Wydział Elektryczny
Katedra Elektrotechniki Teoretycznej i Metrologii

Materiały do wykładu z przedmiotu:
**Informatyka**
**Kod: EDS1A1 007**

# WYKŁAD NR 1

**Opracował: dr inż. Jarosław Forenc**

**Białystok 2018**

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 Teoretycznej i Metrologii
  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 prezentowane na wykładzie
- Konsultacje
  - wtorek, godz. 10:00-13:30, WE-204
  - środa, godz. 09:15-10:00, WE-204
  - piątek, godz. 10:00-12:00, WE-204

# 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. S. Prata: Język C. Szkoła programowania. Wydanie VI. Helion, 2016.
2. B.W. Kernighan, D.M. Ritchie: Język ANSI C. Programowanie. Wydanie II. Helion, Gliwice, 2010.
3. R. Reese: Wskaźniki w języku C. Przewodnik. Helion, Gliwice, 2014.
4. R. Kawa, J. Lembas: Wykłady z informatyki. Wstęp do informatyki. PWN, Warszawa 2017.
5. P. Wróblewski: Algorytmy, struktury danych i techniki programowania. Wydanie V. Helion, Gliwice, 2015.
6. A.S. Tanenbaum: Strukturalna organizacja systemów komputerowych. Helion, Gliwice, 2006.
7. G. Coldwin: Zrozumieć programowanie. PWN, Warszawa, 2015.
8. A.S. Tanenbaum: Systemy operacyjne. Wydanie III. Helion, 2010.

# Efekty kształcenia i system ich oceniania

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

| | |
|---|---|
| **EK1** | identyfikuje i opisuje zasadę działania podstawowych elementów systemu komputerowego oraz charakteryzuje podstawowe zadania systemu operacyjnego |

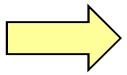
| | |
|---|---|
| **EK2** | formułuje algorytmy komputerowe rozwiązujące typowe zadania inżynierskie występujące w elektrotechnice |

# Zaliczenie wykładu - efekty kształcenia (EK1)

- Student, który zaliczył przedmiot:

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

- Student, który zalicza na ocenę dostateczny (3):
  - wymienia podstawowe elementy systemu komputerowego i podaje ich przeznaczenie
  - wyjaśnia podstawowe pojęcia związane z architekturą i zasadą działania systemów komputerowych
  - podaje definicję i wymienia podstawowe zadania systemu operacyjnego
  - opisuje wybraną metodę przydziału pamięci dyskowej



# Zaliczenie wykładu - efekty kształcenia (EK1)

- Student, który zalicza na ocenę dobry (4) (oprócz wymagań na ocenę 3):
  - opisuje strukturę i zasadę działania wybranych elementów systemu komputerowego
  - wymienia różnice pomiędzy architekturą von Neumana i architekturą harwardzką systemów komputerowych
  - podaje strukturę dysku logicznego w wybranym systemie plików (FAT, NTFS, ext)
  - wyjaśnia pojęcia stronicowania i segmentacji pamięci oraz opisuje zasadę działania pamięci wirtualnej

# Zaliczenie wykładu - efekty kształcenia (EK1)

- Student, który zalicza na ocenę bardzo dobry (5) (oprócz wymagań na ocenę 4):
  - przedstawia cel stosowania oraz zasadę działania pamięci podręcznej
  - opisuje sposób przechowywania informacji o położeniu pliku na dysku w wybranym systemie plików (FAT, NTFS, ext)

# Zaliczenie wykładu - efekty kształcenia (EK2)

- Student, który zaliczył przedmiot:

  formułuje algorytmy komputerowe rozwiązujące typowe zadania inżynierskie występujące w elektrotechnice

