Введение

Целью курсового проекта является написание спецификации и разработка транслятора для собственного языка программирования.

Название языка ­­– PMN-2019. Трансляция будет производиться в язык ассемблера.

Этапы выполнения курсового проекта:

1. Написание спецификации языка программирования;
2. Разработка лексического анализатора;
3. Разработка синтаксического анализатора;
4. Разработка семантического анализатора;
5. Преобразование арифметических выражений;
6. Генерация кода;
7. Тестирование транслятора.

Подробная информация о каждом этапе выполнения курсового проекта приведена в соответствующих разделах пояснительной записки:

В первом разделе приведена спецификация языка.

Во втором разделе описана структура транслятора.

В третьем разделе описаны принцип работы и этапы разработки лексического анализатора, определены разрешенные символы и ключевые слова языка программирования.

В четвертом разделе описан принцип работы синтаксического анализатора, формальная грамматика определена и приведена в нормальную форму Грейбах для выполнения синтаксического разбора.

В пятом разделе описаны принцип работы и основные функции семантического анализатора.

В шестом разделе описаны выражения допускаемые языком, форма, принципы построения и вычисления выражений.

В седьмом разделе описан процесс генерации кода.

В восьмом разделе приведены примеры тестирования транслятора.

1. Спецификация языка программирования
   1. Характеристика языка программирования

Язык PMN-2019 является универсальным компилируемым строго типизированным процедурным высокоуровневым языком программирования, поддерживающим парадигму функционального программирования.

* 1. Определение алфавита языка программирования

Алфавит языка PMN-2019 основан на кодировке Windows-1251, представленной на рисунке 1.1.



Рисунок 1.1 – Алфавит входных символов языка PMN-2019

Символы, используемые на этапе выполнения, на рисунке помечены как разрешенные.

* 1. Применяемые сепараторы

Сепараторы, применяемые в языке PMN-2019, приведены в таблице 1.1.

Таблица 1.1 - Применяемые сепараторы

|  |  |
| --- | --- |
| Разделители | Назначение |
| ‘пробел’, ‘табуляция’,  ‘переход на новую строку’ | Разделяют входные лексемы |
| +, -, \*, /, % | Арифметические операторы. Используются в арифметических операциях |
| = | Оператор присваивания. Используется для присваивания значения переменной |
| <, >, !=, == | Условные операторы. Используются для сравнения переменных и литералов |
| ( ) | Блок параметров функции, так же указывает приоритет в арифметических операциях |
| , | Разделяет параметры функции |
| { } | Ограничивают программные конструкции |
| ; | Признак конца инструкции языка |

* 1. Применяемые кодировки

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

* 1. Типы данных

В языке PMN-2019 поддерживается 3 типа данных: целочисленный, строковый и булевый. Подробная описание типов данных приведено в таблице 1.2.

Таблица 1.2 - Типы данных языка PMN-2019

|  |  |
| --- | --- |
| Тип данных | Характеристика |
| Целочисленный  (dig) | В памяти занимает 1 байт.  Максимальное значение: 127.  Минимальное значение: -128.  Принцип размещения в памяти:  Последний бит числа отведен под знак, оставшиеся 7 бит предназначены для хранения значения числа.  Значение по умолчанию: 0.  Применяемые операции: в арифметических выражениях и условных конструкциях к целочисленным переменным и литералам применимы все арифметические и условные операции соответственно, поддерживаемые языком PMN-2019. |
| Строковый  (str) | В памяти занимает n + 1 байт, где n – количество символов в строке.  Максимальное количество символов: 255.  Принцип размещения в памяти:  Каждый символ строки занимает 1 байт. В конце строки располагается NULL символ (признак конца строки).  Применяемые операции: к строковым переменным и литералам операции не применяются. |
| Булевый  (bool) | В памяти занимает 1 байт.  Может принимать одно из двух значений: true или false.  Принцип размещения в памяти:  В зависимости от значения 1 бит числа установлен true: 1, false: 0.  Применяемые операции: к булевым переменным и литералам применимы все условные операции, поддерживаемые языком PMN-2019. |

* 1. Преобразование типов данных

Преобразования типов данных в языке программирования PMN-2019 не допустимо.

