# Введение

# Что такое компьютерный текст

## Текст

Как бы вы определили понятие *текст*? Наверняка, одно из определений связано с тем, что это самый простой, универсальный и надежный способ зафиксировать результаты умственной деятельности. Заглянем в толковый словарь. Здесь *текст* определяется как словесная запись мыслей, сообщений, речи. Значит, главная ценность текста заключена в кроющемся в нем смысле. Уже в раннем детстве мы учимся составлять из букв слова, а из слов – осмысленные предложения. Практически весь опыт, все знания об окружающем мире человечество накапливает в текстовой форме. Именно поэтому разработка способов компьютерного представления текста была среди первых задач, решавшихся в области компьютерных технологий. Но и до сих пор в области компьютерной обработки текста немало открытых проблем. Например, вычленение из текста смысловых значений и оперирование ими по-прежнему остается для ЭВМ весьма непростой задачей, решаемой программами *искусственного интеллекта*. Но не эти вопросы будут являться темой нашего разговора.

В данном учебном пособии мы остановимся лишь на самых простых технологиях компьютерной обработки текста, для которых смысл текста не имеет значения. Не будет считаться существенным, представляет ли некоторая комбинация символов какое-либо слово, несет ли какой-либо связный смысл предложение, составленное из определенных слов.

Зачем же нужны технологии, игнорирующие основное назначение текста? Они играют роль фундамента, на их основе строятся все остальные, более сложные способы компьютерной обработки текста.

Что же остается от обычного текста, когда из него «изъят» смысл? Структура. Именно структура компьютерного текста и методы работы с ней являются основной темой данного пособия.

## Компьютерный текст

**Компьютерный текст – это цепочка символов.**

Несмотря на простоту и краткость, такое определение является ключом к пониманию материала, изложенного далее, поэтому мы разберем его более подробно.

Под словом *цепочка* условимся понимать линейную последовательность конечной длины $N$ ($N \geqslant 0$), состоящую из некоторых элементов: $a_1, a_2, \ldots, a_N$. Элементом цепочки компьютерного текста является *символ*. Каждая буква, цифра, любой знак препинания, скобка и т.п. – это отдельный символ. Множество всех символов, из которых строится текст, называется *алфавитом*. Например, для представления слова **MUTANTUR** достаточно алфавита из шести символов: {A, M, N, R, T, U}. Само слово при этом является цепочкой из восьми символов.

M → U → T → A → N → T → U → R

Посмотрим теперь, как представить в виде компьютерного текста фразу:

**OMNIA MUTANTUR, NIHIL INTERIT**[1].

Во-первых, нужно расширить алфавит на пять новых символов: {A, E, H, I, L, M, N, O, R, T, U}. Мы теперь можем отдельно составить две цепочки: слово **OMNIA** и слово **MUTANTUR**. Но если их просто соединить[2], то получится цепочка из 13 символов: **OMNIAMUTANTUR**. Как отличить эту цепочку из двух слов от одного слова, состоящего из 13 букв? Напомним, что ссылаться на отсутствие смыслового значения у такого слова нельзя. Выходом является добавление в наш алфавит еще одного символа, имеющего специальное назначение. Мы будем называть его *символом пробела*, а обозначать в записи точкой в середине строки: {•, A, E, H, I, L, M, N, O, R, T, U}. Условимся использовать этот символ для отделения в компьютерном тексте одного слова от другого. Тогда запись двух слов OMNIA • MUTANTUR, из пяти и восьми букв соответственно, будет включать 14 символов:

O → M → N → I → A → • → M → U → T → A → N → T → U → R

В реальной практике работы с компьютерным текстом применяются и другие специальные символы. Например, **символ абзаца**[3]. Для его изображения мы будем использовать следующее графическое представление: ¶. Обычное назначение этого символа – разбиение цепочки символов на абзацы. Таким образом, располагая новым расширенным алфавитом {¶, •, ,, A, E, H, I, L, M, N, O, R, T, U}, можно представить текст, состоящий из двух абзацев: OMNIA • MUTANTUR,¶NIHIL • INTERIT. В данной цепочке 29 символов, которые представляют собой 2 абзаца: два слова в первом абзаце и два во втором.

[1] Все меняется, ничто не погибает (лат.)
[2] Применительно к компьютерному тексту, операцию соединения цепочек называют *конкатенацией*.
[3] Этот символ вводится с клавиатуры клавишей **Enter** и может называться по-разному: символ новой строки, **CR** и др.



## Компьютерный текст и его изображение

Важно понять, что простая цепочка символов определяет только структуру текста и не содержит информации о том, как этот текст *изображать*[1]. «Решение» об изображении компьютерного текста принимается той или иной программой, обрабатывающей текст. Например, программа **Блокнот**[2], входящая в состав операционной системы Windows[3], является одной из самых простых программ для работы с компьютерным текстом. Схема работы программы **Блокнот** очень похожа на то, как работает множество других приложений.

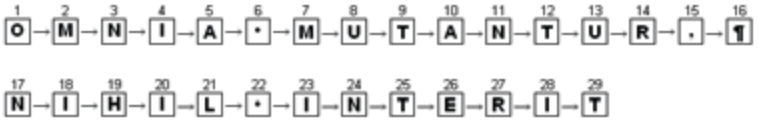

Целью пользователя является создание текстового документа – ввода цепочки символов. Программа **Блокнот** формирует изображение текста

[1] Передавать человеку при помощи тех или иных устройств вывода информации.
[2] В английской версии – Notepad.
[3] Группа программ Стандартные (для английской версии – Accessories).

в своем окне. Пользователь оценивает текущее состояние документа, принимает решение, какие изменения необходимо произвести, и подает управляющие команды при помощи клавиатуры. Программа **Блокнот** принимает команды управления, производит модификацию документа и отображает в окне произведенные изменения. На изображении текста присутствует курсор, указывающий позицию в цепочке между двумя символами. К этой позиции и привязаны вносимые пользователем изменения в компьютерном тексте. Приведем типичные команды управления.

| Клавиша | Назначение |
|---|---|
| Со **стрелками вправо** и **влево** | Перемещение курсора по цепочке на один символ, соответственно, вправо или влево |
| С **буквой** (или **цифрой**, **знаком**) | Вставить в цепочку соответствующий символ |
| **Пробел** | Вставить в цепочку символ пробела |
| **Enter** | Вставка символа абзаца |
| **Backspace** | Удаление из цепочки символа, *предшествующего* курсору |
| **Delete** | Удаление из цепочки символа, *следующего* за курсором |

Рекомендуем выполнить небольшое наглядное задание. Для этого откройте приложение Блокнот и наберите цепочку символов:




Обратите внимание на изображение этой цепочки в окне программы. Установите курсор между 16-м и 17-м символами цепочки (перед буквой N) и нажмите клавишу **Backspace**. Какая цепочка должна получиться в результате выполнения этой команды (см. описание команды в таблице выше)? Как получившаяся цепочка изображается в окне программы?

Программа **Блокнот** осуществляет графическое изображение текста на экране монитора или на принтере. Попутно отметим, что существуют программы для изображения текста и в совершенно другой форме. Например, «читающие» программы, ориентированные на чувство слуха, или программы, воздействующие на осязание и отображающие текст с помощью азбуки Брайля, для слепых.



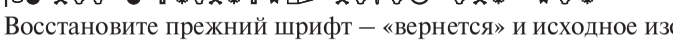
Следующий важный момент в нашем обсуждении – это вопрос о графическом изображении текста, которое строится на основе *компьютерных шрифтов*. Шрифт представляет собой таблицу, где каждому символу соответствует некоторое графическое изображение. Слова при этом имеют вид расположенных вплотную друг к другу изображений символов. Символам пробела и абзаца обычно не соответствует никаких графических изображений, это так называемые *непечатаемые символы*. Программы учитывают их при расположении слов в строках изображения текста. Обратите внимание: если в шрифте каким-либо символам соответствуют нетрадиционные изображения, текст становится нечитаемым. При этом с собственно текстовым документом ничего не происходит. Достаточно воспользоваться «нормальным» шрифтом, и изображение текста снова становится осмысленным.

К вопросу о графическом представлении текста предлагаем «эксперимент».

В окне программы **Блокнот** выделите мышью набранный ранее текст **OMNIA • MUTANTUR, • NIHIL • INTERIT** и подайте команду

**Формат ⇒ Шрифт...**

В появившемся диалоговом окне **Шрифт** обратите внимание на название шрифта, используемого для изображения текста. Замените шрифт, выбрав из списка **Wingdings** и нажав **OK** в диалоговом окне. Обратите теперь внимание на изображение текста в окне программы **Блокнот**:



Восстановите прежний шрифт – «вернется» и исходное изображение текста.

Делаем вывод – при смене шрифта собственно текст не меняется.

## Операция поиска

Подготовка печатного текстового документа – задача важная, но достаточно легко выполнимая и без компьютера, например, при помощи печатной машинки или типографскими средствами. Компьютерное представление текста обладает рядом неоспоримых преимуществ. Первое, наиболее очевидное, – возможность корректировки набранного текста как до, так и после получения его «твердой копии» на бумаге. Другие проявляются в дополнительных возможностях по его обработке. Самая про-

стая операция, рассматривающая лишь структуру текста как цепочки символов, – это *операция поиска*[1].

