21 мая 2013 года в Институте проблем управления Российской академии наук состоялась Международная научно-техническая конференция «Системы и комплексы автоматического управления летательными аппаратами», посвященная 105-летию со дня рождения академика Н.А.Пилюгина.

По итогам конференции выпущен электронный сборник научно-технических докладов. Ниже представлен доклад В.Д.Паронджанова, опубликованный в этом сборнике.

Визуальный язык ДРАКОН и его применение в ракетно-космической отрасли, медицине и других областях

В.Д. Паронджанов В.Д.
ФГУП НПЦ АП им. акад. Н.А. Пилюгина, Москва
vdp2007@bk.ru

Введение

ДРАКОН *(Дружелюбный Русский Алгоритмический язык, Который Обеспечивает Наглядность)* — визуальный алгоритмический язык программирования и моделирования. Был разработан в рамках космической программы «Буран». Разработка языка велась с 1986 года при участии НПЦ АП и Института прикладной математики РАН им. М. В. Келдыша. Язык построен путём формализации, эргономизации и неклассической структуризации блок-схем алгоритмов и программ, описанных в ГОСТ 19.701-90 и ISO 5807-85, а также для разработки программ реального времени [1].

Основной задачей разработчиков было создание единого универсального языка программирования и моделирования, который своей доступностью и мощностью способен заменить специализированные языки: ПРОЛ2 [2] (для разработки бортовых комплексных программ Бурана), ДИПОЛЬ [2] (для создания наземных программ Бурана) и ЛАКС (для моделирования).

Работы по созданию языка ДРАКОН были в основном закончены в 1996 году (спустя 3 года после закрытия программы «Буран»), когда была разработана автоматизированная система проектирования программных систем (CASE-технология) ГРАФИТ-ФЛОКС. Язык ДРАКОН используется начиная с 1996 года во многих крупных космических программах: «Морской старт», «Фрегат», «Протон-М» и др.

В документальном видеофильме «Жирограф и ДРАКОН Пилюгина», снятом на Телерадиостудии Роскосмоса, говорится:

> «Во время работы над «Бураном» был придуман язык технических символов — ДРАКОН: «Дружелюбный русский алгоритмический, который обеспечивает наглядность». Он и стал своеобразным инструментом взаимопонимания в Пилюгинском коллективе инженеров и конструкторов. Разработки академика Пилюгина и сегодня применяются в современной ракетной технике. Тяжелые «Протоны» уходят в небо с его системой управления, а грозные ракетные комплексы “Тополь-М” обеспечивают оборону страны» [3].

ДРАКОН можно определить как общедоступный визуальный язык, предназначенный:

— для систематизации, структуризации, наглядного представления и формализации процедурных (императивных) знаний;

— для описания структуры человеческой деятельности;

— а также для проектирования, программирования, моделирования и обучения [4].

Правила языка ДРАКОН по созданию дракон-схем (drakon-charts) оптимизированы для восприятия и понимания алгоритмов человеком. Таким образом, язык предлагается в качестве инструмента усиления человеческого интеллекта.

1. Применение языка ДРАКОН в ракетно-космической технике

Язык ДРАКОН (и основанная на нем технология разработки алгоритмов и программ ГРАФИТ-ФЛОКС [5]) эксплуатируются в НПЦ АП уже более 16 лет (1996–2013). Они используются в следующих космических программах:

— разгонный блок космических аппаратов ДМ-SL (международный проект «Морской старт»);

— модернизированная ракета-носитель тяжелого класса «Протон-М»;

— разгонный блок космических аппаратов «Фрегат»;

— разгонный блок космических аппаратов «Фрегат-СБ» (со сбрасываемыми баками);

— разгонный блок космических аппаратов «Фрегат УТТХ» (с улучшенными тактико-техническими характеристиками);

— разгонный блок космических аппаратов «Фрегат-МТ» (предназначен для запусков с космодрома «Kuru» в Южной Америке);

— разгонный блок космических аппаратов «Фрегат» СУМ (с модернизированной системой управления);

— разгонный блок космических аппаратов ДМ-SLБ (проект «Наземный старт»);

— разгонный блок космических аппаратов ДМ-03;

— первая ступень южнокорейской ракеты-носителя легкого класса KSLV-1 (Korean Space Launch Vehicle #1);

— ракета-носитель легкого класса Ангара 1.2 первого пуска;

— ракета-носитель легкого класса Ангара 1.2 с агрегатным модулем;

— ракета-носитель тяжелого класса Ангара-А5;

— разгонный блок космических аппаратов КВТК (кислородно-водородный тяжелого класса) *(планируемая работа)* [6, с. 515, 7].

Пуски ракет-носителей и разгонных блоков космических аппаратов, при создании которых использовался язык ДРАКОН, за истекший период (1999–2013 годы) производились с пяти космодромов мира, расположенных на трех континентах (Европа, Азия, Америка) и в океане:

— международный плавучий космодром, производящий пуски с экватора в Тихом океане (экваториальная зона вблизи острова Рождества Республики Кирибати с координатами 154 градуса западной долготы и 0 градусов широты);

— Байконур (Казахстан, Азия);

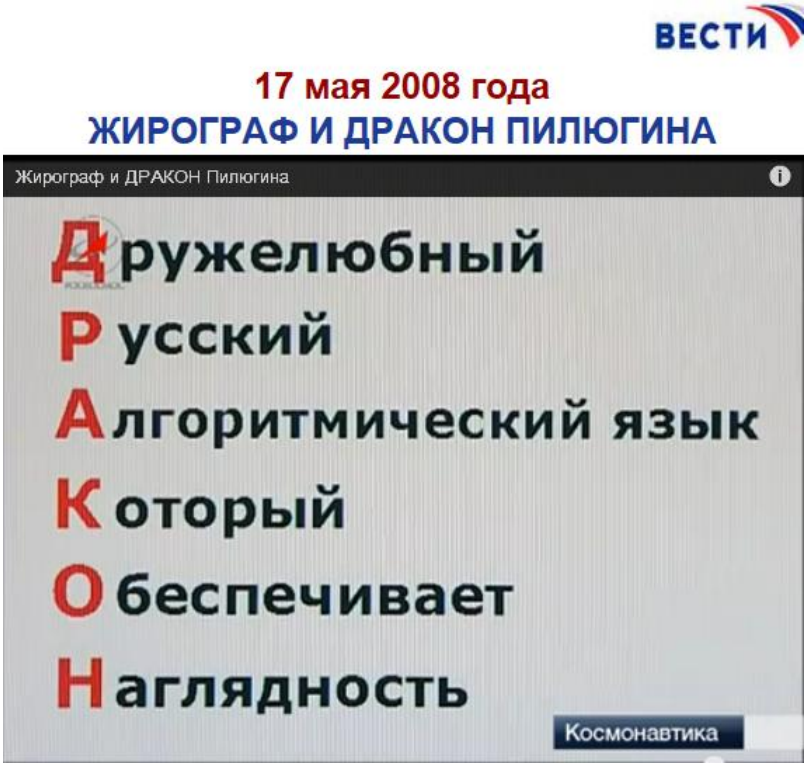
— Плесецк (Россия, Европа);

— Европейский космический центр во Французской Гвиане «Kuru» (Южная Америка);

— Южнокорейский космодром «Naro» (Азия).

Поскольку результаты использования ДРАКОНа были стабильно высокими, руководство НПЦ АП приняло решение об использовании ДРАКОН-технологии во всех последующих проектах [8].

Скриншот 1. Германия, 1945 год. Майор Черток и полковник Пилюгин прибыли в командировку в побежденную Германию, чтобы изучить секретное оружие вермахта — баллистическую ракету ФАУ-2, созданную Вернером фон Брауном. *Кадр из документального видеофильма «Жирограф и ДРАКОН Пилюгина»* [3]



Скриншот 2. Кадр из документального видеофильма «Жирограф и ДРАКОН Пилюгина». Автор – Н. Бурцева Оператор – Е. Петров. Телерадиостудия Роскосмоса [3].

2. Графический алфавит языка ДРАКОН. Иконы и макроиконы

2.1. Преимущества дракон-схем

Чем отличаются дракон-схемы от блок-схем алгоритмов и программ по ГОСТ 19.701–90?

Хотя блок-схемы порою действительно улучшают понятность алгоритмов, однако это справедливо только для простых задач. С ростом сложности алгоритмов блок-схемы стремительно теряют наглядность, становятся запутанными и трудными для понимания.

В отличие от них дракон-схемы отличаются безупречной ясностью, прозрачностью, четкостью, которая не зависит от сложности. Более того, чем сложнее алгоритмы, тем больше выигрыш от языка ДРАКОН. Он обеспечивают быстроту и легкость понимания по принципу: «Посмотрел – и сразу понял!».

Благодаря использованию формальных и неформальных приемов дракон-схемы дают возможность изобразить любой, сколь угодно сложный алгоритм (и любую сложную алгоритмическую систему, состоящую из вложенных друг в друга и параллельных алгоритмов) в наглядной и доходчивой форме.

Это позволяет значительно сократить интеллектуальные усилия персонала, необходимые для разработки и проверки алгоритмов и иерархических алгоритмических систем, формализуемых методом декомпозиции.

Блок-схемы не обеспечивают автоматическое преобразование алгоритма в машинный код. Дракон-схемы, напротив, пригодны для автоматического получения исполняемого кода.

Главное преимущество состоит в том, что дракон-схемы (в отличие от блок-схем) формируются методом логического вывода с помощью разработанного мною визуального логического исчисления, которое я назвал «исчислением икон» [6, с. 427-435]. Дракон-конструктор осуществляет 100%-е автоматическое доказательство правильности дракон-схем, гарантируя принципиальную невозможность ошибок визуального синтаксиса. Иными словами, дракон-конструктор полностью исключает ошибки графического синтаксиса [6, с. 11, 393-424].

Я предполагаю, что язык ДРАКОН снимет с повестки дня проблему сложности алгоритмов (при условии, что программа «дракон-конструктор» спроектирована правильно) [6, с. 393-424]. В данном случае сложность трактуется как трудность понимания алгоритмов человеческим мозгом с помощью человеческого зрительного восприятия и характеризуется (согласно ГОСТ 28806-90 (пункт 3.1 Приложения 2) как затраты усилий человека на понимание логической концепции алгоритма).

2.2. Иконы

Чтобы графический язык стал гибким и выразительным, нужно тщательно спроектировать графический алфавит языка, сделать его эргономичным. Графоэлементы (графические буквы) языка ДРАКОН называются *иконами* (рис. 1). В языке ДРАКОН 26 икон.

Столь богатый алфавит обладает большой выразительной силой. Он позволяет изобразить алгоритмы и иерархические алгоритмические системы любой сложности, включая параллельные процессы и процессы реального времени. И обеспечить максимальную наглядность и понятность полученной «картинки», т.е. математически строгого комплекта эргономичных алгоритмических чертежей.