* 1. Идентификаторы

Идентификатор языка PMN-2019 – это имя, используемое для переменных, функций, параметров. Идентификаторы могут состоять как из одного, так и из нескольких символов. Первым символом должна быть маленькая буква латинского алфавита, а за ним могут стоять маленькие буквы латинского алфавита или цифры. Так же идентификаторы не могут совпадать с ключевыми словами языка PMN-2019.

Пример корректных идентификаторов: stroka1, fdsgf141 и т.п.

Пример некорректных идентификаторов: 14strok, Stroka1, \_stroka1 и т.п.

* 1. Литералы

Литерал языка PMN-2019 – это запись в исходном коде программы, представляющая собой фиксированное значение.

Таблица 1.3 - Литералы языка PMN-2019

|  |  |
| --- | --- |
| Тип литерала | Характеристика |
| Целочисленный | Десятичный:  Последовательность десятичных цифр 0..9 с предшествующим знаком минус или без него.  Двоичный:  Последовательность двоичных цифр 0 и 1 с предшествующим знаком минус или без него, в конце которой стоит символ ‘B’ (признак двоичного целочисленного литерала).  Восьмеричный:  Последовательность восьмеричных цифр 0..7 с предшествующим знаком минус или без него, в конце которой стоит символ ‘O’ (признак восьмеричного целочисленного литерала).  Шестнадцатеричный:  Последовательность шестнадцатеричных чисел 0..F с предшествующим знаком минус или без него, в конце которой стоит символ ‘H’ (признак шестнадцатеричного целочисленного литерала).  Допустимый диапазон значений:  От -128 до 127 в десятичной системе исчисления. |
| Строковый | Набор, состоящий из символов русского и латинского алфавитов и десятичных цифр, заключенный в двойные кавычки.  Допустимый диапазон значений:  От 0 до 255 символов. |
| Булевый | true или false, где true: логическая единица, false: логический ноль.  Допустимый диапазон значений:  true или false. |

Пример корректных литералов: 23, -11, 4FH, -1001B, 34O, false, “sroka” и т.п.

Примеры некорректных литералов: -4А, ‘sroka’, 1010, trui и т.п.

* 1. Объявление данных

Для объявления переменной используется ключевое слов new, после которого указывается тип переменной и имя идентификатора. Так же при объявлении допускается инициализация переменной.

new <тип> <имя идентификатора>;

new <тип> <имя идентификатора> = <литерал>;

Переменную можно объявить в блоке main, в блоке функции, в условном блоке if..else или в блоке while. Область видимости переменной будет соответствовать тому блоку, в котором она находиться.

* 1. Инициализация данных

В языке PMN-2019 присутствует 2 вида инициализации:

1. Инициализация в месте объявления

new <тип> <имя идентификатора> = <литерал>;

1. Инициализация после объявления

<имя идентификатора> = <литерал>;

Так же в языке присутствует инициализация по умолчанию. Переменные типа dig по умолчанию инициализируются нулем. Переменные типа str по умолчанию инициализируются пустой строкой. Переменные типа bool по умолчанию инициализируются значение false: логический ноль.

* 1. Инструкции языка

Инструкции языка PMN-2019 приведены в таблице 1.4.

