Визитка

КОНСТАНТИН РАЧАЕВ, *ведущий инженер, ОАО «Ростелеком»*, *k.rachaev@mail.ru*

Как приручить ДРАКОНа?

ДРАКОН – это визуальный алгоритмический язык программирования и моделирования, представляющий алгоритмы по принципу «взглянул – и сразу понял»

История рождения языка

В 1986 году для решения задач, стоявших перед ракетно-космической отраслью, был начат проект по разработке нового языка. Причиной создания послужила необходимость в разработке бортового программного обеспечения и программного обеспечения наземных испытаний корабля «Буран». По оценкам специалистов, для этой работы требовалось несколько тысяч программистов. После изучения задачи специалисты института прикладной математики им. М. В. Келдыша РАН создали проблемно-ориентированные языки, основанные на терминах, понятиях и форме представления алгоритмов управления и испытаний, используемых разработчиками корабля. Так на свет появились ПРОЛ2, ДИПОЛЬ, ЛАКС. Но из-за их узкой специализации было принято решение разработать один язык, способный решать задачи не только трех вышеназванных, но и задачи, выходящие за круг традиционного программирования, например, облегчение взаимодействия и понимания между представителями различных специальностей и отраслей. Разработка завершилась через 11 лет, так появился Дружелюбный Русский Алгоритмический Язык, Который, Обеспечивает Наглядность, или сокращенно – ДРАКОН.

В 1996 году на базе ДРАКОНа была построена автоматизированная технология проектирования алгоритмов и программ (CASE-технология) под названием «ГРАФИТ-ФЛОКС».

Особенности языка

ДРАКОН – это визуальный алгоритмический язык программирования и моделирования, при разработке которого использовались стандарты ГОСТ 19.701–90 и ISO 5807–85, а, самое главное, особое внимание уделялось повышению восприятия алгоритма или, говоря по-другому, учитывались когнитивные характеристики человека. Человеческий мозг в основном ориентирован на визуальное восприятие, и люди получают информацию при рассмотрении графических образов быстрее, чем при чтении текста. Алгоритмы, реализованные на ДРАКОНе, являются более наглядными и понятными. А эргономичные методы, применяемые в языке, существенно улучшают восприятие алгоритмов. Традиционные языки в этом плане не до конца учитывают специфику зрительных образов. А так как наглядная и понятная схема позволяет легко выявить ошибки в алгоритме, то чем больше ошибок будет выявлено на этапе алгоритмизации, тем надежнее будет разрабатываемое ПО.

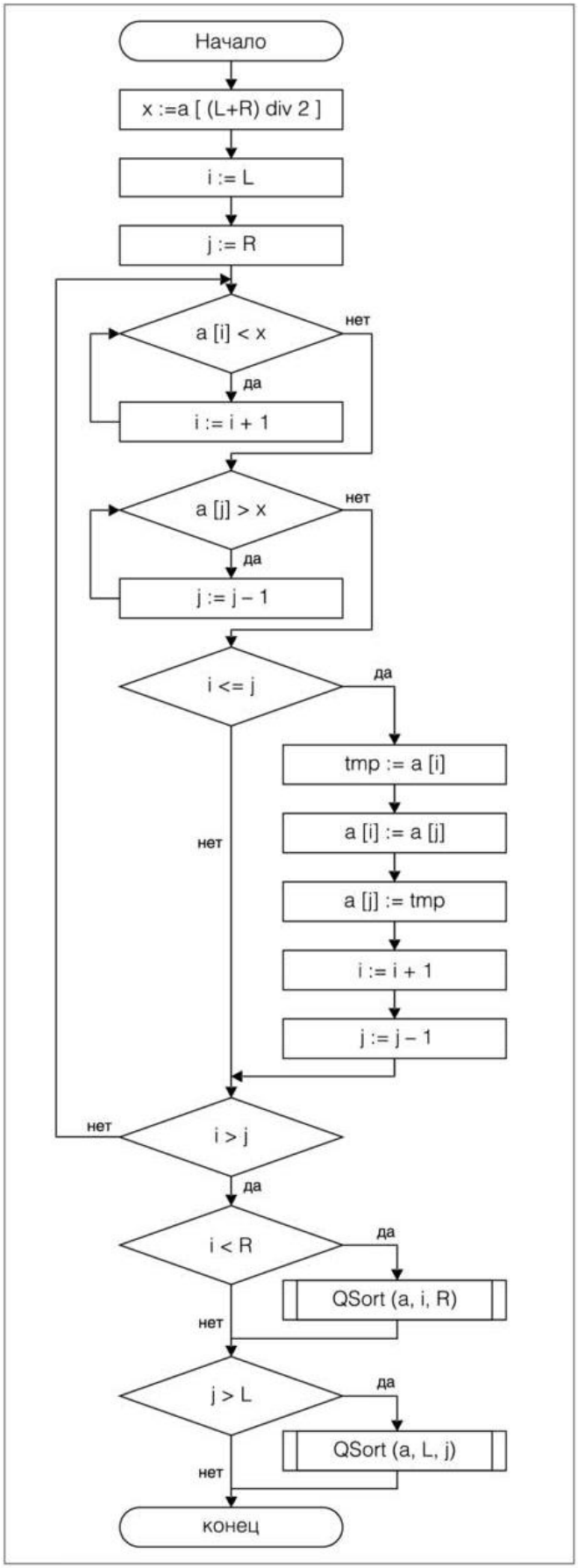
Язык ДРАКОН выполняет две принципиально разные функции. С одной стороны, он позволяет разрабатывать алгоритмы практических прикладных задач специалистам-непрограммистам, которые знают постановку задачи и владеют материалом в практической области. А, с другой стороны, для программистов он служит языком программирования. Исходя из этих двух функций ДРАКОН можно использовать в качестве языка общения между непрограммистами и программистами. В качестве примера предлагаю посмотреть видеоурок по разработке программы управления автоматическим дверным замком [1], где показаны процесс создания алгоритма на псевдоязыке, а затем и кодирование алгоритма.

«Буквами» языка являются графические элементы, называемые иконами, а объединение нескольких икон – макроиконой. В языке используются два типа элементов: графоэлементы – графические фигуры и текстоэлементы – текстовые надписи, расположенные внутри или снаружи графоэлементов. Соответственно выделяется и два типа синтаксиса. Графический синтаксис как набор правил размещения графоэлементов и правил их связи и текстовый синтаксис, который задает алфавит символов, правила их комбинирования и привязку к графоэлементам.

ДРАКОН фактически является семейством языков, которое может включать практически неограниченное число языков. Эти языки семейства имеют одинаковый графический синтаксис, а отличаются своим собственным текстовым синтаксисом. Язык ДРАКОН можно разделать на две группы.

- **ДРАКОН-1** – это графический псевдоязык, в качестве текстоэлементов используется естественный язык.
- **ДРАКОН-2** – графический алгоритмический язык для разработки алгоритмов и программ реального времени.

Рисунок 1. «Обычная» блок-схема алгоритма быстрой сортировки



Семейство также включает в себя так называемые гибридные графические языки, например, ДРАКОН-C, ДРАКОН-Java, ДРАКОН-Lua и другие. Гибридный язык получается за счет объединения по определенным правилам графического синтаксиса ДРАКОНа и текстового синтаксиса сопрягаемого языка.

При описании элементов языка вы не увидите иностранных слов. В нем применяются простые русские слова, понятные даже нетехническому специалисту. Эти слова рождают соответствующие образы, облегчающие понимание как схемы, так и способствующих интуитивному запоминанию понятий и правил самого языка ДРАКОН.

Аналогом дракон-схем являются блок-схемы, диаграммы поведения языка UML. Но, например, в отличие от блок-схем дракон-схемы имеют средства для описания работы в реальном времени.

На официальном сайте [2] можно найти всю необходимую информацию и документацию по языку. Площадкой для обсуждений различных тем, связанных с ДРАКОНом, является форум [3]. Но на форуме можно найти не только информацию, относящуюся к языку, но и различные публикации по космической отрасли – статьи, книги, автобиографии известных в этой сфере людей.

Элементы языка

Давайте рассмотрим элементы языка ДРАКОН на примере алгоритма быстрой сортировки, разработанной английским информатиком Чарльзом Хоаром. С самим алгоритмом можно подробно ознакомиться в [4, 5].

