Wydział Elektroniki
i Technik Informacyjnych
POLITECHNIKA WARSZAWSKA

Programowanie obiektowe
Wprowadzenie

Krzysztof Gracki

krzysztof.gracki@pw.edu.pl
pok. 312

Politechnika
Warszawska

Sprawy organizacyjne

Punktacja i ocena

- Sprawdziany o łącznej punktacji 0 .. 40
  - 20 maja 2025, (20 pkt.)
  - sprawdziany podczas laboratorium (w sumie 20 pkt.)
- Laboratorium o łącznej punktacji 0 .. 35
- Projekt 25 (zalicza min.7 pkt.)

Ocena końcowa wg skali proporcjonalnej:

- 51..60 → 3
- 61..70 → 3.5
- 71..80 → 4
- 81..90 → 4.5
- 91..100 → 5

Sprawy organizacyjne (2)

Zajęcia laboratoryjne (sala 09, 011) – start (najwcześniej) w drugim tygodniu semestru . Zmiana grup za zgodą prowadzących.

Zadanie projektowe

1. Wstępne, prosta klasa, kompilacja, uruchamianie (5 pkt.)
2. Klasy autonomiczne, typy wbudowane, instrukcje złożone (10 pkt.)
3. Klasy, przeciążanie funkcji i operatorów, biblioteka standardowa, szablony, testy (10 pkt.).
4. Klasy ze zmienną strukturą obiektów, reprezentacja grupowa, dziedziczenie (10 pkt.).

Projekt

Dziedziczenie, funkcje wirtualne, sytuacje wyjątkowe, współpraca ze strumieniami, kontenery i iteratory (25 pkt.) :

Termin zakończenia - tydzień **#14**.

Sprawy organizacyjne (3)

Literatura

B. Stroustrup: Język C++, WNT, 2002, 2008.
B. Stroustrup: Język C++. Kompendium wiedzy, WNT, 2014.
B. Stroustrup: Programowanie: teoria i praktyka z wykorzystaniem C++, Helion, Gliwice 2010.
R. Nowak, A. Pająk: Język C++: mechanizmy, wzorce, biblioteki, BTC, Warszawa 2010.
S.B. Lippman: Podstawy języka C++, WNT 2001, 2003.
Jerzy Grębosz: Opus magnum C++11. Programowanie w języku C++, Helion 2017.
Bruce Eckel: Thinking in C++. Edycja polska, tom1, tom2 Helion 2002, 2004.

Internet

- C++ FAQ LITE: http://www.parashift.com/c++-faq-lite/
- Stack Overflow: https://stackoverflow.com/
- C++ reference: https://en.cppreference.com/w/cpp/language/reference

Narzędzia online

- https://www.onlinegdb.com/online_c++_compiler
- https://cppinsights.io/
- https://compiler-explorer.com

Biblioteki

- Boost libraries: http://www.boost.org/

Sprawy organizacyjne (4)

Zakres wykładu: podstawy programowania obiektowe w C++

- C++ Wprowadzenie - prosty przykład
- Istota podejścia obiektowego
- Ogólna struktura ANSI C++, mechanizmy języka, biblioteka
- Mechanizmy pomocnicze
- Klasy autonomiczne, klasy o zmiennej strukturze, proste szablony
- Dziedziczenie i polimorfizm
- Szablony - rozszerzenie
- Obsługa sytuacji wyjątkowych
- Programowanie rodzajowe, struktura biblioteki standardowej, praca z kontenerami, strumieniami; wsparcie algorytmiczne
- Standard języka (ISO/IEC 14882:2014 ... 14882:2024)

Wydział Elektroniki i Technik Informacyjnych
POLITECHNIKA WARSZAWSKA

Wprowadzenie

Paradygmaty programowania

Programowanie obiektowe

Praca nad projektem

- Klasy i diagramy klas
- Budowa programu
- Narzędzia wspomagające pracę
- Migracja do C++
- Uwagi o stylu programowania

Politechnika Warszawska

Języki programowania

Paradygmat programowania to sposób patrzenia programisty na przepływ sterowania i wykonywanie programu komputerowego