- Student, który zalicza na ocenę dostateczny (3):
  - przedstawia rozwiązanie prostego problemu w postaci schematu blokowego opisującego algorytm komputerowy
  - podaje definicję algorytmu komputerowego i wymienia metody opisu algorytmów
  - przedstawia sposób sortowania wektora liczb stosując wybraną, prostą metodę sortowania

# Zaliczenie wykładu - efekty kształcenia (EK2)

- Student, który zalicza na ocenę dobry (4) (oprócz wymagań na ocenę 3):
  - przedstawia rozwiązanie złożonego problemu w postaci schematu blokowego opisującego algorytm komputerowy
  - wyjaśnia pojęcie złożoności obliczeniowej algorytmu, podaje złożoności obliczeniowe przykładowych algorytmów
- Student, który zalicza na ocenę bardzo dobry (5) (oprócz wymagań na ocenę 4):
  - wyjaśnia pojęcie rekurencji i podaje przykłady algorytmów rekurencyjnych

# Zaliczenie wykładu

- Sprawdzian pisemny na koniec semestru:
  - sprawdzian: ostatni wykład w semestrze
  - poprawa: termin do ustalenia (sesja egzaminacyjna)
- Na zaliczeniu oceniane będą dwa efekt kształcenia (EK1, EK2)
- Za każdy efekt kształcenia 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

- Ocena końcowa jest średnią arytmetyczną otrzymanych ocen:

| Średnia | Ocena | Średnia | Ocena |
|---|---|---|---|
| 4,75 - 5,00 | 5,0 | 3,25 - 3,74 | 3,5 |
| 4,25 - 4,74 | 4,5 | 3,00 - 3,24 | 3,0 |
| 3,75 - 4,24 | 4,0 | 0 - 2,99 | 2,0 |

# 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

# 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 - pierwszy program

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

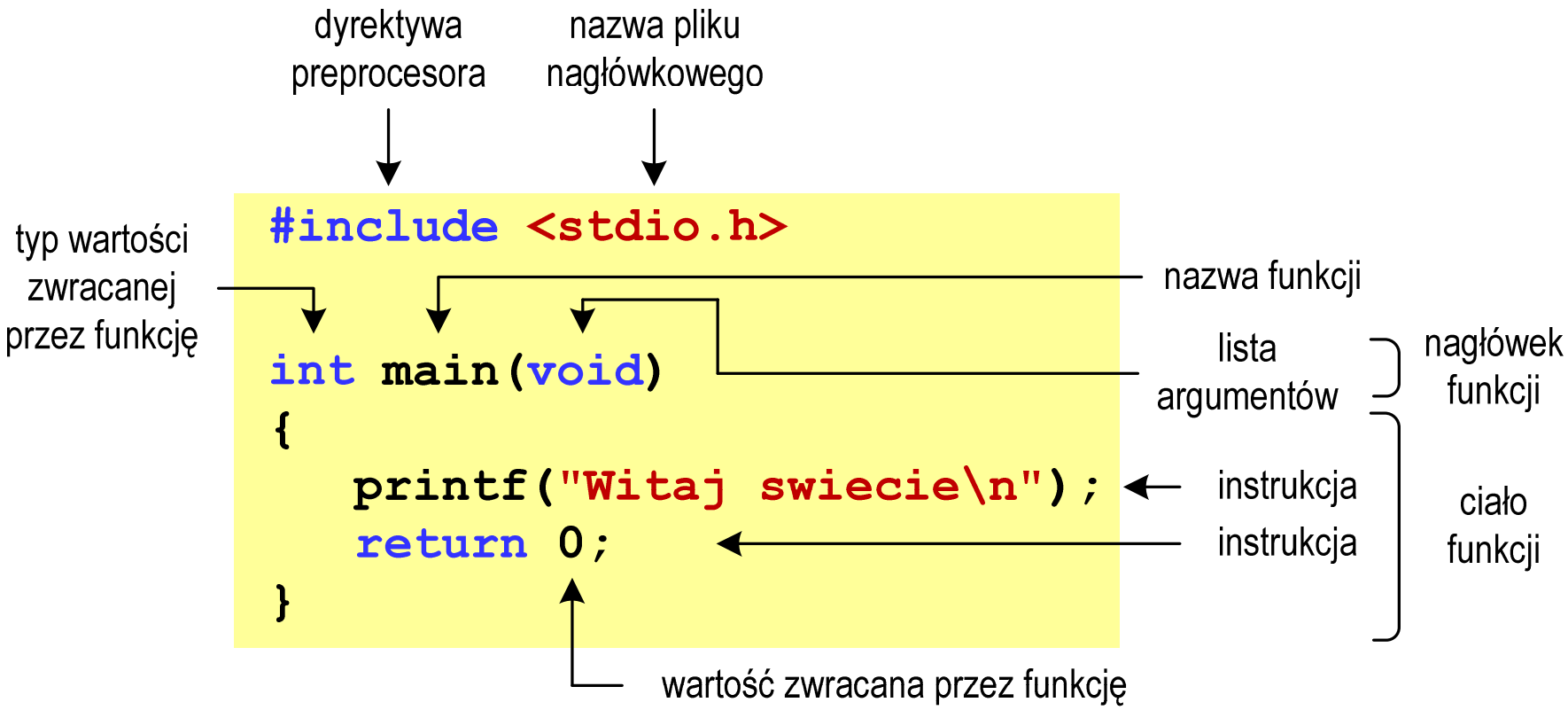
```
#include <stdio.h>

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

- Program konsolowy - wyświetla w konsoli tekst Witaj swiecie

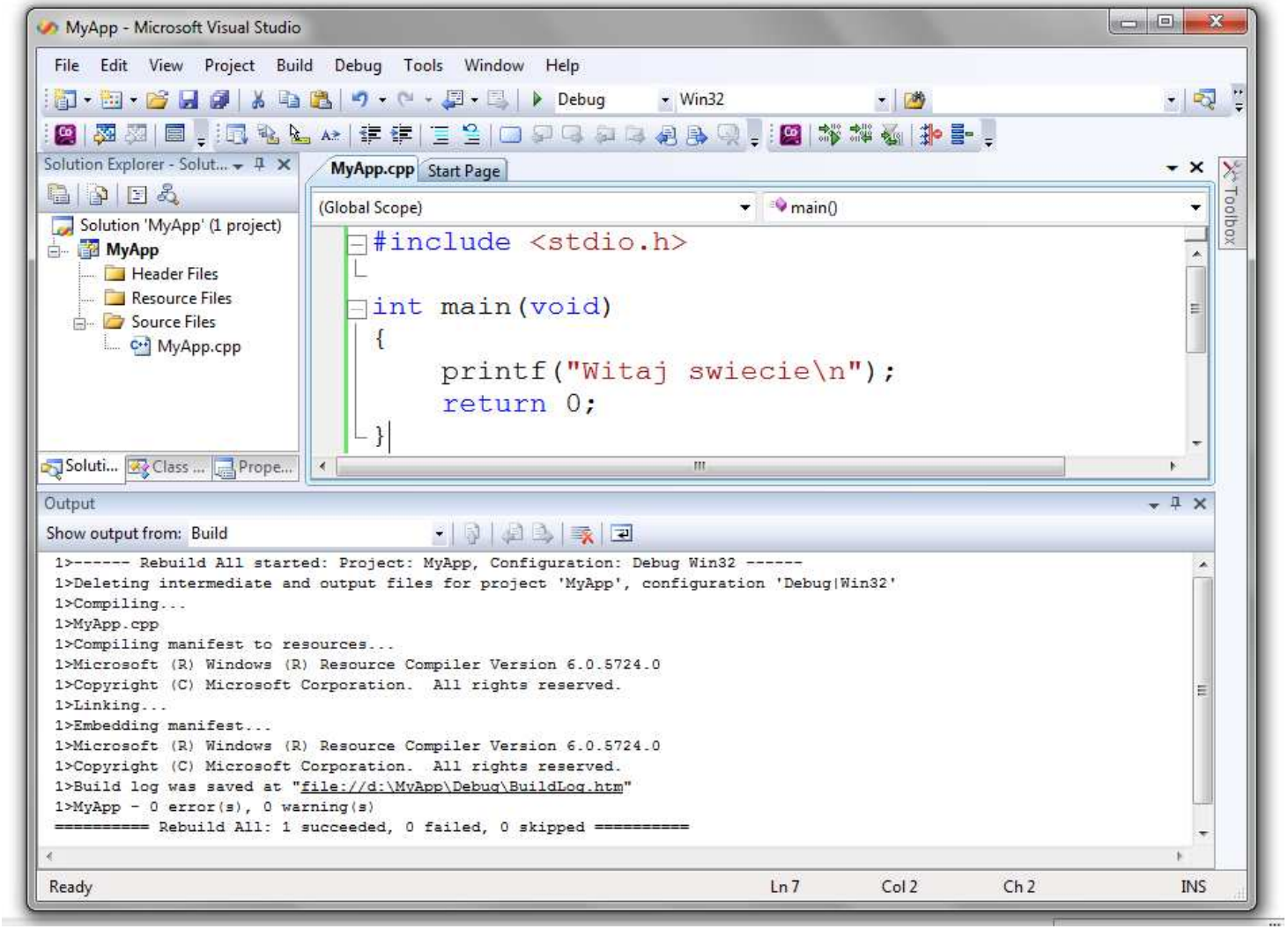
# Język C - pierwszy program

- Wynik uruchomienia programu:

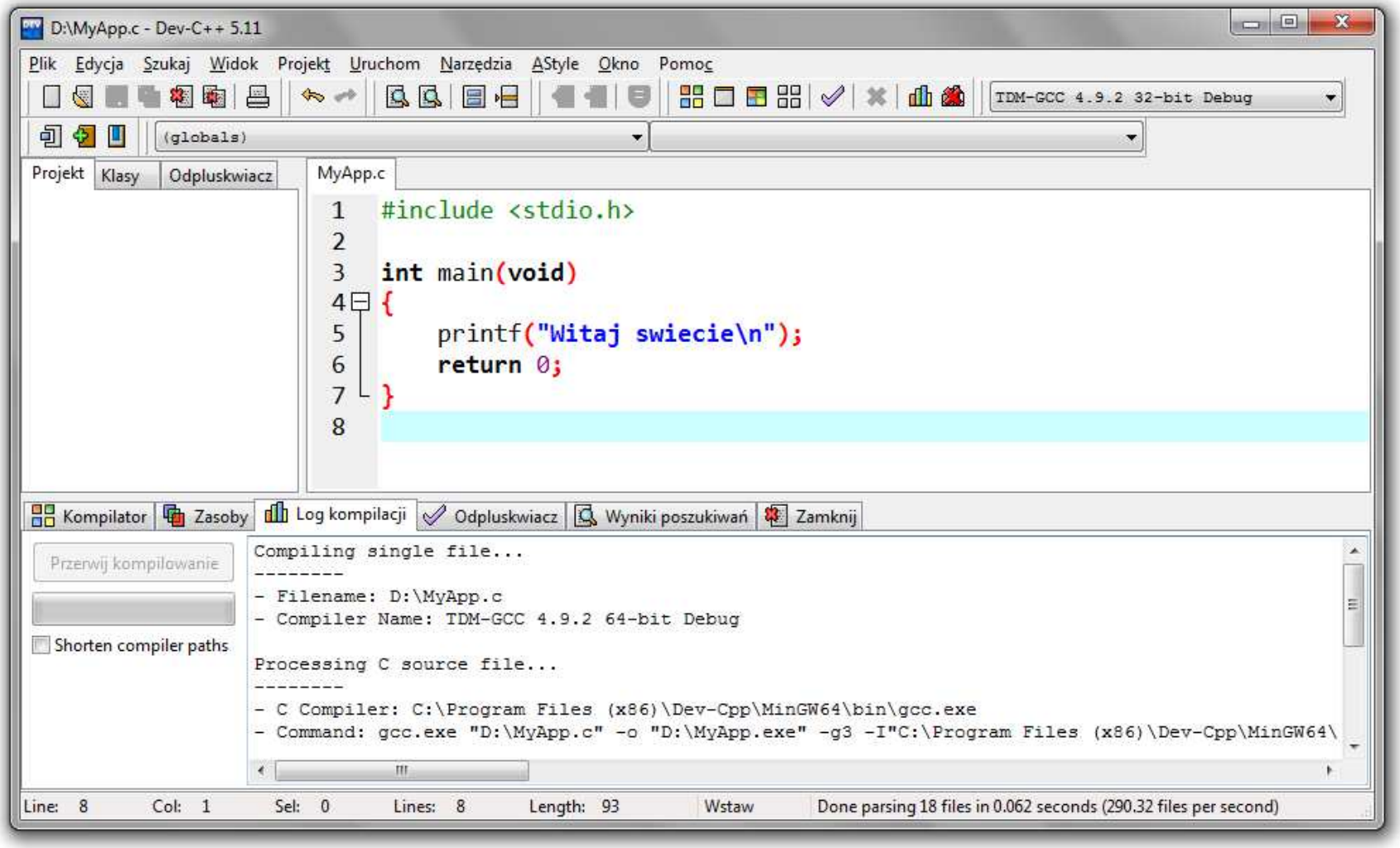
# Język C - struktura programu



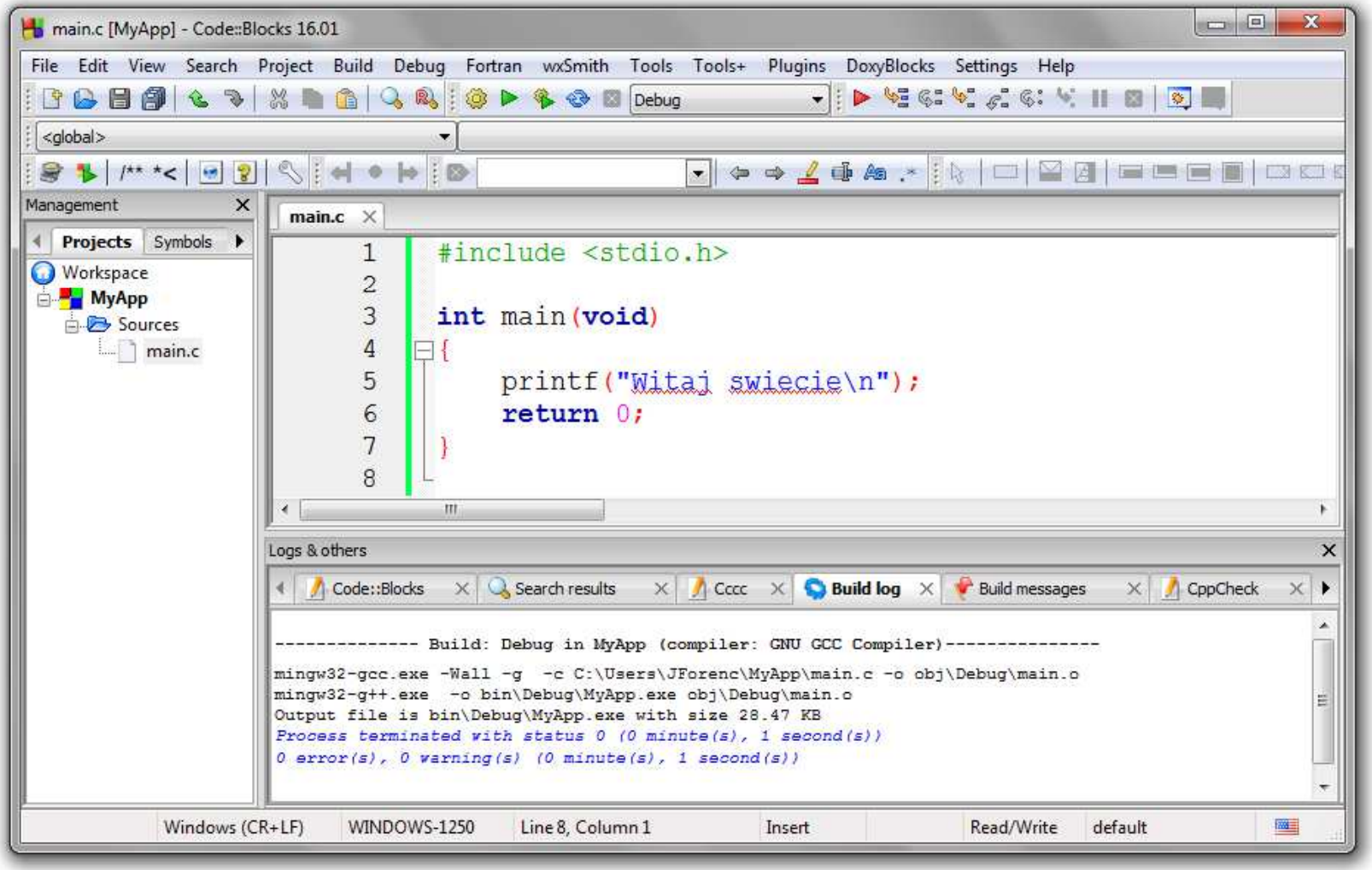
# Microsoft Visual Studio 2008



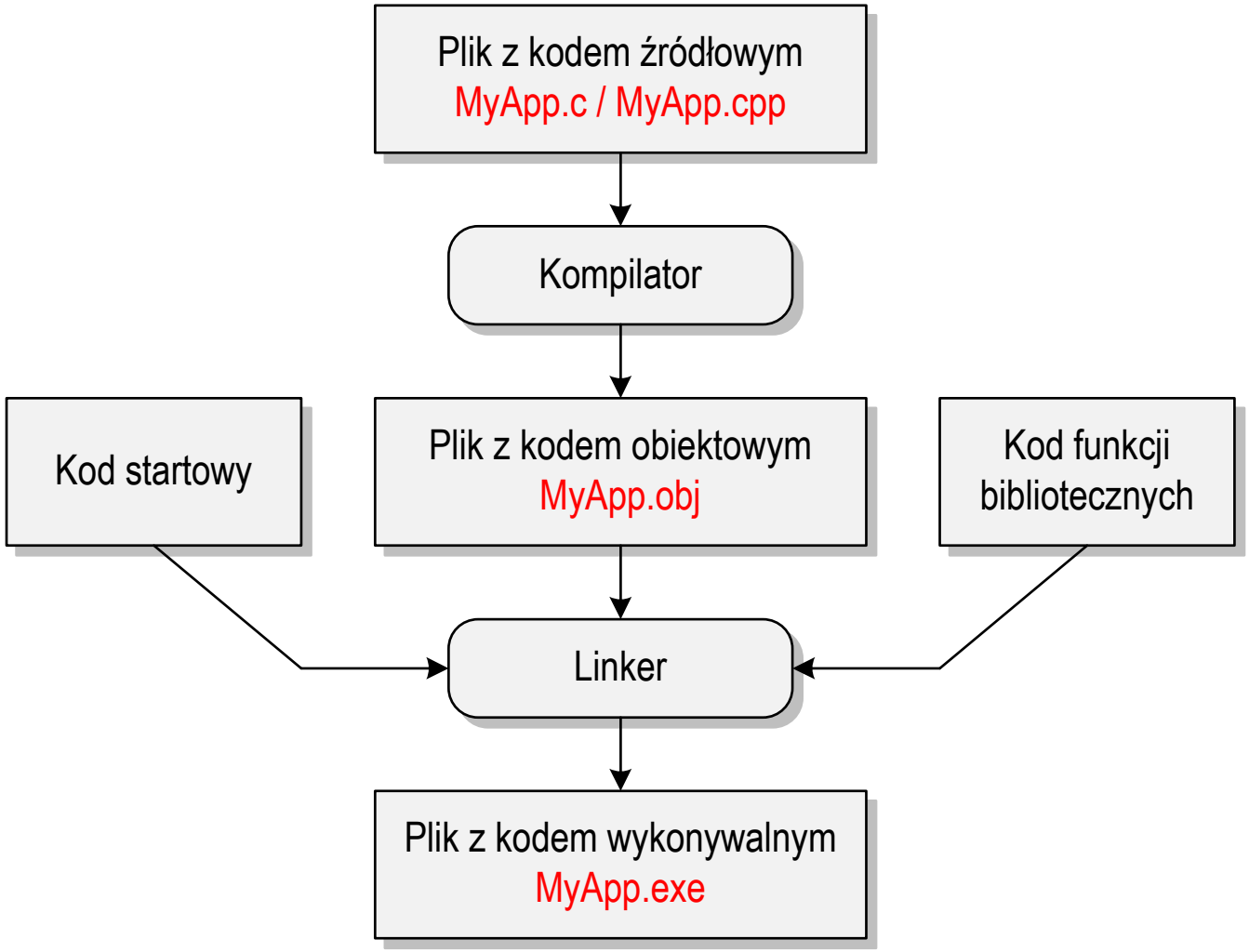
# Dev-C++ 5.11


D:\MyApp.c - Dev-C++ 5.11
Plik Edycja Szukaj Widok Projekt Uruchom Narzędzia AStyle Okno Pomoc
TDM-GCC 4.9.2 32-bit Debug
(globals)
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)

# Code::Blocks 16.01



# 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 | `\a` |
| Backspace | `\b` |
| Wysunięcie strony (ang. form feed) | `\f` |
| Przejście do nowego wiersza (ang. new line) | `\n` |
| CR - Carriage Return (powrót na początek wiersza) | `\r` |
| Tabulacja pozioma (odstęp) (ang. horizontal tab) | `\t` |
| Tabulacja pionowa (ang. vertical tab) | `\v` |

# 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) | `\0oo` |
| 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 swiecie!
```

