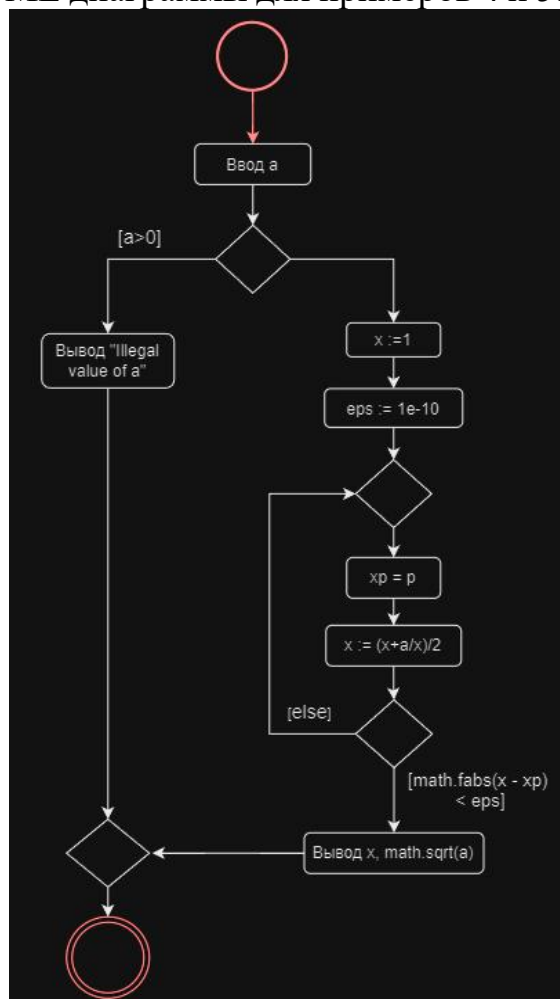


Рисунок 6. UML диаграмма примера 4

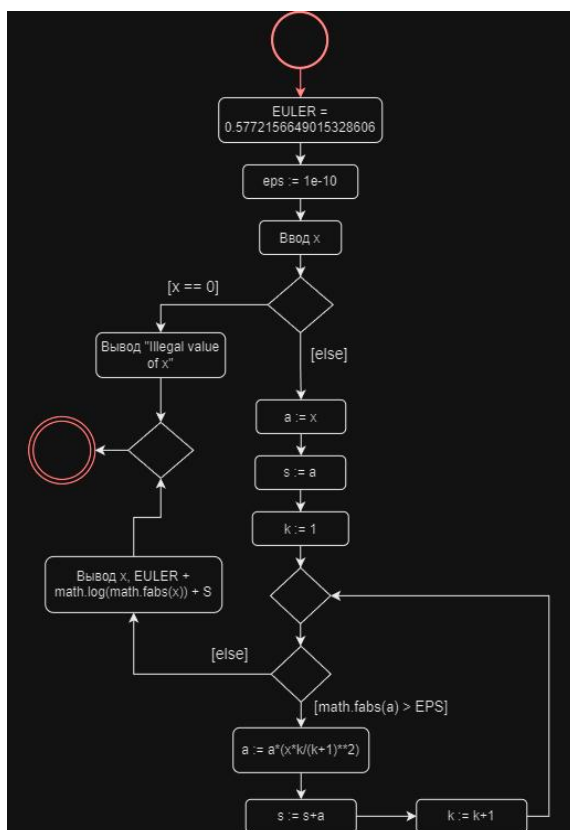
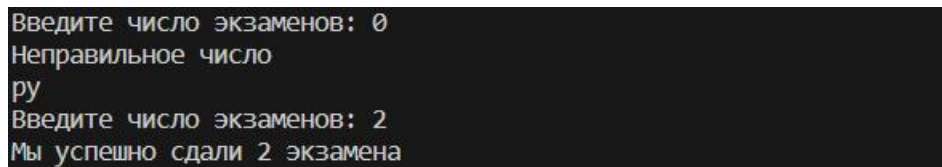


Рисунок 7. UML диаграмма примера 5

4. Выполнил индивидуальное задание 1: Вводится число экзаменов $N \leq 20$. Напечатать фразу "Мы успешно сдали N экзаменов", согласовав слово "экзамен" с числом N.

