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
		одноли терные	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(<аргумент>) <тело программы>

<аргумент> :: = <data>

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

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

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

<продолжение> ::= %

<операнд> ::= <iden>

<операнд> ::= <data>

<знак> ::= +

2. Было построено синтаксическое дерево

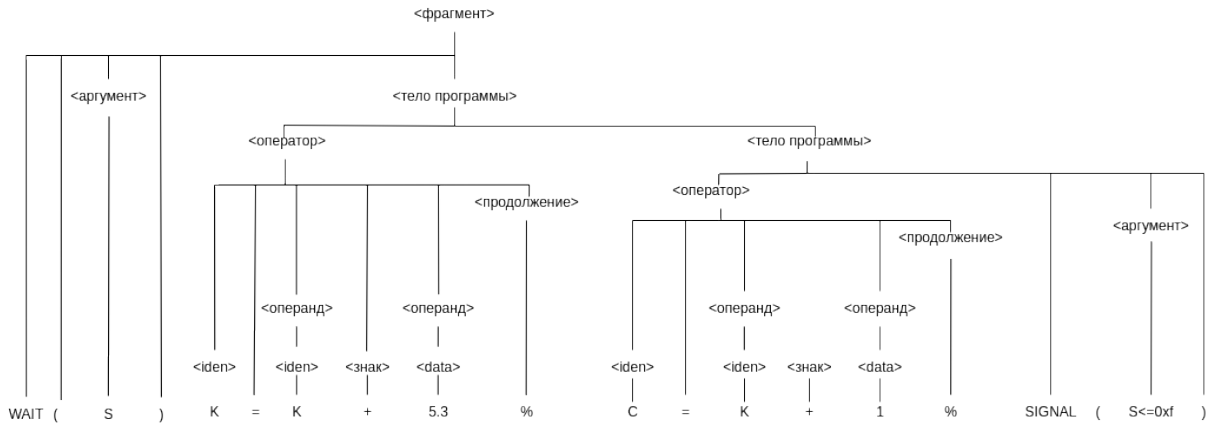

Рисунок 1 – Синтаксическое дерево

Элементами разработанной грамматики являются:

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

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