Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

Кафедра «Вычислительная техника и системы управления»

Курсовой проект

«Компилятор подмножества языка Pascal»  
  
«Системное программное обеспечение»

Направление подготовки

09.03.01 – Информатика и вычислительная техника

Вариант №4

Выполнил: ст.гр. ВТ-122

Киселев А.В.

Руководитель:

Трофимов М.А.

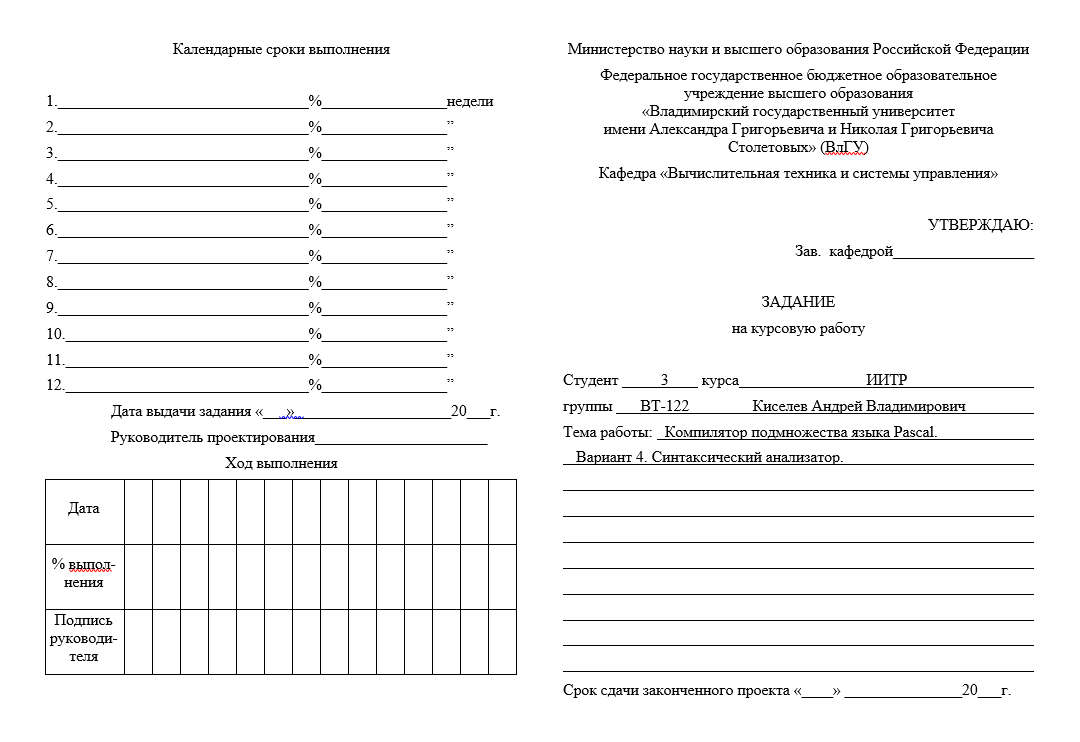
Владимир 2024

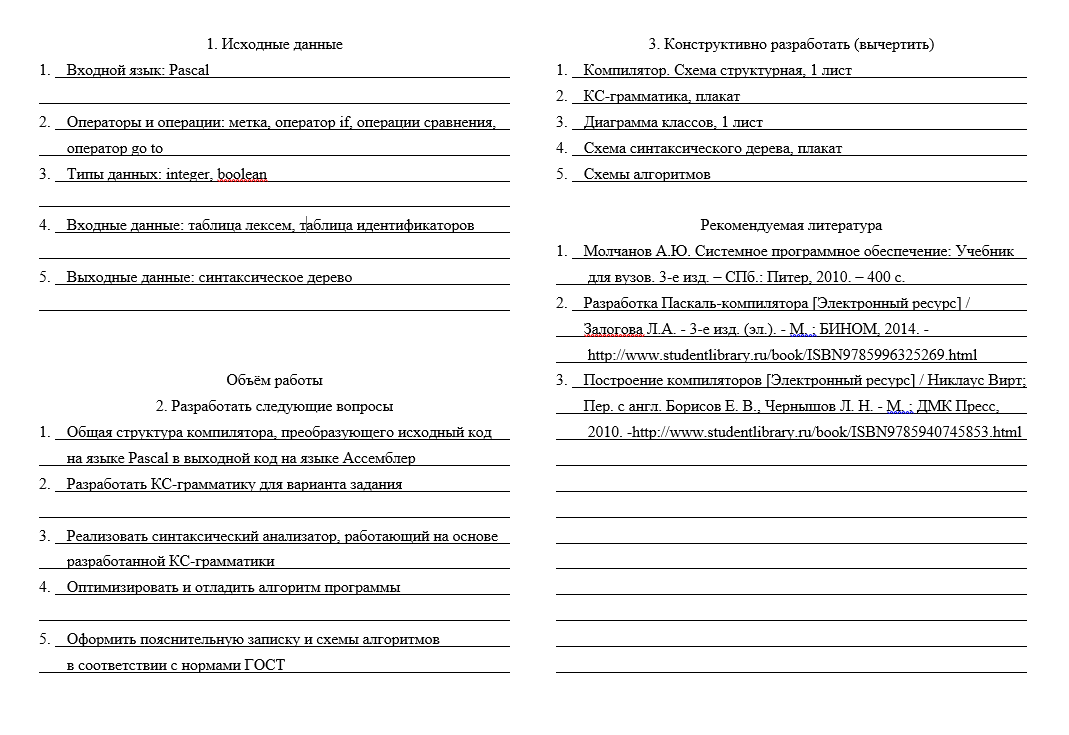
Целью курсового проекта является разработать компилятор подмножества языка Паскаль. Разработать синтаксический анализатор для входного кода программы на языке Pascal.

Документация к курсовому проекту представлена в виде пояснительной записки и программой, написанной на языке С++. Пояснительная записка состоит из ... страниц, 3 рисунков, 3 использованных источников.

The goal of the course project is to develop a compiler for a subset of the Pascal language. Develop a syntactic analyzer for the input code of a program in the Pascal language.

The documentation for the course project is presented in the form of an explanatory note and a program written in C++. The explanatory note consists of ... pages, 3 figures, 3 references.





Содержание

Введение………………………………………………………………………......3

1 Задание……..………………………………………………………………........5

1.1 Цель задания…………………………………………………………………..7

1.2 Требования к заданию………………………………………………….........12

2 Ход работы……………………………………………………………………..13

2.1 Определение синтаксического анализатора.........………………………....24

2.2 Грамматика БНФ.....................................................………………………....26

2.3 Диаграмма классов...................................................………………………...27

2.4 Примеры бинарных деревьев..................................………………………...27

2.5 Тестирование программы........................................………………………...28

2.6 Код программы........................................………………………....................30

Заключение………………………………………………………………………60

ВВЕДЕНИЕ

Компиляция в программировании — это процесс преобразования исходного кода программы, написанного на языке программирования, в машинный код или другой язык, понятный компьютеру. Этот процесс выполняется с помощью программы, называемой [компилятором](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80).

Основные этапы компиляции:

**Лексический анализ (сканирование):** Исходный код разбивается на токены, которые представляют собой минимальные лексические единицы (например, ключевые слова, идентификаторы, операторы).

**Синтаксический анализ:**  Токены анализируются с использованием грамматики языка программирования для построения синтаксического дерева, представляющего структуру программы.

**Семантический анализ:** Проверяется соответствие смысла программы правилам языка программирования, включая типы данных и контекст использования.

**Генерация промежуточного кода (опционально):** На этом этапе может быть сгенерирован код на языке ассемблер для запуска программы.

**Оптимизация (опционально):** Промежуточный код или синтаксическое дерево могут быть оптимизированы для улучшения производительности программы.

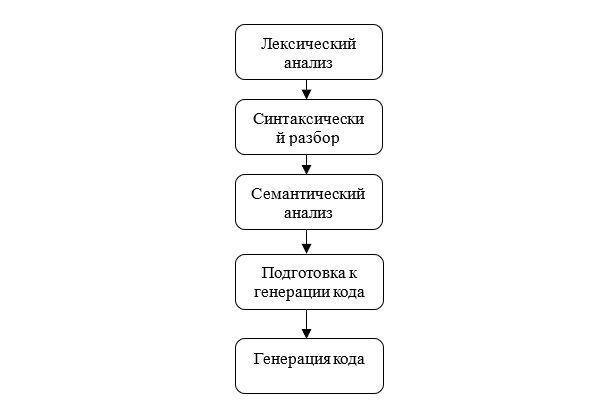


Рисунок 1 – Этапы компиляции

1 Задание

1.1 Цель задания

Целью данной работы является разработка синтаксического анализатора для компилятора подмножества языка Pascal.

