МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Специализация Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Разработка компилятора SAE-2019»

Выполнил студент Шиманель Альберт Эдвардович

(Ф.И.О.)

Руководитель проекта ст.пр. Наркевич Аделина Сергеевна

(учен. степень, звание, должность, подпись, Ф.И.О.)

Заведующий кафедрой к.т.н., доц. Пацей Н.В.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Консультант ст.пр. Наркевич Аделина Сергеевна

(учен. степень, звание, должность, подпись, Ф.И.О.)

Нормоконтролер ст.пр. Наркевич Аделина Сергеевна

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Минск 2019

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 6](#_Toc27413843)

[1. СПЕЦИФИКАЦИЯ ЯЗЫКА ПРОГРАММИРОВАНИЯ 7](#_Toc27413844)

[1.1. Характеристика языка программирования 7](#_Toc27413845)

[1.2. Алфавит языка 7](#_Toc27413846)

[1.3. Применяемые сепараторы 7](#_Toc27413847)

[1.4. Применяемые кодировки 8](#_Toc27413848)

[1.5. Типы данных 8](#_Toc27413849)

[1.6. Преобразование типов данных 9](#_Toc27413850)

[1.7. Идентификаторы 9](#_Toc27413851)

[1.8. Литералы 9](#_Toc27413852)

[1.9. Объявление данных 10](#_Toc27413853)

[1.10. Инициализация данных 10](#_Toc27413854)

[1.11. Инструкции языка 10](#_Toc27413855)

[1.12. Операции языка 11](#_Toc27413856)

[1.13. Выражения и их вычисления 11](#_Toc27413857)

[1.14. Конструкции языка 12](#_Toc27413858)

[1.15. Область видимости идентификаторов 12](#_Toc27413859)

[1.16. Семантические проверки 12](#_Toc27413860)

[1.17. Распределение оперативной памяти на этапе выполнения 12](#_Toc27413861)

[1.18. Стандартная библиотека и её вызов 13](#_Toc27413862)

[1.19. Ввод и вывод данных 13](#_Toc27413863)

[1.20. Точка входа 13](#_Toc27413864)

[1.21. Препроцессор 13](#_Toc27413865)

[1.22. Соглашения о вызовах 13](#_Toc27413866)

[1.23. Объектный код 13](#_Toc27413867)

[1.24. Контрольный пример 14](#_Toc27413868)

[2. СТРУКТУРА ТРАНСЛЯТОРА 15](#_Toc27413869)

[2.1. Компоненты транслятора, их назначение и принципы взаимодействия 15](#_Toc27413870)

[2.2. Перечень входных параметров транслятора 16](#_Toc27413871)

[2.3. Перечень протоколов, формируемых транслятором и их содержимое 16](#_Toc27413872)

[3. РАЗРАБОТКА ЛЕКСИЧЕСКОГО АНАЛИЗАТОРА 18](#_Toc27413873)

[3.1. Структура лексического анализатора 18](#_Toc27413874)

[3.2. Контроль входных символов 18](#_Toc27413875)

[3.3. Удаление избыточных символов 19](#_Toc27413876)

[3.4. Перечень ключевых слов, сепараторов, символов операций соответствующих им лексем. 19](#_Toc27413877)

[3.5. Основные структуры данных 21](#_Toc27413878)

[3.6. Принцип обработки ошибок 21](#_Toc27413879)

[3.7. Структура и перечень сообщений лексического анализатора 21](#_Toc27413880)

[3.8. Параметры лексического анализатора и режим его работы 22](#_Toc27413881)

[3.9. Контрольный пример 22](#_Toc27413882)

[4. РАЗРАБОТКА СИНТАКСИЧЕСКОГО АНАЛИЗАТОРА 23](#_Toc27413883)

[4.1. Структура синтаксического анализатора. 23](#_Toc27413884)

[4.2. Контекстно-свободная грамматика, описывающая синтаксис языка 23](#_Toc27413885)

[4.3. Построение конченого магазинного автомата 24](#_Toc27413886)

[4.4. Основные структуры данных 25](#_Toc27413887)

[4.5. Описание алгоритма синтаксического разбора 25](#_Toc27413888)

[4.6. Структура и перечень сообщений синтаксического анализатора 26](#_Toc27413889)

[4.7. Параметры синтаксического анализатора и режимы его работы 26](#_Toc27413890)

[4.8. Принцип обработки ошибок 26](#_Toc27413891)

[4.9. Контрольный пример 26](#_Toc27413892)

[5. РАЗРАБОТКА СЕМАНТИЧЕСКОГО АНАЛИЗАТОРА 27](#_Toc27413893)

[5.1. Структура семантического анализатора 27](#_Toc27413894)

[5.2. Функции семантического анализатора 27](#_Toc27413895)

[5.3. Структура и перечень сообщений семантического анализатора 28](#_Toc27413896)

[5.4. Принцип обработки ошибок 29](#_Toc27413897)

[5.5. Контрольный пример 29](#_Toc27413898)

[6. ПРЕОБРАЗОВАНИЕ ВЫРАЖЕНИЙ 30](#_Toc27413899)

[6.1. Выражения, допускаемые языком 30](#_Toc27413900)

[6.2. Польская запись и принцип ее построения 30](#_Toc27413901)

[6.3. Программная реализация обработки выражений 31](#_Toc27413902)

[6.4. Контрольный пример 31](#_Toc27413903)

[7. ГЕНЕРАЦИЯ КОДА 33](#_Toc27413904)

[7.1. Структура генератора кода 33](#_Toc27413905)

[7.2. Представление типов данных в памяти 33](#_Toc27413906)

[7.3. Статическая библиотека 34](#_Toc27413907)

[7.4. Особенности алгоритма генерации кода 35](#_Toc27413908)

[7.5. Контрольный пример 35](#_Toc27413909)

[8. ТЕСТИРОВАНИЕ ТРАНСЛЯТОРА 36](#_Toc27413910)

[ЗАКЛЮЧЕНИЕ 38](#_Toc27413911)

[ПРИЛОЖЕНИЕ А 39](#_Toc27413912)

[ПРИЛОЖЕНИЕ Б 40](#_Toc27413913)

[ПРИЛОЖЕНИЕ В 41](#_Toc27413914)

[ПРИЛОЖЕНИЕ Г 42](#_Toc27413915)

[ПРИЛОЖЕНИЕ Д 43](#_Toc27413916)

[ПРИЛОЖЕНИЕ Е 44](#_Toc27413917)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 47](#_Toc27413918)

# ВВЕДЕНИЕ

Целью курсового проекта является разработка транслятора для своего языка программирования: SAE-2019.

Транслятор — это комплекс отдельных программ, позволяющих преобразовывать исходный код на одном языке программирования в исходный код на другом языке программирования. Задача транслятора — сделать программу, написанную на некотором языке программирования, понятной компьютеру. Этого можно добиться одним из двух способов: компиляцией или интерпретацией. Язык SAE-2019 является компилируемым. Компилятор переводит исходную программу в эквивалентную ей на язык ассемблера.

Процесс создания компилятора можно свести к решению нескольких задач, которые принято называть фазами компиляции. Разрабатываемый компилятор состоит из следующих фаз:

* лексический анализ;
* синтаксический анализ;
* семантический анализ;
* генерация кода;

Исходя из цели курсового проекта, были определены следующие основные задачи:

* разработка спецификации языка программирования;
* разработка структуры транслятора;
* разработка лексического и семантического анализаторов;
* разработка синтаксического анализатора;
* преобразование выражений;
* генерация кода на язык ассемблера;
* тестирование транслятора.

# СПЕЦИФИКАЦИЯ ЯЗЫКА ПРОГРАММИРОВАНИЯ

## Характеристика языка программирования

Язык программирования SAE-2019 — это процедурный, строго типизированный, компилируемый язык. Не является объектно-ориентированным.

## Алфавит языка

Алфавит языка SAE-2019 основан на кодировке Windows-1251.

Исходный код SAE-2019 может содержать символы латинского алфавита малого и верхнего регистра, цифры десятичной системы счисления от 0 до 9, в строковых литералах разрешены символы латинского и русского алфавитов, а также знаки препинания. Алфавит языка SAE-2019 представлен в таблице 1.2.

Таблица 1.2.

Алфавит языка

|  |  |
| --- | --- |
| Название подгруппы | Символы подгруппы |
| Символы латинского алфавита | [a-Z] |
| Символы русского алфавита | [а-Я] |
| Цифры | [0-9] |
| Другие символы | [,” !?%+-/\*= ] |

## Применяемые сепараторы

Символы-сепараторы — символы, используемые для разделения отдельных лексических единиц или функциональных элементов в исходном коде программы. Символы–сепараторы представлены в таблице 1.3.

Таблица 1.3.

Сепараторы

|  |  |
| --- | --- |
| Сепаратор | Назначение |
| ; | Разделение инструкций |
| “ ”(пробел) | Разделение цепочек |

Окончание табл.

|  |  |
| --- | --- |
| Сепаратор | Назначение |
| , | Разделение параметров функций |
| { } | Программный блок |
| ( ) | Параметры и приоритетность операций (в выражениях) |
| [ ] | Блок условной конструкции |

## Применяемые кодировки

Для написания исходного кода на языке программирования SAE-2019 используется кодировка Windows-1251.

## Типы данных

В языке SAE-2019 есть 3 типа данных: целочисленный, булевый и строковый. Описание типов данных, предусмотренных в данном языке представлено в таблице 1.5.

Таблица 1.5.

Типы данных языка программирования SAE-2019

|  |  |
| --- | --- |
| Тип данных | Описание типа данных |
| byte | Фундаментальный тип данных. Предусмотрен для объявления целочисленных данных (1 байт). Диапазон от -128 до 127.  Автоматически инициализируется нулевым значением.  Возможные операции:  + – бинарный, суммирование;  - – бинарный, вычитание;  \* – бинарный, умножение;  / – бинарный, деление  %– бинарный, остаток от деления  = – присваивание значения;  ? – бинарный, сравнение; |

Окончание табл.

|  |  |
| --- | --- |
| Тип данных | Описание типа данных |
| bool | Фундаментальный тип данных. Предусмотрен для объявления булевых значений (true, false). |
| string | Фундаментальный тип данных. Предусмотрен для объявления строк. (1 символ – 1 байт). Автоматическая инициализация пустой строки. Максимальное количество символов в строке – 32.  Возможные операции:  = – присваивание значения; |

## Преобразование типов данных

В языке программирования SAE–2019 не поддерживается преобразование типов.

