Komunikacja człowiek-komputer

Wykład 3

Dr inż. Michał Kruk

Reprezentacja znaków

- Aby zakodować tekst, trzeba każdej możliwej kombinacji bitów przyporządkować pewien znak
- Najpopularniejszym standardem kodowania znaków był kod ASCII
- Jeden znak w tym kodzie zajmuje 1 bajt (8 bitów)
- Obowiązuje od systemu DOS po dziś

Przykład

Zdanie

ZNAK	NUMER	NUMER ZAPISANY BINARNIE
A	65	0100 0001
l	108	0110 1100
a	97	0110 0001
(spacja)	32	0010 0000
m	109	0110 1101
a	97	0110 0001
(spacja)	32	0010 0000
k	107	0110 1011
o	111	0110 1111
t	116	0111 0100
a	97	0110 0001
.	46	0010 1110

Kod ASCII

- W latach sześćdziesiątych pojawił się standard ASCII
- Standard ASCII w praktyce używa 7 bitów
- Na 7 bitach można zakodować 128 znaków
- 0-31 specjalne kody sterujące
- Dalej cyfry, małe i duże litery alfabetu
- Na końcu wszystkie znaki znajdujące się na klawiaturze

Kod ASCII

- Pośród znaków ASCII wydzieloną grupę stanowią tzw. białe znaki (ang. *whitespace) albo* inaczej odstępy.
- Są one uznawane za znaki oddzielające pewne części tekstu.
- Należą do nich 4 znaki: spacja (kod 32), tabulacja (kod 9), koniec wiersza (kod 10) oraz powrót karetki (kod 13).

Znaki końca wiersza

- Dwa różne standardy
- W Windows koniec wiersza zaznacza się sekwencją dwóch znaków CR (kod 13) i LF (kod 10)
- W Linux natomiast samym znakiem LF (kod 10)
- Np. notatnik nie odczyta poprawnie znaków końca linii zapisanych w Linuxie. Wyświetli wszystko w jednej linii.

ASCII tablica znaków

Regular ASCII Chart (character codes 0 - 127)

000d	00h		(nul)	016d	10h	▶	(dle)	032d	20h	sp	048d	30h	0	064d	40h	@	080d	50h	P	096d	60h	`	112d	70h	p
001d	01h	☺	(soh)	017d	11h	◀	(dc1)	033d	21h	!	049d	31h	1	065d	41h	A	081d	51h	Q	097d	61h	a	113d	71h	q
002d	02h	☻	(stx)	018d	12h	↕	(dc2)	034d	22h	"	050d	32h	2	066d	42h	B	082d	52h	R	098d	62h	b	114d	72h	r
003d	03h	♥	(etx)	019d	13h	‼	(dc3)	035d	23h	#	051d	33h	3	067d	43h	C	083d	53h	S	099d	63h	c	115d	73h	s
004d	04h	♦	(eot)	020d	14h	¶	(dc4)	036d	24h	$	052d	34h	4	068d	44h	D	084d	54h	T	100d	64h	d	116d	74h	t
005d	05h	♣	(enq)	021d	15h	§	(nak)	037d	25h	%	053d	35h	5	069d	45h	E	085d	55h	U	101d	65h	e	117d	75h	u
006d	06h	♠	(ack)	022d	16h	▬	(syn)	038d	26h	&	054d	36h	6	070d	46h	F	086d	56h	V	102d	66h	f	118d	76h	v
007d	07h	•	(bel)	023d	17h	↨	(etb)	039d	27h	'	055d	37h	7	071d	47h	G	087d	57h	W	103d	67h	g	119d	77h	w
008d	08h	◘	(bs)	024d	18h	↑	(can)	040d	28h	(	056d	38h	8	072d	48h	H	088d	58h	X	104d	68h	h	120d	78h	x
009d	09h		(tab)	025d	19h	↓	(em)	041d	29h	)	057d	39h	9	073d	49h	I	089d	59h	Y	105d	69h	i	121d	79h	y
010d	0Ah		(lf)	026d	1Ah		(eof)	042d	2Ah	*	058d	3Ah	:	074d	4Ah	J	090d	5Ah	Z	106d	6Ah	j	122d	7Ah	z
011d	0Bh	♂	(vt)	027d	1Bh	←	(esc)	043d	2Bh	+	059d	3Bh	;	075d	4Bh	K	091d	5Bh	[	107d	6Bh	k	123d	7Bh	{
012d	0Ch	♀	(np)	028d	1Ch	∟	(fs)	044d	2Ch	,	060d	3Ch	<	076d	4Ch	L	092d	5Ch	\	108d	6Ch	l	124d	7Ch	|
013d	0Dh		(cr)	029d	1Dh	↔	(gs)	045d	2Dh	-	061d	3Dh	=	077d	4Dh	M	093d	5Dh	]	109d	6Dh	m	125d	7Dh	}
014d	0Eh	♫	(so)	030d	1Eh	▲	(rs)	046d	2Eh	.	062d	3Eh	>	078d	4Eh	N	094d	5Eh	^	110d	6Eh	n	126d	7Eh	~
015d	0Fh	☼	(si)	031d	1Fh	▼	(us)	047d	2Fh	/	063d	3Fh	?	079d	4Fh	O	095d	5Fh	_	111d	6Fh	o	127d	7Fh	⌂

Extended ASCII

- 8 bitów daje nam 256 znaków
- Za mało by zakodować znaki różnych alfabetów świata
- 7 bitów ASCII plus jeden dodatkowy bit
- dodatkowe 128 znaków powstałe po użyciu ósmego bitu nie jest ujednolicone
- wiele tzw. stron kodowych (ang. *codepage)*

ANSI i ISO

- Pierwszy z nich to wymyślony przez Microsoft i wykorzystywany w Windowsach ANSI (to jest nieoficjalna nazwa).
- Jego odpowiednikiem wykorzystywanym w innych jest ISO-8859.
- Choć oba mają zdefiniowane ponad dwieście pięćdziesiąt symboli, ta liczba jest cały czas niewystarczająca do pracy we wszystkich językach – dlatego każda rodzina języków posiadała własną wersję tych kodowań.

Extended ASCII

- Używanie różnych stron kodowych jest nieco problematyczne
- Dla samego języka polskiego powstało wiele stron kodowych
- W chwili obecnej najbardziej popularne są dwa:
  - ISO-8859-2 (Latin-2)
  - Windows-1250

ISO-8859-1 i Windows-1252

- Europa zachodnia

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	*Znaki kontrolne*															
1																
2	SP	!	„	#	$	%	&	’	(	)	*	+	,	-	.	/
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	{	|	}	~	
8	*Nie używane*															
9																
A	NBSP	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	SHY	®	¯
B	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	*Znaki kontrolne*															
1																
2X	SP	!	„	#	$	%	&	’	(	)	*	+	,	-	.	/
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	{	|	}	~	*ZK*
8	€	*NU*	‚	*ƒ*	„	…	†	‡	ˆ	‰	Š	‹	Œ	*NU*	Ž	*NU*
9	*NU*	‘	’	“	”	•	–	—	˜	™	š	›	œ	*NU*	ž	Ÿ
A	NBSP	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	SHY	®	¯
B	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

ANSI i ISO

- Do obsługi języków zachodnioeuropejskich jest wersja oznaczona jako Windows-1252 (lub CP1252). Dla środkowoeuropejskich istnieje wersja Windows-1250. Oprócz tego są jeszcze inne strony kodowe wydane przez Microsoft do obsługi pozostałych języków (np. 1251- cyrylica, 932- japoński, 1256- arabski).
- Podobnie rzecz ma się ze standardem ISO-8859. Dla języków Europy zachodniej istnieje strona kodowa oznaczona jako ISO-8859-1, dla środkowoeuropejskich ISO-8859-2, a kolejne numery na końcu to: Europa południowa, północna, cyrylica itd.

Windows-1250 (języki środkowoeuropejskie). Następna tabela to ISO dla centralnej i środkowej Europy – zwróć uwagę na inne rozmieszczenie polskich liter.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	Początek ten sam co w poprzednich, więc ciachnąłem.															
8	€	*NU*	‚	*NU*	„	…	†	‡	*NU*	‰	Š	‹	Ś	Ť	Ž	Ź
9	*NU*	‘	’	“	”	•	–	—	*NU*	™	š	›	ś	ť	ž	ź
A	NBSP	ˇ	˘	Ł	¤	Ą	¦	§	¨	©	Ş	«	¬	SHY	®	Ż
B	°	±	˛	ł	´	µ	¶	·	¸	ą	ş	»	Ľ	˝	ľ	ż
C	Ŕ	Á	Â	Ă	Ä	Ĺ	Ć	Ç	Č	É	Ę	Ë	Ě	Í	Î	Ď
D	Đ	Ń	Ň	Ó	Ô	Ő	Ö	×	Ř	Ů	Ú	Ű	Ü	Ý	Ţ	ß
E	ŕ	á	â	ă	ä	Í	ć	ç	č	é	ę	ë	ě	í	î	ď
F	đ	ń	ň	ó	ô	ő	ö	÷	ř	ů	ú	ű	ü	ý	ţ	˙

ISO-8859-2

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	Znowu przyciąłem.															
8	*Nieużywane*															
9																
A	NBSP	Ą	˘	Ł	¤	Ľ	Ś	§	¨	Š	Ş	Ť	Ź	SHY	Ž	Ż
B	°	ą	˛	ł	´	ľ	ś	ˇ	¸	š	ş	ť	ź	˝	ž	ż
C	Ŕ	Á	Â	Ă	Ä	Ĺ	Ć	Ç	Č	É	Ę	Ë	Ě	Í	Î	Ď
D	Đ	Ń	Ň	Ó	Ô	Ő	Ö	×	Ř	Ů	Ú	Ű	Ü	Ý	Ţ	ß
E	ŕ	á	â	ă	ä	Í	ć	ç	č	é	ę	ë	ě	í	î	ď
F	đ	ń	ň	ó	ô	ő	ö	÷	ř	ů	ú	ű	ü	ý	ţ	˙

UNICODE

- Potrzebny był nowy sposób kodowania znaków, który obejmowałby wszystkie możliwe symbole używane we wszystkich językach świata.
- Tak oto w 1991 oficjalnie został ogłoszony UNICODE.
- Początkowo zawierał on 7161 znaków, ale ta liczba ciągle rosła i obecnie wynosi 110 182. Dużo.

UNICODE

- Od jego wprowadzenia w jednym dokumencie można używać wielu języków, wpisywać skomplikowane formuły
- Do tego UNICODE jest obsługiwany przez wszystkie współczesne systemy i tak samo jak poprzednie – pierwszych 127 znaków jest identycznych z ASCII.
- Dzięki temu o wiele rzadziej występują problemy z krzaczkami.

UNICODE

- Pojawił się jednak inny problem – tak wielka liczba możliwych znaków sprawiła, że do ich zapisu przestał wystarczać 1 bajt.
- Jeden bajt składa się z ośmiu jedynek lub zer, co pozwala ułożyć je na 256 różnych sposobów. Czyli tyle, ile symboli mają strony kodowe.
- Żeby zwiększyć liczbę znaków obsługiwanych przez system kodowania, trzeba zwiększyć liczbę bajtów potrzebnych do wstawienia litery.

UNICODE

- Ostatnio dodany do unikodu znak, przedstawiający turecką Lirę ma symbol  (raczej nie jest jeszcze obsługiwany przez czcionki i programy).
- Numer przypisany do niego to 110 182.
- W systemie zero-jedynkowym (binarnym), do jego zapisania potrzebne są aż 3 bajty: 00000001 10101110 01100110.
- Gdyby teraz każda litera musiała zajmować po 3 bajty, nawet te z początku, którym wystarczy po jednym – zwiększyłoby się miejsce potrzebne do zapisania pliku na dysku.