Таблица 1.4 – Инструкции языка PMN-2019

|  |  |
| --- | --- |
| Инструкция | Форма записи |
| Объявление переменной | **type** <тип данных> <идентификатор>. |
| Объявление переменной с явной инициализацией | **type** <тип данных> <идентификатор> = <значение>|<выражение>;  Значение – литерал, идентификатор, вызов функции соответствующего типа данных |
| Объявление функции | <тип данных> **function** <идентификатор> (<тип данных> <идентификатор>, …)  {  / программный блок /  **return** <идентификатор/литерал>.  } |
| Вызов функции | <идентификатор> (<идентификатор>, ...) |
| Присвоение значения | <идентификатор> = <значение>; |
| Вывод данных | **out** <идентификатор/литерал>; |
| Вывод данных с переходом на новую строку | **outl** <идентификатор/литерал>; |
| Возвращаемое значение | **return** <литерал/идентификатор>. |

* 1. Операции языка

Операции языка PMN-2019 представлены в таблице 1.5.

Наибольшая приоритетность у операций умножения, деления и деления с остатком, затем идут операции сложения и вычитания. Помещение операции в скобки задают ей самый высокий приоритет.

Таблица 1.5 — Операции языка PMN-2019

|  |  |
| --- | --- |
| Тип оператора | Оператор |
| Арифметические операции языка | + – сложение  - – вычитание  \* – умножение  / – деление  % – остаток от деления |
| Логические | > – больше  < – меньше  == – проверка на равенство  != – проверка на неравенство |

* 1. Выражения и их вычисления

Выражение языка программирования PMN-2019 – это совокупность переменных, литералов, вызовов функций, знаков операций, скобок, которая может быть вычислена в соответствии с синтаксисом языка.

Правила составления выражений:

1. Выражения записываются в одну строку;
2. В выражении могут присутствовать только операнды одинакового типа;
3. В выражении могут использоваться функции. Как стандартные, так и пользовательские;
4. В выражении не могут идти подряд два оператора;
5. Допускается использование круглых скобок для смены приоритета операций.

В арифметических выражениях допускаются только операнды целочисленного типа. В выражениях сравнения допускаются операнды булевого и целочисленного типов.

Перед генерацией кода выражения приводятся к ПОЛИЗ для более удобного вычисления на языке ассемблера.

* 1. Конструкции языка

Конструкции языка PMN-2019 приведены в таблице 1.6.

Таблица 1.6 – Конструкции языка PMN-2019

|  |  |
| --- | --- |
| Конструкция | Описание |
| Главная функция | main  {  …  return <имя переменной/литерал>;  }; |
| Пользовательская функция | <тип возвращаемого значения> function(<тип параметра> <имя параметра>, …)  {  …  return <имя переменной/литерал>;  };  Максимальное количество параметров: 8. |
| Условная конструкция | if(<имя переменной/литерал><условный оператор><имя переменной/литерал>)  {  …  }  else  {  ...  }  Блок else необязателен. |
| Конструкция цикла | while(<имя переменной/литерал><условный оператор><имя переменной/литерал>)  {  …  } |

* 1. Область видимости идентификатора

Каждой конструкции языка PMN-2019 соответствует своя область видимости. Причем функции имеют глобальную область видимости. Однако внутри главной и пользовательских функций действует принцип сокрытия переменных. А именно: переменные, определенные внутри условной конструкции или конструкции цикла могут скрывать внешние переменные с тем же именем.

Объявление идентификаторов вне функции невозможно, так как глобальные переменные языком не поддерживаются.

* 1. Семантические проверки

Семантическим анализатором языка PMN-2019 предусмотрены следующие проверки:

1. Наличие блока main, точки входа в программу;
2. Единственная точка входа в программу;
3. Использование идентификаторов до их объявления;
4. Переопределение идентификаторов;
5. Соответствие параметров, передаваемых в функцию, с параметрами в объявлении функции;
6. Соответствие типа возвращаемого значения с типом функции;
7. Соответствие типов в выражениях;
8. Превышение размера целочисленных и строковых литералов.
   1. Распределение оперативной памяти на этапе выполнения

Для запоминания промежуточных результатов в вычислении выражения используется стек. В сегмент констант записываются все литералы языка. В сегмент данных записываются все имена переменных.

* 1. Стандартная библиотека и её состав

В языке PMN-2019 предусмотрена стандартная библиотека, которая включает в себя набор стандартных функций, а так же функций вывода в консоль. Функции, входящие в состав стандартной библиотеки представлены в таблице 1.7.

Таблица 1.7 – Функции стандартной библиотеки языка PMN-2019

|  |  |
| --- | --- |
| Прототип функции | Описание |
| power(dig x, dig y); | Возводит число x в степень y и возвращает результат. |
| abser(dig x); | Берет абсолютное значение числа x и возвращает результат. |
| nout(dig x); | Выводит целочисленный идентификатор или литерал на консоль. |
| sout(str x); | Выводит строковый идентификатор или литерал на консоль. |
| noutl(dig x); | Выводит целочисленный идентификатор или литерал на консоль, переводя каретку на новую строку. |
| soutl(str x); | Выводит строковый идентификатор или литерал на консоль, переводя каретку на новую строку. |

Стандартная библиотека написана на языке С++, подключается к транслированному коду на этапе генерации кода. Вызовы стандартных функций доступны там же, где и вызов пользовательских функций.

* 1. Ввод и вывод данных

Вывод данных на консоль осуществляется за счет операторов **out** и **outl**. Использование данных операторов допускается только с идентификаторами или литералами.

Функции, управляющие выводом данных на консоль, реализованы на языке C++. На этапе генерации кода операторы вывода языка PMN-2019 заменяются на встроенные функции, находящиеся в стандартной библиотеке.

* 1. Точка входа

В языке PMN-2019 точкой входа является блок главной функции **main.** С первой строчки которого начинается выполнение инструкций программы.

* 1. Препроцессор

Препроцессор в языке программирования PMN-2019 не предусмотрен.

* 1. Соглашение о вызовах

В языке PMN-2019 вызов функций происходит по соглашению о вызовах stdcall. Данное соглашение имеет следующие особенности:

1. Все параметры функции передаются через стек;
2. Освобождением памяти занимается вызываемый код;
3. Параметры в стек заносятся справа налево.
   1. Объектный код

Язык программирования PMN-2019 транслируется в язык ассемблера.

* 1. Классификация сообщений транслятора

Описание и классификация сообщений транслятора об ошибках приведено в таблице 1.8.

Таблица 1.8 – Описание ошибок транслятора языка PMN-2019

|  |  |
| --- | --- |
| Интервал кодов | Описание |
| 0-9 | Системные ошибки. |
| 10-19 | Ошибки параметров. |
| 20-29 | Ошибки файлов. |
| 110-129 | Ошибки лексического анализатора. |
| 130-149 | Ошибки семантического анализатора. |
| 600-609 | Ошибки синтаксического анализатора. |

* 1. Контрольный пример

Контрольный пример языка PMN-2019 представлен в разделе Приложения.

1. Структура транслятора
   1. Компоненты транслятора их назначение и принципы взаимодействия

Транслятор – это программа преобразующая исходный код на одном языке программирования в исходный код на другом языке.

Схема, поясняющая принцип работы транслятора, приведена на рисунке 2.1.

Трансляция исходного кода в язык ассемблера разделена на 4 этапа: Лексический анализ, синтаксический анализ, семантический анализ и генерация кода.

Этапы выполняются последовательно. У каждого этапа есть входные и выходные данные, которые тот или иной этап получает от предыдущей и передает следующей части трансляции.

Первой частью трансляции является лексический анализ. На вход лексического анализатора подается исходный код программы. В свою очередь лексический анализатор производит деление исходного кода программы на слова (токены), которые затем идентифицируются и заменяются на лексемы (внутреннее представление). На выходе лексического анализатора мы имеем две таблицы: таблицу лексем и таблицу идентификаторов.

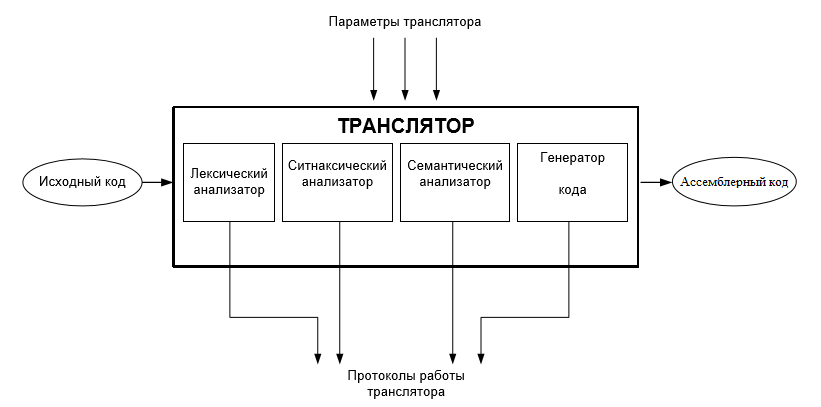


Рисунок 2.1 – Структура транслятора языка PMN-2019

Синтаксический анализ является второй частью работы транслятора. Синтаксический анализатор выполняет синтаксический анализ. Входом для синтаксического анализатора является таблица лексем и таблица идентификаторов. Выходом – дерево разбора.

Затем выполняется семантический анализ. Задача семантического анализатора: проверка соблюдения в исходной программе семантических правил входного языка. Входом для семантического анализатора является таблица идентификаторов и дерево разбора.

Последним этапом трансляции является генерация кода. Вход для генератора когда: таблица лексем и таблица идентификаторов. Выход – кода на языке ассемблера.

* 1. Перечень входных параметров транслятора

Входные параметры транслятора языка программирования PMN-2019 перечислены в таблице 2.1. Входные параметры необходимы для формирования файлов с результатами работы транслятора.

Таблица 2.1 – Входные параметры транслятора языка PMN-2019

|  |  |  |
| --- | --- | --- |
| Ключ и входной параметр | Описание | Значение по умолчанию |
| -in:<путь к файлу> | Текстовый файл с исходным кодом на языке PMN-2019. | Отсутствует |
| -out:<путь к файлу> | Выходной файл, содержащий исходный код на языке ассемблера. | <имя файла in>.out |
| -log:<путь к файлу> | Файл с протоколом работы транслятора. | <имя файла in>.log |
| -tkn:<путь к файлу> | Файл, содержащий исходный текст программы во внутреннем представлении. | <имя файла in>.tkn |
| -lex:<путь к файлу> | Файл с таблицей лексем. | <имя файла in>.lex |
| -id:<путь к файлу> | Файл с таблицей идентификаторов. | <имя файла in>.id |
| -sin:<путь к файлу> | Файл с деревом разбора синтаксического анализатора. | <имя файла in>.sin |

* 1. Протоколы, формируемые транслятором

Протоколы, формируемые транслятором языка PMN-2019 перечислены в таблице 2.2.

Таблица 2.2 – Протоколы, формируемые транслятором языка PMN-2019

|  |  |
| --- | --- |
| Протокол | Описание |
| Файл, заданный параметром “-log:” | Содержит общую информацию о ходе выполнения трансляции: перечисление входных параметров, количество символов и строк, успех или ошибку по каждому этапу трансляции. В случае возникновения ошибки, в файл будет выведен номер ошибки. |
| Файл, заданный параметром “-tkn:” | Содержит исходный текст программы во внутреннем представлении языка. |
| Файл, заданный параметром “-lex:” | Содержит таблицу лексем, итог работы лексического анализа. |
| Файл, заданный параметром “-id:” | Содержит таблицу идентификаторов, итог работы лексического анализа. |
| Файл, заданный параметром “-sin:” | Содержит дерево разбора, итог работы синтаксического анализатора. |

1. Разработка лексического анализатора
   1. Структура лексического анализатора

Лексический анализатор – это программа, преобразующая исходный текст программы, заменяя лексические единицы их внутренним представлением – лексемами, для создания промежуточного представления исходной программы. Структурная схема лексического анализатора приведена на рисунке 3.1.

