Содержание

[Формулировка задачи 3](#_Toc107052660)

[Ход работы 4](#_Toc107052661)

[Вывод 7](#_Toc107052662)

# Формулировка задачи

Необходимо создать парсер и лексер полиномов. Данная комбинация должна проверять корректность подаваемых на вход полиномов, упрощать принимаемые выражения и соответствовать следующим требованиям:

1. Поддерживать удобную форму записи полиномов (2x^2 + 2x + 3).
2. Поддерживать следующие математические знаки бинарных операций: +, -, \* (/ по желанию).
3. Предоставлять красивый, корректный вывод упрощённых полиномов.

# Ход работы

В ходе создания комбинации парсера и лексера, которая в дальнейшем будет называться, в целях сокращения, «компилятором», полиномов, за основу было принято правило, раскладывающее одно выражение в два других с учётом знака между ними:

expression:

t\_Obracket expression t\_Cbracket

| expression t\_SignPlus expression

| expression t\_SignMinus expression

| expression t\_SignMulti expression

| variable

;

Как видно из примера, выражение может быть выражением в скобках, что позволяет порождать скобочные выражения или же может быть представимо в виде переменной. Переменная, в свою очередь, представляет собой либо константу, либо переменную (х), возможно имеющую как степень, так и коэффициент.

С целью корректной передачи аргументов и возможности упрощения выражения были созданы две структуры:

struct variableStruct

{

int coeff;

int degree;

};

struct expressionStruct

{

int isActive[32];

int count;

int coeffsArray[64];

int degreesArray[64];

};

Для того, чтобы данные структуры данных могли быть использованы как для терминалов, так и для нетерминалов, необходимо объявить объединение и применить его:

%union

{

struct expressionStruct exp;

struct variableStruct var;

int num;

char \* str;

}

%token<num> t\_DIGIT

%token<str> t\_SignPlus

%token<str> t\_SignMinus

%token<str> t\_SignMulti

%token<str> t\_END

%token<str> t\_Obracket

%token<str> t\_Cbracket

%type <var> variable

%type <exp> expression

%type <num> number

Как можно видеть, в объединении используются как новосозданные типы данных, так и стандартные типы данных языка Си. Данные типы данных имеют сокращения (нумерования) в объединении, после эти сокращения используются для указания типов данных терминалов и нетерминалов.

Как можно видеть на предыдущем примере, структура, созданная для переменной, имеет в себе только два поля – поле для степени и поле для коэффициента. Этого достаточно для полноценного описания любой отдельностоящей переменной.

Структура же для полинома имеет более сложный формат – используется два массива для тех же самых показателей и используется целочисленная переменная, учитывающая количество переменных в одном выражении.

Соответственно, данная грамматика позволяет порождать выражение, способное быть разложенным на выражение в скобках или же на комбинацию двух выражений со знаком между ними. Любая переменная однозначно сворачивается до выражения. Функции, реализованные для математических выражений, выполняют действия со структурами, позволяя изменять их.

Функции при наличии необходимости могут увеличивать выражения или же уменьшать их. Например, если складывается два полинома, имеющих различия в наборе степеней слагаемых, полином будет увеличен. При вычитании двух полиномов, в ситуации разности двух одинаковых слагаемых, полином будет уменьшен.

# Вывод

В процессе выполнения данной лабораторной работы были изучены принципы построения компилятора для математических полиномов. Созданные парсер и лексер проверяют корректность подаваемого выражения, а также выполняют его преобразование, упрощение.