# C++程序设计精要教程

华中科技大学

辜希武
智能分布式计算(IDC)实验室
1694551702@qq.com

# 教材和参考资料

教材：C++程序设计实践教程（新国标微课版）

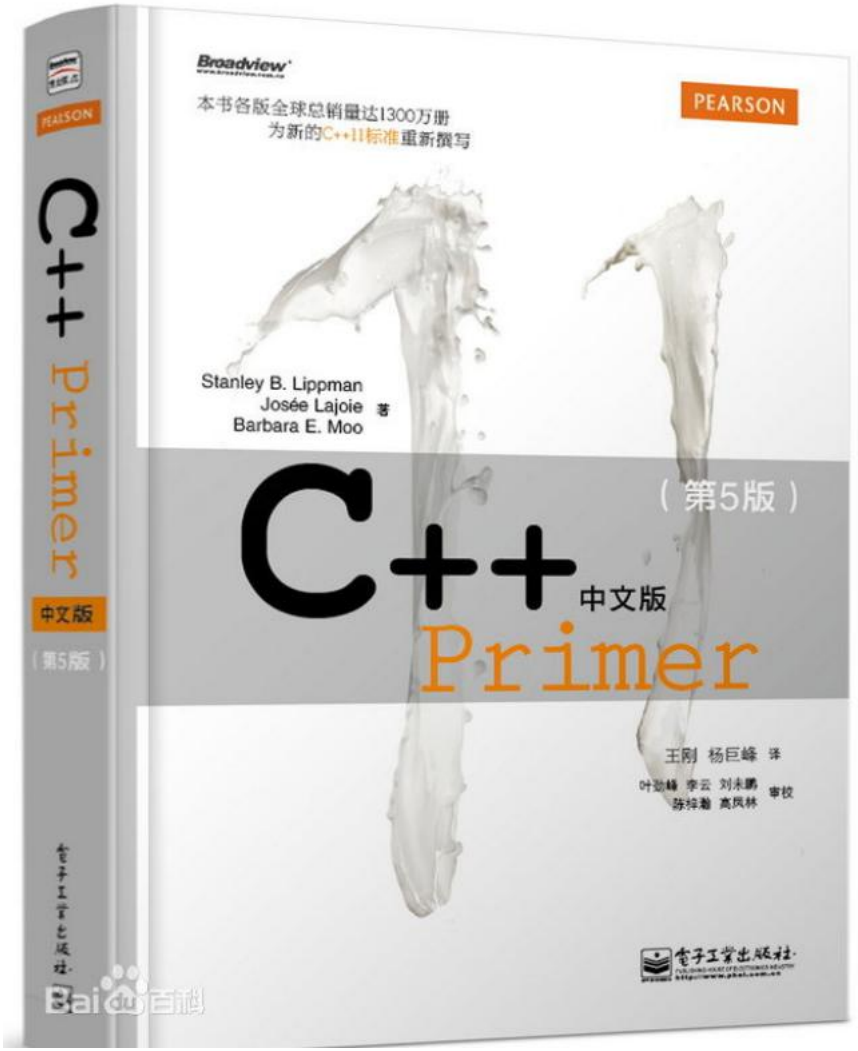
出版：华中科技大学出版社

编著：马光志

参考文献：C++ Primer（第五版）

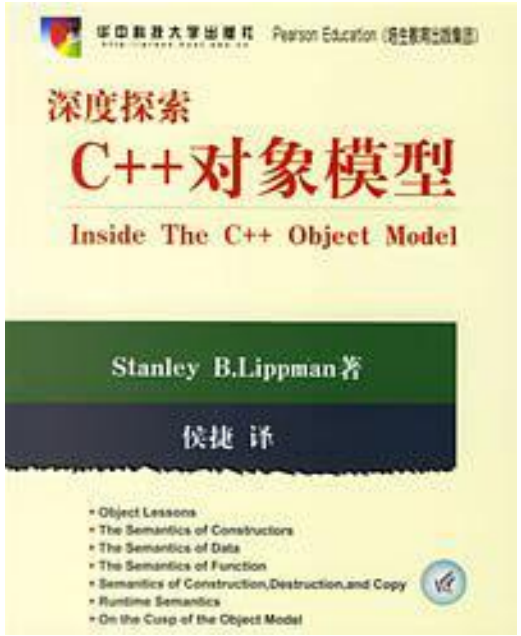
深度探索C++对象模型

C++ 11标准




其内容是C++大师Stanley B. Lippman丰富的实践经验和C++标准委员会原负责人Josée Lajoie对C++标准深入理解的完美结合。

Stanley B. Lippman曾经是迪士尼动画公司的首席软件设计师。当他在AT&T Bell实验室的时候，领导了cfront（第一个C++编译器）编译器开发组。他也是Bell实验室Foundation项目的成员之一，负责C++程序设计环境中的对象模型部分。



# 第1章 C++引论

## ◆ 程序设计语言

机器语言： 计算机自身可以识别的语言(CPU指令)

汇编语言： 接近于机器语言的符号语言（更便于记忆，如MOV指令）

高级语言： 更接近自然语言的程序设计语言，如ADA、C、PASCAL、FORTRAN、BASIC(面向过程，程序基本单元是函数)

面向对象的语言：描述对象“特征”及“行为”的程序设计语言，如C++、Java、C#、SMALLTALK等（程序基本单元是类）

# 第1章 C++引论

## ◆ 程序编译技术

**编译过程**：预处理、词法分析、语法分析、代码生成、模块连接。

**预处理**：通过#define宏替换和#include插入文件内容生成纯的不包含#define和#include等的C或C++程序。

**词法分析**：产生一个程序的单词序列（token）。一个token可以是保留字如if和for、标识符如sin、运算符如+、常量如5和 "abcd"等。

**语法分析**：检查程序语法结构，例如if后面是否出现else。

**代码生成**：生成低级语言代码如机器语言或汇编语言。C和C++语言的标识符编译为低级语言标识符时会换名，C和C++的换名策略不一样。代码生成的是中间代码（如.OBJ文件）

**模块连接**：将中间代码和标准库、非标准库连接起来，形成一个可执行的程序。静态连接是编译时由编译程序完成的连接，动态连接是运行时由操作系统完成的连接。

不同厂家对C++标准的支持程度不一样。一定要确认当前使用的编译器是否支持C++11甚至11以上的标准。

# 第1章 C++引论

## ◆ 程序编译技术

例如假设用户的程序需要调用函数f，f的实现有2个版本，静态库f.lib及动态链接库f.dll。

静态链接：将用户程序生成的obj文件和f.lib链接，并将被调函数f的函数体拷入目标语言程序中（如exe文件），目标语言程序开始执行时，被调函数f的函数体一起装入内存。如果有多个程序都调用f并且都是用静态链接，那么内存中将会有多个f的副本。

动态链接：将用户的obj文件和f.dll链接。只在目标语言程序中保存被调函数f的描述信息。运行时动态链接的函数体f并不随目标语言程序装入内存，只有当调用该函数时才将该函数体装入内存，并保证同一个函数不会在内存里出现多个副本。

- 预处理的例子：
  - 假如stdio.h的文件内容如下：

```
extern  int  scanf (const char *,  …)  ;
extern  int  printf (const char *,  …)  ;
```

  - 程序test.c的文件内容如下：

```
#include <stdio.h>
#define   pi   3.14
void main ( )  {  printf ("area=%lf",  pi*5*5);  }
```

- 预处理的结果，由test.c文件得到如下内容：

```
extern int scanf (const char *,  …)  ;
extern int printf (const char *,  …)  ;
void main ( )  {  printf ("area=%lf",  3.14*5*5)  ;  }
```

# 第1章 C++引论

◆ 面向对象的程序设计语言

纯OO型语言：程序全部由类构成。SMALLTALK、JAVA、 C#、OBJECT-ORIENTED PASCAL。

混合型OO语言：程序由类、全局过程或函数以及全局变量定义构成。如C++。

什么叫全局函数或变量：不包含在任何{}里的变量和函数

# 第1章 C++引论

## ◆面向对象的基本概念

**函数绑定**：函数调用和函数入口的关联过程（找到函数入口地址）。

- 早期绑定：发生在程序开始执行以前，由编译程序静态连接，或者由操作系统动态连接完成，将函数入口地址填写到函数调用处。**程序运行前入口地址已经确定**
- 晚期绑定：发生在程序执行过程中间，由程序自己完成。

**对象**：现实世界具体的或抽象的“事物”，经历产生、活动、死亡等阶段（**生命周期**）。体育比赛的“运动员”和“赛局”分别为具体和抽象的对象。

**类**：描述对象**共同**特征和行为的类型（class）。有结构的类属于复杂类型。简单类型和复杂类型变量(对象)初始化(产生)形式趋向统一。
对象既可以是变量，也可以是常量。就象简单类型既有变量也有常量一样。统一起来，对象分为简单类型对象和复杂类型对象。

# 示例

- 下面是一个圆类：

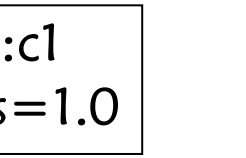
数据成员

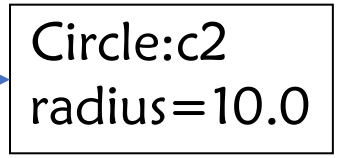
构造函数声明

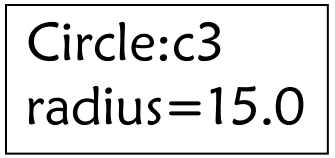
函数成员声明

```
class Circle {
private:
    double radius ;

public:
    Circle() ;
    Circle(double r);

    double findArea() ;
};
```

Circle.h




构造函数定义

函数成员定义

```
#include  “Circle.h”
Circle::Circle() { radius = 1.0;}
Circle::Circle(double r) { radius = r;}

double  Circle::findArea(){
    return radius * radius * 3.14;
}
```

Circle.cpp

```
Circle c1(10.0);
int i(0;)  //C++ Style
int i = 0; //C Style
```

# 第1章 C++引论

## ◆面向对象的基本概念

**封装**：将对象的“特征”(数据结构)和“行为”（算法）包装在一起，形成对象的类型定义，分别描述对象的“组织结构”和“功能”。封装定义了对象的边界，提供了外部访问的接口，屏蔽了对象的“行为”细节。

**交互**：**直接交互**指一对象调用另一对象的“操作”、“功能”或“函数”；**间接交互**通过发送或监听消息完成。

C++程序的对象既可以直接交互，也可以通过操作系统提供的消息机制间接交互。

# 第1章 C++引论

## ◆面向对象的基本概念

**重载：用一个函数名称来定义完成不同功能的多个函数**，参数个数和类型随完成功能的不同而不同。将运算符看作函数，操作数就是参数。-5和8-3分别是有一个和两个参数的减法函数，可记为operator- (int)、operator- (int, int) 。

C++已经自嵌入地重载了简单类型运算函数，故不允许对简单类型进行运算符重载。换言之，运算符重载函数的参数不能都为简单类型，必须至少有一个参数代表对象。例如－5、8－3调用的是自嵌入的减法运算重载函数(函数名见上)。重载函数要么参数个数不同，要么参数类型不同。

# 第1章 C++引论

## ◆面向对象的基本概念

**多态**：**通过一个函数名调用函数能表现出不同行为。**早期绑定（编译时）的多态是**静态多态**，晚期绑定（运行时）的多态是**动态多态**。重载函数表现的行为是静态多态，虚函数表现的行为是动态多态。由此可见，重载函数使用早期绑定，虚函数使用晚期绑定。“多态”一般指动态多态。**绑定越晚越好**。

