|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ | | | | | | | | |  |
|  | | | |  | | |  | | | |
|  | | | ФГАОУ ВО «Пермский государственный национальный исследовательский университет» | | | | |  | | |
|  | | | |  | | |  | | | |
| ОТЧЕТ  по заданию № 1 «Модуль ввода-вывода и лексический анализатор»  по дисциплине «Формальные грамматики и методы трансляции» | | | | | | | | | | |
|  | | | |  | | |  | | | |
|  | | Работу выполнил  Студент гр.ПМИ-1,2  Проскуряков К.А.\_\_\_\_\_  ≪\_\_\_≫\_\_\_\_\_\_\_\_\_\_\_2021 | | |  | Проверил  Ассистент кафедры МОВС  Пономарёв Ф.А.\_\_\_\_\_\_  ≪\_\_\_≫\_\_\_\_\_\_\_\_\_\_\_2021 | | |  | |
|  |  | | |  | | |  | | |  |
|  |  | | | Пермь 2021 | | |  | | |  |

# Постановка задачи

Необходимо разработать модуль ввода-вывода и лексический анализатор компилятора для языка Pascal. Модуль ввода-вывода должен получать на вход путь до файла, в котором записана программа на языке программирования Pascal и путь до выходного файла, в который будет записываться листинг программы с обнаруженными ошибками. Лексический анализатор должен обращаться к модулю ввода-вывода для получения непрочитанных символов и строить их них лексемы, а также сообщать модулю ввода-вывода об обнаруженных ошибках.

Необходимо реализовать поддержку следующих типов лексем:

* Идентификатор
* Операция (в том числе ключевые слова)
* Константы (целочисленные, числа с плавающей точкой и символы)

# Проектирование

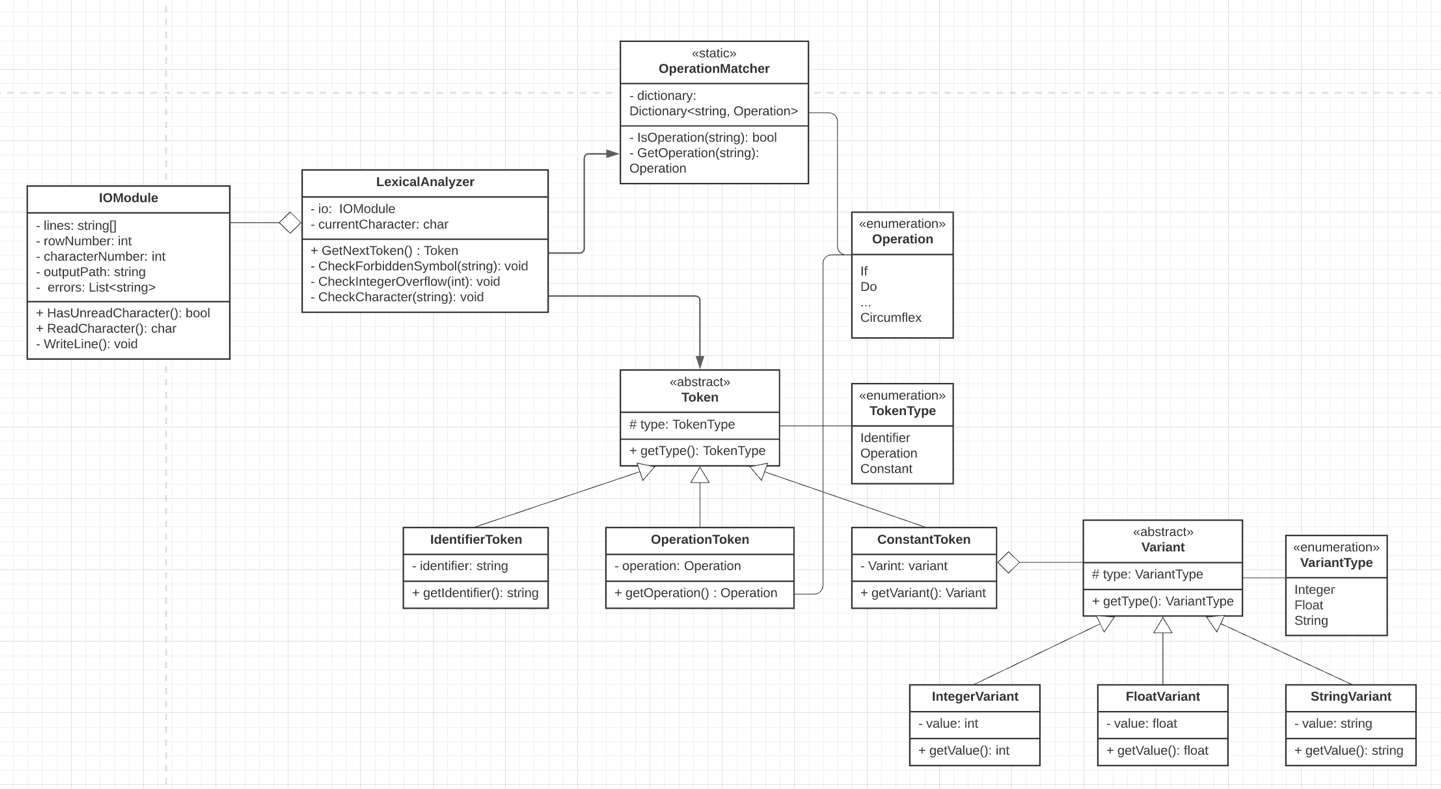
В ходе проектирования было решено сделать класс, представляющий лексему абстрактным, и создать 3 производных класса для каждого типа лексем. Тип лексемы хранится в базовом классе в виде значения специально созданного перечисления типов лексем.

