COMP2113 / ENGG1340

Dr. Heming Cui
https://www.cs.hku.hk/people/profile.jsp?teacher=heming

Department of Computer Science, HKU

September, 2019

My short bio

- PhD from Columbia University, NY, USA; joined HKU in Jan 2015
- Research interests: distributed systems, cloud computing, bigdata &AI systems, and blockchains; focus on reliability&security
- Received several world-wide important research awards
    - Croucher Innovation Award in 2016
    - Three HK government research grants (GRF)
    - Huawei Innovation Research Program (HIRP) Open 2017
    - HIRP Flagship 2018 (blockchain areas)

Teaching Team

Instructors	Office and Contact Hours
Dr. Heming Cui (heming@cs.hku.hk)	HW311/312, Thur/Fri lecture hours

Teaching Assistants	Office and Contact Hours
Shixiong Zhao (sxzhao@cs.hku.hk)	HW311/312, Thur/Fri lecture hours
Tianxiang Shen (stx635@connect.hku.hk)	HW311/312, Thur/Fri lecture hours
Fanxin Li (fxli@hku.hk)	HW311/312, Thur/Fri lecture hours
Haihui Zhou, Yixin Fang, Hanting Huang	HW311/312, Thur/Fri lecture hours
Chawla Singh, Yuming Fu, Agarwal Pranay	HW311/312, Thur/Fri lecture hours

Group Email (reaching all of us, recommended)
1340-2113-fall2019@googlegroups.com

I have already taken ENGG1330 before…

and here's ANOTHER programming course?

What this course is about?

- You'll learn about the various software development technologies and tools that a **competent computer science professional** should know and be good at.

⇨ The Eight Levels of Programmers

- To prepare you with **a solid programming skill and background** to cope with the upcoming challenges in the fundamental and advanced courses in Computer Science curriculum.

Major Topics

- Linux programming environment
- Version control
- C/C++ programming language

What to expect

- Get to further develop coding skills and professional practice
- Get to know of the latest development of
  - coding environment
  - software development tools
  - programming languages

Course Format

- Self-learning course

This is the only lecture in the course!!!

- Help sessions in laboratory
  - Thursdays 10:30-12:20pm: HW311/HW312
  - Fridays 4:30-5:20pm: HW311/312
  - not compulsory but students may be requested to show up in case of unsatisfactory performance
- Module based with progress check

Week	Schedule (materials may release earlier)	Note
1	#1: Linux Environment	
2	#2: Shell Script + Version Control	
3	#3: C/C++ Basics	Assignment 1 release
4	#4: Compilation & Debugging	
5	#5: Arrays	
6	#6: Structure Class & Union	Assignment 2 release
7	Reading Week	
8	#7: Function & Recursion	
9	#7: Function & Recursion	Assignment 3 & 4 release
10	#8: File I/O	
11	#9: Pointers & Memory Management	
12	#10: Data structures	

Modules

- ~10 modules, each due in 3 or 4 weeks
- Learning materials with clear objectives will be provided
- You will need to complete module checkpoints tasks before a given deadline for each module; deadlines are flexible (explain later)
- Practice exercises
- Extra reading & references

Assignments

- (Four) programming assignments which you should complete individually and submit by the deadline
- Can be done outside labs; no in-class quiz in in course
- Suggest submission dates (1 mon); deadlines flexible (explain later)
- Discussions of materials and problems are encouraged. But make sure that your submission is your own work.
- Sample test cases are provided but additional test cases are used for grading

Assignment Grading

- No credit for a program that cannot compile/run in a given standardized environment (to announce later).
- Plagiarism detection software will be used in the grading of every assignment.
- Your code will be auto-graded for technical correctness
- However, the correctness of your implementation — not the auto-grader's judgements — will be the final judge of your score

Assessment

- Final Written Examination (30%)
- Continuous Assessment (70%)
  - Checkpoint Task Submissions [10*2%]
  - Four Programming Assignments [4*12.5%]

What do we expect of you?

- Be responsible for your learning progress
- Check Moodle frequently <- This is the only official direct communication channel for the course
- Follow course schedule
- Read instructions carefully
- Seek help from me and TAs (join every lab hour on Thur/Fri)
- Active engagement in discussions (online / offline)
- Be eager to become a competent computer science professional

What kind of support do you have?

- Moodle online Q&A and discussions
- Help sessions
- Consultation hours (email me first, can be in my office or lab)
- Additional student TA support

Getting effective help

What **NOT** to do when seeking help

- Ask without doing some research if there are already answers to similar issues

Be well-prepared before you ask

- Merely throw your codes at TAs / STAs and hoping that they will debug for you
- Post your assignment code or answers to Moodle

Getting effective help

- Highly recommended to have online communications on Moodle so everyone can share and learn
- How to ask?
  - Be specific

    Bad example
    I don't understand.

    Good example
    Why isn't this variable effective inside this loop?

  - Provide the context / background

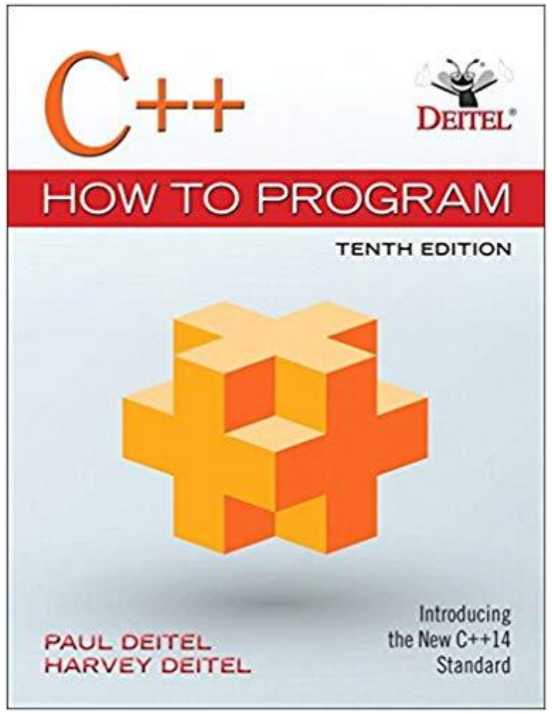
    Bad example
    It doesn't work. Why?

