Języki Programowania

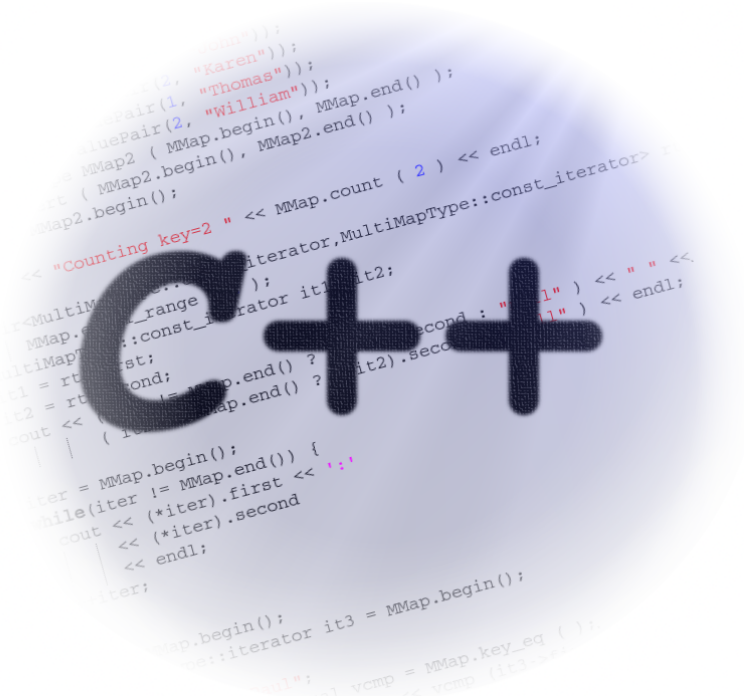
Prowadząca:
dr inż. Hanna Zbroszczyk

e-mail: hanna.zbroszczyk@*pw.edu.pl*
tel: +48 22 234 58 51

Konsultacje:
Piątek: 14:00-15:30

www: http://www.if.pw.edu.pl/~gos/students/jp

Politechnika Warszawska
Wydział Fizyki
Pok. 117b (wejście przez 115)



REGULAMIN PRZEDMIOTU

Informacje ogólne

Zajęcia trwają 15 tygodni (2 godziny wykładu, 2 godziny laboratorium tygodniowo)

Zaliczenie zajęć jest uwarunkowane zaliczeniem zajęć laboratoryjnych

Prowadzący zajęcia laboratoryjne:

dr inż. Łukasz Graczykowski

dr inż. Małgorzata Janik

dr inż. Daniel Kikoła

mgr inż. Diana Pawłowska

mgr inż. Sebastian Siejka

mgr inż. Maria Stefaniak

mgr inż. Paweł Szymański

dr inż. Hanna Zbroszczyk

Strona przedmiotu: http://www.if.pw.edu.pl/~gos

Organizacja zajęć laboratoryjnych

- Przewidzianych jest 14 zajęć laboratoryjnych

(w tym 11 punktowanych, 2 kolokwia, 1 dodatkowe);

- Zajęcia rozpoczynają się od drugiego tygodnia semestru;
- Obecność jest obowiązkowa (możliwe są maksymalnie 2 nieobecności);
- W przypadku osób, które uzyskały rejestrację na semestr w trakcie jego trwania – koniecznym warunkiem do zdobycia pozytywnej oceny z przedmiotu będzie zaliczenie pierwszego kolokwium w terminie;
- Spóźnienie na zajęcia powyżej 15 minut automatycznie jest odnotowane jako nieobecność;
- Zajęcia trwają 90 minut, odbywają się bez przerwy;

Zasady oceniania na zajęciach punktowanych - I

- zajęcia punktowane obejmują wykonanie 11 (jedenastu) zadań o zróżnicowanym stopniu trudności (pierwsze zajęcia są także punktowane);
- dopuszczenie do wykonania zadania może być uwarunkowane zaliczeniem kolokwium wstępnego;
- w trakcie pisania programu wolno korzystać z napisanych przez siebie programów oraz zasobów Internetu*;
- napisany w trakcie trwania laboratorium program należy oddać na tych samych zajęciach;
- przynajmniej jeden, a maksymalnie dwa programy będą pisane w dwuosobowych zespołach;
- przynajmniej jeden, a maksymalnie dwa programy będą pisane przez dwa, niekoniecznie następujące po sobie zajęcia;
- za każde zadanie można otrzymać 0-5 pkt (zrozumienie zadania: 1pkt, wykorzystanie formalnych środków języka C++: 3 pkt, aspekty użytkowe oraz strona estetyczna: 1 pkt)

*) nie wolno korzystać z programów pocztowych (chyba, że prowadzący wyrazi zgodę), komunikatorów internetowych, serwisów społecznościowych (w celu komunikacji z innymi użytkownikami), ani z programów kolegów z grupy swojej, jak i żadnej innej; korzystanie z telefonów komórkowych (smartfonów, tabletów) jest także zabronione.

Zasady oceniania na zajęciach punktowanych - II

- w przypadku nieskończenia programu na zajęciach oceniony zostanie napisany, skompilowany oraz działający jego fragment (w przypadku programu, który nie kompiluje, ani nie wykonuje się poprawnie możliwe jest zdobycie maksymalnie 2 pkt); program należy skończyć we własnym zakresie i przedstawić prowadzącemu najpóźniej w kolejnym tygodniu zajęć (na zajęciach lub konsultacjach); za skończenie programu po zajęciach możliwe będzie zdobycie dodatkowego 1 pkt- ale tylko w przypadku przedstawienia w pełni działającego programu; suma zdobytych punktów za program skończony poza zajęciami nie może być większa niż 4; poprawa polega na zademonstrowaniu działającego programu oraz dyskusji z prowadzącym (co w przypadku prezentacji na kolejnych zajęciach skraca czas pisania programu dedykowanego dla tych konkretnych zajęć); nie dokończenie programu może skutkować niedopuszczeniem do kolejnych zajęć;

Zasady oceniania na zajęciach punktowanych - III

- w przypadku nieobecności studenci są zobowiązani do zrealizowania materiału we własnym zakresie i przedstawienia rozwiązania najdalej 2 tygodnie po nieobecności (na zajęciach lub konsultacjach) - w przypadku usprawiedliwionej nieobecności możliwe jest zaliczenie zaległego programu na mniejszą (4 pkt) ilość punktów; w przypadku nieobecności nieusprawiedliwionej liczba zdobytych punktów wynosi 0 (zero); nie nadrobienie zaległości (zarówno w przypadku nieobecności usprawiedliwionej i nieusprawiedliwionej) może skutkować niedopuszczeniem do kolejnych zajęć;

Zasady oceniania kolokwiów - I

- w trakcie semestru będą 2 (dwa) kolokwia: jedno w połowie semestru, drugie na końcu;

– kolokwium będzie polegało na samodzielnym napisaniu 1 (jednego) programu z materiału zrealizowanego na zajęciach (możliwe jest jednak korzystanie z:

  - własnych programów z zajęć,
  - materiałów wykładu dostępnych w trybie “offline” lub w wersji papierowej,
  - podręczników do programowania w C oraz C++,
  - własnych notatek);

- próby niesamodzielnej pracy będą skutkowały niezaliczeniem kolokwium oraz brakiem możliwości jego poprawy;

Zasady oceniania kolokwiów - II

- napisany program należy przesłać przed końcem trwania kolokwium na adres e-mailowy prowadzącego;
- program będzie oceniany w skali 0-20 pkt (pierwsze kolokwium) oraz w skali 0-25 pkt (drugie kolokwium); oceniane będą:
    - zakres merytoryczny zrealizowanego zadania,
    - wykorzystane środki formalne języka C++,
    - aspekty użytkowe interfejsu,
    - strona estetyczna;

Zasady oceniania kolokwiów - III

Istnieje możliwość poprawy kolokwium na ostatnich zajęciach (w grupie swojej lub innej), przy pierwszej poprawie kolokwium możliwe będzie zdobycie maksymalnie -5 pkt mniej niż w pierwszym terminie, przy drugiej poprawie – 10 pkt mniej.

