|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | Министерство науки и высшего образования РФ | | | | | | |  | | |
|  | | | | |  | | |  | | | | |
|  | | ФГБОУ ВПО «Пермский государственный национальный исследовательский университет» | | | | | | | | |  | |
|  | | | | |  | | |  | | | | |
|  | | | | Отчет  по индивидуальной работе «Создание Pascal компилятора» по дисциплине «Формальные грамматики и методы трансляции» | | | | |  | | | |
|  | | | | |  | | |  | | | | |
|  | Работу выполнил  студент гр. ПМИ-3, 3 курс  Власов А.А. \_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_2021 | | | | |  | Проверил  ассистент кафедры МОВС  Пономарев Ф. А. \_\_\_\_\_\_\_\_\_  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_2021 | | | | |  |
|  | | | | |  | | |  | | | | |
|  | | | | | Пермь 2021 | | |  | | | | |

Содержание

[Модуль ввода-вывода 3](#_Toc88073813)

[Постановка задачи 3](#_Toc88073814)

[Проектирование 3](#_Toc88073815)

[Тесты 4](#_Toc88073816)

[Лексический анализатор 5](#_Toc88073817)

[Постановка задачи 5](#_Toc88073818)

[Проектирование 5](#_Toc88073819)

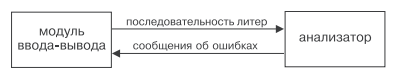
[Тесты 5](#_Toc88073820)

[Синтаксический анализатор 7](#_Toc88073821)

# Модуль ввода-вывода

## Постановка задачи

Модуль ввода-вывода считывает последовательность литер исходной программы с внешнего устройства и передает их анализатору. Анализатор проверяет, удовлетворяет ли эта последовательность литер правилам описания языка, и формирует (в случае необходимости) сообщения об ошибках. Их взаимодействие можно описать в виде следующей схемы



## Проектирование

Исходный текст программы будем считывать по 1 строке из файла, далее из буфера считываются отдельные литеры, за это отвечает метод **getNextLetter**. Также модуль ввода-вывода должен принимать, обрабатывать и хранить информацию об ошибках, их местоположении относительно кода исходной программы. На рисунке 1 представлена схема класса **InputOutputModule**. Все ошибки будем хранить в виде списка, запоминая их позицию в тексте исходной программы. Сообщения об ошибках будем выводить в конце работы компилятора.

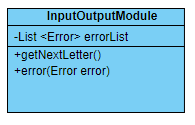


Рис.1

Для ошибки создадим отдельный класс Error (рисунок 2).

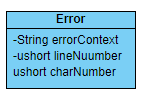


Рис.2

Также на основе WPF создадим простой UI компилятора (рисунок 3), для удобства дальнейшей работы и тестирования. При создании UI я вдохновлялся (копировал) программой PascalABC.NET 3.8.2.

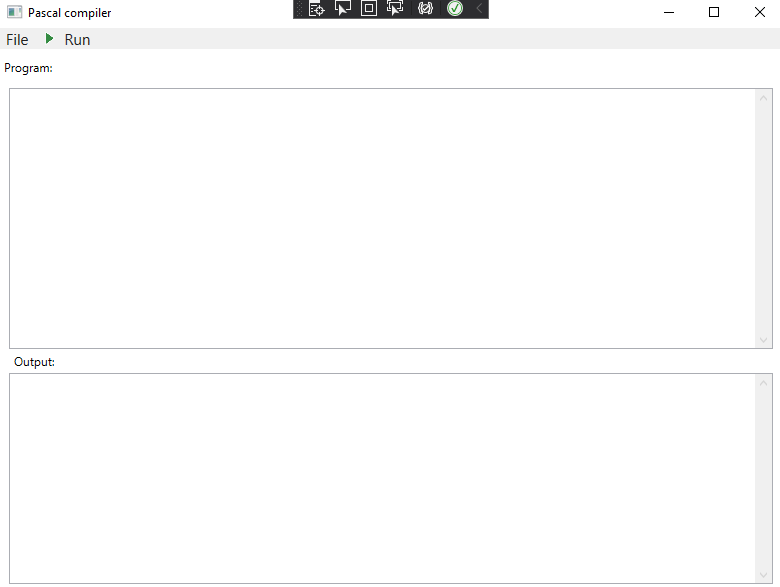
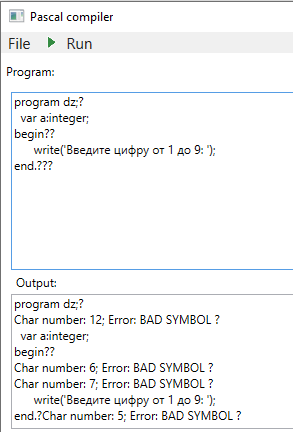
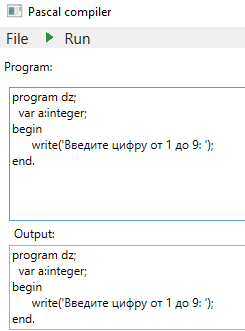


Рис.3

## Тесты

После написания программы нам необходимо проверить на работоспособность 2 основные функции getNextLetter и error. Для этого в окно результата будем выводить литеры, полученные с помощью getNextLetter, и в это же окно будем выводить информацию об ошибках в строке, т.к. модуль ввода вывода сам никак не может определить ошибки, то мы искусственно будем вызывать ошибки при обнаружении символа <?>. Результаты:

Вывод: модуль ввода-вывода разработан, работает корректно.

