# 大道至简

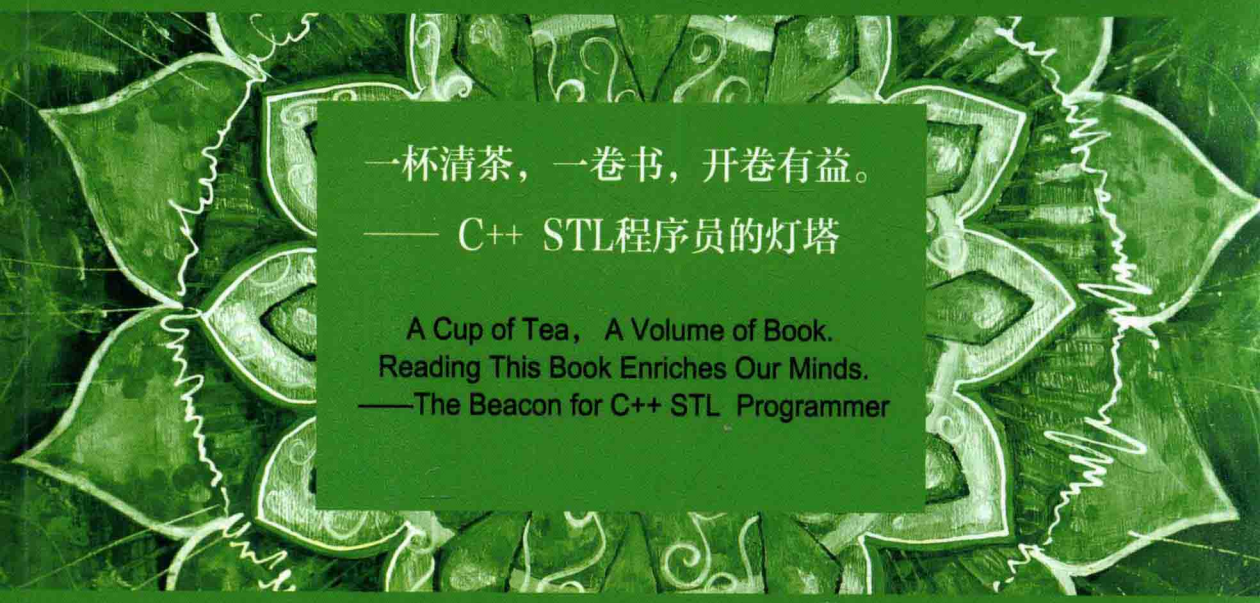
## ——C++ STL

## (标准模板库)精解

The Greatest Truth is The Simplest —

Precision Expounding on C++ STL (Standard Template Library)

闫常友 编著




机械工业出版社
CHINA MACHINE PRESS

# 大道至简

## ——C++ STL（标准模板库）精解

闫常友　编著



机械工业出版社

众所周知，C++是在C的基础上发展起来的。它进一步扩充和完善了C语言，是一种面向对象的程序设计语言。经过几十年的发展，C++现已支持多种编程规范，其相应的C++国际标准也在不断更新。C++ STL现在是C++标准程序库的一部分，其作用是对常用数据结构和算法进行封装——标准化组件，以便让程序员可以直接使用现成的算法和数据结构，这样既避免了重复开发，又提高了效率。

本书按照C++ STL的内容结构由浅入深地讲解了其应用开发技术。主要内容共计17章。第1章主要介绍了与C++ STL相关的基本概念和基础知识，并简要介绍了本书后面会用到的一些模板类型。第2～17章依次介绍了字符串类模板、容器、算法库、迭代器、数值计算模板、输入/输出类模板、异常处理类模板、通用工具类模板、语言支持类模板、检测类模板、国际化类模板、仿函数、配置器、原子操作类、线性控制类模板以及正则表达式模板等内容。由于C标准函数库的相关内容在程序开发中的使用频率较高，因此本书的附录部分给出了几乎所有传统C库函数中的数学计算类函数和数值转换类函数，以供读者使用。

