

Informatyka 1

Politechnika Białostocka - Wydział Elektryczny
Elektrotechnika, semestr II, studia niestacjonarne I stopnia
Rok akademicki 2018/2019

Wykład nr 2 (15.03.2019)

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
 - funkcje printf i scanf
- Konwersje między systemami liczbowymi
- Jednostki informacji cyfrowej
 - bit, bajt

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 (rozdzielalne 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	$-3,4 \cdot 10^{38}$... $3,4 \cdot 10^{38}$
double	8	$-1,7 \cdot 10^{308}$... $1,7 \cdot 10^{308}$
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`, `10IU`, `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
```

znak plus/minus	mantysa (ciąg cyfr z kropką dziesiętną)	e lub E	wykładnik ze znakiem
-----------------	-----------------------------------------	---------	----------------------

- 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 ustalone 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)

```
#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;  
}
```

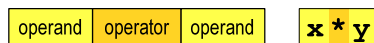
```
Zaczynamy!!!  
Pole = 7.065  
Obwod = 9.42
```

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

$$w = a + b + c; \rightarrow w = ((a + b) + c);$$

$$w = x = y = a + b; \rightarrow w = (x = (y = (a + b)));$$

- Zapis wyrażeń arytmetycznych

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

$$w = a + b / c + d; \quad \text{ŹLE}$$

$$w = (a + b) / (c + d); \quad \text{DOBRZE}$$

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

$$w = (a + b) / c * d; \quad \text{ŹLE}$$

$$w = (a + b) / (c * d); \quad \text{DOBRZE}$$

Język C - Funkcje matematyczne (math.h)

- Plik nagłówkowy `math.h` zawiera definicje wybranych stałych

Nazwa	Wartość	Znaczenie
<code>M_PI</code>	3.14159265358979323846	liczba pi
<code>M_E</code>	2.71828182845904523536	e - liczba Eulera
<code>M_LN2</code>	0.693147180559945309417	ln 2
<code>M_SQRT2</code>	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 - Wyrażenia arytmetyczne

- Podczas dzielenia liczb całkowitych odrzucana jest część ułamkowa

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

$$5 / 4 = 1$$

$$5.0 / 4 = 1.25$$

$$5 / 4.0 = 1.25$$

$$5.0 / 4.0 = 1.25$$

$$5.0f / 4 = 1.25$$

$$5. / 4 = 1.25$$

$$(\text{float}) 5 / 4 = 1.25$$

Rzutowanie:
(`typ`) wyrażenie

Język C - Funkcje matematyczne (math.h)

- Wybrane funkcje matematyczne:

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

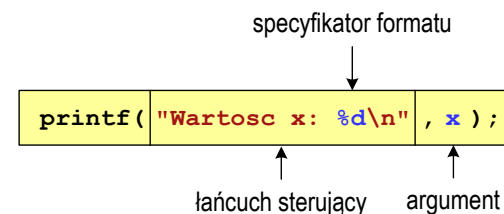
```
Witaj świecie
```

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

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

Język C - Funkcja printf

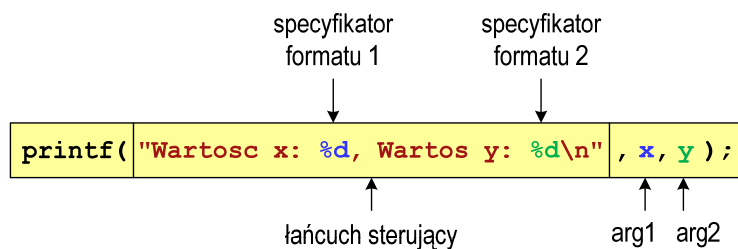
```
int x = 10;  
printf("Wartosc x: %d\n", x);
```



```
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]
```

%[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 = [%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>
```

Przykład: częstotliwość rezonansowa

```
#include <stdio.h>  
#define _USE_MATH_DEFINES  
#include <math.h>  
  
int main(void)  
{  
    double R, L, C, fr;  
  
    printf("Podaj R [Om]: "); scanf("%lf", &R);  
    printf("Podaj L [H]: "); scanf("%lf", &L);  
    printf("Podaj C [F]: "); scanf("%lf", &C);  
  
    fr = 1 / (2 * M_PI * sqrt(L * C));  
  
    printf("-----\n");  
    printf("fr [Hz]: %.3f\n", fr);  
  
    return 0;  
}
```

```
Podaj R [Om]: 100  
Podaj L [H]: 0.01  
Podaj C [F]: 1e-6  
-----  
fr [Hz]: 1591.549
```

$$f_r = \frac{1}{2\pi\sqrt{LC}}$$

Konwersja na system dziesiętny

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

$$21302_{(4)} = ?_{(10)}$$

$$21302_{(4)} = 2 \cdot 4^0 + 0 \cdot 4^1 + 3 \cdot 4^2 + 1 \cdot 4^3 + 2 \cdot 4^4$$

$$21302_{(4)} = 2 \cdot 1 + 0 \cdot 4 + 3 \cdot 16 + 1 \cdot 64 + 2 \cdot 256$$

$$21302_{(4)} = 2 + 0 + 48 + 64 + 512 = 626_{(10)}$$

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

$$AC24_{(17)} = ?_{(10)}$$

$$AC24_{(17)} = 4 \cdot 17^0 + 2 \cdot 17^1 + 12 \cdot 17^2 + 10 \cdot 17^3$$

$$AC24_{(17)} = 4 \cdot 1 + 2 \cdot 17 + 12 \cdot 289 + 10 \cdot 4913$$

$$AC24_{(17)} = 4 + 34 + 3468 + 49130 = 52636_{(10)}$$

Konwersja na system dziesiętny (schemat Hornera)

