Персональный язык программирования

Статья имеет целью познакомить читателя с одним представителем огромной армии неизвестных языков программирования. Этот язык я придумал сам для себя, он ни на какой язык не похож и этим может быть интересен.

Виктор Кон http://kohnvict.narod.ru

Про языки программирования написано очень много статей и философских размышлений. Их много и есть необходимость в их классификации. Главный аспект – это то, что языки могут быть многоуровневыми. Есть ассемблер, есть базовые языки более высокого уровня: Basic, C++, Java и много других. С их помощью можно написать компиляторы и интерпретаторы других языков еще более высокого уровня и так далее. Второй аспект – разделение языков на компилируемые и интерпретируемые. Первые используются для создания независимых и конечных программ, например программы решения квадратного уравнения. Вторые являются текстовыми инструкциями для одной единственной программы – интерпретатора. Бывают и пересечения, когда программа частично компилируется в промежуточный код, который затем интерпретируется. Ну и, наконец, третий аспект – структура самой программы, тут три уровня. Вся программа одним куском, программа с процедурами, программа с объектами. Какой язык и какая программа удобнее для конкретного пользователя зависит от свойств самого пользователя.

Вместо введения

Я хочу рассказать о себе. Я пользователь в том смысле, что пишу программы для своей научной работы. Почти все программы моделируют те или другие физические процессы и часто используются только один раз. Условно конечно, много раз в процессе отладки до тех пор пока не получен достоверный результат. После того, как результат получен – программа уже не нужна. Опять неточно, нужен ее код, чтобы развивать его дальше и учесть более сложные процессы, но на каждом этапе программа пишется на результат, который получается конечным числом ее запусков. Как писать такие программы? Можно конечно одним куском и на ассемблере. Но это будет огромный труд с очень низким коэффициентом полезного действия. Можно писать процедуры и компилировать их в новые сборки программ. Но и это не оптимально. В каждой программе будет очень много одного и того же кода, который будет повторяться, засоряя носители информации. Это не так важно сейчас, но было важно раньше, когда компьютеры были маломощные. Сама практика подсказала, что наиболее оптимальным путем является программа-интерпретатор, выполняющая

Кон Виктор Германович

родился 22.09.1944,
физик, теоретик,
доктор физ.-мат. наук.

Постоянная должность: главный научный сотрудник Теоретического Отдела Института Сверхпроводимости и Физики Твердого Тела Российского Научного Центра «Курчатовский Институт» (РНЦ КИ).

Адрес РНЦ КИ:
пл. Курчатова 1, 123182 Москва, Россия.

команды одну за другой, а наиболее удобным и простым для выполнения конкретных работ является командный язык программирования с возможностью группировать код в процедуры.

Так получилось, что я пришел к этому выводу в далеком 1991-м году, когда программ еще было мало, все работали в операционной системе ДОС, которая мало что умела и о программах типа Mathematica, IGOR-pro и им аналогичных даже и не мечтали. Моя идея состояла в том, что нужно сделать интерпретатор языка программирования высокого уровня, который имел бы команды всех уровней, начиная от прерываний ассемблера и кончая запуском редактора текстов или программы построения графика одной командой. В то же время структура языка должна быть такой, чтобы четкие и конечные операции можно было сделать написав несколько строк текста. Я потратил два года работы и сделал себе такую программу. Работая в ДОС, она имела графические возможности, необычный редактор текстов с уникальными возможностями, хорошие возможности по интерфейсу и очень мне помогала в работе. Самый главный плюс такой программы состоял в том, что я мог очень быстро получить ответ на многие сложные вопросы. А быстрота получения ответа иногда имеет решающее значение. Часто просто нет времени на написание и отладку огромного кода, ответ нужен, как говорят, еще вчера. И я мог это делать.

Минусы у программы тоже были – отсутствие шрифтов. Конечно, я создал свой масштабируемый фонт, который можно было использовать, но это было не очень быстро и не очень красиво. Когда появилась Виндовс, а компьютеры стали более мощными, необходимость в интерпретаторе отпала. Самый главный минус был в том, что программа могла работать только в ДОС. Виндовс не разрешал многие операции, которые она могла делать, да и просто ее не запускал по соображениям безопасности. Я снова рассыпал программу на отдельные куски, но и они постепенно становились все менее привлекательными.

