

## <<C++编程关键路径>>

### 图书基本信息

书名：<<C++编程关键路径>>

13位ISBN编号：9787508470825

10位ISBN编号：7508470826

出版时间：2010-1

出版时间：水利水电出版社

作者：梁永军

页数：291

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

## &lt;&lt;C++编程关键路径&gt;&gt;

## 前言

本书编写的目的就是，通过笔者十几年的C++编程经验，让C++求职者、C++初学者和C++程序员知道自己在即将或已经从事的C++实际开发工作中，需要掌握哪些知识，并期望可以通过本书更快、更好、更深入地掌握C++知识，以便在提高水平的同时，谋求一份合适的工作和更高的薪水。也许在读者看来，这本书更像一本C++求职指南。

C++知识博大精深，浩瀚如海。

一名成熟的c++程序员需要掌握的知识如同需要掌握十八般武艺一样，不可胜数。

需要掌握的知识中，基础知识有c语言知识、C++编程知识、STL编程知识；Windows编程者，还需要掌握Win32编程知识、MFC编程知识、COM编程知识、DCOM知识、COM+编程知识、ATL编程知识等；C++专业领域又有图形编程、网络通信、数据库编程、跨平台编程、游戏软件等。

从c语言基础到后面的专业领域，每一部分知识都是很大的一块知识领域，需要费心钻研。

所以，学习C++非一日、一时之功，往往需要多年的积累才有可能成为一名合格的程序员。

笔者在c++领域工作了很长一段时间，在本书中结合C++知识和自身十几年的经验，编撰成集，引导读者深入C++领域。

本书重点在于讲解在实际工作中需要掌握的C++知识，为C++程序员打好坚实的基础，书中不涉及数据库、网络通信、游戏软件等具体专业领域的讲解。

本书主要是面对C++初学者、没有C++实际开发经验的入门者和实际开发经验只有2~3年的进阶者，期望授之以真正的入门之道，一窥C++知识全景，并顺着笔者的思路逐步深入C++知识领域，掌握其精髓，成为一名合格的C++程序员。

本书对具有多年c++编程经验的老手也有很好的帮助，对他们系统地理解C++知识亦有诸多裨益。

同样一本书，初学者会有初学者的认识，进阶者会有进阶者的理解，编程多年的老手也会另有一番领悟；同样一本书，在不同层次、不同阶段，有不同的收获，这是很自然的事情。

## <<C++编程关键路径>>

### 内容概要

本书把C++知识按照实际开发工作的需要，分为C语言、C++和STL等3部分，共分10章，主要讲解C++在实际工作中需要掌握的知识、需要掌握多深，确保C++程序员在学习和工作中少走弯路，打下扎实的C++基本功。