## Идентификаторы

В имени идентификатора допускаются символы латинского алфавита нижнего и верхнего регистра. Максимальная длина идентификатора равна 10. Если его длина будет превышать максимальное значение, то имя идентификатора будет урезаться.

## Литералы

Литерал — запись в исходном коде, представляющая собой фиксированное значение. В языке существует 3 типа литералов: целого типа, cтроковые и булевые литералы. Краткое описание литералов представлено в таблице 1.8.

Таблица 1.8.

Описание литералов

|  |  |
| --- | --- |
| Тип литерала | Описание |
| Литералы целого типа | Целочисленные литералы двоичного, восьмеричного или десятичного представления. Не имеют дробных частей или экспонент. |
| Булевые литералы | true, false |
| Строковые литералы | Символы, заключённые в “ ” (двойные кавычки). |

## Объявление данных

Переменные объявляются при помощи конструкции:

def <тип данных><идентификатор>;

Область видимости сверху вниз. В языке SAE-2019 требуется обязательное объявление переменной перед её использованием. Переменные должны находиться внутри программного блока языка. Имеется возможность объявления одинаковых переменных в разных блоках. Каждая переменная получает префикс – название функции, в которой она объявлена.

## Инициализация данных

Для инициализации переменной каким-либо значением применяется конструкция: <идентификатор>=<значение>;

Когда объявляется переменная целочисленного типа, она по умолчанию инициализируется значением 0, переменная строкового типа — пустой строкой (“”).

## Инструкции языка

Все возможные инструкции языка программирования SAE-2019 представлены в таблице 1.11.

Таблица 1.11.

Инструкции языка программирования SAE-2019

|  |  |
| --- | --- |
| Инструкция | Синтаксис на языке программирования SAE-2019 |
| Главная функция | main{ } |
| Объявление переменной | def <тип данных> <идентификатор>; |
| Присваивание | <идентификатор>=<значение>; |
| Объявление функции | fun <тип данных><идентификатор>(<тип данных> <идентификатор>,…){ } |
| Условный оператор | if(условие) […] |
| Возврат данных | return <идентификатор>|<литерал>; |
| Вывод данных | print<идентификатор>|<литерал>; |

## Операции языка

Язык программирования SAE-2019 может выполнять операции, представленные в таблице 1.12.

Таблице 1.12.

Операции языка SAE-2019

|  |  |
| --- | --- |
| Операция | Приоритетность операции |
| ( ) | 1 |
| ? | 2 |
| +, - | 3 |
| \*, /, % | 4 |

Максимальный приоритет имеют операции с значением 4, минимальный 1 соответственно.

## Выражения и их вычисления

Выражением называется совокупность переменных, знаков операций, имён функций, скобок, которая может быть вычислена в соответствии с синтаксисом языка программирования. Результатом вычисления выражения является величина определённого типа. Круглые скобки в выражении используются для изменения приоритета операций. Не допускается запись двух подряд идущих арифметических операций.

## Конструкции языка

Ключевые программные конструкции языка программирования SAE- 2019 представлены в таблице 1.14.

Таблица 1.14.

Программные конструкции языка программирования SAE- 2019

|  |  |
| --- | --- |
| Название | Синтаксис |
| Главная функция | main { } |
| Функция | fun <тип данных><идентификатор>(<тип данных> <идентификатор>,…){ } |

## Область видимости идентификаторов

В языке SAE-2019 переменные обязаны находиться внутри программного блока(функции). Объявление глобальных переменных не предусмотрено. Объявление пользовательских областей видимости не предусмотрено.

## Семантические проверки

На этапе семантического анализа код проверяется на наличие следующих ошибок:

* недопустимый тип данных в выражении
* неверное количество передаваемых параметров в функцию
* отсутствует точка входа(main)
* обнаружено несколько точек входа
* тип функции и возвращаемого значения не совпадают

## Распределение оперативной памяти на этапе выполнения

Транслированный код использует две области памяти. В сегмент констант заносятся все литералы. В сегмент данных заносятся переменные и параметры функций. Локальная область видимости в исходном коде определяется за счет использования правил именования идентификаторов и регулируется их префиксами, что и обуславливает их локальность на уровне исходного кода.

## Стандартная библиотека и её вызов

Функции стандартной библиотеки и их описание представлено в таблице 1.18.

Таблица 1.18.

Состав стандартной библиотеки

|  |  |  |  |
| --- | --- | --- | --- |
| Имя функции | Возвращаемое значение | Параметры функции | Описание |
| negative(a) | byte | a — целочисленный тип | Меняет знак параметра а |
| powb(a,b) | byte | a, b — целочисленный тип | Возведение a в степень b |

## Ввод и вывод данных

Вывод данных осуществляется при помощи ключевого слова print, после него следует та информация, которую необходимо вывести.

## Точка входа

Точкой входа является функция main.

## Препроцессор

Препроцессор в языке SAE-2019 не предусмотрен.

## Соглашения о вызовах

Используется соглашение cdecl, то есть все параметры передаются в стек справа налево, память освобождает вызывающий код.

## Объектный код

Язык транслируется в язык ассемблера.

## Контрольный пример

Пример исходного кода приведен в приложении А.

# СТРУКТУРА ТРАНСЛЯТОРА

## Компоненты транслятора, их назначение и принципы взаимодействия

Входными данными для транслятора является исходный код, написанный на языке программирования SAE-2019.

Выходными данными является объектный код, а также протоколы работы транслятора, описанные в пункте 2.3.

Компоненты транслятора приведены на рисунке 2.1.



Рис. 2.1 Структура транслятора SAE-2019

Изначально исходный код подается на вход лексическому анализатору, который проверяет исходный текст на недопустимые символы, выделяет ключевые слова, идентификаторы и литералы, а также формирует таблицы лексем и идентификаторов.

Следующей фазой транслятора является синтаксический анализатор, на вход которому подается таблица лексем, полученная на этапе лексического анализа. При синтаксически правильном построении программы осуществляется переход к следующему этапу трансляции, иначе, работа транслятора останавливается.

Семантический анализатор представляет собой набор функций, осуществляющих проверку на разных этапах работы транслятора правил. В зависимости от критичности ошибок, возникающих на данном этапе, работа транслятора может быть либо прекращена, либо продолжена.

Генерация кода осуществляется посредством чистой интерпретации, то есть не создаётся промежуточное представление кода. Генерация кода является заключительным этапом, во время исполнения которого формируется объектный код.

## Перечень входных параметров транслятора

В таблице 2.2 представлены входные параметры, которые могут использоваться для представления работы транслятора.

Таблица 2.2

Входные параметры транслятора

|  |  |  |
| --- | --- | --- |
| Параметр | Назначение | Тип |
| -in: | Указывает на файл с исходным кодом. Исходный код содержится в файле с расширением \*.txt | Обязательный |
| -out: | Указывает имя протокола. Если не указан явно, то имя протокола не формируется. | Не обязательный |
| -log: | Указывает имя протокола. Если не указан явно, то имя протокола формируется конкатенацией имени файла исходного кода и постфикса «.log» | Не обязательный |

## Перечень протоколов, формируемых транслятором и их содержимое

В ходе работы транслятор формирует протокол, в соответствии с заданными входными параметрами. -log: <путь к файлу> - содержит информацию о входных параметрах, о количестве символов исходного кода, а также таблицу лексем, таблицу идентификаторов, дерево разбора синтаксического анализатора.

# РАЗРАБОТКА ЛЕКСИЧЕСКОГО АНАЛИЗАТОРА

## Структура лексического анализатора

Структура лексического анализатора представлена на рисунке 3.1.

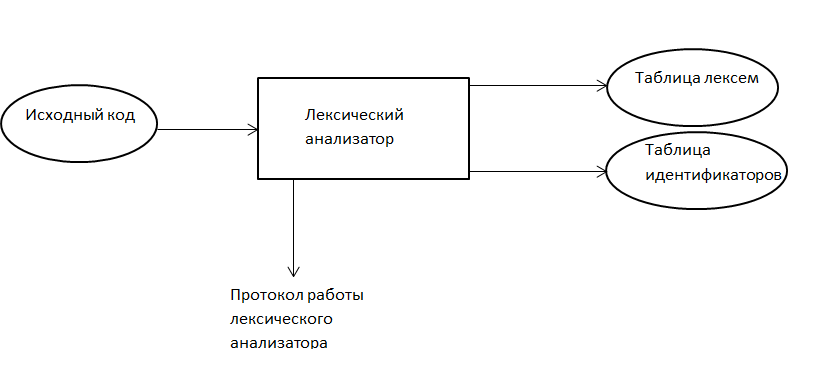


Рис. 3.1 Структура лексического анализатора

Входные данные: исходный код на языке SAE-2019;

Выходные данные: таблица лексем, таблица идентификаторов;

## Контроль входных символов

Таблица допустимости представлена на рисунке 3.2.

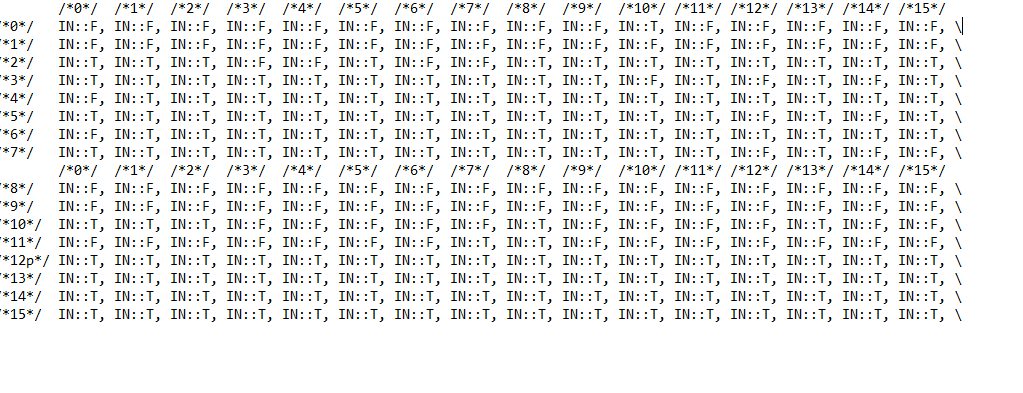


Рис. 3.2 Таблица допустимости входных символов

Таблица допустимости сформирована на основе кодировки Windows-1251. Она предназначена для проверки входных символов на допустимость. Каждому символу в данной таблице соответствует некое числовое значение:

«T» - разрешенные алфавитом символы (также разрешенными являются все символы кроме «F» и «I»);

«F» - запрещенные алфавитом символы;

«I» - символы, которые игнорируются;

## Удаление избыточных символов

Избыточными символами на данном этапе являются пробелы, символы перехода на новую строку.

Алгоритм удаления избыточных символов:

Создаем массив символов SP пробелы между которыми мы можем убрать.

1. Просматриваем текущий символ
   1. Если текущий символ – символ перехода на новую строку, то заменяем его на символ сепаратор “|”
   2. Если встречаем “ (кавычку) -устанавливаем флаг нахождения строкового литерала

1.2.1) если символ равен пробелу и флаг строкового литерала не установлен, то сверяем следующий символ с символами в массиве SP, если символы совпадают удаляем пробел сдвигом исходного кода влево.

2) Перемещаем указатель на байт вправо и переходим к пункту 1.

3) Окончание алгоритма.

## Перечень ключевых слов, сепараторов, символов операций соответствующих им лексем.

Перечень ключевых слов, сепараторов, символов операций соответствующих им лексем представлен в таблице 3.1.

Таблица 3.1

Перечень ключевых слов

|  |  |
| --- | --- |
| Цепочка | Лексема |
| if | c |
| byte | t |
| string | t |
| fun | f |
| print | p |
| return | r |
| main | m |
| dev | d |
| + | v |
| - | v |
| \* | v |
| / | v |
| % | v |
| = | = |
| ( | ( |
| ) | ) |
| { | { |
| } | } |
| [ | [ |
| ] | ] |
| , | , |
| ; | ; |
| идентификатор | i |
| числовой литерал | l |
| строковый литерал | l |
| булевый литерал | l |

## Основные структуры данных

Основные структуры данных, используемые на фазе лексического анализа, представлены в приложении Б.

## Принцип обработки ошибок

При обнаружении критической ошибки, работа транслятора останавливается и происходит запись ошибки в log-файл, а так же происходит вывод в консоль. Подсчет количества ошибок не ведется.

## Структура и перечень сообщений лексического анализатора

Перечень сообщений, генерируемых на этапе лексического анализа, представлен в таблице 3.7.

Таблица 3.7

Сообщения лексического анализатора

|  |  |
| --- | --- |
| Код | Сообщение |
| 40 | Превышен максимальный размер таблицы лексем |
| 41 | Таблица лексем переполнена |
| 42 | Ошибка лексического анализа |
| 43 | Таблица идентификаторов переполнена |
| 44 | Превышен максимальный размер таблицы идентификаторов |
| 52 | Функция main не объявлена |
| 53 | Функция main объявлена повторно |
| 54 | Повторное объявление переменной |
| 55 | Повторное объявление функции |
| 56 | Идентификатор задан неверно |
| 57 | Неизвестный идентификатор |
| 70 | Длинна строки больше максимально допустимой |

## Параметры лексического анализатора и режим его работы

Входным параметром является текст кода на языке SAE-2019. Параметры, определяющие режим работы лексического анализатора, не предусмотрены.

## Контрольный пример

Результат работы лексического анализатора, на вход которого была подана программа на языке SAE-2019, описанная в пункте 1.25, представлен в приложении В.

# РАЗРАБОТКА СИНТАКСИЧЕСКОГО АНАЛИЗАТОРА

## Структура синтаксического анализатора.

Структура синтаксического анализатора представлена на рисунке 4.1.

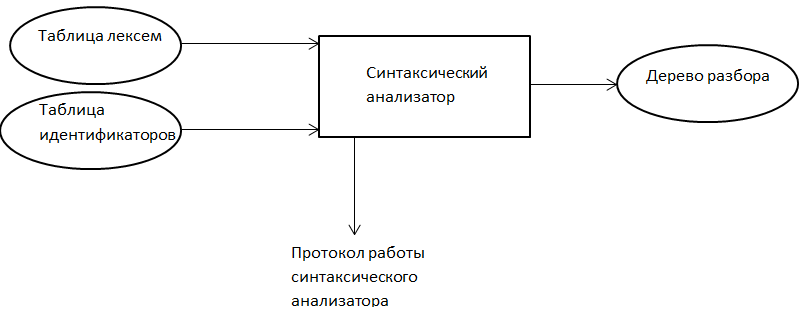


Рис. 4.1 Структура синтаксического анализатора

Входные данные: таблица лексем, таблица идентификаторов.

Выходные данные: дерево разбора.

## Контекстно-свободная грамматика, описывающая синтаксис языка

Грамматика, описывающая язык SAE-2019 представлена в таблице 4.1.

Таблица 4.1

Грамматика языка SAE-2019

|  |  |
| --- | --- |
| Нетерминалы | Описание |
|  | Порождает правила, описывающие общую структуру программы |
|  | Порождает правила, описывающие инструкции языка |
|  | Порождает правила, описывающие выражения |
|  | Порождает правило, описывающее условный блок |
|  | Порождает правила, описывающие формальные параметры функции |
|  | Порождает правила, описывающие фактические параметры функции |

## Построение конченого магазинного автомата

Принцип действия конечного магазинного автомата представлен на рисунке 4.3.



Рис. 4.2 МП-автомат

Формальное описание МП-автомата:

- множество состояний;

- алфавит входных символов;

- специальный алфавит магазинных символов;

-функция переходов автомата , где - множество подмножеств ;

- начальное состояние автомата;

- начальное состояние магазина (маркер дна);

- множество конечных состояний.

Конфигурация (текущее состояние автомата) описывается тройкой , где - текущее состояние автомата, - остаток цепочки, - цепочка-содержимое магазина.

Начальное состояние , - начальное состояние автомата, - входная цепочка, - маркер дна магазина.

Цепочка является допустимой (распознается) автоматом , если и .

Работа автомата

1. состояние автомата
2. читает символ находящийся под головкой (сдвигает ленту);
3. не читает ничего (читает , не сдвигает ленту);
4. из определяет новое состояние , если или .
5. читает верхний (в стеке) символ и записывает цепочку т. к. , при этом, если , то верхний символ магазина просто удаляется.
6. работа автомата заканчивается

## Основные структуры данных

Основные структуры данных и правила перехода, используемые на фазе синтаксического анализа, представлены в приложении Г.

## Описание алгоритма синтаксического разбора

На входе конченого автомата имеется лента, на которой находятся входные символы, а также грамматика в форме Грейбах.

1) если лента не пустая, переход к п.2, иначе переход к п.5

2) если на верхушке магазина нетерминальный символ

2.1) если есть такое правило

2.1.1) если есть цепочка, возвращаем NS\_OK. Переход к п.4.

2.1.2) иначе восстанавливаем состояние. Переход к п.4.

2.2) иначе возвращаем ошибку. Переход к п.4.

3) если на верхушке терминал и он совпадает с символом на ленте, то выталкиваем его из стека и продвигаем ленту. Переход к п.4.

4) Повторить шаг, переход к п.1.

5) Окончание работы

## Структура и перечень сообщений синтаксического анализатора

Перечень сообщений, генерируемых на этапе синтаксического анализа, представлен в таблице 4.6.

Таблица 4.6

Перечень сообщений

|  |  |
| --- | --- |
| Код | Сообщение |
| 80 | Неверная структура программы |
| 81 | Неверная структура функции |
| 82 | Ошибка в выражении |
| 83 | Ошибка в формальных параметрах функции |
| 84 | Ошибка в фактических параметрах функции |
| 85 | Ошибка в условном операторе |

## Параметры синтаксического анализатора и режимы его работы

Входными параметрами для синтаксического анализатора являются таблица лексем и таблица идентификаторов, полученные в ходе лексического анализа. По окончании разбора формируется дерево разбора, которое выводится в протокол работы –log:.

## Принцип обработки ошибок

При обнаружении ошибки в цепочке какого-либо правила, магазинный автомат проходит вверх по дереву разбора, пока не найдет верный вариант. Иначе запоминается самая глубокая ошибка, которая выводится в консоль.

## Контрольный пример

Результатом работы синтаксического анализатора является дерево разбора, содержимое которого представлено в приложении Д.

# РАЗРАБОТКА СЕМАНТИЧЕСКОГО АНАЛИЗАТОРА

## Структура семантического анализатора

Выполнение функций анализатора происходит на различных этапах работы транслятора. Структура семантического анализатора представлена на рисунке 5.1.

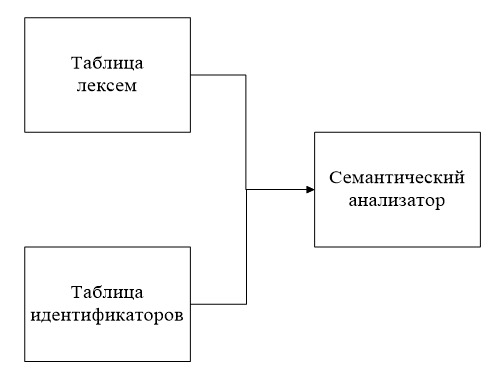


Рис. 5.1. Структура семантического анализатора

## Функции семантического анализатора

Все семантические проверки встроены непосредственно в код этапов транслятора.

В функции PolishNotation для проверки семантических правил используются флаги: Fstring (флаг строкового типа), Foper (флаг оператора), CallFun (флаг вызова функций).

В функции LexAnaliz используются следующие идентификаторы: defFunindex (индекс в таблице лексем идентификатора функции при объявлении), callFunindex (индекс в таблице лексем идентификатора функции при вызове ), bufferCountFunParm (буфер хранящий оставшиеся параметры), numberparm (номер параметра), EqualOperator (индекс в таблице лексем оператора =), Returnindex (индекс в таблице лексем оператора return)

Флаги для семантических проверок: findFun (флаг функции), findParm( флаг параметров), findMain (флаг main), findEqualID (флаг одинаковых идентификаторов), findIF (флаг условной конструкции), findReturn (флаг оператора return).

## Структура и перечень сообщений семантического анализатора

Сообщения, генерируемые при выполнении семантических проверок, представлены в таблице 5.3.

Таблица 5.3

Перечень сообщений

|  |  |
| --- | --- |
| Код | Сообщение |
| 47 | Тип переменной и присваиваемого значения не совпадают |
| 48 | Выражение должно быть допустимым для изменения левосторонним значением |
| 49 | Выражение должно относиться к целочисленному типу |
| 50 | Неверный тип передаваемого параметра |
| 51 | Неверное количество параметров |
| 60 | Функция должна возвращать значение |
| 61 | Оператор return не может находиться внутри условного блока |
| 62 | Неверный возвращаемый тип |
| 71 | Слишком большой целочисленный литерал |
| 72 | Слишком маленький целочисленный литерал |
| 73 | Запрещенный символ литерале восьмеричной системы счисления |
| 74 | Запрещенный символ литерале двоичной системы счисления |

## Принцип обработки ошибок

Ошибки, обнаруженные в ходе семантических проверок, являются критическими и приводят к прекращению работы транслятора. В случае обнаружения ошибки работа транслятора прекращается, сообщение об ошибке выводится в консоль.

## Контрольный пример

Результат работы семантических функций описан в главе 8.

# ПРЕОБРАЗОВАНИЕ ВЫРАЖЕНИЙ

## Выражения, допускаемые языком

В языке SAE-2019 допускаются выражения с использованием числовых идентификаторов и литералов. Также предусмотрены следующие арифметические операции:

* сложения: «+»;
* вычитания: «-»;
* умножения: «\*»;
* деления: «/»;
* остаток от деления: «%»;

Также допустима возможность изменения приоритета выполнения арифметических операций при помощи скобок:

* «(»;
* «)»;

Приоритетность операций представлена в таблице 6.1. Чем выше число, тем выше и приоритет.

Таблица 6.1

Приоритетность операций

|  |  |
| --- | --- |
| Операция | Приоритет |
| «(», «)» | 1 |
| «?» | 2 |
| «+», «-» | 3 |
| «\*», «/», «%» | 4 |

## Польская запись и принцип ее построения

Польская запись — форма записи математических и логических выражений, в которой операнды расположены перед знаками операций. Приоритетность операций приведена в таблице 6.1. Известен следующий принцип построения польской записи:

* исходная строка: выражение;
* результирующая строка: польская запись;
* стек: пустой;
* исходная строка просматривается слева направо;
* операнды переносятся в результирующую строку;
* операция записывается в стек, если стек пуст;
* операция выталкивает все операции с большим или равным приоритетом в результирующую строку;
* отрывающая скобка помещается в стек;
* закрывающая скобка выталкивает все операции до открывающей скобки, после чего обе скобки уничтожаются.

## Программная реализация обработки выражений

Преобразование в польскую запись происходит после этапа лексического и синтаксического анализа.

## Контрольный пример

Контрольный пример разбора выражения содержится в таблице 6.4.

Таблица 6.4

Разбор выражения

|  |  |  |
| --- | --- | --- |
| Выражение | Стек | Результат |
| ((x-3)\*y+1)/7 |  |  |
| (x-3)\*y+1)/7 | ( |  |
| x-3)\*y+1)/7 | ( ( |  |
| -3)\*y+1)/7 | ( ( | x |
| 3)\*y+1)/7 | ( ( - | x |

Окончание табл.

|  |  |  |
| --- | --- | --- |
| Выражение | Стек | Результат |
| )\*y+1)/7 | ( ( - | x 3 |
| \*y+1)/7 | ( | x 3 - |
| y+1)/7 | ( \* | x 3 - |
| +1)/7 | ( \* | x 3 - y |
| 1)/7 | ( + | x 3 - y \* |
| )/7 | ( + | x 3 - y \* 1 |
| /7 |  | x 3 - y \* 1 + |
| 7 | / | x 3 - y \* 1 + |
|  | / | x 3 - y \* 1 + 7 |
|  |  | x 3 - y \* 1 + 7 / |

# ГЕНЕРАЦИЯ КОДА

## Структура генератора кода

В языке SAE-2019 генерация кода является заключительным этапом трансляции. Генератор принимает на вход таблицы лексем и идентификаторов, полученные в результате лексического анализа. В соответствии с таблицей лексем строится выходной файл на языке ассемблера, который будет являться результатом работы транслятора. В случае возникновения ошибок генерация кода не будет осуществляться. Структура генератора кода SAE-2019 представлена на рисунке 7.1.



Рис.7.1 Структура генератора кода

## Представление типов данных в памяти

Элементы таблицы идентификаторов расположены сегментах .data и .const языка ассемблера. Соответствия между типами данных идентификаторов на языке SAE-2019 и на языке ассемблера приведены в таблице 7.2.

Таблица 7.2

Соответствия типов идентификаторов языка SAE-2019 и языка ассемблера

|  |  |  |
| --- | --- | --- |
| Тип идентификатора на языке SAE-2019 | Тип идентификатора на языке ассемблера | Пояснение |
| BYTE | SDWORD | Хранит целочисленный тип данных. |
| BOOL | DWORD | Хранит булевый тип данных. |
| STRING | DWORD | Хранит указатель на начало строки. Строка должна завешаться нулевым символом. |

## Статическая библиотека

В языке SAE-2019 предусмотрена статическая библиотека. Статическая библиотека содержит функции, написанные на языке C++. Объявление функций статической библиотеки генерируется автоматически.

Вызовы стандартных функций доступны там же, где и вызов пользовательских функций. Также в стандартной библиотеке реализованы функции для манипулирования выводом, недоступные конечному пользователю. Эти функции представлены в таблице 7.3.

Таблица 7.3

Дополнительные функции стандартной библиотеки

|  |  |
| --- | --- |
| Функция на языке С++ | Описание |
| void ToConsoleNUM(int num) | Функции для вывода в стандартный поток значения целочисленного идентификатора/литерала. |
| void ToConsoleSTR(char\* str) | Функции для вывода в стандартный поток значения строкового идентификатора/литерала. |
| int negative(int i) | Целочисленная функция. Возвращает число i с противоположным знаком. |

Окончание табл.

|  |  |
| --- | --- |
| Функция на языке С++ | Описание |
| int powb(int x, int y) | Целочисленная функция. Вычисляет и возвращает число x в степени y. |
| void NULLEXCEPTION(int s) | Дополнительная функция. Сигнализирует о делении на 0. |
| void OVERFLOWEXCEPTION(int st) | Дополнительная функция. Сигнализирует о переполнении. |

## Особенности алгоритма генерации кода

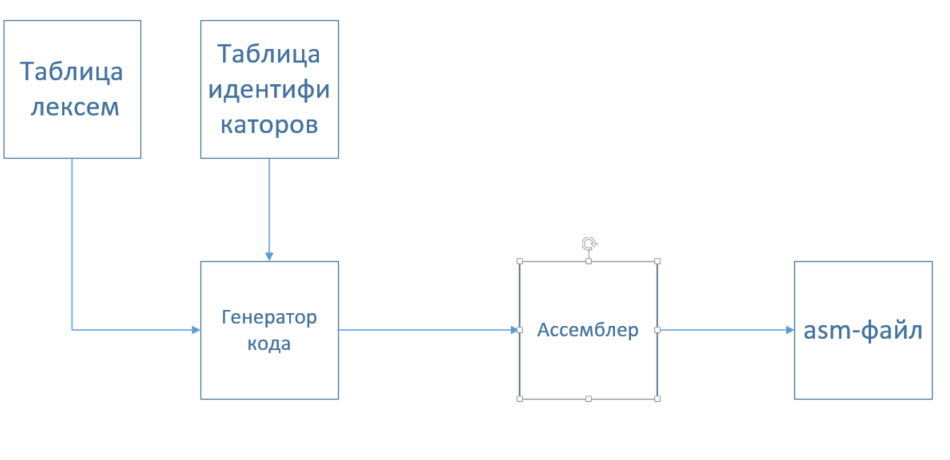
В языке SAE-2019 генерация кода строится на основе таблиц лексем и идентификаторов. Общая схема работы генератора кода представлена на рисунке 7.4

Рис.7.4 Структура генератора кода

## Контрольный пример

Результат генерации ассемблерного кода на основе контрольного примера из приложения А приведен в приложении Е.

# ТЕСТИРОВАНИЕ ТРАНСЛЯТОРА

В данной главе описаны возможные ошибки, возникающие на различных этапах работы транслятора. Результат тестирования представлен в таблице 8.1.

Таблица 8.1

Тестирование транслятора

|  |  |
| --- | --- |
| Исходный код | Генерируемая ошибка |
| main{ def bool bl="str"; } | Ошибка 47: Тип переменной и присваиваемого значения не совпадают Строка: 18 |
| main{def bool bl=true; true=bl; } | Ошибка 48: Выражение должно быть допустимым для изменения левосторонним значением Строка: 18 |
| main{ def byte a= 100b;  print a%2+"str"; } | Ошибка 49: Выражение должно относиться к целочисленному типу Строка: 28 |
| byte fun Fmath (byte x, byte y) { }  main{ def byte b;def string c="Mainc"; b = Fmath(a,c);} | Ошибка 50: Неверный тип передаваемого параметра Строка: 21 |
| byte fun Fmath (byte x, byte y){…}  main{ def byte b;b = Fmath(a,2,2);} | Ошибка 51: Неверное количество параметров Строка: 21 |
| byte fun Fmath (byte x, byte y){…} | Ошибка 52: Функция main не объявлена Строка: 14 |
| main{ }  main{} | Ошибка 53: Функция main объявлена повторно Строка: 33 |
| main{def byte a= 100b; def byte a;} | Ошибка 54: Повторное объявление переменной Строка: 19 |
| byte fun Fmath (byte x, byte y){…}  byte fun Fmath (byte x, byte y){} | Ошибка 55: Повторное объявление функции Строка: 14. |
| main { def byte rezult5=(powb(x,y)-5)\*101b;} | Ошибка 56: Идентификатор задан неверно Строка: 10 |

Окончание табл.

|  |  |
| --- | --- |
| Исходный код | Генерируемая ошибка |
| main{ bl=true;} | Ошибка 57: Неизвестный идентификатор Строка: 17 |
| byte fun Fmath (byte x, byte y){ def byte rezult=(powb(x,y)-5)\*101b; print negative(33o);} | Ошибка 60: Функция должна возвращать значение Строка: 13 |
| def string s=”string”; if(b%50 ? 6);  [return s; print "ABCDe";] | Ошибка 61: Оператор return не может находится внутри условного блока Строка: 24 |
| byte fun Fmath (byte x, byte y)  {def byte result = (powb(x,y) - 5) \*101b;  print negative(33o); return "string";} | Ошибка 62: Неверный возвращаемый тип Строка: 12 |
| string fun Fstr (string s)  { def string str = "Строка из функции Fstrmmmmmmmmmmmmm"; | Ошибка 70: Длинна строки больше максимально допустимой Строка: 3 |
| main{def byte a= 143;} | Ошибка 71: Слишком большой целочисленный литерал Строка: 18 |
| main{def byte a= 19o;} | Ошибка 73: Запрещенный символ литерале восьмеричной системы счисления Строка: 18 |
| main{def byte a= 15b;} | Ошибка 74: Запрещенный символ литерале двоичной системы счисления Строка: 18 |

# ЗАКЛЮЧЕНИЕ

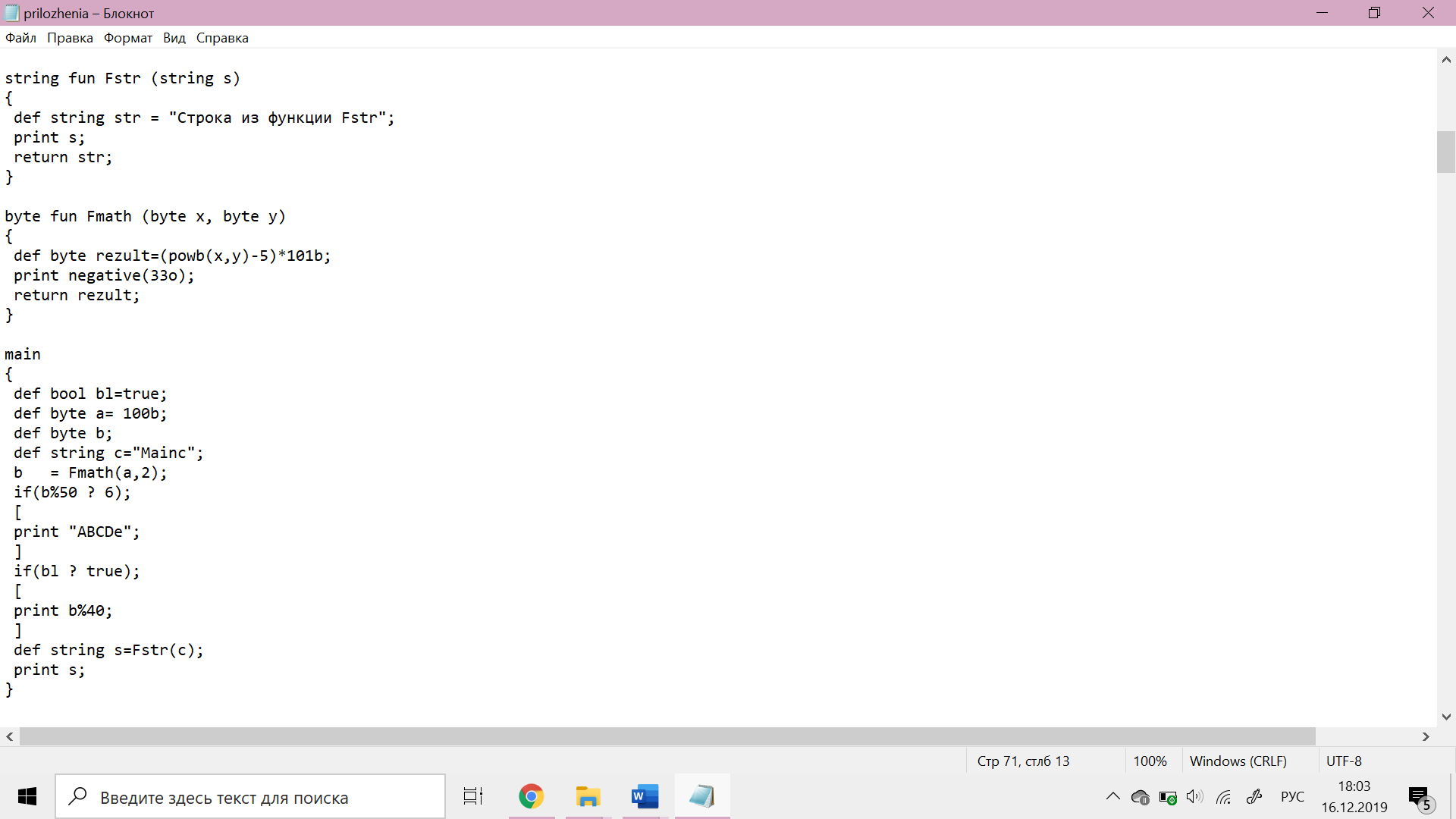
В данном курсовом проекте были выполнены поставленные минимальные требования. В ходе работы было изучено много нового, а также закреплены знания, которые были получены ранее. Также стоит отметить, что данный курсовой проект позволил совместить закрепление знаний сразу по двум языкам программирования, таких как C++ и Assembler. При написании приложения были усвоены такие понятия как синтаксический, лексический и семантический анализатор.

В итоге был получен примитивный язык программирования SAE-2019, который не имеет сложных конструкций, которые реализованы на сегодняшний день во многих других языках программирования.

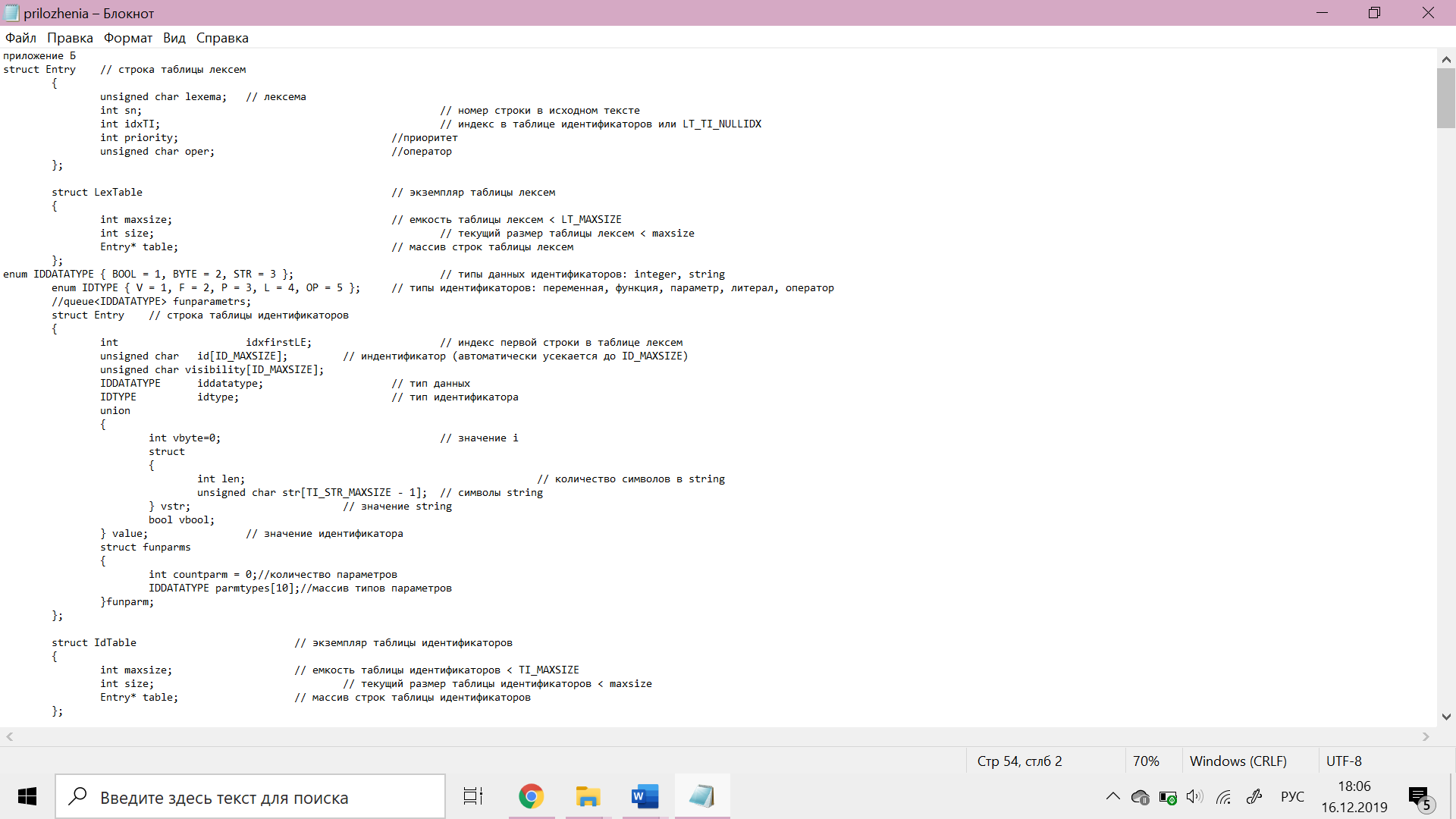
Окончательная версия языка SAE-2019 включает:

1. 3 типа данных;
2. Поддержка оператора вывода;
3. Наличие 5 арифметических операторов для вычисления выражений;
4. Наличие трех систем счисления(двоичной/восьмеричной/десятичной)
5. Наличие условного оператора
6. Наличие двух функций стандартной библиотеки( возведение числа в степень, получение числа с обратным знаком)
7. Большую вариацию ошибок разного рода, большое количество семантических проверок

