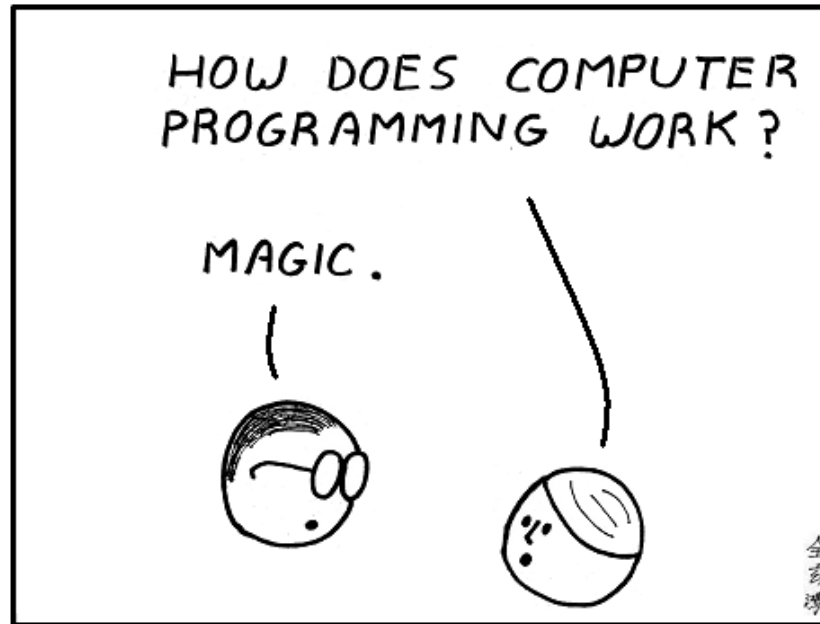


# Podstawy Programowania



[http://www.saltbox.com/img/under\\_the\\_hood.png](http://www.saltbox.com/img/under_the_hood.png)

Any sufficiently advanced technology is indistinguishable from magic.

Arthur C. Clarke

# O mnie...



**dr inż. Małgorzata Janik**

Zakład Fizyki Jądrowej

Wydział Fizyki Politechniki Warszawskiej

[\*majanik@if.pw.edu.pl\*](mailto:majanik@if.pw.edu.pl)

[\*www.if.pw.edu.pl/~majanik/wiki\*](http://www.if.pw.edu.pl/~majanik/wiki)

s. 117d GF

konsultacje: pn. 10-11, cz. 11-12

# Zasady zaliczenia

---

## Suma punktów (max 113 pkt.):

- 9 wejściówek (po 2 pkt.)
- 9 laboratoriów (po 5 pkt.)
- 4 laboratoria kontrolne (po 10 pkt.)
- 5 quizów na wykładach (po 2 pkt.)

## Ocena końcowa:

- (50, 60] – 3,0
- (60, 70] – 3,5
- (70, 80] – 4,0
- (80, 90] – 4,5
- (90, 113] – 5,0

**Warunkiem koniecznym** zaliczenia przedmiotu jest uzyskanie z części kontrolnej **ponad 20 pkt.** oraz z części laboratoryjnej **ponad 50 pkt.**

# Laboratorium komputerowe

---

**Wśród zajęć laboratoryjnych wyróżnia się:**

- laboratorium przygotowawcze – 2. tydzień zajęć (niepunktowane);
- 9 laboratoriów (45 pkt.) poprzedzonych wejściówkami (20 pkt.);
- 4 laboratoria kontrolne (40 pkt.) – 6., 10., 14. i 15. tydzień zajęć.

Dopuszcza się maksymalnie jedną nieobecność nieusprawiedliwioną na zajęciach oraz w sumie maksymalnie trzy nieobecności.

Podczas zajęć laboratoryjnych zabrania się korzystania z telefonów komórkowych i innych środków komunikacji oraz internetu – z wyjątkiem strony z wykładami i strony prowadzącego oraz udostępnionych w treści zadania *explicite* odnośników.

Podczas laboratoriów kontrolnych obowiązuje praca samodzielna (bez pomocy prowadzącego)

# Wejściówki

---

- Na zakończenie każdego wykładu prezentowane będą pytania – zagadnienia podsumowujące, utrwalające zdobytą wiedzę.
- Pytania na kolokwia wstępne (*wejściówki*) na zajęciach laboratoryjnych w następnym tygodniu będą **podobne** do pytań z ww. listy.
- Ocenianie: perfekcyjnie – 1,0; niemalże perfekcyjnie – 0,5; tak sobie lub kiepsko – 0,0 pkt. Wymagane są pełne instrukcje (np. ze średnikami), a nie rzucone „hasło”.

## Regulamin zajęć laboratoryjnych

- Dopuszcza się maksymalnie jedną nieobecność nieusprawiedliwioną na zajęciach oraz w sumie maksymalnie trzy nieobecności. W przypadkach dłuższej nieobecności (np. hospitalizacja), indywidualne warunki zaliczenia ustalone zostaną z wykładowcą.
- Warunkiem usprawiedliwienia nieobecności na zajęciach jest:
  - okazanie dokumentu - podstawy usprawiedliwienia (np. zwolnienie lekarskie);
  - okazanie uzupełnionego programu z opuszczonych zajęć (w ciągu 2 tygodni).
- Program z opuszczonych (usprawiedliwionych) zajęć może być oceniony maksymalnie na 4 pkt. Samodzielność wykonania projektu należy obronić podczas zajęć laboratoryjnych lub konsultacji.
- Studenci spóźnieni ponad 15 minut nie będą dopuszczeni do zajęć laboratoryjnych.
- Zasady oceniania programu:
  - spełnienie założeń (funkcjonalności) określonych w treści zadania, w tym wykazanie się zrozumieniem i analizą problemu: **5 pkt.**
  - kara za niepoprawność lub złą estetykę kodu: **-1 pkt.**
- Program, który po skończeniu czasu przeznaczanego na jego napisanie, nie buduje się prawidłowo, może być oceniony maksymalnie na 2 pkt.
- Podczas zajęć laboratoryjnych zabrania się korzystania z telefonów komórkowych i innych środków komunikacji oraz internetu - z wyjątkiem strony z wykładami i strony prowadzącego oraz udostępnionych w treści zadania *explicite* odnośników.
- Podczas zajęć laboratoryjnych można korzystać z własnoręcznych notatek, wydrukowanych slajdów z wykładów oraz podręczników do nauki języka C.
- Podczas zajęć kontrolnych wymagana jest 100% praca samodzielna. Można natomiast korzystać ze slajdów wykładowych, wcześniejszych własnych programów oraz notatek i podręczników do programowania.

**UWAGA!** Studenci łamiący postanowienia regulaminu (np. przeglądanie internetu, korzystanie z telefonów komórkowych lub powielanie cudzych programów) zostaną zdyskwalifikowani z zajęć laboratoryjnych (tj. kończą zajęcia z wynikiem 0, niezależnie od stopnia zaawansowania programu)

# Konfiguracja poczty

[https://pp.fizyka.pw.edu.pl/poczta\\_pw/](https://pp.fizyka.pw.edu.pl/poczta_pw/)

W toku tych zajęć używana będzie poczta studencka:  
→ domena **pw.edu.pl**

Webmail: <https://portal.office.com/>

Polecam skonfigurować programy pocztowe do odbierania / przekierowywania wspomnianej poczty.

# Komendy systemu linux

**ls** (list) - wyświetla zawartość bieżącego katalogu lub katalogu podanego jako parametr.

**cd** (change directory) - wchodzi do katalogu, np. **cd katalog1**

**cd ..** - wchodzi do katalogu wyżej

**mkdir** (make directory) - do tworzenia katalogów. Przykład: **mkdir nazwa\_katalogu**

**cp** (copy) - do kopiowania plików i katalogów. Przykłady: **cp plik1 plik2**

**cp -r** - kopiuje katalog wraz z zawartością np. **cp -r katalog1 katalog2**

**\*** - gwiazdka zastępuje dowolny ciąg znaków np.:

**cp \* alfa/** - kopiuje wszystkie pliki z bieżącego katalogu do katalogu alfa

**mv** (move) - przenosi plik/pliki, służy też do zmiany nazwy pliku lub katalogu.

**mv plik1 plik2** - zmienia nazwę plik1 na plik2

**rm** (remove) - usuwa pliki. Przykład:

**rm plik1** - usuwa plik1

**rm \*** - usuwa wszystkie pliki z bieżącego katalogu (należy używać bardzo ostrożnie - sprawdzić, czy rzeczywiście chcemy wszystko skasować).

**rm -r** - usuwa cały katalog razem z zawartością

**more** - pozwala na przeglądanie danych (plików, komunikatów poleceń) ekran po ekranie.

**kate plik.txt** – uruchamia edytor kate tworząc plik plik.txt

**cat** - podobnie do polecenia 'more' pokazuje zawartość pliku ale nie zatrzymuje się ekran po ekranie tylko wyświetla od razu całość.



# Prosta kompilacja programu – Linux

Plik z kodem źródłowym: **program00.c**

(pliki z kodem źródłowym języka C powinny mieć rozszerzenie .c)

Plik wynikowy: **program00**

(w środowisku linux programy nie posiadają rozszerzenia, lecz wyróżnia je flaga wykonywalności 'x')

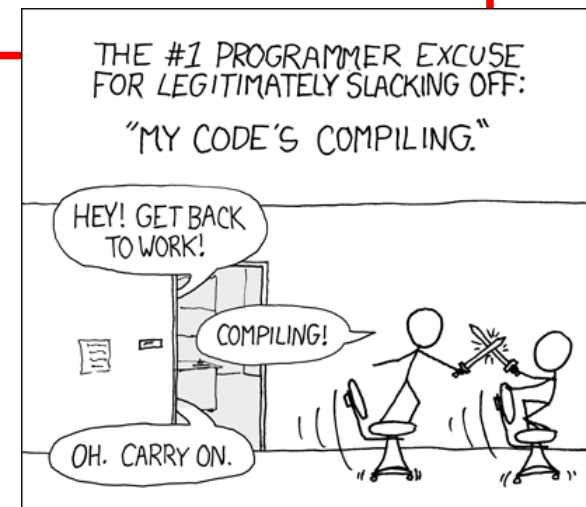
```
gcc -o program00 program00.c -Wall -pedantic -std=c99
```

## Flagi kompilacji:

- Wall – wyświetla wszystkie ostrzeżenia
- pedantic – wyświetla niezgodności ze standardem ISO
- std=c99 – stosuj standard C99

## Dodatkowo:

- O2 – optymalizacja kompilacji i kodu programu



<http://imgs.xkcd.com/comics/compiling.png>

# Pierwszy program

```
/******  
 * Jan Kowalski *  
 * 15.03.2013 r. *  
******/
```

Komentarz blokowy - dowolny tekst  
pomiędzy znakami /\* oraz \*/

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

Instrukcja preprocesora  
- zaczyna się od znaku #

```
int main (void)  
{
```

Funkcja main() - tutaj  
zaczyna się sterowanie programem

```
 // Wyświetla linię tekstu  
 printf("Moj pierwszy program!");
```

Instrukcja  
- linijki na ogół kończą się średnikiem

```
 return 0; // kończy program
```

Komentarz  
- zaczyna się od //  
kończy wraz z końcem linii

```
}
```

# Wypisywanie na ekran

```
int main (void)
{
    puts("Hello world!");
    printf("Hello world 2!\n");
    printf("Hello world %d!\n",3);
    return 0;
}

puts(„Napis”); // pisanie po ekranie
printf(„Napis\n”); // \n - oznacza znak nowej linii
printf(„%d %f %c”,zmienna_int, zmienna_float, zmienna_char);
//wypisywanie zmiennych
```

# (1) „Hello world”

Wypisać na ekran (w terminalu) słowa „Hello World!”

- stworzymy nowy plik tekstowy, nadajemy mu nazwę `hello.c`
- na początku załączamy bibliotekę: `#include <stdio.h>`
- tworzymy funkcję main

```
int main(void)
{
    //tu będziemy wpisywać kod
    return 0;
}
```

- w środku funkcji wypisujemy słowo przy użyciu „puts” : `puts("Napis!");`
- kompilujemy – przez terminal (w terminalu wpisujemy:

```
gcc -Wall -pedantic -O2 -std=c99 hello.c -o hello //jesli hello.c to nasza nazwa pliku
```

- wypisujemy to samo przy użyciu „printf”

## (2) „Typy zmiennych”

Stworzyć funkcję główną (main) w której należy kolejno (można użyć istniejącej):

```
puts("-----");
```

- zadeklarować zmienną całkowitą  $a = 5$  i wypisać ją na ekran (`int a = 5;`)
- zadeklarować zmienną zmiennoprzecinkową  $b = 3.5$  i wypisać ją na ekran (`double .... ;`)
- zadeklarować zmienną zmiennoprzecinkową  $c$  która będzie wynikiem sumowania zmiennych  $a$  i  $b$  (`c = a + b;`)
- wypisać na ekran napis:  $a + b = c$  oraz odpowiednio to samo równanie używając wartości zmiennych (wskazówka: `printf("napis %d + %lf = %lf", x,y,z);`)
- zadeklarować zmienną typu „char” (napis) 'a'. Wypisać ją na ekran.

```
int %d
float %f
double %lf → tylko dwa miejsca po przecinku: %.2lf
char %c
```

# Wczytaj i wypisz

```
int main (void)
{
    int n;
    scanf("%d", &n);

    printf("%d\n",n);
    return 0;
}
```

`scanf("%d", &n);` // standardowe wejście (klawiatura) wpisz do zmiennej

```
if(scanf("%d", &n))
    printf("%d\n",n); // bez ostrzeżenia
```

# (3) scanf

Stworzyć funkcję główną (main) w której należy kolejno (można użyć istniejącej):

```
puts("-----");
```

- wypisać na ekran napis „Ile masz lat?”
- poprosić użytkownika programu o wprowadzenie liczby z klawiatury (`scanf("%d", &wiek);`) (wiek jest liczbą całkowitą!)
- wypisać podany przez użytkownika wiek w postaci „Mam X lat”
- poprosić użytkownika o wprowadzenie pierwszej litery imienia z klawiatury (litera jest znakiem char!)
- poprosić użytkownika o wprowadzenie pierwszej litery nazwiska z klawiatury (litera jest znakiem!)
- wypisać na ekranie „Moje inicjały to X. Y.”

# Instrukcja warunkowa „if”

```
int main (void)
{
```

```
    int n;
    scanf("%d", &n);
```

```
    if (n >= 0)
```

```
    {
```

```
        printf( "Liczba naturalna");
```

```
    }
```

```
    return 0;
```

```
}
```

Jeśli n większe równe 0

wtedy rób to co w klamrach



# Instrukcja warunkowa „if”

```
int main (void)
{
    int n;
    scanf("%d", &n);
    if (n >= 0)
    {
        printf( "Liczba naturalna.\n");
    }
    else
    {
        printf( "Liczba mniejsza niż 0.\n");
    }
    return 0;
}
```

wtedy rób to co w kolejnych klamrach

**W przeciwnym wypadku**

# (3) „Jeśli”

Stworzyć funkcję główną (main) w której należy kolejno(można użyć istniejącej):

```
puts("-----");
```

- wypisać na ekranie „Ile masz lat?”