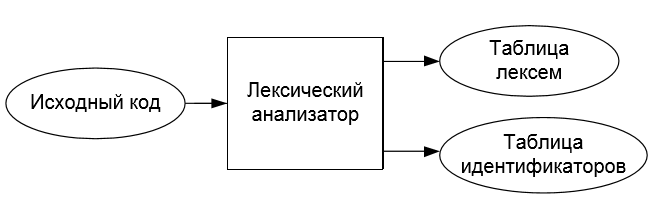


Рисунок 3.1 – Структурная схема лексического анализатора

Лексический анализ в языке PMN-2019 состоит из двух частей:

1. Разбиение исходного кода программы на слова (токены).
2. Идентификация слов (токенов) и последующая их замена на лексемы (внутренне представление). Заполнение таблиц лексем и идентификаторов.

Входные данные: исходный код.

Результат работы: Таблица лексем и таблица идентификаторов.

* 1. Контроль входных символов

Для проверки входных символов на допустимость была написана таблица входных символов, которая дублируют таблицу кодировки Windows-1251. Разрешенные символы в таблице помечены символом T, запрещенные – F. Таблица контрольных символов языка PMN-2019 приведена на рисунке 3.2.

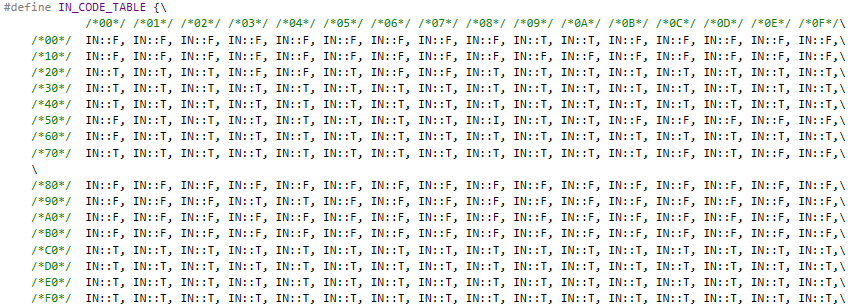


Рисунок 3.2 – Таблица контрольных символов языка PMN-2019

* 1. Удаление избыточных символов

Избыточный символ – это символ, отсутствие которого никоем образом не влияет на исходный текст программы. В языке PMN-2019 символ пробела и табуляции являются избыточными символы. Их удаление предусмотрено на этапе разбиения исходного кода программы на слова (токены).

Алгоритм удаления избыточных символов

Пока есть символы для чтения:

* Читаем очередной символ;
* Если символ является пробелом или табуляцией:

1. Если идёт запись слова, пробел или табуляция являются символом сепаратором, сигнализирующем о начале или конце записи слова (токена);
2. Иначе символ пробела или табуляции пропускаются.
   1. Перечень ключевых слов

Все ключевые слова языка PMN-2019, сепараторы, символы операций, соответствующие им лексемы и регулярные выражения приведены в таблице 3.1.