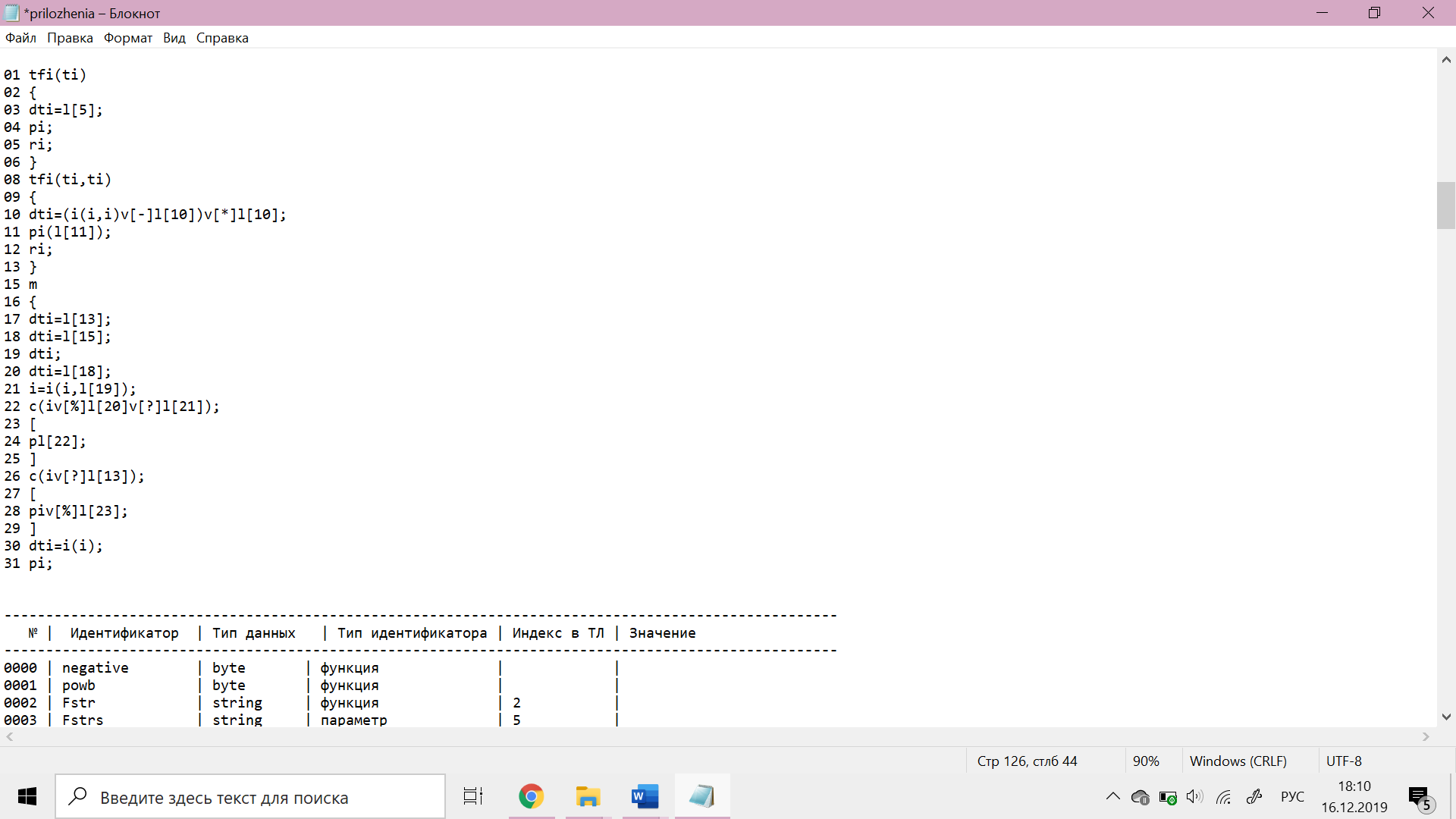
# ПРИЛОЖЕНИЕ А

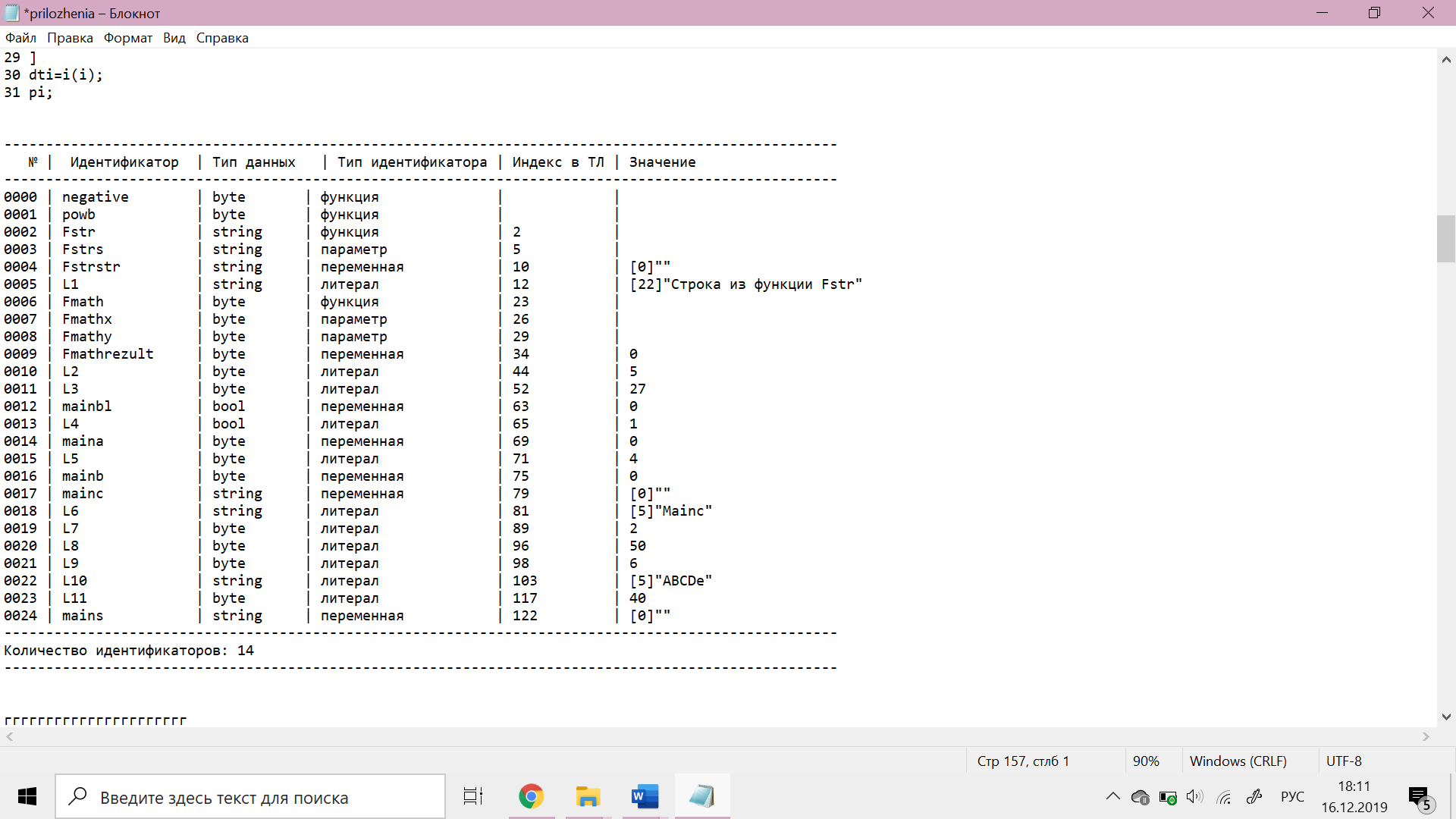


# ПРИЛОЖЕНИЕ Б

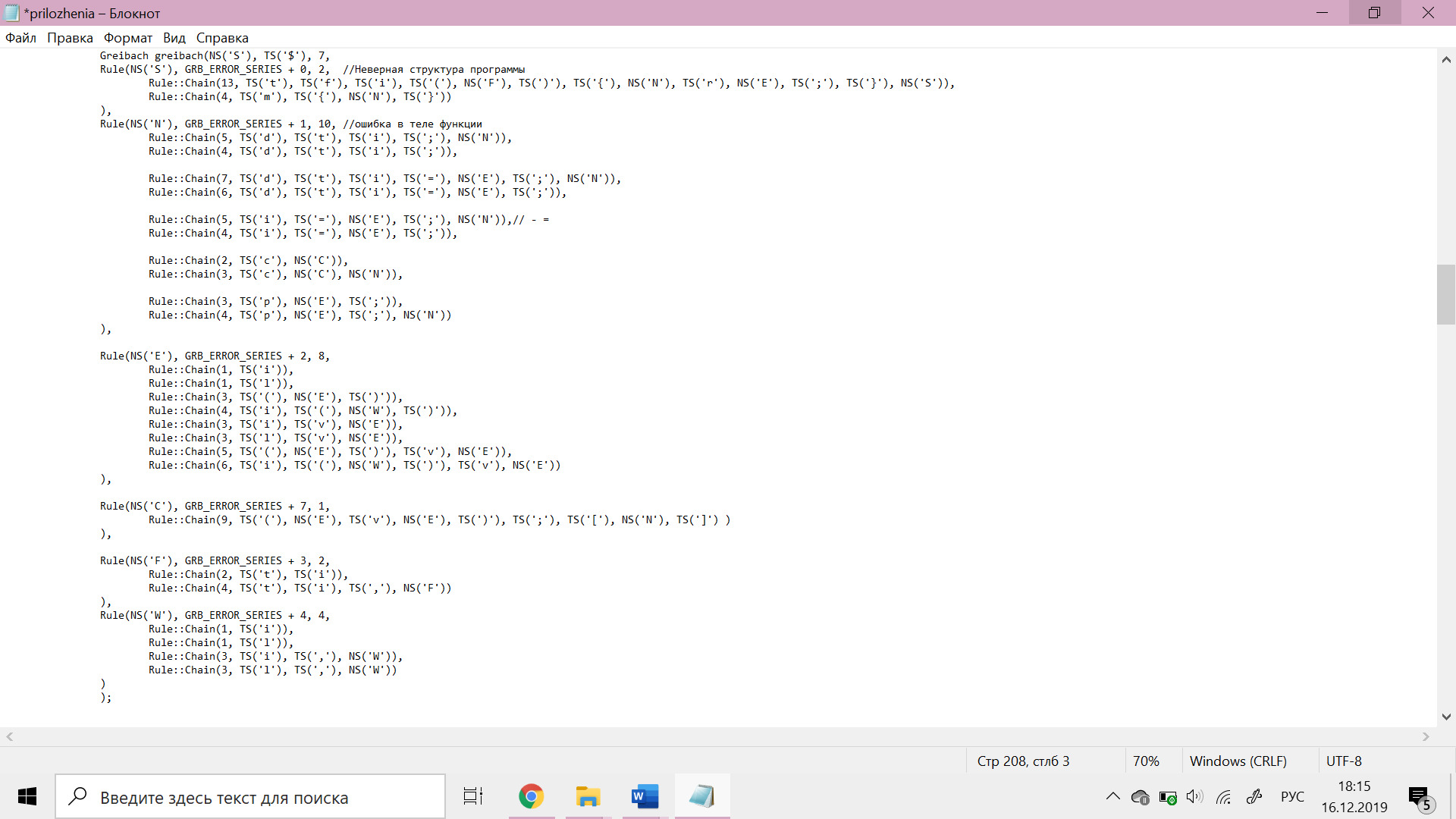