Код программы:

```
if __name__ == '__main__':  
    count = int(input("Введите число экзаменов: "))  
    if count <= 0:  
        print("Неправильное число")  
    elif count == 1:  
        print("Мы успешно сдали 1 экзамен")  
    elif count < 5:  
        print(f"Мы успешно сдали {count} экзамена")  
    elif count <= 20:  
        print(f"Мы успешно сдали {count} экзаменов")  
    else:  
        print("Слишком много экзаменов")
```



```
Введите число экзаменов: 0  
Неправильное число  
ру  
Введите число экзаменов: 2  
Мы успешно сдали 2 экзамена
```

Рисунок 8. Результат работы программы индивидуального задания 1

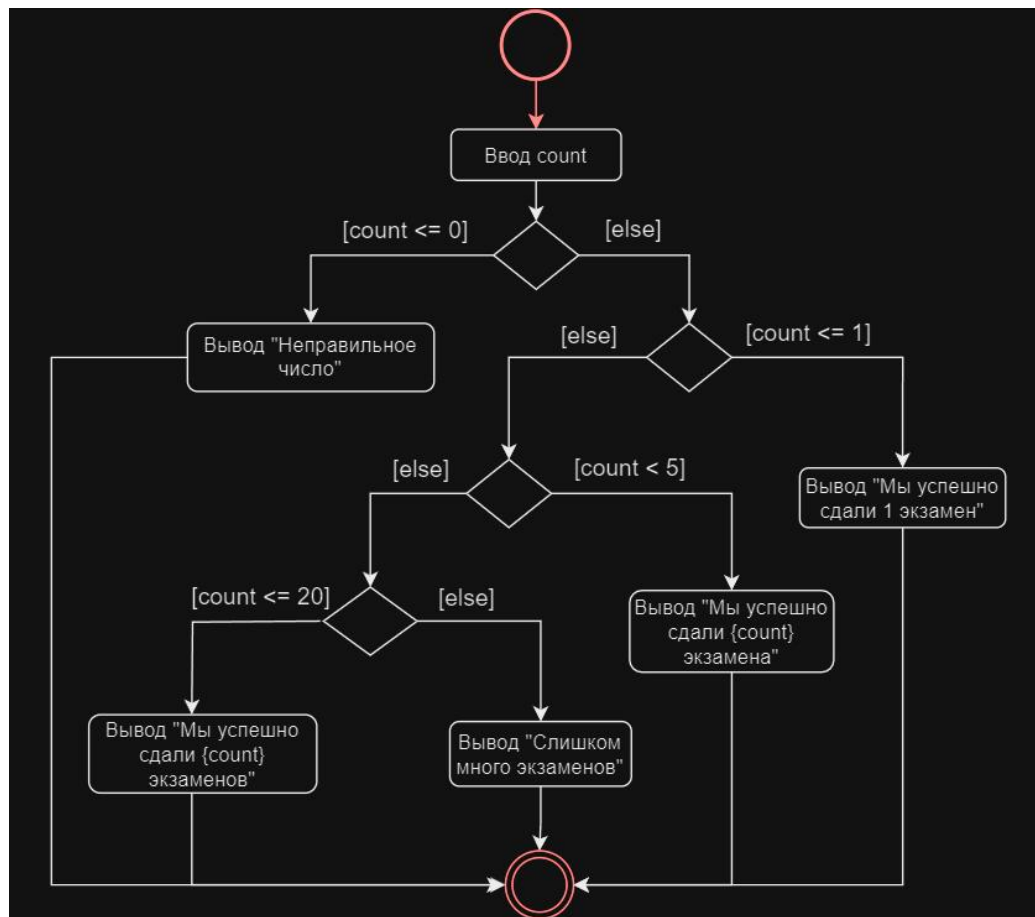


Рисунок 9. UML диаграмма индивидуального задания 1

5. Выполнил индивидуальное задание 2: Найти координаты точки пересечения прямых заданных уравнениями $a_1x + b_1y + c_1 = 0$ и $a_2x + b_2y + c_2 = 0$, либо сообщить совпадают, параллельны или не существуют.

