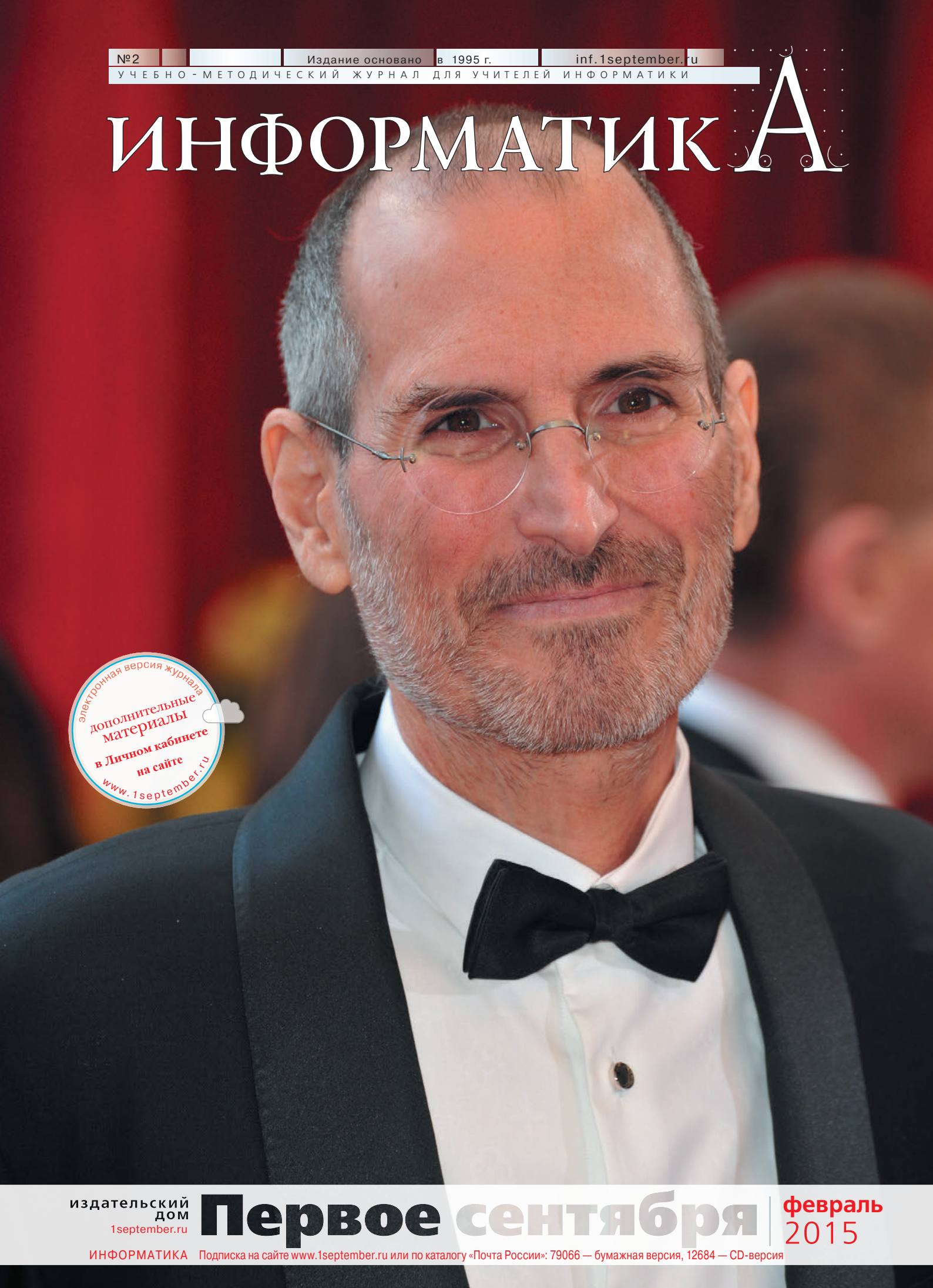
№2 Издание основано в 1995 г. inf.1september.ru
УЧЕБНО-МЕТОДИЧЕСКИЙ ЖУРНАЛ ДЛЯ УЧИТЕЛЕЙ ИНФОРМАТИКИ

ИНФОРМАТИКА



электронная версия журнала
дополнительные материалы в Личном кабинете на сайте www.1september.ru

издательский дом 1september.ru
Первое сентября
февраль 2015
ИНФОРМАТИКА Подписка на сайте www.1september.ru или по каталогу «Почта России»: 79066 — бумажная версия, 12684 — CD-версия

НА ОБЛОЖКЕ

Нет таких слов, которые не были бы сказаны о Стиве Джобсе. Написаны книги, сняты фильмы. Рыночная стоимость Apple и сейчас, спустя четыре года после его смерти, превышает стоимость всего российского фондового рынка (это не камень в огород российского рынка, это лишь демонстрация сравнительного масштаба). Безумно жаль, что он ушел так рано. Как интересно — что бы он еще придумал, чем бы удивил, порадовал, разочаровал, разозлил, довел до бешенства. В любом случае не оставил бы равнодушным. Безумно жаль, что этого не будет. А ему ведь было бы только 60...

Featureflash / Shutterstock.com

В НОМЕРЕ

В ЛИЧНОМ КАБИНЕТЕ

Облачные технологии от Издательского дома “Первое сентября”

Уважаемые подписчики бумажной версии журнала!

Дополнительные материалы к номеру и электронная версия журнала находятся в вашем Личном кабинете на сайте www.1september.ru.

Для доступа к материалам воспользуйтесь, пожалуйста, кодом доступа, вложенным в № 1/2015.

Срок действия кода: с 1 января по 30 июня 2015 года.

Для активации кода:

- зайдите на сайт www.1september.ru;
- откройте Личный кабинет (создайте, если у вас его еще нет);
- введите код доступа и выберите свое издание.

Справки: podpiska@1september.ru или через службу поддержки на портале “Первого сентября”.

ЭЛЕКТРОННЫЕ МАТЕРИАЛЫ

Презентации к статьям номера

ИНФОРМАТИКА

ПОДПИСНЫЕ ИНДЕКСЫ
по каталогу “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 16 000 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, конференции и техническое обеспечение Издательского дома: Павел Кузнецов
Производство: Станислав Савельев
Административно-хозяйственное обеспечение: Андрей Ушков
Педагогический университет: Валерия Арсланьян (ректор)

ЖУРНАЛЫ ИЗДАТЕЛЬСКОГО ДОМА “ПЕРВОЕ СЕНТЯБРЯ”
Английский язык – А.Громушкина
Библиотека в школе – О.Громова
Биология – Н.Иванова
География – и.о. А.Митрофанов
Дошкольное образование – Д.Тюттерин
Здоровье детей – Н.Сёмина
Информатика – С.Островский
Искусство – О.Волкова
История – А.Савельев
Классное руководство и воспитание школьников – М.Битянова
Литература – С.Волков
Математика – Л.Рослова
Начальная школа – М.Соловейчик
Немецкий язык – М.Бузоева
ОБЖ – А.Митрофанов
Русский язык – Л.Гончар
Спорт в школе – О.Леонтьева
Технология – А.Митрофанов
Управление школой – Е.Рачевский
Физика – Н.Козлова
Французский язык – Г.Чесновицкая
Химия – О.Блохина
Школа для родителей – Л.Печатникова
Школьный психолог – М.Чибисова

УЧРЕДИТЕЛЬ:
ООО “ИЗДАТЕЛЬСКИЙ ДОМ «ПЕРВОЕ СЕНТЯБРЯ»”
Зарегистрировано ПИ № ФС77-58447 от 25.06.2014 в Роскомнадзоре
Подписано в печать:
по графику 12.12.2014,
фактически 12.12.2014
Заказ №
Отпечатано в ОАО “Первая Образцовая типография”
Филиал “Чеховский Печатный Двор”
ул. Полиграфистов, д. 1,
Московская область,
г. Чехов, 142300
Сайт: www.chpd.ru
E-mail: sales@chpk.ru
Факс: 8 (495) 988-63-76

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

Немного о флибустьерах...

"Пиратское ПО считать трофейным"
(Шутка безымянного автора)

Не о тех, конечно, которые в старые времена с саблей и черной повязкой на глазу гонялись по южным морям за сундуками, полными пиастров ☺. А о современных "компьютерных пиратах", которые занимаются распространением нелицензионного ПО, а также аудио, видео и электронных книг.

Немного перефразируя всем известную фразу из Стругацких, "с пиратством мы боремся". Приняты соответствующие законы, создан реестр сайтов, заблокированных за наличие нелицензионного контента. Причем блокировка сайта может быть сделана и без соблюдения "презумпции невиновности" — по заявлению любого желающего о том, что на сайте якобы незаконно размещена его интеллектуальная собственность, сайт блокируется еще до суда в рамках "предварительных обеспечительных мер", а далее владельцу контента дается 15 дней на подачу иска о защите авторских прав. А, соответственно, владельцу сайта эти 15 дней даются на то, чтобы удалить нелицензионный контент либо… "доказывать, что он не верблюд": таким образом, новый закон может заодно стать отличным инструментом для нечестной конкуренции, — но это уже "издержки".

Недавно, в ноябре 2014 года, были утверждены поправки к закону, которые распространяют его действие на все виды электронного контента (видео, музыку, тексты, программное обеспечение и прочее, кроме фотографий). Причем если какой-то сайт будет уличен в "пиратстве" (и получит соответствующие предписания по решениям суда) дважды, то такой сайт будут блокировать навсегда.

К чему это приведет? К тому, что российские пользователи резко станут "сознательными" и перестанут пользоваться "пиратским" контентом? Вряд ли. Скорее всего резко вырастет популярность средств, позволяющих обходить блокировки сайтов на уровне DNS (например, браузера TOR), "пиратские" сайты начнут массово переезжать на зарубежные серверы (например, куда-нибудь в Эквадор, где долгое время "проживал" наиболее известный сайт "пиратских" электронных книг Либрусек — ныне легализованный), а большинство "пиратского" контента будет распространяться через пиринговые сети, которые вообще могут не иметь никаких серверов и обеспечивают прямой обмен файлами между пользователями — участниками сети. Либо (если удастся "зажать" и этот канал обмена "нелицензионкой") пользователи, как и в старые времена, будут переписывать файлы друг у друга вручную "по цепочке"…

А что делать, если новый закон действительно окажется неэффективен? Ответы очевидны:

- финансирование образования и науки и оплата труда работников этих сфер (как главных потребителей информационного контента) должно стать достаточным, чтобы они не задумывались над "экономией" при покупке легального контента;
- нужна длительная воспитательная работа для изменения "менталитета" наших сограждан, не считающих "собственностью" информацию (которую нельзя "пощупать", зато можно любое число раз копировать);
- сам лицензионный контент должен иметь действительно высокое качество, чтобы пользователи не жалели потраченных на него денег (тогда как сегодня содержательное качество многих фильмов и книг, да и программного обеспечения совершенно не соответствует запрашиваемым за него ценам).

Пока же можно посоветовать пользователям готовиться к переходу на *свободное ПО* на базе ОС Linux. По крайней мере это поможет решить проблему с "пиратским" программным обеспечением — не тратить уйму денег на продукцию Microsoft и других зарубежных фирм и притом не вступать в конфликт с новым "антипиратским" законом. Тем более что уже выпущен целый ряд неплохих руководств, "самоучителей" и практикумов по освоению свободного ПО (Linux, OpenOffice.org и др.), так что особых сложностей на этом пути не возникнет.

***Д.Ю. Усенков*,**
ст. н. с. Института информатизации РАО,
Москва

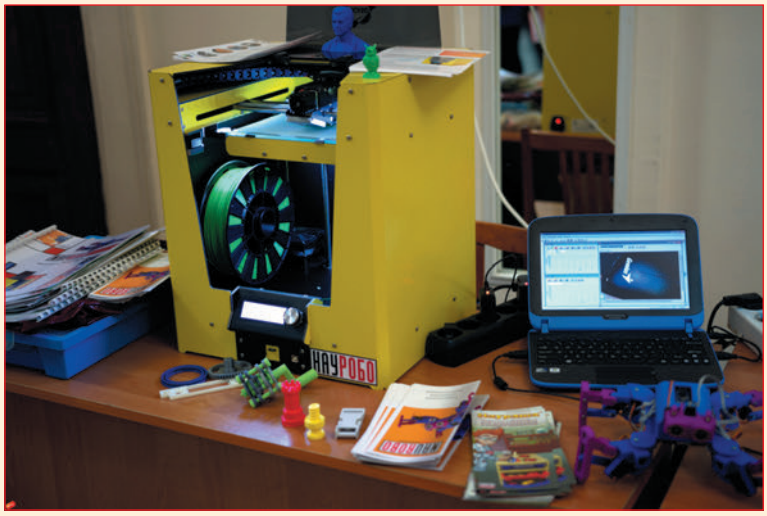
ВСЕРОССИЙСКАЯ НАУЧНО-ПРАКТИЧЕСКАЯ КОНФЕРЕНЦИЯ “Пропедевтика формирования инженерной культуры учащихся в условиях модернизации российского образования”

4–5 декабря 2014 года в г. Челябинске проходила Всероссийская научно-практическая конференция “Пропедевтика формирования инженерной культуры учащихся в условиях модернизации российского образования”.

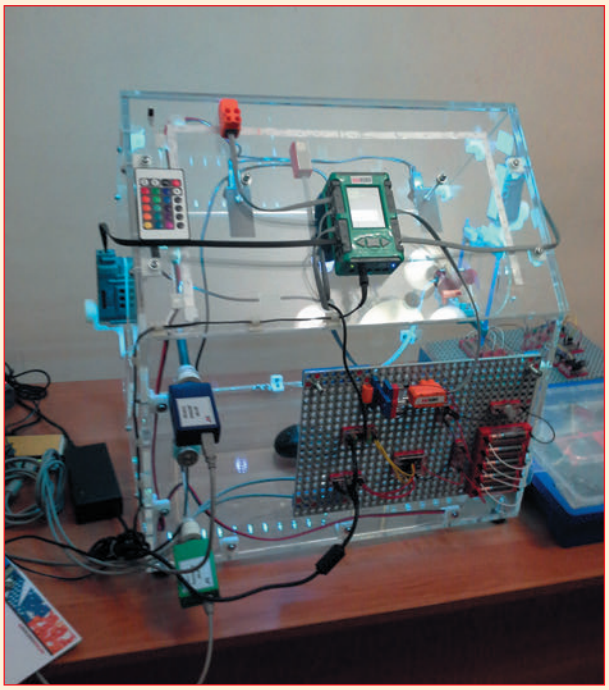
Авторы фото с конференции: **А.А. Елизаров, Н.Н. Самылкина**

На конференции обсуждались следующие актуальные проблемы:

- современное понимание инженерной культуры и ее связь с технологическим образованием;
- пропедевтика формирования инженерной культуры учащихся как актуального результата реализации ФГОС в организациях общего и дополнительного образования;
- дидактические возможности пропедевтики инженерной культуры учащихся средствами предметных областей “Математика и информатика”, “Технология”, “Физика” на всех ступенях обучения;
- использование робототехники как ведущего средства формирования у учащихся базовых представлений в сфере инженерной культуры;
- организация эффективного научного, информационного и методического сопровождения внедрения робототехники в школьное образование;
- пропедевтика формирования инженерной культуры учащихся в рамках неформального образования;
- профессиональная ориентация школьников на инженерно-технические специальности;
- подготовка и повышение квалификации учителей в области пропедевтики формирования инженерной культуры учащихся средствами учебного предмета и организации их развивающего досуга и др.



Престиж инженерного труда в нашей стране остается недостаточно высоким, несмотря на создаваемые государством условия. Инженерные кадры составляют основу промыш-



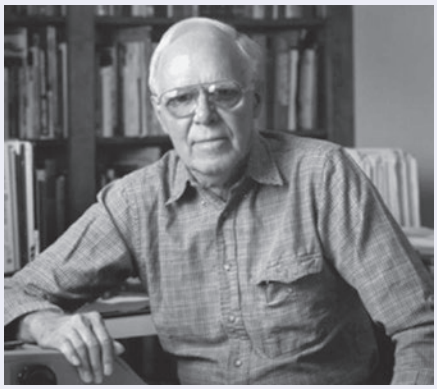
ленного развития страны, и их формирование должно начинаться в системе общего и дополнительного образования.

Понятие “инженерная культура” имеет интегративный, многофакторный характер.

Формирование инженерной культуры представляет собой процесс освоения личностью инженерных знаний, умений, ценностных ориентаций, позволяющих ей стать субъектом профессиональной культуры и включиться в процесс приумножения социально значимых ценностей, обеспечивающих индустриальное развитие страны на мировом рынке. В настоящее время линия образовательной робототехники является наиболее обеспеченной организационно, технически и методически, наиболее подготовленной к внедрению в школу и может служить точкой роста для формирования инженерной культуры вообще. По мере подготовки методического обеспечения иных образовательных линий (ТРИЗ, управление проектами, эргономика, дизайн и др.) эти линии должны быть включены в образовательный процесс.

Волшебные миры Мартина Гарднера

Д.Ю. Усенков,
науч. сотр. Института информатизации образования Российской академии образования, Москва



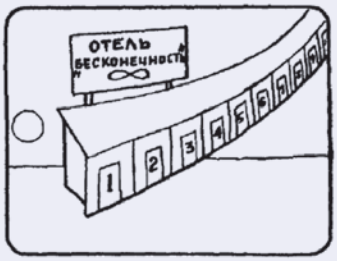
В октябре 2014 года исполнилось 100 лет со дня рождения **Мартина Гарднера (21.10.1914—22.05.2010)**. Тысячам читателей он хорошо известен как ученый-математик, а еще больше известен как популяризатор науки. Ведущий рубрики математических игр и развлечений журнала “Scientific American”, автор десятков статей и книг, в которых Гарднер умел простым, понятным даже для неподготовленных читателей языком изложить основы достаточно сложных разделов математики, теории множеств и логики. И не просто рассказать о них, но и увлечь за собой читателей, открыть перед ними целый волшебный мир, в основе которого, однако, нет ни капли мистики, а только лишь строгие логические построения, мир занимательных математических фокусов, познавательных комиксов, софизмов, игр и головоломок, увлекательных, интересных в познании, но отнюдь не сводящихся к праздной развлекательности. А также — автор нескольких фантастических рассказов и комментариев к творчеству Льюиса Кэрролла (“Алиса в стране чудес”, “Алиса в Зазеркалье”, “Охота на Снарка”) и серии рассказов Г.К. Честертона “Неведение отца Брауна”.

Конечно, охватить все то, о чем успел за свою жизнь написать и рассказать Мартин Гарднер, в одной статье невозможно. Поэтому мы познакомим читателей лишь с некоторыми занимательными задачами “в гарднеровском стиле” и пригласим самостоятельно продолжить знакомство с творчеством Гарднера по его книгам и по публикациям о нем в сети Интернет.

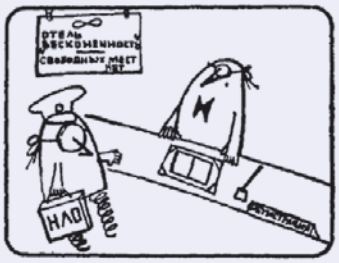
О бесконечности

Бесконечные множества — одна из интереснейших тем, затрагиваемых в математике. На первый взгляд — чистая абстракция, но посмотрим, как изящно и понятно даже для школьника рассказывает о них Гарднер в своей книге “А ну-ка, догадайся!”[1].

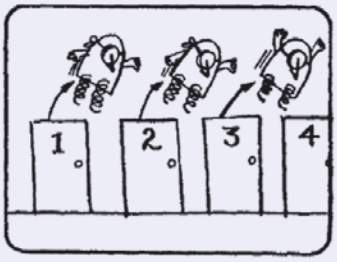
В самом центре нашей галактики находится огромная гостиница “Бесконечность”. В ней, действительно, бесконечно много однокомнатных номеров, уходящих через черную дыру в другое измерение. В гостинице есть первый номер, есть второй (комнаты перенумерованы по порядку), но нет последнего.



Однажды в гостиницу по пути в другую галактику заглянул командир НЛО.



Хотя ни одного свободного места не было, управляющий гостиницей все же нашел способ устроить пилота. Он попросил каждого обитателя гостиницы переселиться в комнату с номером на единицу больше, чем у той, в которой тот проживал прежде, и поселил командира НЛО в освободившийся первый номер.



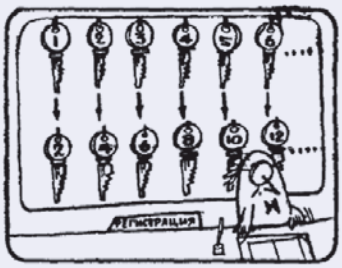
[1] Изд-во “Мир”, 1984 (пер. с англ.). Иллюстрации Джима Глена.

На следующий день в гостиницу прибыли пять супружеских пар, совершавших свадебное путешествие. Управляющий и тут не растерялся и, переселив каждого обитателя гостиницы в комнату с номером на пять больше, чем у той, в которой тот проживал прежде, отвел супружеским парам освободившиеся комнаты с номерами от 1 до 5.

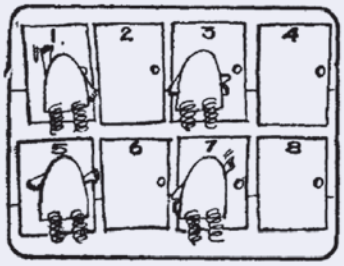
В конце недели в гостиницу нагрянули участники съезда продавцов жевательной резинки. Их было бесконечно много.

Можно понять, как управляющий гостиницы “Бесконечность” мог бы разместить любое *конечное* число вновь прибывших, но как разместить *бесконечное* множество гостей?

Управляющий легко справился и с этой задачей: каждого обитателя гостиницы он переселил в комнату с номером *вдвое* больше, чем у той, которую тот занимал прежде.



Все прежние постояльцы гостиницы оказались после переселения в комнатах с четными номерами, а бесконечное множество освободившихся комнат с нечетными номерами управляющий предоставил продавцам жевательной резинки.



Вот так, максимально наглядно, Гарднер иллюстрирует необычное свойство множеств с бесконечным числом элементов: нарушение интуитивно привычного нам правила “часть меньше целого”. А затем, в следующей главе, так же наглядно и понятно Гарднер рассказывает читателям о теории множеств, о таком понятии, как кардинальное число множества (количество его элементов) и об открытой Георгом Кантором иерархии бесконечностей. Читатель узнает о том, что и бесконечные множества имеют свои кардинальные числа, которые Кантор обозначил первой буквой древнееврейского алфавита — ℵ (“алеф”). И что существует четкая “иерархия бесконечностей”: хотя мы привыкли понимать бесконечность как одну и ту же во всех случаях некую “сверхбольшую” величину, одни бесконечные множества по размерам могут быть больше других бесконечных множеств. Например, бесконечное множество действительных чисел больше, чем бесконечное множество натуральных чисел. И вот, буквально играючи, мы имеем хотя бы минимальное представление о такой на первый взгляд сложной области математики, как теория множеств!

Односторонние поверхности

Возьмем любой плоский объект, — ну хотя бы обычный лист бумаги. Сколько у него поверхностей? Очевидно, две: верхняя и нижняя. На обеих можно, например, нарисовать разные, не зависящие друг от друга картинки, а муравей (не обладающий достаточной смелостью, чтобы переползти через тонкий край с одной стороны листа на другую) будет путешествовать по одной поверхности листа, даже не зная о существовании другой.

А может ли существовать объект, имеющий только одну поверхность?

Да, — это хорошо известная многим “лента Мёбиуса” (*рис.* 1).

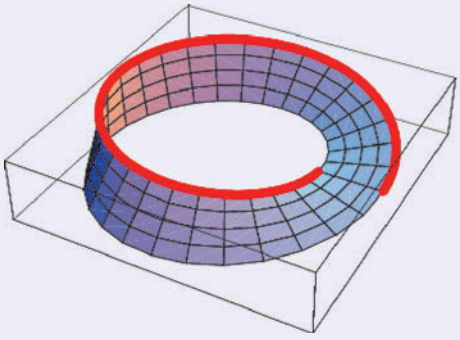


Рис. 1

Изготовить ленту Мёбиуса очень просто — достаточно склеить бумажную ленту в кольцо, предварительно перекрутив один из ее концов на пол-оборота. А вот свойства ленты Мёбиуса оказываются вовсе не так просты. В том, что она имеет только одну поверхность, нетрудно убедиться, попробовав прорисовать вдоль ленты Мёбиуса карандашом или фломастером линию: продолжая рисование, мы с удивлением убедимся, что придем вновь к началу этой линии, хотя нигде не будем “перепрыгивать” через край ленты. Но Гарднер демонстрирует нам и еще одно замечательное свойство: плоская фигура, движущаяся по поверхности ленты Мёбиуса (Гарднер даже придумал для этого “флатландцев” — неких существ, способных жить только в плоскости, тогда как мы живем в трехмерном пространстве), завершив путешествие по ленте Мёбиуса и вернувшись в исходную точку, окажется… зеркально отражена сама относительно себя! Увидеть этот эффект можно, например, на демонстрационном ролике (*рис.* 2), имеющемся на сайте *http://demonstrations.wolfram.com* (для просмотра таких роликов, правда, нужно установить специальный плагин Wolfram, сделать это предложит сам браузер при первой попытке любого ролика). Конечно, при этом речь идет об “идеальной” ленте Мёбиуса, имеющей нулевую толщину, а не о реальной бумажной ленте, но можно продемонстрировать этот эффект и на ленте из прозрачной пленки.

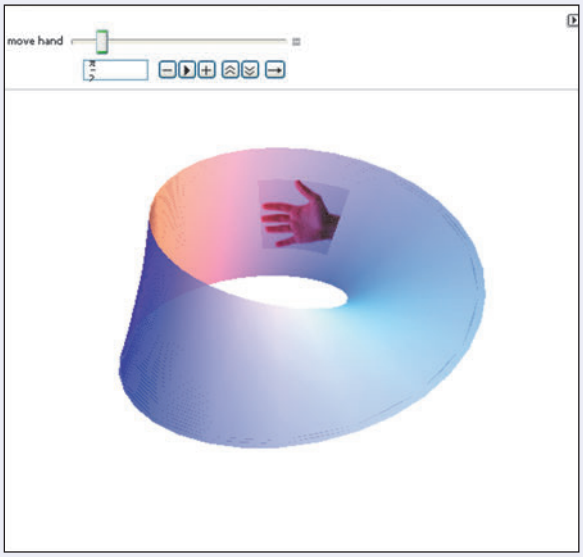


Рис. 2 (*http://demonstrations.wolfram.com/HandMovingOnAMoebiusStrip*)

А можно ли сделать *пространственный* объект, имеющий только одну поверхность и притом не имеющий краев? Да, — это объект, называемый “бутылкой Клейна”. Его можно увидеть (и даже построить самому!) при помощи еще одного интерактивного ролика на сайте *http://demonstrations.wolfram.com* (*рис.* 3). Рекомендуется выставить два верхних параметра (*draw* и *cutaway*) на максимум вправо, а потом поворачивать бутылку Клейна мышкой, чтобы осмотреть ее со всех сторон.

Вроде бы все просто и понятно, не правда ли? А между тем при помощи таких простых примеров Гарднер легко подводит нас к сложнейшим проблемам современной физики, к теориям и предположениям о структуре Вселенной, понятиям о кривизне пространства и антивеществе. И пусть это сделано не в виде строгих формул, с которыми привыкли иметь дело ученые, уже то, что мы понимаем, о чем идет речь, — весьма немало!

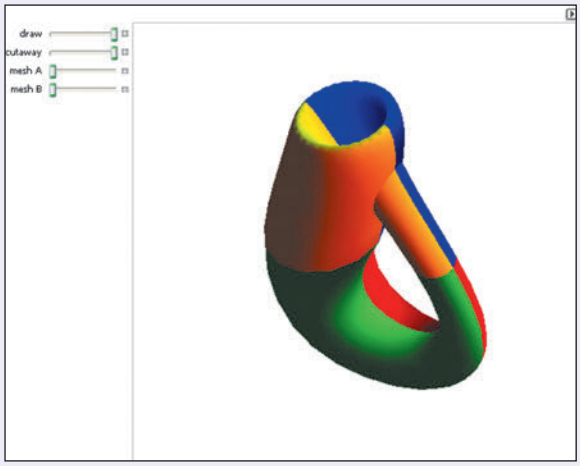


Рис. 3 (*http://demonstrations.wolfram.com/ColoredKleinBottle*)

Вся энциклопедия — одним штрихом!

А вот еще один парадокс, о котором рассказывает Гарднер все в той же книге “А ну-ка, догадайся”. Как вы думаете, можно ли закодировать огромный объем информации, например весь текст Большой Советской Энциклопедии, *одним-единственным* значком — риской на металлическом или пластиковом стержне? Скажете, — не может быть? А Гарднер убедительно доказывает, что это возможно, по крайней мере теоретически. И вот как.

Любой пользователь компьютера сегодня знает, как в нем кодируется текст (в книге же, написанной в 1984 году, Гарднеру приходится более подробно это объяснять для читателей). Поэтому можно преобразовать любой текст любой длины в последовательность чисел (в наших компьютерах — двоичных, а Гарднер брал для примера десятичные коды символов). Например, если английские буквы и прочие знаки пронумеровать числами от 001 до 999, то слово “CAT” можно закодировать строкой цифр: 003001020.

Возьмем энциклопедию и строка за строкой закодируем ее в виде огромного, поистине гигантского числа, состоящего из записанных друг за другом кодов символов текста. А теперь допишем в начале этого числа ноль и десятичную запятую — наше число превратится в длинную (но конечную!) десятичную дробь.

А теперь (теоретически, конечно) можно взять металлический стержень и нанести на него один-единственный знак — риску, которая разделит стержень на две части длиной a и b, так что отношение a/b равно полученной нами дроби.

Вот и все! Причем такое кодирование обратимо: всегда можно измерить с требуемой точностью длины стержня до и после риски, вычислить отношение этих величин, декодировать полученную последовательность цифр и получить на выходе исходный текст!

Конечно, такое кодирование доступно лишь умозрительно, на практике невозможно обеспечить необходимую точность ни при нанесении риски, ни при измерении отрезков стержня. Зато на этом наглядном примере мы получили представление не только о теории кодирования информации, но и сделали первый шаг к теории доказательств и теореме Гёделя, доказательство которой как раз основано на использовании подобного числового кода. А затем Гарднер показывает нам и вовсе феноменальный (но тем не менее строго доказуемый!) вывод: любое иррациональное число — например, запись числа π, если удалось бы получить ее с очень большой точностью (большим числом знаков после запятой), может содержать отрезки, которые при их декодировании различными способами совпадают с текстами принципиально любых существующих текстов, когда-либо, где-либо и кем-либо написанных, и даже текстов, которые еще только будут написаны в будущем!

* * *

Это — лишь три примера занимательнейших задач, с которыми Мартин Гарднер знакомил своих читателей. Но они далеко не единственные.

Гарднер писал и о различных логических играх — таких, как игра “Текс” (Hex), изобретенная Питом Хейном в 1942 году (игра в шашки на ромбовидной доске с клетками в виде шестиугольников — *http://mathworld.wolfram.com/GameofHex.html*), “Икосиан” (поиск гамильтоновых циклов на ребрах додекаэдра — *рис.* 4), полимино — составление фигур из набора заготовок определенной формы (*рис.* 5) и даже “Крестики-нолики” (*рис.* 6) — в английском языке она называется “Tic-Tac-Toe”.

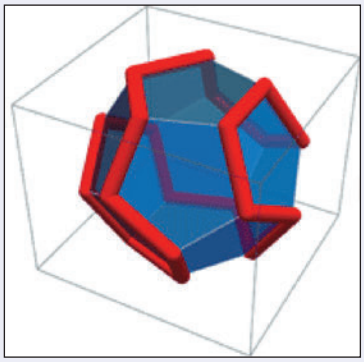


Рис. 4 (*http://mathworld.wolfram.com/IcosianGame.html*)

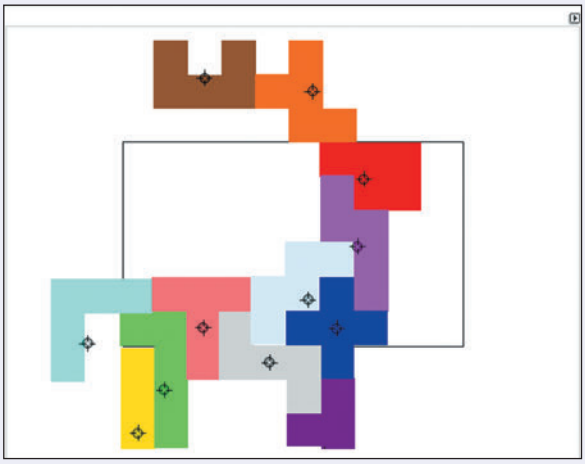


Рис. 5
(*http://demonstrations.wolfram.com/TilingWithPentominos*)

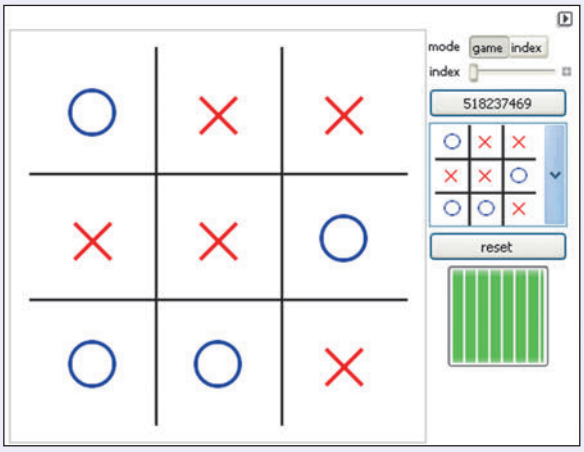


Рис. 6 (*http://demonstrations.wolfram.com/TicTacToe*)

Писал Гарднер и о лабиринтах, кубиках Сома (пространственный аналог полимино), математической логике и “магических квадратах” (*рис.* 7, 8).

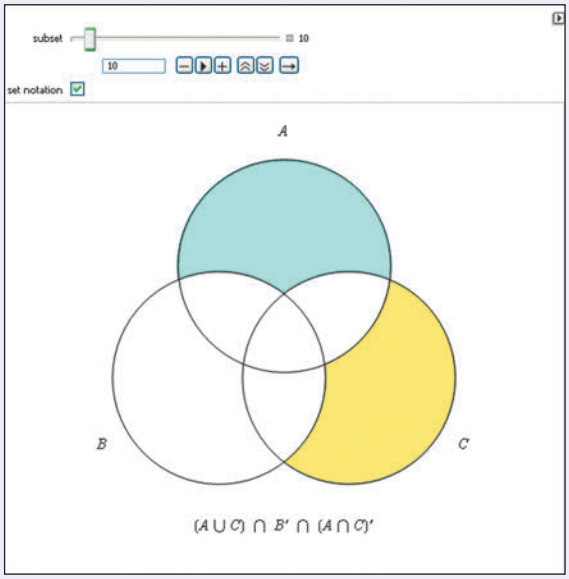


Рис. 7. Диаграммы Венна
(*http://demonstrations.wolfram.com/VennDiagrams*)

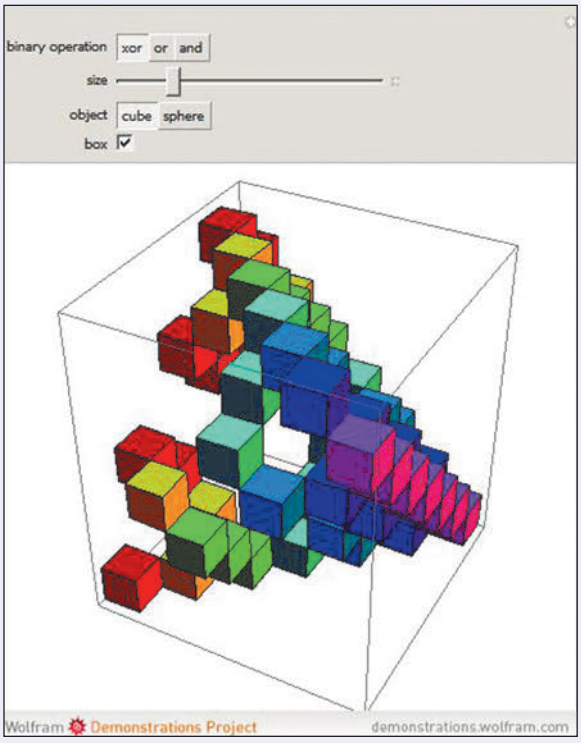


Рис. 8. Трехмерная визуализация логических операций
(*http://demonstrations.wolfram.com/3DViewOfBinaryLogicalOperations*)

В работах Гарднера рассмотрены и такие темы, как “квадрирование квадрата” (разбиение большого квадрата на ряд маленьких, площади которых соответствуют квадратам ряда последовательных натуральных чисел) и “золотое сечение” — пропорция, лежащая в основе человеческого восприятия прекрасного.

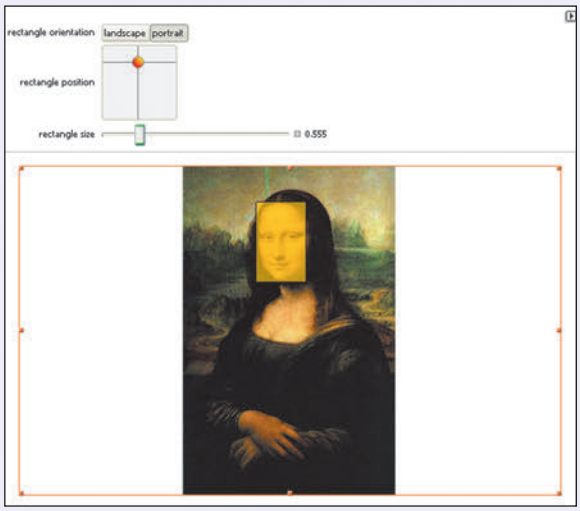


Рис. 9. Предлагается найти соотношения “золотого сечения” в картине Леонардо да Винчи “Мона Лиза”
(*http://demonstrations.wolfram.com/MonaLisaAndTheGoldenRectangle*)

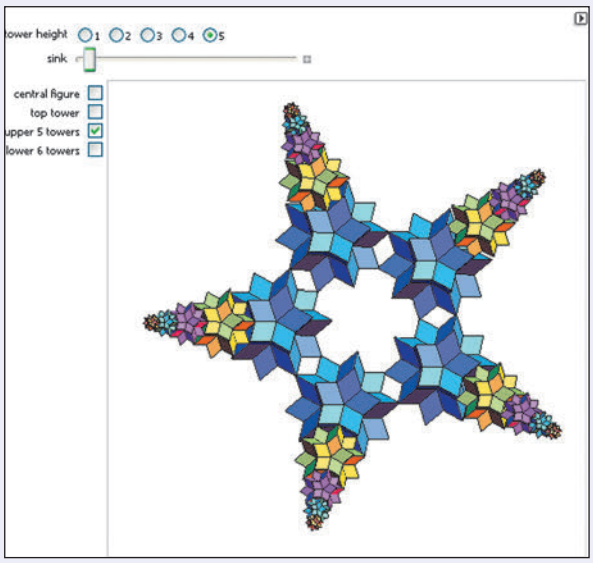


Рис. 10. Пространственная конструкция, построенная с использованием соотношений “золотого сечения” (*http://demonstrations.wolfram.com/RhombicHexecontahedronTower*)

Гарднер нашел решение задачи о пауке и мухе (как пауку найти кратчайший путь к мухе, сидящей на противоположной стене комнаты, — *рис.* 11), писал о задачах упаковки соприкасающихся кругов внутри различных геометрических фигур (прямоугольника, треугольника, круга — *рис.* 12), о свойствах эллипса, конических сечений (*рис.* 13) и числа π (*рис.* 14).

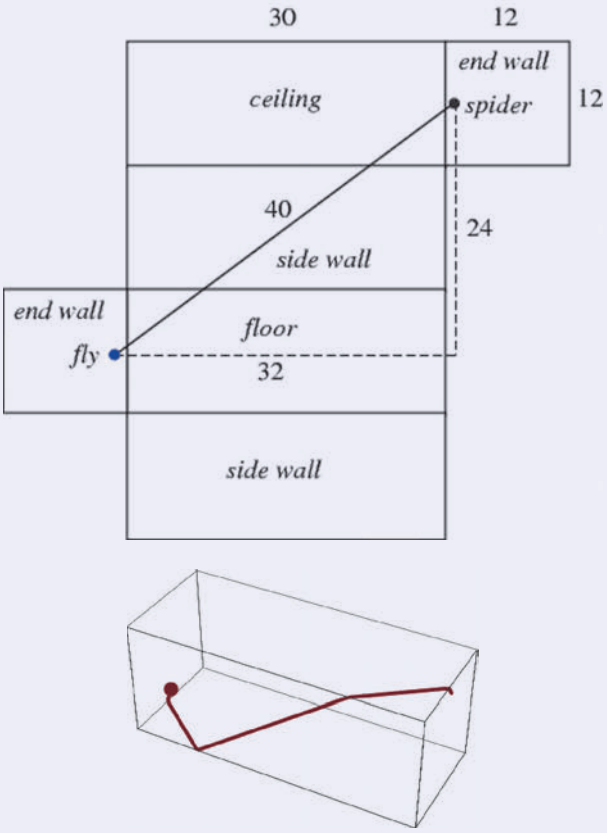


Рис. 11. Решение задачи о пауке и мухе (*http://mathworld.wolfram.com/SpiderandFlyProblem.html*)

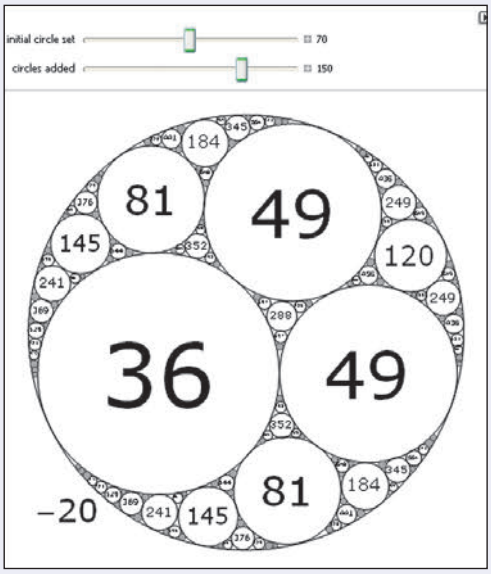


Рис. 12. Задача об упаковке кругов (*http://demonstrations.wolfram.com/TheCirclesOfDescartes*)

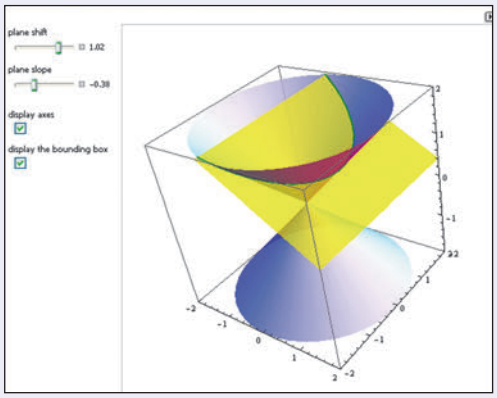


Рис. 13. Интерактивное построение конических сечений (*http://demonstrations.wolfram.com/PlaneCrossSectionsOfTheSurfaceOfACone*)

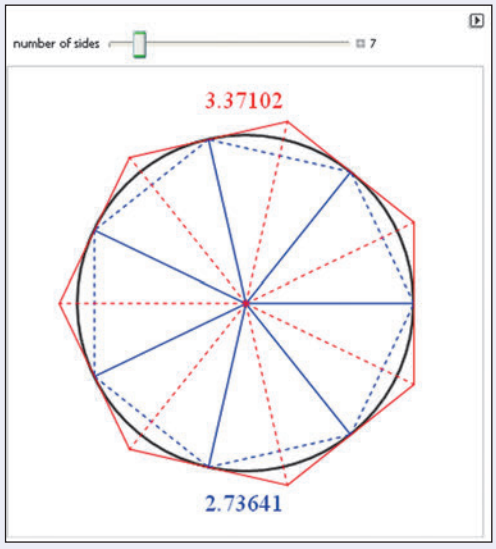


Рис. 14. Вычисление числа π методом Архимеда (последовательными приближениями при увеличении количества сторон описанного многоугольника) (*http://demonstrations.wolfram.com/ArchimedesApproximationOfPi*)

В сферу внимания Гарднера попали и задачи из теории групп, об интервалах между простыми числами, о латинских квадратах и теореме о четырех красках (можно ли раскрасить набор соприкасающихся областей четырьмя различными цветами так, чтобы не было ни одной пары соприкасающихся областей одного цвета). Последняя задача получила дальнейшее развитие и в работах других математиков, например, в работе Хивуда доказано, что для раскраски по указанному правилу карты областей на поверхности тора достаточно семи цветов (*рис.* 15).

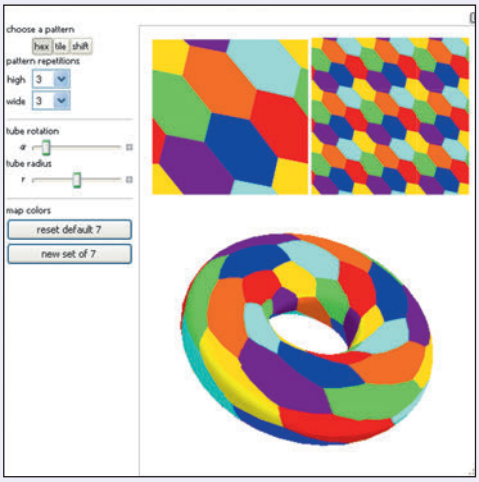


Рис. 15. Семь цветов для закрашивания областей на торе (*http://demonstrations.wolfram.com/MapColoringOnATorus*)

Гарднер писал и о решении логической задачи о восьми ферзях, о кривых постоянной ширины (таких, как треугольник Рёло, позволяющий с помощью круглой фрезы получить… практически квадратное отверстие, — *рис.* 16), о машине Тьюринга и о “гиперкубе” — попытке визуализации четырехмерной геометрической фигуры (*рис.* 17).

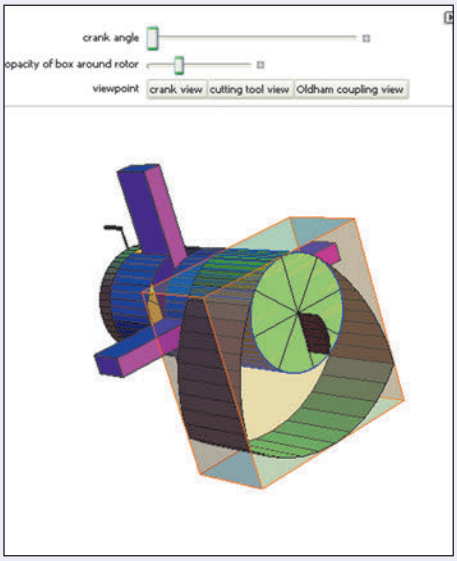


Рис. 16. Сверление квадратного отверстия с использованием треугольника Рёло (*http://demonstrations.wolfram.com/SquareHoleDrillInThreeDimensions*)

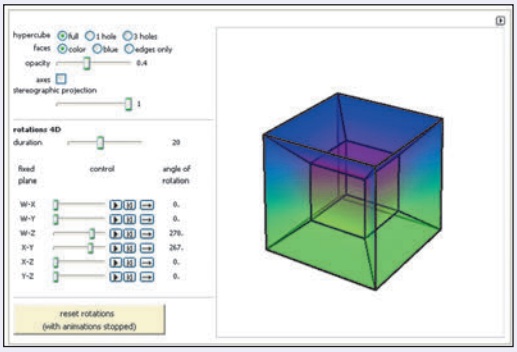


Рис. 17. “Гиперкуб” — попытка визуализации фигуры, построенной в пространстве четырех измерений (*http://demonstrations.wolfram.com/RotatingAHypercubeIn4D*)

Гарднер рассказывал своим читателям о “клеточных автоматах”, реализованных в логической игре “Жизнь” (“Life”), придуманной Джоном Конвеем и демонстрирующей динамику развития биологических популяций (*http://mathworld.wolfram.com/GameofLife.html*), и о дешифровке радиопослания, отправленного землянами в звездное скопление созвездия Геркулеса при помощи радиотелескопа в Аресибо, которое будет лететь со скоростью света к своим адресатам целых 25 000 световых лет (*рис.* 18).

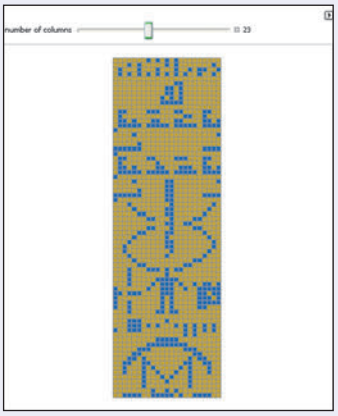


Рис. 18. Интерактивная расшифровка радиопослания из Аресибо (*http://demonstrations.wolfram.com/DecodingTheAreciboMessage*)

И еще множество других тем — широта кругозора Мартина Гарднера была поистине огромной, — так же, как и у русского и советского физика, математика, астронома и популяризатора науки — Якова Исидоровича Перельмана. Впрочем, творчество Я.И. Перельмана заслуживает отдельного разговора и отдельной статьи. А тем же из читателей, кто захочет более подробно ознакомиться с творчеством Гарднера и с теми областями науки, с которыми он старался познакомить своих читателей, можно порекомендовать обратиться к книгам Гарднера (и, поверьте, вы не пожалеете о потраченном на них времени!) и к посвященной 100-летию со дня рождения Мартина Гарднера web-странице *http://habrahabr.ru/post/244563* (и ее англоязычному прототипу *http://blog.wolfram.com/2014/10/21/martin-gardners-100th-birthday*), на основе которых была подготовлена данная статья.

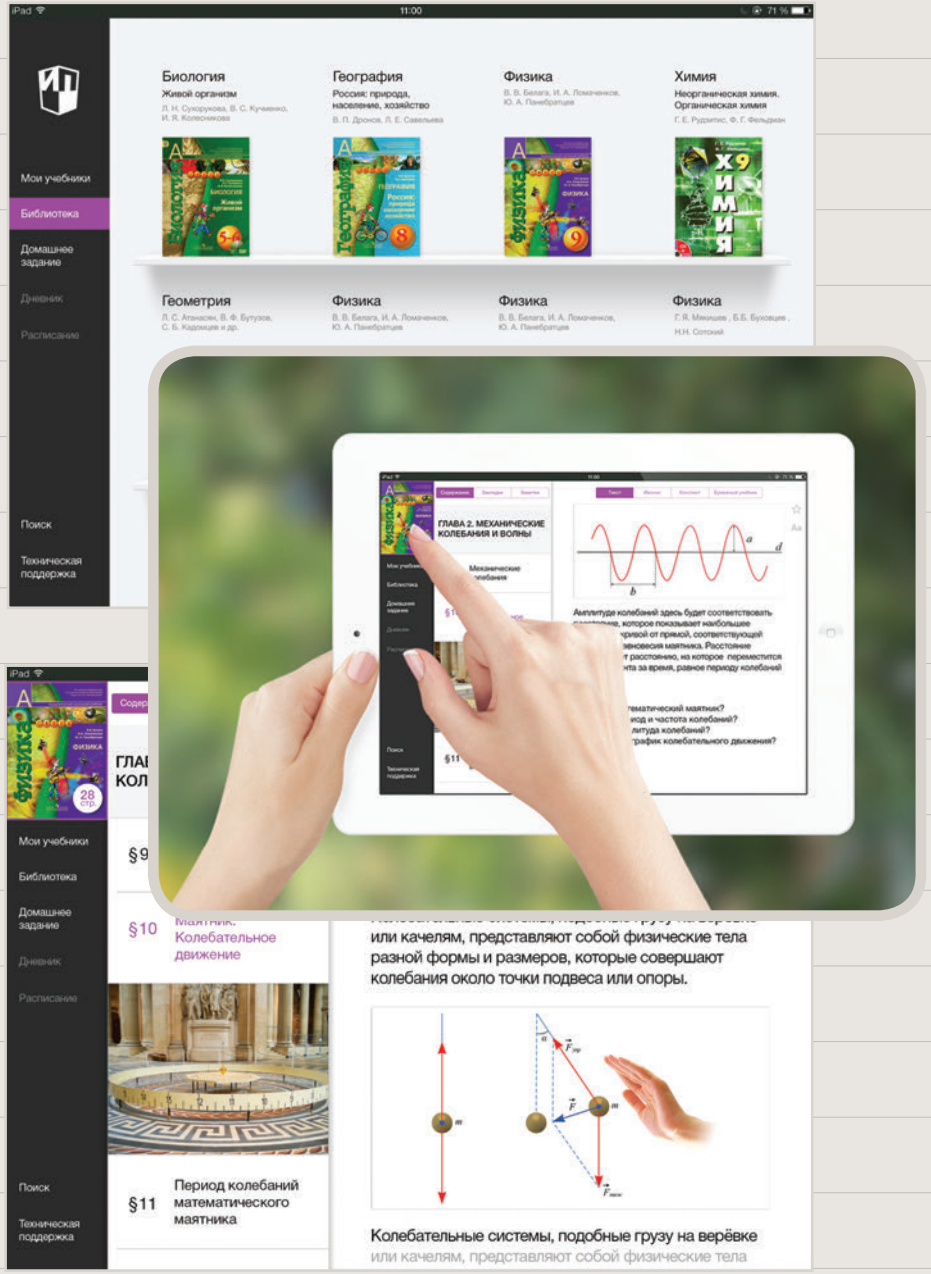
ПРОСВЕЩЕНИЕ
ИЗДАТЕЛЬСТВО

Легко учить, легко учиться!

ЭЛЕКТРОННЫЙ УЧЕБНИК

для учителя и ученика

К 1 сентября 2015 года издательство подготовит электронные учебники по всем предметам школьной программы. Электронные учебники будут доступны для апробации педагогам, образовательные организации которых участвуют в проекте «Школа цифрового века»



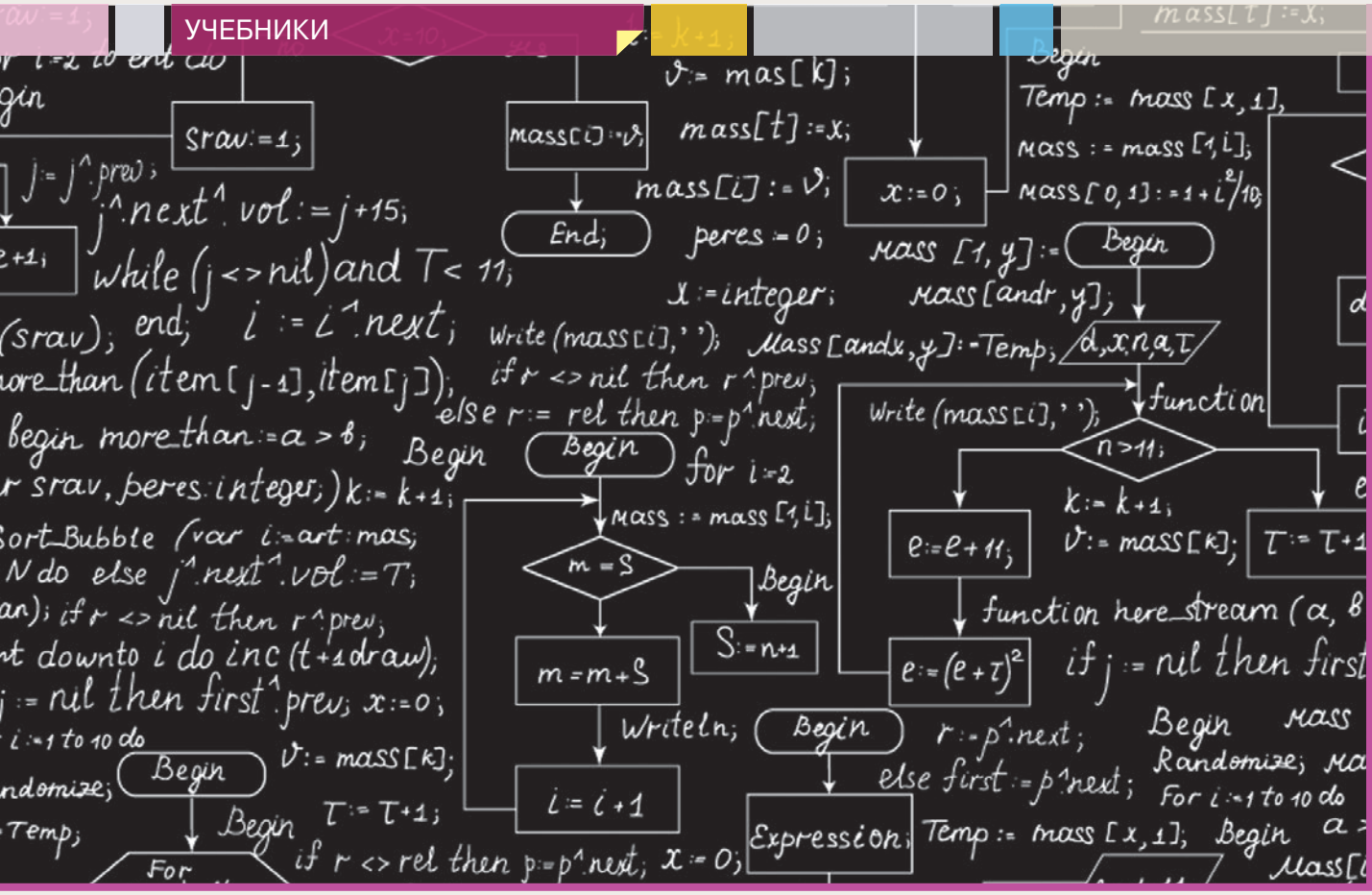
Технологические особенности:

- ✓ Работает на планшетах iOS, Android, Windows
- ✓ Многослойный (тексты, словари, мультимедийные объекты и др.)
- ✓ «Легкий» (30 секунд для скачивания и установки)

Особенности контента:

- ✓ Разные формы представления содержания параграфов
- ✓ Большая база интерактивных объектов и сервисов, в том числе для интерактивной доски
- ✓ Инструменты и сервисы для удобства работы учеников с ограниченными возможностями здоровья

www.prosv.ru



Алгоритмизация и программирование

К.Ю. Поляков,
д. т. н., Санкт-Петербург,
http://kpolyakov.spb.ru,
Е.А. Еремин,
к. ф.-м. н., г. Пермь

Алгоритмы и исполнители

Ключевые слова:
- алгоритм
- исполнитель
- система команд исполнителя
- дискретность
- понятность
- определенность
- конечность
- корректность

Что такое алгоритм?

Знаете ли вы, как сложить два многоразрядных числа? Наверняка знаете! В начальной школе вам рассказали, какие действия и в каком порядке нужно выполнить для того, чтобы решить эту задачу для любых чисел. Такую последовательность действий в информатике называют *алгоритмом*.

Это слово происходит от имени средневекового арабского ученого Мухаммада аль-Хорезми, который в IX веке описал правила вычислений с десятичными числами. Работы аль-Хорезми были переведены на латинский язык и стали известны в Европе. Через некоторое время слово “алгоритм” (от имени автора, которое по-латыни писали как *Algorizmi* или *Algorizmus*) стало обозначать любую систему вычислений по определенным правилам.

Мухаммад аль-Хорезми (IX век н.э.)

С алгоритмами мы постоянно встречаемся в жизни: все пошаговые инструкции, позволяющие, например, внести сумму денег на счет мобильного телефона или заказать билет на самолет через Интернет, — это алгоритмы.

С любезного разрешения авторов продолжаем публикацию глав из будущего (надеемся!) учебника.

Алгоритм — это точное описание порядка действий, которые должен выполнить исполнитель для решения задачи.

В этом определении встретилось новое слово — “исполнитель”, то есть тот, кто выполняет алгоритм. Давайте разберемся, кто это такой.

Кто такой исполнитель?

Сможет ли маленький ребенок сходить в магазин за хлебом? Сможет ли собака оплатить счет за квартиру? Конечно, нет. Поэтому можно сделать вывод: любой алгоритм составляется для какого-то определенного исполнителя.

Исполнитель — это человек, животное или машина, которые могут понимать и выполнять некоторые команды.

Слова “некоторые команды” говорят о том, что исполнитель понимает только определенный набор команд. Компьютер, мобильный телефон и даже современная стиральная машина — это тоже исполнители со своим набором команд.

Система команд исполнителя (СКИ) — это набор команд, который понимает и умеет выполнять исполнитель.

Например, СКИ простейшего исполнителя Робот, который умеет только передвигаться по ячейкам клетчатой доски, содержит всего четыре команды: вверх, вправо, вниз и влево:

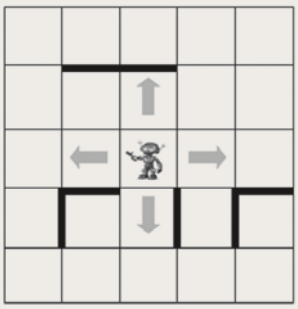


Рис. 1

Других команд этот Робот не понимает и выполнить не может.

На поле Робота могут встречаться стенки, которые обозначены жирными линиями. Робот не может ходить через стенки. Например, при попытке выполнить два раза подряд команду **вверх** Робот столкнется со стенкой и разрушится.

Любой исполнитель работает в некотором окружении (*среде*). Например, среда исполнителя Робот, с которым мы только что познакомились, — клетчатое поле со стенками.

Исполнители могут находиться в разных *состояниях*, и в зависимости от этого могут (или не могут) выполнять разные команды. Например, незаряженное ружье не выстрелит, а лежащий человек не сможет прыгнуть, не изменив свое состояние. Грузовик с пустым кузовом не сможет засыпать яму песком.

Свойства алгоритма

Дискретность — алгоритм состоит из отдельных команд (шагов), каждая из которых выполняется ограниченное время. Никакая команда не может быть выполнена раньше, чем закончится предыдущая.

Понятность — алгоритм содержит только команды, входящие в систему команд исполнителя, для которого он предназначен.

Определенность — при каждом выполнении алгоритма с одними и теми же исходными данными должен быть получен один и тот же результат. Это значит, что исполнитель должен однозначно понимать команду и последовательность выполнения команд и выполнять их каждый раз одинаково.

Конечность (результативность) — любой алгоритм должен заканчиваться за конечное число шагов с некоторым результатом. Результатом может быть и сообщение о том, что задача не имеет решений.

Корректность — для любых допустимых исходных данных алгоритм должен приводить к правильному результату.

Массовость — алгоритм, как правило, предназначен для решения множества однотипных задач с различными исходными данными. Поэтому для составления алгоритма решение задачи нужно написать “в буквах”, вводя имена для исходных данных.

Эти свойства не равноправны. Дискретность, понятность и определенность — фундаментальные свойства алгоритма, то есть ими обладают все алгоритмы. Остальные свойства — это требования к “правильному” алгоритму.

Если работа алгоритма никогда не заканчивается, то говорят, что алгоритм *зациклился*. В этом случае результат его работы не определен.

Как правило, любую задачу можно решить с помощью различных алгоритмов. Например, найти сумму первых 100 натуральных чисел можно последовательным сложением:

$$1 + 2 + 3 + 4 + \ldots + 99 + 100 = 5050$$

Но когда эту задачу задали ученикам в одной немецкой школе, один из них сообразил, что если разбить все числа на пары (симметрично с противоположных концов), то все суммы

$$1 + 100, 2 + 99, \ldots 50 + 51$$

равны 101, поэтому общий результат равен $50 \times 101 = 5050$. Этим сообразительным мальчиком (согласно легенде) был великий математик Карл Фридрих Гаусс.

Какой же алгоритм выбрать? Конечно же тот, который лучше. Но что значит лучше? Обычно алгоритмы сравнивают по времени выполнения. При этом лучшим считается тот алгоритм, который быстрее приводит к правильному результату (требует меньше действий исполнителя). Кроме того, есть и другие критерии, например, количество памяти, которое требуется для работы алгоритма. Поэтому

выбор алгоритма зависит от особенностей задачи, которую вы решаете.

Контрольные вопросы

1. Откуда произошло слово “алгоритм”?
2. Что такое алгоритм?
3. Сформулируйте алгоритмы

а) сложения двух однозначных чисел;

б) вычитания однозначного числа из двузначного;

в) умножения двух двузначных чисел;

г) вычисления остатка от деления двух целых чисел;

д) вычисления среднего арифметического двух чисел.

4. Сформулируйте алгоритмы

а) заварки чая (как это делаете вы);

б) перехода через улицу по пешеходному переходу со светофором;

в) покупки бананов в магазине;

г) заправки автомобиля топливом;

д) оплаты мобильной связи через терминал.

5. Кого (что) называют исполнителем алгоритма?
6. Верно ли, что любой алгоритм составляют для какого-то определенного исполнителя? Докажите свою точку зрения.
7. Как вы думаете, можно ли считать алгоритмом рецепт приготовления блюда? Обоснуйте свою точку зрения.
8. Что такое “система команд исполнителя”?
9. Какие свойства алгоритма вы знаете? Какие из них обязательны?
10. Приведите примеры алгоритмов, работающих бесконечно.
11. Приведите примеры некорректных алгоритмов.
12. Что значит фраза “алгоритм зациклился”?
13. Как сравнивают различные алгоритмы решения одной и той же задачи?

Тема для сообщения:

“Аль-Хорезми и его вклад в науку”

Формальные исполнители

Ключевые слова:

- неформальный исполнитель
- формальный исполнитель
- непосредственное управление
- программное управление
- автоматизация
- Робот
- Черепаха
- Шифровальщик

Какие бывают исполнители?

Человек как исполнитель отличается от компьютера в первую очередь тем, что он может поразному выполнить одну и ту же команду или даже вообще отказаться ее выполнять. Например, пункт кулинарного рецепта “добавить соль по вкусу” каждый повар выполнит по-разному. Такой исполнитель называется *неформальным*, он может обдумывать команды и вносить в ход их выполнения свои поправки.

Во многих случаях такое поведение исполнителя недопустимо. Например, компьютер по команде сложения должен всегда выполнять именно сложение, а не вычитание или умножение.

> **Формальный исполнитель** — это исполнитель, который одну и ту же команду всегда выполняет одинаково.

Все машины, в том числе компьютеры и роботы, — это формальные исполнители. Иногда человеку тоже приходится быть формальным исполнителем, строго выполняя инструкции, например, во время чрезвычайных ситуаций или на военной службе.

Исполнители могут работать в двух режимах — в режиме управления с пульта (его называют также режимом *непосредственного управления*) и в программном режиме.

Режим *управления с пульта* означает, что в течение всего времени выполнения алгоритма кто-то управляет исполнителем, отдавая одну команду за другой. Исполнитель тут же выполняет каждую введенную команду. Так человек обычно управляет телевизором.

Для нас более интересно *программное управление*, когда в память исполнителя заранее записывается (на специальном языке) весь алгоритм, который ему нужно выполнить. После запуска этого алгоритма исполнитель выполняет все действия в автоматическом режиме. Это позволяет использовать *роботов* для автоматизации многих работ, которые раньше выполнял человек.

В информатике рассматривают только формальных исполнителей, поэтому дальше для краткости мы будем использовать термин “исполнитель” в значении “формальный исполнитель”.

Теперь рассмотрим для примера несколько исполнителей.

Исполнитель Робот

С исполнителем Робот мы уже познакомились. Обозначим команды, которые он понимает, числами:

1. **вверх**
2. **вправо**
3. **вниз**
4. **влево**

Алгоритм для Робота можно записать как последовательность номеров команд. Например, запись 23321 означает, что сначала Робот должен выполнить команду 2 (вправо), затем дважды команду 3 (вниз), потом еще один раз команду 2 и в конце команду 1 (вверх). Выполняя этот алгоритм, Робот пройдет из начального положения в клетку, отмеченную буквой А (*рис.* 2), не столкнувшись со стенками.

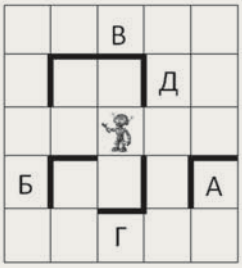


Рис. 2

Исполнитель Черепаха

Исполнитель Черепаха работает на плоскости, оставляя след при движении. Ее СКИ содержит такие команды:

- **вперед n** — передвинуться вперед на *n* шагов (здесь и дальше *n* — целое число);
- **назад n** — передвинуться назад на *n* шагов;
- **влево n** — повернуться на *n* градусов влево (против часовой стрелки);
- **вправо n** — повернуться на *n* градусов вправо (по часовой стрелке);
- **повтори n [...]** — повторить *n* раз все команды, записанные в квадратных скобках.

Черепаха поворачивается относительно своего *текущего направления* (то есть того направления, куда она смотрит в данный момент).

Например, алгоритм для Черепахи

```
повтори 4 [вперед 20 вправо 90]
```

заставляет ее нарисовать квадрат со стороной 20 шагов и вернуться в начальную точку, откуда она начала рисовать:

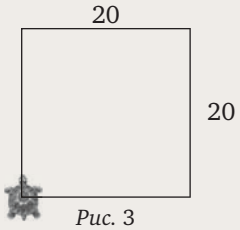


Рис. 3

Заметьте, что после выполнения алгоритма Черепаха вернется в начальное положение.

Исполнитель Удвоитель

Исполнитель Удвоитель работает с числами. Его СКИ состоит всего из двух команд:

```
1. прибавь 1
2. умножь на 2
```

Программа для Удвоителя — это последовательность номеров команд, записанная без пробелов. Например, программа 12211 означает, что сначала нужно выполнить команду 1, затем — дважды команду 2 и потом дважды команду 1.

Например, применим программу 12211 к начальному числу 2. На *рис.* 4 показаны все промежуточные результаты, которые исполнитель получает после каждого шага.

Над сплошными стрелками записаны номера выполняемых команд. В итоге получается число 14.

Программа:

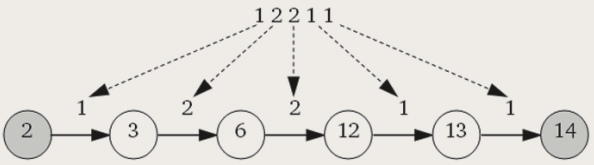


Рис. 4

Исполнитель Шифровальщик

Исполнитель Шифровальщик работает с цепочкой символов (символьной строкой), составленной из заглавных русских букв. Каждую букву строки он заменяет на следующую букву в алфавите[1]: А на Б, Б на В и т.д. Последнюю букву алфавита (Я) Шифровальщик заменяет на первую (А).

Например, после обработки строки ПРИВЕТ ВАСЯ получается РСКГЁУ ГБТА. Это шифровка, которую не сможет прочитать тот, кто не знает, как обработано сообщение.

Как построить исполнителя?

Давайте разберемся, что нужно для того, чтобы построить нового исполнителя.

В первую очередь необходимо определить его *среду* — обстановку, в которой он работает. Например, для Робота среда — это клетчатое поле, для Черепахи — плоскость.

Во-вторых, нужно задать набор команд исполнителя (СКИ) и определить, как именно он должен выполнять каждую из этих команд. Необходимо обдумать, в каких случаях исполнитель не сможет выполнить команду (например, Робот не сможет пройти сквозь стенку).

Контрольные вопросы

1. В каких случаях человек может отказаться выполнять команды? Когда он должен быть формальным исполнителем?
2. В каких случаях формальное выполнение алгоритма — это недостаток? Приведите примеры.
3. Чем отличаются режим управления с пульта и программное управление? Подумайте, в каких случаях лучше работает каждый из методов.

Задачи

1. Составьте две различные программы для Робота, которые переводят его из начальной клетки (*рис.* 2) в клетку Б. Робот не может проходить через стенки.
2. Выполните предыдущее задание для клеток В и Г.
3. *Сколько существует различных программ, которые переводят Робота из начальной клетки (*рис.* 2) в клетку Д?
4. *Может ли Робот перейти из начального положения в клетку Д за семь команд? Ответ обоснуйте.

[1] Шифр, который применяет Шифровальщик, называется шифром Цезаря.

5. Какую фигуру нарисует Черепаха, выполнив такие алгоритмы:

а) `повтори 4 [вперед 40 вправо 120]`
б) `повтори 4 [вперед 50 вправо 45]`
в) `повтори 4 [вперед 30 вправо 180]`
г) `повтори 6 [вперед 40 вправо 72]`
д) `повтори 10 [вперед 30 вправо 45]`
е) `повтори 10 [вперед 20 вправо 90]`

6. *Подумайте, как Черепаха может нарисовать окружность.

7. Какой результат получит Удвоитель, выполнив программу 211221 для начального числа

а) 2; г) 5; ж) 8;
б) 3; д) 6; з) 9;
в) 4; е) 7; и) 10?

8. Платон хочет с помощью исполнителя Удвоитель получить из числа 2 число 9. Сколько есть способов решения этой задачи?

9. *Дана программа для Удвоителя: 2112. Составьте программу из трех команд, которая равносильна заданной, то есть приводит к точно такому же результату для любого начального числа.

10. У исполнителя Калькулятор две команды:

```
1. прибавь 2
2. умножь на 2
```

Какое число получит этот исполнитель, если выполнит программу 122112 для начальных чисел:

а) 1; г) 4; ж) 7;
б) 2; д) 5; з) 8;
в) 3; е) 6; и) 9?

11. Платон хочет с помощью исполнителя Калькулятор из предыдущего задания получить из числа 2 число 14. Сколько есть способов решения этой задачи? Какие числа невозможно получить из числа 2 с помощью этого исполнителя? Почему?

12. У исполнителя Калькулятор две команды:

```
1. вычти 2
2. умножь на 3
```

Какое число получит этот исполнитель, если выполнит программу 211221 для начальных чисел:

а) 1; г) 4; ж) 7;
б) 2; д) 5; з) 8;
в) 3; е) 6; и) 9?

13. Исполнитель Шифровальщик работает с цепочкой символов (символьной строкой), составленной из заглавных русских букв. Каждую букву он заменяет на другую букву, которая стоит в алфавите через три буквы от исходной: А на Г, Б на Д и т.д. При этом букву Я Шифровальщик заменяет на В, букву Ю — на Б и букву Э — на А. Буквы Е и Ё считаются одинаковыми. Расшифруйте сообщения, закодированные с помощью этого алгоритма:

а) ПЛУЦ ПЛУ
б) УСФФЛВ ПСОСЗГВ
в) ЛФХСУЛВ ОБДЕЛ
г) ВЫГ БУНРЦО Е АООЛРЖ

14. Придумайте своего исполнителя: определите его среду, систему команд, способ выполнения каждой команды.

Темы сообщений:
1) “Исполнитель Кузнечик”
2) “Исполнитель Паркетчик”

Способы записи алгоритмов

Ключевые слова:
- словесная запись
- запись по шагам
- блок-схемы алгоритмов
- программа
- язык программирования
- ассемблер
- языки высокого уровня
- логическое программирование

Словесная запись

Как можно записать алгоритм? В первую очередь на естественном языке (русском, английском и др.). Это удобно и привычно, но есть одна проблема: во всех естественных языках есть неоднозначность, поэтому исполнитель-человек может понять алгоритм не так, как задумывал его автор. Особенно трудно разбираться в длинных словесных алгоритмах, занимающих больше десятка строк.

Пример 1. *Автомат получает на вход трехзначное десятичное число. По полученному числу строится новое десятичное число по следующим правилам. Сначала вычисляются сумма старшего и среднего разрядов, а также сумма среднего и младшего разрядов заданного числа. Затем полученные два числа записываются друг за другом в порядке невозрастания (без разделителей).*

Применим этот алгоритм к числу 277. Сначала находим сумму старшего и среднего разрядов: 2 + 7 = 9, и сумму среднего и младшего разрядов: 7 + 7 = 14. Выражение “в порядке невозрастания” означает, что второе число не должно быть больше первого. Так как 9 < 14, запись полученных чисел в порядке невозрастания выглядит так: 149.

Запись по шагам

Для того чтобы в алгоритме легче было разобраться, отдельные шаги принято записывать в виде нумерованного списка:

Пример 2.
Вход: два натуральных числа, a и b.
Шаг 1. Если $a < b$, перейти к шагу 4.
Шаг 2. Уменьшить a на величину b.
Шаг 3. Перейти к шагу 1.
Шаг 4. Стоп.
Результат: значение a.

Обратите внимание, что мы ясно определили, что служит исходными данными для работы этого алгоритма (см. первую строчку — *Вход*) и что будет считаться результатом его работы (строчка *Результат*). Воспринимать такую запись значительно проще, чем длинное словесное описание алгоритма, как в Примере 1.

Подумайте, что вычисляет этот алгоритм и что будет, если применить его к таким исходным данным: $a = 19$, $b = 5$.

Блок-схемы алгоритмов

Словесная запись удобна не во всех случаях. Например, если записать алгоритм на русском языке, то люди, не умеющие читать по-русски, не смогут им воспользоваться. Кроме того, возможно, вы заметили, что даже в простейшем алгоритме из четырех шагов непросто отслеживать переходы между шагами и разобраться, что же происходит с данными. Для более запутанных алгоритмов это становится *очень* сложно.

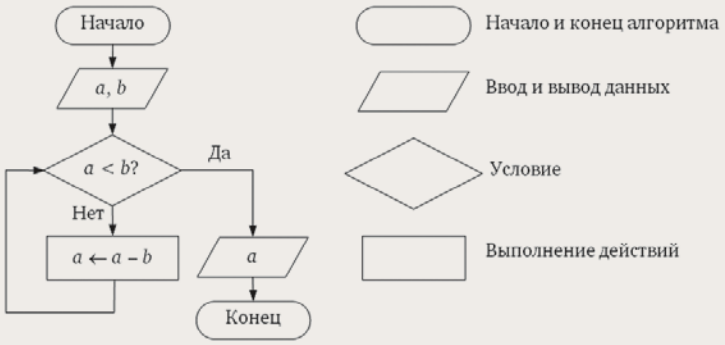


Рис. 5

Поэтому придумали *графическую форму* записи алгоритмов — блок-схемы. Вот как выглядит блок-схема алгоритма из Примера 2 — см. *рис.* 5.

В правой части рисунка показаны условные обозначения, которые используются в блок-схемах. Постарайтесь разобраться в этой схеме, используя словесную запись алгоритма, с которой мы работали раньше.

Основных блоков — четыре: начало и конец алгоритма, ввод и вывод данных, условие и процесс (действие). Внутри блока ▱ записывают данные, которые вводятся или выводятся, в блоке ◇ — условие, которое проверяется, а в блоке ▭ — выполняемое действие. Запись $a \leftarrow a - b$ означает "заменить a на $a - b$".

Стрелки между блоками показывают направление "движения" при выполнении алгоритма. Из блока ◇ ("условие") выходит две стрелки: если записанное внутри условие истинно (верно), нужно перейти по стрелке с надписью "да", а если неверно — то по стрелке с надписью "нет".

Языки программирования

Для того чтобы робот (или другой исполнитель) мог выполнять алгоритмы автоматически (без участия человека), он должен уметь работать в *программном режиме*. Это означает, что алгоритм нужно записать в память исполнителя на специальном языке, который он "понимает". Тогда алгоритм становится *программой*.

> **Программа** — это алгоритм, записанный на языке конкретного исполнителя.

Теперь остается только "запустить" программу, а дальше она будет выполняться автоматически. Для этого исполнитель должен уметь самостоятельно расшифровывать команды и определять, какая команда должна быть выполнена следующей. Это хорошо умеют делать компьютеры — универсальные устройства для обработки информации.

Программы для компьютеров составляются на специальных языках, которые называются *языками программирования*. Этим занимаются люди, профессия которых — *программист*.

Мы уже говорили о том, что все данные в компьютере хранятся в виде двоичных кодов. Программы также хранятся в памяти компьютера как двоичные коды. Но это не значит, что все программы обязательно составляются "на языке нулей и единиц".

Действительно, любой компьютер (точнее, его процессор) понимает и умеет выполнять только команды в двоичном коде. Но так программы никто не пишет, потому что это очень утомительно, отнимает много времени и приводит к многочисленным ошибкам.

Ближе всего к родному языку процессора так называемые *языки низкого уровня*[1], или *языки ассемблера*. В них каждая команда может быть напрямую переведена в соответствующую команду процессора. Например, вот так выглядит программа на языке ассемблера для самого обычного домашнего компьютера, которая складывает числа 5 и 6:

MOV AX, 5	1011 1000 0000 0101 000 000
MOV BX, 6	1011 1011 0000 0110 000 000
ADD AX, BX	0000 0011 1100 0011

Слева от вертикальной линии вы видите команды языка ассемблера[3], а справа — их перевод на "машинный язык", то есть на язык нулей и единиц. Программу, которая переводит символьную запись команд в цепочки битов, называют *ассемблером* (сборщиком).

На языке ассемблера обычно пишут программы, в которых нужно очень высокое быстродействие и небольшой размер — ядро операционной системы, драйверы устройств и т.п.

К сожалению, у каждого семейства процессоров свой язык ассемблера, поэтому программы, написанные для одного семейства, не подходят для другого. Чтобы решить эту проблему, разработали *языки высокого уровня*, которые приближены к естественному языку (чаще всего — к английскому).

Языки высокого уровня никак не связаны с компьютером, на котором будут выполняться.

[2] То есть наиболее близкие к компьютеру.

[3] Конечно, эти команды вам непонятны. Но сейчас важна только общая идея.

Тогда возникает вопрос — как же разные компьютеры будут понимать программы на этих языках? Решение было найдено: для каждого типа процессоров (и операционных систем) создается *транслятор* (переводчик) — программа, которая переводит текст программы, написанной на языке высокого уровня, в двоичные команды нужного процессора.

Первый язык программирования высокого уровня — Фортран — был разработан к 1957 году под руководством известного американского специалиста в области информатики Джона Бэкуса. Язык Фортран до сегодняшнего времени используется в научных расчетах.

В середине 1970-х годов Деннис Ритчи придумал язык Си, который остается одним из самых популярных языков программирования и сегодня. Достаточно сказать, что на нем написаны практически все операционные системы (в том числе *UNIX, Windows, Linux*). На основе языка Си разработано множество современных языков высокого уровня, в том числе C++, C# (читается как “Си шарп”), Java, Javascript и др. Именно эти языки чаще всего используются сегодня для создания программ.

Сейчас очень популярен язык Python, появившийся в 1991 году. Он широко используется в таких известных компаниях, как *Google* и *Яндекс*, применяется для программирования игр. Этот язык прост и содержит большую библиотеку алгоритмов для решения многих стандартных задач, с которыми встречаются программисты. Поэтому решение задачи на языке Python обычно занимает меньше времени, чем при использовании классических языков.

Страницы современных сайтов в Интернете тоже содержат программы. Для этой цели применяют, как правило, языки PHP и Javascript.

Языки Prolog и LISP были разработаны в 1970-х годах как языки для решения задач искусственного интеллекта. Программа на языке логического программирования Пролог, например, — это не алгоритм, а фактически только формулировка задачи. Нужно описать исходные данные, правила вывода (которые используются для решения) и определить цель (что нужно найти). Дальше Пролог-машина сама на основе полученных правил решает задачу, если это возможно. Сейчас эти языки используются редко, в основном — в среде научных работников.

Существуют языки высокого уровня, которые разработаны специально для обучения программированию. Долгое время в качестве первого языка программирования для начинающих использовался язык Бейсик (Basic), но сейчас он вытесняется другими языками.

Один из самых популярных учебных языков — Паскаль — придумал в 1970 году швейцарец Никлаус Вирт. Он назвал новый язык в честь французского физика Блеза Паскаля. Мы начнем изучать язык Паскаль в следующем году.

Со *школьным алгоритмическим языком*, который разработал в 1980-х годах академик А.П. Ершов, мы познакомимся через несколько уроков.

Всего существует несколько тысяч языков программирования, но только 20–30 из них широко используются на практике.

Контрольные вопросы

1. Сравните словесную запись алгоритмов в свободной форме, запись по шагам и блок-схему. Расскажите о преимуществах и недостатках каждой формы записи. Какая из них вам больше нравится? Почему?
2. Что вычисляет алгоритм в Примере 2?
3. Что означает запись $a \leftarrow a + 1$?
4. Какой блок на блок-схеме может иметь более одного входа и более одного выхода? Может ли он иметь один вход и два выхода? А два входа и один выход?
5. Как вы думаете, почему блоки ввода и вывода данных обозначаются на блок-схеме одинаково? Нравится ли вам такое решение?
6. Как вы думаете, почему блок-схемы не используются для записи сложных алгоритмов?
7. Поясните, чем отличаются термины “алгоритм” и “программа”.
8. Чем занимаются программисты? Как вы думаете, какими качествами должен обладать программист? Могла бы вам понравиться эта специальность?
9. Зачем нужны языки ассемблера? В чем их недостатки?
10. Как вы думаете, на каких языках сейчас пишут программы чаще всего: на языках ассемблера или на языках высокого уровня?
11. Как компьютер понимает программу, написанную на языке высокого уровня?
12. Чем отличаются языки логического программирования от других языков высокого уровня?
13. Как вы думаете, почему на практике широко применяются только несколько десятков языков программирования?
14. Какие достоинства и недостатки, на ваш взгляд, имеют языки программирования, разработанные специально для обучения?

Задачи

1. Автомат получает на вход два трехзначных числа. По этим числам строится новое число по следующим правилам.

а) Вычисляются три числа — сумма старших разрядов заданных чисел, сумма средних разрядов этих чисел, сумма младших разрядов.

б) Полученные три числа записываются друг за другом в порядке убывания (без разделителей).

Выполните этот алгоритм для чисел

а) 835 и 196; в) 567 и 291;

б) 138 и 256; г) 239 и 871.

2. Какие из этих чисел не могут быть результатом работы алгоритма из задания 1:

а) 111;
б) 1965;
в) 13127;
г) 131511;
д) 1226;
е) 87?

3. Автомат получает на вход три двухзначных числа. По этим числам строится новое число по следующим правилам:

а) Вычисляются два числа — сумма старших разрядов заданных двухзначных чисел и сумма младших разрядов.

б) Полученные числа записываются друг за другом в порядке возрастания (без разделителей).

Выполните этот алгоритм для чисел:

а) 31, 29, 87;
б) 13, 82, 56;
в) 56, 72, 91;
г) 23, 98, 79.

4. Какие из этих чисел не могут быть результатом работы алгоритма из задания 3:

а) 12168;
б) 121321;
в) 9828;
г) 131511;
д) 1226;
е) 222?

5. Запишите алгоритм в пошаговой форме и в виде блок-схемы: “Дано натуральное число $a > 2$. Присвоить b значение 2 и проверить, делится ли a на b. Если делится, то сделать вывод, что число a — составное. Иначе проверить делимость a на b, где b последовательно принимает все целые значения от 3 до $a - 1$, каждый раз увеличиваясь на 1. Если a не делится ни на одно из этих чисел, сделать вывод, что число a — простое”.

6. Запишите алгоритм в пошаговой форме и в виде блок-схемы: “Дано натуральное число $N > 1$. Присвоить a значение 1, затем присвоить величине d значение 2^a. Если $d \geq N$, завершить выполнение алгоритма и считать результатом a. Иначе увеличить a на единицу, присвоить величине d значение 2^a и перейти к предыдущему шагу”.

Что делает этот алгоритм? Докажите, что он не зациклится при любом натуральном N.

7. Дан алгоритм, записанный в пошаговой форме:

Ввод: натуральное число a.
Шаг 1. Присвоить b значение 1.
Шаг 2. Присвоить b значение $a \cdot b$.
Шаг 3. Присвоить a значение $a - 1$.
Шаг 4. Если $a > 0$, то перейти к **Шагу 2**.
Шаг 5. Стоп.
Результат: значение b.

Что вычисляет этот алгоритм? Нарисуйте его блок-схему. Что получится, если выполнить его для числа 4? Числа 5?

Темы сообщений:

а) “Какие бывают языки программирования?”

б) “Языки программирования с русскими командами”

в) “Академик А.П. Ершов”

Ручное выполнение алгоритмов

Ключевые слова:

- алгоритм
- формальный исполнитель
- ручная прокрутка
- условие
- переменная
- присваивание

Зачем это нужно?

Программистам часто приходится разбираться, почему не работает какой-то алгоритм, свой собственный или даже написанный другим человеком. Для этого нужно выполнить алгоритм вручную для каких-то исходных данных и сравнить результаты каждого шага с тем, что должно было получиться.

Нужно помнить, что мы работаем с формальными исполнителями, которые не могут обдумывать команды, а просто их выполняют.

Выполнение алгоритма по словесной записи

Пример 1. Дан алгоритм обработки цепочки символов:

Сначала вычисляется длина исходной цепочки символов; если она нечетна, то дублируется средний символ цепочки символов, а если четна, то в начало цепочки добавляется буква Г. В полученной цепочке символов каждая буква заменяется буквой, следующей за ней в русском алфавите (А — на Б, Б — на В и т.д., а Я — на А). Получившаяся таким образом цепочка является результатом работы алгоритма. Что получится, если применить этот алгоритм к цепочке ПУСК?

Решение. Согласно алгоритму, найдем длину исходной цепочки ПУСК, она равна 4. Поскольку эта длина четна, в начало цепочки добавляем букву Г, получается ГПУСК.

Теперь каждая буква заменяется на следующую за ней в алфавите: Г ← Д, П ← Р, У ← Ф, С ← Т, К ← Л. Получаем ДРФТЛ.

Пример 2. Вернемся к алгоритму, который встретился нам в начале 0.

Вход: два натуральных числа, a и b.
Шаг 1. Если $a < b$, перейти к шагу 4.
Шаг 2. Уменьшить a на величину b.
Шаг 3. Перейти к шагу 1.
Шаг 4. Стоп.
Результат: значение a.

Вы уже наверняка догадались, что он служит для вычисления остатка от деления a на b с помощью вычитания.

Выполним вручную все шаги этого алгоритма при $a = 19$ и $b = 5$. Это называется *ручной прокруткой алгоритма*.

> **Ручная прокрутка** — это выполнение алгоритма человеком вручную.

Чтобы не держать в памяти все промежуточные результаты, составим таблицу, в которой будем за-

писывать все выполняемые шаги, а также изменения величин a и b:

	Действие		a	b
1			19	5
2	$a < b$?	нет		
3	$a \leftarrow a - b$		14	
4	$a < b$?	нет		
5	$a \leftarrow a - b$		9	
6	$a < b$?	нет		
7	$a \leftarrow a - b$		4	
8	$a < b$?	да		
9	Стоп			

В этой таблице четыре столбца: действие, которое выполняется (или проверяемое условие); результат проверки условия (выполняется или не выполняется?); изменение значений величин a и b. Для удобства строки пронумерованы. В строке 1 записаны начальные значения a и b.

Выполнение алгоритма начинается с шага 1 — проверки условия $a < b$ (строка 2 в таблице). Для начальных значений a и b условие не выполняется, поэтому перехода к шагу 4 нет. Далее выполняется шаг 2: значение a уменьшается на величину b, получается 14. Записываем это новое значение в третий столбец. Затем переходим к шагу 1, условие $a < b$ снова ложно, и т.д.

После того как уменьшение a выполнено трижды, значение a станет равно 4. При очередной проверке условие $a < b$ выполняется (истинно), происходит переход к шагу 4 (на команду Стоп) и алгоритм завершается.

Проверьте самостоятельно, что получится в результате работы этого алгоритма при $a = 15$ и $b = 19$.

Выполнение алгоритма по блок-схеме

Пример 3. Дана блок-схема алгоритма:

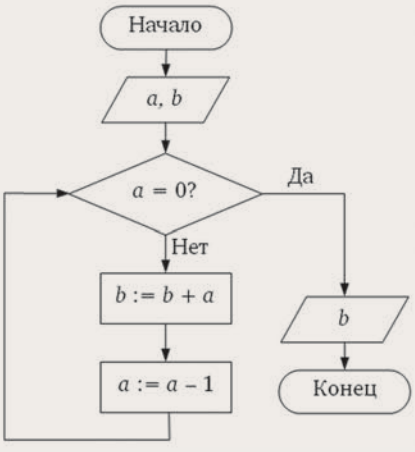


Рис. 6

Требуется определить результат работы этого алгоритма при входных значениях $a = 4$ и $b = 0$.

Наверное, вы заметили, что в двух блоках этой схемы используется новое обозначение ":=". Эти два символа во многих языках программирования обозначают присваивание. Операцию "заменить a на $a - 1$", которую мы до этого записывали как "$a \leftarrow a - 1$", можно записать еще и так: "$a := a - 1$". Именно второй способ используется, например, в школьном алгоритмическом языке и языке Паскаль, которые мы будем изучать в курсе информатики. Это связано с тем, что знака "←" нет на стандартной клавиатуре, и набирать ":=" значительно удобнее и быстрее.

Сначала нужно выполнить *анализ* алгоритма. Слово "анализ" обозначает "разделение на части", изучение внутреннего устройства. Нам необходимо определить:

1) какие исходные данные принимает алгоритм;
2) что будет результатом алгоритма;
3) как работает этот алгоритм.

В начале алгоритма вводятся исходные данные — значения a и b. После завершения работы алгоритма выводится результат — значение b.

На блок-схеме мы видим стрелку перехода к предыдущей команде (снизу вверх). Это означает, что две команды могут выполняться несколько раз: сначала b заменяется на сумму $b + a$, а затем a уменьшается на единицу. Когда значение a станет равно нулю, происходит выход из блока "условие" по ветке "да" на блок вывода результата, и работа алгоритма заканчивается.

Докажем, что алгоритм завершится (не будет работать бесконечно). Начальное значение a равно 4, и оно на каждом шаге уменьшается на 1. Поэтому через четыре шага a станет равно нулю и работа алгоритма завершится.

Выполним ручную прокрутку алгоритма, записывая все промежуточные данные в таблицу:

	Действие		a	b
1			**4**	**0**
2	a = 0?	нет		
3	b := b + a			**4**
4	a := a - 1		**3**	
5	a = 0?	нет		
6	b := b + a			**7**
7	a := a - 1		**2**	
8	a = 0?	нет		
9	b := b + a			**9**
10	a := a - 1		**1**	
11	a = 0?	нет		
12	b := b + a			**10**
13	a := a - 1		**0**	
14	a = 0?	**да**		

В первой строке записываем начальные значения a и b. Далее по блок-схеме прослеживаем ход выполнения алгоритма. Сначала проверяется условие (строка 2), оно ложно, поэтому во втором столбце пишем "нет".

Переходим по ветке "нет". Сначала выполняется присваивание $b := b + a$: определяем по таблице текущие (последние) значения величин a и b, скла-

дываем их и записываем в столбец изменения b (строка 3). Затем уменьшаем значение a на единицу (строка 4). После того как эти операции повторятся четыре раза, условие становится истинным ($a = 0$), и алгоритм завершает работу (строка 14).

Результат работы алгоритма — это значение величины b. Как мы видим из таблицы, последнее значение, присвоенное b, равно 10. Это и есть ответ.

Подумайте, что делает этот алгоритм. Для ответа на этот вопрос проверьте, что изменится, если начальное значение a будет равно 5, 6 и т.д.

При ручной прокрутке алгоритма мы видели, что величины a и b изменяются, им присваиваются новые значения. Такие величины в информатике называются *переменными*.

Переменная — это величина, значение которой можно изменять во время работы алгоритма.

Операция, обозначаемая знаками ":=", — это присваивание нового значения переменной.

Контрольные вопросы

1. Объясните, зачем нужно уметь выполнять алгоритмы вручную.
2. Докажите, что алгоритмы в примерах 1 и 2 обязательно завершатся (не зациклятся).
3. Что такое "анализ алгоритма"? Где в жизни вам приходилось выполнять анализ ситуации?
4. Как по блок-схеме можно определить, что какие-то команды могут выполняться несколько раз?
5. Изменится ли результат работы алгоритма на 0, если поменять местами блоки с операциями $b := b + a$ и $a := a - 1$? Почему? Что произойдет, если их все-таки поменять местами?
6. Объясните, зачем нужны переменные при записи алгоритмов.

Задачи

1. Что получится, если применить алгоритм из примера 1 к словам
   а) УРА; в) КРОНА;
   б) НОРА; г) ЯРАНГА?
2. Что получится, если дважды применить алгоритм из примера 1 к словам
   а) РЕКА; в) РОСТОВ;
   б) МУРКА; г) МАЛЮТКА?
3. Дан следующий алгоритм: "*Сначала вычисляется длина исходной цепочки символов; если она четна, то в середину цепочки добавляется буква А, а если нечетна, то в начало цепочки добавляется буква Б. В полученной цепочке символов каждая буква заменяется буквой, следующей за ней в русском алфавите (А — на Б, Б — на В и т.д., а Я — на А). Получившаяся таким образом цепочка является результатом работы алгоритма*".

Что получится, если применить этот алгоритм к цепочкам
   а) СТАРТ; в) ХРЯМЗИК;
   б) МОЛОКО; г) КОМПЬЮТЕР?
4. *Цепочки символов были обработаны с помощью алгоритма из задания 3. В результате получились такие цепочки:
   а) НПТБЛГБ; в) ВРЕУЕСВФСД;
   б) ВАДПЕБ; г) ВРБСБЩЯУ.
   Определите исходные цепочки.
5. *Цепочки символов были дважды обработаны с помощью алгоритма из задания 3. В результате получились такие цепочки:
   а) ВОБВУР;
   б) ГПЖБТМВ;
   в) ВЩВДВЭЩВ;
   г) ГНРТБГХЪВ.
   Определите исходные цепочки.
6. Алгоритм из задачи 3 применен к слову четыре раза. В результате получилась цепочка из шести символов. Сколько символов было в исходной цепочке? Можете ли вы восстановить некоторые символы полученной цепочки?
7. Определите значение переменной b после выполнения этого алгоритма (здесь и далее для экономии места мы не будем рисовать блоки "начало" и "конец"):

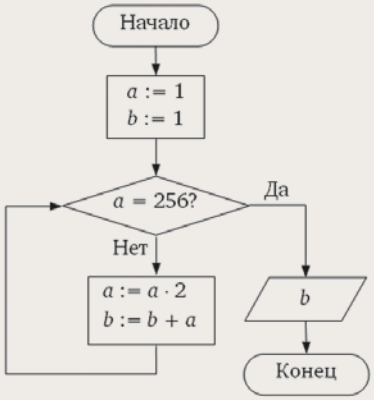


Рис. 7

8. Определите значение переменной b после выполнения следующего алгоритма:

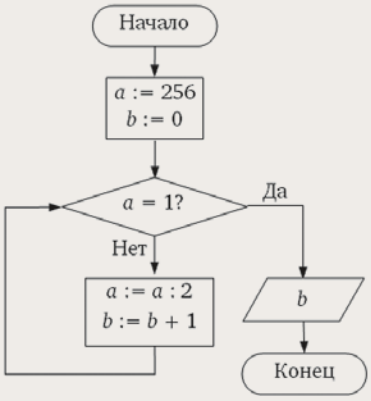


Рис. 8

9. Определите значение переменной b после выполнения следующего алгоритма:

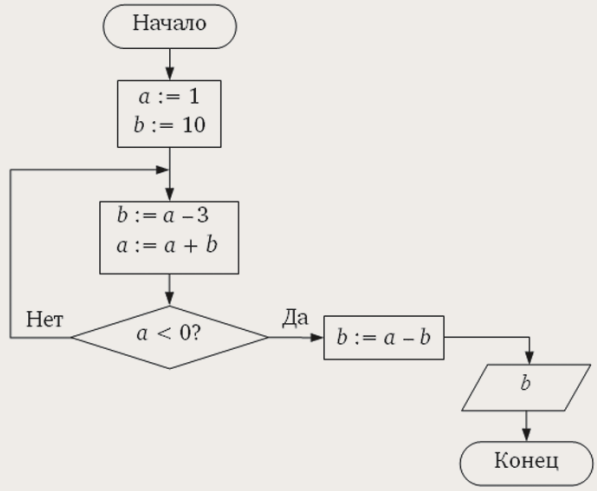


Рис. 9

10. *Определите значение переменной b после выполнения следующего алгоритма:

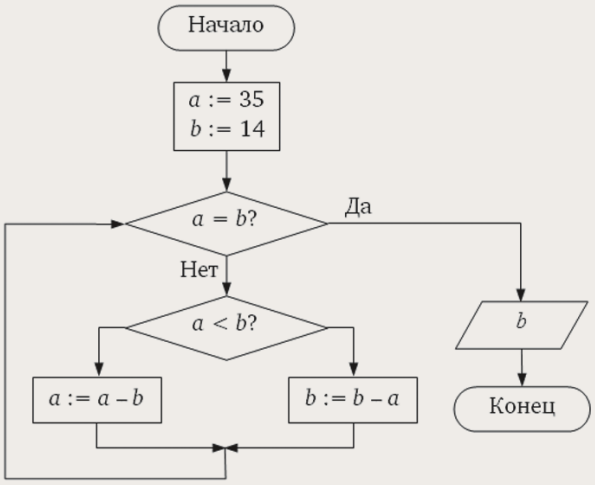


Рис. 10

11. *Определите значение переменной x после выполнения следующего алгоритма (знак “≥” обозначает “больше или равно”):

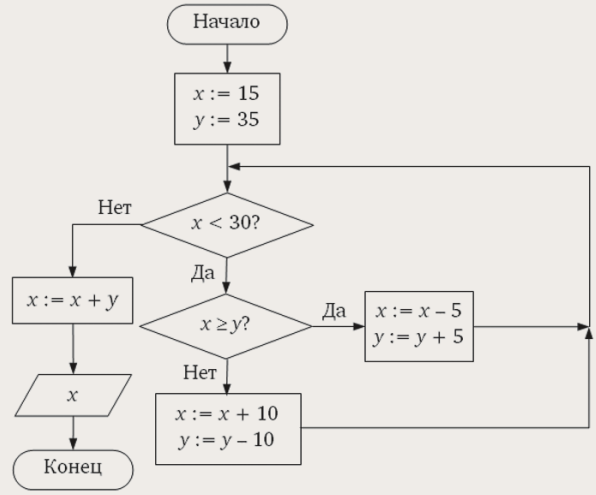


Рис. 11

Линейные алгоритмы

Ключевые слова:
- линейный алгоритм
- дерево вариантов

В этом и следующих параграфах мы познакомимся с различными типами алгоритмов и научимся их составлять.

Что такое линейный алгоритм?

В простейших алгоритмах все действия выполняются последовательно, одно за другим, при любых исходных данных. Пример такого алгоритма — вычисления по готовым формулам.

Задача 1. Сколько километров прошел автомобиль за $t = 2$ часа, если его скорость $v = 60$ км/ч?

Как мы уже говорили, одну такую задачу несложно решить на калькуляторе, а для быстрого решения серии однотипных задач лучше использовать компьютерную программу, например, электронную таблицу.

Алгоритм решения можно записать в словесной форме:

Вход: v, t.

Шаг 1. Присвоить S значение $v \cdot t$.

Результат: S.

Заметьте, что здесь нет ни одной команды перехода, которая направляет исполнителя на другой шаг. Поэтому порядок действий всегда одинаковый, какие бы исходные данные мы ни вводили.

Этот же алгоритм можно записать в виде блок-схемы:

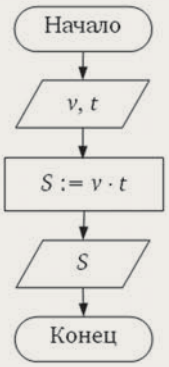


Рис. 12

Видим, что все блоки расположены “в одну линию”, которая идет между начальной и конечной точками алгоритма. Поэтому такой алгоритм называется *линейным*, он имеет единственный вариант исполнения.

> В **линейном** алгоритме команды выполняются в том же порядке, в котором они записаны.

Программы для Удвоителя

Задача 2. Составьте самую короткую программу для Удвоителя, которая преобразует число 6 в 28.

Напомним, что в СКИ Удвоителя всего две команды:

1. прибавь 1
2. умножь на 2

Решение задачи — это линейный алгоритм (последовательность команд).

Возможно, в этой простой задаче вы сразу сообразили, какой будет эта самая короткая программа. Но в более сложных случаях это не так просто. Кроме того, нам нужно доказать, что это действительно самая короткая программа. Поэтому рассмотрим общий способ решения.

Решение. Сначала проверим все программы, состоящие из одной команды. Таких программ всего две, в результате мы можем получить либо число 7 (если выполнена команда 1), либо число 12 (по команде 2):

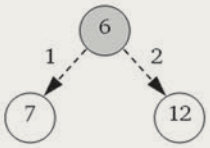


Рис. 13

Числа около стрелок показывают номер выполняемой команды. Таким образом, за один шаг можно получить только числа 7 и 12, никаких других чисел (в том числе заданного числа 28) получить нельзя.

Теперь рассматриваем все варианты программ из двух шагов. Из числа 7 на втором шаге можно получить 8 или 14, а из числа 12 — только 13 и 24.

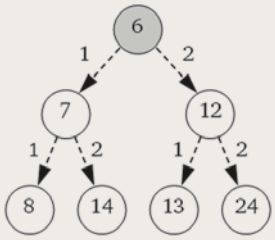
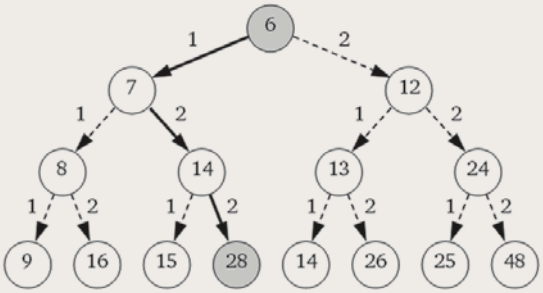


Рис. 14

Следовательно, за два шага получить число 28 тоже не удается.

Проверяем трехшаговые программы:



дерево

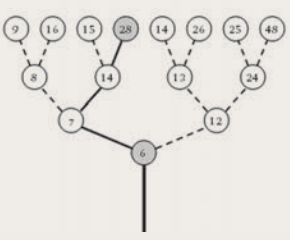


Рис. 15

На третьем шаге одно из полученных чисел — 28. Это значит, что самая короткая программа состоит из трех шагов. Чтобы построить саму программу, нужно пройти по стрелкам от начального числа к нужному результату (этот путь показан сплошной линией) и выписать номера всех встречающихся команд. В данном случае получаем 122.

Ответ: 122.

Схема, которую мы построили, называется *деревом возможных вариантов*. Действительно, мы рассмотрели все возможные программы, состоящие из одного, двух и трех шагов. Если полученный рисунок развернуть вверх ногами, он напоминает дерево.

Построение полного дерева вариантов при большой длине программы (например, 6 или 8 команд) — это очень утомительная работа. Во многих задачах бывает проще двигаться не от начального числа к результату, а наоборот.

Итак, возьмем конечное число 28 и подумаем, какой последней командой мы могли его получить. Так как 28 делится на 2, это могла быть как команда 1 (из 27 получаем 28), так и команда 2 (из 14 также получаем 28).

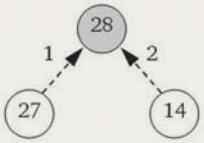


Рис. 16

Ни одно из этих чисел не совпадает с начальным (6), поэтому одной командой задачу не решить. Делаем второй шаг — проверяем, из каких чисел мы могли получить 27 и 14. Число 27 не делится на 2, поэтому оно могло быть получено только командой 1 (из 26), а число 14 можно получить из 13 или из 7:

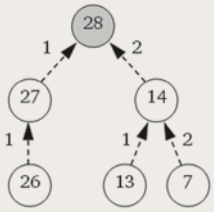


Рис. 17

До начального числа 6 снова добраться не удалось, поэтому решений из двух команд не существует. Делаем третий шаг: число 26 можно получить из 25 или из 13, число 13 — только из 12, и число 7 — только из 6 (а это и есть начальное число):

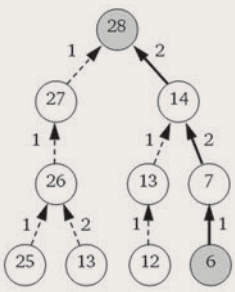


Рис. 18

Для того чтобы получить самую короткую программу, нужно пройти по стрелкам от начального числа к конечному, выписывая встречающиеся номера команд. Для нашей задачи получаем тот же ответ, что и раньше, — 122 (путь выделен сплошной линией).

Обратите внимание, что дерево при движении "в обратном направлении", от конечного числа к начальному, получилось меньше. Нам удалось найти решение быстрее, рассматривая меньше вариантов. Почему в некоторых случаях в дереве не было разделения на две ветки? Только потому, что не все числа делятся на 2, то есть не все числа могут быть получены командой 2.

Контрольные вопросы

1. Приведите примеры линейных алгоритмов, которые вы выполняете в жизни.
2. Какие алгоритмы, с которыми вы уже знакомы, нельзя считать линейными? Почему?
3. Как вы думаете, в чем недостаток линейных алгоритмов?
4. Может ли быть так, что задача для Удвоителя решается с помощью нескольких различных алгоритмов? Если да, приведите примеры.
5. Как можно доказать, что построенная программа для Удвоителя действительно самая короткая?
6. Какие числа можно получить из натурального числа *N* с помощью Удвоителя? Из нуля? Из отрицательного числа?
7. Как быстро построить самую короткую программу для получения некоторого числа *N* из нуля с помощью Удвоителя? Когда эта задача не имеет решений?
8. Исполнитель Калькулятор имеет команды

```
1. прибавь 1
2. раздели на 2
```

Нужно составить самую короткую программу для Калькулятора, с помощью которой из числа *a* можно получить число *b*. Как лучше перебирать варианты программ, от начального числа к конечному или наоборот? Почему?

Задачи

1. Удвоителю нужно получить из числа 1 число 4. Сколькими способами это можно сделать? Выпишите все решения в виде программ для Удвоителя.
2. Удвоителю нужно получить из числа 3 число 12. Сколькими способами это можно сделать? Выпишите все решения в виде программ для Удвоителя.
3. Исполнитель Раздвоитель имеет команды

```
1. вычти 1
2. раздели на 2
```

Напишите самую короткую программу для Раздвоителя, которая преобразует

а) число 8 в число 0;
б) число 9 в число 0;
в) число 16 в число 0;
г) число 15 в число 0.

4. У исполнителя Калькулятор две команды:

```
1. вычти 3
2. умножь на 2
```

Напишите самую короткую программу для Калькулятора, которая преобразует

а) число 8 в число 14;
б) число 7 в число 19;
в) число 8 в число 17;
г) число 8 в число 40.

5. У исполнителя Калькулятор две команды:

```
1. прибавь 2
2. умножь на 3
```

Напишите самую короткую программу для Калькулятора, которая преобразует

а) число 5 в число 71;
б) число 5 в число 83;
в) число 4 в число 50;
г) число 4 в число 128.

6. У исполнителя Калькулятор две команды:

```
1. вычти 2
2. умножь на 3
```

Напишите самую короткую программу для Калькулятора, которая преобразует

а) число 6 в число 24;
б) число 7 в число 33;
в) число 8 в число 50;
г) число 10 в число 80.

7. * У исполнителя Калькулятор две команды:

```
1. вычти 4
2. умножь на 3
```

Напишите самую короткую программу для Калькулятора, которая преобразует

а) число 6 в число 22;
б) число 6 в число 118;
в) число 8 в число 22;
г) число 10 в число 34.

Программирование Робота

Ключевые слова:

- программа
- линейный алгоритм
- алгоритмический язык
- ключевые слова языка
- синтаксические ошибки
- отказы
- логические ошибки

Как оформить программу для Робота?

Вы уже знакомы с исполнителем Робот, который умеет ходить по клетчатому полю. Теперь мы научимся составлять для Робота настоящие программы (программировать исполнителя) в системе КуМир (*Комплект Учебных Миров*).

Оказывается, Робот может не только ходить по клетчатому полю, но и выполнять полезную работу — закрашивать клетку, в которой он стоит. Для этого используется команда

закрасить

Клетки, которые нужно закрасить по заданию, мы будем обозначать серой точкой в правом нижнем углу. Например, вот такая задача для Робота:

Рис. 19

решается следующим *линейным* алгоритмом:

```
вниз
закрасить
влево
закрасить
```

Для того чтобы Робот смог выполнить программу, нужно оформить ее так, чтобы она была понятна системе КуМир. Программа для решения задачи на *рис.* 19 выглядит так:

```
использовать Робот
алг Переход
нач
  вниз
  закрасить
  влево
  закрасить
кон
```

В первой строчке мы подключаем исполнителя. В КуМире несколько исполнителей (Робот, Черепаха, Чертежник, Рисователь), и в начале программы необходимо сообщить, для какого исполнителя написана эта программа. Любая программа для Робота должна начинаться строчкой

использовать Робот.

Программа на алгоритмическом[4] языке начинается словом **алг**, за которым записывают (необязательное) имя алгоритма. Мы назвали его *Переход*. Алгоритм начинается словом **нач** (сокращения от слова *начало*) и заканчивается словом **кон** (сокращение от слова *конец*). Между словами **нач** и **кон** записывают все команды, которые должен выполнить исполнитель. Видим, что в данном случае нет никакой возможности пропустить какую-то команду или выполнить команды не по порядку. Значит, это линейный алгоритм.

Слова **алг**, **нач** и **кон** — это ключевые слова алгоритмического языка. Они всегда имеют единственное значение, которое задали им разработчики языка. Использовать их в других ситуациях (придавать другой смысл) нельзя.

[4] Для краткости мы будем называть школьный алгоритмический язык просто “алгоритмический язык”.

Ключевые слова — это специальные слова языка программирования, имеющие единственное заранее определенное значение.

Для проверки нужно набрать программу в среде КуМир и запустить исполнителя, нажав клавишу F9.

Какие могут быть ошибки?

При написании программ неизбежны ошибки. Это связано с тем, что возможности человека, к сожалению, ограниченны, и даже хороший программист редко может написать сразу без ошибок программу длиннее, чем 40–50 строчек. Бояться ошибок не нужно, нужно научиться обнаруживать и исправлять их.

Прежде всего нужно понимать, что ошибки бывают разные:

1) **синтаксические** ошибки (ошибки типа “не понимаю”) — неверное написание команды. Например, если вместо слова **закрасить** вы напишете **закрась**, Робот не сможет понять такую команду, потому что ее нет в его СКИ (хотя для человека эти команды означают одно и то же);

2) **отказы** (ошибки типа “не могу”) — это ситуации, когда исполнитель понимает команду, но не может ее выполнить. Например, в этой ситуации Робот не сможет выполнить команду **вниз**, потому что под ним — стенка, Робот врежется в нее и разрушится:

Рис. 20

3) **логические** ошибки — это ошибки, которые исполнитель обнаружить не может: он понимает все команды, успешно выполняет их, но результат его работы не совпадает с тем, который нужен. Например, Робот не закрасил все клетки, которые нужно закрасить, или закрасил лишние.

Самое сложное — это обнаружить и исправить логические ошибки. Для этого часто приходится выполнять ручную прокрутку программы или выполнять ее в пошаговом режиме. Это означает, что мы выполняем команды не все сразу, а по одной, останавливая исполнителя после каждой выполненной команды. Так мы сможем определить, на каком шаге исполнитель сделал не то, что мы предполагали.

Пошаговый режим в системе КуМир включается нажатием клавиши F8. Если нажать клавишу F8 еще раз, исполнитель выполняет очередную команду (только одну!). После каждого шага цветным фоном выделяется команда, которая будет выполнена следующей.

Контрольные вопросы

1. Верно ли, что для каждой задачи существует единственный алгоритм решения? Ответ обоснуйте.

2. Как можно сравнить два различных алгоритма решения одной и той же задачи? Как выбрать лучший их них?

3. Два друга по-разному ищут ошибки в программах. Кирилл, написав программу, сразу запускает ее для того, чтобы транслятор обнаружил все синтаксические ошибки. Даниил же сначала внимательно изучает текст программы и пытается найти ошибки сам, а потом уже запускает ее на выполнение. Чем хорош каждый из методов?

4. Чем отличаются синтаксические ошибки и отказы?

5. Какие способы поиска логических ошибок в программах вы знаете?

6. К какому типу ошибок относится зацикливание алгоритма?

Задачи

1. Найдите другой способ решения задачи на *рис.* 19.

2. Сколькими способами Робот может перейти в клетку Б, если ему разрешено выполнить не более четырех команд?

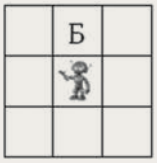


Рис. 21

Можно ли перейти в эту клетку ровно за четыре команды? За 2000 команд? за 2015 команд?

3. Напишите программы для Робота, которые решают следующие задачи (Робот должен закрасить все отмеченные клетки и прийти в клетку Б):

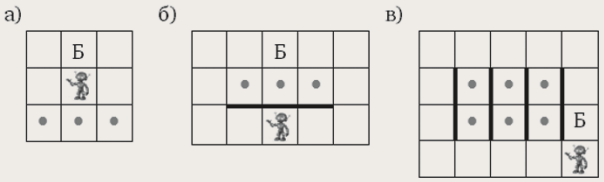


Рис. 22

Сравните ваши решения с решениями ваших одноклассников. У кого оказались самые короткие программы?

4. Тимофей написал программу для Робота, после выполнения которой исполнитель возвращается в начальное положение. Арсений похулиганил и стер одну команду. Помогите Тимофею восстановить ее:

```
влево
вверх
вверх
вправо
вниз
```

В каком месте программы могла быть стертая команда? От чего это зависит?

5. Тимофей написал программу для Робота, после выполнения которой исполнитель возвращается в начальное положение. Арсений похулиганил и стер одну команду. Но при запуске измененной программы Робот снова вернулся в начальное положение. Какую команду стер Арсений?

6. Тимофей написал программу для Робота, после выполнения которой исполнитель возвращается в начальное положение. Арсений похулиганил и переставил местами две команды в программе. Что может произойти при запуске этой программы?

7. Тимофей написал программу для Робота, после выполнения которой исполнитель возвращается в начальное положение. Арсений похулиганил и стер несколько команд. Помогите Тимофею восстановить ее:

```
влево
влево
вверх
вправо
вверх
```

Какое минимальное количество команд нужно добавить, чтобы восстановить программу? Какие это команды?

8. Тимофей написал программу, при выполнении которой Робот движется из клетки А в клетку Б:

```
влево
вверх
закрасить
вверх
вправо
закрасить
вниз
```

Напишите программу, которая переводит Робота в обратном направлении, из клетки Б в клетку А. Робот должен закрасить те же самые клетки.

9. Тимофей написал программу для Робота, при выполнении которой исполнитель закрашивает три клетки. Арсений похулиганил и поменял местами две команды в программе. Может ли новый алгоритм закрашивать

а) 0 клеток;

б) 2 клетки;

в) 4 клетки;

г) 10 клеток?

Тема для сообщения:

“Роботы вокруг нас”

Вспомогательные алгоритмы

Ключевые слова:

- вспомогательный алгоритм
- процедура
- вызов процедуры
- возврат из процедуры
- проектирование “снизу вверх”
- проектирование “сверху вниз” (метод последовательного уточнения)

В этом параграфе вы узнаете, как научить исполнителя выполнять новые команды.

Что такое вспомогательный алгоритм?

В этой задаче Роботу требуется закрасить три квадратных участка размером 2 × 2 клетки:

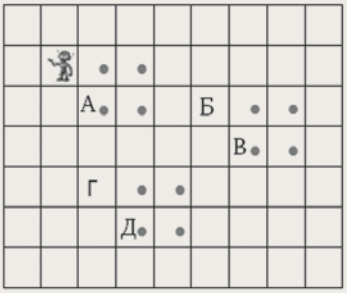


Рис. 23

Мы видим, что Роботу нужно будет трижды выполнить одинаковые команды. Наверное, вы подумали, что было бы хорошо, если бы у Робота уже была команда, которая сразу закрашивает квадрат размером 2 × 2? Тогда бы мы просто три раза использовали эту команду.

Хотя такой команды нет, мы можем ее ввести сами, расширяя СКИ исполнителя. Но при этом мы должны объяснить Роботу, что он должен делать, выполняя эту команду.

Итак, введем новую команду — **Квадрат**, выполняя которую Робот закрасит квадрат размером 2 × 2 клетки и остановится в его левом нижнем углу. Нарисуем условие этой вспомогательной задачи (Робот должен прийти в клетку А):

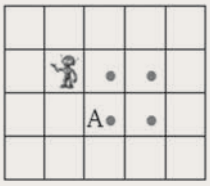


Рис. 24

Мы можем прямо сейчас написать ее решение:

```
алг Квадрат
нач
  вправо; закрасить
  вправо; закрасить
  вниз; закрасить
  влево; закрасить
кон
```

Здесь в каждой строке записаны две команды, их нужно разделять точкой с запятой.

Обратите внимание, что мы должны обязательно определить, где будет находиться Робот перед началом работы (слева от левого верхнего угла квадрата) и где он остановится после завершения работы (в правом нижнем углу квадрата).

> **Вспомогательный алгоритм** — это новая команда, которой мы дополняем СКИ исполнителя.

Как использовать вспомогательный алгоритм?

Теперь используем новую команду Квадрат для решения задачи на *рис.* 23. Видим, что Робот стоит в удачном исходном положении, так что можно сразу применить команду **Квадрат**. После ее выполнения Робот остановится в клетке А. Для того чтобы снова можно было использовать команду **Квадрат**, его нужно перевести в клетку Б:

```
Квадрат
вправо; вправо; вправо
```

Дальше снова применяем новую команду **Квадрат** и выводим Робота из клетки В, где он остановится, в клетку Г. После этого используем команду **Квадрат** в третий раз. Робот обрабатывает последний квадрат и останавливается в клетке Д:

```
Квадрат
вниз
влево; влево
влево; влево
Квадрат
```

Получается такая основная программа (приведем ее целиком):

```
алг Три квадрата
нач  Квадрат
  вправо; вправо; вправо
  Квадрат
  вниз
  влево; влево
  влево; влево
  Квадрат
кон
```

Не забудьте, что в первой строчке программы нужно подключить исполнителя командой

```
использовать Робот
```

Как оформить программу?

Теперь, для того чтобы все сработало, необходимо *ниже основного алгоритма* написать объяснение (расшифровку) команды **Квадрат**. Это *вспомогательный алгоритм*, который часто называют *процедурой*. Обратите внимание, что имя процедуры должно быть точно такое же, какое мы использовали в основной программе, причем заглавные и строчные буквы различаются, то есть **квадрат** и **Квадрат** — это разные имена.

Давайте разберемся, что будет происходить при запуске этой программы. Алгоритм, записанный в самом начале программы (у нас это алгоритм **Три квадрата**), называется основной программой. Исполнитель выполняет именно основную программу. В первой же строке он встречает неизвестную команду **Квадрат**, которая не входит в его СКИ. Это *вызов вспомогательного алгоритма*, или *вызов процедуры*.

> Для того чтобы вызвать вспомогательный алгоритм (процедуру), нужно записать его название в тексте другого алгоритма.

Встретив такой вызов, исполнитель ищет процедуру с названием **Квадрат** ниже основной программы. Если процедура **Квадрат** найдена, исполнитель выполняет все команды, записанные в теле этого вспо-

могательного алгоритма[5]. Затем происходит *возврат из процедуры* — передача управления на ту команду в основной программе, которая записана сразу после команды **Квадрат** (это команда **вправо**).

> После завершения работы вспомогательного алгоритма управление передается обратно, к следующей команде вызывающей программы.

Далее выполняются команды

```
вправо; вправо; вправо
```

(они входят в СКИ и не требуют расшифровки), а потом исполнитель снова встречает вызов процедуры **Квадрат**, уже второй по счету. Он действует так же, как в первом случае.

Обратите внимание, что если в основной программе нет вызова процедуры **Квадрат**, она не выполнится ни разу, хотя и будет записана в тексте программы.

> Вспомогательный алгоритм выполняется только тогда, когда он вызван из основной программы.

В только что рассмотренной задаче мы сначала написали вспомогательный алгоритм, а затем использовали его при составлении основной программы. Такой метод называется проектированием **снизу вверх** — решение основной задачи "собирается" из уже написанных ранее вспомогательных алгоритмов, как из кубиков.

Метод последовательного уточнения

Существует и другой подход к разработке программ, который называется программированием **сверху вниз**, или методом **последовательного уточнения**. Исходная задача разбивается на части (подзадачи) и сначала записывается основной алгоритм. Он использует вызовы еще не написанных вспомогательных алгоритмов, каждый из которых решает свою подзадачу. Затем пишем вспомогательные алгоритмы, если нужно, так же разбивая их на части.

Покажем этот подход на примере еще одной задачи для Робота:

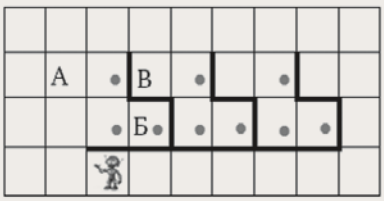


Рис. 25

Сразу видны три одинаковых блока из трех клеток, которые нужно закрасить Роботу. Робот расположен не очень удачно, так что в самом начале его нужно перевести в начальное положение рядом с первым блоком клеток, например, в клетку А. Назовем новую команду, которая это сделает, **Подход**.

Переход между первым и вторым блоками и между вторым и третьим выполняется одинаково. Мы введем еще две команды: **Блок** (обработка одного блока) и **Переход** (переход к следующему блоку). Теперь можно написать основную программу:

```
алг Блоки
нач
  Подход
  Блок
  Переход
  Блок
  Переход
  Блок
кон
```

Обратите внимание, что самих вспомогательных алгоритмов еще нет, но мы их уже используем, основная программа полностью готова.

Конечно, такая программа не будет работать, потому что исполнитель не знает, как выполнить команды **Подход**, **Блок** и **Переход**. Процедура **Подход** должна перевести Робота в клетку А:

```
алг Подход
нач
  влево; вверх; вверх
кон
```

Процедура **Блок** закрашивает блок из трех клеток; если Робот начал работу в клетке А, то он остановится в клетке Б:

```
алг Блок
нач
  вправо; закрасить
  вниз; закрасить
  вправо; закрасить
кон
```

А процедура **Переход** переводит Робота в исходное положение для обработки следующего блока (из клетки Б в клетку В):

```
алг Переход
  влево; вверх; вверх
  вправо; вниз
кон
```

Чтобы собрать программу, нужно сначала записать основной алгоритм, а за ним — все процедуры.

Во время отладки часто бывает нужно "войти" в процедуру и выполнить ее по шагам. Для этого в среде КуМир используется клавиша F7 (если вместо этого нажать клавишу F8, то процедура будет выполнена полностью).

Контрольные вопросы

1. Какие задачи легче решаются с использованием вспомогательных алгоритмов?
2. Можно ли использовать вспомогательные алгоритмы, если в программе нет повторяющихся действий? Зачем это может быть нужно?
3. В каких ситуациях вы не рекомендовали бы использовать вспомогательные алгоритмы?

[5] Как вы думаете, а что будет, если процедура **Квадрат** не будет найдена? Проверьте свою догадку в системе КуМир.

4. Что произойдет, если исполнитель не обнаружит расшифровки новой команды?

5. Где нужно записывать вспомогательные алгоритмы?

6. Что такое основная программа?

7. Вспомогательный алгоритм есть в тексте программы, но не срабатывает. В чем может быть причина?

8. Почему при составлении вспомогательного алгоритма нужно четко определить начальное и конечное положения исполнителя?

9. Можно ли записать в одной строке пять команд? Если да, как это сделать?

10. Как найти в программе вызов процедуры?

11. Как выполняется возврат из процедуры?

12. Куда передается управление, если процедура вызывается в самом конце основной программы (за ним нет других команд)?

13. Какой метод программирования вам больше нравится, “снизу вверх” или “сверху вниз”? Подумайте, какие достоинства и недостатки имеет каждый метод.

14. Как выполнить процедуру по шагам?

Задачи

1. Робот выполнил алгоритм с процедурой:

```
алг Площадка
нач
  к4
  вправо; вверх; вверх
  к4; вправо
  к4
кон
алг к4
нач
  закрасить
  вправо; закрасить
  вниз; закрасить
  влево; закрасить
кон
```

Ответьте на вопросы:

а) Как будет выглядеть площадка, закрашенная Роботом?

б) Сколько клеток закрасит Робот?

в) Сколько клеток будет закрашено дважды?

г) Сколько клеток будет закрашено трижды?

2. Составьте программы для Робота, решающие следующие задачи (Роботу нужно закрасить все отмеченные клетки и прийти в клетку Б):

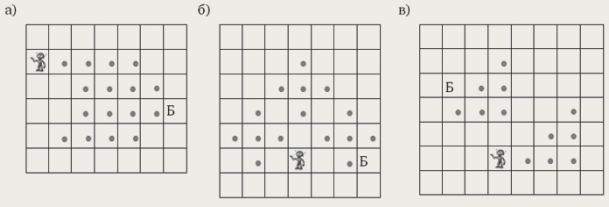


Рис. 26

Используйте вспомогательные алгоритмы.

3. Составьте программы для Робота, решающие следующие задачи (Роботу нужно закрасить все отмеченные клетки и прийти в клетку Б):

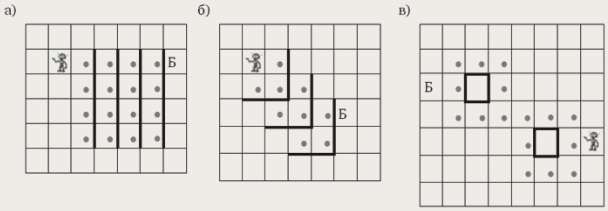


Рис. 27

Используйте вспомогательные алгоритмы:

Циклические алгоритмы

Ключевые слова:

- циклический алгоритм
- тело цикла
- вложенный цикл
- комментарий

Что такое циклический алгоритм?

Рассмотрим такую простую задачу для Робота:

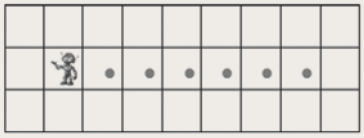


Рис. 28

Напомним, что серые точки в углах клеток означают, что эти клетки нужно закрасить.

Эту задачу можно решить с помощью линейного алгоритма:

```
вправо; закрасить
вправо; закрасить
вправо; закрасить
вправо; закрасить
вправо; закрасить
вправо; закрасить
```

Конечно, для запуска этой программы в среде КуМир нужно еще добавить команды **алг**, **нач** и **кон**, но для сокращения записи мы не будем повторять их каждый раз.

Теперь представьте себе, что Роботу нужно закрасить не шесть клеток, а 1006. Какой длины получится программа? Чтобы не повторять много раз одинаковые строчки в тексте программы, в языки программирования ввели специальные команды, которые говорят исполнителю, сколько раз нужно повторить те или иные действия. Такие команды называют *циклами*, а алгоритмы — *циклическими*.

> **Циклический алгоритм** — это алгоритм, в котором некоторая последовательность действий выполняется несколько раз.

В алгоритмическом языке существует цикл “*N* раз”, который записывают так:

```
нц 6 раз
  вправо
  закрасить
кц
```

Здесь **нц** обозначает “начало цикла”, а **кц** — конец цикла. Строчка **нц 6 раз** — это заголовок цикла, в котором указано число повторений.

Этот алгоритм можно записать в виде блок-схемы:

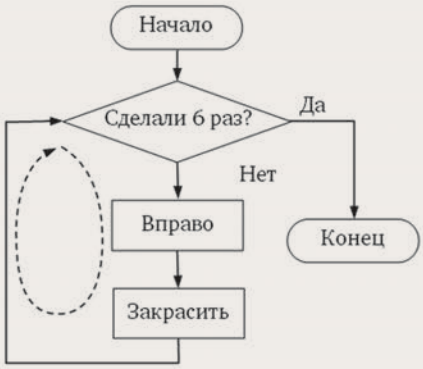


Рис. 29

Отличительная черта циклического алгоритма — замкнутый контур (петля), то есть стрелка, идущая к предыдущим командам (на этой блок-схеме — наверх). На *рис.* 29 эта петля обозначена штриховой линией.

Команды, которые выполняются в цикле (многократно), называются *телом цикла*. В нашем примере тело цикла состоит из двух команд: **вправо** и **закрасить**.

> **Тело цикла** — это команды, которые выполняются несколько раз.

Теперь немного усложним задачу: Роботу придется обходить стенки:

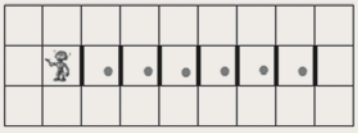


Рис. 30

Как изменится решение? Конечно, команда **закрасить** в конце тела цикла останется, но вместо команды **вправо** нужно использовать серию из трех команд:

```
вверх
вправо
вниз
```

Получается такое решение:

```
нц 6 раз
  вверх; вправо
  вниз;  закрасить
кц
```

Выбор начального положения

В только что решенных задачах Робот занимал очень удачную позицию: можно было сразу начинать цикл. Это не всегда так, чаще всего сначала приходится “подвести” исполнителя к клетке, с которой удобно начать действия в цикле.

В этой задаче Роботу нужно не только закрасить все отмеченные клетки, но и в конце работы прийти в клетку, отмеченную буквой Б (База).

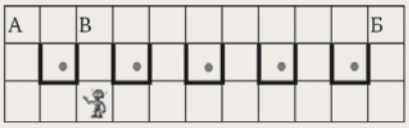


Рис. 31

Поскольку нужно закрасить пять клеток, расположенных в одну линию (хотя и через стенки!), хочется использовать цикл **нц 5 раз**. Но команды в теле цикла будут зависеть от начального положения Робота.

Конечно, можно начать прямо с той клетки, где стоит Робот. Но наиболее выгодная клетка для начала цикла отмечена буквой А. Тогда в теле цикла (на первом шаге) Роботу нужно будет перейти в клетку с буквой В, попутно закрасив клетку внизу, ограниченную стенками. Теперь Робот оказался в правильном исходном положении перед обработкой следующей клетки. Более того, после выполнения пяти шагов цикла он окажется в клетке с буквой Б, то есть вести его туда специально не придется.

Итак, программу можно составить, используя два вспомогательных алгоритма:

- прийти в клетку А (назовем его **Подход**);
- обработать клетку (он будет называться **Клетка**).

Получается такая основная программа:

```
Подход
нц 5 раз
  Клетка
кц
```

Вспомогательные алгоритмы (процедуры) **Подход** и **Клетка** нужно записать после основной программы:

```
алг Подход
нач
  влево; влево; вверх; вверх
кон
алг Клетка
нач
  вправо; вниз
  закрасить
  вверх; вправо
кон
```

При составлении этой программы мы использовали прием “программирование сверху вниз”: разбили задачу на подзадачи, составили основной алгоритм, а затем написали вспомогательные алгоритмы для решения подзадач.

Вложенные циклы

Рассмотрим задачу, в которой Роботу нужно закрасить поле размером 6 × 4 клетки:

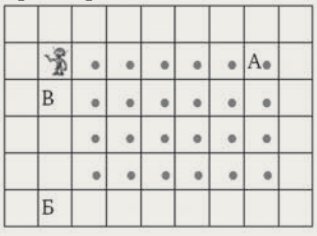


Рис. 32

Давайте сразу будем использовать “программирование сверху вниз” и напишем основную программу. Роботу нужно решать две подзадачи:

• закрасить ряд клеток (назовем эту процедуру **Ряд**);

• перейти к началу следующего ряда, например, из клетки А в клетку В (процедура **Переход**).

В основной программе можно использовать такой цикл:

```
нц 4 раза
  Ряд
  Переход
кц
```

Процедуры **Ряд** и **Переход** получаются очень простые:

```
алг Ряд
нач
  нц 6 раз
    вправо; закрасить
  кц
кон
алг Переход
нач
  вниз
  нц 6 раз влево кц
кон
```

Иногда решения простых подзадач не оформляют в виде коротких процедур, а включают прямо в основную программу. Это немного ускоряет работу программы и сокращает ее текст. Такой прием используют тогда, когда вызов процедуры встречается в программе только в одном месте.

Вот что получится в нашем случае, если встроить текст процедур **Ряд** и **Переход** в основную программу:

```
нц 4 раза
  нц 6 раз
    вправо; закрасить
  кц
вниз
  нц 6 раз влево кц
кц
```

Здесь фоном выделены два цикла, которые оказались внутри другого цикла. Такие циклы называются *вложенными*.

Вложенный цикл — это цикл, находящийся в теле другого цикла.

Но при таком объединении основная программа стала менее понятной, ведь мы ее значительно усложнили. Для того чтобы в программе было легче разобраться, в текст добавляют *комментарии* (пояснения), которые начинаются с символа “|”. Они не обрабатываются компьютером (он никак на них не реагирует) и служат только для того, чтобы человеку было легче понять программу. К нашей программе можно добавить два комментария (они выделены фоном) в тех местах, где начинается решение каждой из подзадач:

```
нц 4 раза
  | обработать ряд
  нц 6 раз
    вправо; закрасить
  кц
  | перейти в начало следующего ряда
  вниз
  нц 6 раз влево кц
кц
```

Комментарий — это пояснение к программе. Комментарии не обрабатываются исполнителем.

Комментарии в сложных программах очень важны, потому что очень часто основная работа программиста состоит в том, чтобы исправлять программы, написанные другими. Для этого нужно разобраться, как они работают, и это значительно легче сделать, если есть комментарии.

Напоследок давайте определим, где остановится Робот после выполнения программы. Вспомним, что в теле цикла решаются две подзадачи — обработка ряда и переход к началу следующего ряда. Это значит, что после выполнения всех шагов цикла Робот остановится в той клетке, откуда он мог бы начать обработку следующего, пятого ряда, если бы его нужно было закрашивать. Эта клетка отмечена буквой Б на *рис.* 32.

Контрольные вопросы

1. Подумайте, как Робот может определить, что он выполнил цикл нужное количество раз?

2. Как по блок-схеме определить, что алгоритм содержит цикл?

3. Когда можно включить текст вспомогательных алгоритмов в основную программу? Расскажите о достоинствах и недостатках такого решения.

4. Юный программист Семен говорит, что никогда не пишет комментариев в программах, потому что это лишняя работа, а он и так все помнит. Согласны ли вы с ним?

Задачи

1. Составьте программы для Робота, решающие следующие задачи (Роботу нужно закрасить все отмеченные клетки и прийти в клетку Б):

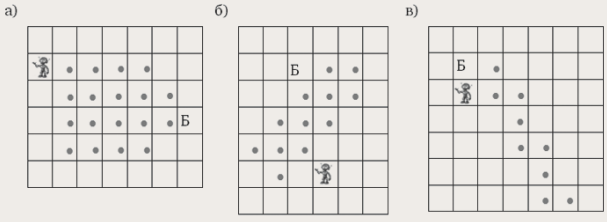


Рис. 33

Используйте циклы.

2. Составьте программы для Робота, решающие следующие задачи:

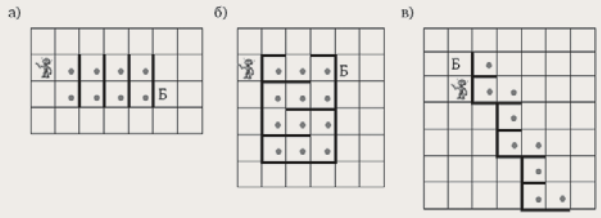


Рис. 34

Используйте циклы.

3. Составьте программы для Робота, решающие следующие задачи (Роботу нужно закрасить все отмеченные клетки и прийти в клетку Б):

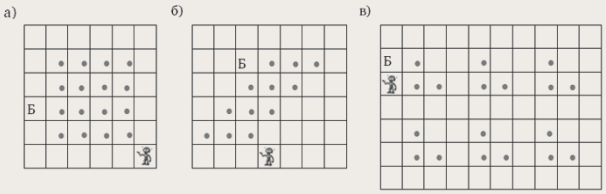


Рис. 35

Используйте вложенные циклы.

Циклы с условием

Ключевые слова:
- условие
- цикл с условием
- логические команды
- обратная связь
- зацикливание
- вложенные циклы

Что такое цикл с условием?

Вспомним алгоритм вычисления остатка от деления, о котором мы говорили ранее:

Вход: два натуральных числа, a и b.

Шаг 1. Если $a < b$, перейти к шагу 4.

Шаг 2. Уменьшить a на величину b.

Шаг 3. Перейти к шагу 1.

Шаг 4. Стоп.

Результат: значение a.

Как показано в разделе “Способы записи алгоритмов” (с. 18), исполнитель несколько раз выполняет шаги 1, 2 и 3 *с разными данными*. Вы уже знаете, что такой алгоритм называется циклическим.

В отличие от циклов, которые мы изучали в предыдущем параграфе, число шагов этого цикла заранее неизвестно, оно зависит от исходных данных. Поэтому мы не можем использовать цикл “N раз”.

Здесь цикл выполняется до тех пор, пока условие $a < b$ не станет истинным (тогда исполнитель перейдет к шагу 4 и работа алгоритма закончится). Можно сказать иначе: цикл выполняется, пока *обратное условие*, $a \ge b$, истинно. Это второй тип цикла, который называется *циклом с условием*.

> **Цикл с условием** — это цикл, который выполняется до тех пор, пока некоторое условие истинно. Количество шагов такого цикла определяется исходными данными.

Пусть Роботу нужно подойти к стене, которая находится слева от него, но неизвестно, на каком расстоянии:

Рис. 36

Если представить себя на месте Робота, можно решить задачу так: делать по одному шагу влево, проверяя, не дошли ли мы до стенки:

Шаг 1. Если слева стена, то перейти к шагу 4.

Шаг 2. Сделать шаг влево.

Шаг 3. Перейти к шагу 1.

Шаг 4. Стоп.

Можно нарисовать блок-схему, соответствующую этому алгоритму:

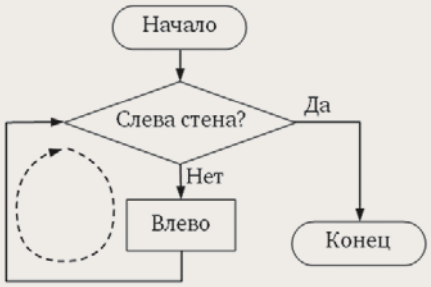


Рис. 37

Мы видим, что на этой блок-схеме есть замкнутый контур (петля), то есть стрелка, идущая к предыдущим командам. Это значит, что алгоритм *циклический*.

Как же исполнитель определит, что слева есть стена? Человек может увидеть ее или почувствовать, дотронувшись рукой. А у Робота тоже есть “органы чувств” — датчики. В систему команд Робота входят *логические команды*:

`сверху стена`	`сверху свободно`
`справа стена`	`справа свободно`
`снизу стена`	`снизу свободно`
`слева стена`	`слева свободно`

Это вопросы, на которые Робот (с помощью своих датчиков) отвечает “да” или “нет”.

> **Логическая команда** — это вопрос, на который исполнитель отвечает “да” или “нет”.

Логические команды также называют *командами обратной связи*, потому что информация передается в обратном направлении — не от управляющего устройства к исполнителю, а наоборот. С помощью этих команд Робот получает информацию об окружающей среде.

> **Обратная связь** — это данные, которые передаются от датчиков к управляющему устройству.

В нашей задаче цикл должен закончиться, когда слева есть стена. Другими словами, цикл работает, пока выполняется обратное условие — слева сво-

бодно. Поэтому решение на алгоритмическом языке запишется так:

```
алг До стены
нач
  нц пока слева свободно
    влево
  кц
кон
```

Здесь ключевые слова **нц** и **кц** обозначают, как и раньше, начало и конец цикла. После слова **пока** записано условие: цикл выполняется, пока это условие истинно (верно). Как только условие становится ложно, работа цикла заканчивается.

Обратите внимание, что использовать здесь цикл "*N* раз" нельзя, потому что мы не знаем точно, сколько шагов нужно сделать Роботу.

Если слева от Робота нет стены, он никогда не остановится, потому что условие **слева свободно** в заголовке цикла никогда не станет ложным. В этом случае говорят, что алгоритм *зациклился*. Чаще всего это означает, что он написан неверно.

Зацикливание — это ситуация, когда условие работы цикла никогда не становится ложным и цикл выполняется бесконечно.

В следующей задаче Роботу нужно закрасить ряд клеток, начиная от своего начального положения до стены справа:

Рис. 38

Здесь на каждом шаге цикла нужно выполнить две команды: **влево** и **закрасить**. Решение нашей задачи выглядит так:

```
нц пока слева свободно
  влево
  закрасить
кц
```

Заметим, что эта программа (как и предыдущая!) работает не для одной начальной обстановки, а для множества, потому что расстояние от Робота до стены справа может быть любым, Робот все равно закрасит все клетки.

Вложенные циклы

Циклы с условием можно вкладывать в другие циклы, так же, как и циклы "*N* раз". Пусть Роботу нужно закрасить четыре ряда клеток между двумя стенками, причем расстояние между этими стенками неизвестно.

Рис. 39

Выделим две подзадачи:

- закрашивание горизонтального ряда клеток (процедура **Ряд**);
- переход в начало следующего ряда (процедура **Переход**).

Тогда основная программа получается такая же, как и в решении задачи на *рис.* 32:

```
нц 4 раза
  Ряд
  Переход
кц
```

Однако здесь обе процедуры будут другие. Так как расстояние между стенками неизвестно, мы будем использовать циклы с условием.

Очевидно, что процедура "Ряд" будет содержать такой цикл:

```
нц пока справа свободно
  вправо
  закрасить
кц
```

Но при этом в каждом ряду не закрашивается клетка, находящаяся около левой стены (например, клетка, в которой стоит Робот в самом начале). Поэтому перед этим циклом нужно добавить еще одну команду — **закрасить**.

Процедура **Переход** включает движение влево до стенки (**пока слева свободно**) и шаг вниз, в начало следующего ряда:

```
нц пока слева свободно
  вправо
кц
вниз
```

Вместо того чтобы оформлять отдельные процедуры, мы можем включить все входящие в них команды в основную программу:

```
нц 4 раза
    | закрасить ряд клеток
  закрасить
  нц пока справа свободно
    вправо; закрасить
  кц
    | перейти в начало следующего ряда
  нц пока слева свободно
    влево
  кц
  вниз
кц
```

Мы получили вложенные циклы: внутри основного цикла "*N* раз" находятся два цикла с условием. Начало каждой из подзадач помечено комментарием.

Теперь еще усложним задачу — пусть рядов будет не четыре, а неизвестно сколько, причем эти ряды заканчиваются там, где заканчивается стена слева. Обратите внимание, что одна из этих задач — это задача с четырьмя рядами, которую мы рассматривали до этого.

Как изменится решение? Понятно, что нужно заменить цикл **нц 4 раза** на цикл с условием.

Определяя условие цикла, мы должны понять, когда его нужно остановить. В нашем случае цикл должен закончиться, когда закончится стенка слева. Это значит, что нужно продолжать цикл, пока выполняется *обратное условие*, то есть слева есть стена. Итак, просто заменяем одну строчку — заголовок основного (внешнего) цикла:

```
нц пока слева стена
  ...
кц
```

Вместо многоточия нужно добавить все команды, которые были внутри основного цикла в решении предыдущей задачи.

Контрольные вопросы

1. Объясните, в каких случаях нужно использовать цикл **N раз**, а в каких — цикл с условием.
2. Какая серьезная ошибка может встретиться при использовании цикла с условием?
3. Сработает ли алгоритм решения задачи на *рис.* 36, если Робот в самом начале уже стоит у стены?
4. Как по блок-схеме алгоритма определить, что он циклический?
5. Можно ли управлять исполнителем без обратной связи? В чем недостатки такого метода?
6. Цикл должен завершиться, когда кончится стена справа от Робота. Как записать заголовок такого цикла?
7. Можно ли включать циклы **N раз** в тело цикла с условием?

Задачи

1. Будет ли верно работать программа для решения задачи на *рис.* 39, если переход в начало следующего ряда выполнить так:

```
вниз
нц пока слева свободно
  влево
кц
```

Ответ обоснуйте.

2. Переведите Робота в клетку Б, считая, что все длины стенок неизвестны. Все отмеченные клетки нужно закрасить.

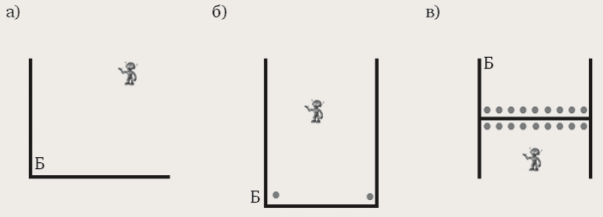


Рис. 40

3. Напишите программу, выполнив которую Робот закрасит все отмеченные клетки. Длины стенок считайте неизвестными.

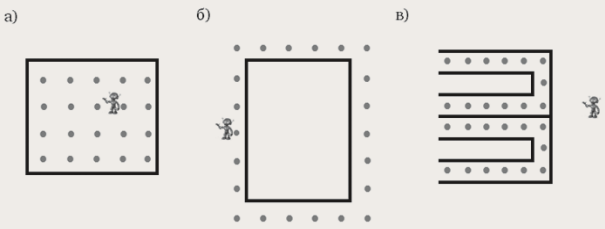


Рис. 41

Тема для сообщения:
“Сравнение двух видов циклов”

Переменные

Ключевые слова:
- переменная
- присваивание
- объявление переменной
- процедура
- параметр

Зачем нужны переменные?
Попробуем решить такую задачу:

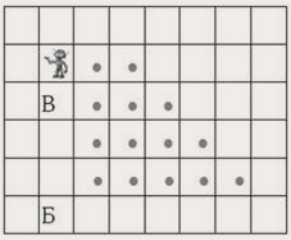


Рис. 42

Эта задача очень похожа на задачу на *рис.* 32. Но есть одно отличие: длины всех рядов тут разные: первый горизонтальный ряд состоит из двух клеток, а длина каждого следующего на одну клетку больше, чем предыдущего.

Обозначим длину очередного ряда буквой *N*. Тогда алгоритм закраски очередного ряда и возврата к началу следующего ряда выглядит так:

```
нц N раз
  вправо; закрасить
кц
вниз
нц N раз влево кц
```

Величина *N* должна изменяться во время работы программы, она называется *переменной*. С этим понятием мы уже знакомы. До начала основного цикла нужно присвоить этой переменной значение 2:

```
N := 2
```

Символ “:=” — это команда (*оператор*) присваивания, она присваивает переменной **N** значение, которое записано в правой части.

После обработки очередного ряда и возврата назад нужно увеличить значение *N* на 1, то есть заменить *N* на *N* + 1. Для этого тоже используется оператор присваивания:

```
N := N + 1
```

В результате получаем такую программу с вложенным циклом:

```
цел N
N := 2
нц 4 раза
    | закраска ряда длиной N
  нц N раз
    вправо; закрасить
  кц
    | переход к следующему ряду
  вниз
  нц N раз влево кц
  | увеличение длины ряда
  N := N + 1
кц
```

В ней есть одна неизвестная строка:

```
цел N
```

Это так называемое *объявление переменной*. Этой командой мы сказали, что в программе будет использована переменная с именем **N** и она может принимать только *целые* значения (ключевое слово **цел**).

Зачем нужно объявление переменной? При объявлении мы определяем

- какие данные можно хранить в этой переменной (целое число, дробное число, строка символов и др.);
- какие операции можно выполнять с переменной;
- сколько места в памяти компьютера нужно выделить для хранения значения.

Отметим, что в некоторых языках программирования, например в языке Python, объявлять переменные не нужно. С одной стороны, это облегчает составление программы, но с другой — может привести к труднообнаруживаемым ошибкам.

Процедуры с параметрами

Ранее мы уже познакомились с процедурами — вспомогательными алгоритмами, которые используются для того, чтобы не повторять в программе несколько раз одинаковые команды. Но до этого мы писали процедуры, каждая из которых всегда делала одно и то же. Часто бывает нужно несколько раз выполнить похожие, но чем-то отличающиеся действия. В этом случае тоже можно составить процедуру, но более сложного типа.

Рассмотрим такую задачу:

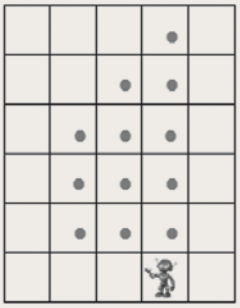


Рис. 43

Если бы все ряды были одинаковой длины (скажем, четыре клетки), мы могли бы написать процедуру (вспомогательный алгоритм) так:

```
алг Ряд
нач
  нц 4 раза
    вверх; закрасить
  кц
кон
```

Но длина ряда меняется, поэтому нужно сделать ее *переменной*, обозначив каким-то именем. Кроме того, надо как-то передать значение длины ряда из основной программы в процедуру (иначе как исполнитель узнает, какая длина ряда нужна именно сейчас?). При вызове мы хотели бы записать, например, так:

```
Ряд ( 4 )
```

надеясь, что исполнитель закрасит ряд из четырех клеток. Так действительно можно делать, но для этого в заголовке процедуры нужно указать, что она принимает *параметр* — значение, от которого зависит работа алгоритма.

В нашем случае параметр — это количество клеток, которые нужно закрасить. Если обозначить его именем *N*, получается такая процедура:

```
алг Ряд ( цел N )
нач
  нц N раз
    вверх; закрасить
  кц
кон
```

Обратите внимание, что в заголовке процедуры мы записали

```
Ряд ( цел N )
```

Это означает, что при вызове процедуры ей нужно передать (в скобках) целое число. Это значение будет записано во внутреннюю (или *локальную*) переменную процедуры с именем **N**.

> **Параметры** — это данные, которые передаются в процедуру. Каждый параметр имеет имя и тип.

Теперь можно полностью написать основную программу, которая решает задачу на *рис.* 43 с использованием процедуры **Ряд**:

```
алг Узоры
нач
  Ряд ( 5 )
  нц 5 раз вниз кц
  влево
  Ряд ( 4 )
  нц 4 раза вниз кц
  влево
  Ряд ( 3 )
кон
```

Попробуйте разобраться в ее работе самостоятельно. Не забудьте, что процедуру **Ряд** нужно записать ниже основного алгоритма.

Можно заметить, что после каждого выполнения процедуры **Ряд** исполнитель идет назад на столь-

ко шагов, на сколько он прошел вперед (на длину ряда), а затем делает шаг влево. Поэтому можно сократить основную программу, добавив в процедуру еще один цикл (“обратный ход”) и команду **влево**:

```
алг Ряд2 ( цел N )
нач
  нц N раз
    вверх; закрасить
  кц
  нц N раз вниз кц
  влево
кон
```

Теперь основная программа выглядит так:

```
алг Узоры2
нач
  Ряд2( 5 )
  Ряд2( 4 )
  Ряд2( 3 )
кон
```

Разница только в том, что в этом случае после выполнения алгоритма Робот остановится в другой клетке (подумайте, в какой?).

Числа, которые записаны в скобках при вызове процедуры **Ряд**, — это фактические значения параметров, которые также называют *аргументами*.

Контрольные вопросы

1. Можно ли решить задачу на *рис.* 42, не используя переменные?
2. Как, на ваш взгляд, компьютер может выполнить цикл **N раз** (как он запомнит, сколько раз уже выполнил тело цикла)?
3. Что означает запись **N := N + N**?
4. Зачем нужно объявлять переменные?
5. Сколько различных процедур без параметров мы могли бы написать для решения задачи на *рис.* 43?
6. Что значит “передать данные в процедуру”? Зачем это нужно?
7. Как по записи процедуры выяснить, принимает ли она параметры, и если да, то какие именно?
8. Объясните разницу между результатами работы алгоритмов **Узоры** и **Узоры2**.

Задачи

1. Определите, какую область закрашивает Робот, выполнив эти программы:

а)

```
цел N
N := 5
нц 3 раза
  нц N раз
    закрасить; вправо
  кц
  нц N - 1 раз влево  кц
  вниз
  N := N - 2
кц
```

б)

```
цел N
N := 1
нц 3 раза
  нц N раз
    закрасить; вправо
  кц
  нц N + 1 раз влево кц
  вниз
  N := N + 2
кц
```

Напишите программу, выполнив которую Робот закрашивает отмеченные клетки:

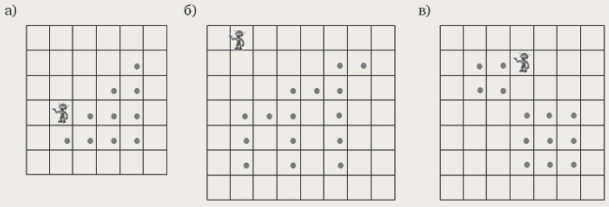


Рис. 44

Составьте два варианта каждой программы: один с вложенными циклами, второй — с процедурой, которая вызывается в цикле.

Разветвляющиеся алгоритмы

Ключевые слова:

- условие
- ветвление
- полная форма ветвления
- неполная форма ветвления
- разветвляющийся алгоритм
- вложенные ветвления

Что такое ветвление?

Очень часто в программе не обойтись линейными и циклическими алгоритмами, потому что исполнитель должен выполнять различные действия при разных исходных данных. Представим себе, что робот измеряет характеристики окружающей среды (например, радиацию) и в зависимости от результатов измерений либо выдает сигнал об опасности (если она больше допустимой), либо сообщает, что все нормально.

Пусть допустимый уровень радиации — 100 микрорентген в час[6]. Тогда алгоритм в словесной форме можно записать так:

Вход: *R*

Шаг 1. Если *R* > 100, перейти к шагу 4.

Шаг 2. Присвоить *M* значение “Все нормально!”.

Шаг 3. Перейти к шагу 5.

Шаг 4. Присвоить *M* значение “Тревога!”.

Шаг 5. Стоп.

Результат: *M*.

Здесь через *R* обозначен измеренный уровень радиации (в микрорентгенах в час), а *M* — это сообщение-результат.

[6] Это в 10 раз больше безопасного уровня, который составляет примерно 10–12 микрорентген в час.

Обратите внимание на две особенности этого алгоритма:

1) последовательность выполняемых команд зависит от исходных данных (результата измерения радиации);

2) в словесной форме запись этого простого алгоритма получилась довольно запутанной, и понять ее непросто.

Для представления алгоритма в виде блок-схемы нам понадобится блок “условие”:

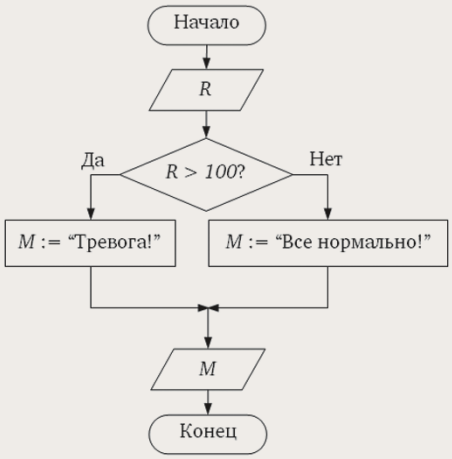


Рис. 45

Алгоритмы, в которых действия исполнителя зависят от исходных данных, называются *разветвляющимися*. На блок-схеме маршрут “расщепляется” на две ветки, появляются два пути выполнения алгоритма.

Разветвляющийся алгоритм — это алгоритм, в котором последовательность действий изменяется в зависимости от выполнения некоторых условий.

Разветвляющиеся алгоритмы позволяют исполнителю реагировать на изменение окружающей обстановки. Это значит, что он может решать более широкий круг задач.

Неполная форма ветвления

Обычно исполнитель должен выводить какие-то сообщения только в случае аварийной ситуации, а при нормальной работе никаких действий от него не требуется. Если бы ваш компьютер или смартфон каждую секунду выводил сообщение “Все нормально!”, это мешало бы работать с ним. Лишнюю информацию, которая только отвлекает человека от решения своих задач, часто называют “информационным шумом”. Желательно, чтобы такого шума было как можно меньше. Например, алгоритм проверки уровня радиации можно изменить так:

Вход: *R*

Шаг 1. Если $R \le 100$, перейти к шагу 4.

Шаг 2. Присвоить *M* значение “Тревога!”.

Шаг 3. Вывести *M*.

Шаг 4. Стоп.

Изменится и блок-схема:

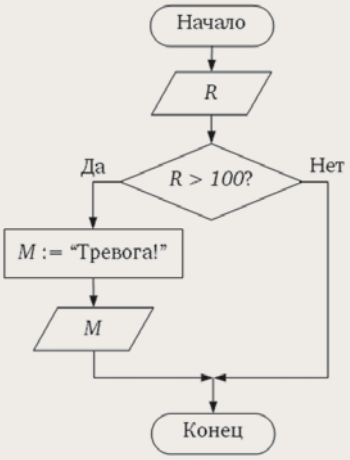


Рис. 46

Итак, если условие не выполняется, то исполнителю ничего делать не нужно, ветка “нет” — пустая! Такое ветвление называется *неполным*, а вариант, который мы использовали сначала, — *полной формой* ветвления.

Неполная форма ветвления отличается тем, что одна из веток на блок-схеме — пустая.

Управление Роботом

Ветвления — это очень мощное средство, позволяющее исполнителю ориентироваться в незнакомой (или не полностью известной) обстановке. С помощью одного и того же алгоритма с ветвлениями исполнитель может решать различные задачи.

Рассмотрим две задачи для исполнителя Робот, в которых требуется перевести его в соседнюю клетку, отмеченную буквой Б:

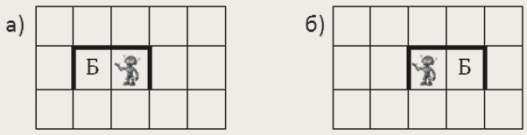


Рис. 47

Каждая из этих задач решается простым алгоритмом:

```
алг Случай а
нач
  влево
кон

алг Случай б
нач
  вправо
кон
```

Они отличаются единственной командой.

А теперь потребуем, чтобы обе задачи решались с помощью одного алгоритма для Робота. Для этого исполнителю нужно как-то решить, какую команду применить: **`влево`** или **`вправо`**.

В случае *а* справа от Робота будет стенка, поэтому туда идти нельзя, нужно идти влево. Если же справа стенки нет, нужно идти вправо (вариант *б*). Как вы уже знаете, в СКИ Робота есть логические команды, которые позволяют получать информацию от датчиков. В нашем случае нужна команда **справа стена**. Получается такая блок-схема алгоритма:

Рис. 48

А вот так запишется соответствующая программа для Робота на алгоритмическом языке:

```
алг В соседнюю клетку
нач
  если справа стена то
    влево
  иначе
    вправо
  все
кон
```

После слова **если** записано условие (вопрос). Если условие **справа стена** выполняется (то есть истинно), выполняются все команды между ключевыми словами **то** и **иначе**, а если не выполняется (ложно), то исполнитель выполняет все команды между словами **иначе** и **все**.

Еще раз подчеркнем, что только что построенный разветвляющийся алгоритм позволяет решать *обе* задачи на *рис.* 47.

Пусть теперь требуется перевести Робота в клетку Б, при этом сначала Робот может находиться как в клетке А, так и в клетке Б:

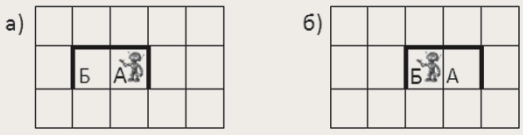


Рис. 49

В случае *а* Робот определит, что слева свободно, и передвинется влево. А в случае *б* исполнитель уже стоит в нужной клетке, поэтому ничего делать не надо. Наверное, вы догадались, что здесь нам нужна неполная форма ветвления:

```
алг К левой стенке
нач
  если слева свободно то
    влево
  все
кон
```

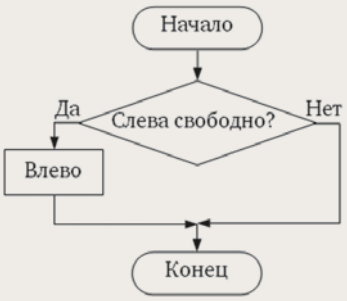


Рис. 50

Обратите внимание, что в записи программы на алгоритмическом языке нет слова **иначе**. Для ветвления в неполной форме второй блок команд (между словами **иначе** и **все**) не нужен, ведь он пустой! И ветка **нет** на блок-схеме тоже пустая.

Вложенные ветвления

Ветвления, как и циклы, могут быть вложенными. Составим алгоритм, который определяет знак числа. Результат его работы для некоторого числа x равен

- 1, если число x положительное;
- –1, если число x отрицательное;
- 0, если число x равно 0.

Вот блок-схема этого алгоритма:

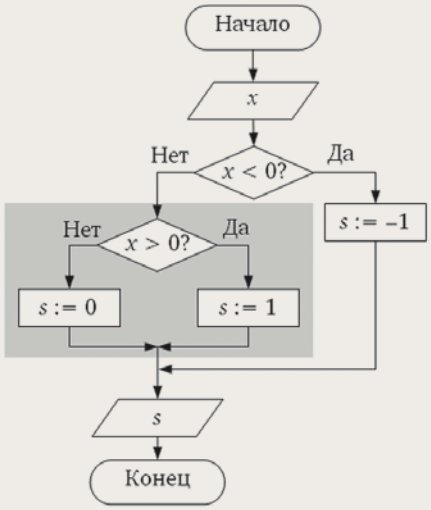


Рис. 51

Сначала проверяем условие $x < 0$. Если оно истинно (верно), то сразу определяем результат — он равен “–1”, записываем это значение в переменную s. Если же условие $x < 0$ ложно (неверно), то остаются еще два варианта, между которыми нужно выбрать. Поэтому внутри ветки “нет” должно быть еще одно ветвление (оно выделено на блок-схеме серым фоном).

Контрольные вопросы

1. Как человек использует разветвляющиеся алгоритмы в жизни? Приведите примеры.
2. Какие возможности появляются при использовании разветвляющихся алгоритмов? В каких задачах без них обойтись невозможно?

3. Как заменить полную форму ветвления на две неполных? Когда два ветвления в неполной форме можно заменить на одно ветвление в полной форме?

Задачи

1. Нарисуйте блок-схему алгоритма, который вычисляет модуль исходного числа.
2. Нарисуйте блок-схему алгоритма, который позволяет найти:
а) наибольшее из двух чисел;
б) наибольшее из четырех чисел;
в) наибольшее из трех чисел;
г) *наибольшее из пяти чисел.

Попробуйте найти решения этих задач, использующие только ветвления в неполной форме.

3. Нарисуйте блок-схему алгоритма, который позволяет расставить три числа в порядке возрастания (неубывания).
4. Нарисуйте блок-схему алгоритма, который позволяет определить, четное число x или нечетное. Нужно вывести ответ “да” или “нет”.
5. Нарисуйте блок-схему алгоритма, который позволяет определить, принадлежит ли значение x отрезку $[a,b]$. Нужно вывести ответ “да” или “нет”.
6. Нарисуйте блок-схему алгоритма, который позволяет определить, можно ли построить треугольник с заданными сторонами a, b и c. Нужно вывести ответ “да” или “нет”.
7. Нарисуйте блок-схему алгоритма, который выполняет деление одного числа на другое, обрабатывая ошибку “деление на ноль”.
8. Определите, что делают эти алгоритмы:

а)

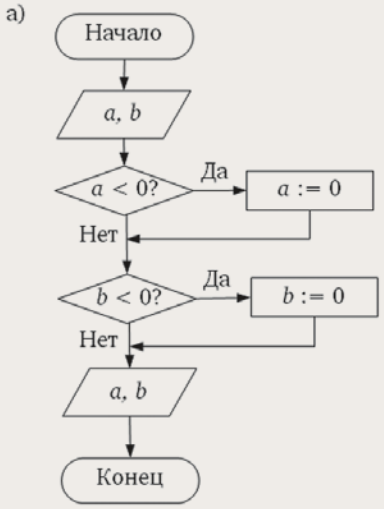


Рис. 52а

б)

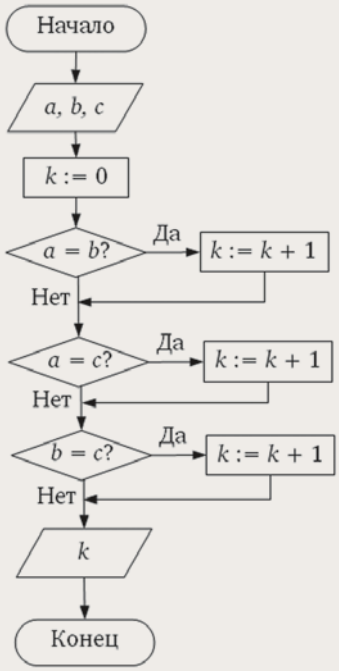


Рис. 52б

Тема для сообщения:
“Разветвляющиеся алгоритмы вокруг нас”

Ветвления и циклы

Ключевые слова:
- ветвление
- цикл
- следование
- алгоритм Евклида
- диалоговая программа

Пример задачи

Любая достаточно сложная программа содержит как циклы, так и ветвления. Пусть Роботу нужно закрасить ряд клеток до клетки, отмеченной буквой Б:

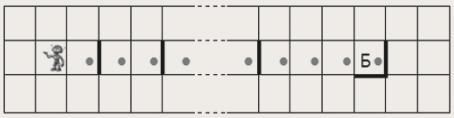


Рис. 53

На его пути в некоторых местах есть стенки, которые можно обойти сверху. Если стенки нет, Робот должен экономить энергию и идти прямо в соседнюю клетку.

Сначала определяем условие остановки Робота в клетке с буквой Б. Там снизу есть стенка, а в других клетках на пути — нет. Поэтому основной цикл можно написать так:

```
нц пока снизу свободно
  ...
кц
```

где многоточие означает какие-то команды.

Что делает Робот на каждом шаге цикла? Он переходит в соседнюю клетку и закрашивает ее. Пере-

ход может быть выполнен двумя способами: если справа свободно, то Робот просто выполняет шаг вправо, а если нет — обходит стенку сверху. Получается такое решение:

```
нц пока снизу свободно
  если справа свободно то
    вправо
  иначе
    вверх; вправо; вниз
  все
  закрасить
кц
```

На блок-схеме (*рис.* 54) хорошо видно, что алгоритм внутри цикла содержит ветвление (оно выделено серым фоном). Кроме того, внутри ветки “да” записана линейная последовательность из трех команд.

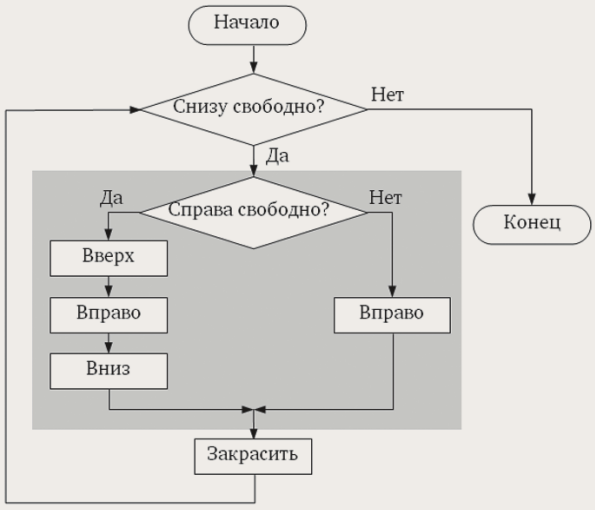


Рис. 54

Исполнитель Раздвоитель

Рассмотрим еще одного исполнителя, которого зовут Раздвоитель. СКИ Раздвоитель содержит две команды:

```
1. вычти 1
2. раздели на 2
```

Можно сказать, что он выполняет обратные действия в сравнении с Удвоителем. Обратите внимание, что команда 2 применима не всегда: ее можно использовать только для тех чисел, которые делятся на 2.

Если Раздвоителю задано натуральное начальное число, то любая из двух команд приводит к уменьшению числа (пока число не станет меньше или равно нулю!). Поэтому он может получить только те числа, которые меньше начального.

Задача. Напишите алгоритм для Раздвоителя, который любое начальное натуральное число превращает в ноль.

Решение. Прежде всего заметим, что таких алгоритмов может быть много. Например, всегда можно просто вычитать единицу (выполнять команду 1), пока не получится ноль:

```
нц пока не ноль
  вычти 1
кц
```

Давайте подумаем, нельзя ли решить задачу быстрее. Чтобы уменьшить время выполнения алгоритма, нужно на каждом шаге уменьшать число как можно быстрее. Команда 2 (**раздели на 2**) быстрее приближает нас к цели, поэтому использовать ее выгоднее, чем команду 1. Но эту команду можно применить не всегда, а только для четных чисел (которые делятся на 2). Следовательно, лучшим решением будет такое:

```
нц пока не ноль
  если четное то
    раздели на 2
  иначе
    вычти 1
  все
кц
```

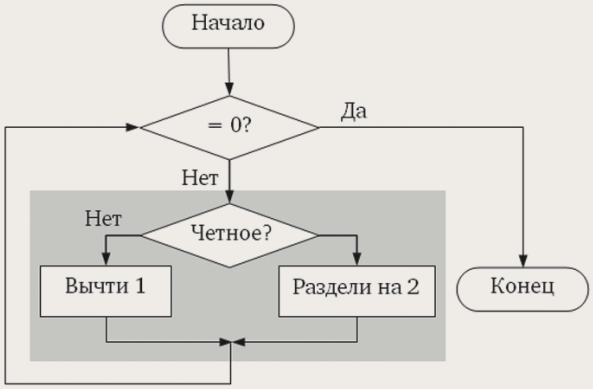


Рис. 55

Этот алгоритм содержит ветвление внутри цикла (оно выделено фоном). Например, для начального числа 20 алгоритм решает задачу не за 20 шагов, как предыдущий, а всего за 6:

Рис. 56

Можно сказать, что второй алгоритм лучше первого, потому что требует меньшего количества действий для получения правильного результата.

Алгоритм Евклида

Древнегреческий математик Евклид, живший в III веке до нашей эры, придумал замечательный алгоритм для вычисления наибольшего общего делителя (НОД) двух натуральных чисел. Этот алгоритм и сейчас используется, например, в задачах шифрования. Напомним, что НОД — это наибольшее число, на которое два заданных числа делятся без остатка.

Алгоритм Евклида для натуральных чисел: заменять бóльшее из двух заданных чисел на их разность до тех пор, пока они не станут равны. Полученное число и есть их НОД.

Алгоритм Евклида можно записать более строго в виде последовательности шагов:

Вход: натуральные числа a, b.

Шаг 1. Если $a = b$, то перейти к шагу 7.

Шаг 2. Если $a < b$, то перейти к шагу 5.

Шаг 3. Заменить a на $a - b$.

Шаг 4. Перейти к шагу 1.

Шаг 5. Заменить b на $b - a$.

Шаг 6. Перейти к шагу 1.

Шаг 7. Стоп.

Результат: a.

Выполним ручную прокрутку алгоритма: найдем НОД чисел $a = 35$ и $b = 14$. Для этого требуется три шага цикла (то есть команды, заключенные в цикл, — шаги 2–5 — выполняются три раза). На каждом шаге заменяем бóльшее число на разность большего и меньшего:

Действие		a	b
		35	14
a = b?	нет		
a < b?	нет		
a := a - b		21	
a = b?	нет		
a < b?	нет		
a := a - b		7	
a = b?	нет		
a < b?	да		
b := b - a			7
a = b?	да		

После этого алгоритм останавливается, потому что a и b стали равны, условие, проверяемое на шаге 1, истинно, и происходит переход к шагу 7 (на команду Стоп).

Теперь составим блок-схему этого алгоритма. Здесь основная выполняемая операция (шаги 2–6) — это замена бóльшего из двух чисел на их разность. Получается такая схема:

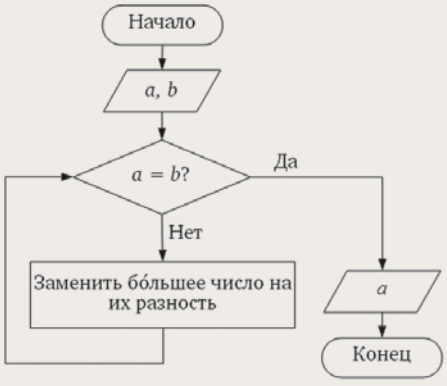


Рис. 57

Согласитесь, что в таком виде алгоритм выглядит понятнее, чем при пошаговом описании.

Обратим внимание, что под блоком “заменить бóльшее число на их разность” скрывается вспомогательный разветвляющийся алгоритм:

Вход: два натуральных числа, a и b.

Шаг 1. Если $a < b$, то перейти к шагу 4.

Шаг 2. Заменить a на $a - b$.

Шаг 3. Перейти к шагу 5.

Шаг 4. Заменить b на $b - a$.

Шаг 5. Стоп.

Результат: значения a, b.

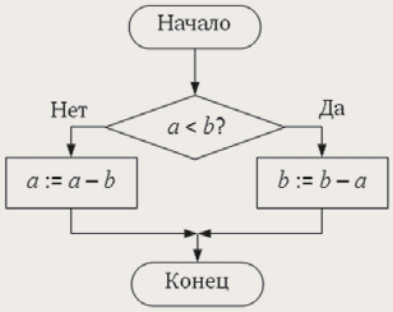


Рис. 58

Блок-схему этого алгоритма можно добавить в основную блок-схему вместо блока “заменить бóльшее число на их разность”:

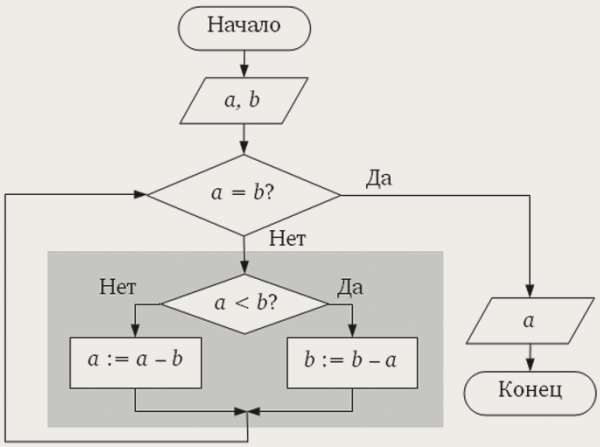


Рис. 59

Таким образом, мы “собрали” всю блок-схему. Обратите внимание, как мы это сделали:

1) сначала выяснили, что алгоритм циклический, и цикл заканчивается при условии $a = b$; это позволило нарисовать схему на *рис.* 57, в которой был один нераскрытый блок — тело цикла;

2) построили отдельно алгоритм для выполнения операций в теле цикла (*рис.* 58);

3) объединили эти два алгоритма в общую блок-схему.

Наверное, вы догадались, что здесь использовался метод проектирования “сверху вниз”, хотя мы не оформляли тело цикла как отдельную процедуру.

Блок-схема на *рис.* 59 описывает достаточно простой алгоритм и уже выглядит довольно запутанно. Поэтому для описания сложных алгоритмов блок-схемы не используют — алгоритм записывают сразу на языке программирования или на *псевдокоде* — смеси естественного языка (русского, английского) и какого-нибудь языка программирова-

ния. Программу на алгоритмическом языке можно записать так:

```
алг Евклид
нач
  цел a, b
  ввод a, b
  нц пока a <> b
    если a < b то
      b := b - a
    иначе
      a := a - b
    все
  кц
  вывод a
кон
```

Она полностью соответствует блок-схеме на *рис.* 59. Условие `a <> b` обозначает “*a* не равно *b*”, оно записывается с помощью двух знаков, “**<**” и “**>**”, без пробела между ними.

В этой программе вы увидели две новые команды школьного алгоритмического языка, **ввод** и **вывод**. Они используются для ввода значений с клавиатуры и вывода результатов на экран, то есть для диалога компьютера с человеком. Такие программы называются *диалоговыми*.

> **Диалоговая программа** — это программа, в которой исходные данные вводятся человеком с клавиатуры, а результаты работы выводятся на экран.

Из чего строятся все алгоритмы?

Как мы увидели, в алгоритмах можно использовать три основных конструкции:

- *следование* (последовательное выполнение, линейный алгоритм);
- *ветвление* (выбор одного из двух вариантов в зависимости от выполнения условия);
- *повторение* (цикл).

В середине XX века было строго доказано, что этих конструкций достаточно для того, чтобы записать любой алгоритм, который только можно придумать.

> Алгоритм решения любой задачи можно составить с помощью трех конструкций — **следования**, **ветвления** и **цикла**. Их называют базовыми алгоритмическими структурами.

Контрольные вопросы

1. Найдите в алгоритмах этого параграфа следования, ветвления и циклы.
2. Подумайте, как можно найти НОД двух натуральных чисел, не используя алгоритм Евклида? Какой метод лучше? Почему?
3. Как вы думаете, в чем достоинства диалоговых программ?

Задачи

1. Придумайте алгоритмы, в которых
а) ветвление находится в теле цикла;
б) цикл находится внутри одной из веток ветвления.
2. Нарисуйте блок-схему алгоритма, который вычисляет остаток от деления одного натурального числа на другое, обрабатывая ошибку “деление на ноль”. Запишите этот же алгоритм в пошаговой словесной форме.
3. Напишите программу для Робота, выполнив которую он закрасит все тупики и остановится в конце коридора в клетке Б (длина коридора неизвестна):

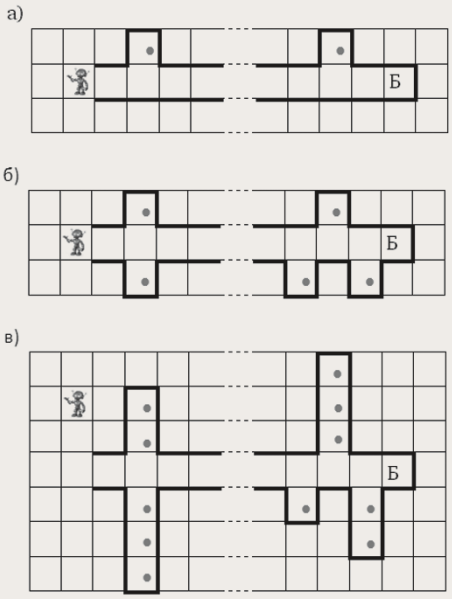


Рис. 60

4. Составьте алгоритм для Раздвоителя, с помощью которого из любого натурального числа $a > 5$ получается число 5 за наименьшее число шагов. Запишите его в словесной форме и в виде блок-схемы.
5. У исполнителя Калькулятор есть команды

```
1. вычти 1
2. раздели на 2
3. раздели на 5
```

Составьте алгоритм для этого исполнителя, с помощью которого из любого натурального числа $a > 3$ получается число 3 за наименьшее число шагов.
6. Напишите программу, которая вычисляет НОД двух целых чисел с помощью алгоритма Евклида. С ее помощью вычислите
а) НОД чисел 64 168 и 82 678; (1234)
б) НОД чисел 358 853 и 691 042; (1111)
в) НОД чисел 6 365 133 и 11 494 962; (171)
г) НОД чисел 17 905 514 и 23 108 855; (3421)
д) НОД чисел 549 868 978 и 298 294 835. (17)

[7] Диалог — это общение двух объектов, например, компьютера и человека.

7. Нарисуйте блок-схему улучшенного алгоритма Евклида:

Вход: натуральные числа a, b.

Шаг 1. Если $a = 0$ или $b = 0$, то перейти к шагу 7.

Шаг 2. Если $a < b$, то перейти к шагу 5.

Шаг 3. Заменить a на a mod b (остаток от деления a на b).

Шаг 4. Перейти к шагу 1.

Шаг 5. Заменить b на b mod a (остаток от деления a на b).

Шаг 6. Перейти к шагу 1.

Шаг 7. Стоп.

Результат: $a + b$.

Приведите пример, когда улучшенный алгоритм Евклида работает быстрее алгоритма, приведенного в тексте параграфа.

8. Попробуйте придумать алгоритм, который нельзя записать с помощью циклов, ветвления и следования. Получилось ли у вас?

Тема для сообщения:

“Алгоритм Евклида”

Графические программы

Ключевые слова:

- графический режим
- исполнитель Рисователь
- холст
- пиксель
- координаты
- оси координат
- переменная
- цикл с переменной

Что такое графическая программа?

Много лет назад человек общался с компьютером только в текстовом режиме — вводил с клавиатуры числа и символы, и получал ответ компьютера тоже в форме текста. Мониторы работали в текстовом режиме. Это значит, что весь экран был разбит на прямоугольники-символы (например, 25 строк по 80 символов в каждой) и программа могла управлять каждым символом. При этом было очень сложно что-то нарисовать на экране, потому что рисунок нужно было тоже строить из символов.

Текстовый режим используется и сейчас, потому что в некоторых задачах (например, при настройке компьютера) его возможностей вполне достаточно. Такие программы называются *консольными* (от английского слова *console* — набор устройств для управления компьютером).

Современные мониторы работают в *графическом режиме*. Это значит, что программа может управлять отдельными точками на экране монитора и строить диаграммы, графики, а также выводить на экран изображения, например, фотографии. Конечно, вы уже использовали графические редакторы и создавали рисунки вручную. Теперь мы научимся составлять программы, которые рисуют без нашего участия, автоматически. Для этого мы будем использовать исполнителя Рисователь, который входит в систему КуМир.

Исполнитель Рисователь

Исполнитель Рисователь рисует на *холсте*. Так называется прямоугольная область экрана, доступная для рисования. Если исполнитель попытается нарисовать что-то за пределами холста, эта часть рисунка будет потеряна.

Холст — это прямоугольник, состоящий из отдельных пикселей, то есть *растровый рисунок*. Каждый пиксель имеет две координаты:

x — расстояние от пикселя до левой границы холста;

y — расстояние от пикселя до верхней границы холста.

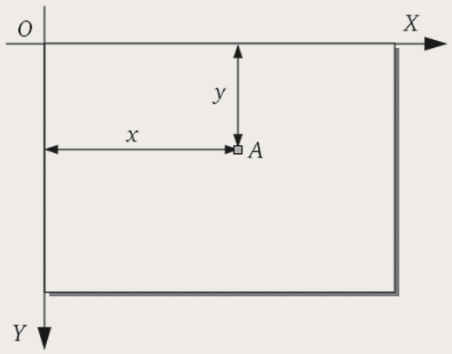


Рис. 61

Можно сказать, что Рисователь использует прямоугольную систему координат, в которой ось X направлена вдоль верхней границы холста вправо, а ось Y — вдоль нижней границы холста **вниз**. Обратите внимание на непривычное направление оси Y — в математике она обычно направлена вертикально вверх.

Пиксель в левом верхнем углу холста имеет координаты (0,0). Нумерация с нуля часто используется в программировании. В нашем случае это значит, что пиксель находится одновременно на левой границе и на верхней границе холста (оба расстояния до границ равны нулю).

Напишем первую программу для Рисователя: создадим холст размером 500 × 400 пикселей (ширина — 500 пикселей, высота — 400 пикселей) и зальем его белым цветом:

```
использовать Рисователь
алг Холст
нач
  новый лист( 500, 400, белый )
кон
```

В первой строке мы подключаем исполнителя Рисователь, а единственная строка основной программы создает нужный холст, вызывая команду **новый лист**, входящую в СКИ Рисователя:

```
новый лист( 500, 400, белый )
```

Первое число означает ширину холста в пикселях, второе — высоту, а третье — название цвета, которым заливается холст.

Вот все простые цвета, которые можно использовать в КуМире:

белый, **черный**, **серый**, **фиолетовый**, **синий**, **голубой**, **зеленый**, **желтый**, **оранжевый**, **красный**.

Управление пикселями

Мы уже говорили, что в графическом режиме программа может управлять отдельно каждым пикселем. Этого достаточно, чтобы нарисовать любой рисунок, даже самый сложный.

У Рисователя есть команда **пиксель**, которая изменяет цвет пикселя с заданными координатами. Например, закрасить синим цветом пиксель с координатами[7] (10,20) можно так:

```
пиксель( 10, 20, синий )
```

Данные в скобках, которые передаются процедуре **пиксель**, называются *аргументами*. Первые два аргумента — это *x*-координата и *y*-координата пикселя, а третий — нужный цвет.

Теперь нарисуем горизонтальную синюю линию из точки (10,20) в точку (15,20). Линия на рисунке состоит из отдельных пикселей, поэтому нужно закрасить синим цветом пиксели с координатами (10,20), (11,20), (12,20), (13,20), (14,20) и (15,20). Программа может выглядеть так:

```
пиксель( 10, 20, синий )
пиксель( 11, 20, синий )
пиксель( 12, 20, синий )
пиксель( 13, 20, синий )
пиксель( 14, 20, синий )
пиксель( 15, 20, синий )
```

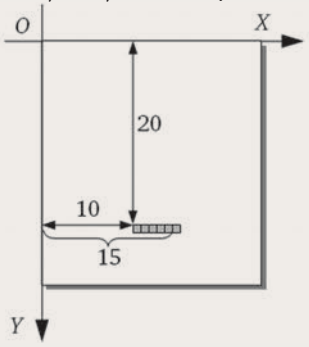


Рис. 62

Чтобы не писать много похожих команд, мы попытаемся построить цикл. Заметим, что в этих командах различается только первая координата (она выделена фоном). То есть это *переменная* величина (вспомните, что такое переменная в программировании). Обозначим первую координату пикселя именем **x**. Тогда нам нужно шесть раз выполнить команду

```
пиксель( x, 20, синий )
```

изменяя значение переменной **x** от 10 до 15. Мы уже можем сделать это так (вспомните цикл *N* раз):

```
цел x
x := 10
нц 6 раз
  пиксель( x, 20, синий )
  x := x + 1
кц
```

Здесь введена целая переменная **x**, которой сначала присваивается значение 10. Затем шесть раз выполняется команда, закрашивающая синим цветом пиксель с координатами (**x**, 20), и каждый раз значение **x** увеличивается на единицу.

В алгоритмическом языке есть еще один вид цикла — *цикл с переменной*, — который позволяет записать то же самое задание для Рисователя проще:

```
цел x
нц для x от 10 до 15
  пиксель( x, 20, синий )
кц
```

В заголовке цикла мы сказали, что тело цикла нужно выполнить для всех целых значений **x** от 10 до 15 включительно. Всю работу по увеличению переменной **x** на каждом шаге цикла программа берет на себя, нам не нужно об этом заботиться.

Итак, мы увидели, что с помощью команды **пиксель** можно достаточно легко нарисовать горизонтальную линию. Это связано с тем, что одна из координат (здесь — *y*-координата) не изменяется, она одинакова для всех точек линии. Вертикальная линия рисуется так же просто (не меняется *x*-координата).

А что если нужно нарисовать наклонную линию, например, соединить линией точки (10,20) и (20,50)? Здесь будут изменяться обе координаты, причем *y*-координата меняется быстрее, чем *x*-координата (подумайте, почему?). Поэтому решить такую задачу, используя только команду **пиксель**, очень непросто (можете попробовать). Еще сложнее нарисовать окружность.

Чтобы облегчить работу программистам, в СКИ графических исполнителей добавляют команды, которые рисуют сразу целые геометрические фигуры: линии, прямоугольники, окружности и др. Они называются *графическими примитивами*, используемыми в векторной графике. В следующем параграфе мы познакомимся с графическими примитивами в СКИ Рисователя.

Контрольные вопросы

1. Чем отличаются программы, работающие в текстовом и графическом режимах?
2. Что такое консольная программа?
3. Как определяются координаты пикселя на холсте? Чем необычна эта система координат?

[7] Координаты пикселей записываются в круглых скобках через запятую, сначала — *x*-координата, потом — *y*-координата.

Как вы думаете, почему в компьютерных программах используется именно такая система координат?

4. Как создать новый холст размером 200 × 100 пикселей и залить его синим цветом?

5. Что будет, если мы вызовем команду **пиксель** так:

```
пиксель( синий, 10, 20 )
```

В чем ошибка?

6. Чем отличается цикл с переменной от цикла **N раз** и цикла с условием?

7. Можно ли зациклить программу, используя цикл с переменной?

8. Почему сложно рисовать наклонные линии и окружности, используя только команду **пиксель**?

Задачи

1. Определите координаты всех пикселей на рисунке:

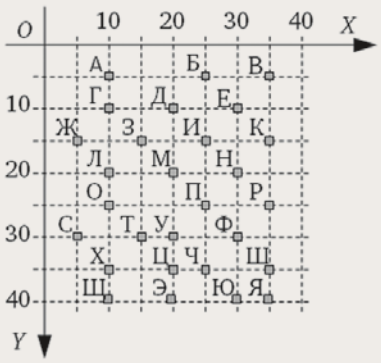


Рис. 63

2. Нарисуйте горизонтальную и вертикальную линии с помощью команды **пиксель** и циклов.

3. Нарисуйте прямоугольник размером 100 × 50 пикселей с помощью команды **пиксель** и циклов.

Окончание в следующем номере

ЭТО ПОЛЕЗНО ЗНАТЬ

Как мы создавали мультфильм

Е.Н. Лунина, учитель информатики школы № 4, г. Реутов Московской обл.

Многие дети, да и взрослые тоже, любят смотреть мультипликационные фильмы. Каждый мультфильм — это неведомый волшебный мир. Интересно, а как же могут двигаться куклы или нарисованные картинки? Как их "оживить"? Что для этого нужно сделать?

В одном мультфильме собраны слова, музыка, рисунок, движение! К тому же "ожившие" куклы, "ожившие" рисунки — это так близко, так понятно любому малышу. Ведь он и сам в своих играх всегда "оживляет" их, заставляет (не только мысленно) двигаться, совершать обычные и необычные поступки. Да, мультипликация приходит к нам очень рано, воспитывает радостью. Поясняет, при этом очень убедительно, и что такое хорошо, и что такое плохо. Приобщает к хорошему, отвращает от плохого — без унылого назидания, без скучных разговоров, которые мы не всегда готовы слушать.

Но одно дело — смотреть мультфильмы в кинотеатре или по телевизору, и совсем иное — создать самому и показывать другим: и взрослым, и сверстникам, и малышам.

Процесс создания мультфильмов — не только увлекательный творческий процесс, он развивает и воспитывает нас. Мы знаем, что хотим сделать, и четко представляем, для чего рисуем, лепим, мастерим. Занятия мультипликацией помогают увидеть привычное по-новому, понять красоту окружающего мира и человеческих отношений. Они формируют эстетический вкус, помогают приобрести технические знания, воспитать любовь к труду. На этих занятиях развиваются творческие способности, воображение, чувство цвета, ритма, восприятие и умение передавать пропорции, объем, движение. А это пригодится человеку любой профессии, в любом виде деятельности, которым он будет заниматься в жизни.

Что такое мультипликация

Слово мультипликация в переводе с латинского означает "умножение". Словарь русского языка объясняет: «**Мультипликáция** (от лат. *multiplicatio* — умножение, увеличение, возрастание, размножение), **анимáция** (англ. *animation* — одушевление, лат. *animare* — оживить) — вид киноискусства, произведения которого создаются путем покадровой съемки отдельных рисунков (в том числе составных) — для рисованных фильмов, или покадровой съемки отдельных театральных сцен — для кукольных фильмов. В настоящее время, помимо привычного термина "мультипликация", широко употребляется также и термин "анимация"».

Давайте рассмотрим этапы, которые необходимо пройти автору при создании мультфильма.

1. Выбрать сюжет

Все начинается с идеи. Это может быть как просто пришедшая кому-то в голову идея, так и конкурс идей или так называемый "мозговой штурм"[1]. Когда идея готова, необходимо сделать наброски портретов будущих героев, пейзажей, сцен и схему фильма. Созданные наброски или картины используют для уточнения сюжета будущего фильма.

2. Написать сценарий

Создание фильма начинается с подробного письменного киносценария. Нужно показать, как каждая сцена будет выглядеть в фильме. Сценарий фильма пишется либо одновременно с раскадровкой (созданием серии последовательных кадров, немного отличающихся друг от друга), либо с небольшим опережением ее.

3. Создать персонажей и обстановку

После написания сценария надо нарисовать или просто составить раскадровку по сценам и по времени.

[1] Метод мозгового штурма (мозговой штурм, мозговая атака) — метод решения проблемы на основе стимулирования творческой активности, при котором участникам обсуждения предлагают высказывать как можно большее количество вариантов решения, в том числе самых фантастичных. Затем из общего числа высказанных идей отбирают наиболее удачные, которые могут быть использованы на практике. — **Прим. ред.**

4. Выставить движения

Собрать мультфильм: проработать анимацию, то есть движения персонажа, используя компьютерные технологии.

5. Подобрать звуковое сопровождение

Для озвучивания используется видеоредактор — программа, имеющая расширенные возможности по работе со звуком. Следует поставить собранные сцены на временную линию. После этого начинается озвучивание: подключают микрофон, включают запись на звуковую дорожку, смотрят на изображение и в нужный момент говорят в микрофон, озвучивая персонажей (можно делать голоса более детскими, применяя специальный фильтр).

6. Создать начальную заставку и "концовку"

Именно они помогут вам ярко начать и эффектно закончить свой фильм.

Покажем все на примере создания автором и его учениками мультфильма "Колобок".

В ходе создания фильма нами были использованы возможности компьютерных программ Microsoft Paint и Microsoft PowerPoint. Paint — графический редактор (программа, предназначенная для создания графических изображений — картинок, открыток и т.п.). PowerPoint является приложением, которое позволяет "оживить" объекты на экране с помощью анимации. Это достигается за счет того, что при смене одного слайда следующим в картинке что-то меняется[2].

В соответствии с описанными выше правилами наши действия по созданию фильма были следующими:

— сначала мы придумали идею для мультфильма;

— далее написали сценарий;

— нарисовали картинки: для того чтобы создать персонажей к фильму, необходимо было познакомиться с графическим редактором Paint, а для анимации объектов использовали программу PowerPoint;

— проработали движения объектов;

— подобрали звук;

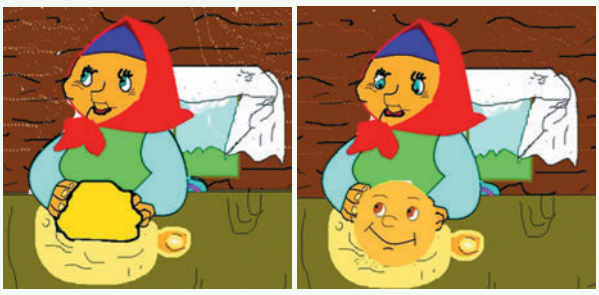
— создали заставку и концовку.

Нами была придумана серия сюжетов фильма (сценарий).

Первый сюжет называется "Дедка с бабкой…". Перед зрителем возникает образ деревенской избы, и он видит, как дедка с бабкой ведут беседу. Этот эффект достигнут нами следующим путем. Сначала был нарисован слайд с несколькими изображениями дедки с бабкой. Затем мы, используя копирование, создали 10 одинаковых слайдов. И на каждом из них мы поменяли лишь положение глаз и губ. Для озвучивания мы использовали мелодию сказки "Тишина, тишина". Так мы создали связь между изображением и звуком, и зрителю кажется, что перед ним настоящие образы сказочных дедки и бабки.

[2] При этом смена слайдов должна происходить через очень небольшой промежуток времени, значение которого устанавливается опытным путем до достижения плавных движений на экране. — **Прим. ред.**

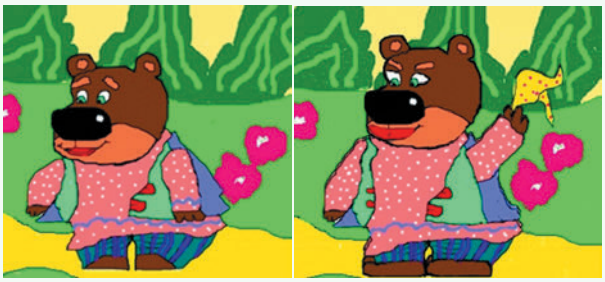
Следующий сюжет — "Колобок". Зритель видит, как на экране появляется главный герой фильма и перемещается по избе, а затем и вне избы. Он на каждом следующем рисунке представлен в чуть иной фазе движения. Для усиления эффекта движения Колобка мы меняли фон, на котором происходит перемещение героя сказки. Он катится по различной местности. Зритель видит, как на экране светит солнце, много цветов. А это подчеркивает движение не только в пространстве, но и во времени. Таким образом, мы желали создать ощущение реальности происходящего. Мы хотели, чтобы зритель смотрел на пейзажи, которые окружают Колобка, и получал эстетическое удовольствие от увиденного и услышанного.



Следующий сюжет мы с ребятами назвали “Встреча с зайцем”. На экране — с пригорка спускается заяц, а навстречу по тропинке катится Колобок. Зритель видит пейзаж: старый дуб на пригорке, на лужайке — цветы. Светит солнце. На лужайке Колобок встречается с зайцем, и вот прямо на наших глазах начинается диалог между героями сказки. Происходит “маленькое чудо”: Колобок исполняет свою песенку и уходит от зайца.

Затем идет сюжет, которому мы дали название “Встреча с волком”. На экране зрители видят нашего героя и волка — серого и злого. Для усиления его злых качеств мы анимировали движение языка в пасти волка. Пейзаж здесь взят другой: на небе облака, вдалеке небольшой лесок. Колобок исполняет свою песенку. Чтобы это сделать более достоверным, кроме озвучания эпизода и всего мультфильма, у Колобка было анимировано движение его рта.

Сюжет “Встреча с медведем” очень красочен и богат и движениями героев сказки, и цветовой гаммой. Здесь мы передали достаточно добрый характер медведя, несмотря на его грозные размеры, через эффект помахивания платочком вслед удаляющемуся Колобку.



Заключительный сюжет мультфильма назван “Лиса и Колобок”, в котором акцент сделан на хитрость лисы (мы реализовали это через движения лисы в сторону Колобка) и на его доверчивость, который прыгает лисе на нос и…

В результате за короткое время, используя рисованную мультипликацию, мы создали компьютерный вариант русской народной сказки “Колобок”. Мультипликация “оживила” неподвижные книжные картинки.

Послесловие

Создание мультфильмов — процесс творческий, он увлекает. Человек, который попробовал их делать, уже не может оторваться. Закончив работу над одним мультфильмом, ему хочется начать делать следующий. Ведь он знает, что его произведение будет интересно посмотреть другим людям.

Литература

1. *Халатов Н.В.* Мы снимаем мультфильмы. М.: Молодая гвардия, 1986.

2. *Босова Л.Л.* Информатика: Учебник для 6-го класса. М.: Бином. Лаборатория знаний, 2009.

3. Большая иллюстрированная энциклопедия интеллекта “Хочу все знать”. М.: Эксмо, 2007.

4. *http://library.by/*,
http://gif2000.narod.ru/15.htm,
http://babyroom.narod.ru/mult,
http://www.compress.ru/article,
http://www.ru/inform22.

ПОИСК ИНФОРМАЦИИ

Пять вопросов

Ответы на заданные вопросы найдите в Интернете или по другим источникам информации.

1. Что называют “королем риса”?

2. В какую игру индийский царь Наль проиграл не только свои владения, но и супругу?

3. Кому пророк Иеремия уподобляет еврейский народ?

4. Какой великий художник после своей смерти целую неделю пролежал в музее забальзамированным?

5. Из какого города был самый известный герой книг Даниеля Дефо?

Правда ли, что…?

1. Пищевая ценность семечек кабачка и тыквы повышается с возрастом. Они относятся к продуктам, которые по мере разложения становятся более питательными. В семенах кабачка и тыквы, хранив-

шихся более пяти месяцев, содержание белка сильно повышено.

2. Когда Николай Васильевич Гоголь был маленьким, его бабушка рассказывала ему о божественной лестнице, которую спускают с неба ангелы, подавая руку душе умершего. Последними словами классика были: “Лестницу! Поскорее давай лестницу!”.

Ответы (можно на отдельные задания и не на все вопросы) присылайте в редакцию.

ЗАДАЧНИК

Трилистник и другие

Мы продолжаем (см. [1–2]) обсуждать изображения так называемых “замечательных кривых”. В этот раз — кривых, вид которых показан на *рис.* 1 и 2.

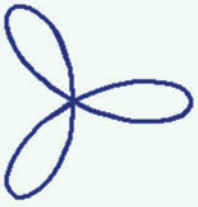


Рис. 1

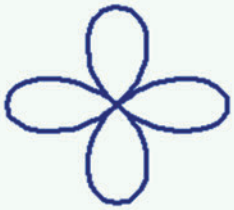


Рис. 2

На *рис.* 1 изображена кривая, которую называют “трилистник”. Ее уравнение в так называемых “полярных координатах” [3]:

$$R = a\sin(3\phi) \qquad (1)$$

Напомним, что в полярных координатах положение точки на плоскости определяется расстоянием R от начала координат и углом ϕ между радиус-вектором точки и лучом, проходящим через начало координат (он, как правило, совпадает с осью x в прямоугольной системе координат — см. *рис.* 3).

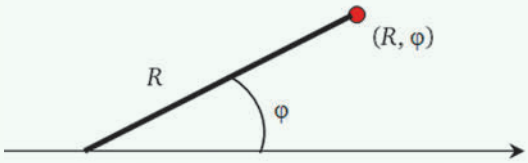


Рис. 3

Уравнение (1) определяет зависимость R от угла ϕ.

В статьях [1, 3] отмечалось, что при построении кривых, заданных уравнением в полярных координатах, на компьютере удобнее все же снова обратиться к декартовым координатам. Вспомнив о синусе и косинусе угла ϕ, можем сказать, что точка (R, ϕ) в полярных координатах — это то же самое, что $(R\cos(\phi), R\sin(\phi))$ в декартовых координатах, и именно ее мы можем построить (см. *рис.* 4).

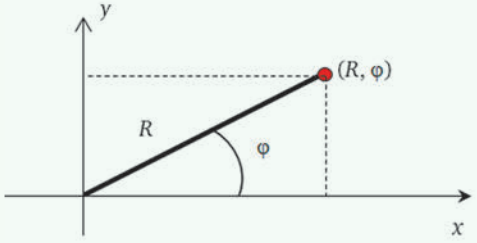


Рис. 4

Поэтому из уравнения в полярных координатах (1) можно легко получить параметрические уравнения трилистника:

$$x = \sin(3\phi)\cos(\phi);$$
$$y = \sin(3\phi)\sin(\phi).$$

Используя параметрические уравнения линии, можно получить ее изображение в программе — для этого надо рассчитать значения координат x и y для всех углов от 1 до 360 градусов через 1 градус и поставить точку в соответствующем месте экрана. Прежде чем представлять программу, решающую такую задачу для трилистника (на школьном алгоритмическом языке), заметим, что в ней используется также величина a — коэффициент увеличения размера графика.

```
алг Трилистник
нач цел x, y, угол, x0, y0, вещ a
вывод "Задайте значение a"
ввод a
|Устанавливаем графический режим
…
|Координаты центра экрана
x0 := int(максX/2); y0 := int(максУ/2)
|Для всех целых значений углов
нц для угол от 1 до 360
  |Переводим угол в радианы
  угол2 := 6.28 * угол/360
  |Рассчитываем координаты
  |соответствующей точки кривой
 x := x0 + int(a * sin(3 * угол2) * cos(угол2))
 y := y0 - int(a * sin(3 * угол2) * sin(угол2))
  |и изображаем эту точку
  точка(x, y)
кц
кон
```

Примечания

1. *x*0 и *y*0 — координаты центра экрана (с учетом этих координат рассчитываются значения x и y); значения *x*0 и *y*0 зависят от величин макс*X* и макс*У*, равных, соответственно, максимальному значению координат x и y в выбранном режиме работы экрана.

2. *угол*2 — величина угла в радианах.

3. Функция `int` возвращает целую часть ее вещественного аргумента.

4. В программе учитывается направление оси y в экранной системе координат.

Можно также использовать электронную таблицу Microsoft Excel или др. Здесь значение a можно не использовать. Фрагмент листа показан ниже (необходимые формулы запишите самостоятельно).

	A	B	C	D
1	Угол в градусах	Угол в радианах	x	y
2	0	0,00	0,00	0,00
3	5	0,09	0,26	0,02
4	10	0,17	0,49	0,09
…	…	…	…	…
73	355	6,19	–0,26	0,02
74	360	6,28	0,0	0,00

По полученным расчетным данным можно построить график (тип диаграммы — **Точечная с гладкими кривыми**).

При этом следует учесть, что на диаграммах и графиках электронных таблиц масштаб по осям *x* и *y* в общем случае не совпадает, в результате чего "листики" изображения могут получиться разными. Сделать масштаб одинаковым можно, меняя размеры области диаграммы.

О том, как называется линия, представленная на *рис*. 2, вы конечно же уже поняли…

Задания для самостоятельной работы

1. Используя электронную таблицу Microsoft Excel или другую или/и разработав программу (на языке программирования, который вы изучаете), получите изображение линии на *рис*. 2. Ее уравнение в полярных координатах:

$$R = \sin(2\phi).$$

2. Определите, как будет выглядеть изображение, если использовать такие уравнения линии в полярных координатах:

а) $R = \sin(6\phi)$;

б) $R = \sin(7\phi)$.

3. Установите, что и как определяет значение k при целом и положительном k в уравнении линии в полярных координатах $R = \sin(k\phi)$[3].

Указания по выполнению

Для построения кривых используйте их параметрические уравнения.

Результаты присылайте в редакцию.

Литература

1. О кардиоиде и о сердце. / "В мир информатики" № 203 ("Информатика" № 12/2014).

2. Еще две замечательные кривые. / "В мир информатики" № 204 ("Информатика" № 1/2015).

3. *Златопольский Д.М.* Полярные координаты. / "В мир информатики" № 193 ("Информатика" № 1/2014).

Дополнительные списки читателей, приславших ответы на задания, опубликованные в сентябрьском выпуске "В мир информатики"

Ответы на задания 1-го тура конкурса № 111 "Переправы" представили и стали участниками конкурса также:

— Головченко Тихон, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Дибров Сергей, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Пилясов Иван, г. Ярославль, школа № 33, учитель **Цикина Е.Н.**;

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

— Фомичева Дарья и Якушов Александр, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина, учитель **Чапкевич И.М.**;

— Щегольков Дмитрий, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**

Пак Александра и Приходько Геннадий разработали ряд программ, предложенных для самостоятельной работы в статье "Рекурсия — эффектно, но не всегда эффективно" (в том числе программ, с помощью которых можно получить красивые изображения). Редакция решила наградить Александру и Геннадия дипломами. Поздравляем!

Задачу "Четыре приятеля" правильно решили также:

— Волков Станислав и Миротина Елена, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Иванов Илья и Якушов Александр, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина, учитель **Чапкевич И.М.**;

— Леженников Тарас, Краснодарский край, г. Приморско-Ахтарск, школа № 22, учитель **Корнеева М.В.**;

— Леоненко Степан, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Русель Вероника, средняя школа села Ошминское Тоншаевского р-на Нижегородской обл., учитель **Попов Г.Н.**;

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

— Чернова Ксения, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**

Правильное решение головоломки "Пляшущие человечки" прислали также:

— Бородулина Елена, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Измайлова Екатерина и Якушов Александр, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина, учитель **Чапкевич И.М.**;

— Цыганков Евгений, Вадьковская средняя школа, Брянская обл., Погарский р-н, учитель **Цыганкова И.Ю.**;

— Чернова Ксения, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**;

[3] Семейство кривых, полярное уравнение которых записывается в виде $R = \sin(k\phi)$, называют "розами", или "кривыми Гвидо Гранди" (по имени итальянского математика XIX века).

— Щегольков Дмитрий, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**

Головоломку “Получить верное равенство”, связанную с перекладыванием спичек, правильно решили также:

— Дибров Сергей, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Измайлова Екатерина, Олейников Андрей и Якушов Александр, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина, учитель **Чапкевич И.М.**;

— Русель Вероника, средняя школа села Ошминское Тоншаевского р-на Нижегородской обл., учитель **Попов Г.Н.**;

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

— Цыганков Евгений, Вадьковская средняя школа, Брянская обл., Погарский р-н, учитель **Цыганкова И.Ю.**;

— Чашникова Марина, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Чернова Ксения, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**

Правильный ответ к задаче “Верблюды и дядя Федор” представили также:

— Филиппенко Михаил, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Чернова Ксения, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**;

— Якушов Александр, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина, учитель **Чапкевич И.М.** (Александр снабдил ответ схемой).

Задачу “Профессор и студенты” правильно решили также:

— Казанец Елена, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Чернова Ксения, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**

Решение числового ребуса “ОДЕЯЛО из ЛОСКУТОВ” представили также:

— Иванов Иван и Якушов Александр, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина, учитель **Чапкевич И.М.** (Александр Якушов описал в письме методику решения);

— Чернова Ксения, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**;

— Шаров Константин, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**

Ответы на головоломку “Суперлабиринт” представили также:

— Бородулина Елена, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Зозуля Анастасия, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Якушов Александр, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина, учитель **Чапкевич И.М.**

Задачу “Подсчет пальцев” правильно решили также:

— Казанец Елена, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Филиппенко Михаил, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Чернова Ксения, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**

Якушов Александр, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина (учитель **Чапкевич И.М.**), представил решение головоломки “Переставить шашки”. Учитывая сложность головоломки, редакция решила наградить Александра дипломом. Поздравляем!

Решения двух японских головоломок, опубликованных в сентябрьском выпуске, прислали также:

— Анфиногенова Дарья и Леухина Екатерина, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Краснёнкова Л.А.**;

— Баженов Михаил, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Калинина Ирина, г. Воронеж, лицей № 2, учитель **Комбарова С.И.**;

— Коростелев Иннокентий и Марун Виталий, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Милушкин Дмитрий, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**

Русель Вероника, средняя школа села Ошминское Тоншаевского р-на Нижегородской обл. (учитель **Попов Г.Н.**) прислала также правильный ответ к заданию “Пляшущие человечки”.

Программу решения одной из задач, предложенных для самостоятельной работы в статье “Рекурсия — эффектно, но не всегда эффективно”, представили:

— Андросенко Дмитрий, средняя школа села Дохновичи Брянской обл., учитель **Клопова Е.В.**;

— Потапов Сергей, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**

Четыре талантливые девочки

Марина, Ирина, Иржина и Карина — талантливые девочки. Каждая из них играет на каком-нибудь музыкальном инструменте и говорит на одном из иностранных языков. Инструменты и языки у них разные. Марина играет на рояле. Девочка, которая говорит по-французски, играет на скрипке. Ирина играет на виолончели, а Иржина не говорит по-немецки. Марина не знает итальянского языка, а Ирина не владеет английским и не играет на скрипке. Иржина не играет на арфе. Карина не знает французского, а виолончелистка не говорит по-итальянски. Определите, кто на каком инструменте играет и кто на каком языке говорит.

Три ученика

Три ученика различных школ из одного города приехали на отдых в летний лагерь. На вопрос вожатого, в каких школах они учатся, каждый дал ответ:

1) Петя: “Я учусь в школе № 24, а Леня — в школе № 8”;

2) Леня: “Я учусь в школе № 24, а Петя — в школе № 30”;

3) Коля: “Я учусь в школе № 24, а Петя — в школе № 8”.

Вожатый, удивленный противоречиями в ответах ребят, попросил их объяснить, почему так. Тогда ребята признались, что в ответах каждого из них одно утверждение верно, а другое — ложно.

Смог ли вожатый определить, в какой школе учится каждый из мальчиков?

Обманутый хозяин

Когда-то в древности один хозяин устроил в своем погребе стеллаж в форме квадрата с девятью отделениями. Среднее (внутри) отделение он оставил свободным для пустых бутылок, а в остальных расположил 60 бутылок масла так, что в каждом угловом отделении их было по 6, а в каждом из средних — по 9. Таким образом, на каждой стороне квадрата было по 21 бутылке:

6	9	6
9		9
6	9	6

Слуга подметил, что хозяин с целью экономии времени проверяет число бутылок, только считая бутылки по сторонам квадрата и следя за тем, чтобы на каждой стороне квадрата было по 21 бутылке. Тогда слуга унес сначала 4 бутылки, а остальные расставил так, что вновь получилось по 21 на каждой стороне. Хозяин пересчитал бутылки своим обычным способом и подумал, что бутылок остается то же число и что слуга только переставил их. Слуга воспользовался оплошностью хозяина и снова унес 4 бутылки, расставив остальные так, что на каждой стороне квадрата выходило опять по 21 бутылке. Так он повторял до тех пор, пока было возможно.

Сколько раз слуга брал бутылки и сколько всего бутылок он унес?

Яд и кролики

Есть три бутылки с водой. В одной из них может быть растворен яд. Для ее нахождения можно использовать кроликов (к сожалению, для научных исследований, для проведения экспериментов и т.п. иногда используются животные — мыши, кролики и другие[4]). Яд действует через два дня и смертелен в любой дозе. Как, имея двух кроликов, за минимально возможное время определить, есть ли яд в бутылках и, если есть, то в какой?

Размеры футбольного поля

На планете τ-Кита футбольное поле имеет размеры, как и на Земле, — 105 × 68 м. Но τ-китяне записывают эти размеры как 125 × 75 м. Почему?

ЯПОНСКИЙ УГОЛОК

Два судоку

Решите, пожалуйста, две японские головоломки “судоку”:

1) простую:

				5	4		2	
5						8	4	6
					9	3		
8	6	4	7	9				
		7				4		
				8	6	2	9	7
		5	9					
3	9	2						1
	8		5	1				

2) сложную:

		2				4	3	1
		3				5	2	9
	9		3				7	
			9					
		5				2	8	
	4	7	6		8			
		9		4	6			2
	6						9	
2				5			6	

Решения (можно не все) присылайте в редакцию.

[4] Отсюда выражение — “подопытный кролик”.

Не ошибается лишь тот, кто ничего не делает. Не бойтесь ошибаться — бойтесь повторять ошибки.

Теодор Рузвельт

“ЛОМАЕМ” ГОЛОВУ

Числовые ребусы в троичной системе. Часть 6

В приведенных ниже ребусах зашифрованы числа, записанные в троичной системе счисления. Одинаковым буквам соответствуют одинаковые цифры. Звездочкой (“*”) может быть любая цифра.

1.
```
    * 
+ * *
-----
D * D
```

2.
```
  A B
+ A B
-----
* * 0
```

3.
```
  A 1
+ A 1
-----
* * *
```

4.
```
  A B
+ A B
-----
  * *
```

Ответы присылайте в редакцию (можно решать не все ребусы).

Восстановите набор чисел

Ниже представлены три набора чисел. Используя первые два, восстановите третий набор. Какое число скрывается под знаком вопроса?

4: 6, 11, 7;

3: 8, 4, 12;

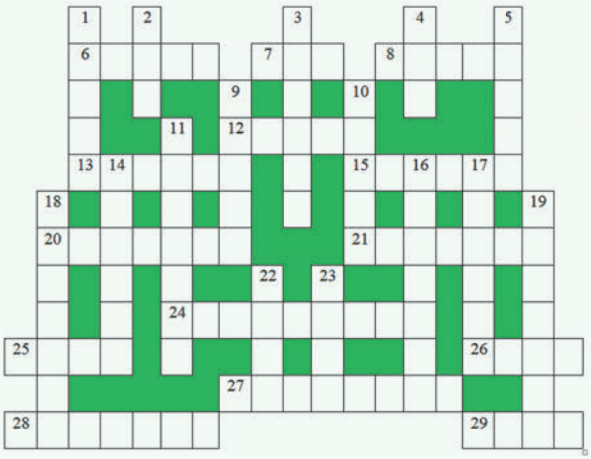
?: 10, 55, 15.

Переложить одну спичку

Переложите одну спичку, чтобы слева и справа от знака равенства было указано одно и то же число.

Кроссворд

Решите, пожалуйста, кроссворд.



По горизонтали

6. Часть адреса в электронной почте.

7. Единица измерения скорости передачи информации.

8. Массив точек для формирования графического изображения на экране монитора.

12. Первая буква греческого алфавита.

13. Перечень людей, предметов и т.п., а в программировании — совокупность элементов, каждый из которых содержит указатель на следующий элемент.

15. В программировании — совокупность значений одного типа, имеющих общее имя.

20. Разновидность, модификация, версия.

21. Утверждение, принимаемое без доказательства.

24. Логическая операция.

25. Цифра восьмеричной системы счисления.

26. Счетное устройство у древних греков и римлян, похожее на счеты.

27. Предприятие общественного питания.

28. Набор цветов, которые могут использоваться в изображении.

29.

По вертикали

1. Порядковый номер байта оперативной памяти.

2. Характеристика файла или переменной величины в программировании.

3. Тип топологии локальной сети.

4. Величина изменения значения переменной цикла.

5. Андрей Петрович… — советский академик, один из основоположников школьной информатики.

9. Комплекс связанных между собой программ.

10. Прямоугольник, ограничивающий меню или т.п.

11. Результат операции целочисленного деления.

14. Секретное слово.

16. Множество закономерно связанных между собой объектов, представляющее собой определенное целостное образование.

17. Элемент детали матричного принтера.

18. Существительное, связанное с первым словом выражения “…распространяемое программное обеспечение”.

19. Язык программирования.

22. Знак препинания.

23. Единица измерения количества информации.

Молочник и трое ребят

Трое ребят пришли к молочнику за молоком с бидонами объемом 3, 4 и 5 литра и попросили на-

лить каждому по два литра. У молочника было две полных больших фляги объемом 50 и 40 литров. Немного подумав, он легко справился с этим заданием. А вы сможете?

Решение, пожалуйста, оформите в виде таблицы:

		Бидон 3 л	Бидон 4 л	Бидон 5 л	Фляга 50 л	Фляга 40 л
	Исходное состояние	0	0	0	50	50
1	Налить из фляги … в бидон …					
2	…					

Три карточки

Перед Алешей и Митей на столе лежат три перевернутые карточки, под одной написано число 1, под другой — 2, под третьей — 3. Алеша их перемешал и вытащил одну из них, но какую — Мите не сказал. Митя может задать ему только один вопрос, на который тот, подумав, должен честно ответить “Да”, “Нет” или “Не знаю”, после чего Митя должен наверняка отгадать число, которое вытащил Алеша. Как?

Какая буква спрятана ☺?

В двойном буквенном неравенстве одна из букв заменена вопросительным знаком:

Д + Б + В + Ж + К < Р < Д + Б + В + Ж + К + ?

Какая?

Автор задачи — Игорь Акулич

Получить 24

Переставьте цифры 1, 3, 4 и 6 в любом порядке и расставьте между какими угодно из них знаки арифметических действий и скобки так, чтобы получилось число 24. Решите задачу двумя способами:

1) со “склейкой” цифр в многозначное число;

2) без “склейки” цифр.

ЗАДАЧНИК

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

Решение задания 2-го тура конкурса № 111 “Переправы”

Напомним условие: “Три жулика, каждый с двумя чемоданами, находятся на одном берегу реки, через которую они хотят переправиться. Есть трехместная лодка, каждое место в ней может быть занято либо человеком, либо чемоданом. Никто из жуликов не доверит свой чемодан спутникам в свое отсутствие, но готов оставить чемоданы на безлюдном берегу. Смогут ли они переправиться?”

Ответ — смогут.

Решение

Обозначим жуликов большими буквами A, B, C, а их чемоданы — маленькими. Схематически переправу можно изобразить так:

1. Ccc →.
2. C ←.
3. CAB →.
4. AB ←.
5. Aaa →.
6. AC ←.
7. ABC →.
8. B ←.
9. Bbb →.

Комментарии

Сначала C перевозит свои чемоданы, затем без багажа возвращается обратно и перевозит A и B (без багажа). После этого A и B возвращаются, и A перевозит свои чемоданы. Наконец, A и C возвращаются и перевозят B, который возвращается один за своими чемоданами.

Ответы прислали:

— Абдувахидова Алина, Абдувахидова Софья, Галкина Эвелина, Головченко Тихон, Лебедева Анастасия, Строкин Константин и Хозин Марат, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Антипов Анатолий и Прокопенко Сергей, средняя школа поселка Осиновка, Алтайский край, **учитель Евдокимова А.И.**;

— Байкова Римма, Дубинина Анна, Иванова Виолетта и Левченко Ирина, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Василенко Татьяна, Кочемасов Даниил и Ухин Станислав, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Васина Светлана, Макаренко Виталий и Хомутова Евгения, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Владимиров Виталий и Яковлева Анастасия, основная школа села Именево, Республика Чувашия, Красноармейский р-н, учитель **Тимофеева И.А.**;

— Водальчук Михаил, Горбунова Ксения, Курдамосова Софья, Пискунова Полина, Попова Елизавета и Петрова Полина, гимназия г. Шелехова, Иркутская обл., учитель **Водальчук С.А.**;

— Гагарин Валерий и Кириллов Иван, г. Сегежа, Республика Карелия, школа № 6, учитель **Соколова В.Н.**;

— Донникова Анна и Ломтев Павел, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Захарова Юлия, Смоленская обл., г. Демидов, школа № 1, учитель **Кордина Н.Е.**;

— Иванов Алексей, Свердловская обл., Красноуфимский р-н, Тавринская средняя школа, учитель **Ярцев В.А.**;

— Карданова Аминат, Коломина Нонна, Медяникова Аделина, Остроухова Валерия и Уткина Ксения, Ставропольский край, Кочубеевский р-н, станица Барсуковская, школа № 6, учитель **Рябченко Н.Р.**;

— Кондрашова Яна, Селиванов Илья и Якушов Александр, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина, учитель **Чапкевич И.М.** (Илья Селиванов подготовил видеосюжет, иллюстрирующий решение задачи);

— Красненков Александр, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Прохоренко Н.Ю.**;

— Ледин Роман, г. Ростов-на-Дону, лицей № 56, учитель **Ли В.М.**;

— Леженников Тарас, Краснодарский край, г. Приморско-Ахтарск, школа № 22, учитель **Корнеева М.В.**;

— Лежнева Александра, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Линькова Кристина и Цуцков Илья, Ардатовский коммерческо-технический техникум, поселок Ардатов Нижегородской обл., преподаватель **Зудин В.П.** (Кристина предложила решение, связанное с использованием веревки, протягиваемой между берегами. Вопрос о возможности такого решения редакция оставляет открытым);

— Новиченко Владислав и Царькова Марина, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Перешнев Алексей, г. Архангельск, школа № 9, учитель **Дудина Т.В.**;

— Пискунова Полина, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Краснёнкова Л.А.**;

— Силаев Дмитрий, г. Ростов-на-Дону, лицей № 56, учитель **Назаренко С.Н.**

Заметим, что некоторые из перечисленных читателей не представили ответ на задание 1-го тура конкурса, однако, учитывая, что это задание было “легче”, чем задание 2-го тура, редакция включила их в список участников конкурса.

Задача “Как выйти из комнаты?”

Напомним условие: “Граф Кристомонте находился в комнате, в которой было четыре двери и маленькое окошко. Три из четырех дверей “мнимые” — то есть за ними глухая стена и только одна вела наружу. У него был ключ, который подходил к замкам всех дверей, но он не знал, какая из четырех дверей ведет на улицу. Если бы граф попытался открыть одну из “мнимых” дверей, то все замки остальных дверей автоматически закрылись бы навсегда. Еще в комнате было темно, только одна свеча давала немного света. Кристомонте смог за одну попытку найти ту дверь, что вела его наружу. А вы смогли бы? Как?”

Решение

Идея решения заключается в том, что нужно в комнате устроить сквозняк — открыть окошко, а затем поочередно подносить свечу к каждой из дверей, точнее, к щели верхней, нижней либо к замочной скважине и следить за пламенем свечи. Где пламя начнет колебаться — там и выход.

Ответы прислали:

— Абдувахидова Алина, Абдувахидова Софья, Бледных Кирилл, Викторов Даниил, Галкина Эвелина, Карпов Иван, Лебедева Анастасия, Манукян Григорий, Милушкин Дмитрий, Строкин Константин, Филиппова Наталья и Хозин Марат, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Байкова Римма, Дубинина Анна и Левченко Ирина, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Бульбова Лидия, г. Пионерский Калининградской обл., школа № 2, учитель **Багрова О.А.**;

— Герасимова Наталья и Костина Евгения, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Дибров Сергей, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Живило Андрей, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Калинина Ирина, г. Воронеж, лицей № 2, учитель **Комбарова С.И.**;

— Краснова Диана, Свердловская обл., г. Ревда, школа № 10, учитель **Игошева А.А.**;

— Леженников Тарас, Краснодарский край, г. Приморско-Ахтарск, школа № 22, учитель **Корнеева М.В.**;

— Макарова Светлана, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Наурбиева Фатима и Пилясов Иван, г. Ярославль, школа № 33, учитель **Цикина Е.Н.**;

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

— Хвойновский Вадим, Царев Иван и Чернова Ксения, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**;

— Худокормова Мария и Якушов Александр, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина, учитель **Чапкевич И.М.**

Заметим, что среди присланных ответов был такой “оригинальный”, как “кинуть свечку в окно, и под какой дверью мелькнет свет, та дверь и есть настоящая” ☹.

Кроссворд (опубликованный в октябрьском выпуске)

Ответы

По горизонтали: 1. Имя. 5. Трек. 8. Марка. 9. Ромб. 10. Диалог. 11. Ось. 12. Ремонт. 13. Логика. 14. Текст. 18. Номер. 19. Ток. 21. Мост. 23. Копирование. 25. Ля. 26. Партия. 28. Тире. 29. Условие. 30. Портал.

По вертикали: 2. Монитор. 3. Программирование. 4. Параметр. 6. Робот. 7. Кольцо. 8. Миллион. 15. Компилятор. 16. Тест. 17. Стек. 20. Копия. 22. Растр. 24. Абрис. 25. Лист. 27. Бод.

Обратим внимание на написание слова “бод”.

Ответы прислали:

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

— Барановская Татьяна и Жукова Ирина, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Владимирова Снежана и Семенова Екатерина, основная школа села Именево, Республика Чувашия, Красноармейский р-н, учитель **Тимофеева И.А.**;

— Вуколов Сергей, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Власова Ольга, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Краснёнкова Л.А.**;

— Гололобов Александр, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Диков Андрей, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Назаркина Татьяна и Петрова Алена, г. Пионерский Калининградской обл., школа № 2, учитель **Багрова О.А.**;

— Нелюбина Елизавета, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Одинцова Екатерина, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Трус Анастасия, средняя школа села Дохновичи Брянской обл., учитель **Клопова Е.В.**;

— Хвойновский Вадим, Царев Иван и Чернова Ксения, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**;

— Хозин Марат, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Якушов Александр, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина, учитель **Чапкевич И.М.**;

— Яснова Дарья и Яснов Федор, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**

Задача “Отмеривание кваса”

Напомним, что в задаче требовалось разработать алгоритм отмеривания 5 л кваса из 20-литровой емкости с использованием ведер на 7 и 13 л.

Решение

Задача решается за восемь переливаний.

Ответы представили:

— Абдувахидова Алина, Абдувахидова Софья, Бледных Кирилл, Викторов Даниил, Лебедева Анастасия и Манукян Григорий, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Байкова Римма и Левченко Ирина, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Багрова Анастасия и Шиколович Ростислав, г. Пионерский Калининградской обл., школа № 2, учитель **Багрова О.А.**;

— Дибров Сергей и Коломиец Екатерина, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Живило Андрей, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Жукова Ольга, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

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

— Калинина Ирина, г. Воронеж, лицей № 2, учитель **Комбарова С.И.**;

— Камынина Елизавета, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Карданова Аминат, Коломина Нонна, Медяникова Аделина, Остроухова Валерия и Уткина Ксения, Ставропольский край, Кочубеевский р-н, станица Барсуковская, школа № 6, учитель **Рябченко Н.Р.**;

— Краснова Диана, Свердловская обл., г. Ревда, школа № 10, учитель **Игошева А.А.**;

— Пилясов Иван, г. Ярославль, школа № 33, учитель **Цикина Е.Н.**;

— Рябов Григорий, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Хвойновский Вадим, Царев Иван и Чернова Ксения, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**;

— Худокормова Мария, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина, учитель **Чапкевич И.М.**

Задание “Четыре с половиной вопроса”

Ответы прислали:

— Андреев Дмитрий, Галяпин Глеб, Долгов Андрей и Калинина Ирина, г. Воронеж, лицей № 2, учитель **Комбарова С.И.**;

— Герасимова Наталья и Костина Евгения, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

	Емкость на 20 л	**Ведро на 7 л**	**Ведро на 13 л**
Исходное состояние	20	7	13
1. Налить из емкости 20 л в ведро 13 л	7	0	13
2. Перелить из ведра 13 л в ведро 7 л	7	7	6
3. Перелить квас из ведра 7 л в емкость 20 л	14	0	6
4. Перелить из ведра 13 л в ведро 7 л	14	6	0
5. Налить из емкости 20 л в ведро 13 л	1	6	13
6. Долить 1 л из ведра 13 л в ведро 7 л	1	7	12
7. Перелить квас из ведра 7 л в емкость 20 л	8	0	12
8. Перелить квас из ведра 13 л в ведро 7 л	8	7	5

— Гусева Маргарита, Жукова Ольга и Одинцова Екатерина, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

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

— Есипова Мария, Круглякова Мария и Яснова Дарья, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Иванова Татьяна, Морозова Елена и Назарова Марина, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Камынина Елизавета, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Огнев Ярослав и Якушов Александр, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина, учитель **Чапкевич И.М.**;

— Портнова Анна, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Хвойновский Вадим, Царев Иван и Чернова Ксения, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**;

— Хозин Марат, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**

Ответы

1. Первым обладателем так называемого “оберегающего посоха” из сандалового дерева был Будда (Будда Шакьямуни), легендарный основатель буддизма.

2. Едва не стал принцем, о чем мечтал с детства, герой произведения “Принц и нищий” американского писателя Марка Твена Том Кенти.

3. Героине детективного романа “Чаша Бланшара” писательницы Джанет Глисон раскрыть преступление помогла кулинария.

Напомним остальные вопросы: “Какой ученый был удостоен того, что благодарные соотечественники поместили на банкноте сразу два его портрета? Каким было достоинство этой банкноты? Чем прославился этот ученый?”

Публикуя их, редакция имела в виду английского физика Майкла Фарадея. Он — основоположник учения об электромагнитном поле (экспериментально открыл эффект вращения магнита вокруг проводника с током и проводника с током вокруг магнита). На банкноте в 20 фунтов стерлингов 1991 года выпуска были представлены два его изображения: в правой части банкноты — портрет, в левой изображен ученый, читающий рождественскую лекцию.

Александр Якушов из лицея № 4 г. Орла нашел информацию о том, что такой же чести был удостоен Карл Ландштейнер, австрийский биолог и врач, обнаруживший вирус полиомиелита. На лицевой стороне купюры 1000 австрийских шиллингов изображен его портрет, на оборотной — ученый в лаборатории.

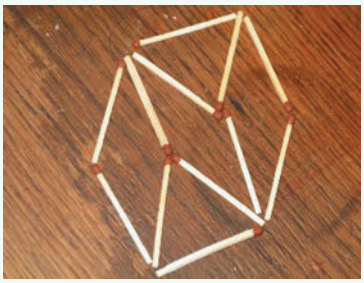
Отметим также развернутый и снабженный иллюстрациями ответ Ярослава Огнева из того же лицея.

Заметим, что в ряде ответов не учитывалось то, что в вопросе речь шла о двух портретах одного и того же ученого.

Головоломка “Двенадцать спичек”

Напомним, что требовалось из 12 спичек на столе составить фигуру, в которой было бы три одинаковых четырехугольника и два одинаковых треугольника.

Решение показано на рисунке.



Ответы прислали:

— Аносов Петр, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Бульбова Лидия, Назаркина Татьяна и Постников Владимир, г. Пионерский Калининградской обл., школа № 2, учитель **Багрова О.А.**;

— Власова Ольга и Миротин Андрей, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Жукова Ольга, Мацишина Валерия и Одинцова Екатерина, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Иванова Татьяна, Морозова Елена и Назарова Марина, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Краснова Диана, Свердловская обл., г. Ревда, школа № 10, учитель **Игошева А.А.**;

— Леоненко Степан, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Макаркин Дмитрий, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Огнев Ярослав, Олейников Андрей, Титкова Ирина, Хоченкова Дарья и Худокормова Мария, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина, учитель **Чапкевич И.М.**;

— Рябов Григорий, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Хвойновский Вадим, Царев Иван и Чернова Ксения, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**;

— Хомяков Антон, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**

Заметим, что в ряде ответов спички были выложены так, что их концы не соприкасались (приводятся и другие ошибочные варианты).

Числовой ребус “ТРОПКА — ДОРОЖКА”

Напомним, что требовалось решить числовой ребус:

ТРОПКА + ТРОПКА + ТРОПКА = ДОРОЖКА

В нем, как принято в таких головоломках, одинаковыми буквами зашифрованы одинаковые цифры, разными буквами — разные цифры.

Решение

Запишем ребус “в столбик”:

	Т	Р	О	П	К	А
+	Т	Р	О	П	К	А
	Т	Р	О	П	К	А
Д	О	Р	О	Ж	К	А

Есть только две цифры, которые, будучи сложенными трижды, дают результат, оканчивающийся на эту же цифру — 5 и 0. Исследуем эти два значения для букв **А** и **К**. А не может быть равно пяти, так как при этом **К** + **К** + **К** не оканчивается на **К**. Значит, **А** = 0, **К** = 5.

Далее исследуем разряд тысяч. Составим таблицу:

О	1	2	3	4	6	7	8	9
3 × **О**	3	6	9	12	18	21	24	27

Так как из разряда сотен в разряд тысяч в “уме” может перейти не больше двух (убедитесь в этом самостоятельно!), то сумма трех цифр **О** может оканчиваться на **О** только при **О** = 4 или при **О** = 9. Но в первом случае (в разряд слева переходит 1) получается противоречие со значением цифры **Р**. Значит, **О** = 9, а **Р** = 4.

Запишем найденные цифры в ребус:

	Т	4	9	П	5	0
+	Т	4	9	П	5	0
	Т	4	9	П	5	0
Д	9	4	9	Ж	5	0

Далее можно увидеть, что **Т** = 6, а **П** = 7 (значение **П** = 8 не подходит, так как при нем **Ж** = 5).

Итак, решение ребуса: 649 750 × 3 = 1 949 250.

Правильное решение привели:

— Абдувахидова Алина, Абдувахидова Софья и Манукян Григорий, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Абраменко Богдан, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Багрова Анастасия, Бульбова Лидия и Дикий Данила, г. Пионерский Калининградской обл., школа № 2, учитель **Багрова О.А.**;

— Вуколов Сергей, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

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

— Зубов Владислав, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Мацишина Валерия и Одинцова Екатерина, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Новиков Алексей, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Портнова Анна, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Торопов Александр, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Хвойновский Вадим, Царев Иван и Чернова Ксения, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**;

— Якушов Александр, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина, учитель **Чапкевич И.М.**

Задача “Семь кошельков”

Напомним, что предлагалось ответить на вопрос, как разложить по семи кошелькам 127 рублевых монет, чтобы любую сумму от 1 до 127 рублей можно было бы выдать, не открывая кошельков (то есть вместе с кошельками).

Решение

Десятичное число 127 равно двоичному 1111111. Любое двоичное число от 1 до 1111111 можно получить, складывая двоичные числа 1, 10, 100, 1000, 10 000, 100 000 и 1 000 000 (убедитесь в этом!). Эти числа есть двойка в степени 0, 1, 2, …, 7, то есть десятичные числа 1, 2, 4, 8, 16, 32 и 64. Поэтому для решения задачи в первый кошелек надо положить одну монету, во второй — две, в третий — четыре, в четвертый — восемь, в пятый — шестнадцать, в шестой — тридцать две, в седьмой — шестьдесят четыре. Если этим кошелькам присвоить условные номера, соответственно, 1, 2, …, тогда любую сумму от 1 до 127 рублей можно получить следующим образом. Нужно перевести эту сумму в двоичную систему счисления, после чего взять те кошельки, номера которых равны номерам тех разрядов двоичной записи суммы, где представлена цифра 1 (разряды следует нумеровать справа налево, начиная с 1). Например, чтобы выдать сумму в 109 рублей, надо отдать кошельки с номерами 7, 6, 4, 3 и 1, так как $109_{10} = 1101101_2$.

Правильные ответы прислали:

— Абдувахидова Алина, Абдувахидова Софья, Лебедева Анастасия и Милушкин Дмитрий, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Абраменко Богдан, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Вуколов Сергей, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Гусева Маргарита и Жукова Ольга, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

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

— Карданова Аминат, Коломина Нонна, Медяникова Аделина, Остроухова Валерия и Уткина Ксения, Ставропольский край, Кочубеевский р-н, станица Барсуковская, школа № 6, учитель **Рябченко Н.Р.**;

— Новиков Алексей, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Портнова Анна, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Рябов Григорий, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Торопов Александр, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Худокормова Мария, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина, учитель **Чапкевич И.М.**;

— Чернова Ксения, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**

Ответ к задаче “У кого какая профессия” представили:

— Абалкина Ирина, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Абдувахидова Алина, Абдувахидова Софья, Головченко Тихон, Кузьмина Злата и Хозин Марат, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Аносов Петр и Шмакова Елена, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

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

— Власова Ольга и Миротин Андрей, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Иванова Ксения и Мухина Светлана, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Калинина Ирина, г. Воронеж, лицей № 2, учитель **Комбарова С.И.**;

— Карданова Аминат, Коломина Нонна, Медяникова Аделина, Остроухова Валерия и Уткина Ксения, Ставропольский край, Кочубеевский р-н, станица Барсуковская, школа № 6, учитель **Рябченко Н.Р.**;

— Красненков Александр, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Прохоренко Н.Ю.**;

— Краснова Диана, Свердловская обл., г. Ревда, школа № 10, учитель **Игошева А.А.**;

— Лазарев Константин и Шухардина Елена, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Мазанова Екатерина, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Гусева Маргарита и Мацишина Валерия, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Пилясов Иван, г. Ярославль, школа № 33, учитель **Цикина Е.Н.**;

— Чернова Ксения, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**;

— Якушов Александр, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина, учитель **Чапкевич И.М.**,

а к задаче “Три одноклассника”:

— Абалкина Ирина и Сердюк Елена, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Аносов Петр, Тупицына Алена и Шмакова Елена, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Бородин Алексей и Бородин Александр, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Гусева Маргарита и Мацишина Валерия, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Иванова Ксения и Мухина Светлана, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Клопов Дмитрий, средняя школа села Дохновичи Брянской обл., учитель **Клопова Е.В.**;

— Лазарев Константин и Шухардина Елена, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Красненков Александр, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Прохоренко Н.Ю.**;

— Краснова Диана, Свердловская обл., г. Ревда, школа № 10, учитель **Игошева А.А.**;

— Мазанова Екатерина, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Миротин Андрей, Новиков Андрей и Турищева Марина, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Чернова Ксения, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**

Четыре числовых ребуса в троичной системе

Напомним ребусы:

1.

		M
+	**N**	**N**
	*	*

2.

	*	*
+	*	*
	*	0

3.

		B
+	**A**	*
	A	1

4.

	*	**C**
+	*	*
	D	**E**

Комментарии к решению

В ребусе 1 число **NN** не может быть равно 22.

В ребусе 2 сумма может быть равна только 20.

В ребусе 3 цифра **A** может быть равна только 2.

В ребусе 4 цифра **D** может быть равна только 2. Следует учесть также, что **E** ≠ **C**.

Решения

1.

		2
+	1	1
	2	0

2.

	1	0
+	1	0
	2	0

3.

		1
+	2	0
	2	1

4.

	1	**0**
+	**1**	**1**
	2	**1**

Ответы представили:

— Абраменко Богдан, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Багрова Анастасия и Бульбова Лидия, г. Пионерский Калининградской обл., школа № 2, учитель **Багрова О.А.**;

— Вуколов Сергей, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

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

— Зубов Владислав, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Краснова Диана, Свердловская обл., г. Ревда, школа № 10, учитель **Игошева А.А.**;

— Новиков Алексей, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Хвойновский Вадим, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**;

— Яковлева Александра, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**

Задача "Большая скорость"

Напомним условие: "Один инопланетянин написал, что за 4 часа, двигаясь со скоростью 400 км/ч, он проехал 3100 км. Как такое могло быть?"

Решение

Это могло быть, если все значения записаны не в десятичной системе счисления. Но в какой?

Запишем общее расстояние, пройденное за 4 часа, как сумму четырех значений скорости, в столбик:

$$\begin{array}{rr} & 400 \\ & 400 \\ + & 400 \\ & 400 \\ \hline & 3100 \end{array}$$

Если обозначить основание искомой системы счисления — p, то можно записать:

$$31_p = 16.$$

Применив так называемую "развернутую" запись p-ичного числа:

$3p + 1 = 16$

— можно определить, что $p = 5$.

Ответ: это могло быть, если все числа записаны в пятеричной системе счисления.

Правильные ответы прислали:

— Абраменко Богдан, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Вуколов Сергей, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

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

— Карданова Аминат, Коломина Нонна, Медяникова Аделина, Остроухова Валерия и Уткина Ксения, Ставропольский край, Кочубеевский р-н, станица Барсуковская, школа № 6, учитель **Рябченко Н.Р.**;

— Мазанова Екатерина, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Новиков Алексей, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Чукмасова Надежда, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Краснёнкова Л.А.**;

— Яковлева Александра, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**

Переставить четыре шашки

Напомним, что требовалось разработать алгоритм перестановки по правилам уголков четырех шашек из левого нижнего угла доски — полей al, a2, bl, b2 в правый верхний.

Решение

Задача решается за 13 ходов (возможны и другие варианты):

1. a2–c2.
2. al–cl–c3.
3. bl–b3–d3.
4. b2–d2–d4.
5. c2–c4–e4.
6. c3–e3–e5.
7. d3–d5–f5.
8. d4–f4–f6.
9. e4–e6–g6.
10. e5–g5–g7.
11. f5–f7–h7.
12. f6–h6–h8.
13. g6–g8.

Ответы представили:

— Березин Василий, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

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

— Краснова Диана, Свердловская обл., г. Ревда, школа № 10, учитель **Игошева А.А.**;

— Листов Алексей, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Пилясов Иван, г. Ярославль, школа № 33, учитель **Цикина Е.Н.**;

— Хвойновский Вадим, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**

Диана Краснова и Иван Пилясов, у которых алгоритм решения задачи состоит из 13 шагов, будут награждены дипломами. Молодцы!

Алгоритм перестановки вагонов на так называемом "маневровом треугольнике" разработали:

— Андриенко Василий, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Березин Василий, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Викторов Даниил, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

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

— Краснова Диана, Свердловская обл., г. Ревда, школа № 10, учитель **Игошева А.А.**;

— Пилясов Иван, г. Ярославль, школа № 33, учитель **Цикина Е.Н.**;

— Хвойновский Вадим, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**;

— Цаплин Константин, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**

Японские головоломки “судоку”, опубликованные в октябрьском выпуске “В мир информатики”, решили:

— Волк Анастасия, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Золотова Надежда, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Фомичева Дарья, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина, учитель **Чапкевич И.М.**;

— Храмцов Алексей, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**

Пять задач на системы счисления. Часть 2

Ответы представили:

— Абраменко Богдан, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Андросенко Дмитрий, средняя школа села Дохновичи Брянской обл., учитель **Клопова Е.В.**;

— Вуколов Сергей, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

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

— Зубов Владислав, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Новиков Алексей, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Пак Александра, Пантелюк Руслан и Приходько Геннадий, г. Пионерский Калининградской обл., школа № 2, учитель **Багрова О.А.**;

— Яковлева Александра, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**

Александра Яковлева, Дмитрий Андросенко и Алексей Новиков, правильно решившие все задачи, будут награждены дипломами. Поздравляем!

Пять задач на системы счисления. Часть 3

Е.А. Мирончик,
учитель информатики лицея № 111,
г. Новокузнецк Кемеровской обл.

1. Выпишите числа $2^{n+1} - 1$ и $2^n + 2^{n-1}$, $2^n + 2^{n-1} + 2^{n-2} + \ldots + 2^2 + 2^1 + 2$, где $n > 10$, в порядке возрастания количества единиц в их двоичной записи. Ответ обоснуйте.

2. Каждую строку черно-белого рисунка размером $n \times n$ пикселей закодировали следующим образом: пиксель черного цвета обозначили единицей, белого — нулем и соответствующее число представили в шестнадцатеричной системе счисления. Весь рисунок при этом кодируется последовательностью шестнадцатеричных чисел: 1C6, 129, 121, 122, 1C1, 189, 149, 126.

Что изображено на рисунке?

Аналогично можно этот рисунок закодировать в восьмеричной системе счисления. Перекодируйте рисунок соответствующим образом.

3. Два некоторых десятичных числа X и Y перевели в системы счисления с основаниями 16 и 8. Часть символов при записи была утеряна. Два из четырех полученных чисел имеют вид (позиции утерянных символов обозначены символом “*”):

$$34*_{16} \text{ и } 16**_{8}.$$

Можно ли сделать вывод о том, какое из чисел X и Y больше, или о равенстве этих чисел?

4. Некоторое десятичное число X перевели в системы счисления с основаниями 16 и 8. Часть символов при записи была утеряна. Позиции утерянных символов обозначены символом “*”:

$$X = 1*0_{16} = 56*_{8}.$$

Восстановите все числа и определите число X.

5. Некоторое десятичное число X перевели в системы счисления с основаниями 16 и 8. Часть символов при записи утеряна (позиции утерянных символов обозначены символом “*”):

$$*A_{16} \text{ и } ***_{8}.$$

Можно ли однозначно определить значение числа X? Если нет, то укажите все возможные значения.

От редакции. Методика решения задач описана в выпуске “В мир информатики” № 201 (“Информатика” № 10/2014).

Нет без явно усиленного трудолюбия ни талантов, ни гениев.

Дмитрий Иванович Менделеев

Образованность и интеллектуальное развитие — это… естественное состояние человека, а невежество — состояние ненормальное. Невежество и полузнайство — это почти болезнь.

Дмитрий Сергеевич Лихачев

Наука всегда будет порождать личности, для которых Истина превыше всего.

Даниил Гранин

У человечества есть только один тиран — невежество.

Виктор Гюго

ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ г. МОСКВЫ
ИЗДАТЕЛЬСКИЙ ДОМ «ПЕРВОЕ СЕНТЯБРЯ»
МОСКОВСКИЙ ПЕДАГОГИЧЕСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

МОСКОВСКИЙ ПЕДАГОГИЧЕСКИЙ МАРАФОН УЧЕБНЫХ ПРЕДМЕТОВ 2015

23 МАРТА – 17 АПРЕЛЯ

РАСПИСАНИЕ ДНЕЙ ПЕДАГОГИЧЕСКОГО МАРАФОНА

23 марта	День учителя технологии *
24 марта	**Открытие Марафона** День классного руководителя
25 марта	День школьного психолога День учителя ОБЖ
26 марта	День здоровья детей, коррекционной педагогики, логопеда, инклюзивного образования и лечебной физической культуры
27 марта	День учителя начальной школы (день первый)
28 марта	День учителя начальной школы (день второй)
29 марта	День дошкольного образования
31 марта	День учителя географии
1 апреля	День учителя химии
2 апреля	День учителя биологии
3 апреля	День учителя информатики
4 апреля	День учителя физики
5 апреля	День учителя математики
7 апреля	День учителя истории и обществознания
8 апреля	День учителя МХК, музыки и ИЗО
9 апреля	День школьного и детского библиотекаря
10 апреля	День учителя литературы
11 апреля	День учителя русского языка
12 апреля	День учителя английского языка
14 апреля	День учителя французского языка
15 апреля	День школьной администрации
16 апреля	День учителя физической культуры
17 апреля	День учителя немецкого языка **Закрытие**

marathon.1september.ru

! Обязательная предварительная регистрация на все дни Марафона с 20 февраля 2015 года на сайте marathon.1september.ru

! Каждый участник Марафона, посетивший три мероприятия одного дня, получает официальный именной сертификат (6 часов)

В дни Марафона ведущие издательства страны представляют книги для учителей

Начало работы каждого дня – 9.00. Завершение работы – 15.00

УЧАСТИЕ БЕСПЛАТНОЕ. ВХОД ПО БИЛЕТАМ

РЕГИСТРИРУЙТЕСЬ, РАСПЕЧАТЫВАЙТЕ СВОЙ БИЛЕТ И ПРИХОДИТЕ!

Место проведения Марафона: МПГУ, ул. Малая Пироговская, дом 1, стр. 1 (в 5 минутах ходьбы от ст. метро «Фрунзенская»)
* Место проведения Дня учителя технологии: ЦО № 293, ул. Касаткина, 1а (ст. метро «ВДНХ»)

По всем вопросам обращайтесь, пожалуйста, по телефону **8-499-249-3138** или по электронной почте marathon@1september.ru