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 Васин М.А.

Проверил: доц. Батин С.Е.

Пермь, 2023

**Содержание**

[Задание 1 3](#_Toc154021531)

[1.1. Постановка задачи 3](#_Toc154021532)

[1.2. Решение задачи, код программы 3](#_Toc154021533)

[1.3. Тестирование работы программы 3](#_Toc154021534)

[Задание 2 3](#_Toc154021535)

[2.1. Постановка задачи 3](#_Toc154021536)

[2.2. Решение задачи, код программы 4](#_Toc154021537)

[2.3. Тестирование работы программы 5](#_Toc154021538)

[Задание 3 5](#_Toc154021539)

[3.1. Постановка задачи 5](#_Toc154021540)

[3.2 Решение задачи, код программы 5](#_Toc154021541)

[3.3. Тестирование работы программы 5](#_Toc154021542)

# Задание 1

## 1.1. Постановка задачи

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

## 1.2. Решение задачи, код программы

%{

#include <stdio.h>

%}

**%%**

[0-9]+ { printf("integer number\n"); }

[+-]?[0-9]+\.[0-9]+ {printf("float number\n"); }

[a-zA-Z]+ { printf("string\n"); }

"+" { printf("plus\n"); }

"-" { printf("minus\n"); }

"\*" { printf("multiplication\n"); }

"/" { printf("division\n"); }

"(" { printf("left bracket\n"); }

")" { printf("right bracket\n"); }

"." { printf("point\n"); }

"," { printf("comma\n"); }

"%" { printf("percent\n"); }

"&" { printf("ampersand\n"); }

"!" { printf("exclamation mark\n"); }

"%" { printf("question mark\n"); }

. { printf("unknown\n"); }

**%%**

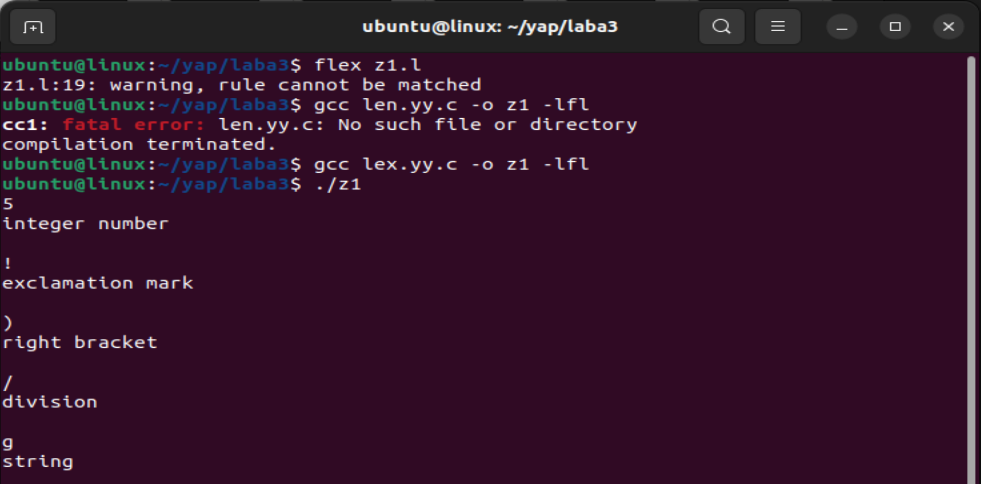
int main() {

yylex();

return 0;

}

## 1.3. Тестирование работы программы



# Задание 2

## 2.1. Постановка задачи

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

## 2.2. Решение задачи, код программы

Flex файл z2.l

%{

#include <stdio.h>

#include <string.h>

#include "z2.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]+ ;

. { Errors();}

"exit" {exit\_ctrlq();}

**%%**

int yywrap(){

return 1;

}

void Errors(){

printf("Something went wrong");

}

void exit\_ctrlq(){

exit(0);

}

Bison файл z2.y

%{

#include <stdio.h>

#include <string.h>

#include "z2.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

**%%**

calclist:

| 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);

}

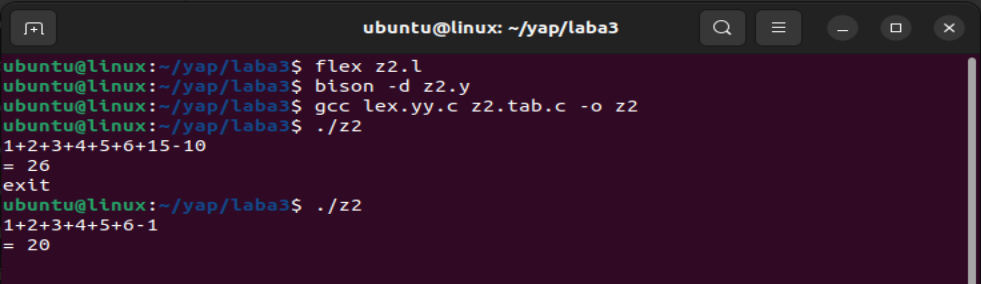
int main(int argc, char \*\*argv){

yyparse();

return 0;

}

## 2.3. Тестирование работы программы



# Задание 3

## 3.1. Постановка задачи

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

## 3.2 Решение задачи, код программы

Flex файл z3.l

%{

#include <stdio.h>

#include "z3.tab.h"

void Errors();

%}

%option noyywrap

**%%**

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

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

\n { return EOL; }

[ \t] { }

. { Errors(); }

**%%**

void Errors(){

printf("Something went wrong");

}

Bison файл z3.y

%{

#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:

| 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);

}

## 3.3. Тестирование работы программы

