

# ЛЕКЦИЯ 9

- 04.10.2023
- Форматы представления данных
- Понятие символов, кодировок и кодирования символов
- Локализация и глобализация

# ЦЕЛИ ЛЕКЦИИ

- Познакомиться с форматами представления данных и форматами файлов
- Познакомиться с кодировками
- Начать разбираться с Юникодом
- Рассмотреть основы локализации и глобализации

(づ。◐\_◐。)づ

# ФОРМАТ ДАННЫХ

- **Формат данных** [data format] – совокупность правил представления и интерпретации данных в памяти компьютера, на внешних носителях, при операциях ввода/вывода и при передаче по каналам связи
  - формат определяет **способ кодирования данных** в некоторой знаковой системе
  - формат – это **описание физической структуры информационного элемента**

# ФОРМАТ ФАЙЛА

- **Формат файла** [file format] – способ записи и извлечения данных из файлов во внешней памяти
  - Например, в операционной системе Microsoft Windows формат файла ассоциирован с расширением файла (частью имени после последней точки)
  - Это именно ассоциация, т. е. смена расширения не влияет на данные, хранящиеся в файле, но затрудняет сопоставление файла со средствами его обработки



# ОБМЕН ДАННЫМИ

Конверторы

Типы данных

Сериализованные  
структурированные данные

Esc-форматированные  
(экранированные) строки

Символьное  
представление атомарных  
данных

Бинарные данные

# ОСНОВНЫЕ ФОРМАТЫ ДАННЫХ

(. \_.)

- **Целое число** [*Integer number*]
- **Вещественное число** [*Real number*]
- **Дата / Время** [Date / Time]
- **Строка** [String]

# ЭКСКУРС В ИСТОРИЮ КОДИРОВАНИЯ ТЕКСТОВОЙ ИНФОРМАЦИИ

**Текст** – последовательность символов некоторого алфавита



0100 0001

**Глиф** – конкретная фигура шрифта

Код



# ТАБЛИЦА КОДИРОВКИ ASCII

	.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

**Коды от 128 по 255** – в различных национальных кодировках одному и тому же коду соответствуют разные символы



# ВЫВОД СИМВОЛОВ ASCII

Что выведет этот код?

```
char ch = '\u0000';  
while(ch++ <= '\u007F')  
{  
    Console.Write(ch);  
}
```

```
char ch = (char)0;  
while (ch++ <= 127)  
{  
    Console.Write(ch);  
}
```

Выведет тоже самое, но такой способ хуже

0000000000

0000000000000000000000123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^\_`abcdefghijklmnopqrstuvwxyz{|}~?

# РАЗЛИЧНЫЕ КОДИРОВКИ

Символ	Windows	MS-DOS	КОИ-8	Mac	ISO	Unicode
А	192	128	225	128	176	1040
В	194	130	247	130	178	1042
М	204	140	237	140	188	1052
Э	221	157	252	157	205	1069
я	255	239	241	223	239	1103

Это неверно, т.к.  
кодировки  
Юникод нет

Тексты, созданные в одной кодировке, не будут корректно отображаться в другой!

# КОНСОРЦИУМ UNICODE

The standards body for the internationalization of software and services

Code Libraries (ICU)

Allow easy integration into operating system, applications, services, and more

Structured Data (CLDR)

Language-specific data for sorting, formatting, and processing of dates, times, numbers, currencies, and more


















Properties & Algorithms (UTC)

Provide character/emoji properties and behavior to ensure correct and consistent use across all platforms

Standard (UTC)

Encoding of nearly 150,000 characters covering the world's languages (and emoji!)

Юникод кодирует порядка 1,1 млн символов

 U+2640	 U+1F37E	 U+2332	 U+1F4A5	 U+53F8	 U+060F	 U+76CA	 U+FF8A
 U+1F923	 U+0298	 U+1F60A	 U+067B	 U+3006	 U+1F600	 U+D392	 U+FF89
 U+2666	Everyone in the world should be able to use their own language on phones and computers.						