Zaliczenie obu kolokwiów jest jednym z warunków zaliczenia przedmiotu!
(warunkiem zaliczenia kolokwium jest otrzymanie za jego napisanie minimum 51% punktów możliwych do zdobycia)

Ocena końcowa - I

Wyniki z kolokwium z laboratorium: 1 * 20 pkt + 1 * 25 pkt = 45 pkt;

Wyniki z programów napisanych na zajęciach 11 * 5 pkt = 55 pkt.

Ocena końcowa wystawiana jest na podstawie procentowego udziału sumy Uzyskanych punktów do sumy punktów możliwej do uzyskania (100 pkt) wg. następującej zależności:

POZIOM ZAAWANSOWANY

>= 51% - 3.0

>= 61% - 3.5

>= 71% - 4.0

>= 81% - 4.5

>= 91% - 5.0

POZIOM PODSTAWOWY

>= 51% - 3.0

>= 67% - 3.5

>= 84% - 4.0

Do realizacji przedmiotu na poziomie zaawansowanym zostało zaproszonych 20 osób, które uzyskały najwyższą liczbę punktów (powyżej 99) w ramach przedmiotu Podstawy Programowania (w Języku C).

Ocena końcowa - II

Osoby, (tylko poziom podstawowy) które do końca grudnia uzyskają przynajmniej 95% punktów możliwych do zdobycia mogą ubiegać się o napisanie poza zajęciami dodatkowego programu (indywidualnie uzgodnionego z prowadzącym zajęcia), który umożliwi uzyskanie oceny 5.0.

Nie ma możliwości podniesienia niższej niż 4.0 oceny dodatkowo napisanym programem.

Osoby uczęszczające na wykłady (dozwolona **jedna!** nieobecność) mogą mieć podwyższoną ocenę z przedmiotu o 0.5 oceny w przypadku zaliczonych zajęć laboratoryjnych oraz zaliczonych obu kolokwiów w pierwszym terminie.

Zaliczenie eksternistyczne (tylko na poziomie zaawansowanym) - I

Dla osób programujących w C++ możliwe jest zaliczenie przedmiotu projektem eksternistycznym. Osoby chcące zaliczyć przedmiot w tej formie powinny zgłosić się do prowadzącego najdalej na drugich zajęciach laboratoryjnych, na trzecich zajęciach napiszą kolokwium kwalifikujące do pracy w tym trybie.

Wymagania do projektów eksternistycznych:

- nietrywialny problem, do którego rozwiązania najlepiej nadaje się podejście obiektowe,
- dokładna specyfikacja projektu,
- stworzony projekt z dokumentacją w kodzie źródłowym,
- dokumentacja użytkownika.

Zaliczenie eksternistyczne (tylko na poziomie zaawansowanym) - II

I) Zaliczenie projektu eksternistycznego polega na zaliczeniu 3 (trzech) etapów kontrolnych w terminach zajęć podanych w nawiasach: *beta (5)*, *release candidate (10)*, *final (15)*.

II) Po etapie *beta* prowadzący może projekt zdyskwalifikować, dlatego do tego czasu zalecane jest uczestniczenie w zajęciach programowych.

III) Po etapie *release candidate*, w przypadku braku możliwości skończenia projektu zawierającego wszystkie elementy języka omawiane na wykładzie prowadzący może projekt zamknąć. Od tego momentu należy uczestniczyć w zajęciach.

IV) Przy ustalaniu oceny ostatecznej brane pod uwagę są oceny z etapów pośrednich.

Zalecana literatura

1) B. Stroustrup – Język C++ (The C++ Programming Language), WNT 2002

2) J. Gręboszc – Symfonia C++ standard, Pasja C++, Edition 2005

3) B. Eckel - Thinking in C++. Edycja polska, Helion 2002

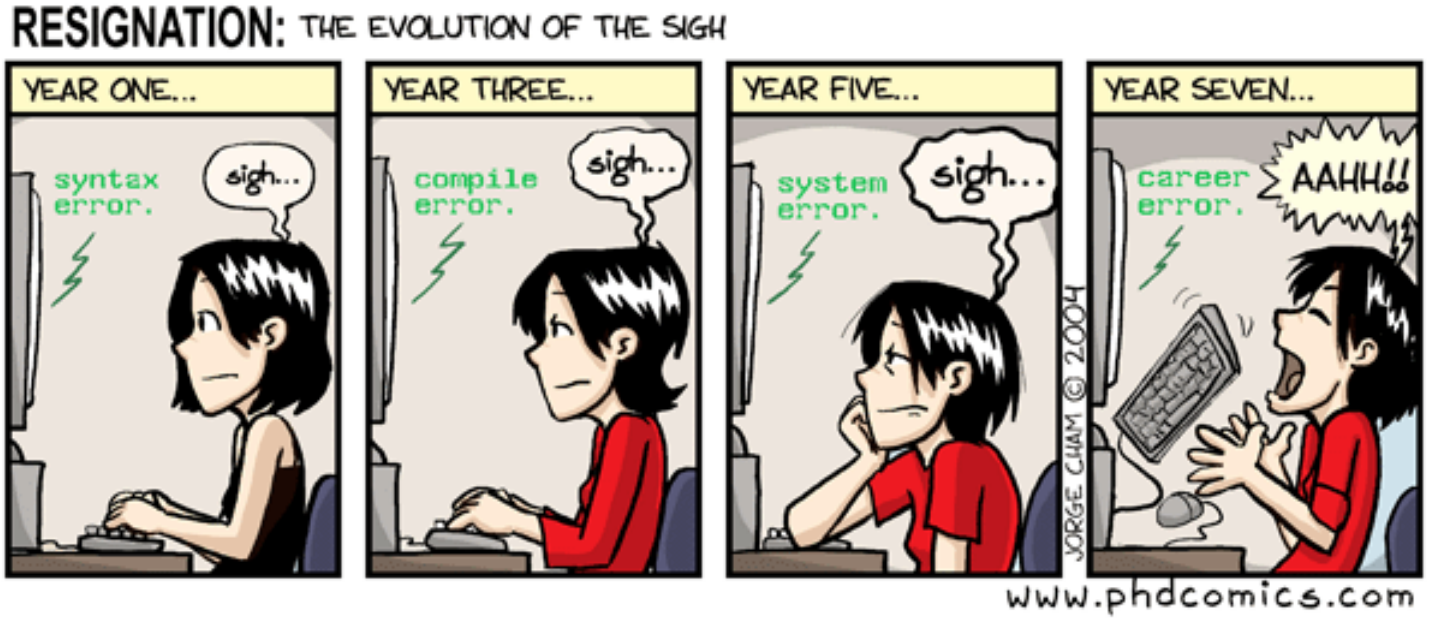
4) S.B. Lippman – Podstawy języka C++ (C++ Primer), WNT 1997

5) J. Liberty – Poznaj C++ w 10 minut, Intersoftland 1999

6) Nicolai M. Josuttis - C++ Biblioteka standardowa. Podręcznik programisty, Helion 2003

Program wykładu

1) Wprowadzenie (zasady zaliczenia przedmiotu), literatura. Język C, a C++.
   Typy referencyjne.
2) Przeładowanie nazw funkcji, wprowadzenie do klas.
3) Konstruktory, destruktory, funkcje zaprzyjaźnione.
4) Przeładowanie operatorów.
5) Dziedziczenie.
6) Funkcje wirtualne.
7) Operacje wejścia / wyjścia. Operacje na plikach.
8) Szablony funkcji.
9) Szablony klas.
10) Elementy biblioteki STL
11) Obsługa sytuacji wyjątkowych.
12) Algorytmy, struktury danych I.
13) Algorytmy, struktury danych II.
14) Konwertery oraz konwersje.


RESIGNATION: THE EVOLUTION OF THE SIGH
YEAR ONE...
syntax error.
sigh...
YEAR THREE...
compile error.
sigh...
YEAR FIVE...
system error.
sigh...
YEAR SEVEN...
career error.
AAHH!!
JORGE CHAM © 2004
www.phdcomics.com

JĘZYK C, A C++

Stwórzmy prosty program do operacji na liczbach zespolonych:
suma, różnica, iloczyn.

Najpierw napiszemy go w C (jedynie z funkcją główną),
następnie także w C, lecz pisząc oddzielne funkcje do każdej operacji,
a następnie przepiszemy go w C++ (w różnych wariantach).

Operacje na liczbach zespolonych – C (jedna funkcja)

```
#include<stdio.h>

struct cmplx {
	float rez, imz;
};

int main(){
	cmplx t[5];

	printf("\n Podaj czesc rzeczywista liczby zespolonej\n");
	scanf("%lf", &t[0].rez);
	printf("\n Podaj czesc urojona liczby zespolonej\n");
	scanf("%lf", &t[0].imz);
	printf("\n Oto liczba zespolona: %lf+%lfi\n",t[0].rez,t[0].imz);

	printf("\n Podaj czesc rzeczywista liczby zespolonej\n");
	scanf("%lf", &t[1].rez);
	printf("\n Podaj czesc urojona liczby zespolonej\n");
	scanf("%lf", &t[1].imz);
	printf("\n Oto liczba zespolona: %lf+%lfi\n",t[1].rez,t[1].imz);
```

```
	/*suma*/
	t[2].rez=t[0].rez+t[1].rez;
	t[2].imz=t[0].imz+t[1].imz;
	printf("\n Oto suma liczb zespolonych:
			%lf+%lfi\n",t[2].rez,t[2].imz);

	/*roznica*/
	t[3].rez=t[0].rez-t[1].rez;
	t[3].imz=t[0].imz-t[1].imz;
	printf("\n Oto roznica liczb zespolonych:
			%lf+%lfi\n",t[3].rez,t[3].imz);

	/*iloczyn*/
	t[4].rez=(t[0].rez*t[1].rez)-(t[0].imz*t[1].imz);
	t[4].imz=(t[0].rez*t[1].imz)+(t[0].imz*t[1].rez);
	printf("\n Oto iloczyn liczb zespolonych:
			%lf+%lfi\n",t[4].rez,t[4].imz);

	return 0;
}
```

Operacje na liczbach zespolonych – C (wiele funkcji) - I

```
#include<stdio.h>

struct cmplx {
	float rez, imz;
};

void read(cmplx *z) {
	printf("\n Podaj czesc rzeczywista liczby zespolonej\n");
	scanf("%f", &(z->rez));
	printf("\n Podaj czesc urojona liczby zespolonej\n");
	scanf("%f", &(z->imz));
}
void print(cmplx z){
	printf("\n Liczba zespolona: %f+%fi\n",z.rez,z.imz);
}

cmplx sum(cmplx z1, cmplx z2){
	cmplex z3;
	z3.rez = z1.rez+z2.rez;
	z3.imz= z1.imz+z2.imz;
	return z3;
}
```

```
cmplx dif(cmplx z1, cmplx z2){
	cmplx z3;
	z3.rez = z1.rez-z2.rez;
	z3.imz= z1.imz-z2.imz;
	return z3;
}

cmplx mul(cmplx z1, cmplx z2){
	cmplx z3;
	z3.rez=(z1.rez*z2.rez)-(z1.imz*z2.imz);
	z3.imz=(z1.rez*z2.imz)+(z1.imz*z2.rez);
	return z3;
}

int main()
	{
	  cmplx t[5];

	  printf("\n Oto pierwsza liczba zespolona: \n");
	  read(&t[0]);
	  print(t[0]);
```

Operacje na liczbach zespolonych – C (wiele funkcji) - II

```
        printf("\n Oto druga liczba zespolona: \n");
        read(&t[1]);
        print(t[1]);

        /*suma*/
        printf(" \n Oto suma liczb zespolonych: \n");
        t[2]=sum(t[0],t[1]);
        print(t[2]);

        /*roznica*/
        printf(" \n Oto roznica liczb zespolonych: \n");
        t[3]=dif(t[0],t[1]);
        print(t[3]);

        /*iloczyn*/
        printf(" \n Oto iloczyn liczb zespolonych: \n");
        t[4]=mul(t[0],t[1]);
        print(t[4]);

        return 0;
}
```

Operacje na liczbach zespolonych – C++ - wersja 1 - I

```
#include<iostream>

using namespace std;

struct cmplx {
        float rez, imz;
        void read() {
                cout<<endl<<"Podaj czesc rzeczywista
                        liczby zespolonej"<<endl;
                cin>>rez;
                cout<<endl<<"Podaj czesc urojona
                        liczby zespolonej"<<endl;
                cin>>imz;

        }
        void print() {
                cout<<endl<<"Liczba zespolona: "
                        <<rez<<"+i"<<imz<<endl;
        }
};
```

```
cmplx sum(cmplx z1, cmplx z2){
        cmplx z3;
        z3.rez = z1.rez+z2.rez;
        z3.imz= z1.imz+z2.imz;
        return z3;
}

cmplx dif(cmplx z1, cmplx z2){
        cmplx z3;
        z3.rez = z1.rez-z2.rez;
        z3.imz= z1.imz-z2.imz;
        return z3;
}

cmplx mul(cmplx z1, cmplx z2){
        cmplx z3;
        z3.rez=(z1.rez*z2.rez)-(z1.imz*z2.imz);
        z3.imz=(z1.rez*z2.imz)+(z1.imz*z2.rez);
        return z3;
}
```

Operacje na liczbach zespolonych – C++ - wersja 1 - II

```
int main()
	{
	 cmplx t[5];

	 cout<<endl<<"Oto pierwsza liczba zespolona:
		"<<endl;
	 t[0].read();
	 t[0].print();

	 cout<<endl<<"Oto druga liczba zespolona:
		"<<endl;
	 t[1].read();
	 t[1].print();

	 //suma
	 cout<<"Oto suma liczb zespolonych:
		"<<endl;
	 t[2]=sum(t[0],t[1]);
	 t[2].print();
	 //roznica
	 cout<<"Oto roznica liczb zespolonych:
		"<<endl;
	 t[3]=dif(t[0],t[1]);
	 t[3].print();

	 //iloczyn
	 cout<<"Oto iloczyn liczb zespolonych: "<<endl;
	 t[4]=mul(t[0],t[1]);
	 t[4].print();

	 return 0;
}
```

Operacje na liczbach zespolonych – C++ - wersja 2 - I

```
#include<iostream>

using namespace std;

struct cmplx {
        float rez, imz;
        void read() {
                cout<<endl<<"Podaj czesc rzeczywista
                        liczby zespolonej"<<endl;
                cin>>rez;
                cout<<endl<<"Podaj czesc urojona
                        liczby zespolonej"<<endl;
                cin>>imz;

        }
        void print() {
                cout<<endl<<"Liczba zespolona: "
                        <<rez<<"+i"<<imz<<endl;
        }
};
```

```
cmplx operator+(cmplx z1, cmplx z2){
        cmplx z3;
        z3.rez = z1.rez+z2.rez;
        z3.imz= z1.imz+z2.imz;
        return z3;
}

cmplx operator-(cmplx z1, cmplx z2){
        cmplx z3;
        z3.rez = z1.rez-z2.rez;
        z3.imz= z1.imz-z2.imz;
        return z3;
}

cmplx operator*(cmplx z1, cmplx z2){
        cmplx z3;
        z3.rez=(z1.rez*z2.rez)-(z1.imz*z2.imz);
        z3.imz=(z1.rez*z2.imz)+(z1.imz*z2.rez);
        return z3;
}
```

Operacje na liczbach zespolonych – C++ - wersja 2 - II

```
int main()
        {
         cmplx t[5];

         cout<<endl<<"Oto pierwsza liczba
                zespolona: "<<endl;
         t[0].read();
         t[0].print();

         cout<<endl<<"Oto druga liczba
                zespolona: "<<endl;
         t[1].read();
         t[1].print();

                  //suma
         cout<<"Oto suma liczb zespolonych: "
                <<endl;
         t[2]=t[0]+t[1];
         t[2].print();
```

```
                  //roznica
         cout<<"Oto roznica liczb zespolonych: "
                <<endl;
         t[3]=t[0]-t[1];
         t[3].print();

          //iloczyn
         cout<<"Oto iloczyn liczb zespolonych: "
                <<endl;
         t[4]=t[0]*t[1];
         t[4].print();

         return 0;
}
```

Operacje na liczbach zespolonych – C++ - wersja 3 - I

cmplx.h

```
#ifndef _CMPLX_H
#define _CMPLX_H

struct cmplx {
        float rez, imz;
        void read();
        void print();
};

cmplx operator+(cmplx, cmplx);
cmplx operator-(cmplx, cmplx);
cmplx operator*(cmplx, cmplx);

#endif
```

Operacje na liczbach zespolonych – C++ - wersja 3 - II

cmplx.cpp

```
#include "cmplx.h"
#include <iostream>
using namespace std;

void cmplx::read() {
        cout<<endl<<"Podaj czesc rzeczywista
                liczby zespolonej"<<endl;
        cin>>rez;
        cout<<endl<<"Podaj czesc urojona
                liczby zespolonej"<<endl;
        cin>>imz;

}
void cmplx::print() {
        cout<<endl<<"Liczba zespolona: "
                <<rez<<"+i"<<imz<<endl;
    }
```

```
cmplx operator+(cmplx z1, cmplx z2){
        cmplx z3;
        z3.rez = z1.rez+z2.rez;
        z3.imz= z1.imz+z2.imz;
        return z3;
}

cmplx operator-(cmplx z1, cmplx z2){
        cmplx z3;
        z3.rez = z1.rez-z2.rez;
        z3.imz= z1.imz-z2.imz;
        return z3;
}

cmplx operator*(cmplx z1, cmplx z2){
        cmplx z3;
        z3.rez=(z1.rez*z2.rez)-(z1.imz*z2.imz);
        z3.imz=(z1.rez*z2.imz)+(z1.imz*z2.rez);
        return z3;
}
```

Operacje na liczbach zespolonych – C++ - wersja 3 - III

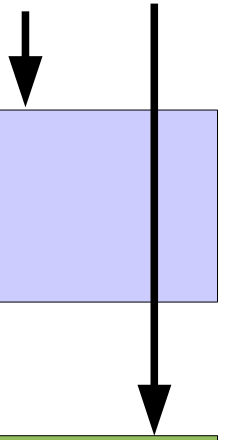
main.cpp

```
#include "cmplx.h"
#include <iostream>
using namespace std;

int main()
        {
          cmplx t[5];

          cout<<endl<<"Oto pierwsza liczba
                zespolona: "<<endl;
          t[0].read();
          t[0].print();

          cout<<endl<<"Oto druga liczba
                zespolona: "<<endl;
          t[1].read();
          t[1].print();

          //suma
          cout<<"Oto suma liczb zespolonych: "
                <<endl;
          t[2]=t[0]+t[1];
          t[2].print();

          //roznica
          cout<<"Oto roznica liczb zespolonych: "<<endl;
          t[3]=t[0]-t[1];
          t[3].print();

          //iloczyn
          cout<<"Oto iloczyn liczb zespolonych: "<<endl;
          t[4]=t[0]*t[1];
          t[4].print();

          return 0;
}
```

