# **Введение**

Основной целью данной курсовой работы является разработка транслятора для языка программирования DVV-2018. Главной задачей для транслятора будет сделать язык DVV-2018 понятным для вычислительной машины. В данном курсовом проекте будет происходить трансляция кода на языке DVV-2018 в код на языке ассемблер.

Для данного курсового проекта были определены следующие задачи:

* разработка спецификации языка программирования;
* разработка структуры транслятора;
* разработка лексического анализатора;
* разработка синтаксического анализатора;
* разработка семантического анализатора;
* обработка выражений;
* генерация кода в Assebler;
* тестирование транслятора.

Решение для каждой из поставленных задач будут отображены в следующих главах курсового проекта:

* спецификация языка программирования;
* структура транслятора;
* лексический анализатор;
* синтаксический анализатор;
* семантический анализатор;
* преобразование выражений;
* генерация кода;
* тестирование транслятора.

Язык программирования DVV-2018 предназначается для выполнения простейших арифметический и логических действий , а также для операций над строками.

# **Глава №1 Спецификация языка программирования**

## **Характеристика языка программирования**

Язык DVV-2018 – это процедурный, универсальный, строготипизированный, компилируемый язык. Так же язык не является объектно-ориентированным.

* Универсальный — язык, система терминов, определенных строго и однозначно, а потому допускающих над собой чисто формальные операции.
* Процедурный – язык при программировании на котором последовательно выполняемые операторы можно собрать в подпрограммы, то есть более крупные целостные единицы кода, с помощью механизмов самого языка.
* Сильной(строгой) типизацией в языке является отсуствия приведения типов
* Компилируемый – язык который конвертируется в машинный код.
  1. **Алфавит языка**

Алфавит языка DVV-2018 основан на кодировке Windows-1251, представленной на рисунке 1.1. Исходный код DVV-2018 может содержать символы латинского алфавита, цифры десятичной системы счисления от 0 до 9, символы русского алфавита можно использовать только в строковых литералах. Код, написанный на языке DVV-2018, может содержать символы : +, -, \*, /, {, }, (, ), ', >, <, =, : .

## **Символы сепараторы**

Символы, которые являются сепараторами в языке DVV-2018 представлены в таблице 1.1.

Таблица 1.1 – Сепараторы

|  |  |
| --- | --- |
| Сепаратор | Назначение |
| (space)  (\n) | разделение инструкций |
| {  } | программный блок |
| (  ) | параметры/приоритетность операций (в выражениях) |

## **Применяемые кодировки**

Для написания исходного кода на языке программирования DVV-2018 используется кодировка Windows-1251. Кодировка Windows-1251 представлена на рисунке 1.1



Рисунок 1.1 – Алфавит входных символов

## **Типы данных**

В языке DVV-2018 есть 3 типа данных: целочисленный integer, строковый string и логический bool. Описание типов данных, предусмотренных данным языком представлены в таблице 1.2.

Таблица 1.2 – Типы данных языка DVV-2018

|  |  |
| --- | --- |
| Тип данных | Описание типа данных |
| integer | Фундаментальный тип данных. Предусмотрен для объявления целочисленных положительных данных.  Автоматически инициализируется нулевым значением.  Возможные операции:  + – суммирование;  - – вычитание;  \* – умножение;  / – деление;  = – присваивание значения; |
| String | Фундаментальный тип данных. Предусмотрен для объявления строк. Автоматическая инициализируется строкой нулевой длины. Максимальное количество символов в строке – 255. |
| Bool | Фундаментальный тип данных. Предусмотрен для использования в конструкции if() else. Может быть инициализирован 2 значениями truth или false. Автоматическая инициализация не предусмотрена.  Возможные операции:  == , < , > – логические операторы сравнения. |

## **Преобразование типов данных**

Так как язык DVV-2018 является строготипизированным , то по определению

сильной(строгой) типизации преобразование(приведение) типов в языке отсуствует.

## **Идентификаторы**

Для идентификатора можно использовать буквы латинского алфавита и цифры десятичной системы счисления от 0 до 9.Имя идентификатора может начинаться только с символа латинского алфавита. Максимальная длина идентификатора может быть не более 6 символов.

## **Литералы**

Для языка DVV-2018 характерно присуствие литералов 2 типов: литералы целого типа и строковые литералы. Описание данных литералов можно прочитать ниже в таблице 1.3.

