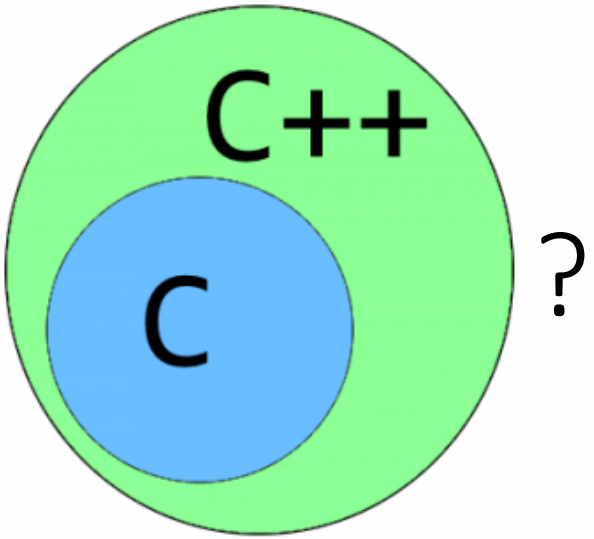
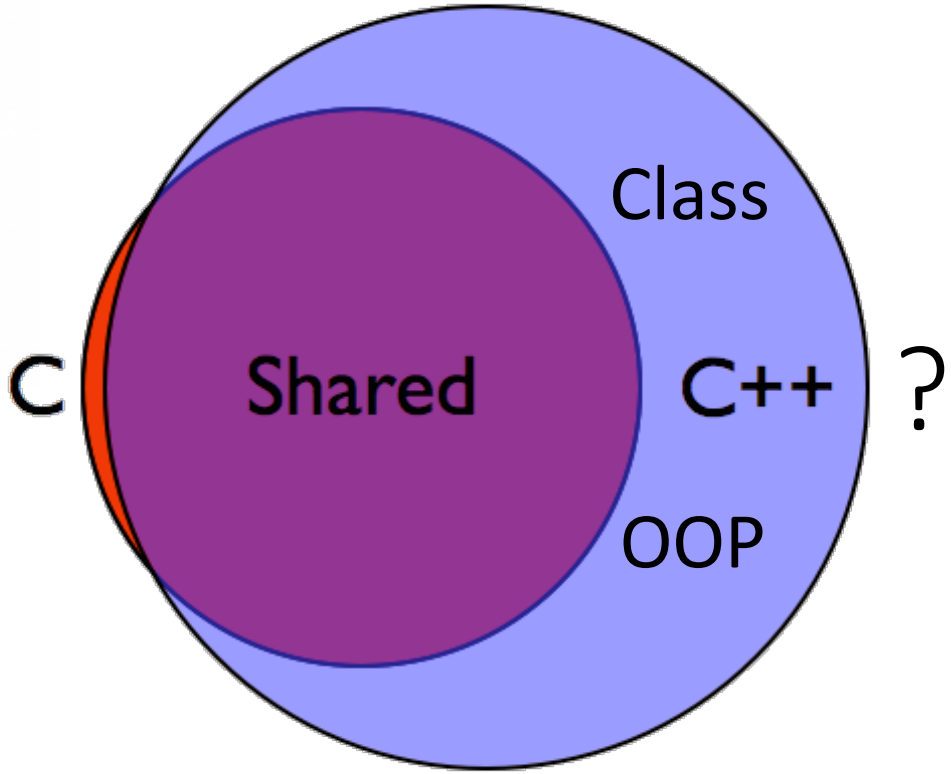

C++
Rixin Li

What is C++?

C++ is a superset of C





What is C++?

A multi-paradigm programming language

Class hierarchies

A hybrid language

Functional programming

Buffer overflows

Template meta-programming!

Classes

It's C!

Too big!

Embedded systems programming language

Generic programming

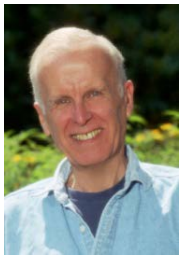
Low level!

