程序设计思维与实践

汪云海

http://www.yunhaiwang.org/

汪云海

- Joined SDU in Dec. 2015
- Research Interests
  - Visualization
  - Machine learning
- Office hours
  - I am available a lot –in N3 403
    Come to talk with me if you have any question
- More about me ….. http://www.yunhaiwang.org/

助教：薛明亮

- 泰山学堂2013级，2017级硕士生
- 610033470@qq.com

课程目标

- 增强写程序的速度与能力

- 提高CCF CSP考试通过率

CCF计算机职业资格认证
Computing Accreditation for Professionals (CAP)

课程目标

- 促进ACM ICPC竞赛成绩

序号	排名	中文校名	实际排名	题数	时间	2016中国决赛排名
1	6	清华大学	6	9	964	1
2	7	北京大学	7	9	1245	6
3	8	复旦大学	8	8	1103	5
4	13	上海交通大学	13	7	767	2
5	13	电子科技大学	15	7	846	4
6	20	福州大学	21	6	659	15
7	20	台湾大学	25	6	747	
8	20	杭州电子科技大学	33	6	1561	8
9	34	香港中文大学	35	5	378	3
10	34	北京邮电大学	53	5	714	9
11	34	武汉大学	55	5	895	28
12	56	浙江大学	56	4	168	7
13	56	北京师范大学	61	4	249	40
14	56	上海大学	63	4	256	48
15	56	华南理工大学	66	4	273	12
16	56	北京航空航天大学	70	4	293	10
17	56	东北大学	72	4	310	
18	56	北京理工大学	74	4	317	32

2016年ACM ICPC中国高校成绩

山东大学？

高校名称	科研排名	入围次数	最好名次	最差名次	HM次数
清华大学	26	19	2	19	1
上海交通大学	81	18	1	14	1
中山大学	209	15	6	28	
复旦大学	147	14	6	39	
浙江大学	88	12	1	48	2
北京大学	84	11	3	20	
电子科技大学	609	7	13	36	
福州大学	0	6	18	39	2
北京邮电大学	409	5	14	48	1
武汉大学	611	5	18	42	2
上海大学	0	5	24	31	2
天津大学	421	5	27	60	
北京交通大学	520	5	34	47	
华南理工大学	498	4	13	60	
中国人民大学	326	4	19	60	1
浙江工业大学	0	4	20	36	2
四川大学	0	4	34	34	3
国防科技大学	246	4	36	47	1
中国科学技术大学	122	3	14	19	1
华南农业大学	0	3	19	39	1
华中科技大学	182	3	20	48	1
哈尔滨工业大学	162	3	27	27	2
北京理工大学	438	3	27	45	1
华东师范大学	529	3	36	42	1
杭州电子科技大学	0	3	49	49	2
吉林大学	783	2	19	19	1
北京航空航天大学	288	2	27	30	
西安电子科技大学	324	2	36	44	
华东理工大学	0	2	44	44	1
北京师范大学	0	2	45	45	1
山东大学	145	2	47	47	1
哈尔滨工程大学	0	2			2
东北师范大学	0	1	14	14	
东华大学	0	1	27	27	
浙江大学宁波理工学院	0	1	27	27	
东南大学	443	1	36	36	
浙江理工大学	0	1	43	43	
厦门大学	626	1	44	44	
合肥工业大学	648	1	44	44	
湖南大学	0	1	45	45	
南京大学	165	1	49	49	
南京航空航天大学	0	1			1
广东工业大学	0	1			1
浙江师范大学	0	1			1
中国科学院研究生院	63	1			1

学习目标

- Given a problem, we want to
  - solve it efficiently
  - by using algorithms and data structures,
  - convert our solution into a program,
  - do it as quickly as possible (under pressure)
  - and do it correctly (without bugs)
- This course will exercise this process

How

- Study common types of problems
- Show common applications of algorithms and data structures you already know from
- Introduce other common algorithms and data structures
- Go over some commonly used theory
- Practice problem solving
- Practice programming
- More practice
- More practice

Course Book

把ACM请下神坛

Ryen 评论 算法竞赛入门经典 ★★★★★ 2010-02-03 16:57:15

一提到ACM竞赛，周围很多同学都觉得高不可攀，感觉是数学天才的专属领域，其实我们常常被很多大牛的光环给误导了，ACM相对于中学的信息学奥赛，难度已经降低了很多，大多数题目即使一般计算机专业的同学去做也完全有能力搞定。

本书的推出再一次大大降低了ACM算法竞赛的门槛。这本书绝对担当的起"入门"和"经典"这两个词。预备知识仅仅是需要会使用C语言，当然有一定的算法基础更好。每章以典型例题的方式讲述了在竞赛中常用的各种算法，并毫不吝啬的介绍了很多在编程老手中间秘而不宣的编程技巧。

相对与前作，作者显然是吸取了读者反映的例题和习题难度太大，不易上手的缺点，这次改版分三本成系列出版，大大的平滑了学习曲线。每章的习题难度都不大。虽然这些习题在竞赛中基本都属于水题的范畴，不过如果你能独立完成大部分，算法能力完全能达到现今公司内程序员的中上水准。

刘汝佳的《算法竞赛入门经典》该怎么学？

12 人赞同了该回答

下面答案是关于第二版粉皮的

第二版比第一版不止是篇幅翻倍，更像是一本全新的书

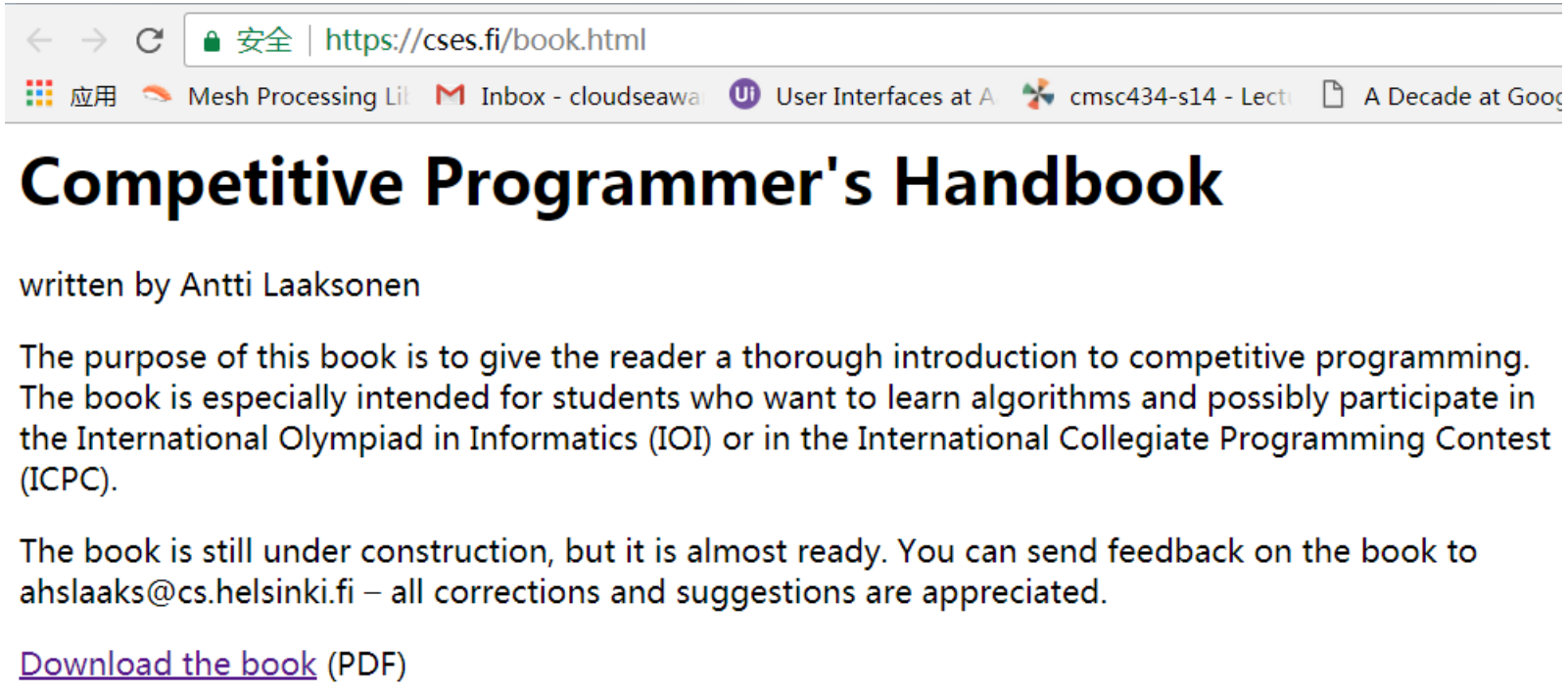
非常适合入门的书，里面的uva题目配合virtual judge就没问题（虽然偶尔还会挂但过会就好）

里面的语言篇关于c语言和stl的部分讲的比较透彻，但是还需要加上自己去调试，并学会遇到问题自行百度（不黑百度，百科或百度一般都能搜到）

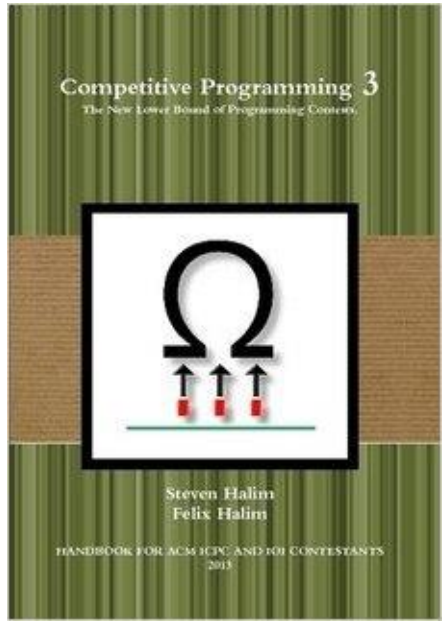
至于题目说的怎么学，那当然是自己配合vj去刷题，里面竞赛题目选讲有余力就做，多刷点题不是什么坏事，不做也没事，但是习题你肯定要做。书里说习题起码要做多少道，就按他书上的来。

不要跳读，要循序渐进，慢慢刷，你会明白这是本神书的。

Course Book


安全 | https://cses.fi/book.html
应用
Mesh Processing Lil
Inbox - cloudseawa
User Interfaces at A
cmsc434-s14 - Lect
A Decade at Goog
Competitive Programmer's Handbook
written by Antti Laaksonen
The purpose of this book is to give the reader a thorough introduction to competitive programming. The book is especially intended for students who want to learn algorithms and possibly participate in the International Olympiad in Informatics (IOI) or in the International Collegiate Programming Contest (ICPC).
The book is still under construction, but it is almost ready. You can send feedback on the book to ahslaaks@cs.helsinki.fi – all corrections and suggestions are appreciated.
Download the book (PDF)

参考书



- Competitive Programming by Steven Halim
- First edition can be downloaded from the book homepage:
  https://sites.google.com/site/stevenhalim/
- We will also loosely follow the first edition
- There's also a 2nd and 3rd edition (both should be compatible with
- our course), but they need to be ordered online

Course Schedule

- C/C++进阶（2次课）
  - 结构化程序设计
  - 数组和字符串
  - 函数和递归
  - C++与STL入门
- 数据结构和基本算法 （3次课）
  - 数据结构基础
  - 复杂度分析
  - 暴力求解算法
- 高级算法 （10次课）
  - 高效算法设计
  - 动态规划
  - 数学概念
  - 图论模型与算法
  - 难题选解

上机时间

山东大学2017-2018年度秋季学期课程清单

系所：	计算机科学与技术学院												注：时间与教室、周次之间		
课程号	课序	课程名	主讲教师	职称	上课班级	人数	周学时	授课学时	学分	上机学时	实验学时	星期三	星期四	星期五	上课教室	上课周次
sd01331680	0	程序设计思维与实践	汪云海3	副教授	计基地16学堂计机16	46	4	64	3	32	32	3-4（13-14周上机）		5-6（3-16周上机）	青岛校区振声苑E108/青岛校区振声苑E301	1-16周上/1-16周上

编译器和调试器

- GCC: gcc test.c –o test
  - Linux系统安装时选择gcc, g++
  - Windows安装MinGW
- GDB命令

附表A-2　gdb常见命令

简写	全称	备注
l	list	显示指定行号或者指定函数附近的源代码
b	break	在指定行号或者指定函数开头处设置断点。如b main
r	run	运行程序，直到程序结束或者遇到断点而停下
c	continue	在程序中断后继续执行程序，直到程序结束或者遇到断点而停下。注意在程序开始执行前只能用r，不能用c
n	next	执行一条语句。如果有函数调用，则把它作为一个整体
s	step	执行一条语句。如果有函数调用，则进入函数内部
u	until	执行到指定行号或者指定函数的开头
p	print	显示变量或表达式的值
disp	display	把一个表达式设置为display，当程序每次停下来时都会显示其值
cl	clear	取消断点，和b的格式相同。如果该位置有多个断点，将同时取消
i	info	显示各种信息。如i b显示所有断点，i disp显示display，而i lo显示所有局部变量

如果对上述解释有疑问，可输入help以获得详尽的帮助信息。

C/C++语言测试

Q1: Identify and correct the errors in each of the following statements: (10)

(a) (5) char *str = f"happy"g; str[1] = "e"; str[2] = "l";

(b) (5) mul (double x, y){ double x, y; return x * y; }

Q2: Recursive (30分)

(a) Consider the following recursive function. Rewrite it using iterative (nonrecursive) approach.

```
int sum(int n) {
if (n < 1) return 1;
return sum(n - 1) * (n - 1) + n;
}
```

(b) Consider the following function sum. Rewrite it as a recursive function.

```
int sum (int n) {
int i, sum = 1;
for (i = 1; i <= n; i++) sum *= i + 1;
return sum;
}
```

Q3: Write the result after executing the following program. (10分)

```
int func (int a, int b)
 {
     b *= 2;
     printf("a = %d, b = %d.\n", a, b);
     return --a * (b - 8);
}
int sub (int *a, int *b) {
     *a -= 6;
     printf("a = %d, b = %d.\n", *a, *b);
     return *a-- * ++*b;
}
```

```
int main() {
     int x = 6, y = 8;
     x = func(y, y);
     printf("x = %d, y = %d.\n", x, y);
     x = sub(&y, &y);
     printf("x = %d, y = %d.\n", x, y);
     return  0;
}
```

Q4:Write a function that passes a string and reverse it (20)

Q5: Write the following function using the following structure for a point. (30)

```
typedef struct {
double x, y;
} Point;
```

(a) (12) A function that passes two points and returns the distance of them.

(b) (18) A function that passes three points and return the area formed by these three points.

答案

Q1:

(a) Error: The extra braces are not required for string assignment. The character of a character pointer cannot be changed.

The character should be enclosed in single quotation marks.

Correction: char str[6] = "happy"; str[1] = 'e'; str[2] = 'l';

(b) Error: The function should have a return data type. The parameters should have data types. The parameters should

not redeclared in the function.

Correction: double mul (double x, double y) { return x * y; }

Q2:

(a)
```
int sum (int n) {
    int i, sum = 1;
    for (i = 1; i <= n; i++) sum = sum * (i - 1) + i;
    return sum;
}
```

(b)
```
int sum(int n) {
    if (n < 1) return 1;
    return sum(n - 1) * (n + 1);
}
```

Q3

- Ans:

a = 8, b = 16.

x = 56, y = 8.

a = 2, b = 2.

x = 9, y = 3

Q4

```
#define SIZE 81
void reverse(char s[]) {
int len = strlen(s), i;
char r[SIZE];
for (i = 0; i < len; i++) r[i] = s[len - i - 1];
for (i = 0; i < len; i++) s[i] = r[i];
}
```

Q5

(a) double distance(Point p1, Point p2) {

```
        return sqrt((p1.x - p2.x) * (p1.x - p2.x) + (p1.y - p2.y) * (p1.y - p2.y));
```

}

(b) double area(Point p1, Point p2, Point p3) {

```
    double a = distance(p1, p2), b = distance(p2, p3), c = distance(p3, p1);
    double s = (a + b + c)/2;
    return sqrt(s * (s - a) * (s - b) * (s - c));
```

}

测试

- 90分？ 5
- 高于80分？ 10
- 高于60分？20

Typical programming contest problems

- Usually consists of
  1. Problem description
  2. Input description
  3. Output description
  4. Example input/output
  5. A time limit in seconds
  6. A memory limit in bytes

- You are asked to write a program that solves the problem for all valid inputs

- The program must not exceed time or memory limits

The problems

Problem description

Write a program that multiplies pairs of integers.

Input description

Input starts with one line containing an integer T, where $1 \leq T \leq 100$, denoting the number of test cases. Then T lines follow, each containing a test case. Each test case consists of two integers A, B, where $-2^{20} \leq$

A, B $\leq 2^{20}$, separated by a single space.

Output description

For each test case, output one line containing the value of A $\times$ B.

Example problem

Sample input	Sample output
4 3 4 13 0 1 8 100 100	 12 0 8 10000

Example solution

```
#include <iostream>
using namespace std;

int main() {
	int T;
	cin >> T;

	for (int t = 0; t < T; t++) {

		int A, B;
		cin >> A >> B;

	cout << A * B << endl;
	}

return 0;
}
```

Example solution

```
#include <iostream>
using namespace std;

int main() {
	int T;
	cin >> T;

	for (int t = 0; t < T; t++) {

		int A, B;
		cin >> A >> B;

	cout << A * B << endl;
	}

return 0;
}
```

- Is this solution correct?

Example solution

```
#include <iostream>
using namespace std;

int main() {
	int T;
	cin >> T;

	for (int t = 0; t < T; t++) {

		int A, B;
		cin >> A >> B;

	cout << A * B << endl;
	}

return 0;
}
```

- Is this solution correct?
- What if $A = B = 2^{20}$?

Example solution

```
#include <iostream>
using namespace std;

int main() {
	int T;
	cin >> T;

	for (int t = 0; t < T; t++) {

		int A, B;
		cin >> A >> B;

	cout << A * B << endl;
	}

return 0;
}
```

- Is this solution correct?
- What if A = B = 2^{20}? The output is 0...

Example solution

```
#include <iostream>
using namespace std;

int main() {
	int T;
	cin >> T;

	 for (int t = 0; t < T; t++) {

		int A, B;
		cin >> A >> B;

	 cout << A * B << endl;
	 }

return 0;
}
```

- Is this solution correct? No!
- What if A = B = 2^{20}? The output is 0...

Example solution

- When A = B = 2^{20}, the answer should be 2^{40}

Example solution

- When A = B = 2^{20}, the answer should be 2^{40}
- Too big to fit in a 32-bit integer, so it overflows

Example solution

- When A = B = 2^{20}, the answer should be 2^{40}
- Too big to fit in a 32-bit integer, so it overflows
- Using 64-bit integers should be enough

Example solution

```
#include <iostream>
using namespace std;

int main() {
int T;
cin >> T;

    for (int t = 0; t < T; t++) {

        long long A, B;
        cin >> A >> B;

    cout << A * B << endl;
    }

return 0;
}
```

Example solution

```
#include <iostream>
using namespace std;

int main() {
      int T;
      cin >> T;

       for (int t = 0; t < T; t++) {

            long long A, B;
            cin >> A >> B;

       cout << A * B << endl;
       }

return 0;
}
```

- Is this solution correct?

Example solution

```
#include <iostream>
using namespace std;

int main() {
      int T;
      cin >> T;

       for (int t = 0; t < T; t++) {

            long long A, B;
            cin >> A >> B;

       cout << A * B << endl;
       }

return 0;
}
```

- Is this solution correct? Yes!

What is ACM ICPC

- ACM: Association for Computing Machinery
  - http://www.acm.org/
  - the world's largest educational and scientific computing society
- ACM ICPC
  - ACM International Collegiate Programming Contest
  - http://en.wikipedia.org/wiki/ACM_International_Collegiate_Programming_Contest

ACM ICPC

- ICPC is a competition among teams of students representing institutions of higher education.
  - Teams compete in Regional Contests, from which top scoring teams advance to the ACM-ICPC World Finals.
- Each team has three students, sharing one computer, given a number of programming problems
  - Coordination and teamwork are essential

Programming Languages and Judge

- You can use:
  - C, C++, Java and others such as: Python, C#
- Online Judge
  - Feedback :
    - Accepted (AC) – congratulations!
    - Presentation Error (PE) – Your program outputs are correct, but are not presented in the specified format. Check for spaces, left/right justification, line feeds, etc.
    - Wrong Answer (WA) – Your program returned an incorrect answer to one or more of the judge's secret test cases
    - Compile Error (CE) – The judge's compiler cannot compile your source code
    - Runtime Error (RE) – Your program failed during execution due to a segmentation fault, floating point exception, or others.
    - Time Limit Exceeded (TL) – Your program took too much time on at least one of the test cases. Try to improve the efficiency of your solution!
    - Memory Limit Exceeded (ML) – Your program tried to use more memory than the judge's settings.

Available Online Judges

- Famous online judges
  - Valladolid OJ (http://acm.uva.es/p)
  - Ural OJ (http://acm.timus.ru)
  - Saratov OJ (http://acm.sgu.ru)
  - ZJU OJ (http://acm.zju.edu.cn)
  - ZJUT OJ (http://acm.zjut.edu.cn)
  - Official ACM Live Archive (http://cii-judge.baylor.edu/)
  - Peking University Online Judge (http://acm.pku.edu.cn/JudgeOnline/)
  - Programming Challenges (http://www.programming-challenges.com)

To get ready some suggested Books

- Art of Programming Contest (free online)
  - http://online-judge.uva.es/p/Art_of_Programming_Contest_SE_for_uva.pdf

- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, Introduction to Algorithms, 2nd Edition, The MIT Press, 2001.

- Robert Sedgewick, Bundle of Algorithms in Java, Third Edition (Parts 1-5), 3rd Edition, Addison-Wesley Professional, 2003. (There is also a C++ version).

- Donald E. Knuth,The Art of Computer Programming, Volume 1, 2, 3.

Subjects

- You should be familiar with:
  - Data Structure
  - Strings and Sorting
  - High precision arithmetic
  - Combinatorics and number theory
  - Divide and conquer & backtracking
  - Dynamic programming
  - Computational geometry
  - Scientific computing

Standard in ACM contest

- Input/Output
  - each program must read the test data from the standard input and print the results to the standard output
    - For C language, use ***scanf***() and ***printf***()
    - For C++, use ***cin*** and ***cout***
      - ***scanf()* and *printf()* are also supported**
    - For Java, refer to http://www.programming-challenges.com/pg.php?page=javainfo
  - Programs are not allowed to open files or to execute certain system calls

Not nice for debugging

```
#include <stdio.h>
int main ()
{
  freopen(“FILE_NAME_FOR_INPUT”,”r”,stdin);
  freopen(“FILE_NAME_FOR OUTPUT”,”w”,stdout);
  Rest of the codes…
  return 0;
}
```

While sending your code to online judges, remember to remove the two lines with freopen.

