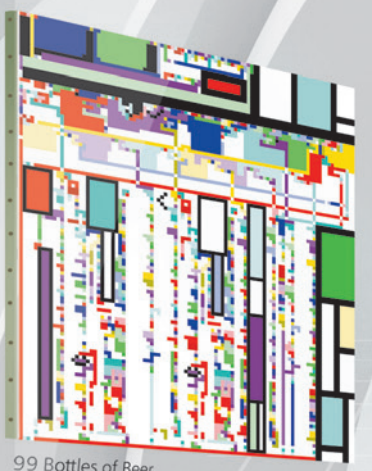
№9
Издание основано в 1995 г.
inf.1september.ru
УЧЕБНО-МЕТОДИЧЕСКИЙ ЖУРНАЛ ДЛЯ УЧИТЕЛЕЙ ИНФОРМАТИКИ

ИНФОРМАТИКА

Piet-program printing "Piet", codel size 6, Thomas Schoch, 2006

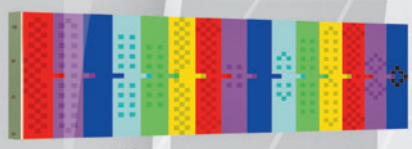


99 Bottles of Beer

Hello World

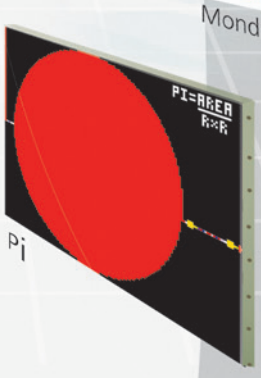
"Композиция с красным, желтым и синим ", Пит Мондриан, 1921

Mondrian Hello World



Artistic Hello World4

Artistic Hello World4



Pi

внутри номера CD и код доступа к электронной версии

издательский дом 1september.ru Первое сентября | октябрь 2012

ИНФОРМАТИКА Подписка: «Роспечать» – 32291 (бумажная версия), 19179 (электронная); «Почта России» – 79066 (бумажная версия), 12684 (электронная)

НА ОБЛОЖКЕ

Среди всех работ Пита Мондриана, представленных на обложке, только одна действительно принадлежит кисти художника. Остальные изображения являются программами на языке Piet, которому посвящена первая статья номера. Эзотерические языки программирования не слишком известны, но среди них встречаются не только "приколы", но и вполне серьезные инструменты. И уж во всяком случае эти языки впечатляют и удивляют. Удивляйтесь!

В НОМЕРЕ

НА ДИСКЕ

ЭЛЕКТРОННЫЕ МАТЕРИАЛЫ:

- Файлы к статье о языке Piet
- Файлы к статье о задаче С4 демонстрационного варианта ЕГЭ
- Исполнители и методические материалы к статье о новом курсе "Азбука Роботландии"
- Презентации к статьям номера

ИНФОРМАТИКА

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

http://inf.1september.ru

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

РЕДАКЦИЯ:
гл. редактор С.Л. Островский
редакторы
Е.В. Андреева,
Д.М. Златопольский (редактор вкладки "В мир информатики")
Дизайн макета И.Е. Лукьянов
верстка Н.И. Пронская
корректор Е.Л. Володина
секретарь Н.П. Медведева
Фото: фотобанк Shutterstock
Журнал распространяется по подписке
Цена свободная
Тираж 12 737 экз.
Тел. редакции: (499) 249-48-96
E-mail: inf@1september.ru
http://inf.1september.ru

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

ГАЗЕТА ИЗДАТЕЛЬСКОГО ДОМА
Первое сентября – Е.Бирюкова

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

УЧРЕДИТЕЛЬ: ООО "ЧИСТЫЕ ПРУДЫ"

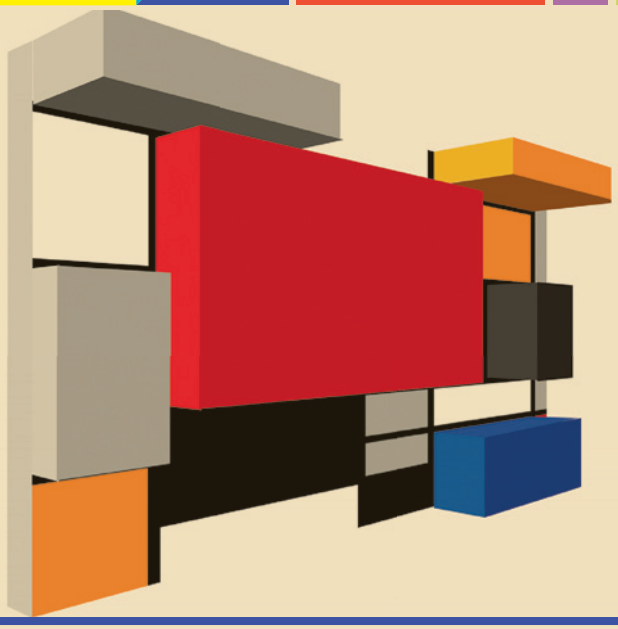
Зарегистрировано ПИ № ФС77-44341 от 22.03.2011 в Министерстве РФ по делам печати
Подписано в печать: по графику 10.09.2012, фактически 10.09.2012
Заказ №
Отпечатано в ОАО "Первая Образцовая типография" Филиал "Чеховский Печатный Двор" ул. Полиграфистов, д. 1, Московская область, г. Чехов, 142300
Сайт www.chpk.ru,
E-mail: salas@chpk.ru,
факс 8 (496) 726-54-10,
8 (496) 988-63-87

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

Отдел рекламы:
(499) 249-98-70
http://1september.ru

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

В этом номере не будет колонки главного редактора. У нас не поднялся курсор (и не опустилась клавиша DEL), чтобы сократить статью о языке Piet. Так что вместо колонки только три слова: "Вот это да!".



Живописное программирование вместе с Piet

И.А. Сукин,
г. Переславль-Залесский

Из сухости науки — в искусство!

Давно минули времена, когда программирование вычислительных машин выражало строгий язык математики и было уделом профессионалов, представлявших с помощью компьютерных программ серьезные технические задачи. Еще 20–30 лет назад мало кто мог подумать о том, чтобы написать программу, не несущую никакой, с виду, практической пользы. Однако с развитием как самой науки "computer science", так и вычислительной техники в целом множество энтузиастов начало ставить всевозможные эксперименты той или иной степени сумасбродности[1]. Целью подобных экспериментов было выяснение пределов сведения информатики как таковой ad absurdum: создание фантастических языков программирования, сложных для чтения, компиляции, написания программ или всего перечисленного сразу, выдумывание хитроумных логических и математических концепций, реализация невероятно запутанных программных систем и многое другое. Любители головоломок и задач на смекалку получили полную свободу в разминке своего ума с помощью ЭВМ, а специалистам-ученым предоставилась прекрасная возможность понаблюдать за "поведением" компьютеров, как искусственных артефактов, в непредсказуемых условиях[2]. Объектом нашего рассмотрения стал чрезвычайно удивительный даже среди своих не менее удивительных собратьев язык программирования — Piet, программы на котором выглядят как произведения современной абстрактной живописи.

Бесполезные или эзотерические?

Стремление человека к созданию странных, внешне нерациональных и даже бессмысленных объектов и предметов присутствовало, пожалуй, во все времена. XX век, привнесший невиданное до этого развитие абстрактной математики и средств моделирования, открыл простор для человеческой фантазии.

Языки программирования — искусственные языки, целью разработки которых изначально было составление "планов" или "программ" для вычислительных машин. На заре информатики единственными важными качествами любого языка программирования были удобство его компилирования в машиночитаемый

[1] Справедливости ради стоит сказать, что подобные эксперименты предпринимались и ранее, но в начале 90-х годов такая деятельность пережила свой золотой век.

[2] Отсылка к фундаментальному труду Г.Саймона "Науки об искусственном" [6].

вид и эффективность работы откомпилированной программы (в первую очередь), а также удобство языка для человека и соответствие его некоторым математическим или философским концепциям. Через некоторое время подобное положение дел наскучило любящим всяческие головоломки математикам, что дало толчок к созданию необычных средств программирования.

Языки, которые при первом рассмотрении не имеют никакой практической полезности, то есть отходят от всех общепризнанных парадигм программирования, принято называть *эзотерическими* [1]. Как я уже замечал ранее, такие языки могут применяться для изучения границ математических и лингвистических концепций, экспериментирования с необычными идеями, в качестве своеобразной гимнастики для ума или просто в роли шутки. В связи с этим стоит разделить все эзотерические языки программирования (ЭЯП) на два класса: тьюринг-полные и тьюринг-неполные. Несмотря на то, что такое разделение привычно для всех, в том числе и не эзотерических, языков программирования, в сфере ЭЯП оно носит более глобальный характер: как правило, тьюринг-неполные ЭЯП вообще непригодны для чего-либо и являются лишь пародиями или утрированными шутками.

ЭЯП полные по Тьюрингу[3] обычно представляют собой тьюринговские трясины: несмотря на то, что с их помощью можно вычислить все что угодно, сделать это необычайно трудно — здесь объект нашего рассмотрения, Piet, является ярким примером.

Простой ЭЯП, не полный по Тьюрингу

Одним из простейших эзотерических языков программирования, не обладающих тьюринговской полнотой, является HQ9+, который имеет всего четыре команды:

- **H** — вывести “hello, world”;
- **Q** — вывести исходный код программы (куайн);
- **9** — вывести текст стихотворения “99 бутылок”;
- **+** — увеличить значение регистра-аккумулятора (никаким другим способом не доступен для программиста).

Как легко заметить, на таком языке даже теоретически невозможно написать какую-либо полезную программу. Подавляющее большинство остальных неполных ЭЯП являются шутливыми вариациями на похожие темы.

Классификация полных ЭЯП

Большинство полных по Тьюрингу эзотерических языков программирования можно разделить на следующие классы (обычно родоначальником каждого класса является один язык):

- **Семейство INTERCAL**. Наиболее яркие представители: INTERCAL, FALSE, Babbage. Основной целью является создание языков, максимально отличных от уже существующих.
- **Семейство Brainf*ck**. Наиболее яркие представители: Brainf*ck, Ook!, Whitespace, BitBitJump. При создании языков этого семейства основное внимание уделяется максимальному сокращению количества управляющих конструкций (инструкций, операторов) при сохранении полноты языка. Существует также похожее направление в проектировании микропроцессоров. Семейство Brainf*ck играет важную роль в оценке полноты “серьезных” ЯП, поскольку сильно приближено к концепции машины Тьюринга. Сюда же можно отнести и минимальные функциональные языки вроде Unlambda, Thue и Lazy K, а также самомодифицирующиеся языки Smetana и Muriel.
- **Семейство Funge**. Наиболее яркие представители семейства: Befunge и его производные, *Piet*. Основной целью создания таких языков является сложность их компиляции. Как правило, такие ЯП многомерны и используют непривычное кодирование текста программы.
- **Семейство Malbolge**, в котором основным представителем является собственно Malbolge. Главной целью таких языков является максимальная сложность написания программ на них; как правило, это практически неосуществимая задача для человека. Так, к примеру, “Hello world” для Malbolge был сгенерирован программой на языке Lisp.
- **Недетерминистские языки**, вносящие в программирование элементы случайности. Они разнятся от откровенно абсурдных вроде Java2K до довольно полезных, таких как the Dada Engine, позволяющих проводить исследования в областях, требующих случайного поиска, и экспериментах с гипотетическими “супертьюринговскими” вычислениями.
- Существует много других эзотерических ЯП, использующих непривычные логики, синтаксис, близкий к литературным произведениям, и прочие подобные ухищрения, однако они несут скорее эстетическую и шутливую ценность.

Также иногда к эзотерическим относят реальные языки программирования, утратившие свою актуальность и использующие устаревшие приемы разработки, что, однако, не является правильным.

Многомерное программирование

Поскольку язык Piet относится к семейству так называемых “фунгеоидных” языков, будет нелишне рассмотреть их основные концепции. Язык Befunge был предложен в 1993 году в качестве языка, максимально сложного для компиляции. Основной отличительной особенностью Befunge является то, что код представлен в виде двухмерной таблицы символов, по которой интерпретатор может перемещаться в различных направлениях. Позднее появились вариации языка с представлением кода в виде трехмерных и четырехмерных таблиц. В качестве дальнейшего усложнения можно предложить увеличе-

[3] Полнотой по Тьюрингу называется теоретическая способность вычисления с помощью некоторого инструмента (в нашем случае — ЯП) любой вычислимой функции. Говоря простым языком, на тьюринг-полном языке можно запрограммировать все, что вообще можно запрограммировать на каком-либо ЯП.

ние числа измерений и возможность движения не в фиксированных дискретных направлениях, а по сложным траекториям. Однако в языке Piet был предложен другой интересный подход.

Программист Пит Мондриан

Уже неоднократно упомянутый мной ЭЯП Piet [2] был создан Дэвидом Морганом-Маром, одним из наиболее продуктивных и хитроумных авторов эзотерических языков. Свое название Piet получил в честь голландского художника Пита Мондриана, одного из основателей геометрической абстрактной живописи. Согласно оригинальной задумке, программы на Piet должны выглядеть как абстракционистские картины.

Программы на Piet — это компьютерные изображения, однако правила выполнения кода не столь очевидны, как может показаться сначала, поэтому расскажу об основных концепциях языка.

Основы языка

Цвета

Основные цвета, используемые в Piet, показаны в табл. 1 на с. 6. Согласно первоначальному описанию, используется 20 различных цветов. Восемнадцать цветов в первых трех строках таблицы наделены следующими циклическими отношениями:

- **Цикл оттенков (слева направо):** красный → желтый → зеленый → бирюзовый → синий → пурпурный → красный → …
- **Цикл уровней яркости (сверху вниз):** светлый → обычный → темный → светлый → …

При этом говорят о понятии “расстояния” в таких циклах. Например, “бирюзовый” на два оттенка отстоит от “желтого”, “темный” — на один от “обычного”, а “светлый” — на один от “темного” (грубо говоря, получается, что светлый “темнее”, чем темный). Белый и черный цвета носят вспомогательную управляющую функцию и не подчиняются описанным циклам отношений.

Важно отметить, что обработка других цветов не декларирована и сильно зависит от реализации. Такие цвета могут быть как дополнительными элементами циклов отношений, так и обрабатываться как “белый” или “черный”. Наиболее простым подходом является обработка таких цветов как “белого” (см. далее).

Коделы

Так как программы на Piet являются изображениями, в них важен каждый отдельный пиксель. Однако подобные изображения могут быть слишком маленькими и неудобными для просмотра, поэтому в Piet введено понятие “кодела”. Кодел является группой пикселей одного цвета (в самом простом варианте — одним пикселем) и представляет собой “пиксель” в понимании интерпретатора Piet. Это позволяет увеличивать программы на Piet без потери их смысла. Обычно кодел имеет квадратную форму, а его размер регулируется неким принятым соглашением (например, он может быть 8 × 8 пикселей). При увеличении программ на Piet важно не допустить интерполяции цветов и ненужного “замыливания” изображения. В дальнейшем при разговоре о пикселях будем иметь в виду коделы.

Цветовые блоки

Непрерывный блок смежных коделов одного цвета является основной составляющей частью программы на Piet. Если коделы смежны только по диагонали, то такой блок не считается непрерывным. В остальном форма блока может быть любой, в том числе иметь внутри себя “дырки”, заполненные коделами другого цвета, не входящими в данный блок, а образующими свой собственный. На *рис.* 1 изображены непрерывные блоки коделов в обычном понимании Piet (полость в сложном нижнем блоке не обведена для удобства).

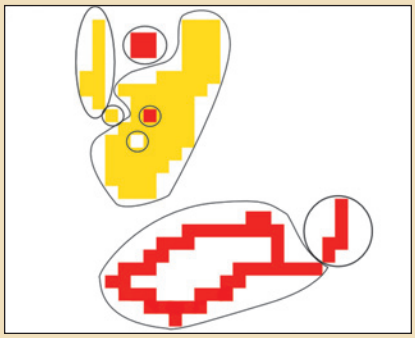


Рис. 1. Непрерывные блоки коделов

Стек

Основной структурой хранения данных в Piet является стек. Язык подобно Forth — бестиповый, данные существуют только в виде целых неотрицательных чисел, но могут быть интерпретированы разными способами — например, выведены на экран как символы Юникода.

Исполнение программ на Piet

В Piet существует два направляющих регистра. Первый из них — DP (Direction Pointer, Указатель направления) может принимать значения “Север”, “Юг”, “Запад”, “Восток” и по умолчанию указывает на “Восток”[4]. Кроме этого, существует второй регистр — CC (Color Chooser, Выборщик цвета), который может принимать значения “Слева” или “Справа” и по умолчанию установлен на “Слева”. Направления, на которые указывают эти регистры, могут часто меняться в ходе выполнения программы.

По умолчанию интерпретатор находится в цветовом блоке, содержащем верхний левый кодел программы.

Основной цикл обработки цветовых блоков выглядит следующим образом:

- Интерпретатор находит границу текущего блока в направлении DP.

[4] Во избежание путаницы, присущей оригинальному описанию, я заменил названия направлений на географические. Здесь “Север” соответствует верхнему краю изображения и т.д.

Таблица 1. Базовые цвета языка Piet

#FFC0C0 Светло-красный	#FFFFC0 Светло-желтый	#C0FFC0 Светло-зеленый	#C0FFFF Светло-бирюзовый	#C0C0FF Светло-синий	#FFC0FF Светло-пурпурный
#FF0000 Красный	#FFFF00 Желтый	#00FF00 Зеленый	#00FFFF Бирюзовый	#0000FF Синий	#FF00FF Пурпурный
#C00000 Темно-красный	#C0C000 Темно-желтый	#00C000 Темно-зеленый	#00C0C0 Темно-бирюзовый	#0000C0 Темно-синий	#C000C0 Темно-пурпурный
#FFFFFF Белый			#000000 Черный		

• Интерпретатор находит наиболее удаленный кодел в данном блоке в направлении DP, если смотреть в направлении CC.

• Интерпретатор пытается перейти с найденного на предыдущем шаге кодела в соседний цветовой блок в направлении DP.

Выглядит это довольно запутанно, поэтому необходимо некоторое наглядное пояснение. Представьте, что вы стоите на полу, покрытом разноцветными квадратными плитками (коделы), и смотрите в заданном направлении DP, а в уме держите направление CC. Для начала вам необходимо найти в данном цветовом блоке все такие плитки (коделы), которые наиболее удалены в направлении DP, например, все самые восточные или самые северные. Таких точек может быть несколько, поэтому, чтобы выбрать одну-единственную, надо из всего множества выбрать ту, которая наиболее удалена в направлении DP, скорректированном по CC. Например, если вы нашли все самые северные точки, а CC указывает влево, то надо выбрать среди самых северных самую западную (или самую восточную, если CC указывает вправо). Чуть позже я разберу выполнение реальной программы на Piet, чтобы показать эту концепцию наглядно.

Другие элементы синтаксиса

Числа. Каждый не черный и не белый цветовой блок в Piet представляет целое положительное число, равное количеству коделов в этом блоке. Не положительные числа необходимо специально конструировать с помощью команд, о которых я расскажу позднее. Кроме всего прочего, числа в Piet необходимо явно класть на стек.

Черные блоки и границы. Если на третьем шаге описанного выше цикла интерпретации интерпретатор Piet пытается перейти в черный блок или за границу изображения, он останавливается и изменяет направление CC на противоположное, после чего пытается повторить цикл. Если повторить цикл не удается, то направление DP меняется на 90 градусов по часовой стрелке и производится еще одна попытка. В ходе попыток CC и DP меняются попеременно. Если после восьми попыток интерпретатор не нашел выхода из блока (блок ограничен со всех сторон), программа завершается.

Белые блоки. Блоки коделов белого цвета совершенно прозрачны для интерпретатора: он проходит сквозь них по прямой линии в направлении DP до тех пор, пока не встретится не белый блок. Если этим блоком будет блок черного цвета или граница изображения, то регистр направления CC меняется, а DP при этом также меняет направление на 90 градусов по часовой стрелке (поскольку внутри белого блока выборки крайних коделов не производится). Такой цикл выполняется до тех пор, пока интерпретатор не повторит свой путь внутри белого блока. Если это случится, программа завершится.

Еще одним важным дополнением касательно белых блоков является то, что проход через белый блок между двумя цветными не вызывает никакой команды, что может использоваться, например, при программировании циклов для “безболезненной” смены цвета.

Команды. Система команд языка Piet очень похожа на упрощенный набор слов Forth или систему команд стекового калькулятора. Код команды всецело определяется расстоянием между цветами в двух циклах отношений, приведенных выше, при переходе интерпретатора из одного цветового блока в другой. Если переход был совершен через белый блок, то не выполняется никакая команда. Схема кодирования команд дана в табл. 2.

Приведу список команд Piet с пояснениями (предположу, что читатели знакомы с принципами функционирования стека):

• **push:** кладет на стек число, соответствующее числу коделов в цветовом блоке, из которого интерпретатор только что вышел;

• **pop:** выкидывает одно число с вершины стека;

• **add, sub, mul, div, mod:** арифметические команды. Снимают два числа с вершины стека, складывают/вычитают/умножают/нацело делят/вычисляют остаток деления их и кладут результат на стек;

Таблица 2. Кодирование команд в языке Piet

	Изменение яркости		
Изменение оттенка	**Нет**	**На один шаг**	**На два шага**
Нет		push	pop
На 1 шаг	add	sub	mul
На 2 шага	div	mod	not
На 3 шага	greater	pointer	switch
На 4 шага	dup	roll	in (число)
На 5 шагов	in (символ)	out (число)	out (символ)

• **not:** заменяет значение на вершине стека на 0, если оно было ненулевым, и на 1 в противоположном случае;

• **greater:** снимает со стека два числа и кладет туда единицу, если второе число от вершины стека больше, чем число, которое было на вершине, и ноль в противном случае;

• **pointer:** снимает число с вершины стека и меняет направление DP по часовой стрелке данное число раз. Если снятое число отрицательно, то вращение происходит против часовой стрелки;

• **switch:** снимает число с вершины стека и меняет направление CC данное число раз;

• **dup:** копирует число на вершине стека и кладет его на стек (создает копию вершины);

• **roll:** снимает два числа — *n* (вершина) и *m* (второе число после вершины) со стека и производит с оставшимися на стеке числами следующую операцию: вращает *m* чисел от вершины стека *n* раз. Вращение *m* чисел один раз производится так: вершина стека кладется на глубину *m*, а глубина всех элементов, располагавшихся на глубине меньшей или равной *m*, уменьшается на единицу. Если *n* отрицательно, то вращение производится в противоположном направлении. Если *m* отрицательно, то команда просто игнорируется;

• **in, out:** команды ввода-вывода; *in* читает из входного потока число (или символ) и кладет его на стек, *out* снимает число со стека и выводит его в выходной поток в виде числа или символа Юникода.

Как можно видеть, программирование на Piet является довольно нетривиальным занятием и требует в первую очередь некоторого навыка общения со стеком.

Программирование на Piet

Каким бы удивительным это ни казалось, но для Piet написаны полноценные интерпретаторы и даже среды удобного интерактивного редактирования программ. Мы рассмотрим одну такую среду, предоставляемую проектом npiet. К сожалению, полноценная работа npiet гарантируется только под ОС семейства GNU/Linux, а пользователям Windows придется несколько помучиться.

Интерпретатор npiet

Интерпретатор npiet является полноценной реализацией языка Piet и в качестве программ на нем принимает изображения в форматах PNG, PNM или GIF. Изображения JPEG не поддерживаются в силу того, что могут быть искажены из-за сжатия. Важной особенностью npiet является возможность как текстовой, так и графической трассировки Piet-программ, что существенно облегчает жизнь экспериментаторам. Мы рассмотрим эти возможности, когда будем писать (рисовать) первую программу.

Среда npietedit

В комплект npiet входит также простейшая среда разработки на Piet, позволяющая с удобством редактировать изображения попиксельно и подсказывающая нужный цвет относительно текущего для создания необходимой команды.

Первая программа

В качестве первой программы на Piet рассмотрим программу, читающую из входного потока число и возвращающую результат умножения этого числа на два (без символа перевода строки), оставив само число на стеке. Один из вариантов такой программы приведен на *рис.* 2.

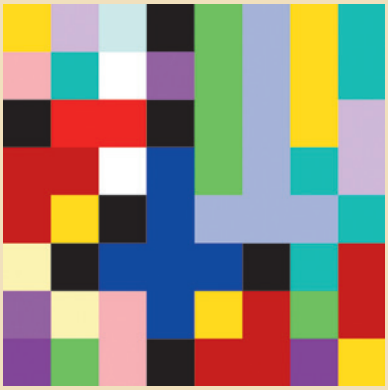


Рис. 2. Программа на Piet, умножающая заданное число на 2

Для начала такую программу надо записать в виде текста с использованием вышеописанных команд Piet:

```
in (число)
dup
push 2
mulout (число)
```

Сначала мы вводим число командой **in** и копируем его с помощью **dup**, после чего помещаем на стек двойку и вызываем **mul**. Полученный результат должен быть выведен командой **out**. Вы могли заметить, что приведенная на *рис.* 2 программа явно избыточна. Действительно, для записи вышеприведенного алгоритма достаточно использовать всего 9 коделов. Однако нам необходим еще способ выхода из программы. Выход — не самое простое действие в Piet, но для его организации существуют некоторые устоявшиеся шаблоны, и я буду использовать один из них, который условно можно назвать “Крест”. Упрощенная версия программы приведена на *рис.* 3, а ее оттрассированная с помощью npiet версия — на *рис.* 4.

Рис. 3. “Скелет” вышеприведенной программы

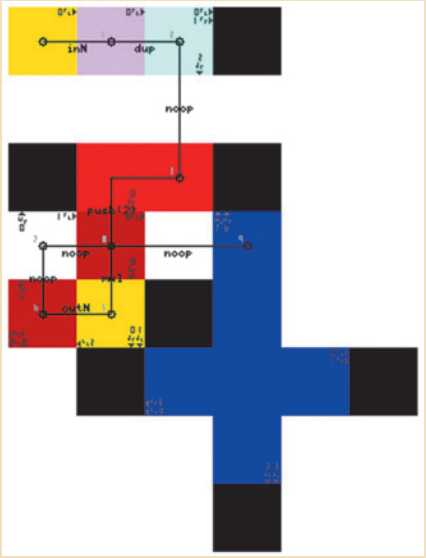


Рис. 4. Результат трассировки программы

Согласно некоторым “правилам хорошего тона”, программа на Piet должна быть как можно более квадратной и как можно более художественной. Ориентировочно, возьмем для нашей программы поле 8 на 8 коделов, затем выберем какой-нибудь начальный цвет, например, желтый. Изначально интерпретатор будет находиться в точке (0, 0), в блоке желтого цвета, его DP будет указывать на восток, а CC — влево.

Последовательность из первых двух команд проста — это первые три кодела в верхней строке *рис.* 3 (можете проверить это сами по таблице). Чтобы не удлинять программу, поставим после них черный кодел, заставив интерпретатор повернуться на юг.

Почему он поворачивается именно на юг? Первый раз интерпретатор пытается войти в черный блок и меняет направление CC, однако после этого он все равно будет пытаться войти в этот же черный блок, следовательно, ему придется повернуть еще и DP.

Теперь нам нужно положить на стек число 2. Чтобы закодировать двойку, нужно перейти в какой-нибудь блок из двух коделов, при этом нам не нужно лишнее выполнение команды, поэтому сделать это надо через белый блок. Выберем красный в качестве цвета для блока, кодирующего двойку. Поскольку интерпретатор пока находится в точке (2, 0) и смотрит на юг, оставим кодел (2, 1) белым. Затем создадим какой-нибудь блок из двух коделов одного (красного, как мы выбрали) цвета, содержащий точку (2, 2). Пусть это будет блок [(1, 2); (2, 2)]. Поскольку DP указывает на юг, а CC — вправо (он изменился, когда интерпретатор пытался войти в кодел (3, 0)), интерпретатор выберет кодел (1, 2) и попытается перейти в блок, расположенный южнее. Поместим в точку (1, 3) кодел темно-красного цвета для кодирования команды push, а в точку (1, 4) — кодел желтого цвета для кодирования команды mul.

Опять же, чтобы чересчур не удлинять код программы, поместим в точку (1, 5) черный кодел, заставив интерпретатор изменить CC (теперь он будет указывать влево) и повернуть DP, чтобы он указывал на запад. После этого поместим в точку (0, 4) темно-красный кодел для кодирования команды out.

Теперь приступим к самой сложной части — выходу из программы. Как я уже говорил, будем использовать для этого технику “креста”. Суть этого метода заключается в том, чтобы “завести” интерпретатор в крестообразный цветовой блок, лучи которого ограничены черными коделами. На *рис.* 3 такой блок синего цвета можно увидеть в нижнем правом углу. Он может иметь и более простой вид, но для целей наглядности оставим его как есть. Вспомним, что наш интерпретатор находится в точке (0, 4), DP указывает на запад, а CC — влево. Два раза “стукнувшись” о западную границу изображения, интерпретатор изменит значения как CC, так и DP (на “Север”) и пойдет в кодел (0, 3). Черный кодел в точке (0, 2) снова повернет интерпретатор на 90 градусов — на “Восток”, значение CC при этом снова будет “Слева”. Безболезненно пройдя через темно-красный кодел в (1, 3) и белый в (2, 3), интерпретатор попадется в нашу ловушку — синий крестообразный блок, из которого и не сможет выйти, заставив программу завершиться.

Как я уже говорил, графическая трассировка нашей программы приведена на *рис.* 4, здесь тонкая черная линия показывает путь интерпретатора, в точности соответствующий тому, который я описал выше.

После того как скелет программы готов, мы можем украсить его по собственному желанию дополнительными цветными коделами, которые не будут нести никакой смысловой нагрузки. Приведу несколько простых эмпирических правил такого украшения, не нарушающих смысл исходной программы:

- блоки коделов, представляющие числа, должны быть окружены коделами другого цвета;
- блоки коделов, служащие для выхода из программы, подобные нашему “кресту”, также должны быть окружены коделами другого цвета;
- важно найти все значащие белые коделы и оставить их без изменения;
- если вы заменяете не значащие белые коделы на пути интерпретатора, заменяйте их коделами текущего цветового блока, чтобы не вызывать лишних команд, но только в том случае, если этот блок не представляет число.

В нашей программе значащих белых кодела два — в точке (1, 2) и в точке (2, 3). Белый кодел в точке (0, 3) не является значащим. Если посмотреть на трассировку программы, видно, что его можно заменить коделом темно-красного цвета (а можно и не менять), поскольку получающийся при этом блок в виде буквы “Г” не представляет число и не меняет семантику программы. Кодел (2, 3) важно оставить белым, чтобы не совершать вызова лишних команд перед выходом из программы.

Как читать трассировку

Приведенная на *рис.* 4 трассировка показывает полный цикл выполнения программы. Кружки внутри коделов отмечают точки, в которых останавливается интерпретатор перед очередной выборкой кодела, а прямые линии, соединяющие их, — собственно путь интерпретатора. Над кружками записываются числа, обозначающие номер шага интерпретатора. Между цветовыми блоками написаны мнемонические обозначения Piet-команд.

В каждом блоке с помощью небольших стрелочек отмечаются попытки перехода интерпретатора через границу и соответствующие значения DP и CC. Цифры рядом с этими стрелочками отмечают номер попытки.

Более сложные программы

В качестве примера более сложной программы, являющейся при этом действительно образцом некоторого визуального искусства, можно привести один из вариантов “Hello world” на Piet (*рис.* 5). Не буду разбирать ее подробно, поскольку это займет слишком много места, очерчу лишь общий принцип ее работы.

Программа на *рис.* 5 развивается по спирали, при этом для поворота на очередной виток применяются команды pointer. В ходе закручивания спирали некоторые символы выводятся сразу, а некоторые формируются заранее и кладутся на стек. В центре изображения можно увидеть еще один шаблон выхода из Piet-программ, очень похожий на “Крест”, — “Клетку” (по сути, она является частным случаем “креста”). Будучи заключенным в блок с черной границей, интерпретатор не может найти выход и завершает программу.

Приведу также трассировку этой программы. Подробное объяснение принципов работы “Hello world” на Piet приведено в [5].

Другие примеры программ на Piet вы можете увидеть в приложении и на веб-страницах по приведенным в конце статьи ссылкам.

Рис. 5. Художественный “Hello world” на Piet

Что же дальше?

Итак, как мы увидели, язык Piet сложен и интересен, а программирование на нем является поистине искусством для избранных. Но есть ли способы еще сильнее усложнить себе жизнь и разработать какой-нибудь еще более невероятный диалект этого языка? Разумеется! Можно, например, добавить еще одно измерение и ввести концепцию “вокоделов” (объемных коделов), дать возможность регистру DP принимать шесть различных значений, а регистру CC — четыре. С другой стороны, можно добавить более сложные варианты уже привычных направлений и траекторий: все возможно. Думаю, вы уже поняли, что ваша фантазия в области создания эзотерических языков программирования ничем не ограничена.

Приложения

На диске к номеру имеется несколько примеров программ на Piet с трассировками и краткими пояснениями[5].

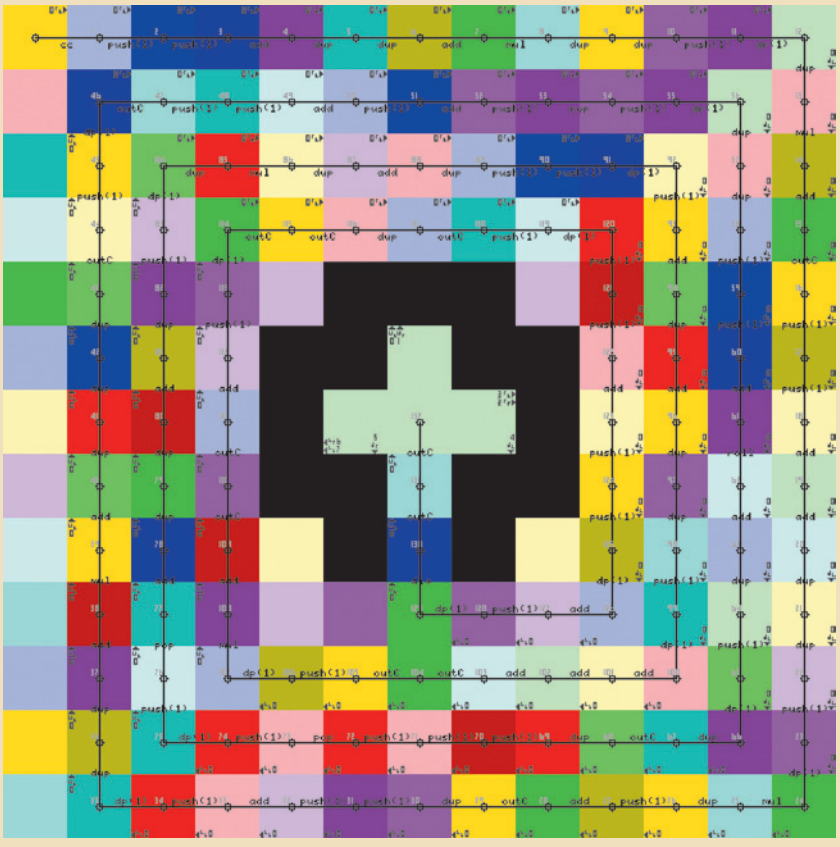


Рис. 6. Трассировка программы “Hello world”

Использованные источники информации

1. Портал, посвященный эзотерическим языкам программирования, *http://esolangs.org*.

2. Официальная страница языка Piet, *http://www.dangermouse.net/esoteric/piet.html*.

3. Примеры программ на Piet, *http://www.dangermouse.net/esoteric/piet/samples.html*.

4. Интерпретатор npiet, *http://www.bertnase.de/npiet/*.

5. Как работает “Hello world” на Piet, *http://www.retas.de/thomas/computer/programs/useless/piet/explain.html*.

6. *Саймон Г.* Науки об искусственном. М.: УРСС, 2004, 144 с., ISBN 5-354-00759-3.

[5] В сети можно найти программу на Piet, якобы печатающую “Piet” и выглядящую при этом удивительно похожей на одну из картин Пита Мондриана. К сожалению, мне не удалось запустить ее с помощью npiet.

“Портабельные” программы: прошлое, настоящее и будущее

Д.Ю. Усенков,
Институт информатизации образования Российской академии образования, Москва

В последние несколько лет все более популярными среди пользователей становятся так называемые *портабельные* программы (или, в англоязычном варианте, — *portable*). Однако для многих это название все еще остается непонятным. Помочь разобраться в этом вопросе им сможет эта статья.

Что такое портабельная программа

Портабельная программа — это транскрипция англоязычного понятия “portable software”, определяющего программы, не требующие для своей работы операции инсталляции. Такая программа всегда готова к запуску (в самом крайнем случае ее достаточно лишь извлечь из Zip- или RAR-архива, в котором она была сохранена).

Портабельные программы можно разделить на две разновидности:

• полностью портабельные — все их файлы (в том числе файлы настроек и временные файлы, создаваемые на время текущего сеанса работы) хранятся только в рабочем каталоге. Такие программы очень удобно хранить на флешке или на внешнем жестком диске и запускать непосредственно с них: единожды настроенные, они на любом другом компьютере будут использовать прежние настройки и “не оставлять следов” на самом этом компьютере;

• частично портабельные — работают без инсталляции, но часть своих файлов (в частности, с настройками) хранят в системной папке Windows или где-то в общей папке “Мои документы”. Поэтому при использовании такой программы на другом компьютере ее настройки придется выполнять заново, и к тому же на каждом компьютере при работе с такой программой образуются “мусорные” файлы.

В некоторых случаях программы причисляют к портабельным ошибочно. Такие программы тоже не требуют для начала работы отдельной операции инсталляции, однако после первого запуска все же выполняют скрытую (неявную) установку, копируя часть файлов в системные каталоги и создавая записи в системном реестре.

Немного истории…

Хотя портабельные программы появились сравнительно недавно и, очевидно, уже после появления ОС Windows (или аналогичных ей) как альтернатива "обычным" программным пакетам, требующим инсталляции с дистрибутива, сами такие программы очень хорошо знакомы всем, кто начинал свое знакомство с персональными компьютерами еще в эпоху ОС MS-DOS. Более того, если вспомнить то славное прошлое (кстати, совсем недавнее!), когда на ПЭВМ еще не было жестких дисков, а все ПО, включая операционную систему и файловый менеджер типа Norton Commander, запускалось исключительно с дискет (тогда еще пятидюймовых, объемом всего в 360 килобайт ☺), то нетрудно догадаться, что ***все*** тогдашние программы, по сути, были "портабельными", потому как не требовали никакой инсталляции (за редкими исключениями, да и то сводящимися только к распаковке файлов из дистрибутивного архива и первичной настройке под аппаратные особенности компьютера) и запускались в работу сразу. Тогда это никого не удивляло и было само собой разумеющимся (хотя сам термин "портабельные" тогда еще не родился). Более того, все программы, которые кто-либо писал сам на каком-нибудь языке программирования, тоже были исключительно "портабельными", — пока не стали появляться различные "продвинутые" турбо- и визуал-оболочки программирования, позволяющие создавать дистрибутивы. В первых компиляторах и простейших оболочках типа Turbo Pascal, как многие помнят, результатом работы программиста являлся всего-навсего исполняемый (exe-) файл, в крайнем случае дополненный библиотеками или файлами данных (например, с закодированными спрайтами — изображениями персонажей компьютерной игры или с текстами диалогов с пользователем). Для работы с такой программой достаточно было, независимо от того, на каком носителе (на дискете или на жестком диске) она в данный момент находится, запустить на исполнение соответствующий exe-файл и — *пользоваться*…

Портабельные программы сегодня

С усложнением операционных систем (и в основном после появления Windows 95) операция инсталляции программ, ранее бывшая скорее "экзотикой", стала общепринятой. Хотя благодаря типовому алгоритму работы с инсталляторами и их во многом стандартизированному интерфейсу инсталлировать программу может практически любой пользователь, даже начинающий, а сама по себе "прописка" программы в операционной системе и возможность за счет этого использовать ресурсы ОС повысила удобство работы и пользователя, и программиста, операция инсталляции все же сопряжена с рядом мелких неудобств. Главные из них — жесткая (как правило) "привязка" установленной программы к конкретному компьютеру и изрядное "засорение" системных папок и реестра Windows "мусорными" файлами и записями после удаления программ (особенно у тех, кто любит пробовать новые freeware-программы, скачанные из Интернета).

Несколько лет назад это особенно никого не смущало (кроме разве что "компьютерных пиратов"): такая "привязка" считалась само собой разумеющейся, а при необходимости можно было повторить операцию на другом компьютере. Однако с все более широким распространением "упрощенных" компьютеров — нетбуков проблема инсталляции вновь становится актуальной.

Нетбук — это, по сути, все тот же ноутбук, но сверхпортативный (по весу и размерам) и максимально дешевый. Например, модель Asus EEE при габаритных размерах примерно с 70-страничную книжку стандартного формата 60 × 90 и весе около одного килограмма стоит около 10 000 рублей. Очевидным следствием из такого соотношения "портативность – цена" является значительная упрощенность нетбука и минимализм его аппаратных ресурсов по сравнению с "настоящими" ноутбуками, а тем более настольными ПК. В самых простых нетбуках вообще вместо винчестера для хранения программ и системных данных предусматривается некоторый небольшой объем флеш-памяти, а в качестве внешних носителей (для пользовательских данных) предполагаются обычные "флешки", внешние жесткие диски, подключаемые через порты USB, либо карточки флеш-памяти, вставляемые в гнездо кардридера.

И вот тут начинаются сложности. Инсталляция требуемого ПО на таком нетбуке может оказаться невозможной, потому что на встроенном диске не хватит места, а установка и запуск программы непосредственно с внешнего носителя чаще всего оказываются невозможными, поскольку при инсталляции обычная программа обязательно должна что-то записать в системной папке. Кроме того, если дистрибутив распространяется на компакт-диске (CD или DVD), то возникает дополнительная сложность: нетбуки, как правило, не оснащены встроенным накопителем CD/DVD,

и требуется приобретать и подключать к ним внешний накопитель.

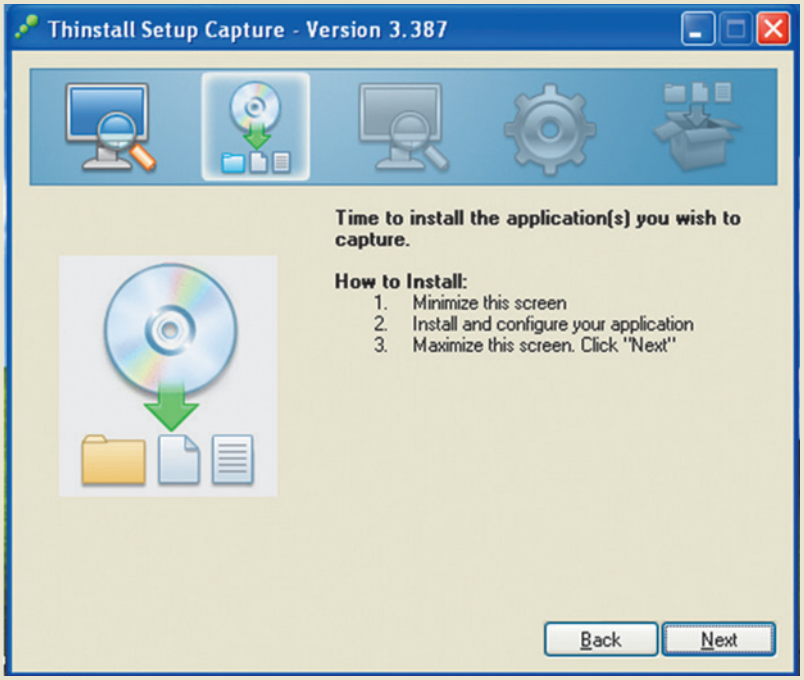
Добавляет сложностей и активизировавшаяся в последнее время борьба с “пиратством”. Если раньше многие программы можно было найти практически на любом компьютере (и дома, и на работе, и в компьютерном клубе), то теперь их владельцы не только сто раз подумают — а нужно ли ставить ту или иную программу, за которую легально придется заплатить достаточно большие деньги, но и (если владелец — юридическое лицо) часто запрещают рядовым сотрудникам вообще что-либо устанавливать на рабочих компьютерах, попросту не давая им прав “администратора”. В результате пользователь, придя, например, в вуз, школу или какую-либо стороннюю фирму, далеко не всегда уверен, что там имеется нужная ему программа для обработки или даже для просмотра требуемого файла, а работники этих учреждений ничем не могут помочь своим клиентам, потому как не имеют прав для установки на своем ПК даже открытого (свободного, некоммерческого) ПО.

Хорошим решением такой проблемы могло бы быть использование программ, “привязываемых” при инсталляции не к конкретному компьютеру, а к *конкретному пользователю*[1]. Например, если бы флешка имела уникальный номер либо обеспечивалось биометрическое распознавание пользователя, то можно было бы учитывать эти данные при инсталляции программы на флешку или на внешний жесткий диск, а пользователь — владелец лицензионной версии мог бы работать с ней на *любом* компьютере — дома, в офисе или в классе либо даже в компьютерном клубе. Но пока разработчики ПО не спешат следовать по этому пути. И поэтому единственным удобным выходом для многих пользователей, чья работа подразумевает повышенную “мобильность”, стали как раз портабельные программы, которые можно запускать непосредственно с той же флешки на любом компьютере, куда эта флешка подключена.

Единственное неудобство портабельных программ (и то — не всех) заключается в частичном усечении их функций. Дело в том, что некоторые возможности, например, связанные с работой с периферийными устройствами, могут быть реализованы в среде Windows только через эту ОС и путем взаимодействия с драйверами устройств и реестром системы. Поэтому, скажем, нетрудно может оказаться найти портабельную программу для чтения PDF-файлов, но может не существовать портабельной программы для создания PDF (хотя обычную, инсталлируемую свободно распространяемую программу для этой цели найти нетрудно), поскольку для выполнения такой операции необходимо наличие установленного в ходе инсталляции программы PDF-конвертора “виртуального принтера”. Но в целом этот недостаток не столь значителен, как кажется на первый взгляд, и проявляется он весьма нечасто.

Где и как?

У большинства читателей наверняка уже готов вопрос: где взять эти портабельные программы? Ответ весьма очевиден: достаточно в любом поисковом сервисе (будь то Google, Яндекс или какой-либо другой) ввести в составе ключевой фразы с названием искомой программы или описанием ее требуемых функций “волшебное” слово **portable**, чтобы получить целую подборку веб-страниц с информацией о них. А если ввести ключевую фразу “portable программы”, то можно получить и большую подборку сайтов-коллекций, предоставляющих возможность скачать портабельные программы различного назначения (в основном — свободно распространяемые). Кроме того, многие разработчики обычных программ предусматривают для них портабельные



[1] Другой вариант — использование сетевых версий ПО, свободного либо “арендуемого” через Интернет, например, за небольшую доплату к стоимости доступа в сеть, — мы здесь не рассматриваем, так как эта сфера тоже еще недостаточно развита, а обязательное условие для обеспечения возможности ею пользоваться — наличие высокоскоростного доступа в Интернет в любом месте и в любое время, в том числе беспроводного, — у нас в России, похоже, еще долго останется только в мечтах и планах.

версии, которые можно переписать на соответствующих сайтах. И, наконец, там же в Интернете можно найти инструментальные средства для "изготовления" из обычного инсталлируемого программного пакета желаемой портабельной программы (такой процесс иногда называют ее "виртуализацией"), — например, таким средством является программа Thinstall Virtualization Suite. Сама она, правда, требует инсталляции, но зато потом позволяет при инсталляции "исходной" программы (подлежащей преобразованию в портабельную) отследить весь ход процесса инсталляции, включая записываемые где-либо файлы и изменения в реестре, а затем с учетом этой информации создает (компилирует) портабельную версию. (Производить такую работу, конечно, лучше в "виртуальной машине" с установленной ОС Windows, чтобы не "замусоривать" свой компьютер и его "основную" ОС при этой "рабочей" инсталляции исходной программы, да и вообще избежать нежелательных сбоев в работе ОС из-за случайных ошибок при преобразовании программы в портабельную.) Инструкцию по работе с программой Thinstall Virtualization Suite и примеры выполнения этой операции тоже можно отыскать в Интернете. При этом интерфейс программы Thinstall построен по принципу пошагового "мастера", и потому процесс создания портабельной программы получается несложным, — хотя, возможно, и потребуется немного поэкспериментировать…

Возможное будущее

Небезынтересно попытаться "заглянуть в будущее" и представить себе, что ждет портабельные программы в ближайшие годы.

Очевидно, с все более широким распространением дешевых нетбуков и планшетов (в том числе в школах в рамках таких мероприятий, как инициатива Intel "Classmate PC" и аналогичная ей инициатива "One Laptop Per Child" (OLPC)) и с ужесточением лицензионной дисциплины спрос пользователей на портабельные программы будет заметно возрастать.

Сторонники борьбы за лицензионный софт уже, наверное, готовы возразить, что такой рост популярности портабельных программ ухудшает ситуацию с "компьютерным пиратством". Отчасти они правы: многие из предлагаемых в Интернете портабельных программ представляют собой "виртуализованные" версии коммерческого ПО, а поскольку портабельные программы уже никак не привязаны к компьютеру, на котором была инсталлирована исходная коммерческая версия, и их можно (по крайней мере с технологической точки зрения) свободно копировать и запускать, то распространение таких портабельных программ — это одно из направлений "компьютерного пиратства". Однако можно ожидать, что разработчики ПО все же начнут "подтягиваться" вслед за ростом интересов со стороны пользователей и начнут предлагать им легально выпущенные портабельные версии своей продукции (например, несколько усеченные по сравнению с обычными или же, скажем, продающиеся через интернет-магазины по сравнительно небольшим ценам, как сегодня это делается с электронными книгами в формате PDF, — тогда рядовому пользователю будет проще заплатить небольшую сумму, чем облазить весь Интернет в поисках требуемого). Кроме того, вполне возможно, что создатели портабельных программ научатся делать их "привязку" к конкретной флешке, сохраняя, однако, главное свойство портабельных программ — возможность работы с этой флешкой на любом компьютере без инсталляции и без "привязки" к самому этому компьютеру.

Следует также заметить, что далеко не все существующие портабельные программы имеют отношение к "пиратству". Многие из них либо построены на базе свободно распространяемых программ ("обычных"), либо сами изначально разработаны как свободно распространяемые. Поэтому целый ряд фирм может попытаться занять нишу "комплектовщиков" и распространителей комплектов портабельных программ, — точно так же, как сегодня многие фирмы занимаются сборкой и продажей дистрибутивов на базе ОС Linux. И подобные примеры уже были! Скажем, хорошо известная пользователям Linux-систем санкт-петербургская фирма "Линуксцентр" (*http://www.linuxcenter.ru*) несколько лет назад предлагала своим клиентам комплект офисного ПО на базе OpenOffice.org, размещенного на флешке и запускаемого с нее (т.е., по сути, предлагался именно комплект портабельных программ).

Подводя итоги, отметим: вполне можно ожидать, что рынок портабельных программ в последующие годы разовьется и станет более цивилизованным. Что, безусловно, пойдет на пользу всем пользователям, повысив удобство их работы.

Методика решения задачи С4 из демонстрационного варианта ЕГЭ по информатике 2013 года

Д.М. Златопольский,
Москва

Условие [1]

На вход программе подаются сведения о пассажирах, желающих сдать свой багаж в камеру хранения на заранее известное время до полуночи. В первой строке сообщается количество пассажиров *N*, которое не меньше 3, но не превосходит 1000; во второй строке — количество ячеек в камере хранения *M*, которое не меньше 10, но не превосходит 1000. Каждая из следующих *N* строк имеет следующий формат:
<Фамилия> <время сдачи багажа> <время освобождения ячейки>,
— где <Фамилия> — строка, состоящая не более чем из 20 непробельных символов; <время сдачи багажа> — через двоеточие два целых числа, соответствующие часам (от 00 до 23 — ровно 2 символа) и минутам (от 00 до 59 — ровно 2 символа); <время освобождения ячейки> имеет тот же формат. <Фамилия> и <время сдачи багажа>, а также <время сдачи багажа> и <время освобождения ячейки> разделены одним пробелом. Время освобождения больше времени сдачи.

Сведения отсортированы в порядке времени сдачи багажа. Каждому из пассажиров в камере хранения выделяется свободная ячейка с минимальным номером. Если в момент сдачи багажа свободных ячеек нет, то пассажир уходит, не дожидаясь освобождения одной из них.

Требуется написать программу (укажите используемую версию языка программирования, например, Borland Pascal 7.0), которая будет выводить на экран для каждого пассажира номер ячейки, которая ему будет предоставлена (можно сразу после ввода данных очередного пассажира). Если ячейка пассажиру не предоставлена, то его фамилия не печатается.

Пример входных данных:

```
3
10
Иванов 09:45 12:00
Петров 10:00 11:00
Сидоров 12:00 13:12
```

Результат работы программы на этих входных данных:

```
Иванов 1
Петров 2
Сидоров 1
```

Решение

Прежде всего обратим внимание на два важных обстоятельства.

1. В условии отсутствует требование, связанное с эффективностью программы (“Вам предлагается написать эффективную, в том числе и по используемой памяти, программу” или т.п.), имевшееся в заданиях демонстрационных вариантов нескольких последних лет, а также в книгах от разработчиков ЕГЭ по информатике [2–3].

2. В задании отсутствует требование кратко описать используемый алгоритм решения, впервые приведенное в задаче С4 демоварианта экзамена 2012 года.

Обсудим методику решения данной задачи.

Чтобы найти номер ячейки, которую надо выделить некоторому пассажиру, нужно знать желаемое им время сдачи багажа (естественно) и информацию о времени освобождения всех ячеек. Последнюю информацию в программе на школьном алгоритмическом языке будем хранить в массиве *время_осв*. Размер этого массива примем равным 1000, а тип элементов может быть разным. Самым простым вариантом, по нашему мнению, является использование данных строкового (в терминах языка Паскаль) типа. Заполнение массива *время_осв* можно проводить в ходе выполнения программы по мере обработки входных строк и “выделения” пассажирам ячеек, записывая в соответствующие элементы массива время ее (ячейки) освобождения.

Итак, для каждого пассажира нужно выделить из входной строки значения времени сдачи багажа в камеру хранения (имя этой величины в программе — *время_сдачи*) и время освобождения ячейки (*время_освоб*). Самый простой способ сделать это — использовать так называемые “срезы”, зная “местонахождение” необходимых значений в строке.

В школьном алгоритмическом языке “срез” из строки можно получить, указав имя переменной, из которой формируется вырезка, и в квадратных скобках — начальный и конечный номера символов среза:

```
время_освоб := свед[длин(свед) - 4 :
               длин(свед)]
время_сдачи := свед[длин(свед) - 10 :
               длин(свед) - 6]
```

— где *свед* — сведения об очередном пассажире (очередная входная строка);

— *длин* — функция, возвращающая длину строки — ее аргумента.

В языке Бейсик для этого используется функция MID$, в языке Паскаль — функция Copy. В этих функциях — три параметра: величина строкового типа, начальный номер символа для вырезки и длина вырезки. Обратим внимание на то, что, хотя мы используем термин “срез”, из самой строки ничего не вырезается — ее значение не изменяется.

Конечно, нужно выделить и фамилию пассажира. Это также можно сделать с помощью “среза”:

```
фам := свед[1 : длин(свед) - 12]
```

В решениях, приведенных в [1], тип переменных *время_сдачи* и *время_освоб* (и элементов используемого массива) — числовой, а значения определяются как количество минут, прошедших с начала суток к соответствующим моментам времени. Для этого выделяются отдельно количество часов и количество минут, прошедших с начала полного часа (в программе на языке Бейсик путем “срезов” и последующего преобразования строковой величины в число, в программе на Паскале — путем посимвольного считывания нужных значений). Фамилия пассажира определяется аналогично. Для сравнения приведем соответствующие фрагменты программ из [1]:

Язык Паскаль

```
name := '';
repeat
  read(c);
  name := name + c
until c = ' '; {считана фамилия}
read(c, c1);
{считаны часы первого времени}
time1 := 60 * ((ord(c) - ord('0')) * 10 +
         ord(c1) - ord('0'));
read(c, c, c1);
{пропущено двоеточие и считаны минуты}
time1 := time1 + (ord(c) - ord('0')) * 10
         + ord(c1) - ord('0');
read(c, c, c1);
{считаны часы второго времени}
time2 := 60 * ((ord(c) - ord('0')) * 10
         + ord(c1) - ord('0'));
readln(c, c, c1);
{пропущено двоеточие и считаны минуты}
time2 := time2 + (ord(c) - ord('0')) * 10
         + ord(c1) - ord('0');
```

Язык Бейсик

```
c$ = MID$(s, 1, 1) 's – входная строка
i = 1
WHILE NOT (c$ = " ")
  i = i + 1
  c$ = MID$(s, i, 1)
WEND
'nm – фамилия пассажира
nm = MID$(s, 1, i)
time1 = (ASC(MID$(s, i + 1, 1)) -
         ASC("0")) * 60 * 10
time1 = time1 + (ASC(MID$(s, i + 2, 1)) -
        ASC("0")) * 60
time1 = time1 + (ASC(MID$(s, i + 4, 1)) -
        ASC("0")) * 10
time1 = time1 + (ASC(MID$(s, i + 5, 1)) -
        ASC("0"))
time2 = (ASC(MID$(s, i + 7, 1)) -
         ASC("0")) * 60 * 10
time2 = time2 + (ASC(MID$(s, i + 8, 1)) -
        ASC("0")) * 60
time2 = time2 + (ASC(MID$(s, i + 10, 1)) -
        ASC("0")) * 10
time2 = time2 + (ASC(MID$(s, i + 11, 1)) -
        ASC("0"))
```

Предлагаем читателям оценить преимущества метода расчета, описанного в данной статье.

Теперь о том, как определить номер ячейки. Допустим, в массиве *время_осв* уже имеются три значения:

10:45	10:00	11:10	

Обратим внимание на то, что значения в массиве в общем случае не упорядочены (убедитесь в этом!).

Рассмотрим несколько значений времени сдачи багажа очередным пассажиром:

1) 10:30;
2) 10:50;
3) 10:00;
4) 10:45.

В первом случае искомый номер равен 2, во втором — 1 (по условию, выделяется свободная ячейка с *минимальным* номером), в третьем — 2 (в 10:00 ячейка с номером 2 будет освобождена), в четвертом — 1. Обобщая, можем сказать, что нам нужно найти первый при просмотре слева направо элемент массива *время_осв*, значение которого не меньше текущего значения времени сдачи багажа *время_сдачи*. В программе для этого может быть использован оператор цикла с предусловием:

```
номер := 1
нц пока номер <= всего_ячеек и
        время_сдачи < время_осв[номер]
  номер := номер + 1
кц
```

— где *номер* — искомый номер ячейки.

Можно также применить оператор цикла с постусловием

```
номер := 0
нц
  номер := номер + 1
кц_при номер > всего_ячеек или
       время_сдачи >= время_осв[номер]
```

Обратим внимание на использование в обоих случаях сложных условий.

После нахождения номера ячейки, выделяемого очередному пассажиру, нужно вывести фамилию последнего и найденный номер, после чего записать в соответствующий элемент массива *время_осв* время освобождения ячейки данным пассажиром *время_освоб*:

```
|Выводим ответ для текущего пассажира
вывод нс, фам, " ", номер
|Меняем время освобождения
|в найденной ячейке
|время_осв[номер] := время_освоб
```

Однако поскольку возможны случаи, когда очередному пассажиру ячейка не может быть предоставлена, то вывод ответа следует проводить по условию:

```
если номер <= всего_ячеек
  то
    вывод нс, фам, " ", номер
    время_осв[номер] := время_освоб
все
```

Как уже отмечалось, в программах, приведенных в [1], используется оператор, обеспечивающий выход из цикла:

Язык Паскаль

```
for j := 1 to K do {K – количество ячеек)
  if p[j] <= time1
     then begin
            p[j] := time2;
            writeln(name, ' ', j);
            break;
          end;
```

Язык Бейсик

```
FOR i = 1 TO k  'k – количество ячеек
IF time1 >= p(i) THEN
  p(i) = time2
  PRINT nm, i
  GOTO 10
ENDIF
NEXT i
'Конец цикла обработки входных строк
10 NEXT j
END
```

Итак, общая схема программы решения обсуждаемой задачи такая:

```
1. Ввод количества пассажиров
   и числа ячеек
2. Цикл для каждого пассажира:
  2.1. Ввод сведений о пассажире
  2.2. Выделение фамилии
  2.3. Определение времени сдачи багажа
  2.4. Определение времени освобождения
       ячейки
  2.5. Поиск свободной к моменту
       времени сдачи ячейки
       с минимальным номером (цикл)
       Если такая ячейка есть
         то
           Вывод фамилии и номера
           ячейки
           Изменение время освобождения
           найденной ячейки
конец цикла
```

Полностью программа на школьном алгоритмическом языке приведена на диске к данному номеру журнала.

В заключение заметим, что, по-нашему мнению, рассмотренное задание является самым простым заданием демонстрационных вариантов ЕГЭ по информатике нескольких последних лет (сравните, например, с [4])[1].

Литература

1. Демонстрационный вариант контрольных измерительных материалов единого государственного экзамена 2013 года по информатике и ИКТ (проект). *http://www.fipi.ru/binaries/1384/infEGE2013.zip*.

2. Самое полное издание типовых вариантов реальных заданий ЕГЭ: Информатика, 2010 / Авт.-сост. *П.А. Якушкин, Д.М. Ушаков*. М.: Астрель, 2010 (Федеральный институт педагогических измерений).

3. *Якушкин П.А., Лещинер В.Р., Кириенко Д.П.* ЕГЭ-2010. Информатика. Типовые тестовые задания. М.: Изд-во “Экзамен”, 2010.

4. *Златопольский Д.М.* Методика решения задачи С4 из демонстрационного варианта ЕГЭ по информатике 2012 года. / Информатика, № 4, 2012.

[1] **От редакции.** Как показывает опыт предыдущих лет, именно в случае задачи С4 в демоверсию помещаются уже использованные задания, а в основных вариантах следующего года – совсем новые, зачастую не похожие на те, что были в демо.

Лучший способ обучения,

причем не только для ваших учеников

Простые в использовании и эффективные технологии, широкие возможности профессионального роста

Интерактивные инструменты MimioClassroom™ делают уроки яркими и запоминающимися. В итоге ваши ученики начинают значительно лучше усваивать материал. Простота использования оборудования MimioClassroom™ позволяет вам экономить время на подготовке к урокам, а вашим ученикам быстро и увлеченно осваивать новые инструменты.

Учителя, использующие интерактивное оборудование Mimio, получают широкие возможности обучения и повышения квалификации: от бесплатных онлайн-ресурсов, созданных опытными педагогами, до бесплатного обучения специалистами непосредственно в школе. Широкий выбор форм и возможностей обучения позволяет вам вывести свое профессиональное развитие на совершенно новый уровень! А система сертификации поможет подтвердить полученные знания.

Разнообразные возможности обучения и профессионального развития помогут вам разрабатывать интерактивные уроки с использованием MimioClassroom™ более эффективно. А это, в свою очередь, отличный способ учиться — и вашим ученикам, и вам самим!

Узнайте больше, посетите наш сайт или позвоните:

http://www.mimioclass.ru

8 (800) 5555-33-0

Звонок по России бесплатный

mimio

a better way to learn

Стереотехнологии в школе: как и для чего

Д.Ю. Усенков,
Институт информатизации образования Российской академии образования, Москва

Известно, что человек (по разным источникам) от 70 до 90% информации воспринимает при помощи зрения. О важности зрительного восприятия для получения знаний об окружающем мире свидетельствуют многочисленные поговорки (например, “Лучше один раз увидеть, чем сто раз услышать”) и даже изначальный смысл некоторых слов (например, слова “очевидно”). Однако так дело обстоит, когда мы непосредственно наблюдаем за происходящими вокруг нас событиями и процессами, наблюдаем те или иные объекты реального мира. Что же касается учебной информации, представленной в различных формах — от печатной (учебники, энциклопедии, тесты на экране компьютера и т.д.) до мультимедийной (фотографии и рисунки, анимации, кино и видео), то здесь дело обстоит заметно хуже. Понятно, что даже наиболее современные технические средства и носители данных не способны передать всю полноту информации об объектах и явлениях реального мира (вспомним, что любая сохраненная тем или иным способом информация представляет собой лишь информационную модель той или иной части окружающего мира). Но мы, при помощи существующих технологий, должны по крайней мере стремиться передать читателю (зрителю) максимально возможную часть информации! Особенно это важно для информационных ресурсов образовательного назначения, которые учащиеся нередко видят впервые: при недостаточном качестве или полноте материала всегда есть риск формирования искаженного представления об изучаемых явлениях или процессах.

Конечно, изучение возможностей максимально полной передачи всех форм информации при помощи технических средств — тема слишком сложная и объемная. Мы же обсудим только одну ее составляющую, но, пожалуй, наиболее важную — представление визуальной информации не только в цвете, но и в объеме (*стереоскопия*).

Природа дала человеку достаточно богатые зрительные возможности — наше зрение является не только четким и цветным, но и стереоскопическим —

благодаря сложным процессам обработки в мозге информации, которую мы видим левым и правым глазами, мы воспринимаем окружающие объекты трехмерными, можем понять, какие из них расположены ближе, а какие — дальше, а также оценить их протяженность в глубину. (Чтобы понять, насколько это важно, попробуйте провести несложный эксперимент — вдеть нитку в иголку, зажмурив один глаз.) Однако при сохранении и воспроизведении визуальной информации человечество долгое время теряло значительную часть этих возможностей. Сначала для представления внешнего вида различных объектов, интерьеров или пейзажей в распоряжении людей были только рисунки, а затем фотографии: статичные (фиксирующие только один какой-то момент) и монохромные. Позже появилось кино, способное передавать изображение в движении, — но тоже монохромное и к тому же "немое" (без звука), хотя этот недостаток довольно быстро был исправлен. Черно-белым долгое время оставалось и телевидение, да и графические изображения, получаемые на экране дисплея (по крайней мере они были монохромными, хотя и не всегда для отображения текста и картинок применялся именно белый цвет). Затем в разное время появились цветное фото, цветное кино, цветное телевидение и видео, а также возможность получения на дисплее многоцветных изображений (в настоящее время — до нескольких миллионов цветовых оттенков при четкости (разрешении) изображения, не уступающей хорошей фотографии). Однако до недавних пор читателям, зрителям и пользователям компьютеров оставались практически недоступными возможности объемного видения изображений. И вот, наконец, сегодня мы присутствуем при очередной технической революции: хотя сами технологии и методы получения и воспроизведения стереоскопических изображений были в большинстве своем созданы в прошлом столетии, лишь в последние годы в продаже стали появляться серийно выпускаемые (а не сделанные отдельными энтузиастами "на коленке") 3D-фото- и видеокамеры, 3D-телевизоры и 3D-дисплеи.

Поскольку "3D" и "стерео" — это термины, которые в последнее время, что называется, "на слуху", очень многих интересует, что это такое, как работают подобные устройства и что они способны показать зрителю. Учителей же, методистов, а также школьников, их родителей и всех, кто так или иначе имеет отношение к сфере образования, не в меньшей мере волнуют вопросы: что 3D-технологии могут дать школе и не вредно ли их использование для детей. Эти проблемы дополнительно усугубляются из-за того, что большинство существующих технических средств, обеспечивающих стереоскопическое воспроизведение фото- и видеоинформации, нынешней школе, увы, пока еще недоступно — из-за их технической сложности (и потому низкой защищенности от намеренных или случайных разрушительных воздействий со стороны школьников), громоздкости и просто из-за слишком высокой цены (хотя в последние годполтора цены на 3D-устройства начинают очень заметно снижаться и скоро станут доступными большинству граждан). Например, редкая школа (даже из числа "образцово-показательных") может развернуть у себя в актовом зале (не говоря уже об обычных учебных классах) стереокинотеатр "Аймакс 3D" (IMAX 3D).

И тем не менее существует целый ряд технологий, позволяющих при сравнительно невысоких затратах обеспечить возможности реализации трехмерного представления учебного материала как в печатных изданиях, так и в мультимедийных электронных учебных пособиях. Но прежде чем рассматривать их, нужно выяснить, какие механизмы лежат в основе стереоскопического восприятия информации человеком.

Как мы видим то, что видим

Хотя процессы, происходящие в мозге человека при обработке поступающей от глаз зрительной информации, еще не изучены во всей их полноте, некоторые основные аспекты уже известны и достаточно просты. Основа стереоскопического восприятия лежит в сравнении мозгом двух изображений, поступающих с левого и правого глаза. Зрительные поля (области зрения) обоих глаз человека частично перекрываются, но видимое изображение имеет некоторые отличия для левого и правого глаза, так как они разнесены по горизонтали на расстояние примерно в 60–70 см (*расстояние стереобазиса*) и смотрят на более близкие предметы "с разных боков" (*рис.* 1).

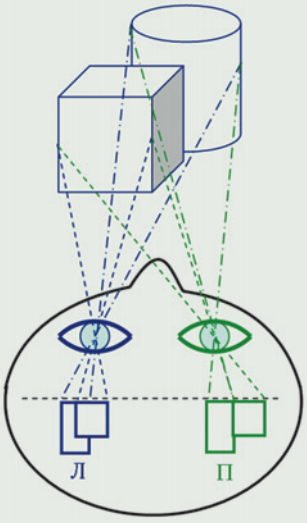


Рис. 1.
Различие видимых левым и правым глазом изображений при бинокулярном зрении. (Условно показано изображение, формируемое на сетчатке глаз, при этом для простоты мы считаем ее плоскостью. Получаемые таким способом изображения — перевернутые. Можно также отметить, что полученные здесь "левое" и "правое" изображения представляют собой перекрестную стереопару.)

Мозг извлекает из поступающей с обоих глаз информацию (для этого служит имеющееся в мозге особое перекрестье нервных волокон — *хи-*

азма) о контурах наблюдаемых объектов и "подстраивает" положение глаз так, чтобы в основном контуры совпадали, — после чего по углам поворота глаз (*угол конвергенции*) при их фокусировании в одной точке наблюдаемого изображения определяет относительное расстояние до этой точки, а по расхождению отдельных контуров получает информацию о том, что те или иные объекты располагаются ближе или дальше, чем выбранная точка фокусировки.

Конечно, задействовать этот "механизм" для получения стереоизображения мы, не вмешиваясь напрямую в работу мозга, не имеем возможности. Но можно сымитировать условия рассмотрения реальных объектов, выдавая мозгу "правильную входную информацию"!

Наиболее совершенным способом такой "имитации" является *голография*: лазерное излучение при голографической съемке создает в светочувствительном слое сложную интерференционную картину, при освещении которой таким же по физическим характеристикам лазерным лучом формируется идентичная реальной система световых волн. Ее человек (наблюдатель) воспринимает точно так же, как воспринимал бы реальный объект. Современные компьютерные технологии, кстати, уже позволяют синтезировать "на лету" требуемую интерференционную картину для того или иного сформированного в памяти ПК трехмерного изображения, при освещении которой лазерным лучом можно получить голографическое изображение достаточно высокого качества. Но голографические технологии пока еще слишком сложны и дороги, особенно при попытках создания голографических кинофильмов, поэтому для школы (да и вообще для бытового применения) пока недоступны.

Другой способ, хотя и обеспечивающий не настолько высокое качество восприятия объема, но зато доступный в реальных, не "лабораторных" условиях и потому используемый в подавляющем большинстве современных стереоскопических технологий, — это формирование *стереопары*. Для этого требуемый объект, интерьер или пейзаж фотографируется (либо снимается на видео) двумя фотокамерами (видеокамерами), расположенными по горизонтали на расстоянии, соответствующем среднему расстоянию между глазами человека (пример стереопары показан на *рис.* 2). При просмотре отснятого таким способом изображения тем или иным способом на левый глаз подается для рассматривания именно левый отснятый кадр, а на правый глаз — именно правый кадр (синхронизированная во времени пара таких кадров и называется стереопарой). А далее — все зависит от того, как осуществляется такое разделение кадров стереопары между глазами.

Среди существующих методов просмотра стереопар можно назвать следующие: рассматривание через стереоскоп (либо невооруженным глазом), анаглифический метод, растровый, поляризационный и затворный.

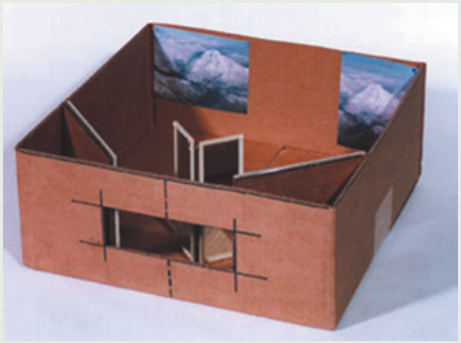
Стереоскоп

Самый простой способ разделения кадров стереопары — непосредственно расположить левый кадр напротив левого глаза (но так, чтобы этот кадр не было видно правым глазом), а правый кадр — напротив правого глаза. Для этой цели служат различные стереоскопы (*рис.* 3, *а–г*).

Рис. 2. Стереопара (Большой каскад в Петергофе)

а) старинный стереоскоп

б) стереоскоп, выпускавшийся в составе комплекта для стереофотографии ленинградским заводом ЛОМО (СССР)



в) самодельный картонный стереоскоп с зеркальной системой

г) стереоскопы для просмотра стереослайдов

Рис. 3. Различные конструкции стереоскопов

Конструкции таких стереоскопов могут быть различными: используются линзы (чтобы сделать изображение четким при его близком размещении перед глазами), зеркала или призмы (чтобы “свести” просматриваемые кадры стереопары в стереоизображение, визуально “накладывая” их друг на друга; такие стереоскопы чаще всего используются сегодня для просмотра стереопар на экране дисплея).

Невооруженным глазом

Стереопары (фотографии, напечатанные в газете, журнале или книге иллюстраций и даже видеоизображения на экране телевизора или дисплея) можно приноровиться рассматривать и без стереоскопа. Для этого нужно только использовать так называемую “перекрестную” стереопару (в которой левый и правый кадры поменяли местами) и постараться так скосить глаза, чтобы визуально “наложить” друг на друга левый и правый кадры (*рис.* 4).

Научиться так рассматривать стереопары не очень сложно, если воспользоваться следующим упражнением.

Упражнение

Посмотрите на *рис.* 5. Здесь тоже изображена “стереопара”, только не сфотографированная, а нарисованная.

Глядя на нее, вытяните перед собой руку с выпрямленным указательным пальцем. Свой па-

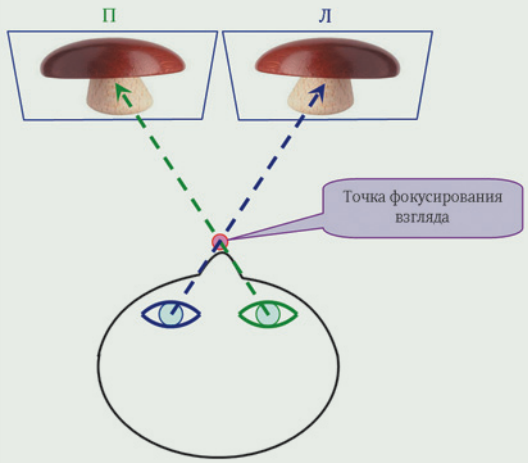


Рис. 4. Просмотр перекрестной стереопары

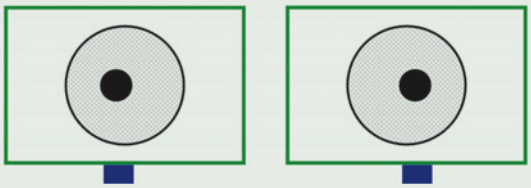


Рис. 5. “Учебная стереопара”

лец вы будете видеть как бы “раздвоившимся”. Расположите палец между своим лицом и экраном с рисунком-“стереопарой” на таком расстоянии, что “раздвоившиеся копии” вашего пальца будут приходиться напротив нарисованных под зелеными рамками темно-синих прямоугольничков.

Теперь переведите взгляд на свой палец. Его “раздвоившиеся копии” сольются в одно четкое изображение. А за ним на экране компьютера вы увидите вместо бывших в “стереопаре” двух рамочек-“кадров” сразу три.

А теперь — самое сложное. Переведите (перефокусируйте) взгляд с пальца на “стереопару” так, чтобы по-прежнему видеть на ней три рамочки — одну, более четкую, в середине и две плохо заметных, “мерцающих” — по бокам. Если эта средняя рамочка у вас “двоится”, то попробуйте передвигать палец чуть ближе или чуть дальше от себя либо немного наклонить голову вправо/влево, чтобы устранить это “двоение”.

Посмотрите внимательно на среднюю из трех рамочек. Это и есть стереокартинка: на ней черный кружочек явственно “висит в воздухе” перед серым кругом, зеленой рамкой и вообще плоскостью экрана. А две картинки по бокам — это “паразитные” изображения, на которые нужно просто не обращать внимания. (Если это пока для вас трудно, то можно загородить их руками.)

Если сразу увидеть стереоэффект не получается, не расстраивайтесь. Потренируйтесь еще, добиваясь результата (только обязательно делайте перерывы, если почувствуете, что глаза устали). Просмотр стереопар невооруженным глазом — это *навык*, которому сложно научить. Его можно только выработать самому путем тренировки. А когда вы, наконец, научились переводить взгляд с пальца на стереопару, не теряя получающегося в ней стереоизображения, попробуйте сделать то же самое “без пальца”, просто скашивая глаза к носу и добиваясь, чтобы кадры “стереопары” совмещались перед вами в среднюю картинку — стереоизображение.

Впрочем, таким способом можно просматривать лишь отдельные стереофотографии (и то — “мимолетным взглядом”), но не стереофильмы: глаза очень быстро устают.

Анаглифический метод

Если использование стереоскопа (равно как и просмотр перекрестной стереопары скашиванием глаз) основано на пространственном разделении кадров стереопары, то анаглифический метод основан на их цветовом разделении. Для этого используются два цветовых оттенка, взаимодополняющих друг друга до черного, — обычно красный и голубой (Cyan). Стереопара представляет собой два монохромных кадра, где один кадр (например, левый) выполняется в красном цвете, а второй — в голубом, и эти две картинки накладываются друг на друга. Возможно и получение таких стереокартинок на основе цветных фотографий, в которых цветовые гаммы соответственно искажаются в красный и в голубой цвет. Получаемое в результате изображение напоминает широко известный полиграфический брак со смещением печатных матриц для разных красок: контуры объектов на такой картинке “двоятся” тем сильнее, чем изображаемый объект находится ближе или дальше от основной плоскости картины (*рис.* 6).

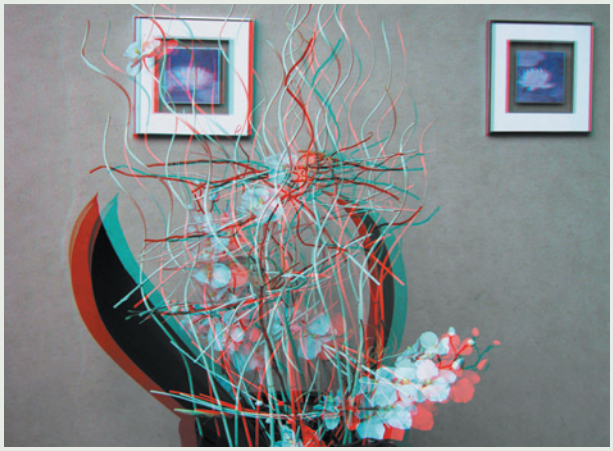


Рис. 6. Пример анаглифического изображения

Соответственно, рассматривать такое изображение нужно через специальные очки, стекла (пленки) в которых тоже окрашены в те же цвета: для приведенного выше примера левое стекло должно быть окрашено в голубой цвет, а правое — в красный (*рис.* 7). Тогда все красные оттенки на “суммар-

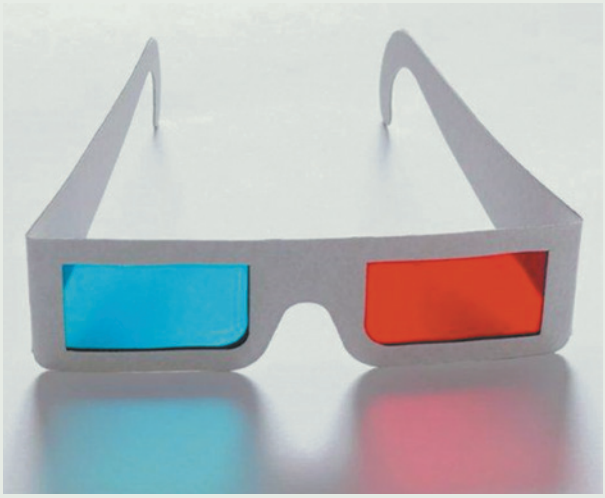


Рис. 7. Анаглифические (“двухцветные”) очки

ной” анаглифической картинке будут видны сквозь голубое стекло как оттенки серого соответствующей плотности, а голубые оттенки сквозь голубое же стекло будут восприниматься как белые. Поэтому левый глаз будет видеть черно-белую левую картинку (которая в “смеси” изображений была красной), но не увидит правую (голубую). И наоборот, правый глаз сквозь красное стекло увидит монохромную правую картинку (которая была в “смеси” изображений голубой), но не увидит левую (красную). В итоге наблюдатель будет воспринимать объемное изображение, хотя оно будет или монохромным (хотя полиграфия — цветная!), или по крайней мере иметь сильно искаженные цветовые оттенки (*рис.* 8). Зато двухцветные анаглифические очки проще, легче и дешевле, чем стереоскоп, их можно надевать поверх обычных очков (хотя это менее удобно), глаза при этом устают заметно меньше, а рассматривать можно стереокартинки любого размера и на любом носителе — на бумаге, настенном экране или дисплее. Купить такие очки сегодня можно во многих местах, а можно и сделать их самому [1].

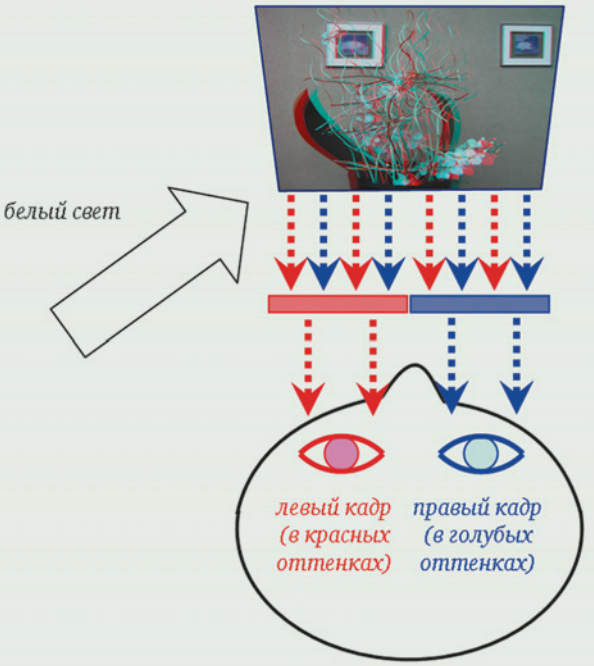


Рис. 8. Анаглифический способ воспроизведения стереоизображений

Растровый метод

Этот метод позволяет создавать стереоскопические изображения как в печатном виде, так и на экране телевизора или дисплея, причем для его просмотра не требуется никаких очков или приспособлений.

Существует два варианта растрового метода. В любом из них два кадра стереопары (левый и правый) разрезаются на узкие полоски (обычно вертикальные) толщиной в 1 пиксель, и из них формируется интегральное изображение, в котором полоски чередуются: например, нечетные берутся из левого кадра, а четные — из правого. В результате получается картинка, подобная *рис.* 9.

Рис. 9. Интегральное изображение, подготовленное для создания стереокартинки по растровому методу

А далее в первом случае поверх такой картинки (или на экран дисплея, на который выводится такая картинка) наклеивается специальная ребристая пленка из треугольных призм или цилиндрических линз. Для просмотра такого изображения нужно найти в пространстве “правильную” точку, при взгляде из которой за счет преломления световых лучей в призмах или линзах левый глаз будет видеть только полоски от левого кадра, а правый глаз — только полоски от правого кадра. В результате формируется стереоэффект (*рис.* 10).

Во втором случае (такая технология носит название “параллаксный барьер” и применяется в электронных устройствах отображения) используются два ЖК-экрана. Первый экран — обычный, цветной, на который можно выводить как обычные растровые изображения (в режиме 2D), так и интегральные, собранные из полосок (как было сказано выше). Второй экран — монохромный, прозрачный, на нем можно выводить (или не выводить) сетку из тонких вертикальных черных полосок. Этот экран может размещаться как перед основным, так и за ним (между основным экраном и осветительной лампой). Тогда если черные (непрозрачные!) полоски выведены, то они (опять-таки при просмотре из “правильной” точки) загораживают от просмотра левым глазом полоски от правого кадра и наоборот, загоражи-

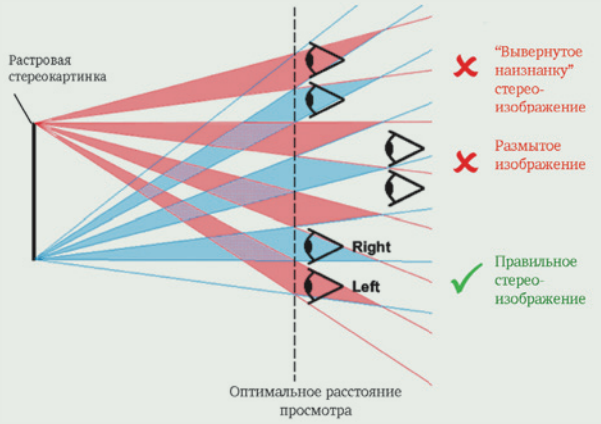


Рис. 10. Поиск правильной точки наблюдения при рассматривании растрового стереоизображения (иллюстрация с сайта *http://www.3d-forums.com*)

вают от правого глаза полоски от левого кадра (*рис.* 11). В результате левый глаз видит только левый кадр, а правый глаз — только правый, и формируется стереоэффект.

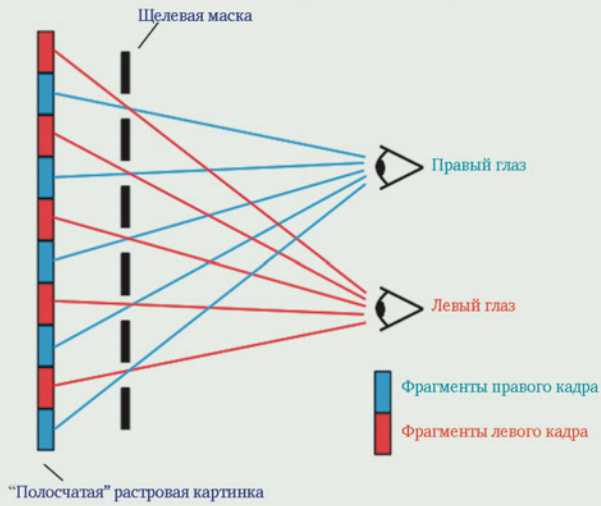


Рис. 11. Получение стереоизображения при использовании щелевой маски (рисунок с сайта *http://www.3d-forums.com*)

Основной недостаток обоих описанных вариантов растрового метода — необходимость искать в пространстве точку наблюдения, в которой реализуются условия просмотра стерео (впрочем, делается это очень легко). Зато отсутствие необходимости в использовании очков — это бесспорное преимущество. Учитывая же, что в устройствах с параллаксным барьером режим 3D (т.е. этот самый параллаксный барьер) можно при желании отключить, и тогда устройство работает как обычный медиаплеер или "электронная книга", а цена таких устройств может быть весьма невысокой (например, простейший 3D-плеер QUMO Reality 3D китайского производства в августе 2012 г. в Москве можно было приобрести всего за 3500 руб.), данный метод получения стерео можно считать весьма перспективным для применения в школьных "электронных учебниках" (ридерах).

Поляризационный метод

Здесь получение стереоэффекта основано на физическом эффекте поляризации света. Как вы помните из курса физики, свет (как электромагнитная волна) может быть поляризован (колебания строго в одной плоскости) при помощи поляризационного фильтра. После этого при пропускании такого поляризованного света через еще один такой же поляризационный фильтр свет может проходить через него практически без потери яркости, если этот фильтр имеет такую же ориентацию плоскости поляризации, как и свет (т.е. как фильтр, с помощью которого этот свет был поляризован). И наоборот, поляризованный свет будет полностью поглощен вторым фильтром, если его плоскость поляризации перпендикулярна плоскости поляризации света.

Кроме описанного эффекта плоской поляризации, существует эффект круговой поляризации света, когда световая волна может быть поляризована с вращением плоскости ее поляризации вокруг продольной оси по часовой либо против часовой стрелки (*рис.* 12). При этом тоже могут быть использованы два вида поляризационных фильтров: если свет поляризован, например, фильтром круговой поляризации по часовой стрелке, то он без потерь пройдет через такой же фильтр, но будет поглощен фильтром с поляризацией против часовой стрелки, и наоборот.

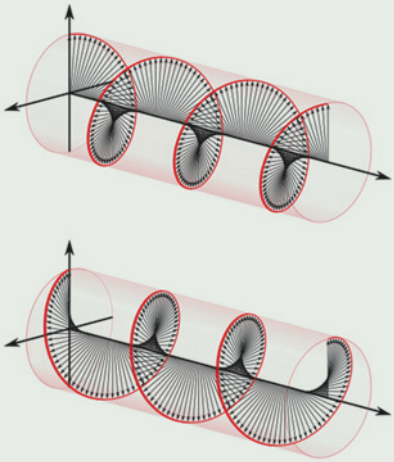


Рис. 12. Эффект круговой поляризации

Именно на таком эффекте поляризации (обычно — круговой) и работает поляризационный метод получения стереоизображений (например, он лежит в основе функционирования 3D-телевизоров LG Cinema 3D). Здесь левый и правый кадры стереопары, как и в растровом методе, разрезаются на полоски толщиной в 1 пиксель, но горизонтальные. Далее на ТВ-экране (или экране дисплея) формируется интегральное изображение, в котором, например, четные ТВ-строки взяты из левого кадра, а нечетные ТВ-строки — из правого. Соответственно, на экран нанесено покрытие из поляризационных фильтров: напротив

пикселей четных строк, например, с поляризацией по часовой стрелке, а напротив пикселей нечетных строк — с поляризацией против часовой стрелки. Зрители же должны для просмотра стереоизображений надеть очки тоже с поляризационными фильтрами: для левого глаза — с поляризацией по часовой, а для правого глаза — против часовой стрелки. Тогда, очевидно, левый глаз будет видеть только строки из левого кадра, а правый — только из правого кадра.

Стоимость таких стереотелевизоров пока довольно существенна, но за прошедший год уже снизилась и продолжает снижаться, а поляризационные очки не очень дороги и практически не напрягают глаза. Поэтому можно предполагать, что стереотелевизоры типа Cinema 3D вскоре будут широко использоваться в школах как устройство отображения информации для показа всему классу. Тем более что в современных таких телевизорах “встроен” алгоритм (на базе технологий распознавания изображений и элементов искусственного интеллекта), позволяющий достаточно правдоподобно преобразовывать в стереоизображение и обычные “плоские” изображения и видео, так что просмотр в 3D возможен даже для “старых” учебных материалов, снятых в “моно”-варианте.

Существуют, кстати, и 3D-медиапроекторы, работающие по этому принципу: либо в виде “спарки” из двух обычных проекторов с поляризационными фильтрами на объективах, которые накладывают проецируемые изображения друг на друга, либо с вращающимся поляризационным фильтром, который “подставляется” под попеременно выводимые левый и правый кадры. Такие 3D-проекторы пока еще дороги, но, видимо, рано или поздно тоже будут попадать в школы.

Затворный, или активный, метод

Наконец, еще один метод получения стерео, применяемый в большинстве современных 3D-телевизоров, а также часто используемый на компьютере, основан на простом загораживании “ненужных” кадров от просмотра. При этом на экран попеременно выводятся то левый, то правый кадр, а зритель должен надеть специальные очки со стеклами из прозрачных ЖК-экранов. Когда на экране телевизора отображается левый кадр стереопары, в очках левое стекло прозрачно, а правое затемняется до непрозрачности. И наоборот, при показе на экране правого кадра правое стекло становится прозрачным, а левое затемняется. (За этим попеременным миганием стекол очков следит встроенная в них электронная схема, синхронизированная с сигналом смены кадров в телевизоре.)

Такой метод при его кажущейся простоте менее удобен, чем поляризационный или чем параллаксный барьер. Во-первых, телевизор или дисплей должен обеспечивать высокую частоту смены кадров (вдвое выше, чем обычный). Во-вторых, при просмотре может возникать мерцание, утомляющее глаза. Наконец, в-третьих, очки для просмотра стерео в этом случае содержат электронную схему (что сильно удорожает их по сравнению с поляризационными) и аккумулятор (который надо периодически заряжать), и к тому же они заметно тяжелее. Поэтому в школах этот метод вряд ли будет использоваться широко.

Что это дает?

Как уже было сказано, стереоизображения имеют заметно бóльшую наглядность и создают в какой-то мере “эффект присутствия” (не говоря уже о повышении интереса к материалу со стороны школьников). А в некоторых случаях стереоскопичность и вовсе является жизненно необходимой. Вот, например, изображение молекулы белка (*рис.* 13; взят из одного из старых номеров журнала “Наука и жизнь”): увидеть в этом хаосе линий истинную пространственную структуру молекулы можно только воспользовавшись стереотехнологиями.

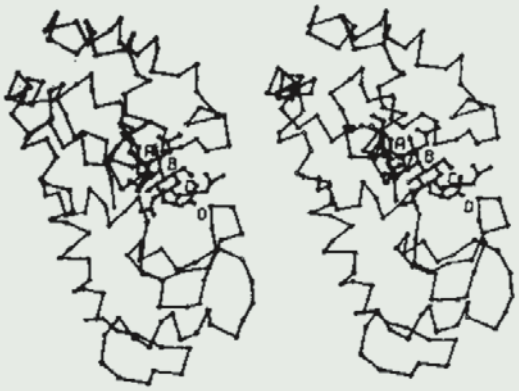


Рис. 13. Перекрестная стереопара молекулы белка

Точно так же стереотехнологии могут быть очень и очень полезны при изучении курса стереометрии (уже само название которого подсказывает нам эту идею!), черчения и основ САПР и в других подобных случаях. Но и не только! Например, любая схема, фотография или видео — будь то рисунок “устройства” живого организма в биологии, фото лабораторной установки или видеозапись эксперимента в физике или химии, схема или фото какого-нибудь периферийного устройства в курсе информатики и т.д. — только выиграет в наглядности, если будет просматриваться в объеме!

Вредно или полезно?

С появлением и все более широким распространением 3D-устройств разгораются споры о том, вредно ли ими пользоваться. Одни утверж-

дают, что “3D портит глаза” и даже может приводить к эпилепсии, другие заявляют о безопасности просмотра 3D, поскольку мы (по крайней мере в подавляющем большинстве) с рождения и всегда видим окружающий мир в объеме и ничуть от этого не страдаем.

Истина лежит, как всегда, посередине. Во-первых, все зависит от используемой технологии просмотра. Скашивание глаз, анаглиф и отчасти затворный метод действительно утомляют глаза, так что смотреть таким способом стерео можно только ограниченное время (при скашивании глаз и использовании анаглифических очков — вообще не более 3–5 минут). А вот стереоскоп, растровый метод и особенно поляризационный на утомляемость глаз практически не влияют.

Что же касается индивидуальных проблем при просмотре стереоизображений (например, у некоторых зрителей может начать болеть голова), то здесь, возможно, просмотр 3D лишь выявляет заболевания органов зрения, которые сложно выявить при обычном медосмотре у окулиста, и возникновение проблем при просмотре 3D — это хороший повод обратиться к врачу. Например, специалисты Американской оптометрической ассоциации [2] провели исследование проблемы применения 3D-технологий в сфере образования и подготовили соответствующий доклад под названием “3D in the classroom” (“3D в школе”). В рамках этих исследований были сделаны следующие выводы:

• ограничения по времени просмотра 3D не отличаются от таких же ограничений для 2D, но при работе с мобильными устройствами из-за более близкого расстояния просмотра требуются более частые перерывы;

• вредность 3D — это, скорее, миф, а тем, кто испытывает дискомфорт при просмотре 3D-видео, рекомендуется пройти обследование у врача-офтальмолога, так как этот дискомфорт может быть признаком скрытых заболеваний глаз, не проявляющихся при просмотре 2D-изображений;

• у большинства детей бинокулярное зрение формируется к трем годам, и с этого возраста они уже могут безопасно наслаждаться 3D-фильмами, телевидением и играми;

• для больных эпилепсией нет особой разницы между 2D и 3D — в обоих случаях вероятность возникновения приступа одна и та же и требуется соблюдать одинаковую осторожность.

Таким образом, применять 3D-технологии в образовательном процессе для повышения наглядности учебных материалов можно начинать уже сегодня.

• Издательства вполне могут начать выпуск для школ дополнительной литературы (альбомов наглядных пособий) с “встроенным” стереоскопом либо включать в печатные издания (учебники, пособия) перекрестные стереопары и анаглифические изображения.

• Фирмы, выпускающие электронные пособия, могут начать выпуск тематических комплектов электронных ресурсов (“электронных наглядных пособий”) и других видов ЦОР с применением 3D-технологий. Для этого нужно осуществлять съемку фото- и видеоизображений либо вычерчивание графических изображений в стереоформате и размещать в составе ЦОР стереопары, а также обеспечить программные средства для просмотра таких стереопар в различных режимах (прямая/перекрестная стереопара, анаглиф, растровый, поляризационный метод) либо преобразования стереопар в интегральные (для переноса на устройства просмотра по растровой или поляризационной технологии), тем более что такие программы уже есть. Например, свободно распространяемая программа StereoSlide Show реализует просмотр стереопар в разных технологиях, а свободно распространяемая программа StereoPhoto Maker выполняет преобразование стереопар в любой требуемый формат (обе программы доступны на сайте *http://stereo.jpn.org*).

• Методисты же и учителя могут (в том числе активно привлекая учащихся в ходе проектной работы) самостоятельно создавать стереоматериалы учебного характера и обмениваться ими с коллегами (например, если кто-либо возьмет на себя труд создать соответствующий сайт). О том, как самому снимать и подготавливать стереоизображения, мы расскажем в последующих статьях.

А напоследок — подскажем читателям еще один метод просмотра обычных “плоских” фото- и видеоизображений с “эффектом стерео”, о котором писал еще Я.И. Перельман в своем двухтомнике “Занимательная физика”. Достаточно при просмотре фотографии или видео зажмурить один глаз, а затем “поварьировать” расстояние от зрителя до просматриваемого изображения (оно зависит от фокусного расстояния объектива при съемке и от масштаба увеличения фотографии или кинокадра), и в какой-то момент просматриваемое изображение если и не приобретет полноценную “глубину”, то его “стереоскопичность” (а значит, и наглядность) заметно возрастет…

Литература

1. *Богомолова О.Б., Усенков Д.Ю.* Искусство презентации: практикум. М.: БИНОМ. Лаборатория знаний, 2010.

2. 3D позволяет делать раннюю диагностику зрения у детей [электронный ресурс]. URL: *http://www.gosbook.ru/node/34412*.

3. Журнал “Мир 3D / 3D World” [электронный ресурс]. URL: *http://mir-3d-world.narod.ru*.

журнал

Информатика – Первое сентября

ТАРИФНЫЕ ПЛАНЫ НА ПОДПИСКУ

1-е полугодие 2013 года

Максимальный – 1254 руб.

бумажная версия (по почте) + CD + доступ к электронной версии на сайте

Оформление подписки – на сайте www.1september.ru или на почте по каталогам:
«Роспечать» – индекс 32291 *(для индивидуальных подписчиков и организаций)*
«Почта России» – индекс 79066 *(для индивидуальных подписчиков и организаций)*

Оптимальный – 594 руб.

электронная версия на CD (по почте) + доступ к электронной версии на сайте

Оформление подписки – на сайте www.1september.ru или на почте по каталогам:
«Роспечать» – индекс 19179 *(для индивидуальных подписчиков и организаций)*
«Почта России» – индекс 12684 *(для индивидуальных подписчиков и организаций)*

Экономичный – 300 руб.

доступ к электронной версии и оформление подписки на сайте www.1september.ru

Бесплатный – 0 руб.

доступ к электронной версии на сайте www.1september.ru для педагогических работников образовательных учреждений, участвующих в Общероссийском проекте «Школа цифрового века» – см. вкладку

Бумажная версия (доставка по почте)

CD с электронной версией журнала и дополнительными материалами для практической работы (доставка по почте)

Электронная версия в Личном кабинете подписчика на сайте www.1september.ru Дополнительные материалы включены

Пользователям электронной версии высылаются по почте подтверждающие документы

ЭКОНОМИЧНЫЙ тарифный план

ОПТИМАЛЬНЫЙ тарифный план

МАКСИМАЛЬНЫЙ тарифный план

При оформлении подписки на сайте www.1september.ru оплата производится по квитанции в отделении банка или электронными платежами on-line

VISA MasterCard QIWI WebMoney

Структуры данных. Алгоритмический подход

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

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

“**Азбука Роботландии**” — это курс информатики для начальной школы.

Первая часть курса — “**Компьютер**” — знакомит первоклассников с базовым инструментом современной информатики (см. “Информатику” № 14 за 2011 год). Вторая часть — “**Информация**” — формирует у второклассников общие представления об информации и информационных процессах и знакомит с такими популярными информационными структурами, как *списки, таблицы, иерархии*.

Алгоритмическая линия курса присутствует на уроках как инструмент изложения материала и как среда для практической работы с новыми понятиями.

В этой заметке сначала в качестве методического материала приводится классификация структур данных, а затем рассказывается, как работа со списками, таблицами и иерархиями представлена в курсе “Азбука Роботландии. Информация”.

Посмотреть демоверсии учебника и методички можно на прилагаемом диске, а также со страницы сайта *www.robotlandia.ru/abc.htm*.

Кроме того, на диске, сопровождающем номер, находится подборка электронных практикумов учебника. Авторы надеятся на любопытство читателя, который непременно откроет этот материал и поработает с ним.

Теоретическое введение

Информацию стараются хранить так, чтобы в ней можно было легко ориентироваться, то есть быстро находить нужный информационный элемент. Информационная свалка никому не нужна. Поэтому информацию *структурируют*, то есть записывают по определенной схеме.

Информационный элемент

Информация (как и материя) не является монолитной конструкцией, а делится на составные части. Так, текстовое сообщение можно разделить на предложения, предложения — на слова, слова — на буквы.

Под **информационным элементом** будем понимать выделенную по какому-либо признаку часть информации и рассматривать ее как материал для построения информационных конструкций.

Информационный элемент будем называть **атомом**, если дальнейшее его разделение невозможно. Так, в текстовом сообщении информационными атомами являются отдельные символы.

При разделении информационного блока на атомы часто используют не только “физическое” разделение (на символы), но и логическое (на лексемы). Так, например, при трансляции программ в машинные коды ключевые слова языка программирования, такие как `ЕСЛИ`, `ТО`, `ИНАЧЕ`, `ПОКА`, считаются атомами (**лексемами**).

Множество

Простейший способ упорядочения информационных элементов — объединение их в *множество*.

Множество — такое же первичное понятие, как информация. Определение множеству дать нельзя, но можно сказать, что множество — это *собрание элементов*.

Можно говорить о множестве всех книг библиотеки, множестве натуральных чисел, множестве компьютеров, образующих Интернет, множестве всех чисел на прямой, множестве решения уравнения, множестве учеников в классе.

Чтобы задать множество, нужно просто перечислить его элементы или указать **характеристический признак**, по которому элементы собираются в множество.

Множество может оказаться конечным (множество цифр), бесконечным (множество натуральных чисел) или пустым (множество коров, которые летают подобно птицам).

Порядок следования элементов множества не важен.

Линейный список

Под *списком* “Азбука Роботландии” понимает именно **линейный список** — *конечное упорядоченное* множество элементов.

Итак, линейный список — это множество, которое имеет конечное число элементов, и элементы в нем следуют в определенном порядке. Например, множество всех натуральных чисел списком не является. А множество цифр является списком, если в нем зафиксирован порядок следования (например, по возрастанию).

Математически список обозначается простым перечислением своих элементов в заданном порядке:

$L = (l_1, l_2, \ldots, l_n)$

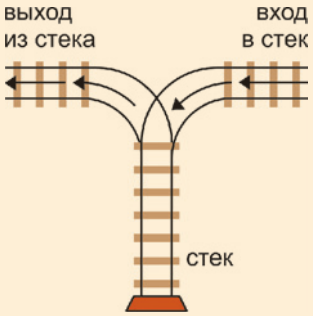
В информатике очень часто встречаются линейные списки, в которых включение и исключение элементов выполняются только через начало или конец списка. Такие списки получили специальные названия: **стек** и **очередь**.

Стек

Стек — это линейный список, в котором включения и исключения элементов делаются в конце списка (**верхушке стека**).

Например, стек служит для хранения чисел в стековом калькуляторе. Числа помещаются в это хранилище по одному, подобно тому, как кольца детской пирамидки нанизываются на стержень. За один раз взять из стека можно только одно число с вершины числовой пирамиды (конца списка). Копия числа в стеке не сохраняется, зато становится доступным следующее число, расположенное “ниже” (“левее”) взятого.

Можно представить стек как железнодорожный разъезд с тупиком (аналогия, предложенная Э.Дейкстрой). Вагоны поступают в стек в прямом порядке, а уходят в обратном (первым покидает стек вагон, пришедший в него последним).

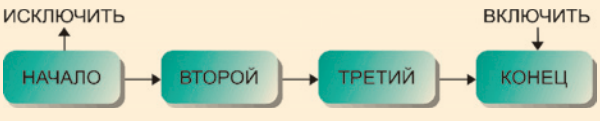


О стеке говорят как об информационной структуре **LIFO** (***L**ast-**I**n-**F**irst-**O**ut* — последним пришел, первым вышел).

Типичным примером использования стека в информатике может служить ситуация с отложенной обработкой: необработанные элементы помещаются в стек, а затем обрабатываются с удалением до тех пор, пока стек не станет пустым. Например, вызовы процедур при выполнении программ помещаются в стек. При глубокой рекурсии стек может оказаться настолько большим, что очень скоро вся доступная память может быть исчерпана.

Очередь

Очередь — это линейный список, в котором включения выполняются в одном конце списка (**конец очереди**), а исключения — в другом (**начало очереди**).



Очередь как информационная структура имеет полную аналогию с бытовым понятием очереди, например, в кассу магазина для оплаты покупок. В силу этого очереди получили широкое распространение при моделировании на компьютере систем массового обслуживания.

Об очереди говорят как об информационной структуре **FIFO** (***F**irst-**I**n-**F**irst-**O**ut* — первым пришел, первым вышел).

Список

В общем случае список (в отличие от линейного списка) может иметь иерархическую структуру:

ведь можно рассматривать списки, элементами которых являются списки.

Введем такое определение. Список, которому поставлено в соответствие некоторое имя (обозначение), будем называть именованным. Имя будем записывать перед скобками, в которые заключены элементы списка.

Список десятичных цифр можно записать в виде такого именованного списка:

```
Цифра(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
```

Фактически именованный список — это уже двухуровневая иерархия. Для списка цифр, например, имя `Цифра` служит родителем (и корнем), а сами цифры — потомками (и листьями).

Теперь легко переписать в виде списка любую иерархию.

Дерево

```
Книга
  Глава 1
    §1
    §2
    §3
  Глава 2
    §1
    §2
```

записывается списком так:

```
Книга(Глава 1(§1, §2, §3), Глава 2(§1, §2))
```

Дерево

преобразуется в список так. Сначала получаем двухуровневую иерархию:

```
Америка(Северная, Южная)
```

Затем трехуровневую:

```
Америка(Северная(Канада, США, Мексика),
Южная(Бразилия, Аргентина, Перу))
```

И, наконец, четырехуровневую:

```
Америка(Северная(Канада, США(Нью-Йорк,
Колорадо, Техас), Мексика),
Южная(Бразилия, Аргентина, Перу))
```

Список является наиболее удобным способом представления иерархии в компьютерной памяти: ведь компьютерная память — это тоже список (ячеек).

Заметим, что именованный список (введенный здесь для наглядности представления) легко переводится в обычный, если имя записывать как первый элемент. Например, именованный список:

```
A(B, C(D, E))                    (*)
```

переписывается как:

```
(A, B, (C, D, E))                (**)
```

Легко видеть, что это преобразование взаимно однозначно. То есть не только по записи (*) можно построить единственную запись (**), но и, наоборот, по записи (**) запись (*) восстанавливается однозначно.

Хранить иерархию можно и в виде таблицы, но это нерационально: в такой таблице будет много пустых клеток (о том, как хранятся в компьютерной памяти таблицы, рассказано ниже).

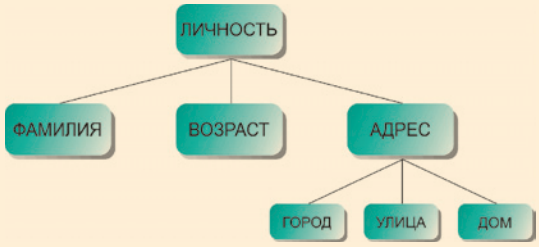
Иерархия

Иерархия (древовидная структура, дерево) — это такая организация информационных элементов, в которой отражаются отношения подчинения или вхождения между элементами.

Можно сказать, что иерархия — это именованный список. Имя списка называется корнем, а элементы списка — потомками.

Каждый потомок, в свою очередь, может снова быть иерархией. Если это не так, то он называется листом.

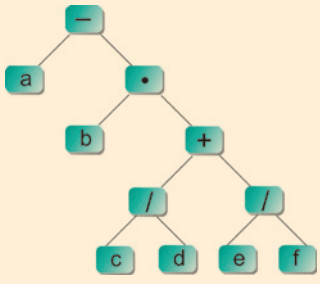
Иерархия как информационная структура довольно популярна в информатике, так как отражает естественную логику взаимосвязей объектов в окружающем мире. Например, для описания отдельных личностей в некоторых случаях вполне может подойти такая структура:



А вот пример записи выражения

```
a – b · (c / d + e / f)
```

в виде дерева:



Строгое определение иерархии — рекурсивное. Вот как, например, определяется иерархия в книге Д.Кнута “Искусство программирования для ЭВМ” (т. 1 “Основные алгоритмы”):

> Иерархия — это конечное множество T, состоящее из одного или более узлов, таких, что:
>
> a) Имеется специально обозначенный узел, называемый корнем данного дерева.
>
> b) Остальные узлы (исключая корень) содержатся в $m \geq 0$ *попарно непересекающихся* множествах T_1, ..., T_m, каждое из которых, в свою очередь, является деревом. Деревья T_1, ..., T_m называются поддеревьями данного корня.

Это определение явно исключает ситуацию, при которой узел может иметь более одного родителя.

В самом деле, если бы узел имел двух родителей, множества узлов этих родителей имели бы в пересечении этот узел, что противоречит определению (“попарно непересекающиеся множества”).

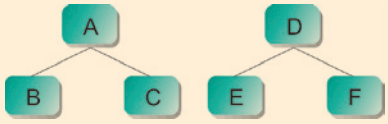
Лес

Лес — это список, состоящий из некоторого числа деревьев, не имеющих общих узлов.

Пример леса:

```
(A(B, C), D(E, F))
```

Тот же лес, изображенный графически:

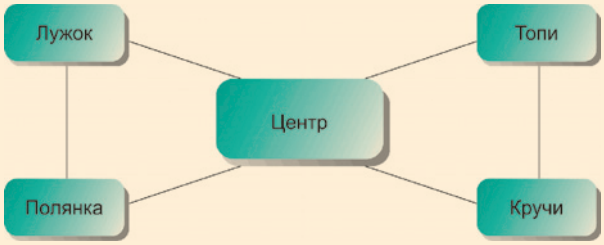


Легко заметить, что узлы любого дерева, за исключением корня, образуют лес. И наоборот, если к любому лесу добавить общий корень, получится дерево.

Пример леса из прикладной информатики — список деревьев, описывающих отдельные личности в рамках принятой информационной модели.

Граф

Граф — это множество информационных элементов (вершин, узлов) и связей (ребер) между ними. Примером графа служит карта дорог, где города — это вершины, а дороги — это ребра.



В информатике графы часто задают в виде таблицы смежности. Заголовки строк и столбцов в этой таблице — название вершин. Клетка $M_{i,j}$ содержит 1, если i-я вершина связана ребром с j-й вершиной, и 0 в противном случае.

Ниже представлена таблица смежности для приведенного выше дорожного графа.

	Центр	Лужок	Полянка	Топи	Кручи
Центр	0	1	1	1	1
Лужок	1	0	1	0	0
Полянка	1	1	0	0	0
Топи	1	0	0	0	1
Кручи	1	0	0	1	0

Понятно, что дерево является частным случаем графа.

Таблица

Таблицу можно рассматривать как список, состоящий из n списков, каждый из которых содержит m элементов. Таким образом, таблица задается списком своих строк.

Именно так, в виде списка строк, таблица хранится в компьютерной памяти. Для доступа к элементу $T_{i,j}$ (элемент, стоящий на пересечении i-й строки и j-го столбца) используют формулу:

```
номер в списке = (i – 1)· m + j
```

Эта формула справедлива тогда, когда нумерация строк и столбцов начинается с единицы. Происхождение ее довольно прозрачно: пропускаются (i-1) полных строк, затем отсчитывается j-й элемент.

Рассмотрим в качестве примера таблицу цен на товары:

$T =$

Товар	Цена
монитор $T_{1,1}$	200 $T_{1,2}$
сканер $T_{2,1}$	80 $T_{2,2}$
принтер $T_{3,1}$	300 $T_{3,2}$

Эта таблица переписывается в виде списка следующим образом:

```
T = ((монитор, 200), (сканер, 80),
     (принтер, 300))
```

Реально в памяти таблица хранится в виде линейного списка:

```
T = (монитор, 200, сканер, 80,
     принтер, 300)
```

Проверим, как работает описанная выше формула доступа. Пусть необходимо получить доступ к элементу $T_{3,2}$:

```
номер в списке = (3 – 1)· 2 + 2 = 6
```

Получаем, что элемент $T_{3,2}$ расположен в списке на шестом месте. Так и есть, это число 300.

Сложные структуры

Комбинируя все описанные выше структуры, можно получать информационные блоки какой угодно степени сложности.

Например, вполне можно представить себе таблицу, в каждой клетке которой расположено дерево, узлами которого являются списки, элементы которых есть стеки.

Построение структуры, адекватно и удобно описывающей информацию, — важная задача практической информатики.

Поиск нужного элемента в информационной структуре — важная операция. Поэтому в “Азбуке Роботландии” вводятся индексы в списках и таблицах, и дети с ними активно работают.

Списки в “Азбуке”

Под списком в “Азбуке” понимается линейный список — множество элементов, в котором важен порядок. Конечность множества предполагается, но явно не оговаривается:

Список — это *набор элементов*, записанный в *одну строку* или *один столбец*. В списке важен *порядок следования* элементов.

Умолчание конечности

К “умолчанию” свойства конечности мы пришли после того, как осознали, что детям трудно понять, чем “бесконечное” отличается от “очень много”. Первый вариант определения списка, с которым мы работали с детьми, включал в себя и признак порядка, и признак конечности. Но простые вопросы вроде “будет ли списком множество всех компьютеров на Земле?” оказались совсем не простыми. Для ответа надо понять, можно ли в принципе пересчитать все компьютеры. И дети чаще всего отвечали — нет, нельзя, их количество бесконечно...

В конце концов свойство конечности мы посчитали само собой разумеющимся и не стали фокусировать на нем внимание, зато с важностью порядка следования элементов в списке поработали изрядно, мотивируя ее важностью доступа к элементу, положение которого в информатике задается порядковым номером.

Наборы из одних и тех же элементов, но в **разном порядке** образуют **разные списки**.

Скажем, список:

Вася, Коля, Петя

и список

Коля, Петя, Вася

для информатики разные. Почему?

Дело в том, что информатика занимается обработкой информации, и для нее важно, какая запись на каком месте расположена, ибо доступ к элементу списка выполняется по его номеру.

Посмотрите на такой алгоритм.

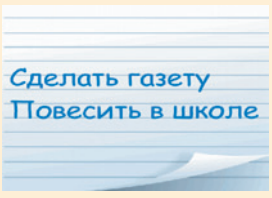
АЛГОРИТМ

1. Сложи первые два элемента в списке.
2. От полученной суммы вычти третий элемент списка.

Список	Результат выполнения алгоритма
1,2,3	Выполняя алгоритм для этого списка, исполнитель получит ответ 0
3,2,1	Выполняя тот же алгоритм для этого списка, исполнитель получит ответ 4

Почему получаются разные ответы? Потому что список 1,2,3 — это не то же самое, что список 3,2,1.

Да и обычному человеку, не информатику, порядок элементов в списке бывает важен. Представьте, например, что список Вася, Коля, Петя есть список призеров конкурса, занявших первое, второе и третье места. Понятно, что список Коля, Петя, Вася будет очень обиден Васе, занявшему 1-е место.



Или вот, посмотрите, в списке этих дел элементы никак нельзя переставить местами, получится чепуха!

Иногда порядок следования элементов в списке человеку бывает безразличен. Список предметов в моей сумке:

книга, тетрадь, карандаш

Но в других случаях порядок в том же списке бывает важен. Список подарков за первое, второе и третье места:

книга, тетрадь, карандаш

Порядок элементов в списке важен или не важен для человека. Информатика работает с элементами списка по их номерам, поэтому для нее порядок элементов важен всегда.

Пустой список

Следующая трудность — понятие пустого списка. Что за странность, называть списком пустое место! Проводится аналогия с нулем в математике.

Пример списка	Описание
Список рук человека: Левая рука Правая рука	Может в списке быть всего два элемента? Да, например, список рук человека
Список дней недели без занятий в школе: Воскресенье	А может в списке быть только один элемент? Да, например, список дней недели без школьных уроков
Список людей в пустом автомобиле: ппппппппп  	А может быть список без элементов? Какой же это список, скажете вы, если в нем ничего не записано? Да, это странно. Но информатика считает это списком и называет его пустым. *Пустой список* — это список, в котором нет элементов

— Зачем нужен пустой список? — спросил Лисенок Хролика.

— Действительно, — поддержали товарища Прям и Трям, — зачем нам пустой список, если в нем ничего нет?

— А зачем нам число ноль? — в свою очередь, спросил ребят Хролик. — Не съешь ведь ноль конфет, не положишь в карман ноль монет, не поиграешь в ноль игр, не пообщаешься, когда у тебя ноль друзей.

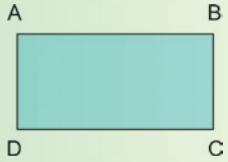
— Ноль удобен в математике, — сказал Лисенок после некоторого размышления. — Надо ведь как-то обозначать число, которое получается в результате вычитания 5–5.

— А пустой список удобен в информатике! Надо ведь как-то обозначать список, из которого удалены все элементы!

Пустой список — это как ноль в математике!

Учимся строить списки по описанию

Новые идеи и понятия становятся привычными в процессе их практического использования. Задания, подобные тем, что приводятся ниже, решаем и обсуждаем в классе на большом экране.

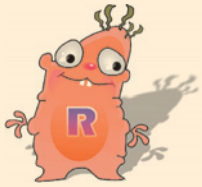
Описание	Можно сделать список?
Набор всех цифр	Да. Его можно записать, например, в порядке возрастания: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 Количество элементов в этом списке равно 10
Набор вершин прямоугольника: 	Да. Его можно записать так: A, B, C, D Количество элементов в списке равно 4
Будет ли списком перечень всех людей на Земле? 	Да, если кто-то потрудится его написать! Количество элементов в этом списке — примерно 7 миллиардов. Один миллиард — это очень большое число, единица с 9 нулями: 1 000 000 000 А это 7 миллиардов: 7 000 000 000 Заметим, что если заносить в список имя одного человека за одну секунду, то на всю работу потребуется примерно 225 лет! Ясно, что один человек вручную с такой работой не справится, но с помощником-компьютером задача выполнима за разумное время
Будет ли списком перечень всех деревьев на Земле? 	Да. Деревьев на Земле очень много, но их можно переписать, поручив эту нелегкую работу, например, компьютеру
Будет ли списком перечень всех коров на этом рисунке? 	Да. Этот перечень можно обозначить так (пустым местом): мммммммммммммм *Это пустой список*. В нем нет ни одного элемента, ведь коров на рисунке нет! Информатика работает с пустыми списками

Вероятно, разговор о списках на этом можно было бы и остановить. Ну, поговорить еще о номерах и маркерах, как об общепринятых способах оформления списка в письменной речи. Но такой подход противоречил бы общей роботландской позиции — без введения формализма и освоения его на практике тема остается абстракцией, малопонятной, малополезной и быстрозабываемой.

Поэтому мы вводим понятие длины списка и индекса элемента в нем. Введение длины еще раз подчеркивает конечность списка, а введение индексов продолжает тему координат, начатую в однострочном редакторе (текстовая строка — это, конечно, список символов) и продолженную в многострочном редакторе (готовы к индексам в таблице).

Давайте послушаем, как Хролик вводит формализмы на списках.

Действия над списками



Это Хролик — веселый, умный и много знающий (на досуге Хролик прочитал всю Википедию). Все жители Роботландии любят поучать. Хролик — не исключение, он делает это с большим удовольствием.

Обозначение списка

Список можно обозначить какой-нибудь буквой или словом и рассматривать эту букву или слово как *имя списка*. Элементы списка будем записывать в круглых скобках.

`X = (5,20,30,15)` Записан список с именем X (латинская буква “икс”). Этот список состоит из четырех чисел: 5, 20, 30 и 15.

Длина списка

Длиной списка называют количество элементов в списке.

```
Y = (5,20,30,15)
Длина(Y) = 4
```

Длина списка Y (латинская буква “игрек”) равна 4 — столько элементов содержит этот список.

Потренируемся вычислять длину списка (интерактивный фрагмент учебника выводится на большой экран в классе).

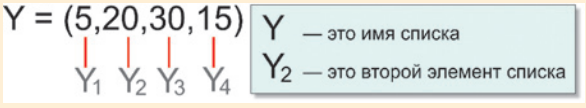
Список	Длина списка
X = (0,2,3,10,5,0)	Длина(X) = 6
X = (кот,лиса,жаба)	Длина(X) = 3
X — список всех цифр	Длина(X) = 10
X — список всех букв русского алфавита	Длина(X) = 33
X = (0)	Длина(X) = 1
X = ()	Длина(X) = 0

Все пустые списки считаются одинаковыми. Длина пустого списка равна нулю.

Индекс элемента списка

Для обозначения элементов списка к его имени приписывают подстрочное число. Это число называют *индексом*.

Индекс обозначает *номер элемента* в списке.



Работаем с примерами на большом экране в классе:

Список	Элемент списка
X = (0,2,3,10,5,0)	X_1 = ? X_4 = ? X_6 = ?
Y = (кот,лиса,жаба)	Y_1 = ? Y_2 = ? Y_3 = ?
X — список всех цифр в порядке возрастания	X_2 = ? X_5 = ? Y_9 = ?

Практика

Практические задания дети выполняют:

- ❑ фронтально на большом экране в классе, с коллективным обсуждением решений;
- ❑ в электронных практикумах учебника (практикумК);
- ❑ за партой, работая с раздаточным материалом (практикумП);
- ❑ при выполнении традиционных физминуток;
- ❑ дома (факультативно).

Как всегда, задания выстраиваются по нарастанию сложности.

Работаем на большом экране в классе

Скрыть ответы

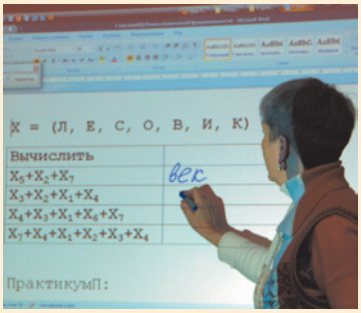
Дано	Вычислить	Результат
X = (3,5,7,1)	Длина(X)-1	...
X = (10,0,7) Y = (заяц,тигр,слон)	Длина(X)-Длина(Y)	...
X = () Y = ()	Длина(X)+Длина(Y)	...
X = (3,0,2,5)	X_1+X_4	...
X = (10,0,7) Y = (6,4,2)	X_1-Y_3	...

X = (10,5,7)	Найти индекс максимального элемента в списке.	...
X = (10,5,7)	Найти индекс минимального элемента в списке.	...
X = (10,12,7,0)	Индекс минимального элемента сложить с индексом максимального.	...
X = (10,10,8,4,4)	От индекса первого минимального элемента отнять индекс последнего максимального.	...

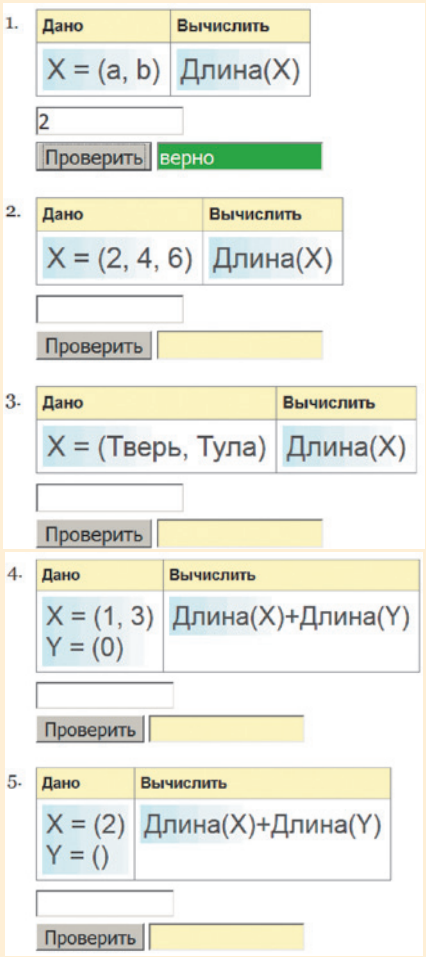
Другой вариант задания. Дан список X: X = (Л, Е, С, О, В, И, К)

Записать результаты вычислений (дети отвечают, учитель записывает ответ):

Вычислить	Результат
$X_5 + X_2 + X_7$	ВЕК
$X_3 + X_2 + X_1 + X_4$	
$X_4 + X_3 + X_1 + X_6 + X_7$	
$X_7 + X_4 + X_1 + X_2 + X_3 + X_4$	



Пример электронного практикума (практикумК)



Физминутка со списками

Дети становятся рядом со своими местами друг за другом. Рассчитываются по порядку. Для разных номеров (индексов) даются разные задания, например, первые (в каждом ряду) прыгают, вторые приседают, третьи кружатся и т.п.

Пример домашнего задания

Пример
Список моих друзей
`X =(Витя, Катя, Коля)`
`Длина(X) = 3`
X_1 = Витя
X_3 = Коля
	Задание 1. Придумайте непустой список, дайте ему имя. Запишите (по образцу примера):
	❑ описание;
	❑ обозначение;
	❑ длину;
	❑ первый элемент;
	❑ последний элемент

Пример
Список цифр, меньших 0
`Y = ()`
`Длина(Y) = 0`
	Задание 2. Придумайте пустой список, дайте ему имя. Запишите (по образцу примера):
	❑ описание;
	❑ обозначение;
	❑ длину

Работаем с карточками за партой (практикумП)

Задание. Дан список X

X = (Л, Е, С, О, В, И, К)

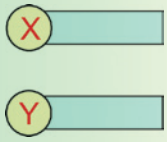
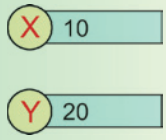
Составить указанные слова из элементов списка X.

Слово	Как получить из списка X
ИЛ	X_6 + X_1
ОКО	
ВОЛК	
СЛОВО	

Алгоритмы на списках

На базе списочного формализма (именование, форма записи, длина, индексы) строится набор алгоритмических заданий со списками. Но сначала Кукарача вводит команду присваивания.

Команда присваивания

	— Привет! Меня зовут *Кукарача*. Сегодня речь пойдет о важной команде, которая присутствует во многих языках программирования и называется командой *присваивания*
	Посмотрите, я изобразил две пустые ячейки памяти. Одна ячейка называется X, другая — Y
	Как записать в ячейку X результат вычисления 8+2, а в ячейку Y — число 20?

Программисты могут написать две таких команды:

АЛГОРИТМ

```
1. X ← 8 + 2
2. Y ← 20
```

Читается это так:

1. X *присвоить* значение, равное сумме 8 + 2.
2. Y *присвоить* значение, равное 20.

Операция, обозначенная стрелочкой, называется **операцией присваивания**. А команда с операцией присваивания называется **командой присваивания**.

Команда присваивания работает так:
1. Выполняются вычисления (если они есть) *справа* от стрелочки.
2. Результат записывается в ячейку, имя которой указано *слева* от стрелочки.

Команды, записанные выше, выполнятся так.

`X ← 8 + 2`	1. Сначала будет найдена сумма 8 + 2. 2. Затем результат 10 будет записан в ячейку X
`Y ← 20`	1. Сначала будет найдено значение правой части — число 20 (вычислять ничего не пришлось). 2. Затем число 20 будет записано в ячейку Y

Команда присваивания имеет вид, подобный изображенному на рисунке:

Операция присваивания разделяет команду на две части. Слева от нее записывается имя ячейки, справа — выражение, которое нужно вычислить. Работает команда присваивания по следующему алгоритму:

АЛГОРИТМ

1. Выполняются вычисления *справа* от знака присваивания.
2. Результат записывается в ячейку, имя которой указано *слева* от знака присваивания.

Практика

Ниже приводятся примеры алгоритмических заданий, предлагаемых ученикам в разных формах (работа на большом экране в классе, электронные практикумы, письменные работы, подвижные игры, домашние задания).

Команда присваивания. Какое значение будет записано в ячейку Z после выполнения алгоритма (знак "|" удаляет один символ, примыкающий к нему слева)?

1	1. Z ← 15–6	**2**	1. X ← 1 2. Z ← 9+X	**3**	1. Z ← 1 2. Z ← Z+1	**4**	1. Z ← ре 2. Z ← Z+ка
5	1. Z ← ле 2. Z ← по+Z	**6**	1. Z ← ель|| 2. Z ← Z+нота|	**7**	1. Z ← лес|| 2. Z ← стон|+Z	**8**	1. Z ← конь| 2. Z ← Z+вертолет 3. Z ← Z||||

Алгоритмы со списками. Какое значение будет записано в ячейку Z после выполнения алгоритма?

1	1. X ← (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) 2. Y ← X_8+Длина(X) 3. Z ← Y–7	**2**	X ← (г, а, р, ш) Z ← X_4+X_2+X_3 Z ← Z|+X_1
3	1. X ← (кот, лиса, тигр) 2. Z ← X_1||+|X_2|| 3. Z ← Z+X_3|||	**4**	1. X ← (лес, поле, рак) 2. Z ← |X_2||+||X_1+|X_3| 3. Z ← X_3||+Z

Использование списков

Списки используют для упорядочивания информации, для придания ей вида, удобного для восприятия. Списки часто записывают в столбик, а элементы либо *нумеруют*, либо снабжают *маркерами*. Первый способ записи называют *нумерованным* списком, второй — *маркированным*.

Нумерованные списки

На с. 37 показаны три *нумерованных* списка. В нумерованных списках номера (индексы) проставляются явно в виде числа с точкой перед каждым элементом.

В виде нумерованного списка оформляют и алгоритмы, ибо порядок следования шагов в алгоритме очень важен.

Список времен года:	Список месяцев года:
1. Зима 2. Весна 3. Лето 4. Осень	1. Январь 2. Февраль 3. Март 4. Апрель 5. Май 6. Июнь 7. Июль 8. Август 9. Сентябрь 10. Октябрь 11. Ноябрь 12. Декабрь
Список цветов радуги: 1. Красный 2. Оранжевый 3. Желтый 4. Зеленый 5. Голубой 6. Синий 7. Фиолетовый	

Маркированные списки

В информатике порядок следования элементов в списке важен всегда. Но в жизни встречаются ситуации, когда порядок элементов в наборе несущественен. В таких случаях вместо номеров перед элементами используют пометки в виде точек, квадратиков, кружков или других фигур. Эти пометки называют *маркерами*, а такую запись — *маркированным списком*. Ниже показаны два маркированных списка покупок:

Список 1	Список 2
❑ Хлеб ❑ Молоко ❑ Соль	❑ Молоко ❑ Хлеб ❑ Соль

С точки зрения информатики это — разные списки. Но идти в магазин можно с любым из них!

Таблицы в “Азбуке”

Таблица вводится как список списков на примере расписания занятий Белочки (она прилежно посещает Лесную школу).

Определение таблицы

Каждый день, кроме выходных, в Лесной школе 3 урока. На каждый день у Белочки заготовлено расписание:

Понедельник	танцы	пение	информатика
Вторник	информатика	рисование	пение
Среда	рисование	информатика	танцы
Четверг	информатика	танцы	пение
Пятница	рисование	информатика	пение

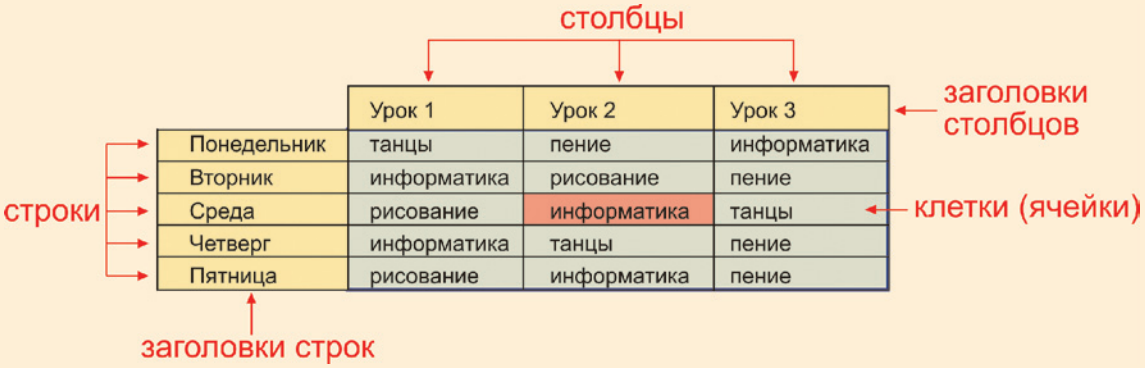
Увидел эти листочки заяц Костя и говорит:

— У тебя 5 списков, длина каждого равна 3. Из этих списков получается замечательная таблица:

	Урок 1	Урок 2	Урок 3
Понедельник	танцы	пение	информатика
Вторник	информатика	рисование	пение
Среда	рисование	информатика	танцы
Четверг	информатика	танцы	пение
Пятница	рисование	информатика	пение

Таблица состоит из *клеток* (их еще называют *ячейками*). Клетки таблицы у нас голубые. По вертикали клетки образуют *столбцы*, по горизонтали — *строки*. Для удобства над столбцами и перед строками по-

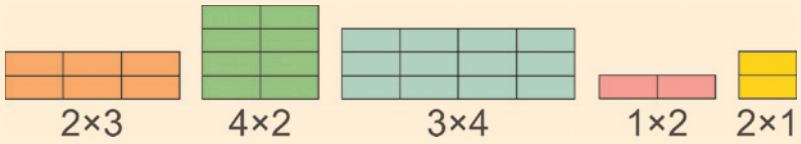
мещают дополнительные клетки (они у нас желтые), в них записывают *заголовки столбцов* (сверху) и *заголовки строк* (слева).



	Урок 1	Урок 2	Урок 3
Понедельник	танцы	пение	информатика
Вторник	информатика	рисование	пение
Среда	рисование	информатика	танцы
Четверг	информатика	танцы	пение
Пятница	рисование	информатика	пение

Как узнать, какой предмет будет вторым уроком в среду? Нужно прочитать название предмета в клетке на пересечении строки “Среда” и столбца “Урок 2”. Это будет — информатика.

Количество элементов в списке, как вы помните, называется *длиной* списка. *Количество строк и столбцов* в таблице называют *размером* таблицы. Размер таблицы обозначают в виде двух соответствующих чисел через знак умножения:



Если число строк в таблице равно числу столбцов, таблицу называют *квадратной*:

Табличные индексы. Утиное правило

Таблицу, как и список, можно обозначать какой-нибудь буквой (или словом), а для обозначения элементов (содержимого клеток) использовать *индексы*. Индексов нужно два. Один будет определять *номер строки*, а второй — номер элемента в строке, то есть *номер столбца*.

Обозначим таблицу с расписанием уроков буквой `P`.

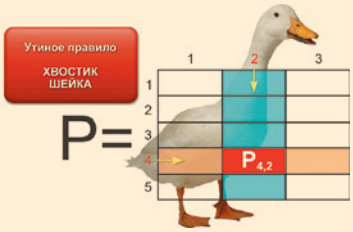
	1	2	3
1	танцы	пение	информатика
2	информатика	рисование	пение
3	рисование	информатика	танцы
4	информатика	танцы	пение
5	рисование	информатика	пение

Тогда $P_{1,2}$ = пение, $P_{2,1}$ = информатика, $P_{5,1}$ = рисование, $P_{4,2}$ = танцы.

Элементы (или клетки) таблицы можно указывать при помощи двух индексов. Первый индекс определяет номер строки, второй — номер столбца.

Порядок следования индексов роботландцы называют “утиным правилом”:

Первый индекс — *строка* от утиного *хвостика*, второй индекс — *столбец* от утиной *шейки* (правило “хвостик, шейка”).



Что можно узнать по таблице

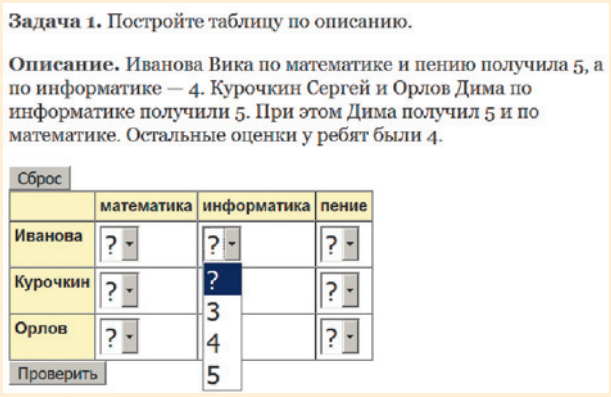
По таблице можно узнать много интересного! Давайте проверим это на примере расписания Белочки.

	Урок 1	Урок 2	Урок 3
Понедельник	танцы	пение	информатика
Вторник	информатика	рисование	пение
Среда	рисование	информатика	танцы
Четверг	информатика	танцы	пение
Пятница	рисование	информатика	пение

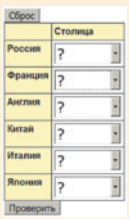
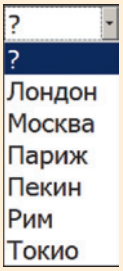
Вопрос	Ответ
Каков размер таблицы?	5×3
Сколько клеток в таблице?	15
Какой предмет будет вторым уроком во вторник?	*рисование*
Какой предмет будет последним уроком в пятницу?	*пение*
В какие дни недели нет рисования?	*в понедельник и четверг* (смотрим по строкам)
Сколько дней недели без информатики?	0 (смотрим по строкам)
В какие дни недели вторым уроком будут танцы?	*только в четверг* (смотрим второй столбец)
Сколько раз третьим уроком поставлено пение?	3 (смотрим третий столбец)
Сколько раз в неделю поставлены танцы?	3 (смотрим все клетки, по строкам или столбцам)

Строитель таблиц

Строитель таблиц — это электронная среда для конструирования таблиц. Содержимое каждой ячейки выбирается из выпадающего списка. Награда за верное решение — одобрение роботландского Профессора.

Начало построения таблицы	Проверка решения
	

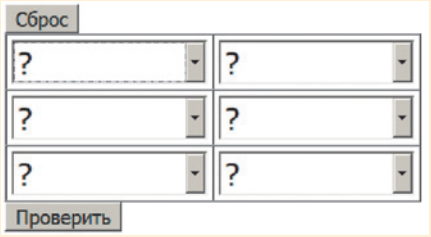
Примеры заданий

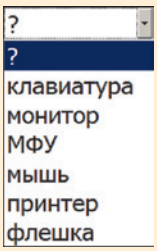
Строитель таблиц	Выпадающий список
Постройте таблицу в соответствии с ее заголовками. 	

Постройте таблицу У с такими ячейками:

$У_{2,1}$ = монитор $У_{1,2}$ = клавиатура $У_{2,2}$ = принтер

$У_{3,2}$ = флешка $У_{1,1}$ = мышь $У_{3,1}$ = МФУ



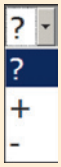


По какому правилу записаны названия устройств в эту таблицу?

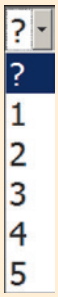
(*Ответ*. В первой строке записаны устройства ввода, во второй — устройства вывода, а в третьей — устройства ввода/вывода.)

Постройте таблицу по описанию.

Описание. Лиса боится только волка. Волк никого не боится, а заяц боится всех.



Постройте таблицу, в которой содержимое каждой ячейки на 1 меньше суммы индексов этой ячейки.



Постройте таблицу по следующим правилам:

Если номер строки…

- …*равен* номеру столбца, в ячейке 0;
- …*больше* номера столбца, в ячейке 1;
- …*меньше* номера столбца, в ячейке 2.

Читатель может поработать со *Строителем таблиц*, *Строителем иерархий* и примерами других практикумов учебника на электронных страницах диска, сопровождающего этот номер.

Использование таблиц

Ученики работают с примерами популярных таблиц.

Таблица сложения

В каждой клетке этой таблицы находится результат сложения двух чисел: номер строки складывается с номером столбца.

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

Результат расположен на пересечении строки и столбца.

Расположите курсор над клеткой таблицы.

4+2=6

Интерактивная таблица сложения демонстрируется на большом экране в классе, ученики работают с ней под управлением учителя.

Таблица умножения

В каждой клетке этой таблицы находится результат умножения двух чисел: номер строки умножается на номер столбца.

	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

Результат расположен на пересечении строки и столбца.

Расположите курсор над клеткой таблицы.

4×2=8

Интерактивная таблица умножения демонстрируется на большом экране в классе, ученики работают с ней под управлением учителя.

Свойства веществ

Роботландский (естественно, ученый) кот Егор увлечен химией. Он исследует свойства разных веществ, а результаты своих опытов и наблюдений заносит в таблицы. Вот пример работы Егора — таблица сравнения свойств льда и снега:

	Цвет	Прозрачность	Действие тепла	Другие свойства
Снег	белый	непрозрачный	тает, превращается в воду	рыхлый
Лёд	бесцветный	прозрачный	тает, превращается в воду	хрупкий

Скрыть ответы

Вопрос	Ответ	Вопрос	Ответ
Каков цвет снега?	... ?	Как действует тепло на снег?	... ?
Лёд прозрачный?	... ?	Снег прозрачный?	... ?

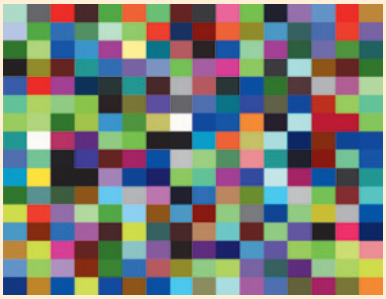
Ученики анализируют таблицу (число строк, столбцов, размерность), отвечают на интерактивные вопросы учебника.

Таблица пикселей

Посмотрите внимательно на экран монитора.

Картинка кажется нам гладкой, но на самом деле она состоит из отдельных квадратиков! Под лупой с большим увеличением мы это ясно увидим:

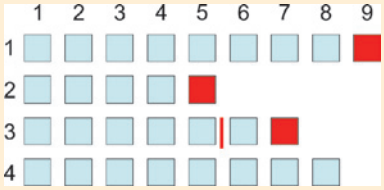
Эти квадратики называются *пикселями*. На экране монитора они образуют таблицу. Пиксель — клетка этой таблицы. Компьютер задает цвет каждого пикселя в соответствии с программой рисования.



Размер пиксельной таблицы называют *разрешением экрана*. Размер пиксельной таблицы в 768 строк и 1024 столбцов не считается слишком большим. Правда, размер пиксельной таблицы принято указывать не по формуле `строки×столбцы`, а по формуле: `столбцы×строки`.

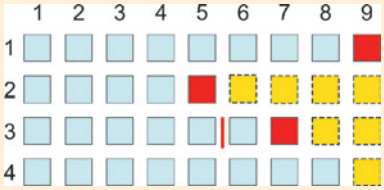
Экран многострочного редактора

В многострочном редакторе текст записывается по строкам, а символы располагаются по столбцам.



Можно ли назвать экран редактора таблицей? Нет, ведь строки в редакторе разной длины.

Но если дополнить строки условными пустыми позициями, то получится настоящая таблица!



Обозначим эту таблицу через P и ответим на интерактивные вопросы учебника:

Скрыть ответы

Вопрос	Ответ	Вопрос	Ответ
Размер таблицы на рисунке?	...	После какого символа стоит курсор?	...
Перед каким символом стоит курсор?	...	Какой символ находится в клетке $P_{3,7}$?	...
Перед каким символом окажется курсор после двух команд Вверх?	...	Перед каким символом окажется курсор после двух команд Вправо?	...

Электронные таблицы

Электронная таблица — это таблица, в которой значение части ячеек рассчитывается автоматически. Ниже показана электронная таблица для заказа детских игрушек.

Сброс

	Цена	Штук	Стоимость
Машинка	50	2	100
Паровозик	200	1	200
Ведёрко	100	3	300
Итого			600

Если изменить данные в столбце *Штук*, столбец *Стоимость* пересчитается автоматически после нажатия на клавишу Enter в строке ввода или при щелчке мыши вне поля ввода.

Ученики работают с таблицей и отвечают на интерактивные вопросы учебника:

Скрыть ответы

Что купили	Сколько заплатили	Что купили	Сколько заплатили
Машинку, паровозик и 2 ведёрка.	...	5 машинок, 2 паровозика и 1 ведёрко.	...

Работа с таблицей покупок продолжается в электронном практикуме учебника.

Задание

Найдите стоимость покупки. Используйте таблицу. Ответ копируйте через буфер обмена.

Примеры покупок

- ❑ 120 машинок, 300 паровозиков и 700 ведерок.
- ❑ По 15 штук каждого товара.
- ❑ 30 машинок и 213 ведерок.
- ❑ 457 паровозиков.
- ❑ Ничего не куплено.

Алгоритмы на таблицах

Продолжаем работу с таблицами, выполняя алгоритмические задания. Примеры приводятся ниже.

Задана таблица:

	1	2	3	4	5
1	2	0	7	1	9
2	1	8	5	6	1
3	4	3	6	7	8
4	1	0	0	1	7

Выполнить алгоритмы и сообщить содержимое ячеек памяти.

1	1. $X \leftarrow T_{3,2}+1$		**4**	1. $X \leftarrow T_{1,5}$ 2. $X \leftarrow X+1$
2	1. $X \leftarrow T_{3,2}+T_{2,3}$		**5**	1. $X \leftarrow 2$ 2. $X \leftarrow T_{2,2}+X$ 3. $X \leftarrow X+X$
3	1. $X \leftarrow T_{1,5}+1$ 2. $Y \leftarrow X-T_{4,5}$			

Бескомпьютерная практика

Школьные уроки в “Азбуке” строятся по следующему “типовому” сценарию:

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

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

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

Ниже приводится пример “табличной” игры-физминутки.

Игра “Заполним таблицу”

Таблица моделируется с помощью парт и стульев, за которыми сидят ученики. Оговаривается порядок строк и столбцов. Ученики получают табличные индексы и занимают соответствующие места в классе.

Иерархии в “Азбуке”

В Азбуке иерархия вводится как

❑ порядок подчинения одних объектов другим (солдаты подчиняются командиру);

❑ или порядок вхождения одних объектов в другие (процессор входит в состав компьютера).

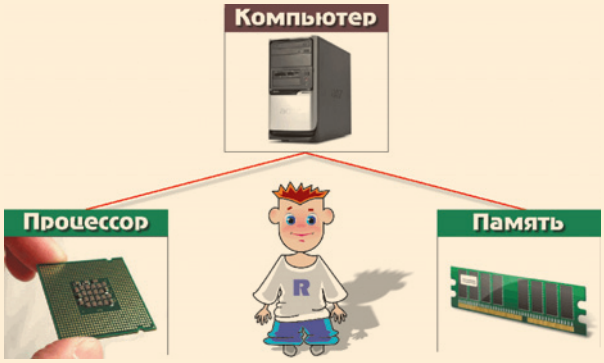
В этом определении описываются *правила*, по которым создаются иерархические структуры. Более точно про иерархию как информационную структуру надо было бы сказать так: “Иерархия — это такая организация информационных элементов, в которой отражаются отношения подчинения или отношения вхождения”. Хотя и последнее определение дает только качественное описание. Строгое определение по Д.Кнуту приводится выше в разделе “Теоретическое введение”.

С другой стороны, иерархией называют не только информационную структуру, но и сам способ ее организации. Мы часто говорим об иерархии, как об отношениях. Поэтому мы упростили определение до сути иерархических отношений, ибо это самое главное, что должно связаться в голове учеников с этим сложным словом.

Определение подкрепляется серией примеров, в которых иерархия изображается в виде перевернутого дерева.

Отношения подчинения:

Отношения вхождения:

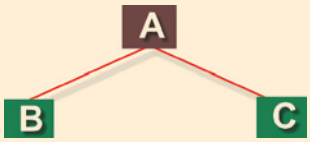


Пример со вхождением из географии:

Читается схема так. Часть света *Америка* состоит из двух материков: *Северная Америка* и *Южная Америка*. В *Северную Америку* входят страны: *Канада*, *США* и *Мексика*. В *США* есть штаты: *Нью-Йорк*, *Колорадо*, *Техас*. В *Южной Америке* расположены страны: *Бразилия*, *Аргентина*, *Перу*.

Видим, что иерархию рисуют из двух элементов: *блоков* и *линий*, соединяющих блоки. Блоки называют *вершинами* (или *узлами*), а соединительные линии — *ветвями* (или *связями*). Вершина содержит *элемент*, а ветви показывают *подчинение* или *вхождение* элементов. Вершину, с которой дерево начинает свой “рост”, называют *корнем*.

Принято говорить о *родственных связях* вершин.



Так, если от вершины A тянутся ветви к вершинам B и C, расположенным ниже, то B и C называют *потомками* вершины A (или *дочерними* вершинами), а между собой вершины B и C — *сестры*. Соответственно, вершину A называют *родителем* по отношению к вершинам B и C. Вершину, которая не имеет потомков, называют *листом*.

Иерархию рисуют по *уровням* сверху вниз. На первом уровне располагается корень. На втором — прямые потомки корня. На третьем — прямые потомки вершин второго уровня и так далее.

Иерархия в виде лесенки

Иерархию часто изображают в виде записи “лесенкой”. Ступеньки (отступы вправо) отражают подчинение или вхождение элементов.

На первой ступеньке (первом уровне) записан корень иерархии:

Обед

На второй ступеньке (втором уровне) записаны прямые потомки корня:

Первое
Второе
Третье

На третьей ступеньке (третьем уровне) записаны прямые потомки вершин второго уровня:

Борщ Уха
Рыба Мясо
Чай Кофе

И так далее.

Запишем в виде лесенки географическую иерархию. Сначала запишем корень (на первом уровне) и его прямые потомки (на втором уровне):

```
Америка
  Северная Америка
  Южная Америка
```

Затем на третьем уровне записываем прямые потомки вершин второго уровня:

```
Америка
  Северная Америка
    Канада
    США
    Мексика
  Южная Америка
    Бразилия
    Аргентина
    Перу
```

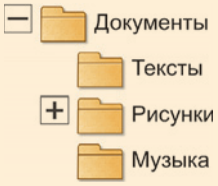
Записываем, наконец, прямые потомки вершин третьего уровня (точки проставлены для наглядности отступов):

```
Америка
  ..Северная Америка
  ....Канада
  ....США
  ......Нью-Йорк
  ......Колорадо
  ......Техас
  ....Мексика
  ..Южная Америка
  ....Бразилия
  ....Аргентина
  ....Перу
```

Использование иерархий

Приводятся примеры иерархий.

Иерархия компьютерных папок



Подробно рассматривается иерархия компьютерных папок, представленная в виде меню выбора в программах-проводниках, обсуждаются интерфейсные вопросы и выполняются практикумы в учебной среде электронного учебника и в “полевых” условиях операционной системы.

Заводские отношения

Иерархию используют тогда, когда нужно описать что-то сложное, большое. Иерархия разделяет сложное и большое на простое и маленькое, используя принцип *подчинения* или принцип *вхождения*.

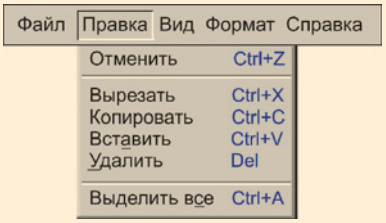
Рассмотрим небольшой завод, состоящий из трех цехов. Опишем подробнее только цех 2, в котором имеются три станка.

Структура завода задается иерархией, построенной по принципу *вхождения*:

Структура производственных отношений людей на заводе описывается иерархией, построенной по принципу *подчинения*:

Сложное меню

В прошлом году сложным мы называли меню, некоторые пункты которого сами являются меню. Видим, что сложные меню образуют иерархию. В качестве примера рассматривается меню программы Блокнот.



Все пункты этого меню содержат подменю. Например, пункт `Правка` содержит подменю команд правки текста.

Заметим, что в полоске меню присутствуют прямые потомки корня `Меню`, а сам корень не изображен.

Эта иерархия переписывается лесенкой так:

```
Меню программы Блокнот
  Файл
  Правка
    Отменить
    Вырезать
    Копировать
    Вставить
    Удалить
    Выделить все
  Вид
  Формат
  Справка
```

Классификация

Мы говорим, что *классификация* — это разделение на части.

Не механическое разделение (“распиливание бревна”), а построение иерархической схемы, в которой исходный объект — “вешалка” объявляется

корнем, а его значения (возможно, тоже “вешалки”) становятся потомками.

Например, все автомобили сначала можно разделить на три части:

```
Автомобили
  Грузовые
  Пассажирские
  Специальные
```

Затем разделить на две части автомобили пассажирские:

```
Автомобили
  Грузовые
  Пассажирские
    Автобусы
    Легковые
  Специальные
```

Затем разделить легковые автомобили на 8 частей:

```
  Грузовые
  Пассажирские
    Автобусы
    Легковые
      Седаны
      Универсалы
      Хэтчбеки
      Лимузины
      Пикапы
      Минивэны
      Купе
      Внедорожники
  Специальные
```

И так далее.

Практика

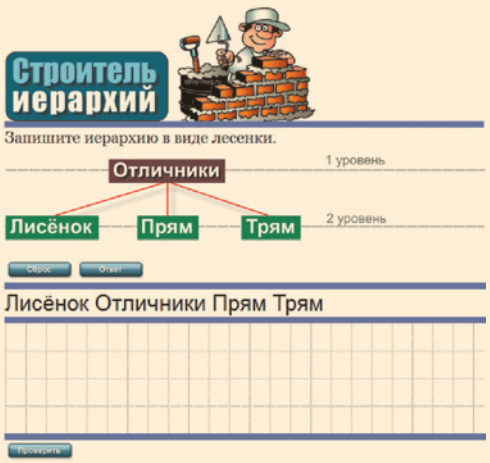
Исполнитель Строитель иерархий

Из Роботландии в помощь ребятам вызван *Строитель иерархий*. Этот джентльмен помогает собирать иерархии в виде лесенки.

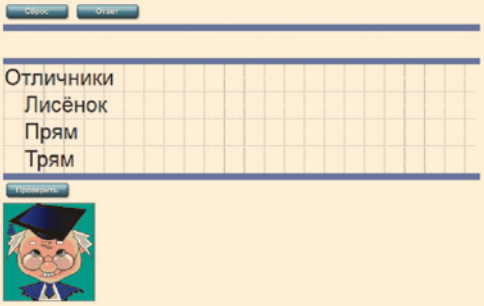
Как с ним работать? Очень просто: перетаскивайте себе слова мышкой на нужные ступеньки, и все! *И не забывайте, что порядок следования вершин важен.*

Роботландские отличники

Лисенок, Прям и Трям сдали все экзамены на отлично! Хролик нарисовал по этому поводу дерево иерархии. Перепишите это дерево в виде лесенки.



Перетаскиваем слова-вершины на нужные места в рабочем поле и получаем лесенку, которую одобряет сам роботландский Профессор:



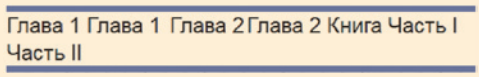
На базе *Строителя иерархий* построено много компьютерных практикумов. К нашей радости и удивлению, с большинством таких заданий дети справлялись легко, быстро и с интересом.

Примеры заданий в среде *Строитель иерархий*

Содержание книги

Книга состоит из двух частей. Каждая часть содержит две главы. Составьте иерархию по этому описанию.

Перетаскиваемые вершины:



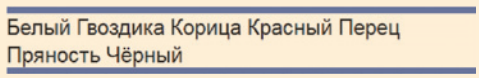
Чаепитие

Перепишите иерархию в виде лесенки.

Пряности

Составьте иерархию по описанию и запишите ее в виде лесенки. *Описание:* к пряностям относятся: гвоздика, корица, перец. Перец бывает черный, белый и красный.

Перетаскиваемые вершины:



Пример задания практикумаП

Собираем иерархию “Символ” из карточек.

Это же задание дети выполняли в прошлом году, тогда иерархические построения мы называли “вешалками”.

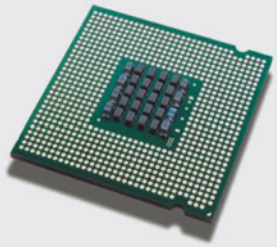
(В Роботландии “*вешалкой*” называют слово, которым можно заменять другие слова. Например, слово *дерево* — “вешалка”! Им можно заменить, или, как любят говорить ребята, на него можно “повесить”, ель, сосну, березу, тополь, дуб, ветлу — любое *конкретное* дерево.)

МОДЕЛИРОВАНИЕ

Моделирование арифметических действий в процессоре

Д.М. Златопольский,
Москва

Как известно, в процессоре компьютера данные хранятся и обрабатываются в двоичном виде. В настоящей статье мы смоделируем выполнение в процессоре четырех арифметических действий над неотрицательными целыми числами средствами языка программирования высокого уровня — школьного алгоритмического языка.



1. Сложение

Прежде всего напомним, что происходит при сложении двух однозначных двоичных чисел (синим цветом и курсивом оформлено значение возможного “переноса” из соседнего справа разряда):

$$\begin{array}{rr} & \textit{0} \\ + & 0 \\ \hline & 0 \end{array} \qquad \begin{array}{rr} & \textit{0} \\ + & 1 \\ \hline & 1 \end{array} \qquad \begin{array}{rr} & \textit{0} \\ + & 1 \\ \hline & 1 \end{array} \qquad \begin{array}{rr} & \textit{0} \\ + & 1 \\ \hline 1 & 0 \end{array}$$

$$\begin{array}{rr} & \textit{1} \\ + & 0 \\ \hline & 1 \end{array} \qquad \begin{array}{rr} & \textit{1} \\ + & 0 \\ \hline 1 & 0 \end{array} \qquad \begin{array}{rr} & \textit{1} \\ + & 1 \\ \hline 1 & 0 \end{array} \qquad \begin{array}{rr} & \textit{1} \\ + & 1 \\ \hline 1 & 1 \end{array}$$

Эту информацию можно записать в виде таблицы:

Таблица 1

Цифра 1-го слагаемого	0	0	1	1	0	0	1	1
Цифра 2-го слагаемого	0	1	0	1	0	1	0	1
Перенос из предыдущего разряда	0	0	0	0	1	1	1	1
Цифра результата	0	1	1	0	1	0	0	1
Перенос в следующий разряд	0	0	0	1	0	1	1	1

Пример:

$$\begin{array}{rrrrrrrrr} & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ + & & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{array}$$

Учитывая это, можем так оформить общую схему решения задачи:

```
цикл для всех разрядов
  1. Выделение очередной двоичной цифры в каждом слагаемом
  2. Выделение очередного результата сложения
  3. Определение значения, которое переходит в соседний слева разряд
  4. Учет очередной цифры в рассчитываемой сумме
конец цикла
```

Обсудим методику выполнения перечисленных этапов.

1. Выделение очередной двоичной цифры в каждом слагаемом

Как и при сложении в “столбик” на бумаге, начинать расчеты нужно с последних цифр слагаемых. Последнюю двоичную цифру (пусть ее имя — *цифра*) десятичного числа *n* можно определить так:

```
цифра := mod(n, 2)
```

— где `mod` — функция, возвращающая остаток от деления ее первого аргумента на второй (в других языках программирования для определения остатка используется не функция, а специальная операция).

Однако поскольку мы поставили задачу моделирования действий в процессоре, то для выде-

ления последней двоичной цифры целесообразно применить операцию, проводимую в процессоре. Такой операцией является логическая команда **AND** (русский вариант — **И**). Напомним [1–2], что логические операции над числами являются *поразрядными*. Например, при сложении двух двоичных цифр возможен перенос в старший разряд (см. выше), а при логических операциях все разряды рассматриваются изолированно друг от друга (без переноса):

X	*Y*	*X* **И** *Y*
0	0	0
0	1	0
1	0	0
1	1	1

Например, при $X = 101011$ и $Y = 1101$ имеем X **AND** $Y = 1001$.

Анализ показывает, что младший разряд числа *X* можно выделить (получить) следующим образом:

X **И** 1

(Проверьте!)

В языке Паскаль существует операция с соответствующим именем — and. Ее можно применять к данным целого типа, и результат, который она возвращает, также является целым числом.

2. Определение очередной двоичной цифры результата сложения

Этот этап можно выполнить, учтя данные табл. 1 и рассмотрев 8 возможных вариантов:

```
выбор
  при цифра1 = 0 и цифра2 = 0
      и перенос = 0: цифра_рез := 0
  при цифра1 = 0 и цифра2 = 1
      и перенос = 0: цифра_рез := 1
  при цифра1 = 1 и цифра2 = 0
      и перенос = 0: цифра_рез := 1
  при цифра1 = 1 и цифра2 = 1
      и перенос = 0: цифра_рез := 0
  при цифра1 = 0 и цифра2 = 0
      и перенос = 1: цифра_рез := 1
  при цифра1 = 0 и цифра2 = 1
      и перенос = 1: цифра_рез := 0
  при цифра1 = 1 и цифра2 = 0
      и перенос = 1: цифра_рез := 0
  при цифра1 = 1 и цифра2 = 1
      и перенос = 1: цифра_рез := 1
все
```

— где *цифра*1, *цифра*2 и *цифра_рез* — соответственно цифры первого и второго слагаемых и цифра результата.

Однако такая конструкция (оператор варианта) является громоздкой[1] и соответствует поставленной задаче моделирования.

Цифру результата сложения двух двоичных цифр можно выделить, используя все ту же логическую команду **И** (**AND**). Если результат сложения двух двоичных цифр (с учетом возможного переноса единицы из разряда справа) представить в виде двузначного двоичного числа *P*:

```
P := цифра1 + цифра2 + перенос
```

— где *перенос* — значение, перешедшее "в уме" из разряда справа, то его правую (нужную нам) цифру можно получить уже знакомым нам методом:

```
цифра_рез := P и 1
```[2]

3. Определение значения, которое переходит в соседний слева разряд

Здесь также использование оператора варианта является нецелесообразным.

Искомое значение можно получить, применив еще одну, так называемую "сдвиговую", команду, выполняемую в процессоре, — **SHR** (от ***Sh**ift **R**ight* — сдвинуть вправо) [1].

В результате ее выполнения в регистрах процессора происходит поразрядный сдвиг двоичных цифр на заданное количество разрядов вправо. При этом соответствующее число цифр в младших разрядах теряется, а освободившиеся старшие разряды заполняются нулями. Например, если в 8 разрядах было представлено двоичное число:

1	1	0	0	1	0	0	1

то после сдвига вправо на два разряда оно примет вид:

0	0	1	1	0	0	1	0

В языке Паскаль также имеется возможность выполнить эту команду в программе. Операция shr имеет формат:

```
<a> shr <b>
```

— где <a> и <b> — натуральные числа.

Результатом ее выполнения:

```
x := a shr b
```

будет десятичное число *x*, которое соответствует результату поразрядного сдвига вправо на *b* разрядов двоичного представления операнда *a*.

Учитывая, что искомое значение — это первая цифра числа *P* (см. чуть выше), то ее можно получить, применив сдвиговую операцию вправо с операндом 1:

```
перенос := P СдвигВправоНа 1
```

Внимание! Приведен оператор на школьном алгоритмическом языке, для которого условно принято, что в нем имеется операция СдвигВправоНа — аналог сдвиговой операции shr.

4. Учет очередной цифры в рассчитываемой сумме

Прежде чем приводить методику выполнения данного этапа, скажем, что, кроме команды **SHR**,

[1] В языке программирования Паскаль для ее применения необходимо предварительно рассмотреть и пронумеровать 8 возможных вариантов с помощью условных операторов.

[2] В школьном алгоритмическом языке учитывается регистр букв (служебное слово **и** и операция **И** будут различаться).

имеется также сдвиговая команда **SHL** (от ***Shift Left*** — сдвинуть влево). В результате ее выполнения в регистрах процессора происходит поразрядный сдвиг двоичных цифр на заданное количество разрядов влево. При этом соответствующее число цифр в старших разрядах теряется, а освободившиеся младшие разряды заполняются нулями. Например, если в 8 разрядах находилось двоичное число:

1	1	0	0	1	0	0	1

то после сдвига влево на два разряда оно примет вид:

0	0	1	0	0	1	0	0

Результатом ее выполнения в программе на языке Паскаль:

```
x := a shl b
```

будет целое десятичное число, которое соответствует результату поразрядного сдвига влево на *b* разрядов двоичного представления операнда *a*.

Теперь о нашей задаче. Можем рассуждать следующим образом. Если, например, до вычислений в пятом справа разряде уже ранее был получен результат:

			1	0	0	1

а в пятом разряде цифра результата равна 1, то новое значение искомого результата будет равно:

		1	1	0	0	1

А как добавить очередную цифру в начало “старого” результата? Можно сказать, что новый результат получен так:

+	…	1	0	0	0	0
	…		1	0	0	1

А это означает, что в данном случае результат может быть получен следующим образом:

```
сумма := цифра_рез СдвигВлевоНа 4 + сумма
```

— где `СдвигВлевоНа` — аналог сдвиговой операции `shl`.

В общем случае:

```
сумма := цифра_рез СдвигВлевоНа (номер_разряда - 1) + сумма
```

Далее, так как описанные 4 этапа необходимо выполнить для каждого разряда, то для этого следует отбросить последнюю цифру обрабатываемого числа. Как это сделать, вы, конечно, уже поняли — применить сдвиговую операцию вправо с операндом 1.

Итак, теперь мы можем оформить всю программу решения задачи сложения двух натуральных чисел *a* и *b*. Как принято в нашем журнале, сделаем это на школьном алгоритмическом языке. К уже сделанным замечаниям в части использования команд `СдвигВправоНа` и `СдвигВлевоНа` примем, что имеется также операция `И` — аналог поразрядной логической операции and. Кроме того, будем считать, что максимальное количество двоичных разрядов для представления чисел равно 16.

```
алг Сложение
нач цел a, b, сумма, P, цифра1, цифра2,
        цифра_рез, перенос, i
 ввод a
 ввод b
 i := 1      |Обрабатываемый разряд
 сумма := 0 |Искомое значение
 перенос := 0
 нц пока i <= 16
   цифра1 := a и 1 |Выделяем последние
   цифра2 := b и 1 |цифры слагаемых
   |Определяем результат сложения
   |в i-м разряде
   P := a + b + перенос
   |Выделяем очередную цифру суммы
   цифра_рез := P и 1
   |Учитываем ее в сумме
   сумма:= цифра_рез
           СдвигВлевоНа(i - 1) + сумма
   |Определяем значение перенос
   перенос := P СдвигВправоНа 1
   |Отбрасываем последнюю цифру
   |в каждом слагаемом
   a := a СдвигВправоНа 1
   b := b СдвигВправоНа 1
   |Новый номер обрабатываемого разряда
   i := i + 1
 кц
|Выводим ответ
вывод нс, сумма
кон
```

2. Вычитание

Здесь также рассмотрим возможные варианты вычитания двух однозначных двоичных чисел (синим цветом и курсивом оформлено значение заимствования после вычитания в соседнем справа разряде, зеленым цветом — значение заимствования в результате данного вычитания):

0	*0*	*0*	*0*
− 0	− 0	− 1	− 1
0	1	0	1
0	1 1	1	0
1	*1*	*1*	*1*
− 0	− 0	− 1	− 1
0	1	0	1
1 1	1 0	0	1 1

Эту информацию можно записать в виде табл. 2:

Таблица 2

Номер варианта	1	2	3	4	5	6	7	8
Заимствование после вычитания в соседнем справа разряде	0	0	0	0	1	1	1	1
Цифра уменьшаемого	0	0	1	1	0	0	1	1
Цифра вычитаемого	0	1	0	1	0	1	0	1
Цифра результата	0	1	1	0	1	0	0	1
Заимствование после данного вычитания	0	1	0	0	1	1	0	1

Пример:

```
    1  1  1  0  0  1  0  1
−      1  1  1  0  0  1  0
-------------------------
 0  1  1  1  0  0  1  1
```

Общая схема решения задачи вычитания при известных значениях уменьшаемого и вычитаемого:

```
цикл для всех разрядов
1. Выделение очередной цифры уменьшаемого (цифра1)
2. Выделение очередной цифры вычитаемого (цифра2)
3. Определение результата вычитания в данном разряде (цифра_рез)
4. Учет его в искомой общей разности
5. Определение величины заимствования в соседнем слева разряде (заим)
конец цикла
```

Прежде чем приводить методику выполнения пяти перечисленных этапов, заметим, что, учитывая особенности определения необходимых значений на этапах 3 и 5, их целесообразно объединить.

1–2. Выделение очередной двоичной цифры в уменьшаемом a и вычитаемом b

Методика выполнения этих этапов аналогична сложению:

```
цифра1 := a и 1
цифра2 := b и 1
```

3, 5. Определение результата вычитания в данном разряде (цифра_рез) и величины заимствования в соседнем слева разряде (заим)

Предварительный анализ табл. 2 показывает, что единую для всех восьми вариантов формулу расчета цифры вычитания и аналогичную формулу для определения величины заимствования составить нельзя — необходимо рассматривать разные группы вариантов. Чтобы не использовать два условных оператора (для нахождения значения *цифра_рез* и для нахождения значения *заим*) и проводится “объединение” этапов 3 и 5.

Все 8 вариантов расчета (см. табл. 2) можно объединить в три группы:

1) в которых *цифра*1 – *заим* ≥ *цифра*2

— где *заим* — величина заимствования после вычитания в соседнем справа разряде.

К этой группе относятся варианты 1, 3, 4, 7. Для них:

— цифра результата вычитания: *цифра_рез* = 0

— величины заимствования в соседнем слева разряде: *заим* = 0;

2) в которых *цифра*1 – *заим* = *цифра*2 – 1.

Указанному условию соответствуют варианты 3, 5, 8. Для них:

— *заим* = 0;

— значение *цифра_рез* может быть определено с использованием логической команды **И** (**AND**):

```
цифра_рез := (цифра1 - цифра2 + заим) и 1
```

(Убедитесь в этом самостоятельно!);

3) вариант 6

Для него

— *цифра_рез* = 0;

— *заим* = 1.

Для рассмотрения этих трех групп вариантов в программе придется использовать три неполных условных оператора или один так называемый “вложенный условный оператор”. Но можно упростить программную конструкцию, учитывая следующее.

Анализ табл. 2 показывает, что в вариантах 1, 3–6 и 8 значение *заим* в результате вычитания в данном разряде не меняется. Поэтому целесообразно рассмотреть отдельно только варианты 2 и 7:

```
|Рассматриваем вариант 2
если заим = 1 и цифра1 = 0 и цифра2 = 1
 то
  цифра_рез := 1
  заим := 1
 иначе
  |Для всех остальных вариантов:
 цифра_рез := (цифра1 - цифра2 + заим)
              и 1
  |Значение заим для варианта 7 меняется
 если заим = 1 и цифра1 = 1 и цифра2 = 0
    то
     заим := 0
  все
все
```

4. Учет цифры результата в искомой общей разности

Здесь, как и при сложении, применяем сдвиговую операцию СдвигВлевоНа:

```
разность := цифра_рез СдвигВлевоНа (номер_разряда - 1) + разность
```

Вся программа, реализующая вычитание:

```
алг Вычитание
нач цел a, b, разность, цифра1, цифра2,
        цифра_рез, заим, i
ввод a
ввод b
i := 1          |Обрабатываемый разряд
разность := 0 |Искомое значение
заим := 0
нц пока i <= 16
  цифра1 := a и 1 |Выделяем последние
  цифра2 := b и 1 |цифры
  |В i-м разряде определяем результат
  |вычитания и значение заимствования
  |в соседнем слева разряде
```

```
    |Рассматриваем вариант 2
    если заим = 1 и цифра1 = 0
         и цифра2 = 1
      то
        цифра_рез := 1
        заим := 1
      иначе
        |Для всех остальных вариантов:
  цифра_рез := (цифра1 - цифра2 + заим)
               и 1
   |Значение заим для варианта 7 меняется
    если заим = 1 и цифра1 = 1
     и цифра2 = 0
          то
            заим := 0
        все
    все
    |Учитываем цифра_рез в общей разности
    разность := цифра_рез СдвигВлевоНа
                (i - 1) + разность
    |Отбрасываем последнюю цифру
    |в уменьшаемом
    a := a СдвигВправоНа 1
    |и в вычитаемом
    b := b СдвигВправоНа 1
    |Новый номер обрабатываемого разряда
    i := i + 1
  кц
  |Выводим ответ
  вывод нс, разность
  кон
```

3. Умножение

Вспомним, как умножаются два двоичных числа “в столбик”:

```
                  1 0 1 1 1
              ×     1 0 1 1
              -----------------
                  1 0 1 1 1
            + 1 0 1 1 1
          0 0 0 0 0
      1 0 1 1 1
      -------------------------
      1 1 1 1 1 1 0 1
```

или

```
                  1 0 1 1 1
              ×     1 0 1 1
              -----------------
                  1 0 1 1 1
            + 1 0 1 1 1
      1 0 1 1 1
      -------------------------
      1 1 1 1 1 1 0 1
```

Можно сказать, что для умножения необходимо выделить каждую цифру одного из сомножителей, и если она равна 1, то выписать первый сомножитель в соответствующей позиции, а затем сложить все выписанные значения с учетом их позиции.

Прежде чем представлять программу, работающую по этой схеме, заметим, что сложение очередного “выписанного” значения первого множителя с учетом его позиции можно провести, как и ранее, с использованием сдвиговой операции СдвигВлевоНа:

```
  алг Умножение
  нач цел a, b, произв, цифра2, i
  ввод a
  ввод b
  произв := 0 |Искомое значение
  |Обрабатываемый разряд во 2-м множителе
  i := 1
  нц пока i <= 16
  |Выделяем последнюю цифру 2-го множителя
    цифра2 := b и 1
    если цифра2 = 1 |Только в этом случае
      то            |добавляем значение a
        произв := a СдвигВлевоНа
             (номер_разряда - 1) + произв
    все
    |Отбрасываем последнюю цифру
    |во 2-м множителе
    b := b СдвигВправоНа 1
    |Новый номер обрабатываемого разряда
    i := i + 1
  кц
  |Выводим ответ
  вывод нс, произв
  кон
```

4. Деление

Рассмотрим примеры деления чисел “в столбик”:

1) десятичных чисел:

```
8 1 0 4 5 2 | 6
6           ---------
            1 3 5 0 7 5
2 1
1 8
  3 0
  3 0
  ---
    0 4 5
      4 2
      ---
        3 1
        3 0
        ---
          1
```

2) двоичных чисел:

```
1 1 0 1 0 1 0 1 | 1 0 1
1 0 1           ---------
      1 1 0       1 0 1 0 1
      1 0 1
      -----
        1 1 0
        1 0 1
        -----
          1 1
```

Можно сказать, что при делении рассматриваются или “сносятся” вниз каждая из цифр делимого, которая добавляется к некоторому числу, которое будем называть “уменьшаемое”. Если оно не меньше дели-

теля, то, в случае двоичных чисел, в частное записывается очередная цифра, равная 1, и из уменьшаемого вычитается делитель. В противном случае в частное записывается очередная цифра 0[3]. Нетрудно увидеть, что полученное частное представляет собой так называемое "целочисленное частное", а последняя разность будет остатком от деления. Иными словами, мы проводим деление нацело с остатком.

Общая схема расчетов такая:

```
цикл для каждой цифры
1. Выделяем очередную цифру
2. Приписываем ее к "старому" значению уменьшаемого
3. Если текущее значение уменьшаемого не меньше делителя
    то
     3.1.1. Приписываем к частному 1
     3.1.2. Вычитаем делитель из уменьшаемого, присваивая разность тому же уменьшаемому
    иначе
     3.2.1. Приписываем к частному 0
    все
конец цикла
```

Так как в программе будет проводиться вычитание, целесообразно оформить вспомогательную функцию, возвращающую разность двух чисел. Это можно сделать на основе программы, разработанной для вычитания:

```
алг цел Вычитан(арг цел a, b)
нач цел разность, цифра1, цифра2,
        цифра_рез, заим, i
i := 1
разность := 0
заим := 0
нц пока i <= 16
  цифра1 := a и 1
  цифра2 := b и 1
  если заим = 1 и цифра1 = 0
       и цифра2 = 1
    то
      цифра_рез := 1
      заим := 1
    иначе
     цифра_рез := (цифра1 - цифра2 +
                   заим) и 1
     если заим = 1 и цифра1 = 1
          и цифра2 = 0
       то
         заим := 0
       все
  все
  разность := цифра_рез СдвигВлевоНа
              (i - 1) + разность
  a := a СдвигВправоНа 1
  b := b СдвигВправоНа 1
  i := i + 1
кц
знач := разность |Значение функции
кон
```

Нужно также знать количество цифр в делимом. Его также удобно определить с помощью вспомогательной функции (разработайте ее самостоятельно). Обозначим это количество — *k*1.

Обсудим методику выполнения этапов и подэтапов 1–3 (см. общую схему действий чуть выше).

1. Выделение очередной цифры делимого

Если, например, количество цифр делимого равно 6:

1	1	0	0	1	0

— то выделение крайней правой цифры можно провести так:

```
цифра := a и 1
```

— четвертой справа цифры:

```
цифра := a СдвигВправоНа 4 и 1
```

— *i*-й справа цифры:

```
цифра := a СдвигВправоНа(i - 1)и 1
```

2. Учет очередной цифры в "старом" значении уменьшаемого

Новое значение уменьшаемого определяется следующим образом:

```
уменьш := уменьш СдвигВлевоНа 1 + цифра
```

3.1.1, 3.2.1. Приписывание 1 или 0 в конец частного

Новое значение величин `частное` рассчитывается соответственно с помощью операторов:

```
частное := частное СдвигВлевоНа 1 + 1
```

и

```
частное := частное СдвигВлевоНа 1
```

3.1.2. Вычитание делителя из уменьшаемого

Здесь используется функция *Вычитан* (см. выше):

```
уменьш := Вычитан(уменьш, b)
```

Вся программа решения задачи имеет вид:

```
алг Деление_нацело_с_остатком
нач цел a, b, частное, k1, уменьш,
        цифра, i
ввод a |Делимое
ввод b |Делитель
k1 := КолЦифр(a)
|Начальное значение искомого частного
частное := 0
уменьш := 0   |и уменьшаемого
|Нумерация справа налево
нц для i от k1 до 1 шаг -1
  цифра := a СдвигВправоНа(i - 1)И 1
  уменьш := уменьш СдвигВлевоНа 1 + цифра
  если уменьш >= b
    то
      частное := частное СдвигВлевоНа 1 + 1
      уменьш := Вычитан(уменьш, b)
    иначе
      частное := частное СдвигВлевоНа 1
  все
кц
```

[3] Можно условно считать, что для приведенного примера частное начинается с двух нулей.

```
|Вывод ответа (частного и остатка)
вывод нс, частное, " ", уменьш
кон
```

Задания для самостоятельной работы

1. Разработайте аналогичные программы на языке Паскаль (задание для читателей, изучающих этот язык программирования).

2. Оформите фрагмент программы (на языке Паскаль или школьном алгоритмическом языке), связанный с выводом на экран значения результата, так, чтобы в нем использовались поразрядные логические и сдвиговые операции.

3. Определите результат сложения двух чисел a и b и типа `byte` при:

1) $a = 255$, $b = 1$;

2) $a = 255$, $b = 120$.

При использовании школьного алгоритмического языка результат получите на бумаге, приняв, что числа хранятся в восьми разрядах.

4. Определите, можно ли из меньшего натурального числа вычесть большее. Объясните полученный результат.

5. Оказывается (но не совсем очевидно), что для вычитания, как и для сложения, все-таки можно получить единую формулу для расчета результата вычитания в одном разряде:

```
Р := цифра1 - цифра2 - заим,
```

и после расчета значения P[4] выделить значение *цифра_рез* и новое значение *заим*:

```
цифра_рез := Р и 1
заим := (Р СдвигВправоНа 1) и 1
```

Проверьте справедливость сделанных утверждений применительно ко всем восьми вариантам табл. 2. Как можно объяснить это?

6. Проверьте, можно ли разработанную программу для вычитания применить к двум отрицательным числам (в языке Паскаль изменив при этом тип данных). Проанализируйте два случая:

1) при вычитании из большего отрицательного числа меньшего;

2) при вычитании из меньшего числа большего.

Объясните полученный во втором случае результат.

7. Определите, сколько двоичных разрядов будет в произведении, если сомножители занимают, соответственно, $k1$ и $k2$ разрядов.

8. Разработайте вариант программы для умножения, в которой поразрядное умножение проводится не для всех 16 разрядов, а для фактического количества разрядов в двоичном представлении второго множителя.

Литература

1. *Златопольский Д.М.* Логические и сдвиговые операции. / “В мир информатики” № 169 (“Информатика” № 15/2011).

2. Еще раз о поразрядных логических и сдвиговых операциях. / “В мир информатики” № 171 (“Информатика” № 17/2011).

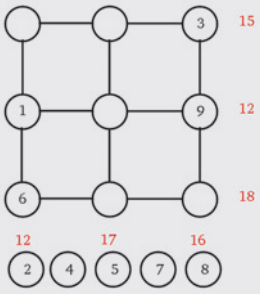
3. *Окулов С.М.*, *Лялин А.В.*, *Пестов О.А.* Алгоритмы целочисленной арифметики. / Информатика и образование, № 3/2011.

ЯПОНСКИЙ УГОЛОК

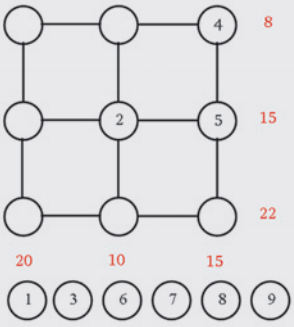
Сан-го-ку

Читателям раздела “В мир информатики”, конечно, известно, что такое “судоку”. Предлагаем подумать над еще одной японской головоломкой — “сан-го-ку”. Ее правила просты: необходимо расставить цифры в свободных кружочках на пустые места так, чтобы сумма цифр каждого ряда равнялась числу справа, а сумма цифр каждого столбца — числу снизу. Решите, пожалуйста, три сан-го-ку.

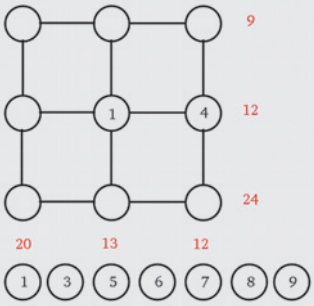
Сан-го-ку № 1



Сан-го-ку № 2



Сан-го-ку № 3



Ответы (можно не все) присылайте в редакцию.

[4] В программе на языке Паскаль эта величина должна быть описана как величина типа *integer* или *shortint*.

Два судоку

Решите, пожалуйста, два судоку:
1) простой:

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

2) сложный:

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

ЗАДАЧНИК

Задача, которую вы решаете, может быть очень скромной, но если она бросает вызов вашей любознательности и если вы решаете ее собственными силами, то вы сможете испытать ведущее к открытию напряжение ума и насладиться радостью победы.

Джордж Пойя

Кофе по утрам

По утрам пятеро друзей (Максим, Дмитрий, Борис, Иван и Вадим) встречаются за чашкой кофе. Один из них выпивает одну чашку кофе, другой — 4, третий — 5, четвертый — 6, пятый — 8, однако кто-то пьет кофе без сахара, другие кладут в свою чашку по 1, 2, 4, 6 кусочков сахара, и несколько человек пьют свой кофе с молоком. Выясните, сколько чашек кофе выпивает каждый; сколько кусочков сахара он кладет в кофе, и кто пьет кофе с молоком, а кто — нет, если известно следующее:

а) Иван кладет в кофе втрое больше кусочков сахара, чем тот, кто выпивает за день 4 чашки кофе;

б) трое, включая того, кто кладет в кофе 4 кусочка сахара, пьют кофе без молока;

в) Борис выпивает только одну чашку кофе в день, он пьет кофе без молока и без сахара;

г) Дмитрий пьет кофе и с молоком, и с сахаром;

д) Максим, который пьет кофе без молока, кладет в него вдвое меньше кусочков сахара, чем тот, кто выпивает вдвое больше него кофе;

е) Вадим выпивает на 2 чашки кофе больше, чем Иван — но Иван кладет в кофе на два куска сахара больше, чем Вадим.

Перестановка в фирме

В фирме “Рога и копыта” работали штатные и внештатные сотрудники, причем средняя зарплата штатных была 45 грошей, а внештатных — 11 грошей в месяц. Руководство решило перевести одного из штатных сотрудников во внештатные (не изменив его зарплату), в результате чего и у штатных, и у внештатных сотрудников средняя зарплата увеличилась на 2 гроша в месяц. Сколько всего сотрудников работало в фирме?

Старые календарики

Митя захотел узнать, на какие дни недели приходятся в 2013 году его день рождения, День Победы, День защитника Отечества и другие праздники. Однако, к сожалению, календари на 2013 год еще не изданы. Какой календарь из папиной коллекции календариков за каждый из последних 20 лет Митя может использовать?

Есть ли в коллекции старый календарик, который Митя мог бы использовать для тех же целей в 2012 году (приняв условно, что календаря на этот год нет)?

Автор задачи — Д.М. Златопольский

Лифт и мальчики

У грузового лифта на 1-м этаже 18-этажного дома собрались 17 мальчиков, которым нужно подняться наверх, причем всем на разные этажи. Лифтер же согласен сделать только один рейс на любой этаж, после чего пусть каждый пойдет

пешком. Известно, что мальчики с одинаковым неудовольствием спускаются вниз на один этаж и с одинаковым двойным неудовольствием поднимаются вверх на один этаж. Какой этаж нужно выбрать, чтобы "суммарное неудовольствие" было наименьшим? Задачу решите аналитически или/и использовав компьютер.

Автор задачи — И.Акулич

Пять емкостей

На столе поставлены в ряд бутылка, кружка, чашка, стакан и кувшин, причем точно в таком порядке, в каком они перечислены. В них находятся различные напитки: кофе, чай, квас, молоко и минеральная вода, но неизвестно, какой напиток в каком сосуде, кроме минеральной воды — она находится в бутылке. Если стакан поставить между сосудами с чаем и молоком (в данный момент он не стоит между ними), то по соседству с молоком будет квас, а кофе будет точно посередине. Определите, в какую посуду что налито.

ПОИСК ИНФОРМАЦИИ

Четыре вопроса

1. Что голландцы едят непременно каждое утро за завтраком, а французы — в конце обеда вместе с десертом?
2. Каким фруктом пахнет во всех комнатах голливудской звезды Эдди Мёрфи?
3. Как называется приз, вручаемый победителю международного кинофестиваля, проводимого в швейцарском городе Веве? Почему?
4. Какую частицу позаимствовали из романа "Поминки по Финнегану"?

"ЛОМАЕМ" ГОЛОВУ

Числовой ребус с книгами

Решите, пожалуйста, числовой ребус:
КНИГА + КНИГА + КНИГА = НАУКА

Две головоломки с числом 2012

Лейб Штейнгарц,
доктор педагогики,
г. Иерусалим, Израиль

1. Числовая лестница

Это задание — для всех ☺.

Для каждой из приведенных ниже строчек в правой части необходимо записать выражение с цифрами числа 2012 так, чтобы каждое равенство было верным. При этом должны быть выполнены следующие условия (в дополнение к условиям, указанным в первой головоломке):

— разрешается в любом примере использовать операции сложения, вычитания, умножения, деления, возведения в степень, извлечения квадратного корня и модуль (абсолютную величину) числа;

— разрешается использовать операцию вычисления факториала ("!"); напомним, что, например, $5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$, а $0! = 1$;

— разрешается использовать целую часть числа, которая обозначается $[x]$ (напомним, что целая часть числа x — это наибольшее целое число, не превосходящее числа x, например: $[4{,}7] = 4$, $[-4{,}7] = -5$);

— никакие другие операции использовать нельзя.

1 = …
10 = …
100 = …
1000 = …
10000 = …
100000 = …
1000000 = …
10000000 = …
100000000 = …
1000000000 = …
10000000000 = …
100000000000 = …
1000000000000 = …
10000000000000 = …
100000000000000 = …
1000000000000000 = …

2. Получить верное равенство

Это задание предназначено для учащихся 1–7-х классов.

Для каждой из приведенных ниже строчек в левой части необходимо записать выражение с цифрами числа 2012 так, чтобы каждое равенство было верным. При этом должны быть выполнены следующие условия:

1) в левой части каждого равенства должны использоваться обязательно цифры 2, 0, 1, 2 и только эти цифры (то есть выражение должны содержать две цифры 2 и по одной цифре 0 и 1);

2) цифры могут быть расположены в любом порядке;

3) можно использовать многозначные числа, например, 21;

4) разрешается использовать знаки сложения, вычитания, умножения и деления (никакие другие операции использовать нельзя);

5) можно использовать скобки.

… = 0;
… = 1;
… = 2;
… = 3;
… = 4;
… = 5;
… = 6;
… = 7;
… = 8;
… = 9;
… = 10;
… = 11;
… = 12.

Восемь монет

Восемь монет выложены в ряд:

Надо собрать их в две группы по 4 штуки в каждой группе (в которой монеты, например, лежат одна на другой), перекладывая за каждый ход одну монету. При этом перекладываемую монету можно соединять лишь с монетой, отделенной от нее двумя монетами (разрозненными или объединенными в группу).

Разработайте алгоритм решения задачи. В нем используйте команды вида **3** к **6**, где 3 — номер перекладываемой монеты, 6 — номер монеты, на которую кладется перекладываемая.

Ребусы по информатике

Решите, пожалуйста, еще ряд ребусов, которые подготовил Денис Синица, ученик гимназии № 1 имени К.Калиновского, г. Свислочь, Республика Беларусь (см. также предыдущий выпуск). Определите, с какой темой они связаны.

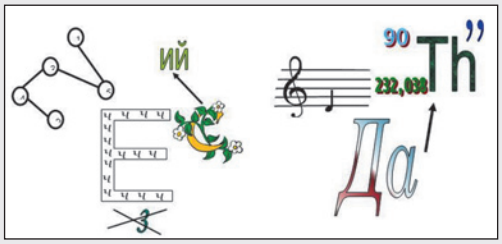
Ребус № 1

Ребус № 2

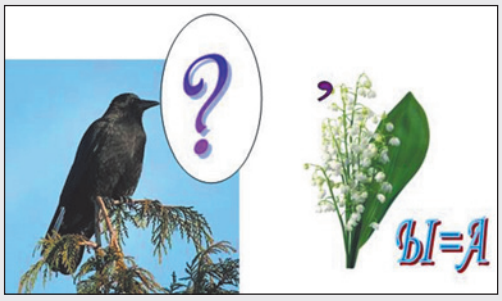
Ребус № 3

Ребус № 4

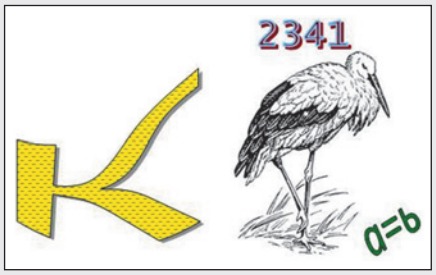
Ребус № 5



Ребус № 6



Ребус № 7

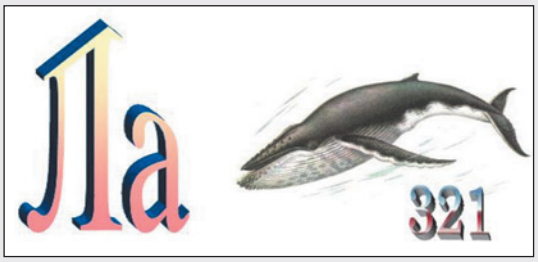


Ребус № 8

Ребус № 9

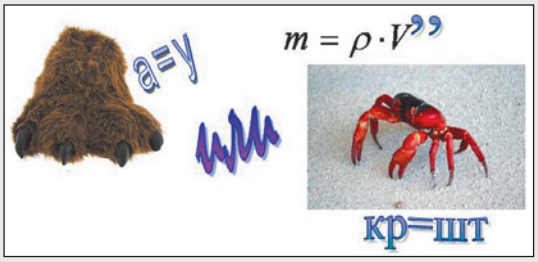
Ребус № 10

Ребус № 11



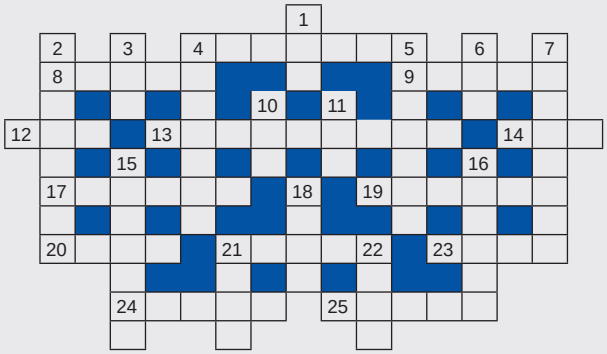
Ребус № 12

Ребус № 13



Кроссворд

Решите, пожалуйста, кроссворд.



По горизонтали

4. Часть персонального компьютера.

8. Прямоугольник, ограничивающий меню, или т.п.

9. Суетливый, слишком подвижный человек (не обязательно программист).

12. Название буквы древнерусского алфавита, напоминающей твердый знак.

13. Создание резервных копий файла.

14. Положение в компьютерной игре, имитирующей древнеиндийскую игру, при котором никто из двух участников не выигрывает.

17. Тип локальной сети, определяемый схемой соединения рабочих станций и сервера.

19. Операция, проводимая с файлом.

20. Дерево, из веток которого изготавливают венок для награждения победителя (не только олимпиад по программированию).

21. Два смежных байта оперативной памяти, а также последовательность букв и цифр, ограниченная с обоих концов пробелами, запятыми, точками, дефисами и т.п.

23. Единица измерения длины (но не файла).

24. Цифра восьмеричной системы счисления.

25. Программа, обладающая способностью к самовоспроизведению.

По вертикали

1. Единица измерения количества информации.

2. Совокупность правил, регламентирующих формат и процедуры обмена информацией между двумя или несколькими устройствами или процессами.

3. Характеристика файла, папки или переменной величины.

4. Так называют двумерный массив.

5. Место хранения информации в процессоре.

6. Число в системе условных обозначений символов.

7. Аргумент или результат процедуры.

10. В программировании — характеристика величины, определяющая множество ее допустимых значений и применимых к ней операций.

11. Величина изменения значения переменной цикла.

15. Часть стандартного устройства для ввода информации в компьютер.

16. Номер элемента массива.

18. Группа операторов и описаний, объединенных в одно целое, а также системный.

21. Несколько соединенных между собой компьютеров.

22. Цифра двоичной системы счисления.

5 золотых и 5 серебряных монет

Имеются 5 золотых и 5 серебряных монет одного веса. Фальшивомонетчик заменил одну золотую и одну серебряную монеты фальшивыми, неотличимыми внешне от настоящих. Фальшивые монеты легче настоящих и имеют одинаковый вес. Надо за три взвешивания на чашечных весах без гирь найти фальшивые монеты. Как это сделать?

ЗАДАЧНИК

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

Задача “Бой часов”

Напомним, что были предложены задания, связанные с двумя часами; первые бьют через каждые 2 сек., вторые — через каждые 3 сек. Совпавшие по времени удары воспринимаются как один. Следовало определить:

1) сколько времени часы били, если они начали бить одновременно и всего было насчитано 13 ударов;

2) через какое время общее число ударов часов станет равно 100?

Ответы — 18 и 148 сек. соответственно.

Ответы представили:

— Александрова Алена, Галикаев Азат и Кузуро Максим, Республика Башкортостан, г. Стерлитамак, школа № 24, учитель **Орлова Е.В.**;

— Андреева Ирина, Антонов Дмитрий и Иванова Елена, основная школа села Именево, Республика Чувашия, Красноармейский р-н, учитель **Тимофеева И.А.**;

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

— Бурикова Мария и Лукьянова Екатерина, Московская обл., г. Наро-Фоминск, школа № 1, учитель **Павлова Н.Н.**;

— Воскресенский Денис, Голик Екатерина и Горячкова Анастасия, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Гайсина Галия, Республика Башкортостан, г. Уфа, школа № 18, учитель **Искандарова А.Р.**;

— Герасимова Олеся и Цыплаков Евгений, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

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

— Дружинин Антон, Кулаев Николай и Леонгардт Анастасия, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Васнина О.В.**;

— Казанцева Татьяна, Салемальская школа-интернат, Тюменская обл., Ямало-Ненецкий автономный округ, Ямальский р-н, учитель **Амирасланов Т.В.**;

— Крысанов Виктор, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

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

— Ломакин Илья, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**

Алена Александрова, Ирина Андреева, Дмитрий Антонов, Юрий Базылев, Галия Гайсина, Екатерина Голик, Елена Иванова, Татьяна Казанцева и Анастасия Леонгардт разработали для решения компьютерные программы; Антон Дружинин, Николай Кулаев и Максим Кузуро — применили электронную таблицу Microsoft Excel.

Обратим внимание на то, что в ряде ответов получены ошибочные результаты.

Задача “Ученики и библиотеки”

Напомним условие: “В классе 35 учеников. Все они являются читателями школьной и районной библиотек. Из них 25 берут книги в школьной библиотеке, 20 — в районной. Сколько из них:

а) не являются читателями школьной библиотеки;

б) не являются читателями районной библиотеки;

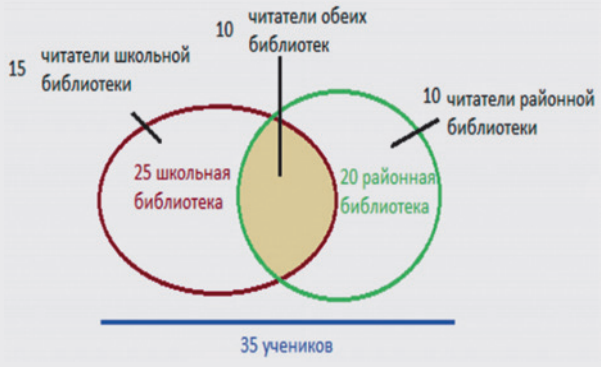
в) являются читателями только школьной библиотеки;

г) являются читателями только районной библиотеки;

д) являются читателями обеих библиотек?”.

Решение

Удобно использовать для решения так называемые “круги Эйлера” — геометрическую схему, наглядно иллюстрирующую отношения между подмножествами (“круги” — это условный термин, вместо них могут быть любые фигуры).



Из схемы легко и наглядно определяются ответы на первые два вопроса:

1) 35 – 25 = 10 учеников не являются читателями школьной библиотеки;

2) 35 – 20 = 15 учеников не являются читателями районной библиотеки.

Самое трудное ☺ — определить число читателей обеих библиотек. Так как 25 учеников берут книги в школьной библиотеке, 20 — в районной, а общее число читателей — 35, то 25 + 20 – 35 = 10 учеников являются читателями обеих библиотек. После этого определяются ответы на третий и четвертый вопросы:

3) 25 – 10 = 15 человек являются читателями только школьной библиотеки;

4) 20 – 10 = 10 учеников являются читателями только районной библиотеки.

Ответы представили:

— Алейникова Анастасия, Вадьковская средняя школа, Брянская обл., Погарский р-н, учитель **Алейникова Г.Н.**;

— Антонова Валерия, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

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

— Белов Дмитрий, Богданова Татьяна, Сараева Александра, Тохтасинова Диана и Хуснутдинова Олеся, Московская обл., г. Наро-Фоминск, школа № 1, учитель **Павлова Н.Н.**;

— Воскресенский Денис, Голик Екатерина, Горячкова Анастасия, Миноцкий Ян, Синицын Никита и Телегин Дмитрий, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Галикаев Азат, Республика Башкортостан, г. Стерлитамак, школа № 24, учитель **Орлова Е.В.**;

— Джурко Марина, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Васнина О.В.**;

— Корчагин Александр, основная школа поселка Михинский, Воронежская обл., Таловский р-н, учитель **Удалова А.А.**;

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

— Крысанов Виктор и Мельниченко Ангелина, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Ломакин Илья, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Муллин Айнур, село Комсомольское, Республика Чувашия, школа № 1, учитель **Родионов П.В.**;

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

— Селин Владислав, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Краснёнкова Л.А.**;

— Сетто Ольга, Удмуртская Республика, г. Можга, школа № 1, учитель **Колесникова С.В.**;

— Смирнов Иван, г. Новочеркасск, Суворовское училище МВД РФ, преподаватель **Воронкова О.Б.**;

— Цыплаков Евгений, г. Пенза, школа № 512, учитель **Гаврилова М.И.**

Задача “Бой часов”

Напомним, что были предложены задания, связанные с двумя часами; первые бьют через каждые 2 сек., вторые — через каждые 3 сек. Совпавшие по времени удары воспринимаются как один. Следовало определить:

1) сколько времени часы били, если они начали бить одновременно и всего было насчитано 13 ударов;

2) через какое время общее число ударов часов станет равно 100.

Ответы — 18 и 148 сек. соответственно.

Ответы представили:

— Александрова Алена, Галикаев Азат и Кузуро Максим, Республика Башкортостан, г. Стерлитамак, школа № 24, учитель **Орлова Е.В.**;

— Андреева Ирина, Антонов Дмитрий и Иванова Елена, основная школа села Именево, Республика Чувашия, Красноармейский р-н, учитель **Тимофеева И.А.**;

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

— Бирюкова Ирина и Хомутов Андрей, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Бурикова Мария и Лукьянова Екатерина, Московская обл., г. Наро-Фоминск, школа № 1, учитель **Павлова Н.Н.**;

— Воскресенский Денис, Голик Екатерина и Горячкова Анастасия, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Гайсина Галия, Республика Башкортостан, г. Уфа, школа № 18, учитель **Искандарова А.Р.**;

— Герасимова Олеся и Цыплаков Евгений, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

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

— Дружинин Антон, Кулаев Николай и Леонгардт Анастасия, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Васнина О.В.**;

— Казанцева Татьяна, Салемальская школа-интернат, Тюменская обл., Ямало-Ненецкий автономный округ, Ямальский р-н, учитель **Амирасланов Т.В.**;

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

— Ломакин Илья, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**

Алена Александрова, Ирина Андреева, Дмитрий Антонов, Юрий Базылев, Галия Гайсина, Екатерина Голик, Елена Иванова, Татьяна Казанцева и Анастасия Леонгардт разработали для решения компьютерные программы; Антон Дружинин, Николай Кулаев и Максим Кузуро — применили электронную таблицу Microsoft Excel.

Обратим внимание на то, что в ряде ответов получены ошибочные результаты.

Задачи на использование песочных часов

Напомним, что требовалось описать, как, используя двое песочных часов, отмерить некоторый непрерывный промежуток времени. Исходная информация представлена в таблице:

Номер задачи	Первые часы	Вторые часы	Время, которое требуется замерить
1	4 мин.	10 мин.	2 мин.
2	3 мин.	4 мин.	2 мин.
3	3 мин.	10 мин.	5 мин.

Решение показано в виде схем:

Задача 1

Вторые часы (10 мин.)	1			
Первые часы (4 мин.)	2	3	н	к

Задача 2

Первые часы (3 мин.)	1	3		
Вторые часы (4 мин.)	2		н	к

Задача 3

Первые часы (3 мин.)	1	3	4	5	6	
Вторые часы (10 мин.)	2				н	к

Примечание. На схемах: числа соответствуют номеру операции (установка и переворачивание часов), “н” — начало отмеривания, “к” — конец.

Ответы представили:

— Абаринова Светлана, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Ахметшин Адэль, Душутин Денис, Колесников Антон, Костылев Игорь, Михайлов Валерий, Разживина Ирина, Сетто Александра и Храбрых Ангелина, Удмуртская Республика, г. Можга, школа № 1, учитель **Колесникова С.В.**;

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

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

— Громова Елизавета, Королёва Виктория, Малютин Максим, Мицик Дарья, Паршин Вячеслав и Растригин Кирилл, Московская обл., г. Наро-Фоминск, школа № 1, учитель **Павлова Н.Н.**;

— Иванов Николай, Ростовская обл., г. Новочеркасск, Суворовское училище МВД РФ, преподаватель **Воронкова О.Б.**;

— Кулябин Максим и Скворцов Сергей, Удмуртская Республика, г. Можга, Центр дополнительного образования детей, руководитель кружка по информатике **Колесникова С.В.**;

— Левченко Ирина и Прохоров Виктор, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

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

— Сердюкова Алевтина, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**

В ряде ответов задача 2 решалась за 4 операции, а задача 3 — за 7 операций. Так что не все так просто, как кажется на первый взгляд, — всегда надо искать оптимальные варианты…

Цепочки из латинских букв

Напомним условие: “Первая строка состоит из одного символа — латинской буквы “A”. Каждая из последующих цепочек создается такими действиями: в очередную строку сначала дважды подряд выписывается предыдущая строка, затем справа приписывается буква, чей порядковый номер в алфавите соответствует номеру строки (на i-м шаге пишется i-я буква алфавита). Вот первые 4 строки, созданные по этому правилу:

(1) A
(2) AAB
(3) AABAABC
(4) AABAABCAABAABCD.

Запишите шесть символов подряд, стоящих в восьмой строке с 72-го по 77-е место (считая слева направо). Ответ обоснуйте.

Латинский алфавит (для справки): A B C D E F G H I J K L M N O P Q R S T U V W X Y Z”.

Решение

Прежде всего обратим внимание на то, что, конечно, требовалось решить задачу, не формируя все строки.

Сначала можно установить зависимость между номером цепочки и количеством букв в ней:

Номер	Количество
1	1
2	3
3	7
4	15
…	…

Эта зависимость: *количество* $= 2^{номер} - 1$.

Заметим также, что всего мы ищем 6 букв.

Далее можно составить таблицу:

Номер буквы в 8-й строке	1–127	128–254	255
“Исходная” строка	7	7	

Видно, что искомые буквы принадлежат 7-й строке (ее первой записи). Исследуем ее:

Номер буквы в 7-й строке	1–63	64–126	127
“Исходная” строка	6	6	

Искомые буквы принадлежат 6-й строке (в ее повторной записи) и имеют в ней начальный номер 72 – 64 + 1 = 9. Проанализируем 6-ю строку:

Номер буквы в 6-й строке	1–31	32–62	63
“Исходная” строка	5	5	

Видно, что искомые буквы принадлежат 5-й строке:

Номер буквы в 5-й строке	1–15	16–30	31
“Исходная” строка	4	4	

а точнее — 4-й:

Номер буквы в 4-й строке	1–7	8–14	15
“Исходная” строка	3	3	

Анализ показывает, что искомые буквы принадлежат 3-й строке (в ее повторной записи) и имеют в ней начальный номер 9 – 8 + 1 = 2. Поэтому из уже известной 3-й строки можно получить ответ: ABAABC.

Ответы представили:

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

— Васильев Андрей, Герасименкова Александра, Лопатинова Елена, Пазухин Данил, Попова Алена и Шахов Дмитрий, Московская обл., г. Наро-Фоминск, школа № 1, учитель **Павлова Н.Н.**;

— Горшкова Оксана, МБОУ “Кадетская школа”, Республика Татарстан, г. Чистополь, пгт Крутая Гора, учитель **Валиева Р.Н.**;

— Евграфов Алексей, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Иванов Глеб, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Кулаев Николай, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Васнина О.В.**;

— Кулябин Максим и Скворцов Сергей, Удмуртская Республика, г. Можга, Центр дополнительного образования детей, руководитель кружка по информатике **Колесникова С.В.**;

— Мейгли Эдуард, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Краснёнкова Л.А.**;

— Никаноров Андрей, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

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

— Фазлыев Фанис, Республика Башкортостан, г. Стерлитамак, школа № 24, учитель **Орлова Е.В.**

Задача “Состав экспедиции”

Напомним, что требовалось определить, кто из трех человек (Виктор, Антон и Сергей) какие должности занял в экспедиции (старший, радист и механик), если были высказаны предположения:

1) старшим будет назначен Антон;
2) Сергей не будет механиком;
3) Виктор будет утвержден радистом;
4) Виктор не будет старшим в экспедиции,

из которых только одно оказалось верным.

Решение

Утверждение 1 не может быть истинным, так как тогда истинным является и утверждение 4, то есть Антон не был назначен старшим в экспедиции.

Предположим, что единственным верным утверждением является утверждение 4. В этом случае старшим в экспедиции назначен Сергей (с Антоном мы уже “разобрались”). Но тогда истинным также будет утверждение 2, чего быть не может.

Значит, старшим являлся Виктор. Тогда Сергей — радист (утверждение 2 — единственное верное), а Антон — механик.

Ответы прислали:

— Алейникова Анастасия, Вадьковская средняя школа, Брянская обл., Погарский р-н, учитель **Алейникова Г.Н.**;

— Антонова Валерия, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Ахметшин Адэль, Душутин Денис, Колесников Антон, Костылев Игорь, Михайлов Валерий, Разживина Ирина, Сетто Александра и Храбрых Ангелина, Удмуртская Республика, г. Можга, школа № 1, учитель **Колесникова С.В.**;

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

— Близнецов Александр, Васильев Андрей, Герасименкова Александра, Лопатинова Елена, Пазухин Данил, Попова Алена, Роду Мария, Чириков Александр и Шахов Дмитрий, Московская обл., г. Наро-Фоминск, школа № 1, учитель **Павлова Н.Н.**;

— Гайсина Галия, Республика Башкортостан, г. Уфа, школа № 18, учитель **Искандарова А.Р.**;

— Голик Екатерина и Катышева Елизавета, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Дружинин Антон, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Васнина О.В.**;

— Иванов Николай, Ростовская обл., г. Новочеркасск, Суворовское училище МВД РФ, преподаватель **Воронкова О.Б.**;

— Кулябин Максим, Сергеева Полина и Скворцов Сергей, Удмуртская Республика, г. Можга, Центр дополнительного образования детей, руководитель кружка по информатике **Колесникова С.В.**;

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

— Марков Денис, Чувашская Республика, г. Канаш, Канашский педагогический колледж, преподаватель **Воеводина Р.В.**;

— Наделяев Денис и Селин Владислав, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Краснёнкова Л.А.**;

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

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

Статья “Еще одна задача о шахматной доске”

Напомним, что предлагалось перечислить все допустимые маршруты, ведущие на шахматной доске из начальной клетки *c*3 в клетку *a*1, соблюдая следующее правило: каждый ход делается либо на одну клетку влево, либо на одну клетку вниз, либо “вниз–влево” (условные обозначения ходов, соответственно, Л, Н, Д).

Ответ

Возможны 13 вариантов маршрутов:

ДД
ДЛН
ДНЛ
ЛДН
ЛНД
НДЛ
НЛД
ЛЛНН
ЛНЛН
ЛННЛ
НЛЛН
НЛНЛ
ННЛЛ

Обратим внимание на удобный способ формирования перечня маршрутов — сначала располагать их в порядке увеличения длины, а при одной и той же длине — в алфавитном порядке обозначения ходов (проанализируйте приведенный перечень).

Ответы представили:

— Александрова Алена, Республика Башкортостан, г. Стерлитамак, школа № 24, учитель **Орлова Е.В.**;

— Антонова Валерия, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

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

— Великороднов Виктор, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Герасименкова Александра и Щербакова Евгения, Московская обл., г. Наро-Фоминск, школа № 1, учитель **Павлова Н.Н.**;

— Голик Екатерина и Катышева Елизавета, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Даниленко Степан, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Доронина Надежда, Нармонская средняя школа, Республика Татарстан, Лаишевский р-н, учитель **Ожмекова Ю.Г.**;

— Егорова Светлана, Коротченко Ирина и Хомутов Андрей, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Голубков Дмитрий, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Васнина О.В.**;

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

Задание “Запросы к поисковому серверу”

Напомним, что необходимо было расположить следующие четыре запроса к поисковому серверу:

1	графика | литература
2	живопись | литература | графика
3	живопись & литература & графика
4	живопись & графика

в порядке возрастания количества страниц, которые найдет поисковый сервер по каждому запросу.

Ответ: 3 4 1 2 (указаны номера запросов).

Обоснование

По запросу 3 будут отобраны страницы, на которых присутствуют все три слова, представленные в нем. Ясно, что таких страниц будет меньше, чем страниц только со словами “живопись” и “графика” (запрос 4).

Символ “|” соответствует логической связке ИЛИ, поэтому по запросу 2 будет выдано больше страниц, чем по запросу 1.

Правильные ответы прислали:

— Алейникова Анастасия, Вадьковская средняя школа, Брянская обл., Погарский р-н, учитель **Алейникова Г.Н.**;

— Антипин Никита, Куминская средняя школа, Тюменская область, Ханты-Мансийский автономный округ — Югра, Кондинский р-н, учитель **Шишигина О.В.**;

— Антонова Валерия, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Ахметшин Адэль, Душутин Денис, Колесников Антон, Костылев Игорь, Михайлов Валерий, Разживина Ирина, Сетто Александра и Храбрых Ангелина, Удмуртская Республика, г. Можга, школа № 1, учитель **Колесникова С.В.**;

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

— Батурина Алена, Гладышева Виктория, Диденко Юлия, Журавлев Денис, Мрыкина Елизавета, Пальцева Евгения, Смирнов Константин, Чижиков Виталий и Яровая Ирина, Московская обл., г. Наро-Фоминск, школа № 1, учитель **Павлова Н.Н.**;

— Гайсина Галия, Республика Башкортостан, г. Уфа, школа № 18, учитель **Искандарова А.Р.**;

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

— Горшкова Оксана, МБОУ "Кадетская школа", Республика Татарстан, г. Чистополь, пгт Крутая Гора, учитель **Валиева Р.Н.**;

— Дукач Светлана, Кирсанова Алеся и Соболев Иван, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

— Иванов Николай, Ростовская обл., г. Новочеркасск, Суворовское училище МВД РФ, преподаватель **Воронкова О.Б.**;

— Кулябин Максим и Скворцов Сергей, Удмуртская Республика, г. Можга, Центр дополнительного образования детей, руководитель кружка по информатике **Колесникова С.В.**;

— Коркина Анна и Одинцова Екатерина, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Марков Денис, Чувашская Республика, г. Канаш, Канашский педагогический колледж, преподаватель **Воеводина Р.В.**;

— Мейгли Эдуард и Селин Владислав, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Краснёнкова Л.А.**;

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

— Фазлыев Фанис, Республика Башкортостан, г. Стерлитамак, школа № 24, учитель **Орлова Е.В.**

На вопрос о названии символа "&" ("амперсенд") ответили Анастасия Алейникова, Юрий Базылев, Карина Галушкова, Светлана Дукач, Алеся Кирсанова, Иван Соболев и "дружная команда" (так ребята представились в письме) из школы № 1 г. Можга.

Новые задачи

1. О шахматной доске

Имеется шахматная доска, на которой обозначен участок "запретных" клеток (куда заходить нельзя) — он оттенен.

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

Начальная клетка — h8. Выяснить, сколько всего маршрутов существует из начальной клетки в клетку a1, если ходить разрешается по следующим правилам: каждый ход делается либо на одну клетку влево, либо на одну клетку вниз. Перечислять все маршруты не нужно, требуется только указать их количество.

2. О запросах к поисковому серверу

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

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

Запрос	Найдено страниц (в тысячах)
Спартак | Локомотив	7770
Локомотив	5500
Спартак & Локомотив	1000

Какое количество страниц (в тысячах) будет найдено по запросу: *Спартак*? Считается, что все запросы выполнялись практически одновременно, так что набор страниц, содержащих все искомые слова, не изменялся за время выполнения запросов.

ВНИМАНИЕ! КОНКУРС

Конкурс № 85

В рамках данного конкурса предлагаем выполнить задания для самостоятельной работы, предложенные в статье "Моделирование арифметических действий в процессоре" (можно выполнять не все задания).

Конкурс № 86

В качестве задания данного конкурса предлагаем решить задачу "Лифт и мальчики" (см. рубрику "Задачник").

Ответы отправьте в редакцию до 1 ноября по адресу: 121165, Москва, ул. Киевская, д. 24, "Первое сентября", "Информатика" или по электронной почте: *vmi@1september.ru*. Пожалуйста, четко укажите в ответе свои фамилию и имя, населенный пункт, номер и адрес школы, фамилию, имя и отчество учителя информатики.

§ 31 ОКТЯБРЯ

Учительская фестиваль § книга
Москва-2012

Предметы естественно-научного цикла

● Математика ● Информатика ● Физика ● Биология ● Химия

БОЛЕЕ 1000 НАИМЕНОВАНИЙ КНИГ ДЛЯ УЧИТЕЛЯ ПРЕДСТАВЛЯЮТ

«Бином», «ВАКО», «ИЛЕКСА», «Интеллект-Центр», «Легион», «Мир Школьника», «Мнемозина», «Новый диск», «Первое сентября», «Рост», «1С», «СГУ», «СМИО Пресс», «Учитель», «Физикон», «Экзамен»

программа дня

9.30	**НАЧАЛО РАБОТЫ**			
10.00	**ОТКРЫТИЕ ДНЯ**			
10.30 – 11.45	Издательство «Интеллект-Центр» **Методическая помощь учителю – серия «Контрольные работы по математике в новом формате»** *А.В. Семенов,* к.п.н., зав. методической лабораторией математики, доцент кафедры математики МИОО; *М.Б. Крайнева,* к.п.н. методист методической лабораторией математики, доцент кафедры математики МИОО	Издательство «Экзамен-Медиа» **Мастер-класс по работе с интерактивной доской с использованием учебных пособий серии «Наглядная школа». Создание сценариев учебных эпизодов и их применение на уроках естественно-научного цикла (химии, физики, математики)** *О.Е. Рыбникова,* заслуженный учитель РФ, ГБОУ гимназия №1257; *А.В. Гурова,* учитель высшей категории, ГОУ СОШ №547; *А.А. Кудрявцев,* руководитель группы разработчиков электронных учебных пособий серии «Наглядная школа»	ИЦ «ВЕНТАНА-ГРАФ» **Система учебников «Алгоритм успеха»: реализация ФГОС ООО в линии учебников по биологии под ред. И.Н. Пономаревой** *С.Н. Исакова,* ведущий методист ИЦ «ВЕНТАНА-ГРАФ»	Издательство «Легион» **Решение алгебраических задач координатно-параметрическим методом** *С.Ю. Кулабухов,* к.физ.-мат.н., зам. генерального директора по научно-методической работе издательства «Легион»
12.00 – 13.15	Издательство «ИЛЕКСА» **Реализация ФГОС нового поколения в инновационном учебно-методическом комплекте по алгебре и началам математического анализа** *Е.П. Нелин,* к.п.н., проф., зав. кафедрой тестовых технологий и мониторинга качества образования Харьковского национального педагогического университета, автор учебников и учебных пособий по математике	Издательство «Экзамен» **ЕГЭ-2013. Задачи типа С** *В.С. Панфёров,* к.физ.-мат.н., доцент МГУ; *И.Н. Сергеев,* д.физ.-мат.н., профессор МГУ	Издательство «СМИО Пресс» **Новая организация повторения и возможности повышения качества обучения по школьному курсу алгебры** *С.Е. Злотин,* к.техн.н., сороковский учитель, 4-кратный лауреат конкурса школьных учителей фонда «Династия»	Издательство «Легион» **Пособия по биологии издательства «Легион» как средство развития ключевых и предметных компетенций школьников** *А.А. Кириленко,* методист кафедры математики и естественных дисциплин ИПК и ПРО РО
13.30 – 14.45	Издательство «Владос» **Реализация стандарта второго поколения в УМК по биологии издательства «Владос»** *А.И. Никишов,* к.п.н., профессор, заведующий кафедрой методики обучения биологии МПГУ; *А.Н. Соколов,* руководитель издательских проектов издательства «Владос»	Издательство «БИНОМ. Лаборатория знаний» **Электронный учебник: традиции + инновации, доступные каждому** *А.А. Елизаров,* руководитель отдела медиаресурсов издательства «БИНОМ. Лаборатория знаний»	ИЦ «ВЕНТАНА-ГРАФ» **Система учебников «Алгоритм успеха»: реализация ФГОС ООО в УМК А.Г. Мерзляка и УМК Ю.М. Калягина по математике** *И.И. Федотова,* методист ИЦ «ВЕНТАНА-ГРАФ»	Издательство «Легион» **Использование комплекта пособий по химии издательства «Легион» для подготовки к итоговой аттестации** *В.Н. Доронькин,* к.хим.н., доцент кафедры химии РГУПС

В программе возможны изменения и дополнения. Номера залов будут определены в день мероприятия.

Последние уточнения программы – на сайте **bookfair.1september.ru**

Вход свободный, но чтобы получить профессиональный подарок, пройдите заранее бесплатную регистрацию на сайте **bookfair.1september.ru**

Регистрация будет открыта с 10 октября 2012 года.

Выставка-продажа книг работает до 13.30

Все мероприятия фестиваля пройдут в московском государственном лицее № 1535 по адресу: ул. Усачева, дом 50 (в 3 минутах ходьбы от станции метро «Спортивная»). Телефон (499) 249-31-38. *Внимание! В лицее нет камеры хранения. Спасибо за понимание.*

Педагогический университет
«Первое сентября»
предлагает для учителя информатики

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

Дистанционные курсы повышения квалификации

вне зависимости от места проживания
(обучение с 1 января по 30 сентября 2013 года)

Имеются два варианта учебных материалов дистанционных курсов: брошюры и брошюры+DVD.
Курсы, включающие видеолекции (DVD), помечены значком
Нормативный срок освоения каждого курса – 72 часа.
Дополнительная информация – на сайте edu.1september.ru
Окончившие дистанционные курсы получают удостоверение установленного образца.
Базовая стоимость курса (без учета скидок) составляет
2190 руб. – для курсов без видеоподдержки,
2390 руб. – для курсов с видеоподдержкой.

Профильные курсы

код	
07-001	*И.Г. Семакин*. Информационные системы в базовом и профильном курсах информатики
07-008	*А.Г. Гейн*. Математические основы информатики
07-009	*С.Л. Островский*. Основы web-программирования для школьного «сайтостроительства»
07-010	*А.Г. Кушниренко, А.Г. Леонов*. Методика преподавания основ алгоритмизации на базе системы «Кумир»

Общепедагогические курсы

код	
21-001	*С.С. Степанов*. Теория и практика педагогического общения
21-002	*Н.У. Заиченко*. Методы профилактики и разрешения конфликтных ситуаций в образовательной среде
21-003	*С.Н. Чистякова, Н.Ф. Родичев*. Образовательно-профессиональное самоопределение школьников в предпрофильной подготовке и профильном обучении
21-004	*М.Ю. Чибисова*. Психолого-педагогическая подготовка школьников к сдаче выпускных экзаменов в традиционной форме и в форме ЕГЭ
21-005	*М.А. Ступницкая*. Новые педагогические технологии: организация и содержание проектной деятельности учащихся
21-007	*А.Г. Гейн*. Информационно-методическое обеспечение профессиональной деятельности педагога, педагога-психолога, работника школьной библиотеки
21-008	*А.Н. Майоров*. Основы теории и практики разработки тестов для оценки знаний школьников
21-009	*В.Д. Шадриков, И.В. Кузнецова, М.Д. Кузнецова*. Формирование и оценка профессиональных качеств современного педагога

Очные курсы повышения квалификации

для жителей Москвы и Московской области
(обучение с 11 февраля по 30 апреля 2013 года)

Нормативный срок освоения каждого курса – 72 часа.
Дополнительная информация – на сайте edu.1september.ru
и по телефону (499) 240-02-24 (звонки принимаются с 15.00 до 19.00).
Окончившие очные курсы получают удостоверение государственного образца.
Базовая стоимость курса (без учета скидки) – 5900 руб.

Я.Н. Зайдельман. Алгоритмизация и программирование: от первых шагов до подготовки к ЕГЭ
И.Б. Полякова. Современные фото- и видеотехнологии в деятельности учителя

Электронную заявку можно в режиме on-line подать на сайте
edu.1september.ru