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 «Языки программирования и методы трансляции»

Выполнил: студент группы ИСТ-22-1Б Язов М.М.

Проверил: Батин С.Е.

Пермь, 2022

Содержимое

[Упражнение 1 3](#_Toc152798779)

[1.1. Постановка задачи 3](#_Toc152798780)

[1.2. Код программы 3](#_Toc152798781)

[1.3 Пример работы программы 4](#_Toc152798782)

[Упражнение 2 5](#_Toc152798783)

[2.1. Постановка задачи 5](#_Toc152798784)

[2.2. Код программы 5](#_Toc152798785)

[2.3. Примеры работы программы 6](#_Toc152798786)

[Упражнение 3 7](#_Toc152798787)

[3.1. Постановка задачи 7](#_Toc152798788)

[3.2 Код программы 7](#_Toc152798789)

[3.3. Пример работы программы 8](#_Toc152798790)

# Упражнение 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("INTEGER\n"); }

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

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

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

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

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

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

"(" { printf("LBRAKE\n"); }

")" { printf("RBRAKE\n"); }

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

**%%**

int main() {

yylex();

return 0;

}

## 1.3 Пример работы программы

На рис. 1 продемонстрирован результат работы программы 1 упражнения.

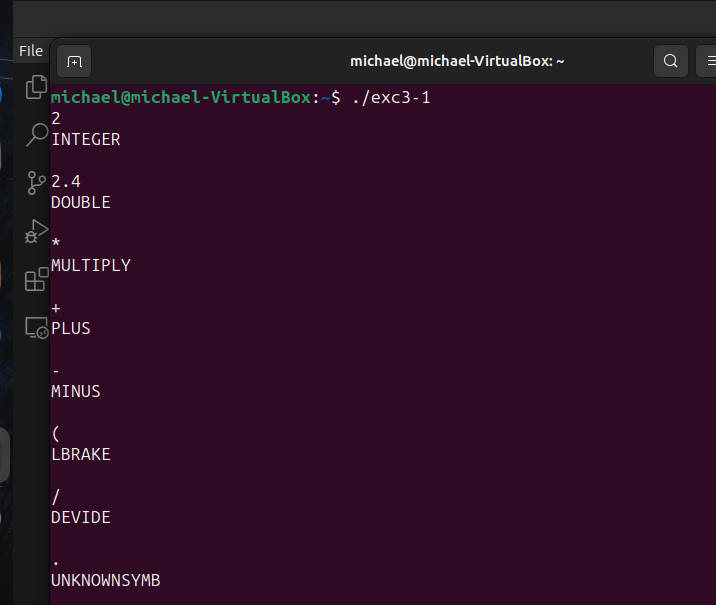


Рисунок 1. Результат программы 1 упражнения

# Упражнение 2

## 2.1. Постановка задачи

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

## 2.2. Код программы

%{

#include "exc3-2.tab.h"

%}

**%%**

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

[-+\n] { return yytext[0]; }

.

**%%**

%{

#include <stdio.h>

// Объявления функций для лексического и синтаксического анализа

int yylex();

int yyparse();

void yyerror(const char \*s);

// Переменная для хранения результата выражения

int result;

%}

%token NUMBER

%%

// Правила синтаксического анализа

expression: expression '\n' {

printf("Результат: %d\n", $1);

result = $1;

}

| expression '+' term { $$ = $1 + $3; }

| expression '-' term { $$ = $1 - $3; }

| term { $$ = $1; }

;

term: NUMBER { $$ = $1; };

%%

// Объявление функции лексического анализатора

int yylex();

// Функция main

int main() {

// Вызов синтаксического анализатора

yyparse();

return 0;

}

// Функция для обработки ошибок

void yyerror(const char \*s) {

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

}

## 2.3. Примеры работы программы

На рис.2 показаны результаты работы программы упражнения 2.

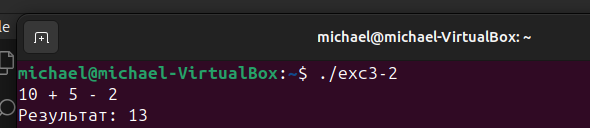


Рисунок 2. Результат программы 2 упражнения

# Упражнение 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 Код программы

%{

#define YYSTYPE double

#include "exc3-3.tab.h"

%}

**%%**

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

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

.

**%%**

%{

#include <stdio.h>

// Объявления функций для лексического и синтаксического анализа

int yylex();

int yyparse();

void yyerror(const char \*s);

%}

// Объединение для представления числа (тип double)

%union {

double number;

}

// Токен для представления числа

%token <number> NUMBER

// Определение типов для символов синтаксического анализа

%type <number> factor term expr

%%

// Правила синтаксического анализа

expr: expr '\n' {

// Вывод результата выражения при завершении строки

printf("\nResult: %.2f\n", $1);

return 0;

}

| term { $$ = $1; } // Основное выражение

| expr '+' term { printf("+ "); $$ = $1 + $3; } // Сложение

| expr '-' term { printf("- "); $$ = $1 - $3; } // Вычитание

;

term: factor { $$ = $1; } // Основной терм

| term '\*' factor { printf("\* "); $$ = $1 \* $3; } // Умножение

| term '/' factor { printf("/ "); $$ = $1 / $3; } // Деление

;

factor: NUMBER {

// Вывод числа и присвоение результата

printf("%.2f ", $1);

$$ = $1;

}

| '(' expr ')' { $$ = $2; } // Обработка скобок

;

%%

// Объявление функции лексического анализатора

int yylex();

// Функция main

int main() {

// Вызов синтаксического анализатора

yyparse();

return 0;

}

void yyerror(const char \*s) {

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

}

## 3.3. Пример работы программы

На рис.3 представлен результат работы программы упражнения 3.

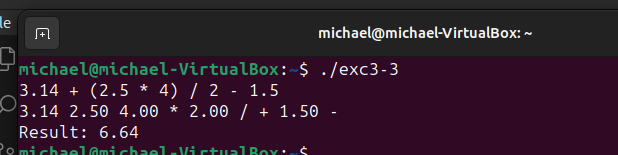


Рисунок 3. Пример работы программы упражнения 3