

# 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

<b>Об авторах .....</b>	<b>16</b>
<b>О рецензентах .....</b>	<b>16</b>
<b>Предисловие.....</b>	<b>17</b>
Для кого эта книга .....	17
Что скрывает обложка .....	17
Как получить от этой книги максимальную пользу.....	20
Загрузка примеров .....	20
Видео .....	20
Цветные иллюстрации .....	20
Используемые сокращения.....	20
Список опечаток .....	21
Нарушение авторских прав .....	21
<b>ЧАСТЬ I. ИНТЕРФЕЙСЫ ЯЗЫКА ПРОГРАММИРОВАНИЯ.....</b>	<b>23</b>
<b>Глава 1. Зачем создавать еще один язык программирования? .....</b>	<b>25</b>
Итак, вы хотите создать свой собственный язык программирования.....	25
Типы реализации языков программирования.....	26
Организация реализации языка байт-кода.....	27
Языки, используемые в примерах.....	28
Язык и библиотека – в чем разница? .....	29
Применимость к другим задачам разработки программного обеспечения .....	30
Определение требований к вашему языку .....	31
Тематическое исследование – требования, которые вдохновили на создание языка Unicon .....	33
Требование Unicon № 1 – сохранять то, что люди любят в Icon.....	33
Требование Unicon № 2 – поддержка крупномасштабных программ, работающих с большими данными.....	34
Требование Unicon № 3 – высокоуровневый ввод/вывод для современных приложений.....	34
Требование Unicon № 4 – обеспечить универсально реализуемые системные интерфейсы .....	35
Заключение .....	35
Вопросы.....	36

<b>Глава 2. Дизайн языка программирования .....</b>	<b>37</b>
Определение видов слов и пунктуации в вашем языке .....	38
Определение потока управления .....	40
Решение о том, какие типы данных поддерживать .....	41
Атомарные типы.....	41
Составные типы.....	42
Типы, специфичные для конкретной области .....	44
Общая структура программы .....	44
Завершение определения языка Jzero .....	45
Тематическое исследование – проектирование графических объектов в Unicon .....	46
Поддержка языка для графики 2D.....	47
Добавление поддержки трехмерной графики.....	49
Заключение .....	50
Вопросы.....	50
<b>Глава 3. Сканирование исходного кода .....</b>	<b>52</b>
Технические требования.....	52
Лексемы, лексические категории и токены.....	53
Регулярные выражения .....	54
Правила регулярных выражений .....	54
Примеры регулярных выражений.....	56
Использование UFlex и JFlex.....	57
Раздел заголовка.....	58
Раздел регулярных выражений .....	58
Написание простого сканера исходного кода .....	59
Запуск сканера .....	62
Токены и лексические атрибуты .....	63
Расширение нашего примера для построения токенов .....	64
Написание сканера для Jzero .....	66
Спецификация Jzero flex .....	66
Код Unicon Jzero .....	69
Код Java Jzero.....	72
Запуск сканера Jzero.....	75
Регулярных выражений не всегда достаточно .....	76
Заключение .....	80
Вопросы.....	80
<b>Глава 4. Парсинг.....</b>	<b>81</b>
Технические требования.....	81
Анализ синтаксиса .....	82
Понимание бесконтекстных грамматик.....	83
Написание правил бесконтекстной грамматики .....	84
Написание правил для программных конструкций .....	85
Использование yacc и yacc/J.....	87
Объявление символов в разделе заголовка .....	88