本书适合大学计算机相关专业的学生、C++初学者、C++实际开发经验较少的程序员和实际开发经验只有2~3年的程序员。

本书是C++程序员的求职和工作指南，是大学院校学生学习C++不可多得的辅助教材。

同时，C语言部分的讲解，也是C程序员必备的教材。

总之，本书通过作者多年的编程经验，讲解了C/C++程序员在实际工作中需要掌握的知识，是C/C++程序员求职和谋求更高薪水的指南。

## &lt;&lt;C++编程关键路径&gt;&gt;

## 书籍目录

前言第1章 C语言关键知识 1.1 C语言知识 1.2 C语言的核心：函数、变量和指针 1.2.1 C语言测试程序 1.2.2 内存存储 1.2.3 程序解答 1.2.4 常量和指针 1.2.5 临时对象 1.2.6 被轻视的数据类型 1.2.7 小结 1.3 蕴藏丰富的Hello World程序 1.3.1 开始Hello World 1.3.2 预处理 1.3.3 控制台 1.3.4 调试 1.3.5 杂项第2章 关键算法和数据结构 2.1 掌握算法的前提：单链表 2.1.1 线性表 2.1.2 单链表 2.1.3 单链表创建算法 2.1.4 单链表查找算法 2.1.5 单链表删除算法 2.1.6 单链表插入算法 2.2 掌握算法的前提：循环链表 2.2.1 循环链表创建算法 2.2.2 循环链表查找算法 2.2.3 循环链表删除算法 2.2.4 循环链表插入算法 2.3 被初学者忽视被资深者推崇的：双向链表 2.3.1 双向链表创建算法 2.3.2 双向链表查找算法 2.3.3 双向链表删除算法 2.3.4 双向链表插入算法 2.3.5 双向循环链表 2.4 高手的挚爱：充分理解“树” 2.4.1 树的基本概念 2.4.2 二叉树 2.4.3 线索二叉树 2.5 几棵最重要的“树” 2.5.1 表达式树 2.5.2 二叉排序树 2.5.3 平衡二叉树 2.5.4 红黑树 2.5.5 堆 2.6 最需要掌握的排序算法和查找算法 2.6.1 插入排序 2.6.2 选择排序 2.6.3 交换排序 2.6.4 归并排序 2.6.5 需要掌握的查找算法第3章 记住C语言的常用库函数 3.1 库函数的使用 3.1.1 使用MSDN 3.1.2 熟练使用库函数 3.2 常用库函数 3.2.1 stdlib.h 3.2.2 stdio.h 3.2.3 math.h 3.2.4 ctype.h 3.2.5 string.h 3.2.6 malloc.h第4章 开始学习C++ 4.1 C++语言知识体系 4.2 预处理 4.2.1 常见的预处理 4.2.2 注意#pragma pack 4.2.3 其他需要知道的预处理宏 4.3 函数 4.3.1 声明和定义函数、函数原型 4.3.2 记住：函数声明前面的压栈声明 4.3.3 必须理解：传值调用和传址调用 4.3.4 尽量采用传址调用(传指针或传引用)来代替传值调用 4.3.5 内联(inline)函数 4.3.6 知道：函数重载 4.3.7 理解extern "C" 4.3.8 理解函数指针 4.3.9 专家的最爱：回调函数 4.3.10 namespace命名空间 4.3.11 STL的基础：函数模板第5章 C++类和继承 5.1 快速了解类的基本概念 5.1.1 类和类对象的定义 5.1.2 静态成员 5.1.3 友元 5.1.4 局部类和嵌套类 5.1.5 类模板 5.2 真正走入C++：类的构造和析构 5.2.1 new / delete和malloc / free以及定位new 5.2.2 构造函数和析构函数 5.2.3 默认构造函数 5.2.4 类的拷贝构造函数 5.2.5 在含有指针数据成员的时候构建自己的拷贝构造函数 5.2.6 类的赋值和拷贝构造函数必须同时设计，尽量用拷贝构造来代替赋值 5.2.7 类的成员初始化表 5.3 C++九鼎之器：类的继承 5.3.1 继承和派生 5.3.2 记住常用的继承关系是公有继承 5.3.3 构造函数、析构函数在继承关系中调用的优先顺序 5.3.4 多继承 5.4 C++虚函数 5.4.1 虚函数 5.4.2 静态联编和动态联编 5.4.3 记住父类的析构函数必须是：虚的析构函数 5.4.4 再谈多继承 5.4.5 纯虚函数和抽象类 5.5 高级话题：虚拟继承和虚基类第6章 走入STL 6.1 STL知识体系一览 6.2 初窥STL 6.2.1 STL简介 6.2.2 STL历史 6.2.3 STL的5个版本 6.2.4 一个小程序——需要仔细理解 6.2.5 程序剖析 6.2.6 从容器开始第7章 STL容器 7.1 用vector敲开STL的大门 7.1.1 vector概述 7.1.2 vector定义源码 7.1.3 vector成员函数 7.1.4 vector数据结构说明 7.2 双向循环链表list 7.2.1 list概述 7.2.2 list定义源码 7.2.3 list成员函数 7.2.4 list数据结构说明 7.3 慎用deque 7.3.1 deque概述 7.3.2 deque定义源码 7.3.3 deque成员函数 7.3.4 deque数据结构说明 7.4 deque的stack、queue与vector的priority\_queue 7.4.1 stack概述 7.4.2 queue概述 7.4.3 priority—queue概述 7.4.4 小结 7.5 关联容器 7.5.1 关联容器概述 7.5.2 红黑树回顾 7.5.3 红黑树定义源码 7.6 set 7.6.1 set概述 7.6.2 set定义源码 7.6.3 set成员函数 7.6.4 set数据结构 7.7 map 7.7.1 map概述 7.7.2 map定义源码 7.7.3 map成员函数 7.7.4 map数据结构 7.8 multiset和multimap 7.8.1 multiset概述 7.8.2 multimap概述第8章 迭代器iterator 8.1 最简单的vector迭代器 8.2 list迭代器 8.3 其他容器的迭代器 8.4 初窥iterator适配器 8.5 迭代器分类 8.6 智能指针第9章 配置器、函数对象和适配器 9.1 配置器allocator 9.1.1 初窥配置器 9.1.2 P.J.Plauger版本的配置器 9.1.3 SGI版本的配置器 9.2 函数对象functors / function objects(又称仿函数) 9.2.1 初窥函数对象 9.2.2 一元函数对象和二元函数对象 9.2.3 常用函数对象 9.3 适配器adapter(又称配接器) 9.3.1 容器适配器container adapter 9.3.2 迭代器适配器iterator adapter 9.3.3 函数适配器function adapter第10章 泛型算法algorithms 10.1 算法分类 10.2 算法中的迭代器 10.2.1 输入迭代器InputIterator 10.2.2 输出迭代器OutputIterator 10.2.3 前向迭代器ForwardIterator 10.2.4 双向迭代器BidirectionalIterator 10.2.5 随机访问迭代器RandomAccessIterator 10.3 常用算法