# Лексический анализатор

## Постановка задачи

Лексический анализатор формирует символы исходной программы и строит их внутреннее представление. Он получает от модуля ввода-вывода поток литер, формирует из них контекстные лексемы (токены, которые содержат тип лексемы). Далее (в зависимости от типа, полученного токена) обрабатывает и передаёт их в следующие модули компилятора.

## Проектирование

Мы выделим 3 типа токенов (Класс CToken), это: ttOper (операторы и ключевые слова), ttIdent (идентификаторы констант, типов, переменных, процедур и функций) и ttValue (значения констант).

ttOper будет хранить тип оператора или ключевого слова, которое он представляет.

ttIdent содержит в себе только имя пользовательского идентификатора.

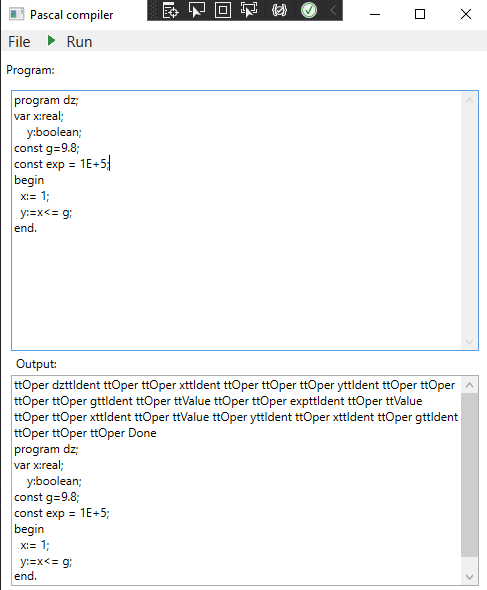
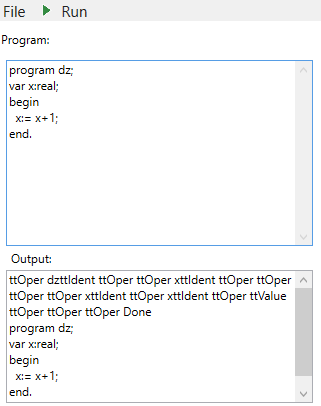
ttValue содержит значение константы, а также её тип (целое/вещественное/символ/строка).

Лексический анализатор также полностью исключает все комментарии, пробелы и прочие разделители, системные символы, а также определяет переполнения для целых/вещественных чисел. Сам же лексический анализатор устроен крайне просто, он запрашивает литеры у модуля ввода-вывода, и в зависимости от того что получил (через оператор switch), возвращает определённый токен (функция GetNextToken).

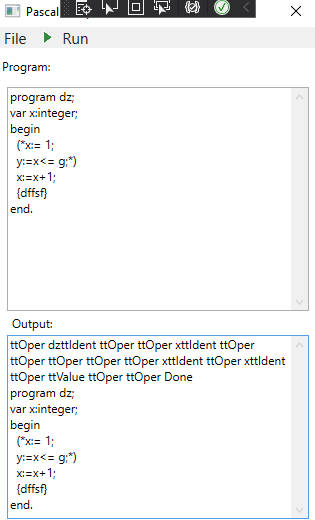
## Тесты

Если ошибок не обнаружено будем выводить тип получаемого токена, если есть ошибки, то исходный код с комментариями об ошибке.

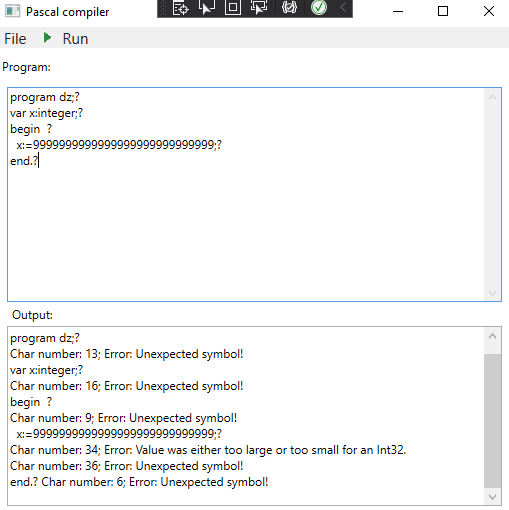
Нет ошибок:



Обработка комментариев, без них ровно 17 токенов:



Ошибки:



# Синтаксический анализатор