Информация в цифровом виде. Кодирования

- Представление информации
- Правила кодирования
- Избыточность в кодировании
- Аналого-цифровое кодирование
- Измерение количества информации в битах
- Простейший цифровой процессор
- Работа с программой TextProcessor

Представление информации

Как мы знаем, каждый исполнитель имеет свой язык, т.е. набор инструкций, которые он способен правильно исполнять. Если исполнитель наталкивается на инструкцию не из его языка – он должен остановиться и выдать сообщение об ошибке. Таким образом, что для одного исполнителя алгоритм может быть ошибочным, для другого – нет.

Означает ли это, что все алгоритмы для данного исполнителя придется писать на одном и том же языке – языке? Напомним, что это не обязательно. Действовать при этом можно по примеру того, как это давно придумано при общении людей на различных языках. А именно, если некто, скажем, знает английский, но не знает французский – он может «понимать» французские слова через французско-английский словарь.

Так и в алгоритмах – исполнитель может использовать инструкции другого, «не своего» языка, но в связке со специальным алгоритмом-переводчиком с этого языка на язык исполнителя. При этом сам алгоритм-переводчик должен быть написан уже на языке исполнителя – иначе исполнитель не сможет алгоритм использовать. Такого рода алгоритмы-«посредники» называются трансляторами (то есть переводчиками – но по-английски).

Самый простой транслятор переводит просто обозначения одних и тех же данных. Такие обозначения называют *кодами* (данных). Одна какая-то система обозначений называется *кодированием*. Для одного и того же набора объектов может быть много различных кодирований.

Почему применяются различные кодирования для одних и тех же объектов, а не обходятся каким-то одним кодированием? Потому что по каким-то причинам одни и те же данные в разных местах гораздо удобнее представлять по-разному. Простой известный пример – обозначения (кодирование) дробей в математике. Для одной и той же дроби есть обозначения ? и 0,25. Используют и те, и другие. Вторые проще сравнивать – не надо производить никаких умножений. Кроме того, их можно использовать «в бесконечном варианте» для представления любых действительных чисел. А первые – «обыкновенные» дроби удобнее в непосредственном представлении результатов деления.

Другой пример – использование азбуки Морзе при телеграфных переговорах. Когда телеграф появился (полтора столетия назад), его надо было использовать для передачи сообщений. Но передавать можно было только простейший электрический сигнал. Поэтому была придумана специальная кодировка букв, удобная для «шифрования», передачи и «дешифрования» специально обученным человеком-телеграфистом.


См фильм «Азбука Морзе»

Если кодируются по-разному наборы одинаковых данных, то коды одинаковых данных можно сопоставить. Такое сопоставление кодов называется кодированием или раскодированием. Кодирование и раскодирование – это процесс перевода одних кодов в

другие. Обычно какое-то представление данных является основным, перевод именно из него в другое представление называется кодированием, а обратный перевод – раскодированием. Например, когда мы нажимаем на клавиши компьютерной клавиатуры, внутри нее происходит кодирование и передача символов, на которые мы нажимаем. Для нас привычные символы на клавишах являются основными, происходит их кодирование. Этот процесс называется входным кодированием. Другие входные кодирования происходят при движении мыши, при движении пера по графической таблетке и так далее.

Наоборот, при выводе на экран компьютерные коды символов, поступающие на графическое устройство, раскодируются в изображения привычных нам символов на экране. Это – выходное кодирование (или раскодирование).

Разумеется, внутри компьютера в происходят многие другие дополнительные кодирования и раскодирования данных (перевод данных из одной формы в другую).

Отметим еще, что раскодирование зачастую называют еще декодированием.

Сам процесс кодирования и раскодирования может быть устроен чрезвычайно просто: через таблицы соответствия кодов. Разумеется, эти таблицы должны быть максимально удобны для применения

Правила кодирования

Для кодировки символов внутри компьютера, конечно, не применяется код Морзе. Поскольку компьютер работает с числами (точнее, опять таки их кодировками), применяются числовые коды для символов – числовые кодирования. Числа при этом применяются двоичные, восьмеричные (группируя двоичные разряды по 4) и шестнадцатеричные (группируя двоичные разряды по 8). Отметим, что Морзе является двоичной кодировкой алфавита.

Ныне в компьютерных программах применяется целый ряд различных кодирований одних и тех же данных, что связано как удобством их использования в различных ситуациях, так и с историческими причинами. Сначала компьютерные программы были проще и для них удобнее были одни кодирования, затем удобнее стали другие, а более старые остались использоваться вместе с более старыми программами.

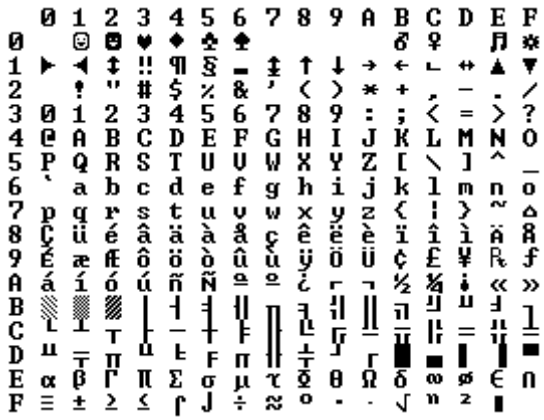
Например, код Морзе по мере развития и автоматизации телеграфных аппаратов в первой половине 20-го века уже стал заменяться другими системами кодов, которые были более удобны. Так появилась система кодировки символов, у которой код каждого символа имел 7 двоичных разрядов. Это было удобно, так как одинаковая длина каждого кода упрощала конструкцию принимающего аппарата. А семь двоичных разрядов позволяли закодировать 128 различных символов, что по тем временам было весьма много.

Эта система, называемая ASCII, перекочевала и в компьютерные программы, причем применялась достаточно долго.

Важно заметить, что при кодировании необходимо различать каждую существенную

особенность того объекта, который кодируется. Например, при кодировании алфавита можно не различать строчные и заглавные буквы. Но тогда при декодировании их также невозможно отличить. Поэтому при кодировании именно алфавитов коды для «больших» и «маленьких» букв разные – а это уже добавка более 30 кодов. Коды были присвоены и другим символа – например, знакам параграфа, доллара, стрелочкам и т.д.

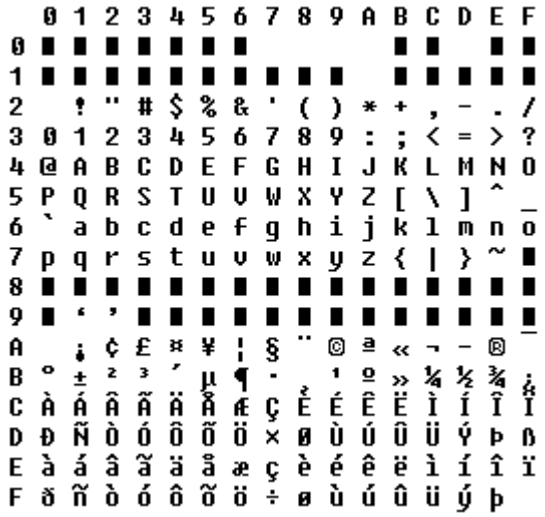
Следующая широко распространившаяся система кодировки применялась в DOS - первой стандартной операционной системе персональных компьютеров. Для нее взяли систему ASCII и к ее кодам добавили еще один разряд – разумеется, слева. Тем самым число кодов в новой системе кодирования DOS достигло 256, а число разрядов – удобного числа 8. При этом первые 128 кодов просто перекочевали в DOS из ASCII. Вот таблица кодировок DOS (кодирования шестнадцатеричные: слева – первая цифра, сверху – вторая цифра шестнадцатеричного кода):



Верхние строки таблицы с забавными символами в таблице кодировок DOS возникли из-за того, что в «телеграфной» таблице там стояли коды, управляющие работой именно телеграфных аппаратов. В компьютерах такие коды надо было использовать как-то по-другому – отсюда такой результат.

Нижняя половина таблицы кодировок DOS, как видим, состоит в добавочных символах алфавита (для некоторых европейских языков), символов, удобных для «сборки» рамок таблиц, символов для операций и т.д.

В дальнейшем, при смене основной операционной системы на Windows, таблицу DOS изменили на более удобную Win. При этом заменили опять-таки «дополнительные» 128 знаков, кроме того, часть кодов оставили неиспользуемыми. Причина исчезновения в кодировках Win многих специальных символов из числа кодируемых в DOS – другой принцип построения изображений в Windows. Если в DOS изображения «собирались из кусочков», то в Windows – собирались из точек. Поэтому в Windows коды «кусочков» из кодировок DOS оказались не нужны – и были убраны из кодовой таблицы.



Поскольку основной язык в американских компьютерных программах английский, для букв русского алфавита в их кодировках места не предусматривалось (как, впрочем, и для всех остальных). Из-за этого при кодировании букв русского алфавита приходится использовать часть кодов других символов – тех, которые используются очень редко. Использование разных кодов под русские буквы привело к появлению разных систем кодировок (поскольку эти кодировки проще всего представить в виде таблиц, они и называются кодовыми таблицами). То же самое произошло и с кодировками для других языков – для них имеются свои кодовые таблицы.

Например, из английской системы кодировки DOS возникла «русская» система кодировки, называемая еще DOS Cyrillic или IBM 866. Из кодовой таблицы Win получилась таблица, называемая Win1251. В ней, по сравнению с Win, заменены коды следующих символов (номера кодов даются десятичные):

192 А	200 И	208 Р	216 Ш	224 а	232 и	240 р	248 ш
193 Б	201 Й	209 С	217 Щ	225 б	233 й	241 с	249 щ
194 В	202 К	210 Т	218 Ъ	226 в	234 к	242 т	250 ъ
195 Г	203 Л	211 У	219 Ы	227 г	235 л	243 у	251 ы
196 Д	204 М	212 Ф	220 Ь	228 д	236 м	244 ф	252 ь
197 Е	205 Н	213 Х	221 Э	229 е	237 н	245 х	253 э
198 Ж	206 О	214 Ц	222 Ю	230 ж	238 о	246 ц	254 ю
199 З	207 П	215 Ч	223 Я	231 з	239 п	247 ч	255 я

Имеется и другая широко используемая кодовая таблица, применимая к русскому языку, называемая KOI-8. В ней буквы русского алфавита имеют следующие десятичные коды:

192 ю	200 х	208 п	216 ь	224 Ю	232 Х	240 П	248 Ь
193 а	201 и	209 я	217 ы	225 А	233 И	241 Я	249 Ы
194 б	202 й	210 р	218 з	226 Б	234 Й	242 Р	250 З
195 ц	203 к	211 с	219 ш	227 Ц	235 К	243 С	251 Ш
196 д	204 л	212 т	220 э	228 Д	236 Л	244 Т	252 Э
197 е	205 м	213 у	221 щ	229 Е	237 М	245 У	253 Щ
198 ф	206 н	214 ж	222 ч	230 Ф	238 Н	246 Ж	254 Ч
199 г	207 о	215 в	223 ъ	231 Г	239 О	247 В	255 Ъ

Кодировка KOI-8 является стандартной при передаче данных по сети Интернет.

Как правило, системы для просмотра и обработки документов имеют средства для перекодирования данных из одой системы кодировки в другую.

Для многих целей кодировать только алфавит недостаточно. Например, текст всегда напечатан каким-то шрифтом, имеющим свое специфическое начертание. Для того, чтобы при декодировании этот шрифт воспроизвести правильно, необходимо его особенности закодировать. Обычно кодируется номер таблицы шрифтов – в этой таблице буква уже имеет свое начертание. Количество шрифтов в современной системе обработки текстов может достигать сотен и тысяч.

Размер шрифта или, как говорят, кегль, также необходимо кодировать. Кегли имеют свои стандартные номера, так что их кодирование проблем не представляет. Сочетание кодов: кодовая таблица – код символа – код шрифта – код кегля дает огромное количество используемых для набора текста вариантов.

Для демонстрации работы различных кодировок, запустите "Пуск - Программы - TextProcessor - TextProcessor"

Наконец, что делать, если кодов много и все коды – числовые? Как различать, где и какие конкретные кодирования применены? Конечно, один вариант - пытаться разбираться по той задаче, которая решается (для каждой задачи какое-то конкретное кодирование).

Другой путь – кодировать сами кодирования! Например, у каждого числа-кода предусмотреть, что первые две цифры – это номер кодирования, а следующие – уже сам код (в этом кодировании). Например, 10 – кодирование шрифта КОИ-8, 11 – кодирование шрифта Windows, 12 – 8-битный код цвета и т.д. Такие коды для кодовых таблиц называются тэгами. Тэги снова могут собираться в таблицы соответствий …

Избыточность в кодировании

До сих пор мы рассматривали кодирования, не оглядываясь на связанные с ними возможные ошибки. А ошибки могут происходить и при работе компьютеров: сам по себе компьютер не абсолютно идеальный исполнитель, а чрезвычайно сложное электронное устройство. Информация в нем передается электрическими сигналами исключительно малой длительности и мощности. При этом такие сигналы могут искажаться по разным причинам, в том числе и внешними (электрическими полям, наводками и т.д.)

Ошибки возможны прежде всего при записи, чтении и передаче данных. Если центральное устройство компьютера обрабатывает «правильные» коды данных неправильным образом – это уже совсем плохо и компьютер требует исправления (замены).

Как могут возникать ошибки при записи, чтении и передаче данных? Компьютер – это электрическая машина, работающая с электрическими сигналами. Части сигнала воспринимается как некоторый код. Если при чтении (записи) или передаче такой сигнал будет как-то нарушен, может произойти ошибка в работе компьютера (неправильное определение закодированных данных, затем, возможно, какие-то действия над этими «неправильными»данными и так далее).

Причин для нарушения (искажения) электрических сигналов может быть очень много, например, воздействия посторонних электрических сигналов (от других приборов, например), статическое электричество и т.д. Все это хорошо известно из электротехники.

См фильм 5.3.1. "Чтение, запись, передача цифрового сигнала и ошибки"

Причины возникновения искажений сигналов стараются учесть и устранить при конструировании компьютеров и их компонентов, но полностью устранить «повреждения» сигналов оказывается невозможным. То есть ошибки все-таки происходят, хотя и очень редко. Редко – это, например, одна ошибка на миллиард переданных, принятых и распознанных битов при передаче данных или еще реже; то же самое с чтением и записью данных. Хотя могут существовать и использоваться устройства, в которых частота возникновения ошибки больше.

Тем не менее, даже эта редкая ошибка, будучи неопознанной, может испортить результаты огромных вычислений – компьютеры считают очень быстро, поток данных идет чрезвычайно большой, а один-единственный ошибочный бит может все испортить. Поэтому надо как-то пытаться бороться с возникающими ошибками, обнаруживать их, предусматривая для этого средства не только в компьютерах, но и в самих алгоритмах чтения, записи и передачи данных. Конечно, прежде всего, в способах кодирования. То есть применять алгоритмы кодирования и декодирования, которые позволяют определять наличие ошибки, а может быть – и позволять исправлять эту ошибку.

Самым примитивным способом такого «устойчивого к ошибкам» кодирования было бы повторять при передаче (или чтении) каждый «обычный» код дважды и проверять на месте,

совпадают эти пары кодов или нет. Если не совпадают (в каком-то одном бите) – запрашивать код повторно (поскольку определить, в каком именно из несовпадающих кодов имеется ошибка – чаще всего невозможно). Поскольку вероятность возникновения ошибки в точности в одном и том же месте ничтожно мала, повторно переданный код должен совпасть в «подозрительном» бите с одним из двух уже принятых кодов. Этот «совпадающий» код уже можно считать верным. Такой способ был бы достаточно надежным.

См фильм 5.3.2. "Контроль двойной передачей данных"

Такой способ возможен, но он оказывается очень неэкономичным, т.е. очень сильно понижает быстродействие компьютера. Во-первых, надо пересылать как минимум вдвое больше данных, во-вторых, надо производить операции сравнения для каждого бита. Впрочем, обычно обработка данных идет гораздо быстрее их передачи, поэтому операции над данными предпочтительны их дополнительной передаче.

Оказывается, есть способы и не намного хуже по надежности, чем двойная передач данных, и при том весьма экономные. Например, способы передачи данных, применяющие проверку на четность. В способы передачи каждая последовательность кодов – а это всегда двоичные числа – разбивается на отрезки определенной длины (например, 4096 двоичных разрядов). Затем эти разряды суммируются как 4096 одноразрядных чисел. Получившийся результат посылается вслед за переданными в этой последовательностями кодами.

При «приемке» данных их коды также складываются тем же способом, после чего результат сравнивается с присланной контрольной суммой. Если две суммы совпадают – результат пересылки принимается; если нет – можно запрашивать повторную пересылку.

Как видим, при, таком способе пересылается дополнительно всего одно число. Длина этого дополнительного числа – не более 16 двоичных разрядов (даже если все пересылаемые разряды в кодах окажутся единицами). Надо будет еще складывать все пересылаемые числа. Но это все равно довольно быстрая операция по сравнению с пересылкой.

Теперь от чего может предохранить такой способ кодирования «с дополнением»? Конечно, может получиться, что произошла ошибка, которая не будет обнаружена. Каждая ошибка здесь – это изменение 0 на 1 или наоборот. Если в 8192 разрядах произошло одновременно 2 такие ошибки, ничего обнаружено не будет. Поэтому от «двойных ошибок» такой контроль четности не предохраняет.

Но здесь вступает в дело вероятность. Если ошибка в среднем возникает один раз на миллиард разрядов, то две ошибки на протяжении 8000 с небольшим разрядов будет в среднем происходить чрезвычайно редко – настолько редко, что с этим можно смириться.

Оказывается удобным разделить все возможные кодирования. Безизбыточное кодирование – это когда ошибку в кодировании ни обнаружить, ни определить в общем случае невозможно. Или возможно в каких-то не самых главных или неприятных случаях. Т.е. все будет декодировано, но не соответствовать исходному объекту – просто будет проставлен не тот объект.

Наоборот, избыточные кодирования могут использовать «излишества» в коде для того, чтобы обнаружить ошибку – или даже иметь средства к ее исправлению.

Например, для контроля и исправления ошибок можно применить следующий способ кодирования. Дополнительно к передаваемой цепочке битов (пусть, например, снова 4096 бит) можно

1) взять (двоичные) номера всех разрядов, на которых в этой цепочке находятся единицы;

2) в этих номерах взять каждый разряд и вычислить сумму по всем номерам;

3) полученные числа переслать как контрольные.

Полученные при вычислении данные не займут много места! (Вычислите, сколько самое большее.)

После приема цепочки битов и контрольных данных производится проверка: для принятой цепочки вычисляются те же числа, что и для исходной цепочки, и сравниваются с принятыми. Теперь составляем двоичное число, у которого единицы в тех и только тех разрядах, которым соответствуют несовпадающие контрольные числа. Это число указывает номер бита исходной цепочки, в котором произошла ошибка.

Построенные подобным образом контрольные данные используются в кодах Хемминга.

См фильм 5.3.4. "Код, исправляющий ошибку"

Имеются устройства хранения информации, в которых количество ошибок и при записи, и при чтении довольно высоко. Это, например, и магнитные носители (диски, дискеты), и оптические диски (CD-диски). Такая повышенная возможность ошибок компенсируется применением большого количества корректирующей информации.

Например, емкость обычного CD-диска составляет примерно 800 мегабайт. Именно столько, например, на него можно записать цифровой музыки – здесь наличие небольших ошибок не играет роли. В то же время при записи компьютерных программ и других цифровых данных, в которых любая ошибка является критической, на CD-диск входит около 700 мегабайт информации. То есть примерно 100 мегабайт занимают блоки данных коррекции ошибок!

Процесс исправления ошибок при чтении CD-диска можно наблюдать непосредственно: иногда можно видеть, как устройство чтения дисков раз ха разом «пытается прочитать» диск. На самом деле чтение происходит при каждой попытке, но при чтении вычисления с использованием кодов коррекции показывают, что при чтении произошла ошибка. Место ошибки (ошибочный блок) отмечается, происходит повторное чтение.

Возможно, снова с ошибкой (не обязательно в том же бите). Попытка снова повторяется и т.д. – до того момента, когда блок был прочитан правильно. После этого процесс чтения переходит на следующие блоки. Именно поэтому некоторые, «плохие» диски читаются довольно медленно, а некоторые перестают читаться вовсе.

Аналого-цифровое кодирование

При кодировании различной информации применяются не только цифровые кодирования. Связано это прежде всего с процессами передачи данных по волноводам – электрическим проводам и кабелям и оптическим кабелям, то есть с электрическими и световыми сигналами.

Как известно из курса физики, по волноводам можно передавать колебания различной частоты и амплитуды (размаха). С помощью этих частот и амплитуд и требуется кодировать передаваемую информацию.

Цифровое кодирование осуществляется следующим образом: через строго определенные промежутки времени измеряется амплитуда (уровень) сигнала. Если он достигает определенной величины – сигнал в этот момент времени распознается как 1, если не достигает – распознается как 0. На практике делают так, чтобы сигналы, соответствующие 0 и 1, достаточно сильно различались – например, для сигнала 0 было от 0 до 0.2 вольта, а для сигнала 1 – от 0.8 до 1.2 вольта. Тогда возможность ошибиться при раскодировании (распознавании) будет гораздо меньше.

Для некоторых видов данных (прежде всего, звука), технически более простыми оказались методы аналогового кодирования, применявшие непрерывные свойства электрического сигнала, а не «различения скачков», как в цифровом кодировании. Эта техническая простота происходила из-за того, что звуковой сигнал, с одной стороны, содержит относительно много информации, которая передается быстро, а с другой – эту информацию можно передавать с некоторыми искажениями, не совсем точно. Поэтому исторически аналоговое кодирование стало массово применяться в телефонии в конце 19 и в радиосвязи в начале 20-го века, когда применение цифровых методов кодирования было весьма ограниченным (если не считать кода Морзе и ему подобных, применявшихся для «медленных» видов передачи данных).

Первым широко распространенным способом кодирования звука для передачи электрическим сигналом был способ прямого преобразования звуковых волн в «такие же» электрические путем использования специального прибора – микрофона. Обратное преобразование осуществлялось также специальным прибором – динамической головкой (применяемой в звуковых головках).

Такой способ кодирования позволял передавать по одной паре проводов только один звуковой сигнал и был, поэтому, весьма неэкономичен в телефонных системах, радиосвязи и так далее. Поэтому быстро была придумана система уплотнения сигнала, использовавшая кодирование амплитудной модуляцией. Этот способ состоит в том, что электрический сигнал некоторой фиксированной частоты (несущей частоты) изменяется по амплитуде так же, как изменяется кодируемый звуковой сигнал.

Декодирование осуществляется выделением амплитудного сигнала из несущей частоты. Если несущая частота много больше звуковой, этот способ позволяет довольно точно передавать звуковой сигнал. Одновременно, используя разнесенные между собой несущие частоты, можно передавать по одной паре проводов одновременно несколько звуковых

сигналов (например, телефонных переговоров), не смешивая их между собой. Разумеется, это влечет некоторое усложнение передающей и принимающей аппаратуры.

Следующим широко распространенным способом аналогового кодирования звука была частотная модуляция. При частотном кодировании несущая частота «смешивалась» с частотой звукового сигнала. Такое кодирование оказалось меньше подверженным помехам, то есть искажениям сигнала при передаче из-за появления посторонних сигналов («шумов»). Декодирование осуществляется «отфильтровыванием» несущей частоты.

См фильм 5.4.4. "Частотная модуляция"

Звуковые сигналы можно кодировать и в цифровом виде. Основной способ – задать кривую, соответствующую звуковой волне, последовательностью точек на этой кривой, таким образом, чтобы по этим точкам можно было достаточно точно восстановить форму кривой. Для этого точки надо «сажать» на кривой достаточно часто.

Звуковая волна после микрофона превращается в электрическую, так что задача состоит в измерении и записи (двоичным числом) значения электрического сигнала в некоторые моменты времени. Для простоты выбирают моменты отсчета расположенными через одинаковые интервалы времени. Таким образом, после кодирования звука получается набор двоичных чисел, каждое из которых соответствует некоторому моменту времени. Эти моменты времени не надо указывать каждый раз, если интервал времени постоянен. Поэтому окончательно получается цепочка (последовательность) двоичных кодов.

Описанное кодирование электрического сигнала (соответствующего звуковому) в цифровом виде осуществляется специальными электронными устройствами – аналого-цифровыми преобразователями (АЦП).

Обратная операция – декодирование последовательности двоичных чисел в электрический сигнал, соответствующий звуковому сигналу, также осуществляется специальными электронными устройствами – цифро-аналоговыми преобразователями (ЦАП). ЦАП иногда еще называют «музыкальным процессором».

Качественно работающие аналого-цифровой и цифро-аналоговый преобразователи - достаточно сложные и дорогие устройства, не уступающие в этом лучшим процессорам для персональных компьютеров. В дешевой бытовой аппаратуре, конечно, устанавливаются их относительно простые и дешевые варианты.

Как уже отмечалось, аналоговые методы кодирования звука широко применяются и по сей день в телефонии. Телефонные сети чрезвычайно широко распространены, а это привело к их использованию и для организации недорогой связи между компьютерами. Здесь задача ставится в обратном порядке – преобразовать цифровой код в аналоговый, из которого на приемном конце извлечь назад цифровой.

При этом, конечно, применяются особые разновидности ЦАП и АЦП, которые устанавливаются в специальные устройства для передачи и приема цифровых данных по

телефонным линиям связи – телефонные модемы. Связь между компьютерами с помощью модемов, по сегодняшним стандартам, достаточно медленная, но и она способна обеспечить приемлемую связь с системами электронной почты, сетью Интернет и т.д.

Измерение количества информации в битах

Двоичные кодирования являются стандартными для представления всевозможной информации в компьютерах. Сюда относятся тексты, формулы, звук, обычные и движущиеся изображения (видео), словом, все, что можно представить и обработать в компьютере. Очевидной задачей является запись и хранение на специальных устройствах информации, записанной в цифровом виде, откуда ее можно при необходимости считывать и использовать.

Другой обычной задачей является передача цифровой информации на различные расстояния по линиям связи.

Как обычно, количество информации, которую можно записать на один носитель информации, является ограниченным. Так же само, количество информации, которое может быть передано по той или иной линии передачи данных, также ограничено. Поэтому и передача, и хранение информации требуют оценок необходимых «объемов» - физических и «скоростных».

Оценка всегда означает измерение. И здесь оказывается, что проще и удобнее всего имеющиеся цифровые данные измерять их длиной, то есть длиной цифрового кода для этих данных.

Как мы знаем, цифровые коды применяются всегда двоичные (другие, восьмеричные, шестнадцатеричные приводятся к ним очевидным образом). Поэтому за единицу записи удобно выбрать одноразрядное двоичное число. Она же будет и единицей «длины» записи. Поэтому одноразрядное двоичное число и принимают за единицу количества информации, называемую один бит. Например, количество информации в записи 27-разрядного двоичного числа – 27 бит. А в трех 14-разрядных двоичных числах количество информации – равно 3*14 = 42 бита.

Как всегда, имеются и более крупные единицы измерения. Однако на практике количества информации чаще измеряют в байтах. Каждый байт соответствует восьмиразрядному двоичному числу, то есть равен 8 битам и соответствует, как правило, коду одного какого-то символа. Измерение количества информации в таких «условных символах» наиболее удобно.

Еще более крупные единицы измерения – килобайты (1024 байт, обозначение КБ), мегабайты (1024 килобайт, обозначение МБ), гигабайты (1024 мегабайт, обозначение МБ). Входят в оборот даже такие единицы, как терабайты (1024 гигабайт, обозначение ТБ)! При приблизительных расчетах можно округлять 1024 до тысячи: например, 1 гигабайт – приблизительно 1 миллиард байт.

Приведем некоторые примеры количеств информации, умещающихся на различных носителях информации.

При передаче данных двоичная единица или ноль занимают некоторую длительность сигнала. Обратная величина к этому временному промежутку называется частотой передачи данных. Эта частота – количество бит, передаваемых в единицу времени, как правило, в

минуту. Обычно применяемые линии передачи данных имеют ограниченную пропускную способность.

Посмотрим, какие объемы информации связаны с оцифровкой звука. Звуковые колебания, которые слышит человек, занимают достаточно узкий диапазон частот – не выше 20 кГц. На самом деле можно – при небольшой потере качества звучания ограничиться звуковыми частотами до 12 кГц, а при передаче разговора (например, для телефонных аппаратов) – 3 кГц и даже менее.

Вместо одинакового расстояния между точками (отсчетами) на кривой (в миллисекундах, например), также удобно говорить о частоте, с которой идут эти точки. Как выяснилось, при частоте взятия точек-отсчетов 44 кГц (тесть 44 тысячи отсчетов в секунду), качество декодированного звука становится удовлетворительным. На практике принимается частота 44,1 кГц.

Теперь возникает вопрос – сколько бит отводить на запись амплитуды сигнала при каждом отсчете. Поскольку память отводится байтами, выбирать надо между 1 байтом, 2 байтами и т.д. Хорошие результаты воспроизведения получаются уже при 2 байтах, то есть 16 битах на каждый отсчет.

Легко видеть, что при таких параметрах записи одна секунда записи музыкального сигнала будет занимать 2 * 44,1 = 88,2 КБ, минута звучания в оцифрованном виде займет около 5,3 МБ, а час – уже около 300 МБ. Час записи стереофонического звучания (удвоение объемов записи) займет уже более 600 МБ.

В настоящее время внедряется новый стандарт – 96000 отсчетов в секунду, то есть отсчеты берутся с частотой 96 кГц. Это позволяет сделать звучание декодированного звукового сигнала отличным.

Цифровое изображение является еще одним видом информации, которое играет огромную роль в современной информатике.

Имеется несколько методов формирования изображения, которые не являются связанными только с компьютерами (например, в обычном телевидении, в полиграфии и т.д.) Обычно изображение «собирается» из точек, настолько маленьких, что глаз эти точки по отдельности не различает. Место каждой точки можно задать ее координатами, если такие ввести для изображения. Иначе говоря, место для каждой точки можно указывать номером строки и номером столбца, в которых эта точка находится. Однако проще указать число строк и столбцов точек в изображении, тогда можно указать просто цепочку точек, а место каждой можно вычислить. Например, если изображение 800*600 точек, то 23456-я точка будет находиться в 40-ой строке (делим 23456 нацело на 600 и прибавляем 1), а в этой строке – в 56 –ом столбце (23456 – 600*39 = 56).

Теперь в каждой точке необходимо закодировать ее цвет. Здесь имеется несколько способов. Например, в телевизорах применяется так называемая модель RGB (а компьютерный монитор – это, по сути, особая разновидность телевизора). Модель RGB использует

следующее свойство человеческого глаза: все цвета можно получить, «смешивая» лучи света трех цветов (длины волны света). Для RGB это красный, зеленый и голубой (Red, Green, Blue – отсюда и название RGB).

Для каждого цвета надо задать его интенсивность (яркость). Как и для звука, можно задать относительную яркость – самую большую как кратное самой маленькой. «Смешивая» в одной точке лучи трех цветов (каждый – с некоторой интенсивностью), получают любые другие цвета любой яркости – в пределах возможностей аппаратуры.

Например, как получается белый цвет? Особенностью человеческого глаза является то, что он воспринимает очень интенсивный свет как «белесый», то есть приближающийся к белому. Поэтому белый цвет – это равное смешение всех трех цветов при большой яркости. Если яркость понижать, то белый цвет начнет «сереть», ну а полное отсутствие яркости (отсутствие всех трех цветовых лучей) будет восприниматься, как и должно быть, как черный цвет.

Следовательно, при кодировании цвета в модели RGB, необходимо

1) разложить этот цвет на три основные составляющие;

2) измерить интенсивность этих составляющих двоичными числами;

3) записать полученный результат.

Стандартом является использование 256 уровней интенсивности для каждого цвета, что, во-первых, соответствует одному байту (8 битам) информации, что удобно для «экономной» записи и обработке в компьютере, а также примерно соответствует тем уровням интенсивности, которые еще способен различить человек.

Таким образом, для кодировании информации об одной цветовой точке (пикселе) необходимо 24-разрядное двоичное число. Поэтому такую схему кодирования часто называют «24-разрядным цветом».

Таким образом при задании изображения в цифровом виде количеством информации будет (число точек на экране) * 3 Байт. Например, если на экране имеется 800 строк и 600 столбцов точек изображения (называемых пикселями), то объем информации будет 800 * 600 * 3 байт, то есть приблизительно 750 КБ.


См фильм 5.4г – цветовое кодирование (RGB).


См фильм 5.4г – цветовое кодирование (CMYK).


См фильм 5.4г – цветовое кодирование (HSB).

А какая скорость передачи информации необходима для передачи телевизионного изображения? Пусть передаваемое изображение имеет 600 строк и 500 точек на экране (что очень приблизительно соответствует телевидению) и передается 24 кадра в секунду. Тогда объем передаваемой информации будет 600 * 500 * 3 * 24 байт, то есть несколько больше 21 МБ в секунду. На обычно CD-диске тогда разместилось бы всего 35 секунд записи!

Как видно, изображение требует огромных объемов кодов, особенно это относится к видео изображениям. Поэтому интенсивно разрабатывались и разрабатываются методы уменьшения объемов необходимых кодов, или, как еще говорят, сжатия изображения. Здесь применяются два основных метода.

Во-первых, можно очень сильно уменьшить объем информации, не очень сильно ухудшив качество картинки. Прежде всего, это уменьшение количество малозаметных переходов цвета в картинке.

Во-вторых, можно еще раз закодировать сам код картинки. Поясним это на примере. Пусть имеется некоторая цепочка битов (по-другому – большое двоичное число). При этом пусть единицы в этой цепочке встречаются достаточно редко, пусть в среднем одна на 500. Тогда можно было бы закодировать эту цепочку так: в новом коде каждый байт будет означать число нулей до следующей единицы в кодируемой цепочке. Тогда, скажем, вместо части цепочки из 400 нулей подряд у нас будет два байта нового кода, то есть всего 16 разрядов.

Заметим, что если первый способ искажает изображение, то по второму способу можно точно восстановить исходный код. Как говорят, первый способ сжатия ведет к потере некоторой части информации, а второй – нет.

Из второго также следует, что «количество» информации, измеряемое в битах – величина, которая зависит от способа кодирования.

Простейший цифровой процессор

Рассмотрим, как может быть устроен простейший цифровой процессор, то есть исполнитель команд, представленных в цифровом виде (иными словами - в форме чисел). Такой процессор мог бы работать в составе компьютера. Здесь мы, однако, резко упростим задачу и не будем пытаться описать реальный процессор персонального компьютера: на сегодняшний день такой процессор – очень сложное устройство, далеко выходящий за минимальные требования к нему.

Начнем с того, что определимся, с какими числами работает процессор. Разумеется, с двоичными. Можно – с восьмеричными, тогда это будут те же двоичные, но с числом разрядов, кратным четырем. В соответствие с разрядностью обрабатываемых чисел, процессоры могут быть 4-битными, 8-битными, 12-битными, 16-битными и т.д.

Как мы знаем, двоичные числа всегда представляют некоторые коды. Закодированы должны быть инструкции алгоритма (алгоритмического языка). В какой форме это может быть сделано и какие вообще команды должен исполнять процессор? Оказывается, даже в простом случае набор команд требует достаточно аккуратного проектирования.

Во-первых, процессор может выполнять некоторые арифметические операции. Например, сложения, вычитания, умножения и деления над числами с запятой. Для исполнения таких инструкций требуется:

1) знать данные, над которыми операция производится;
2) знать саму операцию;
3) знать, куда записывать результат.

Для простоты будем говорить о сложении двух чисел. Данные для этой операции, очевидно, должны храниться в памяти компьютера. Для того, чтобы их использовать, надо эти данные сначала прочесть, то есть переписать в предназначенное для этого место в процессоре. В процессоре всегда имеется своя, пусть и очень маленькая, память – для непосредственно используемых данных, текущих результатов и т.д.

Вообще же память для данных не обязательно является частью процессора. Поэтому наличие такой памяти надо учитывать в инструкциях для процессора: для того, чтобы прочесть данные, надо указать место в памяти, где они записаны. Для этого достаточно указать адрес места в памяти с записанными данными. Это надо сделать для каждого данного – место в памяти, отведенное для одного данного называется ячейкой памяти; удобства ради все ячейки памяти делаются одного размера – одного числа битов.

Адрес ячейки памяти – двоичное число. Удобно, когда адреса ячеек идут подряд, ячейки «пронумерованы». Считывание данных и присылка их в процессор является задачей специального устройства – контроллера памяти: получая адрес ячейки, контроллер пересылает ее содержимое в процессор. Аналогично, получая данное и адрес, этот контроллер записывает данное в ячейку памяти с указанным адресом.

От разрядности процессора зависит, каких размеров адреса ячеек памяти он может

обрабатывать за одну операцию. Например, если для адресов данных в процессоре предусмотрено 8 разрядов, то всего разных адресов может быть 2 в степени 8, то есть всего лишь 256. 16-битные адреса могут иметь уже 65536 различных ячеек, или, как еще говорят, 64К (= 64*1024) ячеек памяти.

Таким образом, чтобы обработать одну инструкцию по сложению двух чисел, надо прочесть саму инструкцию; в нашем примере она будет состоять уже из четырех чисел:

а) кода операции (кода операции «+»);
б) кода адреса первого данного (слагаемого; данные для операций еще называются операндами);
в) кода адреса второго данного (слагаемого);
г) кода адреса для записи результата.

Получая такую инструкцию, надо учитывать, что и данные, и операция заданы кодами. Например, разного вида числа (целые, дробные) могут иметь разные коды. Поэтому предварительно требуется еще привести эти коды к такому виду, который соответствует требованиям аппаратного исполнения этой инструкции, так как в конечном счете инструкция реализуется специальной электронной схемой. Поэтому сначала все данные поступают в декодер инструкций, а уже потом арифметическая операция исполняется в арифметико-логическое устройстве(АЛУ) процессора.

Почему это устройство не только арифметическое, но и логическое? Потому что, как будет видно ниже, в АЛУ исполняются не только инструкции для производства арифметических операций.

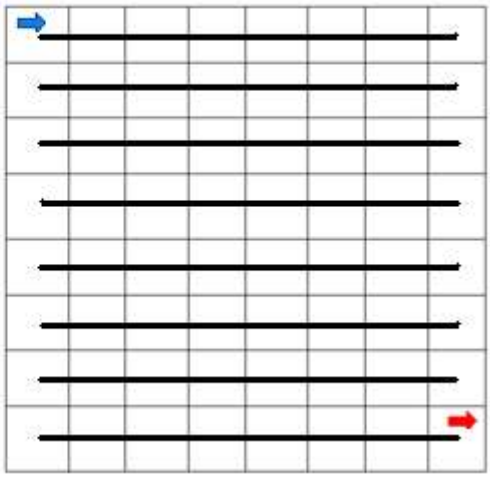
Инструкции для других арифметических операций могут быть устроены аналогично сложению и мы на них отдельно останавливаться не будем. Перейдем к порядку исполнения инструкций.

Сама программа, то есть ее инструкции, также храниться в памяти. Как мы видели, каждая инструкция в нашем примере занимает 4 числа, то есть коды можно было бы располагать один за другим в памяти и читать их последовательно, по 4 штуки. Для начала исполнения программы достаточно было бы указать ей первый адрес, с которого начинаются инструкции. Для окончания программы нужна специальная инструкция. Для ввода данных (например, с клавиатуры или еще откуда-нибудь) и для их вывода (например, на экран), конечно, нужны особые инструкции. Такие инструкции могут включать в себя два кода: код самой операции (скажем, вывода на определенное устройство) и код адреса данного, которое надо вывести. Можно было бы отвести на такую инструкцию снова 4 ячейки памяти (в двух последних ничего не надо писать – эти ячейки просто не используются). Тогда все инструкции имели бы одинаковую длину и определять адрес следующей по порядку инструкции можно было бы единообразным способом.

Однако серьезные программы никогда не используют инструкции только последовательно: нужно иметь возможность перехода не на следующую по порядку инструкцию, а на какую-то другую. Для этого необходимо предусмотреть специальные инструкции перехода: в них достаточно указать адрес начала инструкции, которая должна исполняться следующей. Как видим, такая инструкция требует снова минимум двух ячеек памяти.

Наконец, можно предусмотреть проверку условий. Например, сравнений вида a < b. Здесь необходимы код операции сравнения, два кода данных a и b, а также результат проверки неравенства. Как правило, в зависимости от того, истинно или ложно неравенство, выполняются разные инструкции алгоритма. Поэтому четвертый код может быть кодом той инструкции, которая должна выполняться следующей, если сравнение истинно. В противном случае будет исполняться следующая по порядку инструкция. Таким образом, здесь также можно уложиться в инструкцию из четырех кодов, то есть не выйти за длину инструкции из четырех чисел.

Теперь можно нарисовать схему простейшего процессора в таком виде:



Инструкции для процессора могут быть организованы и разными другими способами. Например, можно уменьшить длину инструкции до двух кодов – кода операции и кода одного данного.

Для этого можно выделить специальную ячейку памяти в процессоре, в которой постоянно будет находиться «второе данное» для операций. Например, второе слагаемое, второй сомножитель, второе член для сравнения и т.д. результат операции записывать в эту же ячейку памяти. Разумеется, тогда будут необходимы еще инструкции чтения из памяти в эту специальную ячейку памяти и записи из нее в память по заданному адресу. Тем самым сами инструкции становятся короче, но число инструкций для различных действий увеличивается.

Работа с программой TextProcessor

Установка TextProcessor

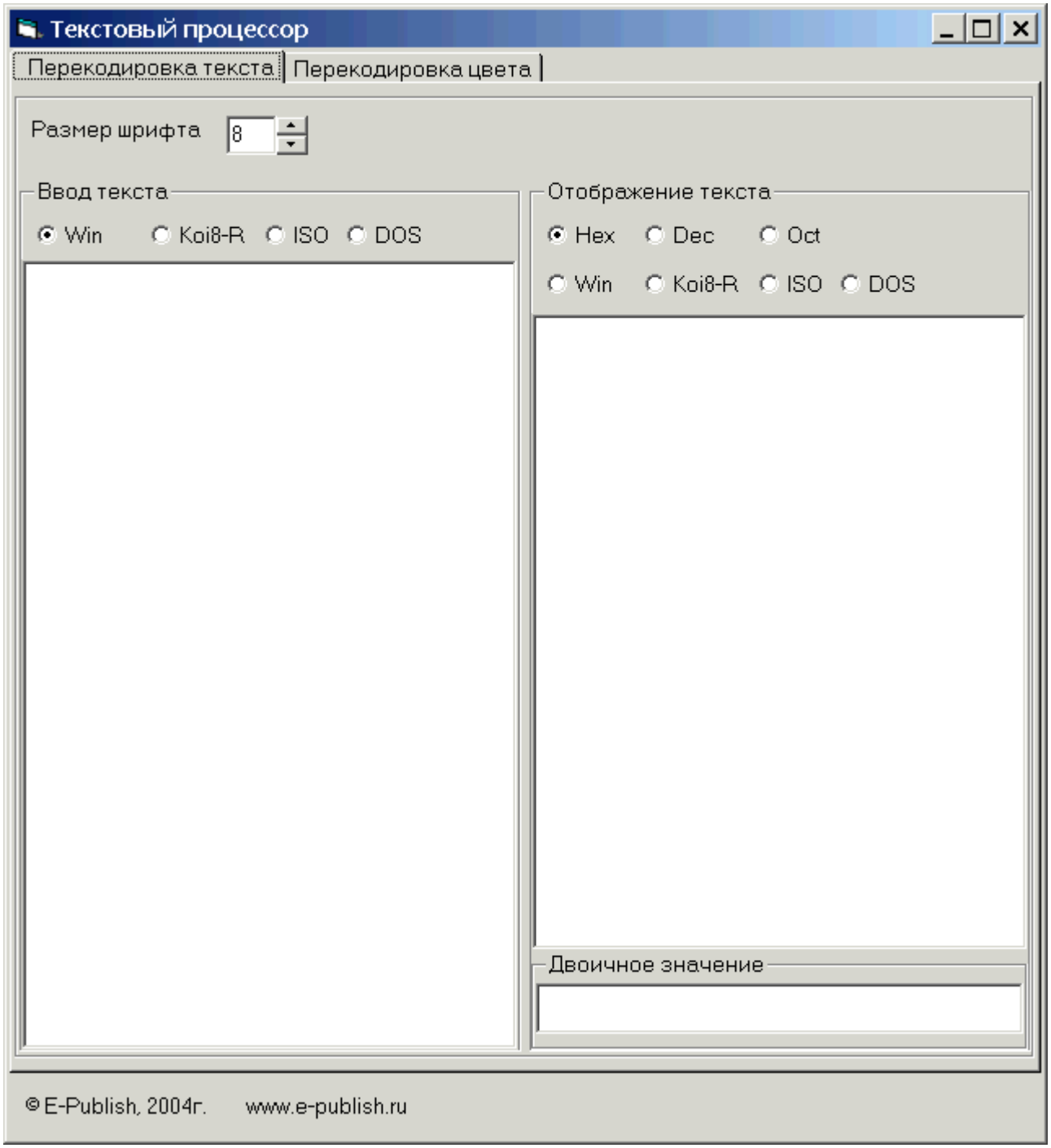
1. На первом диске откройте папку **Distributives\ePublishTextPro** и запустите файл **setup.exe**

2. Установите программу.

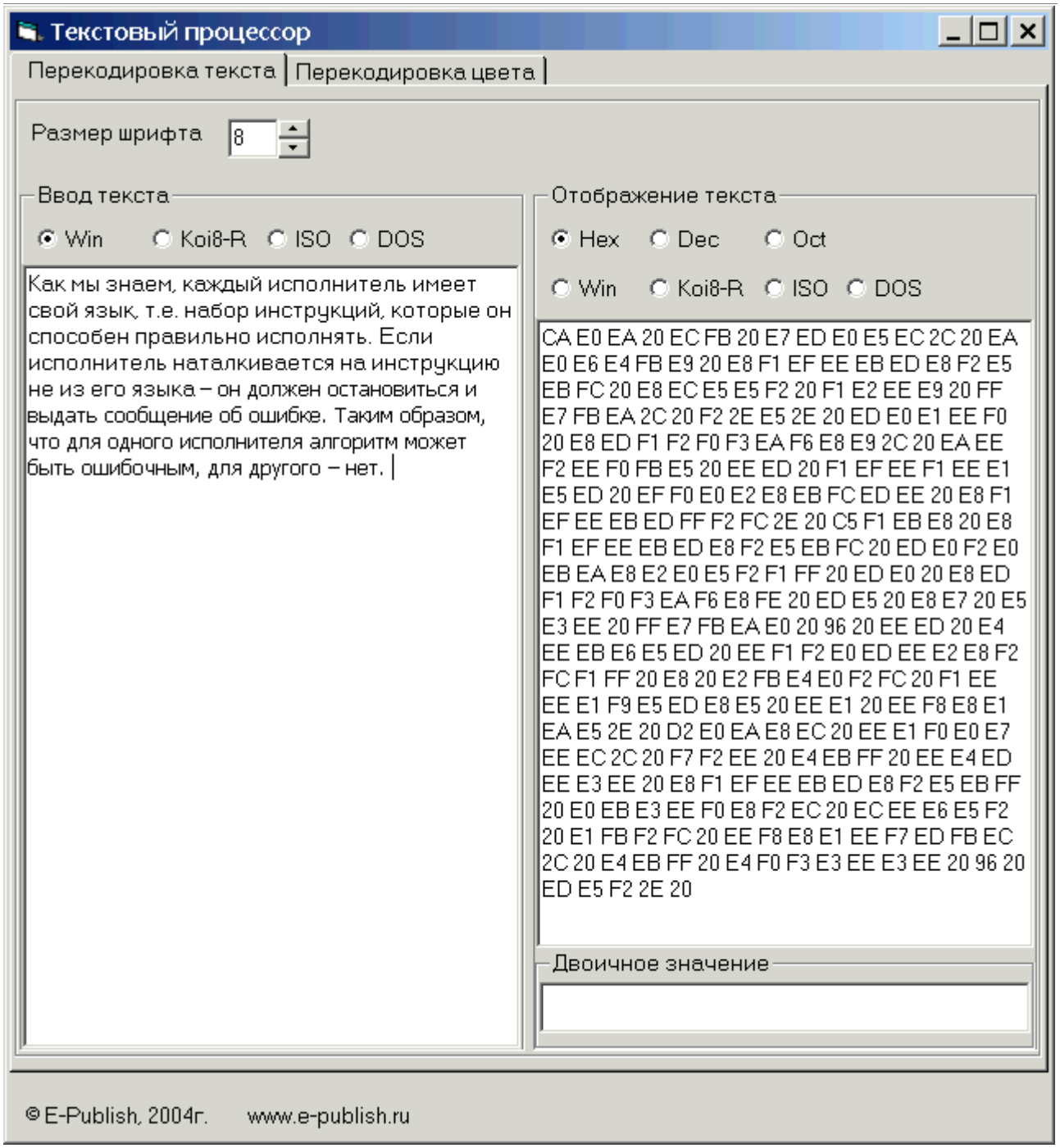
3. Запустите программу: **Пуск\Программы\TextProcessor\TextProcessor**

Перекодировка текста

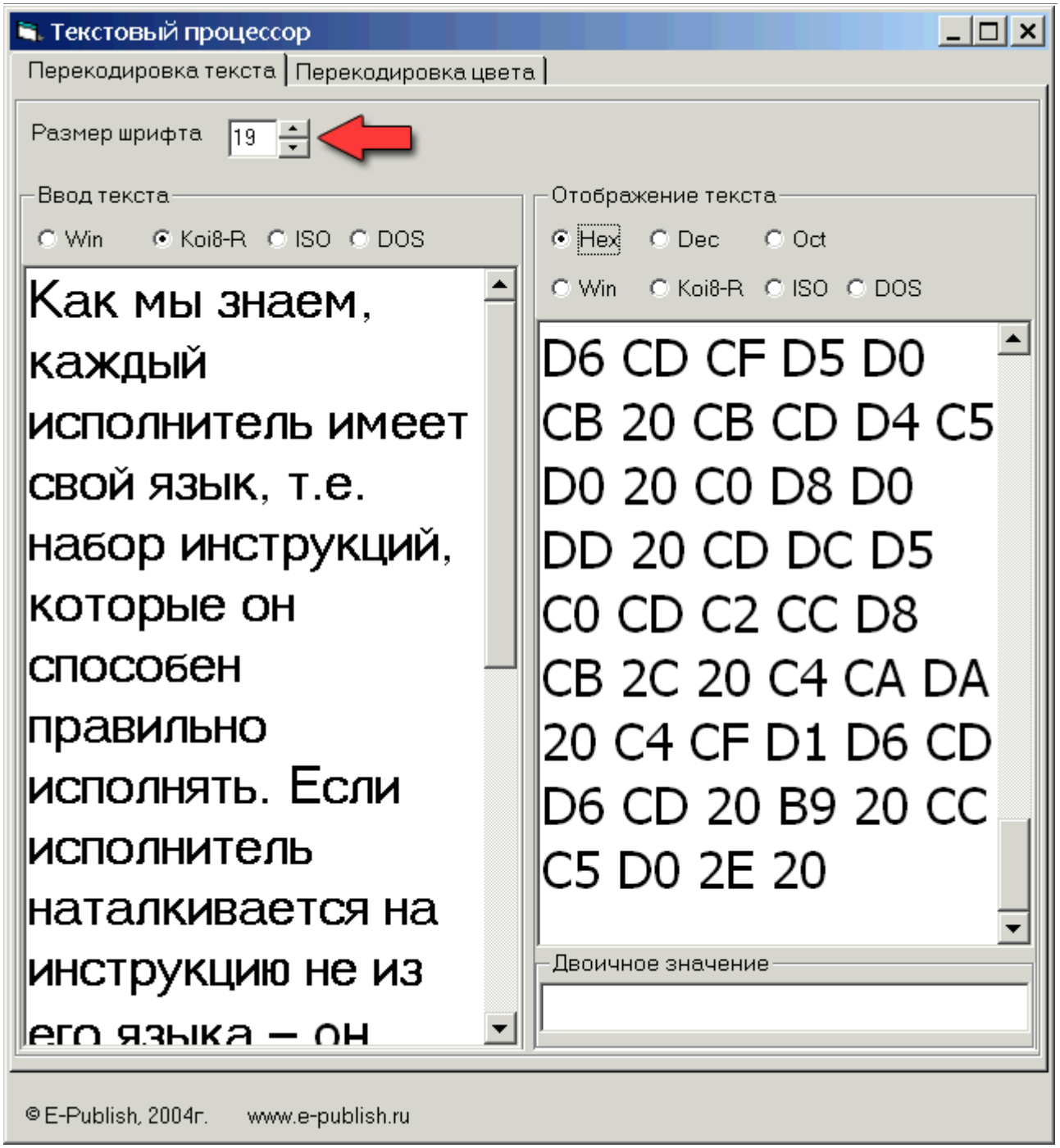
Запустите программу:



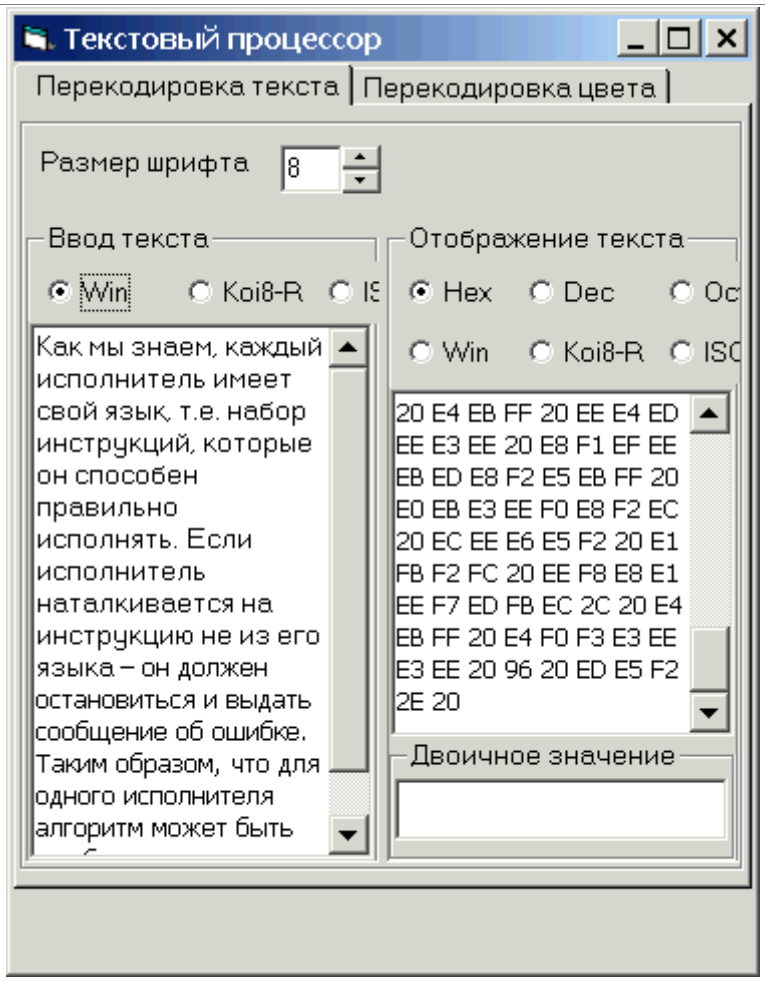
Введите в левое окно текст для перекодировки.

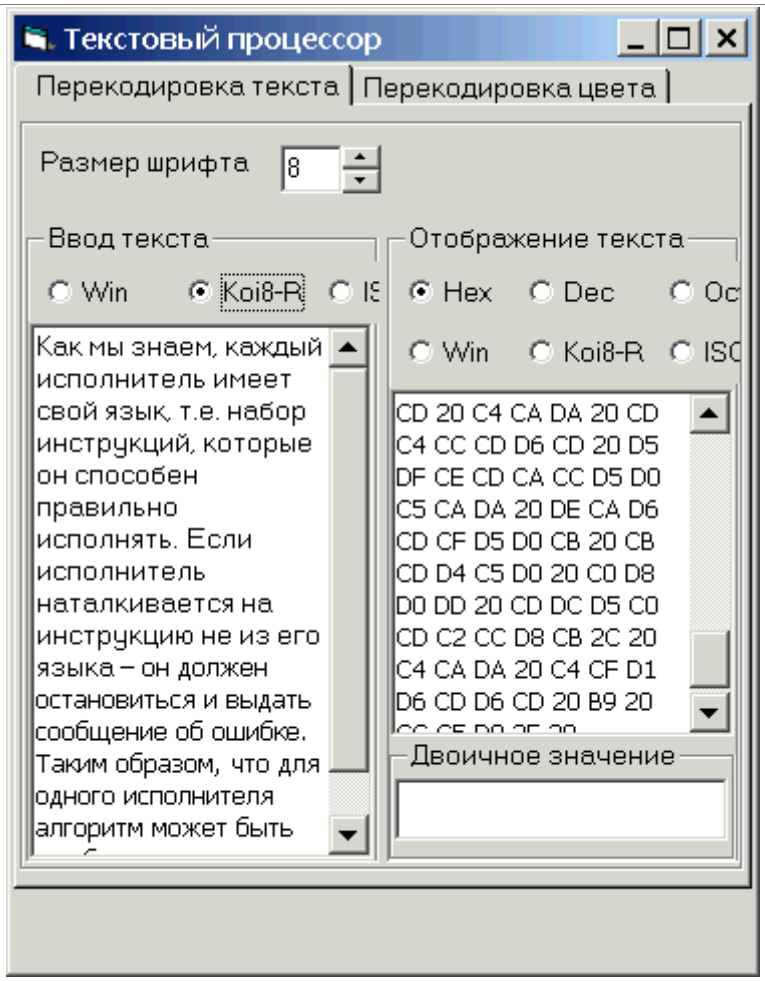


Если вас не устраивает размер шрифта его можно изменить в соответсвующем поле



Так же вы можете изменить начальную кодировку текста. При этом текст в окне ввода будет отображаться по прежнему, а вот результат кодарования изменится.

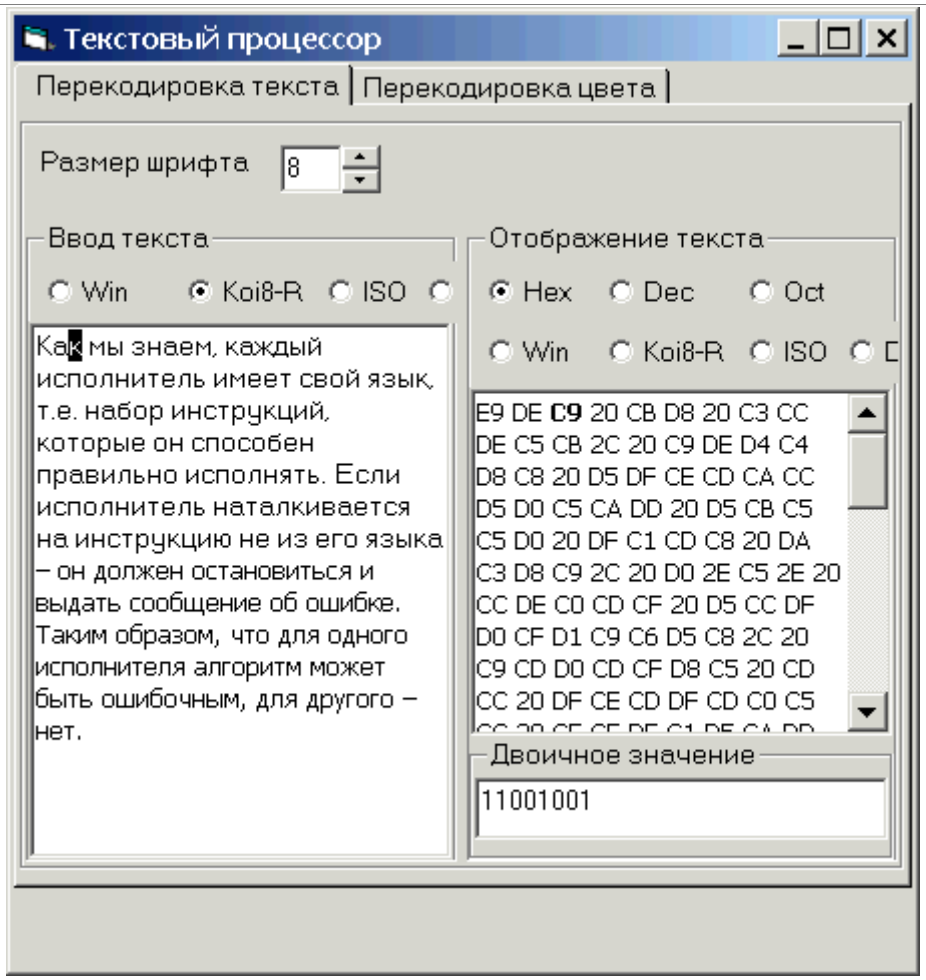

Текстовый процессор
Перекодировка текста
Перекодировка цвета
Размер шрифта
8
Ввод текста
Win
Koi8-R
Отображение текста
Hex
Dec
Win
Koi8-R
Как мы знаем, каждый исполнитель имеет свой язык, т.е. набор инструкций, которые он способен правильно исполнять. Если исполнитель наталкивается на инструкцию не из его языка – он должен остановиться и выдать сообщение об ошибке. Таким образом, что для одного исполнителя алгоритм может быть
20 E4 EB FF 20 EE E4 ED EE E3 EE 20 E8 F1 EF EE EB ED E8 F2 E5 EB FF 20 E0 EB E3 EE F0 E8 F2 EC 20 EC EE E6 E5 F2 20 E1 FB F2 FC 20 EE F8 E8 E1 EE F7 ED FB EC 2C 20 E4 EB FF 20 E4 F0 F3 E3 EE E3 EE 20 96 20 ED E5 F2 2E 20
Двоичное значение



Для введенного текста доступны кодировки Win, Koi8-R, Iso, Dos

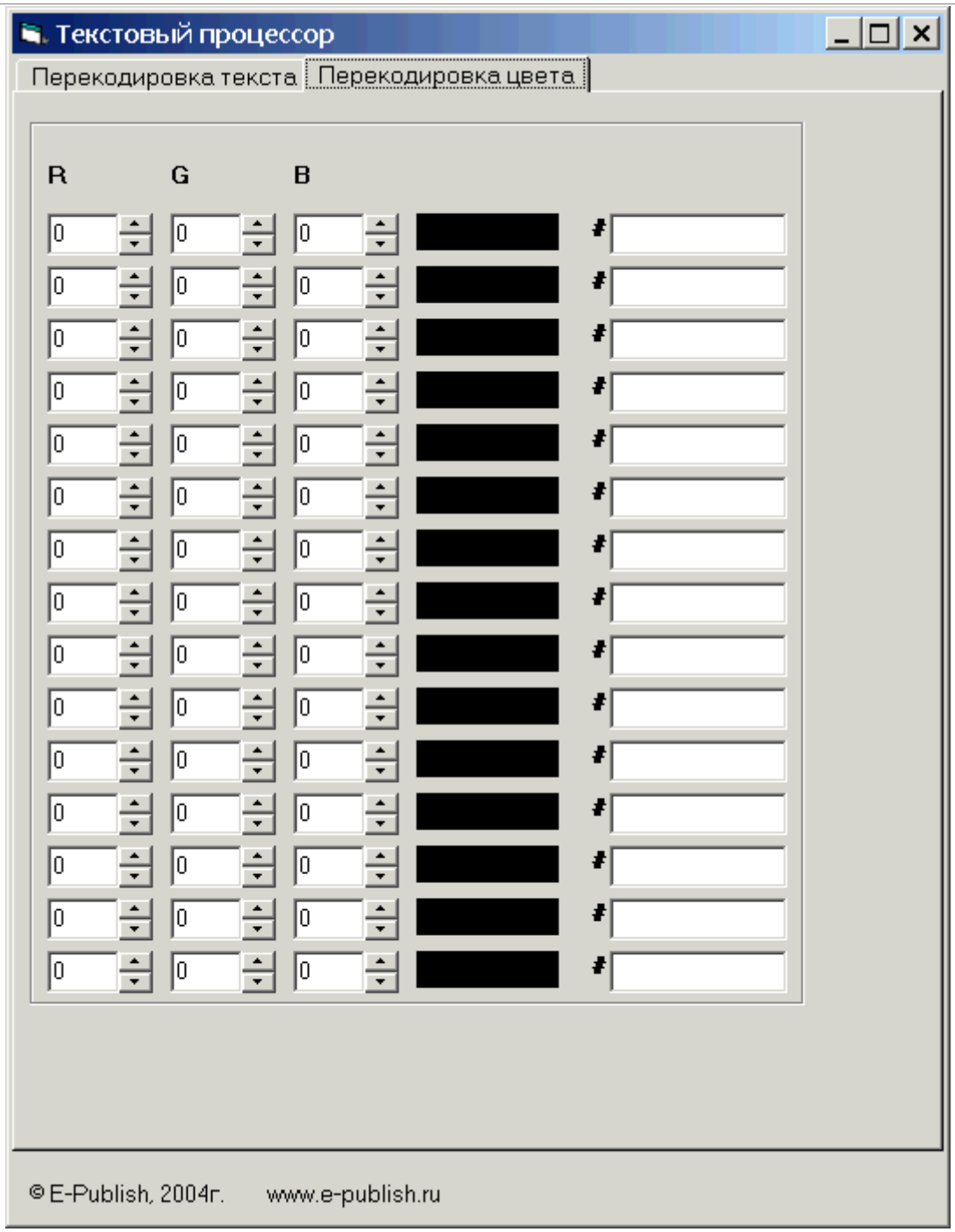
Для перекодированного текста доступны те же кодировки а так же цифровые значения символов вшестнадцатеричной кодировке (Hex), десетяричной (Dec) и восьмиричной (Oct)

Независимо от того, в какой кодировке у вас отображается текст, если щелкнуть по символу или группе симовлов знака, выделиться соответсвующий ему символ в окне ввода и его двоичное значение

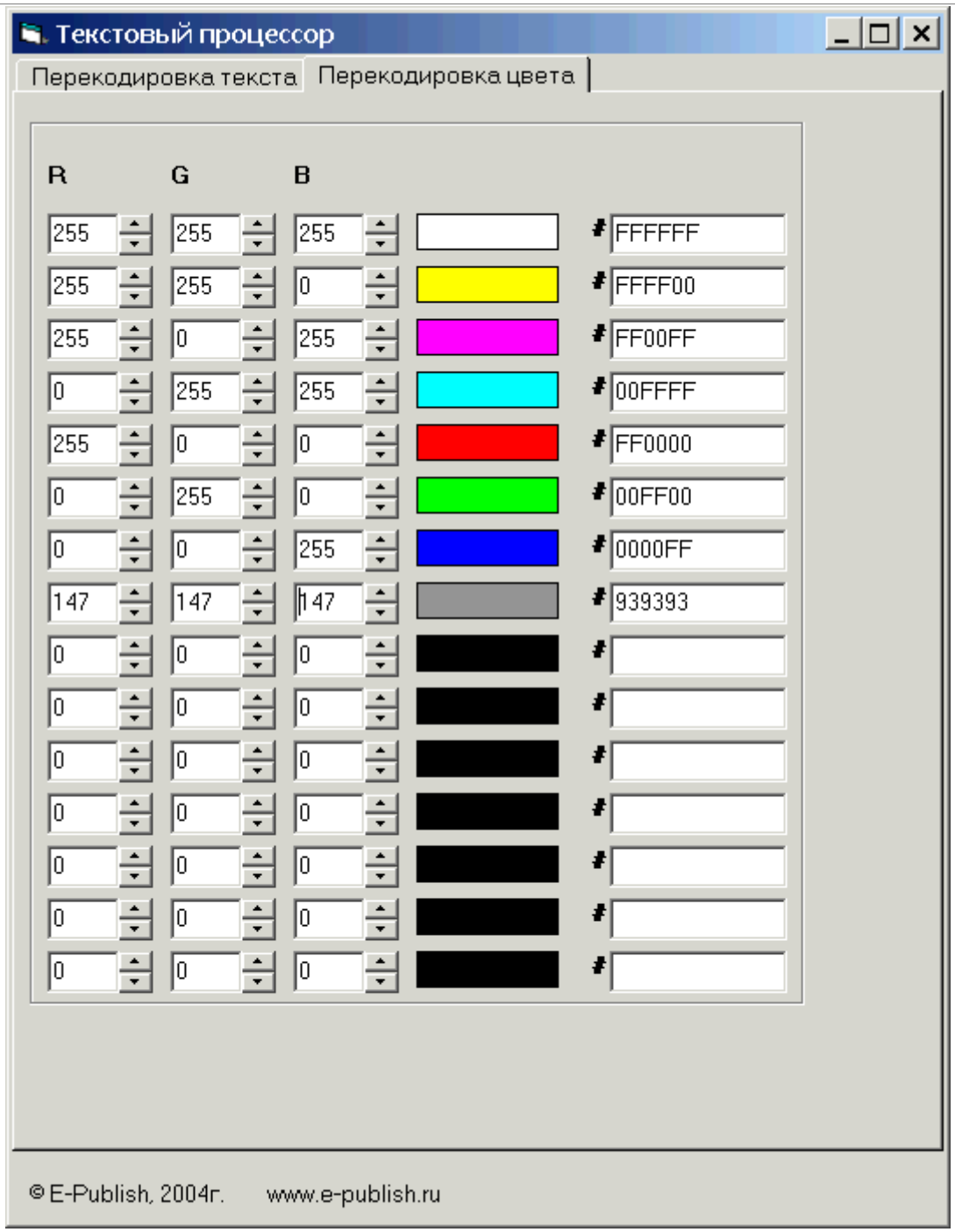


Перекодирование цвета

Перейдите на вкладку перекодировку цвета



Прокручивая соответсвующие ползунки выставите интерисующий вас цвет в RGB в цветовом поле отобразится его цвет и в следующем поле шестнадцатеричный код цвета (который используется в Веб)


Текстовый процессор
Перекодировка текста
Перекодировка цвета
R
G
B
255 255 255 FFFFFF
255 255 0 FFFF00
255 0 255 FF00FF
0 255 255 00FFFF
255 0 0 FF0000
0 255 0 00FF00
0 0 255 0000FF
147 147 147 939393
0 0 0
© E-Publish, 2004г.
www.e-publish.ru