2022 Spring

Advanced Data Structure

Zhengwei QI, Mingkai Dong

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Teachers & TAs

- Zhengwei QI (戚正伟)
  - qizhwei@sjtu.edu.cn
- Mingkai Dong（董明凯)
  - mingkaidong@sjtu.edu.cn
- TAs
  - 陈沛东: 492556292@qq.com
  - 姬浩迪: 575650866@qq.com
  - 王笑然: 1127858301@qq.com
  - 顾翼成: guyicheng98@sjtu.edu.cn

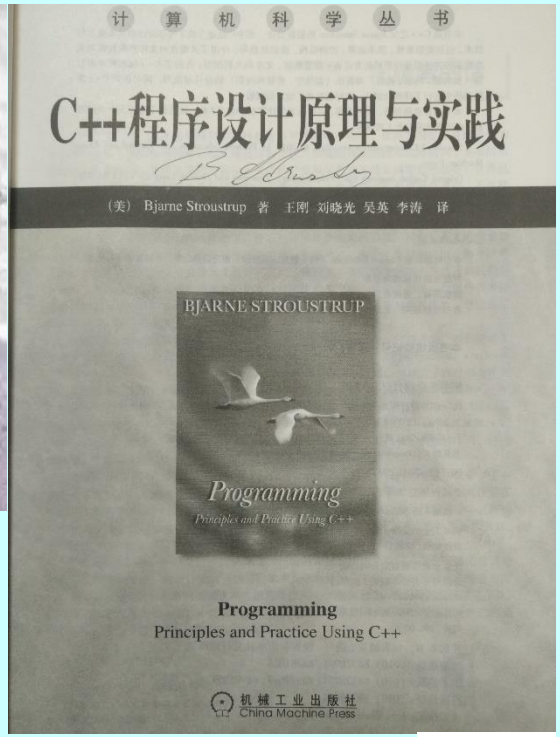
Prerequisites & Textbooks

- Textbooks
  - 数据结构（C++语言版）第三版, 邓俊辉, 清华大学出版社, 2013年9月
  - 《数据结构》讲义, 待出版
- Reference books
  - Roberts, Eric S. Programming Abstractions in C++. ISBN 978-0133454840.

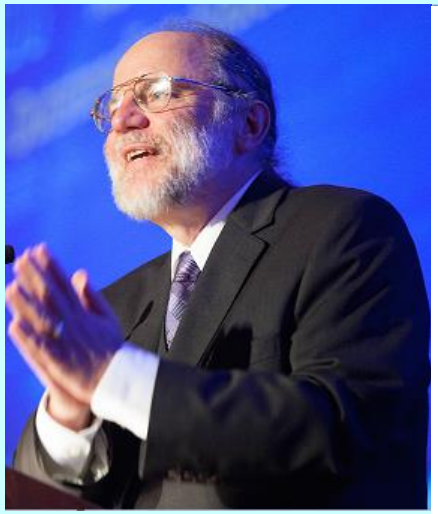
Rough course outline

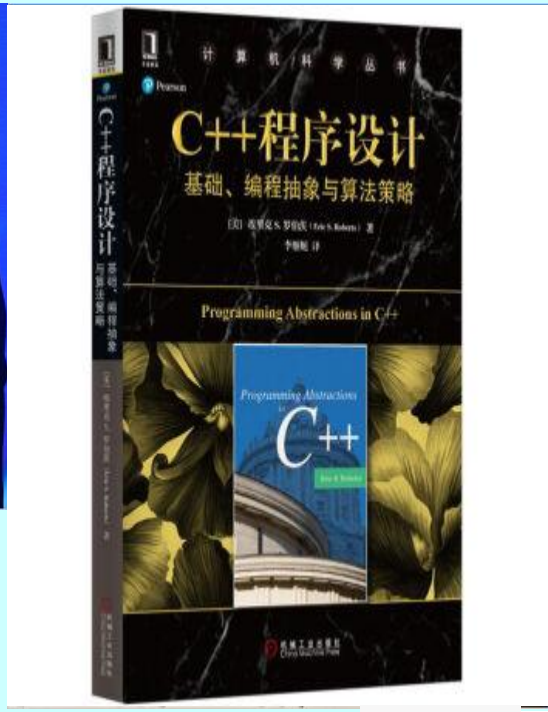
- Advanced Data Structure
  - Advanced data structures, including advanced search trees, graphs, collections, heaps, etc.. And through designing an integrated software system as several projects, training the ability to use a variety of data structures.

数据结构-相关书籍



$47.85





¥117.40

PPP：**Programming : principles and practice using C++**

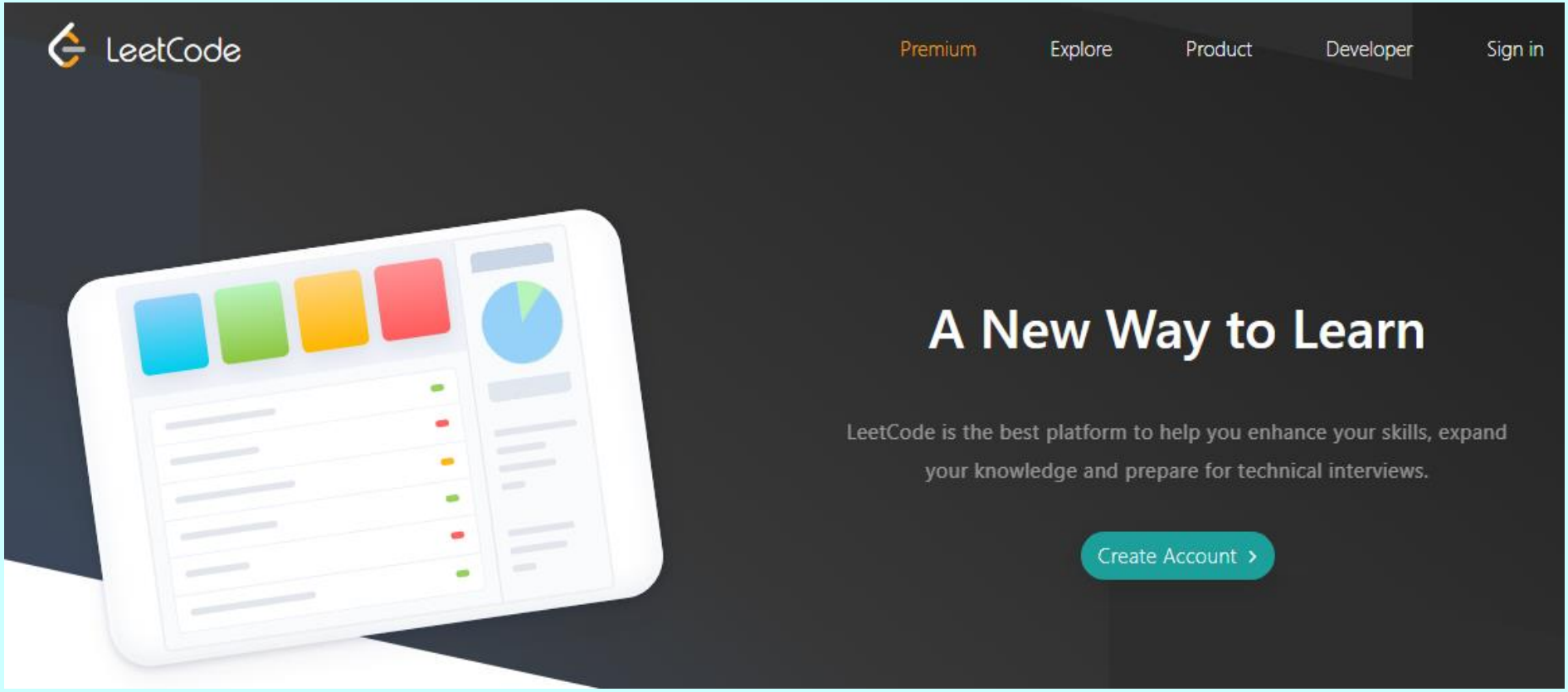
PA：**Programming Abstractions in C++.**

MOOC

https://www.xuetangx.com/course/THU08091000384/10322765

OJ-https://www.luogu.org/

OJ- https://leetcode.com/



Course Web

- https://oc.sjtu.edu.cn/courses/39954

Related Top Conferences

- POPL - Symposium on Principles of Programming Languages
- PLDI - SIGPLAN Conference on Programming Language Design and Implementation
- ECOOP - European Conference on Object-Oriented Programming
- CAV - Computer Aided Verification
- OOPSLA - Conference on Object-Oriented Programming Systems, Languages, and Applications
- ICSE - International Conference on Software Engineering

Paper Example

William Pugh. 1990. Skip lists: a probabilistic alternative to balanced trees. Commun. ACM 33, 6 (June 1990), 668–676.