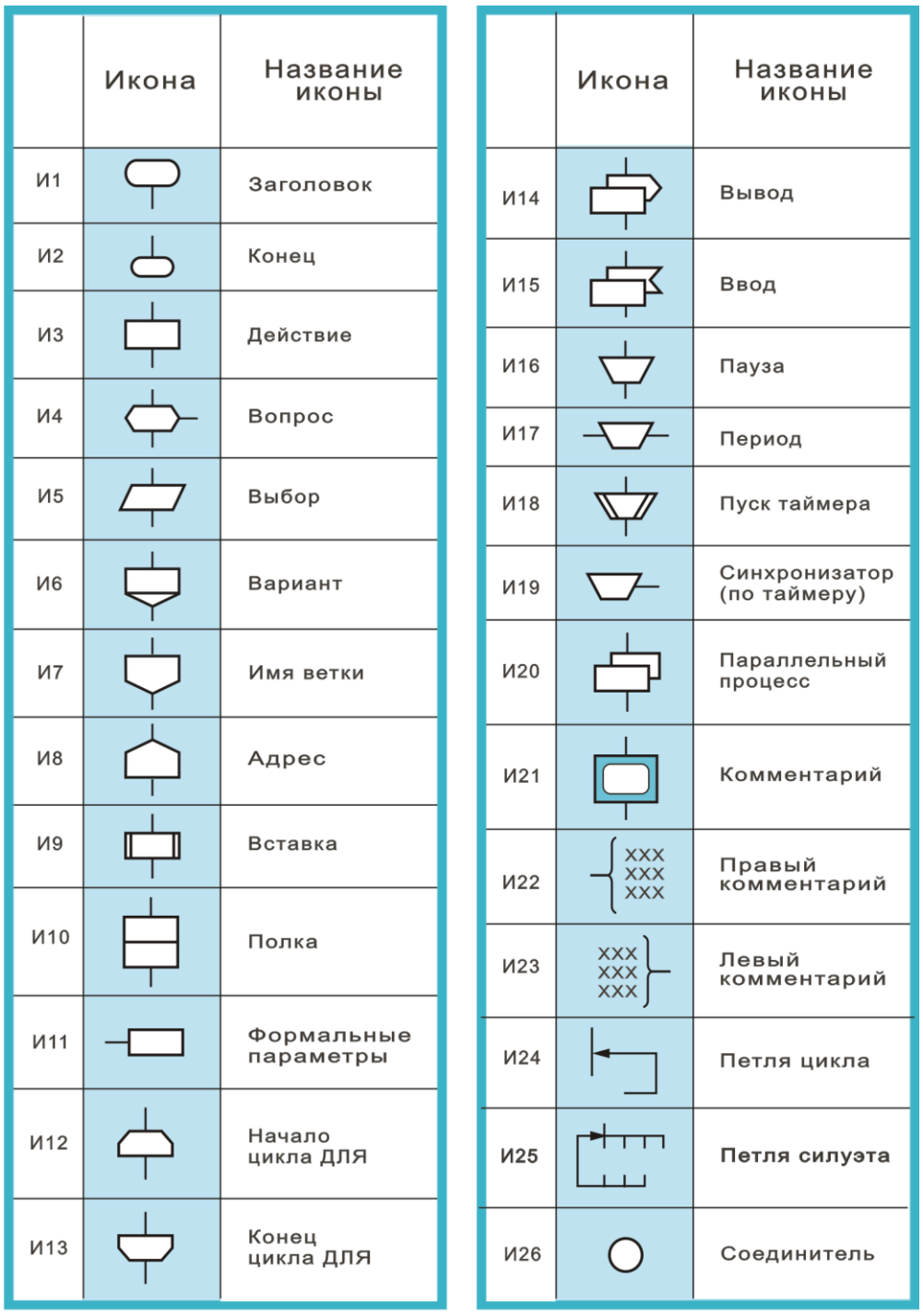


Рис. 1. Иконы языка ДРАКОН

2.3. Макроиконы

Подобно тому, как буквы объединяются в слова, иконы объединяются в составные иконы – *макроиконы* (рис. 2). Макроиконы – это составные алгоритмические структуры.

Соединяя иконы и макроиконы по определенным правилам, можно строить разнообразные алгоритмы и иерархические алгоритмические системы. Общее число икон и макроикон равно 46.

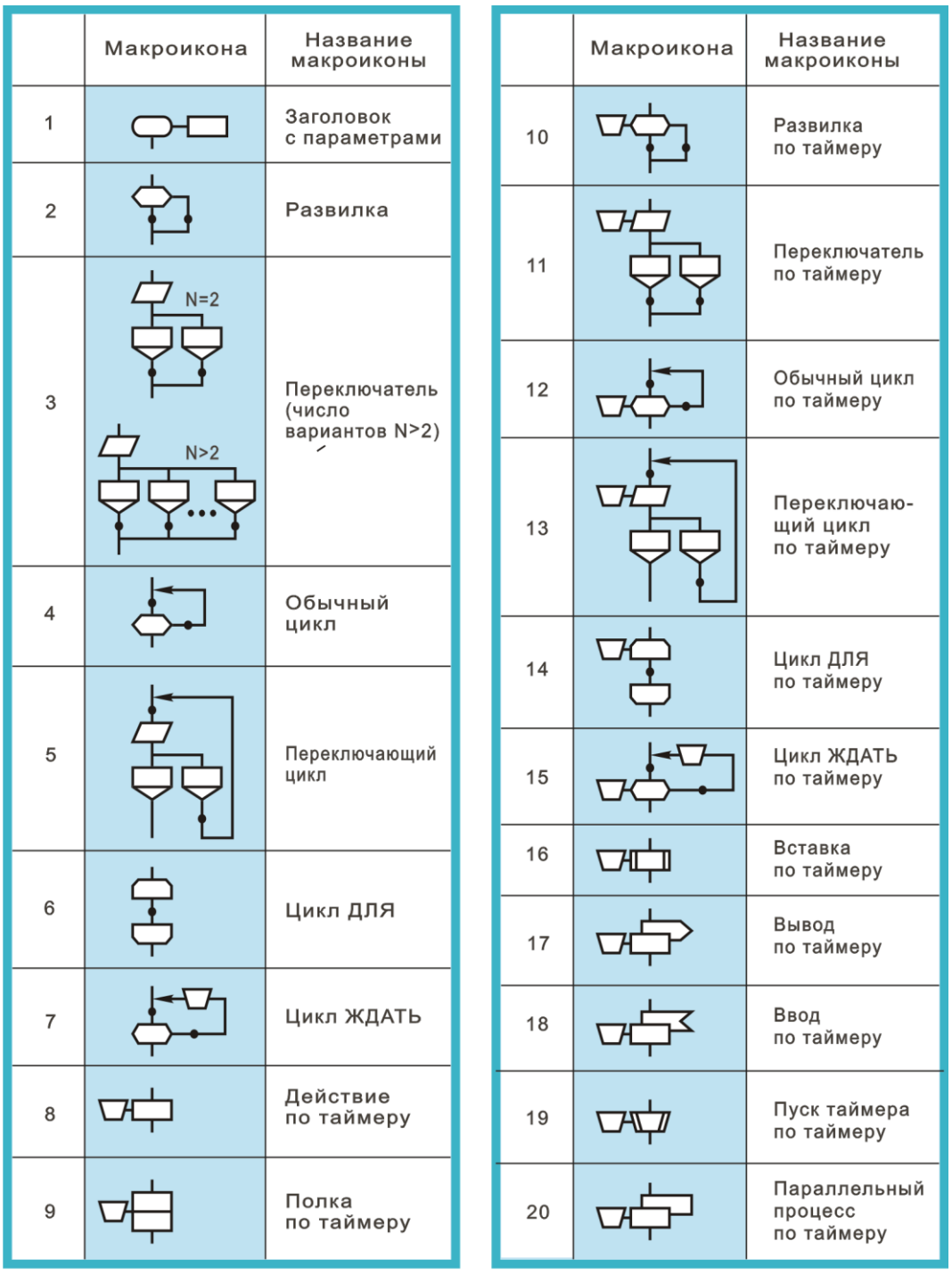


Рис. 2. Макроиконы языка ДРАКОН

3. Алгоритмы реального времени

Упрощенный алгоритм управления светофором показан на рис. 3.

Шапка дракон-схемы представлена на рис. 4. Шапка дает ответ на три наиболее важных вопроса:

1. Как называется задача?
2. Из скольких частей она состоит?
3. Как называется каждая часть?

Вот ответы для рис. 4: (1) Управление светофором. (2) Из трех. (3) Ответ записан в иконах «имя ветки»:

— Управление зеленым светом.

— Управление красным светом.

— Ночной режим.

В иконах «адрес» разрешается писать только те имена, которые указаны в иконах «имя ветки». Икона адрес — это замаскированный оператор перехода (goto). Однако он передает управление не куда угодно, а только на начало выбранной ветки. Черные треугольники обозначают так называемый «веточный цикл».

На рис. 3 первой начинает работу крайняя левая ветка. Сначала выполняется команда ВКЛЮЧИТЬ ЗЕЛЕНЫЙ. Затем команда ПАУЗА, которая отсчитывает время 2 минуты, после чего выполняется команда ВЫКЛЮЧИТЬ ЗЕЛЕНЫЙ и т. д.

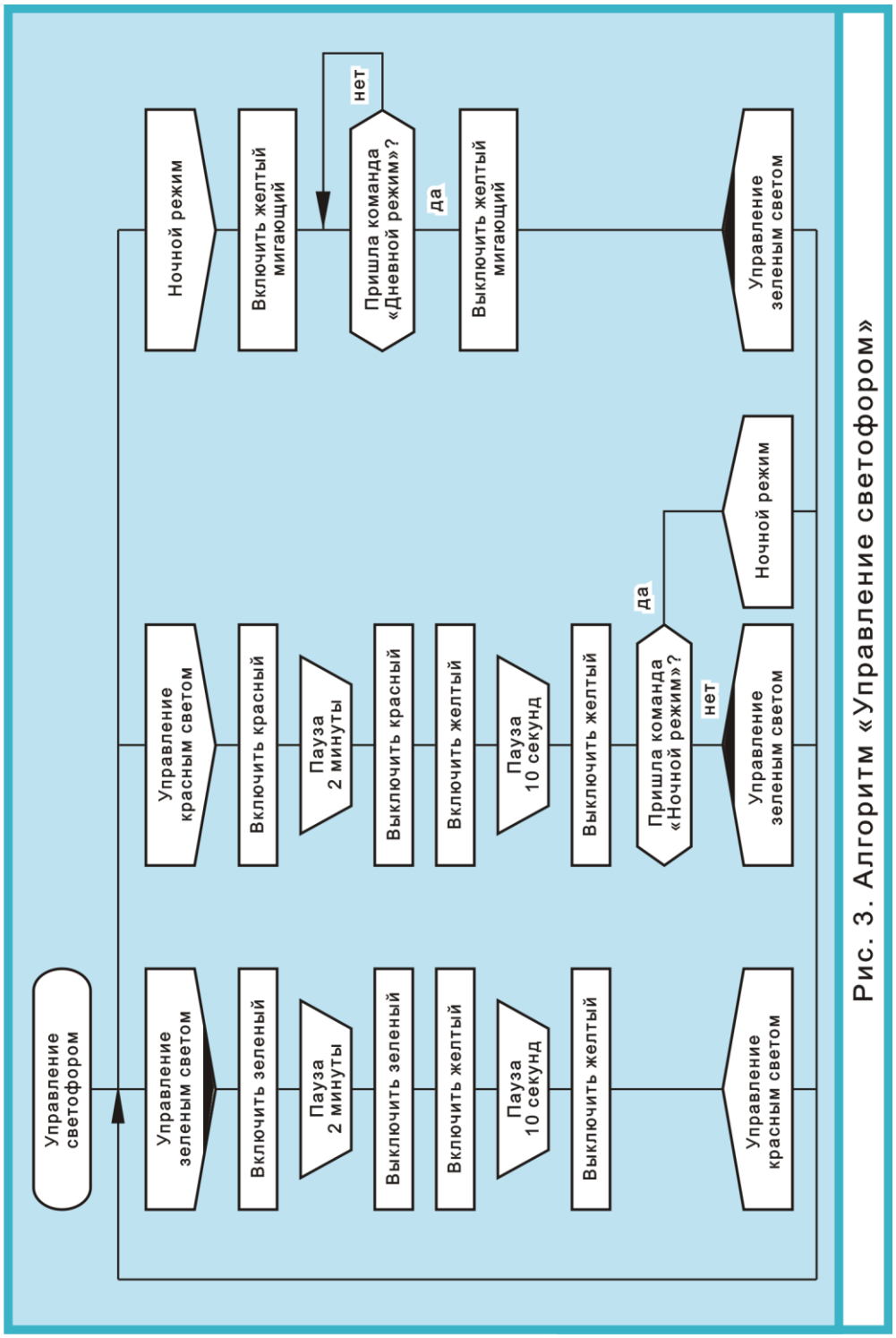


Рис. 3. Алгоритм «Управление светофором»

Рис. 4. Шапка алгоритма «Управление светофором». Показаны икона «заголовок» и три иконы «имя ветки»

3.1. Список операторов реального времени

В языке ДРАКОН имеется пять икон реального времени (рис. 1, иконы И16–И20):
— пауза;
— период;
— пуск таймера;
— синхронизатор (по таймеру);
— параллельный процесс.

Три из них (пауза, пуск таймера и параллельный процесс) – простые операторы. Две другие иконы (период и синхронизатор) служат «кирпичиками» для построения составных операторов и вне последних не используются.

Икона «период» является принадлежностью цикла ЖДАТЬ (рис. 2, макроикона 7). Икона «синхронизатор» служит для образования тринадцати составных операторов (рис. 2, макроиконы 8–20).

3.2. Операторы ввода-вывода

В языке ДРАКОН предусмотрены два визуальных оператора ввода-вывода: «вывод» (рис. 1, икона И14) и «ввод» (рис. 1, икона И15). Оба оператора «двухэтажные». На верхнем этаже пишется ключевое слово или ключевая фраза. На нижнем (в прямоугольнике) – содержательная информация, подлежащая вводу и выводу (рис. 5, 6).

