Государственная корпорация «Роскосмос»
Научно-производственный центр автоматики и приборостроения имени академика Н. А. Пилюгина

НУЖЕН ЛИ ВООРУЖЕННЫМ СИЛАМ РОССИИ И ДРУГИМ СТРУКТУРАМ МИНИСТЕРСТВА ОБОРОНЫ РФ СТАНДАРТ ДЛЯ ОПИСАНИЯ АЛГОРИТМОВ?

Владимир Даниелович ПАРОНДЖАНОВ vdp2007@bk.ru +7-916-111-91-57

Цель документа — обосновать необходимость введения **качественного стандарта для описания алгоритмов**, используемых при разработке и эксплуатации вооружений, а также при функционировании Вооруженных Сил России, включая медицинское и тыловое обеспечение Вооруженных Сил.

Для военно-промышленной отрасли большое значение имеет безошибочность и надежность алгоритмов. Однако действующий стандарт на алгоритмы ГОСТ 19.701—90 не способен решить эту задачу. Более того, он провоцирует и стимулирует появление ошибок, т. е. является своеобразным катализатором ошибок. Я считаю, что данный стандарт устарел и тормозит развитие военно-промышленной отрасли; пользоваться им для записи алгоритмов недопустимо.

Чтобы устранить дефект, я предлагаю принять стандарт визуального алгоритмического языка ДРАКОН ***в качестве стандарта для описания алгоритмов Вооруженных Сил России, включая медицинское и тыловое обеспечение Вооруженных Сил, а также для оборонно-промышленного комплекса РФ.***

Принятие алгоритмического стандарта Вооруженных Сил России на основе языка ДРАКОН позволит:

- значительно сократить число ошибок в алгоритмах, программах и алгоритмических предписаниях (жизнеритмах) военного, военно-промышленного и двойного назначения, а также в целом по оборонно-промышленному комплексу;
- улучшить взаимопонимание между специалистами различных структур и служб Министерства обороны РФ на основе передового опыта, связанного с переходом на использование ***эргономичных алгоритмов высокой точности***, обеспечивающих надежность, удобочитаемость, быстрое восприятие, точное понимание и легкое усвоение алгоритмических знаний;
- облегчить и ускорить овладение алгоритмами для военного образования, а также для тех структур Министерства обороны, которые лишь недавно начали использовать алгоритмы, например, для военной медицины.

Стандарт визуального алгоритмического языка ДРАКОН подробно описан в моих монографиях:

1. Паронджанов В.Д. Алгоритмы и жизнеритмы на языке ДРАКОН. Разработка алгоритмов. Безошибочные алгоритмы. — М., 2019. — 374 с. — https://drakon.su/_media/24_zhizneritm20.pdf.
2. Паронджанов В.Д. Учись писать, читать и понимать алгоритмы. Алгоритмы для правильного мышления. Основы алгоритмизации. — М.: ДМК Пресс, 2012, 2014, 2016. — 520 с. — http://bit.ly/2Mlg4Ou.
3. Паронджанов В.Д. Почему врачи убивают и калечат пациентов, или Зачем врачу блок-схемы алгоритмов? Иллюстрированные алгоритмы диагностики и лечения — перспективный путь развития медицины. Клиническое мышление высокой точности и безопасность пациентов. / Предисловие члена-корреспондента РАН Г.В. Порядина. — М.: ДМК Пресс, 2017. — 340 с.

В издательстве «Юрайт» вскоре выйдет из печати учебное пособие: «Паронджанов В.Д. Алгоритмические языки и программирование: ДРАКОН».

Учебное пособие будет издано в двух сериях:

— для высшего профессионального образования;

— для среднего профессионального образования.

СОДЕРЖАНИЕ

Часть 1. СТАНДАРТИЗАЦИЯ АЛГОРИТМОВ ВООРУЖЕННЫХ СИЛ РФ И ОБОРОННО-ПРОМЫШЛЕННОГО КОМПЛЕКСА

ВВЕДЕНИЕ

В настоящем документе (Части 1 — 7) представлены тщательно обоснованные аргументы, подтверждающие необходимость принять стандарт языка ДРАКОН в качестве стандарта для описания алгоритмов Вооруженных Сил России, включая медицинское и тыловое обеспечение Вооруженных Сил, а также для оборонно-промышленного комплекса РФ.

Принятие алгоритмического стандарта Вооруженных Сил России позволит:

- значительно сократить число ошибок в алгоритмах, программах и алгоритмических предписаниях (жизнеритмах) военного и военно-промышленного назначения, а также в целом по оборонно-промышленному комплексу;
- улучшить взаимопонимание между специалистами различных структур и служб Министерства обороны РФ на основе передового опыта, связанного с переходом на использование ***эргономичных алгоритмов высокой точности***, обеспечивающих надежность, удобочитаемость, быстрое восприятие, точное понимание и легкое усвоение алгоритмических знаний;
- облегчить и ускорить овладение алгоритмами для военного образования, а также для тех структур Министерства обороны, которые лишь начинают использовать алгоритмы, например, для военной медицины.

ИСТОРИЯ ВОПРОСА

При создании системы управления орбитального корабля «Буран» многоразовой транспортной космической системы «Энергия — Буран» был разработан визуальный алгоритмический язык ДРАКОН, предназначенный для программирования, моделирования и обучения. Разработка языка велась с 1986 года при участии Научно-производственного центра автоматики и приборостроения им. акад. Н.А. Пилюгина (НПЦАП) 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] [2] [3] [4] [5] [6].

Основной задачей разработчиков было создание единого языка, который своей доступностью и мощностью способен заменить специализированные языки: ПРОЛ2 (для разработки бортовых комплексных программ Бурана для БЦВМ «Бисер-4»), ДИПОЛЬ (для создания наземных программ Бурана) и ЛАКС (для моделирования).

Работы по созданию ДРАКОНа были в основном закончены в 1996 году (спустя три года после закрытия программы «Буран»), когда была разработана автоматизированная система проектирования программных средств (ДРАКОН-технология).

ДРАКОН можно определить как общедоступный визуальный язык, предназначенный:

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

- для описания структуры человеческой деятельности, потоков работ (workflows) и бизнес-процессов;
- для проектирования, программирования, моделирования и обучения [6] с. 31.

Правила языка ДРАКОН по созданию дракон-схем (drakon-charts) оптимизированы для восприятия и понимания алгоритмов ЧЕЛОВЕКОМ. Язык предлагается в качестве инструмента усиления человеческого интеллекта.

РЕЗУЛЬТАТЫ

Язык ДРАКОН и ДРАКОН-технология программирования эксплуатируются в НПЦАП почти 25 лет (1996–2020 годы). За это время они помогли создать системы управления четырнадцати космических проектов:

Рис. 1. Язык программирования ДРАКОН был создан для орбитального корабля Буран, чтобы заменить три языка: ПРОЛ2, ДИПОЛЬ, ЛАКС

- разгонный блок космических аппаратов ДМ-SL (проект «Морской старт»);
- модернизированная ракета-носитель тяжелого класса ПРОТОН-М;
- семейство разгонных блоков ФРЕГАТ, которое включает: просто ФРЕГАТ, ФРЕГАТ-СБ (со сбрасываемыми баками), ФРЕГАТ-УТТХ (с улучшенными тактико-техническими характеристиками), ФРЕГАТ-МТ, предназначенный для запусков из Южной Америки, ФРЕГАТ-СУМ1 (с модернизированной системой управления);
- разгонный блок космических аппаратов ДМ-SLБ (проект «Наземный старт»),
- разгонный блок космических аппаратов ДМ-03,
- первая ступень южнокорейской ракеты-носителя KSLV,
- три проекта семейства Ангара: 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 КВТК (кислородно-водородный тяжелого класса) [3, с. 515] [7].

ГЕОГРАФИЯ РАКЕТНЫХ ПУСКОВ

Пуски ракет-носителей и космических разгонных блоков, при создании которых использовался язык ДРАКОН, за истекший период выполнялись с шести космодромов мира, расположенных на трех континентах (Европа, Азия, Америка) и в океане:

1. космодром Плесецк;
2. космодром Байконур;
3. космодром Восточный;
4. Европейский космический центр во Французской Гвиане «Kourou»;
5. южнокорейский космодром «Naro»;
6. международный плавучий космодром, производящий пуски в экваториальной зоне Тихого океана вблизи острова Рождества Республики Кирибати.

Рис. 2. Старт ракеты-носителя с межорбитальным космическим буксиром Фрегат. Система управления Фрегата разработана с помощью ДРАКОН-технологии [8].

ЯЗЫК ДРАКОН ЗА РАМКАМИ НПЦАП

Язык ДРАКОН применяется не только в НПЦАП, но и за его пределами — в областях, далеких от ракетно-космической тематики. В 1996 году Государственный комитет Российской Федерации по высшему образованию включил изучение языка ДРАКОН в программу курса «Информатика» [9]. В некоторых университетах организовано обучение ДРАКОНу, например, в СибГИУ, КемГУ и др.

При этом используется качественно иной (по сравнению с НПЦАП) принцип работы. ДРАКОН можно присоединить к некоторым языкам программирования и получить так называемые гибридные языки, например: Дракон-C, Дракон-Java, Дракон-C#, Дракон-Python, Дракон-Javascript, Дракон-Tcl, Дракон-Erlang, Дракон-Lua и др.

В Интернете можно скачать несколько бесплатных программ «ДРАКОН-конструктор» для работы с языком ДРАКОН. Более полно эта сторона дела отражена в книгах [2] [3] [5], в Википедии: в статьях ДРАКОН [10], DRAKON [11] и на официальном сайте [12] и форуме языка ДРАКОН [13].

ГЛАВНАЯ МЫСЛЬ

Тщательный анализ результатов использования алгоритмического языка ДРАКОН в ракетно-космической технике, а также в других предметных областях (медицина, стратегическое управление, бизнес-процессы) позволяет сделать обобщение и выдвинуть следующий тезис:

Основной тезис

- Стандарт визуального языка ДРАКОН во всех отношениях превосходит международный стандарт на схемы алгоритмов ISO 5807-85 и построенный на его основе ГОСТ 19.701-90.
- Язык ДРАКОН имеет значительные преимущества по сравнению с конкурирующими нотациями (UML, псевдокод и др.) для записи алгоритмов и алгоритмических предписаний (жизнеритмов).
- В связи с этим предлагается принять стандарт языка ДРАКОН в качестве стандарта для описания алгоритмов Вооруженных Сил России и других структур Министерства обороны РФ.

Оставшаяся часть документа посвящена разъяснению и обоснованию тезиса.

ПРОСТОЙ ПРИМЕР

Цель примера — показать, как выглядят «дракон-схемы» и убедиться, что ими легко и удобно пользоваться даже людям, совсем не знакомым с программированием. Дракон-алгоритмы красиво нарисованы, эргономичны, в них нет пересечений и обрывов линий; они значительно удобнее, доступнее и нагляднее, чем блок-схемы по ГОСТ 19.701-90, схемы деятельности (activity diagrams) языка UML и псевдокод.

Пример состоит из четырех чертежей, иллюстрирующих принцип декомпозиции алгоритмов. На первой дракон-схеме представлен общий план алгоритма, состоящий из девяти «вставок», или процедур (рис. 3). На остальных чертежах развернута детализация этих процедур.

Мы выбрали простой бытовой пример, понятный с первого взгляда. Общий план технологии на рис. 3 является введением в рассматриваемую проблему. В основной части (рис. 4 – 6) поочередно раскрывается каждый пункт плана и дается детальное описание технологии методом декомпозиции.

Сверху (на рис. 3) в иконе Заголовок видим название алгоритма: «Технология выращивания помидоров на садовом участке». Чуть ниже на жирных вертикалях («шампурах») расположены выполняемые процедуры. Шампуры выполняются последовательно, друг за другом: сначала левый, потом средний, затем правый.

Предположим, бегунок (рабочая точка алгоритма) опустился в конец первого шампура и попал в икону «Уход за сеянцами и рассадой». Возникает вопрос: куда двигаться дальше? Ответ прост: бегунок прыгнет вверх в начало второго шампура. Почему? Потому что именно там записано такое же название (метка). Происходит как бы переход на заданную метку (goto). Точно так же осуществляется следующий переход из второго шампура в третий, на этот раз на метку «Работа в теплице».

А теперь самое главное: это не просто метки, расположенные в верхнем ряду схемы. Это *эргономичные путеводители по алгоритму.* Они дают СОДЕРЖАТЕЛЬНОЕ название каждому шампуру и разбивают алгоритм на три смысловых части:

- Подготовка и посев семян.
- Уход за сеянцами и рассадой.
- Работа в теплице.

Благодаря этому осуществляется структуризация алгоритма, причем полученная структура предъявляется глазам читателя в удобной для восприятия форме.

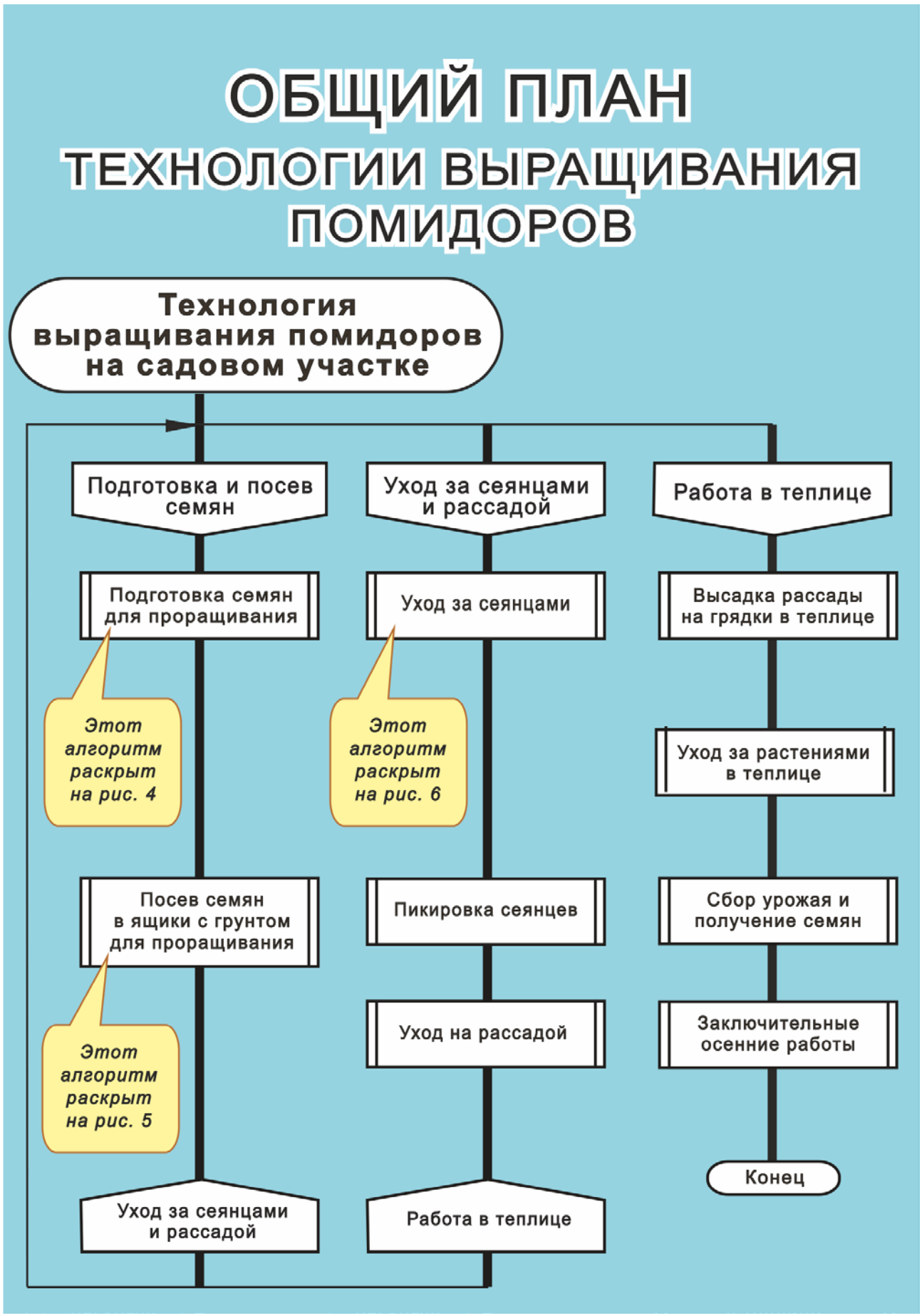


Рис. 3. Технология выращивания помидоров на садовом участке

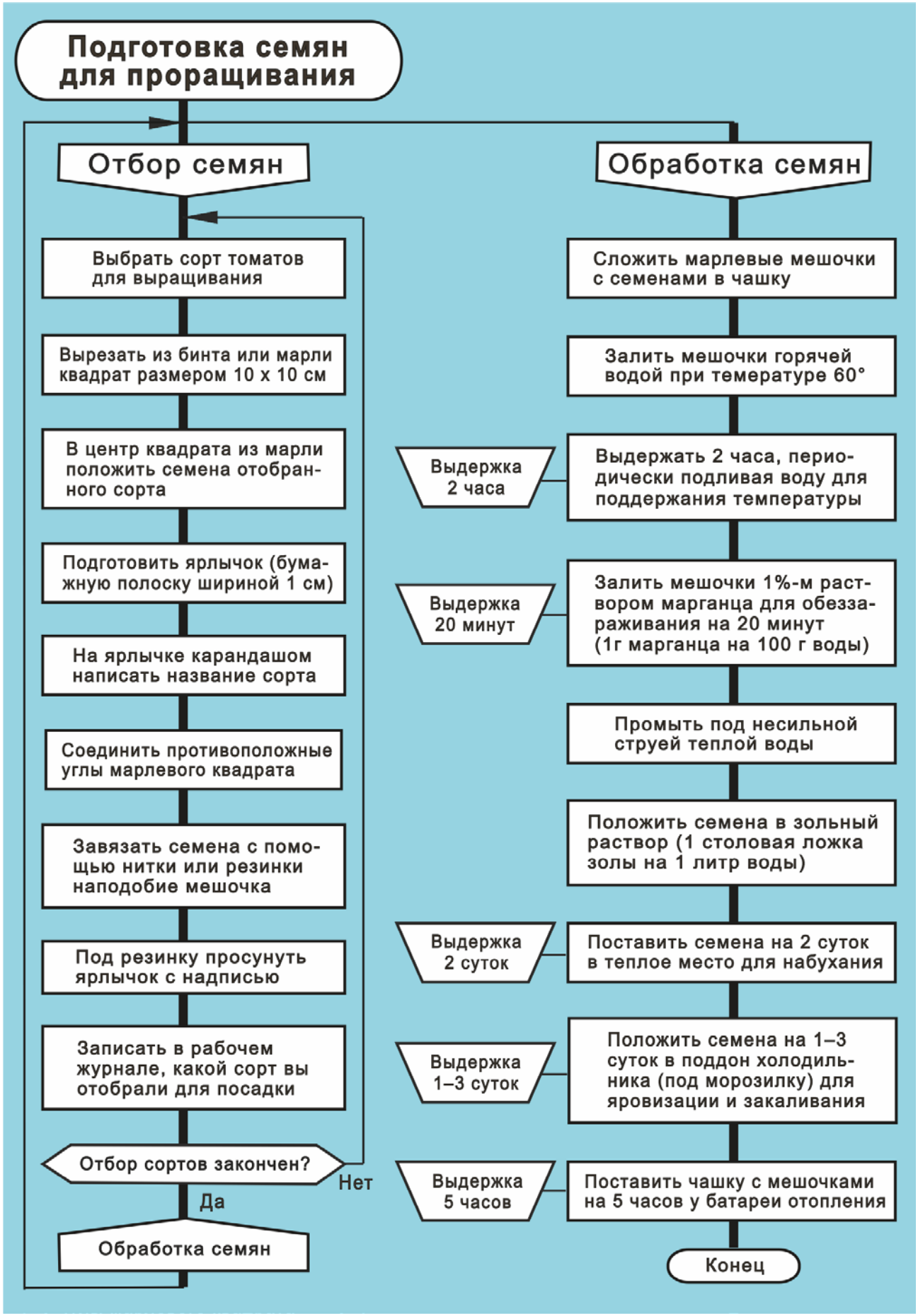


Рис. 4. Подготовка семян для проращивания

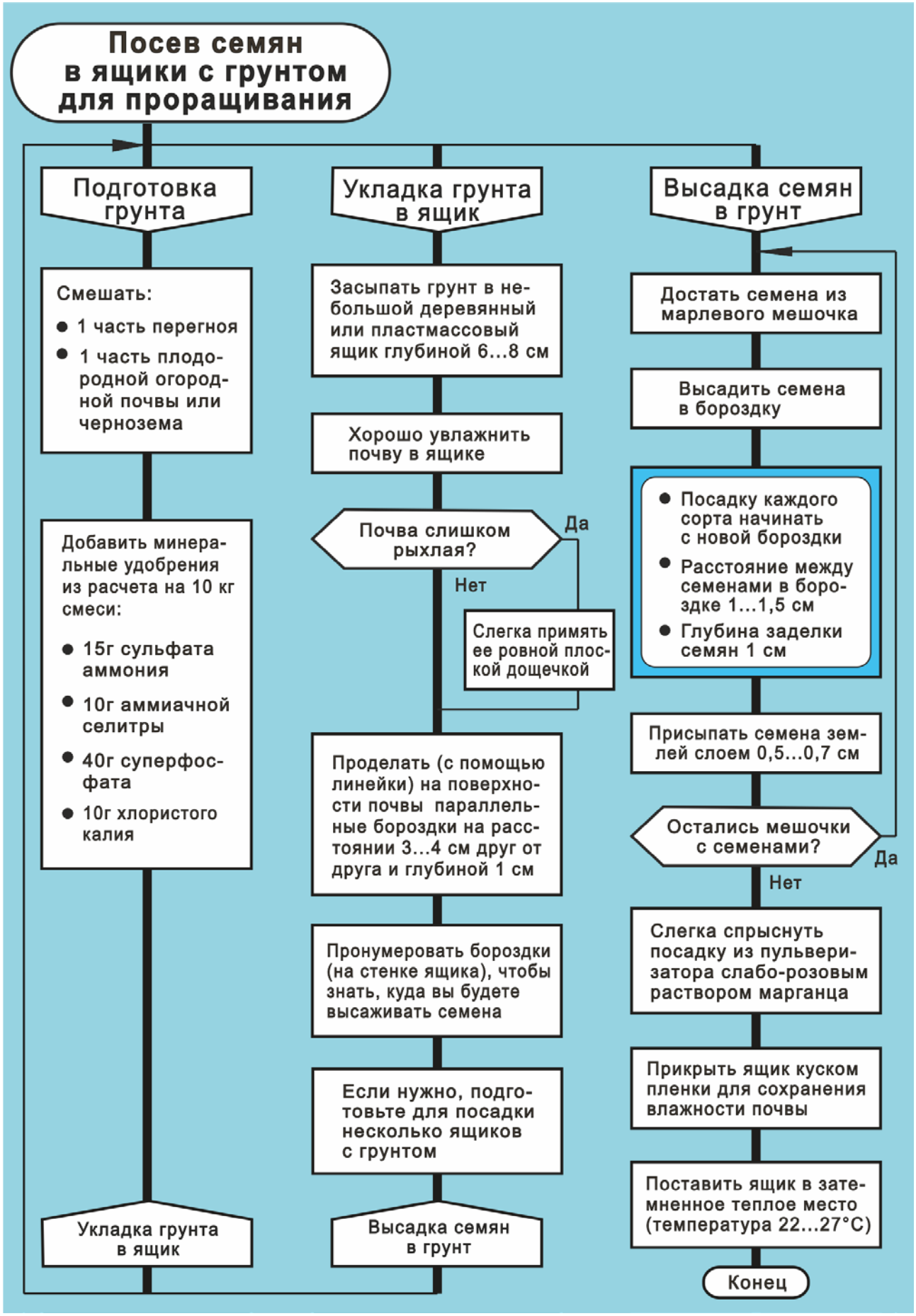


Рис. 5. Посев семян в ящики с грунтом для проращивания

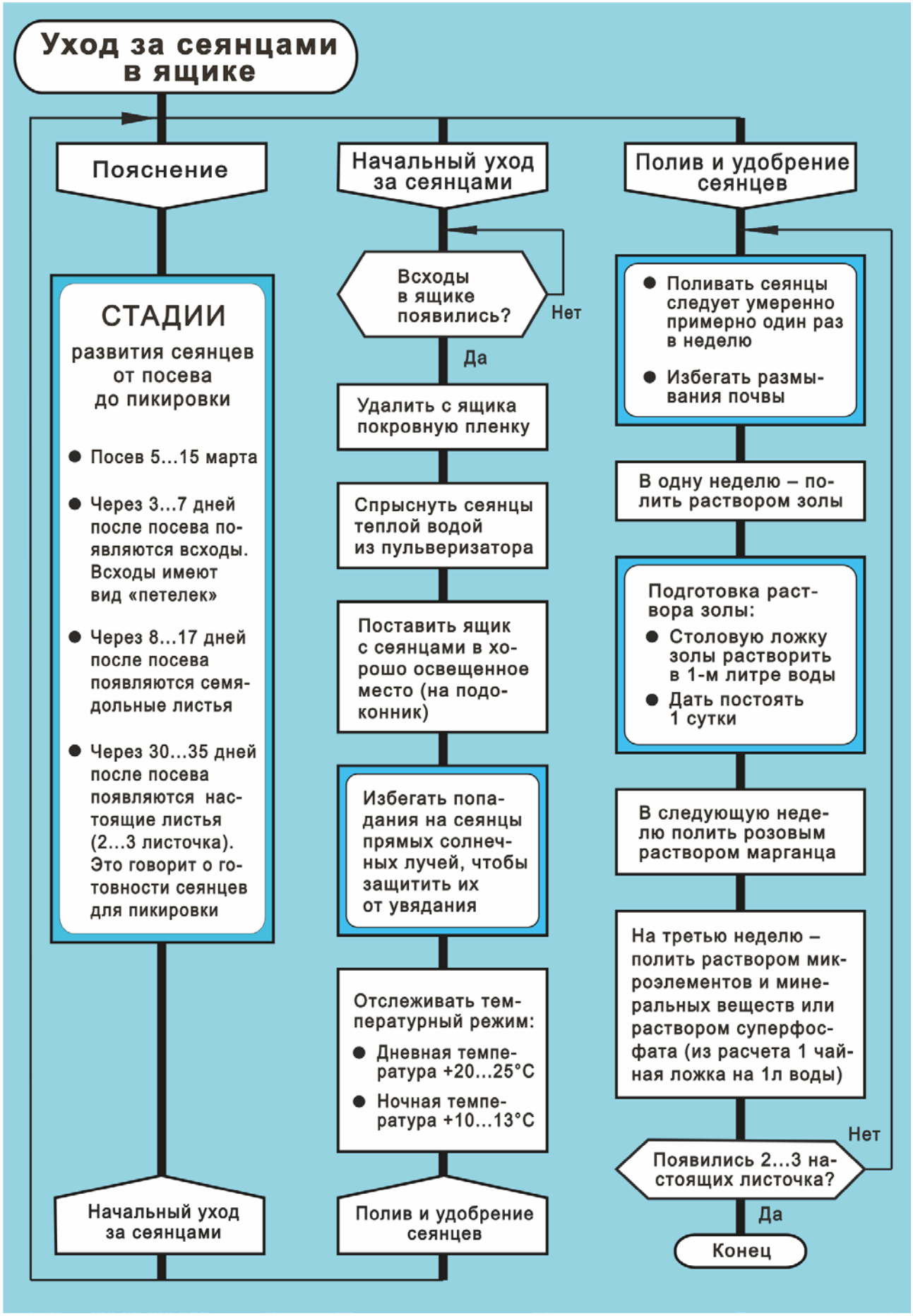


Рис. 6. Уход за сеянцами в ящике

Рассмотрим еще один эргономичный прием. Дракон-схема показывает читателю *все без исключения маршруты* алгоритма. Читатель получает возможность провести пальцем от начала до конца алгоритма, не отрывая палец от бумаги (экрана). Для этого проведите пальцем по линии контура вокруг схемы по часовой стрелке.