Skip Lists: A Probabilistic Alternative to Balanced Trees

Skip lists are a data structure that can be used in place of balanced trees. Skip lists use probabilistic balancing rather than strictly enforced balancing and as a result the algorithms for insertion and deletion in skip lists are much simpler and significantly faster than equivalent algorithms for balanced trees.

William Pugh

Paper Example

Bigtable: A Distributed Storage System for Structured Data

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach
Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber

{fay,jeff,sanjay,wilsonh,kerr,m3b,tushar,fikes,gruber}@google.com

Google, Inc.

The manner in which Bigtable uses memtables and SSTables to store updates to tablets is analogous to the way that the Log-Structured Merge Tree [26] stores updates to index data. In both systems, sorted data is buffered in memory before being written to disk, and reads must merge data from memory and disk.

[26] O'Neil, P., Cheng, E., Gawlick, D., and O'Neil, E. The log-structured merge-tree (LSM-tree). *Acta Inf.* *33*, 4 (1996), 351–385.

Assignments

- Homework： Every week.
- Exercise Course： Project & Labs
  - 0. 简单的热身Lab
  - 1. LSM Tree Project (需要答辩)
  - 2. 高级二叉树的 Lab
  - 3. 图算法（最短路等） Lab
  - 4. 并行编程 Lab

Survey

- Which kind of C++ IDE do you use?
  - A. Visual Studio
  - B. Visual Studio Code
  - C. IDE for Windows (i.e., Qt Creator)
  - D. IDE for Linux (i.e., Code::Blocks)
  - E. IDE for Mac (i.e., Xcode)
  - F. no GUI IDE (i.e., vim/emacs)

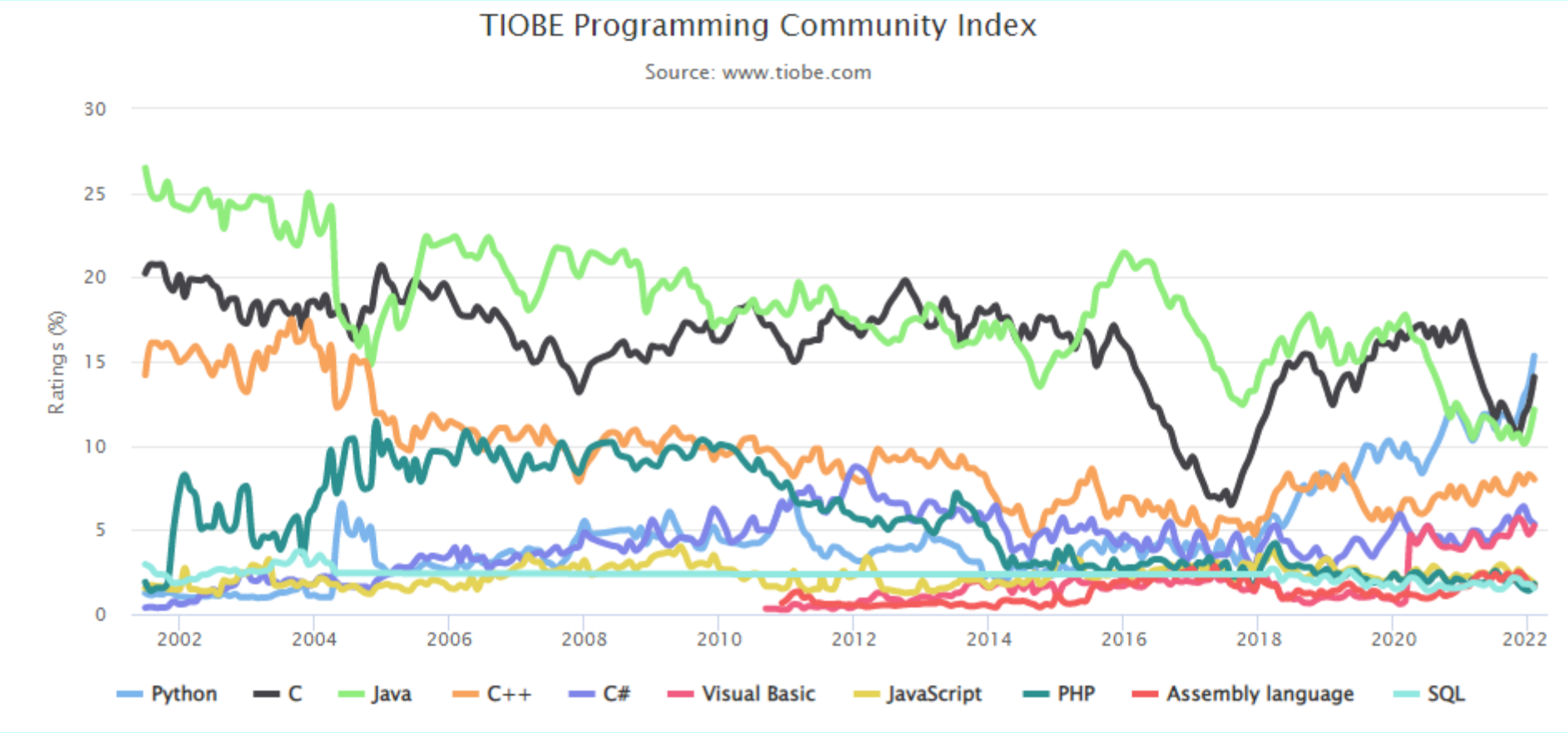
Grading

- Grades
  - Project/Labs : 40% (including lab reports 10%)
  - Final exam: 60%
- Late and re-grade policies
  - Late submissions of project/labs will receive partial or no credit.
  - Re-grade must be submitted within ONE WEEK
- Cheating
  - NOT tolerated!
  - Zero for the assignment and other possible repercussions

