Directions for C++0x

Bjarne Stroustrup
AT&T Labs – Research
http://www.research.att.com/~bs
Prerequisites for goals

- Exciting enough to attract good people to the committee
- Ambitious enough to serve users well
- Compatible enough to make transition simple
- Specific enough to clearly state aim
If we do nothing about direction

- C++ will fossilize
- C++ will become de facto proprietary
- C++ will be dominated by bindings to external "standards"
- The committee will become a small club
  - Will focus on minute details of increasing irrelevance to programmers
- Changes will be made without direction

- Some say that this has already happened, but 1997-2001 was a deliberate period of calm to enhance stability
  - Now is the time to start discussing and planning
Prerequisites imply

- Focus on a few major topics
- Minor changes should be done to increase language and standard library uniformity and consistency
  - No major new language features are needed
- Changes should focused on support for programming styles and for application areas
  - Not language technicalities
Overall goals

- Make C++ a better language for systems programming and library building
  - Rather than providing specialized facilities for a particular sub-community (e.g. numeric computation or Windows application development)
- Make C++ easier to teach and learn
My view of directions

• Extend standard primarily through major standard library additions
  – Provide support for distributed programming
  – Improve support for platform-independent systems programming
• Remove inconsistencies and errors from core language
  – Don’t add major extensions
  – Remove embarrassments
• Offer a merger of C and C++ standards
  – We need a small joint C/C++ group to agree on rules for a merger
• Don’t compromise C++ as a systems programming language
  – 0-overhead principle
• Minimize incompatibilities with C++98
  – Complete compatibility infeasible
  – Be as careful as the C committee was
Standard library ideas
(suggested concrete examples)

- Simple elements of standard platform
  - set of resource handles supporting “resource acquisition is initialization”
  - directories, TCP/IP, advanced I/O (async, multiplexed, memory mapped)…

- Distributed computing
  - XTI (eXtended Type Information)
  - Threads
  - Remote invocation (incl. Async)
  - Remote instantiation, name server interface

- Make the standard library central to bindings to other systems
  - CORBA, SQL, …

- Add a few “general utility” facilities
  - Hash_map
  - Pattern matching
Core language ideas
(suggested concrete examples)

- Increase consistency
  - identical lookup for functions and function objects
- Improve support for generic programming
  - template typedefs, typeof()
- Remove embarrassments
  - Frequent questions, frequent novice errors
    - a vector and a string that are range checked by default
    - Prohibit default copying of objects with destructors
    - Give a class with virtual functions a virtual destructor by default
  - vector<list<int>>