## Учреждение образования

## Белорусский государственный технологический университет

Кафедра полиграфического оборудования и

системы обработки информации

**ЛАБОРАТОРНАЯ РАБОТА № 3**

по дисциплине «Стандартизация и сертификация

информационных систем и технологий»

Тема

«Применение стандартов кодирования

текстовой информации»

Выполнила

Студентка 4к. 9 гр.

Хорова Э.И.

Проверил

Ст. преподаватель Сулим П.Е.

Минск 2021

Лабораторная работа № 3

**ПРИМЕНЕНИЕ СТАНДАРТОВ КОДИРОВАНИЯ**

**ТЕКСТОВОЙ ИНФОРМАЦИИ**

***Цель* *работы*:** Представление текстовой информации – разобраться со стандартами и ответить на вопросы.

|  |
| --- |
| **ASCII** — таблицы кодировок, в которых содержатся основные символы (английский алфавит, цифры, знаки препинания, символы национальных алфавитов(свои для каждого региона), служебные символы) и длина кода каждого символа бит. |

**бит:**

* **ASCII7** — первая кодировка, пригодная для работы с текстом. Помимо маленьких букв английского алфавита и служебных символов, содержит большие буквы английского языка, цифры, знаки препинания и другие символы.

**Кодировки стандарта ASCII ( бит):**

* **ASCII** — первая кодировка, в которой стало возможно использовать символы национальных алфавитов.
* **КОИ8-R** — первая русская кодировка. Символы кириллицы расположены не в алфавитном порядке. Их разместили в верхнюю половину таблицы так, чтобы позиции кириллических символов соответствовали их фонетическим аналогам в английском алфавите. Это значит, что даже при потере старшего бита каждого символа, например, при проходе через устаревший семибитный модем, текст остается "читаемым".
* **CP866** — русская кодировка, использовавшаяся на компьютерах IBM в системе DOS.
* **Windows-1251** — русская кодировка, использовавшаяся в русскоязычных версиях операционной системы Windows в начале 90-х годов. Кириллические символы идут в алфавитном порядке. Содержит все символы, встречающиеся в типографике обычного текста (кроме знака ударения).
  1. **Освоить понятие *Unicode* кодеки: *UTF*-8, *UTF*-16, *UTF*-32.**

**UTF-8** является юникод-кодировкой переменной длинны, с помощью которой можно представить любой символ юникода.  
  
Первым делом надо сказать, что структурной (атомарной) единицей этой кодировки является **байт**. То, что кодировка переменной длинны, значит, что один символ может быть закодирован разным количеством структурных единиц кодировки, то есть **разным количеством байтов**. Так, например, латиница кодируется одним байтом, а кириллица двумя байтами.

**UTF-16** также является кодировкой переменной длинны. Главное ее отличие от UTF-8 состоит в том что структурной единицей в ней является не один, а **два байта**. То есть в кодировке UTF-16 любой символ юникода может быть закодирован либо двумя, либо **четырьмя байтами**. Исходя из этого любой символ юникода в кодировке UTF-16 может быть закодирован либо одной кодовой парой, либо двумя.

Начнем с символов которые кодируются одной кодовой парой. Легко посчитать что таких символов может быть 65 535 (2в16), что полностью совпадает с базовым блоком юникода. Все символы находящиеся в этом блоке юникода в кодировке UTF-16 будут закодированы одной кодовой парой (двумя байтами), тут все просто.

***символ «o» (латиница) — 00000000 01101111***

***символ «M» (кириллица) — 00000100 00011100***

**UTF-32** в [информатике](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) – один из [способов кодирования](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D0%B1%D0%BE%D1%80_%D1%81%D0%B8%D0%BC%D0%B2%D0%BE%D0%BB%D0%BE%D0%B2) символов [Юникода](https://ru.wikipedia.org/wiki/%D0%AE%D0%BD%D0%B8%D0%BA%D0%BE%D0%B4), использующий для кодирования любого символа ровно **32 бита** – **4 байта**. Остальные кодировки, UTF-8 и UTF-16, используют для представления символов переменное число байтов. Символ UTF-32 является прямым представлением его кодовой позиции ([Code point](https://en.wikipedia.org/wiki/Code_point" \o "en:Code point) (англ.)[русск.](https://ru.wikipedia.org/w/index.php?title=Code_point&action=edit&redlink=1)).

Главное преимущество UTF-32 перед кодировками переменной длины заключается в том, что **символы Юникод непосредственно индексируемы**. Получение n-ой кодовой позиции является операцией, занимающей одинаковое время. Напротив, коды с переменной длиной требует последовательного доступа к n-ой кодовой позиции. Это делает замену символов в строках UTF-32 простой, для этого используется целое число в качестве индекса, как обычно делается для строк [ASCII](https://ru.wikipedia.org/wiki/ASCII).

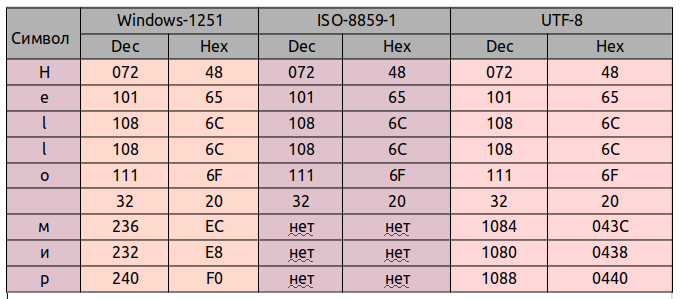
**Главный недостаток UTF-32 — это неэффективное использование пространства, так как для хранения символа используется четыре байта.** Символы, лежащие за пределами нулевой (базовой) плоскости кодового пространства, редко используются в большинстве текстов. Поэтому удвоение, в сравнении с UTF-16, занимаемого строками в UTF-32 пространства, не оправдано.

Хотя использование неменяющегося числа байтов на символ удобно, но не настолько, как кажется. Операция усечения строк реализуется легче в сравнении с UTF-8 и UTF-16. Но это не делает более быстрым нахождение конкретного смещения в строке, так как смещение может вычисляться и для кодировок фиксированного размера. Это не облегчает вычисление отображаемой ширины строки, за исключением ограниченного числа случаев, так как даже символ «фиксированной ширины» может быть получен комбинированием обычного символа с модифицирующим, который не имеет ширины. Например, буква «й» может быть получена из буквы «и» и [диакритического знака](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%BA%D1%80%D0%B8%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D0%B7%D0%BD%D0%B0%D0%BA%D0%B8) «[крючок над буквой](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B0%D1%82%D0%BA%D0%B0)». Сочетание таких знаков означает, что текстовые редакторы не могут рассматривать 32-битный код как единицу редактирования. Редакторы, которые ограничиваются работой с языками с письмом слева направо и [составными символами](https://ru.wikipedia.org/wiki/TrueType#.D0.A1.D0.BE.D1.81.D1.82.D0.B0.D0.B2.D0.BD.D0.BE.D0.B9_.D1.81.D0.B8.D0.BC.D0.B2.D0.BE.D0.BB) могут использовать символы фиксированного размера. Но такие редакторы вряд ли поддержат символы, лежащие за пределами нулевой (базовой) плоскости кодового пространства и вряд ли смогут работать одинаково хорошо с символами UTF-16.

* 1. **Привести примеры записей кодов символов**

**Utf-8**

Давайте возьмем символ «o» (англ). В таблице ASCII символов он находится на 111 позиции, в битовом виде это будет 01101111. В таблице юникода этот символ — U+006F что в битовом виде тоже будет 01101111.



