Министерство образования и науки российской федерации

(минобрнауки россии)

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Санкт-Петербургский государственный политехнический университет» (ФГБОУ ВПО «СПбГПУ»)

**Институт менеджмента и информационных технологий**

(филиал)федерального государственного бюджетного образовательного учреждения высшего профессионального образования

«Санкт-Петербургский государственный политехнический университет» в г. Череповце (ИМИТ «СПбГПУ»)

Кафедра ПО ВТ и АС

ЛАБОРАТОРНАЯ РАБОТА №2

Дисциплина: «Теория языков программирования и методы трансляции»

Тема: «Синтаксический анализ. Автомат с магазинной памятью»

Выполнил студент группы о.291 Шанин Игнат Леонидович

№ зачетной книжки о2080127

Проверил Михайлов Андрей Евгеньевич

«\_\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_201\_\_ г.

\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

отметка о зачете подпись преподавателя

г. Череповец

2012

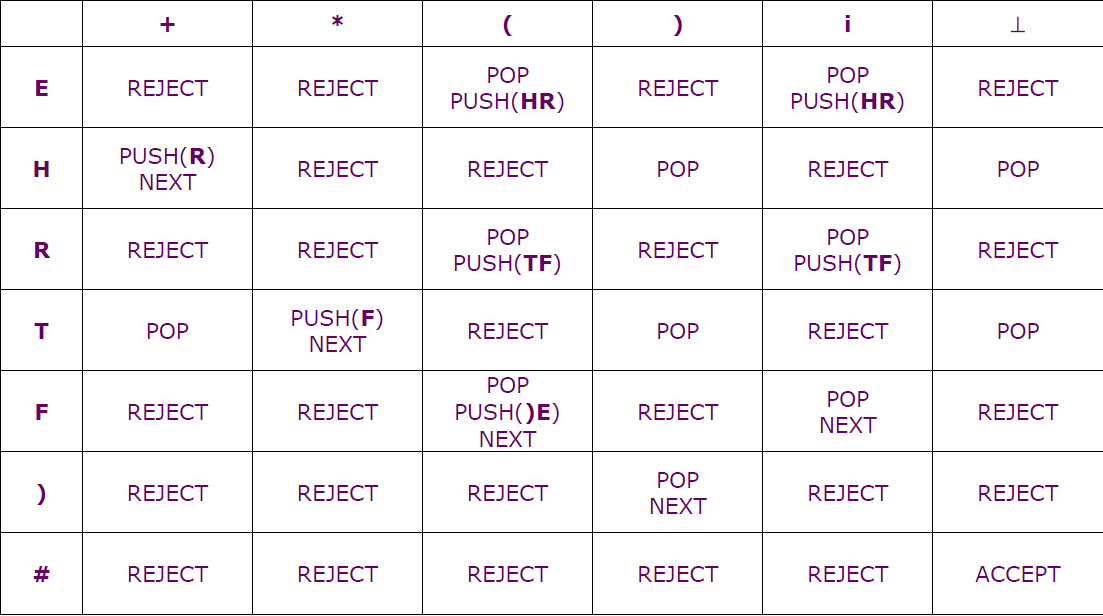
# Задание

На основе табличного описания реализовать автомат с магазинной памятью (МП-автомат) с одним состоянием. Программа должна принимать на вход текстовый файл со строками для анализа и генерировать на выходе файл, только с теми строками, которые были корректно распознаны анализатором.

МП-автомат:

M({+,\*,(,),i, ┴}, {q0}, {E,H,R,T,F,)}, s, q0, E, {q0})

Таблица переходов МП-автомата:



# Краткая теоретическая справка

## Общие сведения

Контекстно-свободными называются грамматики вида  правила , которых имеют вид:

, где .

Контекстно-свободные грамматики традиционно служат основой для синтаксического анализа компиляции. Описание входного языка включает пра­вила синтаксиса, в соответствии с которыми порождаются тексты программ. Каждому синтаксическому правилу в КС-грамматике соответствует правило вывода (продукция) грамматики. Поэтому последовательность правил вывода грамматики, которые применяются для порождения текста программы и опре­деляет ее структуру.