Algorytmy + Struktury Danych = Programy
Niklaus Wirth 1980

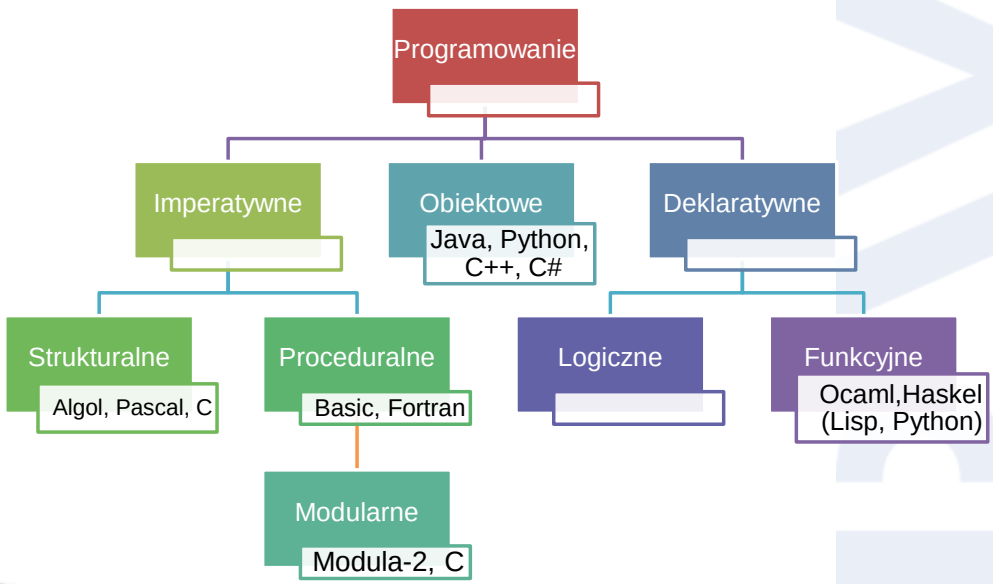
Język programowania to środek do realizacji założeń paradygmatu

Język może:

- wspierać paradygmat (zawiera środki pozwalające wyrażać paradygmat wprost)
- umożliwiać paradygmat (pozwala wyrażać pewien paradygmat, ale kosztem dodatkowej pracy programisty)

język C++ wspiera programowania obiektowe,
język C umożliwia programowania obiektowe

Paradygmaty programowania



Programowanie imperatywne

Podstawowy paradygmat programowania

- Kod jest sekwencją instrukcji
- Program to ciąg rozkazów
- Obliczenia to zmiana stanu maszyny (rejestry, komórki pamięci, ...)

Ścisły związek z architekturą komputera

Politechnika Warszawska

Programowanie proceduralne i modularne

Programowanie proceduralne

- dzielenie kodu na procedury, czyli fragmenty wykonujące ściśle określone operacje

Procedury nie powinny korzystać ze zmiennych globalnych !

Programowanie modularne

- dodatkowy, nadrzędny, poziom „modułów”

Moduł grupuje związane ze sobą dane oraz procedury

Ograniczanie dostępność niektórych danych (patrz **`static`**)

Politechnika Warszawska

Programowanie obiektowe

Program definiowany za pomocą obiektów

Obiekt

- stan (dane)
- zachowanie (metody)

Program to zbiór komunikujących się obiektów

Co nowego?

- dane i procedury są ze sobą ściśle związane
- łatwe tworzenie oprogramowania (wykorzystywanie gotowych obiektów)
- zgodność „z rzeczywistością" – naturalny sposób przetwarzaniu informacji przez mózg

Paradygmat obiektowy

Abstrakcyjne typy danych

- Abstrakcyjny obiekt służy do wykonania określonej pracy (nie ważne jak dokładnie to się dzieje). Może zmieniać swój stan i komunikować się z innymi obiektami

Hermetyzacja - ukrywanie szczegółów (enkapsulacja)

- Obiekt nie może (samowolnie) zmieniać stanu wewnętrznego innych obiektów
- Obiektu komunikuje się poprzez interfejs

Dziedziczenie cech

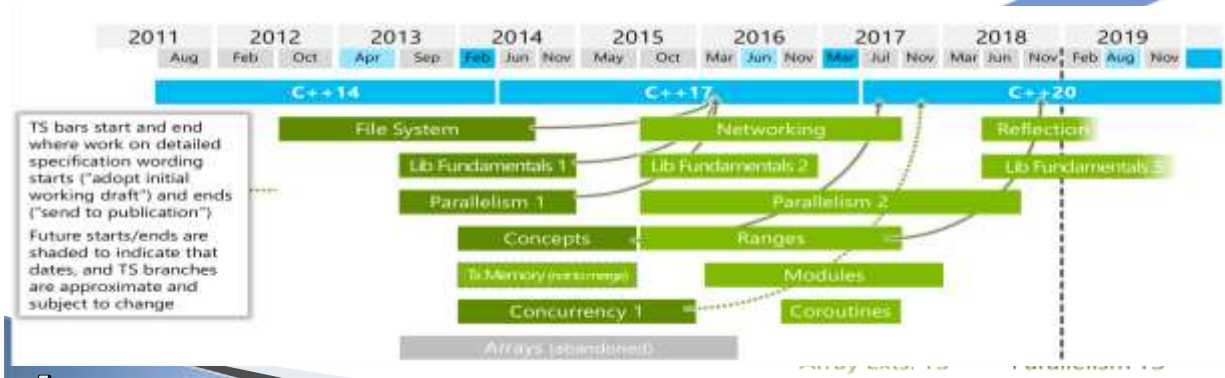
- Na podstawie ogólnych obiektów możemy definiować nowe (specjalizowane) poprzez dodanie nowych funkcjonalności

Polimorfizm (wielopostaciowość)

- Kolekcje obiektów mogą zawierać obiekty różnego typu (o tej samej genealogii), a wywołanie metod następuje dla właściwego obiektu (np. mechanizm tzw. późnego wiązania lub wiązania dynamicznego)

Historia

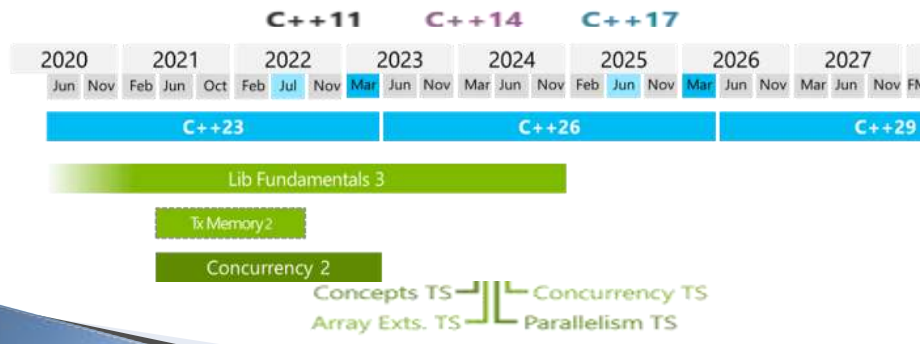
- 1972 – Dennis Ritchie projektuje język C
- Implementacja systemu UNIX
- 1980 – początki prac nad C++ (Bjarne Stroustrup)
- 1989 – standard ANSI C (American National Standards Institute)



http://herbsutter.com/2013/10/03/trip-report-fall-iso-c-standards-meeting/

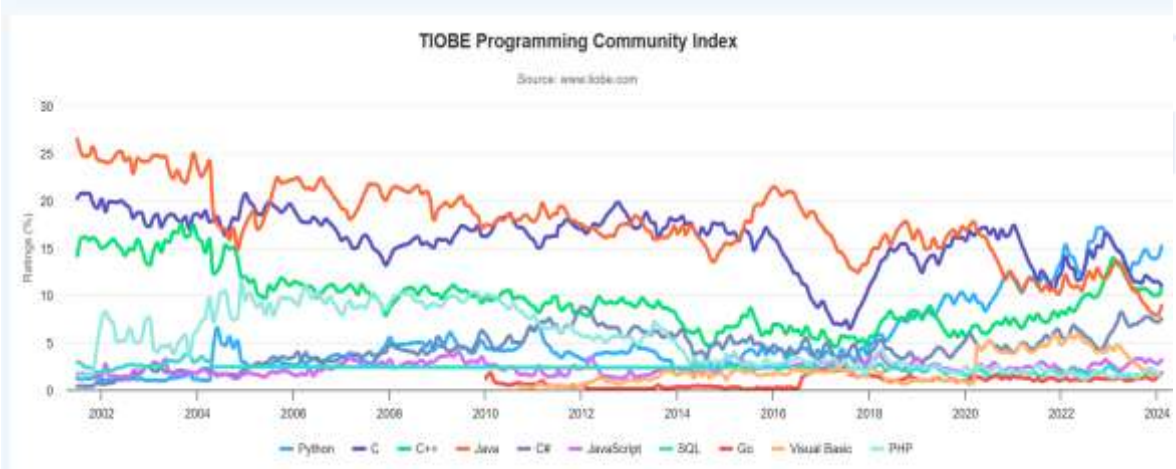
Historia

- 1972 – Dennis Ritchie projektuje język C
- Implementacja systemu UNIX
- 1980 – początki prac nad C++ (Bjarne Stroustrup)
- 1989 – standard ANSI C (American National Standards Institute)



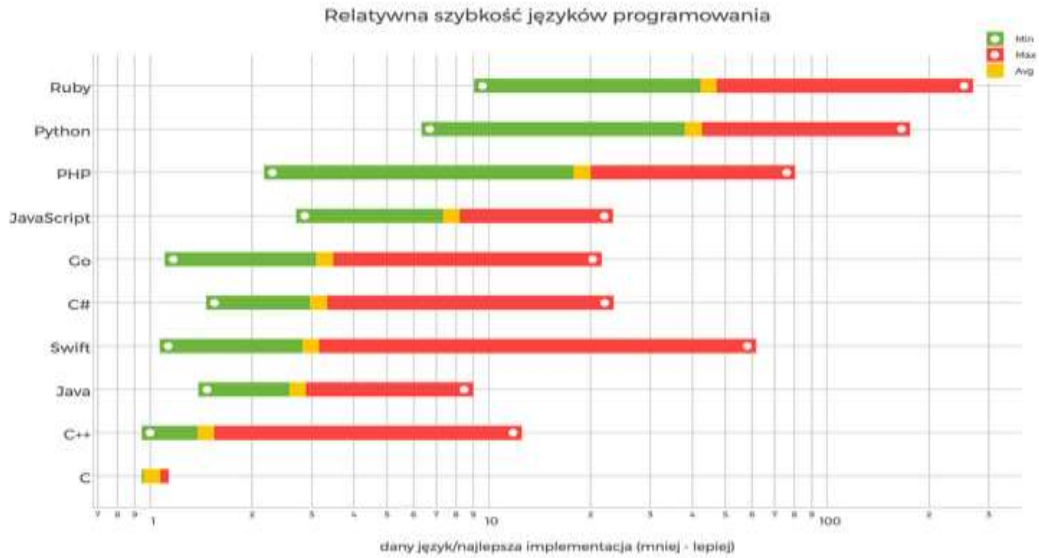
https://isocpp.org/std/status

Popularność języków programowania



https://www.tiobe.com/tiobe-index//

Szybkość języków programowania



https://bulldogjob.pl/news/232-ktory-jezyk-programowania-jest-najszybszy

Zastosowania C++

- Programowanie aplikacji
- Programowanie serwerów
- Gry komputerowe
- Systemy operacyjne

- Zastosowania przemysłowe
- IoT - Internet rzeczy

Co nam daje?
- Wysoką kontrolę nad programem
- Dostęp niskopoziomowy
- Bogata biblioteka standardowa
- Wiele tzw. konceptów programowania

Praca nad projektem

czyli ...
jak przetrwać laboratorium

- Projektowanie aplikacji
- Formalny opis klas
- Przydatne narzędzia
- Styl kodowania

Tworzenie oprogramowania

- Specyfikacja wymagań
- Analiza – zbudowanie modelu dla pojęć i terminów
- Projektowanie – modelowanie rozwiązania, pojęcia niezbędne do implementacji
- Programowanie (implementacja) – budowa modelu działającego na komputerze
- Testowanie
- Integracja, Wdrożenie, Utrzymanie

Budowa diagramu klas

- Identyfikacja klas i ich atrybutów (tzw. encji)
- Usunięcie niepotrzebnych klas i dodanie związków dziedziczenia
- Identyfikacja powiązań między obiektami (asocjacji)
  - określenie liczności
  - identyfikowanie ewentualnych agregacji i kompozycji, itd.

Identyfikacja klas

Klasy wynikające z analizy zadania
(np. Klient, Ubezpieczenie, Szkoda, Odszkodowanie)
Klasy pomocnicze
(np. interfejsu : Okno, Przycisk)

Metody

Karty CRC
(Class Responsibilities and Collaborators)

Nazw klasy	
Odpowiedzialność (obowiązki klasy)	Współpracownicy (klasy pomagające w wypełnianiu obowiązków|)

Metoda rzeczownikowo/czasownikowa

- Identyfikacja klas poprzez wyznaczenie fraz rzeczownikowych
- Redukcja zbioru kandydatów
- Identyfikacja metod

Przygotować symulator **firmy transportowej** którego działanie ma polegać na zaplanowaniu optymalnego przewozu towaru między jej oddziałami. Do przewozu mogą być używane pojazdy przeznaczone do transportu odpowiedniego towaru.
Ponadto proszę przyjąć następujące założenia:
- Firma może składać się z wielu *oddziałów* i połączeń (dróg) między nimi (graf)
- Drogi posiadają pewne ograniczenia dotyczące zarówno dopuszczalnej nośności pojazdów jak i ograniczeń szybkości
- W oddziałach mogą znajdować się pojazdy (różnych typów – służące do przewozu różnych towarów)

UML – Unified Modeling Language

Język modelowania systemów informatycznych (między innymi)
- `70 początki
- 1997 UML 1.0
- 2004 UML 2.0
- UML 2.2

Graficzna reprezentacja problemu

Automatyczne generowanie kodu

http://www.omg.org/spec/

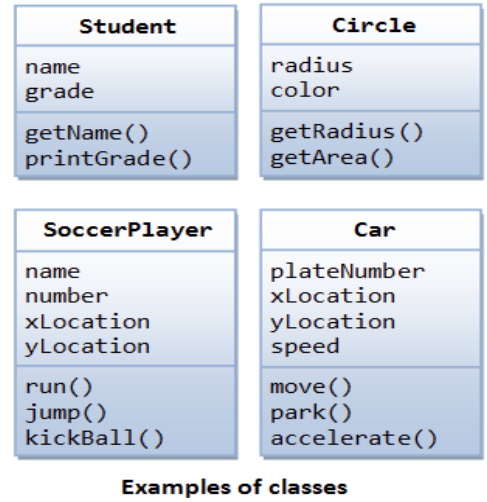
http://www.omg.org/spec/UML/ISO/19501/PDF

Diagramy strukturalne
- **Diagram klas**
- Diagram obiektów
- Diagram struktury złożonej
- Diagram komponentów
- Diagram pakietów
- Diagram rozmieszczenia

Dynamiczne
- **Diagram przypadków użycia**
- Diagram współdziałania
- Diagram stanów
- Diagram działania

UML - specyfikacja klasy

- **Nazwa** – identyfikator
- **Cechy** (właściwości) – umożliwiają przechowywanie informacji o obiekcie
- **Operacje** (metody) – definiują funkcjonalność obiektów



Examples of classes

Klasy - oznaczenie dostępności

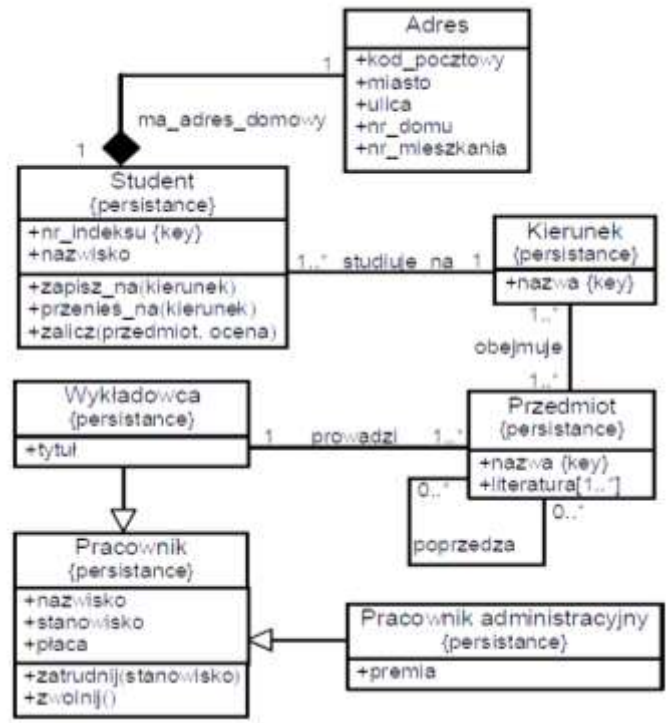
Dostępność właściwości i funkcjonalności

- **+ publiczne** – opisują funkcjonalność klasy
- **– prywatne** – są częścią implementacji klasy
- **# chronione** – są częścią implementacji klasy dostępną w klasach pochodnych

Stos
- size : int //rozmiar stosu - sp : int // wskaźnik stosu
+ push(int) + pop (int) + empty():bool

Powiązania klas

- Powiązanie (linia)
- Agregacja / Kompozycja (romb/ wypełniony)
- Uogólnienie/ Generalizacja (trójkąt)



Programowanie sterowane testami (TDD - Test Driven Development)

- testowanie
- wprowadzanie modyfikacji
- refaktoryzacja kodu

(najlepiej programować parami)

nie powielaj kodu!

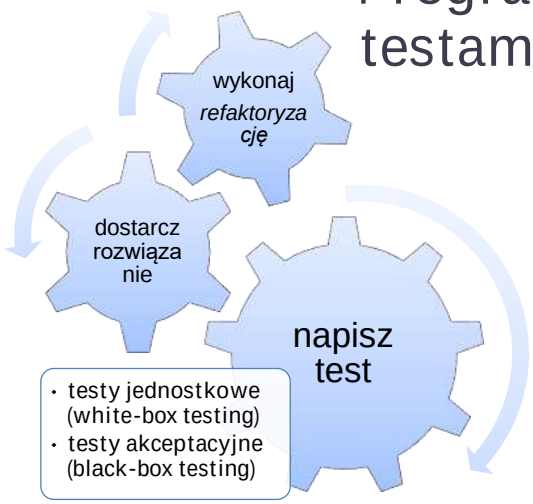
```
class BitSet
{
public:
  bool getBit(unsigned nbit)
        { return false; };
  void setBit(unsigned nbit) {  };
};
 void test(void) {
   BitSet mySet;
   assert(mySet.getBit(10) == false);
   mySet.setBit(10);
   assert(mySet.getBit(10) == true);
 }
```

Programowanie ekstremalne (ang. eXtreme Programming)

- bez projektu
- kolejne wersje programu (iteracje)

Politechnika Warszawska

Programowanie sterowane testami (TDD - Test Driven Development)



- testowanie
- wprowadzanie modyfikacji
- refaktoryzacja kodu

(najlepiej programować parami)

operatory

```
class BitSet
{
public:
  bool& operator[](int i) {
      return …;
  }
};
 void test(void) {
   BitSet mySet;
   assert(mySet[10] == false);
   mySet[10] = true;
   assert(mySet[10] == true);
 }
```

Programowanie ekstremalne (ang. eXtreme Programming)

- bez projektu
- kolejne wersje programu (iteracje)

Politechnika Warszawska

Programowanie sterowane testami (TDD - Test Driven Development)

- testowanie
- wprowadzanie modyfikacji
- refaktoryzacja kodu

(najlepiej programować parami)

biblioteka standardowa

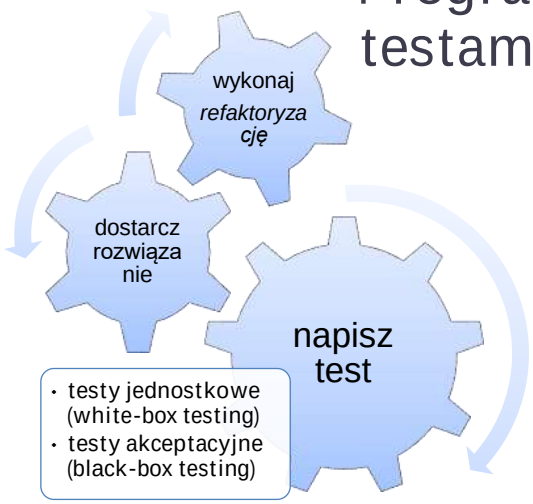
```
class BitSet
{std::array<bool,32> btbl;
public:
  bool& operator[](int i) {
      return btbl[i];
  }
};
 void test(void) {
   BitSet mySet;
   assert(mySet[10] == false);
   mySet[10] = true;
   assert(mySet[10] == true);
 }
```

Programowanie ekstremalne (ang. eXtreme Programming)

- bez projektu
- kolejne wersje programu (iteracje)

Politechnika Warszawska

Programowanie sterowane testami (TDD - Test Driven Development)



- testowanie
- wprowadzanie modyfikacji
- refaktoryzacja kodu

(najlepiej programować parami)

język C

```
typedef unsigned long BitSet;

void set_bit(BitSet *b, unsigned i)
   {};

int get_bit(BitSet *b, unsigned i)
   { return 0; };

 void test(void) {
   BitSet b;
   set_bit(&b, 10);
   assert(get_bit(&b, 10) == 1);
 }
```

Programowanie ekstremalne (ang. eXtreme Programming)

- bez projektu
- kolejne wersje programu (iteracje)

Politechnika Warszawska

Zasady tworzenia lepszego kodu

KISS
(Keep It Simple, Stupid; Keep it Short and Simple)
- zachowaj prostotę, głupku

DRY
(*Don't* Repeat Yourself)
- nie powtarzaj się

YAGNI
(You *Aren't* Gonna Need It)
- nie, nie będzie to potrzebne

Ukrywanie informacji, optymalizacja/czytelność kodu, poprawianie/zaznaczanie błędów, ...

Wydział Elektroniki i Technik Informacyjnych
POLITECHNIKA WARSZAWSKA

Narzędzia programisty

- środowiska programistyczne
- system kontroli wersji
- program make

Politechnika Warszawska

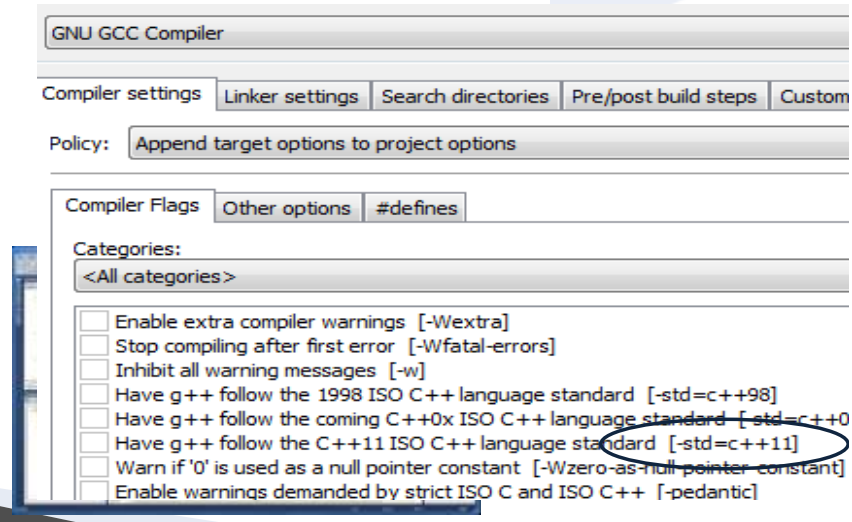
Środowiska programistyczne

Windows

- Microsoft Visual Studio ...19/22 Express/Prof./Prem./Ultimate
- VSCode, Dev-C++
- NetBeans, Borland
- Eclipse CDT
- Code::Blocks
- cLion, Watcom, ...