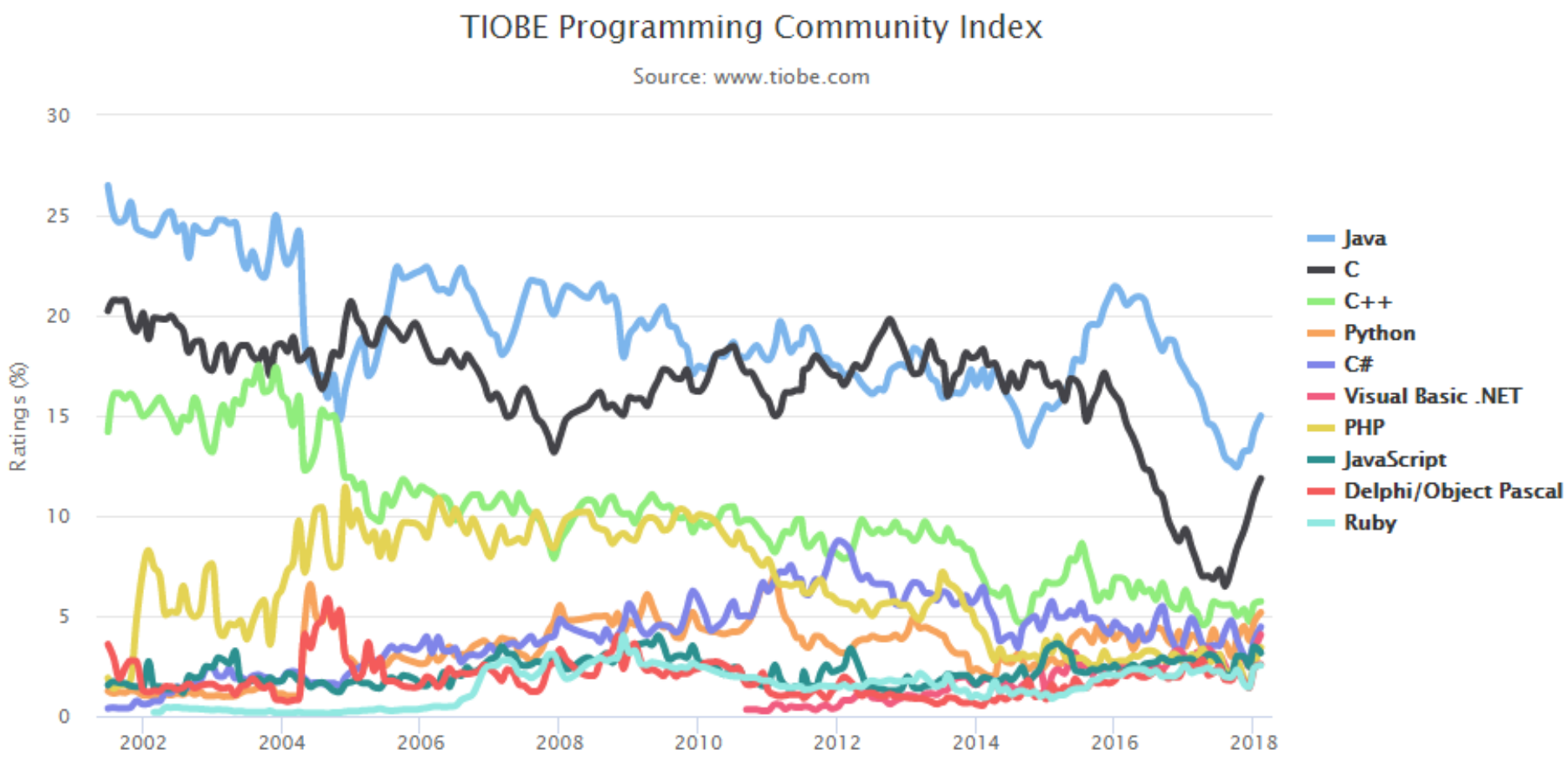
Related Courses

- MIT 6.851: Advanced Data Structures
- Stanford: **cs106 A/B/L** Standard C++ Programming Laboratory (*)
- Berkeley CS 61B: Data Structure
- CMU CS 15-121: Introduction to Data Structures
- CMU CS 15-211: Fundamental Data Structures and Algorithms
- UIUC CS 225: Data Structures and Programming Principles

TIOBE Index for 2022



Compare: TIOBE Index for 2018



C++ 11/14/17/20/23?

C++14 core language features

C++14 feature	Paper(s)	GCC	Clang	MSVC	Apple Clang	EDG eccp	Intel C++	IBM XLC++	Sun/Oracle C++	Embarcadero C++ Builder	Cray	Nvidia HPC C++ (ex Portland Group/PGI)	Nvidia nvcc	[Collapse]
Tweaked wording for contextual conversions	N3323	4.9	3.4	18.0*	Yes	4.9	16.0	13.1.2*	5.15	10.3	8.6	16.1	9.0	
Binary literals	N3472	4.3 (GNU) 4.9	2.9	19.0 (2015)*	Yes	4.10	11.0	13.1.2*	5.14	10.3	8.6	2015	9.0	
`decltype(auto)`, Return type deduction for normal functions	N3638	4.8 (partial)* 4.9	3.3 (partial)* 3.4	19.0 (2015)*	Yes	4.9	15.0	13.1.2*	5.15	10.3	8.6	16.1	9.0	
Initialized/Generalized lambda captures (init-capture)	N3648	4.5 (partial) 4.9	3.4	19.0 (2015)*	Yes	4.10	15.0	16.1.1*	5.15	10.3	8.6	16.1	9.0	
Generic lambda expressions	N3649	4.9	3.4	19.0 (2015)*	Yes	4.10	16.0	13.1.2*	5.15	10.3	8.6	16.1	9.0	
Variable templates	N3651	5	3.4	19.0 (Update 2)*	Yes	4.11	17.0	13.1.2*	5.15	10.3	8.6	17.4	9.0	
Extended constexpr	N3652	5	3.4	19.10*	Yes	4.11	17.0	13.1.2*	5.15	10.3	8.6	17.4	9.0	
Aggregates with default member initializers	N3653	5	3.3	19.10*	Yes	4.9	16.0	16.1.1*	5.14	10.3	8.6	16.1	9.0	
Omitting/extending memory allocations	N3664	N/A	3.4	N/A	Yes	N/A	N/A	N/A	N/A	10.3	8.6	17.4	N/A	
`[[deprecated]]` attribute	N3760	4.9	3.4	19.0 (2015)*	Yes	4.9	15.0* 16.0	13.1.2*	5.14	10.3	8.6	16.1	9.0	
Sized deallocation	N3778	5	3.4	19.0 (2015)*	Yes	4.10.1	17.0	16.1.1*	5.14	10.3	8.6	16.1		
Single quote as digit separator	N3781	4.9	3.4	19.0 (2015)*	Yes	4.10	16.0	13.1.2*	5.14	10.3	8.6	2015	9.0	

https://en.cppreference.com/w/cpp/compiler_support

So what is programming?

- Conventional definitions
  - Telling a **very** fast moron *exactly* what to do
  - A plan for solving a problem on a computer
  - Specifying the order of a program execution
    - But modern programs often involve millions of lines of code
    - And manipulation of data is central
- Definition from another domain (academia)
  - A … program is an organized and directed accumulation of resources to accomplish specific … objectives …
    - Good, but no mention of actually doing anything
- The definition we'll use
  - Specifying the structure and behavior of a program, and testing that the program performs its task correctly and with acceptable performance
    - Never forget to check that "it" works
- Software == one or more programs

From Stroustrup/Programming/2015

Programming

- Programming is fundamentally simple
  - Just state what the machine is to do
- So why is programming hard?
  - We want "the machine" to do complex things
    - And computers are nitpicking, unforgiving, dumb beasts
  - The world is more complex than we'd like to believe
    - So we don' t always know the implications of what we want
  - "Programming is understanding"
    - When you can program a task, you understand it
    - When you program, you spend significant time trying to understand the task you want to automate
  - Programming is part practical, part theory
    - If you are just practical, you produce non-scalable unmaintainable hacks
    - If you are just theoretical, you produce toys

现代软件是复杂工程

- Linux 内核是迄今为止最大的协同软件项目。
  - 在 2016 年，超过 450 家不同公司的 4000 多名开发者对该项目做出了贡献。
  - 该项目共有 6 个版本，每个版本都包含 12000 到 16000 项不同的更改。
  - 在 2016 年底，Linux 内核的规模刚好超过 56000 个文件，其中包括 2200 万行代码、编译脚本和文档（内核版本 4.9）
- 虽然 Linux 内核包含其支持的所有不同芯片架构和硬件驱动程序的代码，但各个系统仅运行一小部分代码库。
  - 一台普通的笔记本电脑需要使用来自 5000 个文件的大约 200 万行内核代码才能正常运行；
  - 而 Pixel 手机需要使用来自 6000 个文件的 320 万行内核代码才能正常运行（因为 SoC 的复杂性有所增加）。

Source: https://source.android.google.cn/devices/architecture/kernel?hl=zh-cn

Maintainability

```
OS_EVENT  *OSSemCreate (INT16U cnt)
{
#if OS_CRITICAL_METHOD == 3                          /* Allocate storage for CPU status register */
    OS_CPU_SR  cpu_sr;
#endif
    OS_EVENT  *pevent;


    if (OSIntNesting > 0) {                          /* See if called from ISR ...               */
        return ((OS_EVENT *)0);                      /* ... can't CREATE from an ISR             */
    }
    OS_ENTER_CRITICAL();
    pevent = OSEventFreeList;                        /* Get next free event control block        */
    if (OSEventFreeList != (OS_EVENT *)0) {          /* See if pool of free ECB pool was empty   */
        OSEventFreeList = (OS_EVENT *)OSEventFreeList->OSEventPtr;
    }
    OS_EXIT_CRITICAL();
    if (pevent != (OS_EVENT *)0) {                   /* Get an event control block               */
        pevent->OSEventType = OS_EVENT_TYPE_SEM;
        pevent->OSEventCnt  = cnt;                   /* Set semaphore value                      */
        pevent->OSEventPtr  = (void *)0;             /* Unlink from ECB free list                */
        OS_EventWaitListInit(pevent);                /* Initialize to 'nobody waiting' on sem.   */
    }
    return (pevent);
} ? end OSSemCreate ?

/*$PAGE*/
```

