

Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчет

По дисциплине: “Теоретические основы построения компиляторов”

Лабораторная работа №2

“ИССЛЕДОВАНИЕ СКАНЕРА ПРИ АНАЛИЗЕ ПРОСТЫХ ЯЗЫКОВЫХ
КОНСТРУКЦИЙ”

Выполнил:

ст.гр. ИС/б-20-1-о

Галенин А. К.

Проверил:

Карлусов В.Ю.

Севастополь

2023

1 ЦЕЛЬ РАБОТЫ

Изучить принципы построения и программирования лексического анализатора на языке С (C++) для простых языковых конструкций. Получить навыки практического построения лексического анализатора (сканера) на основе теории конечных автоматов. Освоить приёмы составления регулярных выражений для описания лексем. Закрепить навыки построения минимального КА, осуществляющего сканирование текста программ.

2 ПОСТАНОВКА ЗАДАЧИ

Вариант 11 – 1.1.15

Таблица 1 – Типы идентификаторов

№ варианта	Описание типа
1	Содержит чередующиеся пары букв и цифр, заканчивается последовательностью символов "1"

Таблица 2 – Типы констант

№ варианта	Тип	Пояснения	Формат
1	F	С фиксированной точкой	± 000.000

Таблица 3 – Фрагмент программы для анализа

№ ва р.	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. Были назначены коды лексемам:

WAIT – 100

SIGNAL – 200

Идентификатор, <iden> – 300

Константа, <data> – 400

(– 501

) – 502

> – 503

= – 504

+ – 505

% – 506

>= – 507

<= – 508

2. Построение минимального конечного детерминированного автомата:

1) Служебные слова являются самоопределяющимися цепочками, составленными из литер $\{W, A, I, T, S, G, N, L\}$

2) Эскизно идентификатор (переменная) будет выглядеть так:

$(B \vee C) \{B \vee C\} 1$

Где B – любая латинская буква, C – любая десятичная цифра, 1 – последовательность 1 , которой по условию задания оканчивается имя переменной.

3) Константа с фиксированной точкой описывается следующим образом:

$(+v-)CCC \cdot CCC$

Где C – любая десятичная цифра

4) Множество букв $= \{W, A, I, T, S, G, N, L, \delta\}$

где δ – буква латинского алфавита, не совпадающая по начертанию с W, A, I, T, S, G, N, L .

5) Множество цифр $C = \{0, 2, 3, 4, 5, 6, 7, 8, 9\}$

6) Множество «не буква, не цифра» $L_1 = \{ (,), >, =, +, \%, >=, <=, \cdot \} \cup L_3$ включает, помимо однолитерных разделителей и компонентов двулитерных разделителей, точку, отделяющую целую часть числа от дробной части, и любой символ, не принадлежащий алфавиту.

7) Множество $L_2 = L_1$ – текущая литера не принадлежит к образующим константу или литерал.

8) L_3 – литера, не принадлежащая алфавиту конечного автомата.

Окончательно получаем дизъюнктивные члены выражения, описывающие конечный автомат:

1) $WAITL_1$ – первое служебное слово

2) $SIGNALL_1$ – второе служебное слово

3) $(W \vee A \vee I \vee T \vee S \vee G \vee N \vee L \vee \delta \vee C) \{W \vee A \vee I \vee T \vee S \vee G \vee N \vee L \vee \delta \vee C\} 1L_1$ – переменная

4) $(+v-)CCC \cdot CCCL_2$ – константа

5) $(\vee) \vee > \vee = \vee + \vee \% \vee >= \vee <= \vee L_3$ – однолитерные и двулитерные разделители

разметка.

[illegible]

Рисунок 1 – Разметка регулярного выражения

МИНИМИЗАЦИЯ.

[illegible]

Рисунок 2 – Функция переходов ДКА

[illegible]