UNICODE

- Trzeba było uniknąć sytuacji, w której liczba 65 powiązana z liter A była zapisana w sposób, marnujący pamięć komputera: 00000000 00000000 0100 0001.
- Programiści wymyślili więc, że znaki nie muszą mieć stałej liczby bajtów.
- Tak powstał UTF-8.
- I teraz bardzo ważna rzecz: **UNICODE to tylko tabelka z wypisanymi wszystkimi znakami i ich numerkami**. A za to, żeby działało to na komputerze, odpowiadają kodowania. Jest wspomniany już UTF-8, UTF-16 i UTF-32

Kodowanie polskich znaków diakrytycznych w różnych standardach:

Standard	Ą	Ć	Ę	Ł	Ń	Ó	Ś	Ź	Ż	ą	ć	ę	ł	ń	ó	ś	ź	ż
ISO-8859-2 (ISO Latin-2, PN-93 T-42118)	161	198	202	163	209	211	166	172	175	177	230	234	179	241	243	182	188	191
	#A1	#C6	#CA	#A3	#D1	#D3	#A6	#AC	#AF	#B1	#E6	#EA	#B3	#F1	#F3	#B6	#BC	#BF
Windows-1250 (Windows CE)	165	198	202	163	209	211	140	143	175	185	230	234	179	241	243	156	159	191
	#A5	#C6	#CA	#A3	#D1	#D3	#8C	#8F	#AF	#B9	#E6	#EA	#B3	#F1	#F3	#9C	#9F	#BF
CP852 (DOS Latin II)	164	143	168	157	227	224	151	141	189	165	134	169	136	228	162	152	171	190
	#A4	#8F	#A8	#9D	#E3	#E0	#97	#8D	#BD	#A5	#86	#A9	#88	#E4	#A2	#98	#AB	#BE
Mac OS Central Europe	132	140	162	252	193	238	229	143	251	136	141	171	184	196	151	230	144	253
	#84	#8C	#A2	#FC	#C1	#EE	#E5	#8F	#FB	#88	#8D	#AB	#B8	#C4	#97	#E6	#90	#FD
Mazovia	143	149	144	156	165	163	152	160	161	134	141	145	146	164	162	158	166	167
	#8F	#95	#90	#9C	#A5	#A3	#98	#A0	#A1	#86	#8D	#91	#92	#A4	#A2	#9E	#A6	#A7
bazowe litery łacińskie	65	67	69	76	78	79	83	90	90	97	99	101	108	110	111	115	122	122
	#41	#43	#45	#4C	#4E	#4F	#53	#5A	#5A	#61	#63	#65	#6C	#6E	#6F	#73	#7A	#7A
Unicode	260	262	280	321	323	211	346	377	379	261	263	281	322	324	243	347	378	380
	#104	#106	#118	#141	#143	#D3	#15A	#179	#17B	#105	#107	#119	#142	#144	#F3	#15B	#17A	#17C
Unicode w transformacji UTF-8	#C4 #84	#C4 #86	#C4 #98	#C5 #81	#C5 #83	#C3 #93	#C5 #9A	#C5 #B9	#C5 #BB	#C4 #85	#C4 #87	#C4 #99	#C5 #82	#C5 #84	#C3 #B3	#C5 #9B	#C5 #BA	#C5 #BC
ISO 8859-13 (ISO Latin-7)	192	195	198	217	209	211	218	202	221	224	227	230	249	241	243	250	234	253
	#C0	#C3	#C6	#D9	#D1	#D3	#DA	#CA	#DD	#E0	#E3	#E6	#F9	#F1	#F3	#FA	#EA	#FD
ISO 8859-16 (ISO Latin-10)	161	197	221	163	209	211	215	172	175	162	229	253	179	241	243	247	174	191
	#A1	#C5	#DD	#A3	#D1	#D3	#D7	#AC	#AF	#A2	#E5	#FD	#B3	#F1	#F6	#F7	#AE	#BF
Windows-1257 (Windows Baltic)	192	195	198	217	209	211	218	202	221	224	227	230	249	241	243	250	234	253
	#C0	#C3	#C6	#D9	#D1	#D3	#DA	#CA	#DD	#E0	#E3	#E6	#F9	#F1	#F3	#FA	#EA	#FD
CP775 (DOS Baltic)	181	128	183	173	227	224	151	141	163	208	135	210	136	231	162	152	165	164
	#B5	#80	#B7	#AD	#E0	#E3	#97	#8D	#A3	#D0	#87	#D2	#88	#E7	#A2	#98	#A5	#A4

UTF-8

- Używa od 8 do 32 bitów
- Każdy tekst w ASCII jest tekstem w UTF-8.
- Żaden znak spoza ASCII nie zawiera bajtu z ASCII.
- Typowy tekst ISO-Latin-X rozrasta się w bardzo niewielkim stopniu po przekonwertowaniu do UTF-8.
- Nie zawiera bajtów 0xFF i 0xFE, więc łatwo można go odróżnić od tekstu UTF-16.
- Znaki o kodzie różnym od 0 nie zawierają bajtu 0, co pozwala stosować UTF-8 w ciągach zakończonych zerem.
- O każdym bajcie wiadomo, czy jest początkiem znaku, czy też leży w jego środku
- Nie ma problemów z little endian vs. big endian.
- Jest domyślnym kodowaniem w XML
- Znaki CJK zajmują po 3 bajty zamiast 2 w kodowaniach narodowych.
- Znaki alfabetów niełacińskich zajmują po 2 bajty zamiast jednego w kodowaniach narodowych.
- UTF-8 nie używa przesunięć zasięgów, co stanowi dodatkowe utrudnienie dla implementacji UTF-8

UTF-8

Mapowanie znaków Unicode na ciągi bajtów:

0x00 do 0x7F – bity 0xxxxxxx, gdzie kolejne „x” to bity – licząc od najwyższego
0x80 do 0x7FF – bity 110xxxxx 10xxxxxx
0x800 do 0xFFFF – bity 1110xxxx 10xxxxxx 10xxxxxx
0x10000 do 0x1FFFFF – bity 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
0x200000 do 0x3FFFFFF – bity 111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx
0x4000000 do 0x7FFFFFFF – bity 1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx

- Znaki z przedziału ASCII (0 do 127) kodowane są jako jeden bajt, czyli m.in. litery alfabetu łacińskiego.
- Polskie znaki diakrytyczne kodowane już są jako dwa bajty.
- W listopadzie 2003 roku kodowanie UTF-8 zostało ograniczone zgodnie z RFC 3629 do 0x10FFFF pozycji, w celu zapewnienia zgodności z ograniczeniami systemu UTF-16.
- Rezultatem jest usunięcie wszystkich sekwencji złożonych z 5 i 6 bajtów oraz około połowy sekwencji 4-bajtowych.

Przykład

- Ą – w UNICODE 260
- 260d = 100000100
- A więc 2 bajty
- 0x80 do 0x7FF – bity 110xxxxx 10xxxxxx
- 110xxxxx 10xxxxxx – 100000100
- 11000100 10000100
- |1100|0100| 1000|0100|
- C 4 8 4

BOM

- **BOM** (ang. ***B**yte **O**rder **M**ark*) – **znacznik kolejności bajtów**

Znaczniki BOM zależnie od kodowania

kodowanie	BOM dla Little-endian	BOM dla Big-endian
UTF-16	`0xff 0xfe`	`0xfe 0xff`
UTF-32	`0xff 0xfe 0x00 0x00`	`0x00 0x00 0xfe 0xff`
UTF-8	`0xef 0xbb 0xbf` (Kolejność bajtów - nie dotyczy)	

Rady praktyczne

- W notatniku kodowanie ANSI to Windows-1250 jeżeli używamy polskiego systemu operacyjnego
- UWAGA! W edytorach on-line (np. w popularnych hostigach) nie ma wyboru kodowania. Bardzo często zmieniają one same kodowanie na ISO!!
- W chwili obecnej powinniśmy uzywać wyłącznie kodowania UTF-8

Tablica unicode

- http://www.tamasoft.co.jp/en/general-info/unicode.html