C++ Style Guide

- Google C++ Style Guide
- JPL Coding Standard
- Stanford Style:
  - [http://web.stanford.edu/class/cs106b/resources/style-guide/]
- PPP Style:
  - [http://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines]
- SJTU Style
  - [https://github.com/sjtu-se117/CppGuidelines/blob/master/CppGuidelines.md]

Example

```
// bad
main
|
+-- function1
    |
    +-- function2
        |
        +-- function3
            |
            +-- function4
            |
            +-- function5
                |
                +-- function6
```

```
// good
main
|
+-- function1
|
+-- function2
|   |
|   +-- function3
|       |
|       +-- function4
|
+-- function5
|   |
|   +-- function6
```

http://web.stanford.edu/class/cs106b/resources/style-guide/

A good program

- Correctness
- Maintainability
- Performance
- Security
- Scalability
- Availability
- Power-efficiency

Security：Example insider attack

- Hidden trap door in Linux, Nov 2003
  - Allows attacker to take over a computer
  - Practically undetectable change
  - Uncovered by anomaly in CVS usage
- Inserted line in wait4()

```
if ((options == (__WCLONE|__WALL)) && (current->uid = 0))
            retval = -EINVAL;
```

不是=而应该是==

  - Looks like a standard error check
  - Anyone see the problem?

See: http://lwn.net/Articles/57135/

The average of two unsigned integers

```
unsigned average(unsigned a, unsigned b)
{
    return (a + b) / 2;
}
```

可能存在overflow的情况

Source: https://devblogs.microsoft.com/oldnewthing/20220207-00/?p=106223

The average of two unsigned integers

```
unsigned average(unsigned a, unsigned b)
{
    return (a + b) / 2;
}
```

Overflow: average(0x80000000U, 0x80000000U)

The average of two unsigned integers

```
unsigned average(unsigned low, unsigned high)
{
    return low + (high - low) / 2;
}
```

Overflow? average(0x80000000U, 0x80000000U)

The average of two unsigned integers

```
unsigned average(unsigned a, unsigned b)
{
    return (a / 2) + (b / 2) + (a & b & 1);
}
```

a&b&1 是对前两项的修正，原因是/2操作会向下取整，可能会有误差

[U.S. patent expired in 2016]

The average of two unsigned integers

```
unsigned average(unsigned a, unsigned b)
{
    return (a & b) + (a ^ b) / 2;
}
```

SWAR, which stands for “SIMD within a register”

Swap 1.0

```
// C Style
void swap (int *x,int *y)

    {
        int temp;
        temp=*x;
        *x=*y;
        *y=temp;
    }
```

```
// C++ optimization
void swap (int & __restrict a,      int & __restrict b)
{           a ^= b; //a=a^b
            b ^= a;
              //b=b^(a^b)=b^a^b=b^b^a=0^a=a
            a ^= b;
              //a=(a^b)^a=a^b^a=a^a^b=0^b=b

}
```

http://blog.csdn.net/ryfdizuo/article/details/6435847

Swap 2.0

宏的定义方式

```
// Macro Style

#include <iostream.h>
#define SWAP(t,x,y) \
{\
   t temp = *y;\
          *y = *x;\
          *x = temp;\
}
main()
{    int a = 10, b = 5;
    SWAP(int,&a,&b)
  cout << a << endl << b<<endl;
}
```

```
// C++ 11 optimization
#define _swap(x, y)                \
    do                             \
    {                              \
        decltype(x) t = (x);       \
        (x) = (y);                 \
        (y) = (t);                 \
    } while (false);

    int main()
    {
        int a = 0;
        int b = 1;
        _swap(a, b);
        return 0;
    }
```

Swap 3.0

```
namespace std {
        template<typename T>
        // typical implementation of std::swap;
        void swap(T& a, T& b) // swaps a's and b's values
        {
                T temp(a);  //copy ctor
                a = b;   //assignment ctor
                b = temp;
        }
}
```

Swap 4.0 (from C++11)

```
template <class T> swap(T& a, T& b)
 {
        T tmp(std::move(a)); // move a to tmp
        a = std::move(b);// move b to a
        b = std::move(tmp);   // move tmp to b
}
```

How many copies?

