МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

|  |  |
| --- | --- |
| Институт (факультет) | Информационных технологий |
| Кафедра | Математическое и программное обеспечение ЭВМ |

КУРСОВАЯ РАБОТА

|  |
| --- |
| по дисциплине Теория автоматов и формальных языков программирования |

|  |  |
| --- | --- |
| на тему | Разработка алгоритмического обеспечения и построение |
| лексического анализатора компилятора | |

|  |
| --- |
| Выполнил студент группы |
| 1ПИб-02-1оп-22 |
| направления подготовки (специальности) |
| 09.03.04, Программная инженерия |
| шифр, наименование |
|  |
| Харламов Денис Алексеевич |
| фамилия, имя, отчество |

|  |
| --- |
| Руководитель |
| Ганичева Оксана Георгиевна |
| фамилия, имя, отчество |
| Доцент |
| должность  Пышницкий Константин Михайлович |
| фамилия, имя, отчество |
| Старший Преподаватель |
| должность |

|  |
| --- |
| Дата представления работы |
| «\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20 г. |
|  |
| Заключение о допуске к защите |
|  |
|  |
|  |
|  |
| Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_,\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| количество баллов |
| Подпись преподавателя\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |

Аннотация

В данной курсовой работе рассматривается тема разработка алгоритмического обеспечения и построение лексического анализатора компилятора. В работе описывается процесс проектирования и реализации программы. Основной целью работы является разработка приложения, которое будет принимать код некоторой программы и обрабатывать его подобно компилятору. Результат работы может быть использован в образовательных целях.

Оглавление

[1. Введение 4](#_Toc166252428)

[2. Описание предметной области 5](#_Toc166252429)

[2.1. Лексический анализатор 5](#_Toc166252430)

[2.2 Синтаксический анализатор 8](#_Toc166252431)

[3. Основная часть 10](#_Toc166252432)

# Введение

В современном мире информационных технологий разработка программного обеспечения играет ключевую роль. Одним из важнейших этапов создания любого программного продукта является процесс компиляции, который преобразует исходный код программы в машинный код, понятный компьютеру.

Работа компилятора делится на следующие этапы:

1. Лексический анализ.

2. Синтаксический анализ.

3. Семантический анализ.

4. Промежуточное представление.

5. Оптимизация кода.

6. Генерация кода.

7. Таблицы символов.

В основе этого процесса лежит лексический анализ – первый этап анализа исходного кода, который отвечает за разбиение текста программы на отдельные лексемы и определение их типа.

Цель данной курсовой работы – рассмотреть создание компилятора лексического анализатора, разбитого на несколько этапов. Изучение данной темы позволит улучшить свои познания в работе основных принципов компилятора, его важность в разработке программного обеспечения и потенциал для оптимизации процесса компиляции.

1. Описание предметной области

В этой части будут рассмотрены определения, назначения и функции лексического и синтаксического анализов.

# 2.1. Лексический анализатор

Лексемы – это минимальные значимые единицы языка программирования, такие как идентификаторы, ключевые слова, операторы, числа, строки и т.п.

Лексический анализ, также известный как сканирование, - это первый этап работы компилятора, который отвечает за разбиение исходного кода программы на отдельные лексемы и определение их типа. На вход лексического анализатора поступает текст исходной программы. Результатом работы лексического анализатора является перечень всех найденных в тексте исходной программы лексем. Эти данные можно представить в виде таблицы, называемой таблицей лексем. Каждой лексеме в таблице лексем будет соответствовать некий уникальный код, зависящий от типа лексемы, и дополнительной информации о ней.

Далее более подробно разберем процесс работы лексического анализатора.

Процесс лексического анализа:

1. Сканирование исходного кода: Лексический анализатор последовательно считывает символы исходного кода программы.

2. Формирование лексем: Анализатор группирует символы в лексемы согласно правилам языка программирования.

3. Определение типа лексемы: После формирования лексемы анализатор определяет её тип. Например, "if" будет определен как ключевое слово, "123" - как целое число, а "abc123" - как идентификатор.

4. Передача информации: Лексический анализатор передает информацию о каждой распознанной лексеме следующему этапу компиляции - синтаксическому анализу.

С теоретической точки зрения лексический анализатор не является обязательной, необходимой частью компилятора. Так как, его функции могут выполняться на этапе синтаксического разбора. Использование же лексического анализатора важно из-за следующих причин:

1) Применение лексического анализатора сокращает объём информации поступающий на обработку следующему этапу компиляции.

2) Качественный лексический анализатор позволяет ускорить процесс компиляции за счет быстрого распознавания лексем.

3) Лексический анализатор улучшает гибкость компиляции. Лексический анализатор отделяет сложный синтаксический анализатор от работы с самим текстом исходной программы. Структура синтаксического анализатора значительно сложнее и может быть разной в зависимости от вычислительной системы. Значит такая конструкция компилятора позволяет для перехода на другую вычислительную систему поменять только лексический анализатор.

4) Обнаружение ошибок на этапе лексического анализа помогает избежать ошибок на последующих этапах и повысить качество итогового кода.

Основные задачи лексического анализатора:

1. Удаление пробелов и комментариев: Лексический анализатор игнорирует пробелы и комментарии, которые не несут смысловой нагрузки.