Кратко алгоритм можно описать так:

- Выбрать из массива элемент, называемый опорным.
- Сравнить все остальные элементы с опорным и переставить их в массиве так, чтобы элементы, меньше опорного, находились слева от него, а больше – справа.
- Для отрезков «меньших» и «больших» значений выполнить рекурсивно ту же последовательность операций, если длина отрезка больше единицы.

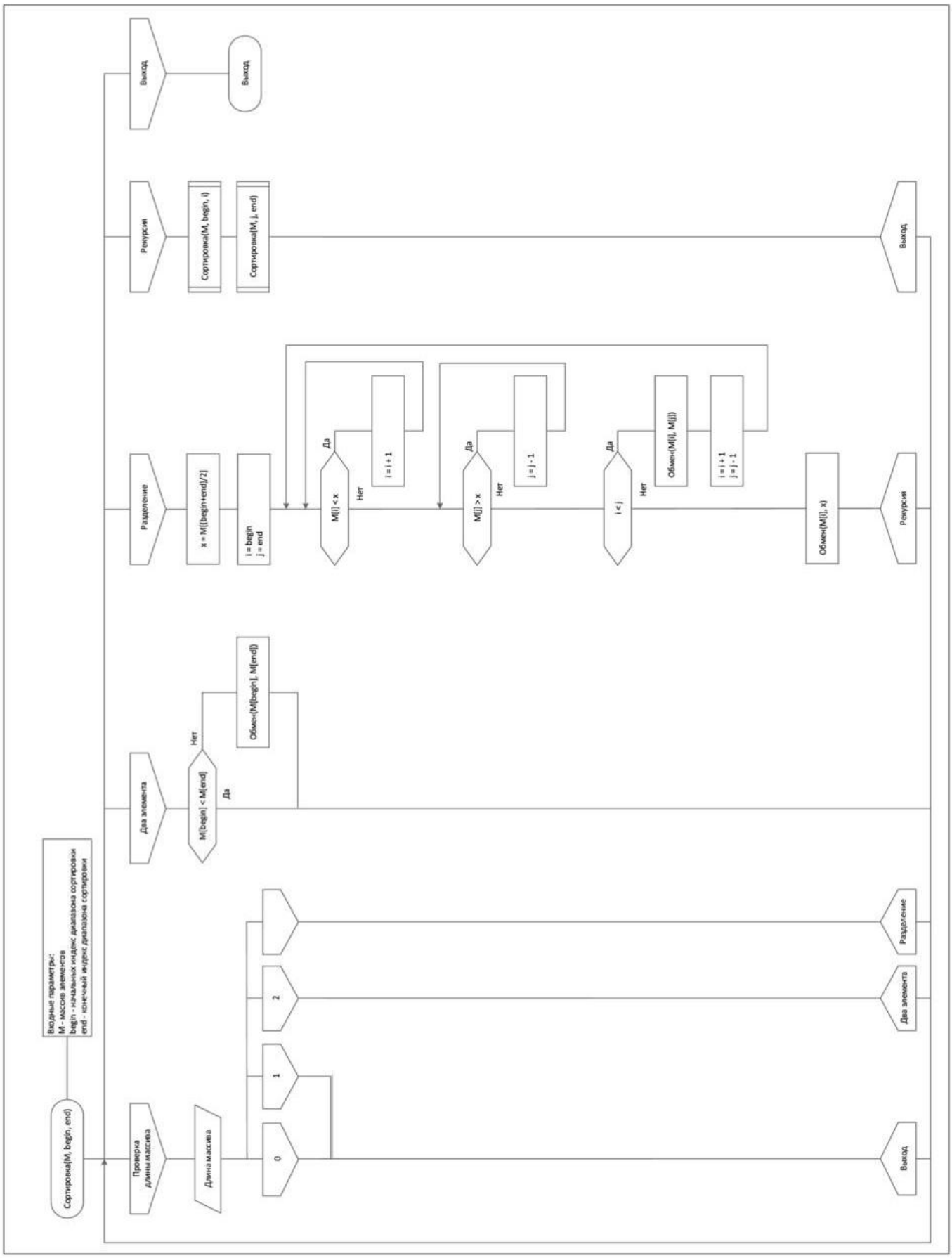
Поискав на просторах Интернета блок-схему алгоритма быстрой сортировки – их оказалось не так уж много и по количеству и по разнообразию, – нашел схему, приведенную на рис. 1. Назовем ее «обычной» блок-схемой.

