Informatyka 1

Politechnika Białostocka - Wydział Elektryczny

Elektrotechnika, semestr II, studia niestacjonarne I stopnia

Rok akademicki 2017/2018

Wykład nr 2 (02.03.2018)

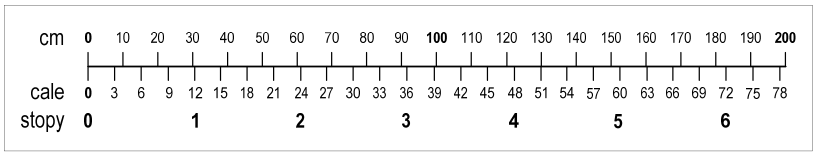
dr inż. Jarosław Forenc

Plan wykładu nr 2

- Język C
  - identyfikatory (nazwy), słowa kluczowe
  - typy danych, stałe liczbowe
  - deklaracje zmiennych i stałych
  - operatory, priorytet operatorów
  - wyrażenia i instrukcje
  - wyrażenia arytmetyczne
  - funkcje printf i scanf
- Konwersje między systemami liczbowymi
- Kodowanie znaków
  - ASCII, ISO/IEC 646, ISO 8859
  - EBCDIC, Windows-1250, Unicode

Przykład: zamiana wzrostu w cm na stopy i cale

- Wybrane jednostki długości w brytyjskim systemie miar:
  - 1 cal (inch) [in] = 2,54 [cm]
  - 1 stopa (foot) [ft] = 12 cali = 30,48 [cm]



  - 1 jard (yard) [yd] = 3 stopy = 91,44 [cm]
  - 1 furlong [fur] = 660 stóp = 201,168 [m]
  - 1 mila (mile) [mi] = 8 furlongów = 1609,344 [m]

Przykład: zamiana wzrostu w cm na stopy i cale

```
#include <stdio.h>

int main(void)
{
    float cm;     /* wzrost w cm */
    float stopy;  /* wzrost w stopach */
    float cale;   /* wzrost w calach */

    printf("Podaj wzrost w cm: ");
    scanf("%f",&cm);

    stopy = cm / 30.48f;
    cale = cm / 2.54f;

    printf("%f [cm] = %f [ft]\n",cm,stopy);
    printf("%f [cm] = %f [in]\n",cm,cale);

    return 0;
}
```

```
Podaj wzrost w cm: 175
175.000000 [cm] = 5.741470 [ft]
175.000000 [cm] = 68.897636 [in]
```

Język C - identyfikatory (nazwy)

- Dozwolone znaki: A-Z, a-z, 0-9, _ (podkreślenie)
- Długość nie jest ograniczona (rozróżnialne są 63 pierwsze znaki)
- Poprawne identyfikatory:

```
temp   u2    u_2   pole_kola   alfa   Beta   XyZ
```

- Pierwszym znakiem nie może być cyfra
- W identyfikatorach nie można stosować spacji, liter diakrytycznych
- Błędne identyfikatory:

```
2u   pole kola   pole_koła
```


Język C - identyfikatory (nazwy)

- Nie zaleca się, aby pierwszym znakiem było podkreślenie
- Identyfikatory nie powinny być zbyt długie

```
_temp   __temp   temperatura_w_skali_Celsjusza
```

- Nazwa zmiennej powinna być związana z jej zawartością
- Język C rozróżnia wielkość liter więc poniższe zapisy oznaczają inne identyfikatory

```
tempc   Tempc   TempC   TEMPC   TeMpC
```

- Jako nazw zmiennych nie można stosować słów kluczowych języka C

Język C - słowa kluczowe języka C

- W standardzie C11 zdefiniowane są 43 słowa kluczowe

```
auto        extern      short       while
break       float       signed      _Alignas
case        for         sizeof      _Alignof
char        goto        static      _Bool
const       if          struct      _Complex
continue    inline      switch      _Generic
default     int         typedef     _Imaginary
do          long        union       _Noreturn
double      register    unsigned    _Static_assert
else        restrict    void        _Thread_local
enum        return      volatile
```


Język C - Typy danych

Nazwa	Rozmiar (bajty)	Zakres wartości
char	1	-128 ... 127
int	4	-2147483648 ... 2147483647
float	4	$-3{,}4\cdot10^{38}$ ... $3{,}4\cdot10^{38}$
double	8	$-1{,}7\cdot10^{308}$ ... $1{,}7\cdot10^{308}$
void	–	–

- Słowa kluczowe wpływające na typy:
  - signed - liczba ze znakiem (dla typów char i int), np. signed char
  - unsigned - liczba bez znaku (dla typów char i int), np. unsigned int
  - short, long, long long - liczba krótka/długa (dla typu int), np. short int
  - long - większa precyzja (dla typu double), long double

Język C - Typy danych

- Zależnie od środowiska programistycznego (kompilatora) zmienne typów int i long double mogą zajmować różną liczbę bajtów

Środowisko	int (bajty)	long double (bajty)
Microsoft Visual Studio 2008	4	8
Microsoft Visual Studio 2015	4	8
Dev-C++ 5.11	4	12
Code::Blocks 16.01	4	12
Borland Turbo C++ 2006	4	10
Borland C++ 3.1	2	10

