Адрес редакции:
119261, г. Москва,
Ленинский пр-т, д. 82/2, комн. 6
Почтовый адрес:
119270, г. Москва, а/я 15
Телефон/факс: (495) 140-19-86
E-mail: readinfo@infojournal.ru
URL: http://www.infojournal.ru
Подписные индексы в каталоге «Роспечать»:
для индивидуальных подписчиков — 81407
для предприятий и организаций — 81408

Подписано в печать 15.12.2017.
Формат 60×90 1/8. Усл. печ. л. 8,0.
Тираж 2000 экз. Заказ № 286.

Отпечатано в типографии
ООО «Принт сервис групп»
105187, г. Москва, Борисовская ул.,
д. 14, стр. 6, тел./факс: (499) 785-05-18,
e-mail: 3565264@mail.ru

Свидетельство о регистрации
средства массовой информации
ПИ № 77-12068 от 11 марта 2002 г.

Журнал входит в Перечень российских рецензируемых научных изданий ВАК, в которых должны быть опубликованы основные научные результаты диссертаций на соискание ученых степеней доктора и кандидата наук

ИНФОРМАТИКА В ШКОЛЕ

№ 10 (133) декабрь 2017

Содержание

Москва • «Образование и Информатика»

ОТ РЕДАКЦИИ

Дорогие друзья!

Вот и снова наступил декабрь, и с каждым днем становится все ближе самый долгожданный, самый волшебный праздник — Новый год. Праздник с ароматом мандаринов и запахом хвои, с блеском шаров и сиянием гирлянд... С волнующим предвкушением исполнения желаний, с ожиданием чуда, тайны, сказки...

В Новый год всегда возникает ощущение, что обязательно должно произойти нечто радостное и необыкновенное. Именно перед Новым годом мы загадываем желание не просто потому, что так надо, а с верой, искренней верой в то, что в наступающем году оно точно исполнится, что именно с этим боем курантов начнется все самое настоящее и чудесное.

Дорогие наши читатели и авторы! Счастья и добра вам в новом году! Пусть наступающий год принесет вам много удачи и радости, приятные встречи и неожиданные открытия. И, конечно, исполнение самого заветного желания!

С Новым годом!

Искренне ваша,
редакция журнала
«Информатика в школе»

Сияя тонким кружевным нарядом,
Сквозь призрачные голубые сны
Зима обводит землю томным взглядом
И опускает полог тишины.

Вчерашний дождик в шубе из снежинок
На цыпочках спускается с небес.
Хрустальным звоном серебристых льдинок
Баюкает себя притихший лес.

Но день настал, и от морозной дремы
Мир пробудился радостно. И вот —
Такой шальной и с детских лет знакомый
Во все дома приходит Новый год.

Обвитый мишурой и серпантином,
Сверкающий гирляндами огней,
Со свежим ароматом мандаринов
Выходит он из расписных саней.

С улыбками и в праздничных одеждах
Встречаем гостя! Щедрый Новый год
Подарит всем веселье и надежду
На лучшее, что нас в грядущем ждет.

Марина Фомина

Е. А. Еремин,
Пермский государственный гуманитарно-педагогический университет

ЗНАКОМСТВО С НАСТОЯЩИМ ООП В DELPHI*

Аннотация

Бытует мнение, что ученик, который в Delphi создал простое визуальное приложение, познакомился с основами ООП. Это, разумеется, не так. В статье предложена методика, как можно по-настоящему продемонстрировать идеи ООП, используя возможности среды Delphi. Для этого тщательно подобрана и детально разработана простая и наглядная задача — индикатор значения целого числа, подобный табло современного электронного прибора.

Ключевые слова: ООП, изучение, преподавание, Delphi, курс информатики.

Контактная информация

Еремин Евгений Александрович, канд. физ.-мат. наук, доцент, доцент кафедры мультимедийной дидактики и ИТО, Пермский государственный гуманитарно-педагогический университет; адрес: 614990, г. Пермь, ул. Сибирская, д. 24; телефон: (342) 238-63-31; e-mail: eremin@pspu.ru

E. A. Eremin,
Perm State Humanitarian Pedagogical University

KNOWING REAL OOP WITH DELPHI

Abstract

An opinion prevails that pupil, who developed in Delphi a simple visual application, learned OOP basics. It is wrong of course. The article proposes a method how OOP ideas can be really demonstrated using possibilities of Delphi environment. For this purpose a simple and descriptive task was carefully selected and particularly developed — it is an indicator of integer number's value similar to the display of modern electronic device.

Keywords: OOP, learning, teaching, Delphi, informatics course.

Широкое распространение в преподавании школьного курса информатики заслуженно получила среда визуального программирования Delphi. Эта система с понятным графическим интерфейсом очень удобна и позволяет дать ученикам начальные представления о современных технологиях программирования.

Система Delphi является *объектно-ориентированной*. На основании этого, без сомнения, правильного утверждения часто делается неверный вывод о том, что любой ученик, который поработал в Delphi и создал несколько простых приложений, уже получил представление об объектно-ориентированном программировании (ООП). Даже в некоторых учебниках информатики (не будем углубляться в детали истории) можно увидеть подобного рода логику. К сожалению, все не так просто.

Delphi использует не одну, а несколько разных технологий. Этот программный продукт как минимум является одновременно [1]:

- системой визуального программирования;
- системой ООП;
- системой программирования на основе событий;
- системой с поддержкой различных технологий доступа к данным (например, ADO);
- системой с поддержкой всевозможных сетевых технологий;
- (как следствие всех своих достоинств) системой быстрой разработки RAD (Rapid Application Development).

Традиционное заблуждение, о котором говорилось выше, состоит в неявном отождествлении первых двух пунктов списка. В пользу того, что указанные пункты напрямую не связаны, можно привести простой, но очень убедительный факт. Начиная с пятой версии, очень известная в недалеком прошлом система программирования Turbo Pascal уже позволяла писать объектно-ориентированные программы. Тем не менее эта среда не имеет *никакого* отношения к визуальному программированию.

Данная статья направлена на то, чтобы продемонстрировать основы настоящего ООП (а не визуального программирования) в Delphi, причем сделать это максимально просто. По мнению автора, излагаемый материал не сложнее тех конструкций программирования, которые принято изучать в традиционном школьном курсе информатики. Попутно дается некоторое обоснование, почему стоит познакомить школьников с наиболее важными идеями ООП.

Приведем еще один дополнительный аргумент в пользу актуальности разработанных материалов. При изучении ООП существует проблема подбора подходящих для новичков примеров [2]. Известно, что преимущества ООП наиболее отчетливо проявляются на сложных проектах, а учебные задачи, напротив, желательно видеть максимально прозрачными и простыми. К тому же опыт показывает, что на коротких программах ООП реализация часто выглядит даже сложнее, чем традиционное решение.

* Материалы к статье можно скачать на сайте ИНФО: http://infojournal.ru/journals/school/school_10-2017/

Описываемые в литературе примеры чаще всего берутся из бизнес-практики, например, в [7] обсуждение ведется на базе банковских счетов. Такой подход для общеобразовательной школы едва ли стоит признать интуитивно понятным. В популярных самоучителях можно найти примеры попроще: в [5], в частности, придуманы задачи про оптимиста и пессимиста, счастливый, несчастный и капризный объекты, а также карточная игра Blackjack.

Много интересных вопросов и заданий можно увидеть в [6]. Здесь и весьма удачная с образовательной точки зрения задача о классификации многоугольников (в главе об отношении объектов), и описание постройки дома (состав объектов), и расчет прохождения автомобилем трассы с учетом расхода бензина (состояние объекта). Для крупного проекта предлагается реализовать широко распространенную в школе игру «Морской бой». Тем не менее все примеры изначально рассчитаны на «бескомпьютерное» обсуждение.

Автор давно занимается преподаванием основ ООП в педагогическом вузе и на основании своего опыта берется утверждать, что наглядная и интуитивно понятная задача, подходящая для демонстрации технологий ООП, может служить методической находкой, достойной внимания учителей информатики. Специально для статьи была разработана такая задача: отображение числа на индикаторе, напоминающем табло современного электронного прибора. Объекты в обсуждаемой задаче просты и естественны — это отдельные цифры и сегменты, из которых они составлены. В конце статьи дополнительно продемонстрировано, что готовые объекты из написанной программы могут быть полезны при решении многих практически интересных задач.

1. Почему стоит познакомиться с ООП?

1.1. Объект — результат развития типов компьютерных данных

Если проанализировать усложнение типов данных в языках высокого уровня, то можно отчетливо видеть, что оно объективным образом приводит к необходимости дополнения переменных (областей памяти, хранящих данные) обслуживающими их программными кодами. А объединение данных и обрабатывающих их программ есть одна из базовых основ ООП. В подтверждение приведем несколько аргументов.

- Значение простой переменной, например, типа integer процессор способен прочитать (а часто и сразу обработать) одной командой; для этого достаточно адреса переменной и количества байтов. Для любого сложного типа, в котором хранится несколько значений (массив, структура и т. п.), прежде чем обратиться к нужному, необходимо вычислить его адрес, а для операций над данными вроде строк дополнительно требуется некоторая программа.
- Если хотя бы один из индексов массива не является константой, вычислить адрес элемента заранее (до запуска программы) невозможно. Следовательно, появляется потенциальная возможность выйти за пределы массива в случае неправильно заданного значения индексной переменной, а значит, возникает проблема проверки правильности обращения к данным.
- При создании динамической переменной, т. е. переменной, которая размещается в памяти в момент выполнения программы, нужно проделать целый ряд действий по выделению свободной области памяти. Появляется дополнительная проверка при обращении к такой переменной: была ли она ранее создана.
- В классическом Паскале текст первоначально предлагалось хранить в массиве (array of char). Однако позднее для облегчения работы со строками в Turbo Pascal был предложен новый тип данных string, который, унаследовав все возможности строкового массива, имел разнообразный ассортимент встроенных строковых процедур и функций (insert, delete, copy и др.). Отчетливо видно, что усложнение типа данных потребовало ***включения в него некоторых стандартных действий*** — по сути программ работы со значениями этого типа.

Так обстоит дело со стандартными, встроенными в язык программирования, типами данных. К сожалению, их недостаточно для описания имеющихся на практике величин. Возьмем для примера не самую сложную конструкцию — дату, состоящую из числа, месяца и года. Очевидно, что, если мы захотим проверить правильность введенной даты, нам придется написать достаточно большую программу, учитывающую, в частности, число дней в месяце и високосность года. При определении возраста нужно определить разницу значений двух переменных данного типа, что тоже потребует не самой короткой программы.

Итак, мы видим, что чем сложнее тип данных (и чем ближе он к реальности!), тем больше разнообразных «обслуживающих» его значения действий требуется. Это подсказывает нам, что надо объединять переменные с типовыми методами (программами) их обслуживания.

1.2. ООП — способ создать программную модель реальных объектов

Благодаря объединению в единую конструкцию всех переменных и программ их обработки мы, по сути дела, получаем некоторый программный эквивалент объекта окружающего нас мира. Иными словами, объектно-ориентированная программа становится своеобразной моделью реального мира.

Считается, что объект в окружающем мире обладает определенными свойствами и поведением. Эквивалентом свойств в объекте ООП являются переменные (*поля*), а поведение определяется процедурами или функциями (*методами*). С помощью методов также происходит взаимодействие объектов.

Многим программистам нравится такое отождествление сравнительно автономных программных частей с объектами реального мира. Они утверждают, что при разработке сложных программных систем мыслить категориями объектов значительно удобнее, чем категориями переменных или структур данных. Справедливости ради отметим, что у ООП подхода имеются и противники.

1.3. Не только программирование

Объекты широко используются не только при написании текстов программ, но также при разработке внутренней организации систем и их интерфейса. Например, объектно-ориентированный подход позволяет описать такие важные понятия, как организация про-

цессов для каждой из задач в современной операционной системе, механизм обмена сообщениями между компонентами сложных систем и даже работу пользователя со стандартными органами управления графического интерфейса [8]. Идеи, похожие на наследование в ООП, обнаруживаются в области искусственного интеллекта, где для представления знаний широко применяются иерархические структуры. Архитектура современных компьютеров строится из отдельных модулей, разнообразных по внутренней организации, но соединенных общими интерфейсами. В образовании мы тоже можем увидеть примеры, когда учебное программное обеспечение конструируется на основе модульных объектов [9].

Такое разнообразное применение объектно-ориентированного подхода в промышленности, теоретической информатике и техническом образовании требует широкого знакомства (хотя бы предварительного) с его идеями.

2. Минимальные сведения об объектах

Прежде чем написать ООП программу, полезно познакомиться с наиболее важными, совершенно необходимыми для этого сведениями.

Объект — это особый тип данных, объединяющий в себе переменные и программы их обработки. Переменные принято называть *полями*, а процедуры и функции, работающие с полями, — *методами*.

Согласно идеологии ООП, программист не должен напрямую обращаться к полям, ***читать и писать значения полей можно только через методы***. Минимальная аргументация для такого подхода может быть следующей: прежде чем положить значение в поле, его необходимо проверить. Кроме того, хранение значений полей внутри объекта может не соответствовать форме представления входных или выходных данных (например, дата и время часто хранятся в виде определенным образом сформированного числа) — в таких случаях потребуется предварительное преобразование данных. Заметим, что в данной статье языковые средства поддержки этой философии (уровни доступа к переменным, конструкция property) для простоты не рассматриваются — это уже следующий шаг знакомства с ООП.

Чтобы не описывать каждый объект в отдельности, одинаковые объекты объединяются в группу, называемую *классом*. Именно класс описывается подробно, а затем от него легко и быстро создается любое число объектов (*экземпляров класса*). Как это можно себе представить? Например, как производство деталей по чертежу или как штамповку изделий с помощью некоторой формы. Полезно провести и более близкую к программированию аналогию. Вспомним, что в языке имеются типы переменных, например integer. В типе предопределено, что хранится в качестве значения переменной и что с ним можно делать (например, два числа можно сложить). Программист создает сколько угодно переменных данного типа, и каждая из них будет без всяких дополнительных объяснений обладать свойствами, заложенными в типе (в частности, любые числовые переменные можно складывать). Так что класс есть тип переменной, а объект — собственно переменная. Суть понятий «класс» и «экземпляр» подробно и доступно описана в школьном учебнике [3] на примере книг личной библиотеки.

Процесс создания объектов заданного класса имеет некоторые особенности [4]. У каждого класса существует специальный метод, который называется *конструктор*. Он создает (формирует) объект в памяти компьютера. В некоторых языках, например в Java, имя конструктора совпадает с именем класса. В Delphi конструктор носит стандартное имя *Create*.

Часто программисты пишут свой собственный конструктор, расширяя тем самым стандартный. Это удобно делать хотя бы для того, чтобы положить в поля создаваемого объекта подходящие начальные значения.

Рассмотрим, как создается объект, на некотором примере (рис. 1).

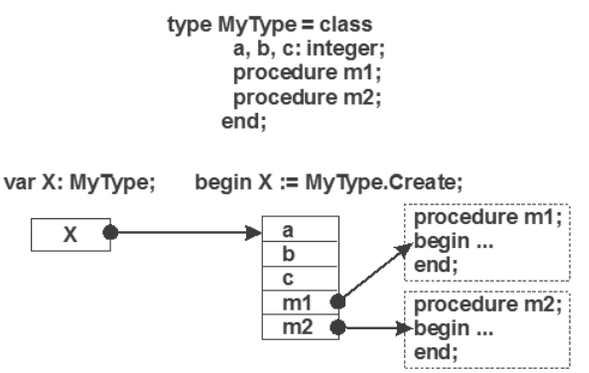


Рис. 1. Объект в памяти компьютера

Пусть класс с именем *MyType* содержит три целочисленных поля *a*, *b* и *c*, а также методы *m*1 и *m*2. Если в тексте программы предварительно сделано общее описание типа *MyType* (см. рис. 1), то можно создать одну или несколько переменных такого типа. В нашем примере есть единственная переменная *X*. Каков ее смысл? Это переменная, в которой позднее будет храниться адрес области памяти с данными об объекте (программисты называют этот адрес *ссылкой* или *указателем*). В первый момент ссылка еще не определена. Чтобы ее заполнить, надо вызвать конструктор класса *MyType.Create*. Выполняя его, компьютер создаст у себя в памяти поля *a*, *b* и *c*. Что касается методов, то они обычно также работают через ссылочный механизм: в *m*1 и *m*2 помещаются ссылки на программы методов. Благодаря использованию ссылок каждый объект может быть «настроен» на свой собственный метод (в частности, методов с одинаковым именем может быть несколько — у каждого класса свой). Теперь, когда объект, наконец, создан, адрес начала области памяти с его данными помещается в переменную *X*, и им можно начинать пользоваться, например, вызвав метод *m*2 обращением *X.m*2 (проследите по рисунку 1, как компьютер «доберется» до нужного метода).

Заметим, что тексты процедур *m*1 и *m*2 не включены в описание объекта и размещаются отдельно. Это не случайно, поскольку тому, кто пользуется объектом, важно знать о существовании метода, а о его внутреннем устройстве — вовсе не обязательно.

При описании нового класса можно использовать уже определенные ранее. Например, в нашем случае можно задать класс *MyType2* так:

```
MyType2 = class(MyType)
```

При этом в новый класс автоматически будут включены все пять компонентов родительского класса *MyType*. Такой «экономный» способ описания классов принято называть *наследованием*. Наследование существенно сокращает описания и поэтому используется практически во всех задачах.

Связанные механизмом наследования классы в объектно-ориентированных программах образуют иерархию, подобную вложенным папкам в современной операционной системе.

Существует и еще одна важная возможность организации иерархии классов. Пусть от класса *MyType* созданы два класса *C*1 и *C*2. В каждом из них, как мы уже знаем, будет метод *m*2. ООП разрешает нам переопределить этот метод для новых классов (здесь оказывается очень кстати ссылочный способ доступа к методам). В таком случае вызовы одного и того же метода смогут выполнять *разные* действия — у каждого класса свои. Такую ситуацию в профессиональной литературе принято называть красивым термином *полиморфизм*.

3. Постановка задачи

Сконструируем программный индикатор для отображения значений целочисленных переменных на базе традиционных для электронных устройств семисегментных индикаторов (вспомните, например, табло электронных часов).

Дополнительно детализируем задачу. Пусть отображаемые индикатором числа могут изменяться в диапазоне от –32768 до 32767, как это бывает у чисел типа integer в классическом Паскале. Очевидно, что индикатор должен содержать знаковый разряд, способный зажечь знак минус, и пять цифр. Договоримся, что все это будет собрано на общей панели (стандартный компонент Delphi под названием *Panel*).

Итак, для нашей задачи имеем следующий, естественным образом возникающий, ассортимент объектов (рис. 2). Основой всех цифр является сегмент, который обладает двумя состояниями: светится или нет. Семь определенным образом расположенных сегментов образуют цифру. Сразу бросается в глаза, что для формирования цифры нужно два разных вида сегментов — горизонтальные и вертикальные, отличающиеся своими размерами. В свою очередь, индикатор числа включает в себя знаковый горизонтальный сегмент и пять цифр.

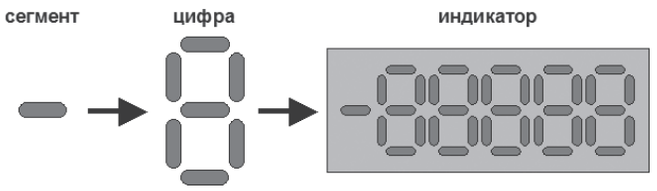


Рис. 2. Построение индикатора из объектов

4. Подготовка к написанию программы

Прежде чем написать первую строку программы, очень полезно предварительно продумать и построить полный набор используемых классов.

Мы уже знаем, что цифры строятся из отдельных сегментов, причем последние могут быть вертикальными и горизонтальными. Очевидно, что эти два типа сегментов различаются по длине и ширине, а также по расположению внутри цифры. Как проще описать это положение? Обратимся к рисунку 3, где подробно нарисована семисегментная цифра. Договоримся обозначать координаты левого верхнего угла цифры как *x*0 и *y*0. Нам нужно написать формулы для координат левого верхнего угла каждого сегмента относительно *x*0 и *y*0.

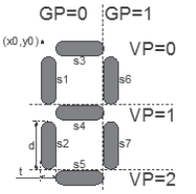


Рис. 3. Формирование цифры из сегментов

Из рисунка видно, что расположение сегментов обладает определенным «периодизмом».

Давайте сначала внимательно посмотрим на вертикальную координату. По ней сегменты легко разделить на три следующих периода:

- первый период (для него номер *VP* удобно принять равным нулю) содержит сегменты *s*1, *s*3 и *s*6;
- второй (*VP* = 1) — *s*2, *s*4 и *s*7;
- и, наконец, третий (*VP* = 2) является неполным: он состоит из единственного сегмента *s*5.

Можно сказать, что сегменты одного полного вертикального периода образуют маленькую букву «п».

Период по горизонтальной координате не так заметен, но он тоже есть:

- первый период (*GP* = 0) включает в себя пять сегментов: *s*1 — *s*5;
- два последних сегмента относятся ко второму (неполному) периоду. Поскольку в нем нет ни одного горизонтального сегмента, на эту разновидность сегментов величина *GP* никак не влияет.

Как вы уже, вероятно, заметили, сегменты горизонтального периода образуют большую букву «Е».

Все эти рассуждения отнюдь не праздны, ибо, зная значения *GP* и *VP* (а также длину *d* и толщину *t* сегмента), можно легко вычислить координаты для любого сегмента по следующим формулам:

горизонтальные сегменты:

```
x = x0+t
y = y0+VP*(d+t)
```

вертикальные сегменты:

```
x = x0+GP*(d+t)
y = y0+VP*(d+t)+t;
```

Заметим, что *d* + *t* — это фактически размер одного периода (как горизонтального, так и вертикального).

Таким образом, для полного описания сегментов достаточно всего трех разновидностей (классов) сегментов: *TSegment*, *THor* и *TVer* (рис. 4).

Базовый класс *TSegment* описывает устройство «абстрактного» сегмента. В нем есть поле *seg*, ссылаю-

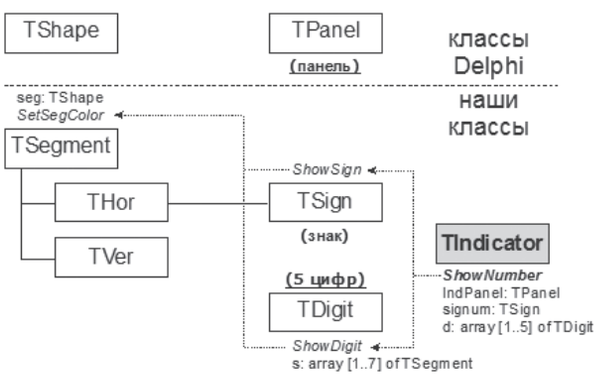


Рис. 4. Все используемые в задаче классы

щееся на стандартный визуальный класс Delphi *TShape*, который изображает прямоугольник со скругленными углами. Описываемый класс «умеет» нарисовать себя по заданным извне размерам и координатам. Для «включения/выключения» сегмента путем изменения цвета предназначен метод *SetSegmentColor* (помните тезис о том, что для обращения к полю нужно использовать метод?).

Порожденные от *TSegment* типы *THor* и *TVer* учитывают особенности горизонтальных и вертикальных сегментов. По сути дела, они задают требуемые размеры сегмента и вычисляют по приведенным выше формулам его положение.

Наконец, если к типу *THor* добавить метод индикации знака *ShowSign*, то получится класс *TSign*.

Следующий тип данных, используемый в нашей задаче, — это *TDigit*, описывающий одну цифру индикатора. Поскольку цифра состоит из семи сегментов, мы видим в ее составе сложное поле *s*, которое является массивом из семи объектов типа *TSegment*. Метод *ShowDigit* для любого заданного числового значения от 0 до 9 зажигает необходимые сегменты.

Обратите внимание на важную деталь. Класс *TSign* является *разновидностью* (*подклассом*) *TSegment*. А вот класс *TDigit состоит* из объектов типа *TSegment* (такое отношение объектов называется *часть—целое*). Разные по своей сути отношения описываются разными языковыми средствами.

Наконец, главный для потребителей класс *TIndicator* собирает вместе панель (стандартный класс Delphi *TPanel*), знак и пять цифр. Его главный метод *ShowNumber* отображает заданное целое число. Если создать переменную обсуждаемого типа с именем *ind*, то вызов

```
ind.ShowNumber(1243)
```

должен высветить на индикаторе стоящее в скобках число. Таким образом, когда все будет готово, достаточно знать о существовании только одного класса — *TIndicator*.

Обратим внимание читателей на пунктирные стрелки на рисунке 4. Они показывают, что метод *ShowNumber* вызывает методы *ShowSign* и пять раз *ShowDigit*, каждый из которых, в свою очередь, зажигает необходимые индикаторы с помощью метода *SetSegmentColor*.

Очень полезно сопоставить рисунок 4 с рассмотренным ранее рисунком 2. Отчетливо видно соответствие классов программы реальным объектам: в обоих случаях четко просматривается цепочка: сегмент — цифра — индикатор.

5. Программа

Тщательно подготовившись к написанию программы, можно теперь перейти к программированию. Полный текст программы приведен в приложении, размещенном на сайте издательства.

Тем читателям, которые захотят повторить пример на своем компьютере, дадим полезный совет из опыта. Не обязательно реализовывать сразу весь проект. Опишите один класс — и сразу его опробуйте, создав на форме одну-две переменных этого класса. Убедитесь, что они после запуска программы появились на экране, и обязательно попробуйте поменять имеющиеся параметры, чтобы увидеть, как они влияют.

Начать надо с того, что запустить Delphi и стандартным образом создать пустое окно (в старых версиях это происходит при старте системы автоматически). Затем, выбрав пункты меню *File*, *New*, *Unit*, создадим новый программный модуль, в котором и будем описывать наш индикатор; назовем файл *indicat*. Такое решение удобно для последующего применения: дополняя любой проект этим файлом, мы тут же получаем возможность создавать и использовать индикаторы.

Найдем в созданном системой модуле служебное слово interface и после него добавим следующий текст.

```
uses Classes, Controls, ExtCtrls, Graphics;
CONST color0 = clSilver; color1 = clBlack;
      panelColor: TColor = $DDDDDD;
      dSeg=10; tSeg=5;
      margin=10; spacing=5;
```

Первая строка подключает необходимые внешние модули из системной библиотеки Delphi. Они позволят нам использовать в своей программе компоненты *Shape* и *Panel*, а также применять для констант стандартные обозначения цветов.

Далее следует небольшой блок «настроечных» констант. Он описывает цвета выключенных (*color*0) и горящих (*color*1) сегментов, а также цвет панели в hex-формате, подобно тому как это делается в HTML. Очевидно, что для панели выбран светло-серый оттенок. Задаются длина и толщина сегментов, а также отступ от краев панели и промежуток между цифрами. Все эти величины обязательно потом попробуйте изменять.

Теперь подошло время объявлять классы. Начнем с базового. Его «декларация» выглядит так:

```
TYPE TSegment = class
  seg: TShape;
  constructor Create(par: TWinControl;
  x,y, w,h: integer);
  procedure SetSegColor(n:integer);
end;
```

Очевидно, что методы *Create* и *SetSegColor* нуждаются в конкретизации. Их текст размещается в модуле после служебного слова implementation и имеет следующий вид:

```
constructor TSegment.Create(par: TWinControl;
  x,y,w,h: integer);
begin seg := TShape.Create(par);
with seg do
  begin Parent := par;
    Shape := stRoundRect; Pen.Color := clSilver;
      Left := x; Top := y;
      Width := w; Height := h;
  end;
end;
procedure TSegment.SetSegColor(n:integer);
begin if n=1
  then seg.Brush.Color:=color1
  else seg.Brush.Color:=color0;
end;
```

Рассмотрим устройство методов подробнее. Начнем с конструктора.

Прежде всего в памяти создается внешний образ сегмента (визуальный объект типа *TShape*), и ссылка на него помещается в переменную *seg*. Аргумент в скобках — это владелец (*owner*) создаваемого компонента, т. е. компонент, который отвечает за создание и последующее удаление «нашего» компонента из памяти. Подчеркнем, что *owner* не отвечает за рисование компонента, поэтому пока сегмента на экране еще нет. Чтобы он возник, надо назначить еще одного «ответственного» под названием

«родитель» (*parent*); этот компонент и будет обеспечивать прорисовку *seg* на экране [4]. В нашем случае *owner* и *parent* совпадают, хотя это и не обязательно.

С точки зрения языка *parent* — это одно из свойств переменной *seg*, поэтому полное обращение к нему должно было бы выглядеть как *seg.parent*. Но поскольку дальше у *seg* устанавливается целый ряд других свойств, применен специальный сокращающий запись оператор *with*. Как видно из текста, данная конструкция позволяет указать *seg* только один раз, а затем имя переменной будет автоматически распространяться на все находящиеся внутри оператора свойства. В частности, свойство *Shape* задает форму прямоугольника со скругленными углами, а цвет пера *Pen.Color* определяет цвет рисования границ компонента. Дальше привычным образом задаются положение и размеры компонента; их числовые значения поступают в конструктор в качестве параметров.

Таким образом, конструктор класса *TSegment* обеспечивает создание в памяти и отображение на экране одного сегмента.

После того как объект *seg* создан, его цвет можно поменять с помощью метода *SetSegColor*. Метод понятен без комментариев. Пожалуй, стоит только напомнить, что *Brush* — это «электронная кисть», «закраска» которой и определяет цвет объекта.

Два следующих класса — *THor* и *TVer*, описывающие горизонтальный и вертикальный сегменты, — очень похожи, поэтому разберем только один. (Если с другим возникнут какие-либо трудности, можно обратиться к полному листингу на сайте.)

```
TVer = class(TSegment)
  constructor Create(par: TWinControl; x0,y0,
  GP,VP: integer);
end;

constructor TVer.Create(par: TWinControl;
x0,y0,GP,VP: integer);
var x, y: integer;
begin x := x0+GP*(dSeg+tSeg);
  y := y0+VP*(dSeg+tSeg)+tSeg;
  inherited Create(par, x,y, tSeg,dSeg);
end;
```

Сразу из объявления рассматриваемого класса видна его особенность: он является потомком класса *TSegment* (указано в скобках после слова *class*). Как уже было сказано, при таком способе создания класс-потомок автоматически наследует все поля и методы класса-предка. В нашем случае это означает, что в классе *TVer*, как и в родительском классе *TSegment*, уже есть и визуальный рисунок сегмента, и способ работы с ним. Имеется также и родительский конструктор, описанный выше, который по заданным координатам и размерам позволяет создать сегмент.

В новом классе объявлен собственный конструктор *Create* с другим набором параметров. Как это следует понимать? Это означает, что в классе *TVer* одновременно существуют *оба* конструктора — и родительский, наследуемый (по-английски *inherited*), и свой собственный. Предназначение новой версии конструктора состоит в пересчете координат цифры $x0$ и $y0$ в координаты конкретного вертикального сегмента x и y. Для этого используются полученные ранее формулы с параметрами *GP* и *VP* (см. обсуждение рисунка 3). Вычисленные координаты x и y передаются в последней строке непосредственно конструктору-предку, который и создает в нужном месте экрана сегмент.

Класс *THor* устроен аналогично, за исключением нескольких небольших отличий. Во-первых, в нем используются другие формулы; поскольку в них нет *GP*, в новом конструкторе будет на один параметр меньше. Во-вторых, размеры горизонтального сегмента отличаются от размеров вертикального, поэтому при обращении к наследуемому конструктору два последних параметра надо поменять местами.

Последний класс, входящий в иерархию сегментов, это *TSign*. Он создается от *THor* и тоже переопределяет его конструктор. Если заглянуть в текст, то он сразу вызывает родительский конструктор, указав только, что для знакового сегмента $VP = 1$ (см. рис. 3).

```
constructor TSign.Create(par: TWinControl;
  x0, y0: integer);
begin inherited Create(par,x0,y0,1);
end;
```

Знаковый сегмент является частью индикатора, он высвечивает минус у отрицательных чисел. Эту функцию выполняет простейшая процедура *ShowSign*:

```
procedure TSign.ShowSign(n: integer);
begin if n<0 then SetSegColor(1)
  else SetSegColor(0); end;
```

Перейдем теперь к отдельно стоящему классу *TDigit*. Его назначение состоит в том, чтобы собрать семь сегментов и организовать их совместную работу — отображение одной отдельно взятой цифры числа.

```
TDigit = class
  s: array [1..7] of TSegment;
  constructor Create(par: TWinControl;
    x0,y0: integer);
  procedure ShowDigit(n: integer);
end;
```

Для индикации цифр необходимо как-нибудь задать соответствие между значениями цифр и высвечивающими их начертание сегментами. Удобнее всего это сделать в виде матрицы, в которой строки соответствуют сегментам $s1$ — $s7$, а столбцы — значениям цифры от 0 до 9.

```
const DMatrix: array [1..7, 0..9] of integer =
      ((1,0,0,0,1,1,1,0,1,1),
       (1,0,1,0,0,0,1,0,1,0),
       (1,0,1,1,0,1,1,1,1,1),
       (0,0,1,1,1,1,1,0,1,1),
       (1,0,1,1,0,1,1,0,1,1),
       (1,1,1,1,1,0,0,1,1,1),
       (1,1,0,1,1,1,1,1,1,1) );
```

Возьмем для примера последнюю строку матрицы, описывающую правый нижний вертикальный сегмент $s7$ (взгляните еще раз на рисунок 3). Он не горит только в одном случае: когда отображается цифра 2. Соответственно, строка состоит из всех 1, кроме третьего (не забываем, что есть цифра 0!) значения. Программа будет работать со столбцами матрицы. Скажем, для индикации цифры 9 будет взят последний столбец, и загорятся все сегменты, кроме $s2$ (проверьте самостоятельно).

Конструктор класса создает в качестве элементов массива семь сегментов — четыре вертикальных и три горизонтальных. Всем им задаются родитель, координаты левого верхнего угла изображения всей цифры и требуемые значения *GP* и *VP* в полном соответствии с рисунком 3. В приложении есть еще один дополнительный фрагмент программы, который зажигает в созданной цифре 0. Для краткости он здесь не приводится.

```
constructor TDigit.Create(par: TWinControl;
  x0, y0: integer);
var i: integer;
begin s[1] := TVer.Create(par,x0,y0,0,0);
      s[2] := TVer.Create(par,x0,y0,0,1);
      s[3] := THor.Create(par,x0,y0,0);
      s[4] := THor.Create(par,x0,y0,1);
      s[5] := THor.Create(par,x0,y0,2);
      s[6] := TVer.Create(par,x0,y0,1,0);
      s[7] := TVer.Create(par,x0,y0,1,1);
end;
```

Наконец, благодаря матрице *DMatrix* отображение любой цифры происходит совсем просто: берем столбец с номером, равным значению цифры, и по очереди в цикле передаем его значения сегментам *s*1 — *s*7.

```
procedure TDigit.ShowDigit(n: integer);
var i: integer;
begin for i:=1 to 7 do
  s[i].SetSegColor(DMatrix[i,n]);
end;
```

Теперь вся подготовительная работа завершена, и нам остается окончательно собрать весь индикатор в единый класс *TIndicator*. Именно с ним и будет непосредственно работать пользователь.

Индикатор состоит из панели, на которой размещены знаковый сегмент и пять цифр. Для каждого компонента предусмотрена специальная переменная, причем последняя — это массив из пяти элементов.

```
TYPE TIndicator = class
  IndPanel: TPanel;
  signum: TSign;
  d: array [1..5] of TDigit;
  constructor Create(par: TWinControl;
    xp,yp: integer);
  procedure ShowNumber(N: integer);
end;
```

Конструктор метода просто создает все перечисленные выше переменные. Обратим внимание на следующую важную деталь. Родитель панели индикатора задается извне — в нашем случае это будет стандартная форма. А вот родителем всех сегментов является сама панель. Такой подход позволяет сделать наш индикатор единым целым и все координаты сегментов считать относительно базовой панели. Кроме того, для перемещения индикатора по форме достаточно просто поменять координаты панели *xp* и *yp*.

```
constructor TIndicator.Create(par: TWinControl;
  xp,yp: integer);
var DigitX, i : integer;
begin {panel}
  IndPanel := TPanel.Create(par);
  with IndPanel do
    begin Parent := par;
      DigitX := 2*tSeg+dSeg;
      Width  := 2*margin+dSeg+5*spacing+5*DigitX;
      Height := 2*margin+2*dSeg+3*tSeg;
      Left := xp; Top := yp;
      Color := panelColor; BevelWidth:=2;
    end;
  {sign}
    signum := TSign.Create(IndPanel, margin-tSeg,
      margin);
    {5 digits}
    for i:=1 to 5
      do d[i] := TDigit.Create(IndPanel,
                        {x0} margin+dSeg+spacing
                            +(i-1)*(DigitX+spacing),
                        {y0} margin);
end;
```

Конструктор рассчитывает положение всех компонентов на панели. При этом используются две «настроечные» константы: *margin* — размеры отступа от краев панели (для простоты одинаковые по обеим координатам) — и *spacing* — расстояние между цифрами. Все формулы достаточно просты по своей сути; автор надеется, что читатели, глядя на рисунок 2, легко разберутся в них.

Остается описать последний метод — отображение на индикаторе произвольного числа.

```
procedure TIndicator.ShowNumber(N: integer);
var i: integer;
begin signum.ShowSign(N); n:=ABS(N);
  for i:=5 downto 1 do
    begin d[i].ShowDigit(n mod 10);
      n:=n div 10;
    end;
end;
```

Сначала отображается знак числа, а само число делается положительным с помощью взятия модуля. Далее используется стандартный алгоритм получения цифр десятичного числа: последняя цифра числа есть остаток от деления его на 10, а следующее значение без последней цифры получается делением нацело на 10. Заметим, что цифры при этом получаются, начиная с младшего разряда числа, поэтому и цикл по цифрам индикатора идет в обратном порядке.

Итак, модуль с описанием индикатора полностью готов! Можно его применить. Именно сейчас станет очевидно, насколько удобно пользоваться готовыми объектами.

Вернемся теперь к модулю формы. В качестве теста создадим на *Form*1 три индикатора. На двух будут отображаться произвольные исходные числа *N*1 и *N*2, а на третьем — их сумма.

Чтобы модуль формы «увидел» модуль с индикатором, необходимо после слова implementation и подключения файла с визуальными параметрами компонентов формы {$R *.dfm} написать строку:

```
uses indicat;
```

После этого опишем переменные для каждого из индикаторов:

```
var indN1, indN2, indSum: TIndicator;
```

Наконец, создадим стандартным для Delphi образом обработчик события *OnCreate* для формы и в него поместим следующий текст:

```
procedure TForm1.FormCreate(Sender: TObject);
var N1, N2: integer;
begin N1:=1106; N2:=1956;
      indN1:=TIndicator.Create(Form1,70,10);
      indN1.ShowNumber(N1);
      indN2:=TIndicator.Create(Form1,70,74);
      indN2.ShowNumber(N2);
      indSum:=TIndicator.Create(Form1,70,138);
      indSum.ShowNumber(N1+N2);
end;
```

Тестовое приложение с тремя (!) индикаторами уже готово. После запуска программы увидим примерно такую картину, как на рисунке 5.

Непременно стоит еще раз окинуть взглядом главную программу. Помимо того что она очень короткая, в ней нет *никаких* намеков на внутреннее устройство индикатора. Поэтому, например, нетрудно представить себе замену семисегментного индикатора каким-либо другим без изменения текста главной программы.

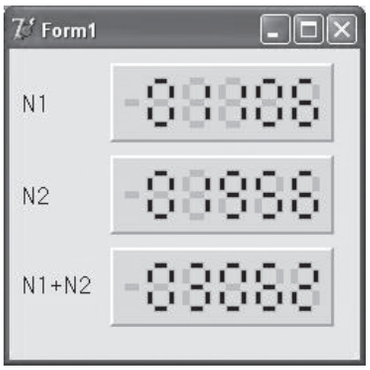


Рис. 5. Тестовая задача с использованием индикаторов

6. Задачи для самостоятельных проектов

Опыт преподавания показывает, что для понимания программы ее надо набрать самому и запустить, самостоятельно исправив все свои ошибки. Но, увы, большинство школьников предпочитают тем или иным способом «заполучить» готовый отлаженный файл, который, естественно, немедленно заработает. Далее, в лучшем случае бегло просмотрев текст, ученик делает вывод (часто достаточно искренне) о том, что он все понял и усвоил.

Существует простой способ проверить, насколько материал в действительности усвоен: надо попробовать модифицировать готовую программу. В ходе этой работы часто выясняется, что далеко не все детали поняты и, как следствие, результата долго не удается достичь.

Приведем ряд вариантов усложнения описанного проекта.

1. Обычным образом при помощи *Палитры компонентов* создать индикатор, который устроен как можно ближе к описанному. Сравнить возможности обоих способов. *Указание*. Используйте компоненты *Shape* и *Panel*.
2. Добавить к форме еще один индикатор для отображения разности чисел. Переделать расположение индикаторов так, чтобы индикаторы, отображающие исходные числа, располагались на одной панели, а результаты — на другой. *Указание*. Учитывайте, что панель является родителем всех тех компонентов, которые на ней находятся.
3. Добавить к форме два компонента *Edit*, в которые будут вводиться значения *N*1 и *N*2. Обеспечить передачу введенных значений на индикаторы и пересчет суммы. *Указание*. Не забывайте, что из поля ввода читается строковое значение и его необходимо перевести в целое число (например, воспользовавшись функцией Delphi *StrToInt*).
4. Написать программу, которая постоянно отображает на двух индикаторах текущие координаты указателя мыши в пикселях.
5. Написать программу, которая отображает на двух индикаторах размеры загруженного из файла в компонент *Image* рисунка.
6. Создать на форме таблицу *StringGrid* для ввода значений целочисленного массива из 10 элементов. Связать первый индикатор с суммой элементов массива, а оставшиеся два — со значениями его максимального и минимального элементов.
7. Создать часы на основе компонента *Timer* и присоединить их к индикаторам, предварительно установив в них нужное количество цифр. Продумать и реализовать возможность коррекции времени. *Указание*. При отладке удобно временно «ускорить» работу часов, уменьшая значение свойства *interval*.
8. Дополнительно к предыдущей задаче добавить мигающие разделители, например, двоеточие между показаниями часов и минут.
9. Реализовать таймер, отображающий промежуток времени между нажатиями на кнопки «Пуск» и «Стоп».
10. Предложенный индикатор отображает 16-битное целое число. Увеличить число разрядов так, чтобы он смог отобразить любое 32-битное число. *Указание*. Вычислите предварительно, сколько десятичных цифр содержит максимальное число такой разрядности.
11. В качестве упражнения модифицировать индикатор так, чтобы цифры на нем располагались не в строку, а в столбик.
12. Переделать индикатор так, чтобы он не отображал незначащие нули слева. Например, вместо числа 00023 должно высвечиваться просто 23. *Указание*. Обязательно проверить правильность отображения числа 0.
13. Дополнительно к предыдущей задаче добиться, чтобы знак «минус» высвечивался в самой правой из свободных цифр, т. е. вместо –00023 было –23.
14. Описанный в статье класс устроен так, что все создаваемые индикаторы имеют одинаковый размер. Сделать так, чтобы размеры сегментов индикатора можно было изменять при его создании. Установить после этого размер индикатора с результатом больше, чем у остальных. *Указание*. Придется добавить еще два параметра в конструктор.
15. Усовершенствовать устройство индикатора так, чтобы он показывал отрицательные и положительные числа разным цветом, подобно тому как это умеет делать Excel.
16. Дополнить цифру индикатора контролем на допустимость введенного значения. Если число не попадает в диапазон 0–9, то высветить символический знак вопроса. Подумайте, возможна ли такая ситуация в описанном индикаторе.
17. Дополнить индикатор возможностью отображать буквы от A до F, т. е. сделать индикатор шестнадцатеричных цифр. Организовать перевод исходного десятичного числа в шестнадцатеричную систему. *Указание*. Продумать, чтобы все цифры и буквы были однозначно различимы. Скажем, заглавная B на индикаторе не будет отличаться от цифры 8.
18. Использовать разработанный в предыдущей задаче индикатор для отображения цвета какого-нибудь компонента в hex-форме, принятой в HTML.
19. Разработать индикатор для вещественного числа. Предусмотреть возможность индикации порядка числа.
20. Поменять взаимное положение сегментов в цифре. Например, сделать у верха сегментов *s*1, *s*3 и *s*6 одинаковую вертикальную координату, а сегменты *s*2, *s*5 и *s*7 выровнять по низу. Возможно, вы захотите сделать какой-то свой вариант.
21. Изучить дизайн семисегментных индикаторов в различных электронных приборах и табло. Поменять дизайн цифр путем рисования сегмента

на компоненте *Image*. *Указание*. Если вы не знакомы с основами графики, почитайте материал, связанный со свойством *Canvas*.

22. Попробуйте написать индикатор так, чтобы он отображал цифры, как это сделано на конвертах.

* * *

Таким образом, оказывается, что технология написания объектно-ориентированной программы заметно отличается от тех приемов, которые обычно применяются при создании простейших визуальных приложений в Delphi. В то же время мы видим, что изучение основ современного объектно-ориентированного подхода к разработке программ в школьном курсе информатики вполне возможно.

Разбор задач, аналогичных обсуждаемой выше, позволяет наглядно продемонстрировать ученикам «многослойный» способ разработки современного программного обеспечения. Его суть заключается в том, что большинство соответствующих технических деталей скрывается (*инкапсулируется*) в нижних программных слоях, тогда как вышележащие слои от необходимости учета этих деталей освобождаются. Благодаря тому что сложность внутреннего устройства объектов никак не влияет на текст использующей их программы, создаются прекрасные условия для повторного применения разработанных объектов в других проектах. Кроме того, когда объект грамотно построен, его можно усовершенствовать без внесения изменений в остальные части программы.

Изложенный выше материал может помочь объяснить школьникам базовые идеи ООП на доступном им уровне. Специально разработанная для статьи наглядная задача интуитивно хорошо понятна, что позволяет сосредоточить внимание учеников на особенностях программирования. Завершив практическую реализацию программы на уроке, учитель может дать задание разработать на ее базе разнообразные индивидуальные проекты.

Список использованных источников

1. *Архангельский А. Я.* Программирование в Delphi 6. М.: Бином-Пресс, 2004.

2. *Еремин Е. А.* Об иерархическом представлении знаний с помощью некоторых программных средств // Информатика и образование. 2009. № 6, 7.

3. Информатика. 7–9 классы. Базовый курс. Теория / под ред. Н. В. Макаровой. СПб.: Питер, 2001.

4. *Конопка Р.* Создание оригинальных компонент в среде Delphi. Киев: НИПФ-ДиаСофт, 1996.

5. *Синтес А.* Освой самостоятельно объектно-ориентированное программирование за 21 день. М.: Вильямс, 2002.

6. *Суворова Н. И.* Информационное моделирование. Величины, объекты, алгоритмы. М.: Лаборатория Базовых Знаний, 2002.

7. *Фридман А. Л.* Основы объектно-ориентированной разработки программных систем. М.: Финансы и статистика, 2000.

8. *Meyer B.* Towards an Object-Oriented Curriculum // 11th International Conference on Technology of Object-Oriented Languages and Systems (TOOLS '93). 1993.

9. *Tait B.* Object Orientation in Educational Software // Innovations in Education and Training International. 1997. V. 34. № 3.

НОВОСТИ

В ВШЭ создали систему, распознающую эмоции в речи

Сотрудники факультета информатики, математики и компьютерных наук Нижегородского филиала Высшей школы экономики создали автоматическую систему, способную определять эмоции по голосу, рассказали в ВШЭ. Разработчики преобразовывали звук в изображения — спектрограммы. Это позволило работать со звуком методами, применяемыми для распознавания изображений. В исследовании использовалась сверточная нейронная сеть. Нейронная сеть может распознать восемь различных состояний: «нейтральный», «спокойный», «счастливый», «грустный», «злой», «испуганный», «отвращение», «удивление». Наиболее успешно программа различает нейтральные и спокойные интонации. А вот счастье и удивление распознаются не всегда: счастье часто воспринимается как страх и печаль, а удивление — как отвращение. Исследователи рассказали, что компьютер правильно определил эмоцию в 70 % случаев.

Психометрическая шкала поможет оценивать степень комфорта человека при общении с роботом

Специалисты Квинслендского технологического университета работают над системой оценки психологического комфорта человека при общении с роботом. В частности, ученые пытаются выяснить, при каких условиях у человека могло бы появиться ощущение взаимопонимания с роботом. Разрабатываемая шкала должна будет позволить дать оценку реакции человека при общении с роботом лицом к лицу. По мнению исследователей, такая мерка может получить широкое применение, например, для выяснения, как люди реагируют на медицинские, обучающие, развлекательные и другие машины на работе и дома. Ученые не исключают, что кто-то может счесть роботов настолько пугающе похожими на людей, что никогда не будет испытывать комфорта при общении с машиной. Пока что, правда, ситуация противоположная. В ходе недавнего исследования, посвященного изучению реакции людей на робота, выяснилось, что ожидания человека от машины слишком завышены — вероятно, из-за того, что роботы в нынешних фантастических фильмах показаны слишком похожими на людей, а реальность еще очень далека от этого.

(По материалам международного компьютерного еженедельника «Computerworld Россия»)

Г. А. Воробьев,
Липецкий государственный педагогический университет имени П. П. Семенова-Тян-Шанского,

И. А. Шуйкова,
Липецкий государственный технический университет,

М. В. Первеев,
Центр поддержки одаренных детей «Стратегия», г. Липецк

ИНТЕЛЛЕКТУАЛЬНО-ОБУЧАЮЩАЯ ИГРА «МАТЕМАТИКИ ПРОТИВ ПРОГРАММИСТОВ»

Аннотация

В статье рассматривается опыт организации образовательно-развивающих игр, соответствующих предметной области ФГОС основного общего образования «Математика и информатика». Описывается игровой турнир, ориентированный на школьников, увлекающихся дисциплинами информационно-математического цикла, апробированный в течение нескольких лет на занятиях Летней многопрофильной школы Центра поддержки одаренных детей города Липецка. Игровые технологии, применяемые в процессе обучения, носят развивающий характер, способствуют развитию мотивации, формированию межпредметных связей, освоению навыков коллективной работы.

Ключевые слова: информационно-математическое образование, игровые технологии, компьютерные технологии в математическом образовании.

Контактная информация

Воробьев Григорий Алексеевич, канд. тех. наук, доцент, доцент кафедры информатики, информационных технологий и защиты информации Липецкого государственного педагогического университета имени П. П. Семенова-Тян-Шанского; *адрес:* 398000, г. Липецк, ул. Ленина, д. 42; *телефоны:* (4742) 32-83-03, 32-83-33; *e-mail:* vorobjev_g_a@mail.ru

Шуйкова Инесса Анатольевна, канд. тех. наук, доцент, доцент кафедры прикладной математики Липецкого государственного технического университета; *адрес:* 398000, г. Липецк, ул. Московская, д. 30; *телефоны:* (4742) 31-15-28, 32-80-71; *e-mail:* shujkova_i_a@inbox.ru

Первеев Михаил Валерьевич, ассистент преподавателя, Центр поддержки одаренных детей «Стратегия», г. Липецк; *адрес:* 398007, г. Липецк, ул. 40 лет Октября, д. 39; *телефон:* (4742) 35-20-23; *e-mail:* perveev_m@mail.ru

G. A. Vorobyev,
Lipetsk State Pedagogical P. Semenov-Tyan-Shansky University,
I. A. Shujkova,
Lipetsk State Technical University,
M. V. Perveev,
Educational Center for Talanted Children "Strategy", Lipetsk

THE EDUCATIONAL GAME "MATHEMATICIANS VS PROGRAMMERS"

Abstract

The article describes the educational game that matches "Mathematicians vs programmers" subject area of the Federal State Education Standards. The game is targeted on schoolchildren which are interested in mathematics and programming. It promotes development of motivation, formation of intersubject communications and mastering of collective work skills.

Keywords: IT and mathematical education, gaming technologies, computer technologies in mathematical education.

Федеральный государственный образовательный стандарт основного общего образования объединяет в одну предметную область математику и информатику. В результате изучения предметной области «Математика и информатика» обучающиеся развивают логическое и математическое мышление, получают представление о математических моделях; овладевают математическими рассуждениями; учатся применять математические знания при решении различных задач и оценивать полученные результаты; овладевают умениями решения учебных задач; развивают математическую интуицию. При освоении указанной области развиваются умения применять изученные понятия, результаты, методы для решения задач практического характера и задач из смежных дисциплин с использованием при необходимости справочных материалов, компьютера, пользоваться оценкой и прикидкой при практических расчетах [6].

Для реализации указанных целей, наряду с другими формами работы, мы предлагаем авторское командное межпредметное соревнование (турнир) «Математики против программистов».

В соревновании могут участвовать команды в составе от трех до пяти человек — учащиеся VI—XI классов, проявляющие заинтересованность в изучении математики и информатики.

Основная цель проведения турнира для школьников — иллюстрация сочетания материала математики и информатики (программирования), возможностей применения компьютерных способов решения задач.

Для участия в соревновании комплектуются команды трех видов:

- «Математики», которые могут применять только математический инструментарий;
- «Программисты», разрабатывающие программный код на одном из популярных языков программирования;
- «Математики и программисты», имеющие право комбинировать математические и программистские методы решения задач.

Команды докладывают решения задач жюри по мере их готовности. Команда, решившая задачу первой, объявляет жюри о готовности сдавать задачу и приглашается в порядке «живой очереди» для выступления с публичным докладом о решении перед всеми желающими участниками турнира. Доклад представляет один участник команды, который не может быть заменен в течение доклада другим участником. Все участники соревнования по своему желанию могут слушать доклад о решении. Длительность доклада — не более 7—10 мин. Решение каждой задачи оценивается по пятибалльной шкале. Если за решение задачи команда получила 4 или 5 баллов, то задача считается «закрытой» и дальше не рассматривается. Если за задачу выставлена оценка от 1 до 3 баллов, то любая из команд в дальнейшем может предъявлять доклад по задаче, пока не будет получена оценка в 4 или 5 баллов. Максимальное количество попыток сдачи каждой задачи от одной команды — три.

Соревнование проводится в два тура в течение одного дня. Для участия во втором туре приглашаются команды, успешно выступившие в первом туре.

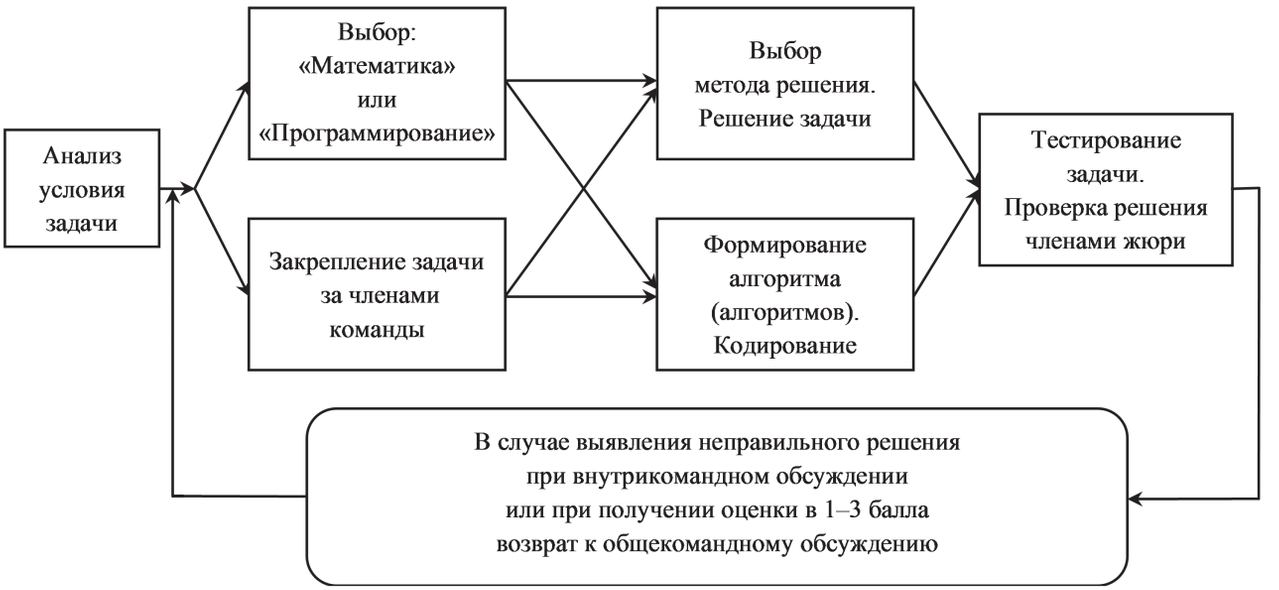


Рис. 1

По итогам соревнования команды ранжируются по набранным баллам. Если баллы равны — то по количеству попыток (более высокий рейтинг у команд, у которых меньше попыток сдачи задач). В процессе турнира ведется онлайн-мониторинг соревнования, схожий с онлайн-мониторингом классических соревнований по программированию.

В рассматриваемом соревновании деятельность команд по решению задач в большей мере приближена к исследовательской деятельности, чем в случае, когда применяется инструментарий одной школьной дисциплины (см. обобщенную схему работы команд над задачами на рисунке 1). Школьники тренируют навыки работы в коллективе, более широко анализируют условия задач и методы их решения.

Для формирования комплекта задач олимпиады могут использоваться различные разделы математики и разные алгоритмы решения задач.

Выделим типичную тематику предлагаемых задач из опыта проведения соревнований (см. табл.).

Естественно, приведенная тематика не охватывает весь спектр задач, которые можно использовать.

Таблица

Типичная тематика задач турнира «Математики против программистов»

№ п/п	Тематический рубрикатор (математика)	Применяемый инструментарий (программирование)
1	Метод математической индукции	Перебор вариантов
2	Принцип крайнего	Реализация
3	Инварианты	Реализация, моделирование
4	Доказательство неравенств	Перебор вариантов
5	Элементы теории графов	Алгоритмы на графах
6	Функции и их свойства	Реализация, моделирование
7	Уравнения и их системы	Численное решение уравнений и их систем. Линейная алгебра
8	Элементы теории чисел	Алгебра. Элементарные алгоритмы
9	Геометрия	Геометрия. Решение задач векторно-координатным методом. Численное решение уравнений и их систем
10	Текстовые задачи на составление уравнений и их систем	Численное решение уравнений и их систем
11	Комбинаторика	Комбинаторные алгоритмы. Динамическое программирование
12	Теория вероятностей	Теория вероятностей. Реализация
13	Игры со стратегией	Построение полного или неполного дерева игры. Теория Шпрага–Гранди

Проанализируем задачи одного из туров (обычно каждый тур содержит 10 задач, но мы дадим их чуть больше). Турнир «Математики против программистов» проводится нами в рамках работы Летней многопрофильной школы Центра поддержки одаренных детей «Стратегия» города Липецка, поэтому в содержании задач присутствует соответствующий антураж.

Задача 1. Изолированные вершины.

Сегодня на занятии по математике младшие школьники узнали, что изолированной вершиной в графе называется вершина, которая не смежна ни с одной другой вершиной (например, таковой является вершина 5 на рисунке 2). Тогда дети заинтересовались: сколько можно нарисовать графов, состоящих из n вершин, в которых есть хотя бы одна изолированная вершина? Вам предлагается решить более простую задачу и найти ответ для $n = 5$. Граф не может иметь петель (ребра, ведущего из вершины в нее же) и кратных ребер (нескольких ребер между одними и теми же вершинами).

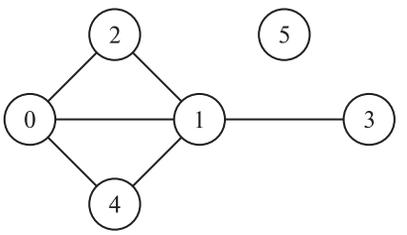


Рис. 2

Решение.

Так как в графе с n вершинами ровно $\frac{n(n-1)}{2}$ ребер, то всевозможных графов на n вершинах без петель и кратных ребер: $2^{\frac{n(n-1)}{2}}$.

Существует $C_n^i \cdot 2^{\frac{(n-i)(n-i-1)}{2}}$ графов с не менее чем i изолированными вершинами.

Тогда по формуле включений-исключений для графа с n вершинами получаем:

$$\sum_{i=1}^{n}(-1)^{i-1} \cdot C_n^i \cdot 2^{\frac{(n-i)(n-i-1)}{2}}.$$

Ответом будет являться:

$$\sum_{i=1}^{5}(-1)^{i-1} \cdot C_5^i \cdot 2^{\frac{(5-i)(4-i)}{2}} = 256.$$

Ответ. 256.

Задача 2. Бинарные последовательности.

Все программисты знают, что всевозможных последовательностей из 0 и 1 длины n может быть ровно 2^n. Попробуйте решить задачу посложнее: пусть никакие две единицы не могут стоять на соседних позициях. Вам предлагается найти ответ для последовательности из 15 цифр.

Решение.

Пусть a_i — количество последовательностей длины i, оканчивающихся на 0, b_i — оканчивающихся на 1, а $f_i = a_i + b_i$ — общее количество искомых последовательностей.

$$a_1 = b_1 = 1,$$
$$b_i = a_{i-1},\ a_i = a_{i-1} + a_{i-2}.$$

Следовательно:

$$f_i = 2a_{i-1} + a_{i-2} =$$
$$= 3a_{i-2} + 2a_{i-3} =$$
$$= 2a_{i-2} + 3a_{i-3} + a_{i-4} =$$
$$= 2a_{i-2} + a_{i-3} + 2a_{i-3} + a_{i-4} =$$
$$= f_{i-1} + f_{i-2}.$$
$$F_i = f_{i-2}.$$

Таким образом, ответом на задачу является $F_{17} = 1597$.

Ответ. $F_{n+2} = F_{17} = 1597$.

Задача 3. Пол для дискотеки.

Администрация многопрофильной школы центра «Стратегия» решила к приезду детей подготовить место для дискотеки. Было решено покрыть танцпол плиткой, каждая плитка имеет размер 1×1 и окрашена в черный или белый цвет. Чтобы узор на полу получился красивым, было решено выкладывать плитку так, чтобы не было ни одного квадрата 2×2 полностью черного или белого цвета.

Учащиеся центра «Стратегия» заинтересовались: сколько различных узоров могло получиться в таком случае? Так как точные размеры танцпола неизвестны, вам предлагается решить задачу для танцпола размером 2×5.

Решение.

Назовем профилем последний замощенный столбец (размера 2×1). Обозначим через w_i количество всевозможных замощений поля $2 \times i$ таких, что профиль состоит только из белых клеток и условие задачи выполняется, b_i — только из черных, а g_i — из одной черной и одной белой клеток.

$$w_1 = b_1 = 1;$$
$$g_1 = 2;$$
$$w_i = g_{i-1} + b_{i-1};$$
$$b_i = w_{i-1} + g_{i-1};$$
$$g_i = 2 \cdot (w_{i-1} + g_{i-1} + b_{i-1}).$$

Ответом на задачу будет являться:

$$w_5 + g_5 + b_5 = 139 + 356 + 139 = 634.$$

Ответ. 634.

Задача 4. Ночные нарушители.

Прошлой ночью директор летней школы обнаружил трех людей, которые нарушали дисциплину в лагере. Теперь он придумал им небольшое наказание. Трое нарушителей сначала пишут мелками на плитке во дворе последовательность цифр 123. Затем на каждой итерации они пишут N единичек, затем N двоек, затем N троек, где N — это номер итерации (первая итерация — это данная последовательность: 123). То есть у них получается последовательность цифр: 123112233111222333111122... Как только они напишут миллион цифр, они приступают к новому заданию: подсчитать, сколько раз в этой последовательности встречается группа цифр 12. Нарушители хотят передохнуть, поэтому попросили вас помочь им с ответом на этот вопрос.

Замечание. Для ответа достаточно привести оценку верхней действительной границы диапазона в радикалах.

Решение.

Последовательность 12 встречается каждый раз как последняя из единиц и первая из двоек.

Пусть n — номер последней полностью выполненной итерации. Тогда получим неравенство:

$$1 + 2 + ... + (n + 1) + 1 + ... + n + 1 + 1 + ... + n \le 1\,000\,000.$$

В результате преобразований получим:

$$3n^2 + 5n + 3 - 2\,000\,000 \le 0.$$

Следовательно:

$$n \le \frac{-5 + \sqrt{24 \cdot 10^6 - 11}}{6}.$$

Максимальное значение n равно 815.

Ответ. $n + 1 = 816,\ n + 1 \le 1 + \frac{-5 + \sqrt{24 \cdot 10^6 - 11}}{6}$. Для математиков возможна оценка верхней действительной границы диапазона в радикалах. Поэтому первый вариант ответа для программистов, а второй — для математиков.

Задача 5. Метаслово языка Тумба-Юмба.

В летней школе устроили день племени Тумба-Юмба. В этом племени всего две буквы — Т и Ю, а все слова состоят в точности из четырех букв этого языка. Все дети вынуждены в этот день говорить на языке племени, например, используя слова ТЮТЮ, ЮЮЮТ и т. д. Воспитатели начали ночью учить язык племени, но им никак не удавалось запомнить больше, чем несколько слов. Что же делать? Они решили попросить программистов составить такое метаслово, которое имело бы минимальный размер (минимально возможную длину строки) и включало бы в себя в качестве подстрок все слова языка племени Тумба-Юмба. Помогите составить такое метаслово.

Решение.

Один из вариантов математического решения — составление графа де Брейна (рис. 3). Вершины — четырехмеры (все слова языка). Ребра — направленные и идут от вершины-суффикса к вершине-префиксу (в суффикс и префикс входит три символа). По такому графу ищем гамильтонов цикл, выписываем из первой вершины все четыре символа, далее дописываем по одному символу и получаем метаслово длиной 19 символов.

Меньше 19 символов недостаточно, в этом случае будет не более 15 слов (слова могут начинаться с 1, 2, ..., 15 символов). А слов в языке племени Тумба-Юмба: $2^4 = 16$.

Рис. 3

Программисты могли решать задачу способом, описанным выше. Составляется граф де Брейна, хранить который в данном случае удобнее в виде матрицы смежности. В отличие от предложенного выше математического решения, вершинами в графе будут являться всевозможные суффиксы слов языка — тримеры, а на ребрах будут написаны четырехмеры. Таким образом, для решения задачи в полученном графе необходимо будет найти эйлеров цикл, что является гораздо более легкой задачей по сравнению с поиском гамильтонова цикла. Найти эйлеров цикл в графе можно при помощи обхода в глубину.

Ответ. ТТЮЮЮЮТТЮТЮЮТЮТТТТЮ — один из вариантов решения.

Задача 6. Шахматный конь.

На шахматной доске стоит конь. Двое играющих ходят по очереди. За один ход разрешается передвинуть коня на две клетки вверх и потом на одну клетку влево или вправо либо на две клетки вправо и потом на одну клетку вверх или вниз. Проигрывает тот, кому некуда ходить.

Подсчитайте, при каком количестве начальных положений коня первый игрок может гарантированно выиграть (если будет использовать оптимальную стратегию).

Решение.

Математическое решение предполагает анализ игры.

Начнем с правого верхнего угла доски. Будем указывать цифрой 2 начальные положения коня, для которых второй игрок при правильной игре выигрывает. Цифрой 1 будем отмечать начальные положения коня, для которых первый игрок при правильной игре выигрывает (рис. 4).

8	1	1	2	2	1	1	2	2
7	1	1	2	2	1	1	2	2
6	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1
4	1	1	2	2	1	1	2	2
3	1	1	2	2	1	1	2	2
2	1	1	1	1	1	1	1	1
1	2	1	1	1	1	1	1	1
	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**

Рис. 4

Для команды программистов возможно решение, аналогичное представленному выше. Будем хранить матрицу 8×8, в которой будут цифры 1 и 2. Анализ начинаем с клетки с координатами (0; 7) — правый верхний угол. Обходить доску удобно при помощи рекурсии.

Ответ. 47.

Задача 7. Гномы и Таня.

В круге сидят $3n$ гномов. У каждого гнома может быть от одной до трех монет. Пронумеруем месторасположения гномов в порядке следования по кругу числами от 0 до $(3n - 1)$. Пусть у гнома, сидящего на i месте, a_i монет. Тогда, если существует целое число i $(0 \le i < n)$, такое, что $a_i + a_{i+n} + a_{i+2n} \ne 6$, то Таня остается довольна.

Посчитайте количество способов выбора a_i, при которых Таня осталась довольна.

Важное замечание. Для математиков необходима только формула, а для программистов — еще и число, равное ответу на задачу при $n = 1000$ по модулю $10^9 + 7$.

Решение.

Количество различных способов рассадки всех гномов равно 3^{3n}. Посчитаем количество способов рассадить гномов так, чтобы в любом треугольнике гномы получили шесть монет, а потом вычтем это число из всех способов. Заметим, что все гномы однозначно разбиваются на треугольники вида: i; $(i + n)$; $(i + 2n)$. Поэтому можно посчитать количество способов независимо по каждому треугольнику, а потом перемножить результаты.

Чтобы получить ответ для треугольника, осталось заметить, что существует ровно семь способов взять гномов с шестью монетами (это все перестановки {1, 2, 3} и {2, 2, 2}). Получаем ответ: $3^{3n} - 7^n$.

Для команды программистов по уже представленному выше решению получаем, что ответ равен $3^{3n} - 7^n$. Найти это число на компьютере не составит особого труда.

Ответ. $3^{3n} - 7^n$, для $n = 1000$ получаем 356 882 060.

Задача 8. Сумма коэффициентов.

После преобразования выражения:

$$x^2 + x + 1 + (x^2 + x + 1)^2 + (x^2 + x + 1)^3 + ... + (x^2 + x + 1)^{12}$$

получился многочлен 24-й степени. Определите сумму числовых коэффициентов полученного многочлена при степенях с нечетными показателями.

Решение.

Обозначим данное выражение через $f(x)$. Тогда $f(1)$ — сумма всех коэффициентов полученного многочлена, $f(-1)$ — сумма коэффициентов при степенях с четными показателями минус сумма коэффициентов при степенях с нечетными показателями.

$\frac{f(1) - f(-1)}{2}$ — искомое значение.

$f(1)$ — сумма первых 12 членов геометрической прогрессии с первым членом, равным 3, и знаменателем, равным 3. Формула суммы n первых членов геометрической прогрессии $(q \ne 1)$:

$$S_n = \frac{b_1(q^n - 1)}{q - 1}.$$

$$f(1) = \frac{3 \cdot (3^{12} - 1)}{3 - 1} = \frac{3 \cdot (3^{12} - 1)}{2}.$$

$$f(-1) = 12.$$

Искомая величина:

$$\frac{f(1) - f(-1)}{2} = \frac{3 \cdot (3^{12} - 1)}{2} - 6 = 398\,574.$$

Для программистов решение может заключаться в написании процедуры для определения коэффициентов многочленов при их умножении. В дальнейшем можно многократно обращаться к полученной процедуре.

Ответ. 398 574.

Задача 9. Велодорожки.

План велодорожек лагеря «Клен» имеет схему, изображенную на рисунке. На всех дорожках введено одностороннее движение: можно ехать только вправо или вверх. Сколько есть разных маршрутов, ведущих из точки A в точку B, при условии, что поворотов ровно пять? Один из примеров маршрута выделен на рисунке 5 жирной линией.

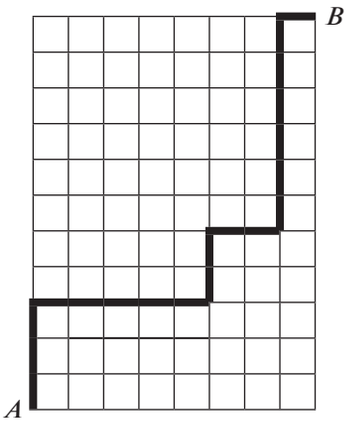


Рис. 5

Решение.

Пусть первое движение было вверх. У каждого перекрестка две координаты узла, на пересечении которого находится клетка. Если мы движемся от точки A (1, 1) до точки B (9, 12), то маршрут имеет вид:

$$(1, 1) - (1, y_1) - (x_1, y_1) - (x_1, y_2) - (x_2, y_2) - (x_2, 12) - (9, 12).$$

Он однозначно восстанавливается по точкам (x_1, y_1) и (x_2, y_2), в которых были сделаны второй и четвертый повороты.

Таким образом, нужно выбрать две пары:

$$1 < x_1 < x_2 < 9,$$

$$1 < y_1 < y_2 < 12.$$

Выбрать такие числа x_1 и x_2 — то же, что выбрать два из шести разных предметов: это можно сделать $C_7^2 = \frac{7 \cdot 6}{2} = 21$ способами. Количество способов выбора y_1 и y_2 равно $C_{10}^2 = \frac{10 \cdot 9}{2} = 45$. Значит, выбрать x_1, x_2, y_1 и y_2 можно $21 \cdot 45 = 945$ способами. Столько же способов для случая, когда первый ход был вправо. Всего способов $2 \cdot 945 = 1890$.

Программисты рекурсивно могут перебрать всевозможные маршруты из левого нижнего угла плана в правый верхний. В качестве параметров рекурсии храним текущие координаты, координаты предыдущей клетки и количество совершенных поворотов. Когда мы приходим в правый верхний угол, если количество поворотов равно пяти, то прибавляем ответ. Таким образом мы переберем все возможные варианты.

Ответ. 1890.

Задача 10. Размножение бактерий.

Нерадивый школьник утащил стакан вкусного компота из столовой к себе в комнату. Вначале в стакане завелось ровно три бактерии. Через минуту их суммарное количество увеличилось до восьми. Будем обозначать количество живых бактерий после i-й минуты как $f(i)$. Каждую i-ю минуту, начиная с третьей, количество бактерий увеличивалось следующим образом: от каждой живой бактерии появлялись две новые бактерии, после этого ровно $f(i-2)$ живых бактерий умирало.

Требуется найти количество живых бактерий через 10 000 002 017 минут.

Для математиков необходима только формула, а для программистов еще и число, равное ответу на задачу, взятому по модулю $10^9 + 7$.

Решение.

Для $n \geq 3$ количество бактерий, живых после n-й минуты, можно выразить следующей формулой:

$$f(n) = 3 \cdot f(n-1) - f(n-2).$$

Покажем теперь по индукции, что $f(n) = F_{2n+1}$, где F_k — k-е число Фибоначчи.

База индукции:

$$f(1) = 3 = F_3,$$

$$f(2) = 8 = F_5.$$

Пусть утверждение верно для $k - n - 1$ и $m - n - 2$. Тогда:

$$f(n) = 3 \cdot f(n-1) - f(n-2) =$$

$$= 2 \cdot f(n-1) + f(n-1) - f(n-2) =$$

$$= 2 \cdot F_{2n-1} + F_{2n-1} - F_{2n-3} =$$

$$= 2 \cdot F_{2n-1} + F_{2n-2} + F_{2n-3} - F_{2n-3} =$$

$$= 2 \cdot F_{2n-1} + F_{2n-2} =$$

$$= F_{2n-1} + F_{2n-1} + F_{2n-2} =$$

$$= F_{2n-1} + F_{2n} = F_{2n+1}.$$

Предположение индукции выполнено. Утверждение доказано.

$$f(10\,000\,002\,017) = F_{20000004035}.$$

Программисты могут воспользоваться динамическим программированием. Будем последовательно вычислять значения $f(i)$ для всех $3 \leq i \leq 10\,000\,002\,017$. Для экономии памяти будем хранить не весь массив f, а только три числа.

Ответ. $f(10\,000\,002\,017) = F_{20000004035}$, по модулю $10^9 + 7$ получим 950 957 558.

Задача 11. Разнообразие паркетов.

Дорогие друзья, спешим сообщить вам, что в настоящее время известны все типы всех возможных паркетов для всех n-угольников. Мы думаем, что вам будет интересно, что долгое время математики не могли доказать конечное количество типов паркетов для пятиугольников ($n = 5$). Мы не будем вас просить доказать, какое же конечное число типов паркетов существует для пятиугольников, — этот факт был доказан буквально недавно, в мае 2017 года. Мы хотим попросить вас привести рисунки всевозможных типов паркетов из пятиугольников.

Внимание! Баллы будут засчитываться за каждый предложенный и правильно описанный тип паркета — необходимо привести его рисунок и указать на свойства выпуклого пятиугольника, из которого складывается паркет. В качестве свойства пятиугольника укажите соотношение углов в фигуре и соотношение сторон. Команда имеет право предлагать по одному варианту паркета сразу, как только найдет его. Также напомним вам, что паркет или замощение — это разбиение плоскости многоугольниками (или пространства многогранниками) без пробелов и перекрытий.

Решение.

Узнать обо всех типах паркетов для выпуклых многоугольников можно в статье [2]. От участников турнира требовалось описать найденный паркет: привести свойства многоугольника и сделать иллюстрацию.

* * *

В заключение отметим, что предлагаемый вариант игрового турнира расширяет спектр соревнований, направленных на повышение уровня компетентности современных школьников, способствует их профориентации и в дальнейшем практической деятельности в смежных областях математики и информатики.

Список использованных источников

1. *Агаханов Н. Х., Подлипский О. К.* Математика. Районные олимпиады. 6–11 классы. М.: Просвещение, 2010.

2. *Андреев Н., Панов М.* Паркеты из выпуклых многоугольников // Квант. 2017. № 5.

3. *Андреева А. Н., Барабанова А. И., Чернявский Я. Н.* Саратовские математические олимпиады. М: МЦНМО, 2013.

4. *Омельченко А. В.* Основы теории графов. Электронный дистанционный курс. http://stepic.org

5. Сайт олимпиад в области точных наук Санкт-Петербургского национального исследовательского университета информационных технологий, механики и оптики. http://olymp.ifmo.ru

6. Федеральный государственный образовательный стандарт основного общего образования (в ред. Приказа Минобрнауки России от 29.12.2014 № 1644). http://минобрнауки.рф/документы/938/файл/749/приказ Об утверждении 1897.pdf

М. Е. Самойлова,
Мордовский государственный педагогический институт им. М. Е. Евсевьева, г. Саранск, Республика Мордовия

СОЗДАЕМ ВИДЕОЗАГАДКИ НА YOUTUBE

Аннотация

В статье описывается один из приемов приобщения школьников к созданию видео на компьютере. В качестве программной платформы выбран видеоредактор YouTube Video Editor. Продемонстрирован пример создания видео, и разработана инструкция по созданию видеозагадок.

Ключевые слова: видеозагадка, обучение, видеохостинг, мультипликация.

Контактная информация

Самойлова Марина Евгеньевна, магистрант 1-го курса физико-математического факультета Мордовского государственного педагогического института им. М. Е. Евсевьева, г. Саранск, Республика Мордовия; *адрес:* 430007, Республика Мордовия, г. Саранск, ул. Студенческая, д. 11а; *телефон:* (8342) 33-92-84; *e-mail:* sme3194@gmail.com

M. E. Samoilova,
Mordovian State Pedagogical Institute named after M. E. Evsevjev, Saransk, Mordovia

CREATING VIDEO PUZZLES ON YOUTUBE

Abstract

The article describes one of the ways of involving schoolchildren in creating a video on a computer. As a software platform the YouTube Video Editor is selected. An example of creating a video is shown and the manual on creating video puzzles is given.

Keywords: video puzzle, training, video hosting, animation, multiplication.

Вне зависимости от возраста многим нравится отгадывать загадки — «метафорические выражения, в которых один предмет отображается через посредство другого, имеющего с ним хотя бы отдаленное свойство» [2].

Загадка является инструментом кодирования информации, т. е. косвенно связана с одним из разделов, который изучается в школьном курсе информатики, — «Кодирование информации». Обычно загадки используются в начальной школе, когда через них младшие школьники по характеристическим признакам определяют объект. В старшей школе загадки практически не применяются. Нами предлагается разработка, в которой показано, как ученики могут самостоятельно разрабатывать и оформлять свои загадки, используя разнообразные информационные ресурсы, например, создавая их в виде видеороликов. Такие загадки можно назвать видеозагадками. ***Видеозагадка*** *— это последовательность сменяющихся кадров с изображениями объектов, каждый из которых раскрывает как минимум одно свойство разгадываемого объекта*. Для усиления воздействия видео подкрепляется музыкальным сопровождением. В некоторых случаях видеозагадка снабжается субтитрами.

Создавать видеозагадки можно различными способами: это могут быть приложения, устанавливаемые на компьютер, например, Sony Vegas Pro, Pinnacle Studio, VSDC Free Video Editor, либо соответствующие интернет-ресурсы (VideoToolbox, FileLab Video Editor, Life.Film). Но для создания видеозагадок можно задействовать и функционал видеохостинга YouTube, когда с использованием встроенного видеоредактора YouTube Video Editor из отдельных кадров формируется видео.

Обучение созданию видеозагадок начинается с доведения информации до учеников о том, что понимается под загадкой вообще и как можно разрабатывать загадки. Далее показывается работа с видеоредактором YouTube Video Editor. В качестве демонстрации можно предложить, например, разгадать следующую видеозагадку: https://youtu.be/zO3R4jrRTDc

Теоретическая часть.

Под ***загадкой*** *понимается образная модель объекта, описываемая в аллегорической форме совокупностью признаков, принадлежащих другим объектам*.

Загадки разрабатываются следующим образом [3].

Выбирается загадываемый объект, например матричный принтер. Выделяются его ключевые признаки — наличие печатающих иголок, используемый носитель для печати. Иголки есть у ежика, кактуса и некоторых других объектов. Материальный носитель, с которым работает матричный принтер, — бумага. Используя эти признаки, получается следующий вариант загадки:

У него иголки есть,
Может он бумагу съесть.

Взяв за основу принцип печати, можно видоизменить загадку:

У него иголки есть,
Выпусть их — и можешь текст прочесть.

Загадка может приобрести и другой вид:

Иголочки-уколочки он резко выпускает,
Текст, рисунки ими набирает.

Здесь добавлен еще один признак — «резко», т. е. дано указание на скорость работы устройства.

Как видно из примера, степень аллегоричности существенно варьируется.

В качестве примеров ученикам можно привести загадки к той или иной содержательной линии школьного курса информатики.

Приведем примеры к содержательной линии «Информация и информационные процессы».

1. Между мною и тобою
   Он находится всегда,
   Может мягким быть, тяжелым,
   Не исчезнет никуда.

   (*Канал связи*.)

2. Маленький мальчонка,
   Короткие ручонки,
   Глазками моргает,
   Единицы и нули запоминает.

   (*Бит*.)

Практическая часть.

Ученикам выдается инструкция по созданию видеозагадки.

1. Определить объект, который будет загадан (например, снеговик).
2. Для объекта подобрать признаки, которые характерны для других объектов (снеговик состоит из снега, морковки, ведра, пуговиц и метлы).
3. Найти или создать рисунки, которые будут использоваться в видеозагадке (зима, дети лепят снежки, ведро, пуговицы, метла, морковь) (рис. 1).
4. Авторизоваться на видеохостинге YouTube и перейти в личный кабинет.
5. Нажать кнопку *Добавить видео*, чтобы попасть на страницу видеоредактора YouTube Video Editor.
6. В правом меню *Создание видео* нажать кнопку *Создать слайд-шоу*. Откроется всплывающее окно, в которое следует загрузить снимки или выбрать существующие файлы в Google+ (рис. 2).

Рис. 1. Образцы для создания видеозагадки

Выбор завершить нажатием кнопки *Далее* для последующего редактирования слайдов.

7. Нажать кнопку *Далее* для перехода в режим *Изменение настройки*. Задать время показа кадров, выбрать эффекты слайдов и перехода, фоновую музыку. Так как аудиодорожки выбираются только из предложенного списка, подобрать соответствующее музыкальное сопровождение.

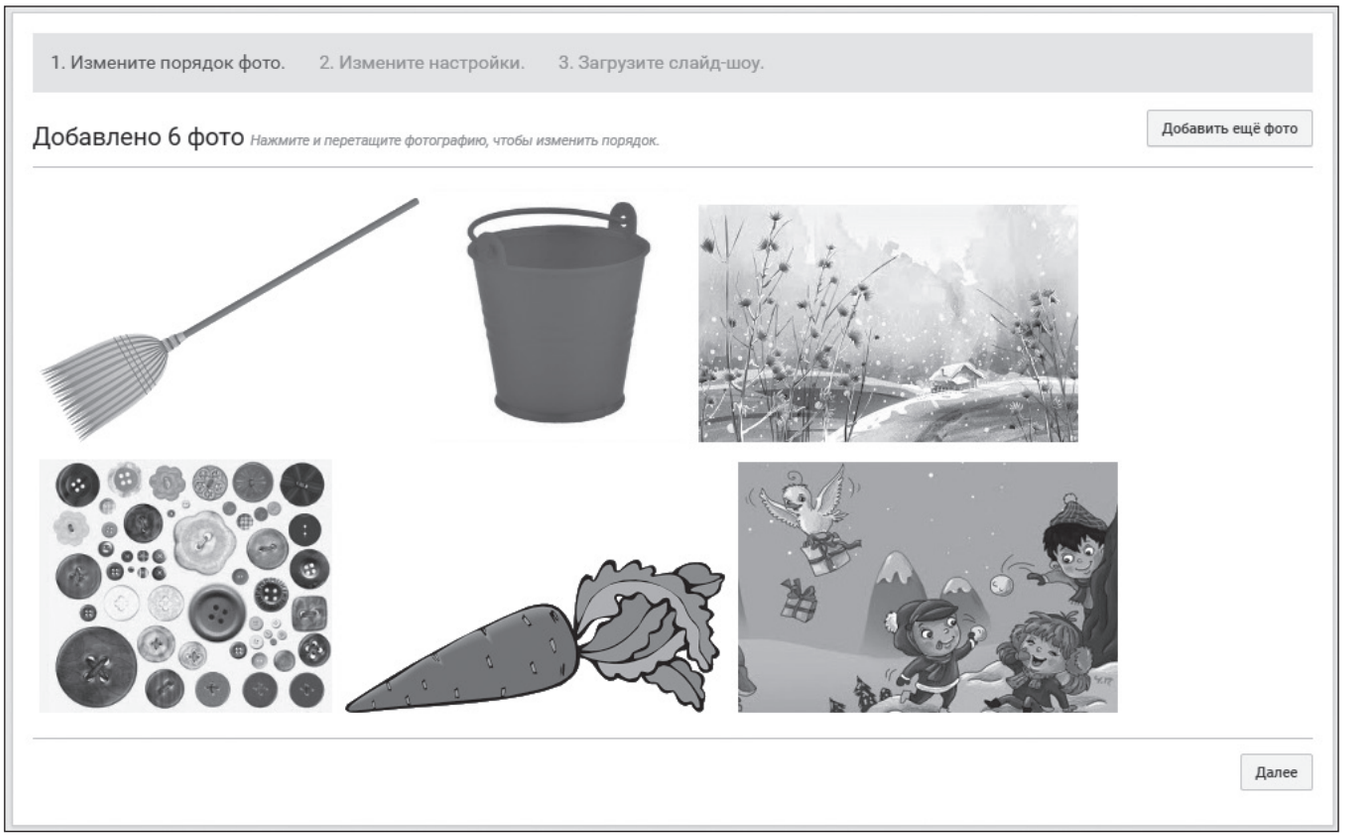


Рис. 2. Окно загрузки изображений в видеоредактор YouTube Video Editor

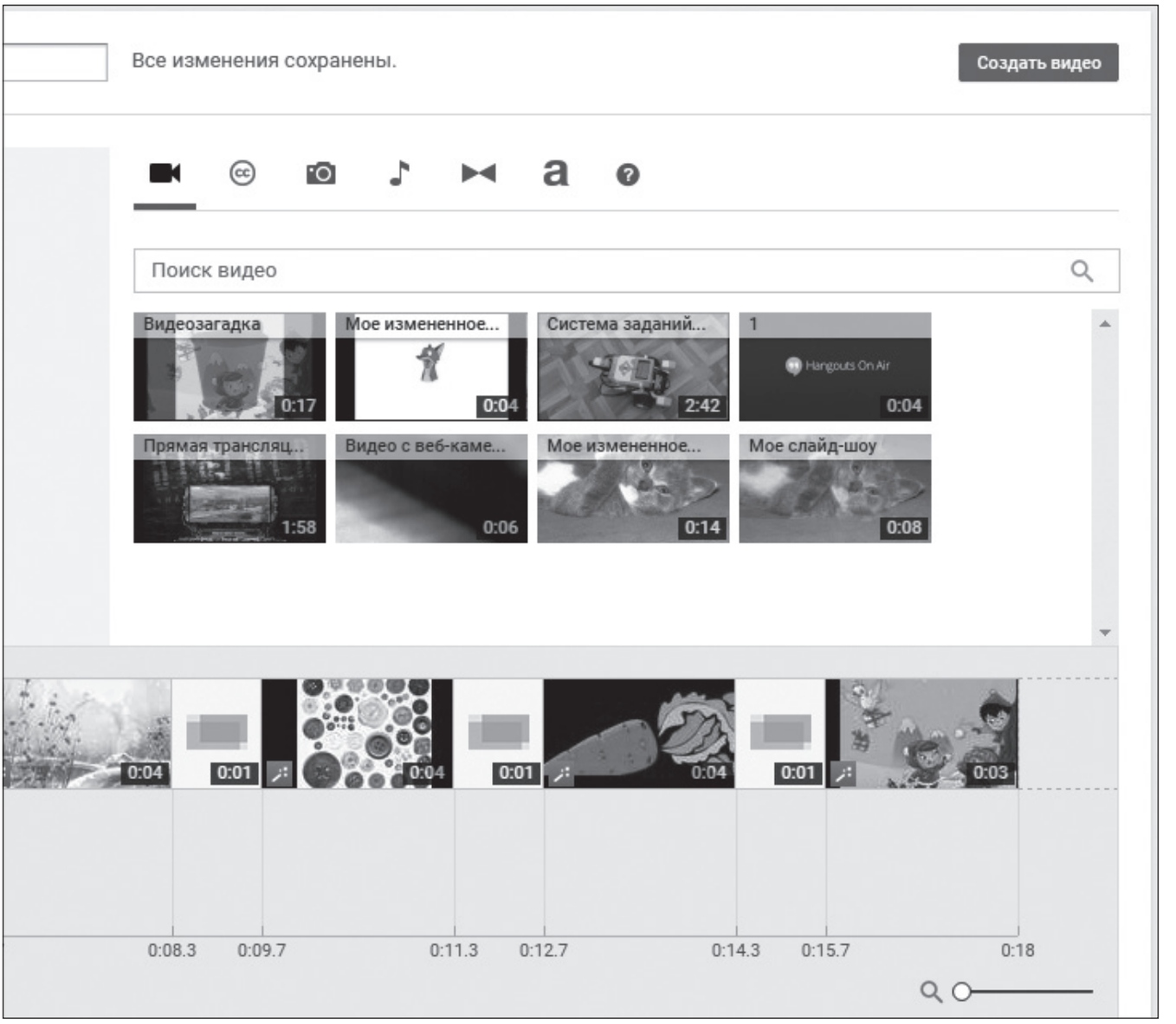


Рис. 3. Созданная видеозагадка в видеоредакторе YouTube Video Editor

8. Для более тонких настроек по созданию слайд-шоу выше окна предпросмотра нажать кнопку *Расширенный редактор*. В открывшемся редакторе слайд-шоу при необходимости поменять местами изображения, добавить или удалить переходы, изменить продолжительность каждого из них, вставить субтитры и текстовые надписи.
9. После обработки видео нажать кнопку *Создать видео* (рис. 3).

Предложить описанный проект по созданию видеозагадок можно ученикам старшей школы в десятом классе: в рамках изучения раздела «Современные технологии создания и обработки информационных объектов» (УМК Л. Л. Босовой) [1] либо на углубленном уровне в рамках изучения раздела «Информационные технологии» (УМК И. Г. Семакина) [4].

Созданные видеозагадки можно использовать в качестве заданий для проведения конкурсов и викторин как по информатике, так и по другим учебным дисциплинам.

Список использованных источников

1. *Босова Л. Л., Босова А. Ю.* Информатика. 10 класс: учебник. М.: БИНОМ. Лаборатория знаний, 2016.

2. *Брокгауз Ф. А., Ефрон И. А.* Энциклопедический словарь: Философия и литература. Мифология и религия. Язык и культура. М.: Эксмо, 2004.

3. *Зубрилин А. А.* Занимательные материалы по информатике: словесные головоломки, ребусы, загадки // Информатика в школе. 2010. № 3.

4. *Семакин И. Г., Шеина Т. Ю., Шестакова Л. В.* Информатика. 10 класс. Углубленный уровень: учебник: в 2 ч. Ч. 2. М.: БИНОМ. Лаборатория знаний, 2016.

Д. А. Кельдышев, Ю. В. Иванов, В. А. Саранин,
Глазовский государственный педагогический институт им. В. Г. Короленко, Удмуртская Республика

ОСОБЕННОСТИ МЕТОДИКИ ОБУЧЕНИЯ ОСНОВАМ РОБОТОТЕХНИКИ В СЕЛЬСКИХ ШКОЛАХ ВО ВРЕМЯ КРАТКОСРОЧНЫХ КУРСОВ*

Аннотация

В статье рассматриваются проблемы разработки методики обучения школьников основам робототехники в рамках краткосрочного курса с использованием платформ Arduino и LEGO MINDSTORMS. Представлены результаты апробации такого курса на базе четырехдневного детского лагеря «ТехноЮность».

Ключевые слова: образовательная робототехника, сетевое взаимодействие, Arduino, LEGO MINDSTORMS.

Контактная информация

Кельдышев Денис Александрович, ст. преподаватель кафедры математики и информатики Глазовского государственного педагогического института им. В. Г. Короленко, Удмуртская Республика; *адрес:* 427621, Удмуртская Республика, г. Глазов, ул. Первомайская, д. 25; *телефон:* (34141) 5-58-57; *e-mail:* denis.keldishev@yandex.ru

Иванов Юрий Владимирович, канд. пед. наук, доцент, доцент кафедры физики и дидактики физики Глазовского государственного педагогического института им. В. Г. Короленко, Удмуртская Республика; *адрес:* 427621, Удмуртская Республика, г. Глазов, ул. Первомайская, д. 25; *телефон:* (34141) 5-58-57; *e-mail:* iva-novs@list.ru

Саранин Владимир Александрович, доктор физ.-мат. наук, профессор, профессор кафедры физики и дидактики физики Глазовского государственного педагогического института им. В. Г. Короленко, Удмуртская Республика; *адрес:* 427621, Удмуртская Республика, г. Глазов, ул. Первомайская, д. 25; *телефон:* (34141) 5-58-57; *e-mail:* val-sar@yandex.ru

D. A. Keldyshev, Yu. V. Ivanov, V. A. Saranin,

Glazov State Pedagogical Institute named after V. G. Korolenko, Udmurtia

FEATURES OF METHODICS OF TEACHING ON ROBOTICS IN RURAL SCHOOLS DURING THE SHORT-TERM COURSE

Abstract

The article describes the problems of developing methodics of teaching students on robotics through the short-term course based on Arduino and LEGO MINDSTORMS. The results of approbation of such course at the four-day children's camp "TechnoYunost" are presented in the article.

Keywords: robotics, network cooperation, Arduino, LEGO MINDSTORMS.

На сегодняшний день существует объективный запрос государства на специалистов в области физико-технического конструирования и робототехники. Их подготовка требует формирования соответствующей мотивации, закладываемой еще в процессе обучения в школе. Для этого в настоящее время в России организуются образовательные центры, оснащенные современными лабораториями («Кванториум», «Школьный технопарк» и т. д.). Однако все они построены на стационарной основе и присутствуют далеко не во всех регионах. Для регионов с высокой плотностью населения выездное посещение школьниками таких центров достаточно просто и малозатратно. В регионах с малой плотностью населения (например, таких, как Удмуртская Республика) среднее расстояние от сельских школ до районных центров составляет порядка 30–50 км. В этом случае обеспечить сельским школьникам регулярный доступ к высокотехнологичным учебным лабораториям весьма затруднительно. Решение данной проблемы возможно путем создания мобильных лабораторий, работающих по сетевому принципу и курируемых крупными региональными образовательными центрами.

В связи с этим нами был предложен **проект «Разработка методологии сетевого взаимодействия образовательных учреждений по развитию творческих способностей сельских школьников на основе мобильной лаборатории физики и робототехники»**.

В рамках реализации этого проекта необходимо:

- разработать критерии отбора оборудования для таких лабораторий;
- обосновать принципы использования лабораторий на основе сетевого взаимодействия образовательных учреждений;
- разработать методику обучения школьников на основе мобильных лабораторий.

В настоящей статье рассматривается вопрос разработки методики обучения, который связан с созданием и программированием робототехнических систем.

Для создания и программирования робототехнических систем и проведения в дальнейшем с помощью них физических экспериментов рассмотрим наиболее популярные образовательные платформы для занятий робототехникой: набор LEGO MINDSTORMS Education EV3 и образовательный набор по робототехнике «Матрешка» на базе микроконтроллера Arduino.

В рамках сетевого сотрудничества участникам сети будет передана мобильная лаборатория, которая кроме образовательного набора по робототехнике будет включать также инструкции по выполнению проектов. С помощью них учащиеся будут выполнять исследовательские работы в области физики, создавая и программируя робототехнические системы. Эти работы они будут проводить без участия представителей центра под руководством своего школьного учителя.

Образовательные наборы по робототехнике — это оборудование, с которым учащиеся никогда не работали. Поэтому перед началом выполнения проектов возникает необходимость показать основы работы с мобильной лабораторией.

Таким образом, необходимо разработать **краткий курс по основам использования мобильной лаборатории,** который **должен удовлетворять следующим требованиям:**

* Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований и Удмуртской Республики в рамках научного проекта № 17-16-18017.

- Курс должен быть кратким, т. е. для одного вида оборудования (LEGO MINDSTORMS Education EV3 или Arduino) продолжительность знакомства не должна превышать трех астрономических часов. Данное ограничение по времени связано с тем, что представители центра должны уложиться в один день поездки.
- Курс должен содержать в себе информацию о принципах работы и назначении основных компонентов мобильной лаборатории, так как они чаще всего будут использоваться в проектах, а о назначении второстепенных элементов учащиеся узнают во время выполнения проектов.
- Курс должен знакомить учащихся с языком программирования и особенностями построения программы в этом языке, поскольку для выполнения проектов учащимся необходимо программировать робототехнические системы.
- Курс должен соответствовать основным дидактическим принципам: доступности, понятности, новизны, т. е. он должен вызывать у учащихся желание работать с лабораторией и выполнять проекты, также курс должен предоставлять обучающимся возможность для развития новых компетенций.

Для проверки выполнимости этих требований был проведен **педагогический эксперимент на базе четырехдневного лагеря «ТехноЮность»**. В работе лагеря принимали участие 40 учеников VIII—X классов. Занятия в лагере проводились с целью популяризации науки и технического творчества в рамках работы детского технопарка в городе Глазове. Детям в начале смены были продемонстрированы различные направления работы технопарка, среди которых есть и направление «Робототехника», в рамках которого ребята работают с наборами LEGO MINDSTORMS Education EV3 и Arduino.

В первый день работы лагеря «ТехноЮность» всего за три часа учащиеся познакомились с возможностями образовательных наборов по робототехнике, увидели своими глазами то, что они смогут создать сами. Это вызвало неподдельный интерес детей к предстоящему обучению. Таким образом, в лагере решались задачи, соответствующие требованиям, предъявленным нами к ознакомительному курсу по использованию мобильной лаборатории.

Опишем **курс по направлению «Робототехника» с LEGO MINDSTORMS Education EV3**, обучение по которому прошли участники лагеря «ТехноЮность».

При создании курса были учтены рекомендации, в которых описана методика использования данного набора [1–3, 5].

На первом занятии учащиеся знакомились с набором LEGO MINDSTORMS Education EV3 и средой программирования для данного набора EV3-G.

Далее ребятам предлагалось создать робота-тележку для прохождения заданной трассы, используя только блоки рулевого управления. В рамках данной деятельности на первом занятии перед учащимися были поставлены следующие задачи:

- узнать названия и назначение датчиков, моторов и других часто используемых деталей конструктора;
- изучить структуру языка программирования EV3-G, который необходим для создания программы для робота;
- создать базовую приводную платформу (двухмоторную тележку);
- создать, загрузить, найти и запустить программу на роботе.

На втором занятии учащиеся создали и запрограммировали робота для выполнения задания «Кегельринг» [5, с. 210]. В данном задании роботу необходимо вытолкнуть из круга объекты (алюминиевые банки объемом 0,33 л). Все учащиеся справились с данным заданием. При этом они узнали принципы работы ультразвукового датчика и датчика цвета, выяснили, как в среде EV3-G реализуется цикл while, установили и использовали ультразвуковой датчик и датчик цвета.

На третьем занятии учащимся предлагалось создать робота для выполнения задания «Лабиринт» с использованием релейного алгоритма [5, с. 170]. Все учащиеся успешно выполнили задание. При выполнении данного задания они узнали принцип работы датчика касания, применили «правило правой руки» (один из способов прохождения лабиринта), узнали способ реализации релейного алгоритма с помощью блока «переключатель» в среде EV3-G.

После проведения всех занятий учащимся был предложен ***тест*** продолжительностью 20 минут.

В первом задании теста учащимся необходимо было подписать каждый электронный компонент набора LEGO MINDSTORMS Education EV3 (рис. 1).

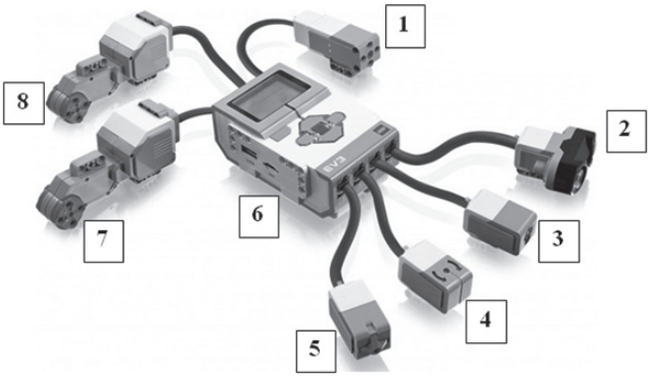


Рис. 1

В задании были представлены восемь электронных компонентов, каждый правильно подписанный элемент оценивался 1 баллом.

Во втором задании школьникам было необходимо установить соответствие между названием блока и его изображением. Например, блок 1 — «Блок рулевого управления», блок 2 — «Ожидание, датчик цвета», блок 3 — «Независимое управление моторами». Всего было представлено 10 блоков, каждое верно установленное соответствие оценивалось 1 баллом (рис. 2).

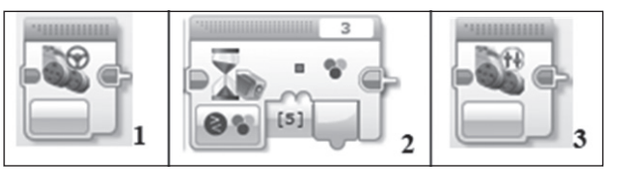


Рис. 2

В третьем задании нужно было описать, что будет делать робот-тележка при выполнении представленной программы (рис. 3). В результате выполнения данного алгоритма робот должен ехать вперед, пока расстояние

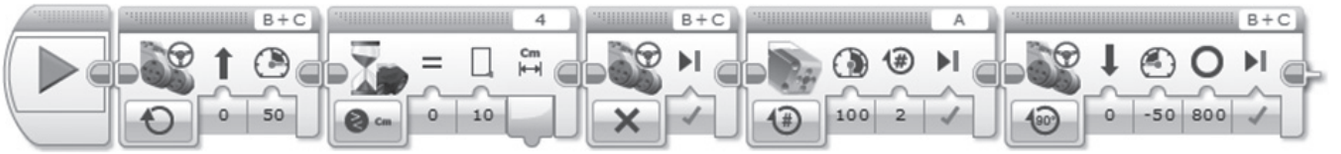


Рис. 3

до объекта не станет равным 10 сантиметрам, затем робот останавливается, средний мотор совершает два оборота, и робот отъезжает назад, прокрутив колеса на 800 градусов вокруг своей оси. Подробное описание оценивалось 10 баллами; описание действий, но без указания их продолжительности — 5–9 баллами; только описание некоторых блоков программы — 1–4 баллами.

Результаты выполнения учащимися теста по LEGO MINDSTORMS Education EV3 представлены в таблице 1.

Таблица 1

Результаты тестирования по LEGO MINDSTORMS Education EV3

Задание	Максимальный балл за задание	Средний балл, который получили учащиеся
1	8	6,71
2	10	8,63
3	10	4,46

В соответствии с результатами работы школьников и результатами тестирования можно сделать следующие **выводы о проведении краткосрочного курса по робототехнике с наборами LEGO MINDSTORMS Education EV3:**

- учащиеся в основном освоили названия всех электронных компонентов набора LEGO MINDSTORMS Education EV3;
- учащиеся запомнили названия большинства программных блоков среды EV3-G;
- в третьем задании теста — при объяснении программы для робота-тележки — учащиеся продемонстрировали низкий уровень усвоения материала.

Полученные результаты проверки знаний и сформированности умений учащихся показали **необходимость коррекции содержания курса:**

- *Во-первых,* в начале курса следует выдавать инструкцию для детей, которая будет содержать в себе основы программирования с использованием графических блоков. Приведем пример из этой инструкции: «Все блоки зеленого цвета — это блоки действия, т. е. робот может запускать моторы, издавать сигналы, менять изображение на экране и мигать разными цветами только с помощью зеленых блоков. Цикл с постусловием (repeat ... until или do ... while) в данной среде программирования легко реализуется с помощью блока ожидания...»
- *Во-вторых,* необходима таблица, в которой описывается принцип работы каждого программного блока.
- *В-третьих,* в процессе обучения следует требовать от учащихся писать для своих программ блок-схемы, т. е. вырабатывать у школьников привычку программирования по созданному ими плану.

Все это позволит обучающимся качественно выполнить исследовательские проекты и повысит уровень осознанности своих действий.

Опишем теперь **курс по направлению «Робототехника» с набором Arduino,** обучение по которому прошли участники лагеря «ТехноЮность».

На занятиях использовались примеры проектов с сайта «Амперка» [4].

На первом занятии учащиеся познакомились с комплектом и создали свою первую программу. При этом перед ними были поставлены следующие задачи:

- узнать строение платы Arduino;
- создать электронную схему на макетной плате с использованием светодиодов и резисторов;
- познакомиться со структурой языка программирования Arduino IDE, в котором создаются программы для робототехнических систем;
- выполнить проекты «Маячок» (подключить к плате светодиод и заставить его мигать с различной периодичностью) и «Маячок с нарастающей яркостью» (светодиод загорается и гаснет пропорционально подаваемому напряжению) [4].

На втором занятии учащиеся выполняли проекты с использованием сервомотора (создавали модель шлагбаума с красным и зеленым светодиодами) и RGB-светодиода (подключали к плате RGB-светодиод и комбинировали его цвета). При выполнении задания они изучили устройство сервопривода и RGB-светодиода, узнали, как подключить библиотеку для работы с сервоприводом, научились использовать переменные в языке Arduino IDE.

На третьем занятии учащиеся выполняли проект «Мерзкое пианино», подключив к плате три кнопки и запрограммировав их на издание звуков с помощью пьезодинамика [4]. Благодаря данной работе учащиеся изучили устройство кнопки и научились подключать ее к плате Arduino; выяснили, как в среде Arduino IDE реализуется цикл с параметром; узнали, как в среде Arduino IDE используются логические переменные.

После проведения всех занятий учащимся также был предложен ***тест*** продолжительностью 20 минут.

В первом задании теста учащимся необходимо было подписать каждый электронный компонент набора, с которым они работали на занятиях. В задании были представлены восемь электронных компонентов, каждый правильно подписанный элемент оценивался 1 баллом. Например: 1 — резистор, 2 — кнопка, 3 — пьезодинамик (рис. 4).

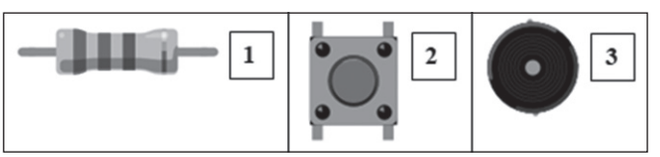


Рис. 4

Во втором задании детям необходимо было установить соответствие между командой и действием. Каждое верное соответствие оценивалось в 1 балл. Например, действие «На пин 9 “низкий сигнал”» — это шестая команда.

Задание 2 из теста по Arduino.

Установите соответствие между командой и действием (запишите номер команды, соответствующей действию).

1) pinMode(13, OUTPUT);
2) void loop()
3) delay(200);
4) digitalWrite (9, HIGH);
5) analogWrite(10, 255);
6) digitalWrite (9, LOW);
7) pinMode(13, INPUT);
8) analogWrite(1, 20);
9) delay(2000);
10) void setup()

На пин 9 «низкий сигнал» ____
Задержка в 0,2 секунды ____
На пин 1 подаем 20/255 от 5В ____
Пин 13 в режим выхода ____
Процедура для настройки портов ____
На пин 9 «высокий сигнал» ____
Пин 13 в режим входа ____
На пин 10 подаем 255/255 от 5В ____
Процедура бесконечного цикла ____
Задержка в 2 секунды ____

В третьем задании нужно было описать, что будет делать схема по заданной программе, в которой использовался один светодиод:

```
void setup()
{
  pinMode(13, OUTPUT);
}
void loop()
{
  digitalWrite(13, HIGH); delay(100);
  digitalWrite(13, LOW); delay(900);
}
```

В данной программе пин 13 настраивался в режим выхода, далее светодиод в бесконечном цикле 100 миллисекунд светил и 900 миллисекунд не светил.

Подробное описание оценивалось 10 баллами; описание действий, но без указания их продолжительности — 5–9 баллами; только описание некоторых строк программы — 1–4 баллами.

Результаты тестирования по работе с Arduino представлены в таблице 3.

В соответствии с результатами практической деятельности детей и результатами теста можно сделать следующие **выводы о проведении краткосрочного курса по робототехнике с наборами на базе платы Arduino:**

Таблица 3

Результаты тестирования по Arduino

Задание	Максимальный балл за задание	Средний балл, который получили учащиеся
1	8	6,72
2	10	9,30
3	10	7,63

- учащиеся в основном освоили названия всех электронных компонентов набора;
- учащиеся запомнили названия и назначения большинства команд, с которыми они работали во время курса;
- учащиеся смогли интерпретировать код программы и объяснить его назначение.

Таким образом, можно считать, что проведенные занятия по LEGO MINDSTORMS Education EV3 и Arduino с учащимися лагеря «ТехноЮность» продемонстрировали свою эффективность и соответствие требованиям, которые предъявляются к краткому курсу по основам робототехники в условиях использования мобильной лаборатории. Оба курса можно применять при использовании мобильной лаборатории в рамках сетевого образовательного взаимодействия. Проведенные исследования также доказали возможность ускоренного овладения учащимися базовыми умениями в области программирования робототехнических систем, что является фундаментом для дальнейшей творческой деятельности школьников в этом направлении. При таком подходе к организации творческой деятельности школьников развитие умения программировать превращается в инструмент для решения технических задач, становится практико-ориентированным, что существенно повышает мотивацию школьников к дальнейшему совершенствованию этого умения.

Список использованных источников

1. *Копосов Д. Г.* Первый шаг в робототехнику: практикум для 5–6 классов. 2-е изд. М.: БИНОМ. Лаборатория знаний, 2015.

2. *Овсяницкая Л. Ю., Овсяницкий Л. Ю., Овсяницкий А. Д.* Курс программирования LEGO MINDSTORMS EV3 в среде EV3: основные подходы, практические примеры, секреты мастерства. 2-е изд., перераб. и доп. М.: Перо, 2016.

3. *Овсяницкая Л. Ю., Овсяницкий Л. Ю., Овсяницкий А. Д.* Пропорциональное управление роботом LEGO MINDSTORMS EV3. М.: Перо, 2015.

4. Примеры мини-проектов с Arduino // Амперка. http://wiki.amperka.ru/

5. *Филиппов С. А.* Робототехника для детей и родителей. СПб.: Наука, 2013.

Р. И. Баженов, Ю. П. Штепа,
Приамурский государственный университет имени Шолом-Алейхема, г. Биробиджан, Еврейская автономная область,
Н. В. Шевченко,
средняя общеобразовательная школа № 3 имени А. И. Томилина, г. Советская Гавань, Хабаровский край

ОРГАНИЗАЦИЯ ПРОЕКТНО-ИССЛЕДОВАТЕЛЬСКОЙ ДЕЯТЕЛЬНОСТИ ШКОЛЬНИКОВ СРЕДСТВАМИ ОБРАЗОВАТЕЛЬНОЙ РОБОТОТЕХНИКИ

Аннотация

Федеральные государственные образовательные стандарты общего образования устанавливают требования к результатам обучающихся, одно из которых — владение навыками учебно-исследовательской и проектной деятельности. Робототехника является эффективным инструментом вовлечения детей в научно-техническое творчество. В статье приводится описание содержательных и процессуальных аспектов курса образовательной робототехники как ресурса организации проектно-исследовательской деятельности школьников.

Ключевые слова: информатика, проектно-исследовательская деятельность, образовательная робототехника, LEGO-конструирование, объектно-ориентированное программирование.

Контактная информация

Баженов Руслан Иванович, канд. пед. наук, доцент, зав. кафедрой информационных систем, математики и методик обучения Приамурского государственного университета имени Шолом-Алейхема, г. Биробиджан, Еврейская автономная область; *адрес:* 679015, Еврейская автономная область, г. Биробиджан, ул. Широкая, д. 70а; *телефон:* (42622) 4-67-32; *e-mail:* r-i-bazhenov@yandex.ru

Штепа Юлия Петровна, канд. пед. наук, доцент, доцент кафедры информационных систем, математики и методик обучения Приамурского государственного университета имени Шолом-Алейхема, г. Биробиджан, Еврейская автономная область; *адрес:* 679015, Еврейская автономная область, г. Биробиджан, ул. Широкая, д. 70а; *телефон:* (42622) 4-67-32; *e-mail:* shtepa2001@mail.ru

Шевченко Надежда Валентиновна, учитель информатики средней общеобразовательной школы № 3 имени А. И. Томилина, г. Советская Гавань, Хабаровский край; *адрес:* 682800, Хабаровский край, г. Советская Гавань, ул. Киевская, д. 2; *телефон:* (42138) 4-43-55; *e-mail:* nadya7999@mail.ru

R. I. Bazhenov, Yu. P. Shtepa,
Sholom-Aleichem Priamursky State University, Birobidzhan, Jewish Autonomous Region,
N. V. Shevchenko,
School 3 named after A. I. Tomilin, Sovetskaya Gavan, Khabarovsk Territory

ORGANIZATION OF PROJECT AND RESEARCH ACTIVITY OF SCHOOLCHILDREN BY MEANS OF EDUCATIONAL ROBOTICS

Abstract

The Federal State Educational Standards establish requirements to results of the students, one of which is possession of skills of educational and research and project activity. Robotics is the efficient instrument of involvement of children in scientific and technical creativity. The description of substantial and procedural aspects of a course of educational robotics as resource of the organization of project and research activity of school students is provided in the article.

Keywords: informatics, project and research activity, educational robotics, LEGO-constructioning, object-oriented programming.

Человечество уже многие десятилетия использует роботов в различных сферах своей деятельности. Несмотря на такие трудности внедрения роботов, как сложность и дороговизна, сферы их применения постоянно расширяются [3]. Очевидной становится необходимость популяризации профессии инженера с раннего возраста.

Одним из средств, позволяющих вовлечь детей в процесс инженерного творчества, является образовательная робототехника. Исследователи Х. Х. Абушкин, А. В. Дадонова, Е. Н. Голобородько и др. отмечают роль образовательной робототехники как ресурса формирования ключевых компетенций обучающихся на разных ступенях образования [1, 2].

Оптимальным способом организации проектно-исследовательской деятельности являются внеурочные занятия по робототехнике. Они позволяют сделать проект действительно творческим и самостоятельным, не ограничивает идею рамками одного конкретного предмета. Основным в организации деятельности работы кружка по робототехнике является постепенное усложнение занятий от технического моделирования до сборки и программирования роботов с теоретическим обоснованием работоспособности и значимости проекта [4].

Анализ существующих рабочих программ по робототехнике показал, что большинство из них ориентированы на сборку и программирование простых конструкций и решение стандартных вопросов, возникающих при подготовке к соревнованиям. Предлагаемая **авторская программа курса по робототехнике** позволяет организовать проектно-исследовательскую деятельность школьников средствами образовательной робототехники.

Цель курса — научить учащихся использовать средства информационных технологий для решения конструкторских задач и сформировать навыки проектно-исследовательской деятельности.

Программа для учащихся V—VI классов рассчитана на два года обучения. В первый год обучения учащиеся знакомятся с базовыми механизмами разработки и программирования рабочих проектов, учатся самостоятельно проводить небольшие исследования с использованием робототехнических датчиков. На втором году обучения учащиеся смогут самостоятельно разрабатывать авторский проектно-исследовательский продукт, исходя из самостоятельно выбранной темы исследования.

Пример учебно-тематического плана работы кружка по робототехнике для учащихся первого года обучения представлен в таблице 1.

Содержание обучения — первый год.

Раздел 1. Знакомство со стартовым набором LEGO MINDSTORMS EV3 и программным обеспечением.

Введение в курс робототехники. Конструкторы компании LEGO. Возможности LEGO MINDSTORMS EV3.

Таблица 1

Учебно-тематический план первого года обучения

№ п/п	Тема	Общее количество часов	Теория	Практика
1	Знакомство со стартовым набором LEGO MINDSTORMS EV3 и программным обеспечением	2	1	1
2	Сборка первого робота. Исследование работы датчиков и возможностей главного модуля	8	2	6
3	Сборка и программирование роботов с использованием базового набора 45544 и программного обеспечения LEGO MINDSTORMS EV3 EDU	15	3	12
4	Решение нестандартных задач, творческие проекты	10	2	8
	ИТОГО:	35	8	27

Раздел 2. Сборка первого робота. Исследование работы датчиков и возможностей главного модуля.

Знакомство с набором LEGO MINDSTORMS EV3 сборки 45544. Сборка робота линейный ползун. Большие моторы. Подключение к главному модулю периферийных частей. Запуск модуля, тестирование работоспособности в режиме demo. Знакомство с блоками программирования модуля EV3. Составление простых программ на блоке управления EV3.

Раздел 3. Сборка и программирование роботов с использованием базового набора 45544 и программного обеспечения LEGO MINDSTORMS EV3 EDU.

Обозначение темы проекта, постановка цели. Разработка механизма на основе конструктора LEGO EV3. Сборка робота Colorsorter (сортировщика цветов). Гусеничный механизм передачи. Малый мотор. Подключение и проверка работоспособности светового датчика. Сервомоторы. Знакомство со средой программирования LEGO MINDSTORMS EV3. Работа в режиме распознавания цвета. Режим ожидания цвета. Работа с переключателем на основе светового датчика. Отладка программы. Сборка балансирующего робота Gyroboy. Гироскопический датчик. Ультразвуковой датчик. Загрузка отсутствующих модулей.

Раздел 4. Решение нестандартных задач, творческие проекты.

Регламенты соревнований. Теоретическое решение поставленной задачи. Практическое решение поставленной задачи. Генерирование моторов с датчиками робота. Самостоятельное написание программы в соответствии с поставленной задачей. Анализ готовой конструкции. Модификация и устранение неисправностей. Регуляторы, используемые в программировании датчика освещенности. Настройка датчика освещенности и выбор его параметров. Езда по черной линии с использованием параметра переключателя. Подготовка презентации, теоретическое обоснование готовой конструкции. Соревнование или защита проекта (по выбору).

Пример учебно-тематического плана работы кружка по робототехнике для учащихся второго года обучения представлен в таблице 2.

Содержание обучения — второй год.

Раздел 1. Введение в проектную деятельность.

Введение в проектную деятельность. Структура проектно-исследовательской работы. Определение направления проектно-исследовательской работы. Выбор темы исследования. Постановка целей и задач проектно-исследовательской работы. Выдвижение гипотезы исследования.

Раздел 2. Конструирование.

Макет проектно-исследовательской работы. Теоретическое обоснование успешности и значимости проекта. Оценка необходимых материалов для создания проекта. Выбор материально-технической базы. Разбиение проекта на составные этапы. Создание пошаговой конструкции проекта. Тестирование проекта на параметры скорости, грузоподъемности, рентабельности и безопасности. Выявление рисков, возможных проблем при программировании и эксплуатации. Пути решения потенциальных проблем, возможности модификации.

Раздел 3. Программирование.

Теоретическое обоснование потенциальных возможностей проекта с точки зрения программирования. Общее

Таблица 2

Учебно-тематический план второго года обучения

№ п/п	Тема	Общее количество часов	Теория	Практика
1	Введение в проектную деятельность	5	2	3
2	Конструирование	10	2	8
3	Программирование	10	3	7
4	Проектная деятельность в группах	10	2	8
	ИТОГО:	35	9	26

тестирование работоспособности конструкции. Создание отдельных программных блоков. Написание программы. Отладка программы. Тестирование проекта в условиях повышенной нагрузки. Корректировка датчиков. Создание журнала наблюдения работы датчиков в стандартных и экстремальных условиях эксплуатации проекта.

Раздел 4. Проектная деятельность в группах.

Выбор типа защиты проекта (соревнование или выставка). Распределение ролей между всеми участниками команд. Подготовка защиты проекта. Компоновка необходимых ресурсов для защиты проекта. Создание баннера, пояснительной записки. Создание технического паспорта проекта. Итоговое тестирование проекта, исключение рассинхронизации датчиков. Подготовка проекта на случай аварийной эксплуатации. Защита проекта.

Занятия в кружковой работе можно строить в зависимости от наполняемости класса и характера выбранного проекта. Также форма занятия зависит от навыков обучающихся и определенного этапа проектно-исследовательской работы.

Формы занятий могут быть следующие:

- Индивидуальная. Используется при формировании базовых компетенций, в которых важна теоретическая проработка материала каждым обучающимся.
- Групповая — парная. Используется при формировании командных навыков в ходе проектно-исследовательской работы. Чаще всего преобладает именно парная форма работы, поскольку при создании проекта обычно один обучающийся в паре выступает в роли техника, второй — в роли программиста.
- Групповая — всем составом. Используется при проведении тренингов, а также на базовых общих лекциях.

Формы проведения занятия варьируются, в рамках одного занятия сочетаются разные виды деятельности: беседы, тренинги, упражнения, занятия по отрабатыванию публичной речи, специальные упражнения, репетиции.

Формирование навыков проектно-исследовательской деятельности происходит последовательно, плавно переходя от одного уровня к другому.

На первом уровне педагог сам ставит проблему и намечает пути ее решения.

На втором уровне педагог ставит проблему, но пути ее решения и само решение обучающимся предстоит найти самостоятельно.

Третий уровень — самый сложный — когда ученики сами ставят проблему и ищут пути ее решения.

Первый и второй уровни проектно-исследовательской деятельности формируются у обучающихся на первом году обучения. На втором году обучения учащиеся переходят к третьему уровню — результатом их работы является готовый авторский проект.

Вне зависимости от того, на каком уровне находятся обучающиеся, **структура проектирования состоит из следующих этапов:**

1. Обозначение темы проекта.
2. Постановка цели и задач проекта.
3. Выдвижение гипотезы исследования.
4. Разработка механизма на основе конструктора LEGO.
5. Составление программы для работы механизма в среде LEGO MINDSTORMS.
6. Тестирование модели, устранение дефектов и неисправностей.
7. Аналитическое описание принципа работы механизма.
8. Защита проекта.

Конечным результатом проектно-исследовательской деятельности школьника является проект, который он представляет на выставке, соревновании или на заседании научного общества. По итогам защиты проектов можно судить о степени проработки материала учащимися и результативности программы кружка.

Список использованных источников

1. *Абушкин Х. Х., Дадонова А. В.* Межпредметные связи в робототехнике как средство формирования ключевых компетенций учащихся // Учебный эксперимент в образовании. 2014. № 3 (71).

2. *Голобородько Е. Н.* Робототехника как ресурс формирования ключевых компетенций обучающихся // Педагогическое образование на Алтае. 2013. № 1.

3. *Семикин С. А.* Робототехника как одно из приоритетных направлений в науке, технике и образовании // Вестник Московского гуманитарно-экономического института. 2015. № 2.

4. *Штепа Ю. П., Шевченко Н. В.* Организация пропедевтической работы по информатике средствами образовательной робототехники // Педагогическая информатика. 2015. № 4.

НОВОСТИ

Продажи персональных компьютерных устройств продолжат снижаться

По прогнозам аналитиков IDC, продажи ПК (настольных, ноутбуков и рабочих станций) и планшетов (обычных и с клавиатурами) будут снижаться по крайней мере до 2021 года. С 435,1 млн в 2016 году продажи устройств этого класса снизятся до 398,3 млн в 2021-м. Впрочем, в некоторых сегментах рынка аналитики предсказывают рост. Продажи ПК в корпоративном сегменте в 2017 году стабилизируются, а с 2019-го начнется их рост. И корпоративный сектор, и особенно потребители по-прежнему стремятся продлить срок службы старой техники. Продажи могут вырасти под влиянием экономических факторов, роста популярности игр, виртуальной реальности и Windows 10, но даже в лучшем случае рост будет незначительным, полагают аналитики.

(По материалам международного компьютерного еженедельника «Computerworld Россия»)

Т. Е. Скорнякова,
Петровская средняя общеобразовательная школа, с. Петровское, Наро-Фоминский район, Московская область

ПРАКТИЧЕСКИЕ РАБОТЫ ПО СОЗДАНИЮ АНИМАЦИИ В LAZARUS*

Аннотация

В статье представлены примеры практических работ в среде программирования Lazarus, которые позволяют учащимся изучить как основы алгоритмизации и программирования, так и основы создания анимации.

Ключевые слова: Lazarus, ветвление, анимация, интерактивное программирование.

Контактная информация

Скорнякова Татьяна Евгеньевна, учитель математики и информатики Петровской средней общеобразовательной школы, с. Петровское, Наро-Фоминский район, Московская область; *адрес:* 143395, Московская область, Наро-Фоминский район, с. Петровское; *телефон:* (496) 342-32-14; *e-mail:* skorni67@mail.ru

T. E. Skornyakova,
Petrovskaya School, Moscow Region

PRACTICAL WORKS FOR CREATING ANIMATION IN LAZARUS

Abstract

The article presents examples of practical works in the Lazarus programming environment that allow students to learn both the basics of algorithmization and programming and the basics of creating animation.

Keywords: Lazarus, branching, animation, interactive programming.

Использование среды программирования Lazarus позволяет учителю знакомить учащихся как с основами алгоритмизации и программирования, так и с основами создания анимации. В статье представлены практические работы в Lazarus для изучения (повторения) алгоритмической структуры ветвления на примере создания занимательных анимационных роликов.

Практическая работа «Светофор»

1. Запустите среду программирования Lazarus.

2. Выполните команду главного меню *Проект*, *Создать проект*. В появившемся диалоговом окне выберите из списка *Приложение* и нажмите кнопку *Создать*. Результатом этих действий будет появление ***окна формы*** и ***окна редактора программного кода***.

3. Сохраните проект в папке *Светофор*: *Проект*, *Сохранить проект как*.

Задание 1 (линейный алгоритм).

1. Измените название формы. Свойству *Caption* присвойте значение *Светофор* (рис. 1).

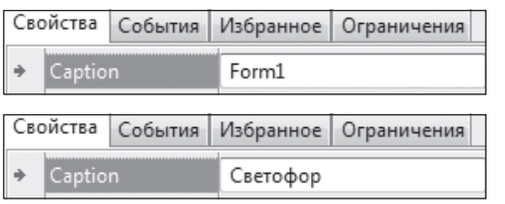


Рис. 1

2. Разместите на форме геометрические фигуры — компоненты *Shape*1 — *Shape*4 (рис. 2).

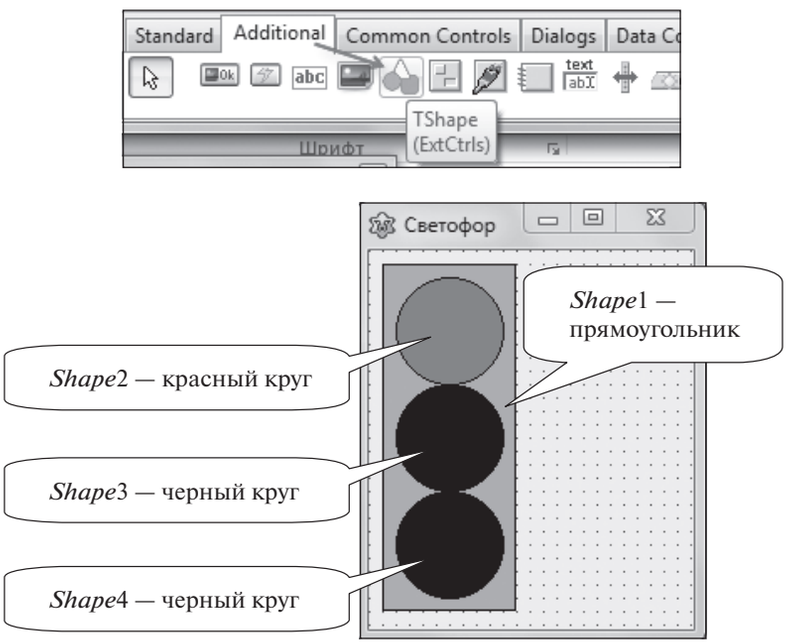


Рис. 2

* Материалы к статье можно скачать на сайте ИНФО: http://infojournal.ru/journals/school/school_10-2017/

Свойства компонента *Shape*1	
Shape	*Brush*
Определяет, какая фигура будет изображена. Варианты: • квадрат, • ромб, • прямоугольник, • круг, • прямоугольник со скругленными углами	Определяет: • *Color* — цвет закрашивания фигуры; • *Style* — стиль заливки

3. Настройте ***управление*** переключением цвета светофора ***по щелчку мыши*** на соответствующем круге.

Для этого выделите объект *Shape*2, в окне *Инспектор объектов* перейдите на вкладку *События* и найдите строку *OnMouseUp* (рис. 3). Введите программный код по изменению цвета кругов в соответствующую процедуру:

```
Shape2.Brush.Color:=clRed;
Shape3.Brush.Color:=clBlack;
Shape4.Brush.Color:=clBlack;
```

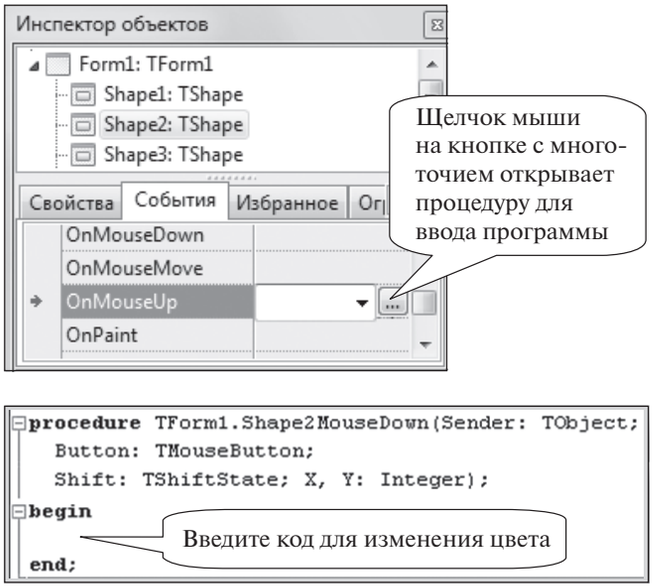


Рис. 3

4. Для объектов *Shape*3 и *Shape*4 аналогичным образом создайте процедуры изменения цвета на желтый и зеленый соответственно.

5. Запустите программу на выполнение (рис. 4).

Рис. 4

Задание 2 (ветвление).

1. Разместите на форме объекты *Shape* различной формы и цвета для создания модели городской улицы.

2. Сконструируйте на дороге машину из объектов *Shape* (рис. 5).

3. Для организации анимации поместите на форму объект *Timer* вкладки *System* (рис. 6).

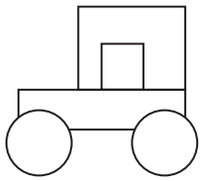


Рис. 5

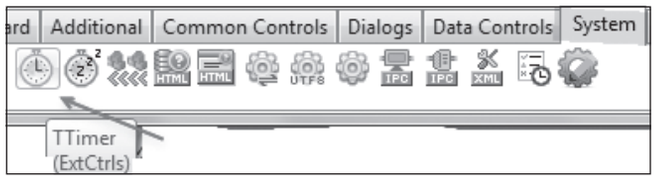


Рис. 6

4. Установите свойства для объекта *Timer*1 в соответствии с таблицей:

Свойства объекта *Timer*1	
Interval	*Enabled*
Определяет интервал в миллисекундах между возникновением событий *OnTimer* (по умолчанию интервал равен одной секунде). Установить: Interval = 10	Определяет, включен или выключен таймер (по умолчанию он включен). Установить: Enabled = false

5. Координаты объекта определяются свойствами *Left* и *Top*, ширина и высота объекта — свойствами *Width* и *Height*. Движение влево задается уменьшением на некоторое число, например на 5, расстояния до левой границы формы при каждом включении таймера:

```
Shape5.Left:= Shape5.Left - 5;
```

Как только объект скроется из вида, запрограммируем его появление справа:

```
if Shape5.Left < 0 then Shape5.Left:=Form1.Width;
```

В окне *Инспектор объектов* для *Timer*1 перейдите на вкладку *События* — выделите строку *OnTimer*. Щелчок мыши на кнопке с многоточием открывает процедуру для ввода двух строк программы, рассмотренных выше.

6. Чтобы при включении зеленого сигнала светофора объект начинал движение, необходимо в процедуру включения зеленого сигнала светофора добавить команду включения таймера (рис. 7).

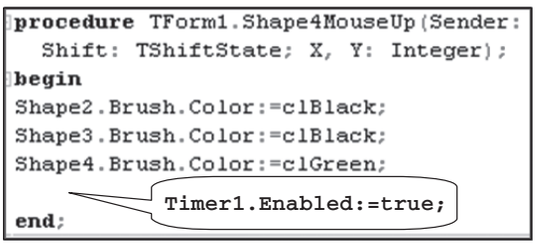


Рис. 7

А в процедуру включения красного сигнала светофора следует добавить команду выключения таймера:

```
Timer1.Enabled:=false;
```

Задание 3*.

1. Сконструируйте на форме пешехода из объектов *Shape*.

2. Отправьте пешехода в путь на разрешающий сигнал светофора.

Практическая работа «Анимация (gif)»

Задание 1.

1. Нарисуйте в Gimp (прозрачный фон) и сохраните как отдельные изображения в папке *Анимация* (формат png) два рисунка, имитирующих движение объекта (рис. 8).

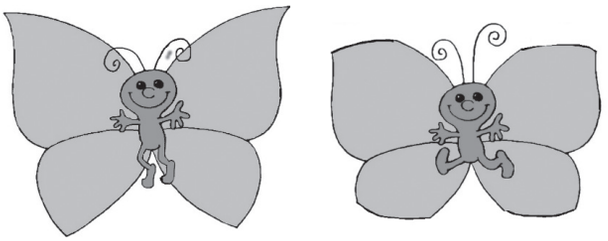


Рис. 8

2. Запустите среду программирования Lazarus.

3. Выполните команду главного меню *Проект, Создать проект*.

4. Сохраните проект в папке *Анимация*: *Проект*, *Сохранить проект как*.

5. Разместите на форме три компонента *Image* (изображение).

6. Установите свойство *Stretch = True* для всех трех компонентов, чтобы сжать большое изображение до размеров компонента *Image*.

7. Для компонентов *Image*1 и *Image*2 установите свойство *Visible = False*, чтобы сделать эти компоненты невидимыми во время выполнения программы.

8. Выделите компонент *Image*1.

Для свойства *Picture* нажмите кнопку с многоточием и в открывшемся окне *Диалог загрузки изображения* (рис. 9) нажмите кнопку *Загрузить*.

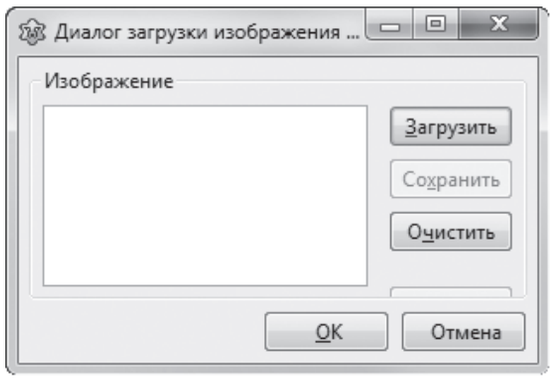


Рис. 9

Перейдите в папку с сохраненными изображениями (см. п. 1), выберите первый рисунок.

9. То же самое сделайте для компонента *Image*2, только выберите другой рисунок.

10. В разделе описания переменных добавьте переменную flag целого типа (рис. 10). В зависимости от значения этой переменной программа будет делать видимым то одно, то другое изображение исходного объекта на компоненте *Image*3.

```
var
  Form1: TForm1; flag:integer;
```

Рис. 10

11. Управление отображением изображения будет осуществляться компонентом *Timer*1.

Установите для объекта *Timer*1 свойство *Interval* равным 300. В процедуру для события *OnTimer* введите следующий программный код:

```
If flag=0 then
  begin
    Image3.Picture:=Image1.Picture;
    flag:=1;
  end
else
  begin
    Image3.Picture:=Image2.Picture;
    flag:=0;
  end;
```

12. Запустите выполнение программы.

Задание 2.

1. Разместите на форме компонент *Timer*2. Запрограммируйте изменение положения анимированного объекта *Image*3. Установите для объекта *Timer*2 свойство *Interval* равным 100.

2. Для объекта *Image*3 на вкладке *События* окна *Инспектор объектов* определите значения свойств *Left* (например, 100) и *Top* (например, 50). Задайте отклонение координат объекта случайным образом от исходного. Для этого в процедуру *OnTimer* для компонента *Timer*2 введите:

```
Image3.Left:=100+random(100);
Image3.Top:=50+random(100);
```

3. Запустите выполнение программы.

Задание 3.

Для управления направлением движения можно использовать клавиши клавиатуры. Так, например, клавиши управления курсором имеют коды:

- влево — 37,
- вверх — 38,
- вправо — 39,
- вниз — 40.

Коды клавиш не изменяются при изменении языка раскладки.

1. Разместите на форме объект *Image*4 (см. п. 5–8 задания 1).

2. Для формы *Form*1 установите свойство формы *KeyPreview = True*.

3. Откройте процедуру события *OnKeyDown*, введите программный код:

```
If (Key=38) and (Image4.Top>=0)
  then Image4.Top:=Image4.Top-5;
If (Key=40) and (Image4.Top<=Form1.Height-
Image4.Height)
  then Image4.Top:=Image4.Top+5;
```

При запуске программы объект будет перемещаться вверх-вниз при нажатии на соответствующие стрелки.

4. Организуйте движение объекта влево-вправо.

5. Запустите выполнение программы.

Список использованных источников

1. *Бешенков С. А., Ракитина Е. А., Миндзаева Э. В.* Курс информатики в современной школе: от компьютерной грамотности к метапредметным результатам // Муниципальное образование: инновации и эксперимент. 2010. № 1.

2. *Власенко В. А.* Познавательная мотивация учащихся в информационной среде учебного проекта по информатике // Вестник Московского городского педагогического университета. Серия «Информатика и информатизация образования». 2013. № 1 (25).

3. *Зиятдинова Т. Л.* Современные технологии в преподавании информатики // Эксперимент и инновации в школе. 2011. № 2.

4. *Каган Э. М.* Обучение программированию как подход к развитию логического, абстрактного и вычислительного мышления у школьников // Вестник Российского университета дружбы народов. Серия «Информатизация образования». 2017. Т. 14. № 4.

5. *Ронжина Т. А.* От идеи к результату: проекты в среде Lazarus // Информатика в школе. 2016. № 3.

КОНКУРСЫ

В оформлении обложки данного номера журнала «Информатика в школе» использованы работы следующих авторов — победителей конкурса цифровых изображений и фотографий:

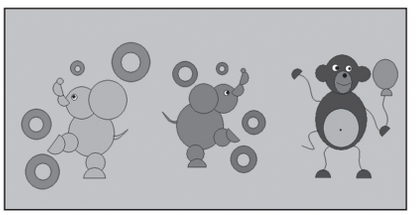
Дарья Неваленая,
воспитанница ГОБУ ДО «Дворец творчества — Мемориал», ученица КОГОБУШ ОВЗ № 50, г. Киров

И. А. Ковальчук,
педагог дополнительного образования Дома детского творчества Спасского муниципального района Республики Татарстан

Екатерина Меняйлова,
ученица средней общеобразовательной школы № 8, г. Ноябрьск, Ямало-Ненецкий автономный округ

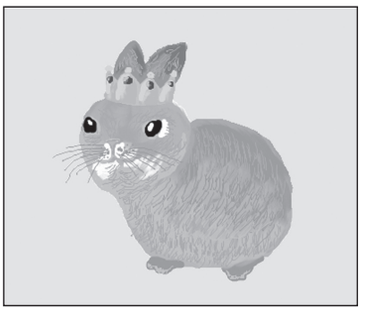
Эвелина Бокан,
ученица средней общеобразовательной школы № 8, г. Ноябрьск, Ямало-Ненецкий автономный округ

Сергей Шамеев,
студент Мамадышского политехнического колледжа, Республика Татарстан



Вячеслав Борисов,
ученик средней общеобразовательной школы № 1, г. Зеленокумск, Ставропольский край

Т. Е. Сорокина,
учитель информатики лицея № 1547, г. Москва



Арина Безбородова,
ученица гимназии № 1, г. Новосибирск

Виктор Шароварников,
студент Барнаульского государственного педагогического колледжа

М. А. Плаксин,
Национальный исследовательский университет «Высшая школа экономики» (Пермский филиал);
Пермский государственный национальный исследовательский университет

ПРОПЕДЕВТИКА ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ В ШКОЛЬНОЙ ИНФОРМАТИКЕ. ПАРАЛЛЕЛЬНАЯ ФОРМА АЛГОРИТМА В ЗАДАЧАХ НА ОБХОД ГРАФА

Аннотация

Для освоения понятия «параллельная (ярусно-параллельная) форма алгоритма» предлагается использовать задачи на построение/обход «лабиринтов». В качестве примера дается разбор задания из конкурса «ТРИЗформашка-2015».

Ключевые слова: информатика, средняя школа, начальная школа, методика обучения, параллельные алгоритмы, параллельная форма алгоритма, ярусно-параллельная форма, обход графа, параллельные вычисления, пропедевтика, ТРИЗформатика, ТРИЗформашка.

Контактная информация

Плаксин Михаил Александрович, канд. физ.-мат. наук;

доцент кафедры информационных технологий в бизнесе Пермского филиала Национального исследовательского университета «Высшая школа экономики»; *адрес:* 614070, г. Пермь, ул. Студенческая, д. 38; *телефон:* (342) 205-52-50;

доцент кафедры математического обеспечения вычислительных систем Пермского государственного национального исследовательского университета; *адрес:* 614990, г. Пермь, ул. Букирева, д. 15; *телефон:* (342) 239-67-72;

e-mail: mapl@list.ru

M. A. Plaksin,

National Research University Higher School of Economics, Perm Branch;

Perm State National Research University

PROPAEDEUTICS OF PARALLEL COMPUTING IN SCHOOL INFORMATICS. PARALLEL FORM OF THE ALGORITHM IN GRAPH TRAVERSAL TASKS

Abstract

It is proposed to use the tasks of building/bypass "labyrinths" to learn the concept of "parallel form of the algorithm (multilevel structure)". As an example the investigation of task from the contests "TRIZformashka-2015" is given.

Keywords: informatics, secondary school, primary school, teaching methods, parallel algorithms, parallel form of algorithm, multilevel structure, graph traversal, parallel computing, propaedeutics, TRIZformatics, TRIZformashka.

Современный этап развития computer science связан с массовым распространением параллелизма вычислений на всех уровнях (многоядерные процессоры, многомашинные кластеры, многопроцессорные ЭВМ). Это делает актуальным включение пропедевтики параллельного программирования в школьный курс информатики.

В рамках работ над «пермской версией» пропедевтического курса информатики (авторский коллектив: М. А. Плаксин, Н. Г. Иванова, О. Л. Русакова; рабочее название курса — «ТРИЗформатика») [7, 8] разработка данной тематики начата в 2013 году. Эффективной площадкой для этого выступил конкурс «ТРИЗформашка» — ежегодный межрегиональный интернет-конкурс по информатике, системному анализу и ТРИЗ (теории решения изобретательских задач) для школьников и студентов [3–5, 9]. В марте 2018 года конкурс состоится в 18-й раз. Возраст участников — от первого класса школы до четвертого курса вуза. Среднее количество команд — около 100 (рекордное — 202). География конкурса — от Владивостока до Риги. Сайт конкурса: http://www.trizformashka.ru

В процессе подготовки конкурса «ТРИЗформашка» были определены типы задач, направленных на освоение базовых понятий параллельных вычислений, и придумано по несколько задач этих типов. Одним из таких типов являются задачи на параллельную (ярусно-параллельную) форму алгоритма. Понятие ярусно-параллельной формы (ЯПФ) описано в [1, 2, 6]. В качестве иллюстрации в [6] рассматривается задание из конкурса «ТРИЗформашка-2014», в котором ярусно-параллельная форма вводится в задаче о строительстве замка. В конкурсах 2015, 2016 и 2017 годов в задании на ЯПФ были использованы задачи на прохождение лабиринта (с точки зрения математики это задачи на обход графа). Такие задачи будут иметь некоторые важные отличия от «строительных» задач. Анализу этих отличий посвящена данная статья. В качестве иллюстрации рассматривается задача из конкурса «ТРИЗформашка-2015».

Ходы кривые роет подземный умный крот… (задача из конкурса «ТРИЗформащка-2015»)

Кроты копают подземный ход, начиная с точки A (рис. 1).

Подземный ход — узкий. Поэтому два крота не могут находиться рядом.

За один день один крот может прокопать расстояние от одного угла клеточки до другого (по горизонтали, вертикали или диагонали).

1. Сколько времени потребуется одному кроту, чтобы вырыть весь подземный ход? Ответ обосновать.

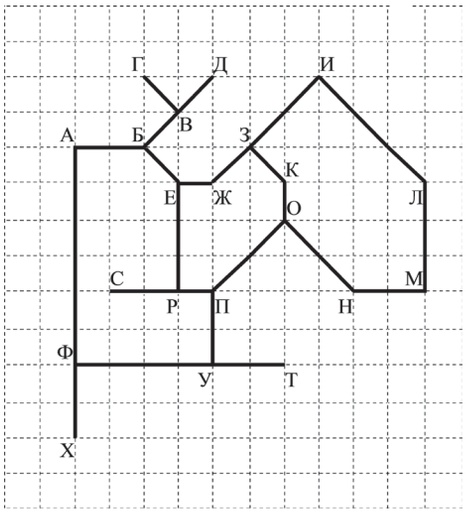


Рис. 1. Схема подземного хода для задачи «Ходы кривые роет подземный умный крот...»

2. За какое минимальное время можно вырыть этот подземный ход, если в рытье может участвовать неограниченное количество кротов? Ответ обосновать.
3. Какое минимальное количество кротов должно участвовать в рытье, чтобы вырыть подземный ход за минимальное время? Ответ обосновать.
4. За какое время этот подземный ход смогут вырыть четыре крота? Ответ обосновать.
5. Сколько кротов надо, чтобы прорыть этот ход за 10 дней? Ответ обосновать.

Перед продолжением чтения читателю рекомендуется самостоятельно решить предложенную задачу.

Некоторые особенности задач на обход графов с точки зрения построения ЯПФ

В [6] понятие ярусно-параллельной формы было проиллюстрировано задачей на построение замка по заданному чертежу (рис. 2). Вершина графа ЯПФ в данном случае соответствует установке одного камня.

Для всех камней в замке однозначно определено, какой камень на какой опирается (или не опирается).

При построении ЯПФ применялось следующее правило:

Каждая вершина попадает на ярус, непосредственно следующий за ярусом, на котором расположены вершины, ей предшествующие. Расположение вершины на ярусе N означает, что последние необходимые для этого вершины были размещены на ярусе (N – 1).

Например, камень 15 будет установлен после камней 16, 17, 19. В ЯПФ строительства замка камень 16 располагается на ярусе 1, камень 19 — на ярусе 3, камень 17 — на ярусе 4. Значит, камень 15 будет расположен на ярусе 5.

ЯПФ, построенная по этому правилу, обладает минимальной высотой и максимальной шириной.

В данной статье рассматривается задача на построение/обход лабиринта. Лабиринт естественным образом трактуется как граф. Точки, обозначенные буквами, трактуются как вершины, коридоры между ними — как дуги. Задача на обход лабиринта представляется задачей на обход графа.

Сравним эту задачу со «строительной» задачей, рассмотренной в [6].

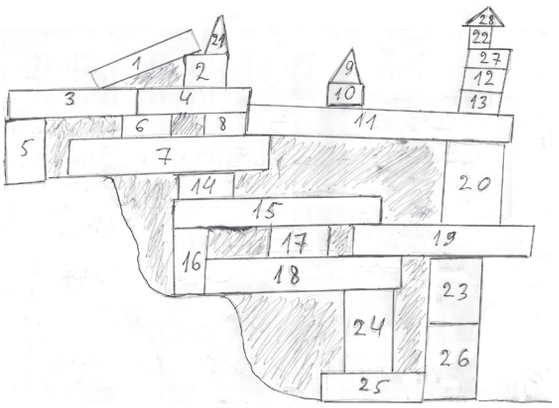


Рис. 2. «Чертеж дворца» для задачи из конкурса «ТРИЗформашка-2014»

Для начала отметим, что аналогом укладки камней из «строительной» задачи в «лабиринтной» задаче будет являться прохождение коридоров лабиринта, т. е. дуг графа. Тем не менее из соображений удобства и наглядности далее мы будем говорить не только о дугах, но и о вершинах.

В «строительной» задаче строительство замка велось строго снизу вверх. Камень, укладываемый сверху, обязательно опирался на камни, уложенные снизу. Для каждого камня было однозначно определено его отношение с каждым другим камнем: предшествует, следует, не связан. Это отношение однозначно определяется статически, по внешнему виду рисунка, без учета динамики построения замка, без учета порядка укладки камней.

В задаче на обход лабиринта это не так. Здесь нет однозначного статически определенного отношения следования! Лабиринт можно пройти в разном порядке. Например, в вершину Б (см. рис. 1) можно прийти из вершины А (по дуге АБ), а можно из вершины Е (пройдя по пути АФУПРЕБ). В вершину О можно попасть из вершины К, можно из вершины П, можно из вершины Н. В зависимости от того, какая вершина окажется предшествующей, поменяются и вершины, являющиеся следующими. Так, для примера с вершиной Б в первом случае последующими окажутся вершины В и Е, во втором — вершины В и А. Для примера с вершиной О в первом случае последующими окажутся вершины П и Н, во втором — вершины К и Н, в третьем — вершины К и П. Лишь для вершин, лежащих на тупиковых ветвях лабиринта, предшествующая вершина определяется однозначно. Так, на рисунке 1 для вершины Х предшествующей является вершина Ф, для вершины Т — вершина У, для вершины С — вершина Р, для вершин Г и Д — вершина В.

Все вышесказанное можно без труда перевести с «языка вершин» на «язык дуг».

Дуга, соединяющая вершины А и Б, может либо не иметь предшествующей (в том случае, если с нее начнется обход графа), либо следовать за дугой ЕБ (после прохождения по одному из путей, начинающихся с АФУП: АФУПРЕБ, АФУПОКЗЖЕБ, АФУПОНМЛИЗЖЕБ). Дуга КО может предшествовать дугам ОП и ОН. Но дуга НО тоже может быть предшествующей — для дуг ОК и ОП, которые будут ей следовать. Наконец, и дуга ПО может оказаться предшествующей, а следовать за ней будут дуги ОК и ОН. (Буквы в названии дуги меняются местами, чтобы указать направление прохода по дуге.)

Для задач обхода графа отношение следования определяется динамически по мере выполнения алгоритма

обхода. По внешнему виду графа установить, в каком порядке будут пройдены дуги, вообще говоря, невозможно! (Очевидные вырожденные случаи нам сейчас не интересны.)

Необходимость строительства «снизу вверх», наличие статически определенного отношения следования делает порядок укладки камней вполне определенным, позволяет достаточно хорошо разграничить уже выполненную часть работы и еще не выполненную: уже уложенные камни расположены снизу, еще не уложенные — сверху. Отсутствие статически определенного отношения следования в задаче обхода графа порождает множество возможных путей обхода. Никаких признаков, по которым можно было бы отделить уже пройденную часть графа от еще не пройденной, здесь нет.

В «строительной» задаче каждый камень опирается на все «предшествующие» камни. Чтобы положить камень, надо сначала уложить все камни, которые ему «непосредственно предшествуют» (на которые он «непосредственно опирается»). В «лабиринтной» задаче для прохождения вершины достаточно подойти к ней по одной из дуг. С прохождением остальных дуг этот факт никак не связан. Если мы переходим в вершину О по дуге ПО, это ровно ничего не говорит о состоянии дуг ОК и ОН. Любая из этих дуг может быть уже пройдена, а может быть нет.

В «строительной» задаче после того, как мы положили некоторый камень, мы должны уложить на него все «непосредственно следующие» камни (т. е. камни, которые непосредственно опираются на данный камень). «Следующие» камни должны быть уложены обязательно все. В «лабиринтной» задаче порядок прохода вершин определяется динамически по мере обхода графа. Поэтому вполне может оказаться, что, когда мы дойдем до некоторой вершины (перейдем в нее по некоторой дуге), часть дуг, исходящих из этой вершины, уже пройдена.

В «строительной» задаче всегда известен камень, имеющий самый высокий уровень. Именно он и будет определять длину критического пути. В «лабиринтной» задаче место завершения обхода заранее не известно. В этой роли могут оказаться многие различные точки.

По этим причинам «лабиринтные» задачи первоначально представляются значительно более сложными, чем «строительные». Попытка решить «лабиринтную» задачу без использования механизма ярусно-параллельных форм вызывает значительные трудности.

Как уже было сказано, при обходе графа ни по внешнему виду, ни по расположению дуги ничего нельзя сказать о том, была она уже пройдена или нет, надо ли ее еще проходить или нет. Это можно только запомнить. А учитывая крайне малый объем сверхоперативной памяти человека (знаменитое «волшебное число 7 плюс/минус 2»), лучше записать. Ярусно-параллельная форма оказывается очень удобной формой фиксации проделанной части работы (пройденной части графа). Поэтому в случае применения ЯПФ практически все вышеназванные трудности снимаются. В частности, в процессе построения ярусно-параллельной формы, ориентированной на минимальное время выполнения алгоритма, мы автоматически определим точку, наиболее отдаленную от входа в лабиринт.

В силу особенностей построения ярусно-параллельной формы для алгоритма обхода лабиринта (у каждой вершины — единственная входная дуга и сколько угодно выходных), ЯПФ для «лабиринтных» задач будет деревом. Более точно: ярусно-параллельная форма будет одним из остовных деревьев исходного графа, т. е. деревьев, которые включают все вершины графа и дуги, минимально необходимые для обеспечения его связности [10].

Разбор задачи «Ходы кривые роет подземный умный крот…»

Вернемся к задаче про мудрого крота.

Для удобства обсуждения перенумеруем строки и столбцы на схеме подземного хода (рис. 3).

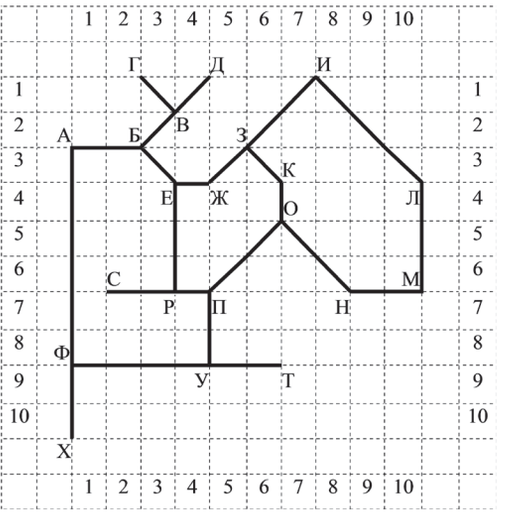


Рис. 3. Схема подземного хода с нумерацией строк и столбцов

Всего одиночных отрезков — 46.

Для ответа на поставленные в задании вопросы необходимо построить ярусно-параллельную форму алгоритма рытья подземного хода. Воспользуемся вышеприведенным правилом из [6] для построения графа минимальной высоты и максимальной ширины.

Граф ЯПФ будет выглядеть так, как представлено на рисунке 4.

В названии вершины графа двумя буквами обозначен отрезок, по которому движется крот. Число, стоящее после букв, — это номер клетки по вертикали или по горизонтали. Оно играет роль для отрезков, имеющих длину в несколько клеток. Ссылка на «номер по вертикали» или «номер по горизонтали» выбрана из соображений удобства без соблюдения каких-либо формальных правил и может быть изменена по желанию читателя.

Заметим, что вершинами графа ЯПФ являются не точки со схемы подземного хода, а «единицы работы», т. е. единичные отрезки, из которых состоит подземный ход.

Высота ЯПФ равна 12, ширина — 8. Это ширина яруса 8.

Напомним, что в ЯПФ, построенной по правилам из [6], высота графа определяет минимальное время выполнения алгоритма, а ширина — максимальную степень распараллеливания.

То есть минимальное время прохождения лабиринта (прокапывания подземного хода) равно 12. Для этого требуется восемь кротов.

Можно ли сжать ЯПФ? Сделать ее более узкой, и тем самым уменьшить количество кротов, необходимых для построения лабиринта?

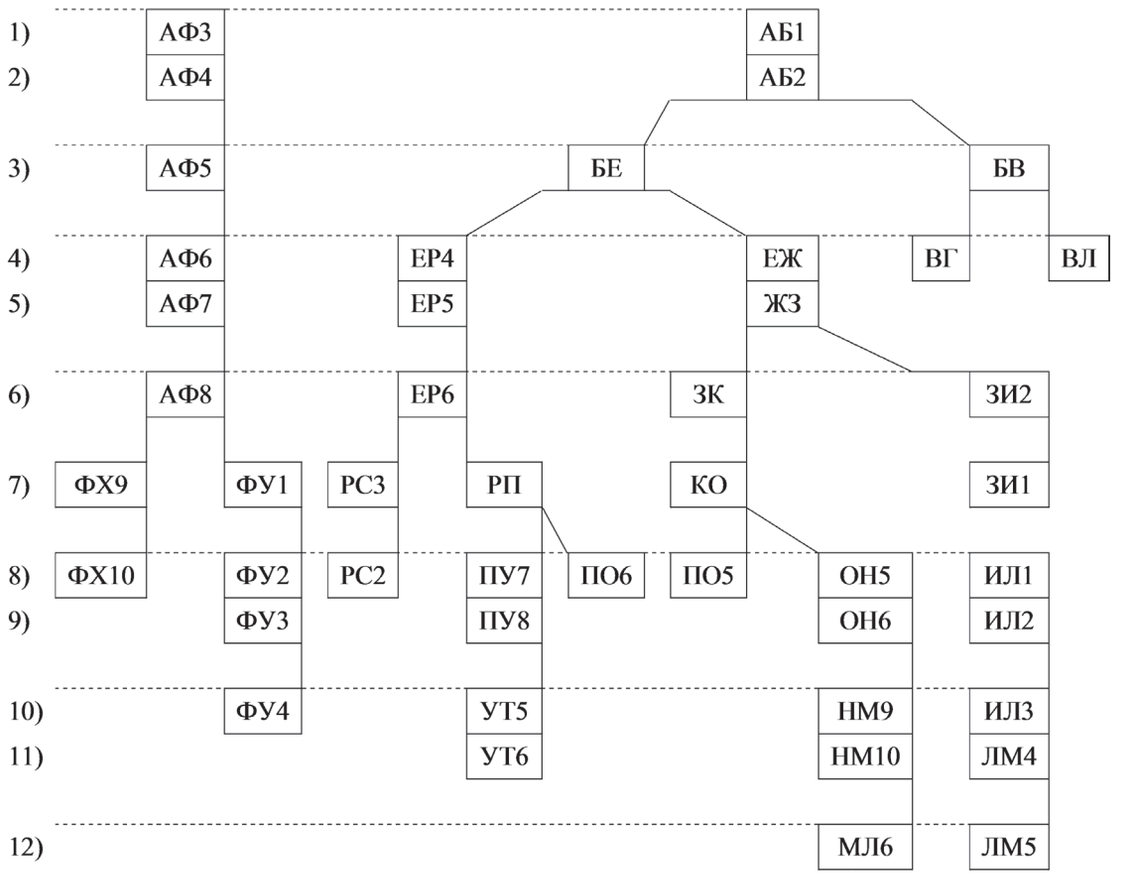


Рис. 4. Ярусно-параллельная форма для задачи про кротов

Граф ЯПФ содержит 46 вершин. При высоте 12 граф может иметь ширину 4 (46 : 12, округленное вверх). Значит, возможность сжатия будет определяться распределением вершин по ярусам. Существует ли возможность спустить вершины с перегруженных ярусов вниз? Да.

Вершину ФХ10 можно спустить на 9-й, или 10-й, или 11-й, или 12-й ярус. Ширина ЯПФ станет равна 7. Это ширина яруса 8. Остальные ярусы у́же. Результат см. на рисунке 5.

Продолжим сужение.

Тройку вершин ФУ2—ФУ3—ФУ4 можно сместить на один или два яруса вниз (на ярусы 9–10–11 или 10–11–12). Ширина ЯПФ станет равна 6. Это ширина ярусов 7 и 8. Остальные ярусы у́же. Результат см. на рисунке 6.

С этих двух ярусов (7-го и 8-го) надо сместить вниз по одной вершине. Это возможно. Ниже находится один ярус шириной 3 и три яруса шириной 4. Например, пару вершин РС3—РС2 сместить вниз на два, три или четыре яруса. Сместим их на ярусы 9–10. Результат см. на рисунке 7.

Теперь ширина ЯПФ стала равна 5. Это ширина ярусов 4, 7, 8, 10.

Дальнейшее сужение возможно только для яруса 4. С него можно спустить вниз вершину ВГ или ВЛ.

Для ярусов 7, 8, 10 сужение невозможно. Спустить с них вершины можно только на ярусы, которые имеют ширину 4. Спуск туда еще одной вершины приведет к увеличению их ширины до 5.

Таким образом, дальнейшее сужение ЯПФ невозможно.

Минимальная ширина ЯПФ равна 5. Следовательно, минимальное количество кротов, которые могут построить такой лабиринт за 12 ходов, равно 5.

Ответы на вопросы задания выглядят так:

1. Сколько времени потребуется одному кроту, чтобы вырыть весь подземный ход? Ответ обосновать.

Ответ. 46. Сложить длину всех отрезков. Одному кроту придется пройти их все по очереди.

2. За какое минимальное время можно вырыть этот подземный ход, если в рытье может участвовать неограниченное количество кротов? Ответ обосновать.

Ответ. 12 (или 13 при невнимательном чтении условий).

Обоснование.

См. ярусно-параллельную форму на рисунке 4.

Задание намеренно сформулировано таким образом, что при невнимательном чтении может спровоцировать ошибку понимания. Из формулировки следует, что два крота не могут находиться рядом. Но не следует, что два крота не могут одновременно рыть один и тот же отрезок с двух сторон. Если эту возможность не учитывать, то критический путь будет равен 13, если учитывать, то 12.

Самая дальняя точка от входа в подземелье — точка на отрезке ЛМ на одну клетку выше точки М.

Длина пути АБЕЖЗИЛМ = 13. Длина пути АБЕЖЗКОНМ = 11.

Остальные пути — меньше или равны:

АФХ = 8,

АФУТ = 12,

АБЕРПУТ = 12,

АБЕРПО = 9,

АБЕЖЗКО = 7.

Через 10 дней один из кротов выйдет в точку Л сверху и двинется вниз.

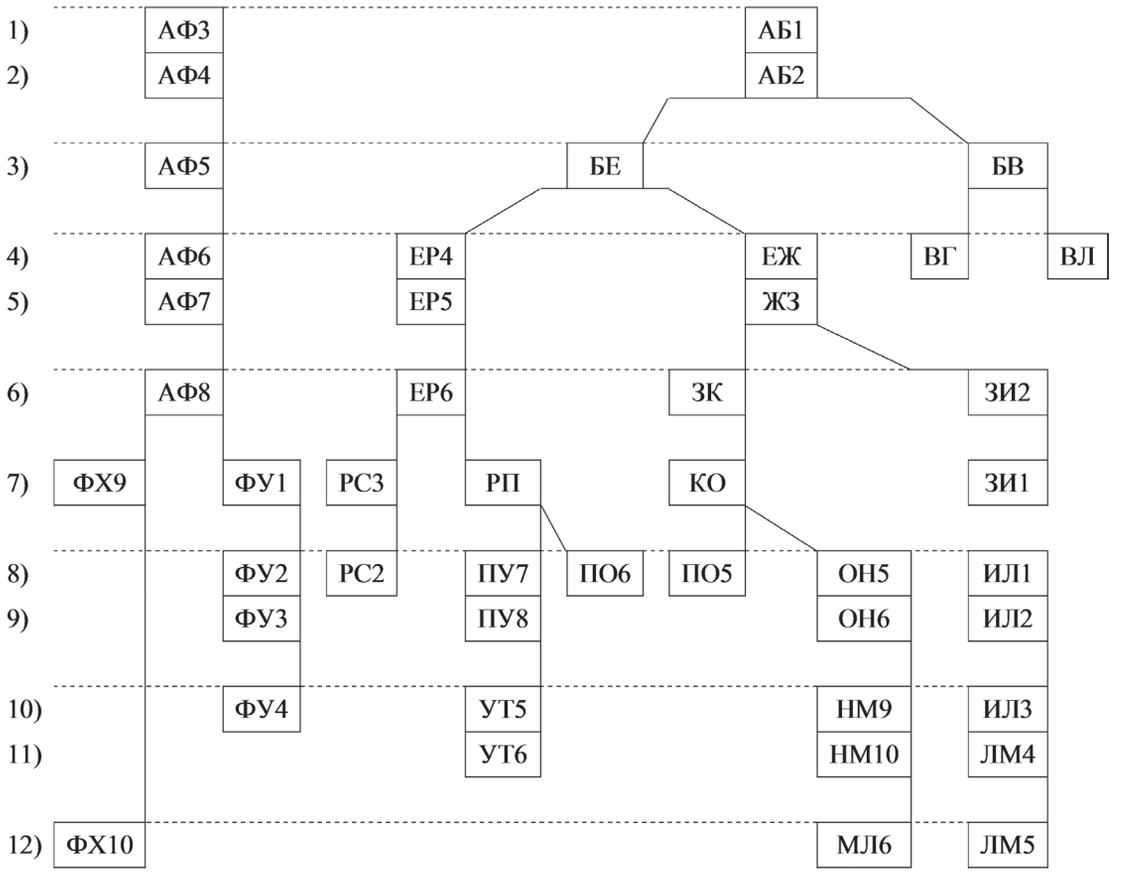


Рис. 5. Ярусно-параллельная форма для задачи про кротов, сжатая до ширины 7

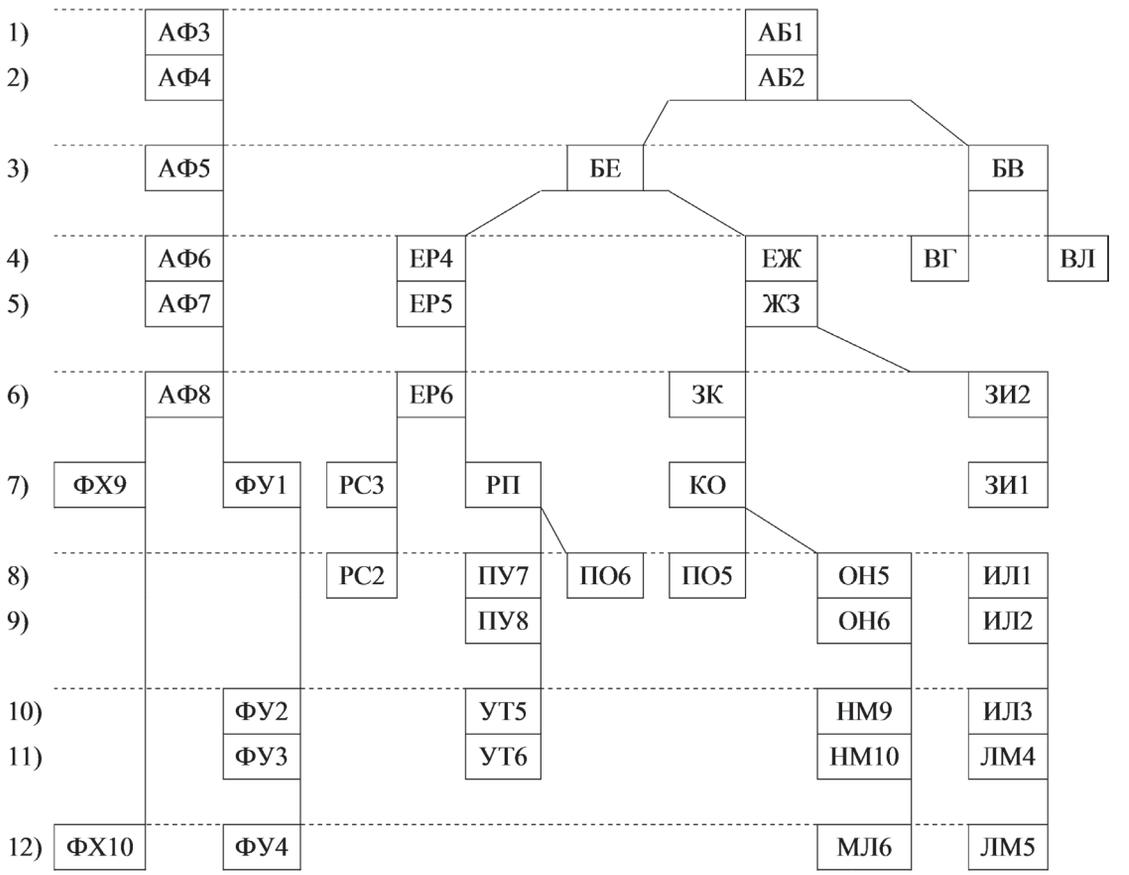


Рис. 6. Ярусно-параллельная форма для задачи про кротов, сжатая до ширины 6

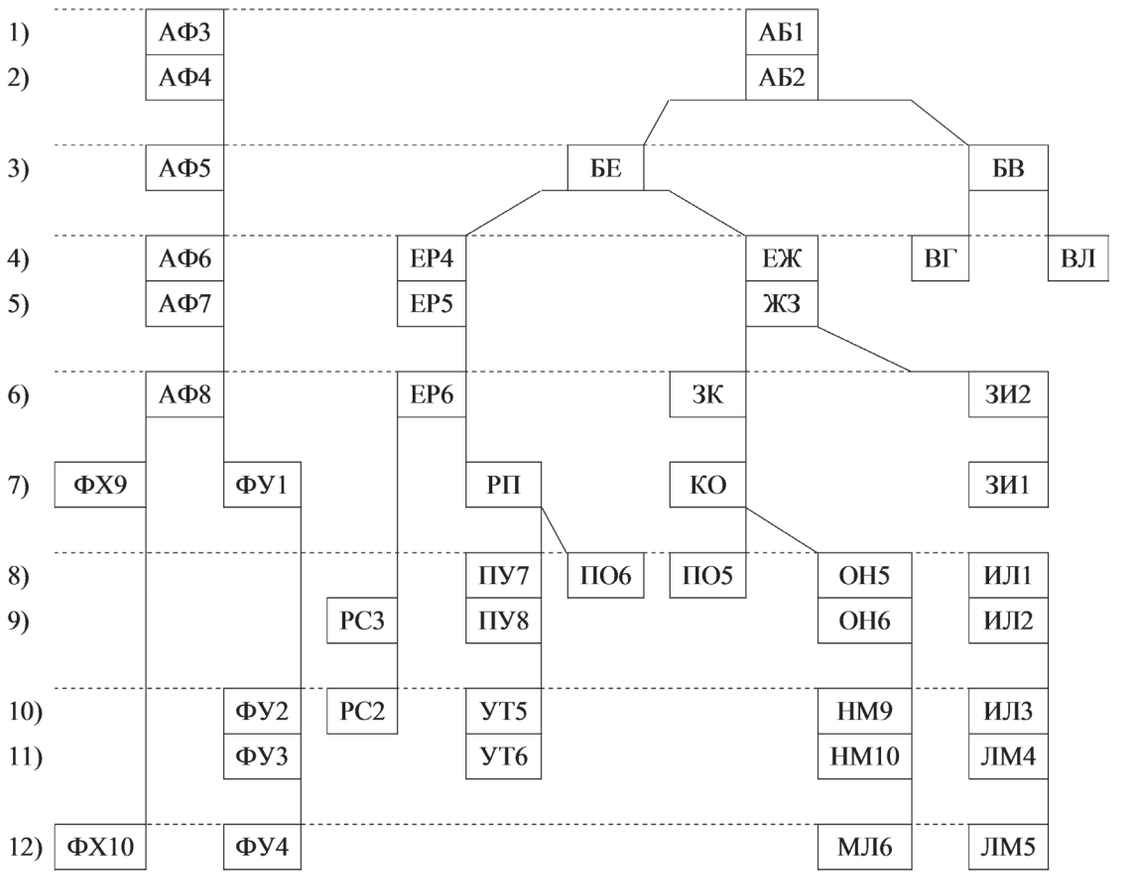


Рис. 7. Ярусно-параллельная форма для задачи про кротов, сжатая до ширины 5

Через 11 дней другой крот выйдет в точку М слева и двинется вверх.

К этому времени верхний крот уже пройдет одну клетку вниз. Встретятся два крота через день в точке, лежащей на одну клетку выше точки М.

Таким образом, длина критического пути равна 12.

Если возможность одновременного копания одного отрезка с двух сторон не учитывать, то критическим будет путь АБЕЖЗИЛМ = 13. При наличии обоснования этот ответ может быть принят, но оценка за него должна быть снижена.

Дополнительная сложность связана с тем, что самая дальняя точка сознательно не обозначена буквой.

3. Какое минимальное количество кротов должно участвовать в рытье, чтобы вырыть подземный ход за минимальное время? Ответ обосновать.

Ответ. Достаточно пяти кротов.

Обоснование. См. ЯПФ на рисунке 7.

4. За какое время этот подземный ход смогут вырыть четыре крота? Ответ обосновать.

Ответ. За 13 дней.

Обоснование. Для ответа на этот вопрос надо ярусно-параллельную форму с рисунка 7 сузить до ширины 4 и определить ее новую высоту. Покажем, как это можно сделать.

Содержательные рассуждения будут выглядеть следующим образом.

Первые два дня будут задействованы по два крота, на третий день — три. На четвертый день будут работать все четыре крота, и при этом образуется «задолженность»: одна вершина ЯПФ будет не отработана. Но на пятый день эта задолженность будет ликвидирована: три крота отработают вершины пятого яруса, а один — непройденную вершину четвертого яруса.

Все последующие ярусы имеют ширину 4 или 5. Так что при их прохождении будут постоянно заняты все четыре крота. При этом при прохождении ярусов, имеющих ширину , будет образовываться «задолженность» в один блок на каждом ярусе. Таких ярусов три: 7-й, 8-й и 10-й. Для ликвидации «задолженности» в три блока четырем кротам понадобится еще один день.

Формальные преобразования графа ЯПФ таковы.

Вершина ВЛ смещена с яруса 4 на ярус 5.

С яруса 7 на ярус 12 вниз смещена вершина ФХ9. Чтобы освободить ей место, вершина ФХ10 была сдвинута вниз на вновь образованный ярус 13.

С яруса 8 на вновь образованный ярус 13 была смещена вершина ПО5, а с яруса 10 — вершина РС2.

Результат сужения — ЯПФ шириной 4 — показан на рисунке 8.

5. Сколько кротов надо, чтобы прорыть этот ход за 10 дней? Ответ обосновать.

Ответ. Нисколько. Это невозможно. Длина критического пути равна 12.

* * *

В настоящей статье мы обсудили особенности задач на обход графа по сравнению со «строительными» задачами (отсутствие статически определенного отношения следования вершин/дуг) и разобрали применение ярусно-параллельных форм для решения таких задач. В данном случае ЯПФ показали себя как очень эффективный меха-

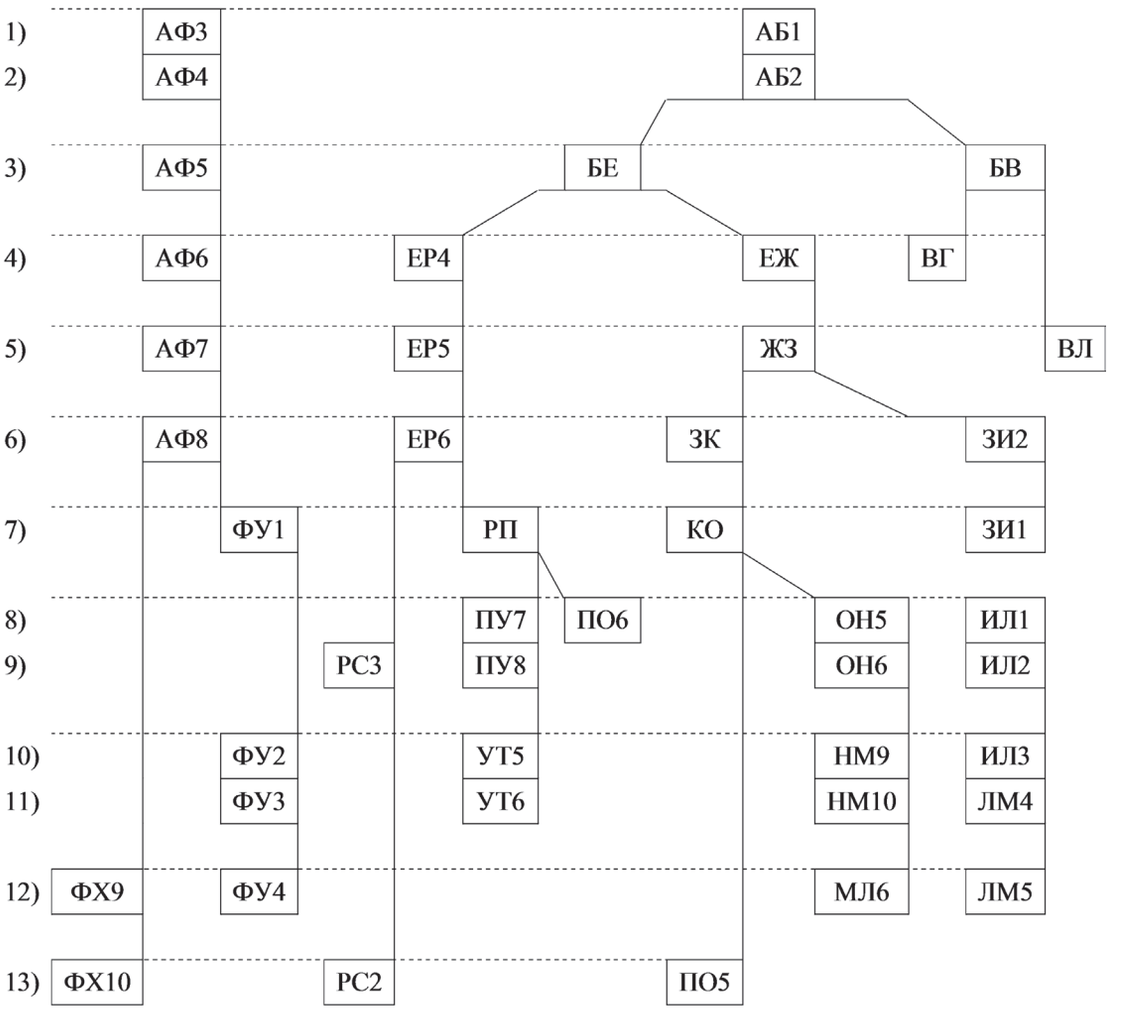


Рис. 8. Ярусно-параллельная форма для задачи про кротов, сжатая до ширины 4

низм. Построение параллельной формы алгоритма автоматически сняло все трудности, которые имелись в задаче.

В конкурсах «ТРИЗформашка-2016» и «ТРИЗформашка-2017» в качестве задач на ЯПФ также использовались «лабиринтные» задачи. Но со своими особенностями-усложнениями. В «ТРИЗформашке-2016» усложнением стало введение в задачу исполнителей с разной производительностью труда и переход от однородного графа к неоднородному (разные точки внутри лабиринта неравноправны). В «ТРИЗформашке-2017» усложнение выразилось в переходе от одноуровневого лабиринта к многоуровневому. Разбор этих задач — тема отдельного разговора.

Список использованных источников

1. *Воеводин В. В.* Вычислительная математика и структура алгоритмов: 10 лекций о том, почему трудно решать задачи на вычислительных системах параллельной архитектуры и что надо знать дополнительно, чтобы успешно преодолевать эти трудности. 2-е изд., стереотип. М.: Изд-во Московского университета, 2010.

2. *Воеводин В. В., Воеводин Вл. В.* Параллельные вычисления. СПб.: БХВ-Петербург, 2002.

3. *Иванова Н. Г., Плаксин М. А., Русакова О. Л.* Задачи на параллельное программирование в конкурсе «ТРИЗформашка-2013» // Информационные технологии в образовании. XXIII Международная конференция-выставка: Сборник трудов. Ч. II. М.: Издательский отдел факультета ВМК МГУ имени М. В. Ломоносова, 2013.

4. *Иванова Н. Г., Плаксин М. А., Русакова О. Л.* Конкурс «ТРИЗформашка» как площадка для апробации заданий на параллельное программирование // Информатика в школе: прошлое, настоящее и будущее. Материалы Всероссийской научно-методической конференции по вопросам применения ИКТ в образовании, 6–7 февраля 2014 г. Пермь: Перм. гос. нац. иссл. ун-т, 2014.

5. *Иванова Н. Г., Плаксин М. А., Русакова О. Л.* ТРИЗформашка // Информатика. 2010. № 5.

6. *Плаксин М. А.* Пропедевтика параллельных вычислений в школьной информатике. Параллельная форма алгоритма // Информатика в школе. 2016. № 10.

7. *Плаксин М. А., Иванова Н. Г., Русакова О. Л.* Информатика: учебник для 3 класса: в 2 ч. М.: БИНОМ. Лаборатория знаний, 2013.

8. *Плаксин М. А., Иванова Н. Г., Русакова О. Л.* Информатика: учебник для 4 класса: в 2 ч. М.: БИНОМ. Лаборатория знаний, 2013.

9. *Плаксин М. А., Иванова Н. Г., Русакова О. Л.* Набор заданий для знакомства с параллельными вычислениями в конкурсе «ТРИЗформашка» // Преподавание информационных технологий в Российской Федерации. Материалы Тринадцатой открытой Всероссийской конференции «ИТ-Образование — 2015» (г. Пермь, 14–15 мая 2015 г.). Пермь: Перм. гос. нац. иссл. ун-т, 2015.

10. *Романовский И. В.* Дискретный анализ. 3-е изд., перераб. и доп. СПб.: Невский Диалект; БХВ-Петербург, 2003.

И. Н. Фалина, Н. А. Булгакова,
Специализированный учебно-научный центр (факультет) — школа-интернат имени А. Н. Колмогорова Московского государственного университета имени М. В. Ломоносова (СУНЦ МГУ),

А. И. Фалин,
факультет вычислительной математики и кибернетики Московского государственного университета имени М. В. Ломоносова

КЛАССИФИКАЦИЯ УЧЕБНЫХ ЗАДАЧ ПО ИНФОРМАТИКЕ ПО УРОВНЮ СЛОЖНОСТИ

Аннотация

В статье описывается способ классификации учебных задач по информатике. В основу классификации положена таксономия Д. Толлингеровой и системно-структурный анализ учебной задачи Л. М. Фридмана. Предлагаемая классификация учитывает как когнитивную сложность задачи, так и ее структурную сложность и описывает три уровня сложности учебных задач по информатике. Вводится понятие методически важной задачи, даны примеры классификации задач по теме «Введение в алгебру логики».

Ключевые слова: учебная задача, классификация учебных задач, когнитивная сложность задачи, структурная сложность задачи, методически важная задача, таксономия Толлингеровой, информатика, классификация логических задач, уровень сложности задачи.

Контактная информация

Фалина Ирина Николаевна, канд. пед. наук, доцент кафедры информатики Специализированного учебно-научного центра (факультета) — школы-интерната имени А. Н. Колмогорова Московского государственного университета имени М. В. Ломоносова (СУНЦ МГУ); *адрес:* 121357, г. Москва, ул. Кременчугская, д. 11; *телефон:* (499) 449-07-15; *e-mail:* falina.irina@gmail.com

Булгакова Наталья Алексеевна, ассистент кафедры информатики Специализированного учебно-научного центра (факультета) — школы-интерната имени А. Н. Колмогорова Московского государственного университета имени М. В. Ломоносова (СУНЦ МГУ); *адрес:* 121357, г. Москва, ул. Кременчугская, д. 11; *телефон:* (499) 449-07-15; *e-mail:* n.bulgakova.msu@gmail.com

Фалин Анатолий Иванович, канд. физ.-мат. наук, доцент кафедры общей математики факультета вычислительной математики и кибернетики Московского государственного университета имени М. В. Ломоносова (ВМК МГУ); *адрес:* 119991, г. Москва, Ленинские горы, МГУ имени М. В. Ломоносова, 2-й учебный корпус, факультет ВМК; *телефон:* (495) 939-55-91; *e-mail:* falin.anatoly@gmail.com

I. N. Falina, N. A. Bulgakova, A. I. Falin,
Lomonosov Moscow State University

CLASSIFICATION OF LEARNING TASKS ON INFORMATICS DEPENDING OF COMPLEXITY LEVEL

Abstract

The article describes a method for classifying learning tasks on informatics. The classification is based on the taxonomy of D. Tollingerova and the system-structural analysis of the learning task of L. M. Friedman. The proposed classification takes into account both the cognitive complexity of the task and its structural complexity and describes the three levels of complexity of learning tasks on informatics. The notion of a methodically important task is introduced, examples of classification of tasks on the topic «Introduction to the algebra of logic» are given.

Keywords: learning task, classification of learning tasks, cognitive complexity of task, structural complexity of task, methodically important task, Tollinger taxonomy, informatics, classification of logical tasks, level of complexity of task.

Современное информационное общество предъявляет новые требования к результатам образования школьников, и не в последнюю очередь это относится к школьной информатике. Для эффективного обучения учителю полезно иметь в своем распоряжении сбалансированную систему задач. Для этого необходимо конструирование наборов задач, разнообразных и по сложности, и по используемым методам решения.

На практике действующие учителя в первую очередь сталкиваются с проблемой определения сложности учебной задачи. Действительно: как определить, конкретная задача будет сложна или нет для учащихся? Если вы сталкиваетесь с новой задачей, то как оценить ее сложность? Вы считаете, что она сложна, а ваш коллега предложил более простой и легкий способ решения. Как быть в этом случае? Как подобрать набор разноуровневых по сложности задач, но по одной теме и на один способ решения? Вопросов очень много, и хотелось бы получить на них ответы.

В отечественной педагогике были предложены разные классификации учебных задач в зависимости от выбранного основания. Этой проблемой занимались И. Я. Лернер, В. А. Крутецкий, А. М. Матюшкин, М. И. Махмутов, Т. В. Кудрявцев, В. В. Власов, Ю. М. Колягин и другие педагоги-исследователи. Универсального способа классификации учебных задач на данный момент не найдено. Интересные классификации по уровню сложности и проблемности учебных задач для естественно-математического цикла были предложены Э. П. Тарасовой [3] и И. Р. Федоровой [6]. Но эти классификации не учитывают в полной мере специфику предмета информатики.

В СУНЦ МГУ для определения сложности учебных задач по информатике разработана и используется классификация учебных задач на основе таксономии Д. Толлингеровой и системно-структурного анализа учебной задачи, предложенного Л. М. Фридманом [7].

В данной статье на примере логических задач из школьного курса информатики по-

казано, как применяется описываемая классификация учебных задач по уровню сложности.

1. Особенности изучения темы «Введение в алгебру логики»

С необходимостью использовать логику как специфическую когнитивную деятельность человек сталкивается на протяжении всей жизни, например, при изложении своих рассуждений, обработке информации и т. п. В современном информационном обществе мы наблюдаем некоторое замещение элементов логического мышления интуитивным восприятием. И никак не умаляя интуитивное восприятие как, возможно, наиболее эффективный способ познания окружающего мира, мы ясно видим, что для будущей профессиональной деятельности роль логического мышления является определяющей.

Способность логически мыслить не является врожденным качеством человека. Это умение можно и нужно развивать. Именно поэтому уже в младших классах на уроках математики и информатики одной из задач, поставленных перед учителем, является развитие логического мышления учащихся.

Тема «Введение в алгебру логики» является важной составляющей школьного курса информатики, но изучение этой темы связано с определенными особенностями:

- учащиеся сталкиваются с трудностями при запоминании и использовании символьных обозначений логических операций и сопоставлении их с таблицами истинности;
- математический аппарат алгебры логики непривычен для школьников (некоторые законы тождественных преобразований, имея одинаковое название в информатике и математике, применяются по-иному, например, закон дистрибутивности);
- высказывания, записанные на русском языке, сложны для формализации на языке алгебры логики из-за многообразия используемых оборотов в разговорной речи;
- у школьников возникают сложности с формализацией задач из-за двузначности алгебры логики (в то время как в повседневной жизни используется трехзначная логика: «да», «нет», «не знаю»).

Эти особенности необходимо учитывать и в процессе преподавания темы, и при определении сложности учебной задачи.

2. Основы предлагаемой классификации учебных задач по информатике

В предлагаемой классификации каждая задача характеризуется парой индексов (K, C), где K — ***когнитивная сложность***, и C — ***структурная сложность***.

Для оценки когнитивной сложности учебных задач по информатике мы используем таксономию Д. Толлингеровой.

Д. Толлингерова классифицирует учебные задачи по их «когнитивному составу», т. е. выделяет те умственные (когнитивные) действия, которые должен выполнить школьник для успешного решения предложенной задачи. В ее работах описаны 27 разновидностей задач и высказано утверждение, что предлагаемая типология исчерпывает все типы задач, встречающиеся в современных учебниках. Эти 27 разновидностей задач она объединила в пять категорий:

1) задачи на воспроизведение знаний (узнавание, воспроизведение фактов, текстов);
2) задачи, требующие простых мыслительных операций (перечисление, описание, анализ, классификация, сравнение, выявление взаимоотношений);
3) задачи, требующие сложных мыслительных операций (трансляция, интерпретация, аргументация, индукция, дедукция, доказательство, оценка);
4) задачи, требующие для своего решения продуктивного мышления (разработка докладов, проекты, визуализация);
5) задачи на продуктивное мышление с порождением на его основе письменного или устного высказывания (решение проблемных задач, постановка вопросов, новая формулировка заданий).

В каждой категории выделяется несколько видов задач, решение которых требует конкретных когнитивных действий. В нашей классификации индексу *когнитивная сложность* мы присваиваем номер категории по таксономии Толлингеровой.

В структуре каждой учебной задачи выделяется некоторое количество взаимосвязанных объектов (параметров), содержащихся в ней. Например, это количество элементарных высказываний в логических задачах. Количество таких структурных элементов оказывает существенное влияние на сложность решения задачи. Так, две задачи, отнесенные по таксономии Толлингеровой к одной и той же категории, но содержащие разное количество параметров, могут потребовать от учащихся разной вычислительной сложности.

Структурная сложность в нашей классификации также выражается числом и зависит от количества объектов/элементов/параметров, которые надо учитывать при решении задачи. Для задач по теме «Введение в алгебру логики» мы используем такие числовые значения этого индекса:

- 1 — 2 структурных элемента (2 простых высказывания);
- 2 — 3 или 4 элемента;
- 3 — 5 или 6 элементов;
- 4 — 7 или 8 элементов;
- 5 — 9 и больше элементов.

Для задач по каждому разделу курса можно устанавливать свои критерии значений индекса *структурная сложность*.

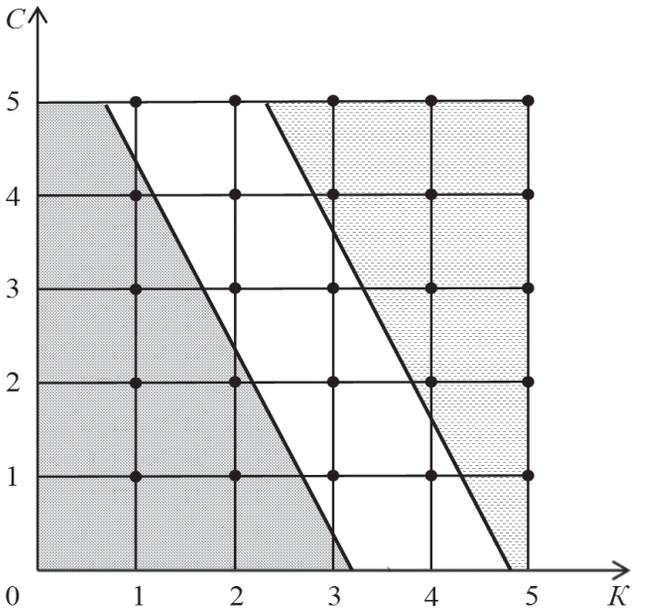
Отметим, что в структурной сложности иногда приходится учитывать и *вычислительную сложность* задачи. В задачах по информатике, собственно, как и в задачах других школьных предметов, вычислительная сложность задачи зависит, с одной стороны, от числа параметров, а с другой стороны, от способа решения задачи. В такой ситуации может возникнуть неоднозначность определения структурной сложности. В качестве одного из путей снятия этой неоднозначности мы предлагаем рассчитывать структурную сложность задачи, опираясь на тот способ решения, который как наиболее эффективный был рассказан школьникам и показан на примере задачи, которую мы называем методически важной.

Под ***методически важной задачей*** мы понимаем такую учебную задачу, алгоритм решения которой

ученикам незнаком или на примере решения которой можно продемонстрировать новый, более эффективный алгоритм решения задач подобного типа.

Именно при разборе методически важных задач учитель знакомит школьников с наиболее эффективными способами решения. Термин «эффективный способ решения» в информатике имеет свою специфику. Так, в программировании под эффективным алгоритмом понимается, как правило, алгоритм с минимальной вычислительной сложностью; иногда требуется построить алгоритм с использованием минимального количества памяти.

Таким образом, в результате описанной таксации каждой задаче по теме «Введение в алгебру логики» будет поставлена в соответствие пара (K, C). Индексы K и C изменяются от 1 до 5, т. е. каждой задаче соответствует одна из точек целочисленной решетки:



Далее мы все точки разбиваем на три множества, соответствующие разным уровням сложности: низкий, средний и высокий. Сделать это можно различными способами.

Мы для описания множеств используем следующие соотношения:

- если $2K + C \le 6$, то задача относится к *низкому уровню* сложности;
- если $7 \le 2K + C \le 9$, то задача относится к *среднему уровню* сложности;
- если $2K + C \ge 10$, то задача относится к *высокому уровню* сложности.

Коэффициент 2 повышает вес индекса *когнитивная сложность*. Очевидно, что индексы K и C могут входить в соотношение с любыми коэффициентами в зависимости от целей формирования системы задач.

В рассматриваемом случае выделенные уровни сложности описываются следующими характеристиками:

- *задачи низкого уровня сложности* предполагают, прежде всего, невысокий когнитивный уровень сложности, т. е. формулировка задания должна быть четкой, условие задачи должно однозначно интерпретироваться, а ее решение требует простых мыслительных операций; при этом структурная сложность не обязательно должна быть самой низкой;
- *задачи среднего уровня сложности* предполагают более высокий уровень когнитивной сложности; индекс структурной сложности может принимать любое из допустимых значений;
- *задачи высокого уровня сложности* отличаются повышенным уровнем когнитивной сложности, большим количеством параметров, громоздкими вычислениями или нестандартным творческим подходом к их решению.

Кроме того, мы делим все задания на следующие группы:

- *методически важные*, т. е. обязательные для разбора учителем на уроке;
- *для самостоятельной работы в классе* — задачи, направленные на отработку алгоритмов решения в течение урока и не требующие обязательной проверки учителем;
- *для домашней работы* — задачи для закрепления пройденного материала;
- *для контроля* — задачи, которые могут быть использованы учителем для проверки усвоения материала темы.

3. Примеры классификации задач по теме «Введение в алгебру логики»

3.1. Задачи на формализацию выражений

Для задач, приведенных в этом разделе, предлагается записать высказывание на языке алгебры логики, выделив простые высказывания. Все задачи, приведенные в статье, были отобраны из источников [1, 2, 8].

Пример 1 (методически важная задача).

Дано высказывание:

«Вранье, что нужно выбирать между участием в олимпиадах по математике и информатике, я успею и то, и другое».

Запишите его на языке алгебры логики.

Решение.

1. Выделим простые высказывания:

M = {*Участвовать в олимпиаде по математике*};

I = {*Участвовать в олимпиаде по информатике*}.

2. Исходному высказыванию соответствует формула:

$$(\overline{M \oplus I}) \,\&\, (M \,\&\, I) =$$

$$= \overline{(M \,\&\, \bar{I} \vee \bar{M} \,\&\, I)} \,\&\, (M \,\&\, I) =$$

$$= (\overline{M \,\&\, \bar{I}} \,\&\, \overline{\bar{M} \,\&\, I}) \,\&\, (M \,\&\, I) =$$

$$= (\bar{M} \vee I) \,\&\, (M \vee \bar{I}) \,\&\, (M \,\&\, I) =$$

$$= (M \,\&\, I \vee \bar{M} \,\&\, \bar{I}) \,\&\, (M \,\&\, I) =$$

$$= M \,\&\, I \,\&\, I \vee M \,\&\, M \,\&\, I = M \,\&\, I.$$

3. То есть исходное высказывание можно переписать в виде:

«*Можно участвовать и в олимпиаде по математике, и в олимпиаде по информатике*».

Ответ. $M \,\&\, I$.

Определение сложности задачи.

- *Когнитивная сложность* = 3. Исходное высказывание не содержит простых высказываний в явном виде, что может ввести учащихся в заблуждение. Эту задачу, по таксономии Толлингеровой, сле-

дует отнести к категории 3, тип 3.2 (задача на интерпретацию, разъяснение смысла).
- *Структурная сложность* = 1 (задача имеет два параметра, т. е. два простых высказывания).
- *Сложность задачи* = 2 · 3 + 1 = 7 ≡ средняя.

Пример 2 (методически важная задача).

Дано высказывание:
«*Если яблоко мелкое, то оно должно быть твердым, и это все только в случае, если оно зеленое*».
Запишите его на языке алгебры логики.

Решение.

1. Выделим простые высказывания:
M = {*Яблоко мелкое*};
T = {*Яблоко твердое*};
Z = {*Яблоко зеленое*}.

2. Высказывание «*если M, то T*» соответствует импликации: $M \to T$.
Высказывание «*A только в случае B*» истинно на единственном наборе, когда истинны оба высказывания A и B, а это соответствует конъюнкции. Важно не спутать эти два высказывания.

Ответ. $Z \,\&\, (M \to T)$.

Определение сложности задачи.
- *Когнитивная сложность* = 3. Формализация таких высказываний, как правило, вызывает трудности из-за наличия в высказывании редко встречающейся логической связки «только в случае...». Ее легко спутать с импликацией. Кроме того, школьники должны понять, какое условие записано — необходимое или достаточное, и знать, какой импликацией оно реализуется — прямой или обратной. Задача, по таксономии Толлингеровой, может быть отнесена к категории 3, тип 3.2 (задачи по разъяснению смысла, обоснование).
- *Структурная сложность* = 2 (три параметра).
- *Сложность задачи* = 2 · 3 + 2 = 8 ≡ средняя.

Пример 3 (задача для самостоятельной работы в классе).

Дано высказывание:
«*Для того чтобы измерить угол, достаточно использовать транспортир*».
Запишите его на языке алгебры логики.

Решение.

1. Выделим простые высказывания:
Y = {*Измерить угол*};
T = {*Использовать транспортир*}.

2. Логическая связка «*для Y достаточно T*» соответствует обратной импликации.

Ответ. $Y \leftarrow T$.

Определение сложности задачи.
- *Когнитивная сложность* = 2. Перевод высказываний, содержащих связки «необходимо» или «достаточно», часто вызывает сложности у учащихся. Эту задачу, по таксономии Толлингеровой, можно отнести к категории 2, тип 2.7 (задачи по выявлению взаимоотношений).
- *Структурная сложность* = 1 (два параметра).
- *Сложность задачи* = 2 · 2 + 1 = 5 ≡ низкая.

Пример 4 (задача для домашней работы).

Дано высказывание:
«*Неверно, что из того, что светит солнце, следует, что на улице нет дождя*».
Запишите его на языке алгебры логики.

Решение.

1. Выделим простые высказывания:
C = {*Светит солнце*};
D = {*Идет дождь*}.

2. Логическая связка «*неверно, что A*» соответствует операции отрицания. Исходное высказывание является отрицанием высказывания A = {*Из того, что светит солнце, следует, что на улице нет дождя*}.
Условие «*из C следует D*» соответствует импликации.

Ответ. $\overline{C \to \overline{D}} = C \,\&\, D$.

Возможно второе решение задачи, если в качестве простого высказывания было выбрано D = {*Дождя нет*}. Тогда ответом будет формула: $\overline{C \to D}$.

Определение сложности задачи.
- *Когнитивная сложность* = 3. От учащихся требуется умение правильно интерпретировать высказывания, подразумевающие использование отрицания вместе с другими связками. Эта задача, по таксономии Толлингеровой, может быть отнесена к категории 3, тип 3.2 (задачи по изложению: интерпретация/обоснование).
- *Структурная сложность* = 1 (два параметра).
- *Сложность задачи* = 2 · 3 + 1 = 7 ≡ средняя.

3.2. Логические задачи

Исходными данными в логических задачах являются высказывания. Взаимосвязи между высказываниями бывают настолько сложными, что решить задачу без использования специальных методов часто очень трудно. Аппарат алгебры логики позволяет построить формальный универсальный алгоритм решения логических задач, основанный на формализации высказываний и тождественном преобразовании логической формулы, построенной из этих высказываний. Для некоторых задач после формализации и построения общей формулы вместо этапа тождественных преобразований можно использовать метод логических рассуждений на основе проверки выдвинутых предположений.

Для ряда логических задач целесообразнее использовать табличный метод решения. Такой метод можно применять, если в задаче используется множество параметров, связанных друг с другом некоторым условием. Тогда это условие удобно отобразить графически — в виде таблицы.

На приведенных ниже примерах покажем, как вычисляется сложность логической задачи.

Пример 5 (задача для домашней работы).

Маша, Таня и Оля сорвали в лесу необычный гриб. Они были не сильны в грибах, а потому высказали по несколько предположений.

Маша: «Это боровик, и его хорошо бы засушить».

Таня: «Это лисичка, и ее не стоит сушить».

Оля: «Это подберезовик, и его хорошо бы отварить».

Бабушка сказала девочкам, как назывался гриб, и приготовила его. В результате оказалось, что каждая из девочек была права лишь в одном предположении.

Что это был за гриб и как он был приготовлен?

Решение.

1. Выделим простые высказывания:
$B = \{$*Гриб — боровик*$\}$;
$P = \{$*Гриб — подберезовик*$\}$;
$Z = \{$*Гриб нужно засушить*$\}$.

Высказывание «*Найденный гриб — лисичка*» является отрицанием простого высказывания $B \vee P$.

Условие «*Гриб нужно сварить*» записывается через отрицание высказывания Z, так как в задаче рассматриваются только два способа приготовления грибов.

2. Запишем на языке алгебры логики высказывания девочек, используя разделительную дизъюнкцию.

Маша: $B \oplus Z$;
Таня: $\bar{B}\,\&\,\bar{P} \oplus \bar{Z}$;
Оля: $P \oplus \bar{Z}$.

3. Задачу будем решать методом тождественных преобразований. Условие задачи можно записать в виде формулы:

$$(B \oplus Z)\,\&\,(\bar{B}\,\&\,\bar{P} \oplus \bar{Z})\,\&\,(P \oplus \bar{Z}) = 1.$$

4. Упростив формулу, получим:

$$\bar{Z}\,\&\,B\,\&\,\bar{P}.$$

Ответ. Найденный гриб — боровик, и его нужно сварить.

Определение сложности задачи.

- *Когнитивная сложность* = 3. По таксономии Толлингеровой, эта задача относится к типу 3.2 (задачи на интерпретацию, разъяснение смысла).
- *Структурная сложность* = 2 (три параметра).
- *Сложность задачи* = $2 \cdot 3 + 2 = 8 \equiv$ средняя.

Пример 6 (задача для самостоятельной работы).

Школьник попросил троих друзей отгадать, какое он загадал число из набора: положительное, отрицательное, четное, нечетное, целое, дробное.

Первый сказал, что если это число четное, то оно положительное.

Второй предположил, что это число четное или же одновременно целое и положительное.

Третий был уверен, что если это число неотрицательное, то оно нечетное.

Оказалось, что все три друга были правы.

Какое было задумано число?

Решение.

1. Выделим простые высказывания:
$P = \{$*Число положительное*$\}$;
$E = \{$*Число четное*$\}$;
$Z = \{$*Число целое*$\}$.

2. Запишем на языке алгебры логики высказывания троих друзей:

первый: $E \to P$;
второй: $E \vee Z\,\&\,P$;
третий: $P \to \bar{E}$ (по условию задачи, число не может быть нулем; следовательно, высказывание «*число неотрицательное*» эквивалентно высказыванию «*число положительное*»).

3. Конъюнкция обращается в единицу тогда, и только тогда, когда в единицу обращается каждый ее аргумент, следовательно:

$$(E \to P)\,\&\,(E \vee Z\,\&\,P)\,\&\,(P \to \bar{E}) = 1.$$

4. Преобразуем полученную формулу:

$$(E \to P)\,\&\,(E \vee Z\,\&\,P)\,\&\,(P \to \bar{E}) =$$
$$= (\bar{E} \vee P)\,\&\,(E \vee Z\,\&\,P)\,\&\,(\bar{P} \vee \bar{E}) =$$
$$= \{\text{для удобства чтения опустим знак конъюнкции}\} =$$
$$= (\bar{E}E \vee P\bar{E}Z \vee EP \vee ZP)\,\&\,(\bar{P} \vee \bar{E}) =$$
$$= (EP \vee ZP)\,\&\,(\bar{P} \vee \bar{E}) =$$
$$= P\bar{P}E \vee P\bar{P}Z \vee PE\bar{E} \vee P\bar{E}Z = P\bar{E}Z.$$

Ответ. Загаданное число — целое, положительное, нечетное.

Определение сложности задачи.

- *Когнитивная сложность* = 2. От учащихся требуется умение уменьшать количество входных параметров и устанавливать простые взаимосвязи, поэтому задача, по таксономии Толлингеровой, относится к 2.4 (задачи по разбору и структуре).
- *Структурная сложность* = 2 (три параметра).
- *Сложность задачи* = $2 \cdot 2 + 2 = 6 \equiv$ низкая.

Пример 7 (методически важная задача).

Для международной конференции, проходящей в Мюнхене, надо из восьми учителей школы выбрать шестерых: физика, химика, биолога, математика, физрука и трудовика. По регламенту запрещено одному человеку представлять разные должности, хотя у некоторых учителей в дипломе значится несколько специальностей. Так, «физика» есть в дипломах Говоровой и Немцова, «биология» — у Корнеевой и Фримена, «химия» — у Фримена и Немцова, «математика» — у Долганова и Лыковой, «физическая культура» — у Долганова и Хоботовой, а дисциплина «труд» — у Волкова и Лыковой. Но из-за существования внутриколлективных связей появились следующие пожелания: Фримен хотел бы поехать с Корнеевой, Лыкова с Хоботовой и Долгановым, Долганов не хотел бы ехать с Немцовым, а Волков — с Корнеевой.

Кого следует послать на международную конференцию?

Решение.

1. Задачу будем решать комбинацией табличного метода и метода логического анализа.

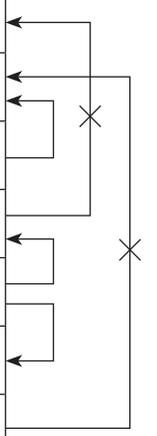
Для решения составим таблицу «Фамилия/Преподаваемые дисциплины», в которой отразим соответствие фамилии преподавателя и специальности в его дипломе.

2. На этой же таблице стрелками покажем, кто с кем хочет или нет ехать на конференцию.

Такая таблица служит основой для логического анализа на основе выдвинутых предположений. При этом будем считать, что выдвинутые пожелания о совместной поездке являются обязательным условием для участия в поездке.

- Предположим, что Волков *не едет* на конференцию. Тогда на основе анализа таблицы получаем, что обязательно едет Лыкова, так как из оставшихся преподавателей только она преподает труд.

Физика	Химия	Биология	Математика	Физкультура	Труд	
+						Говорова
+	+					Немцов
		+				Корнеева
	+	+				Фримен
			+	+		Долганов
			+		+	Лыкова
				+		Хоботова
					+	Волков



- Лыкова едет только вместе с Долгановым и Хоботовой. Анализируя таблицу, получим, что Хоботова едет как физрук, а Долганов едет как математик.
- Так как Долганов едет, то не едет Немцов. Тогда из анализа таблицы получаем, что Говорова едет как физик, Фримен — как химик.
- Фримен едет обязательно с Корнеевой, она преподает биологию.

3. Если же предположить, что Волков *едет* на конференцию, то, проведя аналогичный анализ, мы получим противоречие.

Ответ. На конференцию нужно послать Лыкову (труд), Хоботову (физическая культура), Долганова (математика), Говорову (физика), Фримена (химия), Корнееву (биология).

Определение сложности задачи.

- *Когнитивная сложность* = 3. По таксономии Толлингеровой, задача относится к типу 3.5 (задачи по аргументации и оценке).
- *Структурная сложность* = 5 (в задаче используется более девяти простых высказываний).
- *Сложность задачи* = $2 \cdot 3 + 5 = 11 \equiv$ высокая.

Пример 8 (задача для самостоятельной работы в классе).

В кафе встретились три друга: скульптор Белов, скрипач Чернов и художник Рыжов.

«Замечательно, что один из нас имеет белые, один черные и один рыжие волосы, но ни у одного из нас нет волос того цвета, на который указывает его фамилия», — заметил черноволосый. «Ты прав», — сказал Белов.

Какой цвет волос у художника?

Решение.

Задачу будем решать табличным методом.

1. Составим таблицу «Фамилия/Цвет волос».

2. Заполним ее в соответствии с первым условием задачи, т. е. поставим «–» в клетках «Белов, белый», «Чернов, черный», «Рыжов, рыжий».

3. Первое высказывание принадлежит черноволосому, и черноволосый не Белов. Следовательно, в клетке «Белов, черный» тоже можно поставить «–».

	Белов	Чернов	Рыжов
Белый	–		
Черный	–	–	
Рыжий			–

4. Остальные клетки заполняются по правилу: если в столбце или строке все клетки, кроме одной, помечены «–», то в пустой клетке ставим «+». Если в столбце/строке есть «+», то все остальные клетки помечаем «–».

Ответ. У художника Рыжова волосы черные.

Определение сложности задачи.

- *Когнитивная сложность* = 2. Задание относится к типу 2.7 (задача на выявление взаимоотношений между фактами) по таксономии Толлингеровой.
- *Структурная сложность* = 2 (четыре простых высказывания).
- *Сложность задачи* = $2 \cdot 2 + 2 = 6 \equiv$ низкая.

Пример 9 (задача для контрольной работы).

В соревнованиях по гимнастике участвуют Настя, Соня, Ира и Анжела. Болельщики высказали предположения о возможных победителях:

- первое место займет Настя, а Соня займет второе место;
- Настя займет второе место, а Ира — третье;
- Анжела будет второй, а Ира окажется четвертой.

По итогам соревнований оказалось, что в каждом предположении только одно высказывание истинно, а другое ложно.

Как распределились места между девочками, если все они заняли разные места?

Решение.

1. Выделим простые высказывания:
$H_1 = \{$*Настя заняла первое место*$\}$;
$C_2 = \{$*Соня заняла второе место*$\}$;
$H_2 = \{$*Настя заняла второе место*$\}$;
$A_2 = \{$*Анжела заняла второе место*$\}$;
$I_3 = \{$*Ира заняла третье место*$\}$;
$I_4 = \{$*Ира заняла четвертое место*$\}$.

2. Запишем в формальном виде (на языке алгебры логики) высказывания болельщиков и условия задачи в виде системы уравнений:

$$\begin{cases} H_1 \,\&\, C_2 \vee H_2 \,\&\, I_3 \vee A_2 \,\&\, I_4 = 0; \\ H_1 \vee C_2 = 1; \\ H_2 \vee I_3 = 1; \\ A_2 \vee I_4 = 1. \end{cases}$$

Задачу будем решать методом логического анализа высказанных предположений.

3. Есть только одно высказывание относительно Анжелы. Пусть $A_2 = 1$, тогда $I_4 = 0$. Это следует из первого уравнения системы. Но тогда $H_2 = 0$ и $C_2 = 0$ (второе место может занять только одна девочка). Следовательно, $H_1 = 1$ и $I_3 = 1$.

4. Проверим, возможно ли другое решение. Пусть $A_2 = 0$, тогда $I_4 = 1$ (это следует из четвертого уравнения системы). Отсюда можно сделать вывод, что $I_3 = 0$, так как Ира не может занять два различных места одновременно. Следовательно, $H_2 = 1$ (это следует из третьего уравнения системы). Но тогда $H_1 = 0$, а $C_2 = 1$. Получили противоречие: второе место заняли и Настя, и Соня.

5. Следовательно, первое полученное решение единственное.

Ответ. Настя займет первое место, Анжела — второе, Ира — третье, Соня — четвертое.

Определение сложности задачи.

- *Когнитивная сложность* = 3. По таксономии Толлингеровой, эта задача относится к типу 3.2 (от ученика требуются интерпретация факта и его обоснование).
- *Структурная сложность* = 3 (шесть параметров).
- *Сложность задачи* = $2 \cdot 3 + 3 = 9 \equiv$ средняя.

4. Заключение

Особенность использования предложенной классификации состоит в субъективности оценки требуемых когнитивных действий при решении задачи, что влияет на величину индекса *когнитивная сложность*. С одной стороны, эту особенность можно рассматривать со знаком минус (в силу субъективности и неоднозначности оценки), но, с другой стороны, попытка понять, какие умственные действия должен выполнить учащийся, имеет важное методическое значение. Если учитель обозначил для себя набор необходимых когнитивных действий учащихся, то тем самым он выделил основные особенности задачи, расставил акценты, на которые надо обратить внимание при объяснении способа решения. В этой ситуации учебная задача выступает как механизм опережающего когнитивного развития учащегося.

Кроме того, в отечественной методической литературе можно найти примерное соответствие формулировок задач («дайте определение...», «перечислите части...», «аргументируйте...», «докажите...», «классифицируйте...» и т. д.) и набора когнитивных действий, требуемых для решения конкретной задачи.

Список использованных источников

1. *Андреева Е. В., Босова Л. Л., Фалина И. Н.* Математические основы информатики: учебное пособие. М.: БИНОМ. Лаборатория знаний, 2007.

2. *Кольман Э., Зих О.* Занимательная логика. М.: Наука, 1966.

3. *Тарасова Э. П.* Классификация учебных задач на основе системно-структурного анализа // The World of Scientific Discoveries. 2012. № 4.1 (28).

4. *Толлингерова Д.* К психологической теории учебных задач // Социалистическая школа. 1976/77. № 4.

5. *Толлингерова Д., Голоушова Д.* Психология проектирования умственного развития детей. М.: Роспедагентство, 1994.

6. *Федорова И. Р.* К вопросу о систематике учебных задач // Современные проблемы науки и образования. 2015. № 1 (Ч. 1).

7. *Фридман Л. М.* Логико-психологический анализ школьных учебных задач. М.: Педагогика, 1977.

8. *Willis N. D.* The Little Giant Encyclopedia of Logic Puzzle. New York: Sterling Publishing Co., Inc., 2000.

НОВОСТИ

В Microsoft записали Smoke on the Water на ДНК

Группе исследователей из Microsoft Research совместно с исследователями Вашингтонского университета и Twist Bioscience удалось записать музыкальные произведения Deep Purple и Майлса Дэвиса на ДНК, а потом воспроизвести их. Это первый случай использования ДНК для хранения архивного образца. Он станет частью архива ЮНЕСКО «Память мира». Кодирование происходит путем перевода информации из двоичного кода в структурный код ДНК. При считывании производится обратное перекодирование в двоичный код. По словам исследователей, в сравнении с другими носителями ДНК может безопасно хранить данные тысячелетиями, в то время как, например, копии аудиозаписей в архивах подлежат замене каждые 10 лет. Помимо долговечности ДНК обладает невероятной вместительностью. В объеме ДНК размером всего с кончик карандаша может храниться около 10 Тбайт данных, что в среднем эквивалентно памяти стандартных 600 смартфонов, а вся цифровая информация мира сможет поместиться примерно в 9 литрах этой биологической субстанции.

(По материалам международного компьютерного еженедельника «Computerworld Россия»)

ЗАНИМАТЕЛЬНАЯ ИНФОРМАТИКА

В. Ф. Очков, Д. А. Иванов, А. Д. Моисеева,
Национальный исследовательский университет «МЭИ», г. Москва

ТОМОГРАФИЯ = ИНФОРМАТИКА + МАТЕМАТИКА + ФИЗИКА + БИОЛОГИЯ

Аннотация

В статье описана лабораторная работа в русле междисциплинарных связей на стыке информатики, математики, физики и биологии: моделирование работы медицинского томографа и изучение томографии на четырехмерном объекте.

Ключевые слова: рентген, томография, функция одного, двух и трех аргументов, компьютерная графика, Maple, Mathcad.

Контактная информация

Очков Валерий Федорович, доктор тех. наук, профессор, профессор Национального исследовательского университета «МЭИ», г. Москва; *адрес:* 111250, г. Москва, Красноказарменная ул., д. 14; *телефон:* (495) 362-71-71; *e-mail:* ochkov@twt.mpei.ac.ru

Иванов Дмитрий Александрович, канд. тех. наук, профессор, профессор Национального исследовательского университета «МЭИ», г. Москва; *адрес:* 111250, г. Москва, Красноказарменная ул., д. 14; *телефон:* (495) 362-77-14; *e-mail:* ivanovda@mpei.ru

Моисеева Анастасия Дмитриевна, студентка 1-го курса Института тепловой и атомной энергетики Национального исследовательского университета «МЭИ», г. Москва; *адрес:* 111250, г. Москва, Красноказарменная ул., д. 14; *телефон:* (495) 362-71-71; *e-mail:* MoiseevaAnD@mpei.ru

V. F. Ochkov, D. A. Ivanov, A. D. Moiseeva,
National Research University MPEI, Moscow

TOMOGRAPHY = INFORMATICS + MATHEMATICS + PHYSICS + BIOLOGY

Abstract

The article proposes a laboratory work at the intersection of informatics, mathematics, physics and biology: modeling the work of a medical tomography and studying tomography on a four-dimensional object.

Keywords: X-ray, tomography, function of one, two and three arguments, computer graphics, Maple, Mathcad.

Обычно под словом «томография» понимают современный метод медицинского исследования, с помощью которого можно быстро и точно диагностировать многие заболевания внутренних органов человека, а также контролировать ход лечения.

Говорят, что идея такого исследования пришла в голову одному врачу, когда он шел по рынку в канун Рождества и видел в мясном ряду разрубленные замороженные свиные туши, на срезах которых были четко видны контуры костей и мягких тканей. В анатомических музеях мира можно увидеть такие заспиртованные препараты — срезы человеческих тел. Мечта о том, чтобы получать такие срезы не только у трупов, но и у живых людей, давно волновала многих врачей и ученых. И только относительно недавно она воплотилась в жизнь. Томография стала реальностью благодаря важным открытиям, сделанным физиками, появлению вычислительной техники и созданию специального математического аппарата.

Поговорим немного о физике, о ее достижениях и открытиях, позволивших без ножа заглянуть внутрь человека.

Во-первых, это открытие *рентгеновских лучей*. В городском музее Мюнхена хранится интересный экспонат — первый рентгеновский снимок человека во весь рост. Это был довольно опасный опыт: сведения о том, что рентгеновские лучи далеко не безвредны, появились позже. Тем не менее этот метод исследования широко вошел в медицинскую практику. В частности, рентген помог победить такую страшную социальную болезнь, как туберкулез. Рентген позволяет правильно вправлять сломанные кости. Его применяют и в такой довольно рутинной операции, как лечение зубов. В наши дни мы обязаны ежегодно проходить процедуру рентгеновского фотографирования легких.

Во-вторых, это изобретение *томографа*, с помощью которого можно получать изображения внутренних органов и тканей. В скором времени томография человека тоже станет обязательной процедурой. Врачи считают, что нет здоровых людей, а есть недообследованные — люди, не пропущенные, в частности, через медицинский томограф. Но пока томография — это дорогостоящая процедура, назначаемая врачами только в особых случаях. Рентген намного дешевле и доступнее, но рентгеновские снимки довольно нечеткие. Мы еще это отметим, описывая ниже рисунок 7.

Есть два вида медицинской томографии: КТ и МРТ. Компьютерная томография (КТ) относится к методам рентгеновского исследования. Она основана на измерении и сложной компьютерной обработке данных по разности ослабления рентгеновского излучения тканями живого организма различной плотности. Магнитно-резонансная томография (МРТ) базируется на явлении *ядерного магнитного резонанса* (ЯМР). Этот способ исследования основан на измерении электромагнитного отклика (эха) атомных ядер, в частности, атомов водорода. Концентрация водорода в организме человека разная в разных местах, что позволяет видеть на экране компьютера «срезы» человека со структурой внутренних органов.

Кстати говоря, сначала эти методы исследования назывались рентгеновской томографией (РТ) и ядерно-магнитной томографией (ЯМТ), но потом в обиход ввели термины КТ

и МРТ из-за радиофобии людей, связанной, в частности, с чернобыльской катастрофой. Да, медики часто в благих целях обманывают, скажем мягче, вводят в заблуждение своих пациентов. Хорошо ли это или плохо — ответить трудно. Исследования на магнитно-резонансном томографе вместо компьютерного томографа введены в медицинский обиход в том числе и для снижения дозы ионизирующего облучения.

Мы не случайно чуть выше слово «отклик» дополнили словом «эхо». Есть также способ ультразвукового медицинского исследования (УЗИ), но он не связан с томографией, хотя тоже позволяет заглянуть внутрь человека. Но не так глубоко и не с такой четкостью, как при КТ или МРТ. Хотя УЗИ намного безопаснее, дешевле и доступнее, чем рентген, КТ и МРТ.

Здесь мы не будем вдаваться глубоко в детали этих методов исследования, так как обо всем этом можно прочесть в Интернете. Скажем лишь, что ученые, внесшие наибольший вклад в развитие описанных методов диагностики, получили Нобелевскую премию. Сам же Конрад Рентген стал первым нобелевским лауреатом по физике.

Но термин «томография» мы стали связывать сугубо с медициной сравнительно недавно. Вернее, так: сначала это было чисто геометрическое понятие, а затем оно проникло и в медицину. Краткое справочное описание термина «томография» такое: ***томография*** (др.-греч. τομή — сечение) — получение послойного изображения внутренней структуры объекта.

Томография в медицине позволяет представить трехмерное тело человека в виде множества двумерных объектов — плоских срезов, отображаемых на дисплее компьютера или на бумаге принтера. Врач, двигая мышку и крутя ее ролик, может перемещаться вдоль этих сечений и воссоздавать структуру внутренних органов человека, выявлять в них аномалии и задумываться в случае чего о возможном хирургическом вмешательстве или терапевтическом лечении.

Давайте для начала создадим томограмму не трехмерного, а четырехмерного объекта на простом геометрическом примере.

Есть такая «народная» задача оптимизации. Народная в том смысле, что она широко «гуляет» по бумажным учебникам и Интернету, но ее конкретный автор неизвестен. В этом несложно убедиться, если в каком-либо поисковике Интернета сделать запрос по ключу «Коробка максимального объема». В [4] дан один из примеров многочисленных публикации этой задачи.

Суть задачи в следующем. Берется квадратный лист бумаги (картона, жести и т. д.) с длиной стороны, равной единице. В углах квадрата вырезаются четыре одинаковых квадрата меньшего размера. Далее из такой крестообразной заготовки складывается коробка загибанием прямоугольных участков вверх (рис. 1). Спрашивается: каким должно быть значение переменной x (длины стороны маленького квадрата), чтобы объем полученной коробки был максимальным? Длина стороны исходного квадрата равна, повторяем, единице.

Эту задачу несложно решить и без компьютера. Достаточно вспомнить азы математического анализа. Для решения можно создать функцию, возвращающую объем коробки (произведение площади ее основания $(1 - 2x)^2$ на высоту x) в зависимости от значения x, а затем взять производную от этого кубического полинома. Получится квадратный полином вида $12x^2 - 8x + 1$, у которого два нуля: 1/2 и 1/6. Известно, что у гладкой непрерывной

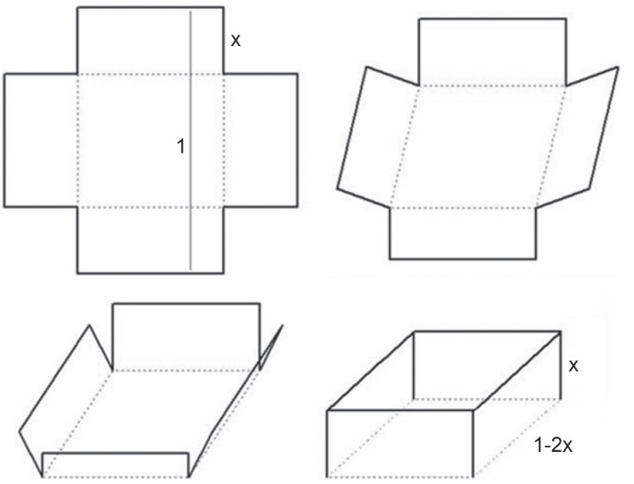


Рис. 1. Изготовление коробки из квадратной заготовки

функции в точках максимума, минимума или перегиба первая производная равна нулю. Следовательно, решение нашей задачи о коробке максимального размера такое: $x = 1/6$. Другой нуль квадратного полинома (1/2) — это ложное решение: точка минимума, а не максимума. В этом несложно убедиться, построив график функции $(1 - 2x)^2 \cdot x$ на интервале от 0 до 0,5 или подсчитав значение второй производной при $x = 1/2$ и $x = 1/6$. В первом случае оно будет положительным (минимум), а во втором — отрицательным (максимум).

Показанное выше решение, повторяем, можно найти во многих бумажных и интернетовских источниках. Но задача о коробке максимального объема имеет свежее и довольно интересное «бесконечное» продолжение. Можно четыре квадрата, отрезанные от исходной заготовки (см. рис. 1), не выбрасывать, а пустить в дело — сделать из них четыре новые коробки меньшего размера по той же схеме раскроя. Из шестнадцати $(4 \cdot 4)$ новых отрезанных квадратов опять же можно сделать новые коробки. Из шестидесяти четырех $(16 \cdot 4)$ новых отрезанных квадратов опять же можно сделать новые коробки. И так далее до бесконечности. Это будет некий фрактал [3], требующий оптимизации — определения размеров сторон квадратных вырезок, при которых суммарный объем всех полученных коробок будет максимальным.

В общем решении для пяти коробок (рис. 2) вводится функция пользователя с именем V_5, у которой два аргумента — x и y. Далее решается система двух алгебраических уравнений: равенство нулю частных производных функции V_5 по x и по y. Это, повторяем, необходимое условие экстремума для непрерывной гладкой функции двух аргументов. Координаты точек экстремума функции V_5 обязаны удовлетворять этой системе уравнений. Далее можно в каждой найденной точке проверить выполнение достаточного условия экстремума. Но можно результат проанализировать не аналитически, а графически — построить линии уровня функции двух переменных (контурный график — см. рис. 2). Полученная система уравнений имеет четыре корня, один из которых — это точка максимума, т. е. решение нашей задачи о пяти коробках. Три других корня — это точка минимума ($x = 0{,}5$, $y = 0{,}5$) и две так называемые «седловые точки», что ясно видно из графика, показанного на рисунке 2. Ниже на рисунке 4 мы этот контурный график отобразим в виде поверхности.

Из решения, показанного на рисунке 2 (да и по логике вещей), ясно, что квадратные заготовки для четы-

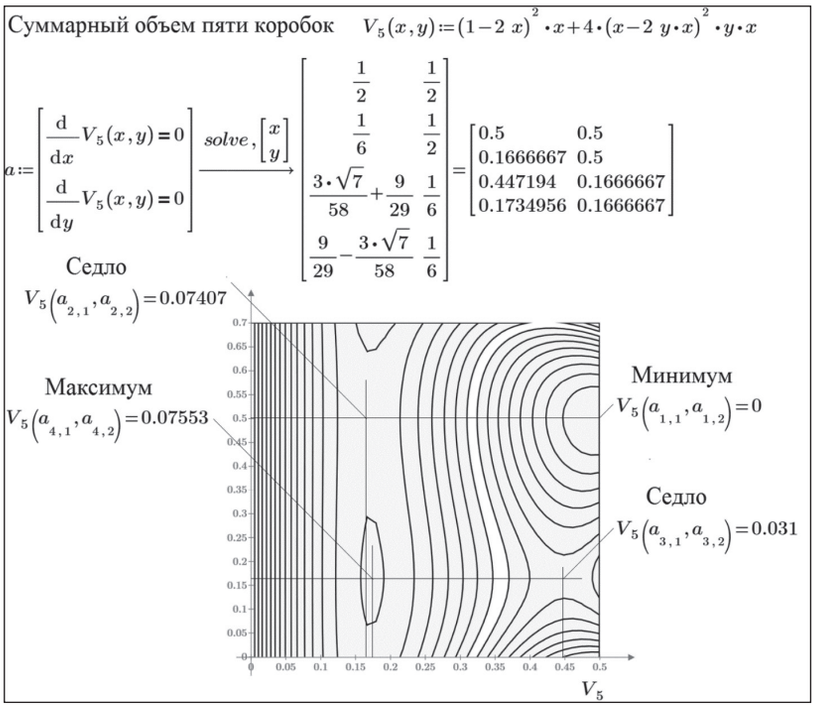


Рис. 2. Задача о пяти коробках

рех коробок нужно кроить в пропорции $y = 1/6$: задача о пяти коробках имеет не два неизвестных, а одно. Два аргумента нам нужны для графического анализа решения. Соответствующая же пропорция для центральной большой коробки (x) немного увеличится. На рисунке 2 мы получили некий срез функции двух аргументов, по которому легко сделать анализ функции — определить особые точки. Но томограмма — это не один срез, а несколько. Поэтому идем дальше.

Из вырезанных 16 квадратов, оставшихся после изготовления пяти коробок, также можно изготовить новые коробки. Спрашивается: при каком раскрое заготовок суммарный объем 21 коробки — одной большой, четырех средних и 16 маленьких — будет максимальным? Мы будем работать с функцией трех аргументов — x, y и z (рис. 3), хотя ясно, что тут можно обойтись функцией одного аргумента z, присвоив аргументам (параметрам) x и y значения, найденные ранее (см. рис. 2). Тут можно

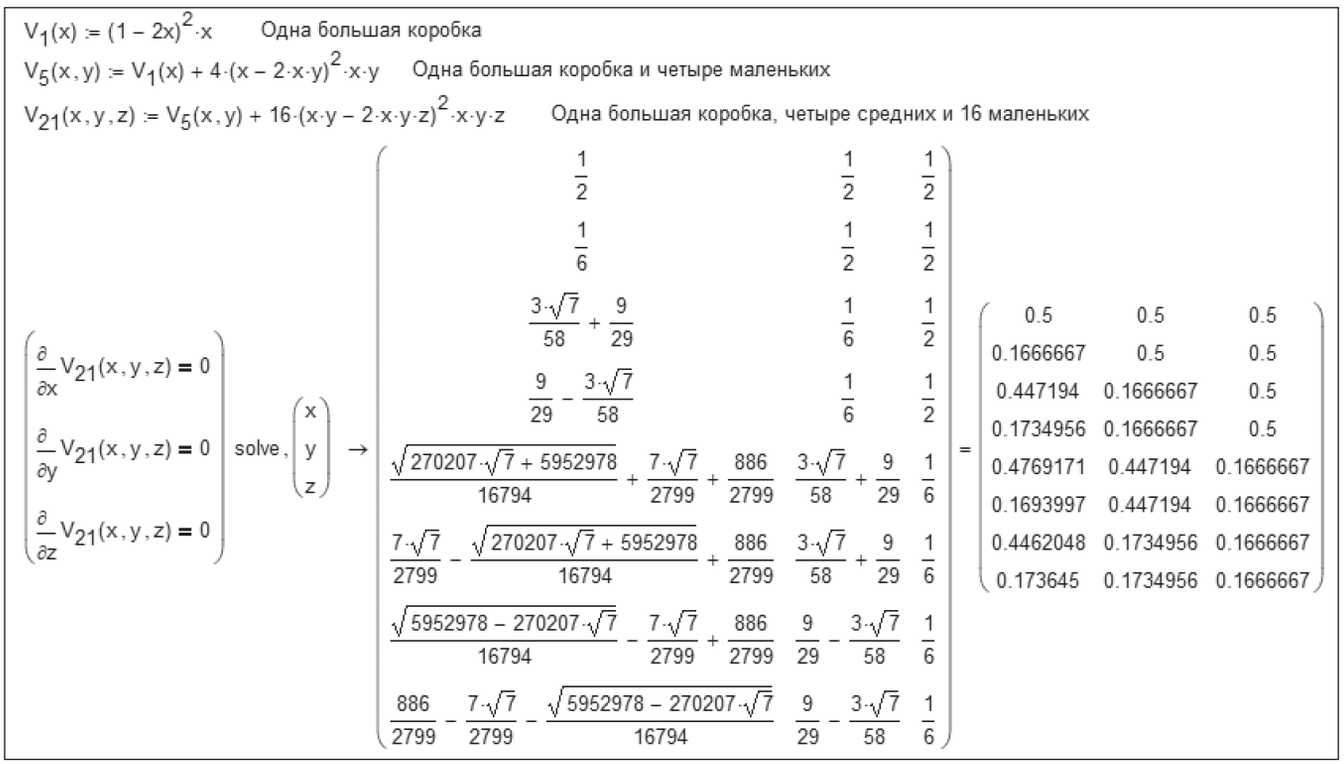


Рис. 3. Задача о 21 коробке

вывести рекуррентную формулу для определения очередного оптимального значения при очередном шаге раскроя коробок — 5 коробок, 21 коробка, 85 коробок и т. д. [3]. Функция трех аргументов нам будет нужна для ее визуального анализа, для построения ее *томограммы*.

А как визуально проанализировать функцию трех аргументов V_{21}, заданную на рисунке 3? Тут нам поможет анимация пакета Mathcad! Кадры анимации — это трехмерные срезы графика функции $f(y, z) := V_{21}(x, y, z)$, где каждой поверхности $f(y, z)$ соответствует определен-

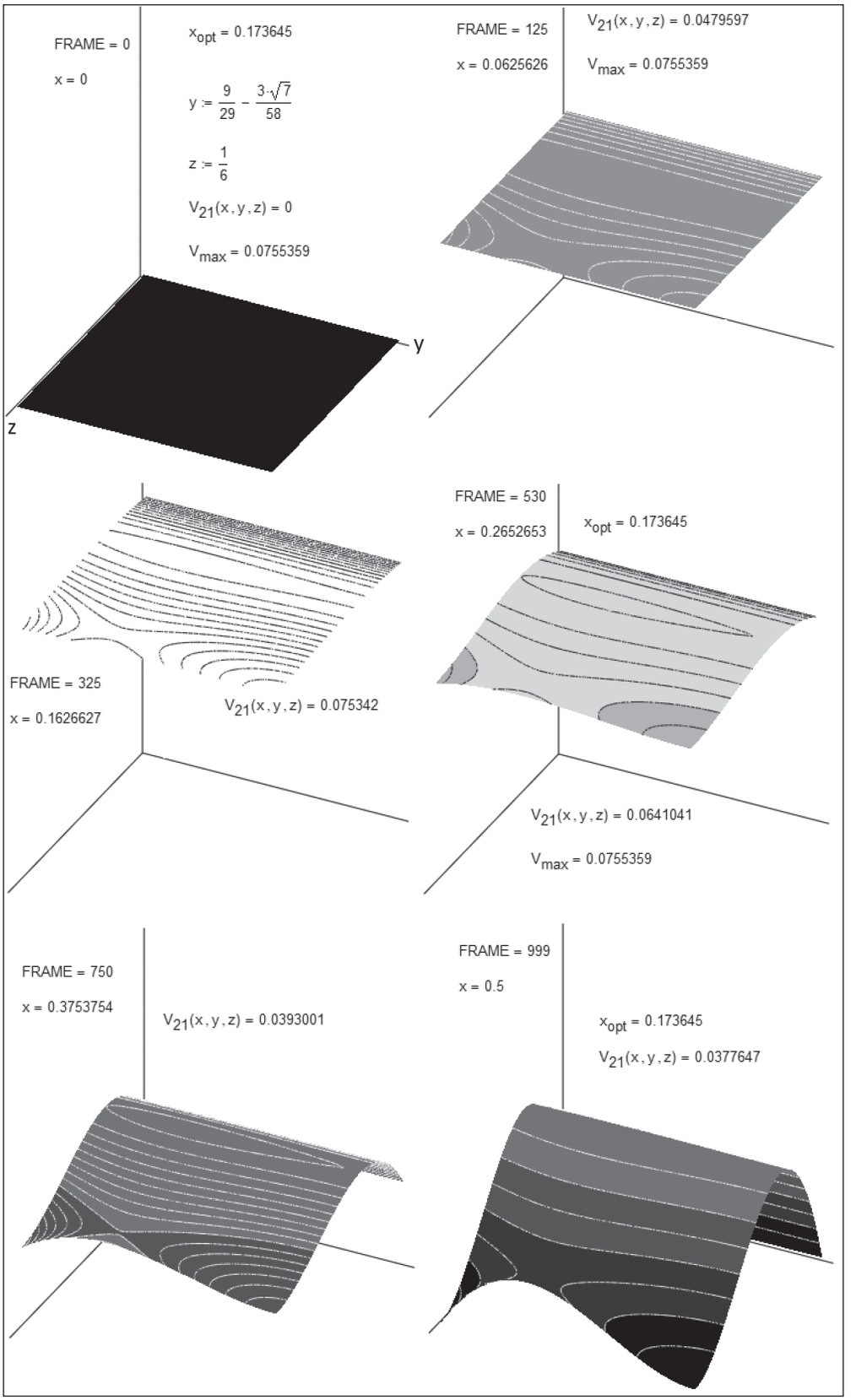


Рис. 4. Томограмма четырехмерной поверхности

ное значение переменной x в интервале от 0 (коробки отсутствуют) до 0,5 (20 коробок — исходную заготовку разрезали на четыре квадрата, а из каждого квадрата сделали пять коробок).

При значении x, равном нулю (левое верхнее изображение на рисунке 4), функция $V_5(x, y, z)$ также равна нулю, и мы имеем плоскую поверхность, ограниченную осями y и z. При значении x, равном 0,5 (левое нижнее изображение на рисунке 4), мы, повторяем, имеем не 21, а 20 коробок: исходный квадрат делится на четыре одинаковых квадрата, из которых делаются четыре большие коробки и 16 маленьких (см. рис. 2). Если менять значение параметра x от нуля до 0,5, то можно видеть деформацию сечений четырехмерной поверхности функции трех аргументов. Так с помощью томограммы можно визуализировать функцию трех аргументов и заглянуть внутрь четырехмерного графического объекта — см. один кадр такой анимации на рисунке 5.

Но давайте вернемся к трехмерным телам — к человеческому телу, например.

На рисунке 6 показано построение двух замкнутых поверхностей в среде Maple. Эта математическая программа имеет встроенную функцию *implicitplot3d* из пакета *plots* для графического отображения замкнутых (implicit) алгебраических выражений: не $z = f(x, y)$, а $f(z, y, z) = 0$. В расчет вводится формула деформи-

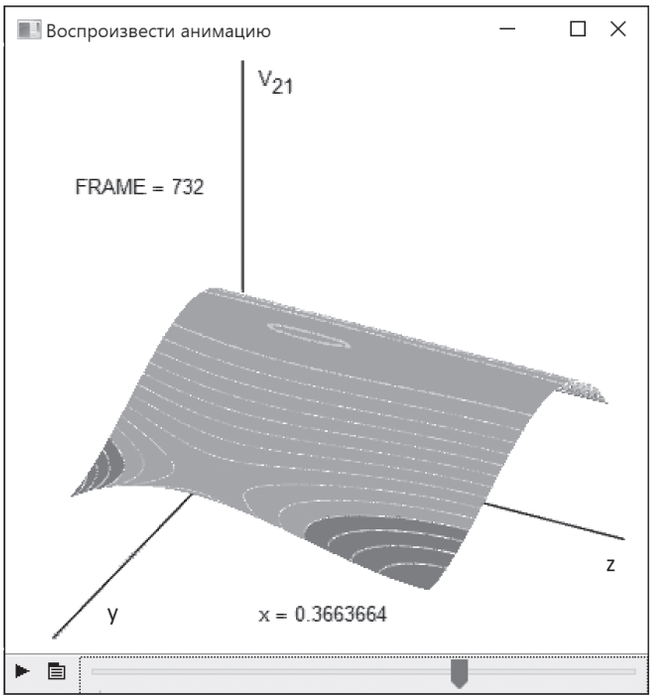


Рис. 5. Видеоплеер, показывающий томограмму четырехмерной поверхности

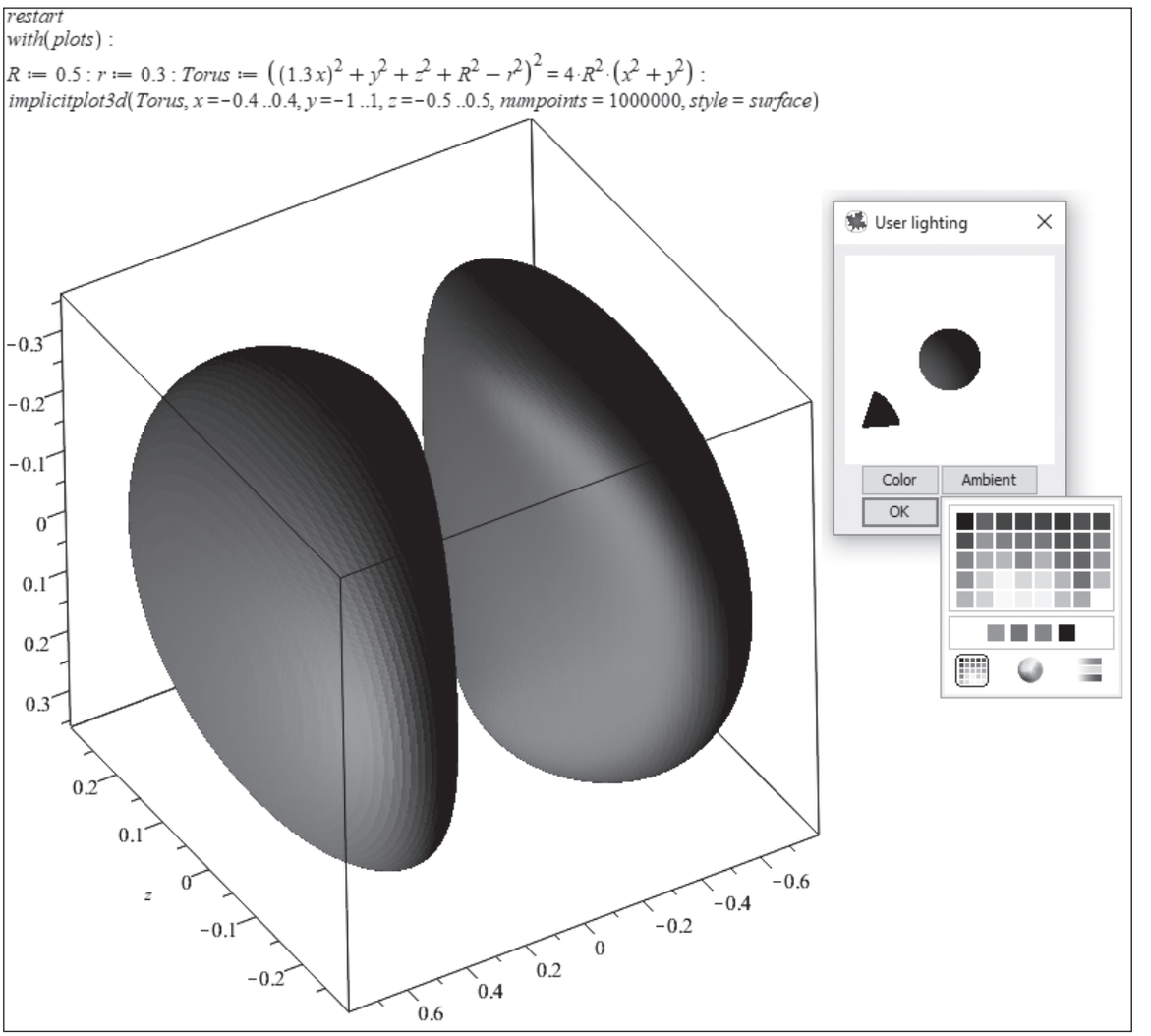


Рис. 6. Построение поверхности почки/легкого с помощью пакета Maple

рованного и разорванного на две половинки тора*: у аргумента x, возводимого в квадрат, стоит не единица (тор), а константа 1,3. Будем считать, что это у нас... две человеческие почки или две половинки легких. Можно в формулах замкнутых поверхностей (тора, шара, эллипсоида и т. д.) вводить дополнительные коэффициенты и получать другие интересные замкнутые и разомкнутые поверхности, похожие на человеческие органы — печень, желудок, селезенку и т. д. Подход, описанный в этой статье, позволяет выполнять и эту интересную работу.

На рисунке 6 также отображено диалоговое окно освещения (lighting) объемной конструкции: можно мышкой двигать фонарь с заданным цветом (см. конус в диалоговом окне освещения — *User lighting*) и добиваться максимальной выразительности трехмерного объекта, повышения его объемности. Еще для этого используют также туман, перспективу и другие инструменты форматирования трехмерной графики.

На сайте статьи можно увидеть анимацию трансформации тора в две почки (в две половинки легких) при плавном изменении коэффициента при аргументе x с единицы (тор) до значения 1,4 (почки).

Но трехмерная графика математических пакетов — это скорее рекламная штучка, нежели серьезный инструмент математического моделирования. Изображения**, подобные тому, которое показано на рисунке 6, раньше помещали на коробках с дистрибутивом математических пакетов, а сейчас — на сайтах фирм-разработчиков. Если нужно проектировать сложные трехмерные конструкции, то работают со специальными программами САПР, а не с математическими пакетами. Если же нужно отобразить какой-то математический объект (поверхность или геометрическое тело), то трех координат здесь часто не хватает — см. выше рисунок 4. Кроме того, мы работаем с плоским экраном дисплея и плоским листом бумаги принтера. Поэтому-то пакет Mathcad и не имеет встроенной функции, подобной той, какая показана на рисунке 6. Для практических инженерных расчетов объемная графика особо не нужна. Поэтому ее возможности были резко сокращены в новой версии Mathcad — в Mathcad Prime.

Тем не менее наш тор, превращенный в две почки или половинки легких, можно построить и в среде Mathcad методом, описанным в [2], — заполнением плоскости или объема точками, координаты которых удовлетворяют неким условиям.

На рисунке 7 показано, как наши почки/легкие можно построить в среде Mathcad, дополнительно поместив в них маленький шар, который будет имитировать почечный камень или некое затемнение в легких, диагностируемое с помощью... томографа.

В параллелепипед, ограниченный координатами от –0,4 до 0,4 по оси x, от –0,8 до 0,8 по оси y и от –0,3 до 0,3 по оси z, бросаются n случайных точек. Те из них, которые формируют поверхность почек/легких и объем

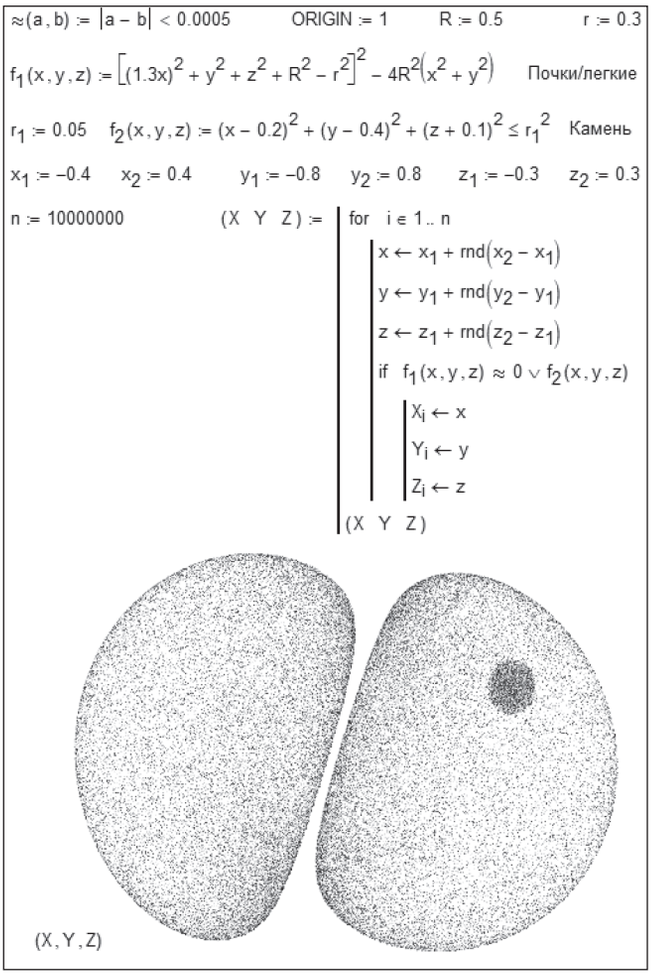


Рис. 7. Поточечное построение «почек/легких» с «камнем/затемнением» внутри с помощью пакета Mathcad

камня в почке (шара), запоминаются в трех векторах X, Y и Z. Которые затем отображаются на трехмерном графике. Генерацию случайных точек выполняет встроенная в Mathcad функция *rnd*, возвращающая случайное число в диапазоне от нуля до значения, заданного аргументом этой функции. Для поточечного формирования поверхности используется оператор «примерно равно» (≈; он задается первым оператором на рисунке 7), а для формирования геометрического тела — оператор «меньше или равно» (≤). Полученные конструкции (почки с камнем или легкие с затемнением) показаны в нижней части рисунка 7.

Но вернемся к медицине. Есть два способа рентгеновского исследования внутренних органов человека: рентгенография (получение рентгеновских снимков) и рентгеноскопия (рентгеновское просвечивание), когда изображение объекта получают не на пленке, а на светящемся (флуоресцентном) экране. Перед таким экраном в старые времена мог сидеть врач и руками поворачивать торс больного, ища очаги туберкулеза легких, например. Руки такого врача при этом были защищены толстыми перчатками, изготовленными из специальной резины с добавкой свинца, а на тело накинут тяжелый фартук из такой же резины. Ведь при таком исследовании опасному для здоровья ионизирующему облучению подвергался не только пациент, но и сам врач. Мы же можем безопасно вращать наши «почки/легкие», показанные на рисунке 7, всего лишь одной рукой с помощью компьютерной мыш-

* Геометрическое тело тор тут оказалось не случайно. Пациента при томографии пропускают через большое устройство в виде тора, в центре которого фокусируются либо рентгеновские лучи (КТ), либо электромагнитное поле (МРТ). Сквозь этот тор пропускают человека, лежащего на специальной движущейся кушетке.

** Авторы показывали рисунок 6 своим коллегам и просили сказать, на что это похоже. Называли кофейные зерна, кедровые орешки и др., но никто не назвал это почками или легкими человека. Тем не менее мы будем считать это почками, внутри одной из которых образовался камень, или легкими с затемнением.

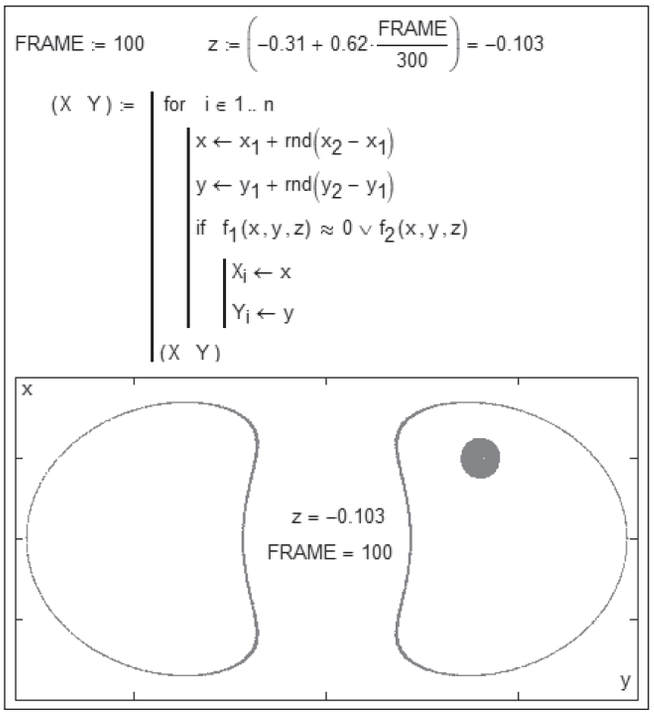


Рис. 8. Получение среза почек/легких с помощью пакета Mathcad

ки. Правда, раньше, когда в дисплеях были телевизионные трубки, тоже был риск небольшого ионизирующего облучения. Теперь же мы используем более безопасные плоские дисплеи другого типа.

Но наши почки/легкие допустимо увидеть более четко с помощью... *томографии*! Изображение на рисунке 7, повторяем, можно считать неким нечетким рентгеновским снимком, где изображение одних структур накладывается на изображения других. Медицинская томография для того и была придумана, чтобы повысить четкость и, следовательно, информативность медицинского исследования.

На рисунке 8 отображены уже описанные вычислительные действия, но не в объеме (рис. 7), а на плоскости. Для этого фиксируется значение переменной z (вертикальная ось), которое связывается с системой переменной *FRAME*, отвечающей в среде Mathcad за анимацию (счетчик кадров анимации [1]). У нас будет 300 кадров — 300 сечений почек/легких. Меняя значение переменной z, можно получить томограмму (сечения) нашего трехмерного объекта (рис. 9).

Двигая бегунок видеоплеера, показанного на рисунке 9, можно фиксировать аномалии в наших почках/легких.

Кстати, в Интернете по запросу «формула формы почки» ничего не было найдено. Поэтому-то авторам и пришлось деформировать и рвать на две половинки тор. Но по запросу «формула формы сердца» было найдено неявное алгебраическое выражение с переменными x, y и z, по которому можно построить сердце — не в анатомическом, а в «романтическом» виде (рис. 10). Но и наша статья описывает томографию в некоем «романтическом», научно-популярном стиле!

Было бы интересно найти в Интернете или вывести самому формулы и для других человеческих органов — желудка, печени, селезенки и т. д. Их можно было бы поместить в некоем объеме человека и проводить его исследование на томографе. Вернее, упрощенно моделировать это исследование на уроке информатики

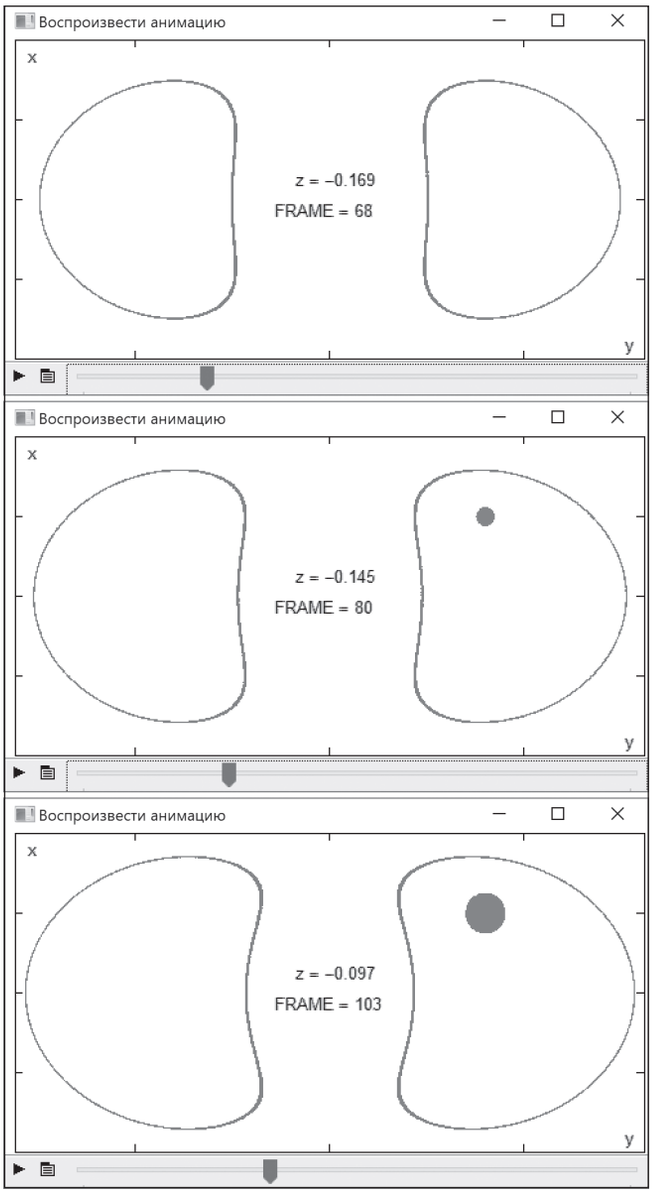


Рис. 9. Кадры томограммы почек с камнем внутри (легких с затемнением)

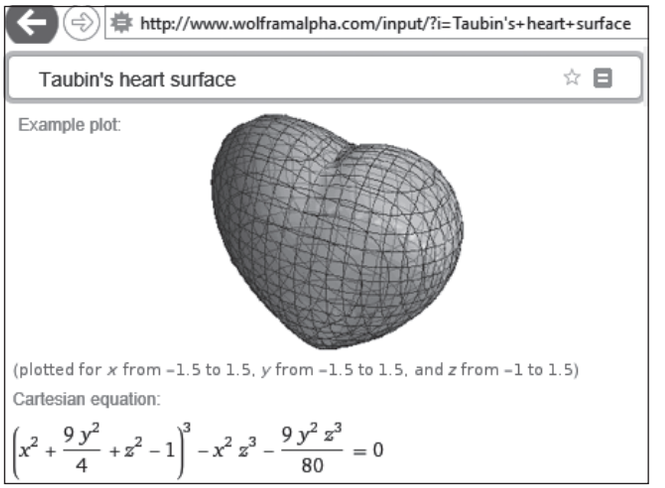


Рис. 10. Построение поверхности «сердце» с помощью сайта WolframAlpha (http://www.wolframalpha.com/input/?i=Taubin's+heart+surface)

с междисциплинарными связями с физикой, математикой и биологией.

Сайт статьи с цветными рисунками и анимациями: https://community.ptc.com/t5/PTC-Mathcad-Blog/Tomography/ba-p/483596

Список использованных источников

1. *Очков В. Ф.* Живые кинематические схемы в Mathcad // Открытое образование. 2013. № 3. http://twt.mpei.ac.ru/ochkov/Mathcad-15/kinematic.html

2. *Очков В. Ф., Диаз Фалькони А.* Информатика, алгебра, геометрия: четыре арифметические кривые с покемоном // Информатика в школе. 2016. № 9. http://twt.mpei.ac.ru/ochkov/4-curves.pdf

3. *Очков В. Ф., Калова Я., Никульчев Е. В.* Оптимизированный фрактал или ФМИ // Cloud of Science. 2015. Т. 2. № 4. http://twt.mpei.ac.ru/ochkov/Opt_Fractal.pdf

4. *Kwan S. P.* Learning and Teaching Mathematics with GeoGebra and Microsoft Excel // The Asian Conference on Education 2012. Osaka, Japan. http://iafor.org/archives/offprints/ace2012-offprints/ACE2012_0674.pdf

НОВОСТИ

Стоматологический робот впервые произвел операцию самостоятельно

Спроектированный в Китае робот-стоматолог успешно провел автономную операцию. Пациентке были установлены два искусственных зуба, распечатанных на трехмерном принтере. Вся процедура заняла один час. Хотя все это время в операционной находились хирурги, они не принимали активного участия и не брали на себя управления роботом. Преимуществом роботизированного подхода является точность, недоступная при использовании других методов. Кроме человеческого фактора, при работе в ротовой полости хирургу могут помешать плохой обзор или недостаточное освещение. В данном случае эти факторы не имели значения, и погрешность при проведении операции составила лишь 0,3 мм.

Google и Ford заставят интернет-зависимых водителей пережить виртуальное ДТП

Ford в партнерстве со студией виртуальной реальности Happy Finish и Google запустили приложение Ford Reality Check («Проверь на практике»), погружающее интернет-зависимых водителей в ситуацию, когда отвлечение за рулем приводит к трагическим последствиям. Приложение с помощью технологии виртуальной реальности Google Daydream VR помещает пользователя на место водителя, заезжающего за друзьями по дороге на вечеринку. За внимание водителя конкурируют сообщения в мессенджере, телефонные звонки и сами пассажиры, что приводит к нескольким аварийным ситуациям. По итогам первых испытаний приложения 90 % пользователей заявили, что они поменяют свое поведение за рулем после испытанного в виртуальной реальности. Синдром боязни упустить что-то важное подвергает риску молодых водителей. Согласно последним данным, на дорогах Европы в ДТП каждый год погибают около 4 тыс. молодых людей, и две трети из них — водители.

Смартфоны превращаются в ключи от автомобилей

Компания Bosch разработала цифровую систему доступа к автомобилю Perfectly Keyless, которая подразумевает, что водители смогут обходиться без обычных ключей. Как только водители приближаются к своим автомобилям, их смартфоны, подключенные к системе Perfectly Keyless, идентифицируются с помощью встроенных в машину датчиков, после чего автомобиль открывается. По тому же принципу без ключа можно будет запустить двигатель или запереть автомобиль в конце пути. Как только автомобиль закрывается, активизируются заранее установленные индивидуальные настройки. Когда в конце пути водитель покидает автомобиль, система продолжает виртуально следить за смартфоном. В момент, когда водитель с телефоном отходит от автомобиля дальше чем на два метра, машина автоматически и безопасно закрывается, система посылает уведомление на смартфон водителя.

В МТИ научили дроны искать на складе предметы с RFID-метками

Система, разработанная в МТИ, позволяет небольшим летательным аппаратам считывать RFID-метки с расстояния в несколько десятков метров. При этом система способна определить положение метки с точностью до 20 см. Главной проблемой при разработке системы оказалось то, что те дроны, которым по соображениям безопасности разрешено летать в складских помещениях, обладают слишком малой грузоподъемностью и на них нельзя поставить сканеры RFID с дальностью считывания больше нескольких сантиметров. Исследователи решили проблему, установив на дронах ретрансляторы сигналов стандартных сканеров. Для этого пришлось, в свою очередь, разработать алгоритмы разделения сигналов, передающихся на одной частоте. Кроме того, дрон не может нести на себе антенную решетку, необходимую для определения положения метки. Зато он может перемещаться сам и по сдвигу фазы сигнала вычислять направление на передатчик.

(По материалам международного компьютерного еженедельника «Computerworld Россия»)

Варвара Клюева,
ученица средней общеобразовательной школы № 4 имени Героя Советского Союза Д. П. Левина, г. о. Сызрань, Самарская область

СОЗДАНИЕ АНИМИРОВАННОЙ ОТКРЫТКИ-МОТИВАТОРА НА ЭКОЛОГИЧЕСКУЮ ТЕМУ*

Создание компьютерной анимации представляет интерес потому, что это очень творческое и полезное занятие. Когда мультфильм получается именно так, как было задумано по сценарию, то получаешь большое удовлетворение от своей работы. Появляется уверенность в своих силах.

Анимацию можно создавать в различных программах, на онлайн-сервисах, путем фотографирования отдельных кадров. Большие возможности для создания анимации представляют языки программирования. И среди них язык Scratch. В программе Scratch можно создать двумерную анимацию и мультфильмы. На ней и остановимся.

Даже самая простая анимация создается по сценарию. Значит, необходимо придумать сценарий. Год экологии... Обращаем внимание на существующие экологические проблемы: загрязнение рек, вырубка леса, истребление животных. Грустная тема. Пожалуй, красота + экология будет лучше. Есть идея — анимированная открытка-мотиватор!

К идее подходит такой сценарий. Ночь, постепенно светлеет, наступает утро. Утром просыпаются цветы и насекомые. Что может быть красивее нашей планеты на рассвете! Раскрылись ромашки на поле, летают бабочки над цветами. Красота! Не забудем напомнить, что эту красоту необходимо беречь, — появляется соответствующая надпись. И все это на фоне красивой музыки или песни. Песня «Не рвите цветы» будет в тему.

Сценарий готов, теперь воплощаем идею в программе Scratch. Рисуем фоны во встроенном в программу графическом редакторе (рис. 1).

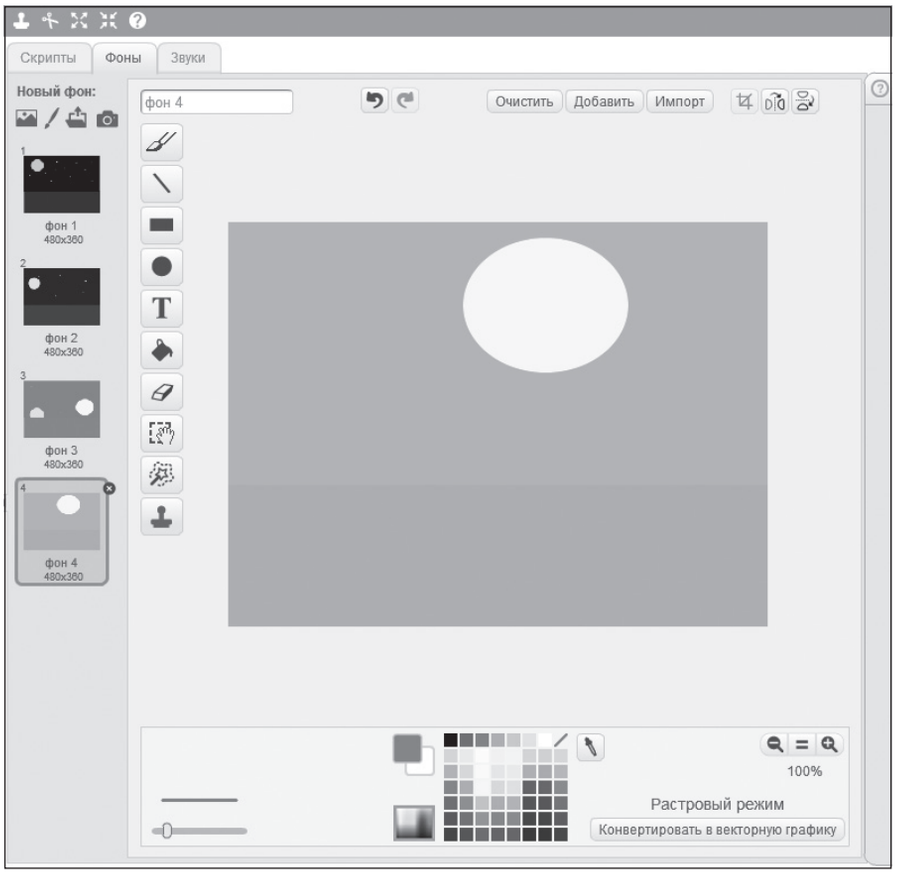


Рис. 1. Создание фонов

* Материалы к статье можно скачать на сайте ИНФО: http://infojournal.ru/journals/school/school_10-2017/

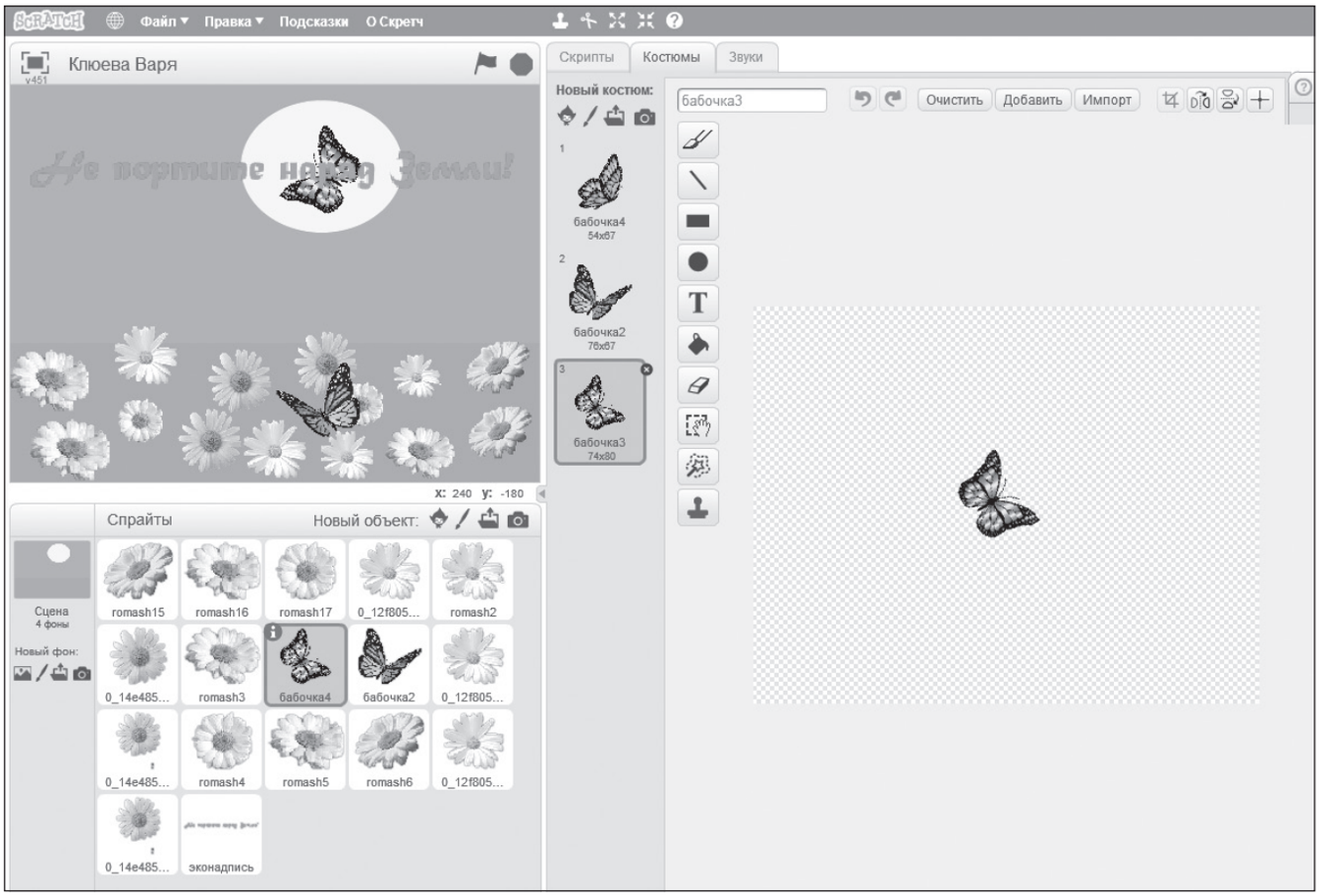


Рис. 2. Размещение объектов на сцене

Ищем в Яндекс-библиотеке изображения ромашек и бабочек в фазе полета. Размещаем ромашки и бабочки на сцене. Причем у бабочек будет три костюма, чтобы они летали как можно более реалистично. Создаем красивую надпись в онлайн-генераторе надписей и тоже размещаем ее на сцене (рис. 2).

Фиксируем полет бабочек горизонтально, чтобы кверху брюшком они не летали (рис. 3).

А дальше — программируем. Сцена будет менять свои костюмы через определенное время и проигрывать фоновую музыку (рис. 4).

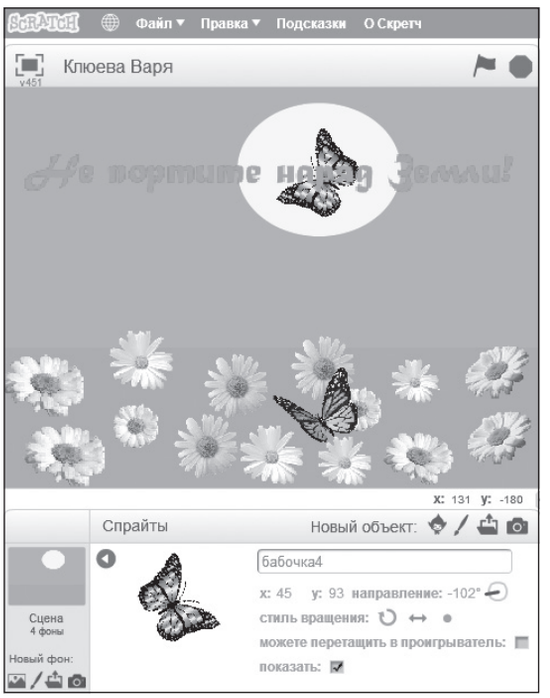


Рис. 3. Стиль вращения — горизонтальный

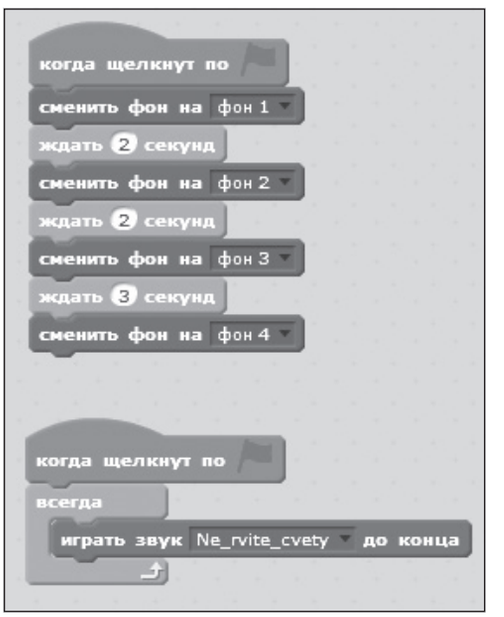


Рис. 4. Скрипт для сцены

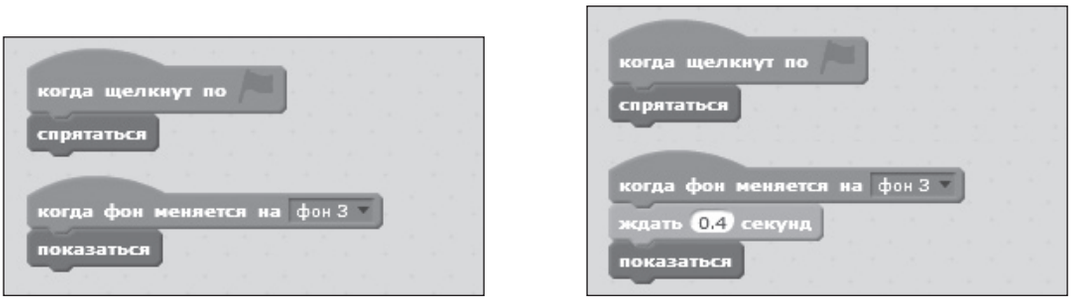


Рис. 5. Скрипты ромашек

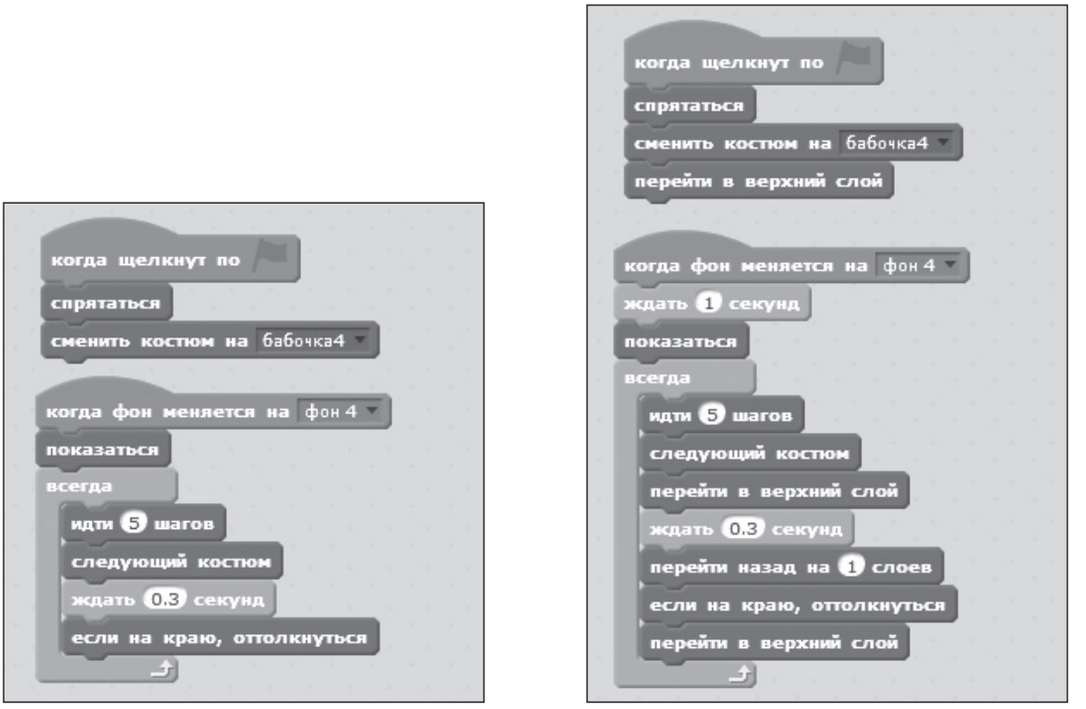


Рис. 6. Скрипты бабочек

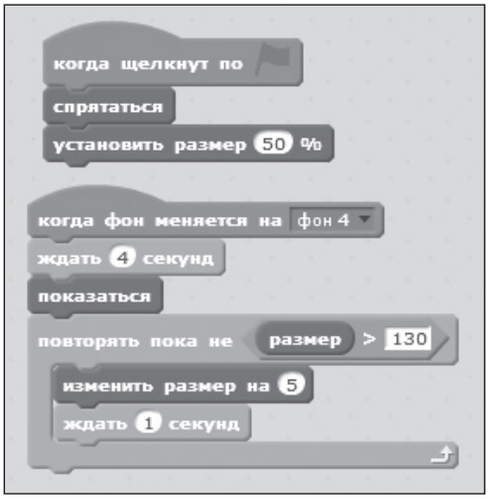


Рис. 7. Скрипт надписи

Ромашки появятся в определенное время, значит, у них скрипты будут немного различаться (рис. 5).

Скрипты бабочек немного различаются, бабочки могут летать между ромашками (рис. 6).

Надпись будет постепенно увеличиваться при появлении (рис. 7).

Снимаем видео программой записи видео с экрана и еще сохраняем файл в формате swf в программе ScratchFlash.

Анимированная открытка-мотиватор готова! Если после просмотра открытки кто-то задумается, прежде чем сорвать цветок, о сохранении красоты природы, значит, работа была сделана не зря.

Готовую открытку можно посмотреть по ссылке: https://youtu.be/A_PMYghxqhU

Михаил Мусин,
ученик средней школы № 3, г. Актобе, Актюбинская область, Республика Казахстан

МУЛЬТФИЛЬМ ПО МОТИВАМ КАЗАХСКОЙ НАРОДНОЙ СКАЗКИ «ЛИСА И КОЗА»*

На уроках информатики в шестом классе изучают программу PowerPoint из пакета Microsoft Office. С помощью этой программы создается файл презентации, который содержит набор слайдов. PowerPoint предоставляет пользователю большое количество шаблонов презентаций на различные темы. Такие шаблоны содержат слайды, оформленные определенным образом. Возможности данной программы огромны, и невозможно изучить все ее функции на нескольких уроках информатики в школе.

Чтобы лучше изучить возможности PowerPoint и узнать, что еще можно создавать в ней кроме презентаций, я решил провести исследование.

Очень мало людей знают все возможности программы PowerPoint и умеют создавать такие презентации, которыми можно любоваться. В основном люди создают слайды, содержащие только текстовую информацию, — в них нет ни картинок, ни анимации, ни переходов, ни других «фишек» программы. Но сейчас, когда без презентации не обходится ни одно выступление, важно уметь создавать правильную презентацию, так как презентация — это лицо вашей работы. Знание всех возможностей данной программы позволит в будущем ученикам готовить высококлассное сопровождение своих докладов и выступлений.

А мультфильм, созданный по мотивам казахской народной сказки «Түлкі мен ешкі» («Лиса и коза»), способствует воспитанию патриотических чувств у детей, гордости за родной язык и родной народ. Мультфильм может быть использован на уроках казахской литературы, а также на уроках информатики при изучении PowerPoint.

Смотреть мультфильмы любят практически все дети. Некоторые малыши, замерев от восторга, могут часами наблюдать за приключениями мультипликационных героев. Мультфильмы оказывают большое влияние на развитие и мировоззрение детей. Однако родителям нужно следить за тем, что смотрит их ребенок. Есть весьма поучительные мультики, которые подсказывают ребенку, как лучше поступить в той или иной ситуации, рассказывают ему о том, что нехорошо дразниться, ябедничать, жадничать. Такие анимационные фильмы помогают родителям в воспитании и дают детям жизненный урок на примере персонажей.

Малышам лучше начать знакомство с миром мультипликации с разных сказочных мультиков. Это добрые и наивные истории о зверюшках, а также поучительные народные сказки. Такие мультфильмы учат малышей дружбе, вежливости, доброте и состраданию, а также воспитывают патриотические чувства, любовь к своему языку и своему народу.

В ходе моего исследования был проведен социологический опрос, в результате которого была получена информация о наиболее популярных мультфильмах, которые смотрят сейчас дети. Всего в опросе участвовали 120 человек, которым были заданы следующие вопросы:

- Самый любимый мультфильм?
- Мультфильм, где ты больше всего смеялся?
- Мультфильм, где тебе захотелось плакать?
- Любимый мультяшный герой?

Результаты проведенного опроса представлены ниже.

«Мой любимый мультфильм»:
- «Черепашки ниндзя 2» — 35 %.
- «Звездные войны: повстанцы» — 20 %;
- «Леди Баг и СуперКот» — 20 %;
- «Монстры на каникулах» — 10 %;
- «Король Лев» — 15 %.

«Мультфильм, где я больше всего смеялся»:
- «Миньоны» — 60 %;
- «Город героев» — 20 %;
- «Ледниковый период» — 15 %;
- «Черепашки ниндзя 2» — 5 %.

«Мультфильм, где мне захотелось плакать»:
- «Король Лев» — 50 %;
- «Русалочка» — 25 %;
- «Бэмби» — 20 %;
- «Черепашки ниндзя 2» — 5 %.

«Мой любимый герой из мультфильма»:
- Маринетт Дюпен-Чен — 30 %;
- Мэйбл Пайнс — 25 %;
- Сабина — 20 %;
- Черепашка Лео — 20 %;
- Стюард — 5 %.

Можно сделать следующие выводы: все участники опроса в основном положительно относятся к мультфильмам, и все смотрят зарубежные мультфильмы. Отечественные мультфильмы дети смотрят мало. Поэтому я счел актуальным выбор казахской народной сказки «Түлкі мен ешкі» для создания своего мультфильма. Ведь именно мультфильмы по сюжетам национальных сказок воспитывают такие качества, как взаимопомощь, милосердие, любовь к миру, трудолюбие и любовь к своему народу.

Экспериментальным путем мне удалось оживить персонажей мультфильма, применив эффекты анимации в среде PowerPoint.

Для этого были выполнены следующие шаги.

Шаг 1. Выбор сценария мультфильма.

Я выбрал небольшую по содержанию казахскую народную сказку (рис. 1).

Шаг 2. Создание рисунков (кадров).

1) Открыл графический редактор Paint.
2) Создал рисунок (1-й кадр), сохранил.
3) Скопировал 1-й кадр и вставил в программу Paint.

* Материалы к статье можно скачать на сайте ИНФО: http://infojournal.ru/journals/school/school_10-2017/

Рис. 1

4) Произвел необходимые изменения в рисунке (2-й кадр), сохранил.
5) Выполнял п. 2–4 до тех пор, пока задуманный сценарий не был завершен.

На рисунках 2–5 продемонстрированы кадры сказки.

Шаг 3. Сборка мультфильма.

1) Запустил PowerPoint, выполнив команду *Пуск*, *Программы*, *Microsoft PowerPoint*.
2) Оформил титульный слайд (рис. 6).
3) Выполнил команду *Вставка*, *Слайд*.
4) Выполнил команду *Вставка, Рисунок* (кадр 1).
5) Вставил все рисунки (кадры).
6) Оформил заключительный слайд.

Шаг 4. Организация смены слайдов (кадров).

1) На вкладке *Анимация* в разделе *Переход к этому слайду* выбрал эффект смены слайдов.
2) Установил скорость смены слайдов, щелкнув стрелку около кнопки *Скорость перехода*, а затем выбрал нужную скорость.
3) Выбрал способ смены слайдов *Автоматически после 1 секунды*, для некоторых слайдов — 4 с.
4) Для подтверждения выбранных параметров нажал кнопку *Применить ко всем*.

Шаг 5. Просмотр мультфильма.

На вкладке *Показ слайдов* в разделе *Начать показ слайдов* выбрал *С начала*.

Шаг 6. Сохранение мультфильма.

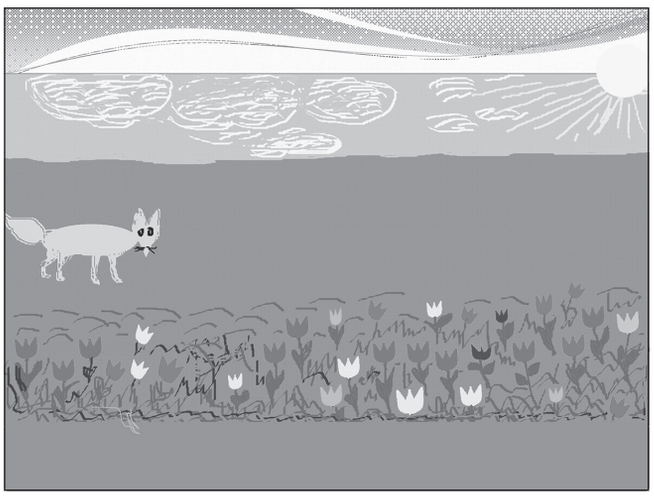


Рис. 2

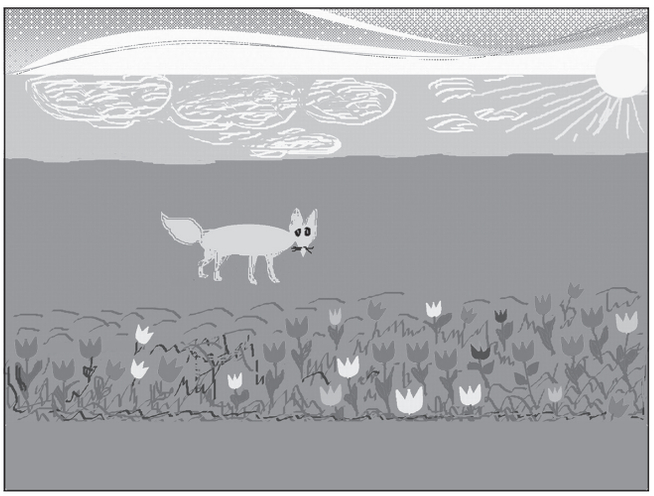


Рис. 3

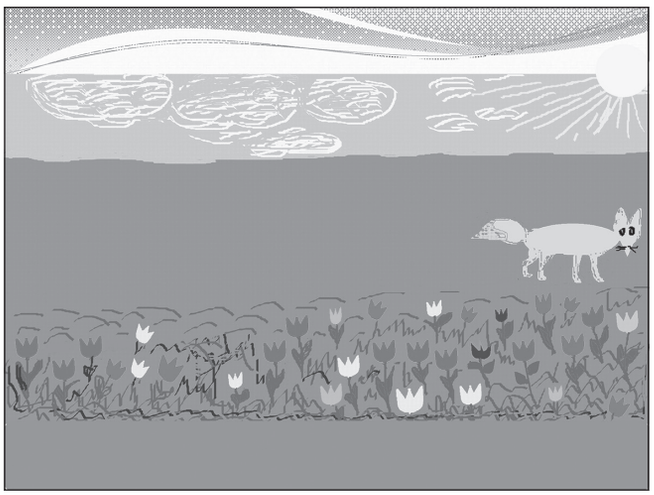


Рис. 4

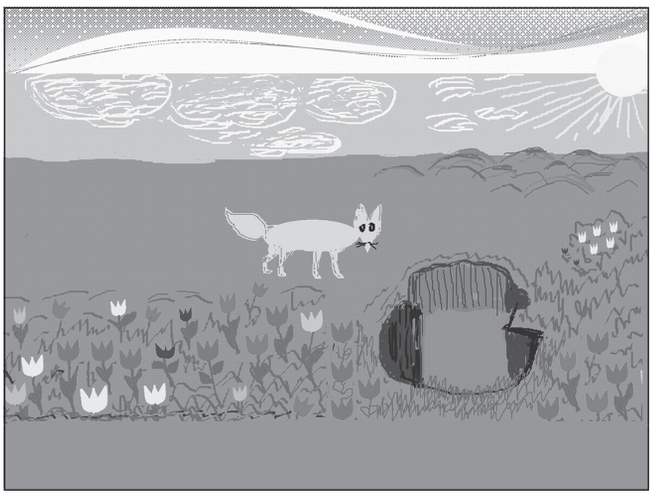


Рис. 5

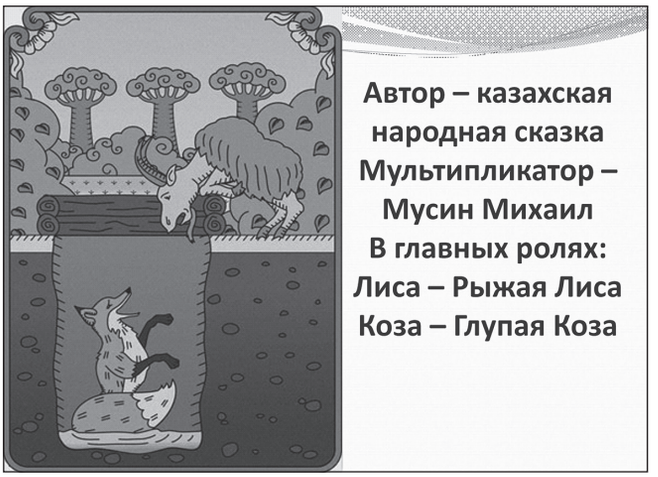


Рис. 6

Итог проделанных шагов — мультфильм «Түлкі мен ешкі». В мультфильме организована непрерывная циклическая смена слайдов таким образом, что создается ощущение движения лисы и козы. В мультфильме присутствует звук.

Результаты проведенной работы по исследованию возможностей программы MS PowerPoint позволяют сделать вывод о том, что функциональные возможности этой программы разрешают создать небольшой мультфильм. Необходимо лишь выполнить все предложенные шаги по порядку и проявить свою фантазию и творческие способности.

Для настройки параметров анимации я использовал обычный режим и выполнил следующие действия:

1) Скомпоновал на слайде все объекты, подлежащие анимации.
2) Выбрал команду *Показ слайдов*, *Настройка анимации*.

Диалоговое окно включает в себя четыре вкладки: *Порядок и время*, *Видоизменение*, *Видоизменение в диаграмме*, *Настройка мультимедиа*.

Представленные на вкладке *Порядок и время* параметры позволяют изменить последовательность появления анимированных объектов на экране. Для этого нужно выделить объект и щелкнуть на кнопке со стрелкой вверх или вниз. В соответствии с этим изменится и положение объекта в списке. Кроме того, можно установить, будут ли события происходить автоматически через определенное время или же по щелчку. Чтобы события происходили одно за другим без задержки, можно установить переключатель *Автоматически*, оставив без изменений установленное время — 00:00. Именно это время позволяет кадрам меняться с такой скоростью, что движение лисы кажется непрерывным.

При создании мультфильма в программе PowerPoint можно изучить различные возможности программы:

- организовывать непрерывную циклическую демонстрацию презентации;
- добавлять в презентацию звук таким образом, чтобы голос звучал непрерывно на протяжении всей презентации;
- организовывать непрерывную смену слайдов по определенно заданному времени;
- вставлять в презентацию не только текст, но и рисунки из других файлов. В пятом классе мы изучали графический редактор Paint, и я научился вставлять рисунки, созданные в этом редакторе, в свою презентацию. Оказалось, что если нарисовать множество рисунков в Paint, т. е. отдельных кадров, а потом организовать быструю смену кадров, то можно добиться эффекта движения персонажей;
- осуществлять ввод и форматирование текста презентации;
- использовать графику и специальные эффекты.

По результатам создания мультфильма можно сказать, что программа Microsoft PowerPoint является простым в освоении и очень мощным инструментом для создания привлекательных презентаций.

Список использованных источников

1. *Аромаштам М.* Дети смотрят мультфильмы: психолого-педагогические заметки. Практика производства мультфильмов в детском саду. М.: Чистые пруды, 2006.

2. *Куклина И. Д.* Создание динамических объектов в программе PowerPoint 2010 // Информатика. 2016. № 03 (685).

3. *Лазарев Д. И.* Презентация: Лучше один раз увидеть! М.: Альпина Бизнес Букс, 2009.

4. *Лоу Д.* PowerPoint 2010 для чайников. М.: Диалектика, 2011.

5. *Можаров Р. В., Можарова Н. Р., Евтеев В. В., Кузьменко О. А., Шевченко М. О.* Программное обеспечение персональных компьютеров. М.: Финстатинформ, 2014.

6. *Фролов М. И.* Учимся анимации на компьютере. Самоучитель для детей и родителей. М.: Лаборатория Базовых Знаний, 2012.

НАПЕЧАТАНО В 2017 ГОДУ

НАПЕЧАТАНО В 2017 ГОДУ

НАМ 15 ЛЕТ

МЕТОДИКА ОБУЧЕНИЯ ИНФОРМАТИКЕ: ОПЫТ МПГУ

КОНКУРС ИНФО-2016

УРОКИ ИНФОРМАТИКИ

ИНТЕГРИРОВАННЫЕ УРОКИ

МЕТОДИЧЕСКАЯ КОПИЛКА

СУПЕРКОМПЬЮТЕРНОЕ ОБРАЗОВАНИЕ В ШКОЛЕ

ИТОГОВАЯ АТТЕСТАЦИЯ ПО ИНФОРМАТИКЕ

ЗАДАЧИ

ОЛИМПИАДЫ ПО ИНФОРМАТИКЕ

ИНФОРМАТИКА И ИКТ В НАЧАЛЬНОЙ ШКОЛЕ

ЗАНИМАТЕЛЬНАЯ ИНФОРМАТИКА

ИНФОРМАТИКА ГЛАЗАМИ ШКОЛЬНИКА

КОНКУРСЫ

ПРАВИЛА ДЛЯ АВТОРОВ

Общие положения

Все присланные статьи рецензируются. Публикация статей возможна только при наличии положительного отзыва рецензентов.

Поскольку рецензирование и предпечатная подготовка материалов занимают не менее трех месяцев, статьи следует присылать в редакцию заблаговременно.

Редакция не берет платы за публикацию рукописей аспирантов.

Требования к файлам рукописи

1. Текст статьи должен быть представлен в формате текстового редактора Microsoft Word (*.doc, *.rtf):
   - формат листа — А4;
   - все поля по 2 см;
   - шрифт — Times New Roman, кегль — 12 пт, расстояние между строками — 1,5 (полтора) интервала;
   - графические материалы вставлены в текст.

2. Файл со статьей должен содержать следующие данные для публикации, **необходимо строго придерживаться указанной ниже последовательности** (пожалуйста, проверяйте оформление по образцу статьи, представленному на сайте ИНФО):
   - **И. О. Фамилия** автора(ов) на русском языке.
   - **Место работы** автора(ов) на русском языке. Необходимо указать место работы **каждого** автора. Если из названия организации не следует принадлежность к населенному пункту, через запятую надо указать название населенного пункта.
   - **Название статьи** на русском языке.
   - **Аннотация** на русском языке (3–5 строк в указанном выше формате).
   - **Ключевые слова** на русском языке (не более 10, через запятую).
   - **Подробная информация об авторах** — для каждого из авторов:
     – фамилия, имя, отчество (полностью);
     – ученая степень;
     – ученое звание;
     – должность;
     – место работы;
     – адрес места работы (обязательно с индексом);
     – рабочий телефон (обязательно с кодом города);
     – адрес электронной почты (e-mail).
   - **И. О. Фамилия** автора(ов) на английском языке.
   - **Место работы** автора(ов) на английском языке.
   - **Название статьи** на английском языке.
   - **Аннотация** на английском языке.
   - **Ключевые слова** на английском языке (через запятую).
   - **Текст статьи** в указанном выше формате.
   - **Список литературных и интернет-источников,** упорядоченный в алфавитном порядке.

3. При отправке статьи в редакцию в полях электронной формы необходимо указать подробные сведения об авторе:
   – фамилия, имя, отчество (полностью);
   – домашний почтовый адрес (с индексом);
   – домашний телефон (обязательно с кодом города);
   – мобильный телефон;
   – адрес электронной почты (e-mail).

Данные сведения необходимы для оперативной связи с автором статьи и пересылки авторского экземпляра журнала и **не подлежат публикации**.

Если авторов несколько, необходимо представить указанные сведения **обо всех авторах**.

4. При необходимости статья может сопровождаться дополнительным материалом в электронном виде (презентации, листинги программ, книги Excel, примеры выполнения работ и др.), который будет размещен на сайте ИНФО.

5. Иллюстрации следует представлять в виде отдельных графических файлов (даже при их наличии в документе Word) в формате TIFF или JPG, разрешение — не менее 300 пикселей на дюйм.

Уважаемые коллеги!

Статьи для публикации в журналах «Информатика и образование» и «Информатика в школе» должны отправляться в редакцию **только через электронную форму на сайте ИНФО (раздел «Авторам → Отправка статьи»)**:

http://infojournal.ru/authors/send-article/

Обращаем ваше внимание, что для отправки статьи необходимо предварительно зарегистрироваться на сайте ИНФО (или авторизоваться — для зарегистрированных пользователей).

С требованиями к оформлению представляемых для публикации материалов можно ознакомиться на сайте ИНФО в разделе **«Авторам»**:

http://infojournal.ru/authors/

Дополнительную информацию можно получить в разделе **«Авторам → Часто задаваемые вопросы»**:

http://infojournal.ru/authors/faq/

а также в редакции ИНФО:

e-mail: readinfo@infojournal.ru

телефон: (495) 140-19-86

Журнал «Информатика в школе»

Индексы подписки (агентство «Роспечать») на 1-е полугодие 2018 года

- **81407 — для индивидуальных подписчиков**
- **81408 — для организаций**

Периодичность выхода: 5 номеров в полугодие (в январе не выходит)

Редакционная стоимость:
индивидуальная подписка — 150 руб.
подписка для организаций — 300 руб.

Федеральное государственное унитарное предприятие "Почта России" Ф СП - 1

Бланк заказа периодических изданий

АБОНЕМЕНТ На ~~газету~~ журнал _____ (индекс издания)

Информатика в школе (наименование издания) Количество комплектов _____

На 20**18** год по месяцам

1	2	3	4	5	6	7	8	9	10	11	12

Куда _____ (почтовый индекс) _____ (адрес)

Кому _____

Линия отреза

			ДОСТАВОЧНАЯ КАРТОЧКА	_____ (индекс издания)
ПВ	место	литер		

На ~~газету~~ журнал **Информатика в школе** (наименование издания)

Стоимость			Количество комплектов
	подписки	_____ руб.	
	каталожная	_____ руб.	
	переадресовки	_____ руб.	

На 20**18** год по месяцам

1	2	3	4	5	6	7	8	9	10	11	12

			Город
почтовый индекс			село
			область
			Район
код улицы			улица
дом	корпус	квартира	Фамилия И.О.