1.2 Требования к заданию

Требуется выполнить синтаксический анализ входной программы на языке программирования Pascal. Cинтаксический анализатор по заданию должен уметь обрабатывать арифметические операции; унарные операции; условный оператор if; операции сравнения; метки; оператор go to; типы данных: integer, boolean.

2 Синтаксический анализатор

2.1 Определение синтаксического анализатора

Синтаксический анализатор - это часть компилятора, которая отвечает за выявление основных синтаксических конструкций входного языка. В задачу синтаксического анализа входит: найти и выделить основные синтаксические конструкции в тексте входной программы, установить тип и про­верить правильность каждой синтаксической конструкции, наконец, представить синтаксические конструкции в виде, удобном для дальнейшей генерации текста результирующей программы (в данном случае, в виде бинарного дерева).

2.2 Регулярная грамматика в форме Бекуса-Наура

Регулярная грамматика БНФ для языка Pascal:

- <prog> ::= program <prog-name> <dec-list><begin-end>.

- <prog-name> ::= id

- <dec-list> ::= <dec>; | dec; { dec;}

- <label-dec> ::= label <id-list>

- <dec> ::= var <id-list> : <type> | <label-dec>

- <id-list> ::= id | id{, id}

- <type> ::= integer | boolean

- <stmt-list> ::= <stmt> | <stmt> { ; stmt }

- <stmt> ::= <assign> | <if> | <goto>

- <assign> ::= id := <exp>

- <exp> ::= <term> | <exp> + <term> | <exp> - <term>

- <term> ::= <factor> | <term> \* <factor> | <term> DIV <factor>

- <factor> ::= id | int | <exp>

- <begin-end> ::= begin <stmt-list> end

- <if> ::= if <condition> then <stmt>

| if <condition> then <stmt> else <stmt>

| if <condition> then <begin-end>

| if <condition> then <begin-end> else <begin-end>

| if <condition> then <stmt> else <begin-end>

- <condition> ::= id <rel-op> id | int <rel-op> int | id <rel-op> int | id <rel-op> int

- <rel-op> ::= = | <> | < | <= | > | >=

- <label-dec> ::= label <id-list>

- <goto> ::= goto id

2.3 Диаграмма классов

Диаграмма классов представлена на рисунке 2.



Рисунок 2 – Диаграмма классов

2.4 Схема синтаксического дерева

Схема синтаксического дерева представлена на рисунках 3-7.

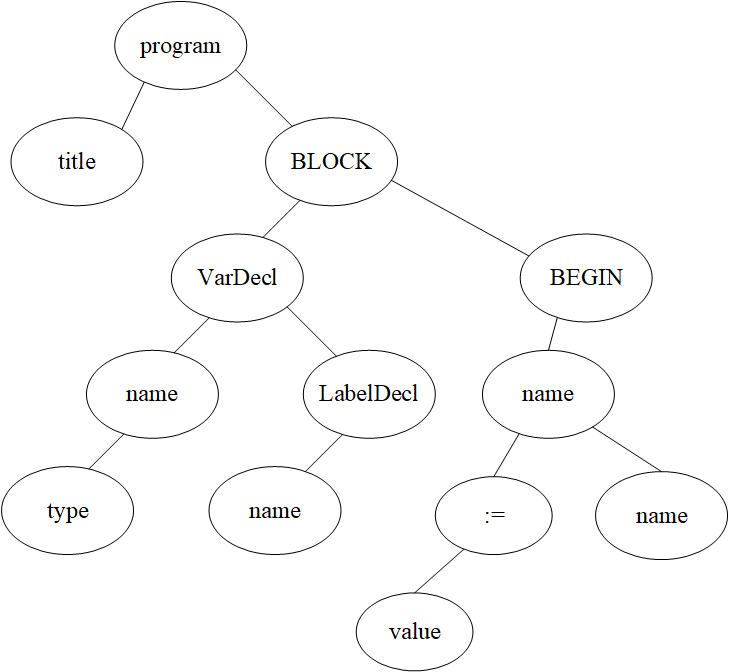


Рисунок 3 – Общий вид синтаксического дерева

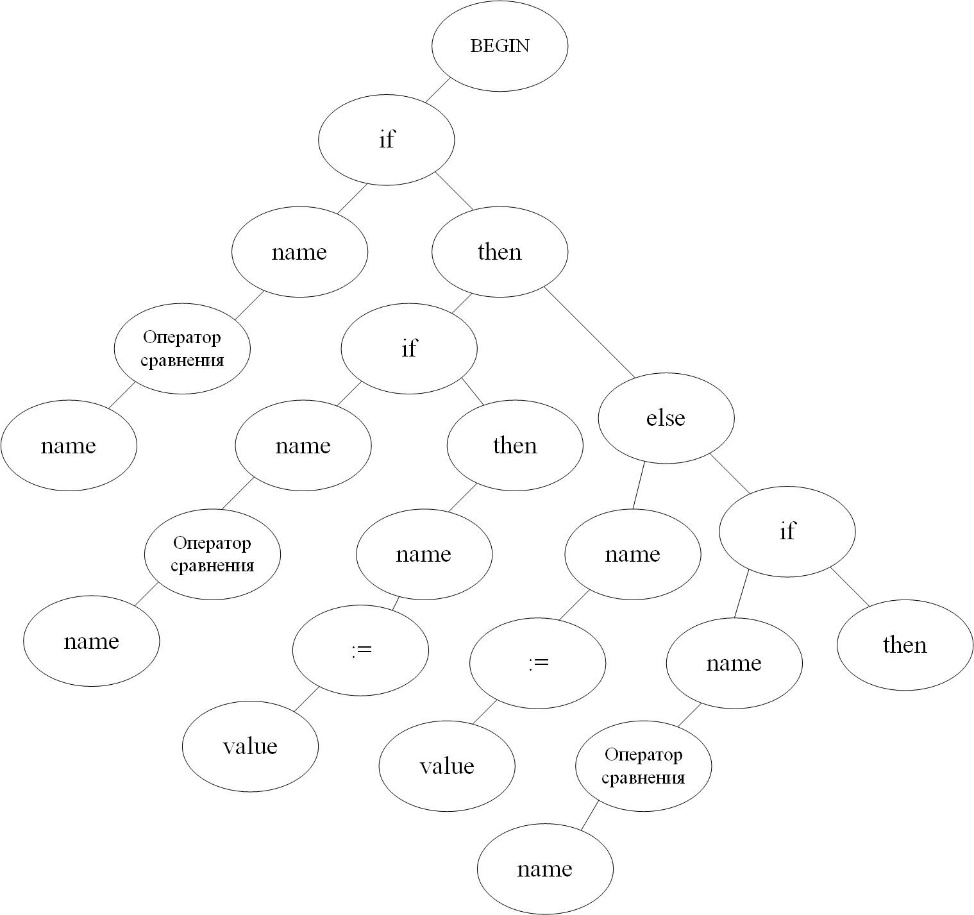


Рисунок 4 – Пример дерева условного оператора

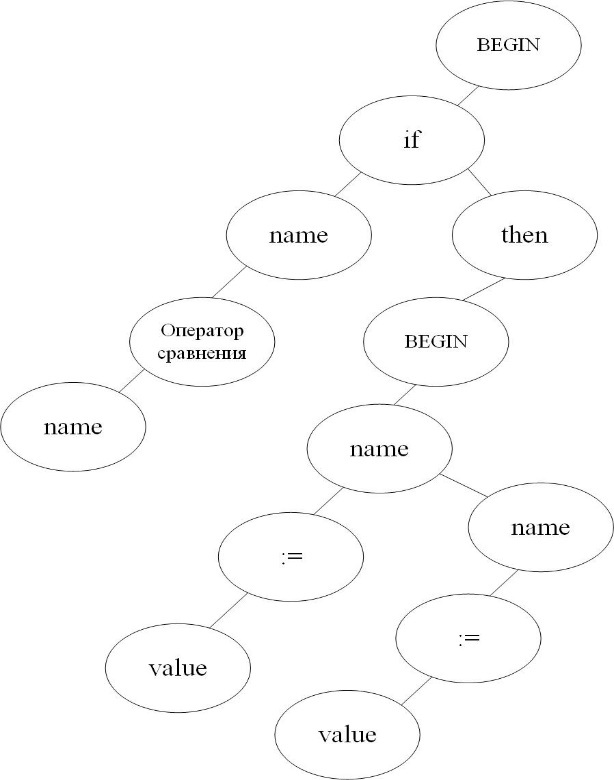


Рисунок 4 – Пример дерева условного оператора

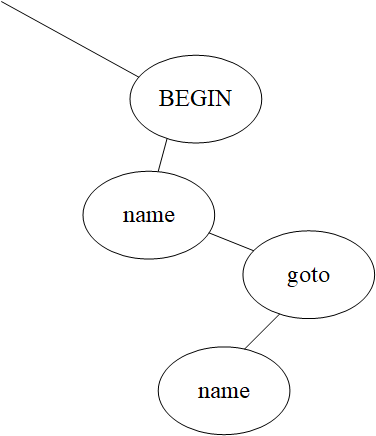


Рисунок 5 – Часть дерева с оператором безусловного перехода

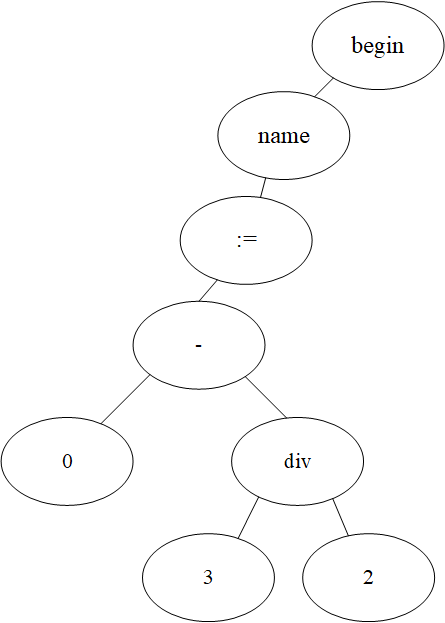


Рисунок 6 – Пример дерева арифметического выражения

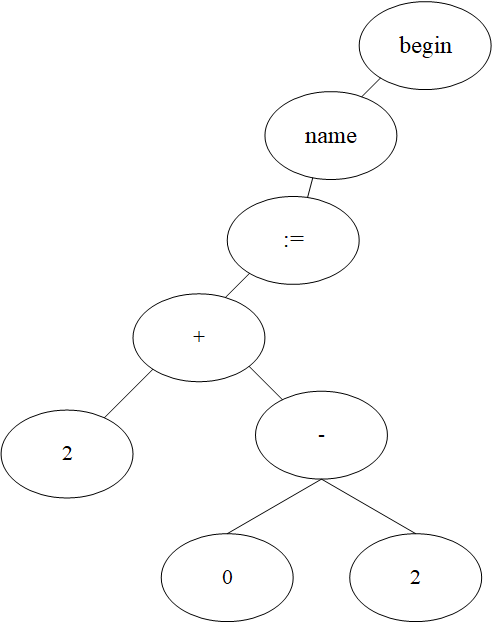


Рисунок 7 – Пример дерева арифметического выражения

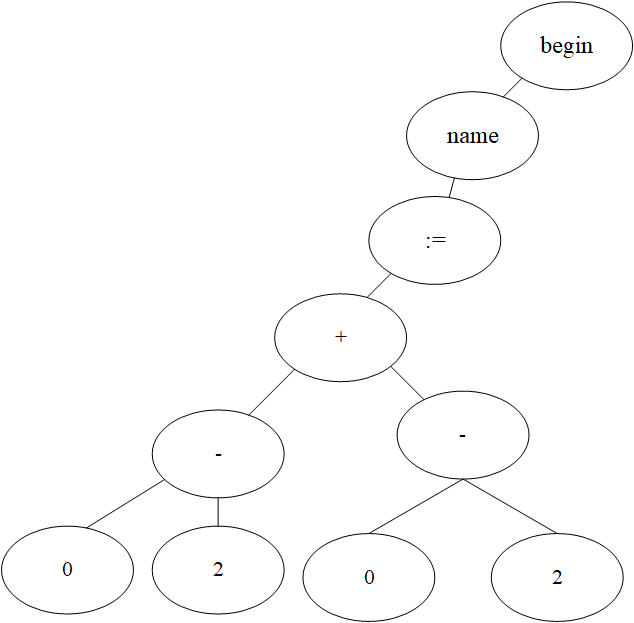


Рисунок 8 – Пример дерева арифметического выражения

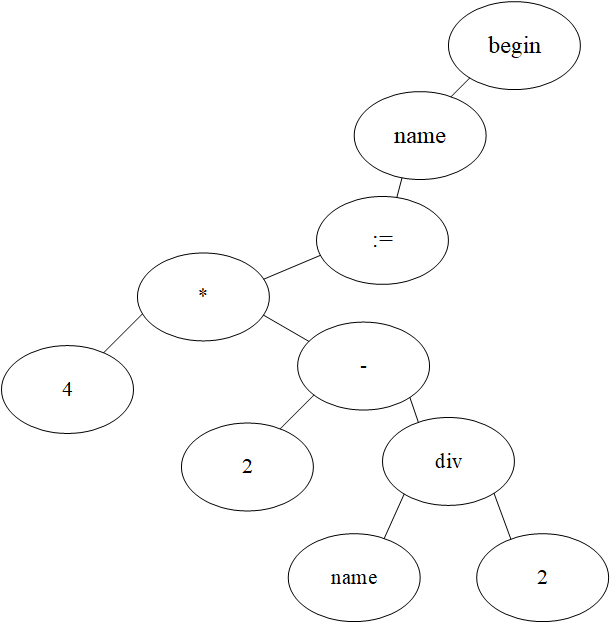


Рисунок 9 – Пример дерева арифметического выражения

2.5 Тестирование программы

Результаты тестирования программы представлены на рисунках 8-25.