Прошло время – появились КПК (карманные персональные компютеры), это чудо на ладони с большими возможностями по части поиграть музыку или показать кино. Но они не умели делать вычисления, не было программ. А уж о том, чтобы программировать прямо на КПК не было и речи. Я снова вспомнил про свой язык программирования и свой интерпретатор. Код интерпретатора был написан на фортране, но для КПК нужно было писать на чем-то другом. Так получилось, что я в то время (это уже был 2004 год) начал работать на Java, поэтому выбор был сделан. Я просто перетранслировал вручную весь код с фортрана на Java, даже порой уже не понимая, что этот код конкретно делает. И все получилось – код заработал. Конечно, пришлось отказаться от прерываний, немного по-другому написать редактор текстов, по-другому сделать интерфейс. Но сам язык программирования и его интерпретация почти не изменились. Эта программа до сих пор работает у меня на КПК, который я как раз и купил в начале 2004 года. Он до сих пор работает с той же самой батареей. Это отдельная тема и я не буду ее развивать здесь.

Важно, что мне снова понравилось работать на своем языке, и я сделал клон этой программы также и для ПК, то есть настольного компьютера и ноутбука. Эта программа имеет больше возможностей и настолько удобна, по крайней мере, для меня, что я уже не мыслю себе другой жизни. Вот вчера мне понадобилось выделить из pdf файла книги в 150 страниц всего 4 страницы и записать в другой pdf файл. На моем яыке это делается в несколько строк, но и их писать не надо – они уже написаны. А вычислить, скажем,

таблицу значений функции Бесселя и построить график можно с помощью программы в одну строчку.

Но это только примеры. Реально программа умеет очень много всего, практически все, что лично мне нужно и самое главное – если чего-то нехватает, то просто дописываем интерпретатор и создаем новую команду. Сложность только в том, что, как и у любого интерпретируемого языка, команд очень много. Но они хорошо структурированы и есть описание с хорошей навигацией. Так что перечитав описание можно все легко вспомнить. Наизусть я свой язык не помню.

Вместо рекламы

Цель этой статьи – просто познакомить уважаемую публику с тем, что у меня есть. С самого начала я не стремился делать коммерческий продукт, но в какой-то момент работы над программой я подготовил вариант, который содержит только команды общего назначения, могущие представлять интерес для всех. Этот вариант я выложил в интернет на отдельный сайт http://vkacl.narod.ru, а также разместил в каталоги программ. Неожиданно программа стала популярной на сайте FreeSoft, где ее скачали уже более 40 тыс. раз. Были и письма, из которых я понял, что кое-кто действительно научился пользоваться. У программы есть несколько вариантов. Первый, основной, содержит интерпретатор языка, который я назвал ACL (advanced command language), среду разработки, включающую встроенное описание языка, и примеры готовых программ. Эта программа называется vkACL.jar. Как любая Java программа, она сама запускается через виртуальную машину Java или JRE, которую предварительно надо установить на компьютер. Дистрибутив JRE можно бесплатно скачать хоть с родного сайта, хоть с разных зеркальных сайтов, например [1]. Скриншот программы можно посмотреть вот по этому адресу [2].

Есть также вариант проигрывателя с окном, то есть это программа, которую можно настроить таким образом, что она будет показывать меню и выполнять (интерпретировать) одну ACL программу. Такой вариант не содержит помощи и выглядит как конечная программа на одну функцию. При этом jar-файл может иметь любое имя. Программу в таком варианте я использую для разработки конкретных программ, с которыми могли бы работать мои коллеги, не умеющие программировать. Эти программы выглядят как обычные Java программы. Есть еще и второй вариант проигрывателя, который совсем не имеет головного окна. Такая программа либо выполняет свою работу, не общаясь с пользователем совсем (так работали старые программы в эпоху до ПК), либо средства общения с пользователем написаны в самой ACL программе. Иногда такой вариант бывает оптимальным.

