算法艺术与信息学竞赛

- 适合语言零基础的初学者
- 涵盖算法竞赛的主要知识点
- 大量经验教训与比赛技巧
- 简洁、清晰、高效的示例代码
- 丰富的辅助教学资源与配套习题

# 算法竞赛入门经典

◎ 刘汝佳　编著

清华大学出版社

刘汝佳，1982年12月生，高中毕业于重庆市外国语学校。

2000年3月获得NOI2000全国青少年信息学奥林匹克竞赛一等奖第四名，进入国家集训队，并因此保送到清华大学计算机科学与技术系。大一时获2001年ACM/ICPC国际大学生程序设计竞赛亚洲–上海赛区冠军和2002年世界总决赛银牌（世界第四），2005年获学士学位，2008年获硕士学位。

学生时代曾为中国计算机学会NOI科学委员会学生委员，担任IOI2002–2008中国国家队教练，并为NOI系列比赛命题十余道。现为NOI竞赛委员会委员，并在NOI 25周年时获得中国计算机学会颁发的“特别贡献奖”。

2004年至今共为ACM/ICPC亚洲赛区命题二十余道，担任6次裁判和2次命题总监，并应邀参加IOI和ACM/ICPC相关国际研讨会，发表论文两篇。

2004年初作为第一作者出版专著《算法艺术与信息学竞赛》，2009年出版译著《编程挑战》。

多年来在全国二十余个城市进行中学生竞赛培训工作，为北京、上海、吉林省等地的著名高校授课与宣讲，并多次与TopCoder、百度和网易有道等知名企业合作举办比赛，让更多的IT人才获得展示自我的平台。

算法艺术与信息学竞赛

# 算法竞赛入门经典

刘汝佳　编著

清华大学出版社

北　京

## 内 容 简 介

本书是一本算法竞赛的入门教材，把 C/C++语言、算法和解题有机地结合在了一起，淡化理论，注重学习方法和实践技巧。全书内容分为 11 章，包括程序设计入门、循环结构程序设计、数组和字符串、函数和递归、基础题目选解、数据结构基础、暴力求解法、高效算法设计、动态规划初步、数学概念与方法、图论模型与算法，覆盖了算法竞赛入门所需的主要知识点，并附有大量习题。书中的代码规范、简洁、易懂，不仅能帮助读者理解算法原理，还能教会读者很多实用的编程技巧。另外，书中包含的各种开发、测试和调试技巧也是在传统的语言、算法类书籍中难以见到的。

本书可作为全国青少年信息学奥林匹克联赛（NOIP）的复赛教材及 ACM 国际大学生程序设计竞赛（ACM/ICPC）的入门参考，还可作为 IT 工程师与科研人员的参考用书。



图书在版编目（CIP）数据

算法竞赛入门经典/刘汝佳编著．—北京：清华大学出版社，2009.11
（算法艺术与信息学竞赛）

ISBN 978-7-302-20608-8

I. 算…　II. 刘…　III. ①电子计算机-算法理论-教材 ②C 语言-程序设计-教材　IV. TP301.6　TP312

中国版本图书馆 CIP 数据核字（2009）第 118407 号

责任编辑：朱英彪　郭　伟
封面设计：张　岩
版式设计：王世情
责任校对：王　云
责任印制：

出版发行：清华大学出版社　　地　　址：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn

印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185×260　印　　张：15　字　　数：347 千字
（附光盘 1 张）
版　　次：2009 年 11 月第 1 版　印　　次：2009 年 11 月第 1 次印刷
印　　数：1～5000
定　　价：24.00 元

# 前　言

“听说你最近在写一本关于算法竞赛入门的书？”朋友问我。
“是的。”我微笑道。
“这是怎样的一本书呢？”朋友很好奇。
“C 语言、算法和题解。”我回答。
“什么？几样东西混着吗？”朋友很吃惊。
“对。”我笑了，“这是我思考许久后做出的决定。”

## 大学之前的我

12 年前，当我翻开 Sam A.Abolrous 所著《C 语言三日通》的第一页时，我不会想到自己会有机会编写一本讲解 C 语言的书籍。当时，我真的只花了 3 天就学完了这本书，并且自信满满：“我学会 C 语言啦！我要用它写出各种有趣、有用的程序！”但渐渐地，我认识到了：虽然浅显易懂，但书中的内容只是语言入门，离实际应用还有较大差距，就好比**小学生学会造句以后还要下很大功夫才能写出像样的作文**。

第二本对我影响很大的书是 Sun 公司的 Peter van der Linden（PvdL）所著的《C 程序设计奥秘》。作者称该书应该是每一个程序员“在 C 语言方面的第二本书”，因为“书中绝大部分内容、技巧和技术在其他任何书中都找不到”。原先我只是把自己当成是程序员，但在阅读的过程中，我开始渐渐了解到硬件设计者、编译程序开发者、操作系统编写者和标准制定者是怎么想的。继续的阅读增强了我的领悟：**要学好 C 语言，绝非熟悉语法和语义这么简单**。

后来，我自学了数据结构，懂得了编程处理数据的基本原则和方法，然后又学习了 8086 汇编语言，甚至曾没日没夜地用 SoftICE 调试《仙剑奇侠传》，并把学到的技巧运用到自己开发的游戏引擎中。再后来，我通过《电脑爱好者》杂志上的一则不起眼的广告了解到了全国信息学奥林匹克联赛（当时称为分区联赛，NOIP 是后来的称谓）。“学了这么久编程，要不参加个比赛试试？”想到这里，我拉着学校里另外一个自学编程的同学，找老师带我们参加了 1997 年的联赛——在这之前，学校并不知道有这个比赛。凭借自己的数学功底和对计算机的认识，我在初赛（笔试）中获得全市第二的成绩，进入了复赛（上机）。可我的上机编程比赛的结果是“惨烈”的：第一题有一个测试点超过了整数的表示范围；第二题看漏了一个条件，一分都没得；第三题使用了穷举法，全部超时。考完之后我原以为能得满分的，结果却只得了 100 分中的 20 多分，名落孙山。

痛定思痛，我开始反思这个比赛。一个偶然的机会，我拿到了一本联赛培训教材。书上说，比赛的核心是算法（Algorithm），并且推荐使用 Pascal 语言，因为它适合描述算法。我从别人那里复制来了 Turbo Pascal 7.0（那时网络并不发达），开始研究起来。由于先学

的是 C 语言，所以我刚开始学习 Pascal 时感到有些不习惯：赋值不是“=”而是“:=”，简洁的花括号变成了累赘的 begin 和 end，if 之后要加个 then，而且和 else 之间不允许写分号……但很快我就发现，这些都不是本质问题。在编写竞赛题的程序时，我并不会用到太多的高级语法。Pascal 的语法虽然稍微啰嗦一点，但总体来说是很清晰的。就这样，我只花了不到一天时间就把语法习惯从 C 转到了 Pascal，剩下的知识就是在不断编程中慢慢地学习和熟练——**学习 C 语言的过程是痛苦的，但收益也是巨大的，“轻松转到 Pascal”只是其中一个小小的例子**。

我学习计算机，从一开始就不是为了参加竞赛，因此，在编写算法程序之余，我几乎总是使用熟悉的 C 语言，有时还会用点汇编，并没有觉得有何不妥。随着编写应用程序的经验逐渐丰富，我开始庆幸自己先学的是 C 语言——在我购买的各类技术书籍中，几乎全部使用的是 C 语言而不是 Pascal 语言，尽管偶尔有用 Delphi 的文章，但这种语言似乎除了构建漂亮的界面比较方便之外，并没有太多的“技术含量”。**我始终保持着对 C 语言的熟悉，而事实证明这对我的职业生涯发挥了巨大的作用**。