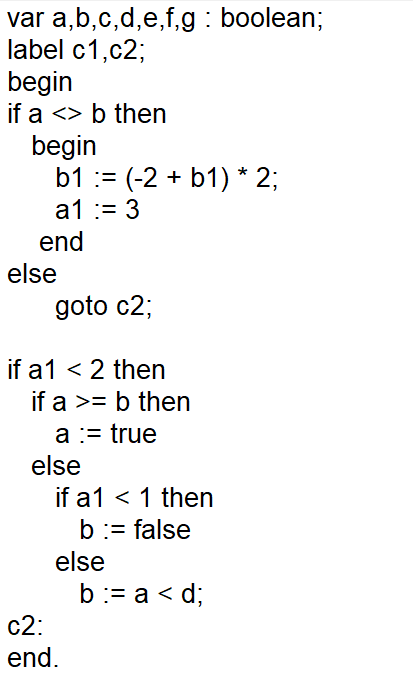


Рисунок 8 – Входной код программы



Рисунок 9 – Бинарное синтаксическое дерево программы



Рисунок 10 – Бинарное синтаксическое дерево программы

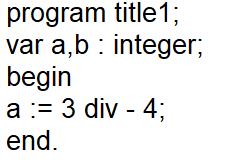


Рисунок 10 – Входной код программы

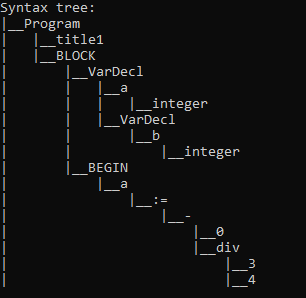


Рисунок 11 – Тестирование программы

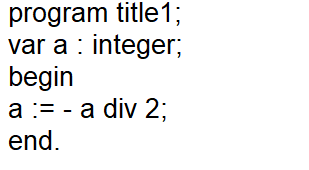


Рисунок 12 – Входной код программы

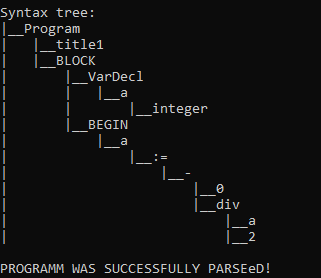


Рисунок 13 – Тестирование программы

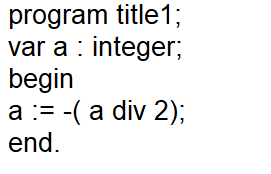


Рисунок 14 – Входной код программы

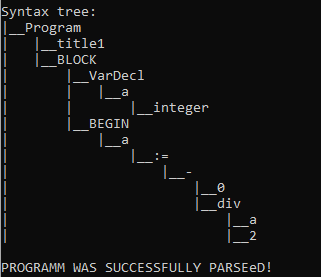


Рисунок 15 – Тест программы



Рисунок 16 – Ошибка в коде



Рисунок 17 – Обработка ошибки

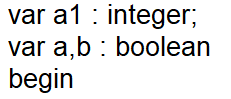


Рисунок 18 – Код с ошибкой



Рисунок 21 – Обработка ошибки

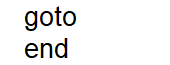


Рисунок 22 – Входной код с ошибкой



Рисунок 23 – Обработка ошибки

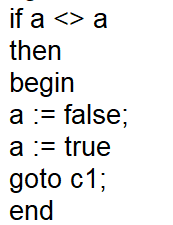


Рисунок 24 – Код с ошибкой



Рисунок 25 – Обработка ошибки

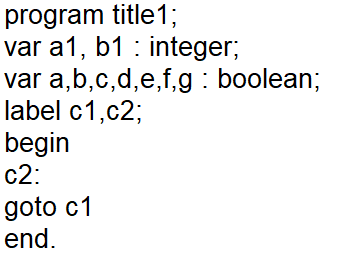


Рисунок 26 – Код с ошибкой



Рисунок 27 – Вывод ошибки

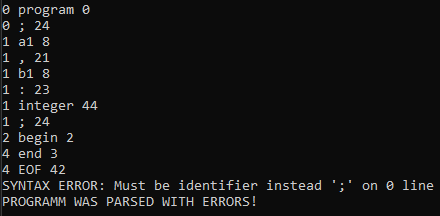


Рисунок 28 – Не указан title программы

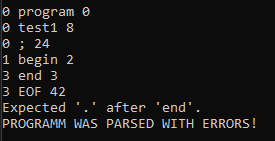


Рисунок 29 – Отсутствие точки после end

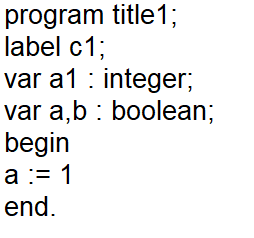


Рисунок 30 – Ошибочное преобразование типов



Рисунок 31 – Вывод ошибки

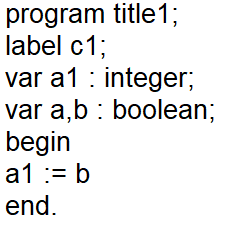


Рисунок 32 - Ошибочное преобразование типов



Рисунок 33 – Вывод ошибки

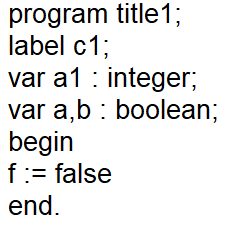


Рисунок 34 – Неизвестный идентификатор



Рисунок 35 – Вывод ошибки

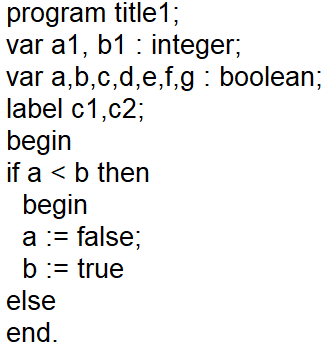


Рисунок 36 - Отсутствие end



Рисунок 37 – Вывод ошибки

2.6 Код программы

Код программы представлен на рисунках 37-62.