На рис. 1 видно, что иконы ввода-вывода имеют мнемоническую форму. Икона И14 содержит полую стрелку, направленную наружу, что символизирует «вывод», а икона И15 – стрелку, направленную внутрь (ввод).

3.3. Оператор «пауза»

Предположим, управляющий компьютер должен выдать серию электрических команд, которые по линиям связи передаются в исполнительные органы и вызывают срабатывание электромеханических реле. В результате происходит открытие трубопровода, включение насоса и другие операции, необходимые для функционирования управляемого объекта.

Такая ситуация может встретиться во многих системах управления реального времени. Например, при заправке топливом космических ракет, на атомных электростанциях, нефтеперерабатывающих заводах и т. д.

Рассмотрим пример. Предположим, управляющий компьютер должен:
• выдать команду ОТКРЫТЬ.ТРУБОПРОВОД;
• подождать две минуты;

• выдать две команды: ВКЛЮЧИТЬ.НАСОС и ОТКРЫТЬ.ЗАСЛОНКУ;
• подождать 45 секунд;
• выдать команду ПОДАЧА.ТОПЛИВА;
• подождать три минуты;
• выдать команду ПУСК.АГРЕГАТА.

Соответствующий алгоритм представлен на рис. 5. Задержка выдачи команд реализуется с помощью иконы «пауза». Внутри последней указывается время необходимой задержки. Например, 2мин (2 минуты), 45с (45 секунд) и т. д.

Верхний оператор «пауза» на рис. 5 работает так. После выдачи команды ОТКРЫТЬ.ТРУБОПРОВОД в управляющем компьютере запускается программный счетчик времени на 2 минуты. По истечении этого времени компьютер выдает в линию связи команды ВКЛЮЧИТЬ.НАСОС и ОТКРЫТЬ.ЗАСЛОНКУ.

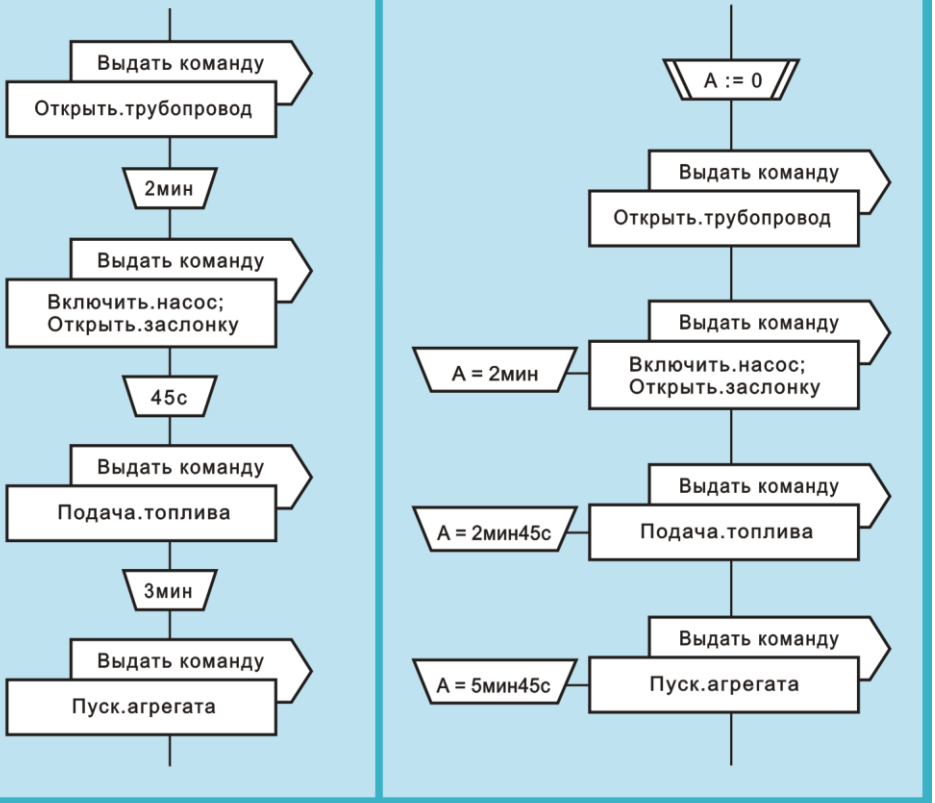


Рис. 5. Пример использования оператора «пауза»

Рис. 6. Пример использования операторов «пуск таймера» и «синхронизатор»

3.4. Операторы «пуск таймера» и «синхронизатор»

Изменим задачу. Будем считать, что разработчик управляемого объекта хочет указать время выдачи команд не по принципу «задержка после предыдущей команды», а по принципу секундомера. Это значит, что все времена отсчитываются от единого начального момента (совпадающего с пуском секундомера).

Исходя из этого, сформулируем задачу управляющего компьютера. Он должен:
• включить «секундомер», то есть обнулить и запустить таймер;
• выдать команду ОТКРЫТЬ.ТРУБОПРОВОД;
• когда таймер отсчитает две минуты, выдать пару команд ВКЛЮЧИТЬ.НАСОС и ОТКРЫТЬ.ЗАСЛОНКУ;
• когда таймер отсчитает 2 минуты 45 секунд, выдать команду ПОДАЧА.ТОПЛИВА;
• когда таймер отсчитает 5 минут 45 секунд, выдать команду ПУСК.АГРЕГАТА.

Описанный алгоритм изображен на рис. 6. В нем используются операторы «пуск таймера» и «синхронизатор», совместная работа которых обеспечивает нужный эффект.

Оператор «пуск таймера» порождает, обнуляет и запускает таймер и присваивает ему имя *A*. Оператор «синхронизатор» задерживает выполнение размещенного справа от него визуального оператора до наступления момента, указанного в иконе «синхронизатор».

Например, синхронизатор *A* = 2мин45с на рис. 6 задерживает выдачу команды ПОДАЧА.ТОПЛИВА до момента, когда таймер *A* отсчитает 2 минуты 45 секунд.

Сравнивая алгоритмы на рис. 5 и 6, можно заметить, что они почти эквивалентны. Почему почти?

Чтобы разобраться, рассмотрим идеальный случай. Предположим, что время, необходимое для выдачи одной команды равно нулю. Имеется в виду, что перечисленные ниже команды выдаются за время, равное нулю:

• ОТКРЫТЬ.ТРУБОПРОВОД;
• ВКЛЮЧИТЬ.НАСОС;
• ОТКРЫТЬ.ЗАСЛОНКУ;
• ПОДАЧА.ТОПЛИВА;
• ПУСК.АГРЕГАТА.

В этом случае оба алгоритма будут выдавать команды синхронно.

Однако в действительности идеальные случаи встречаются далеко не всегда. Иногда бывает, что время выдачи одной команды больше нуля.

В таком случае алгоритмы работают по-разному. Практика показывает, что в некоторых ситуациях предпочтительным является *принцип паузы* (когда используется икона «пауза»). А в других – *принцип таймера* (когда используются иконы «пуск таймера» и «синхронизатор»). Оба инструмента оказываются в равной степени необходимыми и полезными.

3.5. Пример алгоритма реального времени

На рис. 7 представлен более сложный алгоритм, в котором применяются операторы «пауза», «пуск таймера» и «синхронизатор».

В средней ветке изображена икона «пауза» с надписью 2мин48с. Это означает, что после завершения процедуры ВОЛШЕБНЫЙ РЕМОНТ ТАРЕЛКИ отсчитывается пауза длительностью 2 минуты 48 секунд. И только после этого производится снятие признака АВАРИЯ ТАРЕЛКИ.

Еще одна 4-секундная пауза предусмотрена в левой ветке.

В правой ветке есть икона «пуск таймера» с надписью *A* = 0. Данный оператор порождает, обнуляет и запускает таймер *A*.

В той же ветке установлены три иконы «синхронизатор по таймеру» с надписями *A* = 3мин, *A* = 5мин и *A* = 8мин. При этом вызов процедуры ВКЛЮЧИТЬ ТЕЛЕПОРТАЦИЮ произойдет не сразу, а только после того, как таймер *A* отсчитает 3 минуты. Соответственно включение в работу процедур ОТКЛЮЧИТЬ ГРАВИТАЦИЮ и ВЫХОД ИЗ АСТРАЛЬНОГО ТЕЛА будет задержано до тех пор, пока таймер *A* не примет значения 5 и 8 минут соответственно.

На рис. 7 видно, что оператор «пуск таймера» можно применять двумя способами:

• во-первых, совместно с иконой «синхронизатор» (этот случай мы обсудили);
• во-вторых, совместно с иконой «вопрос».

Последний случай рассмотрен в следующем параграфе.

3.6. Цикл ЖДАТЬ

Предположим, нужно в течение 3 минут ждать появления хотя бы одного из двух признаков ЛЕВЫЙ ДВИГАТЕЛЬ В НОРМЕ и ПРАВЫЙ ДВИГАТЕЛЬ В НОРМЕ. При

наступлении этого события (появлении одного из признаков) необходимо включить плазменный реактор. Если же названные признаки отсутствуют, по истечении трех минут следует включить фотонный двигатель.

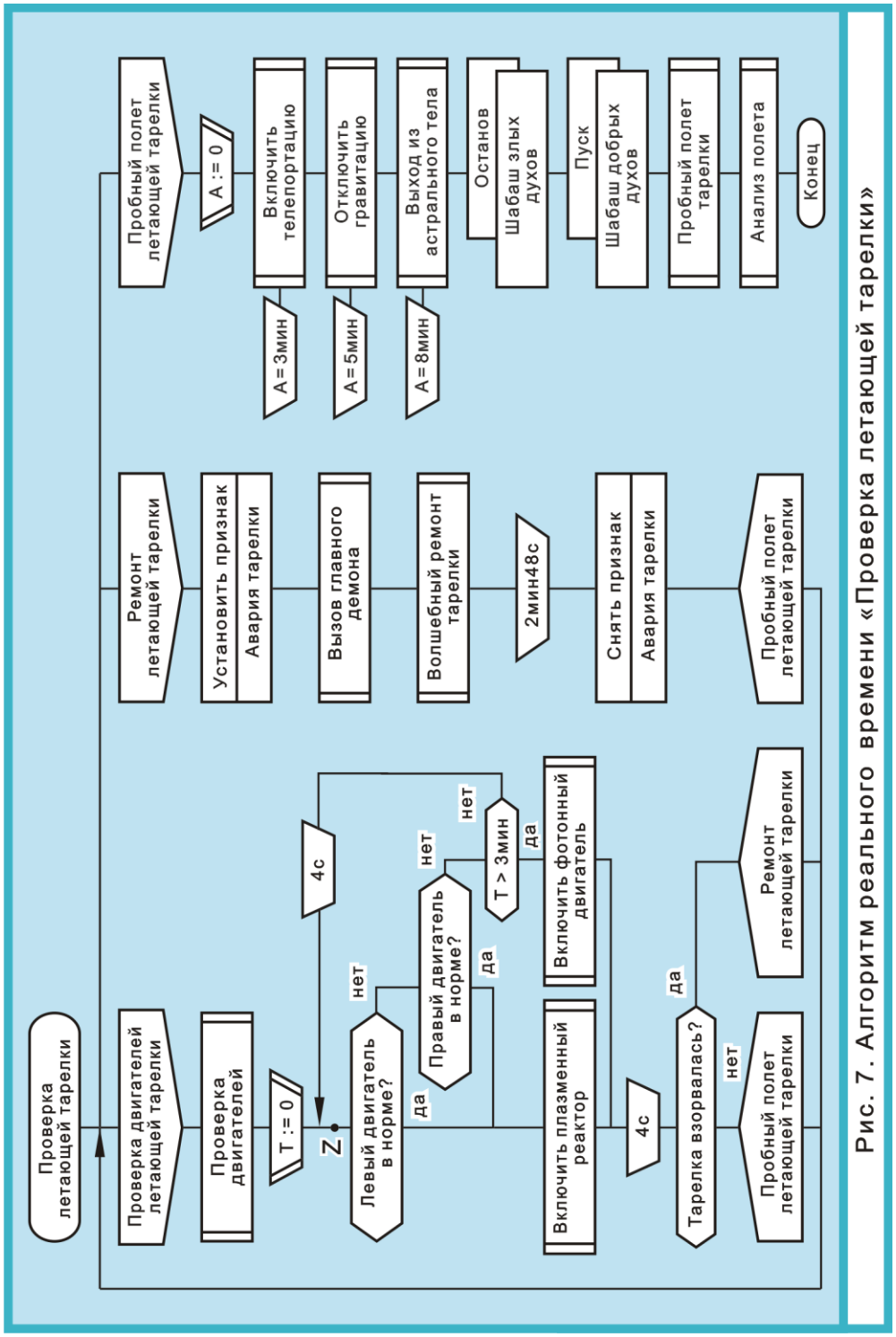


Рис. 7. Алгоритм реального времени «Проверка летающей тарелки»

Для решения задачи на рис. 7 используются два оператора:
• пуск таймера *T*, отсчитывающего три минуты;
• цикл ЖДАТЬ.
В состав последнего входит икона «период» и три иконы «вопрос».
В последних размещены надписи:
• ЛЕВЫЙ ДВИГАТЕЛЬ В НОРМЕ?
• ПРАВЫЙ ДВИГАТЕЛЬ В НОРМЕ?

