Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н. Э. Баумана

(национальный исследовательский университет)» (МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Космический факультет» КАФЕДРА «Прикладная математика, информатика и вычислительная техника»

**ОТЧЕТ**

по лабораторной работе № 6

по курсу «Системное программное обеспечение»

на тему: «Ознакомление с генератором программ синтаксического разбора YACC»

Вариант № 14

Студент \_\_К3-56Б\_\_  **\_\_\_\_\_\_\_\_\_\_** Несмеянов С. А.

(Группа) (Подпись, дата) (И.О.Фамилия)

#### Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_ Чернышов А. В.

(Подпись, дата) (И.О.Фамилия)

2024 г.

**Задание**



**Список используемых команд и правил**

**Директивы**:

* %{ ... %}: Используется для объявления библиотек и других включаемых файлов на C, которые будут доступны в вашем коде.
* %token DATA: Определяет токен DATA, который будет использоваться для представления чисел в выражениях.

**Правила**:

* spisok: Правило для обработки последовательности выражений, где:
  + spisok : /\* пустое правило \*/: Указывает, что последовательность может быть пустой.
  + spisok '\n': Указывает, что строка может заканчиваться новой строкой.
  + spisok wyrag '\n' { printf("\t%g\n", $2); }: Обрабатывает выражение и выводит его результат.
* wyrag: Правило для определения выражений, здесь оно переходит к правилу summa.
* summa: Обрабатывает операции сложения и вычитания:
  + summa : product: Указывает, что сумма может состоять из одного произведения.
  + summa '+' product { $$ = $1 + $3; }: Операция сложения.
  + summa '-' product { $$ = $1 - $3; }: Операция вычитания.
* product: Обрабатывает операции умножения и деления:
  + product : power: Указывает, что произведение может состоять из одной степени.
  + product '\*' power { $$ = $1 \* $3; }: Операция умножения.
  + product '/' power { $$ = $1 / $3; }: Операция деления.
* power: Обрабатывает операции возведения в степень:
  + power : unary: Указывает, что степень может состоять из одного унарного выражения.
  + unary '^' power { $$ = pow($1, $3); }: Операция возведения в степень.
* unary: Определяет унарные операции, включая унарный минус:
  + unary : factor: Указывает, что унарное выражение может быть простым фактором.
  + '-' factor { $$ = -$2; }: Обрабатывает унарный минус для отрицательных чисел.
* factor: Определяет факторы выражения:
  + factor : DATA { $$ = $1; }: Определяет, что фактор может быть числом.
  + factor : '(' wyrag ')' { $$ = $2; }: Обрабатывает скобки для изменения приоритета операций.

**Функции**:

* yyerror(char \*s): Функция для обработки ошибок, выводит сообщение об ошибке.
* yylex(): Лексический анализатор, который читает входные данные и возвращает токены:
  + Пропускает пробелы и табуляции.
  + Обрабатывает числа и возвращает их как токен DATA.

**Код программы**

%{

#define YYSTYPE double

#include <math.h>

#include <stdio.h>

#include <ctype.h>

int yyerror(char \*s);

int yylex();

%}

%token DATA

%%

spisok : /\* пустое правило \*/

| spisok '\n'

| spisok wyrag '\n' { printf("\t%g\n", $2); }

;

wyrag:

summa

;

summa:

product

| summa '+' product { $$ = $1 + $3; }

| summa '-' product { $$ = $1 - $3; }

;

product:

power

| product '\*' power { $$ = $1 \* $3; }

| product '/' power { $$ = $1 / $3; }

;

power:

unary

| unary '^' power { $$ = pow($1, $3); }

;

unary:

factor

| '-' factor { $$ = -$2; }

;

factor:

DATA { $$ = $1; }

| '(' wyrag ')' { $$ = $2; }

;

%%

int yyerror(char \*s)

{

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

return 0;

}

int yylex()

{

int c;

do {

c = getchar();

} while (c == ' ' || c == '\t');

if (c == EOF)

return 0;

if (c == '.' || isdigit(c)) {

ungetc(c, stdin);

scanf("%lf", &yylval);

return DATA;

}

return c;

}

int main()

{

printf("Введите выражение:\n");

yyparse();

return 0;

}

**Результат работы программы**

