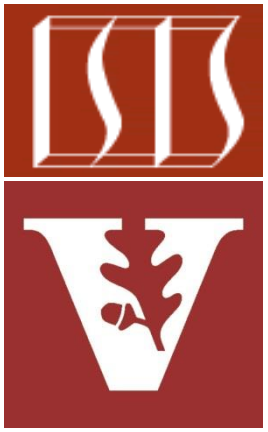
Overview of C++: Brief History

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt



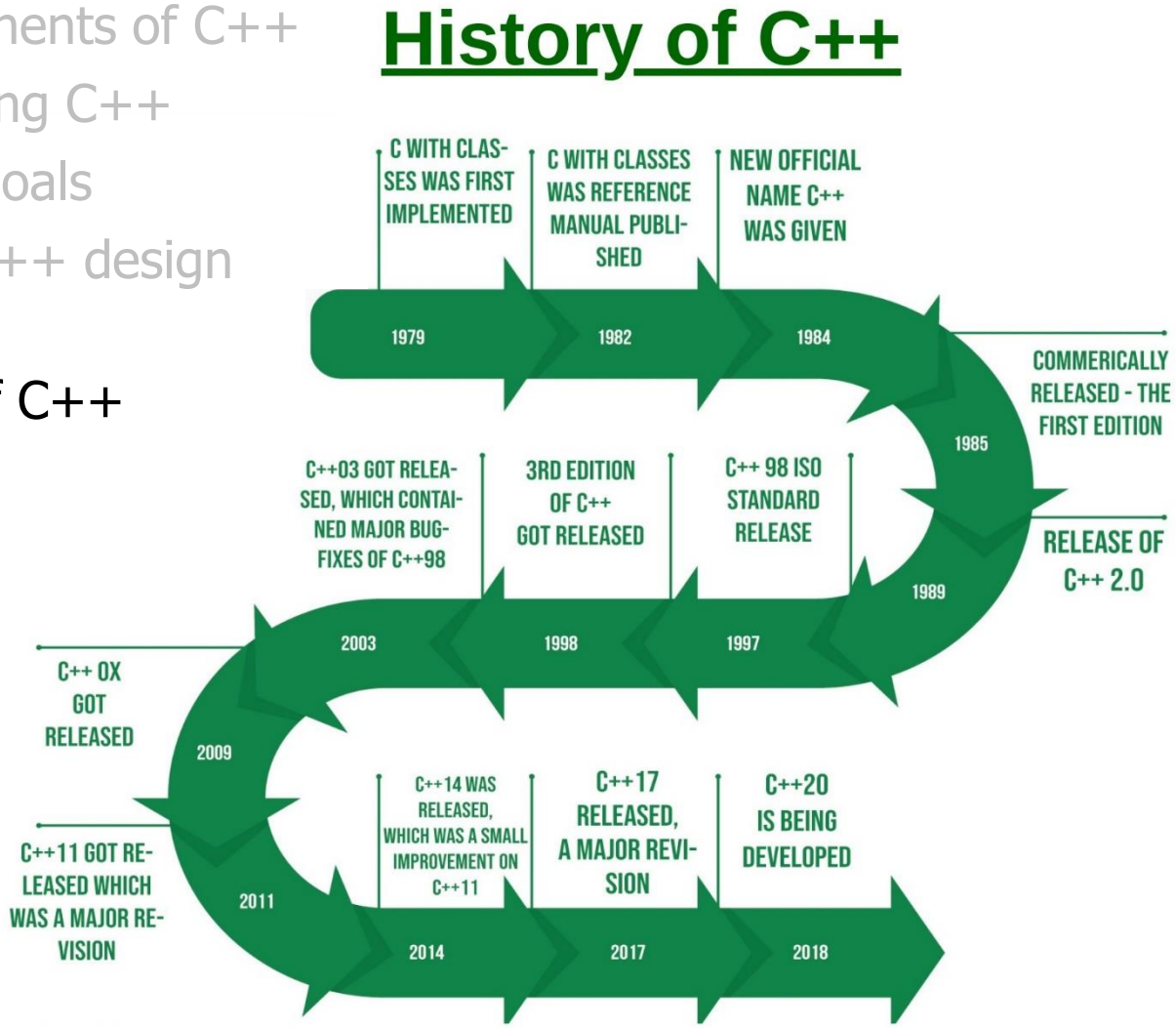
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

