

Informatyka 1 (ES1E2009)

Politechnika Białostocka - Wydział Elektryczny
Elektrotechnika, semestr II, studia stacjonarne I stopnia
Rok akademicki 2020/2021

Wykład nr 4 (15.03.2021)

dr inż. Jarosław Forenc

Plan wykładu nr 4

- Język C - pętle
 - pętle while i do...while
- Jednostki informacji cyfrowej
 - bit, bajt słowo, FLOPS
- Kodowanie znaków
 - ASCII, ISO/IEC 646, ISO 8859
 - EBCDIC, Windows-1250, Unicode
- Kodowanie liczb
 - NKB, BCD, kod 2 z 5, kod Graya

Przykład: pierwiastek kwadratowy

```
#include <stdio.h>
#include <math.h>

int main(void)
{
    float x, y;

    printf("Podaj liczbe: ");
    scanf("%f", &x);

    if (x>=0)
    {
        y = sqrt(x);
        printf("Pierwiastek liczby: %f\n", y);
    }
    else
        printf("Blad! Liczba ujemna\n");

    return 0;
}
```

Podaj liczbe: -3
Blad! Liczba ujemna

Podaj liczbe: 3
Pierwiastek liczby: 1.732051

Przykład: pierwiastek kwadratowy (pętla while)

```
#include <stdio.h>
#include <math.h>

int main(void)
{
    float x, y;

    printf("Podaj liczbe: ");
    scanf("%f", &x);
    while (x<0)
    {
        printf("Blad! Liczba ujemna\n\n");
        printf("Podaj liczbe: ");
        scanf("%f", &x);
    }
    y = sqrt(x);
    printf("Pierwiastek liczby: %f\n", y);

    return 0;
}
```

Podaj liczbe: -3
Blad! Liczba ujemna

Podaj liczbe: -5
Blad! Liczba ujemna

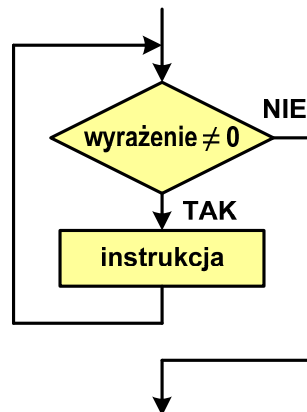
Podaj liczbe: 3
Pierwiastek liczby: 1.732051

Język C - pętla while

```
while (wyrażenie)  
instrukcja
```

- „dopóki wyrażenie w nawiasach jest prawdziwe wykonuj instrukcję”

- Wyrażenie w nawiasach:
 - prawdziwe** - gdy jego wartość jest różna od zera
 - falszywe** - gdy jego wartość jest równa zero
- Jako wyrażenie najczęściej stosowane jest **wyrażenie logiczne**



Język C - pętla while

```
while (wyrażenie)  
instrukcja
```

- Instrukcja:
 - prosta** - jedna instrukcja zakończona średnikiem
 - złożona** - jedna lub kilka instrukcji objętych nawiasami klamrowymi

```
int x = 10;  
while (x>0)  
    x = x - 1;
```

```
int x = 10;  
while (x>0)  
{  
    printf("%d\n", x);  
    x = x - 1;  
}
```

Przykład: suma liczb dodatnich

```
#include <stdio.h>  
  
int main(void)  
{  
    int x, suma = 0;  
  
    printf("Podaj liczbe: ");  
    scanf("%d", &x);  
  
    while (x>0)  
    {  
        suma = suma + x;  
        printf("Podaj liczbe: ");  
        scanf("%d", &x);  
    }  
  
    printf("Suma liczb: %d\n", suma);  
  
    return 0;  
}
```

```
Podaj liczbe: 4  
Podaj liczbe: 8  
Podaj liczbe: 2  
Podaj liczbe: 3  
Podaj liczbe: 5  
Podaj liczbe: -2  
Suma liczb: 22
```

Język C - pętla while

- Program pokazany na poprzednim slajdzie zawiera typowy schemat przetwarzania danych z wykorzystaniem pętli **while**

```
printf("Podaj liczbe: ");  
scanf("%d", &x);
```

wczytanie danych

```
while (x>0)  
{
```

```
    suma = suma + x;
```

operacje na danych

```
    printf("Podaj liczbe: ");  
    scanf("%d", &x);  
}
```

wczytanie danych

- Dane mogą być wczytywane z klawiatury, pliku, itp.

Język C - pętla while (break, continue)

- **break** i **continue** są to instrukcje skoku

```
int x=0;
while (x<10)
{
    x++;
    if (x%2==0)
        continue;
    if (x%5==0)
        break;
    printf("%d\n", x);
}
```

□ **continue** przerywa bieżącą iterację

□ **break** przerywa wykonywanie pętli

Język C - pętla while (najczęstsze błędy)

- Postawienie średnika po wyrażeniu w nawiasach powoduje powstanie pętli nieskończonej - program zatrzymuje się na pętli

```
int x = 10;
while (x>0);
    printf("%d ", x--);
```



- Brak aktualizacji zmiennej powoduje także powstanie pętli nieskończonej - program wyświetla wielokrotnie tę samą wartość

```
int x = 10;
while (x>0)
    printf("%d ", x);
```

10 10 10 10 10 ...

Język C - pętla while (pętla nieskończona)

- W pewnych sytuacjach celowo stosuje się pętlę nieskończoną (np. w mikrokontrolerach)

```
while (1)
{
    instrukcja
    instrukcja
    ...
}
```

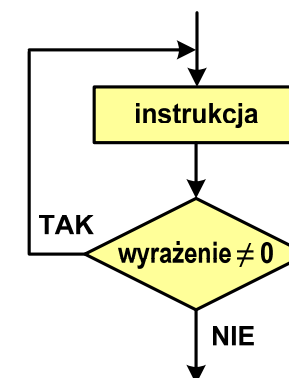
- W układach mikroprocesorowych program działa aż do wyłączenia zasilania

Język C - pętla do ... while

```
do
    instrukcja
while (wyrażenie);
```

- „wykonuj instrukcję dopóki wyrażenie w nawiasach jest prawdziwe”

- Wyrażenie w nawiasach:
 - **prawdziwe** - gdy jego wartość jest różna od zera
 - **falsywe** - gdy jego wartość jest równa zero



Język C - pętla do ... while

```
do
    instrukcja
while (wyrażenie);
```

- Instrukcja:
 - **prosta** - jedna instrukcja zakończona średnikiem
 - **złożona** - jedna lub kilka instrukcji objętych nawiasami klamrowymi

```
int x = 10;
do
    x = x - 1;
while (x>0);
```

```
int x = 10;
do
{
    printf("%d\n", x);
    x = x - 1;
}
while (x>0);
```

Język C - pętla do ... while (break, continue)

- **break** i **continue** są to instrukcje skoku

```
int x=0;
do
{
    x++;
    if (x%5==0)
        break;
    if (x%2==0)
        continue;
    printf("%d\n", x);
}
while (i<10);
```

- **break** przerywa wykonywanie pętli
- **continue** przerywa bieżącą iterację

Przykład: suma liczb < 100

```
#include <stdio.h>
int main(void)
{
    int x, suma = 0;
    do
    {
        printf("Podaj liczbę: ");
        scanf("%d", &x);
        suma = suma + x;
    }
    while (suma<100);
    printf("Suma liczb: %d\n", suma);
    return 0;
}
```

```
Podaj liczbę: 34
Podaj liczbę: 9
Podaj liczbę: 26
Podaj liczbę: -8
Podaj liczbę: 67
Suma liczb: 128
```

Jednostki informacji - bit

- **Bit** (ang. **binary digit**) - podstawowa jednostka informacji stosowana w informatyce i telekomunikacji
- Określa najmniejszą ilość informacji potrzebną do stwierdzenia, który z dwóch możliwych stanów przyjął układ
- Bit przyjmuje jedną z dwóch wartości:
 - 0 (zero)
 - 1 (jeden)
- Bit jest tożsamy z cyfrą w systemie dwójkowym
- Oznaczenia bitów:
 - standard IEEE 1541 (2002) - mała litera „b”
 - standard IEC 60027 - „bit”



Jednostki informacji - bit

- Wielokrotności bitów:

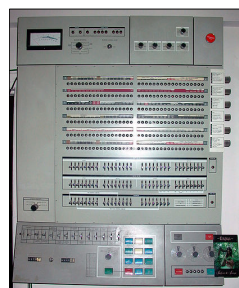
Przedrostki dziesiętne (układ SI)		
Nazwa	Symbol	Mnożnik
bit	b	---
kilobit	kb	$10^3 = 1000^1$
megabit	Mb	$10^6 = 1000^2$
gigabit	Gb	$10^9 = 1000^3$
terabit	Tb	$10^{12} = 1000^4$
petabit	Pb	$10^{15} = 1000^5$
eksabit	Eb	$10^{18} = 1000^6$
zettabit	Zb	$10^{21} = 1000^7$
jottabit	Yb	$10^{24} = 1000^8$

Przedrostki binarne (IEC 60027-2)		
Nazwa	Symbol	Mnożnik
bit	b	---
kibibit	Kib	$2^{10} = 1024^1$
mebibit	Mib	$2^{20} = 1024^2$
gibibit	Gib	$2^{30} = 1024^3$
tebibit	Tib	$2^{40} = 1024^4$
pebibit	Pib	$2^{50} = 1024^5$
eksbibit	Eib	$2^{60} = 1024^6$
zebibit	Zib	$2^{70} = 1024^7$
jobibit	Yib	$2^{80} = 1024^8$

- Przedrostki binarne** - wprowadzone w 1999 roku w celu odróżnienia przedrostków o mnożniku 1000 (10^3) od przedrostków o mnożniku 1024 (2^{10})

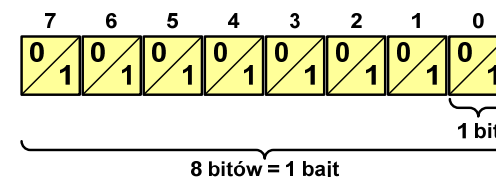
Jednostki informacji - bajt

- W pierwszych komputerach bajt mógł mieć inną liczbę bitów: 4, 6, 7, 9, 12
- 8-bitowy bajt:
 - koniec 1956 r. - pierwsze zastosowanie
 - 1964 r. - uznanie za standard (IBM System/360)
- Inna nazwa 8-bitowego bajtu - **oktet**
- Najczęściej stosowanym skrótem dla bajtu jest wielka litera „B”
 - „B” używane jest także do oznaczania **bela** - jednostki miary wielkości ilorazowych
 - zamiast bela częściej używa się jednostki podwielokrotnej - **decybel** (dB) więc nie ma problemu z rozróżnieniem obu jednostek



Jednostki informacji - bajt

- Bajt** (ang. byte) - najmniejsza adresowalna jednostka informacji pamięci komputerowej składająca się z bitów
- W praktyce przyjmuje się, że jeden bajt to 8 bitów

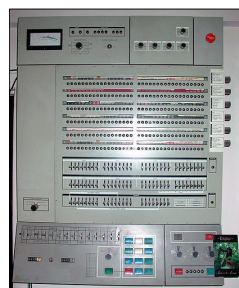


- Za pomocą jednego bajtu można zapisać $2^8 = 256$ różnych wartości:

0000 0000	→	0
0000 0001	→	1	1111 1101	→ 253
0000 0010	→	2	1111 1110	→ 254
...	1111 1111	→ 255

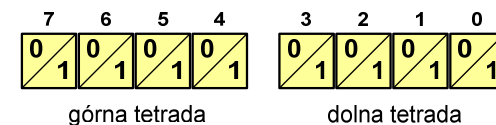
Jednostki informacji - bajt

- W pierwszych komputerach bajt mógł mieć inną liczbę bitów: 4, 6, 7, 9, 12
- 8-bitowy bajt:
 - koniec 1956 r. - pierwsze zastosowanie
 - 1964 r. - uznanie za standard (IBM System/360)
- Inna nazwa 8-bitowego bajtu - **oktet**
- Najczęściej stosowanym skrótem dla bajtu jest wielka litera „B”
 - „B” używane jest także do oznaczania **bela** - jednostki miary wielkości ilorazowych
 - zamiast bela częściej używa się jednostki podwielokrotnej - **decybel** (dB) więc nie ma problemu z rozróżnieniem obu jednostek

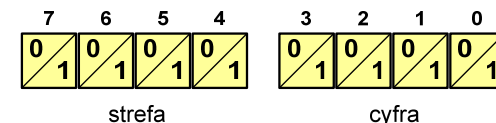


Jednostki informacji - tetrada

- Bajt 8-bitowy można podzielić na dwie połowki 4-bitowe nazywane **tetradami** (ang. nibbles)
- Rozróżniamy bardziej znaczącą (górną) i mniej znaczącą (dolną) tetradę



- Spotyka się też określenie **strefa** i **cyfra**



Jednostki informacji - bajt

- Wielokrotności bajtów:

Przedrostki dziesiętne (układ SI)		
Nazwa	Symbol	Mnożnik
bajt	B	---
kilobajt	kB	$10^3 = 1000^1$
megabajt	MB	$10^6 = 1000^2$
gigabajt	GB	$10^9 = 1000^3$
terabajt	TB	$10^{12} = 1000^4$
petabajt	PB	$10^{15} = 1000^5$
eksabajt	EB	$10^{18} = 1000^6$
zettabajt	ZB	$10^{21} = 1000^7$
jottabajt	YB	$10^{24} = 1000^8$

Przedrostki binarne (IEC 60027-2)		
Nazwa	Symbol	Mnożnik
bajt	B	---
kibibajt	KiB	$2^{10} = 1024^1$
mebibajt	MiB	$2^{20} = 1024^2$
gibibajt	GiB	$2^{30} = 1024^3$
tebibajt	TiB	$2^{40} = 1024^4$
pebibajt	PiB	$2^{50} = 1024^5$
eksbibajt	EiB	$2^{60} = 1024^6$
zebibajt	ZiB	$2^{70} = 1024^7$
jobibajt	YiB	$2^{80} = 1024^8$

Jednostki informacji - bajt

- Przedrostki binarne (dwójkowe) nie zostały przyjęte przez wszystkie środowiska zajmujące się informatyką
- Producenci nośników pamięci korzystają z przedrostków dziesiętnych

Prefiks	Nazwa	System SI	System binarny	Różnica
k	kilo	$10^3 = 1000$	$2^{10} = 1024$	2,40%
M	mega	$10^6 = 1000000$	$2^{20} = 1048576$	4,86%
G	giga	$10^9 = 1000000000$	$2^{30} = 1073741824$	7,37%
T	tera	$10^{12} = 1000000000000$	$2^{40} = 1099511627776$	9,95%

- Z ulotki „Dysk Desktop HDD - zestawienie danych”, Seagate:
 - w przypadku oznaczania pojemności dysków, jeden gigabajt (oznaczany także jako „GB”) jest równy jednemu miliardowi bajtów, a jeden terabajt (oznaczany także jako „TB”) jest równy jednemu bilionowi bajtów

Jednostki informacji - bajt

- Seagate ST1000DM003 (1 TB)

- Drive specification:

- formatted capacity: 1000 GB (1 TB)
- guaranteed sectors: 1,953,525,168
- bytes per sector: 4096
(4K physical emulated at 512-byte sectors)



- Pojemność dysku:

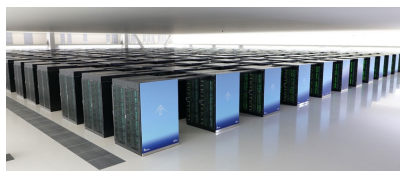
- $1.953.525.168 \times 512 = 1.000.204.886.016$ bajtów
- $1.000.204.886.016 / (1024) = 976.762.584$ kB
- $1.000.204.886.016 / (1024 \times 1024) = 953.870$ MB
- $1.000.204.886.016 / (1024 \times 1024 \times 1024) = 931,5$ GB

Słowo maszynowe (słowo)

- Słowo maszynowe** (słowo - ang. word) - jednostka danych używana przez określony komputer (określoną architekturę)
- Słowo składa się odgórnie określonej liczby bitów, nazywanej **długością** lub **szerokością słowa** (najczęściej jest to potęga 2, np. 8, 16, 32, 64 bity)
- Zazwyczaj wielkość słowa określa:
 - rozmiar rejestrów procesora
 - rozmiar szyny danych i szyny adresowej
- Architektury:
 - 8-bitowa: Intel 8080, Z80, Motorola 6800, Intel 8051
 - 16-bitowa: Intel 8086, Intel 80286
 - 32-bitowa: Intel od 80386 do i7, AMD od 5x86 do Athlona, ARM
 - 64-bitowa: Intel Itanium, Pentium 4/EM64T, Core 2, Core i7
AMD Opteron, Athlon 64, Athlon II

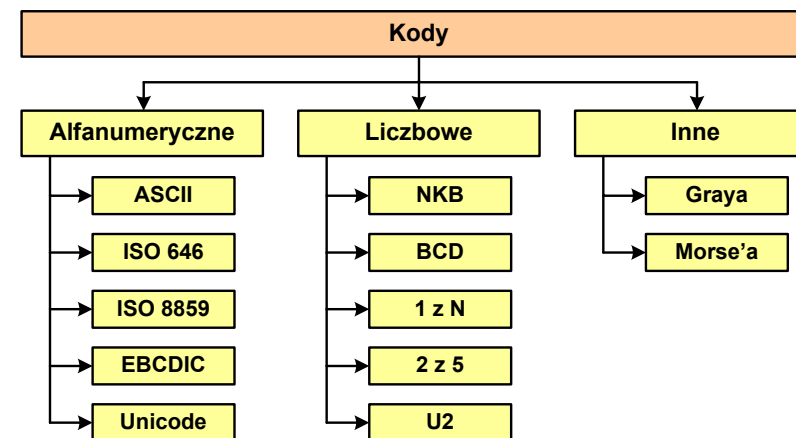
FLOPS

- **FLOPS** (FLoating point OPerations Per Second)
 - liczba operacji zmiennoprzecinkowych na sekundę
 - jednostka wydajności układów zmiennoprzecinkowych
- Przykłady wydajności procesorów (teoretyczne):
 - Intel Core i7 975 3,46 GHz - 55,36 GFlops
 - Intel Core2 Quad Q9650 3,00 GHz - 48 GFlops
 - Intel Core2 Duo E8400 3,00 GHz - 24 GFlops
 - najszybszy system równoległy na świecie:
Supercomputer Fugaku (Japan) - 442.010.000 GFlops
RIKEN Center for Computational Science
processors: Fujitsu A64FX 2.2 GHz
nodes: 158.976, cores: 7.630.848
Red Hat Enterprise Linux
www.top500.org



Kodowanie

- **Kodowanie** - proces przekształcania jednego rodzaju postaci informacji na inną postać



Kod ASCII

- **ASCII - American Standard Code for Information Interchange**
 - 7-bitowy kod przypisujący liczby z zakresu 0-127:
 - literom (alfabet angielski)
 - cyfrom
 - znakom przestankowym
 - innym symbolom
 - poleceniom sterującym.

Dec Hex Char	Dec Hex Char	Dec Hex Char	Dec Hex Char
0 0 NUL	32 20 Space	64 40 @	96 60 `
1 1 SOH	33 21 !	65 41 A	97 61 a
2 2 STX	34 22 "	66 42 B	98 62 b
3 3 ETX	35 23 #	67 43 C	99 63 c
4 4 EOT	36 24 \$	68 44 D	100 64 d
5 5 ENQ	37 25 %	69 45 E	101 65 e
6 6 ACK	38 26 &	70 46 F	102 66 f
7 7 BEL	39 27 ^	71 47 G	103 67 g
8 8 BS	40 28 (72 48 H	104 68 h
9 9 TAB	41 29)	73 49 I	105 69 i
10 A LF	42 2A *	74 4A J	106 6A j
11 B VT	43 2B +	75 4B K	107 6B k
12 C FF	44 2C ,	76 4C L	108 6C l
13 D CR	45 2D -	77 4D M	109 6D m
14 E SO	46 2E .	78 4E N	110 6E n
15 F SI	47 2F /	79 4F O	111 6F o
16 10 DLE	48 30 0	80 50 P	112 70 p
17 11 DC1	49 31 1	81 51 Q	113 71 q
18 12 DC2	50 32 2	82 52 R	114 72 r
19 13 DC3	51 33 3	83 53 S	115 73 s
20 14 DC4	52 34 4	84 54 T	116 74 t
21 15 NAK	53 35 5	85 55 U	117 75 u
22 16 SYN	54 36 6	86 56 V	118 76 v
23 17 ETB	55 37 7	87 57 W	119 77 w
24 18 CAN	56 38 8	88 58 X	120 78 x
25 19 EM	57 39 9	89 59 Y	121 79 y
26 1A SUB	58 3A :	90 5A Z	122 7A z
27 1B ESC	59 3B ;	91 5B [123 7B {
28 1C FS	60 3C <	92 5C \	124 7C
29 1D GS	61 3D =	93 5D]	125 7D }
30 1E RS	62 3E >	94 5E ^	126 7E ~
31 1F US	63 3F ?	95 5F _	127 7F DEL

Kod ASCII - Kody sterujące

- Kody sterujące - 33 kody, o numerach: 0-31, 127

Dec Hex Char	Dec Hex Char
0 0 NUL (null)	16 10 DLE (data link escape)
1 1 SOH (start of heading)	17 11 DC1 (device control 1)
2 2 STX (start of text)	18 12 DC2 (device control 2)
3 3 ETX (end of text)	19 13 DC3 (device control 3)
4 4 EOT (end of transmission)	20 14 DC4 (device control 4)
5 5 ENQ (enquiry)	21 15 NAK (negative acknowledge)
6 6 ACK (acknowledge)	22 16 SYN (synchronous idle)
7 7 BEL (bell)	23 17 ETB (end of trans. block)
8 8 BS (backspace)	24 18 CAN (cancel)
9 9 TAB (horizontal tab)	25 19 EM (end of medium)
10 A LF (NL line feed, new line)	26 1A SUB (substitute)
11 B VT (vertical tab)	27 1B ESC (escape)
12 C FF (NP form feed, new page)	28 1C FS (file separator)
13 D CR (carriage return)	29 1D GS (group separator)
14 E SO (shift out)	30 1E RS (record separator)
15 F SI (shift in)	31 1F US (unit separator)
	127 7F DEL

- W języku C:

0 (NULL) - \0 7 (BEL) - \a 8 (BS) - \b
 9 (TAB) - \t 10 (LF) - \n 13 (CR) - \r

Kod ASCII - Pliki tekstowe

- Elementami pliku tekstowego są **wiersze**, mogą one mieć różną długość
- W systemie Windows każdy wiersz pliku zakończony jest parą znaków:
 - CR**, ang. carriage return - powrót karetki, kod ASCII - $13_{(10)} = 0D_{(16)}$
 - LF**, ang. line feed - przesunięcie o wiersz, kod ASCII - $10_{(10)} = 0A_{(16)}$

- Założmy, że plik tekstowy ma postać:

```
Pierwszy wiersz pliku
Drugi wiersz pliku
Trzeci wiersz pliku
```

- Rzeczywista zawartość pliku jest następująca:

```
00000000: 50 69 65 72 77 73 7A 79|20 77 69 65 72 73 7A 20 | Pierwszy wiersz
00000010: 70 6C 69 68 75 0D 0A 44|72 75 67 69 20 77 69 65 | plikuDrugi wiersz
00000020: 72 73 7A 20 70 6C 69 6B|75 0D 0A 54 72 7A 65 63 | rz plikuTrzec
00000030: 69 20 77 69 65 72 73 7A|20 70 6C 69 68 75 0D 0A | i wiersz pliku
```

- Wydruk zawiera:
 - przesunięcie od początku pliku (szesnastkowo)
 - wartości poszczególnych bajtów pliku (szesnastkowo)
 - znaki odpowiadające bajtom pliku (traktując bajty jako kody ASCII)

ISO/IEC 646

- ISO/IEC 646** - norma definiująca modyfikację 7-bitowego kodowania ASCII, stosowana w latach 70-tych i 80-tych
- W normie określono 10 pozycji na znaki w języku kraju, który przyjął tę normę oraz 2 pozycje na znaki walut

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	Znaki kontrolne															
10	Znaki kontrolne															
20	SP	!	"	#	\$	%	&	\	()	*	+	,	-	.	/	
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{		}	~	

- zółty - znaki narodowe
- niebieski - znaki walut

- Wszystkie pozostałe znaki są zgodne z ASCII

Kod ASCII - Pliki tekstowe

- W systemie Linux znakiem końca wiersza jest tylko **LF** o kodzie ASCII - $10_{(10)} = 0A_{(16)}$

- Założmy, że plik tekstowy ma postać:

```
Pierwszy wiersz pliku
Drugi wiersz pliku
Trzeci wiersz pliku
```

- Rzeczywista zawartość pliku jest następująca:

```
00000000: 50 69 65 72 77 73 7A 79|20 77 69 65 72 73 7A 20 | Pierwszy wiersz
00000010: 70 6C 69 68 75 0A 44|72 75 67 69 20 77 69 65 72 | plikuDrugi wiersz
00000020: 73 7A 20 70 6C 69 68 75|0A 54 72 7A 65 63 69 20 | sz plikuTrzeci
00000030: 77 69 65 72 73 7A 20 70|6C 69 68 75 0A | wiersz pliku
```

- Podczas przesyłania pliku tekstowego (np. przez protokół **ftp**) z systemu Linux do systemu Windows pojedynczy znak **LF** zamieniany jest automatycznie na parę znaków **CR** i **LF**
- Błędne przesłanie pliku tekstowego (w trybie binarnym) powoduje nieprawidłowe jego wyświetlanie:

```
Pierwszy wiersz plikuDrugi wiersz plikuTrzeci wiersz pliku
```

ISO/IEC 646 - odmiany narodowe

USA																	Niemcy																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F																	
00	Znaki kontrolne																Znaki kontrolne																
10	Znaki kontrolne																Znaki kontrolne																
20	SP	!	"	#	\$	%	&	\	()	*	+	,	-	.	/	SP	!	"	#	\$	%	&	\	()	*	+	,	-	.	/			
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
50	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_	P	Q	R	S	T	U	V	W	X	Y	Z	ä	ö	ü	^	_	
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	
70	p	q	r	s	t	u	v	w	x	y	z	{		}	~		p	q	r	s	t	u	v	w	x	y	z	ä	ö	ü	ß		

ISO/IEC 8859

- ISO/IEC 8859 - zestaw standardów służących do kodowania znaków za pomocą 8-bitów
- Wszystkie zestawy ISO 8859 mają znaki $0_{(10)}-127_{(10)}$ ($00_{(16)}-7F_{(16)}$) takie same jak w kodzie ASCII
- Pozycjom $128_{(10)}-159_{(10)}$ ($80_{(16)}-9F_{(16)}$) przypisane są dodatkowe kody sterujące, tzw. C1 (obecnie nie są używane)
- Od czerwca 2004 roku ISO 8859 nie jest rozwijane.

ISO/IEC 8859-1

- ISO/IEC 8859-1, Latin-1 („zachodnioeuropejskie”)
- kodowanie używane w Amerykach, Europie Zachodniej, Oceanii i większej części Afryki
- dostępne języki: albański, angielski, baskijski, duński, estoński, fiński, francuski, hiszpański, irlandzki, islandzki, kataloński, łaciński, niderlandzki, niemiecki, norweski, portugalski, retoromański, szkocki, szwedzki, włoski
- 191 znaków łacińskiego pisma.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	Znaki kontrolne															
10	Znaki kontrolne															
20	SP	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{		}	~	
80	Nie używane															
90	Nie używane															
A0	NBSP	ı	ç	£	¥	ı	Š	•	©	ª	«	¬	SHY	®	™	
B0	°	±	²	³	µ	¶	·	¸	¹	º	»	¼	½	¾	¿	
C0	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D0	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E0	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F0	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

SP - spacja
NBSP - twarda spacja
SHY - miękki dywiz (myślnik)

ISO/IEC 8859

- Stosowane standardy ISO 8859:
 - ISO 8859-1 (Latin-1) - alfabet łaciński dla Europy zachodniej
 - ISO 8859-2 (Latin-2) - łaciński dla Europy środkowej i wschodniej
 - ISO 8859-3 (Latin-3) - łaciński dla Europy południowej
 - ISO 8859-4 (Latin-4) - łaciński dla Europy północnej
 - ISO 8859-5 (Cyrillic) - dla cyrylicy
 - ISO 8859-6 (Arabic) - dla alfabetu arabskiego
 - ISO 8859-7 (Greek) - dla alfabetu greckiego
 - ISO 8859-8 (Hebrew) - dla alfabetu hebrajskiego
 - ISO 8859-9 (Latin-5)
 - ISO 8859-10 (Latin-6)
 - ISO 8859-11 (Thai) - dla alfabetu tajskiego
 - ISO 8859-12 - brak
 - ISO 8859-13 (Latin-7)
 - ISO 8859-14 (Latin-8) - zawiera polskie litery
 - ISO 8859-15 (Latin-9)
 - ISO 8859-16 (Latin-10) - łaciński dla Europy środkowej, zawiera polskie litery

ISO/IEC 8859-2

- ISO/IEC 8859-2, Latin-2 („środkowo”, „wschodnioeuropejskie”)
- dostępne języki: bośniacki, chorwacki, czeski, węgierski, polski, rumuński, serbski, serbsko-chorwacki, słowacki, słoweński, górno- i dolnołużycki
- możliwość przedstawienia znaków w języku niemieckim i angielskim
- 191 znaków łacińskiego pisma
- do 02.11.2015 kodowanie to było zgodne z **Polską Normą**

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	Znaki kontrolne															
10	Znaki kontrolne															
20	SP	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{		}	~	
80	Nie używane															
90	Nie używane															
A0	NBSP	ı	ç	£	¥	ı	Š	•	©	ª	«	¬	SHY	®	™	
B0	°	±	²	³	µ	¶	·	¸	¹	º	»	¼	½	¾	¿	
C0	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D0	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E0	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F0	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

SP - spacja
NBSP - twarda spacja
SHY - miękki dywiz (myślnik)

ISO/IEC 8859-2 - Litery diakrytyczne w j. polskim

■ 18 liter:

- Ą - ą
- Ć - ć
- Ę - ę
- Ł - ł
- Ń - ń
- Ó - ó
- Ś - ś
- Ż - ż
- Ź - ź

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	Znaki kontrolne															
10	Znaki kontrolne															
20	SP	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{		}	~	
80	Nie używane															
90	Nie używane															
A0	NB	SP	Ą	Ć	Ę	Ł	Ń	Ś	Ş	Ź	Ż	SHY	Ż	Ż		
B0	°	ą	ć	ę	ł	ń	ś	ş	ź	ż	~	ż	ż			
C0	Ŕ	Ą	Ć	Ę	Ł	Ń	Ś	Ş	Ź	Ż	SHY	Ż	Ż			
D0	Đ	Ń	Ń	Ó	Ô	Ö	×	Ŕ	Ů	Ů	Ů	Ů	Ů	Ů	Ů	Ů
E0	ř	á	â	ã	ä	å	ı	ć	ç	č	é	ę	ě	í	ı	ď
F0	đ	ñ	ñ	ó	ô	ö	÷	ř	ů	ů	ů	ů	ů	ů	ů	ů

EBCDIC

- EBCDIC - Extended Binary Coded Decimal Interchange Code
- 8-bitowe kodowanie znaków stworzone jako rozszerzenie kodowania BCD

- używane głównie w systemach IBM w latach 60-tych XX wieku
- umożliwia zapisanie do 256 różnych symboli
- brak zachowania kolejności liter zgodnie z kolejnością kodów, np. po R nie ma S
- kody EBCDIC nie są zgodne z ASCII.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	Znaki kontrolne															
10	Znaki kontrolne															
20	Znaki kontrolne															
30	Znaki kontrolne															
40	SP	NB	SP	À	Á	Â	Ã	Ä	Å	Ç	È	É	Ê	Ë	Ì	Í
50	&	é	ê	ë	ì	í	î	ï	Ĵ	Š	*)	;	^			
60	- /	À	Á	Â	Ã	Ä	Å	Ç	È	É	Ê	Ë	Ì	Í	Ĵ	Š
70	ø	È	Ê	Ë	Ì	Í	Ĵ	Š	ˆ	ˆ	ˆ	ˆ	ˆ	ˆ	ˆ	ˆ
80	ø	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
90	°	j	k	l	m	n	o	p	q	r	°	æ	Æ	Æ	Æ	Æ
A0	µ	~	s	t	u	v	w	x	y	z	ı	ı	ı	ı	ı	ı
B0	ç	£	¥	€	©	§	¶	¼	½	¾	ı	ı	ı	ı	ı	ı
C0	{	A	B	C	D	E	F	G	H	I	SHY	ó	ô	õ	õ	õ
D0	}	J	K	L	M	N	O	P	Q	R	ı	ü	ü	ü	ü	ü
E0	\	÷	S	T	U	V	W	X	Y	Z	²	ö	ö	ö	ö	ö
F0	0	1	2	3	4	5	6	7	8	9	³	ü	ü	ü	ü	ü

EBCDIC i ISO 8859-1 - porównanie

ISO 8859-1

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	Znaki kontrolne															
10	Znaki kontrolne															
20	Znaki kontrolne															
30	SP	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{		}	~	
80	Nie używane															
90	Nie używane															
A0	NB	SP	İ	Ç	£	¥	€	©	§	¶	¼	½	¾	ı	ı	ı
B0	°	ı	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C0	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D0	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E0	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F0	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

EBCDIC

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	Znaki kontrolne															
10	Znaki kontrolne															
20	Znaki kontrolne															
30	Znaki kontrolne															
40	SP	NB	SP	À	Á	Â	Ã	Ä	Å	Ç	È	É	Ê	Ë	Ì	Í
50	&	é	ê	ë	ì	í	î	ï	Ĵ	Š	*)	;	^			
60	- /	À	Á	Â	Ã	Ä	Å	Ç	È	É	Ê	Ë	Ì	Í	Ĵ	Š
70	ø	È	Ê	Ë	Ì	Í	Ĵ	Š	ˆ	ˆ	ˆ	ˆ	ˆ	ˆ	ˆ	ˆ
80	ø	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
90	°	j	k	l	m	n	o	p	q	r	°	æ	Æ	Æ	Æ	Æ
A0	µ	~	s	t	u	v	w	x	y	z	ı	ı	ı	ı	ı	ı
B0	ç	£	¥	€	©	§	¶	¼	½	¾	ı	ı	ı	ı	ı	ı
C0	{	A	B	C	D	E	F	G	H	I	SHY	ó	ô	õ	õ	õ
D0	}	J	K	L	M	N	O	P	Q	R	ı	ü	ü	ü	ü	ü
E0	\	÷	S	T	U	V	W	X	Y	Z	²	ö	ö	ö	ö	ö
F0	0	1	2	3	4	5	6	7	8	9	³	ü	ü	ü	ü	ü

Windows-1250

- **Windows-1250 (CP-1250)** - strona kodowa używana przez system Microsoft Windows do reprezentacji tekstów w językach środkowoeuropejskich używających alfabetu łacińskiego
- Obsługiwane języki: albański, chorwacki, czeski, polski, rumuński, słowacki, słoweński, węgierski (ale także niemiecki)
- Windows-1250 jest podobny do ISO 8859-2 - posiada wszystkie jego drukowalne znaki (a także kilka dodatkowych), lecz kilka z nich zajmuje inne miejsca.

ISO 8859-2 i Windows-1250 - porównanie

ISO 8859-2																Windows-1250																	
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	Znaki kontrolne																Znaki kontrolne																
10	Znaki kontrolne																Znaki kontrolne																
20	SP	!	"	#	\$	%	&	'	()	*	+	,	-	.	/	SP	!	"	#	\$	%	&	'	()	*	+	,	-	.	/	
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
50	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_	
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	
70	p	q	r	s	t	u	v	w	x	y	z	{		}	~	.	p	q	r	s	t	u	v	w	x	y	z	{		}	~	.	
80	Nieużywane																Nieużywane																
90	Nieużywane																Nieużywane																
A0	NB	A	~	z	z	z	z	z	z	z	z	z	z	z	z	z	NB	A	~	z	z	z	z	z	z	z	z	z	z	z	z	z	
B0	SP	°	±	·	µ	¶	·	·	·	·	·	·	·	·	·	·	SP	°	±	·	µ	¶	·	·	·	·	·	·	·	·	·	·	
C0	R	Á	Â	Ã	Ä	Å	Ā	Ĉ	Č	Ď	Ě	Ě	Ě	Ī	Ī	Ī	R	Á	Â	Ã	Ä	Å	Ā	Ĉ	Č	Ď	Ě	Ě	Ě	Ī	Ī	Ī	
D0	Đ	Ń	Ń	Ń	Ń	Ń	Ń	Ń	Ń	Ń	Ń	Ń	Ń	Ń	Ń	Ń	Đ	Ń	Ń	Ń	Ń	Ń	Ń	Ń	Ń	Ń	Ń	Ń	Ń	Ń	Ń	Ń	
E0	ř	á	â	ã	ä	å	ā	č	ć	ď	ě	ē	ē	ī	ĭ	đ	ř	á	â	ã	ä	å	ā	č	ć	ď	ě	ē	ē	ī	ĭ	đ	
F0	đ	ñ	ñ	ñ	ñ	ñ	ñ	ñ	ñ	ñ	ñ	ñ	ñ	ñ	ñ	ñ	đ	ñ	ñ	ñ	ñ	ñ	ñ	ñ	ñ	ñ	ñ	ñ	ñ	ñ	ñ	ñ	

Problem kodowania polskich liter diakrytycznych

- Problem z wyświetlaniem polskich liter diakrytycznych
 - Tekst zapisany w standardzie ISO-8859-2:

Ā Ć Ę Ź Ń Ó Ś Ź Ź
 ā ć ę ź ń ó ś ź ź

- Tekst wyświetlony w Notatniku systemu Windows (Windows-1250):

Ā Ć Ę Ź Ń Ó Ź Ź
 ā ć ę ź ń ó Ź Ź

Unicode (Unikod)



- Komputerowy zestaw znaków mający obejmować wszystkie pisma i inne znaki (symbole techniczne, wymowy) używane na świecie
- Unicode przypisuje unikalny numer każdemu znakowi, niezależny od używanej platformy, programu czy języka
- Rozwijany przez konsorcjum utworzone przez firmy komputerowe, producentów oprogramowania oraz grupy użytkowników
 - <http://www.unicode.org>
- Pierwsza wersja: **Unicode 1.0** (10.1991)
- Ostatnia wersja: **Unicode 13.0** (10.03.2020)
 - The Unicode Consortium. The Unicode Standard, Version 13.0.0, (Mountain View, CA: The Unicode Consortium, 2020)
 - <http://www.unicode.org/versions/Unicode13.0.0/>
 - koduje 143.859 znaków



Unicode - Zakresy

Zakres:	Znaczenie:
U+0000 - U+007F	Basic Latin (to samo co w ASCII)
U+0080 - U+00FF	Latin-1 Supplement (to samo co w ISO/IEC 8859-1)
U+0100 - U+017F	Latin Extended-A
U+0180 - U+024F	Latin Extended-B
U+0250 - U+02AF	IPA Extensions
U+02B0 - U+02FF	Spacing Modifiers Letters
...	
U+0370 - U+03FF	Greek
U+0400 - U+04FF	Cyrillic
...	
U+1D00 - U+1D7F	Phonetic Extensions
U+1D80 - U+1DBF	Phonetic Extensions Supplement
U+1E00 - U+1EFF	Latin Extended Additional
U+1F00 - U+1FFF	Greek Extended
...	



Unicode

- Standard Unicode definiuje nie tylko kody numeryczne przypisane poszczególnym znakom, ale także określa sposób bajtowego kodowania znaków
- Kodowanie określa sposób w jaki znaki ze zbioru mają być zapisane w postaci binarnej
- Istnieją trzy podstawowe metody kodowania:
 - 32-bitowe: UTF-32
 - 16-bitowe: UTF-16
 - 8-bitowe: UTF-8
 gdzie: UTF - UCS Transformation Format
UCS - Universal Character Set
- Wszystkie metody obejmują wszystkie kodowane znaki w Unicode.



Unicode

- Metody kodowania różnią się liczbą bajtów przeznaczonych do opisanego kodu znaku

A 00000041	Ω 000003A9	語 00008A9E	𐄆 00010384	UTF-32
A 0041	Ω 03A9	語 8A9E	𐄆 D800 DF84	UTF-16
A 41	Ω CE A9	語 E8 AA 9E	𐄆 F0 90 8E 84	UTF-8



Unicode - kodowanie UTF-32

- UTF-32 - sposób kodowania standardu Unicode wymagający użycia 32-bitowych słów

A 00000041	Ω 000003A9	語 00008A9E	𐄆 00010384	UTF-32
---------------	---------------	---------------	---------------	--------

- Kod znaku ma zawsze stałą długość 4 bajtów i przedstawia numer znaku w tabeli Unikodu
- Kody obejmują zakres od 0 do 0x10FFFF (od 0 do 1 114 111)
- Kodowanie to jest jednak bardzo nieefektywne - zakodowane ciągi znaków są 2-4 razy dłuższe niż ciągi tych samych znaków zapisanych w innych kodowaniach.

Unicode - kodowanie UTF-16



- UTF-16 - sposób kodowania standardu Unicode wymagający użycia 16-bitowych słów

A 0041	Ω 03A9	語 8A9E	Ⅲ D800 DF84	UTF-16
-----------	-----------	-----------	----------------	--------

- Dla znaków z przedziału od U+0000 do U+FFFF używane jest jedno słowo, którego wartość jest jednocześnie kodem znaku w Unicode
- Dla znaków z wyższych pozycji używa się dwóch słów:
 - pierwsze słowo należy do przedziału: U+D800 - U+DBFF
 - drugie słowo należy do przedziału: U+DC00 - U+DFFF.

Unicode - kodowanie UTF-8



- UTF-8 - kodowanie ze zmienną długością reprezentacji znaku wymagające użycia 8-bitowych słów

A 41	Ω CE A9	語 E8 AA 9E	Ⅲ F0 90 8E 84	UTF-8
---------	------------	---------------	------------------	-------

- Znaki Unikodu są mapowane na ciągi bajtów
 - 0x00 do 0x7F - bity 0xxxxxxx
 - 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

Unicode



	010	011	012	013	014	015	016	017
0	Ā	Đ	Ġ	İ	Ĳ	Ō	Š	Ū
1	ā	đ	ġ	ı	ł	ő	š	ū
2	Ă	Ē	Ģ	Ĳ	Ĳ	Œ	Ţ	Ū
3	ă	ē	ģ	ĳ	Ń	œ	ţ	ū
4	Ą	Ĕ	Ĥ	Ĵ	ń	ŕ	ŧ	Ű
5	ą	ė	ĥ	ĵ	ņ	í	ť	ű
6	Ć	Ė	Ħ	Ķ	Ŗ	Ŧ	Ŷ	
7	ć	ė	ħ	ķ	ŗ	ŧ	ŷ	

European Latin

- 0100 Ā LATIN CAPITAL LETTER A WITH MACRON
≡ 0041 A 0304 0
- 0101 ā LATIN SMALL LETTER A WITH MACRON
• Latvian, Latin, ...
≡ 0061 a 0304 0
- 0102 Ă LATIN CAPITAL LETTER A WITH BREVE
≡ 0041 A 0306 0
- 0103 ă LATIN SMALL LETTER A WITH BREVE
• Romanian, Vietnamese, Latin, ...
≡ 0061 a 0306 0
- 0104 Ą LATIN CAPITAL LETTER A WITH OGONEK
≡ 0041 A 0328 0
- 0105 ą LATIN SMALL LETTER A WITH OGONEK
• Polish, Lithuanian, ...
≡ 0061 a 0328 0
- 0106 Ć LATIN CAPITAL LETTER C WITH ACUTE
≡ 0043 C 0301 0
- 0107 ć LATIN SMALL LETTER C WITH ACUTE
• Polish, Croatian, ...
→ 045B ģ cyrillic small letter tshe
≡ 0063 c 0301 0

Unicode



27308

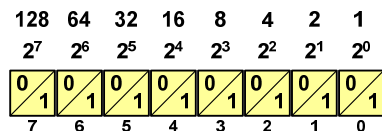
CJK Unified Ideographs Extension B

27342

27308 虫 142.8 𧈧	27309 虫 142.8 𧈨	2730A 虫 142.8 𧈩	2730B 虫 142.8 𧈪	2730C 虫 142.8 𧈫	2730D 虫 142.8 𧈬	2731B 虫 142.8 𧈭	2731C 虫 142.8 𧈮	2731D 虫 142.8 𧈯	2731E 虫 142.7 𧈰	2731F 虫 142.8 𧈱	27320 虫 142.8 𧈲	2732F 虫 142.8 𧈳	27330 虫 142.9 𧈴	27331 虫 142.8 𧈵	27332 虫 142.8 𧈶	27333 虫 142.8 𧈷	27334 虫 142.8 𧈸
-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------

Kody liczbowe - Naturalny Kod Binarny (NKB)

- Jeżeli dowolnej liczbie dziesiętnej przypiszemy odpowiadającą jej liczbę binarną, to otrzymamy **naturalny kod binarny** (NKB)



Liczba dziesiętna	Kod NKB	Liczba dziesiętna	Kod NKB
0	0000	8	1000
1	0001	9	1001
2	0010	10	1010
3	0011	11	1011
4	0100	12	1100
5	0101	13	1101
6	0110	14	1110
7	0111	15	1111

Kody liczbowe - Kod BCD

- Przykład:**

$$168_{(10)} = ?_{(BCD)}$$

$$\underbrace{0001}_1 \underbrace{0110}_6 \underbrace{1000}_8$$

$$168_{(10)} = 000101101000_{(BCD)}$$

$$1001 | 0101 | 0011_{(BCD)} = ?_{(10)}$$

$$\underbrace{1001}_9 \underbrace{0101}_5 \underbrace{0011}_3$$

$$100101010011_{(BCD)} = 953_{(10)}$$

- Zastosowania:**

- urządzenia elektroniczne z wyświetlaczem cyfrowym (np. kalkulatory, mierniki cyfrowe, kasy sklepowe, wagi)
- przechowywania daty i czasu w BIOSie komputerów (także wczesne modele PlayStation 3)
- zapis części ułamkowych kwot (systemy bankowe).