Стандарт v.15.1 анонсирован 12.09.2023

# Unicode / Юникод

Кодовые точки  
[code points]

код

глиф

Plane

Basic Multilingual Plane,  
BMP

BMP code point

Формат трансформации  
[transformation format]

Способ хранения  
и размещения в  
памяти

UTF-8

UTF-16

Кодовая единица  
[code unit]

Byte Order Mark,  
BOM

Формат трансформации юникода  
[Unicode transformation format]

Представляемый  
пользователем символ  
[user-perceived character]

Начертание

Глиф, Шрифт

...

Закодированный символ  
[coded character]

U+1F428

Абстрактный символ  
[abstract character]

Раскодированный символ  
[encoded character]



# ПЛОСКОСТИ ЮНИКОДА

**Плоскость** [Plane] - непрерывный диапазон кодовых позиций Юникода

- Плоскостей 17, номера от 0 до 16

U+0000... U+FFFF

U+10000... U+1FFFF

...

U+100000... U+10FFFF

## Основная многоязыковая плоскость

[Basic Multilingual Plane, BMP] в диапазоне U+0000..U+FFFF

Decimal	Hex	Пример	Описание
10	U+000A	Н/Д	<a href="#">Перевод строки ↵</a>
97	U+0061	а	<a href="#">Латинская строчная буква A ↵</a>
562	U+0232	Ÿ	<a href="#">Латинская заглавная буква Y со знаком долготы ↵</a>
68 675	U+10C43	𐌵	<a href="#">Древнетюркская буква (язык орхоно-енисейских надписей) ↵</a>
127 801	U+1F339	🌹	<a href="#">Эмодзи "Роза" ↵</a>

# БАЗОВЫЕ ПОНЯТИЯ ЮНИКОДА

**Кодовая точка** [code point] – некоторое целочисленное значение в кодовом пространстве Юникода (от 0 до U+10FFFF)

буквы

эмодзи

символы

способы отображения

действия


**Кодовая единица** [code unit] – минимальная последовательность бит, позволяющая представить единицу раскодированного текста

U+3243F

UTF-8	UTF-16	UTF-32
f0 b2 90 bf	d889 dc3f	0003243f
4 кодовые ед	2 кодовые ед	1 кодовая ед

Как это будет сохранено в памяти?





Грубо говоря, нет никаких  
«символов» есть кластеры  
графем



〱(ツ)〱

**Кластер графем** [grapheme cluster] – последовательность закодированных символов,  
которая должна храниться совместно



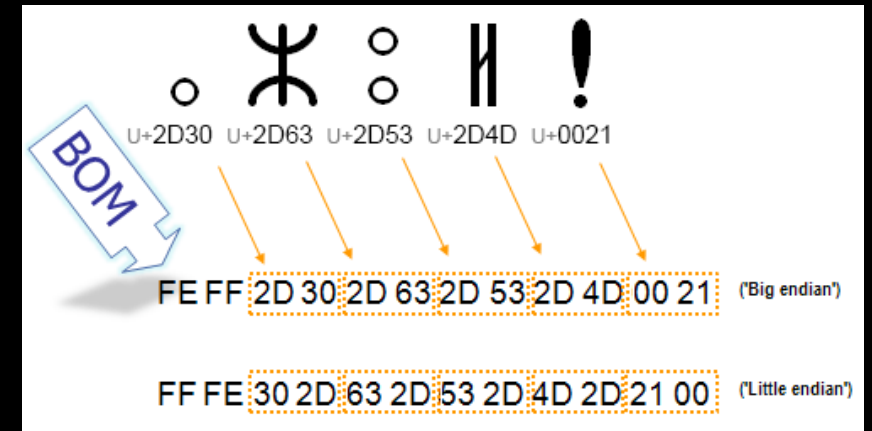
# BYTE ORDER MARK

**Маркер порядка байт** [*byte order mark (BOM)*] – код символа U+FEFF в начале потока данных для идентификации порядка байт в представлении символов и формата кодирования в рамках *UTF*

- Высокоуровневые протоколы (обмена данными) могут явно требовать или запрещать BOM в потоках данных

**UTF-8** не делает различий в порядке байт

BOM	Формат потока
00 00 FE FF	UTF-32, big-endian
FF FE 00 00	UTF-32, little-endian
FE FF	UTF-16, big-endian
FF FE	UTF-16, little-endian
EF BB BF	UTF-8

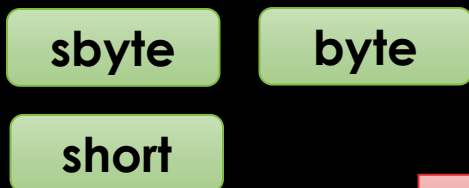


# СТРУКТУРА SYSTEM.CHAR

**char** – псевдоним для структуры System.Char .NET, служит для представления в программах символов, как кодовых точку Юникода с использованием кодировки UTF-16

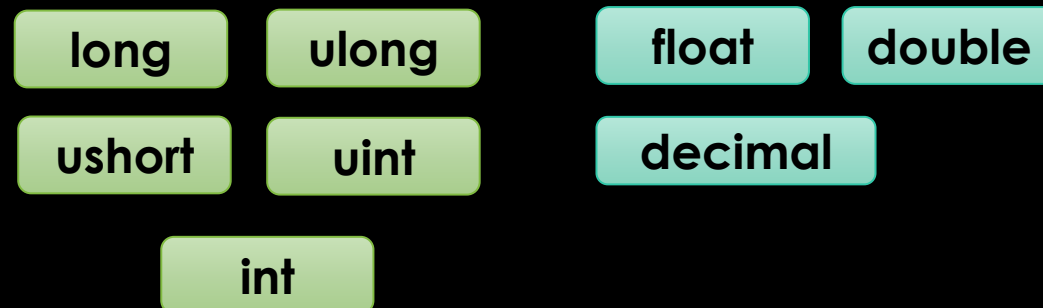
Тип	Диапазон	Размер	Тип .NET
char	От U+0000 до U+FFFF	16 разрядов	<u><a href="#">System.Char</a></u>

## Явные преобразования:



Неявных преобразований в char нет

## Неявные преобразования:



# НАЗНАЧЕНИЕ ЗНАЧЕНИЙ СИМВОЛАМ

символьный литерал

'F'

escape-последовательность  
Юникода

'\u006A'

шестнадцатеричная escape-  
последовательность

'\x006A'

# КАТЕГОРИИ СИМВОЛОВ

Lu	=	Letter, uppercase
Ll	=	Letter, lowercase
Lt	=	Letter, titlecase
Lm	=	Letter, modifier
Lo	=	Letter, other

Mn	=	Mark, nonspacing
Mc	=	Mark, spacing combining
Me	=	Mark, enclosing

Nd	=	Number, decimal digit
Nl	=	Number, letter
No	=	Number, other

Pc = Punctuation, connector

Pd = Punctuation, dash

Ps = Punctuation, open

Pe = Punctuation, close

Pi = Punctuation, initial quote (may behave like Ps or Pe depending on usage)

Pf = Punctuation, final quote (may behave like Ps or Pe depending on usage)

Po = Punctuation, other

Sm = Symbol, math

Sc = Symbol, currency

Sk = Symbol, modifier

So = Symbol, other

Zs = Separator, space

Zl = Separator, line

Zp = Separator, paragraph

Cc = Other, control

Cf = Other, format

Cs = Other, surrogate

Co = Other, private use

Cn = Other, not assigned (including noncharacters)

Основные  
категории  
Юникода (v. 15.0)  
<https://www.unicode.org/versions/Unicode15.1.0/ch04.pdf#G124142>

# ОПРЕДЕЛЕНИЕ КАТЕГОРИИ СИМВОЛА В C#

Перечисление, представляющее категорию символа Юникода

```
using System.Globalization;

char[] test = { 'A', 'b', '3', 'ë' };

UnicodeCategory uc = UnicodeCategory.UppercaseLetter;

foreach (char c in test)
{
    if(CharUnicodeInfo.GetUnicodeCategory(c) == uc)
    {
        Console.Write(c);
    }
}
```

Константа перечисления (конкретное значение) - 0

Статический метод, возвращающий тип перечисления  
**System.Globalization.UnicodeCategory**

Класс для извлечения данных о символе Юникода

# РАБОТА С КОДИРОВКАМИ





# КОДИРОВАНИЕ И ДЕКОДИРОВАНИЕ

**Кодирование** - это процесс преобразования набора символов Юникода в последовательность байтов

```
char[] test = { 'A', 'b', '3', 'ë' };
```

```
Encoding enUTF8 = Encoding.UTF8;  
PrintArray(test);
```

```
byte[] cTest = enUTF8.GetBytes(test);  
PrintArray(cTest);
```

**Декодирование** — это процесс преобразования последовательности закодированных байтов в набор символов Юникода

```
static void PrintArray(Array ar)  
{  
    foreach(var cur in ar)  
    {  
        Console.Write($"<{cur}>");  
    }    Console.WriteLine($"{{Environment.NewLine}}");  
}
```

<A><b><3><ë>

<65><98><51><209><145>



# КОДИРОВКИ – СВОЙСТВА КЛАССА ENCODING

Класс **System.Text.Encoding** – используется для представления кодировок символов

Имя	Описание
ASCII	Кодировка ASCII. Для представления символов используется 7 младших бит байта.
Default	Кодировка ANSI операционной системы. (Зависит от контекста, например, windows-1251)
Unicode	Кодировка UTF-16. Для представления символа латиницы используется 2 байта (... - до 4 байт)
UTF8	Кодировка UTF-8. Для представления символа латиницы используется 1 байт (кириллицы – 2 байта; ... до 4 байт)

# КОДИРОВАНИЕ И ДЕКОДИРОВАНИЕ

```
Encoding enUTF8 = Encoding.UTF8;  
PrintArray(test);
```

```
byte[] cTest = enUTF8.GetBytes(test);  
PrintArray(cTest);
```

<A><b><3><ë>

<65><98><51><209><145>

Это ё

```
Encoding enUTF7 = Encoding.UTF7;  
PrintArray(test);  
byte[] cTest = enUTF7.GetBytes(test);  
PrintArray(cTest);
```

```
char[] test = { 'A', 'b', '3', 'ë' };
```

```
static void PrintArray(Array ar)  
{  
    foreach(var cur in ar)  
    {  
        Console.Write($"<{cur}>");  
        Console.WriteLine($"{Environment.NewLine}");  
    }  
}
```

<A><b><3><ë>

<65><98><51><43><66><70><69><45>

И это ё

# СКОЛЬКО В БАЙТАХ?

```
char[] test = { 'A', 'b', '3', 'ë' };
```

<A><b><3><ë>

Bytes:: 6

<65><98><51><43><66><70><69><45>

```
static void PrintArray(Array ar)
{
    foreach(var cur in ar)
    {
        Console.Write($"<{cur}>");
    }
    Console.WriteLine($"{Environment.NewLine}");
}
```

```
Encoding enUTF7 = Encoding.UTF7;
Encoder enc = Encoding.UTF7.GetEncoder();
PrintArray(test);
Console.WriteLine($"Bytes:: {enc.GetByteCount(test, 0, test.Length, false)}");
byte[] cTest = enUTF7.GetBytes(test);
PrintArray(cTest);
```

# ПРЕОБРАЗУЕМ БАЙТЫ

```
char[] test = { 'A', 'b', '3', 'ë' };
```

```
string coding = "UTF-8";  
byte[] codes = new byte[test.Length];
```

```
foreach (char c in test)  
    Console.Write($"{c} ");
```

```
codes = Encoding.GetEncoding(coding).GetBytes(test);  
Console.WriteLine();
```

```
Console.Write($"{BitConverter.ToString(codes)} ");
```

Метод, возвращающий кодировку, связанную с указанным именем кодовой страницы

```
A b 3 ë  
41-62-33-D1-91
```

Как вывести результат в двоичном виде?

# ЕЩЕ РАЗ О РАЗНЫХ КОДИРОВКАХ

```
// Итог: 10 байтов.  
byte[] utf8Bytes = Encoding.UTF8.GetBytes("0-нуль");  
// Итог: 12 байтов.  
byte[] utf16Bytes = Encoding.Unicode.GetBytes("0-нуль");
```

```
Console.WriteLine(Encoding.UTF8.GetString(utf8Bytes));  
Console.Write(Encoding.Unicode.GetString(utf16Bytes));  
Console.WriteLine();
```

```
Console.WriteLine(Encoding.UTF8.GetString(utf16Bytes));  
Console.Write(Encoding.Unicode.GetString(utf8Bytes));
```

0-нуль  
0-нуль

0-=C;L  
?????

# ГЛОБАЛИЗАЦИЯ И ЛОКАЛИЗАЦИЯ



# ГЛОБАЛИЗАЦИЯ И ЛОКАЛИЗАЦИЯ

**Локализация** [localization] – адаптация приложения к национальным особенностям страны (региональные настройки)

**Глобализация** [globalization] – адаптация приложения к работе с разными языками и региональными настройками

**Региональные настройки (региональные стандарты / culture) определяют:**

- Язык
- Символ валюты
- Формат даты
- Формат вывода чисел (точка/запятая)

**В .Net предусмотрены:**

- механизм сателлитных сборок (satellite assemblies)
- классы из пространств имен:
  - System.Globalization
  - System.Resources
  - System.Text



# РЕГИОНАЛЬНЫЕ НАСТРОЙКИ (CULTURE)

**Региональные настройки** (culture) идентифицируются строкой, содержащей **главный** (primary) и **вспомогательный** (secondary) **тэги**, или Int32 кодом (LCID - Local Culture Identifier)

- Коды определены в стандарте Internet RFC 1766
- Имена “язык” – “страна/регион” определены в стандартах ISO (International Standards Organization)

## Группы настроек региональных стандартов:

- **Invariant** – не зависят от языка и страны, имя “” (код != 0)
- **Neutral** – определяют только язык, например, “ru”, “en” (два символа в нижнем регистре)
- **Specific** – определяют язык и страну/регион, например, “en-CA”, “en-GB”, “ru-RU”, “tt-RU”

# КЛАСС CULTUREINFO

System.Globalization

Текущие региональные настройки определяются значениями двух свойств выполняемого потока:

1. **Thread.CurrentCulture** – получает и задает язык и региональные параметры для текущего потока (формат даты / чисел / валюты)
2. **Thread.CurrentUICulture** – получает или задает текущие язык и региональные параметры, используемые диспетчером ресурсов для поиска ресурсов, связанных с языком и региональными параметрами, во время выполнения (загружаемые ресурсы)

**CultureInfo** – объект, представляющий язык и региональные параметры, используемые текущим потоком

# ВЫЗОВЫ TOSTRING() С ИСПОЛЬЗОВАНИЕМ CULTUREINFO

```
using System;
using System.Globalization;

// CultureInfo для английского языка в USA.
Console.WriteLine(100.ToString("c", new CultureInfo("en-US")));

// CultureInfo для России, форматы по умолчанию.
Console.WriteLine(100.ToString("c", new CultureInfo("ru-RU", false)));

// CultureInfo для России, форматы из установок пользователя.
Console.WriteLine(100.ToString("c", new CultureInfo("ru-RU", true)));
```

Как будет вести себя вывод этого кода?