Вывод в КС-грамматике называют левосторонним(правосторонним)*,* если правила вывода применяются к самому левому (правому) вхождению нетерминального символа каждой цепочки вывода.

Получение синтаксической структуры программы в форме шагов вывода не совсем удобно – в выводе не просматривается явно структура программы. Более наглядным является представление структуры программы в виде дерева вывода (дерева синтаксического разбора).

Под деревом синтаксического разбора будем понимать конечный неориентированный граф, обладающий свойствами:

* у него имеется одна вершина, в которую не входит ни одна дуга; эту вершину будем называть корнем дерева, корень дерева соответ­ствует начальному символу грамматики — аксиоме;
* в каждую из его остальных вершин входит лишь одна дуга; верши­ны, **не** имеющие выходящих из них дуг, называются заключительнымивершинами дерева или листьями*,* в КС-грамматике им соответствуют терминальные символы грамматики; не заключительным вершинам де­рева соответствуют нетерминальные символы грамматики;
* он не содержит контуров.

## Табличный способ задания функции переходов МП-автомата

Множество правил перехода, реализуемых управляющим устройством, можно задать несколькими способами. В за­висимости от получаемой информации, управляющее устройство реализует один такт работы МП-автомата, который включает в себя три типа операции:

1) операции над магазином:

* втолкнуть в магазин определенный символ (ВТОЛКНУТЬ (А),
  + где А — магазинный символ);
* вытолкнуть верхний символ из магазина (ВЫТОЛКНУТЬ);
* оставить содержимое магазина без изменений;

2) операции над состоянием:

* перейти в заданное новое состояние (СОСТОЯНИЕ (S), где S — следующее
  + состояние);
* остаться в прежнем состоянии;

3) операции над входом:

* перейти к следующему входному символу и сделать его текущим (СДВИГ);
* оставить данный входной символ текущим, т.е. держать его до следующего шага (ДЕРЖАТЬ).

Так же к командам можно отнести состояние выходов автомата, называемых ДОПУСТИТЬ и ОТВЕРГНУТЬ. В этом случае автомат часто называют МП-распознавателем.

Описание функции переходов можно задать несколькими способами.

Рассмотрим **табличный способ задания** на примере.

Зададим МП-автомат, распознающий слова языка , т.е. слова, состоящие из n нулей, вслед за которыми идет ровно такое же коли­чество единиц. Для этого автомата , – символ конца цепочки на входной ленте, # — маркер дна магазина,  — на­чальное состояние, # — начальное содержимое магазина. Работа управляюще­го устройства для данного автомата осуществляется в соответствии с управляющими таблицами для состояний q, и q, соответственно. В таблицах показаны действия автомата для каждого сочетания входного символа и верхнего символа магазина.

Состояние q1:

|  |  |  |  |
| --- | --- | --- | --- |
| Магазин | Вход | | |
| 0 | 1 | ⊥ |
| Z | СОСТОЯНИЕ(q1)  ВТОЛКНУТЬ(z)  СДВИГ | СОСТОЯНИЕ(q2)  ВЫТОЛКНУТЬ  СДВИГ | ОТВЕРГНУТЬ |
| # | СОСТОЯНИЕ(q1)  ВТОЛКНУТЬ(z)  СДВИГ | ОТВЕРГНУТЬ | ОТВЕРГНУТЬ |

Состояние q2:

|  |  |  |  |
| --- | --- | --- | --- |
| Магазин | Вход | | |
| 0 | 1 | ⊥ |
| Z | ОТВЕРГНУТЬ | СОСТОЯНИЕ(q2)  ВЫТОЛКНУТЬ  СДВИГ | ОТВЕРГНУТЬ |
| # | ОТВЕРГНУТЬ | ОТВЕРГНУТЬ | ДОПУСТИТЬ |

