

# Wstęp do programowania

INP001213Wcl

rok akademicki 2018/19

semestr zimowy

## Wykład 1

Karol Tarnowski

[karol.tarnowski@pwr.edu.pl](mailto:karol.tarnowski@pwr.edu.pl)

A-1 p. 411B



# Plan wykładów (1)

- Algorytmy i programy
- Proste typy danych
- Rozgałęzienia i iteracje
- Funkcje
- Tablice i wskaźniki
- Złożone typy danych



# Plan wykładów (2)

- Rekurencja
- Złożoność czasowa algorytmów
- Zasada dziel i zwyciężaj
- Algorytmy porządkowania
- Przeszukiwanie z nawrotami
- Poprawność i skończoność algorytmów
- Złożoność i efektywność algorytmów
- Kolokwium zaliczeniowe
- Kolokwium poprawkowe



# Plan prezentacji

- Co to jest algorytm?
- Co to jest program?
- Zapis algorytmów
- Pierwszy program w języku C

Na podstawie:

- M. M. Sysło, *Algorytmy*
- G. Perry, D. Miller, *Język C Programowanie dla początkujących*
- D. Harel, *Rzecz o istocie informatyki. Algorytmika*



# Co to jest algorytm?

- Algorytm jest przepisem opisującym krok po kroku rozwiązanie problemu lub osiągnięcie jakiegoś celu
- Algorytmika to dziedzina zajmująca się algorytmami i ich właściwościami

# Algorytm

- Słowo algorytm wywodzi się od nazwiska arabskiego matematyka i astronoma. Muhammad ibn Musa al-Chuwarizmi żył na przełomie VIII i IX wieku, jest uznawany za prekursora metod obliczeniowych w matematyce



[https://pl.wikipedia.org/wiki/Muhammad\\_ibn\\_Musa\\_al-Chuwarizmi](https://pl.wikipedia.org/wiki/Muhammad_ibn_Musa_al-Chuwarizmi)



# Przykład algorytmu

## KAKAOWE CIASTO BEZ PIECZENIA

POPULARNE CIASTA

SZYBKIE CIASTO

DESERY BEZ PIECZENIA

CIASTA

CIASTA CZEKOLADOWE

CIASTA KOSTKI

CZEKOLADA

Rewelacyjne ciasto bez pieczenia! Kakaowe hebatniki przełożone kakaową masą budyniową i dżemem porzeczkowym (można połączyć pół na pół z powidłami śliwkowymi).



# Przykład algorytmu

## SKŁADNIKI

### KREM CZEKOLADOWY

1 litr mleka

3 łyżki mąki pszennej

3 łyżki mąki ziemniaczanej

1 szklanka cukru

3 łyżki kakao

2 żółtka

### ORAZ


ok. 600 g kakaowych herbatników  
"petit beurre"

200 g masła

1 słoiczek dżemu porzeczkowego  
lub powideł śliwkowych

kakao

## PRZYGOTOWANIE

DODAJ NOTATKĘ 

### KREM CZEKOLADOWY

- Odląć 1 i 1/2 szklanki mleka i dokładnie wymieszać je (np. rózgą) z mąką pszenną i ziemniaczaną, cukrem, żółtkami oraz likierami jeśli ich używamy.
- Resztę mleka zagotować (dokładnie, aż zaczną kipieć), następnie wlewać do niego mieszankę mleka, mąki i żółtek, jednocześnie energicznie mieszając rózgą. Zagotować co chwilę mieszając.
- Po zagotowaniu gotowy budyń odstawić z ognia, przelać do czystej miski i całkowicie ostudzić (na wierzch można położyć folię spożywczą aby nie zrobił się kożuch).
- Miękkie masło ubijać przez ok. 3 minuty aż się napuszy, następnie stopniowo, w krótkich odstępach czasu, dodawać budyń ciągle ubijając.



# Przykład algorytmu

## PRZEŁOŻENIE

- Formę o wymiarach ok. **20 x 30 cm** (może być większa) wysmarować masłem i wyłożyć papierem do pieczenia. Układać warstwami na przemian herbatniki i krem budyniowy, otrzymując 4 lub 5 takich warstw, druga warstwa od dołu ma mieć zamiast kremu - dżem. Na wierzchu ma być cienka warstwa kremu.
- Ciasto oprószyć kakao i wstawić do lodówki na kilka godzin lub noc. Ciasto można przygotować dzień wcześniej.