Типичная постановка задачи поиска заключается в следующем. Дан исходный текст с символами $a_1a_2a_3...a_n$ и текст, являющийся образцом для поиска $b_1b_2b_3...b_m$. Требуется определить, является ли образец частью исходного текста, и, если является, то начиная с какого номера символа цепочки. Иными словами, нужно найти индекс $i$ такой, что $a_i = b_1$, $a_{i+1} = b_2$, ... $a_{i+m-1} = b_m$ ($i>0$, $i+m<n$). Простейший алгоритм решения задачи поиска очевиден. Нужно последовательно перебирать индексы, начиная с единицы, и проверять, выполняется ли указанное выше условие поиска. Как только обнаружится индекс, для которого условие выполняется, поиск можно прекратить, так как образец найден, следовательно, поставленная задача решена. Если индекс выходит за допустимые границы, то поиск тоже можно прекратить, так как в этом случае образец просто не будет найден.

За кажущейся простотой операции поиска скрываются огромные возможности. Достаточно сказать, что доходы компаний, специализирующихся на поиске в текстах всемирной паутины WWW (World Wide Web), исчисляются миллионами долларов. Речь при этом идет, конечно же, не о рассмотренном выше тривиальном алгоритме, а о его усложненных вариантах. Мы здесь лишь наметим два направления усложнения операции поиска.

Во-первых, можно работать над повышением эффективности алгоритма, не меняя при этом постановки задачи. Это важно для поиска в очень длинных текстах, когда каждый перебор от начала до конца текста может занимать значительное время. Как ни странно, возможностей для улучшения простейшего алгоритма достаточно много. Например, можно уменьшить число повторных сравнений символов в тех случаях, когда для некоторого $i$ совпадают лишь часть исходного текста и образца. Неполное совпадение означает, что образец еще не найден и нужно увеличивать индекс. Но при этом обычно можно увеличить индекс не на единицу, а на большее число: мы ведь уже проверили несколько символов текста, следующих за $i$-м, и знаем, какие из них не совпадают с $b_1$ и, следовательно, не могут являться началом образца. Более подробное описание алгоритмов поиска можно найти в многочисленных учебниках по программированию.

Во-вторых, чаще всего, образцом для поиска является не единственная цепочка текста, а множество цепочек. Задача при этом превращается в поиск первого вхождения любого из элементов множества образцов. На практике множество может быть задано различными способами.

[1] Применение операции поиска в Microsoft Office Word 2003 рассмотрено в параграфе **Найти и заменить** главы 3.



Первый способ называется обычно «независимостью от регистра». Действительно, заглавным и строчным буквам в тексте соответствуют разные символы (символы разного регистра). Тем не менее, в операции поиска часто требуется найти некоторое слово, независимо от того, каким регистром оно набрано в тексте. Эту задачу можно сформулировать как поиск первого вхождения в текст любого из множества образцов, каждый из которых состоит из одинаковых букв, но в разном регистре. Так, если в слове $k$ букв могут быть в одном из двух регистров, то множество образцов будет состоять из $2^k$ различных элементов. Другим явным способом построения множества образцов поиска является использование «символов-заместителей» или *регулярных выражений*. Чисто теоретически алгоритм поиска можно представлять почти таким же, как было описано выше, но для каждого индекса $i$ сравнивать исходный текст со всеми элементами множества образцов — и только в случае, когда ни один из них не совпал, переходить к следующему индексу. Но на практике такой подход оказывается чрезвычайно неэффективным, поэтому используют совершенно иные алгоритмы.

## Текстовый документ Microsoft Word

Итак, мы обсудили некоторые важные понятия, связанные с компьютерным текстом и средствами работы с ним. Следующая, основная часть пособия нацелена на формирование практических навыков работы с текстовыми документами существенно более сложной структуры. Программа Microsoft Word обладает для этого богатейшими возможностями, и эти возможности вам будет проще систематизировать и освоить, если иметь в виду следующие моменты:

- ✓ Схема работы с Microsoft Word аналогична схеме работы с программой Блокнот[1]. Разница лишь в том, что в центре этой схемы будет находиться программа Word, а внизу – документ Word. Соответственно, цель пользователя – создать документ Word определенной структуры, определенного содержания и, соответственно, определенным образом изображаемый на экране монитора или на принтере.
- ✓ Документ Word в основе своей также содержит компьютерный текст, но структура этого документа намного богаче описанного выше «плоского» текста[2]. Такой текст, следуя дословному переводу *Rich Text*, называют «богато оформленным».

Итак, структуру документа Microsoft Word можно отобразить следую-

[1] См. рисунок этой схемы на стр. 3.
[2] Plain Text.

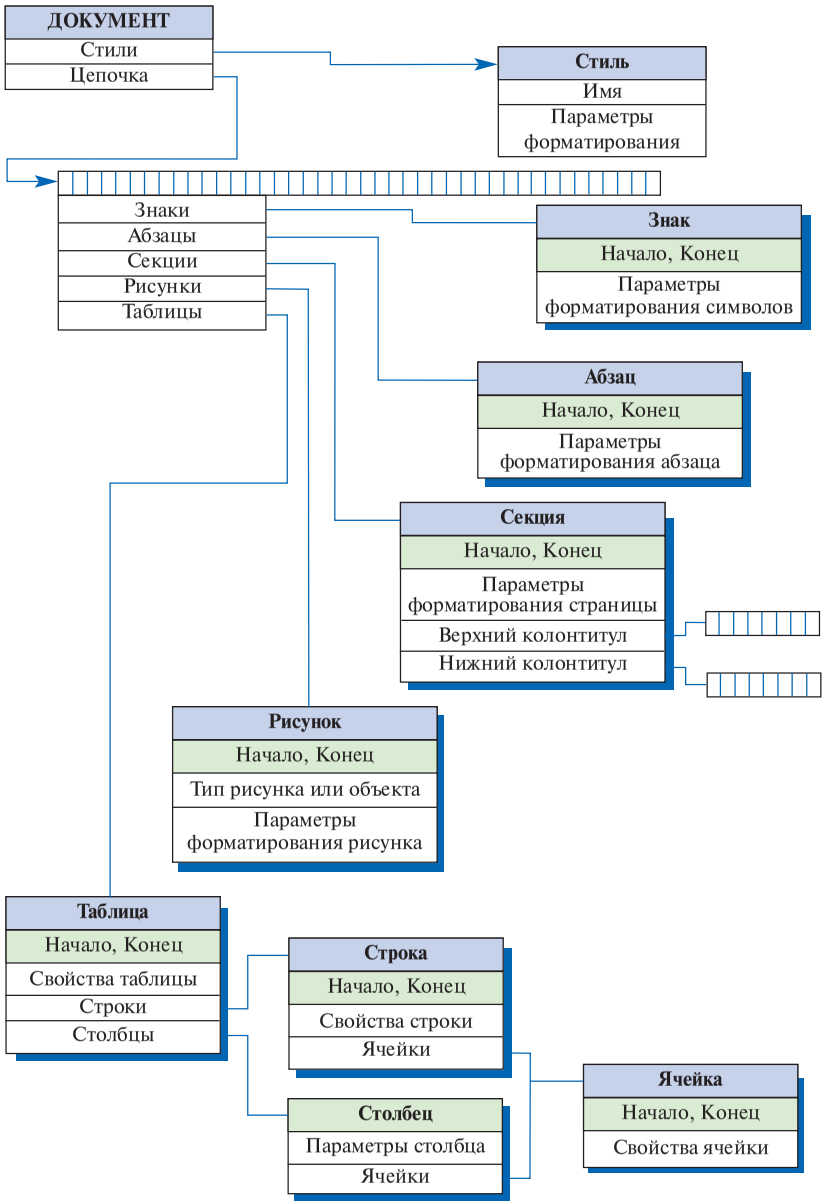



**Рис. 2.** Схема документа **Word**



щей схемой.

Простой плоский текст состоит из символов, текст документа Word состоит из *знаков*. Знак Word имеет сложную структуру. Помимо собственно символа, каждый знак содержит параметры *форматирования* символа, то есть значения свойств, определяющих способ изображения. К параметрам форматирования относятся, например, название шрифта, требуемый размер, цвет рисунка символа и другие[1].

Программа Word использует специальные знаки для того, чтобы выделять в тексте различные структурные элементы: абзацы, секции, таблицы, рисунки. Причем каждый из этих элементов несет собственный набор параметров, влияющих на отображение текста. Например, параметры форматирования абзаца определяют, как Microsoft Word будет располагать слова, составляющие абзац, на строках изображения текста.

Безусловно, не имея достаточных навыков в работе с текстовым документом, разобраться в приведенной выше схеме документа Word непросто. Но в процессе практического изучения той или иной возможности программы Microsoft Office Word мы рекомендуем возвращаться к нашей схеме и находить на ней элементы структуры документа, соответствующие изучаемой возможности. В конечном итоге, это поможет сформировать осмысленное и целостное восприятие документа Microsoft Office Word как формы представления компьютерного текста, удобной и наглядной в повседневной работе за компьютером.

[1] В главе 3 **Форматирование текста** будут подробно рассмотрены практические вопросы форматирования текста.