

# 从 C 到 C++

胡船长

初航我带你，远航靠自己

《船说：C++零基础到高级》  
第1章-从C到C++

# 一、概览：从C到C++

1. C++的组成部分
2. C++风格的头文件变化
3. 老概念新形式：名称空间
4. 比指针更方便的『C++ 引用』

## 二、C++的输入输出

1. printf 的接班人：cout
2. scanf 的接班人：cin
3. printf 与 scanf 应该被打入冷宫么？

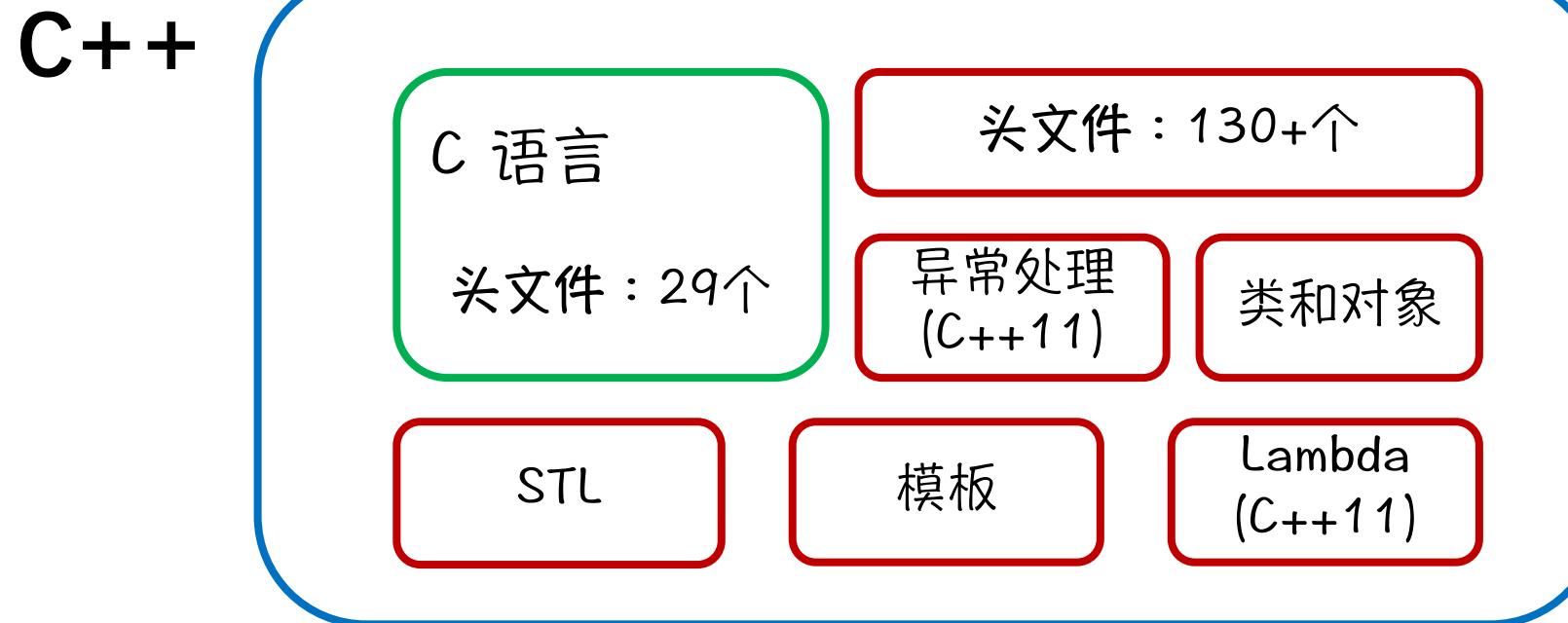
### 三、体验：C++中的编程工具

1. string 类的使用
2. map 与 set 的使用
3. 用 C++ 创造一个大整数类型
4. 更方便的函数指针：function

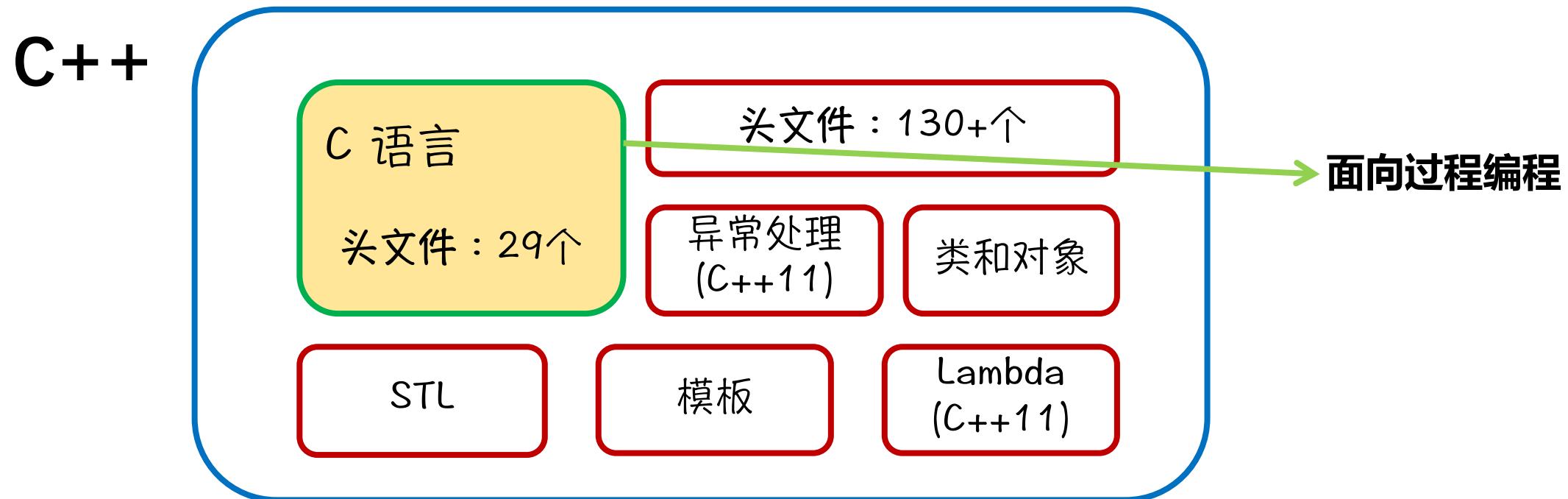
# 一. 概览：从C到C++

# 1-1. 从C到C++到底多了什么？

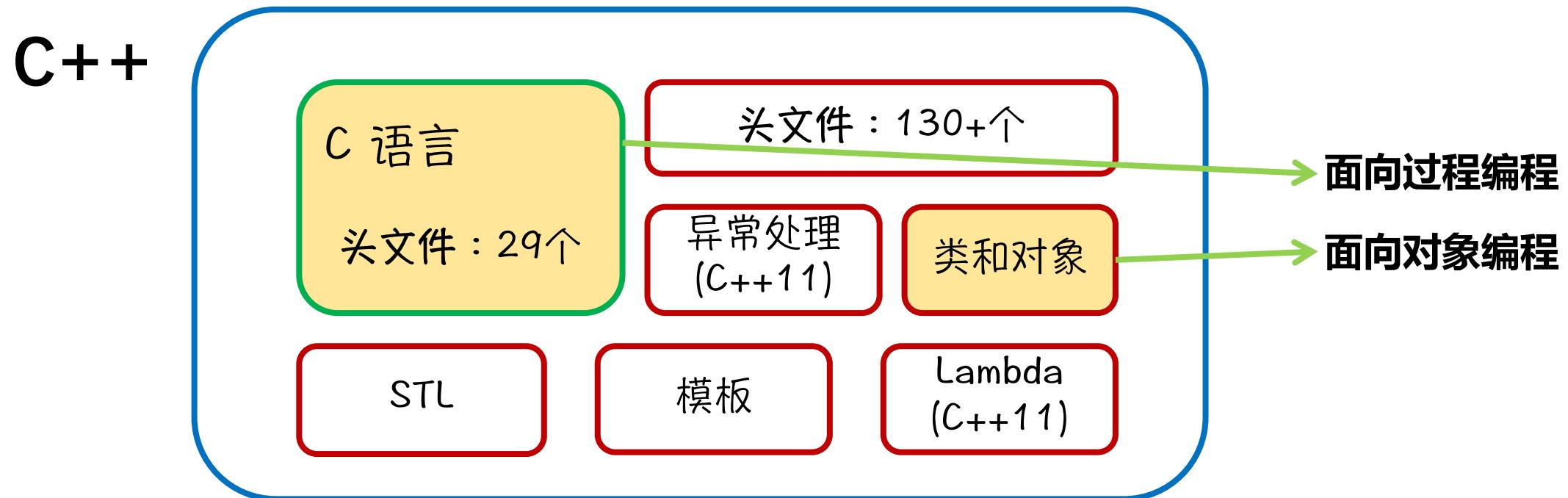
# 从C到C++到底多了什么？



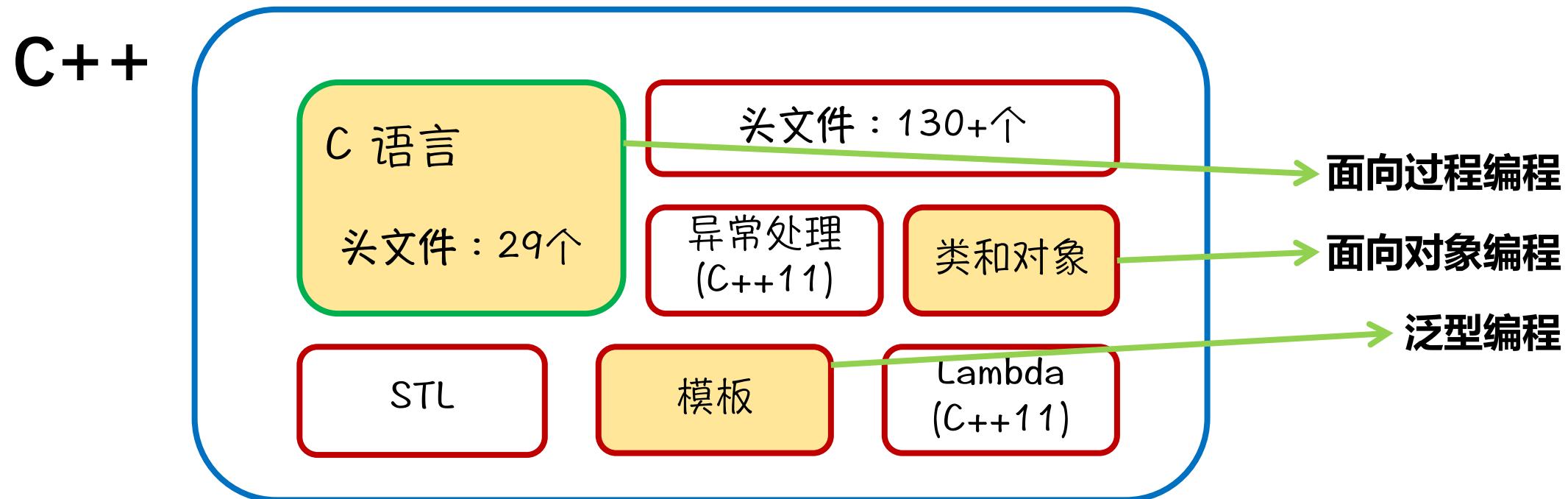
# 从C到C++到底多了什么？



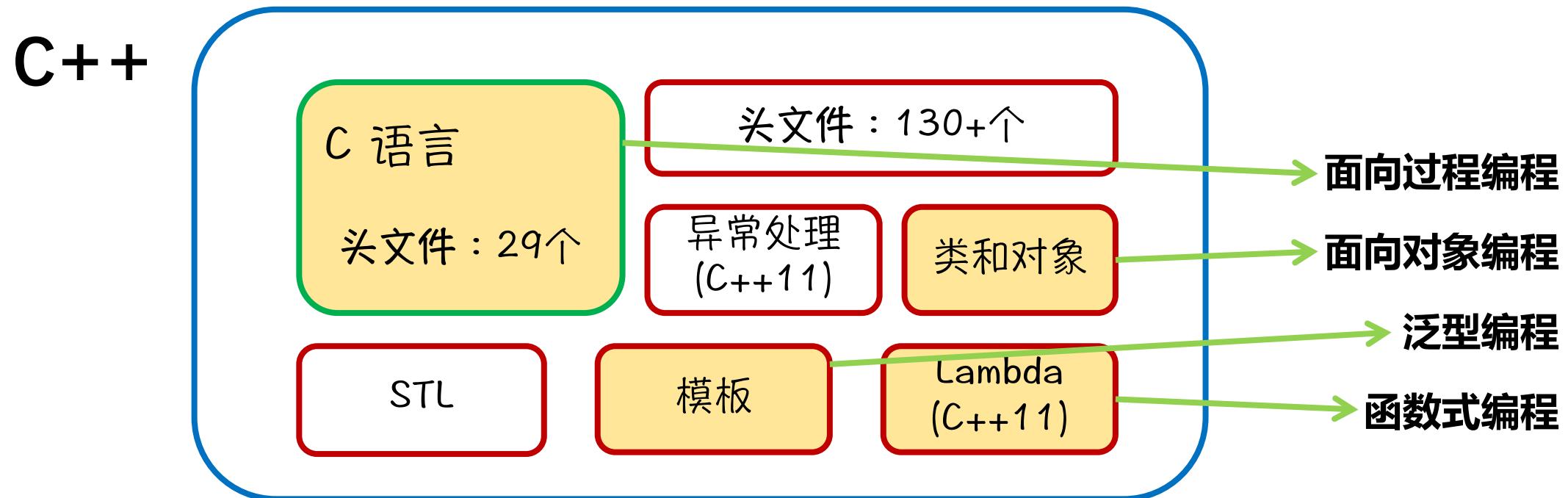
# 从C到C++到底多了什么？



# 从C到C++到底多了什么？



# 从C到C++到底多了什么？



# 编程范式的对比

	C 语言	C++ 语言
面向过程编程	✓	✓
面向对象编程	✗	✓
泛型编程	✗	✓
函数式编程	✗	✓

# 编程范式的对比

	C 语言	C++ 语言
面向过程编程	✓	✓
面向对象编程	✗	✓
泛型编程	✗	✓
函数式编程	✗	✓

# 1-2. C++风格的头文件变化

# 头文件风格的变化

头文件类型	形 式	示 例
C++旧式风格	以.h结尾	iostream.h
C 旧式风格	以.h结尾	stdio.h
C++新式风格	没有扩展名	iostream
转换后的 C	没有扩展名，加上前缀 c	cstdio

# 1-3. 老概念新形式：名称空间

# 名称空间：基本语法

```
namespace name {
```

各种定义；

```
}
```

# 名称空间：核心作用

```
namespace name {
```

各种定义；

```
}
```

按照名字，**分组管理**全局标识符

# 名称空间：如何使用

```
namespace name {  
    各种定义;  
}
```

1. 通过域限定符 :: 访问
2. 通过 using 引入

# 名称空间：总结

1. 避免程序中，不同代码段中相同变量名的冲突
2. 提高程序的『可读性』、『可维护性』与『可扩展性』
3. 匿名的名称空间，可以取代『静态全局变量』
4. 类是一种特殊的名称空间

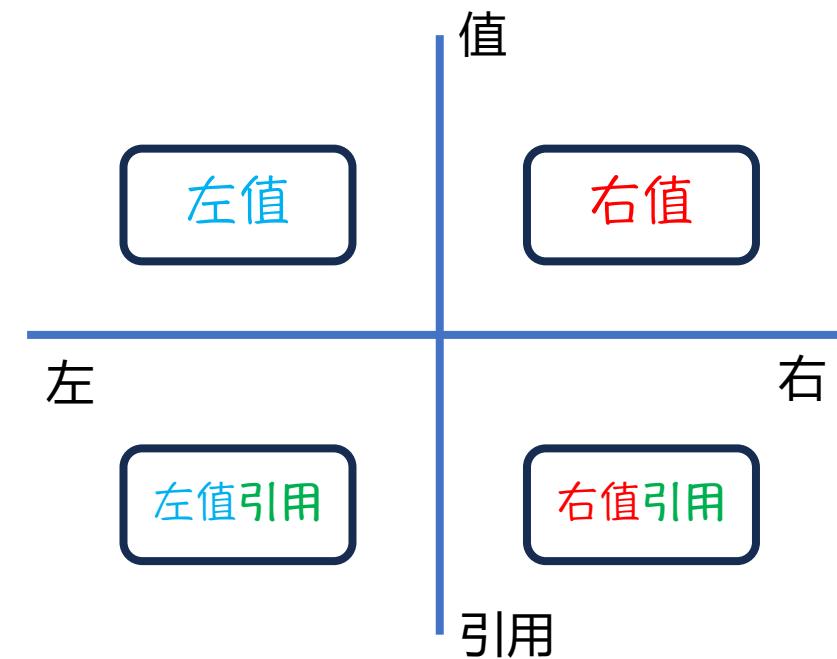
# 1-4. 比指针更方便的『C++引用』

# 从一段代码开始

```
11 void swap(int *a, int *b) {  
12     int c = *a;  
13     *a = *b;  
14     *b = c;  
15     return ;  
16 }
```

# C++引用：总结

1. 区分三个概念：实参、形参、引用参数
2. 引用需要在定义时，完成『绑定』
3. 左值、右值、左值引用，右值引用



# 二. C++的输入输出

## 2-1. printf 的接班人： cout

## 2-2. scanf 的接班人： cin

# 2-3. printf 与 scanf 应该被打入冷宫么？

# printf与scanf应该被打入冷宫的理由

1. printf 与 scanf 是旧时代产物，我们新时代不用
2. cin 和 cout 明显更方便
3. printf 与 scanf 存在安全风险

结论：各有优势，关键在『用』

# 三. 体验：C++中的编程工具

# 3-1. string 类的使用

# string 类说明

**string** 类：字符串

头文件：**string**

命名空间：**std**

声明：**string s1, s2;**

<code>s1 == s2</code>	字符串判等
<code>s1 &lt; s2</code>	字典序小于
<code>s1 &gt; s2</code>	字典序大于
<code>s1 += s2</code>	字符串连接
<code>s1.length()</code>	字符串长度
<code>s1.substr(pos, n)</code>	从 pos 处向后取 n 位

# string 类说明

string 类：字符串

头文件：string

命名空间：std

声明：string s1, s2;

1. HZOJ-166: 字符串操作1
2. HZOJ-167: 字符串操作2
3. HZOJ-170: 禁止吸烟

## 3-2. map 与 set 的使用

# map 类说明

map 类：映射表

头文件：map

命名空间：std

声明：`map<key_type, value_type> arr;`

<code>arr.find(key)</code>	判断某个 key 值是否在 map 中
<code>arr[key] = value</code>	将 value 存储在 key 位上
<code>arr[key]</code>	访问 key 值对应的 value
<code>arr.begin()</code>	映射表的起始位置
<code>arr.end()</code>	映射表的结束位置

# set 类说明

**set** 类：有序集合

头文件：**set**

命名空间：**std**

声明：**set<data\_type> s;**

<code>s.insert(data)</code>	将 data 插入到集合 s 中
<code>s.find(data)</code>	查找集合 s 中是否存在元素 data
<code>s.erase(iter)</code>	删除 iter 指向的元素
<code>s.begin()</code>	有序集合的起始位置
<code>s.end()</code>	有序集合的结束位置

## 3-3. 用 C++ 创造一个大整数类型

画饼：当你掌握了『继承』与『运算符重载』

# 3-4. 更方便的函数指针：function

# 回忆：函数指针

```
int (*add) (int, int);
```

# 回忆：函数指针

C风格：`int (*add) (int, int);`

C++风格：`function<int(int, int)> add;`

# 回忆：函数指针

C风格 : int (\*add) (int, int);

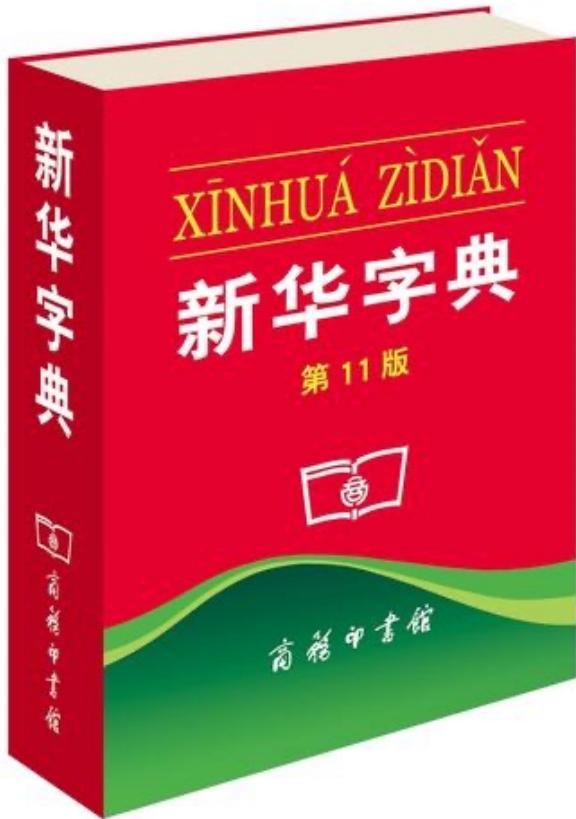
C++风格 : function<int(int, int)> add;

类 型

变 量

# 附言. 那些看似不重要的重要事情

# 学好编程：你需要一本字典



《船说：C++零基础到高级》  
第1章-从C到C++

zh.cppreference.com

学好编程：  
cppreference

## 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 support library

Program utilities  
source\_location (C++20)  
Coroutine support (C++20)  
Three-way comparison (C++20)  
Type support  
numeric\_limits – type\_info  
initializer\_list (C++11)

#### Concepts library (C++20)

#### Diagnostics library

exception – System error  
basic\_stacktrace (C++23)

### 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

#### Parallelism library extensions v2 (parallelism TS v2)

simd

#### Concurrency library extensions (concurrency TS)

#### Transactional Memory (TM TS)

#### Reflection (reflection TS)

### Memory management library

unique\_ptr (C++11)  
shared\_ptr (C++11)  
weak\_ptr (C++11)  
Memory resources (C++17)  
Allocators – Low level management

### 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)  
String conversions (C++17)  
Formatting (C++20)  
bitset – Bit manipulation (C++20)  
Debugging support (C++26)

### Strings library

basic\_string – char\_traits  
basic\_string\_view (C++17)  
Null-terminated strings:  
byte – multibyte – wide

### 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)

#### Algorithms library

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

#### Numerics library

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

#### Date and time library

Calendar (C++20) – Time zone (C++20)

#### Localization library

locale – Character classification  
text\_encoding (C++26)

#### 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)

#### Regular expressions library (C++11)

basic\_regex – Algorithms  
Default regular expression grammar

#### 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)

# 学好编程：编码规范

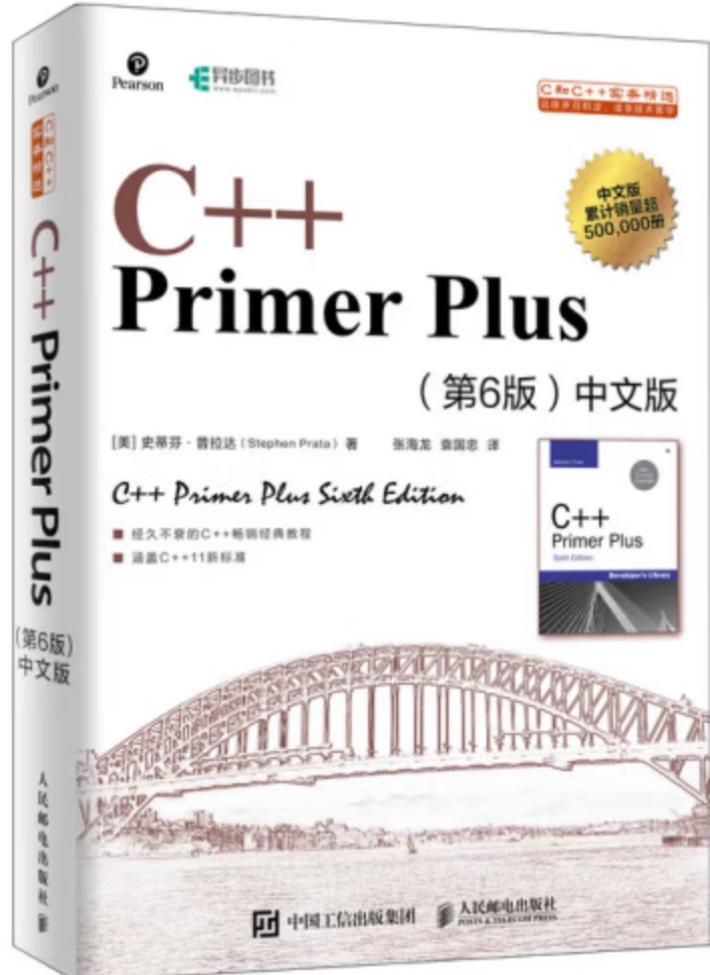
百度 + 谷歌  
编码规范

The screenshot shows the first page of the Baidu C++ Coding Standard document. At the top, it says "C++ 编码规范" and "标准委员会(c-styleguide@baidu.com)". Below that, it lists the main committee members: 石远(PS, 主席) 殷海波(BDG) 陈力捷(BDG) 包能辉(PCS) 吕斯哲(ECOM) 李衡宇(CID) 李瀚 (LBS RD) 蒋锦鹏(PS). It also mentions other participants: 曾凡松(INF) 陈宇(OP) 谭卓鹏(URD) 许健(CID) 韩凤娟(INF) 张倩琼(BDG) 周岩(INF). The Windows C++ standard committee is also listed. The main content is a table of contents for the standard, organized into sections like "前言", "语法", "面向对象编程", "类与结构", "字的选择", "构造", "函数职责", "默认", "构造函数", "其他C++特性和风格约定", "命名空间", "头文件引用", "运算符", "重载", "函数/方法重载", "默认参数", "数", "运行时", "全局", "文件组织", "注释", "命名规范", "顺序", "#include<...>和#include"等。

阿里 + 谷歌  
编码规范

The screenshot shows the first page of the Alibaba and Google C++ Coding Standard document. At the top, it says "Google C++ 编程风格指南". In the center, there is a large Google logo. The page content is mostly blank, suggesting it might be a cover or a page with a placeholder.

