算法考试和考研机试

权威宝典

# ALGORITHM

# 算法笔记

胡凡　曾磊　主编

- 适合语言零基础的算法初学者
- 玩转研究生复试
  上机考试和PAT甲乙级考试
- 突破CCF的CSP认证
  或其他算法考试
- 海量例题实战解析
- 二维码内容实时互动更新
- 求职面试时的基础算法

机械工业出版社
CHINA MACHINE PRESS

# 算法笔记

胡凡　曾磊　主编

机械工业出版社

本书内容包括：C/C++快速入门、入门模拟、算法初步、数学问题、C++标准模板库（STL）、数据结构专题（二章）、搜索专题、图算法专题、动态规划专题、字符串专题、专题扩展。本书印有二维码，用来实时更新、补充内容及发布勘误的。

本书可作为计算机专业研究生入学考试复试上机、各类算法等级考试（如 PAT、CSP 等）的辅导书，也可作为“数据结构”科目的考研教材及辅导书内容的补充。本书还是学习 C 语言、数据结构与算法的入门辅导书，非常适合零基础的学习者对经典算法进行学习。

（编辑邮箱：jinacmp@163.com）

**图书在版编目（CIP）数据**

算法笔记 / 胡凡，曾磊主编. —北京：机械工业出版社，2016.7

ISBN 978-7-111-54009-0

Ⅰ. ①算… Ⅱ. ①胡… ②曾… Ⅲ. ①电子计算机—算法理论 Ⅳ. ①TP301.6

中国版本图书馆 CIP 数据核字（2016）第 129818 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：吉　玲　责任编辑：吉　玲　吴晋瑜　王小东

封面设计：鞠　杨　责任印制：李　洋　责任校对：刘怡丹

北京振兴源印务有限公司印刷

2016 年 7 月第 1 版 • 第 1 次印刷

184mm×260mm • 30.75 印张 • 782 千字

标准书号：ISBN 978-7-111-54009-0

定价：65.00 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　　网络服务

服务咨询热线：010-88361066　　机 工 官 网：www.cmpbook.com

读者购书热线：010-68326294　　机 工 官 博：weibo.com/cmp1952

010-88379203　　金　书　网：www.golden-book.com

**封面无防伪标均为盗版**　　教育服务网：www.cmpedu.com

# 前　言

最初打算写这本书是在自己刚考完研之后。那段时间，我每天都在浙江大学天勤考研群里给学弟学妹们答疑，在感受着他们的努力与进步的同时，自己仿佛又经历了一次考研，感慨颇多。渐渐地，出于兴趣，我感觉自己还能为他们做些什么，于是便萌生了写一些东西的想法。由于浙江大学机试就是 PAT 考试，因此一开始只是打算把 PAT 考试题目的题解都写一遍，但是在写作过程中慢慢发现，题解本身并不能给人带来太多的提高，而算法思想的理解和学习才是最为重要的。考虑到当时的算法入门书籍要么偏重于竞赛风格，要么偏重于面试风格，因此我便打算写一本适用于考研机试与 PAT 的算法书籍，以供考研的学弟学妹们学习。因为浙江机试的考试范围已经能覆盖大部分学校的机试范围，所以对于报考其他学校的同学也同样适用。

第一次试印的版本给当年浙江大学机试的平均分提高了十多分，反响不错。但我深知书中仍有许多不足，也有许多想要添加的内容没来得及加进去，因此便又花费了半年时间增加了许多内容。至此，本书已经覆盖了大部分基础经典算法，不仅可以作为考研机试和 PAT 的学习教材，对其他的一些算法考试（例如 CCF 的 CSP 考试）或者考研初试的数据结构科目的学习和理解也很有帮助，甚至仅仅想学习经典算法的读者也能从本书中学到许多知识。由于书中很多内容都来源于自己对算法的理解，因此最终把书名定为《算法笔记》。

本书希望让一个 C 语言零基础的读者能很好地进入本书的学习，因此在第 2 章设置了 C 语言的入门详解，使读者不必因自己不会 C 语言而有所担心，并且在对 C 语言的讲解中融入了部分 C++的特性内容，这样读者会更容易书写顺手的代码。第 3 ~ 5 章是入门部分，其中介绍了一些算法思想和数学问题，读者可从中学习到一些基础但非常重要的算法思想，并培养基本的思维能力和代码能力。第 6 章介绍了 C++标准模板库（STL）的常用内容和 algorithm 头文件下的常用函数，以帮助读者节省写代码的时间。第 7 ~ 12 章是进阶部分，其中介绍了各类经典数据结构、图算法以及较为进阶的重要算法，以使读者对经典算法和数据结构有较为深入的学习。第 13 章补充了一些上面没有介绍的内容，以帮助读者拓宽视野。

另外，书中印的二维码，是用来更新或补充书籍内容及发布本书勘误的。通过扫描本书的**勘误**和**内容更新日志**二维码，读者可以得到实时更新的相应内容。


最后，由于编者水平有限，尽管对本书进行了多次校对，书中可能仍有一些待改进的地方，敬请广大读者提出宝贵建议！

## 本书的适用范围

- 研究生复试上机考试
- PAT 甲级、乙级考试
- CCF 的 CSP 认证（或其他算法）

- 求职面试时的基础算法考试
- 考研初试数据结构科目
- 经典算法的入门学习

## 致谢

在本书写作过程中，得到了许多朋友给予的帮助，他们是鲁蕴铖、徐涵、王改革和周伟，他们在本书的内容、细节等方面给出了很多建设性的意见，在此表示衷心的感谢。

参加本书编写的人员还有：曾磊、唐晓瑜、庞志飞、冯杰、刘伟、王改革、柯扬斌、何世伟、朱逸晨、林炀平、杨晓海、庞博、张也、刘阳、吴联坤、于志超、朱清华、陈鸿翔、柴一平、李幸超、李邦鹏、范旭民、李疆、胡学军、厉月艳、朱华、鲁蕴铖、徐涵、王巨峰、金明健、刘欧、田唐昊。

感谢维护 PAT 的浙江大学陈越老师、维护 Codeup 的浙江传媒学院张浩斌老师，他们耐心回复了我关于 PAT 和 Codeup 的使用问题，使我能够更好地使用上面的题目作为例题和练习题。

感谢本书最初试印版本的读者，他们发现了书中的许多错误，并就本书的内容提了许多建议，使得本书更为完善。还有很多朋友对本书的写作十分关心，在他们的鼓励下，我才能在巨大的学业压力中最终完成本书，在此一并表示感谢。

感谢书链团队为本书提供的二维码与资源管理系统，它让本书成为一本可以动态添加内容的书籍，增强了本书的可扩展性。

最后，还要特别感谢机械工业出版社的吉玲编辑，在她的鼓励和帮助下，我顺利完成了本书的编写，并把更好的内容展现给读者。

胡凡

# 目　录

# 第 1 章　如何使用本书

## 1.1　本书的基本内容

本书旨在让一个 C 语言零基础算法的学习者循序渐进地学习经典算法，因此在第 2 章对 C 语言的语法进行了详细的入门讲解，并在其中融入了部分 C++的特性，以便读者能够更容易地书写代码。

第 3 ~ 5 章是入门部分。其中第 3 章将初步训练读者最基本的编写代码能力，内容比较少，建议读者用较少的时间完成；第 4 章对常用的基本算法思想进行介绍，内容非常重要，建议读者多花一些时间仔细思考和训练；第 5 章是一些数学问题，其中 5.7 节的内容和 5.8 节的后半部分内容相对没有那么容易，读者可以选择性阅读。

第 6 章介绍了 C++标准模板库（STL）中的常用容器和 algorithm 头文件下的常用函数，通过学习本章，读者可以节省许多写代码的时间，把注意力更多地放在解决问题上。需要说明的是，此部分内容不难，读者不必对难度有所担心，但还应认真实现其中给出的实例。另外，部分内容可能会在前几章用到，因此，如果在前几章中需要用到本章的内容（需要使用时书中会给出说明），那么请在本章来阅读相关内容，然后再回头去继续学习。