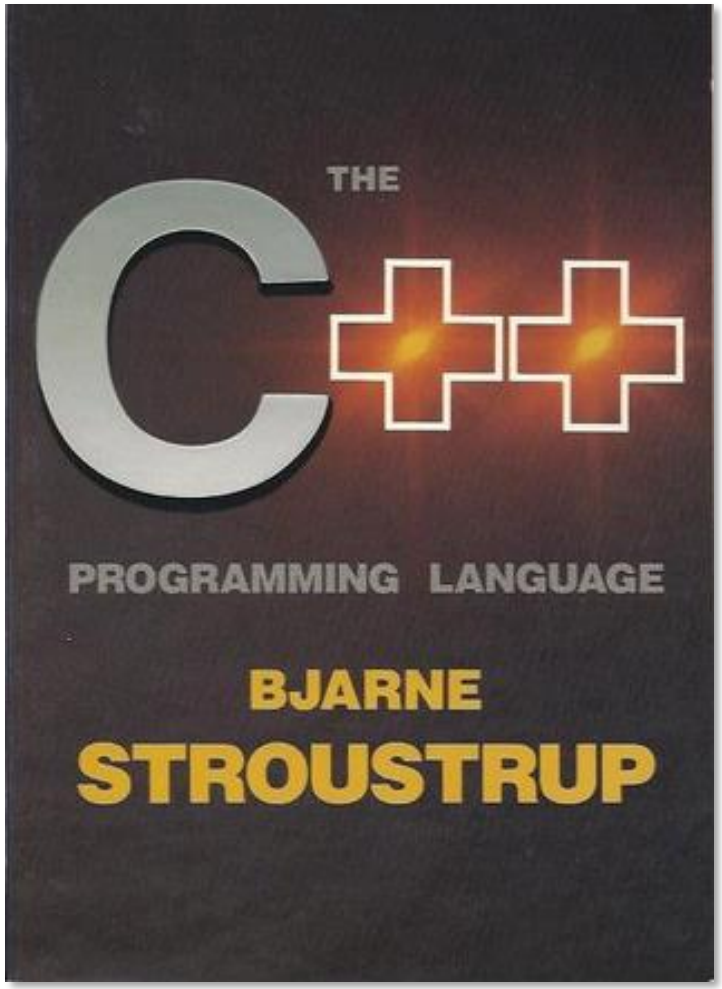
- Recognize the key components of C++
- Know strategies for learning C++
- Understand C++ design goals
- Learn about conflicts of C++ design goals
- Be aware of the history of C++



Overview of C++ History

C++ History in a Nutshell

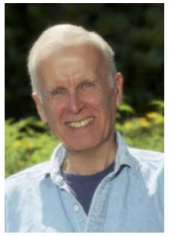
- C++ was designed at AT&T Bell Labs by Bjarne Stroustrup in the early 80's



See www.softwarepreservation.org/projects/c_plus_plus

C++ History in a Nutshell

- C++ was designed at AT&T Bell Labs by Bjarne Stroustrup in the early 80's
  - It originated from the confluence of object-oriented programming languages with systems programming languages



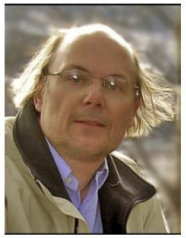
Domain-specific abstraction

Fortran

Cobol

General-purpose abstraction

Simula



Java

Direct mapping to hardware

C++

C++11

C++14

C#

Assembler

BCPL

C

C++: From Translator to Native Compiler

- The original cfront translated C++ into C for portability

Cfront

From Wikipedia, the free encyclopedia

Cfront was the original compiler for C++ (then known as "C with Classes") from around 1983, which converted C++ to C; developed by Bjarne Stroustrup at AT&T Bell Labs. The preprocessor did not understand all of the language and much of the code was written via translations. Cfront had a complete parser, built symbol tables, and built a tree for each class, function, etc. Cfront was based on CPre (C compiler, which was started in 1979).

As Cfront was written in C++, it was a challenge to bootstrap on a machine without a C++ compiler/translator. Along with the Cfront C++ sources, a special "half-preprocessed" version of the C code resulting from compiling Cfront with itself was also provided. This C code was to be compiled with the native C compiler, and the resulting executable could then be used to compile the Cfront C++ sources.

