№ 36 (229) сентябрь 1999

еженедельное
ПРИЛОЖЕНИЕ К ГАЗЕТЕ
«ПЕРВОЕ СЕНТЯБРЯ»

ИНФОРМАТИК

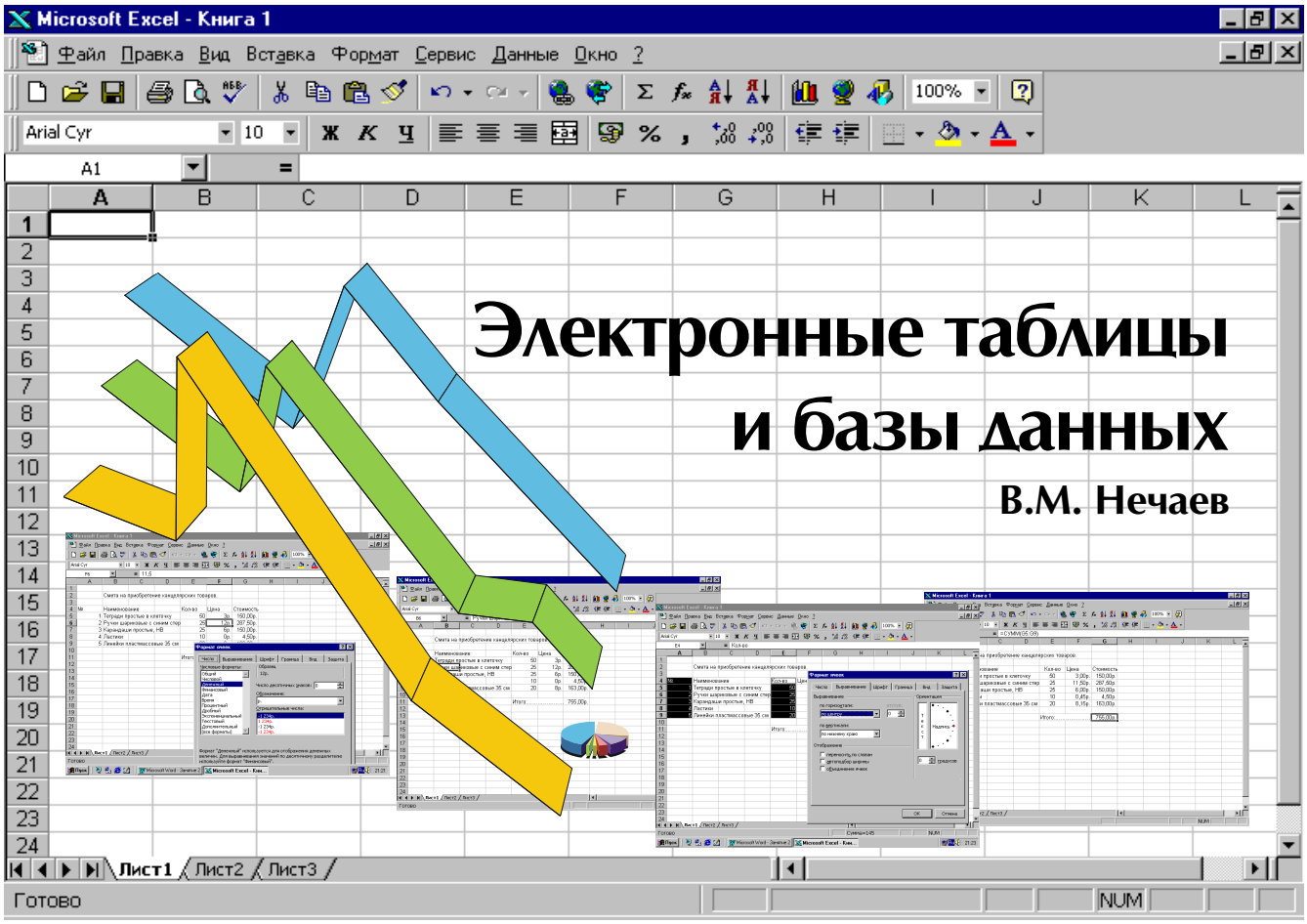
Не все так однозначно...

20 лет назад (в 1979 г.) появился большой международный журнал Fuzzy Sets and Systems, посвященный теории нечетких множеств, а 10 лет назад в Японии был создан

Международный институт нечеткой технологии

Подробнее о нечетких технологиях читайте на с. 32

Читайте в номере



От редакции

Вашему вниманию предлагается серия статей, основанная на материалах занятий одноименного дистанционного курса, который был проведен автором. Это “происхождение” наложило отпечаток на структуру серии. В частности, каждая статья соответствует одному занятию, в конце каждой статьи приводятся контрольные вопросы и задания, которые выполняли слушатели курса.

По нашему мнению, эти статьи, рассматриваемые в них задачи и задания могут быть полезны при планировании и проведении занятий, посвященных электронным таблицам и базам данных.

Содержание

Курс ориентирован на школьных учителей информатики и в части, касающейся электронных таблиц, преследует следующие три цели:

а) выработать у пользователя необходимые навыки создания электронных таблиц в системе Excel 97, проведения в них расчетов, а также соответствующего их оформления;

б) пополнить багаж примеров и заданий, которые учитель может предлагать ученикам на уроках информатики и для домашней работы;

в) расширить кругозор учителя информатики путем знакомства его с интересными фактами, касающимися смежных областей — физики, химии, астрономии, математики.

В пунктах **б** и **в** отражены отличия данного курса, причем в ходе занятий предусмотрено смещение акцента именно на содержательную сторону рассматриваемых задач, вопросы же технической реализации принимаемых решений постепенно отходят на второй план.

Структура

Всего будет представлено восемь занятий.

Первые четыре занятия образуют ***основной блок***. Его главное назначение — помочь учащемуся освоить разнообразные приемы, употребляемые при построении таблиц. Рассматриваются следующие вопросы:

— способы расположения текста в ячейках таблицы;
— применение разных форматов для исходных числовых данных;
— ввод формул в ячейки и использование в этих формулах встроенных функций;
— вложение функций (логических, статистических и т.д.) одной в другую;
— копирование содержимого ячеек и связанная с этим относительная и абсолютная адресация;
— приемы автозаполнения и создание списков;
— изменение ширины столбцов и высоты строк, разлиновка и заливка столбцов и строк;
— параметры страницы и ее предварительный просмотр, защита содержимого ячеек;
— построение диаграмм и графиков.

В качестве примеров здесь используются преимущественно обычные задачи из практики:

— составление денежной сметы на покупку товаров;
— расчет жилой площади городской квартиры и денежного дохода семьи;
— ведение баланса мелкооптовой торговли;
— ведение табеля успеваемости учащихся и ведомости на почасовую оплату преподавателей;
— определение победителей в спортивном и учебном турнирах.

Рассматриваются также некоторые вопросы, связанные с обработкой уже имеющихся данных (из разных областей науки):

— статистика метеорологических наблюдений;
— обзор значений атомного веса для химических элементов в таблице Менделеева;
— вычисление плотности вещества для планет Солнечной системы.

Далее идет ***дополнительный блок***, также состоящий из четырех занятий. Здесь необходимо уже довольно свободное владение арсеналом основных технических приемов — с тем, чтобы можно было сосредоточиться на существе задачи, а именно на ее постановке и разработке алгоритма решения.

Тематика примеров, использованных в этом блоке, лежит где-то на стыке математики, физики и информатики, причем очень важную роль в их решении будут играть случайные величины и операции их усреднения.

В частности, речь пойдет:

— об одном уникальном свойстве экспоненциальной функции и его природном проявлении;
— о способах сжатия двоичной информации и связанном с этим ее искажении.

Особую роль играют последние два занятия этого блока. Тут сам расчет является только средством для более серьезной деятельности, а именно исследования определенного природного явления, т.е. формирования подходящей математической модели и выявления на ее основе определенных закономерностей.

Примерами таких явлений будут служить:

— преломление света на границе двух прозрачных сред;
— падение тела на землю с учетом сопротивления воздуха.

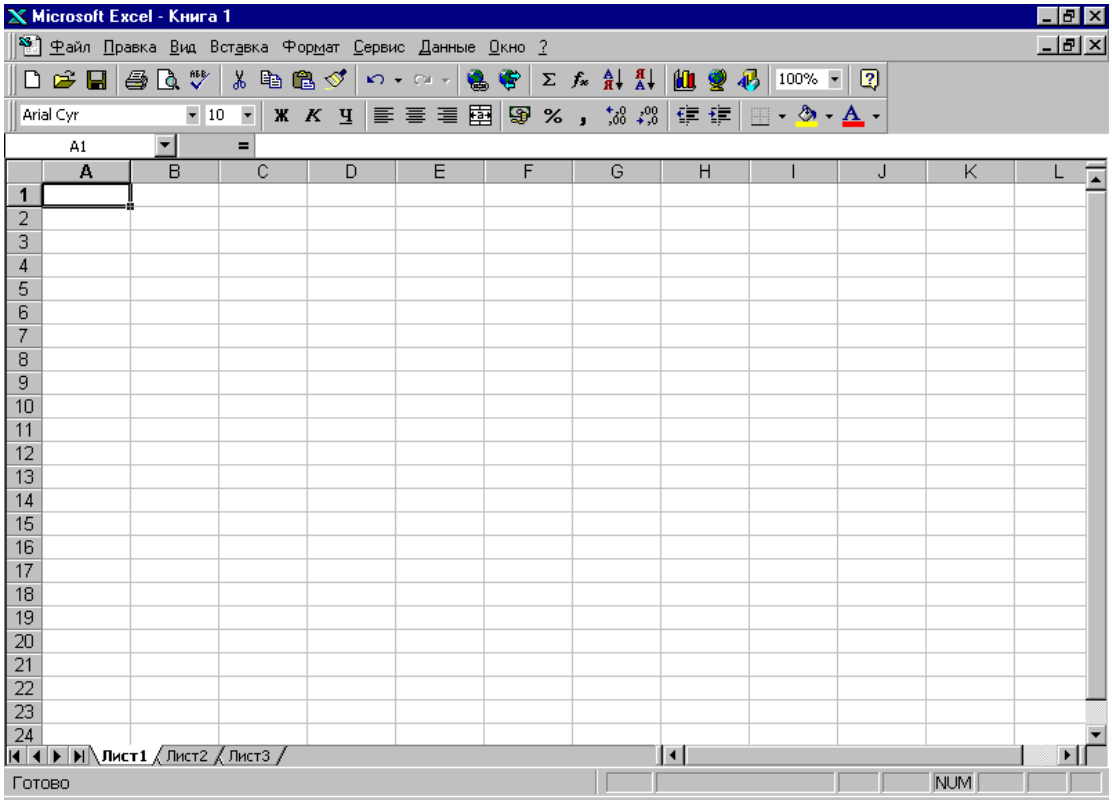
Часть курса, касающаяся ***баз данных***, призвана сформировать у слушателей общее представление о работе с документами соответствующего типа в среде Access 97 и помочь выработать одно конкретное умение — пересылать в базы данных информацию из электронных таблиц Excel, и наоборот — получать информацию для обработки из базы данных.

Будет описано одно занятие, на котором надо создать простую базу данных, предназначенную для хранения сведений о параметрах и рыночной стоимости компьютерных комплектующих. При этом она должна не только включать в себя таблицу хранящихся записей, но и обслуживать запросы на выборку фирм, торгующих компонентами указанного типа “в заданном ценовом диапазоне”. Кроме того, должен обеспечиваться обмен входными и выходными данными с документами ранее рассмотренного типа — электронными таблицами.

Занятие 1. Смета на покупку товаров

Столбец, строка, ячейка

Диалог пользователя с программой Microsoft Excel, как и у многих других программ, осуществляется с помощью окна. Обычно после запуска программы на выполнение оно занимает весь экран, а внутри него раскрыто еще одно — уже не окно программы, а окно документа, над которым в данный момент идет работа. Управлять размерами и того, и другого окна (свернуть до минимума, уменьшить до некоторого промежуточного состояния, закрыть вообще) можно с помощью соответствующих кнопочек в правом верхнем углу.



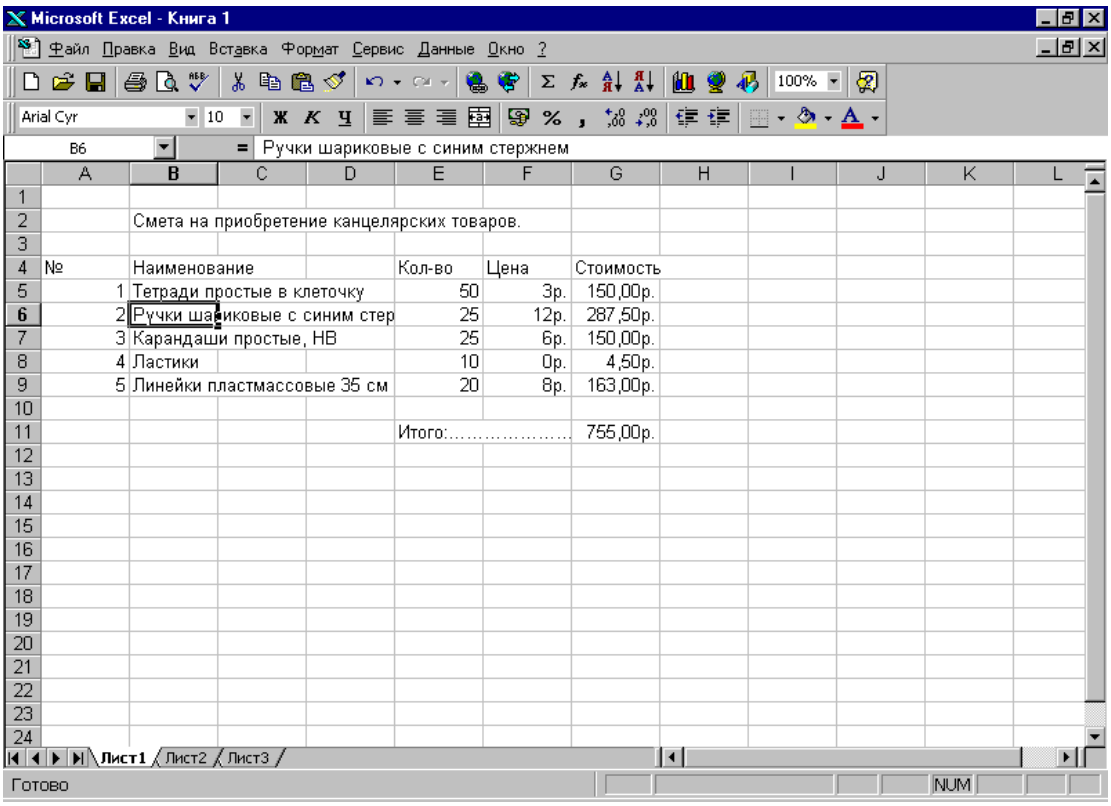
Заголовок внешнего (программного) окна выделен синим цветом и содержит имя документа `Книга 1`. Это имя временное, дежурное, оно автоматически дается всем новым документам, и по окончании занятия, когда электронная таблица будет заполнена и ее потребуется для хранения записать на диск, мы, естественно, заменим его настоящим именем — пусть это будет `Смета на покупку товара`. Но как бы ни назывался документ, все равно по своей сути это будет именно книга, т.е. набор нескольких отдельных страниц, или листов, наполненных разной информацией. Им тоже изначально даны дежурные наименования — `Лист1`, `Лист2`, `Лист3`, которые видны внизу. Такими их и можно оставить, хотя лучше все-таки подобрать что-нибудь, отражающее содержание каждого листа. Разумеется, и количество листов тоже можно изменить от одного, для простого документа, до нескольких десятков (если, конечно, такое потребуется).

Итак, книга состоит из рабочих листов. Лист, в свою очередь, разделен на множество прямоугольных ячеек, и каждая из них имеет свой адрес, которым определяется ее местоположение. Адрес образован двумя координатами: латинской буквой (горизонтальная ось координат) и числом (вертикальная ось координат). С учетом имени листа получается даже три измерения: глубина, ширина, высота — или, соответственно, лист, столбец, строка. Правда, часто бывает, что имя листа указывать нет нужды, допустим, если он всего один. Поэтому обычно ограничиваются только буквой столбца и номером строки, например, `F5` или `G8`. Когда же алфавит кончается, то для обозначения следующих столбцов используются уже две буквы; так, если дойти по какой-то строке, скажем по четвертой, до `X4`, `Y4`, `Z4`, то после этого пойдут ячейки `AA4`, `AB4`, `AC4` и т.д. — столбцов хватит для очень большой таблицы (строчек тоже очень много — несколько тысяч!).

Перемещение по ячейкам производится либо с помощью клавиш управления курсором (вправо-влево, вверх-вниз), либо, если так удобнее, посредством стандартных линеек прокрутки (справа внизу — горизонтальная, а сбоку — вертикальная). Надо только после того, как дойдешь до нужной тебе ячейки, не забыть щелкнуть по ней мышью, чтобы вокруг появилась рамочка, — вот тогда можно начинать с этой ячейкой работать. Так же и для перехода на другой лист, потребуется всего лишь щелкнуть по закладке с его именем.

Числа и текст

Адрес текущей ячейки, т.е. той, с которой мы в данный момент имеем дело, отображен слева, вверху (поначалу там было A1). Справа от адреса, в так называемой строке формул, показывается, что конкретно в данной ячейке записано в настоящий момент, какая информация (поначалу, очевидно, не было никакой). В любую ячейку можно вводить что-то новое или просматривать то, что в нее уже было введено ранее.



Ячейки могут содержать информацию нескольких типов:

- текстовую — для того чтобы указать, например, наименование какого-то товара;
- числовую — для того чтобы задать номер, количество единиц и цену для каждого наименования;
- и, наконец, это могут быть формулы, скажем, для вычисления полной стоимости покупки.

