Белорусский государственный университет информатики и радиоэлектроники

Кафедра информатики

Лабораторная работа № 2

**Синтаксический анализатор**

Выполнил студент. гр. 453504: Ровдо Н.Р.

Проверил ассистент КИ: Шиманский В.В.

Минск, 2017

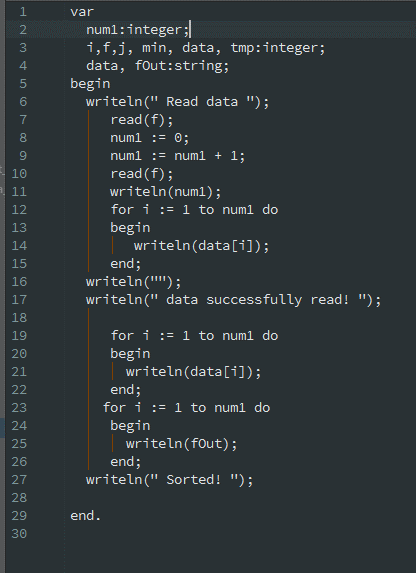
**1.Постановка задачи**

В данной работе ставится задача исследовать область синтаксических анализаторов, рассмотреть существующие аналоги и написать свой собственный синтаксический анализатор, выбранного подмножества языка программирования. Построить синтаксическое дерево.

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

Для написания анализатора использован язык программирования C#.

На рис. 1.1 показан код программы на языке Pascal:

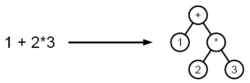


**Рис. 1.1. Исходный код программы на языке Pascal**

**2.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.



**Рис.2.1. Пример разбора выражения в дерево**

В ходе синтаксического анализа исходный текст преобразуется в структуру данных, обычно — в дерево, которое отражает синтаксическую структуру входной последовательности и хорошо подходит для дальнейшей обработки.

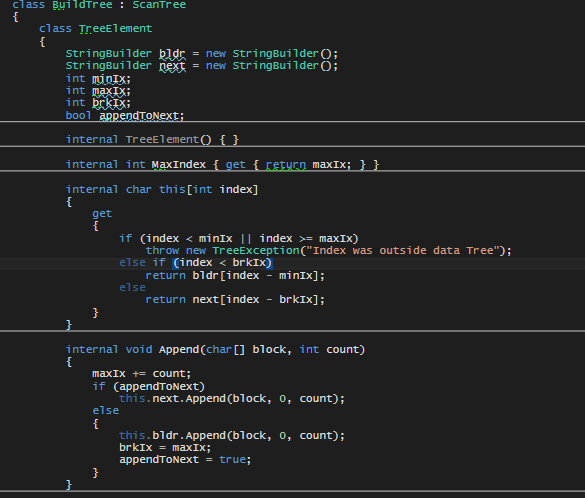
Как правило, результатом синтаксического анализа является синтаксическое строение предложения, представленное либо в виде дерева зависимостей, либо в виде дерева составляющих, либо в виде некоторого сочетания первого и второго способов представления.

Типы синтаксических анализаторов:

1. **LL-анализатор** (***LL parser***) — в информатике [нисходящий](https://ru.wikipedia.org/wiki/%D0%9D%D0%B8%D1%81%D1%85%D0%BE%D0%B4%D1%8F%D1%89%D0%B8%D0%B9_%D1%81%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7) [синтаксический анализатор](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7) для подмножества [контекстно-свободных грамматик](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%BD%D0%BE-%D1%81%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0). Он анализирует входной поток слева направо, и строит [левый вывод](https://ru.wikipedia.org/w/index.php?title=%D0%9B%D0%B5%D0%B2%D1%8B%D0%B9_%D0%B2%D1%8B%D0%B2%D0%BE%D0%B4&action=edit&redlink=1) грамматики. Класс грамматик, для которых можно построить LL-анализатор, известен как [LL-грамматики](https://ru.wikipedia.org/wiki/LL-%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0).
2. **LR-анализатор** (***LR parser***) — [синтаксический анализатор](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%82%D0%BE%D1%80) для [исходных кодов](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) программ, написанных на некотором [языке программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), который читает входной поток слева направо и производит наиболее правую продукцию [контекстно-свободной грамматики](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%BD%D0%BE-%D1%81%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0).