Таблица 3.1 – Все ключевые слова, сепараторы и т.д. языка PMN-2019

|  |  |  |
| --- | --- | --- |
| Слово (токен) | Лексема | Описание |
| dig | d | Целочисленный тип. |
| str | s | Строковый тип. |
| bool | b | Булевый тип. |
| идентификатор | i | Идентификатор любого типа языка. |
| литерал | l | Литерал любого типа языка. |
| while | w | Начало конструкции цикла. |
| if | q | Условный оператор и истинная ветвь условного оператора. |
| else | 2 | Ложная ветвь условного оператора. |
| function | f | Начало объявления функции. |
| new | n | Объявление переменной. |
| return | r | Выходи из функции и возврат значения. |
| out | o | Вывод данных в консоль. |
| outl | \_ | Вывод данных в консоль с переносом каретки на новую строку. |
| main | m | Главная функция (точка входа в программу). |
| pow | p | Стандартная функция, возведения в степень целочисленного литерала, языка. |
| abs | a | Стандартная функция, взятия абсолютного значения целочисленного литерала, языка. |
| , | , | Разделитель параметров функции. |
| ; | ; | Признак конца инструкции. |
| { | { | Начало тела функции. |
| } | } | Конец тела функции. |
| ( | ( | Начало перечислений параметров у функции, приоритет операций в выражениях. |
| ) | ) | Конец перечислений параметров у функции, приоритет операций в выражениях. |
| + | + | Оператор сложения. |
| - | - | Оператор вычитания. |
| \* | \* | Оператор умножения. |
| / | / | Оператор деления. |
| % | % | Оператор остатка от деления. |
| > | < | Логический оператор меньше. |
| < | > | Логический оператор больше. |
| == | & | Логический оператор равно. |
| != | ! | Логический оператор не равно. |
| = | = | Знак присваивания. |

Пример графов переходов конечных автоматов соответствующих регулярных выражений приведены на рисунке 3.3.

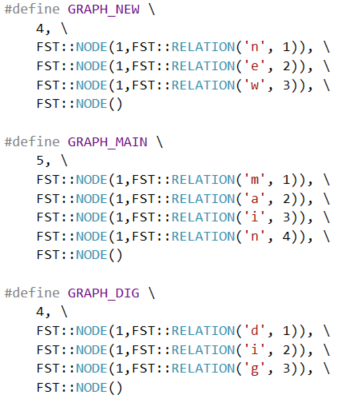


Рисунок 3.3 – Фрагмент графов переходов языка PMN-2019

На рисунке 3.4 приведен фрагмент кода функции на языке C++, реализующей алгоритм разбора входной цепочки в соответствии с графами переходов языка PMN-2019.

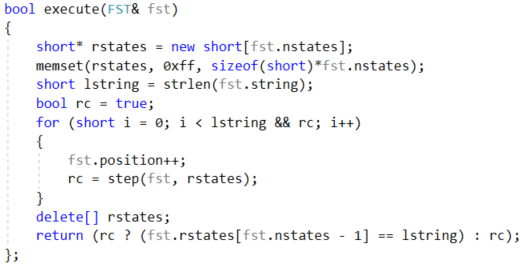


Рисунок 3.4 – Функция разбора входной цепочки на языке PMN-2019

* 1. Основные структуры данных

Основные структуры данных лексического анализатора: таблица токенов, таблица лексем и таблица идентификаторов.

На рисунке 3.5 приведена реализация структуры таблицы токенов:

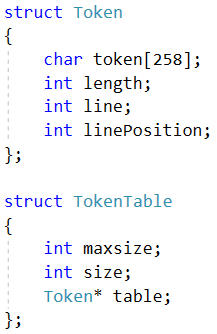


Рисунок 3.5 – Структура таблицы токенов языка PMN-2019

Структура TokenTable представляет таблицу токенов, где переменная maxsize хранит число равное максимальному размеру таблицы, size текущий размер таблицы, а table – указатель на строку таблицы.

Структура Token представляет строку таблицы TokenTable, где в массив token записывается само слово, переменная length хранит длину слова, line номер строки в исходном тексте программы, а linePosition – позицию в строке.

На рисунке 3.6 приведена реализация структуры таблицы лексем.

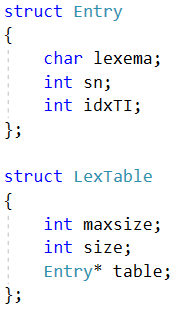


Рисунок 3.6 – Структура таблицы лексем языка PMN-2019

Структура LexTable представляет таблицу лексем, где переменная maxsize хранит число равное максимальному размеру таблицы, size текущий размер таблицы, а table – указатель на строку таблицы.

Структура Entry представляет строку таблицы LexTable, где переменная lexeme хранит лексему, sn номер строки в исходном тексте программы, а idxTI – номер в таблице идентификаторов.