• $T > 3$мин.

Последний оператор проверяет: значение таймера *T* больше трех минут?

Если оба признака отсутствуют, а значение таймера не превышает 3 минут, опрос условий периодически повторяется. При этом период опроса указывается в иконе «период». В данном примере он равен 4 секундам.

На рис. 7 видно,что цикл ЖДАТЬ закончится в момент обнаружения одного из ожидаемых признаков, а если они не появятся, – через 3 минуты.

• Задача ожидания нескольких признаков (когда система должна по-разному реагировать на каждый признак) является типичной при разработке систем реального времени.

• Цикл ЖДАТЬ – удобное средство для решения подобных задач.

3.7. Оператор «период»

Сравнивая макроиконы 4 и 7 на рис. 2 (обычный цикл и цикл ЖДАТЬ), мы видим, что они очень похожи. Поэтому во избежание путаницы нужно иметь различительный признак. Эту функцию выполняет икона «период». Если она есть в петле цикла – перед нами цикл ЖДАТЬ. Если нет – обычный цикл.

3.8. Оператор «параллельный процесс»

Пусть заданы два алгоритма *A* и *B*, причем *A* – основной алгоритм, а *B* – вспомогательный. Алгоритмы *A* и *B* могут работать последовательно (рис. 8) или параллельно (рис. 9).

Чтобы организовать *последовательную* работу, необходимо в дракон-схеме основного алгоритма *A* нарисовать икону-вставку с надписью *B*. В этом случае алгоритм В называется *процедурой.*

Например, на рис. 7 в основном алгоритме ПРОВЕРКА ЛЕТАЮЩЕЙ ТАРЕЛКИ имеется процедура ПРОВЕРКА ДВИГАТЕЛЕЙ. Эти алгоритмы действуют последовательно. Основной алгоритм передает управление процедуре ПРОВЕРКА ДВИГАТЕЛЕЙ и прекращает работу.

Возобновление работы алгоритма ПРОВЕРКА ЛЕТАЮЩЕЙ ТАРЕЛКИ произойдет только тогда, когда процедура ПРОВЕРКА ДВИГАТЕЛЕЙ закончится. В общем виде ситуация показана на рис. 8.

Отличие параллельного режима состоит в том, что после начала вспомогательного алгоритма *B* основной алгоритм *A* не прекращает работу и действует одновременно с алгоритмом *B* (рис. 9).

Как работает оператор «период»? Поясним на примере. На рис. 7 цикл ЖДАТЬ «крутится» по своей петле с периодичностью 4 секунды, пока не выполнится одно из трех условий. После чего произойдет выход из цикла. Таким образом, оператор «период» задает период повторения цикла ЖДАТЬ.

Чтобы организовать параллельную работу, нужно в дракон-схеме основного алгоритма *A* нарисовать икону «параллельный процесс» (рис. 1, икона И20).

Икона «параллельный процесс» двухэтажная. На верхнем этаже пишут ключевое слово, обозначающее команду, изменяющую состояние параллельного процесса, например, «Пуск», «Останов» и т. д. На нижнем этаже помещают идентификатор параллельного процесса.

Обратимся к примеру на рис. 7. В правой ветке находятся два оператора управления параллельными процессами. После окончания процедуры ВЫХОД ИЗ АСТРАЛЬНОГО ТЕЛА производится останов параллельного процесса ШАБАШ ЗЛЫХ ДУХОВ и пуск процесса ШАБАШ ДОБРЫХ ДУХОВ.

При этом предполагается, что до начала алгоритма ПРОВЕРКА ЛЕТАЮЩЕЙ ТАРЕЛКИ некий третий алгоритм выдал команду «Пуск» и запустил параллельный процесс

ШАБАШ ЗЛЫХ ДУХОВ. Последний работает одновременно с алгоритмом ПРОВЕРКА ЛЕТАЮЩЕЙ ТАРЕЛКИ вплоть до момента выдачи команды «Останов» (см. последнюю ветку на рис. 7).

Указанная команда ликвидирует параллельный процесс ШАБАШ ЗЛЫХ ДУХОВ. В этот момент одновременная работа заканчивается.

Однако следующая команда «Пуск» запускает другой параллельный процесс – ШАБАШ ДОБРЫХ ДУХОВ, который начинает работать одновременно с алгоритмом ПРОВЕРКА ЛЕТАЮЩЕЙ ТАРЕЛКИ.

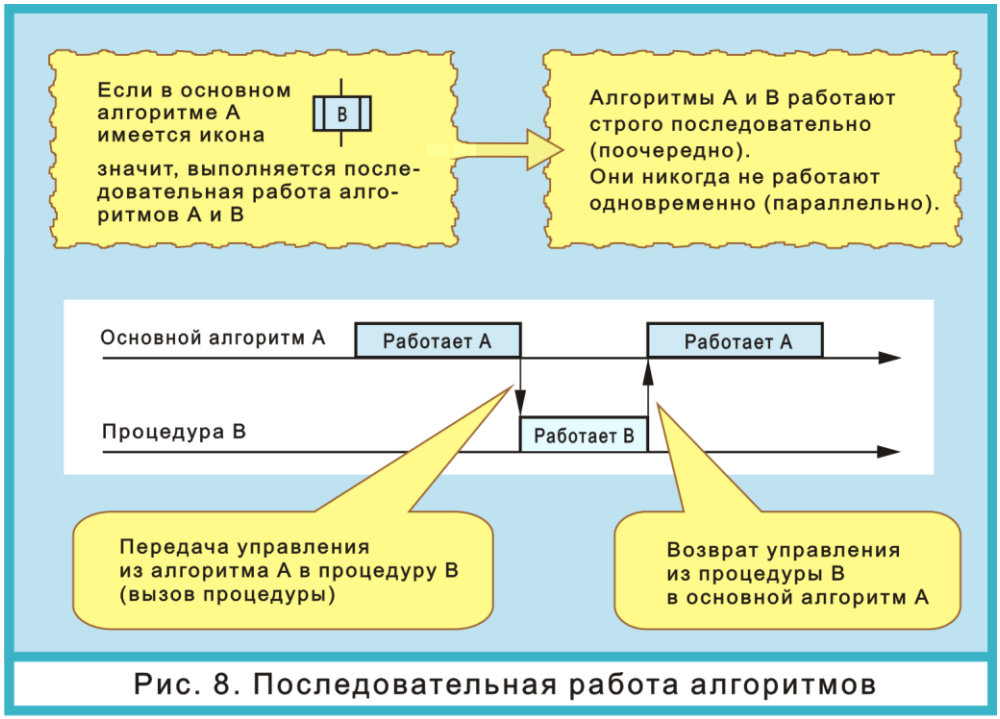


Рис. 8. Последовательная работа алгоритмов

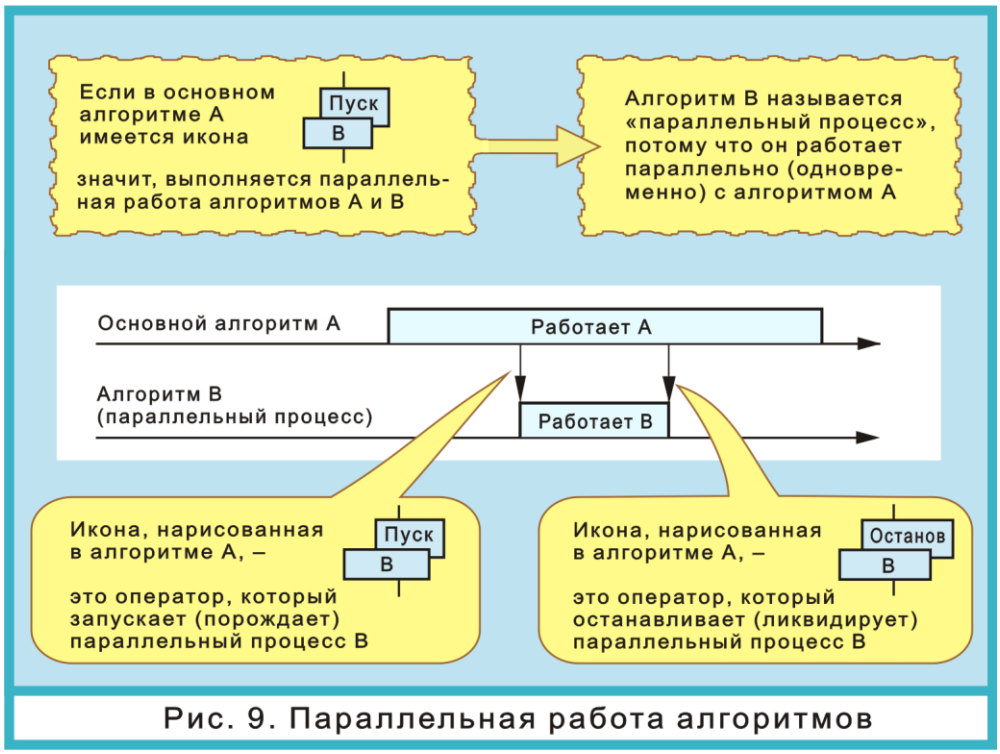


Рис. 9. Параллельная работа алгоритмов

3.9. Особенности операторов реального времени

Операторы реального времени – это формальные операторы языка ДРАКОН. Однако их можно использовать и при неформальном изображении алгоритмов. Например, для построения наглядных «картинок», позволяющих легко объяснить ту или иную идею, относящуюся к системам реального времени.

Примеры таких картинок представлены на рис. 3 и 10. При этом в цикле ЖДАТЬ икону «период» обычно опускают, чтобы не загромождать рисунок (см. последнюю ветку на рис. 3). Однако если длительность периода нужна для понимания, икону «период» можно сохранить (рис. 10).

3.10. Взаимодействие прикладной программы с операционной системой реального времени

На рис. 3, 5-7 показаны операторы реального времени: «пауза», «пуск таймера», «синхронизатор», «период», а также цикл ЖДАТЬ. Эти операторы нарисованы внутри алгоритмов. Поэтому может создаться впечатление, что они реализуются этими алгоритмами (то есть прикладными программами реального времени). Но это не так.

На самом деле перечисленные операторы реализуются совместно:

• прикладной программой реального времени;

• дракон-диспетчером, входящим в состав операционной системы реального времени.

Когда в прикладной программе встречается оператор «пауза», происходят события, не показанные на дракон-схемах. А именно, выход из прикладной программы и передача управления в дракон-диспетчер (с одновременной передачей параметра, записанного в иконе «пауза»).

Получив параметр, диспетчер отсчитывает время, указанное в паузе. Когда время истечет, диспетчер возвращает управление в прикладную программу – в точку, расположенную после иконы «пауза».

Иными словами, всякий раз, когда на рисунке алгоритма изображена пауза происходят два события:

• выход из прикладной программы (в начале паузы);

• вход в прикладную программу (в конце паузы).

Рассмотрим еще один пример – оператор «период». Длительность периода отсчитывает не прикладная программа на рис. 7, а дракон-диспетчер, входящий в состав операционной системы реального времени.

Оператор «период» означает выход из прикладной программы.

Управление переходит к дракон-диспетчеру (с одновременной передачей параметра 4с). Через каждые 4 секунды дракон-диспетчер передает управление в начало цикла ЖДАТЬ (точка *Z* на рис. 7). Если все три условия дают ответ «нет», оператор «период» возвращает управление в дракон-диспетчер. Таким образом, функционирование цикла ЖДАТЬ обеспечивается совместными усилиями прикладной программы и дракон-диспетчера.

Этот вывод относится ко всем операторам реального времени.

