



Fundusze Europejskie
Wiedza Edukacja Rozwój



Rzeczpospolita
Polska

Unia Europejska
Europejski Fundusz Społeczny



Wydział Elektryczny
Katedra Elektrotechniki Teoretycznej i Metrologii

Materiały do wykładu z przedmiotu:

Informatyka

Kod: EDS1B1007

WYKŁAD NR 2

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 2

- Identyfikatory (nazwy), słowa kluczowe
- Typy danych, stałe liczbowe, deklaracje zmiennych i stałych
- Operatory, priorytet operatorów
- Wyrażenia

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

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

Język C - Typy danych

Nazwa	Rozmiar (bajty)	Zakres wartości
<code>char</code>	1	-128 ... 127
<code>int</code>	4	-2147483648 ... 2147483647
<code>float</code>	4	$-3,4 \cdot 10^{38}$... $3,4 \cdot 10^{38}$
<code>double</code>	8	$-1,7 \cdot 10^{308}$... $1,7 \cdot 10^{308}$
<code>void</code>	-	-

- 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 niezmiennione 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_stalej wartość_stalej

```
#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;
}
```

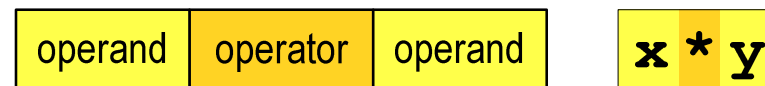
Zaczynamy!!!
Pole = 7.065
Obwod = 9.42

Język C - Operatory

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



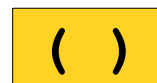
- Operator dwuargumentowy



- 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	<i>typ x</i>	7
a > 3	int	1 (prawda) / 0 (fałsz)
x > 5 && x < 8	int	1 (prawda) / 0 (fałsz)

Koniec wykładu nr 2

Dziękuję za uwagę!