Памяти переводчика

Семена Викторовича Минца,

просто очень хорошего человека

Эта книга – его последний перевод.

С Семеном было очень приятно работать – он был немногословен и деловит.

Отлично знал информатику и программирование и перевел для нас «Введение в логическое программирование», «Объяснимые модели искусственного интеллекта на Python», «Искусство неизменяемой архитектуры» и эту последнюю.

Он ушел слишком несправедливо рано, мог бы еще многое сделать.

Будет не хватать его. Людей, особенно талантливых, заменить невозможно.

Заместитель главного редактора
Сенченкова Елена

Оглавление

Эта книга посвящается Сьюзи, Кертису, Кэри и всем, кто создает свои собственные языки программирования.

Клинтон Л. Джеффери

Об авторах

Клинтон Л. Джеффри – профессор и заведующий кафедрой компьютерных наук и инженерии Горно-технологического института Нью-Мексико. Он получил степень бакалавра в Вашингтонском университете, а также степень магистра и доктора философии в Университете Аризоны в области компьютерных наук. Проводил исследования и написал много книг и статей по языкам программирования, мониторингу программ, отладке, графике, виртуальным средам и визуализации. Вместе с коллегами изобрел язык программирования Unicon.

О рецензентах

Филлип Ли – доброволец Корпуса мира в Сараваке, Малайзия. Он получил степень бакалавра в Университете штата Орегон, магистра, докторскую степень в Университете Вашингтона, степень магистра в области малайской/индонезийской литературы в Университете Малайзии и степень магистра в области вычислительной техники в Университете Мердока в Перте. Преподавал для студентов и аспирантов в Оклендском университете и Университете Мердока. У Филиппа есть публикации по латинской, греческой, малайской и индонезийской литературе. Он является сопрограммистом библиотеки Конгресса *thomas.loc.gov*, поисковой системы Конгресса Национальной медицинской библиотеки *toxnet.nlm.nih.gov*. Кроме того, трудится разработчиком программ анализа текста для англо-иранского словаря Фонда Тун Джуга.

Стив Уамплер получил степень доктора философии в области компьютерных наук в Университете Аризоны. После чего он был адъюнкт-профессором компьютерных наук с 1981 по 1993 год. Стив работал разработчиком программного обеспечения в нескольких крупных проектах телескопов, включая проект Gemini 8m Telescopes Project и Солнечный телескоп Daniel K Inouye, в рамках Ассоциации исследований в области астрономии. Наряду с этим он был рецензентом программного обеспечения для ряда крупных телескопов, в том числе LSST, TMT, GMT, Keck, VLT ESO и GTC. Стив был техническим рецензентом первого издания книги Марка Собелла «Практическое руководство по операционной системе Linux», 1997 год.

Предисловие

После 60 лет высокоуровневой разработки языков программирование все еще остается сложным. Спрос на программное обеспечение постоянно увеличивающегося объема и сложности реализации резко возрос из-за аппаратных достижений, в то время как языки программирования совершенствуются гораздо медленнее. Создание новых языков для конкретных целей – одно из противоядий от кризиса программного обеспечения.

Эта книга посвящена созданию новых языков программирования. Вводится тема проектирования языка программирования, хотя основной акцент делается на реализации языка программирования. В рамках этой интенсивно изучаемой темы новым аспектом данной книги является слияние традиционных инструментов компиляции (Flex и Byacc) с двумя языками реализации более высокого уровня. Язык очень высокого уровня (Unicon) обрабатывает структуры данных и алгоритмы компилятора, как нож масло, в то время как основной современный язык (Java) показывает, как реализовать тот же код в более типичной производственной среде.

Для кого эта книга

Эта книга предназначена для разработчиков программного обеспечения, заинтересованных в идее создания собственного языка или разработки языка, специфичного для конкретной предметной области. Студенты, изучающие информатику на курсах построения компиляторов, также найдут эту книгу весьма полезной в качестве практического руководства по реализации языка в дополнение к другим теоретическим учебникам. Чтобы извлечь максимальную пользу из данной книги, требуются знания среднего уровня и опыт работы с языком высокого уровня, таким как Java или C++.

Что скрывает обложка

В *главе 1 «Зачем создавать другой язык программирования?»* обсуждается, когда следует создавать язык программирования, а когда вместо этого создавать библиотеку функций или библиотеку классов. Многие читатели этой книги уже знают, что они хотят создать свой собственный язык программирования. Некоторые должны вместо этого создать библиотеку.

Глава 2 «Проектирование языка программирования» описывает, как точно определить язык программирования, что важно знать, прежде чем пытаться создать язык программирования. Это включает в себя разработку лексических и синтаксических особенностей языка, а также его семантики. Хорошие языковые проекты обычно используют как можно больше знакомого синтаксиса.

Глава 3 «Сканирование исходного кода» представляет лексический анализ, включая регулярные обозначения выражений и инструменты Ulex и JFlex.

В конце вы будете открывать файлы исходного кода, читать их символ за символом и сообщать об их содержимом в виде потока токенов, состоящих из отдельных слов, операторов и знаков препинания в исходном файле.

В *главе 4 «Синтаксический анализ»* представлен синтаксический анализ, включая контекстно-свободные грамматики и инструменты iyacc и byacc/j. Вы узнаете, как отлаживать проблемы в грамматиках, которые препятствуют синтаксическому анализу, и сообщать о синтаксических ошибках, когда они возникают.

В *главе 5 «Синтаксические деревья»* рассматриваются синтаксические деревья. Основным побочным продуктом процесса синтаксического анализа является построение древовидной структуры данных, которая представляет логическую структуру исходного кода. Построение узлов дерева происходит в семантических действиях, которые выполняются для каждого правила грамматики.

В *главе 6 «Таблицы символов»* показано, как создавать таблицы символов, вставлять в них символы и использовать таблицы для выявления двух видов семантических ошибок: необъявленных и незаконно повторно объявленных переменных. Чтобы понять ссылки на переменные в исполняемом коде, необходимо отслеживать область действия и время жизни каждой переменной. Это достигается с помощью табличных структур данных, которые являются вспомогательными для синтаксического дерева.

Глава 7 «Проверка базовых типов» посвящена проверке типов, которая является основной задачей, требуемой в большинстве языков программирования. Проверка типов может выполняться во время компиляции или во время выполнения. В этой главе рассматривается общий случай статической проверки типов во время компиляции для базовых типов, также называемых атомарными, или скалярными, типами.

В *главе 8 «Проверка типов массивов, вызовов методов и доступа к структурам»* показано, как выполнять проверку типов массивов, параметров и возвращаемых типов вызовов методов в подмножестве Java Jzero. Более сложные части проверки типов – это когда должны быть проверены несколько массовов или составные массивы.

Глава 9 «Генерация промежуточного кода» показывает вам, как генерировать промежуточный код, рассматривая примеры для языка Jzero. Прежде чем сгенерировать код для выполнения, большинство компиляторов превращают синтаксическое дерево в список машинно независимых инструкций промежуточного кода. На этом этапе обрабатываются ключевые аспекты потока управления, такие как генерация меток и инструкции goto.

В *главе 10 «Раскрашивание синтаксиса в среде IDE»* рассматривается задача включения информации из синтаксического анализа в среду IDE, чтобы обеспечить раскрашивание синтаксиса и визуальную обратную связь о синтаксических ошибках. Язык программирования требует большего, чем просто компилятор или интерпретатор, – он требует экосистемы инструментов для разработчиков. Эта экосистема может включать в себя отладчики, интерактивную справку или интегрированную среду разработки. Эта глава представляет собой пример Unicon, взятый из среды разработки Unicon IDE.

Глава 11 «Интерпретаторы байт-кода» посвящена разработке набора команд и интерпретатора, который выполняет байт-код. Новый язык, специфич-

ный для конкретной предметной области, может включать в себя высокоуровневые функции программирования предметной области, которые напрямую не поддерживаются основными процессорами. Наиболее практичным способом генерации кода для многих языков является генерация байт-кода для абстрактной машины, набор команд которой напрямую поддерживает целевое назначение языка с последующим выполнением этой программы путем интерпретации команд.

Глава 12 «Генерация байт-кода» рассматривает прохождение по гигантскому связанному списку, перевод каждой инструкции промежуточного кода в одну или несколько инструкций байт-кода. Как правило, это цикл для обхода связанного списка с разным фрагментом кода для каждой промежуточной кодовой инструкции.

Глава 13 «Генерация собственного кода» содержит обзор генерации собственного кода для x86_64. Некоторые языки программирования требуют собственного кода для достижения своих требований к производительности. Генерация собственного кода похожа на генерацию байт-кода, но более сложна, включает в себя выделение регистров и режимы адресации памяти.

Глава 14 «Реализация операторов и встроенных функций» описывает, как поддерживать языковые функции очень высокого уровня и специфичные для предметной области, добавляя операторы и функции, встроенные в язык. Языковые возможности очень высокого уровня и специфичные для предметной области часто лучше всего представлены операторами и функциями, встроенными в язык, а не библиотечными функциями. Добавление встроенных модулей может упростить ваш язык, улучшить его.

Глава 15 «Структуры управления доменом» описывает, когда вам нужна новая структура управления, и предоставляет примеры структур управления, которые обрабатывают текст с помощью сканирования строк и отображают графические области. Общий код в предыдущих главах охватывал основные условные и циклические структуры управления, но языки, зависящие от предметной области, часто имеют уникальную или настраиваемую семантику, для которой они вводят новые структуры управления. Добавление новых структур управления существенно сложнее, чем добавление новой функции или оператора, но именно это делает языки, специфичные для предметной области, достойными разработки, а не просто написания библиотек классов.