Код программы:

```

if __name__ == '__main__':
    A1, B1, C1 = map(float, input('Введите A1, B1, C1 через пробел: ').split())
    A2, B2, C2 = map(float, input('Введите A2, B2, C2 через пробел: ').split())
    if ((A1 == 0) and (B1 == 0)) or ((A2 == 0) and (B2 == 0)):
        print('прямые не существуют')
    else:
        if (A1 * B2 == A2 * B1) and (A1 * C2 == A2 * C1):
            print('прямые совпадают')
        elif A1 * B2 == A2 * B1:
            print('прямые параллельны')
        else:

```

$$X = (C1 * B2 - C2 * B1) / (B1 * A2 - B2 * A1)$$

$$Y = (C2 * A1 - C1 * A2) / (B1 * A2 - B2 * A1)$$

```
print(f'X = {X:.2f}, Y = {Y:.2f}')
```

```
Введите A1, B1, C1 через пробел: 3 2 1
Введите A2, B2, C2 через пробел: 3 2 1
прямые совпадают
PS C:\Users\dzuev> & C:/Users/dzuev/AppData/Local/Microsoft/WindowsApps/python3.12.exe c:/Users/dzuev/Untitled-1.
ру
Введите A1, B1, C1 через пробел: 8 7 4
Введите A2, B2, C2 через пробел: 8 7 5
прямые параллельны
PS C:\Users\dzuev> & C:/Users/dzuev/AppData/Local/Microsoft/WindowsApps/python3.12.exe c:/Users/dzuev/Untitled-1.
ру
Введите A1, B1, C1 через пробел: 1 5 2
Введите A2, B2, C2 через пробел: 4 8 5
X = -0.75, Y = -0.25
```

Рисунок 10. Результат работы программы индивидуального задания 2

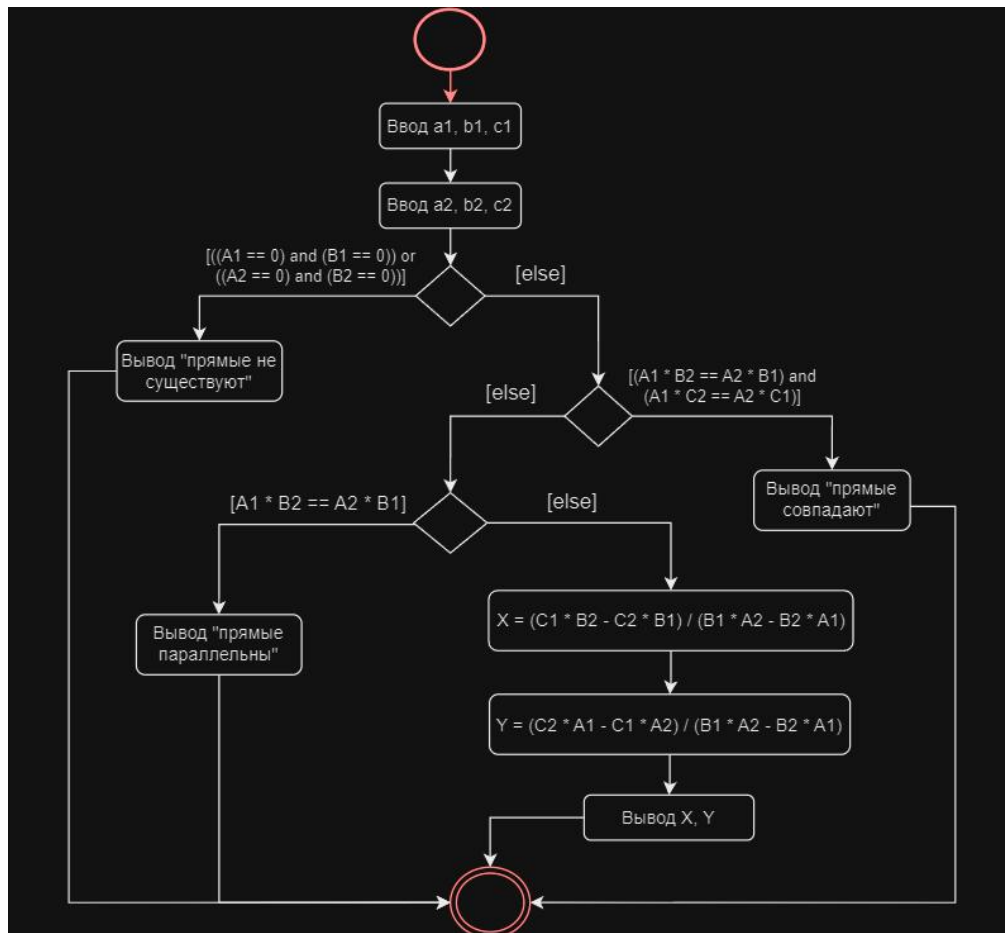
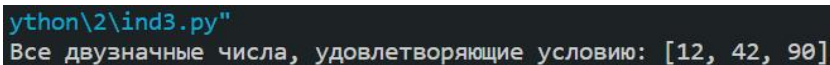