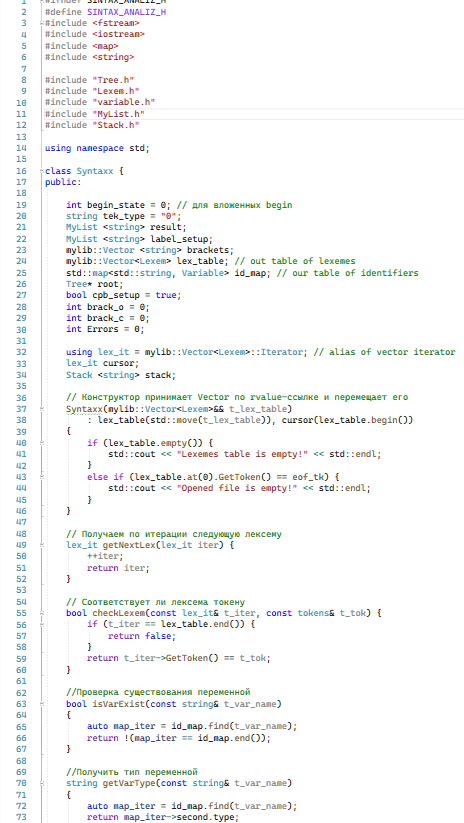


Рисунок 26 - Заголовочный файл синтаксического анализатора



Рисунок 27 - Заголовочный файл синтаксического анализатора

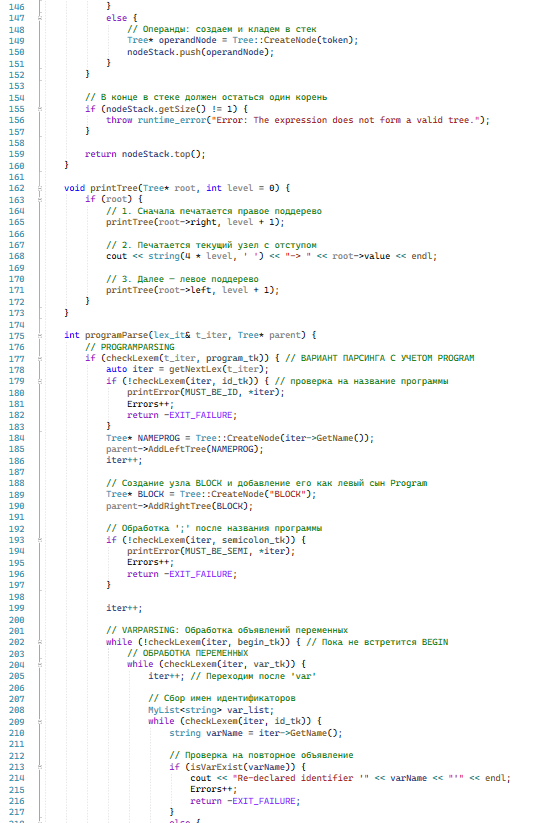


Рисунок 28 - Заголовочный файл синтаксического анализатора

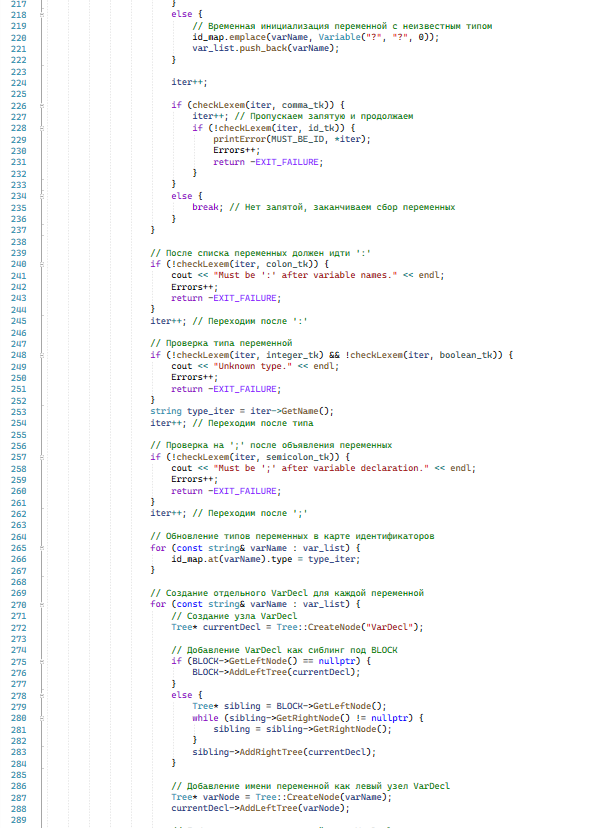


Рисунок 29 - Заголовочный файл синтаксического анализатора

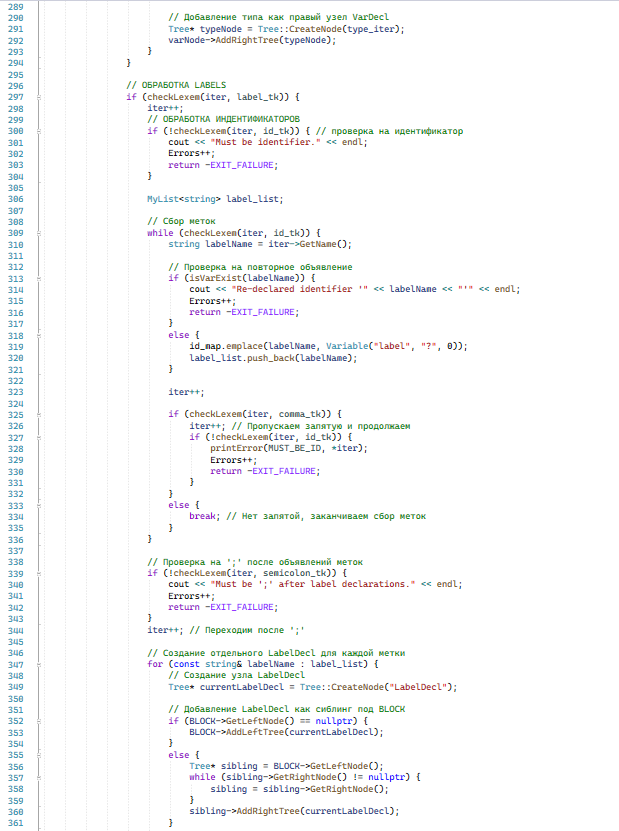


Рисунок 30 - Заголовочный файл синтаксического анализатора

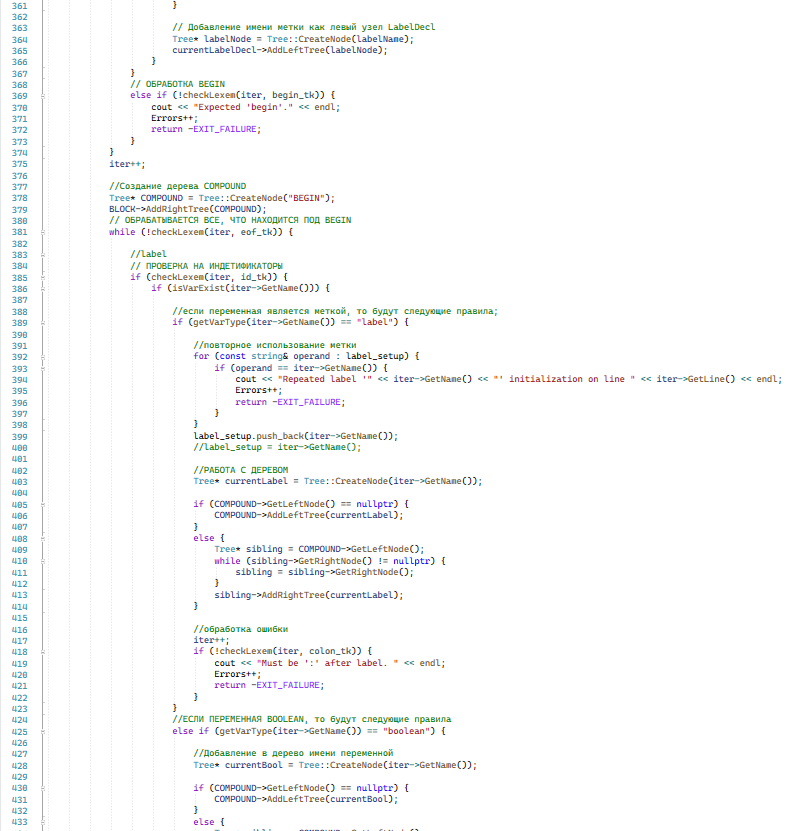


Рисунок 31 - Заголовочный файл синтаксического анализатора



Рисунок 32 - Заголовочный файл синтаксического анализатора

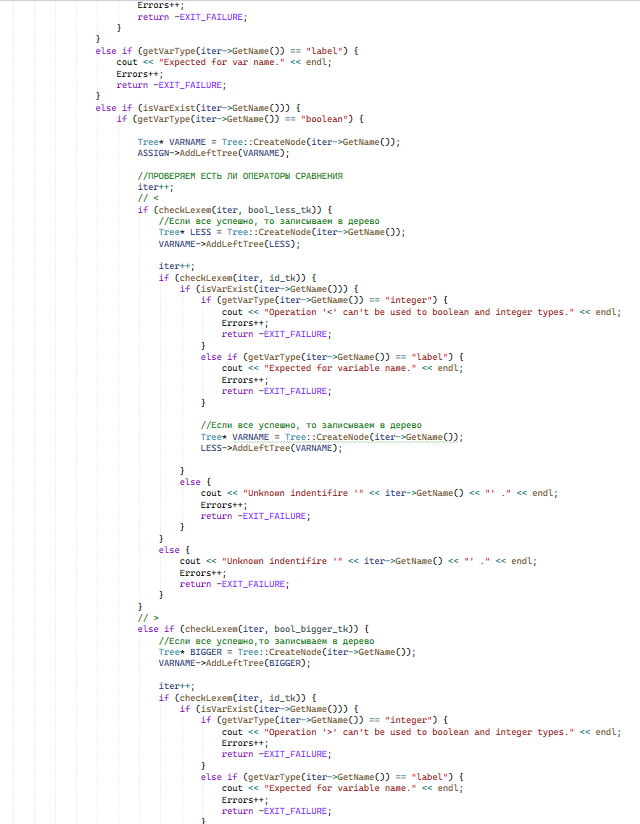


Рисунок 33 - Заголовочный файл синтаксического анализатора



Рисунок 34 - Заголовочный файл синтаксического анализатора



Рисунок 35 - Заголовочный файл синтаксического анализатора



Рисунок 36 - Заголовочный файл синтаксического анализатора



Рисунок 37 - Заголовочный файл синтаксического анализатора