An object-oriented programming language

A random collection of features

Credit: Stroustrup - C++11 Style Spring'13

The roots of C++



Domain-specific abstraction

Fortran

Cobol

General-purpose abstraction

Simula

Java

Direct mapping to hardware

Assembler

BCPL

C

C++

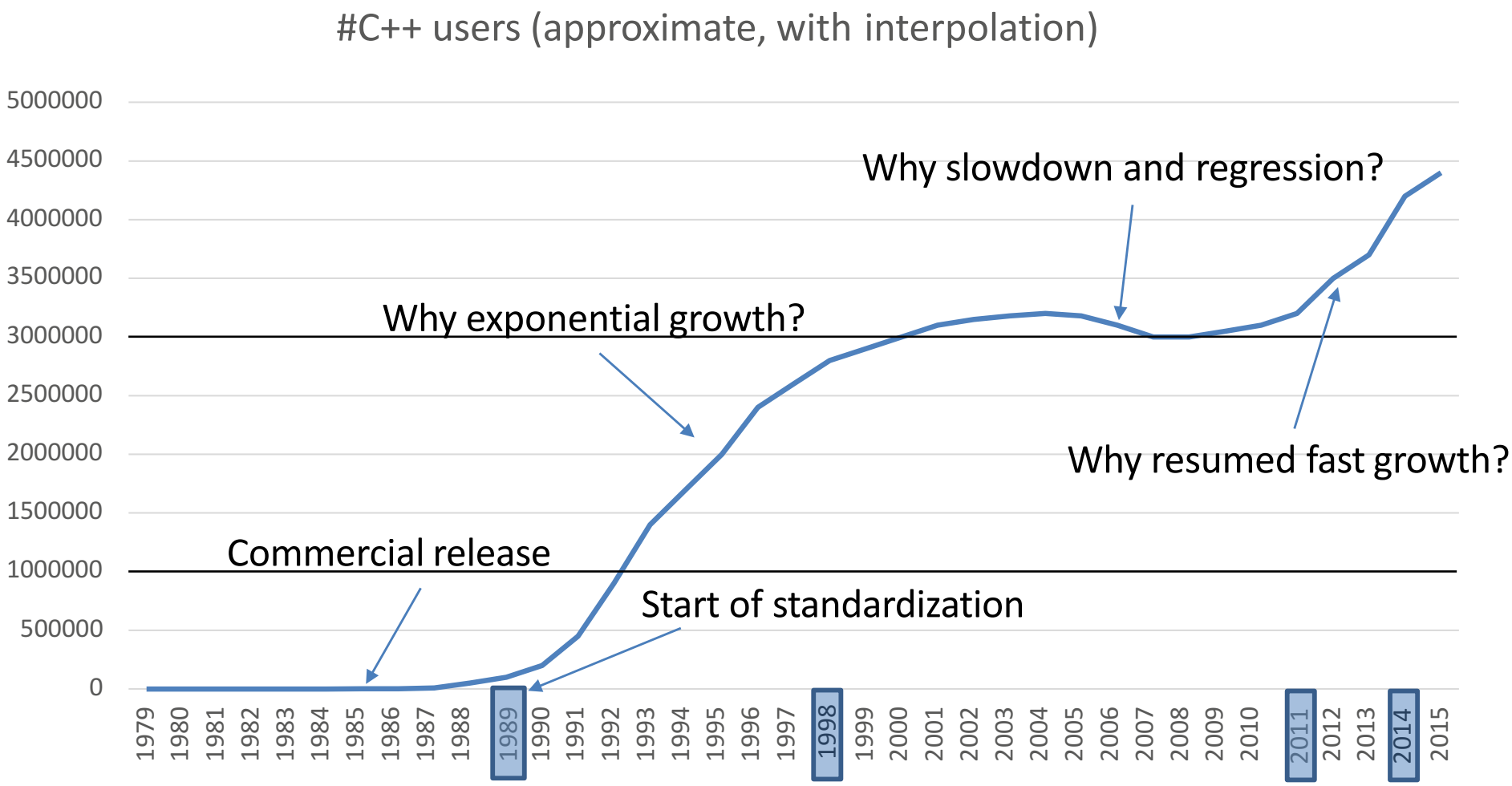
C++11

C++14

C#

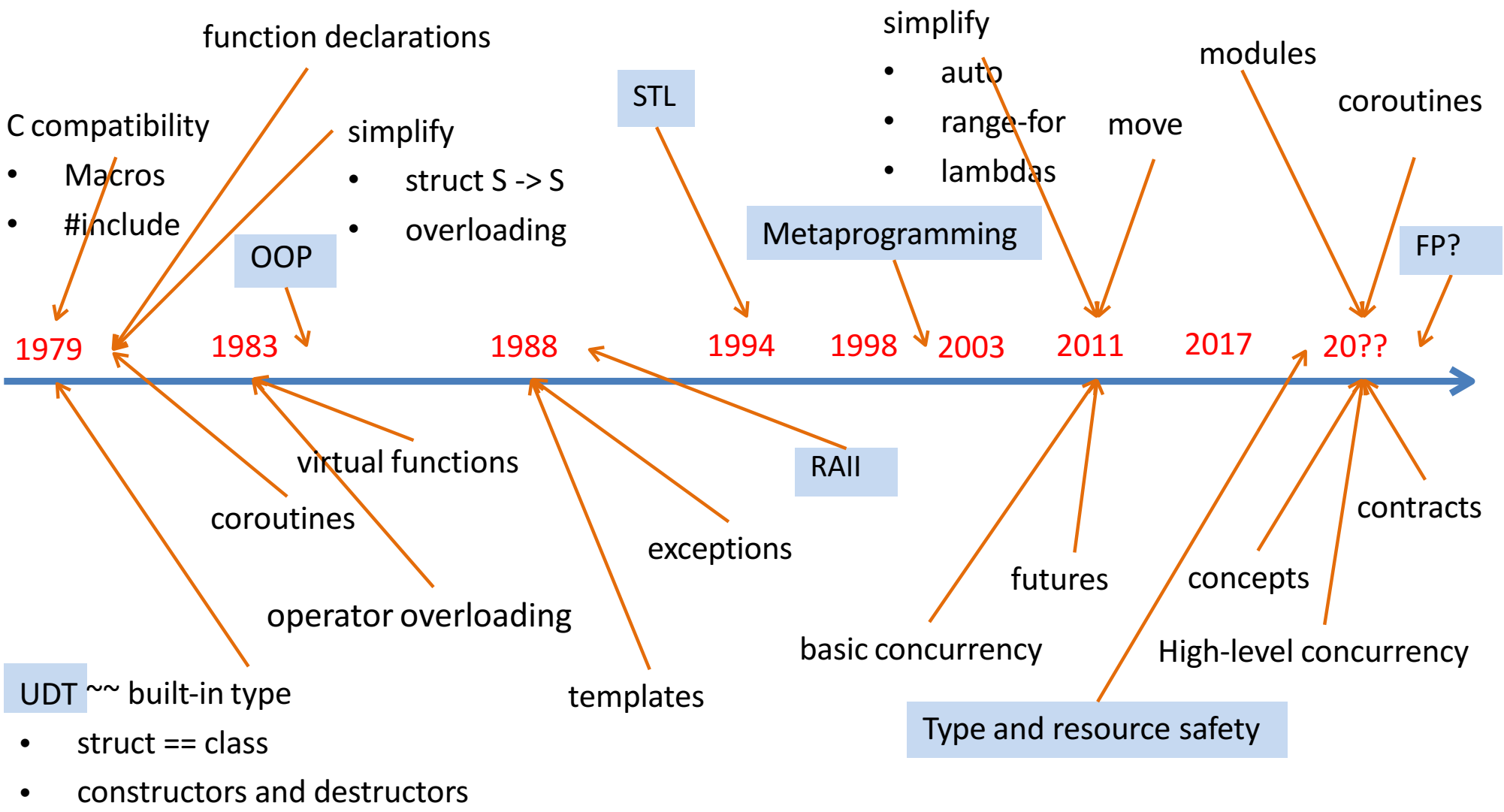
Credit: Stroustrup - CppCon'16

C++: Success



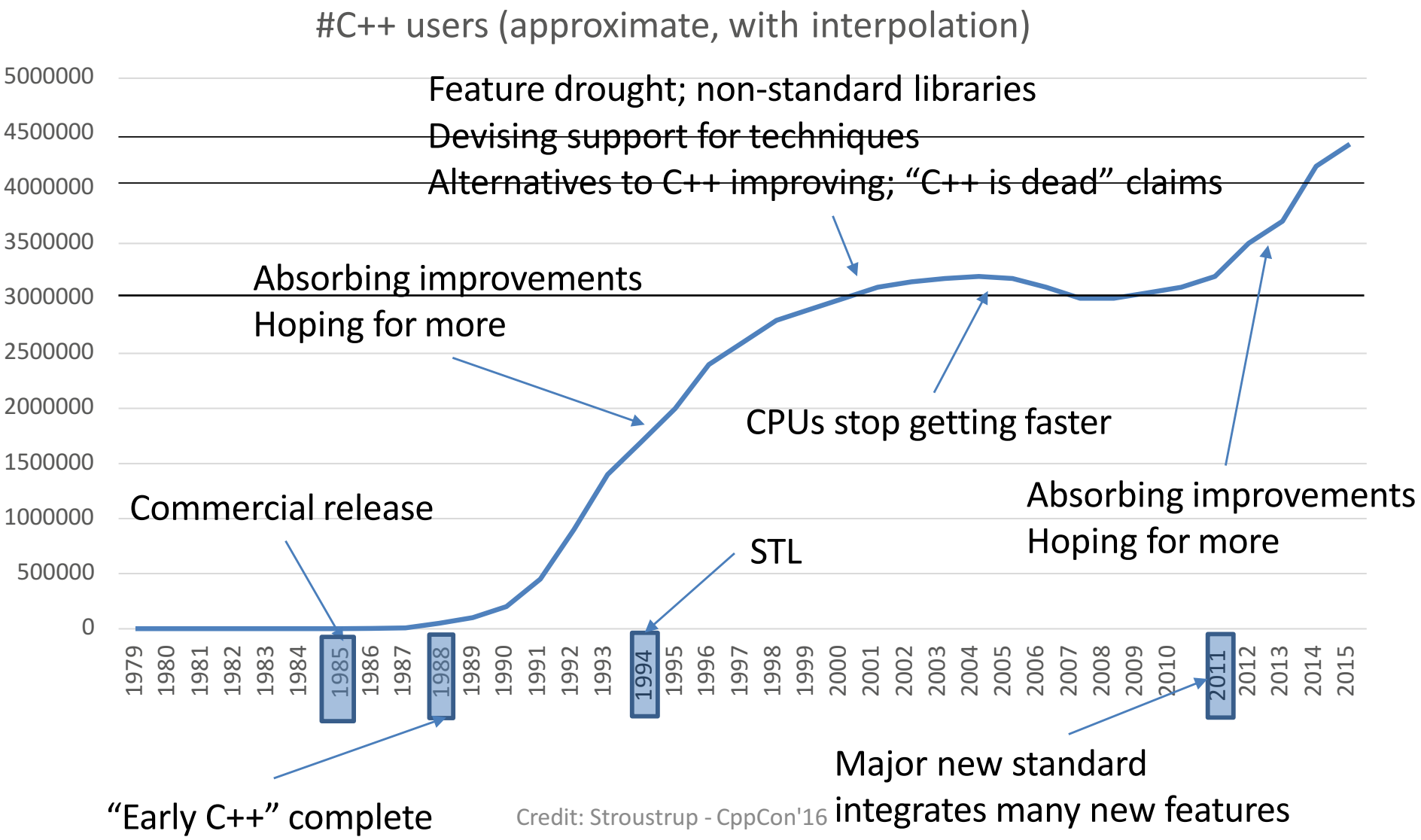
Credit: Stroustrup - CppCon'16

Major design decisions: Evolution is bursty



Credit: Stroustrup - CppCon'16

C++: Success



Credit: Stroustrup - CppCon'16

C++ in two lines

- Direct map to hardware
  - of instructions and fundamental data types
  - Initially from C
  - Future: use novel hardware better (caches, multicores, GPUs, FPGAs, SIMD, …)
- Zero-overhead abstraction
  - Classes, inheritance, generic programming, functional programming, …
  - Initially from Simula (where it wasn't zero-overhead)
  - Type- and resource-safety, concepts, modules, concurrency, …

Credit: Stroustrup - CppCon'16

Performance

- Direct map to hardware
  → low-level with **machine efficiency**

- Zero-overhead abstraction
  → high-level with **programmer efficiency**

Language features + **compiler** + **optimizer** deliver performance

C++ itself (syntactic sugar)	GCC	-O2 / -O3	Benchmark overall efficiency
Standard Libraries (STL)	Clang	-march=native	
Boost, Intel MKL, etc.	Intel C++	-ffast-math	e.g., qsort() vs. sort()
	etc.	etc.	

Syntactic Sugar Example

Deduction of the type with auto

- The compiler determines the type:

```
auto myString= "my String";                    // C++11
auto myInt= 5;                                 // C++11
auto myDouble= 3.14;                           // C++11
```

- Get a iterator on the first element of a vector:

```
vector<int> v;
vector<int>::iterator it1= v.begin();          // C++98
auto it2= v.begin();                           // C++11
```

- Definition of a function pointer:

```
int add(int a,int b){ return a+b; };
int (*myAdd1)(int,int)= add;                   // C++98
auto myAdd2= add;                              // C++11
myAdd1(2,3) == myAdd2(2,3);
```

Syntactic Sugar Example

Deduction of the type with decltype

- The compiler determines the type of an expression:

```
decltype("str") myString= "str";                    // C++11
decltype(5) myInt= 5;                               // C++11
decltype(3.14) myFloat= 3.14;                       // C++11
decltype(myInt) myNewInt= 2011;                     // C++11

int add(int a,int b){ return a+b; };
decltype(add) myAdd= add;  // (int)(*)(int, int)    // C++11
myAdd(2,3) == add(2,3);
```

Credit: Rainer Grimm C++11: An Overview

Syntactic Sugar Example

The range-based for-loop

- Simple iteration over a container:

```
vector<int> vec={1,2,3,4,5};
for (auto v: vec) cout << v << ",";        // 1,2,3,4,5,

unordered_map<string,int> um= {{"C++98",1998},{"C++11",2011}};
for (auto u:um) cout << u->first << ":" << u->second << "    ";
   // "C++11":2011  "C++98":1998
```

- Modifying the container elements by auto&:

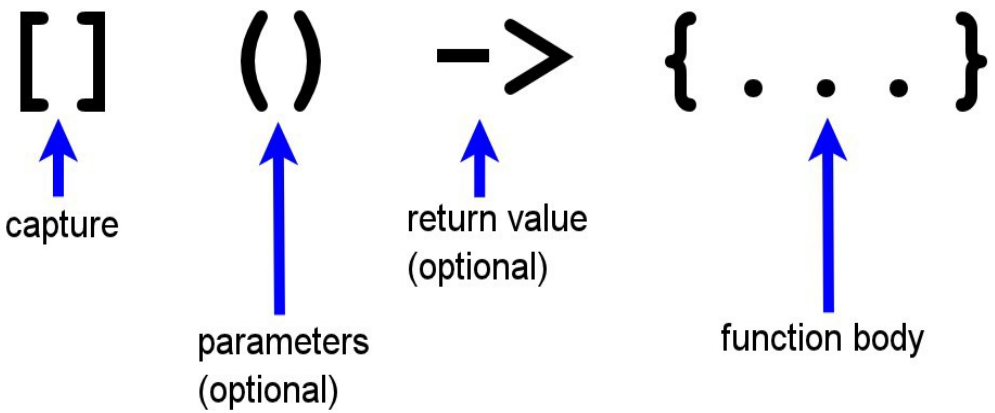
```
for (auto& v: vec) v *= 2;
for (auto v: vec) cout << v << " ,";        // 2,4,6,8,10,

string testStr{"Only for Testing."};
for (auto& c: testStr) c= toupper(c);
for (auto c: testStr) cout << c;   // "ONLY FOR TESTING."
```

Credit: Rainer Grimm C++11: An Overview

Syntactic Sugar Example

Lambda functions



- `[]`: captures the used variables per copy of per reference
- `()`: is required for parameters
- `->`: is required for sophisticated lambda functions
- `{}`: may include expressions and statements
- Sum the elements of a vector:

```
vector<int> vec={1,2,3,4,5,6,7,8,9,10};
auto sum = 0;
for_each(v.begin(),v.end(),[&sum](int x) {sum += x;});
```

