Практическое упражнение по сравнению языков программирования

Алексей Недоря, Евгений Кандзюба, Дмитрий Соломенников, Дмитрий Кузнецов

Исходный вопрос: Является ли Тривиль Обероном на кириллице?

Алексей Недоря (АН):

Хороший вопрос, но для того, чтобы ответить на него, надо понять, что такое Оберон с точки зрения сущностных характеристик. Понятно, что смотреть надо не на синтаксис, он важен, но вторичен. Смотреть надо на семантику - статическую и динамическую. Если мы выпишем сущностные характеристики Оберона, скажем 5-10 штук, и, после этого, сопоставим их с характеристиками Тривиля, то мы сможем обоснованно сказать - Тривиль - это Оберон, почти Оберон, совсем не Оберон.

Домашнее задание тем, кто хочет разрабатывать языки программирования: напишите в комментарии сущностные характеристики Оберона. Подсказка, одна из таких характеристик - сборка мусора.

Евгений Кандзюба (ЕК):

Сущностные характеристики языка Оберон:

1. Простота и минимализм: Оберон разработан с акцентом на минимализм. Язык обладает небольшим количеством ключевых слов и простым синтаксисом, что упрощает его изучение и использование. Отсутствие избыточных и сложных конструкций делает код более понятным и поддерживаемым.

2. Модульность: Язык поддерживает модульное программирование. Программы в Обероне разделяются на отдельные модули, что способствует структурированию кода, облегчает его сопровождение и повторное использование.

3. Строгая типизация: Оберон является статически типизированным языком. Типы переменных определяются во время компиляции, что позволяет выявлять и исправлять ошибки на ранних этапах разработки.

4. Расширяемые записи (Объектно-ориентированные элементы): Оберон вводит концепцию расширяемых записей, позволяя создавать иерархии типов. Это обеспечивает основы полиморфизма и наследования, характерные для объектно-ориентированного программирования, хотя язык не является полностью объектно-ориентированным.

5. Динамическое распределение памяти и сборка мусора: Оберон поддерживает динамическое распределение памяти через указатели и автоматическую сборку мусора. Это облегчает работу с динамическими структурами данных и снижает риски утечки памяти.

6. Строгий контроль доступа: Модули в Обероне могут определять интерфейсы с четко

обозначенными импортируемыми и экспортируемыми сущностями. Это обеспечивает инкапсуляцию и контроль доступа к данным и функциям.

7. Компиляция в машинный код: Оберон компилируется непосредственно в машинный код, что обеспечивает высокую производительность создаваемых программ. Компилятор Оберона известен своей скоростью и эффективностью.

8. Тесная интеграция с операционной системой Оберон: Язык и операционная система Оберон были разработаны одновременно, что обеспечивает гармоничную работу программ с системой. Это позволяет использовать ресурсы системы более эффективно.

9. Портативность: Благодаря своей простоте и стандартизации, программы на Обероне могут быть легко перенесены на разные аппаратные платформы и операционные системы с минимальными изменениями.

10. Поддержка системного и прикладного программирования: Оберон пригоден для разработки как системного программного обеспечения (например, операционных систем), так и прикладных программ. Это достигается за счет наличия низкоуровневых возможностей без ущерба для высокого уровня абстракции.

11. Отсутствие избыточных функциональностей: В языке намеренно отсутствуют некоторые особенности, присущие другим языкам программирования (например, перегрузка операторов, множественное наследование), чтобы избежать сложности и неоднозначности.

12. Библиотека стандартных модулей: Оберон поставляется с набором стандартных модулей, предоставляющих базовый функционал для работы со строками, файлами, вводом/выводом и прочими распространенными задачами.

13. Сильная поддержка академического сообщества: Язык широко используется в образовательных и исследовательских целях, что способствует его развитию и поддержке со стороны академических учреждений.

14. Лексический и синтаксический контроль: Строгие правила написания кода и именования обеспечивают единый стиль программирования и облегчают чтение и понимание кода.

15. Простая семантика: Язык избегает сложных семантических конструкций, что упрощает анализ и понимание поведения программ, а также способствует надежности и предсказуемости кода.

Эти характеристики делают Оберон уникальным в мире языков программирования, сочетая в себе простоту и мощность. Он предоставляет разработчикам инструменты для создания эффективных и надежных программ, одновременно способствуя лучшему пониманию фундаментальных концепций программирования.

Дмитрий Соломенников (ДС):

Хороший анализ. В принципе, через подобный список нужно пропускать любой разрабатываемый язык.

АН:

Хороший анализ, но надо серьезно доработать. Следите за руками.

Во-первых, уберем то, что не относится к языку, а это пункты 7, 8 и 13 и уберем п.14, который ничего не дает для анализа, все языки строго определяют лексику и синтаксис.

Далее разделим пункт 1 на а) простота и б) минимализм. Теперь очевидно надо склеить а) и 15 - это простота, потом б) и 11 - это минимализм, и склеиваем 2 и 6 - это модульность.

Далее пункт 9 тоже ничего не дает, переносимость Оберона ничем не отличается от переносимости остальных языков. Пункт 10 ничего не дает, C++ (и большинство других языков) тоже можно использовать для системного и прикладного программирования, и что?
П.12 - тоже не совсем про язык, хотя и про язык тоже, но я бы (стыдливо) убрал этот пункт, чтобы не говорить о плохом.

Остается:
1) Простота
2) Модульность
3) Строгая типизация
4) Расширяемые записи
5) Сборка мусора
6) Минимализм

Теперь подумаем о простоте. Мы говорим о простоте для кого? Что проще новый автомобиль в который сел и поехал или велосипед, у которого надо накачать шины и проверить тормоза, прежде чем поехать. Для кого Оберон точно простой? Например, для разработчика компилятора. Вот у меня открыт Oberon report (Revision 1. 10. 90) и он размера 16 страниц. Это дает нам численную оценку, которую я бы сформулировал так: Объем документации, достаточный для написания компилятора.

Минимализм - это очень неоднозначно, тем более: but not simpler. Где так граница до которой можно, а дальше нельзя? Меньше это лучше, или хуже?

Итого, я бы оставил пункты, пригодные для объективного сравнения:
1) Простота для разработчика компилятора (измеряется в страницах спецификации)
2) Модульность
3) Строгая, статическая типизация
4) Расширяемые записи (частичная поддержка ООП)
5) Сборка мусора

Хорошо, но не полностью описывает Оберон. Я бы добавил, как минимум, еще 3 сущностные характеристики. Есть предложения?

ДС - парадигма структурного программирования:

Оберон тяготеет к структурному программированию больше, чем к другим парадигмам.

АН:

Как из этого сделать характеристику?

ДС - Читаемость:

Я бы еще добавил читаемость преобладает над удобством написания.

АН:

Читаемость - это хороший пункт, хотя скорее субъективный, чем объективный. Впрочем, здесь можно опереться на явные заявления авторов языка. Например, readability first - Тривиль, Carbon (что очень странно, но так написано, см. wiki/Carbon), writability first - Go, Swift. Так что да, можно добавить.

Но есть еще минимум 3. Дам подсказку, мы же собираемся сравнивать языки, значит, надо думать о характеристиках, по которым можно разделять языки на группы.
Например, типизация - статическая, динамическая , структурная (Typescript), статическая с элементами динамики, gradual typing и т.д. Или управление памятью - ручное, сборка мусора, ARC, borrowing, ownership, ...

Дмитрий Кузнецов (ДК):

П.12 - тоже не совсем про язык, хотя и про язык тоже, но я бы (стыдливо) убрал этот пункт, чтобы не говорить о плохом.

Не стал бы убирать, потому что из этого пункта следует, что надо смотреть на интероп с другими языками.

Тогда и практически полезная «мощность» языка будет подрастать за счёт других экосистем.

Это хорошо видно по Расту, который пылесосит Си, по Котлину встроенному в инструментарии Явы, думаю, что так код макОС на Свифт плавно перезжал.

АН - Интероп:

Интероп - как сформулировать и как сравнивать языки?

АН - Первый заход на сравнение:

Характеристика	**Оберон**	**Тривиль**
Простота (число страниц спеки)	26 страниц The Programming Language Oberon-2	43 страницы Язык программирования Тривиль (0.9.4)
Модульность	Да	Да
Типизация	строгая, статическая	строгая, статическая
Поддержка ООП	частичная - расширяемые записи	частичная класса на минималках
Управление памятью	Сборка мусора	Сборка мусора

Читаемость важнее	Да	Да

Глядя на эту таблицу можно сказать Тривиль - это Оберон. Но мы не посмотрели со стороны Тривиля, и тут надо ответить на вопросы

1. Что есть в Тривиле из того, чего нет в Обероне?
2. Что есть в обоих языках и сущностно отличается?

В зависимости от ответов мы можем получить, что один язык является подмножеством другого или что они существенно отличаются.

Безопасность ссылок (null safety)	Нет	Да
Обобщенное программирование	Нет	Да
Утиная типизация	Нет	Да
Явные указатели	Да	Нет
Типы - значения (value types)	Да	Нет

Если посмотреть на получившийся набор из 10 сущностных характеристик, то мой ответ: Оберон и Тривиль **сущностно разные языки**.

Но что интересно, мы получили очень интересный базис для сравнения не только этих двух языков.

АН - добавим Go к сравнению:

№	Характеристика	Оберон	Go	Тривиль
1	Читаемость важнее	Да	Нет	Да
2	Простота (число страниц спеки)	26 страниц The Programming Language Oberon-2	116 страниц The Go Programming Language Specification (v1.23)	43 страницы Язык программирования Тривиль (0.9.4)
3	Модульность	Да	Да	Да
4	Типизация	строгая, статическая	строгая, статическая	строгая, статическая
5	Поддержка ООП	частичная - расширяемые записи	частичная - композиция	частичная класса на минималках
6	Управление	Сборка мусора	Сборка мусора	Сборка мусора

	памятью			
7	Безопасность ссылок (null safety)	Нет	Нет	Да
8	Обобщенное программирование	Нет	Да (с версии 1.18)	Да
9	Утиная типизация	Нет	Да	Да
10	Явные указатели	Да	Да	Нет
11	Типы - значения (value types)	Да	Да	Нет
	Одинаковость с Обероном (покрашено)		6/11	5/11

Замечание: число страниц для Go получено сохранением в PDF текста на вебе.

При подсчете одинаковости:

- Go - размер спеки существенно разный (0)
- Тривиль - размер спеки несущественно разный (+1)
- Поддержка ООП во всех языках есть, и во всех существенно разная, поэтому (0)

Интересно, что в отличие от Тривиля, Go, если не учитывать разный подход к ООП, является над-множеством Оберона. В 9 пунктах Go или такой же, как Оберон, или шире.

АН - что дальше?

Можно попытаться обобщить таблицу, добавив другие сущностные характеристики, например, для многих языков сущностной характеристикой является unified type system.

ЕК: сравнительная таблица по итогу анализа семантических особенностей языков с категоризацией по сущностям

Категория	Характеристика	Oberon	Тривиль
Дизайн языка	Простота (число страниц спеки)	26	43
Дизайн языка	Происхождение	Семейство Паскаль	Современные языки (Go, Swift, Kotlin, Oberon)
Дизайн языка	Основной синтаксис	Английский	Русский

Дизайн языка	Цель	Системное программирование	Разработка компиляторов и общее назначение
Структура программы	Модульность	Да	Да
Структура программы	Импорт/Экспорт	Явный	Явный
Структура программы	Инициализация модуля	Последовательность операторов после BEGIN	Блок 'вход'
Система типов	Типизация	Строгая статическая	Строгая статическая
Система типов	Базовые типы	BOOLEAN, CHAR, INTEGER, REAL, SET	Лог, Символ, Цел64, Вещ64, Байт, Слово64, Строка, Строка8
Система типов	Массивы	Фиксированные и открытые	Только динамические (векторы)
Система типов	Записи/Классы	RECORD	класс
Система типов	Указатели	Явные (POINTER TO)	Нет явных указателей
Система типов	Обобщенное программирование	Нет	Да (обобщенные модули)
Система типов	Утиная типизация	Нет	Да (для протоколов)
Система типов	Может быть тип (nullable)	Нет	Да (мб T)
ООП	Наследование	Расширение записей	Наследование классов
ООП	Полиморфизм	Через процедурные переменные	Через протоколы и наследование
ООП	Инкапсуляция	Частичная (экспорт из модулей)	Да (приватные поля классов)
Управление памятью	Сборка мусора	Да	Да
Управление памятью	Явные указатели	Да	Нет
Функции и процедуры	Синтаксис объявления	PROCEDURE	фн

Функции и процедуры	Параметры	По значению и VAR	Входные (:) и выходные (:=)
Функции и процедуры	Возврат значения	RETURN	вернуть
Функции и процедуры	Функции как значения	Нет	Да (тип функции)
Управление потоком	Условный оператор	IF ... THEN ... ELSIF ... ELSE ... END	если ... { ... } иначе если ... { ... } иначе { ... }
Управление потоком	Цикл с предусловием	WHILE ... DO ... END	пока ... { ... }
Управление потоком	Цикл со счетчиком	FOR ... TO ... BY ... DO ... END	цикл ... среди ...
Управление потоком	Оператор выбора	CASE ... OF ... END	выбор ... { когда ... : ... }
Безопасность	Проверка границ массивов	Во время выполнения	Во время выполнения (авария)
Безопасность	Обработка исключений	Нет	Да (оператор 'авария')
Дополнительные возможности	Многопоточность	Нет упоминания	Нет упоминания
Дополнительные возможности	Вложенные процедуры	Да	Нет
Дополнительные возможности	Полиморфные параметры	Нет	Да (*)
Дополнительные возможности	Вариативные параметры	Нет	Да (...)

