# Informatyka 1 (EZ1E2008)

Politechnika Białostocka - Wydział Elektryczny

Elektrotechnika, semestr II, studia niestacjonarne I stopnia

Rok akademicki 2021/2022

## Wykład nr 2 (11.03.2022)

dr inż. Jarosław Forenc



## Plan wykładu nr 2

- Język C
  - stałe liczbowe, deklaracje zmiennych i stałych
  - operatory, priorytet operatorów, wyrażenia, instrukcje
  - wyrażenia arytmetyczne, funkcje matematyczne (math.h)
  - funkcje printf i scanf



## Przykład: zamiana wzrostu w cm na stopy i cale

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

```
#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;
}
```



## 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
- W miejscu występowania stałej wstawiana jest jej wartość (przed właściwą kompilacją programu)

# Przykład: pole i obwód koła

```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;
}
```



# Przykład: pole i obwód koła

```c
/***
...
#endif  /* _INC_STDIO */

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

    printf("Zaczynamy!!!\n");
    pole = 3.14 * r * r;
    obwod = 2 * 3.14 * r;

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

    return 0;
}
```

zawartość pliku stdio.h

```
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 | 0.69314718055994530942 | 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



# Przykład: częstotliwość rezonansowa

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

int main(void)
{
    double L, C, fr;

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

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



# Język C - Funkcja printf

- Ogólna składnia funkcji printf

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

- W najprostszej postaci printf wyświetla tylko tekst

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

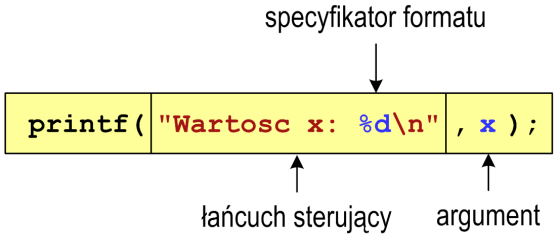
```
Witaj swiecie
```

- 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);
```

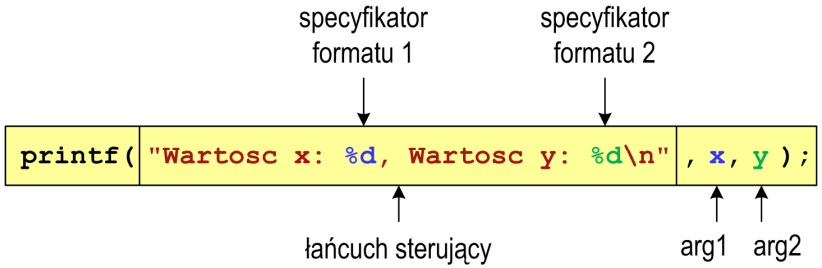



```
Wartosc x: 10
```



# 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<br>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 = [1.234568]
```

```
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 = [    1.234568]
```

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

```
x = [   123], y = [       1.235]
```

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

```
x = [   123], y = [1.235]
```



# Język C - Funkcja printf

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

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

```
x = [  +123], y = [   +1.234568]
```

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

```
x = [123   ], y = [1.234568    ]
```

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

```
x = [000123], y = [00001.234568]
```



# 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 = [31.481482]
```

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

```
x = [123], y = [2.222222]
```



# 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: pierwiastek kwadratowy

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

int main(void)
{
    float x, y;

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

    y = sqrt(x);

    printf("Pierwiastek liczby: %f\n",y);

    return 0;
}
```

```
Podaj liczbe: 15
Pierwiastek liczby: 3.872983
```

```
Podaj liczbe: -15
Pierwiastek liczby: -1.#IND00
```



# 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: 15
Pierwiastek liczby: 3.872983
```

```
Podaj liczbe: -15
Blad! Liczba ujemna
```

# Koniec wykładu nr 2

Dziękuję za uwagę!