Блок-схема, построенная по правилам языка ДРАКОН, называется дракон-схемой. Дракон-схема алгоритма быстрой сортировки приведена на рис. 2. На мой взгляд, «обычная» блок-схема уступает по наглядности и понятности дракон-схеме.

Давайте рассмотрим основные элементы этой схемы. Любая дракон-схема имеет икону «Заголовок», которая содержит понятное и точное название алгоритма, и икону «Конец» (в программах реального времени икона «Конец» может отсутствовать). При этом вертикальная линия, соединяющая иконы «Заголовок» и «Конец», называется Шампур. Согласитесь, рождается яркий и понятный образ.

В языке существует две разновидности дракон-схем: Примитив и Силуэт. И, прежде чем пояснить разницу между ними, нужно ввести понятие Ветка. Веткой называется часть дракон-схемы, образующаяся при делении алгоритма на смысловые части. Например, на нашей дракон-схеме выделены ветки «разделение» и «рекурсия». Таким образом,

Рисунок 2. Дракон-схема алгоритма быстрой сортировки



Силуэт – это дракон-схема, разделенная на ветки, а Примитив – это дракон-схема, не имеющая веток, у которой иконы «Заголовок» и «Конец» лежат на одной вертикали – Шампуре.

Входом ветки служит икона «Имя ветки», а выходом из ветки – икона «Адрес», которая передает управление только в начало одной указанной в ней ветки. При этом очередность работы веток зависит только от команд «Адрес», а не от расположения в схеме. Но, располагая ветки, надо руководствоваться правилом: «Чем правее, тем позже», то есть располагать их в той последовательности, в какой они включаются в работу. Ветки силуэта можно рассматривать как состояния, а сам силуэт, соответственно, можно рассматривать как конечный автомат.

Использование языка ДРАКОН в виде гибридного языка, наверное, возможно только в каких-то узких направлениях

Шапкой в языке ДРАКОН называют верхнюю часть схемы, которая включает «Заголовок» алгоритма и комплект икон «Имя ветки». В нашем случае шапкой являются Сортировка (M, begin, end) и заголовки веток. Очень полезна иконка «Формальные параметры», которая соединена с «Заголовком».

Согласно [6] ДРАКОН предоставляет трехэтапный метод познания алгоритма. На первом этапе, анализируя шапку, читатель узнает назначение алгоритма и его деление на смысловые части (ветки). На втором осуществляет углубленный анализ каждой ветки. На третьем производит разбор взаимодействия веток.

Каждая дракон-схема содержит шампур-блок – часть схемы, имеющая один вход сверху и один выход снизу, расположенные на одной вертикали, например, икона «Действие» со значением Обмен (M[begin], M[end]). Шампур в этом случае выступает как путеводная нить, облегчающая понимание схемы.

Маршрут – это путь, ведущий от начала до конца алгоритма. При этом можно выделить главный и побочные маршруты. Существует правило главного маршрута, которое гласит, что главный маршрут алгоритма должен идти по шампуру. Соблюдение этого правила делает схему зрительно упорядоченной и понятной. Побочный маршрут – это любой маршрут разветвленного алгоритма за исключением главного. При этом побочные маршруты изображаются только справа от шампура по такому же принципу, что и ветки: «Чем правее, тем хуже».

Еще одна икона, применяемая в нашем схеме, это икона «Вопрос». Внутри нее пишется вопрос, на который можно ответить строго либо «Да», либо «Нет». Икона «Вопрос» имеет один вход сверху и два выхода: вниз и вправо. Выход влево запрещен для использования.

Инверсии логических условий не всегда положительно сказываются на понимании алгоритма. Например, язык C требует выхода из цикла только по false, и это явно противоречит рекомендациям эргономики, согласно которым отрицательные вопросы нежелательны, потому что провоцируют ошибки. В языке ДРАКОН знак логического отрицания всегда можно исключить из дракон-схемы. Для этого надо поменять местами слова «да» и «нет» на выходах иконы «Вопрос». При этом иконы, находящиеся в плечах развилки, следует оставить на своих местах.

