Всем здравствуйте/добрый день Меня зовут Борисов Антон студен 4 группы мой язые BAA-2019

Начнем с этапов разработки. В процессе создания транслятора я прошел через 7 этапов они представлены на слайде…

* Язык BAA-2019 – это процедурный, строго типизированный, транслируемый в ассемблер язык, не имеющий элементов ООП. Используется кодировка ASCII
* В языке есть 3 типа данных: знаковый целочисленный, строковый и символьный. Преобразования не допускаются.
* С знаковым целочисленным типом данных можно выполнять операции деления, сложения, умножения, остаток от деления, операторы сравнения больше меньше равно не равно, а также поддерживается сокращенная запись операций с арифметическими операторами
* Со строковым и символьными типами данных можно использовать операторы сравнения равно не равно, можно узнать размер строк
* В выражениях могут присутствовать отрицательные литералы, отрицательные идентификаторы, а также вызов функций (которые возвращают значение)
* Также язык поддерживает вызов функций процедур циклов условных выражений рекурсии и возможность записи комментариев

Содержимое стандартной библиотеки представлено на слайде

На ней присутсвует функция рандома , вычисления длины строки, возведение в степень , систем паузе , так же в библиотеки присутсвуют функции вывода строки в консоль и в консоль с переводом строки.

ЛЕКСИЧЕСКИЙ АНАЛИЗАТОР:

Первая стадия работы компилятора называется лексическим анализом, а программа, её реализующая, – лексическим анализатором (сканером). На вход лексического анализатора подаётся исходный код входного языка. Лексический анализатор выделяет в этой последовательности простейшие конструкции языка,. производит предварительный разбор текста, преобразующий единый массив текстовых символов в массив токенов.

Примеры лексических единиц: идентификаторы, числа, символы операций, служебные слова и т.д. Лексический анализатор преобразует исходный текст, заменяя лексические единицы их внутренним представлением – лексемами, для создания промежуточного представления исходной программы. Каждой лексеме сопоставляется ее тип и запись в таблице идентификаторов, в которой хранится дополнительная информация.

Функции лексического анализатора:

− удаление «пустых» символов и комментариев. Если «пустые» символы (пробелы, знаки табуляции и перехода на новую строку) и комментарии будут удалены лексическим анализатором, синтаксический анализатор никогда не столкнется с ними (альтернативный способ, состоящий в модификации грамматики для включения «пустых» символов и комментариев в синтаксис, достаточно сложен для реализации);

На слайде предоставлен кусочек контрольного примера и соотвествующий ему массив токенов

СИНТАКСИЧЕСКИЙ АНАЛИЗАТОР:

* Синтаксический анализатор – часть компилятора, выполняющая синтаксический анализ, то есть проверку исходного кода на соответствие правилам грамматики. Входной информацией для синтаксического анализа является таблица лексем и таблица идентификаторов. Выходной информацией является дерево разбора

В синтаксическом анализаторе транслятора языка BAA-2019 используется контекстно-свободная грамматика , где

T – множество терминальных символов (было описано в разделе 1.2 данной пояснительной записки),

N – множество нетерминальных символов (первый столбец таблицы 4.1),

P – множество правил языка (второй столбец таблицы 4.1),

S – начальный символ грамматики, являющийся нетерминалом.

Эта грамматика имеет нормальную форму Грейбах, т.к. она не леворекурсивная (не содержит леворекурсивных правил) и правила  имеют вид:

1. , где ; (или , или );
2. , где — начальный символ, при этом если такое правило существует, то нетерминал  не встречается в правой части правил.

Описание нетерминальных символов содержится в таблице 4.1.

Конечный автомат с магазинной памятью представляет собой семерку. Подробное описание компонентов магазинного автомата представлено в таблице 4.2.

|  |  |  |
| --- | --- | --- |
| Компонента | Определение | Описание |
|  | Множество состояний автомата | Состояние автомата представляет из себя структуру, содержащую позицию на входной ленте, номера текущего правила и цепочки и стек автомата |
|  | Алфавит входных символов | Алфавит представляет из себя множества терминальных и нетерминальных символов, описание которых содержится в таблица 3.1 и 4.1. |
|  | Алфавит специальных магазинных символов | Алфавит магазинных символов содержит стартовый символ и маркер дна стека (представляет из себя символ $) |
|  | Функция переходов автомата | Функция представляет из себя множество правил грамматики, описанных в таблице 4.1. |
|  | Начальное состояние автомата | Состояние, которое приобретает автомат в начале своей работы. Представляется в виде стартового правила грамматики |
|  | Начальное состояние магазина автомата | Символ маркера дна стека $ |
|  | Множество конечных состояний | Конечные состояние заставляют автомат прекратить свою работу. Конечным состоянием является пустой магазин автомата и совпадение позиции на входной ленте автомата с размером ленты |

