1. **Выбор способа представления грамматики.  
   Для представления грамматики в нашем языке была выбрана Форма Бэкуса — Наура.**
2. **Написание лексического анализатора.  
   На вход компилятору поступает текстовый файл (исходник). Лексический анализатор нужен для того, чтобы выделить в этом файле токены.**
3. **Создание синтаксического парсера.  
   Его задача, используя токены, полученные от лексического анализатора, сформировать своего рода дерево, в котором иерархия и связи отображают структуру кода.**
4. **Получение Абстрактного Синтаксического Дерева(AST ).  
   После того, как у нас будет готов парсер, мы будем строить Abstract Syntax Tree (AST) (Абстрактное синтаксическое дерево). AST отображает программу таким образом, что для более поздних стадий компилятора (например, генерации кода) она легко интерпретируется.**
5. **Генерация LLVM.  
   С готовым AST мы можем генерировать LLVM. В основе LLVM лежит промежуточное представление кода (intermediate representation, IR), над которым можно производить трансформации во время компиляции, компоновки (linking) и выполнения.**
6. **Расширение языка: Поток управления.  
   Теперь, когда у нас есть работающий простейший язык программирования, мы сможем расширить его управляющими конструкциями (if/then/else и циклы).**
7. **Расширение языка: Изменяемые переменные.  
   Добавление пользовательских локальных переменных с помощью оператора присваивания.**
8. **Расширение языка: Создание стандартной библиотеке.  
   Разработка основных функций в нашем языке.**
9. **Расширение языка: Пока не придумали.**

**Ссылки:**

1. [**https://habr.com/post/133780/**](https://habr.com/post/133780/)
2. [**https://habr.com/post/120424/**](https://habr.com/post/120424/)
3. [**https://habr.com/post/119850/**](https://habr.com/post/119850/)