Linux

- Code::Blocks
- KDevelop
- NetBeans
- Eclipse CDT



Politechnika Warszawska

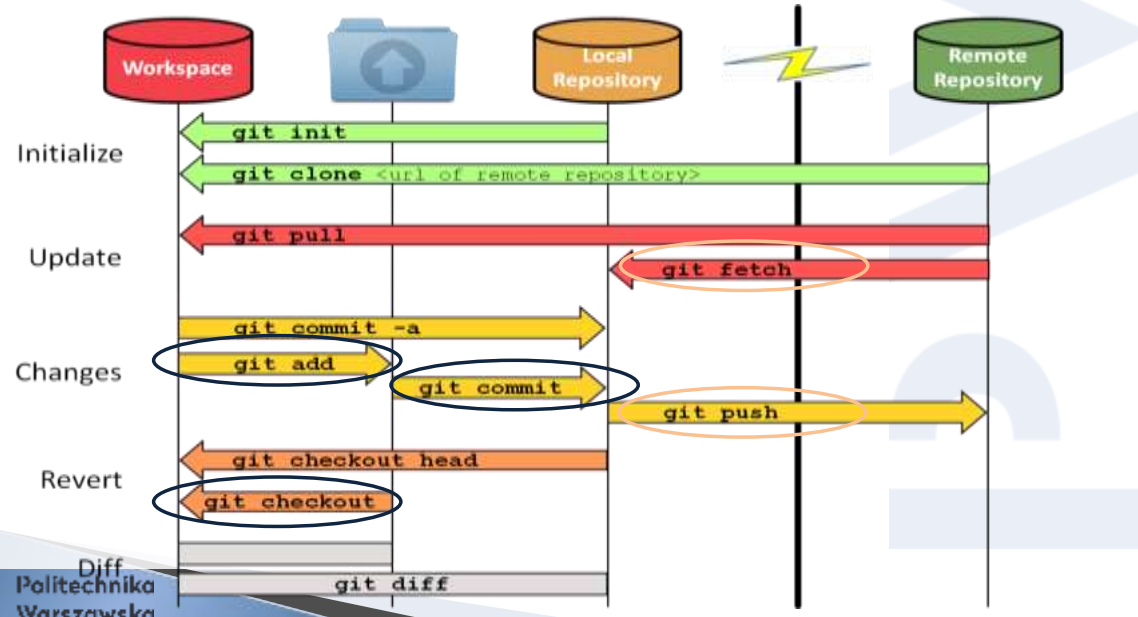
System kontroli wersji

- umożliwia śledzenie zmian w plikach
- powrót do dowolnej wersji pliku czy stanu projektu
- synchronizację danych przy praca z różnych lokalizacji

http://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

Politechnika Warszawska

System kontroli wersji (2)



Program make i makefile

program **make** wykonuje komendy shell'a domyślnie zawarte w pliku **makefile**

```
#prosty makefile z jawnymi regułami
CC = gcc #nazwa kompilatora
main.exe : main.o foo.o
	$(CC) main.o foo.o -o main.exe

main.o : main.c
	$(CC) -c main.c

foo.o : foo.c
	$(CC) -c foo.c
```

Makro

Komentarz

Cel

Pliki źródłowe do realizacji celu

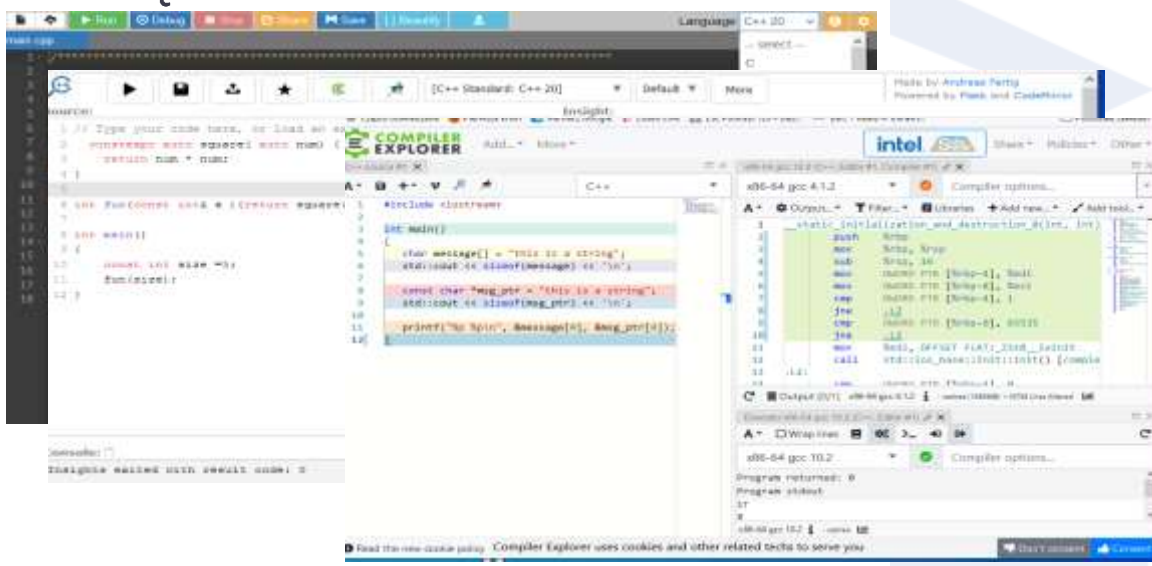
Odwołanie

Instrukcja do wykonania

Cel

Pliki źródłowe do realizacji celu

Narzędzia online



https://www.onlinegdb.com/online_c++_compiler#
https://cppinsights.io/
https://godbolt.org/

Politechnika Warszawska