Итак, наш алгоритм содержит девять процедур. Три из них раскрыты на рис. 4 – 6.

- На рис. 4 раскрыт алгоритм «Подготовка семян для проращивания».
- На рис. 5 показан алгоритм «Посев семян в ящики с грунтом».
- На рис. 6 представлен «Уход за сеянцами в ящике».

Алгоритмы заимствованы из брошюры [14]. В брошюре все технологические процессы показаны в виде подробных дракон-схем. Пример сложного алгоритма из ракетно-космической области показан ниже в Части 3 на рис. 15.

ЭРГОНОМИЧНАЯ ДЕКОМПОЗИЦИЯ И ВЫВОД

Мы рассмотрели пример, использующий простую двухступенчатую декомпозицию. Вместе с тем пример показывает, что глубина декомпозиции ничем не ограничена. Если на чертежах 4 – 6 использовать дополнительные вставки (процедуры), то, раскрывая их, мы легко получим третью ступень декомпозиции. Затем четвертую, и далее по индукции.

В электронном варианте, щелкая мышью по процедурам, мы моментально переходим в нужный чертеж, где изображена соответствующая процедура.

Эргономичная декомпозиция сложных алгоритмов, а также тщательно разработанные эргономичные правила представления сложных алгоритмов, описанные в книгах [2] [3], позволяют существенно упростить и облегчить алгоритмическую задачу. И тем самым преобразовать и усовершенствовать запутанный алгоритм, «распутать» его, улучшить форму представления алгоритма, сделать ее эргономичной (people-friendly), наглядной и удобной для личного состава и курсантов военно-учебных заведений.

Вывод

- Язык ДРАКОН обладает мощными средствами эргономичной декомпозиции.
- Благодаря этому любой, сколь угодно сложный и запутанный алгоритм можно представить в виде стройного комплекта ясных, доходчивых и дружелюбных (people-friendly) дракон-схем.
- Дракон-схемы обеспечивают быстрое восприятие, точное понимание и легкое усвоение алгоритмических знаний, используемых в Вооруженных Силах России.

ЭРГОНОМИЧНЫЙ АЛГОРИТМ

Если алгоритм непонятный, в нем трудно или даже невозможно заметить затаившуюся ошибку. И наоборот, чем понятнее алгоритм, тем легче найти дефект. Поэтому более понятный, эргономичный алгоритм намного лучше обычного. Лучше в том смысле, что он облегчает выявление ошибок, а это очень важно.

Ведь чем больше ошибок удастся обнаружить при визуальной проверке за столом (до проведения моделирования, тестирования и приемо-сдаточных испытаний), тем больше вероятность, что вновь созданный алгоритм окажется правильным, безошибочным, надежным. Кроме того, эргономичные алгоритмы удобны для изучения, их намного проще объяснить другому человеку.

Важным качеством алгоритма является его удобочитаемость (понимаемость). Известно, что

> «качественная программа должна обладать, помимо надёжности и эффективности, ещё и такими важнейшими свойствами как понимаемость и сопровождаемость» [15].

Согласно ГОСТ 28806-90 *понимаемость* определяется как «совокупность свойств программного средства, характеризующая затраты усилий пользователя на понимание логической концепции этого программного средства» [16].

Преимущество эргономичных алгоритмов в том, что они намного понятнее, яснее, нагляднее и доходчивее, чем обычные. Эта особенность эргономичных алгоритмов закреплена терминологически с помощью термина «сверхвысокая понимаемость», что является отличительной чертой эргономичных алгоритмов.

ЭРГОНОМИЧНЫЕ АЛГОРИТМЫ В ВООРУЖЕННЫХ СИЛАХ

Эргономичный алгоритм – это алгоритм, удовлетворяющий критерию сверхвысокого понимания. Чтобы облегчить и ускорить изучение алгоритмов военного и двойного назначения, понятие «эргономичный алгоритм» является полезным и играет все возрастающую роль. Данный прием позволяет использовать удобную для личного состава, представителей военной приемки и курсантов военно-учебных заведений эргономичную форму представления алгоритмов. Изюминка в том, что алгоритм превращается в легкий для понимания рисунок (чертеж). При этом достигается синергетический (взрывной) эффект, который состоит в том, что происходит значительное облегчение работы с алгоритмами. В литературе подробно объясняются преимущества эргономичных алгоритмов [2– 6].

Представители Вооруженных Сил как заказчики новых сложных вооружений получат возможность более качественно контролировать процесс принятия системы на вооружение, если представители промышленности и предприятий-разработчиков будут включать в состав документации на систему вооружений комплект эргономичных алгоритмов, выполненных по стандарту языка ДРАКОН.

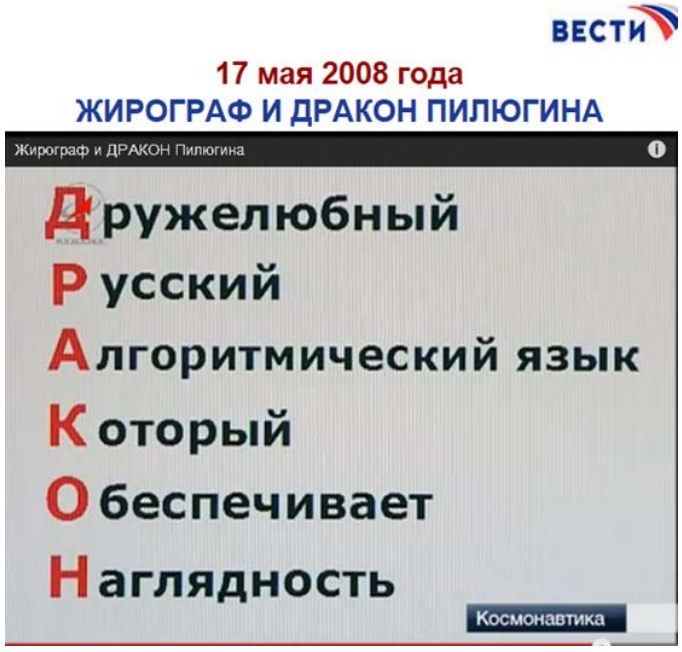


Рис. 7. Кадр из фильма «Жирограф и ДРАКОН Пилюгина». Телестудия Роскосмоса. Фильм выпущен к 100-летию со дня рождения Главного конструктора систем управления ракет-носителей академика Н. А. Пилюгина [17]

АЛГОРИТМЫ И ЖИЗНЕРИТМЫ

С математической точки зрения, алгоритм на рис. 3–6 — это, строго говоря, не алгоритм. А что же это такое? Это всего лишь алгоритмическое предписание (далее, для краткости, жизнеритм). Жизнеритм, по сравнение с алгоритмом, вещь неточная, математически неполноценная.

Между алгоритмом и жизнеритмом имеются существенные отличия. Необходимо точно выявить эти отличия. Мы будем опираться на тезис, который сформулировал математик Н.Н. Непейвода [18]:

> «Необходимо различать алгоритм и алгоритмическое предписание, имеющее внешнюю форму алгоритма, но включающее не до конца определенные шаги».

На этот счет существует литература, восходящая к пионерской работе Льва Ланды «Алгоритмизация в обучении» [19]. Ланда первым указал различие, но мы предпочитаем более точную формулировку Непейводы.

Тезис Ланды-Непейводы создает научный фундамент для четкого разграничения алгоритмов и жизнеритмов.

Определенность (детерминированность) алгоритма говорит о том, что каждое указание алгоритма должно быть точным, четким, однозначным и не оставлять места для случайных, произвольных толкований и действий. Жизнеритм — в отличие от алгоритма — не обладает свойством определенности.

В тезисе Непейводы есть фраза «не до конца определенные шаги». Что она означает?

Смысл в том, что в жизнеритме отсутствует строгий математический порядок. Некоторые правила описания и выполнения шагов алгоритма или не указаны, или неизвестны, или не соблюдаются.

Таким образом, в жизнеритме (в отличие от истинного алгоритма) нет необходимой точности, нет однозначности. В связи с этим возникает серьезная проблема.

ВАЖНОЕ ПРОТИВОРЕЧИЕ

Проблема жизнеритмов связана с необходимостью осознать противоречие, о котором часто забывают. С одной стороны, наличие не полностью определенных шагов является важным дефектом, которое исключает возможность использования жизнеритмов в компьютерном программировании.

С другой стороны, жизнеритмы в программировании не нуждаются. Их ценность в другом. Графические (визуальные) жизнеритмы выполняют следующие функции и социальные роли:

- позволяют осуществлять компьютерное моделирование сложных разветвленных и циклических процессов;
- упрощают сложные текстовые описания, заменяя их удобными графическими рисунками; обеспечивают взаимопонимание между людьми;
- выполняют важную дидактическую (педагогическую) функцию, превращая сложные задачи в относительно простые.

Таким образом, за рамками компьютерного программирования существует множество сложных человеческих и социальных проблем, связанных с описанием человеческих и автоматических действий. Жизнеритмы выполняют важнейшую социальную функцию, связанную с моделированием и решением этих проблем.

Указанные проблемы имеют самое непосредственное отношение к медицинскому, тыловому и боевому обеспечению Вооруженных Сил. Поясним суть дела на примере медицины и проблемы безопасности пациентов.

Часть 2. КЛИНИЧЕСКИЕ АЛГОРИТМЫ И МЕДИЦИНСКАЯ СЛУЖБА МИНИСТЕРСТВА ОБОРОНЫ РФ

КЛИНИЧЕСКИЙ АЛГОРИТМ И ЕГО ОСОБЕННОСТИ

Клинические алгоритмы – это любые клинические действия и решения, а также составленные из них сложные и разветвленные цепочки операций, выполняемые при профилактике, диагностике, лечении, экстренной помощи, реанимации, реабилитации, прогнозе. См. пример на рис. 8.

Уточним: алгоритмами являются не сами действия и решения, а их точные пошаговые описания на бумаге или экране.

Понятие *алгоритм* пришло в медицину недавно и не имеет единого строгого определения, допускает различные толкования и графические представления. Стандартизация отсутствует. Главный недочет большинства клинических трактовок понятия *алгоритм* — отсутствие необходимой точности, приблизительность, недосказанность, отсутствие нужных уточнений, подробностей и условий.

В медицинских учебниках, стандартах, клинических руководствах используются неточные, приблизительные, неудовлетворительные описания клинических алгоритмов. Подобные описания нарушают элементарные алгоритмические и эргономические правила, провоцируют врачебные ошибки и опасны для пациентов.

Все эти погрешности зачастую исключают возможность воспроизвести алгоритм, опираясь на его описание. Это значит, что нельзя воспроизвести (повторить) процессы диагностики и лечения, используя только алгоритмы. Потому что указанные процессы описаны в клинических алгоритмах далеко не полностью.

Такое положение имеет серьезные последствия, негативно отражающиеся на безопасности пациентов.

ВРАЧЕБНЫЕ ОШИБКИ И БЕЗОПАСНОСТЬ ПАЦИЕНТОВ

Врачебные ошибки опасны тем, что могут привести к смерти, стойкой инвалидности пациентов или причинить иной вред. До последнего времени статистика появления медицинских ошибок не была известна. Ситуация изменилась в 1999 и 2000 году, когда был опубликован 300-страничный доклад Национальной академии медицины США (Institute of Medicine, National Academy of Medicine) под названием «Человеку свойственно ошибаться: создание более безопасной системы здравоохранения» [20].

На основании тщательных исследований было установлено, что медицинские ошибки в больницах США являются причиной смерти от 44 000 до 98 000 человек в год [20, pp. 1, 26, 31]. Это значит, что «в американских больницах каждые полгода погибает больше американцев, чем за всю Вьетнамскую войну» [21].

Авторы доклада признают, что врачебные ошибки занимают одно из ведущих мест в структуре смертности населения США. Даже если взять нижнюю оценку (44 000 человек), смертность из-за врачебных ошибок превышает значение восьмой ведущей причины смерти в США. По вине врачей умирает больше людей, чем от дорожно-транспортных происшествий (43 458 жертв), от рака молочной железы (42 297 жертв), от СПИДа (16 516 жертв) [20, pp. 1, 26].

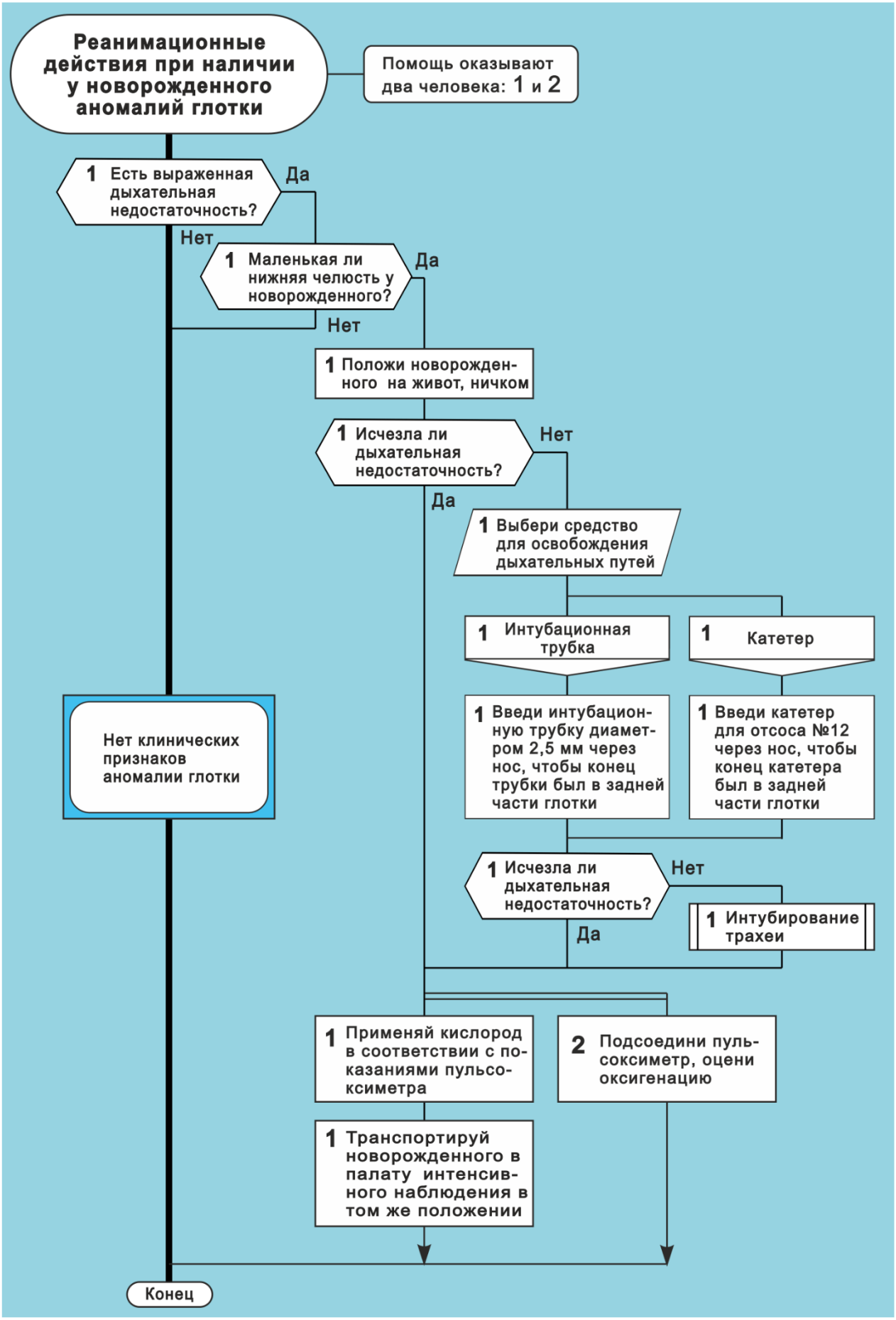


Рис. 8. Клинический алгоритм «Реанимационные действия при наличии у новорожденного аномалий глотки» [22]

По рекомендациям Академии медицины в конгрессе США были проведены слушания и принят закон о безопасности пациентов (Patient Safety and Quality Improvement Act of 2005), подписанный президентом Джорджем Бушем мл. 29 июля 2005 года. Таким образом, понятие безопасности пациентов приобрело не только научный, но и юридический статус.

Доклады Национальной академии медицины США [20] [23] [24] [25] впервые поставили в центр научного исследования исключительно трудную междисциплинарную проблему — проблему врачебных ошибок. В докладах предложена развернутая программа противодействия этому злу в интересах безопасности пациентов.

К сожалению, Академия медицины США не сумела предложить исчерпывающего решения проблемы; она лишь обосновала необходимость научного подхода и предложила чрезвычайно важные (хотя и недостаточные) меры по ее решению, преимущественно организационного характера.

КРИТИКА ВЫВОДОВ АКАДЕМИИ МЕДИЦИНЫ США

Врачебные ошибки зависят от многих причин, в том числе, от недостатков профессионального медицинского языка, который, будучи естественным языком, не приспособлен для точного и удобного описания клинических алгоритмов и не имеет необходимых для этого специальных средств.

Клинические алгоритмы профилактики, диагностики, лечения, скорой помощи, реанимации, реабилитации, прогноза являются научной проблемой первостепенной важности, которая имеет прямое отношение к предотвращению врачебных ошибок и безопасности пациентов. Однако эта проблема была полностью упущена из виду в указанных докладах, что, несмотря на несомненные достоинства и социально значимые позитивные последствия глобального характера, снижает ценность полученных выводов и рекомендаций [4] с. 23, 24.

НОВАЯ ПОСТАНОВКА ВОПРОСА

Согласно тезису Ланды-Непейводы, клинические алгоритмы не являются строгими алгоритмами; это всего лишь суррогаты алгоритмов, то есть жизнеритмы (алгоритмические предписания), которые имеют ограниченную точность (определенность).

Задача состоит в том, чтобы кардинально повысить точность (определенность) клинических алгоритмов и в максимально возможной степени приблизить их свойства к идеалу – к свойствам математических алгоритмов. Это принципиально новая постановка проблемы [4] с. 65.

Математический алгоритм содержит в себе все указания, необходимые для его выполнения. В отличие от него *клинический* алгоритм содержит лишь часть указаний, а остальные опускает, рассчитывая на знания и сообразительность профессиональных врачей.

Проблема неопределенности в медицинской литературе является актуальной, поскольку она неблагоприятно отражается на здоровье населения и личного состава Вооруженных Сил.

Для решения задачи вводится понятие «клинический алгоритм высокой точности» (ДРАКОН-алгоритм), чтобы существенно повысить качество клинических алгоритмов, уменьшить степень неопределенности, снизить вероятность врачебных ошибок и обеспечить безопасность пациентов [4] с. 65.

ФУНДАМЕНТАЛЬНЫЙ НЕДОСТАТОК ПРОФЕССИОНАЛЬНОГО МЕДИЦИНСКОГО ЯЗЫКА

Профессиональный медицинский язык (язык медицинской литературы, учебников, стандартов, руководств, клинических рекомендаций, клинических протоколов) имеет серьезный дефект. Он недостаточно точен и плохо приспособлен для описания сложных и разветвленных медицинских действий и решений, выполняемых при профилактике, диагностике, лечении.

Чтобы устранить дефект, нужно осуществить глубокую реформу медицинского языка, расширив его возможности с помощью визуального медицинского алгоритмического языка — языка ДРАКОН. Последний предназначен для стимулирования клинического мышления врачей, повышения безопасности пациентов, предотвращения врачебных ошибок и стандартизации представления клинических алгоритмов в медицинской литературе [4] с. 283, [26] [27] [28].

МЕДИЦИНСКИЙ АЛГОРИТМИЧЕСКИЙ ЯЗЫК ВЫСОКОЙ ТОЧНОСТИ

Язык ДРАКОН выполняет две существенно разные функции:

- Описывая математически строгие алгоритмы, он функционирует как алгоритмический язык программирования.
- Описывая клинические алгоритмы (жизнеритмы), ДРАКОН значительно улучшает их качество, превращая их в жизнеритмы высокой точности, открывая путь к алгоритмической медицине.

Согласно развиваемой идее, прежние (неточные, не полностью определенные) описания клинических алгоритмов должны со временем отмереть, навсегда сойти со сцены. И уступить место принципиально новому типу алгоритмов — *клиническим алгоритмам высокой точности*. Последние, разумеется, остаются жизнеритмами (алгоритмическими предписаниями). Принципиальное отличие состоит в том, что жизнеритмы *высокой точности* приближаются по своим свойствам (в пределах возможного) к настоящим алгоритмам.

В качестве медицинского алгоритмического языка высокой точности предлагается использовать язык ДРАКОН, который имеет существенные преимущества по сравнению с конкурирующими средствами.

АПРОБАЦИЯ

Литовские врачи первыми обратили внимание на возможность крупномасштабного использования языка ДРАКОН в медицине [29] [30]. За последнее время в Литве изданы четыре медицинских учебника на русском языке, в которых используется ДРАКОН:

1. Начальная неотложная акушерская помощь [31].
2. Специализированная реанимация новорожденного [22].
3. Неотложная медицинская помощь [32].
4. Травма [33].

Учебники апробированы в ряде стран (Литва, Казахстан, Азербайджан, Таджикистан, Туркменистан, Киргизия) в рамках курсов повышения квалификации врачей. Курсы проводили высококвалифицированные специалисты из Литовского университета медицинских наук (Lietuvos sveikatos mokslų universitetas) для местных врачей. Проведенная апробация дала положительные результаты [34].

ВОЗМОЖНА ЛИ ФОРМАЛИЗАЦИЯ МЕДИЦИНЫ?

Современный этап развития медицины можно охарактеризовать как «очень непростой и болезненный процесс ломки прежних взглядов, в ходе которого прежняя неформальная парадигма медицинского мышления постепенно уступает место новой, более строгой алгоритмической парадигме» [35].

Трудность в том, что подавляющее большинство клинических алгоритмов сегодня описаны в виде словесного текста, который принципиально не пригоден для записи безошибочных алгоритмов. Отсюда следует, что «все виды медицинской литературы, в которых трактуются, разъясняются или описываются клинические алгоритмы, устарели. Это означает, что в той или иной степени устарели и нуждаются в совершенствовании:

- медицинские учебники,
- медицинские стандарты,
- медицинские руководства,
- клинические рекомендации,
- протоколы диагностики и лечения и пр.

Суть в том, что устарело прежнее понимание и прежние способы записи клинических алгоритмов. Они должны уступить место клиническим алгоритмам высокой точности» [4] с. 308.

