Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчет

По дисциплине: “Теоретические основы построения компиляторов”

Лабораторная работа №3

“ИССЛЕДОВАНИЕ АЛГОРИТМА НИСХОДЯЩЕГО РАСПОЗНАВАТЕЛЯ ЯЗЫКА LL(к) ГРАММАТИКИ”

Выполнил:

ст.гр. ИС/б-20-1-о

Галенин А. К.

Проверил:

Карлусов В.Ю.

Севастополь

2023

1 ЦЕЛЬ РАБОТЫ

Изучить процедуру синтаксического анализа, основанную на осуществлении рекурсивного спуска. Приобрести навыки в исследовании грамматики на принадлежность к классу LL(к). Освоить методы синтеза формальных грамматик класса LL(к) и, по необходимости, преобразования грамматики к этому классу

2 ПОСТАНОВКА ЗАДАЧИ

Вариант 11 – 1.1.15

Таблица 1 – Фрагмент программы для анализа

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № вар. | Служебные слова | Разделители | | Фрагмент программы для анализа |
| однолитерные | двули-терные |
| 15 | WAIT  SIGNAL | (  )  >  =  +  % | >=  <= | WAIT(S)  K=K+5.3 %  C=K+C+1 %  SIGNAL(S<=0xf) |

3 ХОД РАБОТЫ

1. Построение формальной грамматики.

<фрагмент> ::= WAIT(<iden>) <тело программы>

<тело программы> ::= <оператор><тело программы>

<выражение> ::= <iden><знак><операнд>

<тело программы> ::= SIGNAL(<выражение>)

<оператор> :: = <iden> = <iden><знак><операнд> <продолжение>

<продолжение> ::= %

<операнд> ::= <iden>

<операнд> ::= <data>

<знак> ::= +

<знак> ::= <=

<знак> ::= >=

2. Было построено синтаксическое дерево

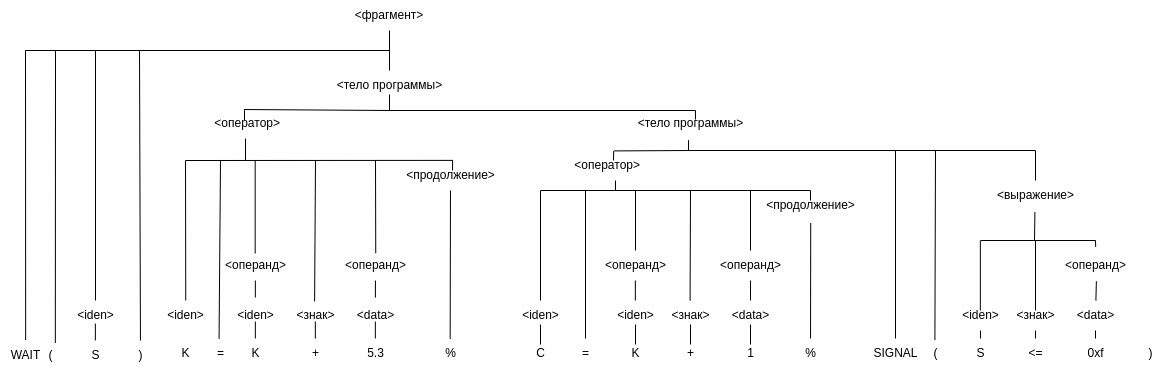


Рисунок 1 – Синтаксическое дерево

Элементами разработанной грамматики являются:

1) словарь нетерминалов Vn ={фрагмент, идентификатор, аргумент, тело программы, оператор, продолжение, операнд, знак, выражение };

2) словарь терминалов Vt = { WAIT, SIGNAL, (, ), +, %, =, <iden>, <data>};

3) аксиома грамматики – нетерминальный символ <фрагмент>;

Таблица 2 – Правильные и ошибочные тестовые последовательности

|  |  |
| --- | --- |
| Тестовая последовательность | Результат обработки |
| 100 501 400 502 300 504 300 505 400 506 300 504 300 505 400 506 200 501 400 502 | Цепочка верна |
| 100 501 400 502 200 501 400 502 | Цепочка верна |
| 100 501 400 502 300 504 300 300 | Ожидался знак с операндом или разделитель Синтаксическая ошибка |
| 200 503 400 300 506 200 | Ожидалось служебное слово WAIT Синтаксическая ошибка |