本书适合已有一定C/C++基础的读者阅读，也适合有意于深度钻研C++ STL技术的朋友们阅读。

**图书在版编目（CIP）数据**

大道至简：C++STL（标准模板库）精解/闫常友编著. —北京：机械工业出版社，2015.5

ISBN 978-7-111-51399-5

Ⅰ.①大… Ⅱ.①闫… Ⅲ.①C语言—程序设计 Ⅳ.①TP312

中国版本图书馆CIP数据核字（2015）第206435号

机械工业出版社（北京市百万庄大街22号 邮政编码100037）

策划编辑：申永刚 责任编辑：申永刚 吴晋瑜

版式设计：霍永明 责任校对：张晓蓉 刘怡丹

封面设计：马精明 责任印制：乔 宇

保定市中画美凯印刷有限公司印刷

2015年11月第1版第1次印刷

184mm×260mm·46.75印张·1插页·1158千字

0001—3000册

标准书号：ISBN 978-7-111-51399-5

定价：125.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务

服务咨询热线：010-88361066

读者购书热线：010-68326294

010-88379203

编辑热线：010-88379425

网络服务

机工官网：www.cmpbook.com

机工官博：weibo.com/cmp1952

金书网：www.golden-book.com

教育服务网：www.cmpedu.com

# 前　言

本书详细介绍了C++标准模板库（C++ Standard Template Library，C++ STL）的所有内容。本书主要是参照 ISO/IEC 14882 Second edition 的内容编写的，同时也参考了最新的C++ 11 国际标准（ISO/IEC 14882：2011）。

C++经历了多年的发展，其新增特性较多，而C++ STL是众多成员辛勤劳动的结晶，其中大量的可重复代码为广大程序员提供了巨大的方便，节省了大量时间和人力。C++ STL的开发者主要是 Alexander Stepanov、David Musser 以及 MengLee 三位大师，其中 Alexander Stepanov 被誉为“STL之父”。1994 年 7 月，美国国家标准学会（ANSI）通过投票决定，将STL纳入C++标准，使之成为C++库的重要组成部分。

C++ STL作为C++的一部分，历经多次修改，经过多个程序员团队的“精加工”，已经不同于最初的版本。STL为编程人员提供了诸多方便和好处，以前用传统C++编写的复杂代码，现在通过使用STL，仅仅几句话就可以实现。在STL中，模板使用得可谓淋漓尽致。通过使用模板，用户可获得优质并且高效的代码。STL的优越性使其迅速流行起来，并且发展劲头强劲。近些年，国内的STL热也迅速升温，但相关的资料并不丰富，较好的资料更是少之又少。对程序员来说，掌握STL编程技术，并精通C++高级编程技术，是非常有必要的。

本书以广大程序员的角度，详细介绍了C++ STL标准库，对其中的模板技术更是进行了细致深入的讲解。书中通过大量例题（均由作者亲自编写或摘自MSDN）对各知识点进行实例讲解，希望读者认真阅读。

鉴于作者水平有限，书中难免存在不足之处，敬请广大读者批评指正。

# 目　录

# 第1章

# 预备知识及简介

模板是C++的一个新特性。在现今的C++标准模板库中，几乎所有东西都被设计为模板形式。如果不支持模板，就无法使用标准模板库。模板是实现代码重用的一种工具，即针对一个或多个尚未明确的类型，编写一套函数或类型，以供用户使用。通过使用模板，C++允许推迟对某些类型的选择，直到需要对模板进行处理时才使用。使用模板，还可以使程序员针对不同的相似特性开发出更加大众化的代码。

本章首先介绍了C/C++语言中常用的一些基本概念；其次介绍了类模板定义、成员模板、函数模板、友元模板及类模板的参数等内容；最后介绍了模板库的相关知识。

## 1.1 基本概念

本节主要介绍C/C++语言中常用的一些基本概念。通过学习本节内容，读者应能正确使用这些概念，以便更加方便、高效地开发代码。

### 1.1.1 何谓“命名空间”

命名空间（Namespace）是指标识符的可见范围或者有效范围。定义命名空间的目的是为了防止出现名称冲突现象。命名空间是C++中一个较新的特性。为了将多个程序员开发的代码便捷、高效地组合起来，防止出现重复的函数名、类名等，命名空间将不同的代码封装在自己的有效范围内。

命名空间需要使用using来声明，并且每个命名均需要使用using。标准模板库（Standard Template Library，STL）采用命名空间技术解决了大规模软件开发的难题。C++ STL内的所有标识符都被定义为一个名为std的命名空间中，因而可以直接使用指定的标志符号std。例如：

```
std::cout << 2.0 << std::endl;
```

如果在上述语句之前添加以下声明：

```
using std::cout;
using std::endl;
```

那么上述语句可以写为：

```
cout << 2.0 << endl;
```

如果嫌麻烦，需要逐个声明名称，可以直接声明、使用命名空间std，例如：

```
using namespace std;
...
cout << 2.0 << endl;
```

显而易见，使用C++标准库的命名空间 std 之后，可以任意使用标准库中的函数和变量。参见例 1-1。

**例 1-1**

```
#include <iostream>
#include <stdio.h>
using namespace std;
void main()
{
cout << "欢迎您开始步入C++标准模板库!" << endl;          //输出"欢迎您开始步入C++标准模板库!"
getchar();                                              //等待输入任意键,之后程序退出
return;
}
```


在例 1-1 中，使用了命名空间 std，即在 main( ) 函数前添加如下语句：

```
using namespace std;
```

之后在 main( ) 函数中，可以直接使用 cout 函数和 endl 函数。语句 using namespace std 可以放在源代码的任意位置，不同的位置代表 std 不同的有效范围。


本小节主要讲述了命名空间的概念及其最简单的使用方法，并通过例 1-1 使读者对命名空间有一个直接的认识。

### 1.1.2 头文件

在上一节的例 1-1 中，有#include <iostream>这样的语句，其中 iostream 就是被 include 语句包含的头文件。在原来的 C 语言中，头文件都是以 .h 的形式存在的；而在C++ 中，头文件只有名字，没有扩展名，例如 iostream。这部分头文件均在原有名字前添加字符 c 来加以标志，例如 cmath、ctring 等。这类头文件的使用方法如下所示：

```
#include <cmath>
#include <cstring>
```

值得注意的是：在书写头文件名时，一定要认真，不要添加其他字符，例如空格。如果添加了其他字符，有些编译程序会提示无法找到该头文件（Visual C++ 6.0 编译程序可以避免类似的问题）。

头文件的功能主要是将原程序片段收集到一起，形成一个提供给编译程序的文件。一般情况下，头文件中只包含各种声明、常量定义、预编译、注释、类型定义、模板定义等。常规的函数定义、数据定义、导出的模板定义等，不能出现在头文件中。

C++ 标准模板库头文件是标准模板库的外在表现形式。使用标准模板库的唯一途径就是包含相应的头文件。标准模板库头文件是没有后缀的。


请关注头文件 cmath 的使用，并注意例 1-2 中的中文注释。

**例 1-2**

```
#include <iostream>
#include <cmath>                                    //使用头文件 cmath
#include <stdio.h>
using namespace std;                                //使用命名空间 std
void main()
{
float pi=3.1415916;
float radius=0;
float area=0.0;
float girth=0.0;
cout<<"请输入圆的半径:"<<endl;
      cin>>radius;
area=pi* pow(radius,2);                             //使用幂函数 pow();乘方
girth=2* pi* radius;
cout<<"area="<<area<<";  girth="<<girth<<endl;

getchar();                                          //等待程序退出
return;
}
```

提示 在例 1-2 中，使用了 C 语言幂函数 pow（double x，double y）。这个函数有两个参数 x 和 y，函数的功能是求解 x 的 y 次方。

标准模板库提供了大量的头文件，使用头文件主要是为了提供类、函数、变量的声明，以供用户方便地使用这些类、函数、变量等。

### 1.1.3 面向对象的程序设计

面向对象（Object Oriented，OO）是 C/C++ 语言解决问题的一种方法。之前的程序开发是面向过程的。面向过程的程序设计主要考虑解决问题的先后顺序、措施安排等，具有典型的过程性。而面向对象的程序设计主要是建立在各种对象基础上的软件开发，每一个具有独立特性和相应功能的个体均可以作为一个对象来加以处理。