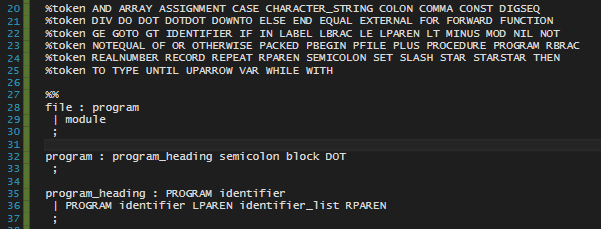
**3. Программа и комментарии**

На рис. 3.1. показан класс, используемый для построения синтаксического дерева:



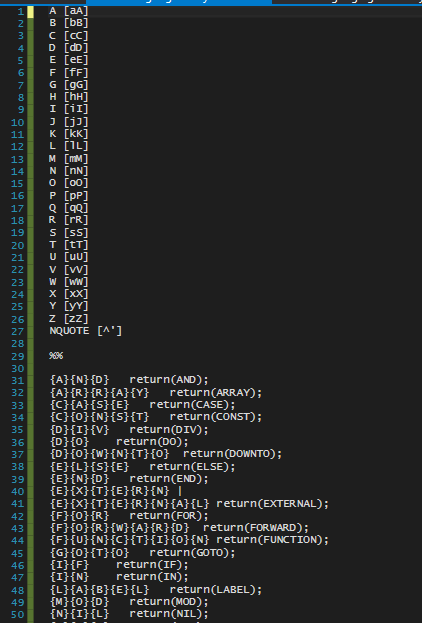
**Рис. 3.1. Класс и основные методы для построения синтаксического дерева**

Во время выполнения работы использован синтаксический анализатор yaccandLex. На основе шаблонов выделяются синтаксические правила языка, которые и составляют дерево , например (Рис. 3.2.):

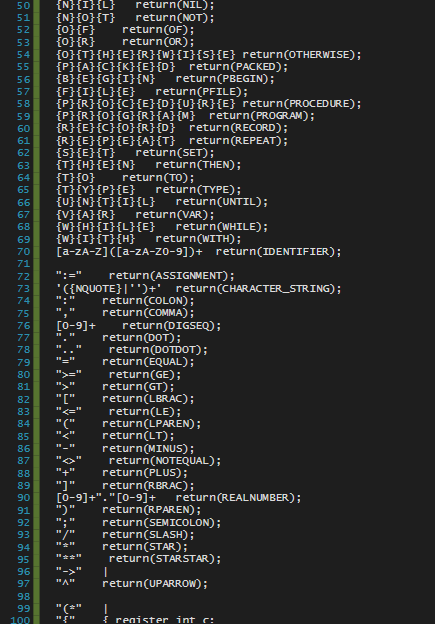


**Рис. 3.2. Правило синтаксической конструкции program и некоторых ключевых слов**

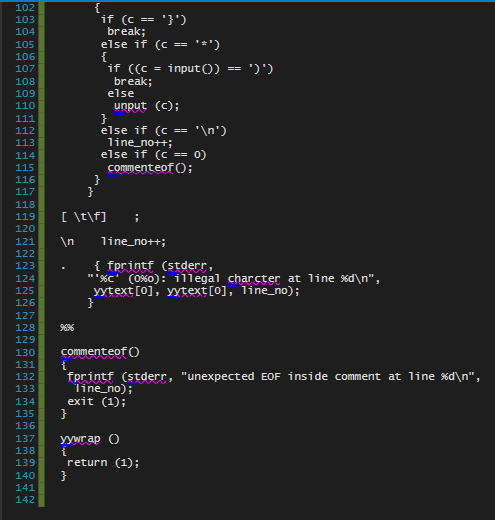
Все правила (Рис. 3.3.-3.6.):