Język C - Typy danych (sizeof)

- sizeof - operator zwracający liczbę bajtów zajmowanych przez obiekt lub zmienną podanego typu

```
sizeof(nazwa_typu)
sizeof(nazwa_zmiennej)
sizeof nazwa_zmiennej
```

- Operator sizeof zwraca wartość typu size_t
- Zależnie od środowiska programistycznego typ size_t może odpowiadać typowi unsigned int lub unsigned long int
- W standardach C99 i C11 wprowadzono specyfikator formatu %zd przeznaczony do wyświetlania wartości typu size_t (Uwaga: nie działa w Visual Studio 2008)

Język C - Typy danych (sizeof)

```
#include <stdio.h>

int main(void)
{
    int x;

    printf("int: %d\n",sizeof(int));
    printf("int: %d\n",sizeof(x));
    printf("int: %d\n",sizeof x);

    printf("long double: %d\n",sizeof(long double));

    return 0;
}
```

```
int: 4
int: 4
int: 4
long double: 8
```


Język C - Stałe liczbowe (całkowite)

- Liczby całkowite (ang. integer) domyślnie zapisywane są w systemie dziesiętnym i mają typ int

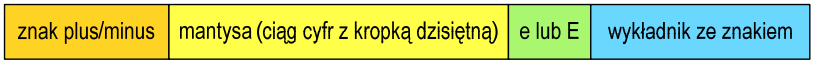
```
1    100    -125    123456
```

- Zapis liczb w innych systemach liczbowych
  - ósemkowy: 0 na początku, np. 011, 024
  - szesnastkowy: 0x na początku, np. 0x2F, 0xab
- Przyrostki na końcu liczby zmieniają typ
  - l lub L - typ long int, np. 10l, 10L , 011L , 0x2FL
  - ll lub LL - typ long long int, np. 10ll, 10LL , 011LL , 0x2FLL
  - u lub U - typ unsigned, np. 10u, 10U, 10lU, 10LLU, 0x2FUll

Język C - Stałe liczbowe (rzeczywiste)

- Domyślny typ liczb rzeczywistych to double
- Format zapisu stałych zmiennoprzecinkowych (ang. floating-point)

```
-2.41e+15   -2.41e+15   +4.123E-3   +4.123E-3
```



- W zapisie można pominąć:
  - znak plus, np. -2.41e15, 4.123E-3
  - kropkę dziesiętną lub część wykładniczą, np. 2e-5, 14.15
  - część ułamkową lub część całkowitą, np. 2.e-5, .12e4

Język C - Stałe liczbowe (rzeczywiste)

- W środku stałej zmiennoprzecinkowej nie mogą występować spacje
- Błędnie zapisane stałe zmiennoprzecinkowe:

```
- 2.41e+15   -2.41 e+15   -2.41e +15
```

- Przyrostki na końcu liczby zmieniają typ:
  - l lub L - typ long double, np. 2.5L, 1.24e7l
  - f lub F - typ float, np. 3.14f, 1.24e7F

Język C - Deklaracje zmiennych i stałych

- Zmienne (ang. variables) - zmieniają swoje wartości podczas pracy programu
- Stałe (ang. constants) - mają wartości ustalane przed uruchomieniem programu i pozostają niezmienione przez cały czas jego działania
- Deklaracja nadaje zmiennej / stałej nazwę, określa typ przechowywanej wartości i rezerwuje odpowiednio obszar pamięci
- Deklaracje zmiennych:

```
int x;
float a, b;
char zn1;
```

- Deklaracje stałych:

```
const int y = 5;
const float c = 1.25f;
const char zn2 = 'Q';
```

- Inicjalizacja zmiennej:

```
int x = -10;
```


Język C - Stałe symboliczne (#define)

- Dyrektywa preprocesora #define umożliwia definiowanie tzw. stałych symbolicznych

#define nazwa_stałej wartość_stałej

```
#define PI 3.14
#define KOMUNIKAT "Zaczynamy!!!\n"
```

- Wyrażenia stałe zazwyczaj pisze się wielkimi literami
- Wyrażenia stałe są obliczane przed właściwą kompilacją programu
- W kodzie programu w miejscu występowania stałej wstawiana jest jej wartość

Język C - Stałe symboliczne (#define)

```c
#include <stdio.h>
#define PI 3.14
#define KOMUNIKAT "Zaczynamy!!!\n"

int main(void)
{
    double pole, obwod;
    double r = 1.5;

    printf(KOMUNIKAT);
    pole = PI * r * r;
    obwod = 2 * PI * r;

    printf("Pole = %g\n", pole);
    printf("Obwod = %g\n", obwod);

    return 0;
}
```

```
Zaczynamy!!!
Pole = 7.065
Obwod = 9.42
```


Język C - Operatory

- Operator - symbol lub nazwa operacji
- Argumenty operatora nazywane są operandami
- Operator jednoargumentowy

