№ 11 (348) 16–22 марта 2002

ПОДПИСКА: (095) 249-47-58

Еженедельная газета Издательского дома «ПЕРВОЕ СЕНТЯБРЯ»

ИНФОРМАТИК

Даламбер

В 2002 году исполняется 285 лет со дня рождения выдающегося французского ученого, члена Парижской и Петербургской академий наук Жана ле Рона Даламбера (1717–1783)

Жан ле Рон Даламбер был внебрачным ребенком аристократической дамы, которого обнаружили на ступенях церкви св. Жана ле Рона в Париже (по названию этой церкви ему и дали имя) [1, 2]. Мальчику нашли кормилицу, а позже он стал жить в семье стекольщика.

С ранних лет Даламбер удивлял учителей выдающимися способностями. Свой знаменитый принцип, названный впоследствии его именем (*принцип Даламбера*), Жан ле Рон сформулировал в "Трактате о динамике", когда ему было 26 лет. "Данное правило, — писал он позже, — приводит все задачи, относящиеся к движению тел, к более простой задаче о равновесии" [3]. А во введении к "Трактату о динамике" он отмечал [1]: "В настоящем сочинении ставилась двоякая цель — расширить рамки механики и сделать подход к этой науке гладким и ровным. Больше всего я заботился о том, чтобы одна задача решалась с помощью другой, т.е. я стремился не только вывести принципы механики из наиболее ясных понятий, но и расширить область их применений. Наряду с этим я стремился показать как бесполезность многих принципов, употреблявшихся до сих пор в механике, так и выгоды, которые можно получить для прогресса этой науки от объединения остальных. Одним словом, я стремился расширить область применения принципов, сокращая в то же время их число".

Важные результаты получены Даламбером в математике [1, 2, 4]. Он предложил метод решения дифференциального уравнения второго порядка с частными производными, описывающего поперечные колебания струны (волнового уравнения). Данная работа вместе с последующими исследованиями Леонарда Эйлера и Даниила Бернулли легла в основу математической физики. Для решения некоторых дифференциальных уравнений Даламбер впервые применил функции комплексного переменного. Имя этого исследователя носит широко используемый признак сходимости рядов (*признак Даламбера*). Кроме того, он ввел понятие предела. "Математика, — говорил Даламбер, — моя самая старая и верная любовь" [1].

Жан ле Рон Даламбер был широко образованным человеком. Вместе с Дени Дидро он в 1751 году начал издавать "Энциклопедию, или Толковый словарь наук, искусств и ремесел". Это был очень большой труд. В 1751—1780 годах вышли 35 томов "Энциклопедии". Вначале были изданы 17 томов текста (60 тысяч статей) и 11 томов иллюстраций к ним. Затем — еще 4 тома текста и том иллюстраций и, наконец, 2 тома указателей.

Ученые и писатели, принимавшие участие в создании "Энциклопедии", вошли в историю как энциклопедисты. Это Вольтер, Жан-Жак Руссо, Шарль де Монтескье, Поль Анри Гольбах и другие мыслители [1, 4]. Ответственным редактором и идейным вождем "Энциклопедии" являлся Дидро. Вторым редактором был Даламбер. Ему принадлежит статья "Предварительное рассуждение", которой начинается первый том "Энциклопедии". В ней были сформулированы цели и задачи издания — рассказать об основах всех человеческих знаний, дать детальное описание отдельных наук и показать тесную связь между ними. Даламбер подготовил также вступительную статью — "Очерк происхождения и развития наук" и все статьи, касающиеся физики и математики: "Дифференциалы", "Уравнения", "Динамика", "Геометрия" — и целый ряд других.

Даламбер покровительствовал многим ученым. Так, по его предложению прусский король назначил президентом Берлинской академии наук Жозефа Луи Лагранжа. Сам Даламбер отказался занять данный пост. Не принял он и предложения русской императрицы Екатерины II быть воспитателем ее сына Павла. Даламбер говорил, что не может жить вне Франции, вне Парижа [1].

В последние годы жизни ученый занимался историей науки и написал биографии многих членов Парижской академии наук. Интересовался он также теорией музыки и подготовил работу "О свободе музыки".

Литература

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

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

3. *Гернет М.М.* Курс теоретической механики: Учебник для вузов. Изд. 5-е, испр. М.: Высшая школа, 1987.

4. Философский словарь / Под ред. И.Т. Фролова. Изд. 5-е. М.: Политиздат, 1986.

Читайте в номере

Следующий номер – олимпиадный

Наши постоянные подписчики знают, что каждый год весной мы выпускаем тематический олимпиадный номер. И в этом году, несмотря на то что и в течение года публикуется достаточно олимпиадных материалов, мы решили не нарушать традиции. Помимо очередной лекции Е.В. Андреевой (в этот раз лекция будет посвящена конечным автоматам), в олимпиадном номере будут опубликованы решения всех задач II Всероссийской командной олимпиады школьников и решения задач рыбинской городской олимпиады школьников по информатике (эта олимпиада состояла из двух туров — теоретического и практического, мы опубликуем решения всех задач обоих туров).

Мы также обращаем внимание наших читателей на то, что в рамках Дня учителя информатики пройдут целых два олимпиадных семинара. Семинар, посвященный традиционным олимпиадам, проведет Е.В. Андреева, а об уникальном опыте наших пермских коллег, которые в течение нескольких лет проводят олимпиады по базовому курсу информатики, расскажет С.В. Русаков.

День учителя информатики в Москве

15 апреля,
Московский городской дом учителя
(улица Пушечная, дом 4, строение 2,
станция метро "Кузнецкий мост")

Дорогие коллеги! Пожалуйста, постарайтесь найти возможность прийти 15 апреля в Дом учителя. Мы готовим замечательную программу, будет очень интересно.

Подробную информацию о Дне учителя информатики читайте на с. 48.

"ЖАРКОЕ ЛЕТО-2002"

Уникальный комплект замечательных материалов к новому учебному году

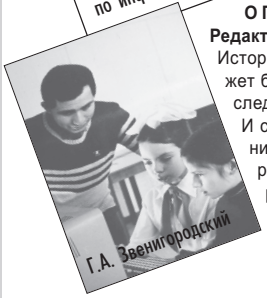
Как всегда, планируя очередное "Жаркое лето", мы учитывали пожелания наших читателей и их мнения о сериях тематических номеров прошлых лет. При этом мы стремились отдавать предпочтение материалам, которые наши подписчики смогут использовать на всем протяжении следующего учебного года. Исключение составляет, пожалуй, лишь номер, посвященный А.Г. Звенигородскому. Но мы считаем принципиально важным беречь память о тех, кто сделал для школьной информатики в России неизмеримо много. Обращаем ваше внимание на необычный номер — "Интеллектуальные игры". Не станем скрывать, что непосредственно к предмету "информатика" этот номер имеет мало отношения. Но все мы работаем в школе и знаем, что школьная жизнь не заканчивается звонком с последнего урока. Номер "Интеллектуальные игры" пригодится вам именно в этой "второй" жизни.

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. 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.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**.

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.



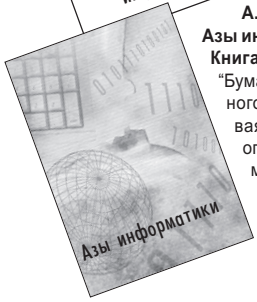
О Г.А. Звенигородском.
Редактор-составитель — Н.А. Юнерман
История становления школьной информатики не может быть написана без страниц, посвященных исследованиям и разработкам Г.А. Звенигородского. И сегодня остается актуальным воплощенное в них единство тематических и педагогических сторон работы с 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.

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".

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.



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.

Дорогие коллеги! "Информатика" распространяется только по подписке.
Подписаться на нашу газету можно по каталогу "Роспечати", индекс подписки для индивидуальных подписчиков – 32291.

Современные педагогические технологии и частные методики обучения информатике

Лекции читает И.Н. Фалина

Лекция 10. Алгоритмы сортировки и поиска

Поиск и сортировка — одни из наиболее распространенных процессов современной обработки данных. И хотя некоторые алгоритмы сортировки используют алгоритмы поиска, в данной лекции вначале будут рассмотрены алгоритмы сортировки.

Определение: *Сортировкой* называется распределение элементов множества по группам в соответствии с определенными правилами.

Например, сортировка элементов последовательности, в результате которой получается последовательность, каждый элемент которой, начиная со второго, не больше стоящего от него слева, называется *сортировкой по невозрастанию*.

В школе алгоритмы сортировки изучают "на массивах". Однако работа с типом данных *массив* достаточно сложна для школьников. Связано это с тем, на мой взгляд, что тип *массив* описывает сложную структуру данных, сложную в том смысле, что она имеет свою структуру, свою взаимосвязь между элементами. Как правило, все изучаемые до этого момента структуры были "унарны", и имя переменной соответствовало "единице" данных, при использовании же типа данных *массив* имени переменной соответствует набор "единиц" данных.

В процессе преподавания темы "Алгоритмизация" в СУНЦ МГУ выработались следующие методические приемы, которые помогают усвоению понятий *массив данных*, *тип данных "массив"*.

1) При объяснении алгоритмов сортировки (до их реализации на ЭВМ) слово "массив" вначале лучше вообще не употреблять, а вместо него использовать слово "последовательность". Например, упорядочение по алфавиту последовательности фамилий, упорядочение по возрастанию последовательности целых чисел и т.д. Все школьники уверенно работают с понятием "последовательность", тем более "конечная последовательность". Они знают, что все элементы в последовательности как-то занумерованы, индекс нумерации соответствует местоположению элемента в последовательности и т.п.

2) Перед реализацией алгоритмов сортировки на каком-либо алгоритмическом языке надо вновь вернуться к теме "Структуры данных" и рассказать, что структура данных *последовательность* реализуется в алгоритмических языках типом данных *массив*, рассказать о том, что определяет имя переменной в этом случае, как индексируется каждый элемент и т.д.

3) Слово "переменная" в данном случае лучше заменить словом "объект" (с оговоркой, что к объектно-ориентированному программированию это отношения не имеет). Эта замена существенно поднимает уровень понимаемости излагаемого материала. Предложение "рассмотрим переменную типа *массив* с именем *A*" школьники вначале не понимают и воспринимают как некий звуковой фон. Предложение "рассмотрим объект типа *массив* с именем *A*" воспринимается школьниками адекватно, так как, по-видимому, в слове "объект" уже заложена структурированность.

4) Для улучшения восприятия понятия *массив* можно использовать и такой прием. Представим массив в виде улицы с одинаковыми домами, имя массива — это название улицы, адрес каждого дома (элемента) — это имя улицы с соответствующим индексом; в каждый дом может войти "гость" (данное) строго определенного типа.

Перед объяснением темы "Алгоритмы сортировки" необходимо прорешать достаточно много "технических" задач: на поиск минимального и максимального элементов в массиве (значение и местоположение), обмен значениями элементов, сдвиг элементов массива влево/вправо и т.д.

В ситуации ограниченности времени, отводимого на изучение темы "Алгоритмизация", как всегда, эффективно применять групповую форму работы. В СУНЦ МГУ, например, в качестве групповой формы работы используется "информатический бой". О проведении информатических боев на уроках информатики будет рассказано в одной из следующих публикаций.

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

АЛГОРИТМЫ СОРТИРОВКИ

Начинать объяснение этой интересной, но достаточно сложной темы следует на доступном примере.

Пример. Сортировка колоды карт.

Постановка задачи: Имеется колода карт. На каждой карте написано одно натуральное число (все числа попарно различны). Требуется отсортировать, т.е. упорядочить колоду так, чтобы написанные на картах числа образовывали монотонно возрастающую последовательность.

Для решения этой задачи можно предложить разные алгоритмы:

1. Сортировка путем перестановки.
2. Сортировка путем выбора.
3. Сортировка путем вставки.

1. Заданная колода *X* сортируется с помощью следующих предписаний (команд исполнителя):

- если *X* содержит две неупорядоченные карты, то они меняются местами, после чего к полученной колоде применяется тот же алгоритм;
- если в *X* не встретилось ни одной неупорядоченной пары карт, то *X* отсортирована.

2. Пусть наша колода разделена на две части: *X* — неотсортированная, *Y* — отсортированная, заданная колода *X* сортируется с помощью следующих предписаний:

- если *X* пуста, то колода отсортирована;
- если *X* не пуста, то из *X* выбирается "наибольшая" карта и вставляется в начало *Y*.

3. Пусть наша колода разделена на две части: *X* — неотсортированная, *Y* — отсортированная, заданная колода *X* сортируется с помощью следующих предписаний:

- если *X* пуста, то колода отсортирована;
- если *X* не пуста, то из *X* берется любая карта и вставляется в *Y* в подходящее место так, чтобы *Y* оставалась отсортированной.

Предложите учащимся написать алгоритмы этих трех сортировок, используя указанные команды исполнителя. Заметим, что в каждом из трех алгоритмов в качестве команд исполнителя (элементарных шагов) выступают разные действия.

Предложите учащимся оценить, какой алгоритм "лучше", эффективнее, красивее и т.д. Рассмотрите с ними, какие алгоритмы работают быстро на почти упорядоченной последовательности, какие показывают лучший относительный результат на плохо упорядоченной последовательности. Именно в этом обсуждении учащихся надо заставить вспомнить про *сложность алгоритма*.

Напомним, что *сложность алгоритма* — количество действий в вычислительном процессе этого алгоритма. Обратите внимание, именно в вычислительном процессе, а не в самом алгоритме. Основным критерием выбора любого алгоритма является его сложность. Очевидно, для сравнения сложности разных алгоритмов необходимо, чтобы сложность подсчитывалась в терминах одинаковых действий.

Рассмотрим перечисленные выше алгоритмы сортировки на примере сортировки одномерного массива.

З а д а ч а . Дан одномерный массив целых чисел. Требуется отсортировать его так, чтобы все элементы были расположены в порядке неубывания ($A[i] \le A\ [i\ +\ 1]$).

I. Обменная сортировка ("пузырьковая")

1. Приведем алгоритм обменной сортировки в словесной (текстовой) форме.

Алгоритм начинается со сравнения 1-го и 2-го элементов массива. Если элементы расположены не по порядку, то они меняются местами. Этот процесс повторяется со 2-м и 3-м, 3-м и 4-м и т.д. элементами, пока пара [(N — 1)-й и N-й элемент] не будет обработана. За один такой "проход" самый большой элемент массива встанет на старшее (N-е) место. Далее алгоритм повторяется, причем на p-м "проходе" уже только первые (N — p) элементов сравниваются со своими правыми соседями. Если на очередном "проходе" перестановок не было или $N = p$, то алгоритм свою работу закончил.

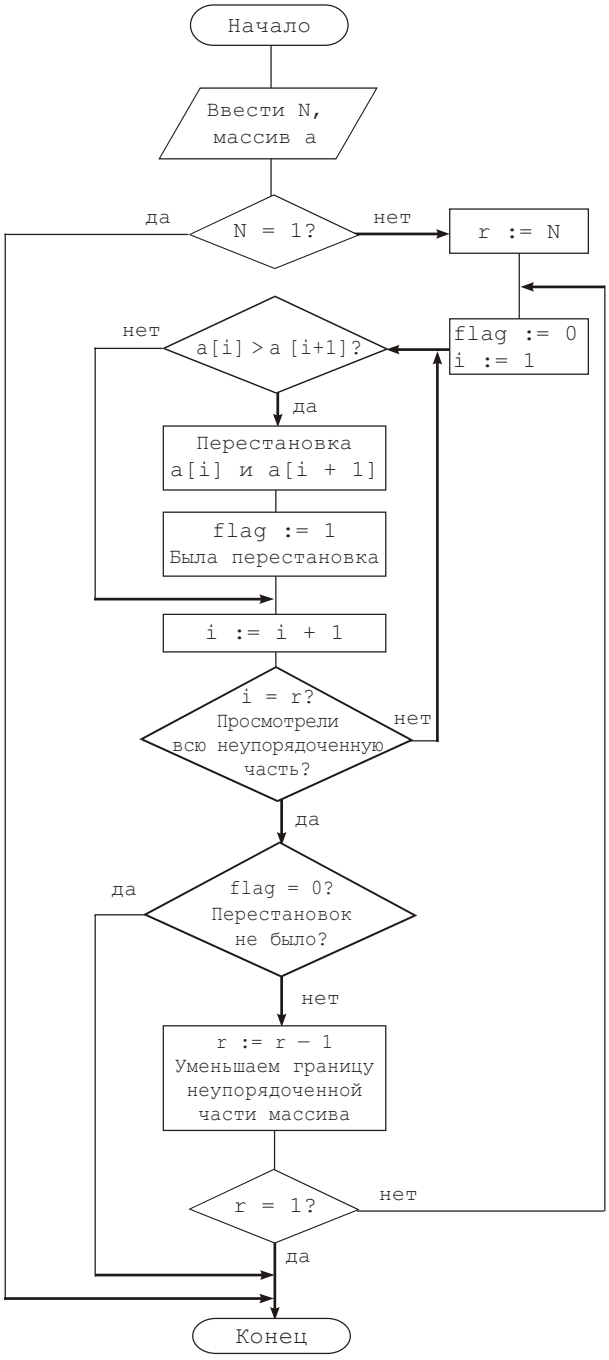
2. Приведем алгоритм обменной сортировки в виде блок-схемы.

Будем использовать следующие обозначения:

N — количество элементов в массиве;

r — граница неотсортированной части массива;

flag — переменная, содержащая признак, была ли на данном проходе сделана хотя бы одна перестановка.



3. Приведем формальное описание алгоритма обменной сортировки, записанное на алгоритмическом языке Турбо Паскаль.

```
{Сортировка "методом пузырька"}
uses crt;
const n = 10;
type myarray = array[1..n] of integer;
var i, per, r, j : integer;
```

```
    priznak : boolean;
    a : myarray;
begin
 clrscr;
 randomize;
 {инициализация массива}
 for i := 1 to n do a[i] := random(100);
 {печать массива до сортировки}
 for i := 1 to n do write (a[i0]:3);
 r := n;
 repeat
  {признак произошедшей перестановки}
  priznak := false;
  for i := 2 to r do
  begin
   if a[i - 1] > a[i] then
   begin {перестановка элементов массива}
    per := a[i - 1];
    a[i - 1] := a[i];
    a[i] := per;
    priznak := true
   end
  end;
  r := r - 1;
 until not priznak {пока "пузырек" не всплывет}
 writeln;
 {вывод отсортированного массива}
 for i := 1 to n do write(a[i]:3);
 readln
end.
```

4. Подсчитаем сложность алгоритма в единицах "количество сравнений" и "количество перестановок".

Пример I.1. Дан упорядоченный массив {1 2 3 4 5}.

После первого "прохода", сделав 4 сравнения и 0 перестановок, алгоритм закончил свою работу.

Пример I.2. Дан "случайный" массив {1 2 4 3 5}.

После 1-го "прохода", сделав 4 сравнения и 1 перестановку, получим {1 2 3 4 5}.

После второго "прохода", сделав 4 сравнения и 0 перестановок, алгоритм свою работу закончил.

Всего сделано 8 сравнений и 1 перестановка.

Пример I.3. Очевидно, что максимальное количество сравнений и перестановок будет сделано при работе с обратно упорядоченным массивом {5 4 3 2 1}.

1-й "проход" → {4 3 2 1 5}
(4 сравнения, 4 перестановки).

2-й "проход" → {3 2 1 4 5}
(3 сравнения, 3 перестановки).

3-й "проход" → {2 1 3 4 5}
(2 сравнения, 2 перестановки).

4-й "проход" → {1 2 3 4 5}
(1 сравнение, 1 перестановка).

Алгоритм свою работу закончил.

Всего $4 + 3 + 2 + 1 = 10$ сравнений и 10 перестановок.

Сортировка "методом пузырька" легко запоминается. Именно так мы сортируем предметы по высоте, ширине и т.д. Но сам по себе этот метод сортировки не используется в том виде, как было описано (алгоритм неэффективен, т.е. имеет достаточно высокую сложность): много раз приходится просматривать массив, выполнять много перестановок. На основе алгоритма сортировки "методом пузырька" можно построить много улучшенных модификаций.

1) Если на каком-либо проходе первые k пар не участвовали в обмене, то на следующем проходе просмотр можно начинать с k-го элемента.

2) За один проход сравнение можно вести с двух концов ("сжигание свечки" с двух концов).

3) Делать обмен не с соседним элементом, а просмотреть вперед и пропустить все элементы, меньшие текущего, затем выполнить сдвиг нужного участка массива на 1 влево.

II. Сортировка выбором (линейная сортировка)

1. Приведем алгоритм сортировки выбором в словесной (текстовой) форме.

Находится наибольший элемент в массиве из N элементов. Пусть он стоит на месте с номером p. Меняем его с элементом, стоящим на N-м месте, при условии, что $N \neq p$. Из оставшихся неупорядоченными $(N - 1)$ первых элементов снова выделяется наибольший элемент и меняется местами с элементом, стоящим на $(N - 1)$-м месте, и т.д. Алгоритм заканчивает свою работу, когда элементы, стоящие на 1-м и 2-м местах в неупорядоченной части массива, будут упорядочены (для этого понадобится $N - 1$ "проход" алгоритма).

2. Запишем алгоритм сортировки выбором по шагам.

Будем использовать следующие обозначения:

$a[1..N]$ — исходный массив;

i — число "проходов" (итераций) алгоритма;

r — количество элементов в неупорядоченной части массива.

❶ Положим $i = 1$, $r = N$.

❷ Находим наибольший элемент в массиве $a[1..r]$. Его место обозначим через max.

❸ Если $a[max] \neq a[r]$, то меняем местами элементы $a[max]$ и $a[r]$.

❹ Передвигаем границы упорядоченной и неупорядоченной частей массива: $r = r - 1$ $(r = N - i)$, т.е. первые $(N - i)$ элементов будут образовывать неупорядоченную часть массива. Последние i элементов образуют упорядоченную по возрастанию часть массива.

❺ $i = i + 1$. Если $i = N$, то конец работы алгоритма, иначе переход на п. ❷.

3. Приведем формальное описание алгоритма сортировки выбором, записанное на алгоритмическом языке Турбо Паскаль.

```
uses crt;
const n = 20;
type myarray = array[1..n] of integer;
var j, i, per, max, r : integer;
    a : myarray;
begin
 clrscr;
 randomize;
{инициализация массива}
 for i := 1 to n do a[i] := random(100);
 for i := 1 to n do write(a[i]:3);  {печать}
 writeln;
 for i := 1 to n - 1 do
  begin
```

```
   max := 1; r := n – i + 1;
  {поиск максимального элемента в массиве}
  for j := 2 to r do
    {запоминаем номер максимального элемента}
    if a[max] < a[j] then max := j;
   if a[max] <> a[r] then
   begin
   {ставим максимальный элемент на свое место}
    per := a[max];
    a[max] := a[r];
    a[r] := per
   end
  end;
 writeln;
 for i := 1 to n do write(a[i]:3);
 readln
end.
```

4. Подсчитаем сложность алгоритма в единицах "количество сравнений" и "количество перестановок".

Пример II.1.
Дан упорядоченный массив → {1 2 3 4 5}.
1-й "проход": 4 сравнения, 0 перестановок.
2-й "проход": 3 сравнения, 0 перестановок.
3-й "проход": 2 сравнения, 0 перестановок.
4-й "проход": 1 сравнение, 0 перестановок.
Алгоритм свою работу закончил.
Всего 4 + 3 + 2 + 1 = 10 сравнений и 0 перестановок.

Пример II.2.
Дан "случайный" массив → {1 2 4 3 5}.
После 1-го "прохода" получим → {1 2 4 3 5}
(4 сравнения, 0 перестановок).
После 2-го "прохода" получим → {1 2 3 4 5}
(3 сравнения, 1 перестановка).
После 3-го "прохода" получим → {1 2 3 4 5}
(2 сравнения, 0 перестановок).
После 4-го "прохода" получим → {1 2 3 4 5}
(1 сравнение, 0 перестановок).
Алгоритм свою работу закончил.
Всего 10 сравнений и 1 перестановка.

Пример II.3.
Дан обратно упорядоченный массив {5 4 3 2 1}.
1-й "проход" → {1 4 3 2 5}
(4 сравнения, 1 перестановка).
2-й "проход" → {1 2 3 4 5}
(3 сравнения, 1 перестановка).
3-й "проход" → {1 2 3 4 5}
(2 сравнения, 0 перестановок).
4-й "проход" → {1 2 3 4 5}
(1 сравнение, 0 перестановок).
Алгоритм свою работу закончил.
Всего 4 + 3 + 2 + 1 = 10 сравнений и 2 перестановки.

III. Сортировка вставками

Сортировка выбором и обменная сортировка относятся к сортировкам с *убывающим шагом*. Сортировка вставками построена на несколько ином принципе.

1. Приведем алгоритм сортировки вставками в словесной (текстовой) форме.

Вначале упорядочиваются два первых элемента массива. Они образуют начальное упорядоченное множество S. Далее на каждом шаге берется следующий по порядку элемент и вставляется в уже упорядоченное множество S так, чтобы слева от него все элементы были не больше, а справа — не меньше обрабатываемого. (Оптимальный вариант — место для вставки текущего элемента в упорядоченное множество S ищется методом деления пополам.) Алгоритм сортировки заканчивает свою работу, когда элемент, первоначально стоящий на N-м месте, будет вставлен на соответствующее место. (Именно таким образом игроки обычно упорядочивают свои карты.)

Сложность данного алгоритма существенно будет зависеть от того, каким способом ищется место вставки обрабатываемого элемента (алгоритм поиска). Алгоритмы поиска будут рассмотрены ниже. Но в любом случае алгоритм "Сортировки вставками" состоит из двух частей:

1) алгоритм поиска места вставки;

2) алгоритм сдвига массива на необходимое количество элементов (вправо или влево).

2. Запишем алгоритм сортировки вставками по шагам.
Будем использовать следующие обозначения:
$a[1..N]$ — данный массив;
r — количество элементов в упорядоченной части массива;
j — текущий элемент;
pos — место, на которое будем вставлять текущий элемент.

❶ Упорядочим два первых элемента.
❷ Положим $r = 2$.
❸ $j = r + 1$.
❹ Для j-го элемента находим место pos в упорядоченной части массива.
❺ Если $pos \le r$, то $a[j]$ запоминаем в b, элементы массива с pos по r сдвигаем на 1 вправо и $a[pos] = b$, иначе текущий элемент уже стоит на нужном месте.
❻ Передвигаем границу упорядоченной части массива: $r = r + 1$, т.е. первые r элементов будут образовывать упорядоченную часть массива.
❼ Если $r = N$, то конец работы алгоритма, иначе переход на п. ❸.

3. Приведем формальное описание алгоритма сортировки выбором, записанное на алгоритмическом языке Турбо Паскаль. Поиск места вставки выполняется простейшим (далеко не оптимальным) способом — методом последовательного поиска.

```
uses crt;
const n = 20;
type myarray = array[1..n] of integer;
var j, i, b, max, r, pos : integer;
    a : myarray;
begin
 clrscr;
 randomize;
 {инициализация массива}
 for i := 1 to n do a[i] := Random(100);
 for i := 1 to n do write(a[i]:3); {печать}
 writeln;
{упорядочение первых двух элементов}
 if a[1] > a[2] then
  begin
    b := a[1];
    a[1] := a[2];
```

```
    a[2] := b
  end;
 r := 2;
 {Вставка элементов в отсортированный массив}
 for i := 3 to n do
 begin
  b := a[i];       {Вставляемый элемент}
  pos := 1;
  while (b >= a[pos]) and (pos <= r) do
    inc(pos);  {Место вставки}
  for j := i - 1 downto pos do
    {Сдвиг элементов упорядоченного массива}
    a[j + 1] := a[j];
  a[pos] := b; {Вставка обрабатываемого элемента}
  r := r + 1
 end;
 for i := 1 to n do write(a[i]:3);
 readln
end.
```

4. Подсчитаем сложность алгоритма в единицах "количество сравнений" и "количество перестановок".

Пример III.1.

Дан упорядоченный массив {1 2 3 4 5}.
1-й "проход": 1 сравнение, 0 перестановок
(первые два элемента упорядочены).
2-й "проход": 2 сравнения, 0 перестановок
(третий элемент стоит на своем месте).
3-й "проход": 3 сравнения, 0 перестановок
(четвертый элемент стоит на своем месте).
4-й "проход": 4 сравнения, 0 перестановок
(пятый элемент стоит на своем месте).
Алгоритм свою работу закончил.

Всего $1 + 2 + 3 + 4 = 10$ сравнений и 0 перестановок.

Пример III.2.
Дан "случайный" массив {1 2 4 3 5}.
После 1-го "прохода" получим {1 2 4 3 5}
(1 сравнение, 0 перестановок;
первые два элемента упорядочены).
После 2-го "прохода" получим {1 2 4 3 5}
(2 сравнения, 0 перестановок;
третий элемент стоит на своем месте).
После 3-го "прохода" получим {1 2 3 4 5}
(2 сравнения, 1 перестановка;
четвертый элемент встал на свое место).
После 4-го "прохода" получим {1 2 3 4 5}
(4 сравнения, 0 перестановок;
пятый элемент стоит на своем месте).
Алгоритм свою работу закончил.

Всего 9 сравнений и 1 перестановка.

Пример III.3.
Рассмотрим обратно упорядоченный массив {5 4 3 2 1}.
1-й "проход" → {4 5 3 2 1}
(1 сравнение, 1 перестановка).
2-й "проход" → {3 4 5 2 1}
(1 сравнение, сдвиг упорядоченной
части массива на 2 позиции вправо).
3-й "проход" → {2 3 4 5 1}
(1 сравнение, сдвиг упорядоченной
части массива на 3 позиции вправо).
4-й "проход" → {1 2 3 4 5}
(1 сравнение, сдвиг упорядоченной части
массива на 4 позиции вправо).
Алгоритм свою работу закончил.

Всего $1 + 1 + 1 + 1 = 4$ сравнения
и $3 + 4 + 5 + 6 = 18$ присваиваний при сдвигах.

Ниже для каждого типа сортировок приведены максимальное и минимальное количество операций сравнения и присваиваний, выполняемых при перестановках. Для алгоритма вставками количество присваиваний подсчитано при использовании метода бинарного поиска.

Количество сравнений и количество присваиваний, выполняемых при перестановках

Обменная сортировка ("метод пузырька")

Количество сравнений:
минимальное: $N - 1$
максимальное: $N - 1 + N - 2 + N - 3 + ... + 1 = N(N - 1)/2$

Количество присваиваний:
минимальное: 0
максимальное: $3 \cdot (1 + 2 + 3 + ... + N - 1) = 3 \cdot N(N - 1)/2$

Алгоритм сортировки выбором

Количество сравнений:
в любом случае: $N - 1 + N - 2 + N - 3 + ... + 1 = N(N - 1)/2$

Количество присваиваний:
минимальное: 0
максимальное: $3 \cdot (N - 1)$

Алгоритм сортировки вставками

Количество сравнений:
в любом случае: (при реализации бинарного поиска)
$1 + 2 + [\log_2 3] + 1 + [\log_2 4] + 1 + ... +$
$+ [\log_2 (N - 1)] + 1 \approx N - 1 + \log_2 (N - 1)!$

Количество присваиваний:
минимальное: 0
максимальное: $3 \cdot (1 + 2 + ... + N - 1) = 3 \cdot N(N - 1)/2$

АЛГОРИТМЫ ПОИСКА

Задачами сортировки начали заниматься уже на заре развития ЭВМ, в то же время для разработки алгоритмов поиска было сделано сравнительно мало. Связано это было с ограниченными возможностями ЭВМ. Машины первых поколений располагали небольшой внутренней памятью и только последовательными устройствами типа лент для хранения больших объемов информации. Организовать поиск на таких ЭВМ было либо совершенно тривиально, либо почти невозможно. Это весьма показательный пример взаимозависимости развития вычислительной техники и программного обеспечения: компьютер как исполнитель (вернее, ограниченность его возможностей) тормозил разработку алгоритмов машинной обработки информации.

Но с увеличением объема памяти ЭВМ со случайным доступом привело математиков и программистов в 50-х годах к пониманию того, что поиск как таковой является

интересной задачей. Сегодня задача нахождения нужного элемента в большом объеме информации является чрезвычайно важной, и совершенствование алгоритмов поиска приобретает все большую значимость.

В алгоритмах поиска существуют две возможности окончания: либо поиск оказался *удачным*, т.е. позволил определить положение соответствующего элемента, либо он оказался *неудачным*, т.е. показал, что необходимого элемента в данном объеме информации нет. Хотя целью поиска является информация, алгоритмы поиска в случае удачного окончания выдают местоположение искомого элемента, например, номер элемента в массиве.

Во многих программах поиск требует наибольших временных затрат, так что замена плохого метода поиска на хороший часто ведет к существенному увеличению скорости работы программы.

Между поиском и сортировкой существует определенная взаимосвязь. Например: рассмотрим следующую задачу: "Даны два множества чисел $A = \{a_1, a_2, ..., a_m\}$ и $B = \{b_1, b_2, ..., b_n\}$. Требуется определить, является ли A подмножеством B".

Напрашиваются, как минимум, два решения:

1. Сравнивать каждое a_i последовательно со всеми b_j до установления совпадения.

2. Упорядочить A и B, затем совершить один последовательный проход по обоим множествам, проверяя соответствующие условия.

Каждое из этих решений имеет свои привлекательные стороны для различных диапазонов значений m и n. Решение 1 хорошо при очень малых m и n, а при возрастании m и n лучшим будет решение 2.

Алгоритм последовательного поиска в неупорядоченном массиве

Несмотря на простоту алгоритма последовательного поиска, на задачах с его использованием можно исследовать ряд очень интересных идей.

Сформулируем более точно алгоритм последовательного поиска в неупорядоченном массиве (очевидно, что этот алгоритм можно применять и для поиска в упорядоченном массиве).

Имеется массив $a[1..N]$, требуется найти элемент массива, равный P.

❶ Установить $i = 1$.

❷ Если $a[i] = P$, алгоритм оканчивается удачно.

❸ Увеличить i на 1.

❹ Если $i \le N$, то переход на шаг ❷. В противном случае алгоритм заканчивается неудачно.

Сложность алгоритмов поиска естественно оценивать по числу сравнений. Так для поиска максимального или минимального элемента в неупорядоченном массиве потребуется $N - 1$ сравнение. Для поиска максимума и минимума одновременно (одна из возможных интересных задач) требуется $3 \cdot (N$ **div** $2) - 2$ операций сравнения для четных N и $3 \cdot (N$ **div** $2)$ операций сравнения для нечетных N.

Алгоритм поиска в упорядоченном массиве

Если необходимо найти элемент в большом массиве, то о последовательном поиске почти не может быть речи. Наиболее распространенный алгоритм поиска в упорядоченном массиве имеет название *бинарный поиск*, его иногда называют *логарифмическим поиском*, или *методом деления пополам*.

Основная идея бинарного поиска довольно проста, детали же нетривиальны, и для многих хороших программистов не одна попытка написать правильную программу закончилась неудачей. Одна из наиболее популярных реализаций этого алгоритма использует две переменных-указателя (u и l), соответствующих верхней и нижней границам поиска.

1. Запишем алгоритм бинарного поиска по шагам.

С помощью этого алгоритма ищется элемент k в упорядоченной по возрастанию последовательности a, содержащей N элементов.

❶ Начальная установка границ поиска: $l = 1$, $r = N$.

❷ Если $r < l$, то алгоритм заканчивается неудачно. В противном случае делим последовательность, в которой ведется поиск, на две части (пополам). Для этого находим середину последовательности: $i = (l + r)$ **div** 2 (i указывает примерно в середину рассматриваемой части последовательности).

❸ Если $k < a_i$ (искомый элемент, если он есть, находится в левой части последовательности), то перейти на п. ❹, если $k > a_i$ (искомый элемент, если он есть, находится в правой части последовательности), то перейти на п. ❺, если $k = a_i$, алгоритм заканчивается удачно.

❹ Изменяем верхнюю границу поиска: $r = i - 1$ (будем рассматривать только левую часть последовательности) и переход на п. ❷.

❺ Изменяем нижнюю границу поиска: $l = i + 1$ (будем рассматривать только правую часть последовательности) и переход на п. ❷.

Для этого алгоритма число сравнений при удачном поиске приближенно равно $\log_2 N - 1$.

2. Приведем формальное описание алгоритма бинарного поиска, записанное на алгоритмическом языке Турбо Паскаль.

```
var a:array [1..N] of integer;
. . .
L := 1; R := N;
found := false;
while (L <= R) and (not found) do
begin
  i := (L + R) div 2;
  found := a[i] = k;
  if a[i] < k then L := i + 1
             else R := i - 1
end;
if found then ...
```

Более подробно об алгоритмах сортировки и поиска можно прочитать в [2, 3].

Литература

1. Информатика. Задачник-практикум в 2 т. / Под ред. И.Г. Семакина, Е.К. Хеннера. М.: Лаборатория базовых знаний, 2001.

2. *Андреева Е.В., Котик У.В., Фалина И.Н., Чернокожин Е.В.* Информатика (Пособие для поступающих в вузы). М.: Диалог-МГУ, 2000.

3. *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, 10/2002

Билет № 5

1. Операционная система компьютера (назначение, состав, способ организации диалога с пользователем). Загрузка компьютера.

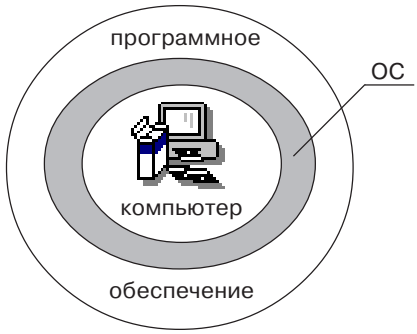
2. Создание, преобразование, сохранение, распечатка рисунка в среде графического редактора.

* * *

1. Операционная система компьютера (назначение, состав, способ организации диалога с пользователем). Загрузка компьютера.

Операционная система — это важнейшая часть системного программного обеспечения, которая организует процесс выполнения задач на ЭВМ, распределяя для этого ресурсы машины, управляя работой всех ее устройств и взаимодействием с пользователем. Иными словами, это своеобразный администратор компьютера, распределяющий его ресурсы так, чтобы пользователь мог решать свои задачи максимально удобно.

Примечание. Ресурсами компьютера являются процессорное время, память всех видов, устройства ввода/вывода, программы и данные.



Роль операционной системы можно наглядно представить себе с помощью следующего рисунка. В центре его изображен собственно компьютер, т.е. все то оборудование, которое стоит на вашем столе и которое можно непосредственно "потрогать руками" (в информатике эта часть часто называется **hardware**). Внешней оболочкой является разнообразное программное обеспечение (**software**), позволяющее многочисленным пользователям решать свои прикладные задачи из всех областей человеческой деятельности. ОС организует их совместную работу и служит своеобразным программным расширением управляющего устройства компьютера. Вы можете спросить: а так ли нужен еще один дополнительный слой? *Очень нужен*, учитывая тот факт, что невозможно заложить в центральный блок информацию обо всех устройствах, которые к нему могут быть подсоединены. И, кроме того, новое устройство может быть изобретено уже после изготовления компьютера! Отсюда очевидно, что загружаемая (а следовательно, изменяемая) программная часть, обеспечивающая работу компьютерной аппаратуры, совершенно необходима.

С другой стороны, наличие операционной системы очень существенно облегчает разработку нового программного обеспечения. Все наиболее часто встречающиеся при работе компьютера задачи сконцентрированы в ОС. Поэтому программисту уже не требуется заботиться о размещении своей программы в объеме памяти каждого конкретного компьютера или описывать отдельные технические детали взаимодействия со всевозможными внешними устройствами разнообразных фирм-изготовителей — для этого достаточно просто обратиться к соответствующей функции операционной системы. Приведем простой частный пример. Если бы об этом не заботилась ОС, каждая программа должна была бы самостоятельно проверять наличие дискеты в дисководе при записи информации или факт подключения принтера перед печатью на бумагу. И таких ситуаций существует великое множество.

Но наличие операционной системы удобно и пользователю. Поскольку на современных компьютерах диалог с ним ведется именно средствами ОС, то интерфейс (проще говоря, способы взаимодействия с человеком) во всех программах получается примерно одинаковым. Так, освоив 2—3 программы в системе Windows, пользователь может довольно быстро научиться работать с еще одной, даже совершенно новой для него.

Таким образом, мы видим, что операционная система решает целый комплекс важных задач управления компьютером. Сформулируем их по возможности более полно. Итак, ОС современного компьютера выполняет следующие функции.

- Организация согласованного выполнения всех процессов в компьютере. Планирование работ, распределение ресурсов.
- Организация обмена с внешними устройствами. Хранение информации и обеспечение доступа к ней, предоставление справок.
- Запуск и контроль прохождения задач пользователя.
- Реакция на ошибки и аварийные ситуации. Контроль за нормальным функционированием оборудования.
- Обеспечение возможности доступа к стандартным системным средствам (программам, драйверам, информации о конфигурации и т.п.).
- Обеспечение общения с пользователем.
- Сохранение конфиденциальности информации в многопользовательских системах.

Первые операционные системы (CP/M, MS-DOS, Unix) вели диалог с пользователем на экране текстового дисплея. Это был в полном смысле слова диалог, в ходе которого человек и компьютер по очереди обменивались сообщениями: человек вводил очередную команду, а компьютер, проверив ее, либо выполнял, либо отвергал по причине ошибки. Такие системы в литературе принято называть ОС **с интерфейсом командной строки**. Типичный пример возможного фрагмента сеанса работы приведен на следующем рисунке.

Пользователь последовательно набрал две команды вывода каталога дисков, причем первую компьютер выполнил нормально, и на экране появился требуемый список файлов, а вторую "отказался" делать, поскольку оператор ошибочно указал имя несуществующего диска. Очевидно, что подобный способ общения не очень удобен для человека, поскольку требует постоянно держать в голове жесткий синтаксис всех допустимых команд и очень внимательно их вводить. Поэтому почти сразу же стали появляться сервисные системные программы, тем или иным способом облегчающие работу с ОС. Наиболее ярким примером таких программ-оболочек может служить широко известный Norton Commander, который был настолько распространен, что многие пользователи искренне считали его частью операционной системы.

Развитие графических возможностей дисплеев привело к коренному изменению принципов взаимодействия человека и компьютера. Командная строка была безвозвратно вытеснена **графическим интерфейсом**, когда объекты манипуляций в ОС изображаются в виде небольших рисунков, а необходимые действия тем или иным образом выбираются из предлагаемого машиной списка — так называемого меню. При подобном методе диалога набор текста полностью отсутствует и вполне достаточно всего нескольких клавиш. Существенным дополнением к графическому способу ведения диалога явилось появление нового устройства ввода информации в компьютер — манипулятора "мышь", без которого сейчас просто невозможно представить современный компьютер. Примерами операционной системы с графическим интерфейсом служат довольно похожие ОС для компьютеров "Macintosh" (не имеет специального названия и обозначается просто System с номером версии) и "IBM PC" — OS/2 и Windows. Последняя система в нашей стране распространена необычайно широко.

Перейдем теперь к описанию состава операционных систем. Он, конечно, может быть довольно разным для различных систем. Так, для "классических" ОС с командной строкой довольно четко выделяются три основные части:

- машинно-зависимая часть для работы с конкретными видами оборудования;
- базовая часть (ядро), не зависящая от конкретных деталей устройств: она работает с абстрактными логическими устройствами и при необходимости вызывает функции из предыдущей части; отвечает за наиболее общие принципы устройства ОС;
- программа ведения диалога с пользователем (ее часто называют **командным процессором**).

Значительная часть операционной системы находится в памяти постоянно, что обеспечивает ее эффективную работу. Программы для некоторых редко используемых операций типа форматирования дискет чаще всего оформляются в виде самостоятельных служебных программ и хранятся на внешних носителях. Такие программы обычно называют **утилитами**. Кроме того, в ОС, как правило, включают небольшой стандартный набор самого необходимого программного обеспечения, например, простейший текстовый редактор.

Состав операционных систем с графическим интерфейсом типа Windows заметно шире, но в целом имеет похожее строение.

В момент включения компьютера в ОЗУ нет осмысленной информации. Поэтому особый интерес представляет вопрос о том, как операционная система загружается. Процесс этот в заметно упрощенном виде выглядит так. При включении компьютера (или при нажатии кнопки сброса) счетчик процессора аппаратно устанавливается на начальный адрес ПЗУ, и стартует выполнение программы начальной загрузки. Прежде всего ищется и тестируется установленное оборудование. Современные компьютеры в основном используют внешние устройства "**plug and play**" (переводится — "включил и работай"), поэтому они способны сообщить процессору свои основные характеристики и условия работы. Опрос внешних устройств и проверка их работоспособности занимают достаточно длительное время, несмотря на высокое быстродействие компьютера. В случае если все оборудование функционирует нормально, происходит переход к следующему этапу — поиску начального загрузчика операционной системы. Он может находиться на жестком диске, на дискете, на CD-ROM и даже быть получен с помощью сетевой платы. Поэтому компьютер опрашивает перечисленные устройства по очереди, в определенном порядке, до тех пор, пока не обнаружит требуемую информацию (в скобках заметим, что порядок поиска при наличии достаточных навыков и знаний может быть легко изменен). Итак, загрузчик, представляющий собой не что иное, как *программу дальнейшей загрузки*, обнаружен и прочитан в память. Дальнейшие действия машины уже определяются тем, что введено извне. Поскольку начальный загрузчик очень мал, то он умеет очень немного — найти и прочесть первый файл ОС с фиксированным именем и передать ему управление. И только после этого будет загружена в ОЗУ остальная часть операционной системы и машина сможет, наконец, нормально общаться с пользователем.

Примечание. Несколько слов для тех, кого удивила сложность описанного процесса. Почему загрузка ОС такая многоступенчатая и почему, например, нельзя просто записать на-

чальный загрузчик в то же самое ПЗУ? Технически это не представляет никакого труда, но все дело в том, что тогда мы сможем пользоваться *только одной(!)* операционной системой, а именно той, загрузчик для которой жестко "зашили" в ПЗУ.

И в заключение еще одно дополнительное замечание. Может быть, не стоит требовать этот материал с учеников, но рассказать об этом, по-моему, стоит. Всегда ли существовала ОС и может ли компьютер работать без нее? Как ни странно, ответ на оба вопроса отрицательный. Операционная система существовала *не всегда*, а возникла на стыке второго и третьего поколений.

Поколение	Количество пользователей	Количество задач	Система управления	Внешние устройства	Средства диалога
1	1	1	нет	п/карты, п/лента, ЦПУ	машинные коды
2	1	1	транслятор с 1—2 языков	м/лента, АЦПУ	+ языки высокого уровня
3	много	много	**ОС**	**м/диск**, дисплей, графопостроитель	+ язык управления заданиями
4 ПК	1	много	ОС	большое разнообразие устройств	+ пользовательский интерфейс
сервер	много	много			

Как видно из приведенной таблицы, существенными причинами возникновения ОС являются появление сложных внешних устройств — в первую очередь магнитных дисков, и необходимость разделения ресурсов между задачами и пользователями. Что касается работы без ОС, то теоретически можно написать такую программу, которая будет сама загружаться и работать с внешними устройствами без всякого участия ОС. На практике это чересчур сложно и никогда не делается. Даже если компьютер целыми днями работает по единственной программе (кассовый аппарат в магазине или учет переводов в сберкассе), в нем все равно обычно используется операционная система.

Литература

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

2. *Брябрин В.М.* Программное обеспечение персональных ЭВМ. М.: Наука, 1990, 272 с.

3. *Смирнов Н.Н.* Программные средства персональных ЭВМ. Л.: Машиностроение, 1990, 272 с.

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

5. *Угринович Н.* Информатика и информационные технологии. Учебное пособие для общеобразовательных учреждений. М.: БИНОМ, 2001, 464 с.

6. *Гук М.* Аппаратные средства IBM PC. Энциклопедия. СПб.: Издательство "Питер", 2000, 816 с.

* * *

2. Создание, преобразование, сохранение, распечатка рисунка в среде графического редактора.

Один из многочисленных вариантов задания может выглядеть следующим образом.

Задание

С помощью графического редактора создать, сохранить в файл с указанным учителем именем и в заданный каталог, а также распечатать изображение, представленное на рисунке.

Сохранить уменьшенную в 4 раза копию рисунка в файл с другим именем в тот же самый каталог.

Комментарии к заданию.

При создании данного рисунка проверяются следующие навыки работы с графическим редактором:

- умение рисовать графические примитивы (линия, прямоугольник, окружность);
- рисование линий разной ширины (контуры домов и границы дороги);
- использование различных инструментов: заливка (дорога) и распылитель (небо);
- выбор цветов с помощью палитры;
- нанесение надписей на рисунок;
- работа с фрагментами рисунка: выделение, копирование, перенос;
- масштабирование изображения.

В ходе выполнения задания ученик также должен продемонстрировать умение работать с файловой системой и принтером.

При оценке ответа следует не просто смотреть на предъявленный рисунок, но и обязательно просить ученика продемонстрировать, как он выполнил то или иное действие. Дело в том, что совсем не обязательно он строил рисунок рационально, например, вместо рисования прямоугольника он мог строить 4 линии, а вместо построения жирной линии — проводить линию обычной ширины несколько раз и т.д.

Особое внимание, по нашему мнению, следует уделить алгоритму построения изображения. Он тоже должен быть рациональным. Например, сначала рисуется закрашенный прямоугольник окна, затем он копируется дважды. Затем полученные 3 окна можно копировать еще раз, и сразу получается весь этаж. Тиражируя его по подобной схеме, получаем один дом, который затем также копируем. Аналогично можно поступать и при рисовании деревьев.

Билет № 6

1. Файловая система компьютера. Папки. Файлы (имя, тип, путь доступа). Операции с файлами и папками в среде операционной системы.

2. Решение задачи на построение графика функции в электронных таблицах.

* * *

1. Файловая система компьютера. Папки. Файлы (имя, тип, путь доступа). Операции с файлами и папками в среде операционной системы.

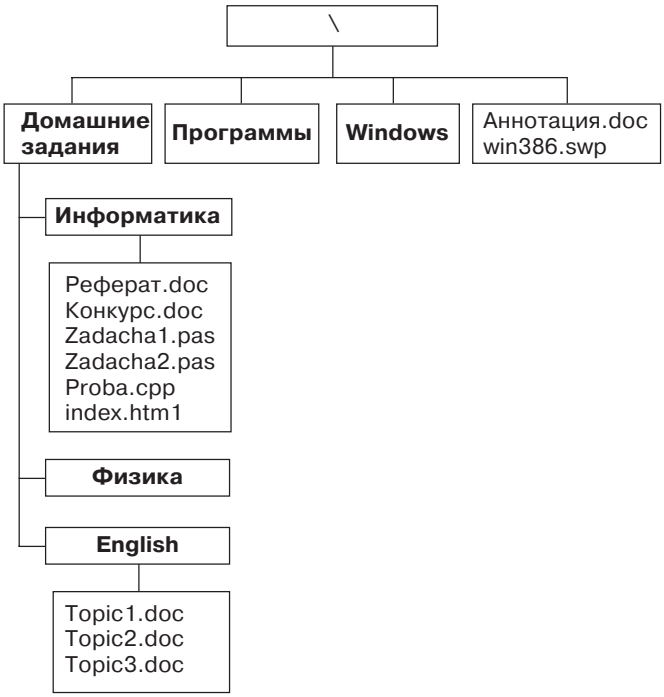
Главное назначение носителей внешней памяти — долгосрочное хранение информации. Любая информация (текст, изображение, программа, видеофильм и т.д.) на внешнем носителе хранится в виде *файла*. *Файл* (*file*) — это поименованная область на диске, в которой хранится отдельный экземпляр информации определенного типа.

Файл характеризуется набором *параметров* (имя, расширение, размер, дата создания, дата последней модификации) и *атрибутами*, используемыми операционной системой для его обработки ("архивный", "системный", "скрытый", "только для чтения", "каталог" и др.).

Файловая структура может быть *одноуровневой* — это простая последовательность файлов. *Многоуровневая файловая структура* — древовидный способ организации файлов на диске. При этом существуют специальные файлы, которые в одних операционных системах называют *каталогами* (*directory*) (в других — *папками*), назначение которых — регистрация в них файлов (в том числе и других каталогов). Наличие поддержки каталогов в операционной системе позволяет выстроить иерархическую (многоуровневую) организацию размещения файлов на носителе. В этом случае файлы, имеющие одинаковую природу (файлы операционной системы, документы, офисные программы, игровые программы, результаты расчетов, домашние задания, рисунки и т.д.), размещаются в отдельных каталогах. Такая структура хранения информации позволяет уверенно ориентироваться в принадлежности той или иной информации, особенно если учесть, что на современных носителях информации могут храниться тысячи, а то и десятки тысяч файлов! Работа с информацией была бы значительно затруднена, если бы она была беспорядочно размещена на носителе.

Любой носитель изначально имеет один каталог, который создается операционной системой без нашего участия, — *корневой*. Корневой каталог на каждом носителе внешней памяти существует в единственном экземпляре. Все другие каталоги создаются либо пользователем, либо могут быть автоматически созданы программами.

На рисунке приведен пример иерархической структуры размещения информации на носителе ("\" обозначает корневой каталог, имена каталогов выделены полужирным шрифтом, файлов — обычным).



Файлы и каталоги, зарегистрированные в одном каталоге, **должны иметь уникальные имена**. Файлы (или каталоги), зарегистрированные на одном и том же носителе информации, но в разных каталогах, могут иметь совпадающие имена.

Полное имя файла однозначно определяет местоположение любого файла на носителе. Оно состоит из *пути к файлу*, включающему *логическое имя устройства* и иерархическую систему каталогов, от корневого каталога до того, в котором содержится файл, и *собственно имени файла* и *расширения*.

Правила задания имени файла определяются операционной системой и используемой файловой системой. Вообще *файловая система* определяет общую структуру именования, хранения и организации файлов в операционной системе. *Файловая система FAT* (***F**ile **A**llocation **T**able*) поддерживается операционными системами DOS и Windows (в DOS — FAT16; в Windows9x — FAT16 и FAT32). Это файловая система, основанная на таблице размещения файлов, которая поддерживается операционной системой для отслеживания состояния различных сегментов дискового пространства, используемого для хранения файлов. *NTFS* (Windows ***NT File System***) — файловая система операционных систем Windows NT и Windows 2000. Улучшенная по сравнению с FAT файловая система, разработанная для использования специально с ОС Windows NT. Она выполняет те же функции, что и FAT, но, кроме того, поддерживает средства восстановления файловой системы и допускает использование чрезвычайно больших носителей данных. Также поддерживает объектно-ориентированные приложения, обрабатывая все файлы как объекты с определяемыми пользователем и системой атрибутами. Каждый файл на томе NTFS представлен записью в специальном файле, называемом "главной файловой таблицей" (MFA).

В операционных системах семейства DOS имя файла может содержать от 1 до 8 символов, можно использовать символы латинского алфавита, арабские цифры и некоторые другие символы; есть ряд символов, использование которых в имени запрещено. В операционных системах семейства Windows имя может содержать уже от 1 до 255 символов, причем набор символов, из которых можно составлять имена файлов, расширяется. В частности,

можно использовать буквы национальных алфавитов, пробелы и т.д. Windows, как правопреемница DOS, обеспечивает совместимость собственных "длинных" имен с короткими именами DOS, т.е. у файла Windows есть дополнительный атрибут — имя этого файла в DOS. Строчные и прописные буквы в именах файлов не различаются. По-другому дело обстоит в операционных системах семейства Unix. Там строчная и прописная буквы различаются, поэтому имена, записанные одними и теми же буквами, но имеющие различия в регистрах, будут разными.

Расширение имени файла записывается после точки и может содержать от 1 до 3 символов в DOS и больше 3 — в Windows. Чаще всего в расширение вкладывается определенный смысл (хотя пользователь может задавать и бессмысленные расширения) — оно указывает на содержимое файла или на то, какой программой был создан данный файл. Например, DOC, TXT — расширения текстовых файлов, COM, EXE — исполнимых файлов, INI — инициализационных файлов, PAS, BAS, CPP — исходные тексты программ на соответствующем языке программирования, и т.д. В операционной системе Windows именно по расширению файлы ассоциируются с определенной программой, с помощью которой они могут быть открыты для просмотра или модификации.

Примеры имен файлов:

```
a:\mydir\f1.txt
c:\windows\temp\abcd.tmp
myfile.doc
```

Размер файла измеряется в байтах.

В зависимости от значений *атрибутов файлов* операционная система разрешает или запрещает те или иные действия над файлами.

Обычно в Windows по отношению к файлам и каталогам используют несколько иную терминологию. Познакомимся и с ней по материалам публикации [1, дополнительная литература].

Наиболее простыми являются ***документы*** и ***программы***. Документы — это объекты, содержащие ту или иную информацию: тексты, картинки, звуки и т.д. Развитие мультимедийных возможностей компьютера приводит к тому, что в некоторых документах могут содержаться несколько видов информации одновременно, например, движущееся изображение и звук. Программы служат для обработки документов — это своеобразные инструменты воздействия на документы. Часто их еще называют *приложениями*, например, приложение MS-DOS или приложение Windows. Между отдельными программами и документами существует устойчивая связь: текстовый редактор работает с текстовыми документами, программа-фонограф воспроизводит звуки и т.п. Windows запоминает такие связи и способна самостоятельно их использовать при просмотре и работе с документами.

Группа однотипных документов, а также программы для их обработки могут быть помещены в общую ***папку***. Папка является еще одним, более крупным объектом Windows. В отличие от документов и программ, являющихся простыми и "неделимыми" объектами, папка может содержать другие объекты, в том числе и новые папки; в частном случае папка может быть пустой.

Независимо от операционных систем персональных компьютеров все файлы можно разделить на *текстовые* и *бинарные* (по-другому — *двоичные*) файлы. Текстовыми называют файлы, в которых используются в качестве информационных символы с десятичными кодами 32-126 и 128-254. Двоичные файлы представляют собой последовательность из любых символов. Их длина определяется из заголовка файла. Это разделение является важным для различных операционных систем, поскольку назначение и обработка бинарных и текстовых файлов в операционных системах различаются.

Также файлы можно разделить на *исполняемые* (программы) и *неисполняемые* (*файлы данных* и документов). Исполняемые файлы могут запускаться операционной системой на выполнение, а неисполняемые файлы могут только изменять свое содержимое в процессе выполнения программ. Далее можно разделить файлы на *основные*, присутствие которых обязательно для работы операционной системы и программных продуктов, *служебные*, хранящие конфигурацию и настройки основных файлов, *рабочие*, содержимое которых изменяется в результате работы основных программных файлов и собственно ради которых и создаются все остальные файлы, а также *временные файлы*, создающиеся в момент работы основных и хранящие промежуточные результаты.

В процессе работы над файлами и каталогами (далее они называются объектами) производят следующие операции:

- *создание* (в текущем каталоге создается новый экземпляр объекта, ему дается имя. Созданный объект при этом может быть и пустым);
- *копирование* (копия объекта создается в другом каталоге или на другом носителе);
- *перемещение* (производится копирование объекта в другой каталог или на другой носитель, в исходном каталоге объект уничтожается);
- *удаление* (в исходном каталоге объект уничтожается);
- *переименование* (изменяется имя объекта).

В ОС DOS, Unix эти операции выполняются подачей из командной строки специальных команд. В семействе ОС Windows для этих целей служит специальная служебная программа ***Проводник*** (***Explorer***). Кроме того, графический интерфейс позволяет осуществлять эти же операции и другими способами, например, с использованием контекстного меню. Большинство пользователей всех ОС, включая графические, пред-

почитают применять при работе с файлами специальные программы-оболочки. У отечественного пользователя DOS наибольшей популярностью пользовалась программа-оболочка Norton Commander, у пользователей Windows — Far, Windows Commander и др.

Основная литература

1. *Гейн А.Г., Сенокосов А.И., Шолохович В.Ф.* Информатика. 7—9-е классы. Учебник для общеобразовательных учебных заведений. М.: Дрофа, 1998.

2. *Каймин В.А., Щеголев А.Г., Ерохина Е.А., Федюшин Д.П.* Основы информатики и вычислительной техники: Пробное учебное пособие 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.

5. *Угринович Н.* Информатика и информационные технологии. Учебное пособие для общеобразовательных учреждений. М.: БИНОМ, 2001, 464 с. (§ 1.4. Файлы и файловая система, с. 32—35).

6. Информатика. 7—8-е классы / Под ред. Н.В. Макаровой. СПб.: ПитерКом, 1999, 368 с.

Дополнительная литература

Еремин Е.А. Windows 95. Информатика и образование, 1997, № 4, с. 37—41; № 5, с. 88—96.

* * *

2. Решение задачи на построение графика функции в электронных таблицах.

Даны функция $y = f(x)$ и отрезок $[a, b]$. Построить график этой функции на заданном отрезке, используя табличный процессор.

Пусть $f(x) = x \cdot \cos(x)$; $a = -10$; $b = 10$.

Для решения задачи воспользуемся ЭТ MS Excel.

Решение состоит из двух шагов:

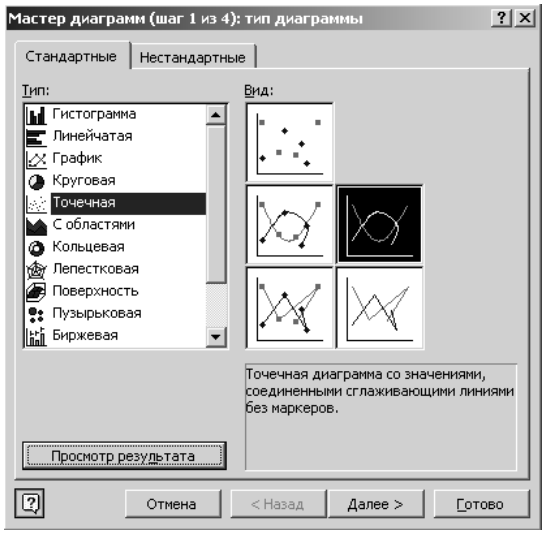
1) протабулировать заданную функцию на заданном отрезке, т.е. вычислить ее значения с заданным шагом.

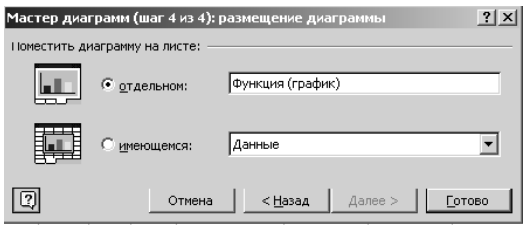
Занесем начало и конец отрезка в отдельные ячейки, чтобы при необходимости можно было изменить начало и конец отрезка. В один из столбцов поместим значения аргумента, в другой — значения функции. Ниже приведено начало таблицы в режиме отображения формул.

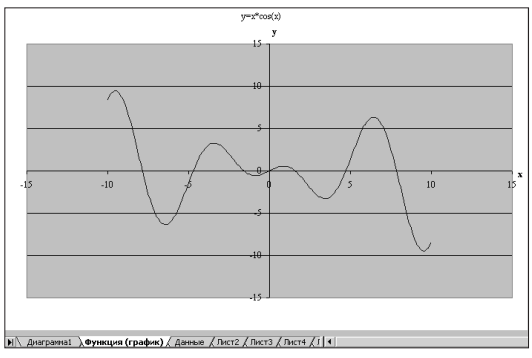
	A	B	C	D	E	F
1	*x*	*y*		*a*	*b*	*h*
2	=$D$2	=A2*COS(A2)		-10	10	0,1
3	=A2+$F$2	=A3*COS(A3)				
4	=A3+$F$2	=A4*COS(A4)				
5	=A4+$F$2	=A5*COS(A5)				

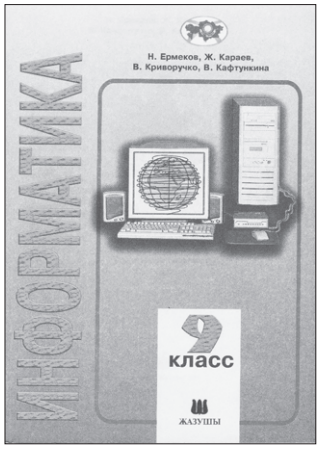
2) Получив необходимые значения, переходим собственно к построению графика. Для этого воспользуемся мастером диаграмм. Из всех диаграмм наиболее подходящей представляется точечная.

Ниже приведены серия рисунков, иллюстрирующих процесс (шаги) построения графика, и фрагмент таблицы, содержащей конечный результат.









Моделирование

Н.Турлынович,
Казахстан

Окончание. См. № 9, 10/2002

Пример. Программа для расчета биоритмов человека.

Существует теория, что жизнь человека подчиняется трем циклическим процессам, называемым "биоритмами". Эти циклы описывают три стороны самочувствия человека: физическую, эмоциональную, интеллектуальную. "Взлетам" графика, представляющего собой синусоиду, соответствуют более благоприятные дни. Дни, в которые график переходит через ось абсцисс, являются критическими, то есть неблагоприятными. В некоторых странах в критические дни, когда ось абсцисс пересекают две или три кривые, людям рискованных профессий предоставляется выходной.

За точку отсчета всех трех биоритмов берется день рождения человека. Указанные циклы можно описать следующими выражениями, в которых переменная t соответствует возрасту человека в днях:

физический цикл — $R_ф(t) = \sin(\frac{2\pi t}{23})$,

эмоциональный цикл — $R_э(t) = \sin(\frac{2\pi t}{25})$,

интеллектуальный цикл — $R_и(t) = \sin(\frac{2\pi t}{33})$.

Согласно табл. 1 перенесем на форму элементы, изменим их размеры и месторасположение, как показано на *рис.* 1, и с помощью окна свойств установим значения.

Элементы `Frame` нужны для оформления приложения. Создадим массив из шести элементов `TextBox`, три из которых поместим в рамку `Frame1`, оставшиеся — в рамку `Frame2`. Перенесем из панели инструментов один объект `PictureBox` в рамку `Frame3`, другой — в рамку `Frame4`, третий — в рамку `Frame5`.

Самостоятельно подберите значения свойства `Font` для элементов `Frame` и `TextBox`.

Таблица 1

Объект	Свойство			
	Name	**Caption**	**ForeColor**	**BackColor**
Форма	Form1	Биоритмы	Белый	Серый
Рамка	Frame1	Дата рождения	Белый	Сине-зеленый
Рамка	Frame2	Текущая дата	Белый	Сине-зеленый
Рамка	Frame3	Физический цикл	Красный	Серый
Рамка	Frame4	Эмоциональный цикл	Зеленый	Серый
Рамка	Frame5	Интеллектуальный цикл	Синий	Серый
Кнопка	Command1	ПУСК		
Кнопка	Command2	ВЫХОД		

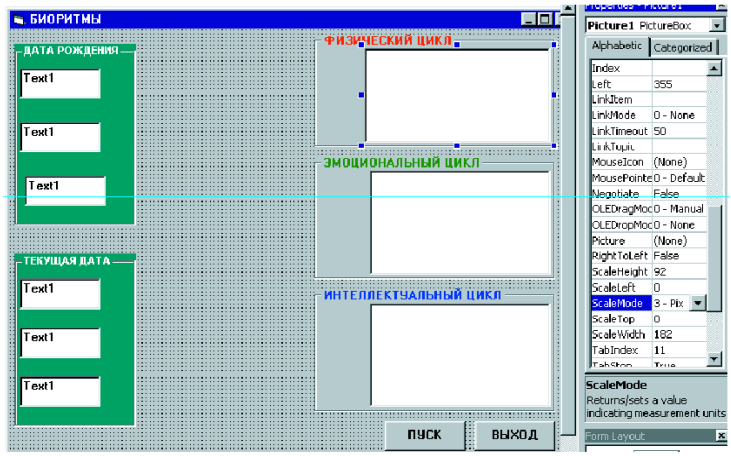


Рис. 1

Таблица 2

Объект	Свойства		Назначение
	Name	**Index**	
TextBox	TextBox1	0	Ввод числа, месяца и года рождения.
TextBox	TextBox1	1	
TextBox	TextBox1	2	
TextBox	TextBox1	3	Ввод числа, месяца и года расчета биоритмов.
TextBox	TextBox1	4	
TextBox	TextBox1	5	
PictureBox	PictureBox1		Построение графиков физического, эмоционального и интеллектуального циклов.
PictureBox	PictureBox2		
PictureBox	PictureBox3		

Идея решения задачи:

1. Определение возраста человека:

❶ Записать дату рождения и текущую дату в массив. Мы убедились, что так удобнее хранить и обрабатывать информацию.

Таблица 3

a(1)	a(2)	a(3)	a(4)	a(5)	a(6)
11	5	1974	25	8	2000
день	месяц	год	день	месяц	год
рождения			расчета биоритмов		

❷ Определить количество дней в годах, начиная с года рождения (1974) до года (2000), как $s = s + 365$ для невисокосного года и $s = s + 366$ для високосного года (см. табл. 4).

❸ Так как при расчете s были добавлены лишние дни месяцев:

$s1$ — для 1974 года (месяцы с 1-го по 5-й),

$s2$ — для 2000 года (месяцы с 8-го по 12-й),

необходимо откорректировать s:

$s = s - s1 - s2$.

Таблица 4

Год	1974	1975	1976	1977	...	2000
Количество дней	365	365	366	365	...	366

❹ Осталось добавить к s прожитое количество дней в 5-м месяце 1974 года и в 8-м месяце 2000 года:

$s = s + d1 - a(1) + 1 + a(4)$, где $d1$ — количество дней в мае (5-й месяц).

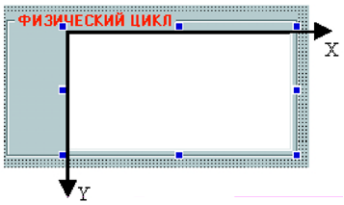
2. Построение графиков.

Построение графиков включает в себя реализацию циклического процесса биоритмов на текущий месяц и отметку на графике текущего дня. Формулы для расчета:

$$R_ф(t) = \sin(\frac{2\pi t}{23}), \quad R_э(t) = \sin(\frac{2\pi t}{25}), \quad R_и(t) = \sin(\frac{2\pi t}{33}).$$

Чтобы сделать прогноз на текущий месяц, необходимо определить значение s на первый день текущего месяца (ss) и количество дней в этом месяце ($d2$):

$$ss = s + d1 - a(1) + 1.$$



Выделим объект `Picture1` и найдем его свойства:

`DrawWidth` — **толщина линии**,
`ScaleLeft` — **координата x левого верхнего угла объекта** `Picture1`,
`ScaleTop` — **координата y левого верхнего угла объекта** `Picture1`,
`ScaleHeight` — **высота объекта** `Picture1`,
`ScaleWidth` — **ширина объекта** `Picture1`,
`ScaleMode` — **задает масштаб системы координат, по умолчанию используется единица твип — 1/20 точки**.

Установим значение `ScaleMode` равным `3-Pixel`, тогда устанавливается привычный масштаб для рисования.

Строка программы (кода), используя метод `Pset`, позволяет построить графики:

```
Picture1.Pset(ScaleLeft + t * 6,ScaleTop + 46 + 20 * sin(6.28 * (ss + t) / 23)), RGB(255,0,0)
```

— где 46 — `ScaleHeight/2`

Команда `Picture1.Pset` означает, что изображение точки будет помещено внутрь объекта `Picture1` относительно его координатных осей.

Выпишем из окна свойств значения:

```
ScaleLeft = 0
ScaleTop = 0
ScaleHeight = 92
ScaleWidth = 182
```

Они будут использоваться для построения графиков.

Процесс рисования точки и других фигур подобен применяемому в языке QBasic. Можно было бы обычным способом задать цвет рисования. Аргумент RGB расширяет возможности метода `Pset`, позволяя программисту самому задавать долю красного, зеленого и синего цветов, формируя палитру из 255 цветов и их оттенков:

`RGB(`*красный, зеленый, синий*`)`

Если задать `RGB(255,0,0)`, точки внутри объекта `Picture1` будут изображаться красным цветом. Программа позволяет дорисовать на созданной форме дополнительные элементы (*рис.* 2).

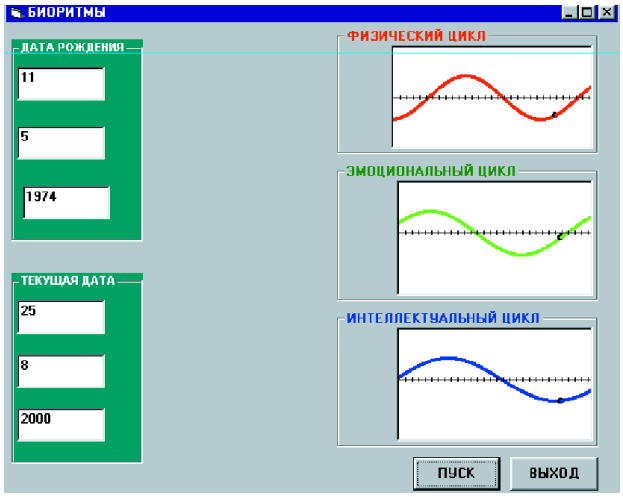


Рис. 2

```
Dim c, n, d, d1, d2, t1, i As Integer      Описание переменных.
Dim s, s1, s2, ss As Single
Dim a(1 To 6) As Integer                   Описание массива a.
Private Sub Command1_Click()
For i = 0 To 5                             Запись в массив введенной
a(i + 1) = Val(Text1(i).Text)              даты рождения и текущей даты.
Next
'Расчет s
s = 0
For i = a(3) To a(6)
If i / 4 = Int(i / 4) Then s = s + 366 Else s = s + 365
Next
If a(3) / 4 = Int(a(3) / 4) Then c = 1 Else c = 0
s1 = 0
For i = 1 To a(2)                          Расчет s1, при этом используется
Call dni(c, i, d)                          процедура dni.
```

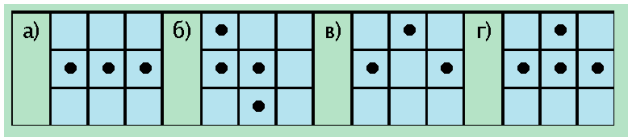
```
If i = a(2) Then d1 = d
s1 = s1 + d
Next
If a(5) / 4 = Int(a(5) / 4) Then c = 1 Else c = 0
s2 = 0
For i = a(5) To 12  Расчет s2, при этом используется
Call dni(c, i, d)   процедура dni.
s2 = s2 + d
If i = a(5) Then d2 = d
Next
s = s - s1 - s2
s = s + (d1 - a(1) + 1) + a(4) Корректировка s.
ss = s + (d1 - a(1) + 1)  Расчет ss.
'Построение графика физического цикла
t1 = 0
For t = 0 To d2 Step 0.1
t1 = t1 + 1
Picture1.DrawWidth = 2
Picture1.PSet(ScaleLeft+t*6, _
    ScaleTop+46+20*Sin(6.28*(ss+t)/23)), RGB(255, 0, 0)
'Отметка на графике текущего дня:
If t1/10 = a(4) Then Picture1.Circle (ScaleLeft + t * 6, _
    ScaleTop+46+20 * Sin(6.28 * (ss + t) / 23)), 2, RGB(0, 0, 0)
Next t
'Рисование оси абсцисс
Picture1.DrawWidth = 1
Picture1.Line (ScaleLeft, ScaleTop + 46)-_
    (ScaleLeft + 182, ScaleTop + 46), RGB(0, 0, 0)
'Рисование единичных отрезков на оси абсцисс.
For t = 1 To 30 Step 1
Picture1.Line (ScaleLeft + t * 6, ScaleTop + 44)-_
    (ScaleLeft + t * 6, ScaleTop + 48), RGB(0, 0, 0)
Next t
'Построение графика эмоционального цикла
t1 = 0
For t = 0 To d2 Step 0.1: t1 = t1 + 1
Picture2.DrawWidth = 2
Picture2.PSet (ScaleLeft + t * 6, ScaleTop + 46 + _
    20 * Sin(6.28 * (ss + t) / 28)), RGB(0, 255, 0)
If t1 / 10 = a(4) Then Picture2.Circle (ScaleLeft + t * 6, _
    ScaleTop + 46 + 20 * Sin(6.28 * (ss + t) / 28)), 2, RGB(0, 0, 0)
Next t
Picture2.DrawWidth = 1
Picture2.Line (ScaleLeft, ScaleTop + 46)-_
    (ScaleLeft + 182, ScaleTop + 46), RGB(0, 0, 0)
For t = 1 To 30 Step 1
Picture2.Line (ScaleLeft + t * 6, ScaleTop + 44)-_
    (ScaleLeft + t * 6, ScaleTop + 48), RGB(0, 0, 0)
Next t
'Построение графика интеллектуального цикла
t1 = 0
For t = 0 To d2 Step 0.1: t1 = t1 + 1
Picture3.DrawWidth = 2
Picture3.PSet (ScaleLeft + t * 6, ScaleTop + 46 + _
    20 * Sin(6.28 * (ss + t) / 33)), RGB(0, 0, 255)
If t1 / 10 = a(4) Then Picture3.Circle (ScaleLeft + t * 6, _
    ScaleTop + 46 + 20 * Sin(6.28 * (ss + t) / 33)), 2, RGB(0, 0, 0)
Next t
Picture3.DrawWidth = 1
Picture3.Line (ScaleLeft, ScaleTop + 46)-_
    (ScaleLeft + 182, ScaleTop + 46), RGB(0, 0, 0)
For t = 1 To 30 Step 1
Picture3.Line (ScaleLeft + t * 6, ScaleTop + 44)-_
    (ScaleLeft + t * 6, ScaleTop + 48), RGB(0, 0, 0)
Next t
End Sub
Private Sub Command2_Click()
'Окончание работы с приложением.
End
End Sub
```

```
Private Sub Form_Load()
' Установка формата ввода в поля даты рождения и даты расчета биоритмов
For i = 0 To 5
If i = 2 Or i = 5 Then Text1(i).Text = "0000" Else Text1(i).Text = "00"
Next
End Sub
Sub dni(c, n, d)
Select Case n
Case 1, 3, 5, 7, 8, 10, 12
d = 31
Case Is = 2 And c = 0
d = 28
Case Is = 2 And c = 1
d = 29
Case 4, 6, 9, 11
d = 30
End Select
End Sub
```

Процедура использует программу примера 1 для определения количества дней в месяце.

Вопросы и задания

1. Что понимают под моделью?
2. Назовите основные виды моделей. Подтвердите их примерами.
3. Какие виды модельных представлений показаны в примерах?
4. Получить на экране компьютера фигуры Лиссажу (см. № 9/2002) и записать параметры, при которых они получены.
5. Исследуйте эволюцию четырех тетрамино, изображенных на рисунке:



6. Среди любителей игры "Жизнь" принято давать характерным конфигурациям не слишком научные, но зато выразительные названия. Измените программу для игры "Жизнь" так, чтобы можно было вводить начальные конфигурации "Улей", "Вертушка", "Восьмерка", "Навигационные огни", "Чеширский кот":

а) "Улей". На каком поколении из начальной конфигурации "Улей" получится "Пасека"?

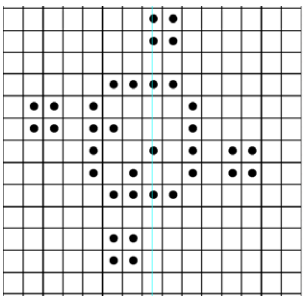
б) Покажите, что внутренняя часть конфигурации "Вертушка" поворачивается на 90° на каждом ходу.

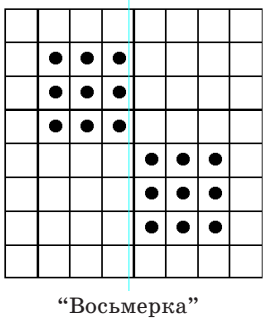
в) Покажите, что конфигурация "Восьмерка" не только внешне повторяет эту цифру, но и повторяется циклически.

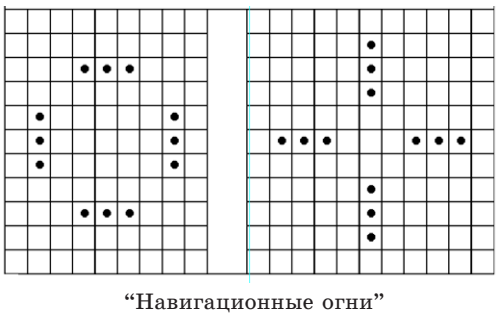
г) На каком шаге из пентамино получаются навигационные огни?

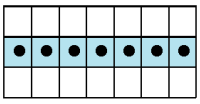
д) Чеширский кот — это персонаж сказки Л.Кэрролла "Алиса в стране чудес". Проследите за судьбой Чеширского кота. На каком ходу от него остается только улыбка, а затем след лапы (блок из четырех фишек)?

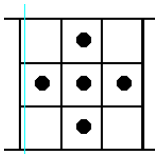
7. Выполните модель калькулятора, вычисляющего значения математических функций.
8. Определите по биоритмам наилучший день текущей недели для отгадывания кроссвордов.
9. Среди одноклассников подберите себе по биоритмам помощников для интеллектуальной работы.
10. Постройте суммарный график биоритмов.

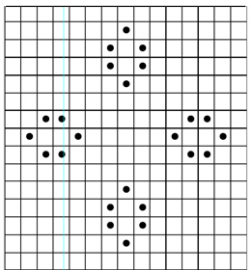

"Вертушка"

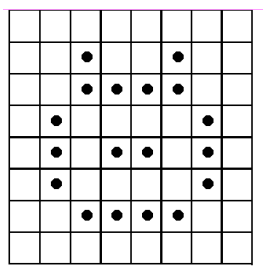

"Восьмерка"


"Навигационные огни"


"Улей"


"Пентамино"


"Пасека"


"Чеширский кот"

Азы информатики

А.А. ДУВАНОВ

Материалы Роботландского университета

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

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

6. ПЕРЕДАЧА ИНФОРМАЦИИ

Читальный зал

Передача информации

Хорошо, когда на складе много товаров. И все же лучше, если товары служат людям, а не лежат без движения на складе. Об информации можно сказать то же самое: чтобы информация была полезной, ее надо своевременно передать. Информацию передают друг другу животные и растения.

Своими "танцами" пчелы-разведчики "рассказывают" соседям по улью, как далеко и в какую сторону надо лететь за нектаром.

В жаркой Индии по берегам рек растут густые заросли растения с удивительным названием "стыдливая мимоза". Когда начинается тропический ливень, стыдливая мимоза спешит свернуть свои листочки, спасая их от сильных струй.

Но самое интересное в поведении этого растения состоит в том, что, как только первые капли дождя упадут хотя бы на одно из растений, сигнал о наступающем дожде передается от ветви к ветви, и все растения длинной цепи зарослей сворачивают свои листья.

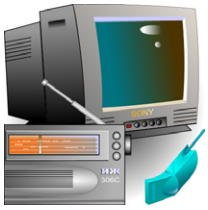
Деятельность людей всегда связана с передачей информации.

Древний способ передачи информации — письмо, отправляемое с гонцом.

В африканских джунглях в старину самые важные новости передавались от одного племени к другому барабанным боем.

Моряки иногда пользуются для передачи информации флажковой азбукой.

Разговаривая, мы передаем друг другу информацию.



Люди придумали много разных устройств для быстрой передачи информации: телевизор, радио, телеграф, телефон.

К числу устройств, передающих информацию с большой скоростью, относятся электронные вычислительные машины. Компьютер может быстро получить информацию, прочитав ее, например, с лазерного диска. Еще быстрее ЭВМ выводит информацию на экран монитора — это тоже передача информации.

В современном мире компьютеры связаны между собой в единую информационную сеть — Интернет. Получить информацию через Интернет можно гораздо быстрее, чем по бумажной почте, а главное, в Интернете хранится практически все, что вам может понадобиться. Нужно обязательно научиться пользоваться этой огромной информационной копилкой.

В передаче информации всегда участвуют две стороны: тот, кто передает информацию, — **источник** и тот, кто ее получает, — **приемник**.

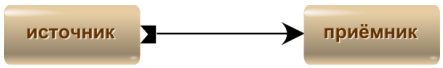


схема передачи информации

Источник информации иногда называют **передатчиком**. Эти два термина обозначают одно и то же: объект, который передает информацию.

Объясняя урок, учитель передает информацию ученикам. В этом случае учитель — источник информации, а ученики — приемники.

А когда ученик отвечает на вопрос, он становится источником, а учитель — приемником.

Передача не должна изменять информацию, а должна только переносить ее от источника к приемнику.

Канал передачи

Положите руку во время разговора на свое горло. Вы почувствуете, как горло дрожит. Эти колебания создаются голосовыми связками — двумя складками слизистой оболочки гортани. Голосовые связки похожи на струны. Они заставляют колебаться воздух, и эти колебания передаются на расстояние, достигая ушей тех, кто слушает.

А что происходит дальше? Колебания воздуха, созданные голосовыми связками, заставляют вибрировать в такт барабанные перепонки — тонкие листики, образованные соединительной тканью в ухе человека. Колебания барабанных перепонок и воспринимаются нами как звук.

Из сказанного можно сделать вывод: когда информация передается голосом, то она поступает к приемнику через воздух (колебания воздуха). Воздух в этом случае является проводником информации, или, как говорят, **каналом передачи**.

Известны водные каналы, которые соединяют между собой моря.

Беломорско-Балтийский канал, например, соединяет Белое море с Балтийским. Канал передачи соединяет между собой источник и приемник информации.

Канал передачи часто называют и по-другому — **каналом связи**. Эти два термина обозначают одно и то же.

Приведенную выше схему передачи информации можно теперь уточнить:

схема передачи информации

В зависимости от типа сигналов, используемых для передачи сообщений, различают звуковые, оптические, электрические и радиоканалы связи:

Тип канала	Тип сигнала	Пример передачи
Звуковой канал	Звуковые волны	Разговор
Оптический канал	Свет	Флажковая азбука
Электрический канал	Электричество	Телефон
Радиоканал	Радиоволны	Сотовый телефон

Когда информацию передают голосом в классе, то каналом связи является воздух, тип этого канала — звуковой.

Звук можно передавать не только по воздуху. Когда один водолаз передает другому сигнал, постукивая камешком о камешек, каналом связи является вода, а тип этого канала по-прежнему звуковой.

Телефонная трубка преобразует звуки в электрические сигналы, которые передаются по проводам. Провода — это канал связи электрического типа.

Информацию можно передавать вместе с носителем, который ее содержит.

Так работает всем известный канал связи — почта.

Сообщение, написанное на листочке, можно передать по телефону, листочек при этом останется у отправителя. А можно положить листочек в конверт и отправить по почте. Тогда адресат получит информацию вместе с носителем.

Почтой можно считать любую доставку предметов потребителю. Наверное, вы не раз пользовались почтой в классе, передавая записки друг другу.

Преобразование информации при передаче

Информацию можно передать по почте вместе с информационным носителем.

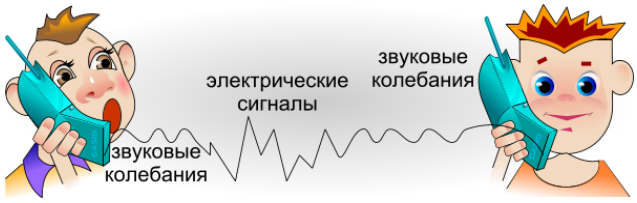
А можно устроить передачу, используя звуковые, оптические, электрические или радиосигналы. Тогда возникает прямая задача о преобразовании информации в эти сигналы и обратная: преобразование сигналов в информацию исходного вида.

Давайте рассмотрим обычный жизненный пример преобразования информации: Вася прочитал интересную книжку о кошках и пересказал ее Пете.

Информация о кошках хранилась в Васиной памяти. Васины голосовые связки преобразовали ее в звуковые сигналы. Эти сигналы (звуковые колебания) поступили по воздушному каналу на барабанные перепонки Пети. Барабанные перепонки преобразовали звуковые колебания, и теперь информация стала храниться в голове Пети в том же виде, в котором она хранилась в голове Васи.

Другой пример: Петя пересказывает книжку по телефону. Звуковые колебания попадают на мембрану телефонной трубки, которая устроена точно так же, как барабанная перепонка. Мембрана (металлический листик) преобразует звуковые колебания в электрические сигналы, и они передаются по проводам к телефонной трубке Васи.

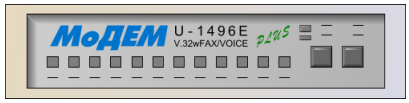
Телефонная трубка имеет две мембраны: в одну из них мы говорим, а другую слушаем. Когда электрические сигналы добрались до Васиной трубки, звуковоспроизводящая мембрана преобразовала их в звук (колебания воздуха) и Вася услышал рассказ Пети.



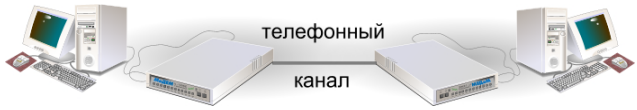
Передача информации в Интернете

В Интернете информация от компьютера к компьютеру передается в виде электрических сигналов по телефонным проводам.

Прибор, который преобразует компьютерные сигналы в телефонные, называется **модемом**.



Модем выполняет и обратные преобразования: телефонные сигналы преобразует в компьютерные.



Важная характеристика модема — скорость передачи (и приема) информации. Современные модемы могут работать со скоростями около 10 000 символов в секунду, то есть они способны за одну секунду передать 10 000 символов (примерно 3 страницы обычной бумажной книги).

Гораздо большей скорости передачи можно достичь, если соединять компьютеры не по телефонным проводам, а при помощи оптического кабеля. Через такой кабель передаются не электрические сигналы, а световые.

Скорость передачи при этом достигает 10 миллионов символов в секунду.

Три огромных бумажных тома по 800 страниц в каждом можно передать при такой скорости за одну секунду!

Оптическая проводная связь в Интернете сейчас еще не слишком распространена, значит, скорость передачи данных в глобальной компьютерной сети остается модемной, то есть достаточно низкой.

Интернет почти никогда не связывает пользователя с источником информации напрямую: передача выполняется через промежуточные компьютеры:

Если компьютер А передает информацию компьютеру В со скоростью 100 символов в секунду, а В передает С со скоростью 1000 символов в секунду, то скорость передачи от А к С составит только 100 символов в секунду (скорость самого медленного канала передачи).

Компьютер В может передавать очень быстро, но получает-то он медленно, значит, вынужден "простаивать", ожидая прихода очередного символа.

Это наблюдение говорит о том, что, несмотря на отдельные оптические соединения компьютеров в Интернете, скорость его остается по-прежнему телефонной.

Конспект

Чтобы информация приносила пользу, ее надо своевременно передать.

Информацию передают друг другу животные и растения.

Деятельность людей всегда связана с передачей информации.

Люди придумали много устройств для передачи информации: телевизор, радио, телеграф, телефон, компьютер, Интернет.

В передаче информации всегда участвуют две стороны: тот, кто передает информацию, — **источник** и тот, кто ее получает, — **приемник**.

Канал связи — это среда, по которой передается информация.

Почта — это канал связи, по которому информация передается вместе с информационным носителем.

При передаче по каналам связи информация может быть преобразована в сигналы.

Модем — это прибор, который преобразует компьютерную информацию для передачи ее по телефонным линиям.

Вопросы

1. Как называют объект, который передает информацию?
2. Как называют объект, который получает информацию?
3. Что такое канал связи?
4. Какие бывают типы каналов связи?
5. Чем отличается канал связи "почта" от каналов, которые передают сообщения в виде сигналов?
6. Для чего используют преобразование информации при передаче?
7. Расскажите о том, как передается информация при разговоре. Что является каналом передачи? Какого типа этот канал? Как происходит преобразование информации?

8. Расскажите о том, как передается информация при телефонном разговоре. Что является каналом передачи? Какого типа этот канал? Как происходит преобразование информации?

9. Приведите несколько примеров передачи информации людьми, животными, техническими устройствами. Для каждого примера укажите источник, приемник информации, канал связи и его тип.

10. С помощью каких органов чувств человек получает информацию?

Укажите источники информации, канал связи и его тип, назовите орган чувств, который является приемником.

11. В следующих примерах укажите источник, приемник информации, канал связи и его тип:

— вы читаете письмо;
— звенит будильник;
— учитель пишет на доске задание;
— ваша семья смотрит телевизор;
— вы работаете с компьютером.

12. Где хранится информация Интернета?

13. Как передается информация в Интернете? По каким каналам связи?

14. Что такое модем?

15. Что такое скорость передачи?

16. С какой скоростью информация передается в Интернете?

Задания на дом

Вариант 1

1. Запишите примеры Читального зала в виде таблицы:

	Вид информации	Источник	Приемник	Канал	Содержание
Танец пчелы	Визуальная	Пчела-разведчик	Рабочие пчелы	Оптический	Наличие корма и расстояние до него
...	...	...	...	...	...

2. Радист передает радиограммы со скоростью 49 знаков в минуту.

Сколько времени потребуется ему для передачи информации:

Прибыли на место крушения японского траулера в 16.20. Все люди спасены. Следуем в порт Владивосток.

Вариант 2

Подсчитайте время, которое вам потребуется для набора с помощью программы Блокнот следующего текста:

Шотландские народные стихи
в переводе И.Токмаковой.

Лошадка пони

Мою лошадку пони
Зовут Малютка Грей.
Соседка наша в город
Поехала на ней.
Она ее хлестала
И палкой и кнутом,
И под гору и в гору
Гнала ее бегом.

Не дам ей больше пони
Ни нынче, ни потом.
Пускай хоть все соседи
Придут просить о том!

Запишите в тетради время и выполните расчет для ответа на вопрос: какова ваша скорость передачи информации с помощью клавиатуры?

Вариант 3

1. Вспомните примеры передачи информации в окружающем нас мире, про которые не рассказывалось в Читальном зале. Запишите ответ в виде таблицы:

	Вид информации	Источник	Приемник	Канал	Содержание
Мох на дереве	Визуальная	Дерево	Человек	Оптический	Мох растет с северной стороны
...	...	...	...	...	...

2. Соберите информацию о технических устройствах для передачи информации, изобретенных в XIX—XX веках. Представьте эту информацию в виде таблицы, отображающей автора и время изобретения, вид передаваемой информации, используемые средства связи, примерную скорость передачи.

Практикум

Исправление клавиатурных ошибок

Ошибки появляются неизбежно у тех, кто занимается делом. Их не надо бояться, их надо уметь исправлять!

Ошибка в тетради весьма неприятна, особенно если она сделана ручкой: после исправления появляется грязь.

На компьютере писать удобнее: ошибки исправляются просто и без всяких следов.

Ошибки при письме или при клавиатурном наборе бывают трех типов:

1. Написана лишняя буква. Лишнюю букву надо удалить.
2. Пропущена буква. Недостающую букву надо вставить.
3. Вместо правильной буквы написана неверная. Неверную букву надо заменить.

Правила исправления ошибок одинаковы и для букв, и для цифр, и для других знаков.

Для всех знаков на клавиатуре: букв, цифр, знаков препинания, специальных знаков (#, &, %, <, >, = и других) — существует общее название **символ**.

Разберем алгоритмы исправления ошибок во всех трех случаях.

Продолжение читайте на с. 26

Основные понятия DNS

Как известно, протокол TCP/IP был создан под руководством Винтона Серфа группой *Internet Network Working Group* (INWG). Он предусматривал пакетный обмен информацией между компьютерами, входящими в сеть. И перед разработчиками встала проблема маршрутизации пакетов, чтобы обеспечить передачу данных на нужный компьютер. Для этого было принято решение дать каждому компьютеру уникальный четырехбайтовый номер. Обычно он записывается в виде четырех чисел, представляющих значения каждого байта в десятичной форме и разделенных точками, например, 128.10.2.30 — традиционная десятичная форма представления адреса, а 10000000 00001010 00000010 100011110 — двоичная форма представления этого же адреса.

Адрес состоит из двух логических частей — номера сети и номера узла в сети. Какая часть адреса относится к номеру сети, а какая — к номеру узла, определяется значениями первых битов адреса. Значения этих битов являются также признаками того, к какому классу относится тот или иной IP-адрес.

На *рис.* 1 показана структура IP-адреса разных классов.

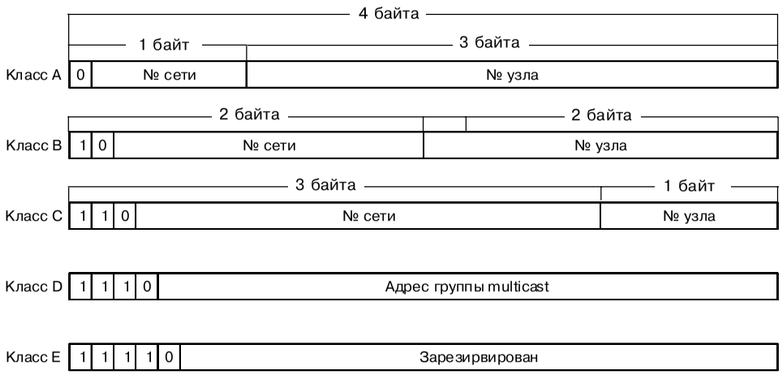


Рис. 1. Структура IP-адреса

Если адрес начинается с 0, то сеть относят к классу A и номер сети занимает один байт, остальные 3 байта интерпретируются как номер узла в сети. Сети класса A имеют номера в диапазоне от 1 до 126. (Номер 0 не используется, а номер 127 зарезервирован для специальных целей, о чем будет сказано ниже.) Сетей класса A немного, зато количество узлов в них может достигать 2^{24}, то есть 16 777 216 узлов.

Если первые два бита адреса равны 10, то сеть относится к классу B. В сетях класса B под номер сети и под номер узла отводится по 16 бит, то есть по 2 байта. Таким образом, сеть класса B является сетью средних размеров с максимальным числом узлов 2^{16}, что составляет 65 536 узлов.

Если адрес начинается с последовательности 110, то это сеть класса C. В этом случае под номер сети отводится 24 бита, а под номер узла — 8 бит. Сети этого класса наиболее распространены, число узлов в них ограничено 2^8, то есть 256 узлами.

Если адрес начинается с последовательности 1110, то он является адресом класса D и обозначает особый, групповой адрес — multicast. Если в пакете в качестве адреса назначения указан адрес класса D, то данный пакет должен быть доставлен сразу нескольким узлам, которые образуют группу с номером, указанным в поле адреса.

Если адрес начинается с последовательности 11110, то это значит, что данный адрес относится к классу E. Адреса этого класса зарезервированы на будущее.

В табл. 1 приведены диапазоны номеров сетей и максимальное число узлов, соответствующих каждому классу сетей.

Таблица 1. Характеристики адресов разного класса

Класс	Первые биты	Наименьший номер сети	Наибольший номер сети	Максимальное число узлов в сети
A	0	1.0.0.0	126.0.0.0	2^{24}
B	10	128.0.0.0	191.255.0.0	2^{16}
C	110	192.0.1.0	223.255.255.0	2^8
D	1110	224.0.0.0	239.255.255.255	Multicast
E	11110	240.0.0.0	247.255.255.255	Зарезервирован

Большие сети получают адреса класса A, средние — класса B, а маленькие — класса C.

Особый смысл имеет IP-адрес, первый октет которого равен 127. Он используется для тестирования программ и взаимодействия процессов в пределах одной машины. Когда программа посылает данные по IP-адресу 127.0.0.1, то образуется как бы "петля". Данные не передаются по сети, а возвращаются модулям верхнего уровня как только что принятые. Поэтому в IP-сети запрещается присваивать машинам IP-адреса, начинающиеся со 127.

Числовая адресация удобна для машинной обработки таблиц маршрутов. Однако то, что удобно машинам, неудобно людям. Есть спорное мнение, что сама человеческая натура протестует против запоминания чисел типа 192.168.1.34 (что не мешает нам запоминать телефонные номера с кодом города и страны, например 380-564-40-06-24). К тому же IP-адреса совсем не информативны. По IP-адресу невозможно понять, что это: сервер, ПК, маршрутизатор или сетевой принтер. Приятней работать с осмысленными именами, такими, как www.1september.ru.

Поэтому в Сети сначала стали использовать таблицы соответствия числовых адресов именам машин. Эти таблицы сохранились до сих пор и используются многими прикладными программами. Физически они располагаются в самых обычных текстовых файлах.

Таблица 2. Таблица соответствия

IP-адрес	Имя машины
127.0.0.1	Localhost
144.206.160.32	Polyn
144.206.160.40	Apollo

В "стародавние времена", когда Интернет еще назывался ARPANET и Сеть состояла лишь из небольшого количества мощных многотерминальных компьютеров, каждый компьютер имел такой файл (обычно `/etc/hosts`). При появлении в Интернете нового компьютера информация о нем заносилась в файл `hosts`, затем этот файл рассылался на все другие машины.

Недостатки такой схемы начали проявляться довольно быстро: с переходом от больших машин к персональным и с ростом Интернета. Трафик, связанный с обновлением информации при добавлении компьютеров в Интернет, становился до неприличия большим. Кроме того, каждое имя в сети должно быть уникальным, а сделать это становилось все труднее и труднее. Поэтому к середине 80-х годов появилась другая, более гибкая система именования — **система имен доменов**.

DNS (*Domain Name System*) — это распределенная база данных, которая содержит информацию о компьютерах, включенных в сеть Интернет. Характер данных зависит от конкретной машины, но чаще всего информация включает имя машины, IP-адрес и данные для маршрутизации почты. Доменная система имен выполняет несколько задач, но основная ее работа — преобразование имен компьютеров в IP-адреса и наоборот.

Система DNS выполняет функции телефонной книги, в которой каждому номеру телефона поставлена в соответствие запись о фамилии или названии фирмы. Системе передается имя (например, www.1september.ru), а она возвращает IP-адрес 212.5.67.3).

DNS реализует иерархическое пространство имен. Единицей измерения является домен — совокупность имен, у которых несколько старших составных частей совпадают. Например, имена www.zil.mmt.ru, ftp.zil.mmt.ru, yandex.ru и sl.mgu.ru входят в домен ru, так как все эти имена имеют одну общую старшую часть — имя ru.

Если один домен входит в другой домен как его составная часть, то такой домен могут называть "поддоменом" (*subdomain*), хотя название "домен" за ним также остается. Обычно поддомен называют по имени той его старшей составляющей, которая отличает его от других поддоменов. Например, поддомен mmt.ru обычно называют "поддоменом" (или "доменом") mmt. Имя поддомену назначает администратор вышестоящего домена. Хорошей аналогией домена является каталог файловой системы.

Необходимо подчеркнуть, что компьютеры входят в домен в соответствии со своими составными именами, при этом они могут иметь совершенно различные IP-адреса, принадлежащие к различным сетям и подсетям. Например, в домен mgu.ru могут входить хосты с адресами 132.13.34.15, 201.22.100.33, 14.0.0.6. Доменная система имен реализована в сети Интернет, но она может работать и как автономная система имен в крупной корпоративной сети, использующей стек TCP/IP, но не связанной с Интернетом.

В DNS вся сеть представляется в виде единого иерархического дерева. На вершине располагается корневой домен (обозначается символом "."). Ниже находятся домены первого уровня. Поскольку Интернет развивался в первую очередь в США и за счет американских налогоплательщиков, это вызвало некоторый крен при формировании доменов первого уровня: Интернет как бы оказался поделенным между США и всем остальным миром.

Наиболее известные домены первого уровня:

com — коммерческие организации (главным образом в США);

edu — учебные заведения США;

gov — правительственные учреждения США;

mil — военные учреждения США;

net — различные сетевые агентства и Интернет-провайдеры;

int — международные организации;

org — некоммерческие учреждения.

Позднее, когда сеть перешагнула национальные границы США, появились национальные домены типа uk, jp, au, ch и т.п. Для СССР тоже был выделен домен su. После 1991 года, когда республики Союза стали суверенными, многие из них получили свои собственные домены: ru, by, ua, la, li и т.п. Однако Интернет — не СССР, и просто так выбросить домен su из сервера имен нельзя, на основе доменных имен строятся адреса электронной почты и доступ ко многим другим информационным ресурсам Интернета. Поэтому гораздо проще оказалось ввести новый домен к существующему, чем заменить его. Таким образом, в Сети существуют организации с доменными именами, оканчивающимися на su. Мало того, совсем недавно было принято решение возобновить регистрацию доменов в этой зоне.

Доменов верхнего уровня очень немного — всего около 250. Большая часть из них — географические домены. Если бы вам захотелось зарегистрировать еще один домен верхнего уровня, то для этого потребовалось бы предоставить такие серьезные обоснования, что гораздо проще было бы организовать свое маленькое государство и для него уже получить географический домен.

В Интернете корневой домен управляется центром InterNIC.

Вслед за доменами верхнего уровня следуют домены, определяющие либо регионы (msk), либо организации (1september). Далее идут следующие уровни иерархии, которые могут быть закреплены либо за небольшими организациями, либо за подразделениями больших организаций.

Иерархия доменных имен аналогична иерархии имен файлов, принятой во многих популярных файловых системах. Дерево имен начинается с корня. Затем следует старшая символьная часть имени, вторая

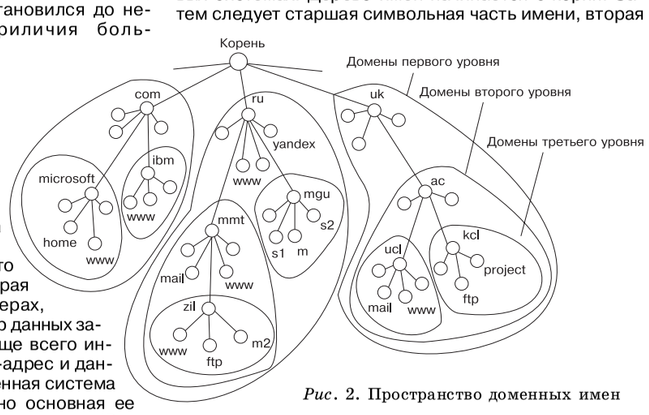


Рис. 2. Пространство доменных имен

по старшинству символьная часть имени и т.д. Младшая часть имени соответствует конечному узлу сети.

В отличие от имен файлов, при записи которых сначала указывается самая старшая составляющая, затем составляющая более низкого уровня и т.д., запись доменного имени начинается с самой младшей составляющей, а заканчивается самой старшей. Составные части доменного имени отделяются друг от друга точкой. Например, в имени partnering.microsoft.com составляющая partnering является именем одного из компьютеров в домене microsoft.com.

Разделение имени на части позволяет разделить административную ответственность за назначение уникальных имен между различными людьми или организациями в пределах своего уровня иерархии. Так, для примера, приведенного на *рис.* 2, один человек может нести ответственность за то, чтобы все имена, которые имеют окончание ru, имели уникальную следующую вниз по иерархии часть. Если этот человек справляется со своими обязанностями, то все имена типа www.ru, mail.mmt.ru или m2.zil.mmt.ru будут отличаться второй по старшинству частью.

Разделение административной ответственности позволяет решить проблему образования уникальных имен без взаимных консультаций между организациями, отвечающими за имена одного уровня иерархии. Очевидно, что должна существовать одна организация, отвечающая за назначение имен верхнего уровня иерархии.

Процедура получения имени, например в зоне ru или com, называется регистрацией домена. Конечно, каждая компания, подключающаяся к Интернету, стремится зарегистрировать как можно более естественное и легкое для запоминания имя. Так, для Microsoft inc. логично зарезервировать домен microsoft.com.

Каждый домен администрируется отдельной организацией, которая обычно разбивает свой домен на поддомены и передает функции администрирования этих поддоменов другим организациям. Чтобы получить доменное имя, необходимо зарегистрироваться в какой-либо организации, которой InterNIC делегировал свои полномочия по распределению имен доменов. В России такой организацией является РосНИИРОС, которая отвечает за делегирование имен поддоменов в домене ru.

Использованные источники информации

1. Имена Интернета.
Подстрешный Артем, **http://www.radio-MSU.net**.
2. Основы сетевых технологий. **http://ssem.zsu.zp.ua**.
3. Как работает DNS. **http://bykov.dp.ua/aix**.

Продолжение. См. с. 20—23

Лишний символ

Удаления лишнего символа можно выполнить при помощи такого алгоритма:

Алгоритм удаления символа

1. Установить курсор перед лишним символом.
2. Нажать клавишу удаления Del.

Проверка алгоритма

Проверим алгоритм на таком примере: вместо слова "лишний" с ошибкой написано:

лишьний

1. С помощью стрелок клавиатуры ставим курсор перед лишним символом:

лишьний

2. Нажимаем клавишу удаления Del:

лишний

Пропущенный символ

Алгоритм вставки символа

1. Подвести курсор к месту вставки (установить за последним правильным символом).
2. Нажать клавишу с нужным символом.

Проверка алгоритма

Проверим алгоритм удаления на таком примере: вместо слова "пропущенный" написано:

пропщенный

1. С помощью стрелок клавиатуры ставим курсор на место вставки:

пропщенный

2. Нажимаем клавишу с нужным символом:

пропущенный

Неверный символ

Алгоритм замены символа

1. Подвести курсор к неверному символу (установить слева).
2. Нажать клавишу Del.
3. Нажать клавишу с верным символом.

Проверка алгоритма

Вместо слова "неверный" написано:

ниверный

1. С помощью стрелок клавиатуры ставим курсор перед неверным символом:

ниверный

2. Нажимаем клавишу удаления Del:

нверный

3. Нажимаем клавишу с нужным символом:

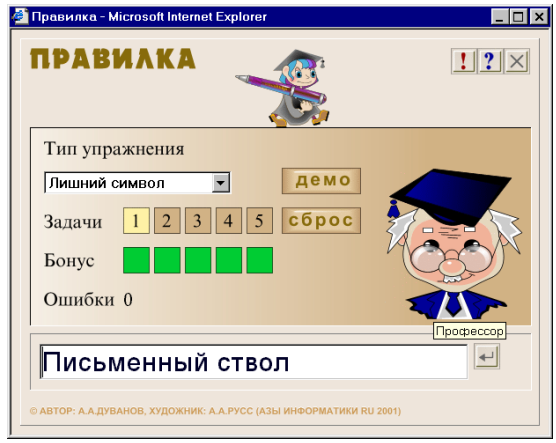
неверный

Тренажер "Правилка"

Освоить простые приемы исправления ошибок поможет тренажер "Правилка".

Тренажерами называют такие устройства или программы для ЭВМ, с помощью которых человек может натренировать какие-либо способности.

Тренажер "Правилка" тренирует умение исправлять ошибки, которые возникают при наборе текстов на клавиатуре.



Режимы работы "Правилки":

— Лишний символ;
— Пропущенный символ;
— Неверный символ;
— Смесь;
— Суперсмесь;
— Контрольная.

В каждом режиме "Правилка" предложит 5 заданий. Начинаем тренировку в звании "Профессор" и стараемся удержать высокий титул до конца работы!

Вопросы практикума

1. Перечислите типы ошибок, которые возникают при наборе текста на клавиатуре.

2. Расскажите алгоритм исправления ошибки "Лишний символ".

3. Расскажите алгоритм исправления ошибки "Пропущенный символ".

4. Расскажите алгоритм исправления ошибки "Неверный символ".

Задания практикума

1. Составьте алгоритм удаления двух символов, идущих подряд.

2. Составьте алгоритм удаления нескольких последних символов, при условии, что курсор расположен сразу за набранным текстом.

3. Для исправления ошибок можно предложить алгоритмы, отличные от тех, которые приводятся в Практикуме. Придумайте и запишите эти новые алгоритмы.

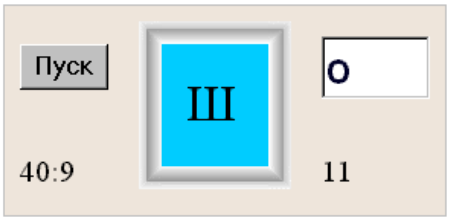
4. Два алгоритма, решающие одну и ту же задачу, можно сравнивать по **быстродействию**, то есть по скорости их выполнения.

Быстродействие алгоритмов исправления ошибок зависит от числа нажатий на клавиши: чем нажатий меньше, тем быстрее работает алгоритм.

Расположите алгоритмы Практикума и придуманные вами алгоритмы по типу ошибки и для каждого типа оцените алгоритмы по быстродействию.

Определитель скорости набора

Компьютерная книга содержит стенд для подсчета вашей скорости передачи информации в компьютер через клавиатуру.



Ваша скорость: 35 **символов в минуту.**

Алгоритм работы на стенде

1. Включить режим русских букв.
2. Нажать кнопку “Пуск”.
3. Пока часы не остановятся, нажимайте клавишу с тем символом, который появляется в окошке.

Зачетный класс

1. Петя рассказал Васе алгоритм исправления неверного символа в тексте.

Установите в меню правильные позиции.

Информационный процесс:
- — Хранение;
- — Передача;
- — Обработка.

Источник:
- — Вася;
- — Петя.

Приемник:
- — Вася;
- — Петя.

Тип канала связи:
- — Звуковой канал;
- — Оптический канал;
- — Электрический канал;
- — Радиоканал;
- — Почта.

Вид информации:
- — Визуальная — Звуковая — Вкусовая — Осязательная — Информация запаха.

2. Вова Бякин передал через Федю Крякова записку Леночке Соколовой.

Установите в меню правильные позиции.

Информационный процесс:
- — Хранение;
- — Передача;
- — Обработка.

Источники:
- — Вова;
- — Вова и Федя;
- — Федя;
- — Федя и Леночка;
- — Леночка.

Приемник:
- — Вова;
- — Вова и Федя;
- — Федя;
- — Федя и Леночка;
- — Леночка.

Тип канала связи:
- — Звуковой канал;
- — Оптический канал;
- — Электрический канал;
- — Радиоканал;
- — Почта.

Вид информации:
- — Визуальная;
- — Звуковая;
- — Вкусовая;
- — Осязательная;
- — Информация запаха.

3. Матрос Швабрин просигналил проходящему мимо сухогрузу приветствие семафорной азбукой.

Установите в меню правильные позиции.

Информационный процесс:
- — Хранение;
- — Передача;
- — Обработка.

Источник:
- — Швабрин;
- — Сухогруз.

Приемник:
- — Швабрин;
- — Сухогруз.

Тип канала связи:
- — Звуковой канал;
- — Оптический канал;
- — Электрический канал;
- — Радиоканал;
- — Почта.

Скорость передачи:
- — 1000 знаков в секунду;
- — 100 знаков в секунду;
- — 1 знак в секунду;
- — 1 знак в минуту.

Вид информации:
- — Визуальная;
- — Звуковая;
- — Вкусовая;
- — Осязательная;
- — Информация запаха.

4. Вася придумал маленький кроссворд по информатике. Запишите слова, которые должны располагаться в клетках этого кроссворда. Как обычно, в кроссвордах вместо буквы “ё” пишут букву “е”.

По вертикали:

1. Тот, кто передает информацию.

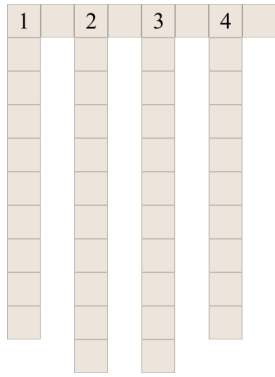
2. Тот, кто выполняет алгоритмы.

3. Использование механизмов.

4. Различные сведения.

По горизонтали:

Тот, кто принимает информацию.



5. Передатчик А передает информацию со скоростью 19 символов в секунду приемнику В.


```
|-----------------|
| Мохнатая азбука |
| --------------- |
| В азбуке этой,  |
| Увидите сами!   |
| Буквы живые:    |
| С хвостами,     |
| С усами.        |
|-----------------|
```

Через сколько секунд приемник получит сообщение (начало стихотворения Бориса Заходера), если передаются все символы в записи, в том числе и знаки: “|”, “—”?

6. Устройство А передает информацию устройству С через В. Устройство В принимает от А сообщение целиком, а затем пересылает его С. Скорость передачи А и В — 50 символов в секунду. Через сколько секунд С получит от А сообщение в 200 символов?

7. Устройство А передает информацию устройству С через В. Устройство В принимает символы от А и незамедлительно пересылает их С. Скорость передачи А и В — 50 символов в секунду. Через сколько секунд С получит от А сообщение в 200 символов?

8. Устройство А передает информацию устройству С через В. Устройство В принимает символы от А и незамедлительно пересылает их С. Скорость передачи А — 50 символов в секунду, а скорость передачи В — 200 символов в секунду. Через сколько секунд С получит от А сообщение в 200 символов?

Книга для учителя

6. ПЕРЕДАЧА ИНФОРМАЦИИ

История связи

Тема позволяет организовать интересное исследование по истории связи. В качестве источника информации можно использовать энциклопедии, книги, Интернет.

Одной из тем, за освоение которых дети возьмутся с большой охотой, может быть история телефона и поучительная биография его изобретателя.

Александер Грейам Белл — изобретатель телефона. Родился в Эдинбурге (Шотландия) 3 марта 1847 года. В 1876 году Белл продемонстрировал изобретенный аппарат на всемирной выставке в Филадельфии, что стало сенсацией. Изобретению помогли знания Белла в области акустики и электротехники, а также случайные отклонения, возникшие по ходу проводимых им экспериментов.

Однако первые “телефоны”, если под этим термином понимать передачу звука на большие расстояния, появились существенно раньше.

У персидского царя Кира (VI век до н.э.) в “телефонном” штате состояли 30 000 человек. Эти “телефонные” служащие располагались друг от друга в пределах слышимости и передавали за один день сообщение на расстояние тридцатидневного перехода.

Игры в передачу информации

Хотя материал излагается в форме, доступной младшим школьникам, все же приводимые примеры достаточно абстрактны, а тема допускает различные нетрадиционные методические приемы. Например, простейшую внутриклассную “почту” из записочек можно на несколько минут легализовать, чтобы продемонстрировать процесс передачи информации.

Столь же увлекательной покажется детям передача информации, если вы научите их хотя бы нескольким буквам азбуки глухонемых и попросите передать цепочку букв (или осмысленное слово) из одного угла классной комнаты в другой. Вот, например, слово, составленное из трех простых букв этой азбуки:

Можно придумать свой дактильный алфавит, закрепляя за жестом не букву, а слово или целое предложение.

При демонстрации передачи звуковых сообщений обратите внимание детей на то, что звук всегда создается при помощи вибрирующих элементов в источнике. Попросите детей прикоснуться пальцами к своему горлу и произнести несколько слов. Пальцы почувствуют вибрацию голосовых связок.

Последующее за играми обращение к детям с просьбой привести примеры передачи информации, несомненно, поднимет много рук. Надо, чтобы, рассказывая о факте

передачи, дети называли источник и приемник, канал связи, указывали носители информации, а также способ преобразования для передачи по каналам связи.

Ответы на вопросы практикума

1. Перечислите типы ошибок, которые возникают при наборе текста на клавиатуре.

Лишний символ исправляется удалением.

Пропущенный символ исправляется вставкой.

Неверный символ исправляется заменой.

2. Расскажите алгоритм исправления ошибки "Лишний символ".

Алгоритм удаления символа

1. Установить курсор перед лишним символом.
2. Нажать клавишу удаления Del.

3. Расскажите алгоритм исправления ошибки "Пропущенный символ".

Алгоритм вставки символа

1. Подвести курсор к месту вставки (установить за последним правильным символом).
2. Нажать клавишу с нужным символом.

4. Расскажите алгоритм исправления ошибки "Неверный символ".

Алгоритм замены символа

1. Подвести курсор к неверному символу (установить слева).
2. Нажать клавишу Del.
3. Нажать клавишу с верным символом.

Ответы на задания

1. Составьте алгоритм удаления двух символов, идущих подряд.

Возможный вариант:

Алгоритм удаления двух символов

1. Подвести курсор к первому лишнему символу (установить слева).
2. Нажать два раза клавишу Del.

2. Составьте алгоритм удаления нескольких последних символов, при условии что курсор расположен сразу за набранным текстом.

Возможный вариант:

Алгоритм удаления последних символов

Нажать клавишу BS столько раз, сколько символов нужно удалить.

3. Для исправления ошибок можно предложить алгоритмы, отличные от тех, которые приводятся в Практикуме. Придумайте и запишите эти новые алгоритмы.

Возможные варианты:

Алгоритм удаления символа

1. Установить курсор сразу после лишнего символа.
2. Нажать клавишу BS.

Алгоритм замены символа

1. Установить курсор справа от неверного символа.
2. Нажать клавишу BS.
3. Нажать клавишу с верным символом.

4. Два алгоритма, решающие одну и ту же задачу, можно сравнивать по **быстродействию**, то есть по скорости их выполнения.

Быстродействие алгоритмов исправления ошибок зависит от числа нажатий на клавиши: чем нажатий меньше, тем быстрее работает алгоритм.

Расположите алгоритмы Практикума и придуманные вами алгоритмы по типу ошибки и для каждого типа оцените алгоритмы по быстродействию.

Если курсор находится за местом исправления.

Алгоритм	В практикуме	В ответах
Удаление	n	$n - 1$
Замена	n	$n - 1$

Если курсор находится перед местом исправления.

Алгоритм	В практикуме	В ответах
Удаление	n	$n + 1$
Замена	n	$n + 1$

Ответы на вопросы

1. Как называют объект, который передает информацию?

Источник или передатчик.

2. Как называют объект, который получает информацию?

Приемник, получатель или потребитель.

3. Что такое канал связи?

Канал связи — это проводник информации — среда, по которой передается информация.

4. Какие бывают типы каналов связи?

В зависимости от типа сигналов, используемых для передачи сообщений, различают звуковые, оптические, электрические и радиоканалы связи:

Тип канала	Тип сигнала	Пример передачи
Звуковой канал	Звуковые волны	Разговор
Оптический канал	Свет	Флажковая азбука
Электрический канал	Электричество	Телефон
Радиоканал	Радиоволны	Сотовый телефон

5. Чем отличается канал связи "почта" от каналов, которые передают сообщения в виде сигналов?

Информация передается по каналу вместе с информационным носителем.

6. Для чего используют преобразование информации при передаче?

Если канал связи не может передавать информацию в том виде, в котором она хранится на информационном носителе, то информацию преобразуют на стороне источника к виду, пригодному для передачи. На стороне приемника производится обратное преобразование.

7. Расскажите о том, как передается информация при разговоре. Что является каналом передачи? Какого типа этот канал? Как происходит преобразование информации?

Голосовые связки вибрируют и создают звуковые колебания (прямое преобразование). Звуковые колебания распространяются по воздуху (канал

передачи звукового типа) и достигают барабанных перепонок слушателя.

Барабанные перепонки преобразуют звуковые колебания к виду, который воспринимает человеческий мозг. Информация воспринимается и запоминается в голове слушателя.

8. Расскажите о том, как передается информация при телефонном разговоре. Что является каналом передачи? Какого типа этот канал? Как происходит преобразование информации?

Звук преобразуется мембраной в электрические сигналы и передается по телефонным проводам (канал передачи электрического типа) на звукообразующую мембрану.

9. Приведите несколько примеров передачи информации людьми, животными, техническими устройствами. Для каждого примера укажите источник, приемник информации, канал связи и его тип.

1. Вася подмигнул Лене.

Источник — Вася.

Приемник — Лена.

Канал связи — воздух, тип канала — оптический.

2. Петух нашел зернышко и подзывает кур, издавая характерные звуки.

Источник — петух.

Приемники — куры.

Канал связи — воздух, звукового типа.

3. Телевидение.

Источник — телевизионная студия.

Приемники — телевизоры.

Канал связи — радиоволны, тип канала — радиоканал.

10. С помощью каких органов чувств человек получает информацию?

Укажите источники информации, канал связи и его тип, назовите орган чувств, который является приемником.

1. Чувство восприятия информации — **зрение**. Орган чувства — глаза.

Источники информации — предметы, которые испускают или отражают свет.

Приемником является человек, животное или техническое устройство (например, фотоаппарат), которое способно воспринимать свет.

Канал связи — та среда, по которой распространяется свет (воздух, вода, космос, специальный кабель), тип канала — оптический.

2. Чувство восприятия информации — **слух**. Орган чувства — уши.

Источники информации — предметы, которые создают звуковые колебания (вибрируют, дрожат).

Приемником является человек, животное или техническое устройство (например, микрофон).

Канал связи — та среда, по которой распространяется звук (воздух, вода), тип канала — звуковой.

3. Чувство восприятия информации — **обоняние**. Орган чувства — нос.

Источники информации — предметы, которые пахнут.

Приемником является человек, животное.

Канал связи — та среда, по которой распространяется запах (воздух, вода), тип канала — канал передачи запахов.

4. Чувство восприятия информации — **вкус**. Орган чувства — язык.

Источники информации — различные предметы.

Приемником является человек, животное.

Канал связи — отсутствует, информация вкуса не передается на расстояние, она возникает при непосредственном контакте органа чувства (языка) с предметом.

5. Чувство восприятия информации — **осязание**. Орган чувства — кожа.

Источники информации — различные предметы.

Приемником является человек, животное.

Канал связи — отсутствует, осязательная информация не передается на расстояние, она возникает при непосредственном контакте органа чувства (кожи) с предметом.

11. В следующих примерах укажите источник, приемник информации, канал связи и его тип.

1. Вы читаете письмо.

Источник — отправитель письма, приемник — адресат, канал связи — почта.

2. Звенит будильник.

Источник — будильник, приемник — тот, кто слышит сигнал будильника, канал связи — воздух, тип канала — звуковой.

3. Учитель пишет на доске задание.

Источник — учитель, приемники — дети в классе, канал связи — воздух, тип канала — оптический.

4. Ваша семья смотрит телевизор.

Источник — телевизор, приемники — зрители, канал связи — воздух, тип канала — оптический.

5. Вы работаете с компьютером.

При передаче информации от компьютера к пользователю источниками являются: экран монитора (оптический канал), звуковые колонки (звуковой канал), а приемником — пользователь. При вводе данных в компьютер источником является пользователь, а приемником — компьютер. Информация может передаваться через клавиатуру, мышь, считываться с дискеты или лазерного диска, а также восприниматься как звук (через микрофон).

12. Где хранится информация Интернета?

Интернет — это мировое объединение компьютеров. То есть в Интернете нет ничего, кроме компьютеров и каналов связи между ними. Информация в Интернете хранится на компьютерах.

13. Как передается информация в Интернете? По каким каналам связи?

По телефонным проводам, по электрическим и оптическим кабелям, при помощи радиоволн.

14. Что такое модем?

Модем — это устройство для преобразования компьютерных сигналов в телефонные и обратно.

15. Что такое скорость передачи?

Скорость передачи — это количество информации, которое передается по каналу за единицу времени.

16. С какой скоростью информация передается в Интернете?

Скорость передачи в Интернете определяется скоростью самого медленного канала, входящего в цепочку передачи сообщений. Так как большинство каналов связи в Интернете — телефонные линии, то средняя скорость передачи определяется именно ими и составляет несколько тысяч символов в секунду.

Решения зачетного класса

1. Петя рассказал Васе алгоритм исправления неверного символа в тексте.

Установите в меню правильные позиции.

Информационный процесс: передача.

Источник — Петя. Приемник — Вася. Тип канала связи — звуковой канал.

Вид информации — звуковая.

2. Вова Бякин передал через Федю Крякова записку Леночке Соколовой.

Установите в меню правильные позиции.

Информационный процесс: передача.

Источники — Вова и Федя.

Приемники — Федя и Леночка.

Тип канала связи — почта.

Вид информации — визуальная.

3. Матрос Швабрин просигналил проходящему мимо сухогрузу приветствие семафорной азбукой.

Установите в меню правильные позиции.

Информационный процесс: передача.

Источник — Швабрин.

Приемник — сухогруз.

Тип канала связи — оптический канал.

Скорость передачи — 1 знак в секунду.

Вид информации — визуальная.

4. Вася придумал маленький кроссворд по информатике. Запишите слова, которые должны располагаться в клетках этого кроссворда.

По вертикали:

1. Передатчик.
2. Исполнитель.
3. Механизация.
4. Информация.

По горизонтали: Приемник.

5. Передатчик А передает информацию со скоростью 19 символов в секунду приемнику В.

```
|-----------------|
| Мохнатая азбука |
| --------------- |
| В азбуке этой,  |
| Увидите сами!   |
| Буквы живые:    |
| С хвостами,     |
| С усами.        |
|-----------------|
```

Через сколько секунд приемник получит сообщение (начало стихотворения Бориса Заходера), если передаются все символы в записи, в том числе и знаки: "|", "—"?

9

6. Устройство А передает информацию устройству С через В. Устройство В принимает от А сообщение целиком, а затем пересылает его С. Скорость передачи А и В — 50 символов в секунду. Через сколько секунд С получит от А сообщение в 200 символов?

8

7. Устройство А передает информацию устройству С через В. Устройство В принимает символы от А и незамедлительно пересылает их С. Скорость передачи А и В — 50 символов в секунду. Через сколько секунд С получит от А сообщение в 200 символов?

5

8. Устройство А передает информацию устройству С через В. Устройство В принимает символы от А и незамедлительно пересылает их С. Скорость передачи А — 50 символов в секунду, а скорость передачи В — 200 символов в секунду. Через сколько секунд С получит от А сообщение в 200 символов?

5

Продолжение следует

Материалы Роботландского университета

Тропинка конструктора

А.А. Дуванов,
Ю.А. Первин

Продолжение. См. № 4, 6/2002

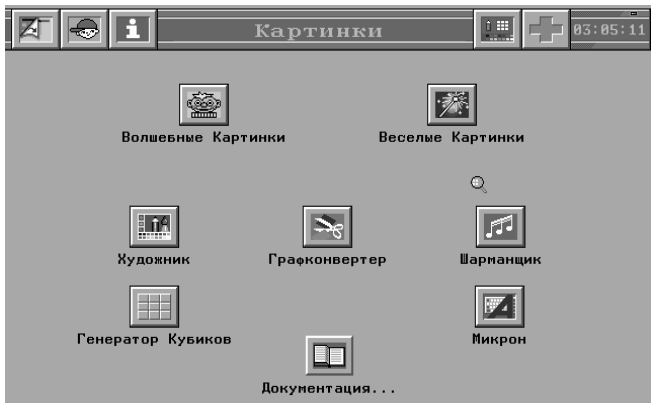
Книга для ученика

ГЛАВА 3. ВОЛШЕБНЫЕ КАРТИНКИ

1. Взгляд со страницы Монитора

Когда дедушка Фёрстов объявил, что сегодня предстоит знакомство с новой программой — *Волшебные картинки*, Тим очень быстро нашел страницу *Монитора* с названием *Картинки*. Количество пиктограмм на этой странице его удивило:

— Ого, как много! Кнопки с документацией даже не поместились на экране, и для них поставлена кнопка-переход на другую страницу. Наверное, *Картинки* — это очень сложное приложение?



Но дедушка успокоил:

— Как раз наоборот: это очень простая среда. Работать с *Картинками* сможет даже дошколенок. Очень просто построить и свою новую задачу. Это можно сделать буквально за пять минут.

Эти слова, однако, не сняли все смятения в душе Дины:

— А почему же так много кнопок?

— Задачи, построенные с помощью *Картинок*, можно оснащать собственной графикой, видеоэффектами, даже музыкальным сопровождением. Хотя на первых порах многие из этих возможностей можно и опустить.

Несмотря на то, что дедушка призвал ребят никуда не спешить, Тиму не терпелось проявить свою эрудицию, и, показывая на пиктограммы, он стал перечислять:

— Тогда понятно: текстовый редактор *Микрон* нужен для того, чтобы писать слова и вообще всякие тексты, графический редактор *Художник* — для того чтобы рисовать картинки, а музыкальный редактор *Шарманщик* — чтобы записывать и слушать музыку.

Но вдруг он остановился:

— А что это за кнопка — *Графконвертер*?

Дедушке было нетрудно удовлетворить его любопытство:

— Графический конвертер — очень удобная штука. Эта программа позволяет просматривать рисунки на диске, вырезать из них фрагменты и переводить графические файлы из одного графического формата в другие.

— Ой! Ему все понятно! — с подковыркой сказала сестра. — Ведь самые главные кнопки на этой странице — это наверняка *Волшебные картинки* и *Веселые картинки*. Даже сама программа так называется. Правда, дедушка?

Дедушка поддержал Дину:

— С этим трудно не согласиться, Диночка. Эти две пиктограммы вызывают на экран готовые программы. Остальные кнопки — вспомогательные. Они пригодятся тогда, когда мы начнем конструировать собственные задачи. Программы *Волшебные картинки* и *Веселые картинки* похожи друг на друга, только в первой из них задачи построены на основе текстовой информации, а во второй — на основе графической. Но и в той, и в другой решение задач в них сводится к построению правильной цепочки упорядоченных объектов.

Тиму пришлось признать:

— Красивые слова, но не очень понятные.

— Поясню на простом примере, — предложил дедушка. — Вот задача (и дедушка в появившемся окне отметил курсором-подсветкой строку *Счет по-английски*), с которой вы, конечно, справитесь. Ведь вы же уже начали изучать английский в школе.

Предвидя новые испытания, Дина скромно призналась за себя и брата:

— Ну, мы только в этом году начали изучать английский. Мы знаем совсем немножко.

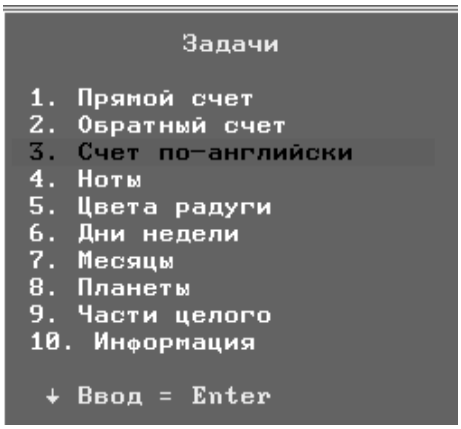
Дедушка не огорчился. Даже обрадовался:

— Ну вот и хорошо. Значит, задание, которое мы сейчас посмотрим, будет для вас полезным.

2. Волшебные картинки

2.1. Решение задач

Ребята устремили свои взоры на экран, где в меню программы была выделена строка *Счет по-английски*:



Дина отметила:

— Целых 10 задач. Выбирай любую!

Дедушка уточнил:

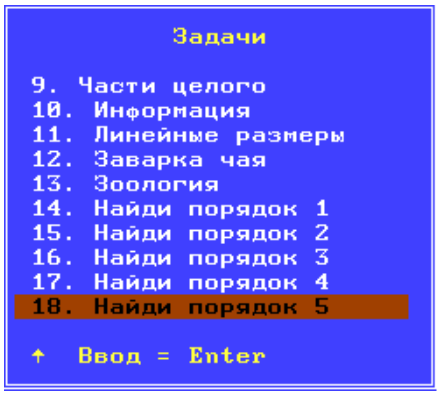
— На самом деле в нашем меню сейчас не 10 задач, а 18. Обратите внимание на стрелку, нарисованную в нижней, служебной, строке. Она подсказывает, что список можно прокручивать дальше.

— Стрелками клавиатуры? — уточнил Тим.

— Да, мышь в этой программе не работает.

Дина, сидевшая у клавиатуры, нажала на клавиши:

— Нажимаю стрелку [↓], список прокручивается... Да, действительно, всего задач в нем 18:

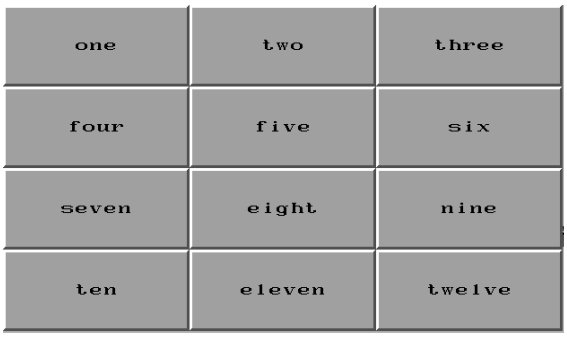


Теперь настала очередь дедушки сделать уточняющее замечание:

— Как вы можете догадаться, на число задач можно сейчас не обращать внимание, поскольку в нашей воле удалить устаревшие или неинтересные задачи, а также добавить новые, которые вы сами придумаете.

— Можно я войду в задание *Счет по-английски*? — попросила Дина.

И, получив дедушкино согласие, она сразу нажала клавишу выполнения. Ребята увидели на экране 12 карточек, покрывающих весь экран. На карточках записаны 12 английских слов — числительных.



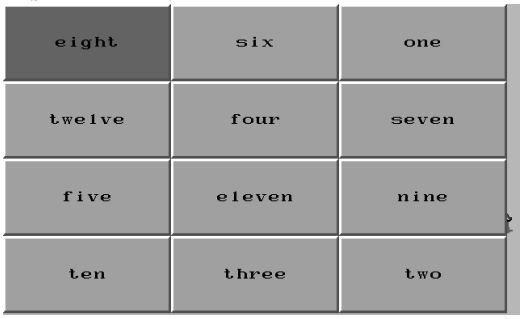
— Понимаете ли вы, в каком порядке расположены карточки на экране?

— Да, конечно, это английские числа по порядку.

И ребята начали хором считать. Выслушав этот хор, дедушка предложил:

— Теперь, Дина, перемешай карточки, — сказал он и, увидев вопросительный взгляд внучки, добавил: — Для этого достаточно нажать любую клавишу.

Карточки закружились по экрану, как снежинки на ветру. Когда через несколько секунд они замерли, экран стал неузнаваем: полнейший беспорядок! Правда, одна из карточек окрасилась голубым цветом.



— Вот тут-то и начинается задача, — сказал дедушка Фёрстов.

Он объяснил, что голубым цветом карточки отмечается положение курсора в *Волшебных картинках*. При помощи управляющих стрелок можно установить эту голубую подсветку — курсор — на любое слово.

— Догадываюсь! — воскликнул Тим. — Надо ставить курсор по очереди на карточки и каждый раз нажимать на клавишу выполнения.

— Молодец! — похвалил дедушка. — Верно!

Тим рассчитывал, что такая искренняя похвала должна обернуться для него местом за клавиатурой, но Дина как ни в чем не бывало продолжала задачу: вела курсор от карточки, нажимая при этом на [Enter], да к тому же еще и приговаривала:

— One, two, three, four, ...

Каждое правильное нажатие на карточку убирало ее с экрана и тем самым постепенно открывало спрятанную под карточками яркую картинку.

— А что будет, если я ошибусь? — вдруг, оробев и приостановившись, спросила Дина.

Дедушка предложил ей:

— А ты сделай эксперимент — ошибись и тогда сразу увидишь, что произойдет.

И хотя Дина понимала, что после “seven” надо выбирать “eight”, она нарочно нажала “eleven”. Но ничего не произошло. Только раздался легкий писк из компьютерного динамика. Карточка же осталась на месте.

— Вот этот самый писк и есть сигнал о твоей ошибке. Из-за нее в конце задачи тебе будет снижена оценка.

— Ну, тогда я больше ошибаться не буду, — сказала Дина и уверенно завершила задание — “nine”, “ten”, “eleven”, “twelve”.

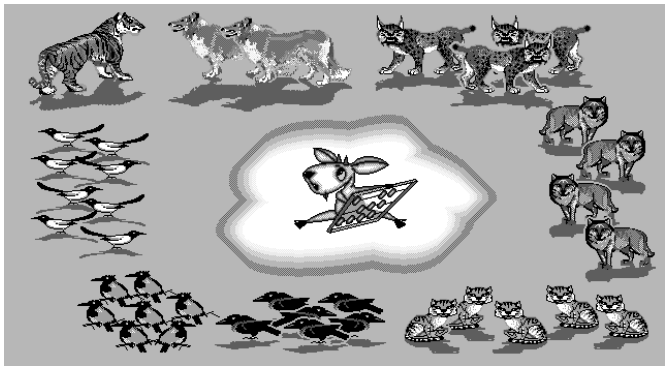
Теперь картинка открыта полностью: на большой зеленой лужайке разместились несколько групп зверюшек — один тигр, две собаки колли, три рыси... А в самом центре лужайки сидел симпатичный голубоглазый ослик и на счетах пересчитывал зверюшек.

С последней открывшейся карточкой зазвучала приятная, знакомая детям мелодия песни о том, чему учат в школе.

— Это словно поздравление с успешным решением задачи, — радостно улыбнулась Дина.

— Так оно и есть, — согласился дедушка. — Если ты добралась до конца задачи, то, конечно, заслужила поздравление. Теперь осталось только посмотреть, какую оценку ты получаешь за свою работу, Дина. Ну-ка, нажми любую клавишу.

Как только Дина нажала клавишу пробела, на экране появилось небольшое окно с интересной картинкой:



Дедушка пояснил:

— Высшее звание в нашей программе — МАСТЕР. Его заслуживает тот, кто выполнит задание без единой ошибки. Утешительное звание УЧЕНИК получает новичок, допустивший много ошибок. А вот твое звание, Дина, оказалось как раз посредине между ними.

— Но ведь у меня была только одна ошибка!

— Стремись к совершенству.

В разговор вмешался чуть-чуть обиженный Тим:

— Компьютер приглашает нас продолжить работу — нажать Enter.

— Что ж, Тим, давай продолжим. Садись за клавиатуру и нажимай клавишу выполнения.

Тима не надо было заставлять. Он тут же щелкнул по клавише. Картинка со званием, присвоенным Дине, ушла с экрана, и на нем снова появилось меню заданий.

Дедушка предложил ребятам работу.

Задание 1

Выполните по очереди задачи *Волшебных картинок* под номерами 1—13.

Дина поправила:

— Кроме задачи 3. Ведь я ее уже сделала.

— Верно, — подтвердил дедушка Фёрстов.

Тим смело заявил:

— С этим заданием справиться легко. Задачи-то совсем малышовые.

— Ты, пожалуй, не прав, Тим, — возразил дедушка. — Конечно, среди этих задач есть совсем простые, как прямой счет и обратный счет. Такие задачи, согласен, можно решать в первом классе. Но есть задачи потруднее. Когда задача тебе знакома, она кажется простой. Например, если тебе известны семь нот, то музыкальная задача про ноты для тебя пара пустяков. Но если ты не знаешь планеты Солнечной системы, то в задаче *Планеты* тебе будет трудно получить звание Мастера.

Поэтому, прежде чем перепутывать карточки, внимательно в них вглядитесь и постарайтесь догадаться, в каком порядке они расположены. Ведь самое главное в задачах программы *Волшебные картинки* — это не запомнить исходное состояние карточек, а найти закономерность в их расположении. И только после этого приступайте к перемешиванию. Когда вы справитесь с заданием 1, мы перейдем к задачам посложнее.

Дина уточнила мучивший ее вопрос:

— А картинки в заданиях разные?

— Разными могут быть не только картинки, но и поздравительные мелодии.

Тим и Дина взялись за дело дружно — то по очереди, то вместе: ничто не может сдружить лучше, чем общее интересное дело. И они скоро убедились, что дедушка оказался прав: далеко не все задания *Волшебных картинок* оказались простыми.

Так, задача про единицы измерения, хотя казалась несложной — надо было расположить карточки с длинами отрезков в порядке увеличения длин, — но сами величины были записаны в разных единицах измерения: одни — в миллиметрах, другие — в дециметрах. Очень легко запутаться, если забыть о постоянной внимательности, требующей перехода от одних единиц к другим. А задачу про

классификацию животных Тим и вовсе не решил бы, если бы ему не подсказывала Дина, которая посещала школьный кружок юных биологов.

Когда дети справились с первыми тринадцатью задачами, дедушка предложил им новую серию.

— В задачах с 14-й по 18-ю, которые имеют названия *Найди порядок 1*, *Найди порядок 2* и т.д., вам придется поломать голову. Давайте решим первую и, пожалуй, самую простую из них:

Тим, как обычно, отреагировал мгновенно:

— Я понял! Каждое следующее слово на одну букву длиннее предыдущего!

Но дедушка возразил:

— А как же пара “наши — дедушка”? “Дедушка” здесь длиннее на три буквы.

Тиму пришлось, немного подумав, уточнять свое предложение:

— Каждое следующее слово длиннее предыдущего, просто длиннее, может быть, на одну букву, а может — на несколько, но длиннее.

— Но тогда за словом “наши” можно поставить слово “родственники”: ведь оно длиннее его?

Но Тим сумел отстоять свою позицию:

— Я же сказал, КАЖДОЕ следующее слово длиннее предыдущего. Если после слова “наши” мы поставим слово “родственники”, то “дедушку” поставить будет просто некуда!

— Как так некуда, да хоть после слова “я”, ведь “дедушка” длиннее!

Но и на этот явный подвох со стороны дедушки Фёрстова Тим нашелся:

— Тогда для слова “он” в цепочке не найдется места.

Дедушке ничего не оставалось, как похвалить находчивого внука:

— Все правильно, ты молодец! Я просто проверял твои логические способности.

Задание 2

Решить задачи *Волшебных картинок* под номерами 14—18.

2.1. Изменить задачу очень легко

Разговор с внуками о возможностях изменения заданий в *Волшебных картинках* дедушка Фёрстов начал с определения:

— *Волшебные картинки* — открытая среда. Это означает, что мы легко можем изменить существующие задачи или добавить свои новые.

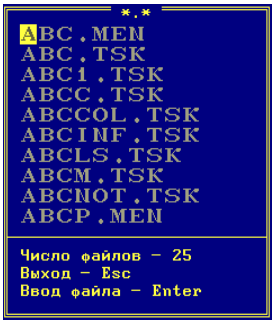
— А как это делать? — тут же спросил Тим.

— Изменить задачу совсем легко, надо просто знать ее имя на диске. Сейчас я покажу, как это делается, на примере самой простой, первой задачи *Прямой счет* с цепочкой 1-2-3-4-5-6-7-8-9-10. Продолжим ее до 20.

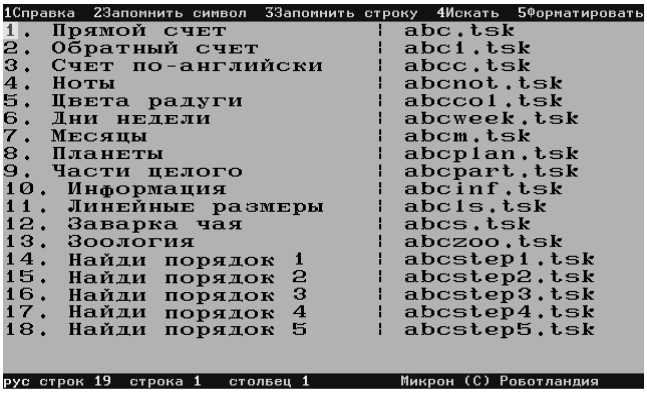
Сначала в *Мониторе* на странице *Картинки* нажимаем кнопку с *Микроном*. Затем в *Микрон* загружаем меню с задачами. Этот файл называется ABC.MEN.

— Это я помню: для загрузки файла вызываю меню *Микрона* клавишей перехода (Esc) и выбираю в этом меню раздел *Каталог*:

А затем в вызванном каталоге — списке файлов — выбираю нужный файл.



И вот, наконец, этот разыскиваемый файл — ABC.MEN — появляется на компьютерном экране:



Дедушка объяснил смысл работы, выполненной Тимом:

— Файл, содержащий меню *Волшебных картинок*, сейчас нам нужен только для того, чтобы посмотреть в нем дисковое имя первой задачи. Видите, нужное имя — ABC.TSK — написано в первой строке.

А к вам у меня, ребята, вопрос: уж если файл с меню на экране, может быть, вы расскажете мне, как этот файл устроен?

Дина решительно взялась рассказывать об устройстве файла:

— Это настолько просто, что понятно даже без объяснений. Для каждой задачи в этом файле отводится отдель-

ная строка. В первой части строки записывается текст (он появляется в экранном меню *Волшебных картинок*), а во второй части записано имя файла, в котором, вероятно, эта задача и описана. А между текстом и именем файла — разделитель, символ "|", вертикальная черточка.

Дедушке это внучкино выступление понравилось:

— Все правильно. Давайте теперь загрузим на экран файл ABC.TSK с описанием первой задачи.

Тим, уже усвоивший операции с каталогом, тоже высказался очень точно:

— Опять вызываю меню *Микрона*, отмечаю позицию *Каталог* и выбираю в нем нужный файл.

В результате действий Тима на экране высветилось содержимое файла ABC.TSK:

```
1Справка 2Запомнить символ 3Запомнить строку 4Искать 5Форматировать
Картинка digit.pic
эффект 7810
музыка sho.mus
1 2 3 4 5 6 7 8 9 10
```

Дедушка начал объяснять:

— Пока не обращайте внимание на первые три строки, а в четвертой строке...

Тим и на этот раз не смог удержаться:

— Я вижу числа, которые появляются на карточках в программе.

Дедушка посмотрел на Тима с укоризной, но, увидев смущение внука, сделал вид, что ничего не заметил:

— Заметьте, что числа первоначально появляются на карточках в том же порядке, в котором расположены в этом файле.

Дина включилась в разговор с радостной догадкой:

— Значит, чтобы увеличить число карточек, мне надо просто дописать недостающие числа в четвертой строке?

Дедушка не мог не порадоваться:

— Совершенно верно.

Однако у Тима все-таки осталось сомнение:

— Но в строке уместились только шестнадцать чисел:

```
1Справка 2Запомнить символ 3Запомнить строку 4Искать 5Форматировать
Картинка digit.pic
эффект 7810
музыка sho.mus
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
```

— Ничего страшного, — ответил дедушка. — Ты можешь продолжить цепочку чисел в следующих строчках — это допускается.

Тогда Тим быстро дописал последовательность чисел до последнего, двадцатого, включительно и предложил посмотреть:

— Вот что у меня получилось:

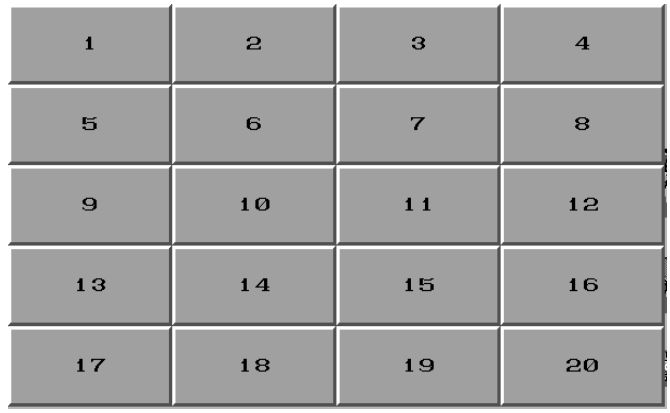
```
1Справка 2Запомнить символ 3Запомнить строку 4Искать 5Форматировать
Картинка digit.pic
эффект 7810
музыка sho.mus
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
17 18 19 20
```

Дедушка понял, что Тим ждет похвалы:

— Отлично! Теперь запиши на диск измененный файл под прежним именем и запусти *Волшебные картинки*.

Тим внимательно проделал все необходимые операции, не забывая их комментировать (как учит дедушка):

— Так. Для этого сначала надо войти в меню текстового редактора *Микрон*. Затем выбираю в меню *Микрона* позицию *Запись*, имя файла менять не надо — просто нажимаю клавишу выполнения. Теперь выхожу из *Микрона* и с той же страницы Монитора запускаю *Волшебные картинки*, то есть щелкаю по ее пиктограмме. В меню *Картинок* выбираю задачу под номером 1. Удивительно! Программа сама построила нужное число карточек: все двадцать на месте! Изменить задачу и в самом деле легко.



2.2. Анатомия *Волшебных картинок*

Согласившись с выводом Тима, дедушка Фёрстов сказал:

— Теперь мне надо рассказать вам, как устроена среда *Картинок*, и привести несколько схем для наглядности.

Подперев руками подбородки, Дина и Тим приготовились внимательно слушать дедушку.

— Каждая задача для этой программы хранится в отдельном текстовом файле, имя которого имеет расширение TSK. Все эти файлы находятся в том же каталоге, что и сама программа (она имеет дисковое имя ABC.EXE). Все это вы сможете увидеть, когда я нарисую вам схему нашего пакета *Конструктор* (мы с вами такие схемы уже рисовали):

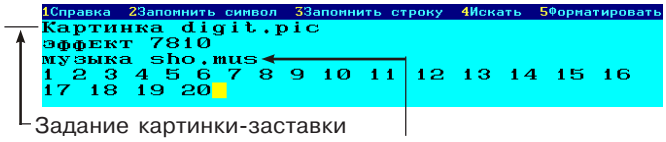
```
C:\
 KURSROB — каталог пакета Конструктор
  1
   ABC — каталог Картинок
    DO — каталог документации
     ...
     gs.bat — командный файл для запуска Графконвертера
     paint.bat — командный файл для запуска Художника
     sharm.bat — командный файл для запуска Шарманщика
     micron.bat — командный файл для запуска Микрона
     ...
     abc.exe — Волшебные картинки
     abc.pcx
     abcz.pcx
     scr.pcx                  вспомогательные файлы
     abc.lib
     ...
     abc.men — меню задач
     ...
     abc.tsk
     abc1.tsk                 задачи
     ...
     abcstep5.tsk
     ...
     abczoo.pic
     ...                      экранные заставки
     digit.pic
     ...
     bony.mus                 музыкальные заставки
     sho.mus
```

Ребята не могли не заметить, что, кроме файлов с задачами, в каталоге много всяких других файлов.

Дедушка, словно извиняясь, сказал:

— На самом деле моя схема далеко не полна, но все, что на ней изображено, подписано. Группа файлов с расширением BAT служит для запуска редакторов и графического конвертера. Для работы программы ABC.EXE нужны вспомогательные файлы, и они на схеме выделены в отдельную группу. Файлы с расширением PIC — это картинки, а с расширением MUS — музыкальные фрагменты.

Эти файлы задаются в описаниях задач, — с удовольствием вспомнила Дина.

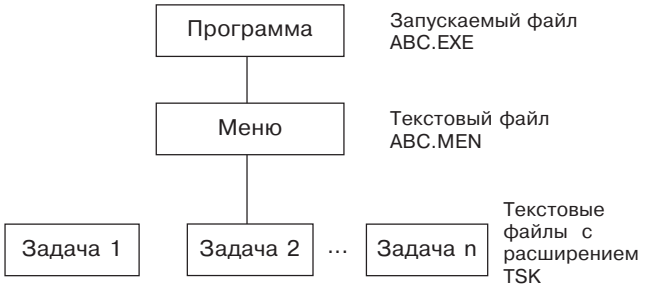


Тим добавил:

— И файл ABC.MEN мы тоже знаем — это текстовый файл с описанием меню задач.

Дедушка согласно кивнул и продолжил свой рассказ:

— А теперь я покажу вам другую схему. В ней отражается порядок работы программы ABC.EXE.



На схеме показано, что при своей работе программа ABC.EXE использует меню, которое разработчик задач описывает в текстовом файле ABC.MEN. Когда задача в меню выбрана, ее описание считывается из соответствующего текстового файла с расширением TSK.

Дина спросила:

— А расширение TSK обязательное?

— Нет, но удобно использовать одно расширение для одних и тех же объектов. Тогда эти объекты легко отличить от других. По крайней мере мы будем и дальше использовать расширение TSK для файлов с описанием задач *Волшебных картинок*. Скажу больше: обозначение файла с меню как ABC.MEN тоже не является обязательным.

Тим, поразмышляв над дедушкиными словами, удивился и задал свой вопрос:

— С задачами понятно. Их дисковые имена программа читает из файла-меню. А как она узнает имя файла, содержащее само меню?

Дедушке и на этот вопрос легко было ответить.

— Программу ABC.EXE можно запускать с параметром, который и должен быть именем файла-меню:

ABC <имя файла-меню>

Но Тим усомнился:

— Я смотрел в *Мониторе* паспорт кнопки с вызовом *Волшебных картинок*. Там видно, что программа вызывается без параметра.

Но дедушка нашел слова, чтобы убедить сомневающегося внука:

— Когда программа вызывается без параметра, она автоматически пытается найти в своем каталоге файл с фиксированным именем ABC.MEN. Иными словами, запуск программы без параметра равнозначен запуску ABC ABC.MEN.

А вообще в *Мониторе* можно создать несколько кнопок для вызовов программы ABC.EXE с разными файлами меню. В них можно сгруппировать задачи по определенным темам. Например, в *Мониторе* могут быть кнопки с вызовами:

ABC HIS.MEN — задачи по истории;

ABC ENG.MEN — задачи по английскому языку;

ABC GEO.MEN — задачи по географии.

Хотя весь этот разговор Дине был понятен, она все же призналась:

— Хорошо бы еще раз посмотреть устройство файла-меню и файла-задачи.

2.3. Устройство файла-меню

Дедушка, чуть поудобнее усевшись в кресле, с удовольствием принялся отвечать на вопрос Дины:

— Как правило, за один сеанс работы с программой приходится решать не одну, а несколько задач. Программа автоматически строит меню выбора на основе текстового файла-меню, подготовленного в текстовом редакторе. В этом файле каждая строка воспринимается как описание одной строки будущего меню на экране программы.

Меню, появляющееся на экране программы ABC.EXE

1. Прямой счет 2. Обратный счет 3. Счет по-английски 4. Ноты 5. Цвета радуги 6. Дни недели 7. Месяцы 8. Планеты 9. Части целого 10. Информация	
Ввод — Enter	
11. Линейные размеры 12. Заварка чая 13. Зоология 14. Найди порядок 1 15. Найди порядок 2 16. Найди порядок 3 17. Найди порядок 4 18. Найди порядок 5	Эта часть не видна на экране, но может появиться при прокрутке.

Текстовый файл с описанием строк меню (ABC.MEN)

1. Прямой счет	ABC.TSK
2. Обратный счет	ABC1.TSK
3. Счет по-английски	ABCC.TSK
4. Ноты	ABCNOT.TSK
5. Цвета радуги	ABCCOL.TSK
6. Дни недели	ABCWEEK.TSK
7. Месяцы	ABCM.TSK
8. Планеты	ABCPLAN.TSK
9. Части целого	ABCPART.TSK
10. Информация	ABCINF.TSK
11. Линейные размеры	ABCLS.TSK
12. Заварка чая	ABCS.TSK
13. Зоология	ABCZOO.TSK
14. Найди порядок 1	ABCSTEP1.TSK

15. Найди порядок 2 ABCSTEP2.TSK
16. Найди порядок 3 ABCSTEP3.TSK
17. Найди порядок 4 ABCSTEP4.TSK
18. Найди порядок 5 ABCSTEP5.TSK

Строки в файле-меню должны иметь следующую структуру:

<элемент выбора> <разделитель> <имя файла>

Здесь:

<элемент выбора> — любой текст, он появляется на экране в меню выбора;
<разделитель> — символ "|" (вертикальная черточка);
<имя файла> — имя файла — описания задачи.

Тим восхищенно взглянул на дедушку:

— Объяснил, словно лекцию прочитал. Но я все понял!

2.4. Устройство файла-задачи

Дедушка Фёрстов улыбнулся словам внука и заговорил об описаниях задач:

— Основное назначение файла с описанием задачи — указать объекты в том порядке, в котором их нужно отмечать на экране. Объекты записываются через один или несколько пробелов в одну или несколько строчек. Программа помещает каждый объект на отдельную карточку. Как правило, объектом является слово, но на карточку можно поместить одно или даже несколько небольших предложений или вообще любой набор символов. Для программы смысл текста не важен.

Придирчивая Дина спросила:

— Ты говоришь, дедушка, "любой набор символов"? А как быть с пробелами, ведь они отделяют объекты друг от друга?

— Очень просто. Если пробел должен входить в состав объекта, то вместо него в файле описания надо использовать символ подчеркивания. Вот он, — показал дедушка, — на одной клавише с минусом, только на верхнем регистре.

(— "На верхнем регистре" — это значит, что надо нажимать минус, надавив перед этим на клавишу Shift, — шепнул Тим на ухо сестре.

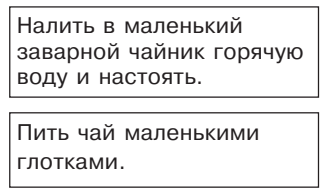
— Знаю, знаю, — так же тихонько ответила Дина.)

Дедушка Фёрстов, увлеченный рассказом об описаниях задач, не заметил этого перешептывания и продолжал:

Например, объекты:

```
Наполнить_чайник_холодной_водой.

Согреть_маленький_заварной_чайник.

Насыпать_заварки_в_маленький_заварной_чайник.

Налить_в_маленький_заварной_чайник_горячую_воду_и_настоять.

Пить_чай_маленькими_глотками.
```

будут размещены на 5 разных карточках, при этом символы "_" на экране автоматически заменятся на пробелы:



Кроме объектов, файл с описанием задачи может содержать еще и дополнительные указания программе. Такие указания записываются в виде:

<ключевое слово> <параметр>

Вот на этом листочке я распечатал на принтере список возможных указаний из файла с программной документацией.

И дедушка подвинул к ребятам лежащий около него лист бумаги с аккуратно напечатанными строчками:

Комментарий

Ключевое слово — символ ";" (точка с запятой).

Параметр — любой текст до конца строки.

Комментарий никак не влияет на работу программы и служит для записи пояснений.

"Волшебная" картинка

Ключевое слово — "Картинка:" (синонимы: "Картинка", "картинка:", "картинка").

Параметр — имя файла с полноэкранной картинкой в форматах PIC (Художник) или PCX.

Картинка "подкладывается" под карточки и появляется на экране полностью, когда задача решена.

Музыкальная заставка

Ключевое слово — "Музыка:" (синонимы: "Музыка", "музыка:", "музыка").

Параметр — имя файла с музыкальным фрагментом в формате MUS (Шарманщик).

Музыкальный фрагмент включается тогда, когда экран "очищен" от всех карточек, то есть служит сигналом окончания решения задачи.

Видеоэффект убирания карточек с экрана

Ключевое слово — "Эффект:" (синонимы: "Эффект", "эффект:", "эффект").

Параметр — *xyzt*.

Здесь:

x — код эффекта; *z* — "темп";
y — "шаг"; *t* — "ускорение".

Задание видеоэффекта "оживляет" работу программы. Используя специальную таблицу, можно выбрать *x* (код эффекта) и экспериментально подобрать значения *y*, *z*, *t*.

Поля "темп" (*z*) и "ускорение" (*t*) в коде эффекта могут быть как цифрами, так и любыми латинскими буквами. "Темп" — это задержка между "шагами" вывода на экран. "Ускорение" — это скорость показа одного "шага".

Когда для кодирования используются латинские буквы, то следует считать их как бы продолжением цифр после 9. Буква "*a*" — это как бы цифра со значением 10, буква "*b*" — цифра со значением 11 и так далее.

Дина и Тим слушали, удивляясь необычным обозначениям.

Дедушка решил не пугать их обилием информации:

— Пусть вас не смущают этот листок и большущая таблица. Вы вообще можете не пользоваться эффектами и задавать в описаниях только последовательности объектов на карточках. Например, для работы первой задачи *Прямой счет* достаточно написать в файле ABC.TSK:

```
1Справка  2Запомнить символ  3Запомнить строку  4Искать  5Форматировать
1 2 3 4 5 6 7 8 9 10
```

— И это будет работать? — не поверила Дина.

— Конечно!

— А какая картинка будет тогда подложена под карточки?

— Если картинка не задана в описании, то программа автоматически использует "свою" картинку с именем ABC.PCX.

Тим поинтересовался:

— А что на ней нарисовано?

— На ней нарисован старинный замок.

Но Дина не могла успокоиться — ничего нет в описании задачи, а она работает:

— А как же музыка и эффекты?

Дедушка продолжал убеждать ее, что ничего страшного не произойдет:

— Вот музыки точно не будет, и карточки будут исчезать с экрана сразу, без всяких выкрутасов, но задача работать будет!

— А если я все же хочу использовать эти "выкрутасы"?

— Ну тогда разбирайся в таблице, экспериментируй и пиши описание вроде того, которое написано для первой задачи:

```
1Справка  2Запомнить символ  3Запомнить строку  4Искать  5Форматироват
Картинка digit.pic
эффект 7810
музыка sho.mus
1 2 3 4 5 6 7 8 9 10
```

2.5. Задача, которую придумала Дина

Вдруг Дина неожиданно воскликнула:

— Ой, а мне кажется, что я придумала подходящую задачу для *Волшебных картинок*!

Дедушка сразу же предложил:

— Ну, Дина, рассказывай!

— Ты же знаешь, дедушка, как мне трудно дается запоминание всяких исторических дат. Вот я и решила помочь себе при помощи нашей программы *Волшебные картинки*.

— Ты попала в точку! — Было видно, что дедушке идея понравилась. — Построить задачу, основанную на датах, в *Волшебных картинках* действительно очень легко, ведь задания этой программы всегда основываются на каком-нибудь упорядоченном множестве элементов. А уж с датами всегда полный порядок! Какие конкретно даты ты хочешь использовать?

— Даты рождения русских писателей.

— Очень хорошо, — поддержал дедушка.

Дина продолжала:

— Для своих задач я хочу создать на диске отдельное меню и для этого поставить в *Мониторе* на странице *Картинок* новую кнопку. Как мне сделать новое меню?

Тут подсказал Тим:

— Разве это не ясно? Конечно, в *Микроне*.

Вдохновленная идеей создания задачи, Дина не обратила внимания на тон, которым брат сделал замечание, и стала подробно пояснять дальнейшие действия:

— Хорошо, вызываю его на экран. — И она щелкнула по пиктограмме с названием *Микрон*. — Теперь записываю в нем строку для моей задачи:

```
1Справка  2Запомнить символ  3Запомнить строку  4Искать  5Форматировать
Русские писатели                    !dina1.tsk
```

— Ничего, что мое меню будет состоять из одного "блюда"? — спросила Дина, повернувшись к дедушке.

— Программа работать будет.

— Я назову задачу... — Дина чуть задумалась. — Задача на диске будет иметь имя DINA1.TSK. Это потому что она моя и первая.

Тим тут же сделал заявление:

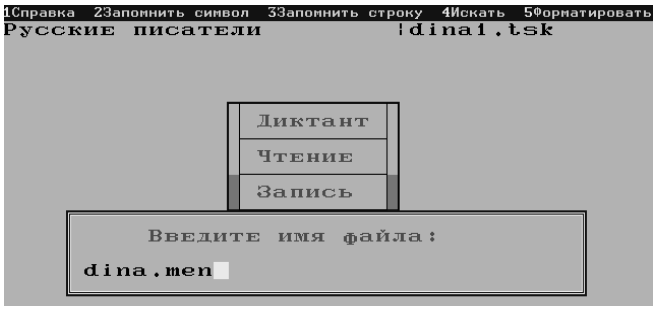
— Тогда мои задачи будут называться TIM1.TSK, TIM2.TSK и так далее.

Дедушка согласился с обоими выступлениями:

— Нет возражений. Теперь, Дина, запиши этот файл на диск.

Дина так и сделала:

— Клавишей перехода вызываю меню *Микрона*, выбираю в нем позицию *Запись*. Дальше записываю имя DINA.MEN — так на диске будет называться мое меню.



— Теперь надо поставить новую кнопку в *Мониторе*.

Но дедушка предложил более разумный прием:

— Раз уж ты сейчас находишься в *Микроне*, то напиши сначала файл с задачей, а пиктограмму поставишь потом.

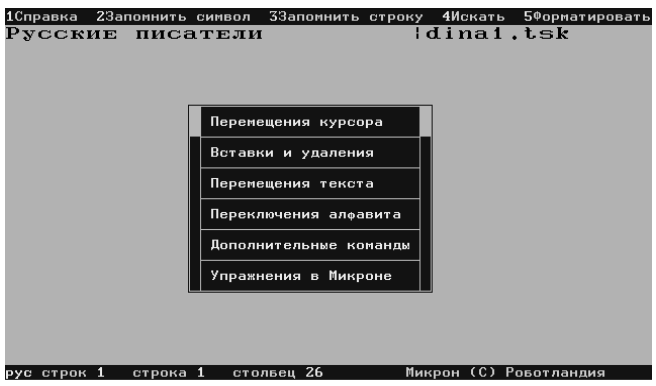
Дина согласилась:

— Правда, так будет быстрее. А как мне стереть с экрана текст меню?

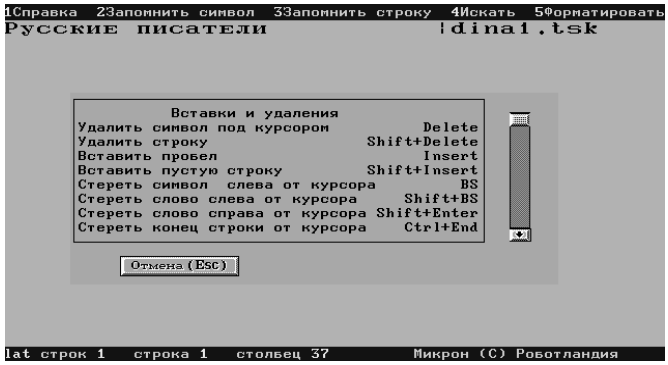
Тим снова проявил недовольство сестрой:

— Ты забыла, как работать с *Микроном*? Нажми клавишу F1.

— Пожалуй, ты прав, — сказала Дина вполне миролюбиво. — Клавиша F1, конечно, не "очистит" *Микрон*, но зато вызовет на экран меню для выбора раздела подсказки:

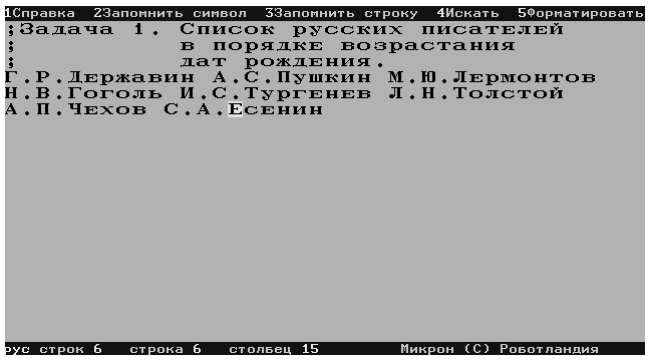


— Я хочу удалить текст с экрана, поэтому мне нужно выбрать позицию *Вставки и удаления*:



Подсказка и в самом деле оказалась полезной. Дина покачала головой, словно укоряя себя за забывчивость:

— Да, теперь я вспомнила: весь текст с экрана стирается сложным аккордом Ctrl + Shift + Delete. Теперь я запишу свою задачу:



— Молодец, Дина, порядок карточек в твоей задаче указан правильно.

Дина была довольна результатом. Дедушка спросил:

— Что ты сделаешь теперь?

— Мне осталось только сохранить этот текст. Я запишу его под именем DINA1.TSK и, нажав аккорд Ctrl + Break, выйду из *Микрона*.

Дедушка отметил завершение важного этапа конструирования задачи в *Волшебных картинках*:

— Дина сделала все, для чего нам мог быть полезен *Микрон*. Теперь самое время поставить на экране *Монитора* новую пиктограмму.

— Можно теперь я? — спросил замучившийся ожиданием Тим.

Дедушка посмотрел на Дину. Она прочитала в его взгляде то, для чего слова были вовсе не нужны. Она подвинула клавиатуру поближе к брату и просто сказала:

— Пожалуйста.

Мгновенно преобразившийся Тим тут же принялся за дело, приговаривая примерно так же, как Дина:

— Вызываю на экран меню операций *Монитора*... и выбираю в нем позицию *Добавить*. Теперь нам остается только заполнить паспорт кнопки.

— Что ж, Тим, действуй смелее!

Тима нельзя было упрекнуть в отсутствии смелости, однако его решительности хватило только на написание имени кнопки: в этом окошке он написал название, предложенное для задачи Диной, — *Русские писатели*. Но дальше он остановился и посмотрел на дедушку:

— Подскажи, дедушка, что надо написать в окошке *Программа*.

Дедушка потянулся к схеме пакета *Конструктор*, которую он недавно обсуждал с детьми:

— Имя программы *Волшебные картинки*, как вы знаете, — ABC.EXE. Однако, для того чтобы *Конструктор* мог ее разыскать, в окне *Программа* надо указать не только ее собственное имя, но и весь путь до нее. Поэтому, глядя на схему пакета, в этом окне следует писать:

.\KURS\1\ABC\ABC DINA.MEN

Кстати, собственное имя программы (здесь речь идет о программе ABC.EXE) можно писать и без расширения EXE. Надеюсь, вы помните, что мы совсем недавно говорили о том месте, где при вызове *Волшебных картинок* следует указывать имя файла-меню. Если бы речь шла о файле-меню с именем ABC.MEN, то такое имя можно и не писать. Но уж если Дина решила воспользоваться файлом-меню, которому она дала имя DINA.MEN, то именно это имя и надо поместить сразу же вслед за ABC.EXE (через пробел).

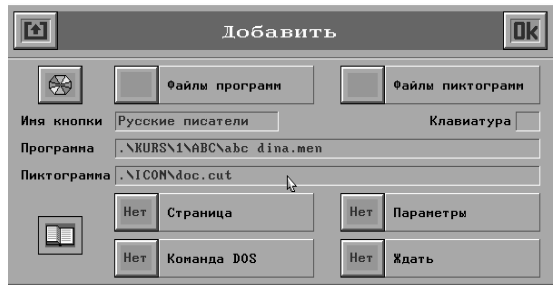
— Все ясно, — сказал Тим, без ошибок заполнил окошко *Программа* и сразу же спросил: — А как быть с окошком *Пиктограмма*?

Дедушке и здесь пришлось давать пояснения:

— Можно было бы придумать и нарисовать пиктограмму, подходящую для нашей задачи, но мне не хочется, чтобы вы сейчас тратили время на эту знакомую вам работу. Поэтому я предлагаю в этом окошке написать имя пиктограммы DOC.CUT, которая часто используется в программах пакета *Волшебные картинки*. Добраться до этой пиктограммы можно так: .\ICON\DOC.CUT. Именно этот путь и следует тебе, Тим, записать в паспорте кнопки.

И еще об одной возможности заполнить эти окна я хочу вам сказать: кнопка *Файлы программ* открывает путь к каталогам программ, а кнопка *Файлы пиктограмм* — к каталогам пиктограмм. А это означает, что вы можете, заполняя окошки паспорта, не набирать трудную запись пути на клавиатуре, а выбрать нужные имена в каталоге. Тогда выбранное имя попадет в окно с требуемым путем.

— Хорошо, — сказал Тим. — Только, когда я буду составлять свою задачу для *Волшебных картинок*, я все-таки нарисую свою собственную пиктограмму.



Заполнив все окна панели *Добавить* по дедушкиным рекомендациям, Тим хотел пригласить Дину полюбоваться на его работу, однако Дина и без того очень внимательно следила за действиями брата.

Тим проверил вызов своей задачи из *Монитора* и убедился, что все работает нормально.

Задание 3

Повторить все действия Дины и Тима по созданию новой задачи для программы *Волшебные картинки*.

2.6. Графический конвертер

После перерыва, который дедушка использовал для чаепития, Тим заявил:

— Хотелось бы для моей задачи использовать не "автоматическую" картинку со старинным замком, а какую-нибудь другую.

— Что ж, — предложил дедушка, — ты можешь выбрать один из двух способов — подобрать другую картинку или нарисовать свою собственную.

Тима больше привлекала вторая возможность, поэтому он поинтересовался:

— Рисовать надо в графическом редакторе *Художник*?

Дедушка подтвердил:

— Это лучше всего. В *Художнике* получаются графические файлы формата PIC. *Волшебные картинки* понимают и файлы в формате PCX. Вообще говоря, можно рисовать картинки в любом редакторе, который позволяет получать PIC- или PCX-файлы, но если этот редактор — не *Художник*, то перед рисованием его надо настроить на тот режим экрана, в котором работают *Волшебные картинки*, то есть на 16-цветную палитру и разрешение 640 × 350.

Дина попросила:

— Напомни, дедушка, что такое формат графического файла.

— Картинки хранятся на диске в кодированном виде. Способ кодирования называют **форматом** графического файла. Файлы одного формата имеют одинаковое расширение имени. Обычно по расширению имени графического файла называют и сам формат. У всех файлов в формате PIC расширение имени PIC, а у всех файлов в формате PCX расширение PCX. Существуют и другие графические форматы, отличные от PIC и PCX.

Тим тоже не остался в стороне от вопросов:

— А что ты нам скажешь по поводу палитры и разрешения экрана?

— **Палитра** — это набор тех цветов, которыми можно рисовать на экране.

Экран не похож на обычный лист бумаги для рисования. Раскрашивать на нем можно только отдельные точки, расположенные на одинаковых расстояниях друг от друга. Число доступных для рисования точек на экране и называют **разрешением** экрана. Разрешение обычно записывают в виде двух чисел через знак умножения. Первое число показывает, сколько точек умещается в одной строке, а второе — число строк на экране.

Дина сообразила, пощелкав предварительно на калькуляторе:

— Значит, разрешение 640 × 350 означает, что на экране в этом режиме помещаются 224 тысячи точек?

— Да. И надо сказать, что это не так уж и много. Современные дисплеи могут работать в разрешении 1024 × 768, т.е. в этом случае на экране умещается более миллиона графических точек, или, как еще говорят, **пикселей**.

Тим уточнил вопрос:

— Чем больше точек на экране, тем ведь лучше?

— Это точно. Да, чем выше разрешение, тем качественнее картинка и глаза устают гораздо меньше. Хотя на качество изображения еще сильно влияет частота, с которой дисплей перерисовывает свой экран. Чем она выше, тем больше монитор компьютера похож на обычную книгу.

— Спасибо, дедушка, — не забыл поблагодарить Тим, — но ты сам сказал, что у нас мало времени и мне сейчас некогда рисовать новую картинку. Как бы мне подобрать готовую?

Дедушка воспользовался этим интересным (и вполне естественным!) вопросом, чтобы рассказать еще об одной программе:

— Удобно взять для этого программу *Пик* — это роботландская "смотрелка" и "преобразовалка" графических файлов.

Тим высказался по этому поводу:

— Подозреваю, что в *Мониторе* для этой программы заготовлена кнопка с надписью *Графконвертер*.

— Точно! Вызывай его на экран.

Дина заметила:

— Программа выглядит на экране в каких-то совсем необычных тонах.

— Это потому, что картинка со старинным замком (она загружена сейчас на экран) имеет нестандартную палитру. Не обращайте пока на это внимание! Давайте лучше я объясню вам назначение кнопок управления конвертером.

Ребята внимательно устремили взоры на экран:

— В верхней полоске расположены кнопки основного меню:

 — информация о разработчиках,

 — переход на дополнительную страницу, (вырезание фрагментов),

 — запись (сохранение) картинок,

 — удаление картинок с диска,

 — печать картинок,

 — справочная система,

 — выход из программы.

Рабочая область слева на экране содержит каталог для выбора картинок, а сами картинки отображаются в небольшом окошке. Они уменьшены в два раза. Посмотреть картинку в обычных размерах можно, нажав на клавишу Tab.

Тим, уже привыкший работать с каталогами, спросил:

— Чтобы посмотреть картинку, нужно просто отметить ее в каталоге?

Дедушка кивнул ему. Тим и Дина просмотрели несколько картинок в каталоге и наиболее подходящей для своих целей нашли картинку, на которой была изображена раскрытая книга с рисунками к пушкинским сказкам. Они дружно решили, сказав дедушке:

— Пожалуй, для нашей задачи мы возьмем картинку SKAZKI.PCX:

Дедушка предупредил:

— Но тогда ты должен вернуться в *Микрон* и изменить файл DINA1.TSK.

— Понятно, — согласился Тим. — Теперь он будет выглядеть так:

```
1Справка 2Запомнить символ 3Запомнить строку 4Искать 5Форматировать
; Задача 1. Список русских писателей
;           в порядке возрастания дат
;           рождения
Картинка: skazki.pcx
Г.Р.Державин А.С.Пушкин М.Ю.Лермонтов
Н.В.Гоголь И.С.Тургенев Л.Н.Толстой
А.П.Чехов С.А.Есенин
```

2.8. Шарманщик

Хотя ребята довольно быстро продвигались вперед, разбираясь с *Волшебными картинками*, все же в вопросе, который Дина подготовила дедушке, послышалось некоторое огорчение:

— Дедушка, в тех задачах *Волшебных картинок*, которые мы успели посмотреть, была каждый раз слышна мелодия в конце задачи. И даже разные мелодии: в каждой задаче — своя. Да, и в описаниях задач мы видели строку со словом "музыка". А ты так ничего и не сказал: как можно включить музыку в задачу?

Тим к этому вопросу тоже присоединился, но полушутя высказал свое мнение:

— Предполагаю, что опять это можно сделать двумя способами: выбрать из готовых мелодий или написать свою собственную.

Но дедушка отнесся совершенно серьезно к этому замечанию:

— Я вас и не собирался обманывать с музыкой в *Волшебных картинках*. Просто, как вы сами видели, пока до этого руки не доходили. Зато сейчас благодаря вопросу Дины мы и поговорим, правда совсем немного, о записи и воспроизведении музыкальной информации на компьютере. А что касается замечания Тима, то он совершенно прав. И я вам покажу, что и записать мелодию, и сыграть ее на компьютере можно в *Шарманщике* — так называется простой учебный редактор музыкальной информации.

Услышав слово "редактор", Дина верно рассудила:

— Раз уж это редактор, значит, в нем можно и читать, и писать, и искать в каталоге файлы с музыкальной информацией.

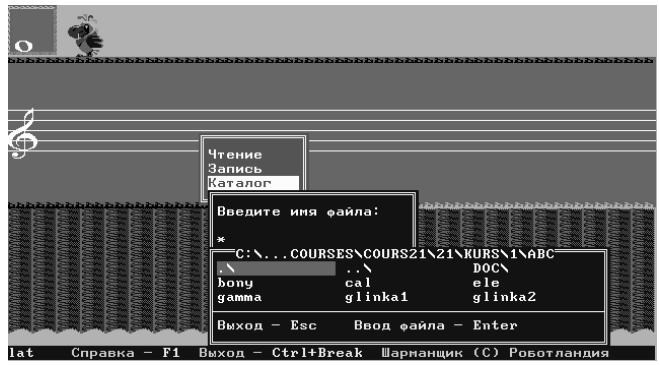
А Тим, воспользовавшись тем, что сидел за клавиатурой, сказал:

— Для начала я просто выберу мелодию из уже готовых. Та-ак. Вызываю на экран музыкальный редактор.

И он смело щелкнул по пиктограмме *Шарманщика*.

Дедушка, понимая, что он вправе пользоваться аналогией с текстовым и графическим редакторами, сказал:

— Дисковое меню в *Шарманщике* вызывается клавишей перехода — Esc. Надо войти в раздел *Каталог* и в открывшемся меню музыкальных файлов выбрать нужный.



Хоть Тим и без труда выполнил эти знакомые еще по *Микрону* операции, он в смущении остановился:

— Но я не смогу определить мелодию по имени файла!

Дедушка порекомендовал:

— Придется загружать музыкальные фрагменты в *Шарманщик* и прослушивать. Или ты найдешь то, что тебе понравится, или придется набирать музыкальный фрагмент из альбома с нотами.

Тим и Дина некоторое время прослушивали в *Шарманщике* мелодии, записанные в разных музыкальных файлах, потом остановили свой выбор на мелодии GLINKA1.MUS.

Дина удовлетворенно сказала:

— Ну вот, теперь наша задача будет с музыкой:

```
1Справка 2Запомнить символ 3Запомнить строку 4Искать 5Форматировать
; Задача 1. Список русских писателей
;           в порядке возрастания дат
;           рождения.
Картинка: skazki.pcx
Музыка: glinka1.mus
Г.Р.Державин А.С.Пушкин М.Ю.Лермонтов
Н.В.Гоголь И.С.Тургенев Л.Н.Толстой
А.П.Чехов С.А.Есенин
```

2.9. Доработка задачи Дины и Тима

После того как в задаче про русских писателей зазвучала музыка, Тим сказал:

— Ну, теперь я, кажется, готов к тому, чтобы ввести в нашу задачу какое-нибудь эффектное исчезновение карточек с экрана. Скучно, когда они стираются в одно мгновение. Пусть постепенно "растворяются" в воздухе.

Дедушка развернул знакомую детям таблицу видеоэффектов:

— Для этой цели вам надо будет использовать команду *Эффект* в описании задачи. Я думаю, что для ваших целей может пригодиться эффект с кодом *и* из этой большой таблицы, которую я вам уже показывал. Надо подобрать другие компоненты для параметра *xyzt*.

Дина проверила, правильно ли она поняла дедушку:

— Значит, код эффекта для *x* мы выбрали — это *и*. Остается подобрать значения величин *y*, *z*, *t*. Напомни, пожалуйста, что означает величина *y*?

— Эффект с кодом *и* рисует на карточке подобие шахматной доски. Белые поля прозрачны с самого начала, а темные постепенно заменяются картинкой. Величина *y* задает размер шахматной клетки, а величины *z* и *t* отвечают за скорость "растворения" клетки, как Тим говорит, в воздухе.

После небольших совместных и поочередных экспериментов Дина и Тим подобрали значения для всех составляющих параметра видеоэффекта, и теперь файл с описанием задачи про русских писателей стал выглядеть так:

```
1Справка 2Запомнить символ 3Запомнить строку 4Искать 5Форматировать
;Задача 1. Список русских писателей
;          в порядке возрастания дат
;          рождения
Картинка: skazki.pcx
Музыка: glinka1.mus
Эффект: u810
Г.Р.Державин А.С.Пушкин М.Ю.Лермонтов
Н.В.Гоголь И.С.Тургенев Л.Н.Толстой
А.П.Чехов С.А.Есенин
```

Задание 4

Продолжить меню DINA.MEN новыми задачами из разных областей знаний.

3. Веселые картинки

И Дина, и Тим отлично понимали, что если на экране *Монитора* расположены рядом две кнопки — *Волшебные картинки* и *Веселые картинки*, и речь на занятиях все время идет только об одной из них, то рано или поздно наступит черед знакомства и с *Веселыми картинками* тоже. И ребята договорились между собой, что не будут торопить дедушку, пока он сам не возьмется рассказать об этой "таинственной" программе. И вот этот день настал.

Дедушка сказал о новой программе совсем просто:

— Вижу по вашим глазам, что вам хочется узнать про *Веселые картинки*.

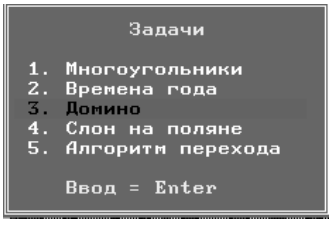
И, вопреки своим планам, дети закивали головами.

— Вы увидите сейчас, что эта программа очень похожа на *Волшебные картинки*, но ее задачи выстраивают не текстовые упорядоченные цепочки, а графические. Как и раньше, я предлагаю сначала порешать готовые задачи, а потом придумать несколько своих собственных.

3.1. Решение задач

Такие дедушкины слова нельзя было воспринять иначе, как предложение щелкнуть по кнопке *Веселые картинки*. Дина, усевшаяся ближе к коврику с мышкой, так и сделала:

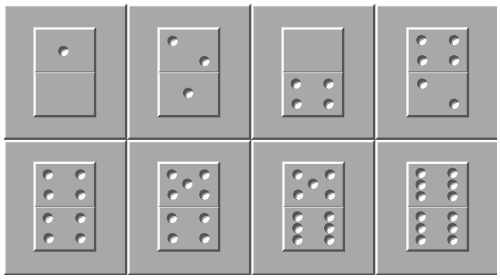
— Запускаю программу и вижу на экране такое же меню, как и в *Волшебных картинках*, но список задач здесь другой. Покороче. Я выберу задачу про домино, можно? — спросила она дедушку.



Разрешение было получено.

Дедушка пояснил:

— Видите: как и в *Волшебных картинках*, экран состоит из карточек, но на них — не слова, а картинки. Кстати, карточки в этой программе роботландцы называют кубиками.



— Я поняла! — воскликнула Дина. — Косточки домино построены по возрастанию общего числа точек на них. Это задача для первоклашек.

Дедушка сразу спросил:

— А вы могли бы написать аналогичную задачу для *Волшебных картинок*?

Дина, не колеблясь, ответила:

— Конечно! Если исключить всякие там эффекты, музыку и даже картинку, то задача имела бы такое описание (она быстро написала карандашом на листочке бумаги):

Домино в текстовом виде							
1	3	4	6	8	9	11	12

Дедушка согласился с простотой решения:

— Это совсем похоже на задачу *Прямой счет* из *Волшебных картинок*, только числа здесь идут не подряд, хотя и строго по возрастанию. А теперь давайте посмотрим, как такая задача решается в *Веселых картинках*.

Дина нажала на пробел, кубики с косточками домино перемешались на экране, и она приступила к решению.

Задание 5

Решить все задачи *Веселых картинок*.

3.2. Анатомия *Веселых картинок*

Дине было понятно, что произошло на экране, и она стала делать пояснения вместо дедушки:

— Задача решается просто. Только кубики не исчезают с экрана, а перемещаются на "свои" первоначальные места.

Но она тут же остановилась и спросила:

— Дедушка, а как с "анатомией" этой программы? Ты, кажется, так называл устройство файлов меню и описания задач в *Волшебных картинках*? Она отличается от *Волшебных картинок*?

— Очень немного. Программа называется на диске ABCP.EXE ("Добавилась одна буковка в имени", — отметил Тим), и она может запускаться с параметром. Параметром служит имя файла-меню. Если параметр не задан, то программа ищет на диске меню с именем ABCP.MEN. Другими словами, запуск без параметра равнозначен запуску ABCP ABCP.MEN.

А что касается самого меню задач, то оно устроено точно так же, как и в *Волшебных картинках*.

Тим сделал вывод:

— Значит, все отличие состоит в устройстве файла с описанием задачи?

— Верно. Во-первых, имена файлов с задачами имеют расширение TS, а не TSK. Это принятое (но не обязательное!) обозначение позволяет сразу отличить эти файлы по названию от других файлов на диске, в том числе и от файлов-задач *Волшебных картинок*. Ну и, конечно, описание задач устроено немного по-другому. Давайте загрузим в *Микрон* описание задачи про домино и посмотрим, как оно выглядит.

Дина пояснила, как она намерена выполнять дедушкину рекомендацию:

— Сначала я загружу в *Микрон* файл-меню ABCP.MEN, для того чтобы отыскать в нем дисковое имя задачи:

```
1Справка 2Запомнить символ 3Запомнить строку 4Искать 5Форматировать
1. Многоугольники     ¦ abcp1.ts
2. Времена года       ¦ abcp2.ts
3. Домино             ¦ abcp3.ts
4. Слон на поляне     ¦ abcp4.ts
5. Алгоритм перехода  ¦ abcp5.ts
```

Вижу: дисковое имя задачи *Домино* — это ABCP3.TS. Теперь загружаю в *Микрон* этот файл:

```
1Справка 2Запомнить символ 3Запомнить строку 4Искать 5Форматировать
картинка abcp3.pic
эффект f422
музыка bony.mus
кубики    2 4 8
```

Я вижу только одну новую строку, в ней — ключевое слово *Кубики*.

— Главное отличие здесь в картинке, — сказал дедушка. — На ней должны быть нарисованы как сами кубики, так и их содержимое.

Но на этот раз Дине не все было ясно:

— А как же программа узнает, сколько кубиков на картинке и как они расположены по строкам и столбцам?

— Именно на этот вопрос и отвечает та новая строка, которую ты только что отметила. Указание `Кубики: 2 4 8` означает, что картинка будет разбита на 2 полоски по 4 кубика в каждой и все 8 кубиков участвуют в перемешивании. В общем случае это указание имеет вид:

Ключевое слово — `Кубики`: (синонимы: Кубики, кубики:, кубики).

Параметры:

<число строк> <число столбцов>
<число перемешиваемых кубиков>

Пример: указание `Кубики 2 4 7` предполагает разбиение картинки на 2 строки и 4 столбца. В этом случае последний, восьмой, кубик не участвует в перемешивании, так как третий параметр — 7, а число кубиков на экране — 8.

Число строк и столбцов не может быть произвольным. Число строк должно быть делителем числа 350, а число столбцов должно быть делителем числа 80. Вот посмотрите на таблицу. В ее клетках написано общее число кубиков, которое может быть на экране нашей программы при допустимых количествах строк и столбцов.

Число строк	Число столбцов				
	2	4	5	8	10
2	4	8	10	16	20
5	10	20	25	40	50
7	14	28	35	56	70
10	20	40	50	80	100

Тим, подумав, с грустью сказал:

— Мне кажется, что нарисовать картинку с точным расположением всех кубиков невероятно сложно.

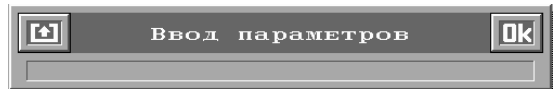
— Ты прав. Но в *Веселых картинках* есть специальное средство: *Генератор кубиков*. Эта программа делает заготовку для будущей картинки, расчерчивая ее на кубики нужным образом.

Дина проявила интерес к *Генератору кубиков*:

— Ой, как здорово! А как запускать эту программу?

— В *Мониторе*, на странице *Картинки*, есть пиктограмма, которая так и названа — *Генератор кубиков*. Попробую теперь ее нажать... Смотрите, на экране появилось окно:

— Это *Монитор* просит тебя ввести параметры для запускаемой программы.



— И что я должна написать?

— Нужно указать четыре параметра через пробелы:

<имя файла-картинки> <число строк> <число столбцов> <тип кубика>

(рассказывая о способах указания параметров и их значениях, дедушке пришлось делать записи фломастером на большом бумажном листе).

Вот что они означают:

<имя файла-картинки> — это дисковое имя будущей картинки. Формат картинки определяется по расширению: PIC — для формата PIC, PCX — для формата PCX;

<число строк> — число строк на экране, соответствует аналогичному параметру в строке *Кубики* программы *Веселая картинка*;

<число столбцов> — число столбцов на экране, соответствует аналогичному параметру в указании *Кубики* программы *Веселые картинки*;

<тип кубика> — число 0 или 1, задает вид кубика на экране.

Тим не мог не спросить:

— А чем отличаются кубики двух типов — 0 и 1?

— Кубик типа 0 — "плоский", а типа 1 — "объемный". Плоские кубики на экране отделяются один от другого обычными отрезками прямой, а у объемных эти отрезки изображаются с тенью и выглядят поэтому объемными. Обычно указывается тип 1, а для задач, в которых целая картинка должна "разрезаться" на части, а потом при решении — восстанавливаться, указывается тип 0. В нашем пакете *Веселых картинок* "плоский" тип кубиков имеет задача *Слон на полянке*.

— Хорошо, — сказала Дина после обсуждения типов кубиков, — щелкаю мышкой в поле ввода и записываю в нем...

Дина задумалась:

— Какие бы параметры написать?

Дедушка посоветовал:

— Напиши сначала имя картинки. Советую тебе написать 1.PIC: на этой картинке, как ты увидишь, нарисованы границы ячеек-кубиков, в которые будет удобно вырисовывать изображения. Если ты хочешь нарисовать восемь кубиков в две строки, то тогда ясно, что следующие два параметра будут...

Тим не смог удержаться и, не дав договорить дедушке, радостно выкрикнул:

— 2 и 4.

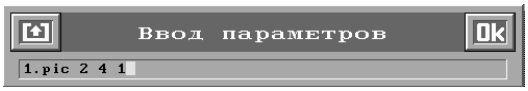
Дедушка своей рукой поворошил шевелюру внука, и Тим понял — дедушка не сердится, хотя Тим поступил неправильно ("Ответил правильно, а поступил — непра-

вильно, — про себя отметил Тим; еще одна мысль промелькнула у него, хотя и ее он тоже не высказал вслух: — А все-таки хороший у нас дедушка!").

Тем временем дедушка Фёрстов продолжал свой рассказ, обращаясь к Дине:

— А последний параметр следует писать в зависимости от того, какую картинку ты задумала делать.

— Я еще не решила, что я буду рисовать. Мы сейчас с Тимом посоветуемся. Но мне хочется, чтобы это была картинка, где разные отдельные кубики надо было бы переставлять, разгадывая их порядок. Как в *Домино*. Для такой картинки тип кубиков, как ты сказал, должен быть 1.



Значит, теперь я нажимаю клавишу выполнения и экранную кнопку "*Ok*"... На экране промелькнула какая-то картинка, но я не успела ее рассмотреть. А хотелось бы. Ведь ты уже дважды, рассказывая о картинке в задании для *Веселых картинок*, употребил слово "будущая". Потому-то мне и хочется знать, что же получилось?

Дедушка утешил ее тем, что и не надо было стараться ее разглядеть:

— Эту картинку (ты указала ее имя — 1.PIC) сделал *Генератор кубиков*. Точнее, ее следовало бы называть даже не картинкой, а только заготовкой для картинки, потому что *Генератор кубиков* создает только сетку из заданного количества клеток (в том примере, что Дина написала в задании *Генератору*, это $2 \times 4 = 8$), а в каждую из них еще надо будет вырисовать сейчас изображения. Заготовку можно посмотреть, вызвав *Графконвертер*. А можно сразу же запустить *Художника*, вызвать в нем эту заготовку и начать рисование.

3.3. Новая задача

Тим решил предложить идею новой задачи. Поэтому он торопился и ему не хотелось тратить время на просмотр заготовки для картинки ("Все равно ее будет видно, когда начнем рисование изображений", — подумал он).

— Я бы из двух возможностей выбрал вызов *Художника*, — сказал он. — У меня появилась идея для новой задачи.

Дедушка попросил его высказаться:

— Расскажи!

— На восьми кубиках (как предложила Дина) будут нарисованы: черепаха, ежик, петух, человек, гепард, автомобиль, самолет и ракета.

— И по какому правилу выстроена твоя цепочка? — спросила сестра.

— В этом и состоит "изюминка" задачи. Сначала нужно догадаться, а уж потом перемешивать кубики.

Дедушка оценил:

— Хорошая задача, Тим. Кажется, я "раскусил" порядок в придуманной тобой цепочке: объекты расположены по возрастанию максимально возможной скорости их передвижения.

— Точно!

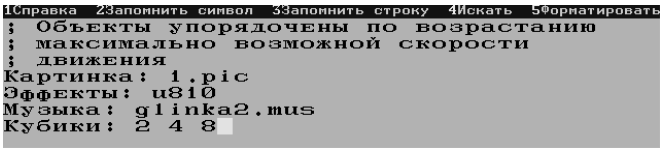
Разговор о последовательности объектов на картинке заставил Тима изменить мнение о порядке предстоящих операций:

— Давайте картинку мы нарисуем потом, тем более что тут придется рисовать не единую картинку, а целых восемь отдельных рисунков, по одному на каждый кубик. А сначала лучше написать файл с описанием задачи.

Идею оценила и Дина, которая даже предложила:

— Такую задачу надо назвать TIM1.TS.

И поскольку автор идеи возражать не стал, то объединенными усилиями детей на экране быстро появилось созданное в *Микроне* описание:



Дина пояснила:

— Видеоэффект я перекопировала из своей задачи для *Волшебных картинок*, а в качестве музыкального фрагмента использовала файл GLINKA2.MUS — в нем звучит "Попутная песня" М.И. Глинки: уж очень она подходит для задачи про скорость.

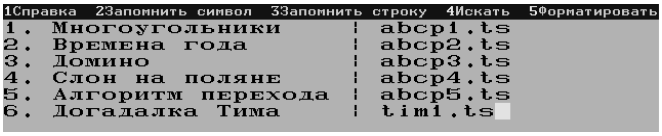
Дедушке понравилась дружная работа над описанием. Он напомнил детям о заключительном этапе конструирования задачи:

— Теперь надо вписать эту задачу в меню *Веселых картинок*.

Тим сказал:

— Это легко сделать в *Микроне*. Ведь это наверняка делается так же, как в *Волшебных картинках*.

И, выпросив на минуту клавиатуру у сестры, он быстро отредактировал файл ABCP.MEN:



3.4. *Художник*

Дедушка сказал:

— Теперь в задаче не хватает самого главного — картинки. Придется засучить рукава и вызывать *Художника*.

Тим воспринял эту идею без должного воодушевления:

— Редактор я, конечно, вызову, но из меня художник, как из гуся музыкант...

— Ты можешь не рисовать сам, а подобрать персонажей из готовой коллекции. Она состоит из файлов с расширением SPR. В *Художнике* такие файлы-картинки называются **объектами**. Объекты *Художника* можно легко расставлять на экране поверх готового рисунка.

Тут Тим повеселел:

— Вот это мне подходит! Вызываю *Художника*. Выбираю в его главном меню раздел *Файлы*, в подменю — раздел *Картинка*, в новом подменю — раздел *Каталог*, а в каталоге — файл 1.PIC. Готово! Картинка с пустыми кубиками на экране.

Дедушка настаивал на продолжении:

— Теперь проделай то же самое, но открой каталог с объектами.

Тим выполнил рекомендацию:

— В главном меню выбираю раздел *Файлы*, в подменю — раздел *Объекты*, в новом подменю — раздел *Каталог*:

— Первым в списке стоит файл CAR. Когда я просматривал, как работает *Графконвертер*, то видел в этом файле изображение автомобиля. Вот его-то я и выберу! Готово! Что я вижу: с экрана все пропало, кроме моих кубиков, и появился большой прямоугольный курсор.

Дедушка в этом не нашел ничего удивительного:

— Этот курсор — как раз размером с твой автомобиль или черепаху. Редактор *Художник* предлагает тем самым выбрать место для расположения объекта.

Тим нашел подходящее место:

— Автомобиль стоит у меня шестым в списке после гепарда, поэтому я должен поставить его на шестой кубик. Подравниваю курсор по центру этого кубика, нажимаю левую кнопку мыши и... автомобиль на экране! Но вот курсор остался прежним.

И для этого удивившего Тима явления у дедушки нашлось объяснение:

— *Художник* думает, вдруг ты захочешь поставить машину еще где-нибудь на картинке.

— Но мне больше автомобилей не надо!

— Тогда нажми правую кнопку мыши или клавишу перехода на клавиатуре. Эти два разные нажатия, как вы помните, работают совершенно одинаково.

Тим продолжил свою работу в *Художнике*, и вскоре все кубики были заполнены. Когда он снова вызвал *Веселые картинки* и запустил свою задачу, радости внука не было предела. Задача получилась красивой и работала, как часы.

Задание 6

Повторите работу Тима по созданию задачи *Догадалка Тима*.

Книга для учителя

ГЛАВА 3. ВОЛШЕБНЫЕ КАРТИНКИ

Материал пособия для детей излагается в следующей последовательности:

1. Место страницы *Картинок* в иерархии страниц *Монитора* и ее состав;
2. Работа с *Волшебными картинками*:
   2.1. Решение задач;
   2.2. Структура файлов *Волшебных картинок* на диске;
   2.3. Редактирование готовых задач;
   2.4. Создание новых задач.
3. Работа с *Веселыми картинками*:
   3.1. Решение задач;
   3.2. Структура файлов *Веселых картинок* на диске;
   3.3. Создание новых задач.

Основная часть работы детей посвящается созданию оригинальных заданий. При этом желательно выбирать для конструирования темы, непосредственно связанные с основным учебным процессом школьников. Дети могут разрабатывать задания по информатике, биологии, физике, математике и другим школьным предметам.

Наиболее интересным представляется последовательная разработка какой-нибудь одной темы школьного учебника.

Дети должны осознавать, что результаты их труда найдут практическое применение на уроках как в своей родной школе, так и в других школах (при дистанционном обучении). Эта мотивация, в дополнение к учебной, творческой, соревновательной и игровой, является важным стимулом деятельности ребенка.

Процесс создания задачи внутри выбранной темы можно разделить на два этапа:

- Поиск и построение принципиальной модели задачи.
- Реализация задачи в программной среде пакета.

Первый этап — наиболее сложный и в методическом плане ответственный. Важно, чтобы ребенок осознал предмет поиска: множество упорядоченных объектов внутри исследуемой темы.

Примеры. Математика, 4-й класс.

1) Обыкновенные дроби, записанные в обозначениях информатики и упорядоченные по убыванию.

3/2, 4/3, 2/2, 3/4, 2/3, 1/2, 1/3, 1/4

2) Обыкновенные дроби, записанные словами и упорядоченные по возрастанию.

двадцать три тысячных	девяносто тысячных
одна двадцать пятая	одна десятая
одна двадцатая	сто сорок три тысячных
восемь сотых	двадцать три сотых

3) Числовые множества. Каждое следующее множество в цепочке является подмножеством предыдущего.

число, положительное число, положительная дробь, целое положительное число, натуральное число, цифра

4) Множество целых чисел, упорядоченное по возрастанию последней цифры в записи.

10, 1231, 22, 3, 2524, 95, 96, 7177, 18, 99

Проанализируем приведенные примеры.

Их можно разделить на три группы.

— Основные понятия. В задания выносятся упорядоченные множества основных теоретических понятий исследуемой темы — пример 3.

— Упражнения. Задания на использование основных понятий темы на практике — примеры 1, 2.

— "Черные ящики". Задания на поиск закономерностей — пример 4.

Формулировка задачи в каждой группе имеет свои особенности.

В группе "Основные понятия" задания основываются на теоретических понятиях темы. В примере 4 оно может быть сформулировано так: "расположите понятия в цепочку в соответствии с классификацией множества чисел".

В группе "Упражнения" задания носят тренировочный характер. В формулировках задач описывается признак упорядочивания элементов множества. Для примера 1: "расположите обыкновенные дроби в порядке убывания их числовых значений".

В группе "Черные ящики" формулировка задания должна содержать фразу типа "найдите правило, по которому построена цепочка объектов".

Этап придумывания задачи наиболее ответственен и требует максимальных усилий руководителя. Он должен определить темы, помочь детям найти внутри них подходящий материал для задач, из множества предложений отобрать наиболее интересные и полезные, подредактировать детские формулировки заданий.

На втором этапе отобранные задачи должны быть реализованы программными средствами пакета. Представляется разумным разделить детей на небольшие группы и каждой из них предложить серию задач для реализации. Удобно, когда каждая группа состоит из трех человек (подгрупп):

- главный конструктор,
- художественный оформитель,
- музыкальный оформитель.

Графические и музыкальные файлы, содержащиеся в пакете, не смогут покрыть все темы и конкретные сюжетные линии создаваемых задач. Поэтому, вероятно, возникнет потребность в новых графических и музыкальных иллюстрациях.

Новые музыкальные фрагменты необходимо создавать, используя редактор *Шарманщик*. При этом дети могут придумывать свою музыку или переписывать ноты в *Шарманщик* из нотных альбомов.

Графические иллюстрации могут создаваться в Художнике или в любом другом графическом редакторе, позволяющем записывать файлы на диск в графических форматах PIC или PCX. Если рисунок создается не в *Художнике*, то "чужеродный" графический редактор необходимо настроить на "роботландский" режим экрана: 16-цветную палитру и разрешение 640 × 350.

Задания. Новые задачи для программ *Волшебные картинки и Веселые картинки.*

Задачи, придуманные детьми, должны объединяться в одну или несколько серий, отправляемых в учебный центр дистанционного обучения в виде отдельных архивов.

Архивная запаковка каждой серии содержит:

— файл-меню заданий (файл с расширением MEN),

— файлы-задачи (файлы с расширением TSK, описанные в меню заданий),

— графические и музыкальные иллюстрации к задачам (файлы с расширениями PIC, PCX и MUS),

— пиктограмма для кнопки *Монитора* (файл с расширением CUT),

— описание задач (текстовый файл с именем TASK.TXT).

Файл с описанием задач имеет следующую структуру:

— Название серии задач,

— Регистрационный номер команды,

— Электронный адрес команды,

— Руководитель команды,

— Краткая характеристика серии,

— Описание каждой задачи.

Описание задач должно строиться по следующей схеме:

— название задачи в меню заданий;

— имя файла, содержащего задачу;

— имя файлов с музыкальными и графическими иллюстрациями;

— автор(ы);

— постановка задачи;

— правило построения цепочки;

— рекомендации по использованию задачи на школьных уроках.

Если дистанционный контроль проводится в конкурсной форме, то зачастую вводятся ограничения на количество работ, отправляемых на конкурс. Это обстоятельство позволяет идеи дистанционного контроля перенести в условия локальных учебных занятий: руководитель группы не ограничивает учащимся количество работ, но затем, проведя внутри группы коллективную "перекрестную" проверку, организует отбор лучших работ для отправления на конкурс в дистанционный учебный центр.

Более того, отойдя от среды дистанционного обучения, такую же педагогическую работу (с сокращенным объемом документации) можно сделать достоянием проектной деятельности школьников как в кружковой работе, так и в урочных формах учебного процесса.

Продолжение следует

План проведения Дня учителя информатики в Московском городском доме учителя

15 апреля, ул. Пушечная, дом 4, строение 2,
станции метро "Лубянка" или "Кузнецкий мост"

Время проведения мероприятия	БОЛЬШОЙ ЗАЛ Секция **"Преподавание информатики в начальной, средней и старшей школе"**	МАЛЫЙ ЗАЛ Секция **"Страницы повышения квалификации"**	ПАРКЕТНЫЙ ЗАЛ Секция **"Мастер-классы"**
10^{00} — 10^{30}	**Открытие.** **Пленарное заседание.** Тема пленарного заседания: **"Место курса информатики в действующем и экспериментальном учебных планах"**.		
10^{30} — 10^{50}	Перерыв		
10^{50} — 12^{10}	**Семинар "Информатика в начальной школе"**. Семинар проводит *Ю.А. Первин*	**Семинар "Олимпиады по информатике. Пути к вершине"**. Семинар проводит *Е.В. Андреева*	**Семинар "Виртуальная школа — всепроникающие информационные технологии"**. Семинар проводит *А.И. Сенокосов*
12^{10} — 12^{30}	Перерыв		
12^{30} — 13^{50}	**Семинар "Базовый курс информатики"**. Семинар проводит *Н.Д. Угринович*	**Семинар "Современные педагогические технологии и частные методики обучения информатике"**. Семинар проводит *И.Н. Фалина*	**Семинар "Роботландский сетевой университет"**. Семинар проводит *В.А. Козлова*
13^{50} — 14^{10}	Перерыв		
14^{10} — 15^{30}	**Семинар "Профильные курсы информатики"**. Семинар проводит *Е.К. Хеннер*	**Семинар "Введение в специальность "учитель информатики"**. Семинар проводит *А.Г. Гейн*	**Семинар "Олимпиады по базовому курсу информатики"**. Семинар проводит *С.В. Русаков*
15^{30} — 15^{50}	Перерыв		
15^{50} — 16^{50}	**Пленарное заседание**. Тема пленарного заседания: **"Обновленное содержание курса информатики, стандарты, итоговая аттестация"**. **Закрытие**	*В течение всего дня в* ***Педагогической гостиной*** *Дома учителя работает музей истории вычислительной техники, а в* ***Голубой гостиной*** *– выставка-продажа учебной литературы.*	
16^{50} — 17^{10}	**Перерыв**		
17^{10} — 18^{00}	**Концерт**		

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

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

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 06.03.2002.
Номер подписан 06.03.2002.
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.