## WSKAZÓWKI

Opcjonalnie do kremu można dodać 4 łyżki likieru pomarańczowego lub amaretto lub orzechowego lub 2 łyżki mocnego alkoholu.



kwestia smaku



# Algorytm Euklidesa

- Algorytm wyznaczania największego wspólnego dzielnika dwóch dodatnich liczb całkowitych został sformułowany w IV w. p.n.e. przez Euklidesa

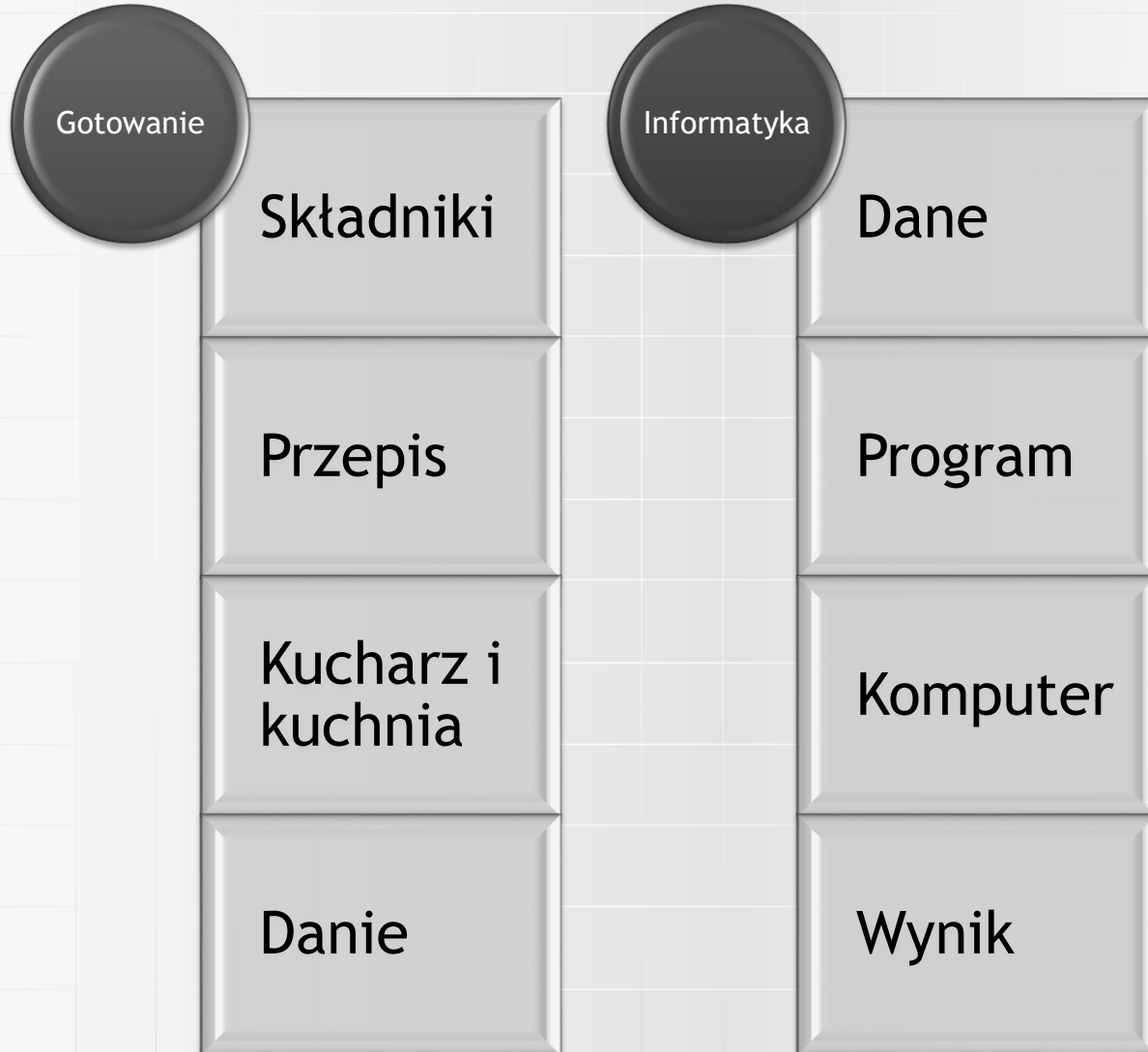


# Co to jest program?

- Program to lista szczegółowych instrukcji przekazywanych komputerowi do realizacji określonych zadań



# Program i przepis a algorytm





# Komputer

- O komputerach można myśleć, jako o zestawie przełączników - bitów.
- Każdy bit może znajdować się w jednym z dwóch stanów oznaczających 0 lub 1.
- Komputer może wykonać bezpośrednio niewielką liczbę prostych operacji (przerzucenie, wyzerowanie, sprawdzenie).