Далее, поскольку константа может иметь один из трёх типов, было решено также сделать этот класс абстрактным и наследовать от него 3 новых класса. Для хранения типа константы поступили также как и в прошлом случае, создав поле типа специального перечисления.

Модуль ввода читает следующий непрочитанный символ и передаёт его лексическому анализатору. Если все символы текущей строки прочитаны, модуль должен вывести строку и все обнаруженные в ней ошибки в файл вывода.

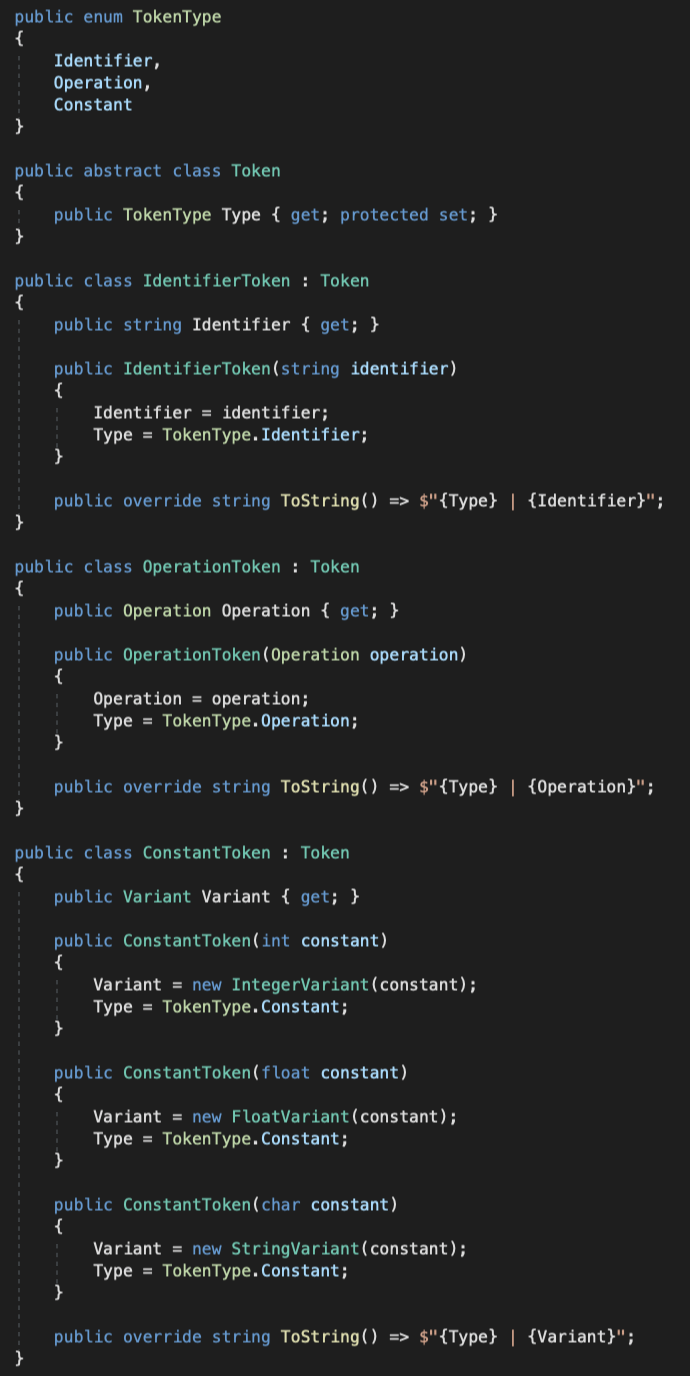