# ПРИЛОЖЕНИЕ В

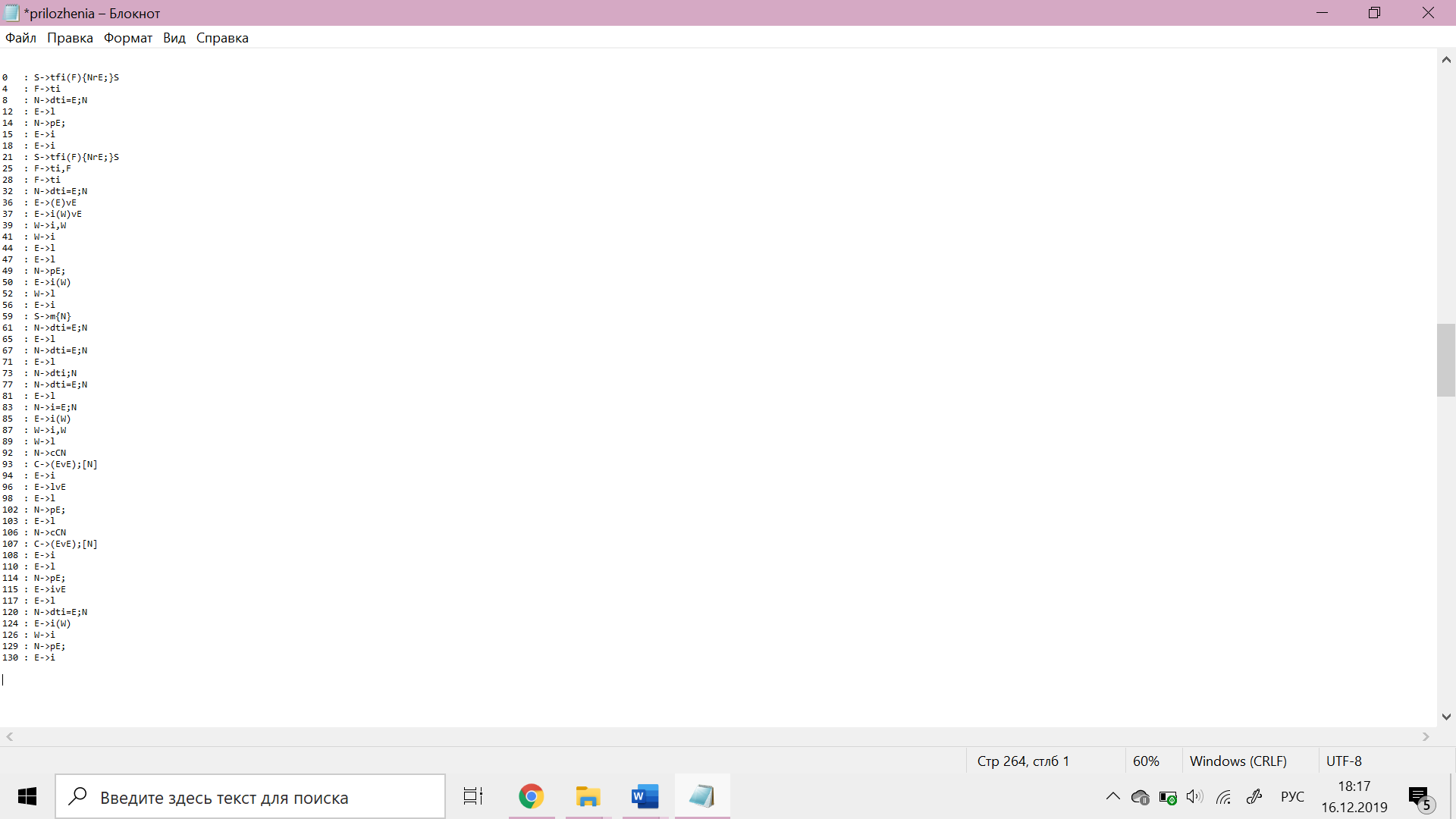




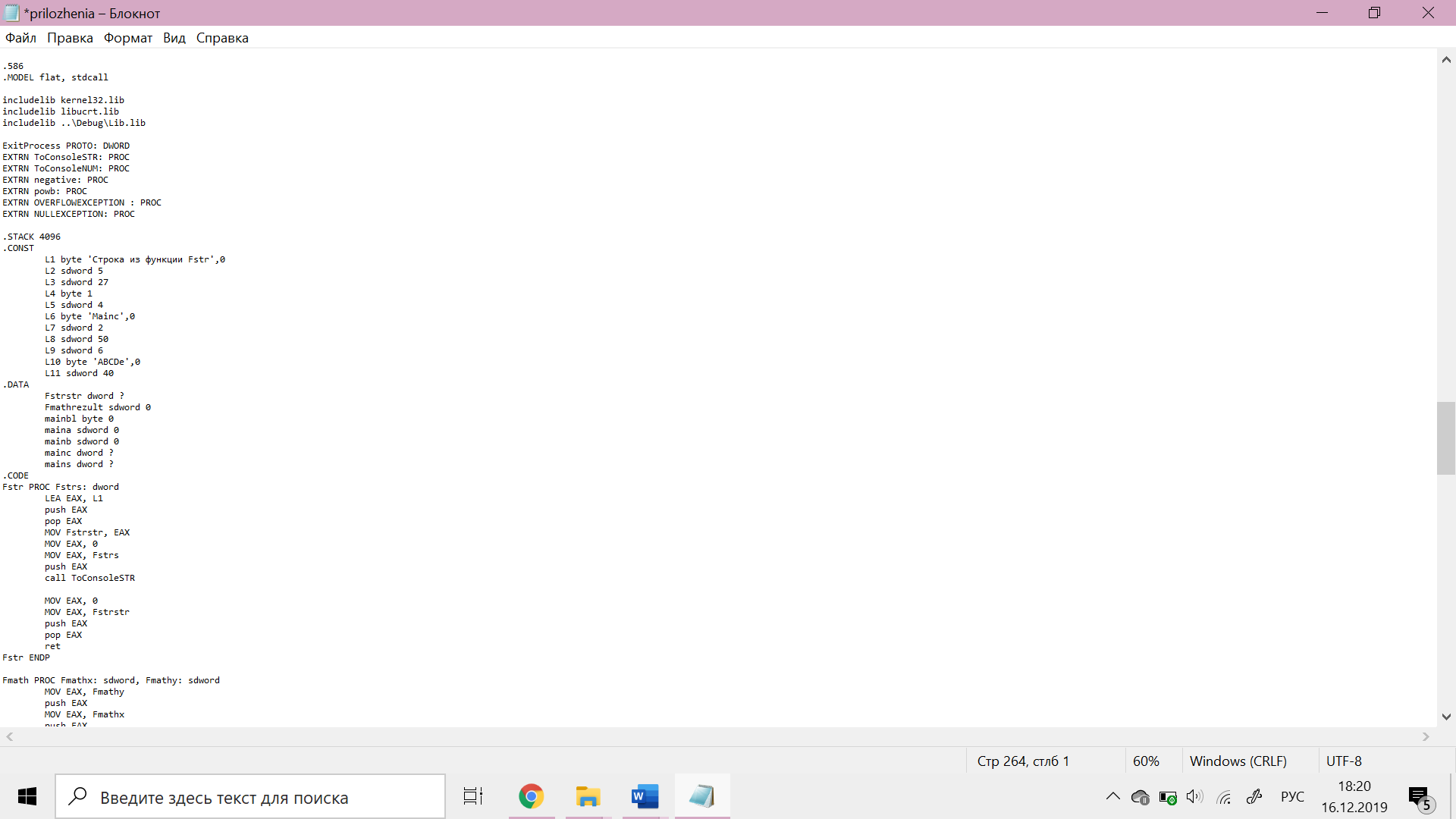
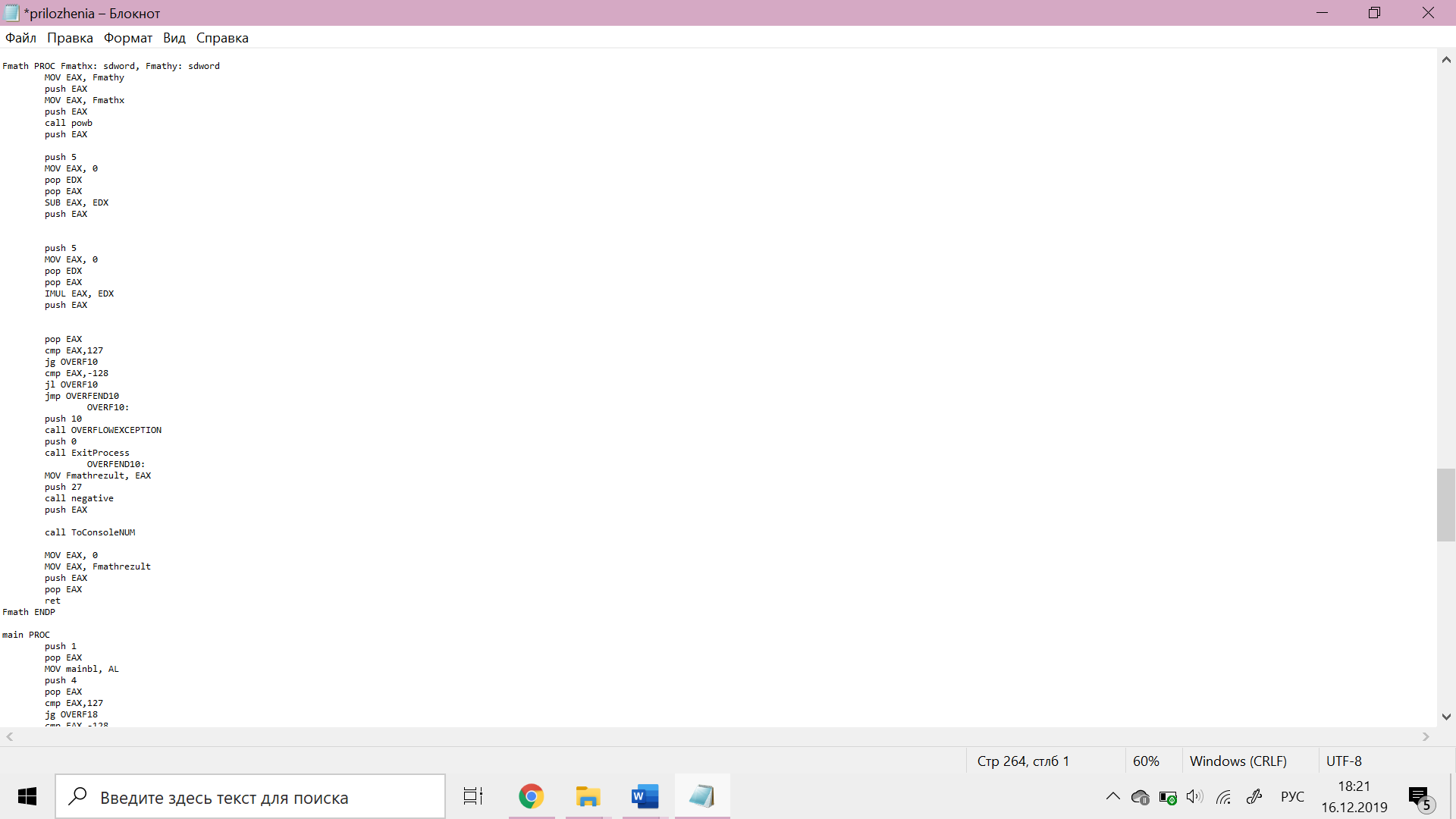
# ПРИЛОЖЕНИЕ Г

