1 ЭССЕ ПО КНИГЕ ”КОНЦЕПЦИИ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ”

1.1 Предисловие

Хотелось бы сразу описать формат в котором я буду излагать свои знания о материале книги. Так как материал книги довольно большой (более 800 страниц), то логичным представляется вариант краткой сводки-рецензии по каждой из глав книги. Таким образом, в каждом разделе будет описано самое по-моему мнению интересное и полезное в отношении проекта и в целом.

Я читал книгу на английском языке, поэтому буду стараться правильно переводить термины в соответствии с текущим консенсусом в индустрии. Во избежание неясностей также будет приводится оригинальное название термина.

1.2 Глава 1. Введение

В этой главе описывается предназначение книги, а также изучения языков в целом. Описывается общая структура языков программирования и их связь с остальными компонентами индустрии ПО. Из интересного, в главе расписываются выдвигаемые авторами критерии по оценке языков программирования.

Создается небольшая матрица соотношения характеристик языков и их ”основных” свойств:

- Удобство чтения программ
- Удобство написания программ
- Надежность

Эта матрица представлена на рисунке 1.1

В целом, она, хоть и неформально, описывает некоторые аспекты, которые могут влиять на важные свойства языка. К примеру, слабая ортогональность в дизайне языка будет соответствовать языку с малой выразительностью

	CRITERIA		
Characteristic	READABILITY	WRITABILITY	RELIABILITY
Simplicity	•	•	•
Orthogonality	•	•	•
Data types	•	•	•
Syntax design	•	•	•
Support for abstraction		•	•
Expressivity		•	•
Type checking			•
Exception handling			•
Restricted aliasing			•

Рисунок 1.1 – Зависимость характеристик языков от субъективных показателей

при большом количестве конструкций. Однако, высокая ортогональность будет сложна для восприятия человеком.

Также, в главе описаны подходы к построению языков и почему развитие языков выглядит именно так как выглядит — описывается компьютерная архитектура и структура типичного компилятора, а также описывается процесс вычисления в общем виде.

1.3 Глава 2. Эволюция языков

Глава описывает относительно краткую историю развития языков программирования с наиболее старых до современных (на момент книги, т.е. примерно 2003 год). Хорошо описаны ранние языки, такие как `Fortran` или `Cobol`. Также, глава построена таким образом, чтобы продемонстрировать разные категории языков из разных областей.

Следует отметить, что ввиду возраста книги в ней не отражены современные мультипарадигменные языки, вобравшие в себя опыт императивных ООП языков, популярных в индустрии (`C++, Java, C#, PHP`) — например `Golang, Rust` или современный `JavaScript`

1.4 Глава 3. Описание синтаксиса и семантики

В главе излагается небольшое количество классической теории, позволяющей описывать синтаксис и семантику ЯП. Для описания синтаксиса ЯП излагается структура `BNF` и `EBNF`. Для описания семантики используются следующие формализмы:

- Атрибутные грамматики
- Операционная семантика
- Денотационная семантика
- Аксиоматическая семантика

1.5 Глава 4. Лексический и синтаксический анализ

Глава описывает стандартные техники лексического и синтаксического анализа:

1. Разбиение на лексемы и создание ленивого потока
2. Парсинг
    а) Top-down (рекурсивный спуск) парсеры
    б) Bottom-up парсеры

Описываются также принципы работы табличных парсеров и генераторов парсеров. Рассматриваются грамматики `LL` и `LR`, а также соответствующие подходящие им парсеры (рекурсивный спуск и табличный соответственно)

1.6 Глава 5. Имена, привязка имен и области видимости

Эта глава одна из самых интересных в том отношении, что авторы вводят понятие ”времени связывания” для объяснения семантики языков. К примеру, семантика числа `3` или ключевого слова `while` вводится авторами в ”language design time”, а семантика например пользовательской переменной `i` становится известна только во время исполнения, так как значение в ней хранящееся заранее неизвестно.

Также, в главе хорошо излагается структура такой программной сущности как ”переменная”. По словам авторов, она состоит из шести компонентов:

1. Имя
2. Адрес в памяти
3. Значение
4. Тип
5. Время жизни
6. Область видимости

Такая богатая классификация действительно позволяет описать семантику почти любого языка с переменными. Авторы раскрывают каждый аспект в отдельности и такое грамотное разделение позволяет проще описывать семантически сложные языки, например C.

1.7 Глава 6. Типы данных

В отличие от книги Б. Пирса (которую я тоже изучаю в рамках проекта) здесь описание типов можно характеризовать несколькими аспектами:

- Типы объясняются в первую очередь как данные в памяти
- К каждому типу прилагается анализ семантики в отношении проблем дизайна языка
- Не проводится чёткой классификации (в отличие от типов в теоркате)

В целом, глава очень богатая так как содержит много практической информации с учетом опыта индустрии в отношении тех или иных типов данных.

1.8 Глава 7. Выражения и присваивания

В целом, глава довольно разнородна, так как помимо обычных, универсальных для языков характеристик выражений (приоритеты операторов, наличие side-effects, short-circuit evaluation) добавляются другие, специфичные для определенных языков. Например, перегрузка операторов (ad-hoc polymorphism) неявные приведения (coercions) и присваивания смешанного типа (mixed-modes assignments). Такая информация больше зависит от выбранной системы типов.

1.9 Глава 8. Управляющие конструкции

В главе описываются принципы реализации стандартных управляющих конструкций: выбора и итерации. В отношении `if` описывается стандартная проблема dangling else при использовании `C`-подобной грамматики. Из необычного, описываются не ставшие популярными конструкции Дейкстры: guarded conditional и guarded iteration. Такие конструкции приводятся как пример упрощения семантики с получением возможности формальной верификации.

1.10 Глава 9. Подпрограммы

Эта глава одна из самых больших, так как в любом языке подпрограммы (а также функции, процедуры, методы, замыкания и другие аналоги) являются одним из основных блоков для создания программ. В главе описываются все стандартные аспекты подпрограмм как с точки зрения функциональности (от подпрограмм как вставок кода к обобщенным функциям) так и с точки зрения сложности реализации (пролог функции, функции как объекты первого класса, замыкания)

В конце главы рассмотрены корутины как вариант асинхронного исполнения кода.

1.11 Глава 10. Реализация подпрограмм

В данной главе рассмотрены основные сложности реализации функций, разве что не упомянут TCO или функциональные стили передачи управления (например CPS)

В целом, по данной и предыдущей главе, можно сказать что эта книга слабо раскрывает аспекты реализации функций в стиле функциональных языков, предпочтение отдается в первую очередь императивному подходу.

1.12 Глава 11. Абстрактные типы данных

В общем виде глава не представляет настолько сильного интереса, как предыдущие так как объясняются стандартные техники сокрытия и инкапсуляции из конвенциональных языков программирования. Часть объясняемой семантики связана с типами, другая часть с областями видимости, а третья с управлением состоянием (state).

Нельзя сказать, что глава бесполезна, но в ней не хватает явного разделения концепций между собой.

1.13 Глава 12. ООП

В данной главе излагаются общие семантические аспекты присущие ООП языкам: полиморфизм через наследование, динамическое связывание методов и отношение подкласс-подтип. Хорошо описаны разница между ООП семантикой ”подкласс” и семантикой ”подтип” которая встречается в других языках. Объяснена классическая ”diamond problem” и способы решения этой проблемы в различных языках.

Из недостатков можно сказать, что всколзь упомянуты мультиметоды и вообще не упомянуты ”polymorphism a la carte”, которые активно используются в других языках: `Closure, Scala, JavaScript`. Предположительно это связано с возрастом книги.

1.14 Глава 13. Конкуррентность

По конкуррентности (concurrency) пишут целые книги, поэтому данная глава является лишь верхушкой айсберга в отношении количества информации. Тем не менее, здесь описываются наиболее популярные модели конкуррентности, а именно:

- Примитивы синхронизации в сочетании с тредами
  - Семафоры Дейкстры
  - Мониторы
- Синхронная передача сообщений (рандеву, на примере `Ada`)

– Параллельное выполнение

К сожалению, в главе очень не хватает современных подходов к конкуррентности, например `Promise` из `JavaScript` или `Fibers` из `Golang`. Также, практически никакой информации о других моделях конкуррентности, например `STM`. Но в целом, представленная информация доходчиво описывает существующие модели и их семантику.

1.15 Глава 14. Исключения и события

В целом, глава хорошо описывает принципы дизайна исключений и событий в языке. Рассмотрены разные языки с исключениями и их отличия, которые влекут за собой определенные плюсы и минусы. В отношении событий, рассмотрен базовый случай реактивного UI с привязкой действий к UI элементам.

Можно сказать, что в целом семантически и то и другое является частным случаем передачей управления, поэтому фундаментальной информации в главе меньше чем практической.

1.16 Глава 15. Функциональные языки

Глава кратко и по верхам раскрывает основные аспекты функциональных языков на примере `Scheme`. Также, упомянаются `Haskell` и `ML`. Основные концепции излагаются, но в рамках практического обозрения, без серьезного затрагивания теории. Например, лямбда исчисление приводится как нотация, не более.

В целом, по функциональным языкам тоже пишут целые книги и курсы, поэтому логично, что глава не имеет большого количества информации. Для тех целей которые ставятся в книге, однако, описание языков достаточное. Но конечно не хватает большей теоретической обоснованности.

1.17 Глава 16. Логические языки

Как и ожидалось, в главе излагаются математические основы логических языков и все примеры приведены на языке `Prolog`. Так как рассматриваемый язык по-сути один, то плюсом главы является её доходчивость и практическая применимость. Минусом же является узкая выборка языков.

В главе рассматриваются основные проблемы логических языков и их преимущества в реальных областях (например в СУБД или экспертных системах).

1.18 Вывод

Если описать прочитанное в нескольких предложениях, то они будут такими — книга хорошо подходит для начинающего дизайнера языков так как имеет много примеров ”do” и ”don’t” в отношении функиональных особенностей языков. Однако, возраст книги и её упор на популярные императивные языки строит несколько однобокую картину сферы ЯП. При дальнейшем изучении стоит дополнить книгу более углубленной литературой по другим языкам и/или другим концепциям языков.