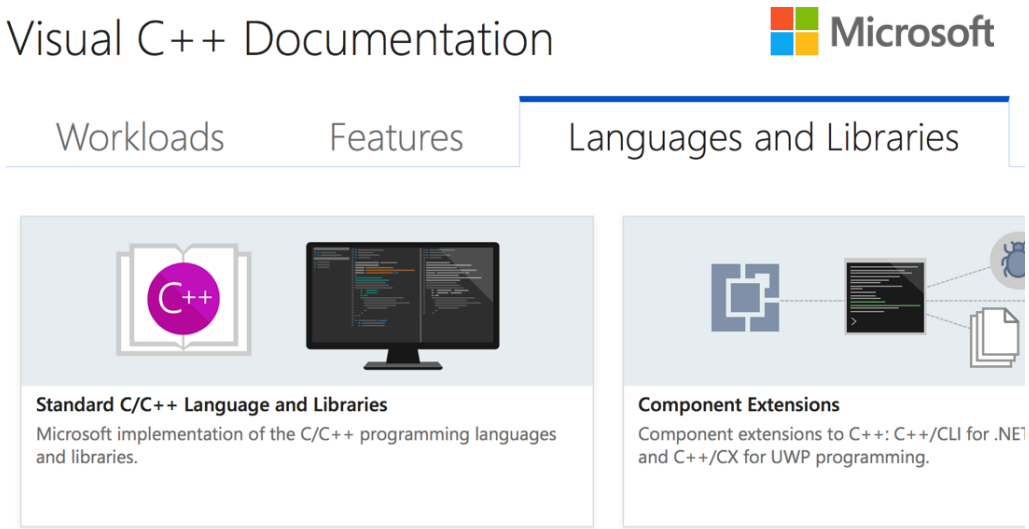
Credit: Rainer Grimm C++11: An Overview

There are still tons of new features + libraries.

- Lambda Expressions
- Automatic Type Deduction and `decltype`
- Uniform Initialization Syntax
- New Smart Pointer Classes
- Deleted and Defaulted Functions
- Delegating Constructors
- Rvalue References
- C++11 Standard Library (More Algorithms)
- Threading Library and Multithreading
- `nullptr`
- …
- See more on here

Online References

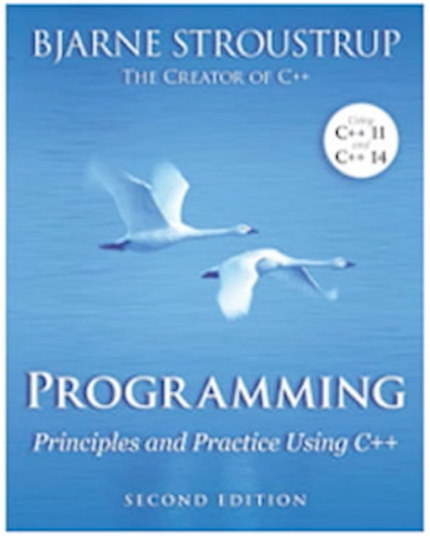
There are still tons of new features + libraries.



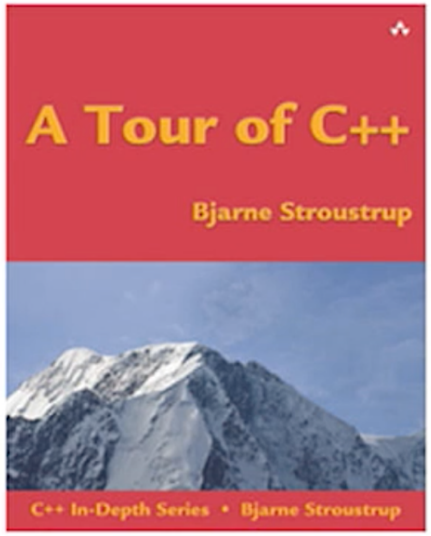
For better programmer efficiency, you need a good IDE.

Visual Studio, Xcode, Clion, Eclipse, etc.

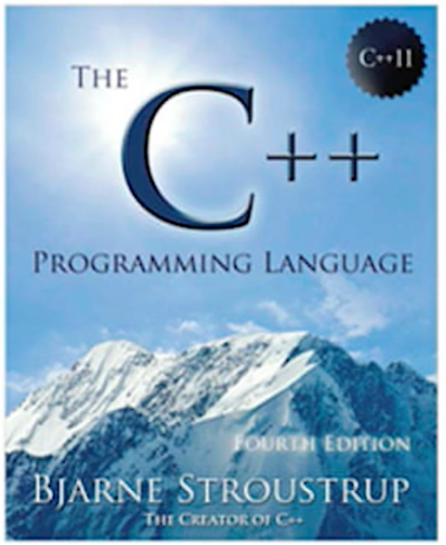
Recommended Books



A programming text book aimed at beginners who want eventually to become professionals; includes simple graphics