# КЛАСС CULTUREINFO И НАСТРОЙКИ ПОТОКА

```
Console.WriteLine(Thread.CurrentThread.CurrentUICulture);
```

ru-RU

```
Thread.CurrentThread.CurrentUICulture = new CultureInfo("ja-JP");
```

```
Console.WriteLine(Thread.CurrentThread.CurrentUICulture);
```

ja-JP

# ЛОКАЛИЗАЦИЯ И ИНТЕРНАЦИОНАЛИЗАЦИЯ (1)

- Стандарты серии “**ISO 639** Language Codes”:
  - ISO 639-1:2002 Codes for the representation of names of languages — Part 1: **Alpha-2 code** (<http://www.iso.org/standard/22109.html>)
  - ISO 639-2:1998 Codes for the representation of names of languages — Part 2: **Alpha-3 code** (<http://www.iso.org/standard/4767.html>)
  - ISO 639-3:2007 Codes for the representation of names of languages — Part 3: Alpha-3 code for comprehensive coverage of languages (<http://www.iso.org/standard/39534.html>)
  - ISO 639-4:2010 Codes for the representation of names of languages — Part 4: General principles of coding of the representation of names of languages and related entities, and application guidelines (<http://www.iso.org/standard/39535.html>)
  - ISO 639-5:2008 Codes for the representation of names of languages — Part 5: Alpha-3 code for language families and groups (<http://www.iso.org/standard/39536.html>)
  - ISO 639-6:2009 Codes for the representation of names of languages — Part 6: **Alpha-4 code** for comprehensive coverage of language variants (<http://www.iso.org/standard/43380.html>)

# ЛОКАЛИЗАЦИЯ И ИНТЕРНАЦИОНАЛИЗАЦИЯ (2)

- Стандарты серии “**ISO 3166 Country Codes**”:
  - ISO 3166-1:2013 Codes for the representation of names of countries and their subdivisions — Part 1: **Country codes** (<http://www.iso.org/standard/63545.html>)
  - ISO 3166-2:2013 Codes for the representation of names of countries and their subdivisions — Part 2: **Country subdivision code** (<http://www.iso.org/standard/63546.html>)
  - ISO 3166-3:2013 Codes for the representation of names of countries and their subdivisions — Part 3: Code for formerly used names of countries (<http://www.iso.org/standard/63547.html>)
- Используйте платформу (OBP) (<http://www.iso.org/obp/ui/#search>)
  - Ищите “**Language Codes**” и “**Country codes**”!



# ЛОКАЛИЗАЦИЯ И ИНТЕРНАЦИОНАЛИЗАЦИЯ (3)

- Internationalization techniques: Authoring HTML & CSS (<http://www.w3.org/International/techniques/authoring-html>)
- Таблицы:
  - Codes for the Representation of Names of Languages ([http://www.loc.gov/standards/iso639-2/php/English\\_list.php](http://www.loc.gov/standards/iso639-2/php/English_list.php))
  - HTML **Language Code** Reference ([http://www.w3schools.com/tags/ref\\_language\\_codes.asp](http://www.w3schools.com/tags/ref_language_codes.asp))
    - Alpha-2 code from ISO 639-1
  - ISO **Country Codes** ([http://www.w3schools.com/tags/ref\\_country\\_codes.asp](http://www.w3schools.com/tags/ref_country_codes.asp))
    - The result is the language code and the country code through a hyphen:  
`html lang="en-US", html lang="ru-RU"`
  - ISO **Language Codes** (639-1 and 693-2) and IETF Language Types (<http://datahub.io/core/language-codes>)



# ИСПОЛЬЗОВАННАЯ ЛИТЕРАТУРА

- UNICODE (<http://www.unicode.org>)
- <https://www.unicode.org/glossary/>
- Unicode Character Table (<http://unicode-table.com>)
- A Tutorial on Data Representation Integers, Floating-point Numbers, and Characters (<https://www3.ntu.edu.sg/home/ehchua/programming/java/DataRepresentation.html>)
- <https://danielwertheim.se/utf-8-bom-adventures-in-c/>
- <https://learn.microsoft.com/ru-ru/dotnet/api/system.char?view=net-7.0>
- [UTF-8 Everywhere \(utf8everywhere.org\)](http://utf8everywhere.org)
- <https://learn.microsoft.com/ru-ru/dotnet/standard/base-types/character-encoding-introduction#unicode-code-points>
- <https://docs.microsoft.com/ru-ru/dotnet/api/system.globalization.cultureinfo?view=net-5.0>