Сергей Свердлов

Арифметика синтаксиса

Источник: PC Week/RE, 1998, №42-43

Мало найдется в компьютерном деле тем, которые вызывали бы столько противоречивых суждений и споров, как сравнение языков программирования. В вопросе этом царит субъективизм и обсуждение чаще бывает похожим на спор религиозных фанатиков, а не на ученую дискуссию.

Когда я был молодым аспирантом, для нас были организованы курсы по ФОРТРАНУ. Тогда вовсю разворачивались ЕС ЭВМ, на которых ФОРТРАН был одним из основных языков. Я к тому времени немного знал Алгол-60, написал на нем несколько небольших программ. Посещая эти курсы, я все время пытался понять чем ФОРТРАН лучше Алгола. Чтобы разрешить свои сомнения, я приставал с вопросами к старшим товарищам. Никто из них тезис о преимуществе ФОРТРАНА не подверг сомнению. А вот почему он лучше, можно было услышать самые разные аргументы. Запомнился такой, например, ответ: "В ФОРТРАНЕ массивы могут быть семимерными, а в Алголе - нет". В этом ответе все. И желание скрыть поверхностное знакомство с Алголом и ФОРТРАНОМ. И выдвижение на передний план вещей второстепенных. И смешение понятий собственно языка и его конкретной реализации. С тех пор, между прочим, мне так ни разу и не пришлось употребить семимерные массивы.

Не вполне аргументированный подход с сравнению языков можно даже понять и в известной мере оправдать. В реальной работе не так уж часто возникает потребность одному человеку активно работать на разных языках.

Немалую роль играют пропаганда и мода. Сейчас, например, мы все являемся свидетелями массированной пропагандистской кампании, развернутой вокруг языка Ява. Впервые язык программирования выводится на рынок такими же маркетинговыми приемами, что и традиционные товары.

Какой язык лучше?

Существует ли ответ на этот вопрос. Нет, наверное. Как минимум, следует уточнить. Лучше для чего? Для какой сферы применения. Для какой задачи. Для какого проекта. Для какого программиста, наконец. И уж, конечно, оценивая тот или иной язык, хорошо бы опираться на хоть сколько-нибудь объективные критерии.

Мне несколько раз встречались публикации, в которых предпринималась попытка оценить языки по определенному набору критериев. За основу берутся, как правило, экспертные оценки.

Но все это достаточно субъективно. К тому же теперь за оценки беспристрастных экспертов бывает выдается откровенная реклама. И еще один момент. Когда вы, например, слышите, что язык Ява проще Си++, то спрашивается, а насколько проще? Или, может, во сколько раз? Задавшись однажды таким вопросом, я и пришел к идее того небольшого исследования, результатами которого хочу поделиться.

Какой язык проще?

Не будем пытаться выстроить критерий, значение которого можно было бы вычислить и который без каких-либо оговорок можно было бы назвать "сложность языка программирования". Займемся тем, что в любом языке, начиная с Алгола-60, строго формализовано — описаниями синтаксиса языков. И оценкой объема этих описаний.

Конечно синтаксис — это лишь внешняя форма, но он играет важную роль. Синтаксические единицы являются одновременно и основными смысловыми понятиями любого языка: модуль, класс, блок, функция, процедура, метод, оператор, выражение. Ну а что, если не система понятий и их количество, может характеризовать сложность языка.

Те, кто знаком с устройством компиляторов, знают что именно синтаксический анализатор является как бы скелетом всего компилятора.

Итак, *будем рассматривать в качестве меры сложности языка программирования количественные характеристики формализованного описания его синтаксиса.*

Объекты

Не вздрагивайте. Я не буду говорить про инкапсуляцию, наследование и полиморфизм. Речь пойдет о языках программирования, которые станут объектами нашего исследования. Во-первых, поскольку мы собрались оценивать только сложность языков, нужно взять языки сопоставимые в функциональном отношении. Во-вторых, нужно, чтобы способы описания синтаксиса рассматриваемых языков можно было привести к некоторому общему знаменателю. И, наконец, желательно, чтобы к обсуждаемым языкам существовал общественный интерес. Предметом нашего исследования будут: Алгол-60, Паскаль (в версии Н.Вирта), Си (K&R), Ада, Модула-2, ряд версий Турбо (Борланд) Паскаля, Си++, Объектный Паскаль (Delphi), Оберон, Оберон-2, Ява. Коротко об участниках соревнований.

Алгол-60

Это первый язык, синтаксис которого был описан формально с помощью специально для этого созданной нотации – формул Бэкуса-Наура (БНФ). Вот как в этой нотации выглядит определение идентификатора:

```
<идентификатор> ::= <буква>| <идентификатор><буква> | <идентификатор><цифра>
```

В 60-е и в начале 70-х Алгол был самым популярным языком в СССР. Сейчас Алгол не используют. Но он рассматривается здесь поскольку является родоначальником по сути всех обсуждаемых языков. Нам же будет интересно проследить насколько усложнены (или, может, упрощены) современные языки по сравнению с Алголом.

Паскаль

Прямой потомок Алгола. При описании синтаксиса Паскаля его автор Н.Вирт использовал БНФ, добавив в нотацию скобки { и }, означающие повторение заключенной внутри них конструкции. Паскаль в оригинальной авторской версии не содержит средств раздельной компиляции – модулей, многообразных числовых типов, строк переменной длины и многого из того, что добавлено в известных многим реализациях.

Си

Рис. 1. Деннис Ритчи – автор языка Си.

Этот язык достаточно традиционен. Стал очень популярен благодаря многим остроумным решениям, сделавшим запись программы на Си весьма компактной. Накладывает на программиста не слишком много ограничений и дает возможность для разнообразных трюков, чем тоже многим импонирует. При описании языка использована нотация, в принципе эквивалентная БНФ. Каноническим считается описание языка, данное в книге Б.Кернигана и Д.Ритчи - автора языка.

Именно, описание из русского издания этой книги (Керниган Б., Ритчи Д., Фьюэр А. Язык программирования Си. Задачи по языку Си. М., Финансы и статистика, 1985) мы и будем использовать, говоря о языке Си.

Ада

Официальный язык программирования американских военных. Происходит от Паскаля, но заметно сложнее его (интересно будет узнать насколько). Описан очень точно и строго. Для описания синтаксиса использован вариант БНФ.

Модула-2

Рис. 2. Никлаус Вирт — создатель Паскаля, Модулы и Оберона.

Язык, который должен был заменить Паскаль, устранив основное его ограничение – отсутствие модульности. Но этого в полной мере не произошло. Благодаря политике компании Борланд, которая в тот момент, когда нужно было обеспечить модульность в создаваемых ею системах программирования, решила, что выгоднее добавить новые элементы в Паскаль, а не переходить на Модулу-2. Тем не менее, известно, что Модула-2 использовалась и используется в проектах, где важнейшую роль играет надежность. Средства межмодульного контроля Модулы-2 заметно совершенней аналогичных возможностей Турбо Паскаля и Си.

При описании синтаксиса языка Н.Вирт использовал так называемые расширенные формулы Бэкуса-Наура (РБНФ). В нотацию введены средства выражения итерации и задания необязательных частей, что позволяет в большинстве случаев обходиться без рекурсивных определений. Идентификатор теперь определяется так:

```
Идентификатор = буква { буква | цифра }.
```

Фигурные скобки показывают, что заключенная в них конструкция может повторяться ноль или более раз. Получается заметно проще и удобней. При наших измерениях синтаксические правила для всех исследуемых языков были преобразованы именно в РБНФ.

Турбо Паскаль и Объектный Паскаль

Компилятор Турбо Паскаль, разработанный Андерсом Хейльсбергом, (Anders Hejlsberg) был выпущен в продажу фирмой Борланд в 1983 году.

Рис. 3. Андерс Хейльсберг – разработчик Турбо Паскаля и Delphi.

Уже эта версия содержала расширения языка, хотя и небольшие. В последующих версиях расширений становилось все больше и больше: встроенная графика (версия 3.0), от которой потом отказались, модули (4.0), средства ООП (5.5) и т.д. и т.д. Начиная с версии 7.0, язык стал называться Борланд Паскаль, а с появлением системы Delphi был переименован в Объектный Паскаль. Сейчас входной язык Delphi содержит уже очень много синтаксических расширений по сравнению со стандартным Паскалем. Существует даже мнение, с которым я вполне солидарен, что именно отсутствие необходимости придерживаться стандарта языка объясняет тот факт, что визуальная среда программирования была создана компанией Борланд сначала для Паскаля, а уж потом для Си. Но думаю также, что не менее, а скорее более важной причиной этого было то, что разработкой Delphi занимался выдающийся программист А.Хейльсберг.

Си++

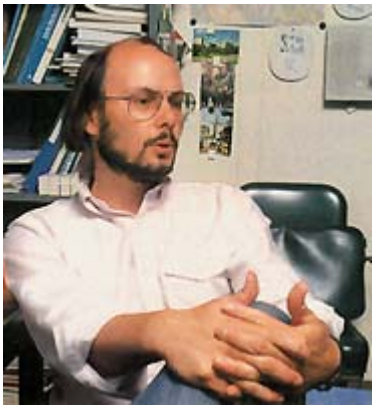


Рис. 4. Бъярн Строуструп придумал Си++.

Первоначальное название "Си с классами". В язык Си Бъярном Строуструпом введены средства ООП и ряд других добавлений. До последнего времени самый модный язык. Сейчас принято считать, что Си++ сложноват и не слишком надежен. Многие говорили об этом с самого начала.

Оберон

Разработан Н.Виртом в 1987 году. Представляет собой существенно упрощенный вариант Модулы-2, в который добавлены расширяемые записи – основной механизм ООП. Язык необычайно прост. При этом сохраняет универсальность и в функциональном отношении не уступает другим языкам.

Оберон-2

Рис. 5. Ханспетер Мёссенбёк – соавтор Н.Вирта по языку Оберон-2.

В 1992 году были приняты расширения языка Оберон, предложенные молодым коллегой Н.Вирта Ханспетером Мёссенбёком. В язык введены так называемые связанные процедуры — аналог виртуальных методов в других языках.

Ява

Рис. 6. Джеймс Гослинг — "отец" языка Ява.

Самый молодой и самый обсуждаемый ныне язык. Является непосредственным наследником Си++. Отличается от него отсутствием некоторых потенциально ненадежных механизмов, а также тем, что устранены любые, не относящиеся к ООП, средства. Объекты и ничего кроме объектов. Говорят, что Ява простой язык. Но стоит заглянуть в его официальное описание (James Gosling, Bill Joy, Guy Steele. The Java Language Specification Version 1.0), как возникает повод усомниться. Тяжелый (в прямом и переносном смысле) 700-страничный документ, насыщенный многословными и громоздкими определениями.

Программа

Круг исследуемых языков определен. Сформулируем теперь порядок проведения подсчетов:

- Описание синтаксиса каждого из сравниваемых языков записывается в нотации РБНФ в текстовые файлы.
- С помощью специальной программы эти файлы обрабатываются с целью подсчета количественных характеристик описаний синтаксиса.

Программа, с помощью которой выполнялись измерения устроена точно на тех же принципах, что и компиляторы языков программирования. Не удивительно. Ведь РБНФ-нотация сама представляет собой язык со своей лексикой и синтаксисом. Программа проверяет корректность обрабатываемых описаний. В ходе анализа строятся таблицы и ведутся подсчеты.

Критерии

Правила, записанные на РБНФ, (как и текст на языке программирования) состоят из отдельных элементов – лексем. Лексемами являются названия понятий, называемые в теории формальных языков нетерминальными символами или просто нетерминалами. Например, в правиле

```
ПоследовательностьОператоров = Оператор {";" Оператор}.
```

нетерминалами являются *ПоследовательностьОператоров* и *Оператор*. Терминальные символы — это те знаки, из которых и состоит в конечном счете (terminal — конечный, заключительный) программа. При записи на РБНФ терминальные символы записываются в кавычках. В приведенном примере один терминал — ";". Терминальные символы это тоже лексемы РБНФ. Наконец к числу лексем относятся специальные знаки, используемые в самой РБНФ. В правиле про последовательность операторов это знак равенства, фигурные скобки и точка в конце.