# Język C - Wyświetlenie tekstu

```c
#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  |
------------------
```

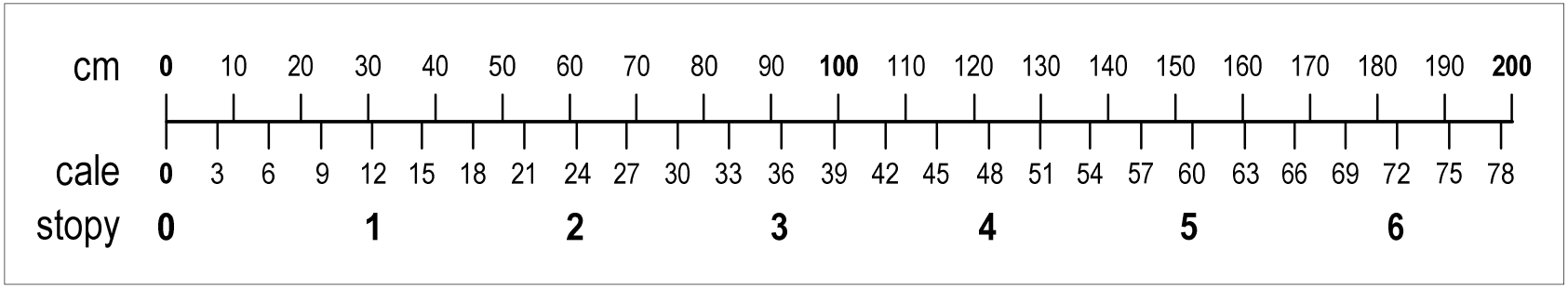
# Język C - Komentarze

- Komentarze są pomijane podczas kompilacji

```
/*
  Nazwa: MyApp.cpp
  Autor: Jarosław Forenc, Politechnika Białostocka
  Data:  19-02-2018 12:15
  Opis:  Program wyświetlający tekst "Witaj swiecie"
*/

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

int main(void)         // nagłówek funkcji main()
{
    printf/* funkcja */("Witaj swiecie\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]

# 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 (rozróżnialne 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

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

# Język C - Typy danych

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

- 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)

```c
#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
```


- 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
- Wyrażenia stałe są obliczane przed właściwą kompilacją programu
- W kodzie programu w miejscu występowania stałej wstawiana jest jej wartość

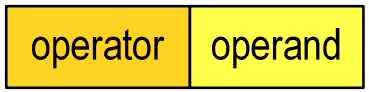
# Język C - Stałe symboliczne (#define)

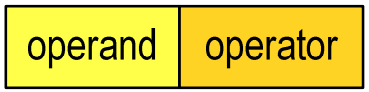
```c
#include <stdio.h>
#define PI 3.14
#define KOMUNIKAT "Zaczynamy!!!\n"

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;
}
```

```
Zaczynamy!!!
Pole = 7.065
Obwod = 9.42
```

# Język C - Operatory

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



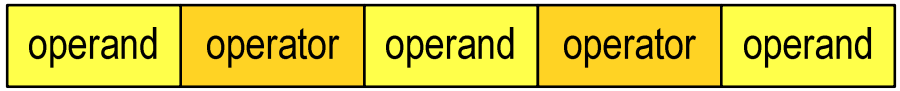

```
-x
```

```
x++
```

- Operator dwuargumentowy


```
x * y
```

- Operator trójargumentowy




```
x > y ? x : y
```

- Operator wieloargumentowy

```
( )
```

# Język C - Operatory

| Typ | Symbol |
|---|---|
| Arytmetyczne | `+ - * / %` |
| Inkrementacji / dekrementacji | `++ --` |
| Porównania (relacyjne) | `< > <= >= == !=` |
| Logiczne | `&& \|\| !` |
| Bitowe | `& \| ^ << >> ~` |
| Przypisania | `=`<br>`+= -= *= /= %=`<br>`<<= >>= &= \|= ^=` |
| Inne | `() [] & * -> .`<br>`, ? : sizeof (typ)` |

# Język C - Priorytet operatorów (1/2)

| Priorytet | Operator / opis |
|---|---|
| 1 | `++` `--` (przyrostki) `()` `[]` `.` `->` |
| 2 | `++` `--` (przedrostki) `sizeof` `(typ)`<br>`+` `-` `!` `~` `*` `&` (jednoargumentowe) |
| 3 | `*` `/` `%` |
| 4 | `+` `-` (dwuargumentowe) |
| 5 | `<<` `>>` |
| 6 | `<` `>` `<=` `>=` |
| 7 | `==` `!=` |
| 8 | `&` (bitowy) |
| 9 | `^` |

# Język C - Priorytet operatorów (2/2)

| Priorytet | Operator / opis |
|---|---|
| 10 | \| |
| 11 | && |
| 12 | \|\| |
| 13 | ? : |
| 14 | = <br> += -= *= /= %= <<= >>= &= \|= ^= |
| 15 | , (przecinek) |

# Koniec wykładu nr 1

# Dziękuję za uwagę!