МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

**Дальневосточный федеральный университет**

▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬

ООП 010503.65 МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ И АДМИНИСТРИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ

**РЕФЕРАТ**

Сценарные языки программирования

Выполнил студент гр. c8504

А.B. Войцеховский

г. Владивосток

2012

Оглавление

**Элементы оглавления не найдены.**

# Введение

Сцена́рный язы́к или скри́птовый язы́к (англ. scripting language, в русской литературе принято название язык сценариев) — высокоуровневый язык программирования для написания сценариев — кратких описаний выполняемых системой действий. Разница между программами и сценариями довольно размыта. Сценарий — это программа, имеющая дело с готовыми программными компонентами [1].

Сценарии обычно интерпретируются, а не компилируются, хотя сценарные языки программирования один за другим обзаводятся JIT-компиляторам.

JIT-компиляция - Just-in-time compilation (компиляция «на лету») – технология увеличения производительности программных систем, использующих байт-код, путём компиляции байт-кода в машинный код непосредственно во время работы программы. Таким образом достигается высокая скорость выполнения по сравнению с интерпретируемым байткодом[1] (сравнимая с компилируемыми языками) за счёт увеличения потребления памяти (для хранения результатов компиляции) и затрат времени на компиляцию.

## Недостатки сценарных языков. Заблуждения и стереотипы

Итак, Лисп, по всей видимости, был праотцем сценарных языков. Но что же такое сценарные языки? Это, пожалуй, ключевой вопрос, ответить на который отнюдь не просто. В отношении сценарных языков уже сформировались ложные стереотипы. В частности, это касается таких критериев оценки, как компиляция/интерпретация кода, система типов, быстродействие, требования к памяти, надежность. Склонность к поддержке интерпретации, а не компиляции кода считается едва ли не первым признаком сценарных языков. Интерпретаторы проще в исполнении, нежели компиляторы, и к тому же покрывают более широкий спектр языков. Однако те же традиционные языки Лисп, Снобол, Пролог, Форт и даже «пограничный» Бейсик чаще всего имеют реализации в виде интерпретаторов. Тогда как среди сценарных языков, хоть и нечасто, но можно встретить компиляторы. Например, на платформе Microsoft .NET реализованы компиляторы Perl и Python, порождающие промежуточный MSIL-код, исполняющая (с динамической компиляцией) в рамках среды Common Language Runtime. В настоящее время все чаще используют смешанные схемы, когда код частично компилируется, частично интерпретируется (это свойственно сценарным языкам в индустрии компьютерных игр).

Принято считать, что сценарные языки либо имеют слабую типизацию, либо вообще бестиповые. Это справедливо для части языков, но далеко не для всех. Более того, можно привести примеры бестиповых языков, не являющихся сценарными, взять хотя бы тот же BCPL, прародитель языка Си.

Недостаточная надежность сценарных языков — тоже из разряда заблуждений. Так, яркий представитель сценарных языков 1990-х гг. — язык Python обладает средствами обработки исключений, построенными по образу и подобию аналогичного механизма в языке Modula-3. А ведь именно из него были заимствованы решения структурной обработки исключений (SEH), внедренные корпорацией Microsoft сначала в Си и Си++, а затем и в среду CLR (Common Language Runtime) платформы .NET.

Главная характерная черта для сценарных языков — динамическая природа, нередко позволяющая трактовать данные как программный код (и наоборот), а также простота освоения тех средств, которые буквально тут же дают видимый результат. Но это поверхностное наблюдение. Чтобы глубже разобраться в сути, необходимо выяснить, откуда пошли сценарные языки, для каких целей их создавали и что послужило катализаторами их развития.

Классификация языков

Существуют разные подходы к классификации языков программирования. Все они в той или иной мере упрощают реальную картину и охватывают лишь отдельные характеристики языков. Сложность классификации понятна: 50 лет эволюции языков программирования привели к тому, что взаимопроникновение концепций языков, которые используют различные модели и парадигмы, достигло едва ли не своего апогея. Почти каждый новый язык представляет собой «гремучую смесь» разных концепций и механизмов. Одной из наиболее примечательных является классификация моделей языков, предложенная Дж. Бэкусом в 1977 г. В соответствии с ней выделяются три категории языков (табл. 1):

1. Простые операционные модели (языки, основанные на конечных автоматах, машине Тьюринга);
2. Аппликативные модели (языки на основе лямбда-исчисления Чёрча, системы комбинаторов Карри, чистого Лиспа);
3. Модели фон Неймана (традиционные языки программирования).