На рисунке 3.7 приведена реализация структуры таблицы идентификаторов.

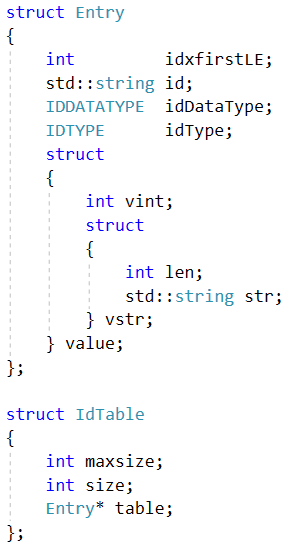


Рисунок 3.7 – Структура таблицы идентификаторов языка PMN-2019

Структура IdTable представляет таблицу идентификаторов, где переменная maxsize хранит число равное максимальному размеру таблицы, size текущий размер таблицы, а table – указатель на строку таблицы.

Структура Entry представляет строку таблицы IdTable, где переменная idxfirsttLE хранит индекс первого вхождения в таблицу лексем, id идентификатор, idDataType тип данных, idType тип идентификатора, vint целочисленное значение, len длину строку, а str – строку.

* 1. Структура и перечень сообщений лексического анализатора

При возникновении ошибки в лексическом анализаторе формируется ошибка в следующем формате: Номер ошибки, пояснительный текст, строка в исходном тексте, позиция в строке.

Перечень сообщений лексического анализатора приведен в таблице 3.2.

Таблица 3.2 – Перечень сообщений лексического анализатора языка PMN-2019

|  |  |
| --- | --- |
| Номер ошибки | Пояснительный текст |
| 110 | Недопустимый символ в исходном файле (-in:) |
| 111 | Превышена емкость таблицы лексем |
| 112 | Превышено количество строк в таблице лексем |
| 113 | В таблице лексем отсутствует строка с заданным номером |
| 114 | Превышена емкость таблицы идентификаторов |
| 115 | Превышено количество строк в таблице идентификаторов |
| 116 | В таблице идентификаторов отсутствует строка с заданным номером |
| 117 | Не удалось определить тип лексемы |
| 118 | Превышена емкость таблицы токенов |
| 119 | Превышено количество токенов в таблице токенов |
| 120 | Ошибка с разбиением исходного текста на токены |
| 121 | Ошибка с разбором строкового литерала |

* 1. Принцип обработки ошибок

При обнаружении ошибки в исходном коде программы лексический анализатор формирует сообщение об ошибки и выводит его в файл с протоколом работы, заданный параметром –log:.

* 1. Параметры лексического анализатора

Входным параметром лексического анализа является таблица, состоящая из структур, полями которых являются токен и номер его строки в исходном файле, полученные на этапе проверки исходного кода на допустимость символов.

* 1. Алгоритм лексического анализа

Алгоритм лексического анализа языка PMN-2019:

* Считываем исходный текст программы и делим его на токены, формируя структуру таблицы токенов;
* Слова из таблицы токенов пропускаем через графы, определяя тип лексем;
* Составляем таблицу лексем и идентификаторов.

Программный код, реализующий данный алгоритм, приведен на рисунке 3.8.



Рисунок 3.8 – Программный код, реализующий лексический анализ

* 1. Контрольный пример

Результат работы лексического анализатора, полученный при выполнении контрольного примера, а именно таблица лексем и таблица идентификаторов представлен в приложении.

1. Разработка синтаксического анализатора
   1. Структура синтаксического анализатора

Синтаксический анализ языка PMN-2019 выполняется после завершения работы лексического анализатора. Синтаксический анализатор предназначен для сопоставления последовательности лексем языка PMN-2019 c его формальной грамматикой.

Входные данные: таблица лексем и таблица идентификаторов.

Результат работы: Дерево разбора.

* 1. Контекстно-свободная грамматика, описывающая синтаксис языка