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 lex и yacc»

по дисциплине «Языки программирования и методы трансляции»

Выполнил: студент группы ИСТ-21-1б Егоров В.С.

Проверил: Батин С.Е.

Пермь, 2022

Содержание

[Задание 1 3](#_Toc119767790)

[1.1. Постановка задачи 3](#_Toc119767791)

[1.2. Код программы 3](#_Toc119767792)

[1.3. Примеры работы программы 3](#_Toc119767793)

[Задание 2 4](#_Toc119767794)

[2.1. Постановка задачи 4](#_Toc119767795)

[2.2. Код программы 4](#_Toc119767796)

[2.3. Пример работы программы 5](#_Toc119767797)

[Задание 3 5](#_Toc119767798)

[3.1. Постановка задачи 5](#_Toc119767799)

[3.2. Код программы 5](#_Toc119767800)

[3.3. Пример работ программы 7](#_Toc119767801)

[Задание 4 7](#_Toc119767802)

[4.1. Постановка задачи 7](#_Toc119767803)

[4.2. Код программы 7](#_Toc119767804)

[4.3. Пример работы программы 9](#_Toc119767805)

# Задание 1

## 1.1. Постановка задачи

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

## 1.2. Код программы

**%{**

#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"**);**

**}**

**[**A**-**Za**-**z0**-**9**]+** **{**

printf**(**"ID\n"**);**

**}**

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

**%%**

## 1.3. Примеры работы программы

На рис.1 приведен пример выполнения кода.

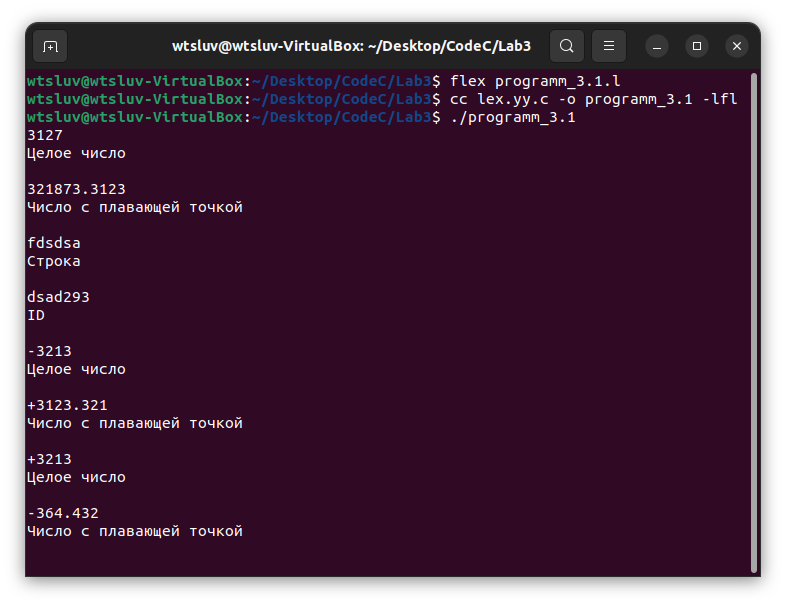


Рисунок 1. Программа 1

# Задание 2

## 2.1. Постановка задачи

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

## 2.2. Код программы

**%{**

#include <stdio.h>

#include <string.h>

#include "programm\_3.2.tab.h"

**%}**

**%%**

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

"+" **{** **return** ADD**;** **}**

"-" **{** **return** SUB**;** **}**

"\*" **{** **return** MUL**;** **}**

"/" **{** **return** DIV**;** **}**

"(" **{** **return** OBRACE**;** **}**

")" **{** **return** EBRACE**;** **}**

\n **{return** EOL**;}**

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

**%%**

**%{**

#include <stdio.h>

#include <string.h>

#include "programm\_3.2.tab.h"

int yylex**(**void**);**

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

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

**}**

int yywrap**(){**

**return** 1**;**

**}**

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

yyparse**();**

**}**

**%}**

**%**token INTEGER

**%**token ADD SUB MUL DIV

**%**token OBRACE EBRACE

**%**token EOL

**%%**

calclist**:** /\* nothing \*/

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

**;**

exp**:** factor

**|** exp ADD 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**;** **}**

**;**

**%%**

## 2.3. Пример работы программы

На рис.2 приведен пример выполнения кода.