    Good example
    I got this error when compiling my code with this Makefile

⇨ How do I ask a good question? (stackoverflow)

Textbook

- C++: How to program
  - Hardcopy (10th edition) available from bookstore
  - Electronic version (9th edition) available from HKU library https://proquestcombo-safaribooksonline-com.eproxy.lib.hku.hk/9780133378795
- A point to note: we will teach C/C++ side-by-side
  - Strings (C-string vs. string class)
  - Standard I/O (printf/scanf vs. cin/cout)
  - file I/O (FILE * vs. fstream)
  - Memory management (malloc/free vs new/delete)



SETL reviews in my past COMP2123 course

- "I can learn many basic techniques on Linux, c++, C, python. Once I learn the basic concept, I can find what language I'm interested. In the course, I find that c++ is very interesting so that I try to learn more advance technique on it by reading book in library."
- "I find I have better memory of the stuff I go through the lab materials thoroughly which is less likely to achieve in lecture-based class."
- "I grasp and remember the material faster and easier by working on tasks/problems."
- "Teacher/TA is not needed in the classrooms and labs"
-

The roles of TAs and I

- We understand not everyone is born to be a good self-leaner, so we will learn to be a good self-learner in this course
- Interaction and caring individual learning progress, you are encouraged to interact with us.
- We will provide support and guidance, helping you through the learning process.
- Fostering self-learning atmosphere with peer support
- Team learning is encouraged: you can form study groups and share your ideas and findings with peers; will announce on Moodle soon

Friendly hints for success in this course

- Attitude is everything! E.g., submit all self-learning checkpoints and assignments, 99% pass
- Question/confusion/difficulty: find us; we are nice and fair!
- It is your responsibility to push Submit button correctly on Moodle
    - You verify each submission is successful by downloading it
    - Do not send submissions to us by emails (bad impressions)

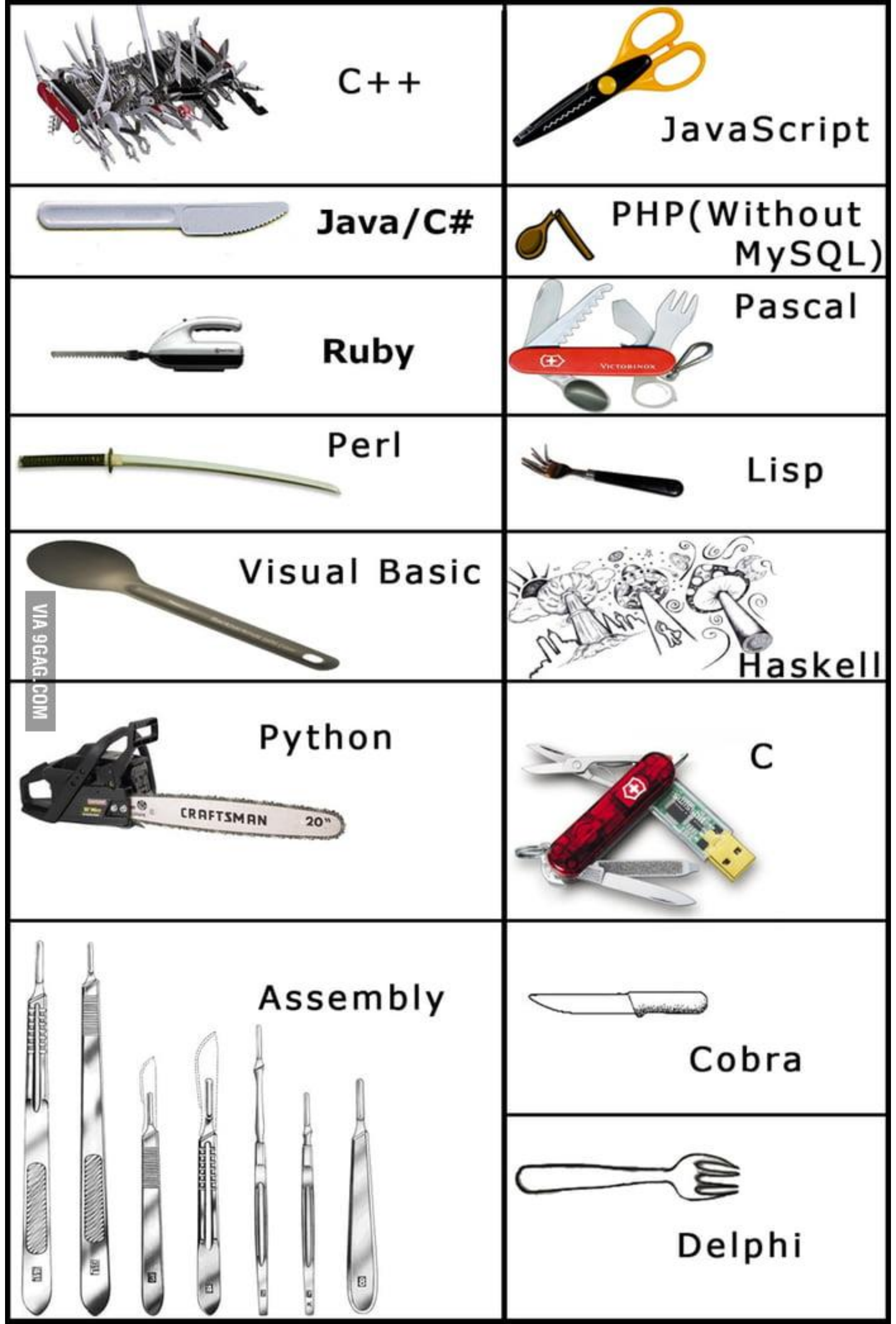
Deadline policy: flexible

- Deadlines of each checkpoint & assignment, see Moodle. E.g.,
  - Due date: Oct 10, 2019 (11:59pm, HK time)
  - Cutoff date (1 week after): Oct 17, 2019 (11:59pm, HK time)
- For all self-learning checkpoints (for modules) and assignments, each student can have five late submissions in total
  - A submission submitted after due date is defined as late (everyone can have up to five late submissions, no penalty at all)
  - Each late submission can be as late as 1 week (must be submitted before the cutoff date; otherwise, Moodle submission site will close, and zero score will be received)
  - Starting from the sixth late submission, all late submissions will get zero
  - All assignments will be due on Dec 15 (so their cutoff dates will be on Dec 22); I recommend you submit all checkpoints on time and leave late submission chances for assignments; no exchange
  - Do not ask for special treatments from me about the policy; it is already very flexible & fair