## &lt;&lt;C++编程关键路径&gt;&gt;

## 章节摘录

插图：在给出答案之前，先来了解一下C / C++中的内存存储区的情况，这很重要。不了解C / C++的内存存储区，就不可能把程序中的问题解答出来，也不可能正确地解释其中的原因。

在C / C++中，通常可以把内存理解成4个分区：栈、堆、全局 / 静态存储区和常量存储区。

(1) 栈：通常是用于那些在编译期间就能确定存储大小的变量的存储区，用于在函数作用域内创建、在离开作用域后自动销毁的变量的存储区。

通常是局部变量、函数参数等的存储区。

它的存储空间是连续的，两个紧挨着定义的局部变量，它们的存储空间也是紧挨着的。

栈的大小是有限制的，通常Visualc++编译器默认栈的大小是1M，所以不要定义`inta[1000000]`这样的超大数组。

(2) 堆：通常是用于那些在编译期间不能确定存储大小的变量的存储区，它的存储空间是不连续的，一般由`malloc`（或`new`）函数来分配内存块，并且需要用`free`（或`delete`）释放内存。

如果程序员没有释放掉，那么就会出现常说的内存泄漏问题。

需要注意的是，两个紧挨着定义的指针变量，所指向的`malloc`出来的两块内存并不一定是紧挨着的。

另外需要注意的一点是，堆的大小几乎是不受限制的，理论上每个程序最大可达4GB。

(3) 全局 / 静态存储区：和“栈”一样，通常是用于那些在编译期间就能确定存储大小的变量的存储区，但它用于的是在整个程序运行期间都可见的全局变量和静态变量。

(4) 常量存储区：和“全局 / 静态存储区”一样，通常是用于那些在编译期间就能确定存储大小的常量的存储区，并且在程序运行期间，存储区内的常量也是全局可见的。

这是一块比较特殊的存储区，它们里面存放的是常量，不允许被修改。

现在，已经了解了C / C++内存的存储情况，接下来分析程序的5个问题并给出答案。

## <<C++编程关键路径>>

### 编辑推荐

《C++编程关键路径:程序员求职指南》详细讲解了C / C++程序员在实际工作中需要掌握的知识。具有独创性，按照实际工作的需要，把C语言、C++和STL融合在一本书中进行讨论。全面介绍了C++编程中最实用、最重要、最关键的知识。确保C / C++程序员在学习和工作中少走弯路，打下扎实的C++基本功。

## <<C++编程关键路径>>

### 版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>