Санкт-Петербургский национальный исследовательский университет

информационных технологий, механики и оптики

**Лабораторная работа №1**

**Дисциплина «Основы разработки компиляторов»**

**Вариант 12**

**Выполнил:**

Съестов Дмитрий Вячеславович

Группа P3317

**Преподаватель:**

Логинов Иван Павлович

Санкт-Петербург

2019

Задание:

1. Построить матрицу простого предшествования.
2. Написать программу, реализующую распознаватель.

Грамматика:

S → AS | AB

A → aA | bA |c

B → dB | a | ε

Данная грамматика не является грамматикой простого предшествования, т.к. в ней присутствуют конфликты. Для их разрешения введём нетерминал Z. Также устраним эпсилон-правило.

S → ZS | ZB | Z

Z → A

A → aA | bA |c

B → dB | a | d

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | S | Z | A | B | a | b | c | d | $ |
| S |  |  |  |  |  |  |  |  | > |
| Z | = | < | < | = | < | < | < | < | > |
| A | > | > | > | > | > | > | > | > | > |
| B |  |  |  |  |  |  |  |  | > |
| a |  |  | = |  | < | < | < |  | > |
| b |  |  | = |  | < | < | < |  |  |
| c | > | > | > | > | > | > | > | > | > |
| d |  |  |  | = | < |  |  | < | > |
| $ |  | < | < |  | < | < | < |  |  |

Получена матрица простого предшествования.

Основная функция программы:

def parse(self, input):

if input == '':

return False

input += MARKER

stack = [MARKER]

try:

while not (input == MARKER and stack == [MARKER, STARTING\_SYMBOL]):

top, nextToken = stack[-1], input[0]

relationship = self.getRelationship(top, nextToken)

if relationship == '>': # Reduce

self.fullReduce(stack)

else: # Shift

stack.append(relationship)

stack.append(nextToken)

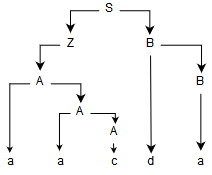
input = input[1:]

except ValueError as err:

return False

return True

Пример ручного разбора цепочки ‘aacda’:



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Шаг** | **Обработанная цепочка** | **Отношение** | **Необработанная цепочка** | **Действие** | **Правило** |
| 0 | $ | < | aacda$ | Сдвиг |  |
| 1 | $a | < | acda$ | Сдвиг |  |
| 2 | $aa | < | cda$ | Сдвиг |  |
| 3 | $aac | > | da$ | Свёртка | A → c |
| 4 | $aaA | > | da$ | Свёртка | A → aA |
| 5 | $aA | > | da$ | Свёртка | A → aA |
| 6 | $A | > | da$ | Свёртка | A → Z |
| 7 | $Z | < | da$ | Сдвиг |  |
| 8 | $Zd | < | a$ | Сдвиг |  |
| 9 | $Zda | > | $ | Свёртка | B → a |
| 10 | $ZdB | > | $ | Свёртка | B → dB |
| 11 | $ZB | > | $ | Свёртка | S → ZB |
| 12 | $S | > | $ | Accept |  |

**Вывод:** в ходе выполнения данной лабораторной работы был реализован восходящий распознаватель, проверяющий цепочку на принадлежность к грамматике простого предшествования. Для этого используется таблица отношений между символами, позволяющая определять основу и выполнять свёртку. Если таким образом удаётся получить стартовый нетерминал, то цепочка принадлежит грамматике.