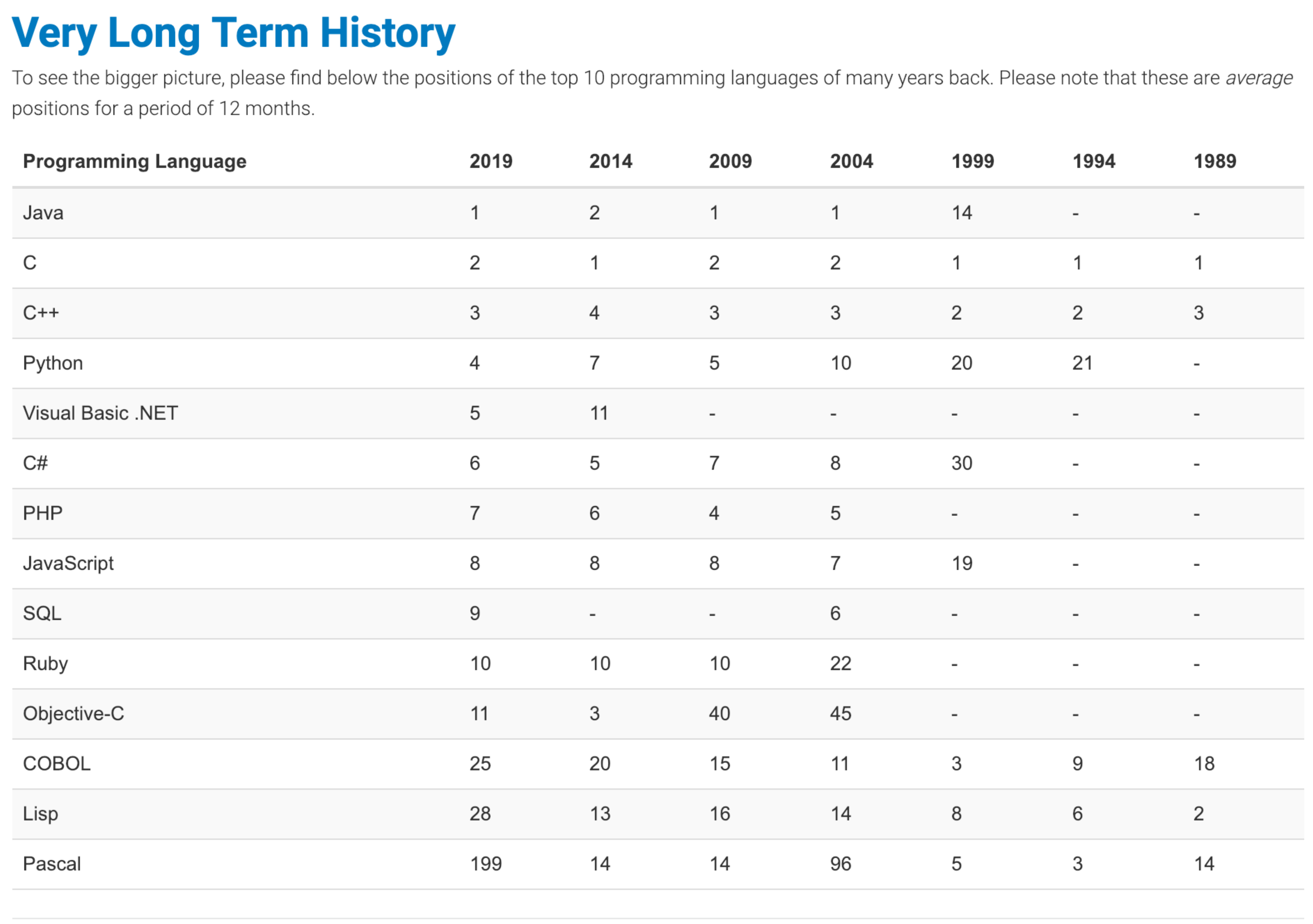
Here we go…

The Introduction (Kinda)

C & C++ are powerful:



TIOBE programming community index: measure of popularity of programming languages



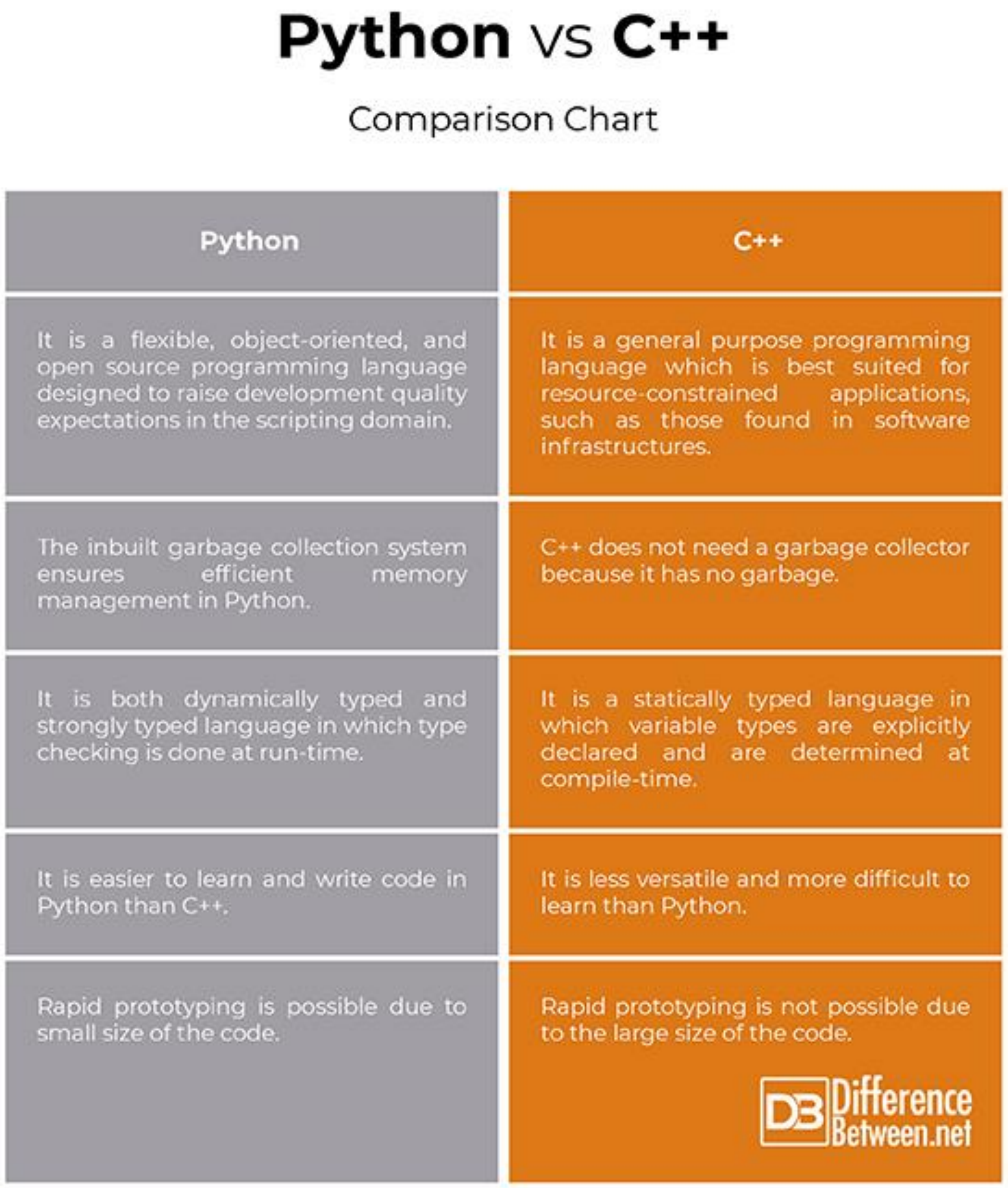
Very Long Term History

To see the bigger picture, please find below the positions of the top 10 programming languages of many years back. Please note that these are *average* positions for a period of 12 months.

