

C++ wstęp

Wykład 1

Kontakt

Dr inż. Bartosz Mindur

D11 p. 127 (Kawiory 26A)

Tel. (617)-47-25

email: bartosz.mindur@agh.edu.pl

Temat email: [CPP] ...

„Reguły gry”

- Wszelkie informacje organizacyjne znajdują się na stronie
 - <http://fatcat.ftj.agh.edu.pl/~mindur/oop/>
 - Dostęp: użytkownik **oop**, hasło **cpp2013**
- Najważniejsze uwagi
- Sposób oceniania jest zgodny z regulaminem studiów.
 - Do egzaminu można przystąpić **wyłącznie** po uzyskaniu zaliczenia z laboratorium
 - Ocena końcowa z przedmiotu jest średnią arytmetyczną wszystkich ocen z egzaminu(-ów) oraz ćwiczeń

„Reguły gry” cd.

- Najważniejsze uwagi dotyczące laboratorium
 - Obecność na laboratorium jest obowiązkowa
 - Każdą nieobecność należy usprawiedliwiać od razu na **następnych** zajęciach, na których jest się obecnym
 - Dopuszczalne są dwie nieobecności nieusprawiedliwione (za 0 pkt. z danych zajęć)
 - Na każdych zajęciach należy być przygotowanym ze **wszystkich** poprzednich wykładów oraz ćwiczeń
 - Zadania należy wykonywać samodzielnie bez żadnych pomocy
 - Każde zajęcia mają taką samą wagę (10 pkt.).
 - Dokumentacja - 1 pkt.
 - Poprawna kompilacja (bez ostrzeżeń) - 2 pkt.
 - Poprawne wykonanie (odpowiednie wyjście, brak wycieków pamięci itp.) - 3 pkt.
 - Ocena kodu (pod względem projektowym, zgodności z OOP, przejrzystości oraz sposobu implantacji) - 4 pkt.
 - **Jest jeden terminów poprawkowy**
 - Przysługuje osobom którym poprawienie max. 2 zajęć pozwala na otrzymanie oceny pozytywnej.

Literatura

- Jerzy Grębosz, *Symfonia C++ Standard*, Oficyna Kallimach, Kraków, 2006 (Wydanie II)
- Jerzy Grębosz, *Pasja C++*, Oficyna Kallimach, Kraków, 2003 (Wydanie III)
- Bjarne Stroustrup, *The C++ Programming Language*, Addison-Wesley, 1997
- Bruce Eckel, *Thinking in C++, 2nd ed. Volume 1 i 2*, <http://www.BruceEckel.com>
- Nicolai Josuttis, *C++ Programowanie Zorientowane Obiektowo*, Helion, 2003
- Nicolai Josuttis, *C++ Bibliotek Standardowa*, Helion, 2003
- Bjarne Stroustrup, *Programowanie. Teoria i praktyka z wykorzystaniem C++*, Helion, 2010
- „Internet”

Język programowania

- **Język programowania** składa się z notacji i reguł, według których pisze się programy
- **Składnia** języka programowania określa jakie kombinacje wybranych symboli, słów kluczowych są dopuszczalne. Formalna składnia zawiera systematyczne warianty kilku struktur sterujących, sposoby definiowania rozmaitych struktur danych i wzorce podstawowych instrukcji. Komputer nie wykona żadnej instrukcji, nawet najbardziej jasnej i jednoznacznej, jeśli nie ma jej wśród tych, które dopuszcza dany język programowania

(składnia to dział językoznawstwa badający zasady, na jakich wyrazy łączone są w dłuższe wypowiedzi, zwł. zdania i ich równoważniki. Zajmuje się relacjami między poszczególnymi elementami składowymi zdań, a także relacjami międzyzdaniowymi w obrębie dłuższego tekstu)

- **Semantyka** języka programowania określa znaczenie każdego wyrażenia dopuszczalnego składniowo

(semantyka to dyscyplina badająca relacje pomiędzy znakami a przedmiotami, do których się one odnoszą. Semantyka zajmuje się badaniem znaczenia słów, czyli interpretacją znaków oraz interpretacją zdań i wyrażeń języka)

Przegląd technik programowania

- Programowanie liniowe
 - Dane globalne
 - Dużo instrukcji **goto** (skoków bezpośrednich)
- Programowanie proceduralne
 - Wykorzystanie funkcji
- Programowanie z ukrywaniem danych
 - Struktury
- Programowanie obiektowe (object based design)
 - Używane są obiekty, ale są sobie obce
 - Różne funkcje dla różnych danych
- Programowanie Obiektowo (Z)Orientowane (OO)
 - Zależności między klasami

Droga do programowanie orientowanego obiektowego

- Wszystkie programy zapewniają pewien poziom abstrakcji
 - Z najprostszym przypadkiem mamy do czynienia, gdy mówimy o assemblerach, zapewniają one minimalny stopień abstrakcji
 - Następne w hierarchii są języki takie jak **Fortran**, **BASIC**, **C**, które pozwalają na znacznie większą abstrakcję jednak dalej wymagają konstruowania programów z myślą o sposobie ich wykonywania przez komputer, a nie w sposób jaki powinny rozwiązywać zadany problem
 - Najwyżej w tej hierarchii stoja języki pozwalające na opisywanie problemu w sposób najbardziej intuicyjny, tzn. wymagający patrzenie od strony problemu a nie maszyny. Takie języki (np. **C++**, **Java**) są zorientowane obiektowo, skąd nazwa **Programowanie Obiektowe Orientowane**

Język C++



- Autorem języka jest **Bjarne Stroustrup**.
- Link do strony domowej autora
 - <http://www.research.att.com/~bs/homepage.html>

C++ historia

- C++ został zaprojektowany jako obiektowa wersja C
 - Konsekwencją tego jest fakt, iż nie jest językiem czysto obiektowym, ale hybrydą
- W 1998 komitet ANSI/ISO przyjął standard języka C++
- W 2003 przyjęto poprawioną wersję powyższego standardu
- C++11 - standard 12-08-2011
 - <http://en.wikipedia.org/wiki/C%2B%2B11>

Wybrane oprogramowanie napisane w C++

- Adobe Systems
- Maya
 - Star Wars Episode I, Spider-Man, Lord of the Rings
- Amazon.com
- Autodesk
- Geant4
 - HEP - symulacja zderzeń cząstek
- Google
- Microsoft
- Nullsoft WinAmp
- Sun OpenOffice
- Games
 - Doom III engine, Diablo, Warcraft
- KDE
- SETI@home
- ...

Cechy języka C++

- C++ jako język programowania obiektowego.
Do jego cech należą:
 - Abstrakcyjne typy danych (klasy)
 - Hermetyzację danych
 - Dziedziczenie
 - Polimorfizm
 - Programowanie uogólnione (szablony)
 - Przestrzenie nazw

Obiekty

■ Obiekty

- ❑ Istotne jest zachowanie obiektu, a nie jego wewnętrzna struktura
- ❑ Atrybuty charakteryzują dany obiekt
- ❑ Reprezentują realny byt (np. samochód) lub są całkowicie abstrakcyjne (pojazd)
- ❑ Struktury pozwalają na rozkład obiektu na poszczególne atrybuty (wbudowane typy danych)

Abstrakcyjne typy danych

- Struktury nie umożliwiają opisu zachowania obiektu
- Zachowanie obiektu reprezentowane jest przez interfejs
- Abstrakcyjne typy danych umożliwiają łączenie powyższych dwóch cech
 - Klasy

Klasy

- Klasa stanowi implementację abstrakcyjnego typu danych
 - Zawiera atrybuty (dane) oraz opisuje zachowanie (metody)
 - Z punkty widzenia programowania klasa stanowi także typ
 - Instancje stanowią realizację obiektów opisywanych przez daną klasę
 - Przykład obiektu?

Hermetyzacja danych - separacja interfejsu i implementacji

- Tylko osoba tworząca daną klasę musi zaimplementować jej składowe, funkcjonalności oraz powiązania
- Wykorzystywanie klas podlega regułom jakie stworzył programista tworzący dany obiekt. Używamy tylko udostępnionych metod i składowych
- Wewnętrzna struktura jest ukryta, dzięki czemu nie trzeba się martwić jak dane zadanie jest realizowane (pilot do telewizora)
- Nie wszystkie składowe i metody muszą być upubliczniane. Często większość z nich jest ukryta i służy tylko lepszemu (prostsze) wykonywaniu danego zadania
- Ukryta implementacja niesie ze sobą możliwość zmieniania wewnętrznej struktury np. w celu przyspieszenia działania lecz nie powoduje zmiany sposobu widzenia obiektu z zewnątrz
- W języku C++ do tego celu używa się słów kluczowych **public**, **protected** oraz **private**. Odpowiednio pierwsze umożliwia publiczny dostęp do składowych i metod, drugie tylko dla obiektów dziedziczących z danej klasy, a trzecie brak dostępu