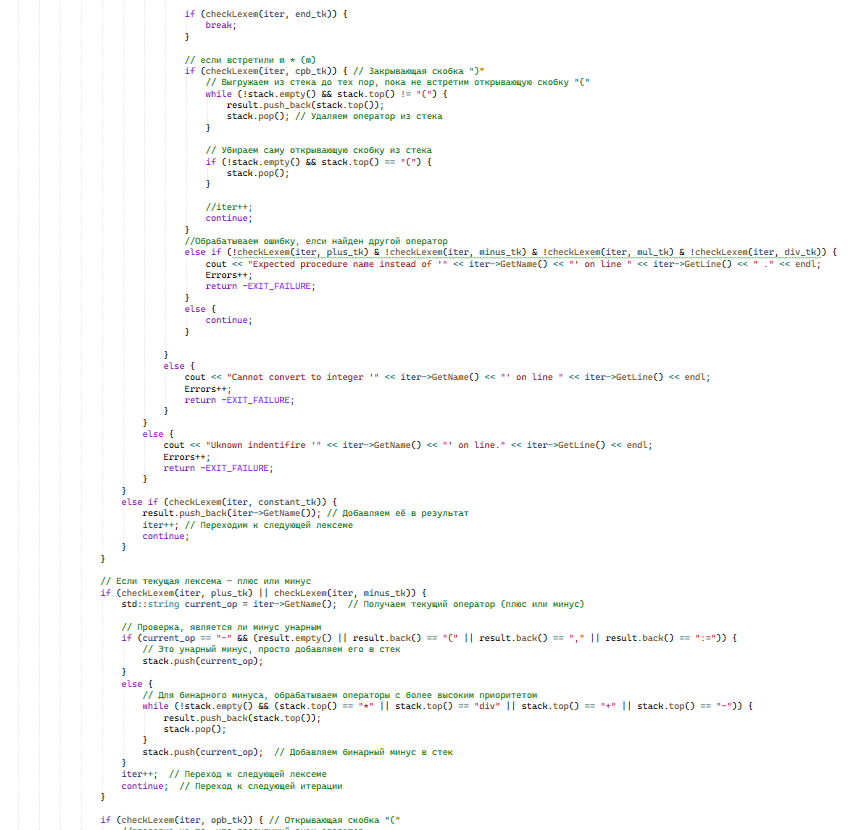


Рисунок 38 - Заголовочный файл синтаксического анализатора



Рисунок 39 - Заголовочный файл синтаксического анализатора

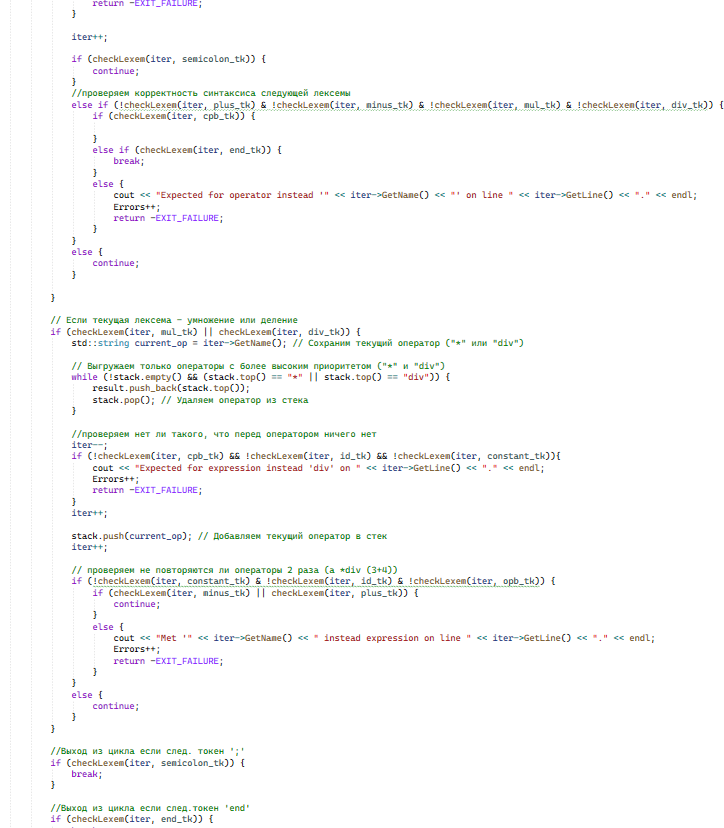


Рисунок 40 - Заголовочный файл синтаксического анализатора

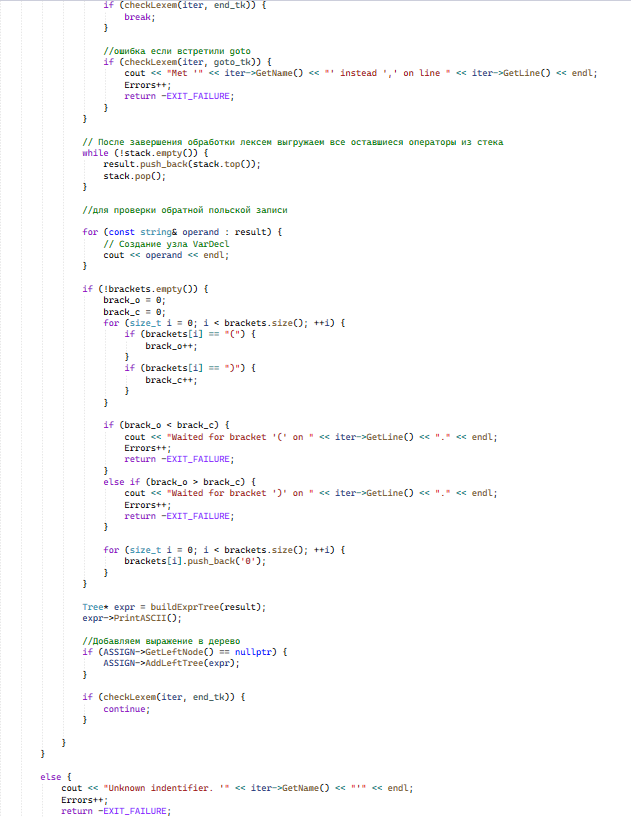


Рисунок 41 - Заголовочный файл синтаксического анализатора



Рисунок 42 - Заголовочный файл синтаксического анализатора



Рисунок 43 - Заголовочный файл синтаксического анализатора



Рисунок 44 - Заголовочный файл синтаксического анализатора



Рисунок 45 - Заголовочный файл синтаксического анализатора

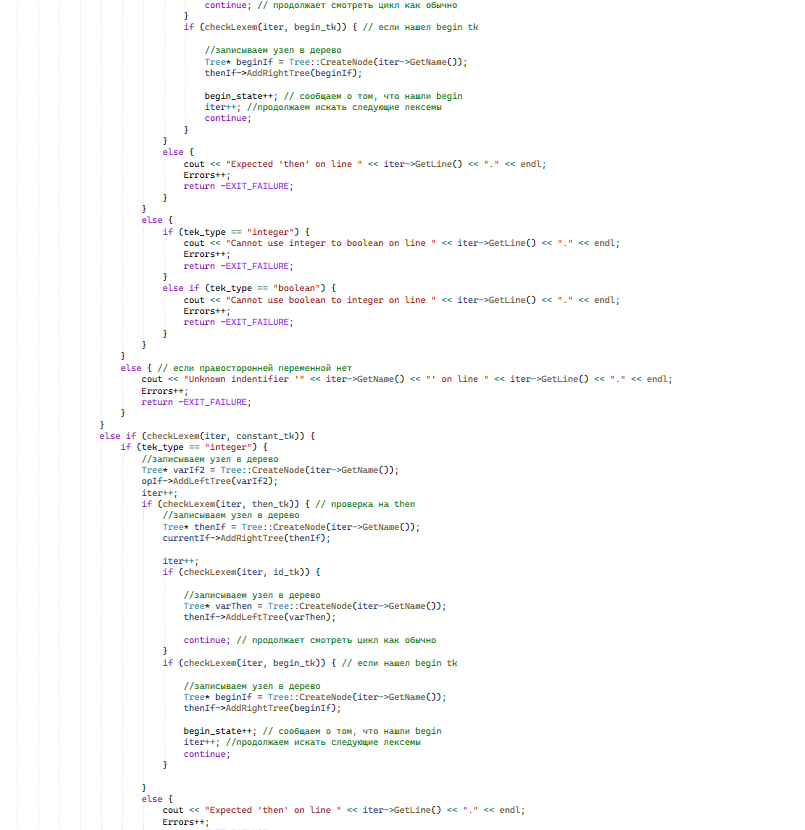


Рисунок 46 - Заголовочный файл синтаксического анализатора



Рисунок 47 - Заголовочный файл синтаксического анализатора



Рисунок 48 - Заголовочный файл синтаксического анализатора

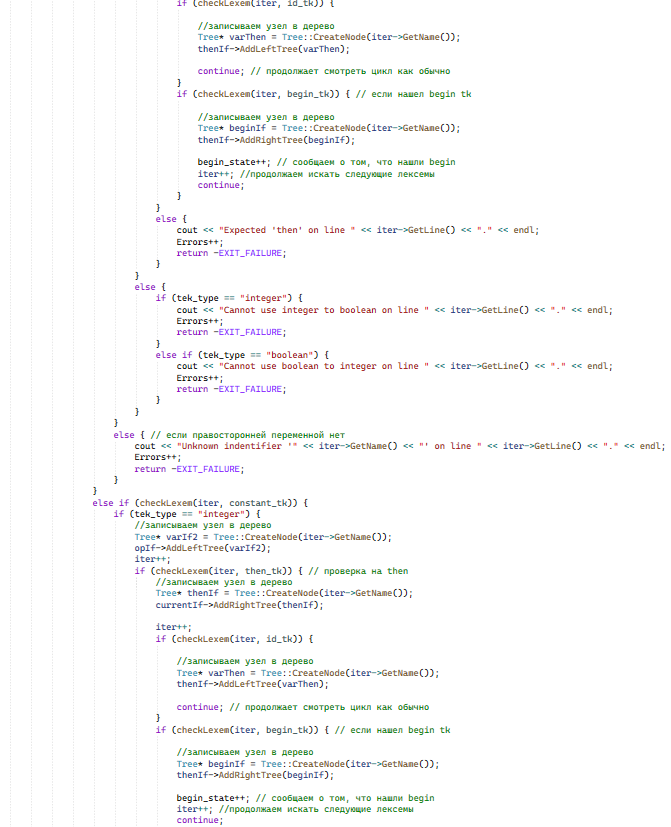
 Рисунок 50 - Заголовочный файл синтаксического анализатора