Programming Language	2019	2014	2009	2004	1999	1994	1989
Java	1	2	1	1	14	-	-
C	2	1	2	2	1	1	1
C++	3	4	3	3	2	2	3
Python	4	7	5	10	20	21	-
Visual Basic .NET	5	11	-	-	-	-	-
C#	6	5	7	8	30	-	-
PHP	7	6	4	5	-	-	-
JavaScript	8	8	8	7	19	-	-
SQL	9	-	-	6	-	-	-
Ruby	10	10	10	22	-	-	-
Objective-C	11	3	40	45	-	-	-
COBOL	25	20	15	11	3	9	18
Lisp	28	13	16	14	8	6	2
Pascal	199	14	14	96	5	3	14

C & C++ are awesome if you can master them!

Most importantly, know the languages and use the right language at the right time.

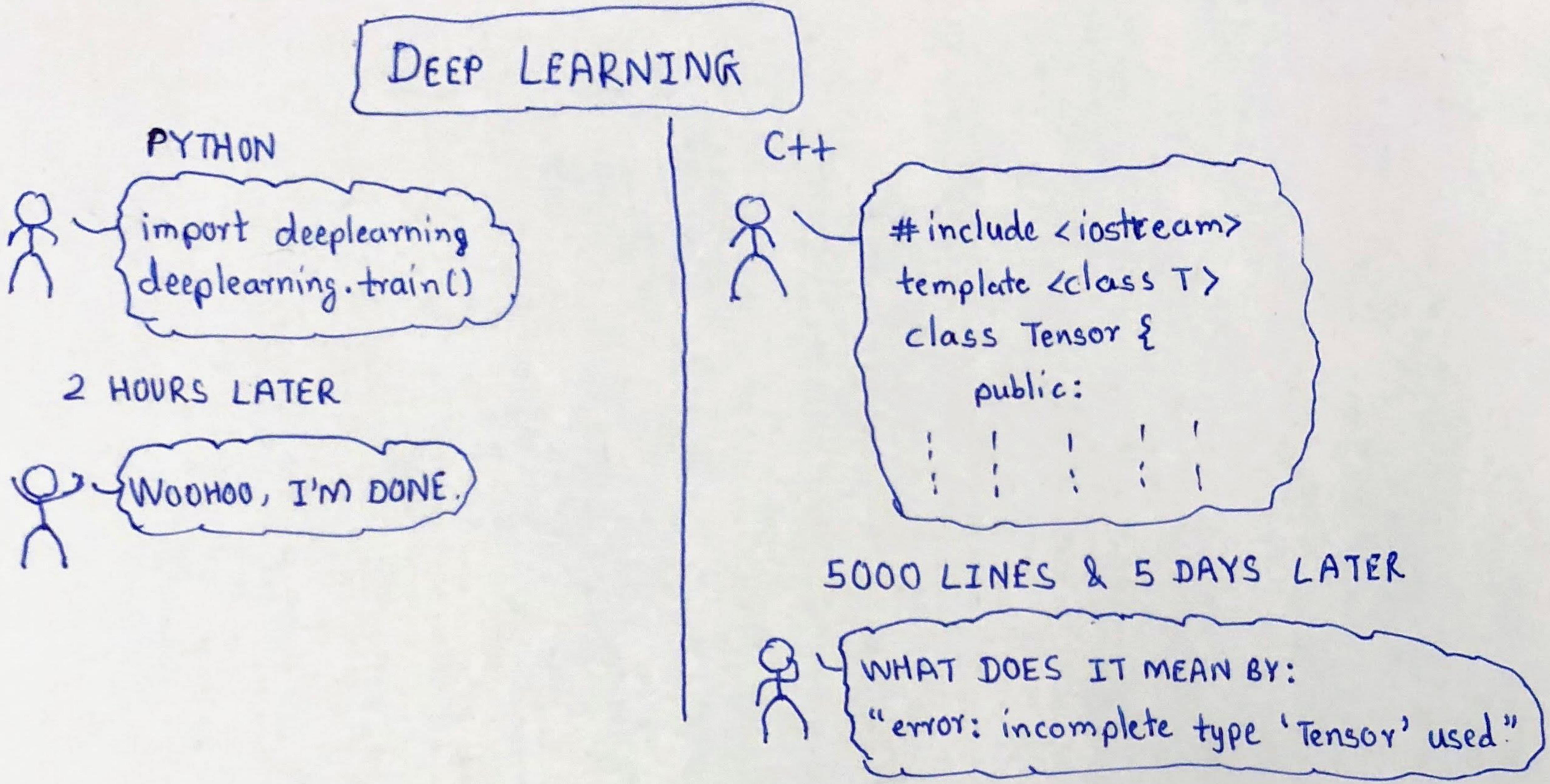


Python vs **C++**

Comparison Chart

Python	C++
It is a flexible, object-oriented, and open source programming language designed to raise development quality expectations in the scripting domain.	It is a general purpose programming language which is best suited for resource-constrained applications, such as those found in software infrastructures.
The inbuilt garbage collection system ensures efficient memory management in Python.	C++ does not need a garbage collector because it has no garbage.
It is both dynamically typed and strongly typed language in which type checking is done at run-time.	It is a statically typed language in which variable types are explicitly declared and are determined at compile-time.
It is easier to learn and write code in Python than C++.	It is less versatile and more difficult to learn than Python.
Rapid prototyping is possible due to small size of the code.	Rapid prototyping is not possible due to the large size of the code.

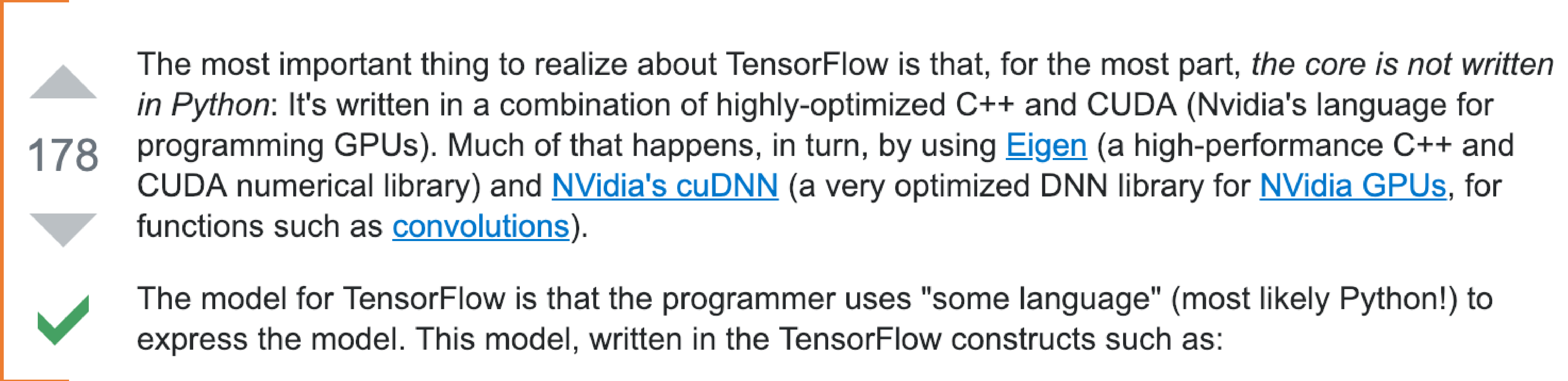
DB Difference Between.net



```
import deeplearning
deeplearning.train()
```

```
#include <iostream>
template <class T>
class Tensor {
    public:
```

