Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа компьютерных технологий и информационных систем

**Отчёт по индивидуальной работе №2**

Дисциплина: Транслирующие системы

Тема: Программирование синтаксического анализатора на языке yacc

Выполнил студент гр. 5130901/20103 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.А. Вагнер

(подпись)

Принял старший преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.Н. Цыган

(подпись)

“ ” 2024 г.

Санкт-Петербург

2024

Оглавление

[Текст задания 3](#_Toc180752836)

[Ход работы 3](#_Toc180752837)

[Вывод 7](#_Toc180752838)

Список иллюстраций

[Рис. 1 – текст исходного файла c.l 4](#_Toc180752828)

[Рис. 2 – текст исходного файла c1.l 4](#_Toc180752829)

[Рис. 3 – текст программы c.l 5](#_Toc180752830)

[Рис. 4 – секция определения c.y 5](#_Toc180752831)

[Рис. 5 - секция правил c.y 6](#_Toc180752832)

[Рис. 6 – Тест программы c.y 6](#_Toc180752833)

[Листинг 1 – Содержание c.l 8](#_Toc180752834)

[Листинг 2 – Содержание c.y 9](#_Toc180752835)

Текст задания

Модифицировать базовую программу из раздела "Рекурсивные правила следующим образом:

1) Программа должна стать способной принимать на вход произвольное количество строк (а не единственную строку) с выполнением заданных семантических функций для каждой отдельной строки (подсчет количества чисел).

2) Изменить допустимую структуру строк. (В добавление к предыдущей модификации.) Пусть теперь в списке (в каждой строке) могут встречаться не только целые числа со знаком и без знака, но и идентификаторы. Однако на каждой третьей позиции списка может находиться только идентификатор, а на остальных позициях – любой элемент (целое или идентификатор). Заканчиваться строка может на любой позиции.

3) Заданные семантические функции:

Подсчитывать для каждой строки:

а) суммарное количество целых чисел в строке;

б) суммарное количество идентификаторов в строке.

Дополнительно подсчитывать количество обработанных строк.

Ход работы

В качестве основы для выполнения задания была взята программа из раздела “Рекурсивные правила”. Тексты этих lex и yacc файлов приведены ниже.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рис. 1 – текст исходного файла c.l

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 2 – текст исходного файла c1.l

Для начала модифицируем lex файл, нам требуется, чтобы он определял идентификаторы и возвращал соответствующее название лексемы.

Также для удобства было введено название лексемы конца строки – EOL. Текст полученной программы lex приведён ниже

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 3 – текст программы c.l

Теперь обратим внимание на yacc программу. Для начала, в секции определения создадим переменные для хранения количества чисел, идентификаторов в строке, количество обработанных строк и текущей позиции в строке. Далее следует объявить все упомянутые раннее лексемы.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рис. 4 – секция определения c.y

Чтобы программа могла обрабатывать несколько строк добавим рекурсию в правило лексемы \_\_list. Также для каждого из видов элементов списка добавим связанный скрипт, увеличивающий соответствующий счётчик или выполняющий проверку на присутствие идентификатора на третьей позиции. При обнаружении конца строки следующим за списком элементов, выводим результаты. Этот вывод происходит даже в случае обнаружения ошибки, так как в тексте задания было сказано производить счёт элементов для каждой строки.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рис. 5 - секция правил c.y

Проверим работу программы.

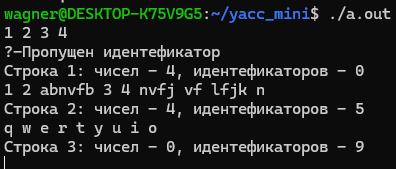


Рис. 6 – Тест программы c.y

Вывод

В ходе данной работы была составлена и протестирована программа синтаксического анализа на языке yacc. Были использованы знания о структуре и синтаксисе программ yacc. Результат соответствует спецификации

*Листинг 1 – Содержание c.l*

%{

#include <stdlib.h>

#include "y.tab.h"

#define YYSTYPE int

**extern** YYSTYPE yylval;

%}

%%

[**0**-**9**]+ { **return** NUM; }

[a-zA-Z\_][a-zA-Z0-**9**\_]\* { **return** ID; }

\n { **return** EOL; }

[ \t] {}

. **return** yytext[**0**];

%%

#ifndef yywrap

**int** yywrap () { **return** **1**; }

#endif

Листинг 2 – Содержание c.y

%{

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

**int** num\_count = **0**;

**int** id\_count = **0**;

**int** line\_count = **0**;

**int** current\_position = **1**;

%}

%token NUM

%token ID

%token EOL

%start \_\_list

%%

**\_\_list:**

|\_\_list \_list

**\_list:** elements EOL

{

printf("Строка %d: чисел - %d, идентефикаторов - %d**\n**", ++line\_count, num\_count, id\_count);

num\_count = **0**;

id\_count = **0**;

current\_position = **1**;

}

;

**elements:**

|element elements

;

**element:**

NUM

{

**if** (current\_position % **3** == **0**) {

yyerror("Пропущен идентефикатор", current\_position);

}

num\_count++;

current\_position++;

}

| ID

{

id\_count++;

current\_position++;

}

;

%%