В *главе 16 «Сборка мусора»* представлена пара методов, с помощью которых вы можете реализовать сборку мусора на вашем языке. Управление памятью является одним из наиболее важных аспектов современных языков программирования, и все классные языки программирования имеют функцию автоматического управления памятью с помощью сборки мусора. В этой главе приводится несколько вариантов того, как вы могли бы реализовать сборку мусора на вашем языке, включая подсчет ссылок и сборку мусора с пометкой и разверткой.

Глава 17 «Заключительные мысли» отражает основные темы, представленные в книге, и дает вам некоторую пищу для размышлений. В ней рассматривается то, что было извлечено из написания этой книги, и дается множество рекомендаций для дальнейшего чтения.

Приложение «Unicon Essentials» описывает язык программирования Unicon в достаточном количестве, чтобы понять те примеры в этой книге, которые

находятся в Unicon. Большинство примеров приведены рядом на Unicon и Java, но версии Unicon обычно короче и легче читаются.

Как получить от этой книги максимальную пользу

Чтобы понять эту книгу, вы должны быть программистом среднего уровня на Java или подобном языке; программист C, который знает объектно-ориентированный язык, подойдет.

Программное обеспечение, упомянутое в книге	Необходимая операционная система
Unicon 13.2, Uflex,	Windows, Linux
Java, Jflex, and Byacc/J	
GNU Make	

Инструкции по установке и использованию инструментов немного расширены, чтобы сократить время запуска, и приведены в главе 3 «Сканирование исходного кода» и главе 5 «Синтаксические деревья». Если вы технически одарены, вы, возможно, сможете запустить все эти инструменты на macOS, но во время написания этой книги они не использовались и не тестировались.

> **Примечание**
>
> Если вы используете цифровую версию этой книги, мы советуем вам ввести код самостоятельно или получить доступ к коду из репозитория книги на GitHub (ссылка доступна в следующем разделе). Это поможет вам избежать любых потенциальных ошибок, связанных с копированием и вставкой кода.

Загрузка примеров

Примеры для данной книге вы можете загрузить на сайте нашего издательсва по ссылке: (_____страница книги)

Видео

Видеоролики с кодом в действии для этой книги можно посмотреть по адресу *https://bit.ly/3njc15D*.

Цветные иллюстрации

Цветные иллюстрации к данной книге вы можете загрузить на сайте нашего издательсва по ссылке: (_____страница книги).

Используемые сокращения

Вот несколько текстовых условных обозначений, используемых на протяжении всей этой книги.

Код в тексте: указывает кодовые слова в тексте, имена таблиц базы данных, имена папок, имена файлов, расширения файлов, пути, фиктивные URL-адреса, вводимые пользователем, и дескрипторы Twitter. Вот пример: «Соответствующий Java main() должен быть помещен в класс».

Блок кода задается следующим образом:

```
procedure main(argv)
procedure main(argv)
 yyin := open(argv[1])
 yyin := open(argv[1]) yyin := open(argv[1])
 yyin := open(argv[1])
 yyin := open(argv[1])
```

Когда мы хотим привлечь ваше внимание к определенной части блока кода, соответствующие строки или элементы выделяются жирным шрифтом:

```
procedure main(argv)
procedure main(argv)
 yyin := open(argv[1])
 yyin := open(argv[1]) yyin := open(argv[1])
 yyin := open(argv[1])
```

Ввод или вывод из командной строки записывается так:

```
procedure main(argv)
 yyin := open(argv[1])
```

Жирный шрифт: обозначает новый термин, важное слово или слова, которые вы видите на экране. Например, слова в меню или диалоговых окнах выделены **жирным** шрифтом. Вот пример: «Выберите **Информацию о системе** на панели администрирования».

> **Советы или важные примечания**
>
> Представляют собой текст, помещенный в рамку.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы удостовериться в качестве наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в тексте или в коде, – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от расстройств и поможете нам улучшить последующие версии данной книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу *dmkpress@gmail.com*, и мы исправим это в следующих тиражах.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и Packt очень серьезно относятся к вопросам защиты ав-

торских прав и лицензирования. Если вы столкнетесь в интернете с незаконно выполненной копией любой нашей книги, пожалуйста, сообщите нам адрес копии или веб-сайта, чтобы мы могли применить санкции.

Пожалуйста, свяжитесь с нами по адресу электронной почты *dmkpress@gmail.com* со ссылкой на подозрительные материалы.

Мы высоко ценим любую помощь по защите наших авторов, помогающую нам предоставлять вам качественные материалы.

Часть I

Интерфейсы языка программирования

В этой части вы создадите базовую конструкцию языка и реализуете интерфейс компилятора для него, включая лексический анализатор и синтаксический анализатор, который строит синтаксическое дерево из входного исходного файла.

Эта часть включает в себя следующие главы:

- глава 1 «Зачем создавать еще один язык программирования»;
- глава 2 «Проектирование языка программирования»;
- глава 3 «Сканирование исходного кода»;
- глава 4 «Парсинг»;
- глава 5 «Деревья синтаксиса».