Programming for fun and profit

Bjarne Stroustrup
Texas A&M University
http://www.research.att.com/~bs
Overview

• What I like about being a computer scientist
• Programming language aims and ideals
• C++
• C++0x
Background

• Århus:
  – childhood

• University of Århus:
  • MS in Math with Computer Science

• Cambridge University:
  • PhD in Computer Science
Career

• Bell Labs Computer Science Research Center
• Texas A&M University
• Awards
  – 2008: Dr. Dobb's Excellence in Programming award.
  – 2005: The William Procter Prize for Scientific Achievement from Sigma Xi (the scientific research society).
  – 2004: Member of the Texas Academy of Medicine, Engineering, and Science.
  – 1995: Named one of “the 20 most influential people in the computer industry in the last 20 years” by BYTE magazine.
  – 1993: AT&T Bell Laboratories Fellow.
  – 1990: Named one of "America's twelve top young scientists" by Fortune Magazine.
template<class Iter>
bool is_palindrome(Iter first, Iter last)
    // first points to the first element, last to the last element
{
    if (first < last)
        return (*first == *last) ? is_palindrome(++first, --last) : false;
    return true; // we reached the middle element
}

string s = "madamImadam";
bool b1 = is_palindrome(&s.front(), &s.back());

array<int> a = { 1, 2, 3, 4, 5, 5, 4, 3, 2, 1 };
bool b2 = is_palindrome(&a.front(), &a.back());
What I like – Great people and places
What I like – Great applications

- Telecommunications
- Google
- Microsoft applications and GUIs
- Linux tools and GUIs
- Games
- PhotoShop
- …

- Mars Rovers
- Marine diesel engines
- Cell phones
- Human genome project
- Micro electronics design and manufacturing
- …
CS is not just programming

• Actually, to write good programs you often need to know your
  – Algorithms
  – Data structures
  – Machine architecture
  – Discrete math
  – Programming languages, tools, and techniques
  – “Systems” (operating systems, data bases, …)
  – Human-machine interactions
  – Build, testing, and validation techniques
  – …

• And most of all: your application area
  – Physics
  – Biology
  – Classical literature
  – Accountancy
  – …
CS is not just programming

• Much of CS work is communication
  – Understanding new things
    • Reading
    • Listening
    • Reasoning, discussing
  – Giving presentations
  – Writing
  – All major projects are teamwork
  – Talking with
    • End users
    • Designers
    • Programmers
    • Managers
  – Teaching
8000+ Programming Languages

- C++’s family tree (part of)

Assembler ➔ Fortran ➔ Algol ➔ Pascal ➔ C ➔ C++ ➔ Ada ➔ Object Pascal ➔ Ada95 ➔ C89/99 ➔ C++0x ➔ C# ➔ Java

Lisp ➔ Smalltalk ➔ ML

- And this is a gross oversimplification!
Programming languages

• A programming language exists to help people express ideas
  – Programming language features exist to serve design and programming techniques
Assembler – 1951

• Machine code to assembler and libraries
  – Abstraction
  – Efficiency
  – Testing
  – documentation

**THE USE OF SUB-ROUTINES IN PROGRAMMES**

D. J. Wheeler
Cambridge & Illinois Universities

WVU the prime objectives to be born in mind when constructing them are simplicity of use, correctness of codes and accuracy of description. All complexities should-if possible -be buried out of sight.
Fortran –1956

• A notation fit for humans
  – For a specific application domain
    • $A(I) = B(I) + C \times D(I)$
  – Efficiency a premium
  – Portability
Simula –1967

• Organize code to model “the real world”
  – Object-oriented design

• Let the users define their own types (classes)
  – In general: concepts map to classes
  – “Data abstraction”

• Organize classes into hierarchies
  – Object-oriented programming
C – 1974

• An simple and general notation for systems programming
  – Somewhat portable
  – Direct mapping of objects and basic operations to machine
    • Performance becomes somewhat portable
C with Classes –1980

• General abstraction mechanisms to cope with complexity
  – From Simula

• General close-to-hardware machine model for efficiency
  – From C

  – Became C++ in 1984
  – Commercial release 1985
ISO Standard C++

• C++ is a general-purpose programming language with a bias towards systems programming that
  – is a better C
  – supports data abstraction
  – supports object-oriented programming
  – supports generic programming

• A multi-paradigm programming language
  – The most effective styles use a combination of techniques

From day 1 (1980)
From mid-1983
From about 1994
What’s distinctive about C++?

- **Stability**
  - Essential for real-world software
  - 1985-2008
  - 1978-2008 (C and C with Classes)

- **Non-proprietary**
  - Yet almost universally supported
  - ISO standard from 1998

- **Direct interface to other languages**
  - Notably C, assembler, Fortran

- **Abstraction + machine model**
  - Zero overhead principle
    - For basic operations (e.g. memory access) and abstraction mechanisms
  - User-defined types receive the same support as built-in types
  - Standard library written in the language itself
    - And most non-standard libraries
Aims for C++

• Support real-world software developers
  – “better software now”
  – by “better” I mean correct, maintainable, efficient, portable, …

• Change the way people think about software
  – Object-oriented programming
  – Generic programming
  – Resource management
  – Error handling

• Functional, not academic, beauty
  – “even I could have designed a much prettier language” – B.S. 1984 or so
C++ ISO Standardization

- About 22 nations
  - 5 to 12 at a meeting
- Membership have varied
  - 100 to 200+
    - 200+ members currently
  - 40 to 100 at a meeting
    - ~60 currently
- Most members work in industry

- Most are volunteers
  - Even many company representatives
- Most major platform, compiler, and library vendors are represented
  - E.g., IBM, Intel, Microsoft, Sun
- End users are underrepresented
C++ standardization – why bother?

• The ISO standards process is central
  – Standard support needed for mainstream use
    • Huge potential for improvement of application code
    • For (far too) many “if it isn’t in the standard it doesn’t exist
  – Significant defense against vendor lock-in
  – C++ has no rich owner
    • who can dictate changes, pay for design, implementation, marketing, etc.
  – The C++ standards committee is the central forum of the C++ community
    • Endless discussions among people who would never meet otherwise
  – The committee receives massive feedback from a broad section of the community
    • Much of it industrial
  – The committee is somewhat proactive
    • Adds features not previously available in the C++ world
C++ ISO Standardization – Results

1998  ISO standard
   – 22-0 vote

2003  Technical Corrigenda
   – “bug fix release”; no new features

2008  Registration draft for C++0x
   – Should lead to C++09

• Technical reports
  – Decimal floating point (2008)
  – Library2
  – Modularity
C++0x: 2002-2008

• Overall goals
  – Make C++ a better language
    • for systems programming
    • for library building
  – Make C++ easier to teach and learn
    • generalization
    • better libraries

• Massive pressure for
  – More language features
  – Stability / compatibility
    • Incl. C compatibility

• Insufficient pressure for
  – More standard libraries
    • The committee doesn’t have the resources required for massive library development
C++0x: Areas of change

• Machine model and concurrency
  – Memory model
  – Threads library, asynchronous return
  – Atomic API
  – Thread-local storage

• Support for generic programming
  – concepts
  – auto, decltype, template aliases, Rvalue references, …
  – General and uniform initialization
  – Lambdas

• Etc.
  – improved enums
  – long long, C99 character types, etc.
  – …

• Libraries
  – Regular expressions
  – Hashed containers
  – …
Retrospective

• Why did C++ succeed?
• What has C++ become?
• Is there a future for the ideals that underlie C++?
Why did C++ succeed?

• Reasons
  – Low-level access plus abstraction mechanisms
    • Performance
    • Direct access to real hardware
    • Very general zero-overhead abstraction
  – C compatibility
  – A useful tool (from day #1)
  – Timing
  – Non-proprietary – ISO standard
  – Stable
  – Evolving

“Being best at one or two things is not enough, you must be good enough at everything someone consider important”
Why did C++ succeed?

• Popular non-reasons
  – Just luck
    • For 25 years!
  – AT&T’s marketing might
    • Must be a joke 😊
  – It was first
    • Except for Ada, CommonLoops, Smalltalk, Eiffel, Objective C, Modula-2, C, Fortran, ML, …
  – Just C compatibility
    • Never 100%
  – It was cheapest
    • Not for most of its lifetime (incl. all the early years)
What is C++?

- A multi-paradigm programming language
- A hybrid language
- Embedded systems programming language
- It’s C!
- Supports generic programming
- A random collection of features
- Low level!
- Too big!
- Buffer overflows
- An object-oriented programming language
- Template meta-programming!
C++0x

• It feels like a new language
• How can I categorize/characterize it?
• It’s not just “object oriented”
  – Many of the key user-defined abstractions are not objects
  – Much of what I do as a programmer has to do with
    • Types
    • Classifications and manipulation of types (types of types)
    • Algorithms (generalized versions of computation)
    • Resource lifetimes
• The pieces (language features) fit together much better than they used to
C++

A language for building software infrastructures and resource-constrained applications

A light-weight-abstraction programming language
Future

• Lots of use
  – C++0x
  – C++1x
  – …

• Is there a future for “the C++ model” beyond C++?
  – Direct map to hardware
  – Zero-overhead abstraction
  – Minimal run-time environment
  – Destructor-based resource management
  – Heavy use of stack

• Challenges
  – Small language (or at least much, much smaller)
  – Complete and enforced type safety
  – Concurrency

I think it can be done

yes
Thanks!

- C and Simula
  - Brian Kernighan
  - Doug McIlroy
  - Kristen Nygaard
  - Dennis Ritchie
  - ...
- ISO C++ standards committee
  - Steve Clamage
  - Francis Glassborow
  - Andrew Koenig
  - Tom Plum
  - Herb Sutter
  - ...
- C++ compiler, tools, and library builders
  - Beman Dawes
  - David Vandevoorde
  - ...
- Application builders
More information

- My HOPL-II and HOPL-III papers
- The Design and Evolution of C++ (Addison Wesley 1994)
- My home pages
  - Papers, FAQs, libraries, applications, compilers, …
    - Search for “Bjarne” or “Stroustrup”
- The ISO C++ standard committee’s site:
  - All documents from 1994 onwards
    - Search for “WG21”
- The Computer History Museum
  - Software preservation project’s C++ pages
    - Early compilers and documentation, etc.
      - http://www.softwarepreservation.org/projects/c_plus_plus/
      - Search for “C++ Historical Sources Archive”