

ИНФОРМАТИКА



Азбука
Роботландии.
От основ до самых
до...

с. 4

Устройства
ввода и вывода.
Теория
относительности.

с. 24

Хит ЕГЭ:
системы
логических
уравнений.

с. 30

Ха! Хаскель!
Серебряная пуля
современного
программирования?

с. 36

О НОМЕРЕ

► Этот номер получился очень содержательным и разнообразным. Большую часть материалов в нем безо всякой натяжки можно назвать эксклюзивными – только у нас и только для нас. Описание исполнителей “Азбуки Роботландии” – уникальный материал, статья о системах логических уравнений – на самую злобу, пусть уже и прошедшего дня (но ведь и оглянуться не успеем, как и новый ЕГЭ повиснет на носу), материал о языке Хаскель – мы давно мечтали его сделать.

Читайте с удовольствием.

В НОМЕРЕ

- 3** ОЛИМПИАДЫ
 - Бои по правилам
- 4** НАЧАЛКА
 - Азбука Роботландии
- 24** МНЕНИЯ
 - Устройства ввода и вывода.
Теория относительности
- 30** ЕГЭ
 - Системы логических уравнений
- 36** СЕМИНАР
 - Haskell: серебряная пуля современного программирования?
- 48** ЗАНИМАТЕЛЬНЫЕ МАТЕРИАЛЫ ДЛЯ ПЫТЛИВЫХ УЧЕНИКОВ И ИХ ТАЛАНТЛИВЫХ УЧИТЕЛЕЙ
 - “В мир информатики” № 168
- ИНФОРМАЦИЯ
 - Педагогический университет “Первое сентября” предлагает очно-заочные курсы повышения квалификации для педагогов Москвы и Московской области
 - Фестиваль “Учительская книга” 31 октября — 3 ноября

НА ДИСКЕ



ЭЛЕКТРОННЫЕ МАТЕРИАЛЫ:

- | Демонстрационные версии “Азбуки Роботландии” — учебника и методички (по три полных урока)
- | Демонстрационные версии практикумов по исполнителям “PC-1” и “Агент РБ”
- | Пакет “Хиты Роботландии” (версия для DOS)
- | Программа для решения систем логических уравнений
- | Исходные тексты программ к статье о языке Хаскель и интерпретатор языка (установочный пакет)
- | Презентации

ИНФОРМАТИКА А

ПОДПИСНЫЕ ИНДЕКСЫ: по каталогу “Роспечати”: 32291 (бумажная версия), 19179 (электронная версия); “Почта России”: 79066 (бумажная версия), 12684 (электронная версия)

<http://inf.1september.ru>

Учебно-методический журнал для учителей информатики
Основан в 1995 г.
Выходит один раз в месяц

РЕДАКЦИЯ:

гл. редактор С.Л. Островский
редакторы

Е.В. Андреева,
Д.М. Златопольский
(редактор вкладки
“В мир информатики”)

Дизайн макета И.Е. Лукьянов
верстка Н.И. Пронская

корректор Е.Л. Володина
секретарь Н.П. Медведева
Фото: фотобанк Shutterstock

Журнал распространяется по подписке

Цена свободная

Тираж 3000 экз.

Тел. редакции: (499) 249-48-96

E-mail: inf@1september.ru

<http://inf.1september.ru>

ИЗДАТЕЛЬСКИЙ ДОМ “ПЕРВОЕ СЕНТЯБРЯ”

Главный редактор:
Артем Соловейчик
(генеральный директор)

Коммерческая деятельность:
Константин Шмарковский
(финансовый директор)

Развитие, IT и координация проектов:
Сергей Островский
(исполнительный директор)

Реклама и продвижение:
Марк Сартан

Мультимедиа, конференции и техническое обеспечение:
Павел Кузнецов

Производство:
Станислав Савельев

Административно-хозяйственное обеспечение:
Андрей Ушков

Главный художник:
Иван Лукьянов

Педагогический университет:
Валерия Арсланьян (ректор)

ГАЗЕТА ИЗДАТЕЛЬСКОГО ДОМА

Первое сентября – Е.Бирюкова

ЖУРНАЛЫ ИЗДАТЕЛЬСКОГО ДОМА

Английский язык – А.Громушкина

Библиотека в школе – О.Громова

Биология – Н.Иванова

География – О.Коротова

Дошкольное образование – М.Аромштам

Здоровье детей – Н.Сёмина

Информатика – С.Островский

Искусство – М.Сартан

История – А.Савельев

Классное руководство и воспитание школьников –

О.Леонтьева

Литература – С.Волков

Математика – Л.Рослова

Начальная школа – М.Соловейчик

Немецкий язык – М.Бузоева

Русский язык – Л.Гончар

Спорт в школе – О.Леонтьева

Управление школой – Я.Сартан

Физика – Н.Козлова

Французский язык – Г.Чесновицкая

Химия – О.Блохина

Школьный психолог – И.Вачков

УЧРЕДИТЕЛЬ:
ООО “ЧИСТЫЕ ПРУДЫ”

Зарегистрировано ПИ № ФС77-44341

от 22.03.2011

в Министерстве РФ

по делам печати

Подписано в печать:

по графику 15.08.2011,

фактически 15.08.2011

Заказ №

Отпечатано в ОАО “Чеховский

полиграфический комбинат”

ул. Полиграфистов, д. 1,

Московская область,

г. Чехов, 142300

АДРЕС ИЗДАТЕЛЯ:

ул. Киевская, д. 24,

Москва, 121165

Тел./факс: (499) 249-31-38

Отдел рекламы:

(499) 249-98-70

http://1september.ru

ИЗДАТЕЛЬСКАЯ ПОДПИСКА:

Телефон: (499) 249-47-58

E-mail: podpiska@1september.ru

Документооборот

Издательского дома

“Первое сентября” защищен

антивирусной программой

Dr.Web



Бои по правилам

28 июля в Таиланде завершилась XXIII Международная олимпиада школьников по информатике (IOI). От каждой страны в IOI может участвовать не более 4 школьников (исключением является страна-организатор, которая может выставить «вторую команду»). В этом году в олимпиаде приняли участие 307 школьников из 80 стран мира. Участники олимпиады пишут программы на одном из двух доступных языков программирования (Free Pascal или C++), после чего отсылают программу для автоматической проверки.

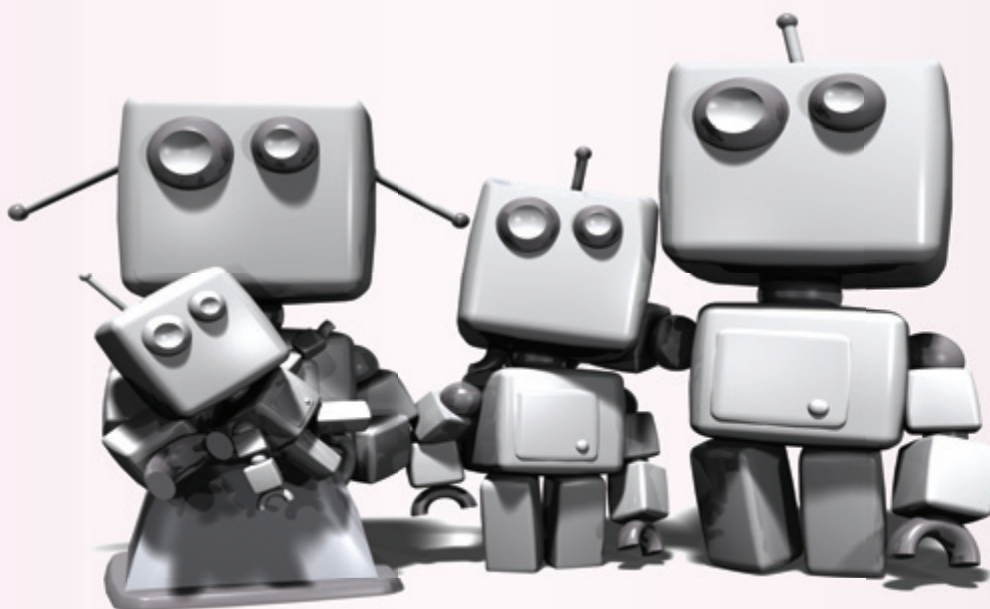
► Уже второй год олимпиада проходит по существенно измененным правилам. Если раньше участник не получал информации о результатах проверки отправленного им решения, то сейчас такая информация участникам предоставляется. Участник может узнать результат проверки отправленного им решения на полном наборе тестов и в соответствии с полученной информацией планировать дальнейшие действия. Сами тесты, конечно, остаются участнику не известными. В зачет идет лучшая из попыток. Заметим, что подобная система проверки не всегда облегчает жизнь участникам. При наличии в программе ошибки (особенно неизвестной ошибки) можно потратить все время тура на отладку одной задачи, которая вроде бы решена, и так и не приступить к решению других задач.

В отличие от 2010 года все задачи относились к традиционной олимпиадной тематике и имели полное решение. Дадим краткую характеристику задачам олимпиады. Задача garden по тематике похожа на задачу быстрого возведения в степень. При ее решении применяется такая идея, как быстрый поиск неперiodической части и периода в циклическом процессе. Задача gate (самая сложная задача первого тура) предполагает знание структур данных и техники хеширования. Задача github использует технику линейного прохождения массива с помощью нескольких указателей. Правда, обычно использу-

ются два указателя, а в данной задаче необходимы три. Задача второго тура crocodile решается с помощью модифицированного алгоритма Дейкстры поиска кратчайшего пути в графе. Задача elephants решается с помощью достаточно сложных структур данных (например, таких, как Декартово дерево по неявному ключу). В задаче ragots участники были введены в заблуждение формулировкой условия — из нее вроде бы следовало, что задача не имеет точного решения и оцениваться будут продвижения в качестве приближенных решений. На самом деле задача решалась с помощью комбинаторных методов, но поняли это только 7 участников.

По результатам соревнований было присуждено 27 золотых, 49 серебряных и 78 бронзовых медалей. Абсолютным чемпионом третий год подряд стал Геннадий Короткевич, десятиклассник из Гомеля. Для Геннадия это уже шестая международная олимпиада и пятая подряд золотая медаль. В этом году победитель набрал полный балл. Третье место занял россиянин Павел Кунявский из Саратова, а пятое — Александр Тимин из СУНЦ МГУ, Москва. Впервые за много лет сразу два россиянина оказались в первой десятке. Два других россиянина — Дмитрий Егоров и Егор Суворов (оба петербуржцы) — получили серебряные медали. Полные результаты можно посмотреть здесь: http://158.250.33.215/~ejudge/player_all.html.

В неофициальном командном зачете по набранным в сумме баллам победителем стала команда Китая, на втором месте — команда России, на третьем — команда США.



Азбука Роботландии

Предисловие

А.А. Дуванов,
г. Переславль-Залесский,
kurs@robotland.pereslavl.ru
Н.Д. Шумина,
г. Тверь,
nshumilina@yandex.ru

► “Азбука Роботландии” — это проект, запущенный в 2010/2011 году. Цель — создать четырехгодичный курс информатики для начальной школы, отвечающий, с одной стороны, современному уровню развития информатики (как по интерфейсу самого УМК курса, так и по его содержанию), а с другой, работающий на опережение за счет опоры на фундаментальные основы информатики и развитие алгоритмического мышления обучаемых.

Учебный курс по информатике базисным учебным планом не предусмотрен, но необходимость в систематическом обучении информатике младших школьников существует. Формирование основ информатики, способствующих развитию мышления школьника (в частности, развитию абстрактного, логического, алгоритмического мышления), поможет детям не только в успешном формировании ИКТ-компетентности, но и в успешном освоении предметов

школьной программы, а также в будущем профессиональном самоопределении.

О необходимости отдельного курса информатики говорится и в “Программе формирования универсальных учебных действий у обучающихся на ступени начального общего образования”, являющейся частью ФГОС начального общего образования. В этой Программе предлагается вариант организации систематического обучения младших школьников основам информатики и информационных технологий, которым и воспользовались авторы курса “Азбука Роботландии”: “Освоение умений работать с информацией и использовать инструменты ИКТ может входить в содержание факультативных курсов, кружков, внеклассной деятельности школьников”.

Факультативный курс (или кружок) “Азбука Роботландии” может быть отнесен к внеурочным занятиям научно-познавательного направления.

В статье описывается процесс работы над проектом, при котором создание курса совмещалось с его преподаванием в школах, и предьявляются результаты первого года — построенный учебно-методический комплекс.

На диске, сопровождающем эту статью, записаны материалы, обеспечивающие проведение *трех первых уроков курса*:

- Учебник.
- Рекомендации по проведению уроков.
- Иллюстративные материалы к урокам (презентации).
- Электронные заготовки бумажных дидактических материалов.

Кроме того, в качестве подарка от Роботландии на диск записан пакет со старым ДОС-продуктом “Хиты Роботландии”.

Роботландия? Где-то я слышал это слово!

Курс с таким названием вошел в историю школьной информатики как образец комплексного педагогического продукта, удачно соединяющего в себе идейный фундамент, учебник для школьника, программное обеспечение и рекомендации для учителя.



Несмотря на солидный возраст этой системы (20 лет!) и безнадежно устаревший досовский интерфейс, “Роботландию” *до сих пор используют в школах*, и это говорит о высоком качестве этого педагогического продукта, ориентированного не на изучение быстротечного программного обеспечения, а на изучение фундаментальных основ информатики, инвариантных по отношению к смене компьютерных поколений.

Успех “Роботландии” в значительной степени определился и тем, что курс рождался на реальных школьных уроках, а не в стенах академической лаборатории. Даже рабочие места сотрудников располагались в школьном компьютерном классе. То, что вечером обсуждалось разработчиками, на следующий день работало с детьми в классе и корректировалось по ходу дела. Обсуждались и уточнялись как общие методические линии, так и конкретные компьютерные программы, оттачивалась их функциональность, тщательно продумывались интерфейс и дизайн.

Старый ДОС-пакет “Хиты Роботландии”, включающий в себя саму “Роботландию”, пакеты: “Конструктор”, “Зимние вечера” и “Азы программирования”, записан на диск, сопровождающий этот материал.

Азбука Роботландии — продолжаем традиции

Разработка продукта в процессе его эксплуатации

Работа над “Роботландией” непосредственно в школьном классе объяснялась очень просто — у одноименного предприятия тогда просто не было отдельного помещения. Сотрудники были *вынуждены* работать в уголке под аккомпанемент урока, который один из них вел в этом же самом классе. Эти трудности и неудобства оказались счастливыми — дети не позволяли разработчикам отрываться от своих возможностей и потребностей, и откровенно голосовали на уроках своим интересом или своею скукою.

Вот она, плодотворная нива для выращивания курсов!

Когда возникла идея создания “Азбуки Роботландии”, мы решили и этот курс возделывать в школьном классе. На виду учителей, детей и их родителей.

Ставка была сделана на:

- разработку продукта в процессе его эксплуатации;
- создание современного комплексного педагогического обеспечения;
- упор на фундаментальные азы информатики и развитие мышления школьника.

Структура нового курса

В “Азбуке” запланировано 4 ключевых раздела, соответствующих четырем годам обучения:

- 1-й год. **Компьютер**
- 2-й год. **Информация**
- 3-й год. **Алгоритмы**
- 4-й год. **Интернет**

Таким образом, “Азбука Роботландии” позиционируется как курс, закладывающий основы компьютерной, информационной, алгоритмической и коммуникационной грамотности младших школьников.

В настоящий момент создан учебно-методический комплекс первого года обучения “Компьютер”, включающий в себя:

- электронный интерактивный учебник-лабораторию (интерактивные тексты, тренажеры, практикумы, зачеты, вопросы и домашние задания);
- электронный сборник поурочных методических рекомендаций;
- сборник поурочных иллюстративных материалов (презентаций);
- сборник дидактических раздаточных материалов.

Упор на фундаментальность

Задача курса по-роботландски очевидна: помочь малышам освоить азы информатики.

Отметим: фундаментальные азы, те, которые устойчивы по отношению к бурным переменам современного общества и которые способны подготовить детей к информационным реалиям на много лет вперед, то есть подготовить к жизни в обществе, об устройстве которого сейчас ничего не известно.

Задача трудная, особенно если первый год курса мы решили посвятить компьютеру, ведь важно на первом этапе освоить базовый инструмент современной информатики. Освоить конкретно, на практике, но фундаментально, даже в той части, которая касается изменчивого интерфейса с учетом современных тенденций его развития.

Интерфейс современных приложений динамичен, как с функциональной, так и с визуальной точки зрения. Визуально такой интерфейс повторяет привычную физику реального мира. Например, реальные предметы не исчезают мгновенно в одном месте и не возникают ниоткуда в другом (нуль-транспортировка пока еще встречается только у фантастов). Современный интерфейс использует для имитации физических свойств экранных объектов такой прием, как “плавный переход” (постепенное растворение, раздвижка/схлопывание с конечной скоростью и т.п.). Даже в арсенале современного Веба сегодня распространены такие интерфейсные шаблоны, как прямое (контекстное) редактирование данных, перетаскивание, всплывающие контекстные инструменты и приглашения, оверлеи, инлеи, виртуальные страницы, автозаполнение, управление жестами и другие приемы, направленные на создание комфортной среды для пользователя, среды, которая не требует чтения длинных инструкций, становится понятной в ассоциации с привычной деятельностью в обычном реальном мире.

Мы хотим дать представление о составе компьютера, о принципах его работы, и мы хотим, чтобы дети освоили современный интерфейс на практике.

Но самое главное — мы хотим научить детей мыслить алгоритмически, ведь основа информатики — это теория и практика составления *алгоритмов* для обработки информации.

Развитие алгоритмического мышления — наша основная цель, она формулировалась в Роботландии, и к ней мы будем стремиться в новом курсе.

Заметим, мы не хотим подменять другие формы мышления алгоритмическим!

Мы хотим сформировать алгоритмическое мышление *наряду с другими формами мышления*. Это наша специфическая образовательная задача.

Алгоритмическое мышление — это способность облечь абстрактную идею в последовательность конкретных шагов, необходимых для ее воплощения на практике.

Алгоритмическое мышление полезно человеку не только при работе с компьютером, оно полезно в любом (самом творческом) деле, ибо учит превращать абстракцию в реальность.

Художник мечтает нарисовать красивый пейзаж с видом на озеро. Он может мечтать об этом сколько угодно, если... Если не начнет мыслить алгоритмически! Он должен облечь свою абстракцию в конкретные формы (выбрать натуру, продумать композицию, освещение, цвет, тональность...). Он должен, наконец, встать с дивана и сделать что-то алгоритмическое, выражаемое глагольными формами: умыться, пойти, найти, купить, собрать, организовать, нарисовать.

Конечно, мы хотим воспитывать по ходу дела и при любой к тому возможности и то, что обозначают оборотом “информационная компетентность”. То есть мы хотим сформировать у детей привычку при решении задач обращаться к адекватному инструменту (не забивать гвозди микроскопом, но находить молоток с удобной ручкой).

Наконец, мы хотим способствовать общему развитию ребенка (это задача любого предмета). Мы хотим привить любовь к чтению, письму, думанию и учению. Мы хотим способствовать формированию привычки находить точные формулировки, уметь корректно вести спор, отстаивать свои позиции.

Кроме всего этого, мы хотим еще двух важных вещей:

- раскрытия индивидуальности;
- проявления коллективизма.

Воспитанию этих качеств будут способствовать выполнение творческих домашних заданий и учебные командные игры в классе.

Задачи курса для первого класса

Мы хотим познакомить первоклассников с базовым инструментом современной информатики — компьютером.

Это основная тема курса в первом классе.

Более конкретно задачи первого года обучения можно сформулировать так:

- Состав и функционирование компьютера. Многообразие устройств, подключаемых к компьютеру.
- Введение в компьютерный интерфейс, реализуемый с помощью физических устройств (мышь, клавиатура, тачпад, сенсорный экран) и системы экранных объектов, предназначенных для взаимодействия с программным обеспечением (значки, окна, курсоры, меню).

Кроме того, дети освоят текстовое редактирование на базе учебного (строчного) редактора, снабженного системой контроля выполнения заданий.

Однако что бы мы ни рассказывали детям, какие действия ни учили бы выполнять на компьютере, мы будем постоянно “гнать” свою линию: “фор-

мировать алгоритмическое мышление”, и будем связывать дела компьютерные с делами от них далекими, показывая, что разные идеи имеют общие корни, как, например, иконка на мониторе и пиктограмма на дверях туалета.

Следует особо отметить, что наряду с неформальными алгоритмами уже на первом году обучения вводятся и формальные — *дети собирают, запускают и отлаживают программы для роботов на экране компьютера.*

Методика

Принцип максимума

Методика курса учитывает, конечно, возрастные особенности младших школьников, но пытается достичь при этом **максимума возможного в развитии детей**, делая ставку на ассоциативные связи и выстраивая последовательность маленьких шагов, связывающих простое и сложное, конкретное и абстрактное.

Особенности обучения

- Использование игровых форм обучения, как при работе за компьютером, так и при коллективной работе с учителем.
- Использование сюжетной основы при подаче нового материала.
- Продвижение к сложной деятельности или абстрактному понятию методом восходящей цепочки шагов: от простого — к сложному, от конкретно — к абстрактному.
- Обязательное подкрепление любой теории практической деятельностью.
- Обязательные целевые установки для каждого задания с понятной детям мотивацией.
- Непрерывный контроль знаний на каждом уроке (система тестов-зачетов) и зачетных занятий по итогам каждой темы.
- Поощрение проявления индивидуальности при выполнении творческих работ.
- Формирование навыков работы в коллективе.
- Предоставление детям для работы в школе (а при желании и дома) обучающих компьютерных сред, выполненных на базе современных дизайнерских и интерфейсных решений.
- Вовлечение в процесс обучения, по возможности, домашних наставников (пап, мам, братьев, сестер...).
- Использование социальных сервисов Интернета для публикации детских работ и создания портфеля достижений каждого ребенка.

Прокомментируем отдельные важные моменты методики.

Иллюстративные презентации

Для проведения урока учитель может использовать иллюстративные материалы (презентации), которые входят в состав УМК курса.

Игры

Игровые формы традиционно используются в обучении младших школьников.

В “Азбуке” предлагается авторская реализация данного принципа. Игра сопутствует практически всем разделам курса и проводимым занятиям.

Например, объяснение нового материала, проверка усвоенных знаний ведется с использованием роботландских героев (Вася Кук, Буквоед, Хролик, Трям, Прям, Кукарача, Агент РБ и других).

Оценки

Результаты работы детей оцениваются не в виде традиционных оценок, а с помощью “почетных званий”: *Профессор, Студент, Торопыжка, Незнайка* и вручением “дипломов” (“профессорам” и “студентам”).

Физминутки

Физкультурные паузы, или физминутки, становятся не только комплексами физических упражнений, но и игровыми фрагментами урока. “Превращаемся в курсор”, “Буратино и Пьеро”, “Окно программы”, “Глаза и пальцы”, “Кто внимательнее?” и другие комплексы упражнений, с удовольствием выполняемые детьми, позволяют лучше усвоить материал занятия. В том числе, физминутки позволяют организовать серии логических и алгоритмических заданий-упражнений.

Театр Роботландии

“Театр Роботландии” — одна из реализаций игровой формы обучения. Театрализация подразумевает розыгрыш в “лицах” некоторого информационного действия или процесса.

Например, дети разыгрывают вывод информации из памяти компьютера на монитор или на звуковые колонки, демонстрируют ввод с клавиатуры текста в память компьютера.

Сначала детям объясняется задача, сценарий, затем распределяются роли, раздаются необходимые реквизиты, и дети разыгрывают сценку.

Роботизированные игрушки

Игровые моменты занятий могут быть связаны с демонстрацией роботизированных игрушек, которые использует учитель или которые приносят дети по теме занятия (например, игрушки, как устройства ввода или вывода информации), или в виде игры с использованием этих игрушек.

Практикум за партой

Одной из особенностей “Азбуки” является предметная деятельность в виде практикума, предшествующая работе за компьютером.

После объяснения нового материала дается задание с использованием дидактических раздаточных материалов: собрать элементы в правильной последовательности, собрать картинку, рассорти-

ровать объекты, поработать на интерактивной доске (вписать пропущенные буквы, установить соответствие, выполнить задание практикума путем перетаскивания элементов).

Практикум за компьютером

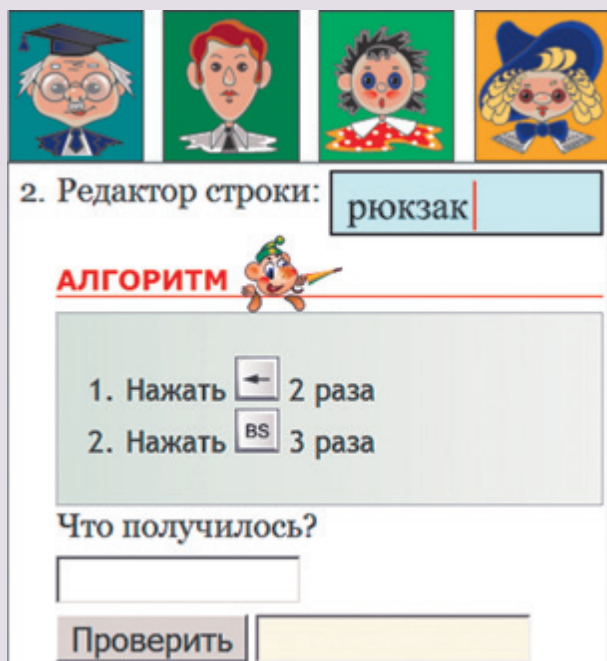


Компьютерные практикумы предполагают активную работу для достижения поставленной цели, и дети попутно наращивают интерфейсные навыки.

В примере, показанном на иллюстрации, дети работают со средой “Паспорт устройства” (на экране стрелками показаны датчики компьютеризированного вездехода).

Ученику предъявляется изображение устройства, подключаемого к компьютеру. Он должен в выпадающих списках выбрать название устройства (“датчики”) и его назначение (“проверка местности”). Затем включить нужную радиокнопку (“ввод”) и, наконец, нажать кнопку Готово.

Зачетный класс



Регулярный контроль знаний осуществляется в разных формах, в том числе и в виде компьютерного зачета по теме занятия.

Оценивание ведется с использованием традиционных визуальных образов: *Профессор* (задание выполнено без ошибок), *Студент* (одна-две ошибки), *Торопыжка* (более двух ошибок), *Незнайка* (много ошибок).

Составные части занятия

Занятие строится с учетом длительности, рекомендованной для начальной школы. Занятия первого года обучения длятся 35 минут, со второго по четвертый классы — 45 минут. Занятие строится по следующему “типовому” сценарию:

1. Повторение пройденного материала в виде ответов на вопросы.
2. Объяснение нового материала.
3. Физминутка.
4. ПрактикумП (за партой).
5. ПрактикумК и Зачет (работа за компьютером).
6. Подведение итогов занятия.

Работа за компьютером является одним из шести этапов занятия и занимает не больше 20 минут в соответствии с гигиеническими “Требованиями к условиям реализации основной образовательной программы начального общего образования” в рамках стандарта начального общего образования второго поколения.

Отметим, что необходимости в более длительной работе за компьютером и не возникает. Ведь на обучение хорошо работают и другие, некомпьютерные части урока.



Потому что мы пилоты

В 2010–2011 учебном году в пилотном режиме была запущена первая часть курса — “Компьютер”. За работу взялись 9 учителей из 7 школ (Мурманск, Тверь, Челябинск и Белорецк Челябинской области) и 15 групп детей (всего 168 первоклассников и 15 второклассников). В работе участвовали не только учителя информатики, но и два учителя начальной школы.

Отдельно отметим важнейший момент — один из авторов “по совместительству” работал тьютором, второй внимательно следил за уроками по видеозаписи.

Кроме учителей-тьюторов в проекте приняли участие 3 советника — они хотели, но по разным причинам не смогли работать со школьниками в рамках проекта.

Курс был реализован в форме кружка. В школах, являющихся пилотными площадками по введению новых стандартов начального общего образования,

кружок действовал в рамках дополнительных занятий школы полного дня.

Первые уроки прошли в конце октября, большая часть групп начала занятия со второй четверти. Последние занятия состоялись в конце мая, параллельно с окончанием учебного года в школах.

Как мы летали

Сетевые сервисы

Для взаимодействия участников проекта был создан электронный список рассылки.

На первом этапе именно он помог всем понять общие идеи курса, его цели и задачи. В дальнейшем список выступил в качестве рабочего коммуникационного инструмента. Через него шло все публичное общение, и тьюторы публиковали на нем отчеты о прошедших уроках.

Кроме того, были задействованы документы Google (публикация домашних заданий, правда, не востребованная в пилоте), веб-альбомы Google (фоторепортажи уроков и работы школьников), YouTube (видеорепортажи уроков и учебные ролики к урокам).

Взаимодействие с родителями

Авторы рассчитывали на активную помощь домашних наставников. Поэтому тьюторам было рекомендовано до начала занятий провести встречи с родителями (во время одного из текущих школьных родительских собраний).

На этой встрече предлагалось объяснить содержательную сторону предстоящих занятий и техническую, относящуюся к работе с учебником в домашних условиях.

Дальнейшее взаимодействие с родителями у каждого учителя строилось по-своему. Учитель начальных классов контактирует с родителями практически каждый учебный день. Учитель информатики лишен такой возможности. Был опробован вариант создания Google-группы для родителей первоклассников.

Список рассылки помог распределить детей по группам в удобное для посещения время, позволил оперативно извещать родителей о датах занятий или их изменениях, об особенностях изучаемых тем и заданий, о характерных ошибках, к которым иногда подталкивали детей родители, исходя из своего понимания заданий (например, в задании с пиксельным рисованием).

От родителей поступали предложения друг другу о совместных домашних занятиях детей, в случае, если кто-то испытывал затруднения с доступом к Интернету. С помощью списка часть родителей отвечала на вопросы анкеты, которая была предложена в конце учебного года.

Всего за учебный год на списке рассылки было опубликовано 77 сообщений. Из них 38 — учителем. Взаимодействие через электронную связь было на-

ложено, Google-группа показала себя действенным инструментом. Связь между учителем и родителями шла и по личным электронным адресам.

Замечание. “Один урок в учебнике — одно учебное занятие”. Подобный режим обучения получался при поддержке школьных занятий систематической домашней работой детей с электронным учебником под руководством наставника. Если учитель не рассчитывает на работу детей дома, время обучения увеличивается. Опыт пилотных групп показал, что в одном случае в течение первого (неполного) года было освоено 4 темы (20 уроков), а в другом 3 темы (15 уроков). При работе только в классе можно рассчитывать на основной (минимальный) уровень освоения курса (означающий выполнение обязательных, но не всех предложенных в учебнике, заданий).

Мониторинг занятий

Учителя описывали каждое проведенное занятие и отправляли свой отчет на список рассылки проекта. В результате возникающих обсуждений вносились коррективы в трактовки учебника, в методику проведения занятий.

В процессе подобных обсуждений, например, родилась идея проведения обобщающего урока в конце каждой темы (“экзамен в Роботландии”).

Уроки, по возможности, фотографировались и снимались на видео.

Выложенные в Интернет, они были доступны для родителей и коллег. В съемке неоценимую помощь оказали родители, а также коллеги-учителя.

Где приземлились

Раскрывая стержневые темы курса:

- знакомство с устройством и функционированием компьютера и
 - введение в компьютерный интерфейс, реализуемый с помощью физических устройств и экранных объектов,
- мы не забывали и о сверхзадачах — о формировании алгоритмического мышления, об общем развитии ребенка, о раскрытии его индивидуальностей (через творческие домашние задания) и демонстрации преимуществ кооперации (через учебные командные игры и поощрение взаимопомощи при выполнении зачетов и практикумов).

Алгоритмика

Говоря о формировании алгоритмического мышления и развитии практических навыков работы на компьютере, необходимо отметить замечательное переплетение двух этих направлений, случившееся в нашем курсе.

Осваивая набор и редактирование текста, мы приступили к выполнению и составлению алгоритмов, и даже к программированию! Причем работа в этом направлении началась со второго занятия, с появления на уроке термина “алгоритм”.

В дальнейшем эта линия была поддержана двумя специальными программируемыми исполнителями.

Ребята с большим удовольствием выполняли как задания, представленные в печатном виде, так и на компьютере, с помощью электронного учебника.





Практикумы, связанные с алгоритмизацией и программированием, — это:

- 15 заданий на выполнение или сборку алгоритмов (из готовых действий);
- 23 задания на программирование редактора строки (исполнитель РС-1);
- 70 упражнений с шифровками (исполнитель Агент РБ).

Навыки набора текста

Параллельно с основными темами не прекращалась работа на клавиатуре.

Дети освоили:

- набор русского и английского текста в строчном редакторе;
- использование модификатора  и переключателя ;
- набор на клавиатуре специальных знаков;
- редактирование текста с помощью операций  и .

В курсе предусмотрено 65 упражнений на освоение клавиатурного набора и редактирование текста.

Первые итоги

Непрерывный мониторинг “полета” (в том числе просмотр видеозаписи уроков), участие одного из авторов в проекте в качестве тьютора, анкетирова-

ние в конце учебного года (тьюторов, советников, родителей) — все это позволяет с уверенностью говорить о том, что, работая по принципу максимума (*максимума возможного в развитии детей*), мы все же не “перегнули палку”, сумели сохранить детский интерес, не превышая при этом допустимые нормы работы за компьютером.

Пилотом остались довольны учителя (в том числе и благодаря солидной методической поддержке курса) и родители.

Чему научились

В результате работы пилота у детей было сформировано:

- представление об устройстве компьютера, его основных и периферийных блоках;
- представление о программном обеспечении — основе автоматической работы компьютера;
- представление об элементах экранных объектов, необходимых для взаимодействия человека и программного обеспечения компьютера (значки, окна, курсоры, меню);
- умение работы с устройствами ввода (клавиатура, мышь);
- первичное умение работать с текстом на компьютере (набирать, редактировать, сохранять);
- умение понимать, выполнять и строить не сложную последовательность действий (работа с алгоритмами и программами).

Все это было достигнуто благодаря совету Васи Кука, одного из главных героев Роботландии, который не устал повторять: “Моя голова помнит лишь то, что делают мои руки!”



РОБОТЛАНДСКИЙ УНИВЕРСИТЕТ ПРОИЗВОДИТ НАБОР СТУДЕНТОВ НА 2011/2012 УЧЕБНЫЙ ГОД

В университет (руководитель — Дуванов Александр Александрович) принимаются коллективные, а на курс “42. Web-конструирование” — коллективные и индивидуальные ученики.

Коллективный студент — это группа детей, работающая под руководством одного или нескольких наставников (наставник, как правило, школьный учитель).

Индивидуальный студент — это учитель, желающий пройти обучение индивидуально, без группы детей.

В конце годичного курса при успешном обучении школьный учитель получает удостоверение от негосударственного образовательного учреждения “Роботландия” и удостоверение государственного образца о прохождении курсов повышения квалификации и переподготовки работников образования. Школьники получают удостоверение от “Роботландии”.

Занятия в университете платные (цены — на сайте www.botik.ru/~robot). Они начинаются 10 октября текущего года и продолжаются

в течение двух семестров до мая следующего года. Для подписчиков “Информатики” предусмотрена 10%-ная скидка за обучение.

Характерные черты Роботландской школы:

1. Совместное обучение учителя и школьников в рамках одной команды.
2. Турнирный цикл обучения.
3. Моделирование коллективной деятельности.
4. Реальная практическая польза детских проектов.
5. Перекрестные проверки работ.
6. Развитые горизонтальные связи.

Заявки принимаются по адресу: kurs@robotland.pereslavl.ru до 1 октября.

Для получения более подробной информации можно написать по адресу kurs@robotland.pereslavl.ru (администрация университета) или заглянуть на сайт www.botik.ru/~robot (здесь можно отправить электронную заявку).

КРАТКОЕ ОПИСАНИЕ КУРСОВ

Номер и название курса	Возраст детей	Куратор курса	Описание курса
11. Зимние вечера	3–4-е классы	Первин Юрий Абрамович	Знакомство с алгоритмами и исполнителями, электронная почта; детские конкурсы
12. Азы информатики-I. Знакомимся с компьютером. Работаем с информацией	3–5-е классы	Кацай Ирина Ивановна	Начала информатики для малышей. Коллективная работа на Wiki-сайте и других сервисах Web 2.0
14. Азы информатики-II. Пишем на компьютере	4–6-е классы	Кацай Ирина Ивановна	В рамках обозначенной темы курс связывает пять контентов: познавательный, инструментальный, концептуальный, дизайнерский и творческий. Коллективная работа на Wiki-сайте и других сервисах Web 2.0
31. Азы программирования-I. Плюстик и Кукарача	6–8-е классы	Садовая Ирина Владимировна	Для детей, знакомых с понятиями “исполнитель”, “алгоритм”, “программа” и желающих поближе познакомиться с программированием
32. Азы программирования-II. Корректор	7–9-е классы	Садовая Ирина Владимировна	Продолжение курса 31 на базе исполнителя Корректор
42. Web-конструирование	Старшие классы	Дуванов Александр Александрович	Создание сайтов на базе HTML+CSS+JavaScript (последнее факультативно). Основы проектирования, веб-дизайна и юзабилити. Планируется кружок по теме, связанной с JavaScript



Алгоритмы и программы “Азбуки Роботландии”

А.А. Дуванов,
г. Переславль-Залесский,
kurs@robotland.pereslavl.ru
Н.Д. Шумилина,
г. Тверь,
nshumilina@yandex.ru

“Азбука Роботландии” — это четырехгодичный курс информатики для младших школьников, создаваемый в процессе (и с учетом) его использования в школьных классах. Отдельная статья “Азбука Роботландии” описывает созданный к настоящему моменту УМК первого года (раздел “Компьютер”) и рассказывает о том, как курс работал (и создавался) в рамках пилотного проекта, запущенного осенью 2010 года.

Данная статья описывает алгоритмическую линию курса. В частности, рассказывает о двух новых программируемых исполнителях, придуманных в процессе работы над проектом.

Развитие алгоритмического мышления — основная задача “Азбуки Роботландии”, и мы старались решать ее на каждом занятии, не ограничиваясь красивыми лозунгами на вратах нашего проекта.

Понятие алгоритма вводится на втором занятии и с этого момента становится рабочим словом. Напомним, что первая часть курса, о которой идет речь

в статье, ориентирована на учеников первого класса.

На диске, сопровождающем эту статью, записаны:

- Демоверсия учебника (с тремя уроками).
- Демоверсия методички (с тремя уроками).
- Набор практикумов исполнителя РС-1.
- Набор практикумов исполнителя Агент РБ.

Икра по-роботландски (фрагмент урока 2 из учебника)

Азбукой называют **алфавит**. А еще азбукой называют **букварь** — книгу для обучения грамоте.

“Азбука Роботландии” — это наш **букварь по информатике**. Мы будем учиться компьютерной грамоте, учиться управлять роботами.

Сегодня выучим четыре буквы из “Азбуки Роботландии”. Эти буквы легко запомнить — из них складывается слово **ИКРА**.

И_{нформатика} **нформатика** — наука и школьный предмет. Учит работать на компьютере — основном инструменте информатики.

Слово “информатика” созвучно со словом **информация**. Это не случайно. Ведь компьютер работает с числами,

словами, картинками, звуками, а это все — **информация**.

Компьютер — прибор, на котором можно:

К_{омпьютер}

- читать;
- писать;
- считать;
- рисовать;
- играть;
- сочинять музыку;
- общаться;
- управлять роботами;
- и многое другое.

Р_{обот}

Робот — это устройство, которое выполняет работу самостоятельно, **автоматически**.

А_{лгоритм}

Алгоритм — это план выполнения работы.

Компьютер всегда работает по плану, то есть по алгоритму.

Алгоритм — это *план*, в котором описано, что и как надо делать.



Мама посылает дочку за продуктами и говорит: — Сходи, Катюша, в магазин, купи пакет молока и кусок сыра.

Это алгоритм. Описание того, что нужно сделать:

1. *Сходить* в магазин.
2. *Купить* молока и сыра.



А если в магазин послать робота? Робот не человек, и ему нужно описать все подробно:

- адрес магазина;
- как идти, быстро или медленно;
- что делать, если молока или сыра нет;
- куда положить продукты;
- как возвращаться.

Алгоритм — это план работы.

Робот работает по **алгоритму**.

Алгоритм — самое важное слово в нашей азбуке!

Замечание. Отметим, что предметом науки “информатика” является *информация* (передача, обработка и хранение информации), а вовсе не компьютер. *Компьютер* — это инструмент информатики, как микроскоп — инструмент биологии, компас — инструмент географии, телескоп — инструмент астрономии.

Однако достижения современной информатики базируются исключительно на возможностях компьютерной техники (невозможны без нее), поэтому информатика однозначно ассоциируется с компьютером.

Первоклассникам нужно объяснить, чем они будут заниматься на том или ином предмете.

Мы говорим:

На математике учимся *считать* (хотя математика — это не счет).

На русском учимся *писать* (хотя русский язык — это не письмо).

На литературе учимся *читать* (хотя чтение — это не литература).

А на уроках информатики? На информатике *учимся работать на компьютере* (хотя компьютер — это не информатика).

Эти формулы (математика — счет, русский — письмо, информатика — компьютер) относятся к первым шагам в том или ином учении: учимся считать, писать, работать на компьютере.

Авторам очень хотелось заменить в учебнике фразу “информатика учит работать на компьютере” на “информатика учит работать с информацией” или “информатика учит работать с алгоритмами” (ибо алгоритмы — основа информатики, а развитие алгоритмического мышления — та основная цель, которую мы ставим перед собой в этом курсе). Но информация и алгоритмы — абстракции, недоступные пока нашим ученикам, а компьютер — это конкретно, вот он, красавец, стоит на виду, и дети действительно будут осваивать на нем компьютерную грамоту. Так что наши высокие цели придется оставить пока неозвученными. Настанет время, и мы с этим разберемся основательно.

Отметим, что под *компьютерной грамотностью* мы понимаем такие навыки, умения и знания, которые *устойчивы* по отношению к бурным переменам современного общества и которые способны подготовить нынешних первоклашек к жизни в информационном обществе после школы. В частности, в ходе обучения мы будем стараться давать не кнопочные инструкции (щелкни на иконке домика), а идейные алгоритмы (вернись на начальную страницу). Ибо кнопки меняются или вообще пропадают (управление жестами), а идеи остаются.

Алгоритмы в действии

Во втором уроке рассказывается (среди прочего), как перетаскивать мышкой объекты по экрану. Инструкция записывается в виде алгоритма:

АЛГОРИТМ



1. **Расположи** курсор над объектом.
2. **Нажми** левую кнопку мыши.
3. **Перемести** объект на новое место, *не отпуская* кнопки.
4. **Отпусти** кнопку.



Первый компьютерный практикум урока закрепляет смысл новых слов и позволяет освоить алгоритм операции перетаскивания на практике.

Ученику предлагается перетащить ярлыки с новыми словами урока (*информатика, компьютер, робот, алгоритм — икра*) на “свои” картинки.



Операция перетаскивания задействована и во втором практикуме сразу с двух сторон: идейно-алгоритмической и практической.

Ученику предлагается собрать алгоритм перетаскивания при помощи перетаскивания!

В уроке 3 речь заходит о назначении процессора:



А что делает процессор?

Процессор **обрабатывает** информацию.

А как он это делает?

Процессор работает по **алгоритму**.

Следом приводится пример конкретного алгоритма для процессора.

Пример (фрагмент из учебника урока 3)



Решая задачу сложения, Вася Кук ввел с клавиатуры в компьютер такой алгоритм:

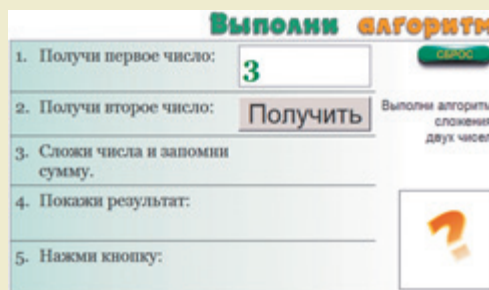
АЛГОРИТМ



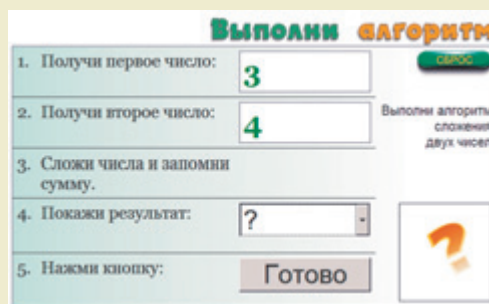
1. **Прочитай** первое число.
2. **Прочитай** второе число.
3. **Сложи** числа.
4. **Покажи** результат.

Затем Вася ввел с клавиатуры два числа: 2 и 3. Процессор выполнил алгоритм, и на экране компьютера засветилось число 5.

Практикум урока (один из двух) поддерживает тему соответствующим интерактивным упражнением:



Слагаемые генерируются автоматически после нажатия на кнопку **Получить**.



Ответ выбирается в выпадающем списке (отрабатываем и этот интерфейсный навык).



Кнопка **Готово** фиксирует результат и показывает оценку.

Физминутка урока 3 тоже алгоритмическая:



Хролик (персонаж курса) решил научить Буквоеда (персонаж курса) алгоритмической гимнастике.

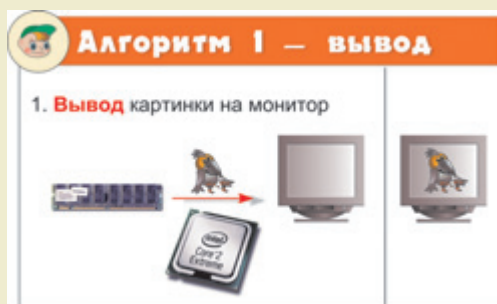
Посмотрите, какой алгоритм составил Хролик! (На иллюстрации изображен именно Хролик.)

Стрелки показывают, в какую сторону надо потянуться или повернуться. Тянемся по очереди каждой рукой вверх, потом в сторону, потом поворачиваемся и прыгаем! Каждое движение повторяем 4 раза, кроме прыжков, а прыгаем 8 раз.

В классе открывается “театр Роботландии”, тоже алгоритмический.

Алгоритм 1. Вывод картинку (фрагмент методички урока 3)

К доске по очереди вызываются команды из трех человек: при выводе они будут изображать процессор, память, монитор; при вводе — клавиатуру, процессор, память. Количество рассматриваемых примеров можно отрегулировать так, чтобы каждый ребенок смог участвовать в инсценировке.



Между детьми распределяются роли (прикрепляются карточки-беджи или надеваются шапочки с надписями, или надеваются карточки с надписями на веревочках).

По слайду уточняем поставленную задачу (“**вывод** картинку на монитор”), поэтому “память” заранее получает в руки картинку.

Обязательно надо указать на слово “вывод” на слайде. Дети должны видеть написание этого слова. На слух дети могут сначала не различить слова “ввод” и “вывод”. А нам важно добиться понимания разницы этих двух процессов.

Процессор должен взять “из памяти” картинку и передать ее “монитору” (выполнить вывод).



Стоит попросить детей проговорить свои действия.

Процессор. Беру из памяти картинку.

Память. Отдаю картинку процессору.

Процессор. Вывожу картинку на монитор.

Монитор (поднимает изображение над головой). Показываю картинку на экране.

Учитель (подводит итог). Процессор *вывел* картинку на монитор. Монитор — устройство *вывода*.



Теоретическое отступление

Прежде чем продолжить описание реализаций алгоритмической линии в курсе “Азбука Роботландии”, сформулируем теоретические послы, через которые она прошла. Признаемся — “реперные” точки нашей линии корректировались в процессе работы с детьми, пройдя трудный путь проб и ошибок.



Отметим, что при работе с алгоритмами одинаковой сложности (в нашем линейном случае примерно совпадающей с количеством пунктов алгоритма):

- легче быть исполнителем
- и гораздо труднее — составителем.

Роль исполнителя детям привычна и на бытовом уровне (умыться, почисти зубы, собери портфель), и в играх (определим ведущего, он считает до 20, все бегут прятаться, ведущий ищет, игрок бежит к “дому” и так далее).

А вот *составить* алгоритм гораздо сложнее. Опыт такой деятельности практически отсутствует. А когда возникает, детям не хватает формального воображения. Ибо алгоритм есть абстракция. При составлении алгоритма нужно описывать действия, которые в данный момент не происходят.

В “Азбуке Роботландии” в качестве основы для создания формальных алгоритмов (программ) мы взяли среду программного управления редактором строки (исполнитель РС-1, он подробно описан ниже в этой статье). Такой выбор был обусловлен началом активной работы с текстом (набор, редактирование).

И что же? Получается, что детям нужно записывать для РС-1 команды о передвижениях курсора, в то время, когда курсор неподвижен. Значит, нужно мысленно представлять эти движения и держать изменяемое состояние среды в голове во все время написания программы. Аналогичные трудности возникают, когда записываются команды на выполнение действий клавишами  и . Кроме того, нужно постоянно держать в голове конечное состояние среды, достигаемое выполнением правильной программы. Чем больше команд в программе, тем труднее дается это умственное напряжение, ибо на сложность восприятия абстракции (представить то, чего нет) накладывается сложность удержания в памяти большого числа элементов.

Очевидный способ смягчения этих трудностей — создание такой среды разработки программ, в которой постоянно отображалось бы и начальное положение среды исполнителя (“дано”), и конечное (“надо”), и промежуточные состояния, достигаемые выполнением уже написанных команд (“выполнение программы”).

Именно так и был построен исполнитель РС-1 (среда исполнителя + визуальная интерактивная среда разработки программ для него), но такое благоприятное устройство исполнителя само по себе не решило проблем с абстракциями (абстрактное мышление все равно надо развивать!) и принесло новую трудность — интерфейсную, ибо работу с исполнителем нужно тоже осваивать.

Разработав среду разработки программ, комфортную для юного разработчика, авторы решили, что дети легко с ней справятся и весело займутся программированием.

Мы придумали серию заданий с разным количеством действий (программы в одну команду показались нам слишком простыми) и с воодушевлением пошли с ними в класс. Но нас постигло разочарование. Дети с трудом ориентировались в среде РС-1 (интерфейсная сложность оказалась выше, чем мы полагали), составление программ, содержащих более одной команды, не получилось, и даже на программах с одной командой дети застревают, не понимая разницы между написанием команды и ее выполнением.

Мы огорчились и решили, что давать формальное программирование детям первого класса рановато. К счастью, поостыв от неудачи, решили повторить попытку, приложив к нашей благоприятной среде разработки программ очевидный, вообще говоря, прием: постепенное продвижение от простого к сложному методом маленьких шажков, многократно выполняемых в разных видах, в том числе и некомпьютерных.

И это принесло удачу! Дети уловили суть программирования, они поняли, что программа пишется для исполнителя, который выполняет программу после запуска. Появился интерес, желание правильно выполнить задание.

Итоги можно сформулировать так.

Алгоритмическое мышление особенно успешно формируется при программировании (составлении алгоритмов для формальных исполнителей). Ибо программы — это как формулы в математике. Формулы — основа математического мышления, программы — основа алгоритмического.

Программирование невозможно без работы с абстракциями, ибо программа описывает действия, которые в момент ее написания не происходят. И в этом состоит сложность поставленной задачи. У детей первого класса абстрактное мышление только формируется.

Поэтому для успешного обучения программированию нужна тщательная педагогическая проработка среды используемого формального исполнителя. Необходимо:

- предъявить ученику среду разработки, в которой предусмотрена визуализация планируемых действий, а также памятка о начальном (“дано”) и конечном (“надо”) состоянии среды;
- максимально упростить интерфейс ученика со средой разработки;
- защитить создаваемую программу от синтаксических ошибок (не написание команды, а ее конструирование из готовых элементов списка выбора);
- задать позитивную визуальную мотивацию: учим программировать Незнайку и помогаем ему получить звание Профессора.

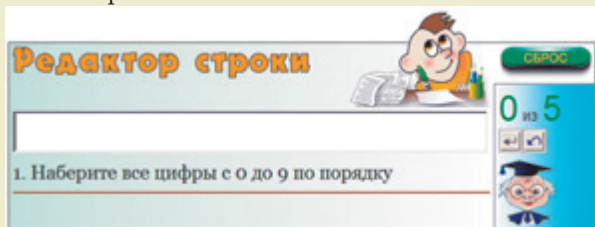
Кроме того, необходимо разработать методическую линию, содержащую систему упражнений для освоения программирования в созданной среде, основываясь на принципе постепенного продвижения от простого к сложному, с применением разнообразных форм представления алгоритмических действий:

- предметных и игровых;
- словесных (устных и письменных);
- знако-символьных в печатном виде;
- и, наконец, в среде самого формального исполнителя.

Далее мы покажем, как эти теоретические размышления были реализованы на практике.

Алгоритм не выполняет работу, он ее описывает

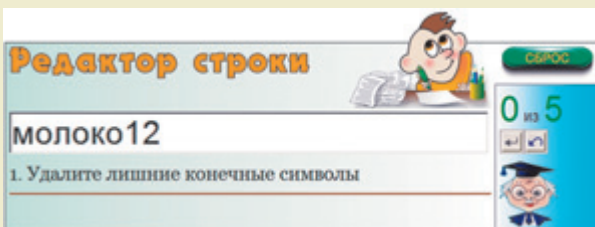
Начиная с урока 7 дети осваивают набор текста в специальном редакторе строки со встроенным блоком контроля:



Урок 9 начинает тему исправления клавиатурных ошибок, и задания следующего типа ученики выполняют без особых проблем.

Задание

Удалите лишние конечные символы с помощью клавиши BS:




В самом деле, какие могут быть сложности? Дети устанавливают курсор в конец записи (клавиатурой или мышкой) и нажимают два раза BS.

В следующем, чуть более сложном практикуме предлагается удалить лишние символы, расположенные в разных местах слова:

Коооооза
Мышшшшшшшкааааааааааа
Парррррррохххххххххход

Эти практикумы связаны с усвоением алгоритма удаления лишнего символа при помощи клавиши BS:

АЛГОРИТМ 

1. **Перемести** курсор к лишнему символу так, чтобы символ оказался слева от курсора.
 2. **Нажми** клавишу BS.
- Закрепляем знание цепочкой упражнений с нарастающей сложностью.

Примеры некомпьютерных упражнений


1 Редактор строки:

АЛГОРИТМ 

1. Нажать BS 4 раза

Что получилось?


2 Редактор строки:

АЛГОРИТМ 

1. Нажать ← 5 раз
2. Нажать BS 2 раза

Что получилось?

3 Редактор строки:

АЛГОРИТМ 

1. Нажать ← 2 раза
2. Нажать BS 3 раза

Что получилось?

Примеры компьютерных упражнений

1. Редактор строки:

Нажали BS один раз. Что получилось?

Проверить

2. Редактор строки:

Нажали BS один раз. Что получилось?

Проверить

3. Редактор строки:

Нажали BS два раза. Что получилось?

Проверить

Для работы в классе были подготовлены бу-
мажные распечатки с набором упражнений, и
дети с удовольствием брали эти задания в каче-
стве домашней работы (приводятся фрагменты
заданий).

Примеры некомпьютерных упражнений			
Дано	Алгоритм	Получилось	
носорог	4		нос
лошадка	3		лодка
	2		
козерог	1		озеро
	5		
	1		
Дано	Алгоритм	Получилось	
море	BS 1 с		морс
девять	←3 BS 1 с		десять
окошко	BS 1 а ←5 BS 1		кошка

Обратим внимание на то, что задания, представ-
ленные в таблице, существенно отличаются от пер-
вых, начальных заданий компьютерных практику-
мов “Удалите лишние символы с помощью клавиши
”, которые имели обобщенную формулировку,
без указания конкретного количества действий с
помощью клавиши .

Задания же в таблице представлены в виде фор-
мальных алгоритмов для исполнителя, который
умеет выполнять записанные команды. Команды
надо выполнять абсолютно точно, в приведенном
порядке и именно такое количество раз, которое
указано в записи. Можно выполнить алгоритм
устно, а можно “поддержать” выполнение явным
зачеркиванием букв в слове, моделируя курсор ка-
рандашом или ручкой.

Задания составлены так, чтобы из одного значи-
мого слова получалось другое значимое слово (до-
полнительная мотивация, задание превращается в
разгадку головоломки).

Урок 11 продолжает алгоритмическую тему.

В уроке 12 обсуждается вариант удаления кла-
вишей (удаление справа от курсора), возмож-
ности редактирования расширяются, а запись

Примеры компьютерных упражнений к уроку 11.

1. Редактор строки: листопад |

АЛГОРИТМ



1. Нажать 4 раза

Что получилось?

лист

Проверить **верно**

2. Редактор строки: слалом |

АЛГОРИТМ



1. Нажать 2 раза

2. Нажать 3 раза

Что получилось?

сом

Проверить **верно**

3. Редактор строки: винчестер |

АЛГОРИТМ



1. Нажать 2 раза

2. Нажать 1 раз

3. Нажать 3 раза

Что получилось?

винт

Проверить **верно**

команд алгоритма принимает все более формаль-
ный вид (приближаясь к командам из СКИ испол-
нителя PC-1).

Примеры компьютерных упражнений

1. Редактор строки: авиатор|

АЛГОРИТМ

- ← 5
- Del 2

автор

Проверить верно

2. Редактор строки: |дяденька

АЛГОРИТМ

- Del 2
- 4
- Del 2

день

Проверить верно

Примеры некомпьютерных упражнений

№	Дано	Команды	Получилось
1	лошадка	← 3 BS 2	
2	козерог	BS 1 ← 5 BS 1	
3	математика	Del 2 → 4 Del 4	
4	самовар	BS 1 ← 3 BS 2	

Фактически теперь дети готовы к программированию формального исполнителя РС-1.

Юные программисты за работой

И вот, наконец, в уроке 13 Кукарача рассказывает о программах и об исполнителе РС-1.

Программирование (фрагмент урока 13)



Кукарача Роботландский — большой специалист по программированию.

— Привет! Меня зовут **Кукарача**. Я буду учить **программировать**. Ах да! Вы не знаете, что такое **программа**!

Но вы знаете, что такое алгоритм!

Алгоритм — это план выполнения работы.

Алгоритм состоит из **шагов** или **пунктов**, которые нумеруются по порядку их выполнения.

Программа — это примерно то же самое, но...

Программа — это такая запись алгоритма, которую понимает робот.

Программа состоит из **команд**, которые выполняются в порядке их записи.

Можно сказать, что **программа** — это алгоритм, записанный на языке робота.



— Поясню на примере разницу между алгоритмом и программой!

В Роботландию поступило задание:

Задание. Из слова *лиса* получить слово *лис*. Курсор расположен в начале строки.

Дано:

лиса|

Надо:

лис|

— Человеку мы можем предложить план работы на клавиатуре, состоящий из двух шагов:

АЛГОРИТМ

- Нажми** клавишу со стрелкой, смотрящей влево, три раза.
- Нажми** клавишу удаления один раз.



В Роботландии есть робот **РС-1**, который умеет работать с текстом.

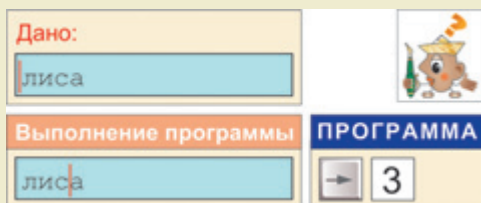
Но, увы, робот **РС-1** запись алгоритма, приведенную выше, не поймет!

Алгоритмы для робота РС-1 пишут так:

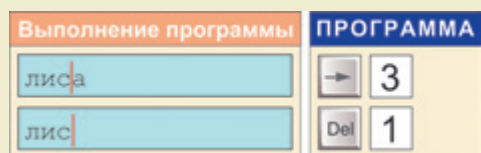


Это уже не просто алгоритм! Эта запись — **программа**. Программа для робота РС-1. Программа состоит из двух команд, и ее выполнение начнется после нажатия на специальную кнопку **Пуск** (кнопка запуска).

Сначала робот выполнит первую команду — передвинет курсор вправо на 3 символа:



Потом вторую — удалит справа от курсора один символ:



РС-1

— А что означают буквы РС и цифра 1 в названии робота?

— РС — это сокращение от “Редактор Строки”, а цифра 1 обозначает первую модель.



Правда, сам робот расшифровывает свое название как “Робот Смелый”, **первый** среди других роботов! А когда выпьет чаю с сухариками, то и вовсе говорит, что он “Роботландский Сапфир”!

Устройство РС-1

Среда исполнителя — *редактор строки*. Система команд исполнителя (СКИ) состоит из 5 команд, имеющих общий формат:

Действие	n	Указанное в команде действие РС-1 исполняет n раз (n может принимать значение от 1 до 9).
----------	---	---

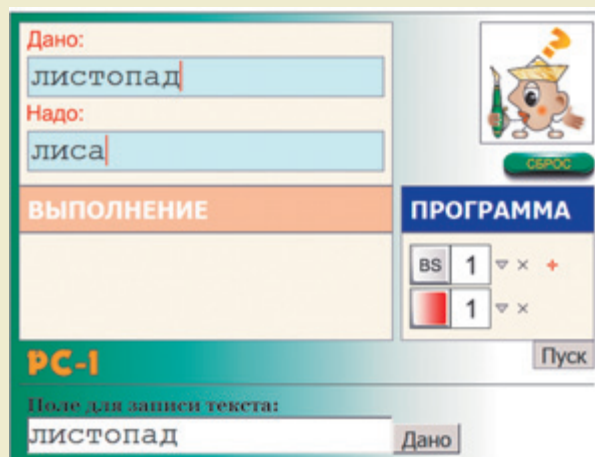
В таблице в правой колонке вверху представлен полный набор возможных действий РС-1.

Среда разработки программ для исполнителя имеет:

- поле с постановкой задачи (*дано, надо*);
- поле для вывода протокола пошаговой работы исполнителя (*выполнение*);
- поле для конструирования программы (*программа*);

Команда СКИ	Как РС-1 выполняет команду
n	Перемещает курсор на n символов влево
n	Перемещает курсор на n символов вправо
BS n	Удаляет n символов слева от курсора
Del n	Удаляет n символов справа от курсора
n	Пустая команда. РС-1 не выполняет никаких действий

- черновик для записи текста (*поле для записи текста*) с кнопкой **Дано**, которая заносит в поле исходное слово;
- кнопку **Пуск** для запуска программы и кнопку **Сброс** для приведения исполнителя в исходное состояние.



Конструирование программы выполняется мышью при помощи следующих интерфейсных действий:

- щелчок по кнопке с названием команды открывает список выбора из 5 возможных вариантов;
- щелчок по кнопке с параметром открывает список выбора из 9 возможных вариантов (выбор числа от 1 до 9);
- щелчок по кнопке вставляет пустую команду за текущей;
- щелчок по кнопке удаляет текущую команду;
- команду можно перетаскивать на другое место в программе за специальную “ручку” .

На уроке, для подготовки к работе с исполнителем РС-1, предлагаются задания, совпадающие с первыми четырьмя заданиями компьютерного практикума: из заданного слова получить другое слово, с учетом начального положения курсора. Отличие в том, что дети не составляют, а выполняют готовые программы.

Дано	Надо	Программа
лист	лис	BS 1
котел	кот	BS 2
экран	кран	Del 1
победа	еда	Del 3

Для работы готовятся карточки в специальном варианте (отдельные буквы и части слов) и складываются в конверт:

лис+т, кот+ё+л, э+кран, п+о+б+еда.

При выполнении задания необходимо:

- Сложить слово.
- Используя карандаш (или ручку) в качестве курсора, выполнить команды по заданию. По командам BS или Del удалять буквы из слова.
- Прочитать полученное слово.

Этот практикум несложен для детей, хорошо понявших работу команд BS и Del, и особенно полезен тем, кто чувствует себя пока не очень уверенно.

После выполнения упражнений с карточками предстоит работа с исполнителем PC-1. Для урока подготовлено 12 заданий на программирование. Шесть основных и шесть дополнительных.

Первые четыре задания (смотрите практикум с карточками) содержат только одну команду. Они знакомят детей как с разными вариантами команд (BS, Del), так и с возможностью разного количества повторений.

Поле программы PC-1 в начальный момент содержит пустую команду с параметром-повторителем 1, и набор программы сводится к последовательному редактированию этой пустой команды:

Номер	Дано	Надо	Программа	Комментарий
001	лист	лис	BS 1	Редактируем только название команды
002	котел	кот	BS 2	Редактируем название и параметр
003	экран	кран	Del 1	Редактируем только название команды
004	победа	еда	Del 3	Редактируем название и параметр

Следующие задания требуют добавления второй команды. Сразу же знакомим с обратным действием — удалением лишней команды (добавленной по ошибке).

Номер	Дано	Надо	Программа
005	риск	рис	→ 3 Del 1
006	леска	лес	→ 3 Del 2

Таким образом, интерфейс осваивается последовательно:

- Практикум 001 — учимся выбирать название команды.
- Практикум 002 — закрепляем и переносим освоенный навык на параметр.
- Практикумы 003 и 004 — закрепление с другой командой.
- Практикумы 005 и 006 — учимся добавлять новую команду (и удалять лишнюю).

Дополнительные практикумы содержат более сложные задания, выстроенные по количеству необходимых команд.

Порядок выполнения каждого задания может проходить по двум вариантам.



Вариант 1. Ставим курсор в исходное положение поля для записи текста. Выполняем нужные действия и описываем их программой. Запускаем, когда все сделано.

Вариант 2. Решаем, какая команда необходима первой. Задаем первую команду. Запускаем программу на выполнение. Получаем результат выполнения одной команды и звание “Незнайка”. Добавляем команду, запускаем программу, пока задача не решена, звание остается прежним — “Незнайка”. Запуск правильной программы меняет Незнайку на Профессора.

Именно второй вариант “заставил” выдвинуть версию: мы учим Незнайку программировать и помогаем ему получить звание Профессора.

Еще раз отметим, что ключевым моментом обучения программированию стала предварительная работа с упражнениями по освоению команд BS и Del, а также верный порядок предъявления заданий для исполнителя PC-1.

Дальнейшие упражнения для PC-1 связаны с редактированием заданной программы:

- Удалить одну лишнюю команду.
- Расположить команды в правильном порядке.
- Добавить команду.

Редактирование заготовки преследует две цели.

• Играет роль промежуточной (подготовительной) ступеньки на пути к сложным заданиям (начинать программирование приходится не с чистого листа).

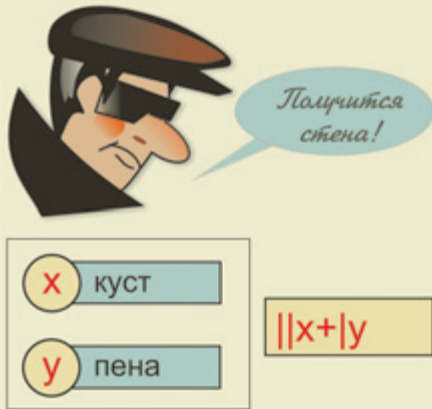
• Демонстрирует на практике процесс тестирования и отладки, без которых программирование невозможно.

Параллельно продолжается работа по составлению программ и в “бумажном” варианте:

Примеры компьютерных упражнений		
Дано	Программа	Надо
чайник		чай
восемь		семь
козлята		коза
шоколад		школа
апельсин		ель

Шпионские страсти. Агент РБ

В урок 17 “забирается” Агент РБ и предлагает свой способ шифрования текста.



Второй формальный исполнитель (*Агент РБ*, “шпион”, задающий шифровки) может обрабатывать слова, удаляя первые или последние символы (оператор `|`) и “склеивая” их (оператор `+`).

ла+па	→	лапа
КИТ +НОТЫ	→	КИНО
УС+ ТРОН	→	СОН
СОР + ОНА +Я	→	СОНЯ

“Шифровка” Агента ориентирована на исполнителя, который умеет выполнять операции “`|`” и “`+`”.

Команда СКИ	Как Агент РБ выполняет команду
<code> </code>	Удаляет одну букву по правилу <code>Del</code> , если команда стоит в начале слова, и по правилу <code>BS</code> , если в конце
<code>+</code>	Выполняет конкатенацию слов

Таким образом, программа для Агента РБ — это выражение, состоящее из текстовых данных (слов) и двух операций:

- “`|`” (унарная операция, более высокий приоритет);
- “`+`” (бинарная операция).

Выполняя программу, Агент РБ вычисляет выражение и получает новое слово. Примерно так мы записываем операции со строками в языках программирования, например, в JavaScript:

```
// переменная x получает значение "лапа"
var x = "ла"+"па"
```

Новый роботландский оператор “`|`” создает дополнительную интригу в программировании выражений с текстовыми данными.

Язык программирования — это еще более высокий уровень абстракции по сравнению с записью программы в виде линейной последовательности команд из СКИ (даже с учетом неявно зашитого в них цикла ПОВТОРИ — числовой параметр команды). Поэтому на данном этапе развития алгорит-

мической линии мы даже не говорим детям, что они программируют вычисление выражения на специальном языке программирования. Говорим о том, что надо разгадать шифровку (а реально, выполнить программу) и составить шифровку (а реально, написать программу на предложенном языке программирования).

Продолжая повышать уровень абстрактного мышления, начинаем разговор о переменных, то есть об именованных объектах, содержащих возможно изменяемые значения.

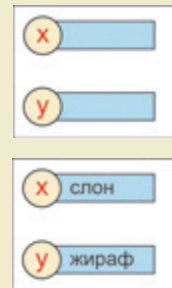


Шифровки (фрагмент урока 17)

— Привет, ребята! Зовите меня *Агент РБ*.

Посмотрите, у меня два пустых бейджика: `x` (икс) и `y` (игрек).

В `x` вставлю карточку со словом “слон”, а в `y` — карточку со словом “жираф”.



Если теперь скажу Буквоеду “`x`”, он будет знать, что я говорю ему “слон”.

Если теперь скажу Лисенку “`y`”, он будет знать, что я говорю ему “жираф”.

— Вы поняли, ребята?

Икс означает “слон”, а *игрек* означает “жираф”!

А если заменю карточки, то `x` и `y` будут обозначать совсем другие слова!



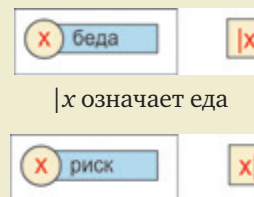
Теперь:

- `x` означает сто;
- `y` означает лица.

— А теперь, внимание: *шифровка!*

- `x+y` означает столица.

— И еще один мой способ шифрования: вертикальная черточка! Черточка удаляет букву в начале или в конце слова!



`x|` означает рис

Задания на работу с шифровками весьма разнообразны, как в компьютерном, так и безкомпьютерном варианте. Ниже приводятся несколько примеров.

Задания первого практикума Агента (из двух) в рамках урока 17 (набрать ответы шифровок):

1. $|x|$, если x =экран
2. $x|$, если x =лист
3. $|x|$, если x =школа
4. $x+y$, если x =лис, а y =ток
5. $x+|y|$, если x =стоп, а y =моль

Задания первого практикума Агента (из трех) в рамках урока 18 (набрать ответы шифровок):

1. три $|+|$ игра $|$
2. $|$ игра $|+уж|+|$ точка
3. дом $|+|$ ива $|+|$ сон
4. крот $|+код+|$ пила $|$
5. $|$ коса $|+|$ топот $|+жнец|+|$ лик

Задания первого практикума Агента (из трех) в рамках урока 19 (набрать шифровки, которые решают поставленные задачи):

1. Из слов “лук” и “она” получить слово “луна”.
2. Из слов “море” и “каша” получить слово “река”.
3. Из слов “степь” и “волна” получить слово “луна”.
4. Из слов “тут” и “сказка” получить слово “утка”.
5. Из слов “крыло” и “кашалот” получить слово “крыша”.

Задания первого практикума Агента (из трех) в рамках урока 20 (набрать значение буквы x):

1. $x+ac=ужас$
2. дом $|+x=$ дорога
3. крот $|+x+|$ шило $|=$ крокодил
4. пол $|+|+x+|$ тесто $|=$ протест
5. $|$ дерево $|+|+x+|$ зал $|=$ редиска

Задания первого практикума Агента (из трех) в рамках урока 22 (набрать ответы шифровок):

1. $x|+y$, x =мак, y =шина
2. $|x|+|y|$, x =новый, y =забор
3. $|x|+|y|+l$, x =легенда, y =вчера
4. $|x|+|y|$, x =сумерки, y =люк
5. $|x|+|y|$, x =правда, y =шесть

Таким образом, для работы с Агентом детям предлагались задания следующих типов:

- отгадать шифровку (вычислить выражение, в том числе и с переменными);
- составить шифровку (записать выражение);
- решить “уравнение” (вычислить значение переменной, входящей в равенство).

Отметим, что дети придумали большое количество собственных шифровок.

Разбор полетов

В 2010–2011 учебном году в пилотном режиме работала первая часть курса — “Компьютер”. В пилоте приняли участие 9 учителей из 7 школ (Мурманск, Тверь, Челябинск и Челябинская область, Белорецк) и 15 групп детей (всего 168 первоклассников и 15 второклассников).

По теме “Алгоритмизация и программирование” в течение учебного года детям предлагалось более сотни компьютерных заданий:

- 15 заданий на выполнение или сборку алгоритмов (из готовых действий);
- 23 задания на программирование редактора строки (исполнитель РС-1);
- 70 упражнений с шифровками (исполнитель Агент РБ).

Параллельно (по желанию) можно выполнить (и многие выполняли) более 50 упражнений, подготовленных для работы на бумаге.

Задания выстроены по степени сложности, т.е. по сложности предполагаемой деятельности и по увеличению количества команд в алгоритме:

- Выполнить алгоритм.
- Составить алгоритм.
- Обработать имеющийся алгоритм или последовательность команд (удалить лишнее, добавить команды, изменить порядок команд, определить недостающую часть).

Ребята с большим удовольствием выполняли как задания, представленные в печатном виде, так и на компьютере, с помощью двух исполнителей электронного учебника. Проведенное в конце учебного года анкетирование родителей показало, что работа со словами и шифровки очень понравились и детям, и самим родителям.

То, с чем читатель познакомился в этой статье, в программе курса выведено одной строчкой:

“В результате первого года обучения дети научатся составлять и выполнять простые алгоритмы и программы”.

Подтверждаем — научились!

Электронная версия “Информатики” для подписчиков на бумажную версию

► Уважаемые коллеги! Напоминаем вам, что если вы получили этот номер по подписке, то имеете возможность прямо сейчас получить его электронную версию в Личном кабинете на сайте www.1september.ru. Если у вас еще нет Личного кабинета — зарегистрируйтесь. Это очень быстро и, разумеется, бесплатно.

В Личном кабинете в разделе “Издания → Коды доступа” необходимо ввести код SE-00332-68580.

Зачем это нужно? Вы ведь и так держите номер в руках, а электронная версия имеется на вложенном диске? Во-первых, удобство в том, что указанный код предоставляет доступ ко всем номерам полугодия. Например, октябрьский номер вы получите в Личном кабинете уже 1 октября — день в день. А по почте он придет позже. Кроме того, электронные версии всегда будут доступны вам в Личном кабинете, что бы ни случилось с бумажной версией или с диском. Например, у моего коллеги диск сгрыз щенок ☺. Он и оглянуться не успел ☺.

ОЧНО-ЗАОЧНЫЕ КУРСЫ ПОВЫШЕНИЯ КВАЛИФИКАЦИИ

Курсы организованы совместно с Московским институтом открытого образования. По окончании обучения слушатели получают удостоверение государственного образца о повышении квалификации (с нормативным сроком освоения 72 часа).

Занятия начинаются с **3 октября 2011 г.** Стоимость обучения – 5400 рублей за один курс. Членам педагогического клуба «Первое сентября» и выпускникам наших курсов предоставляется скидка 10%.

Количество мест в группах ограничено! Прием заявок заканчивается по мере формирования групп.

Перечень курсов первого потока 2011/2012 учебного года (октябрь–декабрь)

АВТОР	НАЗВАНИЕ КУРСА	ДЛЯ КОГО ПРЕДНАЗНАЧЕН КУРС
Абдулаева Е.А.	Современные подходы к организации детской игры	Для педагогов ДОУ, педагогов дополнительного образования
Волков С.В., Храмцова Р.А.	Методика построения современного урока по литературе	Для учителей русского языка и литературы
Зайдельман Я.Н.	Алгоритмизация и программирование: от первых шагов до подготовки к ЕГЭ	Для учителей информатики
Калуцкая Е.К.	Современные образовательные технологии преподавания обществознания в школе	Для учителей истории и обществознания
Копина С.А.	Методы арт-терапии в работе школьного психолога	Для школьных психологов
Мейстер Н.Г.	Творческое развитие детей средствами художественного моделирования из бумаги	Для педагогов ДОУ, педагогов дополнительного образования
Панфилова М.А.	Современные технологии использования сказок и игр в работе с детьми и подростками	Для педагогов, классных руководителей, представителей администрации школы, школьных психологов
Рокитянская Т.А.	Музыкальная грамота в образах и движениях	Для учителей музыки, педагогов дополнительного образования
Рокитянская Т.А.	Обучение игре на музыкальных инструментах в образах и движениях	Для учителей музыки, педагогов дополнительного образования
Садовничий Ю.В.	Подготовка старшеклассников к ЕГЭ и вступительным экзаменам по математике	Для учителей математики
Сапожникова Т.Б., Полякова И.Б.	Современные методы и приемы преподавания изобразительного искусства детям	Для педагогов изобразительного искусства, педагогов дополнительного образования, учителей начальных классов
Селиванова Н.А., Шашурина А.Ю.	Структура и содержание современного урока французского языка	Для учителей французского языка
Соболева А.Е., Савицкая Н.С.	Игровые методы эффективного обучения младших школьников правописанию и чтению	Для учителей начальных классов, логопедов, детских психологов
Стюхина Г.А.	Разрешение конфликтных ситуаций в образовательной среде	Для педагогов, классных руководителей, представителей администрации школы, школьных психологов
Тригубчак И.В.	Теория и практика подготовки к итоговой аттестации по химии в форме ГИА и ЕГЭ	Для учителей химии
Ярославцева И.Б.	Основы кукловодства и кукольного театра для детей	Для педагогов ДОУ, педагогов дополнительного образования

ЗАЯВКИ МОЖНО ПОДАТЬ по телефону (499) 240-02-24 (с 15-00 до 19-00 по рабочим дням)
 или на сайте Педагогического университета «Первое сентября» <http://edu.1september.ru>
 (последнее предпочтительнее, после подачи заявки с вами свяжется сотрудник Педуниверситета)



Устройства ввода и вывода. Теория относительности

► Главные части компьютера — *процессор* и *память*. Процессор выполняет алгоритмы, а память хранит информацию.

Все другие устройства обеспечивают или *ввод* информации в память компьютера, или *вывод* из нее. Соответственно, все другие устройства, подключаемые к процессору и памяти, являются устройствами ввода и (или) вывода.

Что принтер есть устройство вывода компьютера, а сканер — устройство ввода, никто не возражает. А вот в то, что, например, флешка есть устройство ввода/вывода, обычно не верят (“неправда, флешка — запоминающее устройство”).

Хотя на самом деле флешка и МФУ (принтер+сканер) функционально друг от друга не отличаются — они являются для компьютера устройствами ввода/вывода информации.

Заметка объясняет авторскую позицию.

Что есть компьютер

Компьютер не всегда выглядит как системный блок, монитор, мышь и клавиатура. Ноутбук, например, представляет собой единое устройство, в которое интегрированы и монитор, и клавиатура, и тачпад (вместо мыши).

Компьютер в стиральной машине и вовсе не такой, у него нет привычной клавиатуры, мыши и монитора (только переключатели, кнопки и маленький ЖК-индикатор в некоторых моделях).

Получается, монитор, клавиатура и мышь для компьютера необязательны (сразу заметим, что жесткий диск, дисковод, флешка тоже факультативны).

А что же есть компьютер?

Чтобы это узнать, давайте убирать устройства, пока не получим минимальный набор, без которого компьютера точно не бывает.

Убираем принтеры, сканеры, модемы, флешки, дисководы.

Убираем жесткий диск (в компьютере телефона, например, его нет).

Убираем клавиатуру, монитор и мышку (их не имеет компьютер стиральной машины, компьютер внутри робота).

А.А. Дуванов,
г. Переславль-Залесский,
kurs@robotland.pereslavl.ru

Н.Д. Шумилина,
г. Тверь,
nshumilina@yandex.ru

Что же остается? Остается *процессор* (обработчик) и *внутренняя память* (хранилище).

Их убрать нельзя.

Компьютер — это устройство для обработки информации с помощью программ. Программы выполняет *процессор*. Обработываемая информация, как и сама программа, располагается во *внутренней памяти*.

Итак, *компьютер* — это *процессор и внутренняя память*.

Процессор объединяет в себе два устройства — исполнительное (АЛУ, арифметико-логическое устройство) и управляющее (УУ, устройство управления). Процессор выполняет команды программы. Управление (выбор команды, ее анализ, загрузка операндов) возлагается на УУ, выполнение — на АЛУ.

Внутренняя память — это: ОЗУ + ПЗУ + регистры процессора + кэш процессора.

ОЗУ (оперативное запоминающее устройство)— эта память доступна процессору непосредственно, кроме того, в ней хранятся программы, по которым процессор работает. Содержимое ОЗУ при отключении питания пропадает.

ПЗУ (постоянное запоминающее устройство) хранит программу первоначальной загрузки компьютера вместе с данными, необходимыми для работы этой программы. Понятно, что эта память не стирается при выключении питания.

Кроме того, в компьютере есть *регистры* и *кэш* — дополнительная к ОЗУ память, встроенная прямо в микросхему процессора.

Регистры — это сверхбыстрая небольшая память. Команды, в которых операнды расположены в регистрах, выполняются гораздо быстрее команд, операнды которых расположены в ОЗУ.

Кэш — это также сверхбыстрая память, в которую предварительно загружаются операнды команд и команды программы из ОЗУ. В дальнейшем, если операнд (или команда) находится в кэше, он (она) читается из кэша, а не из ОЗУ, что ускоряет выполнение программы.

Замечание. Отметим, что внедрение регистров и кэш-памяти в микросхему процессора не делает их функциональной частью процессора. Процессор — *обработчик*, а не хранилище информации. Как, например, расположение кнопки сигнала на руле не делает руль устройством подачи звукового сигнала, руль — это устройство для поворота и удержания на курсе транспортного средства.

Что есть устройства ввода и вывода

Итак, компьютер — это процессор и внутренняя память (ОЗУ, ПЗУ, регистры и кэш процессора).

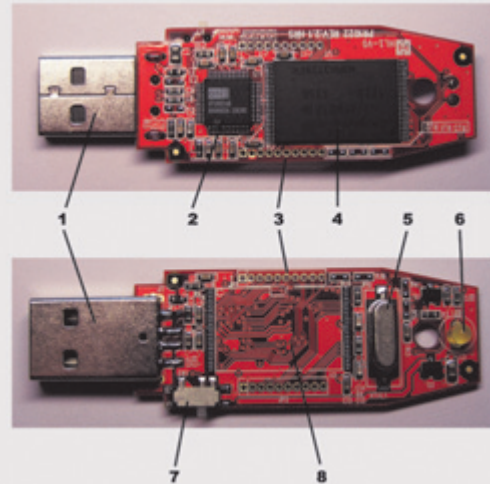
А что же тогда все остальное?

Для компьютера все остальное — *устройства ввода и вывода*.



Когда принтер называют устройством вывода, никто не возражает. Когда говорят, что флешка является устройством ввода/вывода, возникают вопросы.

Флешка, действительно, запоминающее устройство! В корпусе флешки есть микросхема флеш-памяти.



Устройство типичного USB Flash-носителя (показано изделие фирмы Saitek, иллюстрация из Википедии):

- 1 — USB-разъем;
- 2 — контроллер;
- 3 — контрольные точки;
- 4 — микросхема Flash-памяти;
- 5 — кварцевый резонатор;
- 6 — светодиод;
- 7 — переключатель “Защита от записи”;
- 8 — место для дополнительной микросхемы памяти

Но флешка, как и жесткий диск, *совмещает* в одном корпусе информационный носитель (*микросхема памяти*) и устройство чтения/записи данных (*контроллер*, фактически маленький компьютер, с программой чтения/записи, расположенной в его ПЗУ).

Если нужно сохранить результаты вычислений на флешке, мы должны *вывести* на нее данные из ОЗУ, то есть флешка выступает устройством *вывода* компьютера.



Все происходит аналогично работе принтера. Принтер — это устройство, которое позволяет сохранять информацию на бумаге. Но прежде чем информация будет отпечатана, она должна быть передана в принтер из компьютера. Принтер по отношению к компьютеру выступает устройством вывода, хотя он является и устройством сохранения информации на бумаге.

Бумагу из принтера мы вынимаем, чтобы использовать, а память из флешки мы не вынимаем, мы носим ее вместе с флешкой. Но суть не меняется. Можно представить себе принтер, из которого бумага бы не вынималась. Ее можно было бы просматривать через окошко. Тогда принтер мы таскали бы с собой, как флешку.

Флешка и принтер между собой принципиально не отличаются. Для компьютера это устройства ввода/вывода. Хотя, конечно, они же являются и устройствами хранения информации. Процесс хранения не связан с компьютером, а процесс получения информации — связан.



Когда мы говорим о вводе/выводе по отношению к компьютеру, то имеем в виду обмен данными между внутренней памятью компьютера и другим информационным носителем: жестким диском, оптическим диском, флеш-памятью, бумагой.

Работа с внешней памятью для компьютера — это операция ввода/вывода, которую он выполняет через контроллер внешнего запоминающего устройства.



Дисковод оптических дисков еще более похож на принтер и сканер, чем флешка, ибо информационный носитель (диск) является сменным (как бумага в принтере и сканере).

Итак, итог.

Компьютер — это процессор и внутренняя память. Все остальное, что к компьютеру подключается, является для него устройствами ввода/вывода, через которые течет информация в компьютер или из него.

Причина неверного толкования

Казалось бы, все просто! Однако вопросы возникают. Почему?

Причина в том, как изображается схема компьютера в школьных учебниках. Вот как изображается классический вариант фон Неймана:



По этой схеме легко отнести принтер к устройству вывода, клавиатуру — к устройству ввода, а флешку, само собой, — к внешней памяти компьютера.

На этой схеме внешняя память никак не связана с устройствами ввода/вывода, а сами устройства ввода/вывода никак не связаны с внешней памятью: откуда же они вводят и куда выводят?

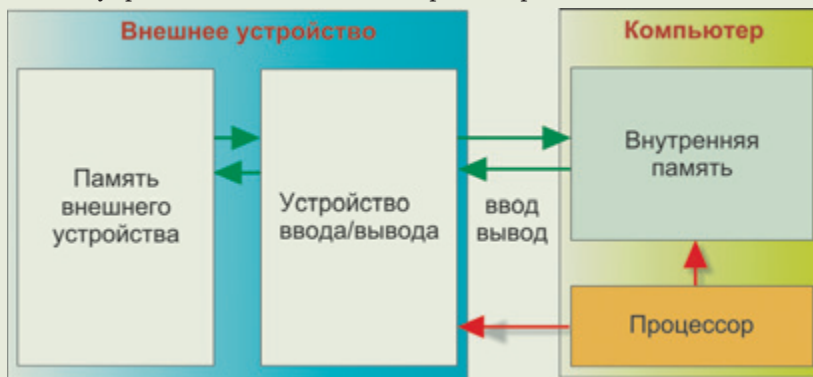
Получается, что для записи на диск последний нужно просто положить на системный блок, и компьютер запишет информацию на диск при помощи волшебства.

В жизни так не получается, и внешняя память стыкуется с компьютером при помощи специального устройства (объединяющего электронную и, возможно, механические части), которое обеспечивает чтение и запись информации на информационный носитель. Было бы логично именно это устройство называть устройством ввода/вывода.

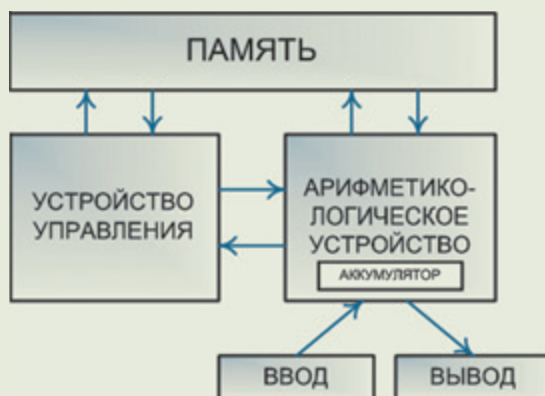
Таким устройством, например, является дисковод. Это устройство объединяет механическую часть и электронную, а сам диск является носителем информации, внешней памятью по отношению к компьютеру.

Когда внешняя память и устройство ввода/вывода собраны в одном корпусе, как флешка или винчестер, мы можем называть это объединение и внешней памятью, и устройством ввода/вывода, ибо верно и то и другое. Для того чтобы сохранить во внешней памяти информацию, ее туда надо передать, а для этого необходимо выполнить операцию вывода информации из компьютера, а значит, в момент передачи информации устройство является устройством вывода.

В момент чтения информации — это устройство ввода. На схеме зеленые стрелки показывают движение информации, а красные — управляющие воздействия процессора:



Отметим, что у самого фон Неймана блока с “внешней памятью” в схеме не было (схема скопирована из Википедии).



Схему современного компьютера в школьных учебниках часто рисуют так:



Очевидно, что долговременная память, например диск CD или бумага, подключается к общей шине только через устройство ввода/вывода. Поэтому правильная схема должна выглядеть так:



Таким образом, флешка для компьютера является устройством ввода/вывода, ибо компьютер работает с контроллером флешки. Контроллер выполняет операции чтения/записи на микросхему памяти флешки.

Теория относительности ввода и вывода

Итак, мы выяснили, что компьютер — это процессор плюс внутренняя память (ОЗУ, ПЗУ, регистры и кэш процессора).



Все остальное — это устройства ввода, вывода или ввода/вывода компьютера.



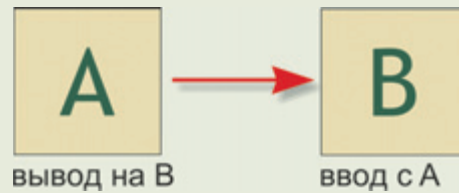
Например, мышь, клавиатура, сканер, тачпад, графический планшет, веб-камера, микрофон, датчики — это *устройства ввода* компьютера (с них он *получает* информацию), а монитор, принтер, звуковые колонки, исполнительные устройства — это *устройства вывода* компьютера (на них он *передает* информацию).



Есть устройства, с которых компьютер может и *получать* информацию и на которые он может информацию *передать*. Эти устройства называют устройствами *ввода/вывода* — винчестер, дисковод, МФУ, флешка, модем, фото- и видеокамеры.

Давайте подробнее разберемся с тем, что мы называем *вводом* и *выводом*.

Ввод и *вывод* связаны с передачей информации от одного информационного носителя к другому.



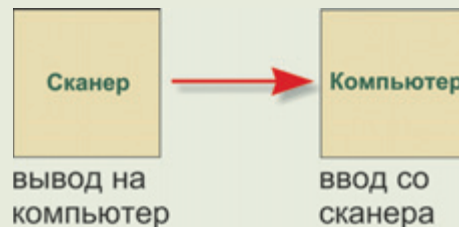
Пусть информация передается с носителя А к носителю В.

Тогда эта передача для носителя А означает *вывод*, а для носителя В — *ввод*.

И получается, что В для А — устройство *вывода*, а А для В — устройство *ввода*.

Видим, что одна и та же передача информации связана и с *вводом*, и с *выводом*, следовательно, может быть названа и *вводом*, и *выводом* в зависимости от того, к какому носителю мы относим эту операцию.

Пусть, например, мы используем сканер для передачи в компьютерную память образа бумажного листа.



Для сканера эта передача означает *вывод* — информация уходит из сканера, а для компьютера *ввод* — информация приходит в компьютер.

Компьютер получает из сканера информацию, значит, происходит *ввод* в компьютер, следовательно, сканер — устройство *ввода* компьютера.

А для самого сканера компьютер является устройством *вывода*! Ведь сканер на него посылает, выводит информацию.

Так что же есть сканер? Для компьютера это устройство *ввода*.

А для человека? А для человека сканер — устройство *вывода*. Мы ведь отдаем ему листок, “*выводя*” из папки, в которой листок хранили.

Но мы никогда не говорим, что выводим листок на сканер. Мы говорим, наоборот, что *вводим* листок, то есть относим процесс передачи информации не к себе, а к сканеру. Более того, мы часто говорим, вставляя листок в сканер, что *вводим* листок в компьютер, и здесь мы говорим уже не о передаче информации от нас сканеру, а о передаче информации от сканера в компьютер, и связываем эту передачу с компьютером.

Любая передача информации сопровождается *выводом* из *источника* и *вводом* в *приемник*. Следовательно, разделение устройств на устройства



ввода и устройства вывода зависит от того, по отношению к какому информационному носителю мы рассматриваем процесс передачи информации.

Для компьютера классификация построена на направлении потока информации *в* или *из* компьютерной памяти.

Ввод — это процесс передачи информации *в компьютер*. Устройство, с которого компьютер получает информацию, является *устройством ввода компьютера*.

Вывод — процесс передачи информации *из компьютера*. Устройство, на которое компьютер передает информацию, является *устройством вывода компьютера*.

Два компьютера (см. рисунок). Их можно соединить друг с другом через порты кабелем без модема и передавать информацию в обе стороны.

Что происходит, ввод или вывод, зависит от того, по отношению к какому компьютеру мы рассматриваем процесс передачи информации.

Из всего этого следует, что на уроке в классе лучше говорить не “колонки выводят звук”, а “колонки воспроизводят звук, который на них выводит компьютер”. Обе фразы правильные, но в них говорится о двух разных передачах информации:

1. Передаче звука от компьютера на колонки (*выводит компьютер*).
2. Передаче звука от колонок к человеку (*выводят колонки*).

Во втором случае колонки для нас являются устройством ввода (мы вводим с них звук в свои уши), а мы для колонок — устройством вывода — колонки выводят на нас звук.

Так как мы рассматриваем устройства ввода и вывода *компьютера*, то лучше говорить “звук *выводится компьютером на колонки*, а колонки *звук воспроизводят*”.

Как не стоит говорить	А как надо сказать
Монитор выводит информацию	Компьютер выводит информацию на монитор
Клавиатура вводит информацию	Компьютер вводит информацию с клавиатуры
Принтер выводит информацию	Компьютер выводит информацию на принтер

И так далее.

Завершим изложение темы набором формальных определений, связанных с вводом и выводом.

Формальные определения

Определение 1

Передача информации — это процесс, в результате которого информация с носителя А передается на носитель В по каналу связи. Обозначим это как $A \rightarrow B$.

Определение 2

Пусть $A \rightarrow B$. Будем говорить, что информация *выводится* с носителя А и *вводится* на носитель В.

Замечание. Понятно, что сам носитель не способен передавать информацию в канал связи и записывать сам на себя информацию из канала связи. Для такого чтения/записи используется дополнительное устройство (искусственное или естественное).

Определение 3

Назовем устройство, обеспечивающее чтение/запись информации на носитель, *контроллером*. И будем обозначать контроллеры маленькими буквами: а — контроллер носителя А, б — контроллер носителя В.

Для обозначения устройства “носитель + контроллер” будем использовать обозначение вида Аа (или Вб).

Замечание. В связке Аа А — хранит информацию, а — организует прием/передачу.

Определение 4

Назовем Сс *устройством вывода* для Dd, если информация передается из D в С.

Определение 5

Назовем Сс *устройством ввода* для Dd, если информация передается из С в D.

Пусть $A \rightarrow B$. Видим, что:

Аа — устройство ввода Вб,

Вб — устройство вывода Аа.



Системы логических уравнений

К.Ю. Поляков,
д. т. н., учитель
информатики школы № 163,
Санкт-Петербург

► Все школьники на уроках математики знакомятся с алгебраическими и трансцендентными уравнениями. Более экзотические логические уравнения, в которых все величины могут принимать только два значения (“истина” и “ложь”), традиционно тоже относились к математике [1–3]. На практике они оказались полезны при разработке цифровых логических устройств [4–5] и в последние годы стали появляться в заданиях ЕГЭ по информатике. Так как ни в одном школьном учебнике соответствующего материала нет, учителю приходится разбираться в этом вопросе самостоятельно. В данной статье мы попытаемся немного прояснить ситуацию.

Далее для сокращения записи истинное значение обозначается единицей, а ложное — нулем. Для логических операций “И”, “ИЛИ”, “исключающее ИЛИ”, “импликация” и “эквивалентность” будем использовать знаки “·”, “+”, “⊕”, “→” и “≡” соответственно, а операцию “НЕ” обозначим чертой сверху.

Решение логических задач

Начнем с логической задачи, которая сводится к системе логических уравнений.

Задача 1. Следующие два высказывания истинны:

(1). Неверно, что если корабль А вышел в море, то корабль С — нет.

(2). В море вышел корабль В или корабль С, но не оба вместе.

Определить, какие корабли вышли в море.

Обозначим буквами высказывания:

А — “корабль А вышел в море”,

В — “корабль В вышел в море”,

С — “корабль С вышел в море”.

Тогда высказывание “если корабль А вышел в море, то корабль С — нет”

можно записать в виде $A \rightarrow \bar{C} = 1$. По условию (1), это высказывание неверно, таким образом, имеем $A \rightarrow \bar{C} = 0$. Кроме того, из (2) получаем $A \oplus B = 1$. Итак, решение задачи сводится к решению системы логических уравнений

$$\begin{cases} A \rightarrow \bar{C} = 0 \\ A \oplus B = 1 \end{cases}$$

Нужно найти тройку логических значений А, В и С, при которых оба уравнения превращаются в истинные равенства. Покажем несколько способов решения этой системы.

Способ 1. Сведение к одному уравнению. Сначала преобразуем уравнения так, чтобы правые части были равны 1 (истинному значению). Для этого применим инверсию (операцию “НЕ”) к обеим частям первого уравнения:

$$\begin{cases} \overline{A \rightarrow \bar{C}} = 1 \\ A \oplus B = 1 \end{cases}$$

Теперь представляем импликацию и “исключающее ИЛИ” через базовые операции (“И”, “ИЛИ”, “НЕ”):

$$\begin{cases} \overline{\bar{A} + \bar{C}} = 1 \\ \bar{A} \cdot B + A \cdot \bar{B} = 1 \end{cases}$$

Поскольку необходимо, чтобы левые части обоих уравнений были равны 1, можно объединить их с помощью операции “И” в одно уравнение, равносильное исходной системе:

$$(\overline{\bar{A} + \bar{C}}) \cdot (\bar{A} \cdot B + A \cdot \bar{B}) = 1$$

Раскрываем инверсию в первой скобке по закону де Моргана и вносим “сомножитель” A в скобку, не забыв, что $A \cdot \bar{A} = 0$ и $A \cdot A = A$:

$$A \cdot C \cdot (\bar{A} \cdot B + A \cdot \bar{B}) = 1 \Leftrightarrow A \cdot \bar{B} \cdot C = 1$$

Последнее уравнение, равносильное исходной системе, имеет единственное решение: $A = 1$, $B = 0$ и $C = 1$. Таким образом, в море вышли корабли A и C .

Способ 2. Таблица истинности. Поскольку логические величины имеют только два значения, можно просто перебрать все варианты и найти среди них те, при которых выполняется данная система уравнений. Для системы с тремя переменными возможны 8 вариантов:

A	B	C	$A \rightarrow \bar{C}$	$A \oplus B$
0	0	0	1	0
0	0	1	1	0
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	0	1
1	1	0	1	0
1	1	1	0	0

Зеленым цветом выделена единственная строчка, для которой выполняются условия задачи, то есть $A \rightarrow \bar{C} = 0$ и $A \oplus B = 1$. Таким образом, $A = 1$, $B = 0$ и $C = 1$, это значит, что в море вышли корабли A и C . Недостаток этого метода — трудоемкость при большом количестве переменных (больше 4).

Заметим, что такой подход особенно удобен при автоматизации решения подобных задач. Именно он (совместно с приведением системы к одному уравнению) используется в программе для решения систем логических уравнений, которая размещена на диске-приложении к этому номеру.

Способ 3. Декомпозиция. Идея состоит в том, чтобы зафиксировать значение одной из переменных (положить ее равной 0 или 1) и за счет этого

упростить уравнения. Затем можно зафиксировать значение второй переменной и т.д. В нашем случае при $A = 0$ получаем

$$\begin{cases} 0 \rightarrow \bar{C} = 0 \\ 0 \oplus B = 1 \end{cases}$$

Известно, что импликация ложна только в одном случае — когда посылка истинна, а следствие ложно ($1 \rightarrow 0 = 0$). Поэтому первое уравнение (и, следовательно, вся система) при $A = 0$ решений не имеет. Теперь рассмотрим второй случай, когда $A = 1$:

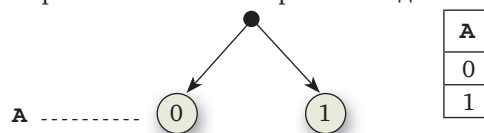
$$\begin{cases} 1 \rightarrow \bar{C} = 0 \\ 1 \oplus B = 1 \end{cases}$$

Из первого уравнения в силу свойств импликации сразу следует, что $\bar{C} = 0$ или $C = 1$, а из второго — $B = 0$ ($A \oplus B = 1$ означает, что $A \neq B$). Таким образом, существует единственное решение системы: $A = 1$, $B = 0$ и $C = 1$; это значит, что в море вышли корабли A и C .

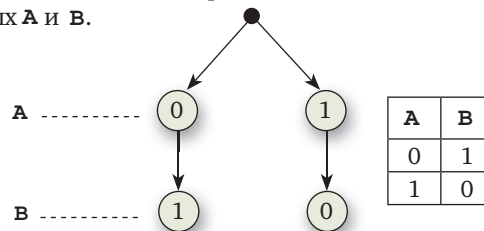
Способ 4. Последовательное решение уравнений. В некоторых случаях можно решать уравнения последовательно, на каждом шаге добавляя по одной переменной в рассматриваемый набор. Чтобы вводить переменные в алфавитном порядке, поменяем местами уравнения в нашей системе:

$$\begin{cases} A \oplus B = 1 \\ A \rightarrow \bar{C} = 0 \end{cases}$$

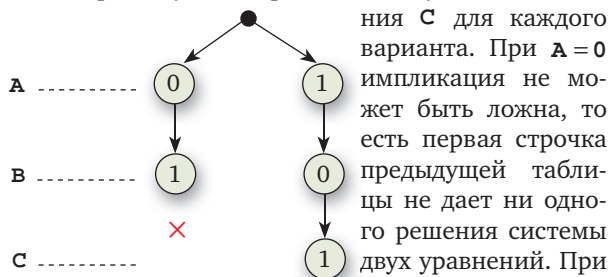
Мы видим, что первое уравнение зависит только от A и B , а в третьем подключается переменная C . Будем строить дерево решений и одновременно записывать его в свернутом виде в таблицу. Переменная A может принимать два значения:



Из первого уравнения $A \oplus B = 1$ следует, что $A \neq B$, поэтому при $A = 0$ получаем единственный вариант $B = 1$, а при $A = 1$ имеем $B = 0$. Итак, первое уравнение имеет два решения относительно переменных A и B .



Теперь “подключаем” второе уравнение $A \rightarrow \bar{C} = 0$, из которого нужно определить допустимые значения C для каждого варианта. При $A = 0$ импликация не может быть ложна, то есть первая строчка предыдущей таблицы не дает ни одного решения системы двух уравнений. При



$A = 1$ получаем единственное решение, для которого $C = 1$:

A	B	C
1	0	1

Таким образом, существует единственное решение системы: $A = 1$, $B = 0$ и $C = 1$. Это значит, что в море вышли корабли **A** и **C**.

Количество решений

Иногда требуется определить только количество решений системы логических уравнений, при этом находить сами решения не нужно.

Задача 2. Найти количество решений системы уравнений:

$$\begin{cases} \bar{x}_1 + x_2 = 1 \\ \bar{x}_2 + x_3 = 1 \\ \dots \\ \bar{x}_9 + x_{10} = 1 \end{cases}$$

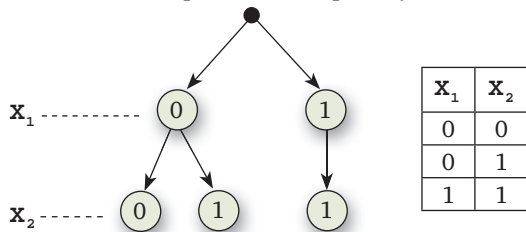
где $x_1 \dots x_{10}$ — неизвестные логические величины.

Здесь 10 переменных, поэтому при решении системы через таблицу истинности нужно заполнить $2^{10} = 1024$ строки, что трудновыполнимо. Поэтому решения, сводящиеся к полному перебору вариантов, нужно отбросить. Поскольку все правые части равны 1, можно легко свести систему к одному уравнению:

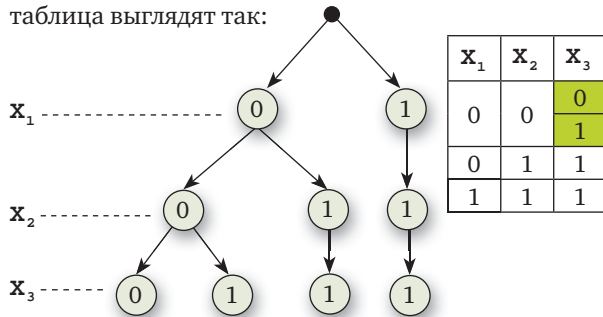
$$(\bar{x}_1 + x_2) \cdot (\bar{x}_2 + x_3) \cdot \dots \cdot (\bar{x}_9 + x_{10}) = 1,$$

но такой подход также не внушает оптимизма.

Несложно заметить, что первое уравнение зависит только от x_1 и x_2 , затем во втором уравнении добавляется x_3 и т.д. Поэтому логично попробовать решать уравнения последовательно. Начнем строить дерево решений, и одновременно будем записывать его в виде таблицы. Первое уравнение, $\bar{x}_1 + x_2 = 1$, обращается в истинное равенство в трех случаях:



Теперь подключаем второе уравнение, $\bar{x}_2 + x_3 = 1$. Допустимые значения x_3 зависят от ранее выбранного значения x_2 : если $x_2 = 0$, то x_3 может принимать любое значение (0 или 1), а если $x_2 = 1$, то $x_3 = 1$. Дерево и соответствующая ему таблица выглядят так:



Легко заметить, что при добавлении очередного уравнения (и очередной переменной) верхняя строка таблицы (где все нули) дает два решения (они выделены зеленым фоном), а остальные строки — по одному. Поэтому количество решений увеличивается на 1. Таким образом, система из трех уравнений имеет 5 решений, из четырех — 6, а исходная система из девяти уравнений — 11 решений.

Рассмотрим более сложный пример на ту же тему.

Задача 3. Найти количество решений системы уравнений:

$$\begin{cases} (x_2 \equiv x_1) + x_2 \cdot x_3 + \bar{x}_2 \cdot \bar{x}_3 = 1 \\ (x_3 \equiv x_1) + x_3 \cdot x_4 + \bar{x}_3 \cdot \bar{x}_4 = 1 \\ \dots \\ (x_9 \equiv x_1) + x_9 \cdot x_{10} + \bar{x}_9 \cdot \bar{x}_{10} = 1 \\ (x_{10} \equiv x_1) = 0 \end{cases}$$

где $x_1 \dots x_{10}$ — неизвестные логические величины.

Здесь, так же как и в предыдущей задаче, удобнее всего последовательно решать уравнения и записывать полученные решения в таблицу (дерево рисовать не будем для сокращения записи). Сначала упростим исходные уравнения, заметив, что $x_2 \cdot x_3 + \bar{x}_2 \cdot \bar{x}_3 = (x_2 \equiv x_3)$, так что исходную систему можно записать в виде

$$\begin{cases} (x_2 \equiv x_1) + (x_2 \equiv x_3) = 1 \\ (x_3 \equiv x_1) + (x_3 \equiv x_4) = 1 \\ \dots \\ (x_9 \equiv x_1) + (x_9 \equiv x_{10}) = 1 \\ (x_{10} \equiv x_1) = 0 \end{cases}$$

В первом уравнении используются три переменных ($x_1 \dots x_3$). Значения x_1 и x_2 могут быть выбраны произвольно четырьмя способами:

x_1	x_2
0	0
0	1
1	0
1	1

Теперь добавляем x_3 и первое уравнение как ограничение. Запишем в таблицу все комбинации переменных, при которых выполняется первое уравнение:

x_1	x_2	x_3
0	0	0
0	0	1
0	1	1
1	0	0
1	1	0
1	1	1

Если $x_2 = x_1$, то значение x_3 может быть любое (эти строки выделены зеленым цветом), а при $x_2 \neq x_1$ получаем только один вариант: $x_3 = x_2$.

Таким образом, при подключении первого уравнения число решений увеличивается на количество строк в таблице, для которых значения X_1 и X_2 (последней рассмотренной переменной) равны. В данном случае таких строк две, получаем 6 решений. Более того, в новой таблице снова осталось всего две строки (верхняя и нижняя), где $X_3 = X_1$. Как следует из второго уравнения, именно эти (и только эти) строки на следующем шаге “раздваиваются”, дают по два решения. Таким образом, при подключении к системе очередного уравнения число решений увеличивается на 2. Для двух уравнений получим 8 решений, для трех — 10, а для восьми — 20 решений.

Остается учесть последнее (особое) уравнение, $(X_{10} \equiv X_1) = 0$. Это означает, что $X_{10} \neq X_1$. Из анализа таблицы видно, что есть всего две строки (верхняя и нижняя), где первая и последняя переменные равны. Поэтому из полученных 20 решений нужно отбросить эти два, не удовлетворяющие последнему уравнению. В итоге исходная система имеет 18 решений.

Замена переменных

В некоторых случаях задача упрощается, если использовать замену переменных.

Задача 4. Найти количество решений системы уравнений:

$$\begin{cases} X_1 \cdot \bar{X}_2 + \bar{X}_1 \cdot X_2 + X_3 \cdot X_4 + \bar{X}_3 \cdot \bar{X}_4 = 1 \\ X_3 \cdot \bar{X}_4 + \bar{X}_3 \cdot X_4 + X_5 \cdot X_6 + \bar{X}_5 \cdot \bar{X}_6 = 1 \\ X_5 \cdot \bar{X}_6 + \bar{X}_5 \cdot X_6 + X_7 \cdot X_8 + \bar{X}_7 \cdot \bar{X}_8 = 1 \\ X_7 \cdot \bar{X}_8 + \bar{X}_7 \cdot X_8 + X_9 \cdot X_{10} + \bar{X}_9 \cdot \bar{X}_{10} = 1 \end{cases}$$

где $X_1 \dots X_{10}$ — неизвестные логические величины.

Сначала упростим эти уравнения, используя свойства операций “исключающее ИЛИ” и “эквивалентность”. В первом уравнении

$$X_1 \cdot \bar{X}_2 + \bar{X}_1 \cdot X_2 = (X_1 \oplus X_2) = \overline{(X_1 \equiv X_2)}$$

$$X_3 \cdot X_4 + \bar{X}_3 \cdot \bar{X}_4 = (X_3 \equiv X_4)$$

Применяя подобные преобразования во всех уравнениях, получаем

$$\begin{cases} \overline{(X_1 \equiv X_2)} + (X_3 \equiv X_4) = 1 \\ (X_3 \equiv X_4) + (X_5 \equiv X_6) = 1 \\ \overline{(X_5 \equiv X_6)} + (X_7 \equiv X_8) = 1 \\ (X_7 \equiv X_8) + (X_9 \equiv X_{10}) = 1 \end{cases}$$

Далее замечаем, что можно ввести новые переменные

$$Y_1 = (X_1 \equiv X_2), \quad Y_2 = (X_3 \equiv X_4)$$

$$Y_3 = (X_5 \equiv X_6), \quad Y_4 = (X_7 \equiv X_8)$$

$$Y_5 = (X_9 \equiv X_{10})$$

и система уравнений принимает вид

$$\begin{cases} \bar{Y}_1 + Y_2 = 1 \\ \bar{Y}_2 + Y_3 = 1 \\ \bar{Y}_3 + Y_4 = 1 \\ \bar{Y}_4 + Y_5 = 1 \end{cases}$$

Важно, что переменные $Y_1 \dots Y_5$ независимы, то есть каждая из исходных переменных $X_1 \dots X_{10}$ входит только в одну из новых переменных.

Полученная система совпадает по форме с системой, которая рассматривалась в задаче 2. Используя результаты решения задачи 2, сразу находим, что наша система из четырех уравнений имеет 6 решений относительно переменных $Y_1 \dots Y_5$.

Остается вернуться обратно к исходным переменным $X_1 \dots X_{10}$. Предположим, что значение $Y_1 = (X_1 \equiv X_2)$ фиксированно (0 или 1). Тогда, согласно таблице истинности операции “эквивалентность”, существует ровно две пары значений (X_1, X_2) , при которых Y_1 имеет заданное значение (как для $Y_1 = 0$, так и для $Y_1 = 1$). Таким образом, каждая комбинация значений $Y_1 \dots Y_5$ дает по две возможных пары (X_1, X_2) , (X_3, X_4) , (X_5, X_6) , (X_7, X_8) и (X_9, X_{10}) , то есть всего $2^5 = 32$ комбинации исходных переменных.

Таким образом, общее количество решений равно $6 \cdot 32 = 192$.

Решение — функция

Рассмотрим более сложную задачу, в которой требуется найти логическую функцию, удовлетворяющую заданному уравнению (или системе уравнений). Задачи такого типа могут встречаться на практике при разработке схем логического управления [4–5].

Задача 5. Найти логическую функцию $X(A, B)$, которая при любых значениях логических переменных A и B удовлетворяет системе уравнений:

$$\begin{cases} X + A = \bar{X} \cdot A + B \\ A + B + \bar{X} = 1 \end{cases}$$

Заметим, что классические методы решений систем уравнений, известные из математики, здесь не работают, потому что для логических величин не определено вычитание и деление.

Такие системы проще всего решать с помощью таблиц истинности. Для каждой возможной комбинации значений переменных A и B получаем систему уравнений с одним неизвестным X :

A	B	Система уравнений	X
0	0	$\begin{cases} X = 0 \\ \bar{X} = 1 \end{cases}$	0
0	1	$\begin{cases} X = 1 \\ 1 = 1 \end{cases}$	1
1	0	$\begin{cases} 1 = \bar{X} \\ 1 = 1 \end{cases}$	0
1	1	$\begin{cases} 1 = 1 \\ 1 = 1 \end{cases}$	0 или 1

Заметим, что в последней строчке значение X может быть любым. Поэтому исходной системе

удовлетворяют две логические функции, которые задаются таблицами истинности

A	B	X ₁
0	0	0
0	1	1
1	0	0
1	1	0

A	B	X ₂
0	0	0
0	1	1
1	0	0
1	1	1

В первом случае получаем $X_1 = \bar{A} \cdot B$, а во втором — $X_2 = \bar{A} \cdot B + A \cdot B = B$. Исходная система уравнений имеет два решения.

Логические уравнения такого типа не всегда имеют решение в двузначной логике. Например, уравнение $X + A = \bar{X} + B$ не имеет решений, потому что при $A = B = 0$ получаем $X = \bar{X}$, что невыполнимо.

Задачи для тренировки:

1) На вопрос “Кто из твоих учеников изучал логику?” учитель ответил: “Если логику изучал Андрей, то изучал и Борис. Однако неверно, что если изучал Семен, то изучал и Борис”. Составьте систему логических уравнений и определите, кто же изучал логику. (Ответ: только Семен.)

2) Суд присяжных пришел к таким выводам: а) если Аськин не виновен или Баськин виновен, то виновен Сенькин; б) если Аськин не виновен, то Сенькин не виновен. Составьте систему логических уравнений, решите ее и сделайте выводы о виновности трех подозреваемых. (Ответ: Аськин точно виновен, про остальных ничего определенного сказать нельзя.)

3) Прогноз погоды выглядит так: “Если не будет ветра, то будет пасмурная погода без дождя. Если будет дождь, то будет пасмурно и без ветра. Если будет пасмурная погода, то будет дождь и не будет ветра”. Составьте и решите систему логических уравнений и определите, какая погода может быть завтра. (Ответ: ясно, ветер, без осадков.)

4) Сколько различных решений имеет система уравнений

$$\begin{cases} (X_1 \equiv X_2) \cdot (X_3 \equiv X_4) + \overline{(X_1 \equiv X_2)} \cdot \overline{(X_3 \equiv X_4)} = 0 \\ (X_3 \equiv X_4) \cdot (X_5 \equiv X_6) + \overline{(X_3 \equiv X_4)} \cdot \overline{(X_5 \equiv X_6)} = 0 \\ (X_5 \equiv X_6) \cdot (X_7 \equiv X_8) + \overline{(X_5 \equiv X_6)} \cdot \overline{(X_7 \equiv X_8)} = 0 \\ (X_7 \equiv X_8) \cdot (X_9 \equiv X_{10}) + \overline{(X_7 \equiv X_8)} \cdot \overline{(X_9 \equiv X_{10})} = 0 \end{cases}$$

где $X_1 \dots X_{10}$ — логические переменные?
(Ответ: 64)

5) Сколько различных решений имеет система уравнений

$$\begin{cases} X_1 \cdot X_2 + \bar{X}_1 \cdot \bar{X}_2 + (X_1 \equiv X_3) = 1 \\ X_2 \cdot X_3 + \bar{X}_2 \cdot \bar{X}_3 + (X_2 \equiv X_4) = 1 \\ \dots \\ X_7 \cdot X_8 + \bar{X}_7 \cdot \bar{X}_8 + (X_7 \equiv X_9) = 1 \\ X_8 \cdot X_9 + \bar{X}_8 \cdot \bar{X}_9 + (X_8 \equiv X_{10}) = 1 \end{cases}$$

где $X_1 \dots X_{10}$ — логические переменные?
(Ответ: 20)

6) Сколько различных решений имеет система уравнений

$$\begin{cases} X_1 \cdot X_2 + \bar{X}_1 \cdot \bar{X}_2 + (X_1 \equiv X_3) = 1 \\ X_2 \cdot X_3 + \bar{X}_2 \cdot \bar{X}_3 + (X_2 \equiv X_4) = 1 \\ \dots \\ X_7 \cdot X_8 + \bar{X}_7 \cdot \bar{X}_8 + (X_7 \equiv X_9) = 1 \\ X_8 \cdot X_9 + \bar{X}_8 \cdot \bar{X}_9 + (X_8 \equiv X_{10}) = 0 \end{cases}$$

где $X_1 \dots X_{10}$ — логические переменные?
(Ответ: 16)

7) Сколько различных решений имеет система уравнений

$$\begin{cases} X_1 \cdot X_2 + \bar{X}_1 \cdot \bar{X}_2 + X_2 \cdot X_3 + \bar{X}_2 \cdot \bar{X}_3 = 1 \\ X_2 \cdot X_3 + \bar{X}_2 \cdot \bar{X}_3 + X_3 \cdot X_4 + \bar{X}_3 \cdot \bar{X}_4 = 1 \\ \dots \\ X_8 \cdot X_9 + \bar{X}_8 \cdot \bar{X}_9 + X_9 \cdot X_{10} + \bar{X}_9 \cdot \bar{X}_{10} = 1 \end{cases}$$

где $X_1 \dots X_{10}$ — логические переменные?
(Ответ: 178)

8) Сколько различных решений имеет система уравнений

$$\begin{cases} X_1 \cdot X_2 + \bar{X}_1 \cdot \bar{X}_2 + X_2 \cdot X_3 + \bar{X}_2 \cdot \bar{X}_3 = 1 \\ X_2 \cdot X_3 + \bar{X}_2 \cdot \bar{X}_3 + X_3 \cdot X_4 + \bar{X}_3 \cdot \bar{X}_4 = 1 \\ \dots \\ X_7 \cdot X_8 + \bar{X}_7 \cdot \bar{X}_8 + X_8 \cdot X_9 + \bar{X}_8 \cdot \bar{X}_9 = 1 \\ X_8 \cdot X_9 + \bar{X}_8 \cdot \bar{X}_9 + X_9 \cdot X_{10} + \bar{X}_9 \cdot \bar{X}_{10} = 0 \end{cases}$$

где $X_1 \dots X_{10}$ — логические переменные?
(Ответ: 42)

9) Сколько различных решений имеет система уравнений

$$\begin{cases} (X_1 \equiv X_2) + X_1 \cdot X_{10} + \bar{X}_1 \cdot \bar{X}_{10} = 1 \\ (X_2 \equiv X_3) + X_2 \cdot X_{10} + \bar{X}_2 \cdot \bar{X}_{10} = 1 \\ \dots \\ (X_9 \equiv X_{10}) + X_9 \cdot X_{10} + \bar{X}_9 \cdot \bar{X}_{10} = 1 \\ (X_1 \equiv X_{10}) = 0 \end{cases}$$

где $X_1 \dots X_{10}$ — логические переменные?
(Ответ: 0)

10) Сколько различных решений имеет система уравнений

$$\begin{cases} (X_1 \rightarrow X_2) + (X_1 \rightarrow X_3) = 1 \\ (X_2 \rightarrow X_3) + (X_2 \rightarrow X_4) = 1 \\ \dots \\ (X_8 \rightarrow X_9) + (X_8 \rightarrow X_{10}) = 1 \end{cases}$$

где $X_1 \dots X_{10}$ — логические переменные?
(Ответ: 232)

11) Найдите все пары логических функций $(X(A, B); Y(A, B))$, удовлетворяющие при любых значениях логических переменных A и B системе уравнений:

$$\begin{cases} X \cdot \bar{A} + B = B \cdot Y \\ Y \cdot \bar{B} + A = A \cdot X \end{cases}$$

(Ответ: $X = A$ или $X = A + B$; $Y = 0$ или $Y = A \cdot B$, причем X и Y могут выбираться независимо друг от друга.)

12) Найдите все пары логических функций $(X(A, B); Y(A, B))$, удовлетворяющие при любых значениях логических переменных A и B системе уравнений:

$$\begin{cases} X \cdot \bar{A} + B = B \cdot \bar{Y} \\ Y \cdot \bar{B} + A = A \cdot \bar{X} \end{cases}$$

(Ответ: $X = 0$ или $X = \bar{A} \cdot B$; $Y = B$ или $Y = A + B$, причем X и Y могут выбираться независимо друг от друга.)

13) Найдите все пары логических функций $(X(A, B); Y(A, B))$, удовлетворяющие при любых значениях логических переменных A и B системе уравнений:

$$\begin{cases} X \cdot \bar{A} + B \cdot \bar{X} = B \cdot Y \\ Y \cdot \bar{B} + A \cdot \bar{Y} = A \cdot X \end{cases}$$

(Ответ: Существует две группы решений по 4 решения в каждой:

I. $X = \bar{A} \cdot B$ или $X = A \oplus B$;

$Y = A$ или $Y = A + B$

II. $X = B$ или $X = A + B$;

$Y = A \cdot B$ или $Y = A \oplus B$.

В пределах каждой группы X и Y могут выбираться независимо друг от друга.)

Дополнительные материалы

1. Тарасова А.П. Системы логических уравнений. <http://festival.1september.ru/articles/416929/>.

2. Левченков В.С. Булевы уравнения. М.: Диалог МГУ, 1999.

3. Закревский А.Д. Логические уравнения. М.: УРСС, 2003.

4. Шальто А.А. Логическое управление. Методы аппаратной и программной реализации. СПб.: Наука, 2000.

5. Вальциферов Ю.Ф., Дронов В.П. Информатика. Часть I. М.: Московский государственный университет экономики, статистики и информатики, 2005.

Автор признателен О.А. Тузовой за ценные замечания, способствовавшие улучшению статьи.

Robots.txt никогда не причинит вред человеку?

► Прошедшим летом много шума наделали истории о появлении в выдаче поисковых систем персональных данных людей, производивших те или иные операции в Интернете. Речь, в частности, шла об отправке SMS с сайта оператора мобильной сети, покупке билетов и т.д. Сразу, как водится, нашлось немалое количество желающих немедленно найти и наказать виноватых. Кого-то нашли, кого-то наказали, но уверенности в том, что те, кто искал и наказывал, хорошо понимали, в чем дело, увы, нет. Так в чем же дело?

Поисковые роботы не занимаются хакерством. Они могут добраться лишь до страниц, которые находятся в открытом доступе — не закрыты теми или иными способами авторизации. Более того, они могут добраться лишь до страниц, о которых им каким-либо способом становится известно. Последнее важно. Если вы разместите на своем сайте страницу, о которой никому не скажете, и ссылку на нее никому не дадите, роботы о ней не узнают. Они не перебирают все возможные строки, выясняя, нет ли страницы с таким именем.

Такой способ “защиты” нередко используется в случаях, когда полноценная авторизация неудобна для пользователей. Пример с отправкой SMS характерен. Представьте себе, что SMS с сайта оператора смогут отправлять лишь зарегистрированные пользователи. Удобно ли это

для самих пользователей? Ведь придется сначала регистрироваться, затем каждый раз вводить логин и пароль, периодически забывать и то и другое и восстанавливать или регистрироваться заново. Поэтому в подобных случаях нередко применяют “защиту”, которая заключается в том, что, например, страница с результатами отправки SMS имеет имя подобное 545d6464ab-64436f128a199e361d73c6c18a72097 (это, разумеется, просто случайный набор цифр и букв). Случайно найти такую страницу нереально, ссылок на нее не выставляется и, как кажется, “защита” вполне надежная? Да, но... Проблема, которая возникла этим летом, была в ряде случаев (не во всех) связана с тем, что на таких страницах за чем-то находились счетчики обычных рейтинговых систем. А счетчик — та же самая ссылка! Он же, фиксируя факт посещения, передает адрес страницы!

От этого можно было бы уберечься, запретив индексацию таких страниц специальными директивами в файле robots.txt. Но и этого сделано не было.

В любом случае, производя те или иные операции в Интернете, надо понимать, что всегда имеется риск утечки данных. Полностью избежать этого нельзя — и администраторы, и программисты, и даже специалисты в области аудита систем безопасности — обычные люди. Они могут ошибиться и по отдельности, и все вместе (последнее случается редко, но, как видим, случается).

Впрочем, риск утечки данных при операциях в оффлайне ничем не меньше (больше — поскольку больше роль человеческого фактора).



Haskell: серебряная пуля современного программирования?

Что такое функциональное программирование?

И.А. Сукин,
г. Переславль-
Залесский

► Описать язык Haskell непросто. Сам основной разработчик и законодатель мод этого языка Саймон Пейтон-Джонс полагает, что главный минус Haskell — его чрезмерная полнота и непривычность для среднестатистического современного программиста. Поэтому я вынужден совершить небольшой экскурс в историю и причины появления такого явления, как функциональное программирование. Надеюсь, что это краткое путешествие позволит вам лучше проникнуть в глубины этого чисто функционального языка.

Традиционно, когда мы в современном мире употребляем термин “программирование” или “язык программирования”, мы имеем в виду программирование императивное и такие широко распространенные в наше время языки, как C, Pascal, Java, C#. У многих информатиков термин “программирование” неустанно следует вместе с понятиями объектов и объектно-ориентированного подхода. Немалое же количе-

ство программистов вообще имеет очень смутные и крайне общие представления о разделении языков и парадигм программирования.

Для тех читателей, что не стесняются относить себя к последней категории (признаться, я сам постоянно сомневаюсь, что в этом хитросплетении сущностей можно хоть как-нибудь разобраться), необходимо ввести некоторые определения и строго показать структуру современного “зверинца” языков программирования.

В современном состоянии вычислительная техника переживает некоторый кризис, связанный с принципиальной несовершенностью существующих компьютеров и отсутствием практических реализаций компьютеров другого типа, например, биологических или квантовых вычислителей. Современная ЭВМ — это машина фоннеймановского типа, нравится нам это или нет. Несмотря на то что идея таких машин считалась устаревшей уже 30 лет назад, они до сих пор являются фактически единственным известным нам способом аппаратно реализовать вычислительное устройство, обладающее приемлемыми для нас характеристиками и производительностью. Все основные императивные языки программирования, будь то C, Pascal, с оговорками Java и C#, являются моделями машины фон Неймана.

Чем же плоха такая машина и чем плохи ее модели, а также какие из возникающих при этом проблем можно устранить при помощи

функционального программирования? Напомню основную структуру машины фон Неймана: в самом общем случае у нас имеются процессор и память, подключенные к одной соединительной шине, через которую они и вынуждены общаться. Эта шина не зря издавна называется “узким местом фон Неймана”: вся основная нагрузка в машине такого типа ложится на нее. В принципе, можете сказать вы, ничего страшного в этом нет. Но вспомните свой опыт программирования, например, на языке Си, или даже на языке ассемблера: основную массу конструкций языка занимают имена, имена данных и операций, а также операции по раскрытию соответствующих имен (вспомните, например, такие вещи, как косвенная адресация) — именно эта нагрузка бесполезным грузом ложится на “узкое место фон Неймана”.

Рассмотрев главную проблему самой концепции фоннеймановской машины (тут уж мы ничего не можем сделать, реализация ЭВМ другого типа выходит за рамки данного обсуждения), обратимся к проблемам собственно языков программирования, являющихся моделями машины фон Неймана. Исследователи выделяют четыре основных критерия оценки подобных моделей: математическая обоснованность, историческая чувствительность (другими словами — степень зависимости текущего состояния модели от ее предыдущих состояний), удобство и тип семантики, ясность и концептуальная чистота программ, описываемых конкретной моделью. Я не буду подробно описывать то, что стоит за каждым из этих понятий, интересующиеся могут обратиться к соответствующей литературе, скажу лишь, что подавляющее большинство столпов нынешней computer science справедливо считают, что модели фоннеймановского типа заслуживают весьма низких оценок по этим критериям. К примеру, возьмем слова создателя языка Fortran Джона Бэкуса: “По-видимому, с языками программирования происходит что-то неладное. Всякий новый язык включает с небольшими изменениями все свойства своих предшественников плюс кое-что еще... но фактически дела обстоят так, что лишь немногие языки снижают затраты на программирование или повышают его надежность...” [1]. Это положение было высказано Бэкусом в его лекции, как раз посвященной функциональному программированию.

В заключение этих нескольких абзацев подчеркну их основную идею: современные популярные языки программирования своими корнями прочно стоят в *императивной парадигме* программирования; *императивная парадигма* программирования является прямой моделью *машины фон Неймана*; мы согласимся со светилами информатики XX века и поддержим мысль о том, что *императивное программирование давно изжило себя*.

Неудивительно, что многие исследователи уже давно пришли к выводу о том, что нужно искать

дополнительные пути и придумывать новые парадигмы программирования. Однако сама идея функционального программирования в виде использования открытий таких математиков и логиков, как Хаскелл Карри, Алонсо Черч, Моисей Исаевич Шейнфинкель, появилась гораздо раньше разочарования программистов в императивных языках и пришла к нам из разработок в области искусственного интеллекта в виде языка Лисп. Лисп был и остается совершенно утилитарным языком, хорошо заточенным под нужды задач, близких к искусственному интеллекту. Идея создания чистого математически обоснованного языка, специально призванного для преодоления проблем фоннеймановских моделей, возникла в конце 60-х — начале 70-х годов XX века в работах таких исследователей, как вышеупомянутый Джон Бэкус, Робин Милнер, Дэвид Тернер, отчасти Валентин Федорович Турчин. Из этих идей родились языки, не привязанные к аппаратной части компьютеров, красиво описываемые математическими формализмами: ML, Рефал, FP, SASL. В начале своего пути эти языки были далеки от требуемого идеала, они являлись лишь доказательством того, что программирование может быть отделено от фоннеймановского стиля. В дальнейшем функциональные языки активно развивались, и венцом этого развития стал виновник нашего “торжества”: язык Haskell (названный в честь уже упомянутого математика Хаскелла Карри, одного из авторов комбинаторной логики, лежащей в основе многих языков функционального программирования).

Справедливости ради надо сказать, что создание функциональной парадигмы было не единственным ответом на моральное устаревание императивных фоннеймановских языков, примером могут служить логическая и объектно-ориентированная парадигмы. Однако эти парадигмы страдают от отсутствия мощной математической базы и могут быть выражены в терминах функционального программирования, поэтому мы не будем сильно заострять на них наше внимание. Интересующиеся могут обратиться к соответствующей литературе [2, 3].

Haskell: о самом важном

Надеюсь, предыдущий раздел оставил у вас положительные впечатления о функциональном программировании и заставил несколько обновить собственные взгляды на программирование императивное. Если нет, у вас еще будет предостаточно времени сделать это, читая остаток этой статьи.

Что же такое Haskell? Haskell — это чисто функциональный язык программирования. Заострите свое внимание на слове *чисто*. Чуть позже вы поймете, почему оно так важно в определении этого языка. В императивных языках работа программы состоит в выполнении компьютером последовательных заданий, описанных на вашем любимом

языке. Выполняя эти задания, машина представляет собой автомат — она имеет набор состояний и выполняет переходы между этими состояниями по заданным правилам. Например, вы можете объявить переменную `a` и присвоить ей некоторое значение, например, 3. Затем вы можете изменить ее. Также вы можете применить целый набор управляющих структур для более хитрого описания действий. Кроме того, вы можете определить несколько функций, которые будут, по сути, обобщением ваших приказаний и управляющих структур (здесь необходимо сделать важное уточнение: в современной литературе принято называть такие функции *блоками*, а термин *функция* оставлять для математически чистых функций, о которых вы прочитаете далее). Описанная схема напрямую соответствует модели машины фон Неймана.

В чисто функциональных языках, и в языке Haskell в первую очередь, вы увидите совершенно иное. В них вы не говорите машине, как она должна действовать, вы описываете ей, что вы хотите получить. Простой пример: факториал. Вы говорите: “Факториал — это произведение всех целых чисел от 1 до заданного. Сумма набора чисел — это сумма первого числа и суммы оставшихся чисел, и т.д.”. Важным фактом является то, что все данные в чисто функциональных языках *неизменяемы* (*immutable*). Вы не можете поменять значение переменной, которой один раз уже было что-то присвоено (если вас возмущает этот факт, подумайте о переменной не как о ячейке памяти, а как о переменной в математическом смысле, дальше вы увидите, что неизменяемость никак не связывает вам руки). И, наконец, одна из самых важных концепций функционального программирования: функции должны быть чистыми. Никаких изменений внешнего контекста программы: изменения переменных, безусловных переходов, — внутри функции происходить не должно. Функция описывается в строго математическом смысле, как отображение из одного множества в другое, и для одних и тех же входных данных она должна вернуть один и тот же результат (для любопытных читателей скажу, что для реализации ввода-вывода, генерации случайных чисел и прочих “нечистых” задач существуют специальные формализмы, подробнее о них можно прочитать в [4, 5, 6]). Чистота функций обеспечивает три важных преимущества: широкие возможности оптимизации кода, простоту верификации (доказательства правильности) программ, что имеет весьма важное значение при тестировании, и простоту иерархического проектирования (в первую очередь проектирования снизу вверх): разработки программ из маленьких составных частей.

Особенности, описанные в предыдущем абзаце, в принципе могут подходить для любого функционального языка программирования. Теперь пару слов о том, что выделяет Haskell среди остальных функциональных языков (кстати,

именно в Haskell нижеописанные возможности достигли своего совершенства). Haskell — это *ленивый* язык. Ленивость языка означает, что значения функций и переменных не будут вычислены до тех пор, пока они не понадобятся на самом деле, полезная работа откладывается на самый последний момент (тут некоторые читатели могут вспомнить свое студенчество). Какие преимущества это обеспечивает? Один из немаловажных плюсов ленивости, особенно заметный в современном мире многопоточных многопроцессорных систем, — это оптимизация выполнения участков кода по времени и оптимальное распределение работы между потоками и процессами. Важный теоретический аспект ленивости — возможность создания бесконечных структур данных, множеств континуальной мощности, так распространенных в математике (другой подход к описанию подобных структур — это так называемые *генераторы*, впервые появившиеся в языке Icon и успешно перекочевавшие в Python). Кроме того, ленивые языки могут представить выражение как самое настоящее *чистое* преобразование данных. Для понимания этого приведем небольшой пример. У нас есть список `xs = [1, 2, 3, 4]` и функция `doubleMe`, удваивающая все элементы в списке и возвращающая новый список. Также представим, что у нас есть выражение `doubleMe(doubleMe(doubleMe(xs)))`. Каким будет вычисление этого выражения в неленивом языке? Третья функция `doubleMe` удвоит значения в списке и отдаст новый список второй функции, вторая функция удвоит значения в полученном списке и вернет новый список первой функции, первая функция удвоит значения в новом полученном списке и вернет результат. Все это произойдет ровно в тот момент, в который компилятор или интерпретатор встретит это выражение. В ленивом языке это выражение начнет вычисляться только тогда, когда его результат действительно будет нужен, например, при выводе на экран. При этом первая функция попросит вторую вернуть результат, вторая попросит третью, третья возьмет первый элемент списка, удвоит его и возвратит второй, вторая удвоит его и возвратит первой, первая удвоит его и возвратит внешней функции, затем потребует следующий элемент и так далее. Заметьте, что в данной схеме список проходит один раз и не имеет дополнительных промежуточных представлений. Вот что называется чистым преобразованием данных: мы можем гарантировать, что наше выражение преобразует список в список и не усложнит ситуацию созданием его дополнительных представлений.

В заключение обзора специфических возможностей языка Haskell необходимо сказать о статической системе типов, унаследованной им из семейства языков ML (изобретенных Робинотом

Милнером, тем самым, что стоял у истоков преодоления проблемы фоннеймановских моделей). Когда вы компилируете вашу программу на Haskell, компилятор абсолютно точно знает, какой тип имеет каждый кусочек, каждая переменная в вашем коде, и не даст вам скомпилировать программу, если вы наделали ошибок, попытавшись совместить переменные разных типов. Это позволяет избавиться от многих ошибок уже во время компиляции. В отличие от императивных языков, использующих тот же подход, Haskell поддерживает очень мощную, математически строгую систему вывода типов: если компилятору известны типы аргументов функции и известен характер функции, он может автоматически и однозначно вывести тип результата функции (не в последнюю очередь это достигается чистотой функций).

Не пугайтесь, если вы что-то не поняли в том, что я сказал о языке Haskell в этом разделе, в дальнейшем вы увидите, что Haskell — это элегантный и мощный язык программирования, и программы на нем получаются значительно более краткими, чем соответствующие программы на императивных языках.

Чтобы понять Haskell, необходимо почувствовать его самостоятельно, поэтому ваши собственные практические занятия по написанию программ будут иметь куда большую пользу, чем мои пространные рассуждения о математической строгости и функциональности. В силу этой причины остаток статьи будет представлять собой нечто вроде наглядного и простого урока, призванного дать вам хороший старт в нелегком деле самостоятельного постижения функционального программирования.

В заключение этого раздела, прежде чем перейти к непосредственному описанию основ языка и правил написания программ на нем, необходимо сказать пару слов о том, что вам необходимо для работы. В первую очередь это компилятор или интерпретатор. Самым популярным компилятором Haskell является свободно распространяемый GHC (Glasgow Haskell Compiler), и если вы решите использовать его, а также планируете “застрячь” в Haskell надолго, то лучше сразу скачать Haskell Platform — полный набор программ для программирования на Haskell. Также GHC может работать в качестве интерпретатора. Если же вы планируете просто глянуть на Haskell одним глазом, то я посоветую вам скачать также свободно распространяемый и специально предназначенный для обучения интерпретатор Hugs.

Пара простых замечаний. Традиционно программы на Haskell хранятся в файлах с расширением *.hs, подобные файлы можно открыть или из контекстного меню, или из командной строки интерпретатора. Как вызвать интерпретатор? Если вы выбрали GHC, то интерпретатор вызывается командой `ghci`, в случае Hugs команда будет

простой: `hugs`. Как открыть файл с программой из командной строки интерпретатора? Допустим, у вас есть файл `myprogram.hs`; чтобы открыть его, вам нужно написать `:l myprogram` в строке интерпретатора. Если во время работы вы изменили файл, вы можете снова набрать `:l myprogram` или же воспользоваться повторной загрузкой файла с помощью команды `:r`. Полученных знаний вам должно хватить для перехода к следующему разделу.

Первые шаги в мире функций

Вы все же решили попробовать Haskell в деле, и это просто потрясающе! Запустите любившийся вам интерпретатор, и вы увидите приглашение, например, `Hugs>` или `Prelude>`; уже сейчас вы можете использовать Haskell в качестве продвинутого калькулятора. Все известные вам правила ассоциативности и коммутативности арифметических операций для него справедливы, необходимо сделать лишь одно небольшое замечание: отрицательные числа крайне рекомендуется записывать в скобках, например: `5 * (-4)`, поскольку в целях чистоты и правильности кода некоторые компиляторы будут ругаться на записи в духе `5 * -4`.

Поиграв с обычной арифметикой, можете обратить свой взгляд на булеву: обозначения операций в ней похожи на соответствующие обозначения в C — `&&`, `||`, но `not`. Операция сравнения на равенство аналогично своему побратиму в C тоже обозначается `==`, операция же сравнения на неравенство имеет несколько непривычный вид `/=`. Булевы значения в Haskell представляются как `True` и `False` и относятся к соответствующему типу `Bool`. Важно заметить, что в силу строгой типизации вы не можете сравнить значения разных типов. Например, если вы попытаетесь написать `3 == True` или `3 == '3'`, вы получите ошибку. Вы не можете сравнить несравнимое (по мнению компилятора). В очень редких случаях типы могут быть неявно приведены, например, в выражении `3 - 1.0` число 3 может быть трактовано как целое или же как вещественное, однако число 1.0 может быть трактовано исключительно как вещественное, поэтому в данном выражении компилятор примет решение в пользу вещественных чисел и приведет `3 к 3.0`, а не `1.0 к 1`. В дальнейшем я опишу типы более подробно, пока что они не являются предметом нашего рассмотрения.

Если вы до этого не имели опыта программирования на Haskell, то вы наверняка могли предположить, что вышеописанные операции вычисляются с помощью операторов. Однако это не так: `+`, `-`, и все другие арифметические и логические операции являются чистыми функциями, и поскольку они заключены между своих аргументов, они называются инфиксными. В языке Haskell не так уж

много инфиксных чисел, и большая их часть пришла из мира математики. Основная масса функций являются префиксными, то есть их имена записываются перед аргументами, например, `max 1 3`. Заметьте особое отличие вызова функций в Haskell от вызовов в других языках — аргументы при записи вызова пишутся без скобок и каких-либо разделительных знаков, поскольку все это имеет несколько другой смысл.

Разумеется, вы запросто можете определить собственные функции. Вы должны разместить описание функции в файле с расширением `*.hs` и загрузить этот файл командой `:l` — определение функций в командной строке интерпретатора работать не будет (в дальнейшем будет описан способ сделать это, если уж очень хочется)! Опишем, к примеру, вышеупомянутую функцию `doubleMe`, которая удваивает собственный аргумент. Это просто:

```
doubleMe x = x + x
```

или:

```
doubleMe x = x * 2
```

оба определения математически правильны и эквивалентны. Разумеется, вы можете определить другие функции, используя уже определенные вами функции (причем сделать это можно даже до определения последних!):

```
doubleMeTwice x = doubleMe (doubleMe x)
```

Также вы можете использовать условные выражения при описании функций. Например, нам надо определить функцию, которая будет удваивать лишь числа не большие ста:

```
doubleFromFirstHundred x = if x > 100
then x else doubleMe x
```

Здесь необходимо заметить, что ветвь `else` в языке Haskell обязательна по следующей причине: условный оператор является *выражением*, которое всегда должно иметь определенное значение. Он ни в коем случае не является чистой функцией, поскольку зависит от значения переменных, определенных вонне его, но он является полноправным математическим выражением.

Константные выражения в Haskell тоже являются функциями: функциями, не имеющими аргументов и, следовательно (вспомните правило: чистая функция всегда должна давать один и тот же результат для одних и тех же входных данных), имеющими единственно возможное значение:

```
temperature = 35
```

Напоследок для людей, заинтересовавшихся правилами именования функций: имя функции в Haskell не может начинаться с заглавной буквы, поэтому имена всех вышерассмотренных функций начинаются со строчных букв. Кроме того, принято еще одно интересное соглашение о наименовании функций: имя функции, заканчивающееся на апостроф, обозначает либо неленивую функцию, либо несколько измененный вариант уже имеющейся функции. Такое именование — стандартная практика при программировании на языке Haskell.

Списки, списки, списки

Список — одна из основных структур данных в языке Haskell. Если вы имеете хоть малейший опыт практического программирования, вы уже, вне сомнения, знаете, что такое список. В Haskell списком может быть однородное множество элементов. Вы можете иметь список чисел или список символов, но не список, содержащий одновременно и числа и символы. Например:

```
ghci> let someList = [1, 2, 3, 4]
ghci> let anotherList = ['h', 'e', 'l',
                        'l', 'o']
```

[Замечание для любознательных читателей: слово `let` и есть та самая конструкция, которая позволяет определять функции прямо в строке интерпретатора (а ведь это определение списка и есть определение функции, как я уже писал выше). К сожалению, в Hugs это так просто не работает.]

Строки являются частным случаем списков — это списки символов. Однако Haskell разрешает не писать каждый раз `['h', 'e', 'l', 'l', 'o']`, когда мы хотим определить строку `'hello'`, вы можете записать ее напрямую.

Одна из основных и простейших операций над списками — конкатенация, или склейка, списков:

```
Hugs> [1, 2, 3, 4] ++ [9, 10, 11, 12]
[1, 2, 3, 4, 9, 10, 11, 12]
Hugs> 'hello' ++ ' ' ++ 'world'
'hello world'
```

Однако с частой конкатенацией очень длинных списков следует быть осторожным, поскольку это довольно ресурсоемкая операция: при склейке двух списков интерпретатор вынужден пройти первый из них от начала до конца, а, как нам известно, время этого прохода линейно зависит от числа элементов в списке.

Теоретическая база списков в Haskell унаследована из языков семейства Лисп и заслуживает некоторого рассмотрения. Допустим, вы хотите добавить элемент в начало списка — это делается оператором *двоеточие*:

```
Hugs> 4:[1, 2, 3]
[4, 1, 2, 3]
```

В данном случае число 4 называется *головой* списка, а список `[1, 2, 3]` — *хвостом*. Рассмотрим сам список `[1, 2, 3]`, его голова — 1, хвост — `[2, 3]`. Рассмотрим этот хвост, его голова — 2, хвост — `[3]`. Выражение `[3]` ни в коем случае не является числом, это список, имеющий голову 3 и хвост `[]`, где `[]` является специальным видом списка — *пустым списком*. Таким образом, список `[4, 1, 2, 3]` имеет альтернативную (имеющую теоретически верный вид) форму `4:1:2:3:[]`, однако при записи программ предпочтительно использовать первую форму записи списков для повышения читабельности.

При записи списков чисел (или других объектов перечислимого типа; о классах типов — дальше) можно использовать промежутки:

```
Hugs> [1 .. 4]
[1, 2, 3, 4]
```

Если правый конец промежутка не указан, то мы получим бесконечный список, начинающийся с заданного элемента (не стоит пытаться его вывести, он все-таки бесконечный), ленивость Haskell легко позволяет работать с такими структурами данных.

Элементами списков также могут быть списки, причем не обязательно, чтобы все внутренние списки одного большого списка имели одинаковую длину: в Haskell тип *список* обозначает список любой длины.

Операций над списками в Haskell великое множество: это и традиционная для массивов индексация, и взятие головы и хвоста списка, и подсчет длины, и нахождение минимального и максимального элементов, и многое-многое другое. За полным перечнем можно обратиться к специальной литературе [4, 5, 6]. Для нас сейчас самыми интересными являются свертки списка.

Суммирование элементов списка:

```
Hugs> sum [1, 2, 3, 4]
10
```

Подсчет произведения элементов списка:

```
Hugs> product [1, 2, 3, 4]
24
```

С помощью таких сверточных функций красиво реализуются многие математические задачи, к примеру, факториал:

```
factorial n = product [1 .. n]
```

Посмотрите на это определение функции, считающей факториал, и скажите, что оно вам напоминает? Да это же один в один математическое определение: “Факториал числа n равен произведению всех чисел от 1 до n включительно!” Вспомните еще хоть один неспециализированный язык, который позволил бы вам определять функции в том виде, в котором они задаются в математике.

Одним из самых мощных средств работы со списками в Haskell являются так называемые *списочные выражения*. Вспомните, как определяются множества в математике, например: “Множество S — это множество x , принадлежащих множеству N и удовлетворяющих системе неравенств...”. В языке Haskell имеются все средства для задания множеств точно таким же языком! Например, множество удвоенных чисел из промежутка от 1 до 10, тогда вы можете записать:

```
Hugs> [x * 2 | x <- [1 .. 10]]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

То выражение, которое вы ввели, и является *списочным выражением*. Оно позволяет описывать множества практически любой сложности и имеет следующий общий вид:

[выражение_содержащее_переменные_ $x_1 \dots x_n$ | $x_1 <-$ множество1, ..., $x_n <-$ множество N , неравенство1, ..., неравенство M]. Если этот общий вид слишком пугает вас, приведу простой пример нетривиального списочного выражения:

```
Hugs> [x * y | x <- [2,5,10],
           y <- [8,10,11], x * y > 50, y < 11]
[80, 100]
```

Это списочное выражение описывает множество всех произведений x на y , для x , принадлежащего множеству [2, 5, 10], y , принадлежащего множеству [8, 10, 11], при условии, что это произведение больше 50, а y меньше 11.

Разумеется, число N , определяющее количество переменных в списочном выражении, может быть равно нулю. В этом случае вместо имен переменных будут использоваться знаки подчеркивания: “_”, которые всегда используются в языке Haskell, когда имя переменной нас не заботит. Это не такое глупое занятие, как может показаться, самым ярким примером будет наше собственное определение функции, считающей длину списка:

```
length' xs = sum [1 | _ <- xs]
```

Рассмотрим это определение ближе. Обратите внимание на букву *s* в конце имени переменной *xs* — это традиционное для Haskell соглашение, называть списки с буквой *s* в конце (от множественного числа в английском языке). Обратимся к списочному выражению [1 | _ <- xs]. Говоря математическим языком, это множество единиц, равнозначное множеству *xs*, то есть список единиц с длиной, равной длине списка *xs*. Соответственно, просуммировав эти единицы, мы получим искомую длину.

Еще одно близкое к спискам понятие — это *кортеж*. Кортежем в языках программирования называется набор, возможно разнотипных, элементов строго определенной длины. Если сравнивать системы стандартных типов императивных и функциональных языков, то, если список похож на массив, кортеж похож на структуру с неименованными полями (если вам не нравится отсутствие имен у полей, вспомните слова по поводу основной нагрузки узкого места фон Неймана).

Важной функцией, связанной с кортежами, является *zip*. Эта функция берет два списка и создает список пар [(первый_элемент_первого_списка, первый_элемент_второго_списка), ..., (последний_элемент_первого_списка), (последний_элемент_второго_списка)]. Если списки неравны по длине, то создание списка пар заканчивается по достижении конца одного из списков. Это значит, что *zip* может легко работать с бесконечными списками:

```
Hugs> zip [1..] ['apple', 'orange',
               'cherry', 'mango']
[(1, 'apple'), (2, 'orange'),
 (3, 'cherry'), (4, 'mango')]
```

Заканчивая разговор о списках, покажу элегантное и истинно функциональное решение несложной

математической задачи. Условие задачи ставится так: необходимо найти все такие прямоугольные треугольники со стороной не больше 10 и периметром, равным 24. В императивных языках эта задача имеет не самое тривиальное и не самое очевидное решение, связанное с неоднократным применением циклов. В Haskell же все ясно: необходимо использовать списочное выражение. Представим треугольник в виде кортежа из значений длин сторон — (a, b, c) . Перечислим все прямоугольные треугольники со сторонами не больше 10:

```
Hugs> [(a, b, c) | c <- [1..10],
a <- [1..10], b <- [1..10],
a^2 + b^2 == c^2]
[(4, 3, 5), (3, 4, 5), (8, 6, 10),
(6, 8, 10)]
```

Добавим ограничение на периметр:

```
Hugs> [(a, b, c) | c <- [1..10],
a <- [1..10], b <- [1..10],
a^2 + b^2 == c^2, a + b + c == 24]
[(8, 6, 10), (6, 8, 10)]
```

Мы получили решение задачи — два треугольника, но, по сути, они являются одним и тем же треугольником. Для того чтобы избавиться от неоднозначности, введем ограничения, что c всегда больше b , а b всегда больше или равно a :

```
Hugs> [(a, b, c) | c <- [1..10],
b <- [1..c], a <- [1..b],
a^2 + b^2 == c^2, a + b + c == 24]
[(8, 6, 10), (6, 8, 10)]
```

Заметьте также, что определения переменных b и a поменялись местами, поскольку в момент определения a , который был раньше, мы ничего не знали о b . Как видите, решение этой задачи на Haskell заняло всего одну строчку кода.

Функции под увеличительным стеклом

Пока что я дал вам лишь очень общие представления о функциях в Haskell, и они не раскрывают всей реальной мощи этих конструкций. Как я уже упоминал, все конструкции строго знают тип своих аргументов. Разумеется, функции не являются исключением. Несмотря на то что компилятор может автоматически вывести необходимые типы, хорошим тоном является явное описание типа функции. Делается это (не в командной строке интерпретатора!) с помощью следующего выражения:

```
имя_функции :: тип_аргумента1 -> ... ->
              тип_аргументаN ->
              тип_результата
```

Например:

```
factorial :: Int -> Int
```

Множества типов могут описываться некоторым неизвестным до этого именем:

```
myFunction :: a -> a
```

Это определение описывает функцию, принимающую и возвращающую значения некоего

одного и того же типа a . Множества типов часто комбинируются с классификаторами типов. В этой вводной статье я не буду касаться этих вопросов, о них можно подробно узнать в [4, 5, 6].

Предварительное описание функций позволяет здорово разгрузить компилятор и улучшить степень оптимизации программ.

Перейдем непосредственно к описанию “вкусностей”, связанных с функциями. Практически ни одна более-менее сложная функция не определяется без использования *сравнения с образцом*, что позволяет избежать проблем, связанных с использованием условных операторов. Рассмотрим следующее определение факториала:

```
factorial :: Int -> Int
factorial 0 = 1
factorial n = n * factorial (n - 1)
```

Это рекурсивное определение, вероятно, знакомо многим людям, близким к программированию. Обычно для его описания необходимо использовать условные конструкции. Однако, посмотрите, как оно может быть реализовано в языке Haskell. Сначала мы говорим, что будем описывать функцию `factorial`, которая принимает целое число и возвращает целое число, затем утверждаем, что факториал от нуля равен единице, а в конце даем полное рекурсивное определение факториала для оставшегося множества натуральных чисел. Техника определения функций через сравнение с образцом очень удобна, она позволяет выделить в области определения функции специальные точки и отдельно описать функцию для этих точек, позволяя основному определению быть как можно более ясным и понятным. Сравнение с образцом особенно часто применяется при работе со списками или кортежами. Опишем, к примеру, функцию, которая находит голову списка:

```
head' :: [a] -> a
head' [] = error 'Empty lists have
              no head!'

head' (x:_) = x
```

В этом определении мы видим, что головой списка называется тот элемент, который прикрепляется к нему спереди. Поскольку остальные элементы не имеют для нас значения, то они не имеют и имени, а обозначаются символом подчеркивания. В заключение рассмотрения сравнения с образцом опишем еще один вариант функции, вычисляющей длину списка:

```
length' :: [a] -> Int
length' [] = 0
length' (_:xs) = 1 + length' xs
```

Специальная точка в области определения функции длины — пустой список — выделена в особый случай, длина такого списка равна нулю. Длина же любого непустого списка равна единице, плюс длина его хвоста. Голова в данном случае нам не нужна, поэтому обозначена подчеркиванием.

Выражение `x : xs`, два частных случая которого — `x : _` и `_ : xs` часто применяется при определении через сравнение с образцом функций, работающих со списками. Здесь `x` является головой списка и имеет тот же тип, что и все элементы списка, например, `a`, а `xs` является хвостом списка и имеет тип “список элементов типа `a`”. В отдельных случаях применяются и более сложные выражения, например, `x : y : xs` или даже `x : y : z : xs`.

Следующим важным моментом в определении функций являются *охраняющие выражения*. Подобная форма записи должна быть знакома многим, ведь каждый, кто изучал в школе математику, хоть раз да определял кусочно заданные функции с ее помощью. Разумеется, это та самая крайне удобная возможность разбить область определения сложной функции на несколько подобластей и определить функцию отдельно для каждой из этих подобластей. Простейший пример — функция, ищущая максимальное из двух целых чисел:

```
max' :: Int -> Int
max' a b
  | a > b = a
  | b >= a = b
```

Как видите, для разбиения области определения функции используется вертикальная черта, после которой следует неравенство, показывающее отношение между входными данными, подобных разбиений может быть любое количество. В вышеописанном определении есть недочет: оно избыточно, содержит лишнее неравенство. Ведь если `a` не больше `b`, то это автоматически обозначает, что `b` больше или равно `a`. Для описания случаев, которые не описываются ни одним из встреченных ранее неравенств, применяется ключевое слово `otherwise`, с ним это определение станет куда элегантнее:

```
max' :: Int -> Int
max' a b
  | a > b = a
  | otherwise = b
```

В мои задачи не входит описать все возможные варианты определения функций, поэтому, рассмотрев лишь самые интересные из них, отсутствующие в привычных нам императивных языках программирования, я закончу речь о подобных формальностях и перейду к краткому описанию действительно мощного метода программирования: использованию функций высших порядков, что является исключительной прерогативой и основой мощи функционального стиля.

Функции от функций от функций

Одна из фундаментальных концепций функционального программирования пришла из теории лямбда-исчисления, своеобразного формализма, подобного машине Тьюринга, созданного для того, чтобы конструктивно описывать математические выражения. Эта концепция носит имя лямбда-

выражений и, иными словами, представляет собой анонимные функции, то есть функции, не имеющие имени.

Для того чтобы понять, зачем они нужны, необходимо сказать, что функции в Haskell являются собственным *полноправным* типом данных: они могут быть аргументами и результатами других функций. В Haskell такая терминология не распространена, но обычно функции, принимающие или возвращающие другие функции, называют *функционалами*, а в среде любителей Haskell — *функциями высших порядков*. В теории функционального программирования выделяют два самых важных функционала: `map` и `filter`. Функционал `map` принимает функцию `f` и список, и применяет эту функцию к каждому элементу списка, формируя новый список:

```
Hugs> map sin [1, 2, 3, 4]
[0.841470984807897,0.909297426825682,
0.141120008059867,-0.756802495307928]
```

Функционал `filter` принимает функцию, которая возвращает логическое значение (предикат) и список, а возвращает список тех элементов исходного списка, для которых применение предиката дает результат `True`.

```
Hugs> filter (> 2) [1, 2, 3, 4]
[3,4]
```

В этом примере мы отсеиваем из списка все значения, которые меньше или равны двойке. Заметьте интересную запись функции — `(> 2)`, этот прием называется *каррингом*, в честь математика Хаскелла Карри. В языке Haskell вы можете любой функции дать неполный набор аргументов и получить при этом новую функцию, которая примет оставшиеся аргументы. Яркий пример использования такой особенности показан выше: применение таких неполных функций в функционалах `map` и `filter`.

Точно такого же эффекта, как применение карринга, но с большей гибкостью, можно достичь с помощью анонимных функций, то есть лямбда-выражений. Перепишем пример для `filter` с их использованием:

```
Hugs> filter (\x -> x > 2) [1, 2, 3, 4]
[3,4]
```

Как видите, анонимная функция определяется следующим образом: `\ аргумент1 ... аргументN -> тело` (символ “\” обозначает греческую букву “лямбда”). Лямбда-выражение может иметь любое число аргументов и в принципе является полноправной функцией, а не урезанной версией существующей, что дает нам большую мощь при их использовании. Один из классических примеров использования анонимных функций: обобщенная версия функции сортировки, принимающая список, подлежащий сортировке, и функцию, которая сравнивает по некоторым правилам два аргумента.

На этом многообразии применения функций высших порядков не заканчивается, но в мою за-

дачу не входит полностью описать эту практически необъятную область. Как и всегда, к моему сожалению, вынужден отправить любопытных читателей в [4, 5, 6].

Как быстро это все работает?

Рассуждения о производительности программных систем всегда очень субъективны, поэтому я даже не стану пытаться выгораживать себя и убеждать вас в том, что мои потуги в этом направлении сильно отличаются в сторону объективности от чьих-либо чужих, коих полно в Интернете, специальной литературе и публицистике. Думаю, у вас уже назрел вполне ожидаемый вопрос: а сколько ресурсов мы теряем для поддержания работы такой мощной среды программирования, в которой работают программы, написанные на языке Haskell? Ведь, разумеется, скажете вы, такие легкие и непринужденные действия и операции над списками, функциями, любыми другими видами данных не могут пройти безболезненно. Вы будете абсолютно правы: программы на Haskell работают медленнее *аналогичных по функциональности* программ на языке C или Pascal и тратят большее число памяти при своей работе. Однако результаты тестов показывают, что обычно производительность программ на Haskell редко падает больше чем в 1,5 раза, и лишь в самых исключительных случаях функциональные программы медленнее соответствующих императивных в 9–10 раз. В настоящее время ведутся активные работы по разработке компилятора Haskell, который сможет сравнять производительность функциональных и императивных программ, и эти работы весьма успешны: в них используются концепции метавычислений, позволяющих создавать оптимизирующие компиляторы, “заточенные” под каждую отдельную программу (!). Из серьезных минусов производительности функциональных сред разработки можно назвать, пожалуй, чрезмерно высокую требовательность компиляторов к ресурсам, ввиду высокой сложности применяемых оптимизаций. К сожалению, эта проблема действительно имеет место, и решить ее, не ухудшив скорости генерируемого кода, не представляется возможным. В целом, суммируя сказанное в этом разделе, можно сказать, что Haskell представляет серьезную угрозу для императивного программирования, являясь одним из немногих действительно *быстрых* функциональных языков, поэтому он как минимум заслуживает того, чтобы к нему присмотрелись поближе.

Заключение: а серебряная ли пуля?

Завершая этот краткий обзор, я надеялся ответить на вопрос: а является ли Haskell той самой

серебряной пулей, которая поможет нам справиться с оборотней сложности проектирования и реализации программных систем? Когда же я напрямую подошел к написанию этого раздела, я понял, что так и не могу дать вам этот ответ и одновременно не запутать вас и не сбить с пути, поэтому, думаю, вам, читателям, предстоит для себя решить, нужен ли и интересен ли вам Haskell и функциональное программирование вообще, кажутся ли они вам мощнее императивных языков, могут ли они стать вашим повседневным инструментом. В этой статье я не успел охватить глубины языка Haskell, но, надеюсь, я смог показать его невероятную простоту вкупе с сильнейшей теоретической и математической обоснованностью всех языковых конструкций. В общем, делайте выводы, выбирайте язык программирования по душе и ни в коем случае не ставьте себя в какие-нибудь жесткие рамки, делая этот выбор.

Примеры

В данном разделе я размещу 20 простых программ на языке Haskell, решающих разнообразные часто встречающиеся алгоритмические и математические задачи. Неясные моменты будут сопровождаться аннотациями. Для запуска в интерпретаторе все примеры должны быть сохранены как определения функций — в отдельном файле и потом загружены с помощью команды `:l`. Запуск осуществляется вызовом соответствующей функции или функции `main`. Результатом некоторых программ могут являться списки (в том числе бесконечные).

1. 99 бутылок колы

Один из самых бесполезных, но в то же время довольно нетривиальных примеров. Задача состоит в том, чтобы вывести стихотворение:

“99 bottles of cola on the wall, 99 bottles of cola.

Take one down and pass it around, 98 bottles of cola on the wall.

98 bottles of cola on the wall, 98 bottles of cola.

Take one down and pass it around, 97 bottles of cola on the wall.

...”

и так далее.

```
bob n = (num n) ++ ' bottle' ++
        (s n) ++ ' of cola'
```

```
wall = ' on the wall'
```

```
pass 0 = 'Go to the store and buy
         some more, '
```

```
pass _ = 'Take one down and pass
         it around, '
```

```
up (x : xs) = x : xs
```



```
num (-1) = '99'
num 0    = 'no more'
num n    = show n
```

```
s 1 = ''
s _ = 's'
```

```
main = putStr $ concat
      [up (bob n) ++ wall ++ ', ' ++
       bob n ++ '\n' ++
       pass n ++ bob (n - 1) ++
       wall ++ '\n\n'
      | n <- [99, 98 .. 0]]
```

2. Программа, определяющая, является ли заданный год високосным

```
isLeap year | 0 == year 'mod'
            100 = 0 == year 'mod' 400
            | otherwise = 0 == year „'mod' 4
```

3. Вычисление факториала по определению

```
factorial n = product [1..n]
```

4. Вычисление бесконечной последовательности чисел Фибоначчи.

```
fib = 0 : 1 : zipWith (+) fib (tail fib)
```

Функция `zipWith` аналогична функции `zip`, только она не соединяет значения `a` в кортежи, а применяет к ним заданную функцию.

5. Реализация функции Аккермана — простейшей функции, не являющейся примитивно рекурсивной

```
ack 0 n = n + 1
ack m 0 = ack (m-1) 1
ack m n = ack (m-1) (ack m (n-1))
```

6. Поиск наибольшего общего делителя по алгоритму Евклида (GCD)

```
gcd' a 0 = a
gcd' a b = gcd' b (a 'mod' b)
gcd 0 0 = error 'GCD of 0 0
          is undefined'
gcd x y = gcd' (abs x) (abs y)
```

7. Поиск наибольшей общей подпоследовательности (LCS)

Подпоследовательность можно получить, удалив из данной последовательности некоторые символы. Задача состоит в поиске наибольших одинаковых подпоследовательностей для двух последовательностей. Используется в утилитах `diff`.

```
longest xs ys = if length xs > length
                ys then xs else ys
```

```
lcs [] _ = []
lcs _ [] = []
lcs (x:xs) (y:ys)
  | x == y    = x : lcs xs ys
  | otherwise = longest (lcs (x:xs) ys)
                      (lcs xs (y:ys))
```

8. Построение бесконечной последовательности, каждый следующий член которой является следующим рядом треугольника Паскаля

Треугольник Паскаля — треугольник, образованный биномиальными коэффициентами. О биномиальных коэффициентах — см. дальше.

```
nextRow row = zipWith (+)
              ([0] ++ row) (row ++ [0])
pascal = iterate nextRow [1]
```

Функция `iterate` строит бесконечный список последовательных применений заданной функции к заданному списку — $[x, f x, f(f x), \dots]$

9. Проверка, является ли число совершенным

Совершенные числа — натуральные числа, равные сумме своих делителей.

```
perfect n = n == sum [i | i <- [1..n-1],
                          n 'mod' i == 0]
```

10. Построение бесконечного списка простых чисел с помощью решета Эратосфена

```
sieve (p:xs) = p : sieve [x | x <- xs, x
                              'mod' p > 0]
primes = sieve [2..]
```

11. Построение бесконечного списка чисел Хэмминга

Числа Хэмминга — натуральные числа, имеющие простые делители, меньшие пяти.

```
merge (x:xs) (y:ys)
  | x < y = x : xs 'merge' (y:ys)
  | x > y = y : (x:xs) 'merge' ys
  | otherwise = x : xs 'merge' ys
hamming = 1 : map (2*) hamming 'merge'
              map (3*) hamming 'merge'
              map (5*) hamming
```

12. Факторизация натурального числа

Факторизация — разложение натурального числа на простые делители.

```
factorize' n pps@(p:ps) = case n
                          'divMod' p of
  (0, 1      )          -> []
  (remainder, 0) ->
    p : factorize' remainder pps
  -> factorize' n ps
```

```
factorize n = factorize' n primes
```

Бесконечный список `primes` взят из задачи 10.

Выражение `имя1@(x:xs)` разбивает данный список на голову и хвост, оставляя за исходным списком имя `имя1`.

13. Решение задачи о Ханойских башнях

Постановка задачи: даны три стержня, на один из которых нанизаны n , причем кольца отличаются размером и лежат меньшее на большем. Задача состоит в том, чтобы перенести пирамиду из n колец за наименьшее число ходов. За один раз разрешается переносить только одно кольцо, причем нельзя класть большее кольцо на меньшее.

```
hanoi' 0 _ _ _ = []
hanoi' n a b c = hanoi' (n-1) a c b ++
                 [(a, b)] ++ hanoi' (n-1) c b a
hanoi n = hanoi' n 1 2 3
```

Функция `hanoi` возвращает решение в виде списка пар (a, b) , каждая из которых значит: “Передвиньте диск со стержня a на стержень b ”.

14. Расчет биномиальных коэффициентов

Биномиальные коэффициенты — коэффициенты в разложении бинома Ньютона по степеням x . Общая формула для расчета: $\text{binom}(n, k) = n! / ((n - k)!k!)$.

```
binom n k = product([k + 1..n]) 'div'
           product([1..n - k])
```

15. Скалярное произведение векторов

```
dot a b | length a == length
        b = sum (zipWith (*) a b)
| otherwise = error 'Sizes don't match'
```

16. Численное решение уравнения

Нахождение численного решения уравнения $f(x) = 0$, на промежутке $[start, stop]$ с шагом $step$ и допустимой ошибкой eps .

```
solve f start stop step eps =
    [x | x <- [start, start+step ..
stop], abs (f x) < eps]
```

17. Гномья сортировка

Описание гномьей сортировки Диком Груном: “Гномья сортировка основана на технике, используемой обычным голландским садовым гномом. Это метод, которым садовый гном сортирует линию цветочных горшков. По существу, он смотрит на следующий и предыдущий садовые горшки: если они в правильном порядке, он шагает на один горшок вперед, иначе он меняет их местами и шагает на один горшок назад. Граничные условия: если нет предыдущего горшка, он шагает вперед; если нет следующего горшка, он закончил”.

```
gnomeSort' xx@(x:xs) (y:ys)
  | x <= y = gnomeSort' (y:xx) ys
  | otherwise = gnomeSort' xs (y:x:ys)
gnomeSort' [] (y:ys) = gnomeSort' [y] ys
gnomeSort' xs [] = reverse xs
gnomeSort [] = []
gnomeSort (x:xs) = gnomeSort' [x] xs
```

18. Сортировка слиянием

В сортировке слиянием список разбивается на две примерно равные части, и каждая часть сортируется отдельно, после чего отсортированные части сливаются вместе.

```
merge [] ys = ys
merge xs [] = xs
```

```
merge xs@(x:xs') ys@(y:ys') |
  x <= y = x : merge xs' ys
  | otherwise = y : merge xs ys'
split (x:y:zs) = let
  (xs,ys) = split zs in (x:xs,y:ys)
split [x] = ([x],[])
split [] = ([],[])
mergeSort [] = []
mergeSort [x] = [x]
mergeSort xs = let (as,bs) = split xs
  in merge (mergeSort as)
(mergeSort bs)
```

19. Быстрая сортировка

Основные идеи быстрой сортировки: выбрать опорный элемент, разбить список на два: меньшие опорного элемента и большие, рекурсивно отсортировать оба списка и склеить их.

```
qsort [] = []
qsort (x:xs) = qsort [y |
y <- xs, y < x] ++ [x] ++
qsort [y | y <- xs, y >= x]
```

20. Проверка, является ли строка палиндромом

Одна из простейших задач на десерт — проверка, является ли строка эквивалентной себе, прочитанной в обратном порядке.

```
isPalindrome s = s == reverse s
```

Список литературы

1. Бэкус Дж. Можно ли освободить программирование от стиля фон Неймана? Лекция при получении премии Тьюринга, 1977.
2. Себеста Р.У. Основные концепции языков программирования. М.: Вильямс, 2001, 668 с.
3. Biancuzzi F. Masterminds of Programming. O'Reilly Media, 2009, 496 с.
4. Душкин Р.В. Функциональное программирование на языке Haskell. М.: ДМК Пресс, 2008, 544 с.
5. O'Sullivan B., Goerzen J., Stewart D. Real World Haskell O'Reilly, 2008, 710 с.
6. Lipovača M. Learn You a Haskell for Great Good! No Starch Press, 2011, 400 с.

System Builder — авторский взгляд на виртуальный мир

В следующем номере “Информатики” не пропустите очень интересную статью про авторскую среду для изучения основ объектно-ориентированного программирования System Builder. Сама среда, разумеется, тоже будет размещена на диске, и вы сразу сможете попробовать ее в деле.

Мы не случайно вернемся к теме изучения основ ОПП, которой лишь недавно, в прошлом номере, был посвящен большой материал. Опубликованный материал был в определенном смысле классическим — таким, каким и положено быть материалу учебника. System Builder же предлагает именно авторский взгляд — визуальную среду, практически не требующую никакого программирования, зато наглядно иллюстрирующую основные понятия ОПП.

Не пропустите!



Учительская



книга

с 31
октября по 3
ноября

ВЫСТАВКА-ПРОДАЖА КНИГ ВЕДУЩИХ ФИРМ И ИЗДАТЕЛЬСТВ



Семинары и встречи с методистами и авторами учебников
БОЛЕЕ 1000 НАИМЕНОВАНИЙ КНИГ ДЛЯ УЧИТЕЛЯ

31 октября



Гуманитарные предметы

- Русский язык • Литература • История • МХК
- Музыка • Изобразительное искусство
- а также
- Управление школой • Технология
- Физкультура • ОБЖ • Здоровье детей
- Библиотека в школе

1 ноября



Предметы естественно-научного цикла

- География • Биология • Химия • Физика
- Математика • Информатика

2 ноября



Начальная школа Дошкольное образование

а также

Психология, воспитание

3 ноября



Иностранные языки

Подробное расписание будет опубликовано позднее в газете и на сайте
www.1september.ru

Чтобы получить профессиональный подарок, **пройдите заранее бесплатную регистрацию** на сайте <http://bookfair.1september.ru>

Все мероприятия фестиваля пройдут с **10.00** до **16.45** в московском государственном лицее № 1535 по адресу: ул. Усачева, дом 52 (в 3 минутах ходьбы от станции метро «Спортивная»).



ВХОД СВОБОДНЫЙ



ЛОГИКА

От кошек — через переключатели — к компьютерам

► Представьте себе, заходите вы однажды в зоомагазин, где можно купить кошку, и говорите продавцу: “Мне нужна кошка мужского пола, сиамская, белая или рыжая; или кошка женского пола, сиамская, любого цвета, кроме белого; или любая кошка черного окраса”. А продавец вам отвечает: “А-а, так вам нужна кошка из множества, описываемого выражением:

$$(M \times C \times (B + P)) + (Ж \times C \times (1 - B)) + Ч$$

Верно?”¹

Вы, конечно, спросите продавца: “А что означают буквы **М**, **Ж**, **Б**, **Р**, **С** и **Ч**, знаки «+», «×» и «1 →?»”. Он объяснит:

— **М** и **Ж** — множества всех кошек, соответственно, мужского и женского пола;

— **Б**, **Р** и **Ч** — множества всех кошек, соответственно, белого, рыжего и черного цветов;

— **С** — множество кошек сиамской породы;

— знак “+” обозначает слово **ИЛИ**, знак “×” — слово **И**;

— запись “1 →” соответствует слову **НЕ** (в смысле *все, за исключением чего-то*).

С учетом этого объяснения записанное продавцом выражение приобретает вид:

$$(M \text{ И } C \text{ И } (B \text{ ИЛИ } P)) \text{ ИЛИ } (Ж \text{ И } C \text{ И } (НЕ B)) \text{ ИЛИ } Ч$$

Прочитав новое выражение, вы, конечно, скажете: “Да! Именно это я произнес словами!”. Действительно, вам нужна кошка, входящая в одно из трех множеств:

$$(M \text{ И } C \text{ И } (B \text{ ИЛИ } P))$$

ИЛИ

$$(Ж \text{ И } C \text{ И } (НЕ B))$$

ИЛИ

Ч

Если принесенная кошка принадлежит к соответствующему множеству, то вместо буквы надо подставить 1 (или **Истина**, или **Да**), в противном случае — 0 (или **Ложь**, или **Нет**).

¹ Наверное, этот продавец, учась в школе, имел по информатике оценку “5” ©.

Для начала продавец приносит вам рыжего кота, о котором говорит, что тот не сиамской породы. Условие покупки этой кошки выглядит так:

$$(1 \text{ И } 0 \text{ И } (0 \text{ ИЛИ } 1)) \text{ ИЛИ } (0 \text{ И } 0 \text{ И } (НЕ 0)) \text{ ИЛИ } 0$$

Обратите внимание, что единице равны только условия **М** и **Р**, поскольку продавец принес рыжую кошку мужского пола.

Теперь нужно вычислить значение этого логического выражения². Но для этого нужно уметь вычислять значения более простых логических выражений, включающих только отдельные слова (или, как их еще называют, логические связки) **И**, **ИЛИ** и **НЕ**. Как вы, очевидно, знаете, это делается по так называемым “таблицам истинности”. Например, для слова **И** такая таблица выглядит так:

Условие 1	Условие 2	Условие 1 И Условие 2
1 (Истина)	0 (Ложь)	0 (Ложь)
0 (Ложь)	1 (Истина)	0 (Ложь)
1 (Истина)	1 (Истина)	1 (Истина)
0 (Ложь)	0 (Ложь)	0 (Ложь)

— то есть результат равен 1 (**Истина**), только если оба условия, которые “соединяет” слово **И**, истинны (равны 1).

Для логической связки **ИЛИ** таблица имеет вид:

Условие 1	Условие 2	Условие 1 ИЛИ Условие 2
1 (Истина)	0 (Ложь)	1 (Истина)
0 (Ложь)	1 (Истина)	1 (Истина)
1 (Истина)	1 (Истина)	1 (Истина)
0 (Ложь)	0 (Ложь)	0 (Ложь)

— то есть результат равен 1 (**Истина**), если хотя бы одно из условий, которые “соединяет” слово **И**, истинно (равно 1).

Таблица истинности для связки **НЕ**:

Условие	НЕ Условие
1 (Истина)	0 (Ложь)
0 (Ложь)	1 (Истина)

² Напомним, что логическими называются выражения, в которых используются условия, каждое из которых может быть истинным или ложным, и слова **И**, **ИЛИ**, **НЕ**; результат логического выражения равен **ИСТИНА** (или 1) либо **ЛОЖЬ** (0). Логическая операция, в которой используется слово **И**, называется конъюнкцией, или логическим умножением, слово **ИЛИ** — дизъюнкцией, или логическим сложением, слово **НЕ** — отрицанием, или инверсией.

— то есть при использовании слова *НЕ* результат противоположен значению условия, записанного после этого слова.

С помощью этих таблиц можно легко вычислить значение логического выражения:

$$(1 \text{ И } 0 \text{ И } (0 \text{ ИЛИ } 1)) \text{ ИЛИ } (0 \text{ И } 0 \text{ И } (\text{НЕ } 0)) \text{ ИЛИ } 0 =$$

$$= (1 \text{ И } 0 \text{ И } 1) \text{ ИЛИ } (0 \text{ И } 0 \text{ И } 1) \text{ ИЛИ } 0 = 0$$

Нулевой результат означает, что эта кошка вам не подойдет.

Тогда продавец приносит вам белую сиамскую кошечку. Подставляем в исходное выражение 0 и 1:

$$(0 \text{ И } 1 \text{ И } (1 \text{ ИЛИ } 0)) \text{ ИЛИ } (1 \text{ И } 1 \text{ И } (\text{НЕ } 1)) \text{ ИЛИ } 0.$$

Это выражение равно:

$$(0 \text{ И } 1 \text{ И } 1) \text{ ИЛИ } (1 \text{ И } 1 \text{ И } 0) \text{ ИЛИ } 0 = 0 \text{ ИЛИ } 0 \text{ ИЛИ } 0 = 0,$$

то есть и эту бедную киску вы тоже отвергаете.

Наконец, продавец приносит дымчатую сиамскую самку (ее цвет попадает в категорию Д — другой, т.е. не белый, не черный, не рыжий). Выражение

$$(0 \text{ И } 1 \text{ И } (0 \text{ ИЛИ } 0)) \text{ ИЛИ } (1 \text{ И } 1 \text{ И } (\text{НЕ } 0)) \text{ ИЛИ } 0$$

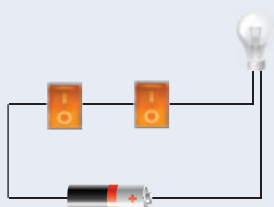
сводится к виду:

$$(0 \text{ И } 1 \text{ И } 0) \text{ ИЛИ } (1 \text{ И } 1 \text{ И } 1) \text{ ИЛИ } 0 = 0 \text{ ИЛИ } 1 \text{ ИЛИ } 0 = 1$$

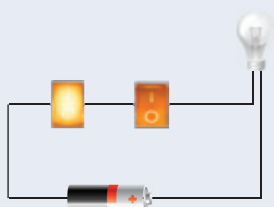
Окончательный результат равен 1 — кошка нашла свой новый дом!

Вечером того же дня кошка дремлет, свернувшись в клубок, у вас на коленях, а вы размышляете о том, можно ли, вместо того чтобы преобразовывать выражения с логическими связками *И*, *ИЛИ* и *НЕ*, соорудить такое устройство, которое определяло бы, подходит покупателю предложенная продавцом кошка или нет. А почему бы и нет! Давайте попробуем.

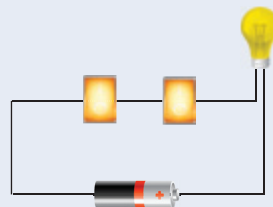
Чтобы начать наш эксперимент, надо соединить лампочку с батарейкой, как обычно, но включить в цепь два переключателя, а не один:



Как вы, очевидно, знаете из уроков физики, переключатели, соединенные таким способом (один за другим), называются подключенными последовательно. Если вы замкнете левый переключатель, то не случится ничего:



Ничего не случится и если вы замкнете правый переключатель при разомкнутом левом. Лампочка загорится лишь при одном условии: если включить левый и правый переключатели одновременно:



Работу цепи можно суммировать в следующей таблице:

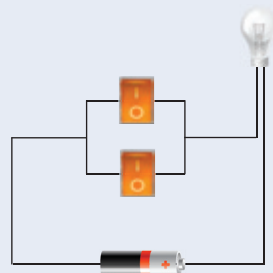
Левый переключатель	Левый переключатель	Лампочка
Замкнут	Разомкнут	Не горит
Разомкнут	Замкнут	Не горит
Замкнут	Замкнут	Горит
Разомкнут	Разомкнут	Не горит

Если разомкнутое состояние переключателя обозначить нулем, замкнутое — единицей, а состояния лампочки — аналогично: 0 (не горит) и 1 (горит), то эта таблица будет иметь вид:

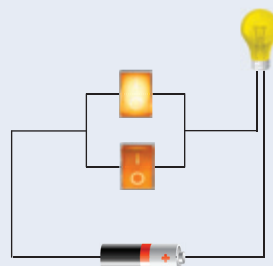
Левый переключатель	Левый переключатель	Лампочка
1	0	0
0	1	0
1	1	1
0	0	0

Вам эта таблица ничего не напоминает? Правильно — таблицу истинности для логической связки *И* (для операции логического умножения)! А это значит, что с помощью собранной нами схемы мы можем “механизировать” часть расчетов по решению вопроса о том, подходит покупателю предложенная продавцом кошка или нет! С ее помощью мы сможем определять результат логических выражений со словом *И*.

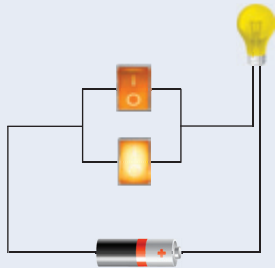
Теперь подключим переключатели иначе:



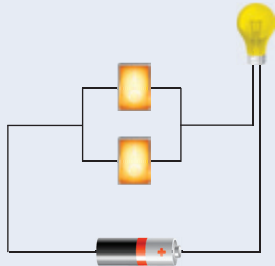
Отличие от предыдущей схемы в том, что лампочка будет гореть независимо от того, включили ли вы верхний переключатель:



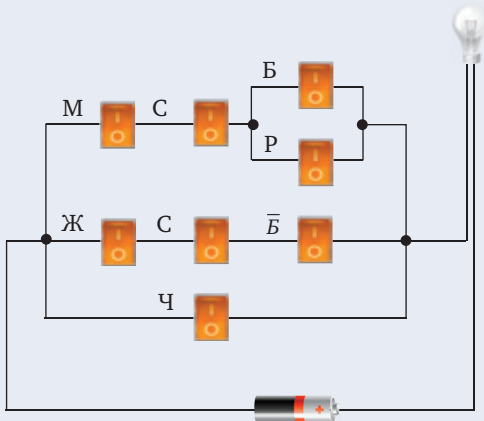
нижний:



или оба:



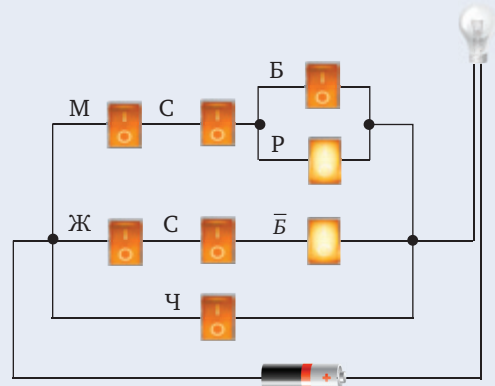
Вы, вероятно, уже догадались, что это таблица истинности для логической связки *ИЛИ* (для операции логического сложения). А это значит, что схема с параллельным соединением выключателей, как и с соединением последовательным, сможет “механизировать” часть расчетов по подбору кошки. Сделав этот вывод, мы можем соединить 8 переключателей в такую схему:



Каждый переключатель в этой схеме помечен буквой — той же, что и в логическом выражении, приведенном в начале статьи ($\bar{Б}$ эквивалентно выражению НЕ Б). Просмотрев схему слева направо и сверху вниз,

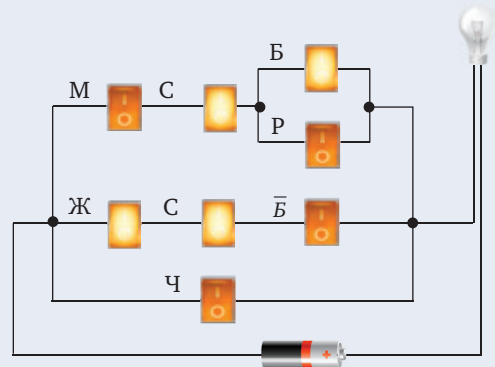
вы увидите, что буквы в ней стоят в том же порядке, что и в выражении. Каждому слову *И* в выражении соответствует фрагмент схемы, в котором два переключателя (или две группы переключателей) соединены последовательно (о таком соответствии мы говорили чуть выше). Каждому слову *ИЛИ* в выражении соответствует часть схемы, в котором два переключателя (или две группы переключателей) соединены параллельно (это мы тоже обсуждали).

Как вы помните, в нашем рассказе сначала продавец принес рыжего кота не сиамской породы. Включите соответствующие переключатели:

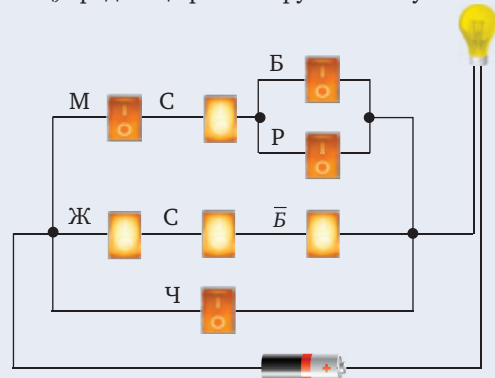


Хотя переключатели *М*, *Р* и $\bar{Б}$ замкнуты, цепь в целом разомкнута, и лампочка не горит. Это значит, что предложенный кот не подходит. Такой же ответ мы получили раньше, анализируя логическое выражение “аналитически”, т.е. наше устройство сработало правильно!

Затем продавец принес белую сиамскую кошечку:



И снова цепь осталась разомкнутой, и лампочка не зажглась, т.е. и в этот раз устройство нас не подвело. Наконец, продавец принес серую сиамскую кошечку:



Вот теперь цепь замкнута, лампочка горит, анализируя о том, что все ваши условия для покупки кошки выполнены!

Итак, мы с вами создали (пусть на бумаге) устройство, в котором с помощью переключателей можно вычислять результат сложного логического выражения.

А не хотите попробовать изготовить подобное устройство? Предлагаем читателям сделать это, обратившись при необходимости за помощью к учителю физики или технологии.

Возникает вопрос: “А нельзя ли аналогичным образом изготовить вычислительное устройство?” — Можно! Только вместо переключателей следует использовать что-то другое. О том, как это сделать, мы расскажем в одном из следующих выпусков.

ЗАДАЧНИК

Ответы, решения, разъяснения к заданиям, опубликованным в разделе “В мир информатики” ранее

1. Числовые ребусы с 2011 годом

Ответы

Задание “Россия в 2011 году”

Вариант 1: $PO^2 + CC^2 + ИЯ = 2011 - 29^2 + 33^2 + 81 = 2011$

Вариант 2: $PO^2 + CC + ИЯ^2 = 2011 - 37^2 + 66 + 24^2 = 2011$

Вариант 3: $PO^2 + CC + ИЯ = 2011 - 43^2 + 77 + 85 = 2011$

Задание “Москва в 2011 году”:

$$M^2 + O^4 + C^4 + K^4 + B^4 + A^2 = 2011 - 3^2 + 1^4 + 2^4 + 5^4 + 6^4 + 8^2 = 2011$$

(возможны и другие, “симметричные” приведенному, варианты ответа).

Задание “Год Кролика”

Вариант 1: $KP^2 + ОЛИК = 2011 - 23^2 + 1482 = 2011$

Вариант 2: $КРОЛ + ИК^2 = 2011 - 1570 + 21^2 = 2011$

Задание “Сумма степеней в год Кролика”

$$K^3 + P^3 + O^2 + L^2 + И^3 + K^3 = 2011 - 9^3 + 0^3 + 4^2 + 5^2 + 8^3 + 9^3 = 2011 \text{ или } 9^3 + 8^3 + 2^2 + 6^2 + 1^3 + 9^3 = 2011$$

(возможны и другие, “симметричные” приведенным, варианты ответа).

Ответы представили:

— Базылев Юрий, Варфоломеев Сергей, Галушкова Карина и Макаров Игорь, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Богданова Л.М.**;

— Бородкина Алевтина, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Васинская Екатерина, Газизуллин Артур и Сафиуллин Ильдар, Республика Башкортостан, г. Стерлитамак, школа № 17, учитель **Орлова Е.В.**;

— Владимирова Юлия, Моронцова Анастасия, Семенов Дмитрий, Тимофеев Анатолий и Яковлев

Дополнительное задание для самостоятельной работы

Используя законы логики, упростите приведенное в начале логическое выражение для выбора кошки. Сделав это, вы сможете изготовить и соответствующее устройство.

Полученное логическое выражение и/или фотографии изготовленного устройства, пожалуйста, присылайте в редакцию. Все изготовившие устройство будут награждены дипломом.

Литература

1. *Петцольд Ч.* Код. М.: Издательско-торговый дом “Русская редакция”, 2001.

Анатолий, основная школа села Именеве, Республика Чувашия, Красноармейский р-н, учитель **Тимофеева И.А.**;

— Вылка Татьяна, основная школа поселка Каратайка, Архангельская обл., учитель **Безумова В.А.**;

— Коростелев Андрей и Филимонова Галина, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Согомонян Серине, Воронежская обл., поселок Каменка, средняя школа № 1 им. Героя Советского Союза В.П. Захарченко, учитель **Старикова М.Е.**;

— Тимошенко Иван, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Ядзевичюс Стас, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**

2. Задача “Змей Горыныч курит и скоро умрет”

Ответ — 22 года.

Правильный ответ прислали:

— Базылев Юрий, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Богданова Л.М.**;

— Демьянова Елена, Костюнин Александр и Хомякова Анна, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Малышева Татьяна и Сафиуллин Ильдар, Республика Башкортостан, г. Стерлитамак, школа № 17, учитель **Орлова Е.В.**;

— Яновский Виталий, Москва, гимназия № 1530, учитель **Шамшев М.В.**

3. Задача “Кто какое место занял?”

Напомним условие. “На соревнованиях Эндрю, Боб, Серж и Дэн заняли первые четыре места. Когда их знакомые девушки начали вспоминать, как эти места распределились, то мнения разошлись:

1) Кэт: Эндрю был первым, Дэн — вторым;

2) Линда: Эндрю был вторым, Боб — третьим;

3) Мэри: Боб был четвертым, Серж — вторым.

Оказалось, что каждая девушка сделала одно правильное и одно неправильное заявление. Кто из юношей какое место занял?”

Решение

Допустим, первое заявление Кэт правдивое, а второе — нет:

Имя	Эндрю	Дэн	Боб	Серж
Место	1	Не второй		

При таком допущении в заявлении Линды первая часть — ложная, то есть правдивая — вторая. Запишем место Боба в таблицу:

Имя	Эндрю	Дэн	Боб	Серж
Место	1	Не второй	3	

или (Дэну “остаётся” 4-е место):

Имя	Эндрю	Дэн	Боб	Серж
Место	1	4	3	2

Полученная картина соответствует высказыванию Мэри (его первая часть ложная, вторая — истинная). Значит, можно сделать вывод о таком распределении мест: первое — Эндрю, второе — Серж, третье — Боб, четвертое — Дэн.

Аналогичный результат можно получить, начав рассуждения с высказываний Линды или Мэри, а также применив метод решения логических задач с помощью схем, разработанный О.Б. Богомоловой и описанный в газете “Информатика” № 8/2008.

Ответы прислали:

— Андриященко Александр и Свистунов Николай, Ставропольский край, Кочубеевский р-н, станица Барсуковская, школа № 6, учитель **Рябченко Н.Р.**;

— Базылев Юрий и Галушкова Карина, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Богданова Л.М.**;

— Васильева Юлия, Даутов Азат и Евченко Мария, Республика Башкортостан, г. Стерлитамак, школа № 17, учитель **Орлова Е.В.**;

— Захарова Мария и Осипова Дарья, Республика Карелия, г. Сегежа, школа № 5, учитель **Меньшиков В.В.**;

— Зорихин Алексей, Свердловская обл., г. Нижняя Салда, школа № 7, учитель **Зорихина Н.Ю.**;

— Кабанов Андрей и Макаров Петр, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Полухович Максим, Суминова Марина и Ядзевичюс Стас, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**;

— Тайбарей Нина, основная школа поселка Каратайка, Архангельская обл., учитель **Безумова В.А.**;

— Филимонова Галина, г. Пенза, школа № 512, учитель **Гаврилова М.И.**

4. Статья “Антианаграммы”

Ответы по первой группе слов

1. Носитель. 2. Накопитель. 3. Истина. 4. Протокол. 5. Трафик. 6. Кабель. 7. Магистраль. 8. Дорожка. 9. Ксерокс. 10. Документ. 11. Микропроцессор. 12. Координата. 13. Интерфейс. 14. Манипуля-

тор. 15. Ярлык. 16. Протокол. 17. Маршрутизатор. 18. Кластер.

Ответы по второй группе слов

1. Плата. 2. Сервер. 3. Адаптер. 4. Апорт.

Ответы прислали:

— Абдулова Эльвира, Буляккулов Рустам, Духнин Семен, Моисеев Игорь, Ноздрин Данил, Трифонова Екатерина и Шафиева Алина, Республика Башкортостан, г. Стерлитамак, школа № 17, учитель **Орлова Е.В.**;

— Андриященко Александр и Свистунов Николай, Ставропольский край, Кочубеевский р-н, станица Барсуковская, школа № 6, учитель **Рябченко Н.Р.**;

— Базылев Юрий и Галушкова Карина, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Богданова Л.М.**;

— Демьянова Елена, Костюнин Александр и Хомякова Анна, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Дюбарова Анастасия, Клименко Надежда, Кожевников Евгений, Лышко Александр, Мурашов Николай, Мурзагалиева Гульнара и Мухамадеев Равиль, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

— Зорихин Алексей, Свердловская обл., г. Нижняя Салда, школа № 7, учитель **Зорихина Н.Ю.**;

— Осипова Дарья, Республика Карелия, г. Сегежа, школа № 5, учитель **Меньшиков В.В.**;

— Полухович Максим, Суминова Марина и Ядзевичюс Стас, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**;

— Штатнов Сергей, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Шадрин Юлия, Чувашская Республика, г. Канаш, Канашский педагогический колледж, преподаватель **Воеводина Р.В.**

Почти все приславшие ответы привели развернутые комментарии к найденным терминам.

5. Числовой ребус с числом ABCD

Напомним, что необходимо было решить числовой ребус:

$$ABCD \times 4 = DCBA.$$

В нем одинаковые цифры зашифрованы одинаковыми буквами, разные цифры — разными буквами.

Решение

Умножение четырехзначного числа **ABCD** на 4 дает четырехзначное произведение при **A = 1** или при **A = 2**. Но единице **A** не может быть равно. Значит, **A = 2**. Запишем это в ребус:

×	2	B	C	D
				4
	D	C	B	2

Анализ последнего разряда показывает, что **D = 3** или **D = 8**. Но произведение четырехзначного числа **ABCD** на 4 не может начинаться с цифры 3. Следовательно, **D = 8**:

×	2	В	С	8
				4
	8	С	В	2

Видно, что $B < 3$. Так как цифра 2 уже “использована”, то $B = 0$ или $B = 1$. Анализ показывает, что выражение $4C + 3$ не может оканчиваться на 0 ни при каких значениях цифры C . Значит, $B = 1$. Все решение ребуса:

×	2	1	7	8
				4
	8	7	1	2

Правильные ответы представили:

— Базылев Юрий и Галушкова Карина, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Богданова Л.М.**;

— Дюбарова Анастасия, Клименко Надежда, Кожевников Евгений, Лышко Александр, Мурашов Николай, Мурзагалиева Гульнара и Мухамадеев Равиль, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

— Кондратьева Наталья, Мальшева Татьяна и Назаров Андрей, Республика Башкортостан, г. Стерлитамак, школа № 17, учитель **Орлова Е.В.**;

— Полюхович Максим, Суминова Марина и Ядзевичюс Стас, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**;

— Тайбарей Нина, основная школа поселка Каратайка, Архангельская обл., учитель **Безумова В.А.**;

— Яковлева Александра, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**

6. Задача “В школьном буфете”

Напомним условие.

В очереди в школьный буфет стояли Юрий, Михаил, Вазген, Семен и Омар.

Юрий стоял раньше Михаила, но после Омара. Вазген и Омар не стояли рядом, а Семен не находился рядом ни с Омаром, ни с Юрием, ни с Вазгеном. В каком порядке стоят ребята?

Ответ

Очередь стояла в таком порядке: Омар, Юрий, Вазген, Михаил, Семен.

Ответы представили:

— Андрущенко Александр и Свистунов Николай, Ставропольский край, Кочубеевский р-н, станица Барсуковская, школа № 6, учитель **Рябченко Н.Р.**;

— Базылев Юрий и Галушкова Карина, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Богданова Л.М.**;

— Васильева Юлия, Герасимов Артем, Даутов Азат и Евченко Мария, Республика Башкортостан, г. Стерлитамак, школа № 17, учитель **Орлова Е.В.**;

— Захарова Мария и Осипова Дарья, Республика Карелия, г. Сегежа, школа № 5, учитель **Меньшиков В.В.**;

— Полюхович Максим и Суминова Марина, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**;

— Тайбарей Нина, основная школа поселка Каратайка, Архангельская обл., учитель **Безумова В.А.**;

— Харитоновна Елена и Хомякова Анастасия, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**

7. Задача “Добейтесь равенства”

Напомним, что необходимо было сделать правильным выражение: $987654321 = 100$, используя 4 знака “+” или “-” в его левой части.

Ответ: $98 - 76 + 54 + 3 + 21 = 100$

Правильные ответы прислали:

— Базылев Юрий и Галушкова Карина, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Богданова Л.М.**;

— Дюбарова Анастасия, Клименко Надежда, Кожевников Евгений, Лышко Александр, Мурашов Николай, Мурзагалиева Гульнара и Мухамадеев Равиль, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

— Кольтякова Анна, Кондратьева Наталья и Мальшева Татьяна, Республика Башкортостан, г. Стерлитамак, школа № 17, учитель **Орлова Е.В.**;

— Осипова Дарья, Республика Карелия, г. Сегежа, школа № 5, учитель **Меньшиков В.В.**;

— Полюхович Максим, Суминова Марина и Ядзевичюс Стас, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**

8. Задание “Шесть вопросов. Вариант 4” (рубрика “Поиск информации”)

Ответы

1. Навархом Спарты в указанный в вопросе период был Лисандр.

2. Слова “Как Магдалина плачешь ты, и как русалка ты хохочешь” принадлежат русскому поэту Евгению Абрамовичу Баратынскому (Боратынскому).

3. От греческого слова “сладкий” происходит название простейшего трехатомного спирта — глицерина.

4. Речь в вопросе идет о Великой Китайской стене. Ее в 214 году до нашей эры приказал построить император Цинь Шихуанди.

5. Канал Санкт-Петербурга, который отходит от реки Екатерингофки, охватывает с севера парк Екатерингоф и у Сутугина моста соединяется с рекой Таракановкой, называется “Бумажный канал”.

6. Французский посол в Бельгии герцог Плесси-Пралин в 1671 году создал рецепт приготовления десерта, который затем был назван его именем — “пранлине”.

Ответы прислали:

— Базылев Юрий и Галушкова Карина, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Богданова Л.М.**;

— Буханов Василий, Григорьев Кирилл и Юхтенко Илья, г. Воронеж, лицей № 2, учитель **Комбаров С.И.**;

— Зорихин Алексей и Шишкина Анастасия, Свердловская обл., г. Нижняя Салда, школа № 7, учитель **Зорихина Н.Ю.**;

— Макурина Юлия и Ходак Ольга, Пензенская обл., поселок Тамала, школа № 1, учитель **Пашина Н.Д.**;

— Микулик Илья, г. Астрахань, школа № 33 им. Н.А. Мордовиной, учитель **Лепехина С.М.**;

— Ноздрин Данил, Республика Башкортостан, г. Стерлитамак, школа № 17, учитель **Орлова Е.В.**;

— Полюхович Максим, Суминова Марина и Ядзевичюс Стас, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**;

— Тайбарей Нина, основная школа поселка Каратайка, Архангельская обл., учитель **Безумова В.А.**;

— Шадрин Юлия, Чувашская Республика, г. Канаш, Канашский педагогический колледж, преподаватель **Воеводина Р.В.**

Большинство приславших ответы привели развернутые комментарии к ним, в том числе с указанием использованных источников информации.

Решения двух японских головоломок “судоку” прислали:

— Базылев Юрий и Галушкова Карина, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Богданова Л.М.**;

— Осипова Дарья, Республика Карелия, г. Сегежа, школа № 5, учитель **Меньшиков В.В.**;

— Кондратьева Наталья, Малышева Татьяна, Семенова Наталья, Республика Башкортостан, г. Стерлитамак, школа № 17, учитель **Орлова Е.В.**;

— Суминова Марина, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**;

— Яковлева Александра, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**

Задачу “Семь кошельков” правильно решили также:

— Вылка Татьяна, основная школа поселка Каратайка, Архангельская обл., учитель **Безумова В.А.**;

— Дюбарова Анастасия, Клименко Надежда, Лышко Александр, Кожевников Евгений, Мурашов Николай, Мурзагалиева Гульнара и Мухамадеев Равиль, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

— Мнацаканян Ашот, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**

Татьяна Вылка прислала также правильное решение задачи “Три министра”.

Решения заданий, опубликованных в нашей газете в феврале, представили также:

— ученики средней школы поселка Озеры Красноярского края (учитель **Филипченко И.С.**):

1) Тарасюк Степан и Юнусова Гульнара — задача “120 тетрадей”;

2) Аникин Сергей, Бондаренко Сергей, Васильева Галина, Донец Андрей, Тарасюк Степан и Хорькова Светлана — ребусов по информатике;

— ученики Караклинской средней школы, Чувашская Республика, Канашский р-н (учитель **Макарова Л.Ф.**):

1) Соловьев Сергей — задачи “Улитка и конфета”;

2) Иванов Вадим — задачи “Стая гусей и три черепахи”;

3) Коньков Александр и Ефимов Петр — японских головоломок “судоку”;

— ученики основной школы села Красный Бор Нижегородской обл. (учитель **Баженова С.А.**):

1) Евсеев Павел и Кондратьев Владимир — задачи “Кто выше?”;

2) Антоненко Татьяна, Антоненко Юлия и Малыхина Анна — головоломки “Как получить три квадрата?”;

— ученики школы № 17 г. Стерлитамак, Республика Башкортостан (учитель **Орлова Е.В.**):

1) Девицын Артем — задач “Семь кошельков” и “Второе дело об украденном компьютере”;

2) Алтыnguзина Эльза и Кондратьева Наталья — головоломки “Слова на диагоналях”.

Спасибо всем!

Штирлиц и IP-адрес

Штирлиц попал в XXI век. Ему для связи дали IP-адрес, но по ошибке он порвал листок с адресом на 4 части:



Разведчик обратился к школьнику, изучающему информатику, и тот рассказал ему, что обычно IP-адреса записываются в виде четырех неотрицательных целых чисел, меньших 256, разделенных точками. После этого Штирлиц восстановил нужный адрес. А вы сможете это сделать?

На олимпиаде по информатике

Об учениках, занявших первые пять мест на олимпиаде по информатике, имеется пять высказываний:

1) первое место занял Вася, а Юра — второе;

2) Саша занял второе место, а Вася — пятое;

3) второе место занял Иван, а Гриша оказался четвертым;

4) на первом месте был Гриша, а Юра — четвертым;

5) Юра был четвертым, а Иван — вторым.

Известно, что в каждом высказывании одно утверждение верное, а второе — нет. Кто какое занял место?

Наименьшее число

Пользуясь только двумя одинаковыми цифрами и знаками действий, запишите как можно меньшее число.

Индюки и жеребята³

По дороге вдоль кустов шло 11 хвостов.

Подсчитать я также смог, что шагало 30 ног.

Это вместе шли куда-то индюки и жеребята.

И вопрос мой к вам таков: “Сколько было индюков?”

И хотел бы также знать, сколько было жеребят?

Решите задачу, не составляя систему уравнений, а методом рассуждений, который был описан в газете “В мир информатики” № 157, 165 (“Информатика” № 3/2011, 11/2011).

Сколько весит бидон?⁴

Полный бидон с молоком весит 7 кг, а наполненный наполовину — 4 кг. Сколько весит бидон?



Пять карточек

Имеются пять карточек, на которых записана одна из букв А, Б, Е, Ж, И. Надо составить последовательность из k карточек, соблюдая такие правила:

1) любая последовательность начинается буквой А;

2) после гласной буквы не может снова идти гласная, а после согласной — согласная;

3) буквы в последовательности не должны повторяться.

Запишите все допустимые последовательности карточек для:

1) $k = 3$;

2) $k = 4$;

3) $k = 5$.

Из Ёлкино — в Палкино

Между четырьмя местными аэропортами: Ёлкино, Палкино, Веревино и Булкино — ежедневно выполняются авиарейсы. Расписание перелетов рейсов между ними:

Аэропорт вылета	Аэропорт прилета	Время вылета	Время прилета
Ёлкино	Веревино	7:30	10:50
Булкино	Ёлкино	8:15	10:35
Палкино	Веревино	11:35	13:25
Веревино	Ёлкино	12:10	14:20
Ёлкино	Булкино	12:30	14:30
Булкино	Палкино	14:10	16:20
Ёлкино	Палкино	14:15	16:40
Веревино	Палкино	14:20	16:30
Палкино	Ёлкино	16:10	18:50
Палкино	Булкино	18:40	20:45

Представьте, что вы оказались в аэропорту Ёлкино в 6 часов утра. Каково самое раннее время, когда вы сможете попасть в аэропорт Палкино? Время на пересадку из самолета в самолет не учитывать.

ПОИСК ИНФОРМАЦИИ

Четыре вопроса

Ответы на приведенные ниже вопросы найдите в Интернете или по другим источникам информации.

1. Название какой модели автомобильной фирмы Renault связано с Интернетом?

2. Приведите еще одну-две фамилии в этом ряду: Ней, Виктор, ...

3. Какие из существующих в настоящее время государств образовались в результате объединения разных государств и какие — в результате разъединения? (Приведите несколько примеров.)

4. Какова была (за время наблюдений) самая низкая и самая высокая температура воздуха в Москве 26 октября?

Ответы присылайте в редакцию (можно отвечать не на все вопросы).

³ Задачу, которая была опубликована в “Календаре школьника” в 1981 году, представила Федорова Л.А., учитель Московского кадетского корпуса “Пансион воспитанниц МО РФ”.

⁴ Задача предназначена для учащихся начальной школы и учеников 5–7-х классов.

ДЛЯ ЭРУДИТОВ

Четыре столицы

О каждой из этих четырех столиц дано лишь несколько фактов. Тем не менее их вполне возможно узнать. Попробуйте!

1

а) В русском языке ее название является омонимом для сельскохозяйственной постройки, в которой сушат и обмолачивают снопы;

б) она входила в Ганзейский торговый союз с 1282 года, первое же упоминание о ней относится к 1198 году. Однако официальной датой основания города является 1201 год;

в) одной из самых “легендарных” построек в историческом центре города является “Дом с черными котами”;

г) в разные периоды она входила в состав Швеции, Речи Посполитой, Российской империи, Германии;

д) среди прочего ее прославил алкогольный напиток.

2

а) Она расположена у одноименного залива Яванского моря и разделена на западный и восточный районы рекой Чиливунг;

б) наиболее раннее упоминание этой столицы относится к периоду существования королевства Тарумы в IV веке нашей эры;

в) сейчас в этом городе проживает 23 миллиона человек;

г) ...но, несмотря на это, столицу хотят перенести — из-за частых затоплений;

д) официально она является не городом, а провинцией со статусом столицы, поэтому управляется не мэром, а губернатором.

3

а) Он является самой западной столицей континентальной Европы;

б) на севере его расположен международный аэропорт Портела;

в) по легенде, этот город был основан Одиссеем, давшим ему свое имя (хотя и в измененной форме);

г) более трех столетий этот город находился под властью исламского халифата;

д) одно из наиболее разрушительных и смертоносных землетрясений в истории 1 ноября 1755 года разрушило этот город и унесло жизни более 100 тысяч человек за 6 минут.

4

а) Длинное и красивое название этой столицы означает “тысяча деревень” или “город тысячи воинов”;

б) она была основана в начале XVII в. правителем Имерины Андриандзакой;

в) три ее района расположены на трех холмах и соединены между собой тоннелями и лестницами;

г) более полувека город был центром французской колонии;

д) одна из главных достопримечательностей — королевский дворец Рува — долгое время был и единственным каменным зданием на острове.

Ответы присылайте в редакцию (можно решать не все задания).

GAMES.EXE

Игры на клетчатой доске

В прошлом учебном году были описаны две игры — “Короли на клетчатой доске” и “Доминошки на клетчатой доске”.

Условие первой из них: “Двое по очереди ставят шахматных королей в клетки доски размером 9×9 клеток так, чтобы они не били друг друга (цвет фигур значения не имеет). Проигрывает тот, кто не сможет сделать ход”. Необходимо определить, кто выиграет в эту игру — начинающий ее или делающий ход вторым.

Вторая игра такая. “Двое играют в такую игру на клетчатой доске размером 10×10 клеток. За ход разрешается накрыть любые две соседние клетки доминошкой (прямоугольником 1×2 клетки) так, чтобы доминошки не перекрывались. Проигрывает тот, кто не сможет сделать ход. Кто выиграет — начинающий ее или делающий ход вторым?”

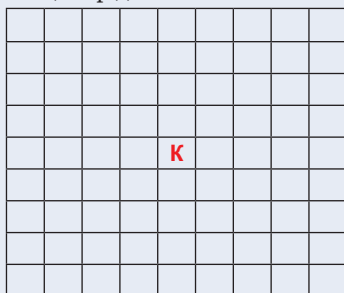
Прежде чем проводить анализ игр, вспомним еще одну игру “Монеты на столе”, опубликован-

ную в газете “В мир информатики” № 148 (“Информатика” № 18/2010). В ней двое по очереди кладут одинаковые монеты на круглый стол, причем так, чтобы они не накладывались друг на друга. Имеется неограниченное число монет. Проигрывает тот, кто не сможет сделать ход. Кто выиграет в эту игру — начинающий ее или делающий ход вторым?

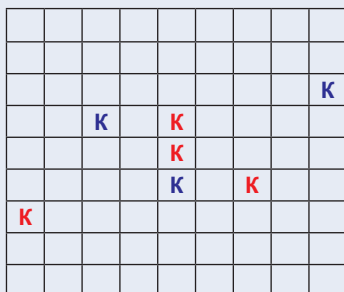
В этой игре выигрывает тот, кто начинает игру, независимо от размеров стола. Первым ходом он должен положить монету так, чтобы центры монеты и стола совпали. После этого на каждый ход второго игрока начинающий должен класть монету симметрично относительно центра стола. При такой стратегии после каждого хода первого игрока позиция будет симметрична. Поэтому если возможен очередной ход второго игрока, то возможен и симметричный ему ответный ход первого. Когда-то делающий ход вторым разместить монету на столе не сможет. Следовательно, побеждает первый игрок.

Использование симметрии лежит в основе и двух первых игр.

Рассмотрим игру с шахматными королями. Начинаящий ее должен своим первым ходом поставить короля в центр доски:

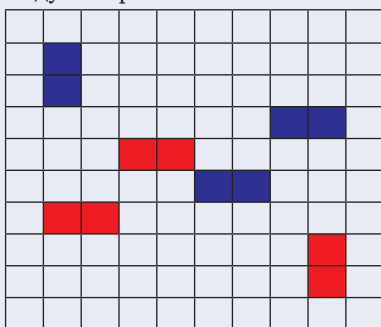


а после каждого хода соперника — размещать “своих” королей симметрично относительно центра:



Следовательно, при такой стратегии начинающего игрока на очередной ход второго игрока всегда будет возможность ответного хода. Но после того как будут заняты все клетки доски, второй игрок сделать ход не сможет и — проигрывает.

Во второй игре делающий ход вторым также должен размещать свою доминошку симметрично последнему ходу соперника:



Ясно, что если возможен очередной ход начинавшего игрока, то возможен и симметричный ему ответный ход второго участника, который в конце концов и выиграет.

Задания для самостоятельной работы

1. Подумайте, пожалуйста, над вопросом: “Зависит ли результат игры «Короли на клетчатой доске» от размеров квадратной доски?” А игры “Доминошки на клетчатой доске”?

2. Двое по очереди ставят коней в клетки шахматной доски так, чтобы они не били друг друга (цвет коней значения не имеет). Проигрывает тот, кто не сможет сделать ход. Кто выиграет в эту игру — начинающий ее или делающий ход вторым?

Ответы присылайте в редакцию (можно отвечать не на все вопросы).

Игра “Три кучки камней”

Имеются три кучки камней: в первой — 10, во второй — 15, в третьей — 20. Играют двое. За ход разрешается разбить любую кучку на две меньшие. Проигрывает тот, кто не сможет сделать ход. Кто выиграет в эту игру — начинающий ее или делающий ход вторым?

Такая задача была опубликована в одном из весенних номеров. Благодаря Юрия Базылева (Республика Карелия, поселок Надвоицы, школа № 1, учитель **Богданова Л.М.**), Татьяну Малышеву и Азата Даутова (Республика Башкортостан, г. Стерлитамак, школа № 17, учитель **Орлова Е.В.**) и Марину Суминову (Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**), приславших ответы, редакция предлагает читателям еще раз подумать над ответом на поставленный вопрос.

Указание по выполнению. Рассмотрите сначала вариант игры с одной кучкой камней при их небольшом количестве.

ШКОЛА ПРОГРАММИРОВАНИЯ

Сколько квадратов в круге?

Нет, уважаемый читатель, речь не будет идти о задаче, которую называют “квадратура круга”, — задаче точного построения квадрата, равновеликого по площади кругу⁵. Мы будем решать следующую задачу: “На координатной плоскости нарисована окружность радиуса R (R — целое число) с центром в начале координат. Сколько полных квадратов размером 1×1 находится внутри этой окружности?”

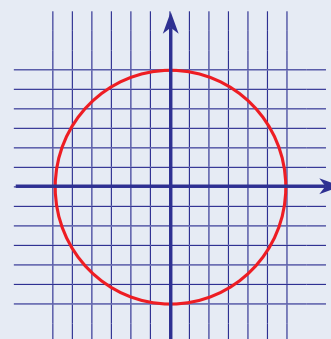


Рис. 1

⁵Эту задачу пытались решить первоначально с помощью циркуля и линейки. Попытки решения задачи, продолжавшиеся в течение тысячелетий, неизменно оканчивались неудачей. С 1775 года Парижская академия наук, а затем и другие академии стали отказываться от рассмотрения работ, посвященных **квадратуре круга**. Лишь в XIX веке было дано научное обоснование этого отказа: строго установлена неразрешимость задачи с помощью циркуля и линейки.

Очевидное решение — перебрать все возможные квадратики и проверить, располагается ли каждый из них внутри окружности. Соответствующая схема программы:

```

цикл для каждого квадратика
  если квадратик внутри окружности
  то
    Увеличить счетчик числа квадратов на 1
  все
конец цикла
    
```

В дальнейших рассуждениях и в программе будем использовать следующие основные величины:

- R — радиус окружности;
- $X_{лн}$, $X_{пн}$, $X_{лв}$, $X_{пв}$ — координаты по оси x , соответственно, левой нижней, правой нижней, левой верхней и правой верхней вершин квадрата;
- $U_{лн}$, $U_{пн}$, $U_{лв}$, $U_{пв}$ — то же по оси y ;
- n — искомое количество квадратов.

Перебор, о котором шла речь чуть выше, можно провести, рассмотрев, например, все возможные значения величины $X_{лн}$, а для каждого из них — все возможные значения величины $U_{лн}$:

```

нц для всех возможных значений  $X_{лн}$ 
  нц для всех возможных значений  $U_{лн}$ 
    Определить координаты трех остальных
    вершин квадратика
    если квадратик внутри окружности
    то
      Увеличить счетчик числа квадратов на 1
    все
  кц
кц
    
```

Координаты x и y трех остальных вершин некоторого квадратика в зависимости от $X_{лн}$ и $U_{лн}$ найти просто:

```

 $X_{пн} = X_{лн} + 1$ 
 $X_{лв} = X_{лн}$ 
 $X_{пв} = X_{лн} + 1$ 
 $U_{пн} = U_{лн}$ 
 $U_{лв} = U_{лн} + 1$ 
 $U_{пв} = U_{лн} + 1$ 
    
```

А как проверить тот факт, что квадратик полностью находится внутри окружности? Ответ такой — это имеет место, если при заданных значениях абсцисс вершин одновременно справедливы два условия (см. рис. 2):

1) координаты по оси y верхней левой и верхней правой вершин квадрата не больше, чем соответствующее значение ординаты соответствующей точки верхней половины окружности;

2) координаты по оси y нижней левой и нижней правой вершин квадрата не меньше, чем соответствующее значение ординаты соответствующей точки нижней половины окружности:

```

если  $U_{лв} \leq U_{впо}$  и  $U_{пв} \leq U_{впо}$  и
       $U_{лн} \geq U_{нпо}$  и  $U_{пн} \geq U_{нпо}$ 
  то
     $n := n + 1$ 
все
    
```

где $U_{впо}$ — ордината соответствующей точки верхней половины окружности, $U_{нпо}$ — нижней половины.

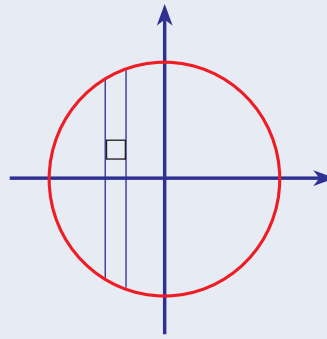


Рис. 2

Зависимость ординаты y некоторой точки окружности от ее абсциссы x можно получить, зная уравнение окружности ($x^2 + y^2 = R^2$):

$$y = \sqrt{R^2 - x^2},$$

причем для верхней половины окружности следует взять положительное значение корня, а для нижней — отрицательное. Учитывая это, можем записать условие размещения квадратика внутри окружности:

$U_{лв} \leq \text{sqrt}(R * R - X_{лв} * X_{лв})$ и $U_{пв} \leq \text{sqrt}(R * R - X_{пв} * X_{пв})$ и $U_{лн} \geq \text{sqrt}(R * R - X_{лн} * X_{лн})$ и $U_{пн} \geq \text{sqrt}(R * R - X_{пн} * X_{пн})$, где sqrt — функция, возвращающая квадратный корень из своего аргумента.

Видно, что для каждой половины окружности проверка проводится дважды (для левой и правой точек). Поэтому целесообразно разработать функцию, определяющую значение ординаты для той или иной половины окружности в зависимости от заданного значения абсциссы x . Для верхней половины такая функция имеет вид:

```

алг вещь  $U_{впо}(\text{арг цел } x)$ 
нач
  знач :=  $\text{sqrt}(R * R - x * x)$ 
кон
    
```

— для нижней:

```

алг вещь  $U_{нпо}(\text{арг цел } x)$ 
нач
  знач :=  $-\text{sqrt}(R * R - x * x)$ 
кон
    
```

С использованием этих функций основная программа может быть оформлена следующим образом:

```

цел  $R$ 
 $R := \dots$ 
алг Вариант 1
нач цел  $X_{лн}$ ,  $X_{пн}$ ,  $X_{лв}$ ,  $X_{пв}$ ,  $U_{лн}$ ,  $U_{пн}$ ,  $U_{лв}$ ,
 $U_{пв}$ ,  $n$ 
   $n := 0$ 
  нц для  $X_{лн}$  от  $-R$  до  $R - 1$  |См. рис. 1
    нц для  $U_{лн}$  от  $-R$  до  $R - 1$ 
       $X_{пн} := X_{лн} + 1$ 
       $X_{лв} := X_{лн}$ 
       $X_{пв} := X_{лн} + 1$ 
       $U_{пн} := U_{лн}$ 
    
```

```

Улв := Улн + 1
Упв := Улн + 1
если Улв <= Увпо(Хлв) и
    Упв <= Увпо(Хпв) и
    Улн >= Унпо(Хлн) и
    Упн >= Унпо(Хпн)
то
    n := n + 1
все
кц
кц
вывод n
кон

```

Примечание. Так как величина R используется во вспомогательных функциях, то она описана как общая (глобальная).

Но, как говорят программисты, “чем быстрее начнешь писать программу, тем хуже она получится”. Имеется в виду, что нужно постараться продумать все детали, особенности и найти и применить самые рациональные варианты решения. В нашем случае можно значительно сократить объем вычислений.

Во-первых, можно подсчитать искомое количество только для одной четверти координатной плоскости, а затем умножить результат на 4.

Кроме того, можно учесть, что квадраты, находящиеся в крайних рядах (столбцах и строках), точно не размещаются внутри окружности, и их можно не рассматривать.

Предлагаем читателям разработать вариант программы (на изучаемом языке программирования), учитывающий два указанных обстоятельства, и прислать его в редакцию. Фамилии всех

приславших правильную программу будут опубликованы.

Может также возникнуть идея решения, основанная на рассмотрении “столбиков” из квадратов и сравнении целых частей $Увпо(X)$ и $Увпо(X + 1)$. Соответствующая программа такая:

```

алг Вариант 3
нач цел X, n
n := 0
нц для X от 0 до R - 2
если int(Увпо(X)) = int(Увпо(X + 1))
то
    n := n + int(Увпо(X))
иначе
    n := n + int(Увпо(X)) - 1
все
кц
вывод nс, n * 4
кон

```

В ее особенностях разберитесь самостоятельно. Проверьте, можно ли использовать этот вариант при любых значениях R . Как надо изменить программу, чтобы она работала при любых R ?

И еще. Если провести расчеты, то можно получить, что при $R = 2$ количество квадратов $n = 4$, а при $R = 3 - n = 16$, т.е. для этих случаев $n = (2R - 2)^2$. Предлагаем читателям проверить, справедлива ли полученная формула для других значений R , или можно получить другую общую формулу, с помощью которой можно рассчитывать значение n , не используя разработанную программу.

Ответы в части использования последнего варианта программы (с обоснованием) и результаты исследований в части формулы, пожалуйста, также присылайте в редакцию.

ЛИЧНОСТИ

190-летие со дня рождения П.Л. Чебышёва

▶ В мае 2011 года исполнилось 190 лет со дня рождения Пафнутия Львовича Чебышёва, выдающегося российского математика, механика, одного из первых создателей отечественной вычислительной техники.

П.Л. Чебышёв родился 4 (16) мая 1821 г. в деревне Окатьево Боровского уезда Калужской губернии в семье богатого землевладельца. Первоначальное воспитание и образование получил дома.

В 1832 году семья переезжает в Москву, чтобы продолжить образование взрослых детей. Летом 1837 года Чебышёв начинает изучение математики в Московском университете на втором фило-софском отделении.

В 1838 году, участвуя в студенческом конкурсе, получил серебряную медаль за работу по нахождению корней уравнения n -й степени. Оригинальная работа была закончена уже в 1838 году

и сделана на основе алгоритма Ньютона. За работу Чебышёв был отмечен как самый перспективный студент.

В 1841 году Пафнутий Львович успешно заканчивает университет и защищает диссертацию.

В 1847 году Чебышёв утвержден в звании доцента и начинает читать лекции по алгебре и теории чисел в Петербургском университете.

В 1850 году он защищает докторскую диссертацию и становится профессором Петербургского университета. Эту должность Чебышёв занимал до старости.

Заслуги П.Л. Чебышёва оценены были ученым миром достойным образом. Его избрали своим действительным членом:

- Петербургская академия наук (1853 г.);
- Берлинская академия наук;



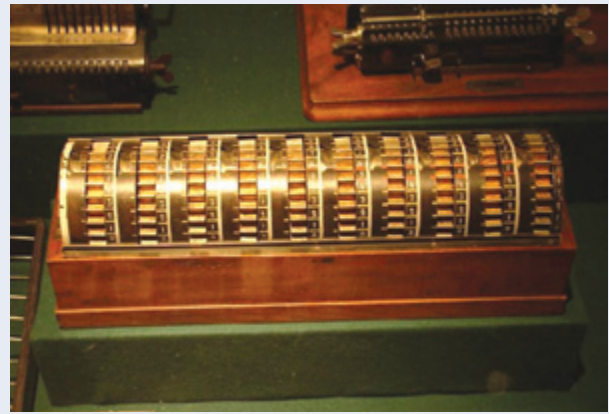
- Болонская академия наук;
- Парижская академия наук (1860 г.).

Избран также членом-корреспондентом Лондонского королевского общества, Шведской академии наук и др., всего 25 различных академий и научных обществ. П.Л. Чебышёв состоял также почетным членом всех российских университетов.

В творчестве Пафнутия Чебышёва удивительным образом сочетаются глубокие исследования по “чистой” математике и устойчивый интерес к сугубо прикладным вопросам. Он был не только математиком, но и изобретателем, причем в основе всех его изобретений лежало стремление внести возможную экономию труда и времени во всякую работу.

К числу выдающихся изобретений Чебышёва прежде всего следует отнести вычислительное устройство — арифмометр, сконструированный им в 1878 году, который, несмотря на сложность, считался одной из наиболее совершенных существующих тогда машин такого рода.

Отличительная особенность арифмометра Чебышёва заключалась в весьма оригинальном при-



способлении для перенесения десятков. Им были разработаны два варианта устройства:

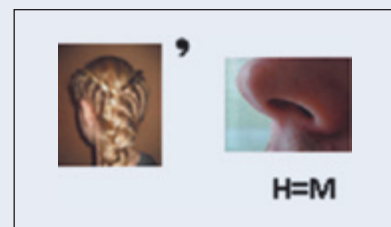
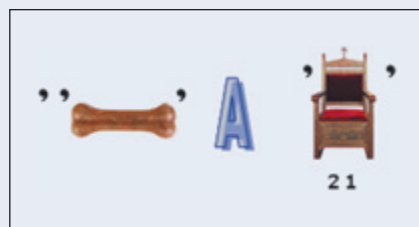
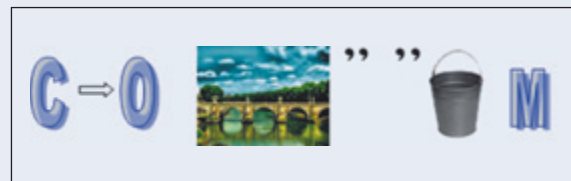
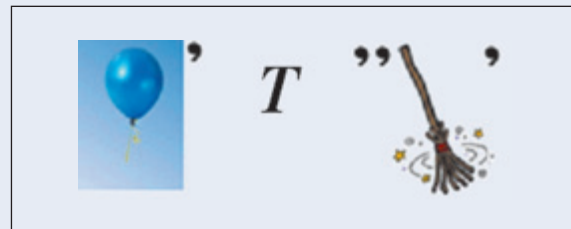
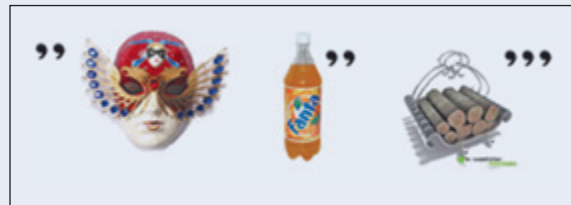
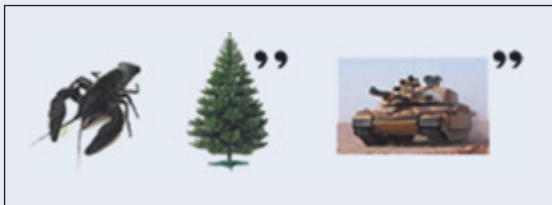
- 1) суммирующая машина;
- 2) арифмометр с приставкой для умножения и деления.

П.Л. Чебышёв скончался 8 декабря 1894 года за письменным столом. Погребен в родном имении в селе Спас-Прогнанье, которое находится в 90 км от Москвы.

ТВОРЧЕСТВО НАШИХ ЧИТАТЕЛЕЙ

Ребусы, посвященные Году космонавтики

► Предлагаем читателям решить ребусы, которые разработали Анатолий и Ульяна Тимофеевы, ученики Именевской основной школы, Красноармейский р-н Чувашской Республики (учитель Тимофеева И.А.).



Ответы присылайте в редакцию (можно решать не все ребусы).

Азбука информатики в определениях

Н.А. Владимирова,
учитель информатики
гимназии № 2,
г. Заозерный
Красноярского края

По приведенным ниже определениям (иногда — шуточным) найдите соответствующие термины, связанные с информатикой и компьютерами.

А 1. Место жительства ячейки.
2. Прапрадедушка калькулятора.
3. МЧС против вируса.

Б 1. Единица информации — малютка.
2. Управляющий гиперссылками.
3. Обменщик данными между приложениями.

В 1. Микроб в компьютере.
2. Связист между процессором и монитором.
3. Организатор видеоконференций на бескрайних просторах Интернета.

Г 1. Компьютерные искусства.
2. “Паутинный” документ.
3. “Переходник” к другому документу.

Д 1. Пожиратель дисков.
2. Все файлы, как листья и ветки.
3. “#” (не “тюрьма”).

Е 1. Академик, первопроходец школьной информатики.
2. Автор самого древнего алгоритма.
3. Условная команда в алгоритме (русский вариант).

Ж 1. Особый режим в компьютере.
2. Чем больше это свойство диска, тем легче нам играть без риска.
3. Долговременный хранитель данных.

З 1. Вид локальной сети с “лучами” к каждому компьютеру.
2. Наглядное представление объекта в операционной системе.
3. Уменьшитель нагрузки на экран.

И 1. Работник в информатике.
2. Этой сетью ловят знания, опыт и друга.
3. Ваш любимый школьный предмет (и специальная наука).

К 1. Бегунок на экране.
2. Тренажер для пальцев.
3. Прибор — “копировщик”.

Л 1. Без нее рассуждать нельзя.
2. Учетное имя для доступа в сеть.
3. Объединитель компьютеров в классе.

М 1. Компьютерное шоссе.
2. Визуальный воспроизводитель символьной и графической информации.
3. Главная плата компьютера.

Н 1. Место скопления информации.
2. Компактный портативный компьютер или электронная записная книжка.
3. Одна из двух главных цифр в компьютере.

О 1. Устройство — одиноличник, о существовании которого знает только процессор.
2. Обрамленная часть экрана.
3. То, над чем проводится операция (не хирургическая).

П 1. У человека — судьба, а у компьютера — ...
2. Секретное слово.
3. “Мозг” компьютера.

Р 1. Местохранилище информации.
2. Сортировщик популярности сайтов.
3. Место в записи числа.

С 1. Работник на узловых компьютерах.
2. Электронный карандаш или ручка.
3. Побратим того, что КЗ.

Т 1. Разделяет целую и дробную части числа.
2. Времязадающая электронная схема.
3. Переводчик программы на язык, понятный компьютеру.

У 1. Сжатый.
2. Алгоритм — развилка.
3. Арифметическая операция.

Ф 1. Хранитель информации, имеющий особое имя.
2. В электронной таблице начинается с “=”.
3. Частица программы.

Х 1. Жаргонное название устройства для подключения компьютеров к локальной сети (происходит от английского слова, означающего “центр деятельности”).
2. Любитель исследовать чужие файлы в Интернете.
3. Размещение сайта или информации на специальном обо-

рудовании, которое обеспечивает его доступность для пользователей сети Интернет.

- Ц**
1. Часть числа.
 2. Многократное повторение кода.
 3. Алгоритм-“попугай”.

- Ч**
1. Набор микросхем.
 2. Виртуальное место встречи в Интернете.
 3. Считывание информации из памяти.

- Ш**
1. Магистраль — связь процессора с оперативной памятью.
 2. Система условных знаков для секретного письма.
 3. Набор символов определенного типа.

- Щ**
1. Действие над мышью.
 2. Это первое блюдо любят многие программисты.
 3. Почти синоним слова “зонд”.

- Э**
1. Лицо компьютера.
 2. Имитатор в информатике.
 3. Свойство оперативной памяти.

- Ю**
1. Пользователь ПК.
 2. Английское “использовать” по-русски.
 3. Народные коллекции видеоклипов на сайте (по-русски).

- Я**
1. Улей — таблица, а соты — ...
 2. Указатель на объект.
 3. Российский поисковик.

Ответы присылайте в редакцию (можно не со всеми терминами).

Числовой ребус “ГРОМ ГРЕМИ”

Решите, пожалуйста, числовой ребус:

$$\begin{array}{r}
 \text{СМЕХ} \\
 + \\
 \text{ГРОМ} \\
 \hline
 \text{ГРЕМИ}
 \end{array}$$

Одинаковыми буквами зашифрованы одинаковые цифры, разными буквами — разные цифры.

101 монета, одна фальшивая

Среди 101 одинаковой по виду монеты есть одна фальшивая, отличающаяся по весу от настоящих. Как за два взвешивания с помощью чашечных ве-

сов без гирь определить, легче или тяжелее настоящих фальшивая монета? Находить фальшивую монету не требуется.

Эликсир бессмертия

Есть двое песочных часов: на 3 минуты и на 8 минут. Для приготовления эликсира бессмертия его надо непрерывно готовить ровно 7 минут. Как это сделать за минимально возможное число операций (установок часов и их переворачиваний)? Время, затрачиваемое на переворачивание часов, не учитывать.

ГРАФИКА

Что за изображение?⁶

По заданным в скобках координатам точек, используя клетчатую или так называемую “миллиметровую” бумагу, постройте изображение, состоящее из нескольких частей.

Часть 1

(-1,5; 3), (-2,5; 1), (-3; -1), (-2,5; -5), (-3,5; -5,5), (-4; -6), (-4; -6,5), (-1,5; -6,5), (-1; -6), (-1; -2), (0; -1,5), (1; -2), (1; -6), (1,5; -6,5), (4; -6,5), (4; -6), (3,5; -5,5), (2,5; -5), (3; -1), (2,5; 1), (1,5; 3), (4; -1), (4; -4), (5; -4), (5,5; -5), (5,25; -5,5), (6; -6), (6; -6,75), (5,5; -7,5), (3,5; -8), (1; -8), (0,5; -8,25), (-0,5; -8,25), (-1,5; -7,75), (-0,5; -7,25), (0,5; -7,25), (1; -7,5), (3,5; -7,5), (5; -7,25), (5,5; -6,75), (5,5; -6,25), (5; -6), (4; -6), (4; -6,5), (1,5; -6,5), (1; -6), (-1; -6), (-1,5; -6,5), (-4; -6,5), (-4; -6), (-5; -6), (-5,5; -5), (-5; -4), (-4; -4), (-4; 1), (-1,5; 3), (-1; 3,25), (-0,5; 2,5), (0; 3), (-0,5; 2,5), (1; 3,25), (-1,5; 3), (2; 3,5), (2,5; 3,5), (2,5; 4), (3; 4), (3; 4,5), (3,5; 5), (3; 5,5), (3,5; 6), (3; 6,5), (3,5; 7), (3; 7,5), (3; 8), (2,5; 8), (2,5; 8,5), (2; 8,5), (2; 9), (1,5; 8,75), (1; 9,25), (0,5; 9), (0; 9,5), (-0,5; 9), (-1; 9,25), (-1,5; 8,75), (-2; 9), (-2; 8,5), (-2,5; 8,5), (-2,5; 8), (-3; 8), (-3; 7,5), (-3,5; 7), (-3; 6,5), (-3,5; 6), (-3; 5,5), (-3,5; 5), (-3; 4,5), (-3; 4), (-2,5; 4), (-2,5; 3,5), (-2; 3,5), (-1,5; 3).

Часть 2

(0; 3,5), (-1,5; 4), (-2,5; 5), (-2,75; 6), (-2,5; 7), (-1,5; 8), (0; 8,5), (1,5; 8), (2,5; 7), (2,75; 6), (2,5; 5), (1,5; 4), (0; 3,5).

Часть 3

1) (-3; 7,5), (-3,5; 8), (-3,5; 8,5), (-3; 9), (-2; 9);
2) (2; 9), (3; 9), (3,5; 8,5), (3,5; 8), (3; 7,5).

⁶ Задание разработала Нарушева Мария, Большетолкайская средняя школа, Самарская обл., учитель Тараканова Т.В.

Часть 4

(-1; 6,5), (1; 6,5).

Часть 5

(-0,5; 4,25), (0; 4,5), (0,5; 4,25), (1; 4,5), (1; 5), (0,5; 5,25), (0; 5), (-0,5; 5,25), (-1; 5), (-1; 4,5), (-0,5; 4,25), (0; 4), (0,5; 4,25).

Часть 6

- 1) (0,5; 5), (2;6);
- 2) (0,5; 4,75), (2,5; 4,75);
- 3) (0,5; 4,5), (2; 4);
- 4) (-0,5; 4;5), (-2; 4);
- 5) (-0,5; 4,75), (-2,5; 4,75);
- 6) (-0,5; 5), (-2; 6).

Что получилось? Что в задании не соответствует правилам построения изображений по пикселям?

ВНИМАНИЕ! КОНКУРС

Конкурс № 87

“Уменьшить количество знаков «+»”

В предыдущем номере нашей газеты была описана программа, моделирующая известную игру крестики-нолики. В ней решается такая частная задача. В двумерном массиве размером 3 на 3, заполненном числами, для каждого элемента проводится суммирование значений в соответствующей строке и в соответствующем столбце, а для некоторых элементов — и на соответствующей диагонали / на соответствующих диагоналях массива.

A	B	C
D	E	F
G	H	I

Например, для элементов, обозначенных **B**, **D**, **F** и **H**, осуществляется суммирование значений в строке (*сумма1*) и в столбце (*сумма2*):

при $x = 1$ и $y = 2$:

```
сумма1 := поле[1, 1] +
        поле[1, 2] + поле[1, 3]
сумма2 := поле[1, 2] +
        поле[2, 2] + поле[3, 2]
```

...

— где *поле* — имя массива с числами, x — номер строки, y — номер столбца.

Для всех четырех элементов знак “+” используется здесь 16 раз.

Для элементов массива, обозначенных **A**, **C**, **G** и **I**, проводится суммирование значений в строке (*сумма1*), в столбце (*сумма2*), на главной (*сумма3*) или на побочной (*сумма4*) диагонали:

при $x = 1$ и $y = 1$:

```
сумма1 := поле[1, 1] +
        поле[1, 2] + поле[1, 3]
сумма2 := поле[1, 1] +
        поле[2, 1] + поле[3, 1]
сумма3 := поле[1, 1] +
        поле[2, 2] + поле[3, 3]
```

при $x = 1$ и $y = 3$:

```
сумма1 := поле[1, 1] +
        поле[1, 2] + поле[1, 3]
сумма2 := поле[1, 3] +
        поле[2, 3] + поле[3, 3]
сумма4 := поле[3, 1] +
        поле[2, 2] + поле[1, 3]
...
```

В результате для четырех указанных элементов знак “+” применяется 24 раза.

Для элемента, обозначенного **E**, проводится расчет четырех значений сумм:

при $x = 2$ и $y = 2$:

```
сумма1 := поле[2, 1] +
        поле[2, 2] + поле[2, 3]
сумма2 := поле[1, 2] +
        поле[2, 2] + поле[3, 2]
сумма3 := поле[1, 1] +
        поле[2, 2] + поле[3, 3]
сумма4 := поле[3, 1] + поле[2, 2] + поле[1, 3]
```

При этом используются 8 знаков “+”.

Задание № 1

Напишите фрагмент программы (на изучаемом языке программирования), в котором для подсчета необходимых сумм для элементов, обозначенных **B**, **D**, **F** и **H**, использовалось бы как можно меньшее число знаков “+”.

Задание № 2

Напишите фрагмент программы, в котором для подсчета необходимых сумм для элементов, обозначенных **A**, **C**, **G** и **I** (см. табличку), применялось бы как можно меньшее число знаков “+”.

Задание № 3

Напишите фрагмент программы, в котором для подсчета необходимых сумм для элемента **E** применялось бы как можно меньшее число знаков “+”.

Указания по выполнению. Используйте оператор цикла, операцию определения остатка и др.

Срок представления ответов — 10 октября. Можно выполнять не все задания.

Станьте «Учителем цифрового века»!

Получите именной сертификат
и индивидуальный код доступа к качественным цифровым
предметно-методическим материалам!



Индивидуальный код доступа, полученный в рамках проекта «Школа цифрового века», позволит вам адресно и бесплатно получать электронные журналы Издательского дома «Первое сентября» в любом объеме. Материалы изданий можно читать on-line, скачивать на компьютер, распечатывать, использовать на уроке.

Заявки на участие в проекте

принимаются с 15 сентября с.г.

на сайте <http://digital.1september.ru>

Заявку подает руководитель вашего образовательного учреждения или его заместитель.

Для школ, участвующих в проекте с 1 января по 30 июня 2012 года, оргвзнос – 2 тысячи рублей.

Коды доступа предоставляются бесплатно по количеству учителей, указанному в заявке.



Общероссийский проект «Школа цифрового века» по комплексному обеспечению школ адресной методической интернет-поддержкой разработан в соответствии с программой модернизации системы общего образования России. Интернет-сопровождение проекта – Издательский дом «ПЕРВОЕ СЕНТЯБРЯ».



Подробности на сайте <http://digital.1september.ru>