- Operator dwuargumentowy

- Operator trójargumentowy

- Operator wieloargumentowy

Język C - Operatory

Typ	Symbol
Arytmetyczne	`+ - * / %`
Inkrementacji / dekrementacji	`++ --`
Porównania (relacyjne)	`< > <= >= == !=`
Logiczne	`&& || !`
Bitowe	`& | ^ << >> ~`
Przypisania	`= += -= *= /= %= <<= >>= &= |= ^=`
Inne	`() [] & * -> . , ? : sizeof (typ)`

Język C - Priorytet operatorów (1/2)

Priorytet	Operator / opis
1	`++ --` (przyrostki) `() [] . ->`
2	`++ --` (przedrostki) `sizeof (typ)` `+ - ! ~ * &` (jednoargumentowe)
3	`* / %`
4	`+ -` (dwuargumentowe)
5	`<< >>`
6	`< > <= >=`
7	`== !=`
8	`&` (bitowy)
9	`^`

Język C - Priorytet operatorów (2/2)

Priorytet	Operator / opis
10	|
11	&&
12	||
13	? :
14	= += -= *= /= %= <<= >>= &= |= ^=
15	, (przecinek)

Język C - Wyrażenia

- Wyrażenie (ang. expression) - kombinacja operatorów i operandów

```
4    -6    4+2.1    x=5+2    a>3    x>5&&x<8
```

- Każde wyrażenie ma typ i wartość

Wyrażenie	Typ	Wartość
4	int	4
-6	int	-6
4 + 2.1	double	6.1
x = 5 + 2	*typ x*	7
a > 3	int	1 (prawda) / 0 (fałsz)
x > 5 && x < 8	int	1 (prawda) / 0 (fałsz)

Język C - Instrukcje

- Instrukcja (ang. statement) - główny element, z którego zbudowany jest program, kończy się średnikiem

Wyrażenie: `x = 5` Instrukcja: `x = 5;`

- Język C za instrukcję uznaje każde wyrażenie, na którego końcu znajduje się średnik

```
8;
x;
3 + 4;
a > 5;
```

- Powyższe instrukcje są poprawne, ale nie dają żadnego efektu

Język C - Instrukcje

- Podział instrukcji:
  - proste - kończą się średnikiem
  - złożone - kilka instrukcji zawartych pomiędzy nawiasami klamrowymi
- Typy instrukcji prostych:
  - deklaracji: `int x;`
  - przypisania: `x = 5;`
  - wywołania funkcji: `printf("Witaj swiecie\n");`
  - strukturalna: `while(x > 0) x--;`
  - pusta: `;`

Język C - Wyrażenia arytmetyczne

- Wyrażenia arytmetyczne mogą zawierać:
  - stałe liczbowe, zmienne, stałe
  - operatory: `+ - * / % = ( )` i inne
  - wywołania funkcji (plik nagłówkowy math.h)
- Kolejność wykonywania operacji wynika z priorytetu operatorów

```
w = a + b;
```
→ `+` → `=`

```
w = a + b * c;
```
→ `*` → `+` → `=`

```
w = (a + b) * c;
```
→ `(+)` → `*` → `=`

```
w = (a + b) * (c + d);
```
→ `(+) lub (+)` → `*` → `=`

Język C - Wyrażenia arytmetyczne

- Kolejność wykonywania operacji

```
w = a + b + c;
```
→
```
w = ((a + b) + c);
```

```
w = x = y = a + b;
```
→
```
w = (x = (y = (a + b)));
```

- Zapis wyrażeń arytmetycznych

$$w = \frac{a+b}{c+d}$$

```
w = a + b / c + d;
```
ŹLE

```
w = (a + b) / (c + d);
```
DOBRZE

$$w = \frac{a+b}{c \cdot d}$$

```
w = (a + b) / c * d;
```
ŹLE

```
w = (a + b) / (c * d);
```
DOBRZE

Język C - Wyrażenia arytmetyczne

- Podczas dzielenia liczb całkowitych odrzucana jest część ułamkowa

$$w = \frac{5}{4}$$

```
5 / 4 = 1
```
```
5.0 / 4 = 1.25
```
```
5 / 4.0 = 1.25
```
```
5.0 / 4.0 = 1.25
```
```
5.0f / 4 = 1.25
```
```
5. / 4 = 1.25
```
```
(float) 5 / 4 = 1.25
```
Rzutowanie: (typ)

Język C - Funkcje matematyczne (math.h)

- Plik nagłówkowy math.h zawiera definicje wybranych stałych

Nazwa	Wartość	Znaczenie
`M_PI`	3.14159265358979323846	liczba pi
`M_E`	2.71828182845904523536	e - liczba Eulera
`M_LN2`	0.69314718055994530941 7	ln 2
`M_SQRT2`	1.41421356237309504880	$\sqrt{2}$

- W środowisku Visual Studio 2008 użycie stałych wymaga definicji odpowiedniej stałej (przed #include <math.h>)

```
#define _USE_MATH_DEFINES
#include <math.h>
```

Język C - Funkcje matematyczne (math.h)

- Wybrane funkcje matematyczne:

Nazwa	Nagłówek	Znaczenie
abs	int abs(int x);	moduł x (x - całkowite)
fabs	double fabs(double x);	moduł x (x - rzeczywiste)
sqrt	double sqrt(double x);	pierwiastek kwadratowy x
pow	double pow(double x, double y);	x^y - x do potęgi y
sin	double sin(double x);	sinus argumentu x w radianach
atan	double atan(double x);	arcus tangens argumentu x
atan2	double atan2(double y, double x);	arcus tangens ilorazu y/x

- Wszystkie funkcje mają po trzy wersje - dla argumentów typu: float, double i long double

Język C - Funkcja printf

- Ogólna składnia funkcji printf

```
printf("łańcuch_sterujący",arg1,arg2,...);
```

- W najprostszej postaci printf wyświetla tylko tekst

```
printf("Witaj swiecie");
```

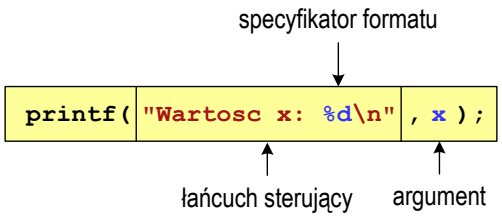
```
Witaj swiecie
```

- Do wyświetlenia wartości zmiennych konieczne jest zastosowanie specyfikatorów formatu, określających typ oraz sposób wyświetlania argumentów

```
%[znacznik][szerokość][.precyzja][modyfikator]typ
```


Język C - Funkcja printf

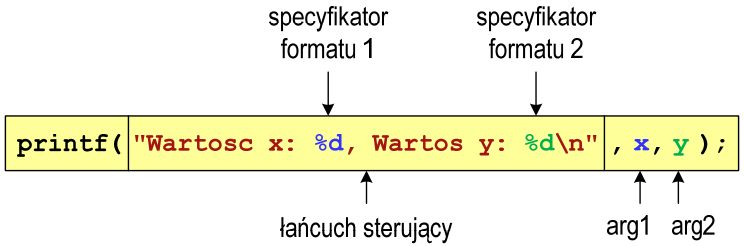
```
int x = 10;
printf("Wartosc x: %d\n", x);
```




```
Wartosc x: 10
```


Język C - Funkcja printf

```
int x = 10, y = 20;
printf("Wartosc x: %d, Wartosc y: %d\n", x, y);
```




```
Wartosc x: 10, Wartosc y: 20
```

Język C - Specyfikatory formatu (printf)

Typ w C	Specyfikator	Uwagi
char	%c	pojedynczy znak
	%d	kod ASCII znaku, liczba całkowita
char *	%s	łańcuch znaków, napis
int	%d %i	liczba całkowita, dziesiętna
	%o %O	liczba całkowita, ósemkowa
	%x %X	liczba całkowita, szesnastkowa
float double	%f	liczba rzeczywista
	%e %E	liczba rzeczywista, format naukowy
	%g %G	liczba rzeczywista (%f lub %e)

Język C - Funkcja printf

```
int x = 123; float y = 1.23456789f;
```

```
printf("x = [%d], y = [%f]\n", x, y);
printf("x = [], y = []\n", x, y);
printf("x = [%d], y = [%d]\n", x, y);
```

```
x = [123], y = [1.123457]
x = [], y = []
x = [123], y = [-536870912]
```


Język C - Funkcja printf

```
int x = 123; float y = 1.23456789f;
```

```
printf("x = [%6d], y = [%12f]\n", x, y);
printf("x = [%6d], y = [%12.3f]\n", x, y);
printf("x = [%6d], y = [%.3f]\n", x, y);
```

```
x = [   123], y = [    1.123457]
x = [   123], y = [       1.123]
x = [   123], y = [1.123]
```

```
%[znacznik][szerokość][.precyzja][modyfikator]typ
```


Język C - Funkcja printf

```
int x = 123; float y = 1.23456789f;
```

```
printf("x = [%+6d], y = [%+12f]\n", x, y);
printf("x = [%-6d], y = [%-12f]\n", x, y);
printf("x = [%06d], y = [%012f]\n", x, y);
```

```
x = [  +123], y = [   +1.123457]
x = [123   ], y = [1.123457    ]
x = [000123], y = [00001.123457]
```

```
%[znacznik][szerokość][.precyzja][modyfikator]typ
```

Język C - Funkcja printf

```
int x = 123; float y = 1.23456789f;
```

```
printf("x = [%d], y = [%f]\n", x, y);
printf("x = [%d], y = [%f]\n", x+321, y*25.5f);
printf("x = [%d], y = [%f]\n", 123, 2.0f*sqrt(y));
```

```
x = [123], y = [1.123457]
x = [444], y = [28.648149]
x = [123], y = [2.119865]
```


Język C - Funkcja scanf

- Ogólna składnia funkcji scanf

```
scanf("specyfikatory",adresy_argumentów);
```

- Składnia specyfikatora formatu

```
%[szerokość][modyfikator]typ
```

- Argumenty są adresami obszarów pamięci, dlatego muszą być poprzedzone znakiem &

```
int x;
scanf("%d", &x);
```