Most of the porting effort in getting Cfront running on a new machine was related to standard I/O. Cfront's C++ streams were closely tied in with the C library's buffered I/O streams, but there was little interaction with the rest of the C environment. The compiler could be ported to most System V derivatives without many changes, but BSD-based systems usually had many more variations in their C libraries and associated stdio structures.

See en.wikipedia.org/wiki/Cfront

C++: From Translator to Native Compiler

- The original cfront translated C++ into C for portability

```
class Base {
private:
  int private_data;
protected:
  int protected_data;
public:
  int public_data;

  Base (int arg = 1)
  : private_data (arg + 100),
    protected_data (arg + 10),
    public_data (arg)  {}
};
```

```
struct Base {
   int private_data__4Base;
   int protected_data__4Base;
   int public_data__4Base;
   struct __mptr *__vptr__4Base;
};
```

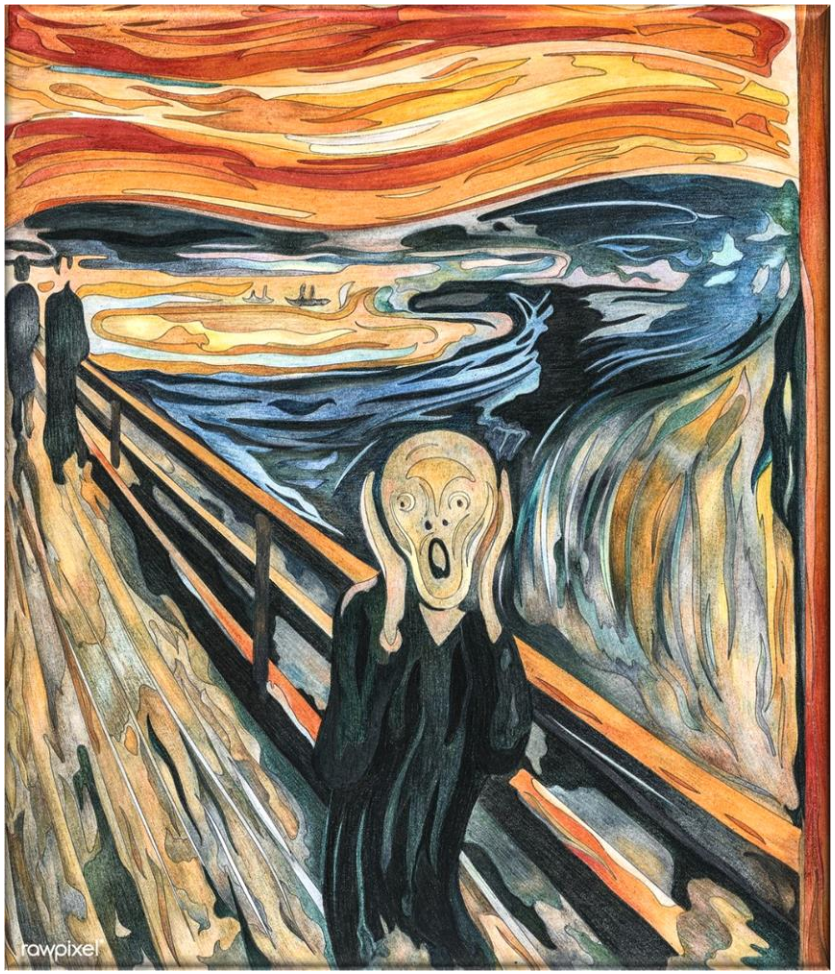
C++: From Translator to Native Compiler

- The original cfront translated C++ into C for portability

```
static struct Base *
__ct__4BaseFi (struct Base *__0this, int __0arg) {
  if (__0this ||
      (__0this =
         (struct Base *) __nw__FUi (sizeof (struct Base))))
            ((((__0this->__vptr__4Base =
               (struct __mptr *) __ptbl_vec__c___src_C_[0]),
            (__0this->private_data__4Base =
               (__0arg + 1000))),
            (__0this->protected_data__4Base =
               (__0arg + 100))),
            (__0this->public_data__4Base =
               __0arg));
   return __0this;
}
```

