Э. А. Опалева
В. П. Самойленко

Языки программирования и методы трансляции

- ***Грамматика и распознающий автомат – формальные модели описания синтаксиса***
- ***Синтаксический анализ – ядро транслятора, определяющего его основные свойства***
- ***Атрибутные грамматики – простой и удобный способ описания семантики языка***

$$G = (\Sigma, N, S, P)$$

$$A \to \beta$$

$$(\alpha A \beta, \alpha' A \beta') \Rightarrow_T (\alpha\gamma\beta, \alpha'\gamma'\beta')$$

```
OUT1:
        i1 := c11;
Label11: if i1 > c12 then goto Label12;
        i2 := c21;
Label21: if i1 > c22 then goto Label2
        top;
OUT2:
        i1 := i1 + c13; goto Label11; Label12
        i2 := i2 + c23; goto Label21; Label22
        i2 := c21;
Label21: if i1 > c22 then goto Label22;
        top;
```

УЧЕБНОЕ ПОСОБИЕ

bhv®

Э. А. Опалева, В. П. Самойленко

Языки программирования и методы трансляции

Рекомендовано УМО по университетскому политехническому образованию для студентов высших учебных заведений, обучающихся по специальности 220400 (230105) – Программное обеспечение вычислительной техники и автоматизированных систем

Санкт-Петербург
«БХВ-Петербург»
2005

УДК 681.3.068+800.92
ББК 32.973-018.1я73
О-60

Опалева Э. А., Самойленко В. П.

О-60 Языки программирования и методы трансляции. — СПб.: БХВ-Петербург, 2005. — 480 с.: ил.

ISBN 5-94157-327-8

Учебное пособие содержит систематическое изложение теоретических основ перевода и компиляции. Рассмотрены общие вопросы разработки, описания и реализации языков программирования, формальные методы описания синтаксиса и семантики языков программирования, методы синтаксического анализа современных языков программирования. Приводится методика разработки описания перевода и пример использования этой методики для построения атрибутной транслирующей грамматики.

Для студентов и преподавателей вузов

УДК 681.3.068+800.92
ББК 32.973-018.1я73

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. гл. редактора	*Людмила Еремеевская*
Зав. редакцией	*Григорий Добин*
Редактор	*Нина Седых*
Компьютерная верстка	*Татьяны Олоновой*
Корректор	*Ксения Вальская*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Рецензенты:
Бузюков Л.Б., д.т.н., профессор, декан факультета сетей связи, систем коммутации и ВТ, зав. кафедрой цифровой вычислительной техники и информатики Санкт-Петербургского государственного университета телекоммуникаций.

Парфенов В.Г., д.т.н., профессор, декан факультета информационных технологий и программирования, зав. кафедрой информационных систем Санкт-Петербургского государственного университета информационных технологий, механики и оптики.

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 01.06.05.
Формат 70×100$^{1}/_{16}$. Печать офсетная. Усл. печ. л. 38,7.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5 Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 5-94157-327-8

©. Опалева Э. А., Самойленко В. П., 2005
© "БХВ-Петербург", 2005

Оглавление

Введение **9**

ЧАСТЬ I. ЯЗЫКИ ПРОГРАММИРОВАНИЯ **5**

Глава 1. Основные концепции языков программирования **15**

1.1. Парадигмы языков программирования 16
1.1.1. Императивные языки 16
1.1.2. Языки функционального программирования 17
1.1.3. Декларативные языки 18
1.1.4. Объектно-ориентированные языки 18
1.2. Критерии оценки языков программирования 19
1.2.1. Понятность 20
1.2.2. Надежность 21
1.2.3. Гибкость 22
1.2.4. Простота 23
1.2.5. Естественность 23
1.2.6. Мобильность 24
1.2.7. Стоимость 24
1.3. Объекты данных в языках программирования 25
1.3.1. Имена 26
1.3.2. Константы 27
1.3.3. Переменные 27
1.4. Механизмы типизации 29
1.4.1. Статические и динамические типы данных 29
1.4.2. Слабая типизация 31
1.4.3. Строгая типизация 32
1.4.4. Производные типы 32
1.4.5. Эквивалентность типов 34
1.4.6. Наследование атрибутов 35
1.4.7. Ограничения 36
1.4.8. Подтипы 37
1.4.9. Анонимные типы и подтипы 37
1.5. Время жизни переменных 38
1.6. Область видимости переменных 40

1.7. Типы данных ... 47
1.7.1. Элементарные типы данных ... 47
1.7.2. Символьные строки ... 55
1.7.3. Перечислимые типы ... 57
1.7.4. Ограниченные типы ... 58
1.7.5. Векторы и массивы ... 59
1.7.6. Записи ... 63
1.7.7. Объединения ... 65
1.7.8. Множества ... 66
1.7.9. Списки ... 69
1.8. Выражения и операторы присваивания ... 70
1.8.1. Арифметические выражения ... 70
1.8.2. Логические выражения ... 72
1.8.3. Операторы присваивания ... 74
1.9. Структуры управления на уровне операторов ... 75
1.9.1. Составные операторы ... 76
1.9.2. Условные операторы ... 76
1.9.3. Операторы цикла ... 80
1.10. Подпрограммы ... 81
1.10.1. Определение подпрограммы ... 82
1.10.2. Формальные и фактические параметры подпрограммы ... 83
1.10.3. Процедуры и функции ... 83
1.10.4. Методы передачи параметров ... 84
1.10.5. Сопрограммы ... 87
Контрольные вопросы ... 88

Глава 2. Описание языка программирования ... 91

2.1. Определение синтаксиса языка ... 92
2.1.1. Форма Бэкуса-Наура ... 92
2.1.2. Синтаксические диаграммы Вирта ... 93
2.2. Описание контекстных условий ... 94
2.3. Описание динамической семантики ... 95
2.3.1. Грамматические модели ... 96
2.3.2. Операционная семантика ... 97
2.3.3. Аксиоматическая семантика ... 98
2.3.4. Денотационная семантика ... 99
Контрольные вопросы ... 100
Упражнения ... 100

ЧАСТЬ II. ФОРМАЛЬНЫЕ ГРАММАТИКИ И РАСПОЗНАЮЩИЕ АВТОМАТЫ ... 103

Глава 3. Формальные грамматики и языки ... 105

3.1. Способы определения формальных языков ... 105

3.2. Формальные грамматики ... 108
3.3. Классификация формальных грамматик ... 111
3.4. Выводы и деревья выводов ... 113
3.5. Неоднозначность грамматик ... 121
3.6. Непустые, конечные и бесконечные языки ... 124
3.6.1. Непустые языки ... 125
3.6.2. Бесконечные языки ... 125
3.6.3. Проблема принадлежности ... 126
3.7. Эквивалентные преобразования КС-грамматик ... 127
3.7.1. Удаление бесполезных символов ... 127
3.7.2. Преобразование КС-грамматики с ε-правилами в эквивалентную неукорачивающую КС-грамматику ... 131
3.7.3. Исключение цепных правил ... 134
3.7.4. Удаление произвольного правила вывода ... 136
3.7.5. Устранение левой рекурсии ... 137
3.7.6. Левая факторизация ... 142
3.8. Нормальная форма Хомского ... 143
3.9. Нормальная форма Грейбах ... 145
3.10. Свойства замкнутости КС-языков ... 158
Контрольные вопросы ... 160
Упражнения ... 161

Глава 4. Конечные автоматы и преобразователи ... 163

4.1. Распознающий автомат ... 163
4.2. Конечный автомат ... 165
4.3. Способы задания конечных автоматов ... 168
4.4. Детерминированные конечные автоматы ... 170
4.5. Автоматные грамматики и конечные автоматы ... 172
4.6. Решение проблемы принадлежности для конечных автоматов ... 174
4.7. Решение проблемы пустоты языка для конечных автоматов ... 175
4.8. Решение проблемы эквивалентности для конечных автоматов ... 175
4.9. Конечные преобразователи ... 177
Контрольные вопросы ... 179
Упражнения ... 180

Глава 5. Автоматы и преобразователи с магазинной памятью ... 183

5.1. Определение автомата с магазинной памятью ... 183
5.2. Расширенные МП-автоматы ... 186
5.3. Эквивалентность МП-автоматов и КС-грамматик ... 192
5.4. Детерминированные МП-автоматы ... 199
5.5. Преобразователи с магазинной памятью ... 204
Контрольные вопросы ... 209
Упражнения ... 210

ЧАСТЬ III. МЕТОДЫ СИНТАКСИЧЕСКОГО АНАЛИЗА.....213

Глава 6. Общие методы синтаксического анализа 215

6.1. Определение разбора 215
6.2. Нисходящий разбор 217
6.3. Восходящий разбор 220
6.4. Моделирование недетерминированного МП-преобразователя 225
6.5. Алгоритм нисходящего разбора.....229
6.6. Алгоритм восходящего разбора.....242
6.7. Алгоритм Эрли.....249
Контрольные вопросы 259
Упражнения 259

Глава 7. *LL*(*k*)-грамматики.....260

7.1. Простые *LL*(1)-грамматики.....260
7.2. Определение *LL*(1)-грамматики 265
7.3. Алгоритм разбора для *LL*(1)-грамматик 267
7.4. *LL*(*k*)-грамматики.....272
7.5. Рекурсивный спуск 274
Контрольные вопросы 277
Упражнения 279

Глава 8. *LR*(*k*)-грамматики 280

8.1. Детерминированный разбор с помощью алгоритма "перенос-свертка" 280
8.2. *LR*(*k*)-грамматика 285
8.3. Алгоритм разбора для *LR*(*k*)-грамматики.....287
8.4. Построение *LR*(*k*)-анализаторов.....292
8.5. Алгоритм построения анализатора для *LR*(0)-грамматики без **ε**-правил 300
8.6. Алгоритм построения анализатора для *SLR*(1)-грамматики без **ε**-правил 308
8.7. Включение ε-правил в *LR*(0)- и *SLR*(1)-грамматики 314
Контрольные вопросы 317
Упражнения 318

Глава 9. Грамматики предшествования 319

9.1. Понятие отношений предшествования 319
9.2. Алгоритм типа "перенос-свертка".....322
9.3. Грамматики простого предшествования.....326
9.4. Грамматики слабого предшествования.....339
9.5. Грамматики операторного предшествования.....344

9.6. Язык Флойда-Эванса ... 350
Контрольные вопросы ... 355
Упражнения ... 356

ЧАСТЬ IV. ФОРМАЛЬНЫЕ МЕТОДЫ ОПИСАНИЯ И РЕАЛИЗАЦИИ СИНТАКСИЧЕСКИ УПРАВЛЯЕМОГО ПЕРЕВОДА ... 357

Глава 10. Промежуточные формы представления программ ... 359

10.1. Польская запись ... 360
10.1.1. Вычисление выражений ... 360
10.1.2. Префиксная форма ... 361
10.1.3. Постфиксная форма ... 362
10.1.4. ПОЛИЗ как промежуточный язык ... 365
10.2. Тетрады ... 373
10.2.1. Переменные с индексами ... 373
10.2.2. Указатели функций ... 374
10.2.3. Операторы ... 374
10.3. Триады ... 377
10.4. Байт-коды JVM ... 378
Контрольные вопросы ... 394
Упражнения ... 395

Глава 11. Формальные методы описания перевода ... 396

11.1. Перевод и семантика ... 396
11.2. СУ-схемы ... 398
11.3. Транслирующие грамматики ... 405
11.4. Атрибутные транслирующие грамматики ... 410
11.4.1. Синтезированные атрибуты ... 411
11.4.2. Унаследованные атрибуты ... 413
11.4.3. Определение атрибутной транслирующей грамматики ... 415
11.4.4. Вычисление значений атрибутов ... 419
11.5. Методика разработки описания перевода ... 421
11.6. Пример разработки АТ-грамматики ... 425
Контрольные вопросы ... 435
Упражнения ... 436

Глава 12. Разработка и реализация синтаксически управляемого перевода ... 439

12.1. *L*-атрибутные и *S*-атрибутные транслирующие грамматики ... 439
12.2. Форма простого присваивания ... 440

12.3. Атрибутный перевод для $LL(1)$-грамматик 444
12.3.1. Реализация синтаксически управляемого перевода для транслирующей грамматики 444
12.3.2. L-атрибутный ДМП-процессор 447
12.3.3. Атрибутный перевод методом рекурсивного спуска 455
12.4. S-атрибутный ДМП-процессор 466

12.4.1. Математическая модель восходящего ДМП-процессора 466
Контрольные вопросы 470
Упражнения 470

Список литературы 473

Введение

В учебных планах подготовки дипломированных специалистов, включенных в Государственный образовательный стандарт подготовки студентов по направлениям 654600 — "Информатика и вычислительная техника" (специальность 220400 — "Программное обеспечение вычислительной техники и автоматизированных систем") и 657100 — "Прикладная математика" (специальность 073000 — "Прикладная математика"), бакалавров по направлениям 552800 — "Информатика и вычислительная техника" и 510200 — "Прикладная математика и информатика" значительное место занимают дисциплины, связанные с изучением теории формальных грамматик, языков и автоматов, методов построения языковых процессоров (компиляторов и интерпретаторов).

Учебный материал, положенный в основу данного учебного пособия, читается авторами с середины 70-х годов в Санкт-Петербургском государственном электротехническом университете (ЛЭТИ им. В. И. Ульянова (Ленина)) в рамках таких дисциплин, как "Теория языков программирования и методы трансляции", "Теория вычислительных процессов и структур", "Теория формальных грамматик и автоматов", "Системное программное обеспечение". Учебное пособие отличается от аналогичных книг тем, что содержит систематическое изложение теоретических основ перевода и компиляции, ориентированное на новые образовательные стандарты и может служить основой базового курса по перечисленным ранее дисциплинам. Наиболее полно содержание данного учебного пособия соответствует учебной программе специальной дисциплины "Теория языков программирования и методы трансляции" Государственного образовательного стандарта, который изучается студентами в VII семестре. К этому времени студенты уже освоили основы информатики, дискретной математики, организации ЭВМ, алгоритмические языки и программирование, структуры данных, поэтому изучение материала данного учебного пособия не должно вызвать у них особых затруднений. Однако учебное пособие может быть полезно не только студентам, изучающим дисциплину "Теория языков программирования и методы трансляции", но и студентам других направлений и специальностей, а также тем, кто самостоятельно собирается создавать несложные языки программирования, в том числе и такие, которые используются в системах автоматизации различных прикладных областей, и разрабатывать языковые процессоры для языков программирования различных уровней сложности.

Издание подобного учебного пособия представляется полезным и своевременным, поскольку имеющаяся литература по теоретическим основам построения языковых процессоров ориентирована в основном на более подготовленных читателей. При изложении материала в учебном пособии широко используются формальные описания. Такой подход, с одной стороны, позволяет наиболее полно описывать алгоритмы, а с другой стороны, прививает студентам навыки, необходимые для самостоятельной работы с научной литературой. Материал учебного пособия излагается по принципу "от простого к сложному", все изучаемые понятия связаны в единую систему, изложение теоретического материала чередуется с рассмотрением большого количества примеров для небольших грамматик, описывающих синтаксические конструкции реальных языков программирования, и экспериментальных грамматик с небольшим числом терминалов, нетерминалов и правил вывода. Каждая глава заканчивается списком вопросов для самоконтроля и заданий для самостоятельной работы, способствующих активному усвоению материала учебного пособия.

Учебное пособие может быть использовано для односеместрового или двухсеместрового курса по теоретическим вопросам проектирования языковых процессоров. Оно состоит из введения и двенадцати глав, разделенных на четыре части.

В первой части книги рассматриваются общие вопросы разработки, описания и реализации языков программирования. Это те вопросы, которыми должен владеть любой высококвалифицированный программист, т. к. их глубокое понимание позволяет создавать более надежные и эффективные программы. Для будущих специалистов по разработке языковых процессоров вопросы, рассматриваемые в первой части, позволят получить прочные знания как о математических моделях, на которых основываются языки программирования, так и о методах и способах их реализации.

Первая часть учебного пособия состоит из двух глав. В *главе 1* приводятся основные парадигмы языков программирования и критерии, наиболее часто используемые для критической оценки существующих и проектируемых языков программирования. В этой главе рассматриваются основные конструкции языков программирования: объекты данных и механизмы их типизации, типы и структуры данных, выражения и управляющие структуры. При рассмотрении основных конструкций языков программирования основное внимание уделяется не синтаксису этих конструкций, а их семантике. В *главе 2* обсуждаются основные методы формального описания синтаксиса языков программирования (форма Бэкуса-Наура и ее модификации, синтаксические диаграммы Вирта), рассматриваются вопросы, связанные с описанием контекстных условий языков программирования. Также в этой главе дается представление о наиболее известных методах формального определения динамической семантики: W-грамматике, операционной, аксиоматической и денотационной семантике.

Вторая часть учебного пособия посвящена формальным методам описания синтаксиса языка. В *главе 3* определяются формальные грамматики и формальные языки, приводится классификация формальных грамматик по Хомскому, рассматриваются эквивалентные преобразования КС-грамматик (исключение бесполезных символов, ε-правил, цепных правил, левой рекурсии), неоднозначность грамматик, свойства КС-языков и свойства детерминированных КС-языков. В *главе 4* дается определение распознающего автомата, рассматриваются типы распознающих автоматов и языки, допускаемые распознающими автоматами. Подробно рассматриваются конечные автоматы и преобразователи, способы их задания, связь автоматных грамматик и конечных автоматов, решение проблемы принадлежности, пустоты и эквивалентности для конечных автоматов. В *главе 5* определяются автоматы с магазинной памятью и их разновидности (расширенные МП-автоматы, детерминированные МП-автоматы), преобразователи с магазинной памятью, рассматривается связь между МП-автоматами и КС-грамматиками. В учебном пособии рассматриваются только те аспекты теории автоматов, которые имеют отношение к построению языковых процессоров, поэтому ряд ее понятий опущен, и, следовательно, учебное пособие нельзя использовать как исчерпывающий курс по теории автоматов. Однако студенты, изучившие материал данного учебного пособия, впоследствии значительно проще воспримут курс теории автоматов, и наоборот: знание основ теории автоматов поможет студентам быстрее освоить материал, изложенный в данном учебном пособии.

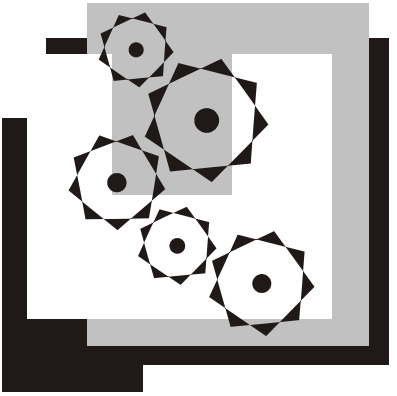
Третья часть учебного пособия посвящена методам синтаксического анализа современных языков программирования. Грамматики, используемые для задания синтаксиса этих языков, позволяют описывать большинство синтаксических конструкций и вместе с тем допускают применение простых детерминированных алгоритмов синтаксического анализа. В *главе 6* даются определение и алгоритмы нисходящего и восходящего разбора, рассматриваются вопросы моделирования работы недетерминированных преобразователей с магазинной памятью, лежащих в основе нисходящих и восходящих синтаксических анализаторов, приводится малоизвестный детерминированный алгоритм Эрли, позволяющий построить правый разбор для входной цепочки, порождаемой произвольной КС-грамматикой. В *главе 7* рассматриваются алгоритмы разбора для $LL(1)$-грамматик ("1-предсказывающий" алгоритм и метод рекурсивного спуска), в *главе 8* — алгоритм разбора "перенос-свертка" и его использование для некоторых подклассов $LR(k)$-грамматик ($LR(0)$- и $SLR(1)$-грамматик). В *главе 9* рассматривается использование алгоритма "перенос-свертка" для грамматик предшествования (простого, слабого, операторного). В конце главы приводится описание языка Флойда-Эванса, ориентированного на разработку синтаксических анализаторов, и рассматривается пример применения этого языка для построения детерминированного восходящего анализатора. При отборе и изложении материала *глав 7—9* большое внимание было уделено не просто методам

синтаксического разбора, а возможности их расширения для описания синтаксически управляемого атрибутного перевода с помощью атрибутных транслирующих грамматик.

В четвертой части учебного пособия излагаются формальные методы описания и реализации синтаксически управляемого перевода. *Глава 10* посвящена промежуточным формам представления программы после этапа синтаксического анализа. В ней рассматриваются как традиционные (польская инверсная запись, тетрады, триады, косвенные триады), так и современные (byte-code JVM) формы представления. В *главе 11* дается формальное определение перевода. В качестве формальных моделей описания перевода рассматриваются схемы синтаксически управляемого перевода, транслирующие грамматики и атрибутные транслирующие грамматики. Приводится методика разработки описания перевода и пример использования этой методики для построения атрибутной транслирующей грамматики. В *главе 12* рассмотрены вопросы разработки и реализации синтаксически управляемого перевода. В этой главе описано, каким образом нужно доработать рассмотренные в третьей части учебного пособия алгоритмы нисходящего и восходящего разбора и каким требованиям при этом должна удовлетворять атрибутная транслирующая грамматика, чтобы можно было выполнять синтаксически управляемый перевод заданным методом.

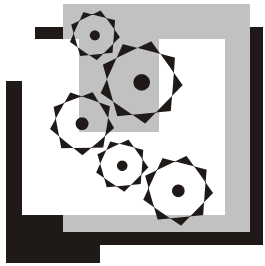
Введение, *главы 1*, *2*, *4*, *5*, *12*, *разд. 7.5*, *10.4*, *11.5—11.6* написаны Эльвирой Александровной Опалевой, а *главы 3*, *6*, *8*, *9*, *разд. 7.1—7.4*, *10.1—10.3*, *11.1—11.4* — Владимиром Петровичем Самойленко. Общее редактирование учебного пособия выполнено Э. А. Опалевой.

В заключение авторы хотят выразить признательность своим родным за их помощь и поддержку в течение всего времени работы над книгой, благодаря которым и смог появиться на свет этот труд. Также хочется выразить благодарность сотрудникам издательства "БХВ-Петербург" Елене Кондаковой, Людмиле Еремеевской и Нине Седых за их терпеливое и доброжелательное отношение в процессе подготовки текста учебного пособия и его корректуры, которые авторам не всегда удавалось делать в срок.



Часть I

Языки программирования



Глава 1

Основные концепции языков программирования

Любую систему обозначений и согласованную с ней систему понятий, которую можно использовать для описания алгоритмов и структур данных, в первом приближении можно считать языком программирования. Однако в настоящем учебнике речь пойдет только об универсальных языках программирования, которые нашли широкое применение при разработке программ в различных областях человеческой деятельности. Каждый из этих языков создавался для определенных целей и имеет свои достоинства и недостатки. Для того чтобы эффективно использовать и реализовывать языки программирования, необходимо хорошо знать фундаментальные понятия, лежащие в основе их построения.

Знание концептуальных основ языков программирования с точки зрения использования и реализации базовых языковых конструкций позволит:

- более обоснованно выбрать язык программирования для реализации конкретного проекта;
- разрабатывать более эффективные алгоритмы;
- систематически пополнять набор полезных языковых конструкций;
- ускорить изучение новых языков программирования;
- использовать полученные знания как методологическую основу для разработки новых языков программирования;
- получить базовые знания, необходимые для разработки трансляторов для языков программирования, поддерживающих разные вычислительные модели.

Используемые программистами языки программирования отличаются как своим синтаксисом, так и функциональными возможностями. Различия в синтаксисе играют незначительную роль при изучении концептуальных основ языка программирования, в то время как наличие или отсутствие тех или иных функциональных возможностей существенно влияет на реализацию и область применения языка.

1.1. Парадигмы языков программирования

На сегодняшний день имеются четыре основные парадигмы языков программирования [36, 38], отражающие вычислительные модели, с помощью которых описывается большинство существующих методов программирования:

- императивная;
- функциональная;
- декларативная;
- объектно-ориентированная.

1.1.1. Императивные языки

Императивные (*процедурные*) *языки* — это языки программирования, управляемые командами, или операторами языка. Основной концепцией императивного языка является состояние компьютера — множество всех значений всех ячеек (слов) памяти компьютера. Данная модель вытекает из особенностей аппаратной части компьютера стандартной архитектуры, названной "фон-неймановской" в честь одного из ее авторов — американского математика Джона фон Неймана. В таком компьютере данные, подлежащие обработке, и программы хранятся в одной памяти, называемой *оперативной*. Центральный процессор получает из оперативной памяти очередную команду на входном языке, декодирует ее, выбирает из памяти указанные в качестве операндов входные данные, выполняет команду и возвращает в память результат.

Программа на императивном языке представляет собой последовательность команд (операторов), которые выполняются в порядке их написания. Выполнение каждой команды приводит к изменению состояния компьютера. Основными элементами императивных языков программирования, ориентированных на фон-неймановскую архитектуру, являются переменные, моделирующие ячейки памяти компьютера, и операторы присваивания, осуществляющие пересылку данных. Выполнение оператора присваивания может быть представлено как последовательность обращений к ячейкам памяти за операндами выражения из правой части оператора присваивания, передача их процессору, вычисление выражения и возвращение результата вычисления в ячейку памяти, представляющую собой переменную из левой части оператора присваивания.

Императивные языки поддерживают, как правило, один или несколько итеративных циклов, различающихся синтаксисом. Итеративные циклы в фон-неймановской архитектуре выполняются быстро за счет того, что команды программы хранятся в соседних ячейках памяти компьютера.

Большинство императивных языков включает в себя конструкции, позволяющие программировать рекурсивные алгоритмы, но их реализация на

компьютерах с фон-неймановской архитектурой не эффективна, что связано с необходимостью программного моделирования стековой памяти.

К императивным языкам относятся такие распространенные языки программирования, как ALGOL-60 [1], BASIC [15, 51], FORTRAN [13, 14], PL/1 [56], Ada [16], Pascal [19, 21], C [57], C++ [53], Java [33, 60].

1.1.2. Языки функционального программирования

В *языках функционального программирования* (*аппликативных* языках) вычисления в основном производятся путем применения функций к заданному набору данных. Разработка программ заключается в создании из простых функций более сложных, которые последовательно применяются к начальным данным до тех пор, пока не получится конечный результат. Типичная программа, написанная на функциональном языке, имеет следующий вид:

$$функция_n\ (\ ...\ функция_2\ (функция_1\ (данные))\ ...\ ).$$

На практике наибольшее распространение получили язык функционального программирования LISP [50, 62] и два его диалекта: язык Common LISP [64] и язык Scheme. Основной структурой данных языка LISP являются связные списки, элементами которых могут быть либо атомы (идентификаторы или числовые константы), либо другие связные списки. При этом список

$$(K\ L\ M\ N)$$

в терминах данных интерпретируется как список из четырех элементов *K*, *L*, *M*, *N*, а в терминах программ — как функция *K* с аргументами *L*, *M* и *N*.

Несмотря на то, что многие ученые в области компьютерных наук указывают на преимущества языков функционального программирования по сравнению с императивными, языки функционального программирования не получили широкого распространения из-за невысокой эффективности реализации их на компьютерах с фон-неймановской архитектурой. На компьютерах с параллельной архитектурой трансляторы для функциональных языков реализуются более эффективно, однако они еще не конкурентоспособны по сравнению с реализациями императивных языков.

Кроме языка LISP, основной областью применения которого являются системы искусственного интеллекта, известны и другие языки функционального программирования: ML (MetaLanguage) [58, 63], Miranda и Haskell [37, 39]. Язык ML наряду с функциональным программированием поддерживает императивное программирование, но, в отличие от императивных языков, функции в языке ML могут быть полиморфными и передаваться между подпрограммами как параметры.

Замечание

Программирование как на императивных, так и на функциональных языках является *процедурным*. Это означает, что программы на этих языках содержат указания, *как* нужно выполнять вычисления.

1.1.3. Декларативные языки

Декларативные языки программирования — это языки программирования, в которых операторы представляют собой объявления или высказывания в символьной логике. Типичным примером таких языков являются *языки логического программирования* (языки, основанные на системе правил).

В программах на языках логического программирования соответствующие действия выполняются только при наличии необходимого разрешающего условия. Программа на языке логического программирования схематично выглядит следующим образом:

разрешающее условие 1 → *последовательность операторов 1*

разрешающее условие 2 → *последовательность операторов 2*

...

разрешающее условие n → *последовательность операторов n*

В отличие от императивных языков, операторы программы на языке логического программирования выполняются не в том порядке, как они записаны в программе. Порядок выполнения операторов определяется системой реализации правил.

Характерной особенностью декларативных языков является их *декларативная семантика*. Основная концепция декларативной семантики заключается в том, что смысл каждого оператора не зависит от того, как этот оператор используется в программе. Так, смысл заданного высказывания в языке логического программирования можно точно определить по самому оператору. Декларативная семантика намного проще семантики императивных языков, что может рассматриваться как преимущество декларативных языков над императивными.

Наиболее распространенным языком логического программирования является язык Prolog [40, 59]. Основными областями применения языка Prolog являются экспертные системы, системы обработки текстов на естественных языках и системы управления реляционными базами данных. К языкам программирования, основанным на системе правил, можно отнести языки синтаксического разбора (например, YACC — Yet Another Compiler Compiler), в которых синтаксис анализируемой программы рассматривается в качестве разрешающего условия.

1.1.4. Объектно-ориентированные языки

Концепция объектно-ориентированного программирования складывается из трех ключевых понятий: *абстракция данных*, *наследование* и *полиморфизм*.

Абстракция данных позволяет *инкапсулировать* множество объектов данных (члены класса) и набор абстрактных операций над этими объектами данных