Содержание

|  |  |  |  |
| --- | --- | --- | --- |
| Введение........................................................................................................ | |  | 2 |
| 1 | Ход работы........................................................................................ |  | 3 |
| 1.1 | Принципы, заложенные в дизайн языка......................................... |  | 3 |
| 1.2 | Описание грамматики ..................................................................... |  | 4 |
| 1.3 | Представление полиномов и реализация операций над ними на низком уровне................................................................................... |  | 5 |
| 1.4 | Все выводимые ошибки и примеры входных цепочек................. |  | 7 |
| 1.5 | Ошибки/предупреждения YACC.................................................... |  | 9 |
| Вывод............................................................................................................. | |  | 10 |
| Приложение 1  Приложение 2 | |  |  |
|  | |  |  |

введение

**Цель работы**: научиться проектировать и реализовывать собственный формальный язык, а также создать язык, реализующий арифметические операции над полиномами.

Требования к заданию:

1. Продуманный синтаксис;

2. Поддержка полиномов от разных переменных (x, y, z и т.п.);

3. Поддержка переменных, которым можно присваивать полиномы;

4. Поддержка комментариев;

5. Реализация сообщений об ошибках в развёрнутом виде и с указанием номера строки.

# ход работы

## Принципы, заложенные в дизайн языка

* В языке реализовано два типа команд: инициализация переменных и вывод переменных на экран
* В ходе исполнения программы, переменным могут быть присвоены новые значения
* В языке реализована работа со скобками, а также следующие операции по правилам их приоритета: +, -, \*, ^
* Язык не поддерживает работу с многочленами от разных переменных, но поддерживается тривиальное сокращение без ошибок: y-y+x+x
* Полиномы хранятся в памяти в виде динамического массива размера до 100 полиномов
* Язык обрабатывает следующие ошибки: Возведение полинома в отрицательную степень или в степень полинома степени больше нуля, возведение нуля в нулевую степень, попытка посчитать полином от разных переменных, по ходу вычисление производится проверка на целочисленное переполнение, проверка на переполнения массива полиномов, проверка на длину имени переменной

## Описание грамматики

Грамматика языка операций над полиномами состоит из небольшого количества правил, которые определяют имя переменной, присваивание полинома, а также непосредственно описывают операции между ними:

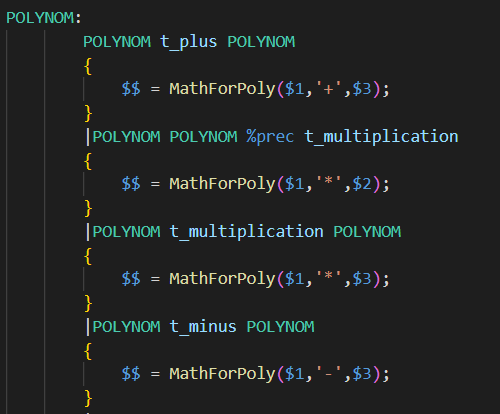


Рисунок 1 – главная составляющая грамматики.

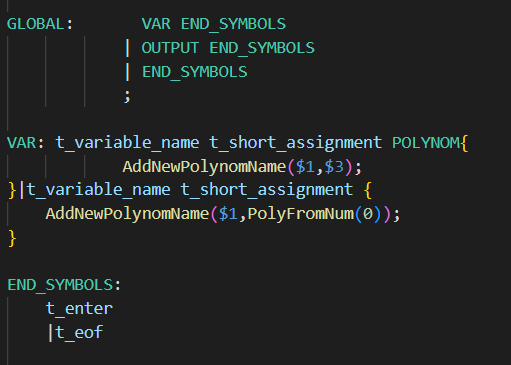


Рисунок 2 – присваивание полинома переменной.

## Представление полиномов и реализация операций над ними на низком уровне

Структура, составляющая полином выглядит следующим образом:

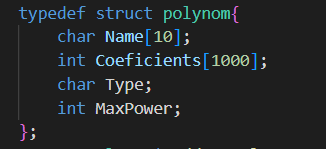


Рисунок 3 – структура полинома.

Поля структуры: имя переменной (выделено 10 символов), коэффициент (1000 знаков), имя неизвестной и степень.