Рисунок 11. UML диаграмма индивидуального задания 2

6. Выполнил индивидуальное задание 3: Если к сумме цифр двузначного числа прибавить квадрат этой суммы, то снова получится это двузначное число. Найти все эти числа.

Код программы:

```
print_numbers = []
if __name__ == '__main__':
    for number in range(10, 100):
        digit_one = number // 10 # Получаем первую цифру
        digit_two = number % 10 # Получаем вторую цифру
        digit_sum = digit_one + digit_two # сумма цифр
        squared_sum = digit_sum * digit_sum # Возводим сумму в квадрат
        if number == (digit_sum + squared_sum):
            print_numbers.append(number)
    print("Все двузначные числа, удовлетворяющие условию:", print_numbers)
```



```
ythn\2\ind3.py
Все двузначные числа, удовлетворяющие условию: [12, 42, 90]
```

Рисунок 12. Результат работы программы индивидуального задания 3

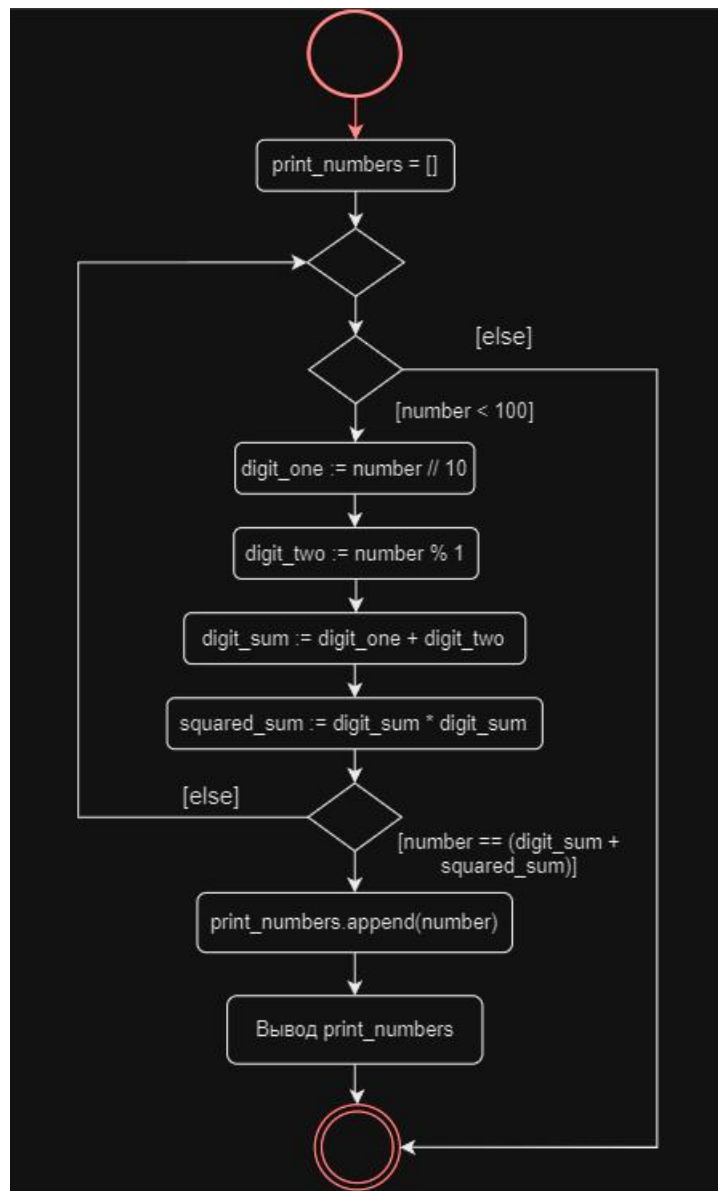


Рисунок 13. UML схема индивидуального задания 3

7. Выполнил задание повышенной сложности вариант 9: составить UML-диаграмму деятельности, программу и произвести вычисления вычисление значения специальной функции по ее разложению в ряд с точностью $\epsilon = 10^{-10}$, аргумент функции вводится с клавиатуры.

$$f(x) = - \int_1^x \frac{\ln t}{t-1} dt = \sum_{k=1}^{\infty} \frac{(-1)^k (x-1)^k}{k^2}, \quad 0 \leq x \leq 2$$