Things to avoid

- Avoid the usage of the ++ or -- operators inside expressions or function calls
- Avoid expressions of the form *p++
- Avoid pointer arithmetic. Instead of (p+5) use p[5].
- Never code like : return (x*y)+Func(t)/(1-s);
  - but like :
    - temp = func(t);
    - RetVal = (x*y) + temp/(1-s);
    - return RetVal;

Things to avoid

- Naming
  - Don’t use small and similar names for your variables. Use descriptive names.

  - Don’t use names like {i,j,k} for loop control variables. Use {I,K,M}.
  - It is very easy to mistake a j for an i when you read code or “copy, paste & change” code,

Nature of problems to solve

- Data structure problems
- Algorithms
  - To solve real problems efficiently
- Categories:
  - Sorting
  - Searching
  - Graph algorithms
  - Scientific computing: matrix, number-theoretic, computational geometry, etc.

  -etc
- Mathematics
  - Everything finally goes back to mathematics!
  - Number theory
  - Geometry
  - Combinatorics
  - Graph theory
  - …

Good Team

- Knowing your strength and weaknesses
- Knowledge of standard algorithms and the ability to find an appropriate algorithm for every problem in the set;
- Ability to code an algorithm into a working program;
- Having a strategy of cooperation with your teammates

Tips & tricks

- Brute force when you can, Brute force algorithm tends to be the easiest to implement.
- KISS: Simple is smart! (Keep It Simple, Stupid !!! / Keep It Short & Simple).
- Hint: focus on limits (specified in problem statement).
- Waste memory when it makes your life easier (trade memory space for speed).
- Don't delete your extra debugging output, comment it out.
- Optimize progressively, and only as much as needed.
- Keep all working versions!

Tips & tricks

- Code to debug:
  - a. white space is good,
  - b. use meaningful variable names,
  - c. don't reuse variables, (we are not doing software engineering here)
  - d. stepwise refinement,
  - e. Comment before code.
- Avoid pointers if you can.
- Avoid dynamic memory like the plague: statically allocate everything.
- Try not to use floating point; if you have to, put tolerances in everywhere (never test equality)

Problem example

Source: http://acm.zju.edu.cn/onlinejudge/showProblem.do?problemCode=1001

- Time Limit:1 second Memory Limit:32768 KB
- **Description:** Calculate a + b
- **Input:**
  - The input will consist of a series of pairs of integers a and b, separated by a space, one pair of integers per line.
- **Output:**
  - For each pair of input integers a and b you should output the sum of a and b in one line, and with one line of output for each line in input.
- **Sample Input:**

  1 5
- **Sample Output:**

  6

Solution

```c
/* C code */
#include “stdio.h”

int main()
{
    int a, b;
    while (scanf(“%d %d”, &a, &b)
!= EOF) {
        printf(“%d\n”, a+b);
    }
    return 0;
}
```

```java
/* Java code */
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        while (in.hasNextInt()) {
            int a = in.nextInt();
            int b = in.nextInt();
            System.out.println(a + b);
        }
    }
}
```

Set of problems that you can train on

- Source: Fundamental Problems http://acm.zjut.edu.cn

1167
1166
1174
1175
1176
1177
1178
1179

1181
1185
1190
1191
1187
1204
1208
1205
1044

- Source: Fundamental Problems http://acm.uva.es

http://acm.uva.es/p/v1/100.html
http://acm.uva.es/p/v101/10189.html
http://acm.uva.es/p/v101/10137.html
http://acm.uva.es/p/v7/706.html
http://acm.uva.es/p/v102/10267.html
http://acm.uva.es/p/v100/10033.html
http://acm.uva.es/p/v101/10196.html
http://acm.uva.es/p/v101/10142.html

评分标准

- 作业: 85%
- 比赛: 15%