2. Выделение лексем и создание их классификации.

2. Обработка ошибок: Анализатор должен уметь обнаруживать и сообщать об ошибках в исходном коде, например, о недопустимых символах или неправильно сформированных лексемах.

Построение эффективного лексического анализатора основывается на ряде принципов, которые помогают обеспечить корректное и быстрое распознавание лексем в исходном коде программы.

Этап первый выбор модели и языка программирования.

Лексический анализатор может быть реализован на различных языках программирования, который поддерживает возможности ввода-вывода, позволяющие считывать входные данные;

Конечные автоматы. Это распространенный подход к построению лексических анализаторов. КА представляют собой абстрактные модели, которые могут находиться в конечном числе состояний и переходить между ними в зависимости от входных символов. Для каждой лексемы или группы лексем строится отдельный КА.

Регулярные выражения. Они используются для описания шаблонов лексем. Регулярные выражения позволяют компактно задать правила формирования лексем с помощью специальных символов и метасимволов. В этом случае лексический анализатор строится на основе преобразования регулярных выражений в конечные автоматы.

Этап второй определение структуры лексем и их классификация:

Анализ алфавита языка: Необходимо определить набор допустимых символов, из которых могут состоять лексемы.

Выделение классов лексем и описание структуры каждой категории лексем: Лексемы группируются по типам, например, идентификаторы, ключевые слова, операторы, числа, строки и т.п., а так же для каждой категории определяются правила формирования лексем, например, с помощью регулярных выражений.

Этап третий обработка ошибок.

Лексический анализатор должен уметь обнаруживать ошибки в исходном коде, например, недопустимые символы или неправильно сформированные лексемы. После чего анализатор должен предоставлять информативные сообщения об ошибках, указывающие на место и тип ошибки.

Построение лексического анализатора – это сложная задача, требующая глубокого понимания теории языков программирования и методов их анализа. Однако, при правильном выборе модели и использовании эффективных алгоритмов можно создать качественный анализатор, который станет основой для успешной работы компилятора.

# 2.2 Синтаксический анализатор

Основной этап анализа в компиляторе – это синтаксический разбор. Без выполнения этого этапа работа компилятора не имеет смысла, в то время как лексический разбор можно пропустить. Если лексический анализатор позволяет облегчить сложный по структуре синтаксический анализатор от выполнения примитивных задач по выделению и запоминанию лексем входной программы, то во время синтаксического разбора могут быть решены все вопросы по проверке синтаксиса входного языка.

Синтаксический анализ, также известный как разбор, - это второй этап работы компилятора, который следует за лексическим анализом. Синтаксический анализатор принимает на вход таблицу лексем (или цепочку лексем), которую получает от лексического анализатора, и исследует только тип каждой лексемы. Информация о других характеристиках лексем используется в более поздних этапах компиляции, таких как семантический анализ, подготовка и генерация кода для итоговой программы.

Основная задача синтаксического анализатора - проверить соответствие последовательности лексем грамматическим правилам языка программирования и построить дерево разбора, представляющее синтаксическую структуру программы.

Синтаксический анализатор определяет, соответствует ли структура программы правилам языка. Это позволяет выявить синтаксические ошибки, такие как пропущенные скобки, неправильный порядок операторов или несогласованные типы данных.

Дерево разбора – это структура данных, которая отражает иерархию и взаимосвязи элементов программы. Дерево разбора является основой для последующих этапов компиляции, таких как семантический анализ и генерация кода.

Этапы синтаксического анализа:

Получение лексем. Данные, полученные от лексического анализатора, передаются на вход синтаксическому анализатору, который анализирует их согласно грамматике входного языка.

Проверка грамматических правил. В основе синтаксического анализатора лежит распознаватель текста входной программы на основе грамматики входного языка. Как правило, синтаксические конструкции языков программирования могут быть описаны с помощью КС-грамматик, реже встречаются языки, которые могут быть описаны с помощью регулярных грамматик. Чаще всего регулярные грамматики применимы к языкам ассемблера, а языки высокого уровня построены на основе синтаксиса КС-языков. Распознаватель дает ответ на вопрос о том, принадлежит или нет цепочка входных символов заданному языку – это основная задача синтаксического анализатора. Кроме этого синтаксический анализатор должен иметь некий выходной язык, с помощью которого он передает следующим фазам компиляции всю информацию о найденных и разобранных синтаксических структурах. Анализатор использует формальную грамматику языка программирования для проверки соответствия последовательности лексем заданным правилам.

Построение дерева разбора. По мере проверки грамматических правил, анализатор строит дерево разбора, которое отражает синтаксическую структуру программы.

Обработка ошибок. Синтаксический анализатор должен уметь обнаруживать и сообщать о синтаксических ошибках в исходном коде, например, о пропущенных скобках или несогласованных типах данных.

Синтаксический анализ является важной частью процесса компиляции, которая обеспечивает проверку корректности программы и построение структуры данных, необходимой для последующих этапов анализа и генерации кода.

Синтаксический анализ является важной частью процесса компиляции, которая обеспечивает проверку корректности программы и построение структуры данных, необходимой для последующих этапов анализа и генерации кода.

# 3. Основная часть