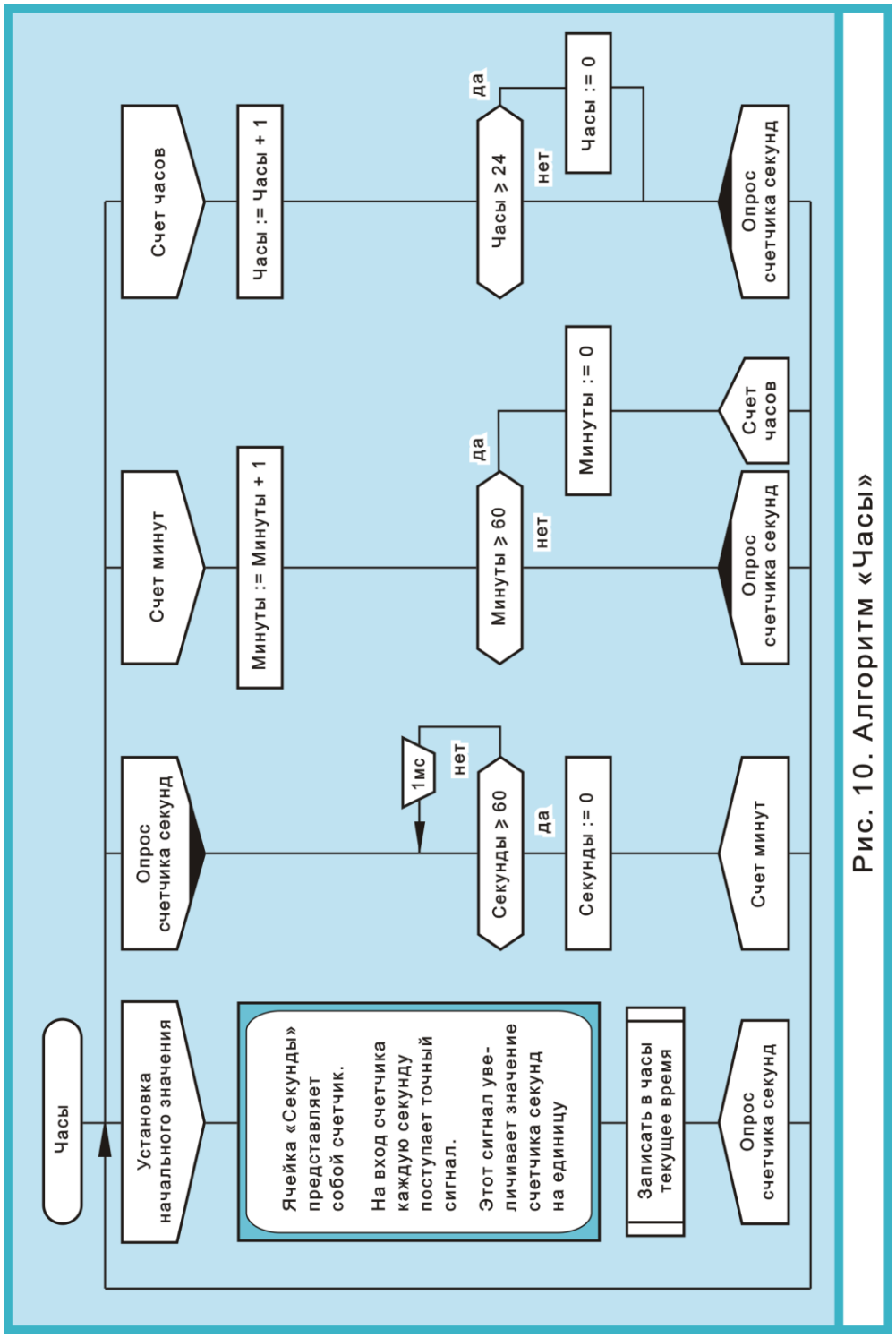
На дракон-схемах показаны алгоритмы, которые имеют одно начало (один вход) и один конец (один выход). В действительности программы реального времени имеют много входов и много выходов.

Дополнительные входы и выходы появляются всякий раз, когда в алгоритм добавляется оператор пауза или период. Но эти дополнительные входы и выходы на рисунках не показаны. Они не показаны из эргономических соображений – чтобы не загромождать схему.

3.11. Выразительные возможности

Наличие визуальных операторов реального времени резко расширяет изобразительные возможности языка ДРАКОН и позволяет использовать его при проектировании и

разработке не только информационных, но и управляющих систем. Это обстоятельство существенно увеличивает область применения языка.



Дополнительным преимуществом является лаконичность выразительных средств, их универсальность. В языке всего пять икон реального времени, однако их алгоритмическая мощь – в сочетании с другими возможностями языка – позволяет охватить обширный спектр задач, связанных с созданием алгоритмов и программ для управляющих систем.

Важную роль играет эргономичность операторов реального времени. Как и другие операторы языка ДРАКОН, они имеют визуальный характер. Это позволяет сделать операции реального времени более наглядными и легкими для понимания по сравнению с традиционной текстовой записью.

Четыре иконы (пауза, период, пуск таймера и синхронизатор) – «близкие родственники» в том смысле, что внутри каждой из них указывается значение времени. Эта родственная связь находит свое эргономическое отражение в том, что перечисленные операторы имеют визуальное «фамильное сходство». Все они построены (с вариациями) на основе одной и той же геометрической фигуры – перевернутой равнобедренной трапеции.

4. Параллельные алгоритмы

4.1. Первый пример. Алгоритм «Проверка агрегата и ракеты»

На рис. 11 изображены 15 параллельных процессов:

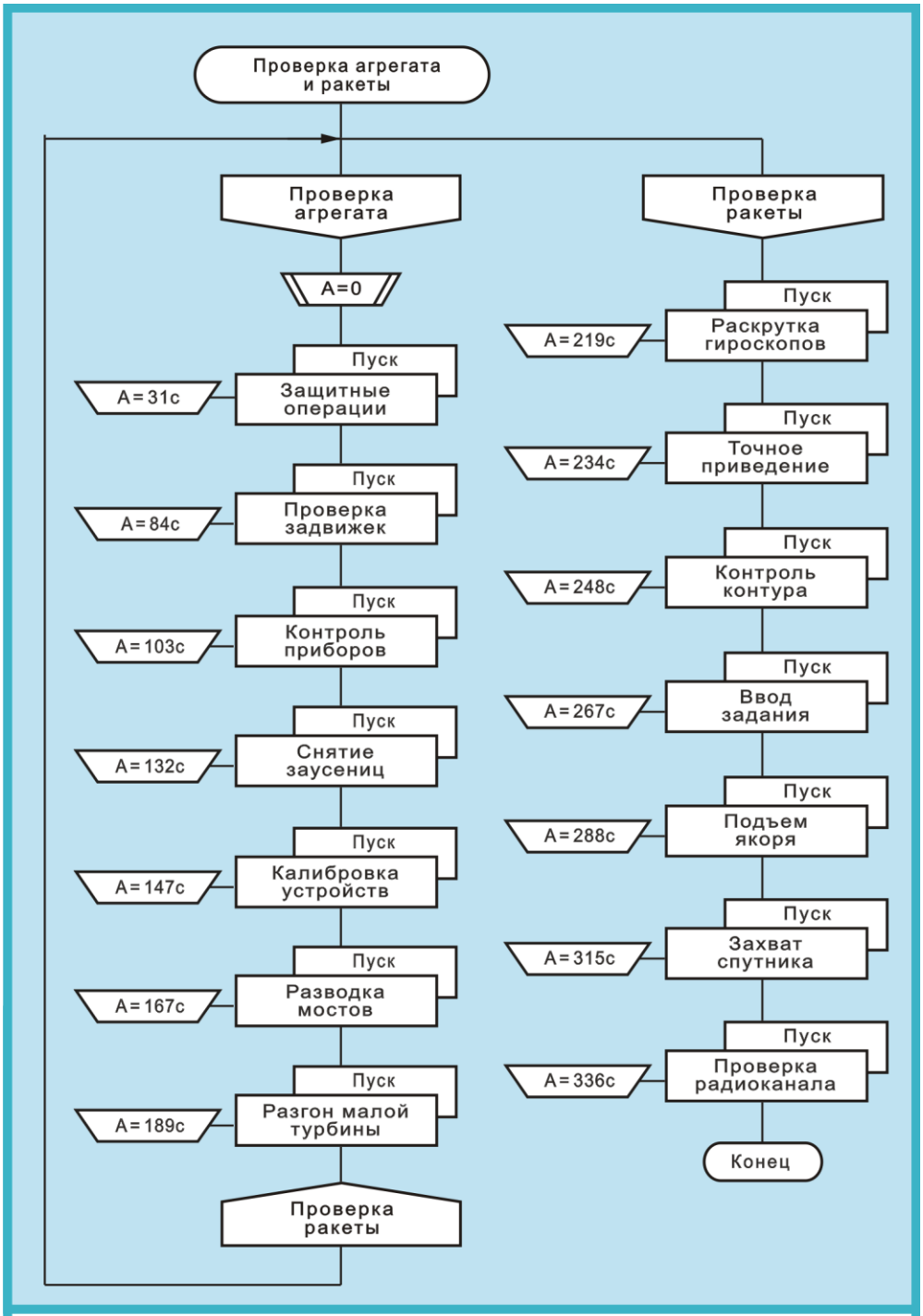


Рис. 11. Алгоритм «Проверка агрегата и ракеты»

• вызывающий алгоритм ПРОВЕРКА АГРЕГАТА И РАКЕТЫ;

• 14 вызываемых алгоритмов, каждый из которых обозначен иконой «параллельный процесс» (7 алгоритмов в первой ветке и 7 – во второй).

Все вызываемые процессы запускаются сигналом ПУСК. Момент запуска точно определен оператором синхронизатор. Например, процесс КОНТРОЛЬ ПРИБОРОВ запускается в момент 103с. Параллельные процессы могут заканчиваться двумя способами:

• по команде «Останов» (см. пример на рис. 7, правая ветка);

• без использования команды «Останов», то есть естественным путем, когда каждый процесс решит свою задачу и достигнет конца.

На рис. 11 показан случай, когда все вызываемые процессы заканчиваются естественным путем. Поэтому команда ОСТАНОВ не используется.

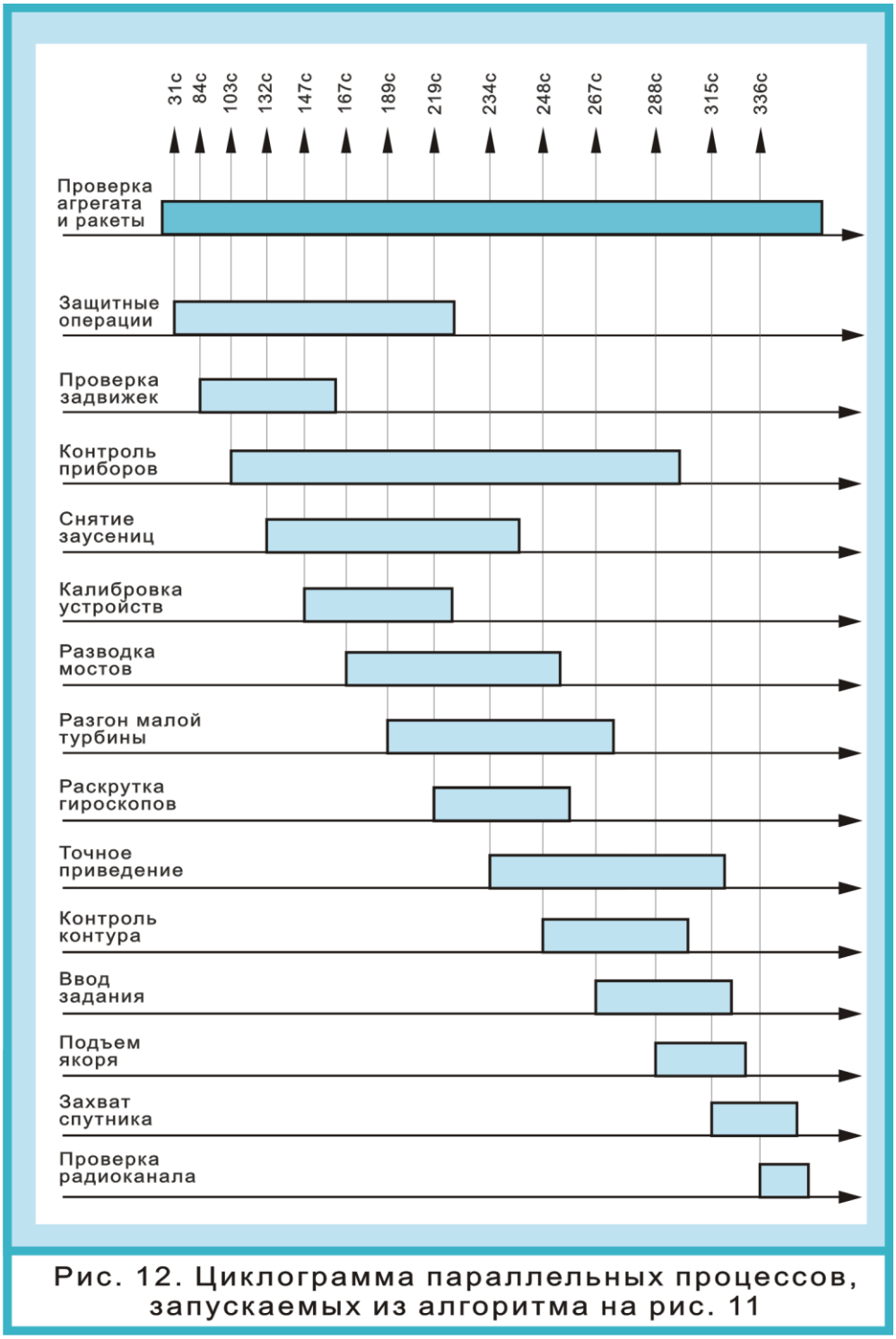


Рис. 12. Циклограмма параллельных процессов, запускаемых из алгоритма на рис. 11

На рис. 12 показана временная диаграмма, иллюстрирующая алгоритм на рис. 11. В верхней строке темным цветом выделен вызывающий алгоритм. Он имеет самую большую длительность. Ниже расположены вызываемые процессы.