**Принцип работы автомата следующий**:

1. В магазин записывается стартовый символ;
2. На основе полученных ранее таблиц формируется входная лента;
3. Запускается автомат;
4. Выбирается цепочка, соответствующая нетерминальному символу, записывается в магазин в обратном порядке;
5. Если терминалы в стеке и в ленте совпадают, то данный терминал удаляется из ленты и стека. Иначе возвращаемся в предыдущее сохраненное состояние и выбираем другую цепочку нетерминала;
6. Если в магазине встретился нетерминал, переходим к пункту 4;
7. Если наш символ достиг дна стека, и лента в этот момент пуста, то синтаксический анализ выполнен успешно. Иначе генерируется исключение.

СЕМАНТИЧЕСКИЙ АНАЛИЗАТОР:

Семантический анализатор – часть транслятора, выполняющая семантический анализ, то есть проверку исходного кода на наличие ошибок, которые невозможно отследить при помощи регулярной и контекстно-свободной грамматики. Входными данными являются таблица лексем и идентификаторов.

Проверяет смысл используемых инструкций

* Выводит список ошибок, если они имеются, иначе подтверждение об их отсутствии

При нахождении ошибок анализ не завершается , а продолжает разбирать все лексемы и находить все ошибки

ПОЛЬСКАЯ ЗАПИСЬ

Алгоритм построения польской записи:

исходная строка: выражение;

результирующая строка: польская запись;

стек: пустой;

исходная строка просматривается слева направо;

операнды переносятся в результирующую строку;

операция записывается в стек, если стек пуст;

операция выталкивает все операции с большим или равным приоритетом в результирующую строку;

отрывающая скобка помещается в стек;

закрывающая скобка выталкивает все операции до открывающей скобки, после чего обе скобки уничтожаются.

При встречи идентификатора типа - функции , идет подсчет передаваемых параметров до закрывающей круглой скобки затем ставится собака @ и число передаваемых параметров

Также на данном этапе преобразуется отрицательный идентификатор, в котором минус перед идентификатором заменяется на тильду

На слайде преведен пример записи выражения с вызовом функции , отрицательным идентификатором. Так же показа сокращенная форма записи сложения числа

ГЕНЕРАЦИЯ КОДА

Генератор кода– часть транслятора, выполняющая генерацию ассемблерного кода на основе полученных данных на предыдущих этапах трансляции. На вход генератора подаются таблица лексем и таблица идентификаторов, на основе которых генерируется файл с ассемблерным кодом.

В языке генерация кода является заключительным этапом трансляции. Генератор принимает на вход таблицы лексем и идентификаторов, полученные в результате лексического анализа. В соответствии с таблицей лексем строится выходной файл на языке ассемблера, который будет являться результатом работы транслятора. В случае возникновения ошибок генерация кода не будет осуществляться. Структура генератора кода BAA-2019 представлена на слайде

Результат работы транслятора программы и выполнение прораммы можно увидеть на слайде

ТЕСТИРОВАНИЕ ТРАНСЛЯТОРА

Также важным этапом разработки языка программирования является его тестирование Примеры нескольких тестов приведены на экране

Деление на ноль

Типы данных в выражении не совпадают

Тип функции и возвращаемого значения не совпадают

Так же можно увидеть необычный синтаксис условного оператора оператор

И в качестве примера кода есть вот такой интересный пример, мы объявляем строку , вычисляем ее длину с пом стандр функции , затем считаем факториал

С помощи функции fact в которой ипользуется рекурсия, затем считаем количество цифр в факториале с помощи функции Inlength генерируем с помощью стандартной функции число в диапозоне от –кол цифр факториала до количества цифр в факториале, проверяем на отрицательность его , если он меньше 0 то меняем знак , и ищем цифру на позиции числа, которое было сгенерировано с помощью функции getDigit.