

Кодировки

Что будет выведено?

```
std::string s1 = "résumé";
```

```
fmt::print("{}.size() == {}\n", s1, s1.size());
```

Что будет выведено?

```
std::string s1 = "résumé";  
std::string s2 = "résumé";
```

```
fmt::print("{} == {}: {}\n", s1, s2, s1 == s2);
```

```
fmt::print("{} .size() == {}\n", s1, s1.size());
```

```
fmt::print("{} .size() == {}\n", s2, s2.size());
```

Что будет выведено?

```
std::string s1 = "résumé";  
std::string s2 = "résumé";
```

```
//  résumé == résumé: false  
fmt::print("{} == {}: {}\n", s1, s2, s1 == s2);  
  
//      résumé.size() == 8  
fmt::print("{} .size() == {}\n", s1, s1.size());  
  
//      résumé.size() == 10  
fmt::print("{} .size() == {}\n", s2, s2.size());
```

ASCII

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

ASCII

- ✓ 1 символ = 1 байт
 - ✓ Константная индексация
 - ✓ Пригодный для сортировки порядок
-
- ✗ Только для английского языка

ASCII

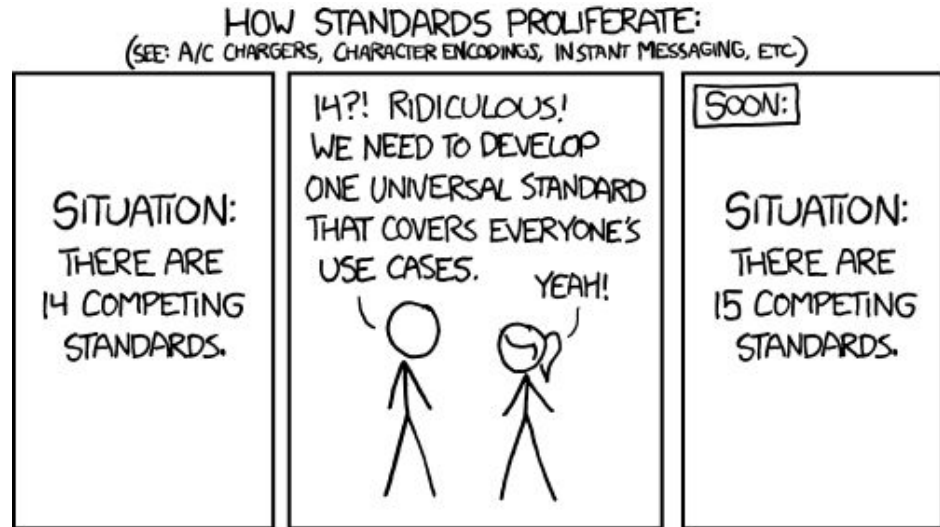
- Для символов 0–127 достаточно 7 бит
- В байте 8 бит (у нас)
- Свободен целый бит: x0000000
- Это еще аж 128 символов (128–255)
- И тут началось...

Code Pages

Только для кириллицы:

- KOI8-R
- cp866
- Windows-1251

И много для других...



KOI8-R																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8x	— 2500	 2502	┐ 250C	┌ 2510	└ 2514	┘ 2518	└ 251C	┐ 2524	┐ 252C	└ 2534	└ 253C	■ 2580	■ 2584	■ 2588	■ 258C	■ 2590
9x	░ 2591	▒ 2592	▓ 2593	∫ 2320	■ 25A0	● 2219	√ 221A	≈ 2248	≤ 2264	≥ 2265	NBSP 2321	Ј 2321	◦ 00B0	2 00B2	· 00B7	÷ 00F7
Ax	= 2550	 2551	ƒ 2552	ё 0451	┐ 2553	┘ 2554	┐ 2555	┐ 2556	┐ 2557	┐ 2558	┐ 2559	┐ 255A	┐ 255B	┐ 255C	┐ 255D	┐ 255E
Bx	 255F	 2560	 2561	Ё 0401	 2562	 2563	┐ 2564	┐ 2565	┐ 2566	┐ 2567	┐ 2568	┐ 2569	┐ 256A	┐ 256B	┐ 256C	© 00A9
Cx	Ю 044E	а 0430	б 0431	ц 0446	д 0434	е 0435	ф 0444	г 0433	х 0445	и 0438	й 0439	к 043A	л 043B	м 043C	н 043D	о 043E
Dx	п 043F	я 044F	р 0440	с 0441	т 0442	у 0443	ж 0436	в 0432	ь 044C	ы 044B	з 0437	ш 0448	э 044D	щ 0449	ч 0447	ъ 044A
Ex	Ю 042E	А 0410	Б 0411	Ц 0426	Д 0414	Е 0415	Ф 0424	Г 0413	Х 0425	И 0418	Й 0419	К 041A	Л 041B	М 041C	Н 041D	О 041E
Fx	П 041F	Я 042F	Р 0420	С 0421	Т 0422	У 0423	Ж 0416	В 0412	Ь 042C	Ы 042B	З 0417	Ш 0428	Э 042D	Щ 0429	Ч 0427	Ъ 042A









KOI8-R

`(KOI8 char) & 0b01111111 = (transliterated char)`

	В	о	п	р	о	с
koi8-r	194	238	239	240	238	241
-128	66	110	111	112	110	113
ASCII	W	O	P	R	O	S

KOI8-R

- KOI8 — Код обмена информацией, 8 bit
- Автор — Андрей Чернов

Code page 866																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8x	А 0410	Б 0411	В 0412	Г 0413	Д 0414	Е 0415	Ж 0416	З 0417	И 0418	Й 0419	К 041A	Л 041B	М 041C	Н 041D	О 041E	П 041F
9x	Р 0420	С 0421	Т 0422	У 0423	Ф 0424	Х 0425	Ц 0426	Ч 0427	Ш 0428	Щ 0429	Ъ 042A	Ы 042B	Ь 042C	Э 042D	Ю 042E	Я 042F
Ax	а 0430	б 0431	в 0432	г 0433	д 0434	е 0435	ж 0436	з 0437	и 0438	й 0439	к 043A	л 043B	м 043C	н 043D	о 043E	п 043F
Bx	 2591	 2592	 2593	┆ 2502	┆ 2524	┆ 2561	┆ 2562	┆ 2556	┆ 2555	┆ 2563	┆ 2551	┆ 2557	┆ 255D	┆ 255C	┆ 255B	┆ 2510
Cx	┆ 2514	┆ 2534	┆ 252C	┆ 251C	┆ 2500	┆ 253C	┆ 255E	┆ 255F	┆ 255A	┆ 2554	┆ 2569	┆ 2566	┆ 2560	= 2550	┆ 256C	┆ 2567
Dx	┆ 2568	┆ 2564	┆ 2565	┆ 2559	┆ 2558	┆ 2552	┆ 2553	┆ 256B	┆ 256A	┆ 2518	┆ 250C	 2588	 2584	 258C	 2590	 2580
Ex	р 0440	с 0441	т 0442	у 0443	ф 0444	х 0445	ц 0446	ч 0447	ш 0448	щ 0449	ъ 044A	ы 044B	ь 044C	э 044D	ю 044E	я 044F
Fx	Ё 0401	ё 0451	Є 0404	є 0454	Ї 0407	ї 0457	Ў 040E	ў 045E	° 00B0	• 2219	· 00B7	√ 221A	№ 2116	¤ 00A4	 25A0	NBSP

Windows CMD



```
Microsoft Windows [Version 10.0.19043.1889]
(c) Microsoft Corporation. All rights reserved.

C:\Users\evgeny>chcp
Active code page: 866

C:\Users\evgeny>
```

Windows-1251																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8x	Ђ	Ѓ	,	Ѕ	„	...	†	‡	€	‰	Љ	<	Њ	Ќ	Ѕ	Ц
9x	ђ	‘	’	“	”	•	–	—		™	љ	>	њ	ќ	ѕ	ц
Ax	␣	Ў	ў	Ј	Ѱ	Г	І	§	Ё	©	Є	«	¬	SHY	®	Ї
Bx	°	±	І	і	Г	μ	¶	·	ё	№	є	»	ј	Ѕ	ѕ	ї
Cx	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
Dx	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
Ex	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
Fx	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я

ASCII + Code Pages

- Это будет работать **всегда**
- Ведь **никто** не передает строки с одного ПК на другой
- И **никто** не говорит более чем на одном языке
- Тем более **не бывает** документов на >2 языках

cp1251 <-> koi8-r

cp1251	В	о	п	р	о	с
Код	194	238	239	240	238	241
koi8-r	б	н	о	п	н	я

cp1251 <-> koi8-r

```
$ echo 'Вопрос' \  
    | iconv -f utf8 -t cp1251 \  
    | iconv -f koi8r -t utf8
```

бНОПНЯ

cp1251 <-> koi8-r

```
$ echo 'Вопрос читал?' \
    | iconv -f utf8 -t cp1251 \
    | iconv -f koi8r -t utf8
```

бНОПНЯ ВХРЮК?

Wiki



Unicode

- Соберем все символы
- Вот совсем все
- Назначим каждому номер
- Определим семантику символов
- ???
- PROFIT!

Unicode Table

U+0061	a	Latin Small Letter A
U+0062	b	Latin Small Letter B
...		
U+0429	Щ	Cyrillic Capital Letter Shcha
U+2658		White Chess Knight

Пример

Hello

U+0048 U+0065 U+006C U+006C U+006F

Как представить это в памяти?

Пример

Hello

U+0048 U+0065 U+006C U+006C U+006F

00 48 00 65 00 6C 00 6C 00 6F

Пример

Hello

U+0048 U+0065 U+006C U+006C U+006F

00 48 00 65 00 6C 00 6C 00 6F

48 00 65 00 6C 00 6C 00 6F 00

Пример

Hello

U+0048 U+0065 U+006C U+006C U+006F

FE	FF	00	48	00	65	00	6C	00	6C	00	6F
FF	FE	48	00	65	00	6C	00	6C	00	6F	00

BOM – Byte Order Mark

BOM

Byte Order Mark

- 0xFEFF — Big Endian, Zero Width No-Break Space
- 0xFFFE — Little Endian, not a character
- Должен быть в начале Unicode¹ строки

¹ Только если кодировка допускает различный byte order.

UCS-2

Universal Coded Character Set

- Всегда 2 байта на символ
- Константная индексация
- 65535 символов хватит всем
- ???
- PROFIT?

UCS-2

Universal Coded Character Set

- Всегда 2 байта на символ
- Константная индексация
- 65536 символов хватит всем
- ???
- Нет, китайских иероглифов уже больше 80000.

UTF-16

- Может представить 1,112,064 code points
- 2 или 4 байта на символ
- Диапазон 0xD800 .. 0xDFFF для суррогатных пар

UTF-16: Суррогатные пары

Если `code_point > 0xFFFF`, `code_point -= 0x10000`

Старшие:

0b**110110**000000000000

0xD800 . . 0xDBFF

Младшие:

0b**110111**000000000000

0xDC00 . . 0xDFFF

Проблемы UTF-16

- Если `CodePoint(H) == 0x0048`, то `strlen("hello") == ?`
- Объем текстовых данных увеличивается в 2 раза
- Варианты UTF-16LE, UTF-16BE

UTF-8

1 0xxxxxxx

2 110xxxxx 10xxxxxx

3 1110xxxx 10xxxxxx 10xxxxxx

4 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

UTF-8

- Variable-length: от 1 до 4 байт
- ASCII-совместима
- Всегда однозначный byte order

Не нужен BOM, но MS считает иначе

UTF-8

```
strlen("Hello") == 5
```

```
strlen("Привет") == 12
```

UTF-32

- Всегда 4 байта
- Константная индексация

Промежуточный итог

Unicode — стандарт, определяет символы и операции

Кодировки:

- UCS-2
- UTF-16
- UTF-8
- UTF-32

Идея абстрактного символа

Щ

U+0449, Cyrillic Small Letter Shcha

Идея абстрактного символа

é

é U+00E9, Latin Small Letter E with Acute

или

e U+0065 Latin Small Letter E

´ U+0301, Combining Acute Accent

Лигатуры

ffi

U+FB03 Latin Small Ligature Ffi

или 3 символа

Возвращаемся к исходной проблеме

```
std::string s1 = "résumé";  
std::string s2 = "résumé";  
  
//  résumé == résumé: false  
fmt::print("{} == {}: {}\n", s1, s2, s1 == s2);  
  
//      résumé.size() == 8  
fmt::print("{} .size() == {}\n", s1, s1.size());  
  
//      résumé.size() == 10  
fmt::print("{} .size() == {}\n", s2, s2.size());
```


str.reverse

```
>>> 'résumé'[::-1]
```

```
'émuśer'
```

ICU — International Components for Unicode

- Свойства символов
- Локали
- Нормализация
- Часовые пояса
- Регулярные выражения
- ...

ICU: Collator

```
const icu::UnicodeString s1 = "résumé";  
const icu::UnicodeString s2 = "résumé";
```

```
UErrorCode error = U_ZERO_ERROR;
```

```
auto collator = std::unique_ptr<icu::Collator>(  
    icu::Collator::createInstance(error));
```

```
const bool result = collator->equals(s1, s2);  
std::cout << result << '\n'; // 1
```

Unicode Normalization

- NFD — Normalization Form Canonical **D**ecomposition
- NFC — Normalization Form Canonical **C**omposition

Unicode Normalization

```
const icu::UnicodeString s1 = "résumé";  
const icu::UnicodeString s2 = "résumé";
```

```
UErrorCode error = U_ZERO_ERROR;  
auto norm = icu::Normalizer2::getNFCInstance(error);  
const auto norm_s1 = norm->normalize(s1, error);  
const auto norm_s2 = norm->normalize(s2, error);
```

```
std::cout << norm_s1.length() << '\n'; // 6  
std::cout << norm_s2.length() << '\n'; // 6
```

Collation

Résumé

resume

1/143



Collation

```
const icu::UnicodeString s1 = "résumé";  
const icu::UnicodeString s2 = "resume";  
  
UErrorCode error = U_ZERO_ERROR;  
auto collator = std::unique_ptr<icu::Collator>(  
    icu::Collator::createInstance(error));  
  
collator->setStrength(icu::Collator::PRIMARY);  
  
bool result = collator->equals(s1, s2);  
std::cout << "cmp: " << result << '\n'; // 1
```

boost::locale

```
using PrimaryCmp = boost::locale::comparator<
    char, boost::locale::collator_base::primary>;

std::map<std::string, std::size_t, PrimaryCmp> words;
for (std::string s : {"résumé", "resume"}) {
    ++words[s];
}

for (const auto& [word, count] : words) {
    std::cout << word << ": " << count << '\n';
}
```


Рекомендации

- Выберите внутреннее представление строк
- Или узнайте нативное для своей платформы
- Linux: UTF-8
- Windows:
 - в API Unicode (UTF-16) + `wchar_t`,
 - Внутри — ваш выбор (utf-8).
- UTF-32 по необходимости

Windows & Unicode

```
#ifdef UNICODE
```

Wide

```
    #define SetWindowText SetWindowTextW
```

```
#else
```

```
    #define SetWindowText SetWindowTextA
```

```
#endif
```

ANSI

Windows & Unicode

- Unicode == UTF-16
- `wchar_t` — 2 байта
- A-функции выполняют перекодирование и вызывают W-функции
- Рекомендуется использовать W-функции

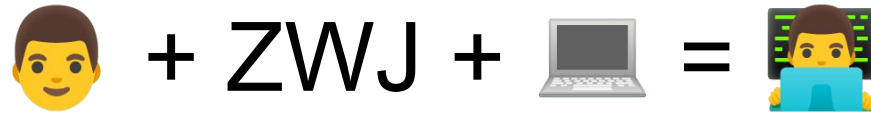
Главный вывод

- Plain Text не существует
- Указывайте кодировку

XML: `<?xml version="1.0" encoding="UTF-8"?>`

HTTP: Content-Type: text/html; charset=utf-8

 + ZWJ + 
U+1F468 + U+200D + U+1F4BB



U+1F468 + U+200D + U+1F4BB

ИСТОЧНИКИ

- [The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets \(No Excuses!\)](#)
- <http://utf8everywhere.org/>
- [MSDN: Working with Strings](#)
- <https://unicode-table.com/>