Реферат на тему: “Стандарты кодирования информации ASCII → UNICODE, {UTF-4, UTF-32}, таблицы, из чего состоят, какой, где применяется. Что такое “суррогатная” пара”

Введение

В эпоху цифровых технологий текст является основным способом передачи информации. Однако компьютеры оперируют не буквами, а числами (битами). Для того чтобы машина «понимала» человеческий язык, необходимы стандарты кодирования — системы соответствия между графическими символами и их числовыми кодами. Эволюция этих стандартов прошла долгий путь: от простых 7-битных таблиц для английского языка до универсальных систем, охватывающих все мировые письменности и даже эмодзи.

---

1. Стандарт ASCII: истоки цифрового текста

ASCII (American Standard Code for Information Interchange) был разработан в США в 1963 году. Он стал первым по-настоящему массовым стандартом.

- **Состав:** Оригинальная таблица ASCII использует **7 бит** для кодирования одного символа, что позволяет описать **128 позиций** (от 0 до 127).
    - **0–31:** Управляющие символы (например, перевод строки, табуляция).
    - **32–126:** Печатные символы (латиница, цифры, знаки препинания).
    - **127:** Символ удаления (DEL).
- **Ограничения:** Главный минус ASCII — отсутствие поддержки других языков, кроме английского. С появлением 8-битных систем (1 байт) возникли «расширенные» версии ASCII (например, Windows-1251 для кириллицы), где вторая половина таблицы (128–255) отдавалась под

национальные алфавиты. Это привело к проблеме «кракозябр» — несовместимости кодировок.

---

2. Unicode: единое пространство для всех

Чтобы решить проблему множества несовместимых таблиц, в 1991 году был создан консорциум **Unicode**. Его цель — присвоить каждому уникальному символу (букве, иероглифу, знаку) свое числовое значение, называемое **Code Point** (кодовая точка).

Unicode — это не кодировка сама по себе, а огромная таблица. Для записи этих чисел в память компьютера используются конкретные форматы преобразования — **UTF** (Unicode Transformation Format).

---

3. Сравнительный анализ: UTF-8 и UTF-32

UTF-8 (Самый популярный стандарт)

Это кодировка с **переменной длиной символа** (от 1 до 4 байт).

- **Структура:**
    - Символы из таблицы ASCII кодируются **1 байтом** (полная обратная совместимость).
    - Кириллица, греческий, арабский — **2 байта**.
    - Китайские/японские иероглифы — **3 байта**.
    - Эмодзи и редкие исторические знаки — **4 байта**.
- **Применение:** Интернет (98% сайтов), операционные системы Linux/macOS, текстовые файлы. Это «золотой стандарт» из-за экономии места при работе с латиницей.

UTF-32 (Максимальная точность)

Это кодировка с **фиксированной длиной символа** (**4 байта** или 32 бита на каждый символ).

- **Структура:** Каждой кодовой точке Unicode просто соответствует 32-битное число. Первые байты часто забиты нулями.

- **Плюсы:** Удобство индексации. Чтобы найти 100-й символ в строке, нужно просто отступить 400 байт. В UTF-8 для этого пришлось бы парсить всю строку с начала.

- **Минусы:** Огромный перерасход памяти (текст на английском будет весить в 4 раза больше, чем в ASCII/UTF-8).

- **Применение:** Внутренняя обработка строк в оперативной памяти некоторых систем и специализированных библиотеках, где важна скорость доступа к произвольному символу, а не размер данных.

---

4. Что такое «суррогатная пара»?

Понятие суррогатной пары относится исключительно к кодировке **UTF-16**, которая изначально задумывалась как фиксированная 16-битная (2 байта на символ). Однако символов в мире оказалось больше, чем 65 536 (2^{16}), и они перестали помещаться в 2 байта.

Суррогатная пара — это способ записи одного «тяжелого» символа (из так называемых дополнительных плоскостей, например, эмодзи 🚀) с помощью двух 16-битных кодов.

- **Как это работает:** В Unicode зарезервирован специальный диапазон кодов (U+D800 — U+DFFF), которые не соответствуют никаким буквам.

    - **High Surrogate** (старший суррогат): первый код в паре (U+D800 – U+DBFF).

    - **Low Surrogate** (младший суррогат): второй код в паре (U+DC00 – U+DFFF).

- Программа, видя код из диапазона суррогатов, понимает, что это не два отдельных знака, а «половинки» одного большого символа.

Заключение

Развитие стандартов кодирования прошло путь от жесткой экономии ресурсов (ASCII) до универсальности и гибкости (Unicode). **UTF-8** победил в вебе благодаря совместимости и компактности, **UTF-32** нашел нишу в сложных вычислениях, а существование **суррогатных пар** напоминает нам о трудностях перехода от старых 16-битных систем к современному многообразию символов. Сегодня Unicode позволяет пользователям по всему миру обмениваться информацией без страха увидеть нечитаемые символы.

Сводная таблица

Стандарт	Размер символа	Совместимость с ASCII	Основное применение
ASCII	7/8 бит	—	Исторические системы, низкоуровневое ПО
UTF-8	1–4 байта	Полная	Интернет, Linux, JSON, конфиги
UTF-16	2 или 4 байта	Нет	Windows API, Java, JavaScript (внутренняя память)
UTF-32	4 байта	Нет	Редкая внутренняя обработка данных