1. **Лексический анализ.**

На вход компилятора и лексического анализатора поступает цепочка символов некоторого алфавита. Работа лексического анализатора заключается в том, чтобы сгруппировать определенные символы в единые синтаксические объекты, называемые лексемами. Какие объекты считаются лексемами, зависит от определения языка. Кроме терминальных символов (+,-,/,\*,(,)), которые сами по себе являются лексемами, в программе некоторые комбинации символов часто рассматриваются как единые объекты. Среди типичных примером можно указать следующие:

* В некоторых языках цепочка, состоящая из одного или более пробелов, обычно рассматривается как один элемент
* Ключевые слова: begin, end, to, do, integer и т.д. каждое считается одним символом
* Каждая цепочка, представляющая цифровую константу, рассматривается как один элемент текста.
* Идентификаторы, используемые имена переменных, функций, процедур, меток и т.д. так же считаются лексическими единицами алгоритмического языка

Лексический анализатор должен исходный текст программы представить в виде последовательности лексем. Для эффективности последующих действий каждая лексема обычно представляется некоторым кодом фиксированной длины (целое число), а не в виде строки символов переменной длины. В случае идентификатора или константы необходимы дополнительные данные (тип и указание на адрес ячейки памяти, где они хранятся). Обычно эти данные находятся в таблице символов, и в качестве дополнительной информации для лексемы типа идентификатор или константа может служить указатель на соответствующий элемент таблицы.

Таким образом, результат обработки лексическим анализатором обрабатываемой программы, можно представить последовательность лексем.

1. **Синтаксический анализ. Метод операторного предшествования.**

Синтаксический анализ – второй этап компиляции. Во время этого этапа предложения программы распознаются как языковые конструкции используемой грамматики. Для того чтобы выяснить, принадлежит ли предложение языку, необходимо построить алгоритм, который для любого предложения, допустимого грамматикой, давал бы последовательность выводов этой цепочки к начальному символу грамматики. Мы можем рассматривать этот процесс, как построение дерева грамматического разбора для транслируемых предложений. Различаются две категории алгоритмов разбора : нисходящий и восходящий. Эти термины соответствуют способу построения синтаксического дерева.

Метод операторного предшествования:

Метод относится к восходящим, начиная работу с конечных узлов грамматического дерева и пытается объединить их построение узлов все более и более высокого уровня до тех пор, пока не будет достигнут корень дерева. Метод операторного предшествования основан на анализе пар последовательного расположенных операторов исходной программы и решения вопроса о том, какой из них должен выполняться первым. В рамках этого метода предложения сканируются слева направо до тех пор, пока не будет найдено подвыражение, операторы которого имеют более высокий уровень предшествования, чем соседние операторы. Далее это подвыражение распознается в терминах вывода используемой грамматики. Этот процесс продолжается до тех пор, пока не достигнут корень дерева, что и будет означать окончание процесса грамматического разбора.

1. **Внутренние формы представления исходной программы.**

На выходе синтаксического анализатора формируется программа во внутреннем представлении. Существует несколько различных способов представления программы в некоторой промежуточной форме для анализа и оптимизации кода: последовательность четверок, последовательность троек, постфиксная/префиксная запись, синтаксическое дерево.

Последовательность четверок:

Каждая четверка записывается в виде:

Операция ор1 ор2 рез

Операция – выполняется объектным кодом функции

Ор1 и ор2 – операнды этой операции

Рез – определяет, куда должно быть помещено значение. Все четверки расположены в том порядке, в котором должны выполняться соответствующие инструкции объектного кода, что существенно облегчает анализ для оптимизации кода. Это означает, что трансляция в машинные коды будет относительно простой. Последовательность четверок и является результатом синтаксического анализатора.

Постфиксная запись:

Обладает двумя свойствами, благодаря которым ее используют как промежуточную. Форму представления исходной программы при трансляции:

1. Для записи любого выражения не нужны скобки (оператор следует за операндами). Нет неопределенности

2. К моменту считывания очередного оператора соответствующие операнды уже прочитаны. Поэтому оператор может быть выполнен без чтения каких-либо дополнительных данных.