МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра прикладной математики

Распознавание типов формальных языков и грамматик

ОТЧЕТ

По лабораторной работе

по дисциплине

Формальные языки и алгоритмы

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Жевнечуг Д.В.

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Зырянов Е.А.

22-ПМ-1

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Содержание

[Задание на лабораторную работу: 3](#__RefHeading___Toc2097_1008937971)

[Теоретическая справка 4](#__RefHeading___Toc2099_100893797)

[Реализация 7](#__RefHeading___Toc2101_100893797)

[Результат 10](#__RefHeading___Toc2103_100893797)

[Список литературы 11](#__RefHeading___Toc2105_100893797)

[Приложение 12](#__RefHeading___Toc2419_100893797)

# Задание на лабораторную работу:

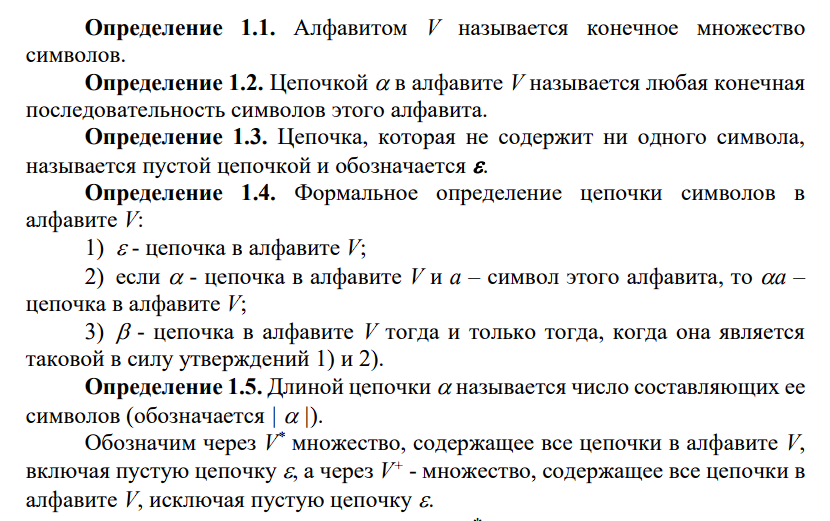
1. Составить грамматику, порождающую формальный язык, заданный в соответствии с вариантом;
2. Определить тип формальной грамматики и языка по классификации Хомского;
3. Разработать программное средство, распознающее тип введенной пользователем грамматики по классификации Хомского.

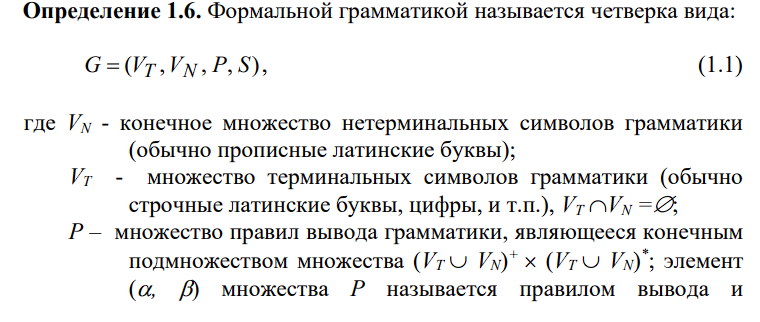
Цель работы:

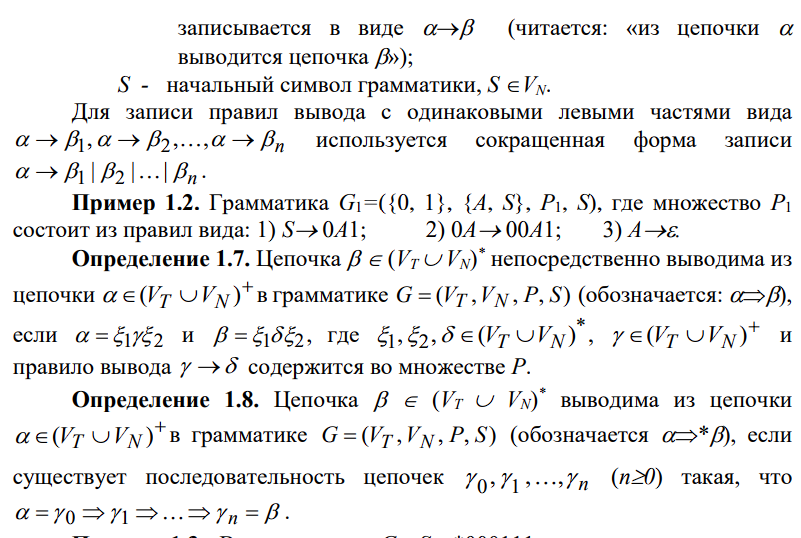
1) Закрепить понятия «алфавит», «цепочка», «формальная грамматика» и «формальный язык», «выводимость цепочек», «эквивалентная грамматика».