Рисунок 3 – Минимизированная таблица переходов ДКА

5. Код программы:

```
import java.io.FileReader;
import java.util.Scanner;
public class Main {
    private static int[][] matrix;
    private static Scanner scanner;
    private static boolean flag = false;
    private static int temp_ch;
    public static void main(String[] args) {
        try{
            int i, n;
            FileReader fr = new FileReader("data.txt");
            matrix = readMatrix(matrix); // управляющая таблица
            i=0; // номер состояния КА
            n=0; // число правильных слов
            String word = new String();
            int ch; // текущая литера
            ch = fr.read();
            while (ch != -1) {
                if (Character.isLetter(ch)){

                    switch(ch) {
                        case 'W':
                            i = matrix[0][i];
                            word += (char) ch;
                            break;
                        case 'A':
                            i = matrix[1][i];
                            word += (char) ch;
                            break;
                        case 'I':
                            i = matrix[2][i];
                            word += (char) ch;
                            break;
                        case 'T':
                            i = matrix[3][i];
                            word += (char) ch;
                            break;
                        case 'S':
                            i = matrix[4][i];
                            word += (char) ch;
                            break;
                        case 'G':
                            i = matrix[5][i];
                            word += (char) ch;
                            break;
                        case 'N':
                            i = matrix[6][i];
                            word += (char) ch;
                            break;
                        case 'L':
                            i = matrix[7][i];
                            word += (char) ch;
                            break;
                        default:
                            i = matrix[8][i];
                            break;
                    }
                }
                else if (Character.isDigit(ch)) {
                    i = matrix[9][i];
                }
            }
            else {
```

```

        switch (ch) {
            case '(':
                i = matrix[11][i];
                word += (char) ch;
                break;
            case ')':
                i = matrix[12][i];
                word += (char) ch;
                break;
            case '>':
                i = matrix[13][i];
                word += (char) ch;
                break;
            case '=':
                i = matrix[14][i];
                word += (char) ch;
                break;
            case '+':
                i = matrix[15][i];
                word += (char) ch;
                break;
            case '%':
                i = matrix[16][i];
                word += (char) ch;
                break;
            case '.':
                i = matrix[17][i];
                word += (char) ch;
                break;
            case '<':
                i = matrix[18][i];
                word += (char) ch;
                break;
            default:
                i = matrix[19][i];
                word += (char) ch;
                break;
        }
    }
    if ((i > 99)&&(i < 500)) {
        /* Анализ кода состояния */
        switch(i){
            case 100:
                System.out.println("Служебное слово 'WAIT'. Состояние =
"+i);
                break;
            case 200:
                System.out.println("Служебное слово 'SIGNAL'. Состояние
= "+i);
                break;
            case 300:
                System.out.println("Завершился идентификатор. Состояние
= "+i);
                break;
            case 400:
                System.out.println("Завершилась константа. Состояние =
"+i);
                break;
        }
        i=0;
        if (ch == '<' || ch == '=' || ch == '>') flag = true;
    }
    else if (i > 500 && i < 800) {
        switch(i) {

```

```

        case 501:
            System.out.println("Разделитель '(' . Состояние = " + i);
            break;
        case 502:
            System.out.println("Разделитель ')' . Состояние = " + i);
            break;
        case 503:
            System.out.println("Разделитель '>' . Состояние = " + i);
            break;
        case 504:
            System.out.println("Разделитель '=' . Состояние = " + i);
            break;
        case 505:
            System.out.println("Разделитель '+' . Состояние = " + i);
            break;
        case 506:
            System.out.println("Разделитель '%' . Состояние = " + i);
            break;
        case 507:
            System.out.println("Разделитель '>=' . Состояние = " + i);
            break;
        case 508:
            System.out.println("Разделитель '<=' . Состояние = " + i);
            break;
        default:
            System.out.println("Ошибка "+i);
    }
    i=0;
}
else if (i > 800) {
    switch(i) {
        case 801:
            System.out.println("Неправильное начало");
            break;
        case 802:
            System.out.println("Ошибка в служебном слове");
            break;
        case 803:
            System.out.println("Ошибка в написании имени переменной");
        case 804:
            System.out.println("Ошибочная константа");
        case 805:
            System.out.println("Ошибка");
        default:
            System.out.println("Неизвестная входная литера");
    }
    i=0;
    ch = fr.read();
}
if (!flag){
    ch = fr.read();
}else{
    flag = false;
}
}
}
catch (Exception e){
    e.printStackTrace();
}
}

public static int[][] readMatrix(int mas[][][]) {
    try{
        scanner = new Scanner(new FileReader("matrix.txt"));
    }
}

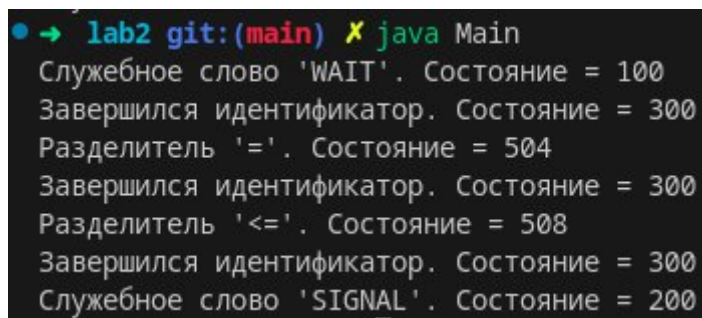
```



```

        int n = scanner.nextInt();
        int m = scanner.nextInt();
        if (scanner.hasNextInt()){
            mas = new int [n][m];
            for (int i=0; i<n; i++) {
                for (int j = 0; j < m; j++ ) {
                    mas[i][j] = scanner.nextInt();
                }
            }
        } catch (Exception e){
            e.printStackTrace();
        }
        return mas;
    }
    public static void Print(int[][] matrix) {
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[0].length; j++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
        }
        System.out.println();
    }
}

```



```

lab2 git:(main) X java Main
Службное слово 'WAIT'. Состояние = 100
Завершился идентификатор. Состояние = 300
Разделитель '='. Состояние = 504
Завершился идентификатор. Состояние = 300
Разделитель '<='. Состояние = 508
Завершился идентификатор. Состояние = 300
Службное слово 'SIGNAL'. Состояние = 200

```

Рисунок 4 – Результат выполнения программы

ВЫВОДЫ

В ходе выполнения лабораторной работы были изучены принципы построения и программирования лексического анализатора на языке С (С++) для простых языковых конструкций. Получены навыки практического построения лексического анализатора (сканера) на основе теории конечных автоматов. Освоены приёмы составления регулярных выражений для описания лексем. Закреплены навыки построения минимального КА, осуществляющего сканирование текста программ.