В самом верху указано время запуска всех процессов по таймеру. Процессы имеют разную длительность, потому что каждый процесс выполняет задачу за разное время.

4.2. Второй пример. Алгоритм «Проверка воздушного снайпера»

На рис. 11 показан упрощенный случай. Одна и та же операция повторяется 14 раз. 14 синхронизаторов задают 14 моментов времени, определяющих запуск 14 параллельных процессов.

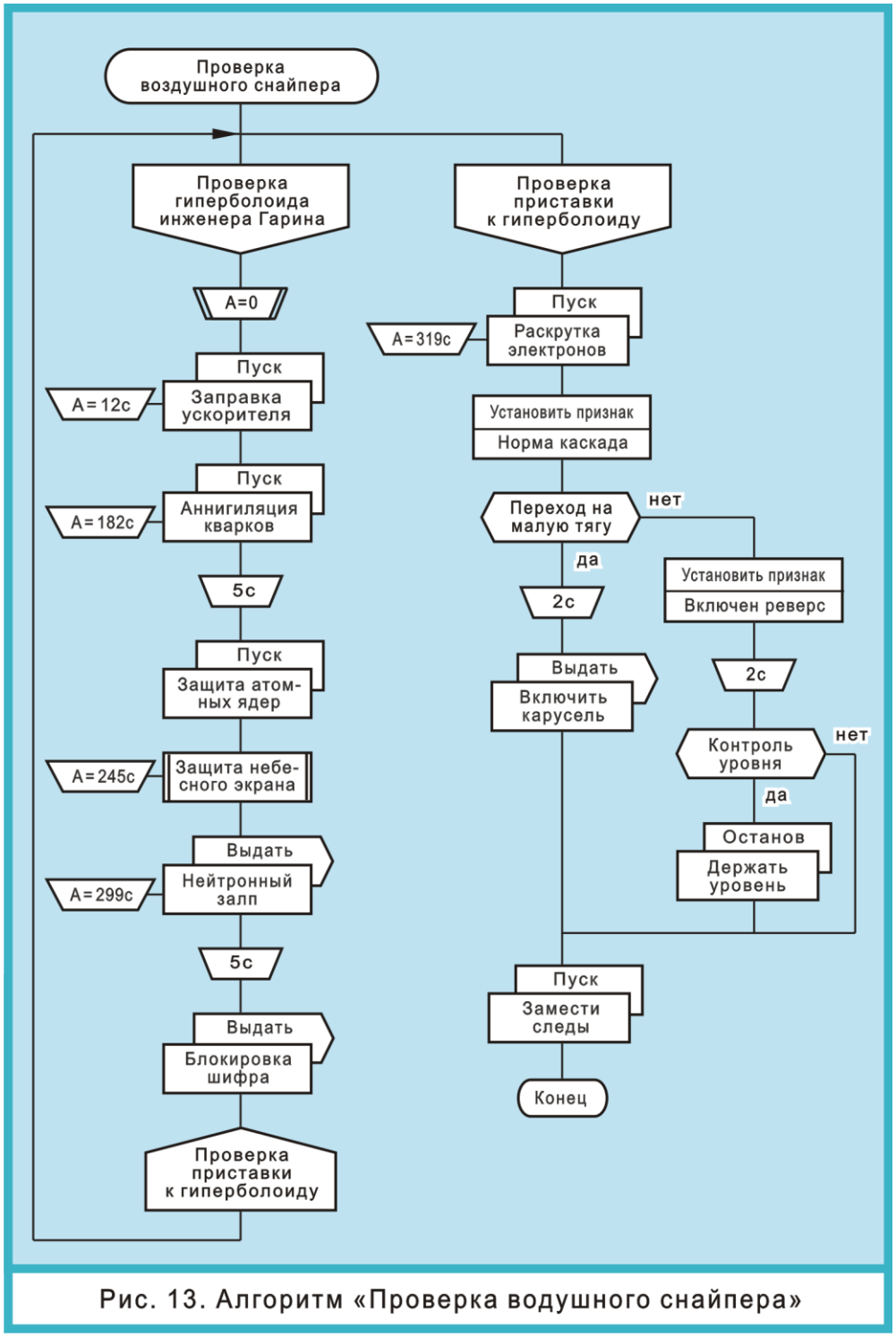


Рис. 13. Алгоритм «Проверка водушного снайпера»

На рис. 13 показан более сложный случай. Наряду с таймером, синхронизатором и процессами применяются следующие иконы: вывод, вставка, вопрос и полка.

В первой ветке имеются 4 синхронизатора. Два из них запускают процессы ЗАПРАВКА УСКОРИТЕЛЯ и АННИГИЛЯЦИЯ КВАРКОВ. Третий включает процедуру ЗАЩИТА НЕБЕСНОГО ЭКРАНА. Четвертый выдает команду НЕЙТРОННЫЙ ЗАЛП.

Используются не только синхронизаторы, но и две паузы длительностью 5 секунд каждая. Первая пауза задерживает пуск процесса ЗАЩИТА АТОМНЫХ ЯДЕР. Вторая задерживает выдачу команды БЛОКИРОВКА ШИФРА.

Во второй ветке выполняются операции:

• через синхронизатор (момент 319 секунд по таймеру) запускается процесс РАСКРУТКА ЭЛЕКТРОНОВ;

• устанавливаются два признака НОРМА КАСКАДА и ВКЛЮЧЕН РЕВЕРС;

• применяются две иконы вопрос: ПЕРЕХОД НА МАЛУЮ ТЯГУ? и КОНТРОЛЬ УРОВНЯ?

• выдается команда ВКЛЮЧИТЬ КАРУСЕЛЬ;

• завершается параллельный алгоритм ДЕРЖАТЬ УРОВЕНЬ;

• и т.д.

В алгоритме на рис. 13 используются пять вызываемых параллельных алгоритмов (процессов). В первой ветке (через синхронизаторы) запускаются два процесса. Во второй ветке применяются три процесса. Один запускается через синхронизатор. Второй – через иконы пауза и вопрос. Третий – через более сложную схему.

5. Как изобразить операторы языка Си на языке Дракон-Си

В качестве примера рассмотрим гибридный язык Дракон-Си. И превратим программы на языке Си в программы на гибридном языке Дракон-Си. В этом случае Си называется *целевым языком*. Пояснения даны на рис. 14 и 15.

В первом примере на рис. 14 рассмотрен оператор *if–else*. В средней графе показана программа на языке Си, в которой используется этот оператор. А в правой графе изображена математически эквивалентная ей программа на языке Дракон-Си.

В чем сходство этих программ?

Некоторые выражения из программы на языке Си без изменения перекочевали в дракон-программу. Вот пример:

• $a>=0$.

В правой графе это выражение помещено в иконе вопрос, которая снабжена выходами «да» и «нет».

Еще пример:

• $x=f1$;

• $y=f2$;

В правой графе эти два оператора присваивания помещены в левую икону действие, присоединенную к выходу «да».

Еще один пример:

• $x=r1$;

• $y=r2$;

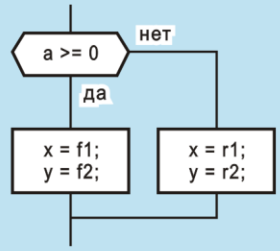
В правой графе эти операторы помещены в правую икону действие, присоединенную к выходу «нет».

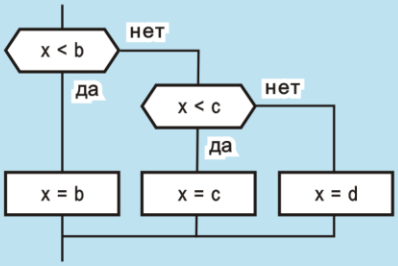
На этом сходство программ кончается. Укажем отличия.

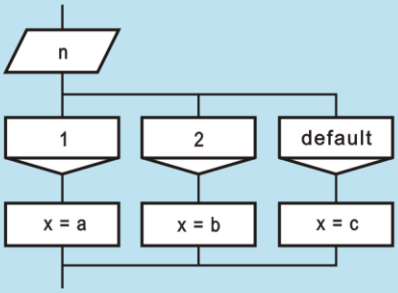
В си-программе мы видим два ключевых слова *if, else,* четыре фигурных скобки и две круглые скобки. В дракон-программе они исчезают и превращаются в чертеж. Благодаря этому схема приобретает важное качество – наглядность.

В остальных примерах на рис. 14 и 15 рассмотрены другие операторы языка Си и показаны соответствующие им правила преобразования программ на языке Си в визуальные программы на языке Дракон-Си.

Операторы языка Си	Программы на языке Си	Программы на языке Дракон–Си

1 if–else	`if ( a >= 0 )`
	`{`
	`  x = f1;`
	`  y = f2;`
	`}`
	`else`
	`{`
	`  x = r1;`
	`  y = r2;`
	`}`
		

2 if– else if– else	`if ( x < b )`
	`  x = b;`
	`else if ( x < c )`
	`  x = c;`
	`else`
	`  x = d;`
		

3 switch, case, break, default	`switch (n)`
	`{`
	`  case 1:`
	`  x = a;`
	`  break;`
	`  case 2:`
	`  x = b;`
	`  break;`
	`  default:`
	`  x = c;`
	`} /* switch */`
		

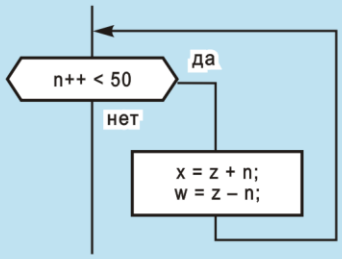
4 while	`while ( n++ < 50 )`
	`{`
	`  x = z + n;`
	`  w = z – n;`
	`} /* while */`
		

Рис. 14. Примеры программ на языке Си и эквивалентные им программы на языке Дракон–Си

