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 Петраков М.В.

Проверил: доцент каф. ВММБ Батин С.Е.

Пермь, 2023

[Задание 1 3](#_Toc151294367)

[1.1. Постановка задания 3](#_Toc151294368)

[1.2. Решение задачи, код программы 3](#_Toc151294369)

[1.3. Тестирование работы программы 3](#_Toc151294370)

[Задание 2 5](#_Toc151294371)

[2.1. Постановка задания 5](#_Toc151294372)

[2.2. Решение задачи, код программы 5](#_Toc151294373)

[2.3. Тестирование работы программы 6](#_Toc151294374)

[Задание 3 8](#_Toc151294375)

[3.1. Постановка задания 8](#_Toc151294376)

[3.2. Решение задачи, код программы 8](#_Toc151294377)

[3.3. Тестирование работы программы 9](#_Toc151294378)

[Задание 4 11](#_Toc151294379)

[4.1. Постановка задания 11](#_Toc151294380)

[4.2. Решение задачи, код программы 11](#_Toc151294381)

[4.3. Тестирование работы программы 13](#_Toc151294382)

[Задание 5 14](#_Toc151294383)

[5.1. Постановка задания 14](#_Toc151294384)

[5.2. Решение задачи, код программы 14](#_Toc151294385)

[5.3. Тестирование работы программы 16](#_Toc151294386)

Задание 1

1.1. Постановка задания

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

1.2. Решение задачи, код программы

После вызова исполняемого файла открывается возможность ввода различных последовательностей символов. Существует возможность распознать целое число, вещественное число, некий оператор, а также строку, составленную из буквенных символов и цифр, причём начинающуюся с буквы.

%{

#include <stdio.h>

%}

**%%**

"-"[0-9]+ {printf("%s - INTEGER NUMBER LESS THAN 0\n",yytext);}

"0"+ {printf("%s - NUMBER EQUALS 0\n",yytext);}

[1-9][0-9]\* {printf("%s - INTEGER NUMBER MORE THAN 0\n",yytext);}

"-"?[0-9]+"."[0-9]+ {printf("%s - FLOAT NUMBER\n", yytext);}

[-+\*/%&] {printf("%s - OPERATOR\n",yytext);}

[a-zA-Z][a-zA-Z0-9]\* {printf("%s - STRING\n",yytext);}

[ \t]+ ;

. ;

**%%**

1.3. Тестирование работы программы

Далее представлена работа программы с четырьмя числами, одним оператором и одной строкой.

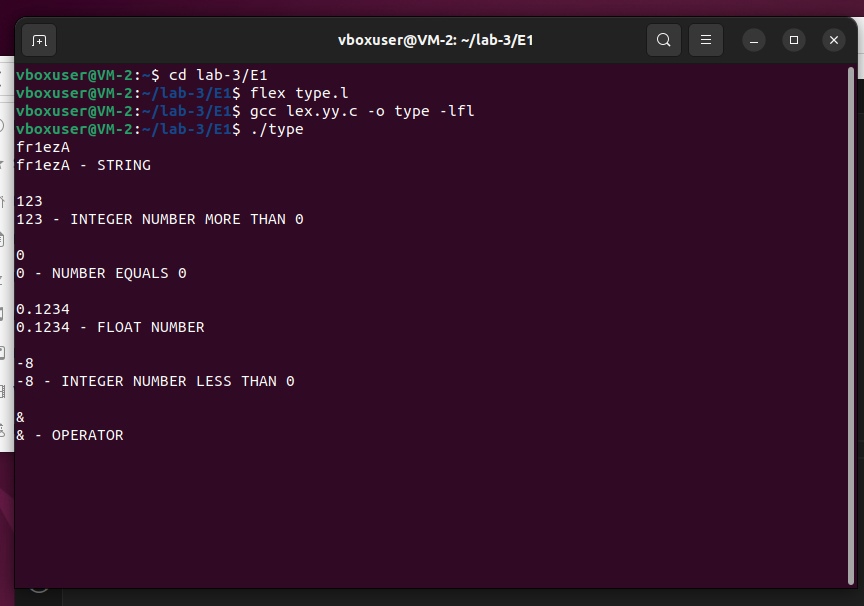


Рис.1 Успешная работа программы type.l

Задание 2

2.1. Постановка задания

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

2.2. Решение задачи, код программы

В ходе решения этой задачи было составлено 2 файла. В первом описаны необходимые токены – числа, знаки “плюс” и “минус”, а также символ перехода на новую строку.

%{

#include <stdio.h>

#include <string.h>

#include "intcalc.tab.h"

%}

**%%**

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

"+" {return ADD;}

"-" {return SUB;}

\n {return EOL;}

[ \t] ;

**%%**

Во втором файле введены необходимые функции, а также описана грамматика вычисления выражений сложения и вычитания целых чисел.