Также объявлено объединение, хранящий в себе указатель на структуру полинома. Это сделано для того, чтобы программа могла оперировать этим типом данных в правилах грамматики.

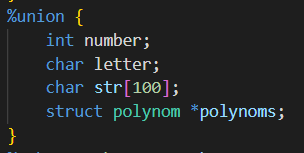


Рисунок 4 – объединение.

Реализованы следующие операции над полиномами:

* Сложение:

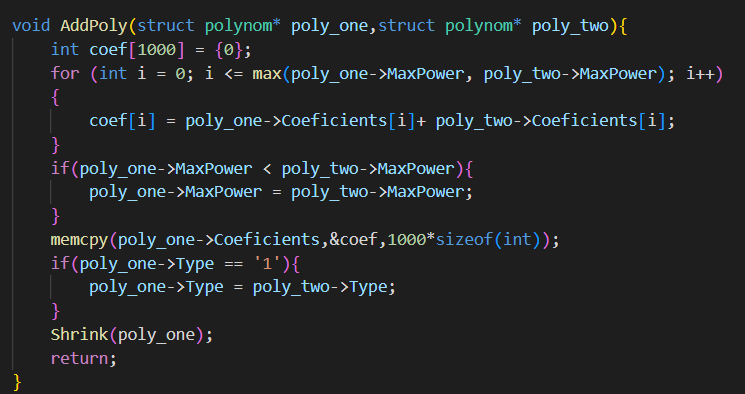


Рисунок 5 – операция сложения.

Происходит инициализация коэффициентов нулями, затем в цикле до значения максимальной степени полиномов складываются их коэффициенты. Если степень первого слагаемого меньше второго, то результирующему полиному присваивается степень второго. Далее может быть 4 пары передаваемых параметра: x+x, x+1, 1+x, 1+1. Следующая проверка смотрит тип следующего полинома. Первый, второй и четвертый варианты без этой проверки пройдут, так как в первом и четвертом два одинаковых типа, во втором запись идет в первый полином и его тип измениться не может, а в третьем эта проверка будет нужна, так как если первый тип == 1, то второй тип или x или 1. После этого вызывается функция Shrink(), убирающая метку полином, если тот при вычислении приравнялся нулю или все его переменные сократились.

* Операция вычитания аналогична операции сложения.
* Умножение:

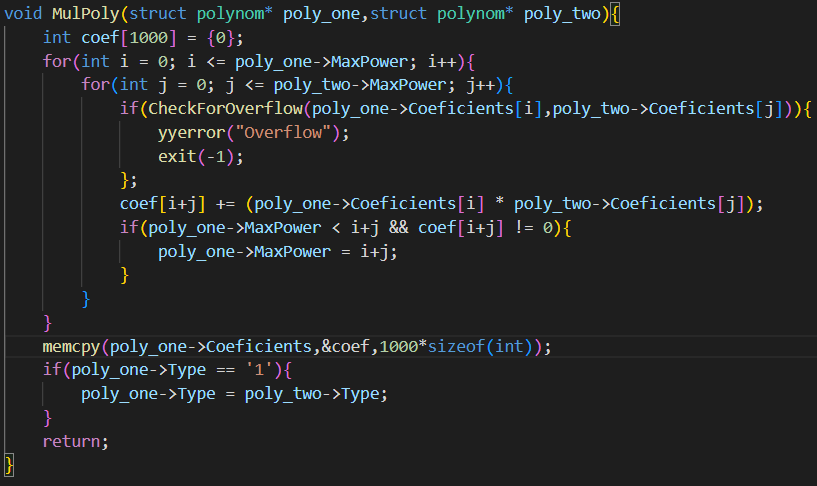


Рисунок 6 – операция умножения.

Происходит проверка на переполнение значений, а затем перемножение коэффициентов и запись новой максимальной степени полинома.

* Возведение в степень.

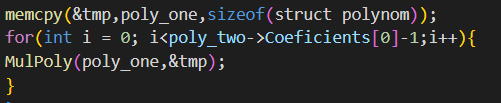


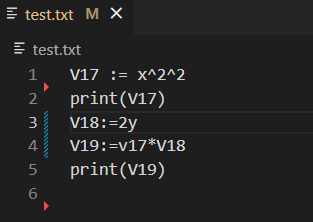
Рисунок 7 – операция возведения в степень.