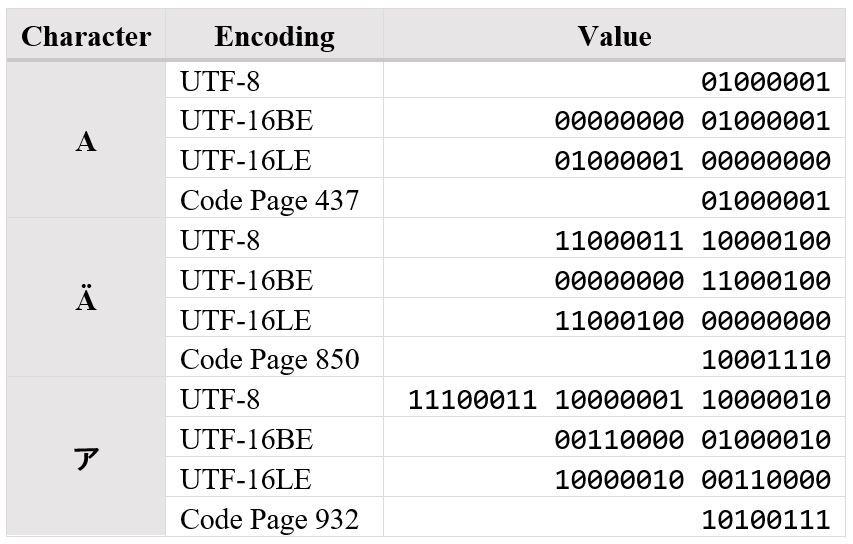
**Utf-16**

***символ «o» (латиница) — 00000000 01101111***

***символ «M» (кириллица) — 00000100 00011100***

**Сравнительная таблица**





* 1. **Разобраться в понятиях *code point*, *code unit*, *character set*, *coded character set*.**

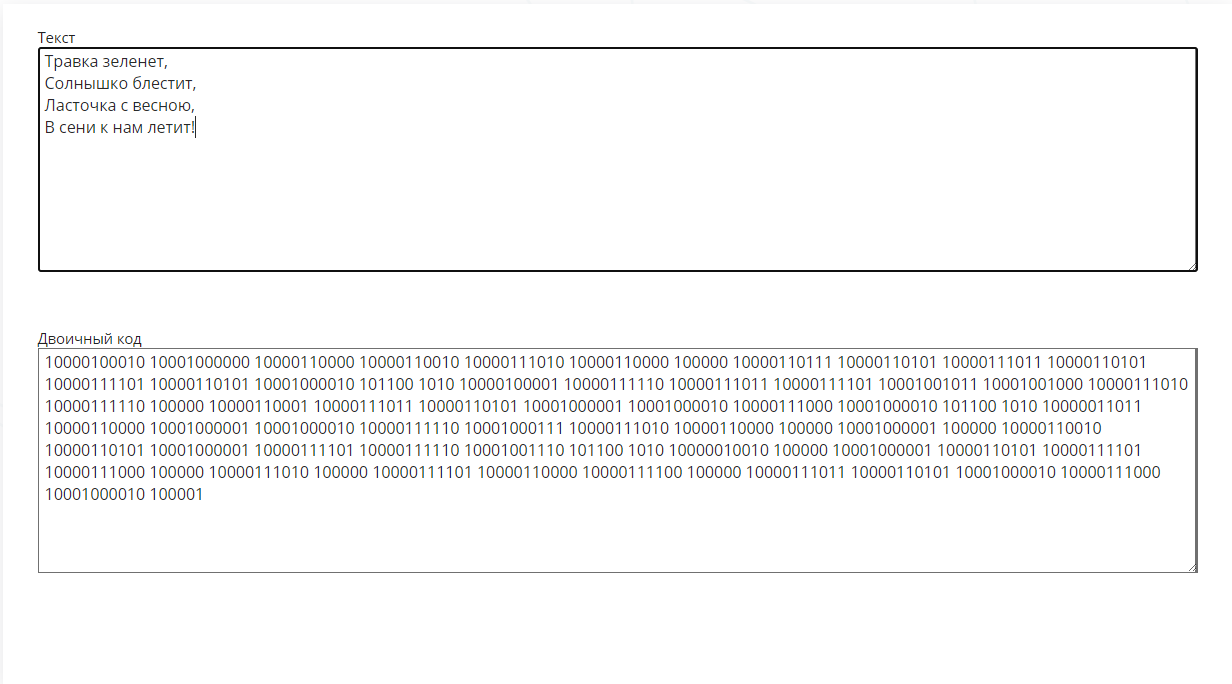
**code point** - это примерно то же, что мы привыкли называть символом. Но не совсем. Например, буква «ё» может быть как одним codepoint'ом, так и двумя – буквой «е» и символом "две точки над предыдущей буквой"

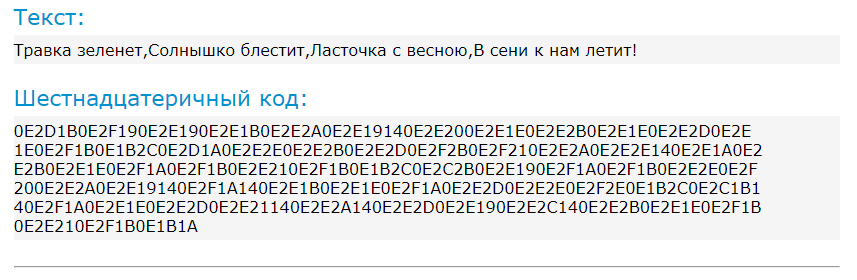
**code unit** - это единицы кодировки. Байт для utf-8, Слово (два байта) для unf-16 или Длинное слово (четыре байта) для utf-32.

***character set –*** это некий набор символов, называемых кодировкой. Разные CHARACTER SET включают в себя различные наборы символов.

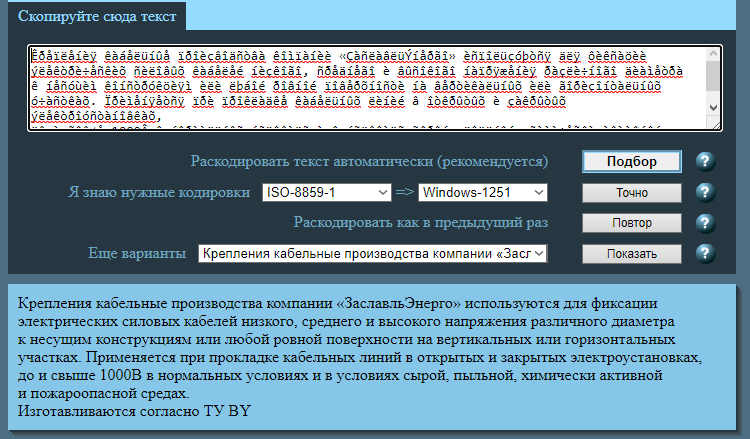
***coded character set –*** набор кодов символов – содержит привязку каждого символа из репертуара к целому неотрицательному числу, называемому кодовой точкой (*code point*)

* 1. **Привести примеры кодирования текстовой информации.**





Пример перекодировки текста:



**Выводы:** в ходе лабораторной работы ознакомились с юникодом и различными стандартами представления текстовой информации. Юникод — это очень большой и сложный мир, ведь стандарт позволяет ни много ни мало представлять и работать в компьютере со всеми основными письменностями мира. Некоторые системы письма существуют уже более тысячи лет, причём многие из них развивались почти независимо друг от друга в разных уголках мира. Люди так много всего придумали и оно зачастую настолько непохоже друг на друга, что объединить всё это в единый стандарт было крайне непростой и амбициозной задачей.  
Чтобы по-настоящему разобраться с Юникодом нужно хотя бы поверхностно представлять себе особенности всех письменностей, с которыми позволяет работать стандарт.