**继承**：一个对象获得另一个或多个对象的“特征”和“行为”，从而实现了软件重用。例如，小孩长相象父母是获得父母“特征”，走路象父亲是获得父亲“行为”。

# 第1章 C++引论

## ◆面向对象的基本概念

**抽象**：一种抽象形式是从对象(事物)到类型(概念)，另一种形式是从低级类型(概念)到高级类型(概念)。从对象“张三”、“李四”抽象出“学生”的概念，从“学生”、“教师”的概念可抽象出“师生”的概念。

**抽象类**：抽象级别最高的类，无法描述具体特征和行为。例如，从“点”、“线”、“圆”抽象出“图形”的概念。无法说出“图形”有何特征，也无法说明其绘图行为。

# 第1章 C++引论

## ◆C++语言的特点

- C的超集，完全兼容C，代码质量高、速度快。
- 多继承的强类型的混合型的OO语言。
- 支持面向对象的运算符重载：至少一个操作数的类型代表对象的类型。
- 提供函数模板和类模板等高级抽象机制。
- 支持面向对象的异常处理。
- 支持名字空间namespace：解决标识符命名重复的问题

# 第1章 C++引论

## ◆C++的程序结构

- C++的标准输入输出

```
#include <iostream>
 using namespace std;
 int x;
 cin >> x;
 x += 2;
 cout << x;
```

- 在iostream声明了cin和cout。cin为标准输入流对象，cout为标准输出流对象
- <<和>>被重载，分别为cin和cout提供输入输出功能

- 继续支持stdio.h，强类型要求必须先#include头文件再使用，其中头文件包含了变量和函数声明：

  int scanf (const char *, …); //返回成功输入变量的个数。

  int printf (const char *, …);//返回成功输出字符的个数。

  int x=printf ("9876543210"); //结果x=10，输出10个字符。

  int y=printf ("%d", 98765); //y=5正好是十进制有效位数。

- 流iostream类对象（变量）cin通过重载>>运算符函数完成输入。就象+运算符一样，>>可以自左至右连续运算。试比较变量cin、x、y、z的运算：

  x+y+z; //x+y的和（一个值） 再和z进行+运算。

  cin>>y>>z; //cin>>y的结果(为cin)再和z进行>>运算。

  //等于cin>>y; cin>>z

- cin关于>>运算的结果为cin，cout关于<<运算的结果cout。可用cin或cout的运算结果连续进行>>和<<运算。

# 第1章 C++引论

## ◆C++的头文件

C++有三种头文件

- 老式的C头文件，如#include <stdio.h>
  这是为了和C语言兼容

- C++新的头文件，后面没有.h后缀
  #include <iostream>
  using namesapce std;
  注意C++规范明确说明应该使用新的头文件，C++新的头文件没有.h后缀

- 老式C头文件的封装(Wrapper)，封装头文件的命名规则为：老式C头文件名前+字母‘c’，然后去掉.h后缀
  例如stdio.h头文件经过封装后的文件为cstdio
  #include <cstdio>
  using namespace std;

# 第1章 C++引论

## ◆C++的头文件

老式头文件和新的头文件及封装的头文件区别

- C语言没有名字空间的概念，因此头文件里声明的变量、函数都是全局的，因此我们可以直接使用定义的变量和函数，如printf(…)。
- 新的头文件及封装的头文件里声明的变量、函数都位于名字空间std里，这就是为什么在include了新式头文件后，一定要using namespace std;后才能使用头文件里的变量和函数，如cin >> x;
- 如果不使用using namespace std；只有通过std前缀来使用头文件声明的变量和函数，如

  std::cin >> x;

# 第1章 C++引论

## ◆C++的头文件

老式头文件是如何被封装的？以cstdio为例，下面是部分代码：

```
#include <stdio.h>
namespace std {
        …
        using ::printf; //::为缺省名字空间
        …
}
```