stackoverflow

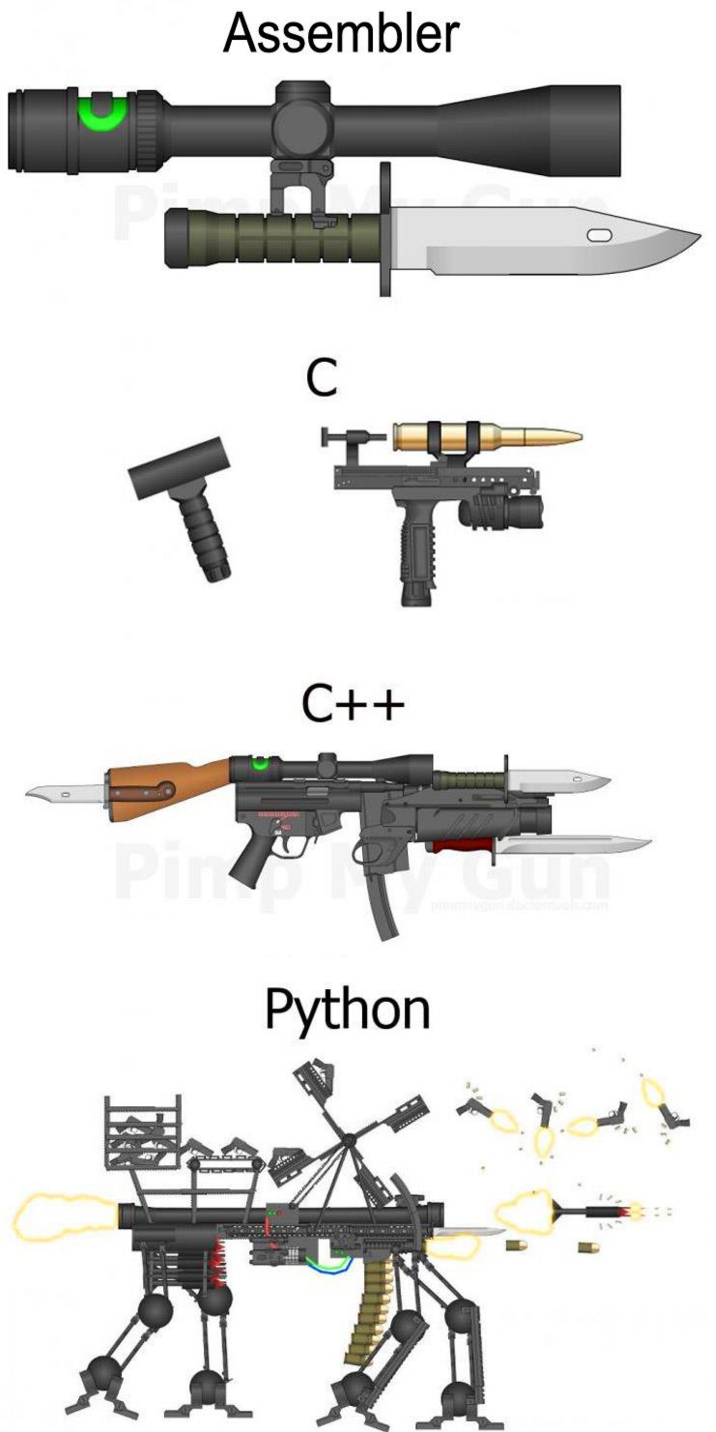


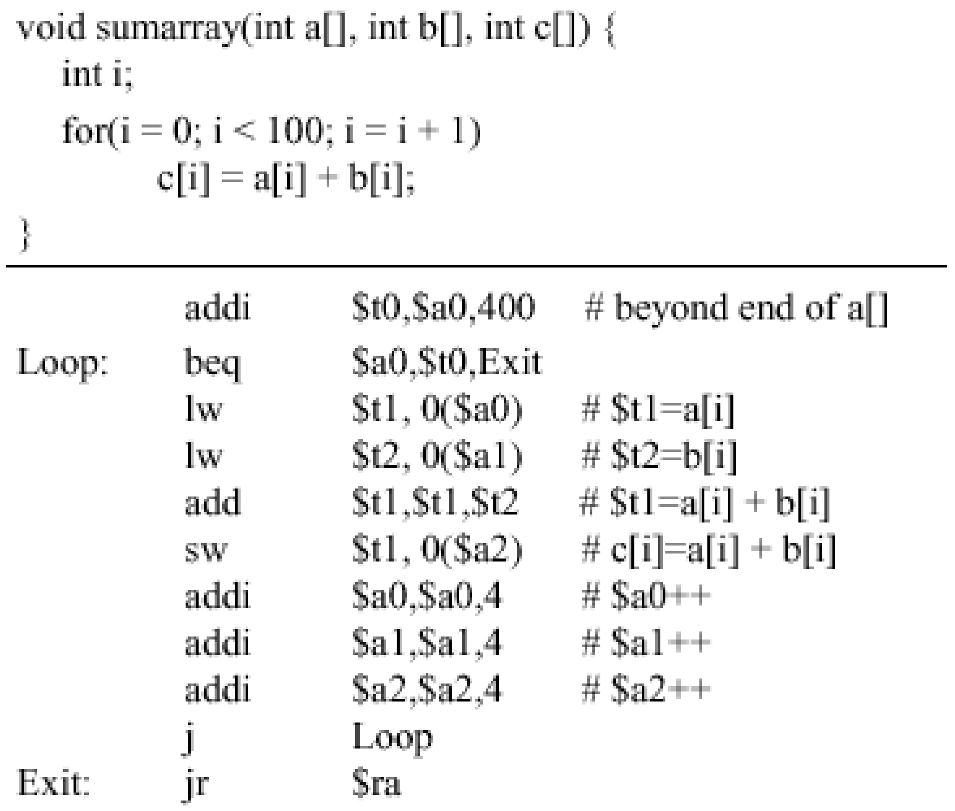
178

The most important thing to realize about TensorFlow is that, for the most part, *the core is not written in Python*: It's written in a combination of highly-optimized C++ and CUDA (Nvidia's language for programming GPUs). Much of that happens, in turn, by using Eigen (a high-performance C++ and CUDA numerical library) and NVidia's cuDNN (a very optimized DNN library for NVidia GPUs, for functions such as convolutions).