## 中学竞赛和教学

在大学里参加完 ACM/ICPC 世界总决赛之后（当时 ACM/ICPC 还可以用 Pascal，现在已经不能用了），我再也没有用 Pascal 语言做过一件“正经事”（只是偶尔用它给一些只懂 Pascal 的孩子讲课）。后来我才知道，国际信息学奥林匹克系列竞赛是为数不多的几个允许使用 Pascal 语言的比赛之一。IT 公司举办的商业比赛往往只允许用 C/C++或 Java、C#、Python 等该公司使用较为频繁的语言（顺便说一句，C 语言学好以后，读者便有了坚实的基础去学习上述其他语言），而在做一些以算法为核心的项目时，一般来说也不能用 Pascal 语言——你的算法程序必须和已有的系统集成，而这个“现有系统”很少是用 Pascal 写成的。为什么还有那么多中学生非要用这个“以后几乎再也用不着”的语言呢？

于是，我开始在中学竞赛中推广 C 语言。这并不是说我希望废除 Pascal 语言（事实上，我希望保留它），而是希望学生多一个选择，毕竟并不是每个参加信息学竞赛的学生都将走入 IT 界。但如果简单地因为“C 语言难学难用，竞赛中还容易碰到诸多问题”就放弃学习 C 语言，我想是很遗憾的。

然而，推广的道路是曲折的。作为五大学科竞赛（数学、物理、化学、生物、信息学）中唯一一门高考中没有的“特殊竞赛”，学生、教师、家长所走的道路要比其他竞赛要艰辛得多。

第一，数理化竞赛中所学的知识，多是大学本科要学习的，只不过是提前灌输给高中生，但信息学竞赛中所涉及的很多东西甚至连本科都不会学到，即使学到了，也只是“简单了解即可”，和“满足竞赛的要求”有着天壤之别，这极大地增加了中学生学习算法和编程的难度。

第二，学科发展速度快。辅导信息学竞赛的教师常常有这样的感觉：必须不停地学习学习再学习，否则很容易跟不上“潮流”。事实上，学术上的研究成果常常在短短几年之内就体现在竞赛中。

第三，质量要求高。想法再伟大，如果无法在比赛时间之内把它变成实际可运行的程序，那么所有的心血都将是白费。数学竞赛中有可能在比赛结束前 15 分钟找到突破口并在交卷前一瞬间把解法写完——就算有漏洞，还有部分分数呢；但在信息学竞赛中，想到正确解法却 5 个小时都写不完程序的现象并不罕见。连程序都写不完当然就是 0 分，即使程序写完了，如果存在关键漏洞，往往还是 0 分。这不难理解——如果用这个程序控制人造卫星发射，难道当卫星爆炸之后你还可以向人炫耀说：“除了有一个加号被我粗心写成减号从而引起爆炸之外，这个卫星的发射程序几乎是完美的。”

在这样的情况下，让学生和教师放弃自己熟悉的 Pascal 语言，转向既难学又容易出错的 C 语言确实难为他们了——尤其是在 C 语言资料如此缺乏的情况下。等一下！C 语言资料缺乏？难道市面上不是遍地都是 C 语言教材吗？对，C 语言教材多，但和算法竞赛相结合的却几乎没有。**不要以为语言入门以后就能轻易地写出算法程序**（这甚至是很多 IT 工程师的误区），多数初学者都需要详细的代码才能透彻地理解算法，只了解算法原理和步骤是远远不够的。

大家都知道，编程需要大量的练习，只看只听是不够的。反过来，如果只是盲目练习，不看不听也是不明智的。本书的目标很明确——提供算法竞赛入门所必需的一切“看”的蓝本。有效的“听”要靠教师的辛勤劳动，而有效的“练”则要靠学生自己。当然，就算是最简单的“看”，也是大有学问的。不同的读者，往往能看到不同的深度。请把本书理解为“蓝本”。没有一本教材能不加修改而适用于各种年龄层次、不同学习习惯和悟性的学生，本书也不例外。我喜欢以人为本，因材施教，不推荐按照本书的内容和顺序填鸭式地教给学生。

## 内容安排

前面花了大量篇幅讨论了语言，但语言毕竟只是算法竞赛的工具——尽管这个工具非常重要，却不是核心。正如前面所讲，算法竞赛的核心是算法。我曾考虑过把 C 语言和算法分开讲解，一本书讲语言，另一本书讲基础算法。但后来我发现，其实二者难以分开。

首先，语言部分的内容选择很难。如果把 C 语言的方方面面全部讲到，篇幅肯定不短，而且和市面上已有的 C 语言教材基本上不存在区别；如果只是提纲挈领地讲解核心语法，并只举一些最为初级的例子，看完后读者将会处于我当初 3 天看完《C 语言三日通》后的状态——以为自己都懂了，慢慢才发现自己学的都是“玩具”，真正关键、实用的东西全都不懂。

其次，算法的实现常常要求程序员对语言熟练掌握，而算法书往往对程序实现避而不谈。即使少数书籍给出了详细代码，但代码往往十分冗长，不适合用在算法竞赛中。更重要的是，这些书籍对算法实现中的小技巧和常见错误少有涉及，所有的经验教训都需要读者自己从头积累。换句话说，传统的语言书和算法之间存在着不小的鸿沟。

基于上述问题，本书采取一种语言和算法结合的方法，把内容分为如下 3 部分：

- 第 1 部分是语言篇（第 1～4 章），纯粹介绍语言，几乎不涉及算法，但逐步引入一些工程性的东西，如测试、断言、伪代码和迭代开发等。

- 第 2 部分是算法篇（第 5～8 章），在介绍算法的同时继续强化语言，补充了第 1 部分没有涉及的语言特性，如位运算、动态内存管理等，并延续第一部分的风格，在需要时引入更多的思想和技巧。学习完前两部分的读者应当可以完成相当数量的练习题。
- 第 3 部分是竞赛篇（第 9～11 章），涉及竞赛中常用的其他知识点和技巧。和前两部分相比，第 3 部分涉及的内容更加广泛，其中还包括一些难以理解的“学术内容”，但其实这些才是算法的精髓。

本书最后有一个附录，介绍开发环境和开发方法，虽然它们和语言、算法的关系都不大，却往往能极大地影响选手的成绩。另外，本书讲解过程中所涉及的程序源代码可登录网站 http://www.tup.tsinghua.edu.cn/下载。

## 致谢

在真正动笔之前，我邀请了一些对本书有兴趣的朋友一起探讨本书的框架和内容，并请他们撰写了一定数量的文字，他们是赖笠源（语言技巧、字符串）、曹正（数学）、邓凯宁（递归、状态空间搜索）、汪堃（数据结构基础）、王文一（算法设计）、胡昊（动态规划）。尽管这些文字本身并没有在最终的书稿中出现，但我从他们的努力中获得了很多启发。北京大学的杨斐瞳完成了本书中大部分插图的绘制，清华大学的杨锐和林芝恒对本书进行了文字校对、题目整理等工作，在此一并感谢。

在本书构思和初稿写作阶段，很多在一线教学的老师给我提出了有益的意见和建议，他们是绵阳南山中学的叶诗富老师、绵阳中学的曾贵胜老师、成都七中的张君亮老师、成都石室中学的文仲友老师、成都大弯中学的李植武老师、温州中学的舒春平老师，以及我的母校——重庆外国语学校的官兵老师等。

本书的习题主要来自 UVa 在线评测系统，感谢 Miguel Revilla 教授和 Carlos M. Casas Cuadrado 的大力支持。

最后，要特别感谢清华大学出版社的朱英彪编辑，与他的合作非常轻松、愉快。没有他的建议和鼓励，或许我无法鼓起勇气把“算法艺术与信息学竞赛系列”以丛书的全新面貌展现给读者。

