|  |
| --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  «Пермский государственный национальный исследовательский университет» |

|  |
| --- |
| Пермь 2021 |

**1. Задание**

Разработать компилятор для программ на языке Pascal. Язык программирования, на котором должен быть написан компилятор, должен быть объектно-ориенитрованным.

**Проектирование**

Компиляция состоит из следующих фаз: модуль ввода-вывода, анализатор, генератор. Анализатор, в свою очередь, состоит из 3 модулей: лексический, синтаксический и семантический анализаторы. Весь компилятор принимает следующую структуру:

Модуль ввода-вывода получает исходный код программы и преобразует в последовательность литер

Лексический анализатор осуществляет лексический анализ, строит идентификаторы, ключевые слова, разделители, числа

Синтаксический анализатор проверяет, удовлетворяет ли программа формальным правилам

Семантический анализатор проверяет смысловую правильность программы

Генератор генерирует объектную программу

**Модуль ввода-вывода**

Модуль ввода-вывода содержит класс IO

public IO(string path).

В конструктор класса IO передается путь к файлу с текстом программы на языке Pascal, в строковое поле ProgramText.

public string ProgramText { get; private set; }

считывается текст всей программы построчно. Из модуля ввода-вывода лексический анализатор берет очередной символ, для этого есть метод. В данном методе обновляется позиция символа по порядку, по номеру строки и позиция в самой строке

public char Nextch()

Для проверки лексем с длиной более 1 необходимо брать следующий символ. В случае, если он не подошел к лексеме, нужно вернуться на символ обратно, для этого есть метод

public void Back()

**Лексический анализатор**

Я разделил лексемы на идентификаторы, ключевые константы, целые и вещественные числа, разделители, символы операции, спецификаторы. Конструктор анализатора получает экземпляр класса ввода вывода

public Lexical(IO Input\_Reader)

Лексема характеризуется типом, значением, номером строки и положением в строке.

Разные типы лексем имеют разные типы значения. Для удобного хранения лексем было организовано наследование классов.

Базовый класс – Лексема

Производные: Вещественные, целые, идентификаторы, ключевые слова, разделители, операции, спецификаторы

public class Lexem

{

public int LinePosition;

public int LineNumber;

public string type;

public Lexem (int pos, int num, string input\_type)

{

LineNumber = num;

LinePosition = pos;

type = input\_type;

}

};

Производные классы содержат поле value соответствующего типа (вещественные, целые, строковые)

Получение следующей лексемы происходит в методе

public Lexem NextSym(out string raw\_value)

Здесь создается экземпляр класса Лексема. Реализация полиморфизма для использования лексем под одним типом «Лексема» будет добавлена позже. Сейчас все значения имеют строковый формат. Метод возвращает экземпляр класса Лексема, а значение лексемы передается по ссылке.

Определение идентификатора: первый символ из латинского алфавита, либо нижнее подчеркивание. Остальные могут быть теми же + цифры.

Определение ключевого слова: все символы должны быть латинскими

Определение целочисленной лексемы: все символы – цифры

Определение вещественной лексемы: если 1 символ лексемы – цифра, а в процессе проверки на целочисленную лексему была найдена точка

Определение операции, разделителя, спецификатора: если список операций содержит данный набор символов

Идентификаторы и ключевые слова начинаются с латиницы, константы с цифры.

Вывод всех лексем в цикле в методе

public void PrintLexem()

Добавить сортировку списка ошибок по порядку в коде