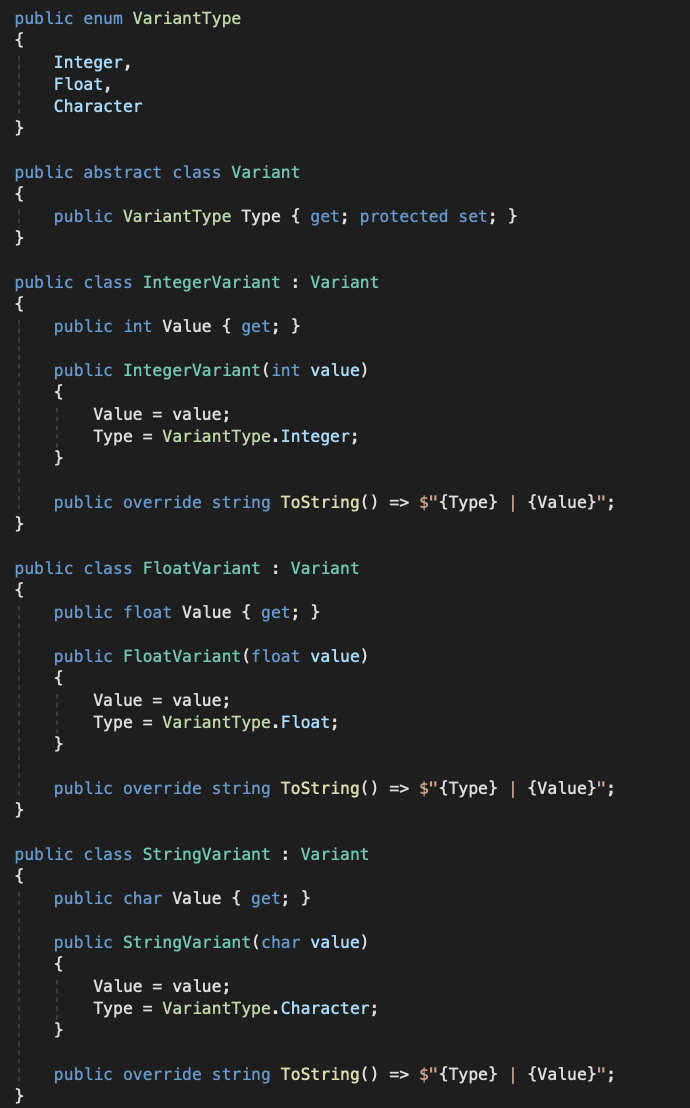
Лексический анализатор обращается к модулю ввода-вывода для получения следующего непрочитанного символа. Далее он строит из них лексемы. Для проверки того, что лексема является операцией или ключевым словом, а также получения соответствующего кода, лексический анализатор обращается к соответствующему статическому классу. Если в ходе работы анализатора обнаруживаются лексические ошибки, анализатор сообщает об этом модулю ввода-вывода.

