|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт информационных технологий |
| Кафедра инструментального и прикладного программного обеспечения (ИиППО) |
|  |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2** | |
| **по дисциплине** | |
| **«**Теория автоматов и формальных языков**»**  **по теме** «Синтаксический анализатор» | |
|  | |
| Выполнил студент группы ИКБО-16-17 | Акжигитов Р. Р. |
| Принял старший преподаватель | Зорина Н. В. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «25» ноября 2019 г. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_2019 г. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Москва 2019

**Оглавление**

[Цель лабораторной работы 3](#_Toc25593031)

[Задание 3](#_Toc25593032)

[Выполнение лабораторной работы 3](#_Toc25593033)

[Результат работы программы 5](#_Toc25593034)

[Заключение 6](#_Toc25593035)

[Список литературных источников 7](#_Toc25593036)

# Цель лабораторной работы

Целью данной лабораторной работы является изучение и разработка синтаксического анализатора с использования инструмента Bison.

# Задание

Доработать программу лексического анализатора из лабораторной работы № 1 так, чтобы генерируемый ею поток токенов поступал на вход синтаксического анализатора. Выполнить программную реализацию синтаксического анализатора, используя генератор синтаксических анализаторов Bison. Результаты работы программы представить в виде дерева синтаксического разбора.

# Выполнение лабораторной работы

Для того, чтобы выполнить задание, необходимо создать файл my.y. В нем будет описан синтаксический анализатор. Также необходим файл y.tab.h, где будут описаны токены и файл simple.txt, где будет описан пример входных данных для анализатора. Все эти файлы должны быть взяты с прошлой лабораторной работы. Изменим файл simple.txt, чтобы входные данные были выражением.

// simple.txt

printf("Hello world!");

abc = 0x15ac;

printf(abc);

printf(0x23bc+abc);

Затем нужно описать, собственно, сам синтаксический анализатор. Для начала опишем все токены, которые включены в анализатор. Далее нужно описать правила разбора синтаксиса, которые позволят программе проанализировать синтаксис. Полный листинг файла my.y представлен ниже:

// my.y

%{

#include <stdio.h>

extern FILE \*yyin;

extern int yylineno;

extern char \*yytext;

void yyerror(char \*);

extern int yylex(void);

%}

%token SEMICOLON BECOMES PLUS MINUS TIMES SLASH LPAR RPAR HEXNUM IDENT UNKNOWN LITERAL COMMA EXCLAM QUEST PRINTF

%%

program: statement | program statement

statement: expr SEMICOLON { printf("statement\n"); }

| print\_statement SEMICOLON

expr: IDENT BECOMES expr\_cmp { printf("assign\n"); }

expr\_cmp:

prim\_expr PLUS prim\_expr

| prim\_expr MINUS prim\_expr

| prim\_expr TIMES prim\_expr

| prim\_expr SLASH prim\_expr

| prim\_expr { printf("expression\n"); }

prim\_expr: IDENT { printf("ident"); }

| HEXNUM { printf("hexnum\n"); }

| LPAR expr\_cmp RPAR { printf("primary expression\n"); }

print\_statement: PRINTF LPAR expr\_cmp RPAR { printf("print expr\n"); }

| PRINTF LPAR LITERAL RPAR { printf("print literal\n"); }

%%

void yyerror(char \*errmsg) {

fprintf(stderr, "%s (%d): %s\n", errmsg, yylineno, yytext);

}

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;

}

yylineno = 1;

yyparse();

return 0;

}

Листинг файла input.l:

// input.l

%option noyywrap yylineno

%{

#include "y.tab.h"

%}

digit [0-9]

hexdigit [a-f]

letter [a-zA-Z]

marks [\?\!\,]

%%

"+" { return PLUS; }

"-" { return MINUS; }

"/" { return SLASH; }

"\*" { return TIMES; }

"=" { return BECOMES; }

"(" { return LPAR; }

")" { return RPAR; }

"'"[^']\*"'" { return LITERAL; }

"\""[^\"]\*"\"" { return LITERAL; }

"," { return COMMA; }

";" { return SEMICOLON; }

"!" { return EXCLAM; }

"?" { return QUEST; }

"printf" { return PRINTF; }

"0x"({hexdigit}|{digit})\* { return HEXNUM; }

{letter}({letter}|{digit})\* { return IDENT; }

%%

# Результат работы программы

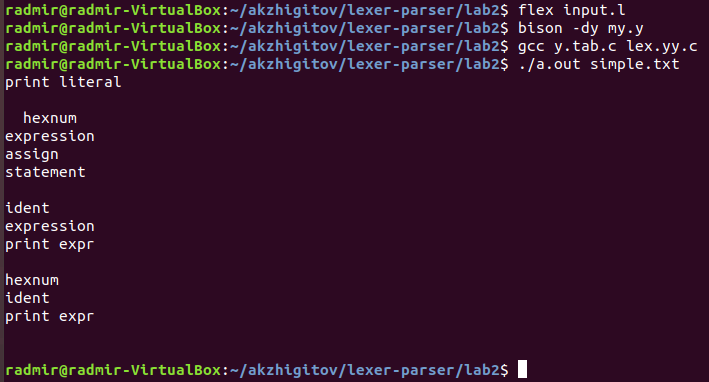


Рисунок – результат работы синтаксического анализатора

# Заключение

В данной лабораторной работе были изучены принципы построения синтаксического анализатора (состав, принцип работы, способы описания, инструментальные средства для генерации анализаторов, например, bison). Выполнено задание, в котором описана необходимая для варианта грамматика, также описана главная функция для считывания символов из файла и передачи их в лексер и парсер, для последующей обработки грамматики.

# Список литературных источников

1. Альфред В. Ахо, Моника С. Лам, Рави Сети, Джеффри Д. Ульман. Компиляторы: принципы, технологии и инструментарий = Compilers: Principles, Techniques, and Tools. — 2-е изд. — М.: Вильямс, 2008. — ISBN 978-5-8459-1349-4.
2. Робин Хантер. Основные концепции компиляторов = The Essence of Compilers. — М.: «Вильямс», 2002. — С. 256. — ISBN 5-8459-0360-2.
3. Молчанов А.Ю. Системное программное обеспечение: Учебник для вузов. 3-е изд. – СПб.: Питер, 2010 – 400 с.
4. Генератор синтаксических анализаторов, совместимый с YACC для Bison версии 1.35, 25 февраля 2002 [URL: https://www.opennet.ru/docs/RUS/bison\_yacc/]