|  |  |
| --- | --- |
| Министерство образования Республики Беларусь  Учреждение образования «Полоцкий государственный университет имени Евфросинии Полоцкой» | |
|  | Кафедра технологий программирования |
| Компиляторные технологии  Отчет по лабораторной работе №1  Вариант 11 | |
| Выполнил | Купаленко А.Д., студент гр. 21-ИТ-1, ФИТ |
| Проверил | Сыцевич Д.Н., Преподаватель-стажер кафедры ТП. |
| Полоцк  2022г. | |

**Лабораторная работа № 1**

**“Лексическии анализатор”**

**Цель работы:** Ознакомится с лексическими анализаторами, принципами их работы и использованием на практике.

**ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ** (ответы на контрольные вопросы):

1. Какую роль выполняет лексический анализ в процессе компиляции?

Лексический анализатор (сканер) читает поток символов, составляющих исходную программу, и группирует эти символы в значащие последовательности, называющиеся лексемами.

1. Как связаны лексический и синтаксический анализ?

На вход лексического анализатора поступает текст исходной программы, а выходная информация передается для дальнейшей обработки синтаксическому анализатору. Для каждой лексемы сканер строит выходной токен вида.

1. Какие проблемы необходимо решить при построении лексического анализатора на основе конечного автомата?

* КА для каждой входной цепочки входного языка дает ответ на вопрос о том, принадлежит или нет цепочка языку, заданному этим автоматом. Однако в общем случае задача лексического анализатора несколько шире, чем просто проверка цепочки символов лексемы на соответствие входному языку. Кроме этого, он должен выполнить следующие действия:

1. определить границы лексем, которые в тексте исходной программы явно не указаны;
2. выполнить действия для сохранения информации об обнаруженной лексеме (или выдать сообщение об ошибке, если лексема неверна).

* Проблема определения границ лексем;

1. Чем отличаются таблица лексем и таблица идентификаторов? В какую из этих таблиц лексический анализатор не должен помещать ключевые слова разделители и знаки операций?

Таблица лексем фактически содержит весь текст исходной программы, обработанный лексическим анализатором. В нее входят все возможные типы лексем, кроме того, любая лексема может встречаться в ней любое количество раз. Таблица идентификаторов содержит только определенные типы лексем — идентификаторы и константы. В нее не попадают такие лексемы, как ключевые (служебные) слова входного языка, знаки операций и разделители.

**Практическая часть**

**Задание**

Для выполнения лабораторной работы необходимо:

1) написать программу, которая выполняет лексический анализ входного текста в соответствии с заданием и порождает таблицу лексем с указанием их типов. Программа должна выдавать сообщения о наличии во входном тексте ошибок, которые могут быть обнаружены на этапе лексического анализа;

2) в качестве вспомогательного средства для генерации кода лексического анализатора использовать Flex.

**Вариант 11**

Входной язык содержит операторы цикла while (...) ... done, разделённые символом ; (точка с запятой). Операторы условия содержат идентификаторы, знаки сравнения <, >, =, строковые константы (последовательность символов в двойных кавычках), знак присваивания (:=).

**Выполнение задания**

%option noyywrap yylineno

%{

    #include <stdio.h>

    int ch;

%}

digit[0-9]

letter[a-zA-Z]

string[\s\S]

delim[();]

oper[<>=]

ws[ \t\n]

%%

while { printf("KEYWORD (%d, %d): %s\n", yylineno, ch, yytext);

    ch += yyleng;

}

done { printf("KEYWORD (%d, %d): %s\n", yylineno, ch, yytext);

    ch += yyleng;

}

"\"Hello, World!\"" { printf("STRING (%d, %d): %s\n", yylineno, ch, yytext);

    ch += yyleng;

}

("\_"|{letter})("\_"|{letter}|{digit})\* {

    printf("IDENTIFIER (%d, %d): %s\n", yylineno, ch, yytext);

    ch += yyleng;

}

[-+]?({digit}\*\.{digit}+|{digit}+\.|{digit}+)([eE][-+]?{digit}+)?[flFL]? {

    printf("NUMBER (%d, %d): %s\n", yylineno, ch, yytext);

    ch += yyleng;

}

{oper} { printf("OPERATION (%d, %d): %s\n", yylineno, ch, yytext);

    ch += yyleng;

}

":=" { printf("OPERATION (%d, %d): %s\n", yylineno, ch, yytext);

    ch += yyleng;

}

{delim} { printf("DELIMITER (%d, %d): %s\n", yylineno, ch, yytext);

    ch += yyleng;

}

{ws}+ { ch += yyleng; }

. { printf("Unknown character (%d, %d): %s\n", yylineno, ch, yytext);

    ch += yyleng;

}

%%

int main(int argc, char \*\*argv)

{

    if(argc < 2)

    {

        printf("\nNot enough arguments. Please specify filename.\n");

        return -1;

    }

    if((yyin = fopen(argv[1], "r")) == NULL)

    {

        printf("\nCannot open file %s.\n", argv[1]);

        return -1;

    }

    ch = 1;

    yylineno = 1;

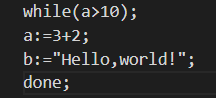
    yylex();

    fclose(yyin);

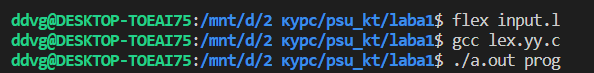
    return 0;

}

*Листинг программы*

**

*Входной файл для анализа*

**

KEYWORD (1, 1): while

DELIMITER (1, 6): (

IDENTIFIER (1, 7): a

OPERATION (1, 8): >

NUMBER (1, 9): 10

DELIMITER (1, 11): )

DELIMITER (1, 12): ;

Unknown character (1, 13):

IDENTIFIER (2, 15): a

OPERATION (2, 16): :=

NUMBER (2, 18): 3

NUMBER (2, 19): +2

DELIMITER (2, 21): ;

Unknown character (2, 22):

IDENTIFIER (3, 24): b

OPERATION (3, 25): :=

STRING (3, 27): "Hello, World!"

DELIMITER (3, 41): ;

Unknown character (3, 42):

KEYWORD (4, 44): done

DELIMITER (4, 48): ;

*Рисунок 1 - Результат работы программы*

Вывод: Ознакомились лексическими анализаторами, принципами их работы и использованием на практике.