A good program

- Correctness
- Maintainability
- Performance
- Security
- Scalability
- Availability
- Power-efficiency

Guido van Rossum

Python vs. C++

Bjarne Stroustrup

- Interpreted
- Very high level language
- Writing code is quick and easy
- *Python* code runs more slowly, but call precompiled C/C++ Libraries
- Dynamic type system

- Compile and Link
- Low-level language (but standardized higher-level libraries available)
- Writing code takes longer
- Code runs very fast
- Static type system

test.py

```python
def f(n):
    if n == 1:
        return 1
    else:
        #print n
        return n*f(n-1)

print f(100)
```

python test.py

test.cpp

```cpp
#include <iostream>
#include<gmpxx.h>
using namespace std;

mpz_class  f(int n){
    if (n ==  1)
        return 1;
    else
        return n*f(n-1);
}

int main()
{
    cout << f(100) << endl;
}
```

g++ test.cpp -o test -lgmpxx -lgmp

test.py

```python
def f(n):
    if n == 1:
        return 1
    else:
        #print n
        return n*f(n-1)

print f(100)
```

```
qizhwei@DESKTOP-5VKIBV2:~/test$ time python test.py
93326215443944152681699238856266700490715968264381621468592963895217599993229915608941463976156518286253697920827223758251185210916864000000000000000000000000

real    0m0.026s
user    0m0.000s
sys     0m0.016s
```

python test.py

test.cpp

```cpp
#include <iostream>
#include<gmpxx.h>
using namespace std;

mpz_class  f(int n){
    if (n ==  1)
        return 1;
    else
        return n*f(n-1);
}
```

```
qizhwei@DESKTOP-5VKIBV2:~/test$ time ./test
93326215443944152681699238856266700490715968264381621468592963895217599993229915608941463976156518286253697920827223758251185210916864000000000000000000000000

real    0m0.008s
user    0m0.016s
sys     0m0.000s
```

```cpp
    cout << f(100) << endl;
}
```

g++ test.cpp -o test -lgmpxx -lgmp

	Energy
(c) C	1.00
(c) Rust	1.03
(c) C++	1.34
(c) Ada	1.70
(v) Java	1.98
(c) Pascal	2.14
(c) Chapel	2.18
(v) Lisp	2.27
(c) Ocaml	2.40
(c) Fortran	2.52
(c) Swift	2.79
(c) Haskell	3.10
(v) C#	3.14
(c) Go	3.23
(i) Dart	3.83
(v) F#	4.13
(i) JavaScript	4.45
(v) Racket	7.91
(i) TypeScript	21.50
(i) Hack	24.02
(i) PHP	29.30
(v) Erlang	42.23
(i) Lua	45.98
(i) Jruby	46.54
(i) Ruby	69.91
(i) Python	75.88
(i) Perl	79.58

	Time
(c) C	1.00
(c) Rust	1.04
(c) C++	1.56
(c) Ada	1.85
(v) Java	1.89
(c) Chapel	2.14
(c) Go	2.83
(c) Pascal	3.02
(c) Ocaml	3.09
(v) C#	3.14
(v) Lisp	3.40
(c) Haskell	3.55
(c) Swift	4.20
(c) Fortran	4.20
(v) F#	6.30
(i) JavaScript	6.52
(i) Dart	6.67
(v) Racket	11.27
(i) Hack	26.99
(i) PHP	27.64
(v) Erlang	36.71
(i) Jruby	43.44
(i) TypeScript	46.20
(i) Ruby	59.34
(i) Perl	65.79
(i) Python	71.90
(i) Lua	82.91

	Mb
(c) Pascal	1.00
(c) Go	1.05
(c) C	1.17
(c) Fortran	1.24
(c) C++	1.34
(c) Ada	1.47
(c) Rust	1.54
(v) Lisp	1.92
(c) Haskell	2.45
(i) PHP	2.57
(c) Swift	2.71
(i) Python	2.80
(c) Ocaml	2.82
(v) C#	2.85
(i) Hack	3.34
(v) Racket	3.52
(i) Ruby	3.97
(c) Chapel	4.00
(v) F#	4.25
(i) JavaScript	4.59
(i) TypeScript	4.69
(v) Java	6.01
(i) Perl	6.62
(i) Lua	6.72
(v) Erlang	7.20
(i) Dart	8.64
(i) Jruby	19.84

Next

- Chapter 9: 跳转表 (**数据结构（C++语言版）第三版**)