Язык – C++.

Содержание

[Модуль ввода-вывода 3](#_Toc101173314)

[Проектирование 3](#_Toc101173315)

[Разработка 3](#_Toc101173316)

[Тестирование 4](#_Toc101173317)

[Лексический анализатор 5](#_Toc101173318)

[Проектирование 5](#_Toc101173319)

[Разработка 6](#_Toc101173320)

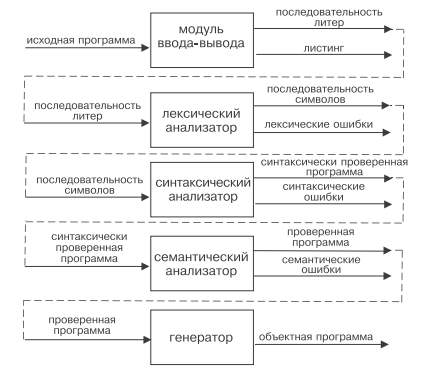
[Тестирование 7](#_Toc101173321)

[Синтаксический анализатор 8](#_Toc101173322)

[Проектирование 8](#_Toc101173323)

[Разработка 8](#_Toc101173324)

[Тестирование 9](#_Toc101173325)

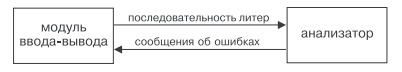


# Модуль ввода-вывода

## Проектирование

Литера – это

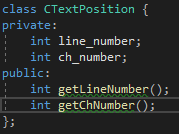
Взаимодействие между модулем ввода-вывода и анализатором можно представить в виде схемы:

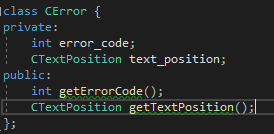


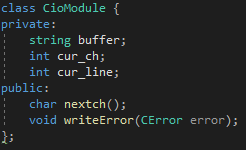
Требования к модулю ввода-вывода:

* Считывает последовательность литер исходной программы и передает ее анализатору
* Формирует листинг

## Разработка







## Тестирование

|  |  |  |  |
| --- | --- | --- | --- |
| Название теста |  | Ожидаемый результат | Фактический результат |
| Чтение символов |  | Модуль считывает все литеры программы |  |
| Вывод ошибки |  |  |  |
|  |  |  |  |

# Лексический анализатор

## Проектирование

Лексический анализатор – это модуль, который формирует символы исходной программы и строит их внутреннее представление.

Лексема – это синтаксическая единица, последовательность допустимых символов языка программирования, имеющая смысл для транслятора.

Токен – это каждая из значимых частей (в виде последовательности символов между разделителями), на которые разбивается текст специальной компьютерной программой.

Требования к лексическому анализатору:

* Формирует символы исходной программы
* Должен передавать другим блокам компилятора следующую информацию: код символа, позицию символа, значение константы, адрес идентификатора в таблице имен.

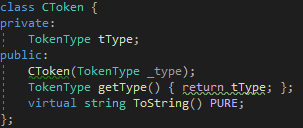
Все токены можно поделить на три группы:

1. ключевое слово,
2. идентификатор,
3. константа.

Таблица ключевых слов

|  |  |
| --- | --- |
| Ключевое слово | Условное обозначение |
| var | varSy, |
| begin | beginSy, |
| end | endSy, |
| and | andSy, |
| or | orSy, |
| not | notSy, |
| < | lessSy, |
| > | moreSy, |
| <= | lessEvenSy, |
| >= | moreEvenSy, |
| <> | notEvenSy, |
| := | assignSy, |
| + | plusSy, |
| - | minusSy, |
| \* | multSy, |
| / | subSy, |
| ( | leftBrSy, |
| ) | rightBrSy, |
| if | ifSy, |
| then | thenSy |
| else | elseSy, |
| while | whileSy, |
| do | doSy, |
| “ | quotSy, |
| : | colonSy, |
| ; | semiColonSy, |
| . | dotSy |
| , | commaSy |
| program | programSy |

## Разработка





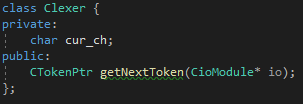












## Тестирование

|  |  |  |  |
| --- | --- | --- | --- |
| Название теста |  | Ожидаемый результат | Фактический результат |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Синтаксический анализатор

## Проектирование

Синтаксический анализатор – это модуль, который формирует символы исходной программы и строит их внутреннее представление.

Требования к синтаксическому анализатору:

* Должен проверять последовательность токенов на синтаксические ошибки

Правила, описывающие синтаксис языка Pascal:

<программа>::=program<имя>;<блок>.

<блок>::=<раздел типов><раздел переменных><раздел операторов>

<раздел переменных> ::=var <описание однотипных переменных> ;{<описание однотипных переменных> ; }| < пусто>

<описание однотипных переменных> ::=<имя> { , <имя>} : <тип>

<раздел операторов>::=<составной оператор>

<цикл с предусловием>::=while<выражение>do<оператор>

<выражение>::=

<оператор>::=

<составной оператор>::=begin<оператор>{;<оператор>}end

<переменная>::=<имя>{[<выражение>{,<выражение>}]|.<имя>|^}

<условный оператор> ::= if <выражение> then <оператор><хвост>

<хвост> ::= else <оператор>| <пусто>

## Разработка

а

## Тестирование

|  |  |  |  |
| --- | --- | --- | --- |
| Название теста |  | Ожидаемый результат | Фактический результат |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |