

№3

Издание основано в 1995 г.

inf.1september.ru

УЧЕБНО - МЕТОДИЧЕСКИЙ ЖУРНАЛ ДЛЯ УЧИТЕЛЕЙ ИНФОРМАТИКИ

ИНФОРМАТИК А

20 лет



УРА! НЮ!!!

издательский
дом
1september.ru

Первое сентября

март
2015

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



НА ОБЛОЖКЕ

ЗЕМЛЯ ГУЛЛИВЕРА

► В марте исполняется 20 лет одной из самых известных интернет-компаний Yahoo!. Предыдущее предложение не случайно заканчивается двумя знаками препинания, поскольку восклицательный знак входит в название. При учреждении Yahoo! его пришлось добавить — название "Yahoo" было уже занято торговой маркой соуса для кетчупа. По поводу же самого слова имеются разные точки зрения, но авторская — точка зрения основателей компании — заключается в том, что название посвящено народу "йеху", земли которого посетил в своих путешествиях Гулливер.

Ken Wolter / Shutterstock.com

В НОМЕРЕ

- 3** ПАРА СЛОВ
 - Google Glass: на закате славы?
- 4** УГЛУБЛЕНКА
 - Изучение имитационного моделирования в AnyLogic в углубленном курсе информатики
- 20** ЕГЭ
 - Тренинг по информатике: "разбор полетов"
- 34** УЧЕБНИКИ
 - Алгоритмизация и программирование
- 46** СЕМИНАР
 - Гармонический ряд
- 48** ЗАНИМАТЕЛЬНЫЕ МАТЕРИАЛЫ ДЛЯ ПЫТЛИВЫХ УЧЕНИКОВ И ИХ ТАЛАНТЛИВЫХ УЧИТЕЛЕЙ
 - "В мир информатики" № 206

В ЛИЧНОМ КАБИНЕТЕ

Облачные технологии от Издательского дома "Первое сентября"

Уважаемые подписчики бумажной версии журнала!

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

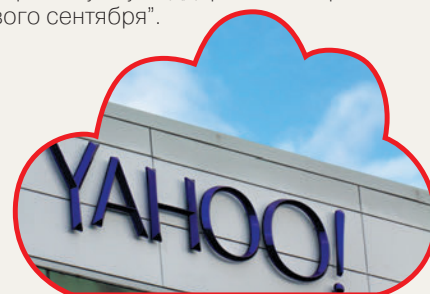
Для доступа к материалам воспользуйтесь, пожалуйста, кодом доступа, вложенным в № 1/2015.

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

Для активации кода:

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

Справки: podpiska@1september.ru или через службу поддержки на портале "Первое сентября".



ЭЛЕКТРОННЫЕ МАТЕРИАЛЫ

Презентации к статьям номера

ИНФОРМАТИКА

ПОДПИСНЫЕ ИНДЕКСЫ

по каталогу "Почта России": 79066 — бумажная версия, 12684 — электронная версия

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

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

РЕДАКЦИЯ:

гл. редактор С.Л. Островский
редакторы

- Е.В. Андреева,
- Д.М. Златопольский (редактор вкладки "В мир информатики")

Дизайн макета И.Е. Лукьянов
верстка Н.И. Пронская
корректор Е.Л. Володина
секретарь Н.П. Медведева
Фото: фотобанк Shutterstock
Журнал распространяется по подписке
Цена свободная
Тираж 16 000 экз.
Тел. редакции: (499) 249-48-96
E-mail: inf@1september.ru
<http://inf.1september.ru>

ИЗДАТЕЛЬСКИЙ ДОМ "ПЕРВОЕ СЕНТЯБРЯ"

Главный редактор:

Артем Соловейчик (генеральный директор)

Коммерческая деятельность:

Константин Шмарковский (финансовый директор)

Развитие, IT и координация проектов:

Сергей Островский (исполнительный директор)

Реклама, конференции и техническое обеспечение Издательского дома:

Павел Кузнецов

Производство:

Станислав Савельев

Административно-хозяйственное обеспечение:

Андрей Ушков

Педагогический университет:

Валерия Арсланьян (ректор)

ЖУРНАЛЫ ИЗДАТЕЛЬСКОГО ДОМА "ПЕРВОЕ СЕНТЯБРЯ"

- Английский язык – А.Громушкина
- Библиотека в школе – О.Громова
- Биология – Н.Иванова
- География – и.о. А.Митрофанов
- Дошкольное образование – Д.Тюттерин
- Здоровье детей – Н.Сёмина
- Информатика – С.Островский
- Искусство – О.Волкова
- История – А.Савельев
- Классное руководство и воспитание школьников – М.Битянова

- Литература – С.Волков
- Математика – Л.Рослова
- Начальная школа – М.Соловейчик
- Немецкий язык – М.Бузоева
- ОБЖ – А.Митрофанов
- Русский язык – Л.Гончар
- Спорт в школе – О.Леонтьева
- Технология – А.Митрофанов
- Управление школой – Е.Рачевский
- Физика – Н.Козлова
- Французский язык – Г.Чесновицкая
- Химия – О.Блохина
- Школа для родителей – Л.Печатникова

Школьный психолог – М.Чибисова

УЧРЕДИТЕЛЬ:

ООО "ИЗДАТЕЛЬСКИЙ ДОМ «ПЕРВОЕ СЕНТЯБРЯ»"

Зарегистрировано
ПИ № ФС77-58447
от 25.06.2014

в Роскомнадзоре

Подписано в печать:
по графику 15.01.2015,
фактически 15.01.2015
Заказ №

Отпечатано в ОАО "Первая Образцовая типография" Филиал "Чеховский Печатный Двор"

ул. Полиграфистов, д. 1,
Московская область,
г. Чехов, 142300

Сайт: www.chpd.ru
E-mail: sales@chpk.ru
Факс: 8 (495) 988-63-76

АДРЕС ИЗДАТЕЛЯ:

ул. Киевская, д. 24,
Москва, 121165
Тел./факс: (499) 249-31-38

Отдел рекламы:

(499) 249-98-70
<http://1september.ru>

ИЗДАТЕЛЬСКАЯ ПОДПИСКА:

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



Google Glass: на закате славы?

► Google Glass (“очки Google”) уже хорошо знакомы всем, кто следит за новинками мира Hi-Tech, — правда, в основном по множеству публикаций в “околокомпьютерных” СМИ и отчасти по выставкам аналогичной направленности. Кто-то посчитал их очередной “модной игрушкой”, кто-то — весьма перспективным, но пока, увы, недоработанным концептом, а многие действительно с нетерпением ждали начала серийной продажи Google Glass к лету 2014 года.

В принципе эту разработку, на которую компания Google потратила немало времени и средств, можно считать весьма многообещающей (по крайней мере теоретически). Именно с ее появлением технология “дополненной реальности” (т.е. органичного соединения синтезируемых компьютером изображений и надписей с тем реальным миром, который пользователь очков видит вокруг себя) стала превращаться из фантазии в действительность. Что она может дать обычному пользователю? Много. Например, владелец очков “дополненной реальности” может не просто воспользоваться картой с нарисованным на ней маршрутом, как в обычном GPS-навигаторе, но и увидеть перед собой “путеводную нить” или цепочку флажков (в зависимости от конкретной программной реализации), отмечающую его путь к цели. Двигаясь в толпе и увидев слегка знакомое лицо, пользователь очков сможет получить информацию об этом человеке (если это действительно его знакомый и он записан, скажем, в личной “адресной книге”). В магазине очки “дополненной реальности” позволят при помощи беспроводного доступа к Интернету сразу получить (и сравнить) цены на те или иные товары, на которые смотрит пользователь (распознав их названия), и дадут подсказку — имеет ли смысл совершить покупку здесь или посетить другой магазин (и подсказку, какой именно). А посмотрев на текст, напи-

санный на иностранном языке, можно будет сразу увидеть подстрочный перевод.

Правда, все это — пока в основном в теории. Очки же Google Glass реализовывали прежде всего возможность фотографирования и съемки видео с выкладыванием этого “архива” в Интернете, и лишь немногие из функций “настоящей дополненной реальности”. Плюс к тому — очень высокая цена (на фоне появляющихся более дешевых моделей — конкурентов из Китая и других стран Азии) и ограничение на продажу — очки Google Glass официально поступили в магазины очень немногих стран (Россия, кстати, к ним не относится), а в целом ряде государств к ним отнеслись с остороженностью, как к “шпионским” средствам, либо вообще запретили.

И вот — финал. По сообщению Washington Post, компания Google после 19 января 2015 г., прекратила продажу Google Glass модели “Explorer” по цене \$1500 и свернула программу Glass Explorer. Также официально заявлено, что проект Google Glass выведен из научно-исследовательской лаборатории Google X и будет передан в самостоятельную исследовательскую структуру под руководством Тони Фаделла, бывшего главы подразделения Apple по выпуску iPod. В итоге в нынешнем виде очки не будут уже продаваться никогда, остатки уже выпущенной партии распродадут различным фирмам, которые уже нашли применение очкам Google, а разработчики будут создавать более продвинутую (соответственно — более конкурентоспособную и интересную для пользователей) модель и, в частности, кардинально изменят дизайн устройства. Впрочем, компания Google пока не дает никаких прогнозов о сроках появления обновленной версии очков.

Источник информации: <http://www.ridus.ru/news/176227> со ссылкой на The Washington Post (http://www.washingtonpost.com/business/technology/google-to-stop-consumer-sales-of-glass-to-re-design-device/2015/01/15/89f4e0e0-9cf1-11e4-86a3-1b56f64925f6_story.html).

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



Изучение имитационного моделирования в AnyLogic в углубленном курсе информатики

И.А. Калинин,
ФГБОУ ВПО Гос. ИРЯ
им. А.С. Пушкина,

Н.Н. Самылкина,
ФГБОУ ВПО МПГУ,

А.И. Калугин,
ФГБОУ ВПО МПГУ

► В предыдущей статье этого цикла [6] мы рассказали вам о мощном методе исследования и поиска решений — имитационном моделировании. Продолжая этот рассказ, мы рассмотрим сегодня еще несколько моделей.

Начнем с одного из видов агентной модели — заодно вспомнив основы работы среды AnyLogic.

Напомним, что бесплатная версия продукта для использования в школах и лицензионное соглашение можно найти по адресу: <http://metodist.lbz.ru/authors/informatika/8/>.

Простейшая модель распространения эпидемии

Одна из ежегодных проблем крупных городов — эпидемии гриппа. Нам требуется создать модель, которая по-

зволит изучить действенность некоторых типовых мер по борьбе с эпидемиями — вакцинации, карантина и т.д.

Для решения этой задачи мы воспользуемся агентной моделью. Моделировать пространство целого города и поведение людей для нас не имеет смысла — поскольку для нас в этой модели не важно, как люди перемещаются по городу. Нам важно только, как численно распространяется заболевание — а для этого “гонять” человечков не надо.

При этом фактически агент будет просто менять свое состояние последовательно: здоров, заражен, болен и выздоровел. Для простоты будем считать, что выздоровевший больше не заболевает.

Мы будем считать, что заболеть человек может двумя путями: “случай-

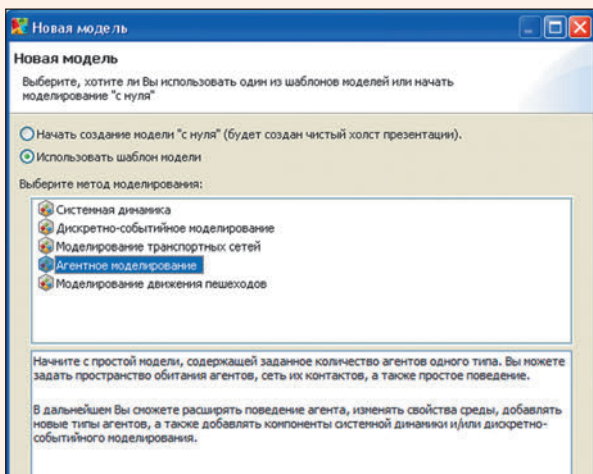
но” (т.е. без непосредственного контакта с заболевшим) и при непосредственном контакте с зараженным или больным. Так что для нас существенными параметрами болезни будут:

- 1) вероятность заражения “из воздуха”;
- 2) инкубационный период;
- 3) среднее время течения заболевания.

Это, конечно, очень упрощенная модель¹, но для проверки основных предположений она вполне годится.

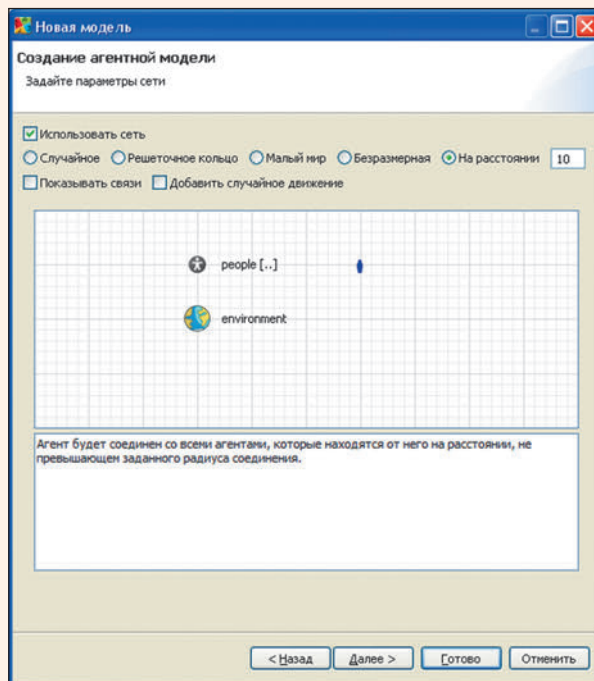
Ключевой вопрос, который мы будем исследовать с помощью нашей модели, это вопрос о параметрах, влияющих на скорость и широту распространения болезни. Опять-таки, упрощая, можем считать ключевой точкой каждого эксперимента момент времени, в который больных больше всего.

Для создания модели мы воспользуемся мастером. Обратимся к команде “Файл/Создать/Модель”, на первом шаге укажем название “Epidemic”, на втором шаге — что за основу берем агентную модель:



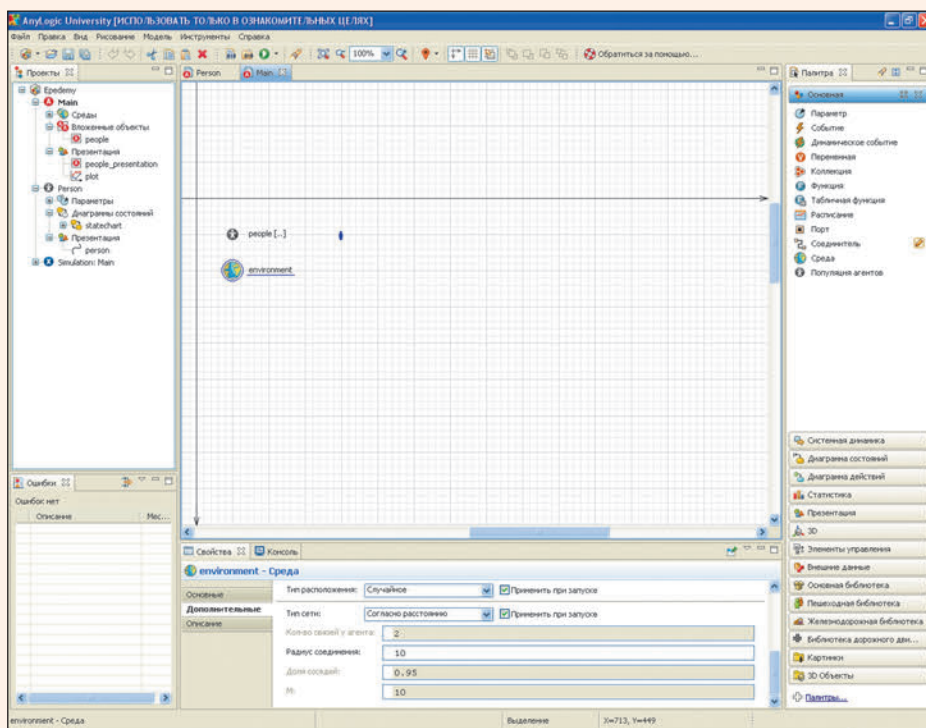
На следующем шаге укажем, что нам понадобятся 500 агентов, далее — что распределены они будут случайно, размер пространства — 400 × 400 (то есть ничего не будем менять).

На шестом шаге укажем, что контактировать объекты будут на расстоянии до 10 единиц:



Больше ничего добавлять не будем и создадим модель — нажмем кнопку “Готово”.

Получим примерно следующее:



¹ Вот пример более аккуратной реализации подхода: http://ipi-cpl.ru/about/articles/kondratyev_epidemic_ntv_2010.

Мы создали модель, в которой при запуске будет создано 500 агентов на пространстве 400 на 400 единиц. Каждый агент будет “связываться с соседями” на расстоянии 10 единиц. Распределятся они по всему пространству случайно, но происходить с ними пока ничего не будет — мы никаких действий им не приписали.

Наша модель, напомним, устроена таким образом, что перемещения агентов нас не интересуют, и фактически агенты будут просто менять свое состояние. Чтобы определить, какие состояния когда будут возникать, мы должны открыть объект “Person” в графическом редакторе и создать диаграмму смены состояний, StateChart.

Наша диаграмма будет показывать последовательность состояний во время заболевания: сначала человек здоров, потом заражен (но еще не знает об этом и свободно перемещается), болеет и, наконец, выздоравливает.

Откроем палитру “Диаграмма состояний” и перетащим на рабочее пространство объекта блоки. Блокам дадим названия: healthOk (Здоров), infected (Заражен), sick (Болен) и cured (Вылечен). Чтобы их было легче различать, сменим цвет заливки.

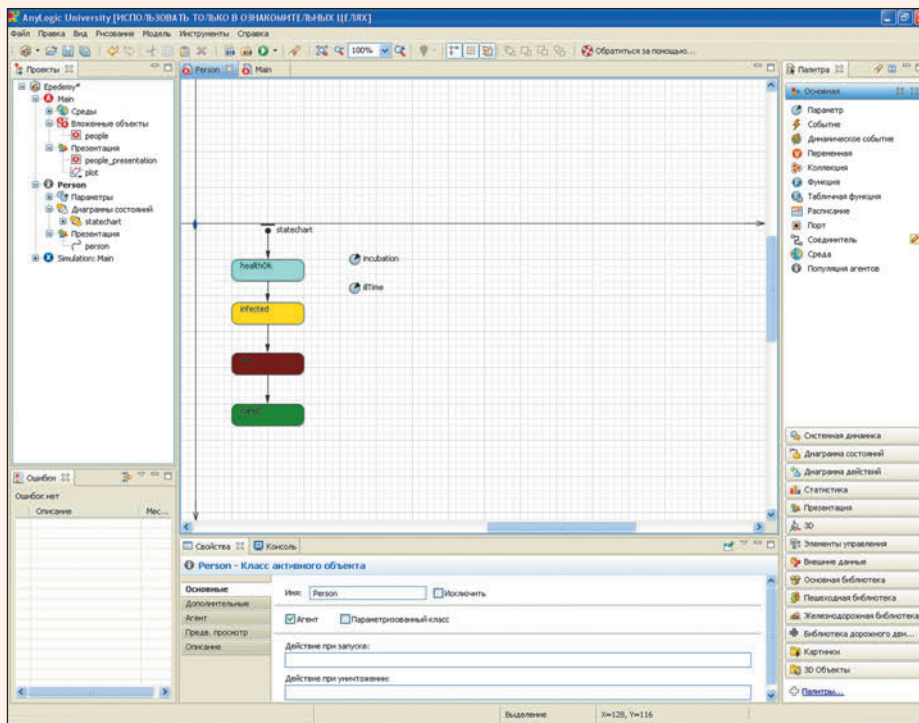
Для простоты предположим, что:

1. Выздоровливают все².
2. Время выздоровления одинаково для всех.
3. Выздоровевший повторно не болеет.

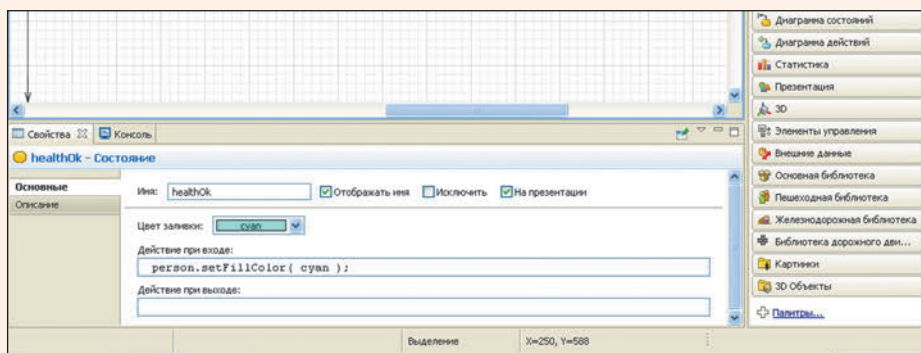
Чтобы впоследствии иметь возможность изучать влияние параметров на процесс, заранее предусмотрим (перетащим) два объекта-параметра из палитры “Основная”:

- incubation (инкубационный период), зададим ему значение целого типа 5;
- illTime (время болезни), зададим ему значение 10.

Блоки свяжем между собой соединителем-стрелкой из палитры “Диаграмма состояний”. Получится примерно следующее:



Стоит заметить, что цвет блоков на состояние знака-человечка в рабочей модели не повлияет. Чтобы следить за происходящим, мы добавим на каждом состоянии команду, меняющую цвет в анимации:

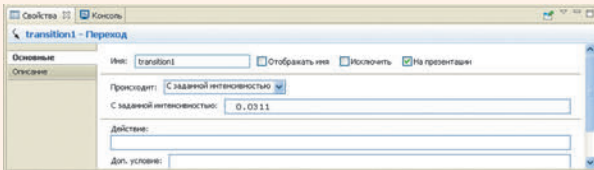


² К сожалению, в реальном мире даже для эпидемии гриппа это не так.

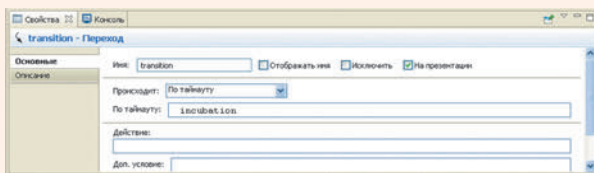
Напишем для каждого блока действие при входе в него. Для первого блока: `person.setFillColor(cyan)`;

Аналогично — для остальных блоков.

Смена состояния будет происходить по стрелкам-связям. Стрелка от “Здоров” к “Заражен” — первичное заболевание будет происходить из-за заражения случайно (без прямого контакта), то есть просто с некоторой заданной интенсивностью. Зададим для стрелки частоту “0.0311”:



Второй переход — от состояния “Заражен” к состоянию “Заболев” будет определяться уже только временем инкубационного периода. Выберем для стрелки параметр “По тайм-ауту”, но вместо конкретного значения сошлемся на имя параметра:



Также поступим и для последнего перехода.

Обратите внимание: пока наша модель никак не зависит от количества контактов и близости людей друг к другу, единственный способ “заразиться” — это попасть под распределение.

Вернемся к общему экрану (закладка Main).

Чрезвычайно неудобно следить за распространением эпидемии “вручную”. Воспользуемся для этих наблюдений средствами автоматического сбора статистики.

Во-первых, включим сбор статистики у агентов. Выберем объект `people`, переключимся на закладку “Статистика”.

Изначально там ничего нет, но у нас есть возможность добавить счетчики. Сначала подсчитаем количество здоровых людей:

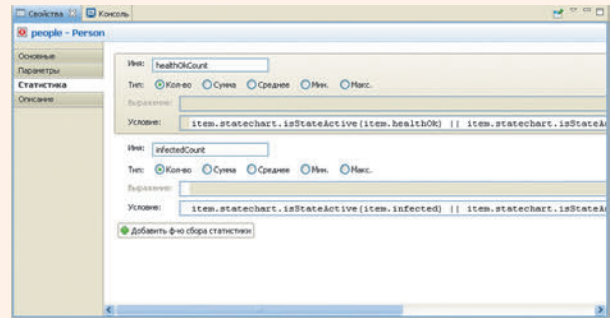
- Добавим функцию сбора статистики.
- Назовем ее `healthOkCount`.
- Подсчитывать будем количество.
- Условие подсчета будет состоять из двух: человек либо находится в состоянии “здоров”, либо “выздоровел”. Вот как выглядит условие “Здоров”:

```
item.statechart.isStateActive(item.healthOk).
```

Соединим его с условием “выздоровел” с помощью связки “или”. Названия статусов мы задали сами.

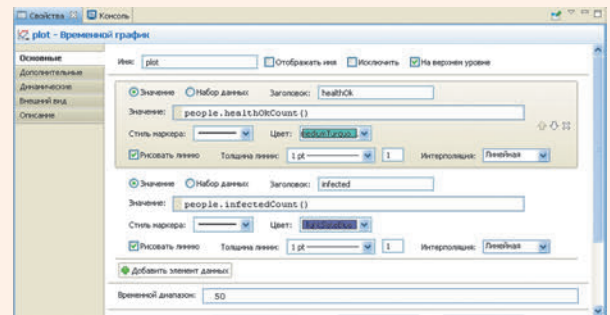
```
item.statechart.isStateActive(item.healthOk) ||
item.statechart.isStateActive(item.cured)
```

- Аналогично добавим счетчик `infectedCount` — с условием “заражен” или “болен”.

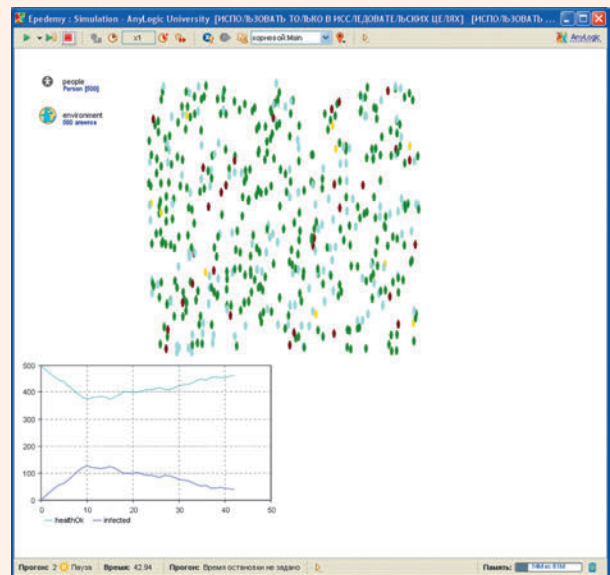


Теперь разместим средство просмотра: временной график.

- Перетащим необходимый объект из палитры “Статистика” на рабочее пространство.
- На закладке “Основные” этого объекта добавим элемент данных (нажмем кнопку).
- Элемент назовем `healthOk`, получать будем значения из нашего объекта `people` с помощью заданной нами функции сбора статистики — `people.healthOkCount()`.
- Точно так же добавим второй элемент данных — `infected`.



Запустим модель и увидим примерно такую картину:

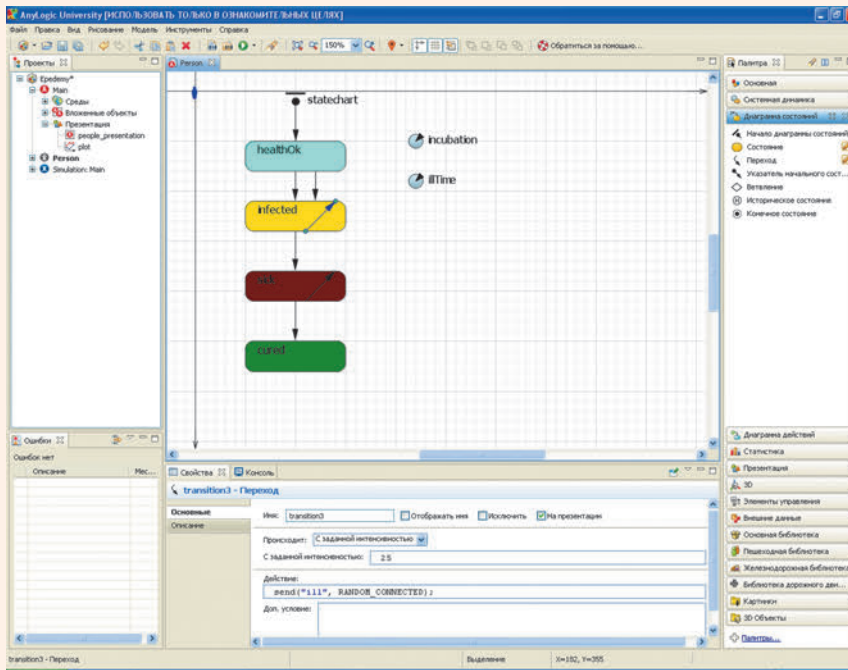


Как видим, максимум заболевших — на 10-й день (удивительно, но это — время инкубационного периода и периода явной болезни, после которо-

го люди начинают выздоравливать). Пока что мы фактически моделируем незаразную болезнь — поскольку заболеваемость от количества уже заболевших и зараженных не зависит никак.

Доработаем нашу модель, учтя распространение инфекции от заболевших. Сделаем мы это с помощью механизма рассылки сообщений.

В диаграмме мы добавим два внутренних (внутри блоков) перехода — внутри состояний `infected` и `sick`. У этих переходов мы с заданной интенсивностью будем рассылать сообщения `“ill”`.



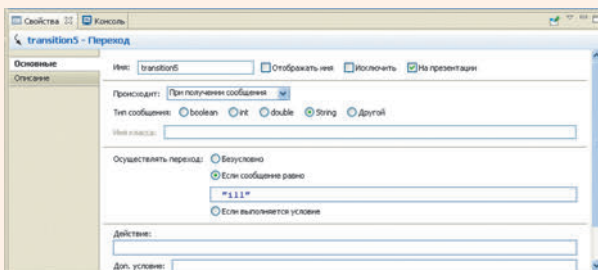
Рассылка сообщений будет случайной, поскольку мы не знаем, с кем контактирует человек во время болезни.

Для отработки сообщения мы добавим второй переход от состояния “Здоров” к состоянию “Заражен”, который будет происходить при получении сообщения `“ill”`:

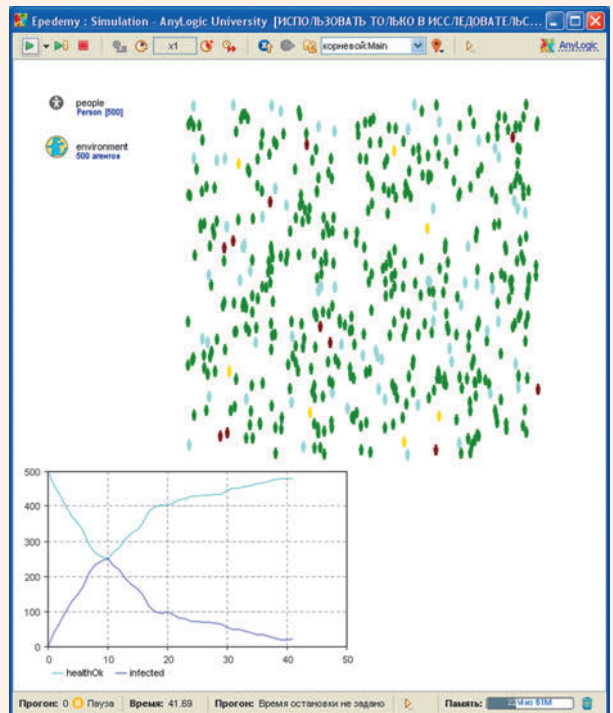
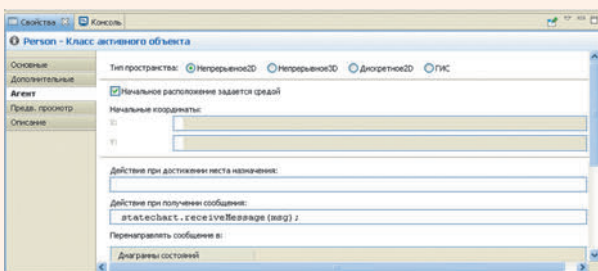
Действие `statechart.receiveMessage(msg);` передает сообщение в диаграмму смены состояний.

Для начала предположим, что больные люди передвигаются по пространству точно так же, как здоровые и зараженные, — то есть поддерживают обычную частоту контактов (25 сообщений в единицу времени).

Выполним прогон и увидим совсем другую картину:

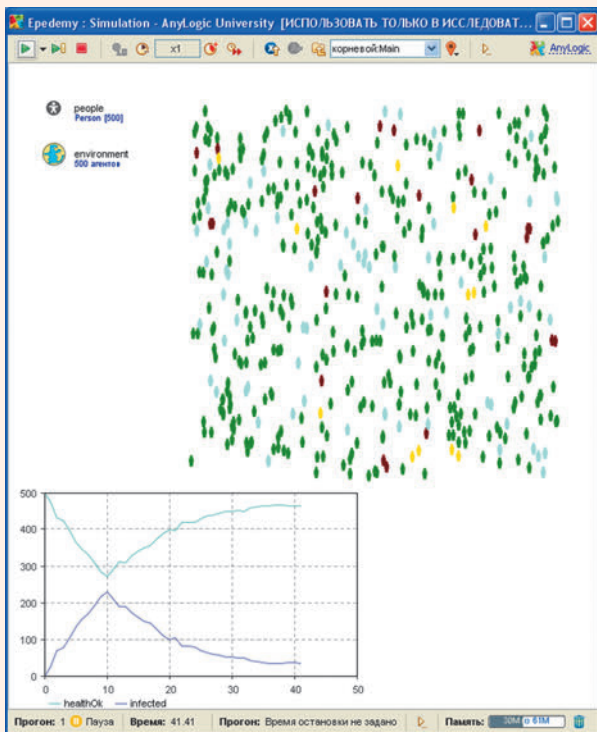


Сообщение будет приходить случайным людям, находящимся на расстоянии 10 единиц (именно это мы задали при создании модели). Чтобы сообщения приходили агентам, зададим свойства класса-агента: выберем в дереве класс `person` и зададим на странице агент действие при получении сообщения:



К моменту начала снижения количества заболевших (10 дней) их количество становится равно количеству здоровых — 250 человек. В прошлый раз оно не успело подняться и до 200.

Введем ограничение: пусть заболевшие сидят в карантине, то есть посылают только три сообщения “болеть”.



Как видим, улучшение есть, но не принципиальное³. Если сделать несколько прогонов, то будет видно, что в обоих вариантах ситуация может быть и хуже, и лучше — в зависимости от распределения людей.

Можно предположить, что существенная разница возникнет в том случае, когда будет несколько “ареалов обитания” людей, — тогда рост очага заболевания будет гораздо ниже. В пределах же уже зараженного поселения статистическая разница будет невелика.

Сделаем нашу модель более подходящей к картине распространения серьезной инфекции:

1. Источником заражения так или иначе всегда является один человек — носитель. Уберем “фоновое” заражение.

2. Если лечения нет, то в распространении эпидемии возникает понятие “смертность”.

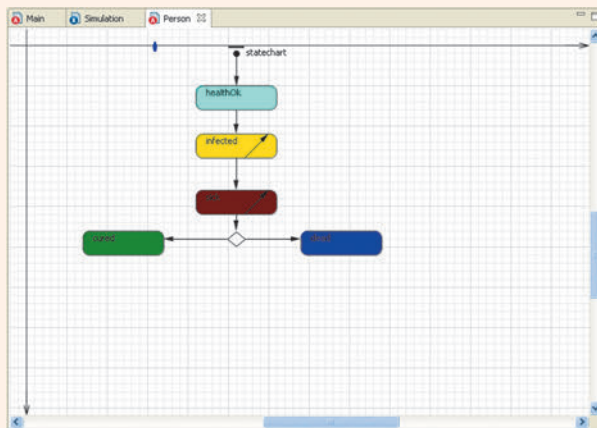
Чтобы исследовать поведение заболевания, в нашей модели, без необходимости каждый раз менять параметры вручную, мы вынесем эти параметры на стартовую страницу и будем их менять.

Первые изменения коснутся главной страницы (Main). Перенесем на нее параметры incubation и

³ Авторам кажется, что это неплохо согласуется с историей распространения, например, чумных эпидемий. И “успешностью”, например, английских запертий.

illTime и добавим новые параметры: contactNorm (со значением 25) и sickNorm (со значением 3) — количество контактов для “свободного” и “заболевшего” людей, а также параметр distance — дистанция, на которой болезнь будет действовать.

Теперь внесем изменения в диаграмму смены состояний. Во-первых, изменим структуру:



1. Уберем первый переход от состояния “Здоров” к состоянию “Заражен”.

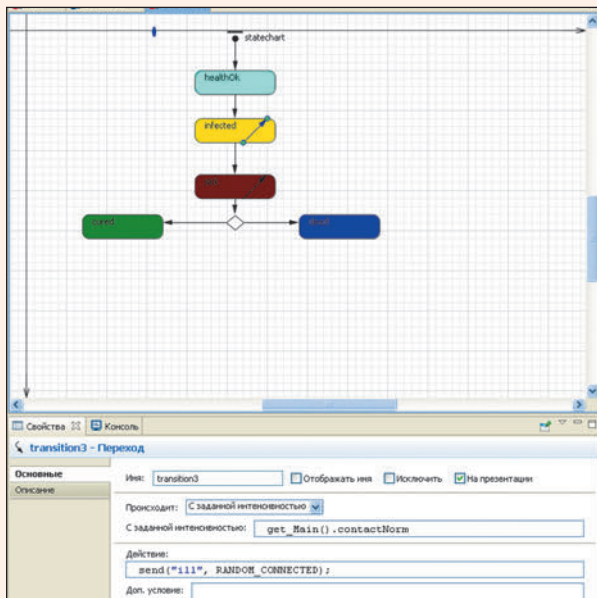
2. Добавим после состояния “Болен” элемент ветвления и от него два исхода: “Выздоровел” и “Умер”.

Условия перехода описываются в параметрах стрелок, выходящих из разветвления. Условие перехода к состоянию “Здоров” будет таким: `uniform() < 0.98`.

То есть если случайное число, равновероятно попадающее в промежуток от 0 до 1, должно оказаться меньше 0,98, — в 98 случаях из 100 человек выздоравливает.

Второе условие мы не будем задавать — под него попадут все остальные.

3. Там, где мы задавали значения частот и таймаутов, в соответствующих полях переходов мы должны будем сослаться на значения параметров из основного объекта. Делается это примерно так: `get_Main().contactNorm`, — так ссылаются из подчиненного объекта на параметр в основном.



Раньше заболевание начиналось с заразившихся “из воздуха”, теперь у нас таких не осталось. Чтобы эпидемия стартовала, в свойствах главного объекта (Main) мы укажем действие — посылку сообщения “ill” первому по списку агенту. Он и будет нашей стартовой точкой. Для этого запишем в поле “Действие при запуске” следующее:

```
people.get(1).statechart.receiveMessage( "ill" );
```

И, наконец, нам нужно будет дать возможность задавать значения параметров на стартовой странице.

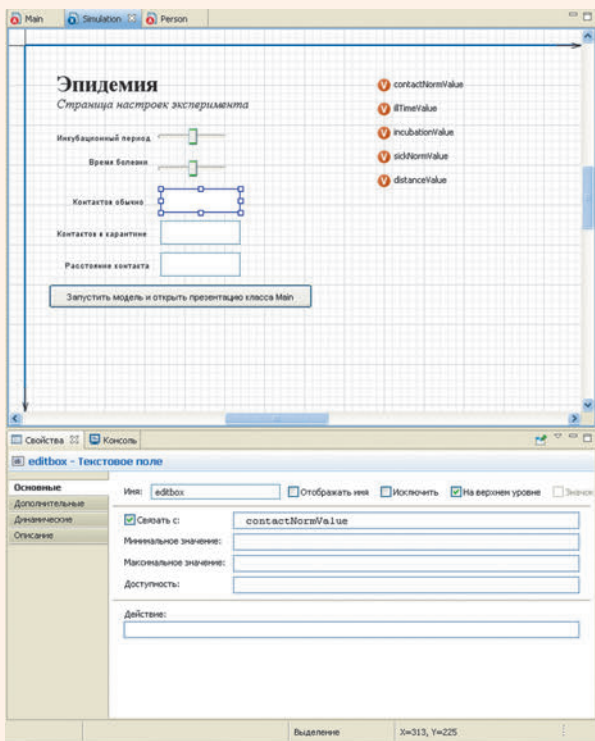
Для этого откроем в графическом редакторе страницу симуляции (simulation) и добавим на нее несколько элементов:

1. Добавим переменные (не параметры, а именно переменные!) из палитры основные. Назовем их так же, как параметры, только с припиской value.

2. Добавим элементы ввода. Сами элементы мы добавим из палитры “Элементы управления”, а подписи к ним — из палитры “Презентация”. Текст подписей наберем как значение свойства Текст.

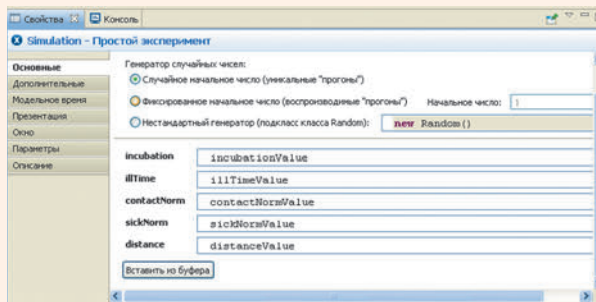
3. Свяжем элементы управления с переменными. Для этого поставим в их свойствах галочку “Связать с” и напишем имя соответствующих переменных.

Должно получиться примерно вот что:



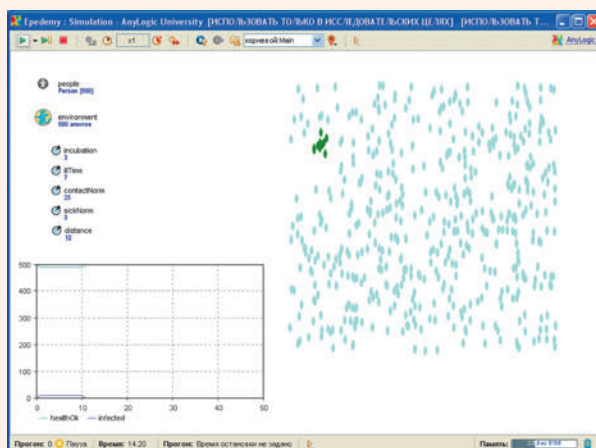
И последнее, мы свяжем параметры с переменными.

4. Выберем в дереве объектов Simulation и на странице Основные зададим в строках появившихся параметров значения переменных:



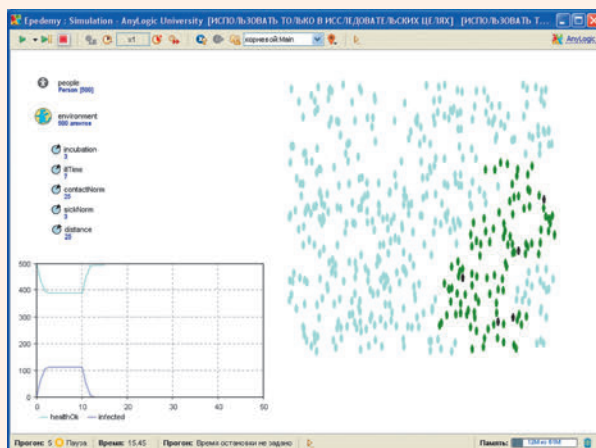
Элементы для сбора и отображения данных о смертности можете сделать сами, как было описано выше.

Теперь выполним несколько прогонов с разными значениями параметров. С параметрами, изначально заданными нами, картина скорее всего будет примерно такой:



Как видим, картина довольно резко изменилась. Болезнь проявилась только на небольшой группе людей. Причина очень проста — остальные не оказались в “радиусе заражения”.

Если же радиус увеличить до 25, картина будет примерно такой:



То есть количество заболевших растет. При этом значения количества контактов, сроков заболевания и т.д. на максимальное количество пораженных будут влиять мало. Основным фактором, как мы и предсказывали, оказывается “площадь контактов”,

то есть фактически — территория плотного расселения. Таким образом, ввод карантина — мера целесообразная, но только для целых районов (то есть областей, не связанных с внешним миром). Внутри пораженного района нам не удастся уменьшить количество заболевших, поскольку для большинства болезней у нас есть период, в котором болезнь еще не заметна, но уже заразна.

Задания

Самостоятельно усовершенствуйте модель, чтобы изучить влияние таких параметров:

1. Влияние плотности расселения (например, в результате увеличения площади).
2. Появление лечения — как варианта, при котором время болезни и смертность сокращаются.
3. Как будет развиваться ситуация, если приобретенный иммунитет окажется временным?
4. Предположим, что мы провели вакцинацию — то есть примерно половина людей не может заболеть (на какой-то промежуток времени). Как это повлияет на распространение болезни во время конкретной эпидемии? В долгосрочной перспективе?

Системно-динамическое моделирование

При организации работы компании сотовой связи в крупном городе один из существенных вопросов — сроки и условия наращивания мощности базовых станций сети. От этого показателя напрямую зависит качество обслуживания клиентов.

Если компания не предпринимает никаких особых усилий по привлечению клиентов, то их приток зависит от количества тех, кто уже обслуживается компанией, — сюда войдут и стандартные затраты на рекламу (фиксированный процент выручки с клиента), и те, кто придет по рекомендации.

Модернизировать станции слишком часто нельзя, это приведет к большим расходам, а оборудование не успеет окупиться.

Если мощность недостаточна, появляются недовольные клиенты. Недовольные компанией клиенты отказываются от ее услуг. Некоторое количество недовольных есть всегда, но если качество работы падает — их становится больше.

Выяснив, что мощность недостаточна, мы начинаем модернизацию — но она тоже потребует времени, а результат далеко не сразу окажет влияние на мнение (и динамику) клиентов.

Основным критерием успешности работы для нас будет отсутствие падения общего количества клиентов, то есть такая организация дела, при которой приток клиентов больше, чем потеря. Основная единица времени для нас в этих подсчетах — неделя.

Вопрос: какое время задержки допустимо? Насколько нужно повышать мощность оборудования?

Структура сети, поведение конкретных клиентов, способ заказа рекламы — не могут оказывать принципиального влияния на ситуацию, поскольку *никак не влияют на взаимозависимости*. Нас фактически интересуют статистические параметры. Даже модернизация никогда не даст скачкообразного эффекта — потому что никогда не будет выполняться разом во всей сети.

Как рассказывается в учебнике [5], для изучения влияния параметров на сложный процесс в крупных системах, как правило, нет смысла (а то и возможности) моделировать точное поведение каждого участника процесса.

Один из наиболее мощных инструментов для решения задач такого рода — метод, получивший название “Системно-динамическое моделирование” ([2], [3]). Его мы и применим.

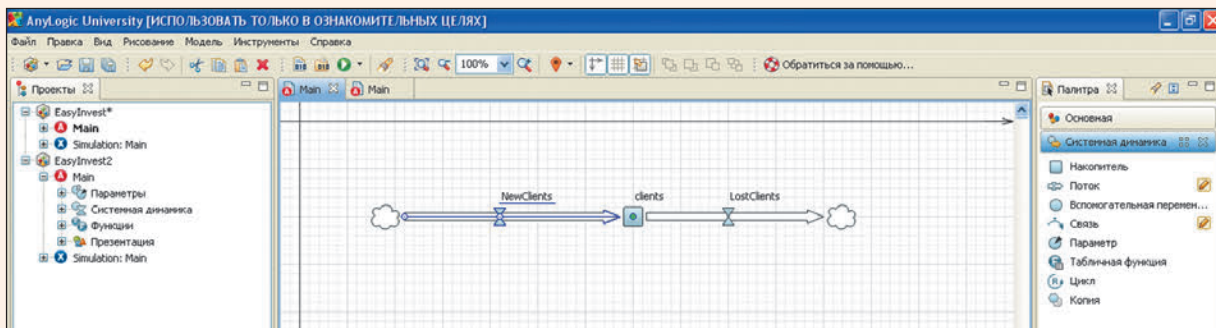
Создадим новую чистую модель — не используя шаблон. Для создания модели воспользуемся палитрой “Системная динамика”. Основой нашей модели будет поток клиентов, который мы будем накапливать в компании.

- Поместим на рабочее поле модели объект “Накопитель” и назовем этот объект Clients. Изначально (по умолчанию) у нас будет 100 условных клиентов.

- Разместим два объекта “Поток”: приток клиентов и их потерю. Приток — NewClients, отток — LostClients.

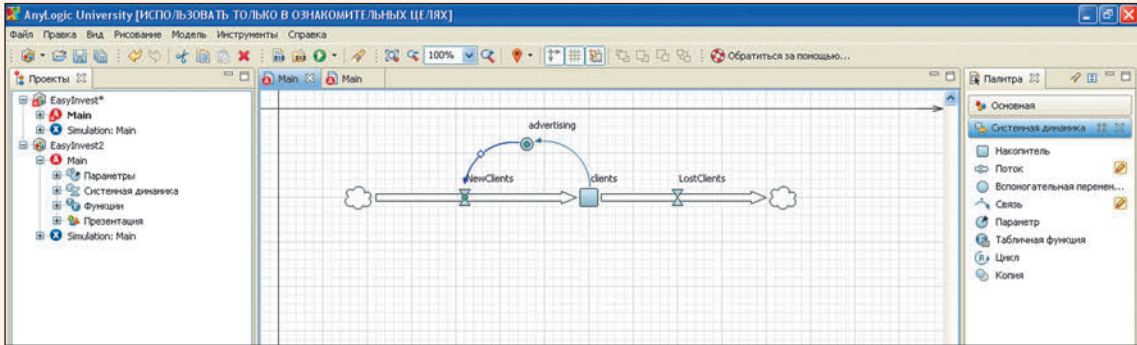
- Конечную точку притока и конечную точку оттока поместим в накопитель — так, чтобы соединитель принял вид, как на иллюстрации.

Обратите внимание — фактически мы предполагаем, что у нас есть бесконечный источник клиентов. В реальных условиях это, конечно, несколько не так.



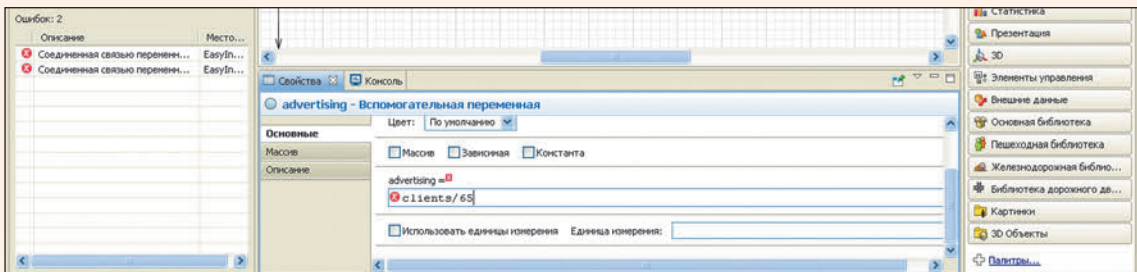
На приток влияет реклама. Мы не изучаем никаких ее параметров, поэтому сведем все к одной вспомогательной переменной. Будем считать, что наш основной способ рекламы — “Сарафанное радио”.

- Поместим на рабочее поле переменную, назовем ее advertising.
- Эту переменную мы свяжем с клиентской базой и притоком новых клиентов с помощью связей — которые тоже перетащим из палитры. Имена связей давать не нужно.

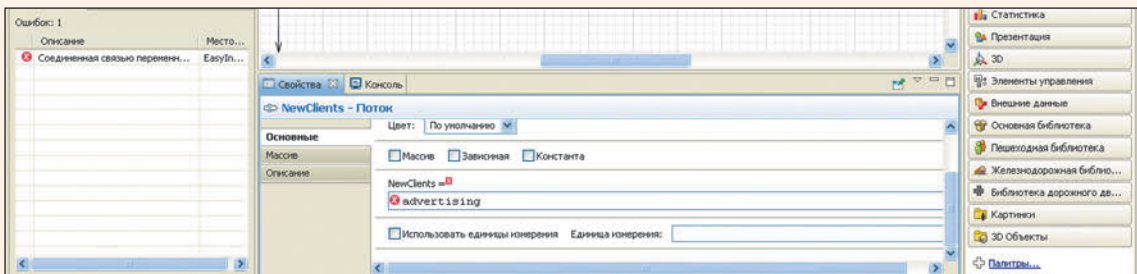


При этом возникнут два предупреждения: о том, что связь есть, но формулы для ее учета не заданы.

- Зададим значение этой переменной как формулу: $clients/65$



- Таким же способом зададим значение притока новых клиентов:



Сообщения об ошибках исчезнут.

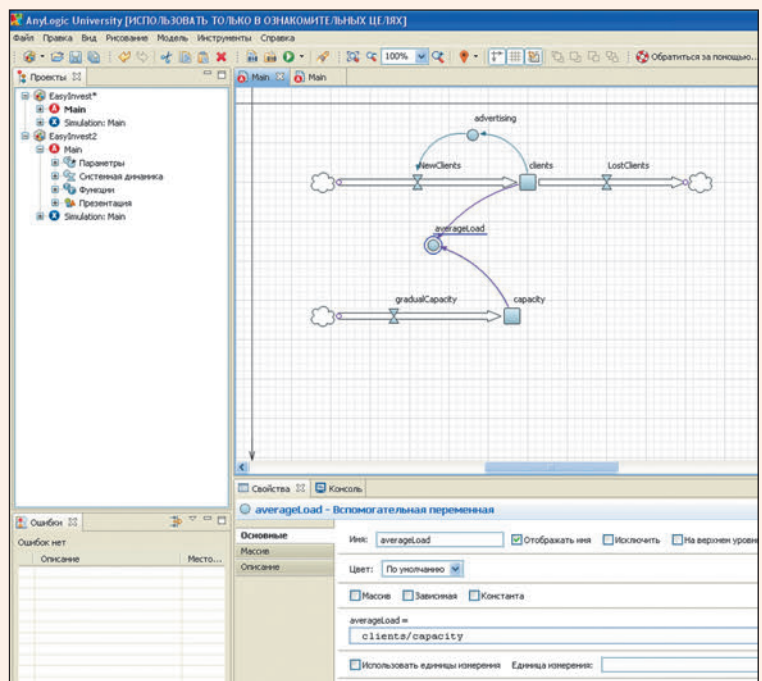
Теперь перейдем к самому наращивающей мощности. Создадим вторую линию:

- Зададим имеющуюся мощность накопителем, который назовем saracity — вместимость. Изначальная мощность сети — 60 условных клиентов.
- Мощность будет наращиваться потоком gradualCapacity — из бесконечного источника. То есть мощность мы наращиваем в зависимости только от наших решений.

Имеющаяся мощность по обработке обращений пользователей будет влиять на среднюю загрузку системы, которую мы вычислим как отношение мощности к количеству клиентов:

$$clients/saracity.$$

- Добавим промежуточную переменную, свяжем ее с имеющимися параметрами и зададим формулу.



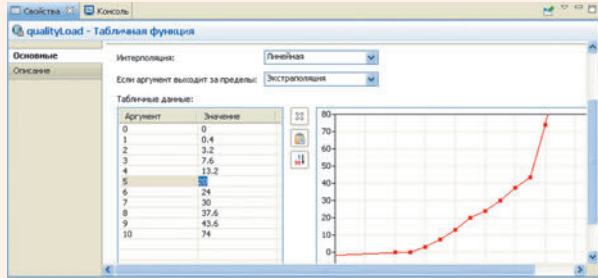
Теперь опишем соотношение качества с нагрузкой.

Качество обслуживания для клиентов — это нагрузка, но не текущая, а ранее работавшая, создавшая впечатление у клиентов.

При этом нам надо будет учесть, что зависимость эта не линейная и заданная некоторыми статистическими наблюдениями.

Для этого мы создадим специальную функцию, в которой зададим зависимость не формулой, а набором значений:

- Перетащим для этого из палитры компонент “Табличную функцию”, назовем ее `qualityLoad` и зададим следующие параметры:



- Создадим еще одну промежуточную переменную — `quality`, и формула для этой переменной будет такой: `qualityLoad(delay(averageLoad,10))`.

То есть оценка качества в виде количества “уходящих” клиентов будет даваться с задержкой на 10 единиц времени. Переменную мы свяжем с переменной средней загрузки и потоком уходящих клиентов. В “вентиле” уходящих клиентов мы просто напишем ее название.

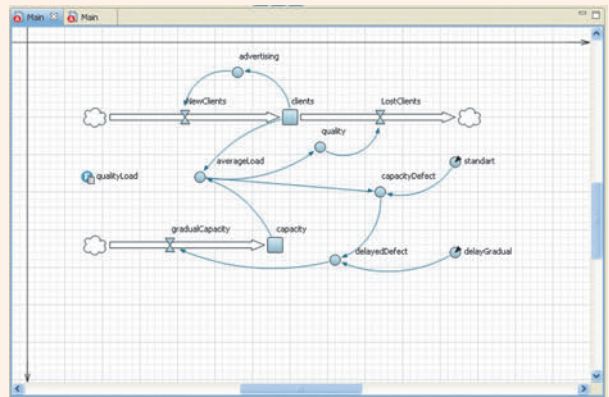
- Для оценки и принятия решения о модернизации мы введем параметр `standart` — задающий нагрузку, при которой необходимо принять решение о модернизации. По умолчанию примем 1,7 — то есть примерно тогда, когда у нас за каждый цикл начнут уходить два клиента.

- Параметр мы свяжем с текущей нагрузкой через промежуточную переменную `defectCapacity` (недостаток мощности), вычислив ее как разницу между загрузкой и стандартом.

Влиять на увеличение пропускной способности она будет тоже с задержкой:

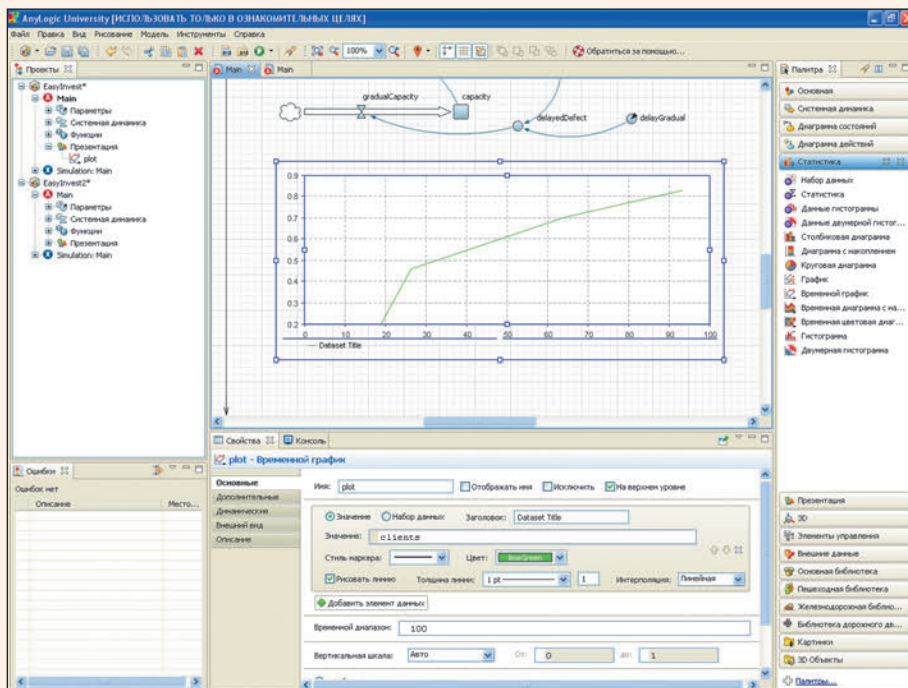
- Создадим параметр `delayGradual`. По умолчанию примем ее за 30 целых единиц.
- Создадим промежуточную переменную для роста пропускной способности с задержкой `delayedDefect`, определим ее формулой: `delay(capacityDefect,delayGradual)`.
- Свяжем ее с текущим дефектом и потоком модернизации.

Итоговая для нас схема:

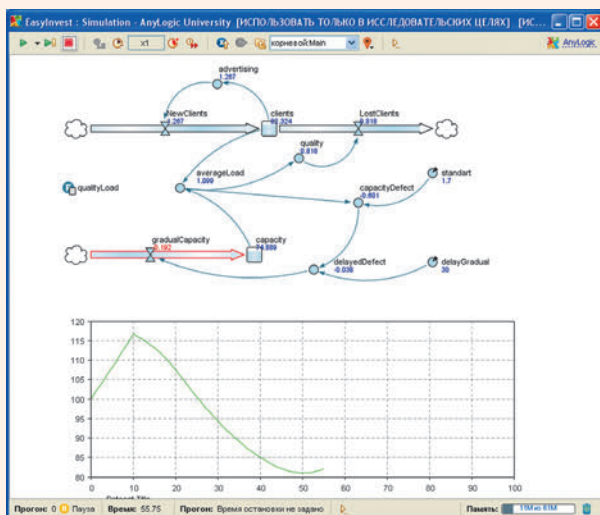


- На переменную `delayDefect` мы сошлемся в “вентиле” потока `gradualCapacity` — формулой `delayedDefect*5`, — поскольку, очевидно, будем наращивать мощность с запасом.

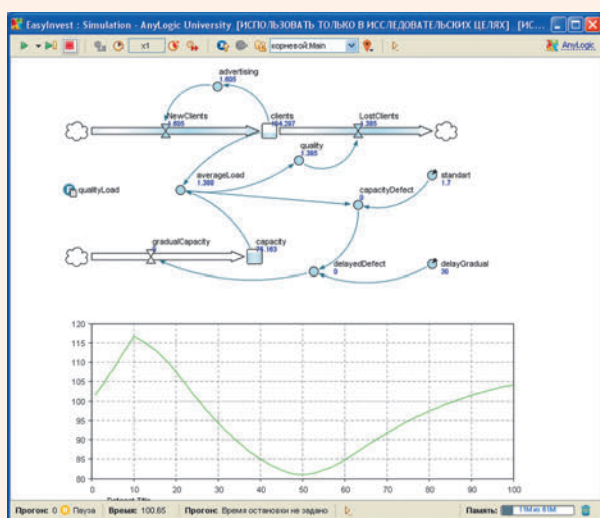
Чтобы отслеживать результаты работы, мы добавим на лист временной график, на котором отразим изменение количества клиентов со временем. Для этого достаточно в графике добавить одну серию данных, связанную с переменной `clients`.



Как ведет себя система при наших значениях по умолчанию? Запустим ее и обнаружим неприятную деталь:



Как только у нас начинает расти пропускная способность, вентиль “увеличения” начинает ее уменьшать. Очевидно, портить нашу сеть мы не будем. Изменим формулу вычисления дефекта на следующую: $\max(\text{averageLoad} - \text{standart}, 0)$. То есть не расти она может, но уменьшаться не должна. Повторим прогон.



Изначально все неплохо. За два месяца количество клиентов возрастает примерно на 17%. Но потом происходит следующее: через десять недель клиенты обнаруживают, что сети на них явно не хватает, и начинают “голосовать ногами”. Несмотря на то что через 30 недель мы начинаем модернизацию, ситуация не улучшается до 50 недель. К этому моменту мы теряем 17% к изначальному количеству, и ситуация исправляется только к 85-й неделе, то есть более чем через год.

Очевидно, что с таким подходом к своей сети компания вряд ли обеспечит прибыль.

Задания

1. Изучите влияние имеющихся параметров на систему, добейтесь улучшения ситуации.

2. Почему мы не предлагаем варьировать в модели “рекламную мощность”, время задержки реакции клиентов?

3. Какой еще параметр можно добавить для управления системой — не меняя ее основной структуры?

Поиск оптимального решения

Любой магазин, любая организация, занимающаяся работой с потоком клиентов — будь то банковский офис или отделение единого информационно-расчетного центра, — рано или поздно сталкиваются с проблемой расчета необходимого количества сотрудников (кассиров, операционистов). Число работников при этом, с одной стороны, не должно быть очень маленьким, чтобы не собирать огромные очереди недовольных клиентов, но, с другой стороны, не должно быть и слишком большим, иначе организация понесет лишние расходы на зарплату. Расчет оптимального количества сотрудников, которое будет зависеть от интенсивности потока клиентов, среднего времени обслуживания и других параметров, является примером решения **задачи оптимизации**.

Задачу эту можно решить, подготовив и решив систему уравнений, однако учет всех параметров и связей даже такой простой системы, как описанная выше, может потребовать введения в модель большого количества переменных, что неизбежно приведет в итоге к достаточно громоздким вычислениям.

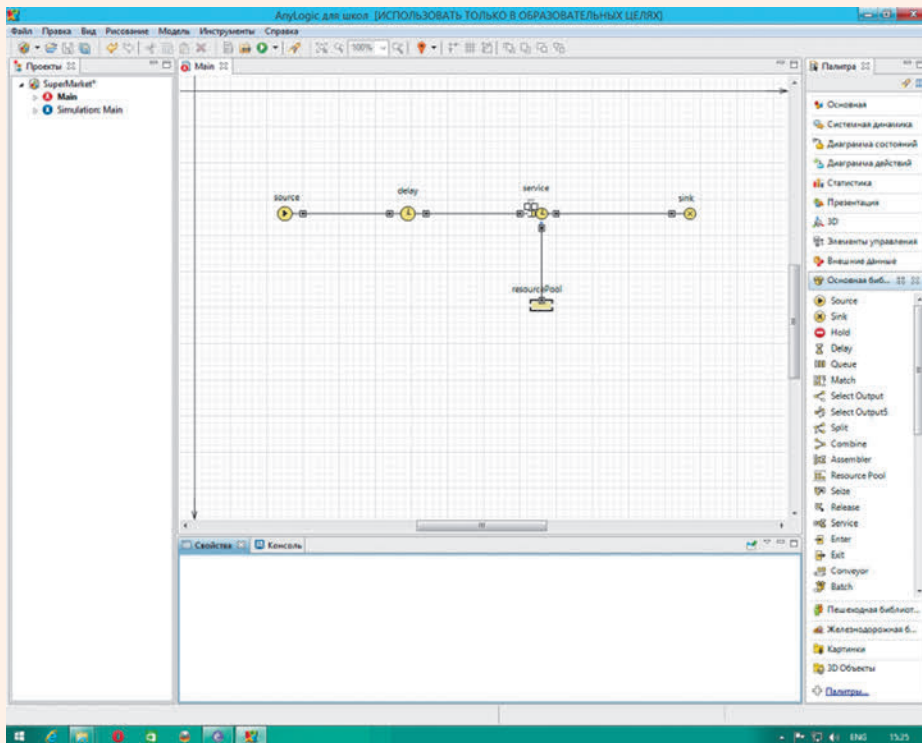
Куда более рациональным будет в этом случае прибегнуть к средствам компьютерного моделирования, позволяющим вместо сложных и не всегда возможных аналитических преобразований свести все к рутинным расчетам, то есть опереться на вычислительные мощности компьютера.

Пользователю в этом случае остается лишь задать объекты, параметры и связи, входящие в моделируемую систему, а также правильно интерпретировать полученные на выходе результаты. Рассмотрим пример решения подобной задачи в среде AnyLogic.

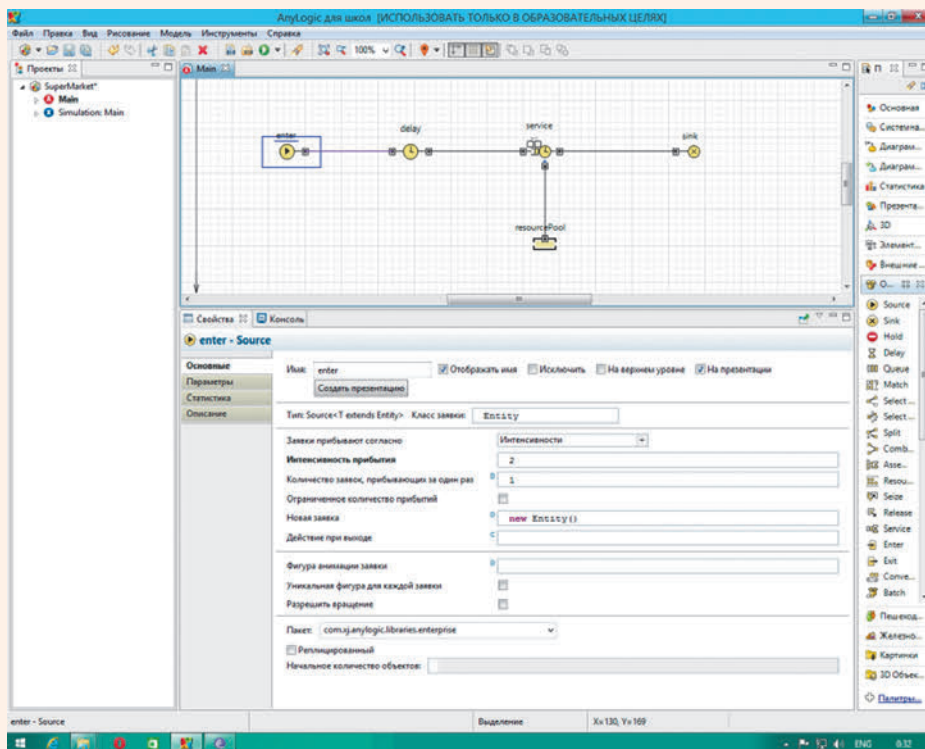
1. Создание дискретно-событийной модели работы касс магазина

Для создания модели магазина самообслуживания нам понадобятся следующие элементы из блока “Основная библиотека”: source (источник заявок, т.е. посетителей магазина), delay (задержка заявок, связанная с тем, что посетителю требуется какое-то время на выбор покупок), service (обслуживание на кассовой линии), resourcePool (элемент, отвечающий за количество кассиров) и sink (уничтожение заявок).

Последовательно перетащим элементы из блока “Палитра” в рабочее окно модели и соединим их линиями так, как показано ниже.



Следующим этапом создания модели будет изменение свойств каждого из добавленных элементов.
 1. Элемент source.



- Так как данный элемент отвечает за появление заявок (покупателей в магазине), переименуем его в enter (вход).

- Очевидно, что в разное время суток покупатели заходят с разной интенсивностью, но пока для простоты будем считать, что частота появления в магазине новых клиентов все время одинакова и равна двум человекам в одну минуту.

2. Элемент delay (см. с. 16).

- Переименуем его в shopping, так как этот элемент отвечает за время, требующееся зашедшему в магазин человеку на выбор покупок.

• Для того чтобы выставить это время, используем функцию треугольного распределения `triangular`. Будем считать, что на процедуру выбора потребуется минимум пять минут, максимум — один час, а наиболее вероятным укажем время в двадцать минут.

• Поле *максимальная вместимость* отметим галочкой, так как в условиях нашей модели будем считать магазин способным вместить сколько угодно покупателей.

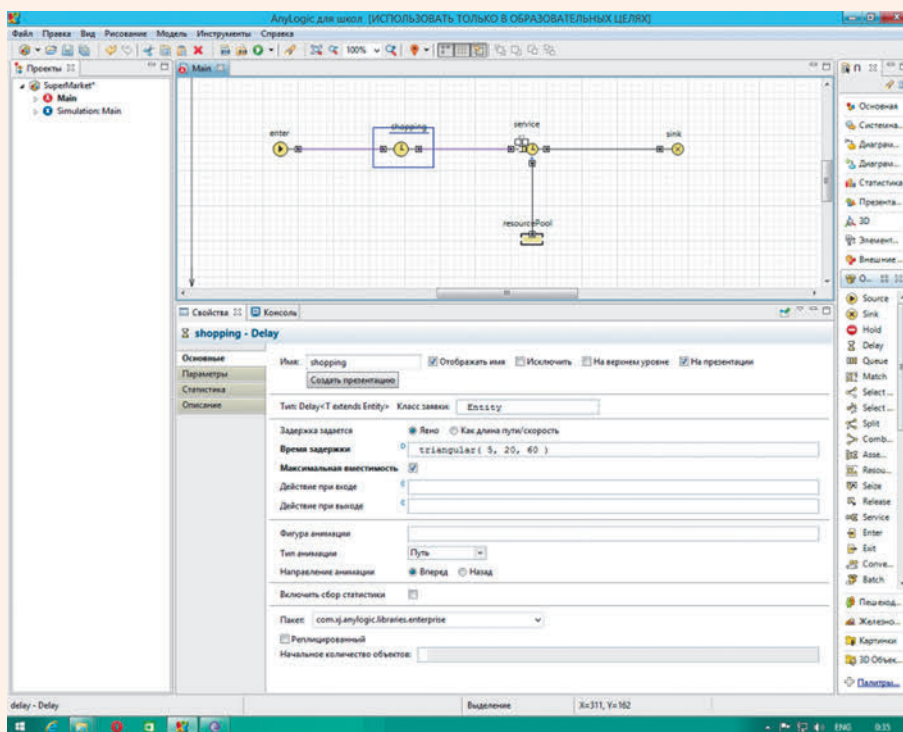
3. Элемент `service`.

• Переименуем данный элемент в `queue` (очередь).

• В поле *количество ресурсов* оставим значение, равное единице, так как на обслуживание одного покупателя обычно требуется лишь один кассир (один ресурс).

• *Время задержки*, т.е. время, необходимое посетителю для оплаты покупок в кассе, укажем также при помощи треугольного распределения `triangular(0.5, 1, 5)`.

• Поле *максимальная вместимость очереди* отметим галочкой, так как в процессе последующего эксперимента при определенных параметрах очередь может оказаться очень большой.



4. Элемент `resourcePool`.

• Переименуем в `cashiers` (кассиры).

• *Количество ресурсов* (в нашем случае количество кассиров) укажем равным четырем. Впоследствии это число мы изменим на параметр, который будет изменяться в зависимости от интенсивности потока покупателей.

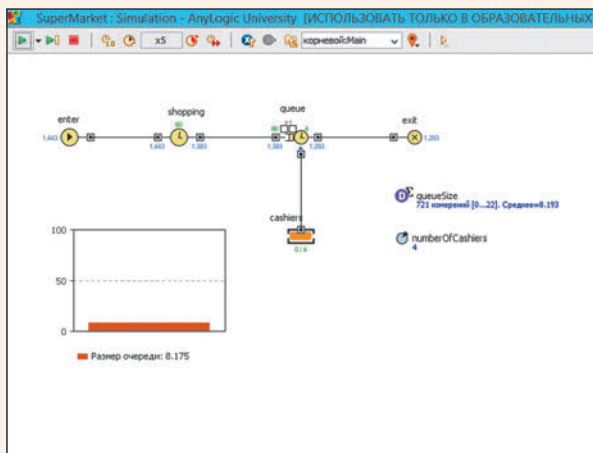
5. Элемент `sink` просто переименуем в `exit` (выход).

Запустим получившуюся модель. Зеленые цифры над элементами `shopping` и `queue` показывают, сколько человек в данный момент выбирают товар, а сколько уже стоят в очередях к кассам. Однако данное отображение не является для нас особенно информативным. Гораздо важнее для нас будет не текущий размер очередей, а их усредненное по времени значение. Для того чтобы собрать такую информацию, необходимо добавить в рабочее окно программы элемент *Статистика*, переименовав его в `queueSize`. В поле *значение* данного элемента пропишем формулу `queue.queueSize()/4`, где цифра четыре означает количество кассиров, так как необходимо одну большую очередь к кассам разделить на четыре равных по величине очереди. В свойствах `queue` разрешим сбор статистики, поставив соответствующую галочку.

Теперь мы можем наглядно отобразить средний размер очереди в кассу при помощи столбиковой диаграммы. Перетащим соответствующий элемент в рабочую область. В свойствах щелкнем на *добавить элемент данных* и зададим в поле *значение* формулу `queueSize.mean()`. Масштаб диаграммы выберите на свое усмотрение.

Запустив снова симуляцию модели, увидим на диаграмме, что средняя очередь постепенно растет и за 720 минут (12 часов) достигает величины порядка восьми человек (см. с. 17). Если же изменить интенсивность прибытия покупателей с двух человек в минуту на один, то увидим, что размер очереди с течением

нием времени не возрастает и по величине близок к нулю. Очевидно, что для такой интенсивности потока клиентов четыре кассира — это слишком много.



Дополнительные задания к части 1

1. Построенная модель не ограничена по времени, в то время как большинство магазинов работают не круглосуточно, а с утра и до вечера. Попробуйте ограничить время работы модели одним днем с 10 утра до 10 вечера. Делается это во вкладке *Модельное время* в свойствах объекта *Simulation*.

2. В нашей модели интенсивность прибытия покупателей в магазин есть величина постоянная. Но в реальной жизни интенсивность зависит от времени суток (утром покупателей очень мало, а вечером, наоборот, много) и дня недели (будни или выходные). Попробуйте задать интенсивность, зависящую от расписания. Для этого перетащите в рабочее окно элемент *Расписание* вкладки *Основная*, задайте в его свойствах нужное расписание. Затем в свойствах объекта *Enter* укажите, что заявки должны прибывать согласно *интенсивности (по расписанию)*, и сошлитесь на объект *Расписание*.

2. Эксперимент по оптимизации

Само собой, ставить четырех кассиров в магазин, где спокойно управляется с работой и меньшее число человек, никто не будет, так как это приведет к излишним затратам на заработную плату. Однако чрезмерная экономия на количестве работ-

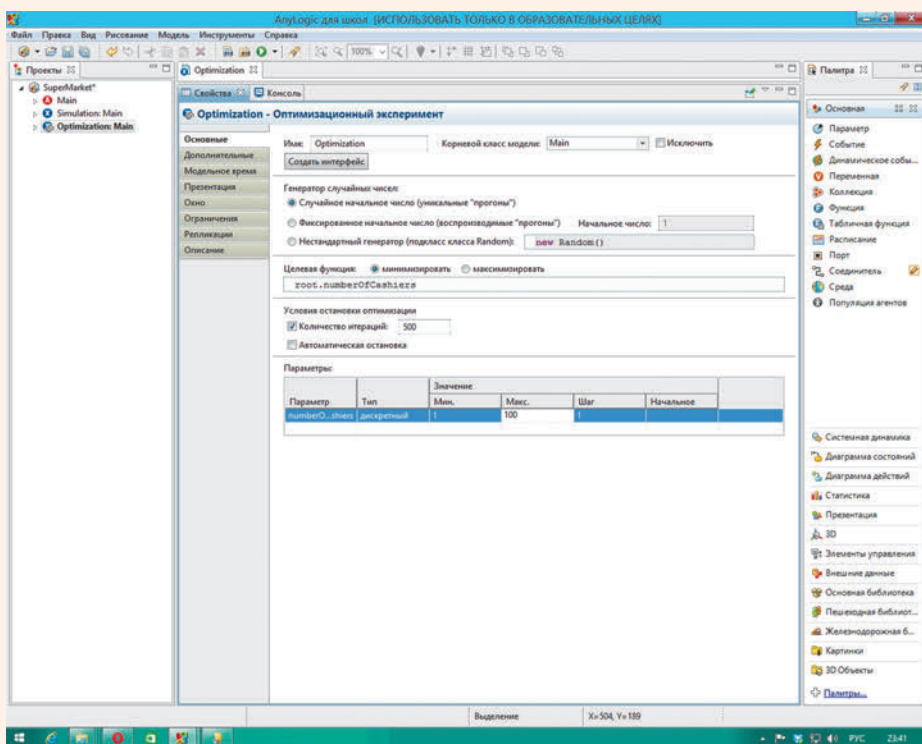
ников повлечет за собой образование больших очередей, что может негативно отразиться на репутации магазина. Разумнее всего в этом случае найти оптимальное решение, т.е. минимальное количество кассиров, при котором средняя очередь не будет превышать заданное значение. Используем для этого *Оптимизационный эксперимент*, встроенный в AnyLogic.

Наша цель — провести большое количество прогонов модели, определить статистически оптимальное значение этого показателя. На основании полученных при этом данных специальный алгоритм *OptQuest*, один из лучших на сегодняшний день, определит оптимальное для этой модели решение. Алгоритм этот сложен, в нем используются методы математической оптимизации, и нейронные сети, и эвристики поиска решения. Алгоритм запатентован, и его содержание правообладатель не публикует.

Конечно, сами прогоны, в отличие от наших экспериментов, показываться не будут — нас будет интересовать только результат, полученный в каждом из них.

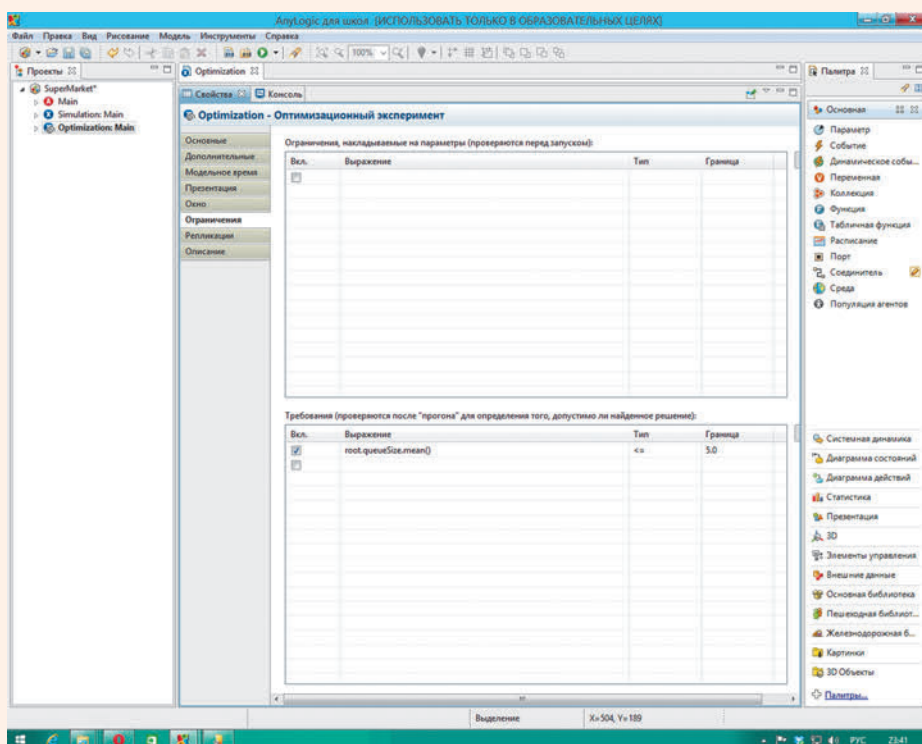
Для начала добавим в рабочее окно модели параметр, перетащив его из палитры *Основная*. Переименуем его в *numberOfCashiers*. Зададим целочисленный тип (*integer*) и значение по умолчанию, равное четырем. Этот параметр в дальнейшем мы и будем оптимизировать. В свойствах объекта *cashiers* в поле *количество ресурсов* заменим число четыре на *numberOfCashiers*.

Можно приступить к созданию нового эксперимента. Для этого необходимо щелкнуть правой кнопкой в левом окне на названии модели и выбрать *Создать — Эксперимент — Оптимизация*.



Для проведения эксперимента необходимо указать целевую функцию, которую мы хотим минимизировать или максимизировать. В нашем случае это будет минимизация добавленного только что параметра `numberOfCashiers`. Поэтому на вкладке *Основные* свойств эксперимента укажем, что в качестве целевой функции мы хотим видеть параметр `root.numberOfCashiers`. Указатель `root` необходим, так как параметр расположен не в окне оптимизационного эксперимента, а в активном классе модели. Чуть ниже зададим дискретное изменение данного параметра от 1 до 100 с шагом 1 (см. рис. на с. 17). Это означает, что эксперимент будет выполняться снова и снова, но количество кассиров будет меняться от одного до двадцати, и для каждого значения параметра `numberOfCashiers` будет измеряться среднее значение очереди.

Суть нашего эксперимента заключается в том, что необходимо найти то минимально допустимое количество работников, при котором средняя очередь в кассу не превысит заданного нами значения. Величина, которой мы ограничиваем очередь, в разных магазинах может быть очень и очень разной и определяется многими факторами, как, например, ценовой категорией магазина (вряд ли клиент дорогого магазина захочет стоять в длинной очереди) или наличием свободного пространства (при его отсутствии длинные очереди могут блокировать проход покупателей, еще не выбравших товар). Так или иначе, но перед запуском оптимизационного эксперимента мы должны точно знать, очереди какой длины являются для нашего магазина предельно допустимыми. Для определенности предположим, что число это будет равно пяти. Перейдем во вкладку *Ограничения* и потребуем, чтобы значение `root.queueSize.mean()` в процессе эксперимента не превышало пяти:



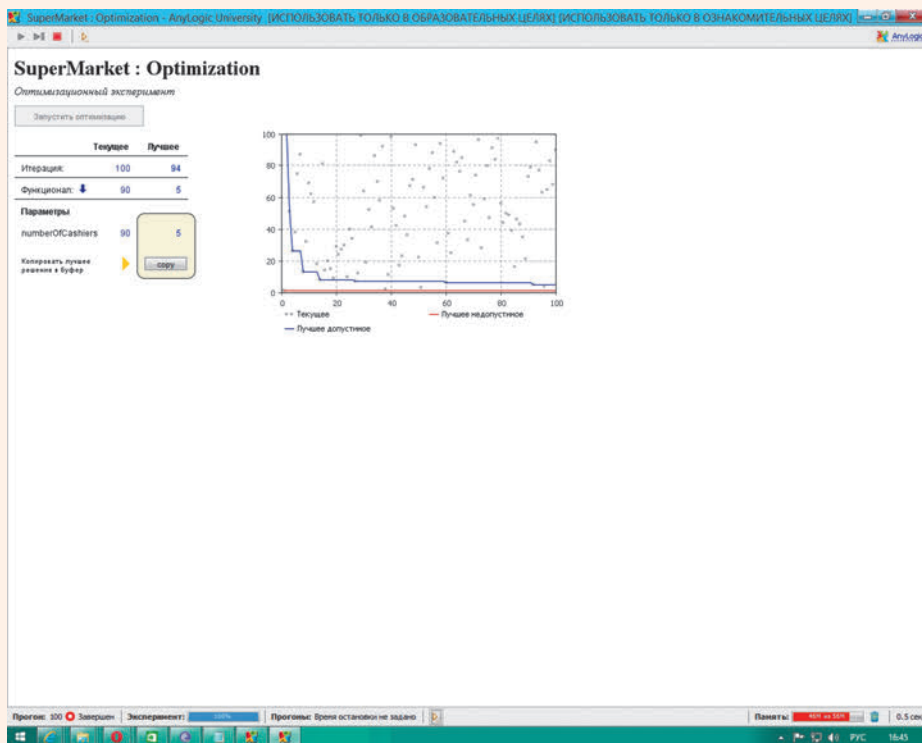
Большая часть магазинов работает с 10 утра до 10 вечера. Поэтому мы ограничим время выполнения эксперимента 12 часами (720 минут). Для этого перейдем на вкладку *Модельное время* и укажем *начальное время* — 0.0, а *конечное время* — 720.0.

После этого возвращаемся во вкладку *Основные* и нажимаем на кнопку *Создать интерфейс*.

Обратите внимание, что теперь, если щелкнуть на кнопке запуска модели, для запуска будут доступны два эксперимента. Один из них — *Simulation* — простой эксперимент по симуляции модели, который мы запускали ранее. Второй же эксперимент и есть оптимизационный. Запустим его.

Перед нами откроется новая форма, в которой щелкнем на кнопку *Запустить оптимизацию*. На с. 19 показан результат выполнения такого эксперимента. На графике справа отображена зависимость значений оптимизируемого параметра от номера итерации (одна итерация — одно выполнение эксперимента при определенных значениях “перебираемого” параметра `numberOfCashiers`). Причем красным отображено лучшее недопустимое значение, т.е. значение, полученное без учета ограничений, наложенных на оптимизируемую модель. А синим — лучшее допустимое. Мы видим, что с увеличением количества итераций значение оптимизируемого параметра стремится к пяти.

Это же значение показано в желтом окошке, слева от графика. Следовательно, для заданных нами начальных условий наименьшее возможное количество работников будет равным пяти.



Дополнительное задание к части 2

В процессе оптимизации может возникнуть необходимость в многократном изменении входных данных (у нас это интенсивность потока клиентов). При этом каждый раз закрывать эксперимент и изменять значение интенсивности вручную — достаточно долгая и неудобная для пользователя процедура. Гораздо проще было бы, если бы в окне эксперимента располагался ползунок, регулирующий входной параметр. Для того чтобы добавить его, перетащите ползунок из палитры *Элементы управления* и самостоятельно настройте его на управление потоком клиентов.

Литература

1. Качала В.В. Основы теории систем и системного анализа. М.: Горячая линия-Телеком, 2007.
2. Форрестер Дж. Основы кибернетики предприятия (индустриальная динамика). М.: Прогресс, 1971.
3. Шеннон Р. Имитационное моделирование систем — искусство и наука. М.: Мир, 1978.
4. Медоуз Д.Х. Азбука системного мышления. М.: Бином. Лаборатория знаний, 2010.
5. Калинин И.А., Самылкина Н.Н. Информатика. 10-й класс. Углубленный уровень. Учебник. М.: Бином, 2013.
6. Калинин И.А., Самылкина Н.Н. Основы имитационного моделирования. М.: “Первое сентября: информатика”, № 5, 2013.



Тренинг по информатике: “разбор полетов”

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

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

► И для 9-х, и для 11-х классов время с поздней осени по раннюю весну — это своего рода “сезон катастроф”, именуемых тренировочными и диагностическими работами в формате ОГЭ и ЕГЭ ☺. Впрочем, и польза от таких тренировок весьма существенна: они не только дают возможность проверить знания школьников, но и подсказывают учителю, “куда двигаться дальше” в плане подготовки детей к настоящему экзамену — на чем заострить внимание, если какие-то уже разобранные ранее задачи “не удаются”, а какие вновь встретившиеся задания — изучить подробно. А заодно и отточить мастерство в решении задач, искать все более оптимальные методы решения, более легкие и

экономные по времени, которого на экзамене, как потом выясняется, так мало...

Ниже представлены некоторые методические выводы и примеры решения некоторых заданий ноябрьской тренировочной работы по информатике для 11-х классов, которые вызвали у школьников трудности. Правда, поскольку МИОО (разработчик заданий) запрещает публикацию своих материалов, разбирать мы будем решение своих собственных заданий, аналогичных по смыслу тренировочным.

1. Снова Фано

Задачи на пресловутое “условие Фано”, определяющее системы кодов, обеспечивающие однозначную расшифровку закодированных сообщений, давно уже “прописались” в экзаменационных задачах. Не стал исключением и данный тренинг.

Для кодирования последовательности символов, состоящей из букв

К, И, Н, О, используется неравномерный двоичный код, удовлетворяющий условию Фано. При этом для буквы К использован код 0, а для буквы И — код 11. Требуется определить наименьшую возможную суммарную длину всех кодовых слов указанных букв.

- 1) 8; 3) 10;
2) 9; 4) 11.

Решение

Напомним, в чем заключается условие Фано.

Закодированное сообщение можно однозначно декодировать с начала, если **никакое кодовое слово не является началом другого кодового слова.**

Для проверки на соответствие кодов условию Фано нужно попарно сравнивать между собой коды по следующим правилам:

- когда длина обоих сравниваемых кодов совпадает, проверяется равенство этих кодов: если один код совпадает с другим, то такая пара кодов неправильна (не удовлетворяет условию Фано);
- когда длина сравниваемых кодов различна, более короткий код записывается под более длинным с выравниванием обоих кодов по левому краю: если все знаки более короткого кода совпадают с соответствующими знаками в начале более длинного кода, то такая пара кодов неправильна (не удовлетворяет условию Фано).

Опираясь на эти правила, будем подбирать коды для оставшихся букв Н и О. Начнем с буквы Н и будем перебирать возможные двоичные числа с возрастающей длиной (ведь нам важно получить наименьшую возможную суммарную длину кодов!).

Код буквы К	Код буквы И	Предполагаемый код буквы Н	Комментарий
0	11	1	Нельзя, так как совпадает с началом кода буквы И
		00	Нельзя — код буквы К совпадает с его началом
		01	Нельзя — код буквы К совпадает с его началом
		10	Допустимый код (не совпадает с двузначным кодом буквы И, а код буквы К не совпадает с его началом)

Итак, можно предположить, что первый код найден. Но посмотрим — удастся ли при этом найти код для оставшейся четвертой буквы О. При этом

можно сразу отбросить те коды, которые не подошли для буквы Н, — ведь код буквы О должен удовлетворять тем же требованиям при сравнении с кодами букв К и И.

Код буквы К	Код буквы И	Код буквы Н	Предполагаемый код буквы О	Комментарий
0	11	10	11	Нельзя — совпадает с кодом буквы И
			000, 001, 010, 011	Нельзя — код буквы К совпадает с его началом
			100, 101	Нельзя — код буквы Н совпадает с его началом
			110, 111	Нельзя — код буквы Н совпадает с его началом

Очевидно, и дальше с увеличением числа разрядов в предполагаемом коде буквы О будет сохраняться та же тенденция: ни один код не пригоден. Как же быть?

Причина этой прискорбной ситуации в том, что, выбрав для буквы Н код 10, мы “закрыли” для себя возможность дальнейшего расширения нашей кодовой системы. Поэтому вместо кода 10 нам придется выбрать для буквы Н более длинный код, — например, 100.

А теперь повторим попытку поиска кода для буквы О:

Код буквы К	Код буквы И	Код буквы Н	Предполагаемый код буквы О	Комментарий
0	11	100	101	Допустимый код (не совпадает с трехзначным кодом буквы Н, а коды букв К и И не совпадают с его началом)

Таким образом, решение найдено. Выпишем коды всех четырех букв:

К	И	Н	О
0	11	100	101

Подсчитаем суммарную длину этих кодов (в знаках): $1 + 2 + 3 + 3 = 9$.

Ответ: 2).

2. Крупинки истинности ☺

Для таблицы истинности функции F известны значения нескольких ячеек:

x_1	x_2	x_3	x_4	F
	1	0		1
0	1			1
1	0			0
	0		1	0

Каким выражением может быть F ?

- 1) $x_1 \wedge x_2 \wedge x_3 \wedge \neg x_4$
- 2) $x_1 \vee x_2 \vee x_3 \vee \neg x_4$
- 3) $\neg x_1 \wedge x_2 \wedge \neg x_3 \wedge x_4$
- 4) $\neg x_1 \vee x_2 \vee \neg x_3 \vee x_4$

Решение

Смысл решения — по очереди “примерять” каждый из предложенных вариантов ответа к таблице истинности и смотреть: если при указанных в ней значениях переменных хоть в какой-нибудь строке **не получается** требуемое значение F , то данное значение не подходит. Если же для всех строк таблицы истинности и всех имеющихся в ней значений переменных требуемое значение F получается, то мы считаем, что такое логическое выражение *может* соответствовать этой таблице и, соответственно, является решением задачи.

Но можно ускорить решение. Вспомним: для логической операции И “критическим” значением является ноль: если **хотя бы одна** переменная равна нулю, то и все выражение тоже равно нулю. Аналогично, для логической операции ИЛИ “критическим” значением является единица: если **хотя бы одна** переменная равна единице, то и все выражение тоже равно единице. (Разумеется, нужно не забывать и заменять ноль на единицу и единицу на ноль при операции отрицания.)

Поэтому нам достаточно при проверке вариантов ответа проверять только часть строк таблицы истинности:

- если логическое выражение составлено из операций И, то прежде всего проверяем строки таблицы истинности, в которых $F = 1$: если хоть в одной ячейке, соответствующей “обычной” переменной, записан 0, а в ячейке, соответствующей переменной с НЕ, записана 1, то такой вариант ответа можно отбросить сразу;
- наоборот, если логическое выражение составлено из операций ИЛИ, то прежде всего проверяем

строки таблицы истинности, в которых $F = 0$: если хоть в одной ячейке, соответствующей “обычной” переменной, записана 1, а в ячейке, соответствующей переменной с НЕ, записан 0, то такой вариант ответа тоже можно отбросить сразу.

Итак, предполагаемая “кандидатура” на правильный ответ — выражение $\neg x_1 \wedge x_2 \wedge \neg x_3 \wedge x_4$. Проверяем его и для оставшихся строк с $F = 0$, убеждаемся, что им оно тоже соответствует (за счет того, что $x_2 = 0$).

Итого ответ может быть в наихудшем случае найден за 10 “проверок” вместо 16 при обычном решении.

Ответ: 3).

3. Сколько единиц?

Определите наименьшее пятизначное восьмеричное число, двоичная запись которого содержит три единицы. (В ответе записать только само число, без индекса системы счисления.)

Решение

На первый взгляд кажется, что все просто: чем правее в двоичном числе записаны единицы, тем это число меньше¹. Значит, наименьшим является число 111_2 .

Так-то оно так, — но нам нужно, чтобы число было в восьмеричном формате пятизначным. А двоичное число 111 , как легко видеть, соответствует восьмеричной семерке, т.е. является однозначным.

Как же найти наименьшее пятизначное восьмеричное число?

Единственный выход — оставляя как можно больше единичек справа, “отодвигать” только одну единицу влево, записывая после нее нули до тех пор, пока не получится такое двоичное число, которое после перевода в восьмеричную систему счисления будет иметь пять цифр. Причем такое число будет наименьшим из возможных, потому что если бы мы перемещали хоть одну из оставшихся справа единиц левее, то еще увеличивали бы наше число.

А сколько нулей дописывать? Вспомним, что каждой восьмеричной цифре соответствуют три двоичных цифры. Значит, нужно “оттаскивать” единицу влево, пока она только-только не “вылезет” в пятую триаду (будет в ней самым младшим битом), — см. схему на с. 23.

№	Выражение	Операция	Проверяем строки таблицы	Комментарий	Вывод
1)	$x_1 \wedge x_2 \wedge x_3 \wedge \neg x_4$	И	$c F = 1$	$x_3 = 0$ (строка 1)	Не годится
2)	$x_1 \vee x_2 \vee x_3 \vee \neg x_4$	ИЛИ	$c F = 0$	$x_1 = 1$ (строка 3)	Не годится
3)	$\neg x_1 \wedge x_2 \wedge \neg x_3 \wedge x_4$	И	$c F = 1$	все соответствует	
4)	$\neg x_1 \vee x_2 \vee \neg x_3 \vee x_4$	ИЛИ	$c F = 0$	$x_4 = 1$ (строка 4)	Не годится

¹ Вообще в любой системе счисления в наименьшем числе чем больше цифра, тем правее она должна стоять в записи числа (т.е. попадать в самые младшие разряды). И наоборот, в наибольшем числе самые большие цифры должны быть записаны ближе к левому краю (в старших разрядах). — Прим. авт.

$$\begin{array}{cccccc} 1 & 000 & 000 & 000 & 011_2 \\ \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} \\ 1 & 0 & 0 & 0 & 3_8 \end{array}$$

Ответ: 10003.

Примечание. Некоторые школьники допустили ошибку при верной общей идее решения — они сместили левую единицу так, чтобы получить слева целую триаду из трех двоичных разрядов (“100”), забыв, что можно дополнить триаду незначащими нулями слева. В результате у них получилось восьмеричное число 40003, которое не является минимально возможным.

4. “Эх, дороги...”

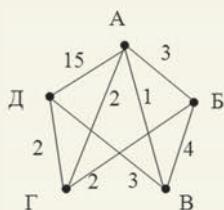
Между городами А, Б, В, Г, Д построены дороги, протяженность которых указана в таблице. Отсутствие числа в ячейке означает, что прямой дороги между соответствующими городами нет.

	А	Б	В	Г	Д
А		3	1	2	15
Б	3		4	2	
В	1	4			3
Г	2	2			2
Д	15		3	2	

Найти длину кратчайшего пути между пунктами А и Д, проходящего через пункт В, если передвигаться можно только по указанным дорогам.

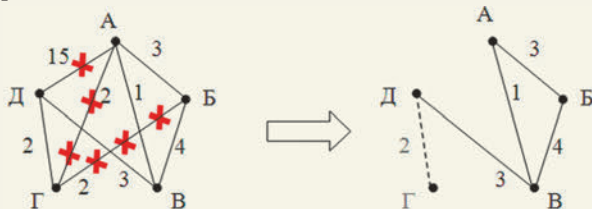
Решение

Как обычно, строим по таблице граф дорог:



Дальше нужно искать все возможные пути из А в Д, отбрасывая те, которые не проходят через город В, подсчитывать их длину и искать путь с минимальным значением его суммарной длины.

Можно ли упростить и ускорить решение? Да! Можно сразу исключить из графа пути “в обход” города В:



Получили совсем простой граф (тем более что в город Г теперь вообще ни одна дорога из А не идет, так что путь ГД тоже можно исключить).

В этом графе — только два возможных пути:

- АВВД с длиной $3 + 4 + 3 = 10$,
- АД с длиной $1 + 3 = 4$.

Причем то, что кратчайший из них — это путь АД, видно “невооруженным глазом”. Его длина 4 — и есть ответ.

Ответ: 4.

5. “Однорукий бандит”

Обычно так называют игровые автоматы, при помощи которых владельцы казино очищают карманы слишком азартных игроков. Но числовой автомат, о котором идет речь в задаче, наверное, тоже представляется ученикам не в самом радужном свете ☺.

Автомат получает на вход трехзначное число. По этому числу формируется новое число по следующим правилам.

1. Складываются первая и вторая, а затем — вторая и третья цифры исходного числа.

2. Полученные два числа записываются друг за другом подряд в порядке возрастания.

Пример. Исходное число: 176. Полученные числа: $1 + 7 = 8$, $7 + 6 = 13$. Результат: 813.

Найдите наибольшее исходное число, для которого автомат выдаст результат 815.

Решение

Сначала определим, как можно “разрезать” результирующее число на два, вычисленных автоматом.

Вспомним, что сумма двух десятичных цифр (которые могут быть равны от 0 до 9) может равняться от 0 ($0 + 0$) до 18 ($9 + 9$).

Если пытаться разделить число 815 как 81 и 5, то первое из этих чисел не соответствует этому возможному диапазону значений сумм цифр (да к тому же тогда числа записаны не по возрастанию). Поэтому остается только один вариант: число 815 как 8 и 15.

Теперь посмотрим, какие цифры могут давать такие суммы:

- $8 = 0 + 8, 1 + 7, 2 + 6, 3 + 5, 4 + 4$;
- $15 = 9 + 6, 8 + 7$.

А теперь — самое главное. По условию, одна сумма — это сумма первой и второй цифр исходного числа, а вторая — это сумма второй и третьей цифр. Вторая цифра, таким образом, должна повториться в обеих суммах.

Ищем такие две суммы в обеих “подборках” (для числа 8 и для числа 15).

Это пары: $0 + 8$ и $8 + 7$; $1 + 7$ и $8 + 7$; $9 + 6$ и $2 + 6$.

Им соответствуют числа (учитывая, что ноль не может быть записан самым первым — тогда он был бы незначащим, а число — двузначным): 780, 178, 871, 962, 269 (везде повторяющаяся цифра стоит в середине).

Остается найти среди них наибольшее. Очевидно, это число 962.

Ответ: 962.

6. “Формула-1” в Excel

Дан фрагмент электронной таблицы. Из ячейки С3 в одну из ячеек столбца D была скопирована

формула. После копирования значение этой формулы стало равным 100.

В какую ячейку была скопирована формула? В ответе надо записать только номер строки, в которой находится эта ячейка.

	A	B	C	D
1	5	13		
2	6	12		
3	7	11	=B3*A\$4	
4	8	10		

Решение

Сначала определим, как меняется формула при ее копировании в ячейки столбца D (с учетом абсолютной адресации):

	C	D
1		=B1*B\$4
2		=B2*B\$4
3	=B3*A\$4	=B3*B\$4
4		=B4*B\$4

Остается только вычислить произведения и сверить их с требуемым. Впрочем, и так сразу видно, что 100 — это $10 \cdot 10$, а подходящая формула — только в ячейке D4 ($=B4 \cdot B$4$).

Ответ: 4.

Примечание. Важно обратить внимание школьников на то, что нужно внимательно читать условие задачи — в том числе о том, как надо записать ответ. Запись ответа в виде “D4” при автоматической проверке ответов будет признана за ошибку!

7. Оцифровка аудио

Выполнена квадратура (четырёхканальная) звукозапись с частотой дискретизации 32 кГц и 16-битным разрешением. В результате получен файл размером 38 Мбайт, причем сжатие данных не производилось. Требуется приблизительно оценить, сколько времени (в минутах) производилась запись. В качестве ответа нужно указать ближайшее к полученному времени записи целое число минут.

Решение

Такие задачи школьникам уже знакомы. Их решение сводится к записи одного-единственного уравнения, — нужно только внимательно записать в него все составляющие:

- количество каналов записи — 4;
- частота — 32 000 колебаний в секунду;
- разрешение — 16 бит;
- длительность — неизвестна, и мы обозначим ее как t ;
- получаемый размер файла равен 38 Мбайт = 38×2^{23} бит.

Главное — ничего не забыть; обязательно преобразовать все величины к одним и тем же размерностям и правильно выполнить вычисления.

$$4 \cdot 32\,000 \cdot 16 \cdot t = 38 \cdot 2^{23},$$

откуда $t = 155,648$. Это — в секундах. А у нас требуют указать длительность в минутах. Значит, полученное значение надо обязательно разделить на 60. А также (согласно условию задачи) — округлить полученное дробное значение **до ближайшего целого**:

$$t = 155,648 / 60 \approx 2,594 \approx 3 \text{ минуты.}$$

Ответ: 3.

8. Parole, parole, parole²...

Сколько “слов” длины 7 символов, начинающихся с английской буквы, можно составить из букв S, И, R, П, Q? Каждая буква может входить в “слово” несколько раз, а сами получаемые “слова” не обязательно должны быть осмысленными.

Решение

Для начала определим, сколько “слов” можно составить из указанных букв **без учета** той самой первой буквы, на которую накладывается особое условие (“только английская”), — т.е. количество “слов” длиной 6 символов.

Сделать это легче всего, если сопоставить каждой букве “свою” цифру, например: S = 0, И = 1, R = 2, П = 3, Q = 4 (порядок цифр в сущности не важен). Тогда нетрудно понять, что мы получили аналогию с числами в соответствующей системе счисления: “сколько различных 6-разрядных чисел можно получить в пятеричной системе счисления”. А это уже легко: количество таких чисел равно n^m , где n — основание системы счисления, а m — количество разрядов. В нашем случае, очевидно, получается $5^6 = 15\,625$ чисел (“слов”).

А теперь вернемся к первой букве, которая должна быть только английской. Английских букв у нас три. И для **каждой** из них возможно 15 625 слов. Значит, общее количество слов будет равно $3 \cdot 15\,625 = 46\,875$.

Ответ: 46 875.

Примечание. А что будет, если в условии будет сказано, что две первые буквы должны быть только английскими?

Тогда решение будет таким:

- кроме этих букв, остаются пять букв любых, тогда таких пятисимвольных “слов” будет $5^5 = 3125$;
- две первые буквы — только английские, а таких букв у нас три; значит, речь идет о количестве двузначных чисел в троичной системе: $3^2 = 9$;
- для каждого из этих девяти вариантов начала “слова” может быть 3125 вариантов продолжения, — значит, всего таких слов будет $9 \cdot 3125 = 28\,125$.

² “Слова, слова, слова...” (из популярной французской песни).

9. “Эх, раз, еще раз!” ...

Имеется рекурсивный алгоритм F :

```
procedure F(n: integer): integer;
begin
  if n > 3 then
    F := F(n - 1) + F(n - 2) + F(n - 3)
  else
    F := 2;
  end;
```

Чему равно значение, вычисленное при вызове этой процедуры в виде $F(6)$?

Решение

Вспомним: рекурсия — это вызов функции или процедуры “самой из себя”. При этом в каждом новом вызове вычисления внутри функции / процедуры производятся уже с соответствующим значением “внутренней” (формальной) переменной. Кроме того, в рекурсивной функции / процедуре обязательно должны быть и “конечные” условия, когда значение переменной определяется однозначно, без новых рекурсивных вызовов. В нашем случае это условие:

```
if n > 3 then
  ...
else
  F := 2;
```

Чтобы не запутаться и не забыть ни один рекурсивный вызов, удобнее всего вычертить схему этих вызовов в виде дерева.

1) Записываем самый первый вызов, для которого $n = 6$:

$$F(6) = F(6 - 1) + F(6 - 2) + F(6 - 3) = \\ = F(5) + F(4) + F(3).$$

2) Рекурсивный вызов $F(5)$ расписываем аналогично:

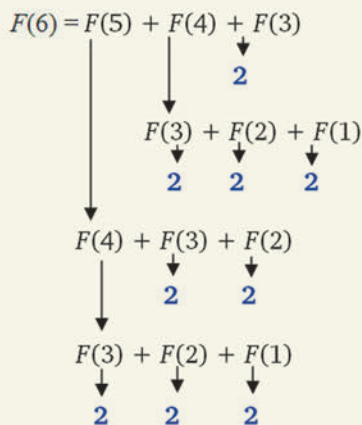
$$F(5) = F(5 - 1) + F(5 - 2) + F(5 - 3) = \\ = F(4) + F(3) + F(2).$$

И точно так же расписываем вызов $F(4)$:

$$F(4) = F(4 - 1) + F(4 - 2) + F(4 - 3) = \\ = F(3) + F(2) + F(1).$$

А вот значения $F(3)$, $F(2)$ и $F(1)$ у нас — “конечные”: согласно “конечному” условию рекурсии, все они равны 2.

3) Тогда дерево вызовов будет следующим:



4) Теперь можно расписать рекурсивные вычисления в виде простой суммы, постепенно “расшифровывая” каждое значение:

$$F(6) = \boxed{F(3) + F(2) + F(1)} + F(3) + F(2) + \boxed{F(3) + F(2) + F(1)} + F(3)$$

Теперь уже можно заменить вызовы $F(1)$, $F(2)$, $F(3)$ известными нам числовыми значениями:

$$F(6) = 2 + 2 + 2 + 2 + 2 + 2 + \\ + 2 + 2 + 2 = 9 \cdot 2 = 18.$$

Ответ: 18.

10. “Призрак Оперы” ... а также Firefox, Chrome и прочих

В сетях TCP/IP *маска сети* — это двоичное число, меньшее 2^{32} ; в маске сначала (в старших разрядах) записаны единицы, а затем с некоторого бита — нули. Маска определяет, какая часть IP-адреса относится к адресу подсети, а какая — к адресу конкретного компьютера (узла) в этой сети. Маска записывается по тем же правилам, что и IP-адрес, — в виде четырех десятичных чисел, каждое из которых соответствует одному байту и отделяется от других точкой. Адрес сети получается в результате применения поразрядной конъюнкции к заданному IP-адресу узла и маске.

Для узла с IP-адресом 167.57.252.220 адрес сети равен 167.48.0.0. Чему равен второй по счету слева байт маски? Ответ нужно записать в виде десятичного числа.

Решение

Обычно требовалось найти адрес сети по известному IP-адресу и маске. Здесь же нам, наоборот, требуется подобрать маску для известных IP-адреса и адреса сети.

Кроме того, заметим: первый байт адреса сети совпадает с первым байтом IP-адреса, — значит, первый байт маски равен 11111111_2 ; третий и четвертый же байты адреса сети нулевые, значит, им соответствуют и нулевые байты маски. Второй же по счету байт маски — самый “интересный”, он должен содержать как единицы, так и нули. Именно поэтому в задаче требуется определить не всю маску, а только этот “ключевой” второй байт.

1) Переводим оба “ключевых” числа в двоичную систему счисления:

- $57_{10} = 111001_2$;
- $48_{10} = 110000_2$.

2) Записываем поразрядную конъюнкцию, в которой второй операнд неизвестен:

$$\begin{array}{r}
 111001 \\
 \& \quad ???? \\
 \hline
 110000
 \end{array}$$

3) Сопоставляем первый операнд и результат:

• первый бит в обоих случаях равен 1, значит, соответствующий бит маски тоже равен 1;

• со вторым битом ситуация та же, значит, второй бит маски тоже равен 1;

• а вот третий бит в IP-адресе равен 1, а в адресе сети уже равен нулю, следовательно, соответствующий третий бит маски должен быть нулевым.

А вот теперь — самое интересное! Четвертый бит IP-адреса, как и четвертый бит адреса сети, равен нулю. Чему равен тогда бит в маске? Ведь он может быть равен как 1, так и 0, — в обоих случаях конъюнкция с нулем дает ноль.

И вот тут-то надо вспомнить, что в маске сначала до какого-то разряда идут только единицы, а начиная с этого разряда — только нули! Поэтому если третий бит маски (как мы однозначно определили) равен нулю, то *все последующие* биты правее этого нуля тоже должны быть только нулями! Поэтому маска получится такой: 110000_2 , или 48_{10} .

Вроде бы все правильно. Но ответ в этой задаче совсем другой! В чем же мы (намеренно), а также и почти все учащиеся, решавшие эту задачу, ошиблись? А вот в чем.

Мы “забыли”, что каждое из четырех чисел, записанных в маске через точку, должно соответствовать одному байту, т.е. восьми битам. А у нас при переводе десятичных чисел 57 и 48 получились 6-битовые двоичные числа. Следовательно, их оба нужно обязательно дополнить до 8-битовых двумя незначащими нулями слева!

Тогда и запись поразрядной конъюнкции должна иметь вид:

$$\begin{array}{r} 00111001 \\ \& \text{????????} \\ 00110000 \end{array}$$

Теперь рассуждения о значениях битов маски тоже нужно начать с точно определяемых значений 3-го и 4-го слева битов: раз соответствующие биты IP-адреса и адреса сети оба равны 1, то и соответствующие биты маски тоже равны 1.

Далее для 5-го слева бита значение бита маски тоже определяется однозначно: оно должно быть равно 0.

А теперь, вспомнив свойство маски, что в ней сначала до некоторого разряда записаны только единицы, а после этого разряда — только нули, определяем, что до найденных нами единичных битов тоже должны быть единицы, а после найденного нами нуля — нули. То есть маска будет такой: $11110000_2 = 240_{10}$. Это и есть правильный ответ.

Ответ: 240.

11. Четыре черненьких чумазеньких чертенка...

Исполнитель Чертежник перемещается по координатной плоскости. Основная команда Чертежника — **сместиться на (a, b)** , где a, b — целые числа. Она перемещает Чертежника из точки с координатами (x, y) в точку с координатами $(x + a, y + b)$. Например, из точки с координатами $(3, 5)$ команда

сместиться на $(-2, 1)$ переместит Чертежника в точку $(1, 6)$.

Цикл

ПОВТОРИ *число* РАЗ

последовательность команд

КОНЕЦ ПОВТОРИ

означает, что заданная последовательность команд будет выполнена указанное количество раз (значение должно быть натуральным).

Чертежник выполнял алгоритм, в котором количество повторений и оба смещения в первой из повторяемых команд неизвестны:

НАЧАЛО

сместиться на $(-2, 3)$

ПОВТОРИ ... РАЗ

сместиться на $(..., ...)$

сместиться на $(-2, -1)$

КОНЕЦ ПОВТОРИ

сместиться на $(-19, -18)$

КОНЕЦ

При этом, выполнив этот алгоритм, Чертежник вернулся в исходную точку. Какое наибольшее количество повторений могло быть указано в конструкции “ПОВТОРИ ... РАЗ”?

Решение

Помните, как на физике решали задачу о движении тела, брошенного под заданным углом к горизонту (“задача о пушечном выстреле”)? Тогда мы составляли два отдельных уравнения движения по координате x и координате y и рассматривали их в виде системы из двух уравнений. В нынешней задаче нужно сделать то же самое, обозначив неизвестные нам значения переменными: k — количество повторений, x и y — неизвестные смещения в первой повторяемой команде. При этом каждое уравнение приравнивается нулю, так как Чертежник после выполнения всех перемещений вернулся в исходную точку.

1) Перемещение Чертежника по координате x :

$$-2 + k \cdot (x - 2) - 19 = 0.$$

2) Перемещение Чертежника по координате y :

$$3 + k \cdot (y - 1) - 18 = 0.$$

3) Объединяем оба уравнения в систему

$$\begin{cases} -2 + k \cdot (x - 2) - 19 = 0, & \begin{cases} k \cdot (x - 2) = 21, \\ 3 + k \cdot (y - 1) - 18 = 0; \end{cases} \\ 3 + k \cdot (y - 1) - 18 = 0; & \begin{cases} k \cdot (y - 1) = 15. \end{cases} \end{cases}$$

4) На первый взгляд данная система нерешаема: два уравнения и три неизвестных. Но обратим внимание — в обоих случаях слева от знака равенства стоят произведения, один из сомножителей в которых один и тот же (k).

Разложим числа справа от знаков равенства на простые сомножители (благо это можно сделать одним-единственным способом): $21 = 3 \cdot 7$; $15 = 3 \cdot 5$.

Заметим: в обоих получившихся произведениях тоже повторяется один и тот же сомножитель — 3. Значит, это и есть k .

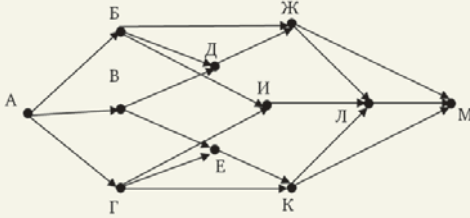
Определив, что $k = 3$, можно при желании (или при необходимости, если это потребуют в условии

задачи) вычислить и значения x и y . Нам же требуется только это значение k .

Ответ: 3.

12. “Эх, дороги” - 2

Дана схема дорог, связывающих города А, Б, В, Г, Д, Е, Ж, И, К, Л, М. По каждой дороге можно двигаться только в направлении, указанном стрелкой. Сколько возможно различных путей из города А в город Л?



Решение

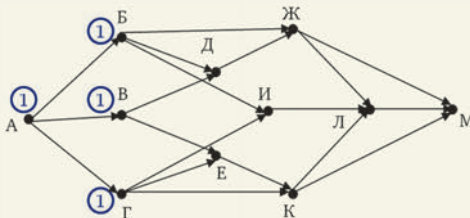
Раньше такие задачи мы решали, строя дерево путей. Но для такого сложного графа (большого количества городов и дорог) вычертить дерево путей очень сложно, оно будет слишком громоздким. Возможен ли другой способ?

Да, возможен. И именно его продемонстрировала в своих работах ученица 11-го “А” класса школы № 1360 Анастасия Ремизова: количество путей, ведущих в каждый город, можно вычислять напрямую.

1) Возле города А записываем единицу. Это — некое “стартовое” значение, поскольку уж один-то путь из А в М есть всегда.

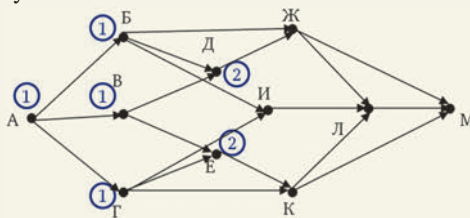
2) Смотрим город Б. В этот узел графа входит только одна стрелка, которая идет от узла А со значением 1. Поэтому возле узла Б тоже записываем единицу.

3) То же с городами (узлами) В и Г — в них по единственной входящей стрелке “переносится” все та же единица из узла А.



4) Смотрим город Д. В него входят две стрелки. Одна идет из узла Б и “несет с собой” оттуда единицу. Вторая же аналогичным способом переносит единицу по стрелке из узла В. Итого в узле Д в сумме получаем значение 2 (1+1).

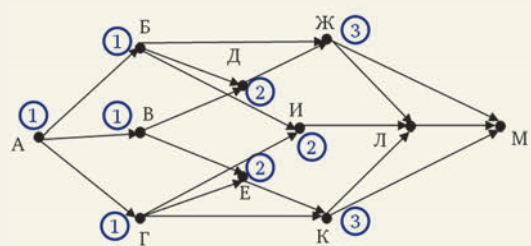
5) То же самое получаем и для узла Е, куда по соответствующим двум стрелкам “приходят” единицы из узлов В и Г.



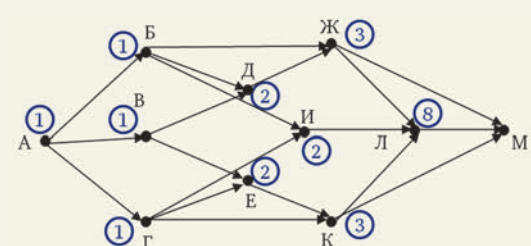
6) В узел Ж тоже входят две стрелки. Одна (из узла Б) “приносит” туда единицу. А вторая (из узла Д) “приносит” уже двойку. Итого в сумме получается 3 (1+2).

7) Для узла К история та же — рядом с ним тоже записываем 3 (1 по стрелке из узла Г плюс 2 по стрелке из узла Е).

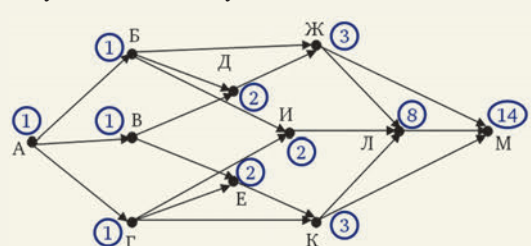
8) Для узла же И две стрелки “принесут” с собой из узлов Б и Г по единице каждая, итого в сумме получаем 2.



9) А вот теперь переходим к самому сложному узлу — Л. В него входят три стрелки. Первая, из узла Ж, “переносит” в Л тройку. Вторая, из узла К, “переносит” тоже тройку. И, наконец, третья, из узла И, “переносит” в Л двойку. В сумме же для Л получаем значение 8 (3+2+3).



10) Остается только узел М. В него приходят тоже три стрелки. Стрелка из узла Ж “приносит” в М тройку. Стрелка из узла К “приносит” в М тоже тройку. А стрелка из узла Л приносит в М восьмерку. В сумме для М получаем 14 (3+3+8).



Объяснение этого решения — гораздо более долгое, чем само решение. Расписать для каждого города соответствующие значения удастся буквально за полминуты. Что позволяет экономить немало (по сравнению с построением дерева) дефицитного времени для решения других задач.

Ответ: 14.

13. “Ищут пожарные, ищет милиция...”

В языке запросов поискового сервера для обозначения логической операции “ИЛИ” используется символ “|”, а для логической операции “И” — символ “&”.

В таблице приведены запросы и количество найденных по ним страниц некоторого сегмента сети Интернет.

Запрос	Кол-во найденных страниц, тыс.
Паскаль & (Си & Бейсик Кобол)	350
Паскаль & Кобол	187
Паскаль & Си & Бейсик & Кобол	48

Какое количество страниц (в тысячах) будет найдено по запросу

Паскаль & Си & Бейсик ?

Считаем, что все запросы выполнялись почти одновременно и набор страниц, содержащих искомые слова, за это время не изменялся.

Решение

“Традиционное” решение такой задачи — построение кругов Эйлера. Но здесь — целых четыре переменных, а числовых данных явно недостаточно, чтобы выполнить полное решение. Как же быть?

Обратим внимание на запрос *Паскаль & (Си & Бейсик | Кобол)* и раскроем скобки в этом логическом выражении: *Паскаль & Си & Бейсик | Паскаль & Кобол*.

А теперь посмотрим на запрос *Паскаль & Си & Бейсик & Кобол*. Правда, похоже? И там и там слева стоит “*Паскаль & Си & Бейсик*”. А если сравнить со вторым запросом — *Паскаль & Кобол*, то мы увидим: в первом запросе справа от “|” записано “*Паскаль & Кобол*” (как во втором запросе), а в третьем запросе правая часть — только “*Кобол*”. Но ведь существует “правило поглощения”: $A \& B \& A = A \& B$ (повторяющийся операнд A можно не повторять). А значит, верно и обратное: любой операнд можно повторить (записав через тот же знак операции) сколько угодно раз, не меняя значение логического выражения!

Воспользуемся этим и повторим в третьем запросе слово *Паскаль*: *Паскаль & Си & Бейсик & Паскаль & Кобол*. Значение выражения не изменилось, но зато его теперь можно представить в виде: $(\text{Паскаль} \& \text{Си} \& \text{Бейсик}) \& (\text{Паскаль} \& \text{Кобол})$.

Итак, четыре наших запроса (включая искомый) теперь выглядят так:

Паскаль & Си & Бейсик | Паскаль & Кобол

Паскаль & Кобол

$(\text{Паскаль} \& \text{Си} \& \text{Бейсик}) \& (\text{Паскаль} \& \text{Кобол})$

Паскаль & Си & Бейсик.

Совершенно очевидно, что можно выполнить “макроподстановку”:

$A = \text{Паскаль} \& \text{Си} \& \text{Бейсик}$,

$B = \text{Паскаль} \& \text{Кобол}$.

Тогда наши запросы примут вид:

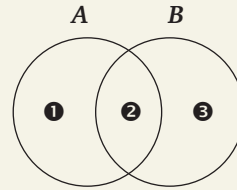
$A | B$ (350 тыс. стр.),

B (187 тыс. стр.),

$A \& B$ (48 тыс. стр.),

A — искомый.

Теперь мы получили задачу всего с двумя логическими переменными (и, соответственно, с двумя кругами Эйлера):



$A B$	$\textcircled{1} + \textcircled{2} + \textcircled{3}$	350 тыс. стр.
B	$\textcircled{2} + \textcircled{3}$	187 тыс. стр.
$A \& B$	$\textcircled{2}$	48 тыс. стр.
A	$\textcircled{1} + \textcircled{2}$	Требуется найти

Составляем уравнения:

$$\begin{cases} \textcircled{1} + \textcircled{2} + \textcircled{3} = 350; \\ \textcircled{2} + \textcircled{3} = 187; \\ \textcircled{2} = 48. \end{cases} \Rightarrow \begin{cases} \textcircled{1} + 48 + \textcircled{3} = 350; \\ 48 + \textcircled{3} = 187; \end{cases} \Rightarrow$$

$$\Rightarrow \begin{cases} \textcircled{1} + \textcircled{3} = 302; \\ \textcircled{3} = 139; \end{cases} \Rightarrow$$

Отсюда легко вычислить, что $\textcircled{1} = 163$, а $\textcircled{1} + \textcircled{2} = 163 + 48 = 211$.

Следовательно, по запросу *Паскаль & Си & Бейсик* будет найдено 211 тыс. стр.

Ответ: 211.

14. Вместо отрезков

Элементами множества X являются натуральные числа. Известно, что выражение

$$(n \in \{2, 3, 6, 15, 26, 30\}) \rightarrow$$

$$\rightarrow (((n \in \{1, 3, 6, 12, 15, 35\}) \wedge \neg(n \in X)) \rightarrow$$

$$\rightarrow \neg(n \in \{2, 6, 15, 26, 30\}))$$

истинно (принимает значение 1) при любом значении n . Требуется определить наименьшее возможное значение произведения элементов множества X .

Решение

Ранее в ЕГЭ были включены похожие задачи, в которых речь шла о принадлежности x некоторым отрезкам. Теперь же вместо отрезков — дискретные множества. Насколько решение такой “дискретной” задачи аналогично уже знакомому учащимся решению задач с отрезками? Посмотрим...

1) Для упрощения решения заменим предложенные множества чисел буквами:

$$\bullet \{2, 3, 6, 15, 26, 30\} = A;$$

$$\bullet \{1, 3, 6, 12, 15, 35\} = B.$$

Тогда исходное уравнение можно записать так:

$$(n \in A) \rightarrow (((n \in B) \wedge \neg(n \in X)) \rightarrow \neg(n \in A)) = 1.$$

2) Точно так же, как мы это делали в задачах с отрезками, заменим утверждения о принадлежности n какому-либо множеству логическими переменными:

$$\bullet (n \in A) = a;$$

$$\bullet (n \in B) = b;$$

$$\bullet (n \in X) = x.$$

Получим выражение в виде:

$$a \rightarrow ((b \wedge \neg x) \rightarrow \neg a) = 1.$$

3) Теперь попробуем упростить это выражение и заменить операцию следования на “ИЛИ” по правилу $A \rightarrow B = \neg A \vee B$:

$$a \rightarrow (\neg(b \wedge \neg x) \vee \neg a) = 1; \Rightarrow$$

$$\Rightarrow \neg a \vee (\neg(b \wedge \neg x) \vee \neg a) = 1.$$

Применим операцию отрицания к содержимому скобок $(b \wedge \neg x)$:

$$\neg a \vee (\neg b \vee x) \vee \neg a = 1.$$

Теперь все операции равноценны и скобки можно убрать:

$$\neg a \vee \neg b \vee x \vee \neg a = 1.$$

Повторяющийся аргумент $\neg a$ можно “поглотить”:

$$\neg a \vee \neg b \vee x = 1.$$

4) А теперь вернемся к исходной записи переменных — аргументов, учитывая, что операция “НЕ” в данном случае означает непринадлежность тому или иному множеству:

$$(n \notin A) \vee (n \notin B) \vee (n \in X) = 1$$

или

$$(n \notin \{2,3,6,15,26,30\}) \vee$$

$$\vee (n \in \{1,3,6,12,15,35\}) \vee (n \in X) = 1.$$

5) Для решения задачи (определения элементов множества X) “в лоб” нужно последовательно перебирать различные натуральные значения n . Впрочем, достаточно будет только начать это делать — искомую закономерность можно будет уловить уже на первых шагах.

- $n = 1$. Используется логическая операция ИЛИ, значит, для получения результата “ИСТИНА” достаточно истинности хотя бы одного из аргументов. Принадлежит ли число 1 первому множеству $(\{2,3,6,15,26,30\})$? Нет. Значит, условие $(n \notin \{2,3,6,15,26,30\})$ истинно. Этого достаточно для получения общего результата “ИСТИНА”, поэтому значения всех остальных аргументов нам безразличны.

- $n = 2$. Это число принадлежит первому множеству $\{2,3,6,15,26,30\}$, значит, соответствующее условие ложно. Тогда смотрим второй аргумент: $(n \in \{1,3,6,12,15,35\})$. Этому множеству число 2 не принадлежит, значит, данное условие истинно, и этого нам достаточно.

- $n = 3$. Замечаем, что это число принадлежит и первому множеству $(\{2,3,6,15,26,30\})$, и второму $(\{1,3,6,12,15,35\})$. Следовательно, ложны и первое, и второе условия. Что в этом случае может обеспечить истинность результата? Правильно: только истинность третьего условия! Оно записано в виде: $(n \in X)$. Значит, чтобы это условие было истинным, надо включить в состав множества X число 3.

В принципе, чтобы понять закономерность, этих трех примеров достаточно. Мы видим, что если какое-либо натуральное число n не входит хотя бы в одно из двух заданных нам множеств, то этого уже достаточно. И только если n имеется

в обоих заданных множествах, это число n требуется включить в искомое множество X : ведь нам требуется в итоге получить наименьшее возможное произведение его элементов, а значит, включать в множество X какие-то “ненужные” числа нам не следует.

6) Теперь легко, сравнивая между собой элементы первого и второго множеств, выписать все множество X :

$$\{2, 3, 6, 15, 26, 30\} \{1, 3, 6, 12, 15, 35\}$$

$$X = \{3, 6, 15\}$$

7) Вычисляем произведение элементов множества X :

$$3 \cdot 6 \cdot 15 = 270.$$

Ответ: 270.

15. Еще одна сумма элементов массива

В программе описан одномерный целочисленный массив. Ниже представлен фрагмент программы, обрабатывающей этот массив:

```
s := 0;
n := 8;
for i := 0 to n - 4 do begin
    s := s + A[i] - A[i + 3]
end;
```

До выполнения этого фрагмента в массиве находились четырехзначные нечетные натуральные числа. Какое наименьшее значение может иметь переменная s после выполнения данной программы?

Решение

Такая задача нам уже встречалась в предыдущем тренинге. Здесь условие лишь немного усложнено, но принцип решения — тот же.

1) Расписываем вычисление суммы в виде одной строки, помня, что значение i меняется в цикле от 0 до 6 (т.е. до $10 - 4$):

$$s = 0 + A[0] - A[3] + A[1] - A[4] + A[2] - A[5] + A[3] - A[6] + A[4] - A[7] + A[5] - A[8] + A[6] - A[9].$$

2) Теперь сокращаем одинаковые переменные (элементы массива), записанные со знаками “плюс” и “минус”:

$$s = A[0] - A[3] + A[1] - A[4] + A[2] - A[5] + A[3] - A[6] + A[4] - A[7] + A[5] - A[8] + A[6] - A[9].$$

Получаем:

$$s = A[0] + A[1] + A[2] - A[7] - A[8] - A[9].$$

3) Получаемая сумма (разность) должна, по условию, иметь наименьшее значение. А сами элементы массива — это четырехзначные нечетные натуральные числа, т.е. числа в диапазоне от 1001 до 9999. Следовательно, те элементы массива, которые записаны в выражении со знаком “плюс”,

должны быть наименьшими из возможных, а элементы массива со знаком “минус” должны быть наибольшими из возможных. Значит, нужно выбрать их следующими: $A[0] = A[1] = A[2] = 1001$, $A[7] = A[8] = A[9] = 9999$.

4) Вычисляем значение s при этих значениях элементов массива:

$$s = 1001 + 1001 + 1001 - 9999 - 9999 - 9999 = 3 \cdot (1001 - 9999) = -26\,994.$$

Ответ: $-26\,994$.

16. И снова числа

Получив на вход число x , программа печатает два числа — a и b . Определите наименьшее из чисел, при вводе которого алгоритм печатает сначала 2, а потом 357.

```
var x, a, b: integer;
begin
  readln(x);
  a := 0; b := 0;
  while x > 0 do
    begin
      a := a + 1;
      b := b + (x mod 1000);
      x := x div 1000;
    end;
  writeln(a); write(b);
end.
```

Решение

И такая задача тоже уже встречалась. Но в этом ее варианте есть небольшой нюанс, который существенно меняет решение.

1) Проанализировав алгоритм, видим, что переменная a — это счетчик проходов цикла. А в переменной b накапливается сумма... чего? Раньше в такой задаче имелись операторы

```
b := b + (x mod 10);
x := x div 10;
```

и это означало вычисление суммы цифр. Теперь же вместо константы 10 записана константа 1000, а значит, из числа выделяются сразу **тройки** цифр.

2) Поскольку в качестве значения переменной a выводится число 2, цикл выполнялся два раза. Следовательно, исходное число могло состоять из четырех, пяти или шести цифр, — только тогда мы получаем из него две триады цифр, одна из которых (левая), возможно, неполная. А поскольку нам требуется, чтобы число было наименьшим, левая “триада” должна состоять из одной цифры.

3) Обозначим неизвестные цифры исходного числа как a, b, c и d . Тогда первый проход цикла отделит цифры bcd , т.е. число $100 \cdot b + 10 \cdot c + d$. А второй проход цикла добавит к сумме однозначное число (цифру) a .

4) Какой может быть эта единственная цифра a , чтобы число было наименьшим? Очевидно, $a = 1$.

Тогда, зная, что итоговая сумма равна 357, сразу получаем, что $bcd = 356$. А все первоначальное число x тогда равно 1356.

Ответ: 1356.

17. Опять логика

Сколько существует различных наборов значений логических переменных $a_1, a_2, \dots, a_5, b_1, b_2, \dots, b_5, c_1, c_2, \dots, c_6$, которые удовлетворяют перечисленным уравнениям?

$$\begin{aligned} (a_1 \rightarrow a_2) \wedge (a_2 \rightarrow a_3) \wedge (a_3 \rightarrow a_4) \wedge (a_4 \rightarrow a_5) &= 1 \\ (b_1 \rightarrow b_2) \wedge (b_2 \rightarrow b_3) \wedge (b_3 \rightarrow b_4) \wedge (b_4 \rightarrow b_5) &= 1 \\ (c_1 \rightarrow c_2) \wedge (c_2 \rightarrow c_3) \wedge (c_3 \rightarrow c_4) \wedge (c_4 \rightarrow c_5) &= 1 \\ a_1 \wedge b_2 \wedge c_3 &= 0. \end{aligned}$$

В качестве ответа нужно указать количество таких наборов переменных.

Решение

1) Анализируем первое уравнение:

$$(a_1 \rightarrow a_2) \wedge (a_2 \rightarrow a_3) \wedge (a_3 \rightarrow a_4) \wedge (a_4 \rightarrow a_5) = 1.$$

Начинаем составлять таблицу возможных значений переменных a_1, a_2, \dots, a_5 . При этом учитываем, что при использовании логической операции “И” результат 1 обеспечивается только единичными значениями всех ее аргументов, а также помним, что операция следования истинна в трех случаях: $1 \rightarrow 1, 0 \rightarrow 1$ и $0 \rightarrow 0$. Поэтому если в каждом следовании первая переменная равна единице, то из нее может следовать только единица, а из нуля может следовать как единица, так и ноль. В результате получаем следующую таблицу значений переменных:

a_1	a_2	a_3	a_4	a_5
1	1	1	1	1
0	1	1	1	1
0	0	1	1	1
0	0	0	1	1
0	0	0	0	1
0	0	0	0	0

2) Второе уравнение полностью аналогично первому, только имена переменных — другие. Поэтому таблица значений переменных будет такая же. И то же самое можно сказать про третье уравнение. Получаем:

b_1	b_2	b_3	b_4	b_5
1	1	1	1	1
0	1	1	1	1
0	0	1	1	1
0	0	0	1	1
0	0	0	0	1
0	0	0	0	0

c_1	c_2	c_3	c_4	c_5
1	1	1	1	1
0	1	1	1	1
0	0	1	1	1
0	0	0	1	1
0	0	0	0	1
0	0	0	0	0

3) Анализируем четвертое уравнение. Оно является “ключевым”:

$$a_1 \wedge b_2 \wedge c_3 = 0.$$

Строим для него таблицу истинности:

a1	b2	c3	Результат
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Из нее нам требуются все строки, кроме последней.

4) Подсчитаем теперь количество значений переменных, соответствующие таблице истинности четвертого уравнения. Для этого в таблицах значений переменных первых трех уравнений выделяем (например, цветом фона) столбцы, соответствующие переменным $a1$, $b2$ и $c3$:

a1	a2	a3	a4	a5
1	1	1	1	1
0	1	1	1	1
0	0	1	1	1
0	0	0	1	1
0	0	0	0	1
0	0	0	0	0

b1	b2	b3	b4	b5
1	1	1	1	1
0	1	1	1	1
0	0	1	1	1
0	0	0	1	1
0	0	0	0	1
0	0	0	0	0

c1	c2	c3	c4	c5
1	1	1	1	1
0	1	1	1	1
0	0	1	1	1
0	0	0	1	1
0	0	0	0	1
0	0	0	0	0

И теперь для каждой строки таблицы значений четвертого уравнения перемножаем друг на друга количество нулей и единиц соответственно, содержащихся в выделенном столбце для каждой из переменных.

Например, в первой строке у нас записаны значения:

a1	b2	c3
0	0	0

Поэтому из трех предыдущих таблиц мы:

- из столбца для $a1$ получаем количество **нулей**, равное 5;
 - из столбца для $b2$ получаем количество **нулей**, равное 4;
 - из столбца для $c3$ получаем количество **нулей**, равное 3,
- и записываем в сводную таблицу произведение: $5 \cdot 4 \cdot 3 = 60$.

Для второй строки имеем значения:

a1	b2	c3
0	0	1

Поэтому из трех предыдущих таблиц:

- из столбца для $a1$ получаем количество **нулей**, равное 5;
- из столбца для $b2$ получаем количество **нулей**, равное 4;
- а вот из столбца для $c3$ надо определить уже количество не нулей, а **единиц**, которое, правда, тоже равно 3.

В результате в сводную таблицу записывается такое же произведение: $5 \cdot 4 \cdot 3 = 60$.

И так поступаем для каждой строки сводной таблицы: по заданным в ней значениям $a1$, $b2$ и $c3$ определяем, надо ли в соответствующих столбцах искать количества нулей или же единиц, определяем эти количества и перемножаем их. А потом суммируем все полученные в каждой строке сводной таблицы произведения. В результате получаем следующую таблицу:

a1	b2	c3	Кол-во наборов переменных
0	0	0	$5 \cdot 4 \cdot 3 = 60$
0	0	1	$5 \cdot 4 \cdot 3 = 60$
0	1	0	$5 \cdot 2 \cdot 3 = 30$
0	1	1	$5 \cdot 2 \cdot 3 = 30$
1	0	0	$1 \cdot 4 \cdot 3 = 12$
1	0	1	$1 \cdot 4 \cdot 3 = 12$
1	1	0	$1 \cdot 2 \cdot 3 = 6$
Итого:			$60 + 60 + 30 + 30 + 12 + 12 + 6 = 210$

Ответ: 210.

Что же касается заданий третьей группы (в предыдущих версиях ЕГЭ — “задачи группы С”), то первые две из них не вызвали у учащихся особых трудностей. Первая задача (на поиск ошибки в программе) и вторая (написание программы) вполне решаемы при условии достаточной подготовки школьников по программированию. Третью задачу (на теорию игр “в камешки”) наши ученики тоже решили практически все. А вот четвертая задача на разработку программы для обработки наборов числовых данных, оптимальной по времени выполнения и затратам памяти, оказалась, конечно, по силам только двум наиболее сильным ученикам. Но это и понятно, задачи “С4” — это настоящий “высший пилотаж” программирования. А потому они заслуживают отдельного разговора, к которому авторы предполагают вернуться в одном из последующих номеров журнала.

1 апреля открывается прием заявок на 2015/16 учебный год

Фестиваль педагогических идей «Открытый урок» festival.1september.ru

Свидетельство о регистрации СМИ Эл. № ФС77-53231

В течение 12 лет – самый массовый, представительный и посещаемый педагогический форум Рунета. Самая большая коллекция авторских разработок учителей.

Разместить публикацию может каждый педагог. Всем авторам предоставляются документы о публикации. По итогам каждого учебного года выпускаются электронные и бумажные сборники.

В рамках фестиваля для желающих проводится конкурс презентаций. Всем участникам конкурса высылаются специальные дипломы.

Удобный Личный кабинет участника фестиваля, возможность автоматического создания личного профессионального портфолио. В помощь участникам – квалифицированные сотрудники оргкомитета. Единственный в России образовательный сайт, имеющий службу поддержки в режиме on-line 7 дней в неделю.



Участвуйте в фестивале, размещайте свои работы, получайте документы о публикации!

Фестиваль творческих и исследовательских работ учащихся «Портфолио ученика» project.1september.ru

Свидетельство о регистрации СМИ Эл. № ФС77-53211

Площадка для публикации работ учащихся, выполненных под руководством педагогов.

Всем ученикам и педагогам предоставляются документы о публикации. По итогам каждого учебного года выпускаются электронные и бумажные сборники.

В рамках фестиваля для желающих проводится конкурс проектных работ.

Все участники конкурса награждаются специальными дипломами.



Участвуйте вместе с учениками!



ДИСТАНЦИОННЫЕ КУРСЫ ПОВЫШЕНИЯ КВАЛИФИКАЦИИ

(с учетом требований ФГОС)

С 1 апреля начинается прием заявок на первый поток 2015/16 учебного года

образовательные программы:

- НОРМАТИВНЫЙ СРОК ОСВОЕНИЯ – **108** УЧЕБНЫХ ЧАСОВ

Стоимость – 4990 руб.

- НОРМАТИВНЫЙ СРОК ОСВОЕНИЯ – **72** УЧЕБНЫХ ЧАСА

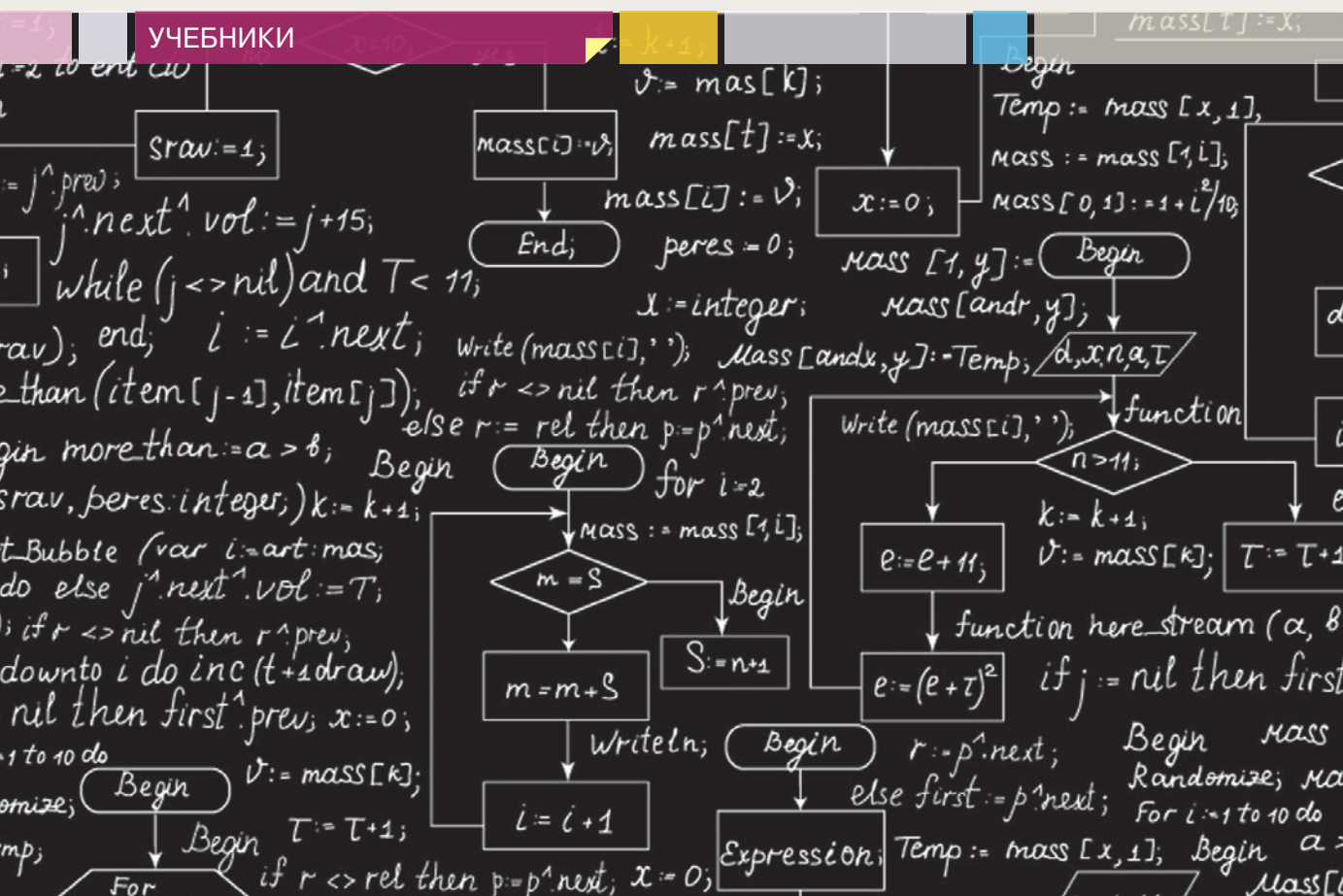
Стоимость – от 3990 руб.

По окончании выдается удостоверение о повышении квалификации установленного образца

Перечень курсов и подробности – на сайте edu.1september.ru

Пожалуйста, обратите внимание:

заявки на обучение подаются только из Личного кабинета, который можно открыть на любом сайте портала www.1september.ru



Алгоритмизация и программирование

Графические примитивы

К.Ю. Поляков,
д. т. н., Санкт-Петербург,
<http://kpolyakov.spb.ru>,

Е.А. Еремин,
к. ф.-м. н., г. Пермь

Ключевые слова:

- графический примитив
- линия
- прямоугольник
- окружность
- ломаная
- контур
- перо
- заливка
- кисть

Что такое графические примитивы?

Программа строит изображение на экране с помощью графических примитивов — элементарных фигур. Каждая из таких фигур рисуется на экране с помощью одной команды.

Графический примитив — это геометрическая фигура, которая добавляется на рисунок с помощью одной команды.

Основные примитивы исполнителя Рисователь:

- пиксель
- линия
- прямоугольник
- окружность

Одну команду — **пиксель** — мы уже знаем, теперь изучим остальные.

Линия рисуется пером. “Перо” — это набор свойств линии: толщина и цвет. Для того чтобы нарисовать линию, нужно сначала выбрать ее характеристики с помощью команды **перо**:

перо (1, синий)

Первый аргумент в скобках — это толщина линии, второй — ее цвет.

Обратите внимание, что эта команда сама по себе ничего не рисует, но устанавливает режим рисования для следующих команд.

Теперь применим команду **линия**:

```
линия ( 10 , 20 , 15 , 20 )
```

Первые два аргумента — это координаты одного конца линии (точнее, отрезка), следующая пара — координаты второго конца. Здесь мы рисуем отрезок из точки (10,20) в точку (15,20).

Вспомните, что ранее мы решили эту задачу с помощью команды **пиксель** и цикла. Решение с помощью команды **линия** значительно проще. Кроме того, она позволяет так же легко рисовать любые отрезки на холсте, в том числе и наклонные. Конечно, эти отрезки также состоят из пикселей, но расчет координат очередной точки берет на себя команда **линия**.

Теперь научимся рисовать прямоугольники. Рисователь умеет строить только прямоугольники, у которых стороны строго вертикальны и строго горизонтальны. Для того чтобы нарисовать прямоугольник, нужно знать координаты двух его противоположных углов — верхнего левого и нижнего правого. Так как две стороны вертикальны, а две горизонтальны, по этим данным можно определить координаты двух оставшихся углов.

Эта программа рисует прямоугольник, противоположные углы которого находятся в точках (20,10) и (40,30):

```
перо ( 1 , синий )  
кисть ( красный )  
прямоугольник ( 20 , 10 , 40 , 30 )
```

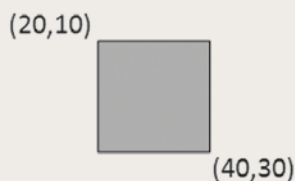


Рис. 64

В первой строчке мы определяем свойства контура, как для линий. Прямоугольник — это замкнутая фигура, поэтому для него можно задать еще и цвет заливки. Это делается с помощью команды **кисть** во второй строке. Если нужно нарисовать только рамку (не заливая внутреннюю часть фигур), устанавливается прозрачный цвет кисти:

```
кисть ( прозрачный )
```

Команды **перо** и **кисть** ничего не рисуют, а только устанавливают режимы рисования для команды **прямоугольник** в последней строке. Этой команде передаются четыре аргумента, они разделены запятыми. Первая пара чисел — это координаты левого верхнего угла, вторая — координаты правого нижнего.

Команда **окружность** предназначена для рисования окружностей и кругов. Ей нужно передать три аргумента: координаты центра и радиус. Окружность — замкнутая фигура, поэтому для нее можно

задавать свойства пера и цвет заливки внутренней части (круга).

Эта программа рисует круг с центром в точке (50, 30) радиуса 20 пикселей:

```
перо ( 1 , синий )  
кисть ( красный )  
окружность ( 50 , 30 , 20 )
```



Рис. 65

Сама окружность будет синего цвета, а внутренняя часть круга заливается красным цветом.

Ломаные

Как мы увидели, примитивов в СКИ Рисователя не так много. Например, нет специальной команды для рисования треугольника (как вы думаете, почему?). Но треугольник можно построить из отрезков, вызвав несколько раз команду **линия**:

```
линия ( 20 , 30 , 30 , 10 )  
линия ( 30 , 10 , 40 , 30 )  
линия ( 40 , 30 , 20 , 30 )
```

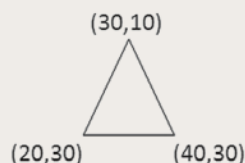


Рис. 66

Заметим, что данные в этих строчках повторяются — следующий отрезок начинается в той же точке, где заканчивается предыдущий. Это означает, что мы рисуем ломаную линию, “не отрывая пера от холста”. Чтобы не повторяться, в таких случаях удобно использовать другие команды, которые приведут к точно такому же результату:

```
в точку ( 20 , 30 )  
линия в точку ( 30 , 10 )  
линия в точку ( 40 , 30 )  
линия в точку ( 20 , 30 )
```

Команда **в точку** переводит исполнителя в заданную точку холста, не оставляя следа, а команда **линия в точку** рисует линию из того места, где стоял исполнитель, в точку с новыми координатами. В первой строке программы мы выводим исполнителя в вершину треугольника (20,30), а затем тремя отрезками рисуем три стороны треугольника, замыкая контур.

С помощью этих команд можно построить и незамкнутую ломаную линию. Для этого не нужно строить последний отрезок, соединяющий конец ломаной с ее началом.

Заливка

Как же залить треугольник? Вспомните, что заливка прямоугольников и кругов происходила при

вызове команд рисования, одновременно с рисованием контура. Здесь контур уже готов, и нужно залить его внутреннюю часть. Для этого в СКИ Рисователя есть команда **залить**, которая заливает одноцветную область, начиная с заданной точки:

кисть (красный)

залить (30, 20)

В этом примере заливка начинается в точке (30,20), область заполняется красным цветом. Где же остановится заливка? Это зависит от цвета пикселя (30,20). Если, допустим, этот пиксель был белым до начала заливки, то заливка остановится на границе белой области.

Пример

Используя только что изученные команды, нарисуем беседку:

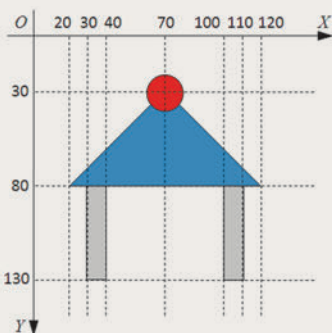


Рис. 67

Два столбика покрашены в серый цвет, крыша — в синий, а шарик на крыше — в красный.

Рисунок состоит из двух прямоугольников, треугольника и круга. Чтобы нам легче было составить программу, мы нанесли на рисунок линии сетки, которые показывают координаты всех угловых точек графических примитивов: прямоугольников, круга и отрезков, из которых строится треугольник крыша.

Поскольку красный круг “накрывает” крышу, его нужно рисовать после крыши. Прямоугольники и треугольник можно рисовать в любом порядке. Вот один из вариантов программы (для удобства объяснения строки пронумерованы):

алг Беседка

нач

новый лист (200, 200, белый) | 1

перо (1, черный) | 2

кисть (серый) | 3

прямоугольник (30, 80, 40, 130) | 4

прямоугольник (100, 80, 110, 130) | 5

в точку (20, 80) | 6

линия в точку (70, 30) | 7

линия в точку (120, 80) | 8

линия в точку (20, 80) | 9

кисть (синий) | 10

залить (70, 40) | 11

кисть (красный) | 12

окружность (70, 30, 10) | 13

кон

Сначала создаем холст размером 200×200 пикселей (строка 1). Затем устанавливаем черный цвет

контура (строка 2) и серый цвет заливки (строка 3). В строках 4 и 5 рисуем столбики-прямоугольники с этими свойствами.

Дальше рисуем крышу из трех отрезков (строки 7–9). После того как контур готов, устанавливаем синий цвет кисти (строка 10) и закрашиваем треугольник (строка 11).

Круг с красной заливкой рисуем в последнюю очередь (строки 12–13).

Контрольные вопросы

1. Зачем в СКИ графических исполнителей добавляют команды для рисования примитивов?
2. Как определить размеры прямоугольника по координатам углов?
3. Алгоритмы какого типа мы использовали в этом параграфе?
4. Как нарисовать замкнутую ломаную линию и залить ее внутреннюю область? С какими проблемами вы можете столкнуться?
5. Как вы думаете, почему обычно в СКИ исполнителей нет команды **треугольник**?
6. Что произойдет, если исполнитель будет рисовать за пределами холста?

Задачи

1. Определите размеры прямоугольников, которые нарисованы с помощью команд
 - а) **прямоугольник (10, 10, 50, 20)**
 - б) **прямоугольник (110, 120, 50, 20)**
 - в) **прямоугольник (110, 20, 50, 120)**
 - г) **прямоугольник (10, 130, 150, 20)**
2. Посмотрите, как нарисованы кнопки в окне какой-нибудь программы. Нарисуйте с помощью Рисователя изображения кнопки в нажатом и отпущенном положениях.
3. Нарисуйте с помощью Рисователя стандартное окно программы в вашей операционной системе.
4. Придумайте и нарисуйте с помощью Рисователя свой рисунок, в котором есть прямоугольники, окружности и ломаные.
5. Напишите программы для Рисователя, которые строят такие рисунки:

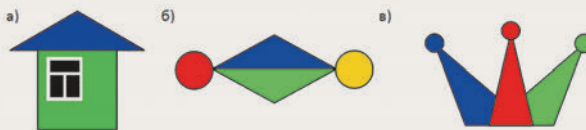


Рис. 68

6. Напишите программы для Рисователя, которые строят изображения трехмерных объектов:

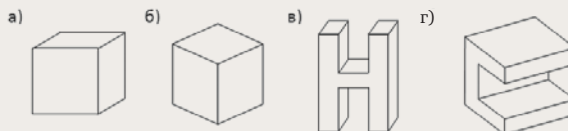


Рис. 69

Темы для сообщений:

- а) “Алгоритмы выполнения заливки”
- б) “Команда **эллипс**”
- в) “Цвет в модели RGB”

Применение процедур

Ключевые слова:

- процедура
- параметры

Когда помогут процедуры?

Как вы знаете, процедуры (вспомогательные алгоритмы) служат для того, чтобы не писать несколько раз одинаковые серии команд. Например, нам нужно нарисовать три прямоугольных треугольника одинаковых размеров:

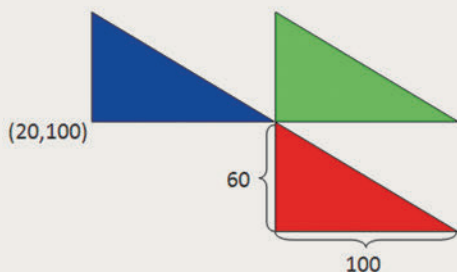


Рис. 70

Хотя треугольники и похожи, они не совсем одинаковы: у них разные

- цвета заливки;
- расположение (координаты углов на холсте).

Эти данные нужно как-то передать в процедуру из вызывающего алгоритма. Таким образом, нужна процедура с параметрами.

Сразу понятно, что один из параметров — это цвет заливки. Подумаем, какие данные нужны для того, чтобы определить место каждого из треугольников на холсте. Конечно, можно задать координаты всех трех вершин, это шесть чисел (три пары координат). При этом мы сможем рисовать любые (!) треугольники. Но в нашей задаче ситуация проще: размеры сторон известны, все треугольники прямоугольные, и самое важное — стороны прямого угла параллельны сторонам холста. Поэтому координаты любого угла позволяют вычислить координаты всех остальных углов и построить треугольник.

Строим процедуру

Предположим, что мы знаем координаты прямого угла. Как найти координаты остальных углов? Построим один треугольник:

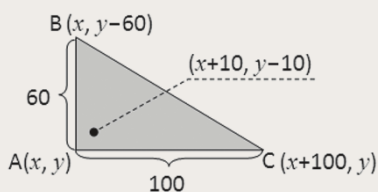


Рис. 71

Пусть (x, y) — координаты прямого угла А. Так как отрезок АВ идет строго вертикально, угол В расположен на таком же расстоянии от левой границы, что и угол А, поэтому их x -координаты совпадают. В то же время y -координата угла В будет меньше y -координаты А на 60 (на высоту треугольника). Поэтому угол В имеет координаты $(x, y - 60)$.

Так как отрезок АС идет строго горизонтально, угол С расположен на таком же расстоянии от верхней границы, что и угол А, поэтому их y -координаты совпадают. В то же время x -координата угла С будет больше x -координаты А на 100 (на длину основания треугольника). Поэтому угол С имеет координаты $(x + 100, y)$.

Для того чтобы залить треугольник, нам понадобится любая точка внутри него, например, точка $(x + 10, y - 10)$.

Теперь можно написать текст процедуры:

```
алг треугольник ( цел x, y, цвет ц )
нач
  в точку ( x, y )
  линия в точку ( x, y - 60 )
  линия в точку ( x + 100, y )
  линия в точку ( x, y )
  кисть ( ц )
  залить ( x + 10, y - 10 )
```

кон

Процедура принимает три параметра:

- два целых значения x и y — координаты прямого угла;
- цвет заливки, обозначенный именем $ц$; эта величина относится к особому типу **цвет**.

В теле процедуры мы рисуем замкнутую ломаную линию, используя рассчитанные координаты углов треугольника, и затем заливаем его тем цветом, который будет передан в процедуру вызывающей программой. Обратите внимание, что все координаты зависят от значений x и y , то есть процедура позволяет нам рисовать треугольники в любом месте холста.

Используем процедуру

Теперь составим основную программу. Она будет содержать три вызова процедуры “треугольник” с разными значениями параметров. Нам нужно знать координаты прямых углов всех треугольников. Для первого (синего) треугольника эти координаты нам известны — $(20, 100)$. Для зеленого треугольника x -координата будет на 100 больше, то есть его прямой угол имеет координаты $(120, 100)$. Красный треугольник находится ниже зеленого на 60 пикселей, поэтому координаты его прямого угла — $(120, 160)$. Теперь можно записать основную программу:

```
использовать Рисователь
алг Треугольники
нач
  треугольник ( 20, 100, синий )
  треугольник ( 120, 100, зеленый )
  треугольник ( 120, 160, красный )
кон
```

Не забудьте, что после этой программы нужно поместить текст процедуры, иначе Рисователь не сможет выполнить неизвестную команду “треугольник”.

При первом вызове процедура выполняется для значений $x = 20$, $y = 100$ и $ц$ = синий. Поэтому фактически будут выполнены команды

```
в точку ( 20 , 100 )
линия в точку ( 20 , 40 )
линия в точку ( 120 , 100 )
линия в точку ( 20 , 100 )
кисть ( синий )
залить ( 30 , 90 )
```

После этого процедура вызывается еще два раза, с другими значениями параметров. Поэтому треугольники будут нарисованы в других местах холста.

Контрольные вопросы

1. Почему процедуры для рисования фигур на холсте, как правило, имеют параметры?
2. Как определить, какие данные включать в список параметров процедуры?
3. Можно ли было включить в параметры процедуры **треугольник** длины сторон треугольника? Какие достоинства и недостатки имеет это решение?
4. Процедура **треугольник** записана ниже основной программы, но треугольники не появляются на холсте. В чем может быть причина? Как вы будете искать ошибку?

Задачи

1. Перепишите процедуру **треугольник**, взяв за точку отсчета правый угол основания треугольника (считайте, что он имеет координаты (x, y) , которые передаются как параметры). Измените основную программу так, чтобы получился точно такой же рисунок, как и раньше.
2. Напишите программы для Рисователя, которые строят такие рисунки с помощью одной процедуры:

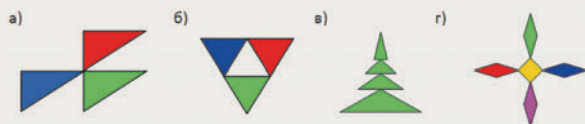


Рис. 72

3. Напишите программы для Рисователя, которые строят изображения трехмерных объектов с помощью одной процедуры:

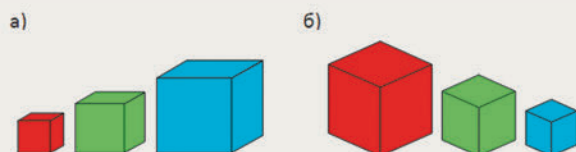


Рис. 73

Применение циклов

Ключевые слова:

- цикл
- узор
- процедура
- свойства по умолчанию

Часто в рисунках встречаются одинаковые элементы, которые удобно рисовать с помощью циклов. В этом параграфе вы узнаете, как грамотно составлять такие циклы.

Узоры

Узор — это рисунок, основанный на повторении одинаковых элементов. Эти элементы могут быть различной сложности, начиная с примитивов — кругов, прямоугольников и др.

Начнем с простой задачи: построим ряд из пяти кругов радиуса 5 пикселей:

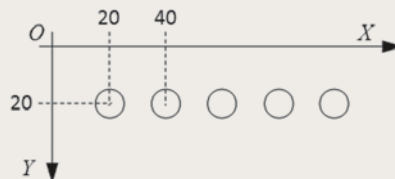


Рис. 74

Конечно, можно рисовать их отдельно, используя пять вызовов команды **окружность**:

```
окружность ( 20 , 20 , 5 )
окружность ( 40 , 20 , 5 )
окружность ( 60 , 20 , 5 )
окружность ( 80 , 20 , 5 )
окружность ( 100 , 20 , 5 )
```

Можно заметить, что в этих вызовах изменяется только x -координата центра. Ее можно сделать переменной и назвать x . Эта переменная будет меняться от 20 до 100 с шагом 20, поэтому можно использовать такой цикл с переменной:

```
цел x
нц для x от 20 до 100 шаг 20
    окружность ( x , 20 , 5 )
кц
```

Здесь в заголовке цикла появилась новая часть — **шаг 20**. Это значит, что изменение переменной происходит с шагом 20: 20, 40, 60, ... По умолчанию (то есть если мы не указали иначе) шаг равен единице.

Теперь построим три одинаковых ряда кругов, расположенных на расстоянии 20 пикселей по высоте друг от друга:

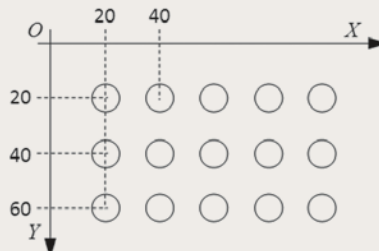


Рис. 75

Один ряд отличается от другого только у-координатой, поэтому можно оформить вспомогательный алгоритм (процедуру):

```

алг Ряд ( цел у )
нач
  цел х
  нц для х от 20 до 100 шаг 20
    окружность ( х , у , 5 )
  кц
кон

```

а затем в основной программе вызвать ее три раза

```

алг Узор
нач
  Ряд ( 20 )
  Ряд ( 40 )
  Ряд ( 60 )
кон

```

Снова замечаем, что у-координата, которая передается процедуре Ряд, изменяется с постоянным шагом 20, поэтому можно обозначить ее как переменную у и использовать цикл:

```

цел у
нц для у от 20 до 60 шаг 20
  Ряд ( у )
кц

```

Вызов процедуры можно заменить на вложенный цикл, в котором изменяется переменная х:

```

цел х , у
нц для у от 20 до 60 шаг 20
  нц для х от 20 до 100 шаг 20
    окружность ( х , у , 5 )
  кц
кц

```

Использование процедур

Вместо команды **окружность** можно вызывать и свою процедуру. Например, построим на экране такой узор из ромбов:

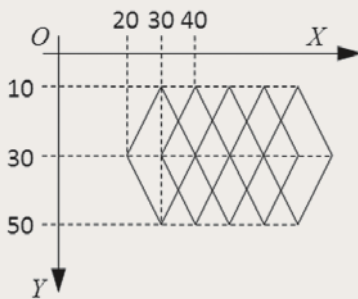


Рис. 76

Чтобы составить процедуру, нарисуем один ромб, определим по рисунку его размеры и обозначим через (x, y) координаты одного из углов. Это позволит рассчитать координаты всех остальных углов так же, как мы делали ранее.

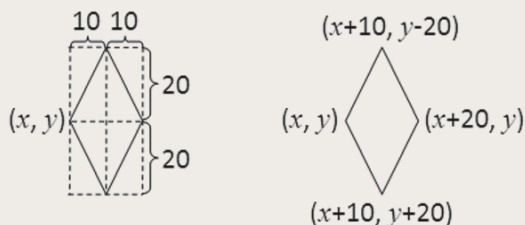


Рис. 77

Составляем процедуру, которая рисует контур ромба:

```

алг Ромб ( цел х , у )
нач
  в точку ( х , у )
  линия в точку ( х + 10 , у - 20 )
  линия в точку ( х + 20 , у )
  линия в точку ( х + 10 , у + 20 )
  линия в точку ( х , у )
кон

```

У процедуры два параметра — координаты левого угла ромба. По схеме рис. 76 находим, что для первого ромба это координаты $(20, 30)$, для второго — $(30, 30)$, для третьего — $(40, 30)$ и т.д.:

```

Ромб ( 20 , 30 )
Ромб ( 30 , 30 )
Ромб ( 40 , 30 )
Ромб ( 50 , 30 )
Ромб ( 60 , 30 )

```

Видим, что х-координата увеличивается с шагом 10, а у-координата не изменяется. Поэтому можно использовать цикл:

```

цел х
нц для х от 20 до 60 шаг 10
  Ромб ( х , 30 )
кц

```

Штриховка

Во многих задачах компьютерной графики нужно заштриховать какие-то области. Например, штриховкой обозначается сечение на чертежах и болота на картах местности.

Штриховка состоит из параллельных линий, которые удобно рисовать в цикле. Выполним вертикальную штриховку прямоугольника:

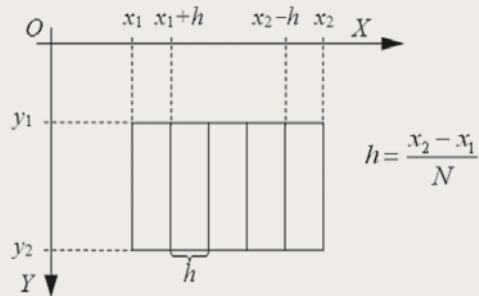


Рис. 78

Решим задачу в общем виде, для любого прямоугольника. Будем считать, что его верхний левый угол находится в точке с координатами (x_1, y_1) , а правый нижний — в точке с координатами (x_2, y_2) . Сначала нарисуем контур прямоугольника:

```

цел х1 = 100 , х2 = 300
цел у1 = 100 , у2 = 200
прямоугольник ( х1 , у1 , х2 , у2 )

```

Заметьте, что в первых двух строчках мы не только объявляем переменные **х1**, **х2**, **у1** и **у2**, но и присваиваем им начальные значения. Конечно, вы можете выбрать и другие числа.

В результате штриховки нужно разделить прямоугольник на N полос, так что шаг штриховки (расстояние между соседними линиями) можно вычислить по формуле

$$h = \frac{x_2 - x_1}{N}$$

В нашей программе мы будем использовать только целые значения шага (в пикселях), поэтому величина h вычисляется с помощью деления нацело, которое в алгоритмическом языке записывается как вспомогательный алгоритм с именем **div**:

```
цел h
h := div( x2 - x1, N )
```

Например, если $x_1 = 100$, $x_2 = 200$ и $N = 5$, мы получим $h = 20$.

Как видно из рис. 78, первая линия штриховки идет из точки $(x_1 + h, y_1)$ в точку $(x_1 + h, y_2)$:

```
линия( x1 + h, y1, x1 + h, y2 )
```

У второй линии x -координата обоих концов отрезка увеличивается на h , у третьей — еще на h и т.д. Для обоих концов последней линии x -координата равна $x_2 - h$, т.к. $(x_2 - x_1)$ в нашем примере делится на N . Можно обозначить эту изменяющуюся величину как переменную x и записать такой цикл штриховки:

```
цел x
нц для x от x1 + h до x2 - h шаг h
линия( x, y1, x, y2 )
```

```
кц
```

Приведем полную программу, которая строит прямоугольник и выполняет штриховку:

```
использовать Рисователь
алг Штриховка
нач
цел N = 5
цел x1 = 100, x2 = 300
цел y1 = 100, y2 = 200
цел h, x
прямоугольник( x1, y1, x2, y2 )
h := div( x2 - x1, N )
нц для x от x1 + h до x2 - h шаг h
линия( x, y1, x, y2 )
кц
кон
```

Контрольные вопросы

1. Подумайте, что нужно изменить в первой программе, чтобы круги выстроились по диагонали:



- Нужен ли для этого вложенный цикл?
2. Сравните два варианта решения второй задачи:
- а) вызов процедуры в цикле;
 - б) вложенные циклы.
- Какой из них вам больше нравится? Почему?

3. Приведите примеры практических задач, где требуется штриховка. С какими из них вы встречались?

4. Что нужно изменить в последней программе, чтобы сделать горизонтальную штриховку?

Задачи

1. Постройте изображение шахматной доски размером 8×8 клеток.

2. Постройте узоры:

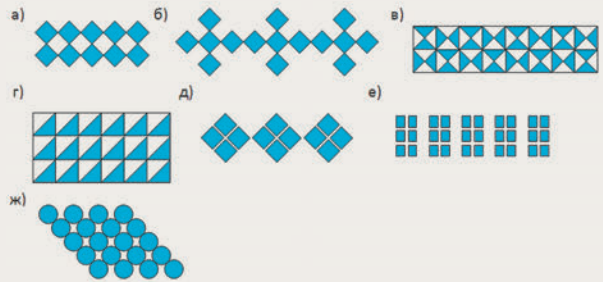


Рис. 79

3. Постройте следующие рисунки (число линий храните в переменной N):

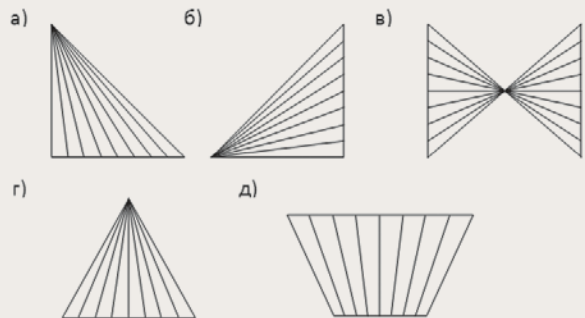


Рис. 80

4. В программе штриховки мы использовали целочисленное деление, чтобы шаг (интервал между линиями) получился целый. В чем недостаток такого подхода? Попробуйте изменить программу так, чтобы он был явно виден. Предложите вариант решения проблемы.

5. Выполните штриховку (число линий храните в переменной N):

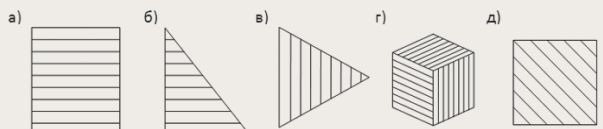


Рис. 81

Тема для сообщения:
“Муаровый эффект”

Анимация

- Ключевые слова:
- анимация
 - кадр
 - смена кадров

- координаты объекта
- текущие координаты
- прозрачный цвет
- пауза

Принципы анимации

Слово “анимация” произошло от латинского слова *animatio*, что означает “оживление”. Анимация на компьютере состоит в том, что простая смена рисунков создает иллюзию движения. Каждый из таких рисунков называют кадром. Допустим, мы нарисовали четыре рисунка одинакового размера:

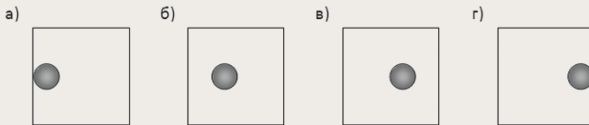


Рис. 82

и меняем их на экране, скажем, через каждую секунду: сначала выводим рисунок а, затем, через 1 с, — рисунок б, еще через 1 с — рисунок в и т.д. Что увидим? Движение шарика слева направо. Конечно, оно не будет плавным, потому что выбран большой интервал времени. Но если менять кадры чаще, чем 16 раз в секунду, глаз перестает замечать смену изображений и видит непрерывное движение.

Однако не всегда можно нарисовать заранее все кадры анимации. Например, в компьютерных играх игрок управляет персонажем с помощью клавиатуры, мыши или джойстика, и поэтому нельзя предсказать, какое изображение возникнет на экране в следующий момент.

Предположим, что нам нужно изобразить движение объекта (например, шарика) на каком-то фоне. Фон может быть одноцветный или в виде картинка. Для перемещения шарика нам нужно

- “стереть” шарик, то есть восстановить фон в том месте, где сейчас нарисован шарик;
- изменить положение шарика (его координаты на холсте);
- запомнить участок фона, который будет испорчен при выводе шарика в новом месте;
- вывести изображение шарика поверх фона.

Если фон одноцветный, то задача упрощается, потому что не нужно запоминать изображение, перекрытое шариком. Чтобы стереть шарик, достаточно залить это место цветом фона.

Используя этот подход, мы напишем программу для моделирования движения шарика на экране.

Анимация движения

Сначала создадим новый холст и закрасим его синим (или каким-нибудь другим) цветом:

```
новый лист ( 200, 200, синий )
```

Размер этого холста — 200 × 200 пикселей.

Шарик будем изображать в виде круга желтого цвета, его радиус — 10 пикселей. Контур рисовать не будем, поэтому в начале программы сразу установим прозрачный цвет пера:

```
перо ( 1, прозрачный )
```

Рисование и стирание шарика можно выполнять одной процедурой, которая принимает три параметра: пару координат (x, y) центра шарика и цвет заливки. Для того чтобы нарисовать шарик, процедуре нужно передать цвет “желтый”, а для стирания — “синий”:

```
алг Шарик ( цел x, y, цвет ц )
```

```
нач
```

```
кисть ( ц )
```

```
окружность ( x, y, 10 )
```

```
кон
```

Теперь подумаем, как использовать эту процедуру. Для хранения текущих координат шарика (то есть координат в данный момент) будем использовать переменные **x** и **y** основной программы. Шарик будет двигаться слева направо по середине холста на уровне $y = 100$ (рис. 83).

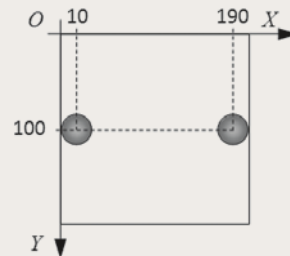


Рис. 83

В начальный момент он находится у левой границы, то есть x -координата его центра равна радиусу (10 пикселей). Шарик выходит за границы экрана, когда расстояние от центра до правой границы холста станет меньше, чем радиус, то есть когда x -координата его центра станет больше, чем $200 - 10 = 190$. В этом случае цикл нужно закончить. Следовательно, цикл выполняется “пока $x \leq 190$ ”. Таким образом, основная программа может выглядеть так:

```
использовать Рисователь
```

```
алг Анимация
```

```
нач
```

```
новый лист ( 200, 200, синий )
```

```
цел x = 100, y = 100
```

```
перо ( 1, прозрачный )
```

```
нц пока x <= 190
```

```
  | движение шарика
```

```
кц
```

```
кон
```

В теле цикла сейчас только комментарий. Там нужно записать все повторяющиеся операции: вывод шарика на экран, скрытие шарика и его перемещение. Для того чтобы нарисовать шарик, нужно просто вызвать процедуру **Шарик**:

```
Шарик ( x, y, желтый )
```

Шарик движется на одноцветном фоне, поэтому для стирания достаточно нарисовать его в том же

месте, задав синий цвет кисти (совпадающий с цветом фона):

```
Шарик ( x, y, синий )
```

Как вы думаете, что произойдет, если нарисовать шарик и сразу же его стереть? Обе эти операции компьютер выполняет очень быстро, поэтому мы даже не заметим, что на экране что-то было нарисовано. Чтобы все-таки увидеть шарик, после того как мы его нарисовали, нужно сделать паузу. Во время этой паузы шарик будет виден на экране. Пауза должна быть небольшая, например, можно выбрать ее длину 20 миллисекунд (1 миллисекунда = 1/1000 секунды). Добавить паузу в программу можно с помощью команды “ждать”:

```
ждать ( 20 )
```

В скобках записано время в миллисекундах.

Остается понять, как сдвинуть шарик. Положение шарика определяется координатами его центра, которые хранятся в переменных **x** и **y**. Поэтому для перемещения шарика достаточно изменить значения этих переменных. Например, чтобы передвинуть шарик вправо на два пикселя, нужно добавить 2 к значению его **x**-координаты:

```
x := x + 2
```

Таким образом, мы можем написать тело цикла:

```
нц пока x <= 190
```

```
  Шарик ( x, y, желтый )
```

```
  ждать ( 20 )
```

```
  Шарик ( x, y, синий )
```

```
  x := x + 2
```

```
кц
```

Обратите внимание, что изменять координаты шарика нужно тогда, когда он скрыт. Иначе мы будем стирать его не в том месте, где до этого нарисовали.

Теперь остается “собрать” всю программу. Вы уже можете сделать это самостоятельно. Не забудьте, что после основной программы нужно поместить процедуру **Шарик**.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Почему в работах профессиональных аниматоров кадры сменяются не реже, чем 16 раз в секунду?
2. Почему обычно перерисовывают не все изображение на экране, а только ту его часть, которая изменилась?
3. Какие проблемы возникают, если перемещать шарик на фоне картинки?
4. Что такое текущие координаты?
5. Как определить, когда шарик коснется края холста?
6. Какие команды в программе определяют скорость перемещения шарика?
7. Как можно ускорить движение шарика по экрану? Предложите два метода.

8. Что нужно изменить в программе, чтобы шарик двигался справа налево? Сверху вниз? По диагонали?

9. Найдите ошибки в основном цикле анимации:

```
нц пока x <= 190
```

```
  Шарик ( x, y, желтый )
```

```
  x := x + 2
```

```
  Шарик ( x, y, синий )
```

```
  ждать ( 20 )
```

```
кц
```

Что будет происходить на экране в этом случае? Проверьте свой ответ экспериментально.

Задачи

1. Измените программу, приведенную в этом параграфе, так, чтобы шарик двигался справа налево.

2. По холсту перемещаются два шарика разного цвета и одинакового радиуса, один — слева направо, второй — справа налево. Скорости движения шариков одинаковые. Разработайте программу для выполнения этой анимации.

3. По холсту перемещаются два шарика разного цвета и разного радиуса, один — слева направо, второй — сверху вниз. Скорости движения шариков разные. Разработайте программу для выполнения этой анимации.

4. *По холсту перемещается шарик, который отскакивает от границ холста. Сначала он движется слева направо, потом, оттолкнувшись от правой границы холста, начинает движение влево и т.д. Разработайте программу для выполнения этой анимации. Используйте бесконечный цикл вида

```
нц пока да
```

```
...
```

```
кц
```

5. *По холсту размером 400 × 200 пикселей перемещается шарик, он движется по диагонали и отскакивает от краев холста. Разработайте программу для выполнения этой анимации.

Тема для сообщения:

“Компьютерная анимация в кино и рекламе”

Управление с помощью клавиатуры

Ключевые слова:

- интерактивность
- скан-код клавиши
- символические имена клавиш
- управление с ожиданием
- управление по требованию

Многие компьютерные игры основаны на том, что игрок управляет персонажем с помощью клавиатуры или мыши. Так обеспечивается интерактивность — взаимодействие между человеком и компьютером. В этом параграфе мы узнаем, как программа может обрабатывать нажатия клавиш на клавиатуре.

Работа с клавиатурой

Давайте подумаем, какие задачи возникают при управлении игрой с клавиатуры. Во-первых, надо уметь определять, была ли нажата какая-нибудь клавиша. В среде КуМир⁹ для этого используется логическая команда **клавиша нажата**. Результат выполнения этой команды — это логическое значение¹⁰: ответ “да” или “нет”. Например, можно использовать такое ветвление:

```
если клавиша нажата то
  | что-то сделать
все
```

В этом случае программа будет реагировать на нажатие любой клавиши.

Вторая задача — определить, какая именно клавиша была нажата. Каждая клавиша на клавиатуре имеет свой числовой код (так называемый скан-код). Для того чтобы узнать скан-код нажатой клавиши, используется команда **код клавиши**:

```
цел к
к := код клавиши
если к = КЛ_ВВЕРХ то
  | передвинуть объект вверх
все
```

Чтобы не запоминать числовые коды всех клавиш, для них введены символьные обозначения. Например, коды клавиш-стрелок обозначаются как **КЛ_ВВЕРХ**, **КЛ_ВНИЗ**, **КЛ_ВПРАВО** и **КЛ_ВЛЕВО** (все буквы заглавные).

Управление с ожиданием

Сначала научимся управлять каким-то объектом, например изображением шарика (см. предыдущий раздел), в режиме ожидания. Это значит, что программа ждет нажатия на клавишу, определяет ее код и после этого перемещает шарик на экране в нужную сторону.

Основной цикл программы на псевдокоде (смеси русского и алгоритмического языков) можно написать так:

```
нц пока да
  Шарик ( к , у , "желтый" )
  к := код клавиши
  Шарик ( к , у , "синий" )
  | переместить шарик
кц
```

Здесь используется цикл с условием **нц пока да**, причем условие (**да**) всегда истинное, поэтому цикл будет работать бесконечно, пока мы не остановим программу.

Для рисования и стирания шарика используется процедура **Шарик**, с которой мы работали в предыдущем параграфе. В строчке

```
к := код клавиши
```

мы сказали программе, что нужно

1) ждать, пока не будет нажата какая-нибудь клавиша;

⁹ Здесь используются названия команд для работы с клавиатурой, которые приняты в тестовой версии КуМир 2х.

¹⁰ Так же, как и для логических команд исполнителя Робот, например, для команды **справа свободно**.

2) когда клавиша нажата, сохранить ее код в переменной **к**.

Обратите внимание, что мы стираем шарик после того, как клавиша нажата (во время ожидания шарик виден на экране).

Остается раскрыть комментарий “переместить шарик”. Как вы уже знаете, нам нужно изменить координаты — значения переменных **х** и **у**. Причем эти изменения будут зависеть от того, какую клавишу нажал пользователь. Например, можно написать четыре условных оператора:

```
если к = КЛ_ВЛЕВО то х := х - 5 все
если к = КЛ_ВПРАВО то х := х + 5 все
если к = КЛ_ВВЕРХ то у := у - 5 все
если к = КЛ_ВНИЗ то у := у + 5 все
```

Получается такая основная программа:

```
использовать Рисователь
алг Управление клавишами
нач
  новый лист ( 200 , 200 , синий )
  цел х = 100 , у = 100 , к
  перо ( 1 , прозрачный )
  нц пока да
    Шарик ( х , у , желтый )
    к := код клавиши
    Шарик ( х , у , синий )
    если к = КЛ_ВЛЕВО то х := х - 5 все
    если к = КЛ_ВПРАВО то х := х + 5 все
    если к = КЛ_ВВЕРХ то у := у - 5 все
    если к = КЛ_ВНИЗ то у := у + 5 все
  кц
кон
```

После нее нужно поместить процедуру **Шарик** из предыдущего параграфа.

Управление по требованию

В только что написанной программе исполнитель ожидал, когда мы нажмем на клавишу, и в это время никаких действий не выполнял. Часто при программировании игр нужен другой режим: события развиваются, но игрок может вмешаться тогда, когда считает нужным. Это управление по требованию. Для того чтобы использовать такое управление, надо определить момент, когда пользователь нажимает на клавишу, и реагировать на это нажатие.

Мы напишем программу, в которой шарик постоянно движется, изменяя свое направление при нажатии клавиш-стрелок. Основной цикл программы выглядит так:

```
нц пока да
  если клавиша нажата
    к := код клавиши
    | изменить направление движения
  все
  Шарик ( х , у , желтый )
  ждать ( 20 )
  Шарик ( х , у , синий )
  | переместить шарик
кц
```

Пока мы еще не решили, как изменять направление движения и как перемещать шарик с учетом направления (эти строчки заменены на комментарии).

Чтобы задать направление движения шарика, нужно определить, как изменяется каждая из его координат на каждом шаге. Будем хранить изменение x -координаты в переменной dx , а изменение y -координаты — в переменной dy . Тогда перемещение шарика записывается в две строчки:

$$x := x + dx$$

$$y := y + dy$$

Куда именно передвинется шарик, зависит от значений dx и dy . Если за один шаг шарик смещается на пять пикселей, получаем такие значения для разных направлений:

Направление	dx	dy
влево	-5	0
вправо	5	0
вверх	0	-5
вниз	0	5
стоп	0	0

Тогда для изменения направления движения при нажатии клавиши достаточно изменить значения dx и dy :

если $k = \text{КЛ_ВЛЕВО}$ то $dx := -5$; $dy := 0$ все

если $k = \text{КЛ_ВПРАВО}$ то $dx := 5$; $dy := 0$ все

если $k = \text{КЛ_ВВЕРХ}$ то $dx := 0$; $dy := -5$ все

если $k = \text{КЛ_ВНИЗ}$ то $dx := 0$; $dy := 5$ все

если $k = \text{КЛ_ПРОБЕЛ}$ то $dx := 0$; $dy := 0$ все

В последней строке мы останавливаем шарик при нажатии на пробел. Эти команды нужно поставить в основной цикл вместо комментария “изменить направление движения”. Полную программу вы уже можете собрать самостоятельно.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие две задачи нужно уметь решать для управления объектом с клавиатуры? Какие задачи невозможно решить, если не будет команды **клавиша нажата**?

2. Что такое скан-код? Верно ли, что при вводе символа “Я” и символа “Z” мы получим один и тот же скан-код клавиши?

3. Что произойдет, если при работе первой программы нажата какая-нибудь другая клавиша, кроме стрелок? Как можно устранить этот недостаток?

4. Как можно при нажатии клавиши изменять радиус шарика?

5. Как можно при нажатии клавиши изменять скорость движения шарика?

6. Зачем во второй программе использована команда **ждать**?

7. Можно ли во второй программе переставить оператор **если ... все**, который обрабатывает нажатие клавиши, в другое место в основном цикле? Ответ обоснуйте.

Задачи

1. Дополните первую программу так, чтобы шарик не мог выйти за пределы холста.

2. Добавьте во вторую программу возможность изменения с клавиатуры

а) скорости движения шарика;

б) радиуса шарика.

3. Добавьте во вторую программу отталкивание шарика от стенок холста.

Тема для сообщения:

“Оператор **выбор** в алгоритмическом языке”

Выводы из прочитанного:

- Алгоритм — это точное описание порядка действий, которые должен выполнить исполнитель для решения задачи. Основные свойства алгоритма — дискретность, понятность и определенность.
- Исполнитель — это человек, животное или машина, которые могут понимать и выполнять некоторые команды. Система команд исполнителя (СКИ) — это набор команд, который понимает и умеет выполнять исполнитель.
- Формальный исполнитель — это исполнитель, который одну и ту же команду всегда выполняет одинаково.
- Алгоритм можно записать в словесной форме, в виде блок-схемы или в виде программы на языке программирования.
- Программа — это алгоритм, записанный на языке конкретного исполнителя. Часто используют псевдокод — смесь естественного языка и языка программирования.
- Ключевые слова — это специальные слова языка программирования, имеющие единственное заранее определенное значение.
- Комментарий — это пояснение к программе. Комментарии не обрабатываются исполнителем.
- Различают три типа ошибок в программах: синтаксические ошибки, отказы и логические ошибки. Для того чтобы найти логические ошибки, используют ручную прокрутку, выполняя алгоритм без исполнителя.
- Для хранения данных используют переменные — величины, значения которых можно изменять во время работы алгоритма.
- Алгоритм решения любой задачи можно составить с помощью следования (линейных алгоритмов), ветвлений (разветвляющихся алгоритмов) и циклов (циклических алгоритмов).
- В линейном алгоритме команды выполняются в том же порядке, в котором они записаны.
- Вспомогательный алгоритм (процедура) — это новая команда, которой мы дополняем СКИ исполнителя. Для того чтобы процедура выполнялась, нужно вызвать ее из основной программы. После завершения работы процедуры управление передается обратно, к следующей команде вызывающей программы.
- Параметры — это данные, которые передаются в процедуру. Каждый параметр имеет имя и тип.
- Существует два метода разработки программ: “снизу вверх” и “сверху вниз”.
- При использовании метода “снизу вверх” сначала составляются вспомогательные алгоритмы, а затем из них строится основная программа.
- В методе “сверху вниз” (методе последовательного уточнения) задача разбивается на подзадачи, каждая из подзадач оформляется в виде вспомогательного алгоритма. Сначала составляется основная программа, а затем все вспомогательные алгоритмы.
- Циклический алгоритм — это алгоритм, в котором некоторая последовательность действий (тело цикла) выполняется несколько раз. Существует два вида циклов: циклы с известным числом шагов и циклы с условием.
- Цикл с условием — это цикл, который выполняется до тех пор, пока некоторое условие не станет ложным. Количество шагов такого цикла определяется исходными данными. Если в цикле с условием сделана ошибка, программа может заиклиться.
- Вложенный цикл — это цикл, расположенный внутри другого цикла.
- Логическая команда — это вопрос, на который исполнитель отвечает “да” или “нет”.
- Разветвляющийся алгоритм — это алгоритм, в котором последовательность действий изменяется в зависимости от выполнения некоторых условий.
- Диалоговая программа — это программа, в которой исходные данные вводятся человеком с клавиатуры, а результаты работы выводятся на экран.
- Программы, работающие в графическом режиме, могут управлять отдельно каждым пикселем области рисования — холста.
- Графический примитив — это геометрическая фигура, которая добавляется на рисунок с помощью одной команды. При рисовании примитивов свойства линии определяются объектом перо, а свойства заливки — объектом кисть.
- Анимация — это создание иллюзии движения на экране. Компьютерная анимация — это быстрая смена рисунков (кадров). Для того чтобы переместить изображение объекта на фоне, нужно скрыть его, изменить координаты и снова вывести на холст. Если фон — это картинка, перед рисованием объекта нужно запомнить часть холста, которая будет изменена.
- Для управления с клавиатуры используют две функции. Одна из них дает ответ на вопрос “нажата ли какая-нибудь клавиша”, а вторая определяет код нажатой клавиши. Каждая клавиша на клавиатуре имеет свой код, который называют скан-кодом.



Гармонический ряд

Д.М. Златопольский,
г. Москва

► Гармонический ряд в математике представляет собой сумму, составленную из бесконечного количества членов, обратных последовательным числам натурального ряда:

$$\sum_{k=1}^{\infty} \frac{1}{k} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots + \frac{1}{k} + \dots$$

Ряд назван *гармоническим*, так как каждый его член, начиная со второго, является гармоническим средним двух соседних. В свою очередь, *средним гармоническим* двух¹ положительных чисел x_1 и x_2 называется число, обратное среднему арифметическому их обратных, т.е. число, равное:

$$\frac{2}{\frac{1}{x_1} + \frac{1}{x_2}}$$

¹ Для n чисел x_1, x_2, \dots, x_n формула для расчета их среднего гармонического:

$$\frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \dots + \frac{1}{x_n}}$$

N -я частичная сумма гармонического ряда, равная

$$H_n = \sum_{k=1}^n \frac{1}{k},$$

называется n -м гармоническим числом.

Вот несколько первых значений частичных сумм (гармонических чисел):

$$H_1 = 1$$

$$H_2 = \frac{3}{2} = 1,5$$

$$H_3 = \frac{11}{6} \approx 1,833$$

$$H_4 = \frac{25}{12} \approx 2,083$$

Гармонический ряд растет (сходится) очень медленно, — для того чтобы частичная сумма превысила 100, необходимо около 10^{43} элементов ряда.

Рассчитать значение H_n при заданном n можно с помощью такого фрагмента программы на школьном алгоритмическом языке:

```
сумма := 1
нц для i от 2 до n
    сумма := сумма + 1/i
кц
```

Учитывая, что $H_n = H_{n-1} + \frac{1}{n}$, можно применить прием, который в программировании называют “рекурсией” [1]. Создадим простую и логичную рекурсивную (использующую саму себя в качестве вспомогательной) функцию:

```

алг вещ Гармоническое_число (арг цел n)
нач
  если n = 1
  то
    знач := 1 |Значение функции
  иначе
    знач := Гармоническое_число(n - 1) + 1/n
  все
кон

```

Задания для самостоятельной работы

1. Разработайте программу (на языке программирования, который вы изучаете) для расчета среднего гармонического n чисел x_1, x_2, \dots, x_n .

2. Докажите с помощью формул справедливость того факта, что каждый член гармонического ряда, начиная со второго, является гармоническим средним двух соседних.

3. Разработав компьютерную программу, определите, при каком n значение H_n превысит 5, 6, ..., 21.

4. Оформите лист электронной таблицы, с помощью которого можно рассчитать значения первых 30 частичных сумм гармонического ряда:

1	1
2	1,5
3	1,833333
4	2,083333
	...
30	3,994987

Постарайтесь вручную ввести как можно меньше формул (используйте копирование последних).

5. Нетрудно увидеть, что оба приведенных выше варианта расчета n -го гармонического числа возвращают вещественное значение. Разработайте программу, в которой значения первых 20 гармонических чисел выводятся на экран в виде простой дроби:

1	1/1
2	3/2
3	11/6
...	
20	...

6. В 1740 году Леонардом Эйлером было получено выражение для приближенного расчета суммы первых n членов гармонического ряда:

$$S_n = \ln(n) + \gamma + \varepsilon_n,$$

— где

$\gamma = 0,5772$,

\ln — натуральный логарифм,

ε_n — погрешность.

Пример использования формулы Эйлера при $n = 10$:

$$S_n = \sum_{k=1}^n \frac{1}{k} = 2,93;$$

$$\ln(n) + \gamma = 2,88;$$

$$\varepsilon_n = 0,05 \text{ (1,7 \%)}.$$

При $n \rightarrow \infty$ $\varepsilon_n \rightarrow 0$.

Определите значение погрешности ε_n при $n = 25$.

7. Так называемый “знакопеременный ряд” отличается от гармонического тем, что в нем..., впрочем, вы и сами все увидите:

$$1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \dots$$

Разработайте программу для расчета суммы n первых членов такого ряда. Оформите два варианта программы:

1) в котором используется условный оператор;

2) без использования условного оператора.

Определите также примерное значение (с точностью до сотых), к которому стремится значение суммы при больших значениях n (например, при $n = 5000 - 5200$).

Ответы, пожалуйста, присылайте в редакцию (можно выполнять не все задания). Фамилии всех приславших ответы будут опубликованы, а лучшие ответы мы поощрим.

Литература

1. Златопольский Д.М. Рекурсия — эффектно, но не всегда эффективно. / “В мир информатики” № 199 (“Информатика” № 7–8/2014).

Культ наук в самом высоком смысле этого слова, возможно, еще более необходим для нравственного, чем материального процветания нации. Наука повышает интеллектуальный и моральный уровень; наука способствует распространению и торжеству великих идей.

Луи Пастер

Где господствует дух науки, там творится великое...

Николай Иванович Пирогов

Память есть кладовая ума, в которой много перегородок, а потому надо все скорее укладывать, куда следует.

Александр Васильевич Суворов



ИСТОРИЯ ИНФОРМАТИКИ

Кто первым описал двоичную систему?

Д.М. Златопольский,
директор музея истории
вычислительной техники
гимназии № 1530 г. Москвы,
Марат Эйнуллаев,
ученик гимназии № 1530
г. Москвы

► В статье [1] отмечалось, что, хотя в большинстве источников можно найти информацию о том, что современная двоичная система счисления была впервые описана в 1703 году великим немецким ученым Готфридом Вильгельмом Лейбницем в статье “Explication de l’Arithmétique Binaire”, это не совсем так. В качестве примера указывались работы Томаса Хэрриота, английского математика, географа и астронома (1560–1621).

Он оставил несколько тысяч страниц неопубликованных рукописей, правда, об этом не было известно вплоть до появления в 1951 г. работы [2]. Выдержки из последней статьи приведены в книге [3].

В [1] приведены таблицы Хэрриота с разными вариантами представления чисел, в том числе в виде суммы степеней двойки, с использованием символов “+” и “-”, цифр 0 и 1.

В записях Хэрриота имеются также примеры перевода чисел из десятичной системы счисления в двоичную и обратно и примеры вычислений в двоичной системе — сложения, вычитания и умножения, методы которых аналогичны “современным” [4]¹.

Так, на рис. 1 показан пример перевода десятичного числа 109 в двоичную систему. Видно, что



Томас Хэрриот

применен метод перевода, который в настоящее время называется “метод выделения максимальных степеней двойки” (цифрами справа показаны показатели степеней, которые потом используются для записи единиц в двоичном представлении).

*Conversio*²

$$\begin{array}{r|l}
 109 & \\
 \underline{64} & 7 \\
 45 & \\
 \underline{32} & 6 \\
 13 & \\
 \underline{8} & 4 \\
 5 & \\
 \underline{4} & 3 \\
 1 & 1
 \end{array}
 \qquad
 \begin{array}{l}
 1101101
 \end{array}$$

Рис. 1

На рис. 2 приведен пример преобразования двоичного числа 1101101 в десятичное (происходит сложение весомостей разрядов — степеней двойки в разрядах с единицей).

Reductio

$$\begin{array}{r}
 1101101 \\
 64 \\
 32 \\
 8 \\
 4 \\
 \underline{1} \\
 109
 \end{array}$$

Рис. 2

На рис. 3–5 представлены примеры, соответственно, вычитания, сложения и умножения двоичных чисел.

Subductionis exempla

$$\begin{array}{r}
 10110010 \\
 \underline{111011} \\
 1110111
 \end{array}$$

$$\begin{array}{r}
 10101001 \\
 \underline{110111} \\
 1110010
 \end{array}$$

Рис. 3

¹ Перевод фрагментов этой работы на русский язык и их анализ выполнил второй автор данной статьи.

² Все заголовки примеров приводятся на латыни, как в оригинале.

Additionis exempla

```
  111011
+ 1110111
-----
101110010
```

```
  110111
+ 1110010
-----
10101001
```

Рис. 4

Multiplicatio

```
  1101101
  1101101
  1101101
  1101101
  1101101
  1101101
+ 1101101
-----
10111001101001
```

Рис. 5

Обратим внимание на отсутствие в приведенных примерах знаков операций.

Описан также оригинальный метод умножения, название которого на латыни можно перевести как “другой метод — последовательного сложения” (рис. 6).

Aliter, cum additione successiva

```
  1101101
  1101101
  1101101
+ 10001000
+ 10110001
+ 10011001
+ 10111001
-----
10111001101001
```

Рис. 6

Анализ показывает, что при вычислениях по этому методу происходит “снос” одной или двух цифр первого множителя в результат и сложение “оставшегося” числа с первым множителем. Затем последовательно с первым множителем складываются промежуточные суммы (также без одной или двух цифр). После последнего сложения получается искомый результат.

Опишем метод Хэрриота подробно.

1. Первый множитель умножается на первую справа цифру второго множителя.

2. Так как во втором множителе вторая справа цифра — 0, то в результат “сносятся” две последние цифры числа, полученного на этапе 1 (01).

3. Оставшееся без “сношенных” цифр только что указанное число (11011) складывается с первым множителем — получается 10001000.

4. Так как во втором множителе четвертая справа цифра — 1, то в результат “сносится” одна последняя цифра полученной суммы (0).

5. Оставшееся число (1000100) складывается с первым множителем — получается 10110001.

6. Так как во втором множителе пятая справа цифра — 0, то в результат “сносятся” две последние цифры полученного результата (01).

7. Оставшееся число (101100) складывается с первым множителем — получается 10011001.

8. Так как во втором множителе седьмая справа цифра — 1, то в результат “сносится” одна последняя цифра полученного результата (1).

9. Оставшееся число (1001100) складывается с первым множителем — получается 10111001.

10. Все цифры второго множителя учтены — вся последняя сумма “сносится” в итоговый результат.

Можно сказать, что определенными преимуществами этого метода является то, что складываются только по два числа, и что в результате иногда получаются сразу две цифры (при традиционном методе умножения, показанном на рис. 5, сначала многократно выписывается первый множитель, после чего результат определяется однократным сложением всех цифр по разрядам).

Все вышеизложенное позволяет сделать вывод о том, что Томас Хэрриот описал использование двоичной системы за много лет до Лейбница.

Задание для самостоятельной работы

Определите результат умножения по “альтернативному” методу двух двоичных чисел — 11011011 и 10101. Ответ (с промежуточными вычислениями, как на рис. 6) присылайте в редакцию.

Литература

1. Томас Хэрриот и двоичная система. / “В мир информатики” № 193 (“Информатика” № 1/2013).
2. Shirley J.W. Binary numeration before Leibniz. / American Journal of Physics, 1951. Vol. 19.
3. Glaser Anton. History of binary and other nondecimal numeration. 1981.
4. history.mcs.st-andrews.ac.uk.

ПОИСК ИНФОРМАЦИИ

Три (и даже больше) вопроса

1. Представители какого кавказского народа принимали участие в битве, произошедшей 15 июля 1410 года?

2. Где проходит единственное в мире соревнование автомобильных команд Формулы-1, проведение которого занимает не три, а четыре дня?

3. Какова высота борта судна, показанного на рисунке, и сколько винтов оно имеет?



“Ведьма” Аньези

Аньези — фамилия итальянского математика Марии-Гаетаны Аньези (1718–1799). Но она, конечно, не ведьма ☺. Правильным было бы назвать статью — “Верзьера Аньези”. Дело в том, что верзьера (versera) — это название линии (ее особенности мы рассмотрим ниже), которое в 1718 году предложил другой итальянский ученый, Гвидо Гранди. Это слово, не смущаясь его двусмысленностью (по-итальянски “верзьера” — ведьма), он произвел от термина *sinus versus* (обращенный синус).

М.-Г. Аньези же рассматривала эту линию в руководстве по высшей математике (1748), пользовавшемся в свое время широким распространением. По ее имени (точнее — по фамилии) и названа кривая.

Если на отрезке OA , как на диаметре, построена окружность и полухорда BC продолжена до точки M , определяемой из пропорции $\frac{BM}{BC} = \frac{OA}{OB}$, то когда точка C описывает построенную окружность, точка M описывает линию, которая и является верзьерой Аньези (см. рис. 1).

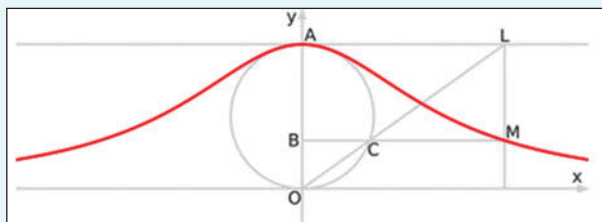


Рис. 1

Приведем программу, с помощью которой можно получить на экране указанную “ведьму” — извините, линию ☺.

Как и в статьях [1–2], используем школьный алгоритмический язык. Однако в данном случае значения координат точек кривой будем получать не из параметрических уравнений, а рассчитаем значения y при различных значениях x , учитывая уравнение верзьеры Аньези:

$$y = \frac{a^3}{a^2 + x^2},$$

где $a = OA$ — диаметр окружности на рис. 1.

Программа для построения очень простая:

алг Верзьера_Аньези

нач цел x , y , вещ a

вывод "Задайте значение диаметра a "

ввод a

| Устанавливаем графический режим

...

| Для всех значений x на экране

нц для x от $-\text{int}(\text{макс}X/2)$ до $\text{int}(\text{макс}X/2)$

| Рассчитываем координату y

$y := \text{int}(\text{макс}Y/2 - a * a * a /$
 $(a * a + x * x))$

| и изображаем соответствующую точку
точка $(\text{int}(\text{макс}X/2) + x, y)$

кц

кон

Примечания

1. Величины $\text{макс}X$ и $\text{макс}Y$ — максимальное значение координат x и y (соответственно) в выбранном режиме работы экрана.

2. Ось y на экране направлена вниз, поэтому в формуле для расчета координаты y использован знак “минус”.

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

Для построения верзьеры Аньези можно также использовать электронную таблицу Microsoft Excel или другую подобную программу. Фрагмент листа, на котором решается эта задача, показан на рис. 2 (необходимые формулы запишите самостоятельно, при этом используйте абсолютную ссылку на ячейку E1 и копирование формул).

	A	B	C	D	E
1	x	y		a =	50
2	-100	10,00			
3	-99	10,16			
4	-98	10,33			
...					
101	-1	49,98			
102	0	50,00			
103	1	49,98			
...					
200	98	10,33			
201	99	10,16			
202	100	10,00			

Рис. 2

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

Задания для самостоятельной работы

1. На языке программирования, который вы изучаете, разработайте программу, с помощью которой можно получить изображение линии на рис. 1.

2. Решите эту же задачу, используя электронную таблицу.

Указания по выполнению

Форма кривой получается аналогичной показанной на рис. 1 при значениях a , не превышающих 70–80. При использовании электронной таблицы следует учесть, что в ней на диаграммах и графиках масштаб по осям x и y в общем случае не совпадает. Сделать его одинаковым можно, меняя размеры области диаграммы (пример показан на рис. 3).

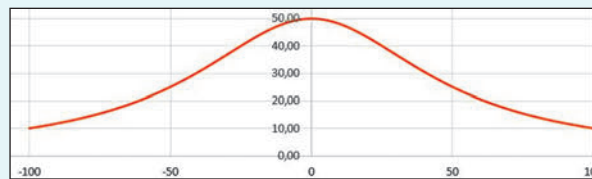


Рис. 3

В заключение заметим, что в некоторых источниках приводится название рассмотренной кривой — “локон Аньези”, не имеющее под собой никаких оснований.

Литература

1. О кардиоиде и о сердце. / “В мир информатики” № 203 (“Информатика” № 12/2014).
2. Еще две замечательные кривые. / “В мир информатики” № 204 (“Информатика” № 1/2015).
3. Трилистник и другие. / “В мир информатики” № 205 (“Информатика” № 2/2015).

Кто какой язык изучает?

Вадим, Сергей и Михаил изучают различные иностранные языки: китайский, японский и арабский. На вопрос, какой язык изучает каждый из них, один ответил: “Вадим изучает китайский, Сергей не изучает китайский, а Михаил не изучает арабский”. Впоследствии выяснилось, что в этом ответе только одно утверждение верно, а два других ложны. Какой язык изучает каждый из молодых людей?

Четырехзначные числа

Сколько четырехзначных чисел в двоичной системе счисления? Задание предназначено для учащихся 1–7-х классов.

Компьютер на яхте

Некий любитель приключений отправился в кругосветное путешествие на яхте, оснащенной бортовым компьютером. Его предупредили, что чаще всего выходят из строя три узла компьютера — a , b , c , и дали необходимые детали для замены. Выяснить, какой именно узел надо заменить, он может по сигнальным лампочкам на контрольной панели. Лампочек тоже ровно три: x , y и z . Инструкция по выявлению неисправных узлов такова:

- 1) если неисправен хотя бы один из узлов компьютера, то горит по крайней мере одна из лампочек x , y , z ;
- 2) если неисправен узел a , но исправен узел c , то загорается лампочка y ;
- 3) если неисправен узел c , но исправен узел b , загорается лампочка y , но не загорается лампочка x ;
- 4) если неисправен узел b , но исправен узел c , то загораются лампочки x и y или не загорается лампочка x ;
- 5) если горит лампочка x и при этом либо неисправен узел a , либо все три узла a , b , c исправны, то горит и лампочка y .

В пути компьютер сломался. На контрольной панели загорелась лампочка x . Тщательно изучив инструкцию, путешественник починил компьютер. Но с этого момента и до конца плавания его не оставляла тревога. Он понял, что инструкция несовершенна и есть случаи, когда она ему не поможет. Какие узлы заменил путешественник?

Цветные удочки

Пятеро друзей — Дима, Тима, Фима, Сима и Клим — решили купить себе удочки. Удочки в магазине были пяти цветов: красного, синего, белого, зеленого и черного.

Известно, что:

- 1) Дима любит красный и синий цвета;
- 2) Симе понравились синяя и зеленая удочки;
- 3) Фима купил зеленую удочку;
- 4) Клим отдал предпочтение красной, синей и черной удочкам.

Кто какую удочку купил, если у всех ребят оказались удочки разного цвета?

Литература

1. Богомолова О.Б. Логические задачи. М.: Бинном. Лаборатория знаний, 2005.

Площадь комнаты

Житель планеты \acute{a} -Кентавр пишет, что размеры его комнаты 4×12 м, а площадь равна 53 квадратных метра. Как такое могло быть?

Четыре задачи на перестановки цифр

1. Дано число 15, записанное в некоторой системе счисления. В какой системе при перестановке его цифр значение увеличивается в три раза?

2. Какое двузначное число, записанное в 9-ричной системе счисления, при перестановке его цифр увеличивается:

- а) в 4 раза;
- б) в 1,5 раза?

3. Есть ли такое восьмеричное число, которое при перестановке его цифр увеличивается в 1,25 раза?

Ответы (можно не на все задачи) присылайте в редакцию.

Сколько лет детям?

В семье четверо детей — Аня, Боря, Вера и Галя. Им 5, 8, 13 и 15 лет. Сколько лет каждому ребенку, если одна девочка ходит в детский сад, Аня старше Бори, и сумма лет Ани и Веры кратна трем?

Дополнительные списки

читателей, приславших ответы на задания, опубликованные в октябрьском выпуске “В мир информатики”

Конкурс № 111 “Переpravы”, тур 2

— Байкова Римма, средняя школа села Восточное Нижегородской обл., учитель Долгова Г.А.;

— Лежнева Александра, г. Пенза, школа № 512, учитель Гаврилова М.И.;

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

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

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

Задача “Семь кошельков”

— Артюхова Татьяна, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

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

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

Задача “Отмеривание кваса”

— Квасов Антон и Мамыкина Елена, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

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

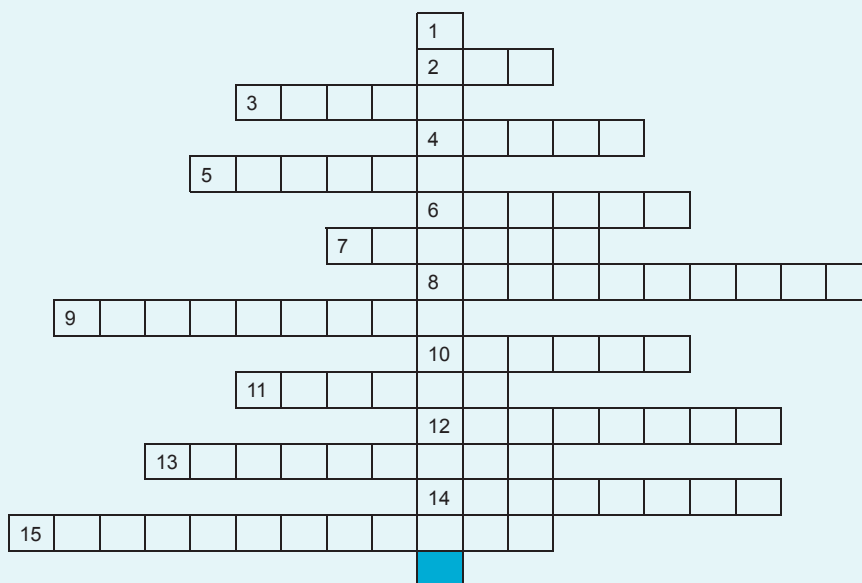
— Рыжиков Антон, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**

“ЛОМАЕМ” ГОЛОВУ

Три кроссворда

Решите, пожалуйста, три кроссворда. Первый разработала Н.Р. Рябченко, учитель информатики школы № 6 станицы Барсуковская Кочубеевского р-на Ставропольского края, второй — ее ученица Нонна Коломина, третий подготовлен редакцией на основе второго.

Кроссворд № 1 “Елочка”



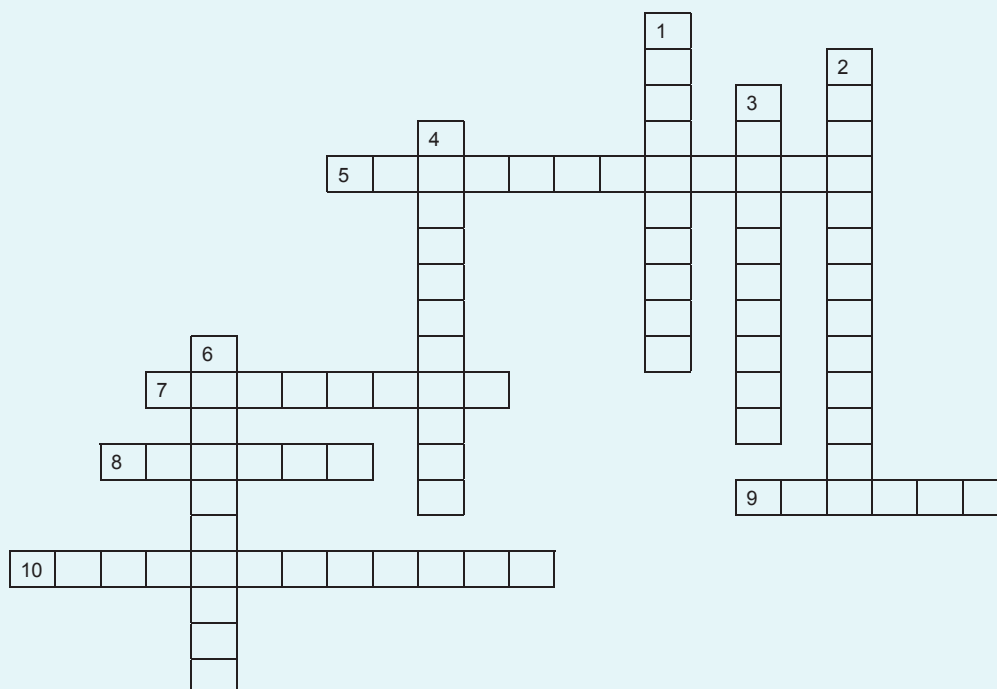
По горизонтали

2. Единица измерения информации.
3. Совокупность объектов, объединенных по какому-либо признаку.
4. Любое словесное высказывание, напечатанное, написанное или существующее в устной форме.
5. Устройство ввода графической информации.
6. Мысленное разделение объекта на составные части или выделение признаков объекта.
7. Наука, изучающая законы и формы мышления, способы рассуждений и доказательств.
8. Сведения об окружающем нас мире.
9. Универсальное электронное устройство для работы с информацией.
10. Часть окружающей нас действительности (конкретный предмет, процесс и т.д.).
11. Граф, иллюстрирующий иерархическую систему.
12. Конечная последовательность шагов в решении задачи, приводящая к требуемому результату.
13. Мысленное объединение однородных объектов в некоторый класс.
14. Тот (то), кто (что) передает информацию.
15. Человек, пользующийся услугами компьютера для получения информации или решения задачи.

По вертикали

1. Мысленное выделение одних признаков объекта и отвлечение от других.

Кроссворд № 2



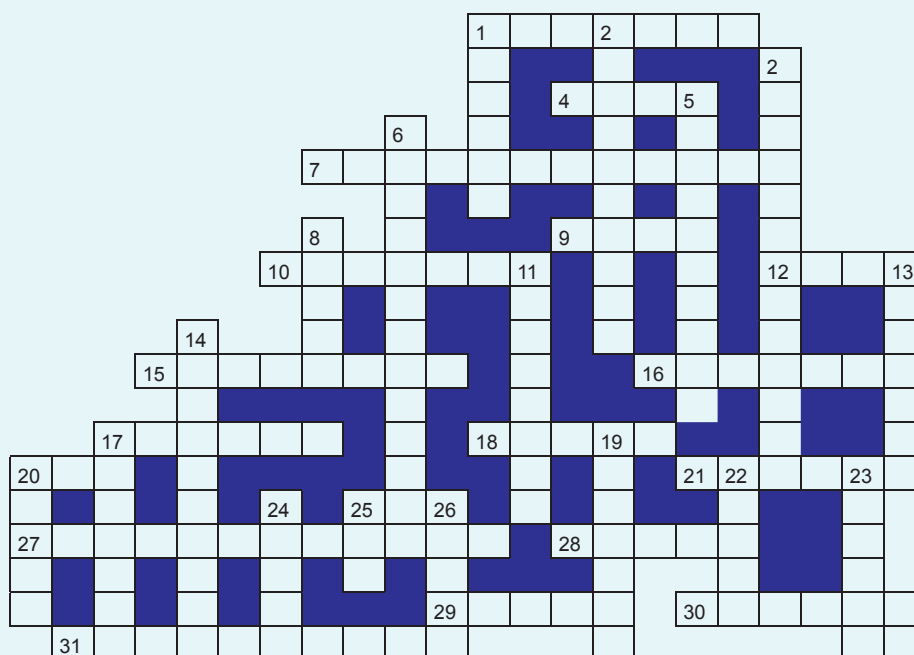
По горизонтали

5. Предложение на любом языке, содержание которого определяется однозначно как истинное или ложное.
7. Описание последовательности действий, приводящих к требуемому результату.
8. Объект, используемый в качестве “заместителя” другого объекта (оригинала) с определенной целью.
9. Место пересечения строки и столбца в электронных таблицах.
10. Замена реального объекта его формальным описанием.


По вертикали

1. Процесс целенаправленного воздействия на объект.
2. Метод познания, заключающийся в создании и исследовании моделей.
3. Логическая операция.
4. Объект, способный выполнять определенный набор команд.
6. Алгоритмическая конструкция, соответствующая линейному алгоритму.

Кроссворд № 3



По горизонтали

1. Единица измерения времени.
4. ...информатики в школе.
7. Предложение на любом языке, содержание которого определяется однозначно как истинное или ложное.
9. Изображаемый на экране список вариантов, из которых пользователь выбирает необходимый вариант.
10. Электронная...
12. Валюта, в которой получают зарплату иранские программисты.
17. Объект, используемый в качестве “заместителя” другого объекта (оригинала) с определенной целью.
18. Взломщик компьютерных программ.
20. Цифра двенадцатеричной системы счисления.
21. Место пересечения строки и столбца в электронных таблицах.
27. Замена реального объекта его формальным описанием.
28. Предварительное оповещение о каком-либо событии (концерте, спектакле, показе кинофильма, выпуске книги или компьютерной программы и т.д.).
29. В текстовом редакторе Microsoft Word — текст, набранный до нажатия клавиши .
30. Непрерывная последовательность данных.
31. “Сестра” информатики.

По вертикали

1. Условный знак или физический процесс, передающие некоторую информацию.
 2. Процесс целенаправленного воздействия на объект.
 3. Метод познания, заключающийся в создании и исследовании моделей.
 5. Логическая операция.
 6. Объект, способный выполнять определенный набор команд.
 8. Процесс написания текста на компьютере, а также полный комплект чего-либо.
 11. Участник почтовой переписки (отправитель).
 13. Часть окна текстового редактора, используемая для установки полей, отступов и т.п., а также измерительный инструмент.
 14. Алгоритмическая конструкция, соответствующая линейному алгоритму.
 15. Описание последовательности действий, приводящих к требуемому результату.
 16. Периодическое изменение цвета или яркости изображения на экране.
 17. Так называют двумерный массив.
 19. Цифра шестнадцатеричной системы счисления.
 20. Знак препинания.
 22. Величина, с помощью которой осуществляется счет.
 23. Элемент манипулятора “мышь”.
 24. Знак, обозначающий число.
 25. Величина изменения значения переменной цикла.
 26. ...“Intel”.
- Ответы (можно на отдельные кроссворды и не на все термины) присылайте в редакцию.

Опять — двуносый чайник

После небольшого перерыва, связанного с публикацией в предыдущем выпуске задачи о разливе молока, мы опять предлагаем задачи на использование чайника с двумя носиками.



Дизайнер чайника — Дороти Грей

Как, используя двуносый чайник, полностью заполнить три чашки:

- 1) вместимостью 3, 4 и 7 условных единиц;
- 2) вместимостью 4, 5 и 18 условных единиц?

Во всех случаях принять, что в чайнике имеется достаточно большое количество воды.

Алгоритмы решения задач, пожалуйста, оформите в виде таблицы:

№	Действие	В 1-й чашке	Во 2-й чашке	В 3-й чашке
Исходное состояние		0	0	0
1	Налить в ...			
2				
...				

Расставить цифры

Расставьте цифры от 1 до 6 в пустые клетки в соответствии с указанными знаками так, чтобы в каждой строке и в каждом столбце все цифры были разными. Некоторые цифры уже стоят в нужных клетках.

	>	2	<		>	1	<		<	
∨	■	∧	■	∨	■	∧	■	∧	■	∧
	<	3	>		<		>		<	
∧	■	∧	■	∧	■	∨	■	∧	■	∨
	<		<		>		<		>	1
∨	■	∨	■	∨	■	∧	■	∨	■	∧
	>	1	<		<		<	5	<	
∧	■	∧	■	∧	■	∨	■	∨	■	∨
6	>		>		>		>	1	<	2
∨	■	∧	■	∨	■	∧	■	∧	■	∧
	<		>		<		>		<	

Что сказал старик? (старинная задача)

Два молодых запорожских казака, Грицко и Опанас, оба лихие наездники, часто бились между собой об заклад, кто кого перегонит. Не раз то один, то другой был победителем. Наконец, это им надоело.

— Вот что, — сказал Грицко, — давай спорить наоборот: пусть заклад достанется тому, чья лошадь придет в назначенное место вторым, а не первым.

— Ладно, — ответил Опанас.

Казаки выехали на своих конях в степь. Зрителей собралось множество — все хотели посмотреть на такую диковинку. Один старый казак начал считать “Раз, два, три!”. Спорщики, конечно, ни с места.

Зрители стали смеяться, судить да рядить и порешили, что такой спор невозможен и что спорщики простоят на месте, как говорится, до скончания века.

Тут к толпе подошел седой старик, выдавший на своем веку многое.

— В чем дело? — спросил он.

Ему сказали.

— Эге ж, — говорит старик, — вот я им сейчас скажу такое, что они поскачут, как ошпаренные.

И действительно, подошел старик к казакам, сказал что-то — и через полминуты казаки уже неслись по степи во всю прыть, стараясь обогнать друг друга. Но заклад, как и договорились, все-таки выиграл тот, чья лошадь пришла второй.

Что сказал казакам старик?

Числовой ребус с “АМУРом”

Решите, пожалуйста, числовой ребус $AMYP \times P = *MYAP$. Как принято в таких головоломках, в нем

одинаковыми буквами зашифрованы одинаковые цифры, разными буквами — разные цифры, а звездочкой (“*”) может быть любая цифра.

Еще раз обратим внимание на то, что редакция имеет в виду решение числовых ребусов методом рассуждений (при возможном переборе нескольких значений).

Числовые ребусы в троичной системе. Часть 7

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

$$\begin{array}{r}
 1. \\
 M \ 1 \\
 + 1 \\
 \hline
 * \ * \ *
 \end{array}$$

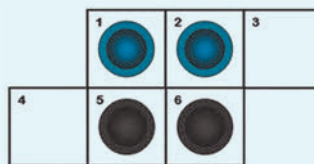
$$\begin{array}{r}
 2. \\
 A \ B \\
 + B \\
 \hline
 * \ *
 \end{array}$$

$$\begin{array}{r}
 3. \\
 C \ D \\
 + D \\
 \hline
 * \ 0 \ *
 \end{array}$$

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

Шесть клеток и четыре фишки

На шести клетках размещены две синие и две черные фишки (см. ниже). Как за минимальное число перемещений поменять местами синие и черные фишки? Разрешается двигать фишки только на смежное пустое место.



Алгоритм решения задачи, пожалуйста, оформите в виде:

1. С-2-3 (синюю фишку переместить с клетки 2 в клетку 3).

2. ...

Красный куб

Представьте себе деревянный куб со сторонами 30 см, вся поверхность которого окрашена в один красный цвет, и ответьте, пожалуйста, на вопросы:

1) сколько потребуется разрезов, чтобы разделить куб на кубики со стороной 10 см;

2) сколько получится таких кубиков;

- 3) сколько кубиков будут иметь по четыре окрашенные грани;
- 4) сколько кубиков будут иметь по три окрашенные грани;
- 5) сколько кубиков будут иметь по две окрашенные грани;
- 6) сколько кубиков будут иметь по одной окрашенной грани;
- 7) сколько кубиков будут неокрашенными?

ШКОЛА ПРОГРАММИРОВАНИЯ

*Когда человек хочет передвинуть гору,
он начинает с того,
что убирает маленькие камни.*

Лестница из чисел

Обсудим методику решения ряда задач программирования. Как принято в нашем издании, анализ будем проводить с использованием школьного алгоритмического языка (система КуМир).

Задача 1

Составить программу, которая выводит на экран следующее (n — заданное число):

```
1
2 2
3 3 3
...
n n ... n
```

Комментарии

Следует использовать программную конструкцию, которая называется “вложенный цикл” (“цикл в цикле”).

Параметр i “наружного” оператора цикла меняется от 1 до n (числа в строках).

Для чисел в каждой строке число повторений равно соответствующему числу, то есть конечное значение параметра “внутреннего” оператора цикла равно i .

Программа

```
алг Задача_1
нач цел n, i, j
  ввод n
  нц для i от 1 до n
    нц для j от 1 до i
      вывод i, " "
    кц
  вывод нс
кц
кон
```

Задача 2

Составить программу, которая выводит на экран следующее (n — заданное число):

```
1
1 2
1 2 3
1 2 3 4
...
1 2 ... n
```

Найти закон формирования последовательности

Найдите закон формирования следующей последовательности:

MN, MMN, MNN, MMMN, MNNN, ...

Закон опишите словесно и в виде формул.

Комментарии

Здесь также следует применить вложенный цикл.

Параметр i “наружного” оператора цикла меняется от 1 до n (номер строки).

В каждой строке выводятся числа от 1 до i .

Программа

```
алг Задача_2
нач цел n, i, j
  ввод n
  нц для i от 1 до n
    нц для j от 1 до i
      вывод j, " "
    кц
  вывод нс
кц
кон
```

Задача 3

Составить программу, которая выводит на экран n строк (n — заданное число) в виде:

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
16 17 18 19 20 21
```

Комментарии

В данной задаче нужно знать первое число в каждой строке. Если эту величину обозначить *перв*, то ее значение для i -й строки можно найти, увеличив значение для предыдущей строки на i .

В каждой строке выводятся i чисел, начиная с числа *перв*.

Программа

```
алг Задача_3
нач цел n, перв, i, j
  ввод n
  перв := 1 | Начальное значение
  нц для i от 1 до n
    нц для j от перв до перв + i - 1
      вывод j, " "
    кц
    вывод нс
    перв := перв + i
  кц
кон
```




Задача 4

Составить программу, которая для n строк, описанных применительно к задаче 3, выводит на экран номер строки, следующей за строкой с заданным числом.

Комментарии

Обозначим заданное число — k , номер строки — y .

Так как определять первое число в каждой строке мы научились, решая предыдущую задачу, то для нахождения искомого номера строки можно использовать оператор цикла с условием, в частности, с предусловием вида $перв \leq k$:

Программа

```

алг Задача_4
нач цел  $k$ , перв,  $y$ 
  ввод  $k$ 
  перв := 1
   $y$  := 1
  нц пока перв <=  $k$ 
    |Переходим к следующей строке
    перв := перв +  $y$ 
     $y$  :=  $y$  + 1
  кц
вывод  $y$ 
кон
  
```

Примечание. Обратим внимание на то, что величина n в программе не используется.

Задача 5

Составить программу, которая для n строк, описанных применительно к задаче 3, выводит на экран номер строки, в которой находится заданное число.

Комментарии

Задача решается аналогично предыдущей, но после окончания работы оператора цикла искомый номер строки должен быть уточнен.

Программа

```

алг Задача_5
нач цел  $k$ , перв,  $y$ 
  ввод  $k$ 
  перв := 1
   $y$  := 1
  нц пока перв <=  $k$ 
    перв := перв +  $y$ 
     $y$  :=  $y$  + 1
  кц
  |Заданное число находится
  |в предыдущей строке
   $y$  :=  $y$  - 1
вывод  $y$ 
кон
  
```

Задача 6

Составить программу, которая для n строк, описанных применительно к задаче 3, выводит на экран первое число строки, в которой находится заданное число.

Комментарии

Задача решается аналогично предыдущей, но после окончания работы оператора цикла следует уточнить искомое значение $перв$.

Программа

```

алг Задача_6
нач цел  $k$ , перв,  $y$ 
  ввод  $k$ 
  перв := 1
   $y$  := 1
  нц пока перв <=  $k$ 
    перв := перв +  $y$ 
     $y$  :=  $y$  + 1
  кц
  |Уточняем значение перв
  перв := перв - ( $y$  - 1)
  |Можно также так:
  | $y$  :=  $y$  - 1
  |перв := перв -  $y$ 
вывод перв
кон
  
```

Задача 7

Составить программу, которая для n строк, описанных применительно к задаче 3, выводит на экран сумму чисел в строке, в которой находится заданное число.

Комментарии

Для нахождения искомой суммы надо знать значение $перв$ и количество чисел в строке, фигурирующей в условии. В свою очередь, указанное количество равно номеру строки y .

Значения $перв$ и y были найдены в предыдущей задаче (в альтернативном варианте, указанном в комментариях).

После определения этих значений искомая сумма находится так:

```

сумма := 0
нц для  $i$  от перв до перв +  $y$  - 1
  сумма := сумма +  $i$ 
кц
  
```

или с использованием формулы для расчета суммы y членов арифметической прогрессии, первый член которой равен $перв$, а разность — 1.

Задача 8

Составить программу, которая для n строк, описанных применительно к задаче 3, выводит на экран номер строки, в которой находится заданное число, и порядковый номер этого числа в строке.

Комментарии

Обозначим искомый порядковый номер — x .

После определения значений $перв$ и y этот номер может быть найден с использованием оператора цикла с условием:

```

x := 1
число := перв
нц пока число < k
  x := x + 1
  число := число + 1
кц
вывод y, " ", x
кон

```

Примечание. *число* — числа, проверяемые в найденной строке (начальное значение этой величины равно *перв*).

Задания для самостоятельной работы

1. Составить программу, которая выводит на экран следующее (при заданном максимальном числе):

```

7
6 6
5 5 5
...
1 1 ... 1

```

2. Составить программу, которая выводит на экран следующее (при заданном максимальном числе):

```

8
8 7
8 7 6
8 7 6 5
...
8 7 ... 1

```

3. Составить программу, которая выводит на экран следующее (где n — одно из чисел 1, 3, 6, 10, 15, 21, 28, 36, 45, ...):

```

n
n - 1 n - 2
n - 3 n - 4 n - 5
...
... 3 2 1

```

4. Составить программу, которая для чисел, описанных применительно к заданию 3, выводит на экран номер строки, следующей за строкой с заданным числом.

5. Составить программу, которая для чисел, описанных применительно к заданию 3, выводит на экран номер строки, в которой находится заданное число.

6. Составить программу, которая для чисел, описанных применительно к заданию 3, выводит на экран первое число строки, в которой находится заданное число.

7. Составить программу, которая для чисел, описанных применительно к заданию 3, выводит на экран сумму и произведение чисел в строке, в которой находится заданное число.

8. Составить программу, которая для чисел, описанных применительно к заданию 3, выводит на экран номер строки, в которой находится заданное число, и порядковый номер этого числа в строке.

ЗАДАЧНИК

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

Решение задания 3-го тура конкурса № 111 "Переправы"

Напомним, что предлагалось решить две задачи.

1. Две семьи (в каждой папа, мама и дочь) хотят переправиться через реку. Есть двухместная лодка. Грести могут только мужчины. Дочери могут быть на берегу или в лодке только вместе с кем-нибудь из своих родителей. Как им всем переправиться на другой берег?

2. То же, но, кроме того, никакую из женщин нельзя оставлять на берегу одну.

Решение задачи 1

Обозначим членов семей первыми буквами, в одной семье большими (П, М, Д), в другой — малыми (п, м, д). Вот схемы переправ:

```

а)
Пм →,
П ←,
пд →,
п ←,
пМ →,
п ←,

```

```

ПД →,
П ←,
Пп →.

```

Решение задачи 2

```

Пп →,
П ←,
ПД →,
п ←,
пМ →,
пП ←,
Пм →,
П ←,
пд →,
п ←,
Пп →.

```

Ответы (на одну или обе задачи) представили:

— Абдувахидова Алина, Абдувахидова Софья, Викторов Даниил, Галкина Эвелина, Журинова Полина, Киселев Владислав, Строкин Константин и Хозин Марат, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

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

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

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

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

— Водальчук Михаил, Курдамосова Софья и Попова Елизавета, гимназия г. Шелехова, Иркутская обл., г. Шелехов, учитель **Водальчук С.А.**;

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

— Деревянченко Дарья, г. Ярославль, школа № 33, учитель **Ярцева О.В.**;

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

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

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

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

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

— Красенков Александр и Пискунова Полина, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Краснёнкова Л.А.**;

— Ледин Роман и Силаев Дмитрий, г. Ростов-на-Дону, лицей № 56, учитель **Ли В.М.**;

— Лежнева Александра, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Перешнев Алексей, г. Архангельск, школа № 9, учитель **Дудина Т.В.**;

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

— Стабрёдов Степан и Стальнова Анастасия, Республика Карелия, г. Сегежа, школа № 6, учитель **Гагарина Э.В.**;

— Цуцков Илья, Ардатовский коммерческо-технический техникум, поселок Ардатов Нижегородской обл., преподаватель **Зудин В.П.**;

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

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

Заметим, что ряд читателей представили решение только задачи 3.2, которое, естественно, является и решением задачи 3.1, однако при этом не являясь оптимальным (в приведенном выше решении задачи 3.1 — 9 “шагов”). В некоторых ответах задачи решались за большее число “шагов”, чем в приведенных выше вариантах.

Напомним, что конкурс проводится в несколько туров, а его итоги будут подводиться с учетом всех туров в целом.

Задача “Обсуждение ответа”

Напомним, что нужно было назвать число, которое обсуждали четверо ребят, если из их утверждений:

1) Коля: “Это число 9”;

2) Роман: “Это простое число”;

3) Катя: “Это четное число”;

4) Наташа: “Это число 15”;

— известно, что и девочки, и мальчики ошиблись ровно по одному разу.

Решение

Нужно рассмотреть один (или несколько) возможных вариантов.

Предположим, что Коля прав, и ответом является число 9. Но тогда ошиблись обе девочки, что противоречит условию. Значит, правду сказал Роман. В этом случае ошиблась Наташа (число 15 простым не является), а Катя сказала правду. Четным (из утверждения Романа) и простым (из утверждения Кати) является только число 2.

Ответ: 2.

Правильный ответ представили:

— Абалакова Светлана и Комарова Елена, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

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

— Антоненко Алевтина и Прохоров Павел, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Гируцкий Павел, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Довгань Алексей, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Деревянченко Дарья, г. Ярославль, школа № 33, учитель **Ярцева О.В.**;

— Иванова Дарья и Сошников Сергей, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

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

— Лежнева Александра, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Мельниченко Алексей, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**

Головоломка “Слова в квадратах”

Напомним, что требовалось вписать в квадратики буквы приведенных терминов вокруг номеров так, чтобы их можно было прочитать (читать термины можно как по часовой стрелке, так и против).

Решение

Слова необходимо расположить в следующем порядке: 1. Ввод. 2. Анод. 3. Шина. 4. Шифр. 5. Фрак. 6. Абак. 7. Банк. 8. Клон. 9. Тело. 10. Стек. 11. Диск. 12. Дизель. 13. Зеро. 14. Порт.

Правильные ответы прислали:

— Абалакова Светлана и Сичак Ольга, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Барановская Татьяна и Жукова Ирина, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Викторов Даниил, Киселев Владислав, Милушкин Дмитрий и Хозин Марат, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Градов Александр, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Закиров Данил, Челябинская обл., г. Златоуст, школа № 9, учитель **Мусатова И.Б.**;

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

— Матвеева Виктория, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Краснёнкова Л.А.**;

— Новикова Анна и Потапова Алевтина, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Токарев Александр и Ухин Алексей, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Худокормова Мария и Чекалин Егор, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина, учитель **Чапкевич И.М.**;

— Чашникова Наталья и Костина Евгения, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

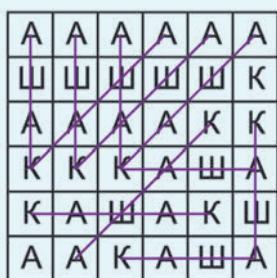
— Чернова Ксения, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**;

— Шукин Алексей, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**

Головоломка “Сколько раз встречается слово?”

Напомним, что следовало определить, сколько раз встречается некоторое слово-существительное в таблице с буквами, если слова можно читать в любом направлении (включая обратные и диагональные), но обязательно по прямой и без разрывов.

Ответ: слово КАША встречается 12 раз (см. ниже).

*Ответы представили:*

— Абалакова Светлана и Комарова Елена, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Акимов Дмитрий, Аتماжар Валерия, Бандыш Михаил, Медведева Анастасия и Попова Елизавета, Владимирская обл., г. Струнино, школа № 11, учитель **Волкова Т.П.**;

— Барановская Татьяна и Жукова Ирина, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Деревянченко Дарья, г. Ярославль, школа № 33, учитель **Ярцева О.В.**;

— Долгополова Алла, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

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

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

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

— Киселев Владислав, Милушкин Дмитрий, Хозин Марат и Чуб Алексей, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Комарова Ольга и Омельченко Елена, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Красненков Артем, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Прохоренко Н.Ю.**;

— Новикова Анна и Потапова Алевтина, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

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

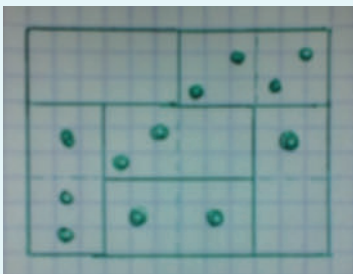
— Шумилова Анастасия и Костина Евгения, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**

Заметим, что в ряде ответов указано меньшее количество слов.

Головоломка “Шесть костяшек домино”

Напомним, что требовалось из шести заданных костяшек домино сложить прямоугольник 3×4 так, чтобы во всех трех строчках точек на костяшках было поровну и во всех четырех столбцах точек тоже было поровну.

Решение показано на рисунке, которое подготовила Мария Худокормова, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина (учитель **Чапкевич И.М.**).



Кроме Марии, правильный ответ прислали:

— Абалакова Светлана и Комарова Елена, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Абдувахидова Алина, Абдувахидова Софья, Викторов Даниил, Лебедева Анастасия, Милушкин Дмитрий и Хозин Марат, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Барановская Татьяна и Жукова Ирина, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Деревянченко Дарья, г. Ярославль, школа № 33, учитель **Ярцева О.В.**;

— Долгополова Алла и Мешалкин Даниил, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

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

— Закиров Данил, Челябинская обл., г. Златоуст, школа № 9, учитель **Мусатова И.Б.**;

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

— Комарова Ольга и Омельченко Елена, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Чашникова Наталья и Костина Евгения, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**

Японские головоломки “судоку”, опубликованные в ноябрьском выпуске “В мир информатики”, правильно решили:

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

— Григоренко Валентин, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Градов Александр, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Деревянченко Дарья, Живой Егор и Комарова Мария, г. Ярославль, школа № 33, учитель **Ярцева О.В.**;

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

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

— Коротков Станислав, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Линьков Александр, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

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

— Хорошилов Иван, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**

Продолжение — в следующем выпуске

ЭТО ПОЛЕЗНО ЗНАТЬ

Хорошая память — залог будущих успехов

Рождаясь на свет без какого-либо жизненного опыта, человек уже имеет генетическую память, с помощью которой развивается первые годы своей жизни. Он имеет рефлекс, жизненные инстинкты, врожденные механизмы поведения. Все это не что иное, как запечатленный и передаваемый из поколения в поколение опыт. Но чтобы жить в современном обществе и вести цивилизованный и культурный образ жизни, необходимы индивидуальные знания и умения, постоянное обновление индивидуального опыта жизни. Для этого человеку нужна хорошая память. Без этого невозможны ни успешное обучение, ни плодотворная деятельность.

Широкое использование информационных технологий изменяет мир быстрее, чем мы способны это заметить. Современный человек активно взаимодействует с техническими средствами информации и все реже ставит перед собой задачу запомнить необходимый материал. К сожалению, это приводит к ослаблению функции

мозга запоминать необходимое количество информации. Люди все чаще записывают, не пытаясь просто запомнить. Но если сейчас не позаботиться о своей памяти, она обязательно подведет в самый неподходящий момент. Увеличение необходимого объема знаний и запоминаемой информации требует рациональных приемов запоминания. Главное в современной жизни — не просто запомнить нужный объем информации, а запомнить его быстро и эффективно. Быстрое и эффективное запоминание необходимо не только школьникам и студентам, оно необходимо и во многих областях человеческой деятельности.

Не все люди одинаково запоминают материал и могут точно и в нужное время его воспроизвести. Запоминание зависит от интересов личностных особенностей, возраста. Одни хорошо запоминают лица, но плохо помнят математический материал, у других хорошая музыкальная память, но возникают трудности в запоминании текстов. Запоминать могут все, но, чтобы запомнить хорошо, нужно знать о некоторых законах памяти и иметь желание освоить приемы эффективного запоминания. Эти приемы помогут запомнить любую необходимую информацию и точно ее воспроизвести.

Многие люди не способны удерживать в сознании даже небольшой по объему материал. Обычный человек может легко запомнить и точно воспроизвести с первого раза не более семи цифр, букв, названий предметов и т.д. Даже семизначный номер телефона с первого раза может запомнить не каждый. Оказывается, чтобы человеку лучше сохранить в сознании какой-либо материал, его необходимо все время видоизменять. Неизменную информацию человек не может долго воспринимать и осознавать, так как она быстро становится ожидаемой. Человек автоматически запоминает любую информацию, но эта информация может также автоматически забываться, если с ней не работать.

Для облегчения процесса запоминания предназначены различные так называемые «мнемотехнические методы», то есть приемы, которые способствуют лучшему запоминанию и позволяют увеличить объем памяти. Они направлены на то, чтобы побуждать человека к искусственному образованию ассоциаций при запоминании нужного материала, но так, чтобы не возникало ошибок при его воспроизведении*. Применяя методы мнемотехники, можно научиться точно запоминать исторические даты, таблицы, большие тексты, телефонные номера и даже случайные цифры. Рассмотрим некоторые из этих приемов.

Создание образов, или Метод свободных ассоциаций

При запоминании списка слов создается воображаемая ситуация, включающая эти слова. Например, надо запомнить следующий ряд слов: котенок, самocat, газета, варенье, дождь и т.д. Первая пара слов «котенок, самocat». Можно представить себе шустро котенка, едущего на самocate. Следующее слово «газета». Теперь наш воображаемый котенок едет на самocate и размахивает газетой. «Варенье» — наш котенок узнал из газеты, где продается самое лучшее варенье, и спешит его купить. «Дождь» — вдруг внезапно начинается дождь, и приходится отложить поездку. И так далее. Такой способ значительно увеличивает количество запоминаемых слов. Но в конечном итоге человек все равно запомнит не только воображаемую картину, но и все предъявленные ему слова. Интересно заметить, что ни у одного человека рассказ не повторяется.

Метод Цицерона

Римский оратор Цицерон использовал следующий прием при подготовке своих докладов. Перед каждым выступлением он ходил по дому и мысленно оставлял части своих лекций в разных местах комнаты. Чтобы вспомнить нужную часть выступления, ему достаточно было вспомнить комнату и размещенные в ней предметы. Цицерон прославился тем, что во время своих выступлений в сенате никогда не пользовался записями и легко воспроизводил по памяти имена, исторические даты, цитаты.

* Вспомните — «Каждый Охотник Желает Знать Где Сидит Фазан».

Допустим, что вы находитесь в своей комнате и должны выучить заданное на дом стихотворение. Попробуйте, прохаживаясь по комнате в определенной последовательности, проговаривать строки у предметов, находящихся в этой комнате. Повторяя все целиком, пройдите снова по комнате в том же направлении, связывая определенные места с частями стихотворения. Для того чтобы в нужное время восстановить информацию, достаточно мысленно пройти по комнате и вспомнить предметы, за которыми были закреплены части стихотворения.

Этот метод встречается под разными названиями: метод мест, система римской комнаты, метод дорог.

Перекодирование

Этот прием позволяет запомнить большое количество цифр. При запоминании ряда чисел их можно перекодировать в известные даты, номера телефонов или квартир. Например, надо запомнить ряд цифр: 5431072589. Перекодируем этот ряд примерно так: 54 — последние цифры моего телефона; 31 декабря — отмечаем Новый год; 07 января — православное Рождество; 25 — это 5×5 ; 89 — номер квартиры бабушки.

Такой же прием можно использовать и при запоминании формул, заменив буквы словами. Например:

$$P = m(g - a)$$

— «Марокко (жара – апельсины)».

Активное повторение

Активное повторение — это перевод материала, который нужно запомнить, в собственную речь. Многократное прочтение материала нельзя назвать повторением. В мнемотехнике повторение — это воспроизведение (пересказ) уже закрепленного материала. Это и дает возможность сохранить в памяти информацию на долгое время. Конечно же нужную информацию надо сначала запомнить. Легкий и небольшой по объему материал лучше запоминается после первых повторений. Более трудный требует многократных повторений. Но не всегда хватает сил повторить большой материал от начала до конца. Школьники быстро устают и начинают повторение опять с самого начала. Опыт педагогов показал, что эффективнее все же запоминается материал, который заучивается частями и повторяется через определенное время. При периодических повторениях увеличивается объем запоминаемой информации. Распределенное повторение также обеспечивает прочность знания и меньшую затрату сил, так как происходит узнавание уже знакомого материала.

Важным условием эффективного запоминания является правильное распределение времени между поступлением и воспроизведением информации. Повторы без перерыва перегружают кратковременную память, и информация с трудом переходит в долговременную память. А сохранение информации на долгое время и есть наша задача. Чем больше времени прошло с на-

чала заучивания материала, тем ниже будет результат. Первое повторение необходимо делать сразу после запоминания, второе — примерно через 40–60 минут, третье — через 3–4 часа, четвертое — на следующий день. В перерыве между утомительными повторениями память должна сама поработать над материалом, а значит, во время перерывов нужно выполнять работу, не связанную с запоминанием. Не стоит готовить одновременно задания по разным предметам, так как смешение информации снижает эффективность запоминания. Этот прием эффективен при подготовке к экзаменам.

Не все приемы и техники быстрого запоминания работают одинаково для каждого человека. Одному человеку необходим один прием, а другому этот же подход может показаться совершенно бесполезным. Разные люди реагируют на образы по-своему. Каждый, кто решится освоить какие-то техники, должен выбрать и подстроить под себя, возможно, изменив уже существующие. Выбранные приемы должны привлекать к себе внимание, содержать простую ассоциацию, легко запоминаться и, самое главное, должны быть удобными и наиболее результативными.

Что нам помогает извлекать информацию из памяти?

Во-первых, осмысленность. Человек гораздо быстрее запоминает то, чему придает смысл. Ряд чисел 1, 2, 3, 4, 5 и т.д. до ста или даже до миллиона легко запомнить и воспроизвести, чем случайный ряд чисел. Или легко запомнить какое-то слово, чем ряд букв, из которых состоит это слово. Например, чтобы запомнить трудный для воспроизведения текст, можно сделать рисунок, который придаст этому тексту смысл, тогда такой материал будет запоминаться уже намного легче. Если материал большой по объему, то сначала его нужно прочитать целиком и осмыслить, выделить главную мысль, а затем разбить на смысловые части. Без напряжения умственных сил и максимальной концентрации внимания здесь не обойтись.

Намного быстрее вспоминается информация, оставляющая яркий след в памяти. Например, слово в ряду цифр запоминается существенно легче, чем ожидаемые знаки. И чем неожиданней материал, тем лучше он запоминается и производится. Поэтому в мнемотехнике, направленной на запоминание слов, нужно придумывать самую невероятную и странную комбинацию предметов. Информацию для запоминания нужно обработать таким образом, чтобы она стала именно яркой и необычной. На этом свойстве основаны многие техники эффективного запоминания.

Связь по смыслу

Связь по смыслу также помогает воспроизведению информации. Близкие по смыслу пары (например, огонь — пламя, стол — стул и т.д.) легче воспроизводятся вместе, если даже в списке они разделены многими другими словами. Иногда

слова группируются в нашей памяти ритмически с подбором рифмы к ним, что и помогает воспроизведению. Но нужно обратить внимание на то, что часто запоминание с использованием ритма и рифмы обеспечивает больше запоминание, чем понимание.

Запоминание и последующее воспроизведение также связано с действием. Лучше запоминается то, что люди сами делают и придумывают. Собственные идеи без усилий запоминаются и хранятся в памяти. Если ученик в процессе обучения достиг чего-то сам (доказал теорему, получил неожиданный экспериментальный результат и т.д.), это запоминается им без особых стараний и на всю жизнь. Чтобы быстрее запомнить, нужно поработать с информацией: что-то подсчитать, сопоставить, поменять местами (в зависимости от характера информации). Таким образом, информация включается в деятельность и значительно улучшает запоминание.

Улучшить процесс запоминания помогает мотивация. Чем выше сила мотивации, тем выше результат. Чтобы лучше запомнить, нужно не только осмыслить материал, но и обдумать, какую положительную роль он может сыграть в жизни.

Можно представить, что именно эти знания помогут вам добиться успеха в глазах окружающих. Тогда эта информация уже будет связана со своей личностью, иметь отношение к успеху и быстрее будет усвоена. Чем больше значит для вас эта информация, тем быстрее вы ее запомните. Нужно постараться установить взаимосвязь материала с будущей деятельностью. Ведь многие старшеклассники сдают экзамены хуже всего по тем предметам, которые, как они считают, не пригодятся им в жизни.

Развитие памяти — это постоянный и целенаправленный процесс. Человек может значительно расширить объем своей памяти. О состоянии памяти нужно заботиться с молодости. Именно в подростковом возрасте происходит значительная перестройка памяти. С 13 до 16 лет наблюдается более быстрый рост запоминания. В этом возрасте происходит переход от механического запоминания к смысловому, а также становится более доступным запоминание абстрактного материала. Возможности памяти в этот период практически безграничны. Поэтому важно именно в этот период обращать внимание на состояние памяти. Если не следить за своей физической формой, то мышцы начинают слабеть и становятся дряблыми. Также, если не тренировать свою память, то рано или поздно умственные способности человека ухудшатся. Необходимо тренировать и укреплять память с помощью специальных упражнений, увеличивать ее объем. Развитие памяти улучшает мыслительные процессы, внимание, интеллект. Чем больше труда будет вложено сейчас, тем легче будет в будущем. Именно сейчас вы закладываете успех будущих побед.

*По материалам статьи
в журнале “Потенциал” (автор — О.Н. Новак).*



ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ г. МОСКВЫ
ИЗДАТЕЛЬСКИЙ ДОМ «ПЕРВОЕ СЕНТЯБРЯ»
МОСКОВСКИЙ ПЕДАГОГИЧЕСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СТРАТЕГИЧЕСКИЙ ПАРТНЕР: ИЗДАТЕЛЬСТВО «ПРОСВЕЩЕНИЕ»



2015

23 МАРТА – 17 АПРЕЛЯ

РАСПИСАНИЕ ДНЕЙ ПЕДАГОГИЧЕСКОГО МАРАФОНА

23 марта	День учителя технологии *	3 апреля	День учителя информатики
24 марта	Открытие Марафона День классного руководителя	4 апреля	День учителя физики
25 марта	День школьного психолога День учителя ОБЖ	5 апреля	День учителя математики
26 марта	День здоровья детей, коррекционной педагогики, логопеда, инклюзивного образования и лечебной физической культуры	7 апреля	День учителя истории и обществознания
27 марта	День учителя начальной школы (день первый)	8 апреля	День учителя МХК, музыки и ИЗО
28 марта	День учителя начальной школы (день второй)	9 апреля	День школьного и детского библиотекаря
29 марта	День дошкольного образования	10 апреля	День учителя литературы
31 марта	День учителя географии	11 апреля	День учителя русского языка
1 апреля	День учителя химии	12 апреля	День учителя английского языка
2 апреля	День учителя биологии	14 апреля	День учителя французского языка
		15 апреля	День школьной администрации
		16 апреля	День учителя физической культуры
		17 апреля	День учителя немецкого языка Заккрытие

marathon.1september.ru

-  Обязательная предварительная регистрация на все дни Марафона с 20 февраля 2015 года на сайте marathon.1september.ru
-  Каждый участник Марафона, посетивший три мероприятия одного дня, получает официальный именной сертификат (6 часов)
В дни Марафона ведущие издательства страны представляют книги для учителей
Начало работы каждого дня – 9.00. Завершение работы – 15.00

УЧАСТИЕ БЕСПЛАТНОЕ. ВХОД ПО БИЛЕТАМ

РЕГИСТРИРУЙТЕСЬ, РАСПЕЧАТЫВАЙТЕ СВОЙ БИЛЕТ И ПРИХОДИТЕ!

Место проведения Марафона: МПГУ, ул. Малая Пироговская, дом 1, стр. 1 (в 5 минутах ходьбы от ст. метро «Фрунзенская»)

* Место проведения Дня учителя технологии: ЦО № 293, ул. Касаткина, 1а (ст. метро «ВДНХ»)

По всем вопросам обращайтесь, пожалуйста, по телефону **8-499-249-3138** или по электронной почте marathon@1september.ru