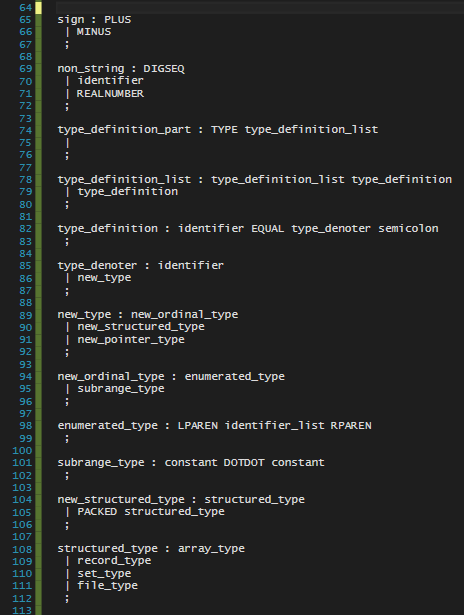
**Рис. 3.3. Правила, описывающие лексемы**

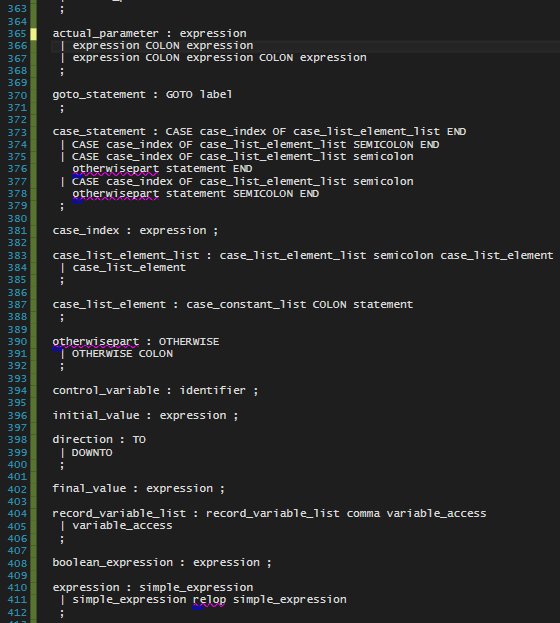


**Рис. 3.4. Правила, описывающие лексемы**

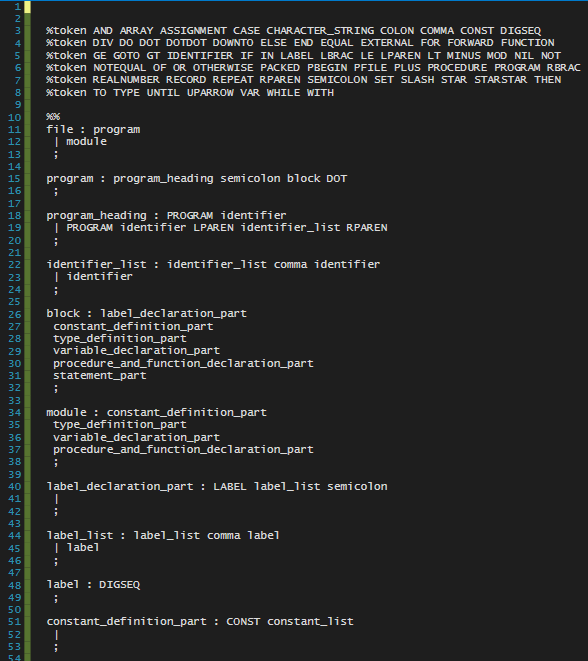


**Рис. 3.5. Правила, описывающие лексемы**

**Рис. 3.6. Синтаксические правила** 



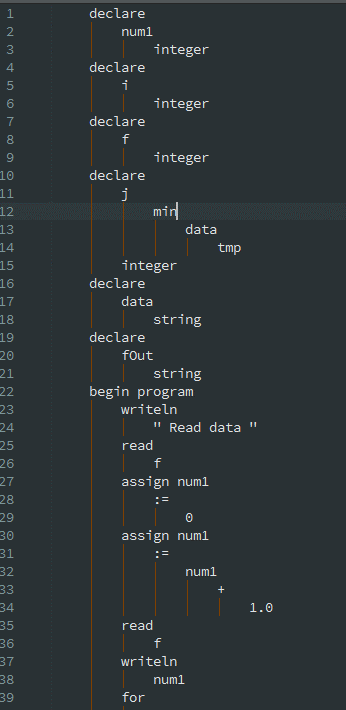
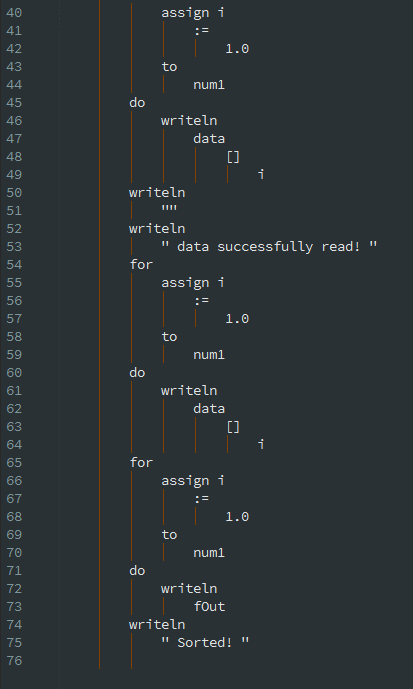
**Рис. 3.7. Синтаксические правила**



**Рис. 3.8. Синтаксические правила**

**4. Результаты выполнения**

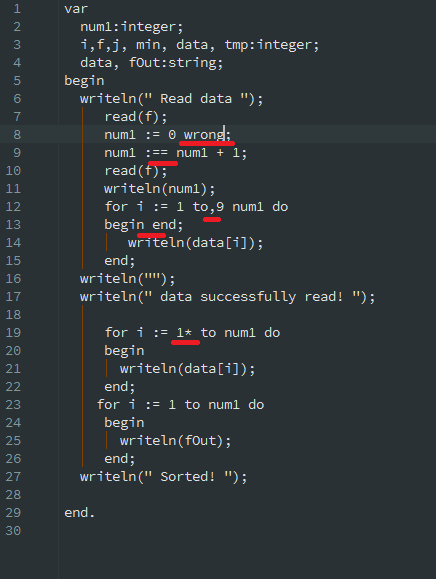
В результате дерево программы имеет следующий вид (Рис. 4.1.-4.4):

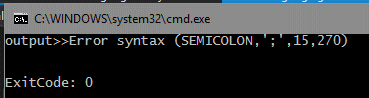


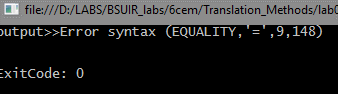
**Рис. 4.1-4.2. Дерево программы**

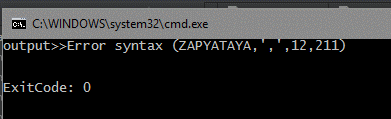
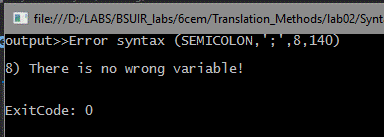
**Код с ошибками:**

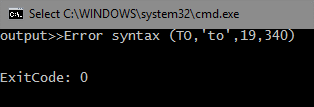
Рассмотрим тот же код программы с добавленными в него ошибками. При обнаружении их происходит вывод уведомления об ошибке:



Результаты работы программы:







**Рис. 4.9. Результаты работы программы c ошибками по порядку**

**5. Вывод**

В результате работы были получены знания о синтаксических деревьях, способах их построения, их предназначения в системе интерпретаторов. Для выделения синтаксических структур из кода был использован нисходящий парсер. В итоге работы был простроен простой синтаксический анализатор на основе уже имеющегося лексического, который способен не только строить и выводить синтаксическое дерево, но и выделять ошибки, уведомлять о них, выводя их на консоль.