Таблица 1.3 – Описание литералов

|  |  |
| --- | --- |
| Тип литерала | Описание |
| Литералы целого типа | Целочисленные неотрицательные литералы, инициализируются 0. Литералы только могут быть только rvalue. Диапазон изменения от 0 до 4294967296. |
| Строковые литералы | Символы, заключённые в " " (двойные кавычки), инициализируются пустой строкой, строковые переменные. Литералы могут быть только rvalue. |
| Логические литералы | Логические литералы могут быть инициализированы только двумя значениями truth или false. |

## **Область видимости идентификаторов**

Область видимости «сверху вниз». В языке DVV-2018 требуется обязательное объявление переменной перед её использованием. Все переменные должны находиться внутри функции. Имеется возможность объявления переменных c одинаковым идентификаторами в разных функциях. Каждая переменная получает постфикс – название функции, в которой она объявлена.

## **Инициализация данных**

|  |  |
| --- | --- |
| Вид инициализации | Примечание |
| <тип данных><идентификатор>; | Автоматическая инициализация: переменные типа integer инициализируются нулём, переменные типа string – пустой строкой, переменные типа bool не имеют автоматической инициализации. |
| <идентификатор> : <значение>; | Присваивание переменной значения. |

## **Инструкции языка**

Все возможные инструкции языка программирования DVV-2018 представлены в общем виде в таблице 1.4.

Таблица 1.4 – Инструкции языка программирования DVV-2018

|  |  |
| --- | --- |
| Инструкция | Запись на языке DVV-2018 |
| Объявление переменной | <тип данных> <идентификатор>; |
| Присваивание | <идентификатор> : <значение>/<идентификатор>; |
| Объявление внешней функции | function <тип данных> <идентификатор> (<тип данных> <идентификатор>, …) { … } |
| Блок инструкций | { … } |
| Вывод данных | print (<идентификатор> / <литерал>); |

## **Операции языка**

Язык программирования DVV-2018 может выполнять арифметические операции, представленные в таблице 1.5.

Таблица 1.5 – Приоритетности операций языка программирования DVV-2018

|  |  |
| --- | --- |
| Операция | Приоритетность операции |
| (  ) | 0 |
| , | 1 |
| \*  / | 3 |
| +  - | 2 |
| = , > , < | Данные операции не имеют приоритета |

Максимальным значением приоритетности является “3”, минимальным “0” соответственно.

## **Выражения и их вычисления**

В выражения круглые скобки используются для изменения приоритета операций. Выражения могут содержать вызов функции. Допускаются выражения которые вычисляют значение из операндов. Вычисление в выражения происходит по приоритетности операций языка

## **Программные конструкции языка**

Ключевые программные конструкции языка программирования DVV-2018 представлены в таблице 1.6.

Таблица 1.6 – Программные конструкции языка DVV-2018

|  |  |
| --- | --- |
| Главная функция (точка входа в приложение) | funс main  {  …  } |
| Функция | funс <тип> <идентификатор> (<тип> <идентификатор>, …)  {  …  } |
| Блок if() else | if(проверяемое условие) {…}  else {…} |

## **Область видимости**

В языке DVV-2018 переменные находятся внутри функций . Каждая функция является отдельной областью видимости. Объявление глобальных переменных, то есть переменных которые доступны во всей программе, не предусмотрено. Объявление пользовательских областей видимости не предусмотрено.

## **Семантические проверки**

Таблица с перечнем семантических проверок, предусмотренных языком, приведена в таблице 1.7.

Таблица 1.7 – Семантические проверки

|  |  |
| --- | --- |
| Номер | Правило |
| 0 | Идентификаторы функций не должны повторяться |
| 1 | Тип данных значений которые передаются в функцию должны совпадать с типом параметров при объявлении этого значения. |
| 2 | Тип данных передаваемых значений в функцию стандартной библиотеки должен соответствовать заявленному. |
| 3 | Объявление идентификатора должно происходить до его использовния. |
| 4 | В арифметических выражениях операнды должны иметь один тип данных. |

## **Распределение оперативной памяти на этапе выполнения**

Размещение всех переменных происходит в куче. Распределение оперативной памяти происходит на этапе генерации. Таблица лексем и таблица идентификаторов сохраняются в структуры с выделенной динамической памятью, которая очищается по окончанию работы транслятора.