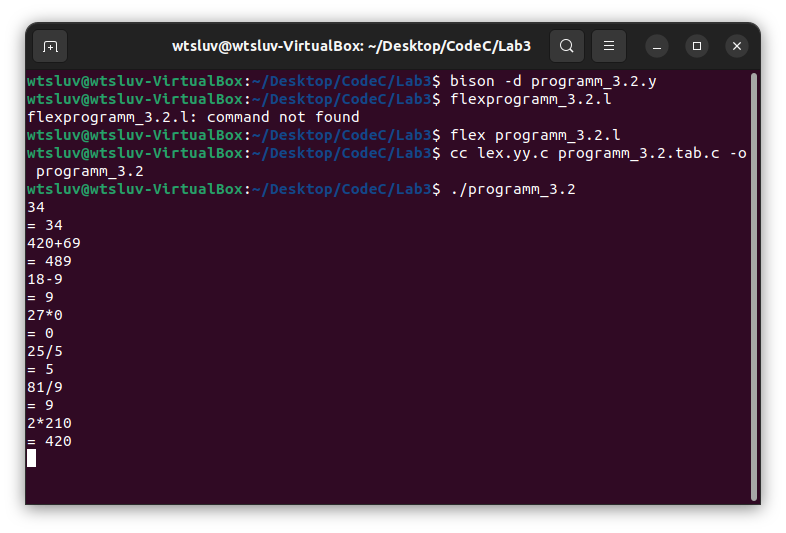


Рисунок 2. Программа 2

# Задание 3

## 3.1. Постановка задачи

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

## 3.2. Код программы

**%{**

#include <stdio.h>

#include <string.h>

#include "programm\_3.3.tab.h"

**%}**

**%%**

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

"+" **{** **return** ADD**;** **}**

"-" **{** **return** SUB**;** **}**

"\*" **{** **return** MUL**;** **}**

"/" **{** **return** DIV**;** **}**

"(" **{** **return** OBRACE**;** **}**

")" **{** **return** EBRACE**;** **}**

"exit" **{** **return** EXIT**;** **}**

\n **{return** EOL**;}**

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

**%%**

**%{**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include "programm\_3.3.tab.h"

int yylex**(**void**);**

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

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

**}**

int yywrap**(){**

**return** 1**;**

**}**

int main**(){**

yyparse**();**

**}**

**%}**

**%**token INTEGER

**%**token ADD SUB MUL DIV

**%**token OBRACE EBRACE

**%**token EXIT

**%**token EOL

**%%**

commands**:** /\* nothing \*/

**|** commands command

**;**

command**:** exp

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

**|** EXIT **{** printf**(**"bye!\n"**);** exit**(**0**);** **}**

**;**

exp**:** factor

**|** exp ADD factor **{** printf**(**"+"**);** **}**

**|** exp SUB factor **{** printf**(**"-"**);** **}**

**;**

factor**:** term

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

**|** factor DIV term **{** printf**(**"/"**);** **}**

**;**

term**:** INTEGER **{** printf**(**"%d"**,** $1**);** **}**

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

**;**

**%%**

## 3.3. Пример работ программы

На рис.3 приведен пример выполнения кода.

