МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**Лабораторная работа №3**

по дисциплине: Теория автоматов и формальных языков

тема: «Регулярные языки и конечные распознаватели»

Выполнил: ст. группы ПВ-201

Морозов Данила Александрович

Проверил:

Рязанов Юрий Дмитриевич

Белгород 2022 г.

**Лабораторная работа №3**

**«Регулярные языки и конечные распознаватели»**

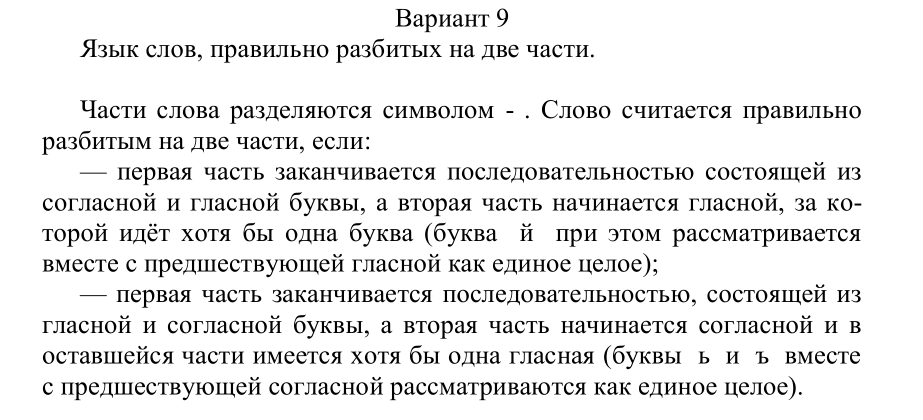
# Цель работы:

Изучить основные способы задания регулярных языков, способы построения, алгоритмы преобразования, анализа и реализации конечных распознавателей.

# Задания к работе:

1. Построить минимальный детерминированный конечный распознаватель заданного языка.
2. Написать программу-распознаватель компиляционного и интерпретационного типа.
3. Написать программу, которая оставляет в исходном текстовом файле только те строки, которые представляют собой цепочки заданного языка.
4. Написать программу, которая исключает из исходного текстового файла строки, являющиеся цепочками заданного языка.

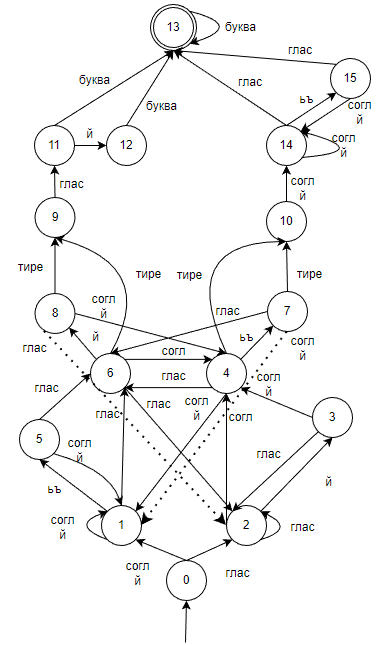
# Задание варианта:



# Выполнение работы:

Задание №1:

Введем следующие обозначения:

* согл – согласные (без «й»)
* глас – гласные
* буква – любая буква: гласная, согласная, твердый/мягкий знак
* тире – символ «-»

Данный распознаватель является детерминированным.

Отметим, что графическое представление вышло довольно сложным, т.к. условия задачи варианта приводят к тому, что мы имеем несколько вариантов путей во многих случаях.

Например, т.к. «й» рассматривается с гласной как единое целое – то везде, куда можно попасть из гласной – можно попасть также из комбинации гласная + «й», при этом необходимо учитывать, что следующая «й» уже будет считаться согласной сама по себе и т.п.

Две длинные стрелки внизу схемы сделаны пунктиром для улучшения видимости.

Табличный вид

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |  |  |
|  | S0 | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 | S9 | S10 | S11 | S12 | S13 | S14 | S15 |
| глас | S2 | S6 | S2 | S2 | S6 | S6 | S2 | S6 | S2 | S11 |  |  |  |  | S13 | S13 |
| согл | S1 | S1 | S4 | S4 | S1 | S1 | S4 | S1 | S4 |  | S14 |  |  |  | S14 | S14 |
| й | S1 | S1 | S3 | S4 | S1 | S1 | S8 | S1 | S4 |  | S14 | S12 |  |  | S14 | S14 |
| ьъ |  | S5 |  |  | S7 |  |  |  |  |  |  |  |  |  | S15 |  |
| буква |  |  |  |  |  |  |  |  |  |  |  | S13 | S13 | S13 |  |  |
| тире |  |  |  |  | S10 |  | S9 | S10 | S9 |  |  |  |  |  |  |  |

Произведем минимизацию.

Для начала определим, что все вершины достижимы – произведем обход в глубину графа состояний:

Все вершины достижимы.

Произведем поиск и замену эквивалентных между собой состояний одним состоянием.

Воспользуемся нахождением классов эквивалентных состояний, которые станут состояниями нашего минимального распознавателя.

Последний столбец справа соответствует состоянию ошибки.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |  |  |  |
|  | S0 | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 | S9 | S10 | S11 | S12 | S13 | S14 | S15 |  |
| глас | S2 | S6 | S2 | S2 | S6 | S6 | S2 | S6 | S2 | S11 |  |  |  |  | S13 | S13 |  |
| согл | S1 | S1 | S4 | S4 | S1 | S1 | S4 | S1 | S4 |  | S14 |  |  |  | S14 | S14 |  |
| й | S1 | S1 | S3 | S4 | S1 | S1 | S8 | S1 | S4 |  | S14 | S12 |  |  | S14 | S14 |  |
| ьъ |  | S5 |  |  | S7 |  |  |  |  |  |  |  |  |  | S15 |  |  |
| буква |  |  |  |  |  |  |  |  |  |  |  | S13 | S13 | S13 |  |  |  |
| тире |  |  |  |  | S10 |  | S9 | S10 | S9 |  |  |  |  |  |  |  |  |

Отвергающие состояния:

Объединим их в класс K1, а допускающие в класс K2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | K1 | | | | | | | | | | | | | K2 | K1 | | |
|  | S0 | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 | S9 | S10 | S11 | S12 | S13 | S14 | S15 |  |
| глас | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K2 | K2 | K1 |
| согл | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 |
| й | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 |
| ьъ | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 |
| буква | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K2 | K2 | K2 | K1 | K1 | K1 |
| тире | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | K1 | | | | | | | | | | | K3 | | K2 | K4 | | K1 |
|  | S0 | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 | S9 | S10 | S11 | S12 | S13 | S14 | S15 |  |
| глас | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K2 | K2 | K1 |
| согл | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K4 | K4 | K1 |
| й | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K3 | K1 | K1 | K4 | K4 | K1 |
| ьъ | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K4 | K1 | K1 |
| буква | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K2 | K2 | K2 | K1 | K1 | K1 |
| тире | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | K1 | | | | | | | | | | | K6 | K3 | K2 | K5 | K4 | K1 |
|  | S0 | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 | S9 | S10 | S11 | S12 | S13 | S14 | S15 |  |
| глас | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K6 | K1 | K1 | K1 | K1 | K2 | K2 | K1 |
| согл | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K5 | K1 | K1 | K1 | K5 | K5 | K1 |
| й | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K5 | K3 | K1 | K1 | K5 | K5 | K1 |
| ьъ | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K4 | K1 | K1 |
| буква | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K2 | K2 | K2 | K1 | K1 | K1 |
| тире | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | K1 | | | | | | | | | K8 | K7 | K6 | K3 | K2 | K5 | K4 | K1 |
|  | S0 | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 | S9 | S10 | S11 | S12 | S13 | S14 | S15 |  |
| глас | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K6 | K1 | K1 | K1 | K1 | K2 | K2 | K1 |
| согл | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K5 | K1 | K1 | K1 | K5 | K5 | K1 |
| й | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K5 | K3 | K1 | K1 | K5 | K5 | K1 |
| ьъ | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K4 | K1 | K1 |
| буква | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K2 | K2 | K2 | K1 | K1 | K1 |
| тире | K1 | K1 | K1 | K1 | K7 | K1 | K8 | K7 | K8 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | K1 | | | | K10 | K1 | K9 | K10 | K9 | K8 | K7 | K6 | K3 | K2 | K5 | K4 | K1 |
|  | S0 | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 | S9 | S10 | S11 | S12 | S13 | S14 | S15 |  |
| глас | K1 | K19 | K1 | K1 | K9 | K9 | K1 | K9 | K1 | K6 | K1 | K1 | K1 | K1 | K2 | K2 | K1 |
| согл | K1 | K1 | K10 | K10 | K1 | K1 | K10 | K1 | K10 | K1 | K5 | K1 | K1 | K1 | K5 | K5 | K1 |
| й | K1 | K1 | K1 | K10 | K1 | K1 | K9 | K1 | K10 | K1 | K5 | K3 | K1 | K1 | K5 | K5 | K1 |
| ьъ | K1 | K1 | K1 | K1 | K10 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K4 | K1 | K1 |
| буква | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K2 | K2 | K2 | K1 | K1 | K1 |
| тире | K1 | K1 | K1 | K1 | K7 | K1 | K8 | K7 | K8 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | K17 | K16 | K15 | K14 | K12 | K13 | K11 | K10 | K9 | K8 | K7 | K6 | K3 | K2 | K5 | K4 | K1 |
|  | S0 | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 | S9 | S10 | S11 | S12 | S13 | S14 | S15 |  |
| глас | K15 | K11 | K15 | K15 | K11 | K11 | K15 | K11 | K15 | K6 | K1 | K1 | K1 | K1 | K2 | K2 | K1 |
| согл | K16 | K16 | K12 | K12 | K16 | K16 | K12 | K16 | K12 | K1 | K5 | K1 | K1 | K1 | K5 | K5 | K1 |
| й | K16 | K16 | K14 | K12 | K16 | K16 | K9 | K16 | K12 | K1 | K5 | K3 | K1 | K1 | K5 | K5 | K1 |
| ьъ | K1 | K13 | K1 | K1 | K10 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K4 | K1 | K1 |
| буква | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K2 | K2 | K2 | K1 | K1 | K1 |
| тире | K1 | K1 | K1 | K1 | K7 | K1 | K8 | K7 | K8 | K1 | K1 | K1 | K1 | K1 | K1 | K1 | K1 |

