№ 17 (210) май 1999

еженедельное
ПРИЛОЖЕНИЕ К ГАЗЕТЕ
«ПЕРВОЕ СЕНТЯБРЯ»

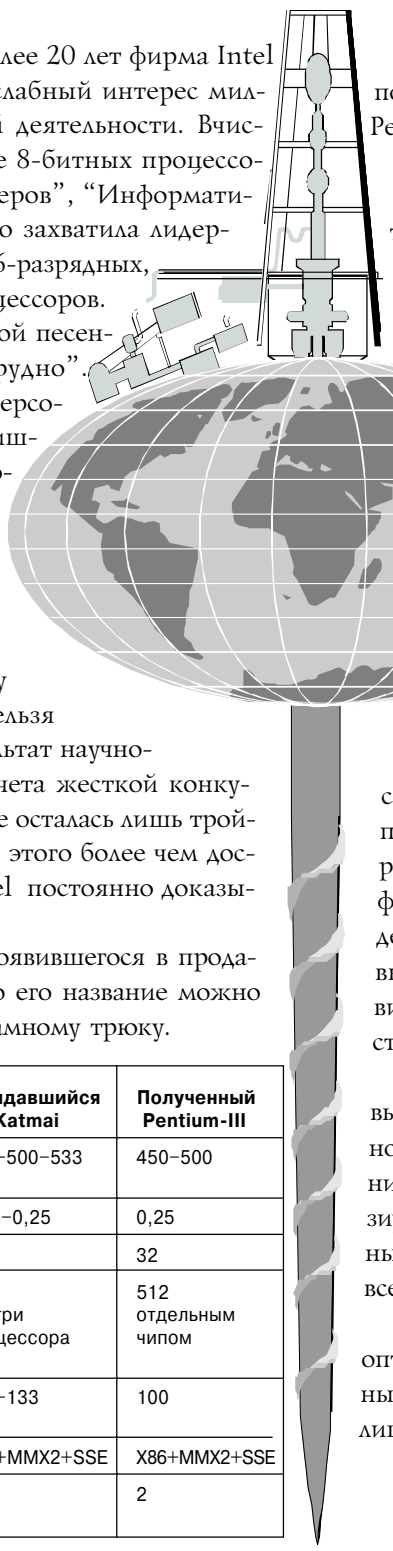
ИНФОРМАТИК

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

Что там inside?

Заметки о новом процессоре Pentium-III

Вот уже на протяжении более 20 лет фирма Intel старается поддерживать неослабный интерес миллионов людей к собственной деятельности. Вчистую проиграв битву на рынке 8-битных процессоров (см.: "История компьютеров", "Информатика", № 11/99), она уверенно захватила лидерство в производстве сначала 16-разрядных, а затем и 32-разрядных процессоров.



Но, как поется в популярной песенке, "лидерам сегодня — трудно". Продажа процессоров для персональных компьютеров — слишком уж лакомый кусочек, чтобы им позволили наслаждаться в одиночку. В разное время фирмы IBM, NexGen, CYRIX, AMD, Centaur Technology активно наступали на пятки Intel, и поэтому выпуск новой продукции нельзя рассматривать лишь как результат научно-технического прогресса без учета жесткой конкурентной борьбы. Сейчас в игре осталась лишь тройка, замыкающая список, но и этого более чем достаточно, чтобы заставлять Intel постоянно доказывать свое первенство.

Что же касается недавно появившегося в продаже процессора Pentium-III, то его название можно всецело отнести к чисто рекламному трюку.

Характеристики	Pentium-II	Ожидавшийся Katmai	Полученный Pentium-III
Тактовая частота в МГц	233–400	450–500–533	450–500
Технология в мкм	0,25	0,18–0,25	0,25
Кэш L1 в Кбайтах	32	64	32
Кэш L2 в Кбайтах	512 отдельным чипом	512 внутри процессора	512 отдельным чипом
Частота шины в МГц	66–100	100–133	100
Набор команд	X86+MMX	X86+MMX2+SSE	X86+MMX2+SSE
Напряжение питания в В	2	1,8	2

Давайте посмотрим, чем же он отличается от появившегося в апреле 1998 года процессора Pentium-II.

Для этого полезно проанализировать таблицу. Не вдаваясь в особые технические подробности, отметим, что увеличение размера кэша первого уровня до 64K могло бы привести к увеличению производительности на 5—10%. И с этой точки зрения тактовая частота вносит гораздо больший вклад в производительность.

Но в нашей таблице есть еще одна строка, которая, собственно, и определяет новизну нового процессора. Это наличие 70 новых команд для обработки чисел с плавающей точкой и двух дополнительных в наборе MMX.

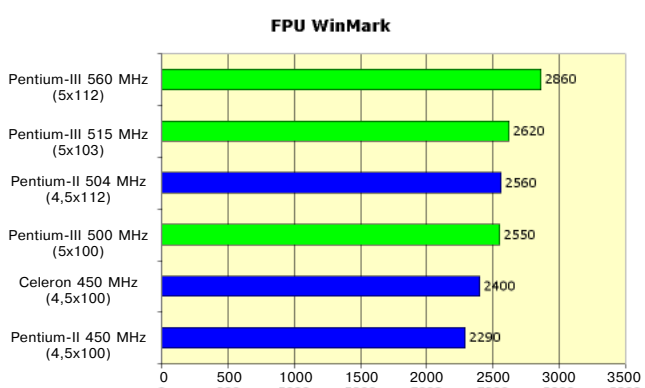
Новые команды появляются в процессорах от Intel регулярно. В 1997 году весь компьютерный мир был взбудоражен рекламным роликом, в котором одетые в диковинные скафандры люди, изображавшие инженеров, выделывали замысловатые па, символизируя новые возможности по обработке звуковой и видеоинформации процессоров Pentium с приставкой MMX.

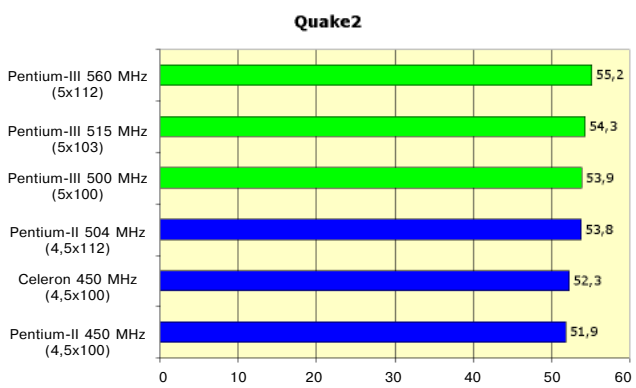
Из этой затеи в общем-то ничего хорошего не вышло. Дело в том, что разработчики программного обеспечения, не имея в своем распоряжении компилятор низкого уровня, могли оптимизировать свои продукты лишь за счет ассемблерных вставок, что в нынешнее время далеко не всем по карману.

Из известных фирм лишь ADOBE выпустила оптимизированный под MMX Photoshop. Остальные дожидаются компилятора, который обещан лишь к июню 1999 года.

Окончание на с. 16

Сравнительные диаграммы производительности различных процессоров





НАШИ ДЕТИ БУДУТ ЖИТЬ В XXI ВЕКЕ

Системы счисления и компьютерная арифметика

Е.В. АНДРЕЕВА, И.Н. ФАЛИНА

Продолжение. Начало в № 14, 15, 16/99.

Ранее была опубликована часть I — "Представление информации (базовый курс)" и начата публикация части II "Системы счисления" (Глава 1. "Позиционные системы счисления").

Теперь представлены:

Глава 2. "Арифметические операции в позиционных системах счисления".

Глава 3. "Перевод чисел из одной позиционной системы счисления в другую".

Продолжение следует

Современные форматы графических файлов

Вечера с Вовой

А.Г. ЛЕОНОВ

Я прожил все юные годы в самом центре Москвы, рядом с Кремлем. Там я ходил в школу, оттуда будучи студентом ездил в МГУ на занятия. Но уже достаточно давно я переехал на юг столицы, в большой шестнадцатиэтажный дом. На окраине намного больше детей и подростков, чем в центре, что и говорить — "спальный" район. Дети сейчас растут быстро, как губка, впитывают всю новую информацию. Самым большим увлечением для современных мальчишек является компьютер и все, что с ним связано. Вот такой интересующийся мальчишка по имени Вова живет с мной на одном этаже и часто заходит в гости, подолгу стоя за моей спиной, пока я брожу по просторам Internet. И задает кучу вопросов: "что это?", "как это?" и т.д. А я терпеливо отвечаю на его вопросы, шаг за шагом раскрывая перед Вовой "страшные компьютерные тайны". И ему эти вечера очень нравятся...

Этот материал базируется на содержании "вечерних бесед" с Вовой. Конечно, я добавил свои комментарии, где это требуется.

1. О пользе соглашений. Вечер первый

Вова опять сидит у меня за спиной. А я работаю за компьютером, корректирую цветопередачу только что загруженной фотографии (использую программу Ulead Photo Express): больше красного, еще больше, еще чуть-чуть... Вроде все. Вполне естественный цвет. Сохраняю результат работы в файле. Вова молча следит за моими действиями. Печатаю файл на принтере Olympus 330 (принтер использует специальную бумагу, получаются почти натуральные фотографии). И тут Вова спрашивает:

— Дядя Саша (дядя Саша — это я. — **прим. автора, дяди Саши**),— а как компьютер понимает, что в файле находится фотография?

Вова уже "продвинутый" мальчик: он свободно ориентируется в Internet, управляется с Windows и многим другим.

— Компьютер сам, конечно, не понимает, — отвечаю я, — да ему это и не нужно. Понимает программа, которую исполняет компьютер. Сейчас у меня исполняется специальная программа Ulead Photo Express, она предназначена для обработки графических файлов, т.е. файлов, содержащих закодированные изображения.

— Закодированных, то есть зашифрованных, чтобы никто не смог посмотреть? — спросил Вова.

— Закодировать и зашифровать — это разные вещи. Когда шифруют, то целью является создать файл, который невозможно прочесть чужаку. Когда кодируют, напротив, просто преобразуют информацию, представляя в некоторой удобной форме (например, для хранения на диске). Это может быть известное тебе двоичное кодирование или какое-нибудь еще. Закодировать можно тысячами способов, но очень важно, чтобы тот, кто кодирует, и тот, кто потом раскодирует, понимали друг друга. То есть у них было бы принято соглашение о методах кодирования.

> Тут необходимо пояснение. Соглашение подразумевает формальное описание того, что означает каждый байт (или бит) закодированной информации. Это и принято называть форматом. Формат описывает метод кодирования, но не фиксирует алгоритм преобразования. Алгоритмов, вообще говоря, может быть много, но все они приводят к одинаковому результату. То есть их можно не различать, для простоты считая, что формат — это соглашение об алгоритмах кодирования/раскодирования.

— Соглашение? — удивился Вова. — Эти соглашения подписывает президент страны? Вон по телевизору показывают?

— Почти. Президенты подписывают соглашения между странами. То есть фиксируют взаимное поведение. Это нужно политикам и простым гражданам. А соглашение об алгоритме кодирования должны знать те, кто работает с закодированной информацией. Если ты кодируешь и затем пересылаешь мне закодированную информацию...

— Через Internet? — перебил меня Вова.

— Да, можно через Internet. Я получаю твой закодированный файл, но если мы не договорились, не "подписали" соглашение об алгоритме кодирования/раскодирования, то твоя информация для меня просто бесполезна. Смотри! — Я написал "ЧЕПУХА".

— Что такое "RENIXA"? — спросил меня мальчик.

— Ну вот и ты купился! Это не РЕНИКСА, а ЧЕПУХА, ты не понял мое послание, потому что мы не договорились, у нас не было соглашения писать по-английски или по-русски. И содержание, смысл послания для тебя потерян. То есть одним из важных пунктов при хранении и передаче информации является выработка соглашения об алгоритмах кодирования/раскодирования. Вот и ответ на твой вопрос: программы для работы с фотографиями используют общепринятые соглашения о кодировании графической информации, т.е. изображений. Именно поэтому эти программы могут отличить, находится ли в файле фотография или там текст письма.

— Или это звуковой файл. — Без сомнения, Вова был сообразительным мальчиком.

— Да, или звуковой файл. Попробуем "подсунуть" чужой файл вот этой программе. — Я имел в виду Ulead Photo Express. — Смотри, она вообще не хочет открывать файл, в котором нет картинки. По расширению имени файла программа определяет, является ли этот файл картинкой или нет. Если файл имеет имя, оканчивающееся на GIF или BMP, то это файл с картинкой.

— Тут целый список! — воскликнул Вова, — и PNG, и TIF, и JPG, и PCX, и еще, и еще...

— Да, различным методам кодирования картинок соответствуют различные расширения в именах файлов. Но попробуем обмануть программу Ulead Photo Express, переименовав файл с письмом, заменив ему расширение TXT на BMP. И снова попробуем его открыть. Смотри! Сообщение об ошибке, программа не может его раскодировать. Значит, не только расширение является определяющим для программы, но и содержимое файла. Не соблюдается соглашение, значит, там не картинка.

— Я не понимаю, как теперь программа Ulead Photo Express отличила письмо от картинки? — недоумевал мой прилежный ученик.

— Все дело в соглашении. Соглашение описывает, что должно находиться в файле с расширением BMP. Это соглашение мы с тобой будем называть "форматом графического файла". Каждый тип графического файла имеет собственный формат, собственное соглашение об алгоритмах кодирования/раскодирования информации, хранящейся в нем.

— И все графические редакторы умеют работать со всеми различными файлами?

— Нет, не все. Например, твой любимый Paint, простейший графический редактор из Windows 98, умеет читать только PCX и JPG-файлы. А другие типы файлов не умеет. Напротив, Ulead Photo Express умеет работать с большим числом различных типов графических файлов. Есть еще масса программ, которые умеют работать практически со всеми наиболее известными типами графических файлов.

— Сколько же всего различных форматов?

— Популярных — десяток и более, тебе для ознакомления хватит и шести:

BMP	TIF
GIF	PNG
PCX	JPG

Прежде чем рассказывать про каждый из них, скажи мне: как на экране компьютера изображаются картинки?

— Ну, это просто, — улыбнулся Вова, — экран состоит из точек, растра точек, как сетка, вот сейчас на экране режим 800×600 = 480 000, 800 точек по горизонтали и 600 по вертикали. Еще бывают режимы 1024×768 и 1280×1024. Эти режимы круче. Каждая из 480 000 точек горит своим цветом. Сейчас 24 бита на цвет точки (*pixel*). Этот режим называется *true color*. Еще есть *high color* и режим 256 цветов.

— Исчерпывающая информация. — Я был доволен. — Так и придумай способ, как сохранять картинку в файле. В качестве подсказки припомни собственные слова, 24 бита на точку!

— Придумал! — закричал мальчик после минутной паузы, — сохраняем весь растр по точкам, на каждую точку приходится по 24 бита=3 байта, т.е. картинка займет 800×600×3 = 1 440 000 байт, то есть около 1,4 мегабайта.

— Давай проведем эксперимент. Жми клавишу Print Screen (*print screen*).

Нажал?

Хорошо! Теперь загружай Paint и жми Ctrl V — вспомни картинку, затем сохрани ее и посмотри размер. Сколько? Около 1,47? Не удивляет ли тебя, что размер файла почти совпадает с размером файла в придуманном тобой формате?

Это потому, что именно так устроен самый простой из графических форматов — BMP. Все вышеперечисленные форматы предназначены для хранения растровой графической информации, где кодируется каждая точка изображения. Но есть и другие способы. Представь себе, что ты рисуешь на листке бумаги карандашом. Пусть у тебя на запястье находится специальное устройство, передающее в компьютер все движения твоей руки. Тогда законченный тобой рисунок можно сохранить в файле не только как набор точек (растр), а как последовательность движений твоей руки (с карандашом). Этот метод представления графической информации называется векторным.

— А зачем его придумали, — удивился Володя, — неужели недостаточно растрового?

— Представь, что ты нарисовал треугольник. В BMP-файле тебе потребуются все те же 1,4 Мб, а если рисовать треугольник векторами, указывая координаты начала и конца отрезков, а координаты двухбайтовые? Какой объем файла потребуется для сохранения такого треугольника?

— 16 байт. — Ответ прозвучал через несколько секунд.

Задача 1. Почему Вова сказал, что ответ — 16 байт? Поясните.

Продолжение в следующем номере

ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Д.М. ЗЛАТОПОЛЬСКИЙ

Динамическое программирование[1] — особый метод поиска оптимальных решений, специально приспособленный к так называемым "многошаговым", или "многоэтапным", операциям [1].

Приведем несколько примеров задач, в которых рассматривается такая "многошаговая", т.е. представляющая собой последовательность "шагов" ("этапов"), операция.

Задача 1. Нужно проложить путь, соединяющий два пункта — *A* и *B* (*рис.* 1), второй из которых лежит к северо-востоку от первого. Прокладка пути состоит из ряда шагов, и на каждом шаге мы можем двигаться либо строго на север, либо строго на восток, т.е. любой путь из *A* в *B* представляет собой ступенчатую ломаную линию, отрезки которой параллельны одной из координатных осей. Затраты на прокладку каждого из таких отрезков известны. Требуется проложить такой путь из *A* в *B*, при котором суммарные затраты минимальны.

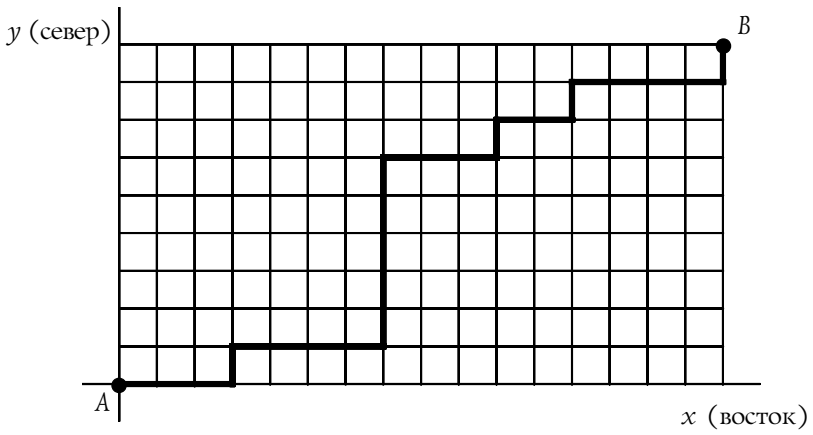


Рис. 1

Задача 2. Имеется определенный набор предметов — $П_1$, $П_2$, ..., $П_n$ (каждый в единственном экземпляре); известны их веса — q_1, q_2, ..., q_n — и стоимость — c_1, c_2, ..., c_n. Грузоподъемность машины равна *Q*. Спрашивается: какие из предметов нужно взять в машину, чтобы их суммарная стоимость (при суммарном весе $\le Q$) была максимальна?

Задача 3. Имеется какой-то объем денежных средств *Q*, который должен быть распределен между предприятиями $П_1$, $П_2$, ..., $П_n$. Каждое из предприятий $П_i$ при вложении в него каких-то средств в объеме *x* приносит доход, зависящий от *x*, т.е. представляющий собой какую-то функцию $f_i(x)$. Все функции $f_i(x)$ $(i = 1, 2, ..., n)$ заданы (разумеется, эти функции — неубывающие). Спрашивается: сколько средств нужно выделить каждому предприятию, чтобы в сумме они дали максимальный доход?

В первой задаче операцией является строительство пути из точки *A* в точку *B*. Здесь шаги выделены уже в условии задачи — шагом является прокладывание очередного отрезка пути.

В постановке второй задачи нет упоминания о времени. Но процесс загрузки машины можно представить как состоящий из *n* шагов, считая за первый шаг принятие решения о том, брать или не брать первый предмет, за второй — то же относительно второго предмета и т.п.

В третьей задаче также можно операцию распределения средств мысленно развернуть в какой-то последовательности и рассматривать решение вопроса о вложении средств в предприятие $П_1$ как первый шаг, в предприятие $П_2$ — как второй и т.д.

В каждой из этих задач общий выигрыш есть сумма выигрышей на каждом этапе.

Как можно решать подобного рода задачи? Очевидно, что это можно сделать по-разному.

Можно перебрать все возможные варианты решения и выбрать среди них лучший. Например, в первой задаче можно рассмотреть все возможные варианты пути и выбрать тот, на котором затраты минимальны.

А можно выбирать лучший вариант пути шаг за шагом, на каждом этапе расчета оптимизировать только один шаг. Обычно второй способ оптимизации оказывается проще, чем первый, особенно при большом числе шагов [2].

Такая идея постепенной, пошаговой оптимизации и лежит в основе метода динамического программирования. К этой идее мы вернемся чуть ниже, а пока заметим, что при решении задач методом динамического программирования употребляются следующие понятия [1], которые будут использоваться и нами:

1) система;
2) шаг (этап);
3) состояние системы;
4) управление;
5) выигрыш.

Точные определения каждого из этих понятий дать достаточно сложно, поэтому мы просто проиллюстрируем их на примере перечисленных задач (см. *табл.* 1).

Таблица 1

Понятие	**Задача 1**	**Задача 2**	**Задача 3**
Система	Сооружаемый путь	Загружаемый автомобиль	Распределяемые средства
Шаг (этап)	Выбор очередного, *i*-го отрезка пути	Принятие решения о том, нужно ли брать в машину *i*-й предмет или нет (i=1, 2, ..., n)	Определение объема средств, выделяемых *i*-му предприятию (i=1, 2, ..., n)
Состояние	Точка, в которой может находиться конец уже построенного до *i*-го шага участка пути	Вес, который еще можно взять в машину после предыдущих шагов	Объем еще не вложенных средств
Управление	Строительство очередного участка в северном или в восточном направлении	Берем *i*-й предмет или не берем	Объем средств, выделяемых *i*-му предприятию
Выигрыш	Затраты на строительство*	Стоимость загруженных в машину предметов	Доход от вложения средств

* На самом деле это, конечно, не "выигрыш", а "проигрыш", который надо минимизировать.

Вернемся теперь к идее пошаговой оптимизации. На первый взгляд она может показаться довольно тривиальной. В самом деле, чего, казалось бы, проще: если трудно оптимизировать процесс в целом, надо разбить его на ряд шагов и оптимизировать каждый шаг. Это сделать нетрудно — надо выбрать на этом шаге такое управление, чтобы эффективность этого шага была максимальна. Не так ли?

Нет, вовсе не так! Принцип динамического программирования отнюдь не предполагает, что каждый шаг оптимизируется о т д е л ь н о, независимо от других. Напротив, управление на каждом шаге должно выбираться дальновидно, с учетом его последствий в будущем. Что толку, если мы выберем на данном шаге управление, при котором эффективность э т о г о шага максимальна, если он приведет нас к проигрышу на последующих шагах?

А как выбрать оптимальное управление на том или ином шаге? Ведь на каждом шаге (естественно, кроме первого) система может находиться в общем случае в нескольких состояниях (в различных точках местности, на которой сооружается путь из *A* в *B*, с различным весом груза, который еще можно взять в машину, и т.п.), а в каждом из этих состояний в общем случае возможны несколько вариантов управления (строить очередной участок пути в северном или в восточном направлении, брать или не брать *i*-й предмет, вкладывать в *i*-е предприятие один из возможных объемов средств или не вкладывать вообще). Какое же решение является оптимальным? Ведь мы не знаем даже, чем закончился предыдущий шаг.

Так как ответить на этот вопрос непросто, давайте решим другую задачу: определим, какое управление является лучшим для *каждого* из состояний системы на том или ином шаге (иными словами, сделаем разные предположения о том, чем закончился предыдущий шаг). Такое управление назовем условным оптимальным управлением для этого состояния, а выигрыш, который дает это управление, — условным оптимальным выигрышем (условное потому, что оно выбирается исходя из условия, что предыдущий шаг кончился так-то и так-то).

Выбрать такой вариант управления можно, сравнив все возможные управления по следующему критерию: *сумма выигрыша от реализации того или иного управления на данном шаге и условного оптимального выигрыша в состоянии системы, в которое она перейдет в результате этого управления* (помните о необходимости на каждом шаге принимать "дальновидные" решения?)[3].

Очевидно, что для использования такого критерия необходимо знать условные оптимальные выигрыши для всех состояний системы на следующем шаге. А из этого, в свою очередь, следует очень важный вывод: определение условных оптимальных параметров надо начинать с последнего шага, затем найти их для предпоследнего шага и т.д.

Для последнего шага сделать это не так уж сложно. Ведь для каждого состояния перед последним шагом известно, какое управление необходимо применить, — только то, которое приведет к конечному состоянию системы (путь должен закончиться в точке *B*, автомобиль должен быть полностью загружен, все средства должны

Продолжение на с. 4

[1] Термин "программирование" в названии метода происходит от слова "программа" в значении *план действий* и не связано с понятием *компьютерная программа*. Поэтому динамическое программирование иначе называют динамическим планированием. Происхождение же термина "динамический" связано с использованием метода в задачах принятия решений через фиксированные промежутки времени.

[2] При большом числе шагов в ряде случаев решение первым способом вообще не может быть реализовано. В задаче 1, например, общее число возможных вариантов пути равно числу сочетаний из (N_1+N_2) по N_1, где N_1 — число отрезков от точки *A* до точки *B* в восточном направлении, N_2 — в северном. При N_1=10 и N_2=10 общее число вариантов равно 184 756. При увеличении N_1 и N_2 это число существенно возрастает (в [2] показано, что при N_1=30 и N_2=30 компьютер, выполняющий 100 000 операций в секунду, рассмотрит все возможные варианты пути за более чем 100 000 лет!).

[3] Не вызывает сомнения тот факт, что именно по такому критерию пришлось бы выбирать оптимальный вариант управления в случае, когда какое-то состояние было бы *исходным*.

ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Продолжение. Начало на с. 3

быть вложены). Такое управление будем называть вынужденным. Запишем условные оптимальные показатели для последнего шага в таблицу (см. *табл.* 2). Это же будем делать и на других шагах.

Таблица 2

Состояние системы	Последний шаг		Предпоследний шаг		...	Второй шаг		Первый шаг	
	УОУ	УОВ	УОУ	УОВ		УОУ	УОВ	УОУ	УОВ

Примечание. УОУ — условное оптимальное управление;
УОВ — условный оптимальный выигрыш.

Определив необходимые параметры для всех состояний последнего шага, можно перейти к рассмотрению ситуации перед предпоследним шагом. На этом этапе система также может находиться в нескольких состояниях. Но для каждого состояния на этом шаге в общем случае возможно *несколько* вариантов управления (например, можно строить очередной участок пути в северном или в восточном направлении, брать или не брать *i*-й предмет, вкладывать в *i*-е предприятие один из возможных объемов средств или не вкладывать вообще). Выбрать из них условное оптимальное управление мы сможем по критерию, который упоминался чуть выше (ведь для каждого из возможных вариантов управления известно, в какое новое состояние перейдет система в результате, а для каждого состояния последнего шага уже найден условный оптимальный выигрыш). Выбор удобно проводить с использованием таблицы следующего вида:

Таблица 3

Управление	Выигрыш от этого управления на данном шаге	Состояние, в которое перейдет система в результате управления	Условный оптимальный выигрыш в состоянии, указанном в графе 3	Критерий сравнения вариантов управления (сумма граф 2 и 4)
1	2	3	4	5

Аналогичным образом можно определить условное оптимальное управление и условный оптимальный выигрыш для каждого состояния предпредпоследнего и других шагов, включая второй.

После этого остается рассмотреть первый шаг. Его особенность в том, что на нем система находится в одном, исходном, состоянии. Для этого состояния также можно выбрать условное оптимальное управление, которое на самом деле является уже не условным, а безусловным. Заметим попутно, что полученный для данного состояния оптимальный выигрыш есть наибольший выигрыш для *всех* шагов решения задачи.

Далее, применив оптимальное управление на первом шаге, мы переведем систему в какое-то новое состояние на втором шаге, для которого уже известен оптимальный вариант действий. Реализовав его, мы перейдем к следующему шагу, который также можем выполнить оптимальным способом, и т.д. до последнего шага, т.е. найдем решение задачи.

Как отмечается в [1], полное понимание основных положений динамического программирования, как правило, возможно только после рассмотрения ряда примеров, поэтому перейдем к решению перечисленных выше задач.

Задача 1

Разделим расстояние от *A* до *B* в восточном направлении, скажем, на 7 частей, а в северном — на 5 (в принципе дробление может быть сколь угодно мелким). Тогда любой путь из *A* в *B* состоит из 7+5=12 отрезков, направленных на восток или на север (*рис.* 2). Проставим на каждом из отрезков число, выражающее (в каких-то условных единицах) стоимость прокладки пути по этому отрезку. По требованию условия задачи необходимо выбрать такой путь из *A* в *B*, при котором сумма чисел, стоящих на отрезках, минимальна.

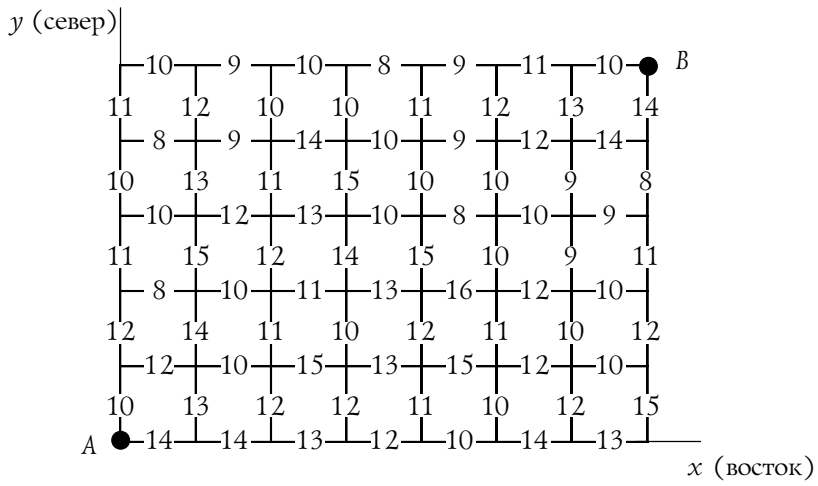


Рис. 2

Будем рассматривать сооружаемый путь как управляемую систему, перемещающуюся под влиянием управления (строительства очередного участка) из начального состояния *A* в конечное *B*. Состояние системы перед началом каждого шага будем характеризовать двумя координатами: восточной x и северной y, обе целочисленные ($0 \le x \le 7$, $0 \le y \le 5$). Для каждого из состояний системы (узловой точки прямоугольной сетки на *рис.* 2) мы должны найти условное оптимальное управление: идти нам на север или на восток? Выбирается это управление так, чтобы стоимость всех оставшихся до конца шагов (включая данный) была минимальна. Эту стоимость мы договорились называть "условным оптимальным выигрышем" для данного состояния системы (хотя в данном случае это, конечно, не "выигрыш", а "проигрыш").

Как указывалось выше, определение условных оптимальных параметров (управления и выигрыша) надо начинать с последнего, 12-го шага. Рассмотрим отдельно правый верхний угол нашей прямоугольной сетки (*рис.* 3). Где мы можем находиться после 11-го шага? Только там, откуда за один (последний) шаг можно попасть в *B*, т.е. в одной из точек B_1 или B_2. Если мы находимся в точке B_1, у нас нет выбора (управление вынужденное): надо идти на восток, и это обойдется нам в 10 единиц. Запишем это число 10 в кружке у точки B_1, а оптимальное управление покажем короткой стрелкой, исходящей из B_1 и направленной на восток. Для точки B_2 управление тоже вынужденное (на север), затраты до конца равны 14, мы их запишем в кружке у точки B_2. Таким образом, условная оптимизация последнего шага сделана, условный оптимальный выигрыш для обоих состояний этого шага найден и записан в соответствующем кружке.

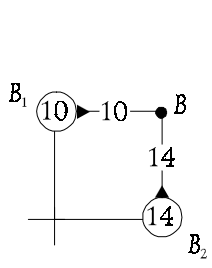


Рис. 3

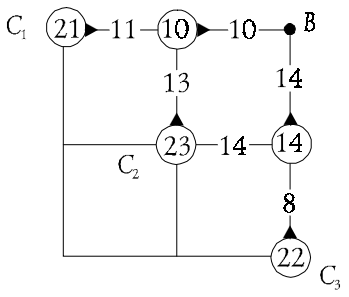


Рис. 4

Теперь давайте оптимизировать предпоследний (11-й) шаг. После предпредпоследнего (10-го) шага мы могли оказаться в одной из точек C_1, C_2, C_3 (*рис.* 4). Найдем для каждой из них условное оптимальное управление и условный оптимальный выигрыш. Для точки C_1 управление вынужденное — идти на восток; обойдется это нам до конца в 21 единицу (11 на данном шаге плюс 10 — условный оптимальный выигрыш для точки B_1, записанный в кружке). Число 21 записываем в кружок при точке C_1. Для точки C_2 управление уже не вынужденное: мы можем идти как на восток, так и на север. В первом случае мы затратим на данном шаге 14 единиц и от B_2 до конца — еще 14, всего 28 единиц. Если пойдем на север, затратим 13+10=23 единицы. Значит, условное оптимальное управление в точке C_2 — идти на север (внимательный читатель, очевидно, заметил, что найдено это управление по критерию, о котором говорилось выше). Отмечаем его стрелкой, а число 23 записываем в кружок у точки C_2. Для точки C_3 управление снова вынужденное — идти на север; обойдется это до конца в 22 единицы (ставим стрелку на север, число 22 записываем в кружок у точки C_3).

Аналогично, "пятясь" от предпоследнего шага назад, найдем для каждой точки условное оптимальное управление, которое обозначим стрелкой, и условный оптимальный выигрыш, который запишем в кружке. Вычисляется он так: расход на данном шаге складывается с уже оптимизированным расходом, записанным в кружке, куда ведет стрелка. Таким образом, на каждом шаге мы оптимизируем только этот шаг, а следующие за ним уже оптимизированы.

Конечный результат показан на *рис.* 5. Из *рис.* 5 видно, что в какой бы из узловых точек мы ни находились, мы знаем, куда идти (стрелка) и во что нам обойдется путь до конца (число в кружке), а в кружке у точки *A* записан оптимальный выигрыш (минимальные затраты) на все сооружение пути из *A* в *B*!

Теперь остается построить путь, ведущий из *A* в *B*, самым дешевым способом. Для этого надо только "слушаться стрелок", т.е. прочитать, что они предписывают делать на каждом шаге. Такая оптимальная траектория пути отмечена на *рис.* 5 дважды обведенными кружками.

Как заметил читатель, для нахождения условного оптимального управления в каждой точке мы двигались по диагоналям, параллельным диагонали B_1—B_2. При этом каждая диагональ находилась юго-западнее предыдущей. Так, следующей после диагонали B_1—B_2 была диагональ C_1—C_3 и т.д. Возможен и другой способ, который мы сейчас опишем. Сначала найдем условное оптимальное управление в самых северных точках, учитывая, что здесь управление вынужденное — только на восток. Для этого движемся от точки *B* на запад до первой точки северной горизонта-

ЗАДАЧИ

ли. Затем находим условное оптимальное управление в самых восточных точках. Здесь управление также вынужденное — только на север, и мы движемся от точки *B* на юг до первой точки восточной вертикали. Теперь можно найти условное оптимальное управление во всех остальных точках. Движемся по горизонталям, начиная со второй с севера и кончая самой южной, а вдоль горизонтали — от второй точки с востока вплоть до самой западной. (Разумеется, тот же результат получим, двигаясь по вертикалям, начиная со второй с востока вплоть до самой западной, а внутри вертикали от второй точки с севера и кончая самой южной.) Наверное, движение по диагоналям выглядит более естественным, но для программной реализации более подходит второй способ.

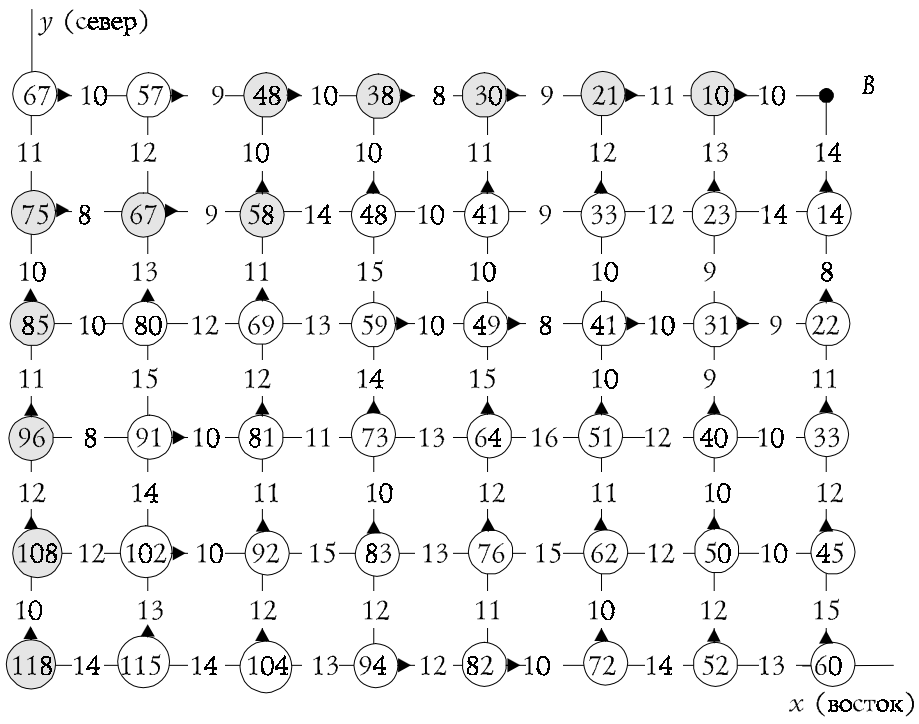


Рис. 5

Перейдем к программной реализации описанного алгоритма. Обозначим число отрезков на север и на восток через `Nnord` и `Neast`. Значения условных оптимальных выигрышей для каждого состояния (числа в кружках на *рис.* 5) и соответствующие условные оптимальные управления (направление отрезка пути в виде символов “в” и “с”) будем записывать в массивы `W[0: Nnord, 0: Neast]` и `a[0: Nnord, 0: Neast]`. При этом первая строка этих массивов соответствует южной горизонтали, а первый столбец — западной вертикали, индексы точки *A* — это (0, 0), точки *B* — `(Nnord, Neast)`. Данные о затратах на прокладку пути по восточным отрезкам (*рис.* 2) храним в массиве `Seast[0: Nnord, 1: Neast]`. Аналогично затраты на прокладку северных отрезков хранятся в массиве `Snord[1: Nnord, 0: Neast]`. Чтобы не усложнять работу с программами, значения элементов этих массивов задаются как константы. Представляем программы[4].

Школьный алгоритмический язык

```
алг Задача 1
нач цел Neast, Nnord, i, j, k, Wnord, Weast
  | Neast:=7 |Число восточных отрезков
  | Nnord:=5 | Число северных отрезков
  | сим таб a[0:Nnord,0:Neast]
  | цел таб W[0:Nnord,0:Neast]
  | цел таб Seast[0:Nnord,1:Neast], Snord[1:Nnord,0:Neast]
  | Заполняем массив Seast
  | Seast[0,1]:=14; Seast[0,2]:=14; Seast[0,3]:=13; Seast[0,4]:=12
  | Seast[0,5]:=10; Seast[0,6]:=14; Seast[0,7]:=13
  | Seast[1,1]:=12; Seast[1,2]:=10; Seast[1,3]:=15; Seast[1,4]:=13
  | Seast[1,5]:=15; Seast[1,6]:=12; Seast[1,7]:=10
  | Seast[2,1]:= 8; Seast[2,2]:=10; Seast[2,3]:=11; Seast[2,4]:=13
  | Seast[2,5]:=16; Seast[2,6]:=12; Seast[2,7]:=10
  | Seast[3,1]:=10; Seast[3,2]:=12; Seast[3,3]:=13; Seast[3,4]:=10
  | Seast[3,5]:= 8; Seast[3,6]:=10; Seast[3,7]:= 9
  | Seast[4,1]:= 8; Seast[4,2]:= 9; Seast[4,3]:=14; Seast[4,4]:=10
  | Seast[4,5]:= 9; Seast[4,6]:=12; Seast[4,7]:=14
  | Seast[5,1]:=10; Seast[5,2]:= 9; Seast[5,3]:=10; Seast[5,4]:= 8
  | Seast[5,5]:= 9; Seast[5,6]:=11; Seast[5,7]:=10
  | Заполняем массив Snord
  | Snord[1,0]:=10; Snord[2,0]:=12; Snord[3,0]:=11; Snord[4,0]:=10
  | Snord[5,0]:=11
  | Snord[1,1]:=13; Snord[2,1]:=14; Snord[3,1]:=15; Snord[4,1]:=13
  | Snord[5,1]:=12
  | Snord[1,2]:=12; Snord[2,2]:=11; Snord[3,2]:=12; Snord[4,2]:=11
  | Snord[5,2]:=10
  | Snord[1,3]:=12; Snord[2,3]:=10; Snord[3,3]:=14; Snord[4,3]:=15
  | Snord[5,3]:=10
  | Snord[1,4]:=11; Snord[2,4]:=12; Snord[3,4]:=15; Snord[4,4]:=10
  | Snord[5,4]:=11
  | Snord[1,5]:=10; Snord[2,5]:=11; Snord[3,5]:=10; Snord[4,5]:=10
  | Snord[5,5]:=12
  | Snord[1,6]:=12; Snord[2,6]:=10; Snord[3,6]:= 9; Snord[4,6]:= 9
  | Snord[5,6]:=13
  | Snord[1,7]:=15; Snord[2,7]:=12; Snord[3,7]:=11; Snord[4,7]:= 8
  | Snord[5,7]:=14
  | Заполняем массивы W и a
  | а) их последнюю строку (самые северные точки)
  W[Nnord,Neast]:=0
  нц для j от Neast – 1 до 0 шаг –1
  |  a[Nnord, j]:="в"            |управление вынужденное
  |  W[Nnord, j]:=Seast[Nnord,j+1]+ W[Nnord,j+1]
  кц
  | б) их правый столбец (самые восточные точки)
  нц для i от Nnord – 1 до 0 шаг –1
  |  a[i, Neast]:="с"            |управление вынужденное
  |  W[i, Neast]:=Snord[i+1,Neast]+W[i+1,Neast]
  кц
  | в) остальные элементы
  | Движемся по строкам от предпоследней строки к первой,
  | а вдоль строки  от предпоследнего элемента к первому
  нц для i от Nnord – 1 до 0 шаг –1
  |  нц для j от Neast – 1 до 1 шаг –1
  |  |  Wnord:=W[i+1,j]+Snord[i+1,j] |Северные затраты
  |  |  Weast:=W[i,j+1]+Seast[i,j+1] |Восточные затраты
  |  |  если Weast>Wnord
  |  |  |  то     |идем на север
  |  |  |     a[i, j]:="с"
  |  |  |     W[i, j]:=Wnord
  |  |  |  иначе          |идем на восток
  |  |  |     a[i, j]:="в"
  |  |  |     W[i, j]:=Weast
  |  |  все
  |  кц
  кц
  | Печатаем результаты
  вывод нс, "Минимальные затраты на сооружение пути: ", w[1,1]
  вывод нс, "Оптимальные направления отрезков пути:"
  i:=1; j:=1
  нц для k от 1 до NNord+NEast
  |  вывод a[i, j]," "
  |  если  a[i, j]="с"
  |  |  то
  |  |  |  i:=i+1
  |  |  иначе
  |  |  |  j:=j+1
  |  все
  кц
кон
```

Язык Паскаль

```
Const
{Число восточных и северных отрезков пути}
Neast=7; Nnord=5;
{Затраты восточных отрезков пути}
Seast:array[0..Nnord,1..Neast] of integer=(
(14, 14, 13, 12, 10, 14, 13),  {горизонталь 0}
(12, 10, 15, 13, 15, 12, 10),  {горизонталь 1}
( 8, 10, 11, 13, 16, 12, 10),  {горизонталь 2}
(10, 12, 13, 10,  8, 10,  9),  {горизонталь 3}
( 8,  9, 14, 10,  9, 12, 14),  {горизонталь 4}
(10,  9, 10,  8,  9, 11, 10)); {горизонталь 5}
{Затраты северных отрезков пути}
Snord:array[1..Nnord,0..Neast] of integer=(
{          Вертикали
 0   1   2   3   4   5   6   7 }
(10, 13, 12, 12, 11, 10, 12, 15),
(12, 14, 11, 10, 12, 11, 10, 12),
(11, 15, 12, 14, 15, 10,  9, 11),
(10, 13, 11, 15, 10, 10,  9,  8),
(11, 12, 10, 10, 11, 12, 13, 14));
Var
 i,j,k,Weast,Wnord:integer;
 a:array[0..Nnord,0..Neast] of char;
 W:array[0..Nnord,0..Neast] of integer;
BEGIN
 {Заполняем массивы W и a
 а) их последнюю строку (самые северные точки) }
 W[Nnord,Neast]:=0;
 for j:=Neast–1 downto 0 do begin
  a[Nnord,j]:='в'; {Управление вынужденное}
  W[Nnord,j]:=W[Nnord,j+1]+Seast[Nnord,j+1]
                              end;
 {б) их правый столбец (самые восточные точки) }
 for i:=Nnord–1 downto 0 do begin
  a[i,Neast]:='c'; {Управление вынужденное}
  W[i,Neast]:=W[i+1,Neast]+Snord[i+1,Neast]
                              end;
 {в) остальные элементы.
 Движемся по строкам от предпоследней строки к первой,
 а вдоль строки  от предпоследнего элемента к первому    }
 for i:=Nnord–1 downto 0 do{Цикл строк}
  for j:=Neast–1 downto 0 do begin{Цикл вдоль строки}
```

Окончание на с. 6

[4] Все программы приведены на нескольких языках программирования: на школьном алгоритмическом языке [3, 4], Паскале, Бейсике (вариант QuickBasic [5]) и Си.

ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Окончание. Начало см. на с. 3—5

```
  Wnord:=W[i+1,j]+Snord[i+1,j];{Северные затраты}
  Weast:=W[i,j+1]+Seast[i,j+1];{Восточные затраты}
  {Выбираем направление с меньшими затратами}
  if Wnord<Weast then {на север}
   begin a[i,j]:='c'; W[i,j]:=Wnord end
  else {на восток}
   begin a[i,j]:='в'; W[i,j]:=Weast end
                              end;{Цикл по строке}
{Печатаем результаты}
writeln('Минимальные затраты на сооружение пути ',W[0,0]);
writeln('Оптимальные направления отрезков пути');
i:=0; j:=0;
for k:=1 to Nnord+Neast do begin
write(a[i,j],' ');
if a[i,j]='c' then inc(i) else inc(j)
                             end;
END.
```

Язык Бейсик

```
DEFINT I-W
'Число восточных и северных отрезков пути
Neast = 7: Nnord = 5
DIM Seast(0 TO Nnord, 1 TO Neast)
DIM Snord(1 TO Nnord, 0 TO Neast)
DIM a(0 TO Nnord, 0 TO Neast) AS STRING
DIM W(0 TO Nnord, 0 TO Neast)
'Затраты восточных отрезков пути записываем по горизонталям
DATA 14, 14, 13, 12, 10, 14, 13
DATA 12, 10, 15, 13, 15, 12, 10
DATA  8, 10, 11, 13, 16, 12, 10
DATA 10, 12, 13, 10,  8, 10,  9
DATA  8,  9, 14, 10,  9, 12, 14
DATA 10,  9, 10,  8,  9, 11, 10
FOR i = 0 TO Nnord 'Формирование матрицы Seast
 FOR j = 1 TO Neast: READ Seast(i, j): NEXT j
NEXT i             'Формирование матрицы Seast
'Затраты северных отрезков пути записываем по вертикалям
DATA 10, 12, 11, 10, 11
DATA 13, 14, 15, 13, 12
DATA 12, 11, 12, 11, 10
DATA 12, 10, 14, 15, 10
DATA 11, 12, 15, 10, 11
DATA 10, 11, 10, 10, 12
DATA 12, 10,  9,  9, 13
DATA 15, 12, 11,  8, 14
FOR j = 0 TO Neast 'Формирование матрицы Snord
 FOR i = 1 TO Nnord: READ Snord(i, j): NEXT i
NEXT j             'Формирование матрицы Snord
'Заполняем массивы W и a
'а)их последнюю строку (самые северные точки)
W(Nnord, Neast) = 0
FOR j = Neast - 1 TO 0 STEP -1
 a(Nnord, j) = "в" 'Управление вынужденное
 W(Nnord, j) = W(Nnord, j + 1) + Seast(Nnord, j + 1)
NEXT j
'б) их правый столбец (самые восточные точки)
FOR i = Nnord - 1 TO 0 STEP -1
 a(i, Neast) = "c" 'Управление вынужденное
 W(i, Neast) = W(i + 1, Neast) + Snord(i + 1, Neast)
NEXT i
'в) остальные элементы.
'Движемся по строкам от предпоследней строки к первой,
'а вдоль строки от предпоследнего элемента к первому
FOR i = Nnord - 1 TO 0 STEP -1 'Цикл строк
 FOR j = Neast - 1 TO 0 STEP -1 'Цикл вдоль строки
  Wnord = W(i + 1, j) + Snord(i + 1, j)'Северные затраты
  Weast = W(i, j + 1) + Seast(i, j + 1)'Восточные затраты
  'Выбираем направление с меньшими затратами
  IF Wnord < Weast THEN 'на север
   a(i, j) = "c": W(i, j) = Wnord
  ELSE 'на восток
   a(i, j) = "в": W(i, j) = Weast
  END IF
 NEXT j                       'Цикл по строке
NEXT i                        'Цикл строк
'Печатаем результаты
     PRINT "Минимальные затраты на сооружение пути "; W(0, 0)
     PRINT "Оптимальные направления отрезков пути"
     i = 0: j = 0
     FOR k = 1 TO Nnord + Neast
      PRINT a(i, j); " ";
      IF a(i, j) = "c" THEN i = i + 1 ELSE j = j + 1
     NEXT k
     END
```

Язык Си

```
#include <stdio.h>
#define Qeast 8 /*Число вертикалей*/
#define Qnord 6 /*Число горизонталей*/
/*Затраты восточных отрезков пути. Начальный столбец не используется*/
int Seast[Qnord][Qeast]={
{0, 14, 14, 13, 12, 10, 14, 13},  /*горизонталь 0*/
{0, 12, 10, 15, 13, 15, 12, 10},  /*горизонталь 1*/
{0,  8, 10, 11, 13, 16, 12, 10},  /*горизонталь 2*/
{0, 10, 12, 13, 10,  8, 10,  9},  /*горизонталь 3*/
{0,  8,  9, 14, 10,  9, 12, 14},  /*горизонталь 4*/
{0, 10,  9, 10,  8,  9, 11, 10}}; /*горизонталь 5*/
/*Затраты северных отрезков пути. Начальная строка не используется*/
int Snord[Qnord][Qeast]={
/*         Вертикали
 0   1   2   3   4   5   6   7 */
{ 0,  0,  0,  0,  0,  0,  0,  0},
{10, 13, 12, 12, 11, 10, 12, 15},
{12, 14, 11, 10, 12, 11, 10, 12},
{11, 15, 12, 14, 15, 10,  9, 11},
{10, 13, 11, 15, 10, 10,  9,  8},
{11, 12, 10, 10, 11, 12, 13, 14}};
void main()
{int i,j,k,Weast,Wnord,Neast,Nnord;
 char a[Qnord][Qeast]; int W[Qnord][Qeast];
 Neast=Qeast-1;/*Число восточных отрезков пути*/
 Nnord=Qnord-1;/*Число северных отрезков пути*/
 /*Заполняем массивы W и a
 a)их последнюю строку (самые северные точки)*/
 W[Nnord][Neast]=0;
 for (j=Neast-1; j >= 0; j--)
  {a[Nnord][j]='в'; /*Управление вынужденное*/
   W[Nnord][j]=W[Nnord][j+1]+Seast[Nnord][j+1];
  }
 /*б) их правый столбец (самые восточные точки)*/
 for (i=Nnord-1; i >= 0; i--)
  {a[i][Neast]='c'; /*Управление вынужденное*/
   W[i][Neast]=W[i+1][Neast]+Snord[i+1][Neast];
  }
 /*в) остальные элементы. Движемся по строкам от предпоследней
строки к первой, а вдоль строки от предпоследнего элемента к первому*/
 for (i=Nnord-1; i >= 0; i--)
  for (j=Neast-1; j >= 0; j--)
   {Wnord=W[i+1][j]+Snord[i+1][j];/*Северные затраты*/
    Weast=W[i][j+1]+Seast[i][j+1];/*Восточные затраты*/
    /*Выбираем направление с меньшими затратами*/
    if (Wnord<Weast) {a[i][j]='c'; W[i][j]=Wnord;} /*на север*/
    else {a[i][j]='в'; W[i][j]=Weast;} /*на восток*/
   }
 /*Печатаем результаты*/
 printf("\nМинимальные затраты на сооружение пути %d",W[0][0]);
 printf("\nОптимальные направления отрезков пути\n");
 i=0; j=0;
 for (k=1; k<=Nnord+Neast; k++)
  {printf("%c ",a[i][j]);
   if (a[i][j] == 'c') i++; else j++;
  }
}
```

Выполнив любую из этих программ с исходными данными, представленными на *рис.* 2, можно обнаружить, что оптимальный выигрыш (минимальные затраты) на сооружение пути из *A* в *B* равен 118, а достигается он при следующем наборе отрезков пути:

с, с, с, с, в, в, с, в, в, в, в, в.

Интересно: а какой результат был бы получен, если бы мы решали задачу наивным способом, о котором говорилось выше, т.е. выбирая на каждом шаге, начиная с первого, самое выгодное для этого шага направление? *Из рис.* 2 видно, что таким способом мы получим путь, состоящий из отрезков

с, с, в, в, в, в, с, в, в, в, с, с.

Подсчитаем расходы для такого пути. Они будут равны

$$10 + 12 + 8 + 10 + 11 + 13 + 15 + 8 + 10 + 9 + 8 + 14 = 128,$$

что больше, чем найденное методом динамического программирования значение 118. В данном случае разница не очень велика, но в других случаях она может быть существеннее.

Отметим также еще один момент. В ходе условной оптимизации мы можем столкнуться со случаем, когда оба возможных управления для какого-то состояния (какой-то точки на плоскости) являются оптимальными, т.е. приводят к одинаковому расходу средств от этой точки до конца. Например, в точке с координатами (1; 4) оба управления являются оптимальными и дают расход до конца, равный 67. Из них мы можем выбрать любое (в нашем случае мы выбрали "в"; с тем же успехом мы могли выбрать "с" — общий выигрыш был бы тем же). Такие случаи неоднозначного выбора оптимального управления часто встречаются в динамическом программировании.

Окончание в следующем номере

ЛИТЕРАТУРА

1. *Вентцель Е.С.* Исследование операций: задачи, принципы, методология. М.: Наука, 1988.
2. Играем с мозаикой списков и слов на Лого. Сборник задач /Составители *Горлицкая С.И., Кузнецова И.Н.* Информатика № 6, 1997.
3. *Кушниренко А.Г., Лебедев Г.В., Сворень Р.А.* Основы информатики и вычислительной техники. М.: Просвещение, 1990.
4. *Эпиктетов М.Г.* Почему школьный алгоритмический? Информатика № 24, 1995.
5. *Зельднер Г.А.* QuickBasik 4.5. М.: ABF, 1994.
6. *Фаронов В.В.* Программирование на персональных ЭВМ в среде Турбо Паскаль. М.: Изд-во МГТУ, 1992.
7. *Ребрина В.А., Попик А.* Динамическое программирование и его применение к решению некоторых задач. Информатика № 13, 1998.
8. *Беллман Р.* Динамическое программирование. М.: Иностранная литература, 1960.
9. *Таха Х.* Введение в исследование операций. М.: Мир, 1985.
10. *Вентцель Е.С.* Элементы динамического программирования. М.: Наука, 1964.
11. *Кузнецкий Е.М.* О динамическом программировании. Информатика № 15, 1996.

ЗАДАЧИ

При наличии непериодической части у исходной P-ичной дроби мы будем поступать так же, как и в первом способе перевода.

Пример 3.13.

$$0{,}10(1001)_2 = 0{,}1_2 + 2^{-2} \cdot \frac{1001_2}{1111_2} = 0{,}5 + \frac{9}{4 \cdot 15} = 0{,}65_{10}.$$

Таким образом, в третьем способе перевода периодических P-ичных дробей в десятичную систему сложным является лишь его обоснование, а вычисления по сравнению с первыми двумя способами, наоборот, оказываются довольно простыми.

3.4. Перевод целого числа из десятичной системы счисления в *P*-ичную

Существует по крайней мере два способа организации вычислений при решении этой проблемы.

Способ 1. Им мы уже пользовались при переводе чисел в двоичную систему счисления в параграфе 6.2 части 1. Запишем число a в P-ичной системе счисления в развернутой форме (1.7)

$$a = a_n \cdot P^n + a_{n-1} \cdot P^{n-1} + \ldots + a_1 \cdot P + a_0,$$

где a_n, a_{n-1}, …,a_1, a_0 пока неизвестны. Тогда, разделив a на P с остатком, получаем целое частное

$$a_n \cdot P^{n-1} + a_{n-1} \cdot P^{n-2} + \ldots + a_1 \qquad (3.3)$$

и a_0 в остатке, который не превышает $P-1$. Таким образом, мы получили последнюю цифру в P-ичной записи числа a.

Разделим полученное частное (3.3) вновь на P, получив в остатке значение a_1, а $a_n \cdot P^{n-2} + a_{n-1} \cdot P^{n-1} + \ldots + a_2$ — новое частное. Таким образом, мы получили уже предпоследнюю цифру в P-ичной записи числа a. Продолжим этот процесс, пока результат деления не станет равен нулю. Более формально данный способ можно записать так.

Алгоритм перевода целого числа из десятичной системы счисления в *P*-ичную:

1) *делим исходное число a на P нацело в десятичной системе и записываем в качестве нового значения десятичного числа a целую часть результата от деления;*

2) *повторяем деление до тех пор, пока число a не станет равным 0, выписывая остатки от деления справа налево и получая при этом запись исходного числа a в P-ичной системе счисления.*

Пример 3.14. Переведем число 123 в троичную систему счисления.

123:3 = 41(**0**). Здесь и далее в скобках указано значение остатка от деления, следовательно, последняя цифра в троичном представлении числа 123 равна 0.

41:3 = 13(**2**)
13:3 = 4(**1**)
4:3 = 1(**1**)
1:3 = 0(**1**).

Окончательный результат: $123 = 11120_3$.

Пример 3.15. Переведем это же число в шестнадцатеричную систему счисления:

123:16=7(**11**)
7:16=0(**7**)

Заменив число 11 на шестнадцатеричную цифру B, получим в результате $123 = 7B_{16}$.

Типичные ошибки при реализации этого алгоритма разбирались в параграфе 6.2 первой части.

Способ 2. Он основан на выделении максимальной степени числа P в исходном десятичном числе.

Заменим в записи числа вида (1.7) все цифры на максимальную,равную $P-1$. Тогда, очевидно, выполняются следующие неравенства: (первое — т.к. $a_n \geq 1$):

$$P^n \leq a_n \cdot P^n + a_{n-1} \cdot P^{n-1} + \ldots + a_1 \cdot P + a_0 \leq (P-1) \cdot P^n + (P-1) \cdot P^{n-1} + \ldots + (P-1) \cdot P + (P-1) = P^{n+1} - P^n + P^n - P^{n-1} + \ldots + P^2 - P + P - 1 = P^{n+1} - 1 < P^{n+1}.$$

Значит, для того чтобы определить левую цифру в P-ичной записи числа — a_n, необходимо найти такую степень числа P, для которой выполняются неравенства: $P^n \leq a < P^{n+1}$. То есть a_n будет равно целой части от деления a на P^n. Остатком же от такого деления является число $a_{n-1} \cdot P^{n-1} + \ldots + a_1 \cdot P + a_0 \geq 0$.

Обозначим его, как и раньше, a. Если оно равно нулю, то и все цифры a_{n-1}, …,a_1, a_0 равны 0 и вычисления заканчиваются, в противном случае мы опять ищем максимальную степень k числа P, для которой справедливо:

$$P^k \leq a_{n-1} \cdot P^{n-1} + \ldots + a_1 \cdot P + a_0 < P^{k+1} \leq P^n.$$

Тогда $n-1-k$ следующих за a_n цифр будут равны нулю (при $n-1=k$ нулевые цифры между a_n и a_k отсутствуют), а a_k получаем в результате деления нацело a на P^k. Пока остаток от такого деления не окажется равен нулю, будем продолжать описанные действия.

Пример 3.16. Переведем данным способом число 100 в двоичную систему счисления.

Используя таблицу степеней двойки, запишем неравенство $2^6 < 100 < 2^7$. Следовательно, двоичная запись числа 100 будет состоять из 7 цифр. Остаток от деления 100 на 2^6 равен 36. Так как $2^5 < 36 < 2^6$, то и вторая слева цифра равна единице. Остаток же от деления 36 на 2^5 равен $4 = 2^2$, то есть третья и четвертая, а также шестая и седьмая цифры в двоичной записи числа 100 нулевые, а $100_{10} = 1100100_2$.

Пример 3.17. Переведем в шестнадцатеричную систему число 525:

Используя таблицу степеней числа 16, запишем неравенство $256 = 16^2 < 525 < 16^3$. Запись числа 525 в шестнадцатеричной системе будет иметь три цифры.

Разделим 525 на 256 — и получим 2 целых и 13 в остатке, следовательно, старшая цифра в шестнадцатеричной записи — 2.

В силу того, что $13 < 16^1$, вторая цифра в шестнадцатеричном представлении 525 равна 0. Заменяем 13 на соответствующую шестнадцатеричную цифру D, в результате получаем число $525_{10} = 20D_{16}$.

Заметим, что второй способ перевода эффективен лишь тогда, когда нам уже известны значения всех степеней числа P, которые не превосходят исходное число. Преимущество же такого метода состоит в естественном порядке записи получившихся P-ичных цифр (слева направо), что бывает важно при программировании: очередная полученная цифра сразу же может выводиться на печать. В первом способе, как вы помните, все цифры требуют предварительного запоминания для последующей распечатки результата в порядке, обратном их получению.

Системы счисления и компьютерная арифметика

Е.В. АНДРЕЕВА, И.Н. ФАЛИНА

Продолжение. Начало см. в № 14, 15, 16/99

Глава 2
Арифметические операции в позиционных системах счисления

2.1. Сложение и вычитание

Во всех традиционных позиционных системах арифметические операции выполняются по одним и тем же правилам, согласно таблицам сложения и умножения. Для них также справедливы правила сложения, вычитания, умножения и деления столбиком, знакомые нам по действиям в десятичной системе и опирающиеся на таблицы сложения и умножения десятичных цифр.

В P-ичной системе счисления таблица сложения представляет результаты сложения каждой цифры алфавита P-ичной системы с любой другой цифрой этой же системы. Составление подобной таблицы не составляет труда. Каждый элемент таблицы равен предыдущему в строке или в столбце, увеличенному на единицу по правилам прибавления единицы в P-ичной системе счисления (первые вычисляемые элементы в строке или столбце равны базовой цифре этой же строки или столбца, так как соответствуют прибавлению к ней нуля).

Выпишем таблицу сложения в шестнадцатеричной системе счисления, опуская нижний индекс 16 в обозначении шестнадцатеричных чисел:

+	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***A***	***B***	***C***	***D***	***E***	***F***
0	0	1	2	3	4	5	6	7	8	9	*A*	*B*	*C*	*D*	*E*	*F*
1	1	2	3	4	5	6	7	8	9	*A*	*B*	*C*	*D*	*E*	*F*	10
2	2	3	4	5	6	7	8	9	*A*	*B*	*C*	*D*	*E*	*F*	10	11
3	3	4	5	6	7	8	9	*A*	*B*	*C*	*D*	*E*	*F*	10	11	12
4	4	5	6	7	8	9	*A*	*B*	*C*	*D*	*E*	*F*	10	11	12	13
5	5	6	7	8	9	*A*	*B*	*C*	*D*	*E*	*F*	10	11	12	13	14
6	6	7	8	9	*A*	*B*	*C*	*D*	*E*	*F*	10	11	12	13	14	15
7	7	8	9	*A*	*B*	*C*	*D*	*E*	*F*	10	11	12	13	14	15	16
8	8	9	*A*	*B*	*C*	*D*	*E*	*F*	10	11	12	13	14	15	16	17
9	9	*A*	*B*	*C*	*D*	*E*	*F*	10	11	12	13	14	15	16	17	18
A	*A*	*B*	*C*	*D*	*E*	*F*	10	11	12	13	14	15	16	17	18	19
B	*B*	*C*	*D*	*E*	*F*	10	11	12	13	14	15	16	17	18	19	1A
C	*C*	*D*	*E*	*F*	10	11	12	13	14	15	16	17	18	19	1A	1B
D	*D*	*E*	*F*	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	*E*	*F*	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	*F*	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

Достаточно простыми, например, являются таблицы сложения в двоичной и троичной системах счисления:

+	**0**	**1**
0	0	1
1	1	10_2

+	**0**	**1**	**2**
0	0	1	2
1	1	2	10_3
2	2	10_3	11_3

Приведем примеры арифметических действий, выполненных согласно выписанным ранее таблицам сложения.

Пример 2.1. Сложение столбиком в двоичной системе счисления.

$$\begin{array}{r} 101{,}01_2 \\ +\ \ 1{,}11_2 \\ \hline 111{,}00_2 \end{array}$$

Пример 2.2. Сложение столбиком в троичной системе счисления.

$$\begin{array}{r} 21_3 \\ +\ 2{,}1_3 \\ \hline 100{,}1_3 \end{array}$$

Пример 2.3. Сложение столбиком в шестнадцатеричной системе счисления.

$$\begin{array}{r} F2A_{16} \\ +\ E9_{16} \\ \hline 1013_{16} \end{array}$$

Очевидно, что любая таблица сложения (и умножения) в силу закона коммутативности симметрична относительно главной диагонали (линии, проведенной из левого верхнего в правый нижний угол таблицы).

Имея перед собой соответствующую таблицу сложения, можно осуществлять действия сложения и вычитания столбиком в любой позиционной системе счисления. Несложно показать, что если результат сложения двух цифр в P-ичной системе счисления больше, чем $P-1$ (то есть полученное число — двузначное), то левая цифра всегда равна 1, так как при сложении даже двух самых больших цифр алфавита мы имеем: $(P-1)+(P-1)=2\cdot P-2=1\,[P-2]_P$. Например, в четверичной системе счисления $3+3=12_4$.

Следовательно, при сложении столбиком цифр справа налево в любой системе счисления в следующий разряд может переходить только единица, а результат выполнения сложения в новом разряде все равно будет меньше, чем $2\cdot P$ (максимум $2\cdot P-1=1\,[P-1]_P$). То есть результат сложения двух положительных P-ичных чисел либо имеет столько же цифр, сколько и максимальное из двух слагаемых, либо на одну цифру больше, но этой цифрой может быть только единица.

Вычитание из большего числа меньшего в P-ичной системе счисления тоже можно производить столбиком аналогично вычитанию в десятичной системе с использованием все той же таблицы сложения в P-ичной системе счисления.

Если нам необходимо вычесть из цифры a цифру b и $a \geq b$, то в столбце таблицы сложения для цифры b мы ищем число a, а первая цифра в найденной строке и будет результатом вычитания. Если же $a<b$, то, занимая единицу из левого разряда, мы придем к необходимости выполнения следующего действия: $10_p+a-b=1a_p-b$. Для этого в столбце для цифры b таблицы сложения мы уже ищем число $1a_p$, а первая цифра в соответствующей строке является результатом вычитания.

Пример 2.4. Вычитание в двоичной системе счисления.

$$\begin{array}{r} 101_2 \\ -\ 10,1_2 \\ \hline 10,1_2 \end{array}$$

Пример 2.5. Вычитание в троичной системе счисления.

$$\begin{array}{r} 210_3 \\ -\ 102_3 \\ \hline 101_3 \end{array}$$

Пример 2.6. Вычитание в шестнадцатеричной системе счисления.

$$\begin{array}{r} A10_{16} \\ -\ 102_{16} \\ \hline 90E_{16} \end{array}$$

Если же таблицы сложения в P-ичной системе у вас под рукой нет, а выписать ее целиком затруднительно из-за большого количества цифр, или требуется провести всего одно-два арифметических действия в данной системе счисления, то возможен и другой подход. А именно: переведем каждое из слагаемых (или уменьшаемое и вычитаемое) в десятичную систему счисления, произведем требуемое действие в десятичной системе, а результат запишем в исходной P-ичной системе счисления. С правилами перевода из одной системы счисления в другую можно подробно познакомиться в главе 3.

Очевидно, что аналогичным способом можно поступать и при выполнении действий умножения и деления, но в следующих параграфах мы рассмотрим их выполнение непосредственно в P-ичной системе счисления, так как это необходимо для дальнейшего рассмотрения темы.

2.2. Умножение

Для выполнения умножения двух многозначных чисел в P-ичной системе надо иметь таблицу умножения в этой системе.

Вычисление элементов такой таблицы представляет собой прибавление базовой цифры столбца к числу, стоящему на одну клетку выше. При этом неопределенными оказываются лишь элементы первой строки (они не имеют вышестоящих клеток), однако первая строка соответствует умножению базовой цифры строки на 0, а результат такого умножения в любой системе счисления равен 0.

Приведем таблицы умножения для двоичной, троичной и шестнадцатеричной систем, опуская нижние индексы, указывающие на принадлежность к соответствующей системе счисления (базовые цифры выделены):

Таблица умножения в двоичной системе счисления

×	**0**	**1**
0	0	0
1	0	1

Таблица умножения в троичной системе счисления

×	**0**	**1**	**2**
0	0	0	0
1	0	1	2
2	0	2	11

которых чисто алгебраический. Причем все три способа касаются лишь преобразования периода, а непериодическую дробную часть из P-ичной системы счисления в десятичную следует переводить отдельно, используя один из способов, описанных в предыдущем параграфе.

Способ 1. Пусть непериодическая часть у P-ичной дроби отсутствует и дробь имеет вид $a=0,(a_1a_2...a_k)_P$, т.е. a — чисто периодическая дробь. Умножим эту дробь на P^k, то есть передвинем запятую на k позиций вправо. В результате мы получим некоторое число b (далее мы будем опускать нижний индекс P у цифровой записи числа):

$$b=a\cdot P^k=a_1a_2\ldots a_k,(a_1a_2\ldots a_k)=a_1a_2\ldots a_k+a.$$

Отсюда очевидна связь между числами a и b, что и позволяет нам составить следующее уравнение:

$$P^k a=a+a_1a_2\ldots a_k,$$

— преобразовав его, мы находим, что $a=\dfrac{a_1a_2\ldots a_k}{P^k-1}$.

В результате для записи числа a мы получили обыкновенную дробь, в которой знаменатель записан в десятичной системе счисления, а числитель — в P-ичной. Переводя теперь целое P-ичное число $a_1a_2...a_k$ (это наш числитель) в десятичную систему, получим обыкновенную дробь, записанную уже в десятичной системе счисления и равную исходному числу a. При необходимости ее можно перевести в десятичную дробь (конечную или периодическую).

Пример 3.8. Пусть $a=0,(1001)_2$. Подставляя в вышеполученную формулу, получаем:

$$a=\frac{1001}{2^4-1}=\frac{9}{15}=\frac{3}{5}=0,6_{10}.$$

При наличии непериодической части вычисления изменятся незначительно. Пусть периодическая P-ичная дробь имеет вид: $b=0,c_1c_2\ldots c_n(a_1a_2\ldots a_k)$. Обозначим через a следующую дробь: $a=0,(a_1a_2\ldots a_k)$. Тогда исходную дробь можно выразить через a следующим образом: $b=P^{-n}a+0,c_1c_2\ldots c_n$ (сдвигаем период на n разрядов вправо и прибавляем непериодическую часть).

Затем отдельно друг от друга переведем в десятичную систему конечную дробь $0,c_1c_2\ldots c_n$ (см. параграф 3.2) и периодическую — $a=0,(a_1a_2\ldots a_k)$ так, как было описано в данном параграфе выше.

Пример 3.9.

$0,10(1001)_2=2^{-2}\cdot 0,(1001)_2+0,1_2=2^{-2}\cdot 0,6_{10}+0,1_2=$
$=0,25_{10}\cdot 0,6_{10}+0,5_{10}=0,65_{10}$.

Пример 3.10.

$0,00(1001)_2=2^{-2}\cdot 0,(1001)_2=2^{-2}\cdot 0,6_{10}=0,15_{10}$

Непериодическая часть этой двоичной дроби состоит из двух нулей, но ее нельзя считать отсутствующей, так как она определяет сдвиг периода.

Способ 2. Запишем исходную бесконечную периодическую P-ичную дробь в виде следующей бесконечной суммы:

$$0,c_1c_2\ldots c_n(a_1a_2...a_k)=0,c_1c_2\ldots c_n+$$
$$+P^{-n-k}a_1a_2\ldots a_k+P^{-n-2k}a_1a_2\ldots a_k+P^{-n-3k}a_1a_2\ldots a_k+\ldots$$

За исключением непериодической части, данное выражение соответствует сумме бесконечной геометрической прогрессии со знаменателем $q=P^{-k}<1$ и первым членом $P^{-n-k}\cdot a_1a_2\ldots a_k$. Как известно, сумма такой прогрессии конечна и равна

$$\frac{P^{-n-k}a_1a_2\ldots a_k}{1-P^{-k}}=\frac{P^{-n-k}a_1a_2\ldots a_k}{1-\frac{1}{P^k}}=\frac{a_1a_2\ldots a_k}{P^{n+k}\cdot\frac{P^k-1}{P^k}}=\frac{a_1a_2\ldots a_k}{P^n\cdot(P^k-1)}.$$

Остается лишь записать числитель и знаменатель полученного отношения в десятичной системе счисления и прибавить к результату непериодическую часть дроби, предварительно переведенную в десятичную систему.

Пример 3.11.

$0,10(1001)_2=0,1_2+2^{-6}\cdot 1001_2+2^{-10}\cdot 1001_2+$
$+2^{-14}\cdot 1001_2+...$

Легко увидеть, что слагаемые этой бесконечной суммы, начиная со второго, представляют собой бесконечную геометрическую прогрессию со знаменателем $q=2^{-4}$ и первым членом $b=2^{-6}\cdot 1001_2$. Сумма этой бесконечной геометрической прогрессии равна $\dfrac{2^{-6}\cdot 1001_2}{1-2^{-4}}$.

Переведем числитель полученной величины в десятичную систему счисления:

$2^{-6}\cdot 1001_2=2^{-6}\cdot(2^3+1)=2^{-6}\cdot 9$.

Тогда $0,10(1001)_2=$

$=0,1_2+\dfrac{9}{4\cdot(2^4-1)}=0,5+\dfrac{9}{4\cdot 15}=0,65_{10}$.

Способ 3. Воспользуемся вторым (периодическим) представлением конечных вещественных дробей (см. замечание в конце параграфа 1.4) и представим единицу в виде $1=0,(P_1)_P$, здесь $P_1=[P-1]$ — максимальная цифра в P-ичной системе счисления. Аналогично единицу можно записать и с помощью k периодически повторяющихся максимальных цифр P-ичной системы счисления, где k — количество цифр в периоде интересующей нас исходной P-ичной дроби, например:

$$1=0,\left(\underbrace{1\ldots1}_{k}\right)_2;1=0,\left(\underbrace{4\ldots4}_{k}\right)_5;1=\left(\underbrace{F\ldots F}_{k}\right)_{16};1=0,\left(\underbrace{P_1\ldots P_1}_{k}\right)_P$$

Тогда $a=\dfrac{a}{1}=\dfrac{0,(a_1a_2\ldots a_k)_P}{0,(P_1\ldots P_1)_P}=\dfrac{a_1a_2\ldots a_k}{P_1\ldots P_1}$.

Последняя обыкновенная дробь записана в P-ичной системе счисления. Для преобразования ее в десятичную дробь числитель и знаменатель переводятся, по правилам параграфа 3.1, в десятичную систему счисления, и уже в ней числитель делится на знаменатель.

Пример 3.12.

$0,(1001)_2=\dfrac{1001_2}{1111_2}=\dfrac{9}{15}=\dfrac{3}{5}=0,6$.

Пример 3.2. Переведем двоичное число 1001101_2 в десятичное:

$1001101_2=2^0+2^2+2^3+2^6=77_{10}$.

При программировании на различных алгоритмических языках операцию перевода целых чисел из двоичной системы в десятичную производить приходится довольно часто, поэтому очень полезным оказывается знание первых шестнадцати степеней двойки:

$2^1=2$;	$2^9=512$;
$2^2=4$;	$2^{10}=1024$;
$2^3=8$;	$2^{11}=2048$;
$2^4=16$;	$2^{12}=4096$;
$2^5=32$;	$2^{13}=8192$;
$2^6=64$;	$2^{14}=16384$;
$2^7=128$;	$2^{15}=32768$;
$2^8=256$;	$2^{16}=65536$.

Выше мы уже говорили, что при вычислении значения P-ичного числа по развернутой форме удобно пользоваться ***схемой Горнера***, позволяющей получить результат с использованием минимального числа арифметических операций сложения, умножения и деления, причем операция возведения в степень не используется. Схема Горнера основана на следующих преобразованиях исходного многочлена:

$$R=a_nP^n+a_{n-1}P^{n-1}+...+a_1P^1+a_0=$$
$$=(a_nP^{n-1}+a_{n-1}P^{n-2}+...+a_1)P+a_0=$$
$$=((a_nP^{n-2}+a_{n-1}P^{n-3}+...+a_2)P+a_1)P+a_0= \quad (3.1)$$
$$=(((a_nP^{n-3}+a_{n-1}P^{n-4}+...+a_3)P+a_2)P+a_1)P+a_0=...=$$
$$=((((a_nP+a_{n-1})P+a_{n-2})P+a_{n-3})P+...a_1)P+a_0.$$

Применим общую схему Горнера в конкретном случае.

Пример 3.3. Переведем в десятичную систему число 2143_5:

$2143_5=2\cdot5^3+1\cdot5^2+4\cdot5+3=((2\cdot5+1)\cdot5+4)\cdot5+3=298_{10}$.

3.2. Перевод конечной *P*-ичной дроби в десятичную

Перевод конечных дробей аналогичен переводу целых чисел и основан на формуле (1.8).

Дана правильная конечная дробь b в P-ичной системе счисления, т.е. $b=0,b_{-1}b_{-2}...b_{-k}$. Требуется получить запись этой дроби в десятичной системе счисления.

Для решения этой задачи представим дробь в развернутой форме $b=b_{-1}P^{-1}+b_{-2}P^{-2}+...+b_{-k}P^{-k}$ (формула (1.8)). Для того чтобы вычислить значение многочлена в десятичной системе счисления, следует число P и коэффициенты многочлена (цифры P-ичного числа) записать в виде десятичных чисел и все вычисления проводить в десятичной системе. Запишем эти правила в виде алгоритма.

Алгоритм перевода конечной *P*-ичной дроби в десятичную:

1) *каждая цифра дробной части числа в P-ичной системе счисления переводится в число в десятичной системе;*

2) *полученные в результате преобразования дробной части числа нумеруются слева направо начиная с единицы;*

3) *десятичное число, соответствующее каждой P-ичной цифре, умножается на P^{-k}, где k — номер этого числа из п.2 , и результаты складываются, причем все эти арифметические действия проводятся в десятичной системе.*

Пример 3.4. Переведем число $0,B0F9_{16}$ в десятичную систему счисления.

$0,B0F9_{16}=0,[11_{10}]\ [0]\ [15_{10}]\ [9]=$

$=11\cdot16^{-1}+15\cdot16^{-3}+9\cdot16^{-4}=0,6912994384765625_{10}$

Здесь согласно пункту (2) алгоритма числа были пронумерованы так: 11 — номер один (коэффициент при P в степени минус один), 15 — номер три, 9 — номер четыре.

При вычислении десятичного значения P-ичной дроби по развернутой форме также рекомендуется пользоваться схемой Горнера, что минимизирует количество необходимых арифметических действий.

Получим схему Горнера для вычисления значения P-ичной дроби, выписав цифры дроби в представлении (1.8) в обратном порядке:

$$b_{-k}P^{-k}+b_{-k+1}P^{-k+1}+...+b_{-1}P^{-1}=$$
$$=(b_{-k}P^{-k+1}+b_{-k+1}P^{-k+2}+...+b_{-1})P^{-1}= \quad (3.2)$$
$$=((b_{-k}P^{-k+2}+b_{-k+1}P^{-k+3}+...+b_{-2})P^{-1}+b_{-1})P^{-1}=...$$
$$=((((b_{-k}P^{-1}+b_{-k+1})P^{-1}+b_{-k+2})P^{-1}+b_{-k+3})P^{-1}+...+b_{-1})P^{-1}.$$

Пример 3.5. Вычислим значение двоичной дроби $0,1101_2$, используя схему Горнера.

$0,1101_2=1\cdot2^{-1}+1\cdot2^{-2}+0\cdot2^{-3}+1\cdot2^{-4}=$

$1\cdot2^{-4}+0\cdot2^{-3}+1\cdot2^{-2}+1\cdot2^{-1}=$

$(((1\cdot2^{-1}+0)\cdot2^{-1}+1)\cdot2^{-1}+1)\cdot2^{-1}=$

$(((\frac{1}{2}+0):2+1):2+1):2=((\frac{1}{2}\cdot\frac{1}{2}+1):2+1):2=$

$=(1{,}25:2+1):2=(0{,}625+1):2=0{,}8125.$

3.3. Перевод бесконечной периодической *P*-ичной дроби в десятичную

Напомним, что дробь называется ***бесконечной периодической***, если в ее записи можно выделить последовательно повторяющуюся группу цифр. Минимальная последовательно повторяющаяся группа цифр после запятой в записи числа называется ***периодом***. Период принято записывать один раз, заключая его в круглые скобки:

$0,67132181321813218...=0,67(13218)$.

Если период начинается сразу после запятой, то дробь принято называть ***чисто периодической***.

Задача перевода такой дроби в десятичную систему, кажущаяся на первый взгляд очень сложной, имеет по крайней мере три различных способа решения, один из

Таблица умножения в шестнадцатеричной системе счисления

×	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	0	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E
3	0	3	6	9	C	F	12	15	18	1B	1E	21	24	27	2A	2D
4	0	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	0	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	0	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	0	7	E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	0	8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	0	9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	0	A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	0	B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	0	C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	0	D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	0	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	0	F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

Приведем примеры выполнения умножения в двоичной, троичной и шестнадцатеричной системах счисления. Действия производятся по правилам умножения столбиком с последовательным умножением цифр первого сомножителя на второй сомножитель. При этом используются выписанные выше таблицы умножения для перечисленных систем счисления.

Пример 2.7

$$\begin{array}{r} \times\ 10100_2 \\ \underline{\quad 1010_2} \\ +\ 101 \\ \ \ 101 \\ \hline 11001000_2 \end{array}$$

В данном примере, как и при десятичном умножении столбиком, умножение на 0 не производится, а все оставшиеся справа нули, не участвующие в умножении, приписываются справа к результату умножения.

Пример 2.8

$$\begin{array}{r} \times\ 212_3 \\ \underline{\quad 1210_3} \\ +\ 212 \\ \ 1201 \\ \hline +\ 12222 \\ \ \ 212 \\ \hline 1111220_3 \end{array}$$

Здесь показано, что если складывать столбиком приходится три и более слагаемых, то действия сложения можно производить последовательно, в противном случае сложные вычисления в непривычной системе счисления обычно порождают ошибки.

Пример 2.9

$$\begin{array}{r} \times\ FFA,3_{16} \\ \underline{\quad D,E_{16}} \\ +\ DFA\ EA \\ \ CFB4\ 7 \\ \hline DDAF,5A_{16} \end{array}$$

При умножении P-ичных дробей количество цифр в дробной части результата равно сумме числа цифр в дробной части каждого из множителей (крайние справа цифры результата, равные нулю, можно опустить как незначащие).

С помощью таблицы умножения в шестнадцатеричной системе можно сформулировать признаки делимости на 2, 3, 5, 8, A, F в шестнадцатеричной системе счисления. Так, шестнадцатеричное число делится на 2, если последняя цифра является четной (кроме четных десятичных цифр, четными здесь также являются цифры A, C и E). Число делится на 3, 5 или F, если сумма его цифр делится на 3, 5 и F соответственно (аналогично 3 и 9 в десятичной системе). Делимость на A предполагает наличие последней четной цифры и сумму цифр, делящуюся на 5. Если последняя цифра шестнадцатеричного числа равна 0 или 8, то число делится на 8 (аналогично 5 в десятичной системе).

2.3. Деление

При делении столбиком в P-ичной системе счисления приходится в качестве промежуточных вычислений выполнять действия умножения и вычитания, а следовательно, используются таблицы умножения и сложения в P-ичной системе счисления.

Пример 2.10. Наиболее просто деление организовать в двоичной системе, так как в ней необходимо лишь сравнивать два числа между собой и вычитать из большего числа меньшее.

$$\begin{array}{r|l} 11110_2 & 110_2 \\ \cline{2-2} -\ 110_2 & 101_2 \\ \cline{1-1} \ \ 110 & \\ -\ 110 & \\ \cline{1-1} \ \ 0 & \end{array}$$

Пример 2.11. Деление столбиком в шестнадцатеричной системе счисления.

```
 F127(16) | 8(16)
-8        |-------
 --       | 1E24,E(16)
 71
-70
 --
  12
 -10
 ---
  27
 -20
 ---
  70
 -70
 ---
   0
```

Дело обстоит сложнее, если результат деления не является конечной P-ичной дробью (или целым числом). Тогда при осуществлении операции деления обычно требуется выделить непериодическую часть дроби и ее период. Продемонстрируем это на следующих примерах:

Пример 2.12. Деление в троичной системе счисления.

```
 10(3)  | 2(3)
-2      |------
 --     | 1,(1)(3)
 10
-10
 --
  2
 -2
 --
  1
```

Так как результат последнего вычитания совпал с предыдущим, то все остальные цифры дробной части результата совпадут с последней найденной цифрой. Повторяющаяся цифра образует период троичной дроби.

Пример 2.13. Рассмотрим еще один пример деления, выполненного в двоичной системе счисления. Предварительно заметим, что в общем случае неизвестно, получим мы конечную дробь или бесконечную периодическую.

```
 1010(2)  | 11(2)
-11       |-------------
 --       | 11,0101..(2)=11,(01)
 100
-11
 ---
  100
 -11
 ---
  100
 -11
 ---
  100
 -11
 ---
   1
```

В этом примере период дроби состоит из двух цифр.

Умение выполнять операцию деления в P-ичной системе счисления окажется полезным в дальнейшем при переводе дробных чисел из одной системы счисления в другую.

2.4. Задачи и решения

1. Подсчитайте сумму троичных чисел в диапазоне от 10_3 до 100_3, включая границы диапазона. Ответ запишите в троичной системе счисления.

Решение. Для решения этой задачи выпишем все троичные числа, попадающие в указанный диапазон: 10_3, 11_3, 12_3, 20_3, 21_3, 22_3, 100_3. Теперь нам надо найти сумму этих чисел. Будем выполнять сложение последовательно в троичной системе счисления (нижний индекс 3 опускаем). Перенос в следующий разряд делаем, если в разряде получилось число, большее или равное трем.

10+11= 21;
21+11= 102;
102+12= 121;
121+20= 211;
211+21=1002;
1002+21=1101;
1101+22=1200;
1200+100=2000.

2. Восстановите цифры двоичных чисел, на месте которых в приведенном примере стоит знак “$*$”: $1*01_2+1**_2=10100_2$.

Решение. Запишем этот пример на сложение столбиком (нижний индекс 2 опускаем):

```
+  1*01
   1**
  -----
  1*100
```

Отметим, что вместо знака “$*$” может стоять либо “0”, либо “1”, т.к. мы работаем в двоичной системе счисления.

Будем анализировать проведенную операцию поразрядно, начиная с самого младшего (нулевого) разряда.

0-й разряд: $1+*=0$, т.к. значение в младшем разряде суммы — ноль, то произошел перенос в следующий разряд, следовательно, вместо “$*$” надо поставить “1”: $1+1=10_2$.

1-й разряд: учитывая перенос из 0-го разряда, запишем $0+*+1_{\text{перенос}}=0$. Очевидно, что вместо “$*$” надо поставить “1”, т.к. и здесь произошел перенос в следующий разряд. Получаем $1+1=10_2$, т.е. ноль в текущем разряде и перенос единиц в следующий разряд.

2-й разряд: учитывая перенос из 1-го разряда, запишем $*+1+1_{\text{перенос}}=1$. Так как $1+1=10_2$, а в результате стоит единица, а не ноль, следовательно, вместо “$*$” должна стоять “1” и опять произошел перенос единицы в следующий разряд.

3-й разряд: учитывая перенос из 2-го разряда, запишем $1+1_{\text{перенос}}=*$. Так как $1+1=10_2$, то “$*$” заменяем на “0” и запоминаем, что произошел перенос единицы в следующий разряд.

4-й разряд: ни в одном из слагаемых нет значащей цифры в 4-м разряде, “1”, стоящая в 4-м разряде суммы, получена за счет переноса, и мы верно восстановили недостающие цифры.

Ответ: $1101+111=10100$.

3. Выпишите в пятеричной системе счисления все четные числа из диапазона от 1 до 20.

Решение. Для решения этой задачи надо, во-первых, выписать все числа, попадающие в указанный интервал, а во-вторых, знать, какие числа являются четными.

Выпишем все числа, попадающие в указанный интервал:

1, 2, 3, 4, 10, 11, 12, 13, 14, 20.

Число называется четным, если оно делится на два без остатка. Чтобы выполнить операцию деления в пятеричной системе счисления, надо иметь таблицу умножения в пятеричной системе. Но эту задачу можно решить гораздо проще. Так как числа выписаны подряд в порядке возрастания и последовательность чисел начинается с нечетного числа, то каждое второе число будет четным.

Ответ: четные числа — 2, 4, 11, 13, 20.

2.5. Контрольные вопросы и упражнения

1. Выпишите таблицы сложения и умножения в пятеричной и двенадцатеричной системах счисления.

2. Выполните следующие действия сложения:

11010101_2+1110_2; 1234_5+4321_5; $ABBA_{12}+BABA_{12}$.

3. Выполните следующие действия вычитания:

11010101_2-1110_2; 4321_5-1234_5; $BABA_{12}-ABBA_{12}$.

4. Выполните следующие действия умножения:

$11010101_2 \cdot 1110_2$; $4321_5 \cdot 123_5$; $ABBA_{12} \cdot 10A_{12}$.

5. Выполните следующие действия деления:

$10010000_2:1110_2$;

$4322_5:3_5$; $ABO6_{12}:A_{12}$.

6. Восстановите цифры двоичной системы счисления, на месте которых в приведенных ниже арифметических действиях стоит знак “$*$”:

а) $**0*0*1**1_2+10111*10**_2=100*1*00010_2$;

б) $***0**00_2-11*11*11_2=1101*1_2$.

7. Подсчитайте сумму двоичных чисел в диапазоне от 100_2 до 111_2, включая границы диапазона. Ответ запишите в двоичной системе счисления.

8. Подсчитайте сумму шестнадцатеричных чисел в диапазоне от 100_{16} до $1A1_{16}$, включая граничные числа. Ответ запишите в шестнадцатеричной системе счисления.

9. Найдите среднее арифметическое троичных чисел из диапазона 10_3 до 100_3, включая границы диапазона. Ответ запишите в троичной системе счисления.

10. Найдите сумму шестнадцатеричных чисел

$F,1+E,2+D,3+C,4+B,5+A,6+6,A+5,B+4,C+3,D+2,E+1,F.$

Ответ запишите в десятичной системе счисления.

11. Даны числа в четверичной системе счисления от 1 до 33. Выпишите все нечетные числа.

12. Более сложные задачи:

Выполните следующие действия деления с выделением непериодической части и периода: $11_2:101_2$; $10010101_2:1110_2$; $10_3:11_3$; $4321_5:3_5$.

13. Какие признаки делимости вы можете сформулировать для двенадцатеричной системы счисления?

Даны числа в четверичной системе счисления от 1 до 33. Выпишите все числа, делящиеся на 3 без остатка.

Глава 3. Перевод чисел из одной позиционной системы счисления в другую

3.1. Перевод целого числа из P-ичной системы счисления в десятичную

Дано число в P-ичной системе счисления $a=a_n a_{n-1}...a_1 a_0$. Требуется получить запись этого числа в десятичной системе счисления.

Для решения этой задачи представим число в развернутой форме $a=a_n P^n+a_{n-1}P^{n-1}+...+a_1 P+a_0$ (формула (1.7)). Для того чтобы получить значение этого многочлена, записанное в десятичной системе счисления, следует число P и коэффициенты при степенях P (цифры P-ичного числа) записать в виде десятичных чисел и все вычисления провести в десятичной системе. Данный способ можно сформулировать в виде следующего алгоритма.

Алгоритм перевода целых чисел из P-ичной системы счисления в десятичную.

1) каждая цифра числа в P-ичной системе счисления переводится в число в десятичной системе;

2) полученные числа нумеруются справа налево начиная с нуля (номера соответствуют степеням P в многочлене (1.7));

3) десятичное число, соответствующее каждой P-ичной цифре, умножается на P^k, где k — номер этого числа из пункта 2, и результаты складываются, причем все эти арифметические действия проводятся в десятичной системе.

Пример 3.1. Переведем число $B0F9_{16}$ в десятичную форму записи:

$B0F9_{16}=[11_{10}]\,[0]\,[15_{10}]\,[9]=11_{10} \cdot 16_{10}^3+15_{10} \cdot 16_{10}+9=45305_{10}$.

Здесь цифра B была заменена на десятичное число 11, а цифра F — на 15. Согласно пункту (2) алгоритма перевода, числа были пронумерованы так: 9 — коэффициент при нулевой степени многочлена, 15 — при первой, 0 — при второй (т.е. такая степень в многочлене отсутствует), 11 — при третьей.

Описанный алгоритм приводит к уже знакомой нам процедуре перевода чисел из двоичной системы в десятичную: надо в десятичной системе счисления сложить те степени двоек, которые соответствуют единицам в записи исходного двоичного числа. Нумерация степеней, как и ранее, ведется справа налево начиная с нулевой.

Логические задачи на вступительных экзаменах по информатике

В.И. РАКИТИН

От редакции

Публикуя данную статью, редакция считает необходимым явно сформулировать свою позицию по вопросу вступительных экзаменов по информатике. К сожалению, в настоящее время с ними сложилась ситуация, немыслимая в отношении экзаменов по любому другому общеобразовательному предмету. Наличие школьных учебников, программ, рекомендованного минимума содержания образования, по-видимому, совершенно не интересует тех, кто составляет задания вступительных экзаменов в большинстве вузов. Складывается впечатление, что вузы стремятся "переплюнуть" друг друга: кто составит задачки "позаковыристее"? Успешно сдать такие экзамены без посещения подготовительных (платных, разумеется) курсов весьма проблематично.

В сложившейся ситуации мы, ни в коем случае не стремясь поощрять это безобразие, хотим прежде всего предоставить нашим читателям максимум информации о том, что может ждать их выпускников. Поэтому мы публикуем эту статью и благодарим ее автора за составление данной подборки задач.

Методический обзор содержит практически все типы задач по элементам логики, предлагавшиеся на вступительных экзаменах по информатике в различных вузах страны. Здесь нет упражнений, подобранных к каждому разделу; их можно найти в методических пособиях по информатике для поступающих, издаваемых в вузах.

Для успешного решения задач по теме "Булева алгебра логики" абитуриент должен владеть:

- основными определениями и понятиями, такими, как: логическая переменная, логические операции над ними, таблицы истинности для логических операций, приоритеты выполнения операций, логическое выражение, логическая функция;
- основными законами алгебры логики для преобразования логических выражений;
- понятиями о различных формах логических функций: ДНФ, КНФ, СДНФ, СКНФ, минимальной форме;
- основами теории множеств и алгебраическими операциями над ними;
- переводом целых чисел из одной системы счисления в другую.

1. Основы булевой алгебры

Переменные булевой алгебры — высказывания, принимающие одно из двух значений: **1** (истина, **true**, или −1 в некоторых языках программирования) или **0** (ложь, или **false**). Верному высказыванию ставится в соответствие **1**; ложному — **0**. Переменные будем обозначать буквами латинского алфавита. Определены три основные операции над булевыми переменными:

- Сложение (дизъюнкция, логическое "или") — операция над двумя переменными A, B (бинарная операция). Обозначения: $A+B$, $A \vee B$, A ***or*** B. Читается: A или B.
- Умножение (конъюнкция, логическое "и") — операция над двумя переменными A, B (бинарная операция). Обозначения: $A \cdot B$, $A \wedge B$, A ***and*** B, $A\&B$. Читается: A и B.
- Отрицание A (дополнение к A, инверсия A) — операция над одной переменной. Обозначения: $\overline{A}$, ***not*** A, $\neg A$. Читается: не A.

Операции сложения, умножения, отрицания определяются следующей таблицей истинности:

A	B	$A + B$	$A \cdot B$	$\overline{A}$
0	0	0	0	1
0	1	1	0	1
1	0	1	0	0
1	1	1	1	0

Кроме трех основных операций, рассмотрим еще две:

- Импликация (следование) — операция над двумя переменными A, B. Обозначение: $A \Rightarrow B$. Читается: "из A следует B", или "если A, то B", или "A — достаточное условие для B", или "B — необходимое условие для A", или "A — частный случай B".
- Эквиваленция (равенство) — операция над двумя переменными A, B. Обозначения: $A \Leftrightarrow B$, $A \sim B$. Читается: "A тогда и только тогда, когда B", или "A — необходимое и достаточное условие B", или "A эквивалентно B".

Операции импликации и эквиваленции задаются следующей таблицей истинности:

A	B	$A \Rightarrow B$	$A \Leftrightarrow B$
0	0	1	1
0	1	1	0
1	0	0	0
1	1	1	1

Справедливы следующие равенства:

1	$\overline{\overline{A}} = A$	
2	$\overline{0} = 1$	$\overline{1} = 0$
3	$A + A = A$	$A \cdot A = A$
4	$A + 0 = A$	$A \cdot 1 = A$
5	$A + 1 = 1$	$A \cdot 0 = 0$
6	$A + \overline{A} = 1$	$A \cdot \overline{A} = 0$
7	$A + B = B + A$	$A \cdot B = B \cdot A$
8	$(A + B) + C = A + (B + C)$	$A \cdot (B + C) = AB + AC$
9	$(A \cdot B) \cdot C = A \cdot (B \cdot C)$	$A + BC = (A + B)(A + C)$

Равенства 7, 8, 9 называют соответственно законами коммутативности, ассоциативности и дистрибутивности.

Справедливы также равенства, называемые законами поглощения и законами де Моргана.

Законы де Моргана	$\overline{A + B} = \overline{A} \cdot \overline{B}$; $\overline{A \cdot B} = \overline{A} + \overline{B}$; $\overline{\overline{A} + \overline{B}} = A \cdot B$; $\overline{\overline{A} \cdot \overline{B}} = A + B$
Законы поглощения	$A + A \cdot B = A$; $A \cdot (A + B) = A$

Для доказательства равенства $\overline{A + B} = \overline{A} \cdot \overline{B}$ построим таблицу истинности для логических выражений левой и правой частей равенства.

A	B	$A+B$	$\overline{A+B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

$\overline{A + B} = \overline{A} \cdot \overline{B}$

$\overline{A}$	$\overline{B}$	$\overline{A} \cdot \overline{B}$
1	1	1
1	0	0
0	1	0
0	0	0

Важное замечание: при преобразовании логических выражений, содержащих операцию импликации, удобно использовать равенство

$$(A \Rightarrow B) = \overline{A} + B$$

Его полезно доказать.

2. Разбор задач

2.1. Таблицы истинности для заданных логических функций

***Задача* 1.** Составить таблицу истинности для функции

$$f(A,B,C) = \overline{A + C} + BC + (A \Rightarrow B).$$

Решение

Логические переменные A, B, C принимают значения 0 или 1 независимо друг от друга. Используя определения операций, вычисляем значение логического выражения для всех возможных комбинаций значений переменных. Количество комбинаций N зависит от числа логических переменных, определяющих функцию. Для одной переменной $N=2$; для двух переменных $N=2^2=4$; для трех переменных $N=2^3=8$; для k переменных $N=2^k$. Число комбинаций определяет число строк в таблице истинности. Маловероятно, что на экзаменах может быть предложена задача рассматриваемого типа, содержащая выражение с числом переменных, больших трех.

A	B	C	$\overline{A+C}$	BC	$A \Rightarrow B$	$\overline{A+C} + BC + (A \Rightarrow B)$
0	0	0	1	0	1	1
0	0	1	0	0	1	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	0	0	1	1
1	1	1	0	1	1	1

Окончание на с. 12

Логические задачи на вступительных экзаменах по информатике

Окончание. Начало на с. 11

Задача 2. (Логические операции над целыми числами — побитовые операции, поразрядные операции.) Вычислить значения функции $f(X,Y,Z)=\overline{X+Z}+YZ+(X\Rightarrow Y)$ и представить результат в виде числа в десятичной системе счисления, если переменные заданы в десятичной системе счисления и $X=7_{10}$; $Y=25_{10}$; $Z=100_{10}$.

Решение

Преобразуем вначале данные числа из десятичной системы счисления в двоичную. Получим:

$X=7_{10}=111_2$; $Y=25_{10}=11001_2$; $Z=100_{10}=1100100_2$.

Используем один и тот же формат из 8 бит для их представления, причем биты перенумерованы от 0 до 7 (младший разряд — 0, старший — 7). Получим $X=00000111_2$; $Y=00011001_2$; $Z=01100100_2$.

Далее, построим таблицу значений функции при заданных значениях аргументов в битах с одинаковым номером и полученные значения функции в каждом из восьми битов будем трактовать как двоичное число 11111001_2.

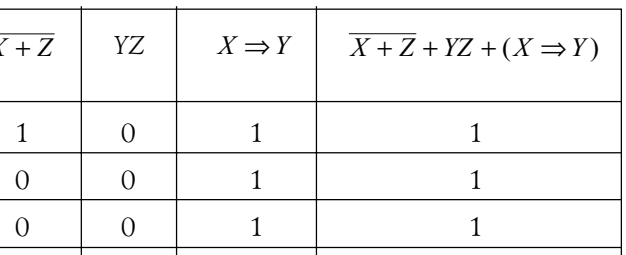
№ бита	X	Y	Z	$\overline{X+Z}$	YZ	$X\Rightarrow Y$	$\overline{X+Z}+YZ+(X\Rightarrow Y)$
7	0	0	0	1	0	1	1
6	0	0	1	0	0	1	1
5	0	0	1	0	0	1	1
4	0	1	0	1	0	1	1
3	0	1	0	1	0	1	1
2	1	0	1	0	0	0	0
1	1	0	0	0	0	0	0
0	1	1	0	0	0	1	1

$\overline{X+Z}+YZ+(X\Rightarrow Y)=11111001_2=249_{10}$.

2.2. Задачи с множествами. Основы теории множеств

Множество — совокупность некоторых объектов. Примеры: множество учащихся одного выпуска, множество неотрицательных чисел, множество точек на прямой. Будем обозначать множества прописными буквами латинского алфавита A, B, C, ..., а элементы множества — строчными. Запись $x\in E$ означает: "x является элементом множества E". Если множество A состоит из элементов, принадлежащих другому множеству E, то оно называется *подмножеством*, или частью этого множества. Запись $A\subset E$ читается: "A подмножество E".

Частями некоторого множества E являются: оно само; его пустая часть — пустое подмножество $\varnothing$; подмножества, состоящие из одного элемента; подмножества, состоящие из двух элементов, и т.д. Из множества E можно образовать новое множество — множество его частей (множество всех подмножеств данного множества).



Определены три основных операции над множествами:

- Объединение (сумма) множеств A и B — это множество, состоящее из всех элементов, принадлежащих либо множеству A, либо B. Обозначение: $A\cup B$ $(A+B)$.
- Пересечение (произведение) множеств A и B — это множество, состоящее из всех общих элементов двух множеств A и B. Обозначение: $A\cap B$ $(A\cdot B)$.
- Дополнение к A в E. Обозначение $\overline{A}$ $(E\setminus A,\ E-A)$ — это множество тех элементов E, которые не принадлежат A.

Пример. Пусть рассматривается множество чисел $E=\{1, 2, 3, 4, 5, 6, 7, 8\}$. Его подмножествами будут: пустое множество $\varnothing$; само множество E; одноэлементные подмножества $\{1\}$, $\{2\}$, ..., $\{8\}$; двухэлементные подмножества $\{1, 2\}$, $\{1, 3\}$, ..., $\{2, 8\}$, $\{2, 3\}$, ..., $\{2, 8\}$, $\{3, 4\}$, ..., $\{7, 8\}$; трехэлементные подмножества $\{1, 2, 3\}$, ..., $\{6, 7, 8\}$, ...; семиэлементные подмножества $\{1, 2, 3, 4, 5, 6, 7\}$, ..., $\{2, 3, 4, 5, 6, 7, 8\}$.

Положим, что каждому элементу множества E поставлен в соответствие один определенный бит со значением **1**, так, как показано в таблице:

$E=$	{1,	2,	3,	4,	5,	6,	7,	8}
№ бита	0	1	2	3	4	5	6	7
Представление множества E	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

Подмножества рассматриваемого множества E будем представлять с помощью восьми битов, причем значение бита равно 0, если соответствующий элемент E не принадлежит подмножеству.

Очевидно, что пустое подмножество $\varnothing$ будет представлено нулями в каждом бите:

0	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Подмножества $A=\{2,6\}$, $B=\{1,4,5,8\}$ и $C=\{3,4,7\}$ будут представлены с помощью набора из нулей и единиц следующим образом:

№ бита	0	1	2	3	4	5	6	7
$A=$	{2,						6}	
Представление подмножества A	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**
$B=$	{1,			4,	5,			8}
Представление подмножества B	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**
$C=$	{	2,					6}	
Представление подмножества C	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**

Общее число подмножеств данного множества из 8 элементов, представляемых восьмибитовыми наборами из 0 и 1, равно $2^8=256$.

В случае, когда количество элементов множества равно n, число подмножеств этого множества равно 2^n.

Логические операции и операции над подмножествами некоторого конечного множества E становятся эквивалентными, если каждому элементу множества поставить в соответствие бит в n-разрядном наборе, где n — число элементов E. Операции над множествами производятся как поразрядные логические операции.

Получим дополнение $\overline{A}$, сумму $A+B$ и произведение $A\cdot B$ подмножеств E с помощью логических операций:

№ бита	0	1	2	3	4	5	6	7	
Представление подмножества A	0	1	0	0	0	1	0	0	$A=\{2,6\}$
Представление подмножества B	1	0	0	1	1	0	0	1	$B=\{1,4,5,8\}$
Представление подмножества $\overline{A}$	1	0	1	1	1	0	1	1	$\overline{A}=\{1,3,4,5,7,8\}$
Представление подмножества $A+B$	1	1	0	1	1	1	0	1	$A+B=\{1,2,4,5,6,8\}$
Представление подмножества AB	0	0	0	0	0	0	0	0	$AB=\varnothing$

В п. 1 перечислены основные законы алгебры логики. Эти законы применимы и к множествам, но в законах 2, 4, 5 и 6, приведенных ранее, логические операции следует заменить на соответствующие операции над множествами:

2	4	5	6
$\overline{\varnothing}=E$	$A+\varnothing=A$	$A+E=E$	$A+\overline{A}=E$
$\overline{E}=\varnothing$	$A\cdot E=A$	$A\cdot\varnothing=\varnothing$	$A\cdot\overline{A}=\varnothing$

Сведение задачи над множествами к задаче над логическими переменными

Задача 3. Пусть даны три подмножества $A=\{2, 4, 6\}$, $B=\{1, 4, 5, 8\}$ и $C=\{3, 4, 7\}$ множества $E=\{1, 2, 3, 4, 5, 6, 7, 8\}$. Найти подмножество, определяемое соотношением $\overline{A+B\cdot\overline{C}}$.

Решение

Сначала подмножества A, B и C представляются в виде восьмибитовых наборов из нулей и единиц так, как это было показано, далее, используя аналогию между логическими операциями и операциями над множествами, строится следующая таблица:

№ бита	A	B	C	$\overline{C}$	$B\cdot\overline{C}$	$A+B\cdot\overline{C}$	$\overline{A+B\cdot\overline{C}}$
0	0	1	0	1	1	1	0
1	1	0	0	1	0	1	0
2	0	0	1	0	0	0	1
3	1	1	1	0	0	1	0
4	0	1	0	1	1	1	0
5	1	0	0	1	0	1	0
6	0	0	1	0	0	0	1
7	0	1	0	1	1	1	0

Окончательный результат $\overline{A+B\cdot\overline{C}}=00100010=\{3, 7\}$.

Продолжение в следующем номере

МОДЕРНИЗАЦИЯ КОМПЬЮТЕРНОГО КЛАССА

Установка Microsoft Office

В.М. НЕЧАЕВ

Продолжение. См. № 48/97; 2, 10, 14, 18, 20, 33, 40, 47/98; 10/99

Как было объявлено ранее, это — последняя статья данного цикла. Модернизацию своего класса, пройдя этапы наращивания памяти, подключения CD-ROM, монтажа и настройки локальной сети, инсталляции системы и уплотнения дисков, я завершаю установкой пакета прикладных программ Microsoft Office 97. Производится она с дистрибутивного лазерного диска, а поскольку устройство для его чтения у меня на весь класс одно, то обращаться к нему надо будет, естественно, по сети. Напомню, что у меня в классе 12 машин, и с каждой из них можно иметь доступ к той, которую я называю учительской. Называю по традиции, так как в одноранговой сети без выделенного сервера (а только такую можно иметь на платформе Windows 95) она представляет собой обычное рабочее место, лишь аппаратно чуть богаче оснащенное, чем другие.

Первым делом, конечно, надо на этой самой учительской машине открыть доступ к собственно устройству CD-ROM: правой кнопкой мыши щелкнуть по значку диска (у меня он обозначен буквой F) и, выбрав в контекстном меню пункт **Доступ**, поставить галочку в строке **Общий ресурс**[1]. Под значком появится изображение руки, наглядно отображая, что теперь данный ресурс может быть использован и с других рабочих мест.

Дальнейший порядок действий таков:

Пуск ⇒ Настройка ⇒ Панель управления ⇒ Установка и удаление программ

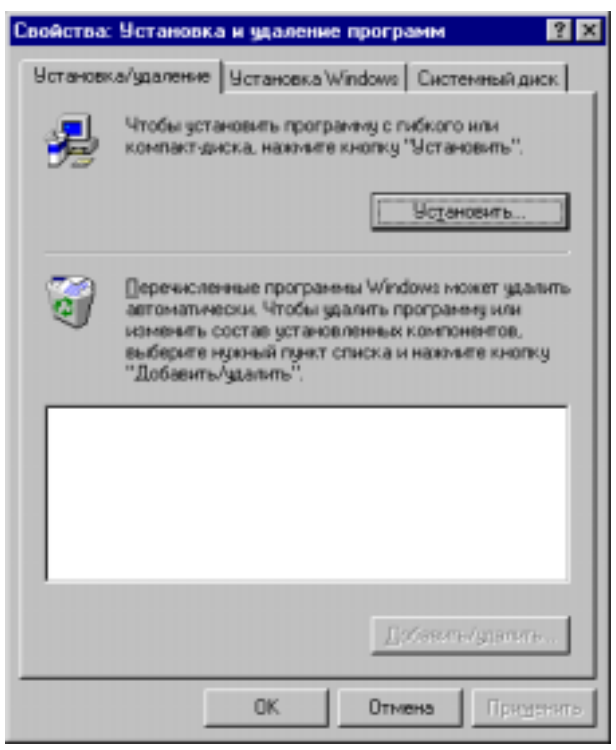


Рис. 1. В первый раз устанавливаю

В диалоговом окне, раскрытом на закладке **Установка/удаление**, пока что пусто — еще, кроме самой системы, ничего не устанавливалось. И вот теперь пришла пора щелкнуть по экранной кнопке **Установить...** с выразительным многоточием, означающим, как всегда, что потребуются кое-какие дополнительные пояснения: что именно установить, откуда, куда, в каком составе. После щелчка система сразу же проявит свою готовность сотрудничать, я бы даже сказал, слишком уж поспешную. А именно, будут проверены гибкие дисководы A и B, нет ли там каких-либо дискет с файлами вроде SETUP или INSTALL. А поскольку дисководы в этот момент вообще пустые, то они лишь зря подергаются и все. На учительском же компьютере, помимо этого, будет просмотрен еще и привод CD-ROM, но обычно на лазерном диске присутствует не один какой-нибудь программный продукт, а несколько, и поэтому их установочные программы находятся не в корневом каталоге, а в отдельных для каждого продукта папках. Таким образом, в корневом каталоге система тоже ничего не найдет и мы придем в итоге к тому, с чего бы и надо было начинать, — к обзору.

На ученическом рабочем месте привода компакт-дисков вообще не имеется, поэтому из полного списка надо выбрать **Сетевое окружение**, а в нем учительский компьютер (у меня он назван WS01). Среди его доступных ресурсов должен быть и CD-ROM в виде папки F, войдя в которую останется только выбрать какой-то конкретный продукт.

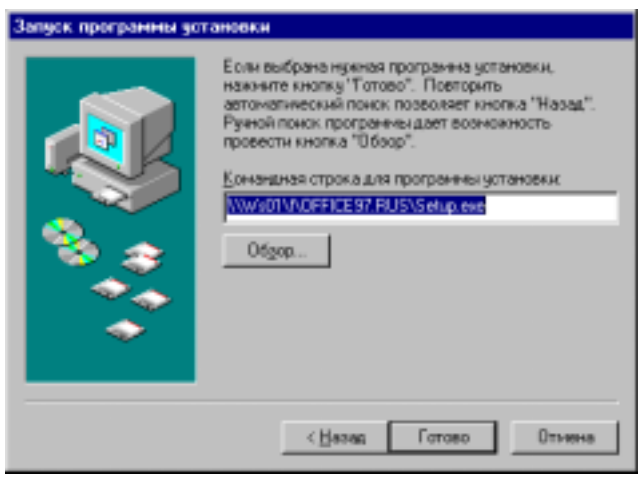


Рис. 2. Путь к источнику через сеть

Мне сейчас нужен пакет Office 97, русская версия. Вот его папка и вот его инсталляционная программа. Выбор сделан. После нажатия кнопки **Готово** и небольшого, с которым я соглашаюсь, ожидания появляется лицензионное соглашение. Затем идут обычные формальности: указание вашей фамилии, вашей организации и, самое главное, 11-значного ключа компакт-диска (CD Key). Этот ключ, т.е. код, должен быть напечатан на коробке или содержаться в текстовом файле на самом диске (обычно он имеет имя README.TXT, или иногда SERIAL.TXT).

После ввода ключа начнется поиск установленных компонентов — пакет, как известно, очень большой, а поиск этот нужен, по-видимому, не только затем, чтобы сократить часть работы, если какие-то части уже установлены, но и для того, чтобы не менять их существующих настроек. Правда, казалось бы, какой же в поиске смысл, если я вообще первый раз имею дело с Microsoft Office? Однако это ведь только мне известно, что в первый, а программа-то установки узнает об этом вот как раз в результате данных своих действий.

Далее мне предлагается принять решение относительно того, куда будет устанавливаться пакет, в какую папку: либо я сам должен указать путь к ней, либо взять то, что предложено. Обычно это C:\Program Files\Microsoft Office, и ничего против этого я не имею.

Тогда мне задается следующий вопрос: каков будет вид установки — стандартный, минимальный или по выбору? Минимальный мне не подходит, потому что тогда бы пришлось все время при работе с Office держать компакт-диск в дисководе. Хоть винчестер у меня и маленький, но все же не настолько, чтобы согласиться на такое неудобство. Ну а вариант "как обычно" я тоже не люблю, принципиально, так что щелкаю по кнопке **Выборочная** на экране и начинаю выбирать.

Между прочим, вот сейчас и станет ясно, что понималось под "обычным набором".

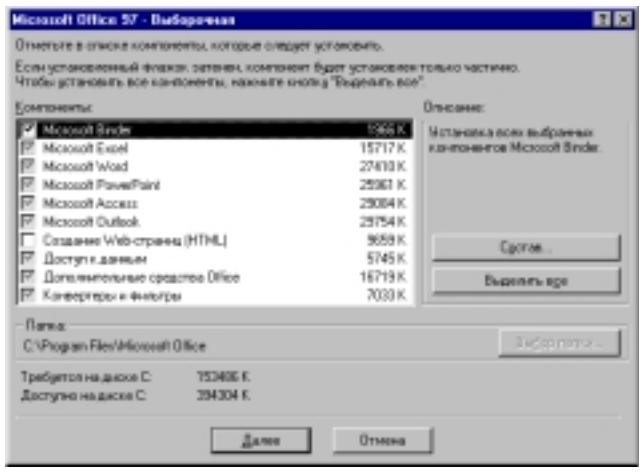


Рис. 3. Этот состав предлагался изначально

Большинство галочек — неяркие, и это следует понимать так, что каждый компонент имеет внутри себя еще некоторые разделы и часть из них помечена к установке, а часть — нет. Полностью заказан только первый компонент — Microsoft Binder, то ли потому, что он очень важен, то ли по той причине, что, являясь самым маленьким по объему, чуть меньше двух мегабайт, уже внутреннего деления не имеет и может быть или взят целиком, или совсем не взят. Все это легко проверить — стоит лишь щелкнуть по кнопке **Состав**.... И, кстати, уже сам факт того, что она доступна, свидетельствует не в пользу второго предположения. Видимо, Binder все же считается важным компонентом пакета Microsoft Office.

Однако мало ли кем и что считается, у меня может быть свое собственное мнение на этот счет, и, скажу сразу, оно отлично от мнения создателей пакета. Упомянутая программа предназначена для работы с так называемыми подшивками, т.е. с наборами разнородных офисных документов. Набор, наверное, создает определенное удобство при использовании как единого целого нескольких отдельных файлов с текстами Word, таблицами и диаграмм Excel, презентациями Power Point, относящимися к одной задаче, к одному какому-то проекту, но я воздержусь пока что от данной службы и галочку все же сниму. Сразу и суммарный счетчик требуемого дискового пространства уменьшит свои показания на соответствующее количество байт — нечего винчестер зря заполнять, и так уже полторы сотни мегабайт начислено.

А вот для инструмента, требуемого при создании web-страниц, я, наоборот, места не пожалею. Его окошко, где могла бы стоять галочка, не то что неяркое, а вовсе пустое; по каким соображениям — трудно сказать. Возможно, галочка отсутствует из-за того, что в операционной системе еще не установлен необходимый для работы в Интернете сетевой протокол TCP/IP (я ведь пока не настраивал компьютеры на такую работу, но в дальнейшем обязательно собираюсь это сделать). Впрочем, какая разница, почему этот пункт не был предложен, главное, что я его могу сам выбрать. Причем как раз вот у этого компонента внутреннее деление отсутствует, его можно брать только целиком. Соответственно счетчик байтов укажет новое число:

161179 = 153486 (начальное) — 1966 (отказ от Binder) + 9659 (добавление Web).

Вторым пунктом общего списка идет Excel; надо только посмотреть, в каком составе.

Рис. 4. Подразделы Excel

[1] Какой тип доступа — полный или только на чтение, — очевидно, не имеет (раз это CD-ROM) никакого значения.

Продолжение на с. 14

Установка Microsoft Office

Продолжение. Начало на с. 13

С самой программы галочку снять нельзя. То есть формально можно, но тогда меня предупредят[2] о бессмысленности дальнейшего выбора составляющих данного компонента. Если я действительно не хочу устанавливать Excel, то надо было бы убрать галочку ранее.

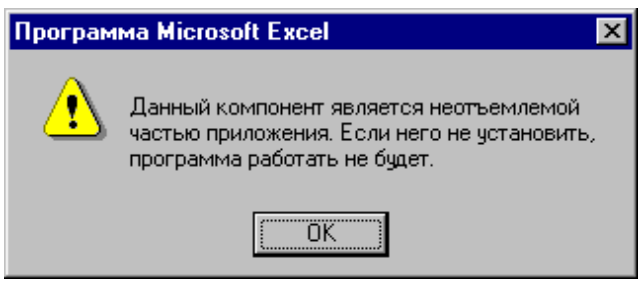


Рис. 5. Ну, ошибся

Часто бывает у неопытных пользователей, что, получив подобное сообщение об ошибочных действиях, они теряются и боятся что-либо делать дальше, невольно ожидая наказания. На самом деле, щелкнув **OK**, я просто признаю свою неправоту и получу возможность галочку восстановить.

Что же касается **Справки и файлов примеров**, то внутри этого раздела есть свои подразделы, поскольку кнопка **Состав**... становится активной, когда выделяется данная строка. Подразделов три: справка по Excel, справка по Visual Basic и файлы примеров. Каждый подраздел, в свою очередь, можно отметить галочкой, причем независимо от других. Но, на мой взгляд, это нерационально — занимать драгоценное дисковое пространство справками, которые вряд ли понадобятся ученикам, когда рядом учитель, или в крайнем случае учебные пособия. Безжалостно снимаю эти галочки, высвобождая почти десять мегабайт! Файлы примеров, правда, можно оставить, займет это всего-навсего 0,2 Мб[3].

Так я буду поступать вообще со всеми справками, и по Word, и Power Point, и по всем другим приложениям. Если у меня имеются кое-какие сомнения на этот счет, то единственно, пожалуй, в отношении Excel. Дело в том, что там имеется множество встроенных функций — математических, логических и т.д., и вот по ним-то как раз удобно бы иметь подсказку под рукой. Впрочем, год-другой работы, и все нужные функции уже помнишь почти наизусть.

Значит, оставив только одну галочку для примеров, я должен щелкнуть по кнопке **OK**, не боясь, что тем самым уже запущу процесс установки. Нет, конечно. Просто я вернусь на предыдущий уровень, чтобы просмотреть другие разделы Excel.

Третью строчку Microsoft Map — анализ и отображение данных на географических картах — я считаю излишеством, хотя и не навязываю свою точку зрения. Иногда, наверное, это кто-то и использует. Надо бы, конечно, и самому попробовать, но... как-нибудь потом, когда будет время, а сейчас пока не до этого. Еще пять мегабайт долой.

Далее, что за шаблоны такие? В **Составе** можно увидеть, что это бланки-заготовки (весьма искусно, кстати, выполненные) распространенных в хозяйственной деятельности документов: авансовый отчет, счет, заказ. Все это неплохо иметь, да и по объему не так уж значительно.

Теперь надстройки. Что там? Оказывается, девять строчек, так много, что уж не все и умещаются в окошке и приходится пользоваться прокруткой.

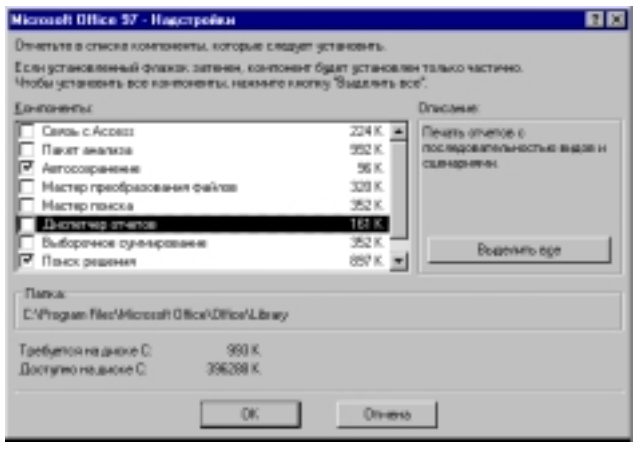


Рис. 6. Какие надстройки полезные

Тут задумаешься: какие нужны, а какие нет. Дело даже не только в экономии места, а в принципе: как-то глупо себя чувствуешь, когда из опасения — а вдруг, мол, без этого что-то в целом будет работать неправильно — заказываешь лишнее. Мне кажется, что лучше следовать такому принципу: брать только то, что тебе, точно знаешь, потребуется. Если что нужное упустил, то это станет ясно в процессе работы, и тогда — пожалуйста, можешь вернуться к этому вопросу и добавить необходимый компонент. Зато уж будешь понимать его предназначение. А так, когда берешь все подряд, то никогда и не узнаешь, что к чему.

На первых порах можно ограничиться указанными двумя надстройками: автосохранением и поиском решения. Впрочем, может, и автосохранение не стоит заказывать. Это когда я сам работаю над каким-нибудь важным документом, то на случай, если вдруг питание в сети пропадет, нелишним бывает включить данный режим, с периодом повтора минут пятнадцать. А вот во время урока в классе появление запросов на сохранение файла вместо пользы скорее внесет хоть и небольшой, но все-таки беспорядок. Я, например, пойду средним путем: данную надстройку закажу, чтобы она всегда была наготове, но активизирую ее[4] только когда появляется реальная в ней потребность.

Вторая надстройка мною выделена потому, что я активно ее использую на своих занятиях. Поиск решения — очень интересная вещь и не очень сложная; в двух словах расскажу о ней. Когда спрашиваешь у того или иного преподавателя, для чего предназначены электронные таблицы, то обычно в ответ слышишь — для расчетов. Это, конечно, верно, но очень уж как-то так, вообще. А я бы выделил три линии. Во-первых, для более гибкого, чем в Word, форматирования текста, когда его надо расположить в клеточках где-то одинакового, где-то разного размера, в одних — по горизонтали, в других — по вертикали или даже по диагонали; при этом еще и разнообразные рамки провести, а не просто сетку, и т.д. Во-вторых, и это, безусловно, главное, действительно для расчетов. Но еще и в-третьих — для численного моделирования. Допустим, я уже связал нужными формулами те клетки, в которых расположены исходные данные, и по ним вычисляется результат. Это обычное дело. Но вот, бывает, задача ставится так, что, наоборот, требуется найти такие значения этих самых исходных данных, которые приводили бы к заранее заказанному результату. Причем это может быть просто подбор одного из параметров (так сказать, подгонка), а может и комплексное манипулирование всей их совокупностью, да еще с соблюдением целого ряда устанавливаемых ограничений. Вот эти обратные задачи с вариациями переменных и обслуживает надстройка **Поиск решения**.

Наконец, последняя строка в перечне разделов — **Конвертеры электронных таблиц**; это мне совсем не нужно, поскольку никакими другими электронными таблицами я не пользуюсь. Таким образом, в итоге по всему приложению Excel получается 11,4 Мб. Причем я заметил, что один и тот же состав компонентов может иметь немного отличающийся объем на разных машинах, более того, даже на одинаковых машинах, но в зависимости от того, уплотнен ли диск, на который производится установка, или нет. Различие небольшое, и упомянул я о нем лишь для того, чтобы оно не было неожиданным (хотя все же интересно, каково происхождение этого различия).

То же самое теперь надо проделать по всем другим приложениям. Так относительно Word я решил:

- что отдавать три мегабайта на справку — это излишняя роскошь;
- что из мастеров и шаблонов от одного, самого большого (с макросами), можно отказаться;
- что проверку правописания желательно иметь в полном объеме;
- что с адресной книгой и с WordMail можно пока что обождать до лучших времен;
- и что "конвертер" текстовых файлов заказать вроде бы надо.

В сумме по Word все это дало 23,7 Мб.

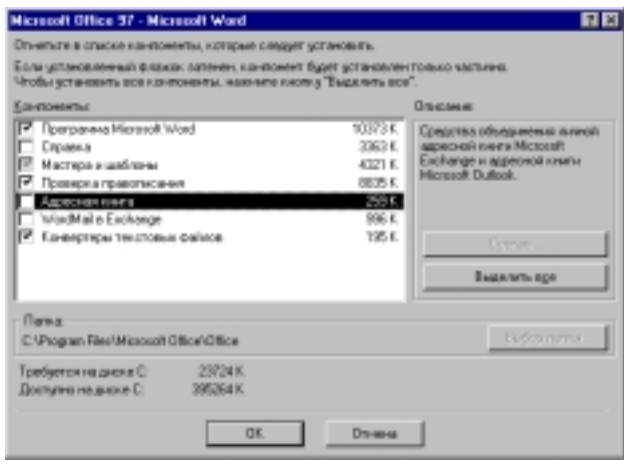


Рис. 7. Word пообъемнее, чем Excel

Аналогичная картина получается с третьим приложением. Сама программа Power Point, естественно, нужна; все шаблоны — и содержания (как основные, так и дополнительные), и особенно дизайна — тоже. Справка опять же, я считаю, ни к чему, как и трансляторы к другим презентационным программам, а вот эффекты озвучивания очень даже пригодятся (разумеется, если есть звуковая плата). Значит, считай, еще почти 20 Мб плюс.

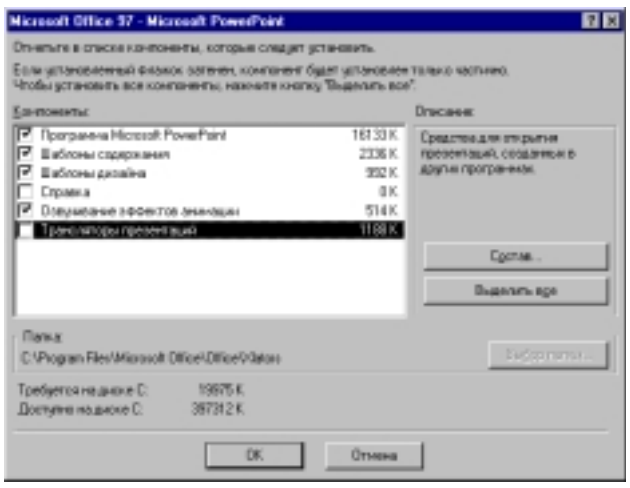


Рис. 8. Power Point тоже нужен

[2] Забавно, что с орфографической ошибкой.

[3] Хотя, может быть, примеры действуют только внутри справок?

[4] Это относится и ко всем другим надстройкам в Excel: даже если они установлены, то в работу они вступят только после моего особого приглашения, т.е. явного вызова, который осуществляется через меню **Сервис⇒Надстройки**.

Состав Access можно сильно облегчить, сэкономив не только, как обычно, на справке, но еще и на дополнительных мастерах. Кроме того, из учебных баз данных, которых всего три, я оставляю только одну, которая попроще, по имени "Борей". И все равно в итоге набирается больше, чем у всех других, — 22 Мб.

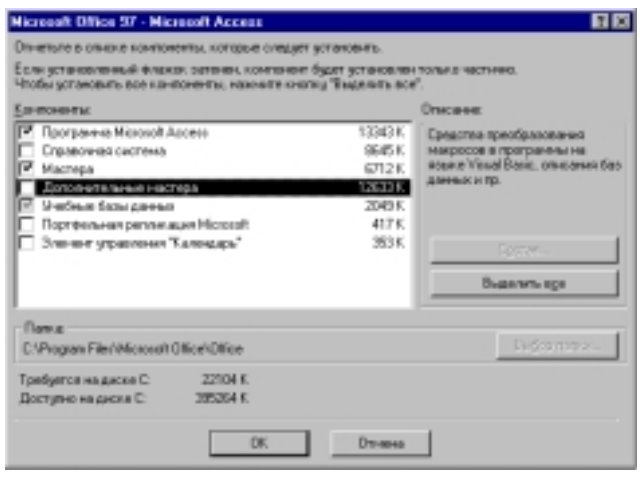


Рис. 9. Access – тяжеловес

Продолжаю выбирать, вернувшись к самому началу. Microsoft Outlook, может, и хорошая программа, да вот я только-только овладел в какой-то мере ее предшественницей — Microsoft Exchange, и что же мне теперь, опять перестраиваться? Нет, обожду немного; снимаю галочку.

Про создание web-страниц я уже сказал, что беру, а по поводу доступа к данным мне так и осталось неясным, обязательно его заказывать или можно опустить. Решаю твердо следовать правилу "пусть практика покажет" и тоже снимаю галочку.

Дело движется к завершению. Остались **Дополнительные средства Office** и опять какие-то конвертеры. Из дополнительных средств, а их всего насчитывается четырнадцать, я выбираю только шесть: к тем, что видны на рисунке, добавляя лишь проверку орфографии. Не знаю, кому как, а мне лично панель Microsoft Office и помощник по Office только мешают, занимая место на экране и немного раздражая навязчивостью. Хотя, конечно, помощник очень забавный, особенно если включены звуковые эффекты. Всевозможных диаграмм и в Excel достаточно, для редактора фотографий у меня сканера нет, а о системе я и так все знаю. Одним словом, десять мегабайт и довольно.

Что касается конвертеров, то их предлагается два типа — для текстовых файлов 15 штук и для графических — 18. И если с первыми я еще проявил коекакую самостоятельность (убрал из заказа конвертеры Excel, WordPerfect, Lotus, Works и восстановления текста, а добавил конвертер HTML), то по поводу графики не решился на большие изменения (отказался только от явно ненужного импорта файлов WordPerfect и Macintosh). И все равно получилось пять с лишним мегабайт.

Что же получилось в итоге? Первоначально предложенный состав пакета объемом в 153 Мб уменьшился почти на треть, в основном за счет справочной системы.

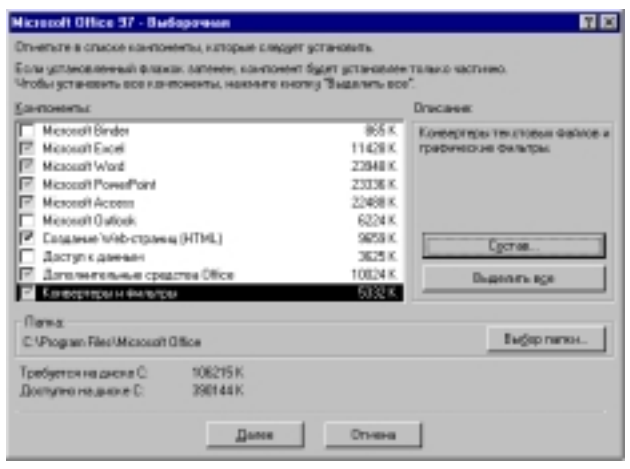


Рис. 10. Мой окончательный выбор

Ничего не поделаешь, как не модернизируй класс, а новых больших винчестеров ждать неоткуда. Да и не так уж она важна, справочная система. Хуже будет, если какие-либо операции не смогут быть выполнены во время работы с программами пакета. Впрочем, и это не беда. Надо только выяснить, какого компонента не хватает, и доустановить его.

Собственно процесс доустановки совсем простой. Ставлю диск, в **Панели управления** снова вызываю **Установку программ**, только щелкаю не по кнопке **Установить**..., а сначала выделяю строчку **Microsoft Office 97, профессиональный выпуск** в списке уже установленных продуктов и заказываю режим **Добавить/удалить**.... Дальнейшие действия — те же самые: поставить галочку, снять галочку. А вот выяснить, какой конкретно компонент тебе нужен, чтобы правильно выполнялась та или иная операция, а какой тут совершенно ни при чем, или, наоборот, присутствие каких компонентов необязательно — в этом-то главная сложность!

* * *

А теперь – о новом цикле

Его тема — применение локальной сети Windows 95 в учебном процессе. Это будут снова, как и в предыдущей серии, десять статей, примерно на год, по следующим вопросам:

Первый раздел — "Администрирование сети"

Поскольку класс уже следует в значительной степени рассматривать как единый организм, с доступом к общим ресурсам, с потенциальной возможностью взаимодействия, а не просто как комплект отдельных, автономно работающих машин, то и подход к нему становится иным. С одной стороны, повышается значение дисциплины использования рабочих мест, поддерживать которую должен, очевидно, учитель. А с другой стороны, сеть дает ему, как администратору, дополнительные и довольно-таки мощные средства управления учебным процессом.

Статья 1. "Пользовательские профили". Речь здесь пойдет о том, какую роль играет указание имени пользователя при запуске компьютера. Ее, эту роль, можно вообще свести на нет во избежание лишних хлопот для учителя — если он, например, в настоящее время так загружен, что ему уже не до того, чтобы разбираться с сетью. А можно и, наоборот, предельно возвысить в стремлении следовать принципу — разделяй и властвуй!

Статья 2. "Системная политика". Если учитель пойдет именно этим, вторым путем, то средствами самой операционной системы можно сделать так, чтобы со своего учительского места он мог бы жестко задавать определенные правила использования компьютеров для всех рабочих мест учеников. Редактор системных правил — это программа, входящая в дистрибутив Windows 95, как раз и позволяющая администратору диктовать свою волю.

Раздел "Электронная почта для рабочей группы"

Даже если в классе нет модема для связи с внешним компьютерным миром, локальная сеть позволяет проводить занятия по овладению навыками работы с электронной почтой, причем именно практические, а не только в теории. Более того, можно наладить реально действующую систему письменного общения учеников разных классов, как между собой, так и с преподавателями. Внешне все будет выглядеть как самая настоящая электронная почта, единственно только, что все адресаты — местные и нет возможности обращаться к ним извне.

Статья 3. "Создание почтового отделения". Какие программы потребуется установить, как это самое почтовое отделение создать, как им пользоваться и как им управлять.

Статья 4. "Подготовка и пересылка почтовых сообщений". Письмо, помимо собственно текста, должно еще иметь адресата и кое-какие служебные пометки. Кроме того, к нему можно приложить посылку — например, файл с рисунком, или звукозапись, или, например, программу. Хотелось бы также узнать, дошло ли письмо?

Раздел "Интранет – это Интернет в пределах класса"

Статья 5. "Установка и настройка протоколов TCP/IP".

Статья 6. "Использование гиперссылок в документах".

Статья 7. "Web-сервер локальной сети".

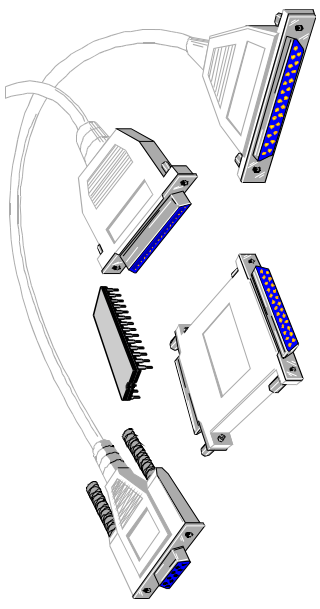
Раздел "Настоящий Интернет в классе"

Статья 8. "Подключение к провайдеру через модем".

Статья 9. "Прокси-сервер, его назначение и использование".

Статья 10. "Что делать в Интернете на уроках".

В связи с проектом по бесплатному подключению московских школ к сети Интернет тематика последних статей цикла приобрела особую актуальность. Но более подробно описать их содержание я сейчас затрудняюсь, поскольку сам еще нахожусь в процессе. Как все это непросто!



Что там inside?

Заметки о новом процессоре Pentium-III

Окончание. Начало на с. 1

Новые 70 команд, по уверению Intel, уже поддержаны программным обеспечением для разработчиков, и они на самом деле способны увеличить скорость обработки графической и особенно звуковой информации на 50—80%.

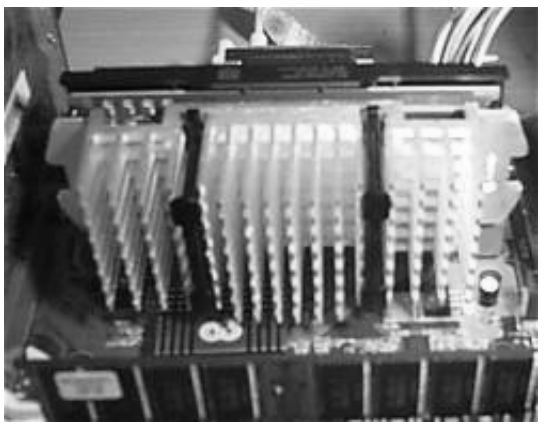
Итак, фирма Intel в условиях жесткой конкурентной гонки просто не могла себе позволить четырехгодичную паузу, как это бывало раньше, на протяжении которой просто повышалась бы тактовая частота процессора. Но и назвать новый процессор Pentium-II — SSE, что наиболее правильно отражает суть дела, было немыслимо.

Во-первых, после MMX это могли счесть откровенной издевкой, а во-вторых, конкуренты будут вынуждены говорить не о SSE-поддержке, а о совместимости с Pentium-III, что, разумеется, играет на престиж.

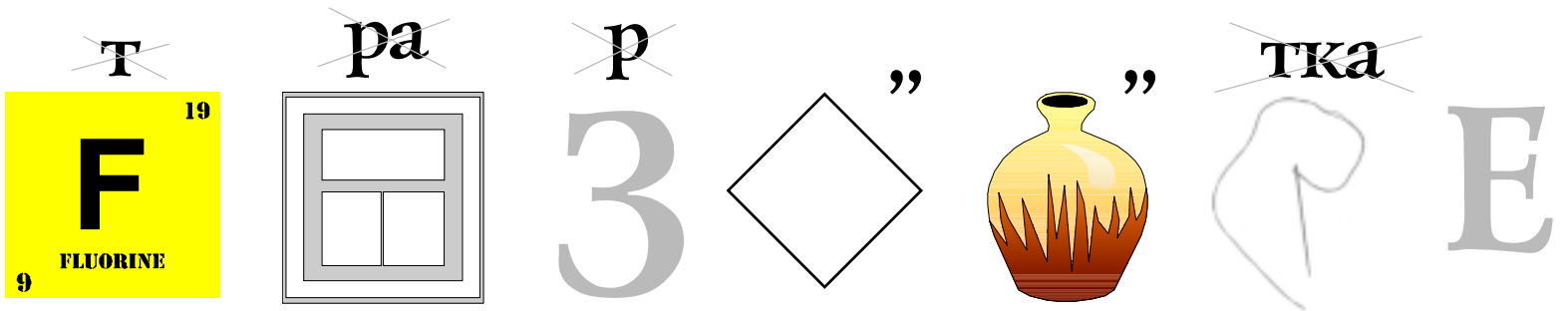
На первой странице приведены диаграммы из статьи И.Гавриченкова (**iXBT.stack.net**), сравнивающие производительность различных процессоров (разогнанных, как это и водится на Руси, выше проектной частоты) в офисных приложениях и в играх, не оптимизированных под SSE.

Глядя на них, довольно легко подвести итог. Intel Pentium-III 500 — это более быстрый (на 50—100 МГц) Pentium-II с дополнительными командами, которые не используются пока что ни в одном приложении.

Возможно, в ближайшие 2—3 года эти новые команды и станут использоваться в большинстве вновь разрабатываемых и модернизируемых программ и это приведет-таки к обещанному 80%-ному росту производительности. Но отдавать за эту гипотетическую возможность прямо сейчас $500—700 довольно бессмысленно. Поскольку даже если эта возможность и будет с блеском реализована, к тому времени и Pentium-III резко подешевеет, и, глядишь, Pentium-VI выйдет (сейчас он носит кодовое название Coppermine).



А.И. СЕНОКОСОВ



Форматирование

Терминал

Ребусы подготовлены В.Г. Федориновым

ВНЕКЛАССНАЯ РАБОТА ПО ИНФОРМАТИКЕ

ОБЪЕДИНЕНИЕ ПЕДАГОГИЧЕСКИХ ИЗДАНИЙ “ПЕРВОЕ СЕНТЯБРЯ”

Первое сентября
А.С. Соловейчик
индекс подписки — 32024

Английский язык
Е.В. Громушкина
индекс подписки — 32025

Биология
Н.Г. Иванова
индекс подписки — 32026

Воскресная школа
монах Киприан (Ященко)
индекс подписки — 32742

География
О.Н. Коротова
индекс подписки — 32027

Здоровье детей
А.У. Лекманов
индекс подписки — 32033

Информатика
Е.Б. Докшицкая
индекс подписки — 32291

Искусство
Н.Х. Исмаилова
индекс подписки — 32584

История
А.Ю. Головатенко
индекс подписки — 32028

Литература
Г.Г. Красухин
индекс подписки — 32029

Математика
И.Л. Соловейчик
индекс подписки — 32030

Начальная школа
М.В. Соловейчик
индекс подписки — 32031

Немецкий язык
Gerolf Demmel
индекс подписки — 32292

Русский язык
Л.А. Гончар
индекс подписки — 32383

Спорт в школе
Н.В. Школьникова
индекс подписки — 32384

Управление школой
Н.А. Широкова
индекс подписки — 32652

Физика
Н.Д. Козлова
индекс подписки — 32032

Химия
О.Г. Блохина
индекс подписки — 32034

Школьный психолог
М.Н. Сартан
индекс подписки — 32898

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

ИНДЕКС ПОДПИСКИ
для индивидуальных подписчиков 32291
для предприятий и организаций 32591
комплекта приложений 32744

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

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

Отпечатано с готовых диапозитивов редакции в ОАО ПО «Пресса-1», 125865, ГСП, Москва, ул. Правды, 24.

Тираж 7000 экз.
Заказ №

тел./факс (095)249 3138, факс (095)249 3184, тел. 249 3386