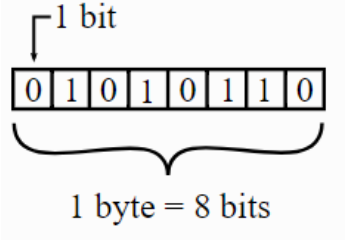
Algorithms and Data Structures for DS

Encoding
Information Units and Character Encoding

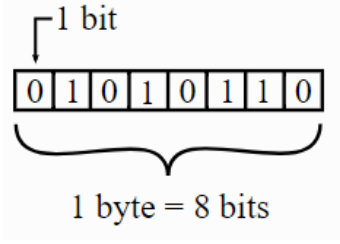


Learning goals

- Bits and bytes
- ASCII
- 8-Bit
- Unicode

BITS, BYTES, UNITS OF INFORMATION

- On computers, all data is strings of 0's and 1's, so base-2
- Bit: single 0/1 digit ("binary" + "digit")
- Byte: unit of digital information, typically 8 bits, 0-255, smallest adressable unit in mem
- Word: 16 bits (2 bytes), 0-65535
- Double Word: 32 bits (4 bytes), 0-4294967295
- Quad Word: 64 bits (8 bytes) 0 - ca. 1.8e19



HEXADECIMAL

- Base-16: uses **0–9, A–F** as digits
- 4 bits (nibble) = 1 hex digit
- Usually prefix hex string with **"0x"**
- Example: 1010 1101 = 0xAD = 173
- Easier to read and write large binary values
- Used to display memory addresses / content and bitwise ops

Binary	Hexadecimal	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

ASCII FOR CHARS AND TEXTS

- American Standard Code for Information Interchange
- 95 printable characters
- 7 bits are used

- Wikipedia:
  ASCII, abbreviated from American Standard Code for Information Interchange, is a character-encoding scheme. ASCII codes represent text in computers, communications equipment, and other devices that use text. Most modern character-encoding schemes are based on ASCII, though they support many additional characters. ASCII was the most common character encoding on the World Wide Web until December 2007, when it was surpassed by UTF-8, which includes ASCII as a subset.

ASCII TABLE

- First 32 are control chars
- E.g. LF is line feed

USASCII code chart

Bits b_7 b_6 b_5 →	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4 b_3 b_2 b_1 ↓ Row	0	1	2	3	4	5	6	7
0 0 0 0	0 NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1 SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2 STX	DC2	"	2	B	R	b	r
0 0 1 1	3 ETX	DC3	#	3	C	S	c	s
0 1 0 0	4 EOT	DC4	$	4	D	T	d	t
0 1 0 1	5 ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6 ACK	SYN	&	6	F	V	f	v
0 1 1 1	7 BEL	ETB	'	7	G	W	g	w
1 0 0 0	8 BS	CAN	(	8	H	X	h	x
1 0 0 1	9 HT	EM	)	9	I	Y	i	y
1 0 1 0	10 LF	SUB	*	:	J	Z	j	z
1 0 1 1	11 VT	ESC	+	;	K	[	k	{
1 1 0 0	12 FF	FS	,	<	L	\	l	|
1 1 0 1	13 CR	GS	-	=	M	]	m	}
1 1 1 0	14 SO	RS	.	>	N	^	n	~
1 1 1 1	15 SI	US	/	?	O	_	o	DEL

Source: ▸ Wikipedia 2024

ASCII AND BITS IN R

```
intToBits(65L)
## [1] 01 00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00
## [19] 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Bit order from left to right (in contrast to "usual" representation)

```
utf8ToInt("A")
## [1] 65

intToUtf8(65)
## [1] "A"
```

ASCII AND BITS IN R

```
coderange = c(32:126)
ascii.tab = data.frame(
  char = intToUtf8(coderange, multiple = TRUE),
  dec = coderange,
  hex = as.raw(coderange)
)

head(ascii.tab, 10)
## char dec hex
## 1 32 20
## 2 ! 33 21
## 3 " 34 22
## 4 # 35 23
## 5 $ 36 24
## 6 % 37 25
## 7 & 38 26
## 8 ’ 39 27
## 9 ( 40 28
## 10 ) 41 29
```

ASCII 8 BIT EXTENSIONS

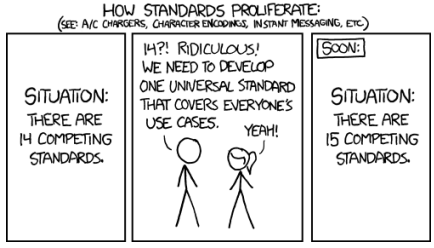
- ISO 8859 / Latin: family of standards for different language groups
- (1) Western Europe, (2) Eastern Europe, (3) Southern Europe, (4) Northern Europe, (5) Cyrillic, (6) Arabic, (7) Greek, ...
- Windows Code Page 1252: similar to ISO 8859; however, positions 128-159 are not dedicated to control characters, but also used for printable characters.
- Common problems: Sequence of bytes alone are not unique. One needs to know which code is used.
- Many Asian languages cannot be represented in 8 bits

UNICODE

- Unified code scheme for any language, maths, music, ...
- Abstract standard describes universal set of chars (like letters, symbols, emojis) and assigns each a unique integer “code point”
- Example: "A" has Unicode code point U+0041
- Up to 2^{32} characters, probably “only” 2^{21} will ever be used
- Human languages use the 2 bytes
- Defined and managed by unicode consortium, an NGO that all major HW and SW companies (Adobe, Apple, Google, Microsoft, Oracle, IBM, etc.) belong to
- Close cooperation with ISO; development of standards for character codes are delegated to unicode

UNICODE TRANSFORMATION FORMATS

- UTF = specific encoding of Unicode
- Describes how to represent Unicode code points as sequences of bytes
- Unfortunately, there are several UTFs
- UTF-8: most common scheme on Unix; Internet Mail Consortium (IMC) recommended that all e-mail clients should be able to handle mail using UTF-8, and W3C recommends UTF-8 as default encoding in XML and HTML
- UTF-16: older, internally used by many “early adopters” like Windows NT (2000, XP, Vista, 7), Java, or Mac OS X; not compatible with ASCII, since it uses 16 Bit



UTF-8

- UTF-8 has a variable width encoding to use 1-4 bytes

Bytes	Avail. Bits	Byte 1	Byte 2	Byte 3	Byte 4
1	7	0xxxxxxx			
2	11	110xxxxx	10xxxxxx		
3	16	1110xxxx	10xxxxxx	10xxxxxx	
4	21	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

- First bits in first byte determine the number of total bytes
- UTF-8 with 1 byte is compatible to ASCII

DATA TYPES IN C I

Type	Explanation	Size (bits)	Range
bool	Boolean type, added in C23.	1 (exact)	[false, true]
char	Smallest addressable unit of the machine that can contain the basic character set. It is an integer type. Actual type can be either signed or unsigned. It contains `CHAR_BIT` bits.	≥ 8	[CHAR_MIN, CHAR_MAX]
signed char	Of the same size as char, but guaranteed to be signed.	≥ 8	[-127, 127]
unsigned char	Of the same size as char, but guaranteed to be unsigned.	≥ 8	[0, 255]

Table: `https://en.wikipedia.org/wiki/C_data_types`

STRINGS IN C/C++, R AND PYTHON

- In C, strings are char-arrays
- End of string is marked by null terminator `'\0'`

- In C++, `std::string` class usually used
- More convenient and safer; automatically manages mem

- R and Python: no built-in type for single chars
- R: Has only `character` vecs, which are actually string vecs
- Python: Built in `str` type for strings, and arrays of strings