Parasol
Smarter computing.
Texas A&M University

Places, People, and Code

Bjarne Stroustrup
Texas A&M University
April 28, 2012

Parasol
Smarter computing.
Texas A&M University

Places

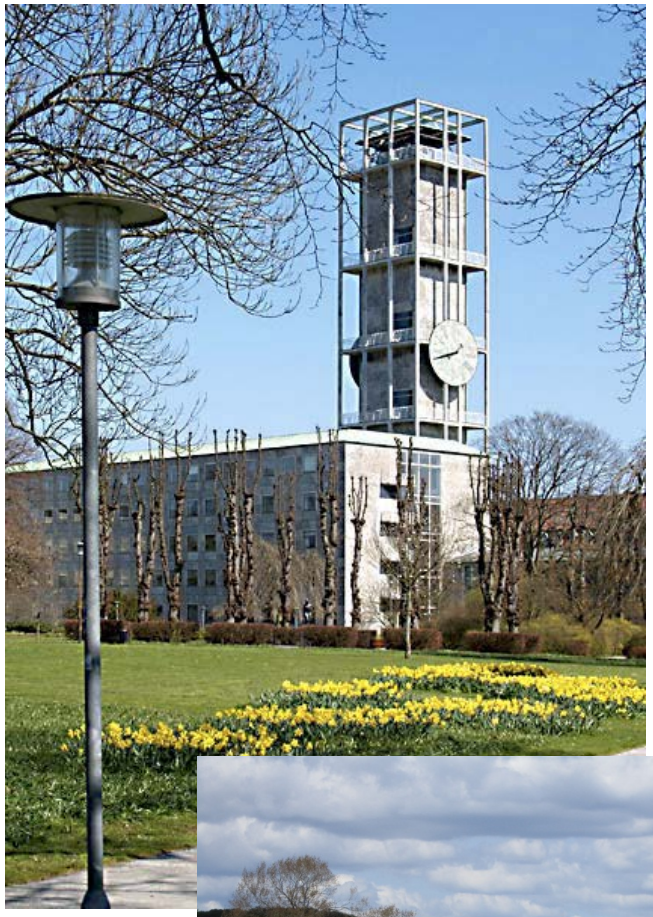
Texas A&M University
College Station

Bell Labs
Murray Hill, NJ

Cambridge

Aarhus

Aarhus




SWEDEN
Hjerring
NORTH SEA
Thisted
Aalborg
Randers
Aarhus Airport (AAR)
Holstebro
Aarhus
Herning
DENMARK
Helsinger
Horsens
Copenhagen
Odense Airport (ODE)
Copenhagen Airport (CPH)
Esbjerg
Odense
Slagelse
Haderslev
Svendborg
Nykobing
GERMANY

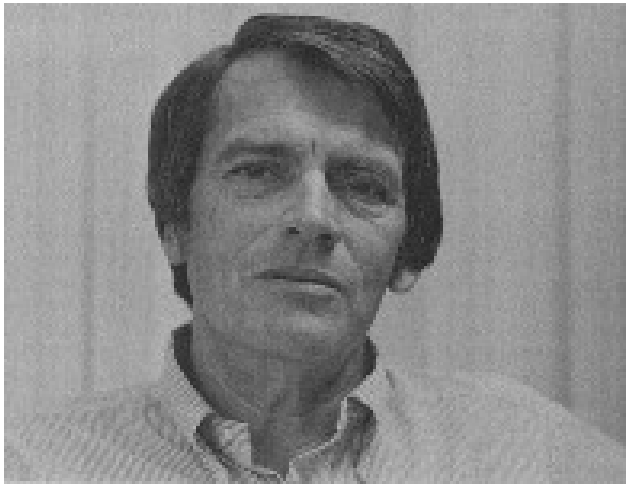

© 2012 Google

Parasol
Smarter computing.
Texas A&M University

Aarhus University

Microprogramming

- Instruction set design
- Machine architecture
- Machine Oriented Languages
- "High-level" instruction sets and interfaces
  - Inspirational talk by Bob Barton
- The OS/hardware interface
- Concurrent execution



Simula

- Object-oriented programming
- Object-oriented design
- Concurrent programming

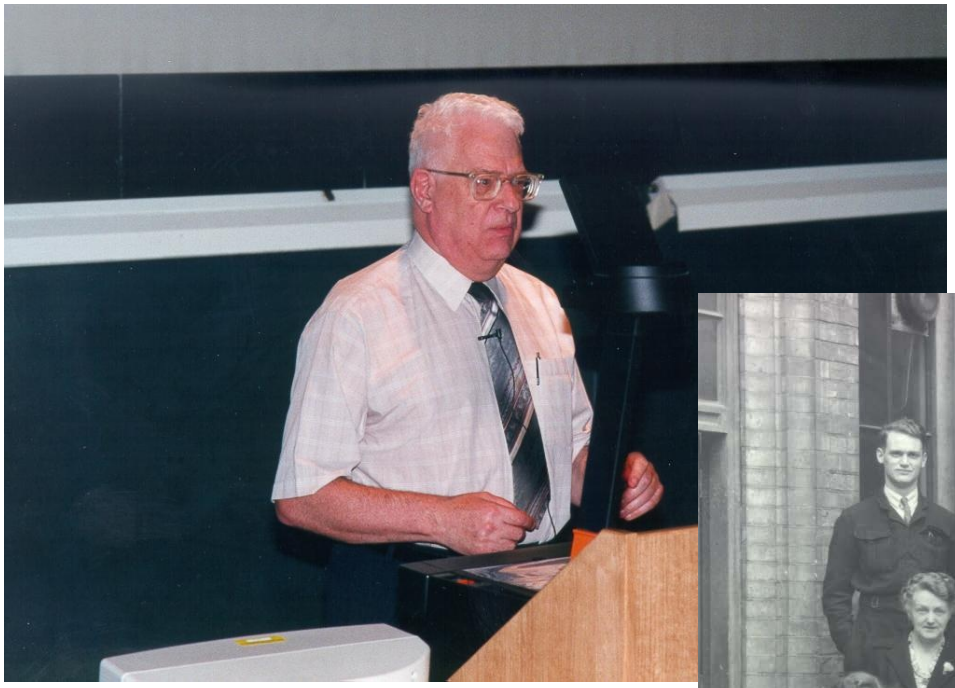
Parasol
Smarter computing.
Texas A&M University

Cambridge

Cambridge Computer Lab

Parasol
Smarter computing.
Texas A&M University

Cambridge



"Keep a high external profile"

1949.
May 6th
Machine in operation for first time. Printed table of squares (0 - 99). time for programme 2 mins. 35 secs. Four tanks of battery 1 in operation.

Cambridge

- *Systems should be build to do useful things for real people* – Roger Needham
- Martin Richards

Parasol
Smarter computing.
Texas A&M University

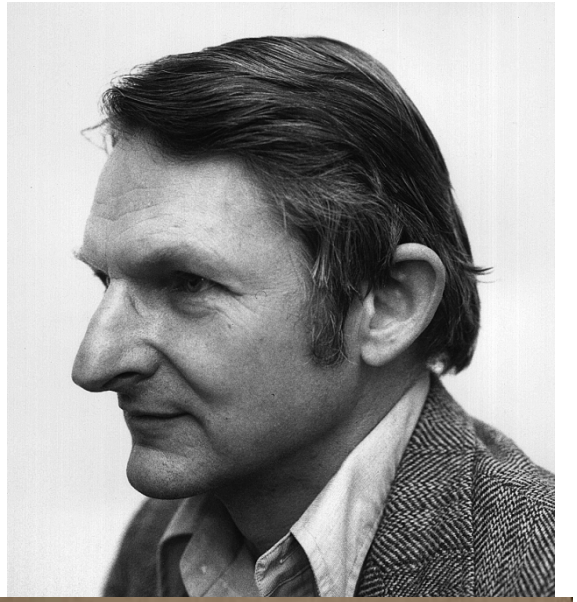
Cambridge

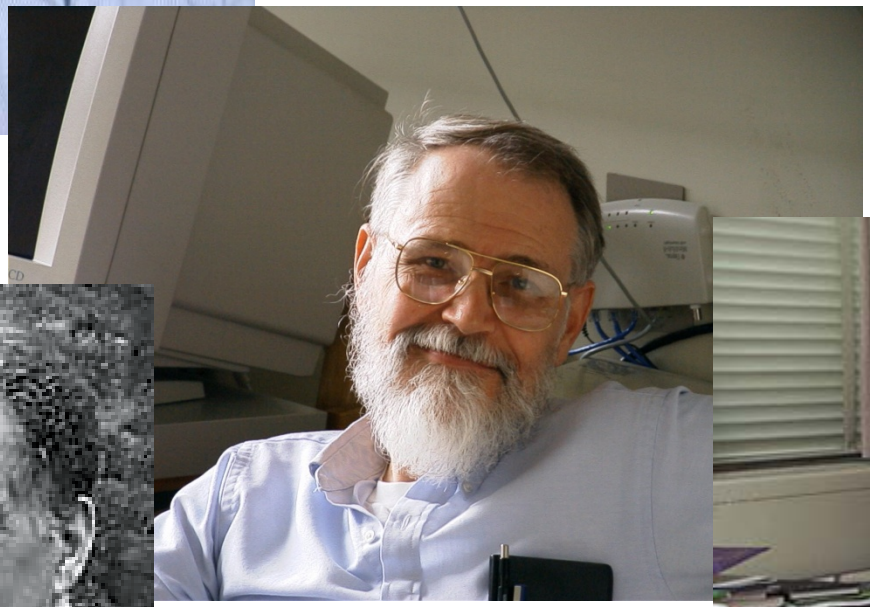
Parasol
Smarter computing.
Texas A&M University

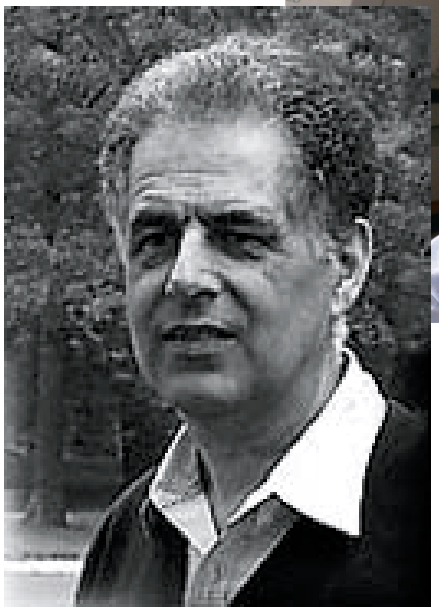
AT&T Bell Labs, Murray Hill, NJ

Parasol
Smarter computing.
Texas A&M University

127
aka
CSRC

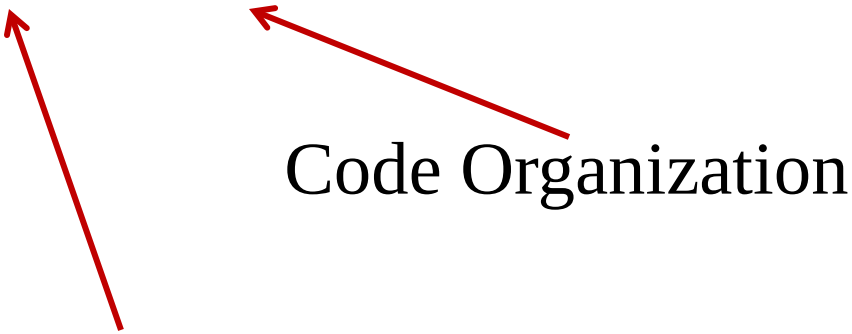






C with Classes

- C + Simula



Systems programming and performance

- A strong and flexible static type system was/is the ideal/aim
- "Elegant ***and*** efficient code"

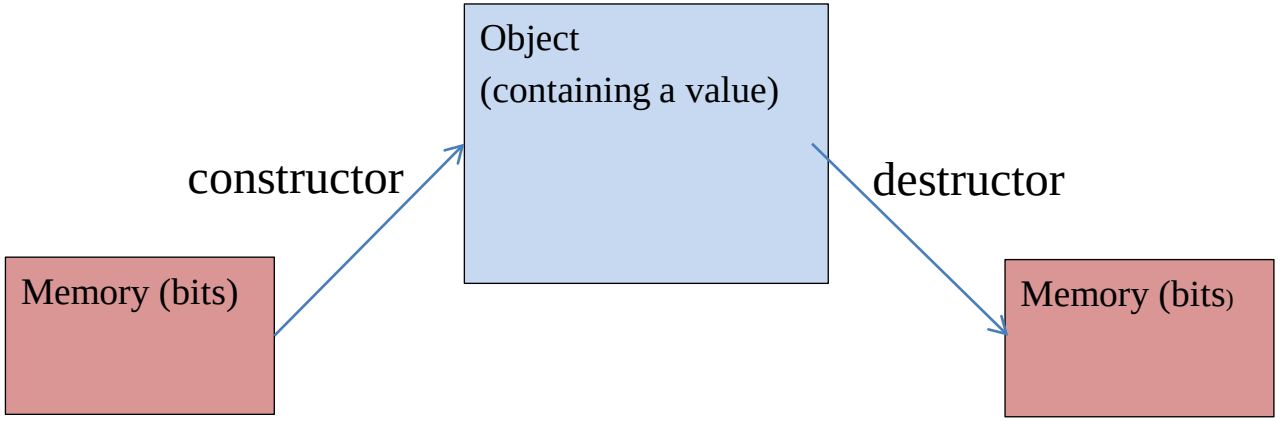
Parasol
Smarter computing.
Texas A&M University

Constructors and destructors

- A constructor established the run-time environment for member functions
  - incl. acquiring needed resources
- The destructor reverses the actions of the constructor
  - incl. releasing owned resources

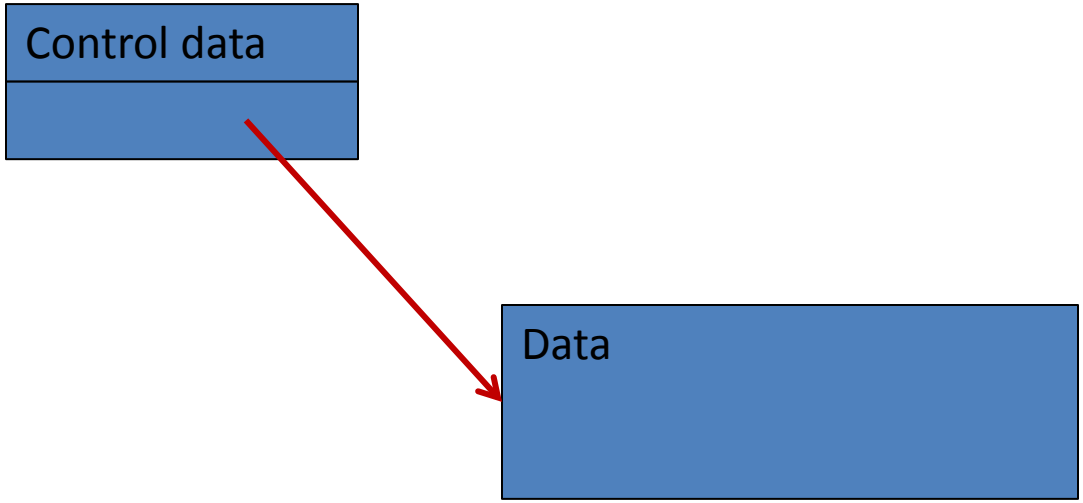
"The idea was to allow the programmer to establish guarantees, sometimes called ''invariants,'' that other member function could rely on."

Constructors and destructors



- Constructor: make an object from memory
  - An object holds a value
  - Has a type
  - Has an interface
  - Has meaning
- Destructor: make an object (back) into memory
  - Memory is just interpreted bits

A resource handle



- Examples
  - Containers
    - vector, list, map, …
  - Smart pointers
    - unique_ptr, shared_ptr, delayed_value, remote_object, …
  - Locks, thread handles, sockets, iostreams, file handles
  - …

Vector: the archetypical resource handle

- Slight simplification of **std::vector**

```
template<typename T>                    // T is the element type
class Vector {
public:
    Vector();                           // default constructor: make empty vector
    Vector(int  n);                     // constructor: initialize to n elements
    Vector(initializer_list<T>) ;       // constructor: initialize with element list
    ~Vector();                          // destructor: deallocate elements
    int size() ;                        // number of elements
    T& operator[](int i);               // access the ith element
    void push_back(const T& x);         // add x as a new element at the end
    T* begin();                         // fist element
    T* end();                           // one-beyond-last element
private:
    int sz;                             // number of elements
    T* elem;                            // pointer to sz elements of type T
};
```

Parasol
Smarter computing.
Texas A&M University

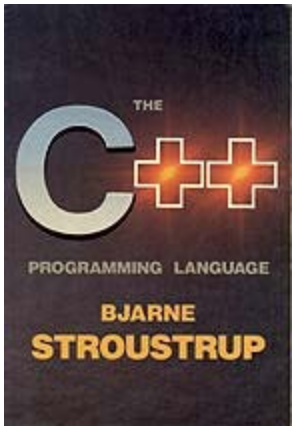
Some liked C++, some didn't

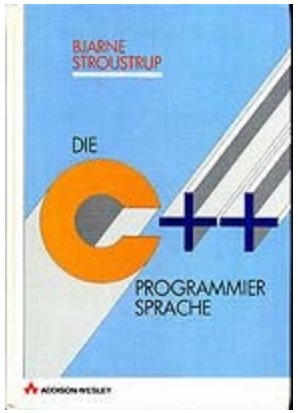
Parasol
Smarter computing.
Texas A&M University

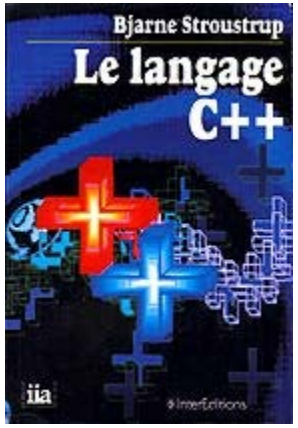
Bell Labs – many years later

Parasol
Smarter computing.
Texas A&M University

TC++PL

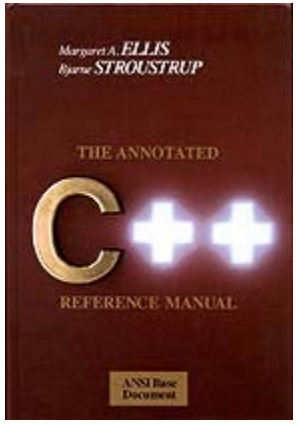


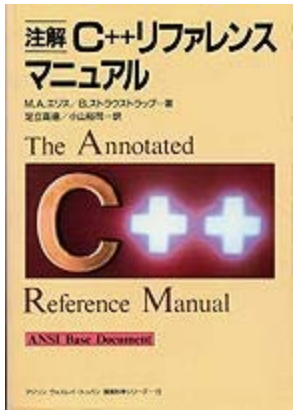


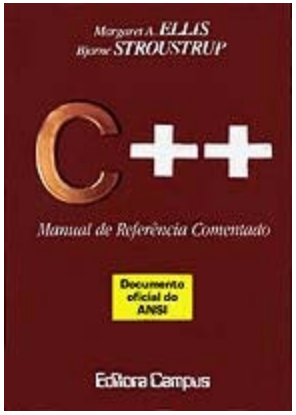


Parasol
Smarter computing.
Texas A&M University

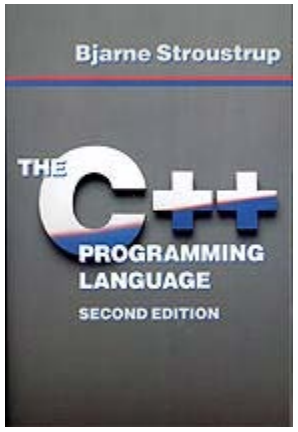
The ARM



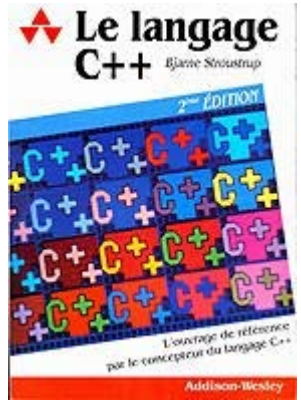




TC++PL2

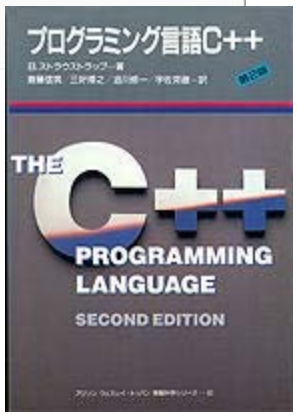

Bjarne Stroustrup
THE C++
PROGRAMMING LANGUAGE
SECOND EDITION


Die C++ Programmier-sprache
2. Auflage

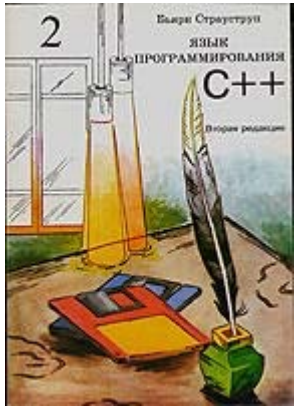

Le langage C++
Bjarne Stroustrup
2ème édition
Addison-Wesley


Bjarne Stroustrup
IL LINGUAGGIO C++
Seconda Edizione
ADDISON WESLEY MASSON


THE C++
程式設計語言
Bjarne Stroustrup 著


プログラミング言語C++
THE C++
PROGRAMMING LANGUAGE
SECOND EDITION

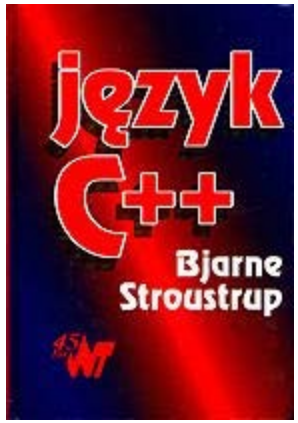

1
Бьярн Страуструп
ЯЗЫК ПРОГРАММИРОВАНИЯ
C++


2
Бьярн Страуструп
ЯЗЫК ПРОГРАММИРОВАНИЯ
C++


EL C++
LENGUAJE DE PROGRAMACIÓN
SEGUNDA EDICIÓN


Bjarne Stroustrup
C++
Programovací jazyk


Bjarne Stroustrup
C++
Die Programmiersprache
ADDISON-WESLEY

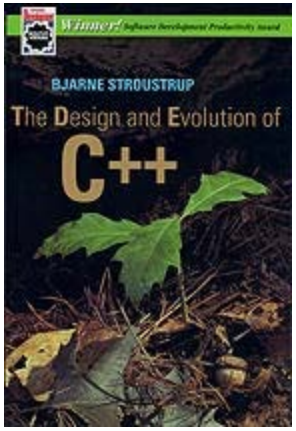

język C++
Bjarne Stroustrup

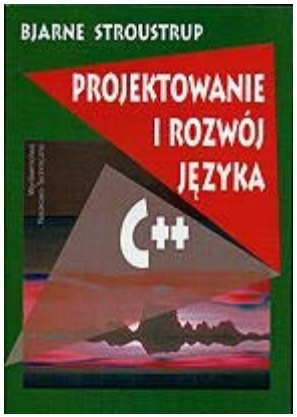

LE LANGAGE C++
2e édition
Bjarne Stroustrup

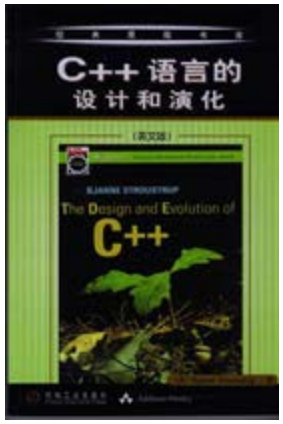

Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ
C++
Bjarne Stroustrup

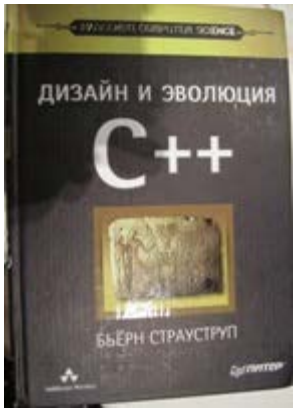
Parasol
Smarter computing.
Texas A&M University

D&E










Parasol
Smarter computing.
Texas A&M University
Bjarne Stroustrup
The Creator of C++
Die C++ Programmiersprache
El Lenguaje de Programación C++
Bjarne Stroustrup
The Creator of C++
Bjarne Stroustrup
(a C++ megalkotója)
A C++ programozási nyelv
ЯЗЫК C++
ПРОГРАММИРОВАНИЯ
специальное издание
Le Langage C++
Bjarne Stroustrup
Le créateur du C++
C++ 程式語言
C++ PROGRAMMING LANGUAGE
THIRD EDITION
EL LENGUAJE DE PROGRAMACIÓN C++
Tercera edición
Bjarne Stroustrup
A Linguagem de Programação C++
3ª Edição
De programmeertaal C++
Bjarne Stroustrup
De ontwikkelaar van C++
C++ 程序设计语言
THE C++ PROGRAMMING LANGUAGE
C++程序设计语言
(特别版)
The C++ Programming Language
Terza Edizione
Bjarne Stroustrup
LE LANGAGE C++
Bjarne Stroustrup
Edition spéciale, revue et corrigée
Специално издание
Програмния език C++
Bjarne Stroustrup
Creatorul limbajului C++
C++
Teora
The C++ Programming Language
C++ 프로그래밍 언어
Język C++
Die C++ Programmiersprache
Bjarne Stroustrup
Die C++ Programmiersprache

Parasol
Smarter computing.
Texas A&M University

Parasol
Smarter computing.
Texas A&M University

C++ Standardization

J16/WG21 Meetings 1979-2012-…

- Austin
- Batavia
- Bellevue
- Berlin
- Bloomington
- Boston
- Curacao
- Copenhagen
- Dublin
- Frankfurt
- Kona
- Lillehammer
- London
- Lund
- Madrid
- Morristown
- Monterey
- Mt. Tremblant
- Nashua
- Oxford
- Portland
- Pittsburg
- Portland
- Rapperswil
- Redmond
- Santa Cruz
- San Diego
- San Francisco
- Sidney
- Seattle
- Somerset
- Sophia Antipolis
- Stockholm
- Summit
- Tokyo
- Toronto
- Valley Forge
- Washington, D.C.

Parasol
Smarter computing.
Texas A&M University

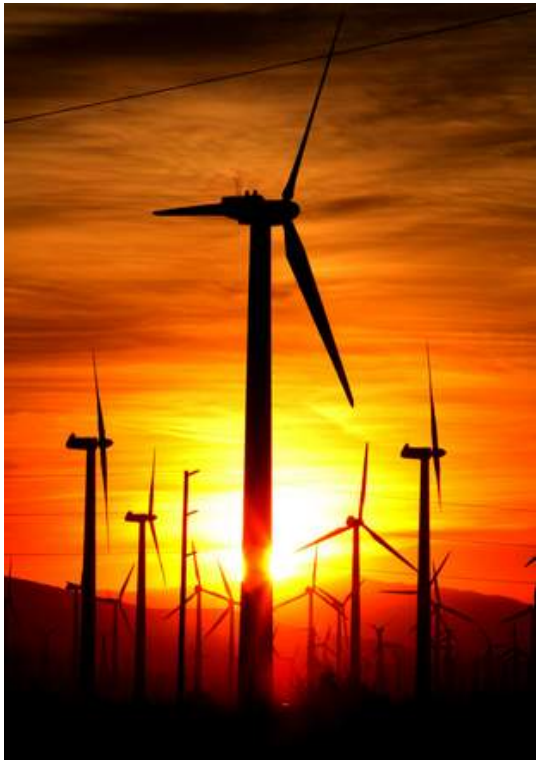
C++ standardization – why bother?

- The ISO standards process is central
  - 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
  - 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

Parasol
Smarter computing.
Texas A&M University

Interlocking themes

- Stability and Compatibility
  - "make the language much better but don't break my code"
- Scale
  - Million-line projects became common
  - Specification – precise and complete
  - Portability
- Resource management
  - Invariants
- Type safety
  - Containers
- Performance
  - Compactness
- Equal support for user-defined and built-in types
  - Value types, scoped objects
- User skills required



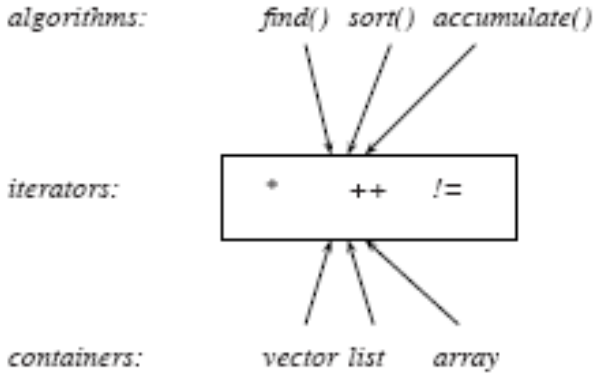
C++98 example: Resource management

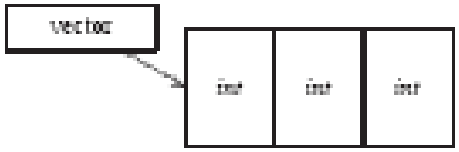
- Standard library containers
  - with exception-safety guarantees (e.g., **vector**)
  - the techniques can be used by every user (e.g., **File_handle**)
- No resources are leaked
  - Based on a simple and systematic view of resource management
    - Exception safety guarantees
    - RAII
  - Destructors do cleanup
    - guaranteed, implicitly

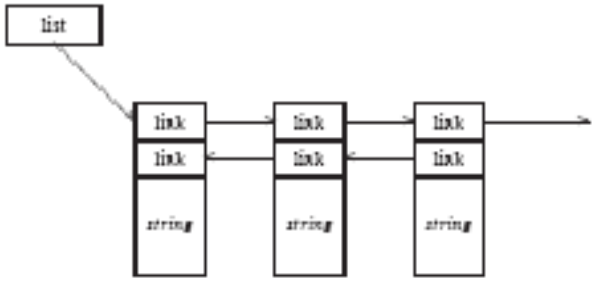
```
void f(string s)
{
    vector<int> v;
    File_handle h(s,"r"");
    // …
    int x;
    while (cin>>x) {
      if (x<=0) throw Bad_value(x);
      v.push(x);
    }
    // …
}
```

The STL

- Ideal: The most general and most efficient expression of an algorithm
  - Focus on algorithms
  - Separate algorithms from data
  - Go from the concrete to the abstract
    - Not the other way
  - Use compile-time resolution to eliminate overheads
    - Inlining and overloading
  - Where needed, parameterize with policies
    - E.g. sorting criteria



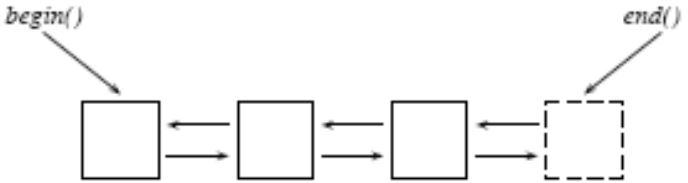




STL example: find_if()

- Definition

```
template<class Iter, class Pred>
Iter find_if(Iter first, Iter last, Pred p)
{
    while (first!=last && !p(*first)) // while not at end and predicate not met
      ++first;                        // advance to next element
    return first;  // return the element reached
}
```

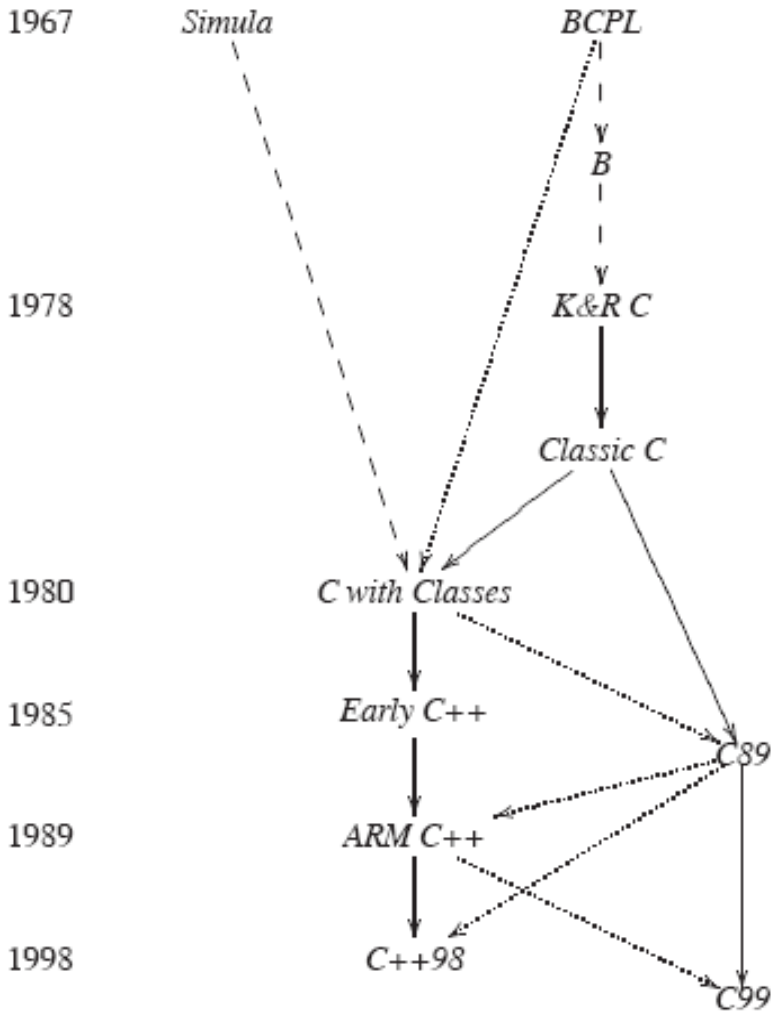


Design

- By committee?

C/C++ compatibility

- A constant sore point
  - Separate standards committees
    - A tragedy
  - Constant borrowing
    - Both ways
    - Often incompatibly
  - Widely demanded by users
    - Rightfully so
  - Widely despised by users
    - “Against OO”
    - “Against the spirit of C”



Parasol
Smarter computing.
Texas A&M University

C++11

- And now – just a year later – large parts of C++ is shipping!
  - GCC4.7
  - Microsoft 11 beta
  - …
  - "Everybody" ships the libraries

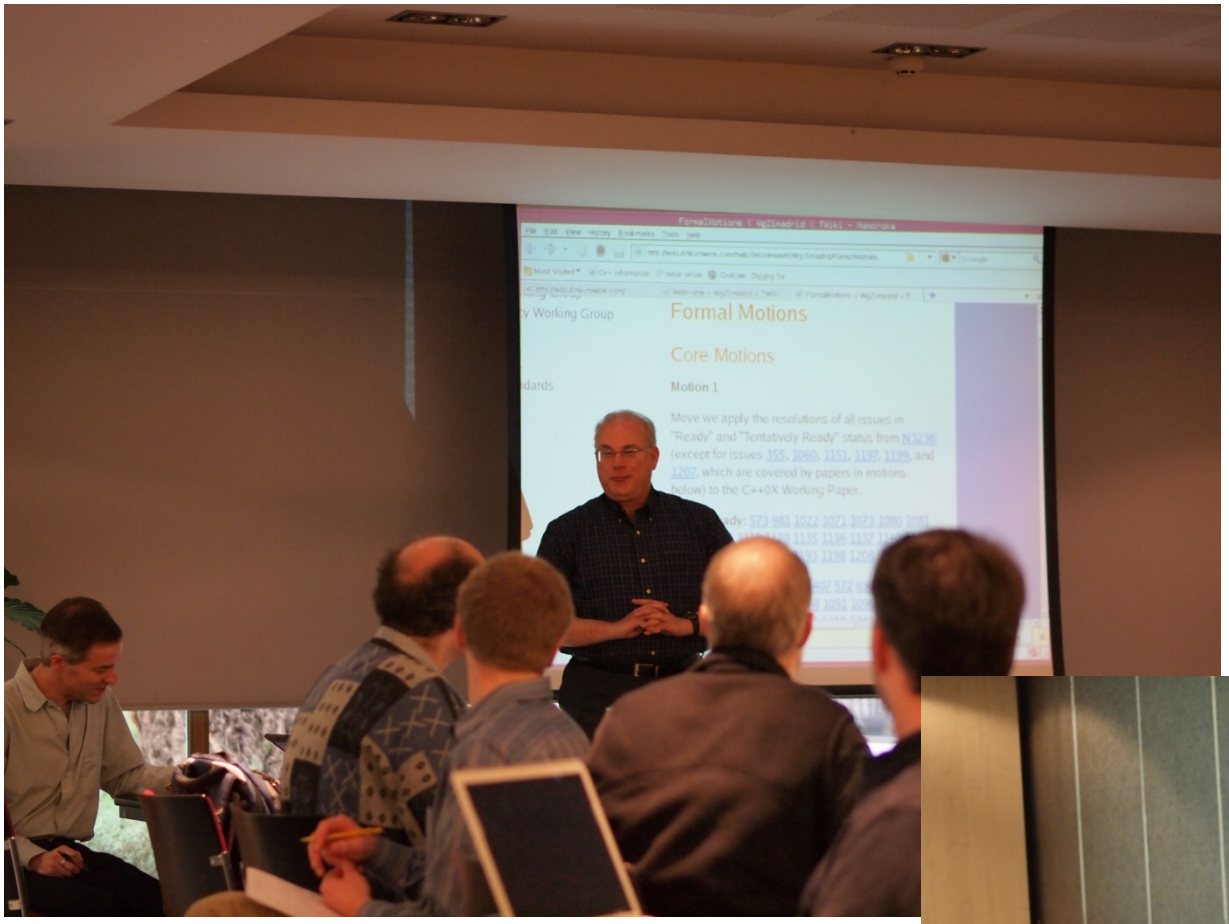
Parasol
Smarter computing.
Texas A&M University

- John Lakos: Bloomberg
- Stepanus Du Toit: Intel, Canada
- Loic Joly: INRIA, France

- Herb Sutter: Microsoft
- Steve Clamage: Oracle (Sun)
- Bill Plauger: Dinkumware
- Howard Hinnant: Apple

- Howard Hinnant: Apple
- Richard Corden : Programming Research
- Clark Nelson: Intel

Parasol
Smarter computing.
Texas A&M University



- Steve Clamage: Oracle (Sun)
- Jean-Paul Rigoux: U. de Provence, France
- J.C. Van Vinkel: Netherlands and Google
- Beeman Dawes: Boost
- Detleff Vollman: Switzerland

Parasol
Smarter computing.
Texas A&M University

- Loic Joly: INRIA, France
- Mike Miller: EDG
- Doug Gregor: Apple
- Roger Orr: UK
- Alisdair Meredith: Bloomberg

Parasol
Smarter computing.
Texas A&M University

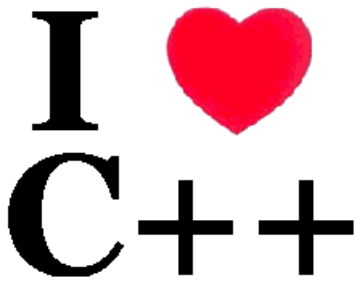
- Martin Sebor: Rogue Wave
- Pete Becker: Roundhouse Consulting
- Jaakko Jarvi: TAMU
- Daniel Garcia: U. Charlos III

- Hans Boehm: HP
- Beeman Dawes: Boost
- Mike Wong: IBM
- Lawrence Crowl: Google

- Herb Sutter: Microsoft
- Daniel Krugler: ???
- John Lakos: Bloomberg

Copy and move

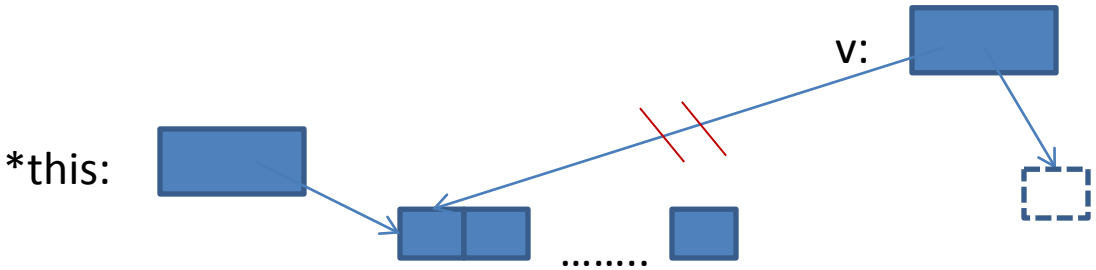
```
Vector<int*> find_all(Vector<int>& v, int val)      // find all occurrences of val in v
{
    Vector<int*> res;
    for (int& x : v)
        if (x==val)
            res.push_back(&x);          // add the address of the element to res
     return res;
}

void  test()
{
    Vector<int> lst {1,2,3,1,2,3,4,1,2,3,4,5};
    for (int* p :  find_all(lst,3))
        cout << "address: " << p << ", value: " << *p << "\
    // …
}
```



Copy and move

- Move constructor

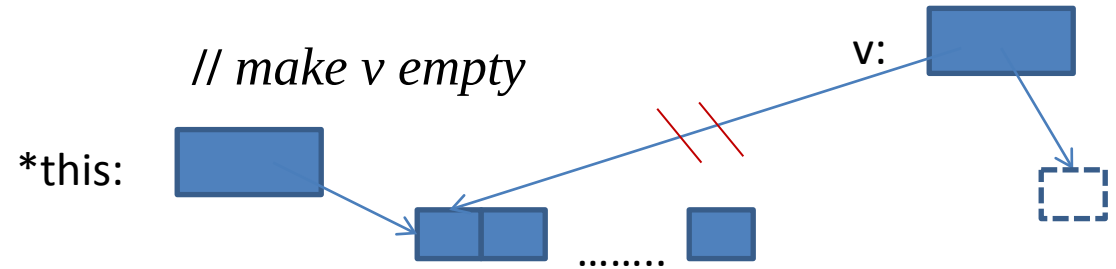
```
template<typename T>
Vector<T>::Vector(const Vector&& v)                    // move constructor
    : sz{v.sz}, elem{v.elem}      // grab v's elements
{
    v.elem = nullptr;             // make v empty
    v.sz = 0;
}
```



Copy and move

- Move assignment

```
template<typename T>
Vector<T>& Vector<T>::operator=(Vector<T>&& v)    // move assignment
{
    destroy<T>(elem,sz);        // delete old elements
    elem = v.elem;              // grab v's elements
    sz = v.sz;
    v.elem = nullptr;           // make v empty
    v.sz = 0;
    return *this;
}
```



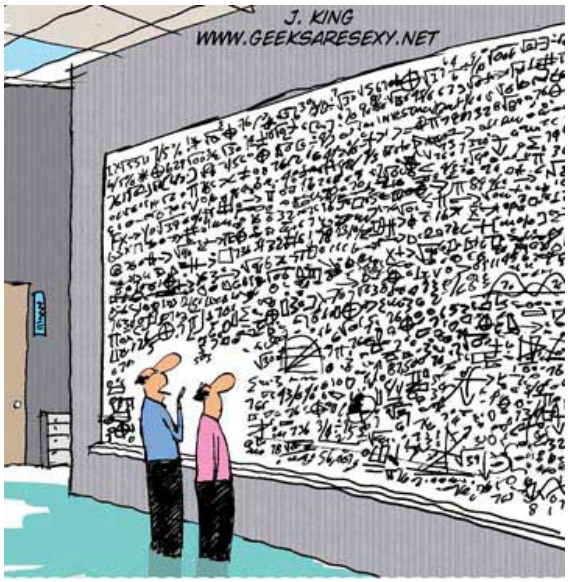
Vector

- Copy and move declarations added to Vector

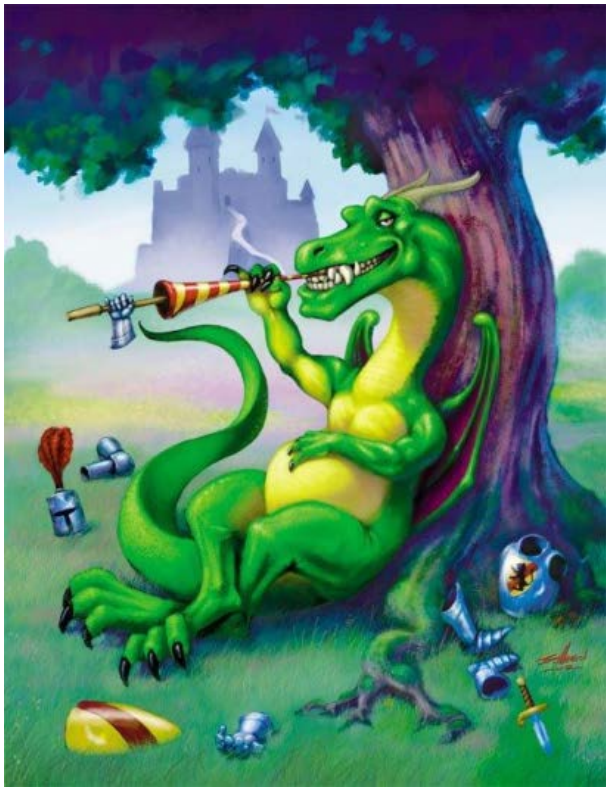
```
template<typename T>          // T is the element type
class Vector {
public:
    // …
    Vector(const Vector&);                  // copy constructor
    Vector(Vector&&);                       // move constructor
    Vector& operator=(const Vector&);       // copy assignment
    Vector& operator=(Vector&&);            // move assignment
    // …
};
```

The real problems

- Help people to write better programs
  - Easier to write
  - Easier to maintain
  - Easier to achieve acceptable resource usage



"...And that, in simple terms, is what's wrong with your software design."



Parasol
Smarter computing.
Texas A&M University

College Station

Parasol
Smarter computing.
Texas A&M University

Texas A&M

Parasol
Smarter computing.
Texas A&M University

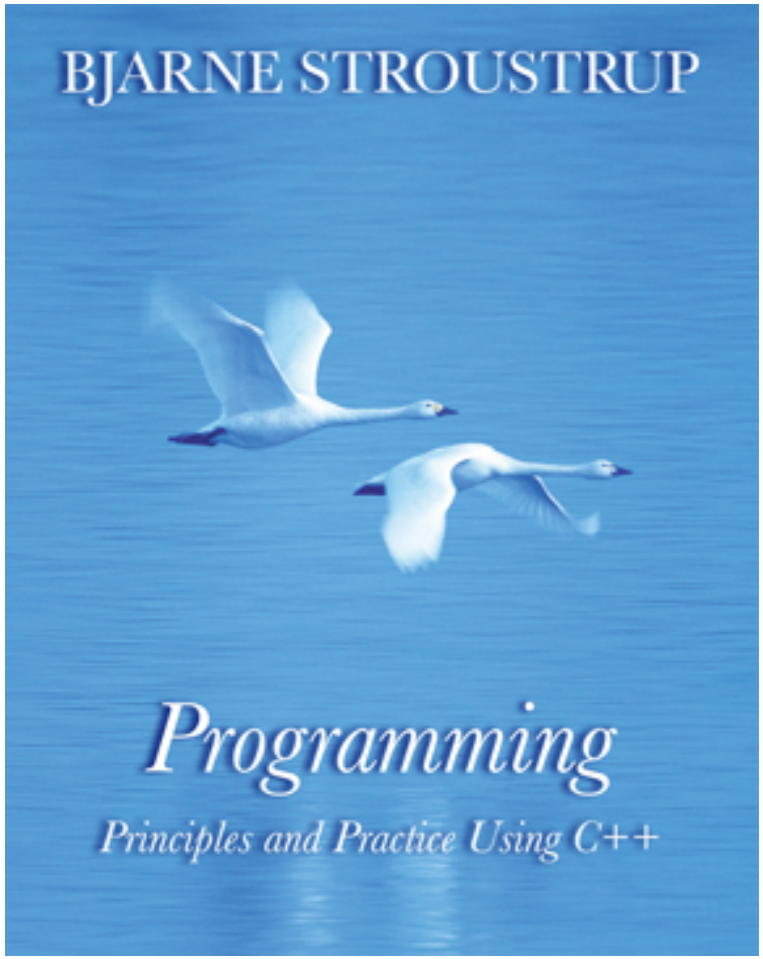
Texas A&M

Parasol
Smarter computing.
Texas A&M University

Texas A&M

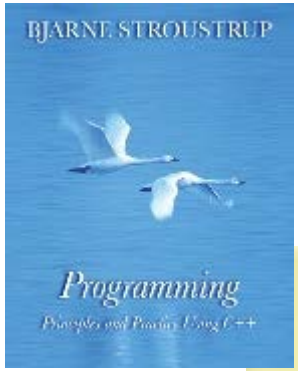
C++ is shockingly misused

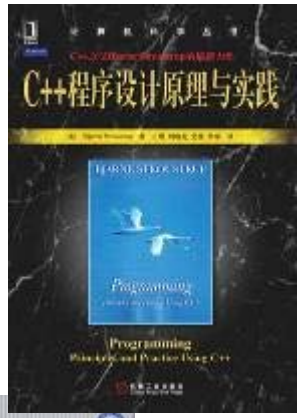
- "If that's C++, then I don't like it either!"
  - Bjarne Stroustrup, 2004
    - (after surveying a couple of dozen C++ textbooks)

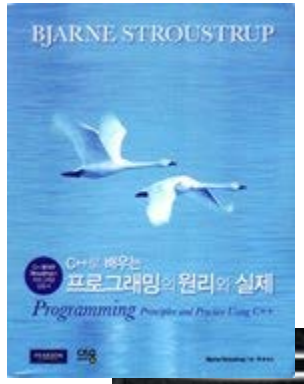


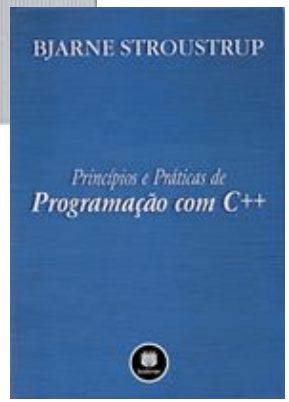
Parasol
Smarter computing.
Texas A&M University