- poprosić użytkownika o wprowadzenie liczby całkowitej z klawiatury

- jeśli użytkownik podał wiek mniejszy niż 18 lat wypisać: „Nie masz 18 lat!” , jeśli większy to wypisać „ Masz XXX lat i możesz przeczytać ten tekst!”

Przykład użycia w kodzie programu „jeśli”

```
if(a > 5)
{
    puts("Liczba a jest większa niż 5!");
}
else
{
    puts("Liczba a jest mniejsza niż 5!");
}
```

# Instrukcja warunkowa „if”

```
int main (void)
{
    int n;
    scanf("%d", &n);
    if (n > 0)
    {
        printf( "Liczba większa niż 0.\n");
    }
    else if(n == 0)
    {
        printf( "Liczba równa 0.\n");
    }
    else
    {
        printf( "Liczba mniejsza niż 0.\n");
    }
    return 0;
}
```

- dopisać jeszcze: jeśli użytkownik podał wiek pomiędzy 16 a 17 lat (włącznie) wypisać: "Już niedługo!"

„i” logiczne to „&&”, czyli np. warunek (a>3 i a <8) to (a>3 && a<8)

# Pętla „for”

```
int main (void)
{
    int n;
    scanf("%d", &n);
    printf("%d\n", n);

    for (int i=1;i<=n;i++)
    {
        printf("%d ",i); // ...
    }

    return 0;
}
```

Pętla „for”: (int i=1;i<=n;i++)

Zaczynając od i równego 0 (int i = 1), do i mniejszego równego n (i<=n), wykonuj raz po raz to co jest w pętli { ... }, przy każdej iteracji zwiększając i (i++)

Czyli: n razy wykonaj to, co jest w pętli za każdym razem zwiększając i

# (4) „pętla for”

Stworzyć funkcję główną (main) w której należy kolejno (można użyć istniejącej):

```
puts("-----");
```

- wypisać na ekranie liczby od 1 do 50

- poprosić użytkownika o wprowadzenie liczb całkowitych (a i b) z klawiatury i wypisać na ekranie co drugą liczbę w przedziale od a do b

```
for (int i=1;i<=n;i++)  
    {  
        // ...  
    }
```

# Indentacja

```
int main (void)
{
    int n;
    scanf("%d", &n);
    if (n > 0)
    {
        printf( "Liczba większa niż 0.\n");
    }
    else if(n == 0)
    {
        printf( "Liczba równa 0.\n");
    }
    else
    {
        printf( "Liczba mniejsza niż 0.\n");
    }
    return 0;
}
```

Brak wcięć nie powoduje błędów kompilacji, jednak prawidłowe używanie wcięć zwiększa czytelność kodu!

# (5) Naucz brata dodawania

Stworzyć funkcję główną (main) w której należy kolejno:

- stworzyć pętlę while

```
bool koniec = false;
```

```
while(!koniec)
```

```
{
```

```
...
```

```
}
```

- pobrać od użytkownika pojedynczy znak z klawiatury (int `a`)

- w zależności od podanego znaku wykonać jedną z trzech rzeczy (1) lub (2)

- jeśli (2) to wyjdź z programu (zmienną **koniec** należy ustawić na **true**) `koniec = true;`

- jeśli (1) to poproś użytkownika o podanie dwóch liczb, następnie poproś użytkownika o podanie sumy tych dwóch liczb. Jeśli podał prawidłową wartość, wypisz „Poprawny wynik!” jeśli zaś nieprawidłowy, to wypisz „Wynik niepoprawny, poprawny wynik to XXX”.

# (6) Kalkulator

Dodawanie  
Odejmowanie  
Mnożenie  
Dzielenie  
Silnia  
Modulo (reszta z dzielenia)  
1/x  
Pierwiastek

Jakie działanie chcesz wykonać?

+

Podaj dwie liczby:

2 3

Wynik:

$2+3=5$

Jakie działanie chcesz wykonać?

\*

Podaj dwie liczby:

4 3

Wynik:

$4*3=12$