3. Код программы:

def s\_equal(test\_sequence, i):

if test\_sequence[i] == 504:

return 1

else:

print('\nОжидался знак равенства')

return 0

def s\_iden(test\_sequence, i):

if test\_sequence[i] == 300:

return 1

else:

print('\nОжидался идентификатор')

return 0

def s\_data(test\_sequence, i):

if test\_sequence[i] == 400:

return 1

else:

print('\nОжидалась константа')

return 0

def s\_operand(test\_sequence, i):

if test\_sequence[i] == 300:

return s\_iden(test\_sequence, i)

elif test\_sequence[i] == 400:

return s\_data(test\_sequence, i)

else:

print('\nОжидалась константа или переменная')

return 0

def s\_separator(test\_sequence, i):

if test\_sequence[i] == 506:

return 1

else:

print('\nОжидался разделитель %')

return 0

def s\_plus(test\_sequence, i):

if test\_sequence[i] == 505:

return 1

else:

print('\nОжидался знак +')

return 0

def s\_continue(test\_sequence, i):

if test\_sequence[i] == 506 and test\_sequence[i+1] == 300:

k = 2

k += s\_equal(test\_sequence, i + k)

k += s\_iden(test\_sequence, i + k)

k += s\_plus(test\_sequence, i + k)

k += s\_operand(test\_sequence, i + k)

k += s\_separator(test\_sequence, i + k)

k += s\_signal(test\_sequence, i + k)

k += s\_arg(test\_sequence, i + k)

return 1 if k == 9 else 0

if test\_sequence[i] == 506:

return s\_separator(test\_sequence, i)

if test\_sequence[i] == 506 and test\_sequence[i+1] == 200:

k = 0

k += s\_signal(test\_sequence, i + k)

k += s\_arg(test\_sequence, i + k)

return 1 if k == 2 else 0

if test\_sequence[i] == 505:

k = 0

k += s\_plus(test\_sequence, i + k)

k += s\_operand(test\_sequence, i + k)

k += s\_continue(test\_sequence, i + k)

return 1 if k == 3 else 0

print('\nОжидался знак с операндом или разделитель')

return 0

def s\_operator(test\_sequence, i):

if test\_sequence[i] == 300:

k = 1

k += s\_equal(test\_sequence, i + k)

k += s\_operand(test\_sequence, i + k)

k += s\_continue(test\_sequence, i+k)

return 1 if k == 4 else 0

else:

return 1

def s\_signal(test\_sequence, i):

if test\_sequence[i] == 200:

return 1

else:

print('\nНет служебного слова SIGNAL')

return 0

def s\_wait(test\_sequence, i):

if test\_sequence[i] == 100:

return 1

else:

print('\nНет служебного слова WAIT')

return 0

def s\_arg(test\_sequence, i):

if test\_sequence[i] == 501 and test\_sequence[i+1] == 400 and test\_sequence[i+2] == 502:

return 1

else:

print('\nОжидался аргумент')

return 0

def s\_body(test\_sequence, i):

k = 2

if test\_sequence[i+k] == 200:

k += s\_signal(test\_sequence, i + k)

k += s\_arg(test\_sequence, i + k)

return 1 if k == 4 else 0

elif test\_sequence[i+k] == 300:

return s\_operator(test\_sequence, i + k)

else:

print('\nОжидалось начало оператора или служебное слово SIGNAL')

return 0

def s\_fragment(test\_sequence, i):

if test\_sequence[i] == 100:

k = 0

k += s\_wait(test\_sequence, i+k)

k += s\_arg(test\_sequence, i+k)

k += s\_body(test\_sequence, i+k)

return 1 if k == 3 else 0

else:

print('\nОжидалось служебное слово WAIT')

return 0

def main():

test\_data = []

with open('test\_data.txt') as file\_handle:

for line in file\_handle:

test\_data.append([int(x) for x in line.split()])

i = 0

for test in test\_data:

i += 1

print("\nЦепочка №{}:".format(i))

try:

if s\_fragment(test, 0) == 0:

print("\tСинтаксическая ошибка")

else:

print("\tЦепочка верна")

except Exception as e:

print(str(e))

print("\tСинтаксическая ошибка")

main()

ВЫВОДЫ

В ходе выполнения лабораторной работы была изучена процедура синтаксического анализа, основанную на осуществлении рекурсивного спуска. Приобретены навыки в исследовании грамматики на принадлежность к классу LL(к). Освоены методы синтеза формальных грамматик класса LL(к) и, по необходимости, преобразования грамматики к этому классу