Курсовая работа

РАЗРАБОТКА КОМПИЛЯТОРА ПОДМНОЖЕСТВА ПРОЦЕДУРНОГО ЯЗЫКА

**Программа и методика испытаний**

**643.02068048.00001-01 51 01**

**Листов 9**

2023

АННОТАЦИЯ

Документ "Программа и методика испытаний" описывает процесс испытаний разработанного компилятора процедурного языка, предназначенного для преобразования исходного кода в код для ассемблера.

СОДЕРЖАНИЕ

[1. Объект испытаний 4](#_Toc135314113)

[2. Цель испытаний и требования к программе 5](#_Toc135314114)

[3. Средства и порядок испытаний 6](#_Toc135314115)

[4. Пример проведения испытаний 7](#_Toc135314116)

1. Объект испытаний

Объект испытаний включает в себя компилятор процедурного языка, разработанный для преобразования исходного кода на процедурном языке в код для Clang.

Компилятор процедурного языка состоит из следующих компонентов:

1. Лексический и синтаксический анализатор (ReviLangListener.py).
2. Абстрактное синтаксическое дерево (AST) (reviLang.g).
3. Таблица символов (AST.py).
4. Генератор кода LLVM (main.py).
5. Цель испытаний и требования к программе

Цель испытаний заключается в проверке работоспособности компилятора процедурного языка и правильности преобразования исходного кода в код для Clang. Основные аспекты, подлежащие проверке, включают:

1. Корректность лексического и синтаксического анализа.
2. Построение абстрактного синтаксического дерева.
3. Генерация кода LLVM.
4. Обработка ошибок.
5. Средства и порядок испытаний

Для проведения испытаний компилятора процедурного языка в код для Clang будут использованы следующие средства:

* Компьютер с установленной операционной системой, совместимой с выбранной целевой платформой для исполнения скомпилированного кода.
* Редактор исходного кода: программное обеспечение, позволяющее создавать и редактировать исходный код на процедурном языке.
* Компилятор процедурного языка: разработанный компилятор, способный преобразовывать исходный код на процедурном языке в код для Clang
* Clang — это транслятор для C-подобных языков, созданный специально для работы на базе LLVM.

Испытания компилятора процедурного языка в код для Clang будут проводиться в следующем порядке:

1. Подготовка тестовых программ
2. Запуск компилятора
3. Проверка результатов компиляции
4. Исполнение скомпилированного кода
5. Пример проведения испытаний

В качестве исходного кода (рис. 1) был взят код с функцией для вычисления синуса.

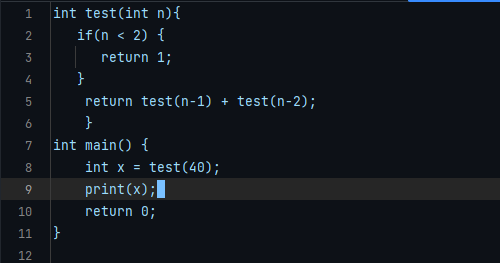


Рисунок 1 – Исходный код программы в файле «test.revi»

Результатом лексического и синтаксического анализа будет дерево сгенерированное Antlr4 (рис. 2).

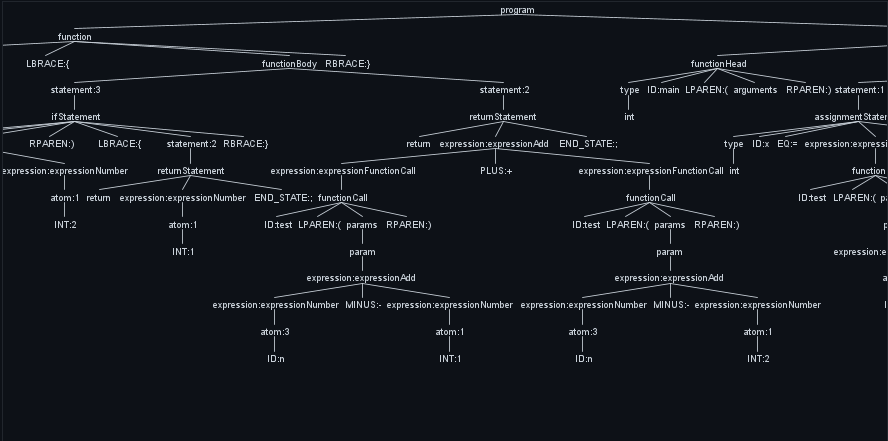


Рисунок 2

После завершения работы всей программы в файле «revi.s» будет находиться целевой код (рис. 3).



Рисунок 3

Если же в коде были допущены различные ошибки, то компилятор должен сообщить о них.

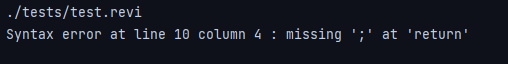


Рисунок 4– Пример сообщения об ошибке.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |