

Wydział Elektryczny
Katedra Elektrotechniki, Energoelektroniki i Elektroenergetyki

Materiały do wykładu z przedmiotu:
Informatyka
Kod: EDS1B1007

WYKŁAD NR 1

Opracował: **dr inż. Jarosław Forenc**
Białystok 2021

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.

Dane podstawowe

- dr inż. Jarosław Forenc
- Politechnika Białostocka, Wydział Elektryczny, Katedra Elektrotechniki, Energoelektroniki i Elektroenergetyki ul. Wiejska 45D, 15-351 Białystok WE-204
- e-mail: j.forenc@pb.edu.pl
- tel. (0-85) 746-93-97
- <http://jforenc.prv.pl>
 - Dydaktyka - dodatkowe materiały do pracowni
- Konsultacje
 - poniedziałek, 16:00-17:30, WE-204 / Teams
 - wtorek, 14:00-15:30, WE-204 / Teams
 - sobota, 10:00-11:00, 13:45-15:00, WE-204 / Teams (zaoczne)

Program wykładu (1/2)

1. **Programowanie w języku C.** Deklaracje i typy zmiennych, operatory i wyrażenia arytmetyczne, operacje wejścia-wyjścia.
2. **Programowanie w języku C.** Operatory relacyjne i logiczne, wyrażenia logiczne, instrukcja warunkowa if, instrukcja wyboru wielowariantowego switch, operator warunkowy, pętle (for, while, do .. while).
3. **Programowanie w języku C.** Tablice jedno- i dwuwymiarowe, łańcuchy znaków, struktury, wskaźniki, dynamiczny przydział pamięci.
4. **Programowanie w języku C.** Funkcje użytkownika, przekazywanie argumentów do funkcji, rekurencyjne wywołanie funkcji, pliki tekstowe i binarne.

Program wykładu (2/2)

5. **Algorytmy komputerowe.** Definicja algorytmu. Klasyfikacje, sposoby przedstawiania i złożoność obliczeniowa algorytmów.
6. **Budowa i zasada działania komputera.** Procesor, pamięć wewnętrzna i zewnętrzna. Komunikacja z urządzeniami zewnętrznymi, interfejsy komputerowe.
7. **System operacyjny.** Zarządzanie procesami, pamięcią i dyskowymi operacjami wejścia-wyjścia (systemy plików).
8. Zaliczenie wykładu.

Literatura (1/2)

1. S. Prata: „Język C. Szkoła programowania. Wydanie VI”. Helion, Gliwice, 2016.
2. B.W. Kernighan, D.M. Ritchie: „Język ANSI C. Programowanie. Wydanie II”. Helion, Gliwice, 2010.
3. P.J. Deitel, H. Deitel: „Język C. Solidna wiedza w praktyce. Wydanie VIII”. Helion, Gliwice, 2020.
4. R. Reese: „Wskaźniki w języku C. Przewodnik”. Helion, Gliwice, 2014.

Literatura (2/2)

5. R. Kawa, J. Lembas: „Wykłady z informatyki. Wstęp do informatyki”. PWN, Warszawa, 2021.
6. P. Wróblewski: „Algorytmy, struktury danych i techniki programowania”. Wydanie VI. Helion, Gliwice, 2019.
7. W. Stallings: „Systemy operacyjne. Architektura, funkcjonowanie i projektowanie. Wydanie IX”. Helion, Gliwice, 2018.
8. G. Coldwin: „Zrozumieć programowanie”. PWN, Warszawa, 2021.
9. A.S. Tanenbaum, H. Bos: „Systemy operacyjne. Wydanie IV”. Helion, Gliwice, 2015.

Efekty uczenia się i system ich oceniania

Podstawę do zaliczenia przedmiotu (uzyskanie punktów ECTS) stanowi stwierdzenie, że każdy z założonych **efektów uczenia się** został osiągnięty w co najmniej minimalnym akceptowalnym stopniu.

EU1	identyfikuje i opisuje zasadę działania podstawowych elementów systemu komputerowego oraz charakteryzuje podstawowe zadania systemu operacyjnego
------------	--

EU2	formułuje algorytmy komputerowe rozwiązujące typowe zadania inżynierskie występujące w elektrotechnice
------------	--

Zaliczenie wykładu

- Sprawdzian pisemny na terenie Uczelni na ostatnim wykładzie: 31.01.2022 (poniedziałek), godz. 12:15-13:00, WE-Aula II
- Na zaliczeniu oceniane będą dwa efekty uczenia się (EU1, EU2)
- Za każdy efekt uczenia się można otrzymać od 0 do 100 pkt.
- Na podstawie otrzymanych punktów wystawiana jest ocena:

Punkty	Ocena	Punkty	Ocena
91 - 100	5,0	61 - 70	3,5
81 - 90	4,5	51 - 60	3,0
71 - 80	4,0	0 - 50	2,0

- Każdy efekt uczenia się musi być zaliczony na ocenę pozytywną (min. 51 punktów)

Zaliczenie wykładu

- Ocena końcowa wyznaczana jest na podstawie sumy otrzymanych punktów:

Punkty	Ocena	Punkty	Ocena
182 - 200	5,0	122 - 141	3,5
162 - 181	4,5	102 - 121	3,0
142 - 161	4,0	0 - 101	2,0

Terminy zajęć

- Wykład nr 1 - 11.10.2021
- Wykład nr 2 - 25.10.2021
- Wykład nr 3 - 08.11.2021
- Wykład nr 4 - 22.11.2021
- Wykład nr 5 - 06.12.2021
- Wykład nr 6 - 20.12.2021
- Wykład nr 7 - 17.01.2022
- Wykład nr 8 - 31.01.2022 (zaliczenie, 1h, 12:15-13:00)

Plan wykładu nr 1

- Historia języka C
- Struktura programu, zapis kodu, sekwencje sterujące
- Komentarze
- Identyfikatory (nazwy), słowa kluczowe
- Typy danych
- Stałe liczbowe, deklaracje zmiennych i stałych
- Operatory, priorytet operatorów
- Wyrażenia, instrukcje
- Wyrażenia arytmetyczne

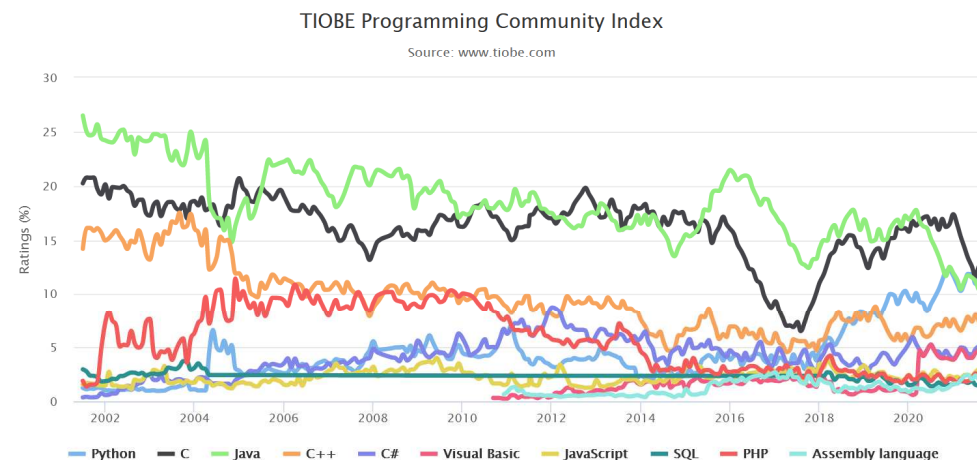
Język C - krótka historia (1/2)

- **1969** - język BCPL - Martin Richards, University Mathematical Laboratories, Cambridge
- **1970** - język B - Ken Thompson, adaptacja języka BCPL dla pierwszej instalacji systemu Unix na komputer DEC PDP-7
- **1972** - język NB (New B), nazwany później C - Dennis Ritchie, Bell Laboratories, New Jersey, system Unix na komputerze DEC PDP-11
 - 90% kodu systemu Unix oraz większość programów działających pod jego kontrolą napisane w C
- **1978** - książka „The C Programming Language” (Kernighan, Ritchie), pierwszy podręcznik, nieformalna definicja standardu (K&R)

Język C - krótka historia (2/2)

- 1989 - standard ANSI X3.159-1989 „Programming Language C” (ANSI C, C89)
- 1990 - adaptacja standardu ANSI C w postaci normy ISO/IEC 9899:1990 (C90)
- 1999 - norma ISO/IEC 9899:1999 (C99)
- 2011 - norma ISO/IEC 9899:2011 (C11)
- 2018 - norma ISO/IEC 9899:2018 (C18 lub C17)

Język C - TIOBE Programming Community Index



Język C - pierwszy program

- Niesformatowany plik tekstowy o odpowiedniej składni i mający rozszerzenie `.c`
- Kod najprostszego programu:

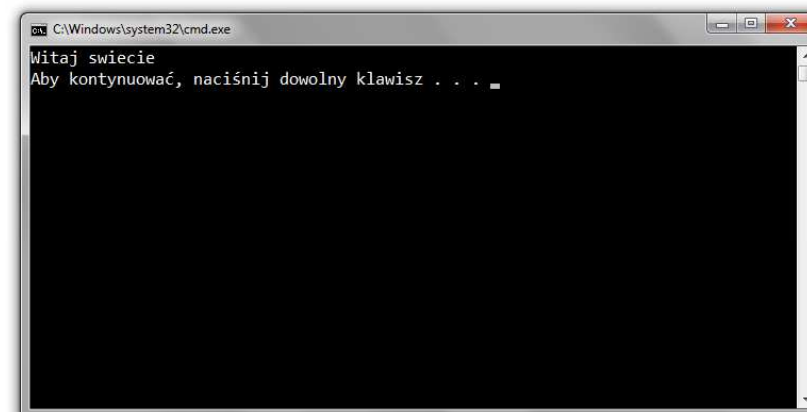
```
#include <stdio.h>

int main(void)
{
    printf("Witaj swiecie\n");
    return 0;
}
```

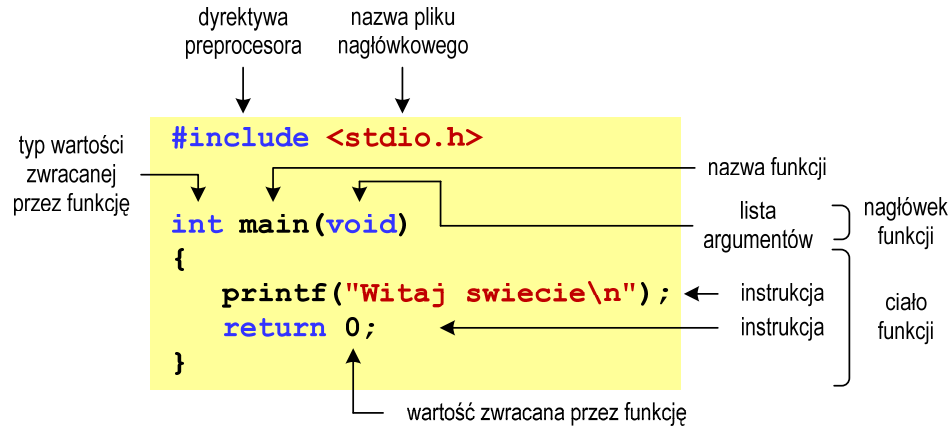
- Program konsolowy - wyświetla w konsoli tekst `Witaj swiecie`

Język C - pierwszy program

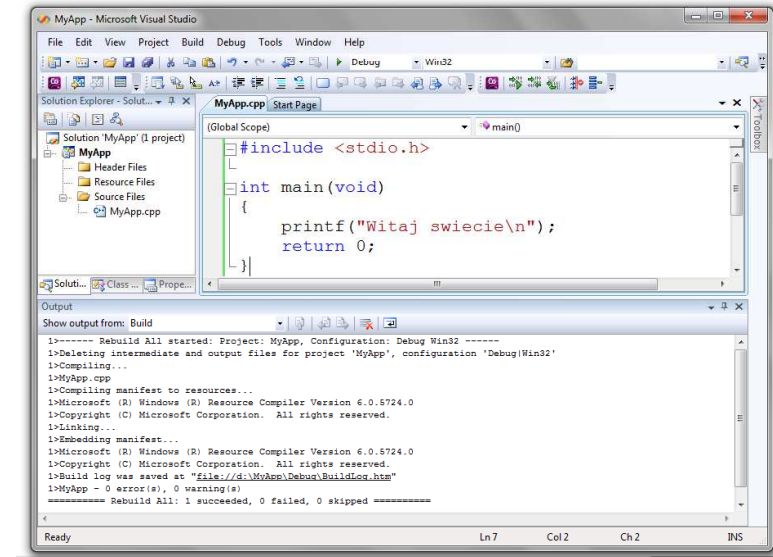
- Wynik uruchomienia programu:



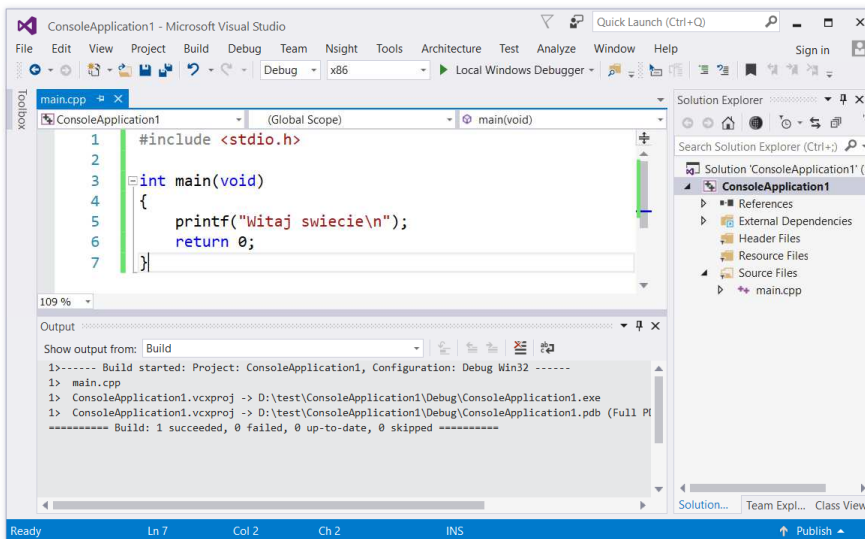
Język C - struktura programu



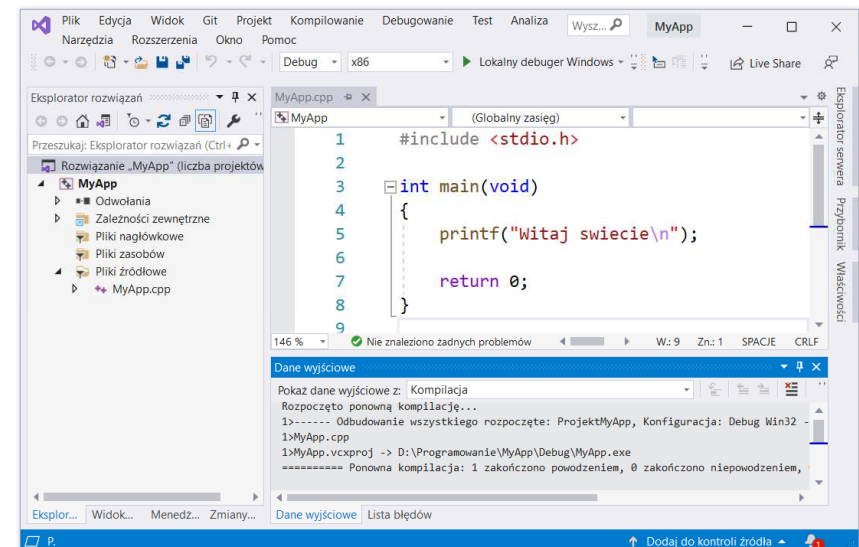
Microsoft Visual Studio 2008



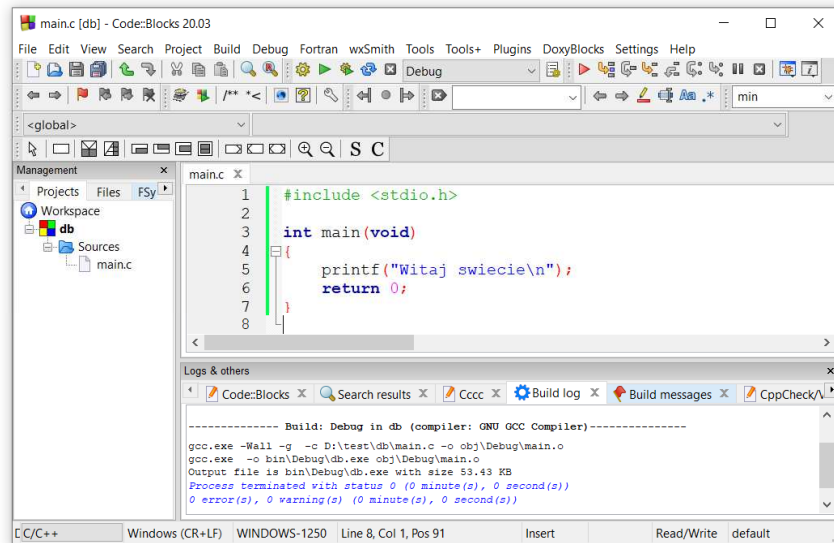
Microsoft Visual Studio 2015



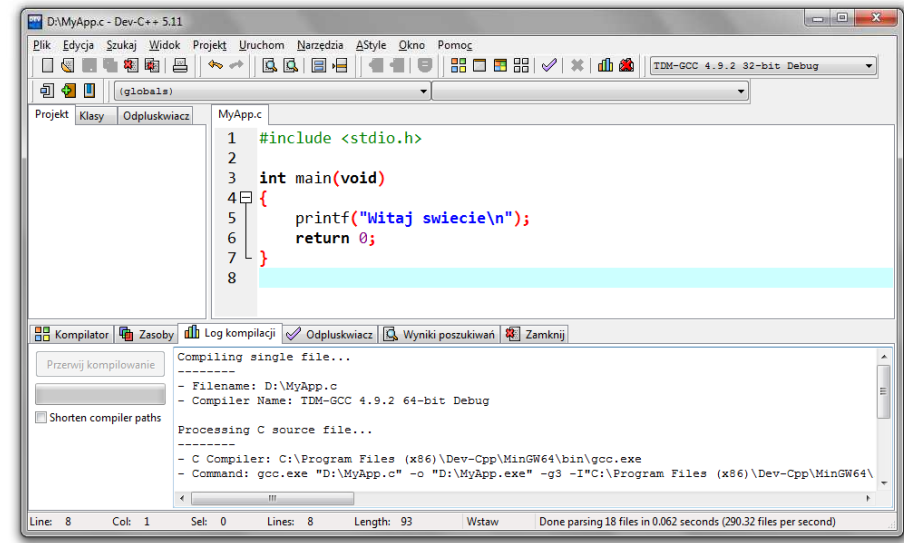
Microsoft Visual Studio 2019



Code::Blocks 20.03

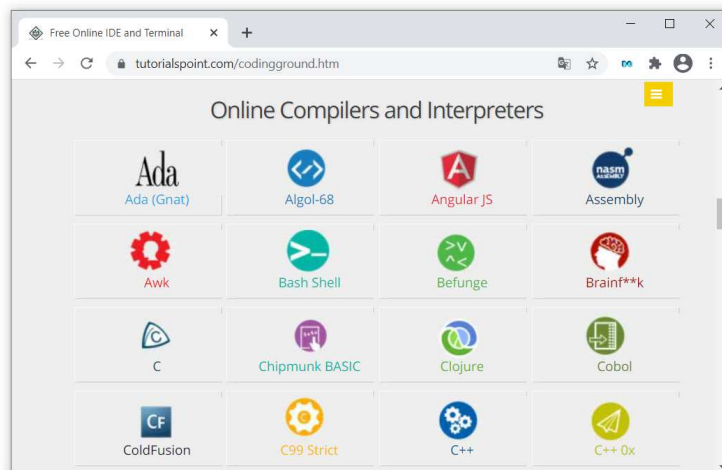


Dev-C++ 5.11



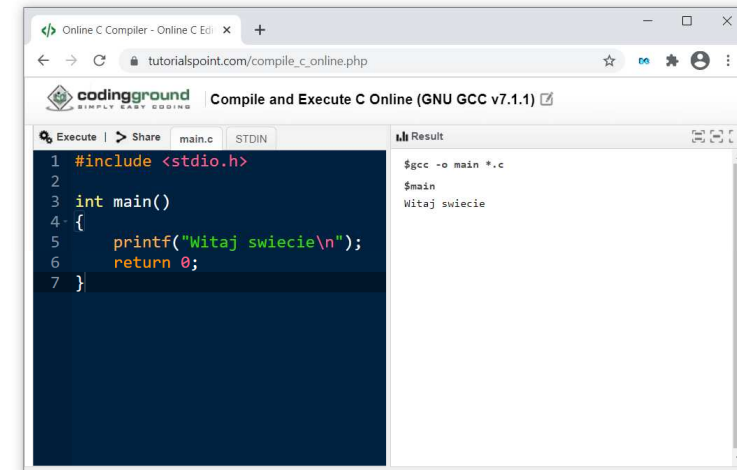
Kompilatory on-line

- <https://www.tutorialspoint.com/codingground.htm>

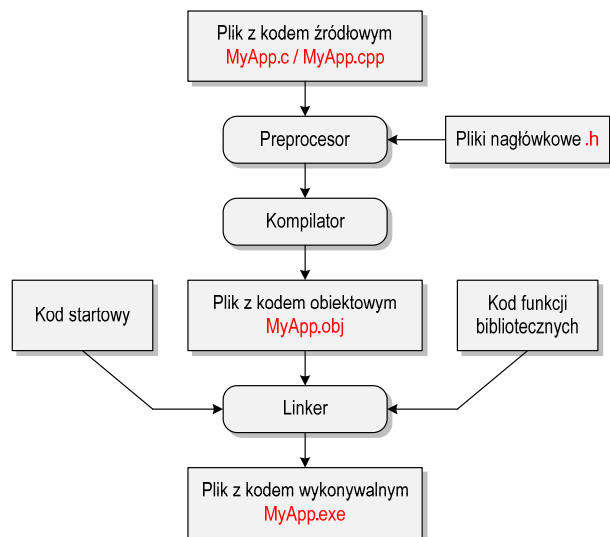


Kompilatory on-line

- <https://www.tutorialspoint.com/codingground.htm>



Język C - kompilacja programu



Język C - wyświetlanie tekstu (printf)

- Znak przejścia do nowego wiersza `\n` może pojawić w dowolnym miejscu łańcucha znaków

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

```
Witaj swiecie
-
```

```
printf("Witaj\nswiecie\n");
```

```
Witaj
swiecie
-
```

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

```
Witaj swiecie
-
```

Język C - zapis kodu programu

- Sposób zapisu kodu programu wpływa tylko na jego przejrzystość, a nie na kompilację i wykonanie
- W takiej postaci program także skompiluje się:

```
#include <stdio.h>
int main(void) {printf("Witaj swiecie\n");return 0;}
```

- W Microsoft Visual Studio 2008 można automatycznie sformatować kod źródłowy programu - **Ctrl + K + F**
- Język C rozróżnia **wielkość liter** - poniższy kod nie skompiluje się:

```
#include <stdio.h>
int Main(void) {printf("Witaj swiecie\n");return 0;}
```

Język C - sekwencje sterujące

- Istnieją także inne sekwencje sterujące (ang. escape sequence)

Opis znaku	Zapis w printf()
Alarm (ang. alert), głośniczek wydaje dźwięk	<code>\a</code>
Backspace	<code>\b</code>
Wysunięcie strony (ang. form feed)	<code>\f</code>
Przejście do nowego wiersza (ang. new line)	<code>\n</code>
CR - Carriage Return (powrót na początek wiersza)	<code>\r</code>
Tabulacja pozioma (odstęp) (ang. horizontal tab)	<code>\t</code>
Tabulacja pionowa (ang. vertical tab)	<code>\v</code>