The model for TensorFlow is that the programmer uses "some language" (most likely Python!) to express the model. This model, written in the TensorFlow constructs such as:

The comparison was used to be between Assembly Language and C Language





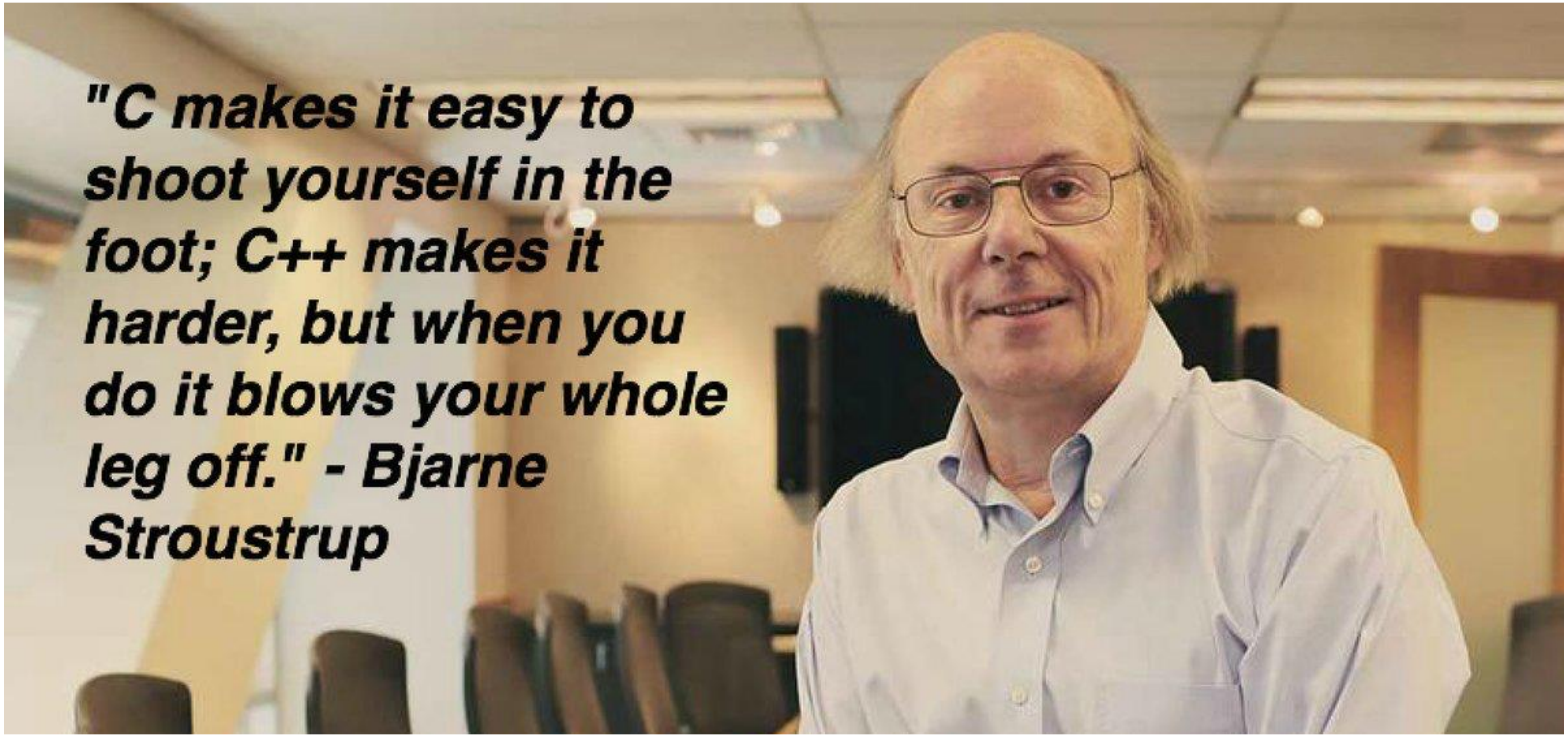
```
void sumarray(int a[], int b[], int c[]) {
    int i;
    for(i = 0; i < 100; i = i + 1)
            c[i] = a[i] + b[i];
}
```

```
          addi    $t0,$a0,400     # beyond end of a[]
Loop:     beq     $a0,$t0,Exit
          lw      $t1, 0($a0)     # $t1=a[i]
          lw      $t2, 0($a1)     # $t2=b[i]
          add     $t1,$t1,$t2     # $t1=a[i] + b[i]
          sw      $t1, 0($a2)     # c[i]=a[i] + b[i]
          addi    $a0,$a0,4       # $a0++
          addi    $a1,$a1,4       # $a1++
          addi    $a2,$a2,4       # $a2++
          j       Loop
Exit:     jr      $ra
```

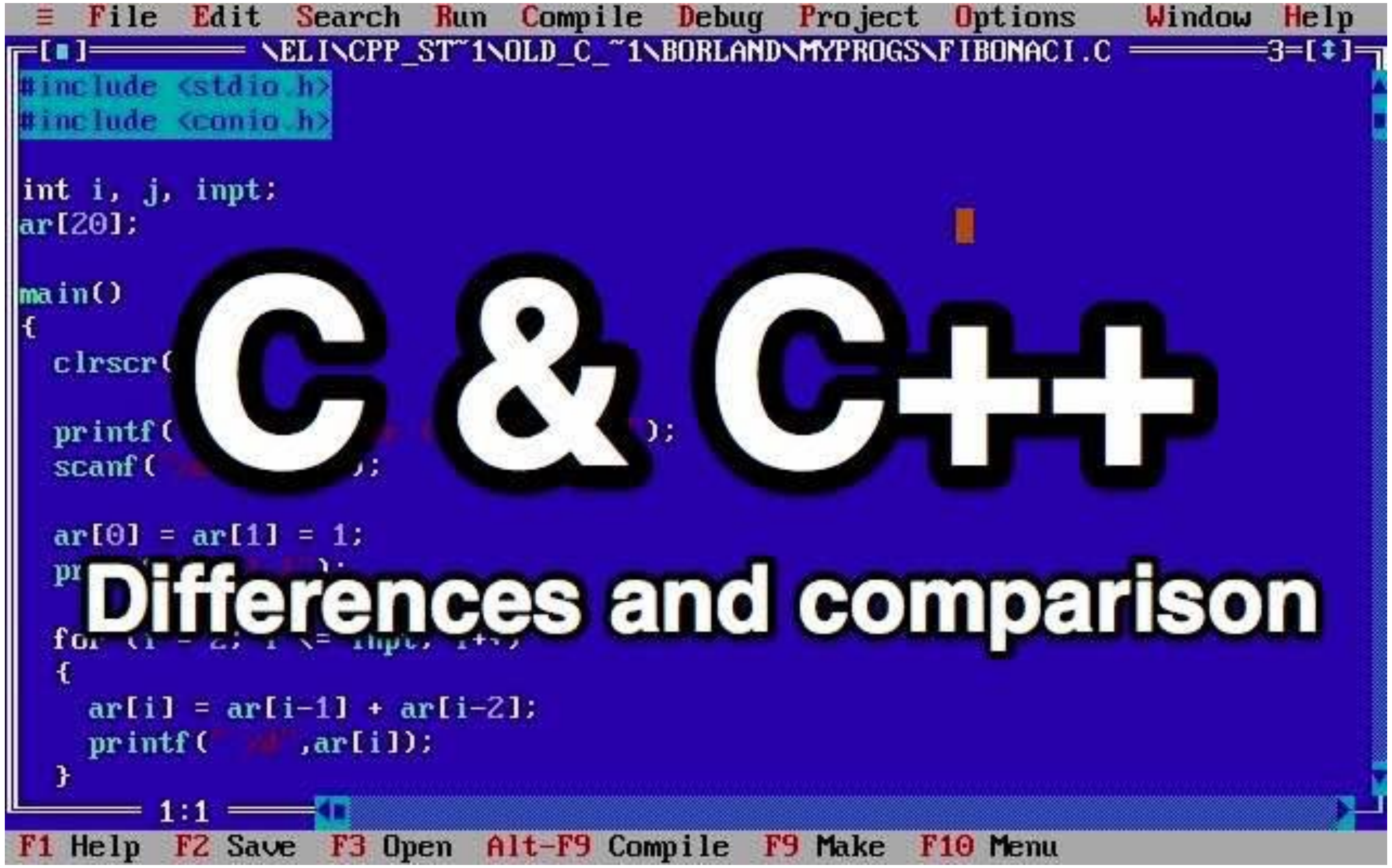
To be a hacker, C is a must, Assembly is the KEY!
(The Computer Hack that Saved Apollo 14, the best hack ever!)