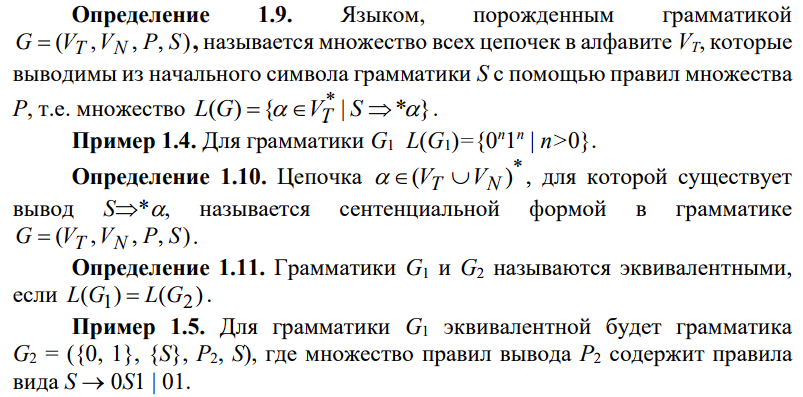
2) Сформировать умения и навыки распознавания типов формальных языков и грамматик по классификации Хомского, построения эквивалентных грамматик.

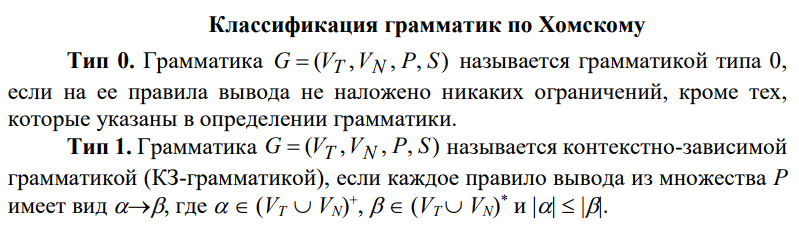
# Теоретическая справка

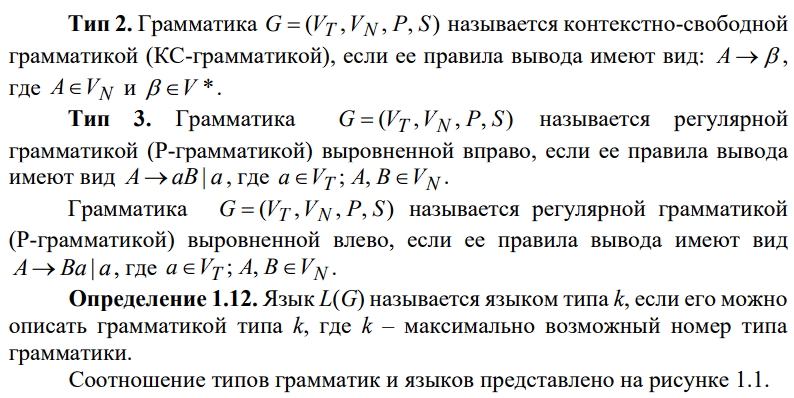


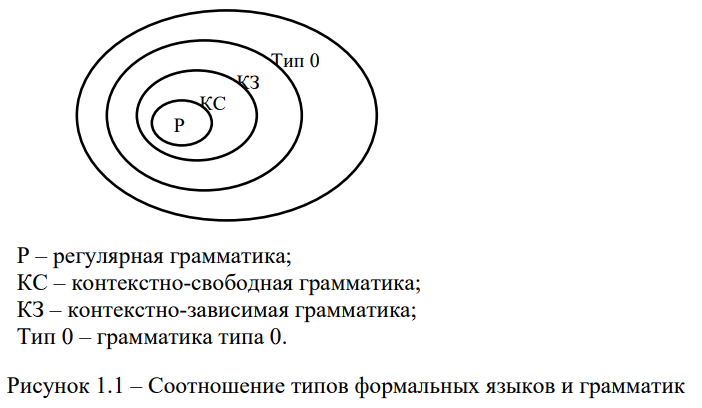












# Реализация

L(G) = { 0n (10)m | n,m >= 0 }

G1 = ({0, 1}, {S, B}, p1, S)

p1:

1) S → 0B

2) 0B → 0B1010

3) B → 1010

4) 0B → 00B

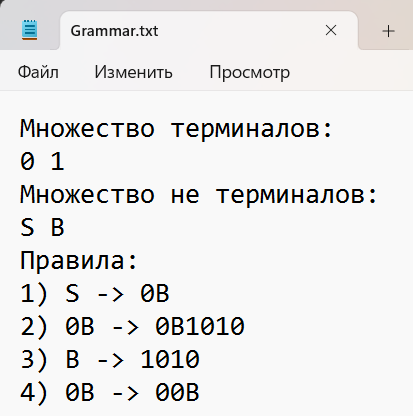
Цепочки:

S→0B→0B1010→010101010;

S→0B→01010.

G1 является 1 типом грамматики.

Создаем файл с грамматикой:

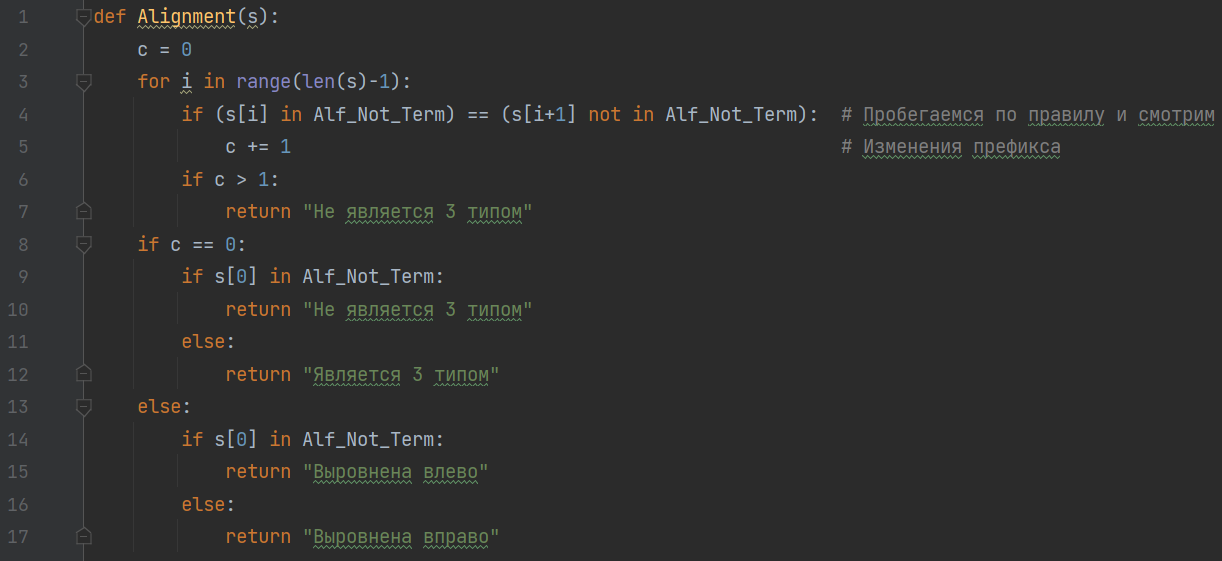


Хочу отметить что именно при таком заполнении файла всё работает, файл должен находиться там же где и программа и правил должно быть не больше 99.