Диаграмма классов модуля изображена на рисунке 1.

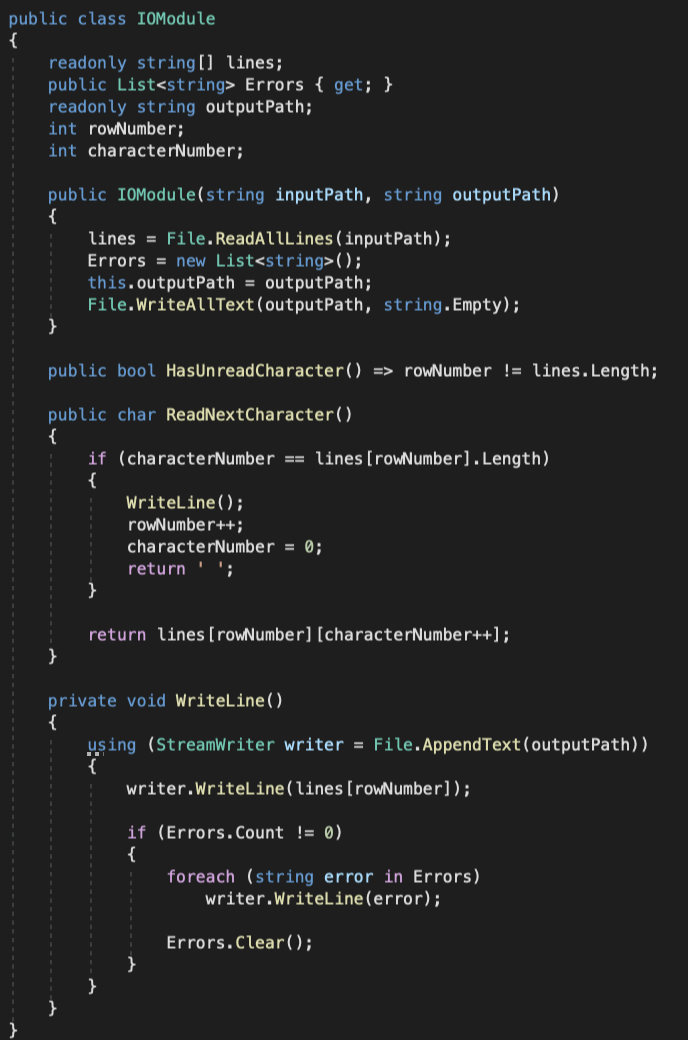
** Рисунок 1 – диаграмма классов**

# Реализация

Сначала был реализован класс Variant, описывающий константу и его производные классы. Далее был написана иерархия класса Token, описывающего лексему.

**** ****

Далее был реализован класс IOModule. Этот класс имеет конструктор, принимающий путь до файла с программой, из которой он читает весь текст и путь до выходного файла. Были реализованы метод проверки, остался ли в тексте непрочитанный символ и метод для чтения следующего символа. Также был написан метод, выводящий обработанную строку и найденные в ней ошибки.



Далее было создано перечисление со всеми ключевыми словами и операциями языка Pascal, а также статический класс, в котором есть статическое поле – словарь, с ключом в виде строки и значением в виде значения этого перечисления, соответствующего ключу. Также в этом классе есть 2 публичных метода – проверка, является ли полученная строка ключевым словом и возвращение кода операции строки.