Język C - Funkcja scanf

- Specyfikatory formatu w większości przypadków są takie same jak w przypadku funkcji printf
- Największa różnica dotyczy typów float i double

Typ w C	Specyfikator	Uwagi
float	%f	liczba rzeczywista
	%e %E	liczba rzeczywista, format naukowy
	%g %G	liczba rzeczywista (%f lub %e)
double	%lf	liczba rzeczywista
	%le %lE	liczba rzeczywista, format naukowy
	%lg %lG	liczba rzeczywista (%f lub %e)

Język C - Funkcja scanf

```
int a, b, c;
scanf("%d %d %d", &a, &b, &c);
```

- Wczytywane argumenty mogą być oddzielone od siebie dowolną liczbą białych (niedrukowalnych) znaków: spacja, tabulacja, enter

```
15 20 -30
```

```
15 20 -30<enter>
```

```
   15   20   -30
```

```
   15   20   -30<enter>
```

```
15
20
-30
```

```
15<enter>
20<enter>
-30<enter>
```

Konwersja na system dziesiętny

- p = 4, D = {0, 1, 2, 3}

$4^4\ 4^3\ 4^2\ 4^1\ 4^0$
$2\ 1\ 3\ 0\ 2$

$$21302_{(4)} = ?_{(10)}$$
$$21302_{(4)} = 2\cdot 4^0 + 0\cdot 4^1 + 3\cdot 4^2 + 1\cdot 4^3 + 2\cdot 4^4$$
$$21302_{(4)} = 2\cdot 1 + 0\cdot 4 + 3\cdot 16 + 1\cdot 64 + 2\cdot 256$$
$$21302_{(4)} = 2 + 0 + 48 + 64 + 512 = 626_{(10)}$$

- p = 17, D = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, G}

$17^3\ 17^2\ 17^1\ 17^0$
$A\ C\ 2\ 4$

$$AC24_{(17)} = ?_{(10)}$$
$$AC24_{(17)} = 4\cdot 17^0 + 2\cdot 17^1 + 12\cdot 17^2 + 10\cdot 17^3$$
$$AC24_{(17)} = 4\cdot 1 + 2\cdot 17 + 12\cdot 289 + 10\cdot 4913$$
$$AC24_{(17)} = 4 + 34 + 3468 + 49130 = 52636_{(10)}$$

Konwersja na system dziesiętny (schemat Hornera)

- p = 4, D = {0, 1, 2, 3}

$$21302_{(4)} = w_{(10)} \qquad x_4\ x_3\ x_2\ x_1\ x_0 = w_{(10)}$$
$$w_{(10)} = 0$$
$$w_{(10)} = x_4 + w_{(10)} \cdot p = 2 + 0\cdot 4 = 2$$
$$w_{(10)} = x_3 + w_{(10)} \cdot p = 1 + 2\cdot 4 = 9$$
$$w_{(10)} = x_2 + w_{(10)} \cdot p = 3 + 9\cdot 4 = 39$$
$$w_{(10)} = x_1 + w_{(10)} \cdot p = 0 + 39\cdot 4 = 156$$
$$w_{(10)} = x_0 + w_{(10)} \cdot p = 2 + 156\cdot 4 = 626_{(10)}$$

Konwersja z systemu dziesiętnego na dowolny

- zamiana liczby z systemu p = 10 na system p = 2

$$626_{(10)} = ?_{(2)} \qquad 626_{(10)} = 1001110010_{(2)}$$

626 / 2	=	313	*reszta*	0
313 / 2	=	156	*reszta*	1
156 / 2	=	78	*reszta*	0
78 / 2	=	39	*reszta*	0
39 / 2	=	19	*reszta*	1
19 / 2	=	9	*reszta*	1
9 / 2	=	4	*reszta*	1
4 / 2	=	2	*reszta*	0
2 / 2	=	1	*reszta*	0
1 / 2	=	0	*reszta*	1

kolejność odczytywania cyfr liczby w systemie dwójkowym

kończymy, gdy liczba dziesiętna ma wartość 0

Konwersja z systemu dziesiętnego na dowolny

- zamiana liczby z systemu p = 10 na system p = 7

$$626_{(10)} = ?_{(7)} \qquad 626_{(10)} = 1553_{(7)}$$

626 / 7	=	89	*reszta*	3
89 / 7	=	12	*reszta*	5
12 / 7	=	1	*reszta*	5
1 / 7	=	0	*reszta*	1

- zamiana liczby z systemu p = 10 na system p = 14

$$626_{(10)} = ?_{(14)} \qquad 626_{(10)} = 32A_{(14)}$$

626 / 14	=	44	*reszta*	10 → A
44 / 14	=	3	*reszta*	2
3 / 14	=	0	*reszta*	3

Szybkie konwersje: 2 → 4,8,16 4,8,16 → 2

2 → 4

$110110011_{(2)} = ?_{(4)}$

$\underbrace{01}_{1} \mid \underbrace{10}_{2} \mid \underbrace{11}_{3} \mid \underbrace{00}_{0} \mid \underbrace{11}_{3}$