## **Стандартная библиотека и её состав**

Функции стандартной библиотеки с описанием представлены в таблице 1.8. Стандартная библиотека написана на языке программирования C++. Способом подключения библиотеки является ключевое слово incude.

Таблица 1.8 – Состав стандартной библиотеки

|  |  |  |  |
| --- | --- | --- | --- |
| Имя функции | Возвращаемое значение | Принимаемые параметры | Описание |
| Strlen | Integer | str s - строка | Возвращает длину строки s |
| Substr | String | string x – строка,  int y – число | Возвращает строку которая была выделена из основной строки начиная с символа с номером y |

## **Ввод и вывод данных**

Ввод данных не предусмотрен в языке DVV-2018.А за вывод данных отвечает функция: print (<идентификатор или литерал>);

## **Точка входа**

Точкой входа в языке DVV-2018 является - главная функция main.

## **Препроцессор**

Препроцессор для языка DVV-2018 не предусмотрен.

## **Соглашения о вызовах**

В языке вызов функций происходит по соглашению о вызовах stdcall. Выбор пал на это соглашение о вызовах так как это stdcall является самым популярным соглашением о вызовах, в C подобных языках.

Особенности stdcall:

* параметры передаются через стек;
* запись в стек идет справа налево
* очистку стека производит вызываемый код

## **Объектный код**

Код на языке DVV-2018 транслируется в код на языке ассемблера.

## **Классификация сообщений транслятора**

При возникновении ошибки в коде программы на языке DVV-2018 и выявления её транслятором в файл протокола выводится сообщение. Классификация сообщений приведена в таблице 1.9.

Таблица 1.9 – Классификация сообщений транслятора

|  |  |
| --- | --- |
| Интервал | Описание ошибок |
| 0-99 | Системные ошибки |
| 100-109 | Ошибки параметров |
| 110-112 | Ошибки открытия и чтения файлов |
| 113-129 | Ошибки лексического анализа |
| 600-699 | Ошибки синтаксического анализа |
| 700-799 | Ошибки семантического анализа |

## **Контрольный пример**

**Глава №2 Структура транслятора**

**2.1 Компоненты транслятора их назначение и принципы взаимодействия**

Транслятор – обслуживающая программа, преобразующая исходную программу, предоставленную на входном языке программирования, в рабочую программу, представленную на объектном языке.

Компоненты транслятора:

* Лексический анализатор

1. Выполняет аналитический разбор входной последовательности символов на распознанные группы — лексемы.
2. Входным параметром является исходный код.
3. Выходными параметрами являются таблица лексем и таблица идентификаторов.

* Синтаксический анализатор

1. Выполняет сопоставление линейной последовательности лексем естественного или формального языка с его формальной грамматикой.
2. Входным параметром является таблица лексем.
3. Выходным параметром является дерево разбора или иначе говоря синтаксическое дерево

* Семантический анализатор

1. Выполняет последовательности действий алгоритма автоматического понимания текстов, заключающегося в выделении семантических отношений, формировании семантического представления текстов.
2. Входными параметрами являются таблица идентификаторов и дерево разбора.
3. Выходными параметрами является семантическая структура

* Генератор объектного кода

1. Занимается построением для программы на входном языке эквивалентной машинной программы.
2. Входным параметром является промежуточное представление программы.
3. Выходным параметром является объектный код.

Подробное описание компонентов транслятора можно найти в главах курсового проекта под названием:

* Лексический анализатор
* Синтаксический анализатор
* Семантический анализатор
* Генерация кода.

**2.2 Перечень входных параметров транслятора**

Перечень входных параметров для языка DVV-2018 представлен в таблице 2.2

Таблица 2.2

|  |  |
| --- | --- |
| Имя входного параметра | Выполняемая функция |
| -in | Если перед файлом указан этот параметр, то он является входным файлом. |
| -log | Если перед файлом указан этот параметр, то он является файлом протокола. |

**2.3 Протоколы, формируемые транслятором**

Для языка DVV-2018 предусмотрено формирование только одного протокола.

Данный файл будет иметь следующую запись: **имяфайла.txt.log.**  В файл протокола будут выводиться:

* Ошибки, если они есть. Иначе будет выводится, что тест выполнен без ошибок.
* Дата, когда был создан этот протокол.
* Перечень входных параметров для языка DVV-2018
* Таблица лексем
* Таблица идентификаторов
* Исходные данные: Количество символов, Количество строк.