№ 15 (352) 16–22 апреля 2002

ПОДПИСКА: (095) 249-47-58

Еженедельная газета Издательского дома «ПЕРВОЕ СЕНТЯБРЯ»

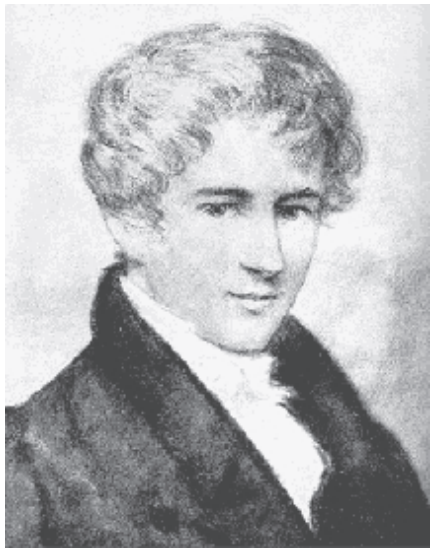
ИНФОРМАТИК

Абель

В 2002 году исполняется 200 лет со дня рождения одного из крупнейших математиков XIX века норвежца Нильса Хенрика Абеля (1802—1829)

"Прожив совсем короткую жизнь, он успел сделать так много, что этого вполне достанет на то, чтобы оставить о нем долгую память; легко себе представить, чего бы он достиг, если бы судьба решила иначе".

Адриен Мари Лежандр, выдающийся французский математик (об Абеле)



"Как поразительно широк круг вопросов, которыми он занимался! ...Все эти проблемы крайне характерны для Абеля, до него их никто не осмеливался даже ставить".

Карл Густав Якоби, выдающийся немецкий математик

Нильс Хенрик Абель сделал открытия, существенно повлиявшие на дальнейшее развитие математики. В различные разделы высшей математики навечно вошли, например, *теоремы Абеля*, *абелевы интегралы*, *уравнения Абеля*, *абелевы группы*, *преобразования Абеля* [1]. При этом все свои работы он выполнил всего за семь лет (до своей ранней смерти в возрасте двадцати шести лет).

Работы Абеля сыграли важную роль в появлении ряда новых математических дисциплин, в частности, теории Галуа, теории алгебраических функций, способствовали признанию теории функций комплексного переменного [1—6].

Первые исследования Абеля относятся к алгебре. В середине 1820-х годов он доказал, что алгебраические уравнения степени выше четвертой в общем случае неразрешимы в радикалах (т.е. их корни не могут быть найдены с помощью четырех действий арифметики и извлечения корней) и указал частные типы уравнений, разрешимых в радикалах. (В науке важно не только открыть что-то новое, но и показать невозможность каких-то вещей [1]. Это освобождает исследователей от напрасной траты времени и сил, позволяет расходовать больше времени на решение других задач.) Позже Абель писал [2, 3]: "Одной из интереснейших проблем алгебры является алгебраическое решение уравнений. Почти все выдающиеся математики исследовали этот вопрос. Без труда были получены общие выражения для корней уравнений первых четырех степеней. Для решения этих уравнений был открыт единый способ, и надеялись, что он применим к уравнениям любой степени; но, несмотря на все усилия Лагранжа и других выдающихся математиков, поставленная цель не была достигнута... Предполагали решать уравнения, не зная, возможно ли это решение. В случае существования решения могли его получить, ничего о нем предварительно не зная; но если, к несчастью, решения не существовало, то его могли бы тщетно искать целую вечность... Вместо того чтобы искать некоторое соотношение, не зная, существует оно или нет, следует поставить вопрос, возможно ли такое соотношение... Этот метод, который, без сомнения, является единственно научным, поскольку лишь он позволяет быть заранее уверенным в достижении поставленной цели, мало применяется в математике только потому, что его применение связано с исключительными трудностями..."

Нильс Хенрик Абель является одним из создателей теории эллиптических функций [3, 4]. Большое значение имеют исследования Абеля, связанные с обоснованием математического анализа; он является автором первой работы, посвященной интегральным уравнениям; его сочинения оставили заметный след в теории интерполирования функций, теории функциональных уравнений, теории чисел.

В первой биографии Абеля, написанной его школьным учителем математики Бернтом Микелем Хольмбое, есть такие строчки [1]: "Абель прожил короткую жизнь и недолго занимался наукой, но, несмотря на это, добился необычайных успехов. Все его статьи свидетельствуют о поразительной способности проникать в глубь вещей и не оставляют сомнений в том, что он был одним из самых одаренных математиков, которых когда-либо знало человечество. Те, кто не в состоянии самостоятельно разобраться в его работах и, таким образом, составить о них собственное мнение, могут убедиться в справедливости наших слов, познакомившись с хвалебными отзывами, которые давали о его статьях самые знаменитые математики".

Литература

1. *Лишевский В.П.* Рассказы об ученых. М.: Наука, 1986.

2. *Виленкин Н.Я., Лишевский В.П.* Нильс Хенрик Абель // Рассказы о математике и математиках. М.: МЦНМО, 2000.

3. Абель // Большая советская энциклопедия. Изд. 2-е. М.: Гл. науч. изд-во "Большая советская энциклопедия", 1949. Т. 1.

4. *Стечкин С.Б.* Абель // Большая советская энциклопедия. Изд. 3-е. М.: Советская энциклопедия, 1970. Т. 1.

5. *Стройк Д.Я.* Краткий очерк истории математики: Пер. с нем. Изд. 4-е. М.: Наука, 1984.

6. *Смирнов О.А., Майорова Т.С., Власова И.Г.* 100 великих имен в математике, физике и географии. М.: Филологическое общество "СЛОВО", 1998.

Читайте в номере

"ЖАРКОЕ ЛЕТО-2002"

Уникальный комплект замечательных материалов к новому учебному году

Д.М. Златопольский. Задачник по электронным таблицам (Excel)

Вопросы, связанные с обработкой информации с помощью электронных таблиц, занимают важное место в школьном курсе информатики. Но специализированного школьного задачника по электронным таблицам нет. Вернее, не было, а теперь есть. Его первая часть будет опубликована в летних номерах нашей газеты.

А.И. Сенокосов. Информатика и информатизация школы. Практические решения

Допустим, вы сделали школьный web-сайт. Протянули провода, настроили локальную сеть, установили программное обеспечение. Все работает. А что же дальше? Как организовать информационное наполнение сайта? Как "встроить" web-сайт в школьную жизнь? Все, кто решал эти вопросы, знают, что они-то и являются настоящими **вопросами**.

А.А. Дуванов. Азы информатики. Книга 3 — "Пишем на компьютере"

"Бумажная" версия третьей книги нового интерактивного курса для малышей "Азы информатики" (первая книга — "Знакомство с компьютером" — была опубликована в № 1, 2, публикация второй — "В мире информации" — продолжается в текущих номерах). Заглавная тема третьей книги (современная обработка текстов) нагружена "анатомией" трех китов информатики: хранение, передача и обработка информации.

А.И. Терентьев. Организация школьного web-сайта

Как сделать школьный web-сайт "с нуля"? Как связать компьютеры в сеть, какое программное обеспечение выбрать и как его установить? Как, наконец, заставить все это "хозяйство" работать? В одном из летних номеров мы познакомим наших читателей с ответами на эти (и не только на эти!) вопросы.

А.Г. Гейн. "Рыба" для учителя информатики

Авторы учебников обычно много говорят о том, что преподавать, и мало про то, как это делать. Свой взгляд на тематическое планирование предлагает один из авторов учебников по информатике.

Л.О. Сергеев. Уроки по теме "Базы данных"

В этом тематическом выпуске мы предложим вниманию читателей цикл уроков по теме "Базы данных". В качестве основного инструмента для изучения этой темы автор предлагает использовать язык SQL. Теоретический материал подкрепляется большим количеством разноуровневых заданий.

Интеллектуальные игры

Что такое спортивное "Что? Где? Когда?" и чем оно отличается от телевизионного? Какие головоломки решают на чемпионатах мира? Во что можно поиграть без компьютера? Это и многое другое, а также вопросы, вопросы, вопросы... в специальном выпуске "Интеллектуальные игры".

Д.М. Златопольский. Внеклассная работа по информатике. Избранные задания

Летом мы предложим подписчикам целый номер с заданиями, которые можно использовать на викторинах, конкурсах и других внеклассных мероприятиях по информатике. В него войдут множество новых заданий, а также лучшие из опубликованных ранее материалов популярной рубрики нашей газеты.

О Г.А. Звенигородском. Редактор-составитель — Н.А. Юнерман

История становления школьной информатики не может быть написана без страниц, посвященных исследованиям и разработкам Г.А. Звенигородского. И сегодня остается актуальным воплощенное в них единство тематических и педагогических сторон работы с учащимися, живой практики и теоретического осмысления материала. 9 августа 2002 г. Г.А. Звенигородскому исполнилось бы 50 лет.

А.А. Дуванов. DHTML-конструирование

"Бумажная" версия электронного учебника продолжает роботландский курс гипертекстового конструирования (ранее были опубликованы "HTML-конструирование" и "JavaScript-конструирование"). Новый учебник посвящен созданию динамических интерактивных приложений. В нем изложены основы CSS (каскадные таблицы стилей) и показаны способы управления содержимым страницы при помощи воздействий на гипертекстовую модель документа.

Уважаемые читатели!

Газета "Информатика" распространяется только по подписке, поэтому не забудьте подписаться на нашу газету.

Современные педагогические технологии и частные методики обучения информатике

Лекции читает И.Н. Фалина

Лекция 12. Алгоритмы обработки текстов

Любой человек, а ученик тем более, в своей деятельности наиболее часто сталкивается с обработкой текстовой информации. Причем эту деятельность можно четко разделить на две группы: связанную с использованием компьютера и, соответственно, не связанную с компьютером.

К наиболее часто используемым "бескомпьютерным" способам обработки информации следует отнести поиск в произвольном тексте нужного слова или предложения, поиск нужного слова в словаре (т.е. поиск в упорядоченном тексте, например, в энциклопедическом словаре или в англо-русском словаре).

При компьютерной обработке текстов наиболее часто используемыми алгоритмами являются

1) удаление лишних пробелов в строке текста;
2) форматирование текста без переноса слов;
3) форматирование текста с переносом слов.

Очень важно показать ученикам вышеперечисленные компьютерные алгоритмы в действии и подчеркнуть, что от того, насколько эффективно они реализованы, зависит скорость обработки текста, например, текстовым редактором или компилятором какого-либо алгоритмического языка (например, любой компилятор опускает "незначащие" пробелы).

Задача поиска нужного слова в словаре

Алгоритм "бескомпьютерного" поиска нужного слова в упорядоченном тексте, например в словаре, не так прост, как может показаться на первый взгляд.

Задача. Дан краткий энциклопедический словарь (содержит 1600 страниц). На каждом развороте на левой странице в левом верхнем углу указаны четыре буквы, с которых начинается первое слово на этой странице, на правой странице в верхнем правом углу указаны четыре буквы, с которых начинается последнее слово на этой странице. Написать алгоритм эффективного поиска нужного слова в словаре, считая, что в качестве закладок можно использовать только свои руки.

Считаем, что исполнитель алгоритма умеет располагать слова в лексикографическом порядке, т.е. в данном случае исполнитель по первым четырем буквам может определить, "раньше" или "позже" находится искомое слово относительно текущего. Под термином "раньше" будем понимать, что искомое слово, если оно присутствует в словаре, должно находиться в словаре перед текущим словом. Термин "позже" означает, соответственно, что искомое слово, если оно присутствует в словаре, должно находиться в нем после текущего слова.

В основе реализации данного алгоритма лежит алгоритм бинарного поиска в упорядоченном множестве. Результатом работы данного алгоритма является выписывание толкования искомого слова или сообщение — "искомого слова в словаре нет".

Решение данной задачи приведено в виде блок-схемы на с. 4.

План публикаций лекций курса "Современные педагогические технологии и частные методики обучения информатике" на "Страницах повышения квалификации".

Номер лекции	Номер газеты
1	37/2001
2	39/2001
3	41/2001
4	43/2001
5	45/2001
6	47/2001
7	5/2002
8	7/2002
9	9/2002
10	11/2002
11	13/2002
12	15/2002

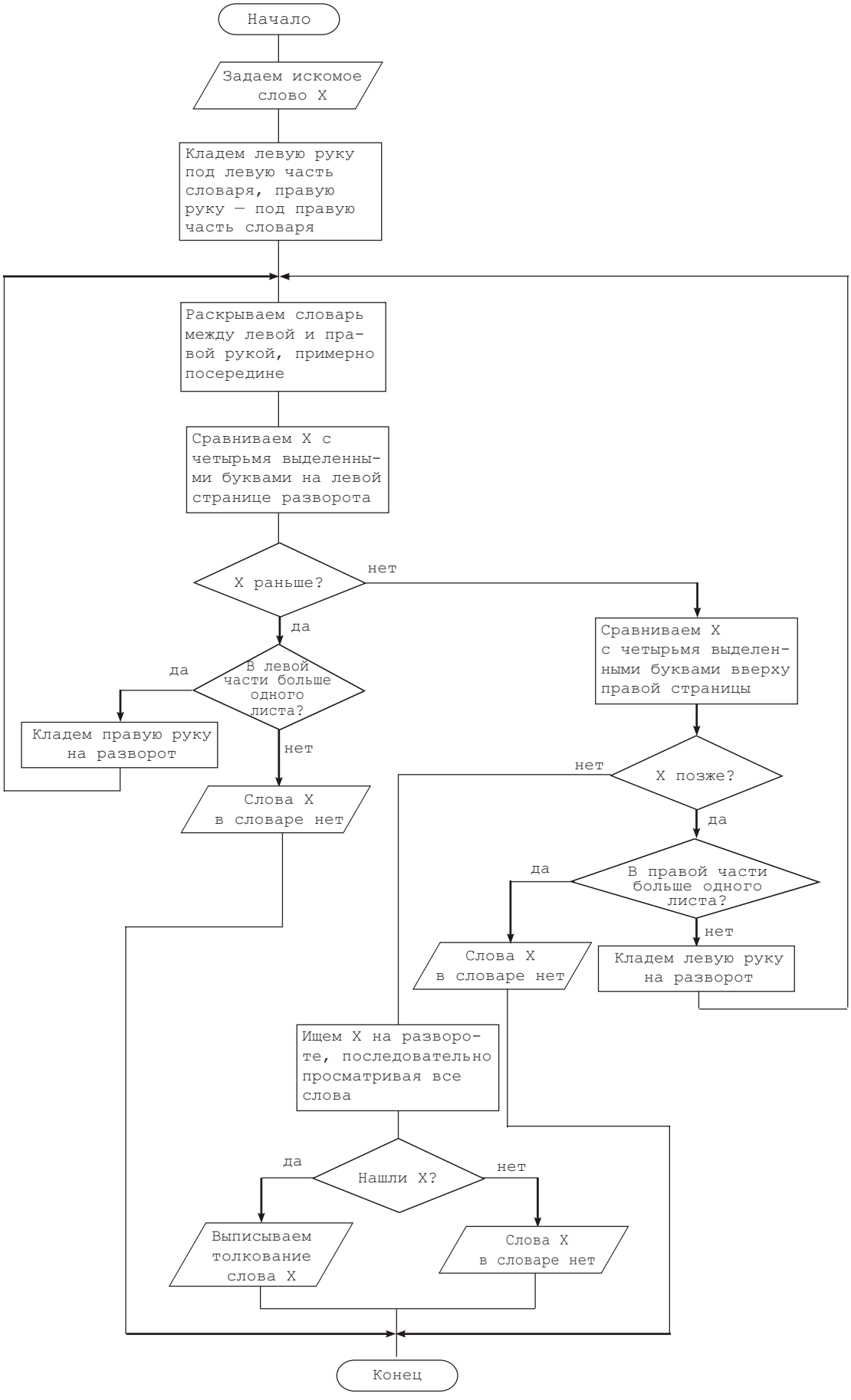
Задача удаления лишних пробелов из строки текста

Реализация алгоритмов компьютерной обработки информации основана на работе со строками и текстовыми файлами. В данной лекции ограничимся рассмотрением алгоритмов без использования файлов, т.е. текст будем представлять последовательностью строк. Более того, будем считать, что каждая строка содержит не более 255 символов.

Строка как структура данных представляет собой последовательность символов, которая заканчивается специальным символом — признаком конца строки. При реализации этой структуры данных в алгоритмических языках используется специальный тип данных **`String`**. Строка типа **`String`** представляет собой одномерный массив символов, который имеет, однако, существенное отличие от данных типа *массив*. Обычный массив символов имеет фиксированную длину (т.е. фиксированное количество элементов), которая определяется при описании. Строка же имеет одновременно две разновидности длины:

- общую длину строки, которая определяет размер памяти, выделяемый под переменную данного типа при описании;
- текущую длину строки (не превосходящую общую длину), которая указывает количество символов, содержащихся в строке в каждый конкретный момент времени.

При такой реализации структуры данных *строка* текущая длина может указываться в нулевом элементе строки.


Начало
Задаем искомое слово X
Кладем левую руку под левую часть словаря, правую руку — под правую часть словаря
Раскрываем словарь между левой и правой рукой, примерно посередине
Сравниваем X с четырьмя выделенными буквами на левой странице разворота
X раньше?
нет
да
В левой части больше одного листа?
да
нет
Кладем правую руку на разворот
Слова X в словаре нет
Сравниваем X с четырьмя выделенными буквами вверху правой страницы
X позже?
нет
да
В правой части больше одного листа?
да
нет
Слова X в словаре нет
Кладем левую руку на разворот
Ищем X на развороте, последовательно просматривая все слова
Нашли X?
да
нет
Выписываем толкование слова X
Слова X в словаре нет
Конец

В этот элемент записывается символ, код которого равен значению текущей длины. При этом нулевой элемент строки сделан невидимым для пользователя, однако использовать его в программах можно, хотя делать это не рекомендуется, особенно начинающим программистам. Очевидно, что при таком способе хранения текущей длины ее максимальное значение может равняться 255 байтам. Такой способ хранения текущей длины строки впервые был введен в языке Турбо Паскаль. Преимуществом такого способа представления строк является чрезвычайно простой доступ к значению текущей длины, что позволяет эффективно выполнять работу со строками. Недостатком является ограничение на длину строки. Подробнее о реализации строк можно прочитать в [1].

Для эффективной работы со строками в алгоритмических языках существуют различные процедуры и функции. Так, например, в языке Турбо Паскаль используются следующие подпрограммы:

`Length(St)` — функция, возвращающая текущую длину строки *St*;

`Insert(St1, St2, Poz)` — процедура, вставляющая подстроку *St*1 в строку *St*2, начиная с позиции *Poz*;

`Pos(St1, St2)` — функция, возвращающая номер позиции первого вхождения подстроки *St*1 в строку *St*2. Если подстрока *St*1 не входит в строку *St*2, то функция возвращает 0;

`Delete(St, Poz, N)` — процедура, удаляющая *N* символов из строки *St*, начиная с позиции *Poz*;

`Copy(St, Poz, N)` — функция, возвращающая последовательность символов строки *St* длиной *N*, начиная с позиции *Poz*. Исходная строка *St* не модифицируется.

Задача. Строка символов представляет собой некоторый текст, слова в котором разделены одним или более пробелами. Преобразовать эту строку так, чтобы все слова разделялись ровно одним пробелом, а ведущие и хвостовые пробелы отсутствовали.

1. Приведем решение этой задачи в виде текстового алгоритма, записанного по шагам. Будем считать, что исполнитель алгоритма умеет выполнять следующие действия над строками:

- находить заданную подстроку в исходной строке;
- делать контекстную замену.

① Ищем в строке два подряд идущих пробела.

② Если поиск удачен, то заменяем два пробела на один и переходим на п. ①.

③ Если строка начинается с пробела, то удаляем его.

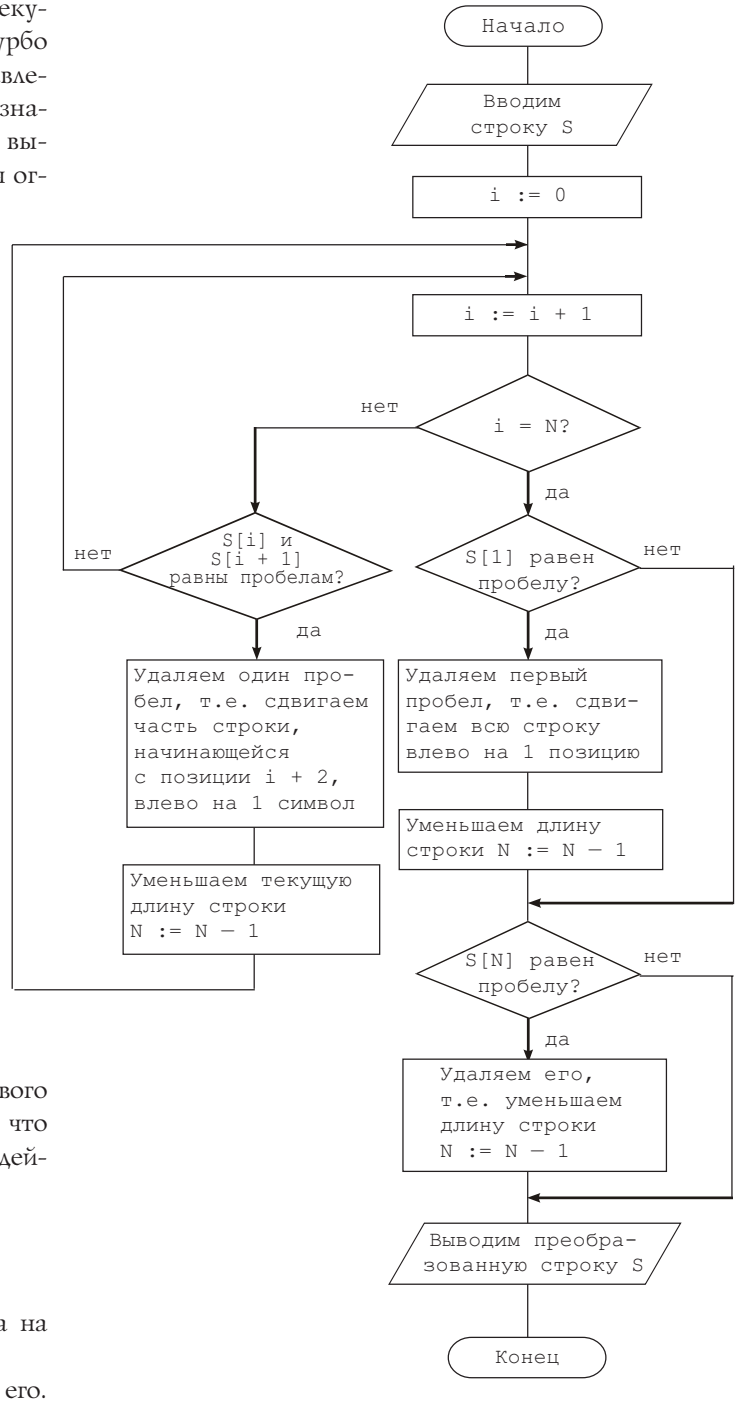
④ Если строка заканчивается пробелом, то удаляем его. Конец алгоритма.

2. Приведем описание этого же алгоритма в виде блок-схемы, но будем считать, что исполнитель не умеет выполнять операции над всем объектом *строка* в целом. То есть исполнитель работает со строкой, как с массивом символов. В этом случае все операции над строкой как единым объектом (они вынесены в алгоритмических языках в процедуры или функции, например, удаление подстроки из строки или поиск подстроки в строке) будем выполнять в данном алгоритме явно.

Будем использовать следующие обозначения:

i — номер анализируемого символа строки;

N — текущая длина строки.



Задача выравнивания текста по ширине строки

Операция форматирования текста — одна из наиболее распространенных при работе с текстовой информацией. Существует несколько операций форматирования, мы рассмотрим операцию выравнивания по ширине строки.

Задача. Дан текст на русском языке, представляющий собой последовательность строк. Все слова в тексте написаны без переносов. Знаки препинания (“.”, “,”, “—” и т.д.) всегда примыкают вплотную к предшествующему слову. В каждой строке имеется по крайней мере два слова, разделенных как минимум одним пробелом. Пробелы в начале каждой строки отсутствуют. За счет изменения длин групп пробелов внутри строк отформатировать текст по ширине, что означает следующее:

1) все строки имеют одинаковую длину, равную самой длинной строке текста;

2) в конце каждой строки пробелы отсутствуют;

3) количество пробелов между словами в одной строке различается не более чем на единицу.

Решение.

① Вначале определим максимальную длину строки в тексте.

② Далее отдельно для каждой строки текста решаем задачу выравнивания по ширине. Обозначим через n количество групп пробелов между словами в строке, через *blanks* обозначим количество пробелов, которое должно быть в строке после выравнивания по ширине. Тогда, чтобы количество пробелов в группах различалось не больше чем на единицу, их число должно быть равно ($blanks$ div n) в (n — $blanks$ mod n) группах пробелов и ($blanks$ div $n + 1$) в ($blanks$ mod n) группах пробелов. Причем хотя бы в одной группе пробелов их будет ровно ($blanks$ div n). Разместим именно столько пробелов в первой группе (между первым и вторым словом).

③ Далее сведем задачу к задаче меньшей размерности. А именно, уменьшим на ($blanks$ div n) количество распределяемых пробелов и на единицу — количество групп пробелов.

④ Тогда на последнем шаге n будет равно единице, и все оставшиеся пробелы будут напечатаны.

Докажем, что предложенный алгоритм верен.

1. Если $blanks \bmod n = 0$, т.е.

$$blanks \ \text{ div } \ n = \frac{blanks}{n}, \text{ то}$$

$$\frac{(blanks - \frac{blanks}{n})}{n-1} = \frac{blanks\,(n-1)}{n\,(n-1)} = \frac{blanks}{n}. \qquad (1)$$

То есть в этом случае наш алгоритм справедлив.

2. В противном случае обозначим

$$k = blanks \text{ div } n;\ m = blanks \bmod n.$$

Тогда $blanks = k \cdot n + m$, и количество пробелов в следующей группе определяется формулой

$$\frac{k \cdot n + m - k}{n-1} = \frac{k(n-1)+m}{n-1} = k + \frac{m}{n-1}.$$

Так как $0 \le m \le n-1$, то количество пробелов в этой группе увеличится на единицу только в случае, когда $m = n - 1$. Но при выполнении этого равенства ($k \cdot n + m - k$) будет делиться на $n - 1$. Следовательно, согласно равенству ① во всех остальных группах количество пробелов уже не изменится и будет равно ($k + 1$).

Приведем алгоритм решения задачи в текстовой форме, записанный по шагам.

Будем использовать следующие обозначения:

i — номер преобразуемой строки текста;

$S[i]$ — обрабатываемая строка текста;

n — количество групп пробелов внутри преобразуемой строки;

blanks — количество пробелов, которые надо расставить в i-й строке;

m — количество слов в i-й строке;

k — номер текущего слова в i-й строке;

SS — временная (вспомогательная) строка.

① Определим максимальную длину строки в тексте (*max*).

② $i = 0$.

③ $i = i + 1$.

④ Если обработаны все строки, то конец алгоритма.

⑤ Определим количество групп пробелов для i-й строки. Для этого надо подсчитать количество слов в строке (m). Одновременно подсчитаем исходное число пробелов в строке (*blanks*).

⑥ Определим количество пробелов в результирующей строке: $blanks = blanks +$ (*max* — длина строки).

⑦ $k = 0$.

⑧ $k = k + 1$.

⑨ Если $k = m$, то k-е слово записываем в строку *SS*, затем содержимое временной строки *SS* записываем в текст на место строки $S[i]$, переход на п. ③.

⑩ Выделяем k-е слово, записываем его во временную строку *SS*. Затем во временную строку *SS* вставляем ($blanks$ div n) пробелов.

⑪ Уменьшаем количество пробелов, которые надо еще вставить в строку $S[i]$:

$$blanks = blanks - (blanks \text{ div } n).$$

⑫ Уменьшаем количество групп пробелов:

$n = n - 1$, переход на п. ⑧.

Задача выравнивания текста по ширине строки с переносом слов

Данная задача существенно сложнее предыдущей. Алгоритм решения этой задачи целесообразно разбирать только на факультативных занятиях. Приведем условие последней задачи, которая рассматривается в [2].

Задача. Дан некоторый текст на русском языке. Требуется выровнять этот текст по ширине, используя разделение слов по правилам переноса. Дополнительные пробелы можно вставлять между словами в случае, если деление слова на перенос “не закрывает” всю строку. В исходном тексте все слова приведены без переноса. Знаки препинания (“.”, “,”, “—” и т.д.) должны прилегать вплотную к предшествующему слову. Ко-

личество знаков в строке при выравнивании по ширине запросить с экрана.

Деление слова на две части для переноса требуется выполнять согласно следующим правилам:

1. Две идущие подряд гласные можно разделить, если первой из них предшествует согласная, а за второй идет хотя бы одна буква (буква "й" всегда рассматривается вместе с предшествующей как единое целое).

2. Две идущие подряд согласные буквы можно разделить, если первой из них предшествует гласная, во второй части слова имеется хотя бы одна гласная буква. Буквы "ъ", "ь" при этом рассматриваются вместе с предшествующей согласной как единое целое.

3. Если две идущие подряд буквы не соответствуют двум первым пунктам, то следует попытаться разбить слово так, чтобы каждая часть содержала более чем одну букву, первая часть заканчивалась на гласную, а вторая содержала хотя бы одну гласную.

На этой достаточно сложной задаче я заканчиваю тему, объявленную в лекции 9. Формулировки и решения задач, подобных этой, можно найти в [2].

Заключение

Школьная информатика ассоциируется у меня с современным бурлящим городом, в котором бок о бок живут и учителя, и наши ученики, и их родители, и многие специалисты, связанные с информатикой в целом. Но мы, учителя, ходим по этому городу пешком, часто по знакомым любимым улицам, неторопливо читая знакомые вывески, заходя в привычные магазины и кафе. А наши ученики проносятся по этому городу кто бегом, кто на велосипеде, а кто-то и на скоростных машинах, выхватывая глазами яркие надписи с рекламных щитов, а вечерами они бродят по таким закоулкам, о которых мы можем только догадываться. И только на уроках информатики, как на перекрестках шумных улиц, мы встречаемся со своими учениками в этом большом городе.

О чем же нам рассказывать им за те несколько минут, пока "горит красный свет" и бегущие по городу ученики с нетерпением ждут "зеленого света"?

Скорее всего мы должны научить их ориентироваться в городе "Информатика", понимать, что скрывается за "яркими рекламными надписями", оценивать качество "рекламируемого товара". Но времени на перекрестке-уроке очень мало, а ученики информацию получают не только от нас, учителей, ведь информатизация проникает во все уголки нашей жизни: где-то новое слово услышат, что-то прочитали, что-то друг показал. И остается нам только одно — продуманная методика построения и изложения курса, только так можно удержаться на плаву в этом все убыстряющемся ритме города по имени "Информатика".

Литература

1. *Марченко А.И., Марченко Л.А.* Программирование в среде Turbo Pascal 7.0. М.: Бином Универсал. Киев: ЮНИОР, 1997.

2. *Андреева Е.В., Фалина И.Н.* Турбо Паскаль в школе. Сборник задач и контрольных работ по информатике. М.: Изд-во Н.Бочкаревой, 1998.

Примерные ответы на примерные билеты

Е.А. Еремин, А.П. Шестаков,
г. Пермь

Продолжение. См. № 9, 10, 11, 13, 14/2002

Билет № 11

1. Электронные таблицы. Назначение и основные возможности.

2. Разработка алгоритма или программы, содержащей команды ветвления.

* * *

1. Электронные таблицы. Назначение и основные возможности.

Современные технологии обработки информации часто приводят к тому, что возникает необходимость представления данных в виде таблиц. В языках программирования для такого представления служат двухмерные *массивы*. Для табличных расчетов характерны относительно простые формулы, по которым производятся вычисления, и большие объемы исходных данных. Такого рода расчеты принято относить к разряду рутинных работ, для их выполнения следует использовать компьютер. Для этих целей созданы *электронные таблицы* (*табличные процессоры*) — прикладное программное обеспечение общего назначения, предназначенное для обработки различных данных, представимых в табличной форме.

Электронная таблица (*ЭТ*) позволяет хранить в табличной форме большое количество *исходных данных*, *результатов*, а также *связей* (алгебраических или логических соотношений) *между ними*. При изменении исходных данных все результаты автоматически пересчитываются и заносятся в таблицу. Электронные таблицы не только автоматизируют расчеты, но и являются эффективным средством моделирования различных вариантов и ситуаций. Меняя значения исходных данных, можно следить за изменением получаемых результатов и из множества вариантов решения задачи выбрать наиболее приемлемый.

Рабочим полем табличного процессора является экран дисплея, на котором электронная таблица представляется в виде прямоугольника, разделенного на строки и столбцы. Строки нумеруются сверху вниз. Столбцы обозначаются слева направо. На экране виден не весь документ, а только часть его. Документ в полном объеме хранится в оперативной памяти, а экран можно считать окном, через которое пользователь имеет возможность просматривать таблицу. Для работы с таблицей используется табличный курсор — выделенный прямоугольник, который можно поместить в ту или иную клетку. Минимальным элементом электронной таблицы, над которым можно выполнять те или иные операции, является такая клетка, которую чаще называют *ячейкой*. Каждая ячейка имеет уникальное *имя* (идентификатор), которое составляется из номеров столбца и строки, на пересечении которых располагается ячейка. Нумерация столбцов обычно осуществляется с помощью латинских букв (поскольку их всего 26, а столбцов значительно больше, то далее идет такая нумерация — `AA`, `AB`, ..., `AZ`, `BA`, `BB`, `BC`...), а строк — с помощью десятичных чисел, начиная с единицы. Таким образом, возможны имена (или *адреса*) ячеек `B2`, `C265`, `AD11` и т.д.

Следующий объект в таблице — *диапазон ячеек*. Его можно выделить из подряд идущих ячеек в строке, столбце или прямоугольнике. При задании диапазона указывают его начальную и конечную ячейки, в прямоугольном диапазоне — ячейки левого верхнего и правого нижнего углов. Наибольший диапазон представляет вся таблица, наименьший — ячейка. Примеры диапазонов — `A1:A100`; `B12:AZ12`; `B2:K40`.

Если диапазон содержит числовые величины, то они могут быть просуммированы, вычислено среднее значение, найдено минимальное или максимальное значение и т.д.

Иногда электронная таблица может быть составной частью *листа*, листы, в свою очередь, объединяются в *книгу* (такая организация используется в Microsoft Excel).

Ячейки в электронных таблицах могут содержать *числа* (целые и действительные), *символьные и строковые величины*, *логические величины*, *формулы* (алгебраические, логические, содержащие условие).

В формулах при обращении к ячейкам используются два способа адресации — *абсолютная* и *относительная*. При использовании относительной адресации копирование, перемещение формулы, вставка или удаление строки (столбца) с изменением местоположения формулы приводят к перестраиванию формулы относительно ее нового местонахождения. В силу этого сохраняется правильность расчетов при любых указанных выше действиями над ячейками с формулами. В некоторых же случаях необходимо, чтобы при изменении местоположения формулы адрес ячейки (или ячеек), используемой в формуле, не изменялся. В таких случаях используется абсолютная адресация. В приведенных выше примерах адресов ячеек и диапазонов ячеек адресация является относительной. Примеры абсолютной адресации (в Microsoft Excel): `$A$10`; `$B$5:$D$12`; `$M10`; `K$12` (в предпоследнем примере фиксирован только столбец, а строка может изменяться, в последнем — фиксирована строка, столбец может изменяться).

Управление работой электронной таблицы осуществляется посредством команд.

Можно выделить следующие режимы работы табличного процессора:

- формирование электронной таблицы;
- управление вычислениями;
- режим отображения формул;

• графический режим;
• работа электронной таблицы как базы данных.

При работе с табличными процессорами создаются документы, которые можно просматривать, изменять, записывать на носители внешней памяти для хранения, распечатывать на принтере. *Режим формирования электронных таблиц* предполагает заполнение и редактирование документа. При этом используются команды, изменяющие содержимое клеток (очистить, редактировать, копировать), и команды, изменяющие структуру таблицы (удалить, вставить, переместить).

Режим управления вычислениями. Все вычисления начинаются с ячейки, расположенной на пересечении первой строки и первого столбца электронной таблицы. Вычисления проводятся в естественном порядке, т.е. если в очередной ячейке находится формула, включающая адрес еще не вычисленной ячейки, то вычисления по этой формуле откладываются до тех пор, пока значение в ячейке, от которого зависит формула, не будет определено. При каждом вводе нового значения в ячейку документ пересчитывается заново — выполняется автоматический пересчет. В большинстве табличных процессоров существует возможность установки ручного пересчета, т.е. таблица пересчитывается заново только при подаче специальной команды.

Режим отображения формул задает индикацию содержимого клеток на экране. Обычно этот режим выключен, и на экране отображаются значения, вычисленные на основании содержимого клеток.

Графический режим дает возможность отображать числовую информацию в графическом виде: диаграммы и графики. Это позволяет считать электронные таблицы полезным инструментом автоматизации инженерной, административной и научной деятельности.

В современных табличных процессорах, например в Microsoft Excel, в качестве *базы данных* можно использовать *список* (набор строк таблицы, содержащий связанные данные). При выполнении обычных операций с данными, например, при поиске, сортировке или обработке данных, списки автоматически распознаются как базы данных. Перечисленные ниже элементы списков учитываются при организации данных:

• столбцы списков становятся полями базы данных;
• заголовки столбцов становятся именами полей базы данных;
• каждая строка списка преобразуется в запись данных.

Рассмотрим примеры обработки данных с использованием табличного процессора.

Пример 1

(Информатика. Задачник-практикум в 2 т. / Под ред. И.Г. Семакина, Е.К. Хеннера: Т. 2. М.: Лаборатория базовых знаний, 1999, 280 с.)

В пещере у реки поселился огнедышащий дракон. Всех, кто пытался его прогнать, он прогонял сам, полыхая на них огнем. Количество полыханий зависело от того, на кого надо полыхать. На царевича дракон полыхал 5 раз, на королевича — 4 раза, на простого рыцаря — 3.

За первые сто лет дракона пытались прогнать 2 царевича, 3 королевича и 5 простых рыцарей. За второе столетие на него покушались 3 царевича, 2 королевича и 7 простых рыцарей. За третий век дракона беспокоили 7 царевичей, 5 королевичей и 6 простых рыцарей. За следующее столетие дракону пришлось иметь дело с 3 царевичами, 6 королевичами и 10 простыми рыцарями. После чего дракона в конце концов оставили в покое и объявили гору, на которой он жил, заповедником для охраны редких видов животных.

Построить электронную таблицу, из которой будет видно: сколько человек пытались прогнать дракона за каждое из столетий в отдельности и за все 4 века вместе; сколько среди них было царевичей, сколько королевичей и сколько простых рыцарей; сколько раз дракону пришлось полыхать на них огнем в течение каждого века и за все 4 столетия вместе; сколько полыханий досталось царевичам, сколько королевичам и сколько простым рыцарям.

Решение

Прежде всего необходимо продумать структуру таблицы и разместить в ней имеющуюся информацию. В приведенном ниже решении информация о царевичах, королевичах и рыцарях занесена в строки, а столбцы содержат сведения о сражениях по векам. Нижняя строка и последние два столбца содержат итоговую информацию согласно условию задачи. Информация о полыханиях, приходящихся на одного царевича, королевича, рыцаря, вынесена отдельно. Это связано с тем, что при изменении этих данных достаточно будет изменить их в указанных ячейках, не изменяя при этом всех формул.

Ниже приведены фрагменты таблицы с решением в режиме отображения формул и с результатами расчетов.

	A	B	C	D	E	F	G	H	I	J	K
1	**История борьбы с огнедышащим драконом в Тридевятом царстве**										
2		***век 1***		***век 2***		***век 3***		***век 4***			
3		Кол-во	Всего полыханий	Кол-во	Всего полыханий	Кол-во	Всего полыханий	Кол-во	Всего полыханий	Всего человек	Всего полыханий
4	Царевич	2	=$F$11*B4	3	=$F$11*D4	7	=$F$11*F4	3	=$F$11*H4	=B4+D4+F4+H4	=C4+E4+G4+I4
5	Королевич	3	=$F$12*B5	2	=$F$12*D5	5	=$F$12*F5	6	=$F$12*H5	=B5+D5+F5+H5	=C5+E5+G5+I5
6	Рыцарь	5	=$F$13*B6	7	=$F$13*D6	6	=$F$13*F6	10	=$F$13*H6	=B6+D6+F6+H6	=C6+E6+G6+I6
7	Всего	=СУММ(B4:B6)	=СУММ(C4:C6)	=СУММ(D4:D6)	=СУММ(E4:E6)	=СУММ(F4:F6)	=СУММ(G4:G6)	=СУММ(H4:H6)	=СУММ(I4:I6)	=СУММ(J4:J6)	=СУММ(K4:K6)
8											
9											
10											
11					Царевич	5					
12					Королевич	4					
13					Рыцарь	3					

	A	B	C	D	E	F	G	H	I	J	K
1	**История борьбы с огнедышащим драконом в Тридевятом царстве**										
2		***век 1***		***век 2***		***век 3***		***век 4***			
3		Кол-во	Всего полыханий	Кол-во	Всего полыханий	Кол-во	Всего полыханий	Кол-во	Всего полыханий	Всего человек	Всего полыханий
4	Царевич	2	10	3	15	7	35	3	15	15	75
5	Королевич	3	12	2	8	5	20	6	24	16	64
6	Рыцарь	5	15	7	21	6	18	10	30	28	84
7	Всего	10	37	12	44	18	73	19	69	59	223
8											
9											
10											
11					Царевич	5					
12					Королевич	4					
13					Рыцарь	3					

Пример 2

Составить форму для решения равнобедренного треугольника по основанию и противолежащему ему углу (вычисления его боковых сторон, периметра, оставшихся углов, площади, высот).

Решение

Разработаем форму, которая обрабатывает только корректные исходные данные, т.е. треугольник с такими данными должен существовать, заданные величины не могут быть отрицательными и т.д. В таблице достаточно зафиксировать верные расчетные формулы, и эта форма будет пригодна для любых вычислений с указанными исходными данными.

Пусть основание равно c, заданный угол — C.

Тогда углы

$$A = B = (180 - C)/2;$$

боковые стороны (по теореме синусов)

$$a = b = (c \sin A)/\sin C;$$

периметр $P = a + b + c$;

площадь $S = 1/2\ ab \sin C$;

высоты $h_a = 2S/a$; $h_b = 2S/b$; $h_c = 2S/c$.

Ниже приведены фрагменты таблицы с решением в режиме отображения формул и с результатами расчетов при $c = 10$, $C = 60°$.

	A	B
1	Угол C	60
2	c	10
3		
4	Угол A	=(180-B1)/2
5	Угол B	=B4
6	a	=B2*SIN(B4/180*ПИ())/SIN(B1/180*ПИ())
7	b	=B6
8	P	=B2+B6+B7
9	S	=0,5*B6*B7*SIN(B1/180*ПИ())
10	ha	=2*B9/B6
11	hb	=2*B9/B7
12	hc	=2*B9/B2

	A	B
1	Угол C	60
2	c	10
3		
4	Угол A	60
5	Угол B	60
6	a	10
7	b	10
8	P	30
9	S	43,30127019
10	ha	8,660254038
11	hb	8,660254038
12	hc	8,660254038

Литература

1. *Гейн А.Г., Сенокосов А.И., Шолохович В.Ф.* Информатика: 7—9-е классы. Учебник для общеобразовательных учебных заведений. М.: Дрофа, 1998, 240 с. (§ 4. Организация вычислений с помощью ЭВМ, с. 23—31.)

2. *Каймин В.А., Щеголев А.Г., Ерохина Е.А., Федюшин Д.П.* Основы информатики и вычислительной техники: Пробный учебник для 10—11-х классов средней школы. М.: Просвещение, 1989.

3. *Кушниренко А.Г., Лебедев Г.В., Сворень Р.А.* Основы информатики и вычислительной техники: Учебник для средних учебных заведений. М.: Просвещение, 1993.

4. *Семакин И., Залогова Л., Русаков С., Шестакова Л.* Информатика: учебник по базовому курсу. М.: Лаборатория базовых знаний, 1998. (Глава 8. Табличные вычисления на компьютере, с. 163—176.)

5. *Угринович Н.* Информатика и информационные технологии. Учебное пособие для общеобразовательных учреждений. М.: БИНОМ, 2001, 464 с. (Глава 12. Технология обработки числовых данных в электронных таблицах, с. 311—328.)

6. Информатика. 7—8-е классы / Под ред. Н.В. Макаровой. СПб.: ПитерКом, 1999, 368 с. (Раздел 6. Прикладная среда — табличный процессор, с. 238—303.)

7. *Шафрин Ю.А.* Информационные технологии. М.: Лаборатория базовых знаний, 1998, 704 с.

* * *

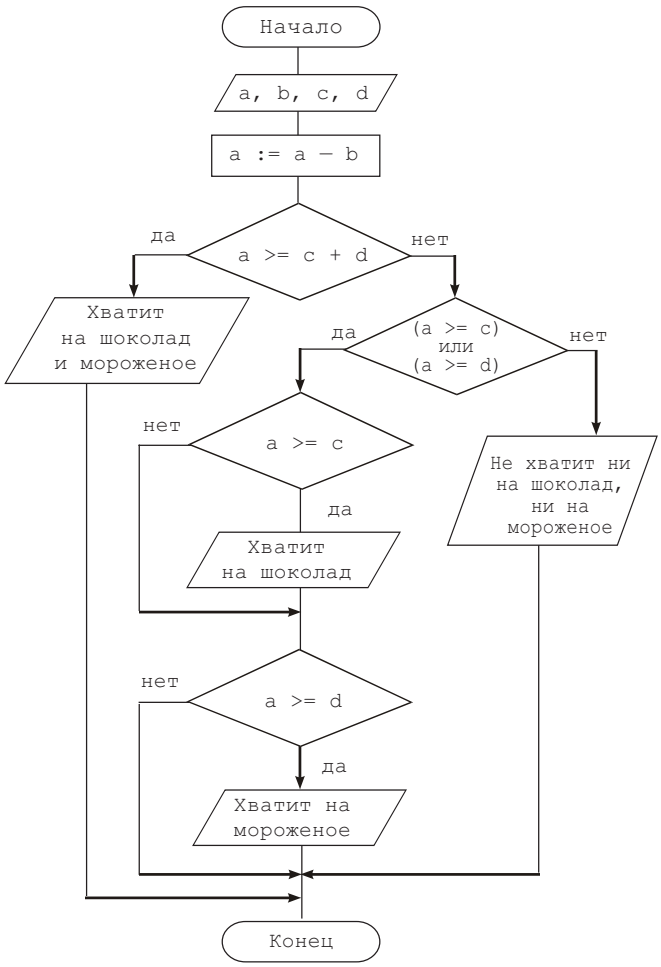
2. Разработка алгоритма или программы, содержащей команды ветвления.

Решить задачу

Мама выдала Васе a руб. На часть этой суммы он должен купить хлеб, стоимость которого b руб. Сможет ли Вася на оставшиеся деньги приобрести шоколад и мороженое, или хотя бы шоколад или мороженое (если можно приобрести шоколад или мороженое, сообщить об обеих возможностях), стоимость которых соответственно c и d руб.?

При решении задачи можно использовать запись алгоритма графически (блок-схема), школьный алгоритмический язык, один из языков программирования или табличный процессор (в зависимости от того, что изучалось в курсе информатики).

Решение



Язык Паскаль

```
program example1;
var a, b, c, d : real;
begin
  write('Введите сумму денег, стоимость хлеба, шоколада и мороженого');
  readln(a, b, c, d);
  a := a – b;
  if a >= c + d then writeln('Хватит на шоколад и мороженое')
  else if (a >= c) or (a >= d) then begin
                                     if (a >= c) then writeln('Хватит только на шоколад');
                                     if (a >= d) then writeln('Хватит только на мороженое')
                                    end
  else writeln('Не хватит ни на шоколад, ни на мороженое')
end.
```

Язык Бейсик

```
input "Введите сумму денег, стоимость хлеба, шоколада и мороженого: "; a, b, c, d
a = a – b
if a >= c + d then print "Хватит на шоколад и мороженое"
              else
                if (a >= c) or (a >= d) then
                 if (a >= c) then print "Хватит только на шоколад "
                 if (a >= d) then print "Хватит только на мороженое"
                             else print "Не хватит ни на шоколад, ни на мороженое"
                end if

end if
end
```

Язык Си

```
#include <stdio.h>
double a, b, c, d;
void main()
{printf("Введите сумму денег, стоимость хлеба, шоколада и мороженого: ");
 scanf("%lf%lf%lf%lf", &a, &b, &c, &d);
 a –= b;
 if (a >= c + d) printf("Хватит на шоколад и мороженое \n");
 else if (a >= c || a >= d) {
        if (a >= c) printf("Хватит только на шоколад \n");
        if (a >= d) printf("Хватит только на мороженое \n");
      }
      else printf("Не хватит ни на шоколад, ни на мороженое");
}
```

Школьный алгоритмический язык

```
алг вычисление1(вещ a, b, c, d, лит y, z)
  арг a, b, c, d
нач
|   a := a – b; y := ""; z := "";
|   если a >= c + d
|   | то y := "Хватит на шоколад и мороженое"
|   | иначе если (a >= c) или (a >= d)
|   |        | то если (a >= c)
|   |        |    | то y := "Хватит только на шоколад"
|   |        |    все
|   |        |    если (a >= c)
|   |        |    | то z := "Хватит только на мороженое "
|   |        |    все
|   |        | иначе y := "Не хватит ни на шоколад, ни на мороженое"
|   |        все
|   все
кон
```

В табличном процессоре MS Excel (приведен фрагмент таблицы с решением задачи в режиме отображения формул)

	A	B	C	D	E	F
1	a=a-b	=F1-F2			a	10
2	a>=c+d	=B1>=F3+F4			b	3
3	a>=c	=B1>=F3			c	4
4	a>=d	=B1>=F4			d	4
5	y	=ЕСЛИ(B2;"Можно купить и шоколад, и мороженое";ЕСЛИ(B3;"Можно купить шоколад";""))				
6	z	=ЕСЛИ(B2;"Можно купить и шоколад, и мороженое";ЕСЛИ(B4;"Можно купить мороженое";""))				

Билет № 12

1. Система управления базами данных. Назначение и основные возможности.

2. Решение расчетной задачи с использованием математических функций в среде программирования.

* * *

1. Система управления базами данных. Назначение и основные возможности.

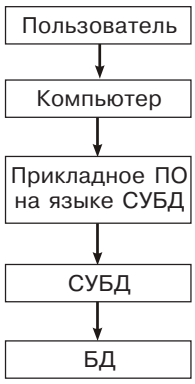
Базы данных используются буквально во всех сферах человеческой деятельности — в науке, производстве, торговле, медицине, криминалистике, искусстве и т.п.

Что такое база данных (БД)? В широком смысле слова можно сказать, что **база данных** — это совокупность систематизированных сведений об объектах окружающего нас мира по какой-либо области знаний, своеобразная информационная модель этой области. Например: БД по сплавам металлов, БД о работниках предприятия, БД в системе продажи билетов, БД документов в той или иной области, БД по видеофильмам — и многие другие.

Обратите внимание на то, что в определении отсутствует упоминание о компьютере. И это не ошибка — хранение систематизированных данных в виде различных картотек использовалось *до* появления самых первых вычислительных машин. Вспомните, например, каталог в библиотеке — традиционные небольшие ящички, заполненные карточками со сведениями о книгах и месте их хранения. Во многих библиотеках они сохранились до сих пор, хотя постепенно их содержимое переносится в обширную память компьютеров. Переход к компьютерному хранению информации дает много преимуществ: практически неограниченный объем данных в сочетании с оперативным доступом к ним, возможность логического контроля вводимой информации, контроль целостности и непротиворечивости информации в базе, регулирование уровня доступа к данным для различных категорий пользователей и, наконец, самое главное — замена механического извлечения отдельных сведений мощными методами обработки запросов человека и автоматическое составление произвольных справок и отчетов. С появлением компьютерных сетей отпала необходимость хранения данных в одной машине и даже в одной стране, возникли так называемые **распределенные** БД. "Вершиной" объединения компьютерных данных может служить Всемирная информационная сеть Интернет.

Тем не менее отдельно взятой БД для реализации всех перечисленных возможностей недостаточно. Помимо собственно данных, требуется специальное программное обеспечение, которое с ними работает. Такое универсальное ПО принято называть **системами управления базами данных,** или сокращенно СУБД. Именно наличие СУБД превращает огромный объем хранимых в компьютерной памяти сведений в мощную справочную систему, способную производить поиск и отбор необходимой нам информации.

Роль и место СУБД в процессах компьютерной обработки данных иллюстрирует следующая схема.



Собственно СУБД, управляющая доступом к данным в базе, является универсальным программным обеспечением. Поэтому для адаптации к конкретной области и учета ее конкретных особенностей необходима возможность "подстройки" программного обеспечения. С этой целью большинство СУБД обладают встроенными средствами подобного рода, т.е. фактически собственным языком программирования. Заметим, что в более ранних разновидностях СУБД, например dBASE и родственных ей (FoxPro, Clipper), это было заметно наиболее отчетливо. В современном программном обеспечении, таком, как MS Access, Paradox, Clarion, создание различных форм и отчетов во многом автоматизировано, но тем не менее встроенные языковые средства по-прежнему сохраняются.

Весь изображенный на рисунке комплекс программных и аппаратных средств, предназначенных для хранения, изменения и обработки информации, а также обеспечивающих взаимодействие с пользователем, в литературе принято называть **информационной системой**.

Сформулируем теперь более четко те функции, которые выполняет современная система управления базами данных.

• **Ввод информации в БД и обеспечение его логического контроля.** Под логическим контролем здесь понимается проверка на допустимость вводимых данных: нельзя, например, вводить дату рождения 31 июня 1057 года.

• **Исправление информации** (также с контролем правильности ввода).

• **Удаление устаревшей информации.**

• **Контроль целостности и непротиворечивости данных.** Под термином "целостность" обычно понимают то, что данные, хранящиеся в разных частях базы данных, не противоречат друг другу, например, дата поступления в школу явно не может быть позже даты ее окончания.

• **Защита данных от разрушения.** Помимо контроля за целостностью, который только что обсуждался, СУБД должна иметь средства защиты данных от выключения электропитания, сбоев оборудования и других аварийных ситуаций, а также возможности последующего восстановления информации. Особую актуальность данный пункт приобретает в сложных многопользовательских системах.

• **Поиск информации с необходимыми свойствами.** Одна из наиболее важных в практическом отношении задач, ради которой ставятся все остальные.

• **Автоматическое упорядочивание информации в соответствии с требованиями человека.** Сюда относится сортировка данных, распределение их между несколькими базами и другие подобные процедуры.

• **Обеспечение коллективного доступа к данным.** В современных информационных системах возможен параллельный доступ к одним и тем же данным нескольких пользователей, поэтому СУБД должны поддерживать такой режим.

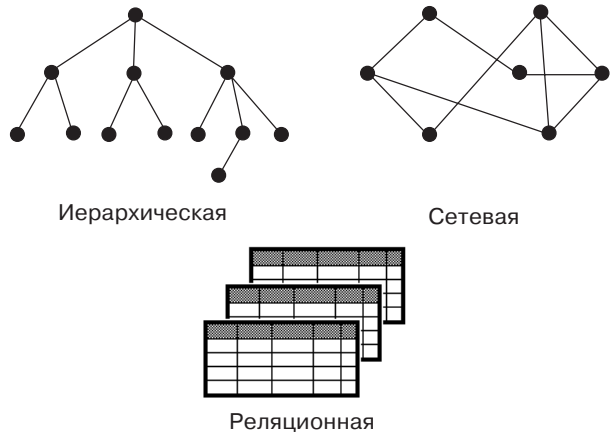
• **Защита от несанкционированного доступа.** Не только ввод новой информации, но даже ее просмотр должны быть разрешены только тем пользователям, у которых есть на это права. Причем речь идет не только о сохранении военной или коммерческой тайны. Например, казалось бы, такой безобидный факт, извлеченный из БД, как неоднократная покупка человеком определенного лекарства в аптеке, в принципе может привести к тому, что при приеме на работу будет взят другой претендент.

• **Удобный и интуитивно понятный пользователю интерфейс.**

Наконец, говоря о СУБД, нельзя обойти стороной еще один общий вопрос, связанный с организацией в них данных. Начнем с того, что, помимо собственно данных, в любой базе имеется информация о ее строении, которую чаще всего называют **структурой**. В простейшем случае структура просто указывает тип информации и объем требуемой для нее памяти. Сведения о структуре позволяют СУБД легко рассчитывать местоположение требуемых данных на внешнем носителе и, следовательно, быстро получить к ним доступ.

Связанные между собой данные, например сведения об одном человеке или объекте, объединяются в БД в единую конструкцию, которая называется **запись**. При этом части, образующие запись, принято называть **полями,** или реже — **элементами данных**. Примерами полей могут служить фамилия, номер паспорта, семейное положение, наличие или отсутствие детей и т.д.

Характер связи между записями в БД определяет три основных типа организации баз данных: **иерархический**, **сетевой** и **реляционный**.



В **иерархической** базе данных записи образуют особую структуру, называемую **деревом** (см. рисунок). При таком способе организации каждая запись может принадлежать только одному "родителю" (более правильный термин — "**владелец отношения**"). В качестве примеров такого рода отношений можно привести следующие: *организация — [**основная работа**] — работник*, *банк — [**вклад**] — сберкнижка*, *футбольная команда — [**хозяин поля**] — матч* и т.п.

Как следует из описанного выше, любой компонент дерева однозначно определяется путем, по которому мы можем его достигнуть, начиная с главного (верхнего) элемента. Отметим, что типичными примерами иерархического способа организации являются хорошо известная система вложенных каталогов в операционной системе, или так называемое генеалогическое дерево, представляющее собой графическое представление родословной.

В **сетевой** базе данных связи разрешено устанавливать произвольным образом, без всяких ограничений, поэтому запись может быть найдена значительно быстрее (по наиболее короткому пути). Такая модель лучше всего соответствует реальной жизни: один и тот же человек является одновременно и работником, и клиентом банка, и покупателем, т.е. запись с информацией о нем образует довольно густую сеть сложных связей. В определенном смысле наличие подобных связей моделирует реальные связи между объектами внешнего мира. Трудность состоит в том, что указанную организацию БД, к сожалению, сложно реализовать.

Хотя описанные выше способы являются более универсальными, на практике распространен самый простой тип организации данных — **реляционный**. Слово "*реляционный*" происходит от английского "*relation*", что значит *отношение*. Строгое определение отношения достаточно математизировано, поэтому на практике обычно пользуются следствием из него: поскольку отношения удобно представлять в виде таблиц, то говорят, что реляционные базы — это базы с табличной

формой организации. В качестве примера рассмотрим следующий фрагмент базы, опубликованной в статье [2] дополнительной литературы:

№	Персонаж	Профессия	Особые приметы	Герой
1	Буратино	деревянный человечек	длинный нос	да
2	Папа Карло	шарманщик		да
3	Карабас Барабас	директор кукольного театра	борода до пола	нет
4	Дуремар	фармацевт	характерный запах тины	нет

Отметим еще одну важную особенность реляционной модели данных. Поскольку в отличие от иерархической и сетевой организации баз в реляционных БД отсутствует понятие ассоциативных связей между парами записей, приходится их специальным образом моделировать. Для этой цели в записях создаются дополнительные поля, в которых ставится ссылка на требуемую запись, например, поля с названием организации и ведомства на следующем рисунке:

Таким образом, задаются необходимые отношения между полями в случае реляционной БД.

По мере развития вычислительной техники и роста ее возможностей роль электронных баз данных, несомненно, будет возрастать. Поэтому умение пользоваться этим видом компьютерной информации является обязательной частью образования современного человека.

Основная литература

1. *Семакин И., Залогова Л., Русаков С., Шестакова Л.* Информатика. Учебник по базовому курсу (7—9-е классы). М.: Лаборатория базовых знаний, 1998, 464 с.

2. Информационная культура: Кодирование информации, информационные модели. 9—10-е классы: Учебник для общеобразовательных учебных заведений. М.: Дрофа, 2000, 208 с.

3. Информатика: Энциклопедический словарь для начинающих / Сост. — Д.А. Поспелов. М.: Педагогика-Пресс, 1994, 352 с.

4. *Шафрин Ю.А.* Информационные технологии (в 2 ч.). М.: Лаборатория базовых знаний, 2000.

Дополнительная литература

1. *Бойко В.В., Савинков В.М.* Проектирование баз данных информационных систем. М.: Финансы и статистика, 1989, 351 с.

2. *Брызгалов Е.В., Шестаков А.П.* Уроки Access. "Информатика и образование" № 7, 1999, с. 18—29.

* * *

2. Решение расчетной задачи с использованием математических функций в среде программирования.

Задача

Рассчитать таблицу углов преломления (в градусах) для среды с относительным показателем преломления n. Угол падения задавать через 10 градусов от 0 до 90. Расчеты выполнить по приведенным ниже формулам:

1. $pi = 4 \cdot \mathrm{arctg}1$ — если транслятор не содержит значения константы π.
2. $r_1 = (g_1/180) \cdot pi$ — перевод угла падения из градусов в радианы.
3. $si = \sin r_1/n$ — sin угла преломления.
4. $r_2 = \mathrm{arctg}(si/\mathrm{sqrt}(1 - si \cdot si))$ — вычисление arcsin величины si; результат — угол преломления в радианах.
5. $g_2 = (r_2/pi) \cdot 180$ — перевод угла преломления из радиан в градусы.

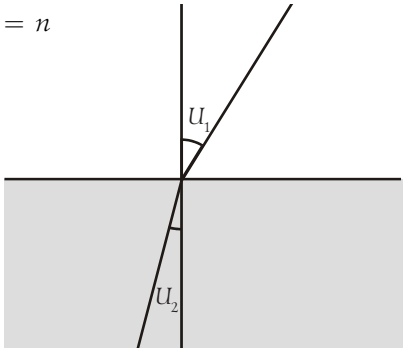
Здесь g_1 и r_1 — угол падения в радианах и градусах,
g_2 и r_2 — угол преломления.

Примечание. Пояснение происхождения данных формул дается ниже, хотя благодаря формальности алгоритмов знание этого совсем необязательно.

Краткая справка

Пусть вас не пугает физика, закон преломления света на границе двух сред необычайно простой:

$$\frac{\sin U_1}{\sin U_2} = n$$



Здесь U_1 называется углом падения, а U_2 — углом преломления. Отношение синусов этих углов для заданных двух сред и есть та самая константа n, которая задана в условии задачи. Возьмем для определенности границу между воздухом и водой, тогда $n = 1{,}33$.

Вот, собственно, и вся необходимая нам сейчас физика. Гораздо большие неприятности нас ожидают при попытке выполнить расчеты по этой, казалось бы, простой формуле. Здесь мы встретим некоторые неожиданные осложнения. Во-первых, все вычисления компьютер *всегда производит в радианах, а не в градусах,* значит, необходимо выполнить соответствующий переход от одних единиц к другим (в скобках заметим, что в некоторых старых версиях языков, например в GW BASIC, это породит еще одну дополнительную трудность — придется задавать константу π, которую транслятор "не знает"). Второе специфическое препятствие, поджидающее нас на этом пути, состоит в том, что в "базовом" наборе тригонометрических функций ($\sin x$, $\cos x$, $\mathrm{arctg}\, x$) отсутствует необходи-

мый для задачи $\arcsin x$ — придется выражать его через $\operatorname{arctg} x$. В итоге получится приведенная выше последовательность формул для вычислений (см. условие задачи).

Решение

Язык Паскаль

```
program svet(input,output);
var n, r1, r2, g1, g2, si: real; i: integer;
begin
write('n= '); readln(n);
for i := 0 to 9 do
  begin
   g1 := 10 * i; {угол падения в градусах}
   r1 := (g1/180) * pi; {угол в радианах}
   si := sin(r1)/n; {sin угла преломления}
  r2 := arctan(si/sqrt(1 - sqr(si))); {угол в рад.}
   g2 := (r2/pi) * 180; {угол прелом. в град.}
   writeln(g1:9:4,g2:9:4)
  end
end.
```

Язык Бейсик

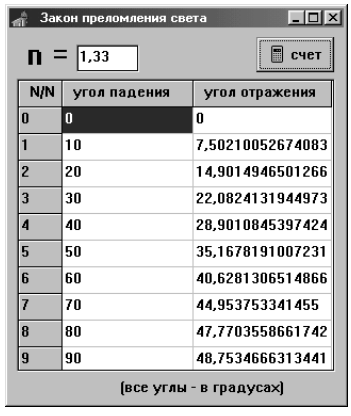
```
input "n=";n
pi = 4 * atn(1)
for i = 0 to 9
g1 = 10 * i 'угол падения в градусах
r1 = (g1/180) * pi 'угол в радианах
si = sin(r1)/n 'sin угла преломления
r2 = atn(si/sqr(1 - si * si)) 'угол в рад.
g2 = (r2/pi) * 180 'угол прелом. В град.
print g1, g2
next
```

Язык Си

```
#include <stdio.h>
#include <math.h>
void main()
{float n, r1, r2, g1, g2, si, pi; int i;
printf("N= "); scanf("%f", &n);
pi = 4 * atan(1);
for (i = 0; i <= 9; i++)
   {g1 = 10*i; //угол падения в градусах
    r1 = (g1/180) * pi; //угол в радианах
    si = sin(r1)/n;   //sin угла преломления
    r2 = atan(si/sqrt(1 - si * si)); // угол в рад.
    g2 = (r2/pi) * 180; // угол прелом. в град.
    printf("%9.4f %9.4f \n", g1, g2);
   };
}
```

Примечание. В языке Си в модуле `math.h` определена функция `asin` для вычисления $\arcsin x$, но для общности алгоритмов в решении она не использована.

И в заключение еще одно дополнительное решение описываемой задачи, выполненное в более современной среде программирования — Delphi. В ней также используется язык Паскаль, но дополнительно разработаны методы быстрого создания органов управления Windows-интерфейса. Общий вид окна с вычислениями приведен на рисунке.



Пользователь вводит необходимое значение n и нажимает на кнопку "Счет". После этого компьютер заполняет приведенную на рисунке таблицу.

Текст программы, за исключением стандартной формируемой системой автоматически "шапки", помещен ниже. Фактически программа состоит всего из двух процедур: первая, срабатывающая единственный раз в момент создания окна (ее название — `FormCreate`, она просто формирует заголовок таблицы), и вторая, `BitBtn1Click`, которая вызывается каждый раз по нажатию мышкой на кнопку "Счет". Последняя процедура очень похожа на соответствующее решение на Паскале. Главное отличие заключается в способах ввода исходного n и вывода полученных результатов; в тексте программы комментарии в этих местах выделены подчеркиванием.

Для ввода n в спроектированном окне предусмотрена специальная область редактирования текстовой строки, имя которой в программе — `input_n`. Набранный пользователем текст берется из свойства `input_n.text` и с помощью стандартной процедуры `StrToFloat` он преобразуется в вещественное число с плавающей запятой.

Вывод результатов производится в специальную таблицу `results`, а точнее, в ее ячейки `results.cells[x,y]`, первый индекс у которых означает номер столбца таблицы, а второй — номер строки. При этом также используются соответствующие стандартные процедуры перевода чисел в строковое представление, поскольку клетки таблицы предназначены для сохранения текстовой информации.

```
{$R *.DFM}
procedure TForm1.FormCreate(Sender: TObject);
begin
  results.cells[0,0] := '  N/N';
  results.cells[1,0] := '  угол падения';
  results.cells[2,0] := '  угол отражения';
end;
procedure TForm1.BitBtn1Click(Sender: TObject);
var n, r1, r2, g1, g2, si: real;
    i: integer;
begin
  {считать значение N}
  n := StrToFloat(input_n.text);
for i := 0 to 9 do
 begin
  g1 := 10 * i; {угол падения в градусах}
  r1 := (g1/180) * pi; {угол падения в радианах}
  si := sin(r1)/n;   {sin угла преломления}
  {arcsin(si)}
  {угол преломления в радианах}
  r2 := arctan(si/sqrt(1 - sqr(si)));
  {угол преломления в градусах}
  g2 := (r2/pi) * 180;
  {занести результаты в таблицу}
  results.cells[0, i + 1]:=IntToStr(i);
  results.cells[1, i + 1]:=FloatToStr(g1);
  results.cells[2, i + 1]:=FloatToStr(g2);
 end
end;
```

Видно, что приведенное решение ненамного сложнее классического "паскалевского".

Полный набор файлов Delphi-проекта, как обычно, можно найти на сайте **http://info-bilet.narod.ru**.

Жидкие кристаллы

Некоторое время тому назад необычной популярностью в США пользовалась новинка ювелирного производства, получившая название “перстень настроения”. За год было продано 50 миллионов таких перстней, т.е. практически каждая взрослая женщина имела это ювелирное изделие. Что же привлекло внимание любителей бижутерии к этому перстню? Оказывается, он обладал совершенно мистическим свойством реагировать на настроение его владельца. Реакция состояла в том, что цвет камешка перстня следовал за настроением владельца, пробегая все цвета радуги, от красного до фиолетового. Вот это сочетание таинственного свойства угадывать настроение, декоративность перстня, обеспечиваемая яркой и меняющейся окраской камешка, плюс низкая цена и обеспечили успех “перстню настроения”.

Конечно, каждому владельцу перстня хотелось знать его секрет слежения за настроением. Однако ничего толком не было известно, говорилось только, что камешек перстня сделан на жидком кристалле...

Разумеется, само по себе словосочетание “жидкий кристалл” было известно широкой публике и до того. Жидкокристаллические панели к тому времени широко применялись в электронных часах и калькуляторах. А сегодня мониторы на их базе все активнее теснят добрые старые изделия на электронно-лучевых трубках. Одним словом, мало кто сейчас не слышал о жидких кристаллах, а уж пользуется ими практически каждый человек.

Как ни странно, но жидкие кристаллы старше электронно-лучевых трубок почти на десять лет.

Первое описание этих веществ было сделано еще в 1888 году австрийским ботаником Ф.Райнитцером (Friedrich Reinitzer). Он заметил, что синтезированное им вещество холестерилбензоат имеет как бы две точки плавления. При нагревании образца этого материала до температуры 145° он наблюдал, как кристалл превращается в мутную жидкость. Далее, при достижении температуры 179°С, жидкость становилась прозрачной.

Самое интересное, что в “мутной” фазе вещество обнаружило необычные свойства — оно обладало двупреломлением. Иными словами, под поляризационным микроскопом вещество по-разному пропускало свет в зависимости от угла поворота образца.

Явление двупреломления — это типично кристаллический эффект, и демонстрируется оно в школьном курсе физики на примере полевого шпата. Объясняется это явление анизотропией кристалла, т.е. различными механическими, электромагнитными и оптическими свойствами кусочка вещества по разным направлениям. Это обусловлено ориентацией молекул внутри вещества вдоль некоторой оси. На *рис*. 1 показано расположение молекул в твердом кристалле, жидком кристалле и жидкости.

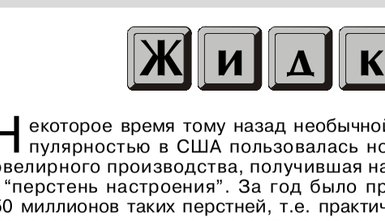


Рис. 1. Расположение молекул в различных фазах вещества

Впрочем, понятие анизотропии можно продемонстрировать и на обычном полене: все знают, что его гораздо легче расколоть вдоль волокон, чем против.

Долгое время жидкие кристаллы воспринимались лишь как своеобразный курьез, и никто не знал, как их применить на практике. И только в 1930 году исследователи из британской корпорации Marconi получили патент на их промышленное применение. Впрочем, дальше этого дело не пошло, поскольку технологическая база в то время была еще слишком слаба.

Первый настоящий прорыв совершили ученые Фергесон (Fergason) и Вильямс (Williams) из корпорации RCA (***R**adio **C**orporation of **A**merica*). Один из них создал на базе жидких кристаллов термодатчик, используя их избирательный отражательный эффект, другой изучал воздействие электрического поля на нематические кристаллы. И вот в конце 1966 г. корпорация RCA продемонстрировала настольные цифровые часы с жидкокристаллическим индикатором. Именно тогда впервые был применен термин LCD (***l**iquid **c**rystal **d**isplay* — дисплей на жидких кристаллах). Значительную роль в развитии LCD-технологии сыграла корпорация Sharp. Она и до сих пор находится в числе технологических лидеров. Первый в мире калькулятор CS10A был произведен в 1964 г. именно этой корпорацией. В октябре 1975 г. тоже на базе LCD были изготовлены первые компактные цифровые часы. Во второй половине 70-х годов начался переход от восьмисегментных жидкокристаллических индикаторов к производству матриц с адресацией каждой точки. Так, в 1976 г. Sharp выпустила черно-белый телевизор с диагональю экрана 5,5 дюйма, выполненного на базе LCD-матрицы разрешением 160 × 120 пикселей.

Но вернемся к первоистокам и рассмотрим более подробно, что же собой представляют жидкие кристаллы. Во-первых, вещество жидкого кристалла должно состоять из молекул, имеющих удлиненную палочкообразную форму (см. *рис*. 2).

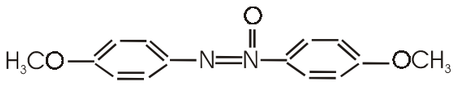


Рис. 2. Молекула параазоксианизола обладает жидкокристаллическими свойствами в интервале температур 114—135°С

Силы взаимодействия выстраивают их параллельно друг другу, и ведут они себя как обычные молекулы жидкости, но с учетом единственного ограничения — при всех перемещениях должно сохраняться (в целом) некоторое выделенное направление длинных осей, что, собственно, и приводит к анизотропии. Жидкость описанного выше типа принадлежит обширному классу веществ, называемых “нематическими жидкими кристаллами”. Слово “немос” по-гречески “нить”, и действительно, молекулы таких жидких кристаллов напоминают бусинки, укрепленные на нити.

Существуют и другие типы молекулярной архитектуры, создающие анизотропию. Так, укладка молекул слоями и пачками приводит к еще одному классу жидких кристаллов — сметическим. Такая упаковка молекул создает анизотропию не только оптических, но и механических свойств, поскольку слои легко смещаются относительно друг друга. Название этой группы связано с греческим словом “смектос” (мыло). Такое расположение молекул характерно для мыльных растворов, эмульсий и т.д.

Третьим распространенным типом жидких кристаллов являются холестерические, в которых молекулы укладываются в плоскостях подобно описанным выше нематическим кристаллам, но сами плоскости повернуты друг относительно друга. Вектор, связанный с длинной осью, так называемый “директор”, описывает в пространстве спираль. Названием этот класс жидких кристаллов обязан печально известному холестерину, у которого впервые были обнаружены подобные свойства.

Прежде всего было найдено, что воздействие электрического поля на жидкие кристаллы приводит к электрооптическим эффектам, не имеющим аналогов среди прочих оптических сред. Электрооптическая ячейка состоит из двух стекол, между которыми находится тонкий слой жидкого кристалла (см. *рис*. 3).

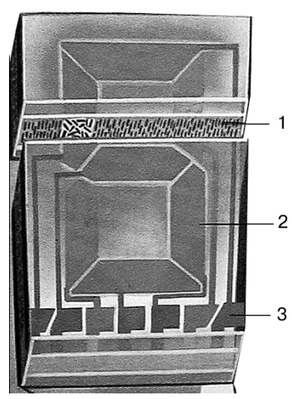


Рис. 3. Электрооптическая ячейка

Таким образом, получают как бы прозрачный конденсатор, диэлектриком внутри которого служит слой жидкого кристалла (1). В нормальном состоянии его молекулы “выстроены” по вертикали (т.е. находятся практически в кристаллической фазе), благодаря чему свет почти беспрепятственно проникает к нижнему зеркалу и отражается от него.

Теперь достаточно создать между электродами (3) некоторую разность потенциалов, чтобы нарушить ориентацию молекул жидкого кристалла, и они начинают препятствовать отражению. Жидкость между электродами становится как бы мутной (переход в жидкокристаллическую фазу). Свет, до сих пор беспрепятственно проходивший через жидкий кристалл, рассеивается, и участки с повышенной напряженностью поля становятся видны. Буквы и цифры порождаются различными комбинациями заряженных и незаряженных сегментов.

Электропроводящие участки поверхности стекла могут быть выполнены в виде букв или любых геометрических фигур. На *рис*. 3 изображены традиционные сегменты (2), которые используются почти во всех электронных часах и калькуляторах. Подавая на них соответствующие напряжения, можно формировать прозрачные и непрозрачные участки, то есть с ничтожными затратами энергии создавать подвижные и неподвижные картины. Использование динамического рассеяния на слое жидкого кристалла толщиной в несколько микрометров позволяет получить изображение, затрачивая мощность порядка микроватт. При этом из-за тонкости слоя жидкого кристалла необходимое напряжение на ячейке составит всего несколько вольт.

Широкому распространению жидкокристаллических индикаторов довольно долгое время мешала достаточно дорогая технология создания миниатюрных электродов. Ситуация изменилась с изобретением специального токопроводящего материала на резиновой основе, названного “эластомером”. Тонкие полоски этого материала (эластомерные соединители) передают электрические импульсы на электроды, выгравированные в слоях индикатора (см. *рис*. 4).

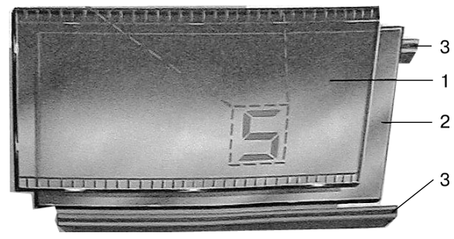


Рис. 4. Схема LCD-индикатора для наручных часов
1 — прозрачное стекло;
2 — отражающее стекло;
3 — эластомерный соединитель.

На схеме не пронумерованы жидкокристаллические ячейки, располагающиеся между слоями 1 и 2.

В конечном итоге пользователь видит, например, вот такой циферблат (см. *рис*. 5).

Рис. 5

Удивительные превращения происходят с лучом света при взаимодействии с холестерическим жидким кристаллом, т.е. периодической спиралью. Освещенный белым светом, он кажется окрашенным и при поворотах (при изменении угла наблюдения) начинает переливаться всеми цветами радуги. Этот эффект возникает потому, что в различных направлениях чешуйки кристалла, отражающие свет, расположены на различных расстояниях и отражают из белого цвета лишь волны с определенной длиной.

Такой простой и красивый эффект дает ошеломляющую возможность.

Например, пусть какой-то участок поверхности нагрет на сотые доли градуса выше окружающих. Достаточно нанести на поверхность пленку холестерического жидкого кристалла, чтобы в “горячей” точке шаг спирали чуть-чуть увеличится и на пленке появилась точка иного цвета. Покрыв готовое изделие (это может быть интегральная схема или деталь двигателя) слоем холестерического вещества, можно получить цветную картину теплового распределения, на которой контрастными пятнами проступают любые дефекты и неоднородности, даже глубоко скрытые. Кристаллы этого же типа были использованы в перстнях, о которых шла речь в начале нашей статьи.

Поскольку цвет окраски жидкого кристалла так чувствителен к температуре, этот эффект можно использовать при преобразовании инфракрасного изображения в видимое. Основным элементом такого устройства является пленка холестерического жидкого кристалла, помещенная на тонкую черную мембрану. Мембрана поглощает сфокусированное на ней инфракрасное излучение и передает тепло слою жидкого кристалла. Цвет жидкокристаллической пленки (в отраженном свете) зависит от температуры, поэтому при освещении пленки белым светом получается видимое изображение инфракрасного излучения. Ранее для преобразования инфракрасного излучения в видимое использовали значительно более сложные устройства и дорогие фотоэмиссионные или фосфоресцирующие электронные устройства. Предельная простота и малая стоимость позволяют, например, оснастить каждого солдата качественным устройством ночного видения.

Из смеси холестерических веществ можно изготавливать температурные индикаторы в интервале температур от 20 до 250°С. Индикаторы представляют собой тонкую гибкую пленку жидкого кристалла, заключенную между двумя полимерными пленками. Такую пленку можно накладывать на поверхности деталей для регистрации температурных градиентов в различных направлениях.

Жидкие кристаллы холестерического типа (или их смеси) весьма чувствительны и к присутствию паров различных химических веществ. Даже несколько миллионных долей вещества достаточно для изменения структуры жидкого кристалла. Излишне говорить, что это имеет громадную практическую ценность.

Возвращаясь к нематическим жидким кристаллам и дисплеям на их основе, отметим, что, кроме рассмотренных выше пассивных ЖК-панелей, отражающих падающий на них свет, существуют активные ЖК-панели. В них за панелью находится источник света, а сама панель работает на просвет. В последнее время бурно развиваются технологии использования активных панелей в телевизионных приемниках и мониторах, но об этом — в следующий раз.

Использованные источники информации

1. Знакомьтесь: компьютер. М.: Мир, 1989.
2. Яndex-энциклопедия. **www.Yandex.ru**.
3. LCD-мониторы. **www.monitors.narod.ru**.
4. Virtual Textbook. Introduction to Liquid Crystals. **http://plc.cwru.edu/tutorial**.
5. Жидкие кристаллы. Реферат. Автор неизвестен. МГТУ им. Баумана, кафедра химии, 1998. **www.referats.ru**.

Азы информатики

А.А. ДУВАНОВ

Материалы Роботландского университета

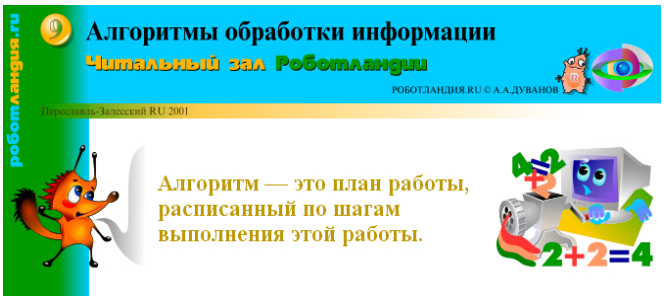
Книга 1. См. № 1, 2/2002
Книга 2. См. № 5, 6, 7, 8, 9, 11, 13, 14/2002

Демо-версию гипертекстовой книги (700 Кб) можно скопировать с адреса:
ftp://ftp.botik.ru/rented/robot/univer/azinfd.zip

Книга 2. В мире информации (продолжение)

Книга для ученика

9. АЛГОРИТМЫ ОБРАБОТКИ ИНФОРМАЦИИ



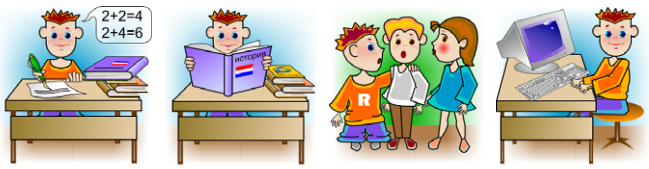
Читальный зал

Понятие алгоритма

Алгоритм — это план выполнения работы, расписанный по шагам.

Вася составил план выполнения домашнего задания:

1. Решу задачу, которую задали по математике.
2. Прочитаю параграф из учебника истории.
3. Погуляю часок на улице с ребятами.
4. Займусь заданиями по информатике.



1 2 3 4

Этот план является алгоритмом. В нем четыре пункта. Каждый пункт описывает отдельный этап работы. Пункты выполняются последовательно, друг за другом, в том порядке, в котором записаны в плане. Выполняя указания плана, Вася "шагает" от первого пункта к последнему. Поэтому пункты алгоритма называются шагами.

Тот, кто придумывает алгоритм, называется **составителем** алгоритма.

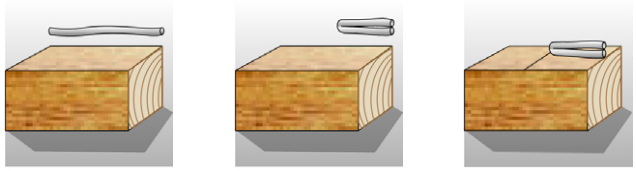
Тот, кто выполняет алгоритм, называется **исполнителем** алгоритма.

В приведенном выше примере Вася сам написал алгоритм домашнего задания и сам его выполнил. Значит, Вася был и составителем, и исполнителем алгоритма.

К Васе зашел соседский малыш Костя: ему нужно найти середину деревянного бруска, чтобы распилить его на две равные части.

Вася предложил такой алгоритм:

1. Отрежь кусок бечевки точно по длине бруска.
2. Сложи бечевку пополам.
3. Приложи один конец сложенной бечевки к началу бруска и отметь на бруске ее конец. Это и есть середина бруска.



1 2 3

Костя побежал домой, выполнил алгоритм Васи, и у него все получилось!

В этом примере составителем алгоритма был Вася, а исполнителем — Костя.

Составитель алгоритма должен проявить свои знания, изобретательность и умение разделить сложную работу на простые шаги.

Работа исполнителя более проста. Все, что от него требуется, — это аккуратно выполнить каждый расписанный составителем шаг.

Когда исполнителем является человек, он вполне может проявить инициативу и выполнить какие-то шаги алгоритма по-своему.

Исполнительное устройство на это не способно. Компьютер или робот всегда точно выполняют указания алгоритма, если, конечно, они находятся в исправном состоянии.

Компьютерные алгоритмы

Компьютер — это универсальный исполнитель, выполняющий алгоритмы обработки информации.

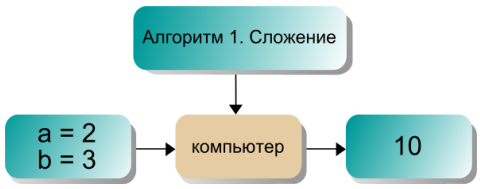
В отличие от человека этот исполнитель не проявляет инициативы, он строго выполняет полученные предписания, работает очень быстро и не совершает ошибок, если их нет в самом алгоритме.

Рассмотрим несколько примеров.

Пример 1. Сложение

Компьютер работает по такому алгоритму:
1. Сложи числа a и b.
2. Результат первого действия умножь на 2.
3. Запиши результат второго действия на экране.

Вася Кук ввел в компьютер данные: $a = 2$, $b = 3$.
Компьютер написал: результат = 10.



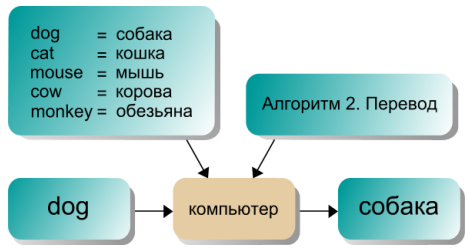
Пример 2. Перевод

В компьютер введены данные:
dog = собака;
cat = кошка;
mouse = мышь;
cow = корова;
monkey = обезьяна.
(Эти английские слова читают так:
“dog” — “дог”, “cat” — “кэт”, “mouse” — “маус”, “cow” — “кау”, “monkey” — “манки”.)
Затем у Васи с компьютером произошел диалог:

Вопрос компьютеру	Ответ компьютера
dog	собака
cow	корова
mouse	мышь

Здесь обработка информации заключается в переводе слов с английского языка на русский. Алгоритм перевода таков: компьютер отыскивает английское слово в левом столбике заранее заданного словарика. Найдя его, выводит русское слово из той же строки правого столбца.

Конечно, и алгоритм работы, и словарик должны быть введены в компьютер перед началом работы.

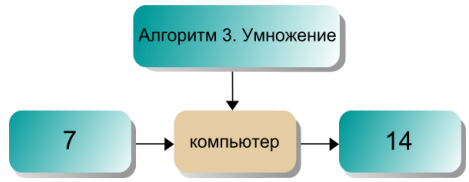


Пример 3. Умножение

Вася не знал алгоритма, по которому работал компьютер. О том, как шла обработка данных, можно судить по следующему диалогу:

Вася	Компьютер
кот	не понимаю
1	2
3	6
7	14

Вася отгадал алгоритм обработки информации: исходной информацией должно быть число, при обработке это число умножается на 2.

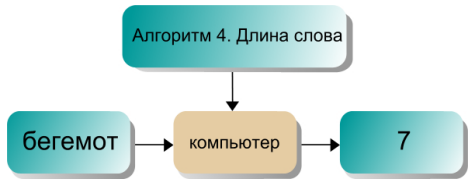


Пример 4. Длина слова

Вася знал, что в компьютер введен новый алгоритм обработки информации. Но какой это алгоритм? Кук попытался “поговорить” с машиной, чтобы это узнать:

Вася	Компьютер
1	не понимаю
а	1
кот	3
бегемот	7
крокодил	8

Васе нетрудно было догадаться, что компьютер подсчитывает число букв в словах.



Компьютерные программы

Алгоритм вводится в компьютер в виде программы.

Программа — это запись алгоритма на языке, понятном компьютеру.

Такие языки называют **языками программирования**.

Алгоритмы, записанные на обычном языке, понятны человеку. Для того чтобы они стали понятны компьютеру, их нужно переписывать на языке программирования.

Ниже приводится план, по которому обычно работают **программисты** — люди, разрабатывающие компьютерные программы.

План работы программиста

1. Сначала надо сформулировать задачу так, чтобы можно было легко составить алгоритм выполнения будущей программы.
2. Затем надо написать алгоритм работы программы. Такую запись обычно выполняют на родном языке (русском, например).
3. Когда алгоритм готов, его надо перевести в программу, то есть записать на каком-нибудь языке программирования.
4. Программу загружают в компьютер и приступают к отладке. Отладка состоит в пробной работе с программой, во время которой выявляются и устраняются ошибки.
5. Когда программа отлажена, ее передают пользователю для работы.

Пример разработки программы

Задача. Вычислить площадь прямоугольника.

1. Сформулируем задачу более удобным для программирования образом.

Пусть a и b — стороны прямоугольника. Обозначим через d площадь. Тогда для вычисления площади можно использовать формулу:

$d = ab$

2. Алгоритм.

1) Запросить у пользователя числа a и b.
2) Найти произведение ab.
3) Вывести на экран результат предыдущего действия.

3. Программа. Для работы решено было использовать компьютер Малыш. На языке программирования Малыша программа записалась так:

ввод (*a*)	Ввод числа *a*
ввод (*b*)	Ввод числа *b*
d := *a* * *b*	Вычисление площади
вывод (*d*)	Вывод результата

4. Проверка. Для проверки правильности работы программы было заготовлено несколько различных наборов входных данных:

Номер теста	*a*	*b*	Ответ	Результат программы
1	0	0	0	?
2	3	5	15	?
3	10	3	30	?

При работе программы выяснилось, что она выдает правильные ответы на все заготовленные входные значения.

5. Передача пользователю. Программа была передана пользователям для работы.

Конспект

Алгоритм — это план работы, расписанный по шагам выполнения этой работы.

Составитель алгоритма — это тот, кто придумывает алгоритм.

Исполнитель алгоритма — это тот, кто выполняет алгоритм.

Языки программирования — это специальные языки, на которых записывают алгоритмы для компьютера.

Программа — это запись алгоритма на языке программирования.

План работы программиста:

1. Сформулировать задачу.
2. Написать алгоритм.
3. Перевести алгоритм в программу.
4. Отладить программу.

Вопросы

1. Назовите три основных действия, которые можно совершать над информацией.

2. Как называется план обработки информации?

3. Дайте определение алгоритма.

4. Кого называют составителем алгоритма?

5. Кого называют исполнителем алгоритма?

6. Запишите ваш алгоритм выполнения домашнего задания.

7. Расскажите алгоритм нахождения четвертой части длинного бруска.

8. Составьте компьютерный алгоритм нахождения суммы трех чисел.

9. Расскажите алгоритм работы компьютерной программы перевода слов.

10. Составьте компьютерный алгоритм нахождения половины произведения двух чисел.

11. Составьте компьютерный алгоритм нахождения суммы длин двух слов.

12. Что такое компьютерная программа?

13. Чем алгоритм отличается от программы?

14. Где хранятся компьютерные программы?

15. Кто такие программисты?

16. Кто такие пользователи?

17. Что такое язык программирования?

18. Расскажите план работы программиста.

19. Что такое отладка программы и зачем она нужна?

20. Как программист или пользователь узнает, что в программе есть ошибка?

21. Как следует поступить пользователю, если он обнаружил ошибку в программе?

22. Васю Кука вызвали к доске, чтобы он решил пример:

$$(35 + 5) : 4 - 2 \cdot 3$$

Запишите алгоритм, по которому Вася обработал информацию.

23. Вася Кук и Катя Пушкова играют в такую игру. Кук придумывает алгоритм, но не рассказывает о нем. Катя сообщает Васе данные; Вася, обработав информацию, сообщает результат. Катя должна отгадать алгоритм Васи.

Катя	Вася
Три медведя	Тре мидвидя
Пришли они в лес	Прешле оне в лис

По какому алгоритму Кук обрабатывает информацию?

Задания на дом

Вариант 1

1. Запишите в тетради алгоритм приготовления блюда из макарон.

2. Утром в магазине мама покупала обычно 3 маленьких булочки, буханку хлеба, 2 пакета молока и 2 килограмма овощей. А поскольку цены на булочку, хлеб, молоко и овощи каждый день были разными, мама попросила Олю составить алгоритм и написать компьютерную программу для ежедневного подсчета истраченных денег. Помогите Оле справиться с этой работой.

Вариант 2

1. Запишите в программе "Блокнот" алгоритм приготовления блюда из макарон.

Распечатайте полученный текст на принтере.

2. По книге А.Я. Котова "Вечера занимательной арифметики": М.Ю. Лермонтов был не только великим поэтом, но и большим любителем математики. Чтобы доставить удовольствие окружающему обществу, он, к примеру, угадывал число, ничего не спрашивая у собеседника, а предлагая ему выполнить над задуманным числом некоторые арифметические действия. Например:

Задумать число.
Прибавить к нему 25.
Прибавить еще 10.
Отнять 15.
Вычесть задуманное число.
Остаток умножить на 4.
Полученное число разделить на 2.
Получится 40. Верно?

Почему можно безошибочно угадывать результат, в чем состоит разгадка этого фокуса? Запишите в программе "Блокнот" алгоритм и компьютерную программу для отгадывания задуманных чисел.

Вариант 3

1. Фокусник предложил: "Задумайте число. Отнимите 1. Остаток удвойте и прибавьте первоначально задуманное число. Скажите результат, и я угадаю задуманное число". Кто в этой ситуации является составителем алгоритма, а кто — исполнителем? Составьте алгоритм для подсчета результата и напишите компьютерную программу.

2. Найдите в литературе описания других математических фокусов, составьте алгоритм и напишите по нему программу компьютерного фокусника.

Рекомендуемая литература:

— *Перельман Я.И.* "Живая математика".
— *Аменицкий Н.Н., Сахаров И.П.* "Забавная арифметика".
— *Котов А.Я.* "Вечера занимательной арифметики".
— *Кордемский Б.А.* "Математическая смекалка".

Практикум

Малыш

Компьютер Малыш удобен для программирования простых вычислений.

У Малыша есть:

— Монитор для отображения хода выполнения программы.
— Четыре ячейки памяти с именами *a*, *b*, *c*, *d* для хранения чисел.
— Поле для записи программы.
— Меню для выбора алгоритма.
— Кнопка "Пуск" для запуска программы.

Порядок работы с Малышом:

1. Выбрать алгоритм (в меню выбора).
2. Записать программу (в поле программ).
3. Выполнить программу (кнопка "Пуск").

Алгоритмы Малыша

Исполнитель Малыш — это учебный компьютер. Начинающему программисту трудно сразу написать программу целиком. Вот почему у Малыша есть меню выбора алгоритма.

Когда алгоритм выбран, в программном поле появляется соответствующая программа. Но в ней есть пропуски, заполнить которые вам придется самостоятельно.

Исполнитель настроен на работу со следующими алгоритмами:

— Нахождение площади прямоугольника.
— Нахождение периметра треугольника.
— Нахождение периметра прямоугольника.
— Решение задачи о болтах и гайках.

Цена болта — *b* рублей, а гайки — *c* рублей. Купили *a* болтов и в 2 раза больше гаек. Найти стоимость покупки.

Язык программирования Малыша

Программа для Малыша записывается в виде списка команд. Ниже приводится пример программы для нахождения суммы двух чисел:

ввод (*a*)	Ввод первого числа
ввод (*b*)	Ввод второго числа
d := *a* + *b*	Нахождение суммы
вывод (*d*)	Вывод результата

В языке Малыша всего три команды.

Команда ввода

Эта команда позволяет ввести число в ячейку памяти. Например, по команде:

ввод (a)

исполнитель предложит пользователю записать число. Заданное число будет отправлено на хранение в ячейку a.

Для ввода числа в ячейку b нужно написать такую команду:

ввод (b)

Команда вывода

Команда вывода показывает содержимое ячейки на мониторе исполнителя.

Например, содержимое ячейки d будет показано на экране при помощи такой команды:

вывод (d)

Команда присваивания

Команда имеет левую часть (имя ячейки), правую часть (арифметическое выражение) и знак присваивания — два символа ":=".

Работает команда так:

1. Сначала вычисляется арифметическое выражение.
2. Затем полученное значение записывается в ячейку, имя которой записано слева от знака присваивания.

Например, команда:

$$d := 2 + 3$$

будет работать так. Сначала вычисляется сумма, затем полученное значение (число 5) записывается в ячейку d. Старое содержимое ячейки d при этом пропадает.

А как будет работать команда, записанная ниже?

$$d := a + b$$

Как исполнитель будет складывать буквы?

Вы, конечно, уже догадались: будут складываться не буквы, а числа, которые находятся в ячейках a и b. Если в этих ячейках записаны числа 15 и 10, то после выполнения команды присваивания в ячейку d будет записано число 25.

Запись арифметического выражения в языке Малыша имеет свои особенности:

1. Выражение записывается в одну строчку.
2. Знак умножения никогда не опускается в записи.
3. Для записи операций используются следующие обозначения:

+	сложение
—	вычитание
*	умножение
/	деление

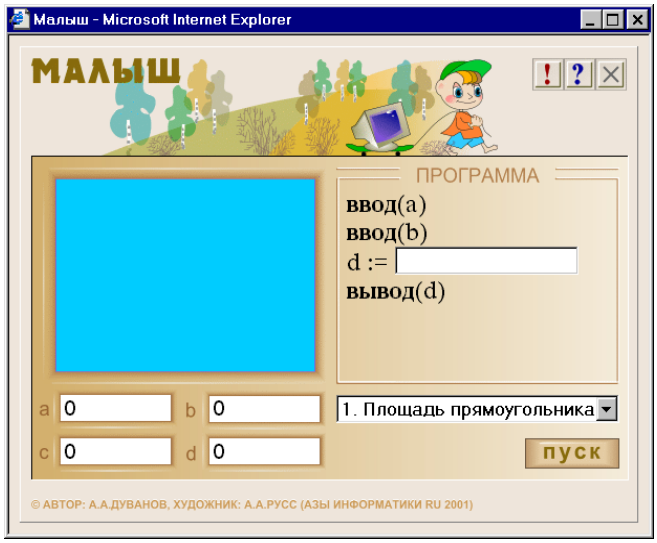
Например, привычная математическая запись:

$$\frac{25 - 2(3 + 2)}{5} + 4a$$

переписывается на языке Малыша так:

$$(25 - 2 * (3 + 2))/5 + 4 * a$$

Задание



Для каждого алгоритма исполнителя написать программу, отладить ее и решить предложенные ниже задачи.

1. Найти площади прямоугольников.

Ширина	Высота	Площадь	Проверка
25	48		Пуск
125	342		Пуск
1711	943		Пуск
1024	1024		Пуск
0	0		Пуск

2. Найти периметры треугольников.

a	b	c	Периметр	Проверка
82453	23834	99999		Пуск
777777	888888	666666		Пуск
25	2525	797979		Пуск
1024	0	32847		Пуск
0	0	0		Пуск

3. Найти периметры прямоугольников.

Ширина	Высота	Периметр	Проверка
9898	9898		Пуск
77777	99999		Пуск
878787	5656		Пуск
9712345	0		Пуск
0	0		Пуск

4. Найти стоимость покупки.

Число болтов	Цена болта	Цена гайки	Стоимость покупки	Проверка
315	25	15		Пуск
2567	7	35		Пуск
56	735	351		Пуск
346	56	0		Пуск
4665	0	0		Пуск

Зачетный класс

1. Сан Саныч придумал новую игру и написал соответствующий алгоритм. Петя взял алгоритм и перевел его в программу. Вася сел за компьютер и играл целый час. Запишите на месте вопросов ответы.

Составитель — ?
Исполнитель — ?
Программист — ?
Пользователь — ?

2. Программист написал программу:

ввод (a)
ввод (b)
$d := a/b$
вывод (d)

Какое число будет выведено на монитор, если по запросу программы пользователь ввел сначала 100, затем 20?

3. Программист написал программу:

ввод (b)
ввод (a)
$d := a * b - b/a$
вывод (d)

Какое число будет выведено на монитор, если по запросу программы пользователь ввел сначала 10, затем 5?

4. Программист написал программу:

ввод (a)
ввод (b)
ввод (c)
$d := (a + b + c)/3$
вывод (d)

Какое число будет выведено на монитор, если по запросу программы пользователь последовательно ввел числа: 10, 20, 30?

5. Программист написал программу:

ввод (c)
ввод (b)
ввод (a)
$d := 3 * (a - b)/c$
вывод (d)

Какое число будет выведено на монитор, если по запросу программы пользователь последовательно ввел числа: 10, 30, 50?

6. Перепишите формулу в виде, пригодном для команды присваивания:

$$\frac{a : b - 2b}{10}$$

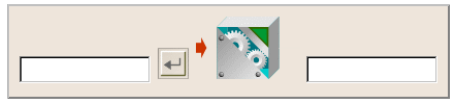
7. Перепишите формулу в виде, пригодном для команды присваивания:

$$\frac{ab - 2c - 1}{c}$$

8. Перепишите формулу в виде, пригодном для команды присваивания:

$$\frac{a - b(a - b)}{c}$$

9. Устройство ввода исполнителя испортилось и может принимать только 4 знака:



Что получится в результате обработки слова “иерархия” после того, как устройство ввода будет отремонтировано?

Результаты возможных опытов с исполнителем:

Вход	Выход
а	аа
он	но
кот	тк
лиса	ал
иерархия	?

10. Испорченное устройство ввода может принимать только 4 знака.

Что получится в результате обработки слова “программа” после того, как устройство ввода будет отремонтировано?

Результаты возможных опытов с исполнителем:

Вход	Выход
а	а
он	но
кот	ток
лиса	асил
программа	?

Книга для учителя

9. АЛГОРИТМЫ ОБРАБОТКИ ИНФОРМАЦИИ

Алгоритмы и программы

Этот урок имеет большое значение для курса информатики. Понятие алгоритма закрепляется в нем на базе конкретных алгоритмических построений, а понятие программы, описанное до этого совершенно абстрактно, получает явную поддержку практической деятельностью ребенка.

На что нужно обратить внимание:

— Для составления алгоритма нужно сформулировать задачу так, чтобы ее решение можно было легко разделить на простые части.

— Алгоритм состоит из шагов (пунктов).

— Каждый шаг описывает отдельный этап работы по решению поставленной задачи.

— Менять шаги алгоритма местами, как правило, нельзя, это приведет к неверному решению.

— Если порядок следования некоторых шагов не важен, то выполнение алгоритма можно поручить группе исполнителей. Алгоритмы, в записи которых присутствуют указания на распределение работы между исполнителями, работающими одновременно, называются параллельными (смотрите урок 7 книги 1 "Знакомство с компьютером").

Ниже показан параллельный алгоритм для вычисления суммы произведений.

Алгоритм вычисления $s = ab + cd$

Шаг	Исполнитель 1	Исполнитель 2
1.	$x := a * b$	$y := c * d$
2.	$s := x + y$	

Урок достаточно насыщен материалом, но при высоком темпе изучения или резерве времени полезно предложить детям составить несколько параллельных алгоритмов.

Тема параллельных вычислений очень важна для современной информатики: сейчас это один из самых главных способов повышения производительности компьютеров. Наиболее полно тема будет раскрыта в книгах "Алгоритмические этюды" и "Исполнители".

— Составителем алгоритма, как правило, является человек, а исполнителем — человек, животное, природа, техническое устройство.

— Бывают случаи, когда составителем алгоритма может стать другой алгоритм. Например, алгоритм поведения персонажа компьютерной игры автоматически генерирует алгоритмы поведения объекта в зависимости от окружающей игровой обстановки. Но так как алгоритм — генератор алгоритмов придумывает человек, то и автоматические алгоритмы можно отнести к творчеству человеческой мысли.

— Кроме человека, природа является великолепным составителем алгоритмов. Природа не только составитель алгоритмов, но и отличный программист: она записывает свои программы на языке генетических кодов.

— Важным качеством алгоритма является свойство массовости — способности алгоритма решать не одну конкретную задачу, а целый класс однотипных задач.

Этот алгоритм не обладает свойством массовости:

1. $d := 20 * 30$
2. вывод (d)

Этот алгоритм свойством массовости обладает:

1. ввод (a)
2. ввод (b)
3. $d := a * b$
4. вывод (d)

Малыш

Малыш имеет ключевое значение для начального курса информатики. Этот исполнитель решает следующие методические задачи:

— Показывает соотношение понятия алгоритма (плана выполнения) и программы (запись плана на языке программирования).

— Знакомит с тремя базовыми конструкциями языков программирования — команды: ввод, вывод, присваивание.

— Наглядно показывает способ хранения информации в ячейках памяти, соотношение между содержимым ячейки и ее именем, использование имен ячеек (переменных) в компьютерных программах.

— Позволяет на базе минимальных знаний и умений записать несколько "настоящих" программ и проверить их в работе.

— Демонстрирует важное свойство алгоритмов и программ — массовость: одна и та же программа способна решить не одну конкретную задачу, а целый класс однотипных задач.

— Погружает новичка в реальные технологические процессы программирования: тестирование, отладка, опытная эксплуатация.

Как видите, несмотря на простоту и даже примитивность Малыша, он решает очень серьезные педагогические задачи. Именно поэтому очень важно правильно построить работу детей с Малышом и придать ей повышенную эмоциональную окраску: "Сегодня вы будете выполнять работу настоящих программистов!".

Работу с Малышом можно построить по следующей схеме.

Знакомство с архитектурой Малыша и его экстерьером

Малыш сделан "прозрачным": его экстерьер отражает его внутреннее устройство.

Мы видим ячейки памяти Малыша и наблюдаем, как меняется их содержимое в процессе выполнения программы.

Монитор Малыша отображает протокол работы с пользователем: входную информацию, которая поступает

по команде **ввод**, выходную информацию, которая записывается по команде **вывод**, информационные сообщения ("старт", "конец") и сообщения об ошибках ("ошибка в программе").

Малыш имеет специальное устройство управления для записи программ (программное поле) и кнопку "Пуск" для запуска программы на выполнение.

Меню выбора алгоритма — это вспомогательное учебное средство: оно выводит в программное поле заготовку программы с пустой правой частью команды присваивания. Именно здесь детям придется программировать самостоятельно.

Устройство ввода Малыша — это вспомогательная панель, которая отображается на экране при выполнении команды **ввод**.

Устройство вывода — это монитор Малыша.

А вот устройство обработки (процессор) "спрятано" внутри. Но оно, конечно, есть. Именно оно выполняет команды программы. На этот факт нужно обратить внимание детей. Подробный анализ информационного устройства компьютера — тема урока 12.

Знакомство с языком программирования Малыша

Программа для Малыша — это список команд (вспомним определение списка). Работа программы состоит в последовательном выполнении команд, начиная с первой в списке. Программа заканчивает свою работу после выполнения последней команды.

Команд в языке всего три, и они подробно описаны в тексте Практикума.

Нужно разобрать алгоритмы выполнения каждой команды и уделить практическое внимание линейной записи арифметического выражения в команде присваивания.

Разработка программы

Разработка программы должна проводиться по схеме, предложенной в конце Читального зала:

План работы программиста.

1. Сформулировать задачу удобным для программирования образом.
2. Составить алгоритм работы исполнителя.
3. Перевести алгоритм в программу.
4. Отладить программу.

На этапе отладки обратите внимание детей на правильный подбор тестов (входных данных). Тесты должны отражать все возможные случаи: типичные наборы входных данных с разным относительным значением величин (ширина больше, меньше, равна высоте), крайние случаи (ширина, высота или обе вместе равны нулю).

Конечно, в настоящем программировании надо проверять еще реакцию программы на неверные данные (ширина, высота или обе вместе меньше нуля). Однако линейные программы Малыша не способны реагировать на неверный вход (нужна команда ветвления). Эта часть отладки (проверка защиты программы) остается пока вне поля зрения обучаемого.

Эксплуатация программы

На этом этапе можно использовать таблицы с данными, расположенные в тексте Практикума. Эти таблицы не только предлагают входные данные, но и выполняют контроль полученных результатов. Таким образом, учебная эксплуатация программы продолжает этап отладки.

Не бойтесь Малыша — он хороший!

Малыш, программируемый на языке с переменными и строчной записью арифметического выражения, вероятно, может здорово испугать учителя, привыкшего к другой последовательности обучения в школьной информатике:

1. Сначала понятие алгоритма.
2. Затем понятие исполнителя, среды, СКИ (системы команд исполнителя).
3. Понятие программы.
4. Программирование на языке без переменных.
5. И только после этого — введение понятия переменной и практика строчной записи арифметического выражения (в старших классах).

Почему же автор "обрушил" Малыша в начале курса, когда почва еще "не подготовлена"?

Постараюсь обосновать свою затею и развеять страхи, хотя, конечно, окончательный приговор будет вынесен детьми после "малышовых" уроков.

Буду весьма признателен за письма как с лестными словами в адрес исполнителя, так и с жесткой критикой этой маленькой бомбы активного действия!

Итак, вот конспект моих размышлений, которые породили идею исполнителя, вдохновили на придумывание и создание этого маленького безумства.

Оправдательная речь автора

Как вы заметили, Азы — курс очень активного обучения. Многие сложные понятия вводятся в нем с большим опережением, смысл этих понятий непрерывно уточняется, приобретает все более объемные черты.

Все это замешивается на активной деятельности ребенка, неразрывно связанной с излагаемыми темами.

В Читальных залах нет тем, которые бы не подкреплялись активной практикой.

Я всегда боялся учебников, рассчитанных на зазубривание. Тексты в них могут быть правильными или нет, но это не имеет большого значения.

Добросовестный школьник заучит тексты к уроку, разгильдяй — нет, но итог будет один: через пару дней и тот, и другой материал забудут.

Другое дело, когда текст подкрепляется активной практикой, — такое не забывается. Из абстрактной сферы понятие перемещается в бытовую и прочно закрепляется в сознании (можно часами говорить о том, что сахар сладкий, но запомнить вкус можно, только лизнув языком).

В уроке 9 говорится об алгоритмах, программах, о тонкой разнице между этими понятиями. Значит, одними словами о том, что "программа — это запись алго-

ритма на языке программирования", отделаться от школьника нельзя. Нужно дать ему почувствовать эту разницу на собственном опыте.

Итак, необходим практикум, подкрепляющий слова.

Что вдохновляет?

— Понятия: "алгоритм", "программа", "исполнитель" — были введены на первом уроке книги "Знакомство с компьютером" и обыгрывались с тех пор на всех последующих уроках.

— Школьник много раз составлял алгоритмы, знаком с понятием "хранение информации". Значит, можно перейти к составлению линейной программы, а также показать ячейку памяти компьютера, как область для хранения данных.

Что мешает?

— Новичку трудно освоить жесткий формализм записи программы. Борьба с бесконечными синтаксическими ошибками может погасить слабую искру понимания сути происходящего. Новичку может показаться, что программирование — это непрерывная борьба человека с компьютерной тупостью, которая (тупость, конечно) всегда побеждает. Впрочем, этот тезис не так далек от истины, если посмотреть на современное программное обеспечение.

К компьютерному нигилизму надо привыкать долго (и порой мучительно).

— Ужасно сложное понятие компьютерной переменной и связанное с ней присваивание $x := x + 1$.

Это просто не укладывается ни в какие рамки, даже математические.

— Для записи программ нужно владеть экранным редактированием.

— Совсем непросто записать обычную вычислительную формулу в виде арифметического выражения языков программирования.

Нельзя, правда, не отметить полезность этого навыка не только в области программирования, но в области обычных забот компьютерного пользователя.

Записывать линейно двухмерные вычислительные конструкции требуется и в электронных таблицах, и в базах данных, и в некоторых инженерных калькуляторах.

Взвесив все "за" и "против", я впал в депрессию, которая разрешилась идеей Малыша.

Малыш, на мой взгляд, решает все проблемы и дает школьнику практикум, достойный темы центрального урока этой книги.

Школьники не будут записывать программу с начала и до конца.

Малыш предложит им готовые шаблоны.

Школьники не будут вникать в трудную жизнь компьютерных переменных. Даже слова такого они не услышат. Речь пойдет о памяти для хранения чисел. Ячейки памяти обозначены буквами, но никаких заклинаний $x := x + 1$ не будет. А то, что устройства могут хранить числа, дети уже знают на примере обычного калькулятора, а также механизма, который придумал Кузя еще в уроке 1.

Школьники действительно изучили только редактор строки, правда, очень тщательно. Этот факт и прошедшая работа с Правилкой (исправление ошибок) вселяют надежду, что проблем со строчным набором, даже со специальными символами ("+", "—", "*", "/"), у школьников не будет. Малыш построен так, что для его программирования следует сделать только одну строчную запись.

Остается только одна проблема — линейная запись вычислительных формул. Известно, что в старших классах эта тема вызывает большие трудности. Но согласитесь, что перевод плоских формул в линейную запись не требует фундаментальных знаний, таких, как, например, перевод обычного выражения в польскую нотацию. (Замечу в скобках, что с последней работой маленькие дети успешно справлялись в старой Роботландии на исполнителе Плюсик.) Смею предположить, что и эту рутину младший школьник освоит быстрее старшеклассника, ибо на него еще не давят стереотипы, складывающиеся у "старичков" в выпускных классах.

Итак, Малыш с нетерпением ждет ваших детей у прогретого пульта управления!

Работа с буфером обмена

Практикум является отличным поводом для демонстрации приемов работы с окнами и буфером обмена для передачи данных из одного окна в другое.

Первые опыты с Малышом дети должны провести без всяких дополнительных сложностей, но когда они освоятся и почувствуют неудобство "ручного" переноса чисел из одного окна в другое — самое время рассказать о буфере обмена операционной системы.

Буфер обмена — это специальная область памяти ОС, служащая для временного размещения информации. Информация, помещенная в буфер обмена, становится доступна всем открытым окнам, позволяя тем самым переносить данные из одного окна в другое.

Для копирования объекта в буфер обмена нужно сначала выделить его на экране мышкой по следующему алгоритму:

1. Нажать левую кнопку мыши в начальной точке объекта.

2. Не отпуская нажатой кнопки, протягивать мышь, пока весь объект не будет отмечен.

3. Отпустить кнопку, завершая выделение.

Выделенный объект копируется в буфер обмена по клавиатурному аккорду Ctrl + C (или Ctrl + Ins).

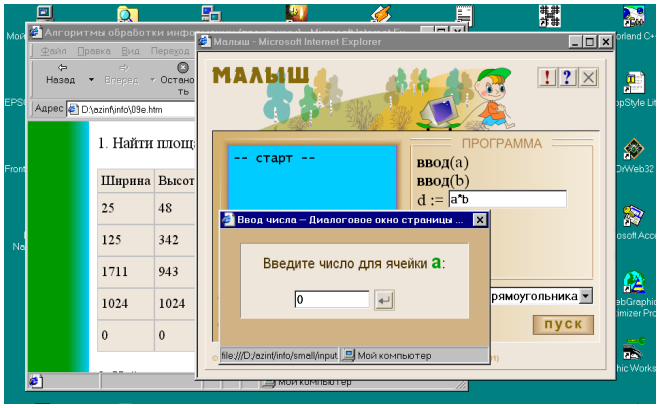
Вставка информационного объекта из буфера обмена выполняется аккордом Ctrl + V (или Shift + Ins).

Если окно имеет полоску меню с позицией "Правка", то копирование выделенного объекта в буфер обмена можно выполнить при помощи пункта "Копирование", а вставку из буфера обмена — при помощи пункта "Вставка".

Однако работа с клавиатурными аккордами гораздо удобнее.

Выполнение заданий Практикума

Откроем в Практикуме окно Малыша:



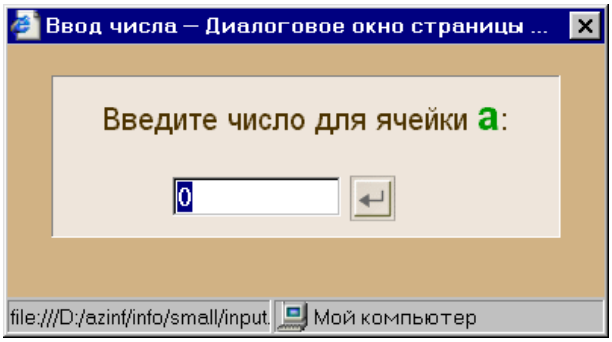
Построим работу над заданиями так:

— Выделим в окне Практикума первое входное число 25:

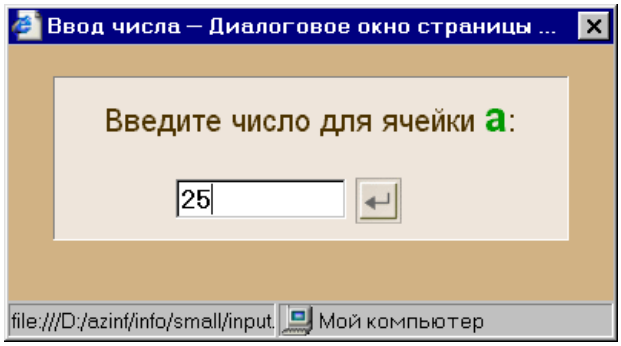
Ширина	Высота	Площадь	Проверка
25	48		Пуск
125	342		Пуск

— Занесем выделенный объект в буфер обмена (аккорд Ctrl + C).

— На панели Малыша выделим число в строке ввода:



— Скопируем число из буфера обмена (аккорд Ctrl + V) — оно заменит выделенный ноль в строке ввода Малыша:



— Аналогично выполним ввод второго числа (значение переменной b).

— Результат вычислений (число 1200) скопируем в буфер обмена.

— В таблице Практикума установим текстовый курсор в строку ввода колонки “Площадь” и выполним копирование из буфера обмена:

Ширина	Высота	Площадь	Проверка
25	48	1200	Пуск
125	342		Пуск

Теперь остается только нажать кнопку “Пуск” для проверки результата.

Ошибки ручного набора исключаются.

Описанный выше алгоритм работы особенно эффективен для больших чисел (их достаточно много в заданиях Малыша).

Ответы на вопросы

Ответ на вопрос 1. Хранение, передача и обработка.

Ответ на вопрос 2. Алгоритм.

Ответ на вопрос 3. Алгоритм — это план работы, расписанный по шагам выполнения этой работы.

Ответ на вопрос 4. Того, кто придумывает и записывает алгоритм.

Ответ на вопрос 5. Того, кто выполняет алгоритм.

Ответ на вопрос 6. Пример описан в Читальном зале.

Ответ на вопрос 7.

1) Отрежь кусок бечевки точно по длине бруска.
2) Сложи бечевку пополам.
3) Сложенную бечевку сложи пополам еще раз.
4) Приложи один конец сложенной в пункте 3 бечевки к началу бруска и отметь на бруске ее конец. Это и есть четвертая часть бруска.

Ответ на вопрос 8. Возможный вариант:

1) Введи первое число в ячейку a.
2) Введи второе число в ячейку b.
3) Введи третье число в ячейку c.
4) Сложи числа из ячеек a, b, c, а результат помести в ячейку d.
5) Выведи содержимое ячейки d на экран.

Ответ на вопрос 9. Возможный вариант:

1) Введи слово в ячейку a.
2) Найди слово из ячейки a в первом столбце словарика.
3) Запиши в ячейку b перевод из второго столбца словарика или текст “Такого слова в словаре нет”, если слово не найдено.
4) Выведи на экран содержимое ячейки b.

Ответ на вопрос 10. Возможный вариант:

1) Введи первое число в ячейку a.
2) Введи второе число в ячейку b.
3) В ячейку c помести произведение чисел из ячеек a и b.
4) В ячейку d помести результат деления числа из ячейки c на два.
5) Выведи на экран содержимое ячейки d.

Ответ на вопрос 11. Возможный вариант:
1) Введи первое слово в ячейку *a*.
2) Введи второе слово в ячейку *b*.
3) Запиши в ячейку *a* длину слова из ячейки *a*.
4) Запиши в ячейку *b* длину слова из ячейки *b*.
5) Запиши в ячейку *c* сумму чисел из ячеек *a* и *b*.
6) Выведи на экран содержимое ячейки *c*.

Ответ на вопрос 12. Программа — это алгоритм, записанный на языке программирования.

Ответ на вопрос 13. Алгоритм записывается на обычном языке (например, русском), программа — на языке программирования.

Ответ на вопрос 14. Общий ответ — в компьютерной памяти. Более подробно: компьютерные программы могут храниться на винчестере, лазерном или магнитном диске, на магнитной или бумажной ленте, на перфокарте. Перед выполнением программа записывается в оперативную память компьютера (ОЗУ).

Ответ на вопрос 15. Люди этой профессии занимаются разработкой алгоритмов и составлением компьютерных программ.

Ответ на вопрос 16. Пользователь — это человек, который работает за компьютером.

Ответ на вопрос 17. Язык программирования — это искусственный язык, созданный для записи компьютерных программ. Люди изучают языки программирования по бумажным и электронным книгам, посещая уроки в школе, институте или университете. Для того чтобы компьютер понимал язык программирования, создают специальную программу — транслятор. Транслятор выполняет перевод текста с языка программирования в команды компьютера (процессора).

Было время, когда люди программировали компьютер на его собственном языке. Однако такие программы получались слишком длинными и трудными для человеческого восприятия. Язык программирования существенно облегчил работу программиста: программа на таком языке выглядит привычно для человеческого глаза, а транслятор автоматически переводит ее в машинные коды.

Обычный человеческий язык не подходит на роль языка программирования — он слишком богат, а конструкции его — многозначны. Язык программирования — это разумный компромисс между естественным языком и языком машинных кодов. С одной стороны, программа на языке программирования близка к обычному тексту, а с другой — лишена избыточности и многозначности.

Языки программирования (их классификация, история, синтаксис и семантика) — это отдельная интересная тема для исследования. Первым языком программирования обычно называют Fortran (разработан в Америке в 1954 году). Существуют тысячи различных языков программирования. Не берусь приводить здесь сравнительную статистику популярности, просто перечислю несколько "живых" названий: C, Pascal, Basic, Java, Refal, Forth, Perl, Logo, Lisp, Prolog.

Ответ на вопрос 18.
План работы программиста:
1) Сформулировать задачу.
2) Написать алгоритм.
3) Перевести алгоритм в программу.
4) Отладить программу.

Ответ на вопрос 19. Отладка программы — это проверка ее работы на заранее подготовленном наборе входных данных. Цель отладки — поиск ошибок в программе и алгоритме и их устранение. Набор отладочных данных должен по возможности учитывать все типичные случаи, особые случаи и случаи поступления в программу неверных данных. Если данные правильные, программа должна вычислять прогнозируемый результат, если данные ошибочные — реагировать подходящим действием или информационным сообщением.

Ответ на вопрос 20. Если программа выдает заведомо неверный результат или прекращает реагировать на действия пользователя ("виснет") — значит, в ней ошибка.

Ответ на вопрос 21. Нужно подробно описать ситуацию, которая приводит к неверной работе программы или ее "зависанию", и передать свои записи программисту для проведения дополнительной отладки и исправления ошибок.

Ответ на вопрос 22.
1) Сложи 35 и 5.
2) Результат первого пункта раздели на 4.
3) Умножь 2 на 3.
4) От результата второго пункта отними результат третьего.

Ответ на вопрос 23. Кук просматривает текст и выполняет замены: букву "и" заменяет на "е", а букву "е" на "и". Действия Кука можно записать в виде такого алгоритма:
1) Посмотри на первый символ текста.
2) Пока текст не закончится, выполняй:
2.1) Если текущий символ есть буква "и", замени ее на "е".
2.2) Если результат проверки в предыдущем пункте отрицательный, проверь, не является ли текущий символ буквой "е". Если это так, замени букву на "и".
2.3) Посмотри на следующий символ текста.

Решения зачетного класса

Ответ к заданию 1.
Сан Саныч — составитель.
Петя — программист.
Вася — пользователь.
Компьютер — исполнитель.

Ответ к заданию 2. 5.
Ответ к заданию 3. 48.
Ответ к заданию 4. 20.
Ответ к заданию 5. 6.
Ответ к заданию 6. $(a/b - 2 * b)/10$.
Ответ к заданию 7. $(a * b - 2 * c - 1)/c$.
Ответ к заданию 8. $(a - b * (a - b))/c$.
Ответ к заданию 9. яи.
Ответ к заданию 10. аммаргорп.

Встроенный редактор MSOffice как пример векторного графического редактора

О.П. Шарая,
Санкт-Петербург

В школьном курсе информатики желательно познакомить учащихся как с векторными, так и с растровыми графическими редакторами. И если растровый графический редактор Paintbrush хорошо описан в разнообразной методической литературе, доступной для учителей, то про векторные редакторы этого сказать нельзя. К тому же современные векторные редакторы не всегда доступны. Неплохим решением проблемы может стать использование редактора, встроенного в пакет, ставший де-факто промышленным стандартом в России, — Microsoft Office. Он доступен, имеет исключительно подробную интерактивную справочную систему, достаточно прост и подходит для использования даже в 7-м классе.

Немного о сути векторной графики. В случае растровой графики компьютер обрабатывает и сохраняет изображение как массив пикселей, каждый из которых имеет свой цвет. В векторной графике изображение состоит из набора графических примитивов, описываемых математическими формулами. Так, для примера, окружность задается координатами центра, радиусом, толщиной линии, цветом... — всего несколькими числами. И поэтому простые векторные картинки занимают меньше места в памяти, чем растровые. Кроме того, векторные картинки можно разделить на составляющие их графические примитивы и редактировать их по отдельности.

Примером растрового редактора может служить редактор Paint, встроенный в Windows.

Примером векторного редактора может служить графический редактор, встроенный в Microsoft Office.

Покажем на примере несложной лабораторной работы специфические отличия векторного редактора от растрового. Следует отметить, что этот векторный редактор доступен во всех приложениях Microsoft Office, но удобнее всего его изучить как дополнительную возможность MS Word.

Описание работы

Запустите текстовый редактор MS Word.
Установите альбомное расположение листа.
Включите панель “Рисование”.

1. Рисование фона

- На панели “Рисование” найдем инструмент “Прямоугольник”. Названия инструментов появляются во всплывающих рамках-подсказках при наведении на них указателя мыши.
- С его помощью нарисуем прямоугольник шириной 14 см и высотой 6 см.
- Выберем инструмент “Выбор объектов”. Именно с его помощью объекты перемещаются, меняются их размеры, цвет.
- Выделим прямоугольник. Найдем инструмент “Цвет заливки”.
- Щелкнем по стрелочке справа от значка инструмента и в “Способах заливки” выберем подпункт “Градиентная заливка”.
- Выберем двухцветную заливку с синим и голубым цветами (синий — сверху) и горизонтальный режим.
- Нарисованный прямоугольник — это небо на нашей картинке.
- Точно так же нарисуем море, выбрав синий и черный цвета.
- Поместим прямоугольники один под другой.
- Щелкнем по одному прямоугольнику, затем нажмем клавишу Shift и щелкнем по второму. Это стандартный способ выделения нескольких объектов. Нажмем клавишу “Действия”, подпункт “Выровнять/распределить” и выровняем объекты по левому краю. Мы получили изображение горизонта.

2. Теперь нарисуем солнце

- Выберем инструмент “Овал”. При нажатой клавише Shift с его помощью можно нарисовать круг.
- Заливаем его с помощью градиентной заливки оранжевым и красным цветами, но используем не горизонтальный вариант, а способ “от центра”.
- Перемещаем солнце к горизонту. С помощью уже знакомой кнопки “Действие” меняем порядок расположения объектов и размещаем солнце за морем, но перед небом (см. *рис.* 1).

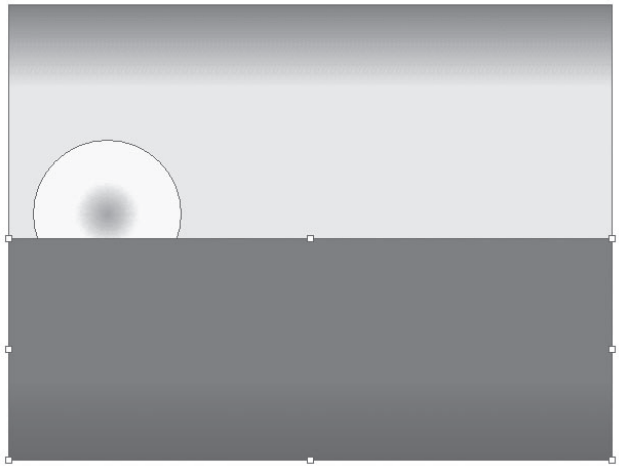


Рис. 1

3. Нарисуем отражение солнца в воде

- Скопируем солнце на свободное место. Для этого поместим на него указатель, щелкнем по правой клавише мыши и выберем вначале пункт “Копировать”, а затем “Вставить”.
- Повторим эту операцию трижды и с помощью указателя превратим три одинаковых круга в овалы разных размеров.
- Поместим их друг под другом и, выделив все одновременно с помощью клавиши Shift, выровняем по центру, используя все ту же клавишу “Действие”.
- Сгруппируем овалы и перенесем их под солнышко на поверхность моря. Если необходимо, изменим порядок расположения.

4. Приступим к рисованию кораблика

- Корпус кораблика рисуем, используя инструменты “Прямоугольник” и “Треугольник”, который выбирается в выпадающем меню кнопки “Автофигуры”.
- Далее, с помощью “Свободного вращения” поворачиваем треугольник, объединяем его с прямоугольником. Для этого надо выбрать их одновременно (Shift) и, щелкнув правой клавишей мыши, использовать пункт выпадающего меню “Группировка”.
- В этом уже новом объекте убираем черные линии контуров треугольника и прямоугольника. Для этого служит кнопка “Тип линии” — подпункты “Другие” — “Нет линий”.

Получаем примерно следующее:

Рис. 2

5. Нарисуем паруса

- Оказывается, в тех же “Автофигурах” есть очень симпатичный рисунок парусов. Только там он называется “Блок-схемы — Сохранение данных”. Поместив эти фигурки одну под другой, получим паруса.
- Можно их раскрасить, используя двухцветную градиентную заливку от синего к белому. Ее тип — угловая (см. *рис.* 3).

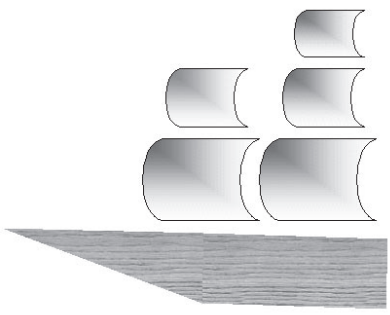


Рис. 3

- Группируем корпус судна и паруса так же, как мы делали это с треугольником и прямоугольником, рисуя корпус.

6. Нарисуем отражение кораблика в воде

- Выделим кораблик.
- Скопируем его, как мы копировали, к примеру, солнышко.
- Выберем кнопку “Действия”, подпункты “Повернуть/отразить” — “Отразить сверху вниз”.
- Осталось залить полученное отражение черным цветом.

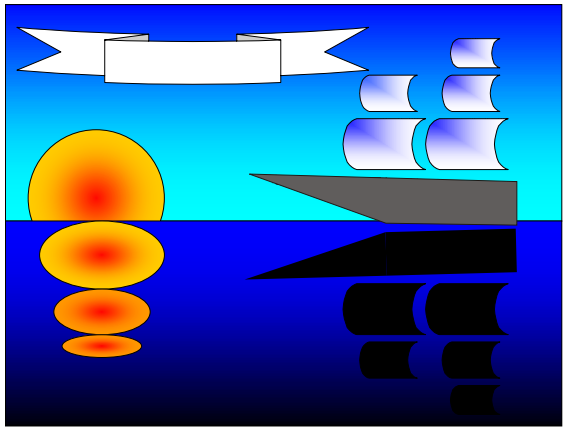
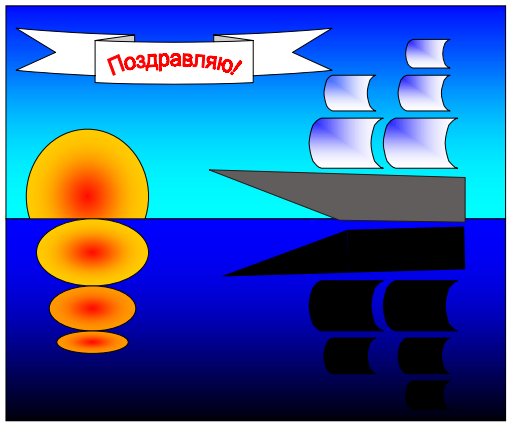


Рис. 4

7. Украсим нашу картинку надписью

- Нажмем уже известную кнопку “Автофигуры”. В подразделе “Звезды и ленты” выберем понравившуюся ленту.
- На ленте помещаем красивую надпись. Для этого нажмем кнопку “Добавить объекты WordArt” и выберем приглянувшийся тип надписи.
- В появившейся затем рамочке вводим нужный текст.
- Изменяем размеры надписи с помощью указателя и помещаем ее поверх ленты в нужном месте рисунка.
- Если вы захотите изменить цвет надписи, то не забудьте, что у букв есть заливка и контур (линия). Нажимая кнопку “Тип линии”, подпункт “Другие”, определите цвет заливки и цвет линии.

В итоге получается примерно такой рисунок.



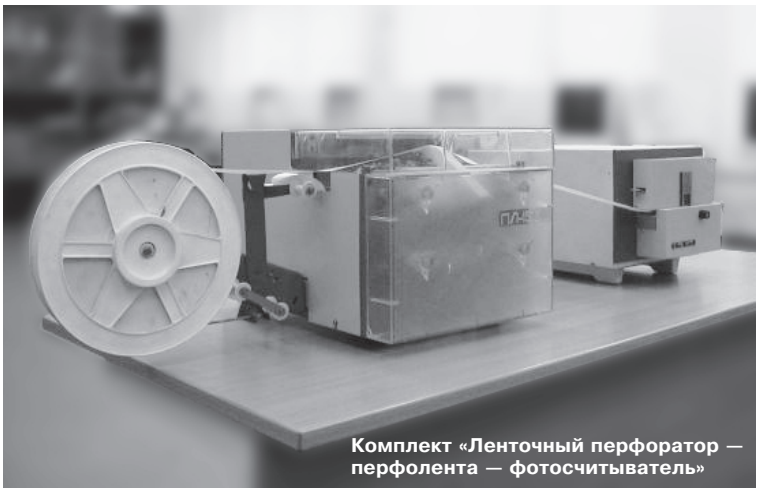
При желании лабораторная работа дополняется рисованием гранитной набережной с шарами, решетками и колоннами.

ИНФОРМАТИКА

15 апреля 2002 г. Московский городской дом учителя
ДЕНЬ УЧИТЕЛЯ ИНФОРМАТИКИ

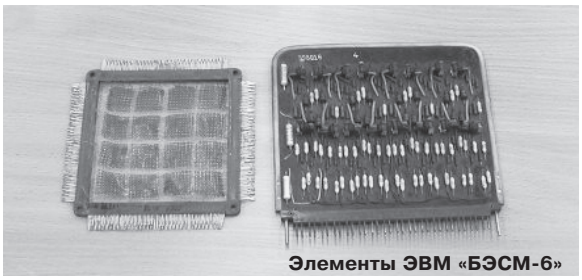
Московский педагогический МАРАФОН учебных предметов

15 апреля в Московском городском доме учителя прошел День учителя информатики. В майских номерах мы познакомим наших читателей с подробными отчетами о мероприятиях, которые прошли в рамках этого праздника. А сегодня — краткий фоторепортаж из **МУЗЕЯ ИСТОРИИ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ**, который в этот день разместился в Педагогической гостиной Дома учителя.



Комплект «Ленточный перфоратор — перфолента — фотосчитыватель»

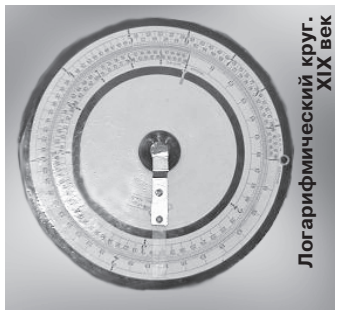
Старинные японские счеты «Соробан» (современный вариант) — 2 вида



Элементы ЭВМ «БЭСМ-6»

ЭВМ «Искра-226» с дискетами диаметром 8 дюймов

Ламповый элемент памяти (триггер) от БЭСМ-1. 1953 год выпуска



Логарифмический круг. XIX век

Постоянный адрес музея — московская гимназия № 1530 “Школа Ломоносова”. Справки о работе музея можно получить у Дмитрия Михайловича Златопольского (служебный телефон: 268-50-59, e-mail: zlato@orc.ru).

Гл. редактор
С.Л. Островский
Зам. гл. редактора
А.И. Сенокосов
Редакция:
Е.В. Андреева
Н.Л. Беленькая
Л.Н. Картвелишвили
Н.П. Медведева
Дизайн и верстка:
Н.И. Пронская
Корректоры:
Е.Л. Володина,
С.М. Подберезина

Адрес редакции и издателя:
121165, Киевская, 24
тел. 249-48-96
Отдел рекламы
тел. 249-98-70

ИНДЕКС ПОДПИСКИ
для индивидуальных подписчиков 32291
комплекта изданий 32744

Учредитель: ООО “Чистые пруды”

Зарегистрировано в Министерстве РФ по делам печати. ПИ № 77-7230 от 12.04.2001.
Отпечатано в ОИД “Медиа-Пресса”,
125993, ГСП-3, Москва, А-40, ул. “Правды”, 24.
Тираж 7000 экз.
Срок подписания в печать по графику 03.04.2002.
Номер подписан 04.04.2002.
Заказ №
Цена свободная

Тел.: (095)249-31-38, 249-33-86. Факс (095)249-31-84

Internet: inf@1september.ru
WWW: http://www.1september.ru

ИЗДАТЕЛЬСКИЙ ДОМ «ПЕРВОЕ СЕНТЯБРЯ», ГЛАВНЫЙ РЕДАКТОР — А.СОЛОВЕЙЧИК

Газеты ИЗДАТЕЛЬСКОГО ДОМА: **Первое сентября** — гл. ред. Е.Бирюкова, **Английский язык** — гл. ред. А.Громушкина, **Библиотека в школе** — гл. ред. О.Громова, **Биология** — гл. ред. Н.Иванова, **Воскресная школа** — гл. ред. монах Киприан (Ященко), **География** — гл. ред. О.Коротова, **Дошкольное образование** — гл. ред. М.Аромштам, **Здоровье детей** — гл. ред. А.Лекманов, **Информатика** — гл. ред. С.Островский, **Искусство** — гл. ред. Н.Исмаилова, **История** — гл. ред. А.Головатенко, **Литература** — гл. ред. Г.Красухин, **Математика** — гл. ред. И.Соловейчик, **Начальная школа** — гл. ред. М.Соловейчик, **Немецкий язык** — гл. ред. М.Бузоева, **Русский язык** — гл. ред. Л.Гончар, **Спорт в школе** — гл. ред. Н.Школьникова, **Управление школой** — гл. ред. А.Адамский, **Физика** — гл. ред. Н.Козлова, **Французский язык** — гл. ред. Г.Чесновицкая, **Химия** — гл. ред. О.Блохина, **Чудесная газета** — гл. ред. М.Аромштам, **Школьный психолог** — гл. ред. М.Сартан.