$110110011_{(2)} = 12303_{(4)}$

2 → 8

$10110011_{(2)} = ?_{(8)}$

$\underbrace{010}_{2} \mid \underbrace{110}_{6} \mid \underbrace{011}_{3}$

$10110011_{(2)} = 263_{(8)}$

2 → 16

$1011010_{(2)} = ?_{(16)}$

$\underbrace{0101}_{5} \mid \underbrace{1010}_{A}$

$1011010_{(2)} = 5A_{(16)}$

4 → 2

$12303_{(4)} = ?_{(2)}$

$\overset{1}{01}\ \overset{2}{10}\ \overset{3}{11}\ \overset{0}{00}\ \overset{3}{11}$

$12303_{(4)} = 110110011_{(2)}$

8 → 2

$263_{(8)} = ?_{(2)}$

$\overset{2}{010}\ \overset{6}{110}\ \overset{3}{011}$

$263_{(8)} = 10110011_{(2)}$

16 → 2

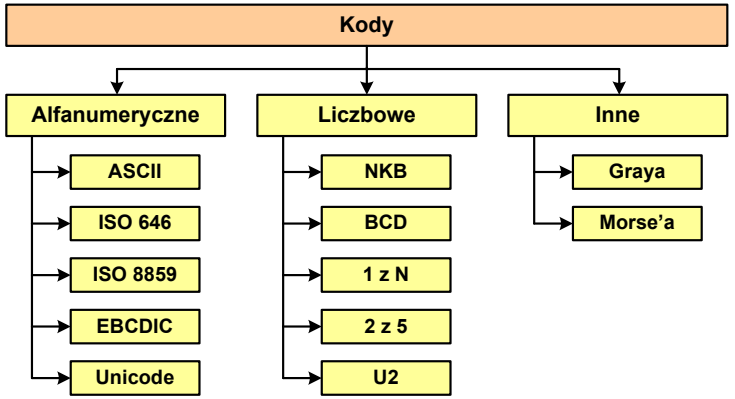
$5A_{(16)} = ?_{(2)}$

$\overset{5}{0101}\ \overset{A}{1010}$

$5A_{(16)} = 1011010_{(2)}$

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 tramsmission)	20	14	DC4	(device control 4)
5	5	ENQ	(enquiry)	21	15	NAK	(negative acknowlege)
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	(subtitute)
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 - $10_{(10)} = 0A_{(16)}$
- Załóżmy, 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 6B 75 0D 0A 44|72 75 67 69 20 77 69 65 | pliku■■Drugi wie
00000020: 72 73 7A 20 70 6C 69 6B|75 0D 0A 54 72 7A 65 63 | rsz pliku■■Trzec
00000030: 69 20 77 69 65 72 73 7A|20 70 6C 69 6B 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)

Kod ASCII - Pliki tekstowe

- W systemie Linux znakiem końca wiersza jest tylko LF o kodzie ASCII - $10_{(10)} = 0A_{(16)}$
- Załóżmy, 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 6B 75 0A 44 72|75 67 69 20 77 69 65 72 | pliku■Drugi wier
00000020: 73 7A 20 70 6C 69 6B 75|0A 54 72 7A 65 63 69 20 | sz pliku■Trzeci
00000030: 77 69 65 72 73 7A 20 70|6C 69 6B 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 pliku■Drugi wiersz pliku■Trzeci wiersz pliku■
```


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																
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	{	|	}	~	

- żółty - znaki narodowe
- niebieski - znaki walut

- Wszystkie pozostałe znaki są zgodne z ASCII

ISO/IEC 646 - odmiany narodowe

USA

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	*Znaki kontrolne*															
10																
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	{	|	}	~	

Niemcy

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	*Znaki kontrolne*															
10																
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	Ä	Ö	Ü	^	_
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	ä	ö	ü	ß	

Francja

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	*Znaki kontrolne*															
10																
20	SP	!	"	£	$	%	&	`	(	)	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	à	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	°	ç	§	^	_
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	é	ù	é	¨	

Polska

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	*Znaki kontrolne*															
10																
20	SP	!	"	#	ż	%	&	`	(	)	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	ę	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	ź	\	ń	ś	_
60	ą	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	ó	ł	ż	ć	

Norma: BN-74/3101-01

ISO/IEC 8859

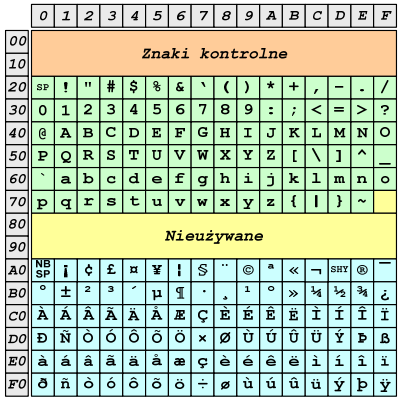
- ISO/IEC 8859 - zestaw standardów służących do kodowania znaków za pomocą 8-bitów
- Wszystkie zestawy ISO 8859 mają znaki $0_{(10)}$-$127_{(10)}$ ($00_{(16)}$-$7F_{(16)}$) takie same jak w kodzie ASCII
- Pozycjom $128_{(10)}$-$159_{(10)}$ ($80_{(16)}$-$9F_{(16)}$) przypisane są dodatkowe kody sterujące,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

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 8859-1

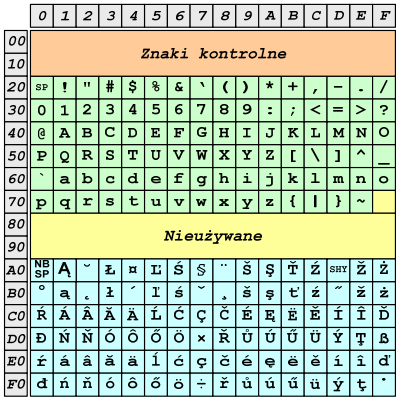
- ISO/IEC 8859-1, Latin-1 („zachodnioeuropejskie”)
- kodowanie używane w Amerykach, Europie Zachodniej, Oceanii i większej części Afryki
- dostępne języki: albański, angielski, baskijski, duński, estoński, fiński, francuski, hiszpański, irlandzki, islandzki, kataloński, łaciński, niderlandzki, niemiecki,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



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

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
- kodowanie zgodne z Polską Normą.



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
  - Ź - ź
  - Ż - ż

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	Znaki kontrolne															
10																
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	Nieużywane															
90																
A0	NB SP	Ą	˘	Ł	¤	Ľ	Ś	§	¨	Š	Ş	Ť	Ź	SHY	Ž	Ż
B0	°	ą	˛	ł	´	ľ	ś	ˇ	¸	š	ş	ť	ź	˝	ž	ż
C0	Ŕ	Á	Â	Ă	Ä	Ĺ	Ć	Ç	Č	É	Ę	Ë	Ě	Í	Î	Ď
D0	Đ	Ń	Ň	Ó	Ô	Ő	Ö	×	Ř	Ů	Ú	Ű	Ü	Ý	Ţ	ß
E0	ŕ	á	â	ă	ä	ĺ	ć	ç	č	é	ę	ë	ě	í	î	ď
F0	đ	ń	ň	ó	ô	ő	ö	÷	ř	ů	ú	ű	ü	ý	ţ	˙

ISO/IEC 8859-1 i ISO/IEC 8859-2 - porównanie

ISO 8859-1

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A0	NB SP	¡	¢	£	¤	¥	¦	§	¨	©	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

ISO 8859-2

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A0	NB SP	Ą	˘	Ł	¤	Ľ	Ś	§	¨	Š	Ş	Ť	Ź	SHY	Ž	Ż
B0	°	ą	˛	ł	´	ľ	ś	ˇ	¸	š	ş	ť	ź	˝	ž	ż
C0	Ŕ	Á	Â	Ă	Ä	Ĺ	Ć	Ç	Č	É	Ę	Ë	Ě	Í	Î	Ď
D0	Đ	Ń	Ň	Ó	Ô	Ő	Ö	×	Ř	Ů	Ú	Ű	Ü	Ý	Ţ	ß
E0	ŕ	á	â	ă	ä	ĺ	ć	ç	č	é	ę	ë	ě	í	î	ď
F0	đ	ń	ň	ó	ô	ő	ö	÷	ř	ů	ú	ű	ü	ý	ţ	˙

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.

EBCDIC

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

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																
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	Nieużywane															
90																
A0	NB SP	¡	¢	£	¤	¥	¦	§	¨	©	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

EBCDIC

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

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

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	Znaki kontrolne															
10																
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	Nieużywane															
90																
A0	NB SP	Ą	˘	Ł	¤	Ľ	Ś	§	¨	Š	Ş	Ť	Ź	SHY	Ž	Ż
B0	°	ą	˛	ł	´	ľ	ś	ˇ	¸	š	ş	ť	ź	˝	ž	ż
C0	Ŕ	Á	Â	Ă	Ä	Ĺ	Ć	Ç	Č	É	Ę	Ë	Ě	Í	Î	Ď
D0	Đ	Ń	Ň	Ó	Ô	Ő	Ö	×	Ř	Ů	Ú	Ű	Ü	Ý	Ţ	ß
E0	ŕ	á	â	ă	ä	ĺ	ć	ç	č	é	ę	ë	ě	í	î	ď
F0	đ	ń	ň	ó	ô	ő	ö	÷	ř	ů	ú	ű	ü	ý	ţ	˙

Windows-1250

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	Znaki kontrolne															
10																
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	€		‚		„	…	†	‡		‰	Š	‹	Ś	Ť	Ž	Ź
90		‘	’	“	”	•	–	—		™	š	›	ś	ť	ž	ź
A0	NB SP	ˇ	˘	Ł	¤	Ą	¦	§	¨	©	Ş	«	¬	SH Y	Ž	Ż
B0	°	±	˛	ł	´	µ	¶	·	¸	ą	ş	»	Ľ	˝	ľ	ż
C0	Ŕ	Á	Â	Ă	Ä	Ĺ	Ć	Ç	Č	É	Ę	Ë	Ě	Í	Î	Ď
D0	Đ	Ń	Ň	Ó	Ô	Ő	Ö	×	Ř	Ů	Ú	Ű	Ü	Ý	Ţ	ß
E0	ŕ	á	â	ă	ä	ĺ	ć	ç	č	é	ę	ë	ě	í	î	ď
F0	đ	ń	ň	ó	ô	ő	ö	÷	ř	ů	ú	ű	ü	ý	ţ	˙

Problem kodowania polskich liter diakrytycznych

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

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

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