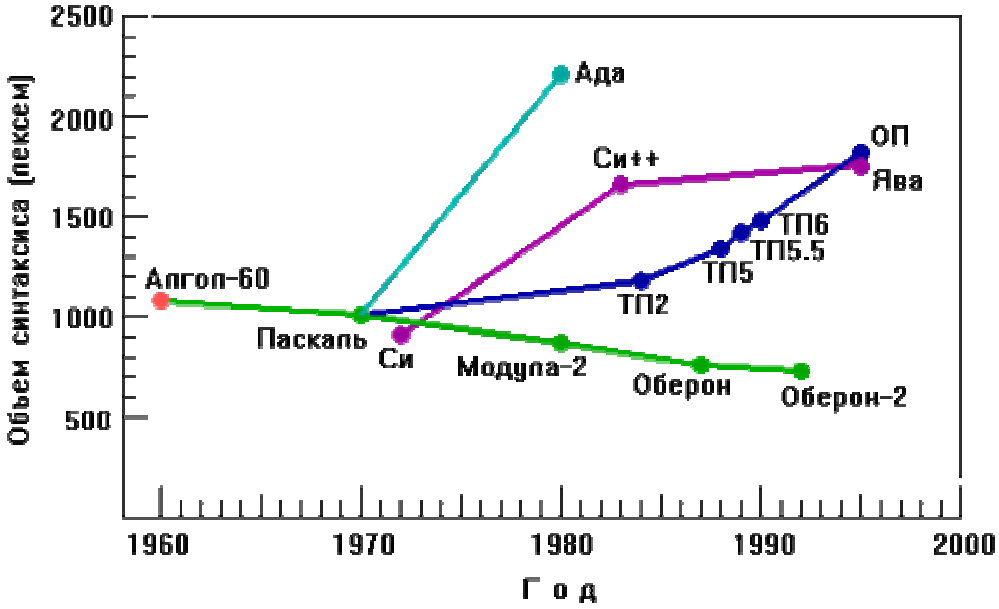
Общее число лексем в описании синтаксиса языка может служить обобщенной характеристикой размера этого описания. Число лексем использовать в качестве меры объема гораздо лучше, чем, скажем, число знаков в описании. В этом случае значение нашего критерия не будет зависеть от того, на каком языке (русском, английском) или какими конкретно словами названы нетерминалы — понятия языка. *Число различных нетерминалов* — следующая характеристика, которую мы будем вычислять. Количество используемых для описания языка понятий — несомненно важнейшее свойство, от которого зависит легкость освоения этого языка. Можно заметить, что число нетерминалов должно быть равно *числу правил* в описании синтаксиса, поскольку для каждого понятия обязано существовать ровно одно правило. Набор и *количество различных терминальных символов* языка, упомянутых в синтаксических формулах, характеризуют лексику языка — набор знаков и специальных символов. Во всех обсуждаемых нами языках существуют служебные слова, которые могут употребляться только в строго определенном смысле. Их, программист, вообще-то, должен знать наизусть. Подсчет *количества служебных слов* позволит оценить объем зубрежки.

Результаты

Элементы	А-60	Паскаль	ТП2	ТП5	ТП5.5	ТП6	ОП	М2	О	О2	Си	Си++	Java	Ада
Лексемы	1085	1012	1184	1331	1410	1488	1825	887	765	726	917	1662	1771	2206
Нетерминалы	119	110	124	127	135	143	180	70	62	43	53	126	174	226
Терминалы	92	84	87	87	87	89	90	88	90	91	123	131	121	102
Служебные слова	25	35	42	48	52	55	83	39	32	34	27	47	48	63

Линия Вирта

Это линия языков, начинающаяся от Алгола-60 и образованная языками Паскаль, Модула-2, Оберон и Оберон-2, автором которых является Никлаус Вирт, и, что самое главное, линия, которой Вирт неуклонно придерживается: наращивание мощи языка без его усложнения. Паскаль намного богаче Алгола, но не сложнее его. Модула существенно мощнее и совершеннее Паскаля, но проще. Оберон обогатил Модулу средствами объектно-ориентированного программирования — расширяемыми записями, и при этом не только не стал более сложным, но заметно упрощен.



Удивительным выглядит то, что Оберон-2 оказался проще Оберона, расширением которого является. В отношении размера определения синтаксиса так оно и есть. Да и по существу нововведения Оберона-2 оформлены очень экономно. Кроме того авторы языка объединили отдельные правила для каждой разновидности операторов в одно правило для нетерминала "Оператор". То же сделано в отношении правил для типов. По-другому, более компактно, определен синтаксис некоторых конструкций. И хотя получившееся упрощение отчасти формальное, но экономия понятий — это именно то, к чему и следует стремиться, как заметил еще У.Оккам почти 700 лет назад.

Кроме того, что остаются простыми формальные описания синтаксиса языков Н.Вирта, очень лаконичны и полные тексты спецификаций этих языков. Сравнивая авторское описание Паскаля, тоже очень небольшое, и описания языков Оберон и Оберон-2, можно увидеть как маэстро совершенствовал свое мастерство. Я очень рекомендую специалистам по Си++ и Яве почитать спецификацию Оберона, даже если вы не собираетесь использовать этот язык. Всего 20 страниц! И при том, это документ, необходимый и достаточный для реализации языка. Сравните с аналогичного назначения книгами Б. Строуструпа и Д. Гослинга по Си++ и Яве.

Линия Борланд

Берет начало от виртовского Паскаля, а дальше "все выше и выше и выше...". Версии Турбо - Борланд — Объектного Паскаля становятся сложнее и сложнее. По другому просто не может быть, поскольку, избрав однажды путь расширения старого языка с сохранением обратной совместимости, язык можно только усложнять. Отказаться от чего-либо уже невозможно. Недаром по линии Борланд идет монотонный рост всех без исключения критериев.