第 7 ~ 12 章是进阶部分。其中第 7 章介绍了栈、队列和链表，第 8 章介绍了深度优先搜索和广度优先搜索，它们是树和图算法学习的基础，需要读者认真学习并掌握；第 9 章和第 10 章分别讲解了树和图的相关算法，它们是数据结构中非常重要的内容，在很多考试中也经常会出现，需要读者特别关注；第 11 章介绍了动态规划算法的几个经典模型，并进行了相应的总结；第 12 章对字符串 hash 和 KMP 算法进行了探讨。

第 13 章是在前面章节的基础上额外增加的内容，需要读者花一些时间来阅读并掌握。

根据不同的需要，读者可以不同的方式来学习本书。就研究生复试上机来说，不同的学校对机试的要求不同，因此需要根据报考学校的机考大纲来确定需要学习哪些内容。本书覆盖了大部分学校的机试内容，对于 PAT 乙级考试，前 7 章内容已经基本够用；对于 PAT 甲级考试，本书的大部分内容都要掌握（冷门考点会在“本章二维码”中给出）；对于 CCF 的 CSP 认证，本书能覆盖竞赛内容以外的考点（最后一题偶尔会涉及 ACM-ICPC 的内容，超出了本书范围）；对于考研初试数据结构科目，本书能帮助读者更好地理解各种数据结构与算法。

另外，本书还有一本配套习题集——《算法笔记上机训练实战指南》，书中给出了 PAT 乙级前 50 题、甲级前 107 题的详细题解。若读者想要对知识点的熟练度有进一步的提升，推荐同时使用本书的配套习题集（最新的考题会在本书的二维码中添加更新）。习题集的题目顺序是按本书的章节顺序编排的，并配有十分详细的题解，以便读者更有效地进行针对性训练。**推荐使用“阅读一节本书的内容，然后做一节习题集对应小节的题目”的训练方式。**

## 1.2　如何选择编程语言和编译器

很多考试都会限定程序的运行时间的上限，因此选择尽可能快的编程语言是非常重要的。

一般来说，可供选择的语言有C、C++、Java等，但是Java的执行比较慢，因此较常使用的是C或者C++。考虑到C++的语法向下兼容C，并且C的输入输出语句比C++的要快很多，因此我们可以在主体上使用C语言的语法；而C++中有一些特性和功能非常好用（例如变量可以随时定义、拥有标准模板库STL等），因此在一定程度上我们可以混用部分C++的语法（事实上，由于C++向下兼容C，因此一般都是在C++中写C语言的语法，相关内容参见第2章）。

编译器的选择则因人而异、因现场环境而异。不同的考试可能提供不同的编译器，要根据具体情况来选择。但一般来说，可能出现的编译器有VC 6.0、VS系列、Dev-C++、C-Free、Code::Blocks、Eclipse等，其中VC 6.0因为标准过于古老，很多语法在其中没办法通过编译，所以尽量不要使用；Dev-C++、C-Free、Code::Blocks则是轻便好用的编译器，推荐使用，可以根据具体情况来选择；VS系列是较为厚重的编译器，在没有其他轻便编译器可供选择的情况下使用；Eclipse则更常用于Java代码的编写。

## 1.3 在线评测系统

在各类考试中，判断程序写得对不对，一般需要借助在线评测系统（Online Judge，OJ）。一般来说，在OJ上可以看到题目的题目描述、输入格式、输出格式、样例输入及样例输出。我们需要根据题目来写出相应的代码，然后提交给OJ进行评测。OJ的后台会让程序运行很多组数据，并根据程序输出结果的正确与否返回不同的结果（具体的结果在1.4节中讲述）。**注意：即便代码能通过样例，也不能说明是完全正确的，因为后台会有很多组数据，样例只是一个示范而已。**

本书的例题与练习题将来自PAT与codeup，下面对其分别介绍。

**（1）PAT**

PAT的全称为Programming Ability Test，是考察计算机程序设计能力的一个考试，目前分为乙级（Basic）、甲级（Advanced）和顶级（Top）三个难度层次（需要选择其中一个报名），难度依次递增，其中顶级将涉及大量ACM-ICPC竞赛的考点，报考的人数也相对较少。本书能够覆盖乙级和甲级的考试知识点，并且书中有很多例题都来自PAT甲、乙级的真题。

为便于区别，本书中来自乙级的题目将以B开头，例如B1003表示乙级的题号为1003的题目。PAT乙级真题题库地址如下：

http://www.patest.cn/contests/pat-b-practise

同样的，本书中来自甲级的题目将以A开头，例如A1066表示甲级的题号为1066的题目。PAT甲级真题题库地址如下：

http://www.patest.cn/contests/pat-a-practise

对于来自PAT甲、乙级的题目，只需要在网站上面注册一个账号即可提交，例如对B1001来说，在阅读完题目并写出相应的代码后，只需要单击最下方的“提交代码”，接着选择代码对应的语言（例如C或者C++代码的提交都可以选择“C++ (g++ 4.7.2)”这一项），然后把代码粘贴在编辑框内，单击“提交代码”即可，稍等片刻刷新页面即可得到程序评测的结果。注意：**PAT的评测方式为“单点测试”**，即代码只需要能够处理一组数据的输入即可，后台会多次运行代码来测试不同的数据，然后对每组数据都返回相应的结果。**“单点测试”的具体写法见2.10.1节。**

**（2）codeup**

codeup 是一个有着很多题目的题库，当然它也是一个在线评测系统，本书的部分例题和练习题将来自于 codeup，其地址如下：

http://www.codeup.cn/

在注册一个账号之后，单击最上面一栏中的“题目集”即可进入题库，之后可以根据题目标题的关键字或者题号本身来搜索题目。当然，它对题目进行了归类，读者也可以直接从分类中选择自己想要训练的算法类型。

对 codeup 的题目而言，页面中同样有着题目描述、输入格式、输出格式、样例输入和样例输出。一旦写好了代码，只需要单击最下方的“提交”，然后选择代码对应的语言（例如 C 和 C++只需要一律选择 C++即可），接着在编辑框内粘贴代码，单击“提交”，过几秒后刷新页面即可得到程序评测的结果。注意：**codeup 的评测方式为“多点测试”（除了第 2 章的 C 语言练习题外）**，即代码需要能够处理所有数据的输入，也就是说，后台只会运行代码一次来测试不同的数据，只有当所有数据都输出正确的结果时才会让程序通过，只要有一组数据错误，就会返回对应的错误类型。**“多点测试”的具体写法见 2.10.2**。

除了 PAT 和 codeup 之外，还有很多优秀的在线评测系统，但是大多数都是侧重于竞赛的，因此如果不是算法竞赛的话，一般不需要去它们上面做题。下面是几个较知名 OJ 的地址：

POJ：http://poj.org/

HDOJ：http://acm.hdu.edu.cn/

ZOJ：http://acm.zju.edu.cn/

CodeForces：http://codeforces.com/

UVa：uva.onlinejudge.org

ACdream：http://acdream.info/

## 1.4　常见的评测结果

**（1）答案正确（Accepted，AC）**

恭喜你！所提交的代码通过了数据！这个评测结果应该是大家最喜欢见到的，也非常好理解。如果是单点测试，那么每通过一组数据，就会返回一个 Accepted；如果是多点测试，那么只有当通过了所有数据时，才会返回 Accepted。

**（2）编译错误（Compile Error，CE）**

很显然，如果代码没有办法通过编译，那么就会返回 Compile Error。这时要先注意是不是选错了语言，然后再看本地的编译器能不能编译通过刚刚提交的代码，修改之后再次提交即可。

**（3）答案错误（Wrong Answer，WA）**

“答案错误”是比较令人懊恼的结果，因为这说明代码有漏洞或者算法根本就是错误的，只是恰好能过样例而已。不过有时可能是因为输出了一些调试信息导致的，那就删掉多余的输出内容再输出。当然，大部分情况下都需要认真检查代码的逻辑有没有问题。

**（4）运行超时（Time Limit Exceeded，TLE）**

由于每道题都会规定程序运行时间的上限，因此当超过这个限制时就会返回 TLE。一般来说，这一结果可能是由算法的时间复杂度过大而导致的，当然也可能是某组数据使得代码

中某处地方死循环了。因此，要仔细思考最坏时间复杂度是多少，或者检查代码中是否可能出现特殊数据死循环的情况。

**（5）运行错误（Runtime Error，RE）**

这一结果的可能性非常多，常见的有段错误（直接的原因是非法访问了内存，例如数组越界、指针乱指）、浮点错误（例如除数为 0、模数为 0）、递归爆栈（一般由递归时层数过深导致的）等。一般来说，需要先检查数组大小是否比题目的数据范围大，然后再去检查可不可能有特殊数据可以使除数或模数为 0，有递归的情况则检查是否在大数据时递归层数太深。

**（6）内存超限（Memory Limit Exceeded，MLE）**

每道题目都会有规定程序使用的空间上限，因此如果程序中使用太多的空间，则会返回 MLE，例如数组太大一般最容易导致这个结果。

**（7）格式错误（Presentation Error，PE）**

这应该是最接近 Accepted 的错误了，基本是由多输出了空格或者换行导致的，稍作修改即可。

**（8）输出超限（Output Limit Exceeded，OLE）**

如果程序输出了过量的内容（一般是指过量非常多），那么就会返回 OLE。一般是由输出了大量的调试信息或者特殊数据导致的死循环输出导致的。

## 1.5 如何高效地做题

一般来说，按照算法专题进行集中性的题目训练是算法学习的较好方法，因为这可以一次性地对某个算法有一个较为深入且细致的训练，而随意乱做或是按题号从小到大地“刷”题目并不是一个很好的选择，也无法形成完整的知识体系。

如果在做一道题时暂时没有想法，那么可以先放着，跳过去做其他题目，过一段时间再回来重新做，或许就柳暗花明了（例如你可以设置一个未解决题目的队列，每次有题目暂时不会做时就扔到队列里，然后隔三差五取出里面的题目再想一下，想不出来就再扔到队列里……）。当然，如果题目本身较难，做了很久也没有想法，也可以查看题解，知道做法之后再自己独立完成代码过程。

另外，在做题时也可以适当总结相似题目的解题方法，这也是进行专题训练时可以顺带完成的事，可起到事半功倍的效果。

本章二维码

# 第2章　C/C++快速入门

考虑到一些参加考研机试或其他算法考试的读者并没有学过C语言或者C++，而在机考中最适合使用的就是这两个语言，因此编写了C/C++入门的部分。该部分旨在让没有接触过C语言的读者能够快速上手编写C程序，因此对一些不影响实际编程的语法点采取了舍去的策略。如果读者备考的时间比较短且又没有学习过C语言，那么建议只要把本书提到的语法点掌握即可；如果读者想要系统、完整地学习C语言，则建议参考那些专门讲解C语言的书籍。

虽然是C/C++的入门，但是本书主要侧重于C语言的讲解，这是因为C++中大部分语法在机考中都是用不到的，并且C++向下兼容C，因此读者可以用C++中一些很好用的特性来取代C语言中那些不太顺手的设定。

如果读者确实从未接触过编程语言，那么最好把下面接触到的程序和语句都自己尝试编写并运行一下，而不是仅仅只是“看过看懂”的状态，因为在实际编程之前，任何“觉得自己会了”的想法都不过是“纸上谈兵”。

如果读者已经学习过C语言或者C++，那么可以选择跳过这部分，但是最好可以大致浏览一下，看看有没有以前没有确切接触过的细节，例如“引用”、int型范围、指针的错误写法、结构体的构造函数、浮点型的比较等。

