C++ wstęp

Wykład 1

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"

Outdated

Internet

- http://www.stroustrup.com/
- https://isocpp.org/std
- https://www.cprogramming.com/
- https://en.cppreference.com/w/
- http://www.cplusplus.com/
- https://www.reddit.com/r/cpp/
- https://stackoverflow.com/questions/tagged/c%2b%2b
- https://twitter.com/isocpp
- https://www.boost.org/doc/libs/

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)

Technik programowania

- Programowanie maszynowe
  - W języku maszynowym bezpośrednio
  - W asemblerze (mnemoniki)
- 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

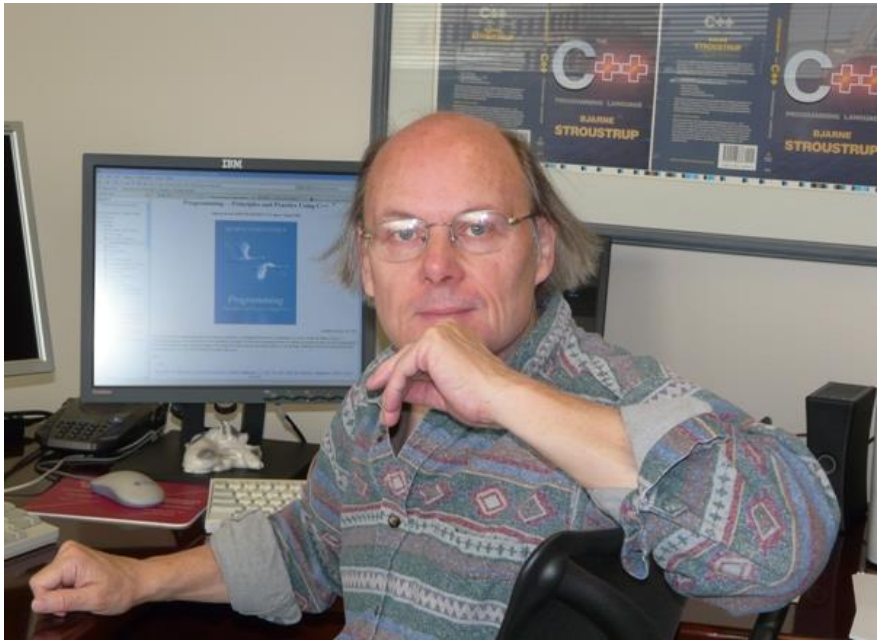
Technik programowania

- Programowanie uogólnione
  - Metapogramowanie
  - Pozwala na pisanie kodu programu bez wcześniejszej znajomości typów danych
- Programowanie funkcyjne
  - Forma programowania deklaratywnego
  - Brak stanu (zmienny lokalnych i globalnych), dużo wykorzystania referencji
- Programowanie aspektowe
  - Separację zagadnień i rozdzielenie programu na niezwiązane części
- ...

Droga do programowanie orientowanego obiektowego

- Wszystkie programy zapewniają pewien poziom abstrakcji
  - Z najprostszym przypadkiem mamy do czynienia, gdy mówimy o asemblerach, 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
  - (Prawie) najwyżej w tej hierarchii stoją języki pozwalające na opisywanie problemu w sposób najbardziej intuicyjny, tzn. wymagający patrzenie od strony problemu a nie maszyny. Takie języki (np. C++, Java) są zorientowanie obiektowo, skąd nazwa **Programowanie Obiektowe Orientowane**

Język C++



- Autorem języka jest **Bjarne Stroustrup**
- Link do strony domowej autora
  - http://www.stroustrup.com/

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
- C++14 – standard 15-12-2014
  - http://en.wikipedia.org/wiki/C%2B%2B14
- C++17 (C++1z) – standard 12-2017
  - https://en.wikipedia.org/wiki/C%2B%2B17
- C++20 – nazwa nieformalna
  - https://en.wikipedia.org/wiki/C%2B%2B20

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 charakteryzują
  - Abstrakcyjne typy danych (klasy)
  - Hermetyzację danych
  - Przestrzenie nazw
  - Dziedziczenie
  - Polimorfizm
  - Programowanie uogólnione (szablony)

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
  - Zawierają tylko składniki – czyli dane
- Zachowanie obiektu reprezentowane jest przez interfejs
  - Sam interfejs nie zawiera żadnych danych
- Abstrakcyjne typy danych umożliwiają łączenie powyższych dwóch cech
  - Klasy
  - Klasy abstrakcyjne
  - Klasy czysto abstrakcyjne

Klasy

- Klasa stanowi implementacje abstrakcyjnego typu danych
  - Zawiera atrybuty (dane) oraz opisuje zachowanie (metody)
  - Z punkty widzenia programowania klasa stanowi także typ
    - Ogromne znaczenie w silnie typowanych językach programowania
  - Instancje stanowią realizację obiektów opisywanych przez daną klasę
    - Typ jest jeden, instancji może być wiele
  - 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
- Nie wszystkie składowe i metody muszą być upubliczniane
  - Często większość z nich jest ukryta i służy tylko lepszemu (prostszemu) 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 (czyli interfejsu)
- W języku C++ do tek celu używa się słów kluczowych
  - `public`
  - `protected`
  - `private`

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 jeśli klasa jest przeznaczona do dziedziczenia
- Ponowne wykorzystanie już gotowych klas w dziedziczeniu
  - Inna forma niż klasyczne zawieranie

Dziedziczenie

- Tworzenie klas potomnych na podstawie klas bazowych o bardziej ogólnych cechach
  - Często nawet abstrakcyjnych
- 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
  - Znaną z C metodą 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
  - gcc http://gcc.gnu.org/
  - clang https://clang.llvm.org/
  - Cygwin (GNU C++) http://www.cygwin.com/
- Komercyjne
  - Intel C++ https://software.intel.com/en-us/c-compilers
  - Microsoft Visual C++ http://msdn.microsoft.com/visualc/
- Inne (stare)
  - DJGPP http://www.delorie.com/djgpp/
  - Bloodshed Dev-C++ http://www.bloodshed.net/index.html
  - Comeau C++ http://www.comeaucomputing.com/
  - Borland C++Builder