Наконец, есть вариант программы в виде Java-апплета, который сразу работает в интернете и запускается вот по этому адресу [3]. Этот вариант сделан совсем недавно и я его сейчас развиваю. Если у кого есть быстрый интернет, то можно сразу пользоваться программой с любого компьютера. К сожалению Java-апплет не может работать с файлами пользователя, но я организовал ввод и вывод данных через редактор текстов. Любую информацию, в том числе и картинки, можно ввести в программу и вывести из программы копированием текста через буфер обмена (клавиши Ctrl-A, Ctrl-C, Ctrl-V), а готовые картинки можно скопировать с экрана, используя любую программу с функцией snapshot, то есть фотографирования части экрана в файл. Таких программ много, подробности можно прочитать в моей статье http://kohnvict.narod.ru/Data/advice.htm.

Java - апплет содержит несколько готовых программ общего назначения, в частности таблицу Менделеева в виде книги, а также мои научные программы для коллег. Также каждый может скопировать и выполнить свою программу.

Что такое ACL?

Невозможно в двух словах объяснить, как работает космический корабль, слишком там много деталей. Такая же проблема с языком программирования, который рассчитан на то, чтобы делать все на свете. Когда я начинал, то не имел никакого опыта, да тогда вообще все было в очень ограниченном ассортименте. Поэтому я ничего не читал, а все выдумал сам. Первый момент – никаких строк и меток. Символ конца строки ничего не значит. В эпоху ДОС это было революционно. Программа пишется таким образом, что как раз комментарии ничем не выделяются, они пишутся свободно, а вот сами команды языка программирования начинаются с символа (#). Этот символ для интерпретатора служит знаком, что пора начинать работу. После этого знака пишется имя команды. Принцип написания всех имен следующий: можно писать сколько угодно букв, но слитно (без пробелов). Реально команды различаются максимум по трех первым буквам, а часто даже просто по одной букве. Я не люблю много писать, поэтому мне не нравится даже Java, хотя я считаю ее лучшим языком программирования. Команды из одной или двух букв абстрактны, но это дело привычки. Любой язык хорош, который знаешь. Вот у меня фамилия Кон. И это намного удобнее, чем Константинополев. Итак, читается слово до пробела и выбирается три первые буквы или меньше, если их меньше.

Что дальше?

Каждая команда – это отдельный блок программы, который знает, умеет и выполняет свою работу. Но часто ему нужны данные, конкретизирующие задачу. Входные данные раделены на два типа. Первый тип – это целочисленные параметры, которые имеют имена. Сделано это исключительно для более удобного понимания программы. Параметры универсальные и используются разными командами, причем один раз определенный параметр сохраняет свое значение до следующего определения. Определять параметры можно после каждой команды в поле, заключенном в квадратные скобки. Есть дополнительно к целочисленным несколько текстовых параметров. А всю остальную информацию надо записывать как аргументы сразу после квадратных скобок. Аргументы задаются как во многих командах многих интерпретаторов, хоть в ДОС (батч-файлы), хоть в компиляторах и прочих программах с командной строкой. Потому язык и называется командным.

Самое интересное, что больше ничего и нет. Точнее нет грамматики языка. Одни команды. И вся грамматика тоже реализована через команды. Одна команда запускает огромную готовую программу, другая реализует грамматику, но структурно они неразличимы. Возникает законный вопрос – а как делать вычисления? А где циклы, условия, логика? Ведь языки программирования, как и разговорные языки – как ни пиши, а предмет то один и тот же. Все верно. Но у меня «быстрый» язык, он кстати в первой редакции назывался quick. Для него главное – запускать готовые блоки, а все остальное вспомогательное. Поэтому все остальное реализовано по минимуму. Переменные все реальные и их имена короткие – одной буквой от (a) до (z), затем от (A) до (Z) и еще три символа ($), (%), (&). Можно писать одну букву и одну цифру, например (z1=x3;). И все – больше переменных нет, а зачем много? Массивы – есть один целый массив i(), один реальный массив r() и один массив символов (уникодов) t(). Индексы в

круглых скобках. Каждый из массивов имеет фиксированный (большой) размер, а все команды работают с частями этих массивов, причем есть специальные параметры (b [begin]) и (le [length]), которые как раз и выделяют индексы массивов, используемые в программе. То есть фактически вместо имени массива используется его начальный индекс в большом массиве. А зачем имена? Проблема выбора для некоторых людей – непреодолимое препятствие. А так все массивы плотно и динамически упаковываются в одном. Нет строк. Массив символов и строка – одно и то же. Более того, символы и числа – тоже одно и то же. Элементы символьного массива могут участвовать в вычислениях.

Как делаются вычисления? А очень просто. Открывается команда вычислений #cal или #c или просто #. Это единственная команда, которая может иметь нулевое имя. И все формулы вычислений являются ее аргументами. Она вычисляет и переопределяет переменные и массивы, это ее работа. В вычислениях можно использовать имена функций, кроме стандартных есть еще функции Бессела и интегралы Френеля. Пример: # z=sin(3.14*20/180); X1=5/exp(-2.7)+r(25); #% А что с циклами? Есть простой цикл, релизованный командой #rep N (repeat) с одним аргументом (числом повторений). Все команды после этой выполняются до команды #end , которая возвращает сканирование программы в самую первую команду после #rep и так ровно N раз. Все индексы и переменные должны переопределяться внутри цикла вручную. А условные циклы? Вот тут интереснее. Все сделано с помощью всего одной команды – и условия, и условные циклы. Зачем много? Опять проклятая проблема выбора. Просто есть еще одна команда цикла, она называется #case N.

Эта команда так же точно, выполняет все команды до команды #end. Но при одном условии, что ее аргумент совпадает со значением переменной (&). Если не совпадает, то вообще ничего не делается. В данном случае команда #end снова проверяет это же самое условие. Если не изменить переменную (&), то цикл будет делаться бесконечно. Поэтому внутри тела цикла ее надо изменить как раз тогда, когда надо. Таким способом можно сделать выбор: # &=2; #case 1 . . . #end | #case 2 . . . #end | #case 3 . . . #end | Вот в этом коде выполнится только многоточие после #case 2. Знак вертикальной черты (|) после #end через пробел означает автоматическую смену переменной (&) на значение 12345. Таким образом, тело цикла выполнится один раз. Перебор вариантов мы делаем. Но нам нужна логика. Программа должна уметь мыслить.

На это я могу авторитетно заявить. Не умеет компьютер мыслить. Все что он умеет – это сравнивать два числа на больше и меньше, а еще точнее – определять знак числа. И одной этой операции ему хватает, чтобы обыгрывать человека в шахматы. И не только в шахматы, а вообще делать разумную работу. Я не стал напрягаться и делать как у всех. Я просто ввел две новые операции < и > в математические вычисления. Результатом операции a<b является минимум из двух значений a и b. Аналогично a>b вычисляет максимум. И все, больше ничего не надо. Это позволяет писать полноценные программы, реализующие любую логику при переборе вариантов.

Заключение

Есть еще много интересных нюансов в языке и много такого, что позволяет ему иметь неограниченные возможности по созданию очень большой программы, включающей разные куски кода, написанные в разных файлах где угодно, в том числе и на серверах в интернете. Есть готовая научная графика, возможность работы с картинками, zip-архивами, делать любые операции с файлами и текстом. Но об этом можно очень долго

писать. Моя задача состояла в том, чтобы заинтересовать читателя обратиться на указанные выше сайты за более подробной информацией. Я надеюсь, что эту задачу я выполнил. Напоследок скажу чего нет. Нет работы с базами данных стандартного вида, хотя аппарат работы с небольшими базами данных есть. Я не работаю с базами данных и мне это не нужно. До сих пор я не освоил аппарат работы с xml файлами, каюсь. Но и это мне не нужно. Персональный язык программирования не обязан быть универсальным.

Ресурсы

. Дистрибутив JRE http://java.sun.com или http://www.java.com/ru/download/index.jsp
. Скриншот программы
http://img-fotki.yandex.ru/get/3904/kohnvict.12/0_24d12_9b5a6cf9_orig
. Вариант программы в виде Java-апплета http://vkacl.narod.ru/applets/00/vkACLa.htm