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

$$21302_{(4)} = w_{(10)} \quad x_4 \ x_3 \ x_2 \ x_1 \ x_0 = w_{(10)}$$

$$w_{(10)} = 0$$

$$w_{(10)} = x_4 + w_{(10)} \cdot p = 2 + 0 \cdot 4 = 2$$

$$w_{(10)} = x_3 + w_{(10)} \cdot p = 1 + 2 \cdot 4 = 9$$

$$w_{(10)} = x_2 + w_{(10)} \cdot p = 3 + 9 \cdot 4 = 39$$

$$w_{(10)} = x_1 + w_{(10)} \cdot p = 0 + 39 \cdot 4 = 156$$

$$w_{(10)} = x_0 + w_{(10)} \cdot p = 2 + 156 \cdot 4 = 626_{(10)}$$

Konwersja z systemu dziesiętnego na dowolny

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

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

$626/2 = 313$	$reszta$	0	↑ kolejność odczytywania cyfr liczby w systemie dwójkowym
$313/2 = 156$	$reszta$	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	

Konwersja z systemu dziesiętnego na dowolny

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

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

$626/7 = 89$	$reszta$	3	↑
$89/7 = 12$	$reszta$	5	
$12/7 = 1$	$reszta$	5	
$1/7 = 0$	$reszta$	1	

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

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

$626/14 = 44$	$reszta$	$10 \rightarrow A$	↑
$44/14 = 3$	$reszta$	2	
$3/14 = 0$	$reszta$	3	

Szybkie konwersje: $2 \rightarrow 4, 8, 16$ $4, 8, 16 \rightarrow 2$

$2 \rightarrow 4$

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

$$\begin{array}{c|c|c|c|c} 01 & 10 & 11 & 00 & 11 \\ \hline 1 & 2 & 3 & 0 & 3 \end{array}$$

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

$2 \rightarrow 8$

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

$$\begin{array}{c|c|c} 010 & 110 & 011 \\ \hline 2 & 6 & 3 \end{array}$$

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

$2 \rightarrow 16$

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

$$\begin{array}{c|c} 0101 & 1010 \\ \hline 5 & A \end{array}$$

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

$4 \rightarrow 2$

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

$$\begin{array}{c|c|c|c|c} 1 & 2 & 3 & 0 & 3 \\ \hline 01 & 10 & 11 & 00 & 11 \end{array}$$

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

$8 \rightarrow 2$

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

$$\begin{array}{c|c|c} 2 & 6 & 3 \\ \hline 010 & 110 & 011 \end{array}$$

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



$16 \rightarrow 2$

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

$$\begin{array}{c|c} 5 & A \\ \hline 0101 & 1010 \end{array}$$

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

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:

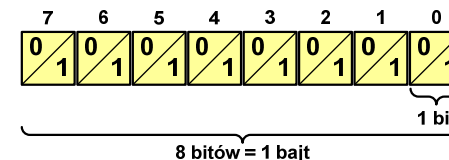
Przedrostki dziesiętne (układ SI)		
Nazwa	Symbol	Mnożnik
bit	b	---
kilobit	kb	$10^3 = 1000^1$
megabit	Mb	$10^6 = 1000^2$
gigabit	Gb	$10^9 = 1000^3$
terabit	Tb	$10^{12} = 1000^4$
petabit	Pb	$10^{15} = 1000^5$
eksabit	Eb	$10^{18} = 1000^6$
zettabit	Zb	$10^{21} = 1000^7$
jottabit	Yb	$10^{24} = 1000^8$

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 1998 roku w celu odróżnienia przedrostków o mnożniku 1000 (10^3) od przedrostków o mnożniku 1024 (2^{10})

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

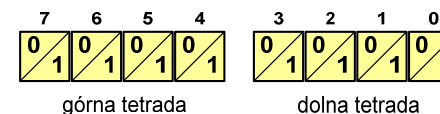
Jednostki informacji - bajt

- 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 - **decybela (dB)** więc nie ma problemu z rozróżnieniem obu jednostek

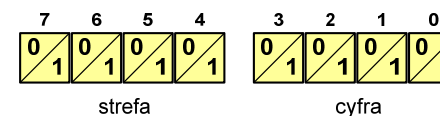


Jednostki informacji - tetrada

- Bajt 8-bitowy można podzielić na dwie połówki 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



Koniec wykładu nr 2

Dziękuję za uwagę!
(następny wykład: 22.03.2019)