С точки зрения такой классификации сценарные языки ближе всего к категории B. Если составить несколько таблиц, куда будут сгруппированы наиболее значимые и известные языки, которые по тем или иным причинам можно назвать сценарными, то получится четыре класса таких языков:

1. командно-сценарные;
2. прикладные сценарные;
3. языки разметки;
4. универсальные сценарные.

Командно-сценарные языки

Появились ещё в 1960-х годах для управления заданиями в операционных системах. Из языков того времени наиболее известен JCL для OS/360. В этот класс входят языки пакетной обработки англ. batch language и языки командных оболочек, например, sh, csh для Unix. Эти языки чаще всего используются в пакетном режиме обработки[7].

* JCL
* sh
  + bash
  + csh
  + ksh
* Pilot[7]
* REXX
* AppleScript
* COMMAND.COM и cmd.exe
* Visual Basic Script

Например, язык AppleScript операционной системы MacOS имеет редактор Script Editor, который позволяет записывать действия по мере их выполнения пользователем в системе в файл сценария (текстовый файл) и оформлять в виде исполняемой программы. Такой подход позволяет составлять простейшие сценарии непрограммирующим пользователем[8].

Прикладные сценарные языки

Сценарные языки этого типа начали появляться в 1980-е годы, когда на промышленных персональных компьютерах стало возможным интерактивное общение с ОС. В клиент-серверной архитектуре такие языки работали в клиентской части программного обеспечения[7].

* AutoLISP
* Emacs Lisp
* ERM
* Game Maker Language
* MQL4 script
* UnrealScript
* VBA
* LotusScript[7]

К этой категории можно отнести JavaScript и его диалекты (JScript, ECMAScript)[7].

Языки разметки

Основная статья: Язык разметки

Несколько особняком среди сценарных языков стоят языки разметки (тегированные языки). Главная характерная черта этих языков — встраивание специальных кодов (тегов) в обычный текст не только для целей структурирования и форматирования, но и для определения динамического поведения[7].

* GML
* TeX
* SGML
* XML

Родственны этим языкам Postscript и RTF[7].

Универсальные языки

Этот тип сценарных языков наиболее известен, особенно в применении к программированию для веба. Языки этого типа стали возникать с 1990-х годов[7].

* Tcl (Tool command language)
* Lua
* Perl
* PHP
* Python
* REBOL
* Ruby

Следует заметить, что многие языки этой категории имеют более широкое применения, чем в качестве просто языков сценариев.

Командно-сценарные языки

Командно-сценарные языки (табл. 2) зародились еще в 1960-х гг., когда возникла острая потребность в языках, обеспечивающих управление программами, иначе говоря, языках управления заданиями. Среди них наиболее известен JCL, разработанный для OS/360. Менее знаком Pilot, ставший, пожалуй, первой ласточкой среди сценарных языков. Он поддерживает всего два типа данных (строки и числа) и имеет крайне ограниченный набор команд (TYPE, ACCEPT, MATCH, JUMP, USE, COMPUTE, END, YES, NO).

команд CLI (command language interpreter), так называемые языки пакетной обработки (batch language) и языки для построения системных командных оболочек (яркий пример — sh, csh и их вариации для UNIX). Как правило, такие языки ориентируются не на интерактивный, а на пакетный режим обработки, когда участие человека на этапе выполнения сведено к нулю и все работает в непрерывном потоке. Эти языки не только могли непосредственно взаимодействовать с соответствующей операционной системой, но и снабжались средствами грамматического разбора программ и трансформации данных. С их помощью можно было создавать различные программные фильтры, используемые, в частности, в конвейере (pipe) системы UNIX. В число известных языков такого типа входят Awk, впервые появившийся в AT&T UNIX Version 7, а теперь ставший частью стандарта POSIX Command Language and Utilities. К ключевым особенностям языка, нашедшим впоследствии широкое применение в среде универсальных сценарных языков, можно отнести механизм регулярных выражений, без которого разбор текста производится не очень эффективно.

Языки этого класса ориентировались также на обработку системных событий, генерирование текста и высокоуровневый доступ к базам данных. Здесь стоит отметить язык RPG (Report Program Generator). Он до сего времени успешно применяется для создания отчетов из корпоративных БД, работающих преимущественно на мэйнфреймах (RPG/400 для компьютеров IBM AS/400).

Наиболее активно из языков этого класса в наши дни используется Rexx, созданный в исследовательских лабораториях IBM. По набору средств он мало чем отличается от универсальных сценарных языков, однако выполнен в виде классического блочно-структурированного процедурного языка и предназначен преимущественно для интеграции и расширения функциональности приложений.