ОТЗЫВЫ ВРАЧЕЙ О ЯЗЫКЕ ДРАКОНЕ

Доктор медицинских наук А. Кудрявичене, неонатолог:

> «Язык ДРАКОН – отличный инструмент для обучения практическим навыкам и их стандартизации. Он позволяет выявить все, даже мельчайшие, но очень важные действия» [4] с. 316.

Доктор медицинских наук , профессор М. Ключинскас, акушер-гинеколог:

> «Язык ДРАКОН позволяет систематизировать процессы с минимальным применением текста – как при организации работы, так и при выполнении медицинских процедур. Он помогает всем одинаково понимать и выполнять конкретные действия… Позволяет ускорить запоминание действий» [4].

Доктор медицинских наук, профессор Ж. Дамбраускас, абдоминальный хирург:

> «Огромным преимуществом языка ДРАКОН является то, что он позволяет конкретно выявить все этапы процедуры или процесса… Мысленно можешь повторить процесс этап за этапом, а затем каждый этап разделить на шаги… Процедуру или процесс можно выполнить мысленно, а затем и в реальности. ДРАКОН является инструментом мысленной тренировки» [4].

Врач Б. Кумпайтене, анестезиолог-реаниматолог:

> «Польза языка ДРАКОН для разрабатывающего алгоритм автора состоит в том, что проявляется, кристаллизуется и стандартизируется каждый навык, каждая процедура. Польза для обучающегося – это ясный путь выполнения действий. ДРАКОН дает ответ на вопросы “что делать, если”» [4], с. 316.

Сотрудник медицинского Центра исследования кризисов А. Вилейките:

> «Применение языка ДРАКОН позволяет стандартизировать и эргономично представить самую сложную процедуру… Если всё правильно описано на ДРАКОНе, значит, всё будет отлично выполнено» [4], с. 316.

Доктор медицинских наук, профессор Динас Вайткайтис, зав. кафедрой экстремальной медицины:

> «Язык ДРАКОН даёт ясность и чёткость процессам, применяемым в медицине. Он позволяет “автоматизировать” обучение студентов практическим навыкам. Может стать основой для технологии принятия клинических решений» [4], с. 314.

Доктор медицинских наук П. Добожинскас, исполнительный директор медицинского Центра исследования кризисов:

> «Применение языка ДРАКОН действенно помогает в создании и описании сложных, динамичных решений медицинских проблем. Тем самым значительно облегчается проведение стандартизированного симуляционного обучения, внедряя культуру безопасности пациентов и принципы качественного оказания медицинских услуг в масштабах медицинского учреждения, региона или государства» [4], с. 314.

Доктор медицинских наук, профессор Р.Й. Надишаускене, зав. клиники акушерства и гинекологии:

> «Алгоритмизация медицины подразумевает значительную перестройку системы медицинского образования и перевод ее на алгоритмический путь… Накопленный в Литве практический положительный опыт использования языка ДРАКОН для представления сложных и разнообразных медицинских алгоритмов может послужить серьезной основой для принятия крупных структурных решений руководителями здравоохранения и системы медицинского образования в области алгоритмизации медицины» [4], с. 317.

РОЛЬ МЕДИЦИНСКОЙ СЛУЖБЫ МИНИСТЕРСТВА ОБОРОНЫ РФ

Медицинская служба Министерства обороны РФ располагает высококвалифицированными кадрами медицинских специалистов и ученых и способна решать масштабные задачи. Одной из таких задач является алгоритмизация военной медицины с помощью медицинского алгоритмического языка ДРАКОН.

Часть 3. АЛГОРИТМЫ И ПРОГРАММИРОВАНИЕ

ТРУДНЫЕ ВОПРОСЫ И ПРЕДЛАГАЕМЫЕ РЕШЕНИЯ

Современная теория алгоритмов не имеет удобного (эргономичного) языка, позволяющего облегчить и ускорить понимание алгоритмов ЧЕЛОВЕКОМ. Она не применима к клиническим алгоритмам и не содействует повышению безопасности пациентов. Она не оказывает практической помощи при разработке бизнес-процессов, потоков работ (workflows) и пр.

Современные языки программирования используют управляющие слова *(if, then, else, case, of, switch, break, while, do, repeat, until, for, foreach, continue, loop, exit, when, last и др.)*, которые играют роль визуальных помех, провоцируют появление ошибок и мешают понять смысл алгоритма в терминах предметной области. Чтобы устранить недостаток, управляющие слова целесообразно заменить на *управляющую графику* языка ДРАКОН. Ниже показано, как это можно сделать на примере языка Си.

На рис. 9 слева показана программа на языке Си. Справа изображена математически эквивалентная ей программа на языке Дракон-Си.

В чем сходство этих программ?

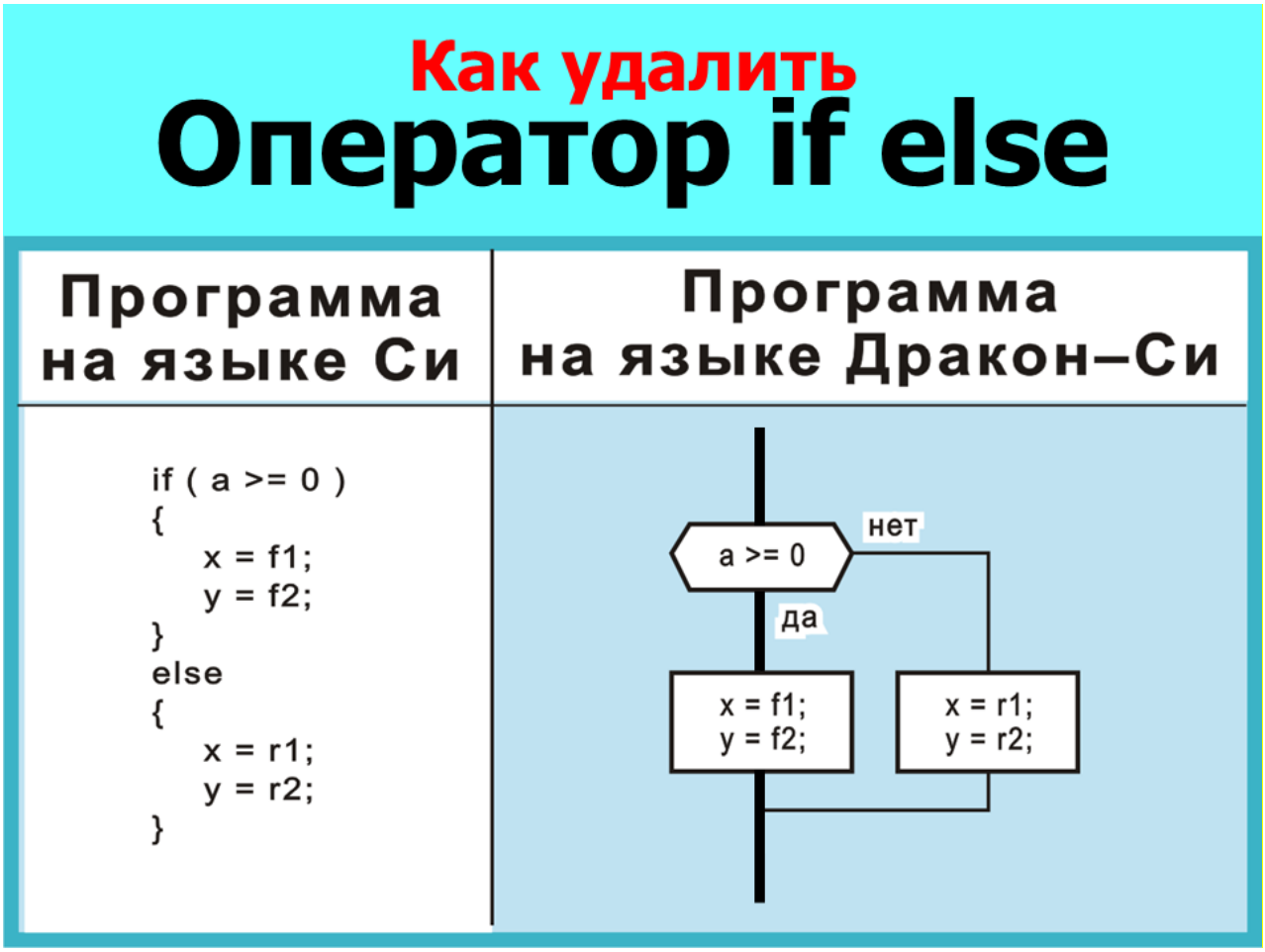


Рис. 9. Текстовый оператор *if else* можно удалить и заменить на управляющую графику языка ДРАКОН

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

- a>=0 (а больше нуля)

На языке Дракон-Си (на рис. 9 справа) это выражение помещено в иконе Вопрос, которая снабжена выходами «Да» и «Нет».

Еще пример:

- x=f1;
- y=f2;

В правой графе 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 чем разница?

- 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.

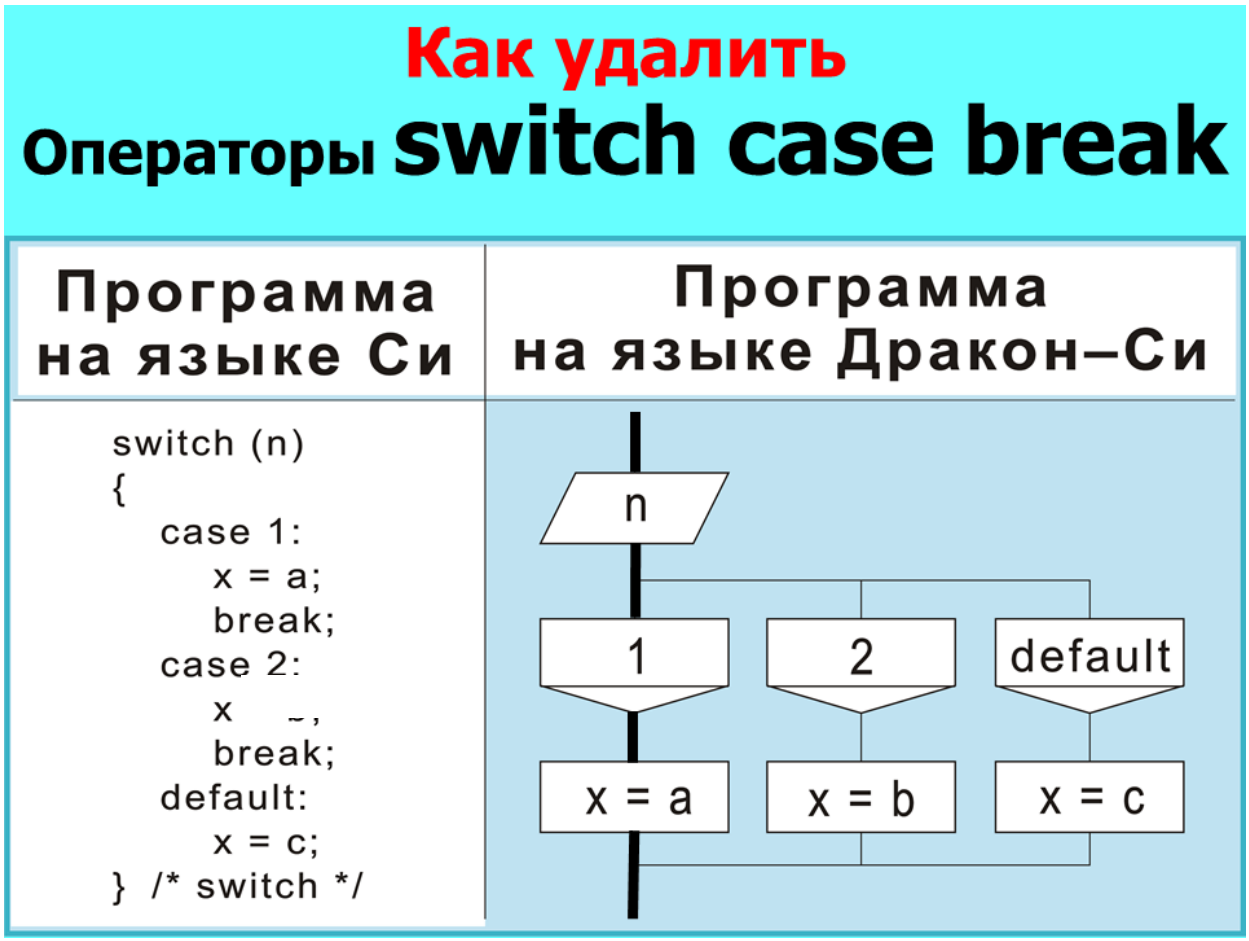


Рис. 10. Текстовые операторы *switch, case, break* можно удалить и заменить на управляющую графику ДРАКОНа

Для рис. 10 ниже представлен список выражений, которые не изменяются. В неизменном виде они переходят из Си-программы в Дракон-программу (в иконы Выбор, Вариант и Действие):

- n
- 1
- 2
- default
- x=a
- x=b
- x=c

В Си-программе видим большое количество избыточных слов и текстовых символов:

- switch,
- case 1,
- case 2,
- break,
- две фигурные скобки,
- две круглые скобки,
- три двоеточия,
- пять точек с запятой,
- две звездочки.

Результат сравнения

- Нужны ли все эти символы? Нет, не нужны!
- В Дракон-программе эти знаки исчезают. Они превращаются в приятный для глаза графический образ, пригодный для быстрого симультанного (панорамного) восприятия.

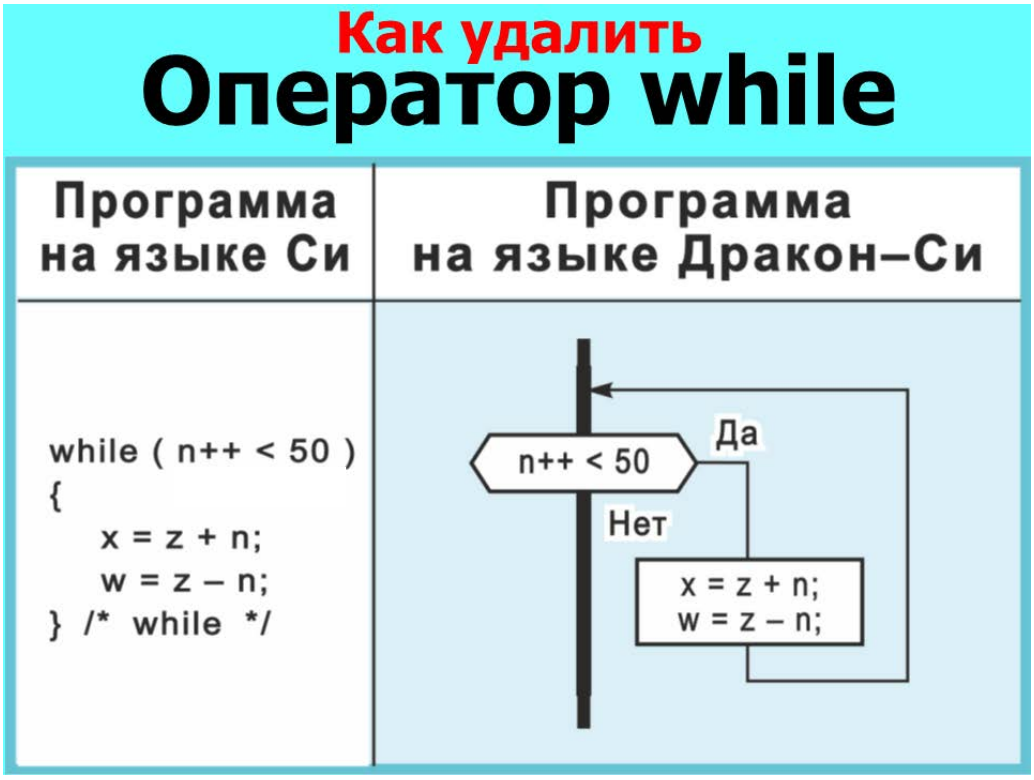


Рис. 11. Текстовый оператор *while* можно удалить и заменить на управляющую графику ДРАКОНа

Глядя на рис. 11, перечислим выражения, которые без изменений перемещаются из Си-программы в Дракон-программу (в иконы Вопрос и Действие):

- n++<50
- x=z+n;
- x=z–n;

В Си-программе имеются избыточные элементы:

- while;
- две круглые скобки;
- две фигурные скобки;
- две косые скобки;
- две звездочки.

Итог

В Дракон-программе все избыточные «паразитные» знаки отсутствуют, превращаясь в линии чертежа. Чертеж наглядно показывает маршруты алгоритма и петлю обратной связи цикла.

Рис. 12. Текстовый оператор *do while* можно удалить и заменить на управляющую графику ДРАКОНа

Проведем анализ рис. 12 и укажем выражения, которые без изменения переходят из Си-программы в Дракон-программу (в иконы Действие и Вопрос):

- y=p–a;
- y=p+a;
- n-->b

Как и в предыдущих случаях, в Си-программе имеются избыточные элементы:

- do;
- while (два раза);
- две круглые скобки;
- две фигурные скобки;
- две звездочки.

Итог

В Дракон-программе избыточные элементы не нужны; вместо них используются линии и фигуры. Чертеж гораздо лучше, чем текст, показывает пространственную структуру программы и петлю обратной связи цикла.

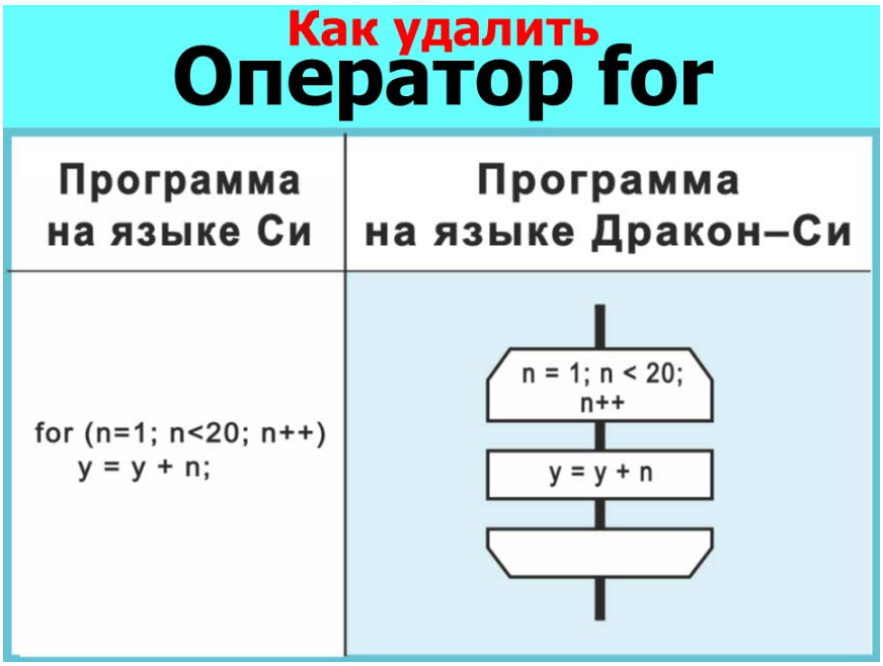


Рис. 13. Текстовый оператор *for* можно удалить и заменить на управляющую графику

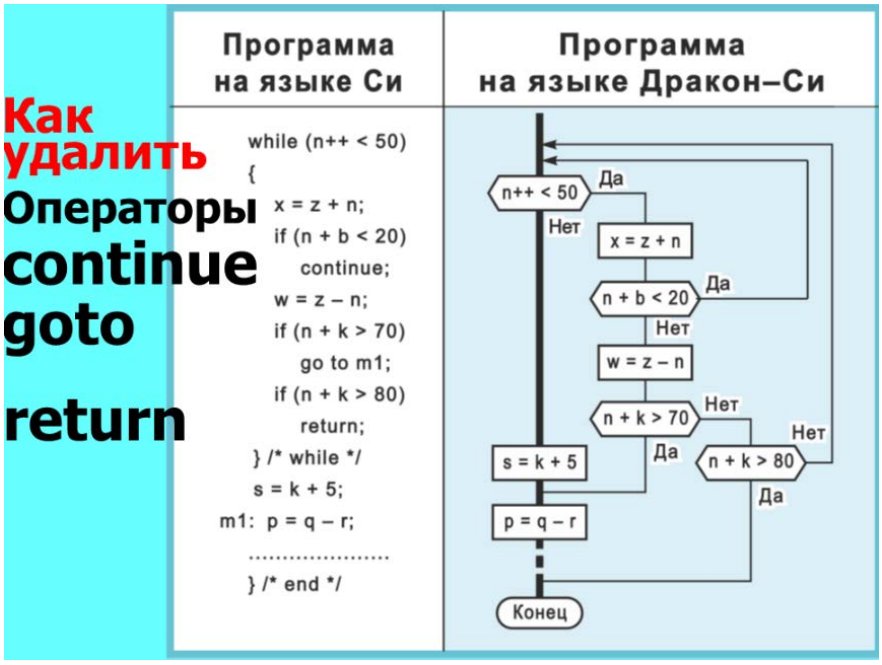


Рис. 14. Текстовые операторы *continue, goto, return* можно удалить и заменить на управляющую графику

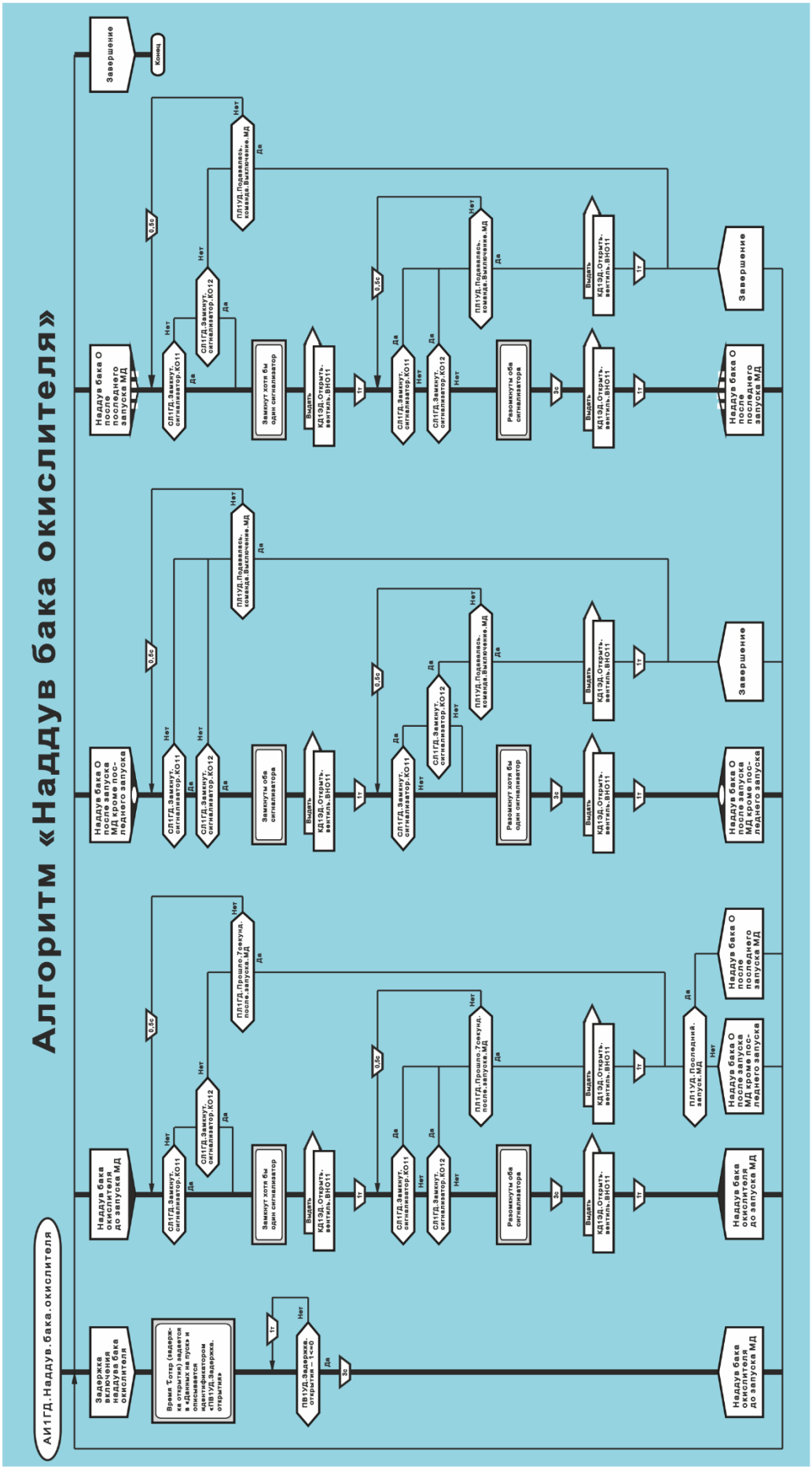


Рис. 15. Алгоритм «Наддув бака окислителя» пневмогидросистемы подачи топлива в Маршевый двигатель разгонного блока космических аппаратов

ДРАКОН УПРОЩАЕТ ПРОГРАММИРОВАНИЕ

Современные процедурные (императивные) и функциональные языки программирования относятся к классу *текстовых* языков. В этих языках для управления ходом выполнения программы широко используются зарезервированные (ключевые) управляющие слова: if, then, else, case, of, switch, break, while, do, repeat, until, for, foreach, continue, loop, exit, when, last и др. Мы убедились, что указанные слова могут быть безболезненно удалены и заменены на простейшую графику. Это означает частичный переход от текстового программирования к визуальному.