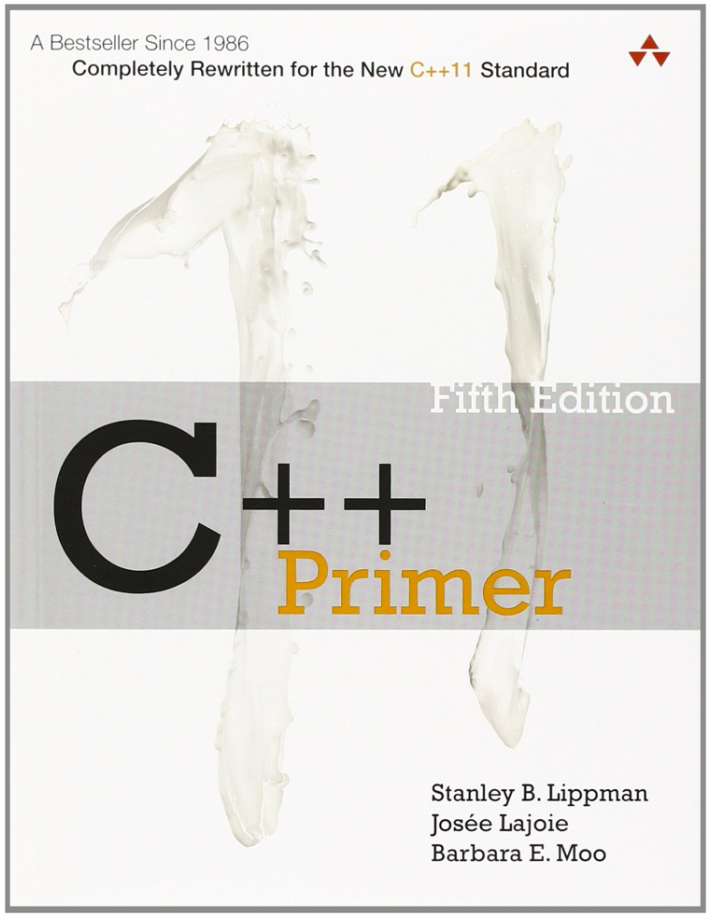
A brief - 180 page - tour of the C++ programming language and its standard library for experienced programmers

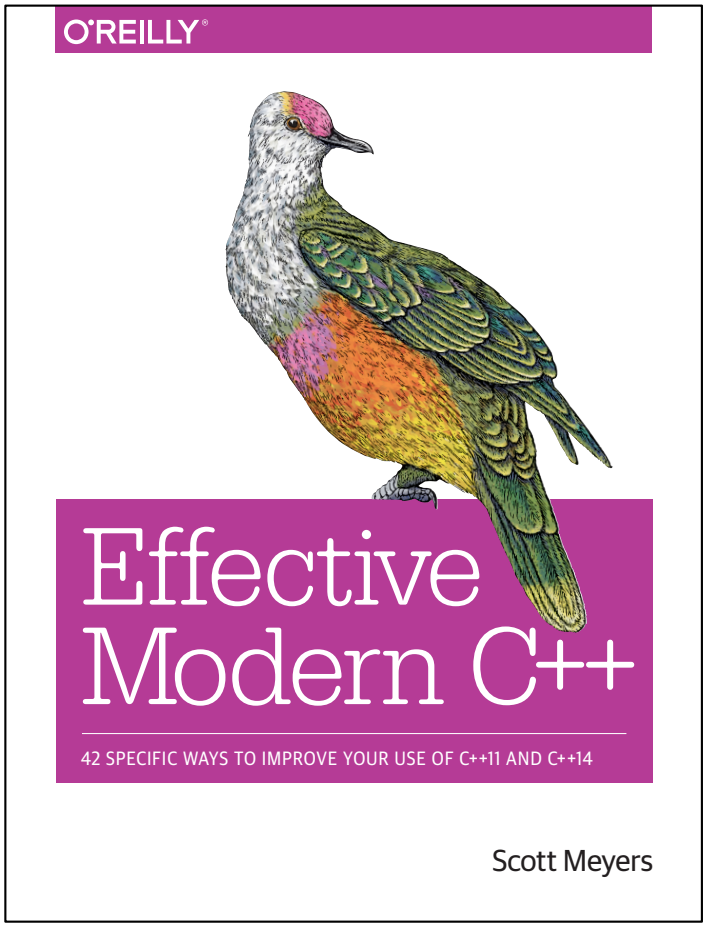


An exhaustive description of the C++ Programming language, its standard library, and fundamental techniques for experienced programmers

Credit: Stroustrup - Teaching C++ - CppCon'17

Recommended Books





Appendix: C++20 Draft