# Komputer

przerzuć  
bit

0	1	0	1	1
---	---	---	---	---

przerzuć  
bit

0	1	0	0	1
---	---	---	---	---

0	1	1	0	1
---	---	---	---	---

wyzeruj bit

0	1	0	1	1
---	---	---	---	---

wyzeruj bit

0	1	0	0	1
---	---	---	---	---

0	1	0	0	1
---	---	---	---	---



# Poziomy szczegółowości

PRZYGOTOWANIE

DODAJ NOTATKĘ



## KREM CZEKOLADOWY

- Odląć 1 i 1/2 szklanki mleka i dokładnie wymieszać je (np. różgą) z mąką pszenną i ziemniaczaną, cukrem, żółtkami oraz likierami jeśli ich używamy.
- Resztę mleka zagotować (dokładnie, aż zacznie kipieć), następnie wlewać do niego mieszankę mleka, mąki i żółtek, jednocześnie energicznie mieszając różgą. Zagotować co chwilę mieszając.
- Po zagotowaniu gotowy budyń odstawić z ognia, przelać do czystej miski i całkowicie ostudzić (na wierzch można położyć folię spożywczą aby nie zrobił się kożuch).
- Miękkie masło ubijać przez ok. 3 minuty aż się napuśzy, następnie stopniowo, w krótkich odstępach czasu, dodawać budyń ciągle ubijając.

- Przykład: mnożenie  $528 \times 46$



# Reprezentacje problemów

- Oblicz wartość funkcji  $f(x) = \frac{x}{|x|}$
- Konieczne jest sformułowanie specyfikacji: warunki jakie powinny spełniać dane wejściowe, oraz jaki jest związek wyników z danymi.
- Dane: dowolna liczba rzeczywista  $x$ .
- Wynik: wartość funkcji  $f(x)$ , jeśli  $x$  jest różne od zera i zero w przeciwnym przypadku



# Reprezentacje problemów

## Opis słowny

- Dane: dowolna liczba rzeczywista  $x$ .
- Wynik: wartość funkcji  $f(x)$ , określonej wzorem

$$f(x) = \begin{cases} -1, & \text{dla } x < 0 \\ 0, & \text{dla } x = 0 \\ 1, & \text{dla } x > 0 \end{cases}$$

# Reprezentacje problemów

## Lista kroków

- Dane: dowolna liczba rzeczywista  $x$ .
- Wynik: wartość funkcji  $f(x)$ , określonej wzorem

Krok 0. Wczytaj wartość danej  $x$ .

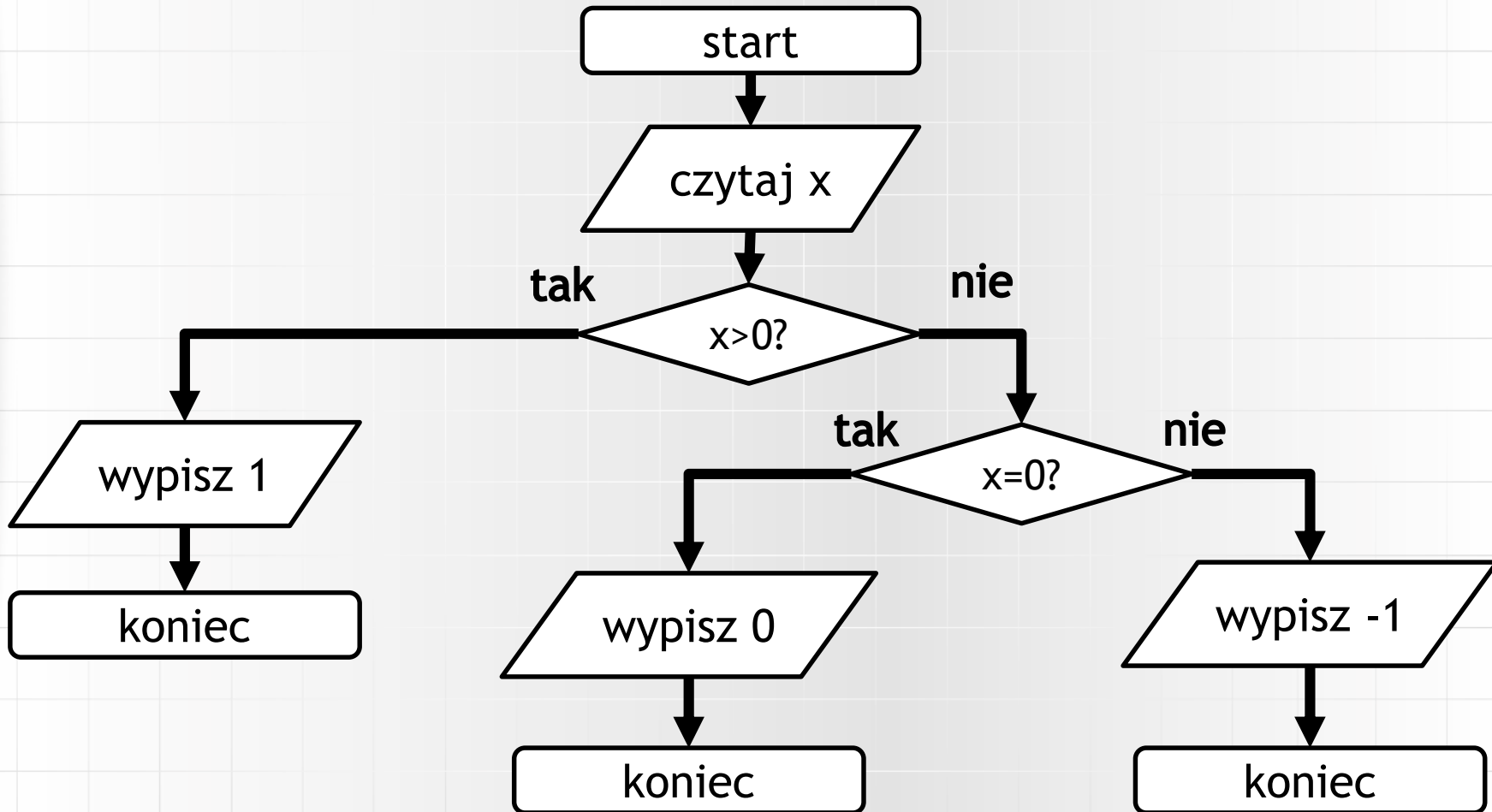
Krok 1. Jeśli  $x > 0$ , wynikiem jest 1. Zakończ algorytm.

Krok 2. *{W tym przypadku  $x \leq 0$ }* Jeśli  $x = 0$ , wynikiem jest 0. Zakończ algorytm.

Krok 3. *{W tym przypadku  $x < 0$ }* Wynikiem jest -1. Zakończ algorytm.

# Reprezentacje problemów

## Schemat blokowy





# Reprezentacje problemów

## Program w języku C

```
Start here x reprezentacja_algorytmu.c x
1 //reprezentacja_algorytmu.c
2
3 #include <stdio.h>
4
5 int main()
6 {
7     float x;
8     printf("Podaj x:"); scanf("%f",&x);
9
10    if(x>0)
11        printf("f(x) = 1");
12    else //w tym przypadku x <= 0
13        if(x==0)
14            printf("f(x) = 0");
15        else //w tym przypadku x < 0
16            printf("f(x) = -1");
17    return 0;
18 }
19
```

# Reprezentacje problemów

## Różnorodność języków programowania

- <http://www.99-bottles-of-beer.net>

### Language C

(standard version)

**Date:** 04/20/05  
**Author:** Bill Wein  
**URL:** n/a  
**Comments:** 4  
**Info:** n/a  
**Score:** ★★★ (2.81 in 124 votes)

```

/*
 * 99 bottles of beer in ansi c
 *
 * by Bill Wein: bearheart@bearnert.com
 */

#define MAXBEER (99)

void chug(int beers);

main()
{
    register beers;

    for(beers = MAXBEER; beers; chug(beers--))
        puts("");

    puts("\nTime to buy more beer!\n");

    exit(0);
}

void chug(register beers)
{
    char howmany[8], *s;

    s = beers != 1 ? "s" : "";
    printf("%d bottle%s of beer on the wall,\n", beers, s);
    printf("%d bottle%s of beeeer . . .,\n", beers, s);
    printf("Take one down, pass it around,\n");

    if(--beers) sprintf(howmany, "%d", beers); else strcpy(howmany, "No more");
    s = beers != 1 ? "s" : "";
    printf("%s bottle%s of beer on the wall.\n", howmany, s);
}

```

### Language Matlab

(vectorized version)

**Date:** 04/20/05  
**Author:** Rich Stein  
**URL:** n/a

### Language Perl

(bottled by Acme::EyeDrops)

**Date:** 06/04/05  
**Author:** Andrew Savige  
**URL:** n/a  
**Comments:** 76  
**Info:** <http://search.cpan.org/dist/Acme-EyeDrops/lib/Acme/EyeDrops.pm>  
**Score:** ★★★★★ (3.51 in 2731 votes)

```

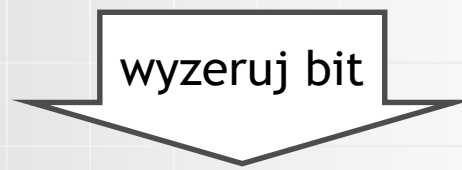
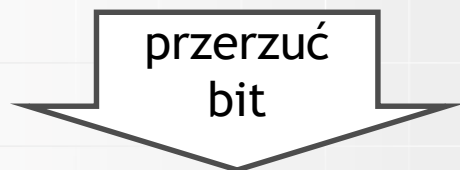
% M
% R
bot
lin
num
for
for
for
for
for
for
% b
fpr
fpr
fpr

```

# Reprezentacje problemów

## Języki programowania a język maszynowy

- Język maszynowy zestaw podstawowych instrukcji, jakie dany procesor potrafi wykonać



- Program zapisany w języku programowania jest przetwarzany na język maszynowy



# Co jest potrzebne do pisania programów w języku C?

Compiler	Author	Microsoft Windows	Unix-like	Other OSs	License type
AMPC	Axiomatic Solutions Sdn Bhd	No	Yes	Yes	Proprietary
Dev C++	Dev Inc.	Yes	No	No	Proprietary
Aztec C	Manx Software Systems	No	No	CP/M, CP/M-86, DOS, Classic Mac OS	Proprietary
Amsterdam Compiler Kit	Andrew Tanenbaum and Ceriel Jacobs	No	Yes	Yes	BSD
CCS C Compiler	CCS, Inc.	Yes	Yes	Yes	Proprietary
Ch	SoftIntegration, Inc	Yes	OS X, FreeBSD, Linux, Solaris, HP-UX, AIX, Qnx	Yes	Freeware
Clang	LLVM Project	Yes	Yes	Yes	BSD
CodeWarrior	Metrowerks	Yes	Yes	Classic Mac OS	Proprietary
CParser/libFirm	Matthias Braun, Christoph Mallon and Michael Beck	Yes	Yes	Yes	GPL
DeSmet-C	C-Ware Corporation	No	No	DOS	GPL
Digital Mars	Digital Mars	Yes	No	No	Proprietary
Dignus Systems/C	Dignus, L.L.C	Yes (host)	Yes (host)	Z/Architecture	Proprietary
Edison Design Group	Edison Design Group	Yes	Yes	Yes	Proprietary
GCC C	GNU Project	MinGW, Cygwin	Yes	IBM mainframe, AmigaOS, VAX/VMS, RTEMS	GPL
Hippo-C	Hippopotamus Software, Haba Systems	No	No	Classic Mac OS, Atari ST	Proprietary
IAR C/C++ Compilers	IAR Systems	Yes	No	No	Proprietary
Interactive C	KISS Institute for Practical Robotics	Yes	Unix, OS X, Linux, IRIX, Solaris, SunOS	No	Freeware
LabWindows/CVI	National Instruments	Yes	Yes	Yes	Proprietary
Lattice C	Lifeboat Associates	No	Yes	DOS, OS/2, Commodore, Amiga, Atari ST, Sinclair QL	Proprietary
lcc	Chris Fraser and David Hanson	Yes	Yes	Yes	Freeware (source code available for non-commercial use)
Mac C	Consulair	No	No	Classic Mac OS	Proprietary
Mark Williams C	Mark Williams Company	Yes	Coherent	Yes	Proprietary
Micro-C Compiler (mcc)	Dunfield Development Services	No	No	DOS	Freeware (source code available)
Micro C Compiler (mcc)	Roshan Singh	Yes	Yes	Yes	Freeware (source code available for non-commercial use)
MikroC Compiler	Mikroelektronika	Yes	Yes	Yes	Proprietary
MPW C	Apple	No	No	Classic Mac OS	Proprietary
Neatcc	Ali Gholami Rudi	No	Yes	No	BSD
Nwcc	Nils Weller	No	Yes	No	BSD
Open64	AMD SGI Google HP Intel Nvidia PathScale Tsinghua University and others	No	Yes	Yes	GPL
Open Watcom	Sybase and SciTech Software	Yes	Linux	OS/2, DOS	Sybase Open Watcom Public License
Orange C compiler	David I. Rauber	Yes	No	DOS	BSD

# Co jest potrzebne do pisania programów w języku C?

- Code::Blocks ([www.codeblocks.org](http://www.codeblocks.org))



- Wieloplatformowe darmowe środowisko programistyczne o otwartym kodzie źródłowym dla programistów C, C++, Fortrana





# Co jest potrzebne do pisania programów w języku C?



## Code::Blocks

Code::Blocks - The IDE with all the features you need, having a consistent look, feel and operation across platforms.

Home

Features

Downloads

Forums

Wiki

### Main

- Home
- Features
- Screenshots
- Downloads
- Plugins
- User manual
- Licensing
- Donations

### Quick links

- FAQ
- Wiki
- Forums
- Forums (mobile)
- Nightlies
- Ticket System
- Browse SVN

## The open source, cross platform, free C, C++ and Fortran IDE.

Code::Blocks is a *free C, C++ and Fortran IDE* built to meet the most demanding needs of its users. It is designed to be very extensible and fully configurable.

Finally, an IDE with all the features *you* need, having a consistent look, feel and operation across platforms.

Built around a plugin framework, Code::Blocks can be *extended with plugins*. Any kind of functionality can be added by installing/coding a plugin. For instance, compiling and debugging functionality is already provided by plugins!