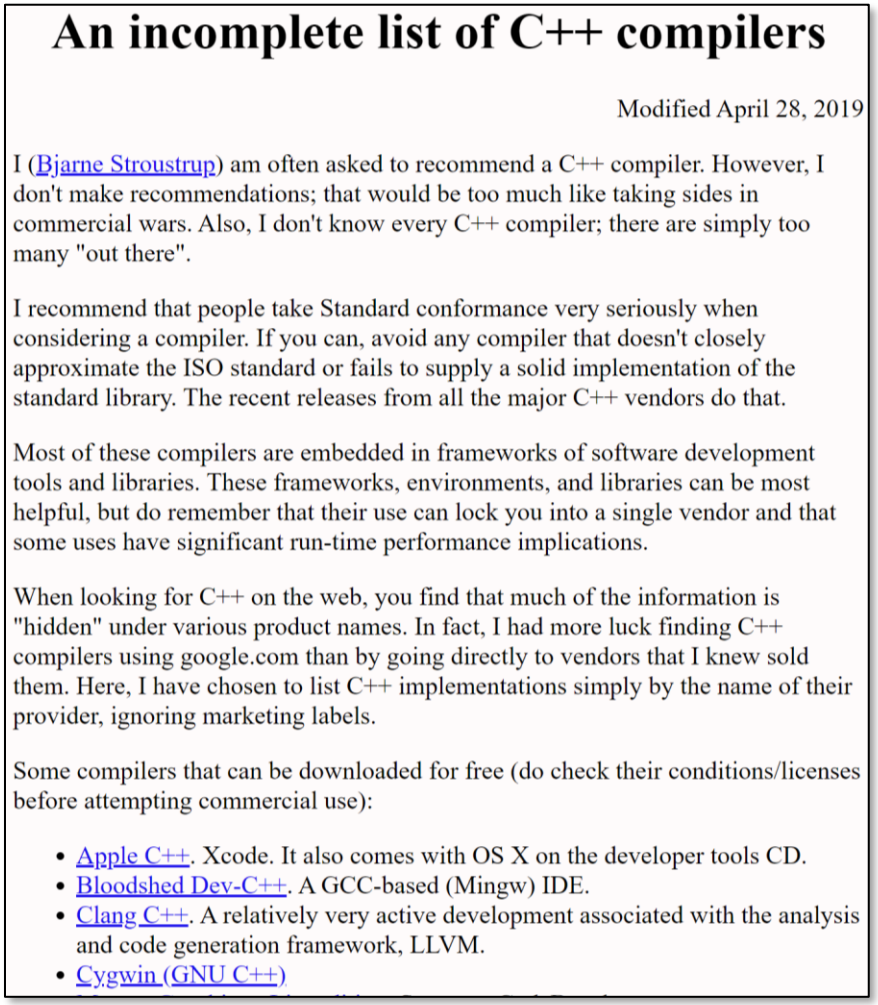
C++: From Translator to Native Compiler

- The original cfront translated C++ into C for portability
  - However, cfront output was hard to debug & was often inefficient for more advanced features



C++: From Translator to Native Compiler

- The original cfront translated C++ into C for portability
  - However, cfront output was hard to debug & was often inefficient for more advanced features
  - *Many* native host machine compilers now exist
    - e.g., GNU G++, LLVM Clang, IBM C++ Compiler, Microsoft Visual C++, Oracle Developer Studio, C++ Builder, etc.



An incomplete list of C++ compilers

Modified April 28, 2019

I (Bjarne Stroustrup) am often asked to recommend a C++ compiler. However, I don't make recommendations; that would be too much like taking sides in commercial wars. Also, I don't know every C++ compiler; there are simply too many "out there".

I recommend that people take Standard conformance very seriously when considering a compiler. If you can, avoid any compiler that doesn't closely approximate the ISO standard or fails to supply a solid implementation of the standard library. The recent releases from all the major C++ vendors do that.

Most of these compilers are embedded in frameworks of software development tools and libraries. These frameworks, environments, and libraries can be most helpful, but do remember that their use can lock you into a single vendor and that some uses have significant run-time performance implications.

When looking for C++ on the web, you find that much of the information is "hidden" under various product names. In fact, I had more luck finding C++ compilers using google.com than by going directly to vendors that I knew sold them. Here, I have chosen to list C++ implementations simply by the name of their provider, ignoring marketing labels.