 Рисунок 51 - Заголовочный файл синтаксического анализатора



Рисунок 52 - Заголовочный файл синтаксического анализатора



Рисунок 53 - Заголовочный файл синтаксического анализатора

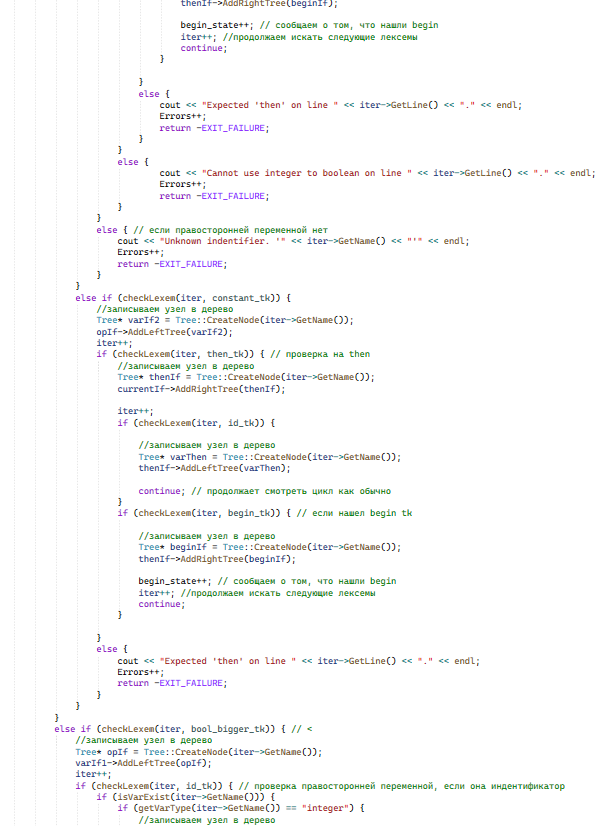


Рисунок 54 - Заголовочный файл синтаксического анализатора

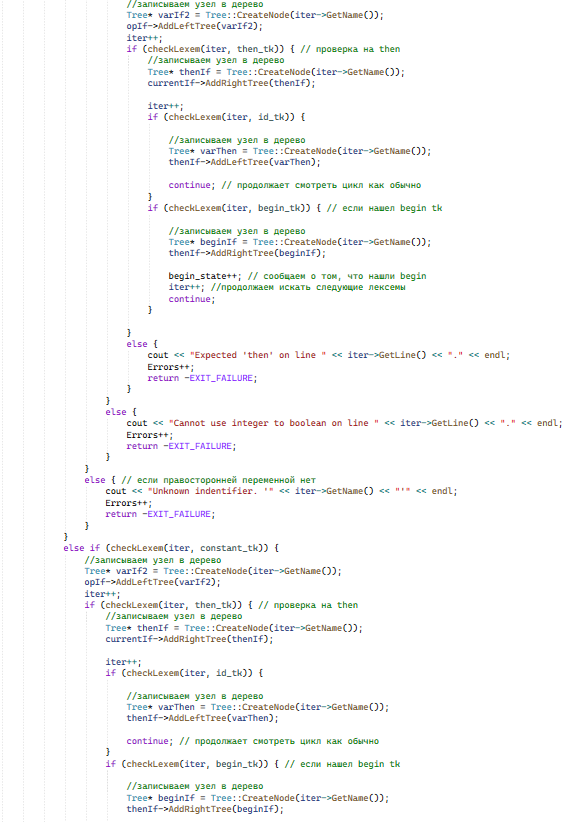


Рисунок 55 - Заголовочный файл синтаксического анализатора



Рисунок 56 - Заголовочный файл синтаксического анализатора



Рисунок 57 - Заголовочный файл синтаксического анализатора

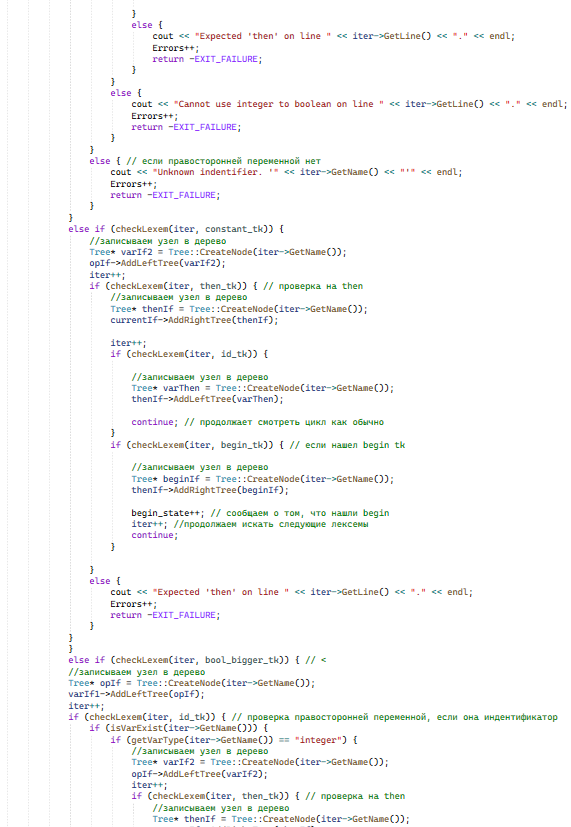


Рисунок 58 - Заголовочный файл синтаксического анализатора

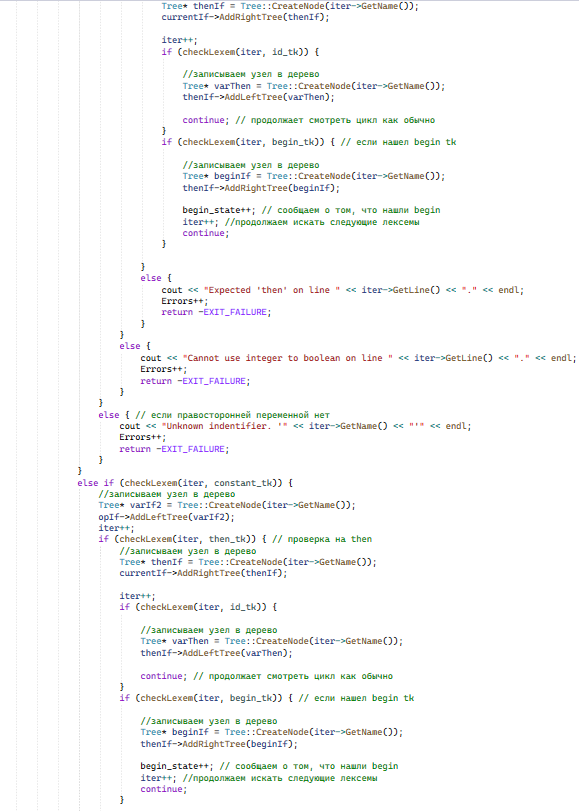


Рисунок 59 - Заголовочный файл синтаксического анализатора



Рисунок 60 - Заголовочный файл синтаксического анализатора

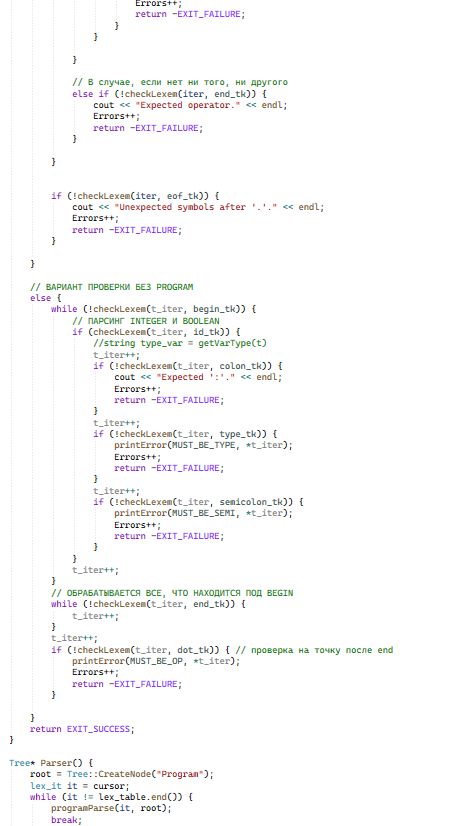


Рисунок 61 - Заголовочный файл синтаксического анализатора



Рисунок 62 - Заголовочный файл синтаксического анализатора

Код файла Stack.h для реализации стека представлен на рисунке 63.

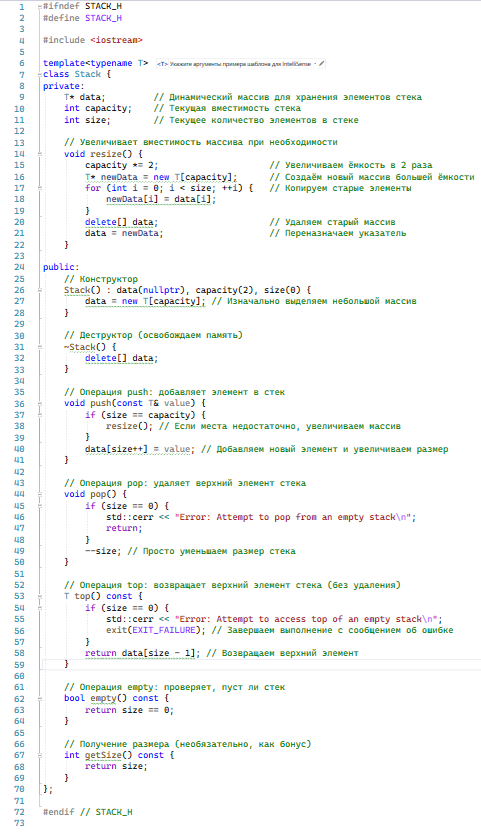


Рисунок 63 – Заголовочный файл для реализации стека

На рисунке 64 представлен код файла Tree.h для реализации дерева.