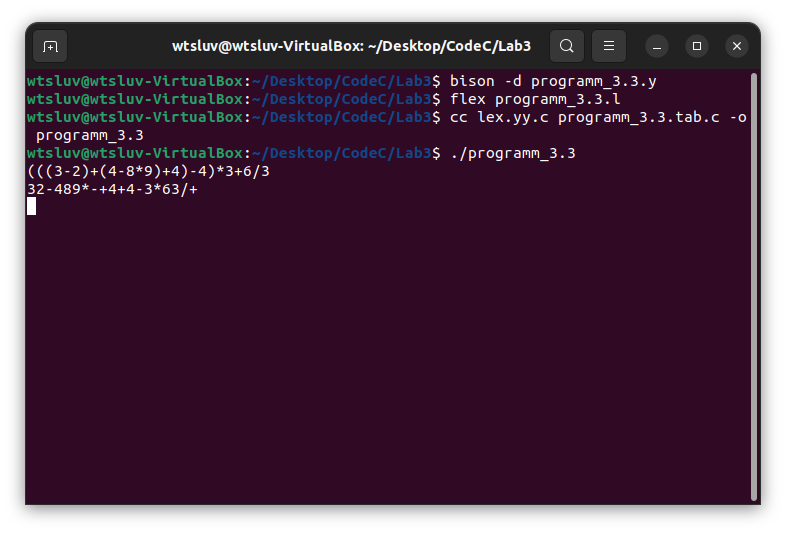


Рисунок 3. Программа 3

# Задание 4

## 4.1. Постановка задачи

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

## 4.2. Код программы

**%{**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include "programm\_3.4.tab.h"

**%}**

**%%**

**[**0**-**9**]+[**\**.]?[**0**-**9**]\*** **{** yylval**.**dval **=** atof**(**yytext**);** **return** FNUMB**;** **}**

"+" **{** **return** ADD**;** **}**

"-" **{** **return** SUB**;** **}**

"\*" **{** **return** MUL**;** **}**

"/" **{** **return** DIV**;** **}**

"(" **{** **return** OBRACE**;** **}**

")" **{** **return** EBRACE**;** **}**

"exit" **{** **return** EXIT**;** **}**

\n **{return** EOL**;}**

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

**%%**

**%{**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include "programm\_3.4.tab.h"

int yylex**(**void**);**

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

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

**}**

int yywrap**(){**

**return** 1**;**

**}**

int main**(){**

yyparse**();**

**}**

**%}**

**%**union **{**

double dval**;**

**}**

**%**token FNUMB

**%**token ADD SUB MUL DIV

**%**token OBRACE EBRACE

**%**token EXIT

**%**token EOL

**%**type **<**dval**>** FNUMB

**%**type **<**dval**>** exp

**%**type **<**dval**>** term

**%**type **<**dval**>** factor

**%%**

commands**:** /\* nothing \*/

**|** commands command

**;**

command**:** exp

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

**|** EXIT **{** printf**(**"bye!\n"**);** exit**(**0**);** **}**

**;**

exp**:** factor

**|** exp ADD factor **{** printf**(**"+"**);** **}**

**|** exp SUB factor **{** printf**(**"-"**);** **}**

**;**

factor**:** term

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

**|** factor DIV term **{** printf**(**"/"**);** **}**

**;**

term**:** FNUMB **{** printf**(**"%f"**,** $1**);** **}**

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

**;**

**%%**

## 4.3. Пример работы программы

На рис.4 приведен пример выполнения кода.

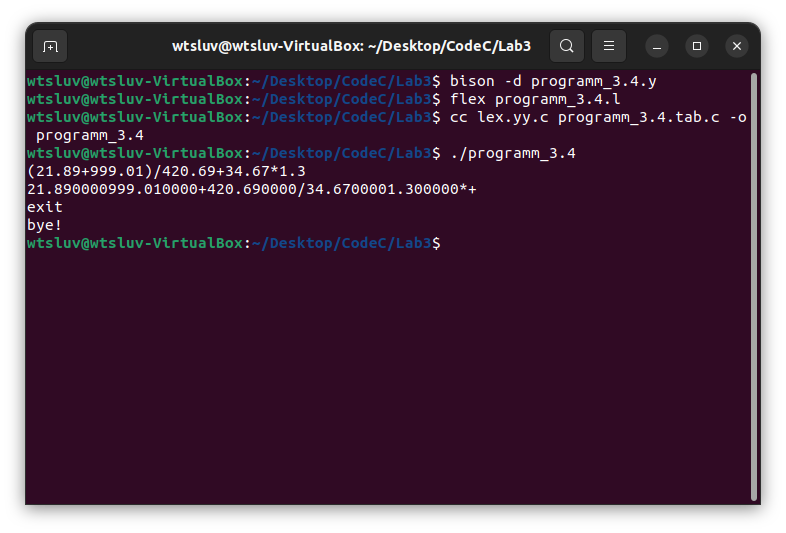


Рисунок 4. Программа 4