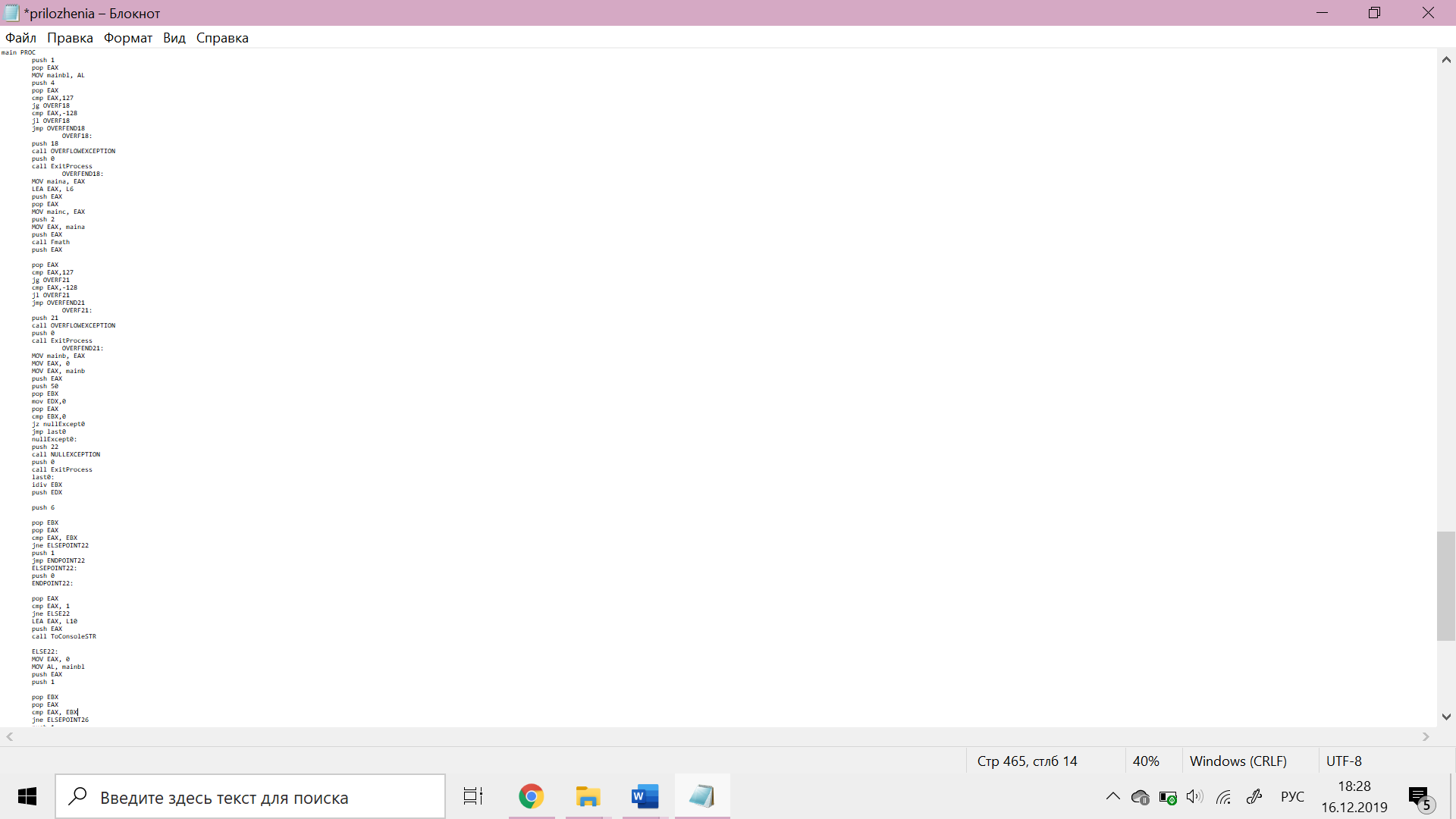
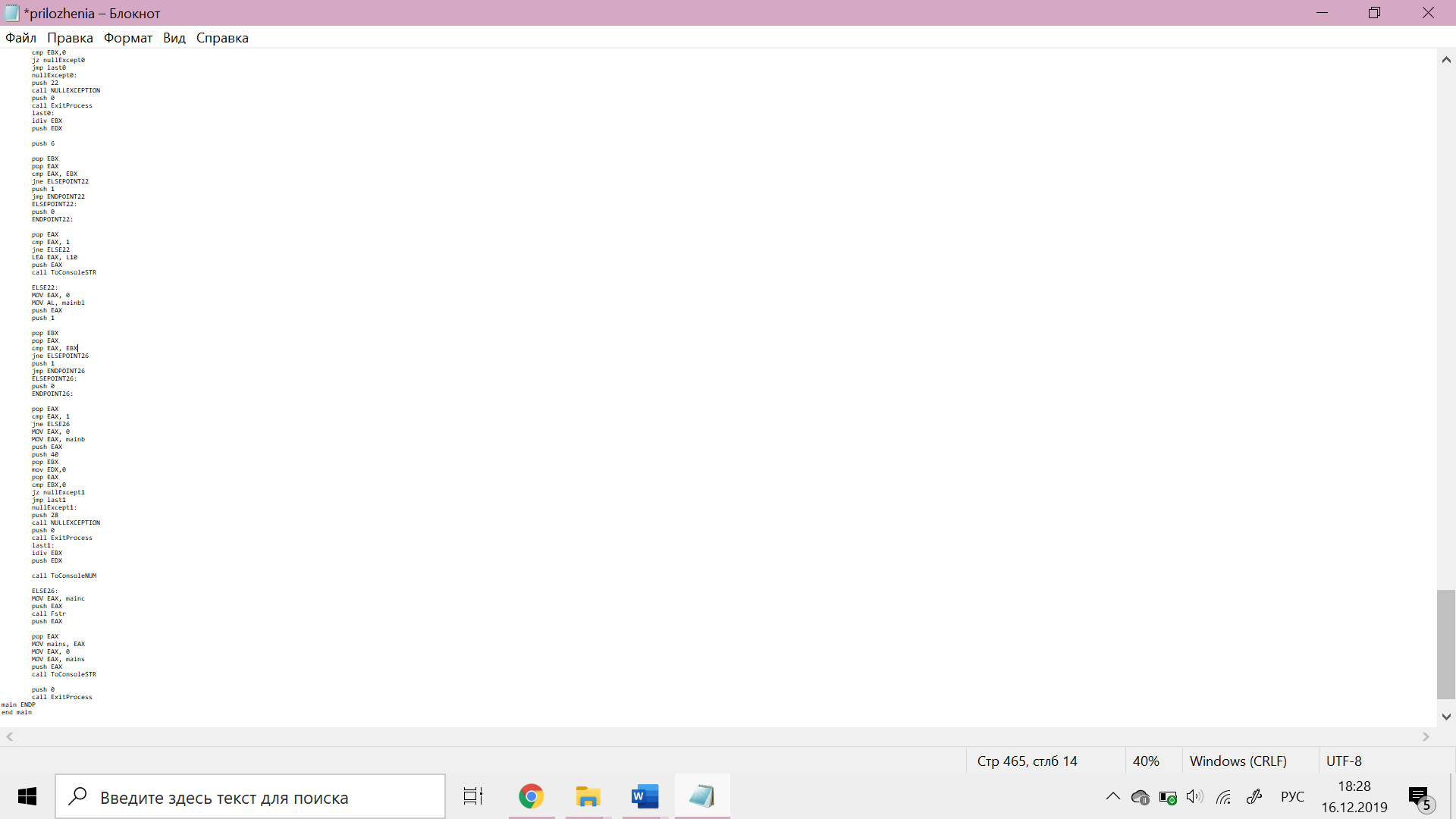


# ПРИЛОЖЕНИЕ Д



# ПРИЛОЖЕНИЕ Е



СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Ахо, А. Компиляторы: принципы, технологии и инструменты / А. Ахо, Р. Сети, Дж. Ульман. – M.: Вильямс, 2003. – 768с.

2. Смелов, В.В. Курс лекций по предмету языки программирования – 2016

3. Прата, С. Язык программирования С++. Лекции и упражнения / С. Прата. – М., 2006 — 1104 c.

4. Страуструп, Б. Принципы и практика использования C++ / Б. Страуструп – 2009 – 1238 с