При таком переходе из текста программы удаляются не только управляющие слова, но и многие знаки пунктуации: фигурные скобки, круглые скобки, косые скобки, двоеточия, точки с запятой, звездочки.

Перечисленные управляющие слова и знаки препинания являются обязательными в текстовом программировании, где без них обойтись невозможно.

В визуальном программировании, напротив, они становятся лишними, избыточными, ненужными. По этой причине они рассматриваются как «паразитные» элементы и визуальные помехи, которые загромождают программу ненужными подробностями, провоцируют появление ошибок и мешают понять смысл алгоритма в терминах предметной области.

В языке ДРАКОН широко используются средства повышения надежности и безошибочности программ, например, визуальное логическое исчисление, позволяющее автоматически из визуальных аксиом методом визуального логического вывода получать дракон-схемы. Указанные средства подробно описаны в литературе в разделе «Теоретические основы языка ДРАКОН» [3], с. 425–472.

Выводы

- Язык ДРАКОН устраняет многие управляющие слова и предлагает метод безошибочного визуального программирования, который повышает надежность программ, что очень важно для военно-промышленных задач.
- Теоретические основы языка ДРАКОН являются дополнением к теории алгоритмов, предлагая метод визуального исчисления икон и эргономизации алгоритмов, позволяющие повысить безошибочность алгоритмов, облегчить и ускорить понимание алгоритмов ЧЕЛОВЕКОМ

Часть 4. СРАВНИТЕЛЬНЫЙ АНАЛИЗ БЛОК-СХЕМ АЛГОРИТМОВ ПО ГОСТ 19.701—90 И ДРАКОН-АЛГОРИТМОВ

ДРАКОН-СХЕМА — ЭТО УПОРЯДОЧЕННАЯ БЛОК-СХЕМА

Блок-схемы алгоритмов по стандарту ISO 5807:85 и ГОСТ 19.701—90 не позволяют изображать сложные алгоритмы с необходимой полнотой и наглядностью. С ростом сложности графические схемы алгоритмов быстро теряют наглядность, линии начинают пересекаться и сплетаются в невразумительный клубок. Чтобы сохранить удобочитаемость, приходится упрощать и сокращать чертеж. Полноценный алгоритм превращается в усеченный, неточный, приблизительный вариант.

Чтобы устранить дефект, нужно упорядочить блок-схемы. Упорядоченные блок-схемы называются «дракон-схемы» (drakon-charts). Последние подчиняются строгим формальным правилам и правилам эргономичных алгоритмов.

В дракон-схемах запрещено пересечение линий, которое путает читателей и затрудняет понимание алгоритма, устранены эргономические недочёты. Дракон-схемы позволяют ликвидировать или существенно ослабить недостатки блок-схем.

КОГНИТИВНО-ЭРГОНОМИЧЕСКИЙ АНАЛИЗ БЛОК-СХЕМ АЛГОРИТМОВ

Ниже рассмотрены примеры блок-схем алгоритмов, взятые из технической литературы. Проведен когнитивно-эргономический анализ блок-схем и выявлены ошибки эргономического характера.

Для каждой блок-схемы рядом с ней в качестве образца изображена дракон-схема, решающая ту же задачу. Показано, что ошибки, присущие блок-схемам, в дракон-схемах отсутствуют — они выявлены и устранены.

Стандарт ГОСТ 19.701-90 не может обеспечить наглядность при вычерчивании сложных алгоритмов. Это объясняется тем, что концепция стандарта отстала от жизни и построена без учета идей когнитивной эргономики.

УДОБОЧИТАЕМОСТЬ АЛГОРИТМОВ

Алгоритм, представленный в виде графической схемы, предназначен для зрительного восприятия человеком. Следовательно, алгоритм представляет собой зрительную сцену. Или, если угодно — зрительный образ, зрительную картину.

Эргономичность алгоритма — это эстетическая привлекательность его зрительного образа, в частности, блок-схемы или дракон-схемы.

Мы помним, что алгоритм можно назвать эргономичным, если процесс понимания алгоритма протекает быстро и облегчает выявление ошибок.

Чем эргономичнее алгоритм, тем быстрее и легче можно его понять. Отсюда вытекает, что удобочитаемость и элегантность алгоритмов открывают путь к экономии умственных усилий. Но не только.

Чем эргономичнее созданы зрительные образы частей алгоритма, чем изящнее они соединены в общую алгоритмическую картину, тем приятнее на них смотреть. Чем точнее и быстрее зрительный «пейзаж» обнажает глубинный смысл алгоритма, тем плодотворнее мышление.

Чем больше эргономичность, тем глубже понимание алгоритмов. Тем скорее течет наша алгоритмическая мысль. Тем легче мы постигаем суть дела. Тем быстрее и качественнее протекает важнейший производственный процесс, играющий немалую роль в мировой экономике, — процесс массовой разработки алгоритмов и программ.

И наоборот, если зрительный образ алгоритма кажется запутанным и некрасивым, процесс понимания, обдумывания и поиска ошибок неизбежно замедляется, что снижает производительность умственного труда.

ДРАКОН — графический язык, язык зрительных образов. С учетом сказанного можно уточнить: ДРАКОН — язык эргономичных зрительных образов. Но эргономичность ДРАКОНа не самоцель. Она позволяет ощутимо повысить производительность труда при создании алгоритмов.

Я исхожу из того, что зрительные образы алгоритмов следует сознательно проектировать. Для этой цели можно (в разумных пределах) использовать *средства художественного конструирования.*

Уместно напомнить слова видного психолога Б. Ф. Ломова, основателя Института психологии Академии наук СССР:

> «Средства художественного конструирования в конечном счете направлены на то, чтобы вызвать тот или иной эффект у работающего человека…
>
> Применяя средства художественного конструирования, мы создаем положительные эмоции, облегчаем операцию приема информации человеком, улучшаем концентрацию и переключение внимания, повышаем скорость и точность действий.
>
> Короче говоря, мы пользуемся этими средствами для управления поведением человека в широком смысле слова, для управления его психическим состоянием» и умственной работоспособностью [36].

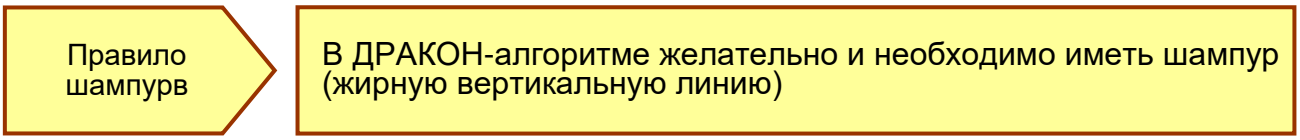
ЭРГОНОМИЧНОСТЬ — ЭТО НАБОР ПРАВИЛ

Наша ближайшая цель — разъяснить, что эргономичность есть набор правил, которым должны подчиняться зрительные образы, представленные на бумаге или экране. Возьмем за основу зрительные образы языка ДРАКОН, построенные из икон и их комбинаций.

Мы рассмотрим десяток правил ДРАКОНа и на конкретных примерах покажем, что в блок-схемах правила систематически нарушаются, а в дракон-схемах — строго соблюдаются.

ПРАВИЛО ШАМПУРА

На первое место следует поставить правило шампура:



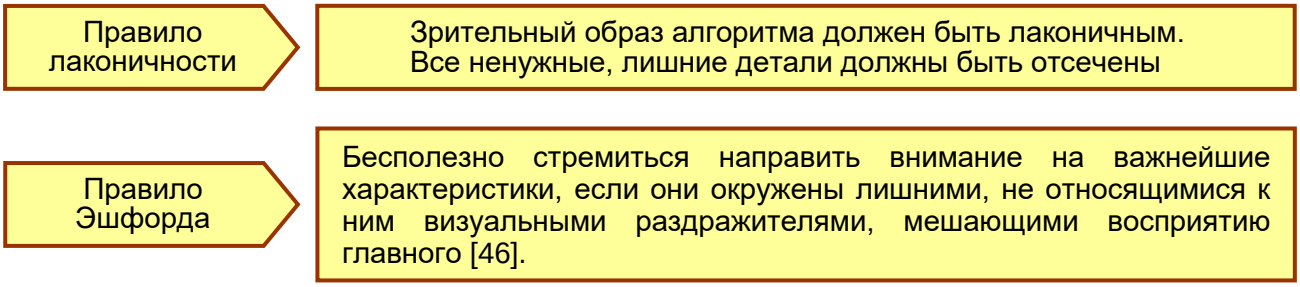
. Шампур создает систему отсчета, в которой проектируется графическая схема алгоритма.

Ниже на многих примерах будут продемонстрированы типичные ошибки блок-схем:

- нет шампура,
- разрыв шампура.

СХЕМА ДОЛЖНА БЫТЬ ЛАКОНИЧНОЙ

Схема алгоритма должна содержать лишь те элементы, которые необходимы для сообщения читателю существенной информации, точного понимания ее смысла и стимулирования правильных решений и разумных действий. Пустые украшения, избыточные, затемняющие детали должны быть удалены из схемы.



СЛЕДУЕТ ИЗБЕГАТЬ НЕОПРАВДАННЫХ ИЗГИБОВ СОЕДИНИТЕЛЬНЫХ ЛИНИЙ

Существуют вредные мелочи, которые затрудняют понимание схемы. Одна из них — неоправданные изгибы соединительных линий.

Сравним две схемы на рис. 16 и 17. На рис. 16 показана обычная блок-схема, заимствованная из книги сотрудников IBM [37]. На рис. 17 изображена эквивалентная дракон-схема.

Рисунки позволяют выявить различия между неприглядной блок-схемой и красивой дракон-схемой. С точки зрения правил удобочитаемости, блок-схема на рис. 16 имеет следующие недостатки:

- Неоправданно большое число изгибов линий (в блок-схеме 12 изгибов, а в дракон-схеме только 4).
- Большое число паразитных элементов: 14 стрелок и 3 кружка, которые в дракон-схеме отсутствуют (поскольку они совершенно не нужны и представляют собой визуальные помехи, затемняющие суть дела).

Правило минимизации изгибов

- Чтобы алгоритм был удобным для чтения, количество изгибов соединительных линий должно быть минимальным.
- Из двух схем лучше та, где число изгибов меньше

Мы рассмотрели два эргономических правила (правило лаконичности и правило минимизации изгибов). И убедились, что в дракон-схеме они строго соблюдаются, а в блок-схеме грубо нарушены.

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ДВУХ СХЕМ

Обратимся снова к рис. 16 и 17. Мы сделали лишь первый шаг к устранению графических недочетов. На рис. 16 осталось еще немало огрехов, которые необходимо выявить и исправить.

Итак, продолжим наш критический анализ. Блок-схема на рис. 16 имеет следующие недостатки.

- Для обозначения развилки используется ромб, который занимает слишком много места. Ромб не позволяет поместить внутри необходимое количество удобочитаемого текста, состоящего из строк равной длины. В дракон-схеме верхний и нижний углы ромба отрезаны. Поэтому схема становится компактной и удобной как для записи текста, так и для чтения.
- Функционально однородные иконы *Д, Е, Ж, И* хаотично разбросаны по всей площади чертежа, занимая три разных горизонтальных уровня (что путает читателя). В дракон-схеме они расположены на *одном* уровне, что служит для читателя подсказкой об их функциональной однородности.
- Ромбы имеют выход влево, что разрушает шампур и не позволяет применить правило главного маршрута. В дракон-схеме выход влево не допускается.
- Икона *Д* и ее вертикаль расположены слева от шампура (в дракон-схеме это запрещено).
- Ниже икон *Ж и И* находятся три уровня горизонтальных линий, которые имеют паразитный характер. В дракон-схеме три уровня сведены в одну линию, что делает схему более наглядной и компактной.

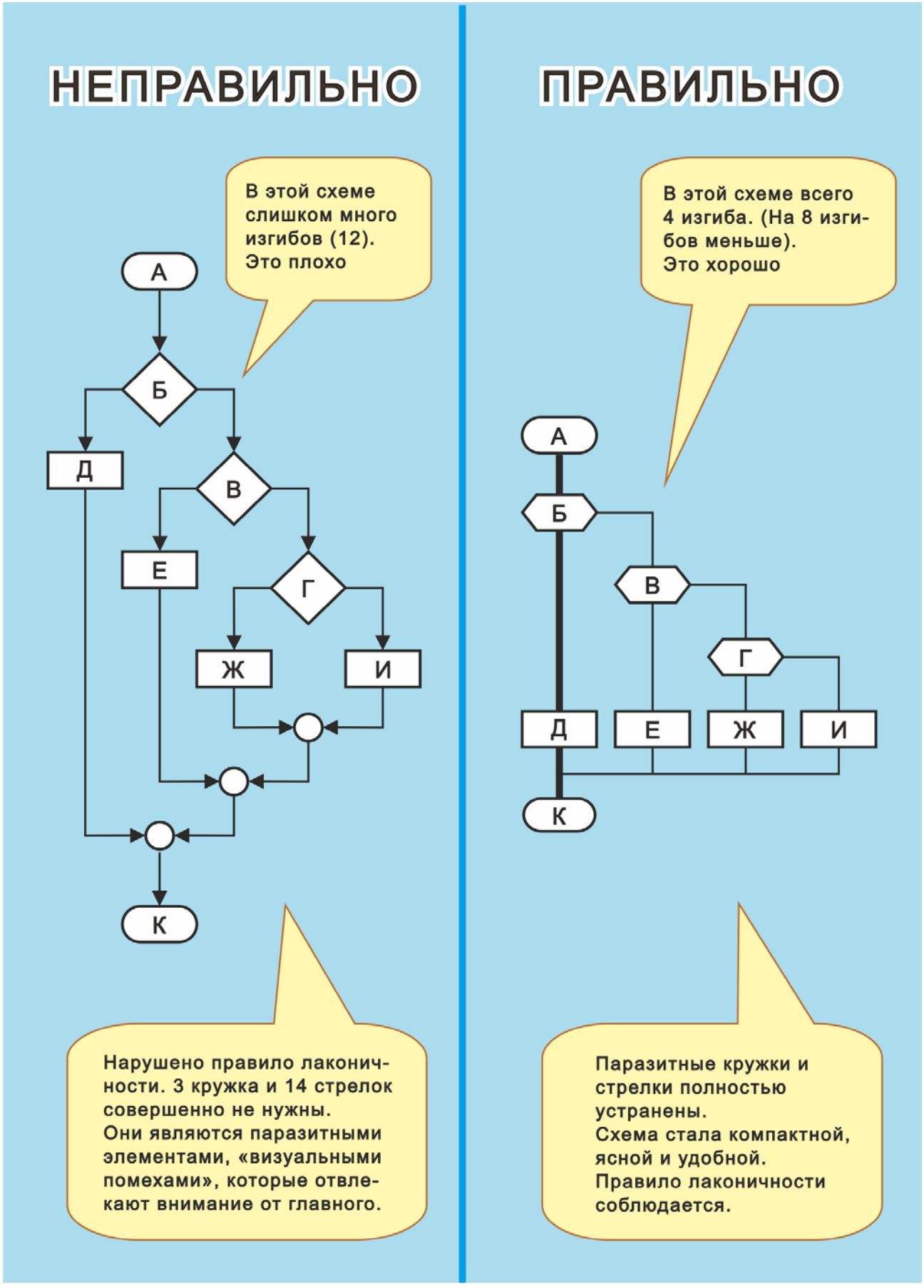


Рис. 16. Плохая схема. Недостатки: слишком много изгибов; имеются паразитные элементы

Рис. 17. Хорошая (эргономичная) схема. Она нарисована по правилам языка ДРАКОН

Да, конечно, каждое из этих улучшений является незначительным и не делает погоды. Но когда мелкие улучшения повторяются многократно и становятся массовыми, ситуация может измениться. Количество переходит в качество. В этом случае облегчение умственного труда может стать значительным.

КРИТИКА БЛОК-СХЕМ АЛГОРИТМОВ В ТЕХНИЧЕСКОЙ ЛИТЕРАТУРЕ

Цивилизация не может жить без чертежей, не может жить и без алгоритмов. Схемы алгоритмов очень важны для понимания секретов и тайн современного производства и управления. Однако многие преподаватели и разработчики алгоритмов создают неприглядные и путаные блок-схемы, в которых трудно разобраться. Иногда их даже называют «мусорными» блок-схемами, потому что хитросплетения блоков, соединенные хаосом петляющих линий, больше напоминают кучу мусора, нежели регулярную структуру.

Образчик подобного мусора представлен на рис. 18 [38], с. 102. Этот «мусорный» алгоритм можно вылечить и превратить в изящную дракон-схему (рис. 19). Сравним что было и что стало.

Схема на рис. 18 имеет много изъянов.

- Слева от иконы *Ж* есть пересечение линий (в дракон-схеме пересечения запрещены).
- Возле иконы *Е* имеется линия под углом 45° (в дракон-схеме наклонные линии не допускаются).
- Иконы *Д, Е* и *Ж* имеют более одного входа (в дракон-схеме это запрещено).
- Иконы *В, Д, Е, Ж* имеют входы сбоку, что придает схеме неряшливый вид. В дракон-схеме вход разрешается только сверху, что упорядочивает алгоритм и создает в нем четкую ориентацию «сверху вниз»).
- Отсутствует шампур, так как выход иконы Заголовок и вход иконы Конец не лежат на одной вертикали. Исчезновение шампура означает, что в схеме отсутствует зрительный остов, художественно-композиционная главная вертикаль. Тем самым уничтожается основа для выделения главного маршрута.

Блок-схема на рис. 18, как и предыдущая (рис. 16), по всем параметрам проигрывает дракон-схеме. Мы убедились, что алгоритмическая красота достигается благодаря совокупному действию многих правил, каждое из которых, взятое по отдельности, выглядит скромным и будничным.

РАЗРЫВ ШАМПУРА — СЕРЬЕЗНАЯ ОШИБКА

Разорванный шампур искажает зрительный образ схемы и может повлечь за собой неприятности. Между тем, такая схема не только не осуждалась, а наоборот, в свое время была рекомендована стандартом *ANSI* (Американский национальный институт стандартов).

На рис. 20 представлена схема, взятая из источника [39], где она характеризуется как «стандартная блок-схема *ANSI*».

Сравнение этой схемы с эквивалентной дракон-схемой на рис. 21 позволяет выявить различные дефекты:

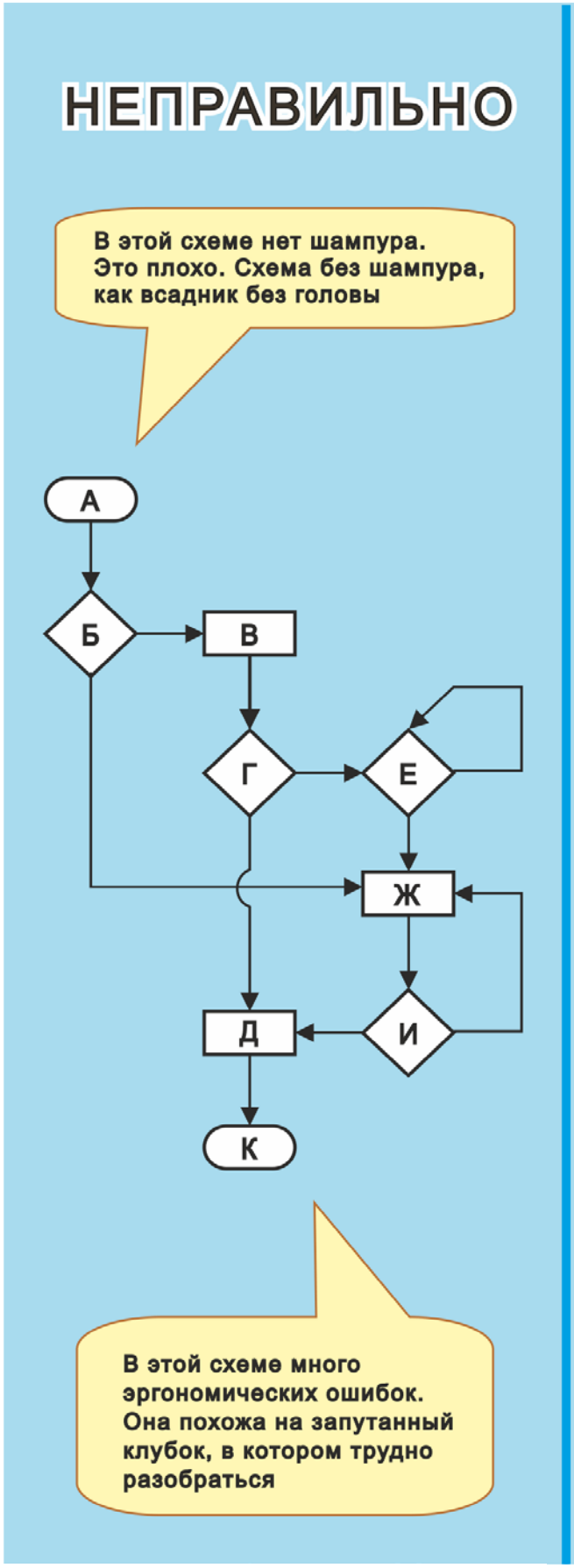


Рис. 18. Плохая схема. Такие схемы часто рисуют многие уважаемые ученые, забывающие об эргономике

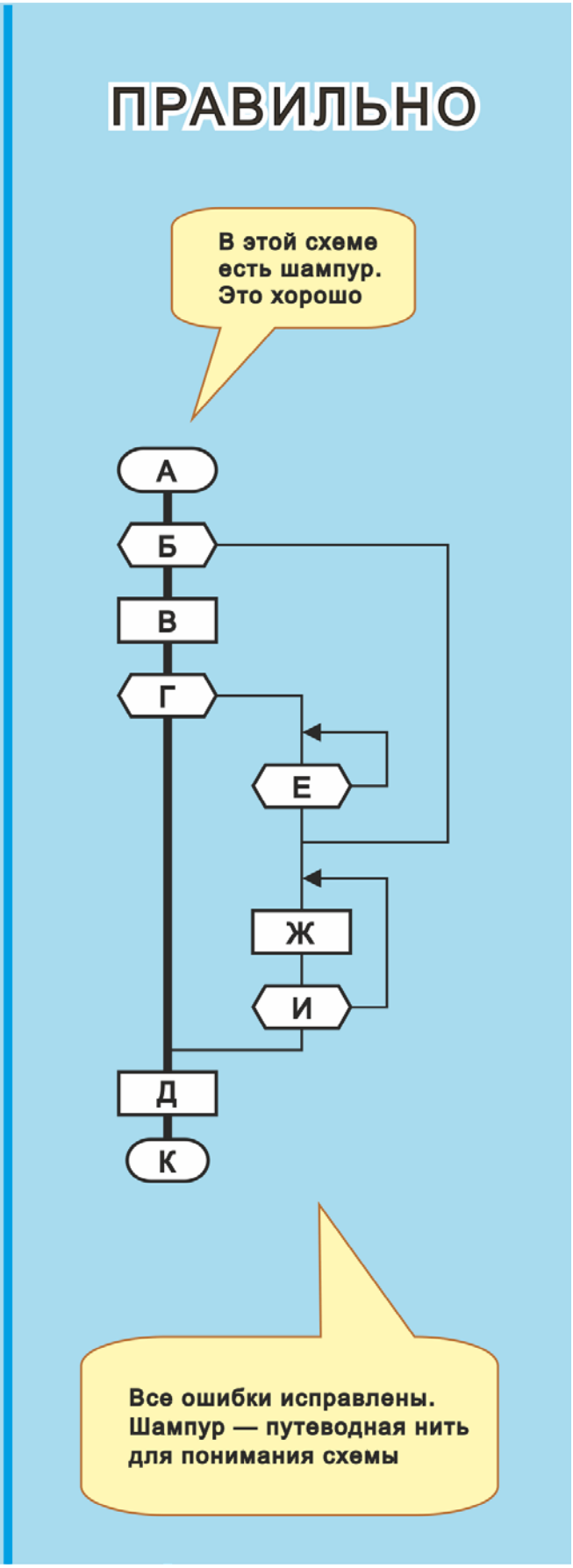


Рис. 19. Хорошая (эргономичная) схема. Она нарисована по правилам языка ДРАКОН

- Ниже иконы *G* имеет место разрыв шампура (нарушено правило, согласно которому один из путей, идущих от входа к выходу, должен проходить по главной вертикали).
- Икона *G* имеет два входа (в дракон-схеме разрешается только один вход).
- Икона *G* имеет вход сбоку (в дракон-схеме это запрещено).
- У иконы *G* выход находится слева (в дракон-схеме он должен быть снизу).
- Две петли обратной связи обычного цикла находятся слева от шампура и закручены по часовой стрелке (в дракон-схеме они расположены справа от шампура и закручены против часовой стрелки).
- Используются неудобные ромбы (в дракон-схеме их заменяют эргономичной иконой Вопрос).
- Ромб *L* имеет выход слева (в дракон-схеме он должен быть справа).
- Используются 12 стрелок, из которых 10 — паразитные (в дракон-схеме всего 2 стрелки).
- Имеется один избыточный изгиб линии (в блок-схеме 9 изгибов, в дракон-схеме только 8).

Таким образом, данная блок-схема, как и предыдущие примеры, проигрывает дракон-схеме.