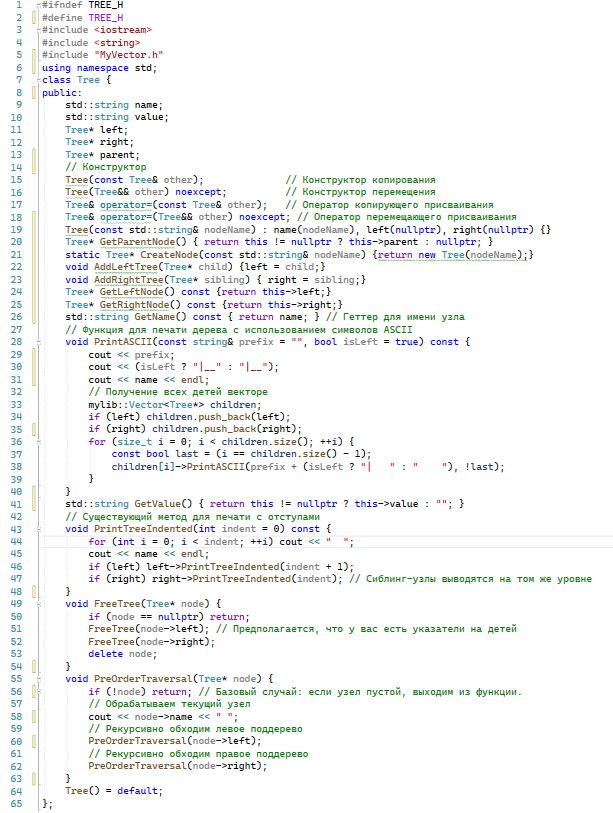


Рисунок 64– Заголовочный файл для реализации дерева

На рисунках 65-66 представлен код заголовочного файла Vector.h для реализации вектора.

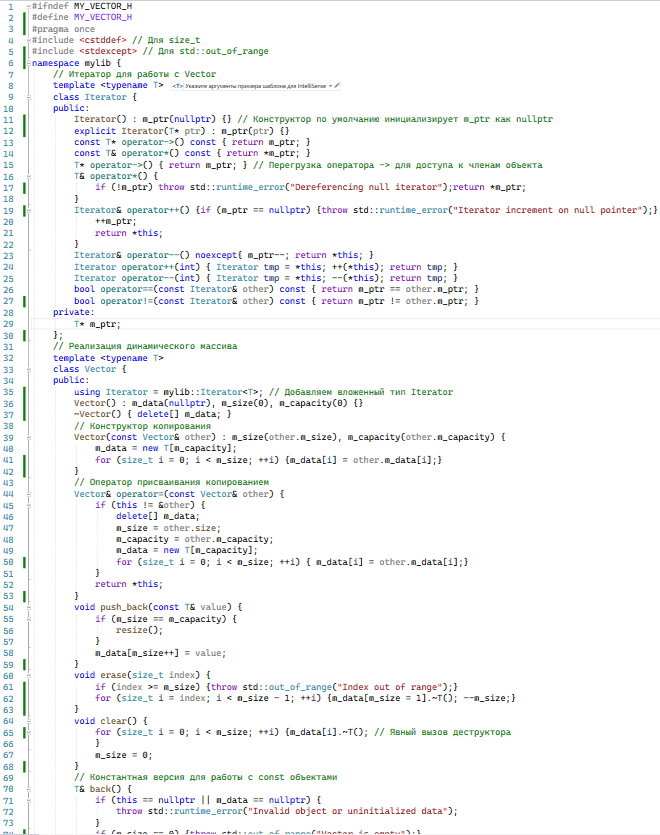


Рисунок 65 – Заголовочный файл для вектора

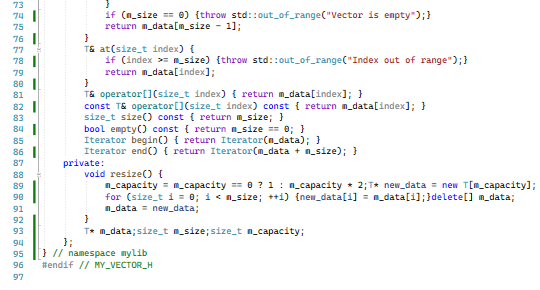


Рисунок 66 – Заголовочный файл для вектора

На рисунке 67 представлен код заголовочного файла MyList.h для реализации списка.

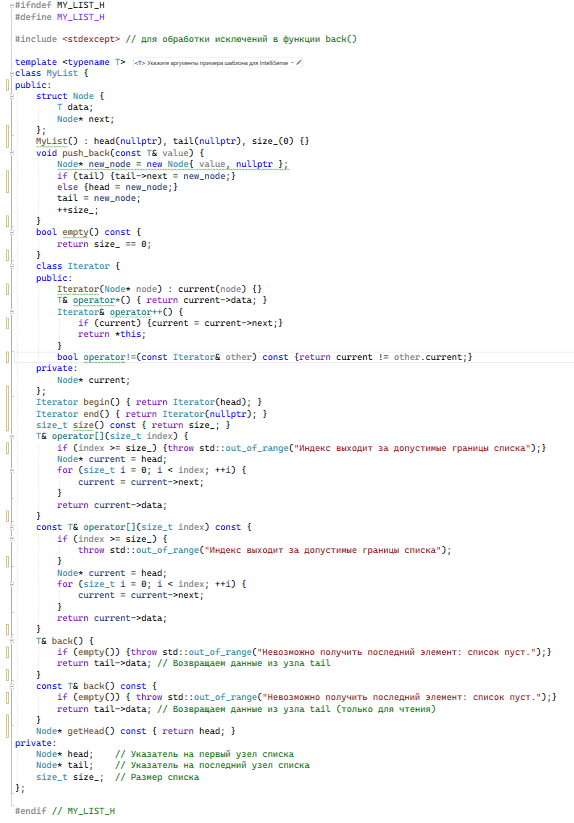


Рисунок 67 – Заголовочный файл для вектора

Код заголовочного файла CourseProject.h для определения функции запуска компилятора представлен на рисунке 68.

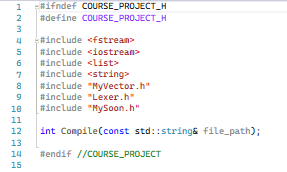


Рисунок 68 - Заголовочный файл

Код файла CourseProject.cpp для функции запуска компилятора представлен на рисунке 69.

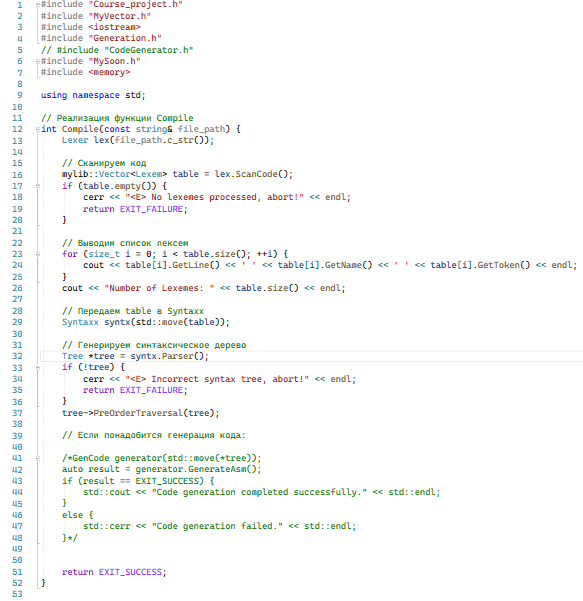


Рисунок 69 – Код файла для запуска компиляции

Код заголовочного файла Variable.h для определения типа переменной для таблицы идентификаторов на рисунке 70.

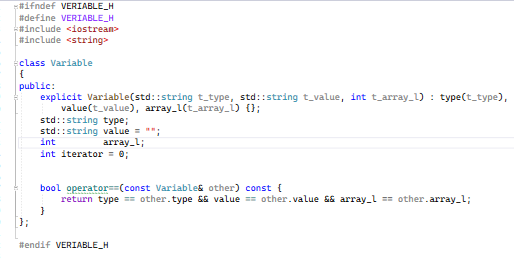


Рисунок 70 – Заголовочный файл для идентификации переменной.

Код внутри главной функции main представлен на рисунке 71.

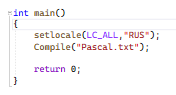


Рисунок 71 – Код внутри главной функции

Заключение

В ходе выполнения данной курсовой работы был выполнен синтаксический анализ входного кода подмножества языка Pascal. По входным данным из таблицы лексем, таблицы идентификаторов, на выходе было получено бинарное синтаксическое дерево программы, обходить которое следует прямым методом обхода бинарного дерева.