```
˘ Ć Ę Ł Ń Ó ¦ ¬ Ż
± ć ę ł ń ó ¶ Ľ ż
```


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 10.0 (20.06.2017)
  - The Unicode Consortium. The Unicode Standard, Version 10.0.0, (Mountain View, CA: The Unicode Consortium, 2017)
  - http://www.unicode.org/versions/Unicode10.0.0/
  - koduje 136.690 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
...	

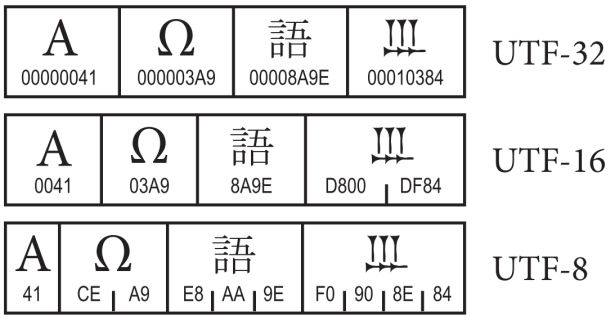
Unicode

- 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 opisania kodu znaku



źródło: The Unicode Consortium. The Unicode Standard, Version 8.0

Unicode - kodowanie UTF-32

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

- 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	Ω	語	𝌆	
0041	03A9	8A9E	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	Ω	語	𝌆			
41	CE	A9	E8	AA	9E	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 0100	Đ 0110	Ġ 0120	İ 0130	ŀ· 0140	Ő 0150	Š 0160	Ű 0170
1	ā 0101	đ 0111	ġ 0121	ı 0131	Ł 0141	ő 0151	š 0161	ű 0171
2	Ă 0102	Ē 0112	Ģ 0122	IJ 0132	ł 0142	Œ 0152	Ţ 0162	Ų 0172
3	ă 0103	ē 0113	ģ 0123	ij 0133	Ń 0143	œ 0153	ţ 0163	ų 0173
4	Ą 0104	Ĕ 0114	Ĥ 0124	Ĵ 0134	ń 0144	Ŕ 0154	Ť 0164	Ŵ 0174
5	ą 0105	ĕ 0115	ĥ 0125	ĵ 0135	Ņ 0145	ŕ 0155	ť 0165	ŵ 0175
6	Ć 0106	Ė 0116	Ħ 0126	Ķ 0136	ņ 0146	Ŗ 0156	Ŧ 0166	Ŷ 0176
7	ć 0107	ė 0117	ħ 0127	ķ 0137	Ň 0147	ŗ 0157	ŧ 0167	ŷ 0177

European Latin

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

Unicode

27308 **CJK Unified Ideographs Extension B** **27342**

27308 虫 142.8	婦 UCS2003	婦 GKX-1086.03	婦 T4-4721	2731B 虫 142.8	𗈛 UCS2003	𗈛 GKX-1088.15	𗈛 T6-617B	2732F 虫 142.8	蝉 UCS2003	蝉 GHC
27309 虫 142.8	蛙 UCS2003	蛙 GKX-1086.05	蛙 T5-4955	2731C 虫 142.8	𗈜 UCS2003	𗈜 GKX-1088.16	𗈜 T6-6221	27330 虫 142.9	蟬 UCS2003	蟬 GHC
2730A 虫 142.8	蛾 UCS2003	蛾 GKX-1086.08	蛾 T4-467D	2731D 虫 142.8	𗈝 UCS2003	𗈝 GKX-1088.17	𗈝 T5-4960	27331 虫 142.8	螯 UCS2003	螯 G4K
2730B 虫 142.8	蜘 UCS2003	蜘 GKX-1086.10	蜘 T6-6223	2731E 虫 142.7	𗈠 UCS2003	𗈠 GKX-1088.18		27332 虫 142.8	蟹 UCS2003	蟹 GHC
2730C 虫 142.8	蟹 UCS2003	蟹 GKX-1086.12	蟹 T5-495F	2731F 虫 142.8	𗈡 UCS2003	𗈡 GKX-1088.19	𗈡 T6-6174	27333 虫 142.8	蠅 UCS2003	蠅 GHC
2730D 虫 142.8	蜍 UCS2003	蜍 GKX-1086.22	蜍 T4-4677	27320 虫 142.8	𗈢 UCS2003	𗈢 GKX-1088.20	𗈢 T6-617D	27334 虫 142.8	蜷 UCS2003	蜷 T5-4953

Koniec wykładu nr 2

Dziękuję za uwagę!

(następny wykład: 16.03.2018)