# 学好编程：工具书

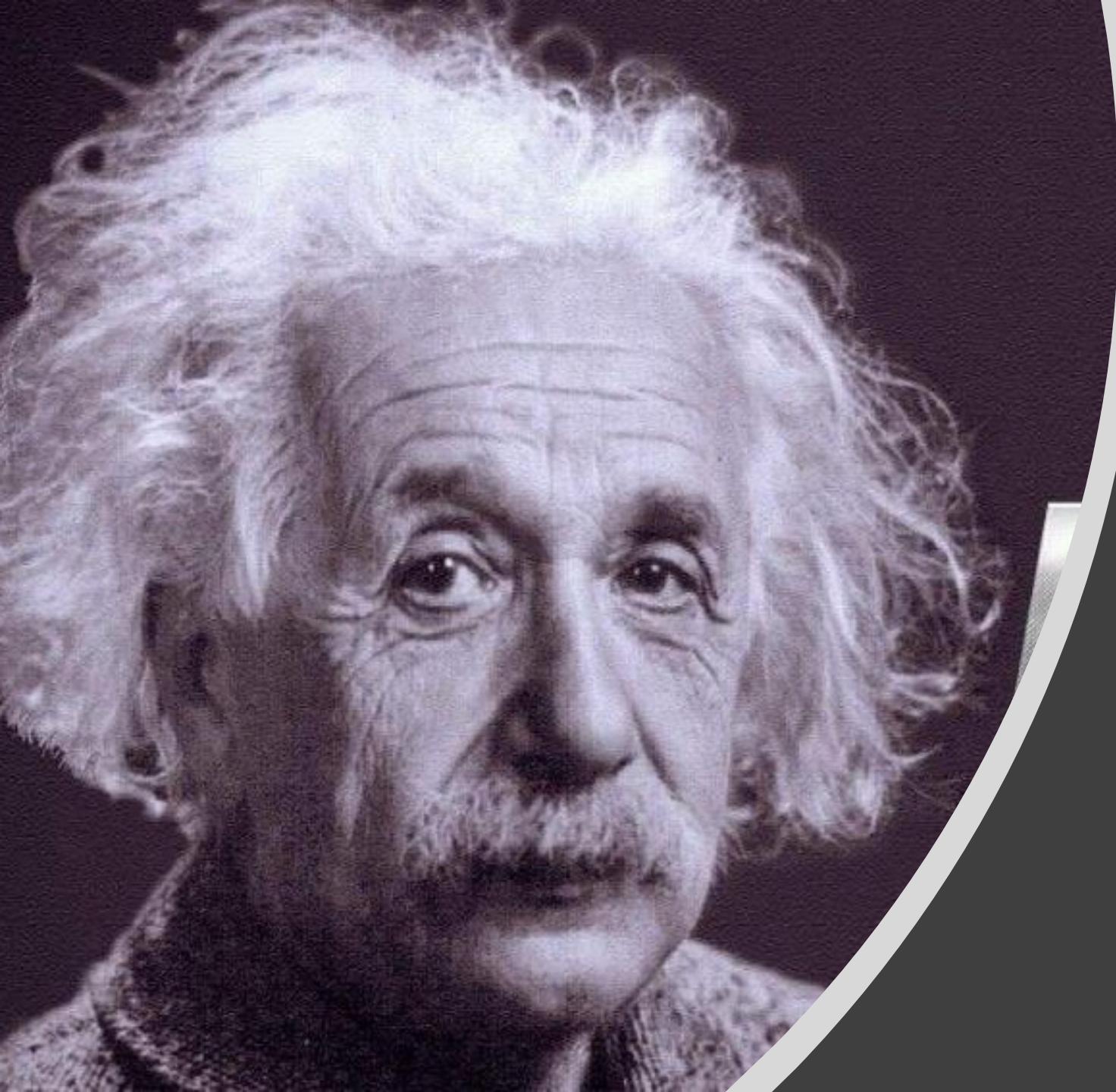


书 名: C++ Primer Plus(第6版) 中文版  
出版社: 中国工信出版集团 人民邮电出版社

## 阅读建议:

- 不要把这本书当教材
- 不要从这本书开始学习
- 不要零散的学习其中的知识点

《船说：C++零基础到高级》  
第1章-从C到C++



为什么  
会出一样的题目？