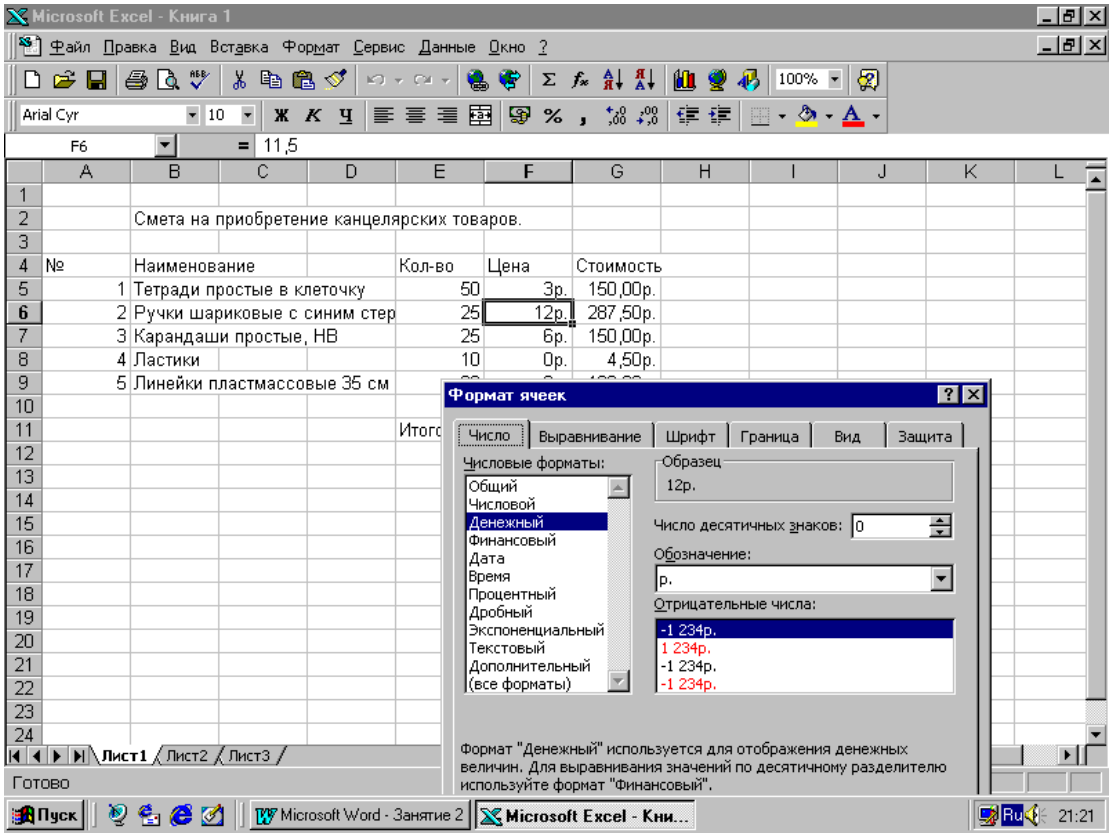
Следует обратить внимание, что содержимое ячейки, которое мы видим в строке формул, далеко не всегда совпадает с его отображением в самой таблице. Например, при вводе текста очень часто он весь не может войти в довольно узкие рамки своей клеточки. В таком случае непоместившаяся часть текста просто переходит в соседнюю ячейку, если только в ней ничего нет. На рисунке видно, что хотя предложение "Ручки шариковые с синим стержнем" все целиком записано по адресу B6 и хранится там (об этом свидетельствует строка формул), в самой таблице оно простирается лишь до границы E6, т.е. до первой ячейки, которая не пуста, а содержит какие-то другие данные. Если перевести рамочку с B6 вправо на один шаг (в C6) или два шага (в D6), то строка формул подтвердит, что там ничего нет, несмотря на то, что в таблице "поверх" этих ячеек виден текст. Все предложение полностью как бы "прописано" в B6, в соседних же клетках оно только отображается, и то лишь потому, что они оказались свободными.

А вот ячейка E6 — занята. В ней указано число 25. Легко, кстати, заметить, что все числа в столбце E выровнены по правому краю, в отличие от слов "Кол-во" и "Итого", выровненных по левому краю. Конечно, это неспроста. Здесь действует общее правило "по умолчанию", как его принято называть. Если мною не выражено в явной форме, с помощью той или иной команды форматирования, какое-нибудь особое пожелание (не только в смысле выравнивания, а вообще какое бы то ни было), то программа действует по своему усмотрению.

И почти всегда это оказывается наиболее подходящим вариантом. В самом деле, у чисел скорее всего единицы должны стоять под единицами, десятки под десятками, сотни под сотнями. Что же касается наименований товаров, то здесь наоборот, поскольку они разной длины, — начало каждого наименования желательно помещать под предыдущим. "Умолчание" — это очень удобно. Более того, уже сам способ выравнивания позволяет мне с первого взгляда на столбец F догадаться, что в нем, несмотря на букву "р.", содержатся не просто цифровые символы, а все-таки (раз они выровнены по правому краю) числа. И, значит, они могут использоваться для вычислений.

Формат числа

Однако может возникнуть вопрос: почему же у этих, как оказывается, чисел несколько отличный от соседей слева вид? Ведь не потому же, что над ними стоит слово “Цена”, хоть оно и относится по своему смыслу к деньгам, как известно, счет любящим. Да, конечно, это я сам, уже после ввода числовых данных, установил для ячеек с F5 по F9, в приказном порядке, выбранный мною формат. И, между прочим, не очень-то удачный.



Что-то не то выходит: 25 ручек, каждая из которых стоит по 12 рублей, будут стоить вовсе не 287 рублей 50 копеек; и без калькулятора ясно, что уж копеек здесь совсем не должно получаться, только целые рубли. Да и тех должно быть не 287, а 300! Неужели ошибка? В подобных ситуациях всегда обращаются к строке формул, только она одна сообщит полную правду о содержимом ячейки. Сразу становится видно, что на самом деле цена ручки не “12 р.”, а 11,5 — это просто дробное число, которое было введено с клавиатуры. А вот формат ему придали хоть и “денежный”, но “с числом десятичных знаков 0”. Ну и поскольку десятичных знаков ноль, то 11 с половиной округлилось и стало 12. Таким образом, в таблице просто оказалась неверно отображенной цена, а ошибки никакой нет.

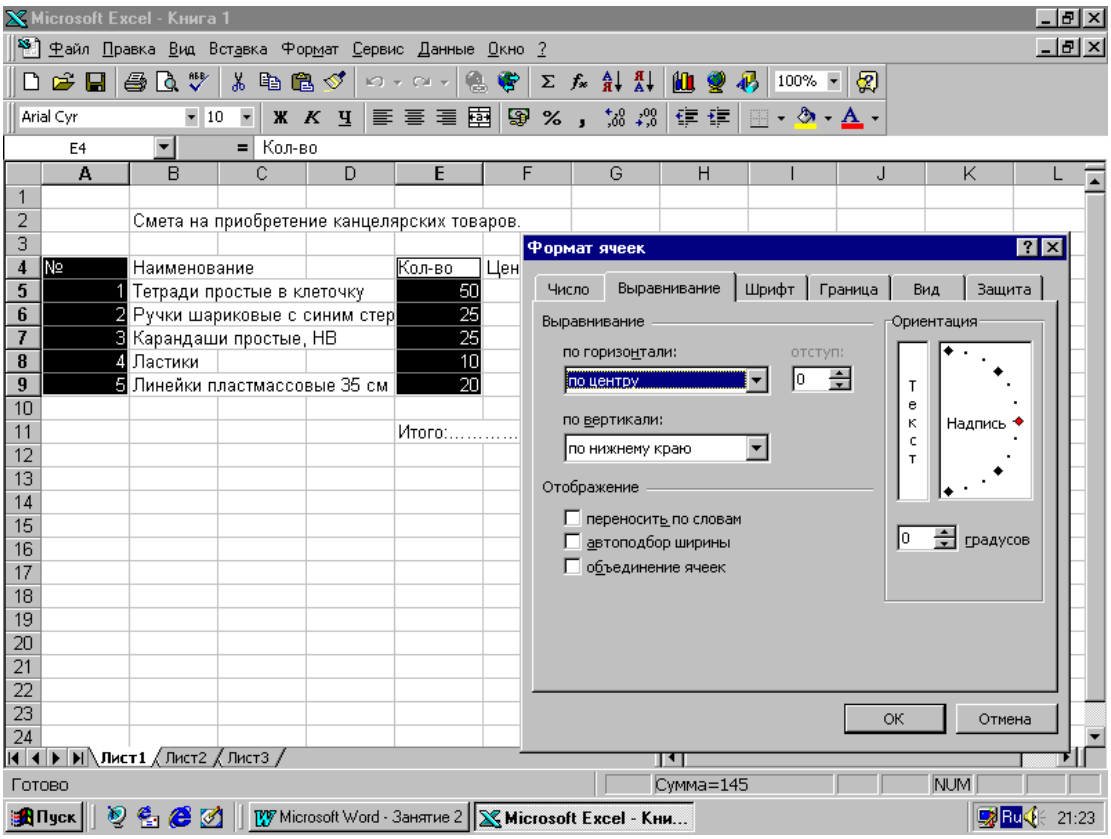
Неточность вкралась не в само число, а лишь в его представление на экране. Стоимость-то подсчитана правильно: 25 · 11,5 = 287,50 р. То есть в расчете участвовало как раз содержимое ячейки F6, а не ее видимый облик, оказавшийся обманчивым из-за неудачного выбора формата. Заказать следовало бы два десятичных знака, как это, видимо, было сделано для числовых ячеек соседнего столбца, с G5 по G11 включительно.

Еще интереснее эта “ошибка неподходящего формата” проявляется по отношению к ластикам. Там уж совсем получается абсурд — каждый ластик ничего не стоит, а за 10 штук тем не менее придется заплатить около пяти рублей. Сумма, конечно, небольшая, но дело принципа! Причина все в том же: фактически в ячейке F8 содержится число 0,45 и вполне благополучно участвует в вычислениях, давая правильный результат 0,45 · 10 = 4,50 р. Но в таблице оно представлено округленным до ближайшего целого, т.е. до нуля, и потому выглядит нелепо. Так что и для данной клеточки требуется установить точность до двух знаков после запятой.

Да, пожалуй, и для всех ячеек столбца F надо это сделать, даже для тех, в которых все вроде бы и так неплохо. Ничего, что тетради будут не по 3 р., а по 3,00 р. — так даже лучше, потому что для таблиц вообще предпочтительнее единообразие: уж если в одном месте надо с копейками, то и везде пусть будет с копейками. Причем одинаковый формат желательно установить сразу для всех тех ячеек, где он требуется, за один прием. То есть можно, конечно, выбирать и по одной ячейке, для каждой выполняя один и тот же набор действий, но удобнее, естественно, для всех сразу дать общую команду.

Выделение ячеек

Для задания какого-либо формата, как, впрочем, и для выполнения любой другой команды, необходимо сначала выделить нужные ячейки. Часто бывает, что они расположены рядом друг с другом, образуя прямоугольную область шириной в несколько столбцов (или в один столбец) и высотой в несколько строк (или в одну строку). Тогда по ним проводят мышью с нажатой левой кнопкой (ячейки при этом закрашиваются в черный цвет). Или употребляют другой способ, без мыши: расположив рамочку в левой верхней ячейке, начинают смещать ее с помощью клавиши управления курсором (→ ↓) до противоположного угла, удерживая при этом клавишу Shift (левую или правую, все равно) нажатой. Эффект будет тот же самый — зачернение выделяемой области. Если на первых порах будет случайно выделено что-нибудь лишнее, то следует, отпустив все клавиши, щелкнуть по какой-либо пустой клетке, а потом повторить попытку.



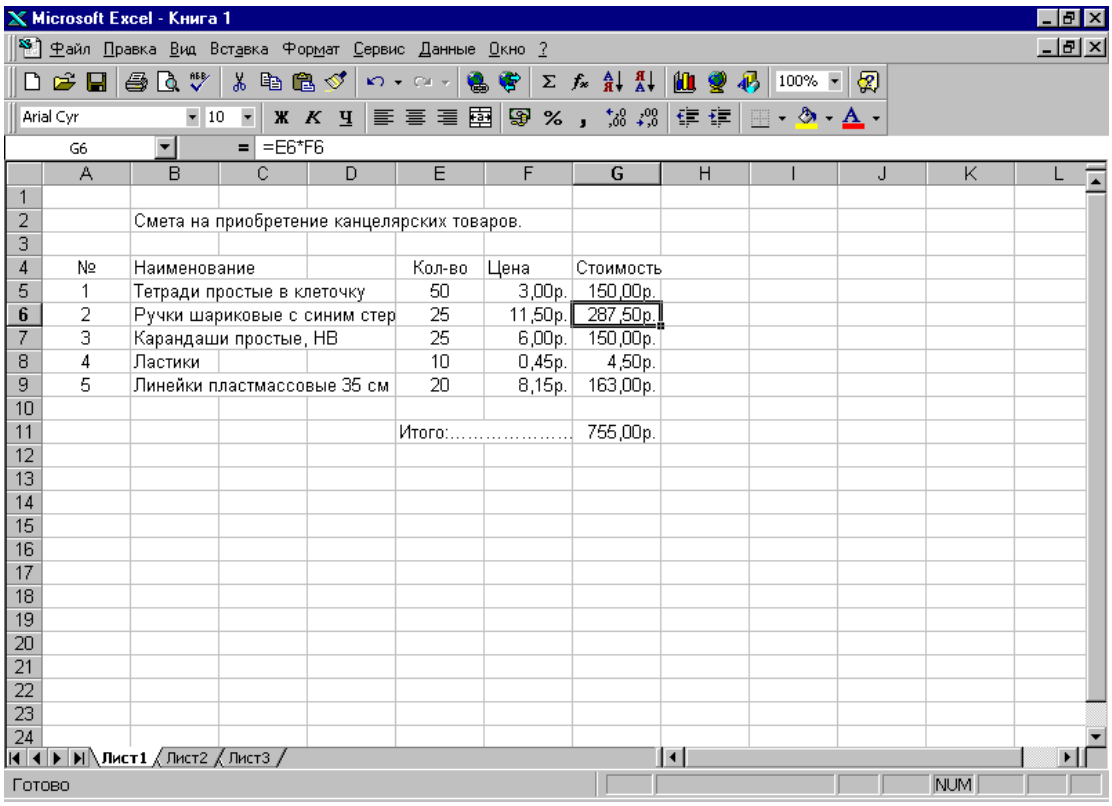
Теперь из меню Формат можно уже выбирать мышью, не снимая, конечно, выделения, пункт Ячейки... и в появившейся карточке (правильно говорить — в диалоговом окне) раскрывать одну из шести закладок. Например, закладку Число, о которой шла речь выше, или как сейчас, когда захотелось расположить данные о количестве товара симметрично, — Выравнивание. А заодно надо отцентрировать и порядковый номер. (Символ № получается при использовании комбинации клавиш Shift + 3 на русской раскладке клавиатуры.) Две несмежные области (или большее их количество) удается пометить черным для общей команды посредством удерживания (дополнительно) клавиши Ctrl при выделении отдельных фрагментов.

Многие и, что особенно ценно, наиболее часто употребляемые команды не обязательно выбирать из меню, есть более быстрый и удобный способ обращения к ним (но опять-таки после того, как нужные ячейки выделены) — через панели инструментов в верхней части окна. Панелей вообще-то имеется много, но изначально предлагаются две: Стандартная (повыше) и Форматирование. Остальные легко заказать в любой момент, стоит лишь щелкнуть по меню Вид и выбрать соответствующий пункт, но в принципе, кроме этих двух, пожалуй, ничего и не надо.

Вот, скажем, на панели форматирования имеются кнопочки и для увеличения или уменьшения разрядности числа, и для выравнивания по горизонтали (правда, по вертикали нет). И шрифты всячески можно менять — и в смысле начертания, и по размеру, и по цвету. Надо только знать, что команд в меню все же побольше и, кроме того, там они детальнее разработаны. Например, если щелкнуть по кнопке Границы (точнее говоря, по черному треугольничку справа от нее, раскрывающему список возможностей), то увидишь 12 вариантов вычерчивания рамок. В меню же Формат — Ячейки — Граница можно сконструировать самому чуть ли не сотню таких рамок!

Формулы в ячейках

Что еще можно вводить в ячейки, помимо текста и чисел? Главная особенность электронных таблиц, как отдельного класса документов, состоит в возможности использования в них разнообразных формул. Собственно, даже не сами формулы играют определяющую роль, а то, что в этих формулах можно употреблять адреса других ячеек. Не просто значения исходных данных, а именно адреса ячеек, в которых эти данные содержатся.



Так, для получения стоимости ручек в ячейке `G6` производится, согласно введенной в нее формуле, операция перемножения данных из двух соседних ячеек (знак умножения — звездочка). Сама формула, как это можно видеть в строке ее редактирования, не содержит конкретных числовых значений цены и количества, а просто отсылает за ними по указанным адресам. При таком подходе результат автоматически корректируется при изменении исходных данных: если, например, вместо 25 штук потребуется купить 30, то на стоимости это отразится сразу же, как только новое количество будет введено в ячейку `E6`.

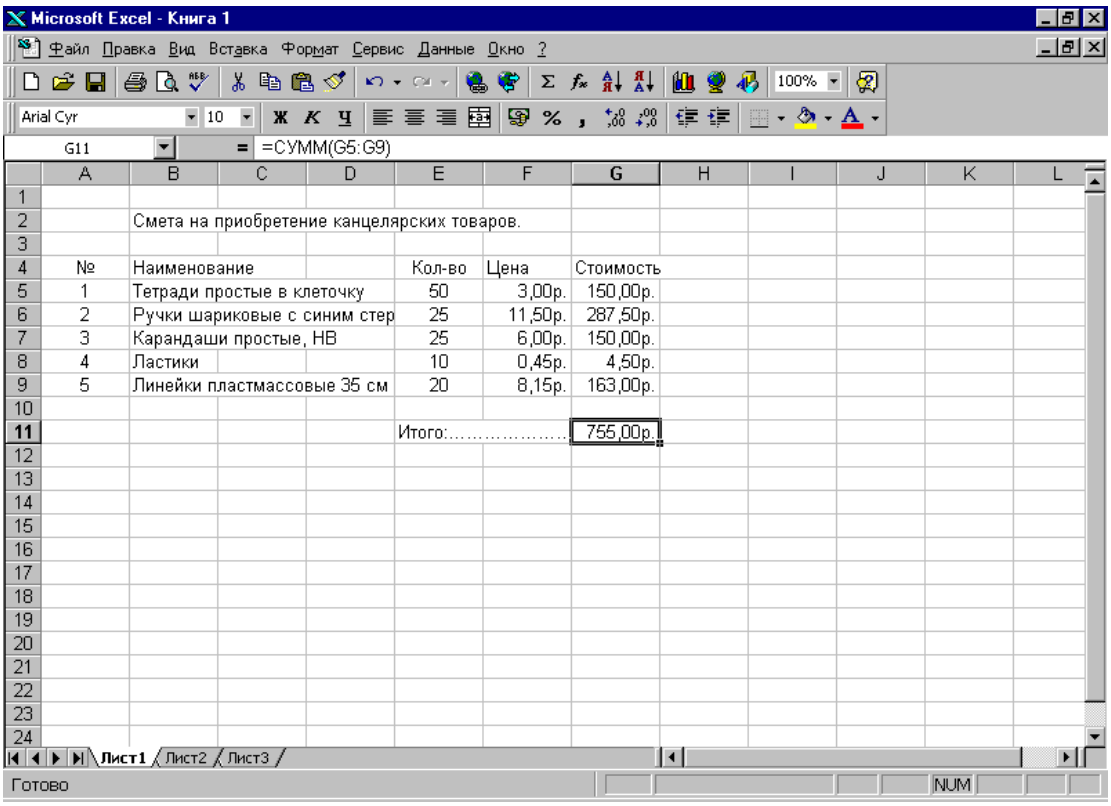
Каким же образом программа в момент ввода формулы в ячейку `G6` поняла, что это именно формула, а не просто какой-то набор латинских букв, образующих хотя и бессмысленный по внешнему виду, но все-таки текст? Отличительным признаком служит, очевидно, знак равенства, указанный первым. Он как бы говорит: “В данную клеточку таблицы на экране следует выводить не текст, который сейчас последует, а результат тех действий, что будут этим текстом описаны”. При этом, конечно, описание не может быть произвольным, а должно строиться по каким-то определенным правилам, принципы которых, вообще говоря, ясны.

Если формула будет ссылаться по адресу на какую-нибудь клетку таблицы, где уже вписана другая формула, то ничего страшного в этом нет. Такое разрешается. Например, в ячейке `G11` итоговая стоимость наверняка вычисляется по формуле. И вряд ли она столь громоздкая, как `=E5*F5+E6*F6+E7*F7+E8*F8+E9*F9`, ссылающаяся исключительно на исходные данные. Скорее уж там стоит более компактное выражение: `=G5+G6+G7+G8+G9`, — несмотря на то, что в нем упоминаются ячейки, в свою очередь, содержащие формулы. При введении новых значений цены и количества любого товара автоматически будет пересчитана как его стоимость, так и зависящая от этой стоимости итоговая сумма.

Вообще указанный механизм ссылок может приводить к неопределенности. Так будет, если формула, записанная в какую-нибудь ячейку, содержит ссылку на нее же саму. Особенно велика вероятность такой, как ее называют, циклической ссылки, когда она не прямая, а косвенная, т.е. приводит к себе не непосредственно, а через последовательность нескольких промежуточных этапов. Допустим, формула в `R15` ссылается на адрес `F7`, где имеется ссылка на `S38`, а там записана формула, ссылающаяся “обратно” на `R15`. Однако в подавляющем большинстве обычных расчетов подобные нетривиальные ситуации не встречаются.

Встроенные функции

При составлении формул очень часто используют заготовленные в самой программе функции, в частности, математические. Так, в ячейке G11 указана не довольно длинная комбинация =G5+G6+G7+G8+G9, как предполагалось, а функция суммирования (это видно из строки формул), дающая тот же результат, но более рациональным способом.



Поскольку функция — тоже формула, то и начинается ее запись со знака равенства. Далее идет имя функции, в нашем случае это СУММ. Вообще встроенных функций в программе очень много — одних только математических несколько десятков. Сюда входят, скажем, тригонометрические функции (SIN, COS, TAN и т.п.), степенные и показательные функции (EXP, LOG, КОРЕНЬ, СТЕПЕНЬ и т.д.); кроме того, имеются всевозможные округления (ОКРУГЛ, ОКРУГЛВНИЗ, ОКРУГЛВВЕРХ, ЦЕЛОЕ, ЧЕТН, НЕЧЕТ), а также суммирования (СУММКВ, СУММКВРАЗ, СУММПРОИЗВ) и многое, многое другое.

Но и это еще не все. Помимо математических, имеются функции другого рода — например, предназначенные для работы не с числами, а с текстом или с датой и временем, с логическими переменными, со статистическими данными. И у каждой — свое собственное имя.

Однако для большинства функций указания одного только имени недостаточно. Как правило, требуются еще пояснения: какие объекты участвуют в действиях функций, каковы особенности процесса их выполнения. Поэтому после имени функции следует (в круглых скобках) перечисление ее параметров, которые называют еще аргументом.

У функции, указанной в ячейке G11, аргумент, как мы видим, один — диапазон адресов той области таблицы, для которой выполняется суммирование. В данном примере речь идет о пяти клеточках, причем все они из одного столбца. Это и отражено в скобках, как начало и конец (G5:G9), через двоеточие.

Суммирование — простое действие, поэтому больше ничего уточнять не нужно. Однако вполне можно представить себе такую ситуацию, когда суммировать потребуется величины, расположенные в большой прямоугольной области, состоящей из целого ряда идущих друг за другом строк и столбцов. Но и в этом случае аргумент у функции будет один, поскольку область тоже останется одна, только обозначена она будет с помощью адресов верхнего левого и нижнего правого углов. Например, (B13:E17) для $4 \cdot 5 = 20$ ячеек, образующих единое целое.

А вот если бы область суммирования была раздроблена, состояла бы из нескольких отдельных участков, то, соответственно, и аргументов у функции было бы столько же, разделенных точкой с запятой. Скажем, так: (A14:B18; D14:E18; C20:H20), для трех областей из 10, плюс 10 и еще плюс 6 — всего, таким образом, 26 клеточек, — пример, конечно, искусственный, приведенный исключительно ради иллюстрации общего принципа.

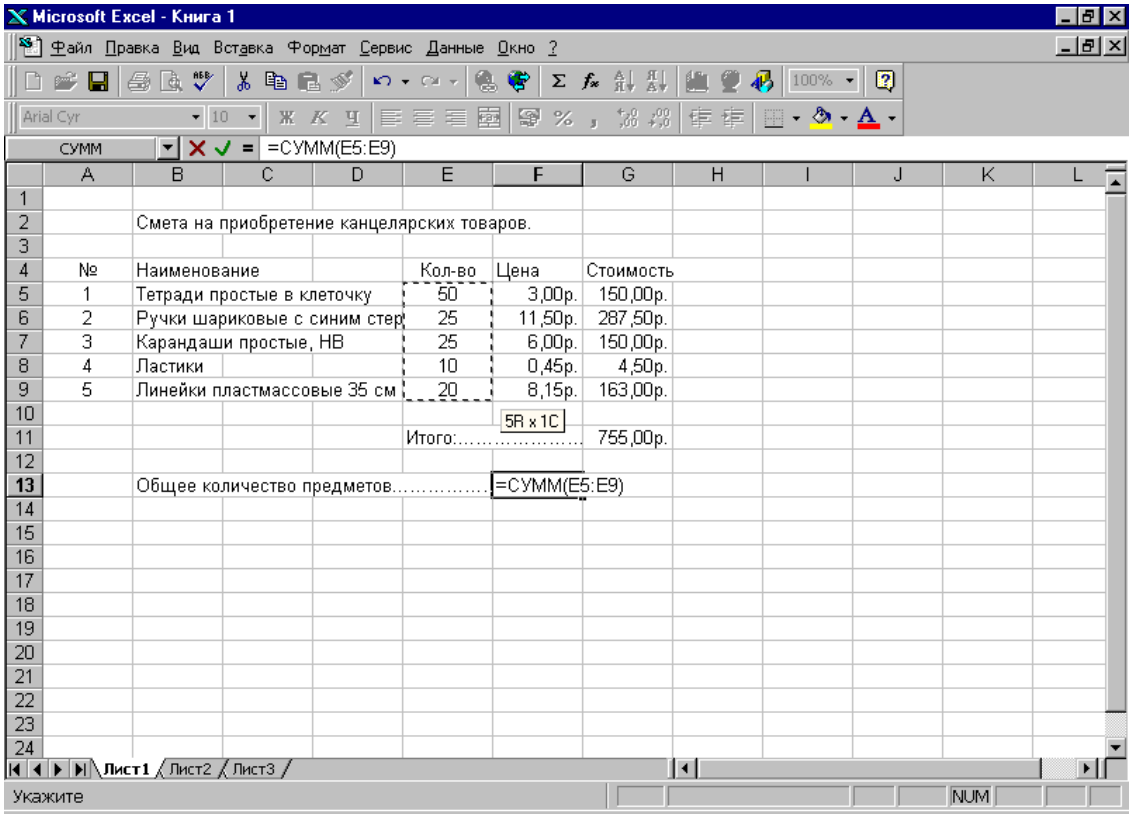
Автосуммирование

Предположим, что в рассматриваемую нами таблицу со сметой на приобретение канцелярских товаров необходимо добавить еще и общее количество покупаемых предметов. Для этого в ячейку B13 надо ввести пояснительный текст, а в ячейку F13 — соответствующую формулу.

С текстом все в порядке: соседние справа от B13 клетки таблицы свободны, и потому он будет виден целиком. И даже вместе с дополнительным рядом точек, хоть и необязательных, но делающих документ более удобным для чтения.

Что же касается ввода формулы, тут самый прямой путь — поместить курсорную рамочку на клетку F13 и начать со знака равенства. А затем, чередуя адреса ячеек и знак плюс, "пройтись" по всей области суммирования: =E5+E6+E7+E8+E9. Конечно, нет нужды вводить адреса с клавиатуры, да еще следить при этом, чтобы использовались только латинские буквы (символы кириллицы в адресах употреблять нельзя). Гораздо удобнее в нужный момент просто щелкать по соответствующей клетке, и ее адрес автоматически будет добавляться в конструируемую в строке формул команду. Завершается набор, естественно, нажатием клавиши Enter.

Однако тут (как часто бывает) прямой путь не является оптимальным, и чем больше наименований товаров присутствует в таблице, тем сильнее это проявится. Для сложения нескольких, да к тому же еще рядом расположенных величин сам Бог велел использовать не просто формулу, а функцию автосуммирования. Вызвать ее можно очень быстро, быстрее любой другой функции, поскольку данная операция чаще других употребляется в табличных документах. Находясь в ячейке F13, куда требуется ввести формулу, щелкнем по значку Σ (автосумма), расположенному на стандартной панели инструментов. Сразу же введется нужная функция, и даже аргумент ее будет указан в виде предположения, что, мол, вот эту именно область, обведенную пунктирной рамкой, надо суммировать, не правда ли?



Очень часто это предположение оказывается именно таким, которое можно принять. Например, когда использовалась автосумма в клетке G11, самой программой было предложено (в форме аргумента) складывать все числовые значения из того же столбца, расположенные над местом ввода формулы, т.е. из области (G5:G10). Удачное ли это было предложение? Удачное, несмотря даже на то, что клетка G10 тут вроде бы ни к чему, поскольку она пустая — ведь, как правило, итоговую сумму и подводят "по вертикали".

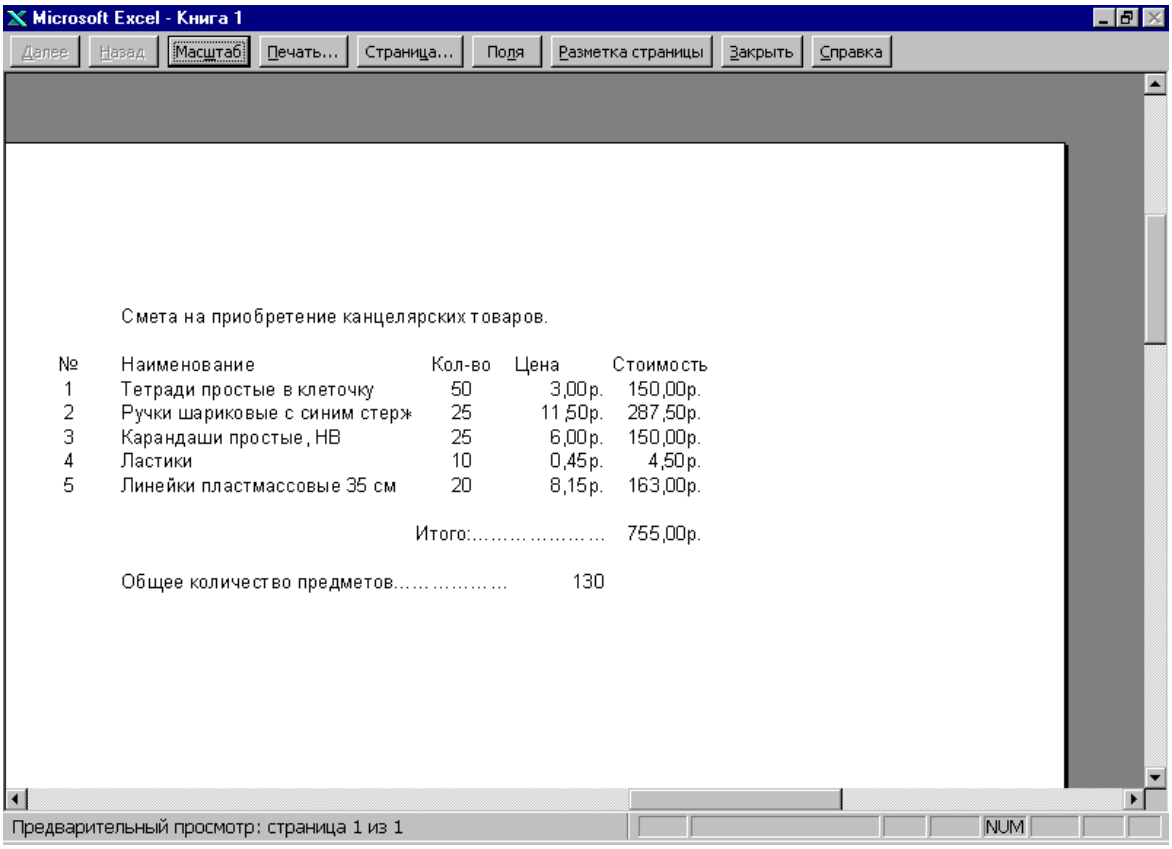
И в нашем случае для нахождения полного количества товаров автосумма предложит взять в качестве аргумента то, что указано выше, в том же самом столбце G, но теперь нам это совсем не подходит и от предложения придется отказаться. Собственно, даже не отказаться, а просто ввести другие адреса, "проведя мышью" с нажатой левой кнопкой по нужной области (E5:E9). И уже затем нажать клавишу Enter в знак согласия с исправленным вариантом.

Предварительный просмотр

В завершение "разбора" простейшего примера таблицы хотелось бы посмотреть, как будет выглядеть соответствующая распечатка на бумаге. Можно, конечно, просто выполнить печать, для чего достаточно щелкнуть по соответствующей кнопочке с изображением принтера на стандартной панели инструментов. Тогда все станет ясно, как говорится, по факту. Но обычно документ сначала все-таки просматривают на экране, только не в обычном рабочем режиме, а в специальном, имитирующем работу печатного устройства, и уж потом, убедившись, что документ выглядит как нужно, действительно посылают его на принтер.

Такой специальный режим носит название "Предварительный просмотр" (в литературе иногда встречается английский термин *preview*). На той же стандартной панели инструментов есть и его кнопка — справа от принтерной.

Правда, сразу после щелчка по ней документ будет виден уменьшенным, зато вся страница умещается на экране. В этом формате хорошо виден "общий план": как таблица расположена на листе бумаги, не выходит ли за его края и вообще красиво ли смотрится. Однако буквы и цифры при этом настолько мелки, что прочитать что-нибудь практически невозможно — слишком мал масштаб. Для детального же просмотра изображение следует увеличить, щелкнув по экранной кнопке вверху, которая так и называется — "Масштаб".



Смета на приобретение канцелярских товаров.

№	Наименование	Кол-во	Цена	Стоимость
1	Тетради простые в клеточку	50	3,00р.	150,00р.
2	Ручки шариковые с синим стерж	25	11,50р.	287,50р.
3	Карандаши простые, НВ	25	6,00р.	150,00р.
4	Ластики	10	0,45р.	4,50р.
5	Линейки пластмассовые 35 см	20	8,15р.	163,00р.
	Итого:.....			755,00р.

Общее количество предметов..... 130

Внимательно просмотрев документ, можно либо сразу приступить к печати (если все выглядит удовлетворительно), либо закрыть просмотр и вернуться, так сказать, к обычному виду для продолжения работы над ним с целью улучшения вида.

Каким, например, будет решение в нашем случае? В принципе и так сойдет, не слишком уж важная бумага. Но если стремиться к совершенству во всем, то кое-какие изменения надо, безусловно, внести, замеченные недостатки исправить.

Первое, что бросается в глаза, — это отсутствие разделительных линий, без которых и таблица — не таблица. Далее, плохо, что повсюду шрифт одного и того же размера. Название документа, заголовки колонок (графы), итоговые сведения — все это лучше сделать покрупнее или, допустим, использовать жирный шрифт или курсив, а может быть, даже и выделить цветом (причем не обязательно иметь цветной принтер, ведь и у серого цвета имеется множество оттенков-градаций).

Что еще? Заголовок "Цена" смещен, число "130" расположено неудачно, слова "...стержнем" так полностью и не видно. Колонку "Стоимость" не мешало бы сдвинуть вправо, а всю таблицу целиком, пожалуй, переместить на середину листа, а то занята только его верхняя часть, а остальное пространство пустое. Однако пора остановиться и сохранить то, что получилось. Создадим на диске, скажем, С, в папке "Упражнения" файл с именем "Смета". И на этом пока все. Первое занятие завершено.

Проверочные вопросы

1. Сколько столбцов и сколько строк содержит один рабочий лист таблицы?
2. По какой малозаметной детали можно определить, открыт ли документ, представленный на первом рисунке, только что, или в нем уже производились какие-то изменения?
3. При перемещении указателя мыши по таблице его вид меняется в зависимости от того, над каким элементом изображения он проходит. Сколько всего таких видов?
4. Сколько в рассмотренной таблице имеется пустых ячеек внутри области (A1:G11)?
5. Сколько внутри той же области ячеек, содержащих какую-либо информацию?
6. Сколько ячеек внутри той же области содержит текст, сколько — числа, сколько — формулы?
7. В скольких ячейках использована встроенная функция суммирования?
8. В каких клетках таблицы произойдут изменения при вводе новой цены тетрадей?
9. Каков формат числа в ячейке для общего количества предметов?
10. Как выровнен текст в ячейке со словом "цена"?

Задания для самостоятельной работы

1. Городская семья из нескольких человек проживает в трехкомнатной квартире. Требуется представить в виде таблицы распределение всей площади квартиры по ее отдельным составляющим (комнаты, кухня, коридор, подсобные помещения), а также общую и жилую площадь в сумме и на одного жильца.

Указания

- Считать, что в плане все помещения в квартире имеют прямоугольную форму.
- Длину и ширину каждого помещения выражать в метрах, а площадь — в квадратных метрах.
- Точность для длины и ширины — два знака после запятой, для площади — один знак.

2. В той же семье каждый трудоспособный член семьи получает какой-то оклад на основной работе, который облагается подоходным и пенсионным налогами. Представить все эти сведения в виде таблицы и рассчитать суммарный месячный доход семьи, а также доход, приходящийся в среднем на каждого ее члена и на каждого ее работника.

Указания

- Подоходный налог принять равным 12%, а пенсионный — 1% от оклада.
- Пенсию работающих пенсионеров приплюсовывать к их окладу (вычитая из него сумму налога).
- Пенсию неработающих пенсионеров приплюсовывать к суммарному доходу.
- Все денежные показатели, в том числе и средние, указывать в рублях без копеек.

Третьи Соловейчиковские чтения

Ярмарка педагогических идей
Школа сотрудничества

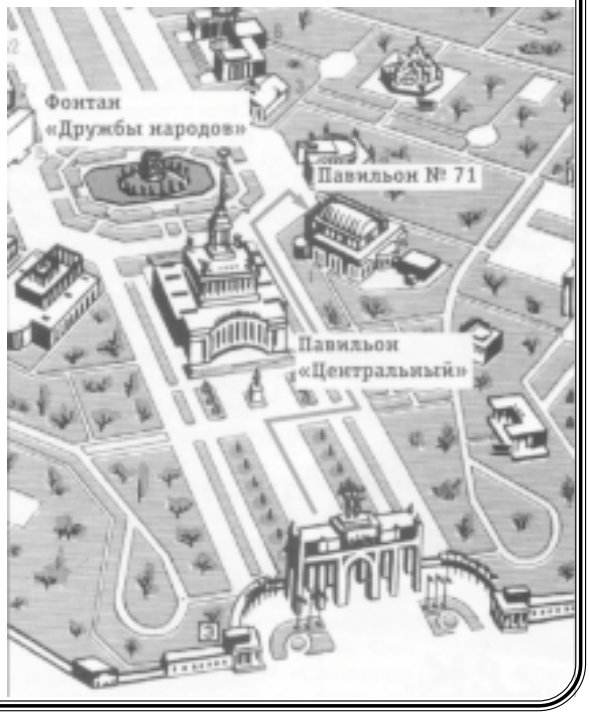
Дорогие коллеги!

Приглашаем вас, учителей и администрацию вашей школы, а также родителей ваших учеников посетить ЯРМАРКУ ПЕДАГОГИЧЕСКИХ ИДЕЙ. Она будет проходить 1 и 2 октября в ВВЦ в рамках Третьих Соловейчиковских чтений.

Мы будем рады встрече с вами и надеемся, что вам будет по-настоящему интересно.

На Третьих Соловейчиковских чтениях вы также сможете встретиться с сотрудниками редакции нашей газеты, узнать о планах публикаций, приобрести различные номера "Информатики".

Ждем вас 1 и 2 октября
в павильоне 71 ВВЦ с 10 до 18 часов



Третья международная конференция "Перспективы систем информатики"

С 6 по 9 июля 1999 г. в Новосибирске проходила 3-я международная конференция "Перспективы систем информатики", которая привлекла большое внимание отечественных и зарубежных специалистов по теоретической и прикладной информатике.

Слово организаторам и участникам конференции.

"ШКОЛЬНАЯ ИНФОРМАТИКА"

Идея проведения секции "Школьная информатика" в рамках конференции "Перспективы систем информатики" возникла в Санкт-Петербурге при подведении итогов Российской олимпиады по информатике для школьников. Круг участников этой секции оказался ограниченным, но тем не менее учителя информатики получили здесь долгожданную возможность широкого обсуждения актуальных вопросов преподавания информатики в школе.

В работе секции приняли участие более тридцати школьных педагогов-практиков и около двадцати научных работников, преподающих информатику по совместительству. Если доклады и сообщения в основном отражали мнение представителей научной среды, то "круглый стол" по теме "Какой станет "вторая грамотность" в третьем тысячелетии" позволил высказаться практически всем присутствующим.

Специально подготовленный бюллетень знакомил желающих с деятельностью недавно созданного Фонда академика А.П. Ершова для поощрения школьников за самостоятельные разработки, школьных педагогов — за постановку и методическое обоснование школьных программистских проектов, руководителей студенческих работ — за организацию коллективных и тематически взаимосвязанных проектов, а также активных организаторов различных мероприятий — за поддержку школьной информатики.

А теперь более подробно о содержании докладов и сообщений.

Повышение интереса к истории науки во всем мире, по мнению Я.И. Фета (доклад "История информатики: исследования, публикации, преподавание"), заслуживает пристального внимания для информатики, становление, стремительное развитие и достижение зрелости которой состоялось в пределах жизни одного поколения. Появляются работы в этой области и в России. Важную роль начинают играть виртуальные музеи. Использование опыта прошлого может являться эффективным средством просвещения, смягчающим противоречия в развитии информатики, особенно противоречия этического и культурного характера.

Тематика других докладов отражает проблемы и перспективы образовательной информатики. А.А. Берс (доклад "Информатика и образование. К пятидесятилетию понятия подпрограмма") предлагает трактовать основные понятия и структуры информатики и образования исходя из общефилософских представлений о деятельности, моделируемых на компьютерах. Смена парадигм информатики соотнесена Берсом с развитием средств свертки данных и действий и уточнением грани между объектами и субъектами деятельности. Применительно к образованию эта модель позволяет легко разделять образовательную и педагогическую деятельность в зависимости от ее субъекта.

Понятие информационной модели как универсального средства изучения мира учениками выбрано А.Г. Гейном (доклад "Школьный курс информатики в свете его межпредметных связей") в качестве ведущего фактора при структуризации школьного курса информатики с акцентом на роль знаковых моделей в процессе применения научных методов в учебном процессе. Вытекающая из таких рассмотрений смена лозунгов — от программирования к технологиям — беспокоит Ю.А. Первина (доклад "Школьная информатика. Сегодняшние уроки наследия А.П. Ершова") в связи с опасностью принижения роли учителя информатики и нелепостью расчленения этой дисциплины на две составляющие (технология и алгоритмика). Автор убежден, что мудрость стратегии А.П. Ершова, выделившего инварианты и динамические компоненты школьной информатики, проявляется и сегодня.

В других докладах и сообщениях были проанализированы более конкретные подходы к преподаванию информатики в разных условиях. Сведе́ние понятия "информационные процессы" к понятию "система", выполненное К.М. Поляком, иллюстрирует методику системного анализа при минимизации учебного материала по информатике. Предлагается ограничить сложность этого понятия четырьмя типами элементов (источник, приемник, хранитель-накопитель, преобразователь) с четырьмя нормированными связями, символизирующими кольцевую передачу информации.

Организация современной информационной среды начального обучения информатике и программированию с плавными переходами к более сложным уровням, со сменной национальной лексикой, электронным учебником и структурным редактором разрабатывается как система "Азбука АП" (в честь основателя "образовательной" информатики) в Самаркандском госуниверситете (доклад С.С. Кобилова и Ш.И. Ходиева). Н.А. Юнерман описала (в тезисах) попытки раскрыть общеобразовательный потенциал информатики путем сравнительно легко-

го моделирования трудных для изучения школьниками явлений с помощью среды ЛогоМиры. Способность детей к развитию интуиции в области организации асинхронных процессов по приобретению навыков составления алгоритмов, приводящих к последовательным программам, Т.И. Тихонова предлагает поддержать специальными системами исполнителей, моделирующими понятные ученикам процессы. Функциональное программирование задач компьютерной алгебры, по мнению Л.В. Городней и Н.А. Калининой, позволяет сократить сроки освоения поддержки полного жизненного цикла программ при изучении современной программотехники на математических факультетах университетов.

Ряд сообщений был посвящен активизации способностей и творческих интересов учащихся и поддержке изобретательства.

Исследованию методов работы за компьютером без особого вреда для здоровья посвящено сообщение А.А. Руденко и Н.В. Соседкиной. А.Н. Буров рассказал о многолетней практике разработки и применения системы развития способностей через навыки быстрого чтения и пространственного представления, классификации и усвоения информации, основанной на знании оптимальных алгоритмов запоминания и гармоничного развития памяти, приводящих к саморазвитию личности. Тревога за детей, не находящих контакта с педагогами при традиционной схеме учебного процесса в условиях недостаточного разнообразия учебной литературы, заставила Соседкину разработать курс "Элементарной информатики для малышей", побуждающий учащихся к самостоятельным рассуждениям. В сообщении А.Г. Марчука, Ф.А. Мурзина и других был предложен перенос методики ТРИЗ[1] из области технического творчества на задачи поиска информации в Интернет с формализацией основных понятий теоретико-множественными, логическими и топологическими построениями, реализующими программную поддержку. Анализ требований к организации системы подготовки и воспитания специалистов, способных работать в трудных условиях (доклад Л.В. Городней и Т.Г. Чуриной), выполнен в форме "спортивных тренировок" школьников для участия в олимпиадах по информатике.

Несколько работ было посвящено методам организации работы студентов, осваивающих прикладное программирование.

А.И. Куликов отмечает, что профессиональное владение методами и алгоритмами современной компьютерной графики — важное требование подготовки специалистов научно-технического профиля, поддерживающих программное обеспечение для решения задач визуализации. Именно привлекательность компьютерной графики — залог успеха деятельностного подхода, взятого на вооружение Л.Г. Алсынбаевой, Л.А. Голубевой и Л.А. Москвиной для введения старшеклассников в мир прикладного программирования. При таком подходе тематика выполняемых учениками программных проектов, описанных в работе Ю.И. Молородова и В.И. Новожиловой, может требовать применения ряда математических методов и знания физических явлений, тем самым содействуя их усвоению. Эта линия получает продолжение в электронном учебнике М.М. Бежановой и Л.А. Москвиной "Практическое программирование" для первокурсников университета. Сложившаяся в рамках данного подхода система учебных проектов положена в основу комплексного учебного плана для преподавателей, школьников и студентов, осваивающих информационные технологии в качестве членов ассоциации "Информатика XXI века", которой посвящено отдельное сообщение Л.Г. Алсынбаевой.

Работы В.Г. Казакова и И.Н. Скопина нацелены на автоматизацию труда лектора и педагога. Казаков разработал универсальный программный компонент, обеспечивающий многократное использование учебного материала в рамках клиент-серверной организации информационных систем учебного назначения. Скопин рекомендует создать деятельно ориентированную систему, обеспечивающую и ведение, и подготовку уроков.

Авторы нескольких сообщений делились личными впечатлениями от встреч с академиком А.П. Ершовым. Л.Н. Корниенко благодарит судьбу за возможность общения с необыкновенным человеком, который помог ей, научному работнику, найти себя в школе. Н.А. Садовская, для которой этот человек стал Учителем с большой буквы, призывает создать банк воспоминаний об Андрее Петровиче Ершове.

При подготовке "круглого стола" "Какой станет "вторая грамотность" в третьем тысячелетии" были сформулированы достаточно интересные вопросы:

1. Общечеловеческие аспекты изучения информатики.
2. Влияние информатики на этику.
3. Что включать и что не включать в информатику.
4. Соотношение программирования и информатики.
5. Возможна ли "вторая грамотность" без первой и наоборот?
6. В каком возрасте, чему и как учить?

Дискуссия по этим и другим проблемам продолжалась в заключительный день конференции с 11 до 18 часов и могла бы идти много дольше. При этом многие вопросы требовали дальнейшего обсуждения. Очевидно, нужны новые дискуссии. Анализ результатов проведенного "круглого стола" заслуживает отдельного рассмотрения.

Л.В. Городняя,
к. ф.-м. н., старший научный сотрудник Института систем информатики СО РАН

О РАБОТЕ СЕКЦИИ "ШКОЛЬНАЯ ИНФОРМАТИКА"

Мне представилась возможность принять участие в работе секции "Школьная информатика".

Отмечая разнообразную и актуальную тематику рассмотренных на секции докладов и тезисов, хотелось бы рассказать о дискуссии "круглого стола". Заседание

[1] ТРИЗ — теория решения изобретательских задач.

"круглого стола" стало завершающим этапом работы секции "Школьная информатика", причем возникло оно не экспромтом, как это иногда случается, а было предусмотрительно включено организаторами в план работы.

Выступления на предложенную тему "Какой станет "вторая грамотность" в третьем тысячелетии" затронули многие актуальные вопросы образования в области информатики. Выступающие вынесли на обсуждение самые насущные проблемы сегодняшнего дня: содержание образования в области информатики, формы обучения, вопросы этики в информационном обществе, охрана здоровья ребенка, работающего за компьютером. Были затронуты такие важные вопросы, как тема взаимоотношения "ученик — учитель", "учить или помогать учиться?", и другие.

Обсуждалась также проблема достаточности курса информатики в общеобразовательной средней школе, соответствия его содержания требованиям сегодняшнего дня. Были рассмотрены вопросы работы с одаренными детьми, проблемы подготовки детей к олимпиадам по информатике, обсуждались такие формы работы, как воскресные, заочные и летние школы по информатике.

В ходе дискуссии было отмечено, что культуру общения с компьютером нужно прививать с детства (как и многие необходимые навыки). Часто дети начинают общаться с компьютером уже в дошкольном возрасте. Если школа не сможет увлечь дошкольников, а тем более школьников и не поможет им научиться пользоваться компьютером как мощным инструментом в процессе познания и освоения нового, не подарит радость открытий, то досуг ребенка займут пресловутые компьютерные игры, что резко ограничит богатый и разнообразный мир общения с компьютером.

Особое внимание учителей информатики привлекли выступления, в которых обсуждалась тема здоровья. Именно с детства человек должен помнить о своем здоровье, научиться беречь его, в том числе и работая за компьютером. Актуальность и злободневность данной темы несомненны.

Еще одна проблема связана с возрастающей дистанцией между учителем и учеником. Ученик уже использует в процессе обучения сеть Internet как источник информации (особенно это относится к гуманитарным наукам), а школьный учитель-предметник порой не только не владеет компьютерными технологиями, но и не испытывает в этом никакой потребности. Может быть, помочь учителю-предметнику освоить информационные технологии — задача не менее актуальная, чем изучение информатики учениками?

Мне удалось перечислить лишь часть вопросов, поднятых на заседании "круглого стола". Их обсуждение вряд ли позволит разрешить существующие проблемы, но оно отражает активную жизненную позицию участников и организаторов этой конференции, демонстрирует их несомненную заинтересованность в деле развития информатики.

Во время работы конференции участники имели возможность получить информационные материалы, обменяться мнениями. Возникли деловые связи и взаимовыгодные контакты, которые, несомненно, помогут в дальнейшей работе. Большая подготовительная работа оргкомитета позволила провести работу секции "Школьная информатика" на высоком уровне, создать дружескую атмосферу. Интересной и полезной оказалась оживленная дискуссия за "круглым столом".

В заключение хотелось бы отметить важность и полезность работы секции "Школьная информатика" и конференции в целом. Хочется пожелать организаторам дальнейших творческих успехов и выразить надежду, что секция "Школьная информатика" станет значимой частью всех будущих "Ершовских конференций", а учителя информатики — ее активными участниками.

Г.Д. Шапалова,
зам. директора по НИТ,
средняя школа № 20, г. Тюмень

ПЕРСПЕКТИВЫ СИСТЕМ ИНФОРМАТИКИ

Третья международная конференция "Перспективы систем информатики", посвященная, как и две предыдущие, памяти пионера отечественного программирования, одного из наиболее выдающихся российских ученых в области информатики, академика Андрея Петровича Ершова, проходила с 6 по 9 июля 1999 г. в новосибирском Академгородке. Эти регулярные мероприятия известны в стране и в мире как "Ершовские конференции". Как и всегда, конференция носила научный характер, на ней были представлены свежие результаты теоретических и методологических исследований, касающихся перспективных направлений системной информатики.

В качестве организатора конференции выступил Институт систем информатики им. А.П. Ершова Сибирского отделения РАН, а готовил ее программу представительный программный комитет, сопредседателями которого стали широко известные ученые Динес Бьорнер (Дания), Манфред Брой (Германия) и Александр Замулин (Россия), а членами — ведущие российские и зарубежные ученые (11 членов программного комитета были из России, а 24 — из Франции, Германии, США, Швеции, Польши, Финляндии, Украины, Эстонии, Хорватии, Макао). Строгий отбор удалось пройти менее чем половине из присланных докладов. Состав докладчиков был поистине международным: 13 докладов из России, 7 — из Германии, по 5 — из Дании и Франции, по 2 — из Швейцарии, Англии, Нидерландов, Швеции, Испании, Японии и Австрии и по одному — из Италии, Индии, Румынии, США и Узбекистана. Кроме того, программный комитет пригласил ряд ведущих ученых из США, Франции, России, Дании и Шотландии выступить на конференции с приглашенными докладами.

Проблемы, которые обсуждались на конференции, относились к весьма значимым для развития системной информатики.

Применение формальных методов в спецификации и верификации программ — один из наиболее перспективных путей к повышению доказательности программирования и надежности программного обеспечения. Этой тематике были посвящены два приглашенных доклада и ряд секционных. В своем приглашенном докладе Д.Санелла (Шотландия) сообщил о международном проекте создания среды для разработки программного обеспечения исходя из алгебраических спецификаций, основанном на языке спецификаций CASL. Алгебраические спецификации рассматривались также в приглашенном докладе М.-К. Годель (Франция) и А.Замулина (Россия). В нем авторы излагали и сопоставляли два интересных подхода, основанных на алгебраических императивных спецификациях, которые успешно применяются в разных областях, связанных с созданием реактивных систем и распределенных систем.

Спецификация и верификация реактивных систем (к ним относится целый ряд сложных программных систем — от управления движением до медицинских систем диагностики), а также сложных распределенных систем — очень важный путь к построению надежного программного обеспечения. В секционных докладах рассматривались вопросы спецификации архитектур программных систем, анализа спецификаций реактивных систем, верификации их графических спецификаций, спецификации гибридных (аналого-цифровых) систем.

Важными компонентами программного обеспечения являются также языки и трансляторы. Здесь весьма актуальными для многочисленных пользователей являются вопросы надежности. В ряде докладов излагались подходы к спецификации синтаксиса и семантики императивных языков, верификации всех фаз трансляции — от перевода до генерации.

Значительную роль в программировании сейчас играют логические подходы. В докладах сообщалось о применении временных логик к анализу поведения взаимодействующих логических программ, о логическом анализе семантики и критериев корректности некоторого языка представления знаний, об использовании m-исчисления для верификации системы взаимодействующих процессов, об использовании теории категорий в проектировании оборудования, о языке функционально-логического программирования, о самоинтерпретации в l-исчислении.

Теория и практика параллельного и распределенного программирования — одна из важнейших областей современного программирования. Применению формальных методов к проектированию, проверке и верификации сложных телефонных систем посвятили свой приглашенный доклад Д. Мери (Франция) и П.Джибсон (Ирландия). В других докладах рассматривались проблемы верификации управляющих протоколов, распараллеливания с учетом разделяемой памяти, сравнения различных формальных моделей параллелизма, развития моделей сетей Петри, а также абстрактные структуры сообщений.

Большая часть докладов была посвящена такому методу построения путем специализаций надежных и качественных программ, как смешанные и частичные вычисления, — направлению, одним из основателей которого был А.П. Ершов. Н.Джонс (Дания) в своем приглашенном докладе предложил и обосновал весьма перспективный подход к специализации — преобразования прогонкой. В докладах излагались результаты применения специального анализа для частичных вычислений, использования смешанных вычислений и преобразований для верификации программ, исследования завершаемости смешанных вычислений, эффективности суперкомпиляции как разновидности смешанных вычислений.

Анализу и преобразованиям программ были посвящены доклады об оптимальном алгоритме чистки структурированных программ и об анализе JAVA-программ с учетом исключений.

Ряд выступлений авторы посвятили общеметодологическим проблемам. В.Базили (США) с соавторами в своем приглашенном докладе сообщил о роли экспериментирования в накоплении знания в технологии программирования, проиллюстрировав это реальными и сложными экспериментами. Р.Моррисон (Шотландия) с большой группой соавторов изложил в приглашенном докладе свой подход к построению программных композиций и устойчивого программного обеспечения, который он назвал гиперпрограммированием. Один из докладов был посвящен общим проблемам распознавания образцов в таких распространенных структурах, как слова и деревья.

Известно, какую важную роль в современном программировании играет парадигма объектно-ориентированного (ОО) программирования. На конференции рассматривались вопросы построения удобной модели множественного наследования, применения объектно-ориентированного подхода к спутниковым базам данных, технологии построения ОО-программ на языке Си++, вызову методов в распределенных ОО-системах.

Дальнейшим развитием ОО-подхода является используемый в системах искусственного интеллекта мультиагентный подход. Агент может быть рассмотрен как более активный и независимый объект, существующий во времени. Рассказывалось о применениях мультиагентного подхода для планирования движения робота, понимания коротких текстов на естественном языке.

Большое внимание уделялось авторами искусственному интеллекту как одной из наиболее активно развивающихся областей системной информатики. Подробно рассматривалось такое связанное с интеллектуальными системами направление, как программирование в ограничениях. Здесь говорилось и о методах и языках для разрешения ограничений (в том числе интервальных ограничений), и о недоопределенных моделях для графов, и о библиотеке решателей для теоретико-множественных ограничений. Рассматривались также проблемы речевого общения с роботами и многоязычного машинного перевода.

И.В. Поттосин,
д. ф.-м. н., профессор, главный научный сотрудник Института систем информатики СО РАН

Компьютер играет в реверси

Ю.А. Соколинский

Введение

Известна и вполне объяснима притягательность компьютерных игр для детей. Живой интерес у них вызывает и программирование таких игр. Для учащихся классов с углубленным изучением информатики предлагается проект: разработка программы игры в реверси. Эта игра предназначена для двух игроков, причем каждый из них обладает полной информацией о ситуации на игровом поле и ресурсах партнера. В этом смысле игра реверси аналогична таким известным "интеллектуальным" играм, как шашки и шахматы, но значительно проще их. Поэтому составление программы игры в реверси, в которой роль одного из партнеров играет компьютер, вполне по силам достаточно подготовленным учащимся. В основном требуется умение работать с циклами и массивами, а также формировать простейшие изображения в графическом режиме. Игровые программы могут значительно отличаться друг от друга по уровню игры и, естественно, по сложности. Здесь предлагается один из простейших вариантов, хотя имеется возможность повысить эффективность игры ценой усложнения алгоритма.

В статье [1] рассказывается о правилах и истории игры в реверси, излагаются идеи и принципы, которыми надо руководствоваться для успешной игры, приводятся примеры интересных партий. В одной из этих партий партнерами были человек и компьютер, причем последний проиграл. В книге [2] можно найти программы игры в реверси на языке QBasic.

1. Правила игры в реверси

Видимо, не все знакомы с игрой в реверси, поэтому начнем с изложения правил этой игры. Они достаточно простые.

Используется доска размером 8 × 8 полей (можно использовать доску и другого размера, но обязательно четного, например, 10 × 10), а также соответствующий (64, 100 и т.д.) комплект плоских фишек, окрашенных с противоположных сторон в два разных цвета (например, красный и зеленый или розовый и желтый).

Каждый игрок выбирает свой цвет, один из цветов дает право первого хода. Этот цвет мы по аналогии с шашками и шахматами будем условно называть "белым", а второй — "черным".

Первоначально на четырех центральных полях ставятся фишки таким образом, что на одной диагонали стоят фишки одного цвета, а на другой — противоположного. На *рис.* 1 представлена начальная позиция.

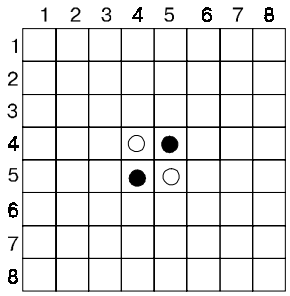


Рис. 1

Цель игры — поставить на доску как можно больше фишек своего цвета.

Партнеры ходят по очереди, если есть возможность поставить фишку своего цвета. В противном случае ход пропускается и повторно ходит соперник.

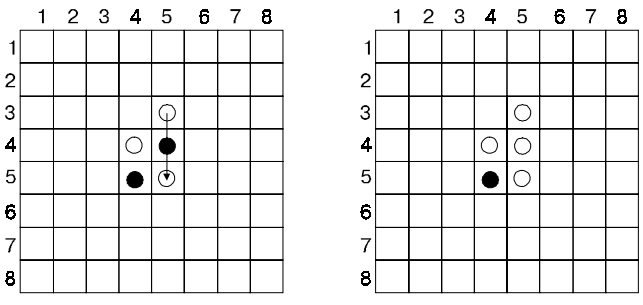
Игра считается законченной, когда оба партнера лишены возможности сделать ход. Выиграл тот партнер, у кого на доске в заключительной позиции больше фишек.

Два последних правила требуют пояснений.

Фишка может быть поставлена на свободное поле лишь в следующем случае: рядом с ней на соседнем поле по горизонтали, вертикали или диагонали находится "чужая" фишка; на продолжении линии, соединяющей эти два поля, могут находиться также несколько "чужих" фишек, а затем обязательно должна стоять "своя" фишка.

При установке новой фишки все "чужие" фишки, находящиеся между ней и замыкающей "своей" фишкой, переворачиваются, т.е. приобретают "свой" цвет. Отсюда и название игры (английское слово *reverse* означает переворот).

На *рис.* 2 показано, как в начальной позиции "белые" ставят свою фишку на поле (3, 5) (*рис.* 2*а*) и переворачивают чужую на поле (4, 5) (*рис.* 2*б*).



а *б*

Рис. 2

Возможно, что имеется несколько линий, соединяющих установленную фишку со "своей" через "чужие". Тогда переворачиваются все "чужие" фишки на каждой линии.

2. Этапы разработки программы

Разработку программы мы выполним в два этапа:

- На первом этапе составляется программа, которая позволяет двум партнерам играть в реверси с помощью и при посредничестве компьютера. Этот этап носит вспомогательный характер, здесь решаются основные технические проблемы. Впрочем, он имеет некоторый самостоятельный интерес, и, возможно, для некоторых учащихся достаточно ограничиться составлением указанной программы.
- На втором, основном этапе эта программа модифицируется таким образом, чтобы одного из партнеров (а именно — играющего "черными") заменил компьютер. Здесь учащиеся знакомятся с подходами к программированию "интеллектуальных" игр, понятием функции оценки хода.

Такой эволюционный подход часто используется при разработке больших и сложных программных комплексов.

3. Этап 1 — программа игры в реверси для двух партнеров

3.1. Сценарий

Сначала разберемся, что, собственно, должна делать эта программа (для краткости будем называть ее программой 1). Другими словами, речь идет о постановке задачи, или, как еще говорят, о составлении сценария. На первый взгляд кажется, что программа 1 просто заменяет доску и комплект фишек. Должна высвечиваться доска с фишками, которые на ней находятся в данный момент. Партнер, которому принадлежит очередь хода, вводит в компьютер координаты поля, куда он ставит свою фишку, на доске производятся соответствующие изменения и т.д. Однако такой мощный инструмент, как компьютер, способен и на большее. Если вы играли в реверси, то, наверное, знаете, что иногда затруднительно найти все свои возможные ходы, а их еще надо оценить и выбрать наилучший. Давайте поиск всех возможных ходов поручим компьютеру, а их оценку и выбор оставим за игроками. Надо учитывать, что если мы этого не сделаем, то все равно придется в программе контролировать корректность ходов.

Тогда возникает вопрос, как сообщить о возможных ходах. Может быть, вывести координаты всех полей, куда можно поставить свою фишку? Это явно не самый удобный способ. Выберем следующий вариант. На одном из возможных полей высвечивается фишка, а чтобы она отличалась от находящихся на доске, она не закрашивается (но ее контур имеет "свой" цвет). Нажатием выбранной клавиши игрок переходит к следующему возможному полю и т.д. Когда все возможные поля кончаются, очередное нажатие приводит к первому возможному полю, так что просмотр при желании выполняется несколько раз. Наконец, когда игрок принял решение, он нажимает другую выбранную клавишу. На доске появляется новая фишка, переворачиваются соответствующие фишки партнера, высвечивается "возможная" фишка цвета игрока, за которым сейчас очередь хода.

Таким образом, надо для каждого партнера выбрать пару клавиш — клавишу просмотра и клавишу хода. Например, можно в качестве клавиш просмотра выбрать A и L, а в качестве клавиш хода — Q и P (они расположены на противоположных сторонах клавиатуры).

Что еще надо поместить на экран? Желательно перед каждым ходом видеть счет игры, т.е. количество фишек каждого партнера. Так, в начале игры он составляет 2 : 2. Кроме того, будет полезно поместить на экран подсказку о клавишах просмотра и хода. На *рис.* 3 дан пример экрана, содержащего очередную позицию и возможный ход белых, счет игры, а также подсказку об управляющих клавишах.

Рис. 3. Пример экрана для программы 1

Как программа сообщит, что игра окончилась? Признаком окончания игры служит отсутствие "возможной" фишки на доске. Отметим, кстати, что игра может кончиться и при количестве фишек на доске, меньшем 64.

Что делать при окончании игры? Временно приостановить программу, чтобы партнеры могли разобраться в ситуации, а затем закончить ее работу нажатием клавиши (лучше любой, но можно ограничиться какой-то определенной, например, Enter или Esc). Конечно, по желанию учащихся можно в конце поместить на экране сообщение о результатах игры, а также запросить партнеров, не хотят ли они сыграть еще одну партию, и при положительном ответе начать игру заново.

3.2. Организация данных

Мы обсудили вопросы, связанные со сценарием, и переходим к разработке алгоритмов. Сначала остановимся на организации данных, участвующих в этих алгоритмах. Разумеется, используемые здесь обозначения данных, процедур и функций не являются обязательными.

Начнем с геометрических параметров. Приводя их примерные значения, мы ориентируемся на монитор типа VGA с размером экрана 640 × 480 пикселей.

Удобно использовать половину размера поля доски, обозначим ее через *a*. Можно принять $a = 15$. Радиус фишки *R* должен быть меньше *a*. Например, примем $R = 12$. Чтобы зафиксировать положение доски на экране, достаточно задать экранные координаты центра одного из полей. Удобно выбрать левое верхнее поле. Значение абсциссы его центра *X*1 определяется из условия, что доска должна располагаться по вертикальной оси экрана. Отсюда

$$X1 = 320 - 7 \cdot a$$

Ордината центра указанного поля может иметь значение $Y1 = 45$. Будем нумеровать горизонтали доски сверху вниз, а вертикали — слева направо, см. *рис.* 1 и 2. Пусть поле находится в *i*-й горизонтали и *j*-й вертикали. Тогда координаты его центра *Xc*, *Yc* определяются соотношениями:

$$Xc = X1 + 2 \cdot (j - 1)\ a$$

$$Yc = Y1 + 2 \cdot (i - 1)\ a$$

Информационной моделью состояния доски служит двумерный массив (матрица) `rvr[1:8,1:8]`. Элементы этого массива `rvr[i,j]` принимают значение:

- 0, если поле *(i, j)* — свободное;
- 1, если поле *(i, j)* занято фишкой первого игрока (или "белой" фишкой);
- —1, если поле *(i, j)* занято фишкой второго игрока (или "черной" фишкой).

При этом верхней горизонтали доски на экране соответствует индекс строки массива $i = 1$, а нижней — 8.

В дальнейшем придется постоянно работать с парой индексов поля *(i, j)*. Поэтому удобно ввести тип поля. В Паскале он определяется как

```
type tF = record i, j: integer end;
```

и аналогично в других языках программирования. Если величина `F` имеет тип `tF`, то `F.i`, `F.j` — индексы строки и столбца поля.

Список возможных полей, на которые можно поставить свою фишку, будем хранить в массиве `lst[1:64]`, его элементы имеют тип поля. Фактическое число возможных полей обозначим через `Clist`.

Информацию об очереди хода в данный момент хранит переменная `FS`. Если сейчас ходит первый партнер ("белые"), то `FS = 1`, иначе ход "черных": `FS = 2`.

Переменные `k1` и `k2` хранят число фишек первого и второго игрока, находящихся на доске в данный момент.

Еще нам понадобятся несколько массивов-констант из двух переменных:

`Col` — цвета фишек первого и второго партнера. Например, можно принять `Col[1]=12` (розовый цвет), `Col[2] = 14` (желтый цвет).

`KeyM` — клавиши хода. Выше рекомендовалось `KeyM[1]='Q', KeyM[2]='P'`.

`KeyI` — клавиши просмотра возможных ходов. Можно использовать соседние клавиши (к тем, что рекомендованы выше): `KeyV[1]='A', KeyV[2]= 'L'`.

`ASign` — знаки партнеров: `ASign[1]=1, ASign[2]=-1`.

Переходим к процедурам и функциям.

3.3. Графические и вспомогательные процедуры и функции

Начнем с графических процедур.

Процедура `ClrF` очищает заданное поле. Входным параметром служит переменная `F` типа поля.

Находятся координаты центра поля по приведенным выше формулам. Изображается закрашенный квадрат с найденным центром и размером 2*a*. Для контура квадрата можно выбрать, например, синий цвет, а для закраски — белый.

Процедура `Draw1` изображает заданное поле. Входной параметр — переменная `F` типа поля.

Чистим поле `F`, обращаясь к процедуре `ClrF(F)`. Далее, если `rvr[i,j]<>0`, то поле содержит фишку. Изображаем ее как закрашенную окружность радиуса *R* в центре квадрата. Цвет закраски определяется знаком `rvr[i,j]`: при `rvr[i,j]>0` цвет берется как `Col[1]`, а иначе как `Col[2]`. Кроме того, при `rvr[i,j]>0` наращивается счетчик `k1`, при `rvr[i,j]<0` — счетчик `k2`.

Процедура `DrawAll` изображает доску с фишками, а также выводит на экран счет игры и подсказку о клавишах просмотра и хода.

Счетчикам `k1` и `k2` даем начальные, нулевые значения. В цикле по строкам и столбцам матрицы `rvr` формируем переменную `F` типа поля и обращаемся к процедуре `Draw1(F)`. В результате получаем изображение доски с фишками. Имея значения счетчиков `k1` и `k2`, формируем и выводим на экран строку счета игры. Выводим на экран подсказку о клавишах просмотра и хода.

Процедура `PossMove` изображает возможное поле. Входной параметр — переменная `iList`, значение которой — индекс (порядковый номер) возможного поля в массиве `lst`.

Находим переменную `F=lst[iList]` и индексы поля *i* и *j*. Рисуем окружность радиуса *R* в центре квадрата *(i, j)*. Цвет определяется порядком хода, т.е. значением `Col[FS]`.

Мы исчерпали все графические процедуры.

Функция Rkey выдает значение клавиши, которую выбрал партнер.

В цикле выполняется команда чтения символа с клавиатуры, символ переводится в верхний регистр, и полученное значение присваивается переменной `key`. Цикл прекращается, когда значение `key` совпадет с клавишей хода или просмотра возможных ходов того партнера, которому принадлежит очередь хода. Оно и является значением функции `Rkey`.

3.4. Процедуры нахождения и выполнения хода

Оставшиеся процедуры позволяют находить возможные ходы и выполнять очередной ход. Зададимся вопросом: как проверить, что на данное свободное поле можно поставить свою фишку? Видимо, надо от данного поля двигаться по очереди во всех возможных направлениях и проверять для каждого из них, стоит ли рядом с данным полем одно или несколько полей с чужими фишками, а затем поле со своей фишкой. Тогда возникает следующий вопрос: сколько существует возможных направлений и чем определяется каждое из них? На *рис.* 4 показаны 8 возможных направлений, и каждое из них определяется приращениями индексов, т.е. парой (*di*, *dj*). Прибавив каждое из этих приращений к соответствующему индексу поля, получим индексы соседнего поля по данному направлению. Можно сказать, что пара (*di*, *dj*) — это вектор направления.

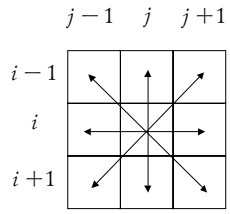


Рис. 4

В таблице представлены все направления и соответствующие приращения индексов.

№	Направление	*di*	*dj*
1	вверх, влево	−1	−1
2	вертикально вверх	−1	0
3	вверх, вправо	−1	1
4	влево по горизонтали	0	−1
5	вправо по горизонтали	0	1
6	вниз, влево	1	−1
7	вертикально вниз	1	0
8	вниз, вправо	1	1

Нам понадобится массив — константа направлений `aDir[1:8]`. Он имеет тип поля и содержит все 8 направлений, указанных в таблице.

Учитывая важность этой процедуры, дадим подробное изложение алгоритма.

```
Находим индексы поля F: i = F.i; j = F.j
Переменной OK присваиваем начальное значение ложь.
Вспомогательной переменной St присваиваем значение 1, что служит признаком первого
   шага по направлению.
Полагаем: i = i1; j = j1. Здесь i1, j1 – индексы очередного поля.
Находим приращения направления di = aDir[Nd].i;  dj = aDir[Nd].j
Начало цикла движения по направлению. Сдвигаемся на один шаг:
   i1 = i1 + di; j1= j1 + dj
Выполняем проверки:
если очередное поле лежит за границами доски,
   то выход из цикла;
если очередное поле — свободное (т.е. rvr[i1, j1]=0)
   то выход из цикла;
если очередное поле — чужое, т.е. rvr[i1, j1]= —aSign[FS]
   и шаг — первый (St=1), то полагаем St=2.
Это является признаком того, что все последующие шаги не первые.
если очередное поле — свое, т.е. rvr[i1, j1]= aSign[FS]
   и шаг — первый (St=1),
то выход из цикла;
если очередное поле — свое, т.е. rvr[i1, j1]= aSign[FS]
   и шаг — не первый (St=2), то:
   полагаем OK = истина; MyF.i = i1; MyF.j = j1; выход из цикла.
конец цикла движения по направлению
конец алгоритма
```

Процедура Check лежит в основе следующих процедур, в которых находятся и выполняются ходы. Она, имея на входе заданное свободное поле F и индекс направления Nd, определяет, можно ли, двигаясь по этому направлению, пройти через чужие фишки к своей. Положительный ответ означает, что на данное поле можно поставить свою фишку. В этом случае выдается поле MyF, на котором находится своя замыкающая фишка. Ответ (положительный или отрицательный) хранит логическая переменная OK.

Отметим, что возможен еще один случай: очередное поле — чужое:

```
rvr[i1, j1]= -aSign[FS])
```

и шаг — не первый (St=2). Тогда надо продолжить цикл. Специальной проверки этого случая не требуется.

Процедура GetList создает список возможных полей, т.е. определяет их число Clist и заполняет массив lst.

В общих чертах алгоритм заключается в следующем.

Счетчику Clist даем начальное, нулевое значение. Просматриваются все поля доски. Для каждого свободного поля выполняется цикл направлений. В этом цикле производятся обращение к процедуре Check и проверка результата. Если он оказался успешным, то:

- наращиваем значение счетчика Clist;
- в позицию Clist списка lst заносим проверенное свободное поле;
- выходим из цикла направлений (достаточно найти одну линию, проходящую от свободного поля к своей фишке через чужие).

Процедура Revers, имея на входе выбранное возможное поле F, ставит туда свою фишку, а также "переворачивает" чужие фишки. Все эти изменения отражаются в массиве rvr.

Принципиальный алгоритм выглядит следующим образом.

Выполняется цикл направлений. В этом цикле производятся обращение к процедуре Check с входным полем F и проверка результата. Если он оказался успешным, т.е. OK = истина, то:

— полагаем F1=MyF. Напомним, что MyF — это поле, на котором находится своя замыкающая фишка;

— выполняем цикл переворота. В нем осуществляется движение от поля MyF к полю F и на каждом шаге выполняется "переворот", т.е. присваивание

```
rvr[i1, j1]= aSign[FS],
```

где i1, j1 — индексы очередного поля.

При этом направление движения противоположно текущему: приращения индексов составляют:

```
di = -aDir[id].i;  dj = -aDir[id].j,
```

где id — порядковый номер направления.

Процедура Chang выполняет ход. Ее входной параметр iList — это индекс выбранного поля в массиве lst.

Выполняются действия:

Находим выбранное поле F = lst[iList]. С этим полем обращаемся к процедуре Revers, а затем — к процедуре DrawAll.

Высвечивается доска так, как она выглядит после выполнения хода.

Процедура Chang получилась очень короткой. Стоило ли выделять процедуру Revers? Дело в том, что она нам пригодится на следующем этапе.

3.5. Основной алгоритм

Теперь мы можем перейти к основному алгоритму программы, который изложим подробно.

```
Элементы массива rvr получают значения, соответствующие начальному состоянию доски:
rvr[4,4]=1; rvr[5,5]=1; rvr[5,4]=—1; rvr[4,5]=—1
Остальные элементы имеют нулевые значения.
Признак очереди хода FS получает значение 2, поскольку в цикле игры он меняет свое
   значение на противоположное.
Вызывается процедура показа доски DrawAll.
Начало цикла игры
   Признак очереди хода FS меняет свое значение на противоположное:
   если FS = 1 то FS = 2
   иначе FS = 1
Индекс списка возможных ходов iList получает начальное значение: iList = 1
Вызывается процедура получения списка возможных ходов GetList.
Если Clist=0, то ходить некуда. В этом случае:
   признак очереди хода FS меняет свое значение на противоположное.
   Вызывается процедура получения списка возможных ходов GetList.
   если Clist=0, то нет ходов у обоих партнеров:
Выход из цикла игры.
Показ первого возможного хода: вызывается процедура PossMove(1).
```

```
С помощью функции Rkey находится значение управляющей клавиши Key.
Выполняется цикл показа возможных ходов.
Пока Key — клавиша показа:
   "Чистим" очередное возможное поле F = lst[iList], используя процедуру ClrField(F)
   Наращиваем индекс списка возможных ходов iList
   Если iList>Clist, то iList=1.
   Показываем очередной возможный ход, вызывая процедуру PossMove(iList)
   Находим новое значение управляющей клавиши Key
   Выполняется очередной ход: вызывается процедура Chang(IList)
Конец цикла игры.
Ожидаем нажатия любой клавиши
Конец алгоритма
```

4. Этап 2 — программа игры в реверси для человека и компьютера

Составив и опробовав программу 1, переходим к основному этапу работы — составлению программы 2 для игры в реверси, в которой вторым партнером ("черными") является компьютер. Понятно, что она является модификацией и развитием программы 1. Все основные данные, а также процедуры и функции этой программы переходят без изменений в программу 2.

Единственное изменение связано с константами KeyI и KeyM. Ясно, что компьютеру не нужны управляющие клавиши, поэтому эти константы уже не являются массивами. Соответственно модифицируется функция Rkey и меняются пояснения, которые высвечиваются процедурой DrawAll.

Чтобы программа выбирала некоторый ход среди своих возможных ходов, надо сформулировать правило выбора. Естественно стремиться к выбору наилучшего хода, а тогда ходы надо как-то оценивать. Следовательно, программа должна содержать *функцию оценки эффективности хода*. Эта функция должна для каждого возможного поля находить его числовую оценку, или *ранг*. Разумеется, лучшему ходу соответствует больший ранг. Имея ранги возможных ходов, можно найти ход с максимальным рангом, его и надо выполнять.

4.1. Функция оценки эффективности хода

Для выбора функции оценки эффективности хода используем "эвристический" подход. Это означает, что наш выбор основан на практическом опыте и здравом смысле, но он не опирается на какую-либо теорию и скорее всего не является самым лучшим.

Будем считать, что ранг анализируемого хода является суммой двух слагаемых: числа переворачиваемых фишек и оценки позиции. С первым слагаемым все ясно: из правил игры следует, что чем больше на очередном ходе оказалось своих фишек, тем лучше. А как оценить позицию? Опыт игры свидетельствует, что наиболее выгодными позициями являются, во-первых, четыре угла доски, а во-вторых, четыре ее крайние линии. Это связано с тем, что фишка, поставленная в угол, уже не перевернется, а возможность переворота фишек в остальных полях крайних линий ограничена.

Обозначим оценку позиции через ValPos, а кроме того, используем еще ряд обозначений:

- i, j — индексы строки и столбца поля F, куда предполагается поставить фишку;
- ir, jr — те же индексы, но отсчитанные от угла доски, ближайшего к F;
- inear — меньший из индексов ir, jr.

Предлагаемая формула для оценки позиции имеет вид:

```
ValPos = (8 — ir — jr) + (3 — inear)
```

Так, на *рис.* 5 крестиком обозначено поле, на которое "черные" предполагают поставить фишку. Для него:

```
i = 5, j = 6, ir = 4, jr = 3,
iк = 8, jк = 8, inear = 3.
```

Подсчет по приведенной формуле дает оценку позиции ValPos = 1.

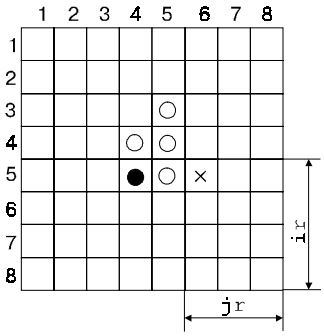


Рис. 5. Возможный ход "черных"

Нельзя сказать, что данная формула вытекает из сказанного выше о выгодных позициях, но она все-таки согласуется с ним. Первое слагаемое отражает тот факт, что позиция улучшается при приближении к углу, а второе — что то же самое имеет место при приближении к крайней линии.

Надо признаться, что в предлагаемой оценке эффективности хода никак не учитывается ответный ход противника, и это ее большой недостаток. Но все же в двух случаях такой учет выполняется. С первым из них

мы встречаемся сейчас. Пусть поле F лежит во второй линии от края доски. Это означает, что выполняется одна из двух альтернатив:

а) ir = 2 и jr ≠ 1

б) jr = 2 и ir ≠ 1.

Тогда надо проверить три поля, находящихся на крайней линии доски и соседних с F. Если противник имеет возможность поставить свою фишку на какое-либо из этих полей, то он прорвался на крайнюю линию, что нежелательно. В таком случае уменьшаем оценку позиции ValPos на две единицы: ValPos = ValPos – 2. На *рис.* 6 показан крестиком возможный ход "черных" на поле (2, 7), т.е. на вторую линию, а вопросительными знаками помечены поля на первой горизонтали, соседние с полем (2, 7). Видно, что "белые" ответным ходом могут поставить свою фишку на поле (1, 7), т.е. на первую горизонталь.

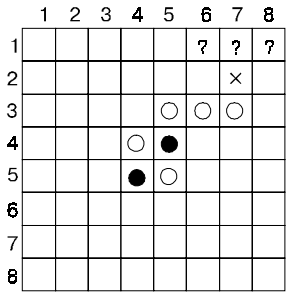


Рис. 6

Мы рассмотрим еще один особый случай, который может иметь место в начальной стадии игры, когда при очередном ходе "черных" (т.е. компьютера) у них на доске всего одна фишка. "Черные" ставят свою фишку, и теперь у них несколько фишек, расположенных подряд. Может случиться, что эти фишки ограничены с одной стороны фишкой "белых", а с другой — свободным полем. Тогда "белые" ответным ходом ставят свою фишку на окаймляющее свободное поле, все фишки "черных" переворачиваются — и они проигрывают "всухую". Пример такой ситуации представлен на *рис.* 7. Вопросительным знаком помечено поле (4, 6), куда "черные" могут поставить свою фишку. Если они это сделают, то белые в ответ поставят свою фишку на поле (4, 7) и выиграют.

Такому ходу даем самый низкий ранг, а конкретно — 8.

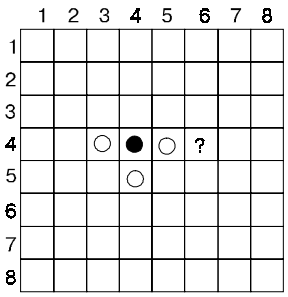


Рис. 7

4.2. Алгоритм определения ранга

Переходим к алгоритму вычисления функции оценки эффективности хода. К перечню основных данных добавляем массив рангов aRank[1:64]. Заносить в этот массив ранги возможных ходов будет процедура GetRank. В свою очередь, она обращается к функции ValPos, вычисляющей оценку позиции, и функции Danger1, которая анализирует случай, когда у "черных" только одна фишка. Начнем с первой из этих функций.

Функция ValPos, как уже сказано, находит оценку позиции. Ее входной параметр — возможное поле F.

Приведем алгоритм функции ValPos.

```
Определяем индексы i, j поля F.
Теперь надо найти индексы ik, jk угла доски, ближайшего к F, а также относительные
индексы ir, jr поля F, отсчитанные от этого угла. Сначала полагаем:
ik =1;  jk =1; ir =1;  jr =1
Затем:
если i>4 то нач ir = 9 — i; ik =8 кон
если j>4 то нач jr = 9 — j; jk =8 кон
Находим меньший из индексов ir, jr.
если  ir<jr то inear = ir иначе inear = jr
По формуле, приведенной выше, находим оценку позиции:
   rank = (8 — ir — jr) + (3 — inear)
Выше указывалось, что если возможное поле F находится на второй линии с края доски,
   то надо проверить, не может ли противник, воспользовавшись нашим ходом,
   поставить свою фишку на край доски. Условие нахождения поля F на второй линии:
   (ir = 2 и jr ≠1) или (jr = 2 и ir ≠1)
Если оно выполняется, то организуем проверку массива F3[1:3], состоящего из трех
   соседних с F полей, находящихся на краю доски. При этом учитываем, что поле F
   может находиться рядом с углом доски, т.е. возможен случай ir = jr = 2.
```

```
Если вторая линия — горизонталь:  ir = 2, то Формируем массив F3, т.е.
   в цикле по i3 от 1 до 3 полагаем: F3[i3].i = ik; F3[i3].j = j+i3-2
   Проверяем массив F3:
   если Check3(F,F3) то: ValPos= rank — 2; выход из процедуры
Если вторая линия — вертикаль: jr = 2, то:
   Формируем массив F3, т.е.
   в цикле по i3 от 1 до 3 полагаем: F3[i3].i = i+i3-2; F3[i3].j = jk
   Проверяем массив F3:
   если Check3(F,F3) то: ValPos= rank — 2; выход из процедуры
Проверка закончена.
```

Здесь функция Check3 осуществляет контроль массива F3. Если на одно из полей F3 можно поставить фишку, то выдается значение истина, а иначе — значение ложь. Эту функцию мы рассмотрим ниже.

Последняя операция: ValPos=rank. Она будет выполняться, если поле F не находится во второй линии или проверка закончилась благополучно.

Функция Check3 имеет входные параметры: поле F на второй линии и массив F3 его соседних полей на крайней линии. Она выдает значение истина, когда на одно из полей массива F3 "белые" могут поставить свою фишку, и ложь в противном случае. Алгоритм функции Check3:

```
Определяем индексы i, j поля F.
Переменной OK даем начальное значение ложь.
В цикле по i3 от 1 до 3 просматриваем поля F3[i3].
Для каждого свободного поля (F3[i3] = 0)выполняем следующие действия:
Находим приращения индексов
di = i —F3[i3].i; dj = j —F3[i3].j
Просматривая массив направлений aDir, находим индекс направления id,
   соответствующий найденным приращениям.
Копируем массив rvr во вспомогательный массив rvr1.
Ставим свою ("черную") фишку на поле F, выполняя обращение Revers(F).
 Передаем очередь хода партнеру ("белым"): FS = 1.
 Проверяем возможность поставить "белую" фишку на поле F3[i3], выполняя обращение
   check(F3[i3],id,OK,MyF).
Восстанавливаем очередь хода: FS = 2.
Восстанавливаем массив rvr, копируя в него массив rvr1.
Если OK = истина,
   то есть возможность поставить белую фишку на поле F3[i3].
      Функции Check3 присваиваем значение OK и выходим из нее.
Конец цикла по i3.
Последняя операция: Check3 = OK.
Она будет выполняться, если ни на одно из полей F3 "белую" фишку поставить нельзя.
```

Функция Danger1, имея на входе возможное поле F, проверяет, можно ли на него ставить свою фишку, для случая, когда на доске только одна своя ("черная") фишка. Она выдает значение истина, если ставить свою фишку нельзя, а иначе ложь.

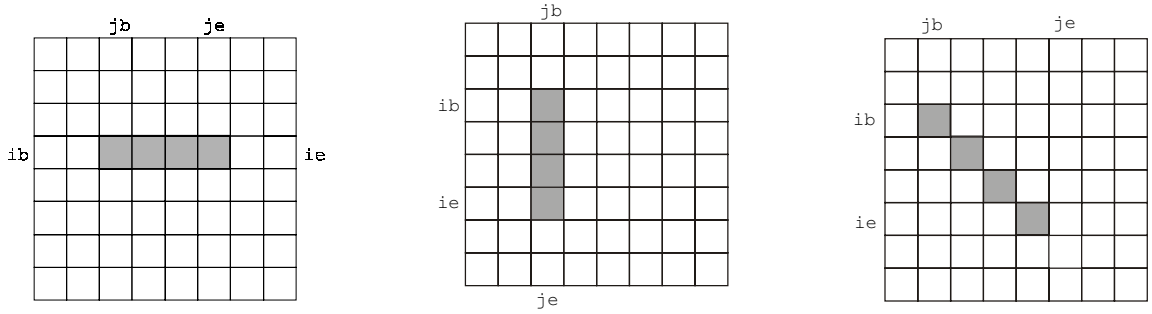
Алгоритм функции Danger1:

```
Переменной Dang присваиваем начальное значение ложь.
Определяем индексы i, j поля F.
Просматривая массив rvr, находим индексы i1, j1 поля F1 со своей (единственной)
   фишкой из условия rvr[i1,j1]= —1.
Находим разности индексов di = i1 — i;  dj = j1 — i.
Нормируем разности индексов:
если di ≠ 0 то = di /abs(di)
если dj ≠ 0 то = dj /abs(dj)
  Теперь di, dj — приращения индексов, позволяющие сделать единичный шаг по
   направлению от поля F к F1.
Определяем индексы ie, je поля Fe, замыкающего линию F — F1 и соседнего с полем F1:
   ie = i1 + di;  je = j1 + dj
```

Аналогично определяем индексы ib, jb поля Fb, замыкающего линию F — F1 и соседнего с полем F: ib = i — di; jb = j — dj.
Если оба поля, Fe и Fb, находятся в пределах доски, то
выясняем, не является ли одно из этих полей
свободным, а другое содержит чужую ("белую") фишку.
Этот факт имеет место при выполнении равенства: rvr[ib,jb] + rvr[ie,je] = 1
В таком случае полагаем Dang = истина.
Функции Danger1 присваиваем значение Dang.

Процедура GetRank завершает эту серию средств оценки хода. Для каждого возможного хода "черных" находится его ранг, который заносится в массив aRank.

Сначала сделаем следующее замечание. Пусть нас интересует число полей nF, расположенных подряд по некоторому направлению. Обозначим через ib, ie индексы строк, содержащих концевые поля jb, je — индексы соответствующих столбцов, см. *рис.* 8.



а — горизонталь *б* — вертикаль *в* — диагональ

Рис. 8

Тогда число полей nF выражается следующим образом:
если линия — горизонталь (т.е. ib = ie), то nF = abs(jb — je) + 1;
если линия — вертикаль (т.е. jb = je), то nF = abs(ib — ie) +1;
если линия — диагональ, то nF = abs(jb — je) + 1 = abs(ib — ie) + 1.

Теперь сформулируем алгоритм процедуры GetRank.

В цикле по iL от 1 до Clist выполняются следующие действия (напомним, что Clist — это число возможных ходов):
Получаем очередное возможное поле F = lst[iList].
Определяем индексы i, j поля F.
Находим оценку позиции rank = ValPos(F).
В цикле по индексу направлений id от 1 до 8:
Выполняем обращение Check(F,id,OK,MyF). Это позволит проверить, можно ли на поле F поставить свою фишку.
Если можно (т.е. OK = истина), то:
находим число чужих ("белых") фишек, расположенных
в полях между F и MyF.
Если aDir[id].i ≠ 0, то
чужие фишки находятся на вертикали или диагонали. В этом случае их число составляет: nw = abs(MyF. i — i) — 1
Иначе чужие фишки находятся на горизонтали и их число определяется аналогичным образом: nw = abs(MyF. j — j) — 1
Добавляем число переворачиваемых фишек к оценке позиции rank = rank + nw
Конец цикла по индексу направлений id.
Если на доске одна "черная" фишка (т.е. k2 = 1), то проверяем, не является ли опасным возможное поле F:
если Danger1(F) = истина, то полагаем rank = —8.
Заносим rank в массив aRank[iL]=rank.
Конец цикла по iL

4.3. Сценарий и основной алгоритм программы 2

По-прежнему человек, играющий в реверси, должен иметь возможность просмотреть свои возможные ходы, прежде чем выбрать какой-то из них. Более того, ему надо дать средство просмотра возможных ходов партнера. Тогда он сможет сравнить ход компьютера с ходом, который он выбрал бы на его месте. Конечно, для просмотра своих и компьютерных возможных ходов достаточно одной клавиши просмотра, и то же имеет место для клавиши выполнения хода. Когда выполняется ход компьютера, то удобно поступить так: "черная" фишка высвечивается на поле, которое выбрал компьютер, и программа приостанавливается до нажатия любой клавиши. При отсутствии указанной паузы не всегда ясно, какое поле выбрал компьютер, и иногда кажется, что его ход некорректен. На *рис.* 9 показана позиция, в которой ход принадлежит "черным" (т.е. компьютеру) и они его выбрали.

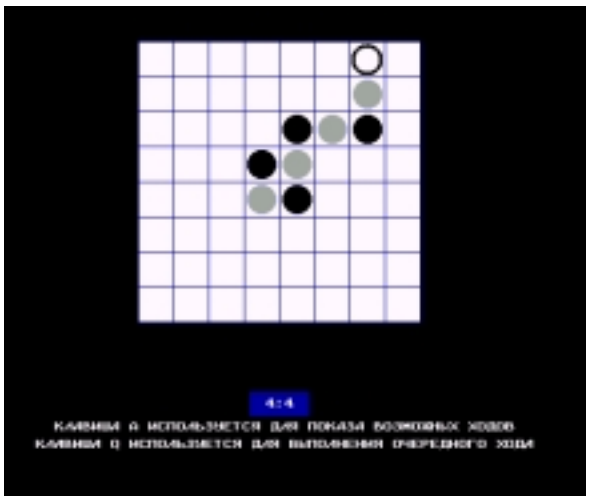


Рис. 9

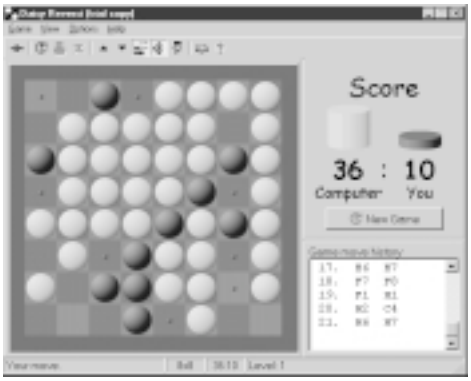
Разумеется, программа 2 содержит процедуру `GetRank`, а также процедуры и функции, к которым она обращается. Все они приведены выше в разделе 4.2.

Как уже говорилось, почти все процедуры и функции программы 1 переходят без изменений в программу 2. Небольшие изменения, касающиеся функции `Rkey` и процедуры `DrawAll`, отмечены выше. Поэтому нам осталось рассмотреть лишь основной алгоритм программы 2, а точнее, его сердцевину — *цикл игры*, поскольку начальная и заключительная стадии этого алгоритма остаются без изменений. Даже и в цикле игры большая часть действий остается неизменной, но все же мы приведем его целиком.

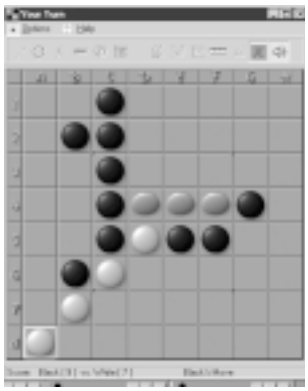
```
Начало цикла игры
Признак очереди хода FS меняет свое значение на противоположное:
если FS = 1 то FS = 2 иначе FS = 1.
Индекс списка возможных ходов iList получает начальное, единичное значение.
Вызывается процедура получения списка возможных ходов GetList.
Если Clist=0, то ходить некуда. В этом случае:
   Признак очереди хода FS меняет свое значение на противоположное.
   Вызывается процедура получения списка возможных ходов GetList.
Если Clist=0, то нет ходов у обоих партнеров: Выход из цикла игры.
Показ первого возможного хода: вызывается процедура PossMove(1)
С помощью функции Rkey находится значение управляющей клавиши Key.
Выполняется цикл показа возможных ходов. Пока Key — клавиша показа:
   Чистим очередное возможное поле F = lst[iList], используя процедуру ClrField(F)
   Наращиваем индекс списка возможных ходов iList
   Если iList>Clist, то iList=1.
   Показываем очередной возможный ход, вызывая процедуру PossMove(iList).
   Находим новое значение управляющей клавиши Key.
 Если очередной ход человека ("белых"), т.е. FS = 1, то:
   Чистим выбранное поле F = lst[iList], используя процедуру ClrField(F).
   Выполняем ход, обращаясь к процедуре Chang(IList)
  Иначе очередной ход компьютера ("черных"), т.е. FS = 2:
   Находим последнее возможное поле F = lst[iList] и чистим его, используя
        процедуру ClrField(F).
   Определяем ранги возможных полей, вызывая процедуру GetRank.
   Просматривая массив aRank, находим индекс возможного поля iList с максимальным
   рангом,т.е. индекс хода.
   Показываем найденный ход, вызывая процедуру PossMove(iList).
   Ожидаем нажатия любой клавиши.
   "Чистим" выбранное поле F = lst[iList], используя процедуру ClrField(F).
   Выполняем ход, обращаясь к процедуре Chang(IList).
конец если
Конец цикла игры
```

Заключение

Итак, мы составили и опробовали программу игры в реверси между человеком и компьютером. Что можно сказать об игре компьютера? Видимо, ее можно оценить как среднюю. Если постараться, то выиграть у компьютера можно. А как усилить его игру?

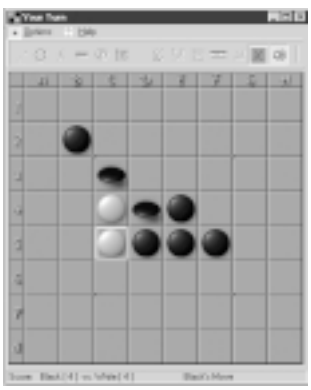


Выше уже указывался недостаток предложенной оценки эффективности хода: в ней, за редкими исключениями, не учитывается ответный ход противника. Можно попытаться учесть это обстоятельство и в функцию оценки хода включить число перевернутых фишек при ответном ходе "белых", разумеется, с отрицательным знаком. На каждый ход компьютера ("черных") возможны различные ответы "белых"; какой из них надо учитывать при оценке эффективности хода "черных"? Надежный подход требует принять во внимание ход, наилучший для противника. Это означает, что из всех возможных ответов "белых" надо учесть тот, при котором ранг хода минимальный. Отметим, что ответ белых может повлиять и на оценку позиции в том случае, когда поставленная фишка при ответном ходе белых оказалась перевернутой.

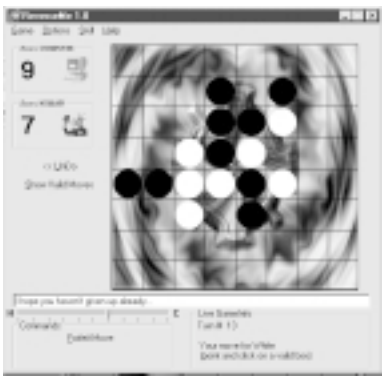


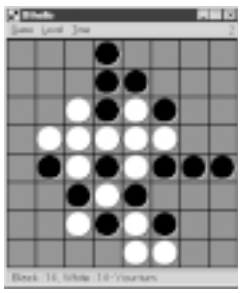
Возможно, что у тех, кто активно участвовал в этом проекте, появятся и другие идеи. Попробуйте их реализовать!

В заключение расскажем о подходе к оценке эффективности хода, основанном на теории игр. Эта глубокая и содержательная математическая теория исследует процессы, связанные с конфликтом интересов. Для таких игр, как реверси, она предлагает метод минимакса, см. [2, 3]. Идея указанного метода заключается в том, чтобы учитывать отдаленные последствия оцениваемого хода и при этом принимать во внимание ходы противника, наилучшие для него.



Применительно к реверси рангом оцениваемого хода будет число "черных" фишек, которые окажутся на доске после выполнения заданного числа N пар ходов "черных" и "белых". Выполнять надо ход с максимальным рангом. Мы полагаем, что читатель достаточно подготовлен, чтобы самостоятельно разобраться в случае $N = 1$. Пусть теперь $N = 2$. Обозначим оцениваемый ход "черных" через b_1, а один из возможных ответных ходов "белых" — w_1. Если выполнены оба хода — b_1 и w_1, то мы находимся в условиях уже знакомого случая $N = 1$. Находим наилучший для этой ситуации ход "черных". Его ранг — это и есть ранг хода b_1 при ответе w_1. Варьируя ответы "белых" — w_1, определяем тот из них, при котором ранг будет минимальным, — это ранг хода b_1. Выполняя описанную процедуру для всех возможных ходов b_1, получаем выполняемый ход с максимальным рангом. Если $N = 3$, то после выполнения первой пары ходов — b_1 и w_1, приходим к случаю $N = 2$ и действуем аналогичным образом и т.д. Наглядное представление метода минимакса связано с т.н. "деревом игры" [3]. Строго говоря, метод минимакса требует довести глубину анализа вплоть до заключительной позиции. На практике ограничиваются фиксированной глубиной. Она определяется ресурсами памяти и быстродействия, а также сложностью алгоритма. При достаточном N уровень игры получается высоким. И, наконец, общий вывод, с которым, надеемся, читатель согласится: программирование игр — это серьезное и в то же время увлекательное занятие.





Литература

1. *Гик Е., Степанов О.* Реверси. Квант, 1991, № 1.

2. *Каспер Е.* Освоим QBasic играючи! М.: Горячая линия — телеком. Радио и связь, 1999.

3. *Карпов А.Е., Гик Е.Я.* Шахматный калейдоскоп. М.: Наука, 1982.

В оформлении страницы использованы экраны различных программ для игры в реверси.

Современные форматы графических файлов

Вечера с Вовой

А.Г. Леонов

Продолжение. Начало в № 17, 18/99

2. Все познается в сравнении. Вечер второй

Вова сидел у меня на кухне и уплетал печенье "Юбилейное", запивая его чаем. Я расположился напротив с ноутбуком в руках. Мальчик с любопытством наблюдал за моими действиями. Его очень расстраивало, что ему не видно экрана. Я подсел к Вове поближе.

— Вчера мы обсуждали с тобой самый простой формат — BMP.

Вова понимающе кивнул головой.

— А теперь давай составим таблицу, в которой будем фиксировать наиболее важные сведения о форматах:

Формат	BMP	PCX	GIF
Число бит/пиксель	24		
Макс. число цветов, *млн*	16,7		
Размер изображения в пикселях	65 535 × 65 535		

— Мы с тобой знаем, что каждая точка рисунка может кодироваться 24 битами.

— Это если true color! — уточнил Вова.

— Да. А если черно-белое изображение, то...

— Один бит, — подхватил прилежный ученик.

— Поэтому в графу таблички "Число бит на пиксель" пишем 24 бита. Ты уже подсчитал, сколько цветов может быть в изображении — 2^{24}. А теперь заполним поле "Размер изображения". Так как изображение состоит из пикселей, то и размер измеряется в пикселях (точках). Максимальный размер рисунка в BMP-файле — $(2^{16}-1)\times(2^{16}-1)$.

Задача 2.1. *Почему размер по горизонтали равен $2^{16}-1$, а не 2^{16}?*

Пояснение. Формат графического файла — это не незыблемый монумент, и время накладывает на формат определенный отпечаток. Все сказанное выше верно для PCX-формата, используемого в Windows 3.1. Однако появление Windows 95 и 98 привнесло в формат некоторые изменения: число бит на точку может достигать 32, а размер изображения с 2-байтного вырос до 31-битного, то есть до 2 Гигабит. Размер также может задаваться отрицательным числом, которое указывает на направление прорисовки изображения: с левого верхнего угла, а не с левого нижнего.

— Дядя Саша, а почему остальные клетки таблицы не заполнены? Ты расскажешь про форматы PCX и GIF?

— Конечно, уже начинаю. Формат графических файлов не возникает на пустом месте. Кто-то первый придумывает новый стандарт и предлагает его к использованию. Так, однажды фирма ZSoft разработала стандарт PCX для использования в программе PC Paintbrush в операционной системе MS-DOS.

— Это та, которая была до Windows? — спросил Вова.

— Да. Программа Paint на твоем компьютере — это обновленная PC Paintbrush. Но PCX не стал столь популярным форматом, как BMP, поэтому в Windows 98 программа Paint уже не "понимает" этого формата. Но если у тебя сохранились файлы в формате PCX, то можно прочитать их программой Imaging, которая поставляется вместе с Windows 98. Нельзя же бросать то, что уже сделано! Тем не менее PCX был распространен на PC-технике, и многие современные умные программы (специализированные графические редакторы CorelDraw, Ulead PhotoExpress, Adobe PhotoDeluxe) умеют его читать и записывать.

— Я работал в CorelDraw, — заметил ученик.

— PCX — это **аппаратно-зависимый** формат. Изначально он разрабатывался для определенного типа видеоплат (видеоадаптеров). Чтобы в файле информация хранилась так же, как и в видеоплате.

Раньше существовали 16-цветные EGA-адаптеры. Память таких адаптеров делилась на непрерывные куски — плоскости, планы. Составляющие цвета пикселя (как правило, 1 бит) находились в соответствующих местах плоскостей. Для того чтобы поставить точку определенного цвета, требовалось исправить в 4 плоскостях по 1 биту. Получалось, что цветная картинка как бы состояла из 4 наложенных монохромных (не путать с черно-белыми). Для поддержания совместимости современные видеоадаптеры умеют работать и в таких EGA-режимах.

Представь на минутку, что графический файл — это грузовики, везущие бочки-байты на склад (видеокарту). При поступлении на склад содержимое бочек должно быть разнесено по соответствующим полкам. Это требует определенного времени: открыть бочку, достать какую-то часть, переложить на полку и т.д. А если заранее распределить по бочкам так, чтобы все из бочки выкладывать на одну полку, то это сильно сэкономит время. Это и есть аппаратная зависимость.

— А на сколько полок нужно раскладывать содержимое бочки? — поинтересовался Вова.

— Это был просто пример, поэтому можешь считать, что на 4 или на 2. Теперь вернемся к формату PCX. Как и в BMP-формате, файл имеет заголовок длиной 128 байт. Возьмем файл с фотографией кукол: DOLLS.BMP размером 1 440 054 байта и преобразуем его в формат PCX. Для этого я воспользуюсь достаточно старой программой iPhoto Plus (у современных графических программ, которые установлены на моем компьютере, не оказалось возможности сохранять файлы в этом формате). А сейчас посмотри на размер файла.

— Он равен 1 375 127 байтам, — сообщил Вова.

— К какому выводу можно прийти?

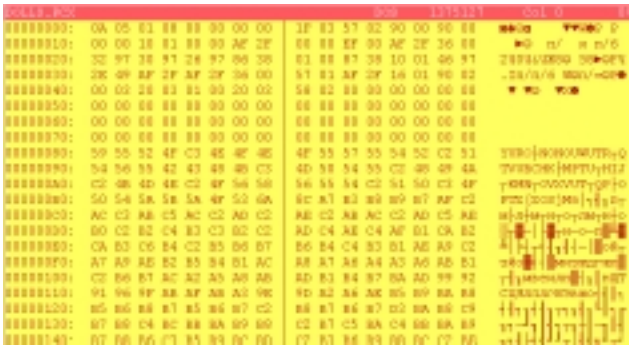
— PCX-формат компактнее, — догадался Вова.

Задача 2.1. *Как вы думаете, почему?*

— Теперь воспользуемся программой HEX-viewer и посмотрим, что там внутри. Не забыл, как переводить из шестнадцатеричной системы счисления в десятичную? Сколько будет в десятичной системе 0A шестнадцатеричное?

— Десять, — подумав, ответил Вова.

— Правильно, именно с этой десятки и начинается PCX-файл. Код 0A означает, что это PCX-формат. Далее через байт следуют номер версии (05), количество бит на пиксель (в данном файле — 8) и т.д. Но если обнаружишь в начале файла признак 0A, то все равно не надо сразу считать, что это PCX-файл. Надо очень внимательно проанализировать 128-байтный заголовок.



— А теперь составь таблицу, какую мы составляли для BMP-файла, я тебе помогу.

— Давай немного ее проанализируем. Появились новые понятия — **левая, правая, верхняя, нижняя граница** изображения. Дело в том, что PCX-формат допускает хранение в растровом массиве большего изображения, чем картинка, видимая на экране. Левая и верхняя граница задают отступ от верхнего угла изображения, а разница между границами задает размер видимой части.

Это было придумано для использования PCX-формата в факсимильных аппаратах.

— Кроме того, ты помнишь, что строки BMP-файла иногда содержали "лишние" байты в конце строки растра?

— Да, если число байт в строке нечетно, там добавляется "выравнивающий" байт, — вспомнил Володя.

— Правильно! Заметь, что пиксели, в которых измеряются границы изображения, нумеруются с 0, поэтому правая граница — 031F. Сколько это будет в десятичной системе счисления?

— Это будет 3 умножить на 256 плюс 16 плюс 15, то есть 799!

— А нижняя граница, соответственно, будет 599. То есть размер изображения — 800 на 600 точек. Есть еще интересный байт в файле, определяющий метод сжатия. Он, как правило, установлен равным 1, что указывает на дополнительную упаковку изображения. Но об этом поговорим в другой раз. Теперь вспомни пример с "бочками". В таблице написано, что в файле DOLLS.PCX используется 8 бит на пиксель. То есть...

— 256 цветов, — подхватил Вова.

— Да, но смотри, что сообщает про этот файл программа PhotoImpact Viewer (программа для просмотра графических файлов различных форматов).

Тип изображения:	RGB True color (24 bit)
Ширина:	800 Pixels
Высота:	600 Pixels
Размер:	1343 KB
Имя:	DOLLS.PCX
Формат:	PC Paintbrush
Сжатие:	RLE compression
Разрешение:	57 Pixels/cm

— A true color — это...?

— 24 бита, — сообщил ученик.

— Правильно. А теперь посмотри на строку "Количество планов" в таблице, которую мы составили. Там стоит число 3. Это означает, что информация о точке хранится в трех различных растровых массивах, по 8 бит в каждом, то есть 3 умножить на 8, получишь те же 24 бита на пиксель. При этом в файле DOLLS.PCX растровые данные хранятся по планам в

Название	Длина (в байтах)	Содержимое файла DOLLS.PCX	Примечание
Заголовок файла			
Признак PCX	1	0A	
Номер версии	1	05	самая последняя
Метод сжатия	1	01	упакованы
Бит/пиксель	1	08	8 бит
Левая граница	2	00 00	0
Верхняя граница	2	00 00	0
Правая граница	2	03 1F	799
Нижняя граница	2	02 57	599
Горизонтальное разрешение	2	00 90	144
Вертикальное разрешение	2	00 90	144
16-цветная палитра	48	00 00 10 ...	16 · 3 = 48
Резервный байт	1	00	
Количество планов	1	03	
Кол-во байт в строке	2	03 20	800
Тип палитры	2	00 01	цветное изображение
Ширина экрана	2	03 20	800
Высота экрана	2	02 58	600
Резервные байты	54	00 — 54 раза	
Изображение			
Растровый массив	переменная	59 55	здесь хранится само изображение
Палитра			
Признак палитры	1	C0	иначе 256-цветная
256-цветная палитра	768		палитра отсутствует

трех растровых массивах, следующих один после другого. А если бы количество планов было равно единице, то...

— В одном массиве, как было в BMP-файле, — подытожил Володя.

— Да, только при true color в PCX-файле всегда 3 плана. А один план бывает при 256-цветном изображении. Именно тогда в конце файла находится палитра. Кроме того, в таблице есть два поля, описывающих разрешение. Для более правильной привязки изображения к экрану: 144 точки на дюйм. А программа PhotoImpact Viewer сообщает, что 57 точек. Как ты думаешь, почему?

Володя задумался на несколько секунд и ответил.

Задача 2.3. *Что ответил Володя?*

— Теперь мы можем заполнить очередные графы нашей таблицы.

Формат	BMP	PCX	GIF
Число бит/пиксель	24	24	
Макс. число цветов, *млн*	16,7	16,7	
Размер изображения в пикселях	65 535 × 65 535	65 535 × 65 535	

— Дядя Саша, а будем заполнять последний столбец? — спросил Вова.

— Будем, только пойдем прогуляемся с собаками, уже время подошло, и я продолжу.

Мы взяли на поводки моих собак— таксу и добермана — и пошли гулять.

— Итак, следующий формат...

— GIF, — напомнил Вова.

— GIF — это *Graphics Interchange Format* (дословно — Формат для Обмена Графикой). Этот формат, разработанный фирмой CompuServe, стал одним из самых распространенных форматов для хранения и обмена изображениями, хотя первоначально предназначался для сети CompuServe. Подчас придуманное кем-то для внутренних нужд вполне подходит и для всех. Формат GIF устроен не так, как известные тебе PCX и BMP. Говорят, что в GIF графическая информация хранится в виде **потока** данных. Информация состоит из блоков, которые следуют один за другим. Нельзя сказать, где начинается 17-й блок, пока не прочитан 16-й. Поэтому при чтении файла информация "льется как вода", непрерывным потоком. Понятно?

— Понятно, — подтвердил Вова.

— К недостаткам GIF-файла можно отнести скромное количество цветов в картинке — всего 256 цветов, правда, из палитры.

— А true color? — удивился мальчик.

— Никакого true color, максимум 256 цветов и все. Но есть и достоинства. Один файл может содержать произвольное число изображений. Правда, большинство файлов, которые ты встретишь, содержат всего одну картинку. Различают две версии формата GIF: GIF87a и GIF89a... Но нам пора домой. Пойдем, я дома расскажу дальше.

И мы возвратились с прогулки домой. Собаки шли спокойно — хорошо погуляли. Мы снова сели пить чай.

— Сначала про GIF87a. Версия выпущена в мае 1987 года, как ты, наверное, догадался из названия: GIF87— 87-й год. Каждый файл состоит из заголовка, необязательной палитры и собственно информации про картинки. Вот все та же картинка DOLLS, уже в GIF-формате.

— А цветов сколько? — спросил Володя.

— Только 256. Теперь смотри внутрь. Начинается с признака GIF-файла и номера версии "87a".

— Давай снова составим таблицу.

Название	Длина файла (в байтах)	Содержимое DOLLS.GIF	Примечание
Заголовок файла			
Признак GIF	3	47 49 46	GIF
Номер версии	3	38 37 61	87a
Ширина экрана	2	03 20	800
Высота экрана	2	02 58	600
Спец. информация	3	F7 00 00	число цветов и прочее
Глобальная 256-цветная палитра	3 · число цветов	59 55	палитра может отсутствовать
Изображение 1			
Признак картинки	1	2C	
Координаты картинки:			
по оси X	2	00 00	0
по оси Y	2	00 00	0
Ширина картинки	2	03 20	800
Высота картинки	2	02 58	600
Спец. информация	2	08 07	наличие палитр и прочее
Палитра картинки 256-цветная палитра	3 · число цветов		палитра может отсутствовать
Данные картинки	переменная		
Изображение 2			
.....			
Завершитель	1	3B	обязателен

— После заголовка идет палитра, которая используется для всех изображений в файле GIF, если у изображения нет собственной палитры. Если глобальная цветовая палитра отсутствует, то каждое изображение, хранящееся в файле GIF, должно иметь собственную палитру.

— А палитры устроены как в BMP-файлах?

— Да, точно так. Причем и палитры картинок, и глобальная палитра устроены совершенно одинаково: тройки байт, задающие красную, зеленую и синюю составляющие цвета точки. А теперь про данные картинки. Как и у PCX-файла, GIF использует специальный алгоритм сжатия данных, для того чтобы объем дисковой памяти при хранении был поменьше. Я тебе говорил, что мы с тобой поговорим чуть попозже о том, как работают алгоритмы сжатия... Сжатые данные картинки хранятся в виде потока блоков размером не более 255 байт. Каждый блок состоит из байта-длины блока и собственно данных. Когда картинка заканчивается, следует блок нулевой длины.

— А я видел, что GIF-файлы при загрузке из сети Интернет как-то интересно появляются, как будто нечетко сначала, а потом все четче и четче. Почему это? — спросил Володя.

— Хороший вопрос, умница. GIF был задуман для передачи изображений именно по сети. Если ты подключен через модем, то ты наблюдаешь загрузку GIF-файла "вживую". Этот формат "умеет" передавать строки картинки чередующимися: сначала 0, 8, 16, то есть каждая 8-я строка, затем 4, 12, 20 — и снова через 8 строк. Далее со 2-й строки и через 4: 2, 6, 10... И последний проход начинается с 1-й строки и через 2: 1, 3, 5... каждая нечетная. Такое изображение, даже "приехав" на 50%, позволяет увидеть всю картинку, хотя, может, и без деталей. Глаз и мозг дополняют картинку сами, хотя получена всего половина изображения.

— А тогда при нечередующихся строках "приедет" только верхняя половина картинки, и мы еще не знаем, что будет внизу картинки, да?

— Молодец! При этом чередующиеся строки на самом деле лежат в этом порядке в файле GIF. Однако это может создать определенные трудности при перекодировании чередующегося GIF-файла в другой формат. Теперь про GIF89a. Эта версия выпущена в июле 1989 года. Формат GIF89 очень похож на GIF87. Общая структура файла такая же: заголовок, необязательная палитра и информация про картинки. Однако в новой версии формата появились новые информационные блоки, которые названы **управляющим расширением**. Такое вот странное название.

— Я запомню,— улыбнулся Вова.

— Их четыре типа: управление графикой, текстом, комментарии и приложения. Эти блоки созданы для управления выводом графики из GIF-файла и для некоторого другого. А теперь подробнее. Управление графикой используется, когда одна картинка GIF-файла перекрывает другую (может даже прозрачную), когда предыдущую картинку надо удалить (или восстановить) или подождать, пока человек нажмет на кнопку мыши. Блок текста позволяет накладывать предложения обычного текста поверх изображения. Это особенно удобно, когда нужно поменять текст в GIF-файле, оставив неизменным фоновое изображение. Комментарии просто пропускаются при выводе изображения, но могут быть полезны авторам рисунка, например, содержать информацию об авторе рисунка, дате создания и еще что-нибудь полезное. Приложения хранят специальные данные, которые воспринимаются специальными же программами. Другие программы, показывающие GIF-файл, просто пропускают этот блок.

— А как устроены управляющие расширения?— спросил Володя.

— Давай рассмотрим, как устроен блок текста, снова составим таблицу.

Название	Длина (в байтах)	Содержимое файла	Примечание
Признак блока	1	21	
Признак расширения	1	01	текст
Размер блока	1	0C	всегда
Позиция текста по оси X	2	00 14	20
Позиция текста по оси Y	2	в пикселях 00 0A	10
Ширина текста по оси X	2	00 38	56
Ширина текста по оси Y	2	00 0A	в пикселях 10
Ширина буквы	1	08	8
Высота буквы	1	0A	10
Цвет фона	1	07	из палитры
Цвет буквы	1	01	
Текст	от 1 до 255	8F 90 88 82 ...	ПРИВЕТ!
Признак конца блока	1	00	всегда

— Я понял,— сказал Вова, внимательно разглядев таблицу,— этот блок задает текст...

Задача 2.4. *Что рассказал Вова? Ответьте подробно.*

— Теперь мы можем наконец заполнить все графы таблицы:

Формат	BMP	PCX	GIF
Число бит/пиксель	24	24	8
Макс. число цветов, *млн*	16,7	16,7	256
Размер изображения в пикселях	65535×65535	65535×65535	65535×65535

— Уфф! — радостно завершил мальчик, — я знаю про три формата: BMP, PCX и GIF! А поиграть на компьютере можно?

— Садись, Вова, к настольному компьютеру. Выбирай игру, отдохни. А я еще поработаю.

Через пару часов мы распрощались с моим маленьким соседом. Вова был счастлив: и узнал много, и поиграл. Мы договорились скоро встретиться снова.

Продолжение следует

Не все так однозначно...

Продолжение. Начало на с. 1

Для работы на современных компьютерах можно использовать только специальные "четкие" языки, строго описывающие решаемые задачи*. В естественном же языке большинство понятий являются нечеткими. Такое различие между естественным и машинным языками представляет для многих серьезное препятствие при освоении компьютерных технологий.

Одно из значений английского слова *fuzz*, от которого образовано прилагательное *fuzzy* (нечеткий), — это ворс — специальный термин, указывающий свойство тканей. Когда мы разглядываем рисунок на ворсистой ткани, он кажется нам размытым, нечетким...

Нечеткая логика — это дисциплина, оперирующая "размытыми" понятиями, например, "светлый", "быстрый", "теплый", и позволяющая применять правила "здравого смысла". Вот пример, ставший уже почти классическим [1].

Употребляя только традиционные методы теории управления, невозможно обеспечить постановку автомобиля на стоянку между двумя другими автомобилями: нельзя с достаточной степенью точности определить состояние дорожного покрытия, состояние шин и указать параметры уравнений движения. Однако и человек, и нечеткая система способны справиться с этой задачей, пользуясь нечеткими указаниями типа: "Повернуть руль направо, двигаться вперед; возвратить руль налево и остановиться; затем, поворачивая направо, двигаться назад и возвратить руль налево; в случае неудачи повторить".

В классической теории множеств объект либо принадлежит, либо не принадлежит множеству. Число 9 полностью принадлежит множеству нечетных чисел и совсем не принадлежит множеству четных чисел. Объект здесь не может одновременно содержаться в множестве и в дополнении этого множества или не содержаться ни в каком множестве. Для нечетких множеств справедливы иные принципы: воздух может быть на 20 процентов прохладным и в то же время на 80 процентов не быть прохладным.

Подобные соотношения для нечетких систем — это совсем не то же самое, что вероятности. Вероятность — численная мера степени объективной возможности события, то есть численная мера того, произойдет событие или нет. Нечеткость же определяет, так сказать, до какой степени случается событие или выполняется условие. Утверждение "Возможность того, что будет прохладно, 30 процентов" выражает вероятность наступления прохладной погоды. Но выражение "Утром на 30 процентов прохладно" означает совсем другое, а именно — что прохлада ощущается лишь в некоторой степени.

Ограничением здесь является лишь то, что степени принадлежности объекта взаимно дополняющим друг друга группам должны в сумме составлять единицу. Если воздух на 20 процентов прохладен, то на 80 процентов он должен не быть прохладным. В частном случае объект на 100 процентов может принадлежать одной группе.

Зарождение нечеткой логики связано с именами английского философа и математика Бертрана Рассела и польского математика Яна Лукашевича. А в 1965 году Лотфи А. Заде из Калифорнийского университета опубликовал основополагающую статью "Нечеткие множества". Однако термин "нечеткая логика" стал неизменно употребляться при описании систем, оперирующих нечеткими множествами, лишь с середины 1970-х годов, когда в Лондоне был сконструирован нечеткий регулятор для парового двигателя.

Теперь нечеткие регуляторы применяются даже для управления поездами метрополитена (в Японии); давно уже выпускаются нечеткие микроволновые печи, холодильники, стиральные машины (меняющие режим работы по мере того, как белье становится чище), пылесосы, видеокамеры. Используются нечеткие системы и в медицине, экономике, автомобилестроении, а также ряде других областей [2]. Одна из наиболее сложных нечетких систем — модель дирижабля, сконструированного в Токийском технологическом институте, которая реагирует на команды, подаваемые человеческим голосом.

Большинство существующих нечетких систем оперируют небольшим количеством переменных. Однако для моделирования работы завода или моделирования экономики требуется рассматривать системы с огромным количеством переменных, и тут возникают серьезные трудности. Не обошла нечеткие системы и другая проблема, а именно — "проклятие размерности" (с которой сталкиваются иногда при математическом и компьютерном моделировании), ввиду того что количество так называемых правил нечеткого управления имеет тенденцию увеличиваться по экспоненциальному закону по мере увеличения количества переменных системы [1]. Чем больше в системе употребляется таких правил, тем больше ее "управляемость", но тем менее она точна. Поэтому здесь необходим компромисс.

Тем не менее при любых компромиссах такого рода весьма часто, применяя нечеткую логику, можно лучше отразить неопределенность реального мира, чем используя "черно-белый" подход классической теории множеств. И это должно позволить нечетким системам найти еще более широкое применение.

ЛИТЕРАТУРА

1. *Коско Б., Исака С.* Нечеткая логика: Пер. с англ. // Информатика. № 1/97.

2. *Асаи К., Ватада Д., Иваи С. и др.* Прикладные нечеткие системы: Пер. с японск. М.: Мир, 1993.

* Обычные цифровые компьютеры способны выполнять и нечеткие операции, но довольно медленно. Поэтому специально разрабатываются так называемые нечеткие компьютеры (и соответствующее программное обеспечение).

Гл. редактор
С.Л.Островский
Зам. гл. редактора
Е.Б.Докшицкая
Редакция:
Н.Л.Беленькая,
Н.П.Медведева
Дизайн и компьютерная верстка:
Н.И.Пронская
Корректоры:
Е.Л.Володина,
С.М.Подберезина

выходит четыре раза в месяц
При перепечатке ссылка
на ИНФОРМАТИКУ обязательна,
рукописи не возвращаются

121165, Киевская, 24
тел. 249 4896
Отдел рекламы
тел. 249 9870

Учредитель: ООО "Чистые пруды»
Регистрационный номер 012868
Отпечатано в типографии ОАО ПО "Пресса-1".
125865, ГСП, Москва, ул. Правды, 24.
Тираж 5000 экз.
Заказ №

ИНДЕКС ПОДПИСКИ
для индивидуальных подписчиков 32291
комплекта приложений 32744

Тел. (095)249 3138, 249 3386. Факс (095)249 3184

Internet: inf@1september.ru
Fidonet: 2:5020/69.32
WWW: http://www.1september.ru