Больше всего трудностей вызывала реализация класса LexicalAnalyzer. В этом классе есть один публичный метод, который должен возвращать следующую лексему программы, или null, если лексем не осталось. Для реализации этого метода было добавлено символьное поле – предыдущий прочитанный символ. Первым делом этот метод обращается к методу модулю ввода-вывода для проверки, остался ли в тексте программы хотя бы один непрочитанный символ. Если символов не осталось, возвращается null. В противном случае, мы читаем символы до тех пор, пока не встретили любой символ, отличный от пробела. Если таким символом оказался символ, открывающий комментарий – игнорируем все символы до конца этого комментария.

Если текущий символ – буква или нижнее подчёркивание, тогда читаем и добавляем символы в текущую лексему до тех пор, пока не встретили пробел или первый встреченный оператор (например, +, :, < и т.д.). Проверяем, не содержит ли лексема запрещенных символом, если содержит, то сообщаем об этом модулю ввода-вывода. Далее проверяем, является ли прочитанная лексема оператором. Если является, возвращаем лексему оператора с соответствующим кодом, иначе возвращаем лексему идентификатора.

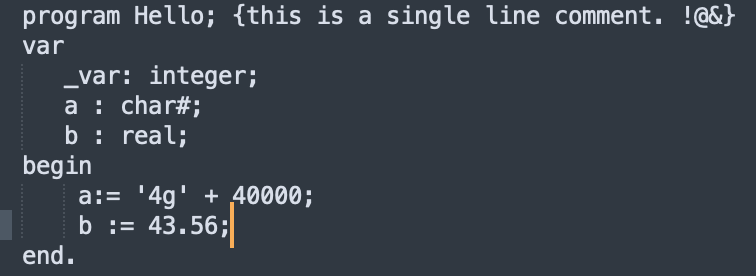
Если текущий символ – цифра, также читаем и добавляем символы в текущую лексему до тех пор, пока не встретили пробел или первый встреченный оператор, игнорируя «.». Если в лексеме содержатся символы «.», «Е» или «е» — значит это число с плавающей точкой, возвращаем соответствующую лексему. Иначе число целочисленное. Для него проверяем, не выходит ли его значение за максимальное допустимое значение типа integer (32768), и, если превышает – сообщаем модулю ввода-вывода.

Если текущий символ – «’», значит текущая лексема символьная константа. Читаем всю константу, и, если её длина не равна 1, сообщаем об этом модулю ввода-вывода.

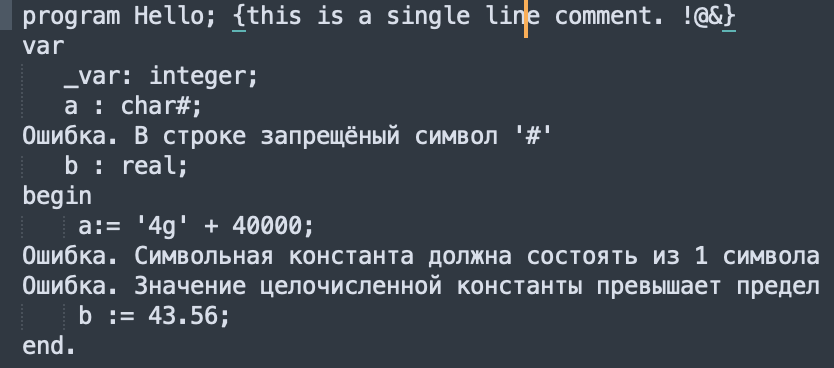
Иначе текущая лексема будет оператором. Поскольку в языке Pascal оператор может состоять из нескольких литер (например, :=, <> и т.д.) необходимо прочитать также следующий символ.

# Тестирование

Входной файл



Выходной файл



Результат разбиения на лексемы:

