



84,21

Рейтинг

Edison

Изобретаем успех: софт и стартапы



MagisterLudi 17 ноября 2015 в 13:35

## Пол Грэм: «Месть ботанов». В чем отличие Lisp

Профессиональная литература, Программирование, Lisp, Блог компании Edison

Продолжаем перевод эссе и книги Пола Грэма «Хакеры и Художники».

В конце статьи тех.директор компании Edison рассказывает, как они портировали Lisp на C#

*«Мы гонялись за C++ программистами. Нам удалось перетащить их целую кучу на полпути к Lisp.»*  
Гай Стил, соавтор Java спецификации.

Оригинал — *Revenge of the Nerds*, Май 2002  
и *What Made Lisp Different*, декабрь 2001  
За перевод спасибо Щёкотовой Яне.

Начало: Пол Грэм: «Месть ботанов», часть 1

### Часть вторая

#### Чем отличается Lisp

Когда Lisp был впервые разработан, он воплощал в себе 9 новых принципов. Сегодня некоторые из них мы воспринимаем как само собой разумеющееся, другие можно увидеть только в более продвинутых языках, а два все еще остаются прерогативой Lisp. Эти 9 принципов перечислены ниже в порядке их применения в основном IT-течении.



**1 Условные высказывания** (Conditionals). Условное высказывание представляет собой конструкцию if-then-else. Сегодня для нас это стандартно, но в Fortran I они отсутствовали. Там был только условный переход goto, на основе низлежащей машинной инструкции.

**2 Функциональный тип данных** (A function type). В Lisp функции — это такой же тип данных, как integer или string. Они имеют буквенное представление, могут быть присвоены переменным, могут передаваться в качестве аргументов, и т.д.

**3 Рекурсия** (Recursion). Lisp был первым языком программирования, который ее поддерживал.

**4 Динамическое типизирование** (Dynamic typing). В Lisp все переменные, в сущности своей, являются указателями. Значения — вот что и есть тип, а не переменные, и присвоение или связывание переменных означает копирование указателей, а не того, на что они указывают.

**5 Сборка мусора** (Garbage-collection).

**6 Программы, составленные из выражений** (Programs composed of expressions). Программы на Lisp представлены в виде деревьев выражений, каждое из которых возвращает значение. Это как противопоставление Fortran и большинству последующих языков, которое различает понятия «выражения» и «операторы».

Для Fortran I это различие было естественным, потому что операторы не могли быть вложенными. И таким образом, пока для работы нужны математические выражения, не было смысла в создании чего-то такого, что возвращало бы значение, т.к. это значение никуда не могло бы сохранено.

Такое ограничение ушло в прошлое с появлением языков с блочной структурой, но к тому моменту было уже слишком поздно. Различие между выражениями и операторами сильно укоренилось и распространилось с Fortran на Algol, а затем и на всех их потомков.

**7 Символьный тип** (symbol type). Символы по факту являются указателями на строки, сохраненные в хеш-таблицах. Таким образом, проверка равенства можно путем сравнения указателей, а не каждого символа.

## 8 Нотации для кода, использующего деревья символов и констант.

**9 Постоянная целостность языка.** В действительности различия между временем чтения кода, временем компиляции и временем выполнения нет. Можно компилировать или запускать код во время чтения, читать или запускать код во время компиляции, и читать или компилировать время его выполнения.

Запуск кода во время чтения позволяет пользователям перепрограммировать синтаксис Lisp. Запуск кода во время компиляции — это базовый принцип макроса. Компиляция во время выполнения — это принцип использования Lisp в качестве языка расширения в программах подобно Emacs. А чтение во время выполнения позволяет программам общаться посредством S-выражений, что не так давно было заново изобретено XML.

Когда Lisp впервые появился, эти принципы были далеко за пределами обычной практики программирования, что, в основном, было продиктовано аппаратурой, появившейся в конце 1950-ых годов. Со временем, базовый язык, воплощенный в последователях популярных языков, постепенно эволюционировал в Lisp. Принципы 1-5 сейчас широко распространены. Принцип под номером 6 еще только начинает проявляться в мейнстриме. Python находится на 7 стадии, хотя, кажется, для этого принципа отсутствуют какие-либо синтаксические правила.

Что касается пункта 8, то он, возможно, является одним из самых интересных. Принципы 8 и 9 стали частью Lisp случайно, потому что Стив Рассел внедрил то, что Маккарти никогда и не собирался делать. И все же, эти принципы оказались лежащими в основе как странного пояса Lisp, так и его наиболее отличительных черт. Lisp выглядит настолько странно не потому что у него своеобразный синтаксис, а потому что у него нет синтаксиса как такового. Вы пишете программы прямо в деревьях грамматического разбора, которые строятся за кулисами во время работы в других языках программирования, и эти деревья состоят из списков, которые являются структурами данных в Lisp.

Выражение языка в его собственных структурах данных оказывается довольно мощным свойством. Принципы 8 и 9 в совокупности означают, что можно написать программы, которые пишут программы. Это, возможно, звучит как бред, но в Lisp это обычное дело. Наиболее распространенным способом для осуществления этого является макрос.

Макрос (в контексте Lisp) все еще, насколько мне известно, редкость для языка Lisp. Частично это потому, что, чтобы написать макрос, Вам, вероятно, придется создать синтаксис Вашего языка программирования таким же странным как и у Lisp. Это также может быть потому, что Вы внедрите этот финальный штрих, то больше нечего и ожидать разработок нового языка, а только нового диалекта Lisp.

В основном, я это все преподношу как шутку, но в ней есть доля правды. Если определить язык, в котором есть car, cdr, cons, quote, cond, а и нотация для функций, выраженных в виде списков, тогда из всего этого можно построить все остальное нутро Lisp. Это, на самом деле, и определяющее качество Lisp: все было организовано так, чтобы Маккарти придал Lisp ту форму, которую он имеет сейчас.

## Где языки имеют значение

Итак, предположим, Lisp представляет некоторое ограничение, к которому асимптотически приближаются популярные языки. Означает ли это, что следует использовать этот язык для написания программного обеспечения? Сколько Вы теряете на использовании менее мощного языка? Лучше ли, иногда, оставаться в стороне от передовых инноваций? И разве популярность до некоторой степени не является своим собственным обоснованием? Разве невежественный начальник не прав, например, в том, чтобы использовать язык, для которого он сможет легко нанять программистов?

Конечно, существуют проекты, где выбор языка программирования не имеет значения. Как правило, чем требовательнее приложение, тем больший выигрыш Вы получаете от использования мощного языка. Но множество проектов не настолько требовательные. Большая часть процесса программирования состоит из написания небольших программ-посредников, где можно использовать любой язык, с которым Вы знакомы, и у которого есть хороший набор библиотек для любых Ваших целей. Если Вам нужно только передать данные из одной Windows программы в другую, конечно же, используйте Visual Basic.

Вы можете также писать программы-посредники и на Lisp (я его использую как настольный калькулятор), но наибольшая выгода от использования языков подобных Lisp в другом спектре применения, где Вам нужно написать сложные программы для решения сложных задач в условиях жесткой конкуренции. Хорошим примером является программа поиска цен на авиаперелеты, право на использование которой ITA Software предоставила компании Orbitz. Эти ребята вошли на рынок, где уже доминировали два крупных сильных противника, Travelocity и Expedia, казалось, просто подавили их с технологической точки зрения.

Ядро приложения ITA составляют 200 000 строк программы на Common Lisp, которая ищет на порядок больше возможностей, чем их конкуренты все еще, по-видимому, использующие технологии эпохи программирования мейнфреймов. (Хотя ITA также в некотором смысле использует программирование эпохи мейнфреймов). Я ни разу не видел ни строчки кода из программы ITA, но, согласно одному из их лучших специалистов, они используют много макросов, чему я несколько не удивляюсь.

*Продолжение следует*

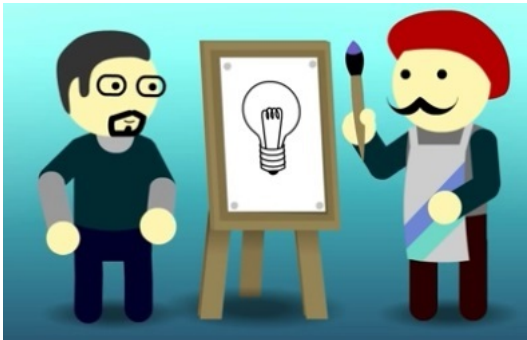
(Кто хочет помочь с переводами статей Пола Грэма — пишите в личку)

### Анекдот, (не)придуманный программистом, который портировал Lisp

Lisp широко использовался для создания экспертных систем ~~еще на перфектах~~, когда программирование только приходило в промышле и оборонку. Поэтому есть «куски» инфраструктуры с суперответственной функцией, которые народ просто боится трогать, ибо может шара: масштабах страны.

Интерфейс, обертку, математику — все можно переписать, а вот переписать экспертную систему — это связано с большими политическим организационными и репутационными рисками и ответственностью, если в результате что-то пойдет не так.

Представьте себе сервак, с огромной вычислительной мощностью, который обслуживает алгоритм, написанный под железо 30-ти летней давности, с низкой производительностью (Математический сопроцессор? — Не, не слышал). И масштабировать его можно только тупым наращиванием мощности. Потому что сам код дописывался вручную на протяжении десятилетий.



*Тех. директор компании Edison:* Мы портировали вычислительный модуль в C#, написанный давным-давно разработчиком на Lisp. Модуль состоял частично из экспертной системы, а частично из математики (преобразования Фурье, статистик: был выбран из-за того, что создатель алгоритма знал только его. Говорили, что он дорабатывал его чуть ли не всю жизнь.

Причина перехода — использовался старый компилятор Lisp, который не давал в производительности. Много лет расчеты на Lisp собирались в виде библиотеки, склеивались с современными программами и вызывались таким образом: в библи передавались исходные матрицы, она надолго задумывалась и потом отдавала результат. Теперь заказчику хотелось использовать новые возможности Windows и

ускорить выполнение расчета. Тем не менее, код хотелось сохранить без изменений, чтобы не испортить «эвристическую» часть.

Решение состояло из двух частей:

- «дословное» переписывание кода, который содержал местами очень некачественные и даже корявые выражения, будто сделанные на
- оптимизация, которая сопровождалась проверкой новой библиотеки. Когда на вход алгоритма подавалось 30 различных схем исходны: данных, а наша задача была в том, чтобы удостовериться в идентичности результата, полученного через старую (Lisp) и новую (C#) библиотеки.

Чтобы понимать охват — система развернута в масштабах страны и обрабатывала десятки тысяч объектов.

**Метки:** [lisp](#), [Пол Грэм](#), [дизайн языков программирования](#), [edisonsoftware](#)

↑ +13 ↓ 61 20,5k 18



**Edison** 84,21

Изобретаем успех: софт и стартапы



171,5

Карма

227,6

Рейтинг

428

Подписчики

Алексей @MagisterLudi

Строю реактивный ранец, он же Jetpack

[Facebook](#)

[Twitter](#)

[Вконтакте](#)

[Google+](#)

Поделиться публикацией

### ПОХОЖИЕ ПУБЛИКАЦИИ

26 сентября 2016 в 11:29

Пол Грэм. Все статьи на русском. Год спустя

10 мая 2016 в 17:51

Пол Грэм, «Хакеры и художники», глава 5: «The Other Road Ahead», продолжение

↑ +9

👁 7,9k

📌 57

💬 5

28 апреля 2016 в 11:51

Пол Грэм: «The Other Road Ahead», часть 1


↑ +8

👁 8,5k

📌 33

💬 12

Комментарии 18




bigfatbrowncat 17.11.15 в 14:59

🔗 📌

↑

Представьте себе сервак, с огромной вычислительной мощностью, который обслуживает алгоритм, написанный под железо 30-ти летней давности с низкой производительностью (Математический сопроцессор? — Не, не слышал). И масштабировать его можно только тупым наращиванием мощности. Потому что сам код дописывался вручную на протяжении десятилетий.

Представил. Это — кошмар вселенского масштаба, от которого надо уходить любой ценой. Постепенно, модуль за модулем переписывая код, обклеив каждый старый и новый модуль тысячами самых заковыристых Unit-тестов. Потому что ситуация, при которой древняя программа принимает решения за людей, которые не понимают, как она работает — классический сюжет для фильма в жанре «техногенный апокалипсис».



Alexeyco 17.11.15 в 17:16

🔗 📌 📄 🔄

↑

> Потому что ситуация, при которой древняя программа принимает решения за людей, которые не понимают, как она работает — классический сюжет для фильма в жанре «техногенный апокалипсис».

Да мы каждый день такую поддерживаем... а переписать нельзя — потому как работает, много времени, приоритеты другие, Иванова (Петрова, Сидорова, Нусмухамедова — нужного подчеркнуть) нету на месте (переходит на новую должность, заболел, ушел в декрет, вышел из декрета), а должен согласовать (передать, рассказать, как должно работать, исправить, запустить, дать доступ).




bigfatbrowncat 18.11.15 в 14:52

🔗 📌 📄 🔄

↑

приоритеты другие

Они могут внезапно и резко измениться :)




klvov 03.02.16 в 20:16

🔗 📌 📄 🔄

↑

Хуже того, это не только «сюжет для фильма в жанре «техногенный апокалипсис»», а ситуация, которая уже происходила в истории, и не поручусь сейчас где-то что-то подобное не происходит или не будет происходить. Именно о ситуации «программа принимает решения за людей, которые не понимают, как она работает» Вейценбаум писал в своей (очень, кстати, поучительной) книжке:

*Чтобы дополнить поучительную историю, рассказанную Моррисоном, нет нужды обращаться к системам, которые появятся в будущем. В время войны США с Вьетнамом вычислительные машины, используемые офицерами, не имевшими ни малейшего понятия о том, что происходит внутри этих машин, фактически определяли, какие деревни должны подвергнуться бомбардировке и в каких зонах концентрация вьетконгов достаточно для того, чтобы «имелись основания» объявить их «зонами стрельбы без предупреждения», т. е. в обширных географических районах летчики наделялись «правом» убивать любое живое существо.*



sashkin 17.11.15 в 19:42

🔗 📌

↑

Что-то я не понял смысла последнего раздела про древний проект на лиспе: сначала ода языку ушедшему вперёд задолго до мейнстримов, а в постах абзацах пример того, как проект с лиспа переписали на C#. Какой-то разрыв шаблона или я что-то не понял.




taskmgr 17.11.15 в 23:10

🔗 📌 📄 🔄

↑

Аналогично




PingWin 18.11.15 в 10:56

🔗 📌 📄 🔄

↑

Я так понимаю, речь о том, что Lisp — красивый академический язык. Интересно спроектированный, академически выверенный и т.д. К которому постепенно стремятся остальные языки/платформы.

Но тем не менее широкой поддержки не снискавший, а прикладные вещи проще писать и поддерживать на широко распространённых платформах: поддерживать проще — да банально людей на рынке больше и они дешевле.



loz 18.11.15 в 13:36

🔗 📌 📄 🔄

↑

Что, простите? Лисп красив академически с какой стороны? Мы ведь про CL говорим? У него куча совсем не академических, и тем более не выверенных особенностей типа системы рестартов или мутабельность структур для производительности, подсказок компилятору и прочего.

Что как бы совсем немного намекает на нацеленность на чисто практическое применение языка. И широту его поддержки ты как оценил? То, чем писали чуть менее чем все исследователи AI, а спонсировали и вкладывались в создание стандарта не столько университеты, сколько реальные корпорации и само мин обороны штатов говорит об обратном.

 RolexStrider 17.11.15 в 23:49

Вы пишете программы прямо в деревьях грамматического разбора, которые строятся за кулисами во время парсинга в других языках программирования, и эти деревья состоят из списков, которые являются структурами данных в Lisp.

было организовано так, чтобы Маккарти придал Lisp ту форму, которую он имеет сейчас.

Снова про «Покайтесь, грешники!» «Восславим же!» «О Аллилуйя!» ибо...

ибо

 igrishaev 18.11.15 в 10:06

В этой статье автор подробно описывает, почему переписал Реддит с Лиспа на Питон. Делает анализ, описывает плюсы и минусы. У вас же последний раздел ни о чем — ну переписали, и что с того.


 vba 18.11.15 в 10:32

Сам последнее время вынужден писать на C# 6, но только что бы содержать семью. Но вот никак не могу взять в толк зачем переносить с Lisp на C# когда есть F#?

 PingWin 18.11.15 в 10:45

А чтение во время выполнения позволяет программам общаться посредством S-выражений, что не так давно было заново изобретено как XML.

Объясните кто-нибудь, при чём тут XML, как наличие функции типа «eval()» (а я так понимаю, речь идёт о ней?) помогает читать XML? Возможно, в оригинале речь шла о JSON или чём-то подобном?

 loz 18.11.15 в 15:42

Вот тут про это очень подробно написано: [www.defmacro.org/ramblings/lisp.html](http://www.defmacro.org/ramblings/lisp.html)

 PingWin 18.11.15 в 15:54

Ааа, подмена синтаксиса lisp под синтаксис xml... Забавно, но в продуктив такое же не пустишь — это ж будет огромная дырень в безопасности. Да и не очень понятно, при чём тут именно XML, так наверное можно любой текстовый формат читать... А теоретически и не только текстовый

 loz 18.11.15 в 16:11

Не подмена, а переизобретение интерпретации в разных применениях где это удобно. Про дырень в безопасности — это зависит от многих факторов, запуск кода в безопасном окружении никто не отменял. Плюс не обязательно именно любой исполняемый код разрешать — можно и валидировать и что угодно делать, зависит от требований, времени и фантазии.

 loz 18.11.15 в 15:45

Причина перехода — использовался старый компилятор Lisp, который не давал высокой производительности.

Так а в чем была проблема просто взять новый, современный, быстрый компилятор типа SBCL? Хочется подробностей.

 PQR 19.11.15 в 10:33

Надо было переписывать на Clojure — вот про такой опыт было бы интересно почитать!

 potan 19.11.15 в 19:57

Сейчас есть куча компиляторов разных Lisпов под разные платформы (в том числе и .net). Не понятно, почему понадобилось переписывать именно C#?

Только полноправные пользователи могут оставлять комментарии. Войдите, пожалуйста.

САМОЕ ЧИТАЕМОЕ

- Сутки
- Неделя
- Месяц

Проектирование пользовательского интерфейса Windows 95

↑ +35

👁 9,5k

📖 54

💬 15

20 лет инициативе Open Source

↑ +41

👁 6,5k

📖 33

💬 15

Собеседование с ног на голову

↑ +14

👁 8,9k

📖 27

💬 10

Микросервисное безумие пройдет в 2018 году

↑ +17

👁 8,2k

📖 58

💬 11

Теневой бан и с чем его едят

↑ +29

👁 7,8k

📖 15

💬 33

ИНТЕРЕСНЫЕ ПУБЛИКАЦИИ

Bloomberg: как акция Илона Маска по продаже огнеметов изменит финансирование стартапов

↑ +5

👁 44

📖 5

💬 0

Как в реальности патч от уязвимостей Meltdown и Spectre влияет на производительность

GT

↑ +6

👁 3,3k

📖 5

💬 30

Как я переехал в ЕС: легализация, изучение языка, поиск жилья и работы

↑ +6

👁 2,3k

📖 21

💬 10

Выпуск#10: ITренировка — актуальные вопросы и задачи от ведущих компаний

↑ +5

👁 628

📖 12

💬 5

Тutorial по Unreal Engine: C++

↑ +12

👁 1,2k

📖 27

💬 4

Аккаунт	Разделы	Информация	Услуги	Приложения
Войти	Публикации	О сайте	Реклама	<div>Загрузите в App Store</div> <div>доступно Google</div>
Регистрация	Хабы	Правила	Тарифы	
	Компании	Помощь	Контент	
	Пользователи	Соглашение	Семинары	
	Песочница	Конфиденциальность		
<div>TM</div> © 2006 – 2018 «TM»		Служба поддержки	Мобильная версия	<div></div>