**注意**：有些读者认为学过C++之后就没有必要学C语言，甚至觉得C语言太麻烦而不学，这是不太正确的。因为就机考使用的语法而言，除了输入和输出部分，其余顺序结构、分支结构、循环结构、数组、指针都是几乎一样的，学习C语言并不会带来什么负担。对于让C++使用者觉得麻烦的scanf函数和printf函数，虽然必须承认cin和cout可以不指定输入输出格式比较方便，但是cin和cout消耗的时间比scanf和printf多得多，很多题目可能输入还没结束就超时了。当然，读者可以在某次使用cin和cout超时，改成scanf和printf后通过的时候，痛下决心以后使用scanf和printf。顺便指出，请不要同时在一个程序中使用cout和printf，有时候会出问题。

最后再次强调，本书会使用一些代码作为举例，希望读者能够亲自照着输入并理解一下，这将对语言的学习大有帮助。如果代码中出现某些暂时还没有提到的语法，不妨只要先知道是什么意思，具体细节可以等后面提到再学。

下面开始介绍C语言的相关内容。

先来看一段C语言小程序：

```
#include <stdio.h>
int main(){
    int a, b;
    scanf("%d%d", &a, &b);
    printf("%d", a+b);
    return 0;
}
```

请读者在编译器中输入这段代码，并将其保存为.cpp 文件（C 语言的文件扩展名为.c，但是为了使用 C++中的一些好用的特性，请把文件扩展名改为 C++的文件扩展名.cpp）。

这个程序分为两个部分：**头文件**和**主函数**。

**1. 头文件**

在上面的代码中，#include <stdio.h>这一行就是头文件。其中，stdio.h 是标准输入输出库，如果在程序中需要输入输出，就需要加上这个头文件。不过一般来说程序都是需要输入输出的，所以基本上每一个 C 程序都需要加上头文件。

stdio 的全称是 standard input output，h 就是 head 的缩写，.h 是头文件的文件格式。我们可以这样理解：stdio.h 就是一个文件，这个文件中包含了一些跟输入输出有关的东西，如果程序需要输入输出，就要通过#include <×××>的写法来包含（include）这个头文件，这样才可以使用 stdio.h 这个文件里的输入输出函数。

既然 stdio.h 是负责输入输出，那么自然还会有负责其他功能的头文件。例如，math.h 负责一些数学函数，string.h 负责跟字符串有关的函数，则只需要在需要使用对应的函数时，将它们的头文件包含到这个程序中来即可。

此外，在 C++的标准中，stdio.h 更推荐使用等价写法：cstdio，也就是在前面加一个 c，然后去掉.h 即可。所以#include <stdio.h>和#include <cstdio>的写法是等价的，#include <math.h>和#include <cmath>等价，#include <string.h>和#include <cstring>也等价。读者在程序中看到这种写法应当能明白它的意思。

**2. 主函数**

```
int main(){
    ...
    return 0;
}
```

上面的代码就是主函数。**主函数是一个程序的入口位置**，整个程序从主函数开始执行。**一个程序最多只能有一个主函数。**

下面来看一下省略号中的内容，读者暂时只需要大致了解每个语句的作用，因为会在后面仔细讲解这些语法。

```
int a, b;
```

这句话定义了两个变量 a 和 b，类型是 int 型（简单来说就是整数）。

```
scanf("%d%d", &a, &b);
```

scanf 用来读入数据，这条语句以%d 的格式输入 a 和 b，其中%d 就是 int 型的输入输出标识。简单来说，就是把 a 和 b 作为整数输入。

```
printf("%d", a + b);
```

printf 用来输出数据，这条语句计算 a + b 并以%d 格式输出。上面说过，%d 就是 int 型的输入输出标识，所以就是把 a + b 作为整数输出。因此这段代码的主函数实现了输入两个数 a 和 b 然后输出 a+b 的功能。

接下来进入正题，讲解一下 C 语言中各个需要使用的语法。

**声明：下文使用的代码请保存成.cpp 文件（即 C++文件），然后选择 C++语言（或 C++）进行提交。**由于 C++向下兼容 C，因此采用这种方式可以尽可能防止一些因 C 与 C++之间的区分而导致的编译错误。