Используется операция умножения полинома на самого себя с помощью вышеописанной функции MulPoly().

## Все выводимые ошибки и примеры входных цепочек

При вводе некорректных данных программа обрабатывает следующие ошибки и выводит соответствующие сообщения:

1. Разные имена переменных.



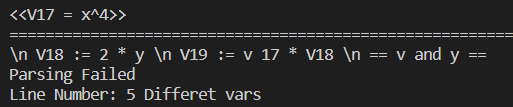


Рисунок 3 – разные имена переменных.

1. Возведения нуля в нулевую степень.

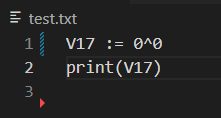
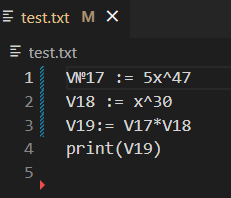




Рисунок 4 – обработка неопределенности.

1. Неизвестный символ в имени переменной.



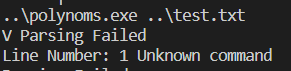


Рисунок 5 – неизвестная команда.

1. Использование неинициализированной переменной

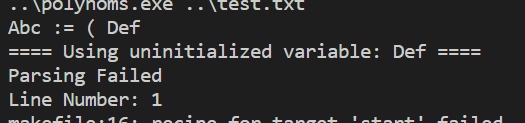


Рисунок 6 – неинициализированная переменная

1. Переполнение коэффициентов (x^1001)

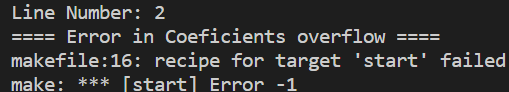


Рисунок 7 – переполнение коэффициентов

1. Переполнение памяти переменных

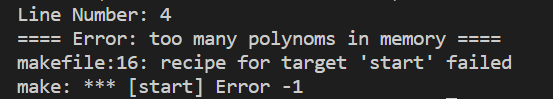


Рисунок 8 – переполнение памяти

1. Слишком длинное имя переменной

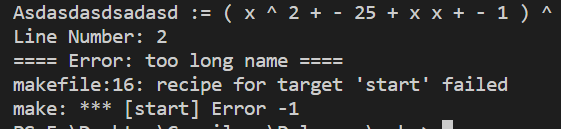


Рисунок 9 – слишком длинное имя

## Ошибки/предупреждения YACC

На момент завершения работы было выявлено 32 конфликта shift/reduce и 8 конфликтов reduce/reduce. С помощью файла y.output можно рассмотреть причину этих конфликтов.

Большая часть конфликтов (в частности, все конфликты reduce/reduce), возникает из-за правил типа POLYNOM «знак» POLYNOM, так как возникает неоднозначность сдвига:

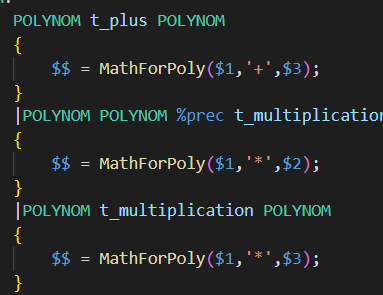


Рисунок 10 – правила, вызывающие конфликты.

вывод

В результате выполнения лабораторной работы были получены навыки создания собственного формального языка, а именно язык операций над полиномами. Было реализовано 4 вида операций: сложение, вычитание, умножение и возведение в степень.

Для реализации такого рода задачи была написана грамматика, разработан способ хранения полиномов в памяти и реализованы функции, выполняющие непосредственно операции над полиномами. В процессе реализации программы была написана лексическая часть и синтаксическая с описанием анализа входного потока.

В процессе разработки грамматики возникали конфликты, связанные с правилом POLYNOM, однако на момент завершения работы решить эти конфликты так и не удалось, так как от них напрямую зависит корректность обработки последовательности ввода и функциональность программы.

Однако возможности созданного формального языка могут быть увеличены, так как в нем еще не обрабатываются такие детали, как деление полиномов, взятие остатка от деления, работа с несколькими переменными.

Несмотря на существующие конфликты и ограниченный функционал, реализованная программа успешно справляется с поставленными задачами.