ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ**

**ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

(НИУ «БелГУ»)

ИНСТИТУТ ИНЖЕНЕРНЫХ И ЦИФРОВЫХ ТЕХНОЛОГИЙ

КАФЕДРА МАТЕМАТИЧЕСКОГО

И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

**Отчет по лабораторной работе 5**

**по дисциплине: «Теория формальных языков»**

студента очного отделения

4 курса 12001801 группы

Капустина Виктора Сергеевича

Проверил(а):

доц. Великая Яна Геннадьевна

Белгород 2022

**Вариант 4**

**Цель занятия:** изучить регулярные выражения, описывающие регулярные множества; изучить описание регулярных выражений средствами языка программирования высокого уровня.

**Задание**

1. Построить регулярное выражение, описывающее формальный язык (**варианты заданий совпадают с вариантами заданий по лаб.работе №3**)

2. Написать программу, которая использует регулярное выражение, в качестве шаблона для распознавания принадлежности строк формальному языку.

Исходные данные: строка.

Результат: “допустить” – если строка представляет собой цепочку за- данного языка

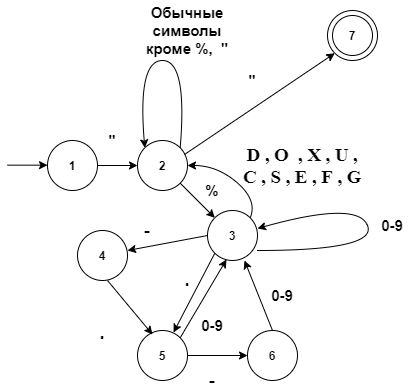
“отвергнуть” – в противном случае.

3. Написать программу, использующую регулярное выражение, которая оставляет в исходном текстовом файле только цепочки заданного языка.

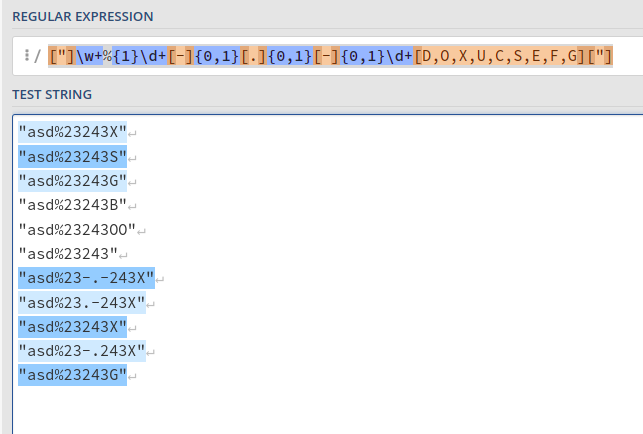
4. Написать программу, использующую регулярное выражение, которая исключает из исходного текстового файла подстроки, являющиеся цепочками заданного языка.

**Индивидуальное задание: *Язык управляющих строк.***

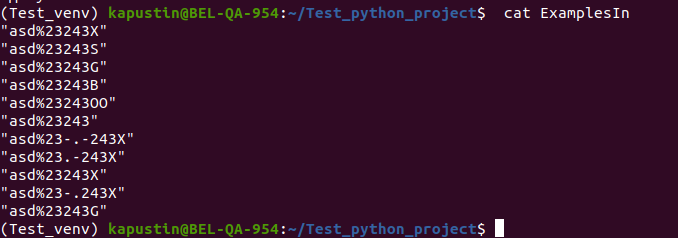
Управляющая строка начинается и заканчивается кавычками, содер­жит обычные символы и спецификации. Спецификация начинается символом *%* и заканчивается символом преобразования. Между *%* и символом преоб­разования может находиться формат, представляющий собой последователь­ность цифр, возможно, разделённых на две части точкой, перед которой или за которой может находиться знак минус. Символами преобразования явля­ются: *D , O , X , U , C , S , E , F , G* . Символом *%* может начинаться только спецификация.

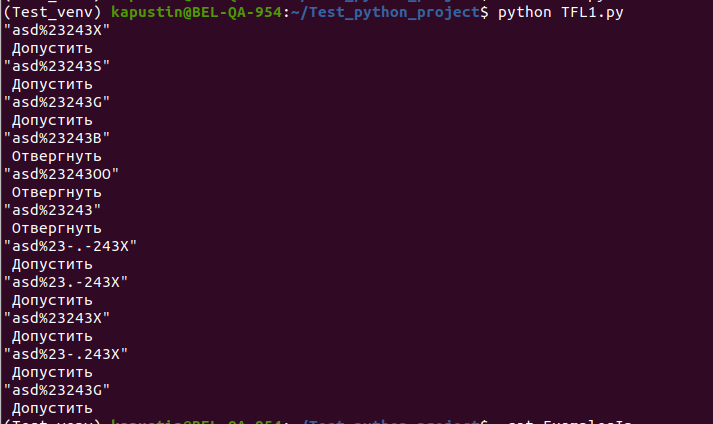
  
Рис. 1. Схема автомата

Для ввода в программу был составлен и проверен(Через специализированный сайт для составления регулярных выражений) список строк соответствующий основным аспектам составления регулярного выражения по заданной схеме.

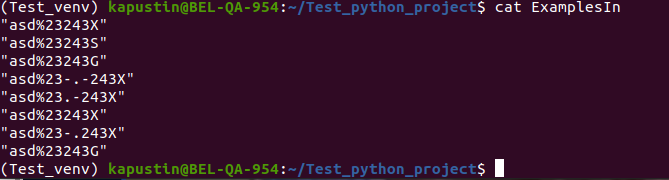
  
Рис. 2 Проверка примеров на вход

Таким образом, все выделенные примеры соответствуют условиям и должны быть допущены. Не выделенные соответственно отвергнуты. Перед запуском программы заполним файл, с которого должна читать программа данными примерами (Рисунок 3)

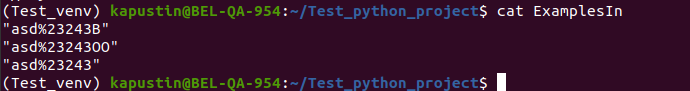
  
Рис. 3 Заполненный файл примерами на вход

  
Рис. 4 Результат работы программы

Как видим из результата работы программы, (Рисунок 4) программы допустила соответствующие регулярному выражению строки, и отвергла строки не соответствующие. Если теперь, мы проверим файл с выходными строками, можно заметить, что там располагаются только строки, которые были допущены в ходе работы программы (Рисунок 5). По аналогии возможно сделать, изменив параметр, по которому записываться в файл будет массив с недопущенными цепочками.

  
Рис. 5 Файл с записанными примерами

Также изменим параметр записи в файл и будем допущенные работы исключать из записи файла. (Рисунок 6)

  
Рис. 6. Файл с записанными примерами

**Вывод:** на данной лабораторной работе изучил регулярные выражения, описывающие регулярные множества; изучил описание регулярных выражений средствами языка программирования высокого уровня.

**Листинг приложения 1**

import re

pattern1= r'["]\w+%{1}\d+[-]{0,1}[.]{0,1}[-]{0,1}\d+[D,O,X,U,C,S,E,F,G]["]'

true\_const = []

false\_const = []

File\_in = open('ExamplesIn')

for line in File\_in:

if re.findall(pattern1, line):

print(line,'Допустить')

true\_const.append(line)

if line not in true\_const:

print(line, 'Отвергнуть')

false\_const.append(line)

File\_in.close()

File\_out = open('ExamplesIn', 'w')

for line in true\_const:

File\_out.write(line)

File\_out.close()

**Конец листинга 1**