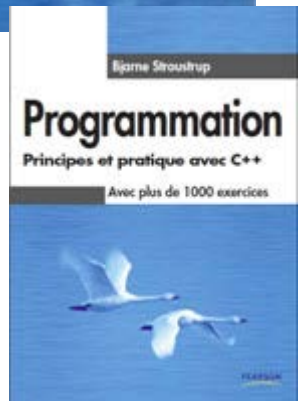
PPP







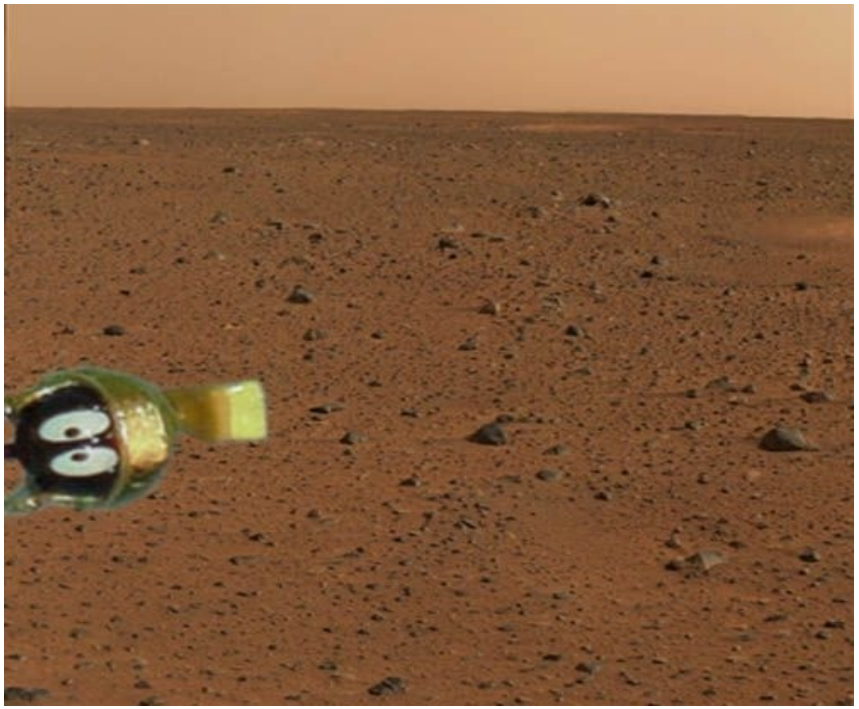




Parasol
Smarter computing.
Texas A&M University

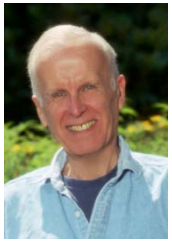
Programming languages

- A programming language exists to help people express ideas
- Programming language features exist to serve design and programming techniques
- The primary value of a programming language is in the applications written in it



The quest for better languages has been long and must continue

Programming Languages



Domain-specific abstraction

Fortran

Cobol

General-purpose abstraction

Simula

Java

C++

C++11

C#

Direct mapping to hardware

Assembler

BCPL

C

Parasol
Smarter computing.
Texas A&M University

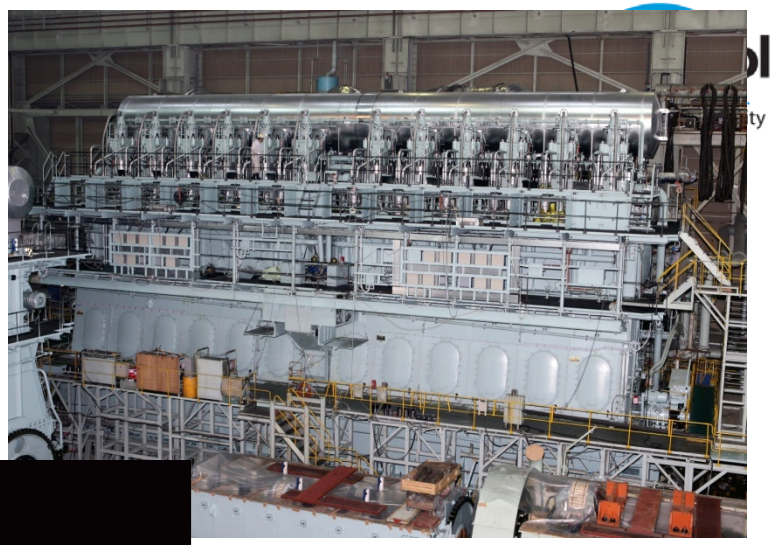
C++

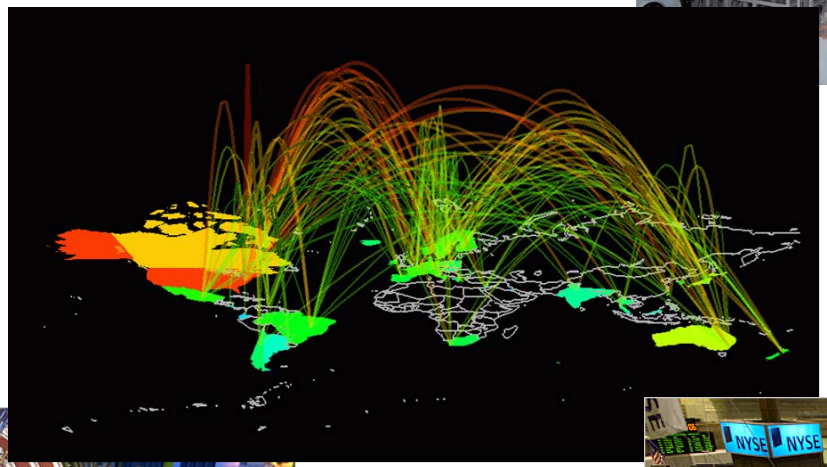
Key areas of strength:

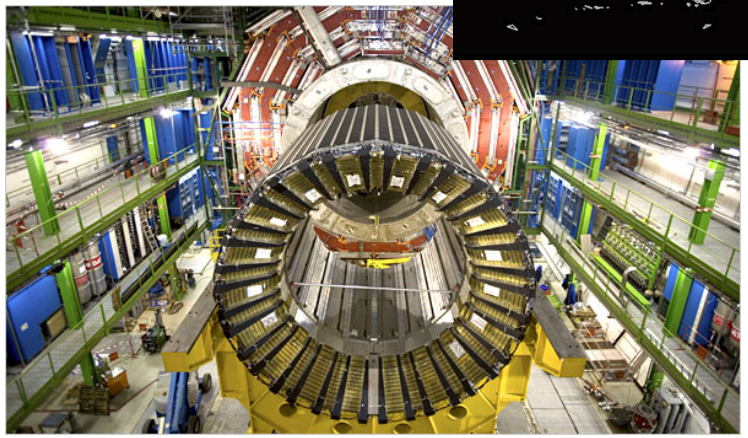
- Software infrastructure
- Resource-constrained applications

A light-weight abstraction programming language

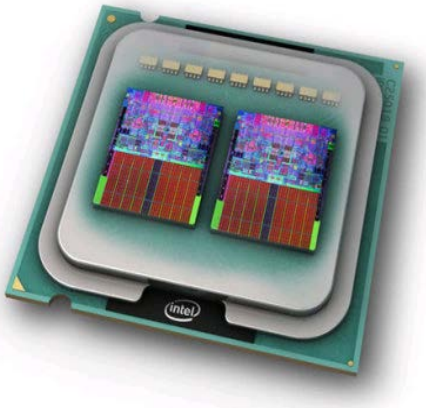
C++ applications







C++ Applications



- www.research.att.com/~bs/applications.html

C++ Applications

Parasol
Smarter computing.
Texas A&M University

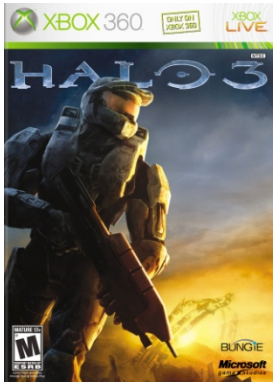
TOMTOM

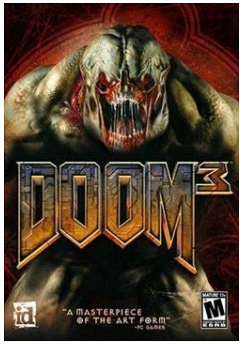
GARMIN

MySQL

PayPal

amazon





www.lextrait.com/vincent/implementations.html

AMADEUS
Your technology partner

Parasol
Smarter computing.
Texas A&M University

Thanks!