Język C - wyświetlenie znaków specjalnych

- Niektóre znaki pełnią specjalną funkcję i nie można wyświetlić ich w tradycyjny sposób

Opis znaku	Znak	Zapis w printf()
Cudzysłów	"	\"
Apostrof	'	\'
Ukośnik (ang. backslash)	\	\\
Procent	%	%%

Ścieżka dostępu: "C:\dane\plik.txt"

```
printf("Ścieżka dostępu: \"C:\\dane\\plik.txt\\n");
```

Język C - wyświetlenie tekstu

```
#include <stdio.h>

int main(void)
{
    printf("-----\n");
    printf("| Punkty | Ocena |\n");
    printf("-----\n");
    printf("| 91-100 | 5,0 |\n");
    printf("| 81-90 | 4,5 |\n");
    printf("| 71-80 | 4,0 |\n");
    printf("| 61-70 | 3,5 |\n");
    printf("| 51-60 | 3,0 |\n");
    printf("| 0-50 | 2,0 |\n");
    printf("-----\n");

    return 0;
}
```

Punkty	Ocena
91-100	5,0
81-90	4,5
71-80	4,0
61-70	3,5
51-60	3,0
0-50	2,0

Język C - wyświetlenie znaku o podanym kodzie

- Można wyświetlić dowolny znak podając jego kod w systemie ósemkowym lub szesnastkowym

Znaczenie	Zapis
Znak o podanym kodzie ASCII (system ósemkowy)	\ooo
Znak o podanym kodzie ASCII (system szesnastkowy)	\xhh

```
printf("\127\151\164\141\152\040");
printf("\x73\x77\x69\x65\x63\x69\x65\x21\x0A");
```

Witaj świecie!

Język C - komentarze

- Komentarze są pomijane podczas kompilacji

```
/*
   Nazwa: MyApp.cpp
   Autor: Jarosław Forenc, Politechnika Białostocka
   Data: 11-10-2021 12:15
   Opis: Program wyświetlający tekst "Witaj świecie"
*/

#include <stdio.h> // zawiera deklarację printf()

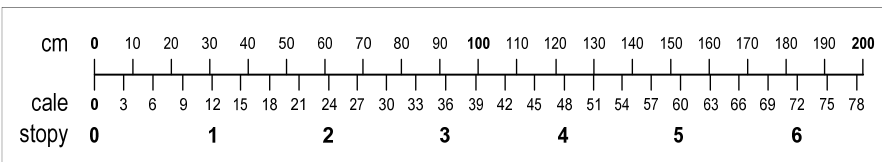
int main(void) // nagłówek funkcji main()
{
    printf/* funkcja */("Witaj świecie\n");

    return 0;
}
```


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]

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
```

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)

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

```
_temp    __temp    temperatura_w_skali_Celsiusza
```

- 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	$-3,4 \cdot 10^{38} \dots 3,4 \cdot 10^{38}$
<code>double</code>	8	$-1,7 \cdot 10^{308} \dots 1,7 \cdot 10^{308}$
<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	<code>int</code> (bajty)	<code>long double</code> (bajty)
Microsoft Visual Studio 2008	4	8
Microsoft Visual Studio 2019	4	8
Dev-C++ 5.11	4	16*
Code::Blocks 20.03	4	16*
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 `%z` który określa, że występujący po nim specyfikator (`d`, `i`, `o`, `u`, `x`, `X`) dotyczy wyświetlania wartości typu `size_t` (np. `%zd`)

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)

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

Język C - Operatory

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

```
operator operand  operand operator  -x  x++
```

- Operator dwuargumentowy

```
operand operator operand  x * y
```

- Operator trójargumentowy

```
operand operator operand operator operand  x > y ? x : y
```

- 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 - 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 - 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

- Kolejność wykonywania operacji

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

→

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

```
w = x = y = a + b;
```

→

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

- Zapis wyrażen arytmetycznych

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

```
w = a + b / c + d;
```

ŹLE

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

DOBRCZE

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

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

ŹLE

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

DOBRCZE

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)

- 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 - 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	0.693147180559945309417	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>
```

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

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

Koniec wykładu nr 1

Dziękuję za uwagę!