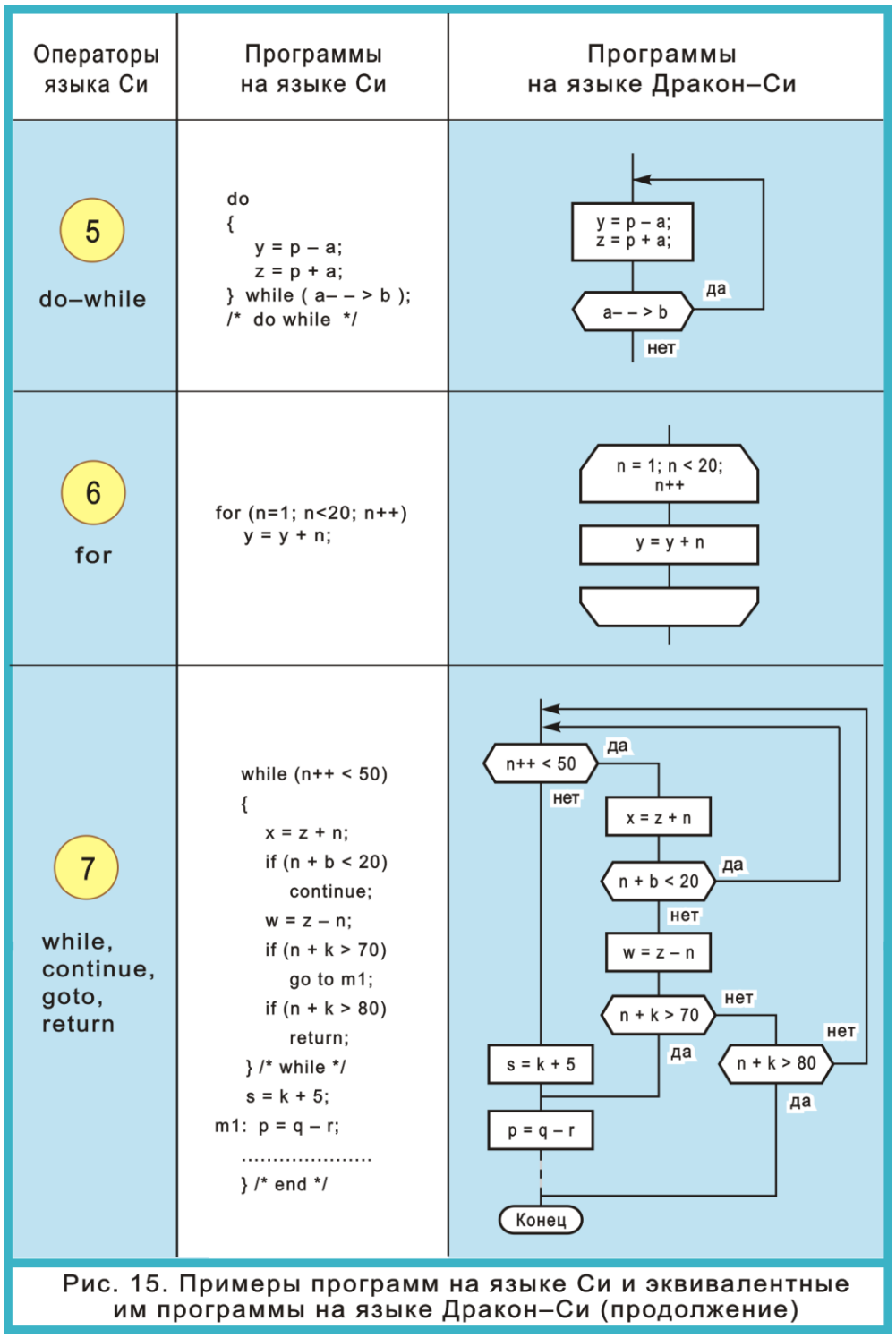


Рис. 15. Примеры программ на языке Си и эквивалентные им программы на языке Дракон–Си (продолжение)

6. Применение языка ДРАКОН в НПЦ АП. Концепция и перспективы развития

Существующие способы записи алгоритмов (принятые во всем мире) слишком трудны для понимания и требуют неоправданно больших трудозатрат.

Это обстоятельство ставит непреодолимый барьер для многих инженерно-технических работников Роскосмоса, работа которых связана с алгоритмами, но которые не имеют резерва времени, чтобы научиться выражать свои профессиональные знания в форме алгоритмов и программ.

При разработке сложных программных комплексов традиционная технология работы имеет существенный недостаток. Он связан с взаимоотношениями между Заказчиком и

Исполнителем. Болевая точка проблемы сконцентрирована на документе под названием «Техническое задание на разработку программного комплекса».

Подразумевается, что инженерно-технические работники совместно с программистами создают Техническое задание на разработку программы. Далее работа по разработке и отработке программного обеспечения возлагается на программистов. А инженерно-технические работники отстраняются от разработки программ.

В НПЦАП используется иной путь, который можно охарактеризовать как «программирование без программистов». При использовании языка ДРАКОН и технологии ГРАФИТ-ФЛОКС разработку алгоритмов и прикладных программ осуществляют в основном не программисты, а инженерно-технические работники.

При этом используется эргономичный способ записи алгоритмов — дракон-схемы. Благодаря этому новшеству алгоритмы становятся более понятными и доступными для ИТР. Дракон-схемы облегчают и ускоряют работу по разработке алгоритмов и прикладных программ для ИТР, то есть для специалистов, не знакомых с программированием.

В итоге ТРУДНЫЕ для понимания способы записи алгоритмов заменяются на более ЛЕГКИЕ. Вследствие этого инженерно-технические работники быстро овладевают дракон-схемами. С помощью конструктора алгоритмов ИТР формализуют свои профессиональные знания, то есть разрабатывают алгоритмы и прикладные программы самостоятельно. Это подтверждает 16-летний опыт эксплуатации технологии ГРАФИТ-ФЛОКС в НПЦ АП.

Такой подход снижает риск появления ошибок, не выявленных по результатам отработки программного обеспечения. Это достигается за счет наглядности, удобочитаемости, эргономичности и строгой формализации дракон-схем и флокс-описаний (описывающих данные), которые помещаются в базу данных ФЛОКС.

Цель, ради которой создан язык ДРАКОН — существенно повысить производительность труда персонала при разработке, анализе и проверке алгоритмов и прикладных программ. Цель достигается за счет использования конструктора алгоритмов, технологии ГРАФИТ-ФЛОКС, эргономичных дракон-схем, а также флокс-описаний, хранящихся в базе данных ФЛОКС. При этом коренным образом изменяется механизм профессионального взаимодействия инженерно-технических работников и программистов. Суть изменения в том, что разработку алгоритмов и прикладных программ выполняют ИТР без участия программистов (или с их минимальным участием).

Секрет успеха заключается в том, что именно инженеры (а отнюдь не программисты) обладают всей полнотой знаний об особенностях функционирования ракет-носителей и разгонных блоков космических аппаратов. Инженеры «первичны», программисты «вторичны» в том смысле, что программисты получают знания об объекте «из вторых рук» — из рук инженеров. Однако до появления языка ДРАКОН инженеры были лишены интеллектуального инструмента, который позволил бы им представить их собственные профессиональные знания в удобной, понятной и одновременно математически строгой форме.

Авторами алгоритмов и прикладных программ являются инженерно-технические работники, которые хорошо знают свою предметную область (физику процесса). Благодаря ДРАКОНу разработка алгоритмов и прикладных программ превращается в формализацию профессиональных знаний инженерно-технических работников.

На выходе конструктора алгоритмов формируются:

— графические дракон-схемы алгоритмов на языке ДРАКОН (для наглядного и удобного восприятия человеком);

— код алгоритма в математически строгой форме, пригодной для преобразования в исполняемый (executable) код.

6.1. Модернизация ДРАКОН-технологии в интересах предприятий Федерального космического агентства

После модернизации и устранения имеющихся недочетов дракон-конструктор обеспечит высокопроизводительную разработку алгоритмов и прикладных программ для ракет-носителей, разгонных блоков, космических аппаратов и других объектов по тематике Роскосмоса, включая объекты конверсионной направленности.

Конструктор алгоритмов рассматривается как начальный этап создания Единой технологии программирования следующего поколения для предприятий Роскосмоса.

Постепенное наращивание функций конструктора алгоритмов и поэтапное превращение его в законченную технологию программирования с улучшенными когнитивно-эргономическими характеристиками имеет хорошо отработанный аналог. Этот аналог создан в НПЦ АП под названием «Технология разработки алгоритмов и программ ГРАФИТ-ФЛОКС».

7. Второй этап разработки языка ДРАКОН. Использование языка ДРАКОН за рамками ракетно-космической отрасли — для решения гражданских задач широкого применения

7.1. Цель разработки

Цель второго этапа — расширить возможности языка ДРАКОН и превратить его из специализированного языка космической отрасли в универсальное средство, пригодное для решения гражданских задач широкого применения.

7.2. Графический и текстовый синтаксис языка ДРАКОН

ДРАКОН — графический (визуальный) язык, в котором используются два типа элементов:

- графические фигуры (иконы);
- текстовые надписи, расположенные внутри или снаружи икон (текстоэлементы) [6, с. 75].

Поэтому язык ДРАКОН имеет не один, а два синтаксиса: графический и текстовый. *Графический (визуальный) синтаксис* охватывает алфавит икон, правила их размещения в поле чертежа и правила связи икон с помощью соединительных линий. *Текстовый синтаксис* задает алфавит символов, правила их комбинирования и привязку к иконам (привязка необходима потому, что внутри разных икон используются разные типы выражений) [6, с. 75].

7.3. Семейство ДРАКОН-языков

ДРАКОН — не один язык, а целое семейство, которое может включать практически неограниченное число ДРАКОН-языков. В состав семейства входит универсальный визуальный алгоритмический язык (являющийся языком моделирования, а не программирования), а также гибридные языки программирования.

Все языки ДРАКОН-семейства имеют одинаковый графический синтаксис, что обеспечивает зрительное сходство дракон-схем различных ДРАКОН-языков. Каждый язык семейства отличается тем, что имеет свой собственный текстовый синтаксис. Строгое разграничение графического и текстового синтаксиса позволяет в максимальной степени расширить сферу применения языков семейства, обеспечивая гибкость и универсальность выразительных средств языка.

При этом единообразие правил графического синтаксиса семейства ДРАКОН-языков обеспечивает их концептуальное единство. Разнообразие текстовых правил (то есть возможность выбора любого текстового синтаксиса), в свою очередь, определяет гибкость языка и лёгкую настройку на различные предметные и иные области [6, с. 263].

7.4. Концепция гибридных языков

На втором этапе разработки языка ДРАКОН предложена концепция гибридных языков программирования. Показано, что императивную (процедурную) часть языка Дракон можно присоединить к некоторым языкам программирования и получить так называемые гибридные языки:

язык Дракон + язык Си = гибридный язык Дракон-Си (см. выше пункт 5)
язык Дракон + язык Java = гибридный язык Дракон-Java
язык Дракон + язык Си# = гибридный язык Дракон-Си#
язык Дракон + язык Питон = гибридный язык Дракон-Питон
язык Дракон + язык Perl = гибридный язык Дракон-Perl
язык Дракон + язык Ruby = гибридный язык Дракон-Ruby
язык Дракон + язык Ада = гибридный язык Дракон-Ада
язык Дракон + язык Оберон = гибридный язык Дракон-Оберон
язык Дракон + язык Tcl = гибридный язык Дракон-Tcl
и т. д.

Пример 1. При создании гибридного языка Дракон-Си необходимо, в частности, создать транслятор из дракон-схемы в исходный код языка Си. В этом случае Си является *целевым* языком.

Пример 2. При создании гибридного языка Дракон-Дельфи необходимо, в частности, создать транслятор из дракон-схемы в исходный код языка Дельфи. При этом Дельфи является *целевым* языком.

При использовании гибридных языков исходным текстом программы считается дракон-схема и только она [6, с. 255, 263-266].

7.5. Как построить гибридный язык программирования

Чтобы построить гибридный язык, нужно выполнить 5 шагов.

Шаг 1. Выбрать целевой язык (например, язык Си).

Шаг 2. Использовать графический синтаксис языка Дракон в качестве графического синтаксиса гибридного языка Дракон-Си.

Шаг 3. Использовать синтаксис целевого языка (синтаксис языка Си) в качестве текстового синтаксиса гибридного языка Дракон-Си.

Шаг 4. Удалить из текстового синтаксиса гибридного языка Дракон- Си все элементы, которые заменяются управляющей графикой ДРАКОНа.

Шаг 5. Создать транслятор из дракон-схемы в исходный код языка Си.

Примечание. Язык Си выбран для примера. Вместо него можно подставить любой целевой язык.

Гибридный язык почти полностью сохраняет концепцию, структуру, типы данных и другие особенности целевого языка. В строго определенном числе случаев текстовая нотация целевого языка заменяется на графическую нотацию Дракона. Такой прием позволяет улучшить эргономический облик гибридного языка и в некоторых областях (например, при программировании микроконтроллеров) повысить производительность труда программистов.

7.6. Инструментальные средства гибридных языков

В рамках концепции гибридных языков некоторые специалисты по собственной инициативе разработали экспериментальные инструментальные средства языка ДРАКОН для гражданских нужд широкого применения для эксплуатации на персональных компьютерах, ноутбуках и т.д.:

— интегрированная среда разработки алгоритмов и программ под названием «ИС Дракон». Разработчик Г.Н. Тышов (Россия, г. Северодвинск) [9].

— DRAKON-Editor. Разработчик S.B. Mitkin (Норвегия, г. Осло) [10].

В результате ДРАКОН превратился в семейство языков моделирования и программирования. Программа ИС Дракон поддерживает работу с гибридными языками программирования Дракон-C, Дракон-Delphi, Дракон-1C, Дракон-ASM, Дракон-Oberon.

