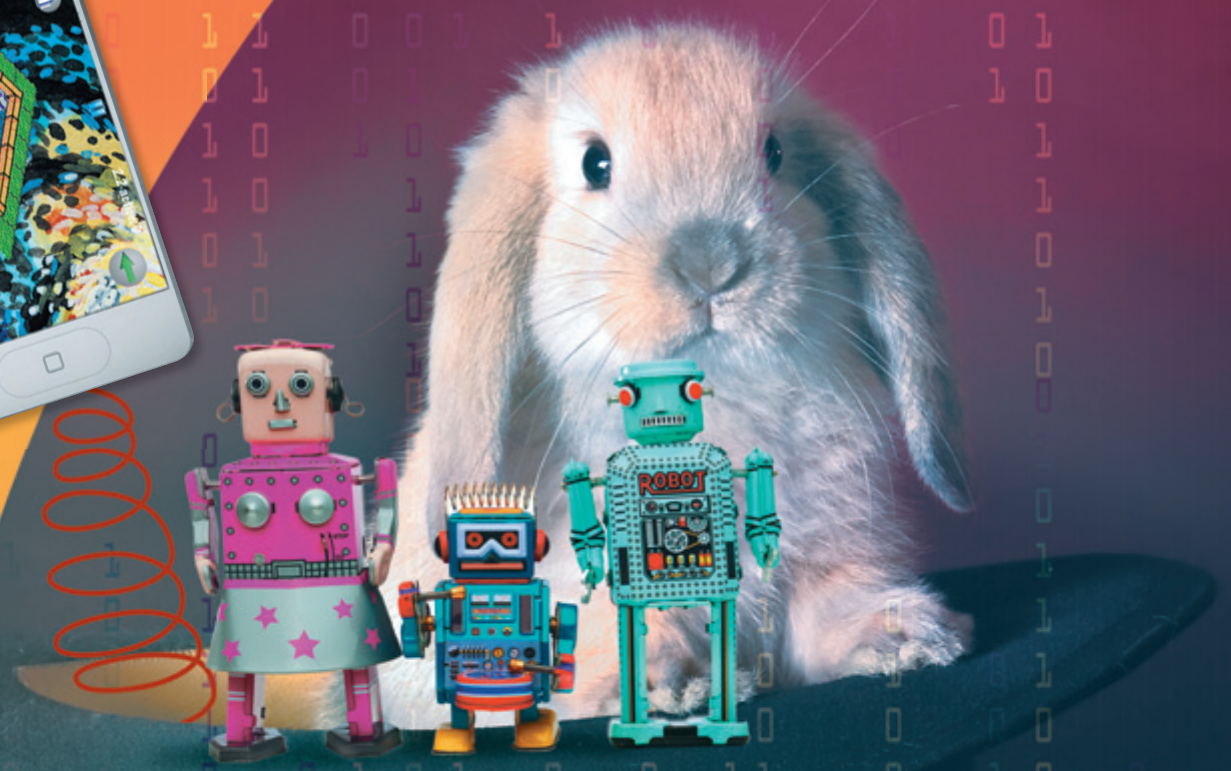
№ 15 Издание основано в 1995 г. inf.1september.ru
УЧЕБНО-МЕТОДИЧЕСКИЙ ЖУРНАЛ ДЛЯ УЧИТЕЛЕЙ ИНФОРМАТИКИ

ИНФОРМАТИКА



издательский дом 1september.ru
Первое сентября | октябрь 2011
ИНФОРМАТИКА Подписка: «Роспечать» – 32291 (бумажная версия), 19179 (электронная); «Почта России» – 79066 (бумажная версия), 12684 (электронная)

О НОМЕРЕ

Предлагаемый вашему вниманию номер получился очень сбалансированным. Причем, что хочется отметить особо, он сбалансирован практически по всем параметрам — темам, сложности, возрасту детей. Особое внимание хочется обратить на статью, посвященную фибоначчиевой системе счисления. Это очень "богатый" материал — он будет интересен и тем, кто работает в 8–9-х классах (в которых, как правило, рассматривается тема "Системы счисления"), и тем, кто уже со старшими школьниками хочет "копнуть поглубже".

Еще одна публикация, которую хочется отметить особо, — статья, посвященная "ПиктоМиру". Хотя сам программный продукт еще находится в разработке, его можно установить, попробовать — все работает! И опыт показывает, что эта "игровая" среда интересна не только дошкольникам и младшим школьникам, но и учащимся старших классов, желающим "поломать голову".

В НОМЕРЕ

НА ДИСКЕ

ЭЛЕКТРОННЫЕ МАТЕРИАЛЫ:

- Авторская сборка среды System Builder к статье "ООП без П"
- Установочный пакет среды ПиктоМир к статье "Программирование для дошкольников и младших школьников"
- Исходные файлы к статье "Excel. Поиск решения"
- Дополнительные материалы к статье "Кодирование информации: фибоначчиева система и компьютер"
- Презентации к статьям номера

Уважаемые подписчики бумажной версии журнала "Информатика"!

В течение второго полугодия 2011 г. вы можете получать и электронную версию нашего журнала в Личном кабинете. Электронная версия в Личный кабинет всегда доставляется точно в срок — 1-го числа каждого месяца.

1. Зайдите на интернет-сайт *www.1september.ru*.
2. Зарегистрируйте Личный кабинет (если у вас его еще нет).
3. В Личном кабинете в разделе "Издания/Коды доступа" введите код SE-00332-68580.

Номера будут доступны в разделе "Издания/Получение".

ИНФОРМАТИКА

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

http://inf.1september.ru

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

РЕДАКЦИЯ:
гл. редактор С.Л. Островский
редакторы
Е.В. Андреева,
Д.М. Златопольский (редактор вкладки "В мир информатики")
Дизайн макета И.Е. Лукьянов
верстка Н.И. Пронская
корректор Е.Л. Володина
секретарь Н.П. Медведева
Фото: фотобанк Shutterstock
Журнал распространяется по подписке
Цена свободная
Тираж 3000 экз.
Тел. редакции: (499) 249-48-96
E-mail: inf@1september.ru
http://inf.1september.ru

ИЗДАТЕЛЬСКИЙ ДОМ "ПЕРВОЕ СЕНТЯБРЯ"
Главный редактор: Артем Соловейчик (генеральный директор)
Коммерческая деятельность: Константин Шмарковский (финансовый директор)
Развитие, IT и координация проектов: Сергей Островский (исполнительный директор)
Реклама и продвижение: Марк Сартан
Мультимедиа, конференции и техническое обеспечение: Павел Кузнецов
Производство: Станислав Савельев
Административно-хозяйственное обеспечение: Андрей Ушков
Главный художник: Иван Лукьянов
Педагогический университет: Валерия Арсланьян (ректор)

ГАЗЕТА ИЗДАТЕЛЬСКОГО ДОМА
Первое сентября – Е.Бирюкова

ЖУРНАЛЫ ИЗДАТЕЛЬСКОГО ДОМА
Английский язык – А.Громушкина
Библиотека в школе – О.Громова
Биология – Н.Иванова
География – О.Коротова
Дошкольное образование – М.Аромштам
Здоровье детей – Н.Сёмина
Информатика – С.Островский
Искусство – М.Сартан
История – А.Савельев
Классное руководство и воспитание школьников – О.Леонтьева
Литература – С.Волков
Математика – Л.Рослова
Начальная школа – М.Соловейчик
Немецкий язык – М.Бузоева
Русский язык – Л.Гончар
Спорт в школе – О.Леонтьева
Управление школой – Я.Сартан
Физика – Н.Козлова
Французский язык – Г.Чесновицкая
Химия – О.Блохина
Школьный психолог – И.Вачков

УЧРЕДИТЕЛЬ:
ООО "ЧИСТЫЕ ПРУДЫ"

Зарегистрировано ПИ № ФС77-44341 от 22.03.2011 в Министерстве РФ по делам печати
Подписано в печать: по графику 13.09.2011, фактически 13.09.2011
Заказ №
Отпечатано в ОАО "Чеховский полиграфический комбинат" ул. Полиграфистов, д. 1, Московская область, г. Чехов, 142300

АДРЕС ИЗДАТЕЛЯ:
ул. Киевская, д. 24, Москва, 121165
Тел./факс: (499) 249-31-38
Отдел рекламы: (499) 249-98-70
http://1september.ru

ИЗДАТЕЛЬСКАЯ ПОДПИСКА:
Телефон: (499) 249-47-58
E-mail: podpiska@1september.ru

В этом номере рубрика “Пара слов” получилась несколько телеграфной. Что, может быть, и неплохо — точное попадание в формат рубрики. Но на самом деле просто много о чем хочется [рас]сказать, и делать это приходится коротко.

Электронная версия тчк

Уважаемые коллеги, подписчики бумажной версии “Информатики”! Пожалуйста, обратите внимание на блок с кодом доступа к электронной версии, размещенный на второй странице. В Личном кабинете на сайте *www.1september.ru* вы имеете возможность получать электронную версию журнала 1-го числа каждого месяца. Как бы мы ни колдовали и “напрягали” почту, бумажная версия не доставляется раньше 15-го числа, реально — позже. А введя код единожды, вы получите доступ ко всем номерам полугодия точно в срок.

Суперцена на следующее полугодие тчк

Подписка на электронную версию “Информатики” с доставкой в Личный кабинет на следующее полугодие стоит всего 200 рублей. За полгода. Это не ошибка.

Появилась электронная версия на CD тчк

Имеются три варианта подписки на следующее полугодие — они описаны на последней странице этого номера. Одно из новшеств — появление CD-версии с доставкой по почте. Разумеется, для всех вариантов включен доступ к версии в Личном кабинете.

В Личном кабинете можно оплатить карточкой тчк

Заказ, сделанный в Личном кабинете, можно оплатить не только по квитанции в банке, но и по кредитной карте в режиме on-line. Возможность оплаты картой зависит от вида карты и от банка, “обычные” карты Visa и Mastercard замечательно работают. С Maestro хуже — лишь не так давно стали выдавать Maestro с возможностью использования для интернет-платежей.

“Школа цифрового века” — и себе и людям тчк

Если ваша школа примет участие в общероссийском проекте “Школа цифрового века”, не только вы практически бесплатно получите полугодовую подписку на “Информатику”, но и все ваши коллеги на любое количество журналов по их выбору. Стоимость участия школы — всего 2000 рублей. Цена фиксированная, от количества педагогов в школе не зависит. При этом все получают всё. Все, что хотят, разумеется. Подробности — на *digital.1september.ru*.

Цикл курсов “Навыки личной эффективности” тчк

Обязательно обратите внимание на новый цикл курсов Педагогического университета “Навыки личной эффективности”, информация о котором опубликована на с. 35. Это очень хорошие и интересные материалы. Стоимость каждого курса — всего 200 рублей. Зачет — в форме теста, который можно пройти в Личном кабинете в режиме on-line.

ООП без П

Е.А. Еремин,
г. Пермь

Введение

Объектно-ориентированное программирование (ООП) — одна из наиболее используемых в данный момент технологий программирования. Поэтому наряду с общими представлениями о классических методах программирования в хорошем курсе информатики очень бы хотелось поговорить и об основах ООП. К тому же объектный подход сам по себе применяется не только в программировании, но и лежит в основе всего современного графического интерфейса, что еще более повышает значимость этой тематики для компьютерного образования.

В то же время, преподавание этой темы наталкивается на целый ряд трудностей. Наиболее принципиальная из них заключается в том, что преимущества ООП видны только при написании сложных программных систем, а приводимые при первоначальном знакомстве примеры должны быть как можно проще и короче[1].

Более того, если решить одну и ту же *простую* задачу традиционным способом и с помощью средств ООП, то текст объектно-ориентированной программы окажется длиннее и потребует знания большего числа конструкций языка. Справедливости ради тороплюсь сказать, что наиболее громоздкая часть с описаниями объектов является универсальной и может без всяких изменений использоваться для очень быстрого решения аналогичных задач. Но едва ли на начальном этапе учащиеся глубоко прочувствуют преимущества повторного использования написанных и отлаженных ранее модулей.

Положение усугубляется еще и тем, что ООП часто подменяют визуальным программированием. Порой считается, что когда ученик размещает на форме несколько кнопок, он получает представление об ООП[2]. В итоге визуальные приемы работы с системой, равно как и необходимость подробных описаний в применяемом языке программирования, указанная выше, заслоняют от ученика существенные идеи ООП.

[1] Обсуждение проблемы сложности подбора примеров при изучении ООП можно найти также в публикации автора [1].

[2] Нет сомнения в том, что ООП при этом используется; проблема в том, что ученик этого не видит, ибо иерархия визуальных компонентов была построена до него и без него.

Неудивительно, что после всего этого слова “инкапсуляция” и “полиморфизм” прозвучат так пугающе.

Таким образом, налицо сложная образовательная проблема: имеется важный материал, который необходимо изучить, но простыми способами вроде “делай как я” это получается не очень… Между прочим, перед нами типичная ситуация, когда отчетливо видна роль частной методики преподавания: преподавателю мало просто знать материал, необходимо еще четко представлять, как лучше его изложить.

В данной статье сделана попытка подойти к изучению ООП не совсем обычным образом. Предлагается делать это в ходе построения на экране дисплея некоторого *виртуального мира*, в котором “живут и взаимодействуют” *виртуальные объекты*: нарисованные машины, зверюшки, ракетки, логические элементы и многое-многое другое, причем выбор ограничен лишь фантазией пользователя и наличием у него необходимых картинок. Важной особенностью предлагаемой методики является также отсутствие привязки к какому-либо конкретному языку вроде Object Pascal или C++. Несмотря на кажущуюся “несерьезность” подхода, при продуманном использовании он, по моему мнению, позволяет рассмотреть практически все основные базовые идеи ООП (чтобы вспомнить их перечень, читатели журнала могут обратиться к недавней публикации [2]).

Конечно, чтобы строить виртуальные миры из объектов, необходимо специальное программное обеспечение. Авторский вариант его реализации можно найти на прилагаемом диске и опробовать. Он назван **System Builder** (SB), т.е. “*Построитель (конструктор) систем*” (из объектов). Программа имеет длинную и не всегда безоблачную историю. Впервые о ней было рассказано на конференции ИТО в 1999 году [3], но выступление не вызвало никакого интереса. В 2000 году англоязычная версия системы представлялась на образовательной конференции в Финляндии [4] и по крайней мере привлекла внимание отдельных преподавателей вузов. В то время я увлекался распространением своего учебного ПО через Интернет и получил несколько положительных отзывов от тех, кто познакомился с SB (впрочем, отдельные критические замечания, честно говоря, тоже были). Затем в результате личных контактов мне удалось заинтересовать своей системой некоторых авторов школьных учебников. В результате довольно длительного сотрудничества SB был существенно улучшен, и появилась его *вторая* версия. Но и она особого распространения также не получила. Возможно, программу надо переписать как-то по-другому, или, быть может, дело в чем-то ином. Тем не менее недавние бурные дискуссии об изложении идей ООП при написании материалов [2] освежили в памяти старые идеи, и по сути своей они мне по-прежнему кажутся разумными. В общем, вниманию читателей предлагаются:

- некоторые оригинальные идеи преподавания ООП;
- программная поддержка для этого (возможно, не лучшая).

Я вовсе не претендую на “законченность и мощь учения” и буду рад даже в том случае, если что-нибудь из предложенного поможет отдельным преподавателям. И буду рад вдвойне, если кто-то из читателей сумеет развить или модифицировать изложенный подход, а может быть, даже напишет более востребованное программное обеспечение и поделится им с нами.

1. Идеи методики

1.1. Что мы хотим изучить

Статья не ставит целью дать описание базовых принципов ООП, тем более что в недавней публикации [2] уже шла об этом речь. Поэтому просто кратко сформулируем, что именно мы хотим продемонстрировать при изучении основ ООП:

- основной тезис ООП: мир состоит из объектов, которые могут появляться, исчезать, изменяться, двигаться и взаимодействовать между собой;
- похожие объекты объединяются в группы, которым даются названия (кошки, дома, стулья и т.д.); мы будем называть такие группы **типами** объектов (в программировании чаще используется термин *класс*);
- любой **объект** — *экземпляр* какого-то типа; чтобы различать объекты, мы будем давать каждому из них неповторяющиеся имена;
- объекты имеют свойства и методы, благодаря которым они могут взаимодействовать с другими объектами; к свойствам и методам данного объекта другие объекты обращаются только по имени, поэтому никакие изменения внутри объекта не потребуют ни малейших изменений в остальных частях системы (принцип **инкапсуляции**);
- **свойства** объектов описывают их параметры; они имеют некоторое значение (в SB это вполне конкретное число или текстовая строка);
- **методы** описывают поведение объекта; это определенная последовательность (программа) действий, поэтому для их описания всегда требуется некоторый язык;
- мир устроен так, что типы образуют иерархию (многоуровневую структуру); в ООП типы-потомки обладают всеми свойствами и методами типов-предков, к которым добавляются их собственные свойства и методы (принцип **наследования**);
- каждый тип объектов может выполнять рассматриваемое действие по-своему, т.е. в каждом из них метод с данным именем может быть написан особым образом (принцип **полиморфизма**).

Таким образом, в основе объектно-ориентированной системы должны лежать концептуальные понятия **объект**, **тип объекта** (класс), **свойство** и

метод. Причем, как видно из приведенного выше перечня базовых принципов, конкретный язык программирования особого значения не имеет.

А теперь подумаем, как все это можно продемонстрировать.

1.2. Как мы будем изучать ООП

Мы будем изучать ООП, создавая некоторый виртуальный мир, состоящий из объектов. Каждый из них будет принадлежать к определенному типу (классу) и, в строгом соответствии с этой принадлежностью, обладать определенными свойствами и методами. Последние нам придется писать, используя некоторый простейший интуитивно понятный язык; это единственное место, которое напоминает о программировании.

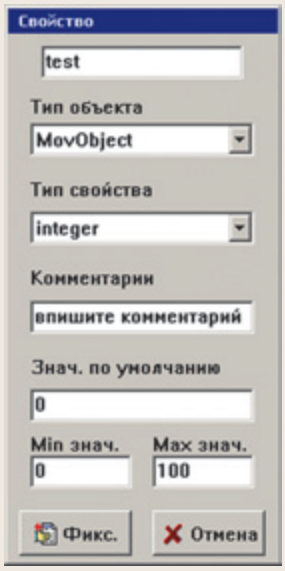


Рис. 1. Создание нового свойства при помощи заполнения “электронной формы”

Чтобы избавиться при конструировании нашего мира от многочисленных описаний на формальном языке, применим принцип заполнения “**электронных форм**” (это известный прием: например, в Delphi подобные формы называются *экспертами*). Несколько забегая вперед, покажем на *рис.* 1, как может выглядеть создание нового свойства без длинных и скучных текстовых описаний. Заносим в верхнее поле имя свойства, которое нам нравится; выбираем из списков[3] тип объекта, к которому будет относиться наше новое свойство, а также тип самого свойства и, наконец, заполняем оставшиеся поля. Нажимаем кнопку “Фикс.”, фиксируя создаваемое свойство. Система проверяет вводимые значения на допустимость (например, если свойство test уже имеется, то она сообщит нам об этом) и сохраняет “внутри себя” информацию о новом свойстве. Системы типа Delphi могут при этом генерировать некоторый текст на Паскале, но в нашем случае это излишне (чем меньше текста — тем проще ориентироваться в проекте). Для обзора структуры проекта везде будем использовать таблицы (их образцы можно посмотреть на *рис.* 5 и 6 на с. 8, 9).

Еще раз подчеркнем, что мы сознательно отходим от конкретных языков программирования и концентрируем все свое внимание на базовых идеях ООП.

А как же мы будем создавать виртуальный мир? Предлагается выделить в этом процессе три следующих взаимосвязанных этапа.

1. **Конструируем** мир. На этом этапе определяются типы имеющихся в мире объектов (например, машины и автобусы), а также их свойства и методы. Очень важно создавать новый тип от наиболее близкого из уже имеющихся[4]. В каком-то смысле это создание плана (эскиза) нашего будущего мира.

2. **Строим** мир: создаем конкретные объекты и задаем значения их свойств. На этом этапе для сконструированных ранее типов создаются их экземпляры (конкретные объекты, например, машины car1, car2 и т.д.). В нашем мире уже появляются видимые объекты, но пока они “неживые” — статичные и неизменные.

3. **Наблюдаем**, что получилось. Наконец-то наш виртуальный мир готов и можно посмотреть, так ли он движется и развивается, как мы запланировали.

Конечно, лишь в идеале этот трехступенчатый алгоритм будет последовательным. Например, при задании свойств объектов может обнаружиться, что мы забыли предусмотреть какое-то свойство, так что придется вернуться назад и добавить его, изменив тем самым предварительную конструкцию типа. А при наблюдении может оказаться, что поведение объектов не такое, как ожидалось. И здесь придется вернуться к более ранним этапам и внести необходимые исправления. Разумеется, система должна “прощать” начинающим допущенные при конструировании неточности и позволять их легко исправить. А еще учебная система должна иметь в своем составе хорошие средства для поиска ошибок при отладке проекта.

Теперь, когда базовые принципы изучения материала определены, потребуется программная система, которая их реализует. Далее будет описан возможный вариант такой системы под названием “SB” (еще раз подчеркнем, что он не единственно возможный!).

Раньше то обстоятельство, что SB **предназначен для работы в операционной системе Windows**, можно было особенно и не подчеркивать, но сейчас, когда определенное распространение получила ОС Linux, это стоит сделать.

2. Программная поддержка

2.1. Знакомство с SB

Все понимают, что знакомиться со сложной системой можно по-разному, но приятнее всего, когда это можно делать непосредственно работая с ней. Надеюсь, SB получилась достаточно простой, чтобы допускать такую возможность.

Программа не нуждается в специальной инсталляции. Для работы с ней просто скопируйте все папки и файлы на свой компьютер. Рекомендую произвести полное копирование папки SB и не изменять имеющиеся имена папок и файлов.

[3] Выбор из списка облегчает заполнение формы и уменьшает количество ошибок.

[4] В любой системе даже в “пустом” проекте уже есть стандартные типы объектов.

Теперь найдите в папке самого верхнего уровня исполняемого файла sbr.exe (имя sbr означает русскую версию SB) и запустите его. Вы увидите следующую приветственную картинку в главном окне (*рис.* 2).

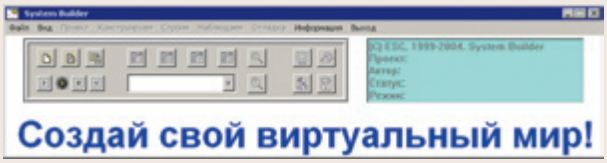


Рис. 2. SB приветствует вас!

Обратите также внимание на то, что программа специально создала светлый задний фон (на *рис.* 2 он не показан), закрывающий элементы рабочего стола компьютера. Тем самым ПО предлагает нам не отвлекаться и сконцентрировать все свое внимание на изучении ООП.

Итак, что есть в главном окне? В верхней части окна расположено традиционное главное меню. Бóльшая часть его пунктов сейчас недоступна, но это не повод для беспокойства: система пока “пуста” — в ней нет никакого проекта. Ниже справа располагается маленький экран с информацией о текущем проекте, а слева — небольшая панель инструментов. Как обычно, оперативные кнопки панели инструментов дублируют наиболее важные действия главного меню и ускоряют их запуск. Заметим, что панель, как легко заметить, разбивается на три зоны:

1) левая — для манипуляции с проектами: создание нового проекта, загрузка и сохранение существующего, а также кнопки управления просмотром (обозначения общепринятые, как у любого видеопроигрывателя, — “старт”, “стоп”, “на начало”; соответственно, располагающийся на этой же линии индикатор горит зеленым, красным или желтым в состояниях “работа”, “стоп” или “пауза”);

2) центральная — работа со свойствами, методами и типами объектов (этап конструирования), а также с самими объектами (строительство); здесь же можно переключать режим работы системы — об этой возможности поговорим позднее;

3) правая — кнопки, используемые при отладке проекта.

Помимо главного, в SB имеется целый ряд вспомогательных окон. Бóльшая их часть содержит автоматически обновляемые таблицы, отражающие текущее состояние проекта. Навигация между таблицами на первый взгляд может показаться сложной и хаотичной, но на самом деле таблицы четко сгруппированы *в соответствии с тремя этапами строительства виртуального мира*, сформулированными выше. Так, одно окно объединяет работу с типами, их свойствами и методами (конструирование мира), а второе — с объектами и значениями их свойств (создание мира). Для этапа наблюдения предусмотрено наиболее интересное для пользователей окно — экран проекта. К тому же быстро попасть на нужную вкладку окон позволяет панель инструментов: щелчком по кнопкам, содержащим на фоне пиктограммы с таблицей буквы “C”, “M”, “T” и “O”, можно сразу выйти на таблицы свойств, методов, типов и объектов соответственно. Окно *Инспектора типа* (оперативная кнопка с лупой и буквой “T”) показывает соответствие между типами объектов и их свойствами и методами, а также полезную информацию по наследованию. Наконец, окно *Инспектор объекта* (лупа с буквой “O”) во многом напоминает окно *Object Inspector* в Delphi.

Еще имеется группа отладочных окон, о которых будет кратко рассказано в разделе 2.6.

Более подробно о назначении каждой конкретной кнопки в каждом конкретном окне можно прочитать в детальном файле помощи, входящем в систему[5]. Помощь вызывается как из меню главного окна (пункт “Информация”), так и из других окон программы, причем в последнем случае обеспечивается поиск нужной страницы текста (контекстно-зависимая организация помощи).

Рис . 3. Демонстрационный проект “Перекресток”

Тем не менее я предлагаю начать знакомство не с чтения описания, а с более простых и приятных вещей. В SB “встроены” более двух десятков готовых проектов. Они условно разбиты на две группы: *примеры* и *демонстрации*. Первые имеют простое устройство и в них легко разобраться, но они не очень эффектно выглядят. Вторые, напротив, достаточно сложны, но смотрятся красиво и вызывают интерес. Начать, конечно, приятнее с демонстраций: они покажут, что в состоянии сделать SB при его глубоком освоении. Далее можно запустить примеры. В их текст уже можно попытаться заглянуть (примеры в меню располагаются в порядке увеличения сложности).

[5] Исторически файл первоначально появился в англоязычной версии программы, что, к сожалению, может где-то проявиться в виде остатков английского текста.

Пункты “Запустить пример” и “Запустить демонстрацию” вы легко найдете в меню “Файл”. Стоит иметь в виду, что на диске есть дополнительные примеры, которые в меню не включены. Их легко загрузить через пункт “Открыть проект…”.

Познакомившись с готовыми проектами, вы уже поймете, интересен ли вам SB. И если да, то продолжите знакомство. Вот теперь самое время почитать файл помощи, причем я убедительно рекомендую не пропустить раздел “Принципы SB”.

Еще один необязательный предварительный шаг — разобраться с примерами и, может быть, модифицировать их. В качестве альтернативы можно начать с разбираемых ниже примеров. Наконец, “набив руку” в работе с системой, можно приступить к реализации собственного виртуального мира!

2.2. Первый проект

Создадим в SB простейший проект, что-то вроде ООП-версии известной задачи “Hello, world”. После непродолжительных размышлений, я полагаю, вы, как и я, придете к выводу, что в нашей системе это должен быть просто движущийся объект. Как положено, в итоге мы действительно напишем всего один оператор, чтобы получить решение задачи.

Итак, запускаем SB и создаем новый проект (через меню “Файл” или лучше через соответствующую оперативную кнопку). В результате автоматически загружается стандартный шаблон проекта (файл standard.sb), и на экране появляется начальная картинка, являющаяся символическим “описанием” системы (см. *рис.* 4). Начнем с того, что сохраним проект под своим собственным именем, используя пункты меню “Файл” — “Сохранить проект как” — “только проект…”. В диалоге сохранения укажите произвольную папку (я создал для упражнений в общем каталоге папку test и сохранил туда). Картинка при этом исчезнет, но она нам все равно не нужна — мы потом используем свою.

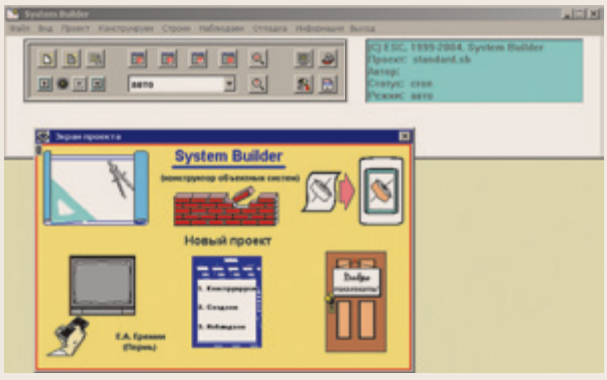


Рис. 4. Создаем новый проект

Заглянем в папку: там появилось два файла — test.sbf (это собственно проект) и test.opt (служебный файл, сохраняющий настройки системы для данного проекта). По правде говоря, это текстовые файлы, и их можно было бы и посмотреть, но мы пока для этого слишком мало знаем о системе.

Итак, проект создан. Позаботимся теперь о его “художественном сопровождении” — подберем картинки. Их можно взять из любого клипарта или нарисовать самому. Но я предлагаю поступить проще. Откроем в SB папку samples\arrow и скопируем оттуда в наш проект файлы wallpap.bmp и pic1.bmp. Первый — это фоновая картинка (wallpaper по-английски “обои”), а второй — первая и единственная картинка проекта (следующая должна иметь имя pic2.bmp и т.д.). Конечно, SB пока “не заметил” наших манипуляций с файлами, но если повторно загрузить проект, то он установит нашу фоновую картинку в виде мишени и “подгонит” под ее размер экран проекта. А еще можно заглянуть в проект (главное меню или правая верхняя оперативная кнопка) и на вкладке “Фон” увидеть фоновый рисунок, а на вкладке “Образы” — неиспользуемый пока рисунок 1 в виде стрелы. Эта самая стрела и полетит в мишень. Но как ее запустить, пока непонятно. Что же, заглянем, что предоставил в наше распоряжение стандартный “пустой” проект.

Удобнее всего начать с типов объектов. Чтобы их посмотреть, найдите в центральной части панели инструмента кнопку, имеющую на пиктограмме букву “T”, и щелкните по ней. Вы увидите очень интересную таблицу (см. *рис.* 5).

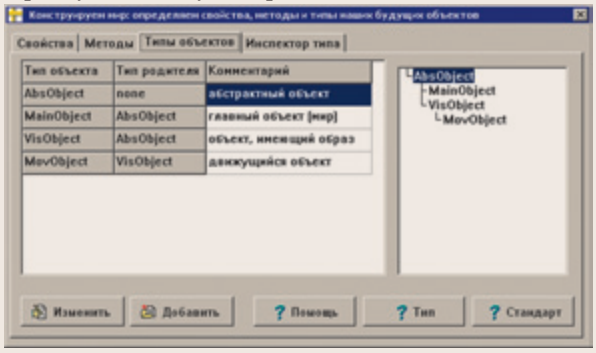


Рис. 5. Стандартные типы в SB

Оказывается, система уже заготовила для нас мини-иерархию типов. Ее вершиной является всеобщий предок — абстрактный тип объектов AbsObject. От него созданы два других — главный объект MainObject (его единственный предопределенный экземпляр — это весь виртуальный мир) и визуальные объекты VisObject. От последнего типа, в свою очередь, порожден тип MovObject, т.е. движущиеся (и благодаря наследованию свойств визуальные!) объекты. Наша стрела, без сомнения, должна иметь тип MovObject. Запомним это и заглянем еще в одну полезную таблицу. Откроем *Инспектор типа* (кнопка с лупой и буквой “T”) и в имеющемся на вкладке списке выберем интересующий нас тип. Не без удивления обнаружим, что там уже “запасено” 11 свойств (из них 9 наследуются) и 5 методов (наследуются 2). Нам для проекта понадобится метод _xmove (имена всех методов в SB начинаются со знака подчеркивания, чтобы легче было отличить их от свойств). Нетрудно дога-

даться, что именно он способен организовать движение вдоль оси X.

И еще, “пока далеко не ушли”, загляните в *Инспекторе типа* на MainObject. Вы обнаружите в нем кучу свойств, на которые мы пока не будем обращать внимания, и единственный заранее предусмотренный метод _main. Этот замечательный метод SB регулярно вызывает через равные промежутки времени, позволяя организовать любые изменения в состоянии виртуального мира. Таким образом, этап конструирования для нашего примитивного примера оказался не нужен: все типы и методы уже есть! Надеюсь, найденные имена методов вы тоже запомнили (или записали).

Теперь начинаем действовать. Создадим новый объект — стрелу. Для этого щелкнем кнопку, содержащую букву “О”, и увидим таблицу с имеющимися объектами. Точнее, пока с единственным объектом World (мир). Сообщим системе, что мы хотим расширить таблицу, щелкнув кнопку “Добавить”. Заполним появившуюся форму: объект назовем a, выберем нужный его тип и впишем необязательный комментарий (см. *рис.* 6). Нажмем кнопку “Фикс.” и тем самым создадим объект. Система подтвердит, что объект создан, и предложит установить его свойства, автоматически открыв нужное окно.

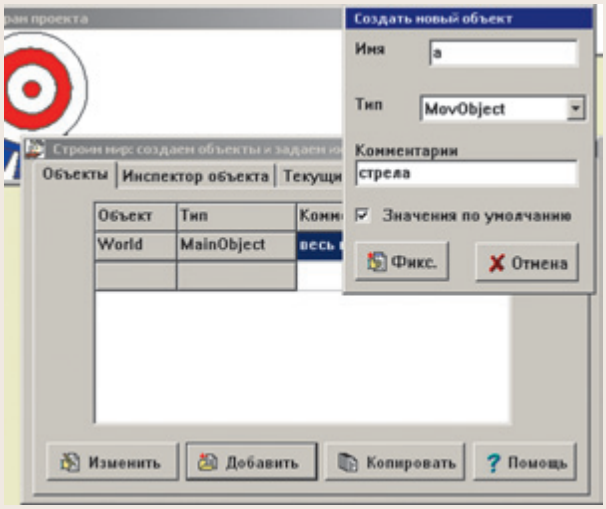


Рис. 6. Создание нового объекта

Начнем с картинки. Щелкнем по кнопке “Рисунок” и в новом появившемся окне по единственной картинке со стрелой (да, кстати, запишите размеры рисунка, они нам потом пригодятся!). Подтвердим выбор. Рисунок появится в текущем окне *Инспектора объекта* и (ура!) на экране проекта. Правда, пока в левом верхнем углу. Исправим это, вписав в свойство left (расстояние от левой границы) значение 300, а top (расстояние от верхней границы) — число 44. Нажмем кнопку “Фикс. измен.” (фиксировать изменения). Наша стрела переместится, но она выглядит какой-то “урезанной”. Дело в том, что по умолчанию ее размер настроен на 32 × 32 (типовой размер иконок), а у нас картинка 76 × 14. Введем в свойства width (ширина) и height (высота) соответствующие значения и зафиксируем изменения. Теперь все в порядке, стрела выглядит нормально.

И еще попутно заменим последнее свойство — xstep. Сейчас там по умолчанию установлена 1, т.е. координата X будет расти и стрела полетит вправо. А хорошо бы наоборот! Поэтому установим новое отрицательное значение –4 (пусть летит влево, и побыстрее).

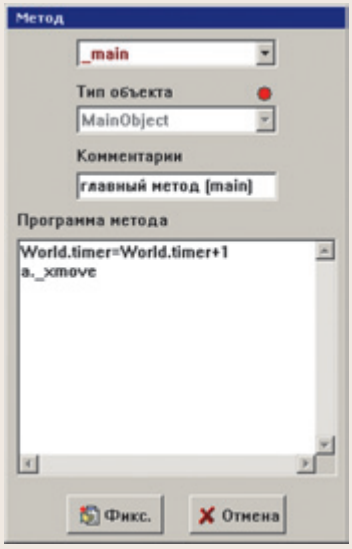


Рис. 7. Ввод метода

Сохраним все изменения в проекте. Подчеркнем, что до сих пор мы еще не написали ни одной строки программы! Но, видимо, чтобы стрела полетела, надо все же что-то написать.

Откроем таблицу с методами (оперативная кнопка с буквой “М”). Найдем в ней главный метод _main и щелкнем по его строчке в таблице. Система откроет редактор метода. В методе уже стоит строка

```
World.timer=World.timer+1
```

которая обеспечивает ход времени в виртуальном мире (свойство timer объекта World каждый раз возрастает на 1). Допишем к ней еще одну (см. *рис.* 7)

```
a._xmove
```

Все! Запускаем просмотр, и стрела летит (*рис.* 8)! Единственный недостаток — она не останавливается при достижении мишени. Чтобы это исправить, достаточно заключить уже имеющуюся строку внутрь условия, контролирующего положение стрелы (свойство left).

```
if a.left>70
a._xmove
endif
```

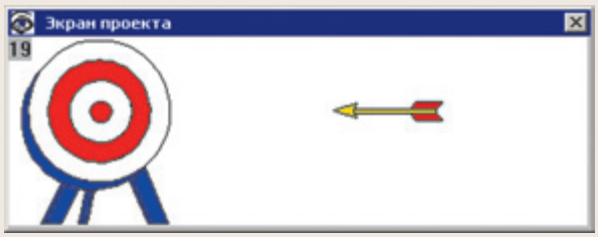


Рис. 8. Итоговая картинка: стрела летит в мишень

Теперь, достигнув центра мишени, стрела останавливается. Заметим, что время, отображаемое в левом верхнем углу, продолжает идти, однако движения на экране больше не происходит.

2.3. О языке написания методов

Прочитав парой абзацев выше о том, как останавливается стрела, попавшая в мишень, внима-

тельный читатель наверняка скажет: а обещали, что языка программирования не будет! Действительно, давайте рассмотрим вопрос о языке SB подробнее.

Начнем с того, что при написании метода без языка не обойтись, так что какой-то язык все равно нужен. Наша цель в том, чтобы он был максимально простой и, согласно принятой стратегии, без всяких описаний.

В SB есть всего несколько разновидностей действий:

- **присвоение значения** свойству (в правой части константа, другое свойство или арифметическое действие над ними); заметим, что переменных как таковых нет;
- **вызов метода** (как обычно, по имени; параметров, естественно, никаких);
- **условный оператор** в форме if — else — endif; допускаются многократные вложения if.

Каждая операция пишется в отдельной строке. Свойство или метод отделяются от имени объекта точкой, например, World.timer или a._xmove (SB **чувствителен к регистру** символов!). Если объект не указан, то SB подставляет тот, для которого вызван метод. Например, свойство без указания объекта .left в методе _myMethod при вызове a._myMethod трактуется как a.left, а при вызове b._myMethod уже как b.left.

Вот, собственно, и весь язык. Он не похож на известные мне языки программирования, разве что очень отдаленно напоминает язык СУБД семейства dBase.

Мне кажется, в первую очередь должно броситься в глаза *отсутствие цикла*[6]. По сути своей он заменяется циклическим вызовом метода _main таймером виртуального мира World.timer. Используя различные комбинации операторов if, можно получить некий аналог цикла “от” и “до” по “мировому времени” или любому другому свойству.

Подводя итоги, подчеркнем, что язык SB действительно максимально прост и интуитивно понятен (а значит, легко изучается!). Он универсален и не привязан напрямую ни к одному языку программирования.

2.4. Более сложный проект: полиморфизм

Проект, описанный в 2.2, сознательно был взят максимально простым. Рассмотрим теперь менее тривиальный проект, на базе которого можно продемонстрировать свойство полиморфизма.

Содержание проекта — путешествие (метод _travel) из одного города в другой на поезде или на воздушном шаре. Очевидно, что путешествие будет протекать по-разному и методы для поездов и воздушных шаров придется написать разные. В этом и есть суть явления полиморфизма.

Я полагаю, что читатели уже поняли идеи навигации между рабочими таблицами в SB, так что в данном разделе не будет описания последовательностей нажатия кнопок. Вместо этого сосредоточим свое внимание на последовательности операций при реализации проекта.

Прежде чем начинать работу с проектом, желательно подобрать картинки. Для фоновой картинки нам потребуются виды двух городов, а для объектов — маленькие изображения поезда и воздушного шара. Предположим, что они у нас уже есть (хотя бы из папки SB\samples\v1_02\polymor). Теперь приступим к строительству виртуального мира.

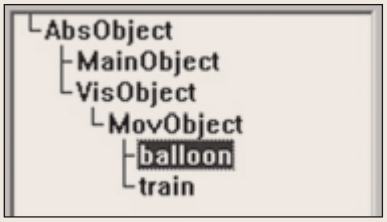


Рис. 9. Иерархия классов проекта

Этап 1 — конструируем мир. К имеющимся стандартным типам дополним свои собственные: train (поезд) и balloon (воздушный шар), образовав их от типа MovObject — итоговая иерархия типов, которую нарисовал SB, приведена на *рис.* 9. Теперь к новым типам надо добавить принадлежащие им методы. Для типа train это единственный метод _travel, а для типа balloon к такому же методу надо добавить вспомогательные методы _lift (подъем шара) и _land (его посадка). Нетрудно сообразить,

Метод для типа MainObject (виртуальный мир)	Метод для типа train (поезд)	Методы для типа balloon (воздушный шар)		
_main	_travel	_travel	_lift	_land

`World.timer=World.timer+1`
`t._travel`
`b._travel`
	`if .left<330`
	`._xmove`
	`endif`
		`if .left=150`
		`._lift`
		`endif`
		`if .top=20`
		`.xstep=1`
		`._xmove`
		`endif`
		`if .left=330`
		`._land`
		`endif`
			`if .top>20`
			`.xstep=0`
			`.ystep=-1`
			`._ymove`
			`endif`
				`if .top<80`
				`.xstep=0`
				`.ystep=1`
				`._ymove`
				`endif`

[6] Во второй версии по просьбе коллег был добавлен цикл FOR, по очереди перебирающий все объекты из определенным образом указанного списка.

Рис. 10. Каждый вид транспорта движется по-своему (пример явления полиморфизма)

что такая разница обусловлена тем, что траектория движения шара сложнее, чем у поезда. Конкретный текст методов из прилагаемого на диске проекта (с уже подобранными константами) приведен в таблице.

Методы по своему устройству достаточно просты. Путешествие для поездов (тип train) — это просто движение в горизонтальном направлении (_xmove) из начальной точки до точки с координатой 330, т.е. от изображения одного города до другого. Метод _travel для воздушных шаров, как уже говорилось выше, сложнее. Пока left=150 (значение начальной координаты), срабатывает метод _lift и шар поднимается. Его координата Y уменьшается до 20 (не забывайте, что по традициям компьютерной графики она идет сверху вниз!). Далее начинается движение вправо, причем после первого же изменения координаты X условие для процедуры подъема перестает выполняться. Движение вправо (шар летит на постоянной высоте) продолжается, пока он не достигнет второго города (X=330), после чего движение вдоль X прекращается и начинается посадка шара (_land). Шар прибудет к месту назначения, когда координата Y достигнет значения 80.

Этап 2 — создаем мир. Теперь, когда поведение поездов и воздушных шаров описано, можно создать конкретные экземпляры этих видов транспорта и установить их свойства. Для нашего примера достаточно одного поезда t и одного воздушного шара b. После того как мы установим их свойства (включая свойство image с номером необходимой картинки), мы увидим в первом городе готовые к путешествию поезд и воздушный шар.

Этап 3 — смотрим, что получилось. На *рис.* 10 приведен вид экрана, демонстрирующий путешествие из одного города в другой.

Еще раз подчеркнем, что полиморфизм в данном примере состоит в следующем. Хотя формально вызов метода _travel для поезда и шара ничем не отличается (см. текст метода _main), объекты эти движутся по-разному: у каждого из них есть собственный метод движения. И метод этот, как мы отчетливо видим, выбирается в строгом соответствии с типом объекта.

2.5. Поиграем в интерактивность

При построении виртуального мира очень хочется, чтобы была возможность влиять на поведение объектов в нем, например, с помощью клавиатуры или мыши (*интерактивность*). SB позволяет легко описать такое вмешательство пользователя, наблюдающего за развитием картины мира на экране дисплея. Для этого в главный объект World встроены методы контроля состояния клавиатуры и мыши. Их имена имеют стандартные начальные символы "_@". Так, метод _@up срабатывает при каждом нажатии клавиши ↑, а _@down — ↓; имеется еще несколько аналогичных методов, в том числе и для фиксации щелчков мыши.

Посмотрим, насколько сложно подключить к нашему проекту обработку клавиатуры. Разберем для этого проект, который обеспечивает управление нарисованным самолетиком, маневрирующим среди грозовых облаков (*рис.* 11). Как и предыдущий пример, он тоже есть на диске. В составе решения содержится 7 методов, их

Метод	Тип объекта	Назначение	Действия
_main	MainObject	Главный метод	Таймер; вызов методов, обслуживающих движение самолета
_@up	MainObject	Стрелка "вверх"	Подъем самолета
_@down	MainObject	Стрелка "вниз"	Снижение самолета
_rightbottom	VisObject	Вычисление свойств "право" и "низ"	По значениям left и width вычисляется right, а по top и height — bottom
_checkcloud	VisObject	Задевает ли самолет облако?	Сравнивает координаты одного облака с координатами самолета
_xymove	MovObject	Движение по X и Y	Приращение координат (самолета)
_fall	MovObject	Перейти к падению	Заменить картинку и шаги по X и Y

_main	_@up	_checkckoud
World.timer=World.timer+1 p._xymove p._rightbottom c1._checkcloud c2._checkcloud c3._checkcloud	if p.top>0 p.top=p.top-1 endif	if p.right>=.left if p.right<=.right if p.bottom>=.top if p.bottom<=.bottom p._fall endif endif if p.top>=.top if p.top<=.bottom p._fall endif endif endif endif
	_@down	
	if p.bottom<World.ysize p.top=p.top+1 endif	

краткая характеристика приведена в таблице, расположенной ниже. Как видно из таблицы, некоторые методы явно носят вспомогательный характер (_rightbottom, _xymove и _fall), поэтому их текст мы не будем здесь приводить (желающие, заглянув на диск, легко разберут те несколько строк, которые в них содержатся!). Сосредоточим свое внимание на остальных методах, и прежде всего на обработке клавиатуры.

Рис. 11. Проект, управляемый с клавиатуры

Обратимся к методу _@up, текст которого приведен в таблице выше. Видно, что при каждом нажатии на клавишу ↑ свойство самолета top уменьшается на единицу. Верху экрана соответствует нулевое значение, поэтому, когда объект дойдет до верхней границы, дальнейшее движение прекратится. Метод _@down устроен аналогично, с той разницей, что предельное значение координаты зависит от размера виртуального мира вдоль оси Y. Вот, собственно, и все управление. Подчеркнем только, что методы контроля клавиатуры и мыши инициируются по соответствующим аппаратным событиям, а не вызываются из текста программы.

Метод _main, текст которого приведен выше, понятен без комментариев[7]. Остается разобрать только метод _checkcloud, который по очереди проверяет каждое грозовое облако на предмет столкновения с самолетом. Метод несложен, просто в нем имеется большое количество вложенных if. Первая пара из них проверяет для горизонтальных координат рассматриваемых объектов двойное неравенство c_i.left ≤ p.right ≤ c_i.right. Смысл этой проверки состоит в том, что самолет правой границей (т.е. своей передней частью) не должен попасть внутрь *i*-го облака. При истинности неравенства возможны две разновидности виртуальной катастрофы (вызов метода _fall): в первом случае самолет может задеть облако сверху своей нижней частью (c_i.bottom ≤ p.bottom ≤ c_i.top), а во втором — снизу (c_i.bottom ≤ p.top ≤ c_i.top). Все остальные случаи считаются допустимыми, и падения самолета не происходит[8].

Кроме описанных выше, в приложении на диске есть еще много других разнообразных примеров (см. *рис.* 12). А еще, осваивая SB, не забывайте, что в вашем распоряжении имеется подробный файл помощи. И в нем, в частности, пошагово разобрана реализация еще одного проекта — “Мигающие звезды”.

2.6. Об отладочных возможностях SB

Если вы изучили описанные выше примеры очень внимательно, то скорее всего ваши проекты сразу заработали правильно. Тем не менее в реальной жизни все не так просто, и очень часто виртуальный мир “не слушается” и ведет себя как-то странно. Как найти ошибку? SB предоставит вам для этого целый ряд отладочных возможностей, которые мы здесь лишь кратко упомянем (более полное описание, как обычно, советую почитать в файле помощи).

[7] Во второй версии SB три последних строки этого метода можно заключить в цикл for.

[8] Тут есть некоторая неточность: мы проверяем касание облака самолетом только по его передней кромке, что при движении строго слева направо наиболее естественно; тем не менее опасное грозовое облако можно задеть и хвостом, когда кабина уже благополучно миновала облако, — убедитесь в теории и на практике, что в написанном варианте эта несколько экзотическая ситуация не фиксируется!

Во-первых, система имеет три режима работы:

• **авто**: метод _main вызывается системой непрерывно, и мир “живет”;

• **по шагам**: метод _main вызывается системой один раз (на каждое нажатие клавиши F8), и затем исполнение прерывается, т.е. мы фактически наблюдаем мир в “покадровом” режиме;

• **по операциям**: остановка происходит после выполнения каждой операции, что позволяет трассировать программу и проверять работу каждой строки программы.

Во-вторых, имеется специальное отладочное окно, где отображается информация после каждой выполненной операции. Кроме того, дополнительно предусмотрено еще одно окно, в котором можно посмотреть текущее значение любого свойства, что также при неправильно работающем проекте очень полезно.

Третья возможность (для углубленной работы с системой) — просмотр внутреннего представления проекта. Опытный пользователь SB может даже аккуратно подредактировать этот текст. И, наконец, “высший пилотаж” — это разобраться с тем, как система устроена, изучив результат трансляции любого из написанных вами методов. Для этой цели в системе также предусмотрено специальное служебное окно.

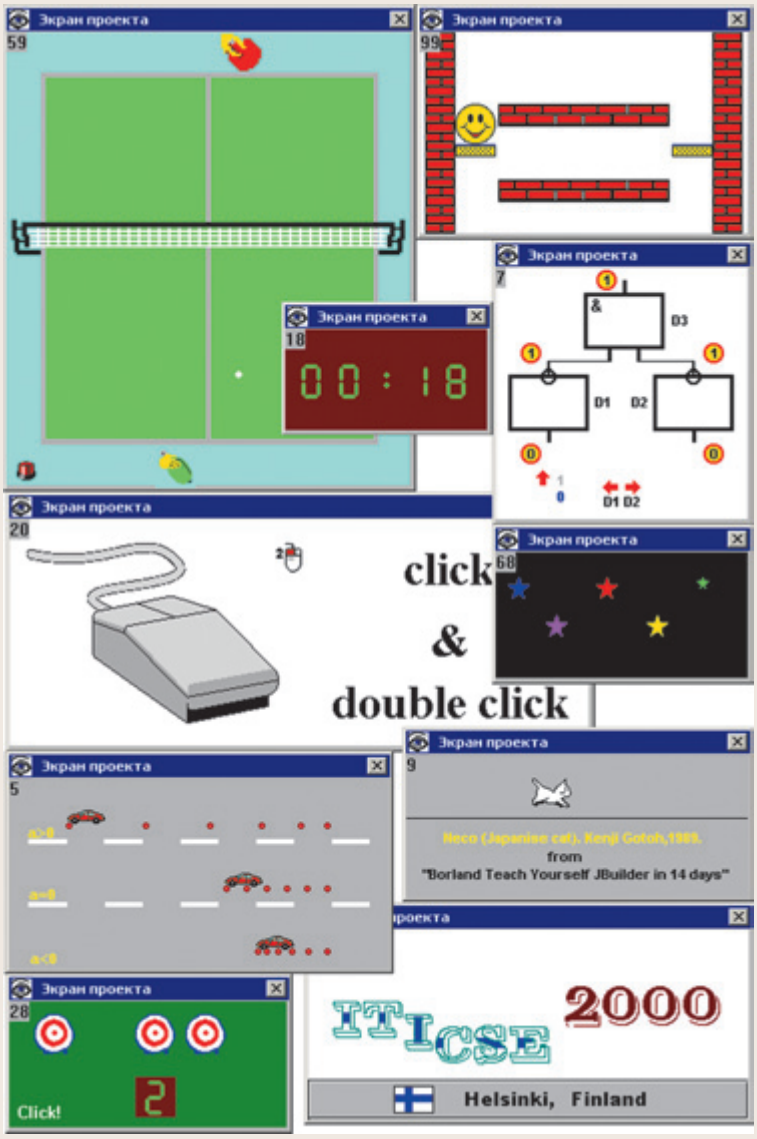


Рис. 12. Некоторые примеры различных проектов

Заключение

В статье описана оригинальная методика изучения основ ООП, этакое “ООП практически без программирования”. Примечательно, что она никак не привязана к конкретному языку программирования и даже может использоваться, если программирование до этого не изучалось.

Буду благодарен, если заинтересованные читатели пришлют свои комментарии и предложения на адрес *e_eremin@yahoo.com*.

Литература

1. *Еремин Е.А.* Об иерархическом представлении знаний с помощью некоторых программных средств. Информатика и образование, 2009, № 6, с. 37–47; № 7, с. 25–36.

2. *Поляков К.Ю., Шестаков А.П., Еремин Е.А.* Информатика № 13/2011.

3. *Еремин Е.А.* Программное обеспечение для изучения объектного подхода в курсе информатики. // IX Международная конференция “Информационные технологии в образовании”. / Сборник трудов, ч. 2. М., 1999, с. 41–43.

4. *Eremin E.* Software System to Learn Objects. Proceedings of the 5th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2000). Helsinki, 2000, p. 188.

По следам ЕГЭ-2011: новые ~~грабли~~ задачи

О.Б. Богомолова, д. п. н., учитель информатики и математики ГОУ СОШ № 1360, Восточный округ г. Москвы

Д.Ю. Усенков, ст. н. с. Института информатизации образования Российской академии образования, Москва

27 мая 2011 года открылся очередной “сезон ЕГЭ”, и одним из первых экзаменов стал ЕГЭ по информатике. В ходе этого экзамена школьники познакомились сразу с пятью новыми задачами, аналогов которых не было ни в демонстрационном варианте ЕГЭ 2011 года, ни в разработанных специалистами МИОО “ЕГЭ-подобных” контрольных, которые предлагались одиннадцатиклассникам в течение 2010/11 учебного года.

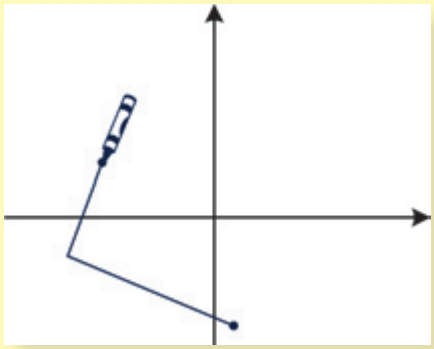
К сожалению, точные формулировки новых заданий пока недоступны (и трудно сказать — будут ли они доступны позже, — КИМ ЕГЭ теперь объявлены Рособрнадзором “документами с информацией ограниченного доступа”). Однако можно по крайней мере определить типологию этих новых задач по рассказам учащихся, сдававших ЕГЭ.

Надеемся, что предложенные ниже тренировочные задачи помогут школьникам при подготовке к ЕГЭ следующих лет.

Рисующая “Черепашка”

Имеется исполнитель типа “Черепашки ЛОГО”, который может выполнять команды: “поднять перо”, “опустить перо” и “перейти в точку с заданными координатами (x,y)”. После выполнения команды текущие координаты точки сохраняются до следующей команды.

Имеется “виртуальный лист бумаги” — координатная плоскость с началом координат в середине листа, осями X и Y и одинаковым масштабом этих осей.



Для этого исполнителя была составлена программа на алгоритмическом языке, но в этой программе оказались пропущены координаты точек в командах рисования. Какие из предложенных координат нужно подставить в пропущенные места программы, чтобы исполнитель начертил линию, похожую на окружность?

```
алг чертеж
нач
  вещ U, R, D
  цел I, N
```

```
U = 0
R = 1
N = 50
D = 2 * PI / N
поднять перо
перейти в точку (            )
опустить перо
нц для I от 1 до N
  U = U + D
  перейти в точку (              )
кц
кон
```

1) *R*, *R*
2) *R* * *R*, *U* * *U*
3) *R* * cos(*U*), *R* * sin (*U*)
4) *R* * sin(*D*), *R* * cos(*D*)

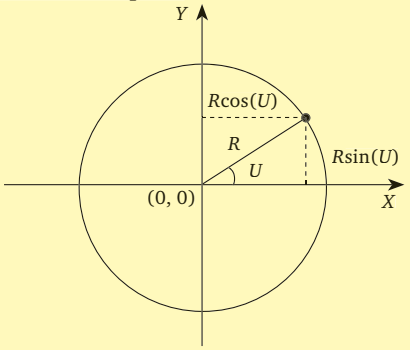
Решение

Проанализируем приведенную программу. Поскольку в ней имеется цикл: **нц для I от 1 до N**, нетрудно догадаться, что, поскольку цикловая переменная *I* нигде в теле цикла не используется, данный цикл служит лишь для отсчитывания *N* шагов отрисовки.

Далее, поскольку в цикле имеется команда `U = U + D`, становится понятно, что величина переменной *U* на каждом шаге цикла увеличивается и определяет очередную точку чертежа. А вспомнив, что ранее значение *D* задано как `D = 2 * PI / N`, а *U* — как `U = 0`, можно догадаться, что *U* — это угол (от 0 до 360 градусов), а *D* — приращение этого угла (вычисляемое делением всей окружности на заданное количество шагов цикла отрисовки).

Тогда команды `поднять перо`, `перейти в точку (  )` и `опустить перо`, стоящие до цикла, соответствуют выводу исполнителя из начала координат в начальную точку рисования, соответствующую углу 0 градусов, "вхолостую" (без черчения). А когда уже в цикле значение угла увеличивается на приращение *D* и вызывается команда `перейти в точку (  )`, то исполнитель (поскольку перо было опущено) вычерчивает отрезок от прежней точки до новой.

Остается выбрать, как должны рассчитываться координаты этих точек, лежащих на окружности. Вспоминаем геометрию:



Следовательно, координата *x* должна вычисляться как *R* * cos(*U*), а координата *y* — как *R* * sin(*U*).

В принципе вариант *R* * sin(*U*), *R* * cos(*U*) нам тоже бы подошел (тогда рисование окружности началось бы со сдвигом по величине угла на 90 градусов), но в четвертом варианте ответа аргументом синуса и косинуса является не величина угла, а величина шага его изменения, поэтому данный ответ нам не подходит.

Ответ: вариант 3.

Маскирование IP-адреса

По заданному IP-адресу и маске определить адрес сети.

IP-адрес: 17.240.120.15
Маска: 255.255.128.0

Решение

Маской IP-адреса называют число (так же, как и IP-адрес, состоящее из четырех записанных через точку байтовых значений от 0 до 255 каждое), которое определяет, какую часть IP-адреса составляет адрес сети. Для этого нужно представить и исходный IP-адрес, и маску как набор двоичных чисел, а затем поразрядно выполнить их конъюнкцию: если в каком-либо разряде маски записана "1", то значение соответствующего разряда исходного IP-адреса переписывается без изменения, а если в маске записан "0", то соответствующий разряд IP-адреса обнуляется (можно сказать, что "0" в маске "гасит" до 0 соответствующий разряд исходного IP-адреса). Полученная в результате запись и будет указывать IP-адрес сети (каждое из ее двоичных чисел можно вновь перевести в десятичную систему счисления).

Выполним эту операцию для заданных значений IP-адреса и маски:

IP-адрес: 17.240.120.15 → 00010001.11110000.01111000.00001111
Маска: 255.255.128.0 → & 11111111.11111111.10000000.00000000
00010001.11110000.00000000.00000000
↓
17.240.0.0

Ответ: 17.240.0.0

Прибавление нулей

Имеется некоторое число, записанное в семеричной системе счисления. К нему справа дописали три нуля. Во сколько раз увеличилось исходное число в результате этой операции?

Решение

Вспомним, чему соответствует дописывание нулей справа к числу в десятичной системе. Например:

Было	Стало
12345	123450
5432	543200
135	135000

Очевидно, что дописывание каждого очередного нуля к десятичному числу справа соответствует его увеличению в 10 раз.

Теперь посмотрим, что происходит при таком дописывании с числами в двоичной системе счисления (для каждого из этих чисел будем записывать в скобках десятичный эквивалент):

Было	Стало
111 (= 7)	1110 (=14)
1010 (= 10)	101000 (= 40)
1011 (= 11)	1011000 (= 88)

Здесь легко заметить[1], что дописывание справа каждого нуля соответствует увеличению исходного числа на 2.

Повторив этот эксперимент, например, с восьмеричной и шестнадцатеричной системами счисления, можно сформулировать правило: ***дописывание к числу справа каждого очередного нуля соответствует умножению этого числа на значение основания системы счисления***.

Тогда в нашей задаче легко вычислить, что при дописывании к числу, записанному в семеричной системе счисления, трех нулей справа это число увеличится в $7^3 = 343$ раза.

Ответ: в 343 раза.

Расплетаем цепочки[2]

Из цифр формируются цепочки (строки) по следующему правилу:

- первая строка состоит из одной цифры 1;
- вторая строка состоит из двух цифр — 2 и 3;
- далее любая цепочка с номером n составляется из двух цепочек — сначала переписывается цепочка с номером $(n – 2)$, а потом к ней дописывается цепочка с номером $(n – 1)$.

Вот первые несколько таких строк:

(1) 1
(2) 23
(3) 123
(4) 23123
…

Сколько нечетных и сколько четных цифр будет в строке с номером 7?

Решение

Как и при решении задач на цепочки из вариантов ЕГЭ за предыдущие годы, сначала нужно проанализировать состав получаемых цепочек и постараться вывести общую формулу. Либо, как альтернативный вариант при наличии достаточного времени, просто “честно” расписать все получающиеся цепочки (в таблице для наглядности и упрощения подсчетов четных и нечетных цифр они “подкрашены” разными цветами фона).

Проанализировав получаемые последовательности количеств четных и нечетных цифр, можно прийти к выводу, что это не что иное, как *числа Фибоначчи*! Только для количества четных цифр последовательность Фибоначчи начинается с пары чисел 0 и 1, а для нечетных — с пары чисел 1 и 1.

Почему получились именно числа Фибоначчи, понять нетрудно. Ведь, согласно правилу записи цепочек, мы в каждую n-ю цепочку копируем две предыдущих, а значит, получаемые количества четных и нечетных цифр в n-й цепочке есть сумма количеств таких цифр в предыдущих двух цепочках. А это как раз и есть принцип формирования последовательности Фибоначчи.

Теперь становится понятно, что любые подобные задачи на вычисление количеств указанных символов (чисел или букв) в цепочках, получаемых переписыванием двух предыдущих — $(n – 2)$ и $(n – 1)$, — можно легко решать, даже не распи-

№ цепочки (n)	Цепочка	Кол-во четных цифр	Кол-во нечетных цифр
1	1	0	1
2	23	1	1
3	123	1	2
4	23123	2	3
5	12323123	3	5
6	2312312323123	5	8
7	1232312323123123123123	8	13

[1] Кстати, именно так в компьютере реализуется “быстрое” умножение двоичных чисел на 2 и их целочисленное деление на 2. Для этого используются *операции сдвига*: сдвиг числа влево приводит к дописыванию нуля справа и к умножению на 2, а сдвиг вправо — к уничтожению последней справа цифры и к делению числа на 2 (без учета остатка от деления).

[2] Задачи на цепочки символов, формируемые по определенным правилам, ранее в ЕГЭ уже встречались (см., например: *Богомолова О.Б., Усенков Д.Ю.* Подготовка к ЕГЭ: расплетаем цепочки, или “Туда и обратно” // Информатика, 2011, № 4. С. 3), однако на ЕГЭ 2011 г. предлагалась задачи с иными правилами формирования цепочки. — **Прим. авт.**

сывая сами цепочки, — достаточно подсчитать количество заданных символов в первых двух "стартовых" цепочках, а потом выстроить на базе этих двух чисел последовательность Фибоначчи.

Что же касается нашей задачи, то — или расписав сами цепочки, как мы сделали в таблице выше, или воспользовавшись уже известной нам закономерностью и составив сразу последовательности чисел Фибоначчи, — мы получим следующий ответ: четных чисел — 8, нечетных — 13.

Ответ: 8, 13.

Анализ системы логических уравнений[3]

Определить, сколько различных решений имеет система уравнений:

$$\begin{cases} (x_1 \equiv x_2) \vee (x_2 \wedge x_3) \vee (\neg x_2 \wedge \neg x_3) = 1; \\ (x_1 \equiv x_3) \vee (x_3 \wedge x_4) \vee (\neg x_3 \wedge \neg x_4) = 1; \\ (x_1 \equiv x_4) \vee (x_4 \wedge x_5) \vee (\neg x_4 \wedge \neg x_5) = 1; \\ x_1 \equiv x_5 = 1. \end{cases}$$

Нужно указать именно количество решений системы уравнения, а не записывать сами эти решения.

Решение

Прежде всего заметим, что здесь используется достаточно "редкая" для школьного курса булевой логики операция "тождество" (≡). Ее результат является истинным тогда и только тогда, когда обе связанные ею логические переменные равны между собой (обе равны 0 либо обе равны 1).

Теперь обратим внимание на последнее уравнение системы. Из него следует, что переменные x_1 и x_5 должны быть равны друг другу.

Далее начнем анализировать остальные уравнения (эта задача несколько облегчается тем, что все они — "типовые").

1) Уравнение $(x_1 \equiv x_2) \vee (x_2 \wedge x_3) \vee (\neg x_2 \wedge \neg x_3) = 1$.

Результат логической операции ИЛИ (∨) равен 1, если хотя бы один операнд равен 1. Значит, достаточно, чтобы $x_1 \equiv x_2 = 1$, или $x_2 \wedge x_3 = 1$, или $\neg x_2 \wedge \neg x_3 = 1$.

Из первой части уравнения следует, что x_1 и x_2 либо обе равны 0, либо обе равны 1.

Из второй части уравнения следует, что обе переменные — x_2 и x_3 — должны быть равны 1.

Из третьей части уравнения следует, что обе переменные — x_2 и x_3 — должны быть равны 0.

Тогда уравнение (1) дает нам следующие возможные комбинации значений переменных (x_1, x_2, x_3):

- (0 0 0), или (0 0 1), или (1 1 0), или (1 1 1) — если истинна операция тождества, то результат остальных операций безразличен, т.е. x_3 может быть любой;
- (0 1 1) или (1 1 1) — если истинна вторая часть уравнения, то третья часть ложна (но это нам не важно), а результат операции тождества нам также безразличен, и значение x_1 может быть любым;
- (0 0 0) или (1 0 0) — если истинна третья часть уравнения, то вторая часть ложна (и это нам не важно), а результат операции тождества также безразличен, и значение x_1 может быть любым.

Итого получаем по уравнению (1) ***шесть*** различных вариантов значений переменных (x_1, x_2, x_3): (0 0 0), (0 0 1), (0 1 1), (1 1 0), (1 0 0) и (1 1 1).

2) Уравнение $(x_1 \equiv x_3) \vee (x_3 \wedge x_4) \vee (\neg x_3 \wedge \neg x_4) = 1$.

Результат логической операции ИЛИ (∨) равен 1, если хотя бы один операнд равен 1. Значит, достаточно, чтобы $x_1 \equiv x_3 = 1$, или $x_3 \wedge x_4 = 1$, или $\neg x_3 \wedge \neg x_4 = 1$.

Рассуждения для него аналогичны предыдущим.

Из первой части уравнения следует, что x_1 и x_3 либо обе равны 0, либо обе равны 1.

Из второй части уравнения следует, что обе переменные — x_3 и x_4 — должны быть равны 1.

Из третьей части уравнения следует, что обе переменные — x_3 и x_4 — должны быть равны 0.

Тогда уравнение (2) дает следующие возможные комбинации значений переменных (x_1, x_3, x_4):

- (0 0 0), или (0 0 1), или (1 1 0), или (1 1 1);
- (0 1 1) или (1 1 1);
- (0 0 0) или (1 0 0).

Итого по уравнению (2) получаем тоже ***шесть*** различных вариантов значений переменных (x_1, x_3, x_4): (0 0 0), (0 0 1), (0 1 1), (1 1 0), (1 0 0) и (1 1 1).

3) Уравнение $(x_1 \equiv x_4) \vee (x_4 \wedge x_5) \vee (\neg x_4 \wedge \neg x_5) = 1$.

Рассуждая аналогично, получим, что уравнение (3) дает следующие ***шесть*** возможных различных вариантов значений переменных (x_1, x_4, x_5): (0 0 0), (0 0 1), (0 1 1), (1 1 0), (1 0 0) и (1 1 1).

Обратим внимание, что поскольку все уравнения системы, кроме последнего, типовые, то тройки значений соответствующих каждому уравнению переменных всегда будут одни и те же, сколько бы ни было таких уравнений.

4) Для последнего уравнения мы уже отметили, что возможны две комбинации значений переменных (x_1, x_5): (0 0) и (1 1).

А теперь нужно "свести" результаты, полученные для каждого уравнения в отдельности, воедино. Вспомним, что объединение уравнений в систему означает, что *все* эти уравнения должны выполняться одновременно. Тогда:

- учитывая результаты для последнего уравнения, надо в предпоследнем уравнении оставить

[3] Это, пожалуй, на ЕГЭ-2011 была самая сложная задача! Ранее на ЕГЭ (и в том числе в демоварианте 2011 года) предлагалось проанализировать *одно* логическое уравнение и найти количество его решений (см., например: *Богомолова О.Б., Усенков Д.Ю.* Логические задачи на ЕГЭ: имена и логические выражения // Информатика, 2011, № 8. С. 4).

только те решения, в которых одинаковы значения x_1 и x_5 (первое и последнее в каждой триаде): (0 0 0) и (1 1 1);

• как видим, сочетание последнего и предпоследнего уравнений выполняется только тогда, когда все три переменные — x_1, x_4 и x_5 — имеют одинаковые значения;

• тогда надо в найденных наборах значений переменных для предпредпоследнего уравнения (оно у нас числится под номером 2) оставить только те варианты, в которых одинаковы значения x_1 и x_4 (первое и последнее в каждой триаде): (0 0 0), (1 1 1);

• видим, что здесь все три переменные — x_1, x_3 и x_4 — тоже должны иметь одинаковые значения;

• тогда в найденных наборах значений переменных для самого первого уравнения тоже надо оставить только варианты, в которых одинаковы значения x_1 и x_3 (первое и последнее в каждой триаде): (0 0 0), (1 1 1);

• то есть и здесь все три переменные — x_1, x_2 и x_3 — тоже должны иметь одинаковые значения.

Таким образом, начиная с последнего уравнения, которое однозначно задает только два возможных варианта комбинаций значений соответствующих переменных, мы "расплетаем" систему уравнений "снизу вверх", на каждом шаге убеждаясь, что все три задействованные в текущем анализируемом уравнении переменные должны быть равными, и тем самым однозначно определяя только два возможных варианта решений в уравнении, предыдущем текущему. Причем такой "цикл типовых рассуждений" у нас повторяется многократно, на каждом его "шаге" мы выявляем еще одну переменную, пока не дойдем до самого первого уравнения, и количество уравнений в системе на наши рассуждения практически не влияет — у нас лишь соответственно увеличится количество "проходов цикла".

Все вышесказанное означает, что, сколько бы ни было таких "типовых" уравнений в системе, мы рано или поздно придем к следующему выводу: в решениях этой системы ***все*** ее переменные должны быть тождественны. А значит, эта система имеет только два возможных решения: (0 0 0 0 0) и (1 1 1 1 1).

Ответ: 2 решения.

Обработка массива

Дан массив из 30 целых чисел от –10 до 10. Требуется найти максимум среди элементов массива, которые являются простыми числами. (Один такой элемент в массиве точно есть.)

Решение

Ранее в ЕГЭ уже были подобные задачи, когда предлагалось найти в массиве минимум, максимум или среднее значение среди элементов, которые обладают некоторым признаком (например, среди нечетных, среди четных, среди кратных какому-то числу и т.д.). Теперь же условие лишь несколько усложнилось (а мы ради большей пользы от тренировки в решении задач усложнили его еще больше, чем, по рассказам учащихся, было в ЕГЭ ☺).

Рассмотрим для начала, как ищется простой минимум в массиве:

1) в качестве предполагаемого минимума запоминаем в отдельной переменной (пусть ее имя — *min*) значение самого первого элемента массива;

2) запускаем цикл просмотра элементов массива; поскольку первый элемент уже включен в рассмотрение, его можно пропустить и строить цикл перебора значений массива со второго до последнего;

3) на каждом шаге цикла при помощи условного оператора проверяем: если текущий элемент окажется меньше предполагаемого минимума, то перезапоминаем этот текущий элемент в переменной *min*;

4) по завершении цикла в переменной *min* окажется искомый минимум.

Листинг программы на Паскале

```
program simply_min;
const n = 10;
var mas : array[1..n] of integer;
    i, min : integer;
begin
  for i := 1 to n do
    read(mas[i]);
  min := mas[1];
  for i := 2 to n do
    if mas[i] < min then min := mas[i];
  writeln('Минимум = ', min)
end.
```

Аналогично ищется в массиве простой максимум, только на каждом шаге цикла в условном операторе проверяется: если текущий элемент массива больше предполагаемого максимума (*max*), в качестве которого изначально взят первый элемент массива, то этот текущий элемент и записывается в переменную *max*.

Листинг программы на Паскале

```
program simply_max;
const n = 10;
var mas : array[1..n] of integer;
    i, max : integer;
begin
  for i := 1 to n do
    read(mas[i]);
  max := mas[1];
  for i := 2 to n do
    if mas[i] > max then max := mas[i];
  writeln('Максимум = ', max)
end.
```

А теперь пусть нам требуется найти в массиве не просто минимум, а минимальное значение среди элементов, обладающих некоторым признаком, — например, среди нечетных. Тогда нам нужно модифицировать рассмотренный выше "типовой" алгоритм поиска простого минимума следующим образом:

- во-первых, надо в условном операторе дописать (через логическую операцию **and**) условие проверки соответствия текущего элемента заданному признаку;
- во-вторых, поскольку самый первый элемент может не соответствовать заданному признаку, лучше (тем более если известен возможный диапазон значений элементов массива) в качестве предполагаемого минимума задать константу, заведомо *большую* любого элемента массива; соответственно, цикл просмотра элементов нужно вести уже с первого до последнего.

Листинг программы на Паскале

```
program min_nechet;
const n = 10;
{элементы = от -10 до 10}
var mas    : array[1..n] of integer;
    i, min : integer;
begin
  for i := 1 to n do
    read(mas[i]);
  min := 11;
  for i := 1 to n do
    if (mas[i] mod 2 <> 0) and
       (mas[i] < min)
       then min := mas[i];
  writeln('Минимум среди
           нечетных = ', min);
end.
```

Обратите внимание на условие проверки нечетности. Если его заменить на, казалось бы, эквивалентное (`mas[i] mod 2 = 1`), то программа будет работать неверно. На Паскале, Си и ряде других языков программирования у нечетных отрицательных чисел остаток от деления на 2 равен –1!!! Именно эта ошибка стала причиной потери одного балла в задании С2, что стало неприятной неожиданностью для многих школьников. При этом экзаменуемые, выбравшие верное условие, часто делали это неосознанно, а не в силу знания описанного факта, то есть им просто повезло.

Аналогично ищется и максимум среди элементов, удовлетворяющих заданному условию: надо взять алгоритм поиска простого максимума и модифицировать его точно так же, как мы меняли алгоритм поиска минимума. Но в этом случае в качестве предполагаемого максимума перед началом цикла просмотра элементов надо взять константу, заведомо *меньшую* любого элемента массива.

Листинг программы на Паскале

```
program max_nechet;
const n = 10;
{элементы = от -10 до 10 }
var mas    : array[1..n] of integer;
    i, max : integer;
begin
  for i := 1 to n do
    read(mas[i]);
  max := -11;
  for i := 1 to n do
    if (mas[i] mod 2 <> 0) and
       (mas[i] > max)
        then max := mas[i];
  writeln('Максимум среди
           нечетных = ', max);
end.
```

А теперь вернемся к нашей задаче. Поскольку в данном случае диапазон возможных значений чисел невелик, их проверку на простоту можно реализовать лишь несколькими сравнениями: "$x = 2$", "$x = 3$", "$x = 5$" и "$x = 7$" (вспомним: простые числа — это натуральные числа, не имеющие других делителей, кроме 1 и самого себя, причем 1 простым числом не считается). А в остальном это уже рассмотренный нами алгоритм поиска минимума среди элементов, удовлетворяющих заданному условию (только в операторе **if** будет записано больше логических операций).

Листинг программы на Паскале

```
program min_prost;
const n = 30;
{элементы = от -10 до 10 }
var mas    : array[1..n] of integer;
    i, min : integer;
begin
  for i := 1 to n do
    read(mas[i]);
  min := 11;
  for i := 1 to n do
    if ((mas[i] = 2) or (mas[i] = 3) or
       (mas[i] = 5) or (mas[i] = 7)) and
       (mas[i] < min)
       then min := mas[i];
  writeln('Минимум среди
           простых = ', min);
end.
```

Программирование для дошкольников и младших школьников

А.Г. Кушниренко,
А.Г. Леонов

Сегодняшние дети уже с трех-четырехлетнего возраста имеют опыт “пультового” управления бытовыми приборами — телевизорами, проигрывателями, электронными игрушками. Отсюда совсем недалеко и до понятия “программа”. Если пяти-шестилетнему ребенку дать новую игрушку с пультом управления, объяснить, как устроена игрушка, какие кнопки-команды есть на пульте и какую задачу управления игрушкой нужно решить, то ребенок окажется способным не только решить задачу методом проб и ошибок, но и будет способен объяснить **еще до начала решения**, какие кнопки на пульте управления и в какой последовательности нужно нажать, чтобы достигнуть намеченной цели. Это значит, что “в голове” у ребенка **есть** программа — *план будущей деятельности*. А вот средств, инструментов выразить этот план в какой-то материальной форме у ребенка, еще не умеющего читать и писать, **нет**. Такой инструмент мы и попытались создать и назвали его “ПиктоМир”[1].

Как известно, дети очень любят играть. Поэтому учить программированию мы будем, вовлекая ребенка в игру с понятным и легко изучаемым интерфейсом.

А так как дети очень любят играть с телефонами родителей, особенно с такими интересными, как iPhone и ему подобные, то, помимо традиционной реализации для десктопов и нетбуков, мы реализовали систему ПиктоМир и для платформы iPhone/iPad (компании Apple) в виде игры с незамысловатым сюжетом.

В далекой-далекой галактике грядет экологическая катастрофа на некой обитаемой планете. И планета просит помощи у всех разумных существ во Вселенной. Земля готова прийти на помощь.

Ракета готова, и дети космонавтов — мальчик Антон и девочка Аня — упрашивают родителей перед отлетом показать им ракету.

После экскурсии дети решают спрятаться на ракете и полететь в космос вместе с родителями. Они тайком пробираются на ракету и начинают изображать из себя космонавтов: садятся за пульт управления и громко отдают команды, которые они слышали на экскурсии. Но ракета полностью готова к запуску, на ней включено голосовое управление и, к ужасу детей, включаются двигатели и ракета стартует.

Земля связывается с детьми. Они целы и невредимы. Собирается галактическая ко-

[1] ПиктоМир — свободно распространяемая многоплатформенная программная система. С сайта *www.piktomir.ru* сегодня можно загрузить собственно систему, работающую под MS Windows, и 6 цепочек заданий, называемых **Играми**. Первая игра вводит команды Робота, понятие подпрограммы и конструкцию “цикл *N* раз”. В играх 2, 3 и 4 используются только эти понятия. В более трудных играх 5 и 6 используются конструкции ветвления и повторения.

миссия ООН. Что делать, как спасти детей, как спасти планету? Далекая планета нуждается в срочной помощи. Все роботы и материалы, которые нужны для ее спасения, погружены на ракету, которая успешно стартовала с Земли. Только вместо подготовленных к управлению роботами космонавтов на ракете оказались дети. На заседании галактической комиссии ООН идет дискуссия. Ученые и родители-космонавты предлагают решение: не сажать ракету с детьми на Землю, а подготовить детей к выполнению спасательных работ, научить их обращаться с многочисленными роботами-помощниками и отправить к планете без взрослых. Комиссия соглашается с этим предложением, и начинается обучение.

Ракета приближается к терпящей бедствие планете, ученые волнуются, удалось ли детям так быстро выучиться управлять сложными роботами?

Ведь именно роботы должны будут подготовить площадку для посадки корабля, после посадки очистить почву вокруг корабля, загрязненную выхлопами из ракеты; посадить и вырастить растения, отфильтровать воду для полива и т.п.

Итак, игра — выполнение спасательной миссии — начинается.

В качестве первого задания ребенок должен научиться управлять роботом-чистильщиком. Нужно очистить часть космодрома от ядовитых отходов (которые имеют зеленый цвет). Управление роботом предельно просто и ведется в ручном режиме. Робот может делать шаг вперед и поворачиваться направо и налево (кнопки управления находятся в нижней левой части iPhone для левой руки). Кроме того, он может залить (закрасить) специальный обеззараживающий состав в клетку, на которой стоит (для этого действия пиктограмма расположена под правую руку внизу экрана). Наш Робот снабжен зачатками интеллекта: если клетка уже очищена, то по команде “закрасить” он не будет ничего делать, экономя дефицитный спасительный раствор.

В первом задании на ручное управление Роботом нет ограничений по времени и по количеству шагов. Однако если задание выполнено за минимальное количество шагов, то ребенок получит золотую звездочку, а иначе он будет награжден только серебряной. Всего на первом этапе нужно выполнить 14 заданий.

Ученик успешно справляется с первым заданием с помощью кнопочного (ручного) управления роботом.

Игра продолжается. Антон и Аня уже выполнили успешно несколько заданий, но тут возникают трудности с ручным управление. В космосе вокруг планеты начинается магнитная буря, и оказывается, что командами по радио управлять роботами на планете нельзя из-за помех радиосвязи из космоса. Поэтому перед детьми возникает новая задача. Для выполнения оставшихся заданий на планету нужно посылать роботов-исполнителей, способных действовать автономно, по заранее составленной программе. Поэтому дети должны научиться составлять программы управления роботами. Программа составляется, “собирается” из пиктограмм

на экране компьютера. Сначала Антон и Аня учатся составлять простейшие, линейные программы.

Постепенно задания усложняются, и для их выполнения приходится освоить повторители. Дело в том, что для каждого задания задан шаблон программы — ограниченный набор пустых гнезд для пиктограмм команд, и для описания длинной последовательности команд не обойтись без повторителей.

В приведенной выше задаче последовательность действий Робота такова: сначала он заливает клетку, затем делает шаг и после этого поворачивается налево. Если этот набор действий повторить 4 раза, то задание будет выполнено. Поскольку в шаблоне программы всего 4 пустых гнезда, а нужно выдать Роботу 12 команд, ребенок просто поставлен в такие условия, когда нужно использовать повторитель[2].

Следующая задача также решается при помощи повторителя. Только здесь нужно учитывать, что часть почвы уже обработана (пригодна для посадки растений, что показано синим цветом).

Здесь решение начинается с последовательности команд:

Закрасить
Вперед
Вперед
Вперед
Налево

Повторяем эту последовательность команд 4 раза, и задача решена. Как и в предыдущем примере, в предоставленном ребенку шаблоне программы не хватает места для явного размещения всех требуемых команд. Нужно использовать повторитель 4.

В следующей задаче шаблон программы предписывает использовать вспомогательные алгоритмы. Мотивация к использованию вспомогательных алгоритмов такая же, как для повторителей, — компактификация программы.

В шаблоне программы предусмотрено использование двух методов компактификации: вспомогательный алгоритм — команда *A* и повторители. Полное решение задачи изображено ниже:

В ПиктоМире поддержаны понятия ветвления и повторения. В начале шаблона алгоритма может быть отведен “ромбовидный” шаблон для пиктограммы команды-вопроса. В случае, если ответ на вопрос утвердительный (условие соблюдается), тело алгоритма выполняется. Если нет, то все команды алгоритма пропускаются без исполнения. Одновременно с ромбовидным шаблоном ветвления перед алгоритмом может быть задан “круглый” шаблон для повторителя. Если он задан,

[2] ПиктоМир — система ограниченного назначения и применения; она ориентирована на бестекстовую работу. Любые понятия, которые естественно выражать в текстовом виде, в ПиктоМире отсутствуют. Введение Исполнителя “счетчик” в сочетании с конструкцией “цикл *К* раз” позволяет “пиктографическими” средствами проиллюстрировать использование в программе одной числовой величины, но на этом ПиктоМир и останавливается. В ПиктоМире не поддержаны такие понятия, как вспомогательный алгоритм с параметрами, тип величины, описание величин разных типов, массивы, арифметические и логические выражения, ввод/вывод и т.п. Все эти понятия естественнее изучать в процедурных языках программирования с традиционной текстовой нотацией, например, в КуМире. Для облегчения перехода от “пиктографической” нотации к традиционной текстовой мы встроили в ПиктоМир команду перевода (сохранения в файле) ПиктоМир-программы в КуМир-нотацию.

то условие проверяется каждый раз перед очередным шагом цикла. Таким образом, если задать повторитель бесконечным, то будет смоделирован цикл “пока”. Такая конструкция используется для решения более сложных задач, например, такой задачи:

Требуется выйти из спирального лабиринта, очищая отмеченные угловые клетки.

Хотя задача и непростая, решение очень компактное и задание шаблона программы оказывается хорошей подсказкой.

Проявив настойчивость, ребенок методом проб и ошибок доходит до решения:

Запустив выполнение программы, видим, что задача успешно решена.

Для сравнения приведем решение той же задачи в текстовом виде:

```
использовать Робот-Вертун
алг
нач
. нц пока Впереди свободно
. . Вперед
. . А
. . Закрасить
. кц
кон

алг А
нач
. если  Впереди стена то Налево все
кон
```

Это текстовое решение для дошкольников и младших школьников оказывается гораздо более сложным, чем пиктографическое.

ПиктоМир создавался как инструмент для изучения азов программирования дошкольниками и младшеклассниками, еще не умеющими или не любящими читать и писать. Накопленный опыт работы с системой ПиктоМир показал, что поставленную задачу она решает.

С помощью ПиктоМира можно научить старших дошкольников и младших школьников базовым понятиям последовательного программирования: *исполнитель* и его *система команд*, *программа и подпрограмма*, чередование этапов *составления и выполнения* программы при ее отладке, использование конструкций *ветвления и повторения* при составлении программы.

Как может быть использована в образовании открывшаяся возможность обучения азам программирования в нетекстовой форме?

Мы предлагаем три варианта раннего введения программирования с помощью системы ПиктоМир:

1. Индивидуальные занятия квалифицированного взрослого с ребенком.

2. Мини-курс в подготовительной группе детского сада (6–8 занятий в группе до 6 человек, за компьютером дети проводят около 15 минут на каждом занятии).

3. Предварительный курс программирования в начальных классах: бестекстовое программирование в ПиктоМире и последующий переход к решению уже разобранных задач в среде текстового программирования КуМир. (Для пятишестиклассников, начинающих изучать программирование по школьным учебникам[3], ПиктоМир может использоваться в 3–4-часовом вводном курсе, особенно, если последующее обучение будет проводиться в среде КуМир.)

Отталкиваясь от проведенного А.П. Ершовым сравнения между программированием и грамотностью, можно рассматривать ПиктоМир как аргумент в поддержку лозунга: “Программирование — одна из базовых грамотностей XXI века”. И авторы данной статьи убеждены, что детям будет полезно овладеть этой грамотностью в достаточно раннем возрасте.

[3] *Ландо С.К., Семенов А.Л., Вялый М.Н.* Информатика, 7-й класс. М.: Просвещение, 2008.

Excel. Поиск решения

“От малых причин бывают весьма важные последствия...”
Козьма Прутков
“Мысли и афоризмы”, № 79а

А.Н. Комаровский,
Россошанская школа-интернат
Воронежской области

Смышленые шестиклашки...

В газете “Информатика” № 2/2011 мое внимание привлекла статья А.И. Сенокосова “Ёксель!”, в которой предлагается система лабораторных работ прикладного характера, рекомендуемая для учащихся 6-го класса, изучающих электронные таблицы. Задачи, рассматриваемые в лабораторных работах, охватывают довольно широкий спектр возможностей MS Excel и привлекательны не только своей практической направленностью, но и неординарной постановкой. Часть подобных задач я и раньше использовал в своей практике, часть (типа лабораторной работы № 5 “Ведомость успеваемости”), если не предлагал детям, то частенько приходилось решать самому, но две сходные лабораторные работы: № 3 — “Экскурсионная поездка” и № 4 — “Перевозка грузов” — меня поставили в тупик.

Вот краткая формулировка первой[1]:

[1] Авторскую формулировку смотрите в вышеуказанном источнике: газета “Информатика” № 2/2011.

Таблица 1

	A	B	C	D	E	F
1	**Класс**	**Едут на экскурсию**	**Автобус № 1**	**Автобус № 2**	**Автобус № 3**	**Автобус № 4**
2	6а	23	1			
3	6б	17		1		
4	6в	14				
5	6г	7	1			
6	7а	18				1
7	7б	5				
8	7в	19				
9	7г	22				
10	8а	21				1
11	8б	11				
12	8в	15				
13	**Всего:**	**172**	**30**	**17**	**0**	**39**

Лабораторная работа № 3. Экскурсионная поездка

Школьники едут на экскурсию на четырех автобусах, каждый из которых вмещает не более 45 пассажиров. Требуется рассадить детей по автобусам так, чтобы учащимся каждого класса не пришлось ехать в разных автобусах.

Завершался текст лабораторной работы примерно такими словами, которые я (да простит меня автор) несколько изменил, не изменяя сути, дабы смысл их был более понятен читателю: “Естественно, что пересчет количества людей в автобусе производится автоматически, то есть в эти ячейки введены формулы. А что это за формулы, вы, наверное, уже догадались”.

Ни в самой статье, ни в материалах к ней на диске, прилагаемом к номеру газеты, методика решения этого и следующего упражнения не приводилась.

По тексту лабораторной работы следует, что компьютерно грамотные шестиклассники, пробегавшие мимо спорящих о проблеме рассадки завуча и классного руководителя, лихо справились с поставленной задачей с помощью электронных таблиц.

…и заторможенный учитель

Конечно, и у меня не возникло проблем с тем, что за формула должна быть в ячейке B13 для подсчета общего количества учащихся, отправляющихся на экскурсию. Полагаю, что **продвинутые** шестиклассники скорее всего вместо довольно длинной формулы =$B$2*C2+$B$3*C3+…+$B$12*C12 для ячейки C13 и последующего ее копирования вправо использовали все же =СУММПРОИЗВ($B$2:$B$12;C2:C12), а может быть, даже еще более короткий вариант — для работы с массивами {=СУММ($B$2:$B$12*C2:C12)}, фигурные скобки в котором появляются при фиксации аккордом Ctrl + Shift + Enter, как это и положено для таких формул.

До сих пор я наивно полагал, что неплохо знаю MS Excel, но что за формулы вводили смышленые шестиклассники в диапазон C2:F12 — никак не мог взять в толк.

Понятно, что если проблема рассадки будет решена для первых трех автобусов, то с четвертым забот не будет. Но чтобы рассадить в три автобуса, надо сначала рассадить детей в два автобуса. Отсюда вытекает, что прежде всего следует укомплектовать первый автобус.

Так как в ячейках проблемного диапазона должны появляться нули, если класс не едет на этом автобусе, и единицы, если он оказался в числе избранных, то напрашивалась такая формула: =ЕСЛИ(C13=45;1;0) или, учитывая, что в случае полностью укомплектованных трех автобусов в четвертом будут ехать только $172 - 45 \cdot 3 = 37$, то, например, следующая: =ЕСЛИ(И(C13>=37;C13<=45);1;0). Но дело в том, что в ячейки строки “Всего:” вписаны формулы, диапазон расчетов которых пересекается с нашим проблемным диапазоном, вследствие чего образуются циклические ссылки (замкнутый круг). Об этом Excel грозно предупреждает при первой же попытке подкинуть ему такую “несъедобную” задачку.

Поиск функции

Из описания лабораторной работы складывалось впечатление, что есть некая, незнакомая мне, функция, с помощью которой удастся благополучно рассадить детишек по автобусам, а сама ситуация с экскурсионной поездкой была похожа на задачу о распределении ресурсов из дискретных задач оптимизации, которая решается с помощью функции Беллмана.

Я “прочесал” все функции MS Excel, имеющие хоть какое-то отношение к теме, но ни одна из них не подсказывала идеи решения. Есть функции для вычисления распределений Вейбулла, Пуассона и Стьюдента, преобразования Фишера, коэффициента корреляции Пирсона… Функции Беллмана в MS Excel не удалось найти даже в самом полном перечне.

В связи с тем, что мне в скором времени предстояла встреча с автором лабораторных работ, я, после безуспешных попыток осилить задачку для шестиклассников, решил отложить эту заботу до нашего рандеву, чтобы обратиться к первоисточнику.

Неожиданное решение

В разговоре с автором лабораторных работ, к моему изумлению, выяснилось, что в этом случае шестиклассники действуют банально просто — методом подбора. Формулы в строке “Всего:” только помогают оценивать ту или иную версию.

Попробовал и я поставить себя на место школьников и найти перебором какой-либо подходящий вариант. Мне это удалось менее чем за пять минут (видно, давно не занимался устным счетом). С точки зрения методической целесообразности такой способ решения вполне допустим, а на уровне учащихся шестого класса, в плане приобретения опыта, даже желателен. Ведь нельзя приступать к изучению умножения, не ощутив всех “прелестей” сложения одинаковых слагаемых. Трудно обращаться к понятию объема, если не усвоены понятия длины и площади. Словом, “помучишься — научишься”. Однако не могу сказать, что был удовлетворен таким вариантом, так как он, в моем понимании, мало способствовал укреплению веры в мощь электронных таблиц.

Акселератор подбора

В свое время заняться изучением языка Visual Basic for Applications меня вынудила проблема циклических ссылок, которую в одной из задач удалось

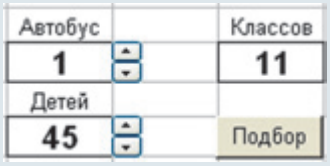


Рис. 1. Интерфейс процедуры подбора

преодолеть с помощью макроса. Естественно, что в данной ситуации мне захотелось помочь детям облегчить и ускорить процедуру подбора средствами VBA. С помощью панели **Формы** были созданы два счетчика. Первый — для номера автобуса, настроен на диапазон изменения от 1 до 4 и связан с ячейкой H5, а второй — для количества детей в автобусе, в диапазоне от 37 до 45 связан с ячейкой H7. В ячейке J5 с помощью формулы

=11-СЧЁТЕСЛИ(СМЕЩ(B2;0;0;11;H5);1)

автоматически определяется число неразмещенных классов до момента начала подбора. Процедура подбора, представленная внизу страницы, запускается соответствующей кнопкой, созданной тем же способом, что и счетчики.

Можно было бы использовать последовательные варианты перебора, но в этом случае или каждый раз работа подпрограммы завершалась бы на одном и том же, первом попавшемся, варианте, или надлежало усложнять процедуру, добавив в нее возможность продолжения исследования нерассмотренных версий. В конце концов выбор был сделан в пользу более реальной для ручного подбора ситуации с использованием генератора случайных чисел. Понятно, что при такой организации даже за полный цикл могут быть испытаны не все возможные варианты, но нам никто не мешает простым нажатием кнопки запустить новую серию испытаний.

Условие досрочного выхода из цикла можно задать в виде интервала, в котором будет варьироваться нижняя граница:

```
If Cells(13, c) >= m And Cells(13, c)
                    <= 45 Then Exit For
```

После того как решение найдено, весь диапазон рассматриваемых данных, включая именование классов и количество учащихся в них, сортируется по возрастанию данных обрабатываемого столбца с тем, чтобы при решении вопроса о наполнении следующего автобуса не отвлекаться на уже найденные решения.

Первые результаты

При поиске следует придерживаться правила последовательного заполнения автобусов от первого к третьему. Если для какого-то автобуса (обычно третьего) после нескольких попыток не удается найти подходящий вариант, необходимо вернуться к предыдущему автобусу или даже к первому, чтобы попробовать другой вариант рассадки. При этом можно исследовать различные условия размещения. Следует

	A	B	C	D	E	F
1	Класс	Едут на экскурсию	Автобус № 1	Автобус № 2	Автобус № 3	Автобус № 4
2	8а	21	1			
3	7г	22	1			
4	6в	14	0	1		
5	7а	18	0	1		
6	8б	11	0	1		
7	6а	23	0	0	1	
8	7б	5	0	0	1	
9	8в	15	0	0	1	
10	6г	7	0	0	0	1
11	6б	17	0	0	0	1
12	7в	19			0	1
13	Всего:	172	43	43	43	43

Рис. 2. Вариант равномерной рассадки

```
Sub Podbor()
m = Range("H7")          'Число детей в автобусе
c = Range("H5") + 2      'Номер обрабатываемого столбца
r = 13 - Range("J5")     'Начальный номер строки диапазона
n = 2 ^ (14 - r)         'Число вариантов перебора
Randomize                'Инициируем генератор случайных чисел
For i = 1 To n           'Цикл перебора
    a = Int(Rnd(1) * n) 'Случайный вариант перебора
    k = 0                'Счетчик смещения по строкам
        While a > 1      'Пока не заполнен вариант перебора
            Cells(k + r, c) = a Mod 2
            'Вносим в ячейку очередную двоичную цифру
            a = a \ 2    'Готовим переход к следующей цифре
            k = k + 1    'Готовим переход к следующей строке
        Wend             'Конец цикла "пока"
    If Cells(13, c) = m Then Exit For
    'Если число детей удовлетворяет условию, То досрочный выход из цикла
    Range(Cells(2, c), Cells(12, c)).ClearContents 'Очищаем диапазон
Next i                   'Конец цикла "для"
Range(Cells(r, 1), Cells(12, c)).Sort _
    Key1 := Cells(r, c), _
    Orientation := xlTopToBottom
'Сортируем диапазон по убыванию данных обрабатываемого столбца
End Sub
```

отметить, что вариант равномерного распределения по 43 ученика в каждом автобусе реален. При этом можно будет, как это и положено, в каждый автобус поместить еще по двое взрослых сопровождающих.

Как показали дальнейшие исследования, при данных начальных условиях возможно осуществление любого из тринадцати различных способов заполнения автобусов без деления классов. Для реализации некоторых вариантов методом подбора, даже с использованием ускорителя, может понадобиться немало времени.

Таблица 2. Возможные варианты рассадки

№	1	2	3	4
1	45	45	45	37
2	45	45	44	38
3	45	45	43	39
4	45	45	42	40
5	45	45	41	41
6	45	44	44	39
7	45	44	43	40
8	45	44	42	41
9	44	44	44	40
10	44	44	43	41
11	44	44	42	42
12	44	43	43	42
13	43	43	43	43

Для того чтобы автоматически получать данные о заполнении последнего автобуса, можно в ячейку F2 ввести формулу =ЕСЛИ(ИЛИ(E2=1;E2="");"";1) и скопировать ее в диапазон F2: F12, а в ячейке F13 вычислять оставшихся как =B13-СУММ(C13:E13).

Нерешенные вопросы

При использовании макроса процесс подбора осуществляется значительно быстрее, большую часть забот берет на себя Excel, появляется возможность исследования различных вариантов. Однако остается ряд вопросов:

— Во-первых, вопрос оптимального выбора, то есть, является ли найденный вариант наилучшим, особенно если условие задано в виде интервала?

— Во-вторых, вопрос оптимального перебора: существует ли стратегия, в которой может быть рассмотрен минимум возможных вариантов?

— В-третьих, вопрос разрешимости задачи, ведь если нам за несколько попыток не удалось найти подходящий вариант, значит ли это, что решения не существует?

— В-четвертых, несмотря на схожесть заданий обеих лабораторных работ, ранее рассмотренные методика и процедура на VBA мало подходят и требуют существенной переработки для решения задачи о перевозках. Здесь самое время изложить ее суть.

Лабораторная работа № 4. Перевозка грузов

Тремя грузовиками грузоподъемностью 12 тонн необходимо перевезти материалы и оборудование, распределив груз примерно поровну. Допустимая перегрузка — не более 150 кг.

Таблица 3

Наименование оборудования	Кол-во	Масса единицы груза (кг)
Станки (штуки)	11	850
Трубы (упаковка)	4	1930
Буровое оборудование (ящики)	2	1700
Отделочный камень (ящики)	4	1250
Промышленные электромоторы (штуки)	7	730
Кабель (бухты)	5	1100

Осознание

В процессе работы над первой задачей пришло понимание, что:

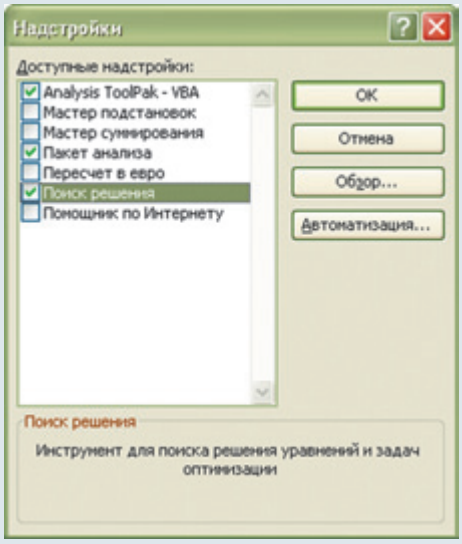
— Во-первых, вряд ли стоит искать какие-то формулы для подстановки в диапазон подбора значений, так как в обоих случаях, по сути, речь идет об обратной задаче нахождения исходных данных по известному результату — значению функции многих аргументов, то есть фактически это задача поиска корней уравнения или системы уравнений. Так что рекомендацию автора к рассматриваемой лабораторной работе: “Любая задача, связанная с обработкой информации, требует в первую очередь осмысления связи между исходными данными и результатами”, — в данной ситуации неплохо бы развернуть и в обратную сторону — осмысления связи между результатами и исходными данными. Функциональная зависимость — это прежде всего отношение “один к одному” или “многие к одному”, но обратное отношение зачастую функцией не является. Далеко не всегда такого рода связь может быть выражена аналитически, только вот поиск методом “научного тыка” в школьном курсе математики как-то не приветствуется. Там господствуют методы тождественных преобразований алгебраических выражений.

— Во-вторых, что предлагаемые задачи фактически являются задачами линейного программирования, которые сводятся к решению системы линейных уравнений и неравенств с несколькими переменными, а значит, должны решаться на основе симплекс-метода. Естественно, что объяснять суть симплекс-метода шестиклассникам рановато, но это еще не значит, что его нельзя применять или по крайней мере продемонстрировать. Ведь большинство из нас пользуется различными устройствами на уровне “черного ящика”, не заморочиваясь вопросом: “как это работает?”.

“Что если?”

Среди нефункциональных сценариев возможного решения проблемы мое внимание привлекли

инструменты анализа “что если?”, но ни диспетчер сценариев, ни подбор параметра, ни таблица подстановки данных для реализации этих лабораторных работ не годились. Как-то не верилось, что Excel “спасовал” перед задачами такого рода. В конце концов пришлось добраться до надстроек. В MS Excel-2003 это делается через меню: **Сервис — Надстройки…**, а в Excel-2007 и 2010 — сложнее, через кнопки: **Office** (вкладка **Файл**) — **Параметры** — **Надстройки** — (внизу) **Перейти…**



Исследование только что подключенных надстроек **Analysis ToolPak — VBA** и **Пакет анализа** не принесло утешения, но название надстройки **Поиск решения** было созвучно моим предыдущим мытарствам, да и комментарий: “Инструмент для поиска решения уравнений и задач оптимизации”, — говорил, что это, кажется, то, что я искал.

Диалог поиска

Для вызова диалогового окна **Поиск решения**, после подключения этой надстройки, в MS Excel-2003 появится в ниспадающем меню: **Сервис** — строка **Поиск решения…**, а в Excel-2007 и 2010 на вкладке **Данные** — новая группа **Анализ** с соответствующей кнопкой. Интерфейс окна **Поиск решения** в Excel-2003 и 2007 совпадает, а в Excel-2010 имеет несущественные отличия, поэтому дальнейшие действия с ним излагаются в основном в версии MS Excel-2003, по причине меньшей площади скриншотов.

Достоинством технологии поиска решений в отличие от не оправдавших надежд инструментов **Анализ “что если?”** является возможность задания не только определенного значения в исследуемой (целевой) ячейке, но и экстремальных значений. Но, главное, допускается возможность поиска аргументов не в отдельной ячейке, как в случае инструмента **Подбор параметра…**, а в целом диапазоне смежных ячеек. К тому же, как на значения аргументов, так и на значения используемых функций можно накладывать ряд ограничений.

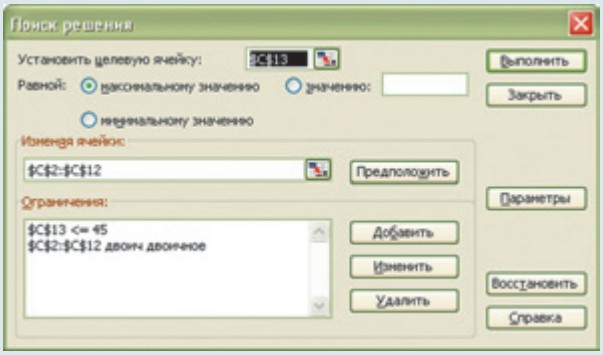


Рис. 4. Укомплектовываем первый автобус

Исходя из логики предыдущего комплектования автобусов, попробуем решить проблему неделимости классов, попытавшись максимально загрузить сначала первый автобус, затем второй, а в последний — разместить оставшихся. В качестве целевой ячейки укажем C13 (см. *рис.* 4), отметив переключателем **максимальное значение**. Нажатие кнопки **Предположить** в данном случае дает эффект, отличный от предполагаемого, поэтому, поместив курсор в поле **Изменяя ячейки**, выделим с помощью мыши нужный диапазон $C$2:$C$12.

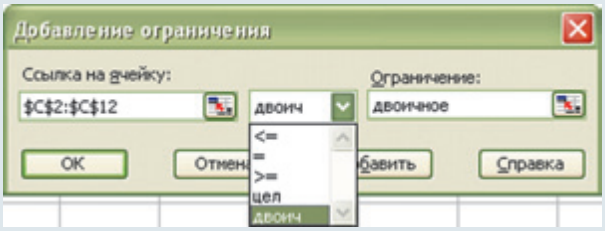


Рис. 5. Устанавливаем ограничения на аргументы

Чтобы указать ограничения, нажмем кнопку **Добавить** и зададим общее количество детей в первом автобусе $C$13<=45, и здесь же с помощью кнопки и раскрывающегося списка добавляем еще одно условие на аргументы, которые лучше определить как двоичные. После нажатия кнопки **Выполнить** почти мгновенно получаем вариант загрузки первого автобуса.

Хотя, как утверждается в справке к инструменту **Поиск решения**, в поле **Изменяя ячейки** можно вводить имена или ссылки на изменяемые ячейки, *разделяя их запятыми*, на практике выясняется, что Excel к такому способу записи относится весьма предвзято, выдавая сообщение “Ошибка в формуле”. Так что придется ограничиться вариантом смежных изменяемых ячеек. В этом случае для решения вопроса о том, какие классы едут во втором автобусе, придется отсортировать строки нашей таблицы по убыванию данных диапазона оптимизации, чтобы в нашем списке неразмещенные классы не чередовались с теми, чья судьба уже определена.

На следующем этапе вызываем опять диалоговое окно **Поиск решения**, в котором сохраняются последние настройки, и корректируем установки в соответствии с задачей комплектования следующего автобуса. В качестве целевой ячейки теперь указываем D13, а диапазон подбора аргументов должен охватывать

только те строки столбца D, которые соответствуют нерассаженным классам. Ограничения редактируем через кнопку **Изменить** и вновь, после укомплектования, сортируем диапазон. Аналогичные действия производим в столбце E. За время, сравнимое со временем ручного подбора, получаем вариант рассадки, удовлетворяющий условиям.

Единовременная оптимизация

А нельзя ли сократить время решения задачи? Ведь фактически мы общую задачу комплектования разбили на несколько последовательных подзадач оптимизации и решали проблему пошагово. Но интерфейс технологии поиска решений позволяет нам вводить достаточно большое число ограничений в один прием, да и весь наш объединенный диапазон подбора аргументов охватывает только смежные ячейки. Остается только определиться с целевой функцией и ограничениями для этого случая единовременной оптимизации.

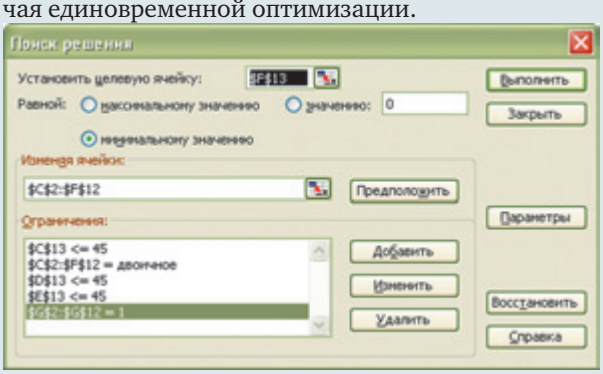


Рис. 6. Условия единовременной оптимизации

В качестве цели можно выбрать минимум детей в одном из автобусов, например, в последнем. Ограничения на количество детей в остальных автобусах можно оставить прежними. А требования двоичного значения аргументов оказываются недостаточными, так как в этом случае одному и тому же классу может быть предложено место в разных автобусах, тогда как другой класс может оказаться вовсе без предложений. Чтобы каждый из классов уехал на экскурсию в соответствии с пожеланиями, придется справа от таблицы вписать в ячейку G2 формулу =СУММ(C2:F2) для контроля их размещения в автобусы, скопировать ее в диапазон G2:G12 и потребовать в ограничивающих условиях, чтобы каждое значение в этом диапазоне равнялось единице.

И еще небольшое замечание из практики. В диапазонах изменяемых и контролирующих ячеек лучше всего задать числовой формат без значащих цифр после запятой, так как, даже несмотря на ограничения целости или двоичности, Excel при решении такого рода задач вместо нуля может неожиданно выдать, например, 2,90578575689931E-11, а вместо единицы — 1,00000000000309 или 0,99999999999846. Не так просто будет объяснить, если это шестиклассники, такой результат.

Неудачный старт

Итак, все, кажется, готово. Нажимаем кнопку **Выполнить**… Проходит минута, другая… шес-ти-де-ся-та-я. Excel, насколько можно судить по мелькающим данным в строке состояния, самозабвенно обсчитывает варианты, захватив максимум процессорного времени и затормозив работу операционной системы и других приложений. Да-а, такой поиск нам не нужен, уж лучше вручную. Приходится прервать его универсальной комбинацией Ctrl + Alt + Delete. Впоследствии, через справку, выясняется, что для такого случая в Excel специально предусмотрена клавиша Esc, которая позволяет обходиться меньшими потерями.

Попробуем разобраться с причинами неоправданной увлеченности Excel процессом перебора, для чего в окне **Поиск решения** нажимаем кнопку **Параметры**. Описываемый эксперимент выполнялся в Excel-2010. Оказывается, в отличие от Excel-2003 и 2007 количество ограничений на пределы решений в этой версии увеличилось. Но если в предыдущих версиях было ограничено начальными значениями максимальное время поиска — 100 с и предельное число итераций — 100, причем эти параметры и не могли быть незаданными или нулевыми, то в последней версии электронных таблиц от Microsoft начальные значения по умолчанию для пределов решения вовсе не заданы. Вот Excel и искал по полной программе наилучший вариант из 2^{44} = 17 592 186 044 416 возможных.

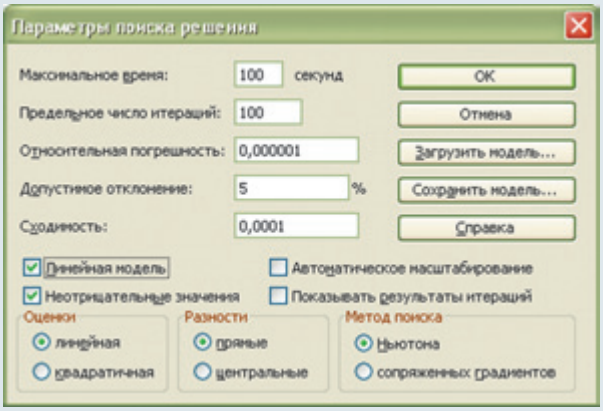


Рис. 7. Окно параметров поиска решения в Excel-2003 и 2007

Но причина затянувшегося поиска, как выяснилось, не только в этом. Для чего предназначен флажок **Линейная модель** в окне параметров поиска решения в Excel-2003 и 2007 интуитивно, может быть, не совсем понятно, видимо, по этой причине в Excel-2010 появилась разъясняющая надпись к списку выбора метода решения: “Для гладких нелинейных задач используйте поиск решения нелинейных задач методом ОПГ[2], для линейных задач — поиск решения задач симплекс-методом, а для негладких задач — эволюционный поиск решения”. Теперь ясно, что в нашем случае обязательно следовало отметить этот

[2] Нелинейный метод обобщенного понижающего градиента.

флажок, ведь наша задача, как уже было отмечено ранее, из категории задач линейного программирования. А симплекс-метод — это именно то, что позволяет значительно сократить количество вариантов перебора, а значит, и время решения. Не помешает в нашем случае и выбор флажка **Неотрицательные значения** для переменных без ограничений.

“Процесс пошел”

После введения дополнительных ограничивающих условий и указаний на линейность ситуация резко изменилась. Теперь для получения решения чаще всего требовалось всего несколько секунд. Это значительно превосходило все предыдущие показатели. Таким образом, подтвердилось, что электронные таблицы имеют в своем составе мощные инструменты поиска решений, и что задача о рассадке все же может быть эффектно решена средствами Excel. Причем такой способ, во избежание разочарований от “ручного” подбора, вполне может быть продемонстрирован учащимся, хотя бы в доказательство огромного потенциала электронных таблиц и наметки дальнейших перспектив его освоения.

При использовании такой технологии появляются возможности для проведения исследовательского этапа решения задачи. Здесь могут быть поставлены вопросы:

— об общем количестве различных вариантов рассадки;

— о реализуемости оптимальной (равномерной) загрузки автобусов;

— об осуществимости всех вариантов рассадки, указанных в табл. 2, при данном распределении количества учащихся по классам.

Для ускорения поиска ответа на последний вопрос может быть организована параллельная работа в группах с элементами состязательности и самостоятельным распределением подзадач. Как показала практика (в подтверждение теории), решения находятся заметно быстрее, если ограничения заданы более жестко, то есть, если в ограничениях вместо нестрогого неравенства используется равенство, а вместо целевого максимума или минимума задается конкретное значение. Правда, в этом случае более вероятно получить сообщение, показанное на *рис.* 8, которое в уточненной интерпретации Excel-2010 представляется как: “В ходе поиска решения не удается найти точку, для которой выполняются все ограничения”. Конечно, это не доказательство отсутствия решения, но тем не менее полезная информация.

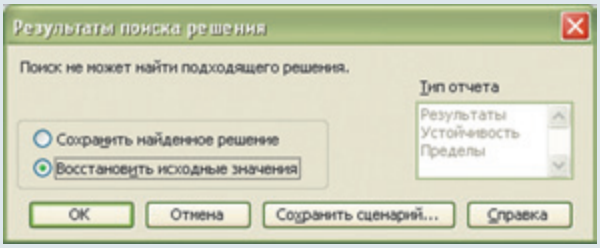


Рис. 8. Решения найти не удалось

Еще один прием, позволяющий ускорить получение результата, заключается в том, что в ячейку F13, предназначенную для подсчета количества учеников в последнем автобусе, вводится формула, вычисляющая это значение как остаток после укомплектования предыдущих автобусов =B13-СУММ(C13:E13). В этом случае диапазон изменяемых ячеек может быть сокращен за счет столбца F, что потребует анализа меньшего числа вариантов.

Казусы

Если сокращение времени поиска в случае более жестких условий вещь вполне предсказуемая и объяснимая, то некоторые “выверты” со стороны MS Excel, выявленные в процессе экспериментов, с позиций по крайней мере моих представлений, не могут быть убедительно истолкованы. Об одном из них — возможности выдачи неидеальных нулей и единиц даже в случае двоичных и целочисленных ограничений, уже упоминалось ранее, но это не весь “репертуар” приколов от Excel.

Выяснилось, что на компьютере с двухядерным процессором и тактовой частотой 2,16 ГГц, при условии целочисленных значений на изменяемые ячейки, Excel-2010 тратит на поиск решения задачи о минимальном количестве детей в последнем автобусе 45 с, а при двоичных ограничениях — почему-то целых 80 с. В то же время Excel-2003, при тех же настройках и требованиях, в первом случае справляется с заданием всего за 3 с, а во втором ему достаточно даже 1 с. Где же логика и где прогресс? Вопрос остается открытым.

Перевозим груз

Впрочем, и без того у нас уже достаточно опыта, чтобы взяться за вторую задачку о перевозе оборудования и материалов. Сначала преобразуем авторский вариант заготовки для решения задачи, более подходящий для простого подбора, к виду, удобному для применения технологии поиска решений, как это показано в табл. 4.

В этой задаче, по сравнению с предыдущей, меньше как ассортимент, так и количество транспортных единиц, зато допустимо распределение оборудования и материалов одного наименования по нескольким грузовикам. А ограничение прежде всего накладывается не на число единиц товара, а на грузоподъемность автомобиля. Хотя понятно, что груз должен быть вывезен весь. Несмотря на сокращение числа строк и столбцов, задача о перевозке груза не стала более легкой, чем предыдущая, если ее решать способом подбора вручную.

Для контроля загруженности машин воспользуемся уже знакомой нам функцией =СУММПРОИЗВ($C2:$C7;D2:D7), адаптированной для данного случая, которую внесем в ячейку D8 и распространим вправо на остальные грузовики. Вместо

Таблица 4

	A	B	C	D	E	F	G
1	**Наименование оборудования**	**Кол-во**	**Масса единицы груза (кг)**	**Грузовик-1**	**Грузовик-2**	**Грузовик-3**	**Остаток (единиц)**
2	Станки (штуки)	23	850				
3	Трубы (упаковка)	17	1930				
4	Буровое оборудование (ящики)	14	1700				
5	Отделочный камень (ящики)	7	1250				
6	Промышленные электромоторы (штуки)	18	730				
7	Кабель (бухты)	5	1100				
8	**Всего груза в машине (кг):**						

двоичности данных изменяемых ячеек диапазона $D$2:$F$7 однозначно придется использовать ограничения на их целочисленные неотрицательные значения. Так же, как в случае с экскурсией, чтобы отследить число единиц погруженных изделий одного наименования, можно правее таблицы в столбце G построчно суммировать данные изменяемого диапазона. Правда, при этом нам придется задавать ограничения для каждой строки отдельно, так как количество единиц груза разных наименований по условию различно. Но на этом пути нет особых препятствий, кроме дополнительной траты времени, так как в справке к инструменту **Поиск решений** на этот счет имеется следующее разъяснение: "…Линейная модель… позволяет задать любое количество ограничений. При решении нелинейных задач на значения изменяемых ячеек можно наложить более 100 ограничений, в дополнение к целочисленным ограничениям на переменные". Попутно отметим, что допускается задание до 200 изменяемых ячеек.

Тем не менее в данной ситуации можно "схитрить", если с помощью формул вида =B2-СУММ(D2:F2) контролировать оставшийся, а не погруженный товар. Так как увезти необходимо весь груз, то остатки в каждой строке в итоге должны быть нулевыми. Это позволяет сделать ограничение сразу на целый диапазон и сэкономить время ввода ограничений.

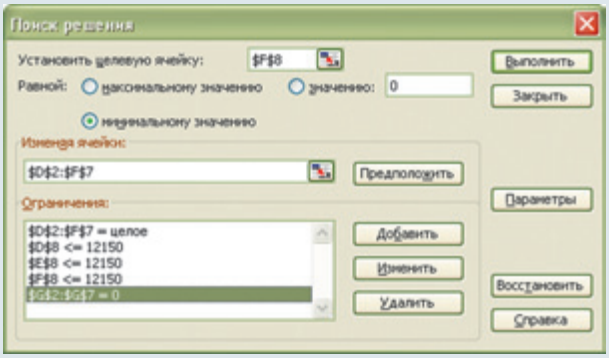


Рис. 9. Ограничения по условию задачи, начальный вариант

Впрочем, и для случая суммирования в столбце G возможен краткий вариант записи ограничений на число единиц товара. Данные в столбцах B и G в итоге должны совпасть. Как удалось установить экспериментальным путем позже, допустима запись ограничений на поэлементное сравнение диапазонов, то есть условие $B$2:$B$7=$G$2:$G$7 легитимно и недвусмысленно. Оно означает, что последовательно будут проверяться соответствующие элементы двух диапазонов на равенство. Как говорится, "на всякого мудреца довольно простоты". Кстати, верхние границы значений смежных ячеек D8, E8, F8 одинаковы, так что и в этом случае три ограничения для них можно заменить одним неравенством для диапазона: $D$8:$F$8<=12150.

Эксперименты, варианты

Первый результат, полученный при заданных ограничениях: 11 930 кг, 12 120 кг, 12 030 кг, — вполне удовлетворяет условию задачи, но два грузовика перегружены, причем один из них близко к предельному, а один — недогружен. Повторный запуск приводит к еще более маргинальному решению: 12 150 кг, 12 080 кг, 11 850 кг.

Чтобы добиться более равномерной загрузки машин, попробуем сформулировать условия ограничения иначе, установив не верхние, а нижние пределы массы перевозимого груза, как на *рис.* 10.

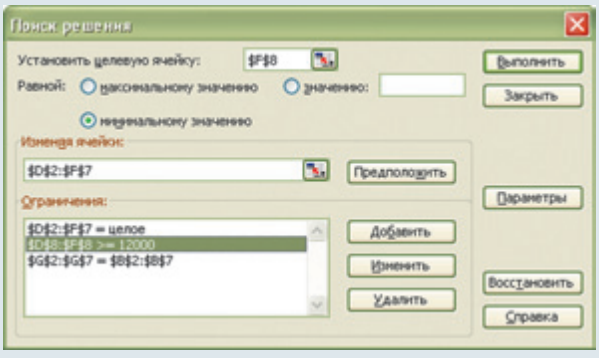


Рис. 10. Ограничения снизу и по совпадению диапазонов (краткая форма)

Получаем довольно приличный результат: 12 020 кг, 12 060 кг, 12 000 кг. Попытаемся его

улучшить. Масса всего груза составляет 36 080 кг. В идеале при равномерном распределении товара перегрузка должна составлять 80 / 3 = 26,7 кг, но так как масса единицы груза указана с точностью до десяти килограммов, то придется установить предел в 30 кг. Как выясняется, даже в этом случае решение не единственно. Вот тому пара примеров (см. табл. 5).

На поиск каждого из них Excel затратил не более 20 с. Так что величина максимальной перегрузки в 150 кг, указанная в условии задачи, была явно завышенной и ориентированной на возможно быстрое получение приемлемого результата, при условии ручного подбора. Для нахождения в тех же условиях варианта наиболее равномерной загрузки пришлось бы потратить немало времени.

Демонстрация возможностей электронных таблиц на примере этого задания, после опыта подбора, была бы эффективным стимулирующим средством к их освоению. Что касается исследовательской части задания о перевозке груза, то ее в качестве небольшого самостоятельного проекта желательно поручить если не шестиклассникам, то по крайней мере учащимся старших классов в рамках элективного или факультативного курса, если есть такая возможность.

Системы уравнений

До сих пор технологию **Поиск решения** мы испытывали на задачах оптимизации, но помнится, что при подключении этой настройки она была рекомендована прежде всего как “инструмент для поиска решения уравнений”. Можно было бы начать с какого-нибудь простенького линейного уравнения, но, учитывая наш предыдущий опыт, не будем мелочиться и попробуем проверить “поиск” сразу на системе трех линейных уравнений с тремя неизвестными, взятой из школьного учебника (см. *рис.* 11).

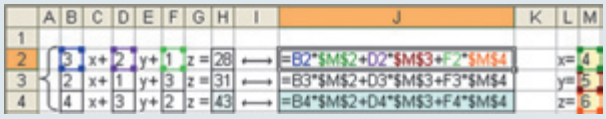


Рис. 11. Решение системы трех линейных уравнений

Данные в столбцы от A до G включительно внесены для наглядности и реализации массовости. Если в уравнениях перенести свободные члены в левую часть, то отпадет необходимость и в столбце H. Для решения взятой конкретной задачи достаточно было бы сразу ввести соответствующие коэффициенты в формулы столбца J, что даже упростило бы оформление.

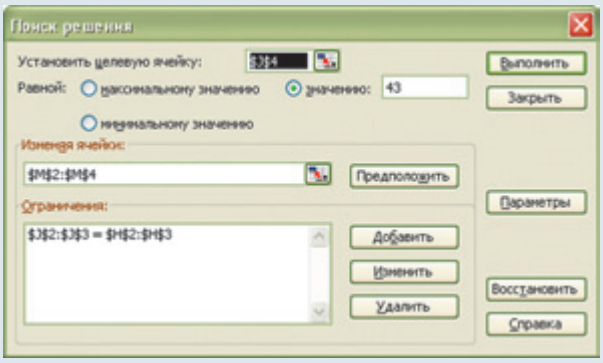


Рис. 12. Настройка поиска корней системы уравнений

В качестве целевой ячейки можно выбрать любую из области J2:J4. В нашем случае выбрана последняя, для которой укажем конкретное значение, равное свободному члену в третьем уравнении. В поле **Изменяя ячейки** зададим диапазон M2:M4, выделенный под значения искомых корней. Осталось задать ограничение на равенство свободных членов первых двух уравнений соответствующим результатам формул, вычисляющих их левые части. Естественно, что в параметрах должен быть отмечен флажок **Линейная модель**. Не помешает для данной системы задать в ячейках с корнями уравнений числовой формат без десятичных знаков, хотя в иных ситуациях лучше один-два знака оставлять.

Щелчок по кнопке **Выполнить**. Excel почти мгновенно блестяще справляется с заданием, не вычисляя детерминантов и не используя метод Гаусса.

Квадратные уравнения

Разобравшись с линейными уравнениями, вернее, их системами, попробуем найти с помощью инструмента **Поиск решения** корни квадратного уравнения, например: $x^2 - 8x + 15 = 0$. Выберем ячейки B4, D4 и F4 для ввода коэффициентов a, b и c соответственно (см. *рис.* 13). В ячейку C6 внесем формулу для вычисления значения левой части

Таблица 5

Наименование	Кол-во	Грузовик-1	Грузовик-2	Грузовик-3	Грузовик-1	Грузовик-2	Грузовик-3
Станки	**11**	3	5	3	0	9	2
Трубы	**4**	3	1	0	3	0	1
Буры	**2**	1	0	1	0	0	2
Камень	**4**	1	0	3	0	0	4
Моторы	**7**	1	5	1	1	6	0
Кабель	**5**	0	2	3	5	0	0
Всего (кг):		12020	12030	12030	12020	12030	12030

уравнения, в которой в качестве переменной будет использована ссылка на ячейку F6.

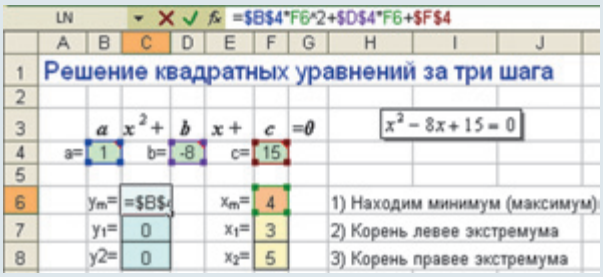


Рис. 13. Решение квадратных уравнений за три шага

Решение через экстремум

Тут возникает проблема. Дело в том, что в этом случае мы можем указать только на одну изменяемую ячейку, а корней в рассматриваемом уравнении два. Можно предложить следующий выход из создавшейся ситуации. Вспомним, что если квадратное уравнение имеет корни, то один из них находится левее точки экстремума квадратного трехчлена, а второй — правее. Что касается минимумов и максимумов, то эта задачка для рассматриваемой технологии является родной. Исходя из этого, с помощью маркера автозаполнения копируем формулу из ячейки C6 в нижележащие C7, C8 и намечаем следующий план действий:

1) Находим значение аргумента x_m, при котором левая часть уравнения принимает наименьшее (наибольшее) значение. В нашем случае получаем $y_m = -1$ при $x_m = 4$.

2) Вводим дополнительное ограничение $x_1 \le x_m$ ($F$7<=4) и находим первый корень.

3) Аналогично находим в ячейке F8 второй корень.

Так как мы теперь имеем дело не с линейным уравнением, то флажок **Линейная модель** в окне **Параметры поиска решений** придется снять.

По теореме Виета

В ограничениях MS Excel-2010 допустимо задание условия “Все разные”. Попытки объединить с его помощью последние два шага нахождения корней в один оказались неудачными, но желание упростить описанный способ решения не иссякло. При этом вариант использования симметричности корней относительно точки экстремума и автоматического нахождения второго корня по первому в расчет не брался. И тут вспомнилась теорема Виета, которая зачастую помогает решать квадратные уравнения фактически методом подбора. С ее помощью квадратное уравнение превращается в систему двух уравнений с двумя неизвестными, а это уже знакомый вариант. Правда, одно из уравнений этой системы линейным назвать нельзя, но это препятствие, как уже описывалось, легко преодолевается щелчком мышки.

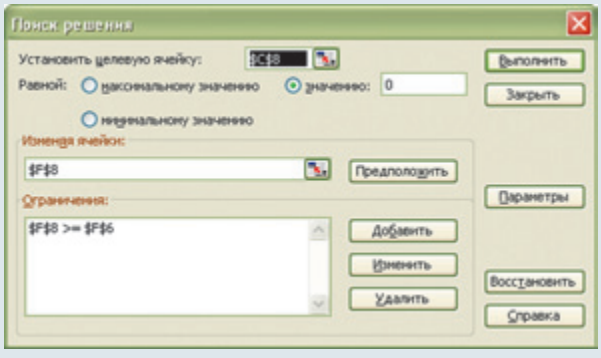


Рис. 14. Ищем второй корень квадратного уравнения

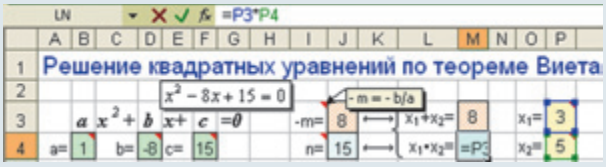


Рис. 15. Решение квадратного уравнения за один прием

По коэффициентам a, b и c данного квадратного уравнения в ячейках J3, J4 сначала вычисляем коэффициенты m и n приведенного квадратного уравнения, один из которых будет определять значение целевой ячейки M4 как произведения корней, а второй — использоваться в ограничениях на сумму корней, как это показано на *рис.* 15 и 16.

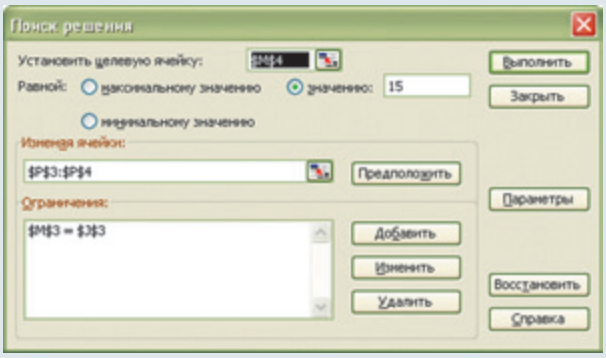


Рис. 16. Поиск корней по теореме Виета

Здесь неожиданно возникла проблема. При нажатии на кнопку **Выполнить** в каждой из изменяемых ячеек для x_1 и x_2 неизменно появлялось число 3,8729833899282 и сообщение: “Поиск не может найти подходящего решения”. Попытки розыска возможных ошибок в записи или настройках оказались безуспешными. Разгадка была найдена случайно. Дело в том, что до сих пор не было необходимости вводить какие-либо начальные значения для изменяемых ячеек. Они, вероятно, воспринимались по умолчанию как нулевые. Похоже, что в данной задаче, возможно по причине симметричности аргументов в уравнениях, равенство нулю начальных значений изменяемых ячеек воспринималось как требование равенства обоих корней. Стоило только до нажатия кнопки **Выполнить** внести в ячейки P3 и P4 произвольные, но разные числа, как решение находилось мгновенно и без проблем.

Трансцендентные уравнения

Рассмотренные выше уравнения и системы уравнений имеют алгебраические решения в общем виде, используя которые можно “обучить” Excel находить корни с еще большей эффективностью. Правда, такое

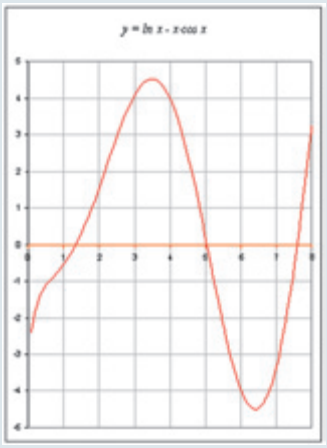
“обучение” должно быть индивидуальным для каждого типа уравнений. Что касается инструмента **Поиск решений**, то он является универсальным, пригодным для довольно широкого круга задач, в том числе и для уравнений, решение которых алгебраическими преобразованиями невозможно.

Для примера попробуем найти корни трансцендентного уравнения $\ln x - x \cdot \cos x = 0$ на интервале $(0; 8]$. В целевую ячейку B4 введем формулу для вычисления левой части уравнения и скопируем ее в несколько нижележащих клеток, на случай, если корней окажется два или более. В качестве изменяемой будем выбирать ячейку соответствующей строки столбца E. Так как незаданные исходные значения аргументов воспринимаются как нулевые, а логарифмическая функция при этом неопределенна, то во избежание проблем зададим начальные значения в изменяемых ячейках чуть правее нуля, например, 0,01 (см. *рис.* 17, 18).

Получив первый корень $x_1 = 1{,}3475803077784$, попробуем найти второй корень правее при ограничении $E$5 >= 1,4. Оказывается, что такой корень есть: $x_2 = 5{,}03910616296059$. При условии $E$6 >= 5,1 поиск увенчался успехом и в этом случае: $x_3 = 7{,}5835423651778$. Попытки найти корни между найденными значениями заканчивались сообщением: “Поиск не может найти подходящего решения”, — но проверка на максимум или минимум на каждом из полученных промежутков, как и положено, выдавала или граничные, или промежуточные значения.

Фактически с помощью инструмента **Поиск решения** мы не только нашли все корни, но и исследовали функцию $y = \ln x - x \cdot \cos x$ на экстремумы и промежутки знакопостоянства, что позволяет сделать заключение о применимости этой технологии не только для решения уравнений и задач оптимизации, но и для решения неравенств.

Для проверки полученных результатов исходное уравнение было решено в Excel графическим методом, а корни уточнены с помощью инструмента **Подбор параметра**. С точностью до пяти знаков после запятой корни совпали, а данные исследований на экстремумы и промежутки знакопостоянства подтвердились, что является еще одним убедительным аргументом в пользу эффективности применения надстройки **Поиск решения**.



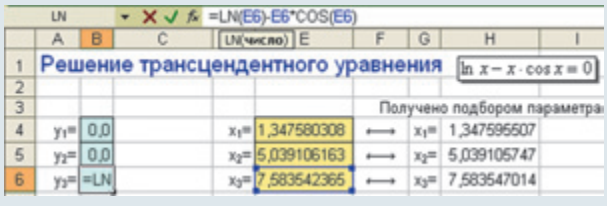


Рис. 17. Решение трансцендентного уравнения

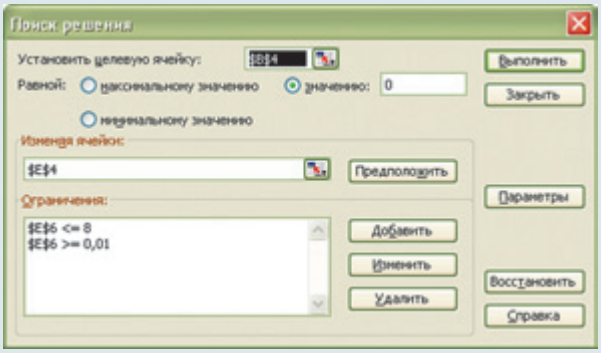


Рис. 18. Поиск первого корня трансцендентного уравнения

Заключение

Вот и близится к концу почти детективная история, начавшаяся, казалось бы, с пустяковой задачи для шестиклассников и приведшая (по недоразумению) поиск ее решения, как ни парадоксально, к **Поиску решения** и исследованию возможностей этого инструмента.

Сама надстройка в MS Excel-2003 находится по адресу: C:\Program Files\Microsoft Office\OFFICE11\Library\SOLVER\ (файлы SOLVER.XLA и SOLVER32.DLL). Кроме этого, по адресу: C:\Program Files\Microsoft Office\OFFICE11\SAMPLES\ есть файл SOLVSAMP.XLS с примерами. В файле приводится краткий обзор технологии работы с надстройкой **Поиск решения** и разбирается несколько примеров решения с ее помощью финансовых, производственных и технических задач, мало подходящих для школьного уровня. Другие сведения об инструменте и технологии работы с ним можно почерпнуть из справки к MS Excel.

Варианты решения всех рассмотренных в этой статье задач имеются на диске, прилагаемом к этому номеру журнала. Следует иметь в виду, что в случае использования инструмента поиска для нахождения нескольких последовательных решений на одном рабочем листе, в файле сохраняются только настройки последнего решения.

Как уже отмечалось, предлагаемые примеры, приемы и находки, надеюсь, могут послужить в качестве иллюстративного материала для демонстрации потенциала электронных таблиц, альтернативных способов решения известных задач математики, стать основой самостоятельного творческого проекта или войти в качестве составной части в элективный или факультативный курс, наконец, просто помочь в освоении технологии применения инструмента **Поиск решения** и тем самым поучаствовать в продолжении этой истории.

Желаю удачи!

Использованные источники информации

1. *Сенокосов А.И.* Ёксель! “Информатика” № 2/2011.
2. Справочная система MS Excel.

Модульные курсы
«Навыки личной эффективности»

Педагогический университет **Первое сентября**

Лицензия 77 № 000349, рег. № 027477 от 15.09.2010, выдана Департаментом образования г. Москвы

Модульные курсы предоставляют уникальную возможность:

- начать обучение в любой момент;
- выбирать удобный график освоения материалов и самостоятельно определять срок окончания изучения модуля (минимальный срок обучения – 1 месяц);
- выполнять контрольную работу в режиме онлайн;
- осваивать знания из психологии, менеджмента, экономики, которые позволят: лучше понять себя и других людей; психологические причины возникновения стрессов и различных заболеваний и сохранить свое здоровье; оптимизировать свою деятельность и др.

Нормативный срок освоения каждого модуля – 6 часов. Форма обучения – дистанционная. После успешного окончания модуля выдается сертификат.

Стоимость одного модульного курса – 200 руб.

ПЕРЕЧЕНЬ МОДУЛЬНЫХ КУРСОВ

очень популярен!

Тайм-менеджмент, или Как эффективно организовать свое время.

Тайм-менеджмент для детей, или Как научить школьников организовывать свое время.

Приемы конструктивного разрешения конфликтных ситуаций, или Конфликты в нашей жизни: способы решения.

очень популярен!

Профессиональное выгорание, или Как сохранить здоровье и не «сгореть» на работе.

Стресс-менеджмент, или Приемы профилактики и преодоления стресса.

Управление имиджем, или Как создать свой стиль.

ПОДАЙТЕ ЗАЯВКУ НА ОБУЧЕНИЕ НА САЙТЕ http://edu.1september.ru

Получить более подробную информацию можно на сайте, по электронной почте: **module@1september.ru** или по телефону **(499) 249-47-82**

Кодирование информации: фибоначчиева система и компьютер

И.Н. Фалина,
СУНЦ МГУ,
Москва

Методическое отступление. Уважаемые коллеги, у каждого из вас наверняка была ситуация, когда в начале изучения темы “Системы счисления” по классу пробегает шумок, лица ребят скучнеют и слышатся реплики: “Ну, опять эти системы счисления…”. Задачи по теме кодирования числовой информации (двоичная, восьмеричная и шестнадцатеричная системы счисления) входят в ЕГЭ и ГИА, поэтому в старших классах этой теме стало уделяться больше внимания. Но… Но, действительно, для сегодняшнего школьника эта тема в традиционном изложении скучна.

Спросите любого старшеклассника: “Какие системы счисления используются для кодирования информации в компьютерах и других цифровых устройствах?”. Большинство на этот вопрос ответят: “Двоичная система счисления”. И в основном они будут правы. Именно поэтому ребята вполне законно задают вопрос: “Зачем мне учить перевод из десятичной системы счисления в восьмеричную, зачем уметь складывать шестнадцатеричные числа, если в реальной “взрослой” жизни я с этим не столкнусь никогда?!”

А тема-то эта интересна, глубока и красива (впрочем, как и большинство тем теоретической информатики). От этой темы тянутся “веточки-цепочки” и к теме “Алгоритмизация и программирование”, и к теме “Архитектура ЭВМ”, и к теме “Устройство компьютера”, и ко многим другим. Для раскрытия глубины этой темы мне и моим коллегам из СУНЦ МГУ (школа им. А.Н. Колмогорова) видится, что в начале изучения надо поставить несколько вопросов, а в процессе изучения постараться получить на них ответы.

На первом занятии раздайте ученикам лист с такими вопросами:

1) Очевидно, что для представления информации в компьютере ее надо закодировать. Для кодирования в современных компьютерных системах традиционно используется двоичная система счисления. Какие другие системы счисления используются или использовались ранее в компьютерах?

2) Разработчики нового компьютера решили использовать для кодирования некоторую другую систему счисления. На что повлияет это решение разработчиков при проектировании компьютера?

3) Какими свойствами должна обладать система кодирования информации в компьютере?

4) Чем хороша и чем плоха двоичная система счисления?

5) Знаете ли вы, что существует другая позиционная неклассическая двоичная система счисления? Как она называется? Можете ли вы выписать ее алфавит и базис?

Проанализируйте ответы учащихся (это покажет вам, как минимум, ширину и глубину знаний по теме "Кодирование информации"). Дайте учащимся краткий ответ на каждый вопрос. Эти ответы будут "скелетом" дальнейшего изложения темы. Даже если вы не будете рассказывать ни о троичной уравновешенной системе счисления и ЭВМ "Сетунь", ни о фибоначчиевой системе счисления и фибоначчи-процессоре, упоминание об их существовании, отсылка к идее и опыту использования других систем счисления для кодирования покажет значимость выбора системы кодирования в "реальном" цифровом мире. И тогда школьникам не так тоскливо будет выполнять примеры на сложение и вычитание двоичных и восьмеричных чисел.

Ниже вашему вниманию предлагается материал, который, смею надеяться, поможет "оживить" уроки информатики, пробудить интерес к предмету "информатика", показать некоторые "пусковые точки" развития науки "информатики". Этот материал может являться основой для факультативного курса, основой исследовательских и реферативных работ учащихся старших классов.

1. Из истории использования недвоичных систем счисления в компьютерах

В современных компьютерах и иных цифровых устройствах для ввода-вывода часто используется двоично-десятичная система счисления. В 60-х годах XX столетия в МГУ им. М.В. Ломоносова была создана троичная ЭВМ "Сетунь", запущенная потом в серию. Для кодирования информации в этой машине использовалась троичная уравновешенная система счисления, вместо бита использовался трит, вместо привычной двоичной логики — троичная. Разработчик этой ЭВМ Николай Петрович Брусенцов и сейчас работает на факультете вычислительной математики и кибернетики МГУ им. М.В. Ломоносова.

Вопрос школьникам. Зачем ученые и инженеры тратят силы и время на создание компьютеров, работающих не на двоичной системе счисления?

В каждой области науки и техники существуют фундаментальные идеи или принципы, которые определяют ее содержание и развитие. В компьютерной науке роль таких фундаментальных идей сыграли принципы, сформулированные независимо друг от друга двумя крупнейшими учеными XX века — американским математиком и физиком Джоном фон Неймаком и советским инженером и ученым Сергеем Лебедевым.

Центральное место среди "принципов Неймана — Лебедева", определяющих фоннеймановскую архитектуру ЭВМ, занимает предложение об использовании двоичной системы счисления. Это предложение было обусловлено рядом обстоятельств: простота выполнения арифметических операций в двоичной системе счисления; ее "оптимальное" согласованием с булевой логикой; простота технической реализации двоичного элемента памяти (триггера).

Однако на определенном этапе развития компьютерной техники выявилось, что использование классической двоичной системы счисления для представления информации в компьютере имеет существенные недостатки, влияющие на скорость работы процессора и надежность передачи информации:

- ✓ Первым из них является так называемая *проблема представления отрицательных чисел*.
- ✓ Второй недостаток двоичной системы счисления получил название *нулевой избыточности*.

Попытка преодолеть эти и другие недостатки двоичной системы счисления стимулировала использование в компьютерах других систем счисления и развитие собственно теории систем счисления.

2. О некоторых "недостатках" традиционной двоичной системы счисления

Выбирая систему кодирования информации в компьютере, разработчики в первую очередь думают о быстродействии процессора, т.е. о скорости обработки закодированной информации. Каждая операция в современных процессорах реализуется через так называемые *предельные алгоритмы*. В термин "предельный алгоритм" вкладывается следующий смысл: вся мощь современного математического аппарата и компьютерной инженерии была использована при его создании. Например, обычное сложение целых многоразрядных чисел реализуется "быстрым" рекурсивным алгоритмом с использованием операций сдвига. Стремясь увеличить вычислительную скорость компьютера, работающего на двоичной системе, разработчики ввели особое *беззнаковое* представление отрицательных целых чисел — так называемый *дополнительный код*.

В десятичной системе счисления для представления отрицательных чисел человек использует знак числа. То, что привычно и хорошо для человека, выполняющего вычисления на бумаге, не всегда хорошо для компьютера. Для того чтобы сложить два целых числа с разными знаками, и человек, и компьютер, вообще говоря, должны выполнить следующие действия:

1) взять модули слагаемых;
2) сравнить эти модули;
3) запомнить знак большего по модулю слагаемого;
4) из большего модуля вычесть меньший модуль;
5) получившийся результат записать со знаком из п. 3.

Многовато будет для простой и часто используемой операции! Для увеличения скорости работы процессора математики предложили все отрицательные целые числа записывать в дополнительном коде [1, 2], и тогда сложение двух целых чисел стало выполняться за одну операцию (т.е. без анализа знаков слагаемых).

Методическое отступление. Если вы в своих классах рассказываете о дополнительном коде, обязательно поясните, почему его ввели, как влияет на скорость выполнения операций над целыми числами кодирование отрицательных целых чисел дополнительным кодом.

Вопрос школьникам. А можно ли записывать целые отрицательные числа без знака, но и без дополнительного кода?

Можно! Например, в любой *P*-ичной уравновешенной системе счисления все целые отрицательные числа записываются без знака [2, 6]. Потратьте один урок, расскажите о троичной уравновешенной системе.

Троичная уравновешенная система является позиционной с основанием 3. Алфавитом системы являются символы { $\bar{1}$, 0, 1}, где $\bar{1}$ равно –1. Выпишите на доске таблицу, аналогичную приведенной ниже. Глядя на эту таблицу, понятно, почему эту систему назвали уравновешенной, или симметричной.

Интересующимся школьникам можно дать доклад о правилах работы в троичной уравновешенной системе счисления. В рамках этого доклада школьник должен сформировать правила построения таблиц сложения и умножения и правило перечисления целых чисел. При серьезном отношении получится очень интересный доклад!

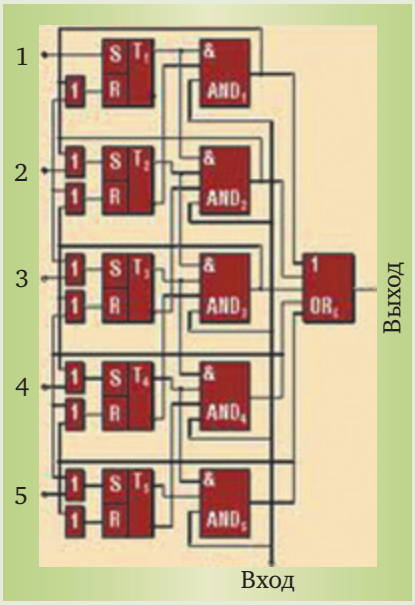
Второй недостаток двоичной системы особенно неприятен при ее использовании для передачи информации. *Нулевая избыточность* двоичного представления означает, что в двоичной системе отсутствует механизм обнаружения ошибок, которые, к сожалению, неизбежно возникают в компьютерных системах. На заре развития двоичной компьютерной техники не возникал так остро вопрос о надежности передачи больших объемов информации, так как еще не существовали локальные компьютерные сети, не было Интернета, да, собственно, не произошло еще всеобщей информатизации. Но времена меняются…

Суть этой проблемы состоит в следующем. Пусть в процессе передачи или хранения информации, представленной, например, двоичным кодом `11011010`, под влиянием внешних или внутренних факторов произошло искажение информации, и она перешла в кодовую комбинацию `11010010` (искаженный разряд выделен). Поскольку комбинация `11010010` (как и любой другой двоичный код) является “разрешенной” в двоичной системе счисления, то без дополнительных действий невозможно определить, произошло искажение информации или нет. Для решения этой проблемы можно, например, для каждого байта (8 разрядов двоичного числа) считать количество единиц, или для группы байтов считать контрольную сумму и т.д. В любом случае должны быть использованы специальные методы избыточного кодирования, что замедляет работу компьютера и требует дополнительной памяти.

В условиях, когда человечество все больше и больше зависит от надежности работы компьютерных систем (управление ракетами, самолетами, атомными реакторами, банковскими системами), вопрос об эффективных механизмах обнаружения ошибок выдвигается на передний план.

Десятичное положительное число	Положительное число в троичной уравновешенной системе	Отрицательное число в троичной уравновешенной системе	Десятичное отрицательное число
1	1	$\bar{1}$	–1
2	1 $\bar{1}$	$\bar{1}$ 1	–2
3	1 0	$\bar{1}$ 0	–3
4	1 1	$\bar{1}$ $\bar{1}$	–4
5	1 $\bar{1}$ $\bar{1}$	$\bar{1}$ 1 1	–5
6	1 $\bar{1}$ 0	$\bar{1}$ 1 0	–6
7	1 $\bar{1}$ 1	$\bar{1}$ 1 $\bar{1}$	–7

Для компьютеров, основанных на классической двоичной системе счисления, не всегда можно эффективно решать эту проблему. В 80-х годах XX столетия группа ученых под руководством профессора Алексея Петровича Стахова из Таганрогского радиотехнического института создала опытный экземпляр помехоустойчивого процессора [3]. Этот процессор мог сам контролировать возникающие в его работе сбои. Для кодирования информации была выбрана фибоначчиева система счисления. Ее использование позволило построить удивительный процессор, названный “Фибоначчи-процессор”, или “Ф-процессор”. И хотя успешная попытка построения помехоустойчивого процессора на основе фибоначчиевой системы счисления носила скорее теоретический, чем практический интерес, изучение этой замечательной системы счисления заслуживает внимания.



3. Понятие избыточности информационной системы

Методическое отступление. Прежде чем вы начнете знакомить школьников с фибоначчиевой системой счисления, надо обязательно рассказать о понятии избыточности системы кодирования.

Известно, что каналы, по которым передается информация, практически никогда не бывают идеальными (каналами без помех). Помехи в каналах образуются по разным причинам, но результат их воздействия на передаваемую информацию всегда один — информация теряется (искажается). В теории связи известно, что избыточность кода позволяет обеспечивать надежную защиту системы и ее помехоустойчивость.

Под **потерей** информации мы будем понимать **искажение** информации при сохранении переданного объема данных.

Например, по каналам связи был передан код доступа к серверу в виде двоичного кода `10101100 10100000 11100000 11100001` (слово “марс” в кодировке ASCII). Длина переданного кода (т.е. объем информации) равна четырем байтам, или 32 битам. Сервер получил этот код в виде:

`10101100 10100100 11100000 11100001`.

Проанализируем ситуацию:

1) формально передача завершена успешно, так как полученный код является допустимой комбинацией в двоичной системе счисления;

2) длина принятого кода равна длине отправленного кода (при успешной передаче нельзя физически потерять один или несколько битов, можно только изменить содержимое битов);

3) произошло изменение (искажение) содержимого одного бита: с точки зрения пользователя, информация при передаче пропала (слово “мдрс” не равно слову “марс”).

В результате пользователю в доступе к серверу отказано!

При передаче информации всегда важно знать ответы на два вопроса:

1) произошло ли искажение (потеря) информации?

2) если произошло искажение, то в каких битах (разрядах)?

Для ответа на эти вопросы необходимо, чтобы передаваемые коды обладали свойством *избыточности*. Или, что то же самое, символьная система, при помощи которой кодируется информация, должна обладать свойством избыточности.

В информатике под избыточностью понимают следующее: код обладает свойством **избыточности**, если количество бит в нем больше, чем это необходимо для однозначного декодирования (восстановления) исходной информации.

Информация кодируется не только в компьютерах. Мы можем считать, что наша письменная речь — это информация, закодированная русскими буквами. Представьте, что вам в руки попадет записка с частично стертыми буквами:

`Вы__т со_т__т__ в н_ч_ле н__бр_.`

Подумав, вы сможете восстановить текст этой записки, причем однозначно. А ведь из 27 букв записки 12 букв были уничтожены! Отметим, что для однозначного восстановления потерянной информации важно знать, на каких местах стояли стертые буквы.

Вы смогли восстановить текст записки потому, что русский язык обладает *избыточностью*. В русском языке существуют слова, однозначно прочитываемые в случае “потери” некоторых букв. Например, С_НТ_БР_, КИ_ТЬ, Д_Р_В_.

Имея текст на русском языке с частично “потерянными” буквами, человек, хорошо владеющий русским языком, сможет однозначно восстановить его. Например, вы без труда прочитаете предложение с пропущенными буквами “П_тр И_ _ч Ч_йк_ _ск_й – в_д__щ_й__ р_с__ий к_мп__ит_р”. Однако

если это предложение будет читать иностранец, едва знающий русский язык и русскую историю, то он не сможет понять, что написано в данном предложении. Мы, носители русского языка, можем с легкостью восстановить окончания, пропущенные буквы в слогах, и можем подобрать подходящие слова (из тех, что нам известны). А иностранцу просто не с чем сравнивать получаемую информацию! Таким образом, для носителя языка обычный связный текст на его родном языке содержит избыточную информацию — ее можно удалить, но смысл текста (для носителя языка) сохранится.

Избыточностью обладает не только русский, но и любой другой язык. Выдающийся американский инженер и математик, основатель теории информации Клод Шеннон проделал такой эксперимент. В литературном английском тексте были удалены 50% букв. Пробелы между словами оставались неизменными, буквы удалялись, конечно, не случайным образом. Например, в английских словах за буквой *q* всегда следует только буква *u*, поэтому "потерянная" буква *u* может быть всегда восстановлена. После "искажения" текст давали читать носителям языка, обладающим достаточно высоким культурным уровнем. Поразительно, но практически все участники этого эксперимента смогли восстановить искаженный текст полностью!

Итак, для обнаружения и восстановления потерянной информации необходимо, чтобы символьная система, при помощи которой кодируется информация, обладала избыточностью. К сожалению, двоичная система счисления не обладает избыточностью, поэтому и говорят, что она обладает *нулевой избыточностью*.

С целью предотвращения потери информации при передаче данных, кодированных в двоичной системе счисления, *были придуманы избыточные коды* (коды с избыточностью). Существуют избыточные коды с обнаружением (они только обнаруживают ошибку) и коды с исправлением (эти коды обнаруживают место ошибки и исправляют ее).

В разных каналах передачи информации возникают разные помехи. Поэтому инженеры и ученые вынуждены разрабатывать различные по своей структуре и избыточности коды. Так, при записи информации на лазерные диски применяют коды с избыточностью по длине в 25%. Примерно с такой же избыточностью применяют коды в системах цифрового спутникового ТВ.

Понятно, что использование избыточных кодов приводит к снижению скорости работы компьютеров, увеличению объема передаваемых сообщений и, соответственно, снижению скорости передачи информации. Известно, и мы это покажем, что фибоначчиева система счисления обладает избыточностью.

4. Фибоначчиева система счисления

В статье вместо длинного (но красивого) названия "фибоначчиева система счисления" будем использовать сокращение ФСС. Для указания, что число записано в ФСС, будем использовать в нижнем индексе сокращение *fib*. Например, $10000101_{fib} = 38_{10}$.

Числа Фибоначчи — элементы числовой последовательности 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10 946, …, в которой каждое последующее число, начиная с третьего, равно сумме двух предыдущих чисел.

Для формального определения чисел Фибоначчи используют следующее рекуррентное соотношение:

$$F_0 = 1,\ F_1 = 1,\quad F_n = F_{n-2} + F_{n-1}\ .$$

Последовательность, известная у нас как числа Фибоначчи, использовалась в Древней Индии задолго до того, как стала известна в Европе после изучения и описания ее Леонардо Пизанским Фибоначчи (1170–1250).

Леонардо Пизанский Фибоначчи. Благодаря книге Фибоначчи "Liber Abaci" Европа узнала индо-арабскую систему чисел, которая позднее вытеснила традиционные для того времени римские числа.

Фибоначчиева система счисления относится к позиционным системам. Алфавитом фибоначчиевой системы являются цифры 0 и 1, а ее базисом — последовательность чисел Фибоначчи 1, 2, 3, 5, 8, 13, 21, 34, 55, 89 … . Обратите внимание, что $F_0 = 1$ в базис не включается.

В табл. 1 на с. 41 перечислены некоторые числа в двоичной и фибоначчиевой системах счисления.

Фибоначчиева система является разновидностью двоичной системы — ее алфавит составляют цифры 0 и 1. Следовательно, эту неклассическую двоичную систему счисления, вообще говоря, мож-

Таблица 1

Десятичное число	Запись числа в двоичной системе счисления	Запись числа в фибоначчиевой системе счисления
1	1	1
2	10	10
3	11	100 или 11
4	100	101
5	101	1000 или 110
10	1010	10010 или 1110
20	10100	101010
50	110010	10100100 или 10011100 или 10100011 или 10011011

но использовать для кодирования информации в компьютере, так как элементная база современной компьютерной техники ориентирована на обработку двоичных последовательностей.

4.1. Алгоритмы перевода целых чисел из ФСС в десятичную систему и обратно

Как известно, все позиционные системы устроены одинаково и, следовательно, перевод из любой позиционной системы счисления в десятичную осуществляется по одному и тому же алгоритму.

В P-ичных системах счисления базис является геометрической прогрессией. Вклад в значение числа цифры a, стоящей на k-м месте слева, равен $a \cdot P^k$, где P — основание системы счисления. Часто говорят, что "вес" k-го разряда равен P^k.

В ФСС "вес" каждого разряда числа также определяется базисом этой системы. Для удобства дальнейшей работы выпишем "веса" первых 10 разрядов ФСС (нумерацию разрядов ведем справа налево, начиная с первого). Такая нумерация разрядов удобна, поскольку в качестве веса k-го разряда используется k-е число Фибоначчи.

10-й разряд	9-й разряд	8-й разряд	7-й разряд	6-й разряд	5-й разряд	4-й разряд	3-й разряд	2-й разряд	1-й разряд
89	**55**	**34**	**21**	**13**	**8**	**5**	**3**	**2**	**1**

Пример 1. Пусть нам дано число $A_{fib} = 10101010_{fib}$, записанное в фибоначчиевой системе счисления. Чему равно это число в десятичной системе счисления?

Чтобы ответить на этот вопрос, запишем цифры числа в разрядную сетку, затем умножим каждую цифру на вес разряда и сложим полученные числа. Так как цифрами фибоначчиевой системы счисления являются 0 и 1, то нам достаточно сложить веса тех разрядов, где стоят единицы.

8-й разряд	7-й разряд	6-й разряд	5-й разряд	4-й разряд	3-й разряд	2-й разряд	1-й разряд
34	**21**	**13**	**8**	**5**	**3**	**2**	**1**
1	**0**	**1**	**0**	**1**	**0**	**1**	**0**

Получим: $A_{fib} = 10101010_{fib} = 34 + 13 + 5 + 2 = 54_{10}$.

Алгоритм перевода целых чисел из фибоначчиевой системы счисления в десятичную

1. Напишем над каждой цифрой в фибоначчиевой записи числа, начиная с младшей цифры, вес соответствующего разряда.
2. Сложим все числа, стоящие над единицами. Полученное число будет десятичным эквивалентом фибоначчиева числа.

Пример 2. Решим обратную задачу. Запишем в фибоначчиевой системе счисления десятичные числа 10_{10}, 25_{10} и 100_{10}. Для решения нашей задачи достаточно подобрать такие числа Фибоначчи, сумма которых равна исходному десятичному числу. Например, число 10 можно представить суммой следующих чисел Фибоначчи: $10_{10} = 5 + 3 + 2$.

Это позволяет записать нам 10_{10} в виде 1110_{fib} (выполнили разложение по базису).

Однако это же число 10_{10} можно записать в фибоначчиевой системе счисления и по-другому:

$$10_{10} = 10010_{fib} = 1 \cdot 8 + 0 \cdot 5 + 0 \cdot 3 + 1 \cdot 2 + 0 \cdot 1.$$

Аналогично, число 25_{10} можно также записать несколькими способами:

$$25_{10} = 1000101_{fib} = 110101_{fib}.$$

Число 100_{10} можно записать как минимум шестью способами:

$$100_{10} = 1000010011_{fib} = 1000010100_{fib} = 110010011_{fib} = 110010100_{fib} = 101110011_{fib} = 101110100_{fib}.$$

На основе примера 2 можно сформулировать алгоритм перевода целых чисел из десятичной системы счисления в одно из представлений фибоначчиевой.

Алгоритм перевода целых чисел из десятичной системы счисления в ФСС

1. Найдем максимальное число F из базиса фибоначчиевой системы, которое не превосходит число A, и зафиксируем его номер k ($F = F_k$). Искомая запись числа A в фибоначчиевой системе будет содержать k цифр. Сформируем "заготовку" искомого фибоначчиева числа в виде $10\underbrace{...0}_{k-1}$.
2. Вычислим разность $A = A - F_k$.
3. Если полученное значение A равно 0, переход на п. 7.
4. Найдем максимальное число F из базиса фибоначчиевой системы, которое не превосходит число A, и зафиксируем его номер p ($F = F_p$).
5. В сформированную заготовку на p-е место поставим 1.
6. Положим $k = p$. Переход на п. 2.
7. Сформированная "заготовка" является искомым числом в ФСС. Конец алгоритма.

4.2. Избыточность ФСС. Операции свертки и развертки в ФСС

Методическое отступление. После знакомства с алфавитом и базисом ФСС, с правилами перевода целых чисел из фибоначчиевой системы счисления в десятичную и обратно следует обратить внимание на такое важное свойство ФСС, как избыточность. Избыточность ФСС проявляется в различных кодовых представлениях одного и того же числа.

Фибоначчиева система счисления обладает удивительным свойством — *свойством избыточности*, оказывается, что многие числа в ФСС могут выглядеть по-разному. Например, десятичное число 40 можно записать в ФСС четырьмя разными способами:

$$40_{10} = 10001001_{fib} = 10000111_{fib} = 1101001_{fib} = 1100111_{fib}.$$

Различные фибоначчиевы представления одного и того же числа могут быть получены друг из друга с помощью специальных фибоначчиевых операций **свертки** ($011 \Rightarrow 100$) и **развертки** ($100 \Rightarrow 011$). Эти операции выполняются над фибоначчиевой записью числа и не меняют значения числа. Покажем это.

Пусть в записи числа A первая слева комбинация "11" стоит на $(p + 1)$-м и p-м местах (на $(p + 2)$-м месте обязательно стоит 0). Тогда значение числа A равно:

$$A = B + F_{p+1} + F_p + C,$$

где B — сумма чисел Фибоначчи, соответствующих ненулевым разрядам в записи числа до комбинации "11", C — сумма чисел Фибоначчи, соответствующих ненулевым разрядам в записи числа после комбинации "11". В соответствии с тождеством

$$F_k = F_{k-1} + F_{k-2} \quad (1)$$

мы можем написать $A = B + F_{p+2} + C$. Это значит, что комбинация "011" заменилась комбинацией "100", но значение числа A не изменилось.

Например, $A = 10011_{fib} = \overset{8\,5\,3\,2\,1}{10011} = \overset{8\,5\,3\,2\,1}{10100} = 11_{10}$.

Аналогичные рассуждения применимы и к операции развертки.

Замечание 1. Операция свертки **уменьшает** количество единиц в фибоначчиевой записи числа, но может увеличить на единицу количество цифр в записи, если исходная запись начиналась с двух единиц. Например, $\underline{11}0101_{fib} = \underline{100}0101_{fib}$.

Действительно, пусть в записи числа A было k цифр, из них две старшие равны 1. Тогда значение числа A равно:

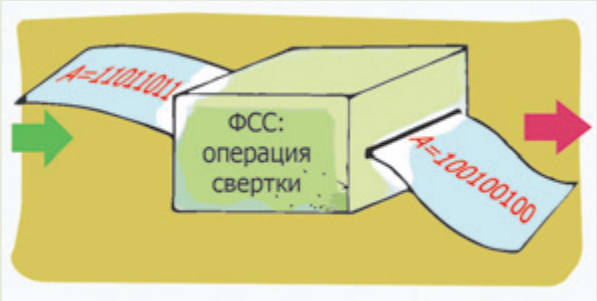
$$A = F_k + F_{k-1} + C,$$

где C — сумма остальных чисел Фибоначчи, соответствующих ненулевым разрядам записи числа. В соответствии с тождеством (1) мы можем написать

$$A = F_k + F_{k-1} + C = F_{k+1} + C.$$

А это и означает, что в записи числа стало на одну цифру больше, при этом старшая $(k + 1)$-я цифра равна 1, а k-я и $(k - 1)$-я цифры равны 0.

Замечание 2. Операция развертки **увеличивает** количество единиц в фибоначчиевой записи числа, но может уменьшить на единицу количество цифр в записи, если исходная запись начиналась с комбинации "100". Например, $\underline{100}0101_{fib} = \underline{11}0101_{fib}$.



Если над записью числа в ФСС выполнить все возможные *свертки*, то мы придем к специальному фибоначчиевому представлению числа, называемому *минимальной формой*, в которой нет двух рядом стоящих единиц. Если же в фибоначчиевой записи выполнить все возможные операции развертки, то придем к специальному фибоначчиевому изображению, называемому *максимальной*, или *развернутой, формой*, в которой рядом не встречается двух нулей. Любое число A представляется в минимальной или максимальной форме единственным способом.

Правило приведения фибоначчиевой записи числа к минимальной форме

1. Просматривая запись числа слева направо, найдем первую комбинацию “11”.
2. Выполним над ней операцию свертки:
   2.1. Если комбинация “11” стоит в начале записи числа, то количество цифр в записи числа увеличится на единицу. К записи числа слева припишется 1, а комбинация “11” заменится на “00”;
   2.2. Если самая левая комбинация “11” находится не в начале записи, это означает, что перед ней обязательно стоит 0. Тогда комбинация “011” заменяется на “100”.
3. Последовательно выполним операции свертки для всех комбинаций “11”, двигаясь по записи числа слева направо.
4. Если в получившейся записи есть две рядом стоящие единицы, то переход на п. 1, иначе получившаяся запись будет искомой минимальной формой.

Пример 3. Даны два числа: $A = 1111111_{fib}$ и $B = 1011101_{fib}$. Требуется привести их к минимальной форме.

Для приведения числа к минимальной форме выполним над ним все возможные операции свертки, двигаясь по записи числа слева направо:

$$A = \underline{11}111111_{fib} \Rightarrow 100\underline{11}1111_{fib} \Rightarrow$$

$$\Rightarrow 10100\underline{11}1_{fib} \Rightarrow 10101001_{fib}.$$

Единицы, над которыми выполняется операция свертки, подчеркнуты.

$B = 10\underline{11}101_{fib} \Rightarrow 1100101_{fib}$. После первого прохода по числу и выполнения операции свертки оказалось, что вновь полученная запись опять содержит комбинацию “11”. Следовательно, для полученной записи надо снова выполнить все возможные свертки:

$$B = \underline{11}00101_{fib} \Rightarrow 10000101_{fib}.$$

То есть $B = 10\underline{11}101_{fib} \Rightarrow \underline{11}00101_{fib} \Rightarrow 10000101_{fib}$.

Вопрос школьникам. Какое самое большое число в ФСС, представленное в минимальной форме, можно записать в k разрядах?

Очевидно, что самая старшая цифра должна быть 1. Далее для построения максимального числа цифры 0 и 1 должны чередоваться. Тогда максимальное число, которое можно записать в k разрядах, равно 10101…10 при четном k и 10101…01 при нечетном k. В табл. 2 приведены максимальные числа для двоичной системы и ФСС, которые можно записать в 8- и 16-разрядной ячейках.

4.3. Использование избыточности фибоначчиевой системы счисления. Базовые операции ФСС

При записи чисел в фибоначчиевой системе счисления принято использовать *минимальную форму записи*: в этой форме нет двух подряд идущих единиц. Тогда число 100 в фибоначчиевой системе счисления будет записано однозначным образом, а именно — $100_{10} = 1000010100_{fib}$. Так как цифрами фибоначчиевой системы счисления являются 0 и 1, то фибоначчиев код содержит только нули и единицы. Но тогда, если при передаче фибоначчиева кода, представленного в минимальной форме, появились две подряд идущих единицы, то можно однозначно сказать, что информация пришла с искажением (потерей).

“Естественная” избыточность кодов Фибоначчи, которая проявляется в “множественности” представлений одного и того же числа, может быть использована для целей контроля числовых преобразований в цифровых устройствах.

Анализ фибоначчиевой арифметики показал, что основными ее операциями являются операции *свертка*, *развертка* и основанная на них операция *приведения кода Фибоначчи к минимальной форме*.

Эти математические результаты стали основой для проектов создания компьютерных и измерительных систем на основе фибоначчиевой системы счисления.

Как известно, компьютерная программа реализуется с помощью команд процессора, сама программа и обрабатываемые данные хранятся в оперативной памяти. К сожалению, невозможно полностью исключить ошибки, возникающие в результате неисправной работы компьютерных элементов. Различают два типа неисправностей:

- *постоянный отказ* элемента (элемент, однажды отказавший, “выбывает из игры”);
- *случайный отказ* (элемент “отказывает” в случайные моменты времени, в то время как в другие моменты времени элемент работает корректно).

Последний тип отказов называется *сбоем*. Сбои в процессоре возникают под влиянием различных факторов, вызываемых внешними и внутренними шумами в компьютерных элементах и их электрон-

Таблица 2

	Двоичная система счисления	Фибоначчиева система счисления
8-разрядная ячейка	$2^8 - 1 = 255$	$10101010_{fib} = 54$
16-разрядная ячейка	$2^{16} - 1 = 65\,535$	$1010101010101010_{fib} = 2583$

ных цепях. Сбои в электронных элементах, в частности в триггерах, возникают значительно более часто, чем постоянные отказы.

Экспериментально доказано, что *сбои триггеров в режиме переключения являются наиболее вероятной причиной ненадежного функционирования процессоров*. Вот почему проектирование самоконтролирующихся цифровых автоматов, гарантирующих эффективный контроль сбоев триггеров, является одной из важнейших проблем проектирования высоконадежных процессоров.

Как уже говорилось ранее, в 80-х годах прошлого столетия группа ученых под руководством профессора Алексея Петровича Стахова из Таганрогского радиотехнического института пришла к идее создания процессора, позволяющего обнаруживать сбои триггеров, возникающих в момент их переключения. Такой процессор они называли *помехоустойчивым процессором*. Основная идея создания помехоустойчивого процессора состояла в следующем:

• вся обрабатываемая информация должна быть представлена в фибоначчиевой системе счисления;

• необходимо выбрать некоторый набор операций, называемых *базовыми микрооперациями*, на основе которых может быть реализован любой алгоритм обработки информации;

• ввести эффективную систему схемного контроля базовых микроопераций.

В качестве таких базовых операций были выбраны:

1) свертка;

2) развертка;

3) перемещение;

4) поглощение.

Первые две операции были рассмотрены нами выше.

Операция **перемещения** является поразрядной двуместной операцией $\mathbf{M}(A, B)$. Если ячейка A имеет двоичную цифру 1 в k-м разряде, а ячейка B в этом разряде имеет цифру 0, то в результате операции перемещения цифра 1 из ячейки A перемещается в ячейку B, а цифра 0 из ячейки B перемещается в ячейку A.

A	1		0
M		=	
B	0		1

Операция **поглощения** $\mathbf{P}(A, B)$ также является поразрядной двуместной операцией и состоит в том, что две двоичные единицы одного и того же разряда ячеек A и B взаимно уничтожаются, то есть заменяются нулями.

A	1		0
P		=	
B	1		0

5. Функциональная полнота базовых фибоначчиевых операций

Методическое отступление. В заключение знакомства с ФСС можно остановиться на достаточно сложном, но интересном материале — полноте базовых фибоначчиевых операций. Над фибоначчиевыми кодами можно выполнять специальные операции: свертки, развертки, перемещения и поглощения. Эти четыре операции составляют функционально полную систему, т.е. через эти операции можно выразить любую арифметическую или логическую операцию. Следовательно, для проектирования фибоначчи-компьютера достаточно уметь строить только четыре элемента, соответствующих операциям свертки, развертки, перемещения и поглощения.

Покажем, что любую логическую операцию можно выполнить через базовые фибоначчиевы операции [4]. Для этого достаточно показать, что с помощью этих операций можно реализовать конъюнкцию, дизъюнкцию и отрицание. А так как эта система операций $\{\wedge, \vee, \neg\}$ является полной [2], то в итоге мы сможем выразить через базовые операции любые другие.

Вопрос читателю. Что нам дает знание того, что некоторая система операций над двоичными последовательностями является полной?

Так как работа всего компьютера построена на логических преобразованиях двоичных последовательностей, то для проектирования компьютера достаточно уметь строить ограниченный набор логических элементов, каждый из которых реализует определенную операцию или функцию, входящую в некоторую полную систему. Известно, что современная цифровая техника строится на основе трех логических элементов — конъюнктора, дизъюнктора и инвертора. Но оказывается, что базовые операции $\{\mathbf{M}, \mathbf{P}\}$ вместе с константой 1 также образуют полную систему логических функций.

1) Покажем, что через базовые операции $\{\mathbf{M}, \mathbf{P}\}$ с использованием константы 1 можно выразить любую логическую операцию булевой логики. Напомним таблицы истинности для основных операций булевой логики — конъюнкции, дизъюнкции и отрицания.

Таблица 3

x	y	$x \wedge y$	$x \vee y$	$\neg x$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Операции перемещения $\mathbf{M}(A, B) = (A', B')$ и поглощения $\mathbf{P}(A, B) = (A'', B'')$ можно рассматривать как логические операции с двумя аргументами и двумя значениями. Составим таблицы истинности для этих операций.

Таблица 4

		$(A', B') = \mathbf{M}(A, B)$		$(A'', B'') = \mathbf{P}(A, B)$	
A	B	A'	B'	A''	B''
0	0	0	0	0	0
0	1	0	1	0	1
1	0	0	1	1	0
1	1	1	1	0	0

Сравнивая табл. 3 и 4, мы видим, что A' содержит результат поразрядной конъюнкции, т.е. $A' = A \wedge B$, а B' — результат поразрядной дизъюнкции, т.е. $B' = A \vee B$.

Логическая операция *отрицание* сводится к выполнению операции *поглощения* над исходной кодовой комбинацией A и константой 1. В результате операции поглощения $(A'', B'') = \mathbf{P}(A, B)$, где $B \equiv 1$, мы получаем $B'' = \overline{A}$ и $A'' \equiv 0$.

6. Выполнение операции сложения в ФСС

Выполнение операции сложения в позиционных системах счисления происходит по таблицам сложения. Для двоичной системы таблица сложения довольно проста.

Таблица 5

0 + 0 =	0
0 + 1 =	1
1 + 0 =	1
1 + 1 =	**10**

Последняя строка в табл. 5 задает правило формирования переноса при двоичном сложении единиц (1 + 1 = 10).

Построим таблицу сложения для ФСС. Если оба слагаемых имеют 1 в разряде с номером $k \geq 3$, то при сложении единица в k-м разряде сохраняется, и возникает перенос двух единиц в соседние младшие разряды: в $(k-1)$-й и $(k-2)$-й. Это следует из тождества:

$$F_k + F_k = F_k + F_{k-1} + F_{k-2}.$$

Для $k = 1$: 1 + 1 = **10** (выделен 1-й разряд).

Для $k = 2$: 1 + 1 = **10**1 (выделен 2-й разряд).

Таблицу сложения в ФСС можно записать в виде:

Таблица 6

	1-й разряд	**2-й разряд**	***k*-й разряд**
0 + 0 =	0	0	0
0 + 1 =	1	1	1
1 + 0 =	1	1	1
1 + 1 =	**10**	**10**1	**11**1

Возникающий при выполнении сложения перенос будем накапливать в отдельной ячейке R, которую назовем *многоразрядным переносом*. Для первого и второго разрядов многоразрядный перенос не формируется, переносимые единицы сразу записываются в формируемую сумму. Таким образом, не возникает ситуация, когда в одном из разрядов перенос состоит более чем из одной единицы.

Алгоритм сложения целых чисел в ФСС

Перед сложением суммируемые фибоначчиевые числа приводятся к минимальной форме. Поразрядное сложение выполняется справа налево.

Будем использовать следующие обозначения:

S — промежуточная сумма;

R — многоразрядный перенос.

1. Выполним поразрядное сложение чисел в соответствии с табл. 6.
2. Промежуточная сумма S приводится к минимальной форме.
3. Если полученный многоразрядный перенос R равен 0, то переход на п. 6.
4. Выполняется сложение $S + R$, в результате которого формируется новое значение S и R.
5. Переход на п. 2.
6. Промежуточная сумма S, приведенная к минимальной форме, и есть результат сложения. Конец алгоритма.

Пример 4. Требуется сложить два числа $a = 10101_{fib}$ и $b = 1001_{fib}$ в ФСС.

В нашем примере числа уже записаны в минимальной форме, две единицы стоят только в первом разряде, поэтому многоразрядный перенос равен 0. В соответствии с табл. 6 получим:

```
  1 0 1 0 1
+   1 0 0 1
S = 1 1 1 1 0
```

Результат приведем к минимальной форме, для этого дважды выполним операцию свертки:

$$\underline{11}110 \Rightarrow 100\underline{11}0 \Rightarrow 101000.$$

Проверим себя, переведем исходные слагаемые и полученный результат в десятичную систему счисления: $a = {}^{8\,5\,3\,2\,1}10101 = 12$; $b = {}^{5\,3\,2\,1}1001 = 6$, $S = {}^{13\,8\,5\,3\,2\,1}101000 = 18$.

Пример 5. Сложим десятичные числа $A = 20_{10}$ и $B = 20_{10}$ в фибоначчиевой системе счисления. Сначала запишем числа в ФСС в минимальной форме: $A = B = 20_{10} = 101010_{fib}$.

Операция сложения будет выполняться в несколько шагов. В этом примере две единицы стоят во втором, четвертом и шестом разрядах, но при сложении на первом шаге перенос будет возникать только для четвертого и шестого разрядов (см. табл. 6).

1)
```
  101010
 +101010
S=101101
R=011110
```
Получили промежуточную сумму $S = 101101$ и многоразрядный перенос $R = 011110$.

2) Приведем S к минимальной форме:

$$101101 \Rightarrow 110001 \Rightarrow 1000001.$$

3) Вычислим $S + R$:
```
 1000001
+  11110
S=1011111
```
Получили промежуточную сумму 1011111 и нулевой многоразрядный перенос.

4) Запишем S в минимальной форме:

$$\underline{10}11111 \Rightarrow 1100\underline{111} \Rightarrow \underline{11}01001 \Rightarrow 10001001.$$

Проверим себя: ${}^{34\,21\,13\,8\,5\,3\,2\,1}10001001 = 34 + 5 + 1 = 40$.

Алгоритм фибоначчиева сложения кажется сложным и утомительным, но после приобретения некоторого опыта эта процедура оказывается даже занятной.

6.1. Реализация операции сложения в ФСС через базовые операции

Покажем, что через базовые операции {**S**, **R**, **M**, **P**} можно реализовать фибоначчиево сложение ячеек A и B. В основе этой реализации лежит идея перебрасывания единиц из k-го разряда ячейки A в k-й разряд ячейки B, если там был 0. При этом на каждом шаге сумма модифицированных ячеек остается прежней. Алгоритм заканчивается, когда ячейка A становится равной 0, в этом случае искомая сумма накоплена в ячейке B.

В качестве примера просуммируем два числа: $A_0 = 10100100 = 50_{10}$ и $B_0 = 01010100 = 32_{10}$.

1-й шаг	Выполним операцию перемещения: $(A_1, B_1) = \mathbf{M}(A_0, B_0)$	A_0 = 0 1 0 1 0 0 1 0 0 ↓↑ ↓↑ B_0 = 0 0 1 0 1 0 1 0 0 A_1 = 0 0 0 0 0 0 1 0 0 B_1 = 0 1 1 1 1 0 1 0 0
2-й шаг	Выполним все возможные операции развертки над A_1: $A_2 = \mathbf{R}(A_1)$	A_2 = 0 0 0 0 0 0 0 1 1
3-й шаг	Выполним все возможные свертки над B_1: $B_2 = \mathbf{S}(B_1)$	B_2 = 1 0 1 0 0 0 1 0 0
4-й шаг	Выполним операцию перемещения: $(A_3, B_3) = \mathbf{M}(A_2, B_2)$	A_2 = 0 0 0 0 0 0 0 1 1 ↓↑ ↓↑ B_2 = 1 0 1 0 0 0 1 0 0 A_3 = 0 0 0 0 0 0 0 0 0 B_3 = 1 0 1 0 0 0 1 1 1
5-й шаг	Приведем B_3 к минимальной форме, выполнив операцию свертки: $B_4 = \mathbf{S}(B_3)$	B_4 = 1 0 1 0 0 1 0 0 1

Проверим себя: $B_4 = 101001001_{fib} = 55 + 21 + 5 + 1 = 82_{10}$.

Можно показать, что операции вычитания, умножения и деления также реализуются через базовые операции. Таким образом, через базовые операции {**S**, **R**, **M**, **P**} можно выразить любую логическую и любую арифметическую операцию. И, следовательно, мы можем проектировать Фибоначчи-процессор на основе базовых микроопераций.

Методическое отступление. Какое преимущество будет иметь такой фибоначчи-процессор в сравнении с классическими компьютерами, основанными на классической двоичной системе счисления?

Ответ на этот вопрос состоит в том, что такой процессор обладает возможностью контроля ошибок, которые могут возникнуть в процессоре под влиянием различных внешних и внутренних факторов [5].

К сожалению, создать фибоначчи-компьютер по разным причинам пока так и не удалось: группа А.П. Стахова спроектировала только экспериментальный Ф-процессор, который действительно сам определял, произошел ли сбой при работе процессора. При разработке элементной базы Ф-процессора основным операционным элементом стало устройство приведения кода Фибоначчи к минимальной форме. Это устройство реализовывалось через RS-триггеры и логические элементы И и ИЛИ.

Теоретические основы данного направления представляют несомненный интерес и могут стать источником новых идей не только в компьютерной области. Современные математики проявляют большой интерес к “фибоначчиеву” направлению. В 1963 г. группа американских математиков, возглавляемая Вернером Хоггаттом, организовала математическую фибоначчи-ассоциацию, которая выпускает журнал “The Fibonacci Quarterly” и ежегодно с 1984 г. проводит международную конференцию “Fibonacci Numbers and Their Applications”.

Литература

1. *Андреева Е., Фалина И.* Системы счисления и компьютерная арифметика. М.: БИНОМ. Лаборатория знаний, 2004.
2. *Андреева Е., Босова Л., Фалина И.* Математические основы информатики. М.: БИНОМ. Лаборатория знаний, 2007.
3. Помехоустойчивые коды: Фибоначчи-компьютер, сб. статей. Серия “Радиоэлектроника и связь”. М.: Знание, 1989.
4. *Стахов А.П.* Введение в алгоритмическую теорию измерений. М.: Сов. радио, 1977.
5. *Стахов А.П.* Коды золотой пропорции. М., 1984.
6. *Фалина И.Н.* Компьютер и фибоначчиева система счисления. М.: Потенциал, № 6, 2011.
7. *Фалина И.Н.* Компьютерная фибоначчиева арифметика. М.: Потенциал, № 8, 2011.

1 НОЯБРЯ

Учительская фестиваль книга Москва-2011

Предметы естественно-научного цикла

География ● Биология ● Химия ● Физика ● Математика ● Информатика

БОЛЕЕ 1000 НАИМЕНОВАНИЙ КНИГ ДЛЯ УЧИТЕЛЯ ПРЕДСТАВЛЯЮТ

«Айрис-пресс», «АСТ-Пресс», «Бином», «ВАКО», «Илекса», «Интеллект-Центр», «Легион», «Мнемозина», «Новый Диск», «Первое сентября», «Потенциал», «СГУ», «СМИО Пресс», «1С», «Учитель», «Экзамен»

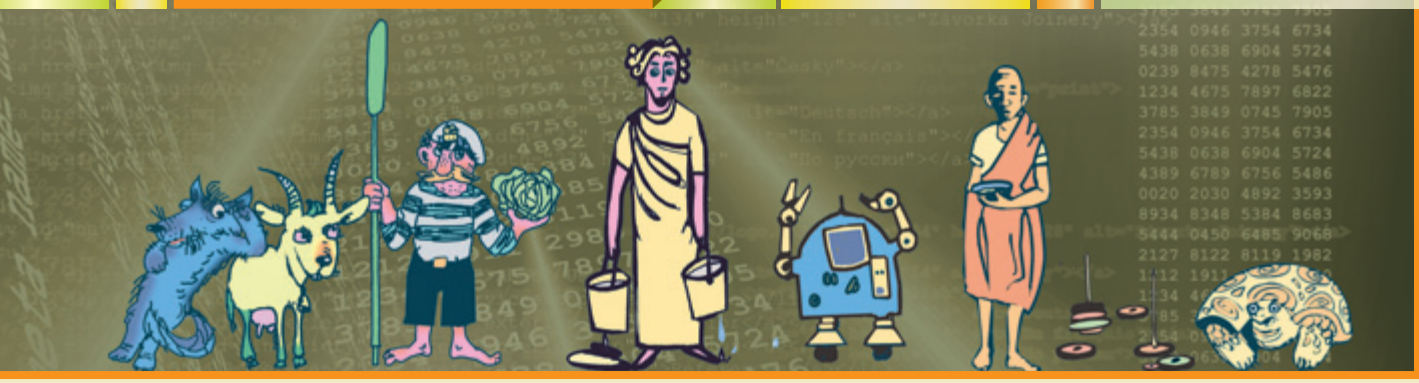
программа дня

10.00 – 11.15	Издательство «Айрис-пресс**»** **9-й класс. ГИА – комплексная подготовка по математике** *В.И. Глизбург,* д.п.н., к.физ.-мат.н., профессор института педагогики и психологии образования ГОУ ВПО МГПУ	Издательство «СМИО Пресс» **Снова о повторении. Достижение успеха в обучении математике. Поурочные разработки** *С.Е. Злотин,* к.тех.н., соросовский учитель, 4-кратный лауреат фонда «Династия»	Издательство «Экзамен» **ЕГЭ 2012. Обучение решению задач группы С. Новые возможности** *И.Н. Сергеев,* д.ф.-м.н., профессор МГУ
11.30 – 12.45	Издательство «Легион» **Организация мониторинга учебных достижений школьников в освоении нового образовательного стандарта по математике с использованием пособий издательства «Легион»** *С.Ю. Кулабухов,* к.физ.-мат.н., начальник отдела математики издательства «Легион», автор пособий по математике	Компания «Бином. Лаборатория знаний» Лекция	Компания «Экзамен-Медиа» **Инновационные, электронные учебно-наглядные пособия по физике в рамках введения новых стандартов образования и реализации национальной образовательной программы «Наша новая школа»** *А.А. Кудрявцев,* преподаватель математики, физики и информатики, разработчик электронных учебных пособий («Экзамен-Медиа»)
13.00 – 14.15	Издательство «Мнемозина» **Реализация новых подходов в методике преподавания профильного курса биологии** *А.В. Теремов,* д.п.н., профессор кафедры методики обучения биологии МПГУ	*расписание уточняется*	Компания «Экзамен-Медиа» **Инновационные, электронные учебно-наглядные пособия по биологии в рамках введения новых стандартов образования и реализации национальной образовательной программы «Наша новая школа»** *В.Л. Шалов,* старший преподаватель кафедры ИКТ ГОУ Педагогической академии («Экзамен-Медиа»)
14.30 – 15.45	*расписание уточняется*	*расписание уточняется*	*расписание уточняется*

Начало работы в 9.00. В расписании возможны изменения и дополнения.
Следите за изменениями в расписании на сайте **http://bookfair.1september.ru**

ВХОД СВОБОДНЫЙ, но чтобы получить профессиональные подарки,
пройдите заранее бесплатную регистрацию на сайте **http://bookfair.1september.ru**

Все мероприятия фестиваля пройдут в московском государственном лицее № 1535 по адресу: ул. Усачева, дом 52 (в 3 минутах ходьбы от станции метро «Спортивная»). Телефон (499) 249-31-38. *Внимание! В лицее нет камеры хранения. Спасибо за понимание.*



ИСТОРИЯ ИНФОРМАТИКИ

Это древнее слово — “компьютер”

В.В. Шилов, Москва

Человек придумал цифры для того, чтобы считать и вычислять. Но на протяжении многих веков потребность в вычислениях ограничивалась в основном **подсчетом** — т.е. определением числа единиц какого-либо объекта, будь то количество овец в стаде или монет в казне. Вероятно, единственными людьми, которые на самом деле покрывали бумагу (точнее, поскольку бумагу еще не изобрели, свитки папируса, куски пергамента или восковые дощечки) рядами цифр, были астрономы древнего мира. А в середине первого тысячелетия нашей эры — после распространения христианства в Европе — одной из самых важных задач астрономии стало вычисление точной даты празднования Пасхи.

Искусство ее вычисления столетиями оставалось центральной проблемой математики и астрономии. Оно получило название *компут* (*computus*, от лат. *computare* – “вычислять”). Люди, занимавшиеся компутом, назывались… Нет, не *компутерами*, а *компутистами*.

Латынь в Средние века была широко распространенным языком, на котором разговаривали придворные и сочинялись ученые трактаты. Постепенно слово *comput*, видоизменяясь, проникало и в другие европейские языки, в том числе в английский. В 1646 году известный английский врач и ученый-энциклопедист Томас Браун в сочинении “Pseudodoxia epidemica” впервые использовал слово *computer* для обозначения человека, профессионально занимающегося пасхальными вычислениями (таким образом, оно заменило латинское *computiste*). Так что можно считать, что слову “компьютер” в 2011 году исполняется 365 лет.

Слово прижилось, поскольку оно отвечало потребностям времени. Действительно, по сравнению с античностью жизнь общества неимоверно усложнилась. Бурно развивались торговля, промышленность, мореплавание, картография, физика и другие науки. Везде надо было проводить вычисления, но, к сожалению, искусство счета не было общим достоянием. Хотя в школах и университетах обучали вести вычисления на бумаге с использованием арабских цифр, но в основном ограничивались умением складывать и вычитать. Умножение же и, особенно, деление были известны (и доступны) далеко не каждому… В подтверждение можно привести, например, такой факт. Известный английский мемуарист Сэмюэл Пепис, начавший работать клерком Адмиралтейства, 4 июля 1662 года записал в своем дневнике: “…я пытаюсь изучить математику (прежде всего стараюсь выучить таблицу умножения)”. То, что Пепис, окончивший Кембриджский университет, не умел ни умножать, ни делить, ярко характеризует уровень математического образования эпохи.

Поэтому во многих европейских странах начиная с XV века сформировалось сословие **мастеров счета**. Были они (правда, несколько позже) и в России. Вспомним бессмертную комедию Д.И. Фонвизина “Недоросль” (1782 год), один из персонажей которой, отставной солдат Цыфиркин, так говорит о себе: “Малу толику арихметике маракую, так питаюсь … у счетных дел. Не всякому открыл Господь науку: так кто сам не смыслит, меня нанимает то счетец поверить, то итоги подвести. Тем и питаюсь: праздно жить не люблю”[1].

До середины XVIII века мастера счета, подобно Цыфиркину, в основном перебивались случайными заказами. Но в 1767 году в Англии вышел первый выпуск ставшего впоследствии ежегодным “Морского альманаха” (“Nautical Almanac”). Основным предназначением издания было помочь морякам определять

[1] Несколько отвлекаясь, заметим, что можно также считать, что в пьесе впервые описано использование двоичной системы счисления. Другой ее герой, Митрофан (Митрофанушка), изучая математику, выяснял, что “единожды нуль — нуль”, а “нуль да нуль — нуль”, что совпадает с действиями в двоичной системе. Еще он пытался определить, сколько будет “Один да один”, правда, при этом задумался… ☺. — **Прим. ред.**

долготу в открытом море, поэтому альманах содержал различные астрономические таблицы (в первую очередь лунные расстояния с интервалом в три часа на каждые сутки года). Для расчета таблиц по всей стране нанимались квалифицированные вычислители, выполнявшие свою работу на дому, причем каждый из них производил полный набор расчетов на определенный период времени. Из сохранившихся списков следует, что первой женщиной-компьютером, привлеченной для выполнения счетных работ, стала некая Мэри Эдвардс. Организатором работ и редактором альманаха был королевский астроном Невил Маскелин.

После этого различные схемы организации вычислительных работ с помощью коллектива людей применялись во многих странах в течение почти 200 лет. История знает такие масштабные проекты использования людей-компьютеров, как предложенный в 1922 году кембриджским математиком Льюисом Ричардсоном план задействовать одновременно 64 000 (!) счетчиков в разных странах для расчета прогноза погоды. Компьютеры вели свои расчеты на бумаге, а ближе к концу XIX века впервые стали использовать арифмометры. Для того чтобы отличить "счетные машины" от "счетных людей", различные устройства для механизации счета стали называть *калькуляторами* (*calculator*) или *вычислительными машинами* (*computing machine*).

Работа за комптометром (суммирующая машина). 1914 год

Своего расцвета профессия компьютера достигла в 1930–1940-е годы. В это время была особенно велика потребность в различных таблицах — математических, астрономических, баллистических, навигационных и других. Для их составления требовалось производить огромный объем вычислений, и количество людей, вовлеченных в эту деятельность, исчислялось тысячами. В штатном расписании научных организаций и фирм значились такие должности, как "младший компьютер", "помощник компьютера", "компьютер" и "старший компьютер". Например, известный американский ученый в области вычислительной техники Герберт Грош, начавший свой трудовой путь в 1935 году именно с должности компьютера, в последние годы жизни в шутку называл себя "самым старым компьютером на Земле".

Компьютеры за работой. Проект "Математические таблицы" (США, 1938 г.)

Когда в середине 1940-х годов стали появляться первые электронные вычислительные машины, их воспринимали всего лишь как большие и очень быстрые арифмометры, способные с огромной скоростью производить вычисления. Они должны были заменить на этой работе людей-компьютеров и, естественно, также получили это наименование. Чтобы избежать терминологической путаницы, в 1945 году знаменитый компьютерный пионер, создатель серии релейных электромеханических машин Джордж Стибиц предложил сохранить названия *калькулятор* и *вычислительная машина* за простыми механизмами, способными выполнять за один раз одну из четырех арифметических операций над парой чисел, а словом *компьютер* называть машины, "способные автоматически выполнять последовательности таких операций и сохранять промежуточные результаты". Людей, которые работают на таких машинах, Стибиц предложил называть *операторами* (*operator*).

Действительно, терминология еще только формировалась. Первая механическая вычислительная машина *Mark-I*, построенная Говардом Айкеном в 1944 году, официально называлась *ASCC* (Automatic Sequence Controlled Calculator) — т.е. калькулятор, а не компьютер. Название знаменитого *ENIAC* (первая электронная вычислительная машина, построенная в США в 1946 году) было аббревиатурой слов Electronic Numerical Integrator and Computer. Первая в Европе электронная вычислительная машина, построенная в Англии в 1949 году, называлась *EDSAC* (Electronic Delay Storage Automatic Calculator). А вот великий английский математик Алан Тьюринг свой проект вычислительной машины назвал *ACE* (Automatic Computing Engine). Слово *engine* — "устройство, машина" — явно отсылало к знаменитой Аналитической машине (*analytical*

engine) его не менее великого соотечественника Чарльза Бэббиджа. Таким образом, слово *computer* далеко не сразу прижилось даже в США и Англии. Окончательно это произошло только в первой половине 1950-х годов.

Тем не менее в итоге действительно утвердились наименования, предложенные Джорджем Стибицем. Например, когда на смену механическим арифмометрам пришли первые электромеханические (в начале 1950-х годов), а затем электронные настольные вычислительные устройства (в начале 1960-х), они также стали называться калькуляторами. Это слово используется и сегодня, хотя нынешние калькуляторы стали уже совершенно иными.

Еще более долгим оказался путь к признанию слова в других странах. В СССР с начала 1950-х годов поначалу утвердилось название *электронная счетная машина* (об этом напоминают названия первых советских машин — *МЭСМ* и *БЭСМ*[2]), однако вскоре ему на смену пришло другое — *электронная вычислительная машина* (ЭВМ). Калькулятор по-немецки — *Rechenmaschine* (вычислительная машина). Поэтому для обозначения электронных вычислительных машин в Германии использовали термины *Elektronenrechner*, *Rechenautomat* или просто *Rechner*. Во Франции компьютер долго называли *calculateur*, а в Италии — *calcolatore*, причем оба этих слова первоначально обозначали людей, производящих вычисления. И только в последние три десятилетия слово *компьютер* во всех языках практически вытеснило все синонимы.

СЕМИНАР

Выбрать кошку помогают… реле

В статье [1] была получена электрическая схема с переключателями, с помощью которой можно выбрать пушистое домашнее животное, соответствующее условиям: “кот, сиамский, белый или рыжий; или кошка, сиамская, любого цвета, кроме белого; или любой/любая кот/кошка черного окраса”:

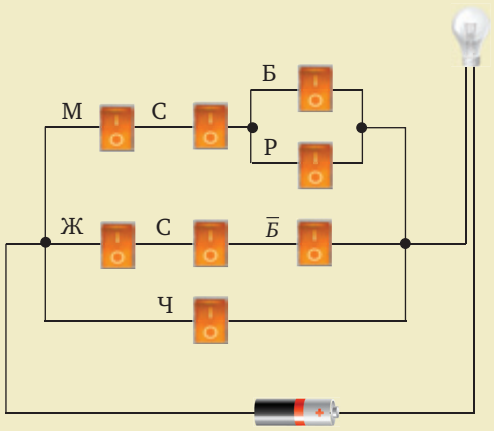


Рис. 1

Напомним, что логическое выражение, которое описывает свойства нужного нам домашнего животного, выглядит так:

$$\mathbf{М} \times \mathbf{С} \times (\mathbf{Б} + \mathbf{Р}) + \mathbf{Ж} \times \mathbf{С} \times (1 - \mathbf{Б}) + \mathbf{Ч}$$

— где **М** и **Ж** — множества всех кошек, соответственно, мужского и женского пола; **Б**, **Р** и **Ч** — множества всех кошек, соответственно, белого, рыжего и черного цветов; **С** — множество кошек сиамской породы; знак “+” обозначает слово *ИЛИ*, знак “×” — слово *И*; запись “1 –” соответствует слову *НЕ* (в смысле *все, за исключением чего-то*).

Если вы, уважаемый читатель, изучали основы логики, то наверняка знаете, что в логике, как и в математике, имеется так называемый “коммутативный закон умножения”, который позволяет изменить порядок переменных, связанных знаком “×”, и переписать выражение в виде:

$$\mathbf{С} \times \mathbf{М} \times (\mathbf{Б} + \mathbf{Р}) + \mathbf{С} \times \mathbf{Ж} \times (1 - \mathbf{Б}) + \mathbf{Ч}$$

Введем две вспомогательные переменные — **Х** и **У**:

$$\mathbf{Х} = \mathbf{М} \times (\mathbf{Б} + \mathbf{Р})$$
$$\mathbf{У} = \mathbf{Ж} \times (1 - \mathbf{Б})$$

С их использованием выражение для “кошачьих свойств” запишется так:

$$(\mathbf{С} \times \mathbf{Х}) + (\mathbf{С} \times \mathbf{У}) + \mathbf{Ч}$$

Видно, что переменная **С** появляется в выражении дважды. Применив еще один закон логики — дистрибутивный закон умножения — относительно суммы, перепишем выражение так, чтобы **С** встречалась в нем только один раз:

$$\mathbf{С} \times (\mathbf{Х} + \mathbf{У}) + \mathbf{Ч}$$

А теперь опять подставим вместо X и Y их значения:

$$\mathbf{С} \times ((\mathbf{М} \times (\mathbf{Б} + \mathbf{Р})) + \mathbf{Ж} \times (1 - \mathbf{Б})) + \mathbf{Ч}$$

В полученном выражении так много скобок, что на первый взгляд его трудно назвать упрощенным. И все же в нем переменных на одну меньше (их теперь не 8, а 7), чем в первоначальном варианте, а значит, и одним переключателем в схеме меньше. Вот как выглядит измененная версия схемы:

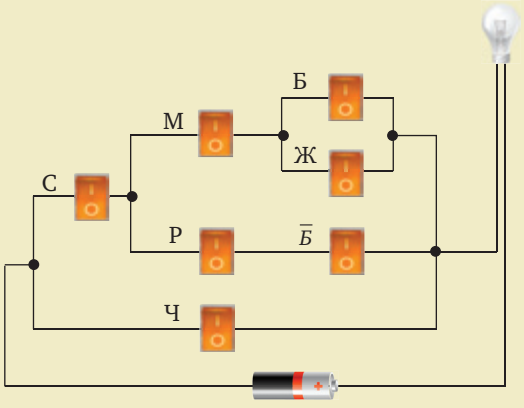


Рис. 2

[2] БЭСМ — сокращение от “**б**ольшая (или **б**ыстродействующая) **э**лектронная **с**четная **м**ашина”; МЭСМ — от “**м**алая **э**лектронная **с**четная **м**ашина”.

А ведь из этой схемы можно убрать еще три переключателя. Теоретически для задания параметров кошки хватит четырех переключателей. Почему четырех? По сути, переключатель — это 1 бит. Один переключатель позволяет задавать пол животного (выключен — мужской, включен — женский), другой служил бы признаком сиамской породы (включен — сиамская, выключен — нет), с помощью оставшихся двух можно указывать цвет. Выбирая кошку, вы задаете четыре признака: белый, черный, рыжий и другой. Известно, что четыре значения можно задать двумя битами (значения: 00, 01, 10 и 11), поэтому для выбора цвета нужно всего два переключателя. Скажем, два переключателя разомкнуты — цвет белый, замкнут только первый переключатель — черный, замкнут только второй — рыжий, оба переключателя замкнуты — все остальные цвета.

Теперь создадим пульт прибора для выбора кошки. На его пульте размещены четыре тумблера и лампочка:

Рис. 3

Будем считать, что если тумблер вверху — переключатель замкнут, внизу — разомкнут. Левый тумблер в паре, "управляющей" цветом, обозначен **Ч**. Это означает, что когда он включен (как показано на рисунке), выбран черный цвет. Правый тумблер в паре обозначен **Р**, и его включение означает, что выбран рыжий окрас. Если оба тумблера вверху, выбран "другой цвет", что символизирует буква **Д**. Если оба тумблера выключены (внизу), выбран белый цвет (буква **Б**).

На компьютерном языке этот набор тумблеров называется *устройством ввода* информации, управляющим поведением всей схемы. У нас тумблеры позволяют ввести 4 бита информации, которые описывают кошку или кота. Электрическая лампочка — *устройство вывода* — загорается, если параметры кошки, заданные тумблерами, соответствуют вашим требованиям. Положение тумблеров на *рис.* 3 соответствует выбору черной не сиамской кошки (проверьте!). Она удовлетворяет нужным критериям, и потому лампочка горит.

Теперь осталось разработать электрическую схему, которая обеспечивала бы работу этого пульта. В этом нам помогут специальные устройства — реле. "А что это такое?" — спросите вы.

Если взять железный стержень, намотать на него пару сотен витков тонкого провода, а затем пропустить по проводу ток, то железный стержень станет магнитом и будет притягивать кусочки железа и стали (провод в электромагните должен быть именно тонким, чтобы обладать заметным сопротивлением и не приводить к короткому замыканию). Если ток выключить, то железный стержень потеряет свои магнитные свойства.

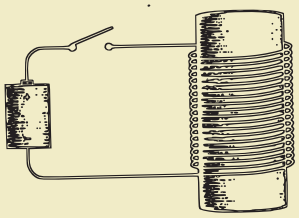


Рис. 4

Эта особенность такого стержня — электромагнита и положена в основу реле. В реле входящий ток приводит в действие электромагнит, который втягивает металлический стержень. Стержень, в свою очередь, используется как часть переключателя, соединяющего батарейку с выходящим проводом:

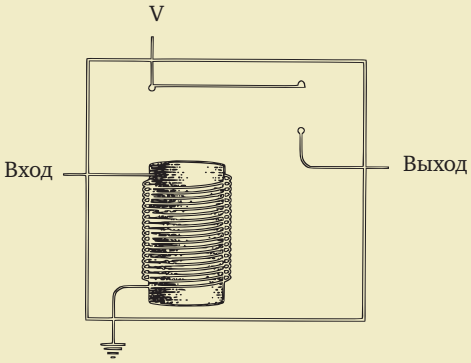


Рис. 5

Несмотря на простоту реле, на их основе были разработаны достаточно сложные устройства. Были созданы даже релейные вычислительные машины. Но вернемся, как говорится, "к нашим баранам", точнее — к кошкам ☺.

Соединим реле с переключателем, парой батареек и лампочкой:

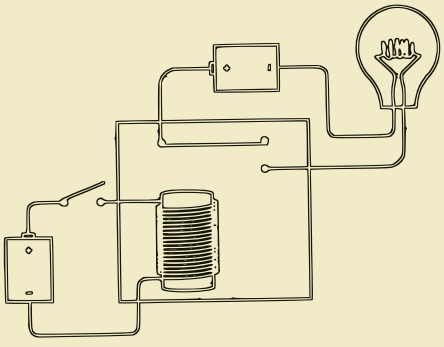


Рис. 6

Переключатель слева разомкнут, и лампочка не горит. Если замкнуть этот переключатель, ток из батареи пойдет по виткам катушки, намотанной на железный сердечник. Сердечник станет магни-

том, притянет гибкую металлическую полоску, она замкнет цепь и включит лампочку:

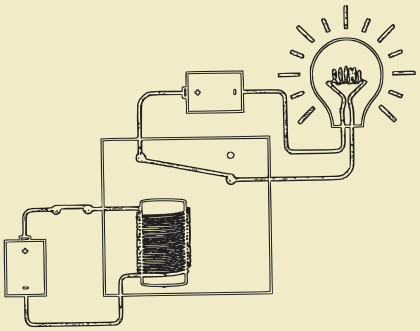


Рис. 7

Если полоска притянута электромагнитом, говорят, что реле *сработало* (защелкнулось).

Когда переключатель размыкается, железный сердечник перестает быть магнитом, и полоска возвращается в первоначальное (незамкнутое) состояние.

Не слишком ли замысловатый способ мы избрали, чтобы включить свет? Конечно, если бы мы были заинтересованы только в этом, то обошлись бы и без реле, но наши притязания включением света не ограничиваются. Мы изготовим не только устройство, с помощью которого можно выбрать кошку, соответствующую тем или иным условиям, но и вычислительное устройство! Так что не будем бояться “сложностей”. Тем не менее давайте немного упростим схему.

Исключим некоторые провода с помощью так называемой “земли” (так электрики называют общий провод, соединенный с “минусовым” контактом источника тока):

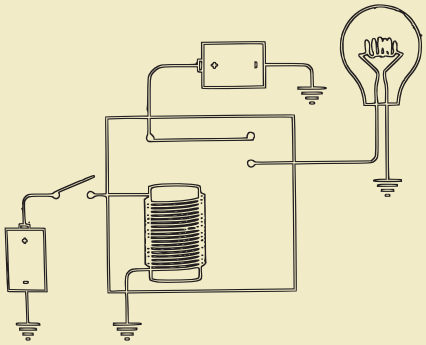


Рис. 8

Конечно, схема не слишком упростилась, но мы еще не закончили. Обратите внимание: отрицательные контакты обеих батареек соединены с землей. Поэтому везде, где мы увидим нечто подобное:

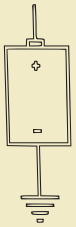


Рис. 9

— заменим это изображение заглавной латинской буквой V. Теперь схема выглядит так:

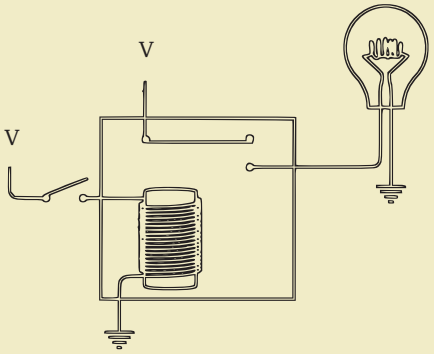


Рис. 10

Когда переключатель замкнут, ток между источником питания и землей протекает через катушку электромагнита, электромагнит притягивает металлическую полоску, и та замыкает цепь между питанием, лампочкой и “землей”. Лампочка загорается (*рис*. 11).

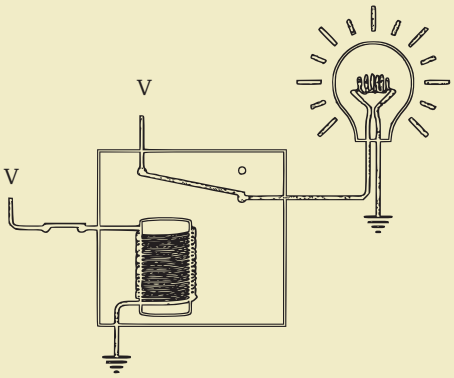


Рис. 11

На этой схеме два источника питания и две “земли”, но на схемах все “земли” можно соединить и все источники питания тоже можно соединить друг с другом. Во всех релейных схемах достаточно одного (достаточно мощного) источника питания. Например, схему на *рис*. 11 можно перерисовать с использованием только одной батареи:

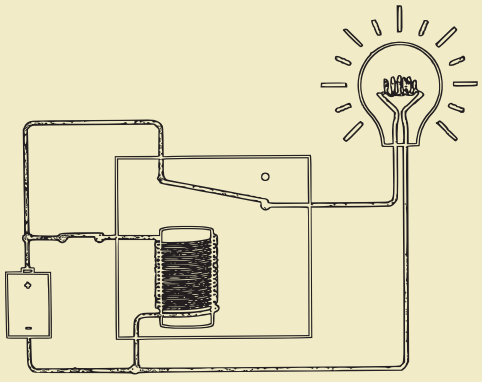


Рис. 12

Но для наших целей эта схема не слишком годится. Лучше избегать замкнутых цепей и описывать работу реле с понятий *входного* и *выходного* сигналов.

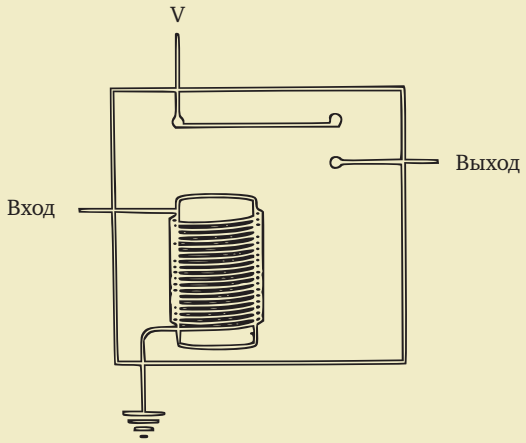


Рис. 13

Если на вход поступает ток (например, если он с помощью переключателя подключен к источнику питания), электромагнит срабатывает, и на выходе появляется напряжение.

На входе не обязательно должен стоять переключатель, а на выходе — лампочка. Выход одного реле может подключаться к входу другого реле, например, так, как показано на *рис*. 14:

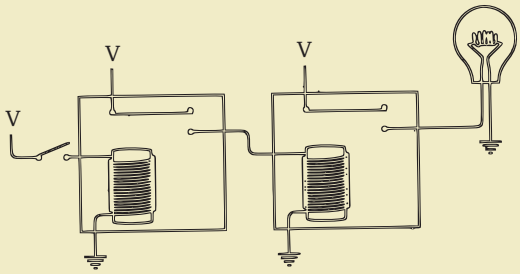


Рис. 14

Когда вы замыкаете переключатель, первое реле срабатывает и подает питание на второе реле. Оно тоже сработает, и включится свет:

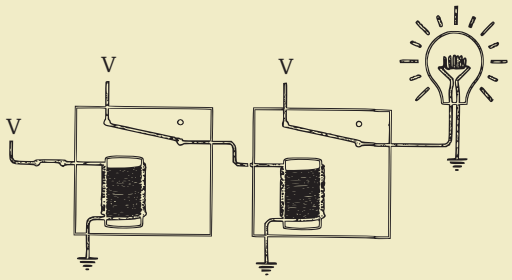


Рис. 15

Соединение нескольких реле лежит в основе построения так называемых “логических вентилей”[3].

В действительности лампочку можно соединить с реле двумя способами. Видите гибкую металлическую полоску, которая притягивается электромагнитом? В состоянии покоя она касается верхнего контакта, а когда электромагнит ее притягивает — нижнего. До сих пор мы использовали в качестве выхода реле нижний контакт, но в этом качестве можно использовать и верхний. В последнем случае лампочка горит, когда переключатель разомкнут (*рис*. 16).

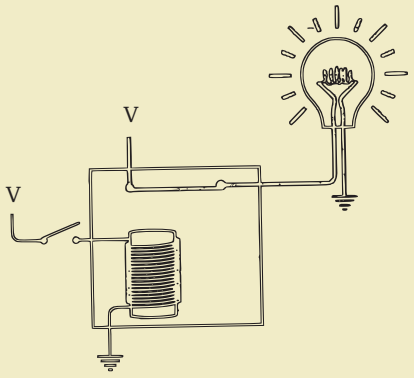


Рис. 16

Когда входной переключатель замкнут, лампочка гаснет:

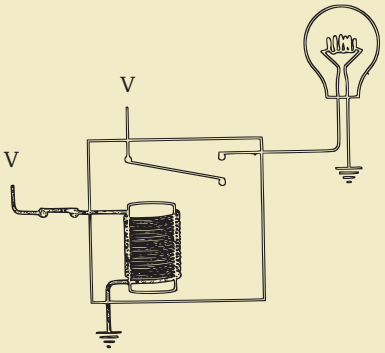


Рис. 17

Такое реле называется *двухпозиционным*. У него два электрически противоположных выхода: когда на одном есть напряжение, на другом его нет.

Подобно переключателям, реле можно соединить последовательно:

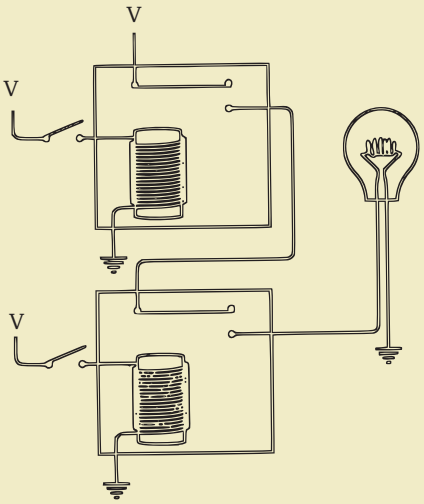


Рис. 18

Выход верхнего реле подает питание на нижнее. Как видите, если оба переключателя разомкнуты, лампочка не горит. Попробуем замкнуть верхний переключатель:

[3] В будущем, когда история вычислительных устройств XX века начнет постепенно забываться, кто-нибудь наверняка предположит, что выражение “логический вентиль” происходит от одноименного сантехнического приспособления. Однако это не совсем так, хотя, как будет показано далее, логические вентили и впрямь похожи на обычные водопроводные краны: они выполняют простые логические операции, останавливая или пропуская через себя электрический ток (как сантехнические вентили — воду).

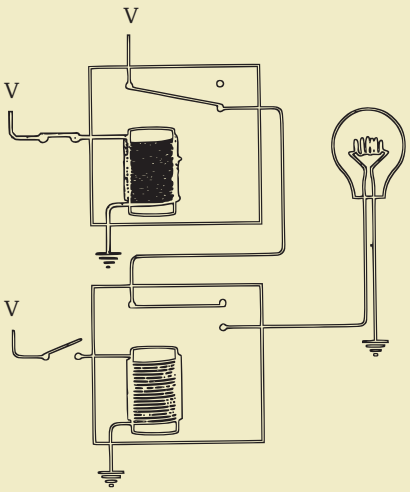


Рис. 19

Лампочка не включилась, так как нижний переключатель разомкнут, нижнее реле не сработало. Теперь разомкнем верхний переключатель и замкнем нижний:

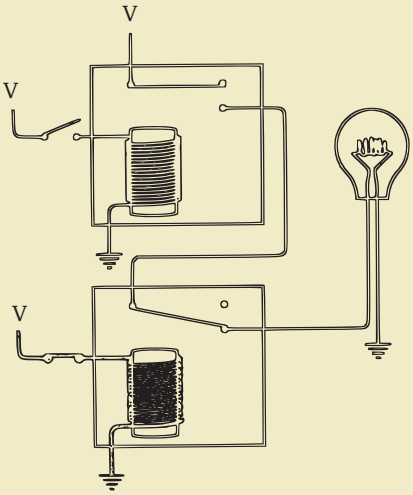


Рис. 20

Все равно не горит. Ток не доходит до лампочки, так как не сработало первое реле. Единственный способ заставить светиться лампочку — замкнуть оба переключателя (*рис*. 21).

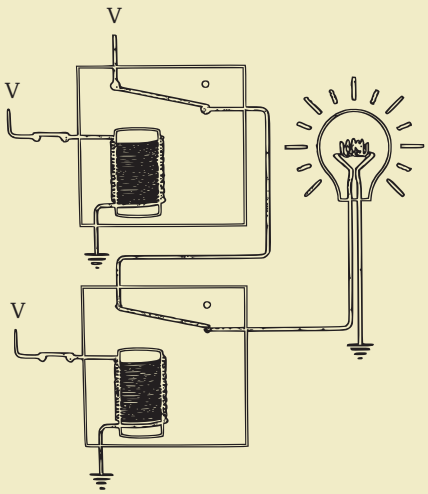


Рис. 21

Теперь сработали оба реле, и ток течет между источником питания, лампочкой и “землей”.

Подобно двум переключателям, соединенным последовательно, эти два реле выполняют простую логическую функцию. Лампочка горит, если сработали оба реле. Эта схема называется *вентилем “И” (AND)*. Символически он изображается так:

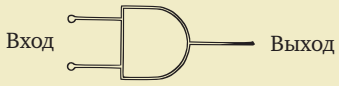


Рис. 22

Это первый из четырех основных логических вентилей. У вентиля “И” два входа (на схеме слева) и один выход (на схеме справа). Вентиль “И” часто рисуют именно так: со входами слева и с выходом справа. Да оно и понятно: людям, привыкшим читать слева направо, и электрические схемы удобно рассматривать слева направо. Но вентиль “И” с тем же успехом можно рисовать и со входами вверху, справа или внизу.

Первоначально схема с двумя последовательно соединенными реле, двумя переключателями и лампочкой выглядела так, как было показано на *рис*. 18. Использовав символическое обозначение вентиля “И”, ту же схему можно изобразить в виде:

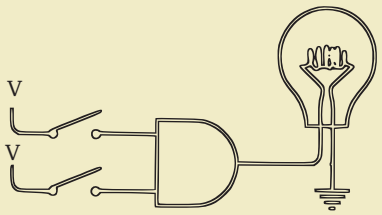


Рис. 23

Символ вентиля “И” не только заменяет два последовательно соединенных реле, но и подразумевает, что верхнее реле соединено с источником питания и оба соединены с землей. Еще раз: лампочка загорается, только если замкнуты верхний и нижний переключатели. Поэтому схема и называется вентилем “И”.

Входы вентиля “И” не обязательно должны соединяться с переключателями, а выход не обязательно должен быть подключен к лампочке. По сути, мы имеем дело только с напряжениями на входе и на выходе. Например, выход одного вентиля И может быть входом второго вентиля И:

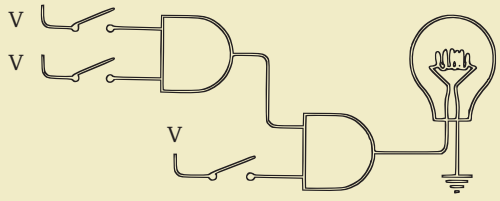


Рис. 24

Лампочка на схеме *рис*. 23 горит только когда замкнуты все три переключателя. Если замкнуты *оба* верхних переключателя, выход первого вентиля “И” заставит сработать первое реле второго вентиля “И”. Нижний же переключатель заставит сработать второе реле второго вентиля “И”.

Если мы обозначим отсутствие напряжения 0, а его присутствие — 1, то выходной сигнал вентиля “И” зависит от входного следующим образом:

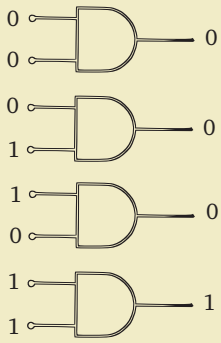


Рис. 25

У вентиля “И” может быть и больше двух входов. Допустим, что мы соединили последовательно три реле:

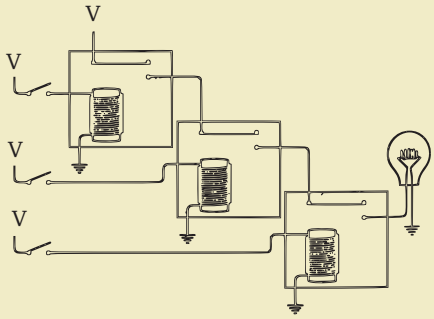


Рис. 26

Лампочка на схеме *рис.* 26 загорится только если будут замкнуты все три переключателя. Эту конфигурацию можно обозначить символом

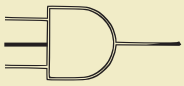


Рис. 27

Называется она “трехвходовым вентилем «И»”.

Следующий логический вентиль состоит из двух реле, соединенных параллельно:

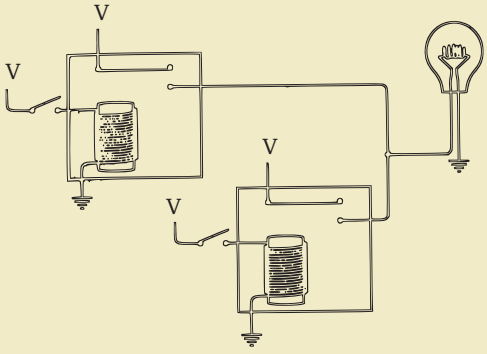


Рис. 28

Выходы двух реле соединены друг с другом, и с этого объединенного выхода питание поступает на лампочку. К включению лампочки приведет срабатывание любого из двух реле. Например, если мы замкнем верхний переключатель, лампочка загорится, так как получит питание от левого реле:

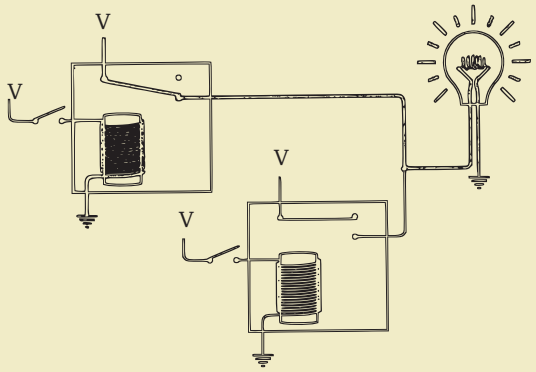


Рис. 29

Если мы разомкнем верхний переключатель и замкнем нижний, лампочка опять загорится. А будет ли гореть лампочка, если замкнуть оба переключателя? Над ответом на этот вопрос подумайте самостоятельно.

В данном случае лампочка горит, когда замкнут верхний *или* нижний переключатель. Ключевое слово здесь “или”, и потому представленная схема называется *вентилем “ИЛИ” (OR)*. На схемах для вентиля “ИЛИ” используют такое символическое обозначение:

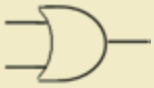


Рис. 30

Этот символ похож на символ вентиля “И”, за исключением того, что сторона входов закруглена.

На выходе вентиля “ИЛИ” есть напряжение, если напряжение подается на любой из двух его входов. Обозначим опять отсутствие напряжения как 0, а его присутствие — 1. Вентиль “ИЛИ” может находиться в следующих состояниях:

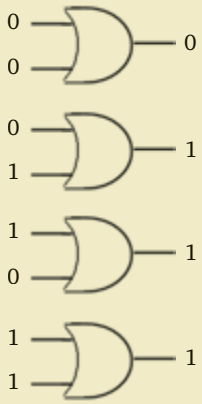


Рис. 31

У вентиля “ИЛИ” может быть более двух входов. Выход его равен 1, если сигнал хотя бы на одном из входов равен 1. Выход, равный 1, соответствует 0 на всех входах.

Ранее мы уже говорили, что наши реле являются двухпозиционными. Обычно, когда переключатель разомкнут, лампочка не горит (см. *рис.* 10–11). Но можно подключить к выходу другой контакт двухпо-

зиционного реле, и тогда лампочка будет гореть, если переключатель разомкнут (см. *рис.* 16–17). В схеме на *рис.* 17 лампочка гаснет, когда мы замыкаем переключатель. Одиночное реле, подключенное таким способом, называется *инвертором*. На схемах он обозначается так:

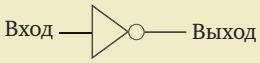


Рис. 32

Как следует из названия, инвертор инвертирует 0 (нет напряжения) в 1 (напряжение есть) и наоборот:

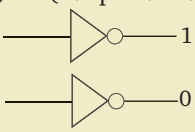


Рис. 33

Теперь, когда в нашем распоряжении имеются инвертор, вентиль “И” и вентиль “ИЛИ”, можно начинать сборку прибора для автоматического выбора нужной нам (вам ☺) кошки. Начнем с переключателей. Первый переключатель замкнут, если вы выбрали кошку, и разомкнут, если выбран кот. Для генерации сигналов Ж и М годится такая схема:

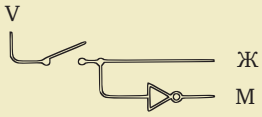


Рис. 34

Когда сигнал на выходе Ж равен 0, сигнал на выходе М равен 1. Подобным же образом можно организовать работу второго включателя, который замкнут для сиамской породы кошек и разомкнут для других:

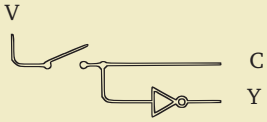


Рис. 35

Работу следующих двух переключателей организовать сложнее. В разных комбинациях они должны воспроизводить четыре разных цвета. Итак, у нас два переключателя, каждый из которых подключен к источнику питания:

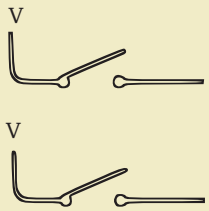


Рис. 36

Когда оба переключателя разомкнуты (как на *рис.* 36), они символизируют выбор белого цвета. Вот как можно, используя два инвертора и один вентиль “И”, сгенерировать сигнал Б, который равен 1 (напряжение есть), если вы выбрали белую кошку, и 0 (нет напряжения), если вы этого не сделали:

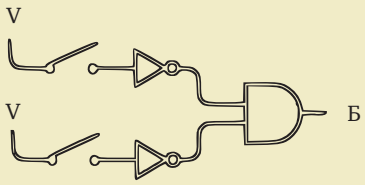


Рис. 37

Когда переключатели разомкнуты, входные сигналы обоих инверторов равны 0, а выходные — 1. А поскольку выходы обоих инверторов являются входами вентиля “И”, на его выходе мы также получаем сигнал 1. Если любой из двух переключателей замкнут, выход вентиля И равен 0.

Чтобы указать черную кошку, мы замыкаем первый переключатель. Это можно реализовать с помощью инвертора и вентиля “И”:

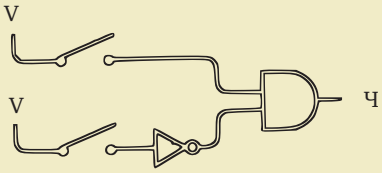


Рис. 38

Выход вентиля “И” равен 1, только если первый переключатель замкнут, а второй разомкнут.

Подобным же образом мы указываем рыжую кошку, замыкая второй переключатель.

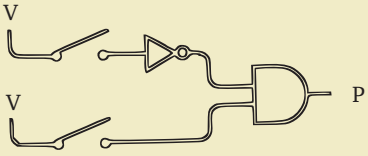


Рис. 39

Наконец, если оба переключателя разомкнуты, выбрано животное “другого” цвета.

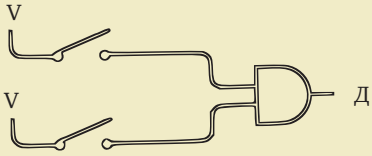


Рис. 40

Теперь объединим эти четыре небольшие схемы в одну большую (как обычно, черные точки на схеме обозначают соединение проводов; провода, на пересечениях которых нет черных точек, *не соединены*):

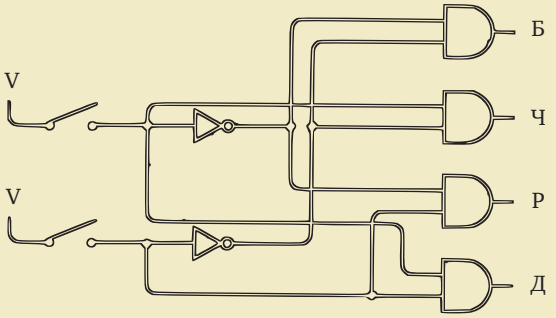


Рис. 41

Конечно, путаница проводов на *рис.* 41 кажется очень сложной. Но если вы приглядитесь

повнимательнее, то убедитесь, что схема действительно работает именно так, как нужно. Проследить, откуда приходят сигналы на входы вентилей И, стараясь не обращать внимание на то, куда *еще* они идут. Если оба переключателя разомкнуты, выход Б равен 1, в любом другом случае он равен 0. Если первый переключатель замкнут, сигнал Ч равен 1, во всех остальных случаях он равен 0, и т.д.

Соединение вентилей и инверторов подчиняется нескольким правилам. Выход одного вентиля (или инвертора) может быть входом одного или нескольких вентилей (или инверторов). Выходы двух или более вентилей (или инверторов) никогда не соединяются между собой.

Эта схема, состоящая из четырех вентилей "И" и двух инверторов, называется *дешифратором 2-й линии на 4*. На вход этого дешифратора подаются 2 бита, которые в различных комбинациях представляют 4 различные величины. На выходе дешифратора имеем 4 сигнала, из которых в любой момент времени значение 1 имеет только один. Какой именно — зависит от входных величин. На тех же принципах строятся дешифраторы 3-й линии на 8, 4-й линии на 16 и т.д.

И вот теперь мы можем нарисовать схему, соответствующую логическому выражению для выбора кошки: **С** × ((**М** × (**Б** + **Р**)) + **Ж** × (1 – **Б**)) + **Ч**. Каждому знаку "+" в выражении на схеме должен соответствовать вентиль "ИЛИ", а каждому знаку "×" — вентиль И. Для реализации выражения "1 – **Б**" можно использовать инвертор. Соответствующая схема показана на *рис*. 42.

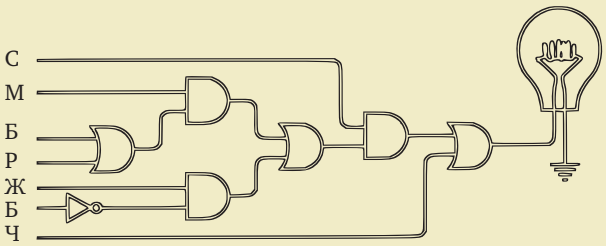


Рис. 42

Символы в левой части схемы идут в том порядке, в каком они записаны в выражении. Соответствующие сигналы приходят от переключателей, соединенных проводами с инверторами и дешифратором 2-й линии на 4.

В заключение предлагаем обратить внимание на название статьи [1]. Конечно, вы уже поняли, что наша главная цель — не кошки, не переключатели, не реле, а компьютеры. О них — в одном из следующих выпусков "В мир информатики".

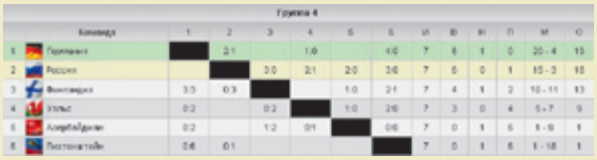
Задания для самостоятельной работы

1. Работу вентиля "И" можно охарактеризовать такой таблицей:

		И
0	0	0
0	1	0
1	0	0
1	0	1

Предложите соответствующую таблицу, имеющую всего три строки, но учитывающую все 4 варианта.

2. Предложите также таблицу, аналогичную упомянутой только что, характеризующую работу вентиля "ИЛИ".



3. Разработайте схему с вентилями "И", "ИЛИ" и инверторами для выбора такого животного: "кот сибирской породы любого цвета, кроме черного, или кошка рыжая сибирской породы или черная любой породы".

Ответы присылайте в редакцию.

Литература

1. От кошек — через переключатели — к компьютерам. / "В мир информатики" № 168 ("Информатика" № 14/2011).

2. *Петцольд Ч.* Код. М.: Издательско-торговый дом "Русская редакция", 2001.

ЭКСПЕРИМЕНТЫ

Эксперимент в графическом редакторе

Выполните, пожалуйста, следующее.

1. В графическом редакторе Microsoft Paint или другом редакторе растрового типа создайте черно-белое изображение окружности. Сохраните файл под номером 1.

2. Увеличьте размеры изображения в два раза (меню **Рисунок — Растянуть/Наклонить**, в группе **Растянуть** установите величину изменения 200% и 200%). Сохраните файл под номером 2.

3. Откройте файл № 1. Уменьшите размеры изображения в два раза (меню **Рисунок — Растянуть/Наклонить**, в группе **Растянуть** установите величину изменения 50% и 50%). Сохраните файл под номером 3.

4. Увеличьте размеры изображения в четыре раза. Сохраните файл под номером 4.

Сравните изображения в файлах № 2 и 4, а также свойства этих файлов.

Сделайте все то же с цветными изображениями (создав 16-цветные рисунки формата bmp).

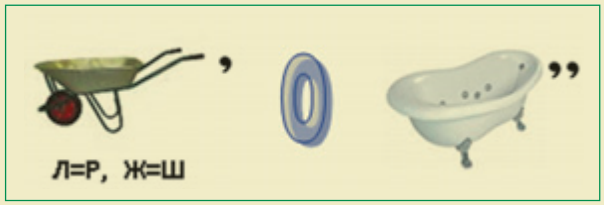
Комментарии к полученным выводам для обоих случаев присылайте в редакцию.

Ребусы, посвященные Году космонавтики. Часть 2

Мы продолжаем публиковать ребусы, которые разработали Анатолий и Ульяна Тимофеевы, ученики Именевской основной школы, Красноармейский р-н Чувашской Республики (учитель **Тимофеева И.А.**).

Ребус № 1

Ребус № 2



Ребус № 3

Ребус № 4

Ребус № 5

Ребус № 6

Ребус № 7

Ребус № 8

Ребус № 9

МИР ИНТЕРНЕТА

Новые домены первого уровня

Вам, конечно, известно, что означают, например, слова *ru* и *com* в адресах сайтов. Это так называемые “домены первого уровня”. В настоящее время в Сети действует более 250 национальных доменов, таких, как *.ru*, *.ua* и *.рф*, а также около 20 наднациональных, таких, как *.com*, *.org* и *.net*.

Недавно корпорация ICANN (*Internet Corporation for Assigned Names and Numbers*), отвечающая за регулирование доменного пространства в Интернете, приняла решение о введении доменных имен верхнего (первого) уровня, совпадающих с названием торговой марки или сообщества. Это означает, что отныне крупные компании, а также группы людей, собранные по интересам (меломаны, болельщики и т.п.), смогут использовать не только уже существующие и привычные всем доменные зоны, но и собственные названия (например, *.nokia* или *.music*). Заводить собственное имя разрешено только крупным корпорациям. Заявки будут рассматриваться экспертами ICANN. Прием заявок начнется 12 января 2012 года и продлится до 12 апреля 2012 года. Реально пользоваться новыми именами можно будет к 2013 году.

Стоимость услуги по получению собственного доменного имени составляет… 185 тыс. долларов…

“ЛОМАЕМ” ГОЛОВУ

Составить слово

Н.А. Владимирова,
учитель информатики гимназии № 2, г. Заозерный Красноярского края

Из букв данных слов, переставив их, необходимо составить термин[4], связанный с информатикой и компьютерами. Например, из слов *морс* и *порт* можно получить слово *просмотр*.

А
1. Сад, ре.
2. Ре, пат, ад.
3. Хек, аут, тир, ар.

Б
1. Бес, кий.
2. Клон, бот.
3. Арбуз, ре.

В
1. Стек, вар.
2. Сиг, дело, вина.
3. Дело, ре, витамин.

Г
1. Пир, стек, тег.
2. Факир, га.
3. Ель, сто, три, порог, фа.

Д
1. Ад, кисет.
2. Стан, дело, ля, тир.
3. Лига, до.

Е Ё
1. Сеть, ком.
2. Шов, ре (следует получить фамилию).
3. Лес, и.

Ж
1. Кий, жест.
2. Кость, жест.
3. Чиж, едок, кристалл, кий, си.

З
1. Го, коза, вол.
2. Ваза, лик.
3. Ваза, скат.

И
1. Пир, том.
2. Мир, нота, икра.
3. Лот, пень, и, лис.

К
1. Пан, кок.
2. Кит, вал, аура.
3. Титул, он, кол.

Л
1. Тис, лак.
2. Иней, лак.
3. Лак, иго.

М
1. Сом, рак.
2. Комод, рана, мак.
3. Ель, дом.

Н
1. Овраг, тина.
2. Сойка, рант.
3. Ель, тон, си.

О
1. Робот, абак.
2. Рот, опера.
3. Точка, си.

П
1. Ар, плита.
2. Магма, тор, пик.
3. Кол, топор.

Р
1. Ор, бот.
2. Сет, рация, риза.
3. Катер, род.

С
1. Крен, ас.
2. Вор, рак, сито.
3. Стан, иск, си.

Т
1. Ре, тайм.
2. Тура, стек.
3. Мел, тиран.

У
1. Ковка, па, у.
2. Ре, лупа, иена.
3. Сукно, вата.

Ф
1. Фарт, ом.
2. Мул, фора.
3. Франт, ор.

Х
1. Ар, хек.
2. Хаки, актер, тир, ас.
3. Тире, холл (следует получить фамилию).

Ц
1. Сон, тост, цель.
2. Лось, цент, сто.
3. Рант, цельный.

Ч
1. Каша, череп.
2. Енот, час.
3. Сота, чат.

Ш
1. Рота, шифр.
2. Док, штрих.
3. Ре, штык.

Щ
1. Акт, щель.
2. Кнут, щель.
3. Лещ, чок.

Э
1. Рак, эн.
2. Ля, эму, рот.
3. Эхо, ноль, корт.

Ю
1. Ре, юз.
2. ВИКА, НЮ (название ЭВМ).
3. ИКС, НЮ.

Я
1. Ор, яд.
2. Кость, яр.
3. Рык, ля.

От редакции

1. Прокомментируйте найденные термины и фамилии ученых.

2. Ответы присылайте в редакцию (можно выполнять не все задания).

Вкусные ломтики

Мама очень вкусно поджаривает ломтики хлеба, пользуясь специальной маленькой сковородкой. Для готовности каждый ломтик должен быть поджарен с двух сторон. Поджаривание каждой стороны ломтика длится 30 секунд, причем на сковороде умещается рядом только два ломтика.

За какое минимальное время при этих условиях мама может приготовить:

а) четыре ломтика хлеба;

б) пять ломтиков?

[4] Или фамилию ученого. — **Прим. ред.**

Три лампочки

Есть две изолированные друг от друга комнаты. В одной из них расположены три лампочки, в другой — трехклавишный выключатель к этим лампочкам, каждая клавиша которого включает одну и только одну из них, но какую именно — неизвестно. Изначально все лампочки выключены. Требуется за минимальное число переходов из одной комнаты в другую определить, какая клавиша какой лампочкой управляет. Как это сделать?

Последовательности цифр

Последовательность строк, состоящих из цифр, строится по следующему правилу. В начальный момент в строке записана цифра 0. На каждом из последующих шагов выполняется следующая операция: в очередную строку записывается удвоенная предыдущая строка, а в конец строки приписывается очередная цифра (на i-м шаге приписывается цифра i).

Ниже показаны первые строки, сформированные по описанному правилу (для удобства в скобках указаны номера строк, начиная с 0):

(0) 0
(1) 001
(2) 0010012
(3) 001001200100123

Всего получены 10 строк. Определите:

1) на какие 10 цифр заканчивается последняя строка;

2) сколько раз в последней строке встречается цифра 4;

3) сколько всего в ней цифр в последней строке;

4) какая цифра стоит в последней строке на 1012-м месте (считая слева направо);

5) сколько всего цифр представлено во всех строках.

61 монета, две фальшивые

Есть 61 монета одинакового внешнего вида. Известно, что две из них — фальшивые. Все настоящие одинакового веса, обе фальшивые — тоже одинакового веса, отличающегося от веса настоящих монет.

Как можно узнать с помощью трех взвешиваний на чашечных весах без гирь, легче или тяжелее настоящих фальшивые монеты? (Определять фальшивые монеты не требуется.)

Звездочки, нули и единицы

Замените звездочки единицами и нулями так, чтобы после преобразования полученных двоичных чисел в десятичные по горизонтали сумма равнялась 135, а по вертикали — 103:

```
      *
    * * *
      *
* * * 0 * * *
      *
    * * *
      *
```

Числовой ребус “БАЙТ на БАЙТ”

Решите числовой ребус:

```
          Б А Й Т
        × Б А Й Т
        ---------
          Б * * *
      * * Б * *
    * * * Б *
  * * * * Б
  ---------
* * * * Б * * *
```


ЗАДАЧНИК

Торговка и пирожки[5]

Шла торговка на рынок продавать пирожки. По дороге она проголодалась и съела сначала пирожок и половину остатка, затем еще пирожок и пол-остатка, затем еще пирожок и пол-остатка. А затем по дороге воры украли 7 пирожков и пол-остатка. На рынок торговка принесла 1 пирожок. Сколько пирожков было?

[5] Задача предназначена для учеников начальной школы и учащихся 5–7-х классов.

Шестнадцать офицеров

В каждом из четырех полков для парада выбрано по 4 офицера разных званий: полковник, майор, капитан, лейтенант. Требуется разместить условные обозначения этих офицеров (М3 — майор из 3-го полка, Л1 — лейтенант из 1-го полка и т.п.) в виде квадрата так, чтобы в каждом столбце и в каждой строке были обозначения офицеров разных званий и разных полков.

На шахматной доске

Имеется шахматная доска с обозначением клеток согласно стандартной шахматной нотации ($a1$ — нижняя левая, …, $h8$ — верхняя правая):

$a8$	$b8$	$c8$	$d8$	$e8$	$f8$	$g8$	$h8$
$a7$	$b7$	$c7$	$d7$	$e7$	$f7$	$g7$	$h7$
$a6$	$b6$	$c6$	$d6$	$e6$	$f6$	$g6$	$h6$
$a5$	$b5$	$c5$	$d5$	$e5$	$f5$	$g5$	$h5$
$a4$	$b4$	$c4$	$d4$	$e4$	$f4$	$g4$	$h4$
$a3$	$b3$	$c3$	$d3$	$e3$	$f3$	$g3$	$h3$
$a2$	$b2$	$c2$	$d2$	$e2$	$f2$	$g2$	$h2$
$a1$	$b1$	$c1$	$d1$	$e1$	$f1$	$g1$	$h1$

Из некоторой начальной клетки нужно проложить маршрут в клетку $a1$, соблюдая следующее правило: каждый ход делается либо на одну клетку влево, либо на одну клетку вниз.

Перечислите все такие маршруты, ведущие из начальной клетки $c5$ (каждый маршрут должен начинаться клеткой $c5$, далее через запятую указываются промежуточные клетки маршрута, заканчивается маршрут клеткой $a1$).

А и Б (но не на трубе ☺)

Для составления цепочек из n букв разрешается использовать две буквы: А и Б, причем никакая буква не должна стоять в цепочке подряд три или более раз. Перечислите все цепочки, удовлетворяющие указанным выше правилам, при $n = 5$.

Если закодировать ВГАБ

Для кодирования букв А, Б, В и Г решили использовать систему кодировки в виде двухразрядных двоичных чисел (от 00 до 11, соответственно). Если таким образом закодировать последовательность символов ВГАБ, а результат записать в восьмеричной системе счисления, то какое число получится?

А какое максимальное число можно при этом получить, если конкретный вариант указанной кодировки неизвестен? А какое минимальное?

Расписание уроков

В понедельник в одном из классов должно быть проведено 4 урока — по математике, физике, информатике и биологии. Учителя высказали свои пожелания для составления расписания. Учитель математики хочет иметь первый или второй урок, учитель физики — второй или третий урок, учитель информатики — первый или четвертый, учитель биологии — третий или четвертый.

Какие при этих условиях могут быть варианты расписания? (Перечислите все возможные варианты.)

В ответе принять следующие обозначения: М — математика, Ф — физика, И — информатика, Б — биология.

Еще раз о правильных часах

Часовому мастеру принесли трое часов и попросили выверить их ход. Мастер включил секундомер и посмотрел на часы № 1 и 2. За 11 минут хода часов № 1 часы № 2 отсчитали 10 минут. Потом он сравнил часы № 2 и 3: за 12,5 минуты хода часов № 2 часы № 3 прошли 12 минут. Посмотрев затем в течение 8,25 минуты на часы № 1, мастер остановил секундомер и впервые взглянул на него — он отсчитал ровно 30 минут. Определите, какие часы идут точно.

Такая задача была опубликована в одном из весенних номеров. Благодаря Юрия Базылева и Карину Галушкову из школы № 1 поселка Надвоицы, Республика Карелия (учитель **Богданова Л.М.**), приславших ответы, редакция решила еще раз предложить задачу для решения. Укажите также номера часов в порядке увеличения “скорости” их работы.

За активное участие в выполнении заданий, предложенных в рубрике “Поиск информатики” в прошлом учебном году, редакция решила наградить дипломами следующих читателей:

— Базылева Юрия и Галушкову Карину, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Богданова Л.М.**;

— Буханова Василия, Григорьева Кирилла и Юхтенко Илью, г. Воронеж, лицей № 2, учитель **Комбарова С.И.**;

— Зорихина Алексея и Шишкину Анастасию, Свердловская обл., г. Нижняя Салда, школа № 7, учитель **Зорихина Н.Ю.**;

— Микулика Илью, г. Астрахань, школа № 33 им. Н.А. Мордовиной, учитель **Лепехина С.М.**;

— Ноздрина Данилу, Республика Башкортостан, г. Стерлитамак, школа № 17, учитель **Орлова Е.В.**;

— Полюховича Максима и Ядзевичюса Стаса, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**;

— Шадрину Юлию, Чувашская Республика, г. Канаш, Канашский педагогический колледж, преподаватель **Воеводина Р.В.**

Списки читателей, приславших ответы на задания, предложенные в апрельских номерах “В мир информатики”, будут опубликованы в следующем выпуске.

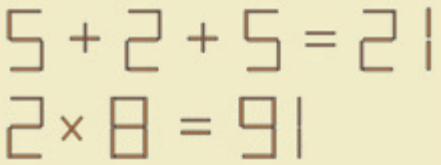
Конкурс № 88

Чемпионат СНГ по перекладыванию спичек ☺

Задачи, включенные в данный конкурс[6], предназначены для учащихся начальной школы и учеников 5–7-х классов.

Задача 1 “Правильные ли примеры?”

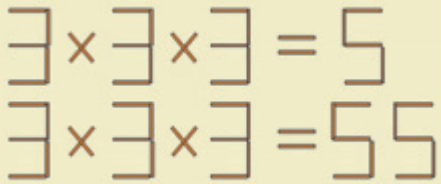
Ученик изобразил при помощи спичек такие непонятные примеры:



Но, как ни странно, учительница похвалила ученика и сказала, что все примеры правильные. Как такое могло произойти?

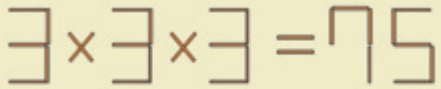
Задача 2 “Тройки и пятерки”

Переставьте в каждом из двух примеров по две спички так, чтобы оба данных равенства оказались верными.



Задача 3 “Снова три тройки”

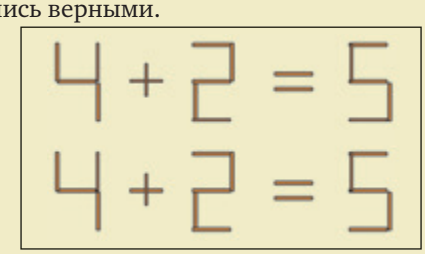
Переставьте две спички так, чтобы данное равенство оказалось верным.



Найдите два различных способа.

Задача 4 “Переставить внутри”

Переставьте внутри прямоугольника две спички так, чтобы одновременно оба данных равенства оказались верными.



[6] Их подготовил Лейб Штейнгарц, Иерусалим, Израиль.

Задача 5 “Ребус из двух фигурок”

При помощи спичек изобразили две одинаковые фигурки:

При некотором воображении (как это принято при разгадывании ребусов с картинками) можно, ничего не меняя, прочесть тут некоторое русское слово. Какое?

Задача 6 “Ребус из четырех фигурок”

При помощи спичек на этот раз изобразили четыре одинаковые фигурки:

Оказывается, можно переставить одну спичку, а еще одну совсем убрать так, чтобы (как и в предыдущей задаче) удалось тут прочесть некоторое русское слово. Что это за слово?

Задача 7 “Ребус из пяти фигурок”

На этот раз достаточно переставить две спички так, чтобы (как и в предыдущих двух задачах) удалось прочесть некоторое русское слово. Какое это слово на этот раз?

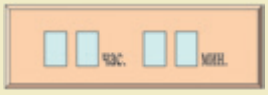
Задача 8 “Шерлок Холмс и электронные часы”

В тот день, хотя и в разное время, в Лондоне совершили три дерзких ограбления. Распутывая это непростое дело, Шерлок Холмс обнаружил странную закономерность: в момент каждого из ограблений на электронных часах высвечивались четыре цифры, показывающие часы и минуты. При этом оказалось, что если изобразить эти цифры при помощи спичек, то каждая цифра будет содержать спичек больше, чем ближайшая к ней цифра справа.

Шерлоку Холмсу по этим данным удалось установить точное время всех трех ограблений.

А удастся ли это вам?

Кстати, не смогли ли бы вы также выяснить, была ли погода солнечной во время этих преступлений?



Задача 9 “Имя космонавта”

Из спичек образовали слово ГОРА.

Переставьте одну спичку так, чтобы получилось имя (краткий вариант) российского космонавта.

Задача 10 “Два бокала с шариками”

Из спичек выложили два одинаковых “бокала”:

Представьте, что внутри этих бокалов имеется по шарику.

Переставьте 3 спички так, чтобы получились точно такие же два бокала, но без шариков.

Задача 11 “Два бокала и центральная симметрия”

Из спичек выложили два одинаковых “бокала” (на этот раз без шариков):

Переставьте одну спичку и еще одну уберите так, чтобы получилась фигура, имеющая центр симметрии.

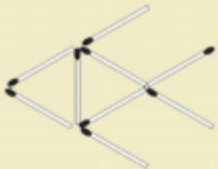
Задача 12 “Два бокала и осевая симметрия”

Из спичек выложили два одинаковых “бокала”:

Переставьте две спички так, чтобы получилась фигура, имеющая ось симметрии. Найдите два различных способа.

Задача 13 “Рыбка”

Из спичек составили фигуру:



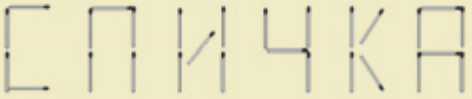
Переставьте 4 спички так, чтобы получить рыбку точно такой же формы:

1) плывущую направо;

2) не плывущую ни вправо, ни влево.

Задача 14 “Спичка”

В слове из спичек:



1) добавьте одну спичку так, чтобы получилось другое слово;

2) переставьте две спички и одну спичку добавьте так, чтобы получилось некоторое другое слово.

Ответы отправьте в редакцию до 5 ноября по адресу: 121165, Москва, ул. Киевская, д. 24, “Первое сентября”, “Информатика” или по электронной почте: *vmi@1september.ru*. Пожалуйста, четко укажите в ответе свои фамилию и имя, населенный пункт, номер и адрес школы, фамилию, имя и отчество учителя информатики, а также класс, в котором вы учитесь (еще раз обращаем внимание на то, что задания конкурса предназначены для учащихся начальной школы и учеников 5–7-х классов).

Можно выполнять не все задания.

Конкурс № 89 “Графические редакторы”

Ответьте, пожалуйста, на следующие вопросы.

1. В векторном редакторе нарисован квадрат со стороной в 10 пикселей. Изображение было сохранено в файле, размер которого составил *N* Кб. Разрешающая способность монитора при этом была равна 1024 на 768 пикселей. Чему будет равен размер графического файла, если сторону квадрата увеличить в 2 раза?

2. Файл в формате bmp содержал рисунок квадрата со стороной в 10 пикселей. Рисунок загрузили в редактор и увеличили сторону квадрата в 5 раз. Как изменится размер графического файла? В обоих случаях изображение записывалось с глубиной цвета 24 бита на пиксель.

3. В растровом редакторе было создано черно-белое изображение, которое сохранили в файле формата bmp. Размер файла составил 350 Кб. Разрешающая способность монитора при этом была равна 1024 на 768 пикселей. После этого она была изменена на 800 на 600 пикселей. Созданный файл был сохранен вновь. Каким будет его размер?

4. Какой графический редактор, растровый или векторный, точнее построит:

1) окружность радиусом в 100 пикселей;

2) окружность радиусом в 200 пикселей;

3) прямую линию с наклоном в 15 градусов;

4) прямую линию с наклоном в 45 градусов?

5. В векторном редакторе было создано изображение с размерами на экране 150 × 300 пикселей. При этом разрешение экрана составляло 96 dpi. Затем изображение экспортировали в формат bmp с разрешением 300 dpi. Каковы будут размеры изображения (в пикселях) при выводе его на экран? А при выводе на бумагу?

Проведите также эксперимент, описанный в соответствующей рубрике в этом выпуске.

Ответы с обоснованием отправьте в редакцию до 5 ноября. Можно отвечать не на все вопросы.

ПОДПИСНАЯ КАМПАНИЯ ОТКРЫТА

журнал

Информатика – Первое сентября

ТАРИФНЫЕ ПЛАНЫ НА ПОДПИСКУ

1-е полугодие 2012 г.

Максимальный – от 999 руб.

бумажная версия + CD + доступ к электронной версии на сайте

Подписаться можно на почте по каталогам «Роспечать» (индекс 32291), «Почта России» (индекс 79066) или на сайте www.1september.ru

Оптимальный – 594 руб.

электронная версия на CD + доступ к электронной версии на сайте

Подписаться можно на почте по каталогам «Роспечать» (индекс 19179), «Почта России» (индекс 12684) или на сайте www.1september.ru

Экономичный – 200 руб.

доступ к электронной версии на сайте

Подписаться по данному тарифному плану можно только на сайте www.1september.ru

Бесплатный – 0 руб.

для педагогических работников образовательных учреждений, участвующих в Общероссийском проекте «Школа цифрового века». Подробности – на digital.1september.ru

Бумажная версия

CD с электронной версией журнала и дополнительными материалами для практической работы

Доступ к электронной версии журнала на сайте. Дополнительные материалы включены

Именные сертификаты – пользователям электронной версии на сайте www.1september.ru

ЭКОНОМИЧНЫЙ тарифный план

ОПТИМАЛЬНЫЙ тарифный план

МАКСИМАЛЬНЫЙ тарифный план

На сайте www.1september.ru подписку можно оплатить по кредитным картам VISA MasterCard