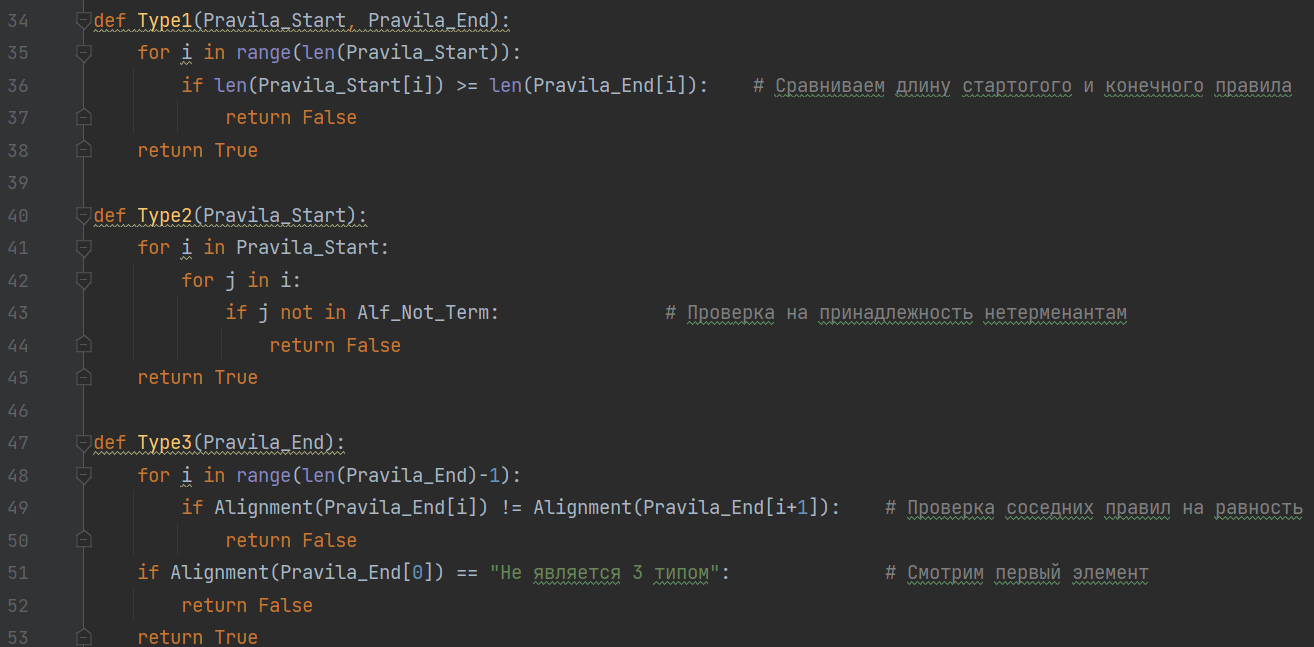
На рисунке 1 представлен фрагмент кода для работы с файлом:



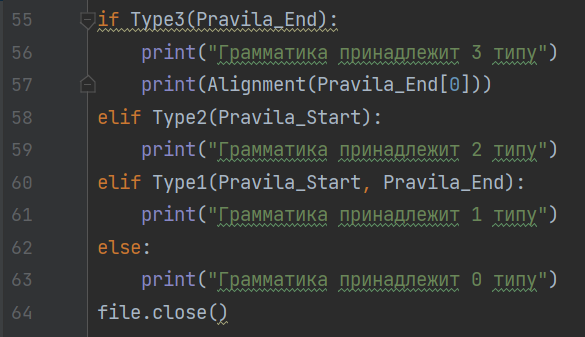
На рисунке 2 представлен фрагмент кода функции выравнивания, которая проверяет изменения префикса:



На рисунке 3 представлен фрагмент кода функции определяющие тип грамматики:

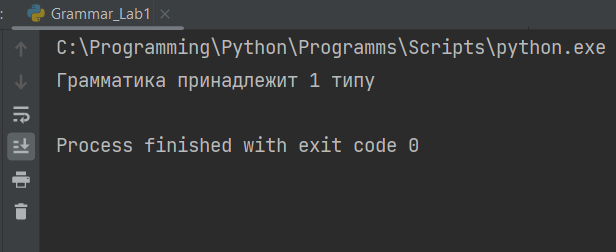


На рисунке 3 представлен фрагмент кода проверки принадлежность нашей грамматики:



# Результат

Проверка работы программы была выполнена на примере исходной грамматики:



Проверка работы программы на примере грамматики G2:

G2 = ({0, 1}, {S, B}, p1, S)

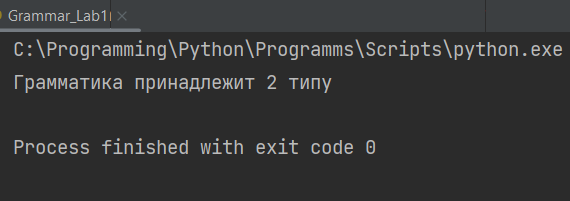
p2:

1) S → 0B

2) B → 01B1010BB

3) B → 1010B

4) B → 00B



Проверка работы программы на примере грамматики G3:

G3 = ({0, 1}, {S, B}, p1, S)

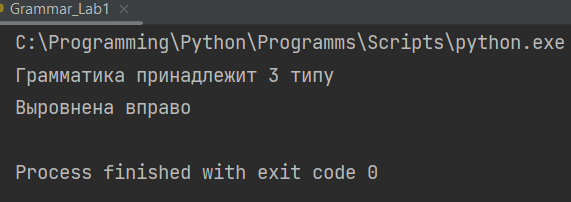
p3:

1) S → 0B

2) 0B → 011010BB

3) B → 1010B

4) 0B → 00B



Проверка работы программы на примере грамматики G4:

G4 = ({0, 1}, {S, B}, p1, S)

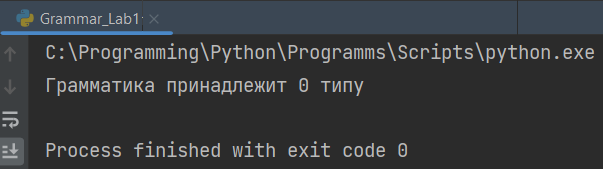
p4:

1) S → 0B

2) 0B → 01B010BB

3) B → 1010B

4) 0BB → 0B



Проверка работы программы на примере грамматики G5:

G5 = ({0, 1}, {S, B}, p5, S)

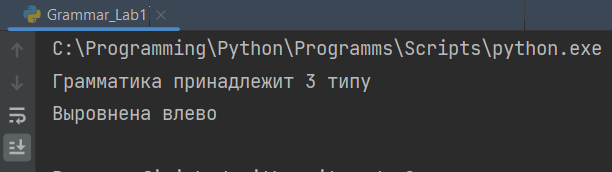
p5:

1) S → B0

2) 0B → BB1010

3) B → B1010

4) BB → B0



Вывод: Закрепили понятия «алфавит», «цепочка», «формальная грамматика» и «формальный язык», «выводимость цепочек», «эквивалентная грамматика».Сформировали умения и навыки распознавания типов формальных языков и грамматик по классификации Хомского, построения эквивалентных грамматик.

# Список литературы

1. Пентус А. Е., Пентус М. Р. Теория формальных языков. Учебное пособие. - 2013.С.80.

2. Волкова И. А., Вылиток А. А., Руденко Т. В. Формальные грамматики и языки. Элементы теории трансляции. – 2009. С.115.

# Приложение

def Alignment(s):  
 c = 0  
 for i in range(len(s)-1):  
 if (s[i] in Alf\_Not\_Term) == (s[i+1] not in Alf\_Not\_Term): # Пробегаемся по правилу и смотрим  
 c += 1 # Изменения префикса  
 if c > 1:  
 return "Не является 3 типом"  
 if c == 0:  
 if s[0] in Alf\_Not\_Term:  
 return "Не является 3 типом"  
 else:  
 return "Является 3 типом"  
 else:  
 if s[0] in Alf\_Not\_Term:  
 return "Выровнена влево"  
 else:  
 return "Выровнена вправо"  
  
Alf\_Not\_Term = "ABCDEFGHIJKLMNOPQRSTUVWXYZ" # Алфавит нетерминантов  
file = open('Grammar.txt', 'r', encoding='utf - 8') # Открываем файл  
s = file.read() # Читаем файл  
s = s.split('\n') # Разделяем на строки  
Term = s[1].split() # Обозначаем терминанты  
Not\_Term = s[3].split() # Обозначаем не терминанты  
  
Pravila\_Start = [] # Список для стартовых правил  
Pravila\_End = [] # Список для конечных правил  
for i in range(5, len(s)-1): # Бежим по правилам  
 x = s[i][2:].replace(' ', '') # Убираем пробелы  
 x, y = x.split('->') # Разделяем правила по стрелке  
 Pravila\_Start.append(x) # Добавляем в нужный список  
 Pravila\_End.append(y) # Добавляем в нужный список  
  
def Type1(Pravila\_Start, Pravila\_End):  
 for i in range(len(Pravila\_Start)):  
 if len(Pravila\_Start[i]) >= len(Pravila\_End[i]): # Сравниваем длину стартогого и конечного правила  
 return False  
 return True  
  
def Type2(Pravila\_Start):  
 for i in Pravila\_Start:  
 for j in i:  
 if j not in Alf\_Not\_Term: # Проверка на принадлежность нетерменантам  
 return False  
 return True  
  
def Type3(Pravila\_End):  
 for i in range(len(Pravila\_End)-1):  
 if Alignment(Pravila\_End[i]) != Alignment(Pravila\_End[i+1]): # Проверка соседних правил на равность  
 return False  
 if Alignment(Pravila\_End[0]) == "Не является 3 типом": # Смотрим первый элемент  
 return False  
 return True  
  
if Type3(Pravila\_End):  
 print("Грамматика принадлежит 3 типу")  
 print(Alignment(Pravila\_End[0]))  
elif Type2(Pravila\_Start):  
 print("Грамматика принадлежит 2 типу")  
elif Type1(Pravila\_Start, Pravila\_End):  
 print("Грамматика принадлежит 1 типу")  
else:  
 print("Грамматика принадлежит 0 типу")  
file.close()