

Informatyka 1 (EZ1F1002)

Politechnika Białostocka - Wydział Elektryczny
Elektrotechnika, semestr II, studia niestacjonarne I stopnia
Rok akademicki 2022/2023

Wykład nr 1 (09.10.2022)

dr inż. Jarosław Forenc

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 - slajdy z wykładu
- konsultacje:
 - poniedziałek, 08:30-10:00, 17:40-18:25, WE-204
 - środa, 16:00-16:45, WE-204
 - sobota, 13:00-13:30, WE-204 (studia zaoczne)
 - niedziela, 16:00-17:00, WE-204 (studia zaoczne)

Program wykładu (1/3)

1. Programowanie w języku C. Ogólna struktura programu. Kompilacja i konsolidacja. Komentarze. Deklaracje i typy zmiennych, operatory i wyrażenia arytmetyczne, operacje wejścia-wyjścia.
2. Pozycyjne i niepozycyjne systemy liczbowe. Konwersje pomiędzy systemami liczbowymi. Jednostki informacji cyfrowej. Kodowanie znaków. Kodowanie liczb. Reprezentacja liczb w systemach komputerowych: stałoprzecinkowa i zmiennoprzecinkowa. Standard IEEE 754.
3. Programowanie w języku C. Operatory relacyjne i logiczne, wyrażenia logiczne, instrukcja warunkowa if, instrukcja switch, operator warunkowy. Pętle for, while, do .. while.
4. Programowanie w języku C. Tablice jedno-, i dwuwymiarowe. Łańcuchy znaków. Struktury w języku C, inicjalizacja zmiennej strukturalnej, odwołania do pól struktury. Pola bitowe i unie. Wskaźniki, operacje na wskaźnikach. Dynamiczny przydział pamięci w języku C.

Program wykładu (2/3)

5. Programowanie w języku C. Funkcje w języku C, ogólna struktura funkcji, deklaracja i definicja funkcji, przekazywanie argumentów do funkcji przez wartość i wskaźnik.
6. Programowanie w języku C. Operacje wejścia-wyjścia w języku C: znakowe, łańcuchowe, sformatowane, rekordowe. Pliki tekstowe i binarne.
7. Sprawdzian nr 1. Algorytmy. Definicja algorytmu. Klasyfikacje i sposoby przedstawiania algorytmów. Rekurencja. Złożoność obliczeniowa. Sortowanie.
8. Architektura komputerów. Klasyfikacja systemów komputerowych (taksonomia Flynna). Architektura von Neumana i architektura harwardzka. Budowa i zasada działania komputera. Procesor, pamięć wewnętrzna i zewnętrzna. Komunikacja z urządzeniami zewnętrznymi, interfejsy komputerowe.

Program wykładu (3/3)

9. System operacyjny. Zarządzanie procesami. Systemy plików (FAT, NTFS, ext). Zarządzanie pamięcią operacyjną.
10. Sieci komputerowe. Topologie i media transmisyjne. Model referencyjny ISO/OSI i model protokołu TCP/IP. Sprawdzian nr 2.

Literatura (1/2)

1. S. Prata: „Język C. Szkoła programowania. Wydanie VI”. Helion, Gliwice, 2016.
2. Kernighan B.W., Ritchie D.M.: „Język ANSI C. Programowanie. Wydanie II”. Helion, Gliwice, 2010.
3. Reese R.: „Wskaźniki w języku C. Przewodnik”. Helion, Gliwice, 2014.
4. Coldwin G.: „Zrozumieć programowanie”. PWN, Warszawa, 2021.
5. R. Kawa, J. Lembas: „Wykłady z informatyki. Wstęp do informatyki”. PWN, Warszawa, 2021.

Literatura (2/2)

6. P. Wróblewski: „Algorytmy, struktury danych i techniki programowania”. Wydanie VI. Helion, Gliwice, 2019.
7. A. Tanenbaum: „Strukturalna organizacja systemów komputerowych”. Helion, Gliwice, 2006.
8. K. Wojtuszkiewicz: „Urządzenia techniki komputerowej. Część 1. Jak działa komputer? Część 2. Urządzenia peryferyjne i interfejsy”. PWN, Warszawa, 2013.
9. Tanenbaum A.S., Bos H.: „Systemy operacyjne. Wydanie IV”. Helion, Gliwice, 2015.
10. Tanenbaum A.S., Wetherall D.J.: „Sieci komputerowe. Wydanie V”. Helion, Gliwice, 2012.

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.

Student zna i rozumie:

EU1	reprezentację znaków i liczb w systemach komputerowych oraz zasadę działania podstawowych elementów systemu komputerowego
EU2	podstawowe zadania systemu operacyjnego oraz strukturę sieci komputerowych
EU3	klasyfikację, sposoby przedstawiania oraz zastosowania algorytmów komputerowych rozwiązujących typowe zadania inżynierskie występujące w elektrotechnice
EU4	zastosowanie podstawowych elementów języka C w programach komputerowych

- Szczegóły: <http://jforenc.prv.pl/dydaktyka.html> lub system USOS

Zaliczenie wykładu

- Zaliczenie wykładu odbędzie się na podstawie wyników dwóch sprawdzianów pisemnych
- Na sprawdzianie nr 1 oceniane będą efekty uczenia się EU1 i EU4
- Na sprawdzianie nr 2 oceniane będą efekty uczenia się EU2 i EU3
- 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

Zaliczenie wykładu

- Każdy efekt uczenia się musi być zaliczony na ocenę pozytywną (min. 51 punktów).
- Prowadzący zajęcia może przyznawać dodatkowe punkty za aktywność na wykładzie
- Ocena końcowa wyznaczana jest na podstawie sumy otrzymanych punktów:

Punkty	Ocena	Punkty	Ocena
364 - 400	5,0	244 - 283	3,5
324 - 363	4,5	204 - 243	3,0
284 - 323	4,0	0 - 203	2,0

Plan wykładu nr 1

- Język C
 - historia, struktura programu
 - kompilacja, zapis kodu
 - sekwencje sterujące, komentarze
 - identyfikatory (nazwy), słowa kluczowe
 - typy danych, stałe liczbowe, deklaracje zmiennych i stałych
 - operatory, priorytet operatorów

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**)

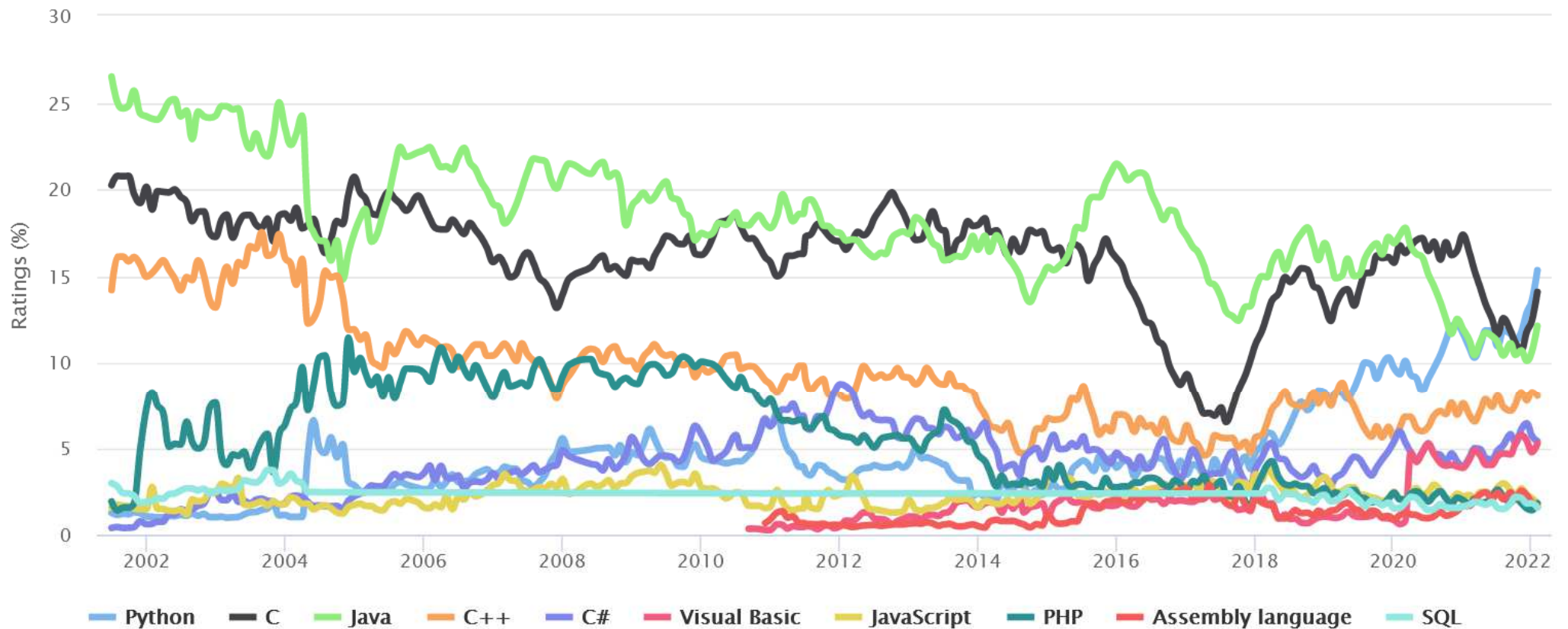
Język C - krótka historia (2/2)

- **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)

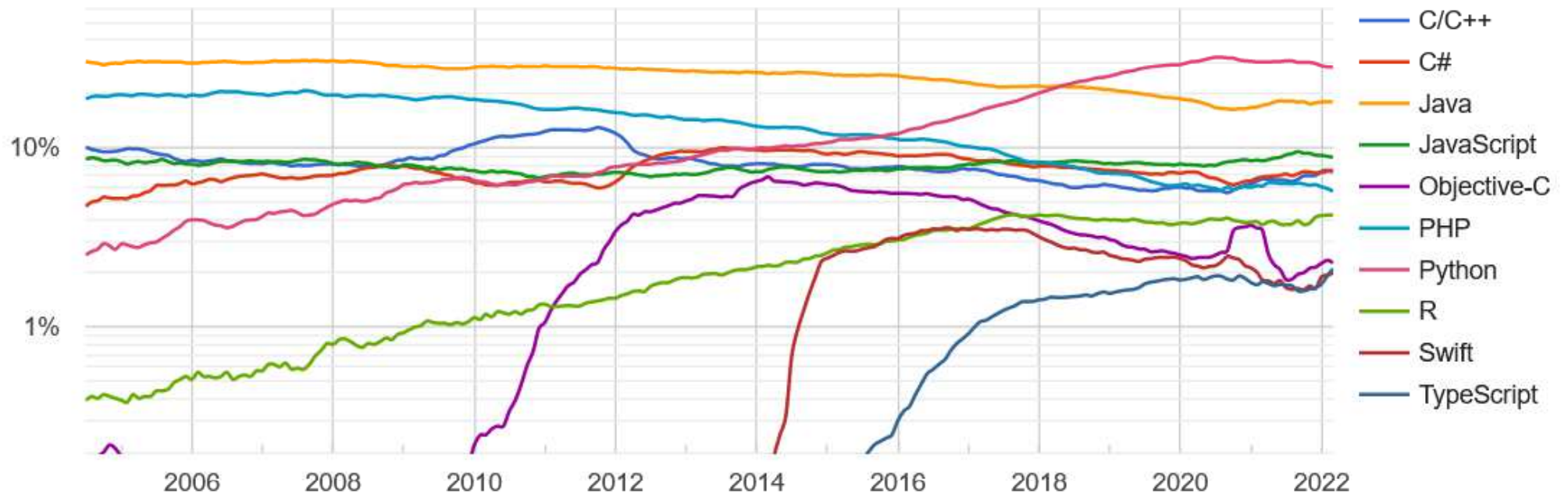
Język C - TIOBE Programming Community Index

TIOBE Programming Community Index

Source: www.tiobe.com



Język C - PYPL Popularity of Programming Language



1	Python
2	Java
3	JavaScript
4	C#
5	C/C++

6	PHP
7	R
8	Objective-C
9	TypeScript
10	Swift

The PYPL Index is created by analyzing how often language tutorials are searched on Google: the more a language tutorial is searched, the more popular the language is assumed to be.

<https://pypl.github.io/PYPL.html>

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 świecie\n");
    return 0;
}
```

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

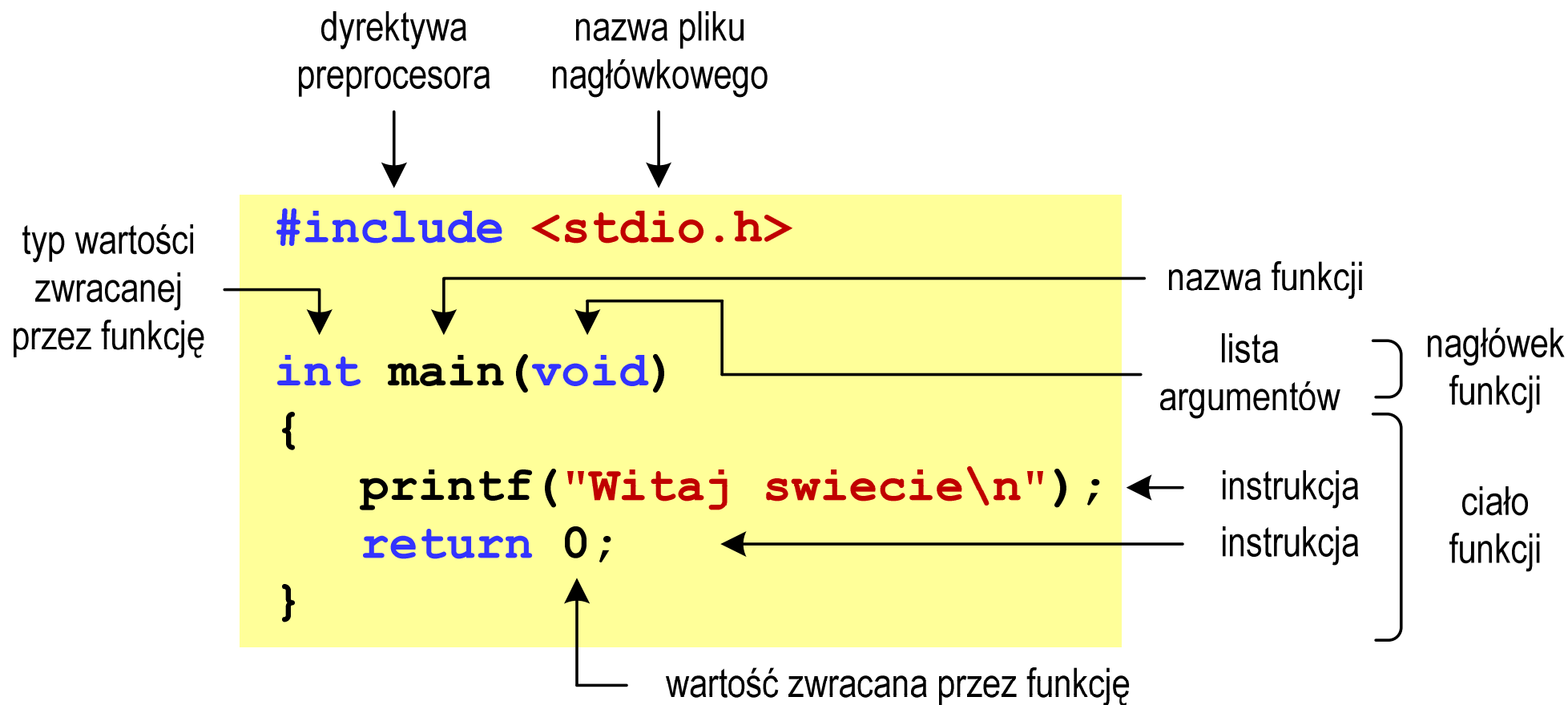
Język C - pierwszy program

- Wynik uruchomienia programu:

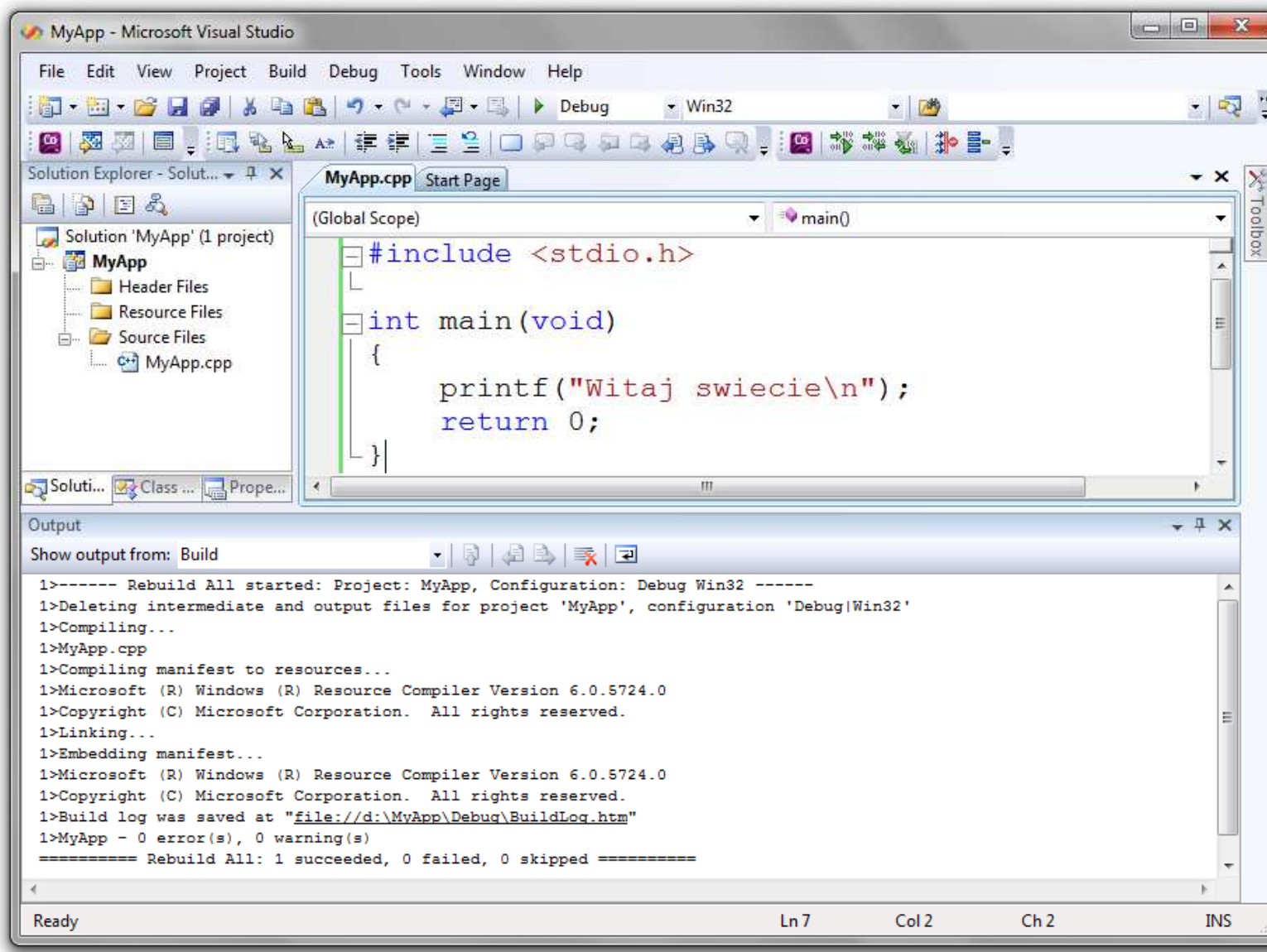


```
C:\Windows\system32\cmd.exe
Witaj swiecie
Aby kontynuować, naciśnij dowolny klawisz . . .
```

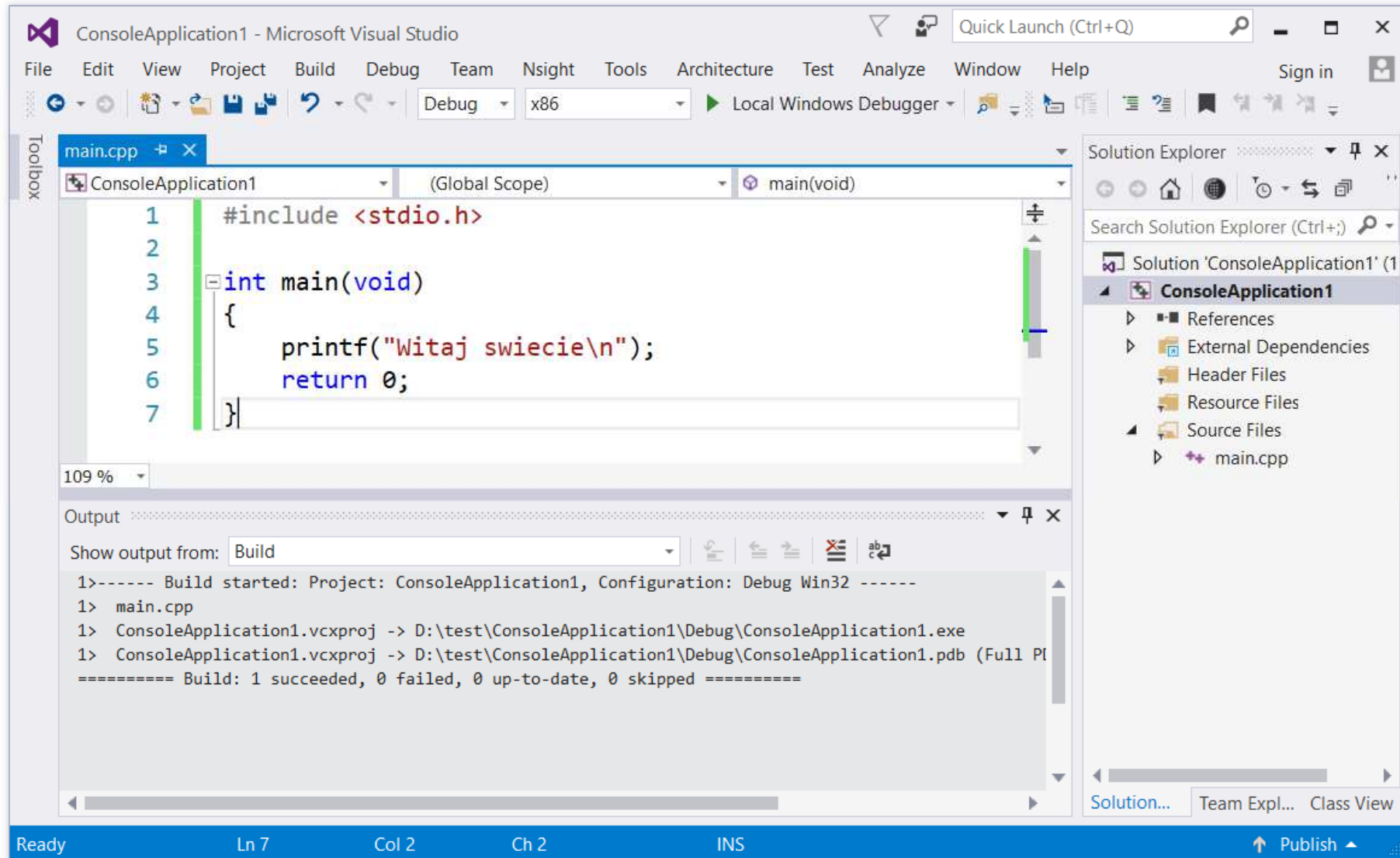
Język C - struktura programu



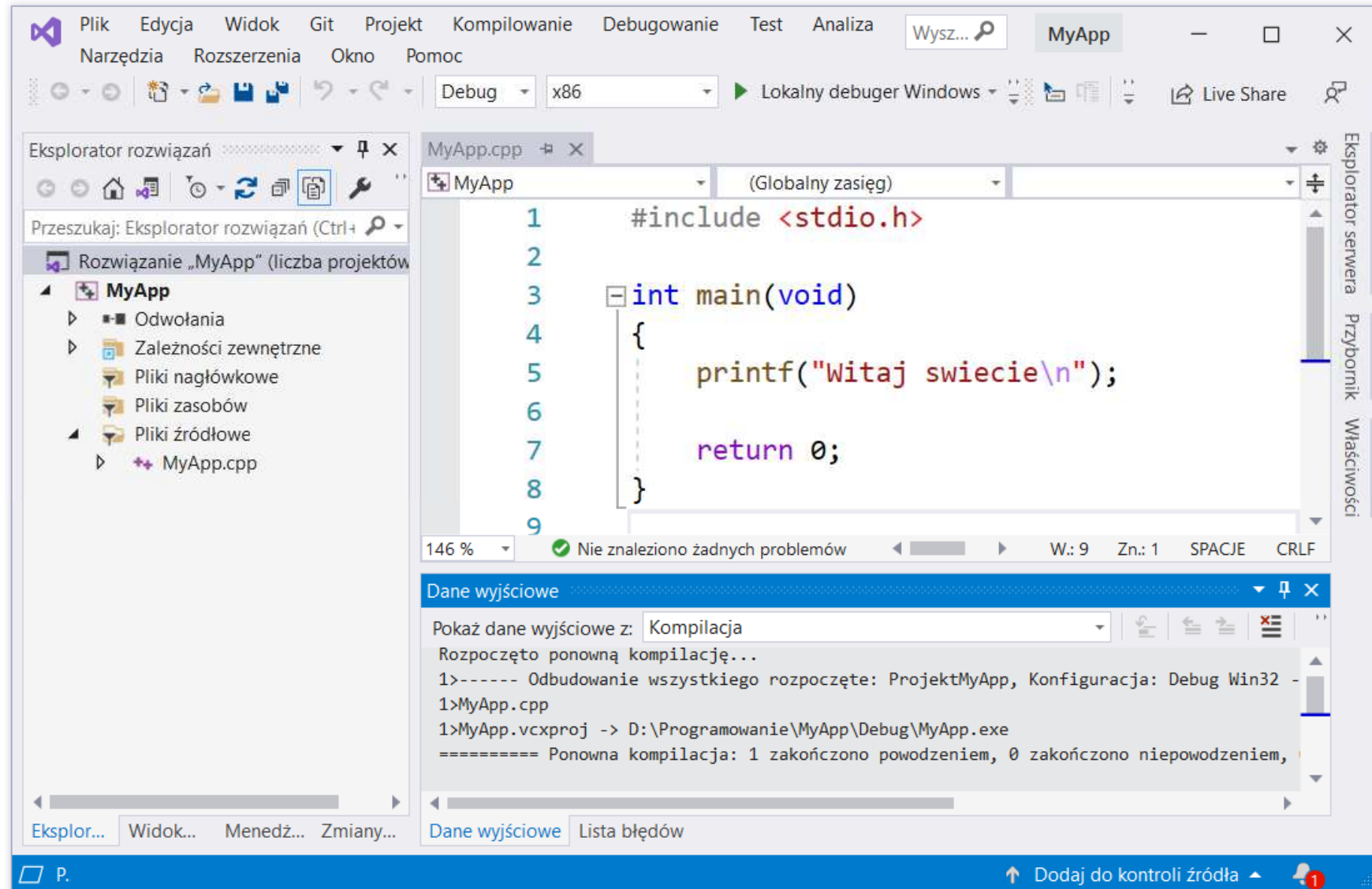
Microsoft Visual Studio 2008



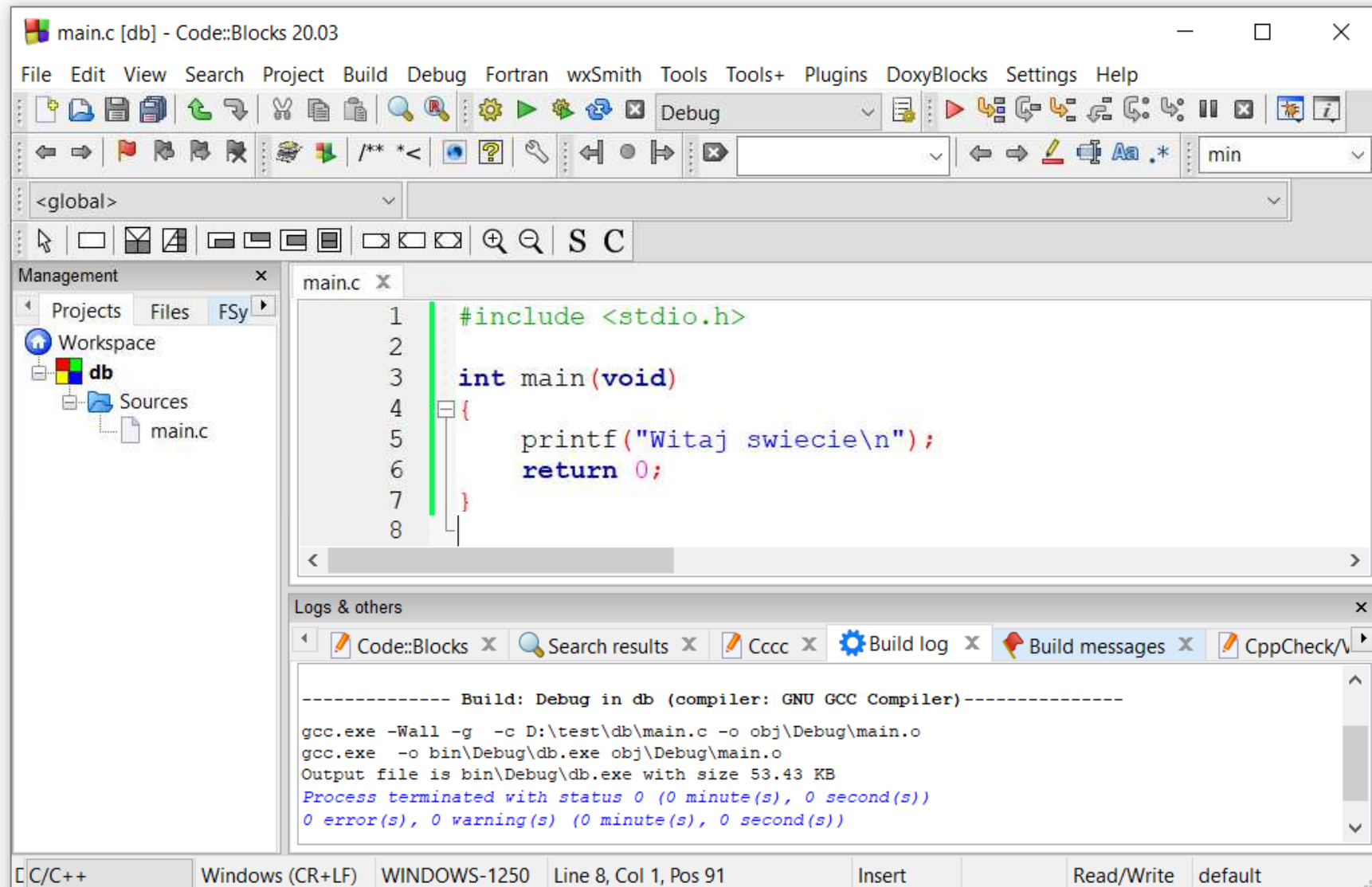
Microsoft Visual Studio 2015



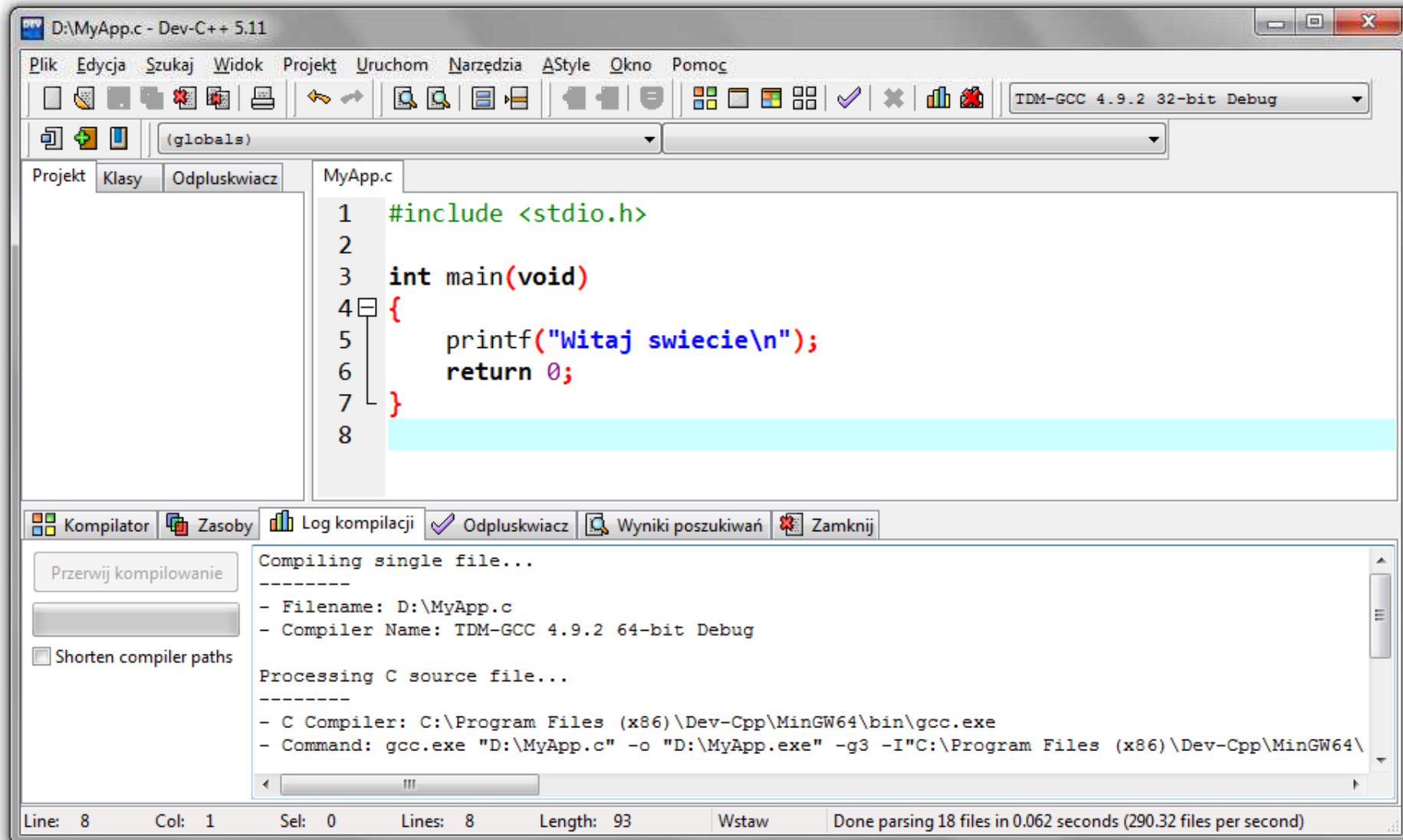
Microsoft Visual Studio 2019



Code::Blocks 20.03



Dev-C++ 5.11



Plik Edycja Szukaj Widok Projekt Uruchom Narzędzia AStyle Okno Pomoc

TDM-GCC 4.9.2 32-bit Debug

Projekt Klasy Odpluskwiacz

MyApp.c

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("Witaj swiecie\n");
6     return 0;
7 }
8
```

Kompilator Zasoby Log kompilacji Odpluskwiacz Wyniki poszukiwań Zamknij

Przerwij kompilowanie
 Shorten compiler paths

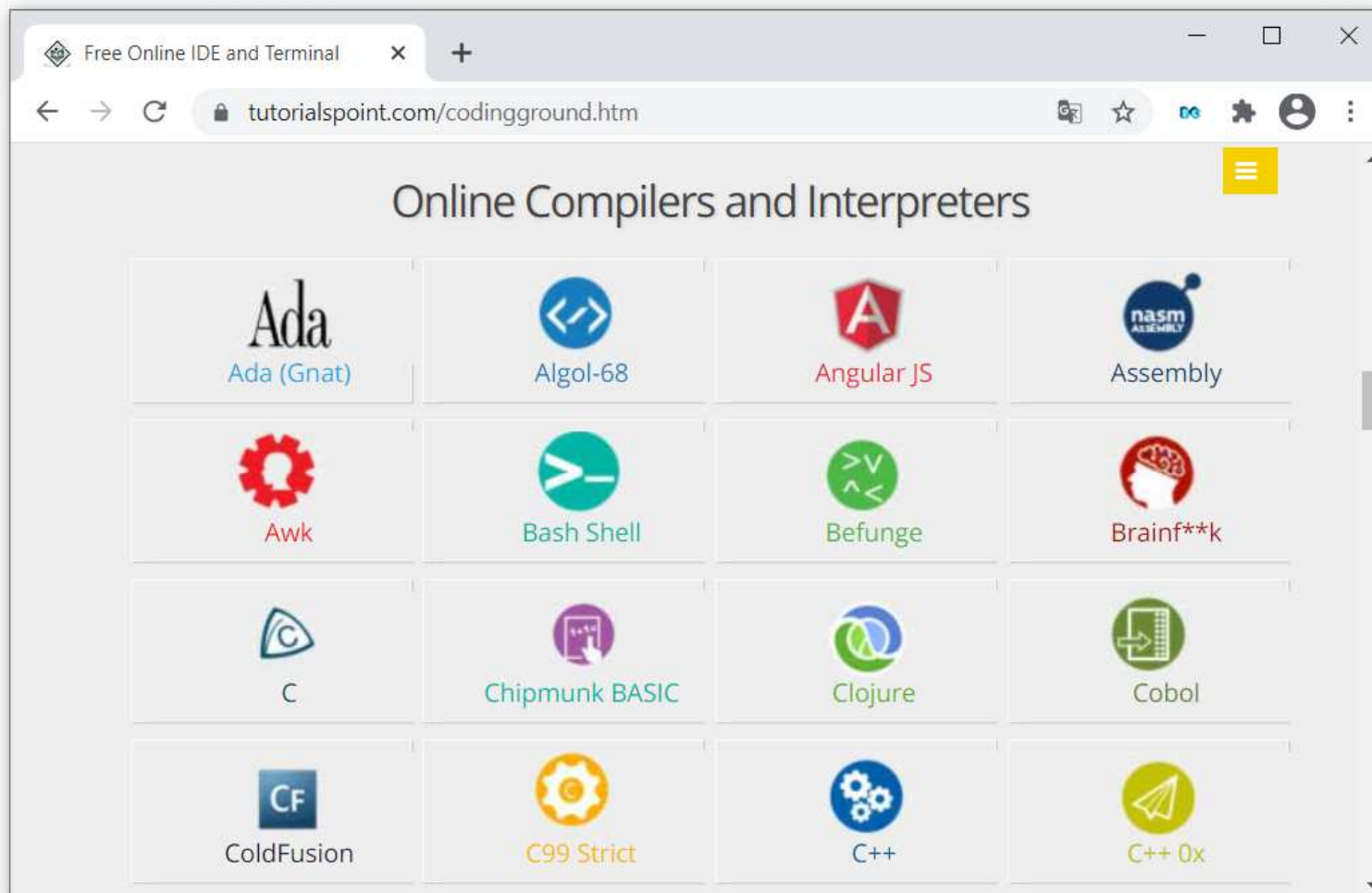
```
Compiling single file...
-----
- Filename: D:\MyApp.c
- Compiler Name: TDM-GCC 4.9.2 64-bit Debug

Processing C source file...
-----
- C Compiler: C:\Program Files (x86)\Dev-Cpp\MinGW64\bin\gcc.exe
- Command: gcc.exe "D:\MyApp.c" -o "D:\MyApp.exe" -g3 -I"C:\Program Files (x86)\Dev-Cpp\MinGW64\
```

Line: 8 Col: 1 Sel: 0 Lines: 8 Length: 93 Wstaw Done parsing 18 files in 0.062 seconds (290.32 files per second)

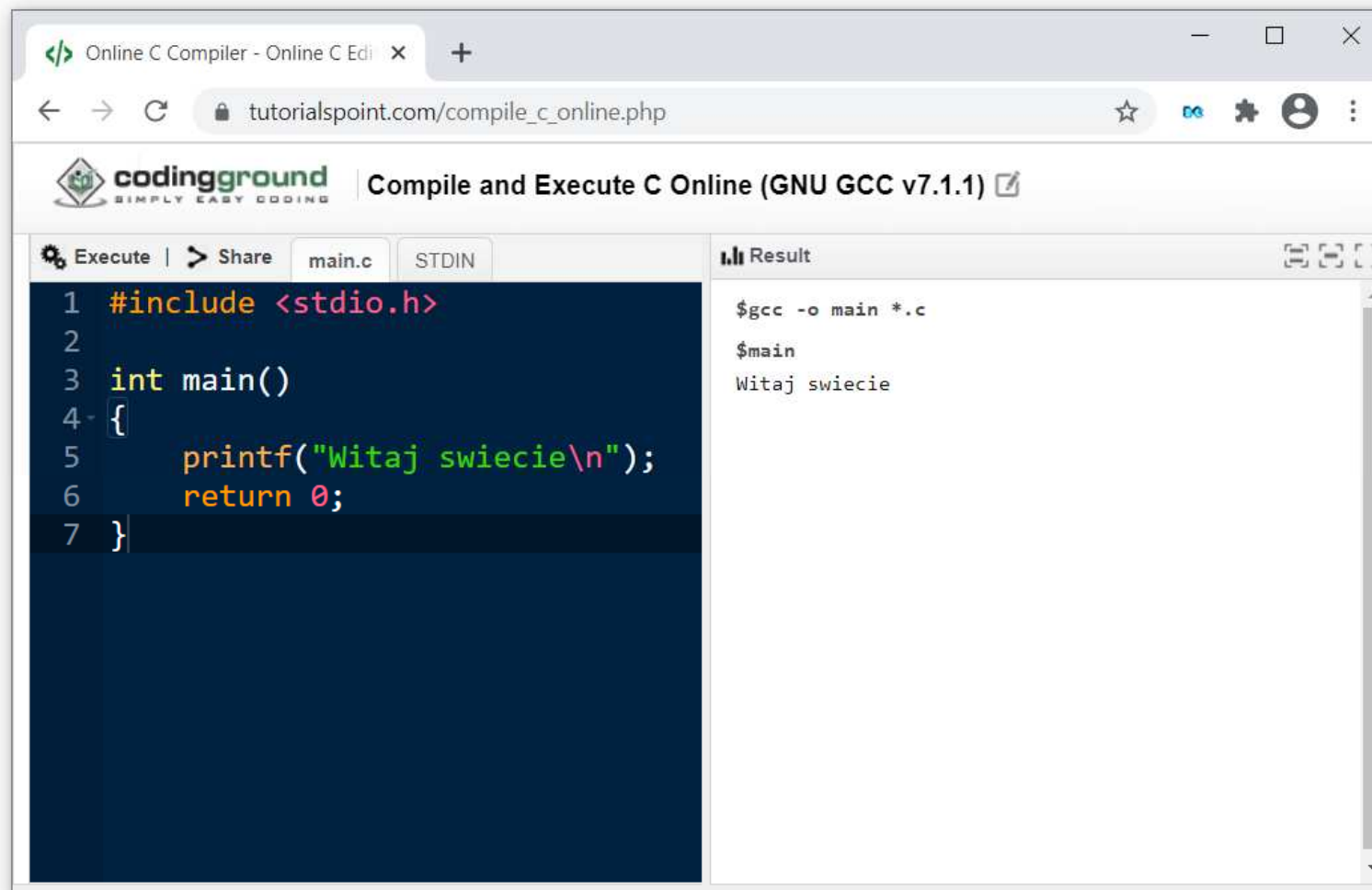
Kompilatory on-line

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



Kompilatory on-line

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



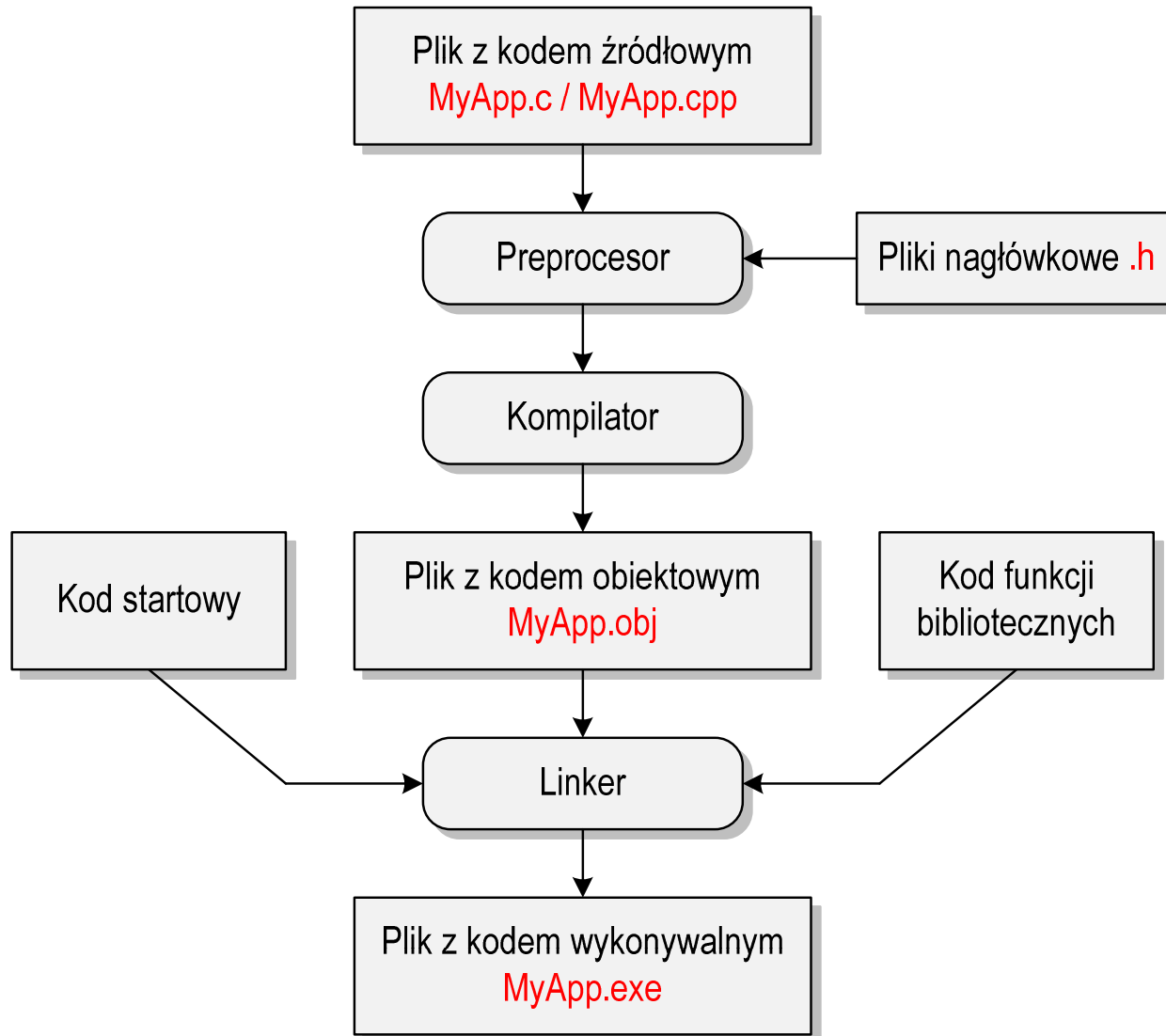
The screenshot shows a web browser window with the URL `tutorialspoint.com/compile_c_online.php`. The page title is "Compile and Execute C Online (GNU GCC v7.1.1)". The interface includes a code editor on the left and a result pane on the right. The code in the editor is:

```
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("Witaj swiecie\n");
6     return 0;
7 }
```

The result pane shows the compilation and execution output:

```
$gcc -o main *.c
$main
Witaj swiecie
```

Język C - kompilacja programu



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 - 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 - 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>
Przejdźcie 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	%	%%

```
Sciezka dostepu: "C:\dane\plik.txt"
```

```
printf("Sciezka dostepu: \"C:\\dane\\plik.txt\\\"\n");
```

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)	<code>\ooo</code>
Znak o podanym kodzie ASCII (system szesnastkowy)	<code>\xhh</code>

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

```
Witaj swiecie!
```

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 - Komentarze

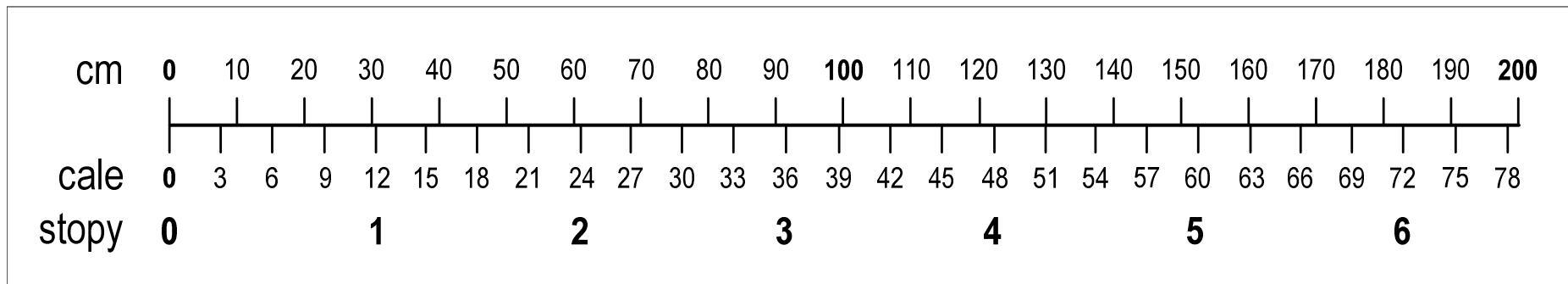
- Komentarze są pomijane podczas kompilacji

```
/*  
  Nazwa: MyApp.c  
  Autor: Jarosław Forenc, Politechnika Białostocka  
  Data: 03-10-2022 12:15  
  Opis: Program wyświetlający tekst "Witaj świecie"  
*/  
  
#include <stdio.h>      // zawiera deklarację printf()  
  
int main(void)          // nagłówek funkcji main()  

```

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]

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	$-3,4 \cdot 10^{38} \dots 3,4 \cdot 10^{38}$
<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	int (bajty)	long double (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**, **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
-----------	-----------	-----------	-----------

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

Przykład: pole i obwód koła

```
#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

```
/**  
...  
#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;  
}
```

Zaczynamy!!!
Pole = 7.065
Obwod = 9.42

zawartość pliku stdio.h

Język C - Operatory

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



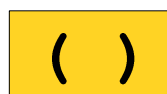
- Operator dwuargumentowy



- 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)

Koniec wykładu nr 1

Dziękuję za uwagę!