Министерство науки и высшего образования РФ

ФГАОУ ВО Пермский национальный исследовательский

политехнический университет

Кафедра «Вычислительная математика, механика и биомеханика»

Отчет по лабораторной работе № 3

тема «Разработка синтаксически управляемого транслятора с использованием flex и bison»

по дисциплине «Языки программирования и методы трансляции»

Выполнил: студент группы ИСТ-22-1б Братчиков З.С.

Задание 1

Постановка задачи:

Написать программу для генератора лексических анализаторов, выводящую в консоль переданный токен.

Решение:

%{

#include <stdio.h>

%}

%%

[0-9]+ { printf("Целое число\n"); }

[+-]?[0-9]+\.[0-9]+ {printf("Число с плавающей точкой\n"); }

[a-zA-Z]+ { printf("Строка\n"); }

"+" { printf("Плюс\n"); }

"-" { printf("Минус\n"); }

"\*" { printf("Умножение\n"); }

"/" { printf("Деление\n"); }

"(" { printf("Левая скобка\n"); }

")" { printf("Правая скобка\n"); }

. { printf("Неизвестный символ\n"); }

%%

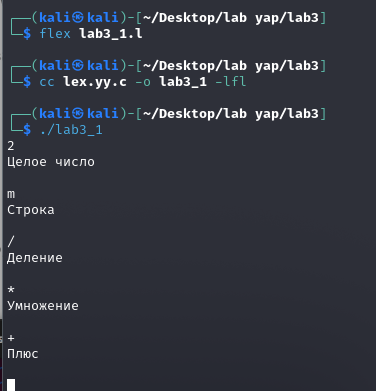
*int* main() {

yylex();

return 0;

}

Вывод программы в терминале:



Задание 2

Постановка задачи:

Описать грамматику, позволяющую вычислять значения арифметических выражений, состоящих из произвольного конечного количества целых чисел и знаков «+» и «-»

Решение:

Flex-файл

%{

#include <stdio.h>

#include <string.h>

#include "lab3\_2.tab.h"

*void* exit\_ctrlq();

*void* Errors();

%}

%%

[0-9]+ { yylval=atoi(yytext); return INTEGER; }

"+" { return SUM; }

"-" { return SUB; }

"\*" { return MUL; }

"/" { return DIV; }

"(" { return OBRACE; }

")" { return EBRACE; }

"\n" { return EOL; }

[ \t]+ ; /\* skip whitespaces \*/

. { Errors();}

"exit" {exit\_ctrlq();}

%%

*int* yywrap(){

return 1;

}

*void* Errors(){

printf("Something went wrong");

}

*void* exit\_ctrlq(){

exit(0);

}

Bison-файл

%{

#include <stdio.h>

#include <string.h>

#include "lab3\_2.tab.h"

*int* yylex(*void*);

*void* yyerror(const *char* \**str*);

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

%}

%token *INTEGER*

%token *SUM* *SUB* *MUL* *DIV*

%token *OBRACE* *EBRACE*

%token *EOL*

%token *NUMBER*

%%

*calclist*: /\* nothing \*/

| calclist exp *EOL* { printf("= %d\n", $2); }

;

*exp*: factor

| exp *SUM* factor { $$ = $1 + $3; }

| exp *SUB* factor { $$ = $1 - $3; }

;

*factor*: term

| factor *MUL* term { $$ = $1 \* $3; }

| factor *DIV* term { $$ = $1 / $3; }

;

*term*: *INTEGER*

| *OBRACE* exp *EBRACE* { $$ = $2; }

;

%%

*void* yyerror(const *char* \**str*){

fprintf(stderr,"Error: %s\n",str);

}

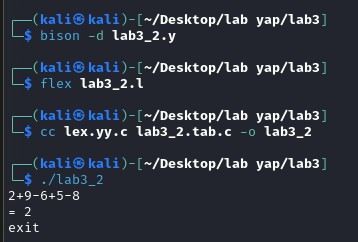
*int* main(*int* *argc*, *char* \*\**argv*){

yyparse();

return 0;

}

Вывод программы в терминале:



Задание 3

Постановка задачи:

Описать грамматику, позволяющую проводить трансляцию арифметических выражений, состоящих из вещественных чисел, знаков «+», «-», «\*», «/» и скобок из инфиксной записи в постфиксную.

Решение:

Flex-файл

%{

#include <stdio.h>

#include "lab3\_3.tab.h"

*void* Errors();

%}

%option *noyywrap*

%%

[0-9]+(\.[0-9]+)? { yylval.number = atof(yytext); return NUMBER; }

[-+\*/(){}] { return yytext[0]; }

\n { return EOL; }

[ \t] { /\* ignore whitespace \*/ }

. { Errors(); }

%%

*void* Errors(){

printf("Something went wrong");

}

Bison-файл

%{

#include <stdio.h>

#include <stdlib.h>

*int* yylex(*void*);

*void* yyerror(const *char* \**s*);

%}

%union {

*double* number;

}

%token <number> *NUMBER*

%token *EOL*

%left '+' '-'

%left '\*' '/'

%%

*comman*: /\* nothing \*/

| comman exp *EOL* { printf("\n"); }

*exp*: term

| exp '+' term { printf("+ "); }

| exp '-' term { printf("- "); }

*term*: factor

| term '\*' factor { printf("\* "); }

| term '/' factor { printf("/ "); }

*factor*: *NUMBER* { printf("%.2f ", $1); }

| '(' exp ')' {}

%%

*int* main(*void*) {

yyparse();

return 0;

}

*void* yyerror(const *char* \**s*) {

fprintf(stderr, "Error: %s\n", s);

}

Вывод программы в терминале:

