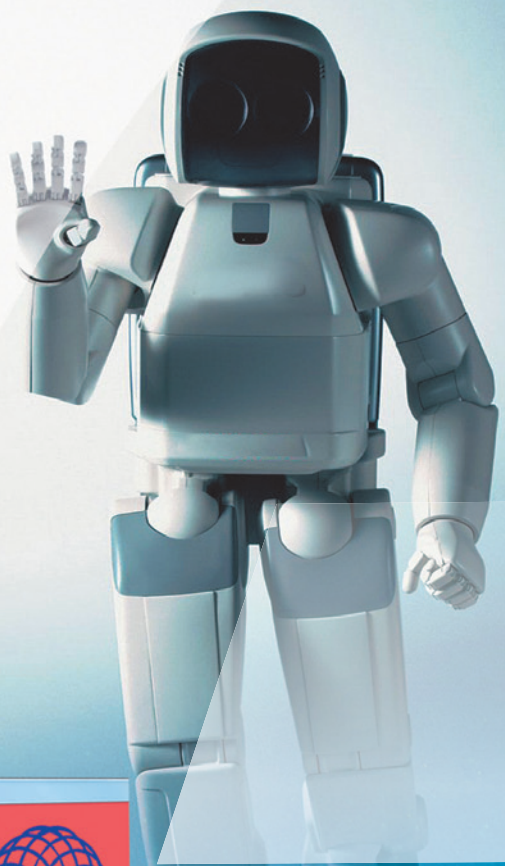
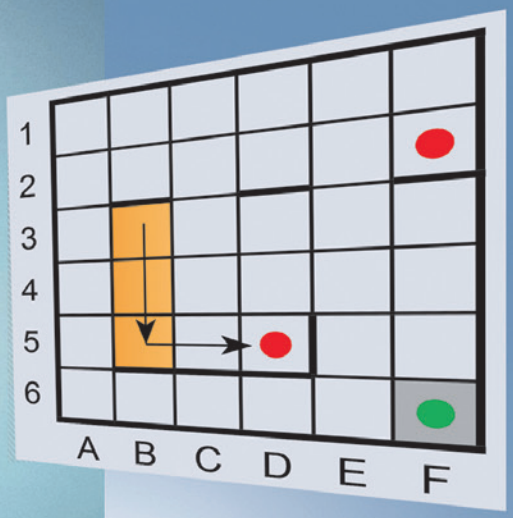
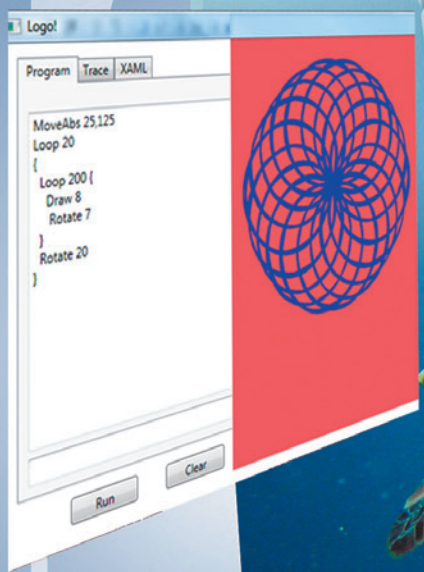
№7 Издание основано в 1995 г. inf.1september.ru
УЧЕБНО-МЕТОДИЧЕСКИЙ ЖУРНАЛ ДЛЯ УЧИТЕЛЕЙ ИНФОРМАТИКИ

ИНФОРМАТИКА







внутри номера CD и код доступа к электронной версии

издательский дом 1september.ru Первое сентября | август 2012
ИНФОРМАТИКА Подписка: «Роспечать» – 32291 (бумажная версия), 19179 (электронная); «Почта России» – 79066 (бумажная версия), 12684 (электронная)

НА ОБЛОЖКЕ

Даже те, кто твердо убежден в “ненастоящести” учебных исполнителей и уверен, что начинать учить алгоритмизации надо на “боевых” современных языках программирования, иногда не удержатся да и напишут какую-нибудь красивую “рекурсивную штучку” на Лого или особо хитрым способом обойдут лабиринт Роботом. Потому что интересно! И детям, и нам. Да не жили бы эти учебные среды так долго (и счастливо!), если бы не были востребованными и продуктивными.

В НОМЕРЕ

НА ДИСКЕ

ЭЛЕКТРОННЫЕ МАТЕРИАЛЫ:

- Программа “Учебный компьютер ЛамПанель”
- Исполнители и методические материалы к статье “Азбука Роботландии. Информация”
- Исходные файлы к статье “Ряды”
- 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 25 189 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 12.05.2012, фактически 12.05.2012
Заказ №
Отпечатано в ОАО “Первая образцовая типография” Филиал “Чеховский печатный двор”: ул. Полиграфистов, д. 1, Московская обл., г. Чехов, 142300
Сайт: www.chpk.ru,
E-mail: salas@chpk.ru,
Факс: 8(496) 726-54-10, 8(495) 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

Люди Y

До нового учебного года осталось совсем немного. Перед теми, кто, читая эти строки, подумал: ничего себе немного — месяц! — я “снимаю шляпу” ☺. Поскольку это означает, что они получили электронную версию августовского номера в день выхода — 1 августа. Для этого надо иметь особое мужество ☺.

Меня самого в школе в новом учебном году ждет изрядное количество проблем. Я до сих пор не до конца понимаю, какие классы буду учить. И есть большая вероятность, что не те, какие хотелось бы. В общем, обычная школьная рутина. Но, “прислушиваясь к себе”, все равно, даже с некоторым удивлением, констатирую — новый учебный год я все равно жду с радостью. Объяснять и видеть — что-то понимают! — поддерживать и видеть — бьются! — мне по-прежнему интересно, как и много лет назад. И мне хочется в школу.

В последние годы я сам много учился — не по своей основной, а по второй — управленческой специальности. Во-первых, конечно, не могу не отметить, что периодически оказываться в роли ученика, в ситуации, когда тебе преподаватель “с умным видом” преподносит “простейший” для него материал, а ты — ну, ни в какую не можешь понять, — очень отрезвляет. Даже дети в школе отметили, что у меня из лексикона начисто пропала и так не слишком часто встречавшаяся фраза “ну, что тут непонятного?”.

Но я не поэтому вспомнил об учебе. Есть такая интересная теория Дугласа Макгрегора. Ее подробное изложение легко найти в Интернете, но кратко суть заключается в следующем. Макгрегор считает, что всех людей можно отнести к одному из двух “классов” — люди X и люди Y. Если ваши глаза еще не успели прочитать следующий абзац, остановитесь! ☺ Постарайтесь догадаться, почему X и Y. Это не просто буквы — в них есть смысл.

Смысл букв X и Y такой. Люди X — это люди “нет-нет-нет”. Скрещенные руки в ответ на обращение — “нет, это не ко мне, не трогайте меня ради бога”. Люди Y — это люди “да-да-да”. Да, конечно, хочу, постараюсь, смогу! Макгрегор утверждал, что, согласно результатам его исследований, “XY-хромосома” достаточно устойчива и проявляется у человека в различных ситуациях. Правда это или нет — не знаю. Безусловно, как и всякая модель, данная ограниченна и предельно упрощена. А к моделям, претендующим на описание человеческой природы, я всегда относился с особой настороженностью. Но, согласитесь, идея красивая.

В общем, в новом учебном году, как бы тяжело ни было (а будет — сомнений нет), я лично постараюсь быть “игреком”. По себе знаю, как это трудно, но все равно не хочется становиться “иксом” ☺.

Сергей Островский,
главный редактор
(so@1september.ru)

Учебный компьютер “ЛамПанель”: практикум

К.Ю. Поляков,
д. т. н., Санкт-Петербург

Зачем это нужно?

Большинство школьников старших классов знают, что процессор “понимает” только один язык — язык машинных команд, которые представляют собой числовые коды. Писать программы в кодах очень трудно, поэтому программисты, как правило, составляют программы на алгоритмических языках (языках высокого уровня), которые затем с помощью специальных программ — трансляторов переводятся в коды, выполняемые процессором. Более того, “уровень” учебных языков все больше повышается, и сортировка списка во многих современных языках программирования выполняется одной командой.

Казалось бы, на этом можно и остановиться. Но любознательная человеческая натура неминуемо увидит здесь “белое пятно”: программируя только на языках высокого уровня, мы не понимаем, как же все-таки работает процессор, как он умудряется понимать двоичные коды команд, выполнять их, организовывать ветвления и циклы, выделять память для переменных и вызывать процедуры.

А надо ли в этом разбираться, ведь и так все работает? По сути, этот вопрос аналогичен другому вопросу: “А зачем изучать теорию, если я хорошо умею работать с сенсорным экраном *iPad*?” По мнению автора, для тех, кто хочет стать специалистом в области информационных технологий, необходимо глубокое понимание происходящих процессов. Практика показывает, что только в этом случае удается находить решения в нестандартных ситуациях, не описанных в руководствах, когда “мастера по нажатию на кнопки” оказываются бессильны. Таким образом, тот, кто понимает, как работает компьютер, может более успешно и эффективно решать сложные задачи.

Как же понять и изучить работу компьютера? Конечно, можно экспериментировать с реальным компьютером, используя программы-отладчики, которые позволяют ввести в память коды машинных команд и данных, а потом запустить программу на выполнение, начиная с некоторого адреса [1]. Но отладчикам всегда не хватало наглядности, поэтому

были разработаны учебные модели компьютеров, которые имеют простую (в сравнении с реальными процессорами) систему команд, но при этом позволяют наглядно показать все этапы выполнения программы [2–4]. Иногда строят и полные модели (*эмуляторы*) существующих процессоров [5].

В этой статье описывается еще одна учебная модель компьютера — программа *ЛамПанель* (это название образовано от слов *ламповая панель*). В отличие от других программ этого класса в качестве устройства вывода используется панель (матрица) лампочек размером 8 на 16, причем каждой лампочкой можно управлять независимо от других. Это приближает такую модель к реальным задачам управления ламповыми табло, например, в системах освещения, рекламных установках и информационных табло.

Возможности программы ориентированы на преподавание профильного курса информатики в школе. Основная форма изучения — практические работы. С помощью программы *ЛамПанель* можно изучать

- представление знаковых и беззнаковых целых чисел в памяти компьютера;
- принципы хранения программ и данных в компьютерах с архитектурой фон Неймана;
- алгоритм автоматической работы процессора;
- организацию ветвлений и циклов;
- принципы работы стека;
- организацию вызовов подпрограмм.

Непосредственный предшественник программы *ЛамПанель* — учебная модель компьютера “Е-97”, разработанная Е.А. Ереминым [3]. *ЛамПанель* имеет аналогичную архитектуру и использует похожую систему команд.

Для лучшего понимания материала статьи желательно предварительно познакомиться с основами компьютерной арифметики, например, по работе [6].

Как устроен учебный компьютер?

Процессор обрабатывает данные, используя сверхбыстродействующие ячейки собственной памяти — *регистры*. Процессор учебного компьютера *ЛамПанель* имеет только четыре 16-битных *регистра общего назначения*, которые называются **R0**, **R1**, **R2** и **R3**. В области 1 на *рис.* 1 вы видите двоичные значения этих регистров (они показаны черным цветом), шестнадцатеричные (синий цвет) и десятичные, без учета знака (зеленый цвет) и со знаком (коричневый цвет).

Ниже показаны еще три служебных регистра (**PC**, **SP** и **PS**), о которых мы поговорим позже.

Область 2 — это устройство вывода — ламповая панель, состоящая из 8 рядов, в каждом из которых по 16 лампочек. Состояние лампочек каждого ряда задается 16-битным числом, записанным в *порт* — регистр контроллера ламповой панели. Слева от каждого ряда записан его номер, точнее, номер порта (нумерация начинается с нуля). Единица в каком-то бите означает зажженную лампочку, а 0 — погашенную. Старший бит управляет самой левой лампочкой в ряду, младший — самой правой. Справа от каждого ряда показано значение, которое записано в порт (в шестнадцатеричной системе счисления).

Область 3 — это текстовый редактор, в котором набирается программа для процессора на специальном языке — *языке ассемблера*. Каждая команда этого языка соответствует одной машинной команде, но она записывается в символьном виде, а не как числовой код. Например, команда “*скопировать данные из регистра* **R2** *в регистр* **R3**”, имеющая шестнадцатеричный код 0123_{16}, может быть записана в виде

```
MOV R2, R3
```

где **MOV** — это название команды “*скопировать данные*” (сокращение от англ. *move* — переместить).

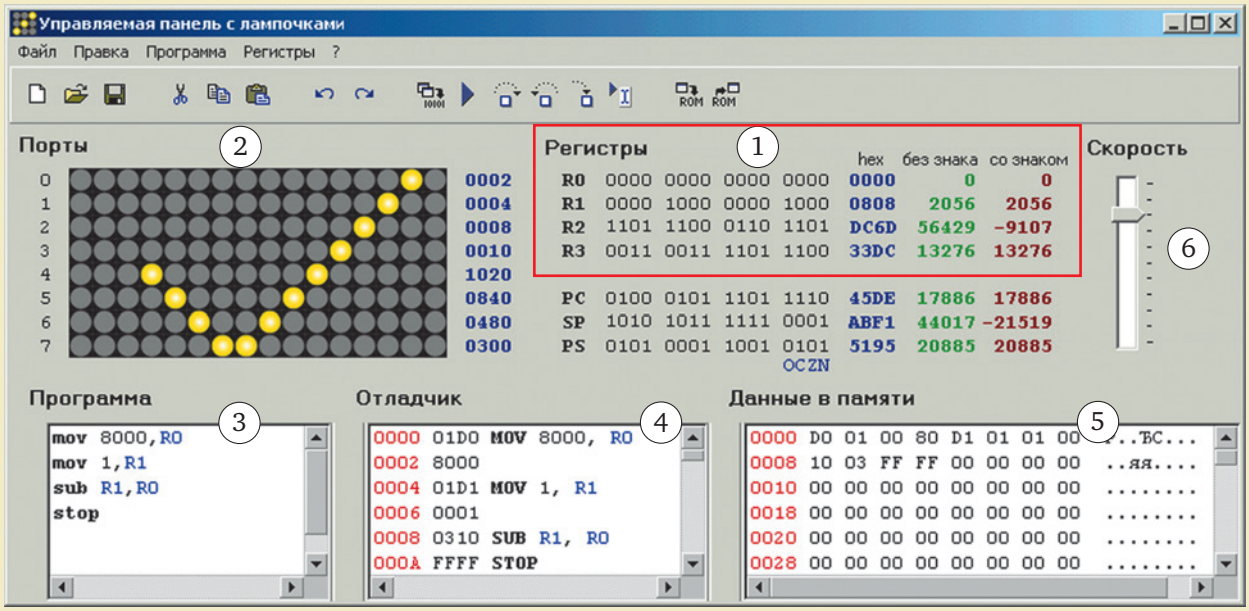


Рис. 1

Очевидно, что символьная запись команды значительно более понятна, чем число 0123_{16}. На псевдокоде эту операцию можно записать так: **R3 := R2**.

Команда **MOV** может не только выполнять копирование данных из регистра в регистр, но и присваивать новое значение регистру. Например, команда

```
MOV 12, R0
```

запишет число 12_{16} = 18 в регистр **R0** (все числа в программе записываются в шестнадцатеричной системе счисления). Эта команда кодируется в виде двух 16-битных кодов, которые записываются в память последовательно: сначала код команды $01D0_{16}$, а затем — число, пересылаемое в регистр:

```
01D0 MOV 12, R0  ; R0 := 12
0012
```

Точка с запятой означает начало *комментария*: ассемблер игнорирует (не обрабатывает) все символы в строке, расположенные после точки с запятой.

Нажав на клавишу F1, можно посмотреть описание всех команд языка ассемблера.

Чтобы программа остановилась, процессор должен выполнить команду **STOP**. Таким образом, простейшая программа состоит из одной команды **STOP**.

Каждая команда программы записывается в отдельной строке. Поэтому полная программа, которая копирует содержимое регистра **R2** в регистр **R3**, будет выглядеть так:

```
MOV R2, R3
STOP
```

При нажатии сочетания клавиш Ctrl + F9 запускается *ассемблер* (англ. *assembler* — сборщик) — программа, которая переводит программу на языке ассемблера в машинные коды, понятные процессору. Результат работы ассемблера — программа в машинных кодах — записывается в оперативную память (область 5 на *рис.* 1), размер которой в программе *ЛамПанель* составляет 256 байт. При нажатии на клавишу F9 программа начинает выполняться, скорость ее выполнения регулируется ползунком в правой части окна программы (область 6 на *рис.* 1).

Клавиша F8 позволяет выполнять программу по шагам — после выполнения очередной команды процессор останавливается и ждет следующего нажатия на клавишу F8 (или на клавишу F9, чтобы дальше выполнить программу без остановки). Во время паузы можно посмотреть, какие данные находятся в регистрах и в памяти; этот режим служит для отладки программ.

Компьютерная арифметика

Особенности компьютерной арифметики на уровне профильного школьного курса информатики подробно рассмотрены в статье [6], поэтому здесь мы не будем повторяться, а расскажем только о возможностях программы *ЛамПанель*.

Для того чтобы сложить два числа, применяют команду **ADD** (от англ. *add* — сложить). Например, команда

```
ADD 15, R0 ; R0 := R0 + 15
```

добавляет число 15_{16} = 23 к регистру **R0**. Можно добавить значение одного регистра к значению другого:

```
ADD R2, R3 ; R3 := R3 + R2
```

Существуют аналогичные команды, выполняющие другие арифметические действия:

SUB — вычитание (от англ. *subtract* — вычесть);

MUL — умножение (от англ. *multiply* — умножить);

DIV — целочисленное деление (от англ. *divide* — делить), остаток отбрасывается.

Команда **NOT** выполняет *инверсию* всех битов регистра, то есть меняет все нули на единицы, а единицы — на нули. Например, команда

```
NOT R0
```

выполнит инверсию регистра **R0**. С помощью этой команды можно, например, получить дополнительный двоичный код числа, находящегося в регистре **R0**, по классическому алгоритму [6]:

```
NOT R0
ADD 1, R0
```

Если первоначально в **R0** было записано число 21 = 15_{16}, то после выполнения первой команды (инверсии битов) значение регистра будет равно $FFEA_{16}$, а после добавления единицы — число $FFEB_{16}$, которое совпадает с дополнительным кодом числа (–21). Эти преобразования можно наглядно увидеть с помощью отладчика, запуская программу в пошаговом режиме (клавиша F8).

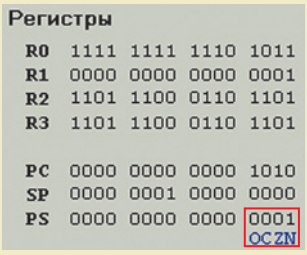


Рис. 2

Кроме регистров общего назначения, с которыми мы уже работали, в процессоре есть служебные регистры. Один из них — *регистр состояния процессора* **PS** (англ. *Processor State register* — регистр состояния процессора), биты которого называются также *флагами состояния* (флаг может быть "спущен" или "поднят", эти состояния обозначаются как 0 и 1 соответственно).

Фактически используются только четыре младших бита регистра **PS** (см. *рис.* 2), которые показывают, какой результат был получен в результате последней операции:

- **бит O** (от англ. *overflow* — переполнение) установлен (равен 1), если произошло **переполнение** разрядной сетки [6], то есть результат вычислений неверный; в остальных случаях бит O сброшен (равен 0);
- **бит C** (от англ. *carry* — перенос) установлен, если произошел **перенос** бита из разрядной сетки [6]; в остальных случаях сброшен;

• **бит Z** (от англ. *zero* — ноль) установлен, если результат последней операции — ноль; в остальных случаях сброшен;

• **бит N** (от англ. *negative* — отрицательный) установлен, если результат последней операции отрицательный; в остальных случаях сброшен.

Эти биты могут учитываться при выполнении ветвлений. Например, для организации цикла используют команду перехода

JNZ *метка*

Если результат предыдущей операции НЕ равен нулю, то происходит переход на указанную метку. Вот пример программы, которая вычисляет сумму натуральных чисел от 1 до 5 в регистре **R0**:

```
MOV 0, R0    ; начальное значение суммы
MOV 5, R1    ; количество шагов цикла
m:           ; метка обозначает начало цикла
  ADD R1, R0 ; R0 := R0 + R1
  SUB 1, R1
  ; R1 := R1 - 1 — оставшееся число шагов
JNZ m        ; переход на метку "m",
             ; если получился не ноль
STOP
```

Здесь в каждой строчке после точки с запятой записан комментарий, объясняющий ее действие. Давайте проследим, как выполняется эта программа:

Команда	R0	R1	бит Z	переход
MOV 0,R0	0		1	
MOV 5,R1		5	0	
ADD R1,R0	5		0	
SUB 1,R1		4	0	
JNZ m			0	да
ADD R1,R0	9		0	
SUB 1,R1		3	0	
JNZ m			0	да
ADD R1,R0	12		0	
SUB 1,R1		2	0	
JNZ m			0	да
ADD R1,R0	14		0	
SUB 1,R1		1	0	
JNZ m			0	да
ADD R1,R0	15		0	
SUB 1,R1		0	1	
JNZ m			1	нет
STOP				

Цикл заканчивается, потому что бит **Z** равен 1 (результат последней операции вычитания — ноль) и перехода по команде **JNZ** не происходит.

Кроме команды **JNZ**, существуют и другие команды перехода:

JMP *метка* — безусловный переход;

JGE *метка* — переход, если результат больше или равен нулю;

JL *метка* — если результат меньше нуля;

JZ *метка* — если результат равен нулю;

JLE *метка* — если результат меньше или равен нулю;

JG *метка* — если результат больше нуля.

Для проверки условий без изменения значений регистров можно применять команду сравнения **CMP** (от англ. *compare* — сравнить). Сравнивать можно число с регистром:

```
CMP 12, R0          ; сравнить 12_16 и R0
```

или регистр с регистром

```
CMP R2, R3          ; сравнить R2 и R3
```

Смысл сравнения состоит в том, чтобы установить *флаги* — биты регистра состояния **PS** — по результату: разности второго и первого операндов. При этом значения регистров не изменяются. Например, вторая приведенная команда сравнения вычисляет разность **R3-R2** и устанавливает флаги состояния по этой разности (например, если эти регистры равны, будет установлен бит **Z**).

В программе *ЛамПанель* можно использовать битовые логические операции [6]: “НЕ” (уже знакомая нам команда **NOT**), “И” (команда **AND**), “ИЛИ” (команда **OR**) и “исключающее ИЛИ” (команда **XOR**). В последних трех командах после названия команды сначала указывается маска, а затем через запятую — регистр, к которому применяется логическая операция. Например, команда

```
AND FF,R0
```

обнуляет старшие 8 бит (старший байт) регистра **R0**. Маска может находиться в регистре, например, последовательность команд

```
MOV FF,R1
OR R1,R0
```

устанавливает в единицу 8 младших бит регистра **R0**, а остальные оставляет без изменений.

Для выполнения сдвигов (см. [6]) используются следующие команды:

```
SHL 1,R0 ; логический сдвиг
         ; влево на 1 бит
SHR 2,R0 ; логический сдвиг вправо
         ; на 2 бита
SAR 1,R0 ; арифметический сдвиг
         ; вправо на 1 бит
ROL 2,R0 ; циклический сдвиг влево
         ; на 2 бита
ROR 3,R0 ; циклический сдвиг вправо
         ; на 3 бита
```

Конечно, сдвиг может применяться к любому регистру общего назначения, а не только к **R0**.

Практикум

1. Используя команду **MOV**, напишите программу, которая заполнит регистры так, как на рисунке. Не забудьте закончить программу командой **STOP**.

```
Регистры
 R0  1111 0000 0000 0000
 R1  1111 1111 0000 0000
 R2  1111 1111 1111 0000
 R3  1111 1111 1111 1111
```

Запишите, какие десятичные числа были только что записаны в регистры:

Регистр	Десятичные значения	
	без учета знака	с учетом знака
R0		
R1		
R2		
R3		

2. Выполните программу

```
SUB 1, R0
NOT R0
STOP
```

при различных начальных значениях регистра **R0** и запишите десятичные значения, полученные в **R0** после выполнения программы:

До	После	
	без учета знака	с учетом знака
5		
10		
25		

Какую операцию выполняет этот алгоритм?

3. Используя программу *ЛамПанель*, вычислите арифметические выражения и запишите результаты в таблицу. Объясните полученные результаты.

Выражение	Результат	
	без учета знака	с учетом знака
65530 + 9		
32760 + 9		
8 - 10		

Подсказка: **65535 = FFFF**$_{16}$, **32767 = 7FFF**$_{16}$

4. Вычислите приведенные выражения с помощью программы. Запишите в таблицу результаты, значения знакового (старшего) бита полученного числа и битов состояния:

Выражение	Результат		Знаковый бит	Биты состояния			
	без учета знака	с учетом знака		**O**	**C**	**Z**	**N**
32760 + 32752							
-32760 - 32752							
256 - 256							

5. С помощью программы, приведенной в теоретической части, вычислите сумму натуральных чисел от 1 до 100.

6. Напишите программу, которая вычисляет значение факториала — произведения всех натуральных чисел от 1 до заданного числа. Например, факториал числа 5 равен 5! = 1 · 2 · 3 · 4 · 5. С помощью программы заполните таблицу:

N	**N!**	
	без учета знака	с учетом знака
5		
6		
7		
8		
9		

Объясните полученные результаты.

1. Напишите программу, которая решает следующую задачу, используя логические операции:

В регистрах **R1**, **R2** *и* **R3** *записаны коды трех десятичных цифр, составляющих трехзначное число (соответственно сотни, десятки и единицы). Построить в регистре* **R0** *это число. Например, если* **R1**=31_{16}, **R2**=32_{16} *и* **R3**=33_{16}, *в регистре* **R0** *должно получиться десятичное число 123.*

2. Используя программу *ЛамПанель*, определите и запишите в таблицу значения регистра **R0** после выполнения каждой из следующих команд, которые выполняются последовательно:

	Команда	**R0**
1	**MOV 1234, R0**	
2	**XOR ABCD, R0**	
3	**XOR ABCD, R0**	

Ответьте на вопросы:

- как изменится результат выполнения программы, если в команде 1 записать в **R0** другое число?
- как изменится результат выполнения программы, если в командах 2 и 3 заменить маску на другую, например, на CB24$_{16}$?
- как изменится результат выполнения программы, если маску в команде 2 изменить, а маску в команде 3 не менять?

3. Запишите в таблицу десятичные числа, которые будут получены в регистре **R0** после выполнения каждой команды этой программы при разных начальных значениях **R0** (две команды выполняются последовательно одна за другой):

Начальное значение	**255**	**254**	**252**	**-255**	**-254**	**-252**
SHR 2,R0						
SHL 2,R0						

В каком случае последовательное выполнение этих двух команд не изменяет данные?

4. Напишите программу, которая решает следующую задачу, используя логические операции и сдвиги:

При кодировании цвета используются 4-битные значения составляющих R (красная), G (зеленая) и B (синяя). Коды этих составляющих записаны в регистрах **R1**, **R2** *и* **R3**. *Построить в регистре* **R0** *полный код цвета. Например, если* **R1** = A_{16}, **R2** = B_{16} *и* **R3** = C_{16}, *в регистре* **R0** *должно получиться число* ABC_{16}.

5. Напишите программу, которая умножает число в регистре **R0** на 10, не применяя команду умножения. Используйте арифметические операции и сдвиги.

Программа и данные

Итак, мы научились работать с регистрами, используя арифметические и логические операции. Теперь пришло время разобраться, как компьютер работает с памятью и как организуется автоматическое выполнение программы.

Память (см. область 5 на *рис.* 1) разбита на ячейки размером 1 байт (8 бит). Значение каждой ячейки записывается в виде двух шестнадцатеричных цифр — каждая из них представляет ровно четыре бита.

Каждая строчка в окне *Данные в памяти* содержит значения 8 байтов памяти; число слева, выделенное красным цветом, — это *адрес* (номер) первой ячейки, показанной в этой строке. Справа от шестнадцатеричных кодов выведена символьная строка из 8 символов — те же данные, только представленные как символы.

Данные можно записывать в память напрямую, используя команду **DATA**, например, можно набрать такую программу:

```
DATA 3132
DATA FFFF
```

Если теперь нажать клавиши Ctrl + F9, происходит *ассемблирование* (“сборка”) — перевод программы в машинные коды, и эти коды записываются в память (см. *рис.* 3).

Посмотрим на окно отладчика. В память записаны два 16-битных слова (4 байта), 3132_{16} и $FFFF_{16}$, причем эти слова процессор распознал как две команды:

```
MOV R3,R2
STOP
```

Такой обратный перевод из числовых кодов команд в их символьное обозначение называется *дисассемблирование* (обратное ассемблирование, “разборка”).

Любая машинная команда в компьютере *ЛампПанель* состоит из целого числа 16-битных слов, то есть из четного числа байтов. Поэтому в окне отладчика нумерация содержит только четные адреса ячеек памяти (команда не может начинаться с ячейки, имеющей нечетный адрес).

Теперь посмотрим на окно *Данные в памяти* (см. *рис.* 3). Видим, что байты 16-битного слова 3132_{16} расположены в памяти “наоборот” — сначала младший байт 32_{16}, а затем — старший 31_{16}. Кроме того, в правой части окна видно, что эти коды соответствуют символам “21яя”. Все специальные коды (не соответствующие каким-то принятым изображениям символов) обозначены точками. Таким образом, компьютер, основанный на архитектуре фон Неймана, не может самостоятельно различить, где данные, а где команды.

Выполним программу в пошаговом режиме. После первого нажатия на клавишу F8 в регистр **PC** (англ. *Program Counter* — программный счетчик) записывается стартовый адрес 0 (*рис.* 4), с которо-

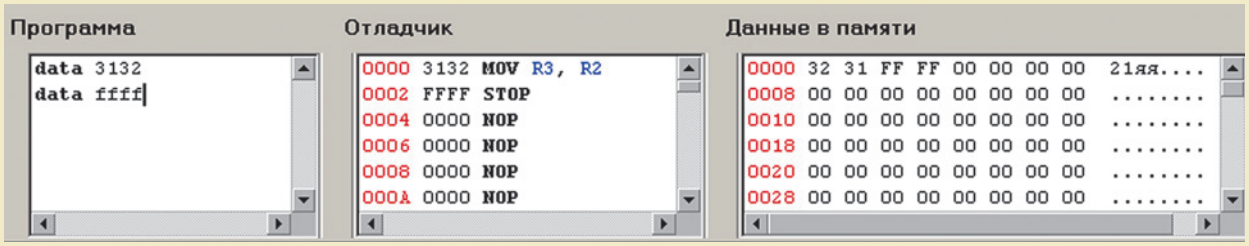


Рис. 3

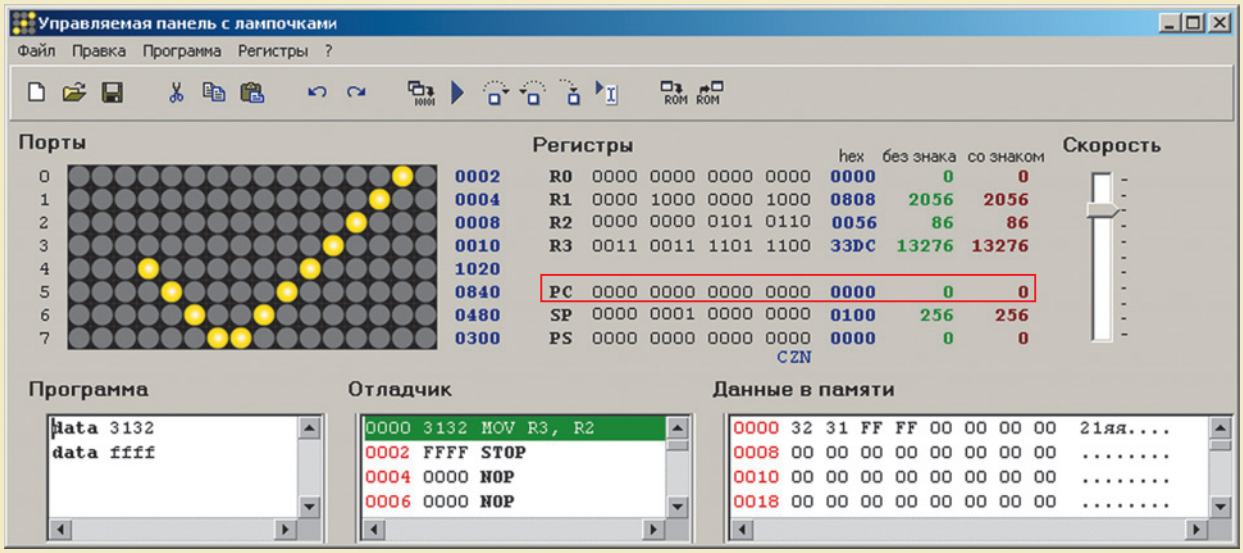


Рис. 4

го начинается выполнение программы. В окне *Отладчик* зеленым цветом выделена первая команда. Она еще не выполнялась, но будет выполнена при повторном нажатии F8. При этом регистр **PC** будет указывать на начало следующей команды (которая еще не выполнялась).

Таким образом:

• регистр **PC** всегда содержит адрес следующей команды;

• процессор воспринимает байты, расположенные по этому адресу, как код команды (а не как данные);

• программа всегда начинает выполняться с некоторого известного адреса, который "вшит" в компьютер и автоматически заносится в регистр **PC** при его включении;

• программа останавливается, когда будет выполнена команда **STOP** с кодом $FFFF_{16}$.

Заметим, что команды, содержащие числовые данные, могут занимать в памяти два 16-битных слова (см. *рис.* 5).

Для того чтобы обрабатывать данные из оперативной памяти, процессор должен загрузить их в регистры. Поскольку программа и данные расположены в одной области памяти, размещать данные можно сразу после команды **STOP**:

```
...
STOP
D:
  DATA 1234
```

Метка **D** нужна для того, чтобы удобно было загружать адрес блока данных в регистр, например, так:

```
MOV @D, R0 ; загрузить адрес метки D в R0
```

Можно считать, что **D** — это переменная программы. После этого легко загрузить в регистр данные из памяти:

```
MOV (R0),R1 ; загрузить в R1 данные,
            ; адрес которых записан в R0
```

Запись **(R0)** означает "значение, находящееся в памяти по адресу, записанному в **R0**", это так называемый *косвенный способ адресации*, когда в регистре находится адрес данных, а не значение. Аналогичной командой можно изменить содержимое ячейки памяти:

```
MOV R2,(R0) ; записать данные из R2
            ; в ячейку, адрес которой
            ; записан в R0
```

Заметим, что так можно сразу обратиться к любой ячейке памяти, поэтому такой вид памяти называется *памятью с произвольным доступом* (англ. *RAM = random access memory*).

Как вы знаете, минимальная ячейка памяти, имеющая собственный адрес, называется байтом. В современных компьютерах 1 байт состоит из 8 битов. Поэтому компьютер должен иметь возможность работать не только с 16-битными словами, но и с отдельными байтами. Покажем, как это происходит на примере задач обработки символьных строк, записанных в однобайтной кодировке.

Для того чтобы разместить символьную строку в памяти, используется команда **DATA**:

```
D:
  DATA "ABCDEFG"
```

Символы строки **"ABCDEFG"** записываются в память последовательно, начиная с первого.

Теперь запишем адрес строки в какой-нибудь регистр, например, в **R0**:

```
MOV @D,R0
```

Чтобы работать с отдельными байтами, используют байтовые версии команд, которые заканчиваются на латинскую букву B. Например, байтовый вариант команды **MOV** называется **MOVB**. Команда

```
MOVB (R0), R2
```

загружает один байт из памяти (расположенный по адресу, который записан в **R0**) в регистр **R2**. Полный список команд можно посмотреть в справочной системе, нажав на клавишу F1.

Рассмотрим такую задачу — преобразовать все заглавные латинские буквы в строчные. Для этого нужно посмотреть, чем отличаются двоичные коды заглавных и строчных букв:

```
A: 01000001   C: 01000011
a: 01100001   c: 01100011

B: 01000010   D: 01000100
b: 01100010   d: 01100100
```

Оказывается, коды заглавных и соответствующих строчных букв отличаются одним битом (этот бит выделен красным цветом в таблице). Поэтому, для того чтобы получить из заглавной буквы строчную букву, нужно установить 5-й бит (биты нумеруются справа налево, начиная с нулевого). Для этого можно, например, использовать логическую операцию "ИЛИ" с маской 0020_{16}, в которой 5-й бит установлен, а остальные — сброшены:

```
OR 20,R2
```

Затем нужно записать результат обратно в память, по адресу, находящемуся в **R0**:

```
MOVB R2,(R0)
```

Для перехода к следующему символу просто увеличиваем **R0** на единицу, сдвигаясь к следующему байту:

```
ADD 1,R0
```

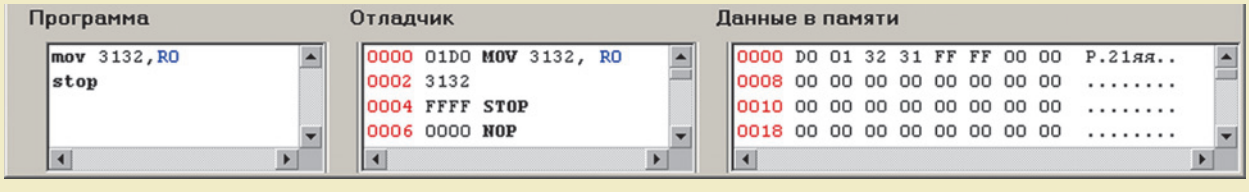


Рис. 5

и выполняем те же самые команды. Чтобы обработать 6 символов, можно организовать цикл со счетчиком в регистре **R1**:

```
MOV @D,R0 ; адрес данных — в R0
MOV 6,R1  ; счётчик шагов цикла
          ; (сделать 6 раз)
M:
  MOVB (R0),R2 ; прочитать байт из памяти
  OR 20,R2     ; заглавную — в строчную
  MOVB R2,(R0) ; записать байт в память
  ADD 1,R0     ; к следующему байту
  SUB 1,R1     ; уменьшить счетчик
               ; оставшихся шагов
JNZ M          ; если не все сделали —
               ; переход на метку M
STOP
D:
  DATA "ABCDEFG"
```

Отметим, что две команды

```
MOVB R2,(R0) ; записать байт в память
ADD 1,R0     ; перейти к следующему байту
```

можно заменить на одну, которая делает то же самое:

```
MOVB R2,(R0)+ ; записать байт в память
              ; и перейти к следующему
              ; байту
```

Практикум

1. Введите программу

```
DATA 01D0
DATA 3536
DATA 0101
DATA FFFF
```

Используя дисассемблер, запишите эту программу на языке ассемблера. Запишите содержимое памяти, в которой располагается эта программа, в виде последовательности символов.

2. Как вы думаете, какой код будет иметь команда **MOV R3,R2**? Проверьте свой ответ с помощью программы.

3. Блок данных программы выглядит так.

```
A:
  DATA 1234
B:
  DATA 4321
SUM:
  DATA 0
```

Напишите программу, которая складывает переменные **A** и **B** и записывает результат в переменную **SUM**.

4. Напишите программу, которая преобразует строчные буквы в заглавные, используя байтовые операции. Что произойдет, если среди исходных данных уже есть заглавные буквы?

5. Усовершенствуйте программу так, чтобы цикл останавливался не после заданного количества букв, а тогда, когда очередной прочитанный байт равен 0.

6. Поскольку в компьютере с архитектурой фон Неймана программа и данные расположены в одной области памяти, программа может менять свой собственный код. Напишите какую-нибудь программу, которая изменяет сама себя во время работы.

Ламповая панель

Наконец, мы подошли к самой интересной возможности программы *ЛамПанель* — управлению ламповой панелью. Ламповая панель (область 2 на *рис.* 1) — это устройство вывода.

Обмен данными процессора и внешнего устройства происходит через *порты* — регистры контроллера панели. У ламповой панели 8 портов, которые называются **P0**, **P1**, **P2**, **P3**, **P4**, **P5**, **P6** и **P7**. Каждый порт "отвечает" за одну строку лампочек; например, для того чтобы "зажечь" всю верхнюю строку, нужно записать в порт **P0** код $FFFF_{16}$ (все 16 бит — единичные, все лампочки горят). Для этого можно использовать, например, команды

```
MOV FFFF, R0
OUT R0, P0 ; запись значения R0 в порт P0
```

К сожалению, записать число сразу в порт нельзя — сначала нужно записать его в регистр общего назначения (в данном примере — в **R0**), а потом — из регистра в порт.

Для того чтобы изменить второй сверху ряд лампочек, нужно записать новое значение в **P1** и т.д.; последний ряд управляется портом **P7**. Например, для того чтобы все ряды лампочек горели одинаково, можно сначала записать нужный код в регистр:

```
MOV AAAA, R0
```

а затем из этого регистра — во все порты:

```
OUT R0, P0
OUT R0, P1
…
OUT R0, P7
```

Здесь многоточие обозначает аналогичные команды записи содержимого регистра **R0** в порты **P2**…**P6**. Однако вместо последней серии из восьми команд можно использовать всего одну:

```
SYSTEM 2
```

Эта команда вызывает *системную подпрограмму* с номером 2, находящуюся в постоянном запоминающем устройстве (ПЗУ) компьютера.

Так же, как и у реального компьютера, ПЗУ — это неизменяемая область памяти, то есть после запуска компьютера *ЛамПанель* изменить содержимое ПЗУ нельзя.

Для того чтобы увидеть все подпрограммы, которые записаны в ПЗУ, нужно щелкнуть по кнопке 

или выбрать пункт верхнего меню *Программа — Просмотр ПЗУ*. После этого появляется окно, в левой части которого перечислены все системные подпрограммы (с их номерами), а в правой части показывается текст выбранной подпрограммы и ее машинные коды:

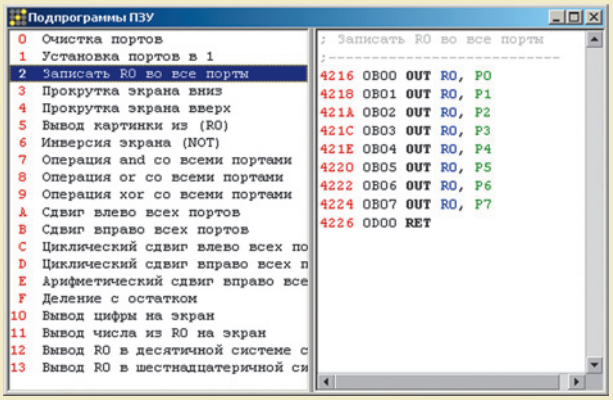


Рис. 6

В этом списке есть много полезных подпрограмм для работы с ламповой панелью, в том числе

0 — очистка экрана (погасить все лампочки);

1 — зажечь все лампочки на панели;

3–4 — прокрутка изображения вниз и вверх;

6–9 — логические операции;

A_{16}–E_{16} — сдвиги битов;

12_{16} — вывод числа, записанного в регистр **R0**, в десятичной системе счисления;

13_{16} — вывод числа, записанного в регистр **R0**, в шестнадцатеричной системе счисления.

Обратите внимание, что номер системной подпрограммы задается в шестнадцатеричной системе счисления.

Щелкнув мышкой на названии подпрограммы в левой части окна, мы увидим в правой части ее содержимое и комментарии, объясняющие, в каких регистрах должны быть расположены параметры подпрограммы и как она возвращает значения-результаты.

Рассмотрим еще одну задачу: вывести на экран рисунок, закодированный в виде шестнадцатеричных чисел (бит, равный единице, обозначает горящую лампочку). Для этого нужно сначала записать коды рисунка в память. Поскольку наш компьютер основан на архитектуре фон Неймана, в нем программа и данные находятся в одной области памяти. Поэтому данные можно записать с помощью специальной команды **DATA** после команды **STOP**:

```
...         ; здесь будет программа
STOP
M:          ; метка — начало блока данных
  DATA AAAA  ; код первой строчки
  DATA 5555
  DATA AAAA
  DATA 5555
  DATA AAAA
  DATA 5555
  DATA AAAA
  DATA 5555  ; код последней строчки
```

Для того чтобы вывести этот рисунок на экран, нужно записать его адрес в регистр **R0** и вызвать системную подпрограмму с номером 5:

```
MOV @M, R0  ; записать адрес метки M
            ; в регистр R0
SYSTEM 5 ; вывести на экран рисунок,
         ; адрес которого в R0
STOP
M:
  DATA AAAA; код первой строчки
  ...
```

ПЗУ учебного компьютера *ЛамПанель* хранится в текстовом файле, который загружается в память при запуске программы (так же, как и у учебной модели компьютера “E97” [3]). Поэтому такое ПЗУ можно изменять! Если требуется добавить новую подпрограмму в ПЗУ, нужно сохранить ее в специальном формате с помощью кнопки 

(или команды меню *Программа — Сохранить как ПЗУ*). Затем текст подпрограммы просто добавляется в конец файла **lampanel.rom**, в котором хранится ПЗУ.

Практикум

1. Составьте программу, после выполнения которой ламповая панель выглядит так:

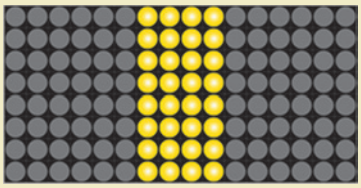


Рис. 7

2. Как вы думаете, что выведет приведенная выше программа, которая вызывает системную процедуру с номером 5? Проверьте ваш ответ с помощью программы *ЛамПанель*.

3. Закодируйте изображение домика и выведите его на экран.

4. Добавьте в предыдущую программу команды, которые сначала шифруют изображение, используя операцию “исключающее ИЛИ” с маской $BCA7_{16}$, а затем — восстанавливают исходное изображение. При изменении маски программа не должна изменяться. Изучите текст системной процедуры, которую вы используете.

5. Напишите программу, которая делает “бегущую строку” из рисунка-домика. *Подсказка*: используйте команды циклического сдвига.

6. Напишите программу, которая организует “обратный отсчет” от 100 до 0, а затем выводит рисунок с домиком и останавливается.

Подсказка: для вывода чисел используйте системную подпрограмму с номером 12_{16}.

Как вызываются подпрограммы

Вызов подпрограмм — это достаточно сложная операция для процессора. Действительно, при этом нужно:

- запомнить *адрес возврата* — адрес команды, на которую нужно перейти после завершения работы подпрограммы;
- передать параметры в *подпрограмму*;

• разместить в памяти локальные переменные подпрограммы;

• для подпрограммы-функции: передать в вызывающую программу результаты работы;

• обеспечить возврат из подпрограммы по правильному адресу.

В решении этих задач важнейшую роль играет *стек* (англ. *stack* — кипа, стопка) — специальная область оперативной памяти, в которой хранятся адреса возврата из подпрограмм и локальные переменные.

Стек в программе *ЛамПанель* размещается в оперативной памяти вместе с программой и данными. Оперативная память имеет размер 256 байт, адреса ячеек (байтов) изменяются от 0 до 255 = $\mathbf{FF}_{16}$. Стек находится в самом конце оперативной памяти и "растет вверх". Это значит, что первое записанное в стек 16-битное слово имеет адрес $\mathbf{FE}_{16}$ и занимает последние два байта памяти с адресами $\mathbf{FE}_{16}$ и $\mathbf{FF}_{16}$. Следующее записанное слово расположено по адресу $\mathbf{FC}_{16}$ и т.д. Как же компьютер разбирается, где начинается стек и сколько чисел туда записано?

В процессоре есть специальный регистр **SP** (от англ. *Stack Pointer* — указатель стека), в котором хранится адрес *вершины стека*, то есть последнего записанного в стек 16-битного значения. При запуске программы в регистр **SP** записывается значение 100_{16} = 256. Этот адрес находится за пределами оперативной памяти и говорит о том, что стек пуст. При добавлении в стек первого значения указатель стека уменьшается на 2 (до $\mathbf{FE}_{16}$), а потом по этому адресу записывается число.

Для того чтобы записать в стек значение из регистра общего назначения, используется команда **PUSH** (от англ. *push* — втолкнуть). Например, при выполнении этих команд в стек будет записано значение регистра **R0**, равное 1234_{16}:

```
MOV 1234,R0
PUSH R0
```

После выполнения команды **PUSH** значение регистра **SP** стало равно $\mathbf{FE}_{16}$, и теперь он указывает на последнее слово памяти, в котором записано число 1234_{16} — значение регистра **R0** (проверьте это с помощью программы).

Добавим в конец программы команду POP (англ. *pop* — вытолкнуть), которая "снимает" 16-битное значение с вершины стека и записывает его в регистр **R2**:

```
POP R2
```

После этого наблюдаем следующее:

• в регистре **R2** находится то же значение 1234_{16}, которое вначале было в **R0**;

• регистр **SP** содержит значение 100_{16}, которое говорит о том, что стек пуст;

• в последних двух байтах памяти осталось значение 1234_{16}, которое было записано в стек, но теперь оно уже не является частью стека, поскольку регистр **SP** изменен.

Как же используется стек при вызове подпрограмм? Напишем подпрограмму, которая возводит в квадрат значение регистра **R0**. Эта подпрограмма содержит всего одну команду умножения (умножить **R0** на **R0**, записать результат в **R0**):

```
MUL R0, R0
```

В начале подпрограммы нужно поставить метку (имя подпрограммы), а в конце — команду возврата **RET** (от англ. *return* — возврат), по которой процессор возвращается в точку вызова. Таким образом, вся подпрограмма, которую мы назовем **SQR**, выглядит так:

```
SQR:
  MUL R0, R0
RET
```

"Паспорт" этой подпрограммы такой:

Имя: SQR

Вход: число в регистре **R0**

Выход (результат): квадрат числа в регистре **R0**

Подпрограмма располагается ниже основной программы. Чтобы вызвать подпрограмму, используют команду **CALL** (от англ. *call* — вызвать), после которой записывают имя подпрограммы — метку, с которой она начинается, адрес точки входа. Вот вся программа вместе с подпрограммой:

```
MOV 12,R0 ; R0 := 12
CALL SQR ; вызвать подпрограмму SQR
STOP  ; конец основной программы
SQR:  ; точка входа подпрограммы
  MUL R0, R0 ; тело подпрограммы –
             ; R0 := R0 * R0
RET          ; возврат из подпрограммы
```

Остается один вопрос: как процессор определяет адрес возврата из подпрограммы, когда выполняется команда **RET**? Заметим, что в самой команде **RET** адрес не указан. Дело в том, что адрес перехода заранее определить нельзя (нельзя поставить команду безусловного перехода **JMP**), потому что подпрограмма может вызываться из разных мест программы, в том числе из других подпрограмм. Оказалось, что эта проблема просто решается с помощью стека:

• команда **CALL** записывает в стек адрес возврата из подпрограммы, то есть адрес команды, следующей за командой **CALL**; поскольку регистр **PC** (программный счетчик) всегда содержит адрес следующей команды, процессору достаточно просто "втолкнуть" содержимое регистра **PC** в стек;

• после этого в регистр **PC** записывается адрес подпрограммы (метки), указанной в команде **CALL**, и ей передается управление;

• команда **RET** снимает с вершины стека адрес возврата и записывает его в регистр **PC**, таким образом, управление передается следующей команде вызывающей программы.

Итак, при вызове подпрограммы адрес, содержащийся в регистре **PC**, временно запоминается в стеке, а по команде **RET** восстанавливается. Это обеспечивает возврат в то самое место программы, откуда вызывалась подпрограмма (на следующую команду после команды **CALL**).

Напишем более сложную подпрограмму, которая возводит **R0** в куб. Теперь для вычисления

придется задействовать еще один регистр, например, **R1**:

```
MOV R0,R1   ; R1 := R0
MUL R0,R0   ; R0 := R0 * R0
MUL R1,R0   ; R0 := R1 * R0
```

Все хорошо, но… в первой команде мы стерли значение регистра **R1**, которое было до вызова подпрограммы. Чтобы при вызове подпрограммы регистры не стирались, их желательно сохранять при входе в подпрограмму и восстанавливать перед самым выходом. В этом случае, вызывая подпрограмму, мы можем не беспокоиться, что она испортит нужные нам данные.

Где сохранять значения регистров? Самый простой выход — использовать стек: сохранять командой **PUSH** и восстанавливать командой **POP**. Таким образом, полный текст подпрограммы **CUBE** с сохранением регистра **R1** выглядит так:

```
CUBE:
  PUSH R1          ; сохраняем R1
  MOV R0,R1        ; R1 := R0
  MUL R0,R0        ; R0 := R0 * R0
  MUL R1,R0        ; R0 := R1 * R0
  POP R1           ; восстанавливаем R1
RET
```

В предыдущих примерах вы уже увидели, что параметры (дополнительные данные) могут передаваться в подпрограмму через регистры общего назначения **R0-R3**. Но этих регистров всего четыре, поэтому таким способом можно передать только четыре 16-битных числа. А что, если нужно передать, например, массив из 100 элементов? В этом случае снова на помощь приходит стек.

Перед вызовом подпрограммы в стек записываются все передаваемые параметры. Рассмотрим сначала "игрушечную" задачу — написать подпрограмму, которая возводит число в квадрат, причем это число передается через стек. Результат должен быть помещен в регистр **R0**.

Перед вызовом подпрограммы запишем в стек значение **R0**:

```
0000 MOV 12,R0    ; R0 := 12
0004 PUSH R0      ; запишем R0 в стек
0006 CALL SQR     ; вызов подпрограммы
000A STOP
000C SQR:
     ...    ; здесь будет подпрограмма
```

Если посмотреть на стек (в нижней части оперативной памяти), то после выполнения команды **PUSH R0** он выглядит так:

```
SP → 00FE  0012
```

Указатель стека **SP** содержит адрес $\mathbf{FE}_{16}$ и указывает на последнее 16-битное слово памяти. В нем записано число 12_{16} — это значение, передаваемое в подпрограмму.

Во время выполнения команды **CALL** в стек запишется адрес возврата из подпрограммы, то есть адрес $\mathbf{000A}_{16}$, по которому расположена команда **STOP**. Регистр **SP** будет указывать на ячейку памяти, содержащую этот адрес:

```
SP → 00FC  000A
     00FE  0012
```

Теперь займемся подпрограммой: как "достать" переданное значение? Сначала нужно скопировать содержимое указателя стека в какой-то регистр общего назначения, например, в **R0**:

```
MOV SP,R0
```

Теперь в **R0** находится адрес вершины стека, но там лежит адрес возврата. Чтобы получить адрес переданного параметра, нужно увеличить **R0** на 2:

```
ADD 2,R0
```

Теперь в **R0** записан адрес параметра, переданного подпрограмме. Можно взять значение по этому адресу и записать его в тот же регистр **R0**:

```
MOV (R0),R0
```

Теперь в **R0** уже получено переданное число и можно возвести его в квадрат. Вот полная подпрограмма:

```
SQR:
  MOV SP,R0
  ADD 2,R0
  MOV (R0),R0
  MUL R0,R0
RET
```

Остается один вопрос — кто же будет освобождать стек, удаляя из него параметры подпрограммы? Тут есть два варианта. Во-первых, этим может заниматься вызывающая программа — после вызова подпрограммы нужно использовать команду **POP**. Кроме того, это может делать и процедура — для этого нужно применить команду **RET** с параметром, обозначающим количество байт, которые нужно "сбросить" со стека. Например, в нашем случае можно применить команду

```
RET 2
```

которая освободит 2 байта (удалит один параметр). Отметим, что параметр команды **RET** — четное число, записанное в шестнадцатеричной системе счисления.

Таким образом, если параметров мало, их удобно передавать через регистры **R0-R3**. Кроме того, параметры можно передавать через стек. Если подпрограмма обрабатывает большой массив, лучше передать ей адрес этого массива, вместо того чтобы записывать его в стек целиком.

При изучении программирования *рекурсивные* подпрограммы (которые вызывают сами себя) выделяются в особый класс подпрограмм и изучаются отдельно. Однако для процессора такая подпрограмма ничем не отличается от "обычной", нерекурсивной. Отличие состоит только в том, что внутри рекурсивной подпрограммы есть вызов **CALL** по адресу той же самой подпрограммы.

Напишем рекурсивную подпрограмму для вычисления факториала числа, находящегося в регистре **R0**. Результат должен быть помещен в тот же регистр **R0**.

Факториал числа N вычисляется как произведение всех натуральных чисел от 1 до N: $N! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot$

$\cdot (N-1) \cdot N$. Вспомним рекурсивное определение факториала:

$$\begin{cases} 1! = 1 \\ N! = (N-1)! \cdot N, \quad N > 1 \end{cases}$$

Это значит, что факториал числа, большего единицы, определяется *рекурсивно* — через факториал другого числа, которое меньше данного на 1. Поэтому основная идея подпрограммы может быть записана на псевдокоде так:

```
R1 := R0
R0 := R0 - 1
; вычисляем факториал R0
; (вызов подпрограммы)
R0 := R1 * R0
```

Перевод на язык ассемблера дает:

```
FACT:
  MOV R0,R1    ; R1 := R0
  SUB 1,R0     ; R0 := R0 - 1
  CALL FACT    ; R0 := факториал R0
  MUL R1,R0    ; R0 := R1 * R0
FINISH:
  RET
```

Однако выполнение этой подпрограммы никогда не закончится, потому что при каждом обращении к подпрограмме она снова вызывает сама себя, и эти вызовы никогда не остановятся. Нужно добавить условие выхода: если **R0** = 1, нужно выйти из подпрограммы, то есть перейти на последнюю команду **RET** (перед ней должна быть метка):

```
FACT:
  CMP 1,R0      ; если R0 = 1, то выход
  JZ FINISH
  MOV R0,R1     ; R1 := R0
  SUB 1,R0      ; R0 := R0 - 1
  CALL FACT     ; R0 := факториал R0
  MUL R1,R0     ; R0 := R1 * R0
FINISH:
  RET
```

Остается еще один недостаток — подпрограмма изменяет значение регистра **R1**. Нужно при входе в подпрограмму сохранить его (в стеке), а перед выходом — восстановить. Как это сделать, мы рассматривали выше.

Практикум

1. Напишите и отладьте программу, которая меняет местами значения регистров **R2** и **R3** с помощью стека (не используя других регистров общего назначения).

2. Введите текст программы

```
MOV 12,R0
CALL SQR
STOP
SQR:
  MUL R0,R0
RET
```

Заполните таблицу, выполнив программу пошагово с помощью клавиши F7 (пошаговое выполнение с входом в подпрограммы):

Адрес	Команда	Регистры после ее выполнения		
		R0	**PC**	**SP**
		?	**0000**	**0100**
	MOV 12,R0			
	CALL SQR			
	MUL R0,R0			
	RET			
	STOP			

3. Напишите и отладьте программу с подпрограммой, которая вычисляет куб числа, записанного в регистр **R0**.

4. Напишите и отладьте программу с подпрограммой, которая и строит RGB-код цвета, 4-битные составляющие которого (R, G и B) записаны, соответственно, в регистры **R0**, **R1** и **R2**. Результат должен быть получен в регистре **R0**.

5. Выполните предыдущее задание при условии, что параметры передаются через стек, а значения регистров **R1** и **R2** не должны измениться.

6. Отладьте программу с рекурсивной подпрограммой, которая вычисляет факториал числа, записанного в регистр **R0**. При выполнении в пошаговом режиме (клавиша F7) наблюдайте, как изменяется регистр **SP** и содержимое стека.

7. Решите предыдущую задачу, применив подпрограмму без рекурсии.

Выводы

В статье представлена новая учебная модель компьютера *ЛамПанель*. Ее главное отличие от существующих — возможность управления ламповой панелью (матрицей, состоящей из лампочек) и современный интерфейс. В профильном курсе информатики с помощью программы *ЛамПанель* можно в наглядной форме изучать основные принципы работы современного компьютера.

Список использованных источников

1. *Еремин Е.А.* Debug и язык ассемблер // Информатика, 2007, № 5, с. 37–39; № 6, с. 37–38; № 7, с. 42–44.

2. *Еремин Е.А.* Учебные модели компьютера. Электронный ресурс (*http://educomp.runnet.ru/*).

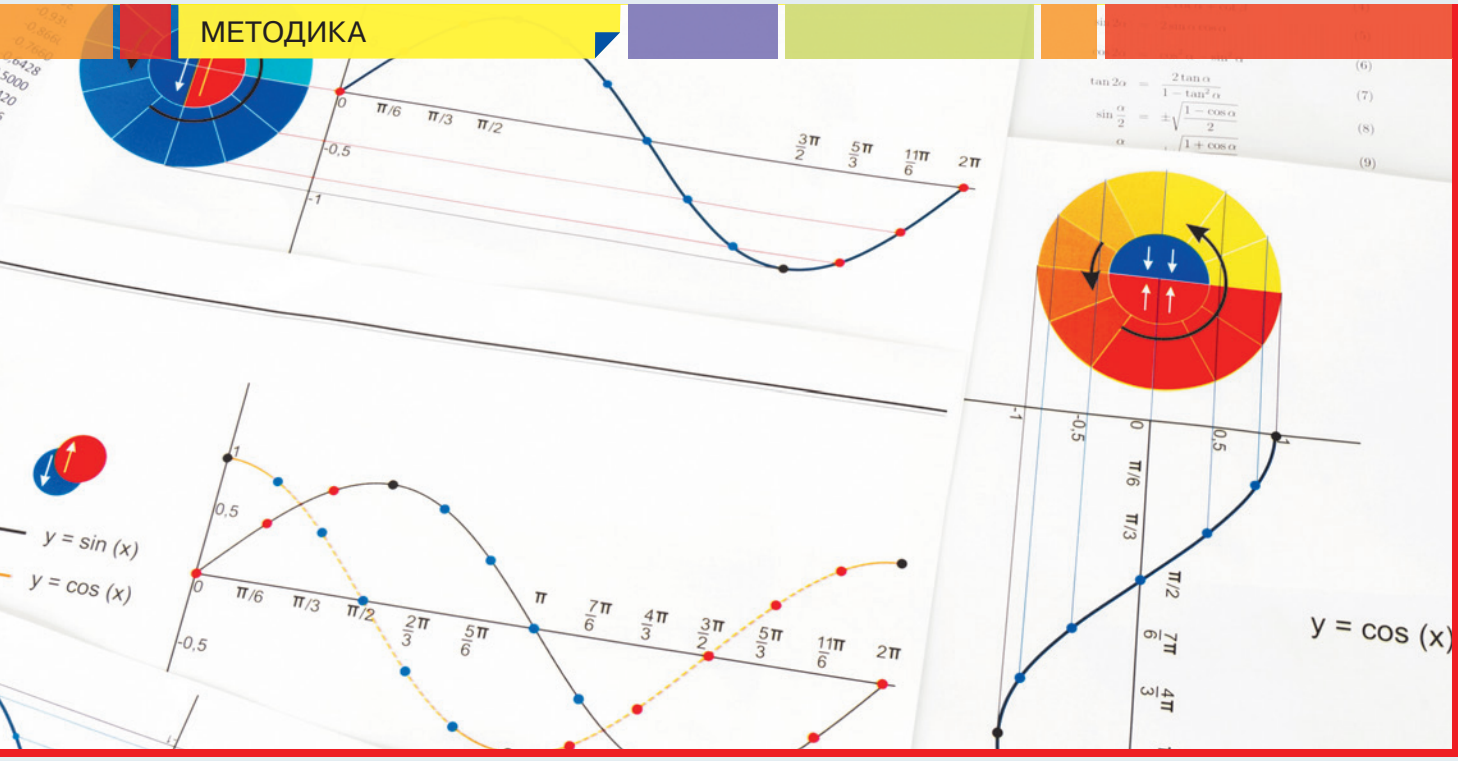
3. *Еремин Е.А.* Учебный компьютер “E97” для курса основ информатики и вычислительной техники // VIII Международная конференция “Информационные технологии в образовании”. Тезисы докладов. Направление B, с. 26–27. Москва, 1998 (*http://ito.edu.ru/1998-99/b/eremin.html*).

4. *Леонов А.Г.* ЭВМ-практикум // Информатика, 2011, № 11, с. 4–12.

5. EMU8086: 8086 microprocessor emulator integrated disassembler. Электронный ресурс (*http://ziplib.com/emu8086*).

6. *Поляков К.Ю., Шестаков А.П., Еремин Е.А.* Компьютерная арифметика // Информатика, № 1, 2011, с. 2–21.

Автор благодарит Е.А. Еремина за обсуждение этой статьи и полезные замечания.



Ряды

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

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

Приходилось ли вам когда-нибудь задумываться о том, что происходит “внутри компьютера”, когда вы используете (в своей программе на том или ином языке программирования, при вычислениях в Excel и т.д.) стандартные функции?

На первый взгляд здесь все предельно просто: достаточно написать имя стандартной функции, в скобках записать требуемые аргументы — константу, переменную или выражение, — и компьютер все сделает за вас: потому такие функции и называют “стандартными” (или “встроенными”). Вспомним, однако, что компьютер (точнее — его процессор) “умеет” выполнять только простейшие арифметические операции. Как же он тогда вычисляет значения стандартных функций?

Самое простое, что может прийти в голову, — это заложить в память ЭВМ пресловутые “таблицы Брадиса” (и аналогичные им таблицы значений для других функций), а потом осуществлять по ним поиск. Но это, как легко догадаться, не выход из ситуации: поиск по такой таблице требует времени, сама таблица займет изрядное место в памяти, к тому же таблицу реально составить только с определенным шагом, что станет причиной ограничения точности (а чем меньше мы сделаем шаг, тем больше будет объем занимаемой такой таблицей памяти и время поиска по ней). А самое главное — все равно такие таблицы нужно каким-то образом составить (вычислить), так что мы опять пришли к тому же, с чего начали разговор.

Впрочем, данная задача в математике уже давно решена. И ее решение заключается в *разложении функций в ряды*.

Если ввести в любой поисковой системе Интернета ключевое слово “ряд”, то “всплывающие подсказки”, которыми сегодня оснащены Яндекс, Google и многие другие “поисковики”, в первую очередь предложат в качестве вариантов для более конкретного поиска фразы “ряд Фурье” и “ряд Тейлора”. Их изучение, правда, относится к курсу высшей математики, так что более подробно с этими рядами сводят знакомство уже студенты вузов. Однако некоторые аспекты этой проблемы не только вполне доступны пониманию школьников, но и могут составить тему весьма интересного учебного проекта или даже исследования (не говоря о том, что некоторые из этих рядов, без упоминания

об их “громком” названии, предлагаются некоторыми авторами в качестве задач по программированию).

Первое знакомство

Говорят, что истинные англичане даже в критической ситуации не станут общаться друг с другом, если не были должным образом друг другу представлены согласно всем тонкостям этикета. Поэтому и мы начнем с подобного взаимопредставления. (“Алиса, это пудинг. Пудинг, это Алиса”, — помните это у Льюиса Кэрролла? ☺)

Ряд Тейлора — разложение функции в бесконечную сумму степенных функций. Ряд назван в честь английского математика Брука Тейлора, хотя ряд Тейлора был известен задолго до публикаций Тейлора — его использовал еще в XVII веке Грегори, а также Ньютон. Ряды Тейлора применяются при аппроксимации функции многочленами (“Википедия”: *http://ru.wikipedia.org/?oldid=43309997*).

Если расшифровать “понятным языком” это достаточно сложное определение, то ряд Тейлора можно рассматривать как возможность вычисления той или иной математической функции путем суммирования (и/или вычитания) неких “типовых” элементов, которые содержат в себе значение аргумента функции, возводимое во все большую степень. Такая сумма (и/или разность) элементов и называется многочленом, а ее сокращенная запись имеет вид:

$$\sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!}(x-a)^k,$$

где $f(x)$ — искомая функция, k — возрастающий от 0 до (в общем случае) бесконечности коэффициент (его можно считать заодно и порядковым номером членов ряда), а a — некоторая точка, вблизи (или, как говорят математики, “в окрестности”) которой функция должна обладать определенным свойством: быть многократно дифференцируемой (т.е. от нее можно вычислить производную, от этой производной — новую производную и т.д. до бесконечности). Символ же “Σ” означает сумму; чтобы “развернуть” такую запись в ряд (в многочлен), достаточно начать записывать формулу, стоящую после этого знака суммы, каждый раз подставляя в нее очередное значение индекса суммирования (в данном случае k).

Если в ряде Тейлора принять значение a равным нулю, то мы получаем частный случай такого ряда, который называют *рядом Маклорена* (в честь выдающегося английского математика Колина Маклорена).

Не вдаваясь более в подробности, приведем для справки несколько примеров записи (т.е. “разложения”) стандартных функций в виде рядов Маклорена:

- экспонента:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots = \sum_{n=0}^{\infty} \frac{x^n}{n!},\ x \in \mathrm{R};$$

- натуральный логарифм:

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \ldots = \sum_{n=0}^{\infty} \frac{(-1)^n x^{n+1}}{n+1} = \sum_{n=1}^{\infty} \frac{(-1)^{n+1} x^n}{n},$$

для всех $|x| < 1$;

- квадратный корень:

$$\sqrt{1+x} = 1 + \frac{x}{2} - \frac{x^2}{8} + \frac{x^3}{16} - \ldots = \sum_{n=0}^{\infty} \frac{(-1)^n (2n)!}{(1-2n)n!^2 4^n} x^n,$$

для всех $|x| < 1$;

- синус:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1},\ x \in \mathrm{R};$$

- косинус:

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \ldots = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n},\ x \in \mathrm{R}.$$

Таким образом, эти функции (и целый ряд других функций) можно, правда, с определенной степенью точности, вычислять, используя обычные арифметические операции (сложение, вычитание, умножение и деление; возведение в степень реализуется как последовательность умножений). Что, собственно, нам и требуется для реализации вычисления значений этих функций на компьютере!

Ряд Фурье — представление произвольной функции f с периодом τ в виде ряда:

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{+\infty} A_k \cos\left(2\pi \frac{k}{\tau} x + \theta_k\right)$$

где A_k — амплитуда k-го гармонического колебания, — круговая частота гармонического колебания, а θ_k — начальная фаза k-го колебания. (“Википедия”: *http://ru.wikipedia.org/?oldid=42761297*.)

Если ряды Тейлора (Маклорена) позволяли нам свести вычисление стандартных функций к четырем действиям арифметики, то ряд Фурье дает возможность уже через стандартные тригонометрические функции (в данном случае — косинус) вычислять *любые* периодические функции. Или, говоря другими словами, любую периодическую функцию (т.е. любые колебания) можно представить как совокупность (сумму) гармонических колебаний.

Ряды — в программировании

“Загрузив” в себя некоторый объем теоретических знаний (необходимых хотя бы для того,

чтобы понимать, о чем идет речь), перейдем к делам более практическим. Так, вычисление рядов Маклорена, соответствующих некоторым функциям, вполне можно рассматривать как весьма благодатную тему задач по программированию циклов, позволяющую не только освоить навыки реализации этой алгоритмической конструкции, но и затронуть целый ряд вопросов оптимизации алгоритма.

Задача 1. Вычисление экспоненты

Итак, значение функции e^x можно вычислить как сумму членов ряда:

$$1+\frac{x}{1!}+\frac{x^2}{2!}+\frac{x^3}{3!}+\ldots+\frac{x^n}{n!}$$

Напишем программу, реализующую такие вычисления для введенных с клавиатуры значений x и n.

Первая идея, которая приходит на ум, — построить цикл **for** для i от 1 до n, в нем каждый раз вычислять значение очередного слагаемого (написав две пользовательские функции для возведения x в степень i и для вычисления факториала числа i). Но, как нетрудно понять, это приводит к значительным затратам вычислительных ресурсов: обе эти подпрограммы должны содержать в себе циклы от 1 до i, а это накладно по времени.

Поэтому лучше построить алгоритм иначе: вычислять очередное слагаемое через предыдущее, запоминая его в отдельной переменной, а при очередном проходе цикла это предыдущее значение слагаемого (члена ряда) умножать на x и делить на значение i. Очевидно, что такие действия как раз и будут в очередном слагаемом увеличивать на 1 показатель степени в числителе, а в знаменателе "довычислять" требуемый факториал, поскольку $n! = (n-1)! \cdot n$.

С учетом сказанного, программа принимает вид (язык Паскаль, версия PascalABC.Net):

```
program for22;
var X,E,Up: real;
    N,i,Dn: integer;
begin
  write('X= '); readln(X);
  write('N= '); readln(N);

  E := 1;
  Up := 1; {числитель}
  Dn := 1; {знаменатель}
  for i := 1 to N do
    begin
      Up := Up * X;
      Dn := Dn * i;
      E := E + Up / Dn;
    end;

  writeln('Результат: ',E);
end.
```

Чтобы проверить правильность ее работы, можно задавать $x = 1$ и постепенно возрастающие значения n, — выдаваемый результат должен все больше приближаться к значению основания натуральных логарифмов $e = 2.7182818284590452\ldots$

При желании, для большей наглядности, можно немного модифицировать эту программу, добавив в нее внешний цикл **for** по n от 1 до заданного n_0 и "заставляя" компьютер выводить значение суммы ряда для каждого такого значения n:

```
program for22_;
var X,E,Up: real;
    N,i,Dn,N0: integer;
begin
  write('X= '); readln(X);
  write('N0= '); readln(N0);

  for N := 1 to N0 do
    begin
      E := 1;
      Up := 1; {числитель}
      Dn := 1; {знаменатель}
      for i := 1 to N do
        begin
          Up := Up * X;
          Dn := Dn * i;
          E := E + Up / Dn;
        end;
      writeln('N= ',N,' Результат: ',E);
    end;
end.
```

Запустив эту программу, можно увидеть, как получаемое значение суммы ряда постепенно приближается (как говорят математики, "стремится", "сходится") к значению 2.7182818284590452…:

X = 1
N0 = 10
N = 1 Результат: 2
N = 2 Результат: 2.5
N = 3 Результат: 2.66666666666667
N = 4 Результат: 2.70833333333333
N = 5 Результат: 2.71666666666667
N = 6 Результат: 2.71805555555556
N = 7 Результат: 2.71825396825397
N = 8 Результат: 2.71827876984127
N = 9 Результат: 2.71828152557319
N = 10 Результат: 2.71828180114638

Задача 2. Вычисление косинуса

Значение функции $\cos(x)$ вычисляется как сумма членов ряда:

$$1-\frac{x^2}{2!}+\frac{x^4}{4!}-\ldots\frac{(-1)^n x^{2n}}{(2n)!}$$

Напишем программу, реализующую такие вычисления для введенных с клавиатуры значений x и n.

После решения предыдущей задачи все кажется достаточно простым. Можно, опять-таки, запом-

нить в отдельной переменной предыдущее значение слагаемого, а на очередном проходе цикла умножать это значение на этот раз на x^2 и делить на произведение $(2{\cdot}i) \cdot (2{\cdot}i - 1)$, чтобы получить, например, из 2! значение 4! И еще, чтобы обеспечить периодическое изменение знака слагаемого, записать в отдельную переменную значение 1, а на каждом очередном шаге цикла прежде всего умножать эту переменную на –1 (меняя ее знак), а в записи суммы домножать на текущее значение этой "знаковой" переменной вычисленное значение очередного слагаемого.

Однако, написав такую программу, нетрудно убедиться, что при задании все возрастающего (начиная с 1) значения n очень быстро наступает момент, когда программа "идет вразнос": стремление результата к "теоретическому" значению (например, к нулю при задании $x = 1{,}57 \approx \pi/2$) явно нарушается. Почему?

Причину понять нетрудно. В отличие от предыдущей программы мы производим возведения в степень, а особенно — вычисление факториала в знаменателе "в два раза быстрее". Это приводит к переполнению разрядной сетки. Отсюда — и ошибка вычисления.

Чтобы увеличить интервал значений n, для которых программа сохраняет работоспособность, можно применить такую хитрость: вместо умножения сразу на x^2, а затем деления сразу на $(2{\cdot}i) \cdot (2{\cdot}i - 1)$ делать эти операции "постепенно", по очереди: сперва умножить ранее запомненное значение слагаемого (члена ряда) на x и разделить на $(2{\cdot}i - 1)$, а потом еще раз умножить на x и разделить на $(2{\cdot}i)$.

С учетом этого программа может иметь вид:

```
program for24;
var X,E,Drob: real;
    N,i,Zn: longint;
begin
  write('X= '); readln(X);
  write('N= '); readln(N);

  E := 1;
  Drob := 1; {дробь}
  Zn := 1; {текущий знак}
  for i := 1 to N do
    begin
      Zn := -Zn; {смена знака слагаемого}
      Drob := Drob * X / (2 * i - 1);
      Drob := Drob * X / (2 * i);
      E := E + Zn * Drob;
    end;

  writeln('Результат: ',E);
end.
```

Для проверки можно задавать $x = 1{,}57 \approx \pi/2$ и возрастающие значения n, либо, как и в предыдущей задаче, модифицировать программу, чтобы она выводила значения суммы ряда для каждого из значений n на диапазоне от 1 до заданного n_0:

```
program for24_;
var X,E,Drob: real;
    N,i,Zn,N0: longint;
begin
  write('X= '); readln(X);
  write('N0= '); readln(N0);
  for N := 1 to N0 do
    begin
      E := 1;
      Drob := 1; {дробь}
      Zn := 1; {текущий знак}
      for i := 1 to N do
        begin
          {смена знака слагаемого}
          Zn := -Zn;
          Drob := Drob * X / (2 * i - 1);
          Drob := Drob * X / (2 * i);
          E := E + Zn * Drob;
        end;
      writeln('N= ',N,' Результат: ',E,
      ' delta= ',abs(cos(x)-E):13:10);
    end;
end.
```

Однако здесь из-за периодической смены знака текущий результат приближается к искомому теоретическому значению "бросками" то в меньшую, то в большую сторону, поэтому дополнительно имеет смысл выводить значение *delta*, равное модулю разности теоретического значения $\cos(x)$ и получаемого значения суммы ряда, чтобы следить за уменьшением этого значения *delta*:

```
X = 1.57
N0 = 10
N = 1    Результат: –0.23245                 delta = 0.2332463267
N = 2    Результат: 0.0207055004166667       delta = 0.0199091737
N = 3    Результат: –9.45993492347079E–05    delta = 0.0008909261
N = 4    Результат: 0.00082093932778262      delta = 0.0000246126
N = 5    Результат: 0.000795864757949509     delta = 0.0000004620
N = 6    Результат: 0.00079633298754937      delta = 0.0000000063
N = 7    Результат: 0.00079632664612552      delta = 0.0000000001
N = 8    Результат: 0.000796326711254585     delta = 0.0000000000
N = 9    Результат: 0.000796326710729955     delta = 0.0000000000
N = 10   Результат: 0.000796326710733358     delta = 0.0000000000
```

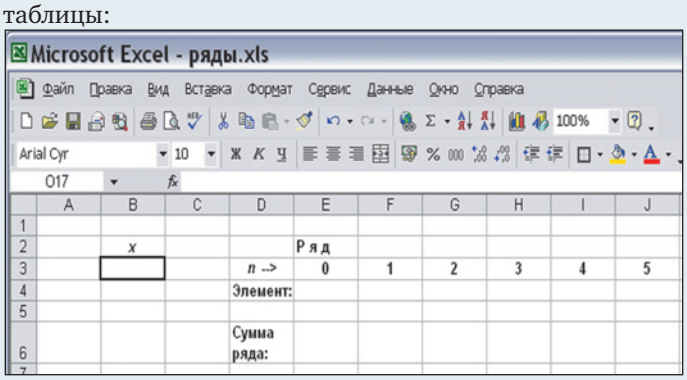
Ряды — в Excel

Хотя при выполнении программы на Паскале по изменению вычисленных значений ряда при различном количестве его членов можно проследить “в числах”, все же это недостаточно наглядно. Попробуем представить это изменение графически. Для этого нам потребуется создать расчетную модель уже в программе Excel.

Задача 1. Вычисление экспоненты

Прежде всего отметим, что в Excel (в отличие от программирования) нет возможности создания цикла (использование языка программирования Visual Basic for Application мы в данном случае не рассматриваем). Зато в Excel мы можем создать таблицу, в которой последовательно вычисляются элементы (члены) ряда, причем каждый последующий (благодаря использованию относительной ссылки) вычисляется через предыдущий.

1. Создаем заготовку таблицы:



Здесь ячейка **B3** отведена для ввода значений x, строка **4,** начиная со столбца **E**, будет хранить вычисляемые друг за другом значения элементов ряда, а строка **6** (также начиная со столбца **E**) предназначается для значений сумм ряда от его нулевого элемента до текущего.

2. Значение нулевого элемента в данном случае определено константой 1. Вводим ее в ячейку **E4**.

3. Создаем в ячейке **F4** формулу для вычисления очередного элемента ряда через предыдущий. Анализируя соответствующую запись ряда Маклорена, мы видим, что очередной элемент ряда можно вычислить, умножая предыдущий элемент на x и деля полученное число на порядковый номер вычисляемого элемента. Тогда за счет умножения на x в текущем элементе будет “накапливаться” соответствующая степень x^n (ведь в предыдущем элементе x был в степени $(n-1)$), а при делении будет “накапливаться” факториал в знаменателе (так как в предыдущем элементе в знаменателе был факториал $(n-1)!$, для получения значения $n!$ достаточно предыдущее значение домножить на n).

Тогда, учитывая все сказанное, в ячейку **F4** надо записать формулу: **=E4*$B$3/F3**
(обратите внимание: ссылка на значение x сделана абсолютной).

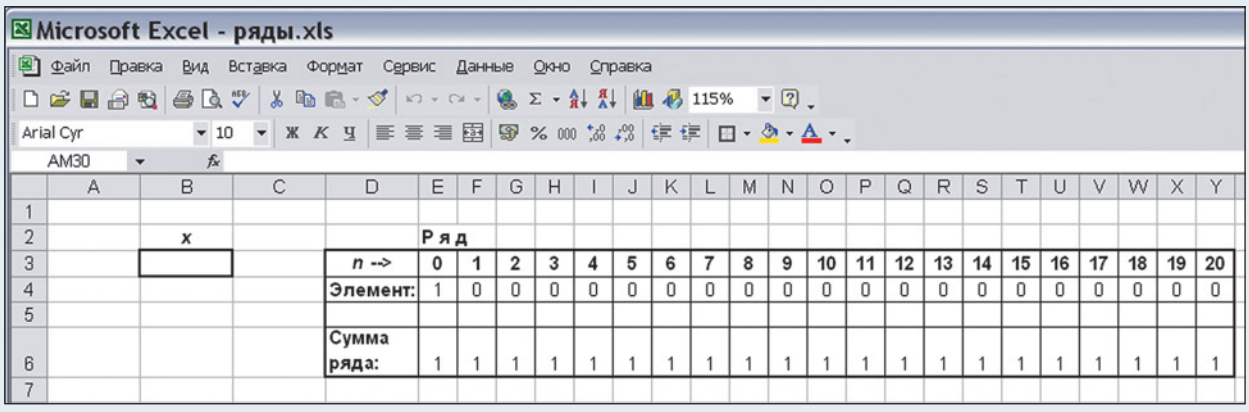
Эту формулу путем автозаполнения копируем во все ячейки правее **F4** (не смущаясь, что везде получаются нули, — ведь ячейку со значением x мы еще не заполняли).

4. Подсчитываем суммы. В ячейку **E6** чисто формально (извините за каламбур) запишем формулу: **=E4**

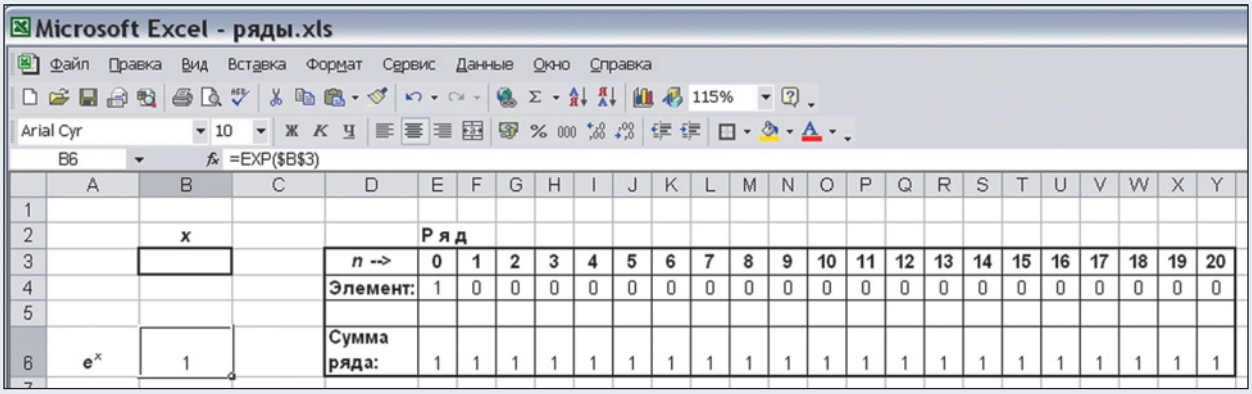
В ячейку **F6** нужно уже записать формулу с функцией суммирования: **=СУММ(E4:F4)**

А вот далее нас ждет небольшая “ручная работа”: после копирования (автозаполнением) формулы из ячейки **F6** во все ячейки правее нее надо вручную “пройтись” по всем полученным формулам и откорректировать в них диапазон суммирования — так, чтобы он всегда начинался с ячейки **E4**.

В итоге получаем (после дополнительного оформления и изменения ширины столбцов) таблицу следующего вида:



5. Добавим в отдельном поле (для сравнения) формулу вычисления значения функции e^x при помощи стандартной функции: **=EXP($B$3)**



	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
1																									
2		x			Ряд																				
3				n -->	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
4				Элемент:	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5																									
6	e^x	1		Сумма ряда:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

6. Вводим значение x, равное 1. Проверяем результат:

x	
1	
e^x	2,718282

Ряд					
n -->	0	1	2	3	4
Элемент:	1	1	0,5	0,1666667	0,0416667
Сумма ряда:	1	2	2,5	2,6666667	2,7083333

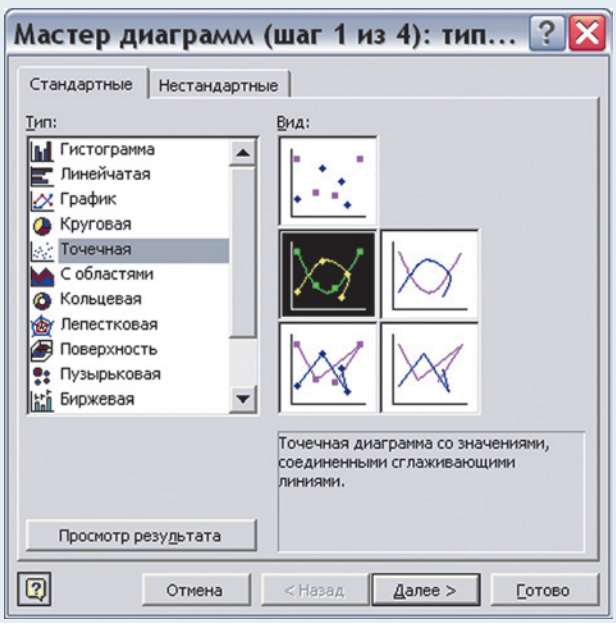
5	6	7	8	9	10	11	12
0,0083333	0,0013889	0,0001984	2,48E-05	2,756E-06	2,756E-07	2,505E-08	2,088E-09
2,7166667	2,7180556	2,718254	2,7182788	2,7182815	2,7182818	2,7182818	2,7182818

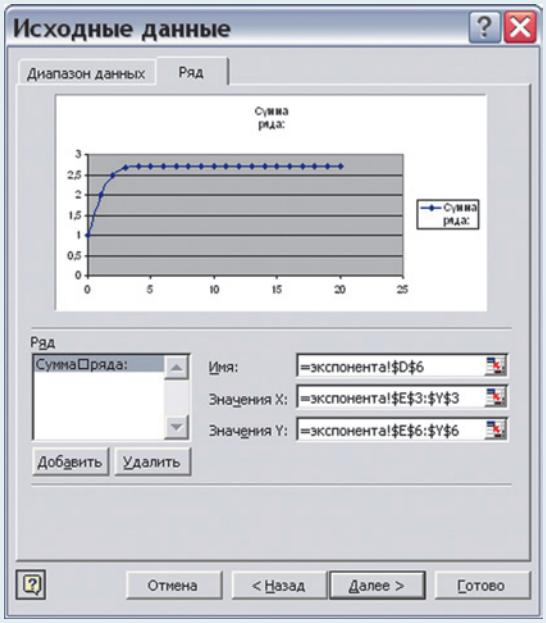
13	14	15	16	17	18	19	20
1,606E-10	1,147E-11	7,647E-13	4,779E-14	2,811E-15	1,562E-16	8,221E-18	4,11E-19
2,7182818	2,7182818	2,7182818	2,7182818	2,7182818	2,7182818	2,7182818	2,7182818

Действительно, мы убедились, что с ростом значения n (количества входящих в сумму элементов ряда) вычисленное значение ряда все больше приближается к “точному” (вернее, принятому нами за “теоретическое”) значению e^x. И уже семи слагаемых достаточно для получения приемлемой точности.

7. Представим полученную зависимость графически — в форме диаграммы **Точечная** (со значениями, соединенными сглаживающими линиями), построенной для строки таблицы “Сумма ряда”.

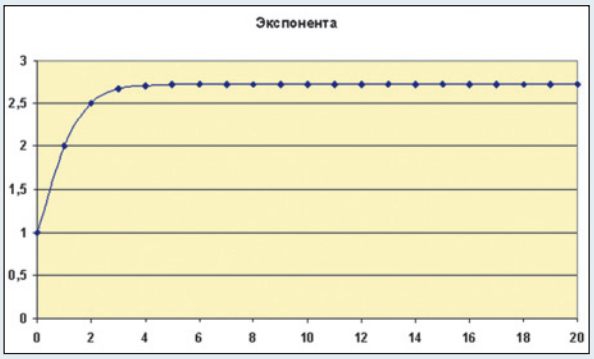
При этом надо на втором шаге Мастера диаграмм на вкладке **Ряд** для поля **Значения X** выбрать диапазон таблицы, содержащий значения n (от 0 до 20).



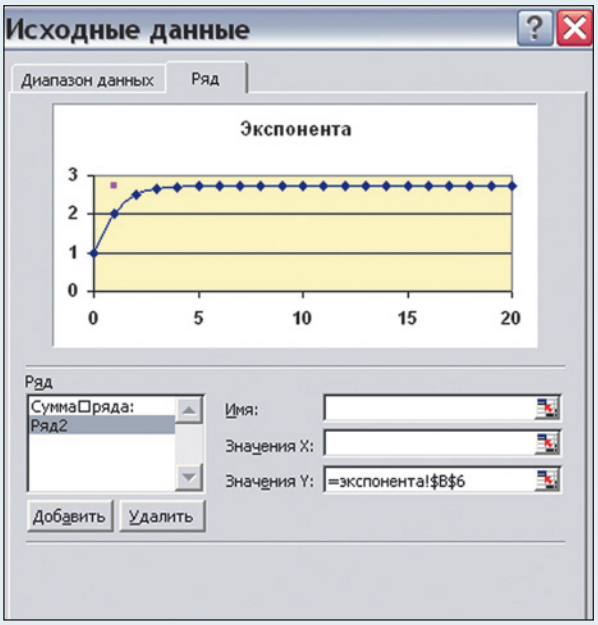


Остальные настройки диаграммы выполняем по желанию.

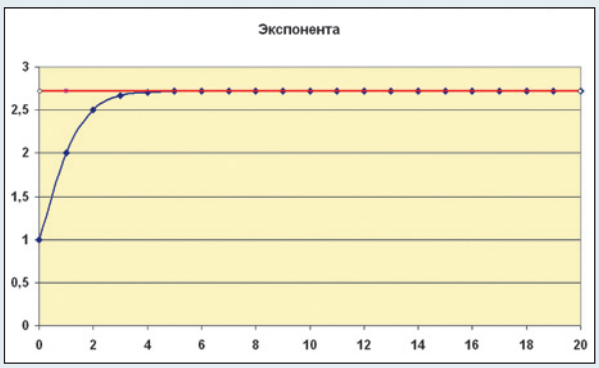
В результате получаем график:



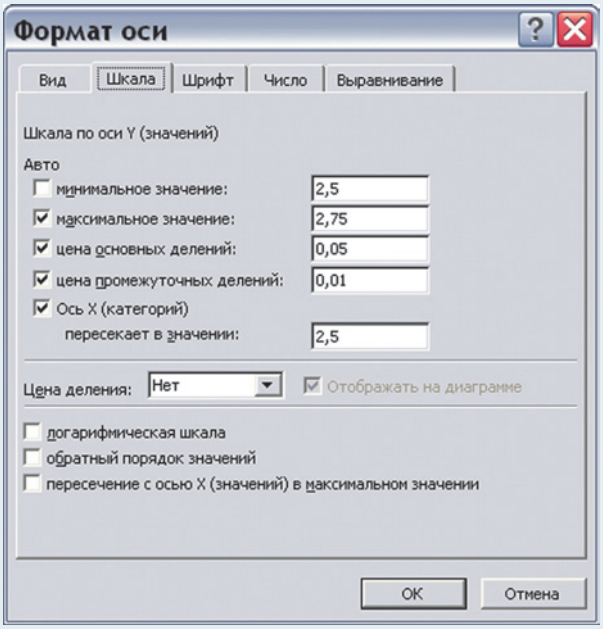
8. Дополним построенный график “опорной линией”, соответствующей “точному” значению функции e^x. Для этого, щелкнув на построенной диаграмме правой кнопкой мыши, выберем в контекстном меню пункт **Исходные данные**, а затем в раскрывшемся одноименном окне на вкладке **Ряд** нажимаем кнопку **Добавить**, помещаем курсор в поле **Значения Y** и в исходной таблице щелкаем мышью на ячейке **B6**, в которой содержится значение e^x:



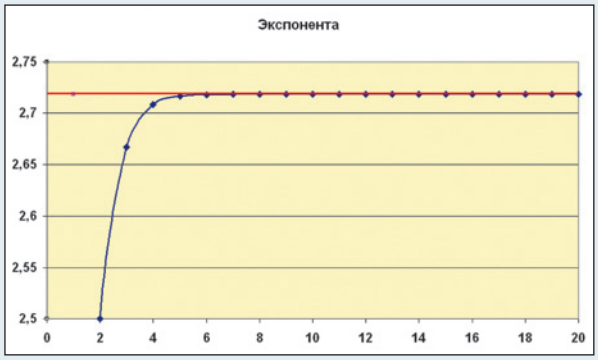
На графике появится точка другого цвета. Обратившись к панели **Рисование**, проводим через середину полученной точки горизонтальную линию:



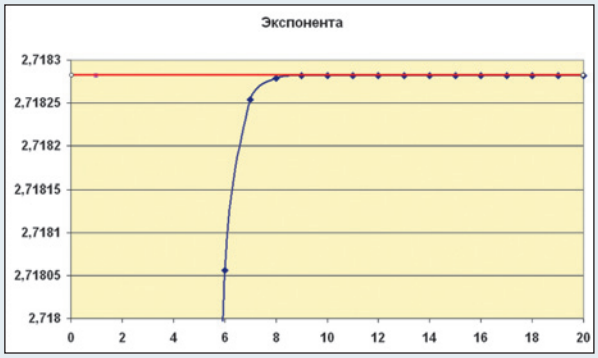
9. Проведем небольшое исследование: изменим диапазон шкалы *Y* на [2,5; 3] (пункт контекстного меню **Формат оси**):



Получаем график (“опорную линию” нужно будет немного подвинуть вручную):



Можно продолжить исследование, еще больше сужая диапазон шкалы по оси *Y*, например:

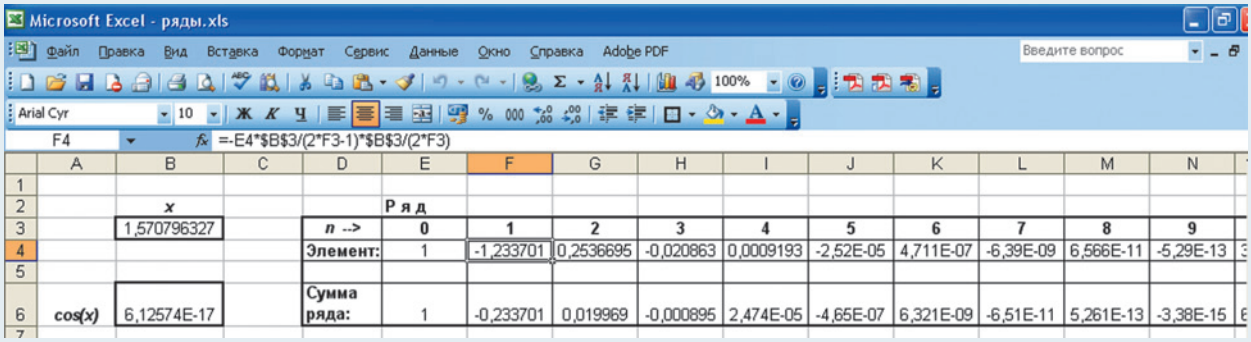


Задача 2. Вычисление косинуса

Эта задача решается аналогично предыдущей, — только формула для вычисления очередного члена ряда по значению предыдущего будет более сложной. Для ячейки F4 эта формула имеет вид:

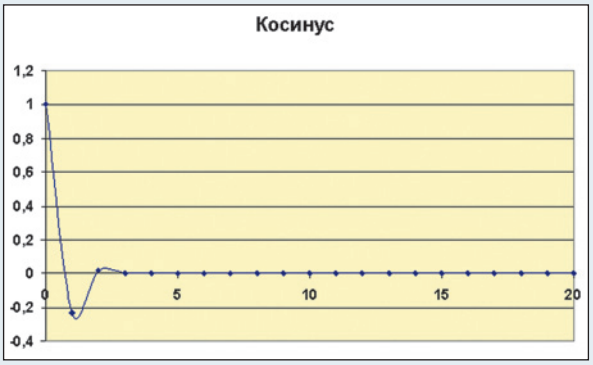
=-E4*$B$3/(2*F3-1)*$B$3/(2*F3)

Здесь мы тоже используем ту же хитрость с попеременным умножением/делением, а знак минуса перед ссылкой на ячейку с предыдущим значением элемента обеспечивает попеременную смену знака слагаемых. Далее эту функцию надо распространить вправо путем автозаполнения. Все остальное заполнение и оформление ячеек остается таким же, как в предыдущей задаче, — только в "контрольной" ячейке **B6** записано вычисление косинуса от значения исходной ячейки **B3** (x), а в качестве этого исходного значения лучше всего вводить не просто константу, а формулу, включающую вызов стандартной функции **ПИ()** (в данном случае — **=ПИ()/2**).

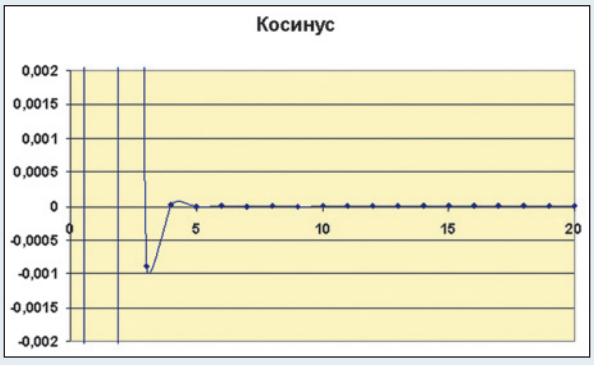


	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1														
2		*x*			Ряд									
3		1,570796327		*n* -->	0	1	2	3	4	5	6	7	8	9
4				Элемент:	1	-1,233701	0,2536695	-0,020863	0,0009193	-2,52E-05	4,711E-07	-6,39E-09	6,566E-11	-5,29E-13
5														
6	*cos(x)*	6,12574E-17		Сумма ряда:	1	-0,233701	0,019969	-0,000895	2,474E-05	-4,65E-07	6,321E-09	-6,51E-11	5,261E-13	-3,38E-15

Построив диаграмму по значениям ячеек строки "Сумма ряда", можно (в отличие от результатов работы соответствующей программы на Паскале) наглядно увидеть, как получаемое значение суммы ряда с ростом n приближается к "теоретическому" значению $\cos(\frac{\pi}{2}) = 0$: не плавно, с одной только стороны ("асимптотически"), а волнообразно, с затухающими колебаниями относительно этого "теоретического" значения.



Причем сходимость ряда очень быстрая — буквально на первых же шагах, а дальше колебания при заданном масштабе просмотра уже практически незаметны. И та же картина сохраняется, если "увеличить масштаб", ограничивая шкалу по оси Y диапазоном [–0,2; 0,2], затем — диапазоном [–0,05; 0,05], диапазоном [–0,01; 0,01] и т.д.



Задание для самостоятельной работы

Попробуйте "поиграть" с другими рядами Тейлора – Маклорена. Напишите соответствующие программы на Паскале (или другом языке программирования) и создайте модель для табличных вычислений в Excel. Исследуйте, как и насколько быстро сходится тот или иной ряд.

Заметки о разметке

И.А. Сукин,
г. Переславль-Залесский

ИСТОРИЯ РАЗМЕТКИ

Потребность в разметке документов самых разных типов и разделении их на логические части возникла перед человеком очень давно, с тех самых пор, как только эти самые документы вошли в нашу жизнь. Особую важность разметка приобрела с изобретением и распространением технологии книгопечатания. Разумеется, с появлением вычислительной техники, глобальной оцифровкой почти всех видов человеческой деятельности и разработкой надежных методов математического моделирования таких чувственных образов, как зрительные и слуховые, необходимость структурирования информации возросла в разы, а спектр обрабатываемых документов серьезно расширился.

Первые поколения электронных вычислительных машин были слишком медленными для серьезной обработки текстовых данных, а соответствующие методы такой обработки в те времена были еще недостаточно проработаны, поэтому первые попытки изобрести машинный язык разметки документов стали предприниматься в начале 70-х годов, что стало мощным толчком к развитию компьютерной типографии.

Одним из первых подобную идею предложил Вильям Танниклифф [1], разработав стандарт, который он называл “generic coding” (универсальное кодирование, или GenCode). Практически параллельно с ним эту идею выдвинул Чарльз Голдфарб из IBM, ставший после основоположником разработки языка GML, лежащего в основе большинства современных языков разметки, являющихся привычными большинству из нас: HTML, XML и т.д. К концу семидесятых годов в результате слияния идей GenCode и GML появился универсальный язык (или скорее даже метаязык) SGML (***S**tandard **G**eneralized **M**arkup **L**anguage*) — обобщенный первый язык такого рода, стандартизованный организацией ISO [2].

Другим пионером языков разметки был известный всем людям, тем или иным образом связанным с информационными технологиями, автор хрестоматийного труда “Искусство программирования” Дональд Кнут. Во время работы над упомянутой монографией Дональд обнаружил, что существовавшие в то время средства компьютерной типографии обладали крайне низким качеством. В то же время он интересо-

вался историей книгопечатания и основами промышленной типографики, которая была распространена в докомпьютерную эпоху. Это натолкнуло его на мысль о создании собственной издательской системы, которая сочетала бы высокое качество печатной продукции, как в эпоху, когда печать производилась вручную, и высокую гибкость в обработке текстов, появившуюся с развитием компьютерной техники. Немаловажной была возможность отображения математических формул и выражений для издания математических книг профессионального качества. Разумеется, для всего этого был нужен какой-то язык разметки. Так появился TEX [3], который с тех пор успел стать стандартом де-факто для публикаций во многих областях науки.

Еще одной ключевой вехой в развитии языков разметки стал Scribe, появившийся в начале 80-х годов прошлого века. Несмотря на то, что сейчас об этом языке знают лишь энтузиасты, он привнес серьезное нововведение в идею разметки — концепцию "стилей", что позволило четко отделить описание логической и графической структуры документа от собственно значащей информации. Scribe серьезно повлиял на GML и TEX, которые при этом эволюционировали в SGML (а далее и в HTML) и LATEX.

С развитием интернет-технологий и Всемирной паутины в 90-х годах из SGML выделились такие специализированные, но знакомые сейчас даже людям "непосвященным" языки разметки, как HTML и XML, к которым потом под влиянием концепции стилей добавились такие языки, как CSS, DTD и XML Schema.

В целом можно сказать, что, подобно языкам программирования, языки разметки развивались от машиночитаемых к человекочитаемым, поэтому третье тысячелетие ознаменовалось распространением языка YAML и его подмножества JSON, которые, оставаясь простыми для чтения и редактирования человеком, одновременно решали задачу не только разметки, но и хранения, сериализации самых разнообразных данных. Тем не менее наиболее популярным языком разметки в наше время остается SGML, а точнее, его подмножество — XML и другие производные языки.

ЧТО ТАКОЕ ЯЗЫК РАЗМЕТКИ?

Рассуждая об истории языков разметки в предыдущем разделе, я так ни разу четко не определил, что же это такое. В настоящее время само понятие разметки настолько прочно вошло в жизнь человека, что почти каждый, имеющий компьютер, так или иначе может без труда догадаться о его значении. Язык разметки в широком смысле — это компьютерный язык, позволяющий каким-либо образом структурировать данные, описывающий информацию об их строении. Данные при этом могут быть практически произвольными: все виды текстовой информации, графика, музыка, данные о строении химических молекул, географические данные, программный код и многое другое. В общем случае язык разметки не является языком программирования, хотя, как и последний, представляет отображение какой-либо нотации. При этом большинство современных языков разметки могут содержать элементы языков программирования и макрообработки.

Типы разметки

Принято выделять два основных типа разметки или подхода к дизайну документов: логическая разметка и визуальная. Визуальная разметка проста для неспециалиста, она позволяет явно задавать свойства отображения объекта, например: "эта строка должна быть написана курсивом" или "эта строка должна отображаться жирным шрифтом 16-го размера". Логическая же разметка является следствием из приведенного ранее определения языка разметки — она позволяет задавать логическую структуру документа, например, "вот эта строка — заголовок первого уровня", а решать вопрос о том, как данные будут отображаться в реальности, будет специальная программа-обработчик. Использование логической разметки позволяет избежать путаницы при изменении документа, его частей или свойств. Важным применением визуальной разметки является создание плакатов, постеров, презентаций, когда желательно, чтобы элементы выглядели в точности так, как это задумывалось автором, а их логический смысл не слишком важен. Несмотря на то, что подавляющее большинство языков разметки ориентированы на логическую разметку, они также содержат средства визуального редактирования информации.

ОБЩИЕ ПОДХОДЫ К РАЗМЕТКЕ

Рассмотрим основные подходы к использованию языков разметки в практических целях. Данные, размеченные с помощью выбранного языка, обычно хранятся в текстовом виде, однако при этом могут содержать некоторые двоичные вставки, например, необработанные изображения.

Для каждого языка существует специальная программа-обработчик, переводящая размеченные данные в некоторое внешнее представление, а начальный этап процесса перевода (как и в случае любых других языков) называется "разбором" (англ. *parsing*). В особых случаях может существовать еще программа-препроцессор, которая переводит информацию, размеченную с помощью одного языка, на другой язык разметки. Подобное бывает необходимо, например, для раскрытия макроподстановок, для генерации документов "на лету" из шаблонов и, не в последнюю очередь, для представления документов в более удобном для какой-либо цели (например, для непосредственного вывода на принтер) виде. Нередки случаи конвейерной обработки

размеченных данных, когда они последовательно обрабатываются несколькими программами.

Упомянутое ранее “внешнее представление” может быть самым разным. Я уже сказал о случае перевода с одного языка разметки на другой, однако внешним представлением может быть изображение на экране компьютера, звук, обработанный текст, напечатанная принтером информация, компьютерная программа, трехмерная сцена, в конце концов — внутреннее представление данных в некоторой сторонней программе. При разборе конкретных языков и их подмножеств я буду уделять отдельное внимание тому, что представляет собой их “внешнее представление”.

Как выглядят размеченные данные?

Вопрос о том, как же все-таки выглядит информация, размеченная с помощью некоторого языка разметки, является сам собой разумеющимся. Обычно подразумевается, что стандарт языка определяет некоторый набор управляющих конструкций, или отметок-“тэгов”. В случае общих языков определяется только набор правил, по которым управляющие конструкции, или тэги, будут записаны. Например, простейшим и знакомым многим правилом из языка SGML (XML) является следующее: тэги записываются в угловых скобках, каждый тэг должен иметь закрывающий тэг, в котором после открывающей скобки следует слэш.

```
<тэг>
…
</тэг>
```

В языке s-выражений, на котором основан Лисп, правило еще проще: каждая конструкция записывается в круглых скобках, при этом скобки должны быть сбалансированы

```
(тэг1 … (тэг2 …) …)
```

В связи с существованием некоторых правил необходимо ввести понятие “правильности” документа: “правильным” называется такой размеченный документ, который строго соответствует синтаксическим правилам некоторого языка разметки.

В заключение введу различие между низкоуровневым языком разметки и высокоуровневым: предполагается, что на высокоуровневом языке информация будет размечаться (и читаться) человеком, а на низкоуровневом — машиной, программой.

ОБОБЩЕННЫЕ ЯЗЫКИ РАЗМЕТКИ

Обобщенные языки разметки получили свое название благодаря тому, что они содержат минимальное количество правил и практически не имеют предопределенных конструкций и тэгов. Такие языки могут служить базой для создания более конкретных (DSL — ***D**omain **S**pecific **L**anguages*) языков разметки или же использоваться для представления некоторых слаботипизированных данных, таких, например, как конфигурационные файлы.

GML

В качестве дани истории я начну свой рассказ с самого первого реально реализованного и работавшего языка разметки — GML. Несмотря на название (***G**eneralized **M**arkup **L**anguage* — Обобщенный Язык Разметки), GML в свое время использовался в основном для разметки текстов. Тэги в GML начинаются с двоеточия и заканчиваются точкой. Если какой-нибудь тэг требует строгого указания, где он кончается, то используется закрывающий тэг. В качестве примера рассмотрим простой текст, размеченный с помощью GML:

```
:h1.Пример текста с GML-разметкой
:p.Новый абзац.
:p.А вот пример тэга, требующего закрытия:
:ol
:li.Упорядоченный список
:li.А вот и второй элемент!
:eol.
Здесь списка уже нет.
:p.Для абзацев можно определить, где они
кончаются, без закрывающего тэга
```

Приведенные выше примеры тэгов взяты из стандартного текстового обработчика IBM — SCRIPT. В качестве внешнего представления для документов GML использовались специфические форматы файлов для вывода на устройство, как правило, на принтер.

S-выражения

Технически появившиеся еще в конце пятидесятых, s-выражения стали активно использоваться в качестве своеобразного языка разметки позже, с появлением необходимых вычислительных мощностей. S-выражение (*symbolic expression* — символьное выражение) представляет собой деревоподобную структуру, состоящую из вложенных списков [4]. Каждый список записывается в круглых скобках:

```
(элемент1 элемент2 … элементN)
```

при этом скобки всех вложенных списков должны быть сбалансированы, а если в элементе встречаются пробелы, то он должен быть заключен в кавычки. В стандартных s-выражениях нет никаких предопределенных тэгов, любая программа может определить свои. Например, текст, подобный примеру, приведенному в разделе о GML, можно записать так:

```
(document
  (h1 "Пример разметки в s-выражениях")
  (p "Новый абзац)
  (p "А вот и список"
    (ol
      (li "Первый элемент")
      (li "Второй элемент"))
   "Каждый тэг легко закрывается скобкой"))
```

Внешнее представление таких данных может быть абсолютно любым, в дальнейшем я рассмотрю несколько примеров специализированных языков разметки, основанных на s-выражениях.

SGML

Одним из важнейших (если не самым важным) современных языков разметки, а также одним из первых стандартизованных ISO языков является SGML [2, 5, 6]. Такие известные многим языки, как HTML, XHTML и XML, являются подмножествами SGML.

Общая идея SGML определяется двумя положениями:

• разметка должна быть декларативной: она должна описывать структуру документа и подобные свойства, а не указывать, каким образом тот должен обрабатываться;

• разметка должна быть строгой, чтобы ее можно было обрабатывать теми же средствами, которыми обрабатываются другие строго определенные объекты, такие, как программы или базы данных.

Основных синтаксических правил SGML довольно много, но мы не будем рассматривать все. Документы на SGML могут содержать:

• описание документа, содержащее информацию о его свойствах;

• пролог, в котором говорится о конкретном типе документа (об этой возможности будет подробно рассказано позднее);

• сам документ, содержащий один тэг верхнего уровня и какое-то количество вложенных тэгов.

Людям, знающим XML, стандартные SGML-тэги также будут знакомы: они записываются в угловых скобках, требуют закрытия, могут иметь "атрибуты" и не могут пересекаться. В описании документа может указываться большое количество опциональных свойств, определяющих его вид. Обратите внимание, что это лишь принятый по умолчанию конкретный синтаксис, в реальности же SGML-документом может быть документ, размеченный любым языком разметки, удовлетворяющий двум вышеуказанным положениям, например, тот же документ GML.

Можно сказать, что SGML — это некое "термоядерное оружие" среди языков разметки, обобщающее практически все, что можно обобщить, и подходящее для решения любой задачи.

В качестве примера использования чистого SGML можно привести электронную версию словарной статьи вроде статей из Оксфордского словаря английского языка (OED)[7]:

```
<entry>
  <hwsec>
    <hwgp>
      <hwlem>Слово</hwlem>
      <pron>Произношение</pron>
    </hwgp>
    <hwgp>
      <hwlem>Вариант слова</hwlem>
      <pron>Вариант произношения</pron>
    </hwgp>
    <etym>Этимология</etym>
  </hwsec>
  <sen>Определение
    <quot>Пример употребления
      <qdat>Когда?</qdat>
      <auth>Кем?</auth>
      <wk>Где?</wk>
      <qtxt>Цитата</qtxt>
    </quot>
  </sen>
</entry>
```

Размеченные таким образом словарные данные о слове затем отображаются на сайте (внешнее представление — HTML и картинки) или в программе-словаре (внешнее представление — картинки и текст).

XML

Без преувеличения можно сказать, что XML [8] является в настоящее время самым распространенным и самым известным языком разметки и представления данных. Число специализированных языков, основанных на нем, а также число программ, использующих его для представления своих данных, трудно поддается счету. Дальше в этой статье будут часто приводиться примеры языков — наследников XML.

XML был разработан как подмножество SGML, которое было бы более строгим, содержало бы гораздо меньше опциональных возможностей и могло быть читаемо одинаково хорошо как человеком, так и машиной. Такие возможности были крайне необходимы для представления данных для Всемирной паутины, поэтому законодателем мод в XML является консорциум W3C. Все стандарты на XML открыты.

Основы синтаксиса XML уже были описаны ранее в разговоре о синтаксисе по умолчанию в SGML. Однако есть ряд важных дополнений, касающихся именно XML:

• XML-документы должны быть представлены в кодировке Юникод;

• имена тэгов являются чувствительными к регистру;

• специальные символы (кавычки, амперсанды, угловые скобки и т.д.) должны быть экранированы с помощью указания их кода или специальной мнемоники;

• строки между <!-- … --> являются комментариями и игнорируются при разборе документа;

• в начале документа желательно указывать его тип.

В качестве простого примера данных, размеченных с помощью XML, приведем уже знакомый нам кусок текста:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- указание типа документа и кодировки -->
<document>
  <h1>Пример XML-разметки</h1>
  <p attr="Бесполезный атрибут">Новый
          абзац</p>
  <p>Список тут:
    <ol>
      <li>Первый элемент</li>
      <li>Второй элемент</li>
    </ol>
  </p>
</document>
```

Тэги, не имеющие содержимого, могут быть закрыты “на месте”:

```
<интересный_тэг/>
```

Одним из популярных представлений XML-документов является DOM (***D**ocument **O**bject **M**odel*, Объектная Модель Документа) — представление размеченного документа в виде объекта (в понимании языков программирования) с соответствующими свойствами, представляющими вложенные тэги. Эта модель довольно удобна для практического использования и используется повсеместно.

Количество возможных применений XML почти не ограничено, и многие из них будут рассмотрены далее, однако язык часто становится объектом критики в основном за свою избыточность и сложность. Здесь необходимо отметить, что изначально он не создавался как средство обмена высокоструктурированными данными и действительно может быть чересчур избыточным в таком случае.

SXML

Для большего удобства, в частности для применения в программах на функциональных языках программирования, был разработан альтернативный синтаксис XML в виде s-выражений [9]. Отображение между двумя синтаксисами является довольно простым:

- атрибуты записываются как (@ (атрибут “значение”) …);
- тэг декларации записывается как (*TOP* …);
- специальные объявления записываются как (*PI* …).

Примерно так будет выглядеть вышеприведенный пример на SXML:

```
(*TOP* (*PI* xml "version='1.0'")
  (document
    (h1 "Пример SXML-разметки")
    (p (@ attr "Бесполезный атрибут")
       "Новый абзац")
    …
    ))
```

YAML

В связи с повышением мощности вычислительных машин, а также усилением недовольства такими языками, как XML и желанием добиться большей человекочитаемости от языков разметки, в последнее время все более популярным становится язык YAML [10], основу которого заложили принципы современных скриптовых языков программирования Perl и Python и формат сообщений электронной почты. YAML — это рекурсивный бэкроним[1] “YAML Ain’t Markup Language” — “YAML это не язык разметки”[2]. Отчасти шутливое, это название призвано показать то, что YAML является не столько языком разметки документов, сколько общим языком описания данных.

Поскольку YAML — это в первую очередь человекочитаемый язык, его правила довольно сложны для машины. Язык содержит три типа сущностей (что уже необычно для обобщенного языка разметки):

- Списки, варианты их записи:
  - — в несколько строк, новый элемент начинается с дефиса;
  - — в квадратных скобках через запятую.
- Ассоциативные массивы, имя ключа в которых заканчивается двоеточием. Варианты записи:
  - — в несколько строк;
  - — в фигурных скобках.
- Простые данные (“скаляры”).

Для отображения вложенности структур могут использоваться отступы.

Еще одной важной особенностью YAML является наличие и распознавание программой-обработчиком простейших стандартных типов данных: целых и вещественных чисел, строк, логических значений, двоичных данных, закодированных в таких форматах, как BASE64 (где данные представляются печатаемыми ASCII-символами). Кроме того, в YAML присутствует возможность явно указать тип данных для элемента.

Определенному объекту YAML может быть присвоен идентификатор, который потом можно использовать для ссылки на этот объект.

Пример представления словарной статьи из OED в виде ассоциативного массива YAML:

```
--- это комментарий
entry:
  hwgp:
    - hwlem: Слово
      pron: Произношение
    - hwlem: Вариант слова
      pron: Вариант произношения
  etym: Этимология
  sen:
    def: Определение
    quot:
    --- Пример употребления
      qdat: Когда?
```

[1] Аббревиатура, содержащая саму себя в качестве составной части, расшифровка которой появилась после появления ее самой.

[2] Изначально расшифровка была “Yet Another Markup Language” — “Еще один язык разметки”.

```
    auth: Кем?
    wk: Где?
    qtxt: Цитата
```

JSON

JSON (***JavaScript Object Notation***, Объектная Нотация JavaScript)[11] — это по сути подмножество языка программирования JavaScript, в то же время являющееся подмножеством YAML.

В отличие от YAML, который является, как и SGML, своеобразным "средством от всего", JSON — довольно простой и маленький язык разметки и представления данных. В нем имеются лишь следующие типы: число, строка, логический тип, массив, объект (ассоциативный массив), null (пустое значение). При этом массивы и ассоциативные массивы записываются только в форме со скобками и запятыми. Приведем пример из раздела о YAML на JSON:

```
{
  "hwgp":
  [  {"hwlem": "Слово",
      "pron" : "Произношение"
     },
     {"hwlem": "Вариант слова",
      "pron" : "Вариант произношения"
     }
  ],
  "etym": "Этимология",
  "sen":
  {"def": "Определение",
   "quot":
   {"qdat": "Когда?",
    "auth": "Кем?",
    "wk": "Где?",
    "qtxt": "Цитата"
   }
  }
}
```

Любые данные, правильно размеченные с помощью JSON, являются также правильным кодом на JavaScript (как и в случае с s-выражениями, когда размеченные ими данные являются кодом на Лиспе).

Как видно из примеров, YAML и JSON лучше, чем XML подходят для разметки сильноструктурированных данных, но требуют введения лишних сущностей для представления таких вещей, как разметка текста.

МЕТАЯЗЫКИ И ЯЗЫКИ СТИЛЕЙ

Языки описания типа документа

Поскольку описанные ранее обобщенные языки разметки накладывают крайне мало ограничений на вид тэгов и управляющих конструкций, может встать вопрос о проверке "правильности" документа относительно требований какой-нибудь области применения. Такой тип правильности принято называть "валидностью", а сам документ — "валидным".

DTD

Для проверки документа на "валидность" существует специальное множество языков разметки, называющееся языками описания типа данных, или "языками схем". Одним из примеров такого языка является SGML (или XML) DTD (***Document Type Definition***)[12]. Не углубляясь, я приведу лишь пример описания на нем, позволяющего проверить на валидность нашу статью из OED:

```
<!ELEMENT entry (hwsec,sen)>
<!ELEMENT hwsec (hwgp+)>
<!ELEMENT hwgp (hwlem,pron)>
<!ELEMENT hwlem (#PCDATA)>
<!ELEMENT pron (#PCDATA)>
<!ELEMENT sen (#PCDATA,quot*)>
<!ELEMENT quot (qdat,auth,wk,qtxt)>
<!ELEMENT qdat (#PCDATA)>
<!ELEMENT auth (#PCDATA)>
<!ELEMENT wk (#PCDATA)>
<!ELEMENT qtxt (#PCDATA)>
```

Здесь декларация #PCDATA обозначает чистые данные, а списки в скобках определяют структуру каждого тэга.

Для указания того, что документ использует это описание, в него нужно включить такую строку:

```
<!DOCTYPE entry SYSTEM oed_entry.dtd>
```

XML Schema

Еще одним языком описания структуры XML-документа является рекомендованная консорциумом Всемирной паутины XML Schema [13]. Перепишем с ее использованием определение из предыдущего абзаца:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="entry" type="entry"/>
  <xs:complexType name="entry">
    <xs:sequence>
      <xs:element name="hwsec" type="hwsec">
       <xs:element name="sen" type="sen">
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="hwsec">
    <xs:sequence>
      <xs:element name="hwgp" type="hwgp"
minOccurs="1">
    </xs:sequence>
  </xs:complexType>
  …
</xs:schema>
```

Можно заметить, что XML Schema — довольно многословный метаязык.

В настоящее время существует довольно много языков для описания типов XML и SGML документов, с помощью которых любой может создать свой тип описания данных.

Языки стилей

Текст, размеченный с помощью языков разметки типа XML и SGML, может быть сильно перегруженным и плохо читаемым для человека. Для целей выделения из него значащих деталей или представления в более удобном виде служат так называемые “языки стилей” (внешнее представление — по-другому размеченный текст). Работа с ними — обширный раздел информатики, выходящий за рамки этой статьи, поэтому мы не будем на нем особо останавливаться.

В качестве простейшего примера приведу стиль на языке XSL (XSLT) [14], который преобразует нашу словарную статью в перечисление вариантов слова:

```
<?xml version="1.0"?>
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/
Transform"
  version="1.0">
  <xsl:output method="xml" indent="yes"/>
  <xsl:template match="entry">
    <words>
      <xsl:apply-templates/>
    </words>
  </xsl:template>
  <xsl:template match="hwgp">
    <word>
      <xsl:apply-templates/>
    </word>
  </xsl:template>
  <xsl:template match="sen"/>
  <xsl:template match="pron"/>
  <xsl:template match="etym"/>
</xsl:stylesheet>
```

После применения такого стиля статья будет выглядеть примерно так (с точностью до пустых строк):

```
<?xml version="1.0"?>
<words>
    <word>
      Слово
    </word>
    <word>
      Вариант слова
    </word>
    …
</words>
```

Для SGML также существует основанный на s-выражениях язык стилей DSSSL.

Формально говоря, языки стилей позволяют разделить описание собственно данных от их внешнего представления. В разделе об HTML я еще вернусь к разговору о стилях.

РАЗМЕТКА ТЕКСТОВЫХ ДАННЫХ

Важнейшей сферой применения языков разметки была и остается компьютерная типография, по большей части представляющая текстовые документы с небольшим количеством графики и сырых двоичных данных. Разумеется, что для целей обработки такой информации было разработано множество мощных языков.

Разметка в Вебе

Несмотря на появление большого количества мультимедийного контента, Всемирная паутина остается преимущественно текстовой.

HTML

Главнейшим языком разметки веб-страниц является HTML [15], своими корнями уходящий в SGML и XML, но значительно ослабляющий требования к соблюдению правил разметки, например, HTML-теги могут пересекаться и не обязаны быть закрытыми. Современные веб-браузеры могут отобразить HTML-документы практически сколь угодно плохо сформированные. Внешним представлением HTML-документа, как правило, является изображение на экране компьютера — веб-страница, открытая в браузере.

Несмотря на то, что HTML в наше время изучается даже в школах и представляет собой один из простейших языков разметки, приведем пример и на нем:

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML-разметка</title>
    <meta http-equiv="content-type"
    content="text/html;charset=UTF-8"/>
  </head>
  <body>
    <h1>Пример HTML-разметки</h1>
    <p>Новый абзац</p>
    <a href="http://www.w3schools.com">
      Ссылка на W3Schools.com!
    </a>
    <p>А вот и список:
      <ol>
        <li>Первый элемент</li>
        <li>Второй элемент</li>
      </ol>
    </p>
  </body>
</html>
```

Важной особенностью HTML является наличие гиперссылок, по которым можно перейти на другой документ или в другое место того же самого документа.

Для HTML существует несколько спецификаций DTD, различающихся своей строгостью. Наиболее строгим вариантом можно считать XHTML Strict — спецификацию, требующую полного соответствия синтаксическим правилам XML.

CSS

Одним из основных языков стилей для семейства XML является CSS (***C**ascading **S**tyle **S**heets*, Каскадные таблицы стилей) [16]. Синтаксис CSS довольно гибок и позволяет задавать стили как для отдельных

элементов, так и для их сложных комбинаций, мы же рассмотрим простейший вариант CSS-стиля, который выделит заголовки в нашем HTML-документе красным шрифтом на зеленом фоне и выберет для всех абзацев гарнитуру шрифта “Arial”:

```
h1 {
  color: red;
  background: green;
  }
p {
  font-family: Arial;
  }
```

В HTML использование файла стилей может быть показано так:

```
<head>
  …
   <link rel="stylesheet" type="text/css"
href="test.css"/>
  …
</head>
```

Как и во всех других языках стилей, синтаксис CSS разбивается на выделение селекторов, таких, как h1 или p, и задание элементам с соответствующим селектором необходимых свойств (цвета, шрифта, выравнивания и т.д.). CSS может применяться для задания стилей любых XML-документов, а не только HTML.

Простая веб-разметка

Для широкого круга пользователей: участников форумов и сетевых дискуссий, редакторов общедоступных энциклопедий и справочников, — были разработаны предельно упрощенные языки разметки, позволяющие за несколько минут освоить основные принципы и начать собственно дискуссию или редактирование.

Вики-разметка

Вики-разметка — это тип разметки, принятый на различных вики-сайтах: зачастую, интернет-справочниках, редактировать которые может любой желающий. Общепринятого стандарта вики-разметки до сих пор нет, но различные варианты во многом похожи между собой. В качестве примера приведу наш текст, размеченный по правилам, принятым в Википедии — крупнейшей вики-энциклопедии [17]:

```
== Пример вики-разметки ==
Абзацы ничем не выделяются.
Пустая строка начинает новый абзац.
А вот и список:
* Первый элемент
* Второй элемент
** Вложенный элемент
[[Ссылки]] - в квадратных скобках обычно
указываются ссылки на страницы внутри дан-
ного вики-проекта
```

bbCode и его подобия

На всевозможных сетевых форумах обычно применяются языки разметки, подобные языку bbCode. В целом все они похожи на HTML с точностью до синтаксиса некоторых тэгов и того, что тэги записываются в квадратных скобках. Как правило, в силу специфики дискуссии на форуме такие языки обеспечивают визуальную разметку, а не логическую. Вот небольшой пример:

```
[b]Это жирный текст[/b]
[i]А это курсив[/i]
[url=http://www.w3schools.com]
  Зачем тут ссылка на W3Schools?
[/url]
[quote]Цитата другого участника дискус-
сии[/quote]
```

Компьютерная типография — LaTeX

Одним из самых серьезных средств логической разметки для компьютерной типографии, практически не имеющим аналогов, является разработанная Дональдом Кнутом система TeX [3] и ее идейный наследник и продолжатель LaTeX [18]. Система используется как в промышленной типографии, так и для подготовки документов к печати в личных целях. Особенно ее облюбовали академические научные учреждения, за неимоверную гибкость, возможность простого представления математических формул, таблиц, сложных рисунков и шрифтов.

Описать весь TeX практически нереально, поскольку он содержит невероятное количество управляющих конструкций и сторонних пакетов, расширяющих его функциональность, а также очень мощные возможности макроопределений и подстановок. Управляющие конструкции TeX имеют в начале бэкслэш (\), после которого следует имя конструкции. После имени в квадратных скобках могут быть указаны различные опции. После имени или списка опций обычно в фигурных скобках указывается значащая информация — имя пакета, тип декларации или текст заголовка. Комментарии начинаются с символа “%”. Ниже представлен пример документа, оформленного с помощью LaTeX:

```
% Это отчет, основной шрифт - 12 пунктов,
% формат бумаги — A4
\documentclass[a4paper,12pt]{report}
% Кодировка — utf8
\usepackage[utf8]{inputenc}
% Включим автоматическую поддержку
% переносов для русского языка
\usepackage[russian]{babel}
% Метаданные
\title{Такой интересный отчет}
\author{Иван Сукин}
% Начало собственно документа
\begin{document}
% Создадим титульную страницу
\maketitle
```

```
\newpage
% И автоматическое оглавление!
\tableofcontents
\chapter{Первая глава}
\section{Первый раздел}
Какой-то текст
\end{document}
```

LaTeX имеет встроенную поддержку самых разных типов документов: от инженерных статей и отчетов до сборников стихов и нотных записей. Внешнее представление LaTeX-документов также может быть самым разным: PS и PDF-файлы, HTML-страницы, рисунки и др. К примеру, эта статья была изначально сверстана в LaTeX.

Отображение математических формул

Важной особенностью TeX является возможность простой записи математических формул любой сложности. Внутристрочные формулы записываются между символами $, выключные — между парами $$, а выключные нумерованные — в окружении:

```
\begin{equation} … \end{equation}
```

К примеру, уравнение Кортевега — де Фриза:

$$\frac{\partial u}{\partial t}+6u\frac{\partial u}{\partial x}+\frac{\partial^3 u}{\partial x^3}=0$$

и его интегралы движения:

$$I_n=\int P_n\left(u,\frac{\partial u}{\partial x},\dots\right)dx$$

в LaTeX запишутся как:

```
$$
\frac{\partial u}{\partial t} +
6u\frac{\partial u}{\partial x} +
\frac{\partial^3u}{\partial x^3} = 0
$$
$$
I_n = \int P_n\left(u,
\frac{\partial u}{\partial x},
\ldots\right)dx
$$
```

LaTeX – запись математических формул принята во многих сферах, например, в Википедии.

MathML

Еще одним стандартом записи формул, рекомендованным для использования в Веб, является основанный на XML язык MathML [19], однако, как и свой родитель, MathML очень многословен. Например, формула для полинома $ax^2 + bx + c$ на нем будет записана так:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCYPE math PUBLIC "-//W3C//DTD MathML
2.0//EN"
        "http://www.w3.org/Math/DTD/math-
ml2/mathml2.dtd">
<math xmlns="http://www.w3.org/1998/Math/
MathML">
  <mrow>
    <mi>a</mi>
    <msup>
      <mi>x</mi>
      <mn>2</mn>
    </msup>
    <mo>+</mo>
    <mi>b</mi>
    <mi>x</mi>
    <mo>+</mo>
    <mi>c</mi>
  </mrow>
</math>
```

Сравните с записью в TeX:

```
$ax^2+bx+c$
```

Однако, запись формулы с помощью MathML лучше отображает ее структуру и проще для машинного разбора.

ЯЗЫКИ РАЗМЕТКИ В ДРУГИХ ОБЛАСТЯХ

Как я уже говорил ранее, языки разметки могут применяться в самых различных областях человеческой деятельности. Спектр этих областей настолько велик, что потребуется многотомная книга, чтобы хоть как-то описать их. Подавляющее большинство языков разметки для этих областей являются теми или иными спецификациями (DTD) XML или SGML, но некоторые возможности доступны и в рамках LaTeX.

Среди самых известных специализированных языков разметки можно выделить следующие:

- SVG, MetaPost, LaTeX-TikZ, POV-Ray, YafaRay — графика;
- COLLADA, VRML, X3D — 3D-графика;
- GeographyML — географические карты;
- Lilypond, Sibelius — музыка;
- XMPP — обмен сообщениями;
- CML, PDB — химия, биохимия;
- Glade, QML — графические интерфейсы пользователя;
- NetCDF — обобщенные научные данные.

Перечислить же все подобные языки не представляется возможным.

НИЗКОУРОВНЕВАЯ РАЗМЕТКА

Важным этапом в обработке размеченного текста является его отображение в наиболее удобном для человека виде. Для этих целей они отправляются на различные устройства — дисплей, принтер и т.п. Разумеется, что рассмотренные в статье языки слишком сложны для обработки такими устройствами, поэтому были разработаны промежуточные низкоуровневые языки разметки. Как правило, эти языки являются полноценными языками программирования и предоставляют средства визуальной разметки. В принципе никто не мешает сразу представлять свои документы с их помощью, хотя это и считается дурным тоном.

POSTSCRIPT

PostScript [20] — это один из стандартных низкоуровневых языков разметки, напрямую поддерживаемых принтерами. Язык представляет собой стековую машину, знакомую людям, имевшим дело с языком Forth, и даже дает все возможности для полноценного программирования (хотя этим на PS занимаются уж очень упорные энтузиасты). В качестве примера можно рассмотреть следующую программу:

```
%!PS-Adobe-1.0
% Переведем систему координат
% из дюймов в миллиметры
72 25.4 div
dup
scale
% Переместим начало координат в
% точку 10мм, 10мм
10 10 translate
% Выберем шрифт
/Arial findfont
% Растянем до 14мм
14 scalefont
setfont
% Перейдем в точку (40, 0)
40 0 moveto
(PostScript test!) show
showpage
```

Эта программа установит новую миллиметровую систему координат, с началом в точке (10мм, 10мм) и выведет в точке (40мм, 40мм), в новых координатах, текст “PostScript test!”. Операция showpage обычно является командой принтеру для печати очередной страницы или командой специальному интерпретатору для вывода страницы на экран.

Документы на PostScript могут содержать бинарные данные, такие, как растровые изображения. Все остальные элементы таких документов являются векторными и не зависят от разрешения устройства.

PDF

Другим низкоуровневым и, более того, двоичным форматом представления данных является PDF. PDF практически нечитаем для человека, однако хорошо читаем машиной, в качестве составных частей PDF-документа могут присутствовать XML-метаданные, специальные команды, двоичные сжатые и несжатые данные. Ниже приведен отрывок из PDF-файла, сгенерированного из PostScript-файла из предыдущего раздела, отвечающий за инициализацию одного из шрифтов:

```
8 0 obj
<</BaseFont/UAKAPD+Arial/FontDescriptor 9 0 R/
Type/Font
/FirstChar 32/LastChar 116/Widths[
278 278 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
667 0 0 667 0 0 0 0 0 0 0 0 0 0 0
0 0 0 500 0 556 0 0 0 222 0 0 0 0 0 556
556 0 333 500 278]
/Encoding/WinAnsiEncoding/Subtype/Type1>>
endobj
```

ЗАКЛЮЧЕНИЕ

В статье были рассмотрены многие основные существующие в данное время стандарты и языки компьютерной разметки документов. Совершенно ясно, что существование вычислительной техники без них практически невозможно. Основной акцент был сделан на обобщенные языки и основные принципы разметки, при этом неописанными остались многие интересные случаи специализированных языков, таких, как DocBook, groff, NetCDF, Lilypond, RTF, и многих других. Заинтересовавшиеся могут обратиться к дополнительной литературе.

ИСПОЛЬЗОВАННЫЕ ИСТОЧНИКИ ИНФОРМАЦИИ

1. *Harvey Bingham.* SGML: In memory of William W. Tunnicliffe. [Электронный ресурс] URL: *http://xml.coverpages.org/tunnicliffe.html*.
2. *C.F. Goldfarb and Y. Rubinsky.* The SGML Handbook. Oxford University Press, 1991.
3. *Кнут Д.* Все про TEX. М.: Вильямс, 2006.
4. *John McCarthy.* Recursive Functions of Symbolic Expressions and Their Computation by Machine. [Электронный ресурс] URL: *http://www-formal.stanford.edu/jmc/recursive/recursive.html*.
5. TR 9573-13. Information technology — SGML support facilities — Techniques for using SGML. Tech. rep. ISO/IEC, 1991.
6. Обзор ресурсов по SGML. [Электронный ресурс] URL: *http://www.w3.org/MarkUp/SGML/*.
7. Оксфордский словарь английского языка. [Электронный ресурс] URL: *http://www.oed.com*.
8. Официальный сайт XML. [Электронный ресурс] URL: *http://www.w3.org/XML/*.
9. XML и Scheme. [Электронный ресурс] URL: *http://ssax.sourceforge.net/*.
10. Официальный сайт YAML. [Электронный ресурс] URL: *http://www.yaml.org/*.
11. Официальный сайт JSON. [Электронный ресурс] URL: *http://json.org/*.
12. Формальное описание DTD. [Электронный ресурс] URL: *http://www.w3.org/TR/REC-xml/#dt-doctype*.
13. DTD для языка XML Schema. [Электронный ресурс] URL: *http://www.w3.org/2001/XMLSchema.dtd*.
14. Описание XSLT. [Электронный ресурс] URL: *http://www.w3.org/TR/xslt20/*.
15. Рабочая спецификация HTML5. [Электронный ресурс] URL: *http://dev.w3.org/html5/spec/*.
16. Спецификация CSS. [Электронный ресурс] URL: *http://www.w3.org/TR/CSS/*.
17. Вики-разметка Википедии. [Электронный ресурс] URL: *http://ru.wikipedia.org/wiki/Википедия:Как_править_статьи*.
18. *Балдин Е.М.* Компьютерная типография LATEX. БХВ-Петербург, 2008.
19. Официальный сайт MathML. [Электронный ресурс] URL: *http://www.w3.org/Math/*.
20. Adobe Systems Inc. PostScript Language Reference. Addison-Wesley, 1999.

Общероссийский проект

Школа цифрового века

Интернет-сопровождение проекта – Издательский дом «ПЕРВОЕ СЕНТЯБРЯ»

2012/13
учебный год

Предметно-методические материалы

Дистанционные модульные курсы

Бесплатно, адресно
каждому учителю!

Участие образовательного учреждения в проекте «Школа цифрового века» в 2012/13 учебном году позволит каждому педагогическому работнику получать с 1 августа 2012 года по 30 июня 2013 года в свой Личный кабинет на сайте www.1september.ru предметно-методические журналы Издательского дома «Первое сентября» и пройти дистанционные модульные курсы по Программе развития профессионально-личностных компетенций педагога.

Заявки принимаются от образовательных учреждений.

Оргвзнос за участие в проекте в течение всего 2012/13 учебного года – 4 тысячи рублей.

Величина оргвзноса не зависит от количества педагогических работников в образовательном учреждении.

Педагогическим работникам образовательного учреждения предоставляются документы, подтверждающие участие в проекте.

Прием заявок от образовательных учреждений

на сайте

digital.1september.ru

Общероссийский проект «Школа цифрового века» по комплексному обеспечению образовательных учреждений методической интернет-поддержкой разработан в соответствии с Федеральной целевой программой развития образования на 2011–2015 годы и направлен на развитие инновационного потенциала образовательных учреждений: вовлечение педагогических работников в цифровое образовательное пространство, повышение эффективности использования современных образовательных технологий (в том числе информационно-коммуникационных технологий) в профессиональной деятельности

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 сентября 2012 по 31 мая 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 1 октября по 30 декабря 2012 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

Электронную заявку можно в режиме on-line подать на сайте
edu.1september.ru

Азбука Роботландии. Информация

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.

Об "Азбуке" и первой части этого курса "**Компьютер**" мы писали на страницах "Информатики" в № 14 за 2011 год. В заметках этого года представляем результаты работы над второй частью курса под названием "**Информация**".

Надеемся, материал будет прочитан с интересом, ибо мы собираемся поделиться с читателями любопытными методическими находками, часть из которых поддержана специально разработанными программными средами, представленными на диске.

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

Кратко об "Азбуке Роботландии"

"Азбука Роботландии" — это проект, запущенный в 2010/2011 учебном году. Цель — создать четырехгодичный курс информатики для начальной школы, отвечающий, с одной стороны, современному уровню развития информатики (как по интерфейсу самого УМК, так и по его содержанию), а с другой, работающий на опережение за счет опоры на фундаментальные основы информатики и развитие алгоритмического мышления учащихся.

В "Азбуке" запланировано четыре ключевых раздела, соответствующих четырем годам обучения:

- 1-й год (1-й класс). **Компьютер** (разработано в 2010/2011 учебном году)
- 2-й год (2-й класс). **Информация** (разработано в 2011/2012 учебном году)
- 3-й год (3-й класс). **Алгоритмы**
- 4-й год (4-й класс). **Интернет**

Таким образом, "Азбука Роботландии" позиционируется как курс, закладывающий основы компьютерной, информационной, алгоритмической и коммуникационной грамотности младших школьников.

Мы хотим научить детей конкретным приемам работы с информацией, но самое главное — мы хотим научить детей мыслить алгоритмически, ведь основа информатики — это теория и

практика *составления алгоритмов* для обработки информации.

Развитие алгоритмического мышления — наша основная цель, но мы не хотим подменять другие формы мышления алгоритмическим. Мы хотим сформировать алгоритмическое мышление *наряду с другими*.

Алгоритмическое мышление — это способность облечь абстрактную идею в последовательность конкретных шагов, необходимых для ее воплощения на практике.

Алгоритмическое мышление формируется на фоне развития абстрактного и логического. Кроме того, мы заботимся о формировании правильного отношения к инструменту, поиску оптимальных алгоритмов работы с ним. Такие рефлекторные мысленные установки образуют то, что, по нашему мнению, составляет понятие информационной компетенции — навык выбора инструмента, адекватного решаемой задаче, и эффективное его использование (компетентностная линия курса).

Наконец, "Азбука Роботландии" закладывает в начальной школе реальную основу базовой (по школьной программе) информатики по всем ключевым ее темам (см. содержание уроков). В старших классах изучение информатики станет более осмысленным.

Основные задачи курса

- Алгоритмическое мышление
- Информационная компетентность

Вспомогательные задачи курса

- Любовь к чтению, письму, думанию и учению
- Умение находить точные формулировки
- Умение вести спор, отстаивать свои позиции
- Раскрытие индивидуальности
- Проявление коллективизма
- Помощь в изучении других предметов (использование примеров и заданий по материалам разных предметов)

Состав УМК

- Электронный интерактивный учебник-лаборатория (интерактивные тексты, тренажеры, практикумы, зачеты, вопросы и домашние задания).
- Электронный сборник поурочных методических рекомендаций.
- Сборник иллюстративных материалов к урокам (презентации).
- Сборник дидактических раздаточных материалов.

Вторая часть курса: "Информация"

Первая часть курса "Компьютер" знакомит первоклассников с базовым инструментом современной информатики. Во второй части закладываются общие представления об информации и информационных процессах, подробно рассматриваются списки, таблицы, иерархии, осваивается многострочное редактирование плоского текста на базе учебного редактора РМ-1 с блоком контроля выполнения заданий и в стандартном (для операционной системы) редакторе плоского текста (Блокнот для ОС Windows).

Термин "*простой текст*" (англ. *plain text*) относится к такой форме электронного письма, в которой присутствуют только символы самого текста, символы табуляции, символы конца строк и нет разметки, отвечающей за структуру текста (заголовки, абзацы, таблицы, цитаты, списки…) и его визуальное представление (цвет, начертание, размер, шрифт, выравнивание…). Такой текст программисты называют "плоским" (строго говоря, это — жаргон, но он вошел в обиход). Плоский текст принимают поля ввода форм (строчные и многострочные), редакторы плоского текста используют для записи кода на языках программирования, в том числе для создания веб-страниц. Программа Блокнот из набора программ ОС Windows является примером редактора плоского текста. Заметим, что автоматическая подсветка ключевых слов в некоторых "плоских" редакторах выполняется программным путем без использования разметки в самом тексте.

Алгоритмы в "Азбуке-2" продолжают алгоритмическую линию курса, присутствуя в уроках и как инструмент изложения материала, и как среда для практической работы с понятиями.

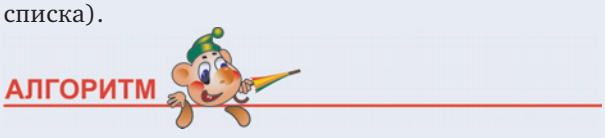
Пример алгоритма как инструмента изложения материала

Выделение текстового фрагмента выполняется по следующему алгоритму:

1. ***Поставить*** курсор на начало выделения.
2. ***Нажать*** клавишу Shift и не отпускать.
3. ***Удерживая*** клавишу Shift, ***установить*** курсор на конец выделения.
4. ***Отпустить*** клавишу Shift.

Пример алгоритма как среды для практической работы с понятиями

Выполните приведенный ниже алгоритм (закрепляем понятия: список, длина списка, индекс списка).



1. X ← (0,1,2,3,4,5,6,7,8,9)
2. Y ← Длина(X)
3. Z ← 12−Y+X_3

Основа изучения курса — практика. Дети осваивают материал, выполняя многочисленные задания:

❑ При чтении и обсуждении интерактивного текста учебника дома или вместе с учителем на большом демонстрационном экране в классе (автоматическая демонстрация правильных ответов по запросу).

❑ При работе с интерактивными фрагментами поурочных презентаций (автоматическая демонстрация правильных ответов по запросу).

❑ При ответе на вопросы повторения в учебнике (автоматическая демонстрация правильных ответов по запросу).

❑ В практикумах и зачетах учебника (автоматическая проверка правильности результата).

❑ При выполнении заданий на штатных программах компьютера (проверяет учитель, одноклассник или домашний наставник).

❑ При работе с раздаточными материалами за партой без компьютера (проверяет учитель или перекрестно друг друга).

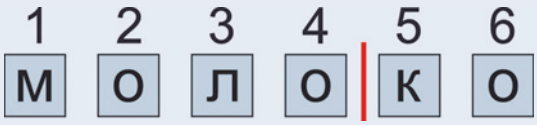
❑ В процессе постановки учебных ролевых игр (“дирижирует” и комментирует учитель).

❑ На алгоритмической или сюжетной физминутке урока (под управлением учителя).

❑ В процессе выполнения домашних заданий (проверяет учитель).

Ниже описывается поурочное содержание второй части курса “Азбука Роботландии”.

Тема 1. Информатика и информация

Название и план урока	Конспект урока
Урок 1. Что такое информация ❑ Никто не знает, что такое информация! ❑ Что изучает информатика ❑ Работаем с информацией • Хранение информации • Передача информации • Обработка информации • Большие и маленькие ❑ Что мы узнали	*Информация* — это любые сведения. *Информатика* — это наука, которая изучает способы *передачи*, *хранения* и алгоритмы *обработки* информации. Информацию можно *хранить*. *Количество информации* можно измерять *числом символов* в записи текста, включая символ пробела, знаки препинания и любые другие знаки. Определение количества информации между двумя вертикальными черточками, пример: |я и ты|. Информацию можно *передавать*. В компьютер информацию можно передавать, нажимая клавиши на клавиатуре. Ира нажимает клавиши со скоростью две клавиши в секунду. За сколько секунд Ира передаст в компьютер слово “молоко”? Объекты бывают разными по некоторой величине. И тогда их можно выстраивать в порядке убывания или возрастания этой величины. Переставьте цифры в числе 9364 в порядке возрастания
Урок 2. Многострочный редактор ❑ Первое знакомство • Одна строка • Когда строк много • Движение курсора • Координаты символов и место курсора • Редактор строки • Многострочный редактор • Алгоритмы перемещения курсора ❑ Урок редактирования в Роботландии • Команды Home и End • Набор текста • Алгоритмы перемещения курсора ❑ Что мы узнали	“Послушно” курсор ведет себя только внутри текста. Команда ВПРАВО на конце строки перемещает курсор на начало следующей строки. Команда ВЛЕВО на начале строки перемещает курсор на конец предыдущей строки. Курсор перемещается не по всей прямоугольной области редактора, а только по тем частям, где написан текст. *Координаты* — это величины, определяющие положение объекта. *Координатой символа в строке* является *номер места* этого символа. Курсор не занимает место символа. Он стоит *перед* или *за* символом.  В примере курсор стоит *перед* символом с координатой (5). Или *за* символом с координатой (4). В многострочном редакторе координатами символа являются два числа: *(номер строки, номер столбца)*. Выполняем алгоритмы, задающие перемещения курсора в многострочном редакторе. Команды Home и End отсылают курсор на начало и конец строки. Команда Enter на конце строки вставляет пустую строку. Набор многострочного текста

Урок 3. Как получить информацию

- ❑ Как человек получает информацию
- ❑ Как животные получают информацию
- ❑ Информацию можно получать по-разному
- ❑ Как компьютер получает информацию
- ❑ Как компьютер передает информацию
- ❑ Устройства ввода/вывода компьютера
- ❑ Как человек передает информацию
- ❑ Работаем с информацией
  - • Кузя считает
  - • Кузя ползет с разной скоростью
- ❑ Что мы узнали

У человека пять *органов чувств*: *глаза*, *уши*, *нос*, *язык*, *кожа*. Эти органы позволяют получать информацию. *Чувствами восприятия информации* называют: *зрение*, *слух*, *обоняние*, *вкус*, *осязание*. Органы чувств есть не только у людей, но и у животных.

Компьютер получает информацию от устройств ввода, таких, как клавиатура, мышь, дисковод, микрофон, сканер, флешка, веб-камера, винчестер, датчики.

Компьютер передает информацию на устройства вывода, такие, как монитор, принтер, дисковод, колонки, наушники, флешка, винчестер, моторы.

Устройства, с которых компьютер может получать и на которые может передавать информацию, называют устройствами ввода/вывода. Примеры: МФУ, дисковод, флешка, винчестер, сенсорный экран, фото- и видеокамеры. Понятно, что устройства ввода/вывода являются устройствами ввода и вывода.

Жук Кузя ползет вдоль текста и считает… число символов, букв, цифр; цифр, равных 5, больших 5, меньших 5.

Жук Кузя ползет вдоль текста с разной скоростью (2 символа в секунду, 1 символ за 2 секунды…). За сколько секунд он проползет весь текст?

Урок 4. Что можно делать с информацией

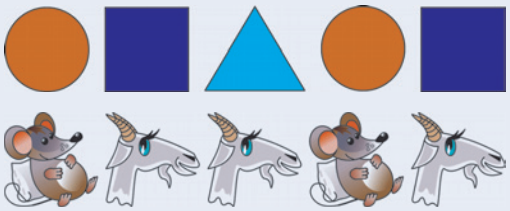
- ❑ Вася работает с информацией
  - • История первая. Вася читает книгу
  - • История вторая. Вася у классной доски
  - • История третья. Вася играет с Костей
- ❑ Информационные процессы в компьютере
- ❑ Работаем с информацией
  - • Кузя ищет продолжение цепочки
- ❑ Многострочный редактор
  - • Секрет клавиши Enter
  - • Клей
  - • Ножницы
- ❑ Что мы узнали

Информацию можно: хранить, передавать и обрабатывать.

Хранение, передача и обработка — это информационные процессы.

Компьютер хранит информацию в своей памяти, обрабатывает при помощи процессора, передает при помощи устройств вывода.

Найти продолжение цепочек:



Разрезать строку можно при помощи команды `Enter` (вставка символа конца строки).

Склеить строки можно при помощи команд `BS` и `Del` (удаление символа конца строки)

Урок 5. Информатика и информация

Тур 1. Информация
Тур 2. Работаем с информацией
Тур 3. Многострочный редактор

Введение понятия байта как единицы памяти компьютера. Повторение материала темы 1.

В Роботландии — экзамен! Экзаменаторы — Вася Кук, Буквоед и строгий РМ-1. Вася, Хролик и РМ-1 поздравили ребят с успешным завершением экзамена. Все участники получили портрет настоящего роботландского Профессора

Тема 2. Хранение информации. Списки

Название и план урока	Конспект урока
Урок 6. Где хранится информация ❑ Информация в природе ❑ Информацию сохраняет человек ❑ Информационные носители ❑ Что мы узнали	Природа хранит информацию в разных формах. Молекула *ДНК* хранит программу развития живого организма. Молекула *ДНК* находится в ядре клетки. Информационный носитель — это объект, на котором хранится информация. Человек придумал и использует многочисленные информационные носители 

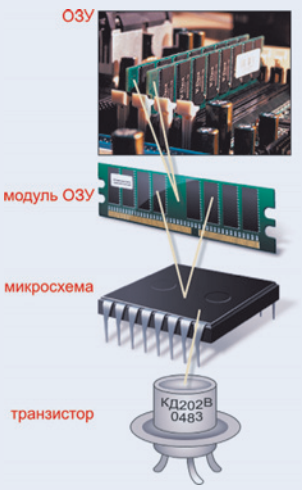
Урок 7. Хранение информации в компьютере

- ❑ Хранение информации в компьютере
- ❑ В каком виде хранится информация
- ❑ Сравнение компьютерных носителей информации
- ❑ Работаем с информацией
- ❑ Что мы узнали

ОЗУ состоит из одного или нескольких модулей памяти. *Модуль памяти* объединяет в себе несколько микросхем памяти. *Микросхема памяти* содержит миллионы транзисторов. *Транзистор* хранит `0`, если не пропускает ток, и хранит `1`, если пропускает ток.

Для описания хранения информации в компьютерной памяти используется язык нулей и единиц.

Сравниваем ОЗУ, винчестер и лазерный диск (по скорости, результату отключения питания, типу носителя, способу кодирования, объему)



Урок 8. Списки и действия над ними

- ❑ Хранить хочется много, а искать быстро
- ❑ Список
  - • Что рассказал о списках Хролик
  - • Примеры списков
  - • В списке важен порядок следования элементов
  - • Действия над списками
    - • Обозначение списка
    - • Длина списка
    - • Индекс элемента списка
- ❑ Работаем с информацией
  - • Вычислить
- ❑ Что мы узнали

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

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

```
X = (5,20,30,15)
```

Записан список с именем `X`. Этот список состоит из четырех чисел: `5`, `20`, `30` и `15`.

Длиной списка называют количество элементов в списке.

Индекс элемента в списке — это номер элемента в списке.

Дано	Вычислить
$X = (3,5,7,1)$	Длина$(X) - 1$
$X = (3,0,2,5)$	$X_1 + X_4$
$X = (10,5,7)$	Найти индекс максимального элемента в списке

Урок 9. Многострочный редактор

- ❑ Ножницы и клей (повторение)
- ❑ Вставка пустой строки
- ❑ Удаление текстового фрагмента
- ❑ Выделение клавиатурой
  - • Быстрое выделение сложных фрагментов
- ❑ Выделение мышкой
- ❑ Что мы узнали

Команда `Enter` позволяет “разрезать” строку в позиции курсора (вставка символа конца строки). Команды `BS` и `Del` позволяют “склеивать” строки, удаляя символ конца строки.

Алгоритм удаления непрерывного фрагмента текста:

1. Выделить фрагмент.
2. Удалить фрагмент.

Алгоритм выделения фрагмента текста:

3. *Поставить* курсор на начало выделения.
4. *Нажать* клавишу Shift и не отпускать.
5. *Удерживая* клавишу Shift, *установить* курсор на конец выделения.
6. *Отпустить* клавишу Shift

Урок 10. Работаем со списками	Модель — это упрощенный вариант некоторого объекта. Сколько списков можно составить из двух разных элементов? Запишите список из четырех элементов, в котором индекс каждого элемента совпадает с его значением. Маркированные и нумерованные списки. Пустой список — это как ноль в математике. Команда присваивания:  Работает команда присваивания по следующему алгоритму: 1. Выполняются вычисления *справа* от знака присваивания. 2. Результат записывается в ячейку, имя которой указано *слева* от знака присваивания. Выполните алгоритм (операция | удаляет один примыкающий к ней символ): 1. Z ← лес|| 2. Z ← стон|+Z
Урок 11. Хранение информации. Списки ❑ Тур 1. Хранение информации ❑ Тур 2. Списки ❑ Тур 3. Многострочный редактор ❑ Тур 4. Алгоритмы	Повторение материала темы 2. 

Тема 3. Хранение информации. Таблицы

Название и план урока	Конспект урока
Урок 12. Таблица. Индексы. Утиное правило ❑ Что такое таблица ❑ Что можно узнать по таблице ❑ Табличные индексы. Утиное правило ❑ Строим таблицы ❑ Что мы узнали	*Таблица* состоит из *клеток* (*ячеек*). По вертикали клетки образуют *столбцы*, по горизонтали — *строки*. Над столбцами и перед строками помещают дополнительные клетки, в них записывают *заголовки столбцов* (обычно сверху) и *заголовки строк* (обычно слева). *Количество строк и столбцов* в таблице называют *размером* таблицы. Элементы (или клетки) таблицы можно указывать при помощи двух индексов. Первый индекс определяет номер строки, второй — номер столбца. Роботландцы называют это утиным правилом:  Работа с исполнителем *Строитель таблиц*: конструируем таблицу по описанию

Урок 13. Использование таблиц

- ❑ Использование таблиц
  - • Таблица сложения
  - • Таблица умножения
  - • Свойства веществ
  - • Таблица пикселей
  - • Экран многострочного редактора
  - • Электронные таблицы
- ❑ Строим таблицы
- ❑ Что мы узнали

Таблицы сложения и умножения. Таблица свойств веществ. Таблица пикселей на экране монитора. Экран многострочного редактора. *Электронная таблица* — таблица, в которой значение части ячеек рассчитывается автоматически. Работаем с электронными таблицами.

Работа с исполнителем *Строитель таблиц*.

Задача 1. Постройте таблицу, в которой содержимое каждой ячейки равно сумме индексов этой ячейки.

Задача 2. Постройте таблицу по следующим правилам:

Если номер строки…

- ❑ …*равен* номеру столбца, то в ячейке число *2*.
- ❑ …*больше* номера столбца, то в ячейке число *1*.
- ❑ …*меньше* номера столбца, то в ячейке число *3*

Урок 14. Алгоритмы и таблицы

- ❑ Алгоритмы и таблицы
  - • Таблицы
  - • Алгоритмы
  - • Задания
- ❑ Многострочный редактор
  - • Откатка и накатка
- ❑ Что мы узнали

Повторяем все про таблицы, вспоминаем команду присваивания и выполняем алгоритмы. Примеры алгоритмов (T — заданная таблица чисел):

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

Редактор хранит “историю” правок в специальной памяти и может восстанавливать предыдущие состояния своего поля по команде *откатка*.

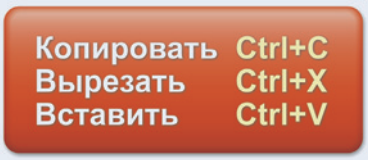
Откатка (Ctrl + Z)— отменяет правку.

Накатка (Ctrl + Y) — восстанавливает правку

Урок 15. Многострочный редактор. Буфер обмена

- ❑ Буфер обмена
  - • Работаем мышкой
  - • Работаем в меню программы
  - • Как работать быстро?
  - • Заключительные наставления: два важных правила
- ❑ Что мы узнали

Буфер обмена — специальная память для копирования *выделенной* информации. Информацию из буфера обмена можно копировать многократно. Команды для работы с буфером обмена:



Операции с буфером обмена можно выполнять и мышкой, пользуясь “всплывающим” *меню правок* (щелчок правой кнопкой). Те же команды можно найти и среди пунктов меню программы в разделе `Правка`.

Самый быстрый способ работы в редакторе — клавиатура. Мышь нужна только для того, чтобы войти в редактор.

Заключительные наставления:

Правило 1. Новая информация стирает старую в буфере обмена.

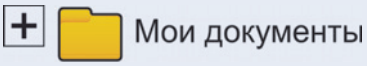
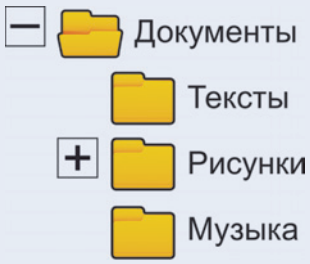
Правило 2. Буфер обмена можно использовать для переноса информации из одной программы в другую

Урок 16. Хранение информации. Таблицы

- ❑ Тур 1. Таблицы
- ❑ Тур 2. Многострочный редактор
- ❑ Тур 3. Алгоритмы

Повторение материала темы 3.

Тема 4. Хранение информации. Иерархии

Название и план урока	Конспект урока
Урок 17. Иерархия в виде дерева ❑ Что такое иерархия ❑ Иерархия в виде дерева ❑ Иерархия — это “вешалка”? ❑ Что мы узнали	Иерархия: ❑ это порядок подчинения одних объектов другим (солдаты подчиняются командиру); ❑ или это порядок вхождения одних объектов в другие (процессор входит в состав компьютера). Уровни, вершины, связи, корень, листья, родственные отношения вершин: потомки — прямые (дети) и не прямые (внуки, правнуки...), предки — прямые (родитель) и не прямые (бабушка, прабабушка…), сестры. В первой части курса (в прошлом учебном году) “вешалкой” называли слово, которым можно заменять другие слова. Например, слово *дерево* — “вешалка”! Им можно заменить слова: *ель*, *сосна*, *береза*, *тополь*, *дуб*, *ветла*. “Вешалка” — это, конечно, иерархия
Урок 18. Иерархия в виде лесенки ❑ Иерархия в виде лесенки ❑ Иерархия компьютерных папок ❑ Что мы узнали	Иерархию часто изображают в виде записи “лесенкой”. Ступеньки (отступы вправо) отражают подчинение или вхождение элементов и соответствуют уровням иерархии в виде дерева. На первой ступеньке (первом уровне) записывают корень иерархии. На втором — прямые потомки корня и так далее. Пример иерархии: компьютерные папки с программами и документами:  Эта иерархия превращена в меню. Знак “+” сообщает: папка `Документы` содержит другие папки. Щелчок на знаке “+” (или на папке) показывает содержимое в виде лесенки. А сам знак “+” превращается в знак “–”: 
Урок 19. Использование иерархии ❑ Использование иерархии • Сложное меню • Программа Проводник • Классификация • Классификация автомобилей ❑ Работаем с информацией • Задачи • Задача 1. Роботландские отличники • Задача 2. Как устроен завод • Задача 3. Что у нас в портфеле? • Задача 4. Содержание книги ❑ Что мы узнали	Структура завода:  Другие примеры: ❑ сложное меню; ❑ работа с папками в программе Проводник; ❑ классификация автомобилей. Решение задач в среде *Строителя иерархий* — построение иерархий по описанию или перевод дерева в лесенку. Обратный перевод: лесенка в дерево

Урок 20. Многострочный редактор ❑ Блокнот • Файл • Правка • Формат • Вид • Справка ❑ Что мы узнали	Знакомство с иерархическим меню Блокнота. Практика: создание текста, запись в файл, редактирование сохраненного текста, печать на принтере. 

Тема 5. Информатика и информация (повторение)

Название и план урока	Конспект урока
Урок 21. Большой экзамен в Роботландии ❑ Тур 1. Информация ❑ Тур 2. Списки ❑ Тур 3. Таблицы ❑ Тур 4. Иерархии ❑ Тур 5. Многострочный редактор	Повторение материала, пройденного за год. 
Урок 22. Конкурс 	На конкурсе предлагается 7 заданий (одно из них домашнее, выполняется до конкурса и присылается учителю по электронной почте). ***Задания*** (К) — задание выполняется на компьютере. (П) — задание выполняется письменно на листочке. 1. (К) *Алгоритмы*: построение алгоритма редактирования текста — домашнее задание. 2. (П) *Устройство компьютера* (повторение первого года обучения). 3. (К) *Информатика и информация:* вопросы на хранение, передачу и обработку информации. 4. (П) *Список:* имя, длина, индекс элемента. 5. (К) *Иерархия:* способы представления (дерево и лесенка). *Строитель иерархий*. 6. (П) *Таблица:* имя, размер, индексы элемента. 7. (К) *Блокнот:* набор текста с листочка

Тема 6. Проекты (уроки 23–28)

Курс завершает работа над проектами. Возможная тема: “Создание бумажной настенной газеты”.

Тексты статей набираются в Блокноте и печатаются на принтере, иллюстрации (авторские рисунки, фотографии, коллажи) собираются на бумаге, при этом можно привлечь графический растровый редактор Paint или векторный редактор из набора сетевых инструментов Google.

Темы проектов можно согласовать с учителем начальных классов, т.е. организовать работу (по созданию газеты, например), значимую для конкретного класса.

Резерв учителя (6 часов)

Методика курса учитывает, конечно, возрастные особенности младших школьников, но пытается достичь при этом ***максимума возможного в развитии детей***, делая ставку на ассоциативные связи и выстраивая последовательность простых шагов, связывающих простое и сложное, конкретное и абстрактное.

Резерв времени может быть использован на дополнительные уроки по сложным темам в тех учебных группах, в которых темп занятий согласно урокам учебника окажется слишком высоким.

Список сложных тем

❑ Списки и действия над ними: усвоение новых понятий (урок 8).

❑ Многострочный редактор: усвоение техники выделения фрагментов текста (урок 9).

❑ Таблица: усвоение новых понятий (урок 12).

❑ Буфер обмена: усвоение техники копирования фрагментов (урок 15).

❑ Иерархия в виде лесенки: представление иерархии двумя способами, в виде дерева и лесенки (урок 18).

❑ Блокнот: подробное знакомство с меню редактора (урок 20).

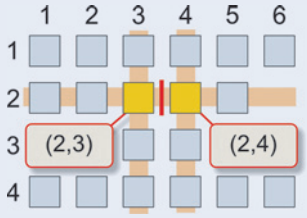
Неиспользованный резерв можно присоединить к проектной работе школьников.

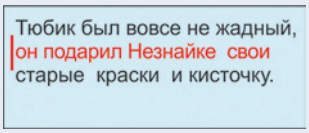
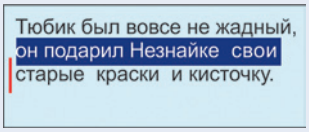
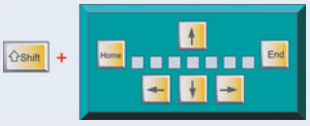
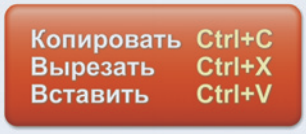
Методические проблемы и их решение

Ранее (“Информатика” № 14/2011) мы описывали, как сумели преодолеть методический барьер, обучая детей основам составления линейных программ. Тогда мы придумали исполнитель РС-1, который программно редактирует текстовую строку, создали среду для сборки программ и подготовили серию заданий на получение одного слова (фразы) из другого слова (фразы) с помощью стандартных команд редактирования текста (ВПРАВО, ВЛЕВО, BS, Del). Приготовленные задания оказались детям не по силам. Роль исполнителя детям понятна и привычна на собственном опыте, а роль составителя гораздо сложнее, ведь при составлении алгоритма нужно описывать действия, которые в данный момент не происходят. Детям первого класса не хватало воображения, чтобы представить последовательность действий, ведущих к требуемому результату (новому слову).

Вначале задача казалась методически неразрешимой, но все же мы преодолели трудности, опираясь на принцип постепенного продвижения от простого к сложному, с применением разных форм представления алгоритмических действий (предметных, словесных, знако-символьных и, наконец, в среде формального исполнителя). И к детям пришло понимание! Они легко получали “кот” из “котла”, “кран” из “экрана”, решали и более сложные задачи, испытывая радость от волшебного превращения слов. В работе детей появился неподдельный азарт.

В этом году возникли новые методические барьеры. Вот примеры того, на чем “споткнулись” наши второклассники.

Проблема	Иллюстрация проблемы	Как решалась
Где находится курсор, ***перед*** или ***за*** данным символом? Была предложена серия задач на определение конечного положения курсора после его перемещений по полю редактора. Введение координат не вызвало существенных затруднений, но, формулируя ответы, дети путались в семантике предлогов ***перед*** и ***за***	 На рисунке курсор стоит ***за*** символом с координатой (2,3) или ***перед*** символом с координатами (2,4)	Перед решением задач на выполнение алгоритмов перемещения курсора были предложены задания на определение положения курсора в состоянии покоя. “Где стоит курсор? ***За*** или ***перед*** символом с координатами (2,3)?” Ответ выбирается из выпадающего списка. Перед выполнением заданий на компьютере проводилось коллективное обсуждение в классе с ролевыми примерами (Катя стоит ***за*** Петей или ***перед*** ним?) и решение аналогичных заданий в интерактивном электронном учебнике на большом экране
Трудности с выполнением алгоритмов, перемещающих курсор по полю многострочного редактора, особенно при переходах через символ конца строки	Определить новое место курсора после выполнения алгоритма.  **Алгоритм** 1. ВНИЗ 2 2. ВПРАВО 4 Ответ: новое место курсора: [за, перед] символом ([1,2,3,4],[1,2,3,4,5,6,7,8,9]). В квадратных скобках показано содержание выпадающих списков выбора	Проанализировав ситуацию, мы поняли, что дети сталкиваются с двумя проблемами. Первая — в “нелогичности” поведения курсора на обоих концах строки (ВЛЕВО перед первым символом заставляет курсор прыгать на конец предыдущей строки, а ВПРАВО за последним — на начало следующей) и оконечные “зигзаги” по вертикали на строках разной длины. Вторая — при выполнении алгоритма, состоящего из двух и более шагов, приходится держать в голове предыдущие состояния курсора. Вторая проблема разрешилась сравнительно просто, ведь уровень абстрактного мышления наших второклассников значительно вырос в прошлом году — мы просто постепенно наращивали количество шагов в наших алгоритмических задачах. Для решения первой проблемы мы открыли “тайну” поведения курсора, рассказав о символе конца строки и о том, как реально строки многострочного редактора хранятся в его “однострочной” (линейной) памяти.

		Трудность не исчезла полностью, ведь конец строки является непечатным символом, его не видно на экране. Но по крайней мере мы смогли обосновать поведение курсора при переходе через символ конца строки. Наличие логики в поведении объекта конструктивнее волшебства и приучает детей во всем “докапываться” до сути
Большинство детей (как, впрочем, и взрослых) предпочитают работать мышкой, а не клавиатурой. В случае текстового редактирования это приводит к необоснованным потерям времени. И речь не столько о повышении эффективности конкретно текстового редактирования, сколько о формировании правильного отношения к любому инструменту, поиску оптимальных алгоритмов его использования	Выделить строку с символом конца строки. Дано:  Надо:  Решение: `Shift + ВНИЗ`	Чтобы заставить детей реально оставить в покое мышь после входа в текстовый редактор, была создана система компьютерных практикумов, которые распознают, чем работает пользователь, мышью или клавиатурой, и насколько эффективно выполняются операции. Подробности описаны в следующей заметке “Алгоритмы редактирования текста”
Детям непросто работать с клавиатурными аккордами. Первая трудность — техника выполнения аккорда. Дети пытаются одновременно нажать две клавиши, что часто приводит к неожиданному набору символа или выполнению другой команды. Вторая трудность — в запоминании самих аккордов	Выделение фрагмента:  Работа с буфером обмена:  	Трудности преодолеваются практикой. Освоение аккордов мы начинаем сначала за партой, имитируя работу с клавиатурой, затем при выделении текстового фрагмента (работаем курсором при ***нажатой*** клавише Shift): 1. *Нажать* клавишу Shift ***и не отпускать.*** 2. *Установить* курсор на конец выделения. 3. *Отпустить* клавишу Shift. Наконец, отрабатываем более сложные аккорды, работая с буфером обмена по следующему алгоритму: 1. *Нажимаем* Ctrl и держим (не отпускаем). 2. *Щелкаем* `Клавишу` (то есть нажимаем и быстро отпускаем). 3. *Отпускаем* Ctrl
Проблемы с понятием пустого списка и со свойством конечности количества элементов в списке	**Список** Под списком в курсе мы понимаем простой линейный список — ***конечное*** множество элементов, в котором ***важен порядок***. **Пустой список** — Зачем нужен пустой список? — спросил Лисенок Хролика. — А зачем нам число ноль? — в свою очередь, спросил Хролик. — Пустой список — это как ноль в математике!	Проблема с пустым списком была решена сравнением пустого списка с нулем в математике. — 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!

	Конечность количества элементов Будет ли списком множество: ❑ всех людей на Земле? ❑ всех чисел? ❑ всех точек в прямоугольнике? ❑ всех звезд на небе?	Проблема с конечностью количества элементов в списке была закрыта “умолчанием”! Множество всех деревьев или всех людей на Земле для детей так же бесконечно, как и множество всех чисел (к “последнему” числу всегда можно прибавить единицу, а за “последним” человеком якобы “всегда” можно поставить следующего). Анализируя проблему, мы поняли, что сами не понимаем бесконечность. Мы, конечные существа, просто привыкли к этому понятию и оперируем с ним, не задумываясь о его смысле. Чего же мы хотим от детей второго класса? Поразмышляв таким образом, мы просто исключили из определения списка свойство конечности, оставив только свойство порядка

Роботландский университет (руководитель — Дуванов Александр Александрович) открывает набор студентов на 2012/2013 учебный год.

В университет принимаются коллективные, а на курс “42. Web-конструирование” — коллективные и индивидуальные ученики.

Коллективный студент — это группа детей, работающая под руководством одного или нескольких наставников (наставник, как правило, — школьный учитель).

Индивидуальный студент — это учитель, желающий пройти обучение индивидуально, без группы детей.

В конце годичного курса при успешном обучении школьный учитель получает удостоверение от негосударственного образовательного учреждения “Роботландия” и удостоверение государственного образца о прохождении курсов повышения квалификации и переподготовки работников образования. Школьники получают удостоверение от “Роботландии”.

Занятия в университете платные (цены — на сайте *www.botik.ru/~robot/ru*). Они начинаются 10 октября текущего года и продолжаются в течение двух семестров до мая следующего года. Для подписчиков “Информатики” предусмотрена 10%-ная скидка за обучение.

Характерные черты Роботландской школы:

1. Совместное обучение учителя и школьников в рамках одной команды.
2. Турнирный цикл обучения.
3. Моделирование коллективной деятельности.
4. Реальная практическая польза детских проектов.
5. Перекрестные проверки работ.
6. Развитые горизонтальные связи.

Заявки принимаются по адресу: *kurs@robotland.pereslavl.ru* до 1 октября.

Заявку можно отправить со страницы *www.botik.ru/~robot/ru/set.htm*.

Краткое описание курсов

Номер и название курса	Возраст детей	Куратор курса	Описание курса
10. Азбука Роботландии	2–3-е классы	Первин Юрий Абрамович	Введение в информатику на базе курса “Азбука Роботландии”. Знакомство с компьютером, исполнителями, алгоритмами и программами. Первичные навыки редактирования информации. Сетевые конкурсы. Развитие алгоритмического и логического мышления
12. Азы информатики-I. Знакомимся с компьютером. Работаем с информацией	3–5-е классы	Кацай Ирина Ивановна	Начала информатики для малышей. Коллективная работа на сетевых сервисах Web 2.0
14. Азы информатики-II. Пишем на компьютере	4–6-е классы	Кацай Ирина Ивановна	В рамках обозначенной темы курс связывает пять контентов: познавательный, инструментальный, концептуальный, дизайнерский и творческий. Коллективная работа на сетевых сервисах Web 2.0
31. Азы программирования-I. Плюсик и Кукарача	6–8-е классы	Садовая Ирина Владимировна	Для детей, знакомых с понятиями “исполнитель”, “алгоритм”, “программа” и желающих поближе познакомиться с программированием
32. Азы программирования-II. Корректор	7–9-е классы	Садовая Ирина Владимировна	Продолжение курса 31 на базе исполнителя Корректор
42. Web-конструирование	старшие классы	Дуванов Александр Александрович	Создание сайтов и гипертекстовых приложений на базе HTML+CSS+JavaScript (последнее факультативно). Основы проектирования, веб-дизайна и юзабилити. Планируется кружок по теме, связанной с HTML5, CSS3 и JavaScript

ЭТО ПОЛЕЗНО ЗНАТЬ

Сайтowixация

А.И. Азевич,
Москва

Путешествуя по бескрайним просторам Интернета и рассматривая красивые сайты, невольно возникает желание создать нечто подобное. Это понятно и даже похвально. Правда, даже начинающему дизайнеру требуются серьезные знания в области веб-конструирования и веб-программирования. HTML, CSS, PHP — эти загадочные аббревиатуры вряд ли о чем-то говорят пользователю, не знакомому с веб-дизайном. Труд создателей сайтов сложен и многообразен. И все же тем, кто хочет порадовать своих родных или друзей виртуальной страничкой, на первых порах можно обойтись без сложных и малопонятных кодов. Чтобы подготовить свой первый виртуальный проект, достаточно воспользоваться одним из многочисленных онлайн-сервисов. Трудолюбивому пользователю он поможет создать яркий и интересный сайт!

Расскажем об одном из таких визуальных конструкторов сайтов. Наберитесь терпения, продумайте тематику, подготовьте соответствующие материалы для будущего сайта (фотографии, рисунки, тексты). А еще освежите в памяти знания в области английского языка, ведь сервис, о котором пойдет речь ниже, пока не имеет русскоязычного интерфейса (но даже не владея английским языком, путем проб и ошибок, можно легко догадаться, для чего нужна та или иная кнопка, пункт меню или команда). Зато он обладает массой полезных функций, которые по плечу любому начинающему “сайтостроителю”.

Итак, заходим на сайт *www.wix.com*. Регистрируемся, нажав на ссылке **SIGN UP**. На главной странице в глаза бросается крупная надпись “CREATE YOUR FREE STUNNING WEBSITE” (*рис.* 1). Это нас призывают бесплатно создать потрясающий сайт. А мы как раз этого и хотим!

Рис. 1

В главном меню программы (справа от логотипа WIX) — пункты **Create**, **Explore** и др. Первый ведет к редактору, в котором мы и будем работать. Кстати, эта команда дублируется на большой яркой кнопке с меняющимися надписями: **StartNow/It’ s Free**. Нас любезно приглашают приступить к созданию сайта, объясняя, что процедура совершенно бесплатна (понятно, что лишь ее базовая часть). Впрочем, для начала хватит и этого…

В нижней части страницы находится галерея шаблонов — уже готовых образцов сайтов, один из которых нам (вам ☺) предстоит заполнить новым содержанием, как говорят специалисты — “контентом” (см. *рис.* 2 на с. 49). Вы уже приготовили тексты, фотографии, рисунки? Тогда — за работу!

Выбираем пункт **Create** в верхнем меню программы и переходим к множеству красочных шаблонов. Они собраны по темам: бизнес, мода, дизайн, музыка и прочее (*рис.* 3). Выберите нужную тему, а в ней — подходящий дизайн сайта. Для определенности попробуем подготовить галерею фотографий. Расскажем немного о себе: увлечениях, друзьях, учебе. Да мало ли о чем еще можно рассказать. Главное — соберите нужные материалы в отдельную папку, чтобы не отвлекаться по мелочам в ходе захватывающего процесса сайтостроения!

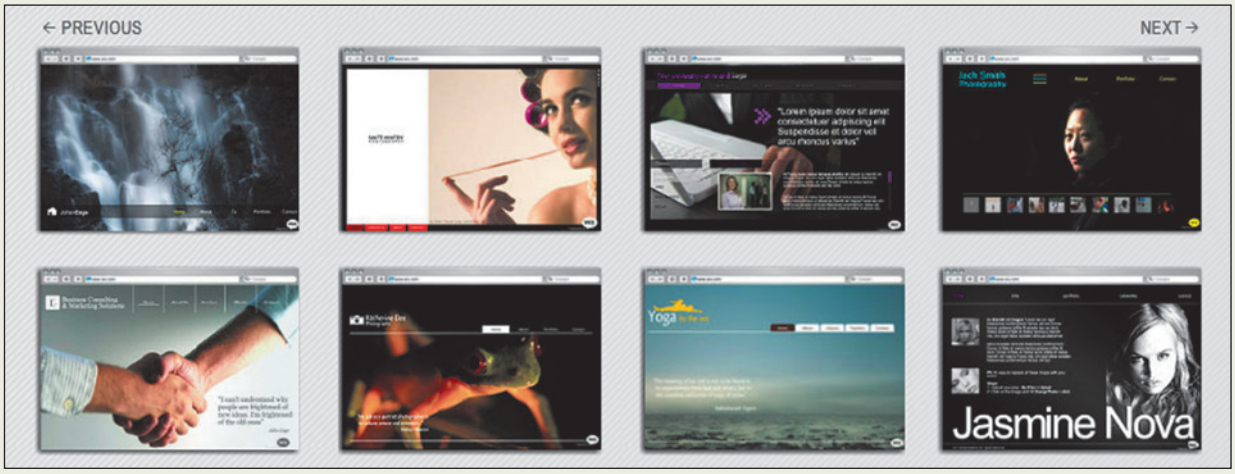


Рис. 2

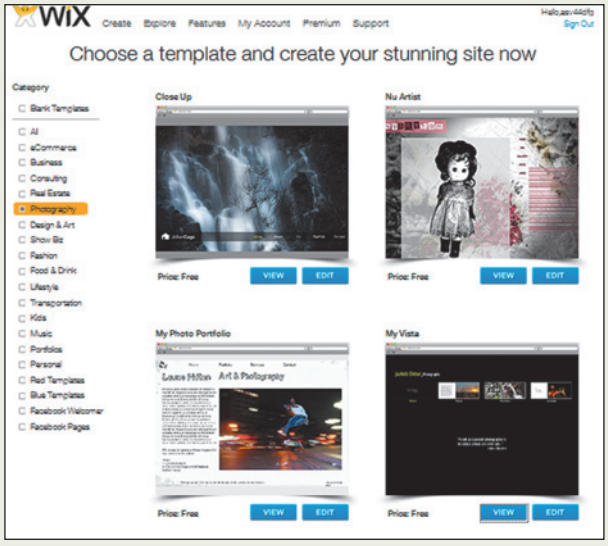


Рис. 3

Рис. 4

Выбрать шаблон непросто. Их так много — на любой цвет и вкус. В группе **All** привлекла внимание фотография с милыми овечками, безмятежно пасущимися на зеленом лугу (*рис.* 4). Нажав на кнопку **VIEW**, рассмотрим шаблон и, если нас все устраивает, перейдем к его редактированию, предварительно воспользовавшись кнопкой **EDIT**.

Нажав на нее, оказываемся в новом окне, в котором появилось множество команд. Вверху — панель с меню, инструментами и кнопками (*рис.* 5), слева — блок полезных функций (*рис.* 6). Рассмотрим их подробнее.

Разберемся с командами меню (*рис.* 5). Меню **File** поможет сохранить или опубликовать проект. Пункт **Edit** содержит инструменты для редактирования страницы. Копирование, удаление отдельных фрагментов, отмена последнего действия — вот лишь некоторые функции этой вкладки. Их гораздо больше, поищите, покопайтесь, изучите. Придумайте такое оформление страницы, чтобы будущий посетитель вашего сайта не захотел с него уходить! ☺

Следующий пункт — **Arrange**. С его помощью можно выполнять различные манипуляции с фотографиями: повороты, выделение отдельных фрагментов и т.д. Команда **Help** понятна без слов. Это помощь по сервису. А дальше идут кнопки, с помощью которых мы можем увеличивать или уменьшать размер шаблона, просматривать страницу в браузере, публиковать готовый проект.

Рассмотрим левый блок команд (*рис.* 6). Работать с ним очень удобно, даже занимательно. Подводим мышку к самой верхней кнопке с надписью **Add**, и тут выезжает полоска команд, с помощью которых можно добавлять на страницу текстовый блок, вставлять фотографию, видео, флэш-ролик, музыкальный фрагмент. Здесь же можно изменить навигацию на сайте.

Рис. 6

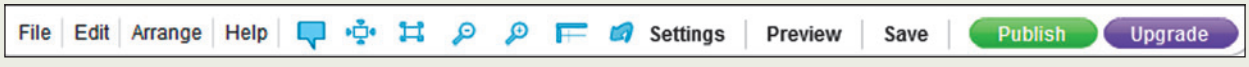


Рис. 5

Кнопка **Background** отвечает за фон страницы. Фоновых рисунков там так много, что глаза разбегаются! **MyFile** — это путь к вашим файлам: рисункам, фотографиям, трекам, видеоклипам. На своем сайте вы можете добавить блог с помощью одноименной кнопки. Вы и ваши друзья будут делиться в нем своими новостями. Это привлечет немало пользователей.

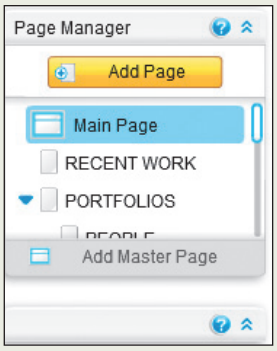


Рис. 7

Еще одна важная панель — **PageManager** — *рис.* 7. Она примыкает к правой части главной страницы. Если нужно добавить новые страницы, то это здесь. Нажмите на кнопку **Add Page** и выберите подходящий дизайн: текстовые блоки, галерею фотографий, коллаж, портфолио или что-то другое (*рис.* 8).

В этой же вкладке можно настроить слайд-шоу из фотографий, изменить вид горизонтального меню сайта. В ней будет храниться архив лучших фотографий. Такая функция тут тоже есть!

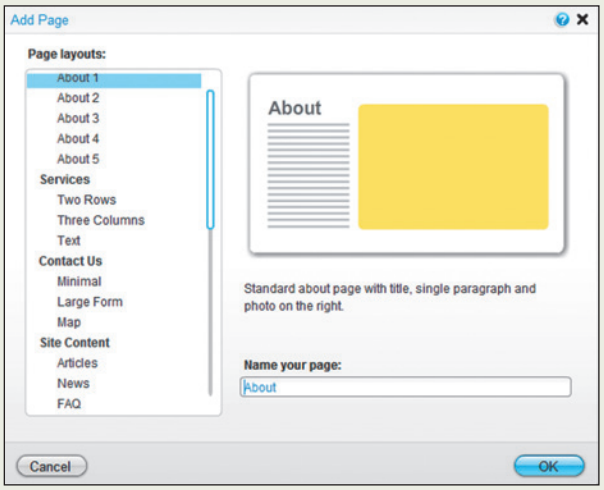


Рис. 8

Возможно, что, побывав на вашем сайте, кто-то захочет с вами связаться. Это тоже предусмотрено сервисом WIX. Создайте форму с контактной информацией, и весь мир узнает о вас! Изучив досконально все возможности последней вкладки, вы значительно улучшите дизайн вашего сайта, сделав его просмотр удобным, а вид — ярким и запоминающимся.

В последней вкладке содержится еще одна важная команда — **AboutMe**. Расскажите о себе. О городе или поселке, в котором вы живете, о школе, в которой учитесь, об одноклассниках, а может быть, о “друзьях наших меньших”. Создайте свой сайт, и друзей у вас станет еще больше. А если хорошо владеете иностранным языком (и хотите улучшить свои знания), пишите на нем. И тогда в группе пользователей будут не только соотечественники, но и жители других стран. Старайтесь, пробуйте, фантазируйте, учитесь. Пусть ваш первый сайт создается на готовом шаблоне, в котором все заранее спланировано. Пусть в нем пока далеко не все совершенно. Но лиха беда начало! Пройдет время, и вы научитесь создавать другие сайты, которые под силу настоящим профессионалам. Для этого надо учиться, шаг за шагом осваивая нелегкую науку сайтостроения.

Да, и не забудьте сохранить то, что сделали! Для этого нажмите на кнопку **Publish**. Затем в открывшемся окне придумайте имя сайта (*рис.* 9). Как только вы напишете несколько букв, сервис сразу же назовет адрес сайта в Интернете. Первый адрес будет не совсем удобным: длинным, с косыми черточками и точками, но зато личным.

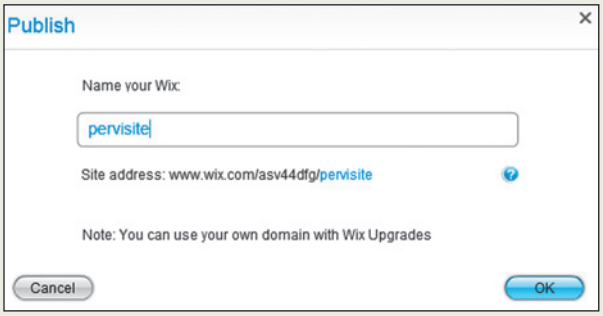


Рис. 9

Собственноручный проект можно с гордостью показать родным, друзьям и знакомым. Пусть они оценят вашу работу. Попросите посетителей оставить запись в вашем блоге. А может, и они увлекутся сайтостроением? Тогда будете создавать сайты вместе. Главное — многие узнают, что есть во Всемирной паутине и ваша собственная уникальная страница!

ПОИСК ИНФОРМАЦИИ

Пять вопросов

1. В опубликованном в 1812 году российским императором Александром I указе о правилах рекрутского набора был указан рост, который должен иметь рекрут. Какой именно (в метрах и сантиметрах)?
2. Как называется комар, чьи личинки предпочитает гамбузия?
3. Когда родился космонавт по имени Педро?
4. В каком фильме Олег Янковский сыграл роль карточного жулика?
5. Мифический ветер и сладкое блюдо — что это?

Ответы присылайте в редакцию (можно отвечать не на все вопросы).

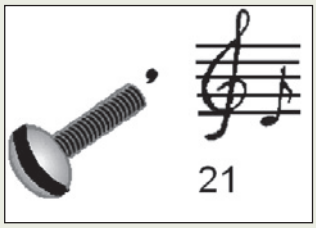
Ребусы по информатике

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

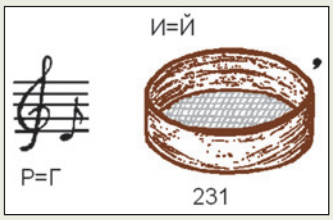
Предлагаю читателям раздела “В мир информатики” решить ряд ребусов, связанных с фамилиями людей, внесших большой вклад в развитие информатики и вычислительной техники.

Ребус № 1

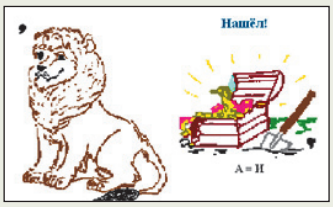
Ребус № 2



Ребус № 3

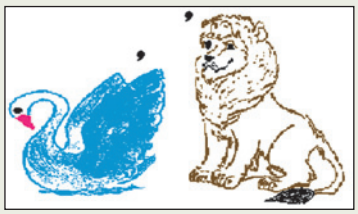


Ребус № 4



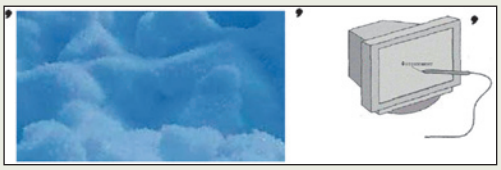
Ребус № 5

Ребус № 6

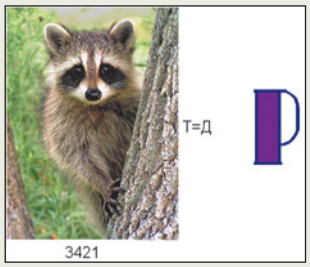


Ребус № 7

Ребус № 8



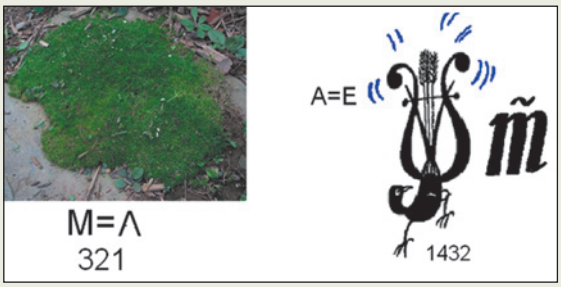
Ребус № 9



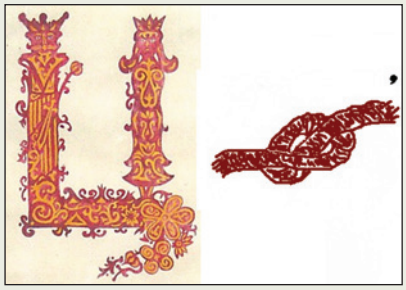
Ребус № 10

Ребус № 11

Ребус № 12



Ребус № 13



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

Головоломка с ДЕЦЛ

Нет, мы не имеем в виду известного рэп-исполнителя ☺. Нужно решить числовой ребус:

ДЕЦЛ = МАЛ + ДА + УДАЛ,

если число **ДЕЦЛ** должно быть как можно меньше. Как обычно в таких головоломках, одинаковыми буквами обозначены одинаковые цифры, разными буквами — разные цифры.

Прилипшая монета

Имеются 9 одинаковых с виду монет. Из них одна монета фальшивая, которая легче настоящих. Одна из монет прилипла в одной из чаш чашечных весов. Отодрать ее невозможно. Как за два взвешивания найти фальшивую монету? Гирь нет.

Двенадцать спичек

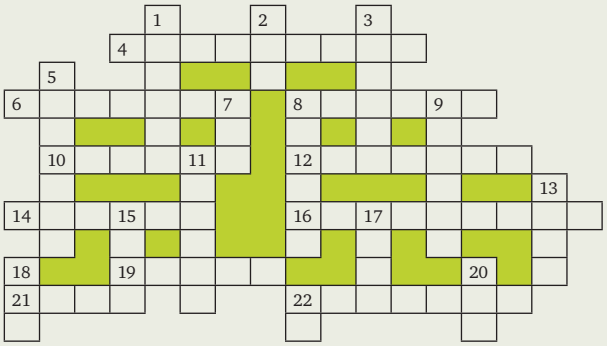
Из 12 спичек нужно на столе составить фигуру, в которой было бы три одинаковых четырехугольника и два одинаковых треугольника. Как это сделать?

Крест-накрест

Переставив буквы в строках приведенного ниже квадрата, получите “осмысленные” слова, при этом в диагоналях квадрата соберутся еще два слова, связанные с информатикой и компьютерами. Найдите все слова и дайте комментарии к ним.

Т	Е	С	К
В	О	Л	П
Н	О	Л	К
Т	А	Й	Б

Кроссворд



По горизонтали:

4. Алгоритм, записанный на языке, “понятном” данной ЭВМ.

6. Русский вариант написания названия социальной сети, которое с английского языка на русский можно перевести буквально как “Книга лиц”.

8. Примечание к тексту, размещаемое в нижней части страницы или в конце документа.

10. Древнегреческий математик, автор алгоритма нахождения наибольшего общего делителя двух натуральных чисел.

12. Так когда-то называли папку (поименованную группу файлов, объединенных по какому-то признаку).

14. Совокупность правил записи данных, записанных в компьютерном файле.

16. Логическая операция.

19. Тригонометрическая функция, расчет значения которой предусмотрен в языках программирования.

21. Совокупность электронных документов, которые воспринимаются как единое целое, оформленных с помощью языков HTML, Java и др.

22. Фамилия советского академика, под руководством которого были созданы первые отечественные ЭВМ, а также фамилия известного российского веб-дизайнера.

По вертикали:

1. Название клавиши.

2. Цифра четверичной системы счисления.

3. Ввоз товаров из-за рубежа, а также вставка в документ приложений Windows объектов из других приложений.

5. Вид связи.

7. Число в системе условных обозначений символов.

8. Участок дорожки магнитного диска.

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

11. Значение переменной величины или константы логического типа (русский вариант).

13. Программная конструкция, обеспечивающая повторение одних и тех же операций.

15. Устройство для соединения двух участков локальной сети, а также транспортное сооружение.

17.

18. …координат.
20. Элемент языка разметки гипертекста.
22. “Шестая” нота.

Два судоку

Решите, пожалуйста, две японские головоломки “судоку”:

1) простую:

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

2) сложную:

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

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

ПРИ1Ы

Три загадки — анекдота

Лейб Штейнгарц,
Израиль

Выберите один из предложенных ответов так, чтобы получился остроумный анекдот.

1. Чтобы сохранить мир в семье, необходимы любовь, нежность, терпение, уважение. Ну, и …, разумеется:

1) взаимопонимание;
2) компьютер;
3) два компьютера;
4) много денег.

2. Билл Гейтс бросил компьютеры и программы и занялся производством автомобилей. Теперь на его автомобилях при нажатии на педаль тормоза на панели будет высвечиваться сообщение:

1) “Вы перепутали педаль!”;
2) “Осторожнее, полиция!”;
3) “Большое спасибо!”;
4) “Вы действительно хотите затормозить?”.

3. Программист сидит за компьютером. Ему жена приносит кофе. Он, на нее не глядя, молча берет чашку, отхлебывает, потом морщится, поворачивается к жене и говорит:

— Я же не люблю без сахара!

— Я знаю, милый, …

Варианты окончания реплики:

1) “но мне так хотелось услышать твой голос!”;
2) “но ведь так полезнее”;
3) “но ведь так экономнее”;
4) “но извини, я просто не успела”.

Ответы присылайте в редакцию.

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 Пойа

Кто этот человек?

Н.Е. Кордина,
учитель информатики школы № 1,
Смоленская обл., г. Демидов

Решив 8 приведенных ниже задач, по полученным результатам определите (используя Интернет или другие источники) фамилию и имя человека, о котором идет речь в заданиях. В ответе приведите также полученную при выполнении частных заданий информацию и языки программирования, разработанные этим человеком.

1. День рождения

Определите, чему будет равно значение переменной *s* после выполнения следующего фрагмента программы:

```
x := 0
y := x
h1 := 1
h2 := 2
s := 2
нц пока x < 2 или y <= 6
  s := s + 2
  x := x + h1
  y := y + h2
кц
s := s + y/x + 3
```

Значение *s* совпадает с номером дня месяца, в котором родился наш герой.

2. Профессия отца

Определите, чему равно значение переменной z после выполнения следующего фрагмента программы:

```
x := -2
y := x + 5
если y > 0 и y < 2
  то
    z := x * y
  иначе
    z := x + y
все
z := x - z
```

Его отец был (в зависимости от полученного результата):

1) –6 — профессор математики;
2) –3 — школьный учитель;
3) 4 — инженер;
4) 1 — физик.

3. Увлечения в детстве

Дан фрагмент программы:

```
нц для i от 1 до 10
  нц для j от 1 до 10
    если i = j
      то
        A[i, j] := 1
      иначе
        A[i, j] := 0
    все
  кц
кц
```

Чему равна сумма значений элементов двумерного массива *A* размером 10 × 10?

Его увлечение в детстве (в зависимости от полученного результата):

1) 10 — авиамоделизм и разработка устройств дистанционного управления для моделей;
2) 100 — решение головоломок;
3) 0 — игра на музыкальном инструменте;
4) 1 — чтение научно-популярной литературы.

4. Учебные заведения

Все 10 элементов массива *A* первоначально были равны 0. Определите, как изменится этот массив после выполнения фрагмента программы:

```
t := A[10]
нц для i от 10 до 2 шаг -1
  A[i] := A[i - 1]
кц
A[1] := t
```

и в зависимости от результата узнайте цепочку высших учебных заведений, которые в хронологическом порядке окончил наш герой:

1) массив остался без изменений — “Лавальский университет, университет в Беркли, Технологический институт”;
2) все элементы получили значение, равное a[1], — “Технологический институт, университет в Беркли, Лавальский университет”;
3) последний элемент стал первым, а остальные переместились на позицию вправо — “Технологический институт, Лавальский университет, университет в Беркли”;
4) первый элемент стал последним, а остальные переместились на позицию влево — “университет в Беркли, Технологический институт, Лавальский университет”.

5. Тема диссертации

Все элементы двумерного массива *A* размером 10 × 10 первоначально были равны 0. Затем значения элементов изменились согласно представленному фрагменту программы:

```
нц для n от 1 до 4
  нц для k от n до 4
    A[n, k] := A[n, k] + 1
    A[k, n] := A[k, n] + 1
  кц
кц
```

Сколько элементов массива *A* в результате будут равны 1? Ответ на этот вопрос определяет математика, в честь которого назван язык программирования, разработанный нашим героем в диссертации:

1) 0 — Декарт;
2) 16 — Виет;
3) 12 — Эйлер;
4) 4 — Пифагор.

6. Созданный язык программирования

Дан фрагмент программы, обрабатывающий двумерный массив *A*:

```
нц для n от l до 6
  нц для m от l до 5
    A[n, m] := A[n, m] + (2 * n - m)
  кц
кц
```

Чему будет равно значение *A*[4, 3], если перед этими командами оно было равно 10? В зависимости от ответа определите язык программирования, разработанный “искомым человеком” во второй половине 1970-х годов в рамках конкурса Министерства обороны США:

1) 5 — Фортран;
2) 10 — APL;
3) 15 — Ада;
4) 25 — Си.

7. Основной принцип при разработке проектов

Значения элементов двумерного массива *A* были равны 0. Затем значения некоторых элементов были изменены в результате выполнения следующего фрагмента программы:

```
n := 0
нц для i от 1 до 5
  нц для j от 1 до 6 - i
    n := n + 1
    A[i, j] := n
  кц
кц
```

Ответ на вопрос “Какой элемент массива *A* будет иметь в результате максимальное значение?” определяет продолжение фразы, формулирующей основной принцип, которого придерживался наш герой при разработке всех своих проектов: “Делай просто, насколько возможно, но…”:

1) A[1,1] — “не забывай о поставленной задаче”;
2) A[1,5] — “правильно”;
3) A[5,5] — “с умом”;
4) A[5,1] — “не проще этого”.

8. Еще один принцип

Еще один принцип, которым всегда руководствовался наш герой, можно сформулировать так: “Инструмент должен соответствовать… Если инструмент не соответствует…, нужно придумать новый, который бы ей соответствовал, а не пытаться приспособить уже имеющийся”. Пропущенное дважды слово зависит от решения следующей задачи.

“Дан фрагмент программы, обрабатывающей двумерный массив *A* размером 5 × 5:

```
нц для n от 1 до 5
  нц для m от 1 до 5
    A[n, m] := (m - n) * (m - n)
  кц
кц
```

Сколько элементов массива *A* будут равны 1?”.

Возможные варианты (в зависимости от ответа):

1) 8 — “задаче”;
2) 2 — “идее”;
3) 5 — “принципу”;
4) 14 — “цели”.

Мальчики и мячи

Пять мальчиков — Петя, Вася, Джамал, Митя и Рашид — решили купить себе мячи. Мячи продавались разного цвета: красного, синего, белого, зеленого и черного. Известно, что:

1) Петя любит цвета, являющиеся базовыми для формирования цветной точки на экране;
2) Мите понравились синий и зеленый мячи;
3) Джамал купил мяч, цвет которого совпадает с цветом разрешающего сигнала светофора;
4) Рашид захотел купить красный, синий или черный мяч.

Кто какой мяч купил, если у всех они оказались разного цвета?

Кодирование чисел

Для кодирования натуральных чисел с помощью буквенных последовательностей был предложен следующий принцип, основанный на использовании латинских букв: A, B, C и D.

Числам 1, 2, 3 и 4 ставятся в соответствие указанные четыре буквы. Последующим 16 числам ставятся в соответствие двухбуквенные коды в следующем порядке: 5 = AA, 6 = AB, 7 = AC, 8 = AD, 9 = BA, 10 = BB, …, 18 = DB, 19 = DC, 20 = DD. Аналогично для последующих чисел используются трехбуквенные коды (от 21 = AAA до 84 = DDD), четырехбуквенные и т.д. Укажите буквенный код числа 295.

В ответе нужно записать последовательность из латинских букв. Решение получите, не выписывая все 295 чисел. Вспомогательные расчеты желательно провести с помощью электронной таблицы Microsoft Excel или др.

Литература

1. *www.diofant.ru/problem/2251* (автор задачи Сергей Котляров).

Мальчики вырезают фигуры

Коля, Витя, Митя, Глеб и Алеша вырезали из бумаги разные фигуры: круг из бумаги в клетку, круг из бумаги в линейку, квадрат из бумаги в клетку, квадрат из бумаги в линейку, флажок из белой бумаги. Глеб и Витя вырезали круги, Глеб и Коля вырезали из бумаги в клетку, Коля и Митя вырезали квадраты. Кто какие фигуры вырезал?

Задание предназначено для учеников начальной школы и учащихся 5–7-х классов.

На эскалаторе

Два человека идут по ступеням эскалатора (метро или магазина). Один идет быстрее другого. Кто из них насчитает больше ступенек или количество подсчитанных ступенек будет у них одинаковым?

Покупка дисков

Джон пошел в супермаркет за дисками. Один диск стоит 1 доллар, но при приобретении *X* дисков ($X < 100$) дается скидка *X* %. Когда он пришел домой с покупкой, брат сказал ему: “Ты заплатил за диски наибольшую возможную сумму денег!”. Сколько дисков купил Джон?

Задачу решите, разработав компьютерную программу или/и использовав электронную таблицу Microsoft Excel или др. 11-классникам предлагаем также решить задачу аналитически. Результаты присылайте в редакцию.

Ложное высказывание

Каково наибольшее целое число *A*, при котором ложно высказывание

$$(A \cdot (A + 1) > 55) \rightarrow (A \cdot A > 50)?$$

Символы "·" "+" и "→" — знаки логических операций, 1 — логическая константа, соответствующая истине.

Коля едет к Васе на лифте

В доме 100 этажей. Вася живет на 19-м, а Коля — на 96-м этаже. Лифт в доме имеет только две кнопки: "+7" (подняться на 7 этажей) и "–9" (опуститься на 9 этажей). За какое минимальное количество нажатий на кнопки Коля может попасть к Васе на лифте?

Задача подготовлена по материалам сайта "Diofant.ru" (www.diofant.ru/problem/2259).

Номера автобусов

Через поселок проходят три автобусных маршрута, их номера — трехзначные числа, причем все они являются точными квадратами. Более того, они записываются одними и теми же цифрами. Какие номера у автобусов? Вспомогательные расчеты проведите с помощью электронной таблицы Microsoft Excel или др. Мы же приведем две функции Microsoft Excel, которые понадобятся при выделении цифр натурального числа:

1) ОСТАТ — возвращает остаток от деления своего первого аргумента на второй;

2) ЦЕЛОЕ — возвращает целую часть своего аргумента.

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

Задача "Три музыканта"

Напомним, что следовало определить, на каких инструментах (скрипке, флейте, альте, кларнете, гобое и трубе) играет каждый из трех музыкантов — Браун, Смит и Вессон, если каждый владеет двумя инструментами и при этом известно, что:

1) Смит самый высокий;

2) играющий на скрипке меньше ростом играющего на флейте;

3) играющие на скрипке и флейте и Браун любят пиццу;

4) когда между альтистом и трубачом возникает ссора, Смит мирит их;

5) Браун не умеет играть ни на трубе, ни на гобое.

Решение

Составим таблицу и отразим в ней информацию из условия задачи, заполнив соответствующие клетки цифрами 0 и 1 в зависимости от того, ложно или истинно соответствующее высказывание.

Так как музыкантов трое, инструментов шесть, и каждый владеет только двумя инструментами, получается, что каждый музыкант играет на инструментах, которыми остальные не владеют.

Из факта 4 следует, что Смит не играет ни на альте, ни на трубе, а из фактов 3 и 5 — что Браун не умеет играть на скрипке, флейте, трубе и гобое. Следовательно, инструменты Брауна — альт и кларнет. Занесем это в таблицу, а оставшиеся клетки столбцов "альт" и "кларнет" заполним нулями:

	Скрипка	Флейта	Альт	Кларнет	Гобой	Труба
Браун	0	0	1	1	0	0
Смит			0	0		0
Вессон			0	0		

Из таблицы видно, что на трубе может играть только Вессон.

Из фактов 1 и 2 следует, что Смит не скрипач. Так как на скрипке не играет ни Браун, ни Смит, то скрипачом является Вессон. Оба инструмента, на которых играет Вессон, теперь определены, поэтому остальные клетки строки "Вессон" можно заполнить нулями:

	Скрипка	Флейта	Альт	Кларнет	Гобой	Труба
Браун	0	0	1	1	0	0
Смит	0		0	0		0
Вессон	1	0	0	0	0	1

Из таблицы видно, что играть на флейте и на гобое может только Смит.

	Скрипка	Флейта	Альт	Кларнет	Гобой	Труба
Браун	0	0	1	1	0	0
Смит	0	1	0	0	1	0
Вессон	1	0	0	0	0	1

Ответ: Браун играет на альте и кларнете, Смит — на флейте и гобое, Вессон — на скрипке и трубе.

Правильные ответы представили:

— Абросимов Денис, Алексеев Николай, Ашурко Анна, Байбабаева Надирахон, Богомолова Елизавета, Вожакова Мария, Васильев Глеб, Гловацкая Виктория, Дуплий Дмитрий, Евдокимова Анна, Ежук Мария, Елина Александра, Запасник Алена, Зунда Ксения, Иванов Глеб, Иванова Александра, Калинин Даниил, Карпенко Кристина, Ковалева Ольга, Королев Кирилл, Корхалева Любовь, Костева Дарья, Лунина Валерия, Малышев Никита, Матвееев Сергей, Москвин Даниил, Окунева Елизавета, Пахоменко Елизавета, Песковая Юлия, Пименова Светлана, Попова Мадина, Почесуй Ксения, Рогалевич Никита, Семенова Анастасия, Ситников Данил, Субоч Дарья, Туху Виталий, Царева Анна, Чепайкина Анастасия, Шаблыко Даниил и Юпилайнен Дарья, Республика Карелия, г. Сегежа, школа № 5, учитель **Меньшиков В.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.**;

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

— Борисова Вероника, Джишкариани Илья, Королев Никита, Коротина Татьяна, Кузнецов Максим, Майорова Ирина, Малыхин Александр, Матвеева Дарья, Неверов Андрей, Соколов Сергей и Халазий Виктор, Московская обл., г. Наро-Фоминск, школа № 1, учитель **Павлова Н.Н.**;

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

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

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

— Галихайдаров Азат и Гималова Алина, средняя школа села Новое Барятино, Республика Башкортостан, Стерлитамакский р-н, учитель **Евдокимова Н.Л.**;

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

— Григоренко Дмитрий, Круглякова Мария и Леоненко Степан, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

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

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

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

— Ларюшкина Светлана и Хомутова Наталья, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Кошелева Анастасия и Никонова Валентина, Куминская средняя школа, Тюменская область, Ханты-Мансийский автономный округ — Югра, Кондинский р-н, учитель **Шишигина О.В.**;

— Поляков Илья, Совхозная средняя школа, Московская обл., Серебряно-Прудский р-н, поселок Успенский, учитель **Жарикова Е.Н.**;

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

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

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

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

— Шаяхметова Татьяна, МБОУ "Кадетская школа", Республика Татарстан, г. Чистополь, пгт Крутая Гора, учитель **Валиева Р.Н.**

Отметим ответы читателей из школы № 5 г. Сегежа (учитель **Меньшиков В.В.**), представивших красочные презентации Microsoft PowerPoint.

Четыре вопроса (рубрика "Поиск информации")

Ответы

1. Чемпион мира по шахматам по имени Хосе (Хосе Рауль Капабланка) родился в г. Гавана, Куба.

2. Писатель, имя которого — Николай (Николай Карамзин), ввел в российскую письменность тире.

3. Крупнейшая во Франции трамвайная сеть находится в г. Нант.

В части перечисленных ответов все читатели были единодушны. А вот в ответе на вопрос № 4: "Какой фрукт тайцы перед съедением посыпают перцем и солью?" — мнения разделились. Указаны: манго (чаще всего), ананас, гуава, папайя, дуриан, джекфрут, помело и даже… арбуз ☺.

Ответы представили:

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

— 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.**;

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

— Гималова Алина, средняя школа села Новое Барятино, Республика Башкортостан, Стерлитамакский р-н, учитель **Евдокимова Н.Л.**;

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

— Димакова Арина, Ермакова Анастасия, Кононенко Александра, Коркина Анна, Молева Александра, Подлесных Константин, Пуговкина Ксения, Рухтин Дмитрий, Танасюк Артем и Рябинина Полина, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Жуков Александр, Ямало-Ненецкий автономный округ, Пуровский р-н, поселок Сывдарма, школа № 2, учитель **Кузьмич Ю.А.**;

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

— Костина Александра и Назаруха Ирина, Совхозная средняя школа, Московская обл., Серебряно-Прудский р-н, поселок Успенский, учитель **Жарикова Е.Н.**;

— Малых Кирилл, в прошлом учебном году — ученик 1-го (!) класса Куминской средней школы, Тюменская область, Ханты-Мансийский автономный округ — Югра, Кондинский р-н, учитель **Шишигина О.В.**;

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

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

— Минаева Екатерина, Романова Олеся и Томилов Дмитрий, Московская обл., г. Наро-Фоминск, школа № 1, учитель **Павлова Н.Н.**;

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

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

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

— Сертаков Игорь, г. Воронеж, лицей № 2, учитель **Комбарова С.И.**;

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

— Шаяхметова Татьяна, МБОУ "Кадетская школа", Республика Татарстан, г. Чистополь, поселок Крутая Гора, учитель **Валиева Р.Н.**

Головоломка "Азбука информатики в пословицах, поговорках и афоризмах"

Напомним, что необходимо было в приведенных пословицах и поговорках найти термин, связанный с информатикой и ИКТ, который образован несколькими последовательно идущими буквами.

Ответы. Абак. Бит. Выбор. Гнездо. Дорожка. Если. Железо. Запрос. Истина. Канал. Лупа. Метки. Начало. Образец. Паутина. Растр. Сбор. Тип (или "то"). Узел. Фон. Хит. Цвет. Чат. Шум. Эхо. Юзер. Ящик.

Ответы представили:

— Аветисян Мариам, Совхозная средняя школа, Московская обл., Серебряно-Прудский р-н, поселок Успенский, учитель **Жарикова Е.Н.**;

— Андрющенко Александр и Свистунов Николай, Ставропольский край, Кочубеевский р-н, станица Барсуковская, школа № 6, учитель **Рябченко Н.Р.**;

— 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.**;

— Борисова Вероника, Королев Никита, Коротина Татьяна и Матвеева Дарья, Московская обл., г. Наро-Фоминск, школа № 1, учитель **Павлова Н.Н.**;

— Бунеева Ксения, Буркова Екатерина, Долгов Роман, Киселева Ксения, Кожевникова Елизавета, Кудинова Юлия, Лесных Любовь, Мохов Сергей, Невструев Дмитрий и Соловцов Алексей, г. Воронеж, лицей № 2, учитель **Комбарова С.И.**;

— Галихайдаров Азат, средняя школа села Новое Барятино, Республика Башкортостан, Стерлитамакский р-н, учитель **Евдокимова Н.Л.**;

— Димакова Арина, Ермакова Анастасия, Кононенко Александра, Молева Александра, Подлесных Константин, Рухтин Дмитрий, Рябинина Полина и Танасюк Артем, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

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

— Егорова Туйаара, Корякина Сайыына и Хабибуллина Диляра, средняя школа села Устье, Республика Саха (Якутия), Сунтарский р-н, учитель **Котельникова Л.И.**;

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

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

— Назаров Иван и Прокоп Сергей, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

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

— Однокозова Анна и Заседателев Никита, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Васнина О.В.**;

— Перевощиков Даниил, Республика Коми, г. Сыктывкар, школа № 27, учитель **Суровцева Ю.Б.**;

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

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

Кроссворд, опубликованный в выпуске "В мир информатики" № 173 ("Информатика" № 2/2012)

Ответы

По горизонтали: 3. Исток. 6. Проход. 8. Описание. 13. Папка. 14. Набор. 16. Алгоритм. 21. Курсор. 22. Графа.

По вертикали: 1. Спам. 2. Лого. 4. Скан. 5. Код. 7. Диск. 9. Пробел. 10. Импорт. 11. Окно. 12. База. 15. Трек. 17. Граф. 18. Мост. 19. Тире. 20. Тег.

Ответы прислали:

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

— Андрющенко Александр и Свистунов Николай, Ставропольский край, Кочубеевский р-н, станица Барсуковская, школа № 6, учитель **Рябченко Н.Р.**;

— Ахматгалиева Диана, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Ахмедова Диана и Макарова Алена, Нармонская средняя школа, Республика Татарстан, Лаишевский р-н, учитель **Ожмекова Ю.Г.**;

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

— Воднева Кристина, Суроватихинская средняя школа, Нижегородская обл., Дальне-Константиновский р-н, учитель **Салова Т.В.**;

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

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

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

— Кашбиев Ильфат, Республика Татарстан, Актанышский р-н, село Актаныш, средняя школа № 2, учитель **Гилязова Г.М.**;

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

— Максимов Александр и Добрецов Иван, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Васнина О.В.**;

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

— Назаров Иван, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Перевощиков Даниил, Республика Коми, г. Сыктывкар, школа № 27, учитель **Суровцева Ю.Б.**;

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

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

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

Четыре задачи на системы счисления

Напомним, что предлагалось решить четыре задачи.

1. В какой системе счисления справедливо равенство $3 \times 4 = 10$?

2. Существует ли система счисления, в которой одновременно:

а) $3 + 4 = 10$ и $3 \times 4 = 15$;

б) $3 + 2 = 5$ и $3 \times 2 = 11$?

3. На доске сохранилась полустертая запись:

$$\begin{array}{cccccc} + & 2 & 3 & & 5 & \\ & 1 & & 6 & 4 & 2 \\ \hline & 4 & 2 & 4 & 2 & 3 \end{array}$$

Восстановите слагаемые.

4. Один школьный учитель информатики написал: “У меня в классе 100 детей, из них 24 мальчика и 32 девочки”. Возможно ли это?

Ответы

1. В 12-ричной.

2.

а) да, существует — семеричная система счисления;

б) нет, не существует — второе равенство возможно только в пятеричной системе счисления, но в этой системе нет цифры 5, присутствующей в первом равенстве.

3.

Последняя цифра первого слагаемого равна 1. Сумма чисел 5 и 4 может оканчиваться на 2 только в семеричной системе счисления. Весь пример:

$$\begin{array}{cccccc} + & 2 & 3 & 4 & 5 & 1 \\ & 1 & 5 & 6 & 4 & 2 \\ \hline & 4 & 2 & 4 & 2 & 3 \end{array}$$

4. Это возможно в шестеричной системе счисления.

Ответы представили:

— Адамова Екатерина и Порошенко Игорь, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

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

— Антохин Алексей, Борисова Вероника, Джишкариани Илья, Дутов Стас, Ельников Роман, Климко Женя, Королев Никита, Коротина Татьяна, Майорова Ира, Матвеева Дарья, Неверов Андрей, Соколов Сергей, Халазий Виктор и Шумилова Олеся, Московская обл., г. Наро-Фоминск, школа № 1, учитель **Павлова Н.Н.**;

— Ахматгалиева Диана, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

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

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

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

— Галихайдаров Азат, средняя школа села Новое Барятино, Республика Башкортостан, Стерлитамакский р-н, учитель **Евдокимова Н.Л.**;

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

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

— Демидов Андрей, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Зарипова Регина и Фасыхова Гульназ, Нармонская средняя школа, Республика Татарстан, Лаишевский р-н, учитель **Ожмекова Ю.Г.**;

— Ильин Иван, село Комсомольское, Республика Чувашия, школа № 1, учитель **Родионов П.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.**;

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

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

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

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

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

Обратим внимание читателей на ответ к заданию 2.

Крепкий орешек

Напомним, что в этой рубрике проводится анализ задач, решение которых вызвало трудности.

Сначала приведем списки читателей, приславших ответы на непростые задания:

1) задача “Шахматный турнир”:

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

— Зуйков Денис, Куклев Константин и Синицын Никита, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Новиков Владислав, Республика Башкортостан, г. Уфа, лицей № 60, учитель **Гильзер Н.В.**;

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

2) задача “Числа на доске”:

— Базылев Юрий, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Богданова Л.М.** (Юрий получил ответ, разработав компьютерную программу);

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

— Куклев Константин, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.** (Константин разработал два варианта компьютерной программы);

— Новиков Владислав, Республика Башкортостан, г. Уфа, лицей № 60, учитель **Гильзер Н.В.**;

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

— Решетников Виталий, Вадьковская средняя школа, Брянская обл., Погарский р-н, учитель **Цыганкова И.Ю.** (Виталий также решил задачу с помощью компьютерной программы);

3) числовой ребус с неизвестным числом:

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

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

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

— Иванова Дарья и Петрова Ольга, г. Москва, филиал школы № 227, учитель **Уколова О.А.**;

— Шуднев Олег, Ставропольский край, г. Пятигорск, школа № 28, учитель **Ильина Н.И.**

Теперь анализ.

Задача “Числа на доске”

В задаче было необходимо (методом рассуждений или разработав компьютерную программу) определить, может ли через некоторое время на доске появиться число 123456, если сначала на ней было написано число 1, а каждую секунду к числу на доске прибавляется сумма его цифр.

Начало решения

Анализ первых чисел, которые последовательно будут выписаны на доске — 1, 2, 4, 8, 16, 23, 28, 38, …, не выявляет какую-либо закономерность, поэтому попробуем исследовать задачу, так сказать, с конца.

Каким могло быть число, предшествовавшее числу 123456?

Во-первых, оно не могло быть пятизначным, так как 99999 (максимальное пятизначное число) плюс 45 (сумма его цифр) меньше, чем 123456.

Во-вторых, оно не может быть равно 123399, поскольку в этом случае очередное число не равно “нашему”.

Итак, можем утверждать, что предпоследним являлось число вида $1234ab$. Сумма его цифр равна $10 + a + b$, поэтому $1234ab + 10 + a + b = 123456$. Запишем последнее равенство так:

$$123400 + ab + 10 + a + b = 123456,$$

где ab — двузначное число, или $ab + a + b = 46$. Так как число ab можно представить как $10a + b$, то имеем: $11a + 2b = 46$. Проверка возможных целых значений a ($a \le 4$) показывает, что решением являются значения $a = 4$, $b = 1$.

Итак, число 123456 на доске может быть написано после числа 123441. А может ли быть получено число 123441?

Дальнейший (аналогичный) анализ проведите самостоятельно.

Задача “Шахматный турнир”

Напомним, что было необходимо заполнить турнирную таблицу, установив, как сыграл каждый шахматист со своими соперниками, по известной сумме очков, набранных игроками *А*, *Б*, *В* и *Д*:

	А	**Б**	**В**	**Г**	**Д**	**Очки**
А	X					3,5
Б		X				1,5
В			X			2,5
Г				X		
Д					X	2,5

Турнир проводился в один круг. Известно также, что участник *Д* не имел ни одного поражения.

Решение

1. При пяти участниках, каждый из которых сыграл с каждым по одной партии, общее число очков, набранных всеми игроками, равно $5\times4/2 = 10$. Значит, игрок *Г* не набрал ни одного очка.

2. По условию, игрок *Д* не имел поражений. Значит, игрок *А*, набравший 3,5 очка, с ним сыграл вничью, а у остальных выиграл:

	А	**Б**	**В**	**Г**	**Д**	**Очки**
А	X	1	1	1	0,5	3,5
Б	0	X		1		1,5
В	0		X	1		2,5
Г	0	0	0	X	0	0
Д	0,5			1	X	2,5

3. Далее следует исследовать возможные результаты двух оставшихся встреч игрока *В*.

Соответствующий анализ проведите самостоятельно.

Числовой ребус с неизвестным числом

Напомним, что требовалось решить числовой ребус:

	A	**B**	**C**	**D**
+	**D**	**C**	**B**	**A**
	X	**X**	**X**	**X**
1	2	3	0	0

в котором **A**, **B**, **C**, **D** — четыре последовательные цифры, идущие в порядке возрастания. Четыре буквы **X** — это те же самые четыре цифры в неизвестном порядке.

Чтобы получить решение, необходимо рассмотреть возможные варианты значения последнего числа **X** с учетом того, что **D** = **A** + 3 (это следует из особенностей цифр, указанных в условии) и что сумма цифр в крайнем правом разряде равна 10 или 20.

Пример анализа

Например, пусть **X** = **A**. В этом случае, подставив указанные выше значения **D**, имеем:

$$\mathbf{A} + \mathbf{A} + 3 + \mathbf{A} = 10;\ 3\mathbf{A} + 3 = 10$$

или

$$\mathbf{A} + \mathbf{A} + 3 + \mathbf{A} = 20;\ 3\mathbf{A} + 3 = 20.$$

В обоих случаях подходящие значения **A** отсутствуют.

Дальнейший анализ проведите самостоятельно.

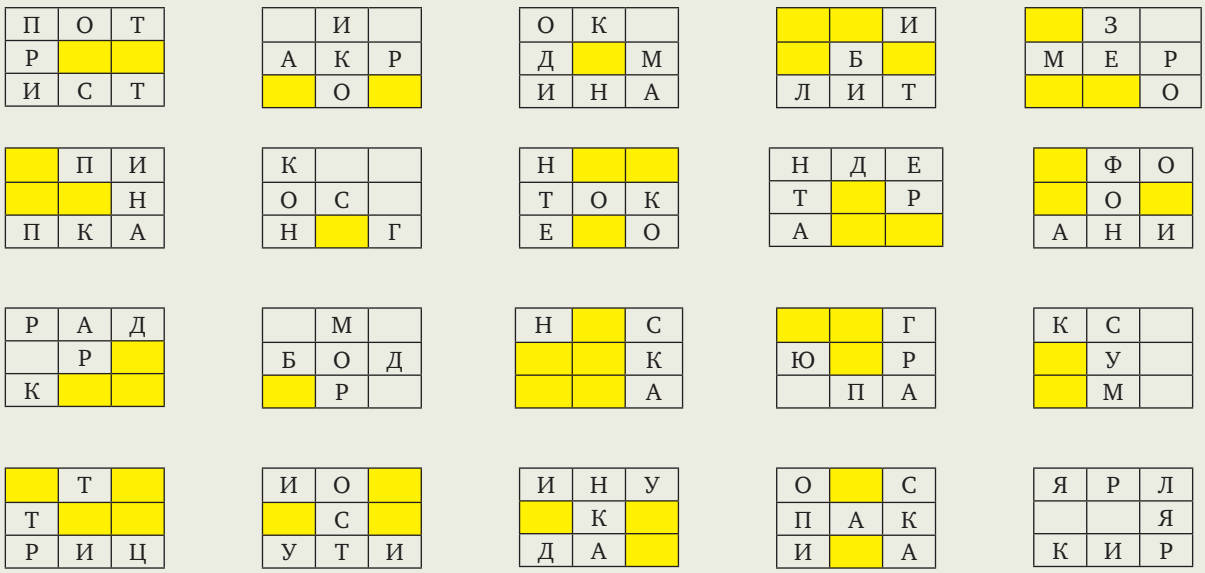
Ответы на все задания присылайте в редакцию. Подумайте также над вопросом, можно ли решить последний ребус, если система счисления, в которой записаны числа, неизвестна?

ВНИМАНИЕ! КОНКУРС

Конкурс № 93 “Мозаика для кроссворда”

Соберите разрезанный на части кроссворд размером 15 на 15 клеток, в котором почти все записанные слова связаны с информатикой и компьютерами. Желтым цветом закрашены пустые “внутренние” клетки. Незакрашенные клетки без букв — наружные.

Какие бы вы задали вопросы к каждому из слов кроссворда, если бы составляли его?



Р	М	А
	А	
Е		Ш

	А	
Е	Т	
	О	Т

И		Е
		Н
И	Ц	А

С		Е
Т	А	Й
А		М

Ы	К	
И	Л	Л

Ответы отправьте в редакцию до 25 сентября 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.

Срок представления ответов на другие задания, предложенные в этом выпуске, — 15 сентября.

Дорогие ребята! Присылайте ответы на вопросы наших конкурсов, даже если по каким-то причинам (поздняя доставка журнала или др.) вы подготовите их позже срока, указанного в условии конкурса. Мы обязательно рассмотрим все присланные ответы и при необходимости дополнительно отметим лучшие из них.

При оформлении ответов будьте, пожалуйста, внимательны — указывайте все необходимые сведения о себе (имя, пожалуйста, приводите полностью), населенном пункте, учебном заведении и учителе информатики.

GAMES.EXE

Компьютерная игра “Поиск сокровищ”

В этой статье мы опишем методику разработки программы, моделирующей игру с такими правилами. В одной из точек игрового поля (позиций экрана) спрятан клад. Играющий условно представлен на экране в виде курсора и может перемещаться по полю с помощью клавиш со стрелками. После каждого перемещения на табло, расположенном в правом верхнем углу экрана, показывается расстояние от играющего до клада. Необходимо найти клад за минимальное число перемещений.

Общая схема программы:

1. Задание местонахождения клада и исходного положения играющего.

2.

цикл

Распознаем нажатую клавишу

выбор

при *нажата клавиша “Стрелка вверх”*:

Смещаем курсор на одну строку вверх

Выводим информацию на табло

при *нажата клавиша “Стрелка вниз”*:

…

при *нажата клавиша “Стрелка вправо”*:

…

при *нажата клавиша “Стрелка влево”*:

…

все

конец цикла при *клад найден*

3. Вывод на экран информации об окончании игры.

В программе на школьном алгоритмическом языке, который применим исключительно для описания методик моделирования игры, используем следующие основные величины:

Хкл — номер строки, на которой находится клад;

Укл — то же, номер столбца (позиции);

Хиг — номер строки, на которой находится играющий;

Уиг — то же, номер столбца (позиции);

Р — расстояние от играющего до клада.

Начнем с простого варианта — в котором местонахождение клада и исходное положение играющего задаются постоянными (конечно, в дальнейшем это надо будет изменить).

Примем следующие значения:

$Хкл = 60, Укл = 10, Хиг = 22, Уиг = 20$.

Значение $Р$ может быть определено с учетом того, что $Р^2 = (Хкл - Хиг)^2 + (Укл - Уиг)^2$.

Прежде чем представлять программу, обсудим одну проблему, связанную с выводом информации на табло (см. выше). Так как для этого придется использовать процедуру установки места вывода на экран, а затем потребуется вернуть курсор на место играющего, то перед выводом следует запомнить положение играющего. Его будем задавать с помощью переменных *ГдеХ* и *ГдеУ*.

Итак, программа:

```
алг Поиск_сокровищ
нач цел Хкл, Укл, Хиг, Уиг, ГдеХ, ГдеУ
  |Задание местонахождения клада
  |и исходного положения играющего
  Хкл := 60; Укл := 10
  Хиг := 22, Уиг := 20
  |Вывод информации на табло
  |Запоминаем текущее положение
```

```
  ГдеХ := ТекХ; ГдеУ := ТекУ;
  |Определяем значение R
   …
  |и выводим его
  УстанКурсора(70, 1)
  вывод R
  |Возвращаемся на поле
  УстанКурсора(ГдеХ, ГдеУ)
  нц
    |Распознаем нажатую клавишу
    выбор
      при нажата клавиша "Стрелка вверх":
        |Меняем положение курсора
        УстанКурсора(ТекХ, ТекУ - 1);
        |Запоминаем текущее положение
        ГдеХ := ТекХ; ГдеУ := ТекУ
        |Определяем значение R
       …
        |и выводим его
        УстанКурсора(70, 1)
        |Возвращаемся на поле
        УстанКурсора(ГдеХ, ГдеУ)
      при нажата клавиша "Стрелка вниз":
       …
    все
  кц_при ТекХ = Хкл и ТекУ = Укл
   Выводим информацию об окончании игры
  УстанКурсора(30, 12)
  |Приостановка программы до нажатия
  |любой клавиши
  нц
  кц_при клав > 0
кон
```

— где: *УстанКурсора* — процедура, устанавливающая курсор в некоторое место экрана*, *ТекХ* и *ТекУ* — функции, возвращающие, соответственно, номер столбца (позиции) и номер строки для текущего положения курсора. Аналоги этой процедуры и этих функций в языках программирования Паскаль и Бейсик приведены в таблице:

	Язык Паскаль	Язык Бейсик
УстанКурсора	Процедура GotoXY	Процедура LOCATE
ТекХ	Функция WhereX	Функция POS
ТекУ	Функция WhereY	Функция CSRLIN

Обратим внимание, что в языке Бейсик в процедуре LOCATE сначала указывается номер строки.

Теперь о самом сложном ☺. Фрагменты программ на Бейсике, с помощью которых можно отслеживать нажатие клавиш со стрелками, приведены ниже:

Язык Паскаль

```
c := readkey;
case c of
#72 {стрелка вверх}: begin
                       …
                     end;
#80 {стрелка вниз}:  begin
                       …
end;
#77 {стрелка влево}:   begin
                       …
                     end;
#75 {стрелка вправо}:  begin
                       …
                     end;
end;
```

Язык Бейсик

```
c$ = INKEY$
IF LEN(c$) = 1 THEN 'c$ содержит символ,
'распознаваемый по ASCII-коду.
SELECT CASE RIGHT$(c$, 1)
  'Клавиша "Стрелка вверх"
  CASE CHR$(72)
  …
  'Клавиша "Стрелка вниз"
  CASE CHR$(80)
  …
  'Клавиша "Стрелка влево"
  CASE CHR$(75)
  …
  'Клавиша "Стрелка вправо"
  CASE CHR$(77)
  …
END IF
```

И еще. Видно, что фрагмент, относящийся к выводу информации на табло, повторяется в программе 5 раз. Целесообразно оформить его в виде отдельной процедуры (вспомогательного алгоритма):

```
алг Табло
нач
  |Определяем значение R
  …
  |и выводим его
  УстанКурсора(70, 1)
  |Возвращаемся на поле
  УстанКурсора(ГдеХ, ГдеУ)
кон
```

Задание для самостоятельной работы

Разработайте программу (на языке программирования, который вы изучаете), отличающуюся одним или несколькими особенностями:

1) после нахождения клада в месте его расположения появляется мигающий цветной символ (например, “*”) и/или звучит звуковой сигнал;

2) в программе используется процедура *Табло*, описанная выше;

3) на табло показывается также количество сделанных шагов;

4) местонахождение клада и исходное положение играющего задаются случайным образом. При этом целесообразно “разнести” эти места по разным частям экрана (например, если клад находится в левой половине экрана, то исходное положение играющего должно быть в правой).

Разработанную программу присылайте в редакцию. Авторы лучших программ будут награждены дипломами.

* В школьном алгоритмическом языке такой процедуры нет (использована условно).

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»

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