$$a_k = \frac{(-1)^k (x-1)^k}{k^2}$$

$$a_{k+1} = \frac{(-1)^{k+1} (x-1)^{k+1}}{(k+1)^2}$$

$$\Rightarrow \frac{a_{k+1}}{a_k} = \frac{(-1)^{k+1} (x-1)^{k+1}}{(k+1)^2} \cdot \frac{k^2}{(-1)^k (x-1)^k} =$$

$$= \frac{-1 (x-1) k^2}{(k+1)^2} = - \frac{(x-1) k^2}{(k+1)^2}$$

$$a_{k+1} = \frac{-(x-1) k^2}{(k+1)^2} a_k$$

$$a_0 = \frac{(-1)^0 (x-1)^0}{1^2} = 1$$

$$a_1 = \frac{(-1)^1 (x-1)^1}{1^2} = -x + 1$$

Рисунок 14. Ход вычисления рекуррентного соотношения

Код программы:

```
import math
import sys
if __name__ == '__main__':
    x = float(input("Введите значение x: "))
    if (x < 0) or (x > 2):
        print("Illegal value of x", file=sys.stderr)
        exit(1)
    result = -x + 1
    k = 1
    EPS = 1e-10
    S = result
    while math.fabs(S) > EPS:
        S *= (-(x-1)*k**2)/((k+1)**2)
        result += S
        k += 1
    print("Вычисление с использованием разложения в ряд: ", result)
```

```

PS C:\Users\dalam\OneDrive\Рабочий стол\projects\projects\Python> python -u
Введите значение x: 2
Вычисление с использованием разложения в ряд: -0.822467033374095
PS C:\Users\dalam\OneDrive\Рабочий стол\projects\projects\Python> python -u
Введите значение x: 0
Вычисление с использованием разложения в ряд: 1.6449240668982423
PS C:\Users\dalam\OneDrive\Рабочий стол\projects\projects\Python> python -u
Введите значение x: 5
Illegal value of x