Прикладные сценарные языки

Прикладные сценарные языки (табл. 3) зарождались в 1980-е гг., в эпоху появления промышленных ПК, когда на первый план стали выходить задачи интерактивного общения с ОС, а также доступа к данным электронных таблиц и БД. Отличительная особенность сценарных языков данного класса — ориентация на клиентскую часть ПО.

Использование объектной модели в языках данного класса уже становится нормой, а не исключением. Их еще нельзя назвать полноправными языками ООП, однако они в значительной мере стараются воспользоваться удобствами объектного подхода.

Среди прикладных сценарных языков резко выделяется Visual Basic, в том числе и такая его разновидность для офисного программирования, как VBA (Visual Basic for Applications). Visual Basic — это тот самый «пограничный» язык, который скорее относится к сценарным, чем к традиционным. Он во многом задал тон такому классу языков, как прикладные сценарные. Более того, работа с пользовательским интерфейсом и встраивание программных компонентов (VBX, OCX, ActiveX) стали едва ли не визитной карточкой данного языка. Стоит заметить, что в ходе эволюции VBA поглотил другие специфические языки, в частности Word Basic и Excel Macro Language, взяв на себя их задачи. Под его влиянием были созданы такие языки, как VBScript (особый диалект языка Visual Basic, ориентированный на создание OLE-компонентов и на работу в рамках браузеров) и LotusScript (своего рода аналог языков VBA и CorelScript, предназначенный для решения задач автоматизации офиса в рамках Lotus Notes).

Несколько особо в этом ряду стоит JavaScript, ставший стандартом де-факто в Web-программировании при реализации клиентской части. Его прототипом был язык LiveScript, являвшийся частью серверного продукта LiveWire компании Netscape и первоначально встроенный в Netscape Navigator 2.0. После появления языка Java корпорации Sun Microsystems он начал играть роль самодостаточной надстройки над этим языком, и его название сменилось на JavaScript. Диалекты этого языка — JScript корпорации Microsoft и ECMAScript (стандарт ECMA-262).

Из новичков в данном классе языков упомянем экспериментальный язык Pnuts, основная идея которого — дать в рамках сценариев наиболее полный доступ к Java API. Его можно использовать для самых разных задач, но прежде всего для оперирования компонентным ПО (подробнее см.: http://javacenter.sun.co.jp/pnuts).

Языки разметки

Языки разметки, или тегированные языки (табл. 4), стоят несколько поодаль от магистральной линии развития сценарных языков. Им ближе всего по своей природе системы макрообработки (всевозможные макропроцессоры), столь популярные в 1960—1970-е гг. Их главная отличительная черта — встраивание специального кода (в виде обособленных «команд» — тегов) непосредственно в обычные тексты. Им родственны такие языки, как Postscript и RTF (чаще воспринимаемый просто как особый формат представления документов). Теги стали использоваться для самых разных целей: для отделения структуры информации от ее содержания, для вкрапления команд форматирования и даже для задания динамического поведения встроенных в документ интерактивных объектов.

Идея отделения структуры информации от содержания возникла давно. Но, пожалуй, первым осознанным решением стал запуск проекта GenCode. В сентябре 1967 г. Уильям Танниклифф, председатель Комитета по композиционным решениям ассоциации GCA (Graphic Communications Association), предложил провести четкую границу между содержанием и форматом представления информации. Руководитель GCA Норман Шарпф принял решение приступить к реализации соответствующего проекта GenCode.

Под влиянием GenCode в 1969 г. американский ученый Чарльз Гольдфарб возглавил работу исследовательской группы в IBM, целью которой была проработка принципов интегрированных информационных систем в области законодательства. Плодом усилий этого коллектива, куда входили также Эдвард Мошер и Реймонд Лори, стал GML — обобщенный язык разметки (Generic Markup Language, Goldfarb-Mosher-Lorie). Многие решения этой группы нашли применение в различных издательских системах IBM. На основе GML и идей системы Scribe, разработанной Брайаном Рейдом, Международная организация по стандартизации (ISO) разработала метаязык SGML (стандарт ISO-8879:1986).

Наиболее значительными достижениями в области языков разметки стали TeX, HTML и XML. Язык TeX (1979) Дональда Кнута на три года опередил Postscript, созданный Джоном Уорноком и др. в компании Adobe и предназначенный для точного описания внешней формы документов с композицией произвольной сложности. В отличие от низкоуровневого Postscript, язык TeX ориентировался на работу пользователей, не имеющих навыков программирования. Наиболее широко этот язык стал применяться в научной среде, где предъявляются самые высокие требования к качеству построения формул сложной структуры.