Wielokrotne używanie kodu

- Zbudowany obiekt można wielokrotnie wykorzystywać np. do tworzenia innych obiektów. W szczególności jeśli mamy do czynienia ze skomplikowanymi obiektami, które już zostały stworzone i poddane procesowi testowania oraz weryfikacji
- Tworzenie nowych obiektów z innych nazywamy **agregacją**. Budowanie w ten sposób klas daje nam dużą elastyczność
- Składowe klasy są zwykle prywatne, dzięki czemu jakakolwiek zmiana w ich implementacji nie wpływa na późniejsze ich wykorzystanie przez innych
 - Ewentualnie chronione
- Możliwe jest także ponowne wykorzystanie już gotowych klas w dziedziczeniu

Dziedziczenie

- Tworzenie klas potomnych na podstawie klas bazowych o bardziej ogólnych cechach
- Dodawanie funkcjonalności w klasach pochodnych lub/i ich zmienianie na bardziej szczegółowe
- Zalety
 - Bardziej spójny kod o mniejszych rozmiarach
 - Zastosowanie polimorfizmu
- Przykład?

Wyjątki - obsługa błędów

- Obsługa błędów w wielu językach programowania jest ignorowana
 - Bardzo niedobre podejście
- W C++ mamy różne możliwości obsługi błędów
 - Znana z C metoda jest zwracanie kodu błędu
 - Metoda niezalecana
 - Obsługa wyjątków
 - Wyjątek jest obiektem rzucanym z miejsca wystąpienia błędu, a następnie obiekt ten jest łapany i obsługiwany przez odpowiednio przygotowany fragment kodu
 - Obecnie najczęściej stosowana metoda obsługi błędów, ale nie tylko

Szablony

- W C++ istnieje ścisła kontrola typów danych
 - Zaleta - kontrola podczas kompilacji
 - Wada - ograniczenia
- W celu uniknięcia powyższej niedogodności wprowadzono szablony
 - Może zostać zaimplementowany dla dowolnego typu
 - W momencie kompilacji generowany jest odpowiedni kod
 - Szablon jest tak naprawdę pseudokodem
 - Zastosowanie szablonów skraca czas tworzenia programu oraz zmniejsza ryzyko popełnienia błędów

Przestrzenie nazw

- Symbole i konstrukcje występujące w programie mogą być grupowane za pomocą przestrzeni nazw
- Przykładowo wszystkie symbole zdefiniowane w standardowej bibliotece są zdefiniowane w przestrzeni nazw `std`

Kompilatory języka C++

■ Darmowe

- ❑ Borland C++
http://www.borland.com/news/press_releases/2000/02_16_00_bcp_compiler.html
- ❑ Cygwin (GNU C++) <http://www.cygwin.com/>
- ❑ DJGPP <http://www.delorie.com/djgpp/>
- ❑ Bloodshed Dev-C++ <http://www.bloodshed.net/index.html>
- ❑ Intel C++ for Linux
<http://www.intel.com/software/products/compilers/clin/noncom.htm>
- ❑ gcc <http://gcc.gnu.org/>

Kompilatory języka C++ ...

■ Komercyjne

- ❑ Borland C++Builder <http://www.borland.com/cbuilder/>
- ❑ Comeau C++ <http://www.comeaucomputing.com/>
- ❑ Intel C++ <http://www.intel.com/software/products/compilers/>
- ❑ Microsoft Visual C++ <http://msdn.microsoft.com/visualc/>
- ❑ Compaq C++
<http://h18000.www1.hp.com/products/software/compilers/canxdcxx.html>

Narzędzia dodatkowe

- doxygen
 - generator dokumentacji dla języków C++, C, Java, ...
- make
 - program powłoki systemowej automatyzujący proces kompilacji programów
- cmake
 - program do generacji plików make lub projektów innych środowisk
 - działa na wielu platformach
- gdb
 - debugger będący częścią projektu GNU
 - Konsolowy
 - Dostępne są graficzne nakładki
- valgrind
 - Wykrywanie wycieków pamięci
 - Profilowanie aplikacji