Kody liczbowe - Kod BCD

- Binary-Coded Decimal** - dziesiętny zakodowany dwójkowo
- BCD** - sposób zapisu liczb polegający na zakodowaniu kolejnych cyfr liczby dziesiętnej w 4-bitowym systemie dwójkowym (NKB)

Cyfra dziesiętna	BCD	Cyfra dziesiętna	BCD
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

- W ogólnym przypadku kodowane są tylko znaki $0 \div 9$
- Pozostałe kombinacje bitowe mogą być stosowane do kodowania znaku liczby lub innych znaczników.

Kody liczbowe - Kod BCD: przechowywanie liczb

- Użycie 4 najmłodszych bitów jednego bajta, 4 starsze bity są ustawiane na jakąś konkretną wartość:
 - 0000
 - 1111 (np. kod EBCDIC, liczby $F0_{(16)} \div F9_{(16)}$)
 - 0011 (tak jak w ASCII, liczby $30_{(16)} \div 39_{(16)}$)
- Zapis dwóch cyfr w każdym bajcie (starsza na starszej połówce, młodsza na młodszej połówce) - jest to tzw. **spakowane BCD**
 - w przypadku liczby zapisanej na kilku bajtach, najmniej znacząca tetrada (4 bity) używane są jako flaga znaku
 - standardowo przyjmuje się 1100 ($C_{(16)}$) dla znaku plus (+) i 1101 ($D_{(16)}$) dla znaku minus (-), np.

$$127_{(10)} = 0001 0010 0111 \mathbf{1100} \quad (127C_{(16)})$$

$$-127_{(10)} = 0001 0010 0111 \mathbf{1101} \quad (127D_{(16)})$$

Kody liczbowe - Kod BCD

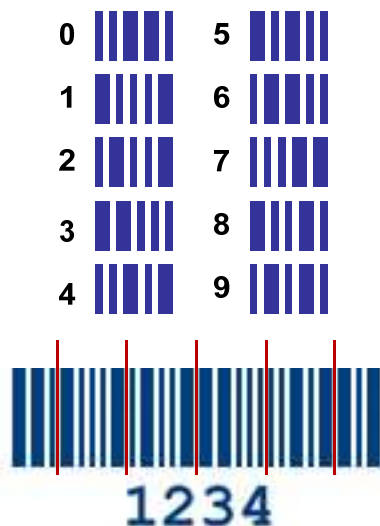
- Warianty kodu BCD:

Cyfra dziesiętna	BCD 8421	Excess-3	BCD 2421	BCD 84-2-1	IBM 1401 BCD 8421
0	0000	0011	0000	0000	1010
1	0001	0100	0001	0111	0001
2	0010	0101	0010	0110	0010
3	0011	0110	0011	0101	0011
4	0100	0111	0100	0100	0100
5	0101	1000	1011	1011	0101
6	0110	1001	1100	1010	0110
7	0111	1010	1101	1001	0111
8	1000	1011	1110	1000	1000
9	1001	1100	1111	1111	1001

- Podstawowy wariant: **BCD 8421** (SBCD - Simple Binary Coded Decimal)

Kody liczbowe - Kod 2 z 5 Industrial (1960 r.)

- Jednowymiarowy kod kreskowy kodujący cyfry: $0 \div 9$
- Znak to 5 pasków: 2 szerokie i 3 wąskie
- Szeroki pasek jest wielokrotnością wąskiego, szerokości muszą być takie same dla całego kodu
- Struktura kodu:
 - start: 11011010
 - numer
 - stop: 11010110



Kody liczbowe - Kod 2 z 5

- Kod 5-bitowy: 2 bity zawsze równe 1, a 3 bity zawsze równe 0
- Koduje 10 znaków (cyfry dziesiętne), kody nie są wzajemnie jednoznaczne (ta sama wartość może być zakodowana w różny sposób)
- Kod stałowagowy
- Kod detekcyjny
- Stosowany głównie w **kodach kreskowych**

Liczba dziesiętna	2 z 5 (01236)	2 z 5 (01234)	2 z 5 (74210)
0	01100	01100	11000
1	11000	11000	00011
2	10100	10100	00101
3	10010	10010	00110
4	01010	01010	01001
5	00110	00110	01010
6	10001	10001	01100
7	01001	01001	10001
8	00101	00101	10010
9	00011	00011	10100

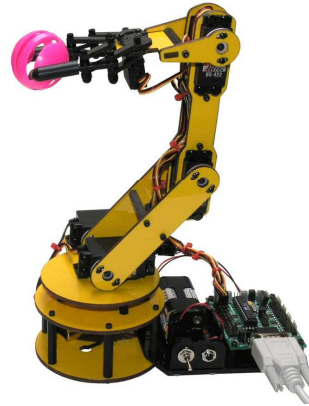
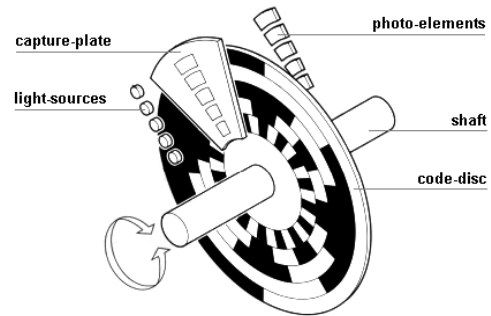
Kod Graya (refleksyjny)

- Kod dwójkowy, bezwagowy, niepozycyjny
- Dwa kolejne słowa kodowe różnią się stanem jednego bitu
- Kod cykliczny - ostatni i pierwszy wyraz również różnią się stanem jednego bitu

kod 1-bitowy	kod 2-bitowy	kod 3-bitowy
0	00	000
1	01	001
	11	011
	10	010
		110
		111
		101
		100

Kod Graya

- Stosowany w przetwornikach analogowo-cyfrowych, do cyfrowego pomiaru analogowych wielkości mechanicznych (np. kąt obrotu)



<http://tams-www.informatik.uni-hamburg.de/applets/hades/webdemos/10-gates/15-graycode/dual2gray.html>

Koniec wykładu nr 4

Dziękuję za uwagę!