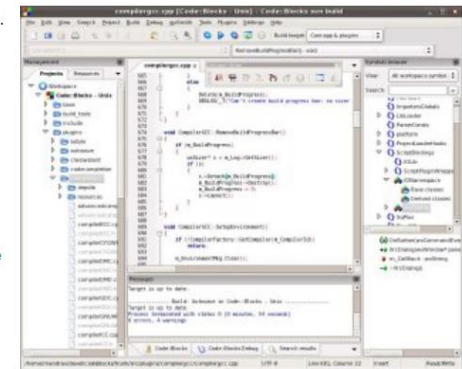
Special credits go to darmar for his great work on the **FortranProject** plugin, bundled since release 13.12.

We hope you enjoy using Code::Blocks!

*The Code::Blocks Team*

## Code::Blocks 17.12 is here!

Written by Mateusz Mielniczak





# Co jest potrzebne do pisania programów w języku C?



The screenshot shows the Code::Blocks website. At the top left is the Code::Blocks logo, a 2x2 grid of colored squares (red, green, yellow, purple). To its right is the text "Code::Blocks" in a large, bold, black font. Further right is the tagline "Code::Blocks - The IDE with all across platforms." Below this is a dark blue navigation bar with white text for "Home", "Features", "Downloads", "Forums", and "Wiki". The "Downloads" tab is selected. On the left side, there is a "Main" section with a list of links: Home, Features, Screenshots, Downloads (with sub-links for Binaries, Source, and SVN), Plugins, User manual, Licensing, and Donations. Below this is a "Quick links" section. The main content area is titled "Downloads" and contains the text: "There are different ways to download and install Code::Blocks on your computer:". Below this text is a red-bordered box containing the bullet point: "• Download the binary release". Underneath this box is the text: "This is the easy way for installing Code::Blocks. Download the setup file, run it on your computer to work with it. Can't get any easier than that!". Below this are two more bullet points: "• Download a nightly build: There are also more recent so-called *nightly builds* (for users) in **Jens' Debian repository** and **Jens' Fedora repository**. Other distributions (Ubuntu, etc.) are also available. Please note that we consider nightly builds to be *stable* for our users." and "• Download the source code". At the bottom of the main content area, there is a paragraph: "If you feel comfortable building applications from source, then this is the recommended way. Building from source code and building it yourself puts you in great control and also makes it easier for you to customize the IDE to your needs."

**Code::Blocks** Code::Blocks - The IDE with all across platforms.

Home Features Downloads Forums Wiki

**Main**

- Home
- Features
- Screenshots
- Downloads
  - Binaries
  - Source
  - SVN
- Plugins
- User manual
- Licensing
- Donations

**Quick links**

## Downloads

There are different ways to download and install Code::Blocks on your computer:

- **Download the binary release**  
This is the easy way for installing Code::Blocks. Download the setup file, run it on your computer to work with it. Can't get any easier than that!
- **Download a nightly build:** There are also more recent so-called *nightly builds* (for users) in **Jens' Debian repository** and **Jens' Fedora repository**. Other distributions (Ubuntu, etc.) are also available. Please note that we consider nightly builds to be *stable* for our users.
- **Download the source code**  
If you feel comfortable building applications from source, then this is the recommended way. Building from source code and building it yourself puts you in great control and also makes it easier for you to customize the IDE to your needs.

# Co jest potrzebne do pisania programów w języku C?

- Licensing
- Donations

## Quick links

- FAQ
- Wiki
- Forums
- Forums (mobile)
- Nightlies
- Ticket System
- Browse SVN
- Browse SVN log



 Windows XP / Vista / 7 / 8.x / 10:

File	Date	Download from
codeblocks-17.12-setup.exe	30 Dec 2017	Sourceforge.net
codeblocks-17.12-setup-nonadmin.exe	30 Dec 2017	Sourceforge.net
codeblocks-17.12-nosetup.zip	30 Dec 2017	Sourceforge.net
codeblocks-17.12mingw-setup.exe	30 Dec 2017	Sourceforge.net
codeblocks-17.12mingw_nosetup.zip	30 Dec 2017	Sourceforge.net
codeblocks-17.12mingw_fortran-setup.exe	30 Dec 2017	Sourceforge.net

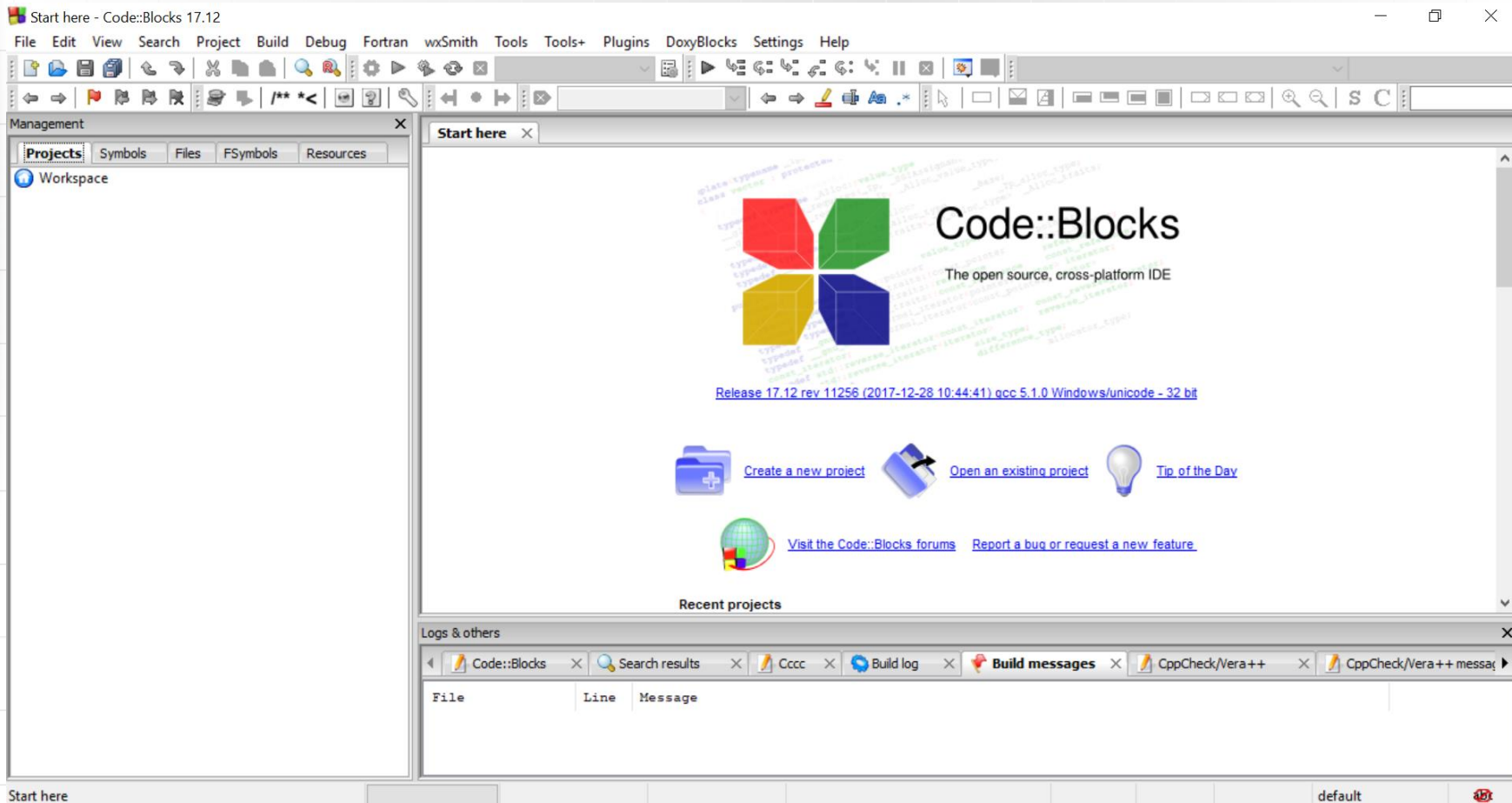
**NOTE:** The codeblocks-17.12-setup.exe file includes Code::Blocks with all plugins. The codeblocks-17.12-setup-nonadmin.exe file is provided for convenience to users that do not have administrator rights on their machine(s).

**NOTE:** The codeblocks-17.12mingw-setup.exe file includes *additionally* the GCC/G++ compiler and GDB debugger from **TDM-GCC** (version 5.1.0, 32 bit, SJLJ). The codeblocks-17.12mingw\_fortran-setup.exe file includes *additionally to that* the GFortran compiler (**TDM-GCC**).

**NOTE:** The codeblocks-17.12(mingw)-nosetup.zip files are provided for convenience to users that are allergic against installers. However, it will not allow to select plugins / features to install (it includes everything) and not create any menu shortcuts. For the "installation" you are on your own.

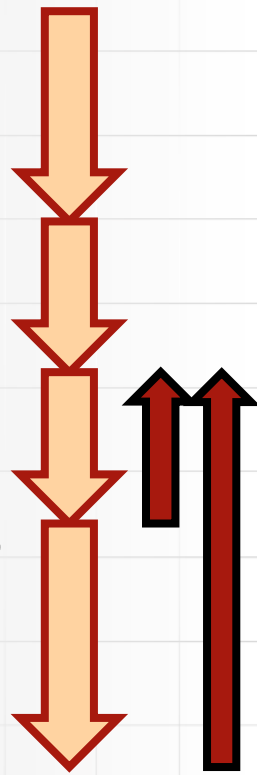
*If unsure, please use codeblocks-17.12mingw-setup.exe!*

