

面向对象程序设计基础

(OOP)

刘知远

`liuzy@tsinghua.edu.cn`

`http://nlp.csai.tsinghua.edu.cn/~lzy/`

课程团队：刘知远 姚海龙 黄民烈

自我介绍

- 单位：计算机系智能技术与系统国家重点实验室
- 研究方向：自然语言处理，知识图谱
- 邮件：liuzy@tsinghua.edu.cn
- 手机（微信）：138 1032 5978
- 主页：<http://nlp.csai.tsinghua.edu.cn/~lzy/>
- 办公室：FIT楼4-506

课程助教

- TA
 - 肖朝军 xcjthu@gmail.com
 - 秦禹嘉 qyj20@mails.tsinghua.edu.cn
 - 胡声鼎 hsd20@mails.tsinghua.edu.cn

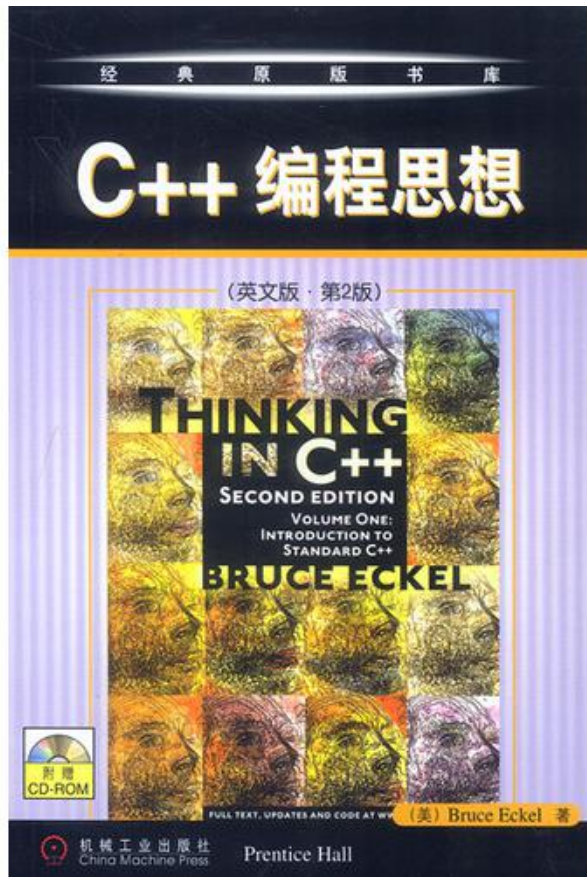
什么是面向对象的编程

- ☐ A 是一门很虐的编程课。
- ☒ B 建立模型体现抽象思维。
- ☐ C 把所有代码都封装在类里。
- ☐ D 与对象一起协作编程。😊

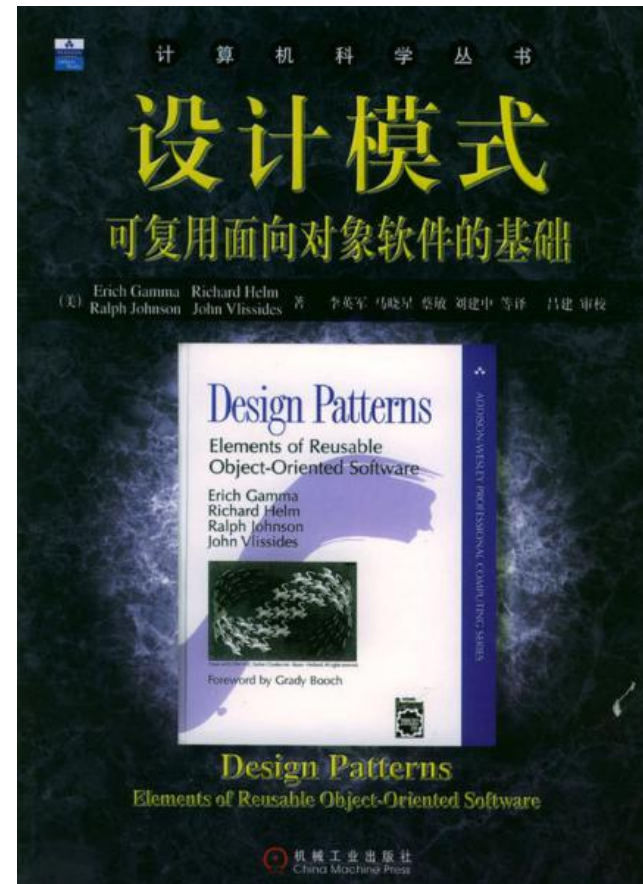
提交

参考书目

C++编程思想
(2017年8月重印)



设计模式



参考书目 (1)

- C++ primer plus (第6版) Stephen Prata (2012年7月 第1版)
 - 人民邮电出版社, 张海龙等 译
- C++编程思想 第2版 第1卷: 标准C++导引 Bruce Eckel
 - 机械工业出版社, 2002.9, 刘宗田等 译
- 深入理解C++11 — C++11新特性解析与应用
 - 机械工业出版社, 2015.5, Michael Wong等 著
- Effective C++ 中文版 2nd Edition Scott Meyers
 - 华中科技大学出版社, 2001.9, 侯捷 译
- 设计模式—可复用面向对象软件的基础 Erich Gamma, etc...
 - 机械工业出版社, 2000.9, 李英军等 译

参考书目 (2)

■ C++标准程序库 Nicolai M.Josuttis

- 华中科技大学出版社, 2002.9, 侯捷、孟岩 译

■ C++沉思录 (Ruminations on C++) Andrew Koenig, Barbara Moo

- 机械工业出版社, 2002.7, 黄晓春 译, 孟岩 校

■ 深度探索C++对象模型 Stanley B.Lippman

- 华中科技大学出版社, 2001.5, 侯捷 译

■ 深入实践C++模板编程

- 机械工业出版社, 2013.6, 温宇杰 著

■ C++语言的设计和演化 Bjarne Stroustrup

- 机械工业出版社, 2002.1, 裘宗燕 译

其他参考

■ www.cplusplus.com

The screenshot shows the homepage of www.cplusplus.com. The header includes the site logo, a search bar, and links for 'Not logged in', 'register', and 'log in'. A left sidebar lists navigation options: 'C++', 'Information', 'Tutorials', 'Reference', 'Articles', and 'Forum'. The main content area is divided into six sections: 'Information' (general C++ info), 'Tutorials' (learning C++ from basics), 'Reference' (Standard Library details), 'Articles' (user-contributed content), 'Forum' (message boards), and 'C++ Search' (website search). Each section contains descriptive text and a list of links or resources.

Information
General information about the C++ programming language, including non-technical documents and descriptions:

- Description of the C++ language
- History of the C++ language
- F.A.Q., Frequently Asked Questions

Tutorials
Learn the C++ language from its basics up to its most advanced features.

- C++ Language: Collection of tutorials covering all the features of this versatile and powerful language. Including detailed explanations of pointers, functions, classes and templates, among others...
- more...

Reference
Description of the most important classes, functions and objects of the Standard Language Library, with descriptive fully-functional short programs as examples:

- C library: The popular C library, is also part of the of C++ language library.
- IOStream library. The standard C++ library for Input/Output operations.
- String library. Library defining the string class.
- Standard containers. Vectors, lists, maps, sets...
- more...

Articles
User-contributed articles, organized into different categories:

- Algorithms
- Standard library
- C++11
- Windows API
- Other...

You can contribute your own articles!

Forum
Message boards where members can exchange knowledge and comments. Ordered by topics:

C++ Search
Search this website:

课程内容的定位

■ 算法基础

- 问题的分析、表示、求解的方法
- 变量、判断、循环、函数的应用

■ 语言基础

- 如何定义和实现算法中的元素
 - 语法规范
-

■ 系统基础

- 操作系统运作的内在机制、机理
- 操作系统提供的底层功能调用库

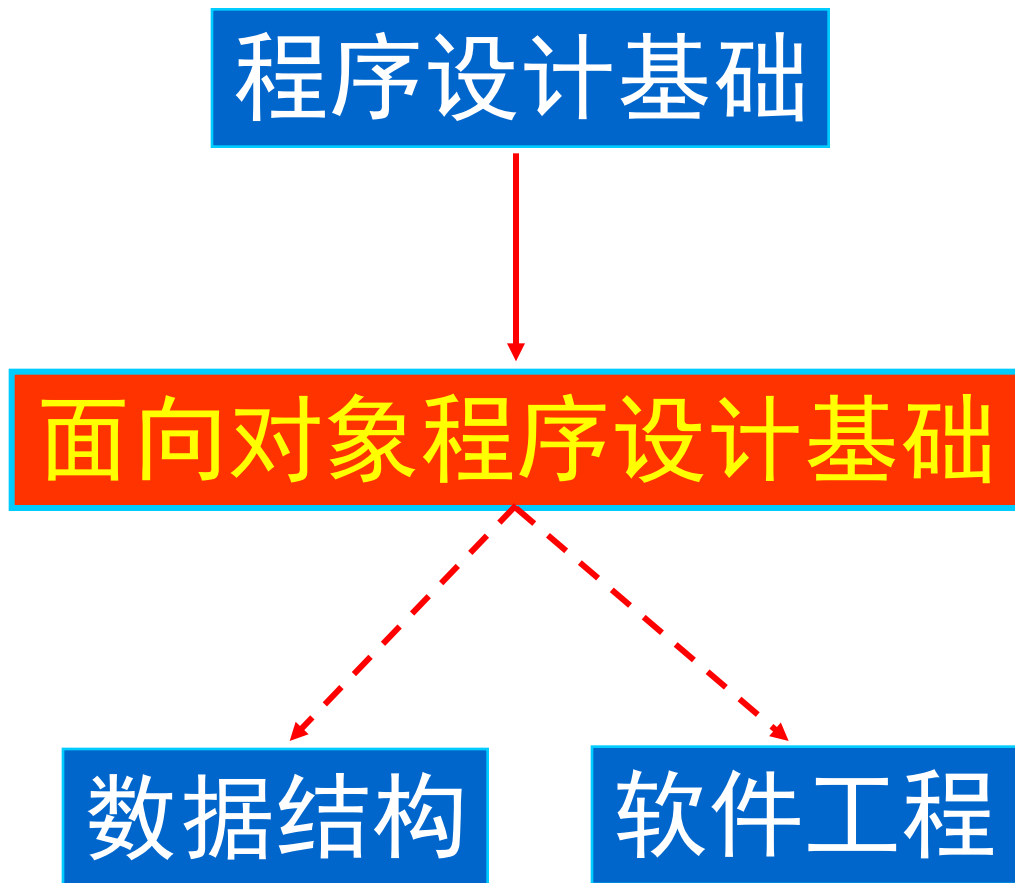
■ 技术基础

- 如：网络通信、硬件接口
-

■ 方法（论）基础

- 如：结构化，基于对象，面向对象，泛型，组件
- 越复杂越重要

先导与后继课程关系



课程目标—编写更好的软件

■ 简单性

- 使程序短而容易管理

■ 清晰性

- 好的可读性；保证程序容易理解，无论是对人还是对机器

■ 普遍性

- 程序在很广泛的情形下都能工作得很好，也容易做修改以适应新出现的情况

如何使自己的程序满足上述三条原则？

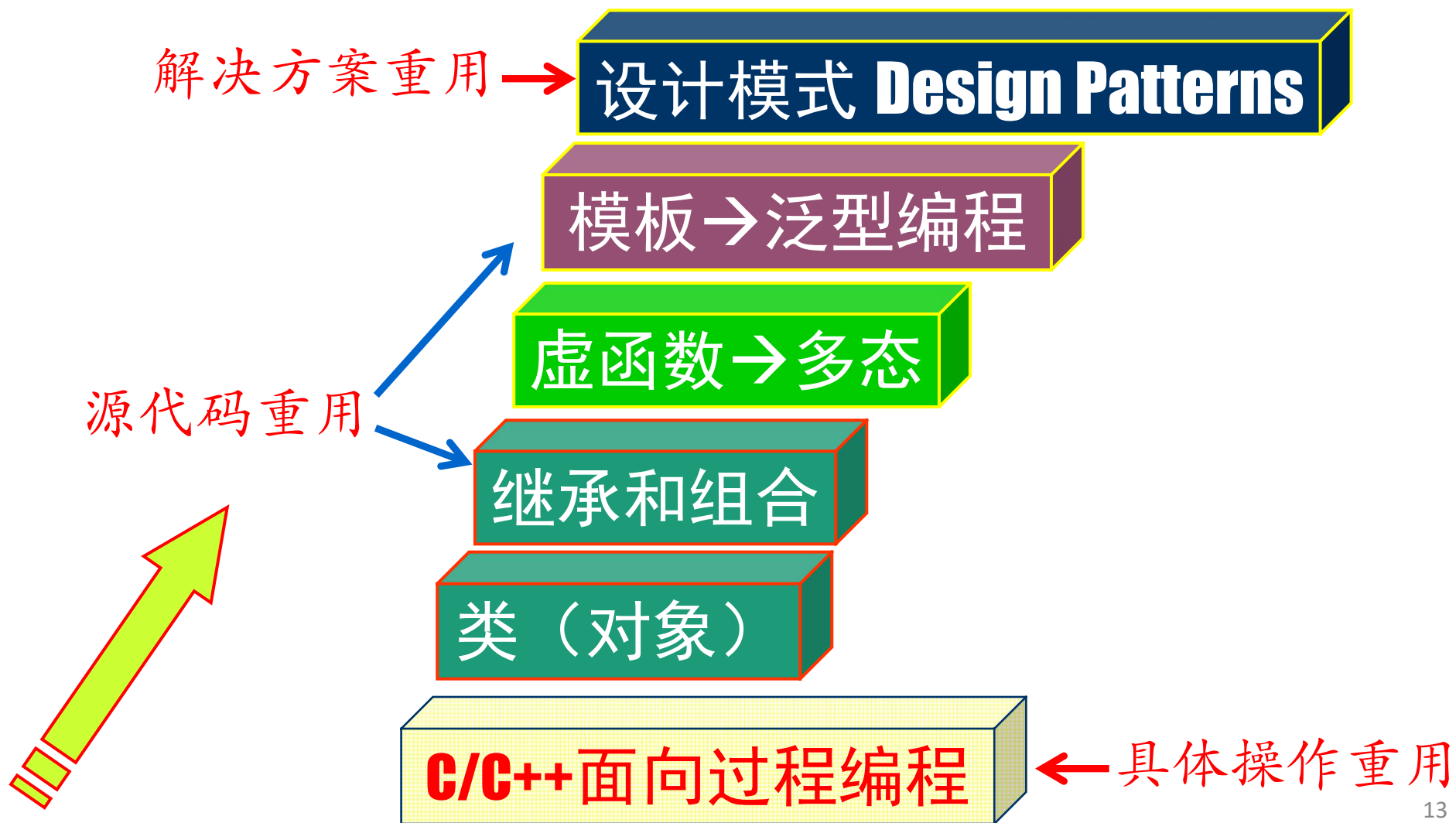
为什么选择 C++ 语言?

Jan 2020	Jan 2019	Change	Programming Language	Ratings	Change
1	1		Java	16.896%	-0.01%
2	2		C	15.773%	+2.44%
3	3		Python	9.704%	+1.41%
4	4		C++	5.574%	-2.58%
5	7	▲	C#	5.349%	+2.07%
6	5	▼	Visual Basic .NET	5.287%	-1.17%
7	6	▼	JavaScript	2.451%	-0.85%
8	8		PHP	2.405%	-0.28%
9	15	▲	Swift	1.795%	+0.61%
10	9	▼	SQL	1.504%	-0.77%
11	18	▲	Ruby	1.063%	-0.03%
12	17	▲	Delphi/Object Pascal	0.997%	-0.10%
13	10	▼	Objective-C	0.929%	-0.85%
14	16	▲	Go	0.900%	-0.22%
15	14	▼	Assembly language	0.877%	-0.32%

**TIOBE编程语言指数排行表 TOP20
(2019.1~2020.1)**

20	11	▼	MATLAB	0.737%	-0.76%
----	----	---	--------	--------	--------

课程内容中的几个台阶



OOP课程与FOP课程的区别

■ 程序设计基础 (FOP)

- 数字化
- 可计算

重点解决 “怎么算” ?

➔ 培养 “计算思维”，通过计算解决复杂问题

■ 面向对象程序设计基础 (OOP)

- 人性化
- 易认知

重点解决 “怎么看” ?

➔ 培养 “抽象思维”，通过抽象认知复杂世界

OOP是一种编程设计的**方法论**

- 如何直观分析问题?
- 如何快速实现算法?
- 如何方便修改代码?

高效实现程序，解决
程序员的开发效率

OOP

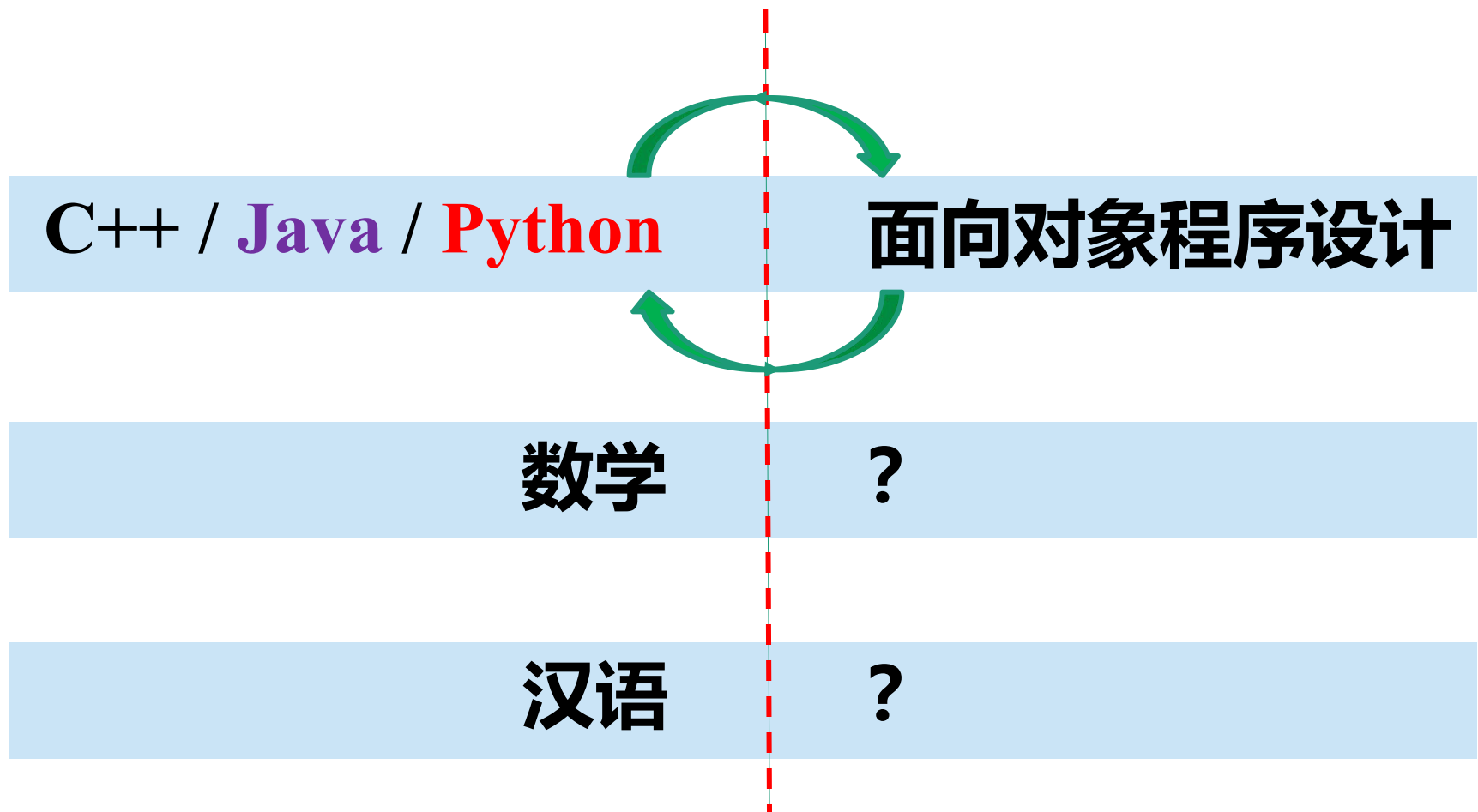
实现高效程序，解决
计算机的运行效率

FOP

语言与思维 (1)

- 人类的任何思维活动都是借助于他们所熟悉的某种自然语言进行的。

语言与思维 (2)



语言与思维 (3)

■ 面向对象方法强调的基本原则

- 人类在认识世界的历史进程中所形成的普遍有效的思维方法，在软件开发中也应该是适用的。
- 人们在日常生活中习惯的思维方式和表达方式，也应在软件开发中尽量采用。

语言与思维 (4) 什么是“对象”？

- 例子（春联）：一元复始，万象更新。
- 对象是对现实世界中实际事物的一种抽象描述，它可以是有形的实体，也可以是无形的概念。
- 对象是构成世界的一个独立单位，它具有自己的静态特征和动态特征。
 - 静态特征：可以用某种数据来描述的特征
 - 动态特征：对象所表现的行为或所具有的功能
- 对象由一组属性和对这组属性进行操作的一组服务构成，是属性和服务的结合体。

语言与思维 (5) 什么是“抽象”？

- 从许多事物中舍弃个别的、非本质性的特征，抽取共同的、本质性的特征，就叫做抽象。
- 抽象是形成概念的必要手段：
 - **过程抽象**：任何一个完成确定功能的操作序列，其使用者都可以把它看作一个单一的实体。
 - **数据抽象**：根据施加于数据上的操作来定义数据类型，并限定数据的值只能由这些操作来修改和观察。

举例：数据到底是什么？

——数学家怎么说(1)...

定义 8.1 设 K 是至少含一个非零数的数集. 如果 K 中任意两个数的和、差、积、商(除数不为零)仍是 K 中的数, 则称 K 为一个数域.

容易验证, 全体有理数集合 Q 、全体实数集合 R 、全体复数集合 C 都构成数域, 分别称为有理数域、实数域、复数域; 而自然数集和整数集都不是数域. 可以证明,

在数学上, “数据” 不仅有值, 而且限定了在值上的操作规范 (约束) 和性质。

程序设计语言中的“类型”的概念与此相似, 如:

“整数相除得整数” : $3/4=0$;

“浮点数相除得浮点数” : $3.2/4.0=0.8$ 。

举例：数据到底是什么？

——数学家怎么说(2)...

定义 8.4 设 V 是一个非空集合, K 是一个数域. 在 V 的元素之间规定了称之为“加法”的运算, 在 K 与 V 的元素之间规定了称之为“数乘”的运算. 如果 V 对于这两种运算封闭, 即对任意 $\alpha, \beta \in V$ 都有 $\alpha + \beta \in V$, 而对任意 $k \in K$ 和 $\alpha \in V$ 都有 $k\alpha \in V$, 且这两种运算满足以下八条运算律(设 $\alpha, \beta, \gamma \in V, k, l \in K$):

- (1) 加法交换律 $\alpha + \beta = \beta + \alpha$,
- (2) 加法结合律 $(\alpha + \beta) + \gamma = \alpha + (\beta + \gamma)$,
- (3) 存在零元 θ , 使 $\alpha + \theta = \alpha$,
- (4) 存在负元, 即对任意 $\alpha \in V$, 存在 $\beta \in V$, 使 $\alpha + \beta = \theta$, 称 β 为 α 的负元, 记为 $-\alpha$,
- (5) $1\alpha = \alpha$,
- (6) 数乘结合律 $k(l\alpha) = (kl)\alpha$,
- (7) 分配律 $(k + l)\alpha = k\alpha + l\alpha$,
- (8) 数因子分配律 $k(\alpha + \beta) = k\alpha + k\beta$,

则称 V 为数域 K 上的线性空间.

举例：数据到底是什么？ ——程序员怎么说...

我们来看看是什么构成了类型。例如，讨厌鬼是什么？受流行的固定模式影响，可能会指出讨厌鬼的一些外表特点：胖、戴黑宽边眼镜、兜里插满钢笔等。稍加思索后，又可能觉得从行为上定义讨厌鬼可能更合适，例如他（或她）是如何应对尴尬的社交场面的。如果将这种类比扩展到过程性语言（如 C 语言），我们得到类似的情形。首先，倾向于根据数据的外观（在内存中如何存储）来考虑数据类型。例如，char 占用一个字节的内存，而 double 通常占用 8 个字节的内存。但是稍加思索就会发现，也可以根据要对它执行的操作来定义数据类型。例如，int 类型可以使用所有的算术运算，可对整数执行加、减、乘、除运算，还可以对它们使用求模操作符（%）。

而指针需要的内存数量很可能与 int 相同，甚至可能在内部被表示为整数。但不能对指针执行与整数相同的运算。例如，不能将两个指针相乘，这种运算没有意义的，因此 C++ 没有实现这种运算。因此，将变量声明为 int 或 float 指针时，不仅仅是分配内存，还规定了可对变量执行的操作。简而言之，指定基本类型完成了两项工作：

- 决定了数据对象需要的内存数量。
- 决定了可使用数据对象执行的操作或方法。

对于内置类型来说，这些信息被内置到编译器中。但在 C++ 中定义用户自定义的类型时，必须自己提供这些信息。付出这些劳动换来了根据实际需要定制新数据类型的强大功能和灵活性。

类型 = 数据的存储与表示 + 数据支持的操作

学习方法

■ 多看

- 相关书籍、网站等

■ 多想

- 思考和理解高层的抽象方法
- 思考和理解底层的运行机制

■ 多练

- 练习：自己给自己出题目
- 疑问：验证自己的理解

■ 多问

- 问题：试过后还理解不了的

■ 多记

- 规纳和整理(而不是纯记忆)自己所得



考核要求 (2021)

- 平时作业练习 约6次 70%
 - 网络学堂发布, UOJ 平台上提交
 - 自动评测得分 + 助教评阅
- ~~期中考试~~
 - ~~语法知识, 闭卷, 选择题~~
- 期末考试 机试 25%
 - 编程
- ~~大作业 (分小组合作完成)~~
- ~~期末总结 (对某个C++主题的研究小报告)~~
- 雨课堂随堂测试 5%
 - 取得分最高的80%部分

教学计划

3课时：绪论、基础编程知识
1课时：对象的基础知识
2课时：对象的创建与销毁
1课时：对象的引用与复制
1课时：对象的组合与继承
2课时：虚函数与多态
3课时：模板与STL、STL进阶
2课时：案例与设计模式

课时总计：15课时
放假占用：1课时

第17/18周：期末考试

课程答疑

- <https://github.com/thunlp/OOP-THU>



小教员招募

■要求：如果你有扎实的C++编程基础，熟悉面向对象设计模式，学有余力，可以报名课程小教员

■职责：

- 主动、耐心地帮助同学解决课程相关的问题
- 记录自己解决的问题，参与课程FAQ列表的建设
- 编写与课程相关的练习题

■选拔和分数评价：

- 需要通过小教员的选拔考试（机试）
- 需要完成答疑和出题任务，表现优秀的可能会额外加分
- 工作量达标的小教员期末考试免参加、记满分

■具体细节之后将在网络学堂通知

本讲内容提要

- 1.1 命令提示符
- 1.2 环境变量设置
- 1.3 主流编译器及IDE
- 1.4 ssh远程登录与操作

命令提示符 (命令行)

- 命令提示符是在操作系统中，提示进行命令输入的一种工作提示符
- 打开命令提示符
 - Windows:
 - 右键->在此处打开命令窗口
 - 新建一个目标为cmd的快捷方式，(以管理员身份)打开快捷方式
 - MAC OS X:
 - LaunchPad中打开 “终端” 程序
 - Command+Space激活Spotlight搜索，输入“terminal.app”
 - Linux:
 - 打开 “终端” 程序
 - 快捷键 (Ctrl+Alt+t)

命令提示符 (命令行)

```
C:\ 命令提示符

D:\>type ex2.cpp
// ex2.cpp © 20090831
#include <iostream>
#include <cstdio>      // atoi()
int main(int argc, char** argv)
```

```
Last login: Thu Feb  6 20:28:18 on ttys000
sillyxu@SHERMANHAN-MB0 ~$ ls
Applications      Pictures          age
Book_KRL          ProKil           eclipse-workspace
```

```
ubuntu@ubuntu: ~/tmp/dem1
ubuntu@ubuntu:~/tmp/dem1$ ls
file_in.txt  tcpclient.c  tcpserver.c
tcpclient    tcpserver
ubuntu@ubuntu:~/tmp/dem1$ vim tcpclient.c
ubuntu@ubuntu:~/tmp/dem1$ gcc -o tcpclient tcpclient.c
```

命令提示符(命令行)常见操作

■ 目录结构

- D:\example\ (Win)
- /home/username/example/ (Linux)
- /Users/username/example/ (MAC)

■ 显示当前目录

- cd (Win)
- pwd (Linux/MAC)

■ 在当前目录下新建OOP2021目录

- mkdir OOP2021 (Win/Linux/MAC)

■ 在当前目录下新建a.cpp文件

- type nul>a.cpp (Win)
- touch a.cpp (Linux/MAC)

命令提示符(命令行)常见操作

■ 查看当前目录下的文件

- `dir` (Win)
- `ls` (Linux/MAC)

■ 进入上一层目录

- `cd ..` (Win/Linux/MAC)

■ 进入OOP2021目录

- `cd OOP2021` (Win/Linux/MAC)

■ 删除a.cpp文件

- `del a.cpp` (Win)
- `rm a.cpp` (Linux/MAC)

■ 删除OOP2021目录以及目录下的所有文件

- `rmdir /s OOP2021` (Win)
- `rm -r OOP2021` (Linux/MAC)

命令提示符(命令行)常见操作

■ 将a.cpp移动至OOP2021目录

- `move a.cpp OOP2021\` (Win)
- `mv a.cpp OOP2021/` (Linux/MAC)

■ 将a.cpp拷贝至OOP2021目录

- `copy a.cpp OOP2021\` (Win)
- `cp a.cpp OOP2021/` (Linux/MAC)

■ 将a目录下的所有文件拷贝到OOP2021目录

- `xcopy /e a OOP2021` (Win)
- `cp -r a OOP2021` (Linux/MAC)

■ OS X上命令提示符与Linux基本一致

■ Windows上命令提示符与Linux较为相似

■ 更多命令提示符用法请大家课后自行学习与练习

Windows Subsystem for Linux (WSL)

■ Windows Subsystem for Linux (简称WSL)

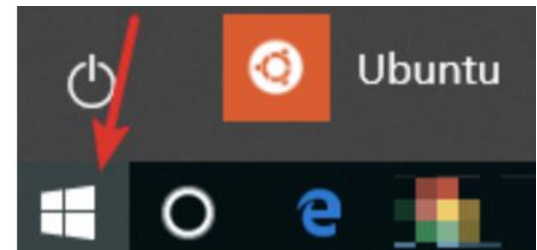
- 一个为在Windows 10上能够原生运行Linux二进制可执行文件的兼容层。

■ 安装步骤

- 控制面板->应用和功能->启用或关闭Windows功能->勾选 “适用于Linux的Windows子系统”
- 重启电脑
- 在微软应用商店搜索Linux，选择适合自己的发行版 (Ubuntu 18.04 LTS) 安装

■ 启动方式

- 在快捷启动方式中找到Ubuntu启动
- 在命令提示符中输入wsl



- 在/mnt目录下查看硬盘分区访问本地文件，
c,d,e文件夹分别表示本地的C盘、D盘、E盘

了解主流编译器

■ Windows上编译器需要自行安装

- MinGW: Minimalist GNU For Windows, 是个精简的Windows平台C/C++、ADA及Fortran编译器
- TDM-GCC: Windows版的编译器套件, 结合了GCC工具集中最新的稳定发行版本
- Clang: Clang是一个C语言、C++、Objective-C语言的轻量级编译器。
- MSVC: Visual Studio中自带的微软开发的C/C++语言编译器

■ Linux建议使用自带的g++编译器

- g++编译, 加-g选项编译配合gdb进行调试
- 该编译器为课程OJ使用的编译器

■ MAC建议使用自带的clang++

- clang++编译, 加-g选项编译配合lldb进行调试

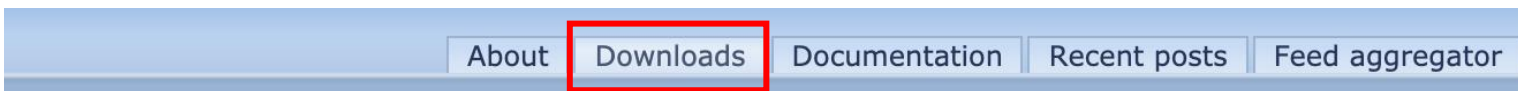
了解主流编译器

■Windows上如何安装编译器

- 一般DEV-C++安装会附带MinGW/TDM-GCC，可以直接使用DEV-C++中自带的编译器

■也可单独到官网下载安装编译器

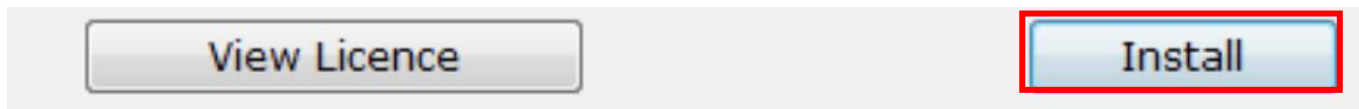
- 登录<http://www.mingw.org/>
- 点击网页右上角 Downloads



- 点击下载安装软件



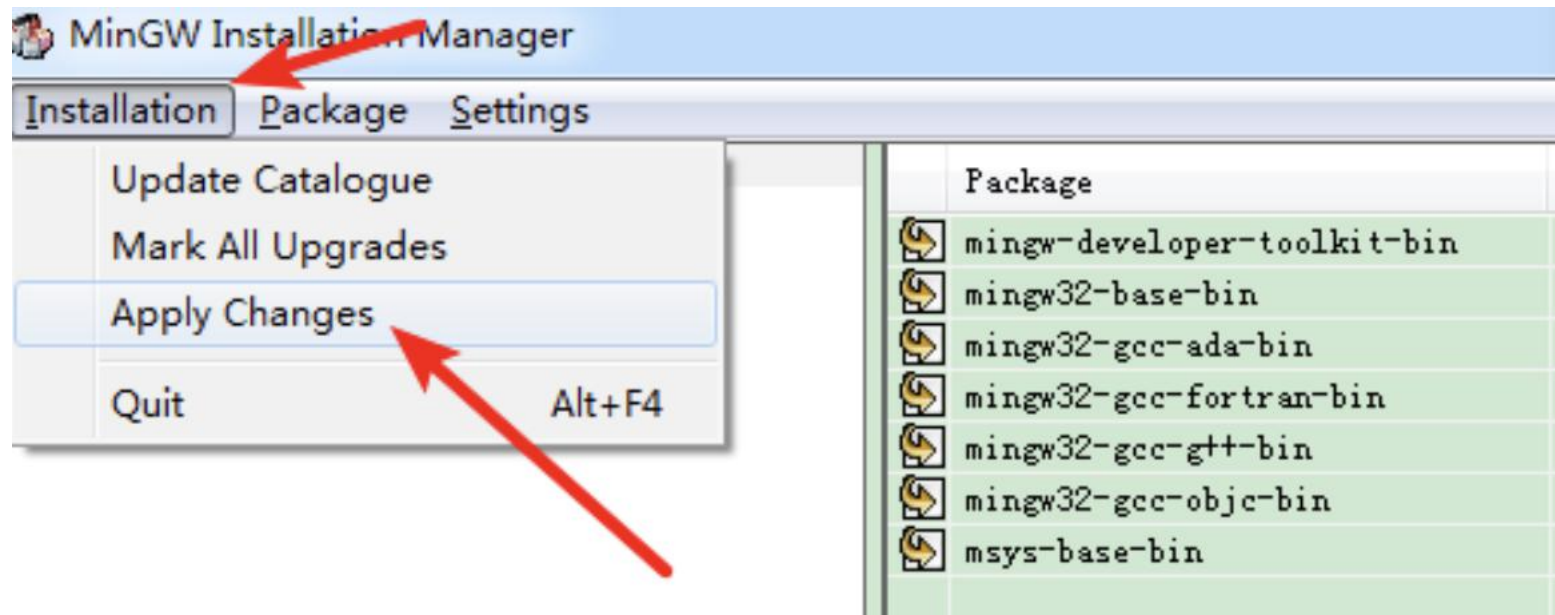
- 打开安装软件，点击安装、继续



了解主流编译器

■也可单独到官网下载安装编译器

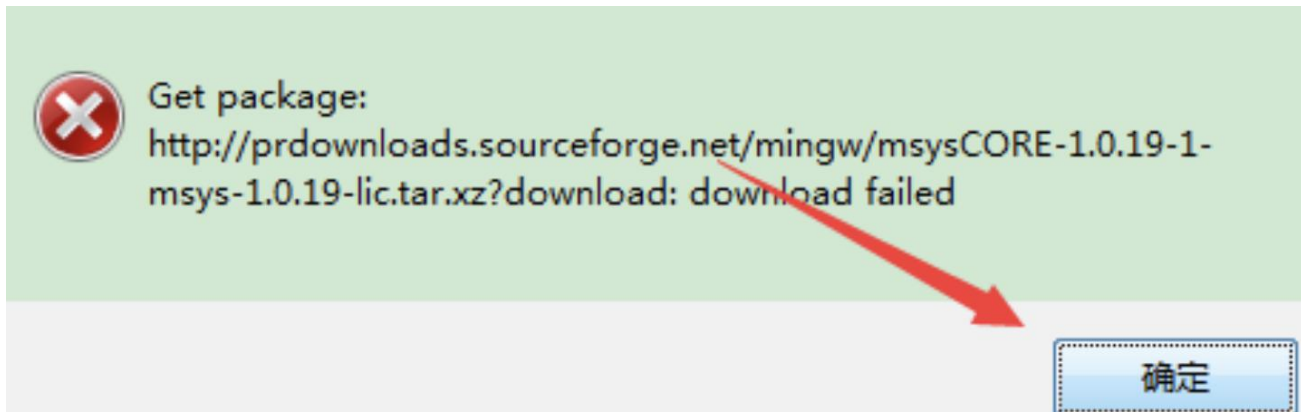
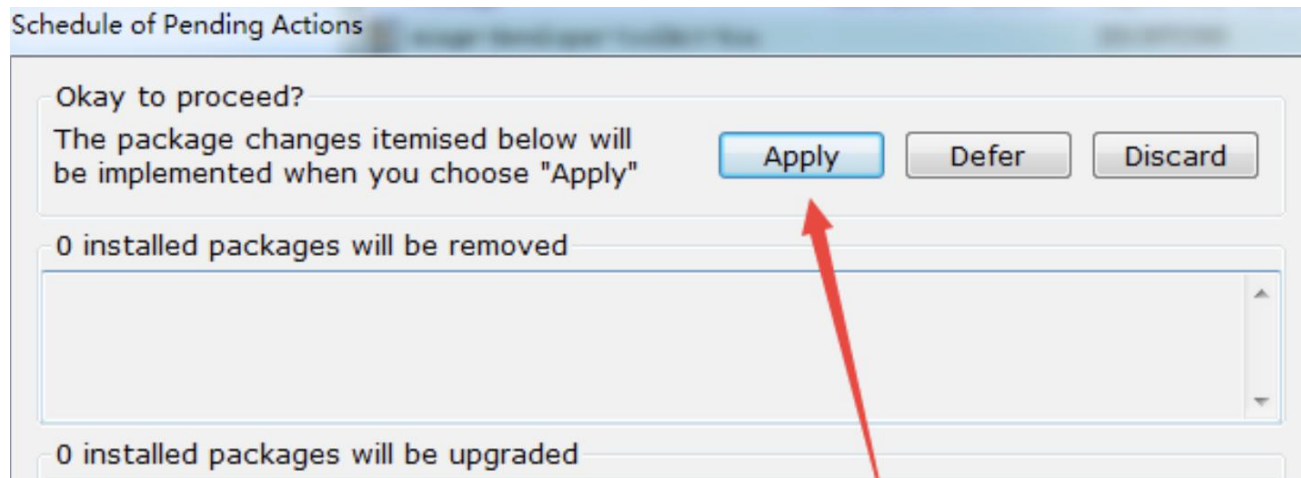
- 把Basic Setup都点满，点击Apply Changes安装



了解主流编译器

■也可单独到官网下载安装编译器

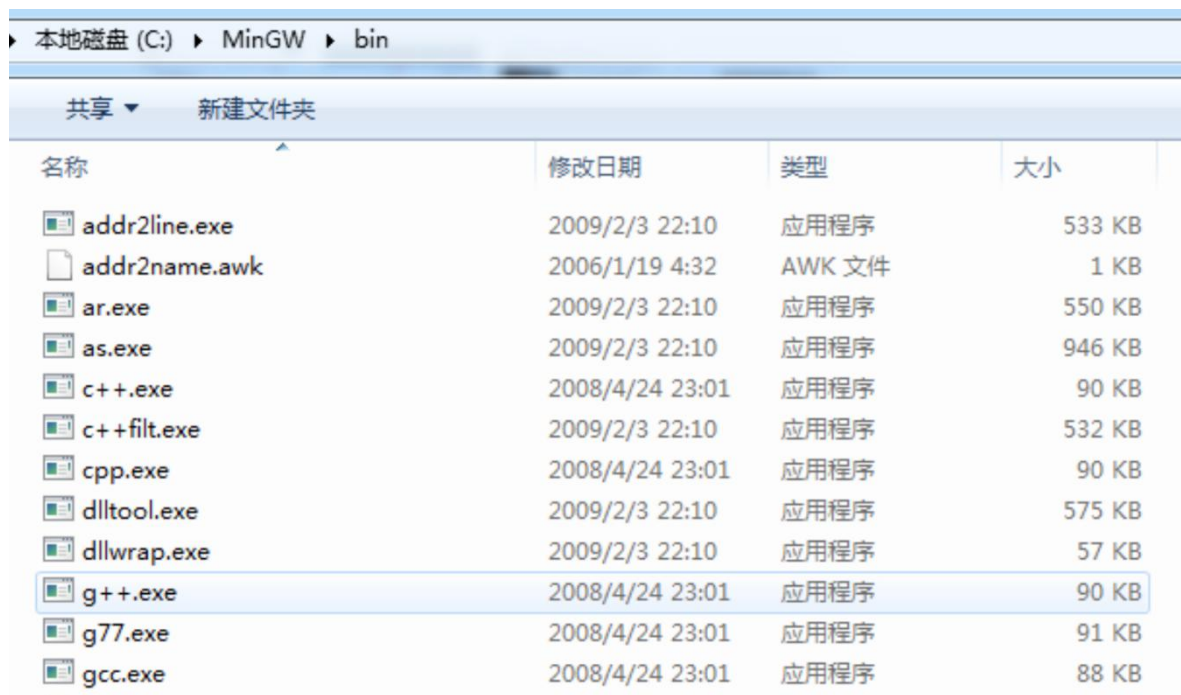
- 点击Apply确认安装，中途报错点击确定跳过即可















了解主流编译器

■也可单独到官网下载安装编译器

- 安装完成后可以看到一系列编译器程序



本地磁盘 (C:) > MinGW > bin			
共享 ▾ 新建文件夹			
名称	修改日期	类型	大小
 addr2line.exe	2009/2/3 22:10	应用程序	533 KB
 addr2name.awk	2006/1/19 4:32	AWK 文件	1 KB
 ar.exe	2009/2/3 22:10	应用程序	550 KB
 as.exe	2009/2/3 22:10	应用程序	946 KB
 c++.exe	2008/4/24 23:01	应用程序	90 KB
 c++filt.exe	2009/2/3 22:10	应用程序	532 KB
 cpp.exe	2008/4/24 23:01	应用程序	90 KB
 dlltool.exe	2009/2/3 22:10	应用程序	575 KB
 dllwrap.exe	2009/2/3 22:10	应用程序	57 KB
 g++.exe	2008/4/24 23:01	应用程序	90 KB
 g77.exe	2008/4/24 23:01	应用程序	91 KB
 gcc.exe	2008/4/24 23:01	应用程序	88 KB

编译器指令

- 各个平台上的编译器指令基本一致，在终端中输入相应命令提示符可以编译程序
- 编译**test.cpp**，生成名为**test**的可执行文件
 - `g++ test.cpp -o test`
- 额外的编译指令，例如优化指令
 - `g++ test.cpp -o test -O3`
- 更多指令将在后续课时中介绍

环境变量

- 在命令行中执行命令时，比如 `mv a.cpp b.cpp`，系统会从当前文件夹寻找 `a.cpp`
- 当我们在命令行使用 `g++` 指令编译程序，当前文件夹下并没有 `g++` 程序
- 系统除了在当前目录下面寻找程序外，还会到 `Path` 环境变量中的目录去找

环境变量

■ 环境变量一般是指在操作系统中用来指定操作系统运行环境的一些参数，如：临时文件夹位置和系统文件夹位置等。

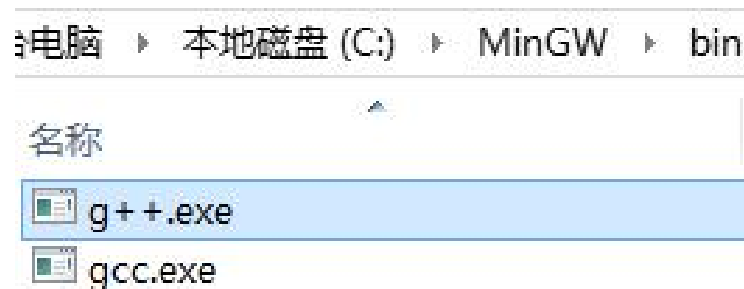
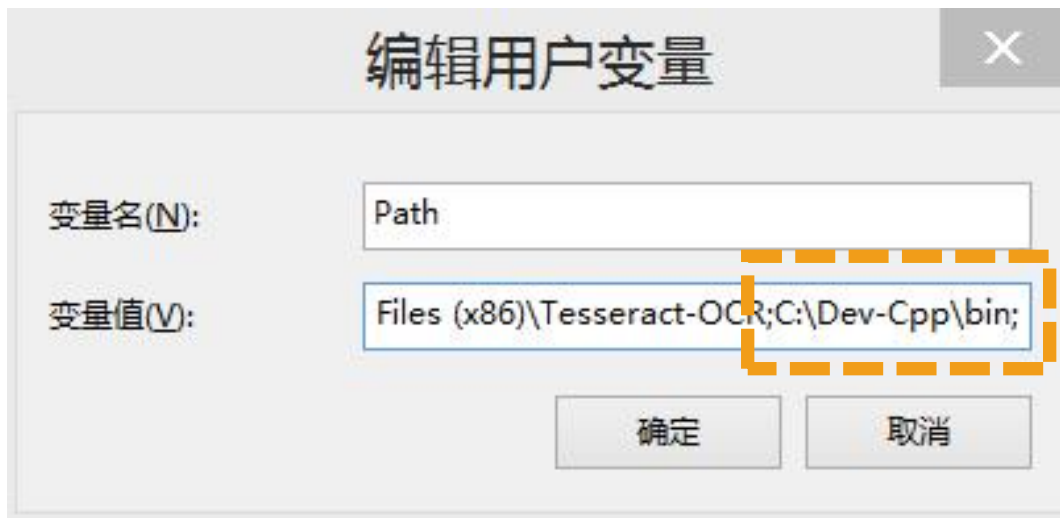
■ 当要求系统运行一个程序而没有告诉它程序所在的完整路径时，系统除了在当前目录下面寻找此程序外，还会到Path中指定的路径去找



环境变量

■Windows 下配置环境变量

- 这台电脑→右键→高级系统设置→高级→环境变量→用户环境变量Path→添加g++所在路径并以分号与其他路径隔开
- 在命令提示符中输入`g++ -v`测试是否配置成功



g++可能的所在路径:

C:\Dev-Cpp\bin

C:\Dev-Cpp\MinGW64\bin

C:\MinGW\bin

C:\TDM-GCC-64\bin

环境变量

■Linux/MAC下配置环境变量

- 配置环境变量编辑~/.bashrc文件
- 立即启用：source ~/.bashrc
- 且当每次打开新的shell时，~/.bashrc被执行

```
export PATH=/usr/local/cuda/bin:/home/yedeming/Python/bin:$PATH
export CUDA_ROOT=/usr/local/cuda
export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH_SCREEN=$LD_LIBRARY_PATH
export TMPDIR=/data/disk3/private/yedeming/tmp
```

- PATH的路径以冒号分隔，上图中新增两项之后以冒号分隔，\$PATH引用原来的环境变量
- 在命令行中使用echo \$PATH可查看当前PATH环境变量

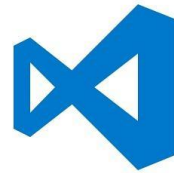
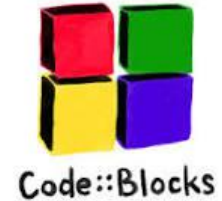
实操演示

- 命令行操作
- MinGW 安装
- 环境变量设置

了解主流IDE

■IDE

- DEV C++
- Code::Blocks
- CLion
- XCode



• Visual Studio Code

■Editor

- Sublime Text
- Notepad++
- Vim



我现在用的IDE是

A

DEV C++

B

Visual Studio

C

文本编辑器

D

其他

提交

Visual Studio Code

■ Visual Studio

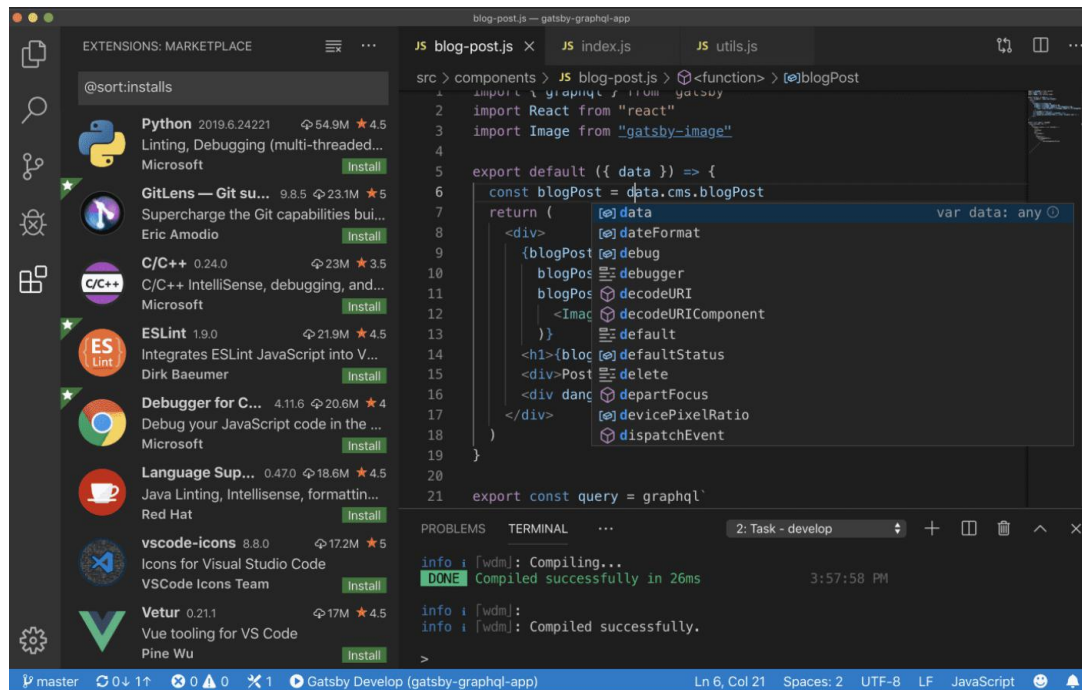


- VS是一个基本完整的开发工具集，它包括了整个软件生命周期所需要的大部分工具，支持Windows/Mac

■ Visual Studio Code



- 代码编译器，支持众多插件，跨平台

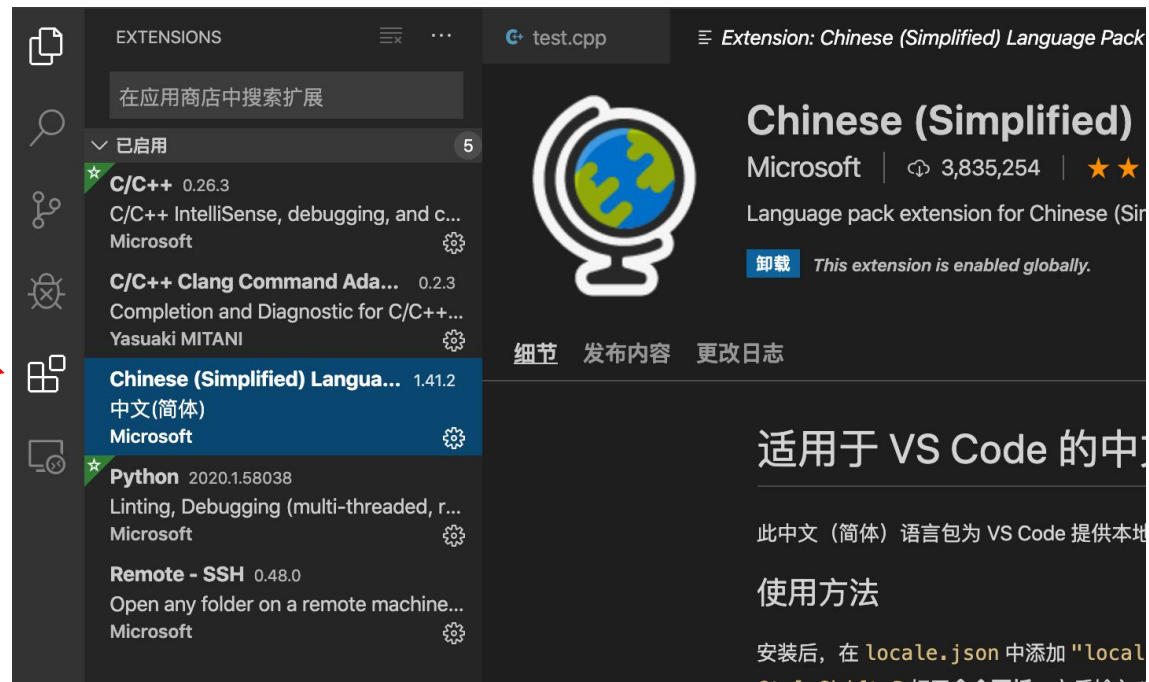


Visual Studio Code插件

■ 推荐安装插件增强用户体验

- Chinese (Simplified): 中文语言包
- C/C++: 提供Debug功能
- Code Runner: 支持右键运行程序
- Remote-SSH: ssh连接服务器，服务器与本地之间文件拷贝

插件管理



Visual Studio Code使用示例

■在以下链接中查看不同系统安装C++插件的教程

- <https://code.visualstudio.com/docs/languages/cpp>

■①配置g++ for Windows / clang++ for MAC

- Windows: 编辑g++环境变量
- Mac: 查看->命令面板->Shell Command: Install 'code' command in path

■②打开文件夹(路径尽量别用中文名), 新建文件, 编辑代码, 保存为test.cpp

- 若#include文件提示错误, 查看->命令窗口->Edit Configuration可编辑.vscode/**c_cpp_properties.json**, 在包含路径中增加 在命令行输入gcc -v -E -x c++ -得到的头文件目录

■③编译: 终端->运行生成任务

- 终端->配置任务可以打开.vscode/**tasks.json**配置运行相关信息

Visual Studio Code 配置

c_cpp_properties.json: 配置头文件目录、C++标准等

```
"configurations": [
  {
    "name": "Mac",
    "includePath": [
      "${workspaceFolder}/**",
      "/usr/local/include",
      "/Library/Developer/CommandLineTools/usr/include",
      "/Library/Developer/CommandLineTools/usr/libc++abi/include",
      "/Library/Developer/CommandLineTools/usr/libc++abi/include/c++/v1",
      "/Library/Developer/CommandLineTools/usr/include/c++/v1"
    ],
    "defines": [],
    "macFrameworkPath": [
      "/System/Library/Frameworks",
      "/Library/Frameworks"
    ],
    "compilerPath": "/usr/bin/clang",
    "cStandard": "c11",
    "cppStandard": "c++17",
  }
]
```

task.json: 运行g++/clang++程序
编译文件

```
"tasks": [
  {
    "type": "shell",
    "label": "clang++ build active file",
    "command": "/usr/bin/clang++",
    "args": [
      "-g",
      "${file}",
      "-o",
      "${fileDirname}/${fileBasenameNoExt}.o"
    ],
    "options": {
      "cwd": "/usr/bin"
    },
    "problemMatcher": [
      "$gcc"
    ],
    "group": {
      "kind": "build",
      "isDefault": true
    }
  }
]
```

Visual Studio Code 调试

■④运行：右键运行 / 调试->启动调试 (F5)

- 调试->打开配置可以打开.vscode/launch.json配置调试相关信息
- Catalina版本的macos需要安装CodeLLdb[插件](#)

Launch.json:

program: 调试程序位置

args: 运行参数

stopAtEntry: 在开始处设置断点

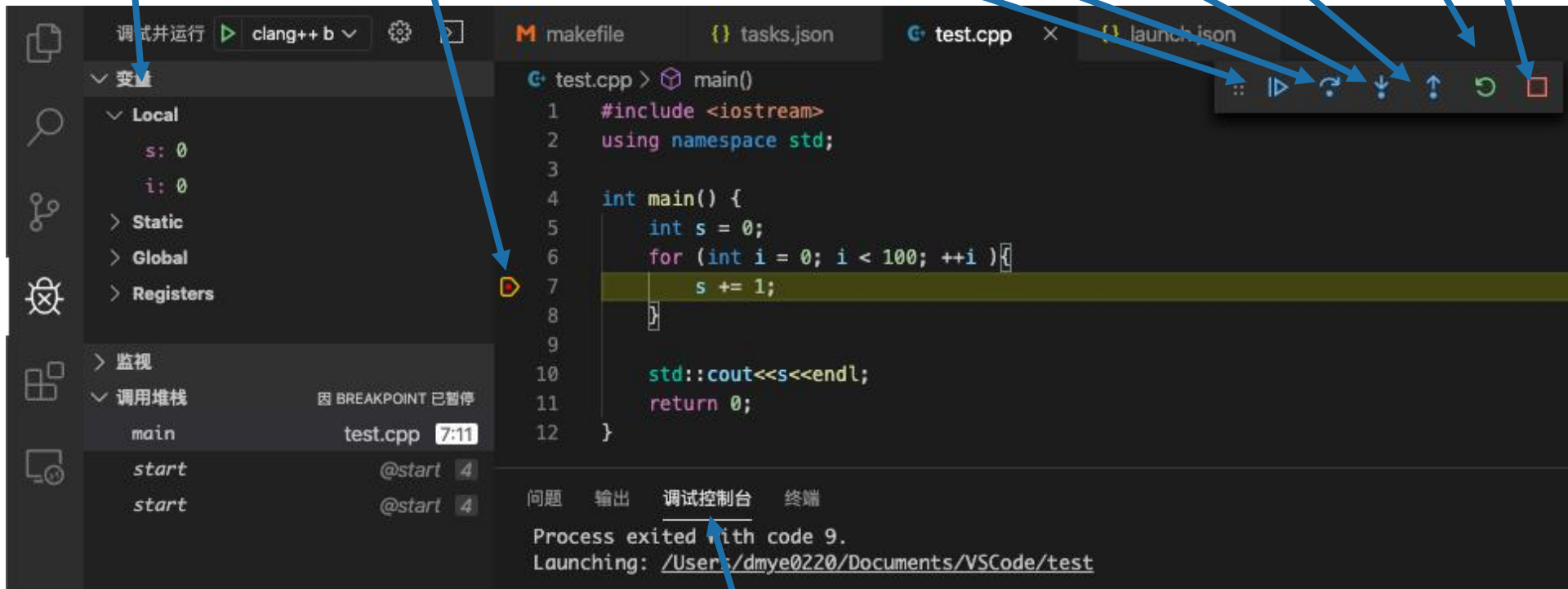
```
"configurations": [  
  {  
    "name": "clang++ build and debug active file",  
    "type": "cppdbg",  
    "request": "launch",  
    "program": "${fileDirname}/${fileBasenameNoExtension}.exe",  
    "args": ["arg1", "arg2"],  
    "stopAtEntry": true,  
    "cwd": "${workspaceFolder}",
```

Visual Studio Code 调试

断点 (可编辑布尔表达式为真才停)

变量值

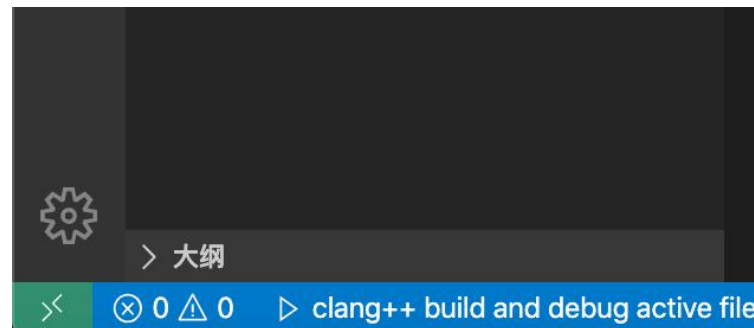
继续 单步跳过 单步调试 单步跳出 重启 停止



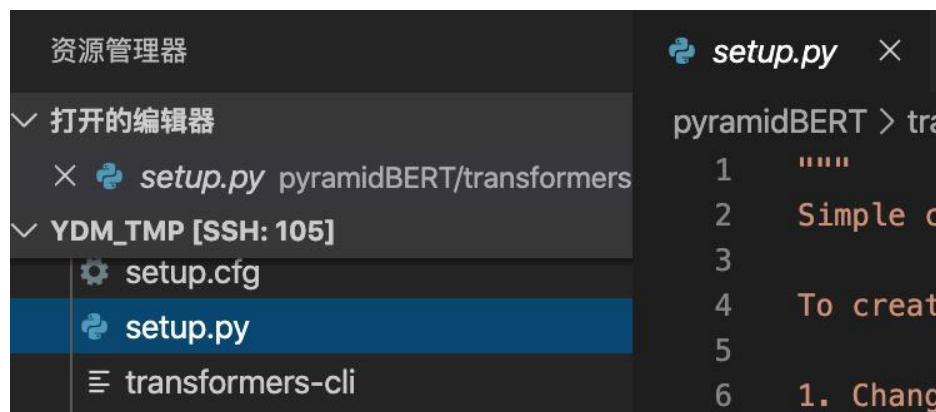
调试控制台：可输入gdb指令

Visual Studio Code 连接服务器

- 左下角可连接服务器



- 打开服务器程序进行编辑，保存后自动同步至服务器端



- 连接服务器端后在插件界面安装调试插件，修改launch.json，可实现在服务器端调试（可带GPU调试）



实操演示

- VSCode 插件安装
- VSCode 任务配置及调试

多听多做多练

■ 主线任务

- (1) 体验常见命令提示符命令
- (2) (Windows用户) 配置g++环境变量
- (3) 使用g++编译程序，运行程序
- (4) 安装VS Code
- (5) 在VS Code上编写程序并调试
- (6) (Linux用户) 学习apt-get命令
- (7) (OS X用户) 学习brew命令

■ 支线任务

- (1) 学习ssh命令
- (2) 开启本地ssh服务器并体验ssh/scp命令
 - 本地地址: 127.0.0.1
 - 同学地址: ifconfig/ipconfig查看ip后登陆
- (3) 学习在命令行中使用vim

多听多做多练

■ 课后练习

- 熟悉编译命令中各个参数的含义及作用，并实际尝试不同的编译选项之间的差异
- 利用VSCode调试功能（断点），观察一个循环中循环变量的值的变化过程

ssh远程登录和操作（自学）

- SSH(Secure Shell) 是建立在应用层基础上的安全协议，可以用其远程登录服务器/其他电脑
- `ssh remote_username@remote_address`
 - user是在远程机器上的用户名
 - remote_address 是远程机器的地址，可以是IP/域名/别名
- SCP(Secure Copy) 是 linux 系统下基于SSH登陆进行安全的远程文件拷贝命令。
- 可添加自己的公钥到远程机器
~/.ssh/authorized_keys 中实现免密码登录
- 修改~/.ssh/config可以给远程机器起简称，可以设置使用简称完成服务器跳转登录

ssh远程登录和操作（自学）

- **SCP(Secure Copy)** 是 linux 系统下基于SSH登陆进行安全的远程文件拷贝命令。
- **从服务器端复制到本地**
 - `scp remote_username@remote_address:remote_file local_folder`
- **从本地复制到服务器端**
 - `scp local_file remote_username@remote_address:remote_folder`
- **SSHFS(SSH Filesystem)** 通过ssh连接挂载远程服务器上的目录到本地
 - `sshfs remote_username@remote_address:remote_folder local_folder`

结束