Dzielenie kodu na kilka plików źródłowych

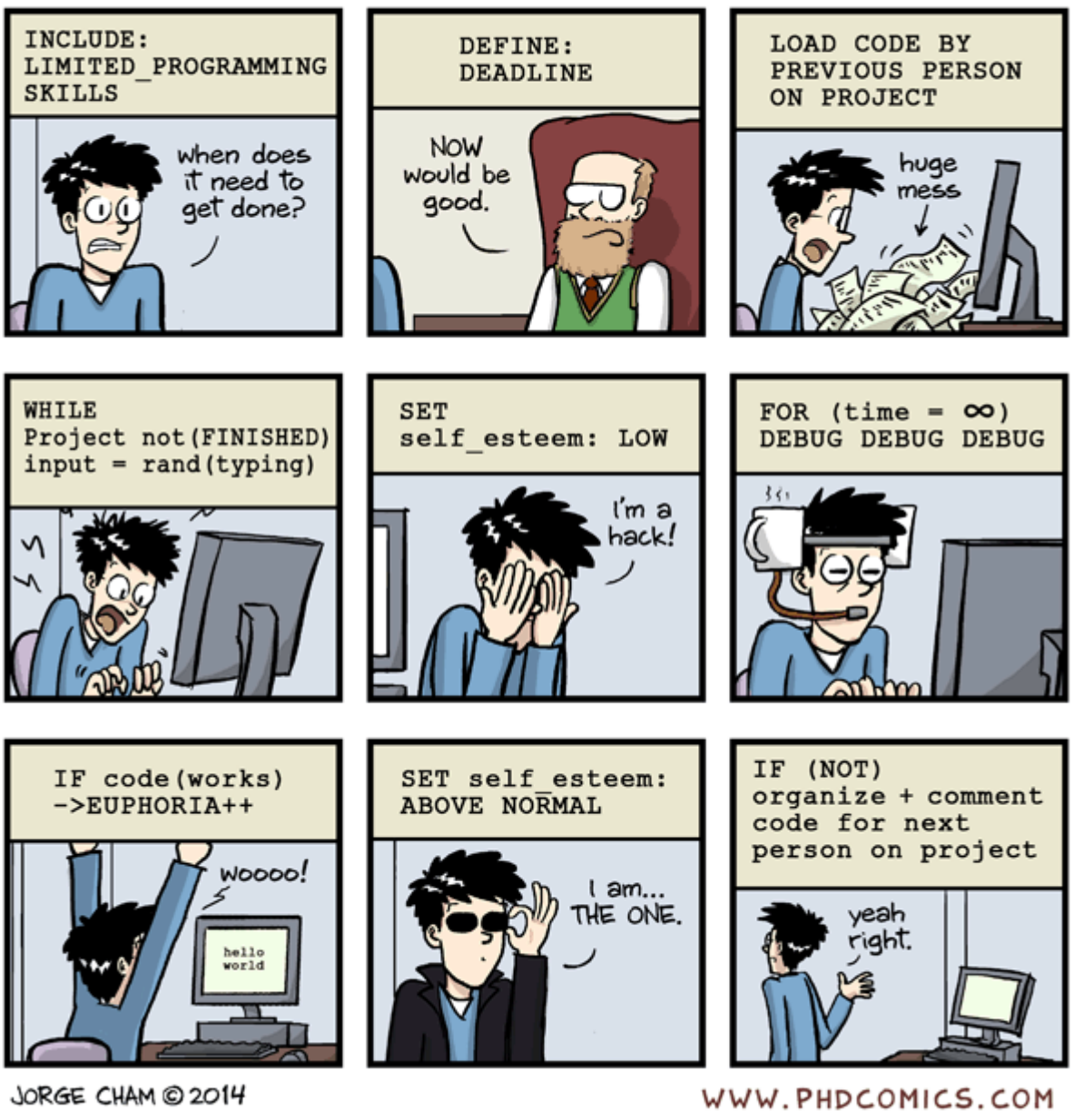
Możemy zapisać nasz program w 3 plikach:

cmplx.h – plik nagłówkowy klasy (tu struktury!)
(definicja klasy complex i deklaracje funkcji
Wspomagających)



cmplx.cpp – plik implementacyjny klasy
(definicje metod klasy complex i definicje
funkcji wspomagających),

main.cpp – główny plik aplikacji
(definicja funkcji main)



KONIEC WYKŁADU 1