

C++ Program Design

-- Introduction

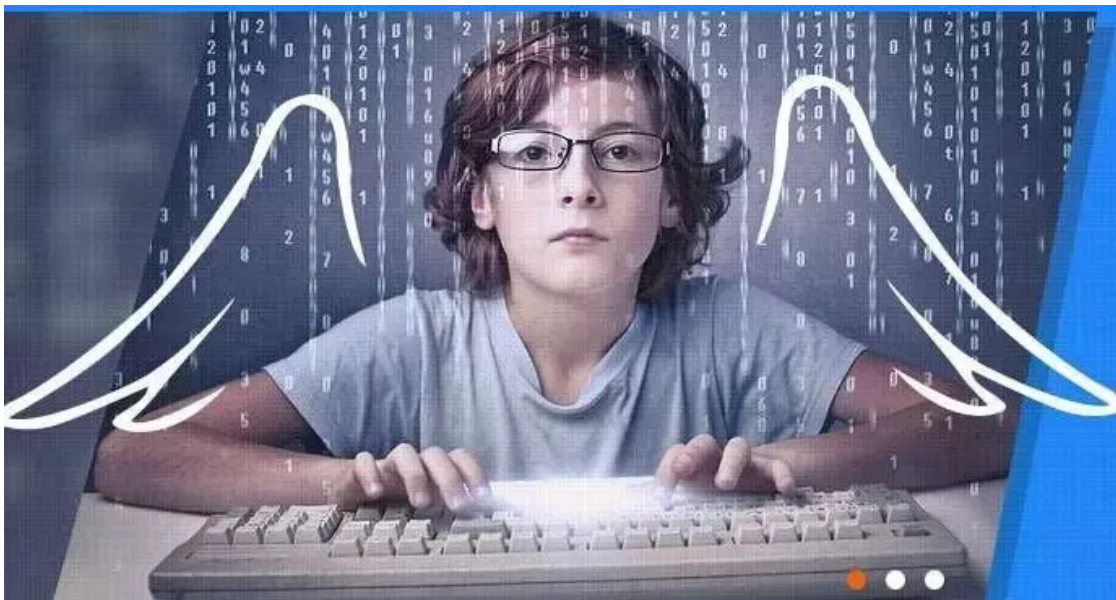


Junjie Cao @ DLUT

Summer 2019

<http://jjcao.github.io/cPlusPlus>

Coding is important



- 生活、工作在数字时代：网络+人工智能
- 许多工作岗位即将或已经正在逐步的被机器所替代的时代。
- 一切过程都需要被程序化
- 别人编程，我享用？

- When human beings acquired language, we learned not just how to listen but how to speak.
- When we gained literacy, we learned not just how to read but how to write.
- And as we move into an increasingly digital reality, we must **learn not just how to use programs but to make them.**
- In the emerging, highly programmed landscape ahead, you will either create the software or you will be the software. It's really that simple: **Program, or be programmed.**
- Choose the former, and you gain access to the control panel of civilization. Choose the latter, and it could be the last real choice you get to make.

本学期的目标

给定数据+问题描述，独立写程序解决这个问题

**Is Matlab/Python the final weapon
for us?**

Why teaching C++



C
Rulez!

Dennis Ritchie
1969 -- 1973 at [Bell Labs](#)
C89, ..., C99, C11, C18



C++
Rulez!

[Bjarne Stroustrup: Why I Created C++ - YouTube](#)
[bijani sdʒəʊsdʒup]

1979--1983 at [Bell Labs](#)
[C++11](#), [C++14](#), [C++17](#), [C++20](#)

Language evolution

- **Machine Language**

- The **very limited set** of instructions that a CPU natively understands is called **machine code** (or **machine language** or an **instruction set**)
- each instruction is composed of a number of binary digits, each of which can only be a 0 or a 1. These binary numbers are often called **bits** (short for binary digit)
 - an example x86 machine language instruction: 10110000 01100001
- each set of binary digits is **translated** by the CPU into an **instruction** that tells it to do a very specific job
 - compare these two numbers
 - put this number in that memory location.
- Different types of CPUs will typically have **different** instruction sets, so instructions that would run on a Pentium 4 would not run on a Macintosh PowerPC based computer.
- Back when computers were first invented, programmers had to **write programs directly in machine language**, which was a very difficult and time consuming thing to do.

- **Assembly Language**

- **High-level Languages**

Language evolution

- **Machine Language**

- an example x86 machine language instruction: 10110000 01100001

- **Assembly Language**

- each instruction is identified by a **short name** (rather than a set of bits), and **variables** can be identified by names rather than numbers
 - must be translated into machine language by using an **assembler**.
 - Assembly languages tend to be very **fast**, and assembly is still used today when speed is critical.
 - However, the reason assembly language is so fast is because assembly language is tailored to a particular CPU. Assembly programs written for one CPU will **not run on another CPU**.
 - Furthermore, assembly languages still require a lot of instructions to do even simple tasks, and are **not very human readable**.
 - the same instruction as above in assembly language: mov al, 061h

- **High-level Languages**

Language evolution

- **Machine Language**

- an example x86 machine language instruction: 10110000 01100001

- **Assembly Language**

- the same instruction as above in assembly language: mov al, 061h

- **High-level Languages**

- C++: **more abstract, easy:**

- Conciseness: 1 = many
 - Maintainability: easier to modify
 - Portability: suitable for different types of processor

```
int main(){  
    return 0;  
}
```

- C++ is a **high-level** language, **compiled** language, strong **types**, **case sensitive**.

编程语言和思想

- Assembly language
- Computation: Fortran 1954
- System programming: C 1969, **C++** 1979, C# 1999, Objective-C
- Application: Java 1995, Java script, **PHP**
- Unix shell to everything: Perl, **Python**, Ruby
- Computation: **Matlab**, Mathematics, Mapple, R
- **The "concept" of "programming languages" are quite "similar"**

Language is the dress of thought.

~Samuel Johnson

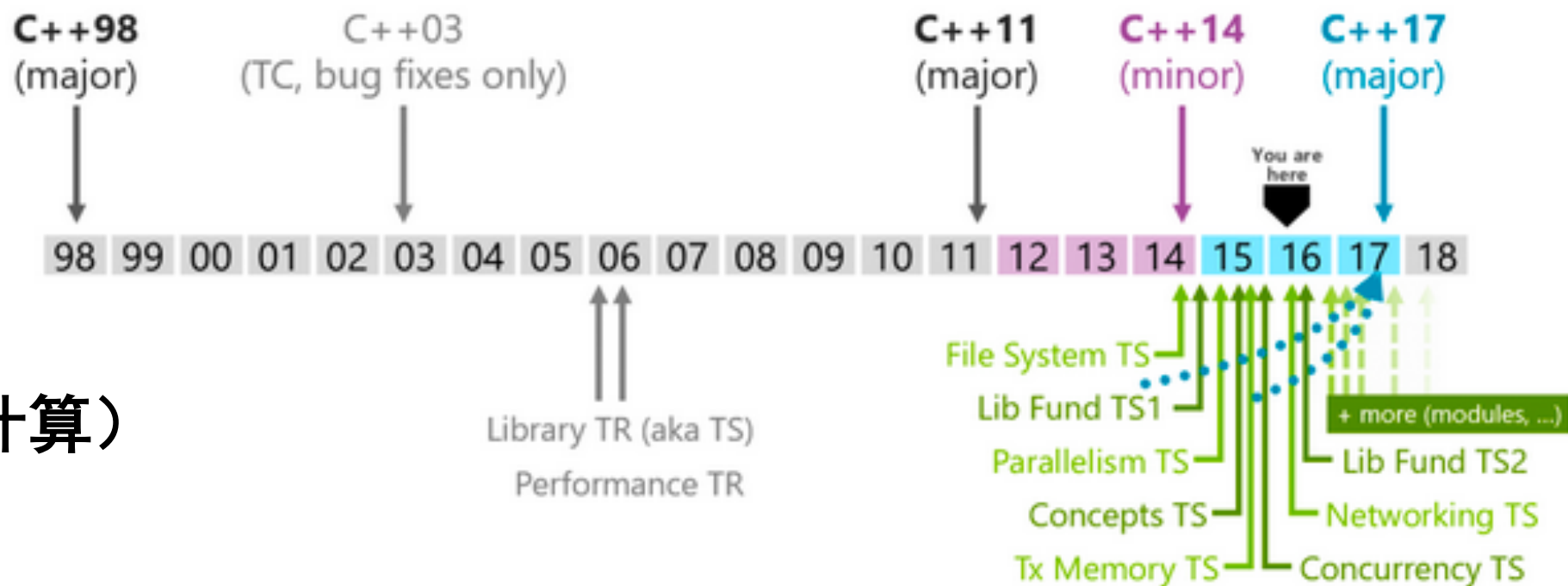
But if thought corrupts language, language can
also corrupt thought.

~George Orwell

Worldwide, Dec 2018 compared to a year ago:

Rank	Change	Language	Share	Trend
1	↑	Python	25.36 %	+5.2 %
2	↓	Java	21.56 %	-1.1 %
3	↑	Javascript	8.4 %	+0.0 %
4	↑	C#	7.63 %	-0.4 %
5	↓↓	PHP	7.31 %	-1.3 %
6		C/C++	6.4 %	-0.4 %
7		R	4.01 %	-0.3 %
8		Objective-C	3.21 %	-0.9 %
9		Swift	2.69 %	-0.7 %
10		Matlab	2.06 %	-0.3 %

C++称霸的领域



- 游戏
- HPC（高性能计算）
 - 人工智能底层
- 编译器
- 金融财务领域：高频交易平台
- 等等

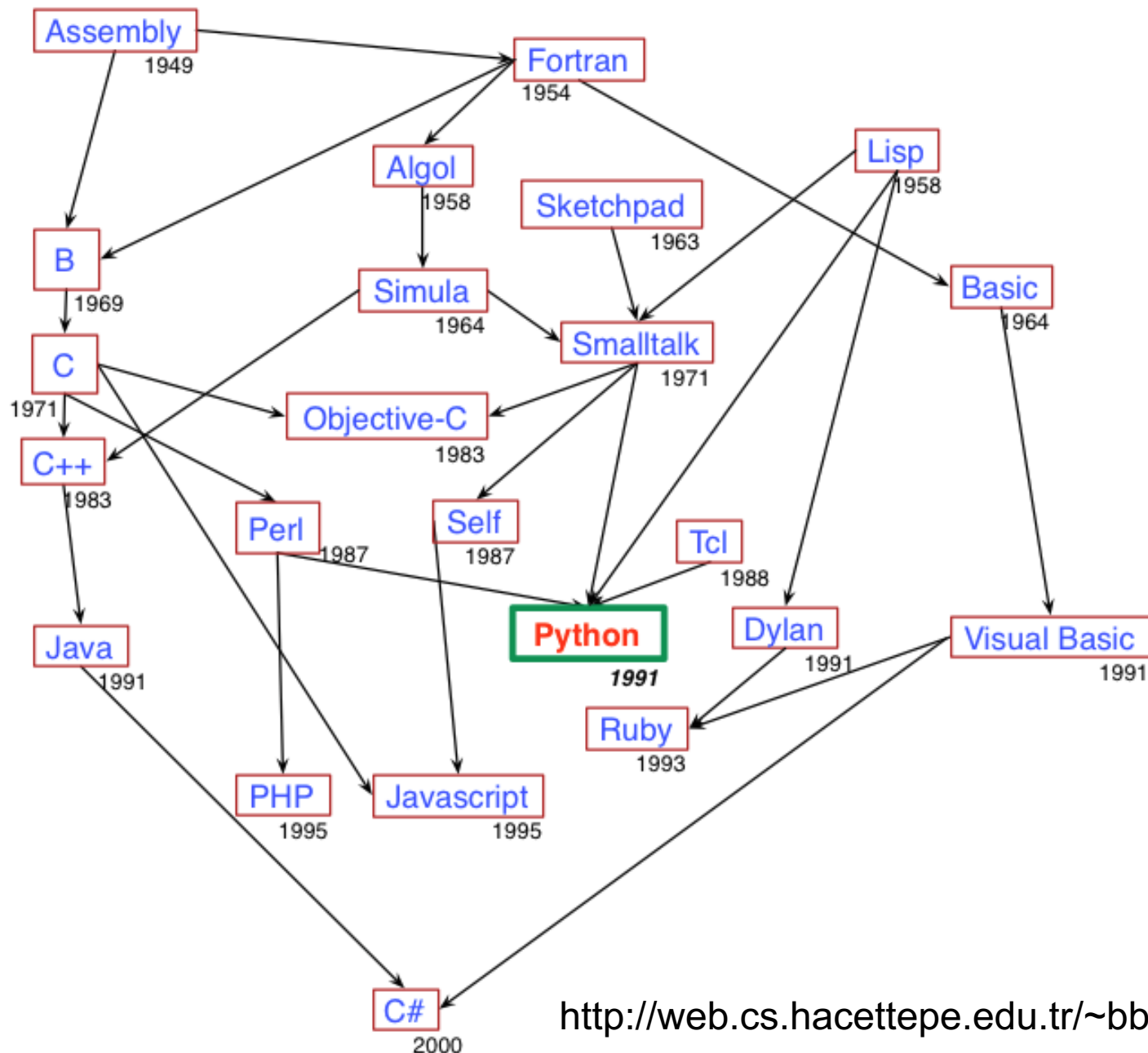
Why teaching C++

- Versatile
 - Python \geq C++ > Matlab
- 易于掌握
 - Python (free) > Matlab (commercial)
- 性能
 - C++。是Matlab和Python的 necessary supplement。
- Prerequisites
 - **Proficiency in Python, high-level familiarity in C/C++**
 - All class assignments will be in Python, but some of the deep learning libraries we may look at later in the class are written in C++.
 - If you have a lot of programming experience but in a different language (e.g. C/C++/Matlab/Javascript) you will probably be fine.

Why teaching C++

- Versatile
 - Python \geq C++ > Matlab
- 易于掌握
 - Python (free) > Matlab (commercial)
- 性能
 - C++。是Matlab和Python的必要补充。
- Java, Matlab & Python 不适合学习数据结构和算法
- The most of **libraries** for science computation are still implemented in C++.
- 其它语言不够 **hard**, C++可以用来区分great programmers and mediocre programmers.

Evolution of Programming Languages



- 其实这么多年我看着各种库的起起落落，还有一种感慨是研究者**不能始终抱着一个大腿**，要与时俱进。但是时代的潮流在哪里也不是随时都能看出来的，也没法时刻保持自己在前沿，但好在**掌握了一个库之后再换另一个库并不是很费劲**。

- --CMU LTI博士研究生
王赞

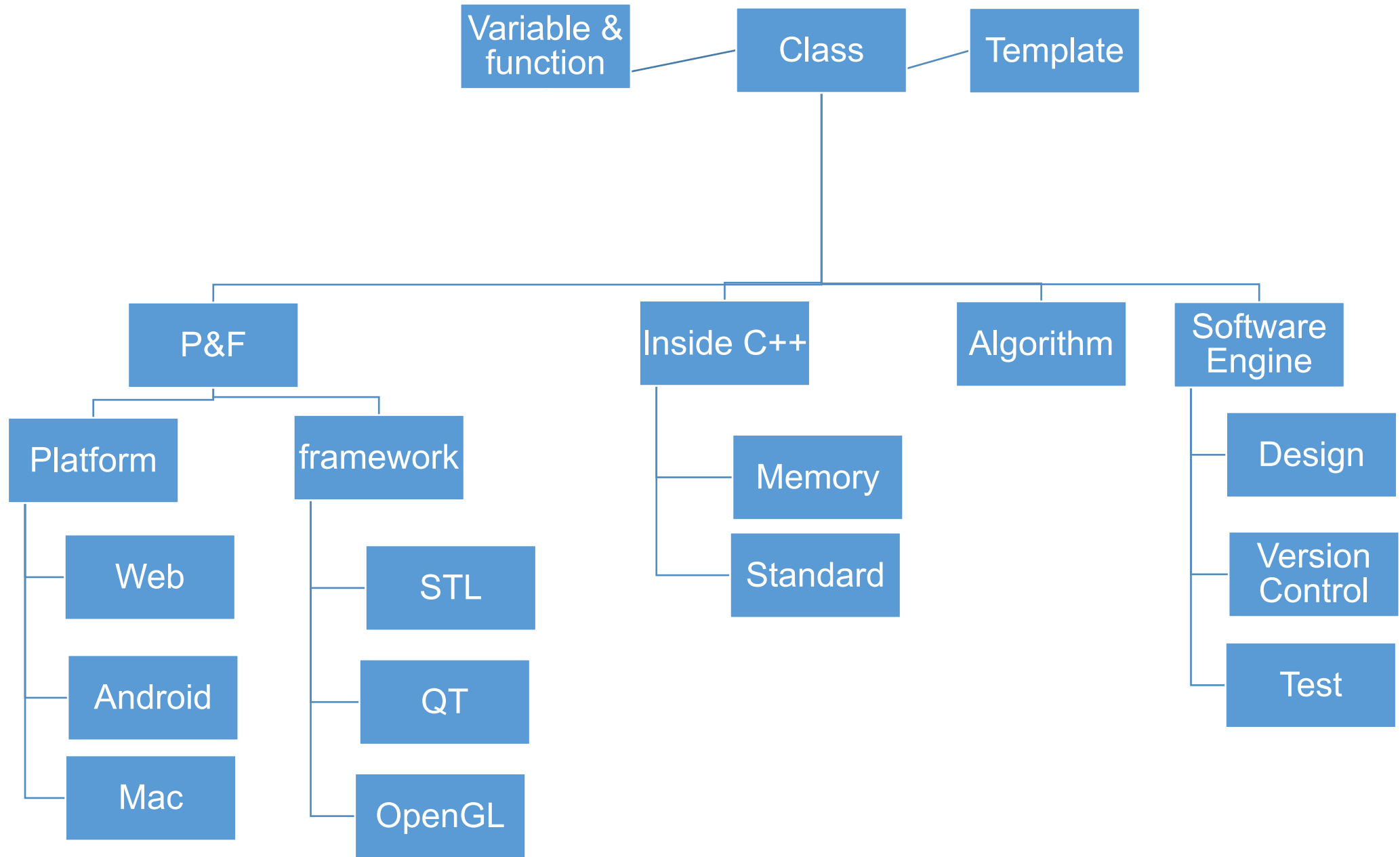
课程相关

- 教师

- [Junjie Cao](http://jjcao.github.io), <http://jjcao.github.io>

- jjcao@dlut.edu.cn

- 主页: <http://jjcao.github.io/cPlusPlus/>



考核

item	ratio
签到、日常测试、作业	30%
Exam	70%

How to Succeed?

- 56 hours (32 talks + 24 practices) in 4 weeks
- 每个人都可以
 - Work hard
 - 精英日课2: 正确的学习方法只有一种风格
 - 多做编程练习胜过多看书
 - “少想多做”，落实到editor内；
 - 增量开发，确保每一步可运行：
 - void main() first
 - Function 1
 - Function 2
 - ...
 - Debug your code
 - 英文搜索错误信息, Google!!!
 - Learn by good example: follow open source projects
 - 代码行数 约等于 编程能力



对数学的学生而言；
进阶要求

Video

- [The birth of the computer](#), George Dyson
- [SageMath – Open source is ready to compete with Mathematica for use in the classroom](#), William Stein

程序员 vs 程序猿



控制台程序(**Console programs**)

远比图形接口程序容易实现和迁移到
不同的操作系统

Hello World

```
// A Hello World program
# include <iostream>
int main()
{
    std::cout << "Hello, world!\n";
    return 0;
}
```


Line-By-Line Explanation

- `// 注释comment`

indicates that everything following it until the end of the line is a **comment**: it is ignored by the compiler.

- `/* and */`

- (e.g. `x = 1 + /*sneaky comment here*/ 1;`)
- multiple lines;

```
// A Hello World program
# include <iostream>
int main() {
    std::cout << "Hello, world!\n";
    return 0;
}
```

- Usages

- Comments exist to **explain non-obvious** things going on in the code. Use them: **document** your code well!

```
// A Hello World program
# include <iostream>
int main() {
    std::cout << "Hello, world!\n";
    return 0;
}
```

- **# preprocessor commands**

- 用#开始的行是预处理命令(preprocessor commands), which usually change what code is actually being compiled.
- **#include** tells the **preprocessor** to dump in the contents of another file, here the iostream file, which defines the procedures for input/output.

```
// A Hello World program
# include <iostream>
int main() {
    std::cout << "Hello, world!\n";
    return 0;
}
```

- **int main()**

- main 函数名
- 跟随main的()说明它是一个函数
- main()之前的int表明该函数返回一个整数值
- 当程序被执行（载入内存），main()是第一个被执行的函数（程序的入口）

```
// A Hello World program
# include <iostream>
int main() {
    std::cout << "Hello, world!\n";
    return 0;
}
```

- 大括号{}表明main()的函数体

- {}把多个命令组成一组命令：multiple commands =》 a block代码块
- 每一个命令/声明（command/statement）必须分号结尾
- More about this syntax in the next few lectures.

```
// A Hello World program
#include <iostream>
int main() {
    std::cout << "Hello, world!\n";
    return 0;
}
```

- `cout <<`
- This is the syntax for outputting some piece of text to the screen.

```
// A Hello World program
# include <iostream>
int main() {
    std::cout << "Hello, world!\n";
    return 0;
}
```

- **std**是一个名称空间Namespaces
 - 作用域解析操作符scope resolution operator ::
 - 通知编译器要调用std中的cout，而不是别处jjcao::cout

using namespace std;

- This line tells the compiler that it should look in the std namespace for any identifier we haven't defined.
- If we do this, we can omit the std:: prefix when writing cout.

```
// A Hello World program
#include <iostream>
int main() {
    std::cout << "Hello, world!\n";
    return 0;
}
```

- 字符串String
 - *Hello, world*
 - 像这样显示指定的字符串，叫string literal.字符串字面量
- \n
 - The \n indicates a **newline** character.
 - 转义序列（Escape sequences）：It is an example of an escape sequence – a symbol used to represent a special character in a text literal.

```
// A Hello World program
#include <iostream>
int main() {
    std::cout << "Hello, world!\n";
    return 0;
}
```

- **return 0**

- 通知OS，本程序成功执行完毕。
- 是main block的最后一行

- **注意**

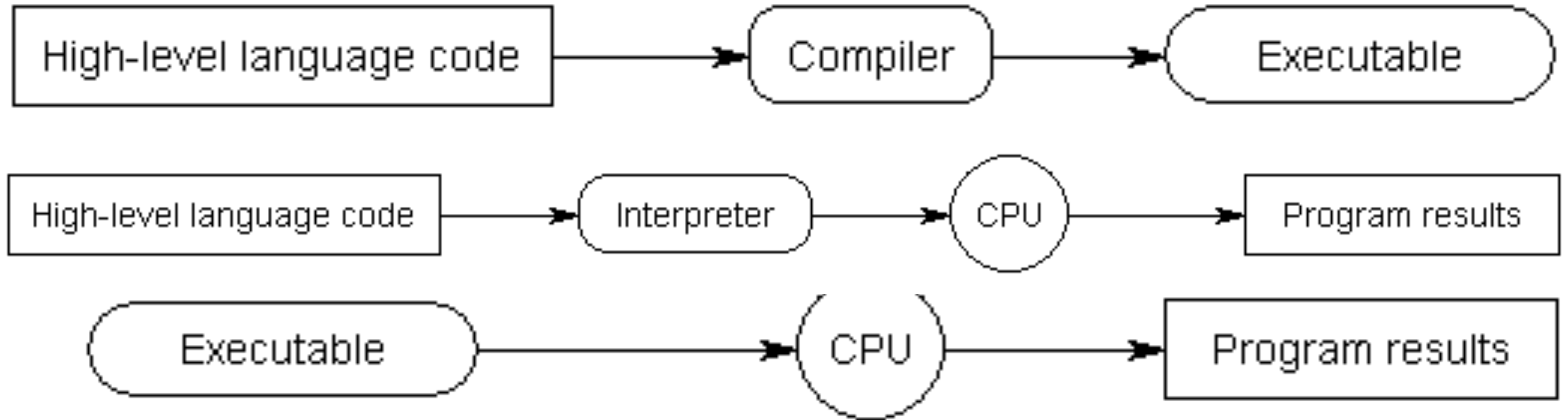
- 每一个声明需要分号结束（预处理命令和{}除外（如果是定义class的时候，{}也要跟着分号））
- 忘记分号，是新手常犯错误

The Compilation Process

Our language v.s. binary language the computer used

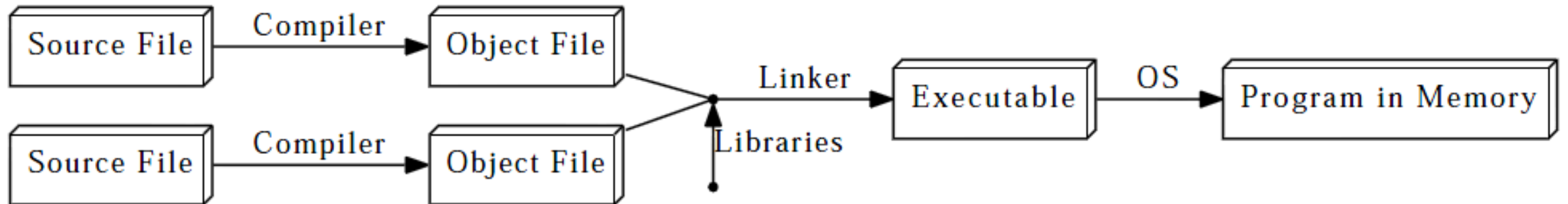
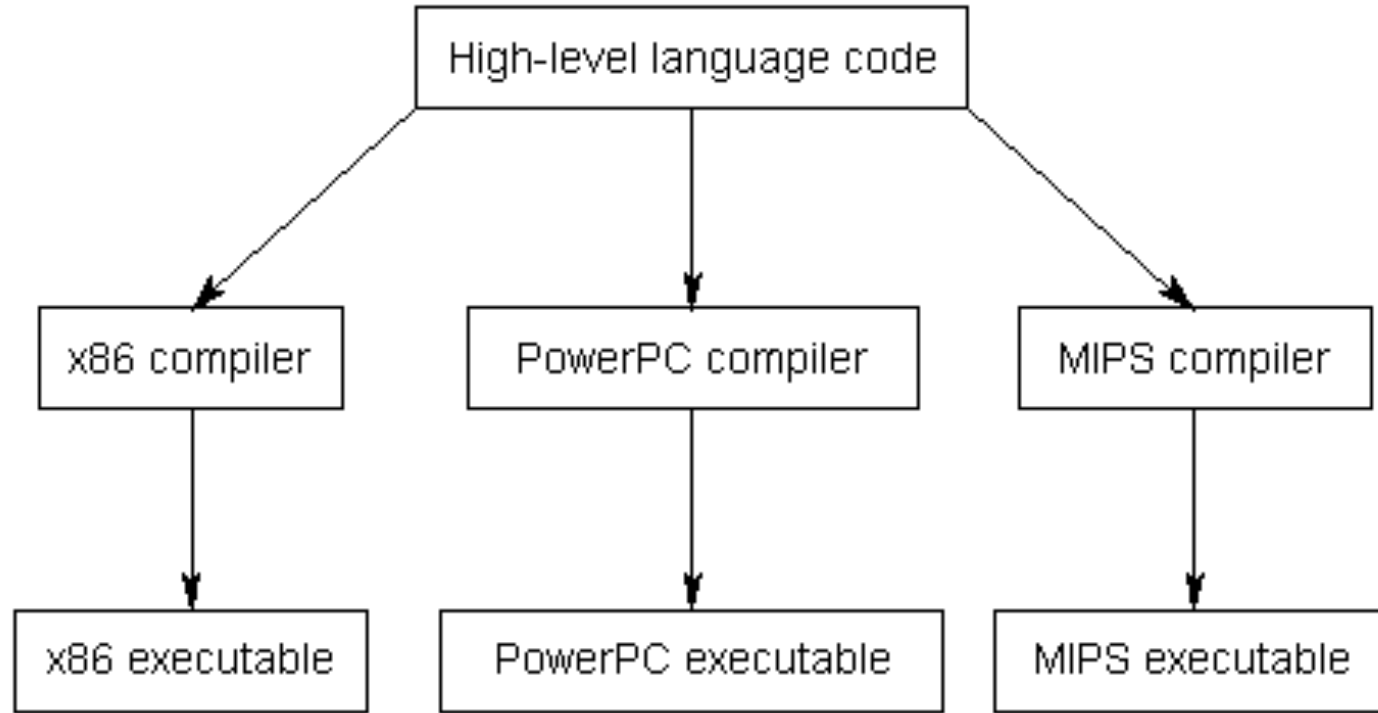
C++ is like natural language

Compiler: make computer understand C++



The Compilation Process

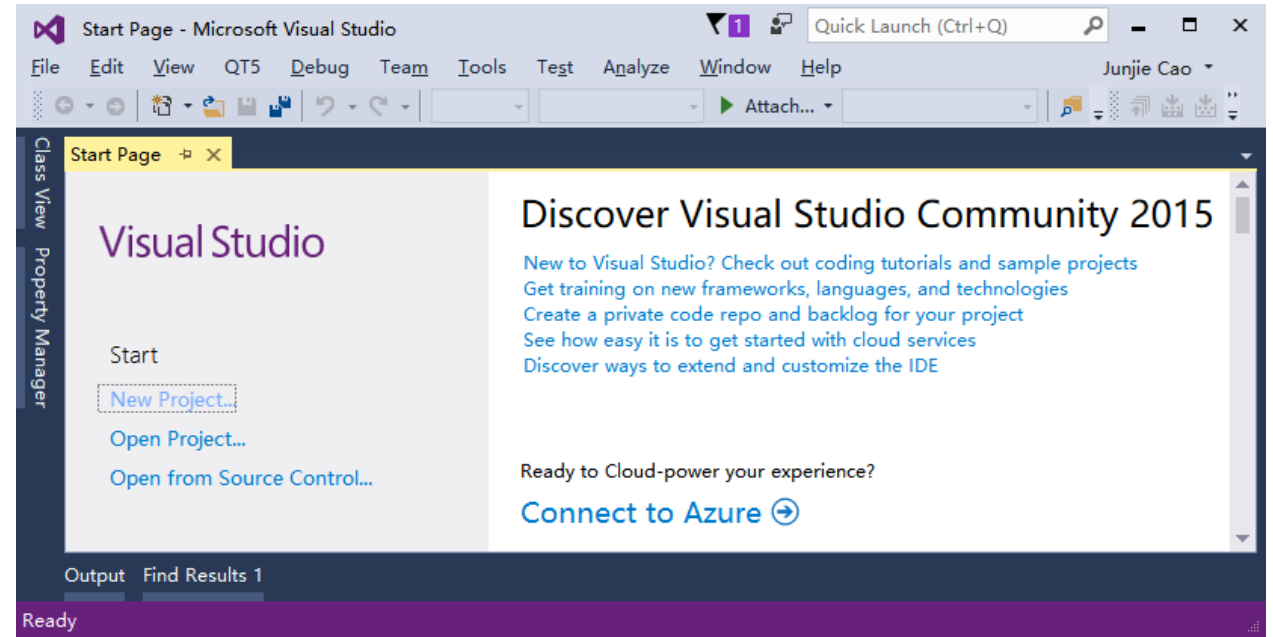
Compiler: make computer understand C++



Integrated Development Environment

集成开发环境IDE

- Visual C++: Windows
- Code::Blocks: Linux
- Xcode, Eclipse: Mac



- **CodeChef: Web based**

- Web-based compilers are fine for dabbling and simple exercises. However, they are generally quite limited in functionality -- many won't allow you to save projects, create executables, or effectively debug your programs. You'll want to migrate to a full IDE when you can.

- [Installing an Integrated Development Environment \(IDE\)](#)



编译你的第一个程序

- [lab01_IDE_VC_Win32ConsoleApplication.pptx](#)
- [LearnCpp.com](#)

C and C++'s philosophy能力与责任

- Underlying design philosophy: “**trust the programmer**”
 - Wonderful
 - compiler will not stand in your way if you try to do something unorthodox that makes sense,
 - Dangerous
 - compiler will not stand in your way if you try to do something that could produce unexpected results.
 - That is one of the primary reasons why knowing **what you shouldn't do** in C/C++ is almost **as important as knowing what you should do** -- because there are quite a few pitfalls that new programmers are likely to fall into if caught unaware.



Reference Courses

- [cpp for school](#)
 - simpler and with assignments, projects, quiz and papers.
- [LearnCpp.com](#)
 - more detail explanations than cpp for school

Reference Books

1. C++ Primer

2. The C++ Programming Language. (more advance than 1)
3. The C++ Standard Library – A Tutorial and Reference
4. Teach Yourself C++ in One Hour a Day
5. Code complete 2nd
6. Clean Code A Handbook of Agile Software Craftsmanship

Useful Links

- <http://www.cplusplus.com>

Academic Integrity

- Honest work is required of a scientist or engineer.
- Integrity is the key for everything!!!
- Discussion is permitted.
- Everything you turn in must be your own work.
- Cite your sources, explain any unconventional action.
- If you have a question, ask.