Составление раздела бесконтекстной грамматики <i>yacc</i> .....	89
Понимание парсеров <i>yacc</i> .....	90
Устранение конфликтов в парсерах <i>yacc</i> .....	92
Исправление синтаксических ошибок.....	93
Создание игрушечного примера .....	93
Написание парсера для Jzero .....	98
Спецификация Jzero lex .....	98
Спецификация <i>yacc</i> в Jzero .....	98
Код Unicon Jzero .....	103
Код парсера Jzero на языке Java .....	105
Запуск парсера Jzero .....	105
Улучшение сообщений об ошибках синтаксиса.....	107
Добавление деталей в сообщения Unicon об ошибках синтаксиса .....	108
Добавление деталей в сообщения Java об ошибках синтаксиса .....	108
Использование Merge для создания лучших сообщений об ошибках синтаксиса .....	109
Заключение .....	110
Вопросы.....	110
<b>Глава 5. Деревья синтаксиса .....</b>	<b>111</b>
Технические требования.....	111
Использование GNU make .....	112
Изучение деревьев .....	115
Определение типа дерева синтаксиса .....	115
Деревья разбора в сравнении с деревьями синтаксиса.....	117
Создание листьев из терминальных символов .....	119
Обертывание токенов в листья.....	120
Работа со стеком значений YACC .....	120
Обертка листьев для стека значений парсера .....	122
Определение нужных вам листьев .....	123
Построение внутренних узлов из правил производства.....	124
Доступ к узлам дерева в стеке значений .....	124
Использование фабричного метода узла дерева .....	126
Формирование деревьев синтаксиса для языка Jzero.....	127
Отладка и тестирование вашего дерева синтаксиса .....	134
Предотвращение распространенных ошибок в дереве синтаксиса .....	134
Распечатка вашего дерева в текстовом формате .....	136
Печать дерева с помощью dot.....	138
Заключение .....	143
Вопросы.....	143
<b>ЧАСТЬ II. ОБХОДЫ ДЕРЕВА СИНТАКСИСА .....</b>	<b>145</b>
<b>Глава 6. Таблицы символов .....</b>	<b>147</b>
Технические требования.....	148
Создание основы для таблиц символов .....	148
Объявления и области видимости.....	148

Присваивание и разыменование переменных .....	149
Выбор подходящего обхода дерева для работы .....	150
Создание и заполнение таблиц символов для каждой области видимости .....	151
Добавление семантических атрибутов к деревьям синтаксиса .....	152
Определение классов для таблиц символов и записей в таблицах символов .....	154
Создание таблиц символов .....	155
Заполнение таблиц символов .....	157
Синтез атрибута isConst .....	159
Проверка наличия необъявленных переменных .....	160
Идентификация тел методов .....	160
Выявление использования переменных в теле метода .....	161
Поиск повторно объявленных переменных .....	162
Вставка символов в таблицу символов .....	163
Сообщение о семантических ошибках .....	163
Обработка пакетов и областей видимости классов в Unicon .....	164
Искажение имен .....	165
Вставка self для ссылок на переменные-члены .....	166
Вставка self в качестве первого параметра в вызовы методов .....	166
Тестирование и отладка таблиц символов .....	167
Заключение .....	169
Вопросы .....	170
<b>Глава 7. Проверка базовых типов .....</b>	<b>171</b>
Технические требования .....	171
Представление типов в компиляторе .....	171
Определение базового класса для представления типов .....	172
Подклассификация базового класса для сложных типов .....	173
Присвоение информации о типе объявленным переменным .....	175
Синтез типов из зарезервированных слов .....	177
Наследование типов в списке переменных .....	178
Определение типа в каждом узле дерева синтаксиса .....	179
Определение типа в листьях .....	180
Вычисление и проверка типов во внутренних узлах .....	182
Проверка типов во время выполнения и вывод типов в Unicon .....	186
Заключение .....	188
Вопросы .....	188
<b>Глава 8. Проверка типов в массивах, вызовах методов и доступах к структурам .....</b>	<b>189</b>
Технические требования .....	189
Операции проверки типов массивов .....	189
Управление объявлениями переменных в массивах .....	190
Проверка типов при создании массива .....	191
Проверка типов при обращении к массиву .....	193
Проверка вызовов методов .....	194

Вычисление параметров и информации о возвращаемом типе.....	194
Проверка типов в каждом месте вызова метода.....	197
Проверка типов в операторах возврата.....	200
Проверка обращений к структурированным типам.....	202
Обработка объявлений переменных экземпляра.....	202
Проверка типов при создании экземпляра.....	203
Проверка типов при обращении к экземпляру.....	205
Заключение.....	208
Вопросы.....	209
<b>Глава 9. Генерация промежуточного кода.....</b>	<b>210</b>
Технические требования.....	210
Подготовка к генерации кода.....	210
Зачем генерировать промежуточный код?.....	211
Изучение областей памяти в созданной программе.....	211
Представление типов данных для промежуточного кода.....	212
Добавление атрибутов промежуточного кода в дерево.....	214
Генерация меток и временных переменных.....	215
Набор инструкций промежуточного кода.....	218
Инструкции.....	218
Декларации.....	219
Аннотирование деревьев синтаксиса метками для потока управления.....	219
Генерация кода для выражений.....	222
Генерация кода для потока управления.....	225
Генерация целевых меток для выражений условий.....	225
Генерация кода для циклов.....	228
Генерация промежуточного кода для вызовов методов.....	229
Проверка сгенерированного промежуточного кода.....	231
Заключение.....	232
<b>Глава 10. Раскраска синтаксиса в IDE.....</b>	<b>233</b>
Загрузка примеров IDE, используемых в этой главе.....	234
Интеграция компилятора в редактор программиста.....	236
Анализ исходного кода из среды IDE.....	236
Отправка выходных данных компилятора в IDE.....	237
Предотвращение повторного разбора всего файла при каждом изменении.....	238
Использование лексической информации для раскрашивания токенов.....	242
Расширение компонента EditableTextList для поддержки цвета.....	242
Раскрашивание отдельных токенов по мере их создания.....	242
Подсветка ошибок с использованием результатов разбора.....	243
Добавление поддержки Java.....	245
Заключение.....	247

