СОДЕРЖАНИЕ

ВВЕДЕНИЕ

ЧТО МЫ ЗНАЕМ ОБ АЛГОРИТМАХ?

> Многие думают, что алгоритмы нужны только программистам и математикам. Это не так. Алгоритмы могут пригодиться всем или почти всем. Начиная от врача и агронома и кончая топ-менеджером.

Почему? Потому что мы живем в мире алгоритмов, хотя зачастую не догадываемся об этом. Современная цивилизация – это цивилизация алгоритмов. Они окружают нас повсюду.

К сожалению, большинство людей не умеют читать, писать и понимать алгоритмы. Впрочем, это дело поправимое. Прочитав книгу, вы быстро получите нужные знания. Как известно, один рисунок стоит тысячи слов. К вашим услугам – четкие, кристально ясные и забавные рисунки. Сделанные так, чтобы читатель «посмотрел – и сразу понял!». Они помогут легко открыть заветную дверь в увлекательное царство алгоритмов.

В ЧЕМ ПРОБЛЕМА?

Трудность в том, что алгоритмы, как правило, очень сложны. Понять их непросто. Нужно изрядно попотеть, затратить много труда и времени.

А нельзя ли найти обходную дорогу и сэкономить время?

Конечно, можно! Секрет в том, что алгоритмы надо сделать *дружелюбными*. Это позволит превратить головоломки в наглядные алгоритмы-картинки, обеспечивающие быстрое и глубокое понимание. Глубина понимания сложных проблем – как раз то, чего всем нам (от студента до министра) ой как не хватает!

Почему алгоритмы трудны для понимания? Потому, что существующий способ записи алгоритмов (принятый во всем мире) выбран неудачно. Он устарел и превратился в досадное препятствие. Он удовлетворяет требованиям математической строгости, но не учитывает требования науки о человеческих факторах – эргономики. Этот устаревший способ упускает из виду психофизиологические характеристики человека-алгоритмиста. И тем самым затрудняет и замедляет работу с алгоритмами.

ЛЕГКИЕ ДЛЯ ПОНИМАНИЯ И УДОБНЫЕ ДЛЯ РАБОТЫ

> В книге излагается новый взгляд на алгоритмы. Мы покажем, что алгоритмы должны быть не только правильными, *но и дружелюбными*. То есть легкими для понимания и удобными для работы.

Этой благородной цели служат *эргономичные* алгоритмические языки. Они создают повышенный интеллектуальный комфорт, увеличивают продуктивность труда. С их помощью вы научитесь легко и быстро, затратив минимум усилий, решать сложнейшие проблемы. Вы сможете проектировать сложную деятельность и бизнес-процессы. Формализовать свои профессиональные знания. И составлять алгоритмы самостоятельно, без помощи программистов.

> Алгоритмы – важная часть человеческой культуры. Умение составлять алгоритмы позволяет улучшить работу ума. Несколько преувеличивая, можно сказать: «Алгоритмы – вторая грамотность!».

БЛОК-СХЕМЫ АЛГОРИТМОВ

Существует несколько способов для записи алгоритмов. Широкую известность получили блок-схемы алгоритмов. Они используются в системе образования для первоначального ознакомления с алгоритмами (см. например, [1–3]). Международная организация стандартизации *ISO* выпустила стандарт на блок-схемы *ISO* 5807–85 [4]. Ему соответствует отечественный стандарт ГОСТ 19.701–90 [5].

Тем не менее, многие специалисты убеждены, что блок-схемы – это вчерашний день. Можно сказать, что блок-схемы постигла печальная участь. Истина в том, что они обладают поистине удивительными достоинствами, которые остаются нераскрытыми. Без преувеличения можно сказать, что блок-схемы алгоритмов – это бесценный бриллиант, который по воле мачехи-судьбы до сих пор не огранен и не вставлен в золотую оправу.

Хотя имеются отдельные попытки приспособить блок-схемы к современным нуждам (язык *UML* и др.), однако в целом блок-схемы явно оказались на обочине бурно развивающегося процесса визуализации алгоритмов. А их громадные потенциальные возможности фактически не используются.

Данная книга посвящена блок-схемам. В ней изложен *новый подход* к решению проблемы блок-схем. Наш труд можно рассматривать как дальнейшее развитие плодотворной идеи изобретателя блок-схем Джона фон Неймана.

Мы покажем, что именно блок-схемы могут придать алгоритмам новую чарующую силу, небывалую наглядность и другие полезные свойства.

ДРАКОН-СХЕМЫ И ЯЗЫК «ДРАКОН»

> Блок-схемы нужно значительно улучшить, сделать математически строгими и эргономически привлекательными. Во избежание путаницы мы присвоили новым блок-схемам название «дракон-схемы».

Дракон-схемы отличаются от традиционных блок-схем, как небо от земли. Они позволяют построить семейство дружелюбных алгоритмических языков под общим названием ДРАКОН (Дружелюбный Русский Алгоритмический язык, Который Обеспечивает Наглядность) [6, 7].

Визуальный язык ДРАКОН обладает уникальными эргономическими характеристиками. Он позволяет легко и быстро создавать дружелюбные и наглядные алгоритмы.

ДРАКОН – оружие интеллекта. Он помогает ясно и четко мыслить. И значительно облегчает творческий процесс создания алгоритмов, делая его доступным для широкого круга работников.

КТО РАЗРАБОТАЛ ЯЗЫК ДРАКОН?

Язык ДРАКОН разработан совместными усилиями Федерального космического агентства России (Научно-производственный центр автоматики и приборостроения им. академика Н. А. Пилюгина, г. Москва) и Российской академии наук (Институт прикладной математики им. академика М. В. Келдыша, г. Москва).

ДРАКОН РОДИЛСЯ В КОСМИЧЕСКОЙ КОЛЫБЕЛИ

ДРАКОН возник как обобщение опыта работ по созданию космического корабля «Буран». На базе ДРАКОНа построена автоматизированная Технология проектирования алгоритмов и программ под названием «ГРАФИТ-ФЛОКС». Она успешно используется во многих крупных ракетно-космических проектах: «Морской старт», «Фрегат», «Протон-М» и др.

ДРАКОН В СИСТЕМЕ ОБРАЗОВАНИЯ

В 1996 году Государственный комитет по высшему образованию Российской Федерации включил изучение языка ДРАКОН в программу курса информатики высшей школы. Этот факт нашел отражение в официальном документе Госкомвуза под названием:

«Примерная программа дисциплины «Информатика» для направлений:
510000 – Естественные науки и математика
540000 – Образование
550000 – Технические науки
560000 – Сельскохозяйственные науки
Издание официальное.
М.: Госкомвуз, 1996. – 21 с.

Авторы программы: Кузнецов В. С., Падалко С. Н., Пароджанов В. Д., Ульянов С. А.

Научные редакторы: Падалко С. Н., Пароджанов В. Д.» [8].

Примечание. Этот документ официально узаконил изучение языка ДРАКОН в системе образования.

ЧТО ТАКОЕ ЭРГОНОМИЧНЫЙ?

В этой книге важную роль играет слово «эргономичный». Что оно означает? Чтобы не тратить лишних слов, мы сочиним краткий словарик, который, хотя и нарушает все каноны научной строгости, зато вполне понятен новичкам.

Словарик

Эргономика – наука о том, как превратить сложную и трудную работу в простую, легкую и приятную.

Когнитивная эргономика – наука о том, как облегчить и улучшить умственную работу.

Эргономичный – дружелюбный, легкий для понимания, удобный для работы.

Эргономичный алгоритм – дружелюбный, легкий для понимания алгоритм.

Эргономичный алгоритмический язык – дружелюбный, легкий для понимания и удобный в работе алгоритмический язык.

Эргономизация алгоритма – превращение недружелюбного алгоритма в дружелюбный.

Эргономизация алгоритмического языка – превращение недружелюбного алгоритмического языка в дружелюбный.

ЦЕЛЬ КНИГИ

> Наша цель – научить читателя самостоятельно создавать дружелюбные, то есть легкие для понимания алгоритмы. И показать, что это простое и даже приятное дело.
> На многочисленных примерах читатель убедится, что дружелюбные алгоритмы имеют огромные преимущества.

Чем понятнее алгоритм, тем легче уяснить его смысл. Чем прозрачнее смысл, тем лучше взаимопонимание между людьми. Чем меньше ошибок, тем выше производительность труда при разработке алгоритмов. Чем меньше усилий затрачивают алгоритмисты, тем быстрее они выполняют свою работу.

АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ

В этой книге мы не будем касаться вопросов программирования. И сосредоточим внимание исключительно на алгоритмах.

Алгоритмизация и программирование – разные вещи. Подчеркнем главное.

> Число людей, которым необходимо знать алгоритмы, во много раз превышает число людей, которым надо знать программирование.

Разумеется, современное общество нуждается в программистах. Но обучение программистов должно быть не массовым, а экономически обоснованным.

И наоборот, обучение алгоритмизации должно быть очень широким или даже массовым. Потому что алгоритмы нужны не только программистам, но и многим другим людям.

В обществе знаний во многих случаях возникает необходимость формализовать собственные процедурные профессиональные знания специалистов. Отсюда проистекает

> *Вывод.* Умение читать, писать и понимать алгоритмы – настоятельная необходимость. Такое умение должно стать частью профессиональной культуры специалистов почти всех специальностей. Эта мысль проходит красной нитью через всю книгу.

СОДЕРЖАНИЕ КНИГИ

Книга состоит из восьми частей.

Часть I (главы 1, 2) носит вводный характер. Приводятся забавные примеры алгоритмов, описывающих бытовую человеческую деятельность.

В части II (главы 3–16) изложен эргономичный алгоритмический язык ДРАКОН. Рассмотрены графический алфавит языка, алгоритмические структуры «силуэт» и «примитив». Представлены математические методы, позволяющие улучшить понятность алгоритмов. Показан богатый ассортимент визуальных (графических) цикличных алгоритмов, визуальная логика, системы реального времени и параллельные алгоритмы. Для удобства читателя изложение основных идей дается на наглядных примерах.

Часть III (главы 17–25) содержит большое число алгоритмов на языке ДРАКОН, взятых из практической жизни. Примеры демонстрируют универсальность языка, показывают широкий спектр его возможностей для различных отраслей и предметных областей. Сюда относятся медицина, промышленность, сельское хозяйство, торговля и многое другое.

В части IV (главы 26–29) читатель вкратце знакомится с математическими алгоритмами. Примеры демонстрируют работу с массивами, поиск данных и др.

В части V (главы 30, 31) даются заключительные рекомендации по созданию дракон-схем. Даны рекомендации по использованию алгоритмических структур «силуэт» и «примитив». Описывается метод дробления веток, позволяющий улучшить понятность алгоритмической структуры «силуэт».

В части VI (главы 32, 33) описывается компьютерная программа «конструктор алгоритмов». Она представляет собой рабочий инструмент, помогающий человеку придумывать и конструировать алгоритмы. Приводится формальное описание языка ДРАКОН.

Часть VII (главы 34–36) – наиболее сложная часть книги. Здесь даны теоретические основы языка ДРАКОН.

Часть VIII (главы 37–41) посвящена социальным, гуманитарным и культурным аспектам алгоритмизации. Обсуждается вопрос: как ликвидировать алгоритмическую неграмотность?

ГДЕ СКАЧАТЬ КОНСТРУКТОР АЛГОРИТМОВ?

Ответ дан на стр. 414.

В ДОБРЫЙ ПУТЬ!

Прочитав книгу, читатель сможет легко ориентироваться в империи алгоритмов. И убедиться, что язык ДРАКОН – удобное средство, помогающее создавать наглядные и понятные алгоритмы.

Конструктор алгоритмов – надежный помощник человека. Он умело подсказывает, как нужно составлять алгоритмы. Кроме того, он осуществляет 100%-е автоматическое доказательство правильности дракон-схем, гарантируя принципиальную невозможность ошибок визуального синтаксиса. Иными словами, конструктор алгоритмов полностью исключает ошибки графического синтаксиса.

Безошибочное проектирование графики дракон-схем – важный шаг вперед, повышающий производительность труда при практической работе.

> По мнению специалистов, управляющая графика ДРАКОНа является мощным инструментом, причем ее мощь легка в освоении и легко применима на практике.

Часть I

АЛГОРИТМЫ (ОБЛЕГЧЕННЫЙ МАТЕРИАЛ ДЛЯ ПЕРВОГО ЗНАКОМСТВА)

Глава 1

АЛГОРИТМЫ – ЭТО ОЧЕНЬ ПРОСТО!

§1. ПЕРВОЕ ЗНАКОМСТВО С АЛГОРИТМОМ

Предположим, мать говорит сыну: «Сходи в булочную. Купи батон белого и батон черного». Слова матери – алгоритм, показанный на рис. 1. Данный алгоритм нельзя признать удачным. Он не отвечает на вопрос: что делать, если в магазине нет хлеба.

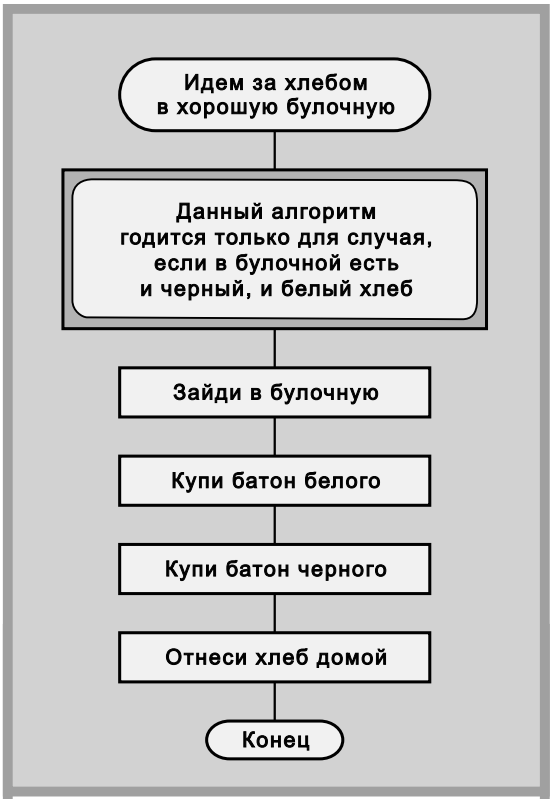


Рис. 1. Алгоритм «Идем за хлебом в хорошую булочную»

Чтобы исправить оплошность, нужно придумать другой алгоритм, который учитывает реальные условия. Предположим, в булочной черный хлеб водится всегда, а с белым случаются перебои. В таком случае мать могла бы сказать: «Купи батон белого и батон черного. Если белого не будет, возьми два черного». Словесная формулировка этого алгоритма показана на рис. 2, а блок-схема – на рис. 3.

К последнему алгоритму тоже можно придраться. Как быть, если в продаже нет черного хлеба? Чтобы учесть все варианты, мать должна дать сыну более сложную инструкцию: «Сходи за хлебом. Один батон белого и один черного.

Белого не будет – купи два черного. И наоборот. А если никакого нет, обойдемся».

Соответствующая блок-схема показана на рис. 4.

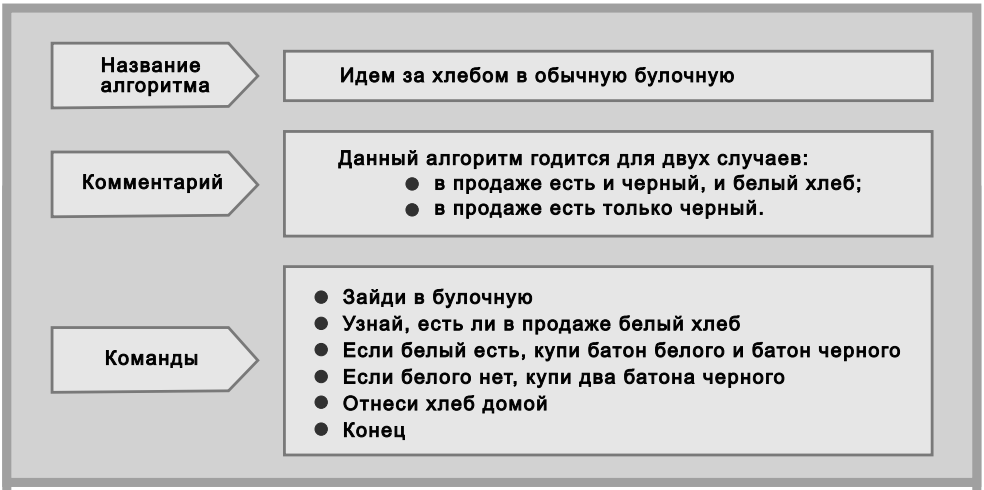


Рис. 2. Словесное описание алгоритма

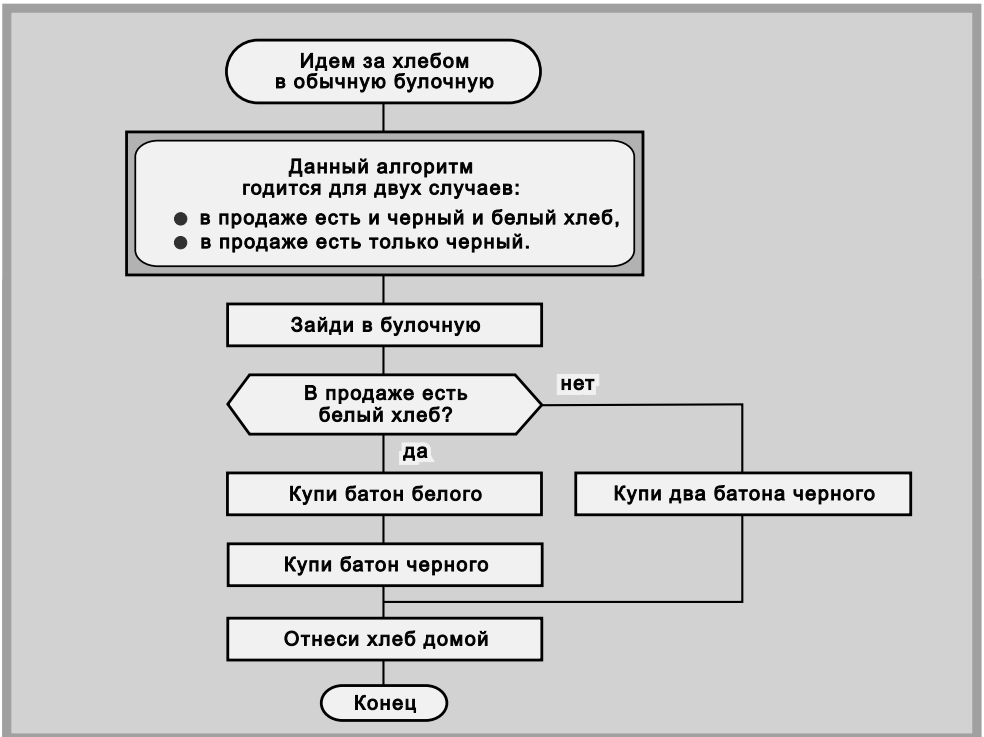


Рис. 3. Алгоритм «Идем за хлебом в обычную булочную». Словесное описание этого алгоритма дано на рис. 2

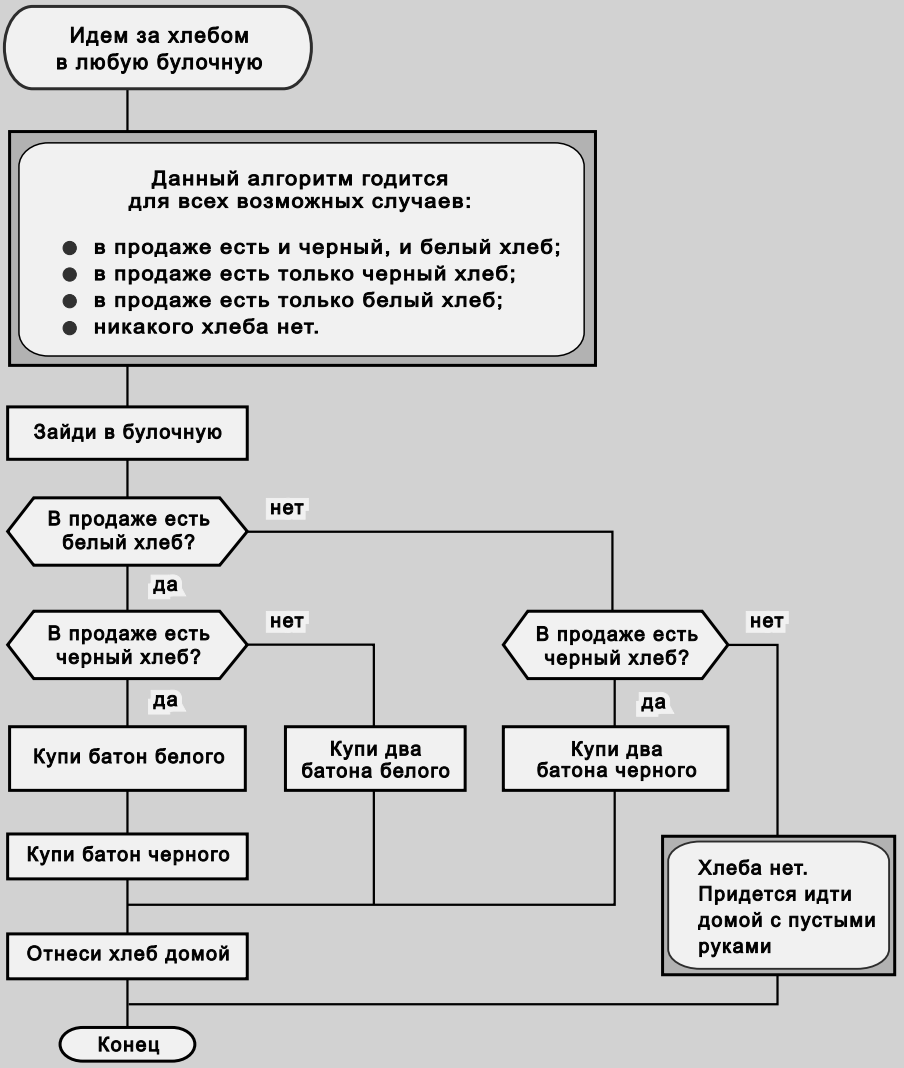


Рис. 4. Алгоритм «Идем за хлебом в любую булочную»

§2. СРАВНЕНИЕ АЛГОРИТМОВ

Взглянем на три алгоритма на рис. 1, 3, 4. Какой из них следует считать хорошим, а какой – плохим? Алгоритм считается хорошим, если он правильно работает при всех реальных условиях. В противном случае алгоритм считается плохим.

Применим это правило к нашим алгоритмам. Легко видеть, что только один алгоритм (рис. 4) будет работать при любых условиях.

В самом деле, если в булочной есть и белый, и черный хлеб, воображаемый «бегунок» побежит от начала к концу алгоритма по левой вертикальной линии.

Если белый есть, а черного нет, бегунок помчится чуть правее – через икону «Купи два батона белого».

Если белого нет, а черный есть, бегунок побежит еще правее – через икону «Купи два батона черного».

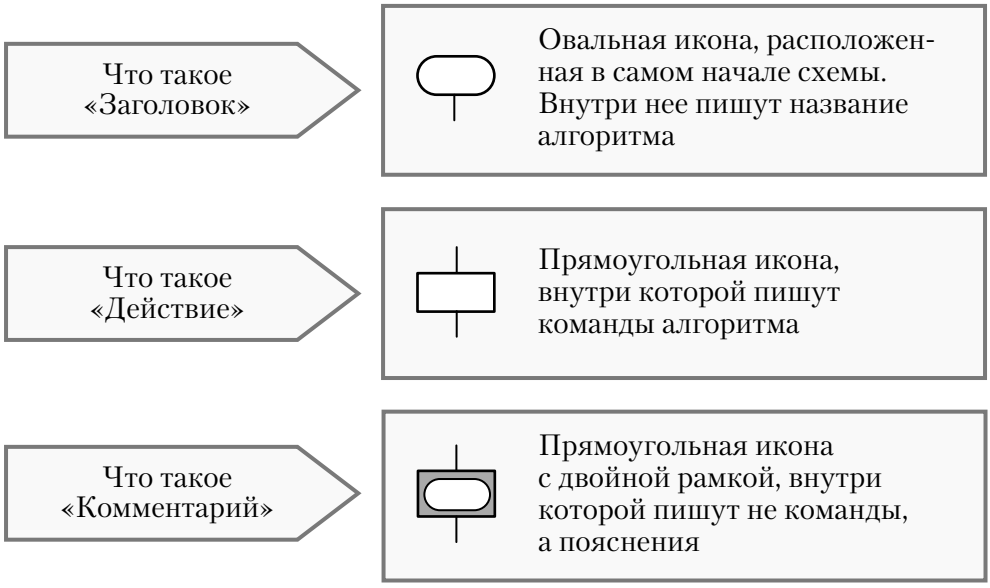
Наконец, если в булочной вообще нет хлеба, бегунок выберет крайний правый путь – через икону «Хлеба нет. Придется идти домой с пустыми руками».

Что же мы узнали? Алгоритм на рис. 4 – хороший алгоритм, так как он выполняет свою работу *при любых условиях*. И наоборот, алгоритмы на рис. 1 и 3 – это плохие, «неполноценные» алгоритмы.

Можно сказать и по-другому. Алгоритм на рис. 4 – правильный, безошибочный, работоспособный алгоритм. А схемы на рис. 1 и 3 не отвечают на вопрос: что делать, если в булочной нет хлеба. Значит, эти алгоритмы неработоспособны; они содержат ошибку. На рис. 4 ошибка исправлена.

§3. ЧТО ТАКОЕ ДРАКОН-СХЕМЫ? ЗАЧЕМ НУЖНЫ ИКОНЫ?

Блок-схемы, нарисованные по правилам языка ДРАКОН, называются *дракон-схемы*. Они состоят из простых рамок – *икон,* соединенных между собой. На рис. 1 таких икон семь. Верхняя называется «заголовок». Та, что пониже – «комментарий». Следующие четыре – «действие». А самая нижняя – «конец».



Что такое «Конец»

Маленькая овальная икона, расположенная в самом конце алгоритма. Внутри нее пишут слово «Конец»

§4. АЛГОРИТМ

Каждый человек, находясь на работе, ведет себя отнюдь не случайно. Он стремится к четко определенной цели и выполняет нужные для этого действия. Таким образом, любую работу (за исключением сложных творческих операций) можно рассматривать как хорошо определенный каскад действий, то есть алгоритм. Нетрудно заметить, что алгоритмы окружают нас повсюду.

Алгоритм – последовательность шагов, ведущих к цели. Можно сказать и по-другому. Алгоритм – последовательность команд, помогающих решить задачу.

Что такое алгоритм

Это последовательность действий, ведущих к поставленной цели

Команды алгоритма можно записать по-разному, например:

- в виде дракон-схемы;
- в виде текста.

Чтобы убедиться в этом, взгляните на рис. 5.

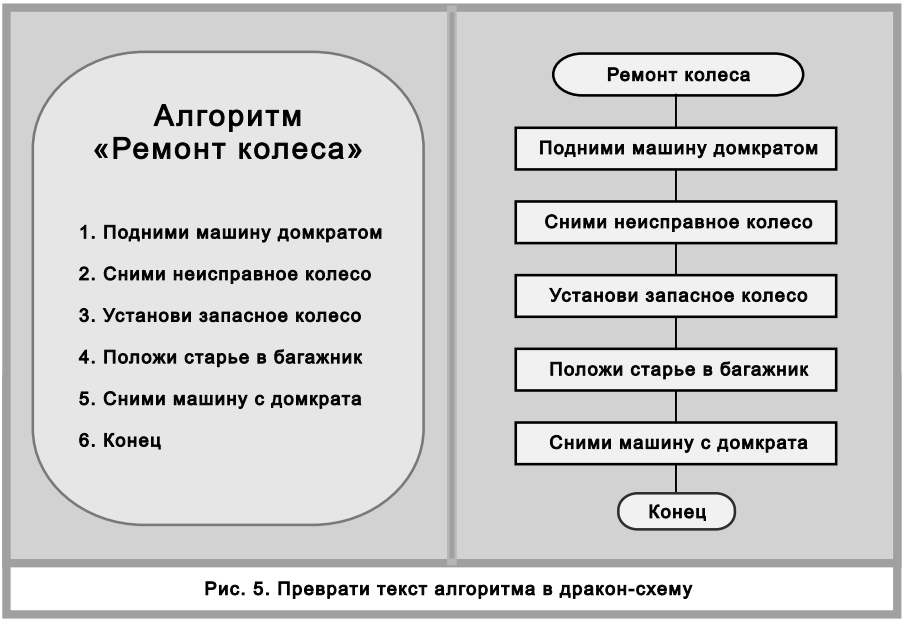


Рис. 5. Преврати текст алгоритма в дракон-схему

§5. КАЖДАЯ КОМАНДА АЛГОРИТМА ДОЛЖНА ВЫПОЛНЯТЬСЯ ТОЧНО И ОДНОЗНАЧНО

Рассмотрим алгоритм на рис. 6. Предположим, алгоритм выполняет робот.

Робот – типичный буквоед. Обычно педантов и буквоедов не любят. Но в мире алгоритмов свои законы. Там буквоеды в большом почете. Чтобы робот понял алгоритм, каждая команда должна быть написана «по-буквоедски». Точно и однозначно.

На рис. 6 алгоритм написан именно так – точно и однозначно. Там ясно сказано – надо купить килограмм картошки и кочан капусты. Это значит, что любые другие покупки запрещены.

А что мы видим на рис. 7? Тут ситуация иная. Команда «Купи продукты» – плохая, неоднозначная команда. Ее невозможно исполнить. Потому что нигде не сказано, какие именно продукты нужно купить.

Такие умолчания в алгоритмах недопустимы. Они рассматриваются, как ошибки.

Выполните упражнение на рис. 8.

§6. ИКОНА «ВОПРОС»

В верхней части рис. 9 изображена икона «вопрос». Она называется так потому, что внутри нее пишут «да-нетный вопрос». То есть вопрос, на который можно ответить либо «да», либо «нет». Все другие ответы запрещены.

Рис. 6. Хороший алгоритм. Здесь ясно сказано: нужно купить килограмм картошки и кочан капусты

Рис. 7. Плохой алгоритм. Его нельзя выполнить, так как неизвестно, какие именно продукты нужно купить

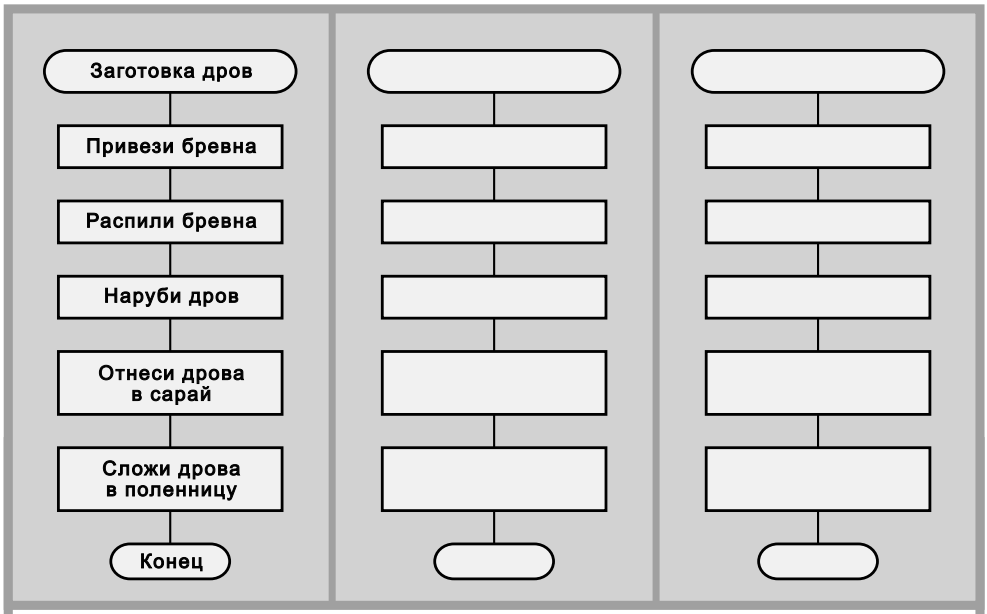


Рис. 8. Придумай два алгоритма по образцу, показанному слева

Вот примеры да-нетных вопросов: утюг сломался? Вася купил хлеб? Поезд пришел? Преступника арестовали? Тетя приехала? «Спартак» выиграл? Эта лужа больше, чем та? На улице температура выше нуля?

Икона «вопрос» имеет один вход сверху и два выхода: вниз и вправо. Выход влево запрещен и никогда не используется.

Бывают случаи, когда нужно выбрать одно действие из двух. В этом случае удобно использовать икону «вопрос». При ответе «да» выполняется одно действие, при «нет» – другое. Икона «вопрос» нужна, чтобы сделать в алгоритме развилку. В этом легко убедиться, взглянув на рис. 9.

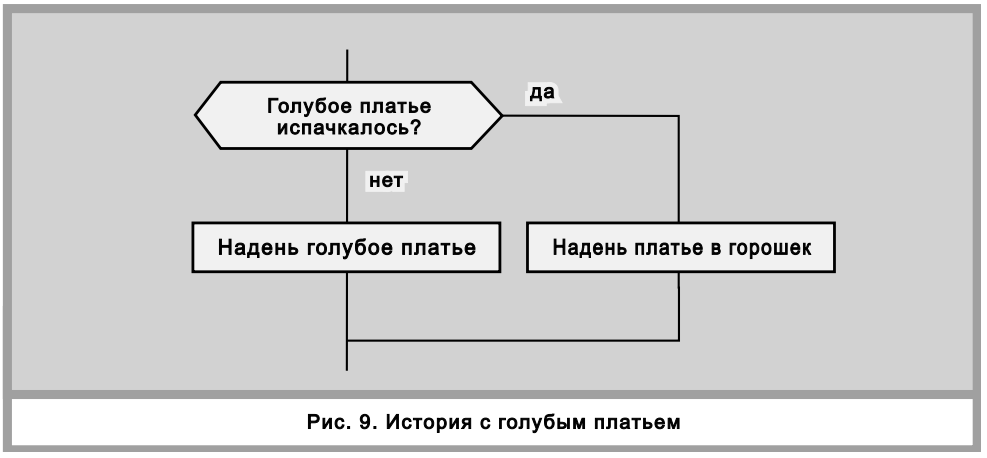
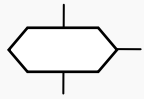


Рис. 9. История с голубым платьем

Иногда в алгоритме надо сделать не одну, а много развилок. Для этого используют несколько икон «Вопрос» (рис. 4).

Что такое «Вопрос»



- Ромб, у которого отрезаны верхний и нижний углы.
- Внутри фигуры пишут да-нетный вопрос, то есть вопрос, на который можно ответить либо «да», либо «нет».

§6. ВЫВОДЫ

1. Умение выразить свою мысль в виде алгоритма имеет много преимуществ. Такое умение пригодится почти каждому – ведь оно делает интеллект более гибким и эффективным.
2. Алгоритмы играют огромную роль в жизни общества. Они оказывают заметное влияние на эффективность экономики, уровень жизни, а также на умственное развитие цивилизации.

Глава 2

АЛГОРИТМЫ И ПРОЦЕДУРНЫЕ ЗНАНИЯ

§1. ПРОЦЕДУРНЫЕ И ДЕКЛАРАТИВНЫЕ ТЕКСТЫ

Давайте совершим экскурсию в библиотеку. На полках стоят тысячи книг. Толстых и тонких. Художественных и научных. Всяких.

Все тексты в книгах можно разделить на две части:

- процедурные,
- декларативные.

Чтобы уяснить, чем они различаются, обратимся к таблице 1.

Таблица 1

Это процедурный текст	Это декларативный текст
Скинь мантилью, ангел милый, И явись, как яркий день! Сквозь чугунные перилы Ножку дивную продень!	По синим волнам океана, Лишь звезды блеснут в небесах, Корабль одинокий несется, Несется на всех парусах.

Текст в левом столбце при желании можно превратить в набор команд:

1. Скинь мантилью.
2. Явись, как ясный день.
3. Продень ножку сквозь перила.

Текст в правом столбце такому преобразованию не поддается. Его невозможно превратить в команды.

Рассмотрим еще один пример (табл. 2).

Таблица 2

Это процедурный текст	Это декларативный текст
Сходи к бабушке, возьми у нее телевизор и отнеси в ремонт.	Телевизор был старый, покрытый толстым слоем пыли. Видимо, им уже давно не пользовались.

Левый текст без труда превращается в команды:

1. Сходи к бабушке.
2. Возьми у нее телевизор.
3. Отнеси его в ремонт.

Правый текст имеет принципиально иную логическую структуру. Как ни старайся, как ни хитри, а в команды его не превратишь!

Если текст содержит достоверные и обоснованные сведения, такие сведения называются знанием.

§2. КАК ПРЕВРАТИТЬ ПРОЦЕДУРНОЕ ЗНАНИЕ В АЛГОРИТМ?

В верхней части рис. 10 находится текст, содержащий процедурное знание. Это знание можно превратить в дракон-схему (алгоритм).

Пробежим взглядом по схеме сверху вниз. Мы обнаружим, что алгоритм нарисован не хаотично, а упорядоченно. Тут действует

> *Правило*. Чем ниже нарисована икона, тем позже выполняется записанная в ней команда. (Для краткости: «Чем ниже – тем позже»).

Данное правило выполняется во всех алгоритмах.

Здесь нас поджидает сюрприз. Переведем взгляд со схемы на исходный текст. Налицо разительный контраст – в тексте данное правило не соблюдается! Таким образом, алгоритм сильно отличается от исходного текста. В чем отличие?

А вот в чем – порядок записи команд коренным образом изменился. Это хорошо видно на рис. 11.

- Первая команда текста (*Сходи к бабушке*) в алгоритме стала не первой, а второй.
- Вторая команда текста (*возьми у нее телевизор*) переехала совсем в другое место.

- Третья команда текста (*и отнеси его в ремонт*) стала предпоследней.
- Четвертая команда текста (*На случай, если бабушки не будет дома*) сохранила свое место – осталась четвертой.
- И наконец, самое удивительное! Последняя команда текста (*возьми с собой ключ от ее квартиры*) перескочила из конца в начало. В алгоритме она стала самой первой.

Как можно объяснить эти чудесные превращения? Ответ довольно прост.

Исходный текст, представленный на рис. 10 и 11, написан на естественном языке. В этом языке закон «Чем ниже – тем позже» не действует. А в алгоритме ситуация обратная. Там этот закон – святая святых и обязательно исполняется.

Отсюда вытекает

Правило
преобразования процедурного текста в алгоритм

Чтобы превратить процедурное знание, записанное на естественном языке, в дракон-схему, необходимо:

- расчленить исходный текст на фрагменты, чтобы в каждом фрагменте содержалась одна команда;
- записать команды в виде упорядоченной последовательности согласно принципу «Чем ниже – тем позже»;
- добавить иконы «заголовок» и «конец»;
- при необходимости внести дополнительные команды, устраняющие пробелы исходного текста и превращающие дракон-схему в логически законченный алгоритмический рассказ.

Примечание. На рис. 10 и 11 дополнительными командами являются:

- Позвони в дверь.
- Открой дверь своим ключом.
- Войди в квартиру.

§3. МАТЕМАТИЧЕСКИЕ И «БЫТОВЫЕ» АЛГОРИТМЫ

Во многих учебниках описаны математические алгоритмы. Это хорошо, но мало. Разумеется, студенты должны назубок знать математику. Но в жизни бывают неожиданности. Некоторые студенты, овладев математикой, затрудняются составлять алгоритмы, в которых отсутствуют математические формулы.

Чтобы избежать подобных неприятностей, студенты должны уметь распознавать «нематематические» алгоритмы, которые постоянно встречаются в жизни.