



C++ Programming

C++初步知识 Preliminaries of C++

2025年2月17日

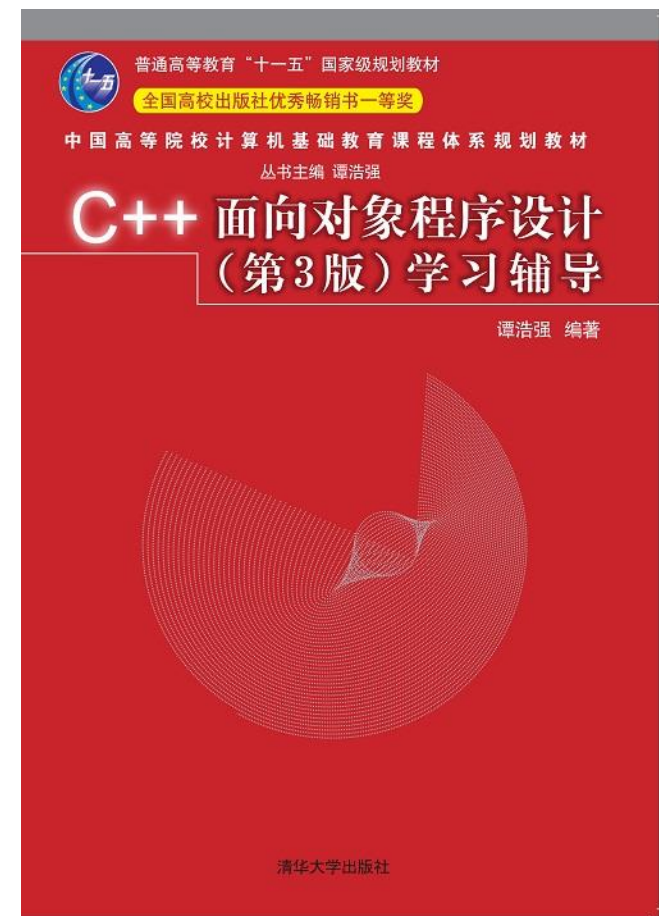
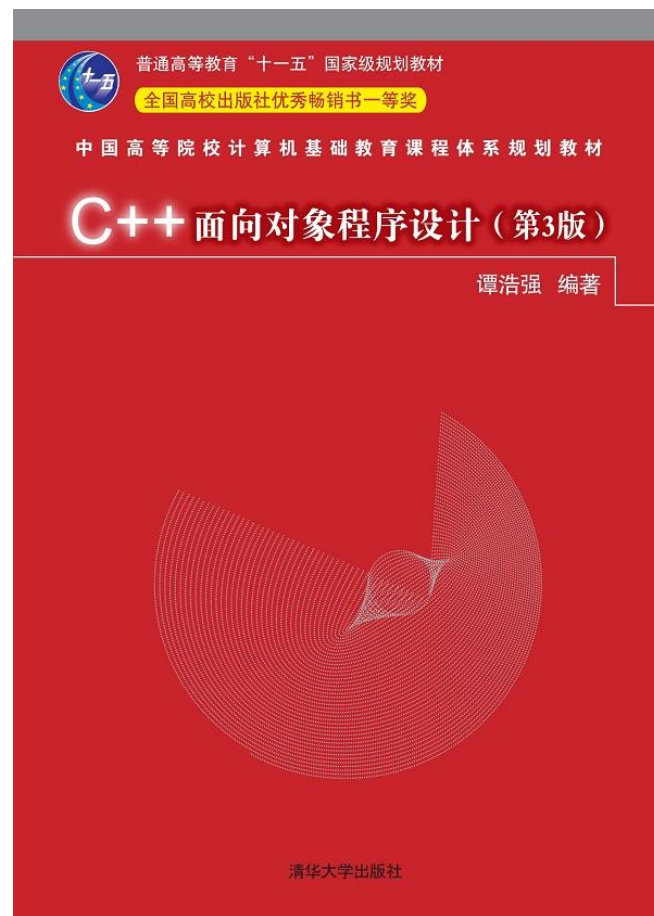
学而不厌 诲人不倦

➤ 教学内容

课程代码及名称

301039 C++程序设计

- 第1章 C++的初步知识
- 第2章 类和对象的特性
- 第3章 如何使用类和对象
- 第4章 对运算符重载
- 第5章 继承与派生
- 第6章 多态性与虚函数
- 第7章 输入输出流
- 第8章 C++工具



主要参考教材



教学内容与考核

➤ 课程目标

1. 了解课程和专业相关前沿

2. 熟练掌握C++编程基本技能：编码、调试

- Learn how to write **powerful** and **elegant** code.
- Write **actual** C++, no libraries
- Understand the design decisions that lead to “**good**” code

3. 掌握利用C++解决实际问题的初步能力

➤ 考核方式

1. 课堂教学30%：出勤、准时、实验作业、主动提出问题等

2. 课程大作业 40%： 预计第4周布置题目，第10周前完成

3. Spotlights 20%：第10-12周完成 每位同学PPT讲解

➤ 先修/后续课程

年级-专业-姓名

C 程序设计→C++面向对象程序设计→数据结构/算法设计



```

graph LR
    A[计算机] --- B[主机]
    A --- C[I/O设备]
    B --- D[运算器]
    B --- E[控制器]
    B --- F[存储器]
    D --- G[中央处理器CPU]
    E --- G
  
```

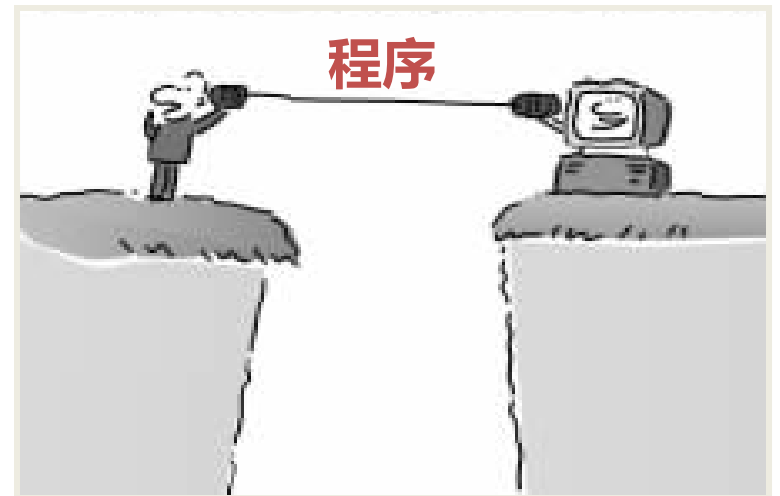
计算机

主机：

- 运算器
- 控制器
- 存储器

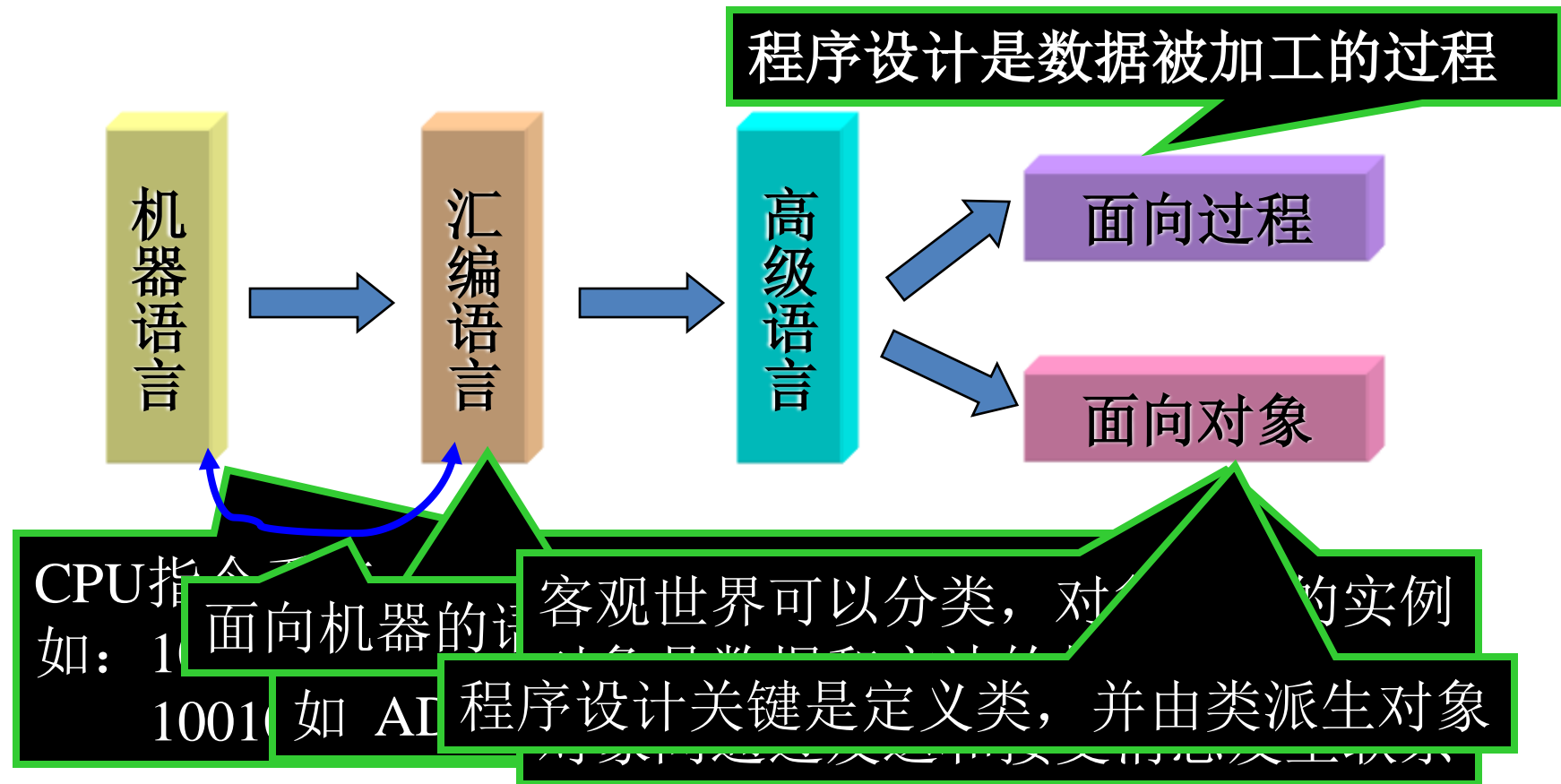
中央处理器CPU

I/O设备：键盘、显示器等

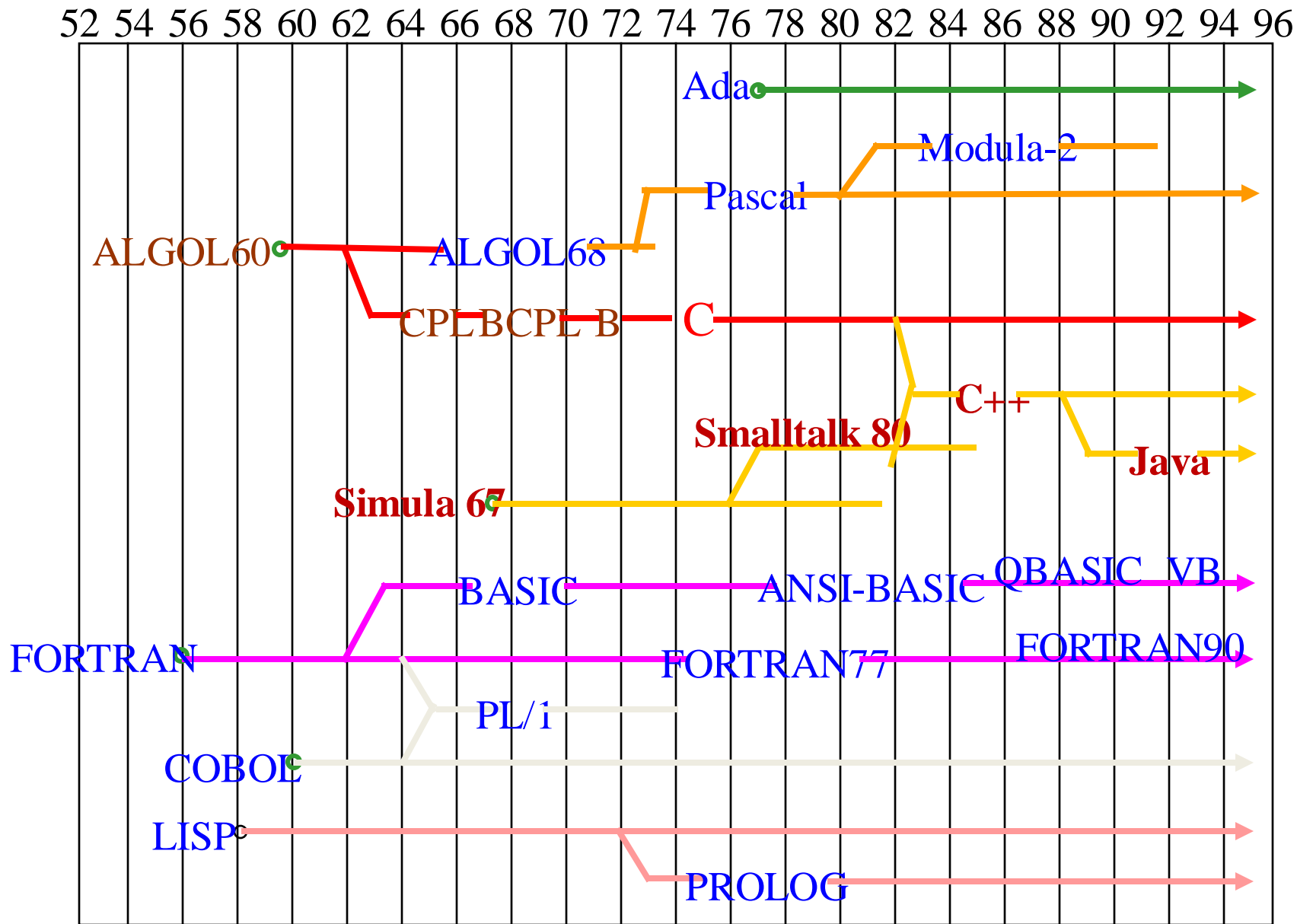


人要和计算机有效地交流，必须通过程序

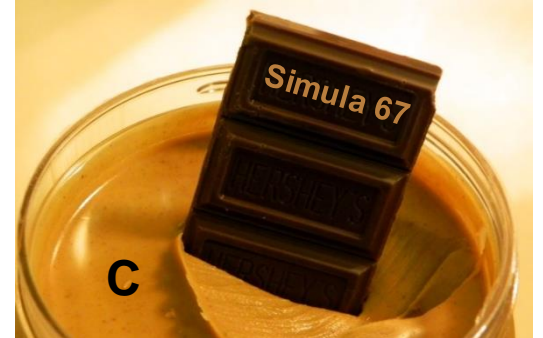
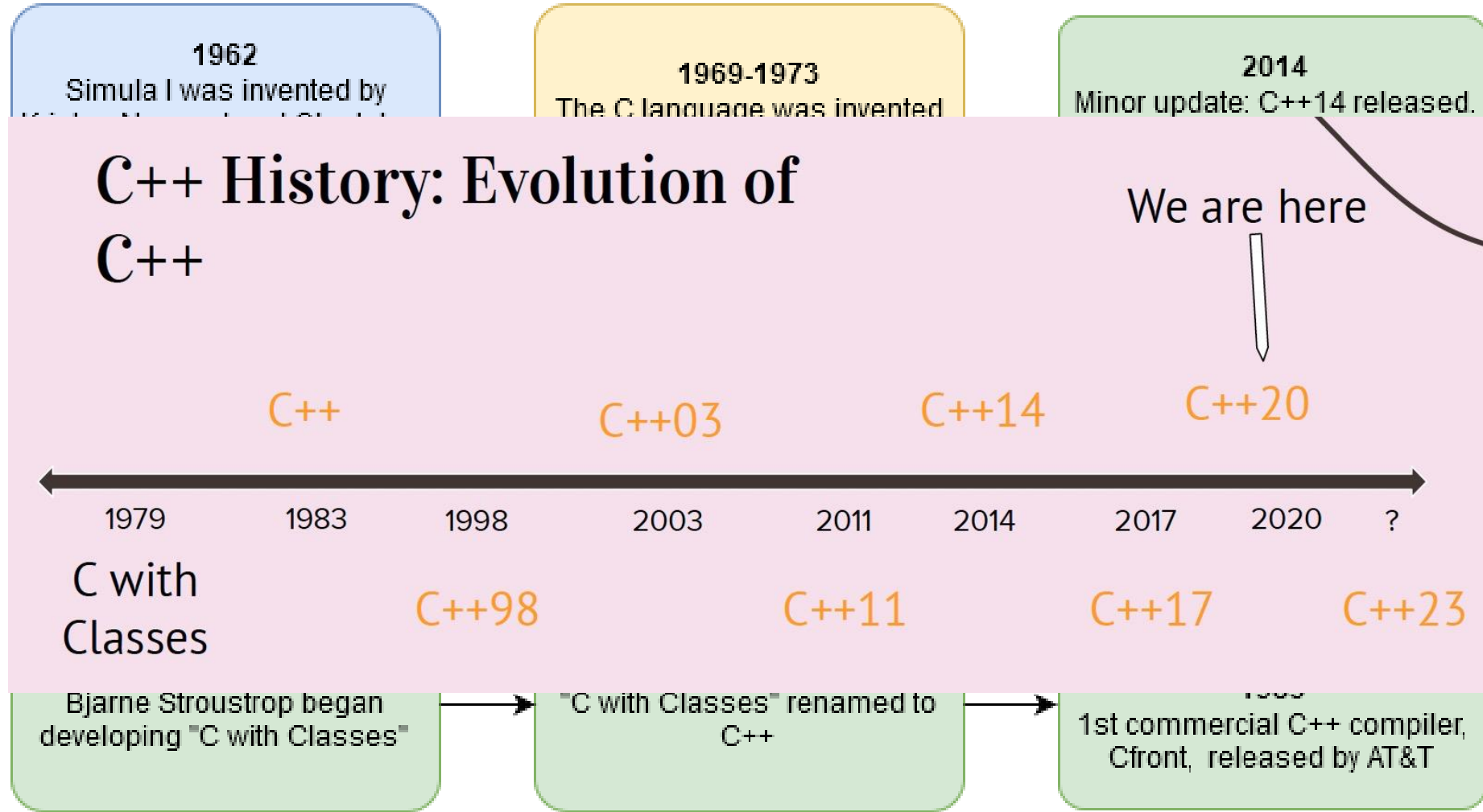
程序设计语言



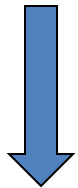
程序设计语言



Very brief history of C++



C



C++

For details more check out [A History of C++: 1979–1991](#)

Why C++

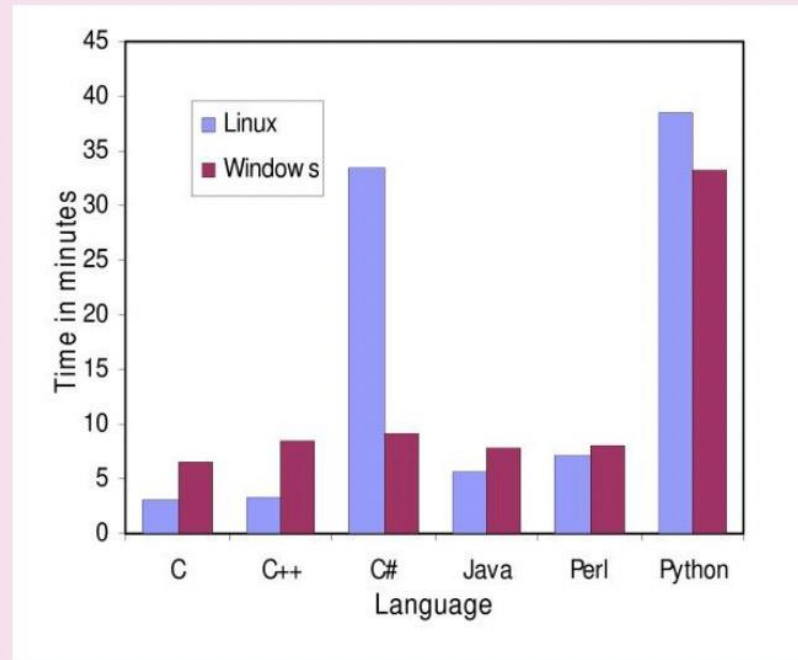
Many Cool Things Use/Were Made with C++

amazon

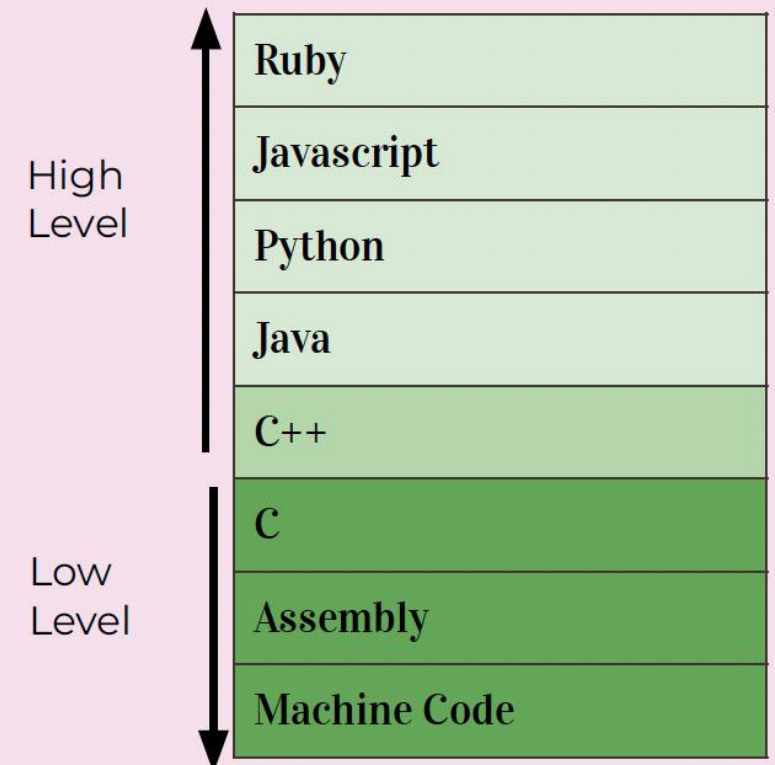


Why C++?

FAST

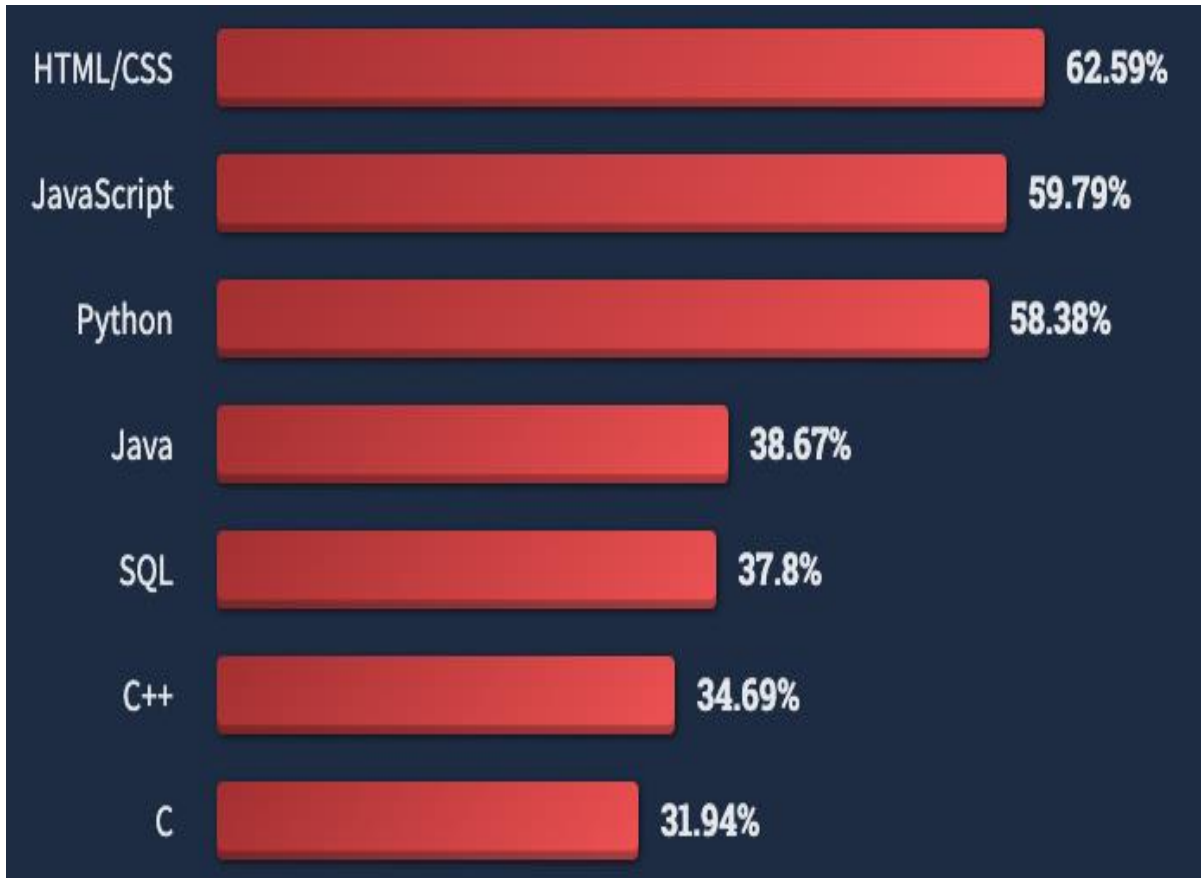


Lower-level control

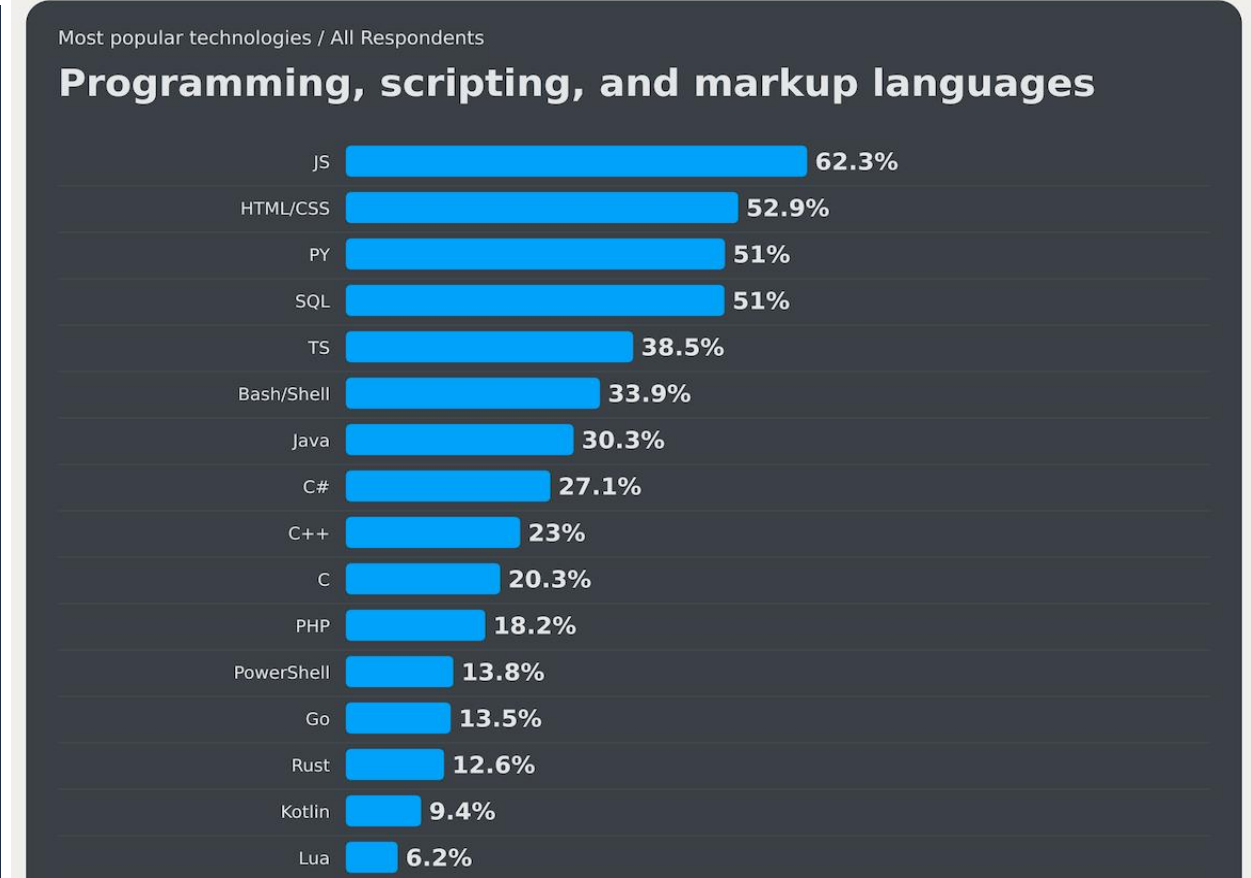


Why C++

The most popular programming languages for novice programmers



<https://survey.stackoverflow.co/2022/#most-popular-technologies-language-learn>



<https://survey.stackoverflow.co/2024/technology#1-databases>

<https://www.adesso.de/en/news/blog/stack-overflow-developer-survey-2022-part-2-2.jsp>



Why C++

Tencent 腾讯校招

首页 青云计划 岗位投递 招聘动态 了解腾讯 求职攻略 登录

UNITREE

你是腾讯产品「背后」的人

技术 应届生

软件开发

职位描述

岗位要求

deepseek

社会招聘

职位列表

首页 / 职位列表 / 职位详情

深度学习研发工程师（AGI 北京/杭州）

全职 | 开发类 | AGI | 浙江-杭州市

发布于 2025-02-11

申请职位

职位描述

岗位职责描述：
1. 既懂算法又懂系统
2. 既能调精度也能调性能
3. 既考虑训练也考虑推理部署

任职要求：
1. 了解机器学习（深度学习）的各类模型，具备较强的工程能力。
2. 编程能力出色，熟练掌握Python和C++，掌握Pytorch/Tensorflow。
3. 熟练掌握深度学习各类模型架构的使用和设计。
4. 在领域内知名比赛取得优异成绩者优先。

职位信息

职位名称	薪资范围	职位性质	职能类型
深度学习研发工程师（AGI 北京/杭州）		全职	开发类
所属部门	工作地点	发布日期	
AGI	浙江-杭州市	2025-02-11	

UNITREE

AI算法工程师

杭州市 | 技术类 | 研发

岗位职责

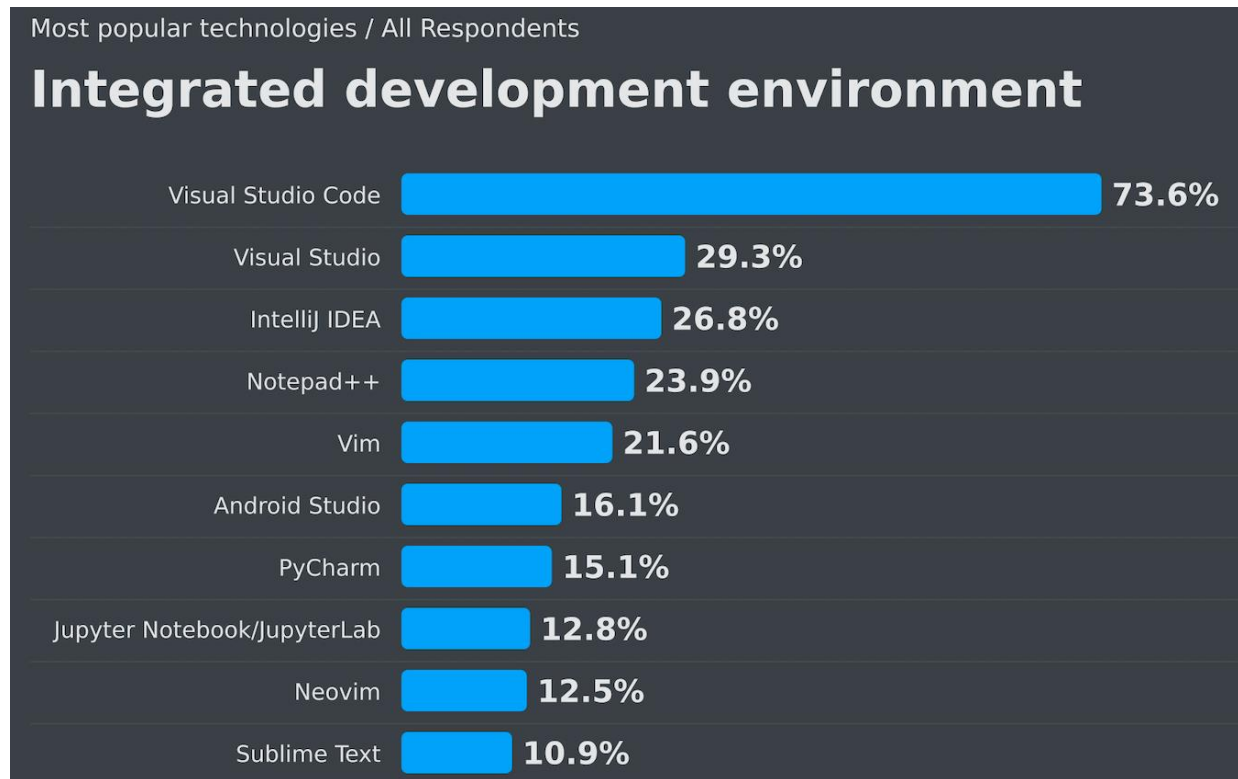
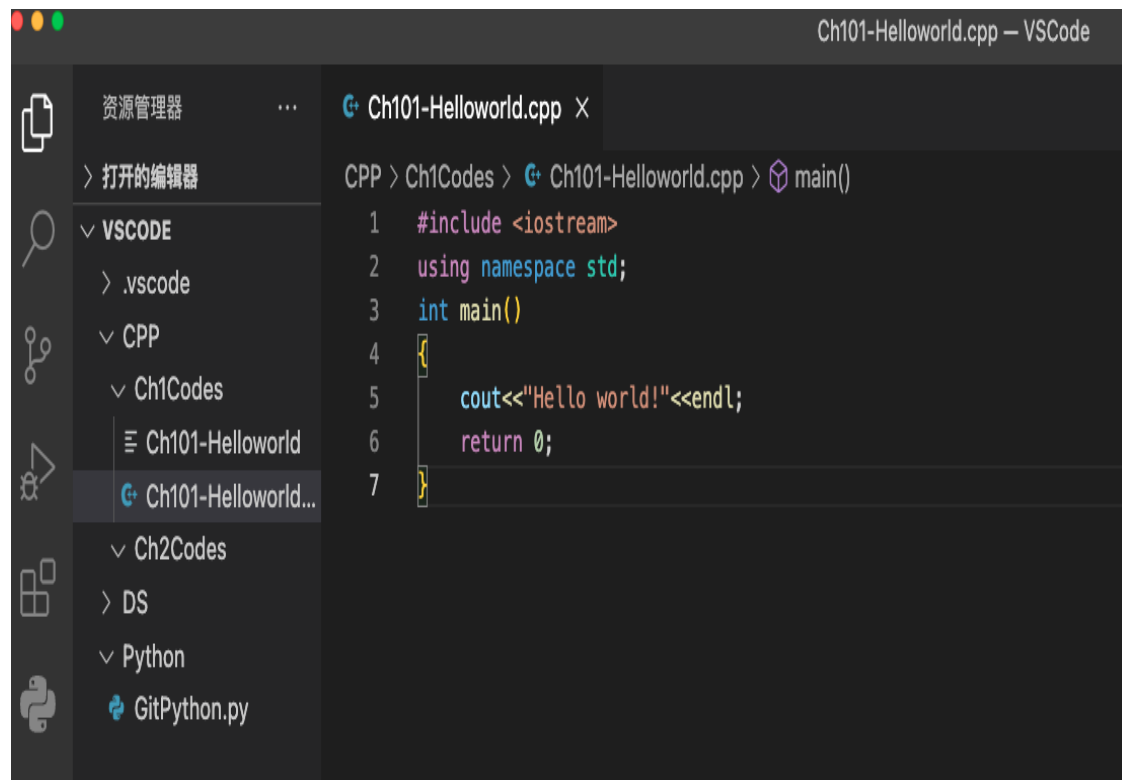
1. 相机硬件的选型
2. 设计、开发和优化
3. 开发和推进基于
4. 调研和推进3D
5. 跟进基于深度学
6. 负责视觉算法开

任职要求

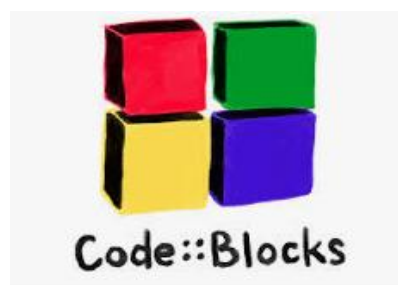
1. 熟悉相机senso
2. 熟悉主流的神经
3. 熟悉至少一种靠
4. 了解主流的3D
5. 了解主流的端到
6. 熟练掌握pytorc
7. 熟悉C++/C和Py
8. 具有计算机科学

https://www.unitree.com/cn/position

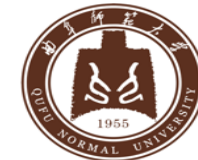
VSCode / Code::Blocks / Microsoft Visual Studio 6.0/Latest Versions



<https://survey.stackoverflow.co/2022/#most-popular-technologies-new-collab-tools>



关于开放课程资源



CS 106L

Standard C++ Programming
Stanford University, Winter 2023

About CS106L

📖 **CS 106L** is a companion class to CS106B/CS106X that explores the modern C++ language in depth. We'll cover some of the most exciting features of C++, including modern patterns that give it beauty and power.

🌱 Anyone who is taking or has taken CS 106B/X (or equivalent) is welcome to enroll. In other words, we welcome anyone that has learned or is learning programming fundamentals like functions and objects/classes.

📅 **CS 106L** is a class for **1 unit**. Students will complete **2 very short** assignments. There are **no exams or papers**. All grades are S/NC. Class will finish in week 9 to give you time for finals.

🏠 **CS 106L** is built for you! Even if you're not taking the class, you're welcome to come to our in-person office hours. Haven: Tuesdays 4:30 - 5:30pm in 260-113. Sarah: Wednesdays 3:15 - 4:15pm in 120-314. If these times don't work for you, we'll also have **virtual** office hours, times by appointment!

Course Information

👤 Haven Whitney

👤 Sarah McCarthy

✉️ cs106l-win2223-staff@lists.stanford.edu

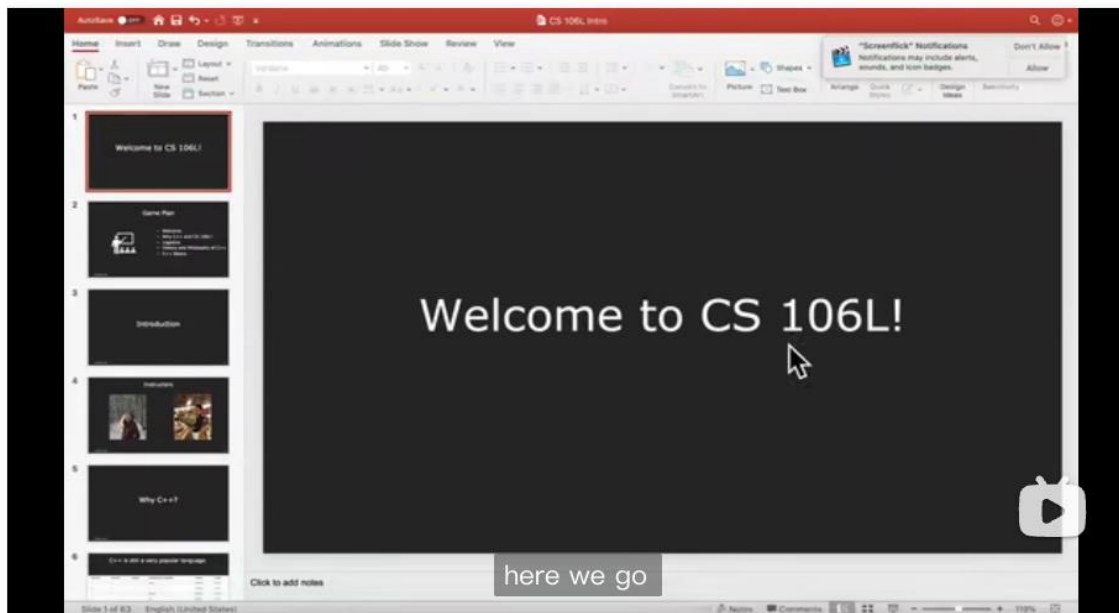
🕒 Tuesday, Thursday; 3:00-4:20pm; 260-113

Week	Tuesday	Thursday
1	JANUARY 10 1. Welcome! Slides Policies	JANUARY 12 2. Types and Structs Slides Code
2	JANUARY 17 3. Initialization and References Slides Code	JANUARY 19 4. Streams Slides Code
3	JANUARY 24 5. Containers Slides	JANUARY 26 6. Iterators and Pointers Slides Code

<http://web.stanford.edu/class/cs106l/>

【中英双语】CS106L: Standard C++ Programming, Special Edition

👁 1.3万 🗨 11 ⌚ 2022-09-17 20:19:58



https://www.bilibili.com/video/BV1K8411b7AU/?spm_id_from=333.337.search-card.all.click

```
if (start == str.length()) {  
    return result;  
}  
// find all subsets of str starting at current index  
// subsets of str, starting at start, are formed by adding in the current character  
//  
Vector<string> result;  
for (string s: subsetsOf(str, start + 1)) {  
    result += str[start] + s;  
    result += s;  
}  
return result;  
}
```

CS106B

Programming Abstractions in C++

To visit the website for fall quarter's CS106B offering, [click here](#).

Midterm 2 Released

March 12, 2021

Our second [midterm exam](#) goes out today. It's due in 47 hours (Sunday, March 14th at 12:30PM Pacific time).

You can do this. Best of luck on the exam!

Assignment 9 Released

March 12, 2021

The last programming assignment of the quarter, [Huffman Coding](#), goes out today. It's due on Friday, March 19th at the start of class (11:30AM Pacific).

Handouts

[Huffman Coding](#)
[Second Midterm Logistics](#)
[Midterm Logistics](#)
[Container Syntax Reference](#)
[Python-to-C++ Guide](#)
[Assignment Submission Checklist](#)
[Debugging Your Code](#)
[Honor Code](#)
[Course Placement](#)
[Course Calendar](#)
[Course Information](#)

Resources

[Stanford C++ Library Documentation](#)
[C++ Standard Library Documentation](#)
[Setting Up Qt Creator](#)
[Blank Stanford C++ Project](#)
[Blank SimpleTest Project](#)
[Assignment Submitter](#)
[Working in Pairs](#)
[Style Guide](#)
[Git Help Session](#)
[Git Help Session Slides](#)

【斯坦福大学】CS106B C++中的抽象编程 · 2018年冬（完结·中英字幕·机翻）

4.5万 92 2020-01-25 19:33:55



The screenshot shows the Stanford CS106B website for the Winter 2018 offering. The page features a navigation bar with links to Home, Handouts, Textbook, Pair Programming, Lectures, Homework, Sections, Exams, Staff/SLs, LaIR Hours, Piazza Forum, Videos, Qt Creator, Stanford C++ Lib, CppRef / C++.com, Style Guide, CodeStepByStep, FAQ, Links, and CS Major. Below the navigation bar, the instructor is listed as Marty Stepp, and the TAs are Ashley Taylor and Amy Xu. The page also includes an announcements section with a welcome message for the 2018 Winter offering.

•课程教材: <https://web.stanford.edu/class/cs106x/res/reader/CS106BX-Reader.pdf>

•课程视频: <https://www.bilibili.com/video/BV1G7411k7jG>

cppreference.com

Create account

Search

Page

Discussion

Standard revision: Diff

View

View source

History

C++ reference

C++11, C++14, C++17, C++20, C++23, C++26 | Compiler support C++11, C++14, C++17, C++20, C++23, C++26

Language

- Keywords – Preprocessor
- ASCII chart
- Basic concepts
 - Comments
 - Names (lookup)
 - Types (fundamental types)
 - The main function
- Expressions
 - Value categories
 - Evaluation order
 - Operators (precedence)
 - Conversions – Literals
- Statements
 - if – switch
 - for – range-for (C++11)
 - while – do-while
- Declarations – Initialization
- Functions – Overloading
- Classes (unions)
- Templates – Exceptions
- Freestanding implementations

Standard library (headers)

Named requirements

Feature test macros (C++20)

- Language – Standard library

Language support library

- Program utilities
 - Signals – Non-local jumps
- Basic memory management
- Variadic functions
- source_location (C++20)
- Coroutine support (C++20)
- Comparison utilities (C++20)
- Type support – type_info
- numeric_limits – exception
- initializer_list (C++11)

Concepts library (C++20)

Diagnostics library

- Assertions – System error (C++11)
- Exception types – Error numbers
- basic_stacktrace (C++23)
- Debugging support (C++26)

Memory management library

- Allocators – Smart pointers
- Memory resources (C++17)

Metaprogramming library (C++11)

- Type traits – ratio
- integer_sequence (C++14)

General utilities library

- Function objects – hash (C++11)
- Swap – Type operations (C++11)
- Integer comparison (C++20)
- pair – tuple (C++11)
- optional (C++17)
- expected (C++23)
- variant (C++17) – any (C++17)
- bitset – Bit manipulation (C++20)

Containers library

- vector – deque – array (C++11)
- list – forward_list (C++11)
- map – multimap – set – multiset
- unordered_map (C++11)
- unordered_multimap (C++11)
- unordered_set (C++11)
- unordered_multiset (C++11)
- Container adaptors
- span (C++20) – mdspan (C++23)

Iterators library

Ranges library (C++20)

- Range factories – Range adaptors
- generator (C++23)

Algorithms library

- Numeric algorithms
- Execution policies (C++17)
- Constrained algorithms (C++20)

Strings library

- basic_string – char_traits
- basic_string_view (C++17)

Text processing library

- Primitive numeric conversions (C++17)
- Formatting (C++20)
- locale – Character classification
- text_encoding (C++26)
- Regular expressions (C++11)
- basic_regex – Algorithms
- Default regular expression grammar
- Null-terminated sequence utilities:
 - byte – multibyte – wide

Numerics library

- Common math functions
- Mathematical special functions (C++17)
- Mathematical constants (C++20)
- Basic linear algebra algorithms (C++26)
- Data-parallel types (SIMD) (C++26)
- Pseudo-random number generation
- Floating-point environment (C++11)
- complex – valarray

Date and time library

- Calendar (C++20) – Time zone (C++20)

Input/output library

- Print functions (C++23)
- Stream-based I/O – I/O manipulators
- basic_istream – basic_ostream
- Synchronized output (C++20)
- File systems (C++17)

Concurrency support library (C++11)

- thread – jthread (C++20)
- atomic – atomic_flag
- atomic_ref (C++20) – memory_order
- Mutual exclusion – Semaphores (C++20)
- Condition variables – Futures
- Latch (C++20) – barrier (C++20)
- Safe Reclamation (C++26)

Execution support library (C++26)

Technical specifications

- Standard library extensions** (library fundamentals TS)
 - resource_adaptor – invocation_type
- Standard library extensions v2** (library fundamentals TS v2)
 - propagate_const – ostream_joiner – randint
 - observer_ptr – Detection idiom
- Standard library extensions v3** (library fundamentals TS v3)
 - scope_exit – scope_fail – scope_success – unique_resource

External Links – Non-ANSI/ISO Libraries – Index – std Symbol Index

cppreference.com

Create account

Search

Page

Discussion

View

Edit

History

C++

C++ language

Classes

Classes

A class is a user-defined type.

A class type is defined by class-specifier, which appears in *decl-specifier-seq* of the [declaration](#) syntax. See [class declaration](#) for the syntax of the class specifier.

A class can have the following kinds of members:

- 1) data members:
 - a) [non-static data members](#), including [bit-fields](#).
 - b) [static data members](#)
- 2) member functions:
 - a) [non-static member functions](#)
 - b) [static member functions](#)
- 3) nested types:
 - a) [nested classes](#) and [enumerations](#) defined within the class definition
 - b) aliases of existing types, defined with [typedef](#) or [type alias \(since C++11\)](#) declarations
 - c) the name of the class within its own definition acts as a public member type alias of itself for the purpose of [lookup](#) (except when used to name a [constructor](#)): this is known as [injected-class-name](#)
- 4) [enumerators](#) from all unscoped enumerations defined within the class, or introduced by [using-declarations](#) or [using-enum-declarations \(since C++20\)](#)
- 5) [member templates](#) (variable templates, [\(since C++14\)](#) class templates or function templates) may appear in the body of any non-local class/struct/union.

All members are defined at once in the class definition, they cannot be added to an already-defined class (unlike the members of namespaces)

A member of a class T cannot use T as its name if the member is

- a static data member,
- a member function,
- a member type,
- a member template,
- an enumerator of an enumeration (unless the enumeration is scoped) [\(since C++11\)](#), or
- a member of a member anonymous union.

<https://en.cppreference.com/w/>

- ➡ **1.1 从C到C++**
- ➡ **1.2 最简单的C++程序**
- ➡ **1.3 C++对C的扩充**
- ➡ **1.4 C++程序的编写和实现**
- ➡ **1.5 关于C++上机实践**

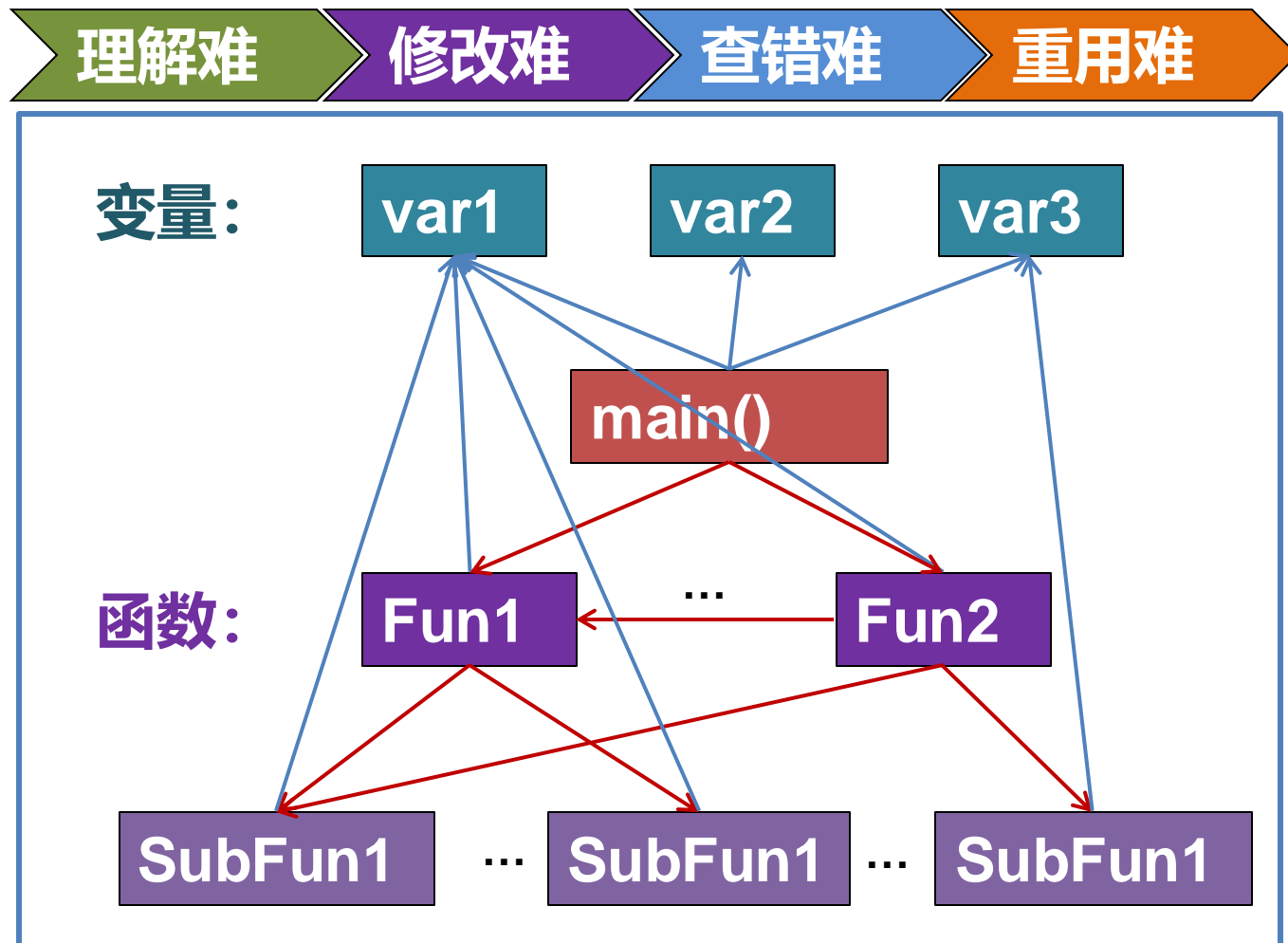
➤ 面向过程编程

按部就班、自顶向下的编程

模块化编程：函数+变量=程序

常量、变量、指针、结构体

顺序、选择/分支、循环结构

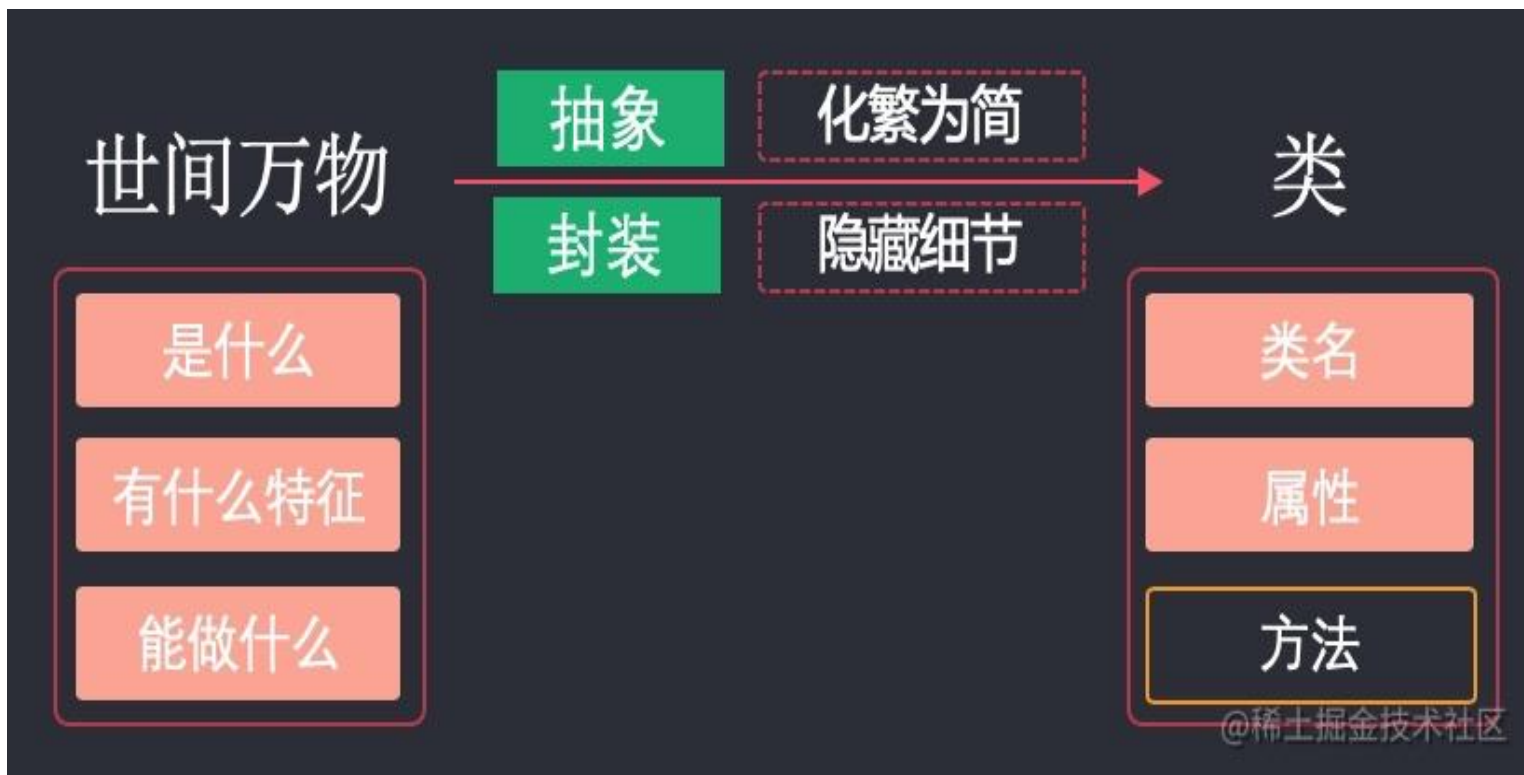


1.1 从 C 到 C++

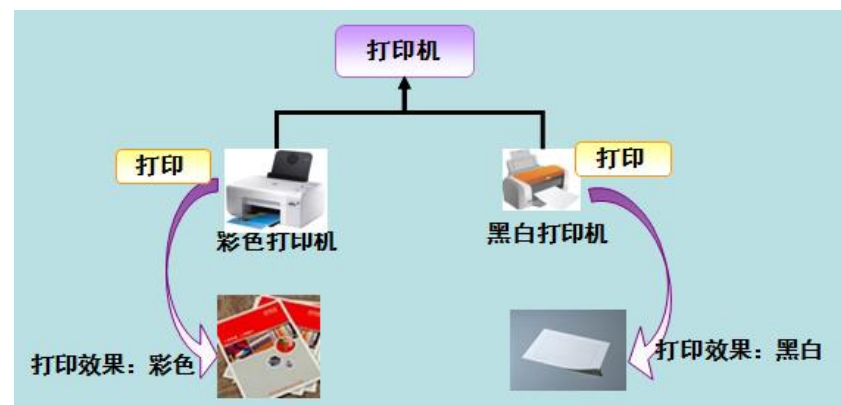
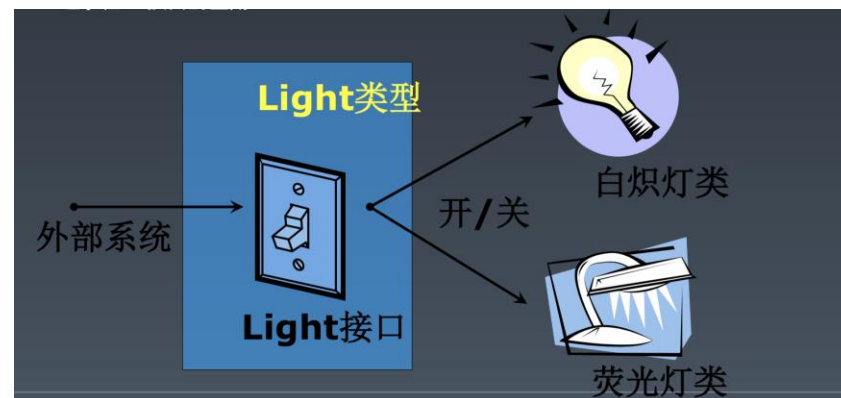
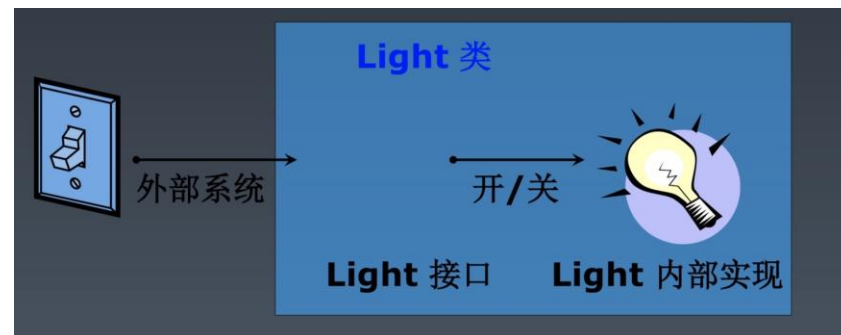
OOP: Object-Oriented Programming

➤ 面向对象编程

物以类聚，人以群分 万物皆对象，对象皆有类



<https://juejin.cn/post/6844904041307176974>

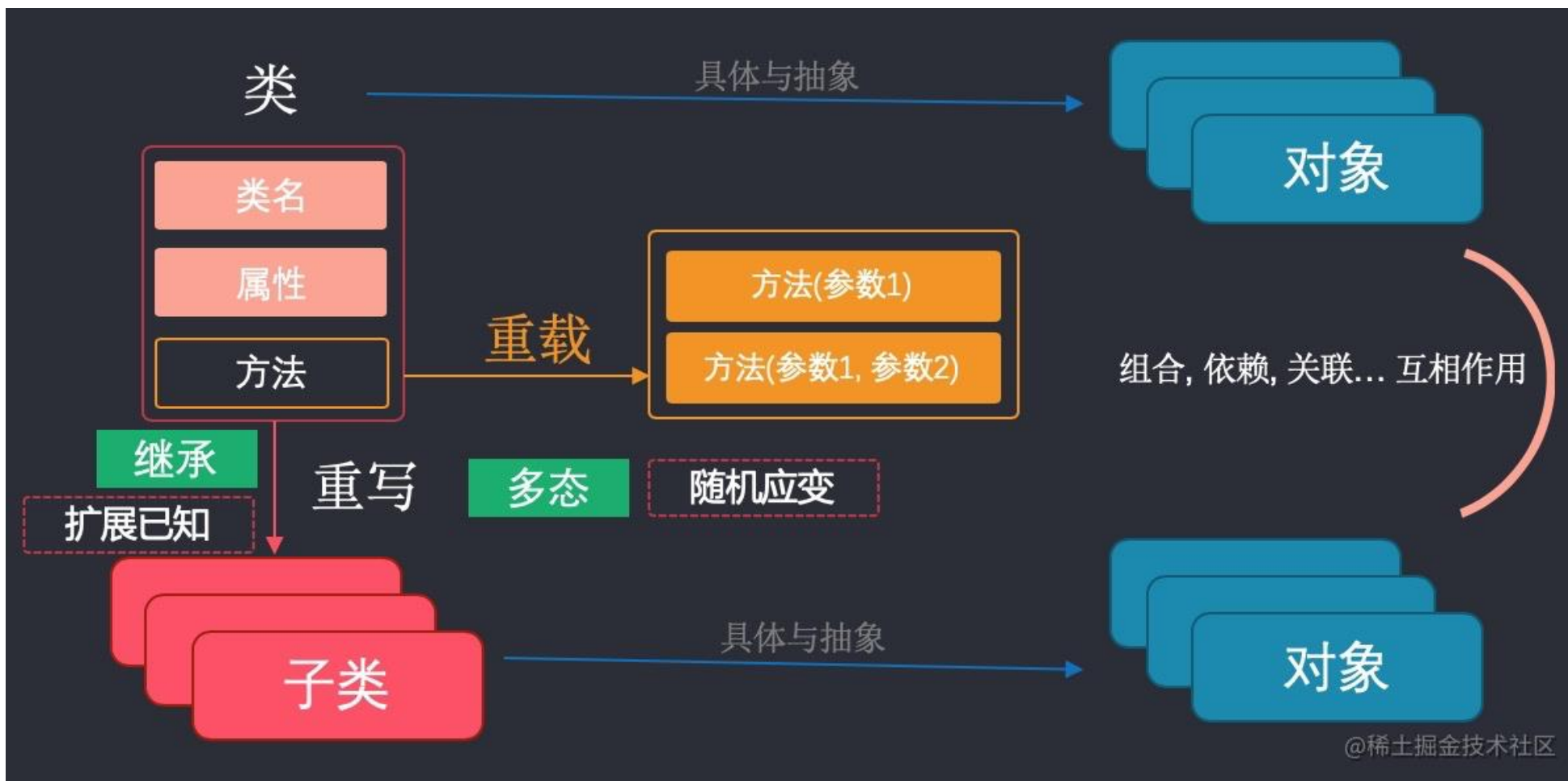


1.1 从 C 到 C++

OOP: Object-Oriented Programming

➤ 面向对象编程

关系驱动世界



➤ 面向对象编程

泛化 (Generalization)

继承关系, 一般与特殊 **is-a**
实线空心三角, 指向父类

实现 (Realization)

类与接口的实现关系 **Implement**
虚线空心三角, 指向接口

组合 (Composition)

整体与部分的关系 **Contains-a**
部分不能离开整体存在
实线实心菱形箭头, 菱形指整体

依赖 (Dependency)

使用关系 **Use-it**
局部变量, 参数, 静态方法调用
虚线箭头, 指向被使用者

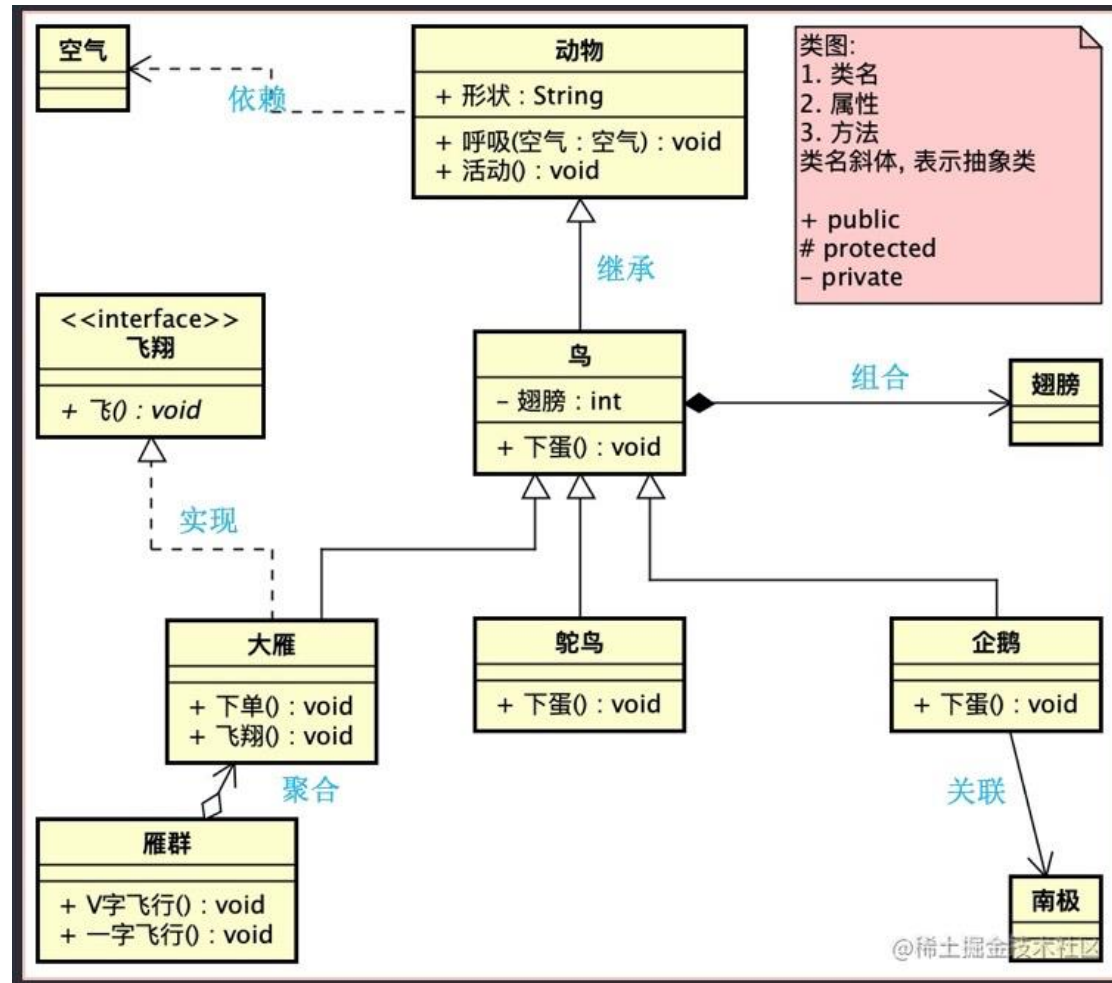
关联 (Association)

拥有关系(成员变量)
实线箭头, 指向被拥有者

聚合 (Aggregation)

整体与部分的关系 **has-a**
部分可以离开整体存在
实线空心菱形箭头, 菱形指整体

@稀土掘金技术社区



程序=算法+数据结构

软件=程序+文档

➤ 问题分析与描述

Problem Analysis and Specification

➤ 设计

Design

➤ 编码

Implementation (Coding)

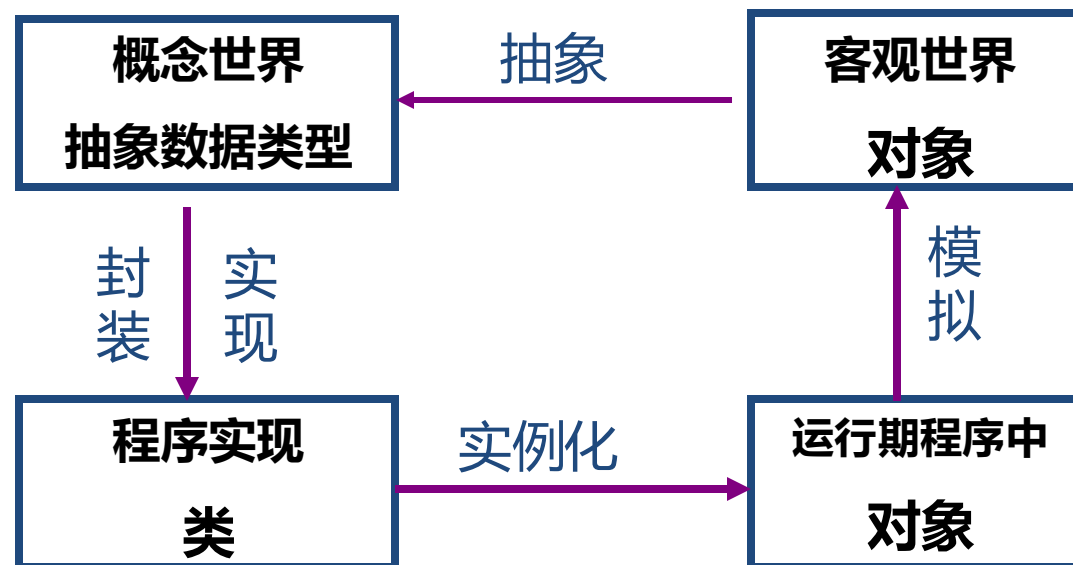
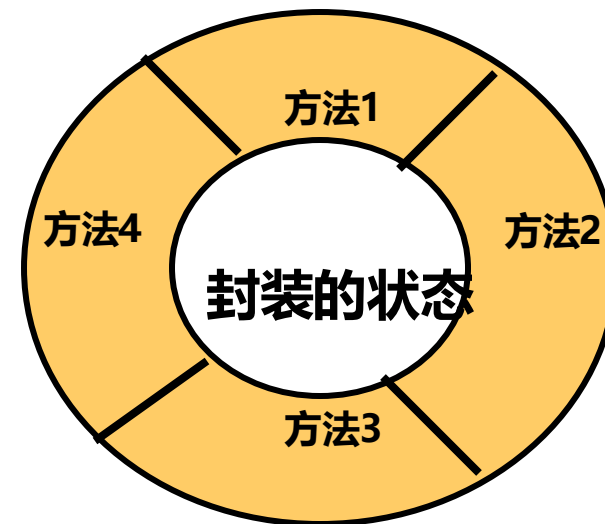
➤ 软件测试、调试

Testing, Execution and Debugging

➤ 软件维护

Maintenance

以对象为中心的设计



- ➡ 1.1 从C到C++
- ➡ **1.2 最简单的C++程序**
- ➡ 1.3 C++对C的扩充
- ➡ 1.4 C++程序的编写和实现
- ➡ 1.5 关于C++上机实践

2023年3月1日

1.2 最简单的C++程序?

回顾：最简单的C程序

```
/* This is the first C program */  
#include <stdio.h>  
  
int main ( )  
{  
    printf ("Hello World!\n");  
    return 0;  
}
```

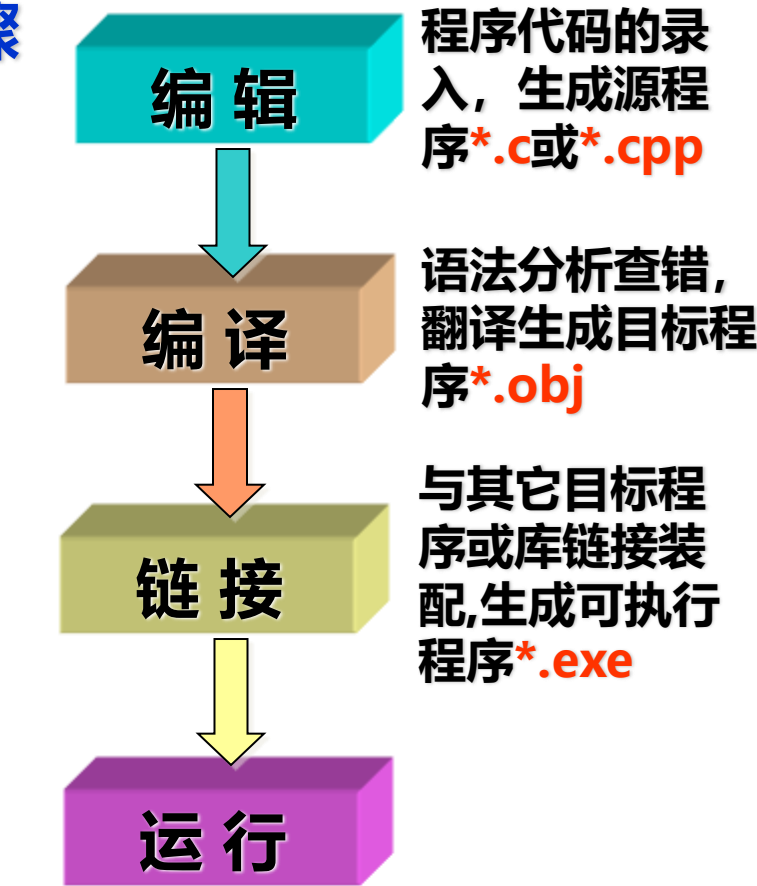
Diagram annotations:

- `/* This is the first C program */`: 注释信息 (Comment information)
- `#include <stdio.h>`: 预处理命令 (Preprocessor command)
- `int`: 返回值为整型 (Return value is integer)
- `main ()`: 无参数、返回值为整型的主函数 (Main function with no parameters and return value of integer)
- `main`: 主函数名 (Main function name)
- `{`: 函数开始 (Function start)
- `printf`: 内部函数名 (Internal function name)
- `"Hello World!\n"`: 回车换行符 (Carriage return and line feed)
- `printf ("Hello World!\n");`: 函数调用 (Function call)
- `0`: 参数 (Parameter)
- `return`: 语句结束标志 (Statement end marker)
- `return 0;`: 函数返回值 (Function return value)
- `}`: 函数结束 (Function end)
- `{ ... }`: 函数体 (Function body)

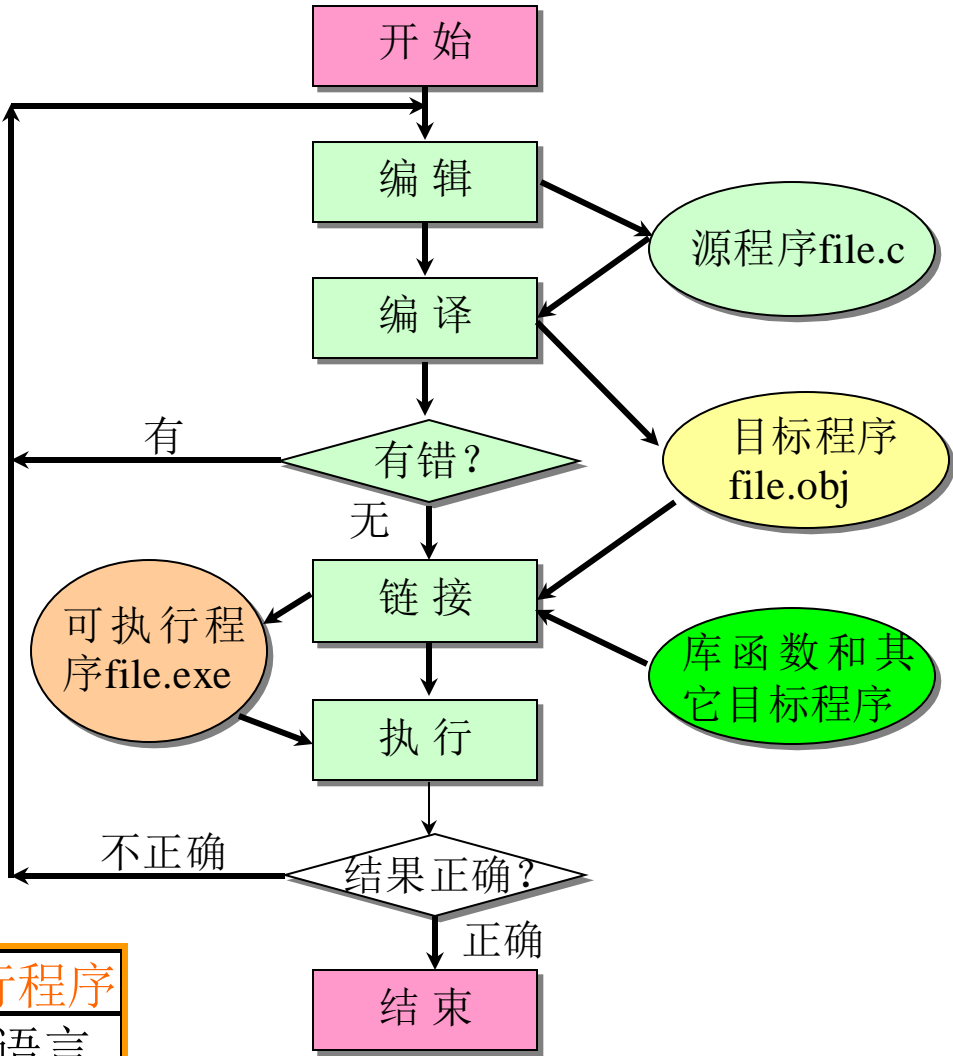
运行结果: Hello World!

1.2 最简单的C++程序?

回顾：编程步骤

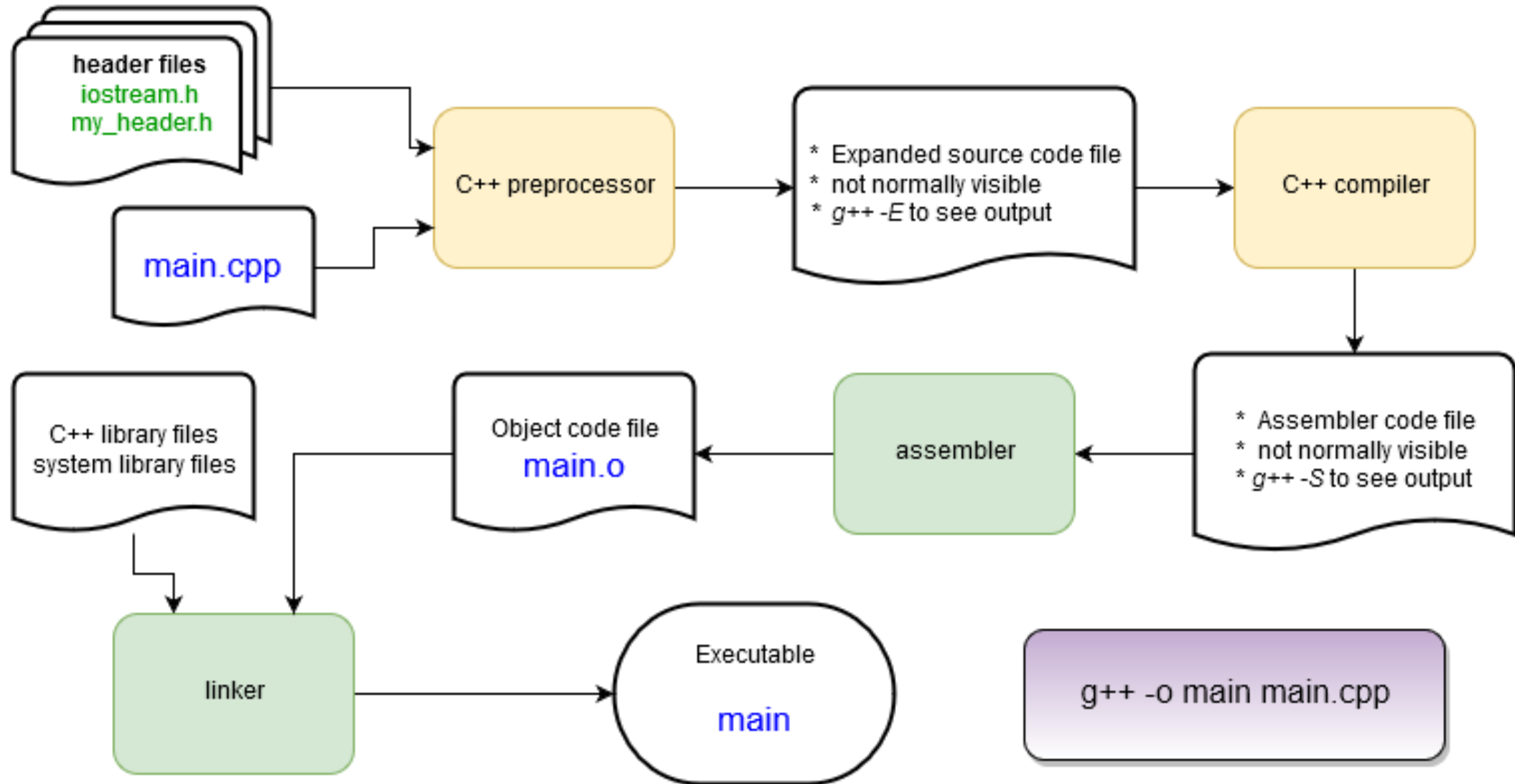


	源程序	目标程序	可执行程序
内容	程序设计语言	机器语言	机器语言
可执行	不可以	不可以	可以
文件名后缀	.c或.cpp	.obj	.exe



调试C/C++程序的流程

Behind the Scenes: The Compilation Process



1.2 最简单的C++程序?

例1.1 输出一行字符 “Hello World! This is a C++ program.”。

```
#include <iostream> //用cout输出时需要用此头文件
using namespace std; //使用命名空间std
int main()
{
    cout<<"Hello World!";
    cout<<"This is a C++ program.\n"; //输出一行
    return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{
    cout<<"Hello world!"<<endl;
    return 0;
}
```

代码基本结构

输入输出方法?

程序的注释方法?

命名空间

程序运行过程

编程工具



什么是命名空间?

using namespace std;

C++新引入的一个机制，主要是为了解决多个模块间命名冲突的问题，就像现实生活中两个人重名一个道理。C++把相同的名字都放到不同的空间里，来防止名字的冲突。

```
#include<iostream>
using namespace std;
int main()
{
    cout<<"Nice to meet you!"<<endl;
    return 0;
}
```

```
#include<iostream>
int main()
{
    std::cout<<"Nice to meet you!"<<std::endl;
    return 0;
}
```

```
#include<iostream>
using std::cout;
using std::endl;
int main()
{
    cout<<"Nice to meet you!"<<endl;
    return 0;
}
```

C++新标准中使用不带.h的头文件包含时，必须要声明命名空间，并且包含头文件在前，声明使用的名字空间在后。

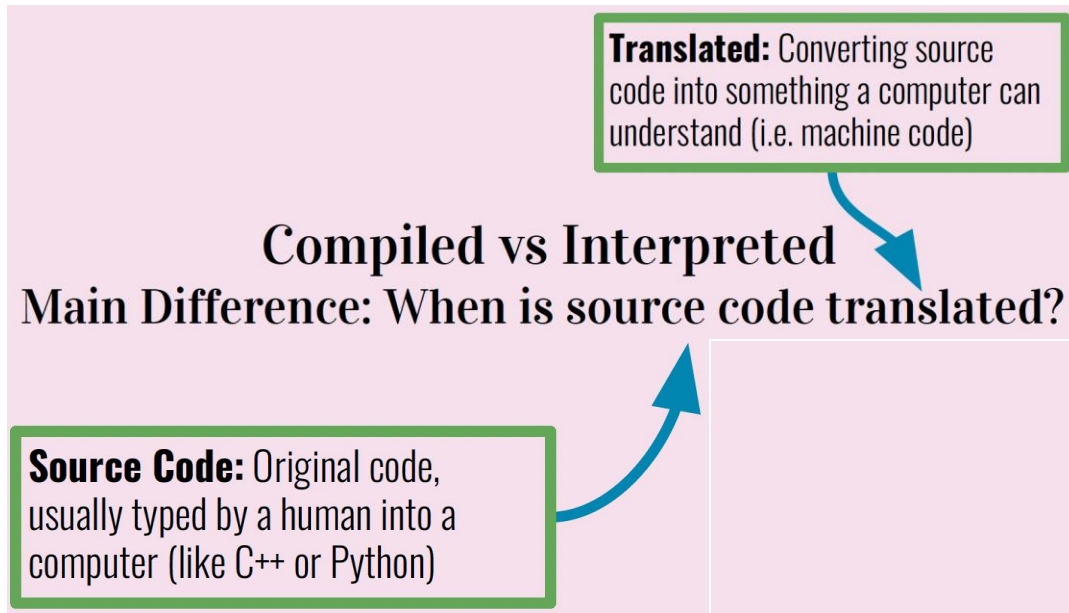
A note about STL naming conventions

- **STL** = Standard Template Library
 - Contains TONS of functionality (algorithms, containers, functions, iterators) some of which we will explore in this class

1.2 最简单的C++程序?

➤ Compiled, Interpreted, Dynamically/Statically typed

CS 106L



Compiled vs Interpreted: When is source code translated?

Dynamically typed, interpreted

- Types checked on the fly, during execution, line by line
- Example: Python

Statically typed, compiled

- Types before program runs during compilation
- Example: C++

Runtime: Period when program is executing commands (after compilation, if compiled)

static typing helps us to prevent errors before our code runs

C++ is a statically typed language

statically typed: everything with a name (variables, functions, etc) is given a type **before runtime**

dynamically typed: everything with a name (variables, functions, etc) is given a type **at runtime** based on the thing's current value

1.2 最简单的C++程序?

➤ **Types** and Structs in C

CS 106L

C++ Fundamental Types

int val = 5; //32 bits

char ch = 'F'; //8 bits (usually)

float decimalVal1 = 5.0; //32 bits (usually)

double decimalVal2 = 5.0; //64 bits (usually)

bool bVal = true; //1 bit

#include <string>

std::string str = "Sarah";

1.2 最简单的C++程序?

➤ **Types** and Structs in C

CS 106L

Fill in the types

```
string a = "test";  
double b = 3.2 * 5 - 1;  
int c = 5 / 2;  
int d(int foo) { return foo / 2; }  
double e(double foo) { return foo / 2; }  
int f(double foo) { return int(foo / 2); }  
void g(double c) { std::cout << c << std::endl; }
```


1.2 最简单的C++程序?

➤ Types and **Structs** in C

CS 106L

```
struct Student {  
  string name; // these are called fields  
  string state; // separate these by semicolons  
  int age;  
};  
  
Student s;  
s.name = "Sarah";  
s.state = "CA";  
s.age = 21; // use . to access fields
```

Definition

struct: a group of named variables *each with their own type*. A way to bundle different types together



1.2 最简单的C++程序?

例1.2 求a和b 两个数之和。

```
// 求两数之和(本行是注释行)
#include <iostream> //预处理指令
using namespace std; //使用命名空间std
int main() //主函数首部
{ //函数体开始
    int a,b,sum; //定义变量
    cin>>a>>b; //输入变量a和b的值
    sum=a+b; //赋值语句
    cout<<"a+b="<<sum<<endl; //输出语句
    return 0; //如程序正常结束,向操作系统返回一个零值
} //函数结束
```

1.2 最简单的C++程序?

例1.3 求 a 和 b 两个数中的大数。

```
//例1.3 求两个数中的大数
#include <iostream>
using namespace std;
int main()
{
    int max(int x,int y); //对max函数作声明
    int a,b,c;
    cin>>a>>b;
    c=max(a,b); //调用max函数
    cout<<"max="<<c<<endl;
    return 0;
}
```

```
int max(int x, int y) //定义max函数
{
    int z;
    if(x>y) z=x;
    else z=y;
    return(z);
}
```



1.2 最简单的C++程序?

例1.4 包含类的C++程序。

```
//例1.4包含类的C++程序
#include <iostream>
using namespace std;
class Student // 声明一个类，类名为Student
{
private: // 以下为类中的私有部分
    int num; // 私有变量num
    int score; // 私有变量score

public: // 以下为类中公用部分
    void setdata() // 定义公用函数setdata
    { cin>>num; // 输入num的值
      cin>>score; } // 输入score的值

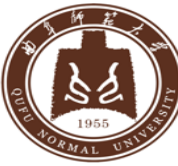
    void display() // 定义公用函数display
    { cout<<"num="<<num<<endl; // 输出num的值
      // 输出score的值
      cout<<"score="<<score<<endl; };
}; // 类的声明结束
```

```
Student stud1,stud2;
//定义stud1和stud2为Student类的变量，称为对象

int main() // 主函数首部
{
    // 调用对象stud1的setdata函数
    stud1.setdata();
    // 调用对象stud2的setdata函数
    stud2.setdata();
    // 调用对象stud1的display函数
    stud1.display();
    // 调用对象stud2的display函数
    stud2.display();
    return 0;
}
```

小结

1. 了解信息科学前沿方向和程序设计语言的流行度
2. 理解面向对象的基本思想
3. 掌握C++编程的基本步骤和编程环境使用方法
4. 掌握C++程序的基本特点
5. 初步理解包含类的C++程序



实验作业与习题

实验作业一 等额本金与等额本息还款计算器

因购房需要从银行贷款（程序可在运行时自行设定数额），贷款期限为10年（或15，20年，程序可在运行时自由设定），程序可在运行时自由设定贷款利率（请查询当前各大银行最新利率），请通过网络查询等额本金和等额本息两种固定利率还款方式的算法，编写程序计算两种方式最终还款总额，并输出贷款年限内每月还款数额。

- 要求：**
1. 用C++语言设计类实现基本功能，功能正确，界面友好。
 2. 代码注释完整、清晰。
 3. 按要求的格式给出实验报告和总结。
 4. 两周内完成，在知新平台上传pdf版实验报告，并准备汇报PPT。
 5. 2025-03-10课上随机抽取同学进行PPT讲解和演示。
 6. 一旦发现雷同或抄袭痕迹，本次实验作业判零分。

2025-02-17



Thank You !

Q & A