№ 5 (198) февраль 1999

еженедельное
ПРИЛОЖЕНИЕ К ГАЗЕТЕ
«ПЕРВОЕ СЕНТЯБРЯ»

ИНФОРМАТИК

Электронные версии газеты “Первое сентября” и приложений http://www.1september.ru

Интернет для начинающих

А.А. ДУВАНОВ

Продолжение. См. № 36, 40, 44, 46/98; 2/99

1.6. Мы писали, мы писали, наши пальчики устали

(Как правильно писать письма)

Ответы Паука

Вася. Я написал Пауку два совершенно одинаковых письма с одной и той же программой, но получил от него две разные сказки:

```
Дорогой Вася!
Специально для тебя я написал такую сказку.

Однажды
стремительный Змей-Горыныч
захромал вместе с Мудрым Кроликом
ждать вчерашнего дня.
Вот добрались они до места.
Смотрят, а там совсем
дурной Соловей-Разбойник.
— Бедный Йорик! —
так подумали
Змей-Горыныч и Мудрый Кролик.
Они стали лаять
очень сильно. И пропал недруг!
Занялись друзья делом,
а сказке конец.
Привет!!!

 \#/
 =O= Паучок (почтовый робот Роботландии)
 / \
```

```
Дорогой Вася!
Специально для тебя я написал такую сказку.

Однажды
стремительный Змей-Горыныч
пополз вместе с тремя поросятами
пасти козла отпущения.
Вот добрались они до места.
Смотрят, а там совсем
дурной Соловей-Разбойник.
— Ах, как нехорошо! —
так подумали
Змей-Горыныч и три поросенка.
Они стали мычать
очень сильно. И пропал недруг!
Занялись друзья делом,
а сказке конец.
Привет!!!
 \#/
 =°= Паучок (почтовый робот Роботландии)
 / \
```

Вася. Как он их сочиняет?

Папа. У Паука есть сказка-шаблон, в который он вставляет информацию из твоего письма и случайные фразы из своих “домашних” заготовок.

Вася. Забавно!

Папа. Предлагаю поговорить сегодня о принятых правилах оформления электронных писем и проблеме ошибок в тексте письма.

Вася. Это как раз вовремя: я собираюсь наконец сообщить свой электронный адрес моему другу из Перми, чтобы начать с ним электронную переписку.

Папа. Специально записывать в письме свой электронный адрес не надо: твой друг автоматически извлечет эту информацию из полученного сообщения.

Вася. Верно! При этом он может избавиться от ошибок ручного набора.

Автоматическая подпись

Папа. Учитывая твое стремление к автоматизации и рационализации, могу предложить заготовить специальный текст-подпись для автоматического добавления к письмам при отправке их корреспондентам.

Вася. Это мне нравится. Как это сделать?

Папа. Сначала нужно придумать текст подписи.

Вася. Ну, это просто, я буду подписываться “Вася Кук”. Или, пожалуй, так: “С дружеским приветом, Вася Кук”.

Папа. Последний вариант выглядит более дружелюбно. В отличие от первого, “сухого”, он способен дополнительно поднять настроение друзьям — показать твое хорошее к ним отношение. Это особенно важно, если само письмо содержит неприятное для корреспондента сообщение. Почему ты смеешься?

Вася. Я представил себе такое письмо: “Миша, ты дурак!” И автоматическую подпись: “С дружеским приветом, Вася Кук”.

Папа. Ты бы хотел получить такое письмо?

Вася. Ну, я был бы не очень доволен. Скорее всего написал бы в ответ: “А ты — осел, и уши у тебя холодные!”

Папа. Интересная у вас получилась бы переписка и, главное, очень содержательная. Стоило ли для таких сообщений создавать Интернет?

Вася. Да нет, это просто шутка. Таких писем я, конечно, писать не буду.

Папа. А если вдруг получишь подобное письмо, то либо вовсе не отвечай на него, либо напиши очень вежливый ответ: “Дорогой Миша! Ценю твое остроумие и жду более содержательных писем в свой адрес”. Вежливость — это гораздо более мощное оружие, чем грубость.

Вася. Это понятно. А как автоматизировать подпись?

Папа. При работе с программой DMAIL нужно создать отдельный текстовый файл с подписью и поместить его в свой почтовый каталог на диске. У тебя этот каталог называется VASIL.

Вася. Файл с подписью назову PODPIS.TXT. Запишу в нем такой текст:

```
С дружеским приветом,
Вася Кук
```

Папа. Теперь нужно соответствующим образом настроить программу DMAIL на этот файл.

Запусти эту программу.

Вася. Готово.

Папа. Теперь выбери строку **Конфигурация DMAIL** в позиции **Установки** главного меню:

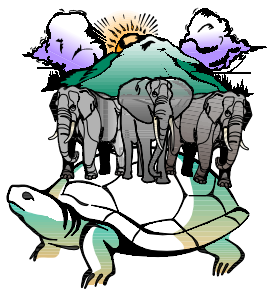
Продолжение на с. 2

НАШИ ДЕТИ БУДУТ ЖИТЬ В XXI ВЕКЕ

12 лекций о том, для чего нужен школьный курс информатики и как его преподавать

Лекция 3

А.Г. КУШНИРЕНКО, Г.В. ЛЕБЕДЕВ



Об основных понятиях, идеях и целях школьного курса информатики “по учебнику” *А.Г. Кушниренко, Г.В. Лебедева, Р.А. Свореня* “Основы информатики и вычислительной техники” (М.: Просвещение, 1990, 1991, 1993, 1996). Дается также ряд практических советов, предлагаются соответствующие методические приемы.

Авторы надеются, что материал окажется полезным для учителей и методистов, использующих указанный учебник, а также для тех, кто желает сравнить разные подходы к преподаванию школьного курса информатики или разработать свой собственный курс.

Продолжение следует

КОНФЕРЕНЦИИ

VIII Открытая российская научно-практическая конференция школьников по физике, математике, информатике, экологии

О работе секции Информатики на конференции, которая состоялась 5—8 января 1999 года в энергофизическом лицее № 1502 при МЭИ.

Интернет для начинающих

Продолжение. Начало на с. 1

Вася. На экране появилось новое окошко с названием **Пользователь: Vasil.**

Папа. Именно в нем, в поле **Файл подписи:**, нужно записать имя файла PODPIS.TXT, а позицию **Добавлять файл подписи** отметить крестиком:

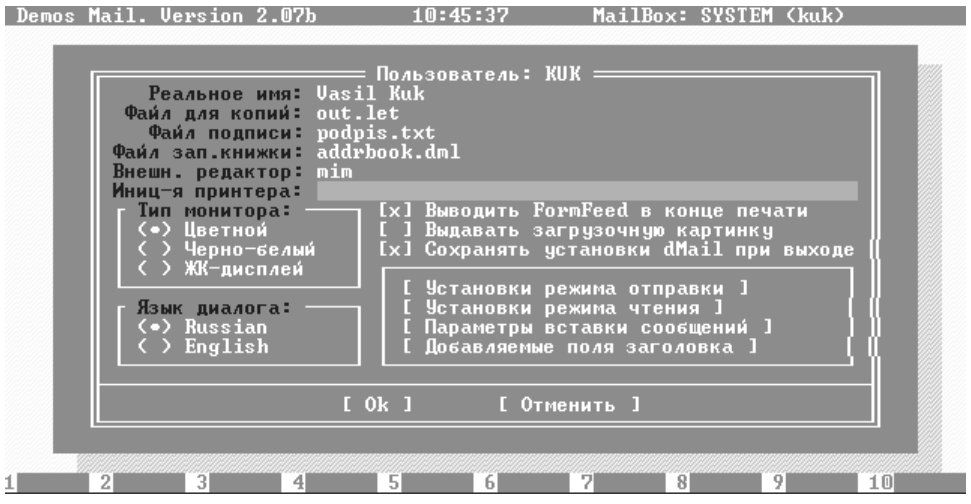
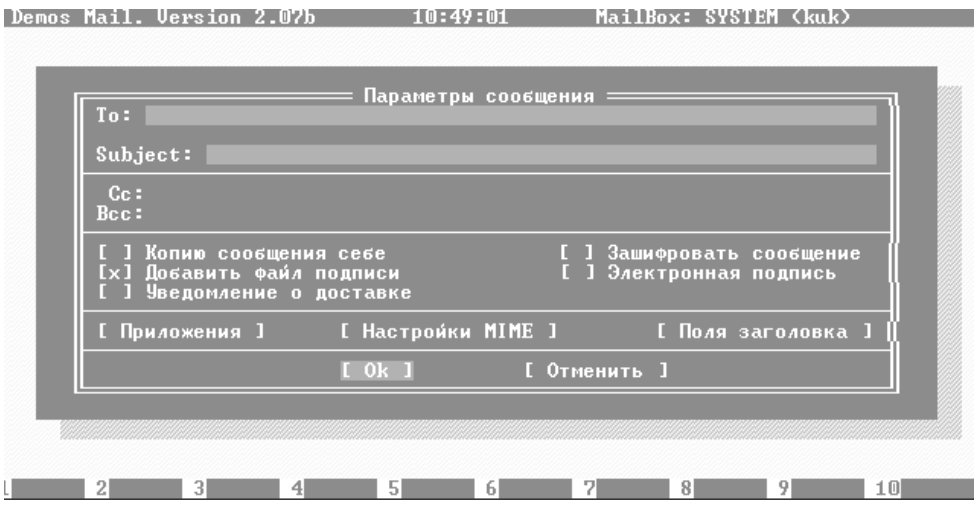


Рис. 1

Папа. Теперь подпись будет добавляться в конец письма автоматически.

Вася. А если в каком-то письме мне нужно подписаться по-другому, то как выключить автомат?

Папа. После настройки программы DMAIL на автоматическую подпись конверт письма будет выглядеть так (позиция **Добавить файл подписи** будет отмечена крестиком):



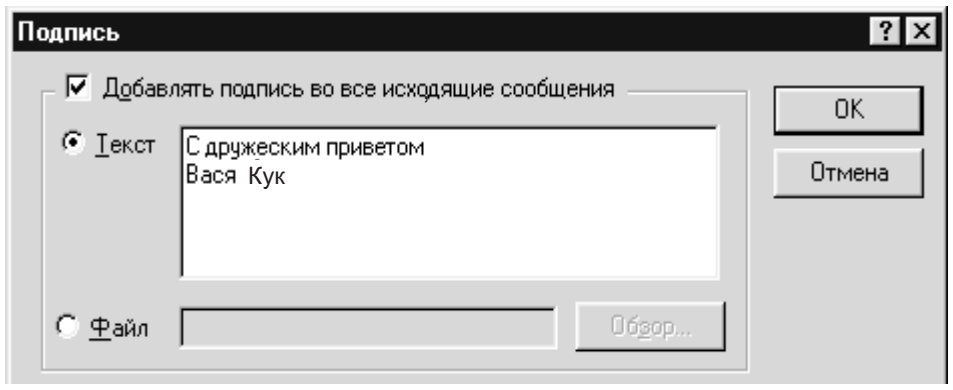
Если автоматическая подпись для данного письма не нужна, то нужно снять соответствующую отметку (крестик).

Вася. Кажется, все понятно. Теперь надо разобраться с почтовой программой Windows.

Папа. Настройка этой программы выполняется похожим образом. Для создания автоматической подписи нужно выбрать строку **Бланк сообщений** в позиции **Сервис** главного меню. Затем в появившемся окне нажать кнопку **Подпись**.

Вася. Появилось окно с названием **Подпись**.

Папа. В этом окне нужно написать текст подписи и поставить отметку в строке **Добавлять подпись во все исходящие сообщения**.



Вася. Здесь это делается быстрее: не нужно самому изготовлять отдельный файл.

Папа. Согласен, многие операции в Windows выполняются более естественным для человека образом, чем в DOS.

Перед тем как ты завяжешь активную переписку с друзьями, хочу обсудить некоторые особенности электронного жанра.

Цитаты

Папа. Первый вопрос — использование цитат.

Вася. О цитировании писем мы уже говорили. Для отделения цитаты от остального текста в начало каждой ее строки вставляется символ “>”. Цитаты напоминают автору, на какое письмо получен ответ.

Папа. Обычно цитируют не все письмо целиком, а только ключевые фразы, которые напрямую связаны с текстом ответа. Например, если бы Миша из Перми прислал такие вопросы:

```
Вася, как зовут твою собаку?
А как зовут твою кошку?
```

То ответ мог бы быть таким:

```
> Вася, как зовут твою собаку?
Миша, мою собаку зовут Мики. Мы с ней большие друзья. :)

> А как зовут твою кошку?
Мою кошку зовут Сима — она большая проказница. :(
```

Вася. Понятно. А что это за странные знаки в конце каждого ответа?

Смайлики

Папа. Это смайлики (от английского слова *smile* — улыбка). Когда человек читает письмо, то он не видит, как его собеседник по переписке улыбается или хмурит брови. В электронном письме нет языка жестов, но есть принятые условные обозначения — смайлики — для выражения эмоций. Их используют очень часто. Посмотри на строку, в которой говорится о собаке. Смайлик “**:)**” обозначает улыбку. Это нетрудно понять, если наклонить голову налево и внимательно посмотреть на этот знак. Сразу видны два глаза и рот до ушей. А вот после слов про кошку поставлен смайлик “**:(**”. Понятно, что Вася Кук не в восторге от проказ хвостатой себялюбицы.

Вот еще некоторые смайлики:

:) или **:-)** — улыбнулся; **:(** или **:-(** — поморщился;

;-) — подмигнул; **:-O** или **8-)** — удивился;

:-| или **:-I** — я совершенно серьезен; **:->** — саркастическая улыбка;

:'-(— я плачу; **:*** — поцелуй.

Вася. Не думал, что обычными символами можно рисовать такие выразительные картинки!

Структура письма

Папа. Что обычно ты пишешь в первой строчке письма?

Вася. Как у Высоцкого: “Во первых строках письма шлю тебе привет!” Ну, обычно я пишу: “Здравствуй, Миша!” или просто “Привет!”.

Папа. Иными словами, письмо начинается с обращения. Для близкого друга вполне подойдет фраза “Здравствуй, Миша!”, или “Дорогой Миша!”, или просто “Привет!”. А как ты начнешь письмо своему директору школы?

Вася. Понятно, я не буду писать “Привет!” и даже “Дорогой Иван Петрович!”.

Папа. Правильно! К взрослому человеку, с которым ты не находишься в родственных отношениях, лучше обратиться так: “Уважаемый Иван Петрович!” или “Глубокоуважаемый Иван Петрович!”.

А как обратиться к человеку, если он не написал в письме свое имя или подписался слишком кратко, например, так: “И.П.Щ.”?

Вася. Напишу: “Дорогой И.П.Щ.!”

Папа. Это некрасиво. С одной стороны, корреспондент виноват сам: не указал в подписи полного имени, но с другой — зачем укорять его, может быть, это вышло случайно?

Вася. Так как же к нему обратиться?

Папа. Если это твой ровесник, то можно так: “Дорогой друг!”

Если это взрослый человек, то лучше так: “Уважаемый корреспондент!”

Конечно, мы обсудили самые общие правила построения обращений. В каждом конкретном случае все решается индивидуально, но так, чтобы в первой же строке не обидеть своего адресата. Особенно неприятно допускать ошибки в имени корреспондента — это может сразу настроить его против тебя. Ошибки в письмах — вещь вообще неприятная, но в обращении они просто недопустимы.

Вася. Я буду тщательно проверять свои письма и особенно обращения.

Папа. Теперь давай поговорим об основной части письма, расположенной между обращением и подписью.

Вася. Думаю, здесь не может быть особых правил: каждый пишет так, как привык.

Папа. Лучше иметь хорошие привычки, чем плохие. Письмо легче понять, если в нем порядок: проставлены нумерованные заголовки и подзаголовки. Письмо тогда будет напоминать книгу с главами, параграфами и пунктами. Самое любопытное в том, что писать такие письма гораздо легче: написал заголовок и сразу задал себе программу — о чем писать в этой небольшой главе.

Вася. Твое правило подходит для деловых писем, а если я сочиняю что-то такое, что не укладывается в главы и параграфы?

Папа. Я соглашусь с тобой, но только наполовину: всегда лучше иметь продуманный план сообщения, чем нажимать клавиши по мере того, что в данный момент пришло в голову. Может получиться не письмо, а манная каша, поверь моему журналистскому опыту!

Продолжение на с. 15

Авторское решение, а более — пространное обсуждение задачи с III Всесоюзной олимпиады школьников по информатике

В.В. ПРОХОРОВ

Вот эта задача (примерно такой бланк был выдан школьникам):

III Всесоюзная олимпиада по информатике

Харьков ***Практический тур*** ***1990 г.***

Задача "СУММА"

Обозначим $S(N, a, b) = \sum_{i=1}^{N} \frac{i}{a^{[\log_b i]}}$, где $\log_b i$ обозначает такое x, что $b^x = i$, $[x]$ означает целую часть x, а $\sum_{i=1}^{N} x_i = x_1 + x_2 + ... + x_n$.

Требуется напечатать с как можно большим числом верных десятичных знаков:

а) $S(100, 3, 2)$;
б) $S(100, 3, 1.9)$;
в) $S(100, 3.1, 1.9)$.

Автор данной задачи может предложить несколько подходов к ее решению, каждый из которых имеет свою мораль (во всем сущем есть своя мораль!). Ради извлечения этих моралей задача автором и конструировалась. Но главная мораль, стоящая над более частными, — что

не правы те, кто полагает, что приближенность решения вычислительных задач — свойство, органически присущее компьютерной обработке информации.

Наоборот, для "почти любой" вычислительной задачи, как правило, возможно получить абсолютно точный ответ, настолько точный, насколько это позволяют сделать исходные данные, которые мы введем в ЭВМ. Не исключение и наша задача: и в ней решение можно найти не с какой-либо точностью, заботясь лишь о том, чтобы погрешность была "не слишком большой", а

в данной задаче ответ можно найти без какой бы то ни было погрешности вообще, точно!

Но еще есть и "самая-самая главная мораль":

прежде чем делать какую-либо работу, полезно сначала подумать.

Нам кажется, это очень хорошая мораль (ее можно извлечь из всего, что далее будет высказано). Хотя, к сожалению, у очень многих людей, оказавшихся в непосредственной близости от компьютера, главное — поскорее выдать первый результат ("думать будем потом"), нажимать кнопки клавиатуры максимально быстро.

Перейдем непосредственно к нашей задаче. Для начала будем ограничиваться лишь рассмотрением первого пункта задачи — вычислением $S(100, 3, 2)$.

Начальные факты и мораль, которую из них можно извлечь

Начнем со стоящей в показателе знаменателя функции, которую нам вроде бы в любом случае придется как-то вычислять, — с $U(b,i) = [\log_b i]$.

Если поступать по принципу "думать будем потом", то скорее всего придется вспомнить какие-то свойства функции $\log_a x$: ведь в языках программирования функции вычисления логарифма с произвольным основанием обычно нет... И вот здесь может проявиться бытующее убеждение, что при всех обстоятельствах выигрывает тот, кто знает много фактов, а не тот, кто предпочитает (и умеет) размышлять. Тот же, кто знает много фактов о разных элементарных функциях и для кого "log" не просто "log", а обозначение логарифма, вспомнит, что справедливо тождество $\log_y x = \ln x / \ln y$.

Примечание: лучше бы он не помнил, что "log" — это логарифм, и эту формулу тоже. Но что поделаешь: хорошая память...

Таким образом, вычисление $U(i)$ свелось к манипулированию стандартными функциями массовых языков программирования: вычислению двух натуральных логарифмов, делению полученных значений и вычислению целой части от частного. Поскольку оба логарифма будут вычислены приближенно и деление вещественных чисел тоже будет выполняться приближенно, то и от результата вычисления $U(x)$ можно ожидать всякого.

Кто-то для повышения точности вычисления U займется проблемами вычисления $\log x$ с максимально высокой точностью. Но вот беда: при более или менее прямолинейных подходах из-за наличия операции вычисления целой части будет иметь место неприятное обстоятельство: может оказаться, что при все большем и большем повышении точности вычисления $\log x$ точность определения $[\log x]$ перестанет повышаться... Такие решения задач математики называют некорректными.

Но что если все-таки попробовать иначе — сначала подумать? И поменьше уповать на память? Тогда можно додуматься до того, что не может быть, чтобы при целых b и i целое значение функции $U(b,i)$ нельзя было определить точно! Напомним, что в условии подсказано: через "log" обозначен такой объект, что если $L = \log_b i$, то

$$b^L = i$$

(так что решающему эту задачу знать, что такое "log", вовсе ни к чему; мы даже могли использовать в обозначении этой функции вместо "l", "o", "g" какие-нибудь другие буквы). Разумеется, из этого уравнения, даже если b и i целые, то есть заданы точно, найти число L точно (оно ведь, "за редким исключением", будет трансцендентным), как правило, будет нельзя. Увы...

Да, но задумаемся: а при чем здесь L? Ведь нас интересует не L, а $U = [L]$, а это далеко не одно и то же. Быть может, его-то все-таки можно найти точно?

Действительно, по определению целой части числа $[L]$ — наибольшее целое число, для которого выполнено неравенство $[L] \le L$. То есть для $[L]+1$ оно уже не выполнено — значит, $[L]+1 > L$. Если записать все это вместе, то $[L]$ — такое целое число, для которого выполнено: $L - 1 < [L] \le L$; эта запись, заметим, всегда определяет некоторое число, притом единственное. Можно и так: $L < [L]+1 \le L+1$. И в нашем конкретном случае тогда можно записать $b^L < b^{U+1} \le b^{L+1}$.

Стало быть, U — такое целое число, для которого выполнено условие

$$i < b^{U+1} \le b \cdot i \qquad (*)$$

Так ведь из последней записи $U(b,i)$ можно найти точно — там нет ни одной "приближенной" операции! (Остается повториться, что эта запись всегда определяет некоторое единственное число.) Как это сделать конкретно? Простейший ("тупой") вариант — перебирать все натуральные числа, начиная с нуля, для каждого проверяя выполнение условия (*). Можно предложить и массу других, более эффективных способов вычисления $U(b,i)$, однако принципиальный вопрос мы решили:

$U(b,i)$ можно вычислить без погрешностей!

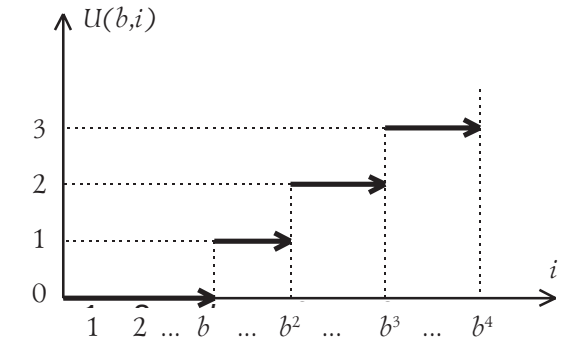
Неплохое начало, подтверждающее две вышеупомянутые главные морали. Это вдохновляет!

Теперь пойдем дальше по пути вычисления значения интересующей нас суммы, рассматривая по очереди разные подходы с разными моралями.

Подход 1-й и его мораль ("удачная расстановка скобок")

Запись формулы в условии задачи (как суммы дробей) провоцирует нас на выполнение операций в такой последовательности: сначала — вычисление дроби (вычислять ее знаменатель мы уже научились), потом — суммирование. Но если реализовать этот подход (подход 0) "в лоб", то деление даст нам приближенный результат — следовательно, и значение всей формулы будет определено приближенно (да еще с не вполне известной точностью; не каждый скажет, какие знаки числа-результата верные, а какие — нет). Так что этот подход мы отдельно не выделяем и особой морали в нем не усматриваем. Хотя, конечно, люди, умудренные практической работой на компьютере, найдут, где приложить свои усилия, и будут добиваться "сколь угодно высокой точности" реализацией "арифметики высокой точности". Правда, для достижения "сколь угодно высокой точности" потребуется "сколь угодно много" времени работы компьютера и, кроме того, автор такой программы, если он хорошо понимает, о чем речь, затруднится сказать, с какого же знака ответа начинаются неверные... Это, видимо, не тот путь.

А нельзя ли нам как-то перегруппировать вычисления, чтобы уменьшить количество "неприятных" "приближенных" операций, осуществлять большей частью точные? Если подумать, так можно. Ну, например, следующим образом. Нетрудно заметить, что функция $U(b,i)$ (b зафиксировано) при изменении i меняется "скачками", меняя значения лишь при i из ряда b, b^2, b^3, b^4 и т.д.:



Стало быть, в нашей сумме некоторые дроби имеют общие знаменатели и их можно не делить поврозь с последующим сложением, а сначала сложить их числители и только потом поделить. Таким образом, мы существенно уменьшим количество "нехороших" делений, вносящих погрешности в результат. И найдем значение суммы с лучшей точностью.

Напрашивается мораль:

сгруппировав вычисления неким подходящим образом, мы можем помочь более успешному решению стоящей перед нами задачи.

Неплохо. Но нельзя ли из этой морали извлечь что-либо большее, позволяющее найти результат еще точнее? Пойти дальше? Попробуем. Посмотрим еще раз на нашу формулу. И если посмотрим внимательно, то увидим, что у нас записана сумма дробей с целыми числителями и целыми знаменателями. Так, может (желая отодвинуть как можно дальше момент применения "нехороших" операций приближенного деления), нам стоит избежать операции деления числитель на знаменатель в этих дробях вообще? Хоть каждой в отдельности, хоть для групп с одинаковыми знаменателями после сложения их числителей (как мы делали, когда извлекли нашу первую мораль). Ведь мы можем, скажем, привести все дроби-слагаемые к общему знаменателю и сложить их числители. Тогда после этой процедуры мы сведем нашу сумму к одной дроби, и нам для получения результата потребуется выполнить лишь одно "нехорошее" (приближенное) деление.

Превосходно! Мы свели процедуру определения требуемого значения $S(100, 3, 2)$ к такой, где на пути к ответу встречается лишь одна дающая погрешность приближенная операция — деление, да и та в самом конце (ответ и есть результат от этого деления).

Так ведь мы получили даже большее, чем хотелось! Мы нашли решение, дающее ответ ВООБЩЕ БЕЗ ВЫЧИСЛИТЕЛЬНОЙ ПОГРЕШНОСТИ! Непонятно? Мы довели вычисления без погрешностей до фазы, где остается поделить

Продолжение на с. 4

Авторское решение, а более — пространное обсуждение задачи с III Всесоюзной олимпиады школьников по информатике

Продолжение. Начало на с. 3

два натуральных числа (результат от деления и будет ответом), но ведь частное от деления двух натуральных чисел можно получить с "неограниченной точностью"! Например, "запустив" алгоритм деления столбиком. И компьютер в процессе работы этого алгоритма будет нам (весьма быстро) печатать один за другим ВЕРНЫЕ знаки ответа. И покуда у нас будет хватать терпения и времени, мы будем иметь все новые и новые верные знаки. Ну а совсем умные вспомнят (или заметят, или догадаются), что результат от деления двух натуральных чисел (в десятичной системе счисления) — бесконечная ПЕРИОДИЧЕСКАЯ десятичная дробь. Тогда мы сможем где-то запомнить цифры периода и просто их печатать в бесконечном цикле (или столько раз, сколько захочется).

Может ли кто-то обоснованно сказать, что мы нашли ответ все же с погрешностью? Строго говоря — нет. Ведь (написав тривиальную процедуру) мы сможем давать точные ответы на вопросы типа: "Каков 1234567890-й (или любой другой) знак числа-результата?" Притом через не зависящее от номера знака время (весьма малое). Можно ли желать чего-либо еще? Что этот скептик может еще понимать под словами "точное решение"? Пусть скажет! Кстати, в том, что мы только что сказали, тоже есть своя (большая!) мораль:

прежде чем о чем-либо спорить до хрипоты, иногда полезно договориться о терминологии (узнать, кто что понимает под конкретными словами) и, быть может, дать новые определения.

Иногда новое определение значит не меньше, чем решение какого-либо класса задач!

Итак, подытожим сделанное. Мы выявили основанный на морали "хорошо расставь скобки в формуле — и будет хорошо" (в ее широком понимании) алгоритм определения точного (не имеющего никакой погрешности) значения результата. От нас требовалось найти значение ответа с минимальной погрешностью, а мы нашли его вообще без погрешности, так что получено решение, лучше которого вообще быть не может ("решение" — в смысле "ответ").

(По ходу дела мы столкнулись с желательностью решения одной задачи, здесь — вспомогательной, но представляющей и самостоятельный интерес, — с задачей определения по числителю и знаменателю рациональной дроби записи представления этого числа в виде в общем случае бесконечной периодической десятичной дроби. Надеемся, читатель справится с этой задачей самостоятельно.)

Но, несмотря на такое достижение, посмотрим, какие еще могут быть варианты решения задачи. Зачем? Во-первых, чтобы ни одному подходу не было обидно. Ведь то, что "расстановка скобок" первой пришла нам в голову, — чистая случайность. (Хотя после того, как мы в этом первом подходе дошли до решения с идеальной точностью, мы уже будем привередливы и среди всевозможных вариантов решения будем обращать внимание лишь на те, которые тоже дают абсолютно точный ответ.) А во-вторых, чтобы из разнообразных подходов к решению извлечь побольше различных моралей. А хорошая мораль — штука очень полезная: не одну ведь эту задачу придется в жизни решать. Глядишь, какая-нибудь из наших "впрок заготовленных" моралей и сослужит когда-нибудь добрую службу!

Подход 2-й и новая мораль ("удачный выбор типа данных")

Итак, еще раз посмотрим: что же нам требуется посчитать? Сумму (дробей). А как обычно вычисляют суммы типа S ($i=1$, N, $x[i]$)? Примерно так:

```
S:=0; for i from 1 to N do S:=S+x[i]
```

Но если нам никак не хочется выполнять операцию деления суммируемых дробей до самого последнего момента, то что же такое $x[i]$? Рациональные дроби. А что если нам и считать, что суммируемые значения в нашей программе имеют тип данных "рациональная дробь"? Вполне возможно. Хотя в наиболее массовых языках такого типа данных нет. Ну и что?

Это нас не должно смущать. Если нам показалось целесообразным использовать данные типа "рациональная дробь" — будем этот тип для краткости обозначать "РД", — так будем же их использовать! Даже если в языках, попадающих в наш кругозор, не запланирован такой тип, его всегда можно организовать "вручную" (например, хранить значение рациональной дроби в виде двухэлементного массива — числитель и знаменатель).

Итак, пусть каждое слагаемое имеет тип РД. Тогда при вычислении суммы есть резон считать сумму тоже имеющей тип РД и использовать (ввести) на данных типа РД операцию РД-сложения. Если в используемом языке нет типа РД, то нетрудно ввести "вручную" как этот тип, так и операцию сложения на нем (с РД-слагаемыми и РД-результатом), ведь это всего-навсего сложение двух дробей. Тогда нам останется для соответствующих $x[i]$ выполнить алгоритм

```
S:= Θ; for i from 1 to N do S:=S⊕x[i],
```

где Θ — "нулевая рациональная дробь", например, 0/1, а $\oplus$ — операция суммирования рациональных дробей.

После выполнения этого алгоритма мы будем иметь значение искомого ответа, но в виде рациональной дроби (все операции по вычислению этой дроби не сопровождались никакими погрешностями). Так что остается реализовать единственную операцию деления, то есть преобразования значения типа "рациональная дробь" в требуемую намеком в условии форму десятичной дроби. А это мы уже рассматривали в подходе 1.

Какая же здесь мораль? Такая:

выбрав типы данных в программе подходящим образом, мы можем помочь более успешному решению стоящей перед нами задачи.

Не только к этой морали, но и к другим тесно примыкает такая:

выбрав язык программирования и средства в нем подходящим образом, мы можем помочь более успешному решению стоящей перед нами задачи.

Ведь некоторые языки имеют как тип данных "рациональная дробь", так и операцию "сложение рациональных дробей". И даже операцию $\oplus$. Так что большая часть задачи в каком-либо из таких языков, в сущности, и программирования-то не требует.

Подход 3-й и та же мораль ("не боимся хитрых типов данных")

В прошлом подходе нежелание делать операцию деления привело нас к введению нового типа данных — рациональных дробей — и операции сложения на них. А почему нам не хотелось делить? Потому что банальное деление (в сущности, преобразование данных из типа "рациональная дробь" в тип "вещественный", последний — в смысле языков Бейсик, Паскаль, Си и т.п., что, к слову, не соответствует пониманию этого термина в математике) сопровождается вычислительной погрешностью. Да, но ведь результат деления двух натуральных чисел — бесконечная ПЕРИОДИЧЕСКАЯ десятичная дробь, и для запоминания такой дроби достаточно конечного количества битов памяти. Давайте введем тип данных "бесконечная периодическая десятичная дробь" (будем обозначать "БПДД") и операцию сложения на данных такого типа! Автор этого не проделывал, но принципиальных затруднений здесь нет. И что важно — преобразование из данных типа РД в данные типа БПДД уже осуществляется без погрешности! Без погрешности возможно и сложение на компьютере данных типа БПДД (впрочем, не обязательно БПДД, можно и БПДвД — "бесконечных периодических двоичных дробей"; хотя о выборе системы счисления — чуть позже).

Так что если реализовать алгоритм

```
S:= Θ; for i from 1 to N do S:=S⊕x[i],
```

где Θ, в сущности, обычный ноль ("ноль целых и ноль десятых в периоде"), а $\oplus$ — операция сложения бесконечных десятичных периодических дробей с результатом того же типа, то полученное S будет почти ответом (останется лишь печатать...). И никаких погрешностей! Заметим только, что, как и в случае рациональных дробей, где 81/90 и 9/10 — одно и то же, здесь также имеет место неоднозначность представления чисел, например, 23.15(9) и 23.16(0) — одно и то же число! (Конечно, и тут не без своей морали.)

Подход 4-й и та же мораль в новом свете ("удачный выбор системы счисления")

Только что мы попробовали все же осуществлять операции по вычислению значения формулы в том порядке, как это следует из самой формулы. В том числе и выполняя деление, когда предписано. А работа с "хитрым" типом данных БПДД позволяет получить ответ точный, без погрешностей. Но мы для представления данных выбрали десятичную систему счисления, а в ней результаты промежуточных делений для нашей формулы являются бесконечными, хотя и периодическими, дробями. Почему мы выбрали десятичную систему счисления? Потому, что очень подходит к нашей задаче? Да нет, просто из-за распространенности десятичной системы у людей, да и результат нам надо выдать в десятичной системе. А что если поработать, если уж выбирать такой подход, в более подходящей системе счисления?

Поскольку мы сейчас рассматриваем случай, когда в формуле в условии задачи принято $a=3$, то имеет прямой смысл избрать для работы троичную систему счисления. Ведь при этом каждый результат деления в каждом из слагаемых будет не бесконечной, как в десятичном представлении, а КОНЕЧНОЙ троичной дробью (причем элементарно определяемой, даже и вычисления-то, в сущности, никакие не нужны)! Да, есть такой факт: для любого числа, выражающегося бесконечной периодической дробью в какой-то системе счисления, можно подобрать такую систему счисления, в которой это же число будет выражаться уже КОНЕЧНОЙ дробью! Алгоритм сложения конечных дробей (в том числе троичных) крайне прост, и его элементарно реализовать. А в заключение, когда будет получено значение S как троичной дроби, останется перевести его в десятичное представление (увы, десятичная дробь будет уже бесконечной...), что весьма несложно.

Подход 5-й и большая мораль ("подумать, пытаясь найти аналитическое решение")

Когда с помощью ЭВМ решается любая задача, то всегда можно выделить некую подготовительную часть работы, где с применением математических методов проводится преобразование, упрощение задачи (в том числе и с учетом того, что принятые в математике традиции связаны с точными преобразованиями), и (как считается, более приземленную) часть, непосредственно примыкающую к использованию ЭВМ. Баланс в использовании этих двух частей бывает весь-

Окончание на с. 13

VIII ОТКРЫТАЯ РОССИЙСКАЯ НАУЧНО-ПРАКТИЧЕСКАЯ КОНФЕРЕНЦИЯ ШКОЛЬНИКОВ ПО ФИЗИКЕ, МАТЕМАТИКЕ, ИНФОРМАТИКЕ, ЭКОЛОГИИ

состоялась 5–8 января 1999 года в энергофизическом лицее № 1502 при МЭИ

Секция информатики была представлена в этом году 25 докладами. Авторы работ — это учащиеся 8–11-х классов. Восьмой раз собрались вместе талантливые, заинтересованные и увлеченные. Они представляли свои работы, и было ясно, что там, где они учатся, идет большая, очень трудная и кропотливая работа учащихся и педагогов. А цель этой работы — сделать творчество и поиск необходимыми элементами процесса обучения.

Традиционно сильные работы представили учащиеся лицеев и гимназий (85% всех призовых мест). Жюри отметило возросший общий уровень выступлений всех участников. Это результат большой подготовительной работы, проводимой как организаторами конференции, так и теми, кто готовил команды к выступлению.

Лучшими на конференции стали доклады ученика лицея № 1502 Евдокимова Кирилла и ученика УВК № 1830 Мусатова Дмитрия (оба — представители московских команд). И если первый был посвящен традиционно любимой теме, связанной с построением трехмерных изображений, то тема второго доклада была новой для данной конференции — речевое воспроизведение текстовых файлов. Несмотря на различие тематики, общим для этих работ было следующее: оригинальность используемых алгоритмов (как хотелось бы, чтобы именно таких работ было больше), творческий подход при реализации поставленной задачи и хорошая подготовленность выступления.

Жюри отмечает высокий уровень работ, занявших вторые и третьи места, особенно хочется выделить доклады Щедрина Валерия (УФМЛ, г. Киев) — “Менеджер памяти”, Агапова Виталия (гимназия “Эврика”, г. Новгород Великий) — “Тестирование квалификации начинающего программиста” и Дячука Дмитрия (лицей № 1, г. Черновцы) — “Распознавание речевых образов при управлении роботами”. Эти работы отличались новизной, грамотным и техничным исполнением. Хорошую технику владения средствами визуального программирования показали Савич Павел (лицей № 1502, Москва), Шишков Денис (лицей № 1, г. Барановичи), Смирнов Олег (городской лицей, г. Мариуполь). Жюри отмечает тенденцию к уменьшению количества представленных обучающих программ, хотя еще год назад казалось, что необходимо будет создавать специальную подсекцию для работ этого класса. Очевидно, это связано с расширением рынка готовых качественных обучающих программ, с одной стороны, и определенным выравниванием возможностей в использовании технических и программных средств — с другой. И, наверное, это можно оценивать как положительную тенденцию, поскольку теперь основным направлением творчества наших талантливых молодых коллег будут поиски оригинальных и нетрадиционных задач, новых алгоритмических находок. Спасибо всем, кто выступал и кто помогал ребятам.

Список команд-участниц

1. Новгород Великий, гимн. “Эврика”
2. Нижний Тагил, политехническая гимн.
3. Санкт-Петербург, Аничков лицей
4. Санкт-Петербург, ФТШ им. Иоффе
5. Санкт-Петербург, сборная (шк. № 242)
6. Саров Нижегородской обл.
7. Ставрополь
8. Тверь
9. Раменское Московской обл.
10. Удельная Московской обл.
11. Барановичи, Беларусь, лицей № 1
12. Киев, Украина, шк. № 38
13. Киев, Украина, Укр. ФМЛ
14. Донецк, Украина, лицей при ДонГУ
15. Запорожье, Украина, лицей № 105
16. Запорожье, Украина, лицей “Выбор”
17. Мариуполь, Украина, городской лицей
18. Черновцы, Украина, лицей № 1
19. Москва, эксп. шк. № 91
20. Москва, шк. № 422
21. Москва, шк. № 444
22. Москва, шк. № 548
23. Зеленоград, шк. № 616
24. Москва, шк. № 1199 “Лига”
25. Москва, лицей № 1303
26. Москва, лицей № 1501
27. Москва, лицей № 1511
28. Москва, лицей № 1550
29. Москва, шк. № 1830
30. Москва, лицей № 1502

ИТОГО:
Москва — 12 команд — 59 чел.
Россия — 10 команд — 43 чел.
СНГ — 8 команд — 33 чел.
ВСЕГО:
30 команд — 135 чел.

ВНИМАНИЕ!

Следующая конференция будет проведена с 3 по 10 января 2000 г.

Чтобы получить приглашение на конференцию, нужно обратиться в Энерго-физический лицей № 1502 при МЭИ по адресу:

111578, Москва, ул. Молостовых, д. 10а, Лаврушко О.В.

Тел./факс (095) 300-15-31.

Со всеми вопросами размещения и проживания можно обращаться по тел. (095) 918-95-95 к Ольге Владимировне Лаврушко.

№ 5 (198) февраль 1999

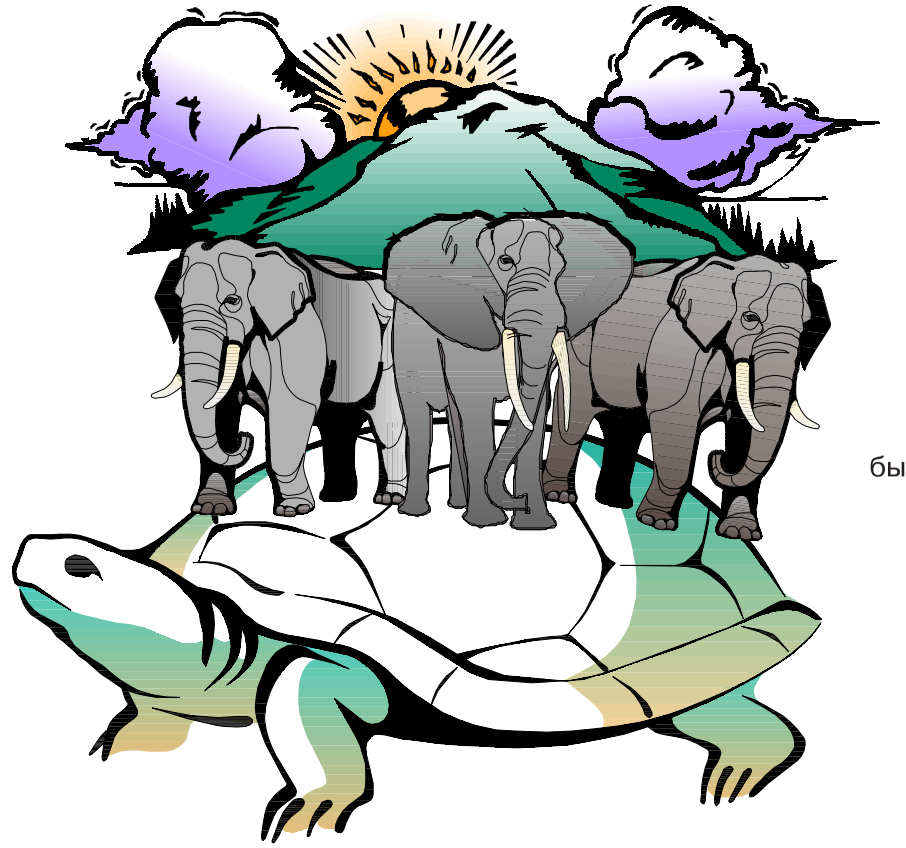
еженедельное
ПРИЛОЖЕНИЕ К ГАЗЕТЕ
«ПЕРВОЕ СЕНТЯБРЯ»

ИНФОРМАТИК

12 лекций о том, для чего нужен школьный курс информатики и как его преподавать

Лекция 3

А.Г. КУШНИРЕНКО,
Г.В. ЛЕБЕДЕВ



Введение, лекции 1, 2 были опубликованы в № 1, 3/99

Книга готовится к изданию в издательстве “Дрофа”.
Выражаем признательность издательству “Дрофа” за содействие в подготовке публикации.

СОДЕРЖАНИЕ

нет, и школьники начинают с энтузиазмом придумывать самых разнообразных исполнителей. Среди таких придуманных исполнителей, запомнившихся мне, был исполнитель Бог, с командами “создать твердь”, “создать хлябь”, “отделить твердь от хляби” и т.д. Все это носит характер художественного творчества. Тем не менее на этом художественном фоне и в такой художественной обстановке можно и нужно обсудить ряд полезных формальных понятий, из которых основное — система команд исполнителя, т.е. набор команд, которые исполнитель умеет выполнять. Прежде всего необходимо, чтобы это был четкий набор, в котором должно быть понятно, как выполнение команды начнется, как команда будет выполняться, что получится в результате и, наконец, когда выполнение команды закончится.

Наконец, даже в четкой системе команд у школьников часто встречается такая ошибка: они придумывают исполнителя, командами которого можно воспользоваться только в одной последовательности, т.е. исполнителя, который фактически умеет делать только одно дело. Другими словами, школьники разбивают это одно дело, скажем, на 5—7 этапов и пишут систему команд типа “выполни этап 1”, “выполни этап 2”, ..., “выполни этап 7”. Для такого исполнителя есть только одна разумная программа, и он совершенно не интересен. Его с успехом можно заменить исполнителем с одной-единственной командой, который по этой команде выполняет все 7 этапов сразу. Именно поэтому нужно требовать от школьников написать *два* существенно разных алгоритма, а не один. Я обычно начинал обсуждение этого упражнения в классе, потом задавал на дом и зачитывал лучшие работы на уроке. Получается довольно интересно и доставляет большое удовольствие и ученикам, и учителю.

В целом же § 5 мало чем отличается от § 4, просто вводится другой исполнитель. Основная разница — команды с аргументами. Это введено в качестве пропедевтики и из некоторых педагогических соображений.

От редакции. Приводим некоторые упражнения, упомянутые в лекции 3

11. (§ 1). В приведенных ниже последовательностях каждый следующий элемент получен по некоторому строгому правилу. Угадайте это правило:
а) а, б, в, г, д, е, ... ;
б) 1, 2, 3, 4, 5, 6, 7, ... ;
в) 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, ... ;
г) 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ... ;
д) 1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 0, 1, 1, 1, 2, 1, ... ;
е) победа, обеда, беда, еда, ... ;
ж) о, д, т, ч, п, ш, с, в, д, д, ... ;
з*) 1, 11, 21, 1211, 111221, 312211, 13112221, ...

12. (§ 1). Петя и Коля играют в следующую игру: Петя задумывает правило преобразования текстовой информации. Коля может задавать Пете любые тексты и узнавать результаты преобразования. Задача Коли — отгадать это правило. Ниже приведены вопросы Коли и ответы Пети в нескольких таких играх. Попробуйте отгадать, какое правило задумал Петя в каждой игре:
а) А → Б; мама → нбнб;
ЭВМ → ЮГН;
язык → аиьл;
б) А → 1;
мама → 4;
ЭВМ → 3;
язык → 4;
в) А → 0+1;
мама → 2+2;
ЭВМ → 2+1;
язык → 2+2;
г) А → А;
мама → амам;
ЭВМ → ВМЭ;
язык → зыкя.

13. (§ 1). В условиях задачи 12 Петя задумывает правило преобразования целых чисел. Ниже приведены вопросы Коли и ответы Пети в нескольких таких играх. Попробуйте отгадать, какое правило задумал Петя в каждой игре:
а) 1 → 0;
5 → 4;
0 → −1;
1990 → 1989;
б) 1 → 0;
2 → 0;
10 → 9;
3 → 3;
20 → 18;
1990 → 1989;
в) 1 → 1;
7 → 1;
10 → 2;
187 → 3;
1990 → 4;
г*) 1 → 0;
8 → 2;
16 → 1;
1990 → 3;
1989 → 4;
100 → 2;
108 → 3;
6 → 1;
7 → 0;
23 → 0;
50 → 1.

12*. (§ 3).** Среди $2n + 1$ различных по массе монет нужно найти среднюю (т.е. такую, которая тяжелее n монет и легче других n монет). Придумайте, как это сделать не более чем за $100n$ взвешиваний на чашечных весах без гирь.

Упражнения

Поскольку ничего нового, кроме самого Чертежника, в § 5 не появилось, упражнения оказываются похожими на те, что были для Робота в § 4. Часть — на анализ алгоритмов, часть — на составление алгоритмов. Правда, при составлении алгоритмов можно на экране получить какие-то картинки, что обычно нравится.

Недостаток сложности алгоритмов (как и в предыдущем параграфе) компенсируется дополнительными условиями из других областей. Например, в упражнении 9 предлагается рисовать картинки, не отрывая пера от бумаги и не проводя линий дважды, т.е. сначала решить топологическую задачу, а потом составить алгоритм ее решения.

Из конкретных упражнений я прежде всего обращаю ваше внимание на начинающиеся со слов "нарисовать какой-нибудь". Например: "Нарисовать какой-нибудь квадрат со сторонами, не параллельными осям координат". Необходимость в них вызывается тем, что при решении большинства задач в информатике правильных ответов бывает очень много. А школьники, да и учителя привыкли к тому, что правильный ответ обычно только один. Чтобы от этого отучить, мы дали несколько задач, в которых правильных ответов заведомо много. Сначала это оказывается трудным — идея, что в задаче может быть много разных правильных решений и нужно придумать только одно из них, для школьника достаточно нова. Но как только эта идея понята на одном примере, все остальные подобные задачи уже не вызывают затруднений.

Кстати, "квадрат со сторонами, не параллельными осям координат", как оказалось, не самая простая задача. И вы можете первоначальную растерянность и последующее "облегчение" учеников превратить в шутку, разрядить атмосферу в классе.

Упражнение 7 — рекламное. Дело в том, что построить графики приведенных в ней функций не так-то просто. А не зная графика, мы пока ничего нарисовать не можем. Продемонстрируйте сложность этой задачи ученикам и порадуйте их тем, что в § 12 они познакомятся с алгоритмом рисования произвольной функции. Так что эта задача просто реклама алгоритма, который в курсе появится позже.

Упражнения параграфа (да и вообще главы 1) можно разбить на несколько типов:

— задачи на выполнение готовых алгоритмов: написан готовый алгоритм, нужно его выполнить, описать результат его выполнения (таковы упр. 2, 11б);
— задачи на понимание алгоритмов: дан алгоритм, требуется понять, что он делает, написать "дано", "надо" и т.п;
— задачи на модификацию алгоритмов, например, упр. 3, 10, 11в, 11г, когда готовый алгоритм надо переделать так, чтобы он делал что-то новое (например, чтобы буквы стали в два раза больше);
— и лишь после этого по сложности идут задачи на составление алгоритмов, когда ничего нет и надо составить алгоритм (упр. 4, 5, 6, 8).

Подбирая упражнения к уроку, желательно использовать задачи из разных типов. Скажем, "выполнить такой-то алгоритм", "изменить такой-то алгоритм", "составить такой-то" и т.д., с тем чтобы различные типы задач были охвачены.

Кроме того, я обращаю ваше внимание на задачи типа упр. 11, которые я бы назвал "на анализ алгоритмов". В них надо что-то сказать про алгоритм, не выполняя его, а глядя на текст. Как это сделать? Например, в упр. 11 последняя строчка "поднять перо" значит: перо будет поднято после выполнения алгоритма, независимо от всего предыдущего. Предпоследняя строчка "сместиться в точку (2,1)" — следовательно, перо окажется в точке (2,1) независимо от всего предыдущего. Такие навыки анализа алгоритмов крайне нужны для их понимания, отладки, поиска ошибок и т.д. Этот класс задач, когда, глядя на текст, надо что-то сказать про алгоритм, очень полезен.

Анализ может быть и более сложным. Например, в упражнении 12, чтобы определить x-координату конечного положения пера после выполнения алгоритма, надо просуммировать первые аргументы всех команд алгоритма.

Наконец, задачи 13, 14 (вы сами можете составить аналогичные задачи) посвящены придумыванию новых исполнителей. Эти задачи должны снять ореол исключительности с Робота и Чертежника и показать, что исполнителей можно придумывать самим. Если вы дали школьникам такую задачу на дом, то соберите решения в письменном виде заранее и запланируйте как минимум пол-урока на обсуждение. Либо, если считаете нужным, сами можете рассказать про других исполнителей, например, про Вездеход, Двуног, Строитель, ГраТекс из комплекта КуМир.

Лучше всего идет Двуног. У него есть две ноги, длинная-длинная шея и голова без туловища. Двуног может двигать ногами и наклонять голову. Если Двуног оторвет от земли одну ногу и наклонит голову, то он начнет медленно падать, пока не упрется в землю ногой. Задача состоит в том, чтобы научить его ходить: сначала по ровному месту, потом по лесенкам, потом в пещере, где нельзя выпрямиться, и т.д.

Можно заниматься всем этим, разнообразить уроки, но если остаться в рамках учебника, то задачи 13 и 14 или обсуждение каких-то других исполнителей становятся необходимыми.

Я также рекомендую дополнить набор упражнений учебника упражнением следующего содержания: "Придумайте какого-нибудь исполнителя, все равно какого, выпишите систему его команд и составьте два разных алгоритма для него". Эта задача почему-то очень привлекательна для школьников: что-то среднее между сочинением сказок и научно-фантастических рассказов. Все позволено, никаких ограничений

Лекция 3

Будем двигаться по учебнику*. Я буду акцентировать внимание на наиболее значимых, на мой взгляд, моментах, понятиях. Буду расставлять акценты, которые в самом учебнике отсутствуют.

Введение

Все "Введение" посвящено раскрытию понятия информатики как науки, изучающей обработку информации с помощью ЭВМ. Три параграфа "Введения" объясняют:

1) что такое информация,
2) что такое ЭВМ,
3) что такое обработка информации на ЭВМ.

§ 1. Информация и обработка информации

Начинается параграф с воодушевляющего введения, суть которого сводится к лозунгу "Информатика — это очень важно".

Если от этого неформального введения отвлечься, то параграф посвящен понятиям "информация" и "обработка информации" (пока без ЭВМ).

Таким образом, основные цели параграфа таковы:

0) сформировать общее ощущение места и значимости информатики (триада "вещество, энергия, информация");
1) показать отличия понятия "информации" в информатике от бытовой трактовки информации как смысла или значимости сообщения;
2) дать представление об "обработке информации" как о техническом преобразовании, производимом по строгим правилам и не связанном с восприятием информации или ее осмыслением.

Здесь сразу надо отметить самое главное — в этом параграфе понятия "информация" и "обработка информации" даются как технические понятия информатики. Основной акцент всего параграфа — отличие формальных понятий информатики от бытового смысла этих слов. Например, разница между "бытовым" представлением об информации как о смысле или значимости и "техническим" как о последовательности битов и байтов.

Для усвоения этого различия в учебнике ***вначале*** речь идет о технических вещах — о двоичном кодировании информации. Причем что такое ***информация***, в этот момент не говорится вообще — вопрос тщательно обходится, "замазывается". Вместо этого обсуждается проблема измерения ***объема информации*** в случае, когда информация ***уже закодирована двоично.***

А потом сообщается, что никакого строгого определения понятия "информация" нам и не нужно. Представляйте, мол, себе порцию информации как последовательность нулей и единиц, а количество информации — как длину этой последовательности. И все.

Я хочу заметить, что при всей своей простоте такой подход совершенно честный. Нам действительно ничего больше и не нужно. Житейские представления о том, что информация может быть актуальной или устаревшей, верной или неверной и т.д., если даже и можно формализовать, все равно бессмысленны в школьном курсе информатики. С понятиями своевременности или нужности информации нам делать абсолютно нечего.

Наконец, и об этом в учебнике говорится прямо, мы не можем дать строгое определение понятия "информация". И не потому, что оно длинное или сложное, а потому, что в "большой" информатике понятие информации считается первичным, неопределяемым.

Отличия понятия "информация" в информатике от бытового смысла этого слова

Итак, первый концентр параграфа, первое, на чем вы должны заострить внимание, — это отличие "технического" понятия "информации" как последовательности битов, байтов, символов от бытового значения "информации" как какого-то содержания, смысла. Если мы, скажем, получаем телеграмму: "ПРИЕЗЖАЮ ПОНЕДЕЛЬНИК ДЕСЯТЬ УТРА = МАМА", а мама час назад позвонила и все уже сказала, то в бытовом смысле эта телеграмма для нас никакой информации не несет: мы и так знаем, когда мама приедет. То есть с бытовой точки зрения мы не получаем никакой информации, получаем "ноль" информации.

А с технической точки зрения — с точки зрения информатики — мы можем подсчитать, сколько символов в этой телеграмме, сколько затрачено времени на ее передачу по телеграфным линиям и т.д., и это совершенно не зависит ни от текста телеграммы, ни от состояния головы получателя телеграммы. С точки зрения информатики информация есть в любой "белиберде". Можно подсчитать объем этой "белиберды" в байтах, можно измерить время передачи ее по телеграфной линии, поделить полученные числа и получить скорость передачи информации. И все это можно делать, не вникая в смысл.

Это "техническое" понятие информации как последовательности битов или байтов, никак не связанное ни со смыслом, ни с значимостью сообщения, — вот главное, что здесь вводится.

---

* **От редакции.** Напоминаем, что курс лекций посвящен учебнику *А.Г. Кушниренко, Г.В. Лебедева, Р.А. Сворень* "Основы информатики и вычислительной техники". Содержание лекции гораздо глубже обычного методического комментария к главе учебника. На наш взгляд, познакомиться с ней будет полезно любому преподавателю.

Как всегда (“черепаха” курса), чтобы это различие усвоить, школьники должны поработать, что-нибудь поделать. И именно для этой цели используется двоичное кодирование. Двоичное кодирование здесь играет роль того средства, повозившись с которым (покодировав, пораскодировав) школьники должны освоить, как-то воспринять понятие информации как последовательности байтов, битов. Тут полезно повозиться с нулями и единицами на любом материале, поиграть в разные шифры, в шифровку и дешифровку. И чем бессмысленнее будут последовательности символов при этой возне, тем лучше. Смысл и значимость должны отойти на второй план, чтобы остались только последовательности нулей и единиц или последовательности символов.

Вы можете также смело задействовать любые иные (взятые из жизни, из литературы или просто придуманные) способы кодирования/декодирования и передачи информации. Здесь можно использовать разные шифры, азбуку Морзе, морскую азбуку (поднятие/опускание флажков) и т.п. От последовательности поднятых/опущенных флажков, например, очень легко перейти к двоичному кодированию, к последовательности нулей и единиц. Можно и наоборот: вместо нулей и единиц использовать флажки. Скажем, в упражнении 6 можно предложить передать оценку с помощью флажка. Рекомендуется при этом постоянно подчеркивать формальность понятия “информация”, предлагать кодировать и передавать не только осмысленные, но и произвольные последовательности символов. Как правило, физические действия (поднятие/опускание флажков) способствуют восприятию единиц измерения информации как не зависящих от ее смысла.

Это первый и основной концентр параграфа. К нему примыкает введение единиц измерения информации, в которых никакого особенного содержания и никакой сложности нет. Нужно просто привыкнуть к новым словам (бит, байт, килобайт, мегабайт) и к их смыслу, для чего в конце параграфа есть несколько упражнений.

Понятие “обработки информации” в информатике

Второй концентр параграфа — это понятие ***обработки*** информации. Здесь также надо показать, что под “обработкой информации” понимается строгое формальное преобразование.

Если я в тексте все буквы “е” меняю на буквы “и” — это формальное преобразование, т.е. обработка информации с точки зрения информатики.

А вот если я посмотрел кино и рассказал другу, какое оно замечательное, то это — с точки зрения информатики — назвать обработкой информации нельзя. Моя оценка качества кино — “человеческое” действие, оценка информации, а не ее формальная обработка по четким и однозначным правилам.

Второй концентр нацелен на то, чтобы сформировать представление об “обработке информации” как о техническом преобразовании, проводимом по строгим формальным правилам. В самом параграфе это небольшой раздел (1.7). Усвоение этого понятия должно происходить в ходе выполнения упражнений, приведенных в конце параграфа.

Игра в “черные ящики”

Я хочу особо обратить ваше внимание на игру в упражнениях 12—13*. Мы узнали об этой игре в г. Переславле-Залесском от авторов “Роботландии” — замечательной программной системы для обучения информатике младшеклассников. (Я не знаю, существовала ли эта игра раньше или была придумана переславцами.) Имеются и компьютерные реализации этой игры, но, по-моему, без компьютера играть интереснее. Переславцы называют эту игру “Черный ящик”. Термин “черный ящик” возник в кибернетике и означает устройство, получающее информацию на входе и выдающее переработанную информацию на выходе, причем внутреннее устройство “ящика” нам не известно (отсюда и термин “черный”, т.е. непрозрачный).

Содержание игры состоит в том, что учащимся предлагается разгадать, как работает тот или иной “черный ящик” (в упражнениях учебника роль “черного ящика” играет Петя): на “вход” “ящика” поступает информация, последовательность символов, на “выходе” получается другая, учащиеся должны догадаться, как этот “черный ящик” работает.

Например, на “вход” поступает информация “1”, на “выходе” получается “2”. На “входе” — “17”, на “выходе” — “18” и т.д. Сделав несколько проб, учащиеся начинают догадываться: “черный ящик” прибавляет единицу. Другой “черный ящик” может, скажем, считать число символов в строке (упражнение 12б).

Хотя внешне это облечено в игровую форму, тем не менее основное и фундаментальное содержание, которое здесь вводится, — формальность правил обработки информации.

В “черные ящики” можно играть на уроках: первоначально учитель придумывает “черные ящики” и играет их роль. Например, я говорю, что я “черный ящик”, вы, ученики, можете мне задавать входные последовательности. Когда ученик диктует, я выписываю на доске входную последовательность, рисую стрелочку, как в учебнике, по своему внутреннему правилу преобразую, выписываю ответ. На доске получается строка “вход → выход”, под ней вторая и следующие. Ученики всем классом смотрят на доску и пытаются догадаться, как же работает “черный ящик”.

---

* Упражнения 11—13 приведены на стр. 15. — **Прим. ред.**

структурирования алгоритмов. Поэтому и появился Чертежник, для которого потом можно будет составлять вспомогательные алгоритмы с аргументами. Скажем, алгоритм рисования квадрата произвольных размеров.

Второе соображение для введения Чертежника состоит в том, что, рассказывая про команды с аргументами и позже про вспомогательные алгоритмы с аргументами, можно устроить пропедевтику понятия “величина”, т.е. объяснить, как ЭВМ запоминает аргументы алгоритма, привести модель памяти ЭВМ и т.д. В этом параграфе это все в зачаточном состоянии — сколь-нибудь связные объяснения и модель памяти ЭВМ появятся в следующем параграфе. Желательно, чтобы к этому времени какие-то слова и понятия уже в головах учеников пусть в зачаточном состоянии, но были.

Исполнителей должно быть несколько

Есть еще один методический аспект, о котором я должен упомянуть. Мы считаем, что исполнителей в школьном курсе должно быть не меньше двух. Просто чтобы материал не был связан только с одним исполнителем. Чтобы в схеме программного управления можно было нарисовать, что ЭВМ управляет и Роботом, и Чертежником. Чтобы можно было написать алгоритм, управляющий двумя исполнителями одновременно. Например, алгоритм, при выполнении которого Робот идет по коридору, измеряя уровень радиоактивности, а Чертежник строит соответствующий график радиоактивности. То есть чтобы у нас появились алгоритмы, управляющие более чем одним исполнителем. Иначе очень трудно понять, зачем ЭВМ отделена от исполнителей и кто же, собственно говоря, понимает программу — Робот или ЭВМ.

Если вы читали книги Г.А. Звенигородского или самые первые школьные учебники информатики А.П. Ершова, то, наверное, помните, что там исполнители “обучались” командам языка (**`если`**, **`цикл пока`** и т.д.). Каждый исполнитель “знал” и понимал весь язык программирования, а никаких ЭВМ не было вообще. Если вам нужно было сделать что-то двумя исполнителями, то вы должны были конструировать нового, который включал бы оба предыдущих и знал весь язык.

У нас этой путаницы нет. Исполнитель имеет только те кнопки, которые имеет. Робот и Чертежник исполняют ***команды,*** а не ***программы.*** А со всеми ***алгоритмами,*** со всеми ***программами*** разбирается ЭВМ. Именно ЭВМ понимает алгоритм и командует — управляет исполнителями. И для прояснения этого нужны как минимум два исполнителя.

Наконец, некоторые педагоги-практики рекомендовали нам ввести не двух исполнителей, а гораздо больше. В программном обеспечении мы это сделали. В базовом комплекте системы “КуМир”, кроме Робота и Чертежника, есть еще упомянутый во Введении Вездеход, который перемещается по непрерывной плоскости, Двуног, Строитель, Редактор и ряд других исполнителей (см. цветную вкладку учебника). Эти исполнители успешно использовались на кружках и факультативах, в летних школах программирования и т.д. Но в основном тексте учебника, опираясь на принцип “ничего лишнего”, мы ввели только двух исполнителей, поскольку их оказалось достаточно. Учитель при желании может использовать в некоторых темах и других исполнителей, но никаких конкретных рекомендаций я вам не дам. Главное — помнить, что наша цель — развитие алгоритмического стиля мышления, а не изучение исполнителей самих по себе. Если интересные исполнители могут помочь в достижении главного — их надо использовать, если лишь отвлекают и занимают время — от них следует отказываться.

Алгоритмы рисования букв — это подготовка к введению вспомогательных алгоритмов

Для подготовки к введению в следующем параграфе вспомогательных алгоритмов здесь разбираются алгоритмы рисования букв (алгоритм А7, *рис.* 12—13 учебника). Основное, на что надо обратить внимание: алгоритмы составлены таким образом, чтобы при их последовательном выполнении получалось слово, т.е. буквы шли бы в ряд с некоторым промежутком между ними. Вопрос о том, кто и как организует это последовательное выполнение, в § 5 намеренно не поднимается. Фактически он откладывается до § 6, чтобы дать школьникам возможность сперва освоиться с Чертежником.

Рисование букв можно заменить чем-то другим, но с буквами легко объяснять, почему к алгоритмам предъявляются такие требования. Ясно, зачем нужно рисовать букву за буквой. Ясно, почему важно уметь рисовать букву с любого места, — чтобы можно было писать потом из этих букв слова и т.д.

Графические пакеты

В принципе если не строго следовать учебнику, то Чертежника можно заменить на какой-нибудь графический пакет, который умеет закрашивать кружки, прямоугольники и прочее. И рисовать разные красивые картинки (именно так обычно и преподают на Бейсике). Впрочем, красивые картинки можно рисовать и в системе КуМир, используя школьный алгоритмический язык и графический пакет (исполнитель) ГраТекс, умеющий рисовать и закрашивать все эти прямоугольники, круги, эллипсы, дуги и пр. По содержанию это будет то же, что и Чертежник, но картинки на компьютере получаются более красивыми и привлекательными.

```
алг 8ж
   дано | Робот в клетке А
        |(см. рис. 8"ж", с. 31 уч-ка)
   надо | Робот в клетке Б
        |(см. рис. 8"ж", с. 31 уч-ка)
нач
закрасить
вниз;вниз;вправо;вверх;закрасить;вверх;закрасить
вниз;вниз;вправо;вверх;закрасить;вверх;закрасить
вниз;вниз;вправо;вверх;закрасить;вверх;закрасить
вниз;вниз;вправо;вверх;закрасить;вверх;закрасить
вниз;вниз;вправо;вверх;закрасить;вверх;закрасить
кон
```

При записи на бумаге (в тетрадях) или на доске ученики обычно ленятся пять раз повторять одну и ту же строку. Чаще всего они записывают одну строку, справа рисуют фигурную скобку и пишут: “5 раз”. Либо придумывают что-то аналогичное. За любую такую форму записи их нужно хвалить, заметив, что позже мы изучим строгую форму записи повторяющихся кусков алгоритма — цикл “*n* раз”.

Если же ученики будут работать на ЭВМ в системе “КуМир”, то, набрав повторяющуюся строку в первый раз, они смогут ее “запомнить” (нажав клавишу F3), а затем “вспомнить” нужное число раз (нажимая клавишу F4). Так что набирать одну и ту же строку несколько раз им тоже не придется.

На самом деле здесь ввиду легкости материала параграфа все основные упражнения сделаны на выделение повторяющихся элементов. Вся их сложность состоит в том, что в них во всех надо выделить повторяющийся фрагмент и его записать. Где-то получается это попроще, где-то посложнее. Где-то надо отдельно записать начальные или конечные нерегулярности (например, в приведенном выше алгоритме 8“ж” требуется отдельно закрасить первую клетку).

В целом же эти упражнения можно рассматривать как пропедевтику понятий вспомогательного алгоритма и цикла.

§ 5. Исполнитель Чертежник и работа с ним

Сам по себе Чертежник — это обычный графопостроитель, типичное широко распространенное устройство ЭВМ, используемое в инженерии и в науке. Он умеет поднимать и опускать перо, ставить его в заданную точку и прямолинейно перемещаться из заданной точки в другую. Путем такого опускания и перемещения пера можно рисовать картинки, составленные из отрезков. Запустите гипертекст “Знакомство с исполнителем Чертежник в системе “КуМир” — и вы увидите примеры работы этого исполнителя.

В принципе можно пройти весь курс и без исполнителя Чертежник, пропустив все, что с ним связано. Введение исполнителя объясняется несколькими соображениями, которые я сейчас изложу.

Команды с аргументами

Первое — возможно более раннее появление команд с аргументами. Команды Робота (“вправо”, “влево” и пр.) — это простые команды без всяких аргументов. У команд Чертежника есть аргументы. Ему можно скомандовать поставить перо в точку с координатами, скажем, (1,3) или сместиться на вектор с координатами (1,1). Чтобы показать команды с аргументами, а точнее говоря, чтобы к ним привыкнуть, прежде чем они понадобятся по существу, и введен Чертежник.

МЕТОДИЧЕСКОЕ ОТСТУПЛЕНИЕ. Подчеркну еще раз — в нашем курсе до § 11 нет “величин”. Нет описания “величин”, нет оператора “присваивания”. Весь курс строится на управлении Роботом и Чертежником. Более того, вспомогательные алгоритмы вводятся и объясняются раньше всех управляющих конструкций, даже раньше простейших “циклов”, скажем, цикла “5 раз”. И управляющие конструкции, и величины появляются позже. Это существенное отличие нашего курса от большинства других. И именно Чертежник позволяет ввести и использовать вспомогательные алгоритмы с ***аргументами*** до начала работы с величинами и даже до команд ветвления и повторения.

Забегая вперед, хочу заметить, что вспомогательные алгоритмы (§ 6) были введены до управляющих конструкций с вполне определенной целью. Опыт показывает, что первые понятия в новой области запоминаются гораздо глубже, чем последующие (в психологии это, кажется, называется *imprinting*, что в переводе означает “впечатывание”, “запечатление”). А одна из распространенных проблем школьных (да и многих других) курсов состоит в том, что школьники не усваивают понятие “вспомогательный алгоритм” и часто пишут алгоритмы “одним куском”. Это и понятно — в иерархии основных понятий алгоритмизации (см. с. 29 пособия) “вспомогательные алгоритмы” стоят во второй колонке, а циклы и величины — в первой.

Вот почему мы постарались построить курс так, чтобы понятие вспомогательного алгоритма “впечаталось” хотя бы за счет отсутствия других конструкций на старте. При нашем построении курса понятие вспомогательного алгоритма с параметрами появляется сразу после понятия алгоритм. То есть с самого начала — до всего остального — вводятся средства

Пример игры в “черный ящик”

Если хотите, можем сыграть. Хотите?

Давайте.

Я задумал некоторое правило преобразования. Задавайте последовательности. (Из зала: “3”, “9”, “– 7”, “0”...). На доске появляются 4 строки:

3	→	0
9	→	0
– 7	→	0
0	→	0

Вы можете меня остановить, если догадались, и сказать:

“Все — я знаю”. Когда какой-то ученик произнес: “Все”, — возможны два пути продолжения игры. Один путь — ученик вслух формулирует правило. Например:

Лектор (обращается к слушательнице): Скажите ваше правило.

Ответ: Вы умножаете на ноль.

Я, поскольку знаю, как “черный ящик” устроен, говорю “правильно” или “неправильно”. Если “правильно”, то игра заканчивается. Другой путь позволяет игру продолжить — я прошу: “Не называйте свое правило, а скажите мне, что оно даст на какой-то последовательности, которой еще нет на доске”. Например:

Лектор (обращается к слушательнице): Вы можете сказать, что дает ваше правило для слова “мама”?

Ответ: Нет.

В такой ситуации я говорю ученику: “Неверно, мое правило для слова “мама” работает и дает выход “2”:

мама → 2

Значит, ученик еще не догадался.

Из зала: “Бабушка”.

Лектор: Бабушка → 3

Из зала: Если число, то умножаем на ноль, если текст, то считаем число слогов.

Лектор: Я задумал число гласных букв. Видимо, это одно и то же — давайте мы пока закончим.

Если бы правило не описали, то мы могли бы поиграть еще. Я дал бы вам пару слов, убедился, что ваше правило дает тот же результат, что и мое, объявил бы вас отгадавшим (не зная вашего правила) и продолжал бы играть с остальными слушателями. То есть если в классе один ученик догадался, то можно продолжать игру. Не обязательно, как только первый додумался, заканчивать. В классе вы сами сообразите, когда продолжать играть, а когда пора закончить или устроить совместное обсуждение результатов.

Использование игры при изложении понятия “обработка информации”

Для раскрытия понятия формальной обработки информации я хочу привести один пример, который нигде в учебнике не фигурирует и который, по-моему, вы обязательно должны использовать на уроках. Представьте себе, что я играю роль “черного ящика” в классе. Ученики задают мне входную информацию, а я в ответ пишу “1” или “0” по следующему “правилу”: если “вход” мне нравится, то я пишу “1”, если нет, то “0”.

Через некоторое время, когда ученики совершенно уже запутаются и “сдадутся”, я раскрываю им свое “правило”: “1” — если нравится и “0” — если не нравится. Обычно класс хором взрывается: “Это нечестно”.

Выражение “это нечестно” является очень содержательным, поскольку в данном случае нет никакой формальной обработки информации, а есть ее “человеческая оценка”. И приведенный пример позволяет четко противопоставить эти два понятия. Это контрпример — пример не обработки, а оценки информации, и именно поэтому он так важен.

Другой очень важный пример “ящика”, демонстрирующий в каком-то смысле противоположную грань термина “обработка информации”, — это “черный ящик”, который выдает на выходе “да” и “нет” по очереди, через раз, независимо ни от какой входной информации. Когда ученики этот “ящик” разгадают, попросите их сначала самих подумать, является ли такое преобразование “обработкой” информации в смысле информатики. Как правило, они начинают путаться, поскольку привычный бытовой смысл термина “обработка” сопротивляется его применению в данной ситуации. А потом объясните им, что: а) они это правило смогли угадать, б) изготовить техническое устройство — “ящик”, который будет осуществлять это преобразование, можно, т.е. “преобразование информации” происходит по совершенно строгому, формальному правилу. Таким образом, речь идет, безусловно, об “обработке информации” в смысле информатики, хотя это преобразование от входной информации и не зависит.

Забегая вперед, замечу, что в игре в “черный ящик” задача учеников состоит в том, чтобы, глядя на результаты преобразования информации, угадать ***алгоритм*** преобразования. Именно поэтому игра в “черные ящики” является основным методическим приемом при прохождении темы “Формальная обработка информации”.

Как и когда следует играть в “черный ящик” в курсе информатики

Резюмируем основные рекомендации по организации игры в “черный ящик”.

1. В классе рекомендуется играть одному против всех. Это значит, что учитель или один из учеников, придумавший “черный ящик”, стоит у

доски и выписывает все задаваемые ему вопросы (“входную информацию”) и свои ответы на них (“выходную информацию”). Все остальные, задавая вопросы и анализируя ответы, пытаются угадать алгоритм.

2. Когда кто-либо сообщает, что он уже угадал алгоритм, можно, не оглашая правила, провести проверку, которая состоит в следующем: учитель (играющий роль “черного ящика”) задает контрольные вопросы угадавшему ученику, а тот, преобразуя информацию *по своему алгоритму,* выдает ответы. Если ответ неверен — значит, алгоритм еще не угадан, учитель (“черный ящик”) должен привести правильный ответ, и игра продолжается. Если ответ верен (это еще не означает, что алгоритм учителя действительно угадан: возможно и просто случайное совпадение), то тем не менее учитель может еще какое-то время поиграть с остальными учениками. Такой подход позволяет ученикам, угадавшим алгоритм, проверить себя, не раскрывая алгоритм другим играющим, что и дает возможность продолжить игру, даже когда несколько учеников уже знают алгоритм.
3. Игра в “черный ящик” наибольший эффект приносит в начале курса. Однако ее можно использовать и в дальнейшем:
   а) для введения новых алгоритмических конструкций (с помощью “ящиков”, алгоритмы которых содержат циклы и развилки);
   б) для разрядки и повышения мотивации, играя в соответствующий “ящик” перед составлением того или иного алгоритма преобразования информации (особенно это относится к литерным величинам в § 15).
4. Наконец, игру можно использовать, если в конце урока остались 2—3 свободные минуты, а также на переменах, школьных вечерах. Полезно устроить конкурс на придумывание лучшего “черного ящика” с награждением победителя и т. п.

Вообще эта игра нравится в любом возрасте, даже в 11-х классах в нее играют с удовольствием. И если сначала “черные ящики” придумывает учитель, то потом ученики начинают предлагать “свои”. Приходит перед уроком ученик и говорит: “Я придумал замечательный новый “черный ящик”. А вы ему в ответ: “Ну хорошо, я в конце 3 минутки оставлю, и мы поиграем”. И в конце урока встает этот ученик и исполняет роль “черного ящика”, а весь класс отгадывает. Возникает даже некоторая соревновательная атмосфера, когда они придумывают все новые и новые “черные ящики”.

Итак, первый параграф учебника концентрируется вокруг двух пунктов. Первое: информация есть последовательность байтов, битов, символов и не зависит от смысла. Второе: обработка информации есть преобразование таких последовательностей по формальным правилам, никакие “человеческие” категории типа “нравится”, “не нравится”, “считаю важным”, “считаю не важным” и прочие не допускаются.

Ответы к упражнениям про “черные ящики”

Поскольку я столько раз повторил вам про важность упр. 11—13 (“черные ящики”), приведу ответы к ним.

Упр. 11

а) алфавит;

б) натуральный ряд: каждое следующее число на 1 больше предыдущего;

в) степени двойки: каждое следующее число вдвое больше предыдущего;

г) последовательность Фибоначчи: третий элемент есть сумма первого и второго, четвертый — сумма второго и третьего, пятый — сумма третьего и четвертого и т.д., каждый следующий — сумма двух предыдущих;

д) берется натуральный ряд, и выписываются цифры, т.е. до девяти идут сами числа, а дальше — цифры чисел 10, 11, 12, ... (1, 0, 1, 1, 1, 2, ...);

е) первый символ просто вычеркивается;

ж) первые буквы слов: “один”, “два”, “три”, “четыре”, “пять”, “шесть”, “семь”, “восемь”, “девять”, “десять” и т.д.;

з*) эта последовательность устроена так: каждое следующее число описывает распределение повторяющихся цифр в предыдущем. Например, первый член последовательности — это одна (1) единица (1), поэтому второй член будет 11. В числе 11 две (2) единицы (1), поэтому третий член равен 21. В числе 21 одна (1) двойка (2) и одна (1) единица (1), поэтому четвертый член равен 1211. В числе 1211 одна (1) единица (1), одна (1) двойка (2), а потом две (2) единицы (1). Поэтому, выписав подряд все цифры в скобках, мы получим число 111 221. Соответственно, следующим будет 312 211 и т.д. Обычно четвероклассники разгадывают этот “ящик” очень быстро, шестиклассники — с трудом, а десятиклассники не разгадывают вообще.

Упр. 12

а) каждая буква заменяется на следующую по алфавиту: “а” на “б”, “б” на “в”, ... , “м” на “н”, ... , “я” на “а”;

б) число символов;

в) считается число согласных плюс число гласных;

г) осуществляется перестановка (которая хорошо видна на “ЭВМ”): первый символ переставляется в конец.

Упр. 13

а) число уменьшается на единицу;

б) берется ближайшее (с недостатком) число, кратное трем (то есть из чисел 0, 1 и 2 получается 0; из чисел 3, 4 и 5 — 3; из чисел 6, 7 и 8 — 6; из чисел 9, 10 и 11 — 9 и т.д.);

> это и один из основных методических приемов при преподавании курса. При введении новых алгоритмических конструкций, при прохождении нового материала, да и просто при решении конкретной задачи один из базовых приемов — спросить у школьников: “Что бы вы делали, если бы работали вручную, если бы управляли Роботом сами?” Они отвечают: “Нужно нажать на такие-то кнопки, сделать то-то и то-то”, т.е. разбираются с новой ситуацией. Дальше мы им говорим: “Хорошо, теперь у нас та же ситуация. Давайте составим ***алгоритм. Опишите,*** что должна делать ЭВМ, чтобы повторить то, что делали вы”.

Такое противопоставление, такой переход является основным в курсе, и § 4 в этом смысле является важным. Он вводит обе схемы. И Робота, самого по себе, с понятными командами, и понятие алгоритма, записи программы для ЭВМ — в данном случае на школьном алгоритмическом языке.

Основная трудность параграфа

Основная трудность параграфа состоит в том, что, пока у нас только пять команд, алгоритмы получаются уж очень простыми. Поэтому на старте, особенно если занятие проводится не за компьютером, может быть вообще непонятно, зачем эти алгоритмы как-то оформлять. Но если мы хотим поручить исполнение алгоритма ЭВМ, то следует (как бы это ни было просто) писать по правилам.

Первый алгоритм учебника, алгоритм (А1), тривиален. И постановка задачи, и решение очевидны — надо два раза скомандовать Роботу “вправо” и один раз — “вниз”. Содержательно задача совершенно неинтересная. Поэтому тут придется на учеников нажать, сказать: “Привыкайте — это форма записи. Мы ведь не собираемся сами всю жизнь алгоритмы выполнять, а хотим это дело перепоручить ЭВМ. А ЭВМ подавай стандартную форму записи алгоритма, так что придется освоить язык, понятный ЭВМ”.

Конечно, лучше просто все делать на ЭВМ. Все эти комментарии (4.5), ошибки (4.7) и запись нескольких команд в одной строке (4.8) на практике осваиваются за несколько минут. Вы можете также запустить гипертекст “Знакомство со школьным алгоритмическим языком” в системе “КуМир” — и демонстрационная программа сама все покажет и даже потребует понажимать на кнопки, чтобы ученик освоил поскорее все технические детали.

Основная тема этого параграфа — переход от непосредственного управления Роботом к записи алгоритма. В данном случае — к записи на школьном алгоритмическом языке.

Ну а дальше идет самое важное — упражнения. Начнем с задач на анализ алгоритмов.

Умение рассуждать про алгоритм в общем виде, предсказывать результат его работы, глядя на текст алгоритма, — важный компонент алгоритмической культуры

В задачах 1а и 1б ученик должен выступить в роли ЭВМ и выполнить алгоритм. Попутно ему придется сделать кое-какую чисто человеческую работу: придумать названия алгоритмов и комментарии. Эти задачи лучше решать без ЭВМ, иначе она возьмет на себя ту работу, которую должен выполнить ученик.

Задачу 2 тоже лучше делать на бумаге или обсудить у доски, чтобы не отвлекаться на нажатия кнопок.

Задачи 3а и 3б требуют мысленного выполнения алгоритма, они довольно простые.

Задача 4 очень хорошая, я ее настоятельно рекомендую. Это упражнение на формальность исполнения и на анализ программы независимо от того, куда и как ходит Робот. Эта задача и на восприятие (анализ) того алгоритма, который приведен в учебнике, и на составление нового алгоритма. В ее условие и решение вовлечено большинство понятий, введенных в параграфе. Прежде чем задавать эту задачу на дом, полезно обсудить ее условие у доски.

Задачи 6, 7 на с. 29 и 7—12 на с. 31—32 (из-за ошибки в нумерации в учебнике встречаются два упражнения с номером 7 и два упражнения с номером 8) имеют формально-логический характер. Их цель — воспитать представление об алгоритме как о некоторой информации, которую можно исследовать и перерабатывать. Попутно работа над этими задачами заставляет продумать и команды Робота, и понятие алгоритма, и схему программного управления. Скажем, задача 11 заставляет задуматься о всевозможных алгоритмах управления Роботом, о том, что из каждого из них получится при перестановке местами двух команд, и о многом другом.

Задачи начиная с 7-й на с. 31 и дальше не вполне алгоритмические, я бы сказал, что они относятся к “комбинаторике на клетчатом поле Робота”.

Задачи на составление алгоритма

Два семейства задач — 5 “а”—“и” и 8 “а”—“з” (с. 31) — посвящены составлению алгоритмов. Их лучше делать на ЭВМ. Эти задачи построены на одной идее: алгоритмы их решения состоят из нескольких повторяющихся одинаковых кусков. Используя правило записи нескольких команд в одной строке, каждый такой кусок можно уместить в одну строку. Тогда алгоритм будет содержать несколько одинаковых строк, подобно алгоритму (А2) на с. 28 учебника. Например, для задачи 8 “ж” алгоритм будет содержать пять одинаковых строк.

ПРОГРАММНОЕ ОТСТУПЛЕНИЕ. Все это можно наглядно продемонстрировать на экране компьютера в гипертексте “Знакомство с Роботом” в системе “КуМир”. На экране будут изображены и поле Робота, и пульт управления, и можно будет мышкой нажимать кнопки на пульте (см. *рис.* 2) и наблюдать результаты выполнения команд.

Рис. 2

Поскольку учебник не содержит никаких обоснований, никакой мотивации для введения схемы программного управления Роботом, я приведу пару таких обоснований. Естественно, вы можете придумать, найти и использовать другие. Мои примеры будут задействовать команды обратной связи, но в этом месте их можно подробно не объяснять. А можно ввести и объяснить по ходу изложения.

Первый пример — космический

Предположим, мы посылаем Робота на какую-нибудь далекую планету. Пусть мы — на Земле, а Робот — на клетчатом поле на Нептуне. Тогда если мы сами командуем Роботом с Земли, то наша команда дойдет до Нептуна примерно за 4 часа, после чего еще столько же времени ответ будет идти назад. Скажем, мы нажали на кнопку **справа стена?**, через 4 часа эта команда дойдет до Робота, Робот проверит, есть ли справа стена, и еще 4 часа ответ будет идти обратно на Землю. И только после этого мы сможем его проанализировать, принять решение, куда двигаться дальше, и послать соответствующую команду Роботу (которая еще 4 часа будет идти до Робота).

Получается такое неторопливое управление — меньше 6 команд в сутки. Могут быть ситуации, в которых такой темп недопустим. Кроме того, на Нептуне за эти 8 часов со стенками может что-нибудь случиться: одни стенки могут исчезнуть, а другие — возникнуть.

Чтобы управлять Роботом с нормальной скоростью, можно сделать следующее: вывести на орбиту вокруг Нептуна спутник и командовать Роботом со спутника. От спутника до Робота сигнал будет идти доли секунды — задержек не будет, и процесс ускорится.

Поскольку, однако, мы не собираемся держать на борту спутника экипаж для управления Роботом, на спутнике надо установить ЭВМ. Тогда мы с Земли сможем послать на спутник программу, алгоритм управления Роботом. Эта программа будет идти до спутника те же 4 часа, но зато потом ЭВМ сможет командовать Роботом без нашего участия, выдавая команды и получая ответы за доли секунды.

МЕТОДИЧЕСКОЕ ОТСТУПЛЕНИЕ. Можно, конечно, ЭВМ смонтировать и на Роботе. Но поскольку одна из наших целей — четкое разграничение того, что делает исполнитель, а что — ЭВМ (см. пункт 6.6 — разделение труда между ЭВМ и исполнителями), то предпочтительнее (если это возможно) держать ЭВМ “на некотором расстоянии” от Робота.

Конечно, это достаточно искусственное построение, хотя оно вполне адекватно отражает суть дела. Мы использовали это введение при преподавании программирования на мехмате МГУ в 1980 году (с луноходом, позже переименованным в “Путника”).

Второй пример — чернобыльский

Можно привести много других примеров, демонстрирующих необходимость программирования, т.е. программного управления исполнителями. Например, Робот должен выполнить некоторую работу недалеко, но в сильно радиоактивной зоне, в которой человек находиться не может. А радиосвязь из-за больших помех тоже не работает. В таком случае нужно смонтировать ЭВМ прямо внутри Робота, за защитным экраном. И опять, еще до входа в опасную зону, нужно будет поместить в ЭВМ программу, алгоритм управления Роботом.

Это объяснение хорошо согласуется с командами Робота по измерению температуры и радиации, задачами по радиационной разведке, закрашиванию радиоактивных клеток коридора и другими примерами учебника (см., например, алгоритмы “вход в радиоактивную зону” (А34), “разметка опасных клеток коридора” (А40), “радиационная разведка коридора” (А66) и др.).

Схему программного управления следует постоянно противопоставлять управлению Роботом “вручную”

Каковы бы ни были обоснования, главная задача учителя здесь — ввести схему программного управления и противопоставить ее непосредственному управлению исполнителями “вручную”.

МЕТОДИЧЕСКОЕ ОТСТУПЛЕНИЕ. Противопоставление схемы программного управления непосредственному управлению “вручную” —

в) количество цифр в числе;

г*) эту трудную задачу придумали третьеклассники в Переславле-Залесском. Ответ здесь — количество “дырок” в цифрах числа. Например, в числе 687 три “дырки” — одна в “6” и две в “8”. Количество “дырок” зависит от цифр и их написания, в “1” “дырок” нет, в “2” и “3” — тоже нет, в “4” (если она сверху “замкнута”) — одна, в “5” — нет, в “6” — одна, в “7” — нет, в “8” — две, в “9” — одна.

§ 2. Электронные вычислительные машины

Этот параграф совсем маленький и не содержит ничего, кроме слов. Если ученик слышит их впервые, то потом ему легче будет услышать их во второй раз. Никакого иного содержания в параграфе нет. Тут имеется небольшая, краткая история вычислительной техники, как обычно, про Лейбница, Беббиджа и т.д. А потом говорится, что в состав любой ЭВМ входят память, процессор, клавиатура, монитор, они обмениваются между собой информацией и т.д. Этот урок обычно совмещают с выходом на ЭВМ. Там просто все показывают “в натуре” (это — клавиатура, это — монитор и пр.), а потом начинают работать с клавиатурным тренажером или еще с чем-то.

Вы можете задать ученикам прочесть этот параграф самостоятельно, а потом ответить на их вопросы, если они будут. Упражнения в этом параграфе в основном арифметические. Например, говорится, что оперативная память ЭВМ “Корвет” — 64 Кбайта. Спрашивается: сколько примерно страниц нашего учебника можно уместить в эту память? Понятно, надо сначала прикинуть, сколько примерно килобайт занимает одна страница, а потом разделить и получить ответ.

Разумно все эти задачи переформулировать для той ЭВМ, на которой класс будет работать. Смысл этих арифметических задач в том, чтобы ученики примерно запомнили некоторые характеристики ЭВМ. Если уж мы поставили цель сформировать у школьников какое-то представление о мире, в котором они будут жить, то они должны уметь прикинуть, помещается учебник на дискету или нет. То есть в процессе решения этих не слишком важных задач у учеников формируются какие-то связи между характеристиками ЭВМ и вещами, встречающимися в повседневной жизни.

В этом смысле, я повторяю, здесь нет ничего, никакой особенной методики, никакого особенного содержания, кроме новых слов. Во всех сжатых курсах этот параграф обычно опускается или задается для самостоятельного чтения, чтобы не занимать время.

§ 3. Обработка информации на ЭВМ

Этот параграф завершает “раскрытие” определения информатики, которое было дано в преамбуле. Уже изложены понятия “информация”, “обработка информации”, “ЭВМ”. Теперь — и это тема третьего параграфа — “обработка информации на ЭВМ”.

Отмечу всего два важных момента.

Первый — ЭВМ всегда работает по программе: сначала мы должны ***записать*** то, что надо сделать, и только потом ЭВМ написанную нами ***программу*** выполнит. Здесь в неявном виде, еще без исполнителей, вводится схема программного управления (с. 13 пособия). Но сама схема не рисуется и противопоставление с непосредственным управлением (за исключением самого начала параграфа) не делается.

Второй важный момент — программа должна быть записана специальным образом — так, чтобы ЭВМ эту программу поняла, т.е. на так называемом “языке программирования”. Приводятся три примера: одна и та же программа, записанная на Бейсике, Паскале и на школьном алгоритмическом языке (с. 20 учебника). Поясняется, что сама программа также является информацией и, как всякая информация, может храниться в памяти ЭВМ и обрабатываться с помощью ЭВМ. Более глубоко это будет раскрыто в главе 2 и в п. 21.6 главы 3.

Итак, в § 3 вводятся понятия программы (алгоритма) и программной обработки информации на ЭВМ. Кроме этих ключевых для раскрытия определения информатики понятий, параграф содержит также важные с мировоззренческой точки зрения разделы 3.4, 3.5. Они посвящены часто обсуждаемой роли информации, информатики, ЭВМ в нашей жизни сейчас и в будущем. Это все изложено в разделах 3.4, 3.5. Поскольку учебник предельно краток, я этот материал повторю.

Информатика и ЭВМ в жизни общества

Говорится следующее. На ранних стадиях автоматизации производства, когда хотели, скажем, новое производство автоматизировать, делали новые станки-автоматы, и всякий раз при налаживании нового производства нужно было переналаживать или даже заново делать станки-автоматы. Это трудоемкая, дорогостоящая работа. С появлением ЭВМ понятие станка-автомата и его переналадки коренным образом изменилось. Станок стал разделяться на механическую часть, способную выполнять элементарные операции, например, резание, сверление, шлифовку, штамповку, и на управляющую часть (ЭВМ), которая командует, в каком порядке эти операции следует выполнять для изготовления необходимой детали. Тем самым единый станок-автомат был заменен на две составляющие:

1) станок, который делает элементарные операции, но не знает, какие детали он будет производить, и

2) ЭВМ, которая управляет этим станком в нужной последовательности, делает те или иные действия и выпускает ту или иную деталь.

Что это дает? Теперь, если мы хотим начать выпуск другой детали по той же технологии, скажем, штамповкой или резанием, то мы должны менять не станок, а программу, т.е. задать, какие операции в какой последовательности надо делать. Другими словами, мы меняем только информационную часть производства, пишем новую программу, а материальная часть остается такой, какой и была. А завод начинает производить новые детали.

Таким образом, если раньше материальная и информационная составляющие были слиты вместе неразрывно и для изготовления новой детали нужно было что-то механически менять, переналаживать, изготавливать "в металле" новые станки, то сейчас составляющие разделились. Отдельно есть материальная составляющая, отдельно информационная.

Доля информационной части все возрастает, поскольку материальная не меняется — от нее ведь требуется только умение выполнять элементарные операции. Смена материальной составляющей нужна только при переходе на какую-то новую технологию. Мы можем от резания, штамповки перейти на порошковые технологии. Для такого перехода нужны новые станки, материальное что-то. Но если мы деталь вырезаем из листа металла и станок у нас универсальный, может просто резать, то с помощью новой программы мы в состоянии вырезать любые детали. Происходит разделение информационной и материальной составляющих в мире. И, следовательно, появляется работа в чисто информационной области — написание программ.

Информационная индустрия

Растет и число работников, обслуживающих информационную компоненту производства. Если программа для производства какой-то детали раньше уже использовалась на данном предприятии, то ее просто нужно поместить в память управляющей станком ЭВМ (взяв из архива, например, с магнитного диска). Если готовой программы нет, то ее можно купить на другом предприятии или поручить изготовление программы какой-нибудь программистской фирме. Таким образом, программы становятся и ***средствами производства***, и ***товаром***. А значит, готовые программы надо где-то хранить, организовывать "компьютерные склады" — базы данных. Для этих складов требуются "кладовщики" — программисты, обслуживающие эти базы данных. Нужны "магазины программ" и квалифицированные продавцы; рабочие, которые будут создавать новые средства производства — программы; начальники, которые разбираются как в самом новом товаре — программах, так и в процессе его производства — программировании. Необходимы учебные заведения и преподаватели, обучающие новым информационным технологиям в производстве и других сферах жизни общества. Именно этот рост информационной индустрии и вызвал к жизни лозунг А.П. Ершова, вынесенный в заголовок п. 3.5: "Программирование — вторая грамотность".

Запись ответа в виде алгоритма

Упражнения в конце параграфа во многом обращены к следующей главе учебника, к более глубокому пониманию алгоритма. Начинаются упражнения с задачи о "Волке, козе и капусте", с "поездов, которые должны разминуться на тупике", и т.д. Все они посвящены развитию алгоритмического стиля мышления. И, что самое главное, в них во всех ответом является ***алгоритм.*** Но в отличие от игры в "черный ящик" здесь школьники уже должны этот алгоритм ***как-то записать.*** Никакой фиксированной формы записи им при этом не предлагается. Форму записи в каждой задаче школьникам приходится придумывать самим. Скажем, решение задачи о "Волке, козе и капусте" они обычно записывают с помощью стрелок, используя какие-то короткие обозначения для волка, козы и капусты.

Главное, что учитель здесь должен требовать, — чтобы решение хоть как-то было ***полностью записано*** на бумаге, чтобы не было двусмысленностей, неясностей и пр.

Это и будет пропедевтикой для введения алгоритмического языка. Именно тут должна возникнуть потребность в какой-то стандартизованной форме записи алгоритмов. И уже можно сказать, что бывают разные формы записи алгоритмов. Чем сложнее задача, тем сложнее форма записи, а если мы вдобавок собираемся поручить исполнение алгоритма ЭВМ, то нужен такой язык, который поймет ЭВМ. И в нашем курсе мы для этих целей будем использовать школьный алгоритмический язык.

Задача, которую нельзя решить

Я также хочу обратить ваше внимание на упр. 12*. Это чрезвычайно сложная задача, которую, я думаю, решить нельзя. Она недаром помечена тремя звездочками. Ее решение существует, и его можно прочесть в трехтомнике Д.Кнута "Искусство программирования" (*Д.Кнут.* Искусство программирования для ЭВМ. М.: Мир, 1976. Т. 2). Задача в каком-то смысле провокационная. Придумана и вставлена по просьбе учителей, чтобы была задача, которую вообще нельзя решить (это бывает полезно в некоторых классах с особо одаренными, скажем так, "компьютерными", ребятами).

---

* Это упражнение приведено на стр. 15. — **Прим. ред.**

Мне не известен ни один человек, включая профессоров механико-математического факультета МГУ, который бы сумел додуматься до решения сам, а не прочел его в книге. Умом я, конечно, понимаю, что кто-то когда-то это решение все-таки придумал, но поверить в это не могу. Методическая значимость этой задачи в том, что задачи с очень простой формулировкой иногда имеют очень сложное содержательное решение. Вы можете рассматривать это как пропедевтику к Роботу, поскольку там тоже будут достаточно сложные задачи с очень простыми формулировками.

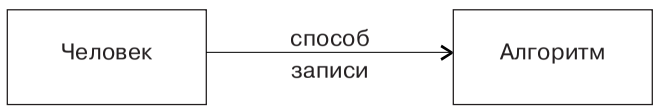
Итак, если говорить о "Введении" в целом, то в нем есть два концентра: формальность понятий "информация" и "обработка информации" (параграф 1) и понятие программы (алгоритма) для ЭВМ (параграф 3). Кроме того, затрагиваются важные с мировоззренческой точки зрения темы о роли информатики (программирования) и ЭВМ в жизни общества.

§ 4. Исполнитель Робот. Понятие алгоритма

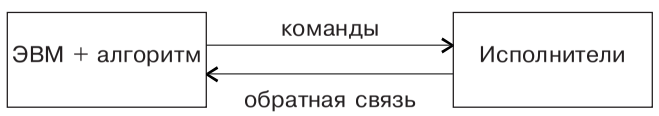
Смысл параграфа сводится к введению схемы программного управления "Человек — ЭВМ — исполнитель", а также к введению алгоритмического языка как способа записи плана будущих действий ЭВМ. Схему мы подробно обсуждали с вами, когда говорили про цели курса, про первого "кита", про развитие алгоритмического стиля мышления. Поскольку схема важная, давайте я ее снова нарисую, чтобы вы вспомнили:

СХЕМА "ЧЕЛОВЕК — ЭВМ — ИСПОЛНИТЕЛЬ"

Составление алгоритма



Выполнение алгоритма



Чтобы ввести и объяснить эту схему, нужны какие-то конкретные исполнители. Таких исполнителей в учебнике два. В § 4 вводится исполнитель Робот, а в следующем § 5 — исполнитель Чертежник.

Я не буду повторять материал, который есть в учебнике, тем более что я уже говорил про Робота, когда описывал главную цель нашего учебника — развитие алгоритмического стиля мышления учащихся.

Но я хочу подчеркнуть ряд важных моментов и объяснить, почему в учебнике сделано так, а не иначе.

Система команд Робота фиксирована, мы не можем "допридумывать" ему команды

Первое такое место — на с. 26 учебника. "Робот умеет выполнять всего 17 команд. Мы пока изучим 5..."

Здесь не случайно точно названо общее число команд Робота — 17. Опыт показывает, что это улучшает понимание "правил игры". Конечно, все понимают, что Робот — придуманное устройство. Но цифрой 17 мы с самого начала говорим: Робот уже ***есть,*** у него есть 17 команд, их можно посмотреть на первом форзаце учебника. Нам пока из 17 нужно только 5 команд. Но произвольно вводить команды Робота нельзя, ведь они заложены в конструкцию Робота, фиксированы заранее.

Этот подход полностью противоположен подходу, при котором Робот "наделяется" новыми командами по мере надобности. Опыт показывает, что фразы типа "давайте введем еще несколько команд Робота" полностью запутывают ситуацию — так объяснять новые команды нельзя. Нужно с самого начала подчеркнуть, что Робот уже есть, все его 17 команд фиксированы (можно показать их на форзаце учебника), и сказать, что ***никаких других*** команд Робот выполнять не умеет.

Другими словами, мы можем выбирать только порядок изучения команд. Но система команд Робота задана и неизменна.

Это же, кстати, относится и к изучению школьного алгоритмического языка. С самого начала, с раздела 4.1, школьный алгоритмический язык вводится как нечто уже существующее, уже заданное. Эта мысль постоянно подкрепляется замечанием, что программы на алгоритмическом языке можно выполнить на любой серийной школьной ЭВМ. (Вы можете добавить: и на любой ЭВМ, совместимой с IBM PC.) При изучении языка недопустимы фразы типа "давайте добавим в язык еще одну конструкцию", так как язык уже существует и мы можем менять только порядок изучения конструкций языка, но не сам язык.

Схема программного управления "Человек — ЭВМ — исполнитель" в § 4 только начинает вводиться. Затем формирование этой схемы продолжается в § 6 (пункт 6.6 — разделение труда между ЭВМ и исполнителями), в § 9 (9.1 — команды обратной связи; 9.2 — использование команд обратной связи при управлении Роботом вручную; 9.4 — диалог ЭВМ — Робот при исполнении цикла "пока") и далее.

В § 4 рассматривается совсем простая схема управления без обратной связи, и вводится она без особых пояснений. На первом этапе используется всего пять команд. Соответственно, можно представлять себе пульт ручного управления Роботом с пятью кнопками и 5 разных сигналов, которые ЭВМ посылает Роботу.

Авторское решение, а более — пространное обсуждение задачи с III Всесоюзной олимпиады школьников по информатике

Окончание. См. с. 3, 4

ма различным и зависит как от квалификации решающего (как в математике, так и в сфере использования ЭВМ), так и от массы других факторов.

Самые красноречивые ситуации — когда задачу возможно решить аналитически, не привлекая ЭВМ вообще.

В данной задаче, оказывается, как раз такая ситуация: ответ можно вычислить "аналитически"! Конечно, речь идет, строго говоря, не об "аналитическом вычислении ответа", а о сведении аналитическими средствами исходной задачи к существенно более простой в вычислительном плане. Считать-то, как правило, все равно приходится. В обсуждаемой задаче, если нет ошибки, приведенная в условии формула может быть средствами математики преобразована к виду

$$S(N, a, b) = \frac{1}{2ak}\,[(a-1)\,[F_1 - F_2] + N(N+1)]\,,$$

где $k = [\log_b N]$,

$$F_1 = \begin{cases} \dfrac{a^{k+1} - b^{2k+2}}{a - b^2} & \text{для } a \neq b^2, \\ k+1 & \text{для } a = b^2, \end{cases}$$

$$F_2 = \begin{cases} \dfrac{a^{k+1} - b^{k+1}}{a - b} & \text{для } a \neq b, \\ k+1 & \text{для } a = b. \end{cases}$$

Ну а эту формулу уже элементарно преобразовать к виду единственной дроби с целыми числителем и знаменателем (мы привели не такую форму из желания дать запись, более эстетичную для созерцания, на наш вкус). Так что придется лишь вычислить (без погрешностей!) эти числитель и знаменатель, после чего, как описывалось в подходе 1, запустить, скажем, алгоритм деления столбиком и печатать верные цифры результата (или что-то похитрее, о чем говорилось при обсуждении подхода 1).

Конечно, как и в некоторых предыдущих подходах, нам будет скорее всего не обойтись без реализации целой арифметики (сложения и умножения) повышенной точности.

В качестве примера приведем текст бесхитростной программы "средней интеллектуальности", соответствующей подходу 2, предназначенной для получения ответа п. **а** условия. Программа написана на языке FORTH. (Почему мы избрали FORTH? Чтобы попутно "убить еще одного зайца" — побудить кого-то из читателей ознакомиться с этим весьма интересным языком.) Для получения ответа по п. **а** необходимо ввести

```
100 3 2 ?
```

И в программе не обошлось без морали:

не всегда надо стремиться к всеобщности, делая лишнее. Для решения более или менее конкретной задачи может потребоваться индивидуальная "ручная подгонка" сложности программы под эту задачу.

Вот так и здесь: от нас требуется по условию не написать программу, которая вычисляла бы $S(N, a, b)$ при любых N, a, b, а вычислить $S(N, a, b)$ при трех конкретных наборах N, a, b. Приводимая программа и рассчитана на вычисление $S(100, 3, 2)$: при вычислении в ней проверяется, не возникает ли переполнение целой арифметики одинарной точности (16-битовой). Оказывается, для $S(100, 3, 2)$ не возникает. (Прикидки показывали, что промежуточные результаты должны поместиться скорее всего в двухбайтные слова, хотя полной уверенности не было.) Но если бы при вычислении $S(100, 3, 2)$ выдалась диагностика "Переполнение", пришлось бы реализовать другую программу, где работа шла бы с числами двойной (или более) длины.

Хотя, с другой стороны, всеобщность (в данном случае, скажем, автоматическая настройка "ячеек памяти" и операций на текущий размер данных) всегда приятна. И нередко позволяет в конечном счете экономить время.

```
QUAN A QUAN B QUAN N QUAN B**X QUAN A**
(описание целых переменных)

: СУМРАЦ (A B C D -> E F - A/B + C/D = E/F при D=k_B)

ROT 2DUP = IF DROP -ROT + SWAP ELSE
OVER > R / ROT * + DUP 0< ABORT" Переполнение"
R > ENDIF;

: РАЦРЕЗ ( N -> SUM I=1,...,N )

0 1 ROT 1+ 1
?DO B**X I > NOT B**X B * I > AND
IFNOT B**X B * TO B**X A** A * TO A** ENDIF
I A** СУМРАЦ LOOP;

: ДЕЛСТОЛБ ( M N -> {ВЫВОД M/N} )
2DUP SWAP . ." / " . ." = "
SWAP OVER /MOD . C" . EMIT 200 0
DO 10 * OVER /MOD C" 0 + EMIT BREAK LOOP
." ..." 2DROP;

: ? TO N TO A TO B
1 TO B**X 1 TO A**N РАЦРЕЗ ДЕЛСТОЛБ;
```

Как быть в случаях "нецелых" a и b?

Ответим коротко, и в этом ответе — одна из важных моралей, позволяющая иначе посмотреть на многие известные вещи. Какие "нецелые" числа бывают в программировании? Чаще всего — так называемые "вещественные" (*real*). Но на самом деле это не те "вещественные", которые рассматриваются в математике. Это просто "рекламное название" того, что строже называют "числами с плавающей запятой" (впрочем, иногда сгодятся и нецелые "числа с фиксированной запятой"). С точки зрения математики то, что в программировании стали называть "вещественными числами", — это пара целых чисел — мантиссы и порядка (десятичного, двоичного, восьмеричного или шестнадцатеричного — по-разному бывает). А еще можно трактовать их как рациональную дробь: скажем, 123.456 — это 123456/1000. А с такими числами работать без вычислительных погрешностей мы уже вроде бы научились.

Так, при "нецелом" b, которое нам удобно представить в виде $b=b_1/b_2$, где b_1 и b_2 — целые, мы для точного вычисления функции $U(b,i)=[\log_b i]$ можем воспользоваться неравенством, полученным из приводившегося выше (*) в связи с целым b: точное значение функции $[\log_b i]$ — это целое U, удовлетворяющее двойному неравенству

$$b_2^{U+1} \cdot i < b_1^{U+1} \le b_1 \cdot b_2^U \cdot i$$

(причем это неравенство определяет одно и только одно U). Дальше — вопрос техники! Так что пункт **б**, где $a=3$, а $b=19/10$ (т.е. $b_1=19$, $b_2=10$), мы, можно считать, решили.

А как пункт **в**, где не только b, но и a "вещественное"? Почти так же. Представим только **а** в виде $a=a_1/a_2$, где a_1 и a_2 — целые, и все станет ясно. Ведь нашу исходную формулу тогда можно будет представить как

$$S(N, a, b) = \sum_{i=1}^{N} \frac{i \cdot a_2^{[\log_b i]}}{a_1^{[\log_b i]}}\,.$$

А работа с этой формулой идейно ничем не отличается от того, что мы обсуждали, решая задачу п. **а**. Да, в п. **в** мы будем брать $a_1=31$, $a_2=10$.

Тексты программ, решающих задачу для пп. **б** и **в**, я не привожу, чтобы не омрачать вам удовольствие от их самостоятельного решения.

Послесловие

Кое-кто, прочитав наши решения и их морали, говорил автору: "Зачем было все это писать? Тут нет ни единого нового слова, это все прекрасно знают, и все это — первое, что приходит в голову любому". В том, что это не вполне так, убеждают результаты решения этой задачи на Всесоюзной олимпиаде. Там большая часть участников пошла по линии "бесхитростного" решения (у большинства, давшего 10—12 верных знаков и несколько неверных; причем, какие же знаки верные, ни один участник не смог указать, к тому же в условии требуется печатать лишь верные знаки...). А то, что по прочтении готового решения очень часто задача кажется тривиальной, факт известный.

Еще одно подтверждение полезности наших моралей дало обсуждение одной из задач, где речь шла о том, СУЩЕСТВУЕТ ли для набора отрезков на плоскости, заданных координатами концов, прямая, проходящая через все эти отрезки. Многие настаивали, что в условии должно оговариваться, что координаты концов отрезков — непременно целые числа (знакомая ситуация?). Обоснование — "иначе решение будет некорректным: если искомая прямая проходит весьма близко от конца одного из отрезков, то вследствие вычислительных ошибок мы в принципе можем, скажем, получить ответ, что прямая пересекает этот отрезок, в то время как она его не пересекает, и наоборот".

Многие уверены, что и целочисленность координат нас не спасет в этой задаче: ведь точка пересечения отрезка, концы которого имеют целочисленные координаты, с произвольной прямой (пусть даже и проходящей через пару "целочисленных" точек — на самом деле потребуется лишь этот случай) не обязательно будет "целочисленной", а стало быть, результат будет приближенным! И даже если точка пересечения отрезка и прямой "математически" будет целочисленной, это все равно ничего не гарантирует: при вычислении координат этой точки нам придется применять операцию "вещественного" деления, так что все равно на машине результат будет приближенным... Есть у многих такая уверенность...

А что если воспользоваться нашими моралями?

Тогда мы скажем: зачем же действовать столь бесхитростно? Ведь для выяснения того, пересекаются ли два отрезка с "целочисленными" концами, вовсе не обязательно в программе вычислять координаты точки пересечения (мораль "не грех сначала подумать"). Далее, в условии существования этого пересечения вовсе не обязательно использовать операцию деления, ведь достаточно в этой формуле привести все к общему знаменателю — и деление исчезнет (морали "удачная расстановка скобок" и "подумать, пытаясь найти аналитическое решение")! Так что определить, пересекаются ли два отрезка с "целочисленными концами", можно ТОЧНО, используя лишь операции целочисленных сложения, умножения и сравнения целых чисел!

А если наши данные еще и "вещественные", то, как следует из одной из наших моралей, должен быть решен вопрос, в какой форме мы эти данные планируем сообщить. Как правило (исключая "аналитические" формы), окажется, что на самом деле эти данные — всего лишь рациональные десятичные дроби. Поскольку рассматриваемая задача инвариантна ("безразлична") к умножению координат всех точек на любое одно и то же число, можно все исходные данные умножить на 1 с достаточным числом нулей — они станут целыми. Таким простейшим способом мы можем свести задачу с "вещественными" исходными данными к "целочисленной". Хотя можно действовать и менее коротким путем, но это в общем-то не столь уж важно.

Так что наши морали все же иногда небесполезны.

Решение логических задач средствами алгебры логики

Л.З. ШАУЦУКОВА,
г. Нальчик

Логические задачи очень разнообразны. Способов их решения тоже немало. Но наибольшее распространение получили следующие три способа решения логических задач:

- средствами алгебры логики;
- табличный;
- с помощью рассуждений.

Познакомимся с первым способом.

Обычно используется следующая схема решения:

1. Изучается условие задачи.
2. Вводится система обозначений для логических высказываний.
3. Конструируется логическая формула, описывающая логические связи между всеми высказываниями, выделенными из условия задачи.
4. Определяются значения истинности этой логической формулы.
5. Из полученных значений истинности формулы определяются значения истинности введенных логических высказываний, на основании которых делается заключение о решении.

Приведем несколько примеров решения логических задач средствами алгебры логики.

Пример 1

Внимание Андрея, Дениса и Марата привлек промчавшийся мимо них автомобиль.

— Это английская машина марки "Феррари", — сказал Андрей.

— Нет, машина итальянская, марки "Понтиак", — возразил Денис.

— Это "Сааб", и сделан он не в Англии, — сказал Марат.

Оказавшийся рядом знаток автомобилей сказал, что каждый из них прав только в одном из двух высказанных предположений.

Какой же марки этот автомобиль и в какой стране изготовлен?

Решение

Введем обозначения для логических высказываний:
А — машина английская;
И — машина итальянская;
П — это "Понтиак";
С — это "Сааб";
Ф — это "Феррари".

Из того факта, что каждый из друзей прав только в чем-то одном, получаем три истинных составных высказывания:

$$А \cdot \overline{Ф} \vee \overline{А} \cdot Ф;$$
$$И \cdot \overline{П} \vee \overline{И} \cdot П;$$
$$\overline{А} \cdot \overline{С} \vee А \cdot С.$$

Если все эти истинные высказывания логически перемножить, то получим следующее истинное логическое высказывание:

$$(А \cdot \overline{Ф} \vee \overline{А} \cdot Ф) \cdot (И \cdot \overline{П} \vee \overline{И} \cdot П) \cdot (\overline{А} \cdot \overline{С} \vee А \cdot С).$$

Для решения задачи нужно определить, при каких значениях логических переменных А, И, Ф, П и С это высказывание истинно.

Упростим высказывание, учитывая те обстоятельства, что машина не может быть одновременно и английской, и итальянской ($А \cdot И = 0$), а также не может одновременно иметь два разных названия ($Ф \cdot С = 0; Ф \cdot П = 0; П \cdot С = 0$):

$$(А \cdot \overline{Ф} \vee \overline{А} \cdot Ф) \cdot (И \cdot \overline{П} \vee \overline{И} \cdot П) \cdot$$
$$\cdot (\overline{А} \cdot \overline{С} \vee А \cdot С) = А \cdot \overline{Ф} \cdot И \cdot \overline{П} \cdot \overline{А} \cdot \overline{С} \vee$$
$$\vee А \cdot \overline{Ф} \cdot И \cdot \overline{П} \cdot А \cdot С \vee А \cdot \overline{Ф} \cdot \overline{И} \cdot П \cdot \overline{А} \cdot \overline{С} \vee$$
$$\vee А \cdot \overline{Ф} \cdot \overline{И} \cdot П \cdot А \cdot С \vee \overline{А} \cdot Ф \cdot И \cdot \overline{П} \cdot \overline{А} \cdot \overline{С} \vee$$
$$\vee \overline{А} \cdot Ф \cdot И \cdot \overline{П} \cdot А \cdot С \vee \overline{А} \cdot Ф \cdot \overline{И} \cdot П \cdot \overline{А} \cdot \overline{С} \vee$$
$$\vee \overline{А} \cdot Ф \cdot \overline{И} \cdot П \cdot А \cdot С = 0 \vee 0 \vee 0 \vee 0 \vee$$
$$\vee \overline{А} \cdot Ф \cdot И \cdot \overline{П} \cdot \overline{А} \cdot \overline{С} \vee 0 \vee 0 \vee 0 =$$
$$= \overline{А} \cdot Ф \cdot И \cdot \overline{П} \cdot \overline{С}.$$

Высказывание $\overline{А} \cdot Ф \cdot И \cdot \overline{П} \cdot \overline{С}$ истинно только при $И = 1, Ф = 1, А = 0, П = 0, С = 0$.

Ответ. Машина итальянская, марки "Феррари".

Пример 2

В клуб служебного собаководства на очередную тренировку пришли со своими собаками Антон, Борис, Петр, Виктор и Олег. Желая подшутить над новым инструктором, на его вопрос "Кто же хозяин каждой из собак?" каждый юноша дал один правильный и один неправильный ответ.

Антон сказал: *"Моя собака — Рекс, а собака Петра — Лайма"*. Борис сказал: *"Рекс — моя собака, а собака Виктора — Джек"*. Петр сказал: *"Собака Виктора — Зевс, а моя — Рекс"*. Виктор сказал: *"Моя собака — Джек, а собака Олега — Бичо"*. Олег сказал: *"Да, моя собака — Бичо, а собака Бориса — Зевс"*.

Кто же на самом деле хозяин каждой из собак?

Решение

Обозначим высказывательную форму *"юноша X — хозяин собаки Y"* как X_Y. Из высказываний молодых людей и того факта, что одно из высказываний истинно, а другое ложно, следуют истинные составные высказывания:

$$А_Р \cdot \overline{П_Л} \vee \overline{А_Р} \cdot П_Л; \qquad Б_Р \cdot \overline{В_Д} \vee \overline{Б_Р} \cdot В_Д;$$
$$В_З \cdot \overline{П_Р} \vee \overline{В_З} \cdot П_Р; \qquad В_Д \cdot \overline{О_Б} \vee \overline{В_Д} \cdot О_Б;$$
$$О_Б \cdot \overline{Б_З} \vee \overline{О_Б} \cdot Б_З.$$

Если все эти истинные высказывания логически перемножить, то получим следующее истинное высказывание:

$$(А_Р \cdot \overline{П_Л} \vee \overline{А_Р} \cdot П_Л) \cdot (Б_Р \cdot \overline{В_Д} \vee \overline{Б_Р} \cdot В_Д) \cdot$$
$$\cdot (В_З \cdot \overline{П_Р} \vee \overline{В_З} \cdot П_Р) \cdot (В_Д \cdot \overline{О_Б} \vee \overline{В_Д} \cdot О_Б) \cdot$$
$$\cdot (О_Б \cdot \overline{Б_З} \vee \overline{О_Б} \cdot Б_З)$$

Выполнив преобразование этого высказывания с учетом того, что у каждого хозяина только одна собака и у каждой собаки только один хозяин, получим следующее равносильное высказывание:

$$\overline{А_Р} \cdot П_Л \cdot Б_Р \cdot \overline{В_Д} \cdot В_З \cdot \overline{П_Р} \cdot \overline{В_Д} \cdot О_Б \cdot \overline{Б_З},$$

которое истинно только при $А_Р = 0, П_Л = 1, Б_Р = 1, В_Д = 0, В_З = 1, П_Р = 0, О_Б = 1, Б_З = 0$.

Ответ

Петр — хозяин Лаймы, Борис — Рекса, Виктор — Зевса, Олег — Бичо, Антон — Джека.

Пример 3

Трое друзей, болельщиков автогонок "Формула-1", спорили о результатах предстоящего этапа гонок.

— Вот увидишь, Шумахер не придет первым, — сказал Джон. — Первым будет Хилл.

— Да нет же, победителем будет, как всегда, Шумахер, — воскликнул Ник. — А об Алези и говорить нечего, ему не быть первым.

Питер, к которому обратился Ник, возмутился:

— Хиллу не видать первого места, а вот Алези пилотирует самую мощную машину.

По завершении этапа гонок оказалось, что каждое из двух предположений двоих друзей подтвердилось, а оба предположения третьего оказались неверными. Кто выиграл этап гонки?

Решение

Введем обозначения для логических высказываний:
Ш — победит Шумахер; Х — победит Хилл; А — победит Алези.

Реплика Ника "Алези пилотирует самую мощную машину" не содержит никакого утверждения о месте, которое займет этот гонщик, поэтому в дальнейших рассуждениях не учитывается.

Зафиксируем высказывания каждого из друзей:

Джон: $\overline{Ш} \cdot Х$; Ник: $Ш \cdot \overline{А}$; Питер: $\overline{Х}$.

Учитывая то, что предположения двух друзей подтвердились, а предположения третьего неверны, запишем истинное высказывание:

$$(\overline{Ш} \cdot Х) \cdot (Ш \cdot \overline{А}) \cdot \overline{\overline{Х}} \vee (\overline{Ш} \cdot Х) \cdot \left(\overline{Ш \cdot \overline{А}}\right) \cdot Х \vee$$
$$\vee \left(\overline{\overline{Ш} \cdot Х}\right) \cdot (Ш \cdot \overline{А}) \cdot \overline{Х} =$$
$$= (Ш \vee \overline{Х}) \cdot Ш \cdot \overline{А} \cdot \overline{Х} = Ш \cdot \overline{А} \cdot \overline{Х}.$$

Высказывание $Ш \cdot \overline{А} \cdot \overline{Х}$ истинно только при $Ш = 1, А = 0, Х = 0$.

Ответ

Победителем этапа гонок стал Шумахер.

Пример 4

Некий любитель приключений отправился в кругосветное путешествие на яхте, оснащенной бортовым компьютером. Его предупредили, что чаще всего выходят из строя три узла компьютера — *a, b, c* — и дали необходимые детали для замены. Выяснить, какой именно узел надо заменить, он может по сигнальным лампочкам на контрольной панели. Лампочек тоже ровно три: *x*, *y* и *z*.

Инструкция по выявлению неисправных узлов такова:

1) если неисправен хотя бы один из узлов компьютера, то горит по крайней мере одна из лампочек *x, y, z*;
2) если неисправен узел *a*, но исправен узел *c*, то загорается лампочка *y*;
3) если неисправен узел *c*, но исправен узел *b*, загорается лампочка *y*, но не загорается лампочка *x*;
4) если неисправен узел *b*, но исправен узел *c*, то загораются лампочки *x* и *y* или не загорается лампочка *x*;
5) если горит лампочка *x* и при этом либо неисправен узел *a*, либо все три узла — *a*, *b*, *c* — исправны, то горит и лампочка *y*.

В пути компьютер сломался. На контрольной панели загорелась лампочка *x*. Тщательно изучив инструкцию, путешественник починил компьютер. Но с этого момента и до конца плавания его не оставляла тревога. Он понял, что инструкция несовершенна и есть случаи, когда она ему не поможет.

Какие узлы заменил путешественник? Какие изъяны он обнаружил в инструкции?

Решение

Введем обозначения для логических высказываний:
a — неисправен узел *a*; *x* — горит лампочка *x*;
b — неисправен узел *b*; *y* — горит лампочка *y*;
c — неисправен узел *c*; *z* — горит лампочка *z*.

Правила 1—5 выражаются следующими формулами:

$$a \vee b \vee c \to x \vee y \vee z, \qquad (1)$$
$$a \cdot \overline{c} \to y, \qquad (2)$$
$$c \cdot \overline{b} \to y \cdot \overline{x}, \qquad (3)$$
$$b \cdot \overline{c} \to (x \cdot y \vee \overline{x}), \qquad (4)$$
$$(a \vee \overline{a} \cdot \overline{b} \cdot \overline{c}) \cdot x \to y. \qquad (5)$$

Формулы 1—5 истинны по условию, следовательно, их конъюнкция тоже истинна:

$$(a \vee b \vee c \to x \vee y \vee z) \cdot (a \cdot \overline{c} \to y) \cdot$$
$$\cdot (c \cdot \overline{b} \to y \cdot \overline{x}) \cdot (b \cdot \overline{c} \to x \cdot y \vee \overline{x}) \cdot$$
$$\cdot ((a \vee \overline{a} \cdot \overline{b} \cdot \overline{c}) \cdot x \to y) = 1$$

Выражая импликацию через дизъюнкцию и отрицание (напомним, что $(a \to b) = (\overline{a} \vee b)$, получаем:

$$(\overline{a} \cdot \overline{b} \cdot \overline{c} \vee x \vee y \vee z) \cdot (\overline{a} \vee c \vee y) \cdot$$
$$\cdot (\overline{c} \vee b \vee y \cdot \overline{x}) \cdot (\overline{b} \vee c \vee x \cdot y \vee \overline{x}) \cdot$$
$$\cdot \left(\overline{(a \vee \overline{a}) \cdot (a \vee \overline{b}) \cdot (a \vee \overline{c}) \cdot x} \vee y\right) =$$
$$= (\overline{a} \cdot \overline{b} \cdot \overline{c} \vee x \vee y \vee z) \cdot (\overline{a} \vee c \vee y) \cdot$$
$$\cdot (\overline{c} \vee b \vee y \cdot \overline{x}) (\overline{b} \vee c \vee x \cdot y \vee \overline{x}) \cdot$$
$$\cdot (\overline{a} \cdot b \vee \overline{a} \cdot c \vee \overline{x} \vee y) = 1.$$

Подставляя в это тождество конкретные значения истинности $x = 1, y = 0, z = 0$, получаем:

$$(\overline{a} \vee c) \cdot (\overline{c} \vee b) \cdot (\overline{b} \vee c) \cdot (\overline{a} \cdot b \vee \overline{a} \cdot c) =$$
$$= (\overline{a} \cdot \overline{c} \vee \overline{a} \cdot b \vee c \cdot b) \cdot (\overline{b} \cdot \overline{a} \cdot c \vee c \cdot \overline{a} \cdot b \vee$$
$$\vee \overline{a} \cdot c) = \overline{a} \cdot b \cdot c = 1.$$

Отсюда следует, что $a = 0, b = 1, c = 1$.

Ответ на первый вопрос задачи. Нужно заменить блоки *b* и *c*; блок *a* не требует замены. *Ответ на второй вопрос задачи получите самостоятельно.*

С любезного разрешения автора печатается по книге: "Информатика: Учебник для 7—11-х классов общеобразоват. учебн. заведений". В 2 кн. Нальчик: Эль-фа, 1997.

МАТЕРИАЛЫ К УРОКУ

Интернет для начинающих

Продолжение. См. с. 1, 2

Правила записи текста

Вася. Твоему опыту я, конечно, верю — статьи и заметки моего папы всегда приятно читать: они написаны очень понятно. Думаю, что со временем и я научусь нажимать на клавиши так, чтобы на экране получался красивый, содержательный текст.

Папа. Спасибо! Мой сын учится говорить комплименты! Раз ты упомянул о красоте текста на экране, то поговорим о длине строки, форматировании и знаках препинания.

Строчки письма не должны быть слишком длинными. Они должны помещаться на экране целиком, иначе читать их будет неудобно. Хороший размер для экранной строки — 64 символа или чуть больше, но обязательно меньше 80 знаков.

Вася. Это понятно: если при чтении письма приходится постоянно пользоваться горизонтальной протяжкой, то к концу текста устанут глаза и руки. Однажды я намаялся, читая "хвостатый" файл Маши.

Папа. Для того чтобы рисунок письма был приятен глазу, его текст нужно выравнивать внутри каждого абзаца по левому и правому краям, то есть форматировать. Понятно, что форматирование желательно выполнять не вручную, а средствами того редактора, в котором пишется текст сообщения.

Вася. Форматированный текст смотрится как текст в книжке — очень красиво. Позволяют ли редакторы наших почтовых программ выполнять эту операцию?

Папа. Такие средства в них есть, и я предлагаю тебе почитать инструкции и встроенные подсказки.

Вася. Задание понял!

Папа. Теперь о знаках препинания. При наборе текста на компьютере к ним нужно относиться с повышенным вниманием.

Вася. Запятые для меня всегда были проблемой. Обычно я их ставлю там, где хочу указать паузу для чтения, но это не всегда совпадает с правилами русского языка.

Папа. Я хочу поговорить не о грамматике, а о том, что часто знаки препинания отделяют от слова пробелом, а это неверно, некрасиво и может привести к ошибкам автоматического форматирования.

Вася. Каким образом?

Папа. Если, например, запятая выделена пробелами, то она воспринимается текстовым редактором как отдельное слово и может при форматировании попасть в начало строки. Как смотрится строчка, начинающаяся с запятой?

Вася. Согласен — не очень привлекательно. Как же правильно записывать знаки препинания в компьютерном тексте?

Папа. Правила просты: нужно запомнить, что знак препинания относится к слову, за которым он идет, поэтому он записывается со "своим" словом слитно, без пробелов. А вот после знака препинания пробел ставится обязательно. Исключение только одно — открывающая скобка. Она записывается слитно со следующим за ней словом, а перед ней располагается пробел.

Ошибки в тексте

Вася. Понять, что запятые нужно писать слитно со словами, за которыми они следуют, очень легко. А вот как научиться ставить их в правильном месте и не делать в тексте других грамматических ошибок?

Папа. Ты, конечно, сам знаешь ответ на этот вопрос: нужно лучше относиться в школе к урокам русского языка. Однако компьютер и здесь может оказаться хорошим помощником. Существуют специальные программы для проверки правильности написания слов. Их следует использовать перед посылкой писем. На нашем компьютере установлена такая программа-корректор RUSP, она работает в DOS. В почтовой программе Windows есть свой встроенный корректор, он запускается автоматически при отправлении каждого письма.

ДЛЯ УЧИТЕЛЯ
(Комментарии для серьезных читателей)

В этом разделе Куки обсуждают формальные правила написания текста письма. Рассматриваются также технические вопросы, связанные с автоматической подписью и проверкой письма специальными программами-корректорами перед отправкой.

Основное практическое задание раздела: организовать и провести учебную переписку участников курса друг с другом.

Проблема, стоящая перед куратором курсов и руководителями детских команд, не столько техническая и организационная, сколько содержательная. О чем писать? И как?

Если не задать заранее тему, переписка может не получиться или принесет разочарование. В качестве основной темы детских писем предлагается придумывание вопросов, связанных с Интернет, и ответы на них. Схема проведения учебного электронного раунда описана в упражнении 17. Согласно этой схеме каждый участник переписки напишет одно письмо и получит на него ответ; к каждому участнику придет одно письмо, на которое он должен послать ответ. При желании корреспондентов и благоприятных сетевых возможностях команд переписка может выйти за рамки упражнения 17 и продолжиться далее сама по себе. В любом случае нужно сохранить электронные адреса партнеров учебной переписки для выполнения сетевых упражнений в следующих разделах.

Вопросы и упражнения

1. Внимательно прочитайте письма Паучка и составьте шаблон, который он использует для "сочинения" своих сказок. В шаблоне используйте квадратные скобки для пометок слов, взятых Пауком из письма-программы, и фигурные — для слов, вставляемых в сказку случайным образом из заранее подготовленных списков.

Ответ:

```
Дорогой [Вася]!
Специально для тебя я написал такую сказку.

Однажды
[стремительный] [Змей-Горыныч]
{захромал} вместе с {Мудрым Кроликом}
{ждать вчерашнего дня.}
Вот добрались они до места.
Смотрят, а там совсем
[дурной] [Соловей-Разбойник.]
{- Бедный Йорик! -}
так подумали
[Змей-Горыныч] и {Мудрый Кролик.}
Они стали {лаять}
очень сильно. И пропал недруг!
Занялись друзья делом,
а сказке конец.
Привет!!!
\#/
=O= Паучок (почтовый робот Роботландии)
/ \
```

2. Пол корреспондента и род имени положительного героя не вставляются в шаблон сказки. Как используется эта информация? Почему в задании на сказку Паучок не просит сообщить род имени отрицательного героя?

Ответ. Пол корреспондента используется для выбора правильного обращения в ответе (дорогой, дорогая). Род имени положительного героя нужен для правильного выбора глагола (захромал, захромала, захромало). Род имени отрицательного героя нигде не используется, кроме определения (захромал), но это определение берется из письма-задания.

3. Нужно ли указывать в тексте письма свой электронный адрес?

Ответ. В этом нет необходимости: при посылке ответа адрес корреспондента можно извлекать автоматически из заголовка (конверта) электронного письма.

4. Каким образом можно извлечь адрес корреспондента из полученного письма при посылке ответа?

Ответ. Это можно сделать двумя способами.

1. При чтении письма в почтовой программе выбрать в меню команду **Ответить автору** — адрес корреспондента запишется на конверт отправления автоматически.
2. Сначала занести адрес в адресную книгу, а потом извлечь его оттуда при подготовке ответа.

5. Каково назначение автоматической подписи электронного письма?

Ответ. Электронная подпись ускоряет подготовку корреспонденции. Выигрыш особенно ощутим в случае, когда подпись большая и содержит сложные данные об авторе письма. Пример:

> Иван Петрович Сидоров, кандидат физ.-мат. наук, главный конструктор предприятия "Электронные примусы", г. Цигель; тел. (8657) 2-48-50.

6. Придумайте для себя автоматическую подпись и настройте почтовую программу на ее использование. Для проверки подписи пошлите письмо произвольного содержания самому себе.

Замечание. В программе BML подпись можно вставлять в письмо из заранее заготовленного файла при помощи строки **Вставить файл** в позиции **Вставить** главного меню программы.

7. Объясните назначение цитат в письмах.

Ответ. Цитаты напоминают корреспонденту, на какие фрагменты его письма получен ответ.

8. Как принято оформлять цитаты при электронной переписке?

Ответ. Письма цитируются строчками, и в начало каждой строки добавляется символ ">".

9. Дойдет ли письмо до корреспондента, если цитаты в нем оформлены неверно (например, вместо знака ">" использован знак "<")?

Ответ. Содержимое письма никак не влияет на его путешествие по Интернету. Но лучше использовать принятые обозначения и правила оформления для того, чтобы получатель не терялся в догадках.

10. Можно ли цитировать письма вручную, набирая заново текст фрагмента полученного письма?

Ответ. Конечно, можно, но так делают крайне редко. Гораздо удобнее воспользоваться автоматической вставкой письма корреспондента в свой ответ как цитаты, с последующим удалением ненужных фрагментов. Все современные почтовые программы позволяют сделать это.

11. Для чего в письмах используются смайлики?

Ответ. Смайлики используются для условного изображения эмоций автора письма. Они оживляют общение, приближая его к обычному разговору.

12. Как Васе написать обращение в своем письме к

а) маме;
б) школьному товарищу;
в) далекому другу по переписке;
г) редактору детского журнала;
д) ребятам из другой команды роботландского университета?

Окончание на с. 16

Интернет для начинающих

Окончание. См. с. 1, 2, 15

Возможные ответы:

а) “Милая мама!”;

б) “Коля, привет!”;

в) “Миша, добрый день!”;

г) “Уважаемый редактор!”;

д) “Дорогие друзья!”.

13. В чем преимущество составления предварительного плана письма с последующей детализацией его пунктов?

Ответ. Составляя план, мы думаем над тем, что хотим сообщить своему корреспонденту. Меньше вероятность забыть написать о чем-то важном. Писать письмо по готовому плану гораздо легче: определены темы фрагментов сообщения, остается только подобрать нужные слова и связать их в простые и понятные предложения.

14. Почему текст письма не должен содержать строк длиннее 80 символов?

Ответ. Для большинства мониторов это максимальное число знаков, умещающихся в одной строке. Если строка длиннее, то ее приходится “прокручивать” по горизонтали, а это неудобно.

15. Найдите в инструкциях своих почтовых программ алгоритмы выполнения форматирования абзацев.

Замечание. Встроенный редактор программы DMAIL имеет средства форматирования абзацев (Alt R, Alt J, Ctrl J), но он не умеет переносить слова по слогам. Если сделать правый край абзаца ровным, между словами может оказаться много пробелов, а это некрасиво.

Но программа DMAIL имеет замечательную возможность — указывать в настройках имя внешнего редактора, которому передается управление на время написания текста письма. В качестве такого редактора подходит Микромир — форматирование абзацев в нем выполняется идеально.

Письмо можно подготовить в Микромире (или другом редакторе) заранее, а потом просто вставить его в редактор почтовой программы. Например, в программе DMAIL вставить текст из файла в письмо можно при помощи аккорда Shift F5 или через строку **Вставить файл** в позиции **Редактирование** меню редактора.

В программе BML для того, чтобы отформатировать абзац, нужно клавишами F7 и клавишами ↓ и ↑ выделить нужную группу строк и выбрать в меню команду **Формат** (или нажать аккорд Alt J).

16. Изучите правила работы с вашей программой-корректором. Используйте эту программу для проверки писем перед их отправкой по электронной почте.

17. Учебная переписка. Это упражнение предполагает обмен электронными письмами детей непосредственно друг с другом. Переписка носит учебный характер, поэтому контролируется и комментируется руководителями команд и куратором курса.

Организация переписки

1. Куратор курса проводит предварительное анкетирование участников переписки и на основе полученных анкет создает базу данных со следующими записями:

```
Имя корреспондента
Электронный адрес
Возраст
Город
Номер команды
Руководитель команды
Область интересов
```

2. Из полученной базы данных формируются списки абонентов, которые направляются командам.

3. Внутри каждой команды происходит распределение полученных адресов исходя из предпочтений детей, но таким образом, чтобы все полученные адреса оказывались задействованными.

4. Учебная переписка включает в себя следующие этапы:

— ученик пишет письмо своему корреспонденту;

— ученик посылает ответ на полученное сообщение.

Руководитель команды контролирует каждое отправляемое письмо и пишет обзор-рецензию на пришедшие письма. Рецензии отправляются куратору курса. Куратор курса на основе полученных рецензий выявляет типичные ошибки электронной переписки и отмечает отдельные наиболее удачные письма детей.

Приложение

ФОРМАТ УЧЕБНОГО ПИСЬМА

```
Тема:
Содержание:
Обращение
Представление автором самого себя
Вопросы по Интернет
Подпись
```

Пример письма

```
Тема: Для Ивановой Светланы
Дорогая Света!
Пишет тебе Петров Игорь из города Благовещенска. Наш город находится на Дальнем Востоке и расположен на границе с Китаем. Город очень красивый и солнечный. Учусь я в 7-м классе средней школы № 5. От моей школы до реки Амур, по которой проходит граница, всего 5 минут неспешной прогулки. Я занимаюсь в клубе мотокроссом. По нашим амурским сопкам езда на спортивном мотоцикле — одно удовольствие. В следующем году мечтаю отправиться с друзьями в мотопробег в город Тынду. Маршрут будет проложен прямо по тайге, и нас, я думаю, ждет интересное приключение.
  Еще я увлекаюсь информатикой и хочу научиться свободно "плавать" по Интернету.
  Специально для тебя я придумал такие вопросы:
  1. Кто изобрел Интернет и в какой стране?
  2. Какие компьютеры могут подключаться к этой суперсети?
  3. Можно ли послать электронное письмо самому себе?
  4. Можно ли послать одно письмо сразу нескольким адресатам?
  С нетерпением жду ответа, Петров Игорь
```

ФОРМАТ УЧЕБНОГО ОТВЕТА

```
Тема:
Содержание:
Обращение
Комментарии авторского представления
Ответы на вопросы по Интернету
Подпись
```

Пример ответа

```
Игорь, добрый день!
  Очень рада твоему письму. С интересом прочитала твой рассказ. Я никогда не была на Дальнем Востоке и никогда не ездила на мотоцикле. Все то, о чем ты написал, мне интересно.
  > В следующем году мечтаю отправиться с
  > друзьями в мотопробег в город Тынду.
  Желаю тебе удачи!
  Я живу в городе Луховицы Московской области и учусь в 6-м классе гимназии № 1. Мои увлечения: современная отечественная музыка (любимый исполнитель - Борис Гребенщиков) и компьютеры.
  Отвечаю на твои вопросы.
  > 1. Кто изобрел Интернет и в какой стране?
  В 60-е годы американская фирма Rand Corporation впервые предложила создать компьютерную сеть, покрывающую всю страну. В первом варианте предложение увидело свет в 1964 году благодаря усилиям сотрудника Rand Corporation Пола Бэрана. Эти сведения я прочитала в толстой книге Д.О'Доннела, Э.Ледда "Microsoft Internet Explorer 3" (издательство BHV, Санкт-Петербург, 1997).
  > 2. Какие компьютеры могут подключаться к этой суперсети?
  К Интернет могут подключаться любые компьютеры самых разных фирм (компьютеры фирм IBM, Apple и других). Необходимым условием подключения компьютера в сеть является наличие для него сетевого программного обеспечения. Если такое обеспечение для компьютеров данного типа существует, их можно подключать к Интернету, если нет, такое программное обеспечение можно написать. Иными словами, нет принципиальных ограничений.
  > 3. Можно ли послать электронное письмо самому себе?
  Конечно! Мы делали это у себя в классе, когда проверяли автоматическую подпись электронного письма.
  > 4. Можно ли послать одно письмо сразу нескольким адресатам?
  Так можно сделать, если написать в адресном поле не один, а несколько адресов. Кроме того, можно наряду с адресным полем Кому использовать дополнительные адресные поля Копия и Слепая.
  С дружеским приветом, Иванова Светлана
```

ФОРМАТ РЕЦЕНЗИИ РУКОВОДИТЕЛЯ

```
Команда: <номер команды>
Руководитель: <имя, отчество, фамилия>
Краткое описание организации учебной переписки в команде.
Общее впечатление руководителя о прошедшей учебной
переписке с замечаниями, рекомендациями и предложениями.
Список вопросов команды
Вопрос (с указанием автора)
Ответ корреспондента (с указанием имени и номера команды)
Комментарий руководителя...
Несколько примеров удачных писем.
```

Продолжение следует

Отпечатано с готовых диапозитивов редакции в типографии “ПРЕССА”, 125865, Москва, ул. Правды, 24

Тираж 7000 экз.
Заказ №