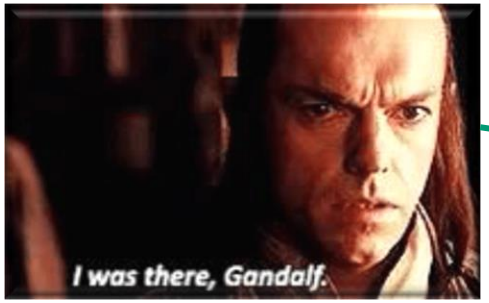
Some compilers that can be downloaded for free (do check their conditions/licenses before attempting commercial use):

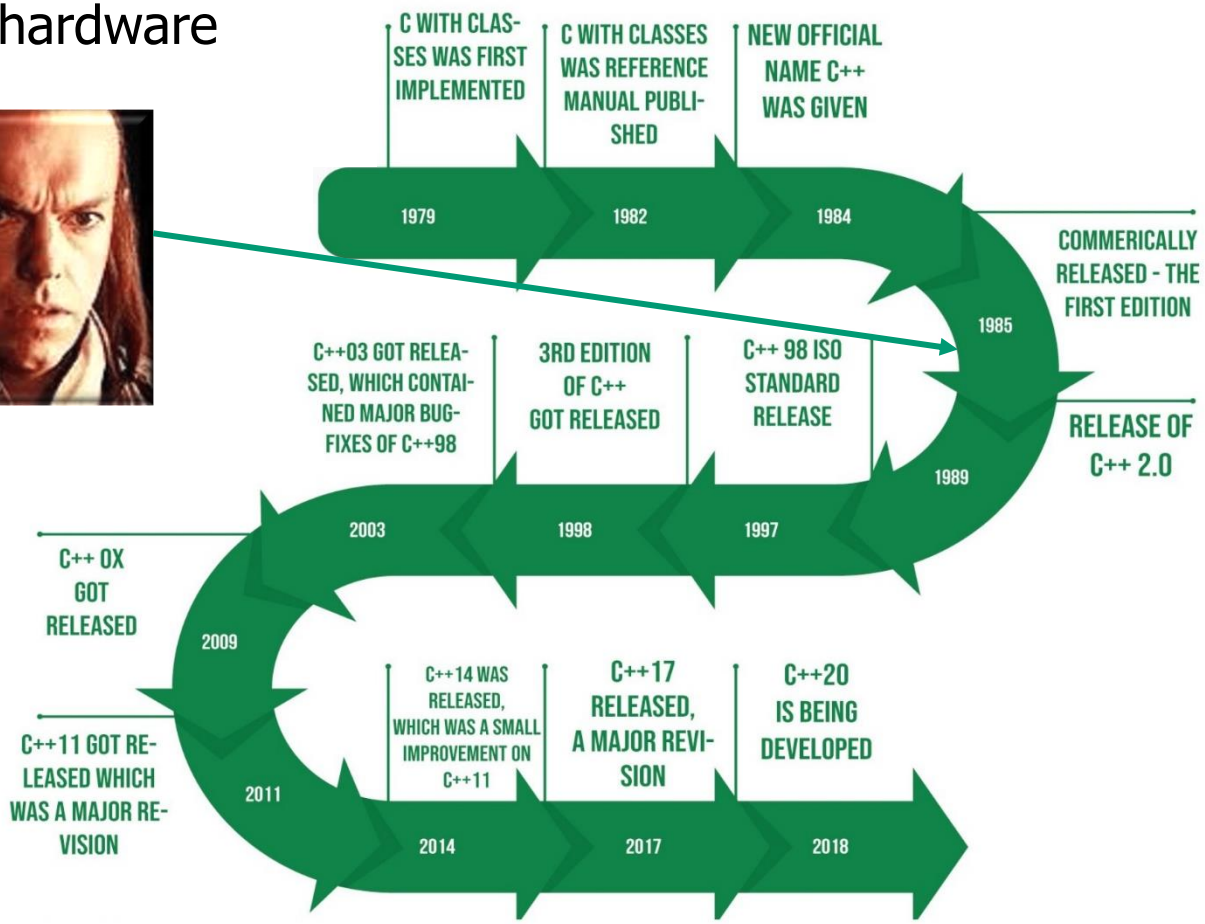
- Apple C++. Xcode. It also comes with OS X on the developer tools CD.
- Bloodshed Dev-C++. A GCC-based (Mingw) IDE.
- Clang C++. A relatively very active development associated with the analysis and code generation framework, LLVM.
- Cygwin (GNU C++)

See www.stroustrup.com/compilers.html

C++ Has Continued to Evolve

- C++ has continually evolved thruout its lifetime to address new programming paradigms, challenges, & hardware platforms





See en.wikipedia.org/wiki/C++#History

End of Overview of C++: Brief History