УДК 681.327

Т. А. Блинова *, **В.Н. Порев** **, кандидаты техн. наук
* Национальный технический университет Украины «КПИ» (Украина, 03056, Киев, пр-т Победы, 37, ФИОТ, корп. 18, тел.: (044) 4549338),
** Ин-т землеупорядочения и информационных технологий (Украина, 03113, Киев, ул. Дружковская, 8, кафедра ГИС, тел.: (044) 4835223, E-mail: kafedra_gis@ukr.net)

Некоторые способы кодирования растров в геоинформационных системах

(Статью представил канд. техн. наук В. В. Аристов)

Рассмотрены вопросы повышения компрессии при кодировании 256-цветных палитровых изображений. Учтена необходимость прямого доступа к растровым данным для геоинформационных систем. Даны описание и оценка эффективности разработанных способов кодирования.

Розглянуто питання підвищення компресії при кодуванні 256-кольорових палітрових зображень. Враховано необхідність прямого доступу до растрових даних для геоінформаційних систем. Дано опис і оцінку ефективності розроблених способів кодування.

К л ю ч е в ы е с л о в а: компрессия без потерь, палитровые изображения, метод RLE, прямой доступ, геоинформационные системы.

При разработке и использовании программного обеспечения геоинформационных систем (ГИС) возникают проблемы, связанные с растровыми изображениями. Формат кодирования 256-цветных палитровых изображений используется, например, для электронных карт. В ГИС требуется читать произвольный фрагмент растра и масштабировать изображения. Для этого необходимо организовать прямой доступ к отдельным элементам растра в файле или группе файлов, что должно сочетаться с эффективной компрессией без потерь. Наиболее часто самый быстрый прямой доступ оказывается у несжатых растров. Попытаемся гармонизировать компрессию и прямой доступ и сформулировать предложения, которые могут помочь разработчикам ГИС в отдельных случаях повысить компрессию при сохранении приемлемой скорости прямого доступа к растрам.

Краткий обзор методов. Для прямого доступа удобен формат TIFF, а также его расширение для ГИС — GeoTIFF. В этом формате можно гибко распределять растр, записывая его отдельными частями в едином файле,

что позволяет организовать быстрый прямой доступ. Формат поддерживает множество методов компрессии.

Для масштабирования и быстрого прямого доступа применяется формат MrSID [1], однако он ориентирован на компрессию с потерями непалитровых изображений. Одним из примеров решения проблемы быстрого доступа к разнообразным растровым данным является технология мозаик, реализованная в ГИС «ОКО» [5].

Среди методов компрессии без потерь, используемых для 256-цветных палитровых растров, наиболее популярны LZ-подобные [2] словарные методы. В первую очередь это LZW [3] и Deflate [4], воплощенные в форматах GIF, PNG и TIFF, которые обеспечивают высокую степень компрессии [5—7]. Их недостатки — относительно невысокая скорость декодирования и большие затраты памяти для словаря. Методы компрессии без потерь на основе кодирования длин повторов (Run Length Encoding — RLE) используются в форматах PCX, TGA и др. [5, 6]. Их достоинство — высокая скорость декодирования, недостаток — малая компрессия.

Следует упомянуть также методы компрессии без потерь на основе преобразования Барроуза—Уиллера [6, 8]. Эти методы могут обеспечить высокую компрессию, но требуют значительных вычислительных ресурсов. Они реализованы в некоторых архиваторах. Использование таких методов в графических форматах файлов авторам статьи не известно.

Основная проблема при чтении сжатых растров в режиме прямого доступа состоит в следующем: для того чтобы прочитать нужный пиксел, необходимо декодировать все предыдущие закодированные пикселы. При этом выполняется много лишних вычислений, что снижает скорость работы. Это характерно для таких режимов прямого доступа, как прореживание пикселов при показе растра в уменьшенном виде или скользящее окно при показе части растра (скроллинг).

Один из подходов к решению этой проблемы — независимое кодирование отдельных небольших блоков растра и соответствующая поддержка в файловом формате. Можно было бы предположить, что в этом случае наиболее подходящим является метод JPEG, ориентированный на блоки растра, однако он не приспособлен для сжатия палитровых изображений без потерь [9]. Можно использовать формат GIF, в котором заложена возможность чересстрочного кодирования, но, учитывая специфику алгоритма LZW, это не повышает скорость прямого доступа в режиме скользящего окна, а при показе с уменьшением эффективное масштабирование ограничено фиксированной чересстрочностью. Подобных ограничений лишен формат TIFF.

Выберем независимое кодирование отдельных строк. Можно заметить, что в стандартных форматах на основе метода RLE растр обычно так и кодируется. Делается это не для прямого доступа, а по той причине, что

сплошное непрерывное RLE-кодирование всего растра не имеет смысла, так как степень компрессии от этого практически не увеличивается. Иная картина наблюдается при использовании LZ-подобных методов. Даже межстрочное кодирование всего растра уменьшает степень компрессии, а независимое кодирование отдельных строк может значительно уменьшить степень компрессии и сделать ее меньшей, чем при использовании простейших вариантов RLE. Преимущества словарных методов проявляются тогда, когда в формировании словаря участвуют все пикселы и их комбинации.

Учитывая изложенное, для облегчения прямого доступа было выбрано независимое кодирование строк методом RLE. Проведем некоторые усовершенствования метода RLE, состоящие в повышении компрессии и, по возможности, сохранении его положительных характеристик.

Введем обозначения: M — число битов двоичного кода, необходимое для кодирования цвета, $M = \log_2$ (число цветов); следует учесть, что в некотором изображении могут использоваться не все цвета из 256-цветной палитры, тогда $M < 8$; $C1$ — число битов, необходимое для кодирования индексов множества главных цветов.

Если сделать сортировку всех цветов палитры согласно частоте их использования, то M-битный код 00..00 означает номер наиболее часто используемого цвета, код 00..01 — номер следующего по популярности и т. д. Будем считать, что кодер и декодер обрабатывают отдельно множество из 2^{C1} главных цветов. Тогда $C1 < M$. Если из M битов выделить $C1$ младших битов, то это и будет индекс главного цвета. Например, если $C1 = 2$, то это означает четыре главных цвета, индексы которых кодируются как 00, 01, 10 и 11 в порядке уменьшения частот использования соответствующих цветов.

Способ кодирования 1. Используем три разновидности кодовых последовательностей:

0c...c (всего M битов) — для одиночных пикселов, у которых старший бит цвета равен 0. Одиночные пикселы цветов 1c...c будут кодироваться как цепочка длины 1. Для кодирования цепочек пикселов предусмотрим такие последовательности:

10n...nc...c — сначала префикс (биты 10), потом $N1$ битов кода длины цепочки (биты n). Завершают последовательность $C1$ битов c, которые означают индекс главного цвета;

11n...nc...c — сначала префикс (биты 11), потом $N2$ битов длины цепочки и далее M битов цвета.

Примечание. Подразумевается, что кодирование одиночного пиксела главного цвета может быть представлено и в виде цепочки длины 1 в случае, когда соответствующий код короче M.

Свойства такого способа кодирования определяются параметрами $C1$, $N1$ и $N2$. Если поставить задачу достижения наименьшего числа битов для

кодированных строк (или иных блоков растра), то оказывается, что различным строкам будут соответствовать разные значения параметров. Можно находить некоторые оптимально усредненные значения этих параметров, например, путем статистического анализа результатов кодирования многих изображений разных типов, в результате зафиксировав конкретные числовые значения параметров $C1$, $N1$, $N2$, и полученные значения рекомендовать для внедрения в некотором файловом формате или в протоколе обмена.

Однако более целесообразным представляется иной путь, а именно: возложить все исследования оптимальных параметров на кодер—устройство (или программу), анализирующее растр в ходе записи его в файл. При кодировании отдельными строками кодер находит оптимальные значения $C1$, $N1$ и $N2$ индивидуально для каждой строки конкретного растра. Если продолжать далее такое кодирование, то можно получить кодер, находящий для каждой строки не только оптимальные параметры одного способа кодирования, но и использующий несколько различных способов, выбирая из них тот, который даст минимум битов для кодирования текущей строки.

Способ кодирования 2. Используем кодовые последовательности двух типов:

0c...c — для одиночных пикселов произвольного цвета, сначала префикс (0), а затем M битов цвета;

1c...cnn...n — для цепочек пикселов главного цвета, сначала префикс (1), затем $C1$ битов индекса главного цвета (так можно закодировать только 2^{C1} наиболее популярных цветов). Завершают кодовую последовательность биты длины цепочки (nn...n). Каждому i-му главному цвету соответствует определенное число битов $n - Ni$.

Пример кодирования цепочек главных цветов для $C1 = 2$:

100nnn — $N0 = 3$ для наиболее популярного цвета (индекс 00);

101nnnnnnn — $N1 = 7$ для главного цвета с индексом 01;

101nnnn — $N2 = 4$ для главного цвета с индексом 10;

111nnnnn — $N3 = 5$ для главного цвета с индексом 11.

Способ кодирования 2 задается множеством параметров $C1$ и Ni ($i = 0 \div \div 2^{C1} - 1$). Выбор оптимальных значений этих параметров осуществляет кодер.

Одно из отличий второго способа кодирования от первого заключается в том, что происходит кодирование одиночных пикселов не более чем $(M + 1)$-разрядным кодом, независимо от цвета. Так учитывается вероятность большого числа одиночных пикселов второстепенных цветов.

Следует заметить, что способ 2 учитывает не только преобладание пикселов некоторых цветов над другими, но и то, что один из главных цветов может быть представлен преимущественно короткими цепочками пикселов, а другой — длинными. В отличие от традиционного RLE-кодирования цепочек, когда сначала идет длина цепочки, а затем цвет, в кодовых последовательностях способа 2 сначала идет цвет, а затем длина

цепочки. Это нужно для того, чтобы декодер был способен правильно расшифровать код длины цепочки.

В способе 2 число начальных битов (1c...c) одинаково для кодов цепочек всех главных цветов. Вариантом способа 2 является следующий способ, в котором цвета кодируются префиксами разного размера, причем длина префикса зависит от частоты использования цвета. Его можно рассматривать как сочетание RLE и метода Хаффмана [10].

Способ кодирования 3. Используются такие кодовые последовательности:

0c...c — для одиночных пикселов произвольного цвета, сначала префикс (0), а затем *M* битов цвета;

1p...pnn...n — для цепочек пикселов, сначала префикс (1p...p), потом биты длины цепочки (nn...n). Как и в способе 2, каждому *i*-му главному цвету соответствует определенное число (*Ni*) битов n.

Кодирование префиксными кодами способом 3 рассмотрим на таком примере:

0c...c — одиночные пикселы;

10nn...n — цепочка цвета 0 (*N*0 битов n);

110nn...n — цепочка цвета 1 (*N*1 битов n);

1110nn...n — цепочка цвета 2 (*N*2 битов n);

1111nn...n —цепочка цвета 3 (*N*3 битов n).

В данном примере длина кода цепочки цвета 0 меньше, чем для способа 2, однако для цветов 2 и 3 коды цепочек длиннее. Параметры способа кодирования 3: число главных цветов и множество значений *NI*.

При кодировании некоторых строк растра могут возникнуть такие проблемы. Допустим, строка содержит много пикселов главного цвета в виде цепочек разной длины. Например, есть много цепочек длины 4 и цепочек длины 100 одного цвета. Какое число битов (n...n) длины цепочки принять в качестве оптимального? Если взять два бита, то цепочки длины 100 должны кодироваться как 25 кодов цепочек длины 4. А если выбрать семь битов, то для коротких цепочек — это избыточный код. Можно принять в качестве оптимального некоторое среднее значение, но при этом появятся лишние биты для коротких цепочек и одновременно будет происходить многократное кодирование длинных. Частично эту проблему решает способ 1, в котором для цепочек главного цвета могут быть использованы либо *N*1 либо *N*2 битов длины. Однако в некоторых случаях лучше использовать следующий способ кодирования.

Способ кодирования 4. Используем кодовые последовательности:

0c...c — для одиночных пикселов произвольного цвета, сначала префикс (0), а затем *M* битов цвета;

1c...cxx...x — для цепочек пикселов главного цвета, сначала префикс (1), затем *C*1 битов для индекса главного цвета. Завершают кодовую

последовательность биты длины цепочки (xx...x). Каждому значению индекса главного цвета (биты c...c) соответствует формат кода длины цепочки, который выбирается из трех форматов: *а*, *б*, *в*.

Формат *а*:

1c...cnn...n — $N1$ битов n длины цепочки ($N1 = 0 \div 15$).

Формат *б*:

1c...c0nn...n— $N1$ битов n ($N1 = 0 \div 15$);

1c...c1nn...n — $N2$ битов n ($N2 = N1+1 \div N1+16$).

Формат *в*:

1c...c0nn...n — $N1$ битов n ($N1 = 0 \div 7$);

1c...c10nn...n — $N2$ битов n ($N2 = N1+1 \div N1+8$);

1c...c11nn...n — $N3$ битов n ($N3 = N2+1 \div N2+8$).

Выбор оптимальных значений параметров $C1$, $N1$, $N2$, $N3$ и формата кода длины для каждой строки растра возлагается на кодер.

Следуя существующим традициям по компрессии информации, набор рассмотренных выше способов 1—4 назовем RLE-БП.

На основе способов 1— 4 были разработаны и другие вариации кодирования длин повторов. Данные способы кодирования и их вариации заложены в файловый формат, для которого были созданы адаптивный кодер и декодер. Файловый формат был сначала использован в преподавательской деятельности, а затем и в мозаиках для ГИС.

Результаты анализа способов кодирования. Было исследовано несколько тысяч файлов, обработанных реальной ГИС, которая использует кодер для компрессии растровых изображений типа чертежей, планов, карт и др. Созданная программа-анализатор может отображать параметры способов кодирования, использованных для каждого файла, а также статистику для групп файлов.

Для каждой строки кодер выбирает способ кодирования, дающий наименьшее число битов кода. Существуют растры, для которых доминирует один способ. Однако есть изображения, для оптимального кодирования которых использованы все способы (1—4). Оказалось, что по внешнему виду изображения почти невозможно спрогнозировать, какой из способов кодер изберет оптимальным. Поэтому здесь не приведены конкретные образцы изображений.

Аналогичная ситуация наблюдается на уровне параметров кодов. Даже для тех изображений, для которых доминирует один способ, разные строки закодированы с различными параметрами кодов. Однако теоретически могут существовать изображения, все строки которых лучше всего кодировать одним способом с одинаковыми параметрами кодов (тривиальный случай — изображение вертикальных полос).

Приведем примеры отдельных параметров способов кодирования. Рассмотрим способ 1, для которого есть три параметра: $C1$, $N1$ и $N2$. Для

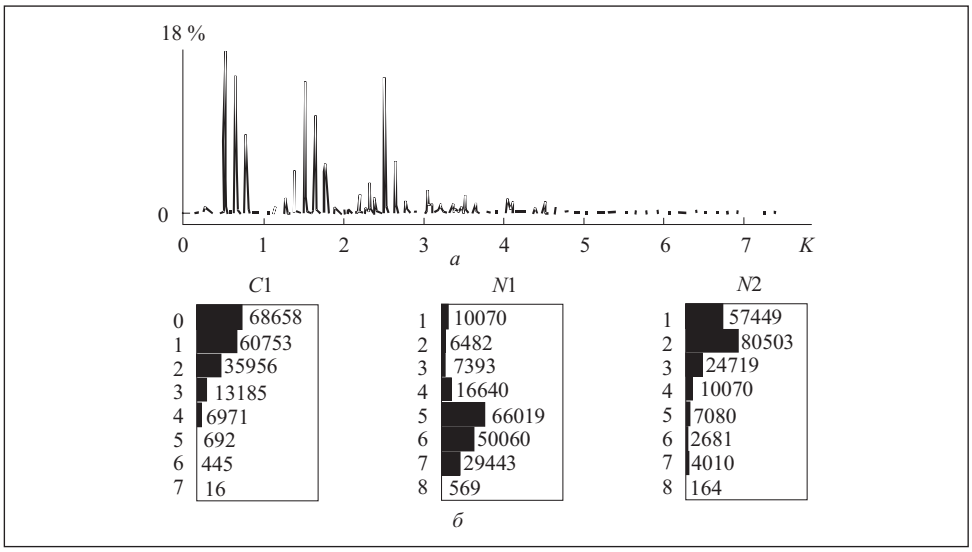

Рис. 1. Гистограмма параметров *C*1, *N*1 и *N*2 (*а*) и распределение их значений (*б*) для первой группы изображений (166 файлов)

анализа ограничим диапазон значений *C*1, *N*1 и *N*2 такими числами: *C*1 от 0 до 7, *N*1 от 1 до 8 и *N*2 от 1 до 8. Результаты анализа кодирования группы файлов покажем в виде гистограммы распределения значений параметров. Поскольку параметров три, то гистограмма должна быть четырехмерной. Чтобы нагляднее представить результаты, преобразуем гистограмму в двумерную. Запишем тройку параметров (*C*1, *N*1, *N*2) в виде двоичного числа $K = ccc.nnnppp$, где *ccc* — биты *C*1, далее — точка дробной части, затем *nnn* — биты *N*1, *ppp* — биты *N*2. Таким образом, 512 комбинаций значений параметров *C*1, *N*1 и *N*2 представлены числами от нуля до семи.

На гистограмме, показанной на рис 1, *а*, столбики означают процент от общего числа строк, закодированных способом 1. Как видим, самый используемый (18 %) код (*C*1, *N*1, *N*2) = (0, 5, 2), с ним конкурируют коды (0,6,2), (1,5,2), и (2,5,1). В абсолютном выражении значения *C*1, *N*1 и *N*2 распределились, как показано на рис. 1, *б*.

Если анализировать распределение значений параметров в одном изображении из данной группы, то оно может быть, например, таким, как показано на рис. 2, *а*. Здесь самый используемый (21 %) код (3, 2, 6). Для этого изображения распределение значений *C*1, *N*1 и *N*2 в абсолютном выражении показано на рис. 2, *б*. Как видим, если фиксировать некоторую тройку значений (*C*1, *N*1, *N*2) как оптимальную и пытаться кодировать с данными параметрами все изображения некоторой группы подобных изображений, то это может привести к снижению компрессии для каждого конкретного образца даже из этой группы.

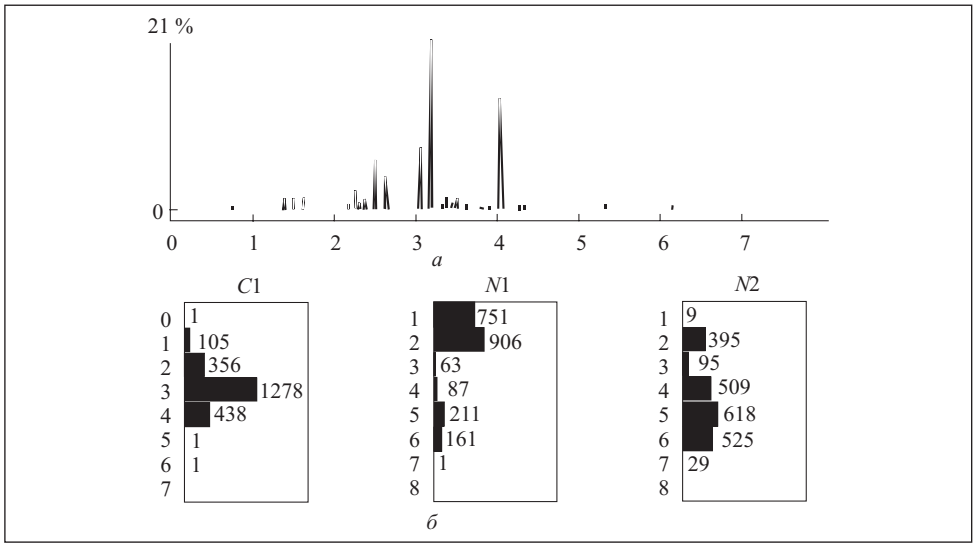

Рис. 2. Гистограмма параметров *C*1, *N*1 и *N*2 (*а*) и распределение их значений (*б*) для одного изображения из первой группы

Поскольку изложить здесь в полном объеме результаты проведенного анализа предложенных способов кодирования затруднительно, приведем только вывод: искать фиксированные оптимальные значения параметров кодирования практически бессмысленно. Индивидуально для каждой строки растра это должен делать кодер. При разработке кодера достаточно учесть общий диапазон значений параметров кодов.

Приведем сравнительные данные о степени компрессии *K* для файлов, представленных в различных форматах. Способ компрессии RLE-БП был реализован в формате GGF4. За единицу принято отсутствие компрессии (формат BMP). Степень компрессии оценивалась отношением размера файлов BMP к размеру файлов иных форматов. Для каждой из двух групп файлов (упомянутой выше первой и второй, состоящей из двух сотен файлов) находили минимальную $K_{\min}$, максимальную $K_{\max}$ и среднюю $K_{\text{ср}}$ степени компрессии (см. таблицу). Средняя степень компрессии для группы файлов равна отношению суммарного объема файлов BMP к суммарному объему файлов исследуемого формата.

Как видно из таблицы, RLE-БП лучше, чем традиционные методы RLE (в форматах PCX и TGA). При сравнении предлагаемого способа со словарными LZ-подобными методами получены приблизительно одинаковые результаты. Так, у 50 % исследованных изображений размеры файлов, представленных в формате GGF4, меньше, чем размеры соответствующих файлов GIF. По степени компрессии формат GGF4 немного уступает формату PNG.

Формат файла	Метод компрессии	Группа файлов					
		I			II		
		$K_{1,ср}$	$K_{1,min}$	$K_{1,max}$	$K_{2,ср}$	$K_{2,min}$	$K_{2,max}$
BMP	—	1	1	1	1	1	1
GIF	LZW	4,9	2,6	27,4	3,4	1,7	7,5
PCX	RLE	3,1	1,5	10,5	2,4	1,4	4,7
PNG	Deflate	5,3	2,9	32,3	3,7	2	7,5
TGA	RLE	3,3	1,4	14	2,4	1,5	4,6
GGF4	RLE-БП	4,7	2,4	23,8	3,3	1,8	6,4

Рассматривая скоростные характеристики RLE-БП, можно заметить асимметрию: время кодирования существенно (5—10 раз) превышает время декодирования. Это зависит от числа вариантов кода и их параметров, анализируемых адаптивным кодером при обработке конкретного изображения. (При работе с растровыми данными в ГИС чаще приходится декодировать файлы, кодирование выполняется один раз при записи файла).

Способы кодирования, используемые в RLE-БП, неравнозначны по вычислительной сложности. Каждому конкретному изображению соответствует определенный набор способов кодирования. Поэтому скорость декодирования для различных изображений может быть неодинакова. С одной стороны, декодирование RLE-БП сложнее, чем RLE (для файлов PCX и TGA), с другой стороны, кодовыми последовательностями RLE-БП могут быть представлены более длинные цепочки пикселов. Многое зависит от реализации методов компрессии, однако этот вопрос выходит за рамки данной статьи. Здесь ограничимся следующим наблюдением. При работе соответствующих программных средств на персональных компьютерах нынешнего поколения, время полного декодирования файлов, сжатых методом RLE-БП, на 5—20 % больше, чем при использовании метода RLE PCX, и на 20—30 % меньше, чем при использовании LZW GIF.

В формате файлов GGF4 предусмотрена поддержка режима прямого доступа. Это означает, что в этом режиме оценки скорости могут существенно отличаться от приведенных выше. Прямой доступ подразумевает быстрый переход от начала файла до заданного блока данных, минуя распаковку ненужных данных. Так, например, в формате PCX не заложена поддержка прямого доступа, поэтому просмотр содержимого всего растра размерами 10000×10000 пикселов на экране монитора может потребовать в десятки раз большего времени, чем просмотр того же растра в формате GGF4, несмотря на то что скорость декодирования PCX может быть выше.

Так же и показ отдельных фрагментов в скользящем окне может быть в десятки раз быстрее в формате GGF4 по сравнению с PCX (это зависит от положения фрагмента относительно начальной точки растра).

Кодирование RLE-БП по степени компрессии сравнимо с LZW GIF и позволяет организовать прямой доступ, что проблематично для LZ-подобных методов.

Таким образом, предложенные четыре способа кодирования, совокупность которых названа RLE-БП, позволяют повысить степень компрессии метода RLE. Кодирование RLE-БП можно рекомендовать для электронных растровых карт, применяемых в геоинформационных системах.

The problems of compression increase when encoding the 256-coloured palette images are considered. The necessity of direct access to raster data for the geographic information systems is taken into account. The description and efficiency estimate for developed encoding models are presented.

1. http://www.lizardtech.com.
2. *Ziv J., Lempel A.* A Universal Algorithm for Sequential Data Compression //IEEE Transactions on Information Theory. — 1977.
3. *Welch T.* A Technique for High-Performance Data Compression //Computer. — June. —1984.
4. *Deutch P.* DEFLATE Compressed Data Format Specification version 1.3.// RFC 1951, Aladdin Enterprises, May, 1996.
5. *Блинова Т. А., Порев В. Н.* Компьютерная графика. — Киев : Юниор, 2005. — 520 с.
6. *Ватолин Д., Ратушняк А., Смирнов М., Юкин В.* Методы сжатия данных. — М. : — Диалог-МИФИ, 2002. — 384 с.
7. *PNG* (Portable Network Graphics) Specification Version 1.0. Massachusetts Institute of Technology. — 1996.
8. *Burrows M., Wheeler D. J.* A block-sorting lossless data compression algorithm. Technical Report 124, Digital SRC, Palo Alto, 1994.
9. *Wallace G. K.* The JPEG still picture compression standard //Communication of ACM.— 1991. — Vol. 34, № 4.
10. *Huffman D. A.* A method for the construction of minimum redundancy codes //Proc. of IRE.— 1952. — Vol. 40. — P. 1098—1101.

Поступила 27.10.06;
после доработки 05.04.07

БЛИНОВА Татьяна Александровна, канд. техн. наук, ст. науч. сотр. каф. вычислительной техники Национального технического университета Украины «КПИ». В 1979 г. окончила Киевский политехнический ин-т. Область научных исследований — высокопроизводительные параллельные вычислительные системы, кодирование информации.

ПОРЕВ Виктор Николаевич, канд. техн. наук, доцент Ин-та землеупорядочения и информационных технологий. В 1981 г. окончил Киевский политехнический ин-т. Область научных исследований — компьютерная графика, геоинформационные системы.