刘汝佳

# 目　　录

## 第1部分　语　言　篇

# 第1部分　语　言　篇

## 第1章　程序设计入门

### 学习目标

- ☑ 熟悉C语言程序的编译和运行
- ☑ 学会编程计算并输出常见的算术表达式的结果
- ☑ 掌握整数和浮点数的含义和输出方法
- ☑ 掌握数学函数的使用方法
- ☑ 初步了解变量的含义
- ☑ 掌握整数和浮点数变量的声明方法
- ☑ 掌握整数和浮点数变量的读入方法
- ☑ 掌握变量交换的三变量法
- ☑ 理解算法竞赛中的程序三步曲：输入、计算、输出
- ☑ 记住算法竞赛的目标及其对程序的要求

计算机速度快，很适合做计算和逻辑判断工作。本章首先介绍顺序结构程序设计，其基本思路是：把需要计算机完成的工作分成若干个步骤，然后依次让计算机执行。注意这里的“依次”二字——步骤之间是有先后顺序的。这部分的重点在于计算。

接下来介绍分支结构程序设计，用到了逻辑判断，根据不同情况执行不同语句。本章内容不复杂，但是不容忽视。

注意：编程不是看会的，也不是听会的，而是练会的，所以应尽量在计算机旁阅读本书，以便把书中的程序输入到计算机中进行调试，顺便再做做上机练习。千万不要图快——如果没有足够的时间用来实践，那么学得快，忘得也快。

### 1.1　算术表达式

计算机的“本职”工作是计算，因此下面先从算术运算入手，看看如何用计算机进行复杂的计算。

**程序1-1　计算并输出1+2的值**

```
#include<stdio.h>
int main()
{
```

```
  printf("%d\n", 1+2);
  return 0;
}
```

这是一段简单的程序，用于计算 1+2 的值，并把结果输出到屏幕。如果你不知道如何编译并运行这段程序，可阅读附录或向指导教师求助。

即使你不明白上述程序除了“1+2”之外的其他内容，仍然可以进行以下探索：试着把“1+2”改成其他东西，而不要去修改那些并不明白的代码——它们看上去工作情况良好。

下面让我们做 4 个实验：

**实验 1**：修改程序 1-1，输出 3−4 的结果

**实验 2**：修改程序 1-1，输出 5×6 的结果

**实验 3**：修改程序 1-1，输出 8÷4 的结果

**实验 4**：修改程序 1-1，输出 8÷5 的结果

直接把“1+2”替换成“3+4”即可顺利解决实验 1，但读者很快就会发现：无法在键盘上找到乘号和除号。解决方法是：用星号“*”代替乘号，而用正斜线“/”代替除号。这样，4 个实验都顺利完成了。

等一下！实验 4 的输出结果居然是 1，而不是正确答案 1.6。这是怎么回事？计算机出问题了吗？计算机没有出问题，问题出在程序上：这段程序的实际含义并非和我们所想的一致。

在 C 语言中，8/5 的确切含义是 8 除以 5 所得商值的整数部分。同样地，（−8）/5 的值是−1，不信可以自己试试。那么如果非要得到 8÷5=1.6 的结果怎么办？下面是完整的程序。

**程序 1-2　计算并输出 8/5 的值，保留小数点后 1 位**

```
#include<stdio.h>
int main()
{
  printf("%.1lf\n", 8.0/5.0);
  return 0;
}
```

注意，百分号后面是个小数点，然后是数字 1，再然后是小写字母 l，最后是小写字母 f，千万不能打错，包括大小写——在 C 语言中，大写和小写字母代表的含义是不同的。

再来做 3 个实验：

**实验 5**：把%.1lf 中的数字 1 改成 2，结果如何？能猜想出“1”的确切意思吗？如果把小数点和 1 都删除，%lf 的含义是什么？

**实验 6**：字符串%.1lf 不变，把 8.0/5.0 改成原来的 8/5，结果如何？

**实验 7**：字符串%.1lf 改成原来的%d，8.0/5.0 不变，结果如何？

实验 5 并不难解决，但实验 6 和实验 7 的答案就很难简单解释了——真正原因涉及整数和浮点数编码，相信多数初学者对此都不感兴趣。原因并不重要，重要的是规范：根据规范做事情，则一切尽在掌握中。