Язык HTML, с появлением которого понятие «гипертекст» стало простым и обыденным, создавался на базе SGML путем максимального упрощения его структуры и свойств. Вряд ли имеет смысл описывать его подробно: это основной язык представления информации в Web-среде, включая Интернет.

Метаязык XML, созданный во многом с подачи Джона Босака (Sun Microsystems), руководителя рабочей группы SGML ERB, также основывался на SGML, но для него характерно куда более бережное отношение к идеям своего предка, да и выполняет он совсем иную роль, нежели HTML. Это своего рода язык транспортирования и промежуточного хранения данных при обмене ими между разнородными и распределенными системами. На его основе можно проводить сколь угодно сложные преобразования документов и текстовой информации, а главное, в унифицированном виде хранить данные реляционно-иерархической структуры, в том числе по настройкам и программированию компонентов. Это, правда, не мешает использовать его и не совсем по назначению — для задания динамики поведения всевозможных объектов. В 2001 г. появилась ревизия HTML, которая получила название XHTML, где были учтены требования XML.

* машины (системный слой, модель);
* преобразователи (связующий слой, контроллер);
* сценарии (прикладной слой, вид).

Под машиной может пониматься любая сервисная машина и приравненный к ней компонент. Преобразователи могут быть универсальными и специализированными. Универсальные имеют общий характер (привычные библиотечные модули); специализированные соединяют машины и сценарии. Их удобнее всего выполнять на языках ООП. Деление на три вида совсем не обязательно должно предусматривать соответствие машин, скажем, уровню модели. Машина может использоваться и на уровне контроллера, если это удобно для данной задачи.

Взаимосвязь между программными и сценарными языками можно проследить и на модели Ершова [12]. В соответствии с ней выделяются три взаимодополняющих друг друга вида программирования:

* синтезирующее (формирование программных фрагментов/компонентов);
* сборочное (сборка программы из готовых фрагментов/компонентов);
* конкретизирующее (адаптация многопараметрической программы к особым условиям ее применения).

Как нетрудно заметить, традиционные языки в значительной степени доминируют в синтезирующем программировании, тогда как сценарные — в конкретизирующем. Их соприкосновение происходит в сборочном программировании, которое становится пограничной зоной двух разных языковых миров. До сих пор остается непонятным, является ли программирование по своей природе индуктивным или дедуктивным процессом. Это еще одна точка различия программных и сценарных языков. Первые исповедуют скорее индуктивный подход, тогда как вторые — дедуктивный.

Как известно, многочисленные попытки создать единый всеобъемлющий язык программирования закончились неудачно. Но это касалось именно языка, а не языковой среды и не лингвистической системы. Здесь важно отметить, что А.П. Ершов был автором не менее интересной идеи — лексикона программирования, который он понимал как «лингвистическую систему с фразовой структурой, содержащую в себе формальную нотацию для выражения всех общезначимых конструкций, употребляемых при формулировании условий задачи, при синтезе и преобразовании программ». Сценарные языки на их нынешнем этапе развития, пожалуй, ближе всего к этой идее.

Что касается формы сценарных языков (их синтаксиса), то немалое влияние на нее оказал и продолжает оказывать язык Си. Заимствование содержательных идей сценарными языками ведется по широкому фронту:

* языки функционального программирования (Лисп, Scheme);
* языки обработки строк (Snobol, Icon);
* объектно-ориентированные языки (Smalltalk, Java, Eiffel, C#);
* языки управления заданиями, командные языки (csh, Rexx);
* языки управления средой (Tcl, VBA);
* языки разметки/макрообработки (SGML, TeX);
* языки моделирования дискретных систем (GPSS, SIMSCRIPT, SLAM II).

В чем же еще проявляется воздействие программных (традиционных) языков на проектирование и развитие языков сценарных? Здесь можно выделить три направления:

1. использование ООП (практически любой новый сценарный язык поддерживает объектную модель);
2. использование идей старых языков (Лисп, Снобол; языки функционального программирования и языки обработки строк по своей динамической природе наиболее близки сценарным);
3. появление языка Java (ряд сценарных языков наиболее тесно интегрируется именно с Java).

# Литература

1. Сузи Р. Сценарные языки: Python. «Мир ПК», 9, 2001
2. Природа и эволюция сценарных языков - № 11, 2001 | Мир ПК | Издательство «Открытые системы» - Режим доступа: <http://www.osp.ru/pcworld/2001/11/162500/>