人类社会历史充满了“抽象”的概念，例如天、地、道等。

在软件开发和程序设计领域，同样存在诸多“抽象”的对象。根据软件开发的需要，每个对象被赋予不同的属性和方法（函数）。当该对象被定义时，用户同样可以方便地使用该对象的属性和方法。大多数程序员都曾学过C++ 中类的概念。类的对象一般有构造函数、析构函数等。类的对象可以被复制、被传递或被作为其他类的成员，甚至有些类的成员还可以直接参加数学计算。

面向对象是当前计算机界关心的重点，它是 20 世纪 90 年代软件开发方法的主流。在对对象有了简单认识之后，广大程序员开始就面向对象的程序设计进行各种尝试和研究，历经多年的发展和变革，才有了今天面向对象软件开发技术的现状。如今，面向对象的概念和应

用已超越了程序设计和软件开发，并扩展到了各个领域。

C++类是一种将抽象类型转换为用户自定义类型的方法。该方法将数据表示和操纵数据的方法组合成一个整齐的包。对于类的实现，一般包括两方面：类的声明和类方法的定义。类的声明是指以数据成员的方式描述数据部分，以成员函数的形式描述公有接口。类方法的定义是指描述如何实现类的成员函数。通俗来说，类的声明提供了类的宏观概貌，而类的定义则提供了细节。下面举一个最简单的例子。

提示 请注意例 1-3 中的中文注释。类 Calculator 作为一个简单的计算器，用于实现一些最简单的数学计算功能。

**例 1-3**

```
#include <iostream>                                    //包含输入/输出流功能的头文件
#include <cmath>                                       //包含数学函数库的头文件
#include <stdio.h>                                     //包含标准输入/输出的头文件
using namespace std;                                   //使用命名空间 std
class Calculator                                       //声明类 Calculator
{
public:                                                //将下述定义为公共成员
    Calculator();                                      //构造函数
    ~Calculator();                                     //析构函数
    float x;                                           //成员变量 x
    float y;                                           //成员变量 y
    float Adder(float x,float y);                      //加法运算
    float Substration(float x,float y);                //减法运算
    float Multiplication(float x,float y);             //乘法运算
    float Division(float x,float y);                   //除法运算
    float CPow(float x,float y);                       //乘方运算
    float CSqrt(float x);                              //开方运算
};                                                     //类声明结束
Calculator::Calculator()                               //定义构造函数
{
    x=0;y=0;
}
Calculator::~Calculator()                              //定义析构函数
{

}
float Calculator::Adder(float x,float y)               //定义实现加法运算的成员函数
{   float He=0;
    He=x+y;
    return He;
}
float Calculator::Substration(float x,float y)         //定义实现减法运算的成员函数
{   float Cha=0;
```

```
    Cha = x - y;
    return Cha;
}
float Calculator::Multiplication(float x,float y)          //定义实现乘法运算的成员函数
{   float Ji = 0;
    Ji = x* y;
    return Ji;
}
float Calculator::Division(float x,float y)                //定义实现除法运算的成员函数
{   float Shang = 0;
    if(y == 0.0)
        return -1;
    Shang = x/y;
    return Shang;
}
float Calculator::CPow(float x,float y)                    //定义实现乘方运算的成员函数
{   float ChengFang = 0;
    ChengFang = pow(x,y);
    return ChengFang;
}
float Calculator::CSqrt(float x)                           //定义实现开平方运算的成员函数
{   float sqrtC = 0.0;
    sqrtC = sqrt(x);
    return sqrtC;
}
void main()                                                //程序入口,main()函数
{   float x = 0;
    float y = 0;
    cout << "请输入 x:" ;
    cin >> x;                                              //输入数据 x
    cout << endl;
    cout << "请输入 y:" ;
    cin >> y;                                              //输入数据 y
    cout << endl;
    Calculator my;                                         //定义类的对象
    float He = my. Adder (x,y);
    cout << "两数之和:" << He << endl;
    float Cha = my. Substration(x,y);
    cout << "两数之差:" << Cha << endl;
    float Ji = my. Multiplication(x,y);
    cout << "两数之积:" << Ji << endl;
    float Shang = my. Division(x,y);
    cout << "两数之商:" << Shang << endl;
    float Pow = my. CPow(x,y);
    cout << "x 的 y 次方:" << Pow << endl;
```

```
    float KaiFang = my.CSqrt(x);
    cout << "x 的平方根:" << KaiFang << endl;
    cout << "任意键程序退出";
    getchar();
}
```

程序执行后的效果如图 1-1 所示。

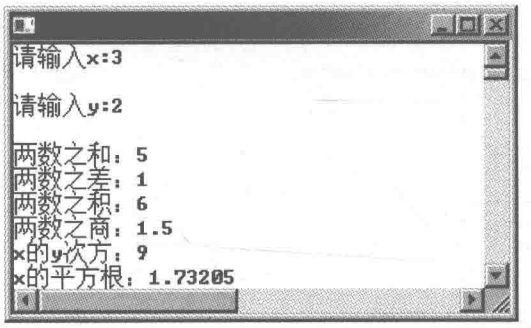



图 1-1　例 1-3 执行效果图

总结　在例 1-3 中，通过声明、定义一个类 Calculator，说明了如何使用类来进行面向对象的编程。通过定义类，形成一个具有一定功能的通用类型或通用模板；使用这个类的对象时，可以调用模板中的所有成员函数。面向对象技术要高于面向过程技术。面向对象的程序开发是给每个对象具备了一定的特殊“能力”，而这些“能力”一般也是通过面向过程的技术实现的。通过本小节的学习，主要理解“面向对象”的概念就可以了。

### 1.1.4　C++ 中的声明和定义

在程序开发领域，声明是指当一个计算机程序需要调用内存空间时，对内存发出的“占位”指令。定义则是指将声明的变量、函数、类等呈现或描述出来，为之提供一个意义相当的表达，并表明其与众不同之处，最终展现其内涵。

在软件开发过程中，常见的声明和定义语句列举如下（注意中文注释语句）。

例如：

```
int a;                                  //声明一个整型变量 a
float b;                                //声明一个浮点型变量 b
struct S                                //声明一个结构 S
    {
      int a;
      float b;
    };
    extern int x;                       //声明一个外部变量 x
    void count(bool YN, int& counter);  //声明一个函数 count
```

上述几行语句均是典型的声明语句。

```
void count (bool YN, int& counter )          //函数 count 的定义
   {
    if (YN)                                  //如果 YN 是 true
      {
       counter ++;                           //计数器加 1
      }
   }
```

上述几行代码对函数 count（bool YN，int& counter）进行定义，用以实现函数的功能。下面通过例 1-4 来验证上述函数的定义。

**例 1-4**

```
#include <iostream>
using namespace std;
void count(bool YN, int& counter);           //函数声明
void count (bool YN, int& counter )          //函数定义
   {
      if (YN)
      {
      counter ++;
      }
   }
void main()                                  //main()主函数
   {
      int counter=0;                         //计数器
      bool YN=1;                             //逻辑变量,用于当条件满足时,退出 while 循环
      while(YN)
      {
      count(YN,counter);                     //计算循环次数
      cout<<counter<<endl;                   //输出次数至屏幕
      if(counter>5)                          //判断循环次数是否大于 5 次
      {
      YN=0;                                  //如果循环次数超过 5 次,将逻辑变量 YN 置 0
      }
    }
   }
```

例 1-4 的执行效果如图 1-2 所示。

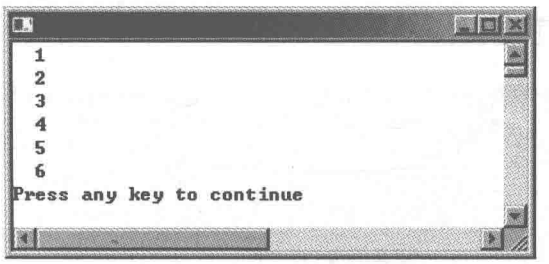



图 1-2　例 1-4 的执行效果