РЕФЕРАТ

Расчетно-пояснительная записка с., рис., табл., источн.
Ключевые слова: .

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Компилятор — это программная система, которая преобразует код, написанный на языке программирования, в форму, пригодную для выполнения на компьютере [1].

Современный мир зависит от языков программирования, поскольку все программное обеспечение на компьютерах написано на том или ином языке, и компиляторы играют ключевую роль в этом процессе [1].

Целью данной работы является разработка компилятора для языка SmallTalk. Компилятор должен выполнять чтение текстового файла, содержащего код на языке SmallTalk и генерировать на выходе LLVM IR программы, пригодный для запуска.

Для достижения поставленной цели необходимо решить следующие **задачи**:

1. проанализировать грамматику языка SmallTalk;
2. изучить существующие средства для анализа исходного кода программы, системы генерации низкоуровневого кода;
3. реализовать прототип компиляторы;
4. провести тестирование компилятора.

1 Аналитический раздел

Компилятор — это программа, которая считывает текст программы, написанной на одном языке — исходном, и транслирует (переводит) его в эквивалентный текст на другом языке — целевом. Одна из важных ролей компилятора состоит в сообщении об ошибках в исходной программе, обнаруженных в процессе трансляции [1].

1.1 Структура компилятора

Конструктивно компилятор состоит из [2, 3]:

— фронтенда (compiler frontend), который занимается построением промежуточного представления из исходного кода и состоит из:
  — препроцессора;
  — лексического, синтаксического и семантического анализаторов;
  — генератора промежуточного представления;
— мидленд (middle-end), включающий в себя различные оптимизации;
— бэкенда (compiler backend), который занимается кодогенерацией.

На рисунке 1.1 представлена схема концептуальной структуры компилятора.

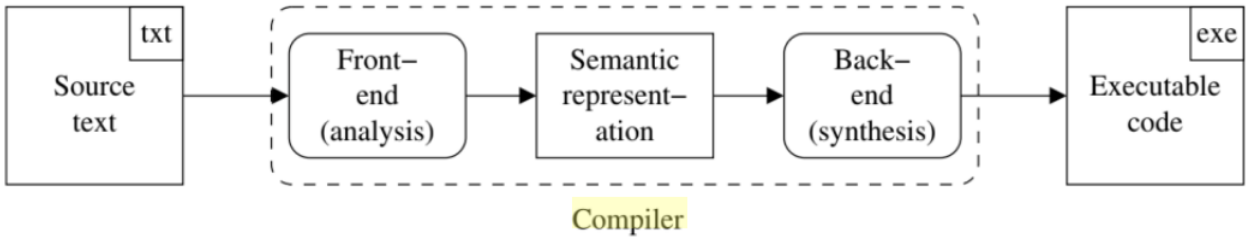

Рисунок 1.1 – Концептуальная структура компилятора

Рассмотрим работу компилятора по фазам [4]. Обобщенная структура компилятора и основные фазы компиляции показаны на рисунке 1.2.

Рисунок 1.2 – Обобщенная структура и фазы компиляции

1.1.1 Препроцессор

Иногда сборка поручается программе, который выполняет предварительную обработку перед фазой фронтенда компилятора.

Препроцессор может [1, 2]:

1. раскрывать макросы в инструкции исходного языка;

2. обрабатывать включение файлов;
3. обрабатывать языковые расширения.

1.2 Лексический анализ

На фазе лексического анализа входная программа, представляющая собой поток литер, разбивается на лексемы — слова в соответствии с определениями языка. Основными формализмами, лежащими в основе реализации лексических анализаторов, являются конечные автоматы и регулярные выражения [4].

Лексический анализатор может работать в двух основных режимах [4]:

1. как подпрограмма, вызываемая синтаксическим анализатором для получения очередной лексемы;
2. как полный проход, результатом которого является файл лексем.

В процессе выделения лексем лексический анализатор может [4]:

— самостоятельно строить таблицы объектов (идентификаторов, строк, чисел и т.д.);
— выдавать значения для каждой лексемы при обращении к ней, в этом случае таблицы объектов строятся на последующих фазах (например, при синтаксическом анализе).

На этапе лексического анализа обнаруживаются простейшие ошибки [4]:

— недопустимые символы;
— неправильная запись чисел;
— ошибки в идентификаторах.

2 Конструкторский раздел

3 Технологический раздел

ЗАКЛЮЧЕНИЕ

В ходе данной работы была достигнута цель: разработан компилятора язык SmallTalk, который выполняет чтение текстового файла, содержащего код на языке SmallTalk и генерирует на выходе LLVM IR программы, пригодный для запуска.

Были решены все задачи:

1. проанализирована грамматика языка SmallTalk;
2. изучены существующие средства для анализа исходного кода программы, системы генерации низкоуровневого кода;
3. реализован прототип компилятора;
4. проведено тестирование компилятора.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Компиляторы / *Альфред Ахо, Моника С Лам, Рави Сети [и др.]* // Принципы, технологии, инструментарий. 2003.

2. *Владимиров Константин.* Оптимизирующие компиляторы. Структура и алгоритмы. Litres, 2024.

3. Modern compiler design / *Dick Grune, Kees Van Reeuwijk, Henri E Bal [и др.].* Springer Science & Business Media, 2012.

4. *Серебряков ВА, Галочкин МП.* Основы конструирования компиляторов // М.: Эдиториал УРСС. 2001. Т. 221, № 1.

ПРИЛОЖЕНИЕ А
Настройка ClickHouse

В листинге А.1 представлена настройка одной связи между топиком Kafka и таблицей в ClickHouse.

Листинг А.1 – Настройка связи между Kafka и ClickHouse

```
create table `kafka-tfidf`
(
    id_user  UInt64,
    id_movie UInt64,
    cosine   Float64,
    time     String
)
engine = Kafka('kafka1:9091', 'tf-idf-recommendations', 'group',
    'JSONEachRow');

create table `tfidf-recommendations`
(
    id_user  UInt64,
    id_movie UInt64,
    cosine   Float64,
    time     DateTime('Europe/Moscow')
)
engine = MergeTree ORDER BY (id_user, id_movie)
SETTINGS index_granularity = 8192;

CREATE MATERIALIZED VIEW WatchBoxRecommender.consumer
TO WatchBoxRecommender.`tfidf-recommendations`
(
    `id_user` UInt64,
    `id_movie` UInt64,
    `cosine` Float64,
    `time` DateTime('Europe/Moscow')
)
AS
SELECT id_user,
               id_movie,
               cosine,
               parseDateTimeBestEffort(time, 'Europe/Moscow') AS time
FROM WatchBoxRecommender.`kafka-tfidf`;
```