Изначальный распознаватель был минимальным.

Задание №2:

Программа компиляционного типа:

*def* is\_consonant(*c* : str) -> bool:

    return *c*.lower() in 'бвгджзклмнпрстфхцчшщ'

*def* is\_vowel(*c* : str) -> bool:

    return *c*.lower() in 'аяуюоеёэиы'

*def* is\_consonant\_full(*c* : str) -> bool:

    return *c*.lower() in 'бвгджзклмнпрстфхцчшщй'

*def* is\_sym(*c* : str) -> bool:

    return *c*.lower() in 'бвгджзклмнпрстфхцчшщйаяуюоеёэиы'

*def* report\_error(*state\_num* : int, *c* : str) -> bool:

    print('{0}Error. S = {1}, character = <{2}>, {3}'.format("\033[31m", *state\_num*, *c*, "\033[0m"))

    return True, *c*

*def* is\_sh(*c* : str):

    return *c*.lower() in 'ьъ'

*class* ParseResult:

*def* \_\_init\_\_(*self*, *is\_right*, *state*, *character*) -> None:

*self*.is\_right = *is\_right*

*self*.state = *state*

*self*.character = *character*

*def* is\_splitted\_right(*line* : str) -> ParseResult:

*line* = *line*.strip()

    S = 0

    err = True

    char = ''

    for c in *line*:

        match S:

            case 0:

                if is\_consonant\_full(c):

                    S = 1

                elif is\_vowel(c):

                    S = 2

                else:

                    err, char = report\_error(S, c)

                    break

            case 1:

                if is\_consonant\_full(c):

                    S = 1

                elif is\_sh(c):

                    S = 5

                elif is\_vowel(c):

                    S = 6

                else:

                    err, char = report\_error(S, c)

                    break

            case 2:

                if is\_vowel(c):

                    S = 2

                elif c == 'й':

                    S = 3

                elif is\_consonant(c):

                    S = 4

                else:

                    err, char = report\_error(S, c)

                    break

            case 3:

                if is\_vowel(c):

                    S = 2

                elif is\_consonant\_full(c):

                    S = 4

                else:

                    err, char = report\_error(S, c)

                    break

            case 4:

                if is\_vowel(c):

                    S = 6

                elif is\_sh(c):

                    S = 7

                elif c == '-':

                    S = 10

                elif is\_consonant\_full(c):

                    S = 1

                else:

                    err, char = report\_error(S, c)

                    break

            case 5:

                if is\_consonant\_full(c):

                    S = 1

                elif is\_vowel(c):

                    S = 6

                else:

                    err, char = report\_error(S, c)

                    break

            case 6:

                if is\_consonant(c):

                    S = 4

                elif is\_vowel(c):

                    S = 2

                elif c == 'й':

                    S = 8

                elif c == '-':

                    S = 9

                else:

                    err, char = report\_error(S, c)

                    break

            case 7:

                if c == '-':

                    S = 10

                elif is\_vowel(c):

                    S = 6

                elif is\_consonant\_full(c):

                    S = 1

                else:

                    err, char = report\_error(S, c)

                    break

            case 8:

                if is\_consonant\_full(c):

                    S = 4

                elif is\_vowel(c):

                    S = 2

                elif c == '-':

                    S = 9

                else:

                    err, char = report\_error(S, c)

                    break

            case 9:

                if is\_vowel(c):

                    S = 11

                else:

                    err, char = report\_error(S, c)

                    break

            case 10:

                if is\_consonant\_full(c):

                    S = 14

                else:

                    err, char = report\_error(S, c)

                    break

            case 11:

                if c == 'й':

                    S = 12

                elif is\_sym(c):

                    S = 13

                else:

                    err, char = report\_error(S, c)

                    break

            case 12:

                if is\_sym(c):

                    S = 13

                else:

                    err, char = report\_error(S, c)

                    break

            case 13:

                if is\_sym(c):

                    S = 13

                    err = False

                else:

                    err, char = report\_error(S, c)

                    break

            case 14:

                if is\_consonant\_full(c):

                    S = 14

                elif is\_sh(c):

                    S = 15

                elif is\_vowel(c):

                    S = 13

                else:

                    err, char = report\_error(S, c)

                    break

            case 15:

                if is\_vowel(c):

                    S = 13

                elif is\_consonant\_full(c):

                    S = 14

                else:

                    err, char = report\_error(S, c)

                    break

    return ParseResult(not err, S, char)

*def* main():

    word = 'прьий-евет'

    print('Разбираемое слово <{0}>'.format(word))

    res = is\_splitted\_right(word)

    if res.is\_right:

        print("Слово разбито верно")

    else:

        print("Слово разбито неверно")

    if not res.is\_right:

        print("Распознаватель был в состоянии S{0}, символ, вызвавший ошибку <{1}>.".format(res.state, res.character))

if \_\_name\_\_ == "\_\_main\_\_":

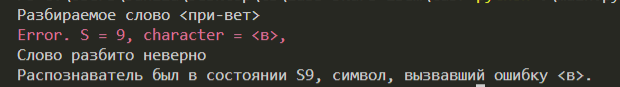
    main()

Результаты работы:



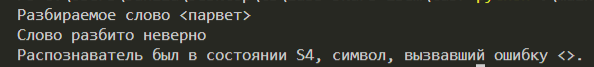


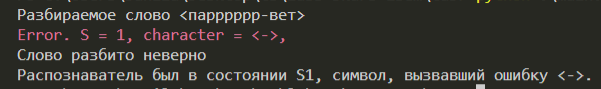






Т.к. обращений к несуществующим переходам не было – то символа, вызвавшего ошибку, нет – то слово все равно разбито неверно, т.к. конечное состояние не является допускающим.







Программа интерпретационного типа:

*def* is\_consonant(*c* : str) -> bool:

    return *c*.lower() in 'бвгджзклмнпрстфхцчшщ'

*def* is\_vowel(*c* : str) -> bool:

    return *c*.lower() in 'аяуюоеёэиы'

*def* is\_consonant\_full(*c* : str) -> bool:

    return *c*.lower() in 'бвгджзклмнпрстфхцчшщй'

*def* is\_sym(*c* : str) -> bool:

    return *c*.lower() in 'бвгджзклмнпрстфхцчшщйаяуюоеёэиы'

*def* report\_error(*state\_num* : int, *c* : str) -> bool:

    print('{0}Error. S = {1}, character = <{2}>, {3}'.format("\033[31m", *state\_num*, *c*, "\033[0m"))

    return True, *c*

*def* is\_sh(*c* : str) -> bool:

    return *c*.lower() in 'ьъ'

*def* get\_index(*c* : str) -> int:

    if is\_vowel(*c*):

        return 0

    elif is\_consonant(*c*):

        return 1

    elif *c* == 'й':

        return 2

    elif is\_sh(*c*):

        return 3

    elif *c* == '-':

        return 4

    else:

        raise(Exception("Wrong index"))

*class* ParseResult:

*def* \_\_init\_\_(*self*, *is\_right*, *state*, *character*) -> None:

*self*.is\_right = *is\_right*

*self*.state = *state*

*self*.character = *character*

*class* Parser:

*def* \_\_init\_\_(*self*, *fileName*) -> None:

*self*.matr = []

*self*.S = 0

*self*.status = ParseResult(False, -100, '')

        with open(*fileName*, "r") as f:

            lines = f.readlines()

            for i in range(len(lines)):

*self*.matr.append([])

                line = lines[i].strip()

                states = line.split(' ')

                for state in states:

*self*.matr[i].append(int(state) if state != '.' else -100)

*def* parse(*self*, *str*) -> ParseResult:

        for c in *str*:

            new\_state = *self*.matr[get\_index(c)][*self*.S]

*self*.status.state = *self*.S

*self*.status.character = c

            if new\_state != -100:

*self*.S = new\_state

                if new\_state == 13:

*self*.status.is\_right = True

            else:

                report\_error(*self*.S, c)

*self*.status.is\_right = False

                return

*self*.status.character = ''

*def* main():

    word = 'прьий-евет'

    print('Разбираемое слово <{0}>'.format(word))

    parser = Parser("table.txt")

    parser.parse(word)

    res = parser.status

    if res.is\_right:

        print("Слово разбито верно")

    else:

        print("Слово разбито неверно")

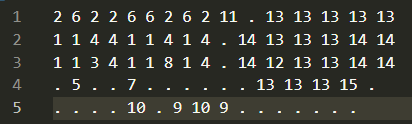
    if not res.is\_right:

        print("Распознаватель был в состоянии S{0}, символ, вызвавший ошибку <{1}>.".format(res.state, res.character))

if \_\_name\_\_ == "\_\_main\_\_":

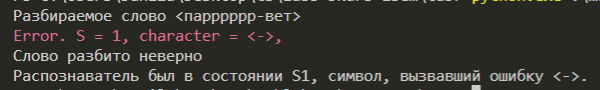
    main()

Таблица переходов, немного модифицированная под интерпретационный вариант:



Результаты работы:













Задание №3:

from main import is\_splitted\_right

*def* main():

    fileName = input("Введите имя файла: ")

    good = []

    lines = []

    with open(fileName, "r") as f:

        lines = f.readlines()

        for i in range(len(lines)):

            if is\_splitted\_right(lines[i]).is\_right:

                good.append(i)

    with open(fileName,"w") as f:

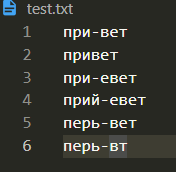
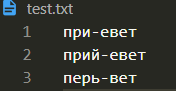
        for i in good:

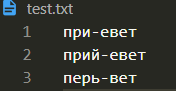
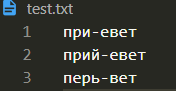
            f.write(lines[i])

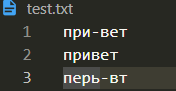
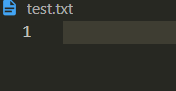
if \_\_name\_\_ == "\_\_main\_\_":

    main()

Результаты работы:

Задание №4:

from main import is\_splitted\_right

*def* main():

    fileName = input("Введите имя файла: ")

    good = []

    lines = []

    with open(fileName, "r") as f:

        lines = f.readlines()

        for i in range(len(lines)):

            if not is\_splitted\_right(lines[i]).is\_right:

                good.append(i)

    with open(fileName,"w") as f:

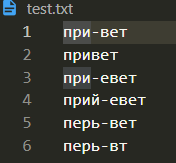
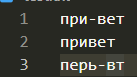
        for i in good:

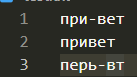
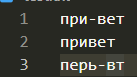
            f.write(lines[i])

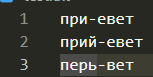
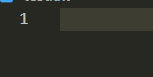
if \_\_name\_\_ == "\_\_main\_\_":

    main()

Результаты работы:

**Вывод:**

Мы изучили основные способы задания регулярных языков, способы построения, алгоритмы преобразования, анализа и реализации конечных распознавателей.