Икона «Вопрос» образует макроикону «Развилка», которая, помимо вопроса, содержит еще правое и левое плечо развилки и точку слияния – место, где соединяются оба плеча. Таким образом, «Развилка» имеет один вход сверху и один выход снизу.

В дракон-схемах применяются циклы. Выделяют следующие циклы: обычный (do-while, while), переключающий, цикл ДЛЯ (это, по сути, FOR из C), веточный цикл, цикл ЖДАТЬ.

В нашей дракон-схеме применяются обычные циклы ПОКА, которые являются составным графическим оператором и содержат иконы «Вопрос» и «Петля цикла». Шампур цикла проходит через икону «Вопрос», поэтому условие продолжения цикла соответствует правому выходу иконы «Вопрос», а условие окончания цикла – нижнему.

Для того чтобы изобразить разветвление в алгоритме можно использовать иконы «Вопрос» или составной графический оператор «Переключатель». В нашей схеме при разветвлении алгоритма по длине массива применяется «Переключатель». «Переключатель» имеет один вход и один выход, содержит одну икону «Выбор» и несколько (две и более) икон «Вариант».

Необычным циклом является Веточный цикл, который в отличие от других циклов используется только в Силуэте. Что же такое веточный цикл? Это повторное исполнение одной и той же ветки. Чтобы построить веточный цикл, нужно написать в иконе «Адрес» название данной ветки (или более левой ветки, если руководствоваться правилом «Чем правее, тем позже»). При этом используются маленькие черные треугольники в иконах «Имя ветки» и «Адрес», позволяющие легко заметить веточный цикл.

В нашей схеме применяется еще одна икона – икона-вставка, которая по принципу действия эквивалентна оператору вызов процедуры. За счет нее в нашем алгоритме реализуется рекурсия.

Соединительные линии используются для связывания икон и могут быть либо горизонтальными, либо вертикальными. При этом наклонные линии не допускаются, а пересечения и обрывы соединительных линий запрещены. Чтобы алгоритм был удобным для чтения, количество изломов соединительных линий должно быть минимальным (Правило устранения изломов).

Еще одной примечательной особенностью языка является возможность проектировать управляющие системы за счет использования операторов реального времени. Для этого применяются пять икон: «Пауза», «Период», «Пуск таймера», «Синхронизатор» и «Параллельный процесс». Все иконы, за исключением «Параллельного процесса», построены на основе одной и той же геометрической фигуры – перевернутой равнобедренной трапеции, – что сделано не случайно, подобным визуальным способом они «сгруппированы», что также сказывается на понимании алгоритма.

Хотя следующие методы не отображены на нашей дракон-схеме, я вкратце расскажу о них. Для улучшения

эргономичности алгоритма можно применять равносильные преобразования, такие как «вертикальное объединение» и «горизонтальное объединение». Они позволяют сократить число вертикалей, но иногда возникают противоречия с правилом главного и побочных маршрутов. В таком случае работает принцип паритета: правило главного и побочных маршрутов имеет более высокий приоритет, нежели стремление уменьшить число вертикалей.

Как вы уже, наверное, заметили, в дракон-схемах практически не используются стрелки для изображения направления движения по схеме, следующий элемент всегда расположен ниже предыдущего. При построении схемы работает правило лаконичности, которое гласит, что зрительный образ алгоритма должен быть лаконичным, все ненужные лишние детали должны быть отсечены.

Исчисление икон

Необычной составляющей языка ДРАКОН является то, что разработчики изначально преследовали цель строгой формализации визуального синтаксиса блок-схем. Для этого они рассматривают свой язык с позиции математической логики. Основным объектом изучения в математической логике являются различные исчисления [7]. В понятие исчисления входят такие основные компоненты, как:

- язык (формальный) язык;
- аксиомы исчисления;
- правила вывода.

