Министерство образования Республики Беларусь Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ

И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАТИКИ

Отчёт по лабораторной работе №2

По теме “Методы трансляции”

Выполнила: студентка гр. 053501 Шурко Т.А.

Проверил: ассистент кафедры информатики Гриценко Н. Ю.

Минск 2023

Содержание

[1. Цель работы 3](#_Toc7030)

[2. Теория 4](#_Toc7031)

[3. Программа и комментарии 7](#_Toc7035)

Примечание. Код программ 8

# Цель работы

Освоение работы с существующими синтаксическими анализаторами. Разработать свой собственный синтаксический анализатор, выбранного подмножества языка программирования.

Построить синтаксическое дерево.

Определить минимум 4 возможных синтаксических ошибки и показать их корректное выявление.

Основной целью работы является написание сценариев, которые задают синтаксические правила для выбранного подмножества языка.

В качестве анализируемого подмножества языка программирования будет использован язык программирования Pascal.

Для написания анализатора использован язык программирования

Python. Рассмотрены возможности его библиотеки ply.yacc.

Ниже представлен код программы на языке Pascal:

var i, j, counter: integer; text: string;

begin

i := 5; j := i + 7; text := 'abc'; for counter := i to j do begin

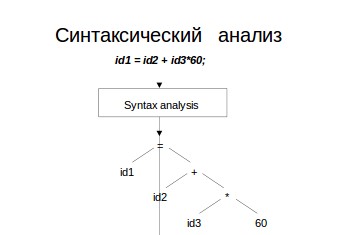
text := text + '\_\*\_'; writeln(text); end;

writeln(j); writeln(text); end.

# Теория

**Синтаксиический анализ** — это процесс сопоставления линейной последовательности лексем (слов, токенов) естественного или формального языка с его формальной грамматикой. Результатом обычно является дерево разбора (синтаксическое дерево). Обычно применяется совместно с лексическим анализом.

**Синтаксический анализатор**— это программа или часть программы, выполняющая синтаксический анализ.



## Пример разбора выражения в дерево

Задача синтаксического анализатора – проверить правильность записи выражения и разбить его на лексемы. Лексемой называется неделимая часть выражения, состоящая, в общем случае, из нескольких символов.

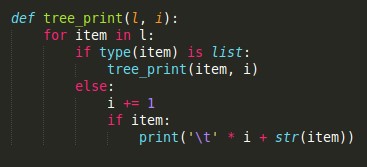
Результатом синтаксического анализа является синтаксическое строение предложения, представленное либо в виде дерева зависимостей, либо в виде дерева составляющих, либо в виде некоторого сочетания первого и второго способов представления.

## Типы алгоритмов

* Нисходящий парсер — продукции грамматики раскрываются, начиная со стартового символа, до получения требуемой последовательности токенов, им соответствуют LL-грамматики
* Восходящий парсер — продукции восстанавливаются из правых частей, начиная с токенов и кончая стартовым символом, им соответствуют LR-грамматики.

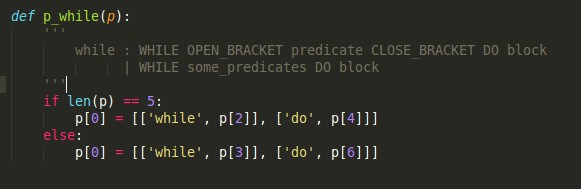
# Программа и комментарии

На Рис. 3.1. показана функция, используемая для построения синтаксического дерева.



**Рис. 3.1. Функция для построения синтаксического дерева**

Во время выполнения данной работы использовали уже написанный лексический анализатор. Из выражений, уже выделенных, составлялись синтаксические конструкции по некоторым правилам Рис.3.2.



**Рис. 3.2. Одно из правил для построения дерева**

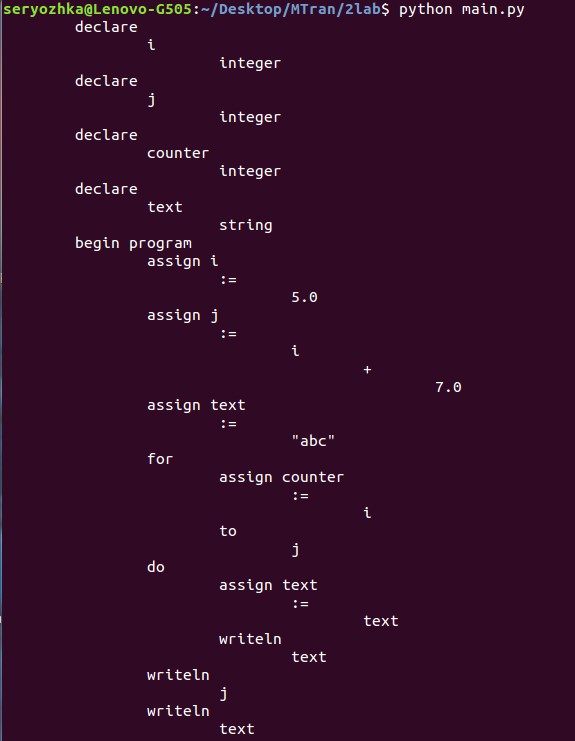
Данное правило говорит, что конструкция while в языке Pascal имеет 2 формы:

1. Только одно правило (конструкция predicate, которая имеет свои собственные формы)
2. Несколько правил (конструкция some\_predicates, которая представляет собой объединение нескольких конструкций predicate, объединенных ключевыми словами and или or)

Код со всеми правилами построения представлен в листинге 1 в конце отчёта.

**4. Результаты выполнения**

В результате дерево программы имеет следующий вид Рис.4.1

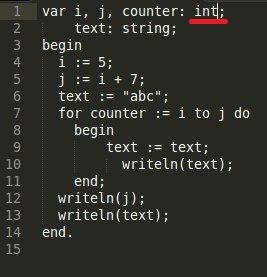


**Рис. 4.1. Построенное дерево анализируемой программы**

**Код с ошибками:**

Рассмотрим тот же код программы с добавленными в него ошибок. При обнаружении их происходит вывод уведомления об ошибке.

Код программы с 1-ой ошибкой Рис.4.4.



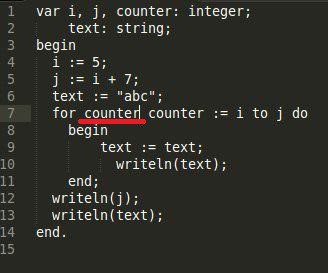
**Рис. 4.4. Синтаксическая ошибка объявления типа переменной**

Результат работы программы с допущением 1-ой ошибки Рис.4.5.



**Рис. 4.5. Результат работы программы с допущением 1-ой ошибки**

Код программы со 2-ой ошибкой Рис.4.6.



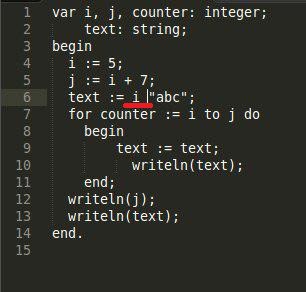
**Рис. 4.6. Синтаксическая ошибка конструкции for**

Результат работы программы с допущением 2-ой ошибки Рис.4.7.



**Рис. 4.7. Результат работы программы с допущением 2-ой ошибки**

Код программы с 3-ей ошибкой Рис.4.8.



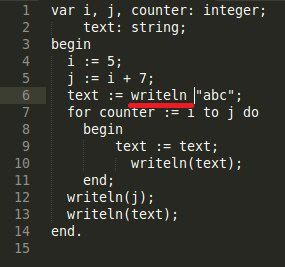
**Рис. 4.8. Синтаксическая ошибка №3**

Результат работы программы с допущением 3-ей ошибки Рис.4.9.



**Рис. 4.9. Результат работы программы с допущением 3-ой ошибки**

Код программы с 4-ой ошибкой Рис.4.10.



**Рис. 4.10. Синтаксическая ошибка неправильный вызов функции**

Результат работы программы с допущением 4-ой ошибки Рис.4.11.



**Рис. 4.11. Результат работы программы с допущением 4-ой ошибки**

## **Вывод**

В результате работы были получены знания о синтаксических деревьях, способах их построения. Для выделения синтаксических структур из кода был использован нисходящий парсер. В итоге работы был простроен простой синтаксический анализатор на основе уже имеющегося лексического с помощью задания правил для синтаксических конструкций, который способен не только строить и выводить синтаксическое дерево, но и находить ошибки, уведомлять о них, генерируя исключения, выводя их на консоль.

В ходе работы изначальная версия анализатора дорабатывалась для улучшения работы парсера. Была заложена основа для создания компилятора для языка Pascal в последующих лабораторных работах.

Конечная версия программы может находить и анализировать ошибки основного синтаксиса. При расширении набора выражений для поиска конструкций языка, анализатор будет способен обрабатывать все необходимые программы на языке, вплоть до создания таблицы токенов и поиска ошибок в полной версии языка. Но данная задача представляется достаточно трудной, из-за большого разнообразия библиотек языка, а так же использовании паттернов.