Программа DRAKON-Editor обеспечивает работу с гибридными языками Дракон-Java, Дракон-C#, Дракон-C, Дракон-Python, Дракон-Tcl, Дракон-Javascript, Дракон-Lua, Дракон-Erlang [10].

7.7. Аналоги

Аналогом семейства языков ДРАКОН является «R-технология производства программ, или технология двумерного программирования» [11, с. 5], созданная в Институте кибернетики имени В. М. Глушкова, причем графика дракон-схем в ДРАКОН-семействе служит аналогом графики Р-схем [12] в R-технологии.

В технологическом комплексе программиста RTK [11, с. 89-257] принцип обработки информации в компьютере подразумевает деление на R-машину [11, с. 30-35], R-язык [11, с. 18, 19, 42-53, 73-79] и R-технологию [11, с. 17-20]. ДРАКОН использует тот же принцип, выраженный с помощью другого понятийного аппарата.

Аналогом дракон-схем (как алгоритмического языка моделирования) являются диаграммы поведения (behavior diagrams) языка UML, в частности, диаграмма деятельности (activity diagram), диаграмма состояний (UML state machine diagram) и некоторые диаграммы взаимодействия (interaction diagrams), например, диаграмма синхронизации (timing diagram). Другими аналогами дракон-схем являются блок-схема алгоритмов, диаграмма Насси-Шнейдермана, псевдокод (язык описания алгоритмов) и др. В отличие от блок-схем, дракон-схемы имеют средства для описания работы в реальном времени.

8. Дистанционное обучение языку ДРАКОН в Интернете

Для тех, кто не знаком с ДРАКОНом и проявляет интерес к данной теме, можно рекомендовать, например, следующий порядок самостоятельного изучения языка ДРАКОН

8.1. Первый этап. Знакомство

Шаг 1. Бегло посмотрите несколько видео о языке ДРАКОН.

Программа ИС Дракон

Часть 1. http://www.youtube.com/watch?v=Ua9dUUONjdk

Часть 2. http://www.youtube.com/watch?v=zeIq_JQhYSI

Часть 3. http://www.youtube.com/watch?v=Sp6AMGzTM78

Часть 4. http://www.youtube.com/watch?v=1PWDuPeJ_bk

Шаг 2. ***Программа DRAKON Editor***

http://www.youtube.com/watch?v=5IJ8Kf7mwDY

http://www.youtube.com/watch?v=_4PV78oSdwk&feature=endscreen&NR=1

Шаг 3. Посмотрите слайд-фильм о ДРАКОНе скандинавской фирмы Visma (на английском языке).

http://www.slideshare.net/ssuser166d9e/drakon-visual-algorithms

8.2. Второй этап. Помощь начинающим и интернет-консультации

Шаг 1. Прочитайте краткое описание языка ДРАКОН (всего 124 страницы):

http://drakon.su/_media/biblioteka/drakondescription.pdf

Шаг 2. Научитесь рисовать дракон-схемы с помощью программы DRAKON Editor (это легче) и программы ИС Дракон (это труднее, но программа имеет больше возможностей).

Шаг 3. Если что-то непонятно, можно получить помощь и интернет-консультации на официальном форуме языка ДРАКОН http://forum.oberoncore.ru/viewforum.php?f=77

8.3. Третий этап. (Для тех, кого интересует программирование для микроконтроллеров)

Шаг 1. Прочитайте статью «Программирование микроконтроллеров на ДРАКОНе» http://we.easyelectronics.ru/drakon/programmirovanie-mikrokontrollerov-na-drakone.html

Шаг 2. Прочитайте комментарии к этой статье.

Шаг 3. Изучите материалы, представленные на сайте «Алгоритмический язык ДРАКОН» http://drakon-practic.ru/ и тему «Инструменты дракон-схем» http://forum.oberoncore.ru/viewforum.php?f=79

8.4. Четвертый этап. Теоретические основы языка ДРАКОН

Изучите материалы «Конструктор алгоритмов и формальное описание языка» http://drakon.su/_media/biblioteka/chast_6._393-424_konstruktor_algoritmov_i_formalnoe_opisanie.pdf, «Теоретические основы языка ДРАКОН» http://drakon.su/_media/biblioteka/chast_7._425-472_teoreticheskie_osnovy_jazyka_drakon.pdf и книгу [6]. См. также статью ДРАКОН в Википедии.

9. Применение языка ДРАКОН в медицине

9.1. Алгоритмы в медицине. Постановка проблемы

Алгоритмы широко используются в медицинском программировании. Но есть важная область, где алгоритмы используются недостаточно. В самом деле, процесс обследования, диагностики и лечения часто представляет собой некоторую последовательность действий. Следовательно, его можно рассматривать как алгоритм.

Алгоритмические описания часто встречаются во многих медицинских руководствах. Сюда относятся описания иммунологических методов, клиническая лабораторная диагностика, микробиологические инструкции по идентификации микроорганизмов и многое другое.

К сожалению, форма представления этих алгоритмов далека от совершенства. Этот недостаток вносит серьезные затруднения в процесс распространения медицинских знаний. И неблагоприятно отражается на разработке новых методов лечения и совершенствовании существующих.

Можно предположить, что учебные альбомы и компьютерные библиотеки медицинских алгоритмов (дракон-схем) могли бы принести ощутимую пользу в медицинских научных исследованиях. А также в системе медицинского образования и во врачебной практике.

Кроме того, дракон-схемы могут существенно облегчить взаимопонимание медиков между собой и со специалистами смежных профессий. В частности, с медицинскими программистами.

9. 2. Критические замечания и предложения

1. Важным недостатком современной медицины является отсутствие эффективных способов описания *процедурных* (императивных) медицинских знаний. *Процедурные* знания — это знания о точной последовательности процессов и действий, например:

• процессов, протекающих в организме человека,

• действий, выполняемых медицинским персоналом.

2. В медицинской литературе доминирует текст и некачественные схемы и рисунки. Правила создания подобных схем и рисунков достались врачам в наследство от предыдущих эпох, что существенно тормозит фиксацию, понимание и передачу медицинских знаний.

3. Чтобы сделать новый мощный прорыв в развитии медицины, необходимы не только новые медицинские знания, но и новые способы описания знаний.

4. Визуализация медицинских знаний и разработка новых методов описания знаний представляют собой самостоятельное направление научных исследований в области медицины.

5. Язык ДРАКОН – новое средство, пригодное для описания структуры медицинской деятельности и медицинских алгоритмов. Оно позволяет выявить и обнажить логические инварианты деятельности, сделать их явными, зримыми, доступными для всех врачей и студентов-медиков.

6. Язык ДРАКОН кардинальным образом облегчает труд формализации процедурных медицинских знаний и повышает его производительность.

7. Широкомасштабное внедрение языка ДРАКОН для описания процедурных медицинских знаний позволит сделать эти знания намного более ясными, понятными, доходчивыми. Не исключено, что продуманное использование языка ДРАКОН в перспективе позволит сократить сроки медицинского образования. И получить другие важные преимущества.

9. 3. Фактическое использование языка ДРАКОН в медицине

Альгирдас Каралюс (Литва) был первым, кто обратил внимание на возможность крупномасштабного использования языка ДРАКОН в медицине. Инициативу Каралюса активно поддержали многие литовские врачи. За последние год в Литве изданы три медицинских учебника, в которых используется ДРАКОН:

1. Начальная неотложная акушерская помощь [13].
2. Неотложная медицинская помощь [14].
3. Специализированная реанимация новорожденного [15].

Учебники предназначены для медицинских работников скорой помощи, специалистов, работающих в приемных отделениях и учащихся — будущих акушеров, фельдшеров и студентов медицинских вузов. В книге [13] в виде дракон-схем показаны медицинские алгоритмы, часто встречающиеся в практической работе неотложной акушерской помощи при диагностике, реанимации и родовспоможении, например:

- алгоритм «Оценка состояния плода»;
- алгоритм «Дистоция плечиков»;
- алгоритм «Специализированная реанимация новорожденного»;
- алгоритм «Эклампсия»;
- алгоритм «Первичный осмотр пострадавшего»;
- алгоритм «Начальная реанимация и дефибрилляция».

В учебном курсе «Неотложная медицинская помощь» широко используются дракон-схемы в качестве графических инструкций для медицинского персонала. Служба скорой помощи — одно из важнейших звеньев системы оказания медицинской помощи населению. Действия специализированных и линейных бригад скорой помощи должны выполняться очень четко; последовательности таких действий описываются с помощью дракон-схем и называются алгоритмами. В учебном курсе «Неотложная медицинская помощь» имеется глава «Как читать алгоритмы?», поясняющая порядок чтения дракон-схем. Дракон-схемы наглядно показывают неотложные спасательные действия и процедуры, которые должны точно и безупречно выполнять работники скорой помощи при угрожающих жизни пациента состояниях. В начале учебного курса говорится: «Последовательность сложных или более

важных действий написана в алгоритмах, подготовленных по методике языка ДРАКОН. Цель алгоритмов — помочь как можно лучше запомнить последовательность действий при оказании неотложной медицинской помощи» [13, с. 5, 19].

Список литературы

1. *Пароnджанов В. Д.* Графический синтаксис языка ДРАКОН // Программирование, 1995, №3, С. 45—62.
2. Штурманы ракет / Под общей редакцией Е. Л. Межирицкого. — М.: БЛОК-Информ-Экспресс, 2008. — 384 с. — С. 192. — ISBN 978-5-93735-008-4
3. *Н. Бурцева, Е. Петров.* Жирограф и ДРАКОН Пилюгина. Телерадиостудия Роскосмоса. (17 мая 2008). — Фильм выпущен к 100-летию со дня рождения Главного конструктора систем управления ракет-носителей, академика Н. А. Пилюгина.
4. *Пароnджанов В. Д.* Как улучшить работу ума. Алгоритмы без программистов — это очень просто!. — М.: Дело, 2001. — 360 с. — С. 31. — ISBN 5-7749-0211-0
5. Структурная схема информационной технологии «ГРАФИТ-ФЛОКС» http://store.oberoncore.ru/lib/paper/grafit_A4.pdf
6. *Пароnджанов В. Д.* Учись писать, читать и понимать алгоритмы. Алгоритмы для правильного мышления. Основы алгоритмизации. — М.: ДМК Пресс, 2012. — 520 с. — ISBN 978-5-94074-800-7
   (Учебное пособие по языку ДРАКОН подготовлено в соответствии с «Примерной программой дисциплины „Информатика“. Издание официальное. — М.: Госкомвуз, 1996. — 21 с. / См. разделы 3 и 4, а также Приложение, пункты 1-7.»).
7. *Морозов В.В.,. Трунов Ю.В., Комиссаров А.И, Пак Е.А., Жучков А.Г., Дишель В.Д., Залихина Е.Е., Пароnджанов В.Д.* Система управления межорбитального космического буксира «Фрегат» // Вестник «ФГУП НПО им. С.А. Лавочкина», 2013 (в печати).
8. *Пароnджанов В. Д.* Дружелюбные алгоритмы, понятные каждому. Как улучшить работу ума без лишних хлопот. — М.: ДМК-пресс, 2010. — 464 с. — С. 13. — ISBN 978-5-94074-606-5
9. Программу ИС Дракон (автор Геннадий Тышов) можно скачать здесь: http://forum.oberoncore.ru/viewtopic.php?p=22669#p22669
   http://forum.oberoncore.ru/viewforum.php?f=79
10. Программу DRAKON Editor (автор Stepan Mitkin) можно скачать здесь: http://sourceforge.net/projects/drakon-editor/files/
11. *Вельбицкий И. В., Ходаковский В. Н., Шолмов Л. И.* Технологический комплекс производства программ на машинах ЕС ЭВМ и БЭСМ-6. — М.: Статистика, 1980. — 263с.
12. Межгосударственный стандарт ГОСТ 19.005-85. Единая система программной документации. Р-схемы алгоритмов и программ. Обозначения условные графические и правила выполнения. Unified system for program documentation. R-charts. Graphical chart symbols and conventions for charting. 1985.
13. Начальная неотложная акушерская помощь. Учебник. / Под ред. профессора Рута Йоланта Надишаускене. — Литва: Центр исследования кризисов, Университет наук здоровья Литвы, 2012. — 204 с. — ISBN 978-609-8033-61-8
14. Неотложная медицинская помощь. Материалы курса. — Литва: Центр исследования кризисов, Каунасский медицинский университет, 2012. — 265 с.
15. Специализированная реанимация новорожденного. Учебник / Под ред. профессора Рута Йоланта Надишаускене. — Литва: Центр исследования кризисов, Университет наук здоровья Литвы, 2012. — 400 с.