АНАЛИЗ ВЛОЖЕННОГО ЦИКЛА «ПОКА»

Графическое изображение цикла должно быть удобным, стандартным и легко запоминающимся. В блок-схемах эта проблема не решена: нередко каждый автор изображает цикл по-своему, что путает читателей. Язык ДРАКОН предлагает стандартное начертание для каждого типа цикла.

На рис. 20 и 21 изображены блок-схема и дракон-схема вложенного цикла ПОКА (while). Блок-схема взята из учебного пособия [40].

На блок-схеме 20 можно указать следующие недочеты:

- Между иконами *E* и *K* имеет место разрыв шампура (нарушено правило, согласно которому один из путей, идущих от входа к выходу, должен проходить по главной вертикали).
- Ниже иконы *E* через разрыв шампура проходят две нежелательные горизонтальные линии
- Две петли обратной связи циклов ПОКА закручены по часовой стрелке (в дракон-схеме они закручены против часовой стрелки).
- Используются неудобные ромбы (в дракон-схеме их заменяют эргономичные иконы Вопрос).
- Используются 8 стрелок, из которых 6 — паразитные (в дракон-схеме всего 2 стрелки).
- Имеется два избыточных изгиба линии (в блок-схеме 10 изгибов, в дракон-схеме только 8).
- В блок-схемах отсутствует графическая стандартизация циклов. В дракон-схемах стандартизация циклов строго соблюдается. Об этом можно судить по рис. 23. Голубая заливка окружает внутренний цикл ПОКА. Белая заливка окружает внешний цикл ПОКА.

Сравнивая рис. 20 и 21 (а также 22 и 23) легко убедиться, что дракон-схемы несопоставимо лучше, чем блок-схемы.

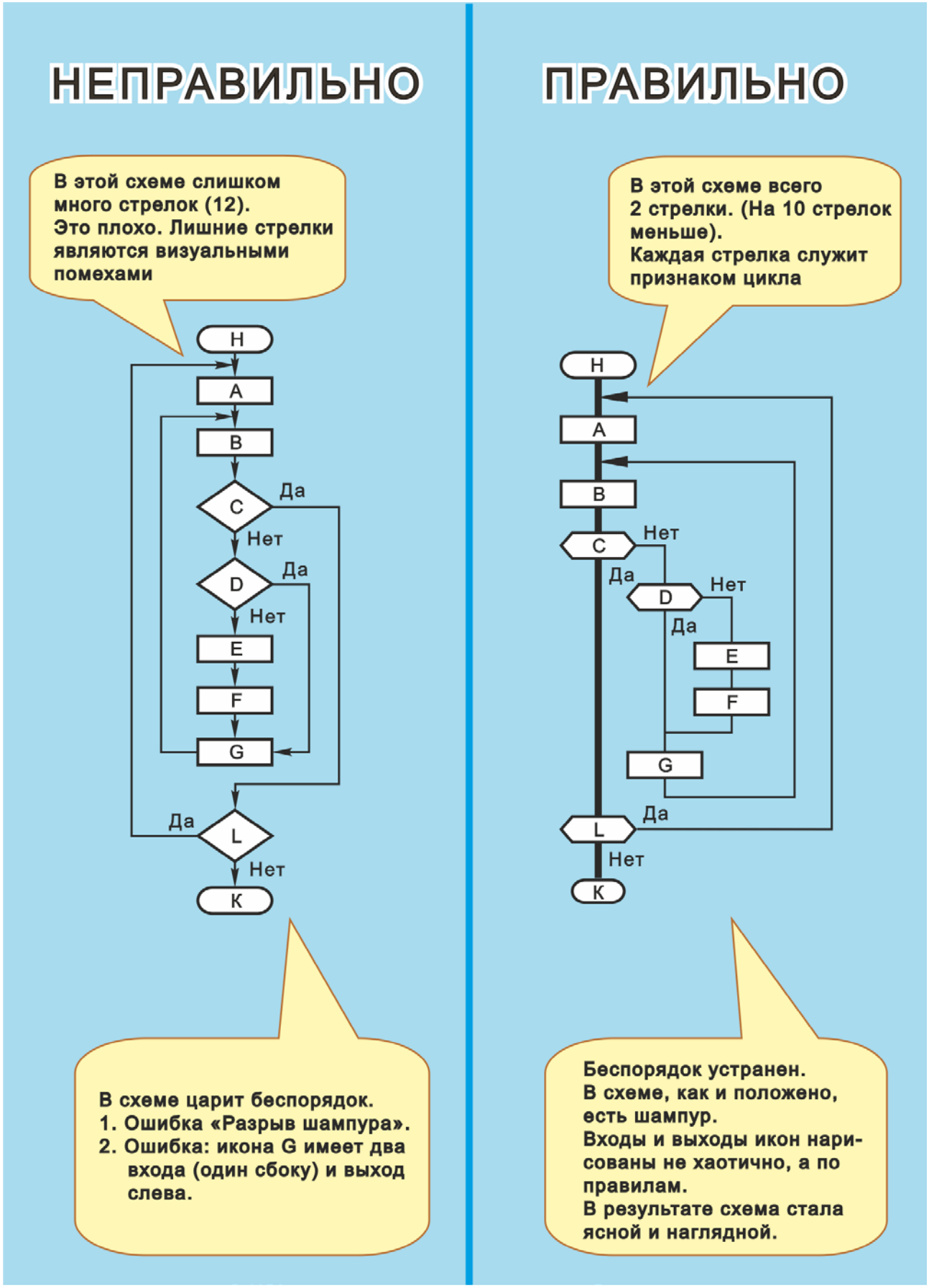


Рис. 20. Плохая схема. В ней много эргономических ошибок, что затрудняет понимание алгоритма

Рис. 21. Хорошая (эргономичная) схема. Она нарисована по правилам языка ДРАКОН

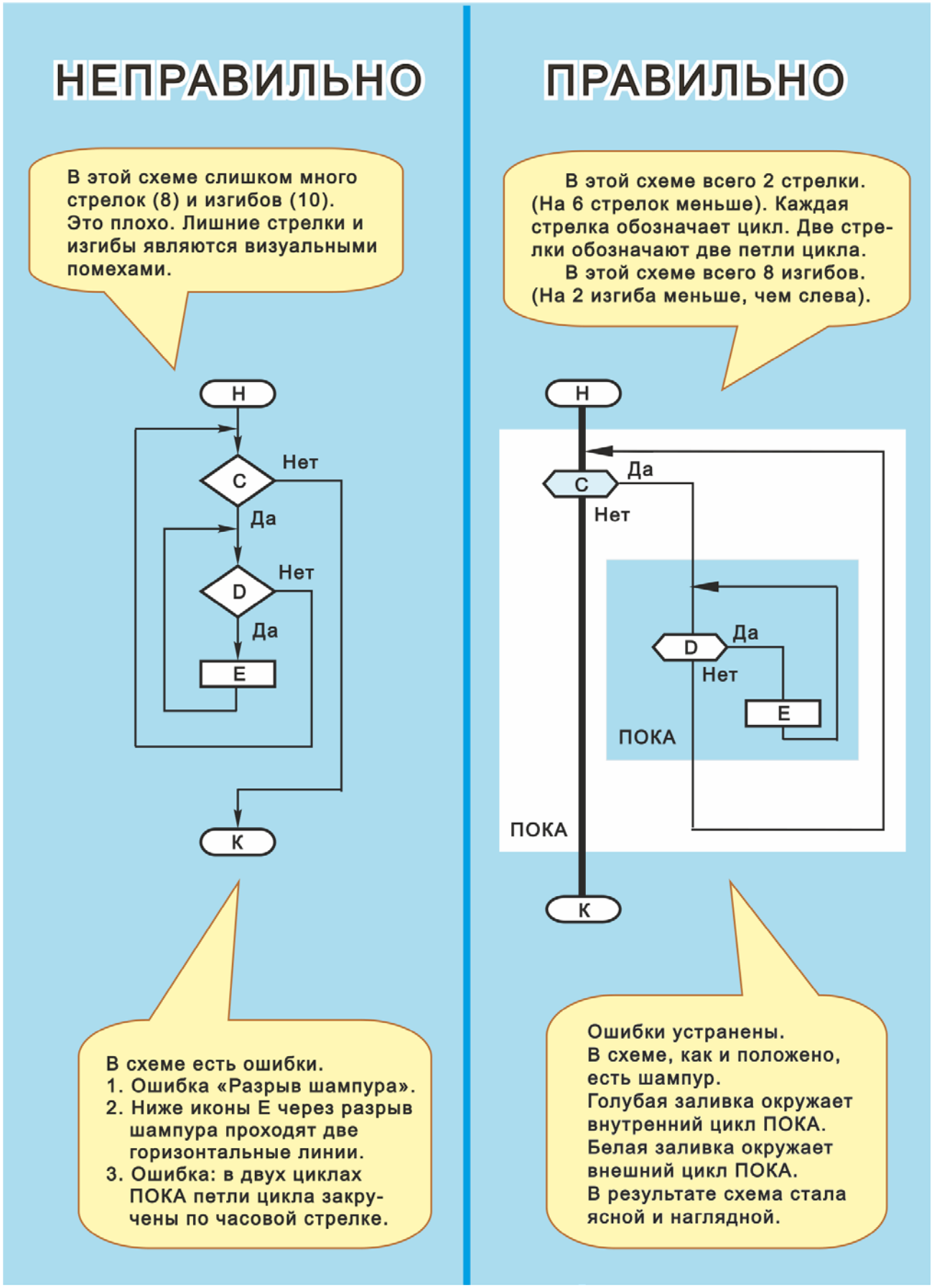


Рис. 22. Плохая схема. Вложенный цикл ПОКА изображен без учета правил эргономики. Шампур разорван.

Рис. 23. Хорошая (эргономичная) схема. Вложенный цикл ПОКА нарисован по правилам языка ДРАКОН

НЕЭРГОНОМИЧНЫЕ «ОБРАЗЦЫ ИТОГОВЫХ ЗАДАНИЙ»

В журнале «Информатика и образование» опубликованы рекомендуемые «образцы итоговых заданий по оценке качества подготовки школьников по информатике» [41].

В материале приведены четыре блок-схемы с блоком «Решение», которые препятствуют использованию шампура. Причина эргономической ошибки состоит в том, что нижний выход ромба удален и нарисован слева (рис. 24). Ошибку можно легко исправить, как показано в дракон-схеме на рис. 25.

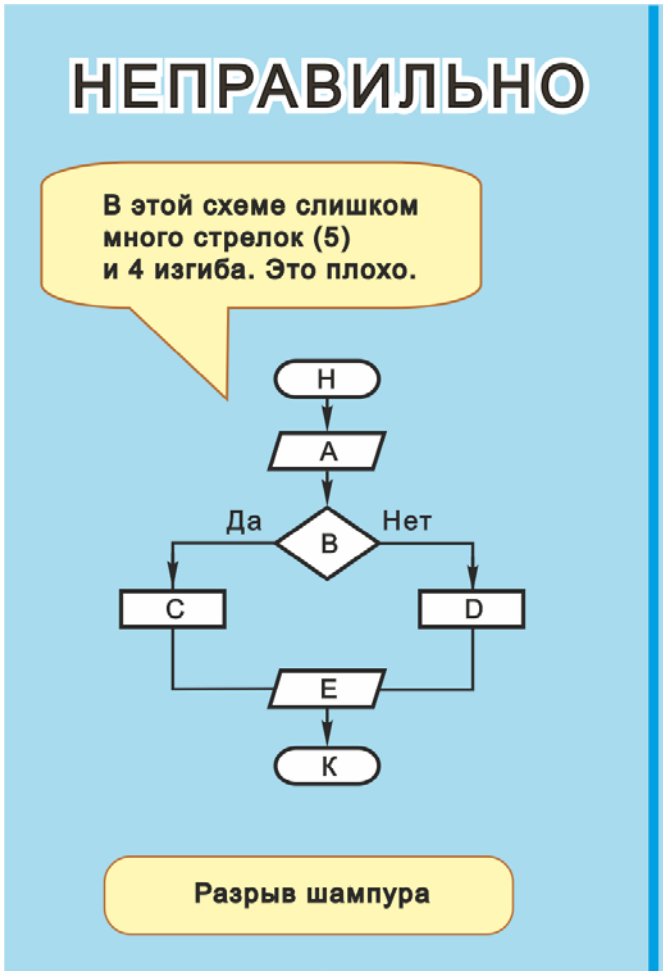


Рис. 24. Плохая схема. Выход из ромба влево (а не вниз) мешает правильному изображению шампура

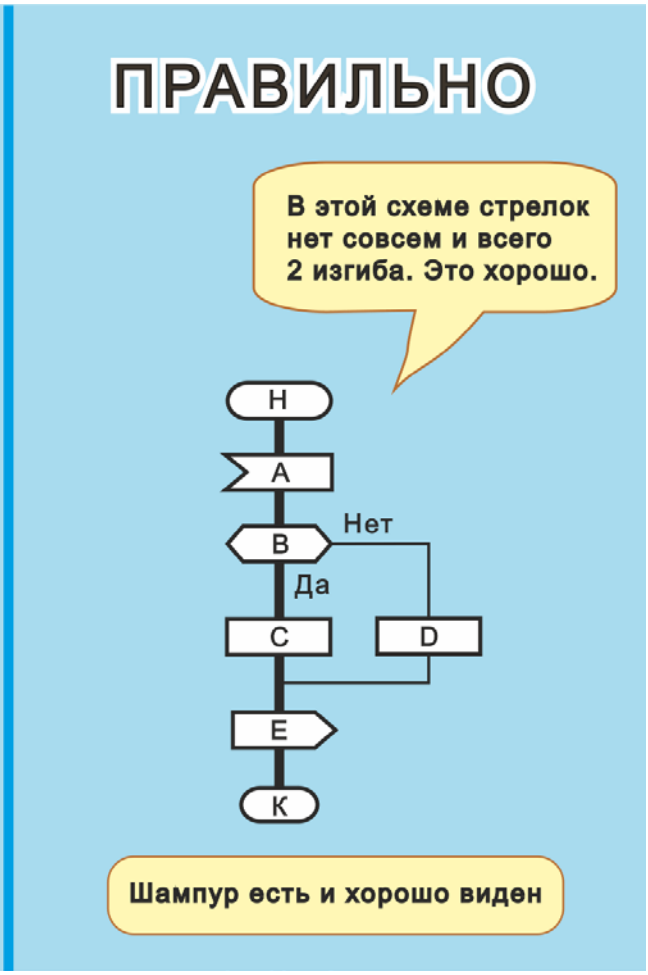


Рис. 25. Хорошая (эргономичная) схема. Шампур нарисован верно, так как икона Вопрос имеет выход вниз.

ТИПИЧНЫЕ ОШИБКИ В БЛОК-СХЕМАХ АЛГОРИТМОВ

На основании анализа алгоритмов на рис. 16 – 25 можно составить перечень эргономических ошибок, которые мы выявили в этой главе:

- нет шампура;
- разрыв шампура;
- блоки Заголовок и Конец на разных вертикалях;
- ненужные стрелки;
- ненужные изгибы линий;

- ненужные кружки;
- два входа в один блок;
- три входа в один блок;
- вход в блок слева;
- вход в блок справа;
- пересечение линий;
- наклонная линия;
- выход из блока Процесс слева.

ПРИМИТИВ И СИЛУЭТ

В данной главе мы рассмотрели простые блок-схемы, состоящие примерно из 10 блоков. Их можно «вылечить» с помощью дракон-схемы примитив.

В сложных проектах могут использоваться большие многостраничные блок-схемы, насчитывающие 100 и более блоков. В таких случаях работать с блок-схемой становится трудно, поскольку блок-схема полностью теряет и наглядность, и регулярность структуры. Чтобы поправить дело, нужно использовать язык ДРАКОН и алгоритмическую конструкцию силуэт, которая обеспечивает наглядность даже для многостраничных схем.

ПРЕИМУЩЕСТВА ДРАКОН-СХЕМ

В отличие от блок-схем алгоритмов, выполненных по ГОСТ 19.701—90, упорядоченные дракон-схемы пригодны для формализованной записи и автоматического получения исполняемого кода.

Упорядоченные дракон-схемы специально сконструированы таким образом, чтобы превратить сложный алгоритм в удобную схему, обеспечивающую быстрое и лёгкое понимание. По мнению специалистов, благодаря использованию дракон-схем алгоритмы становятся более понятными, доходчивыми, ясными, прозрачными.

Эргономичные методы, применяемые в дракон-схемах, существенно улучшают восприятие алгоритмов. Язык усовершенствованных графических схем ДРАКОН обеспечивает разработку сложных алгоритмов «с сохранением наглядности даже для многостраничных схем» [42].

БЛОК-СХЕМЫ НЕ ИМЕЮТ БУДУЩЕГО

ДРАКОН выгодно отличается от блок-схем тем, что его графический узор подчиняется жестким и тщательно продуманным правилам, которые дисциплинируют мышление, облегчают умственный труд [3], с. 60.

Важным дефектом блок-схем является недостаток выразительных средств. Алгоритмическая структура силуэт, являющаяся основным и наиболее мощным инструментом языка ДРАКОН, принципиально не может быть выражена на языке блок-схем в удобном для чтения виде.

Изложенные в литературе теоретические обоснования, сравнительный анализ дракон-схем и блок-схем, а также многочисленные примеры убедительно доказывают, что блок-схемы алгоритмов не имеют будущего [4], с. 197.

ВЫВОДЫ

1. Стандарт ГОСТ 19.701-90 не может обеспечить наглядность при вычерчивании сложных алгоритмов, так как концепция стандарта устарела и построена без учета идей когнитивной эргономики.
2. Дракон-схемы принципиально отличаются от блок-схем тем, что подчиняются когнитивно-эргономическим правилам.
    - В данной главе указаны эргономические недостатки блок-схем и описаны парные им достоинства дракон-схем.
    - В зрительно-смысловой структуре алгоритма шампур играет важную роль. Шампур способен быстро и естественно привлечь к себе внимание читателя, правильно сориентировать его в структуре алгоритма.
    - При отсутствии шампура уничтожается основа для выделения главного маршрута.
    - Разрыв или отсутствие шампура делает зрительный образ алгоритма «бесформенным», лишенным композиционного центра. Читатель лишается необходимых зрительных ориентиров, что затрудняет чтение алгоритма.
    - Зрительный образ алгоритма должен быть лаконичным. Все ненужные, лишние детали должны быть исключены.
    - Схема должна содержать лишь те элементы, которые необходимы читателю для понимания смысла алгоритма и выявления ошибок.
3. Чтобы схема была удобной для чтения, количество изгибов соединительных линий должно быть минимальным.
    - Стрелки нужны только как признак цикла. Стрелки, показывающие направление потока управления, следует удалить.
    - Дракон-схемы созданы на основе блок-схем с целью их совершенствования. Дракон-схемы — это упорядоченные блок-схемы.

Часть 5. КРИТИЧЕСКИЙ АНАЛИЗ БЛОК-СХЕМ АЛГОРИТМОВ ПО ГОСТ 19.701-90

ФОРМАЛЬНЫЕ СРЕДСТВА ПРЕДОТВРАЩЕНИЯ ОШИБОК В ВИЗУАЛЬНОМ ЯЗЫКЕ ДРАКОН

Графический синтаксис языка ДРАКОН содержит средства и правила, специально предназначенные для обеспечения безошибочности. Сюда, в частности, относятся:

- визуальное логическое исчисление;
- теория отростков;
- теория валентных точек;
- теория макроикон;
- теория стрелок.

Теоретической основой языка ДРАКОН служит визуальное логическое исчисление, для краткости *исчисление икон.* Используются две визуальные аксиомы: аксиома-силуэт и аксиома-примитив (рис. 26), из которых методом визуального логического вывода выводится графика любого ДРАКОН-алгоритма [2], глава 32; [3], глава 33.

Наличие указанных средств создает принципиальное отличие графического синтаксиса языка ДРАКОН от графического синтаксиса других языков.

СРАВНЕНИЕ СО СТАНДАРТОМ ГОСТ 19.701-90

Сравним с блок-схемами алгоритмов по ГОСТ 19.701-90 и ISO 5807:85. В блок-схемах формализованы только фигуры. За пределами формализации остаются:

- правила присоединения отростков линий к фигурам;
- точки размещения фигур, или точки ввода фигур;
- связи между фигурами.

Отростки, точки и связи остаются произвольными, что вызывает справедливые нарекания пользователей. Это значит, что в стандартах ГОСТ 19.701-90 и ISO 5807:85 отсутствует должная формализация.

В отличие от блок-схем, в дракон-схемах проведена полная формализация. Строго определены не только иконы, но их отростки, а также соединительные линии между ними и точки ввода фигур. Формализацию соединительных линий обеспечивают формальные отростки икон, валентные точки и макроиконы.

Правила работы с отростками, валентными точками и макроиконами описаны в работах [2] и [3].

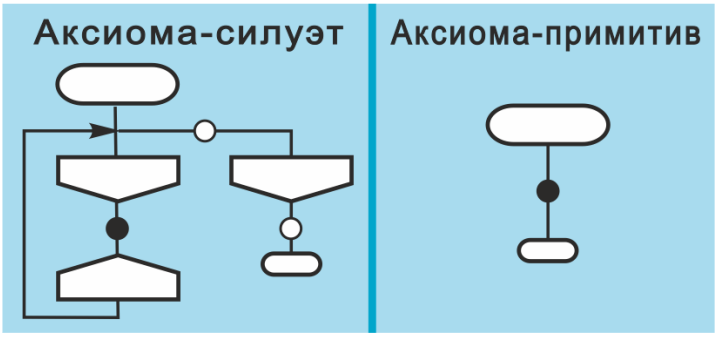


Рис. 26. Аксиома-силуэт и аксиома-примитив.
Черными и белыми кружками отмечены валентные точки

ТЕОРИЯ ОТРОСТКОВ

Назначение теории отростков — устранить неоднозначность присоединения отростков к графическим фигурам (символам), которая характерна для стандарта на алгоритмы ГОСТ 19.701-90 и является источником ошибок.

В языке ДРАКОН формализация отростков выполнена так:

- иконы заданы не отдельно, а вместе с отростками соединительных линий;
- число отростков, тип и направление каждого отростка строго определены.

На рис. 27 показаны две иконы. Икона Действие имеет два отростка, а икона Вопрос — три. Видно, что отростки неотделимы от икон.

Существуют три типа отростков:

- входной,
- выходной,
- нейтральный.

Входной и выходной отростки иконы направлены сверху вниз и лежат на одной вертикали, причем входной отросток входит в икону сверху, а выходной — исходит из нее снизу. Икона Вопрос имеет не один, а два выхода; второй выходной отросток направлен по горизонтали вправо. Все отростки ориентированы к центру иконы.

На рис. 27 представлены входные и выходные отростки. Нейтральный отросток приведен на рис. 8. Это горизонтальный отросток, соединяющий икону Заголовок «Реанимационные действия…» с иконой Пояснение.

Перечисленные правила задают однозначную привязку отростков к иконам и являются средством формализации.

СРАВНИВАЕМ СО СТАНДАРТОМ ГОСТ 19.701-90

Иконам Действие и Вопрос (рис. 27) в стандарте ГОСТ 19.701-90 соответствуют блоки «Процесс» и «Решение» (рис. 28).

Легко видеть, что формализация отростков в ГОСТе отсутствует. Действительно, в стандарте для блока Процесс (рис. 28) предусмотрены четыре варианта. Больше того, число вариантов может возрасти вдвое, если учесть, что стандарт разрешает выполнять чертежи как со стрелками, так и без них:

> «При необходимости или для повышения удобочитаемости могут быть добавлены стрелки-указатели» [43].

Для сравнения: в языке ДРАКОН для иконы Действие (рис. 27) мы видим всего один чертеж, что исключает путаницу и предотвращает ошибки.

Далее. Для блока Решение (рис. 28) ГОСТ разрешает четыре варианта (или даже восемь — с использованием стрелок и без них).

Сравним с языком ДРАКОН и иконой Вопрос (рис. 27). Для нее задан один-единственный чертеж, что обеспечивает строгую формализацию.

Неоднозначность графики стандарта является прямым следствием указания ГОСТа: «Символы могут быть вычерчены в любой ориентации» [43], с.12.

Наличие в стандарте разных способов подключения отростков к блокам говорит об отсутствии формализации соединительных линий в блок-схемах алгоритмов, что является недостатком стандарта на алгоритмы ГОСТ 19.701-90.

Рис. 27. Язык ДРАКОН предлагает однозначный (единственный) вариант присоединения отростков к иконам. Икона Действие имеет два отростка: входной (сверху) и выходной (снизу). Икона Вопрос имеет три отростка: входной (сверху) и два выходных (внизу и справа).

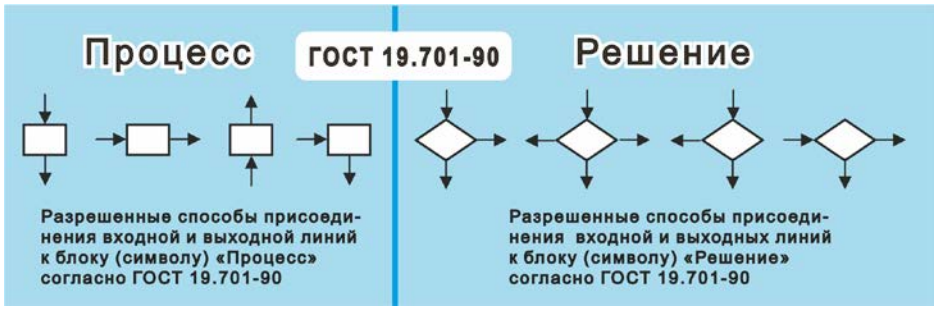


Рис. 28. ГОСТ 19.701-90 разрешает присоединять линии к блокам Процесс и Решение четырьмя разными способами, т. е. неоднозначно. Язык ДРАКОН устраняет недостаток и предлагает однозначный (единственный) вариант

ТЕОРИЯ ВАЛЕНТНЫХ ТОЧЕК

Данная теория служит для формализации точек размещения икон.

Точки размещения икон (точки ввода икон) называются *валентными точками.* Расположение валентных точек на чертеже дракон-алгоритма строго определено. Иконы можно вставлять только в валентных точках, и больше нигде. На рис. 29 показаны валентные точки, находящиеся на отростках икон Действие и Вопрос.

На чертеже дракон-алгоритма валентные точки не изображаются, но подразумеваются. Они визуализируются (на короткое время) только в процессе построения дракон-схемы — при работе инструментальной программы ДРАКОН-конструктор.

Рис. 29. Икона Действие имеет две валентные точки, а икона Вопрос — три.

ДИНАМИКА ВАЛЕНТНЫХ ТОЧЕК

Вспомним, что чертеж дракон-алгоритма формируется методом логического вывода из аксиом. На каждом шаге построения происходит размножение валентных точек. Процесс размножения можно рассматривать как динамический процесс преобразования валентных точек.

Рассмотрим построение дракон-схемы силуэт (рис. 31). В аксиоме-силуэт (рис. 26, слева) всего 3 валентных точки. После добавления к аксиоме ветки силуэта (рис. 31, в центре) получается уже 5 точек. А после вставки иконы Действие (рис. 31, справа) число точек увеличивается до 6. Дальнейший процесс построения силуэта приводит к монотонному росту числа валентных точек.

Это значит, что иконы (или атомы) можно вставлять не куда угодно, а только в строго определенные места. Для каждой валентной точки определен список икон (атомов), которые разрешается вставить в данную конкретную точку.

Ввод атома производится так. Сначала происходит разрыв соединительной линии в выбранной пользователем валентной точке. Затем в место разрыва вставляется атом.

Все списки (списки валентных точек и списки разрешенных икон) хранятся в памяти программы «ДРАКОН-конструктор», который осуществляет визуальный логический вывод. Таким образом, описанная операция строго формализована и защищена от многих ошибок.

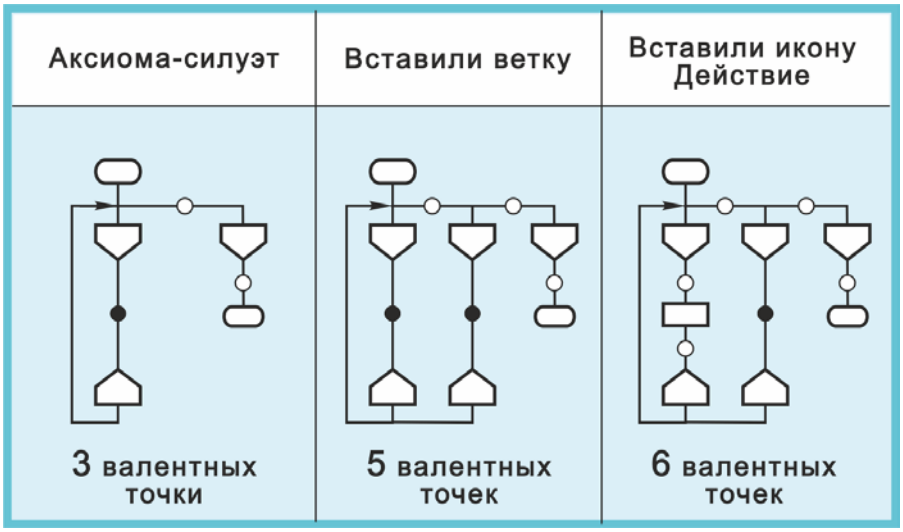


Рис. 30. Размножение валентных точек при проектировании дракон-схемы силуэт

АТОМЫ

Атом есть графическая фигура, имеющая один вход сверху и один выход снизу, лежащие на одной вертикали. Атомы показаны на рис. 31. Атом можно вставить в валентную точку с помощью операции «ввод атома». Почти все макроиконы являются атомами.

КРИТИЧЕСКИЕ И НЕЙТРАЛЬНЫЕ ВАЛЕНТНЫЕ ТОЧКИ

Валентные точки делятся на нейтральные и критические. Точка называется нейтральной, если операция «ввод атома» в данную точку является возможной, но не обязательной. В отличие от нее критическая точка требует обязательного ввода атома.

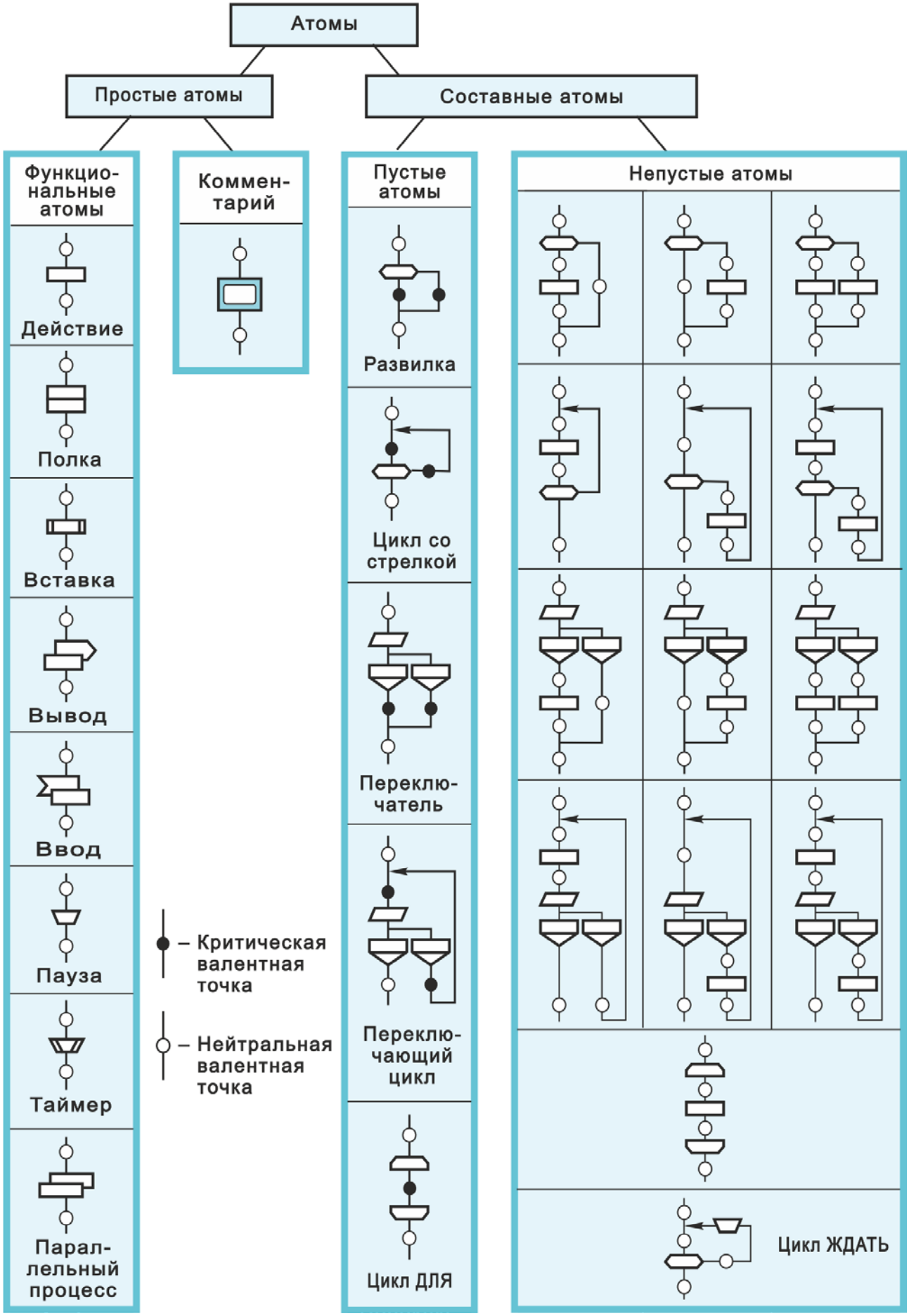


Рис. 31. Атомы и валентные точки

Валентные точки находятся в аксиомах и атомах. Они показаны на рис. 31, где нейтральные точки обозначены белыми кружками, а критические – черными. Если в фигуре (аксиоме или атоме) всего одна критическая точка, ввод атома обязательно производится именно в нее; при этом критическая точка уничтожается.

Если фигура имеет две критические точки, обязательный ввод атома делается только в одну из них. Та критическая точка, в которую произведен ввод, уничтожается, а другая — нейтрализуется, т. е. становится нейтральной.

Полная совокупность критических точек охватывает:

- критические точки в пустых атомах;
- одну критическую точку в аксиоме-силуэт;
- одну критическую точку в аксиоме-примитив.

Полная совокупность нейтральных точек охватывает:

- входные и выходные точки атомов;
- две внутренние точки в атоме «цикл ЖДАТЬ»;
- две точки в аксиоме-силуэт;
- точки, полученные в результате нейтрализации критических точек.

Правило критической точки

- В законченной дракон-схеме не должно быть критических точек.
- Наличие критической точки говорит о наличии пустого оператора и является признаком ошибки.

ТЕОРИЯ МАКРОИКОН

ДРАКОН имеет не только мелкие фигурки (иконы), но и крупные, составные; они называются *макроиконы.*

Подобно тому, как слова состоят из букв, макроиконы (графические слова) состоят из икон (графических букв). Язык ДРАКОН имеет 23 макроиконы [2], рис. 21 и 22.

Иконы и макроиконы — это строительные блоки, из которых составляются дракон-алгоритмы.

Макроиконы служат для предотвращения ошибок при проведении соединительных линий.

МАКРОИКОНЫ РАЗВИЛКА И ЦИКЛ СТРЕЛКА

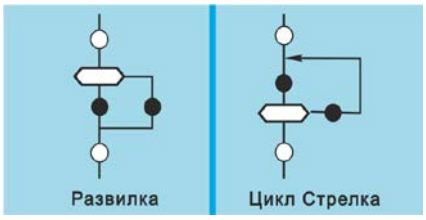


Рис. 32. Две макроиконы: Развилка и цикл Стрелка

Макроиконы Развилка и цикл Стрелка содержат только одну икону — икону Вопрос (рис. 32). Кроме того, они содержат соединительные линии и валентные точки.

Цикл Стрелка — пустой оператор, он служит заготовкой для построения реальных циклов. Если заполнить верхнюю валентную точку, получим цикл ДО (do – while). Если нижнюю — цикл ПОКА (while). Если обе — цикл do – while – do.

Макроикона Развилка — тоже пустой оператор. Если заполнить левую валентную точку, получим оператор ЕСЛИ (if). Если обе — оператор ЕСЛИ ТО (if – else).

Указанные Макроиконы обеспечивают выполнение требований визуального структурного программирования.

МОЖНО ЛИ ИСКЛЮЧИТЬ МАКРОИКОНУ «ЦИКЛ СТРЕЛКА» ИЗ МЕНЮ ПРОГРАММЫ «ДРАКОН-КОНСТРУКТОР»?

Возникает соблазн исключить из меню макроикону «цикл Стрелка» и смоделировать ее из макроиконы Развилка. Казалось бы, это можно легко сделать с помощью операции «Пересадка лианы» — нужно лишь оторвать правый выход иконы Вопрос в валентной точке и пересадить его наверх в нужную точку.

Однако в языке ДРАКОН так поступать запрещено, так как возникает серьезная опасность допустить ошибку.

Почему? Потому что речь идет об образовании нового цикла, аналогичного оператору goto, который реализует переход на оператор, расположенный выше (раньше) на чертеже ДРАКОН-алгоритма. Подобная операция в языке ДРАКОН считается небезопасной и недопустимой.

В самом деле, специалист по надежности программного обеспечения Гленфорд Майерс предостерегает:

> «Наихудшим применением оператора goto считается переход на оператор, расположенный выше (раньше) в тексте программы» [44], с. 135.

Запрет на образование нового цикла вызван тем, что переход на оператор, расположенный выше, является самым плохим (и недопустимым) использованием оператора перехода. Указанный запрет вводится, чтобы предотвратить появление ошибок.

Если же в меню программы «ДРАКОН-конструктор» предусмотрена макроикона «цикл Стрелка», то проблема полностью снимается, так как новый цикл строится на основе цикла Стрелка, и угроза ошибок исчезает.

СРАВНИВАЕМ СО СТАНДАРТОМ ГОСТ 19.701-90

Макроиконы и формализация валентных точек в ГОСТе не предусмотрены. Трудности понимания сложных блок-схем алгоритмов, выполненных по ГОСТ 19.701-90, связаны с тем, что в них отсутствует должный порядок — правила разработки схем не формализованы, не эргономичны и разрешают совершать нежелательные действия:

> «Основной недостаток блок-схем заключается в том, что они не приучают к аккуратности при разработке алгоритма. Ромб можно поставить в любом месте блок-схемы, а от него повести выходы на какие угодно участки. Так можно быстро превратить программу в запутанный лабиринт, разобраться в котором через некоторое время не сможет даже сам ее автор» [45].

В отличие от ГОСТа, в языке ДРАКОН икону Вопрос можно вставить не «в любом месте» чертежа, а только в валентных точках. И повести выходы не на «какие угодно участки», а только и исключительно в те места, которые разрешены согласно теории валентных точек и теории макроикон.

ТЕОРИЯ СТРЕЛОК

Вспомним еще раз правило Эшфорда:

> «Бесполезно стремиться направить внимание на важнейшие характеристики, если они окружены лишними, не относящимися к ним визуальными раздражителями, мешающими восприятию главного» [46].

Чтобы изобразить направление потока управления, в блок-схемах алгоритмов, начерченных по ГОСТ 19.701-90, обычно используют чрезмерно большое количество стрелок-указателей. Согласно Эшфорду, стрелки загромождают чертеж и отвлекают внимание, являясь лишними визуальными раздражителями.

В языке ДРАКОН, направление потока управления, как правило, показано другими средствами, более наглядными и эффективными — без использования стрелок. Стрелки используются крайне редко — только для представления «циклов Стрелка». В результате дракон-схемы становятся более удобными и прозрачными.

ФОРМАЛИЗОВАННЫЙ ЧЕРТЕЖ АЛГОРИТМА

Формализованный чертеж алгоритма — это чертеж, для которого определены:

- формальное описание алфавита графических фигур (икон и макроикон);
- формальное описание синтаксиса, то есть соединительных линий между фигурами;
- визуальное логическое исчисление, позволяющее из аксиомы-силуэт и аксиомы-примитив строить формализованный чертеж алгоритма методом логического вывода.

Язык ДРАКОН разработан исходя из этих предпосылок с учетом элементарных теорий, формализующих отростки, валентные точки, макроиконы и стрелки.

КРИТИЧЕСКИЙ АНАЛИЗ БЛОК-СХЕМ АЛГОРИТМОВ

Перейдем к анализу блок-схем алгоритмов, выполненных согласно стандартам ГОСТ 19.701-90 и ISO 5807:1985.

Лесли Робертсон (Австралия) в своем учебнике программирования характеризует блок-схемы как «альтернативный метод представления алгоритмов» и перечисляет их достоинства:

> «Блок-схемы популярны, так как они графически отображают логику программы с помощью стандартных геометрических фигур и соединительных линий. Блок-схемы относительно легко выучить, и они представляют собой интуитивно понятный метод представления управляющей последовательности алгоритма» [47].

Вместе с тем блок-схемы подвергаются критике. Противники блок-схем утверждают, что они не поддаются формализации, поэтому их «нельзя использовать как программу для непосредственного ввода в машину» [48]. Гленфорд Майерс полагает, что они не согласуются со структурным программированием, поскольку в значительной степени ориентированы на использование оператора перехода goto [44], с. 150. Это подтверждают и другие авторы:

> «Блок-схемы… затемняют особенности программ, созданных по правилам структурного программирования» [49].

> «С появлением языков, отвечающих принципам структурного программирования… блок-схемы стали отмирать» [45].

Существенно, что при большой степени детализации блок-схемы становятся «громоздкими и теряют своё основное достоинство — наглядность структуры алгоритма» [50]. Обозримыми и понятными являются блок-схемы только для небольших алгоритмов [51].

> «Если для простой задачи схемы алгоритмов обеспечивают безусловную наглядность, то по мере роста сложности программы ее логическая структура начинает “тонуть” в “клубке спагетти”, в который постепенно превращается схема алгоритма» [52].

Представление с помощью блок-схем «сложных алгоритмов затруднительно из-за существенно возрастающей громоздкости и быстрой потери наглядности» [53], с. 190.

Бертран Мейер дает блок-схемам жесткую отрицательную оценку:

> «В свое время блок-схемы были весьма популярны для выражения структуры управления программы. И сегодня их можно встретить для описания процессов, не связанных с программированием. В программировании они потеряли репутацию. (Некоторые авторы называют их не “flowchart”, а “flaw chart” — порочными схемами)…
>
> «Наши программы делают все более сложные вещи. Большие блок-схемы с вложенностью внутри циклов и условных операторов быстро становятся запутанными…
>
> «Аккуратно отформатированный программный текст с отступами, отражающими уровень вложенности, дает лучшее представление о порядке выполнения операторов», чем блок-схемы [54], с. 205, 206.

Но и это не все. В настоящее время активно развиваются системы управления и робототехника, широко используются алгоритмы реального времени. Однако блок-схемы плохо подходят для таких алгоритмов.

Почему? Потому что правила вычерчивания блок-схем согласно стандартам ГОСТ 19.701-90 и ISO 5807:1985 не имеют выразительных средств и обозначений для описания алгоритмов реального времени.

Чтобы устранить недостатки стандартов и обеспечить работу в реальном времени, в языке ДРАКОН предусмотрены иконы: Пауза, Период, Таймер и Синхронизатор, отсутствующие в блок-схемах.

ДЕЙСТВУЮЩИЙ СТАНДАРТ НА АЛГОРИТМЫ ГОСТ 19.701-90 НЕ ИМЕЕТ НАУЧНОГО ОБОСНОВАНИЯ

Является ли международный стандарт на блок-схемы ISO 5807:1985 научно обоснованным? Тот же вопрос относится и к его отечественному аналогу межгосударственному стандарту ГОСТ 19.701-90[1]. Последний, в частности, используется в системе образования с целью обучения основам алгоритмизации.

Упомянутые стандарты были созданы давно. Они опираются на опыт XX века и отражают исторически сложившийся набор правил выполнения документации по обработке данных. К настоящему времени правила устарели и *не имеют научного обоснования.*

Стандарты на блок-схемы алгоритмов и программ прямо противоречат принципам структуризации блок-схем, которые предложил Эдсгер Дейкстра в классической работе «Заметки по структурному программированию» [55], с. 25-28.

[1] В 1992 году государства-члены СНГ заключили соглашение, которым признали действующие стандарты «ГОСТ» СССР в качестве межгосударственных с сохранением аббревиатуры «ГОСТ» за вновь вводимыми межгосударственными стандартами.

ЧЕТЫРЕ ПРИНЦИПА СТРУКТУРИЗАЦИИ БЛОК-СХЕМ, ПРЕДЛОЖЕННЫЕ Э. ДЕЙКСТРОЙ

Непосредственный анализ первоисточника [55] со всей очевидностью подтверждает простую мысль. Дейкстрианская «структурная революция» началась с того, что Дейкстра, использовал блок-схемы как инструмент анализа структуры программ и предложил, наряду с другими важными идеями, четыре принципа структуризации блок-схем.

Эти принципы таковы.

1. ***Принцип ограничения топологии блок-схем.*** Структурный подход должен приводить

> «к ограничению топологии блок-схем по сравнению с различными блок-схемами, которые могут быть получены, если разрешить проведение стрелок из любого блока в любой другой блок. Отказавшись от большого разнообразия блок-схем и ограничившись ...тремя типами операторов управления, мы следуем тем самым некоей последовательностной дисциплине» [55], с. 28.

2. ***Принцип вертикальной ориентации входов и выходов блок-схемы.*** Имея в виду шесть типовых блок-схем (if-do, if-then-else, case-of, while-do, repeat-until, следование), Дейкстра пишет:

> «Общее свойство всех этих блок-схем состоит в том, что у каждой из них один вход вверху и один выход внизу» [55], с. 27.

3. ***Принцип единой вертикали.*** Вход и выход каждой типовой блок-схемы должны лежать на одной вертикали.

4. ***Принцип нанизывания типовых блок-схем на единую вертикаль.*** При последовательном соединении типовые блок-схемы следует соединять, не допуская изгибов соединительных линий, чтобы выход верхней и вход нижней блок-схемы лежали на одной вертикали.

Хотя Дейкстра не дает словесной формулировки третьего и четвертого принципов, они однозначно вытекают из имеющихся в его работе иллюстраций [55], с. 25-28. Чтобы у читателя не осталось сомнений, я привожу точные копии подлинных рисунков Дейкстры (рис. 33, средний и левый столбец). На том же рисунке в правом столбце продемонстрировано, что в дракон-схемах принципы структуризации аккуратно выполняются.

В ЯЗЫКЕ ДРАКОН РЕКОМЕНДАЦИИ ДЕЙКСТРЫ СОБЛЮДАЮТСЯ, А В СТАНДАРТЕ ГОСТ 19.701-90 — НАРУШАЮТСЯ

Обычная практика разработки и вычерчивания блок-схем не учитывает рекомендации Дейкстры. Это объясняется тем, что принципы Дейкстры не получили своего закрепления в стандартах на блок-схемы — международном стандарте ISO 5807:85 и ГОСТ 19.701—90.

Рекомендации Эдсгера Дейкстры очень важны, так как они открывают путь к совершенствованию блок-схем, делают их более удобными и наглядными. Дракон-схемы — это усовершенствованные блок-схемы, построенные на основе принципов Дейкстры. Принципы позволяют осуществить четкую структуризацию и формализацию схем алгоритмов.

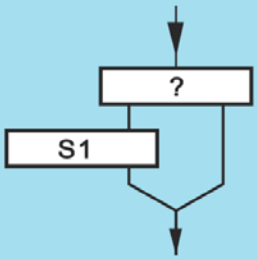
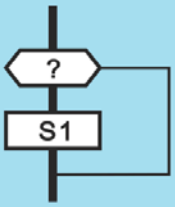
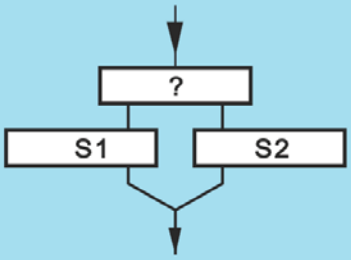
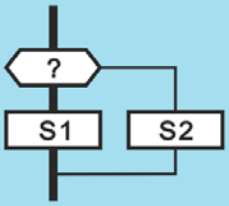
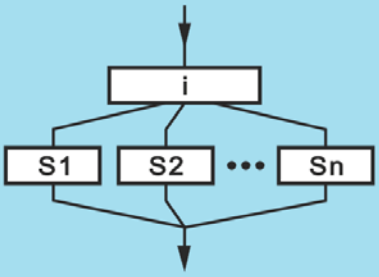
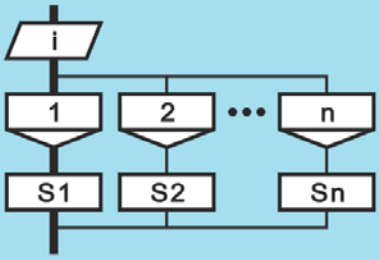
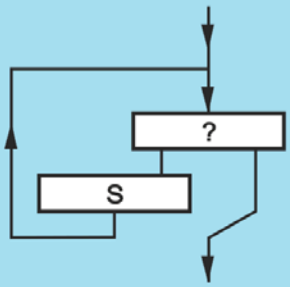
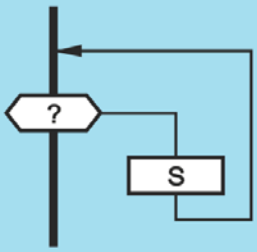
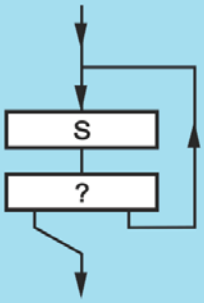
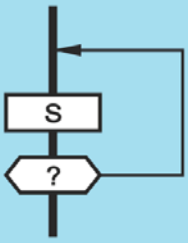
Структурные операторы Дейкстры	Структурные блок-схемы Дейкстры	Дракон-схемы
if ? do S1		
if ? then S1 else S2		
case i of (S1;S2;...Sn)		
while ? do S		
repeat S until ?		

Рис. 33. Структурные блок-схемы Дейкстры (в центре) и эквивалентные им дракон-схемы (справа)

ИДЕИ ДЕЙКСТРЫ И ЯЗЫК ДРАКОН

Идея Дейкстры об ограничении топологии схем программ с целью их лучшей структуризации и формализации лежит в основе визуального алгоритмического языка ДРАКОН и построенного на его основе шампур-метода как абстрактной визуальной модели программы [56], с. 104.

Дракон-схемы есть не что иное, как правильно составленные блок-схемы алгоритмов. Язык ДРАКОН строится на основе блок-схем с целью их улучшения. Использование формальных и эргономичных правил ДРАКОНа позволяет упорядочить графический чертёж алгоритма и обеспечить более эффективное восприятие алгоритма человеком.

БЛОК-СХЕМЫ УСТАРЕЛИ И УСТУПАЮТ МЕСТО ДРАКОН-СХЕМАМ

По мнению экспертов, хотя стандарты на блок-схемы считаются действующими, фактически они давно устарели. С появлением дракон-схем блок-схемы потеряли своё значение. [3], с. 32.

По мнению преподавателя СПбПУ Пышкина Е.В.:

> «Методы проектирования, ориентированные на дракон-схемы, позволяют преодолеть алгоритмическую сложность… Визуальный язык ДРАКОН образует наглядную среду для первоначального обучения программированию и мог бы быть весьма полезен при организации школьных курсов информатики» [56], с. 283.

Учительская газета предоставляет слово методисту Е. Белякову:

> «ДРАКОН — это… эргономичный стандарт для графического представления учебной информации… Язык ДРАКОН учит нас, методистов и учителей правильному составлению блок-схем… Насколько я знаю, нет другой литературы, где тому же самому можно научиться настолько просто и даже увлекательно» [57].

УПРАВЛЯЮЩИЙ ГРАФ АЛГОРИТМА

А.А. Тюгашев полагает, что «теоретической основой графического языка блок-схем служит управляющий граф программы» [58]. Это верно лишь отчасти:

- язык блок-схем не удовлетворяет требованиям, предъявляемым к формальному языку, их нельзя подать на вход компилятора. Как отмечает А.Н. Степанов, для конечного исполнителя-компьютера блок-схемы не обеспечивают свойство понятности: «процессорами компьютеров не воспринимаются алгоритмы, заданные в виде блок-схемы» [53], с. 190.
- принцип ограничения топологии блок-схем Дейкстры накладывает принципиальные ограничения на управляющий граф алгоритма *(control flow graph)* и является частью теоретического фундамента блок-схем. В стандартах ГОСТ 19.701-90 и ISO 5807:1985 эти ограничения никак не учтены, что делает указанные стандарты уязвимыми для критики, недостоверными и нелигитимными.

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ЯЗЫКА ДРАКОН

В отличие от стандарта ГОСТ 19.701-90, теоретической основой языка ДРАКОН служит визуальное логическое исчисление, благодаря чему графический синтаксис ДРАКОНа является не только формальным и безопасным, но и эргономичным, облегчающим выявление слабых мест [3], с. 393-448.

Кроме того, в языке ДРАКОН — на основе четырех принципов структуризации блок-схем Дейкстры — разработан двумерный структурный подход к алгоритмам и программам (шампур-метод) [3], с. 449-472. Метод содержит большое число правил, которые хранит в своей памяти программа ДРАКОН-конструктор. Последний, выполняя черчение дракон-схем, автоматически обеспечивает выполнение правил и не допускает ошибок.

МЕТОД Эдварда АШКРОФТ И Зохара МАННА

Известно, что пересечения соединительных линий в блок-схемах затрудняют понимание. Поэтому ГОСТ 19.701-90 предписывает:

> «В схемах следует избегать пересечения линий… При необходимости линии в схемах следует разрывать для избежания излишних пересечений» [43], с. 14.

Подобные рекомендации свидетельствует об неумении теоретически решить проблему пересечений.

В отличие от стандарта ГОСТ 19.701-90, в языке ДРАКОН эта проблема успешно решена. Разработан теоретический метод, исключающий пересечения и разрывы линий, а также устраняющий внутренние соединители.

Метод использует введение переменной состояния по Ашкрофту-Манна [59] и Эдварду Йодану [60], с. 192-196. Затем переменная состояния удаляется, превращаясь в идентификаторы икон «Имя ветки» и «Адрес» языка ДРАКОН [3], с. 436-448. Метод реализован на практике во внутренних алгоритмах программы ДРАКОН-конструктор.

ВЫВОДЫ

1. Стандарт на блок-схемы алгоритмов ГОСТ 19.701-90 прямо противоречит принципам структуризации блок-схем, которые предложил Эдсгер Дейкстра в классической работе «Заметки по структурному программированию».
2. Принцип ограничения топологии блок-схем Дейкстры накладывает жесткие ограничения на управляющий граф алгоритма (control flow graph) и является частью теоретического фундамента схем алгоритмов.
3. В стандарте ГОСТ 19.701-90 эти ограничения никак не учтены, что делает указанный стандарт уязвимым для критики и нелигитимным.
4. В языке ДРАКОН — на основе четырех принципов структуризации блок-схем Дейкстры — разработан двумерный структурный подход к алгоритмам (шампур-метод).
5. В отличие от блок-схем, в языке ДРАКОН решена проблема пересечений соединительных линий: разработан теоретический метод, исключающий пересечения и разрывы линий. Метод использует введение переменной состояния по Ашкрофту-Манна и Йодану с последующей модификацией.
6. Правила разработки блок-схем алгоритмов по ГОСТ 19.701-90 не формализованы, не эргономичны и разрешают совершать небезопасные действия.

7. В отличие от блок-схем в дракон-схемах проведена:
    - формализация построения графики с помощью визуального логического вывода;
    - формализация отростков соединительных линий;
    - формализация точек размещения икон (валентных точек);
    - формализация макроикон и стрелок.
8. Представление с помощью блок-схем сложных алгоритмов вносит неоправданные трудности из-за существенно возрастающей громоздкости и быстрой потери наглядности.
9. Блок-схемы не имеют выразительных средств для представления алгоритмов реального времени, робототехники и систем управления.
10. С появлением дракон-схем (drakon-charts) блок-схемы алгоритмов по ГОСТ 19.701-90 полностью потеряли свое значение, так как они по всем параметрам хуже дракон-алгоритмов.

Часть 6. КАКИМ ДОЛЖЕН БЫТЬ СТАНДАРТ НА АЛГОРИТМЫ ДЛЯ ВОЕННОГО ОБРАЗОВАНИЯ

АКТУАЛЬНЫЕ ПРОБЛЕМЫ ВОЕННОГО ОБРАЗОВАНИЯ

Специалист по военному образованию В. В. Полич из Новосибирского военного института внутренних войск имени генерала армии И. К. Яковлева отмечает:

> «Военное образование во всем мире справедливо определяется как важная составляющая государственной безопасности. Высококвалифицированные кадры военных специалистов разных уровней есть реальная сила, которая создает оборонный потенциал государства, его военную мощь…
>
> «Процесс модернизации военного образования и науки в России является объективной социальной потребностью как ответ вызовам XXI века… В этом контексте подготовка высококвалифицированных военных специалистов, а также модернизация военного образования и науки – веление времени.
>
> «Динамика развития военного дела во всем мире, неуклонное внедрение достижений военной науки в повседневную деятельность и боевое применение войск, совершенствование военных технологий… делают очевидным, что невозможно подготовить современного офицера на старом багаже, во многом унаследованном от Советской армии…
>
> «В настоящее время Россия в очередной раз находится на этапе реформирования военного образования, процесс которого пока еще не завершен» [61].

К актуальным проблемам военного образования следует также отнести проблему совершенствования и стандартизации военных, военно-промышленных и медицинских ***алгоритмических знаний.*** Это важно по ряду причин, в том числе потому, что для военной школы России характерна «высокая удельная стоимость подготовки офицеров в военных образовательных организациях – порядка нескольких сотен тысяч рублей в год» [61].

Использование стандарта языка ДРАКОН вместо ГОСТ 19.701-90 для фиксации и стандартизации военных, военно-промышленных и медицинских алгоритмических знаний позволит существенно упростить представление алгоритмических знаний в Вооруженных Силах и оборонно-промышленном комплексе России, сделать их более эргономичными, наглядными и удобными (people-friendly) для военных специалистов.

Вследствие этого высокая удельная стоимость подготовки офицеров в военных образовательных организациях может заметно уменьшиться.

ПРОБЛЕМА СТАНДАРТИЗАЦИИ АЛГОРИТМОВ ДЛЯ ВОЕННОГО ОБРАЗОВАНИЯ

Стандарт графического представления алгоритмов (далее стандарт алгоритмов) есть единая система обозначений (единая нотация) для записи алгоритмов. В настоящее время проблема стандартизации алгоритмов не решена.

На практике для записи алгоритмов применяются разнообразные средства: псевдокоды, блок-схемы (flowcharts), схемы деятельности (activity diagrams) языка UML, дракон-схемы (drakon-charts).

В программировании (с появлением структурного программирования) подробные блок-схемы алгоритмов стали ненужными; вместо них используются псевдокоды, как правило, не подлежащие стандартизации.

В российском оборонно-промышленном комплексе при выпуске документации на алгоритмы используется межгосударственный стандарт ГОСТ 19.701-90, полученный методом прямого применения из международного стандарта ISO 5807:1985.

Проблема в том, что стандарты ГОСТ 19.701-90 и ISO 5807:1985 обладают существенными недостатками; они не удовлетворяют требованиям для записи военных, военно-промышленных и медицинских ***алгоритмических знаний.*** И потому вносят существенные трудности для курсантов и слушателей в системе военного образования.

ТРЕБОВАНИЯ К СТАНДАРТУ АЛГОРИТМОВ

Уже говорилось, что существующие нотации и алгоритмические языки не предотвращают алгоритмические ошибки, а наоборот, содействуют их появлению, являясь своеобразным катализатором ошибок [2], с. 352. Новая нотация должна в максимальной степени содействовать сокращению числа ошибок в алгоритмах.

Следует подчеркнуть, что несмотря на значительные финансовые потери, вызванные ошибками, проблема ошибок по-прежнему существенно недооценивается, а предотвращению ошибок в военном деле, оборонно-промышленном комплексе и смежных сферах уделяется явно недостаточное внимание.

Приведу лишь два примера.

- Инцидент с самолетом Боинг 737 Max 8, алгоритмы которого оказались ошибочными и привели к катастрофе. Трагедия повторилось дважды: 29 октября 2018 года в Индонезии и, спустя полгода, 10 марта 2019 года, в Эфиопии. Погибли 346 человек: все пассажиры и оба экипажа.

  Потеря двух новых дорогостоящих самолетов, гибель свыше трехсот человек, запрет на продолжение полетов и отказ ряда авиакомпаний от закупок новых самолетов 737 Max явились тяжелым ударом для Боинга.

  Одновременно это выявило упущения в работе Федеральной авиационной администрации США — Federal Aviation Administration (FAA), которая отвечает за безопасность и сертификацию самолетов и летчиков.

  На слушаниях в конгрессе США расчеты компании Боинг и ее руководителя Денниса Мюленбурга подверглись острой критике. Сенатор Ричард Блюменталь заявил: «У этих пилотов не было шансов. У этих кем-то любимых людей не было шансов. Они находились в летающих гробах».

  Я подробно проанализировал этот случай в работе [5], глава 30.
- Нештатный пуск 28 ноября 2017 года с космодрома ВОСТОЧНЫЙ ракеты-носителя (РН) «Союз-2.1б» с разгонным блоком (РБ) «Фрегат», космическим аппаратом «Метеор-М» и попутной полезной нагрузкой.

  Цель пуска не была достигнута, космический аппарат «Метеор-М» и попутная полезная нагрузка были утеряны. Причиной неудачи, по мнению аварийной комиссии Роскосмоса, была «скрытая проблема в алгоритме, которая не проявлялась десятилетиями успешных пусков связки Союз-Фрегат» [62].

Проблема ошибок является чрезвычайно важной и актуальной; поэтому ***предотвращение ошибок должно быть предусмотрено на уровне стандарта для записи алгоритмов.***

ЧТО ЛУЧШЕ ДЛЯ РОССИЙСКОГО ВОЕННОГО ОБРАЗОВАНИЯ: ДРАКОН-СХЕМЫ ИЛИ БЛОК-СХЕМЫ ПО ГОСТ 19.701-90?

Многие авторы высказываются в поддержку языка ДРАКОН, например:

> «Блок-схемы, нарисованные по правилам языка ДРАКОН, отличаются поразительной четкостью, наглядностью и прозрачностью структуры. А наглядность и доходчивость алгоритмов — это именно то, чего так остро недостает школьным учебникам» [63].
>
> «Алгоритмический язык ДРАКОН… используется в технике, биологии, медицине и образовании. Преимуществом этого языка является то, что схемы легко рисовать и понимать, они очень наглядны» [64].
>
> «Язык усовершенствованных графических схем ДРАКОН обеспечивает разработку сложных алгоритмов с сохранением наглядности даже для многостраничных схем» [65].
>
> «Язык ДРАКОН – удобный инструмент для записи и структурирования деятельности в виде алгоритмов…, даёт глубокое понимание сложных проблем, позволяет проектировать сложную деятельность, бизнес-процессы, формализовать свои профессиональные знания» [66].
>
> «Использование языка ДРАКОН и гибридных языков может позволить полностью отказаться от традиционного подхода к разработке ПО РК [программного обеспечения робототехнических комплексов], снижая интеллектуальную нагрузку на разработчика алгоритма, исключая ошибки толкования исходных данных программистом… Но самое главное — это позволит резко сократить затраты на разработку ПО РК, что сделает роботов более доступными для потребителей самого разного уровня» [67].

Приведем также отзыв рядового пользователя, размещенный в сети интернет участником с ником dvuugl:

> «Если нужно рисовать алгоритм, теперь только и только на Драконе. Считаю, что он должен стать государственным стандартом для блок-схем вместо существующего. Удивительно, что авторы книг продолжают использовать прежние схемы, на которые после Дракона без ужаса смотреть невозможно».

С другой стороны, несмотря на явное преимущество дракон-схем, в большинстве российских школ и вузов язык ДРАКОН не изучают, предпочитая устаревшие блок-схемы. Причина проста. Блок-схемы опираются на авторитет стандартов ГОСТ 19.701-90 и ISO 5807:1985, а дракон-схемы такой поддержки не имеют. Многие преподаватели вузов и учителя школ продолжают знакомить студентов и школьников с неэргономичными блок-схемами, оправдывая свои действия необходимостью соблюдать требования стандарта ГОСТ 19.701-90.

В военном образовании наблюдается та же картина.

СТАНДАРТЫ, КОТОРЫЕ ОТСТАЛИ ОТ ЖИЗНИ

Проведенный анализ позволяет сделать вывод, что морально устаревший стандарт ISO 5807:1985 (и его калька ГОСТ 19.701-90) препятствуют распространению новых, более удобных и эффективных форм представления алгоритмических знаний. Указанные стандарты оказывают негативное воздействие на

систему систему военного образования, а также в целом на систему среднего и высшего образования России.

Таким образом, налицо парадоксальная ситуация. Система образования должна распространять лучшее, а не худшее. Однако на деле происходит обратное. Российским преподавателям, учащимся и специалистам навязан устаревший и не имеющий научного обоснования стандарт только потому, что он скопирован с международного стандарта. Возникло противоречие между устаревшим стандартом и новой практикой.

Это противоречие следует устранить. Учитывая вышеизложенное, необходимо отказаться, прежде всего, в системе военного образования, от некачественного стандарта ГОСТ 19.701-90 и в качестве государственного стандарта на алгоритмы разработать стандарт, основанный на языке ДРАКОН.

Академик Российской академии ракетных и артиллерийских наук Безель Я. В. Главный конструктор ракетной системы Противовоздушной обороны г. Москвы и Московского промышленного района в журнале «Вестник Российской академии наук» указывает:

> «При разработке единого стандарта на блок-схемы, снабженного компьютерной поддержкой и рассчитанного на постепенное внедрение во всех отраслях и предметных областях, целесообразно взять за основу язык ДРАКОН» [68].

ЯЗЫК ДРАКОН УСТРАНЯЕТ НЕДОСТАТКИ БЛОК-СХЕМ

ДРАКОН упорядочивает блок-схемы за счёт формализации, эргономизации и неклассической структуризации [69] [70]. После появления дракон-схем (drakon-charts) блок-схемы алгоритмов по ГОСТ 19.701-90 полностью утратили свое значение, так как они ***во всех отношениях уступают дракон-схемам*** [3]**,** с. 32-36, 242-254.

В 1996г. Государственный комитет Российской Федерации по высшему образованию по решению Президиума научно-методического совета по информатике под председательством академика РАН Ю. И. Журавлева включил изучение языка ДРАКОН в Примерную программу дисциплины «Информатика» для бакалавров для направлений:

510000 – Естественные науки и математика,
540000 – Образование,
550000 – Технические науки,
560000 – Сельскохозяйственные науки [9].

Профессор А.Н. Степанов в «Курсе информатики для студентов информационно-математических специальностей» (2018 год) отмечает:

> «Существуют близкие к блок-схемам языки визуального программирования, такие как… язык программирования и моделирования ДРАКОН. В этом языке используются графические элементы, аналогичные стандартным элементам блок-схем… Но для обеспечения возможности автоматического преобразования графической программы в машинный язык введены строгие правила задания графических и текстовых элементов такой программы» [53], с. 190.

Далее Степанов излагает концепцию языка ДРАКОН и указывает его преимущества:

> «В рамках этого подхода основные управляющие конструкции следования, ветвления и цикла, которые в обычных алгоритмических языках задаются с помощью служебных слов, таких, как *begin, end, if, then,*

else, while, do и т. д., заменяются управляющей графикой, похожей на стандартные элементы блок-схем…

«Язык двумерного структурного программирования ДРАКОН является полным по Тьюрингу и относится к универсальным языкам программирования… Использование гибридных языков позволяет отказаться от текстовых управляющих структур, используемых в обычных языках, и заменить их управляющей графикой языка ДРАКОН. Написание программы становится более понятным и удобным для человека, повышается производительность труда программистов» [53], с. 1017-1019.

СЛЕДУЕТ РАЗЛИЧАТЬ АЛГОРИТМЫ И ПРОГРАММЫ

В литературе термины *алгоритм* и *программа* нередко используются как синонимы. Однако при создании стандарта это недопустимо; их необходимо строго различать.

Стандарт ГОСТ 19.701-90 нарушает это правило; он называется «Схемы алгоритмов, программ, данных и систем» и не проводит границу между алгоритмами и программами.

ТЕЗИС АКАДЕМИКА ДОРОДНИЦЫНА

Академик А. А. Дородницын четко проводит указанную границу, подчеркивая, что «без алгоритмов предмета информатики не существует» [71]. Более того, он предлагает выделить «алгоритмические средства» как отдельную, самостоятельную сущность:

> «…состав информатики — это три неразрывно и существенно связанные части: технические средства, программные средства и алгоритмические средства. Если о первых двух частях никогда не забывают — … они получили специальные термины «hardware» и «software», — то алгоритмическая часть информатики остается почему-то в тени… об этой важнейшей части информатики просто забывают» [71].

Таким образом, согласно Дородницыну, ***алгоритмические средства должны составлять третью, самоценную часть информатики, наряду с программными средствами (software) и техническими средствами (hardware)*** [71].

Чтобы в полной мере реализовать идею Дородницына, нужно иметь отдельный стандарт, посвященный алгоритмам, содержащий эргономичную нотацию для записи алгоритмов, пригодную для подавления ошибок.

ВЫВОДЫ

1. Блок-схемы алгоритмов (flowcharts), описанные в ГОСТ 19.701-90 и международном стандарте ISO 5807:85, обладают серьезными дефектами. Пользоваться ими недопустимо.
2. Вместо блок-схем для записи алгоритмов следует использовать дракон-схемы (drakon-charts).
3. Для Вооруженных Сил и оборонно-промышленного комплекса России необходимо разработать и выпустить отдельный стандарт, посвященный алгоритмам, на основе языка ДРАКОН.

4. В 1996 году Государственный комитет РФ по высшему образованию по решению Президиума научно-методического совета по информатике включил изучение языка ДРАКОН в программу курса «Информатика».
5. Тем не менее, преподаватели военных образовательных учреждений продолжают знакомить будущих офицеров и сержантов с морально устаревшими блок-схемами, обосновывая такую практику необходимостью ориентироваться на действующий стандарт ГОСТ 19.701-90.
6. Преподавателям военных образовательных учреждений можно рекомендовать ознакомиться с языком ДРАКОН, убедиться в его преимуществах и организовать изучение языка ДРАКОН на лекциях, практических занятиях, курсовых и дипломных работах.

Часть 7. УПРАВЛЕНИЕ ВОЙНОЙ И ВОЕННЫЕ АЛГОРИТМЫ

ДЕКЛАРАТИВНЫЕ И ПРОЦЕДУРНЫЕ ЗНАНИЯ В ВОЕННОЙ НАУКЕ

Теория войны, военного дела и военно-стратегического управления на протяжении всей истории (как, впрочем, и вся наука со времен Аристотеля) развивалась преимущественно или даже почти исключительно в форме *декларативного знания* (descriptive knowledge).

Процедурное военное знание (procedural knowledge) в военной науке встречается лишь изредка в форме изолированных «вкраплений», со всех сторон окруженных декларативными знаниями, такими как определения, логические рассуждения, пояснения, обоснования, аргументация, ссылки на уроки минувших войн и т.д. Процедурные и алгоритмические знания в виде точных и подробных пошаговых описаний сложных разветвленных и циклических алгоритмических процессов в военной научной и учебной литературе почти не встречаются.

Процедурные и алгоритмические «вкрапления» в декларативный текст похожи на множество крошечных островков в безбрежном океане военной литературы. Между островками должны быть соединительные алгоритмические мостики, но они, как правило, лишь подразумеваются и в явном виде не описаны. Это фундаментальный недостаток мировой и отечественной научной военной литературы.

Подавляющее число процедурных «вкраплений» существует в военных публикациях в виде изолированных точек. Они остаются одинокими алгоритмическими «сиротами», не объединяются в логически законченные фрагменты и комплексы, и не превращаются в целостную систему военных алгоритмов.

Таким образом, военная научная и учебная литература содержит, как правило, лишь декларативное, но не процедурное и не алгоритмическое знание. Причин тому две:

- недооценка важности процедурного и алгоритмического знания;
- отсутствие удобной нотации и выразительных средств для наглядной записи алгоритмов военного назначения и управления (руководства).

Существует острая потребность в создании выразительной и мощной алгоритмической нотации. Без такой нотации очень трудно и почти невозможно представление процедурных и алгоритмических знаний в наглядном и пригодном для обучения и печати виде.

Отсутствие легко воспринимаемых эргономичных графических алгоритмов высокой точности в военных учебниках и руководствах мешает делу. Оно затягивает сроки обучения и способствуют тому, что профессиональная подготовка будущих офицеров становится неоправданно дорогостоящей. Личный состав и курсанты недополучают нужные знания, которые восполняют лишь в процессе службы.

Сказанное означает, что проблема алгоритмизации военной научной и учебной литературы не только не решена, но даже не осознана.

Принятие стандарта языка ДРАКОН в качестве стандарта Вооруженных Сил России может содействовать устранению этого недостатка и повысить качество подготовки военных специалистов.

ПРОЦЕДУРНОЕ И АЛГОРИТМИЧЕСКОЕ ЗНАНИЕ

Полезно различать *процедурное* и *алгоритмическое* знание. Отличие между ними наглядно показано на рис. 34.

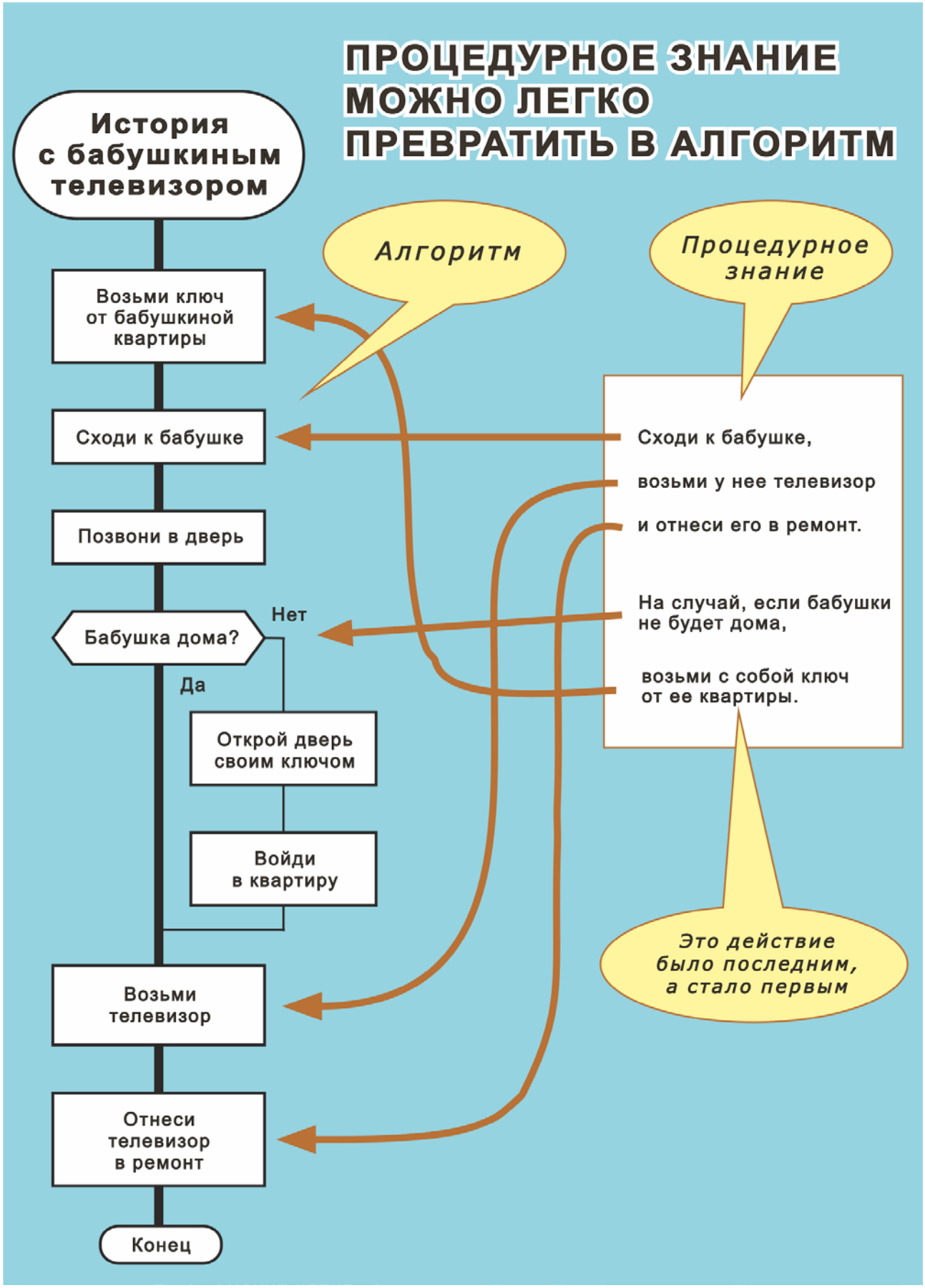


Рис. 34. Справа в рамке показан пример процедурного знания. После незначительных уточнений процедурное знание превращается в алгоритмическое, изображенное слева.

ВОЙНА КАК ЗАДАЧА УПРАВЛЕНИЯ (РУКОВОДСТВА) ПО А.А. КОКОШИНУ

Отмечая, что стратегическое военное управление имеет свою специфику, А.А. Кокошин подчеркивает:

> «по многим параметрам управление в военной сфере имеет общие черты с другими сферами человеческой деятельности, где прилагаются управленческие усилия, действуют те или иные системы управления» [72].

Учитывая это обстоятельство, после необходимой проработки можно использовать алгоритмический язык ДРАКОН единым образом для наглядного **алгоритмического моделирования и управления** в различных областях для стратегического управления государством, войной, медициной, экономикой, образованием, а также для частных военных задач в области военной подготовки, военного планирования, военных действий.

Далее Кокошин продолжает:

> «В силу огромной значимости вопросов войны и мира для любого государства соответствующие вопросы управления (руководства) должны отрабатываться заблаговременно и тщательно, на научной основе, а не решаться спонтанно, без тщательной отработки. Система стратегического политико-военного управления должна быть гибкой и вариативной, способной решать задачи при различных вариантах развития обстановки, при различных масштабах использования военной силы. Такая система управления (руководства) необходима и для обеспечения надежного, убедительного сдерживания» [72].

Чтобы решить задачу, поставленную А.А. Кокошиным, а именно: сделать систему стратегического политико-военного управления *«гибкой и вариативной, способной решать задачи при различных вариантах развития обстановки, при различных масштабах использования военной силы»,* необходимо:

- заблаговременно разработать — опираясь на принципы языка ДРАКОН — **принципы алгоритмическое управления** решением военных задач при различных вариантах развития международной военной обстановки;
- представить решение в виде полного комплекта эргономичных визуальных алгоритмов высокой точности, созданных с помощью языка ДРАКОН;
- провести учебное моделирование развития военной обстановки в динамике, отображаемое на большом экране высокого разрешения, в частности, в ситуационной комнате;
- обеспечить наглядность и удобочитаемость интеллектуальной зрительной сцены на большом экране в удобном виде в режиме анимации, пригодном для самостоятельной работы высшего политико-военного руководства с возможностью по своему желанию управлять и следить за динамикой процесса по перемещению управляемого светового пятна, бегущего по иконам дракон-алгоритма и дополнительных эргономичных средств.
- создать эргономичный тренажер для **алгоритмического управления войной** для высшего политико-военного руководства;

По моему мнению, алгоритмический язык ДРАКОН как источник принципиально новых идей ***может оказать существенную помощь при решении задач военного стратегического управления и моделирования.***

ЗАКЛЮЧЕНИЕ

Принятие качественного алгоритмического стандарта Вооруженных Сил России на основе языка ДРАКОН позволит:

- значительно сократить число ошибок в алгоритмах, программах и алгоритмических предписаниях (жизнеритмах) военного и военно-промышленного назначения, а также в целом по оборонно-промышленному комплексу;
- улучшить взаимопонимание между специалистами различных структур и служб Министерства обороны РФ на основе передового опыта, связанного с переходом на использование ***эргономичных алгоритмов (жизнеритмов) высокой точности***, обеспечивающих удобочитаемость, быстрое восприятие, точное понимание и легкое усвоение алгоритмических знаний;
- облегчить и ускорить овладение алгоритмами для тех структур Министерства обороны, которые лишь начинают использовать алгоритмы, например, для военной медицины;
- уменьшить потери личного состава в боестолкновениях за счет ускоренного алгоритмического обучения [73] [74] военнослужащих доступным приемам реанимации раненых, осуществляемой сослуживцами безотлагательно после ранения, когда помощь военных врачей по разным причинам задерживается или недоступна.

На основании тщательно проанализированных аргументов, представленных в данном исследовании, можно сделать обоснованный вывод:

Необходимо принять стандарт языка ДРАКОН в качестве стандарта для описания алгоритмов, используемых

- **при функционировании Вооруженных Сил России, включая медицинское и тыловое обеспечение Вооруженных Сил;**
- **при разработке и эксплуатации вооружений в оборонно-промышленном комплексе РФ.**

СПИСОК ЛИТЕРАТУРЫ

[1] Паронджанов В. Д. Графический синтаксис языка ДРАКОН // Программирование, 1995, №3. — С. 45-62.

[2] Паронджанов В.Д. Алгоритмы и жизнеритмы на языке ДРАКОН. Разработка алгоритмов. Безошибочные алгоритмы. — М., 2019. — 374 с. — Иллюстраций: 195 — https://drakon.su/_media/24_zhizneritm20.pdf.

[3] Паронджанов В.Д. Учись писать, читать и понимать алгоритмы. Алгоритмы для правильного мышления. Основы алгоритмизации. — М.: ДМК Пресс, 2012, 2014, 2016. — 520 с. — http://bit.ly/2Mlg4Ou.

[4] Паронджанов В.Д. Почему врачи убивают и калечат пациентов, или Зачем врачу блок-схемы алгоритмов? Иллюстрированные алгоритмы диагностики и лечения — перспективный путь развития медицины. Клиническое мышление высокой точности и безопасность пациентов. /, Предисловие члена-корреспондента РАН Г.В. Порядина. — М.: ДМК Пресс, 2017. — 340 с. — Иллюстраций: 130.

[5] Паронджанов В.Д. Дружелюбные алгоритмы, понятные каждому. Как улучшить работу ума без лишних хлопот. — М.: ДМК-пресс, 2010, 2014, 2016. — 464 с.

[6] Паронджанов В.Д. Как улучшить работу ума. Алгоритмы без программистов — это очень просто! — М.: Дело, 2001. — 360 с. — https://bit.ly/3cfXoHx.

[7] Паронджанов В.Д. Визуальный алгоритмический язык ДРАКОН в ракетной технике и медицине // Материалы межведомственной конференции «Современные автоматизированные системы управления реального времени как прикладное развитие научных достижений кибернетики, (к 100-летию со дня рождения И.А. Полетаева)», 24 марта 2016 г. Научные труды 3 ЦНИИ Министерства обороны РФ. — М.: 3 ЦНИИ Минобороны РФ, 2016. — 218 с. — С. 57—78 — https://bit.ly/2yN4DsM.

[8] Морозов В.В., Трунов Ю.В. и др. Система управления межорбитального космического буксира «Фрегат» // Вестник «ФГУП НПО им. С.А. Лавочкина», 2014, №1. — С. 16-25 —, https://bit.ly/3epFZ0m.

[9] Примерная программа дисциплины «Информатика». Издание официальное. — М.: Госкомвуз, 1996. — 21 с. / См. разделы 3 и 4, а также Приложение, пункты 1-7. — http://drakon.su/_media/biblioteka/progr_drakon.pdf.

[10] ДРАКОН // Википедия — http://bit.ly/2VOeRzc.

[11] DRAKON // Wikipedia — https://en.wikipedia.org/wiki/DRAKON.

[12] Сайт языка ДРАКОН. — https://drakon.su/.

[13] Форум языка ДРАКОН. — http://forum.drakon.su/.

[14] Марценюк В.Б. Книга, которая учит как вырастить помидоры. Учебное пособие. — Пермь: ПС Гармония, 2002. — 24 с.

[15] Саркисян А. А. Повышение качества программ на основе автоматизированных методов. — М.: Радио и связь, 1991. — 160 с. — С. 21.

[16] ГОСТ 28806-90. Качество программных средств. Термины и определения.

[17] Бурцева Н., Петров Е. Жирограф и ДРАКОН Пилюгина. Телестудия Роскосмоса. 17 мая 2008. — https://www.youtube.com/watch?v=YFJzHKHRc48&t=7s.

[18] Непейвода Н.Н. Алгоритм // Новая философская энциклопедия: В 4-х т. / Ин-т философии РАН. / Под. ред Степина В.С., Гусейнова А.А. и др. — М.: Мысль, 2010. — Том 1. — 744с. — С. 76. http://iph.ras.ru/elib/0111.html.

[19] Ланда Л. Н. Алгоритмизация в обучении. / Под общей ред. и со вступительной статьей Б. В. Гнеденко и Б. В. Бирюкова. — М.: Просвещение, 1966. — 523 с. — Глава 2. § 1. Понятие предписания алгоритмического типа. — С. 35.

[20] To Err is Human: Building a Safer Health System / Linda T. Kohn, Janet M. Corrigan, and Molla S. Donaldson, editors. — Committee on Quality of Health Care in America, Institute of Medicine. 2000. — 312 p. — http://www.nap.edu/catalog/9728.html.

[21] Hayward R.A., Hofer T.P. Estimating Hospital Deaths Due to Medical Errors: Preventability Is in the Eye of the Reviewer // JAMA: the Journal of the American Medical Association. — July 25, 2001, Vol. 286, No. 4. — pp. 415–420. —, http://jama.jamanetwork.com/article.aspx?articleid=194039.

[22] Специализированная реанимация новорожденного. Учебник. / Под ред. Р.Й. Надишаускене. — Литва: Центр исследования кризисов, Университет наук здоровья Литвы, 2012. — 396 с.

[23] Crossing the Quality Chasm: a New Health System for the 21st Century. — Committee on Quality Health Care in America, Institute of Medicine. 2001. — xxi + 337p. — 364 p. —, ISBN: 0-309-51193-3 — http://www.nap.edu/catalog/10027.html.

[24] Preventing Medication Errors: Quality Chasm Series / Philip Aspden, Julie Wolcott, J. Lyle Bootman, Linda R. Cronenwett, Editors. — Committee on Identifying and Preventing Medication Errors. Board on Health Care Services. Institute of Medicine. 2007. —, 480 p. — ISBN 978-0-309-10147-9. — http://nap.edu/11623 .

[25] Improving Diagnosis in Health Care. / Erin P. Balogh, Bryan T. Miller, and John R. Ball, Editors. — Committee on Diagnostic Error in Health Care. Board on Health Care Services. Institute of Medicine., — The National Academies of Sciences, Engineering, and Medicine. — 472 p. — ISBN 978-0-309-37769-0. — http://nap.edu/21794.

[26] Паронджанов В. Д. Можно ли улучшить медицинский язык? // Человек. — №1. 2016. — С. 105-122.

[27] Паронджанов В. Д. Алгоритмизация медицины и реформа медицинского языка // Евразийский союз ученых, №11 (20), 2015 / Медицинские науки. — С. 144–156.

[28] Паронджанов В. Только со смертью догмы начинается наука. Профессиональная лексика тормозит развитие отрасли // Медицинская газета, 23 декабря 2016, №97.

[29] Космический язык программирования из СССР. Видео. — Русская служба BBC. — https://www.youtube.com/watch?v=Bj6l0W2BBnM.

[30] Космический ДРАКОН. Как заброшенный проект "Роскосмоса" подарил язык литовской медицине / Воронин Н. Корреспондент по вопросам науки. 5 августа 2019. — https://bbc.in/2TWE5N6.

[31] Начальная неотложная акушерская помощь. Учебник. / Под ред. Р. Й. Надишаускене. — Литва: Центр исследования кризисов, Университет наук здоровья Литвы, 2012. — 204 с.

[32] Неотложная медицинская помощь. Учебник. / Под ред. Д. Вайткайтиса. — Литва: Центр исследования кризисов, Университет наук здоровья Литвы, 2012. — 265 с.

[33] Травма. Учебник. / Под ред. Д. Вайткайтиса. — Литва: Центр исследования кризисов, Университет наук здоровья Литвы. — 2012. — 440 с.

[34] Vileikytė A., Nadišauskienė R.J. et al. Algoritminės „Drakon“ kalbos pritaikymas medicinoje // Lietuvos akušerija ir ginekologija. — 2014 rugsėjis, tomas XVII, Nr. 3. 192–196.

[35] Порядин Г.В. Предисловие. Перспективы развития медицины и медицинского образования // Паронджанов В.Д. Почему врачи убивают и калечат пациентов, или Зачем врачу блок-схемы алгоритмов? Иллюстрированные алгоритмы диагностики и лечения. — С. 16-20.

[36] Ломов Б. Ф. Эргономические (инженерно-психологические) факторы художественного конструирования // Учебно-методические материалы по художественному конструированию. — М., 1965.

[37] Хьюз Дж., Мичтом Дж. Структурный подход к программированию / пер. с англ., под ред. В. Ш. Кауфмана. – М.: Мир, 1980. – 278 с. – С. 80.

[38] Криницкий Н.А. Алгоритмы вокруг нас. — М.: Наука, 1984. — 224 с. — С. 6.

[39] Питерс Л. Дж. Методы отображения и компоновки программных средств // Труды института по электротехнике и радиоэлектронике ТИИЭР. 1980. Т. 68, №9. – С. 60.

[40] Семакин И. Г. Основы алгоритмизации и программирования. — М.: ИЦ Академия, 2008. – С. 32, рис. 113в.

[41] Образцы итоговых заданий по оценке качества подготовки школьников по информатике // Информатика и образование. 1999. №9. — С. 8-21.

[42] Фокин Ю. Г. Теория и технология обучения: деятельностный подход. Учебное пособие для студентов высших учебных заведений. — 3-е изд., испр.. — М.: Издательский центр «Академия», 2008. — 240 с. — С. 233.

[43] ГОСТ 19.701-90 (ИСО 5807—85). Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. — М.: Издательство стандартов, 1991. — С. 8.

[44] Майерс Г. Надежность программного обеспечения / Пер с англ. / Под ред. В.Ф. Кауфмана. — М.: Мир, 1980. — 360 с. — С. 292.

[45] Очков В. Ф., Пухначев Ю. В. 128 советов начинающему программисту. 2-е изд. — М.: Энергоатомиздат, 1992. — 256 с. — С. 21.

[46] Венда В. Предисловие к русскому изданию. – В кн.: Боумен У. Графическое представление информации. — М.: Мир, 1971. – С. 9.

[47] Робертсон Л. А. Программирование — это просто. Пошаговый подход / Перевод с 4-го англ. издания. — М.: БИНОМ. Лаборатория знаний, 2008. — 383 с. — С. 265.

[48] Вельбицкий И. В. // Знакомьтесь, Р-технология // НТР: Проблемы и решения. — 1987. № 13. — С. 5.

[49] Толковый словарь по вычислительным системам. / Под ред. В. Иллингуорта, Э.Л. Глейзера, И.К. Пайла / Пер. с англ. / Под ред. Е.К. Масловского. — М.: Машиностроение, 1991. — 560 с. — С. 193.

[50] Семёнов Н. М. Программирование и основы алгоритмизации. Учебное пособие. — Томск: Томский политехнический университет, 2009. — С. 71. — 90 с.

[51] Дробушевич Л. Ф., Конах В. В. Анализ топологий визуальных нотаций для записи алгоритмов и программ // Информационные технологии и системы 2011 (ИТС 2011) : материалы межд. науч. конф., БГУИР, Минск, 26 окт. 2011. — 306 с. — С. 212—213.

[52] Пышкин Е.В. Структуры данных и алгоритмы: реализация на C/C++. — СПб.: ФТК СПб. гос. политехн. ун-т, 2009. — 200 с. — С. 35.

[53] Степанов А.Н. Курс информатики для студентов информационно-математических специальностей. (Серия «Учебник для вузов»). — СПб.: Питер, 2018. — 1088 с.

[54] Мейер Б. Почувствуй класс. Учимся программировать хорошо с объектами и контрактами. Перевод с англ. / Под ред. В.А. Биллига. — М.: ИНТУИТ, БИНОМ, 2011. — 775 с.

[55] Дейкстра Э. Заметки по структурному программированию // Дал У., Дейкстра Э., Хоор К. Структурное программирование. / Пер с англ. / Под ред. Э.З. Любимского, В.В. Мартынюка. — М.: Мир, 1975. — 247 с. – С. 7–97.

[56] Пышкин Е.В. Структурное проектирование: основание и развитие методов. С примерами на языке C++: Учебное. пособие. – СПб.: Изд-во Политехнического ун-та, 2005. – 324 с.

[57] Беляков Е. Новый алгоритм: раздевайся и быстро ложись спать. Диалог на языке «Дракона» // Учительская газета. 13 марта 2001. № 10. — С. 16.

[58] Тюгашев А.А. Графические языки программирования и их применение в системах управления реального времени. – Самара: Изд-во СНЦ РАН, 2009. — 98 с. — С. 50.

[59] Ashcroft E, Manna Z. The translation of “goto” programs into “while” programs // Proceedings of IFIP Congress, Ljubljana, Yugoslavia, 1971. — pp. 250-255.

[60] Йодан Э. Структурное проектирование и конструирование программ. / Пер. с англ. / Под ред. Л.Н. Королева. — М.: Мир, 1979. – 415 с. — С. 252.

[61] Полич В.В. Проблемы военного образования и науки (социально-философский анализ). — Профессиональное образование в современном мире. № 2(17), 2015. — https://bit.ly/2YbtcYW.

[62] Роскосмос. Выводы аварийной комиссии. 12.12.2017. — https://www.roscosmos.ru/24451.

[63] Окулова Л. П. Проектирование образовательного процесса в соответствии с требованиями педагогической эргономики // Вестник Наука и практика (29 мая 2012 — 31 мая 2012) — Мат. конф. «Инновации и научные исследования, а также их примен. на практике». Варшава, — http://xn--e1aajfpcds8ay4h.com.ua/pages/view/730.

[64] Лозовская М. Э., Васильева Е. Б., Клочкова Л. В., Яровая Ю. А., Степанов Г. А. Новый вектор в преподавании фтизиатрии студентам-педиатрам // Туберкулёз и болезни лёгких, Том 97, № 5, 2019. — С. 73, 74.

[65] Фокин Ю. Г. Теория и технология обучения: деятельностный подход: учебное пособие для студентов высших учебных заведений. — 3-е изд., испр.. — М.: Издательский центр «Академия», 2008. — С. 233. — 240 с.

[66] Косова А.Е. Язык ДРАКОН как средство повышения качества профессиональной компетентности // Современное образование: повышение профессиональной компетентности преподавателей вуза —, гарантия обеспечения качества образования: науч.-метод. конф. 1-2 фев 2018. — Томск: изд Томск. ун-та сист. упр. и радиоэлектрон., 2018. — 311 с. — С. 89-91 .

[67] Пичкалев А.В. Разработка программного обеспечения робототехнических комплексов с использованием графического языка ДРАКОН // Робототехника и искусственный интеллект:, мат-лы VII Всеросс. науч.-тех. конф. (г. Железногорск, 11 дек 2015) / под науч. ред. В.А. Углева. – Красноярск: Сибирский федер. ун-т, 2016. – 194 с. — С. 90-94.

[68] Безель Я. Б. Можно ли улучшить работу ума? Новый взгляд на проблему. Размышления над новой книгой // Вестник Российской академии наук, том 73, № 4, 2003. — С. 365.

[69] Пароннджанов В. Д. Учись писать, читать и понимать алгоритмы. Алгоритмы для правильного мышления. Основы алгоритмизации. — М.: ДМК Пресс, 2012, 2014, 2016. — 520 с.

[70] Пароннджанов В. Д. Графический синтаксис языка ДРАКОН // Программирование. — 1995. Т. 3. — С. 45-62. — http://drakon.su/_media/video_i_prezentacii/graphical_syntax_.pdf.

[71] Дородницын А.А. Информатика: предмет и задачи // Вестник Академии наук СССР. 1985. №2. — С. 85-89.

[72] Кокошин А.А. Несколько измерений войны // Вопросы философии. 2016. №8. — С. 5-19.

[73] Hybridlab. Применение языка ДРАКОН в медицине (Видео с субтитрами на русском языке). — https://www.youtube.com/watch?v=SV1sKaCzuGU.

[74] Каралюс А. О внедрении языка ДРАКОН. — Доклад в Институте философии РАН. — https://www.youtube.com/watch?v=KEt7_M8Sojw.

СВЕДЕНИЯ ОБ АВТОРЕ

- Настоящий документ разработал по личной инициативе В.Д. Пароджанов
- Родился 25 июля 1938 года.
- Работал в Роскосмосе (в Научно-производственном центре автоматики и приборостроения имени академика Н. А. Пилюгина) почти 58 лет: с 1.03.1961г. по 31.01.2018г. и еще по договору полгода.
- Кандидат технических наук.
- Автор ряда книг по алгоритмам и языку ДРАКОН.
- За заслуги перед отечественной космонавтикой присвоено почетное звание «Ветеран космонавтики России».
- Автор алгоритмического языка ДРАКОН, который используется в ряде крупных космических программ: международный проект «Морской старт», космический буксир «Фрегат», ракета-носитель «Протон-М», семейство ракет-носителей «Ангара» и др.
- Пуски ракет-носителей и космических разгонных блоков, при создании которых использовался язык ДРАКОН, выполнялись с шести космодромов мира, расположенных на трех континентах и в Тихом океане.
- Язык ДРАКОН находит применение в народном хозяйстве, например, для программирования Сенсорного программируемого контроллера СПК 107 М01 фирмы ОВЕН в шкафу управления установки глубокой переработки широкой фракции легких углеводородов (ШФЛУ) на Южно-Балыкском газоперерабатывающем заводе (ГПЗ) компании "Сургутнефтегаз". Фракция ШФЛУ — продукт переработки попутного нефтяного газа (ПНГ) и газового конденсата https://youtu.be/_XOuhV_8N_I
- В настоящее время пенсионер, ведет общественный проект по развитию языка ДРАКОН в сети Интернет на сайте и форуме языка ДРАКОН и на других онлайн-площадках.

Пароджанов Владимир Даниелович. Награжден орденом «Знак Почета» и медалями

Приложение.
ИСТОРИЯ СОЗДАНИЯ МЕДИЦИНСКОГО ЯЗЫКА ДРАКОН

Медицинский язык ДРАКОН рекомендуется для использования в Медицинской службе Вооруженных Сил РФ и в военной медицине.

История создания языка такова. Литовские врачи прочитали мои книги и начали использовать язык ДРАКОН для наглядного и точного представления клинических алгоритмов, предварительно упростив его, выбросив часть функций и приспособив для медицинских целей.

Ознакомившись с их наработками, я пришел к выводу, что ДРАКОН как язык программирования врачам не нужен. Им нужно другое — язык для точного описания сложных и разветвленных действий и решений врача, позволяющий представить клинический алгоритм (жизнеритм) в наглядной графической форме. То есть в виде простой и удобной инструкции, пригодной для применения при профилактике, диагностике и лечении.

Можно сказать, что черновой медицинский вариант языка ДРАКОН родился в Литве по инициативе литовских врачей и ученых, которые вложили большой труд в его отработку и совершенствование.

Ученые и преподаватели Литовского университета наук здравоохранения (Lietuvos sveikatos mokslų universitetas) проанализировали «космический» ДРАКОН, убрали лишнее, внесли полезные добавления и применили для медицинских нужд.

В связи с этим я принял решение: опираясь на литовский опыт, создать новый алгоритмический язык, предназначенный специально для медицины. При этом решил сохранить «космическое» название ДРАКОН, которое уже получило известность как символ эргономичности и удобства. Например, в Википедии статья ДРАКОН (DRAKON) представлена на восьми языках http://bit.ly/2VOeRzc

Так появился «медицинский язык ДРАКОН», описанный в моей книге [4]. Книга является обобщением опыта литовских медиков и дальнейшим развитием идей когнитивной эргономики. Космическое происхождение медицинского языка сыграло важную роль, так как обеспечило тщательную отработку и надежность результатов, соответствующую высоким стандартам качества ракетно-космической отрасли.

С уважением,
Владимир Даниелович Паронджанов

Mobile +7-916-111-91-57
Viber +7-916-111-91-57
E-mail vdp2007@bk.ru
Skype vdp2007@bk.ru
DRAKON language website http://drakon.su
DRAKON language webforum http://forum.drakon.su