```

Рисунок 15. Результат работы программы задания повышенной сложности

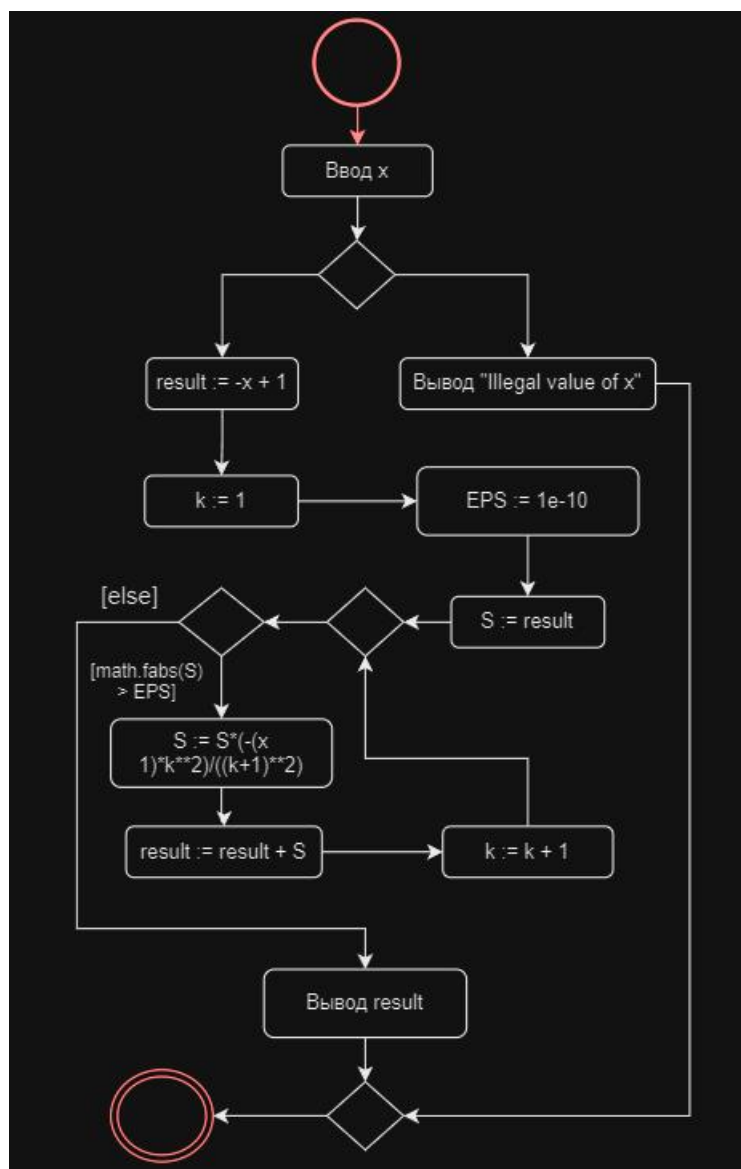


Рисунок 16. UML диаграмма усложненного задания

Ответы на контрольные 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.

2. Что такое состояние действия и состояние деятельности?

В потоке управления, моделируемом диаграммой деятельности, происходят различные события. Можно 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?

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. В UML переход представляется простой линией со стрелкой.

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

4. Какой алгоритм является алгоритмом разветвляющейся структуры?

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. Как видно из рис. 4.3, 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.

Для удобства разрешается использовать ключевое слово `else` для пометки того из исходящих переходов, который должен быть выбран в случае, если условия, заданные для всех остальных переходов, не выполнены.

Реализовать итерацию можно, если ввести два состояния действия - в

первом устанавливается значение счетчика, во втором оно увеличивается - и точку ветвления, вычисление в которой показывает, следует ли прекратить итерации.

5. Чем отличается разветвляющийся алгоритм от линейного?

Линейный алгоритм – алгоритм, все этапы которого выполняются однократно и строго последовательно.

Разветвляющийся алгоритм – 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.

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

Условные операторы – это специальные конструкции, благодаря которым в программе происходит ветвление в зависимости от условий.

Условный оператор имеет полную и краткую формы.

7. Какие операторы сравнения используются в Python?

If, elif, else.

8. Что называется простым условием? Приведите примеры.

Простое условие в программировании – это выражение, которое может быть истинным или ложным. Оно используется для принятия решений в коде на основе значения переменных или других условий.

Пример: $x > 5$

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?

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.

В Python есть 2 типа циклов: – цикл while, – цикл for.

Оператор for выполняет указанный набор инструкций заданное количество раз, которое определяется количеством элементов в наборе. Оператор цикла while выполняет указанный набор инструкций до тех пор, пока условие цикла истинно.

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.

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

for i in 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. Чтобы выйти из цикла нужно использовать оператор break.

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?

Ввод и вывод распределяется между тремя стандартными потоками: stdout – стандартный вывод (экран), stderr – стандартная ошибка (вывод ошибок на экран)

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

Во-первых нужно импортировать sys, а дальше использовать print(..., file=sys.stderr).

22. Каково назначение функции exit?

Функция exit() модуля sys – выход из Python.

Вывод: в результате выполнения работы были приобретены навыки программирования разветвляющихся алгоритмов и алгоритмов циклической структуры, освоены операторы языка Python версии 3.x if , while , for , break и continue, позволяющие реализовывать разветвляющиеся алгоритмы и алгоритмы циклической структуры.