Что же в результате. Из простого и изящного Паскаля получился язык, приближающийся по сложности к языку Ада. По большому счету Паскаль в версии Борланд (Inprise) уже не может считаться общемировым языком программирования. Это фирменный язык одной не очень большой американской компании. В этом смысле он ничем не отличается от Бейсика, языка другой, правда более крупной фирмы. Отсутствие переносимости даже делает не вполне правомерным сравнение этого языка с Си, Си++, Явой, Модулой, Обероном и Адой.

От простого к сложному

Одним из неожиданных для меня результатов измерений стало то, что Си оказался не простым, а очень простым языком. По некоторым параметрам он даже проще Оберона. Недаром же он завоевал в свое время такую популярность. И в самом деле Си весьма гармоничен. И хотя поощряемый им стиль принимают не все, но в изяществе языку не откажешь. Что стоит только знаменитое ++. Интересно, кстати заметить, что Си обладает одним самых обширных наборов терминальных символов. Вот они где эти "штучки" ++, +=, && и т. д.

Ничего хорошего в смысле простоты нельзя сказать про Си++. Сейчас признание чрезмерной сложности этого языка стало общим местом. Так что неожиданностей наши измерения не дали. Разве что можно заявить о том, что Си++ сложнее Си практически вдвое. Вообще, если Си – конструкция весьма цельная, то Си++ представляется достаточно противоестественным соединением языка с незамаскированными понятиями низкого уровня и высокоуровневой концепции объектов. К тому же, мне кажется, что популярность Си возникла естественным путем., а Си++ – искусственно "раскрученный" язык.

Голый король?

А теперь обсудим характеристики самого современного, самого объектно-ориентированного и очень простого языка Ява. Да, но что это? Судя по вычисленным нами критериям Ява не только не может считаться простым, но является одним из самых сложных языков, более сложным чем Си++, и вдвое более сложным чем Оберон.

Но может сопоставление с тем же Обероном некорректно? Ведь, наверное, Ява все же более богатый язык, чем этот ваш Оберон? Ничего подобного! В Яве есть всего две существенные вещи, которых нет в Обероне. Это встроенная многопоточность и обработка исключений. Целесообразность включения средств параллельного программирования непосредственно в язык подвергается сомнению многими специалистами.

Это могло бы решаться на уровне библиотек. К тому же тот механизм, который реализован в Яве, решение отнюдь не самое удачное. Я с содроганием вспоминаю ощущения, полученные при чтении главы из спецификации языка Ява, посвященной потокам. Что касается обработки исключений, то встроенная в Яву, она, во-первых, усложняет язык, во-вторых все время напрягает программиста, вынуждая его употреблять всякие лишние слова, даже если он этого не хочет. Полноценно восстановиться после по-настоящему серьезной исключительной ситуации редко удается, разве что можно выдать собственное сообщение об ошибке вместо системного. Простые соглашения о флагах вполне могли бы решать те же задачи.

Зато в маленьком Обероне есть и полноценные записи (объекты) и нормальные многомерные массивы, а не только указатели на них. Есть в Обероне и привычные строки с нулем на конце, которые являются просто массивами символов, а никакими не объектами, а значит не требуют специальных средств для манипуляций с ними. А еще есть множества, вложенные процедуры, параметры-переменные. Вопрос о последних особенно интересен. В языках Си и Си++, как известно, параметры всегда передаются по значению то есть параметров-переменных тоже нет (*Здесь я ошибся. В Си++ есть параметры-ссылки. Эта ошибка, однако, не меняет смысла последующего замечания. Прим. авт.*), но зато есть возможность передать в качестве параметра адрес, что полностью решает вопрос. В Яве же в борьбе с адресной арифметикой вместе с водой выплеснули и ребенка: параметров переменных нет и адресов тоже. В качестве компенсации предлагается нечто несуразное под названием классы-фантики (wrapper classes).

Вопреки пропаганде Ява содержит мало чего действительно нового. Та же концепция виртуальной машины — просто первое, что приходит в голову, если задуматься о многоплатформности. Лет 25 назад это было удачным и свежим решением. Сейчас разработаны существенно более эффективные подходы.

Впрочем, я, кажется, отошел от темы и впал в религиозный фанатизм. Но, согласитесь, что столкнувшись с вопиющим несоответствием широковещательных заявлений и реального положения дел, трудно остаться равнодушным.