Object-Oriented Programming: Course Introduction

Lectured by Ming-Te Chi 紀明德

First Semester, 2022
https://goolink.cc/ejhk4

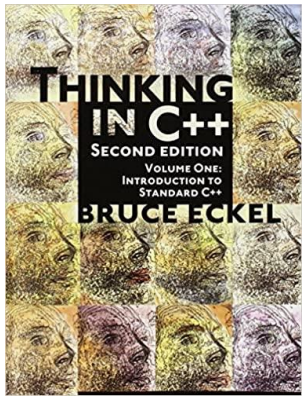
Computer Science Department
National Chengchi University

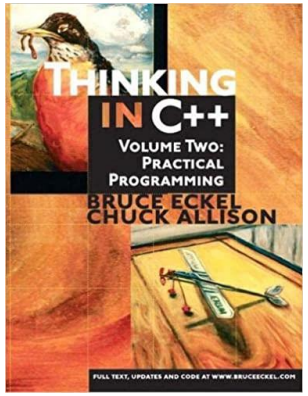
Slides credited from 李蔡彥 and 廖峻鋒

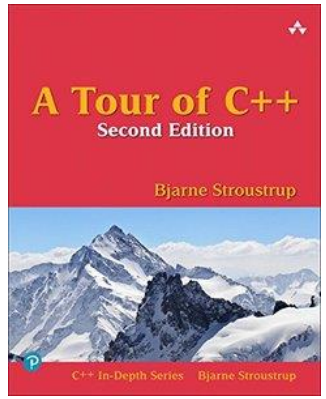
Course Information (I)

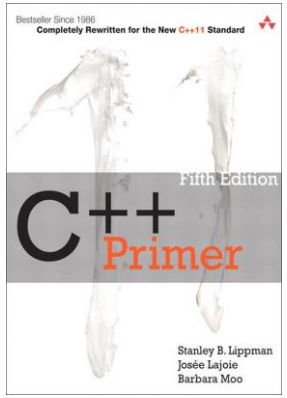
- Time: Monday D, 5, and 6 sessions (1:10pm-4:00pm)
- Place: 大仁樓 Room 3301
- Instructor:
  - Name: Ming-Te Chi（紀明德）
  - Email: mtchi@nccu.edu.tw
  - Phone: 62773
  - Office: 大仁樓200213室
  - Office hours: by arrangement
- Units: 3
- Prerequisites: proficiency in C
- Enrollment: limited to 70 students

Course Information (II)









- References:
  - B. Eckel (2000, 2003), *Thinking in C++,* Volume 1 & 2, Prentice Hall.
  - B. Stroustrup (2018), A Tour of C++, 2nd Edition
  - S. Lippman (2012), C++ Primer, Fifth Edition.
  - C. Ou, Modern C++ Tutorial: C++11/14/17/20 On the Fly. https://github.com/changkun/modern-cpp-tutorial

  - *The C++ programming language*, 4th edition by Bjarne Stroustrup, published by Addison Wesley. 2013
  - *C++ How To program, 10th Edition,* by Deitel and Deitel, published by prentice hall, imported by 全華.

Course Information (III)

- Grading: (tentative)
  - Midterm 25%
  - Final exam 25%
  - Assignments 40% (7-8 programming assignments)
  - Practice 10%
  - Bonus ??% (participation and in-class performance)
- Course material:
  - Slides and Handouts – in the moodle platform

Course Information (IV)

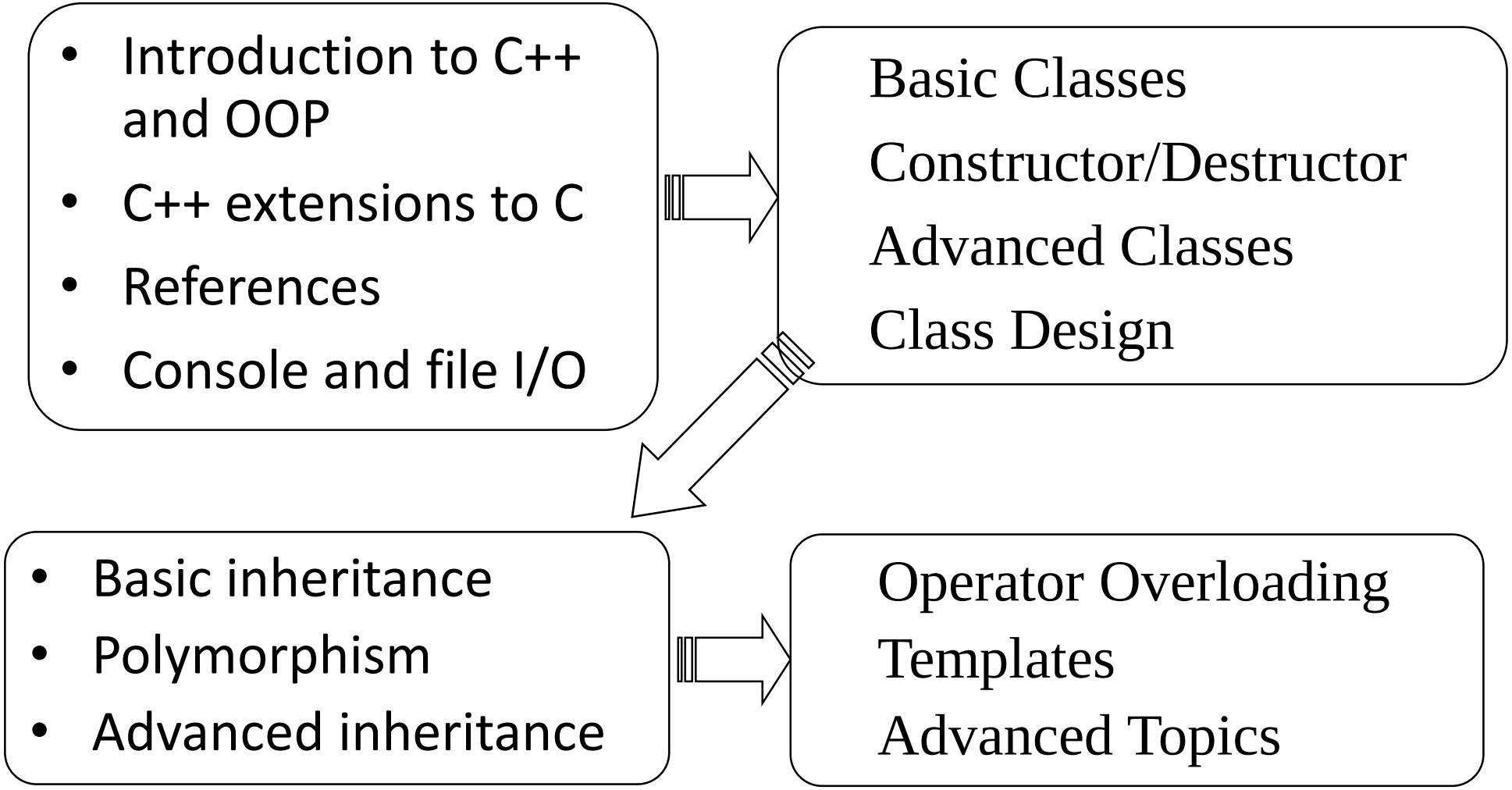
- Communication:
  - Web Site: https://moodle.nccu.edu.tw/course/view.php?id=28857
  - Hardware and software
  - UNIX platform: g++, gdb, and make
- Electronic homework submission system
- Honor code
- Material not covered: basic C, GUI design, etc.

Course Information (V)

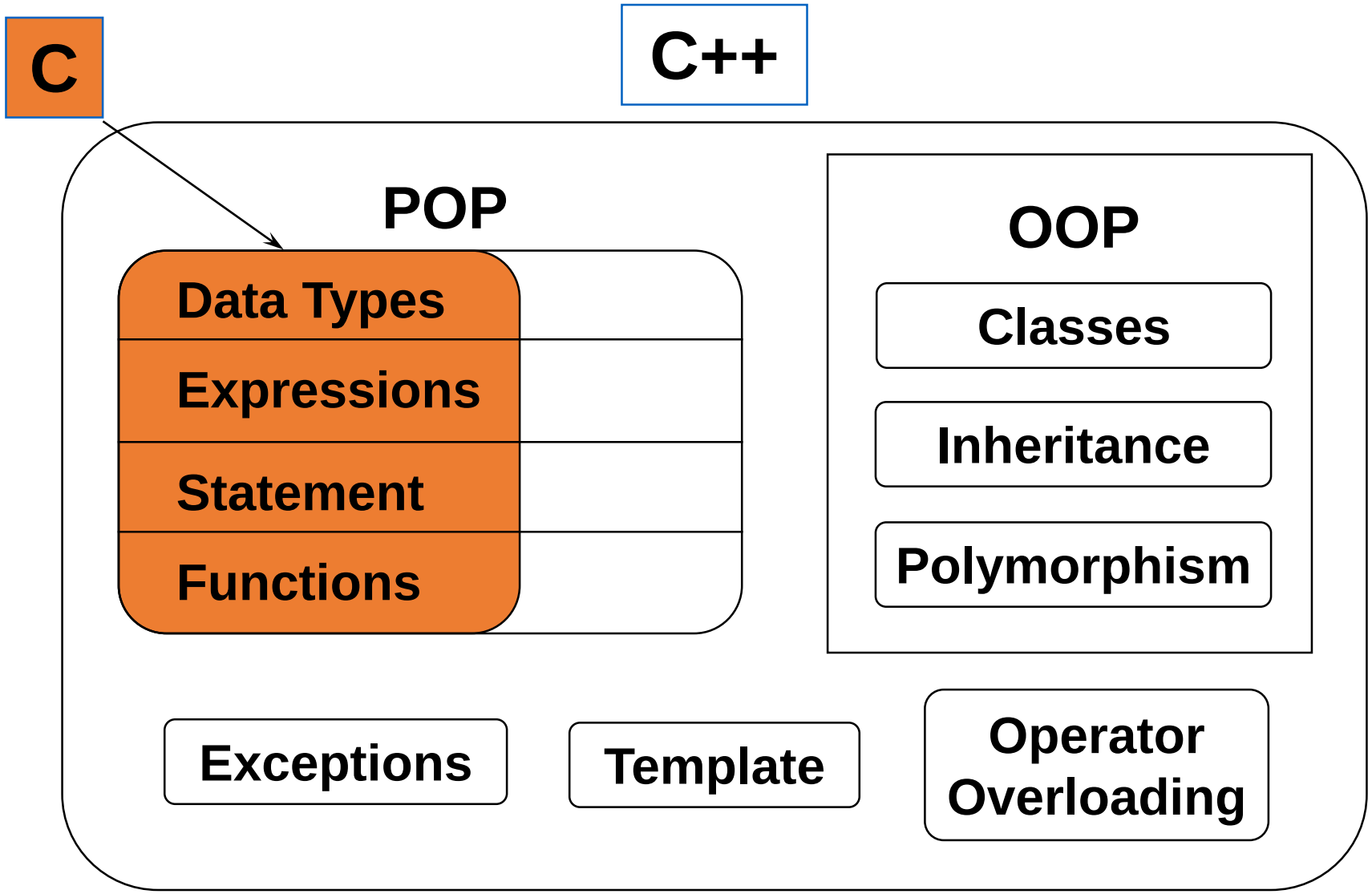
Weeks	Date	Topics	Chapter*	Assignment
1	09/12	Course Introduction	Ch 1	assign #0
2	09/19	Moving from C to C++	Ch 3, 8	assign #1
3	09/26	Function and Reference	Ch 2, 7, 8	
4	10/03	Class and Object	Ch 4, 5, 6, 9	assign #2
5	10/10	National Day (no class)		
6	10/18	More on Classes	Ch 9, 10, 11	assign #3
7	10/25	Class Design		
8	10/31	Operator Overloading	Ch 12 & 13	assign #4
9	11/07	Midterm		
10	11/15	Inheritance	Ch 14	assign #5
11	11/22	Advanced Inheritance	Ch 15, V2, Ch 6	assign #6
12	11/29	Templates	Ch 16, V2 Ch 3	
13	12/06	C++ Stream Input/Output	V2, Ch 2	
14	12/13	Advanced Topics	V2, Ch 7, 8	assign #7
15	12/20	Standard Template Library	V2, Ch 4, 5	
16	12/27	Introduction to other Language		
17	01/02	(no class)		
18	01/09	Final Exam		

*B. Eckel (2000, 2003), *Thinking in C++,* Volume 1 & 2, Prentice Hall.

Course Outline



Relationships Between Various Topics



First Try using C++

```
#include <cstdio>
class Trace {
public:
	void print(const char* s) {printf("%s", s); }
};

int main()
{
	Trace t;
	t.print("begin main()\n");
	//main body
	t.print("end main()\n");
}
```

First Try using C++ - improvement

```
#include <cstdio>

class Trace {
public:
      Trace() {noisy = 0;}
      void print(const char* s) {if (noisy) printf("%s", s); }
      void on() { noisy = 1; }
      void off() {noisy = 0; }
private:
      int noisy;
};
```

First Try using C++ - another improvement

```
#include <cstdio>

class Trace {
public:
	Trace() {noisy = 0; f = stdout}
	Trace(FILE* ff) {noisy = 0; f = ff}
	void print(const char* s) {if (noisy) fprintf(f, "%s", s); }
	void on() { noisy = 1; }
	void off() {noisy = 0; }
private:
	int nosiy;
	FILE* f;
};
```

Try using C

```
#include <stdio.h>
void trace(const char* s) {
        printf("%s", s);
}
```

Try using C

```
#include <stdio.h>
void trace(const char* s) {
        printf("%s", s);
}
```

```
#include <stdio.h>
static int noisy = 1;
void trace(const char* s) {
        if(noisy) printf("%s", s);
}
void trace_on() {noisy = 1;}
void trace_off() {noisy = 0;}
```

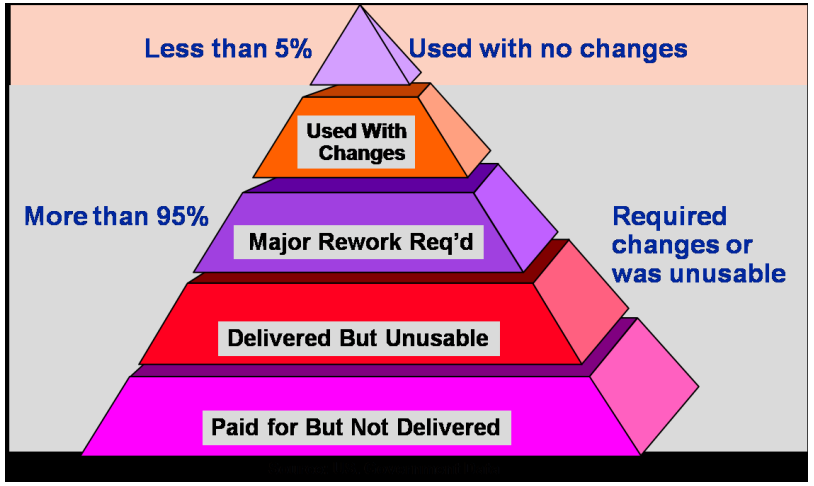
Why Program in C++

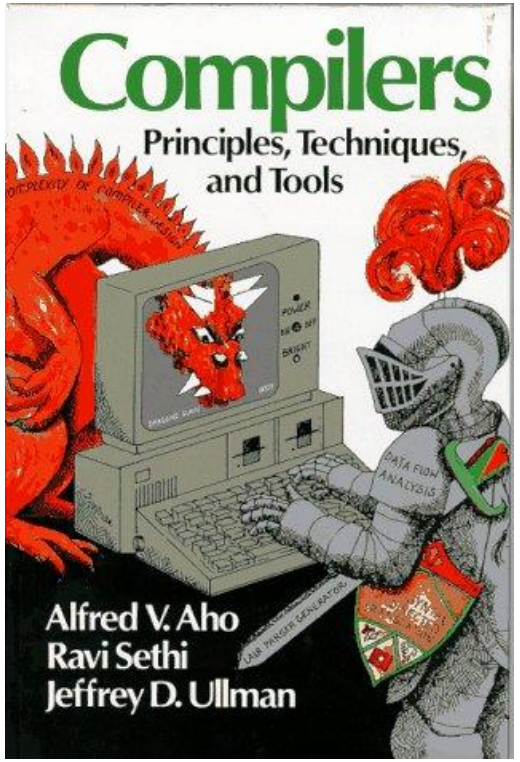
- Advantages of Object-Oriented Programming (OOP)
  - Improved reliability
  - Better reusability
  - Improved maintenance - after release
- Advantages of C++ over C
  - Object-oriented programming
  - Increased data abstraction
  - Improvements to C; e.g., better I/O, templates, references

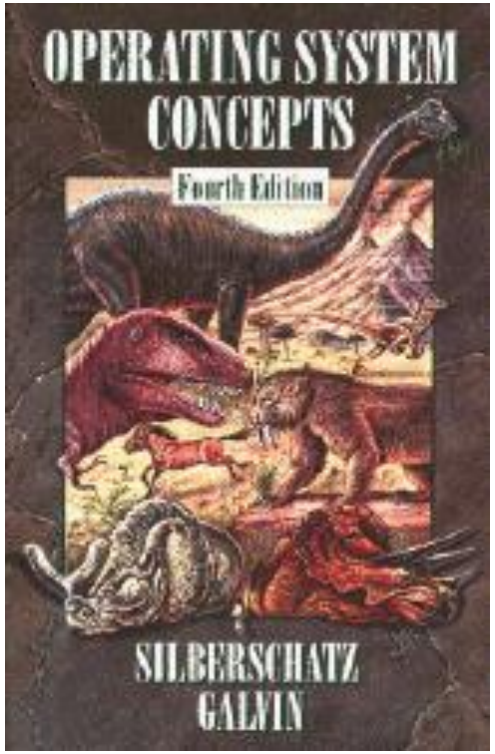
C++: a language to improve the deficiency of C

1968 軟體危機

- 在美國超過95%的軟體專案最終失敗或重做
- 許多人認為開發大型軟體是不可能成功的
- 當C程式碼超過25,000行到100,000行後，程式幾乎無法再維護





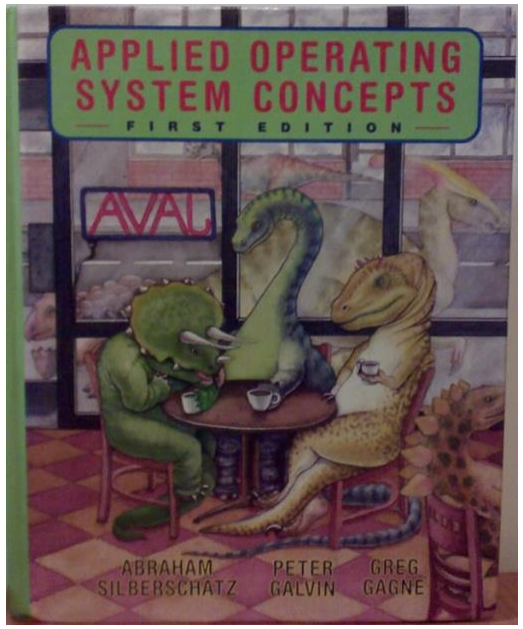


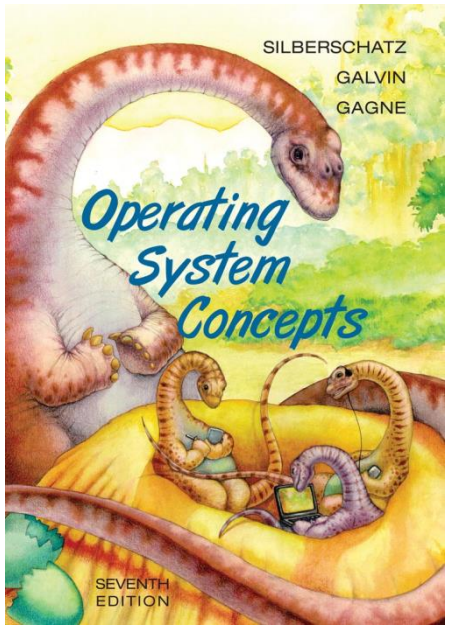
早期，開發大型軟體和跟大型怪獸打架的難度一樣高!

各式物件導向軟體工程方法論

物件導向分析、設計與程式技術
OOA, OOD, and OOP

各式物件導向與軟體工程技術讓我們有能力管理與建構大型、複雜的軟體系統

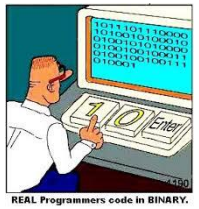




物件導向技術的發展，讓軟體開發不再是惡夢!

The Progress of Abstraction

- Abstraction的類型和品質直接影響你能解決多複雜的問題
  - Machine /assembly language: thinking in (binary) signals
  - C: thinking in steps
  - OO: thinking in “Domain concepts”



Machine Language

Assembly Language

C

C++

C#

Java

Scala

Python

DSL

SQL

電子電路

Programmer

真實的業務領域問題

為什麼要會OO?

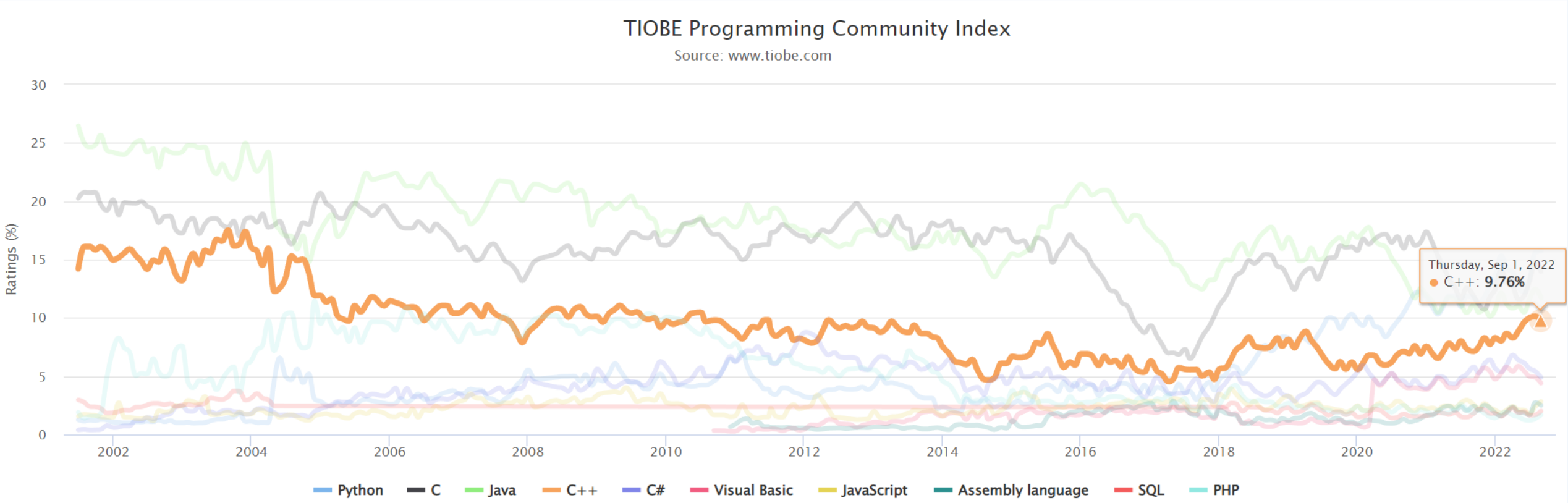
- 物件導向近年來已成為系統分析設計方法的主要思維
- 大部份當代語言均支援Object-based或Object-oriented
  - Java, C#, Ruby, Python, Javascript, Objective C, Scala, PHP, VB.NET, Delphi…
- 軟體技術發展一日千里，但物件導向設計原則十多年來變化較少，值得學習投資

Why Leaning C++?

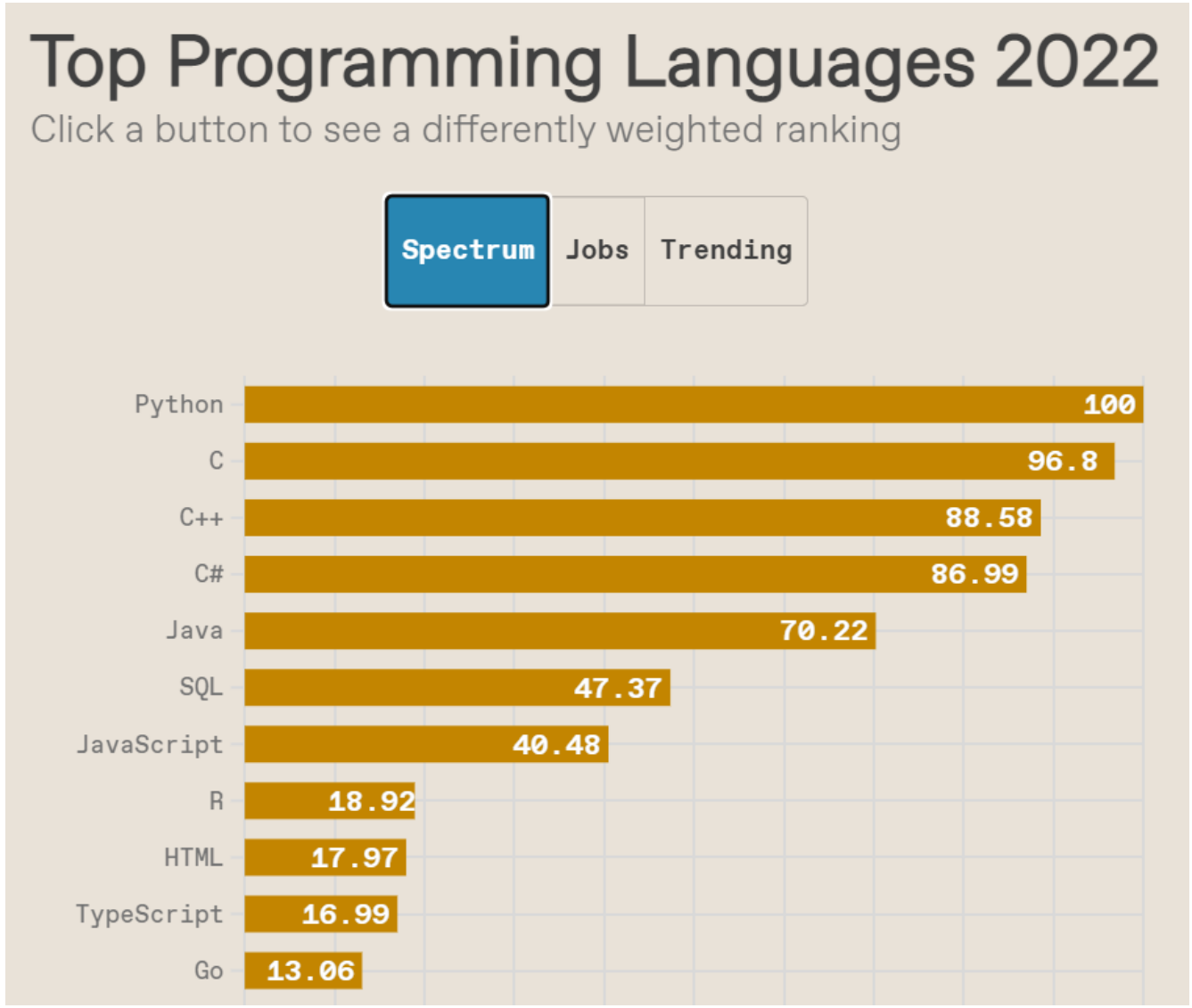
- Invention of C++
  - Bjarne Stroustrup在1980在Bell Lab所發展
  - 早期稱為C with classes，1983正式更名為C++
- 為何要學C++
  - 在眾多新語言的激烈競爭下，在許多先進科技如網路、電信系統、機器人、電腦動畫、電腦視覺、圖像處理等領域仍保有一席之地
  - 上承大一學習的C語言，加入物件導向概念的C++較易上手
  - 進可攻高階的新語言，退可處理系統層級議題

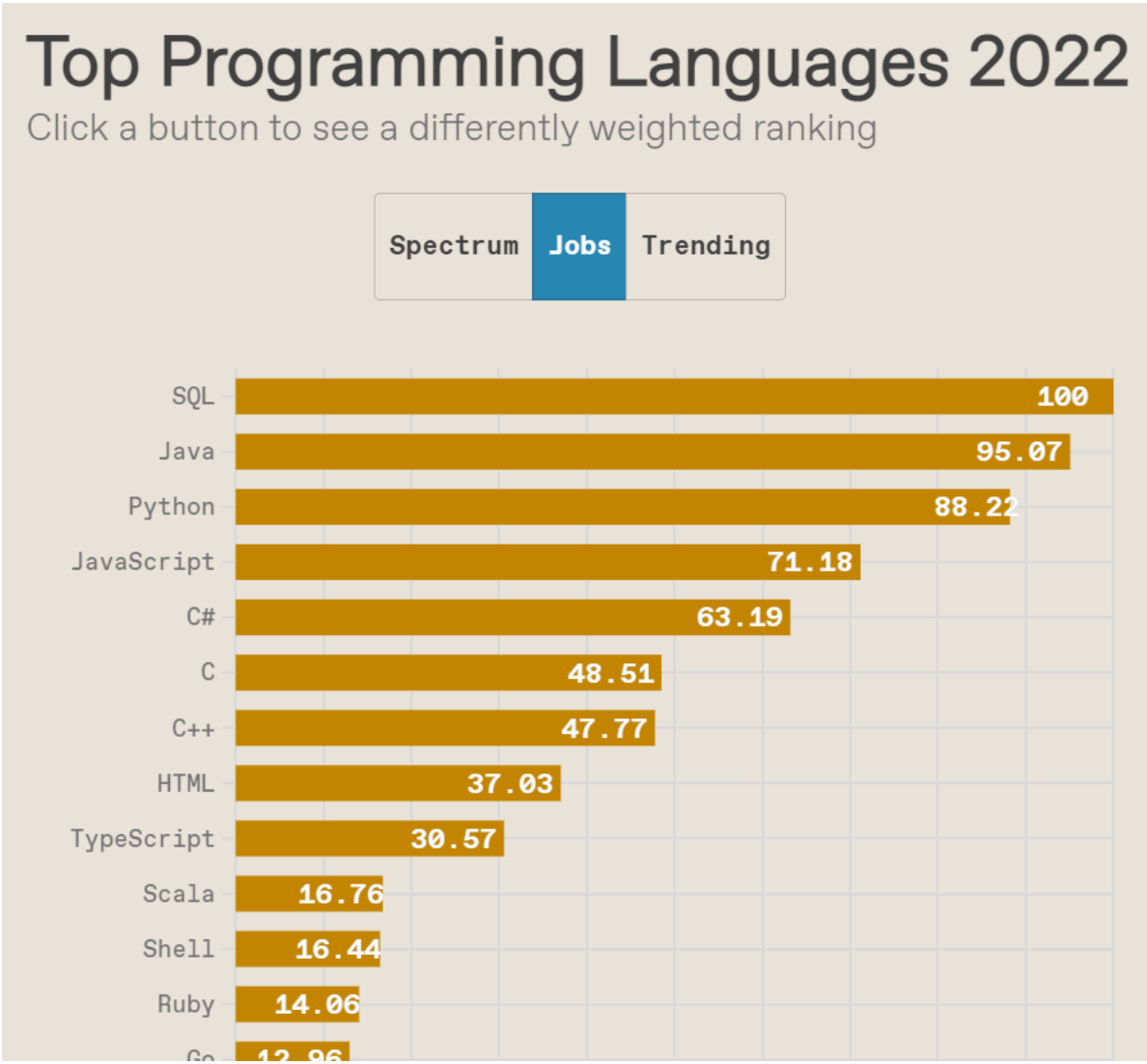
TIOBE Index: Popularity of Programming Language

- Sep 2022



IEEE Spectrum's Top Programming Languages 2022

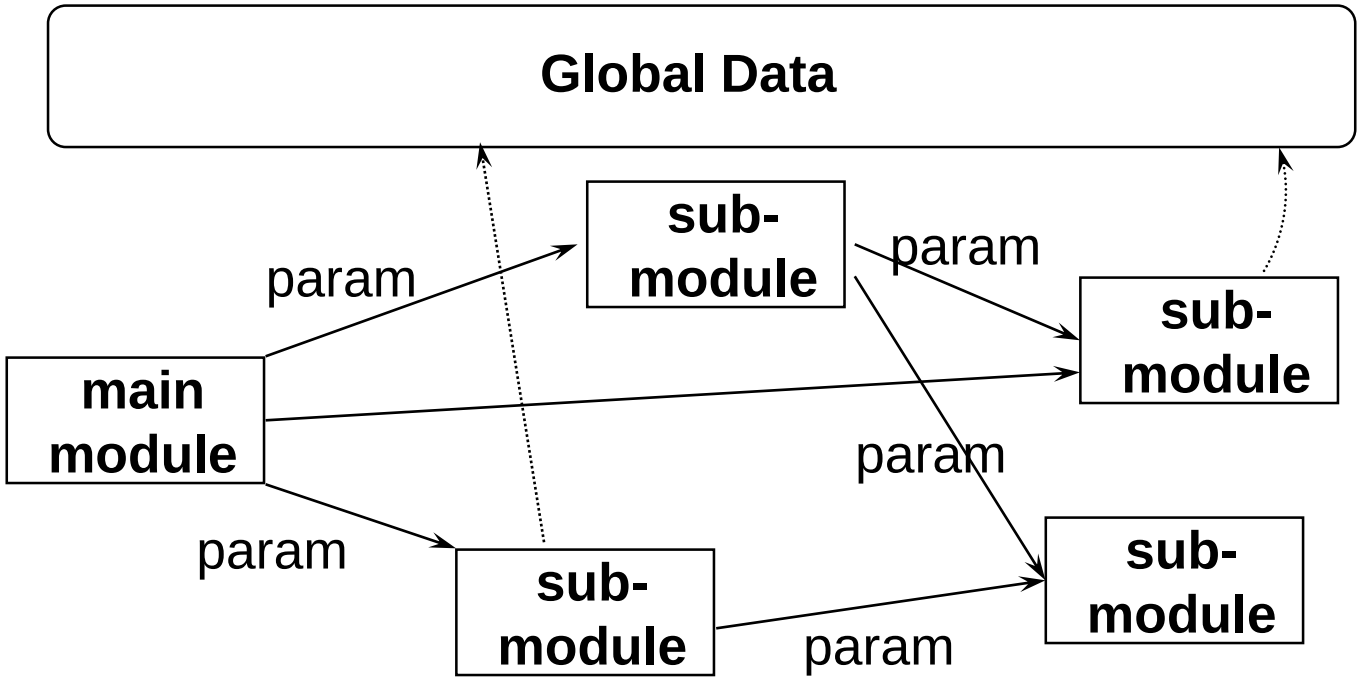




https://spectrum.ieee.org/top-programming-languages-2022

Process-Oriented Programming

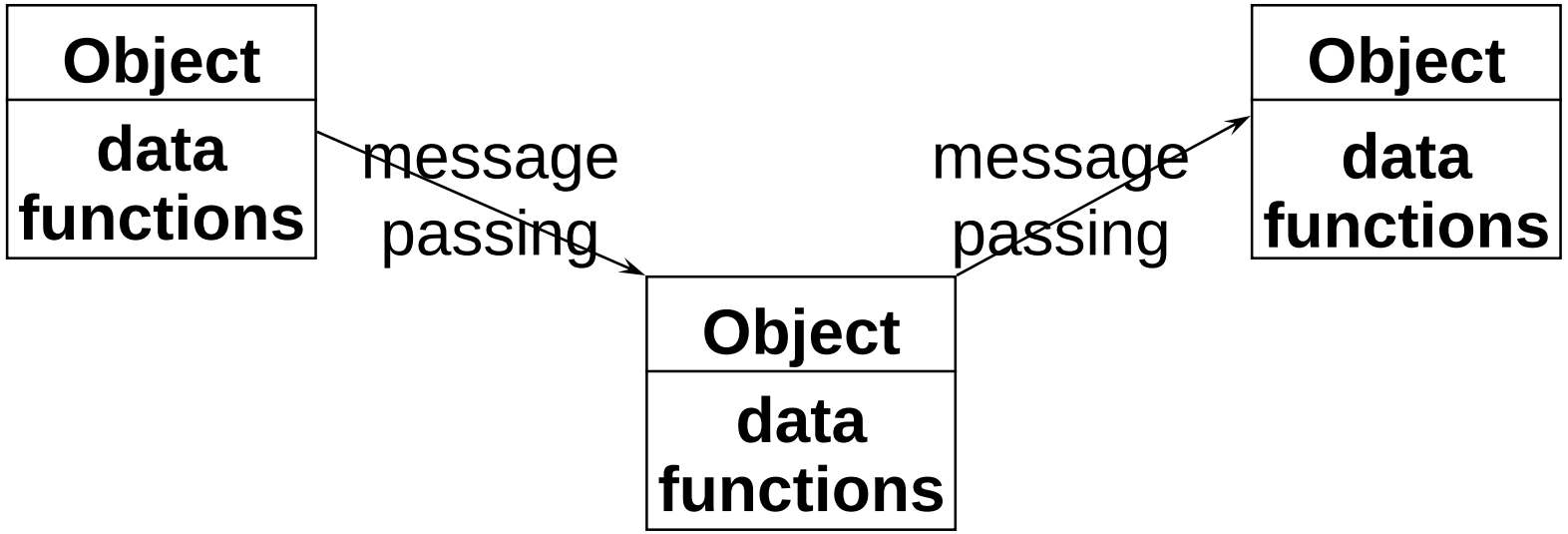
- Data and function are decoupled.
- Program structure is the theme.
- Levels of abstraction is the key.
- Good software engineering discipline is required.



Object-Oriented Programming

- Object - data and functions are coupled.
  - Objects have memory - member data
  - Objects have behavior - member function
- Increased data abstraction
- Better access control
- Better reusability - copying/modifying old code

Encapsulation
Polymorphism
Inheritance



Exercise: objects in a game

Final Fantasy V, SQUAREENIX

Summary on POP *vs.* OOP (I)

- Are there anything that you can do in C++ but not in C?
  - Probably very few but just a matter of how easily they can be done.
- Abstract Data Type (ADT) in C requires good software engineering discipline but ADT in C++ is natural and easy.
- In C++, you can control *access* on a function by function or data by data basis.

Summary on POP *vs.* OOP (II)

- In C, we usually takes *top-down* approach but in C++ we often use *bottom-up* approach. C is task/function oriented while C++ is object-oriented.
- Trend in programming language design:
  - letting the compiler do *more* work
  - detecting more bugs in *compilation-time* than in *run-time*.

Final Reminder

- Come to the first lab (practice session)
- Prepare yourself with the UNIX programming environment:
  - Basic commands, vim, gdb, makefile, …
- Review basic C
- Learn how to submit the homework
- Online judge system
- Prepare textbook/reference books
- Learning how to ask questions on moodle