https://www.youtube.com/watch?v=wSSmNUl9Snw, Scott Manley

What's the difference between C and C++


"C makes it easy to shoot yourself in the foot; C++ makes it harder, but when you do it blows your whole leg off." - Bjarne Stroustrup

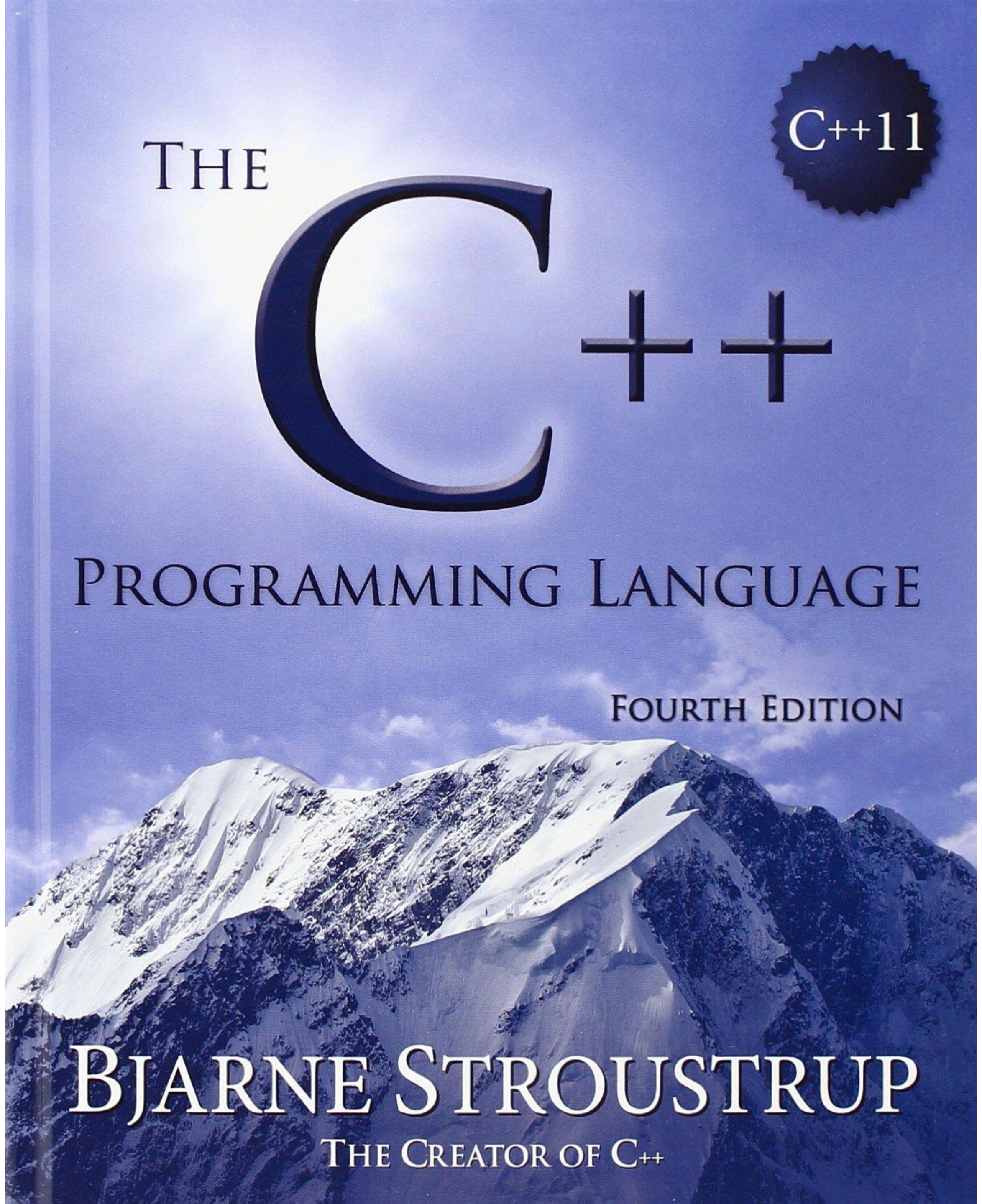
Learn by Parallel Comparison!


≡ File Edit Search Run Compile Debug Project Options Window Help
\ELI\CPP_ST~1\OLD_C_~1\BORLAND\MYPROGS\FIBONACI.C
3
C & C++
Differences and comparison
1:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

