



Wydział Elektryczny
Katedra Elektrotechniki Teoretycznej i Metrologii

Materiały do wykładu z przedmiotu:
Informatyka
Kod: **EDS1B1007**

WYKŁAD NR 3

Opracował: **dr inż. Jarosław Forenc**
Białystok 2019

Materiały zostały opracowane w ramach projektu „PB2020 - Zintegrowany Program Rozwoju Politechniki Białostockiej” realizowanego w ramach Działania 3.5 Programu Operacyjnego Wiedza, Edukacja, Rozwój 2014-2020 współfinansowanego ze środków Europejskiego Funduszu Społecznego.

Plan wykładu nr 3

- Instrukcje, wyrażenia arytmetyczne
- Funkcje printf i scanf

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

<code>w = a + b;</code>	<code>+</code> → <code>=</code>
<code>w = a + b * c;</code>	<code>*</code> → <code>+</code> → <code>=</code>
<code>w = (a + b) * c;</code>	<code>(+)</code> → <code>*</code> → <code>=</code>
<code>w = (a + b) * (c + d);</code>	<code>(+)</code> lub <code>(+)</code> → <code>*</code> → <code>=</code>

Język C - Wyrażenia arytmetyczne

- Kolejność wykonywania operacji

<code>w = a + b + c;</code>	→	<code>w = ((a + b) + c);</code>
<code>w = x = y = a + b;</code>	→	<code>w = (x = (y = (a + b)));</code>

- Zapis wyrażeń arytmetycznych

$w = \frac{a+b}{c+d}$	<code>w = a + b / c + d;</code>	ŹLE
	<code>w = (a + b) / (c + d);</code>	DOBRCZE
$w = \frac{a+b}{c \cdot d}$	<code>w = (a + b) / c * d;</code>	ŹLE
	<code>w = (a + b) / (c * d);</code>	DOBRCZE

Język C - Wyrażenia arytmetyczne

- Podczas dzielenia liczb całkowitych odrzucana jest część ułamkowa

$w = \frac{5}{4}$	<code>5 / 4 = 1</code>
	<code>5.0 / 4 = 1.25</code>
	<code>5 / 4.0 = 1.25</code>
	<code>5.0 / 4.0 = 1.25</code>
	<code>5.0f / 4 = 1.25</code>
	<code>5. / 4 = 1.25</code>
	<code>(float) 5 / 4 = 1.25</code>

Rzutowanie: (typ)

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 - 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");
```

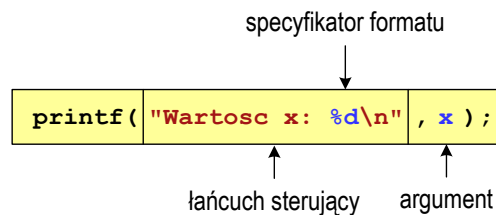
```
Witaj świecie
```

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

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

Język C - Funkcja printf

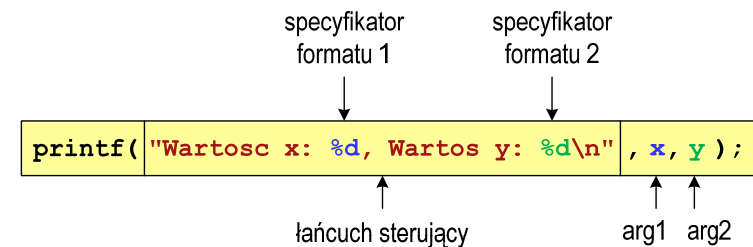
```
int x = 10;  
printf("Wartosc x: %d\n", x);
```



```
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);
```

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

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

```
x = [], y = []
```

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

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

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

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

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

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

Język C - Funkcja printf

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

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

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

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

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

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

```
x = [000123], y = [00001.123457]
```

Język C - Funkcja printf

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

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

```
x = [444], y = [28.648149]
```

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

```
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
```

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

Koniec wykładu nr 3

Dziękuję za uwagę!