



Uniwersytet
Wrocławski

C++: *first things first*

Zbigniew Koza
Wydział Fizyki i Astronomii

Wrocław

WSTĘP

Dlaczego C++?

- Jest używany powszechnie, w tym przez zawodowców
- Wspiera różne paradygmaty programowania: proceduralne, obiektowe, generyczne
- C/C++ to podstawa wielu technologii (np. OpenCV, OpenMP, OpenCL,...)
- Bogate, dojrzałe środowisko narzędziowe
- Niewiarygodne bogactwo bibliotek *Open Source*
- Wolny standard
<http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2013/n3690.pdf>
- C/C++ to kanon wykształcenia informatycznego

C++ jako metajęzyk programowania

- Formalnie: C++ to **język programowania**
- Standard języka jest **ubogi**: nie zawiera żadnych udogodnień dla obsługi grafiki, połączeń sieciowych, nowoczesnych urządzeń we/wy itp.
- Standard języka ukierunkowany jest na ułatwienie tworzenia i używania bibliotek
- Większość użytecznego kodu w C++ to kod bibliotek

C/C++

jako assembler wysokiego poziomu

- Standard C++ zakłada, że programy w C++:
 - Są bezkompromisowo szybkie
 - Mają bezpośredni dostęp do sprzętu
 - Mogą być wykonywane w czasie rzeczywistym z gwarancją zakończenia operacji w określonym czasie

C++ jako „lepsze C”

- C++ nie jest ani lepszy, ani gorszy od C
- C++ jest zgodny na poziomie kodu źródłowego z językiem C: dzięki temu C++ ma dostęp do tysięcy bibliotek napisanych w C
- Programy/biblioteki napisane w C można kompilować kompilatorem C++
- Jednak mając do dyspozycji kompilator C++, programy pisze się inaczej niż w C
(„*unlearn C to become a C++ programmer*”)

C++

jako profesjonalne narzędzie tworzenia oprogramowania

- Standard języka koncentruje się na ułatwianiu:
 - tworzenia wydajnego kodu za pomocą konstrukcji wysokiego poziomu („daleko od sprzętu”)
 - wielokrotnego wykorzystywaniu tego samego kodu
 - tworzenia bezpiecznego kodu
 - tworzenia dużych/ogromnych bibliotek/programów

Nauka C++

- Nauka C++ **jest trudna:**
 - Wysoki próg na wejściu
 - Niezwykłe rozbudowany język
 - Brak bezpośredniego wsparcia dla obsługi grafiki
 - Skoncentrowanie uwagi na potrzebach zawodowych programistów/biznesu

3 główne wersje C++

- C++ sprzed 1998 – (pre)historia
- C++98 – wciąż aktywnie używany standard (domyślny dla kompilatora g++)
- C++11/C++14 – aktualny standard

3 książki Stroustrupa

- Każdej nowej wersji języka towarzyszy ponad 1000-stronnicowy podręcznik jego autora, B. Stroustrupa.
- Przyrost wiedzy jest rudny do ogarnięcia.

Ile czasu potrzeba, by nauczyć się C++?

- 48 godzin?
- 1 semestr?
- 2 lata?
- 10 lat?

Nie znam nikogo,
kto zna „całe C++”

Co to jest C++?

- C++ to:
 - składnia języka
 - sposoby używania języka („idiomy”, „techniki programistyczne”, „paradygmaty programowania”)
 - infrastruktura programistyczna
(narzędzia do tworzenia oprogramowania, jego testowania, diagnostyki, kontroli wersji, dystrybucji,...)
 - grube tysiące bibliotek

Tak szeroko rozumianego C++ nie zna nikt

5 poziomów C/C++

1. Programy pisane dla siebie w „czystym C++” (rebusy)
2. + biblioteka standardowa C++ (super rebusy)
3. + biblioteki zewnętrzne
4. Programy pisane dla kogoś i/lub z kimś (np. aplikacje komercyjne)
5. Biblioteki pisane dla innych programistów

5 poziomów C/C++

- Podręczniki C/C++ koncentrują się na poziomach 1-2
- Pracodawcy oczekują poziomu 3 lub 4

Problem skali

- Typowy studencki program w C++ to 100-1000 wierszy kodu umieszczonego w jednym pliku
- Typowy profesjonalny program/biblioteka C++ to grube tysiące a nawet miliony wierszy kodu
- C++ skupia się na udogodnieniu tworzenia DUŻEGO, ZŁOŻONEGO oprogramowania (poziomy 4-5)
- Dlatego studentom (poziom 1-2) trudno zrozumieć sens wielu (większości?) konstrukcji języka C++

Przykład

- OpenFOAM:
 - 15 046 plików nagłówkowych
 - 9 455 plików źródłowych
 - $2\,939\,075 + 3\,572\,055 \approx$ **6,5 mln wierszy kodu**
 - Powyższe rachunki nie uwzględniają biblioteki standardowej C++ (0,5 miliona wierszy kodu) i innych bibliotek, np. Open MPI

Przykład 2: biblioteki standardowe

- Biblioteka standardowa C
 - 170 plików
 - 67 tys. Wierszy kodu
- Biblioteka standardowa C++
 - 475 plików
 - 0.24 mln wierszy kodu
- Biblioteka boost (quasi standard C++):
 - 10 582 plików
 - 2 mln wierszy kodu
 - „Każdy powinien ją mieć”

Jak się uczyć C++?

- Nie zaczynaj rozpoczynać od lektury całych podręczników języka (tak się uczyłem Pascala, C i FORTRAN-a): C++ jest zbyt bogaty, bez praktyki nic z tych książek nie zrozumiesz
- Uwaga, uwaga!
Żeby programować w C++ nie trzeba znać 100% języka ani nawet 50%, ani nawet 10%
- Zalecana metoda: używaj tych konstrukcji języka, które znasz i stopniowo wzbogacaj swój repertuar

Źródła wiedzy

- <http://stackoverflow.com/>
- <http://www.cplusplus.com/>
- Dokumentacja używanych bibliotek/narzędzi
- B. Stroustrup: [Język C++. Kompendium wiedzy](#)
- Z. Koza, [Język C++. Pierwsze starcie](#)
- J. Grębosz, [Symfonia C++](#)

Ostrzeżenie

- Nie kupuj książek napisanych przed 2011 r.
(tzn. nie opisujących standardu C++11)
- Nie czytaj książek opisujących
C++ sprzed 1998 r.

Twój cel (bliski)

- Twój cel na ten semestr: zapoznać się z wybranymi, najważniejszymi pojęciami związanymi z programowaniem w C++
 - Składnia
 - Sposoby używania języka
- Powinieneś osiągnąć poziom kompetencji, który umożliwi Ci samodzielne czytanie dokumentacji (w razie potrzeby)

Twój cel (daleki)

- Osiągnąć **w ciągu 3 lat** poziom kompetencji w C++, który uczyni z Ciebie obiekt pożądania łowców głów dla firm
- Co nie znaczy, że musisz dać się złowić
- Ale warto mieć poczucie własnej wartości...

Ostrzeżenie

- Nikt nie nauczył się grać (dobrze) w szachy z samych książek, bez gry;
- Nikt nie osiągnął mistrzostwa w szachach na podstawie samej gry, bez studiowania gry innych, bez książek i bez trenerów.
- Czyli: programuj, czytaj, programuj, czytaj,...

Wnioski

- Jak najwięcej programuj
- Staraj się pisać różne programy, wymagające od Ciebie użycia różnych strategii programistycznych
- Czytaj o tym, co robisz – bez teorii daleko nie zajdziesz
- Ucz się od najlepszych

- C++
- Information
 - Tutorials
 - Reference
 - Articles
 - Forum

Welcome to **cplusplus.com**

C++ w akcji

Reference

Description of the most important classes, functions and objects of the Standard Language Library, with

Learn to
advanc

- C
- t
- l
- p
- e
- r

User-co
categor

Wnioski

- Funkcjonalny, nietrywialny program w C++ (przeglądarka WWW) można napisać w kilkadziesiąt minut, pisząc niecałe 100 wierszy kodu
- Warunek: użycie właściwej biblioteki
- Nauka tej biblioteki może być nawet bardziej pracowita niż nauka samego C++

Wnioski (2)

- Na pewno zauważył(a/e)ś, że kod programu zawierał elementy **składni języka**, których dotąd mógł(a/e)ś nie widzieć w programach, np:
 - Operator ->
 - Operator ::
 - Słowa kluczowe `class`, `this`
 - Makra preprocesora `#if`, `#else`, `#endif`
- Znaczenia tych elementów trzeba się będzie nauczyć

Wnioski (3)

- Na pewno zauważył(a/e)ś, że ogromną rolę w tworzeniu aplikacji ma nie tylko język programowania, ale i
 - **środowisko programistyczne** (tu: QtCreator)
 - **dokumentacja**
- Musisz nauczyć się nie tylko języka, ale także jego środowiska (dotyczy to każdego języka programowania)

Wnioski (4)

- Prawdopodobnie **nie** zauważył(a/e)ś jeszcze wielu innych cech programu, np. jak on w ogóle działa czy jak on naprawdę się kompiluje, które są równie ważne jak składnia języka, a które omówię później