Математическая логика и ее основные понятия (исчисление, логический вывод и другие) сформировались в рамках текстовой парадигмы. Но для ДРАКОНа вводятся визуальные аналоги этих понятий, и на их основе строится исчисление икон. Любая абстрактная дракон-схема представляет собой теорему, которая строго выводится (доказывается) из двух аксиом, каковыми являются заготовка-примитив и заготовка-силуэт. В исчислении икон семантика тривиальна. Различные блок-схемы могут быть истинными или ложными. Блок-схема называется истинной, если она либо аксиома, либо выводится из аксиом с помощью правил вывода, то есть является теоремой. В противном случае блок-схема ложная.

Таким образом, все правильно построенные абстрактные дракон-схемы (теоремы) истинны. А неправильно построенные схемы, не удовлетворяющие визуальным правилам языка ДРАКОН, считаются ложными. Подробнее об исчислении икон можно познакомиться в [6].

Области применения языка

Язык ДРАКОН можно применять в самых разных областях деятельности, например, в медицине, в атомной промышленности, в биологии, в учебных заведениях. Такое разнообразие отраслей возможно благодаря тому, что дракон-схемы позволяют изобразить решение любой, сколь угодно сложной процедуры в предельно ясной, наглядной и доходчивой форме. Это дает возможность значительно сократить интеллектуальные усилия персонала, необходимые для зрительного восприятия, понимания и безошибочного решения проблем. В [6] есть пример дракон-схемы по математике по упрощению алгебраического выражения, которое, по-моему, очень наглядно показывает, как решаются подобные математические задачи, и данный формат записи можно использовать в сходных задачах.

Ну и, естественно, язык ДРАКОН применяется в ракетно-космических программах, ради которых он и создавался изначально. Он используется в таких крупных космических программах, как международный проект «Морской старт», разгонный блок космических аппаратов «Фрегат», модернизированная ракета-носитель тяжелого класса «Протон-М» и других.

Дракон-редакторы

Для того чтобы нарисовать дракон-схему, ничего, кроме ручки и бумаги или графического редактора, не требуется. Но для удобства создания алгоритмов разработаны специальные программы – дракон-редакторы. Такие дракон-редакторы позволяют проверить корректность созданной дракон-схемы на соответствие правилам языка и транслировать алгоритм в исходный код определенного языка. На данный момент существует два таких редактора.

Первый редактор – это «ИС Дракон» [8]. Для его применения необходимо выполнить установку программы, которая является платной. Без установки программа работает в режиме просмотра файлов, при этом сохранение файлов не производится. Но в течение ознакомительного периода работает без ограничения функциональности.

Второй редактор – DRAKON Editor [9] – является бесплатной программой. Для ее работы необходимо будет установить интерпретатор Tcl/Tk.

• • •

По моему мнению, использование языка ДРАКОН в виде гибридного языка, наверное, возможно только в каких-то узких направлениях. Очень хорошо он подходит именно в качестве языка для проектирования алгоритмов за счет того, что структурирует процесс мышления при создании алгоритма и позволяет наглядно и понятно изобразить его графически. При этом в плане изучения это довольно простой язык. Для быстрого освоения достаточно просмотреть рисунки дракон-схем с поясняющими комментариями. EOF

[1] Использование языка ДРАКОН для программирования микроконтроллеров с помощью «ИС Дракон» –http://www.youtube.com/watch?v=Ua9dUUONjdk&feature=youtu.be.

[2] Официальный сайт языка ДРАКОН – http://drakon.su.

[3] Форум языка ДРАКОН – http://forum.oberoncore.ru/viewforum.php?f=77.

[4] Кормен Томас. Алгоритмы построения и анализ. Второе издание. – 2005.

[5] Седжвик Роберт. Фундаментальные алгоритмы C++. Третье издание. – 2001.

[6] Пароннджанов В.Д. Учись писать, читать и понимать алгоритмы. Алгоритмы для правильного мышления. Основы алгоритмизации. – М.: «ДМК Пресс», 2012.

[7] Ершов Ю.Л., Палютин Е.А. Математическая логика. Второе издание. – М.: «Наука». Гл. ред. физ.-мат. лит. – 1987.

[8] Редактор «ИС Дракон» – http://drakon.su/programma_is_drakon.

[9] Редактор DRAKON Editor – http://drakon-editor.sf.net.

Ключевые слова: ДРАКОН, язык программирования, моделирование, алгоритмы.