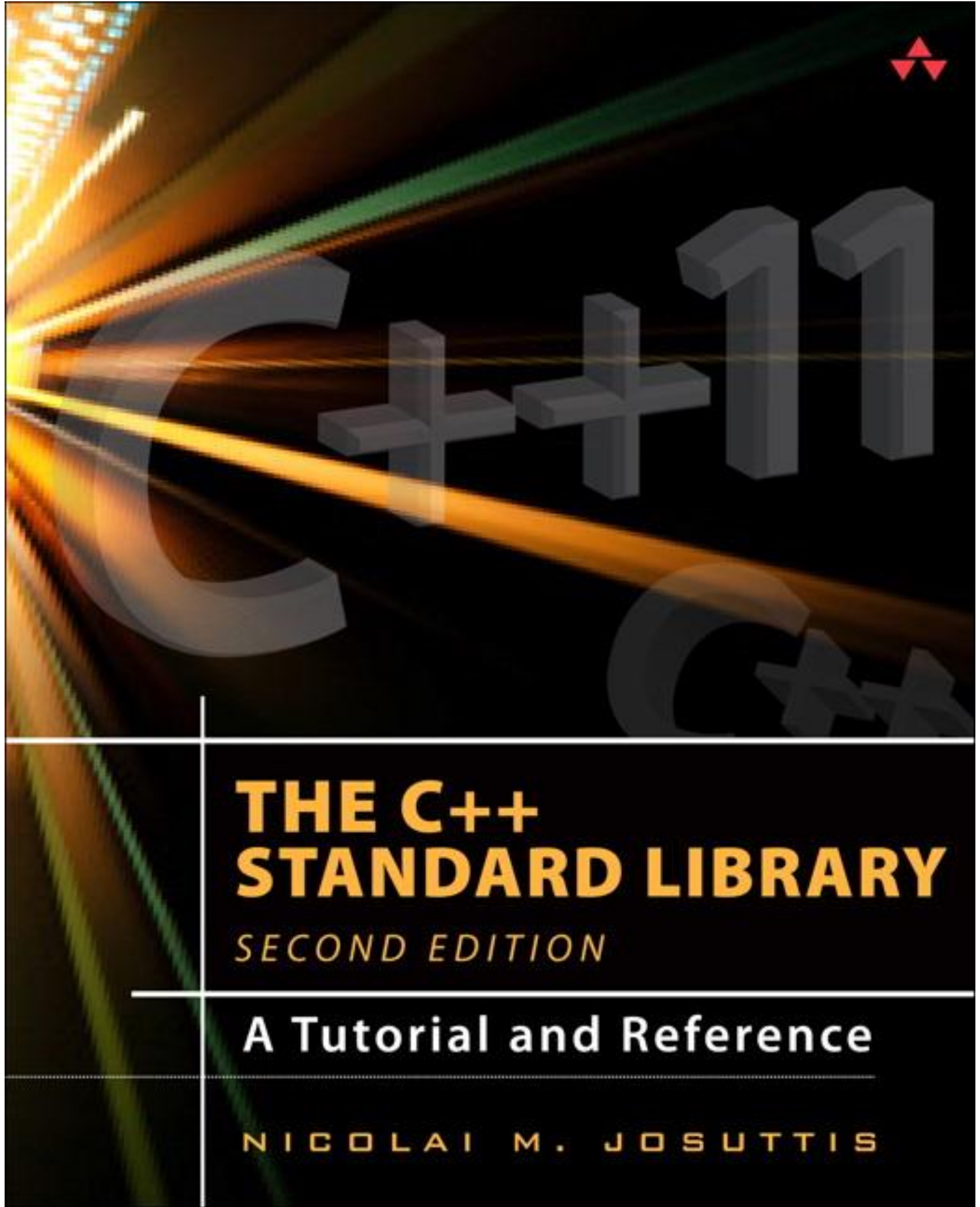
```
#include <stdio.h>
#include <conio.h>

int i, j, inpt;
ar[20];

main()
{
  clrscr(
  printf( [illegible] );
  scanf( [illegible] );

  ar[0] = ar[1] = 1;
  pr[illegible]);

  for (i = 2; i <= inpt; i++)
  {
    ar[i] = ar[i-1] + ar[i-2];
    printf(" %d",ar[i]);
  }
```

If you are a forerunner

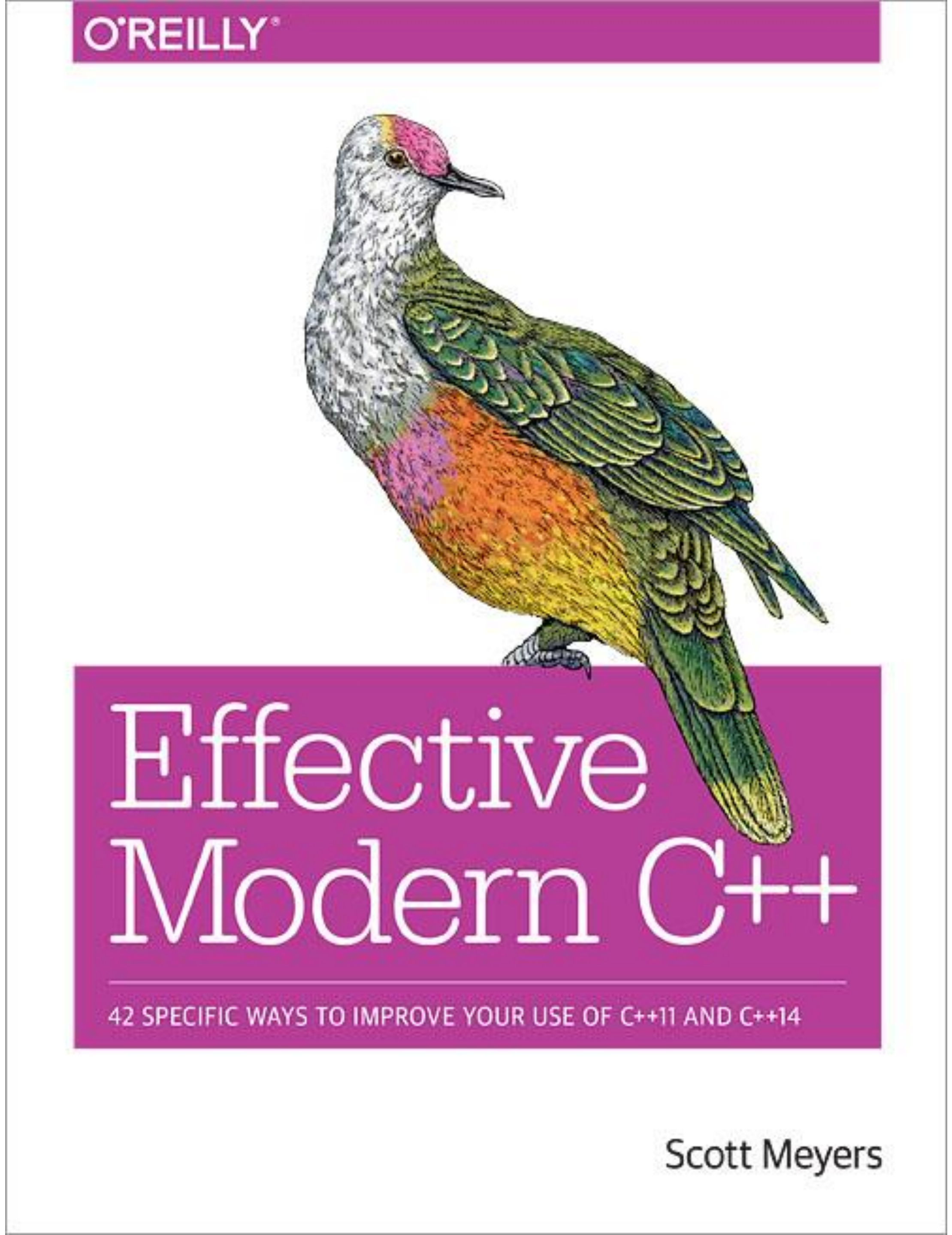
“If you think it's simple, then you have misunderstood the problem.”
— **Bjarne Stroustrup**



Becoming a real master in C++ (1)



Becoming a real master in C++ (2)



O'REILLY

Effective Modern C++

42 SPECIFIC WAYS TO IMPROVE YOUR USE OF C++11 AND C++14

Scott Meyers

Happy learning and we are all hands to help!