# Co jest potrzebne do pisania programów w języku C?



# Proces programowania

1. Dokładne określenie planowanej funkcjonalności programu
2. Napisanie kodu programu
3. Skompilowanie programu
4. Sprawdzenie, czy program nie zawiera błędów, i ewentualnie ich naprawienie
5. Wykonanie programu i sprawdzenie, czy działa prawidłowo (zgodnie z oczekiwaniami), ewentualna poprawa





# Przykład kodu

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

```
main () {
```

```
    printf("Hello");
```

```
    printf(" world!\n");
```

```
    return 0;
```

```
}
```



# Przykład kodu

kompilacja

uruchomienie

The screenshot shows a code editor window titled "wdp\_l01z01.c - Code::Blocks 16.01". The menu bar includes File, Edit, View, Search, Project, Build, Debug, Fortran, wxSmith, Tools, Tools+, and Plugin. The toolbar contains icons for file operations, search, and execution. The main editor area shows the following C code:

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

Red arrows point from the text labels to the toolbar icons: "kompilacja" points to the gear icon (Build), "uruchomienie" points to the green play button (Run), and "kompilacja+uruchomienie" points to the green play button with a gear (Build and Run).



# Przykład kodu - uwagi

- starannie wpisuj kod źródłowy
- dodawaj puste wiersze i wcinaj sekcje kodu
- polecenia i funkcje standardowe pisze się małymi literami





# Funkcja `main()`

- przykłady funkcji:

```
main()  calcIt()  printf()  
strlen()
```

- przykłady poleceń:

```
return  while  int  if  float
```



# Funkcja `main()`

- wykonywanie programu **zawsze** zaczyna się od funkcji `main()`
- program **musi** zawierać funkcję `main()`
- program może zawierać więcej funkcji
- bezpośrednio za napisem `main()` znajduje się otwierający nawias klamrowy (`{`) wyznaczający początek funkcji
- koniec funkcji wyznacza zamykający nawias klamrowy (`}`)



# Funkcja `main()`

- prawie zawsze program będzie pobierał i drukował dane
- pomaga w tym dyrektywa `#include <stdio.h>`



# Rodzaje danych

- znaki
- liczby całkowite
- liczby zmiennopozycyjne



# Rodzaje danych - znaki

- każda litera, cyfra, spacja, symbol specjalny to jakiś znak

**A a 7 % ! \ + ] (**

- dane znakowe umieszcza się między apostrofami

**'A' 'a' '7' '-'**



# Rodzaje danych - znaki

- w apostrofach można umieszczać pojedyncze znaki, a nie ich ciągi
- przykłady danych nie będących prawidłowymi znakami
  - 'Wstęp do programowania'
  - 'WPPT'



# Rodzaje danych - łańcuchy

- ciąg znaków nazywamy łańcuchem
- łańcuchy umieszcza się w cudzysłowie
  - "Wstęp do programowania"
  - "WPPT"
  - "Hello"
  - " world! \n"



# Rodzaje danych - sekwencje specjalne

- w języku C istnieje kilka dwuznakowych kombinacji interpretowanych jako pojedynczy znak, np. `'\n'`





# Rodzaje danych - liczby

- liczby całkowite - nie mają części ułamkowej

12 61 0 -51

- liczby zmiennopozycyjne

12.61 0.51 0.0 -10.48 -2.016

- wybór liczb zależy od rodzaju danych, na których pracuje program





# Przykład

```
Start here x wdp_l01p02.c x
1      #include <stdio.h>
2
3      main() {
4          printf("To jest program w %c.\n",'C');
5          printf("Laboratorium %d z WdP.\n",1);
6          printf("WdP na %.1f ",99.9);
7          printf("procent.\n");
8          return 0;
9      }
10
```

funkcja `main()` to jedyna funkcja w tym programie napisana przez programistę



# Przykład

```
Start here x wdp_l01p02.c x
1      #include <stdio.h>
2
3      main() {
4          printf("To jest program w %c.\n", 'C');
5          printf("Laboratorium %d z WdP.\n", 1);
6          printf("WdP na %.1f ", 99.9);
7          printf("procent.\n");
8          return 0;
9      }
10
```

treść (ciało) funkcji `main()`  
objęta jest nawiasami klamrowymi



# Przykład

```
Start here x wdp_l01p02.c x
1      #include <stdio.h>
2
3      main() {
4          printf("To jest program w %c.\n", 'C');
5          printf("Laboratorium %d z WdP.\n", 1);
6          printf("WdP na %.1f ", 99.9);
7          printf("procent.\n");
8          return 0;
9      }
10
```

w programie wywołano funkcję `printf()`



# Podsumowanie

- Co to jest algorytm?
- Co to jest program?
- Zapis algorytmów
- Pierwszy program w języku C