Тривиль и Оберон

1. "Тривиль сохраняет многие концепции Oberon":
   - Оба языка используют строгую статическую типизацию
   - Оба являются модульными языками с явным экспортом/импортом
   - Оба поддерживают объектно-ориентированное программирование
   - Оба используют сборку мусора для управления памятью
   - Многие базовые конструкции (условные операторы, циклы) концептуально схожи

2. "Тривиль представляет собой более современный язык":
   - Тривиль включает современные концепции, такие как обобщенное программирование и

утиная типизация
- В Тривиле есть понятие "может быть тип" (nullable types), что отражает современные тенденции в обеспечении безопасности кода
- Тривиль поддерживает Unicode, что важно для современных приложений

3. "С рядом дополнительных возможностей":
- Обобщенные модули в Тривиле
- Поддержка протоколов (аналог интерфейсов в других языках)
- Полиморфные и вариативные параметры
- Функции как значения первого класса

4. "Уникальным русскоязычным синтаксисом":
- Все ключевые слова и основные конструкции в Тривиле на русском языке, что уникально для языков программирования такого уровня

5. "Некоторыми упрощениями":
- Отсутствие явных указателей в Тривиле упрощает работу с памятью и повышает безопасность
- Тривиль имеет только динамические массивы (векторы), что упрощает работу с коллекциями данных

6. "Более богатые возможности для обобщенного и функционального программирования":
- Обобщенные модули в Тривиле позволяют создавать обобщенный код, чего нет в Oberon
- Функции как значения и функциональные типы в Тривиле поддерживают функциональное программирование
- Протоколы в Тривиле обеспечивают большую гибкость в дизайне интерфейсов

Таким образом, Тривиль действительно можно рассматривать как эволюцию идей Oberon, адаптированную к современным требованиям программирования, с акцентом на безопасность, выразительность и удобство использования, особенно для русскоязычных разработчиков. При этом язык сохраняет основные принципы Oberon, такие как простота и модульность.

Тривиль и системное программирование

При этом, в отличии от Oberon, Тривиль нельзя рассматривать как инструмент системного программирования. Пример: сборка ядра Linux.

На основе представленных спецификаций языка Тривиль, можно сделать вывод, что собрать ядро Linux на Тривиле в его текущем виде было бы крайне затруднительно, если не невозможно. Вот несколько ключевых причин:

1. Отсутствие низкоуровневого управления памятью:
- Тривиль не предоставляет прямого доступа к памяти через указатели, что критично для разработки операционных систем. Ядро Linux требует точного контроля над размещением данных в памяти.

2. Автоматическое управление памятью:
- Тривиль использует сборку мусора, что неприемлемо для ядра ОС, где требуется предсказуемое и контролируемое управление ресурсами.

3. Отсутствие поддержки ассемблерных вставок:
   - В спецификации Тривиля нет упоминания о возможности использования встроенного ассемблера, что необходимо для низкоуровневых операций в ядре.

4. Ограниченный контроль над аппаратурой:
   - Язык не предоставляет средств для прямого взаимодействия с аппаратным обеспечением, что критично для ядра ОС.

5. Отсутствие поддержки многопоточности:
   - В спецификации не упоминается поддержка многопоточности, которая необходима для современных ОС.

6. Ориентация на высокоуровневое программирование:
   - Тривиль разработан для создания компиляторов и решения задач общего назначения, а не для системного программирования.

7. Зависимость от системы выполнения:
   - Язык, вероятно, требует собственной системы выполнения, что несовместимо с разработкой ядра ОС.

Хотя Тривиль и имеет некоторые черты, унаследованные от Oberon, который был использован для создания операционных систем, он не предназначен для такого низкоуровневого программирования, которое требуется для разработки ядра Linux.

Для возможности разработки ядра ОС, Тривиль потребовал бы значительных изменений и расширений, включая добавление возможностей работы с "голым железом", точного управления памятью, и удаления зависимости от сборщика мусора.

АК (Александр Когтенков)

Характеристики языка, отсутствующие в Тривиль/Оберон, но использующиеся в других языках подобного семейства:

1. Атрибуты, расширяющие или уточняющие статические проверки и динамическое поведение. Например, в Java позволяют уточнять типы, в C# могут говорить, как использовать методы во время выполнения, в Ada могут добавлять характеристики, в т.ч. закрывая выше упомянутый вопрос доступа по абсолютным адресам, структуре полей в памяти и т.п.
2. Встроенная поддержка верификации программ. Например, на основе логики Хоара в Ada, Dafny, Eiffel (есть другие варианты, на основе зависимых типов в Agda, Lean, F*, но они вроде как из другого семейства).
3. Встроенная в язык многопоточность/многопроцессность с определёнными гарантиями. Например, задачи в Ada, separate типы в Eiffel (отсутствие гонок).
4. Управление псевдонимами (aliasing). Например, в Rust.