# Листинг программы

MP\_TABLE = {

    '\_' : [  '+',      '\*',      '(',      ')',      'i',       '|'  ],

    'E' : ['REJECT', 'REJECT', 'PPHR'  , 'REJECT', 'PPHR'  , 'REJECT'],

    'H' : ['PRN'   , 'REJECT', 'REJECT', 'POP'   , 'REJECT', 'POP'   ],

    'R' : ['REJECT', 'REJECT', 'PPTF'  , 'REJECT', 'PPTF'  , 'REJECT'],

    'T' : ['POP'   , 'PFN'   , 'REJECT', 'POP'   , 'REJECT', 'POP'   ],

    'F' : ['REJECT', 'REJECT', 'PP)EN' , 'REJECT', 'PN'    , 'REJECT'],

    ')' : ['REJECT', 'REJECT', 'REJECT', 'PN'    , 'REJECT', 'REJECT'],

    '#' : ['REJECT', 'REJECT', 'REJECT', 'REJECT', 'REJECT', 'ACCEPT'],

}

ACTIONS = {

    'PPHR': **lambda** s, i: (s[:-1] + 'HR', i),

     'POP': **lambda** s, i: (s[:-1], i),

     'PRN': **lambda** s, i: (s + 'R', i[1:]),

    'PPTF': **lambda** s, i: (s[:-1] + 'TF', i),

     'PFN': **lambda** s, i: (s + 'F', i[1:]),

   'PP)EN': **lambda** s, i: (s[:-1] + ')E', i[1:]),

      'PN': **lambda** s, i: (s[:-1], i[1:]),

      'PN': **lambda** s, i: (s[:-1], i[1:])

}

**class** SSPDA:

    """Single state pushdown automaton"""

**def** \_\_init\_\_(self, table=MP\_TABLE, actions=ACTIONS):

        self.\_stack = "#E"

        self.\_table = dict(table)

        self.\_symbols = self.\_table.pop('\_')

        self.\_actions = actions

**def** reset(self):

        self.\_stack = "#E"

**def** check\_chain(self, chain):

        chain += '|'

        self.reset()

**while** True:

            action = self.\_table[self.\_stack[-1]][self.\_symbols.index(chain[0])]

**if** action == 'REJECT':

**return** False

**if** action == 'ACCEPT':

**return** True

**if** len(self.\_stack) > 9000:

**raise** OverflowError()

            self.\_stack, chain = self.\_actions[action](self.\_stack, chain)

**if** \_\_name\_\_ == '\_\_main\_\_':

    A = SSPDA()

**with** open("test.txt") **as** f:

        chains = f.read().replace(" ", "").split('**\n**')

    accepted = []

    rejected = []

**for** c **in** chains:

**if** A.check\_chain(c):

            accepted.append(c)

**else**:

            rejected.append(c)

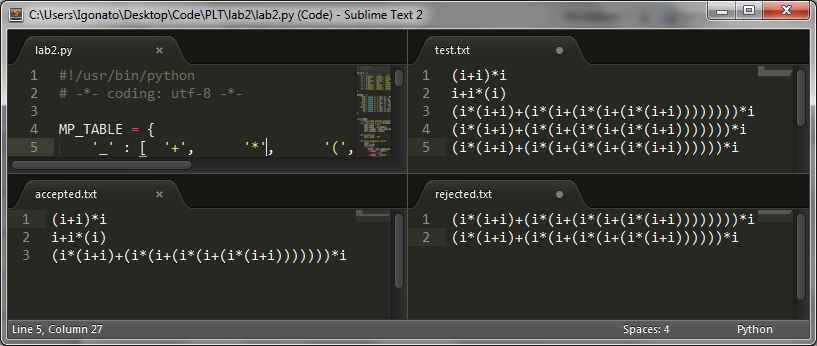
**with** open("accepted.txt", "w") **as** f:

        f.write('**\n**'.join(accepted))

**with** open("rejected.txt", "w") **as** f:

        f.write('**\n**'.join(rejected))

# Результаты работы программы



# Вывод

В результате проделанной работы был написан скрипт, реализующий автомат с магазинной памятью. Входом является текстовый файл, содержащий набор строк. Наша программа либо генерирует файлы: со строками, которые были корректно распознаны, и с теми, в ходе разбора которых возникла ошибка.