%{

#include <stdio.h>

#include <string.h>

#include "intcalc.tab.h"

int yylex();

void yyerror(const char \* s){

    fprintf(stderr,"ERROR: %s\n",s);

}

int yywrap(){

    return 1;

}

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

    yyparse();

}

%}

%token NUMBER ADD SUB EOL

**%%**

calclist:

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

;

exp: factor

|exp ADD factor{$$=$1+$3;}

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

;

factor: term

;

term: NUMBER

;

**%%**

2.3. Тестирование работы программы

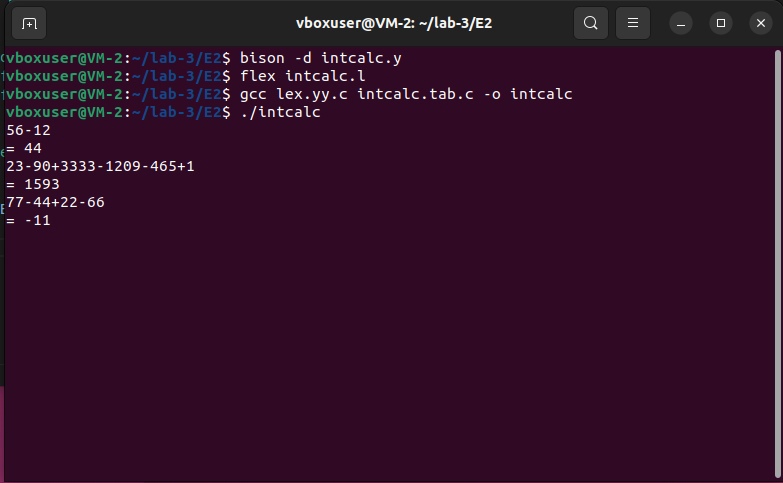


Рис.2 Успешная работа программы intcalc (консоль)

Задание 3

3.1. Постановка задания

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

Решение задачи привело к написанию двух файлов. В первом файле описаны регулярные выражения, определяющие токены – арифметические операции, скобки, переход на новую строку, а также стоп-слово.

%{

#include "translator.tab.h"

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

%}

**%%**

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

"+"     {return ADD;}

"-"     {return SUB;}

"\*"     {return MUL;}

"/"     {return DIV;}

"("     {return OBR;}

")"     {return EBR;}

\n      {return EOL;}

"stop"  {return END;}

[ \t]+ ;

**%%**

Второй файл содержит необходимые функции, а также описание грамматики перевода выражений из инфиксной записи в постфиксную.

%{

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include "translator.tab.h"

int yylex(void);

void yyerror(const char \* str){

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

}

int yywrap(){

    return 1;

}

int main(){

    yyparse();

}

%}

%union{

    double dval;

}

%token FNUM

%token ADD SUB MUL DIV

%token OBR EBR

%token EOL

%token END

%type <dval> FNUM

%type <dval> exp

%type <dval> term

%type <dval> factor

**%%**

commands: /\* void \*/

| commands command

;

command: exp

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

|END {printf("thanks for working\n");exit(0);}

;

exp: factor

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

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

;

factor: term

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

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

;

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

|OBR exp EBR {$$=$2;}

;

**%%**

3.3. Тестирование работы программы

Далее представлен пример работы программы translator.

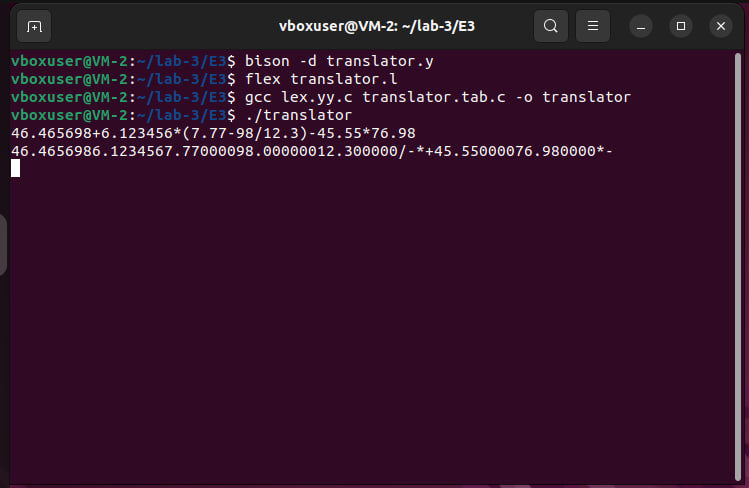


Рис.3 Работа программы translator