## **ЧАСТЬ III. ГЕНЕРАЦИЯ КОДА И СРЕДЫ ВЫПОЛНЕНИЯ ..... 249**

### **Глава 11. Интерпретаторы байт-кода..... 251**

Технические требования .....	251
Понимание, что такое байт-код.....	252
Сравнение байт-кода с промежуточным кодом.....	253
Построение набора инструкций байт-кода для Jzero.....	255
Определение формата файла байт-кода Jzero.....	255
Понимание основ работы стековой машины .....	258
Реализация интерпретатора байт-кода .....	259
Загрузка байт-кода в память .....	259
Инициализация состояния интерпретатора .....	261
Выборка инструкций и продвижение указателя инструкции.....	263
Декодирование инструкций .....	264
Выполнение инструкций .....	265
Запуск интерпретатора Jzero .....	268
Написание среды выполнения для Jzero.....	269
Запуск программы Jzero.....	269
Изучение iconx, интерпретатора байт-кода Unicon .....	270
Понимание целенаправленного байт-кода .....	271
Сохранение информации о типе во время выполнения .....	271
Выборка, декодирование и выполнение инструкций.....	272
Создание остальной части среды выполнения .....	272
Заключение .....	273
Вопросы.....	273

### **Глава 12. Генерация байт-кода ..... 274**

Технические требования .....	274
Преобразование промежуточного кода в байт-код Jzero .....	275
Добавление класса для инструкций байт-кода .....	276
Соответствие адресов промежуточного кода адресам байт-кода.....	276
Реализация метода генератора байт-кода.....	278
Генерация байт-кода для простых выражений .....	278
Генерация кода для обработки указателей.....	280
Генерация байт-кода для безусловных и условных переходов .....	281
Генерация кода для вызовов методов и возвратов .....	282
Обработка меток и других псевдоинструкций промежуточного кода.....	284
Сравнение ассемблера байт-кода с двоичными форматами .....	285
Вывод байт-кода в формате ассемблера .....	285
Вывод байт-кода в двоичном формате .....	287
Линковка, загрузка и включение среды выполнения .....	288
Пример Unicon – генерация байт-кода в iconx .....	288
Заключение .....	290
Вопросы.....	290

**Глава 13. Генерация собственного кода..... 292**

Технические требования.....	292
<i>Принятие решения о генерации собственного кода.....</i>	<i>292</i>
Знакомство с набором инструкций x64.....	293
Добавление класса для инструкций x64.....	294
Соответствие областей памяти регистровым режимам адресации x64.....	294
Использование регистров.....	295
Начинаем с нулевой стратегии.....	296
Преобразование промежуточного кода в код x64.....	299
Соответствие адресов промежуточного кода местоположению в x64.....	300
Реализация метода генератора кода x64.....	303
Генерация кода x64 для простых выражений.....	304
Генерация кода для обработки указателей.....	305
Генерация собственного кода для безусловных и условных переходов.....	306
Генерация кода для вызовов методов и возвратов.....	307
Обработка меток и псевдоинструкций.....	309
Генерация выходных данных x64.....	311
Запись кода x64 в формате ассемблера.....	311
Переход от ассемблера к объектному файлу.....	312
Линковка, загрузка и включение среды выполнения.....	313
Заключение.....	314
Вопросы.....	315

**Глава 14. Реализация операторов и встроенных функций ..... 316**

Реализация операторов.....	316
Подразумевают ли операторы аппаратную поддержку, и наоборот.....	317
Добавление конкатенации строк в генерацию промежуточного кода.....	318
Добавление конкатенации строк в интерпретатор байт-кода.....	319
Добавление конкатенации строк в собственную среду выполнения.....	322
Написание встроенных функций.....	323
Добавление встроенных функций в интерпретатор байт-кода.....	323
Написание встроенных функций для использования в реализации собственного кода.....	324
Интеграция встроенных функций со структурами управления.....	325
Разработка операторов и функций для Unicon.....	326
Написание операторов в Unicon.....	327
Разработка встроенных функций Unicon.....	329
Заключение.....	330
Вопросы.....	330

<b>Глава 15. Структуры управления доменами .....</b>	<b>331</b>
Понимание необходимости новой структуры управления .....	331
Определение структуры управления .....	332
Устранение избыточных параметров .....	333
Сканирование строк в Icon и Unicon .....	333
Среды сканирования и их примитивные операции .....	334
Устранение избыточных параметров с помощью структуры управления .....	336
Рендеринг областей в Unicon .....	337
Отображение 3D-графики из списка отображения .....	337
Указание областей рендеринга с помощью встроенных функций .....	338
Изменение графических уровней детализации с помощью вложенного рендеринга областей .....	339
Создание структуры управления рендерингом областей .....	340
Добавление зарезервированного слова для рендеринга областей .....	340
Добавление правила грамматики .....	341
Проверка wsection на семантические ошибки .....	342
Генерация кода для структуры управления wsection .....	343
Заключение .....	345
Вопросы .....	345
<b>Глава 16. Сборка мусора .....</b>	<b>347</b>
Оценка важности сборки мусора .....	347
Подсчет ссылок на объекты .....	349
Добавление подсчета ссылок в Jzero .....	350
Генерация кода для распределения кучи .....	350
Изменение сгенерированного кода для оператора присваивания .....	352
Учет недостатков и ограничений, связанных с подсчетом ссылок .....	353
Пометка реальных данных и очистка остальных .....	354
Организация областей памяти кучи .....	355
Обход базиса для пометки живых данных .....	357
Восстановление живой памяти и размещение ее в непрерывных фрагментах .....	361
Заключение .....	363
Вопросы .....	364
<b>Глава 17. Заключительные размышления .....</b>	<b>365</b>
Размышления о том, что изучено при написании этой книги .....	365
Решение о том, куда двигаться дальше .....	366
Изучение дизайна языков программирования .....	366
Изучение реализации интерпретаторов и машин байт-кода .....	367
Приобретение опыта в оптимизации кода .....	368
Мониторинг и отладка выполнения программ .....	369
Проектирование и реализация IDE и строителей GUI .....	369
Изучение ссылок для дальнейшего чтения .....	370

Изучение дизайна языков программирования.....	370
Изучение реализации интерпретаторов и машин байт-кода.....	371
Приобретение опыта работы с собственным кодом и оптимизации кода.....	371
Мониторинг и отладка выполнения программ.....	372
Проектирование и реализация IDE и строителей GUI .....	372
Заключение .....	373
<b>ЧАСТЬ IV. ПРИЛОЖЕНИЕ.....</b>	<b>375</b>
<b>Приложение. Основы Unicon.....</b>	<b>377</b>
Запуск Unicon.....	377
Использование объявлений и типов данных Unicon .....	379
Объявление различных типов компонентов программы .....	379
Использование атомарных типов данных.....	381
Организация нескольких значений с помощью структурных типов .....	382
Оценка выражений.....	384
Формирование базовых выражений с помощью операторов.....	384
Вызов процедур, функций и методов .....	387
Итерации и выбор того, что и как выполнять .....	388
Генераторы.....	389
Отладка и вопросы окружения .....	390
Изучение основ отладчика UDB .....	390
Переменные окружения.....	391
Препроцессор.....	391
Мини-справочник функций .....	393
Избранные ключевые слова.....	398
<b>Оценки .....</b>	<b>400</b>
Глава 1.....	400
Глава 2.....	400
Глава 3.....	401
Глава 4.....	401
Глава 5.....	402
Глава 6.....	402
Глава 7.....	403
Глава 8.....	403
Глава 11.....	404
Глава 12.....	404
Глава 13.....	405
Глава 14.....	405
Глава 16.....	407

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

*Клинтон Л. Джеффри*

## 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](http://thomas.loc.gov), поисковой системы Конгресса Национальной медицинской библиотеки [toxnet.nlm.nih.gov](http://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 Yacc) 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 yacc и b yacc/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 массивов 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 издательства 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 издательства 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](mailto: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](mailto: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».