Курсовая работа

РАЗРАБОТКА КОМПИЛЯТОРА ПОДМНОЖЕСТВА ПРОЦЕДУРНОГО ЯЗЫКА

**Руководство программиста**

**643.02068048.00001-01 33 01**

**Листов 9**

2024

АННОТАЦИЯ

Данное руководство предназначено для программистов, которые будут использовать разработанный компилятор процедурного языка в байт-код для JVM.

СОДЕРЖАНИЕ

[1. Назначение и условия применения программы 4](#_Toc135317028)

[2. Характеристики программы 5](#_Toc135317029)

[3. Обращение к программе 6](#_Toc135317030)

[4. Входные и выходные данные 7](#_Toc135317031)

1. Назначение и условия применения программы

Программа, описываемая в данном руководстве, является компилятором процедурного языка, который преобразует исходный код на процедурном языке в байт-код для JVM Разработанный компилятор предназначен для использования в целях разработки программного обеспечения, требующего компиляции и выполнения на целевой платформе, на которой установлена JVM версии 1.8.

* Для успешного использования компилятора необходимы следующие условия:
* Компьютер с операционной системой, на которую можно установить JVM версии 1.8.
* Установленная JVM версии 1.8 и средства для ее запуска (например, командная строка).
* Наличие исходного кода программы, написанной на процедурном языке, которую требуется скомпилировать в байт-код для JVM.

1. Характеристики программы

Ниже приведены основные характеристики разработанного компилятора:

* Поддерживаемый язык: Компилятор поддерживает процедурный язык программирования, который включает в себя возможности для структурирования кода с использованием процедур, условных операторов, циклов, операций ввода-вывода и других основных конструкций.
* Интеграция с JVM: JVM позволяет запустить файл расширения .class.
* Лексический и синтаксический анализ: Компилятор осуществляет лексический анализ исходного кода для выделения лексем и проверки их корректности согласно заданной грамматике. Затем производится синтаксический анализ, который определяет структуру программы и ее синтаксическую правильность.
* Семантический анализ: Компилятор выполняет семантический анализ, который проверяет соответствие типов, определяет области видимости и связывает идентификаторы с их объявлениями. Это позволяет выявить ошибки и несоответствия в программе на ранних этапах компиляции.
* Генерация байт-кода для JVM: После успешного анализа исходного кода, компилятор генерирует соответствующий байт-код, который может быть выполнен средствами JVM.

1. Обращение к программе

Для использования программы необходимо выполнить следующие шаги:

* Разместить исходный код программы в папке с программой в любом файле расширения .w. Убедится, что исходный код соответствует правилам и синтаксису процедурного языка, поддерживаемого компилятором.
* Запустить файл the\_compiler.jar, передав в качестве аргумента полный путь к файлу с исходным кодом программы. В ходе выполнения этого файла последовательно вызываются методы из других файлов для выполнения лексического и синтаксического анализа, семантического анализа и генерации кода.
* Если в процессе компиляции не возникло ошибок, результат компиляции будет доступен после того, как WLang соберет байт-код в файл (.class).

В случае возникновения ошибок в процессе компиляции, программа выведет соответствующие сообщения об ошибках, указывающие на проблемные места в исходном коде. Следует внимательно изучить эти сообщения, исправить ошибки и повторить процесс компиляции.

Разработанный компилятор написан на языке Java 1.8 с использование библиотек ASM ,а также antlr4.

1. Входные и выходные данные

Входные данные для компиляции передаются в виде исходного кода программы, который должен быть размещен в файле расширения .w (рис. 1). Данный файл может находиться где угодно в рамках одного компьютера.

Исходный код должен соответствовать синтаксису и правилам языка, для которого разработан компилятор. Входные данные должны содержать корректные конструкции языка и быть свободными от синтаксических ошибок.

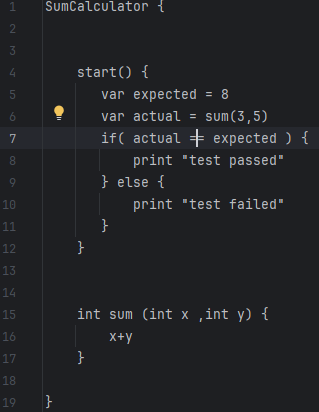


Рисунок 1 – Пример входных данных

После успешной компиляции исходного кода, результат будет сохранен в файле “Название программы.class” (рис. 2).

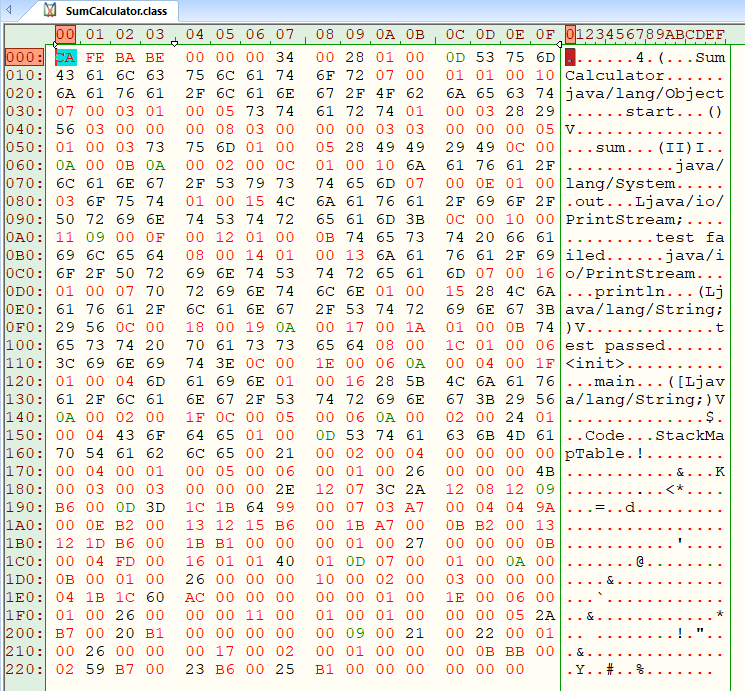


Рисунок 2 – Пример выходных данных

При компиляции исходного кода, в случае возникновения ошибок, программа будет выдавать соответствующие сообщения об ошибках (рис. 3).

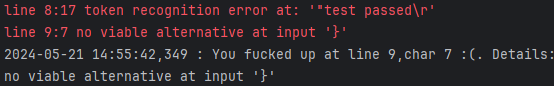


Рисунок 3 – Пример сообщений об ошибках



|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц) в докум. | № докумен-та | Входящий № сопроводит. докум. и дата | Подп. | Дата |
| Изм. | изменен-ных | заменен-ных | новых | аннули-рованных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |